Intelligent Control for Fixed-Wing eVTOL Aircraft

Thesis by

Xichen Shi

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy

Caltech

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2021
Defended January 12, 2021

ABSTRACT

Urban Air Mobility (UAM) holds promise for personal air transportation by deploying "flying cars" over cities. As such, fixed-wing electric vertical take-off and landing (eVTOL) aircraft has gained popularity as they can swiftly traverse cluttered areas, while also efficiently covering longer distances. These modes of operation call for an enhanced level of precision, safety, and intelligence for flight control. The hybrid nature of these aircraft poses a unique challenge that stems from complex aerodynamic interactions between wings, rotors, and the environment. Thus accurate estimation of external forces is indispensable for a high performance flight. However, traditional methods that stitch together different control schemes often fall short during hybrid flight modes. On the other hand, learning-based approaches circumvent modeling complexities, but they often lack theoretical guarantees for stability.

In the first part of this thesis, we study the theoretical benefits of these fixed-wing eVTOL aircraft, followed by the derivation of a novel unified control framework. It consists of nonlinear position and attitude controllers using forces and moments as inputs; and control allocation modules that determine desired attitudes and thruster signals. Next, we present a composite adaptation scheme for linear-in-parameter (LiP) dynamics models, which provides accurate realtime estimation for wing and rotor forces based on measurements from a three-dimensional airflow sensor. Then, we introduce a design method to optimize multirotor configuration that ensures a property of robustness against rotor failures.

In the second part of the thesis, we use deep neural networks (DNN) to learn part of unmodeled dynamics of the flight vehicles. Spectral normalization that regulates the Lipschitz constants of the neural network is applied for better generalization outside the training domain. The resultant network is utilized in a nonlinear feedback controller with a contraction mapping update, solving the nonaffine-in-control issue that arises. Next, we formulate general methods for designing and training DNN-based dynamics, controller, and observer. The general framework can theoretically handle any nonlinear dynamics with prior knowledge of its structure. Finally, we establish a delay compensation technique that transforms nominal controllers for an undelayed system into a sample-based predictive controller with numerical integration. The proposed method handles both first-order and transport delays in actuators and balances between numerical accuracy and computational efficiency to guarantee stability under strict hardware limitations.

PUBLISHED CONTENT AND CONTRIBUTIONS

[1] X. Shi, M. O'Connell, and S.-J. Chung, "Numerical predictive control for delay compensation," *arXiv preprint arXiv:2009.14450*, 2020,
X.S. proposed the concept of the project, formulated theoretical proofs, analyzed simulation data, and participated in the writing of the manuscript.

[2] X. Shi, P. Spieler, E. Tang, E.-S. Lupu, P. Tokumaru, and S.-J. Chung, "Adaptive nonlinear control of fixed-wing VTOL with airflow vector sensing," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 5321–5327. DOI: `10.1109/ICRA40945.2020.9197344`,
X.S. proposed the control algorithm, formulated theoretical proofs, and participated in the writing of the manuscript.

[3] K. Kim, S. Rahili, X. Shi, S.-J. Chung, and M. Gharib, "Controllability and design of unmanned multirotor aircraft robust to rotor failure," in *AIAA Scitech Forum*. 2019. DOI: `10.2514/6.2019-1787`. [Online]. Available: `https://arc.aiaa.org/doi/abs/10.2514/6.2019-1787`,
X.S. implemented the algorithm on hardware, and conducted experiments.

[4] G. Shi, X. Shi, M. O'Connell, R. Yu, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S.-J. Chung, "Neural lander: Stable drone landing control using learned dynamics," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 9784–9790. DOI: `10.1109/ICRA.2019.8794351`,
X.S. designed the controller based on contraction mapping, formulated theoretical proofs, conducted experiments, and participated in the writing of the manuscript.

[5] X. Shi, K. Kim, S. Rahili, and S.-J. Chung, "Nonlinear control of autonomous flying cars with wings and distributed electric propulsion," in *IEEE Conference on Decision and Control (CDC)*, 2018, pp. 5326–5333. DOI: `10.1109/CDC.2018.8619578`,
X.S. proposed the architecture, formulated and implemented the controller, conducted experiments, and participated in the writing of the manuscript.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

NOMENCLATURE

Abbreviations

AFA	Autonomous Flying Ambulance
AoA	Angle-of-attack
AoS	Angle-of-sideslip
BPTT	Back-Propagation-Through-Time
CLF	Control Lyapunov Function
CoM	Center-of-mass
DEP	Distributed Electric Propulsion
DNN	Deep Neural Network
DoF	Degree-of-freedom
eVTOL	Electric VTOL
FDE	Functional differential equations
FOPDT	First-order plus dead time
LiP	Linear-in-parameter
LQR	Linear Quadratic Regulator
ODE	Ordinary differential equation
PD	Proportional-Derivative
PDE	Partial differential equation
PI-PD	Proportional-Integral-Proportional-Derivative
PID	Proportional-Integral-Derivative
ReLU	Rectified Linear Unit
RK	Runge-Kutta
RL	Reinforcement Learning
SBC	Single board computer
SISO	Single-Input Single-Output

SOPDT	Second-order plus dead time
UAM	Urban Air Mobility
VTOL	Vertical take-off and landing

Mathematical Notations

$(\cdot)^+$	Right pseudoinverse
$[\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n]$	Stack of vectors by row
$[\boldsymbol{x}_1; \cdots; \boldsymbol{x}_n]$	Stack of vectors by column
$\dot{(\cdot)}$	Time derivative
$\lambda_{\max}(\cdot)$	Maximum eigenvalue of a matrix
$\lambda_{\min}(\cdot)$	Minimum eigenvalue of a matrix
$\|\cdot\|$	2-norm of vector
$\mathbb{R}^n$	Set of n-dimensional vectors of real numbers
$\mathbb{R}$	Set of real numbers

Symbols

α, β	Angle-of-attack and angle-of-sideslip
$\boldsymbol{\omega}$	Angular velocity of vehicle
$\boldsymbol{B}_A$	Actuation matrix for aerodynamic control surfaces
$\boldsymbol{\Phi}(\cdot)$	Basis function for LiP model
$\boldsymbol{B}_T$	Actuation matrix for thrusters
$\boldsymbol{B}_T'$	Optimized pseudoinverse of $\boldsymbol{B}_T$
$\boldsymbol{I}$	Identity matrix of appropriate size
$\boldsymbol{q}$	Attitude quaternion of vehicle
$\boldsymbol{r}_j$	Vector from vehicle CoM to j-th rotor axle
$\bar{c}$	Reference chord length for nondimensionalization
C_L, C_D, C_Y	Lift, drag and side force coefficients
C_{mx}, C_{my}, C_{mz}	Aerodynamic moment coefficients in x, y, z axis
$\mathcal{M}$	Attainable moment space

Δ	Transport delay
Δ_c	Computation delay
Δ_s	System delay
$\boldsymbol{f}_A$	Aerodynamic force vector
$\boldsymbol{f}_b$	Total force on vehicle in body frame
$\boldsymbol{f}_T$	Combined thruster force on vehicle in body frame
$\boldsymbol{g}$	Acceleration of gravity vector in inertial frame
$\hat{\boldsymbol{k}}_f$	Favorable thrust force direction unit vector
$\hat{\boldsymbol{z}}_j$	Unit vector for j-th rotor direction
$\boldsymbol{J}$	Inertia matrix of vehicle
$\boldsymbol{\tau}_A$	Aerodynamic moment vector
$\boldsymbol{M}$	Masking matrix for available thrust force
$\boldsymbol{\tau}_b$	Total moment on vehicle in body frame
$\boldsymbol{\tau}_T$	Combined thruster moment on vehicle in body frame
$\boldsymbol{p}$	Inertial position of vehicle
$\boldsymbol{R}$	Rotation matrix from body to inertial frame
$S_{\mathbf{ref}}$	Reference area for nondimensionalization
τ	Rotor axial torque
$\hat{\boldsymbol{\theta}}$	Parameter vector for LiP model
T_s	Sampling period for discrete control update
$\mathcal{U}_T$	Attainable thruster control space
$\boldsymbol{u}_T$	Thruster control input vector
$\boldsymbol{v}$	Inertial velocity of vehicle
$\boldsymbol{v}_i$	Incident wind velocity
V_∞	Freestream windspeed
$\boldsymbol{v}_w$	Wind velocity
$\mathcal{W}_T$	Attainable thruster wrench space

$\boldsymbol{w}_T$	Thruster wrench vector
$\bar{\mathcal{W}}_T$	Fitted thruster wrench space
A	Area of rotor disk
A_∞	Area of stream tube far downstream
C_T, C_Q	Rotor thrust and torque coefficients
C_{P_0}	Profile power coefficient
d	Diameter of rotor
F_S	Rotor side force
L, D, Y	Aerodynamic lift, drag and side force
n	Rotor rotation speed
N_v, N_c, N_T	Number of vertical, cruise and total thrusters
P_0	Profile power
P_h	Induced power at hover
P_i	Induced power
P_R	Total required power
T	Rotor thrust force
V_c	Axial freestream windspeed with respect to rotor disk
v_i	Induced speed at rotor plane
W	Weight of vehicle
w	Induced speed far downstream

Chapter 1

INTRODUCTION

Recent interest in using electric vertical take-off and landing (eVTOL) aircraft in urban aerial transportation for cargo and personnel promises great improvement over existing transportation infrastructures. In cases such as the San Francisco Bay area rush hours, a one-way trip in those conceptualized eVTOLs would take as short as 15 minutes compared to the 1.5 hours car ride shown in Figure 1.1 [1]. To achieve these ambitious goals, the proposed vehicle systems need to: (1) operate in cluttered urban environments for fast take-off and landing; (2) possess reasonable range and endurance for inter/intra-city flights; (3) admit a superior safe and intelligent control system when interacting with complex environments or other agents.

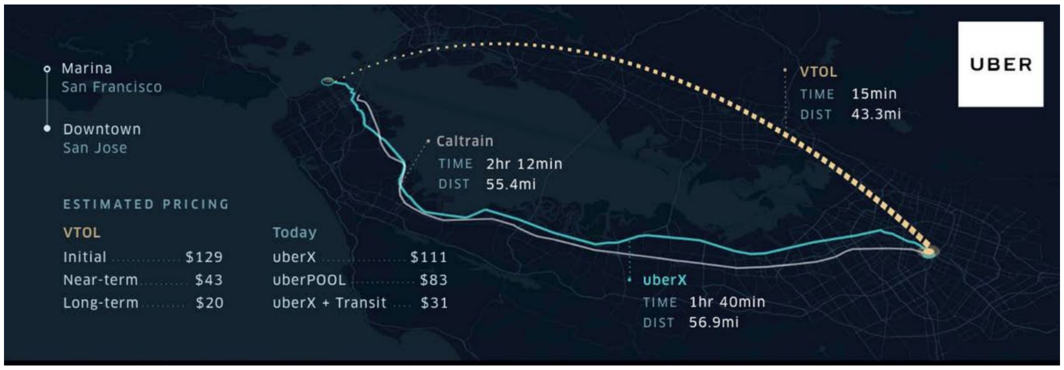


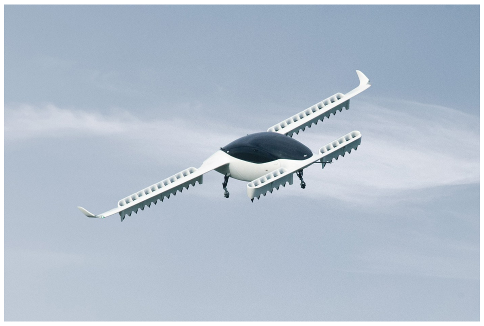
Figure 1.1: Visualization of travel from San Jose to San Francisco by Uber Elevate [1]. It predicts that eVTOL would drastically reduce travel time while being reasonably priced.

1.1 Fixed-wing VTOL Aircraft as Urban Air Mobility

VTOL aircraft have been an area of intense research for most of the past century. It has been conceived by inventors well before the first powered flights. The operational simplicity associated with not requiring a runway and being able to hover in place often outweighs the negative aspects of its design complexity. Different technologies throughout the years have given birth to a variety of VTOL-capable aircraft that spiked interests in military, commercial, and research domains. In recent years, improvements in battery technology, computing power, and sensor availability have spurred the development of electric multirotors. The canonical form of the multirotor uses an even number of symmetric, coplanar rotors to generate vertical thrust. The simplicity in the design, construction, and control of these platforms has made them very popular as a cheap and robust aerial platform. However, a standard multirotor lacks the efficiency for long range flight.

(a) WISK Cora

(b) Kitty Hawk Heaviside


(c) Lilium Jet

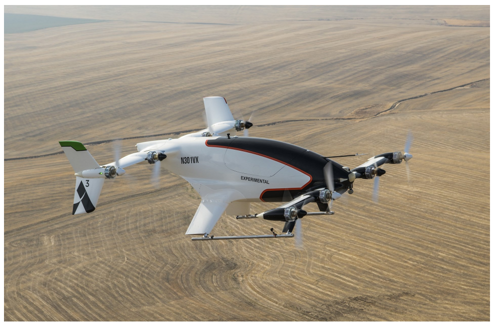

(d) Airbus Vahana

Figure 1.2: Examples of commercial fixed-wing VTOLs for UAM applications.

UAM calls for vehicle designs that can take-off and land in a cluttered urban environment while efficiently flying a good range over metropolitan areas [1]. Married with the trend in electric drones, the class of hybrid VTOL aircraft with both fixed wing surfaces and distributed electric propulsion (DEP) systems haven risen in popularity [2]. They use lifting surfaces to enable longer range and endurance flights and keep VTOL capabilities, eliminating the need for a substantial runway. Some of the popular types of modern fixed-wing VTOL aircraft include: (1) copter-plane: a direct hybrid between planes and multirotors; (2) tilt-rotor/vectored-thrust: thrust can be diverted in different directions; and (3) tail-sitter: thruster is situated in the back, and the aircraft takes off upright and later transitions to horizontal flight. Commercial solutions for UAMs follow this fixed-wing VTOL philosophy as well, with some prominent examples shown in Figure 1.2, which all adopt DEP and fixed-wing combination. At Caltech, we have developed similar technologies named Autonomous Flying Ambulance (AFA) as described in greater details in Appendix B.

1.1.1 Existing Control Methods for the Fixed-wing VTOL Hybrids

Although different in geometry, the underlying control logic is similar for all types of fixed-wing VTOLs. Most controllers designed for such crafts rely on two separate schemes, one for VTOL stage and one for cruise flight stage. A transition strategy is designed to switch between the two. Because of the hybrid nature during this transition period, complex interactions between propellers and wings pose challenges for accurate and safe flight maneuvers. Early works achieve this transition by overlapping the operation envelopes of the two controllers. In [3], [4], reference commands were sent to the VTOL controller such that the vehicle would either reach a high-speed or a low pitch angle state triggering the fixed-wing controller to become active. A common technique for tilt-rotor transition is to vary tilt angles following a monotonic schedule, during which the controller stabilizes the craft [5], [6]. However, little attention was given to aerodynamic and flight-dynamic modeling in this scenario. More recent methods utilize numerical optimization to solve for a trajectory based on accurate vehicle dynamics. In [7], [8], transition trajectories are solved offline and feedback tracking controllers are deployed for online exectution. To further improve the performance, online optimization-based controllers with global aerodynamic model for tail-sitters are proposed in [9], [10]. Such controllers can give solutions between any global states, as long as the onboard computer can solve the problem in real time.

Position trajectory tracking for aerial vehicles relies on their ability to generate forces in desired directions. Although the control of all aerial vehicles often depends on timescale separation property in flight dynamics [11], [12], the key difference between vehicle types results from what forces are considered significant during control design. In both fixed-wing and VTOL multirotors, the collective force from all thrusters is one-dimensional. Specifically, a multirotor points in the direction of the total commanded force whereas a fixed-wing aircraft merely uses thrust to cancel drag force and relies on lift force for maneuver. For a fixed-wing VTOL, the controller should realize its potentials as much as possible to extend the flight envelope. The problem becomes more challenging when the vehicle can produce thrust in multiple directions [13].

As far as multi-dimensional control forces are concerned, fully-actuated multirotors [14]–[18] have garnered much attention in recent years. Because of the added dimension in force space, vehicles can generate forces in any direction without changing attitude [17], [18]. If this is not achieved, then the desired attitude is found via the closest projection of the desired force onto force space [15], [16]. However, all such work does not consider aerodynamic forces and only focuses on the concept of full-actuation which attempts to marginalize attitude determination [17]. Aircraft with thrust vectoring have been around for decades, but they have only found applications in military or been used for moment generation during post-stall maneuvers [19].

1.1.2 Fault Tolerance for Multirotor Distributed Electric Propulsion

For UAM applications, safety is of the utmost importance for these aircraft and it cannot be sacrificed even when they encounter failure of some components and need to operate under abnormal conditions. Having more than six rotors on such aircraft allows them to be robust against rotor failure up to a certain degree. The key question is how to optimize the design of the multirotors such that they remain controllable when different combinations of rotors fail.

A survey on fault-tolerant techniques for multirotor aircraft can be found in [20]. One way of handling the failure is to develop separate controllers to be used when rotors fail [21], [22]. With this approach, control authority in certain axis is sacrificed in order to retain control in the rest. For example, it is well-known that a quadrotor cannot retain full controllability on its position and attitude when it loses one or more rotors as it becomes an underactuated system. In this case, controllers may decide to give up yaw authority [23] or to control only the reduced attitude [24]. In the literature, hexacopters have been a popular platform to study controllability because they seem to be robust to the failure of up to two rotors and be able to fly like quadrotors. Interestingly, however, researchers have shown that symmetric and collinear hexacopters subject to a single rotor failure are not fully controllable [25], and controllers with limited attitude controllability have been developed [22], [26].

As such, carefully designed control laws can handle certain rotor failure cases that they are intended for, but it is difficult to come up with a set of control laws that can cope with all possible cases of rotor failures. Furthermore, the resulting behavior and inherent controllability of multirotor aircraft are limited by their own design choices that govern the sets of admissible forces and moments from the aircraft. Therefore, another way of addressing rotor failure is to optimize the design of aircraft to be robust against rotor failure and enhance survivability of the aircraft. In such spirit, a carefully designed aircraft with only one rotor was proposed in [27], which can perform limited maneuvers and avoid unwanted crashes. Hexacopters with tilted rotors were also proposed in order to retain controllability and to improve disturbance rejection after rotor failure [15], [28], [29]. Although some asymmetric multirotor aircraft have been reported in the literature [30], most multirotor aircraft resort to symmetric configurations [31]–[34], limiting exploration of the full design space as well as their control performance.

1.2 Learning and Adaptive Methods in Flight Control

As noted in the core needs of UAM, control of the craft with high accuracy is of primary concern for efficiency, safety, and comfort of flights. Types of fixed-wing VTOL vehicles inherit modeling complexities from their hybrid nature. Further adding to the challenge is its requirement of frequent interaction with cluttered and uncertain urban environments. Therefore, the need for learning from past interactions and rapid adaptation to unseen scenarios are essential in designing controllers for next generation UAM aircraft.

For a given dynamical system, complexity and uncertainty can arise either from its inherent property or the changing environment. Thus model accuracy is often key in designing a high-performance and robust control system. If the model structure is known, conventional system identification techniques can be used to resolve the parameters of the model. When the system becomes too complex to model analytically, modern machine learning research conscripts data-driven and neural network approaches that often result in bleeding-edge performance given enough samples, proper tuning, and adequate time for training. However, the harsh requirement on a learning-based flight control system calls for both representation power and fast execution simultaneously. Thus it is natural to seek wisdom from the classic field of adaptive control, where successes have been seen using simple linear-in-parameter (LiP) models with provably robust control designs [35], [36]. On the other hand, the field of machine learning has made its own progress toward fast online paradigms, with the rising interest in few-shot learning [37], continual learning [38], [39], and meta learning [40], [41].

There has been some interest in designing a unified feedback controller for fixed-wing VTOL, such as in [42], [43], and our prior work [44], where the controller is flight-mode agnostic through estimating combined forces on the vehicles. The success of these controllers relies primarily on the accurate prediction of aerodynamic forces, which in turn requires high-fidelity models and accurate sensor feedback for states relevant to such forces. Surprisingly, there has been little work on these areas; complex aerodynamic interactions between wing and rotors that are crucial to fixed-wing VTOL transition are often ignored. Although efforts have been made to estimate aerodynamic states such as angle-of-attack and sideslip angle [45], [46], no recent work has been using such information directly in a feedback control manner. On the other hand, adaptive flight control has seen much progress over the years, where the vehicle model is adapted via either aerodynamic coefficients [47], [48] or neural network parameters [49], [50]. Recently, [51] used incremental nonlinear dynamic inversion to estimate the external force through filtered accelerometer measurement, and then apply direct force cancellation in the controller. [52] assumed a diagonal rotor drag model and proved the differential flatness of the system for cancellation, and [44] used a nonlinear aerodynamic model for force prediction. When a LiP model is available, adaptive control theories can be applied for controller synthesis. This does not limit the model to only physics-based parameterizations, and a neural network basis can be used [36], [53]. Adaptive controllers with LiP models have been applied to multirotors for wind disturbance rejection in [54].

One particularly interesting scenario where learning methods can thrive is flying close to ground or objects. Compensating for ground effect is a long-standing problem in the aerial robotics community. Prior work has largely focused on mathematical modeling [55] as part of system identification. These models are later used to approximate aerodynamics forces during flights close to the ground and combined with controller design for feed-forward cancellation [56]. However, existing theoretical ground effect models are derived based on steady-flow conditions, whereas most practical cases exhibit unsteady flow. Alternative approaches, such as integral or adaptive control methods, often suffer from slow response and delayed feedback. [57] employs Bayesian Optimization for open-air control, but not for take-off/landing. Given these limitations, the precision of existing fully automated systems for VTOLs are still insufficient for landing and take-off, thereby necessitating the guidance of a human UAV operator during those phases.

Using DNNs to approximate high-order non-stationary dynamics has recently received considerable attention. For example, [58], [59] use DNNs to improve system identification of helicopter aerodynamics, but not for controller design. Other approaches aim to generate reference inputs or trajectories from DNNs [60]–[63]. However, these approaches can lead to challenging optimization problems [60], or heavily rely on well-designed closed-loop controllers and require a large number of labeled training data [61]–[63]. A more classical approach of using DNNs is direct inverse control [64]–[66], but the non-parametric nature of a DNN controller also makes it challenging to guarantee stability and robustness to noise. [67] proposes a provably stable model-based reinforcement learning (RL) method based on Lyapunov analysis, but it requires a potentially expensive discretization step and relies on the native Lipschitz constant of the DNN.

When adapting to complex system dynamics or fast changing environments, one would expect the network approximator to have enough representation power, which makes a DNN an desirable candidate. However, there are several issues associated with using a deep network for adaptive control purpose. First, training a DNN often requires back propagation, easily leading to a computation bottleneck for realtime control on small drones. Second, continual online training may incur catastrophic inference where previously learned knowledge is forgotten unintentionally. Third, a vanilla network for a regression problem often does not have guarantees on desirable properties for control design, such as output boundedness and Lipschitz continuity. Fortunately, advances have been made in circumventing these issues. Training a deep network by updating the last layer's weight more frequently than the rest of the network is proven to work for approximating Q-function in RL [68], [69]. This enables the possibility of fast adaptation without incurring a high computation burden. Spectral normalization on all the network weights can constrain the Lipschitz constant [70].

1.3 Computation Cost and Actuation Delay

State or control delays occur naturally in a variety of physical and cyber-physical systems. For high accuracy tracking of time trajectory, such delays can become crucial in further improving control performance. Electric VTOL aircraft typically uses electric motors to drive rotor systems, where either rotor speed or pitch is controlled via input signals to achieve desired thrust force output. In either case, delays in actuation would exist due to the electric and mechanical nature of the drive mechanism. This can be especially detrimental when delay time-scale is on the same order as vehicle inertial dynamics. On the other hand, with the sensitivity to total weight of flying vehicles, onboard computation is always limited compared to ground platforms. Thus digital delays due to discrete computing architecture will pose further hurdles for high performing flight control.

Time-delayed dynamics have been an active area of research since its introduction in 1946 [71]–[74], and it is seeing continued interest with the popularization of vast computer networks and internet-of-things (IoT) accompanied by substantial communication lags [75], [76]. Delay compensation techniques have also been widely used in control of power electronics [77], [78] and reinforcement learning settings [79].

For linear systems, delay for unstable processes is often modeled as first or second order plus dead time (FOPDT or SOPDT). Classical linear feedback control can be applied and closed-loop system behavior is analyzed with transfer function approaches. It was shown that properly designed proportional-integral-derivative (PID) controllers can act as a delay compensator [80]. Other popular techniques include relay-based identification [81] and proportional-integral-proportional-derivative (PI-PD) control [82]. For nonlinear systems, the usual consensus on the challenge of continuous delays is that the state space becomes infinite dimensional. Thus, instead of being described by ordinary differential equations (ODEs), these systems need to be modeled as functional differential equations (FDEs) or transport partial differential equations (PDEs) [73], [74]. Accordingly, their analysis requires additional mathematical tools such as Lyapunov-Krasovskii functionals [83], [84]. A prominent class of delay compensation methods rely on state predictions of some kind. This idea was first proposed as the Smith-predictor [72], and has been expanded to handle unstable processes [85], increase robustness against uncertainties [86], or adapt to varying delays [87]. In theory, predictor-based methods can handle arbitrarily large delays for forward complete and strict-feedforward systems [88].

The FDE or PDE modeling approach has the underlying assumption that input signal is continuous in time. For control systems run on digital computers in practice, this assumption is only true when the evaluation time of the control algorithm is much smaller compared to the transport delay of the signal. The statement is largely valid for cases considered in a network control system. However, certain real-time control applications with limited computation capacity tend to violate the continuity assumption, since the controller calculation time runs at similar timescales as other delays. We take interest in the following aspects of such systems: first, the control input often corresponds to commands on actuators, which admit additional layers of control that act as a dynamic delay; second, computation time of the controller is non-negligible and is affected by the complexity of the control algorithm; last but not least, the discrete sampling for the control implementation poses restrictions on the stability for the continuous dynamics.

1.4 Thesis Organization

This thesis covers two parts of materials. In the first part, aspects of fixed-wing VTOL aircraft are discussed in Chapters 2 to 5, where we cover the development process in conceptual design, control system, and fault tolerance. The second part in Chapters 6 and 7 focuses on crucial challenges in controlling such vehicles: how to model the complex aerodynamic interactions with data-driven methods; how to have fast online algorithms that can adapt to previously unseen situations; and how to account for delays in actuation and computation on minimal hardware required for safety.

In Chapter 2, we lay the mathematical foundation to understand the perks of fixed-wing VTOL aircraft. Using rotor momentum theory, finite-wing aerodynamic model, and battery consumption equations, we develop analysis tools for power requirement for different modes of flight. Then we apply said theorems to design cases to reveal the benefits of fixed-wing flight efficiency over multirotor, and justify the need for a hybrid flying vehicle.

In Chapter 3, we start by considering a general flying vehicle that has both active aerodynamic elements providing significant forces, as well as a multi-dimensional thrust generation. A unified architecture of fixed-wing VTOL control is presented along with a rigid body and aerodynamics model used for control design. Robust position and attitude tracking controllers using forces and moments as inputs are then proposed, augmented with a force allocation method that realizes desired forces through attitude change and control allocation. Finally, simulation and experimental results using a prototype eVTOL are presented.

In Chapter 4, we focus on creating a velocity tracking controller with a LiP force model that can adapt to transient wind conditions, with feedback from a novel 3D airflow sensor for accurate aerodynamic force prediction. We propose a composite adaptation scheme that augments our force allocation method, and we prove convergence of tracking and prediction errors in the controller. Experimental results comparing several control techniques are presented, where the superior force tracking accuracy is demonstrated through tight holding of vehicle position under fast transient windspeed changes.

In Chapter 5, we present a novel control-centric design method for multirotor aircraft that ensures robustness against rotor failures. Specifically, we aim to optimize the design in a way that it maximizes the ability of multirotor to reach static hover after rotor failure. The notion of null controllability is first introduced with a derivation for multirotor case. Then, we define a quality measure used to evaluate a given design with the consideration of rotor failure. An optimization problem based on the quality measure is proposed that identifies a set of optimal design parameters maximizing an aircraft's ability to control its attitude. Finally, we illustrate the design procedure that leads to a design with improved tolerance to rotor failures, and demonstrate its performance on hardware experiments.

In Chapter 6, methods from deep learning are studied to help improve physics-based modeling for vehicle dynamics. The resulting DNN-based feedback controller is globally exponentially stable under bounded learning errors. This is achieved by exploiting the Lipschitz bound of spectrally normalized DNNs. It is intriguing that spectral normalization proves essential to the stability of the system both in a learning-theoretic and a control-theoretic sense. The proposed controller is applied to multirotor flying close to ground and large objects, where it learns how ground effects are coupled with unsteady aerodynamics and vehicular dynamics. Experiments are conducted for multirotor trajectory tracking during take-off, landing and cross-table maneuvers, which show improved smoothness and accuracy compared to nonlinear baseline controllers. To further study the capability of neural network in control design, we propose framework that use DNNs to represent full dynamics of a system as well as a associated controller and a observer. The entire approach relinquishes physics-based modeling, and utilizes full neural network structures for better generality in controlling nonlinear systems.

In Chapter 7, general delays in control systems are considered, as they become prominent in further improving control accuracy on certain flying vehicles. We propose a periodic predictor-based controller with numerical integration or differentiation. The system in consideration includes both dynamic and transport delays. We first introduce the undelayed, nonautonomous system of state and actuator input. Then, we describe the sample-based FOPDT model for actuator delay. Augmentations to an existing exponentially stable controller are progressively made to compensate for dynamic and transport delays. Hybrid stability analysis is provided to study the effects of sampling and numerical methods. We test an example system numerically for various attributes theorized.

Chapter 2

PRELIMINARIES AND DESIGNS

To understand the benefits and difficulties in a fixed-wing VTOL design, we will first introduce the aerodynamic principles behind rotors and wings.

2.1 Preliminaries for eVTOL and Fixed-wing Aircraft

2.1.1 Momentum Theory Analysis on Rotors

Although more complex aerodynamics methods exist for modeling forces on rotors such as blade element theory and computational fluid dynamics, we elect to use the simple momentum theory model to gain first-order understandings of the overall system [89].

Figure 2.1 shows the diagrams of how a rotating propeller would interact with the air in different flight conditions. Let d be the diameter of rotor and $A = \pi d^2$ its disk area. V_∞ is the freestream velocity of the air upstream of the rotor. In ascending and descending flight, this is V_c, the axial freestream velocity as depicted in Figures 2.1a and 2.1b. v_i is the additional induced velocity at the rotor plane. w and A_∞ are the induced velocity and area at the stream tube far downstream. Using conservation of mass, momentum, and energy, we can arrive at

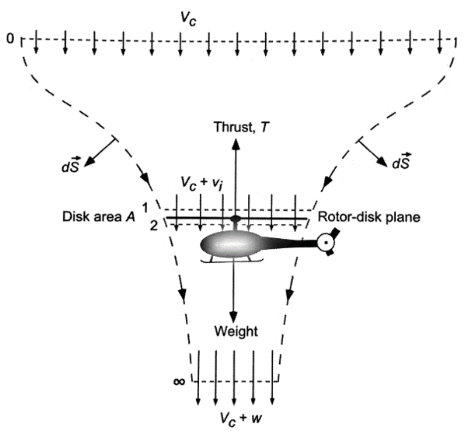
$$\begin{aligned}
\text{Mass:}\quad & \dot{m} = \rho A_\infty (V_c + w) = \rho A (V_c + v_i)\\
\text{Momentum:}\quad & T = \dot{m}(V_c + w) - \dot{m}V_c = \dot{m}w\\
\text{Energy:}\quad & T(V_c + v_i) = \frac{1}{2}\dot{m}(V_c + w)^2 - \frac{1}{2}\dot{m}V_c^2 = \frac{1}{2}\dot{m}w(2V_c + w)
\end{aligned}$$

with T the thrust generated by the rotor, and $\dot{m}$ the air mass flow rate. Thus we can solve for the induced velocity at rotor v_i and its induced power P_i, and its special cases when $V_c = 0$ for hover flight and $V_c < 0$ for descending flight as

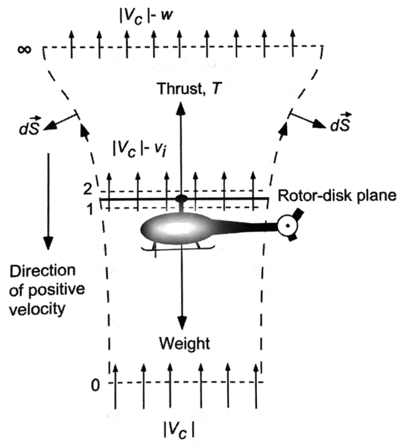
$$V_c > 0: \quad v_i = -\frac{1}{2}V_c + \frac{1}{2}\sqrt{V_c^2 + \frac{2T}{\rho A}}, \quad P_i = \frac{1}{2}TV_c + \frac{T}{2}\sqrt{V_c^2 + \frac{2T}{\rho A}} \tag{2.1a}$$

$$V_c = 0: \quad v_h = \sqrt{\frac{T}{2\rho A}}, \quad P_h = \frac{T^{3/2}}{\sqrt{2\rho A}} \tag{2.1b}$$

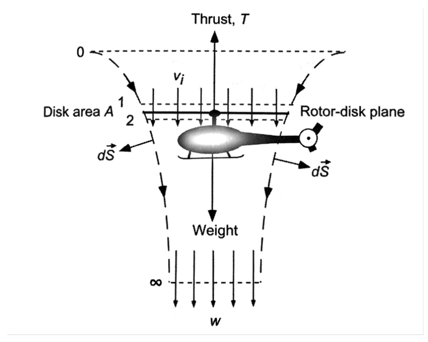
$$V_c < 0: \quad v_i = -\frac{1}{2}V_c - \frac{1}{2}\sqrt{V_c^2 + \frac{2T}{\rho A}}, \quad P_i = \frac{1}{2}TV_c - \frac{T}{2}\sqrt{V_c^2 + \frac{2T}{\rho A}} \tag{2.1c}$$



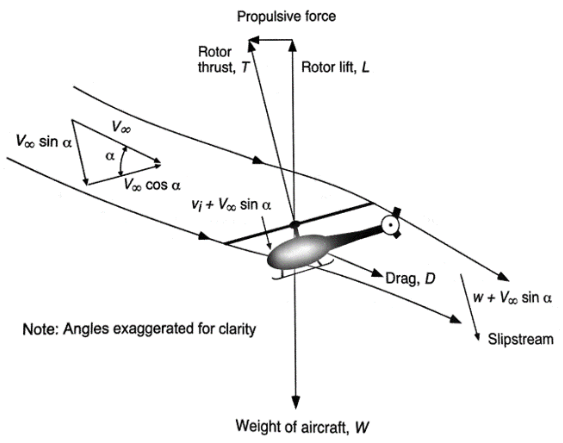
(a) Ascent



(b) Descent



(c) Hover



(d) Forward

Figure 2.1: Diagrams of rotor aerodynamics during ascending, descending, hovering, and forward flight. Pictures are excerpts from [89].

(2.1) is useful to relate power consumption to desired rotor force generated, which will be applied to designing the hybrid between VTOL and fixed-wing aircraft later on.

The limitation of induced power from momentum theory is that it does not consider profile power loss due to the spinning of the rotor. Moreover, real rotors would enter a vortex ring state during slow descending flight, which invalidates momentum theory as illustrated in Figure 2.2.

For forward flight shown in Figure 2.1d, the analysis is slightly complicated. Using the same method to construct equations from conservation laws, we can get the

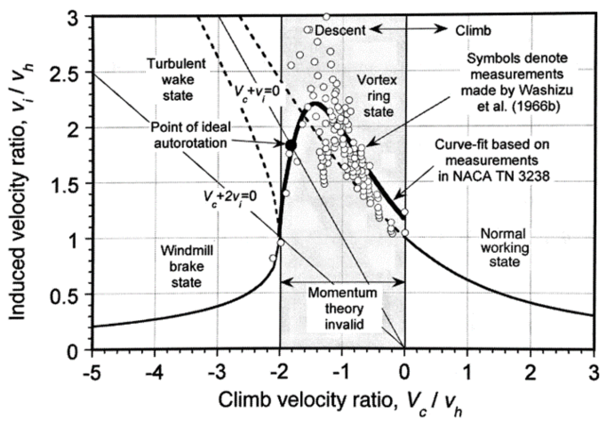


Figure 2.2: Induced velocity vs. climb velocity ratio, illustrating vortex ring state region where momentum theory is invalid. Picture is excerpt from [89].

following equation

$$T = 2\rho A v_i \sqrt{V_\infty^2 + 2V_\infty v_i \sin\alpha + v_i^2}, \tag{2.2}$$

which we can use to solve for v_i, and then substitute in to get induced power $P_i = Tv_i$.

For rotor profile power, i.e the power needed to spin rotors during flight, we have the following equation from [89]

$$P_0 = \rho A n^3 d^3 C_{P_0} \quad \text{with} \quad C_{P_0} = \frac{\bar{\sigma} C_{d_0}}{8}\left(1 + \bar{K}\left(\frac{V_\infty}{nd}\right)^2\right), \tag{2.3}$$

where $\bar{K} \approx 4.7$ and C_{d_0} can take empirical constant from vast test data. $\bar{\sigma}$ is the rotor solidity which can be calculated from rotor geometry.

2.1.2 Wing Aerodynamics

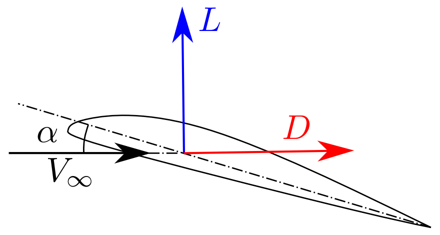


Figure 2.3: Illustration of lift and drag on airfoil.

On the other hand, aerodynamics of the wing is rather well understood for practical analysis [90]. Here we mainly consider the lift and drag forces on a wing with airfoil-shaped cross section as illustrated in Figure 2.3, with ρ as the air density and S_{ref} as the reference wing area. For an aircraft in steady flight, the force is resolved to lift component L normal to V_∞, and drag component D along V_∞. They are normally expressed with non-dimensional coefficients as

$$L = \frac{1}{2}\rho V_\infty^2 S_{\text{ref}} C_L \quad \text{and} \quad D = \frac{1}{2}\rho V_\infty^2 S_{\text{ref}} C_D. \tag{2.4}$$

C_L and C_D are often expressed with linear and quadratic models as

$$C_L = C_{L_0} + C_{L_1}\alpha \quad \text{and} \quad C_D = C_{D_0} + \frac{C_L^2}{\pi e \mathit{AR}}. \tag{2.5}$$

AR is the aspect ratio of a finite-sized wing, and $e < 1$ is the Oswald efficiency [90]. Other constants C_{L_0}, C_{L_1}, C_{D_0} can all be empirically determined or with simple theoretical models [90].

2.1.3 Battery Endurance

We derive a battery endurance equation from Peukert's equation [91]. The time it takes to discharge a battery t_{dc} is approximated by

$$t_{\text{dc}} = Rt^{1-l}\left(\frac{\eta_{\text{tot}} C}{P_{\text{tot}}}\right)^l; \tag{2.6}$$

with battery hour rating $Rt \approx 1$ hr for small batteries, discharge parameter $l \approx 1.3$ for LiPo, total battery capacity C in Watt $\cdot$ hr, total power consumption P_{tot} in Watt, and total transmission efficiency η_{tot}.

2.2 Power Required for Mix-modes Flight

With tools we introduced in Section 2.1, we can now analyze the power required for different stages of a fixed-wing VTOL flight. Suppose N_v is the number of vertical thrusters on the vehicle used during VTOL flight, and N_c is the number of cruise thrusters mounted parallel to wing surface for cruise flight. The total weight of the vehicle is W. Steady level flight refers to the vehicle flying level with forward velocity V_∞. The vehicle will be in force equilibrium during the entire flight stage.

2.2.1 Hover Power

For a vehicle during hover, all the power needed is to generate thrusts to balance its weight. We rewrite (2.1b) as $P_h(A, T)$ showing the functional dependency. Thus the power required $P_{R,h}$ for hover is

$$P_{R,h} = \sum_{j=1}^{N_v} \Big(P_h(A_j, T_j) + P_0(A_j, n_j)\Big) + P_p(v_i). \tag{2.7}$$

with $A_v = \sum_{j=1}^{N_v} A_j$ as the total area of all vertical rotors. It is assumed that all rotors are identical and share equal contribution during thrust generation (moment balanced). n_j will be estimated based on the rotor size and thrust, in general as $n_j(d_j, T_j)$, which in this case is $T_j = W/N_v$. Parasite power P_p comes from drag experienced on bodies and wings. Parasite drag would be nonexistent for $V_\infty = 0$, but can be included if the induced velocity around the vehicle is considered.

2.2.2 VTOL Mode Flight Power

In VTOL mode, the vehicle is assumed to fly similarly to a helicopter or multirotor. The vehicle will pitch down with angle α similarly to the illustration in Figure 2.1d, and we need to balance between thrust, weight, parasite drag, as well as lift generated by the wing. Let us combine (2.4) and (2.5) and write $L(-\alpha, V_\infty)$ and $D(-\alpha, V_\infty)$. Here we assume the downward pitch is needed for forward VTOL flight, but incurs a negative angle-of-attack. First we need to solve for the pitch angle from

$$W - L(-\alpha^*, V_\infty)\tan\alpha^* - D(-\alpha^*, V_\infty) = 0.$$

Then for each vertical rotor we have

$$T_j = \frac{W - L(-\alpha^*, V_\infty)}{\cos\alpha^*\ N_v}.$$

Then we can solve (2.2) as a function of $v_i^*(A_j, T_j, \alpha^*, V_\infty)$. Moreover, the aerodynamic power in forward flight is $P_A = D(-\alpha, V_\infty)V_\infty$, which is needed to overcome the drag of the vehicle. Then the required flight power is

$$P_{R,\text{VTOL}} = \sum_{j=1}^{N_v}\Big(T_j v_i^* + P_0(A_j, n_j, V_\infty)\Big) + P_A. \tag{2.8}$$

2.2.3 Fixed-wing Mode Flight Power

When the wing is utilized for lift generation, we assume a small AoA α during steady flight. However, the wing would typically stall at high AoA and has maximum C_L which corresponds to a minimum stall speed V_{stall}. In our case, vertical rotors can compensate for any deficiency if V_∞ is below stall speed. Thus we have wing lift $\bar{L}$:

$$\begin{cases} \bar{L} = W & \text{if} \quad V_\infty > V_{\text{stall}} \\ \bar{L} = \frac{1}{2}\rho V_\infty^2 S_{\text{ref}} C_{L\,\text{max}} & \text{if} \quad V_\infty \le V_{\text{stall}} \end{cases} \tag{2.9}$$

From (2.4) and (2.5) we can derive the wing drag as function of the wing lift:

$$\bar{D}(\bar{L}, V_\infty) = \frac{1}{2}\rho V_\infty^2 S_{\text{ref}} C_{D_0} + \frac{2\bar{L}^2}{(\pi e A\!\!\!R)\rho V_\infty^2 S_{\text{ref}}}. \tag{2.10}$$

Similarly, it is also easy to get $\bar{\alpha}$ from $\bar{L}$ and for each vertical thruster, we have

$$T_{v,j} = \frac{W - \bar{L}}{\cos\bar{\alpha}\ N_v}, \quad \bar{v}_{ij}(A_j, T_{v,j}, -\bar{\alpha}, V_\infty), \quad and \quad P_{0j} = P_0(A_j, n_j, V_\infty).$$

Note that $-\bar{\alpha}$ is due to the convention difference for AoA between rotor and wing. The total drag force D_A can be calculated as

$$D_A = \bar{D} + \bar{L}\frac{\eta_A \bar{v}_i}{V_\infty}$$

where the additional term is to account for interaction between the vertical rotor and wing, by adding some induced velocity onto the wing surface, tuned by an efficiency factor η_A. Following the same notation, we have aerodynamic power $P_A = D_A V_\infty$. For each cruise thruster, we use

$$T_{c,k} = \frac{D_A}{N_c}, \quad \bar{v}_{ik}(A_k, T_{v,k}, -\bar{\alpha} + 90^\circ, V_\infty), \quad \text{and} \quad P_{0k} = P_0(A_k, n_k, V_\infty).$$

Assembling the terms, we have the require power for fixed-wing flight:

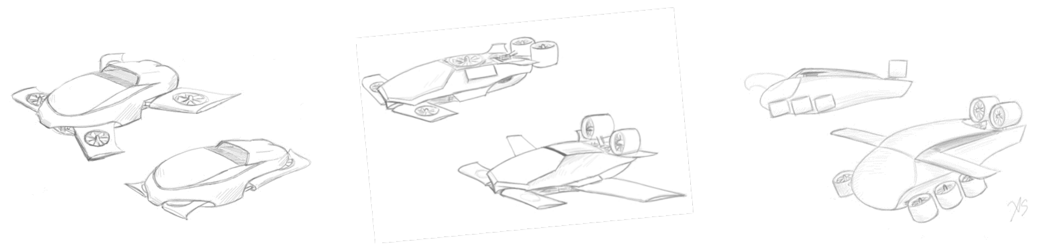
$$P_{R,\text{FW}} = \sum_{j=1}^{N_v}\left(T_{v,j}\bar{v}_{ij} + P_{0j}\right) + \sum_{k=1}^{N_c}\left(T_{c,k}\bar{v}_{ik} + P_{0k}\right) + P_A. \tag{2.11}$$

Now we have the power requirement for all modes of flight. In reality, there exist multiple if not infinitely many solutions for mixed-mode flight configuration as vertical and cruise thrusters can be combined arbitrarily. The settings we selected in our analysis here are based on a conventional flight control philosophy, where the wingborne lift is prioritized.

2.3 Example Fixed-wing VTOL Designs for UAM

With Caltech CAST's initiative in research of the next generation aircraft for UAM, conceptual designs are proposed based on real-world needs. Example concept sketches are shown in Figure 2.4, all including different numbers of vertical and cruise thrusters as well as wing surfaces. Using the mathematical tools developed so far in Section 2.2, we can analyze the potential performance impact on different design choices.

From various concepts, we eventually choose to pursue a tilt-rotor with a multirotor hybrid configuration. Through trade studies on vehicle parameters, we predict the performance of a full-scale vehicle with realistic specs of different components such as battery packs and electric motors. Scaled prototypes are constructed to validate the analysis method used and test automation algorithms for controlling the crafts.



(a) Concept 1 (b) Concept 2 (c) Concept 3

Figure 2.4: Example conceptual designs for fixed-wing eVTOL sketched by the author in the beginning of Caltech's AFA project.

Details of those prototypes are included in Appendix B. A skeletal version (AFA 1.0) to verify the design of rotor and wing configuration is first built and tested. Then a preliminary design for a full-scale version is proposed with a smooth aerodynamic body and shrouded rotors. Similarly, this design is materialized as another scale prototype (AFA 2.0) to further the understanding of its aerodynamic properties. Figure 2.5 shows a computer-rendered image of AFA 2.0 concept as a full-scale vehicle in urban Manhanttan, New York.

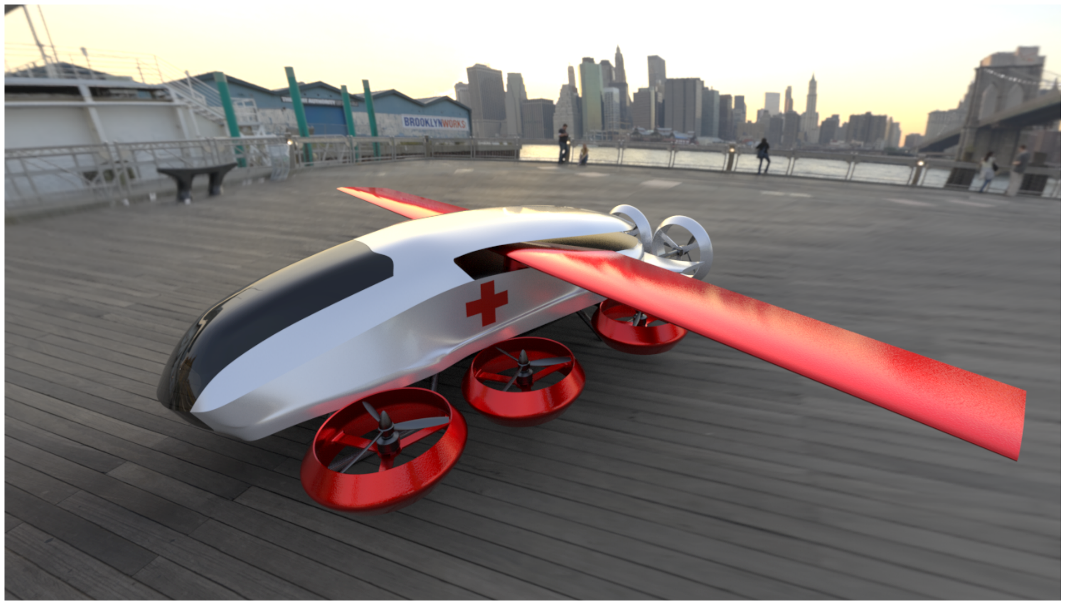
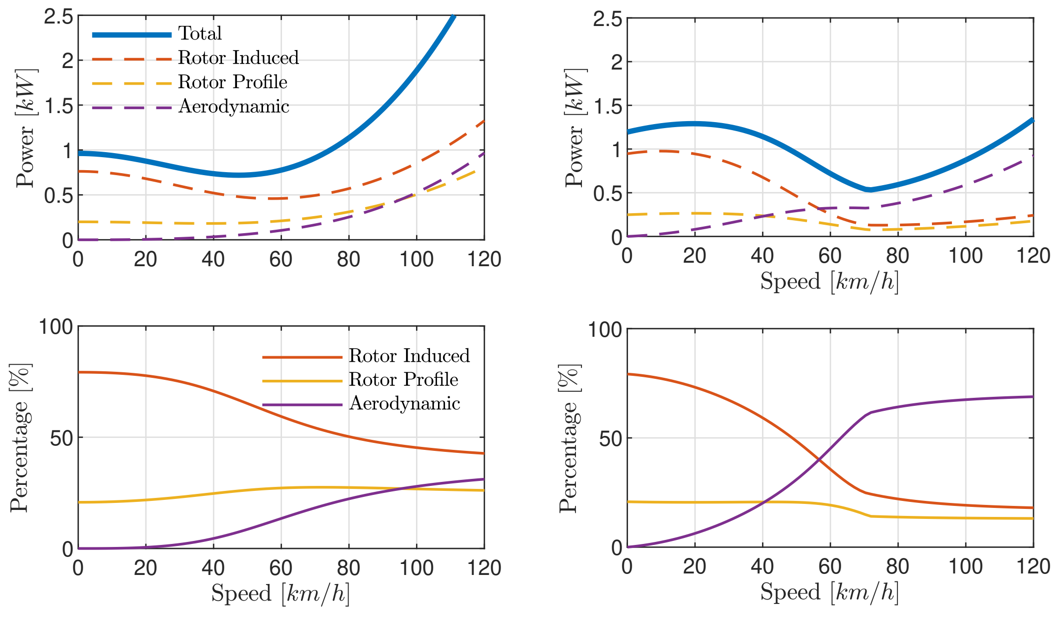


Figure 2.5: Full-scale AFA 2.0 concept design. The image is rendered in Autodesk VRED.

2.4 Example Performance of UAM Design

Using specs of scaled AFA 2.0 from the design process, we can predict the power requirement using equations (2.7), (2.8), and (2.11). Details of AFA 2.0 specs are listed in Appendix B. In summary, it is a 5.5 kg vehicle with a 0.9 m long and 0.3 m wide body. The wing has a 1.6 m span and an aspect ratio of roughly $Æ \approx 8.6$. It consists of 6 dedicated vertical rotors and 2 tiltable cruise thrusters, all with 15 cm diameter propellers. By using some empirical and experimental values for rotor and wing properties, we can calculate the required power for rotors during flight.

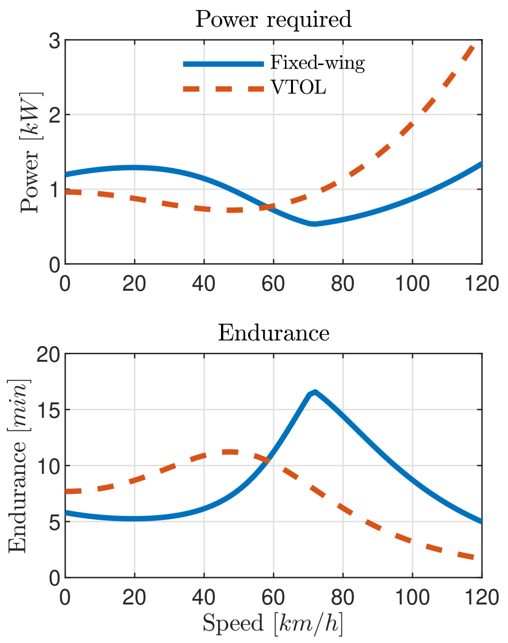


(a) VTOL Mode (b) Fixed-wing Mode

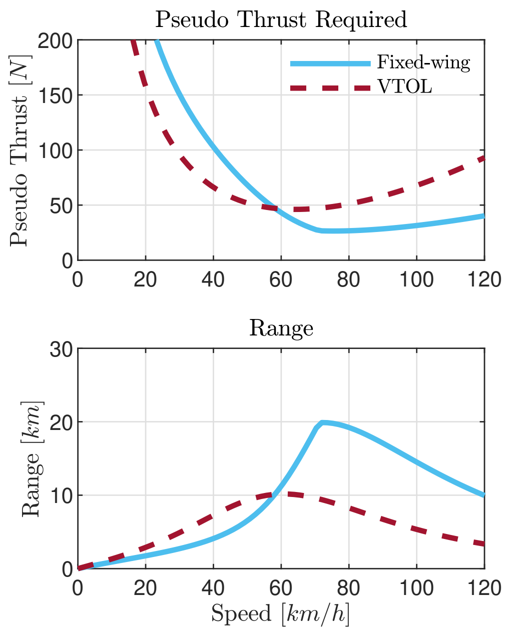
Figure 2.6: Required power P_r breakdowns for VTOL and fixed-wing flight modes.

Figure 2.6 shows the required power for rotors for both VTOL mode and fixed-wing mode. In the case of AFA 2.0, VTOL mode refers to all 8 rotors in vertical configuration and propel the vehicle by pitching forward in the flight direction, similar to a multirotor. Throughout the airspeed range, the induced power from rotors is the dominating power consumption, with profile and aerodynamic power becoming apparent at higher speed. On the other hand, the fixed-wing mode will have 2 cruise rotors in forward configuration, leaving 6 vertical rotors to provide any lift assist if necessary. Contrary to VTOL mode, the induced power from rotors becomes much smaller as wing begins to generate enough lift to support vehicle weight, in which case aerodynamic power becomes the main factor. Also can be seen between graphs in Figure 2.6 is that fixed-wing mode consumes much less energy for faster flight, making it ideal for traversing longer distance. This is the main benefit when proposing fixed-wing VTOL as the prime candidate for UAM application.

The efficiency benefit of fixed-wing flight is further illustrated in Figure 2.7, where



(a) Power required and endurance



(b) Thrust required and range

Figure 2.7: Endurance and range comparisons between VTOL and fixed-wing flight modes.

in addition to power requirement P_r, we also plot a pseudo required thrust P_r/V_∞. We denote the time for the aircraft to stay airborne as *endurance*. From (2.6), we can expect that the maximum *endurance* is determined by the power required to stay airborne. Conversely, the distance traveled by the aircraft on a single battery charge, denoted as *range*, will be determined by pseudo required thrust P_r/V_∞, which has a unit of N.

It is obvious from Figure 2.7 that for both *endurance* and *range*, fixed-wing flight is more efficient at higher speeds compared to VTOL and vice versa. Note that since we have a tilt-rotor configuration, the fixed-wing mode will have fewer rotors to sustain weight at lower speeds, which will not be the case for a copter-plane configuration. Overall, fixed-wing flight provides superior *endurance* and *range* when reaching optimum cruise speed, offering about twice the performance gains.

2.5 Chapter Summary

In this chapter, we began by providing the theoretical foundations for fixed-wing VTOL aircraft. With momentum theory, we first constructed the power-thrust relationship for rotors in different flight conditions, then combined the rotor power with wing aerodynamics to derive the total power required during flights. Using the total flight power model together with the battery consumption model, we gave examples of UAM designs that demonstrated the benefits of fixed-wing flight for better endurance and range. It became clear that a fixed-wing VTOL hybrid was essential for UAM application.

Chapter 3

UNIFIED ARCHITECTURE FOR FLIGHT CONTROL

This chapter is largely based on [44], where unified control architecture for fixed-wing eVTOL vehicles is proposed and analyzed. We consider a VTOL vehicle that can produce forces directly with distributed electric propulsion and non-negligible aerodynamic forces and moments from its body or wings. An example VTOL is given in Figure 3.1, which is comprised of six side rotors that can produce upward forces, two back rotors that are able to produce forward and upward forces, and a pair of wings used for lift production.

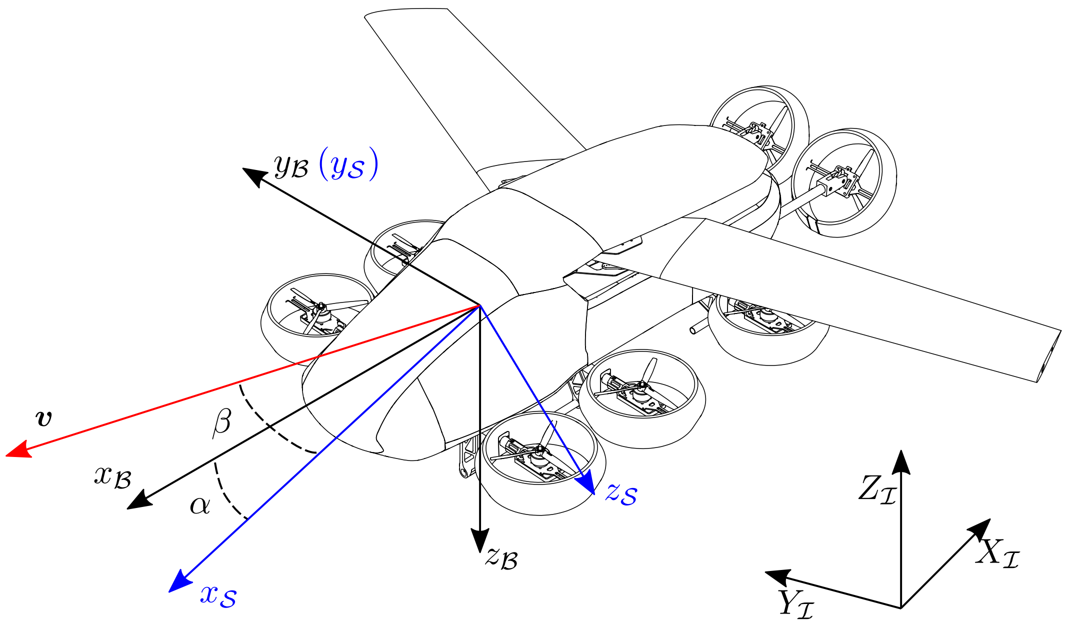


Figure 3.1: Illustration of an example VTOL aircraft and associated frames of references: (1) inertial frame $\mathcal{I}$; (2) body frame $\mathcal{B}$; stability frame $\mathcal{S}$.

3.1 General Fixed-Wing VTOL Dynamic Model

A six degree-of-freedom (DoF) dynamics model for VTOL aircraft is considered. The system states are defined by inertial position $\boldsymbol{p} \in \mathbb{R}^3$ and velocity $\boldsymbol{v} \in \mathbb{R}^3$ at the center-of-mass (CoM) of the vehicle; attitude as rotation matrix $\boldsymbol{R} \in SO(3)$; and angular velocity $\boldsymbol{\omega} \in \mathbb{R}^3$ in the body frame. The dynamics are expressed as:

$$\dot{\boldsymbol{p}} = \boldsymbol{v} \qquad \dot{\boldsymbol{v}} = \boldsymbol{g} + \boldsymbol{R}\boldsymbol{f}_b \tag{3.1}$$

$$\dot{\boldsymbol{R}} = \boldsymbol{R}\boldsymbol{S}(\boldsymbol{\omega}) \qquad \boldsymbol{J}\dot{\boldsymbol{\omega}} = \boldsymbol{S}(\boldsymbol{J}\boldsymbol{\omega})\boldsymbol{\omega} + \boldsymbol{\tau}_b \tag{3.2}$$

where $\boldsymbol{J} \in \mathbb{R}^{3\times3}$ is the inertia matrix of the vehicle in body-frame, $\boldsymbol{g}$ is the constant gravity vector in the inertial frame, and $\boldsymbol{S}(\cdot) : \mathbb{R}^3 \to SO(3)$ is a skew-symmetric mapping such that $\boldsymbol{a} \times \boldsymbol{b} = \boldsymbol{S}(\boldsymbol{a})\boldsymbol{b}$. External forces and moments on the vehicle are grouped into $\boldsymbol{f}_b$ and $\boldsymbol{\tau}_b$. $\boldsymbol{f}_b$ is normalized with mass and has a unit of m/s^2, while moment has a unit of Nm. We can decompose

$$\boldsymbol{f}_b = \boldsymbol{f}_T + \boldsymbol{f}_A, \qquad \boldsymbol{\tau}_b = \boldsymbol{\tau}_T + \boldsymbol{\tau}_A \tag{3.3}$$

since the external forces and moments on flying vehicles generally consist of only contributions from thrusters ($\boldsymbol{f}_T$, $\boldsymbol{\tau}_T$) and aerodynamics ($\boldsymbol{f}_A$, $\boldsymbol{\tau}_A$).

3.1.1 Thrust and Torque from Electric Propellers

Axial thrust T and torque τ are the dominating force and moment generated when spinning up a propeller [90],

$$T = C_T \rho d^4 n^2 \tag{3.4}$$

$$\tau = C_Q \rho d^5 n^2 \tag{3.5}$$

with non-dimensional coefficient C_T, C_Q, air density ρ, propeller diameter d, and propeller rotation speed n. Let the thrust from the j-th propeller in the body frame be $T_j \hat{\boldsymbol{z}}_j$, where $\hat{\boldsymbol{z}}_j$ is the unit vector in the thrust direction. In turn, the axial torque vector would be $\tau_j \hat{\boldsymbol{z}}_j$.

Let $\boldsymbol{r}_j$ be the vector from the vehicle's CoM to the rotor axle. The moment at CoM of the vehicle from rotor j can be then written as

$$\begin{aligned} \boldsymbol{\tau}_{j,\text{cm}} &= \boldsymbol{r}_j \times T_j \hat{\boldsymbol{z}}_j + \tau_j \\ &= T_j \underbrace{\left(\boldsymbol{S}(\boldsymbol{r}_j)\hat{\boldsymbol{z}}_j + \frac{C_Q d}{C_T} \hat{\boldsymbol{z}}_j \right)}_{\boldsymbol{\mu}_j}. \end{aligned} \tag{3.6}$$

In practice, we can obtain the physical parameters in the model through bench testing. Thus without loss of generality, we can safely assume that direct control of axial thrust for each rotor is given. Let $\boldsymbol{u}_T = [T_1, \cdots, T_{N_T}]^\top$ be the vector of control input with N_T as number of rotors on the vehicle, then

$$\begin{bmatrix} \boldsymbol{f}_T \\ \boldsymbol{\tau}_T \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{z}}_1 & \cdots & \hat{\boldsymbol{z}}_{N_T} \\ \boldsymbol{\mu}_1 & \cdots & \boldsymbol{\mu}_{N_T} \end{bmatrix} \begin{bmatrix} T_1 \\ \vdots \\ T_{N_T} \end{bmatrix} = \boldsymbol{B}_T \boldsymbol{u}_T. \tag{3.7}$$

Through careful design of the vehicle's configuration, one can get a well-conditioned $\boldsymbol{B}_T$ matrix.

3.1.2 Aerodynamic Force and Moment on Wings

Forces and moments on wing surfaces are mainly depended on the relative wind velocities. We define such incident wind velocity as

$$\boldsymbol{v}_i = \boldsymbol{R}^\top (\boldsymbol{v} - \boldsymbol{v}_w) \tag{3.8}$$

which is the combination of ambient wind velocity $\boldsymbol{v}_w$ and vehicle's inertial velocity $\boldsymbol{v}$. Then we can define the following quantities related to aerodynamic force and moment calculations. Let V_∞, V_{xz}, and V_{xy} be free-stream wind speed, and projected speeds in body xz-plane and xy-plane, respectively. They can be calculated as

$$V_\infty = \|\boldsymbol{v}_i\|, \ \alpha = \arctan\left(\frac{v_{iz}}{v_{ix}}\right), \ \beta = \arcsin\left(\frac{v_{iy}}{V_\infty}\right), \tag{3.9}$$

which are incident wind speed V_∞, angle-of-attack (AoA) α, and angle-of-sideslip (AoS) β. For aerodynamic forces, we define lift L and drag D in the body xz-plane, and side force Y in body y-axis. Similarly, aerodynamic moments $\boldsymbol{\tau}_A = [\tau_{Ax}, \tau_{Ay}, \tau_{Az}]$ are defined for each body axis. They are often non-dimensionalized with

$$L = \frac{1}{2}\rho V_\infty^2 S_{\text{ref}} C_L, \qquad D = \frac{1}{2}\rho V_\infty^2 S_{\text{ref}} C_D, \qquad Y = \frac{1}{2}\rho V_\infty^2 S_{\text{ref}} C_Y. \tag{3.10}$$

$$\tau_{Ax} = \frac{1}{2}\rho V_\infty^2 S_{\text{ref}} \bar{c} C_{mx}, \quad \tau_{Ay} = \frac{1}{2}\rho V_\infty^2 S_{\text{ref}} \bar{c} C_{my}, \quad \tau_{Az} = \frac{1}{2}\rho V_\infty^2 S_{\text{ref}} \bar{c} C_{mz}, \tag{3.11}$$

with S_{ref} as the reference wing area and $\bar{c}$ as the chord length [92].

For aircraft, the non-dimensional coefficients C_L, C_D, C_Y, C_{mx}, C_{my}, and C_{mz} are typically tabulated from wind tunnel tests as a function of various body and aerodynamic states. We instead use the following method that combines models of different accuracy. The general full range flat-plate model $(\cdot)^{\text{fr}}$ can be used to get the overall trend, while the linear model $(\cdot)^{\text{lin}}$ provides more accuracy at low α:

$$\begin{aligned} C_L^{\text{fr}} &= k_L \sin(2\alpha), & C_L^{\text{lin}} &= C_{L_0} + C_{L_1}\alpha, \\ C_D^{\text{fr}} &= k_{D1}\sin^2\alpha + k_{D0}, & C_D^{\text{lin}} &= C_{D_0} + C_{D_1}\alpha + C_{D_2}\alpha^2, \\ C_Y^{\text{fr}} &= k_Y \sin\beta, & C_Y^{\text{lin}} &= C_{L_\beta}\beta. \end{aligned} \tag{3.12}$$

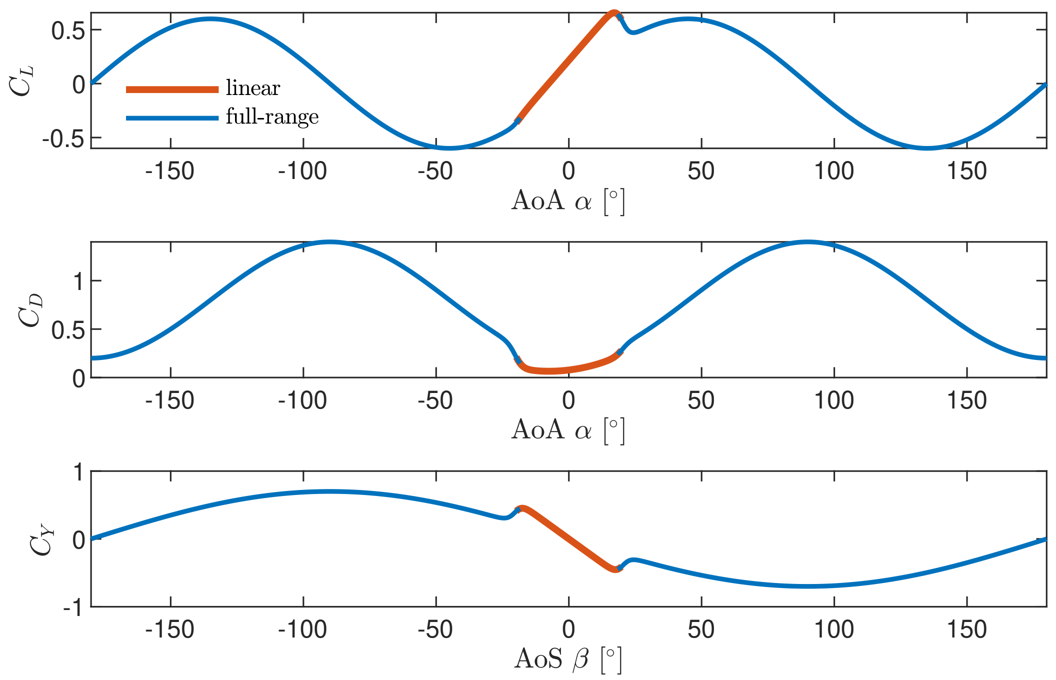


Figure 3.2: Demonstration of aerodynamic coefficient from $[-180, 180]$ degrees.

We use a hyperbolic tangent tanh blending function to aggregate two models together [93]. The blending of two models can be tuned for accurate prediction within the desired operating range, and bounded error everywhere else. Figure 3.2 shows an example of this method plot on full ranges of α and β, which demonstrates a similar trend compared to test data for both airfoils [94] and real aircraft [8]. To summarize, we can express $\boldsymbol{f}_A$ by using (3.12):

$$\boldsymbol{f}_A = \frac{1}{2}\rho S_{\text{ref}} V_\infty^2 \begin{bmatrix} C_L(\alpha)\sin\alpha - C_D(\alpha)\cos\alpha \\ -C_Y(\beta) \\ -C_L(\alpha)\cos\alpha - C_D(\alpha)\sin\alpha \end{bmatrix}, \tag{3.13}$$

On the other hand, the moment coefficients are computed using a linear model refered from [92], assuming that a VTOL vehicle is designed to be symmetric with respect

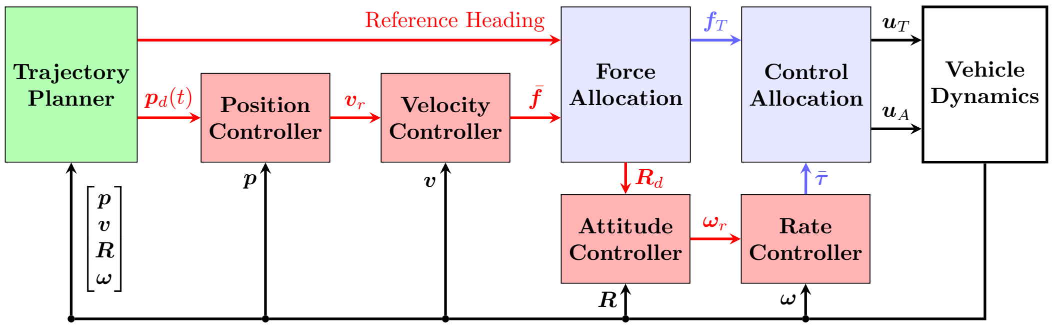


Figure 3.3: Schematic of the proposed controller framework for winged VTOL aircraft or flying cars.

to the xz-plane. We find

$$\begin{aligned} C_{mx} &= \underbrace{k_{x0} + k_{x1}\frac{\bar{c}\omega_x}{2V_\infty} + k_{x2}\frac{\bar{c}\omega_z}{2V_\infty} + k_{x3}\beta}_{C'_{mx}} + \Delta C_{mx} = C'_{mx} + \Delta C_{mx}, \\ C_{my} &= \underbrace{k_{y0} + k_{y1}\frac{\bar{c}\omega_y}{2V_\infty} + k_{y2}\alpha}_{C'_{my}} + \Delta C_{my} = C'_{my} + \Delta C_{my}, \\ C_{mz} &= \underbrace{k_{z0} + k_{z1}\frac{\bar{c}\omega_x}{2V_\infty} + k_{z2}\frac{\bar{c}\omega_z}{2V_\infty} + k_{z3}\beta}_{C'_{mz}} + \Delta C_{mz} = C'_{mz} + \Delta C_{mz}. \end{aligned} \tag{3.14}$$

The control surface deflections δ_{Aj} are assumed to only affect the moment coefficients linearly:

$$\Delta C_{m_i} = \sum_{j=1}^{n_A} k_{i,\delta_j}\delta_{A,j}, \qquad i \in \{x, y, z\}. \tag{3.15}$$

We can then define control input $\boldsymbol{u}_A = [\delta_{A1} \cdots \delta_{An_A}]$ as vector of deflection angles, with n_A being the number of deflection surfaces. Thus $\boldsymbol{\tau}_A$ can be simplified to

$$\begin{aligned} \boldsymbol{\tau}_A &= \underbrace{\frac{1}{2}\rho S_{\text{ref}}\bar{c}V_\infty^2 \begin{bmatrix} C'_{mx} \\ C'_{my} \\ C'_{mz} \end{bmatrix}}_{\boldsymbol{\tau}'_A} + \underbrace{\frac{1}{2}\rho S_{\text{ref}}\bar{c}V_\infty^2 \begin{bmatrix} k_{x,\delta_1} & \cdots & k_{x,\delta_{n_A}} \\ k_{y,\delta_1} & \cdots & k_{y,\delta_{n_A}} \\ k_{z,\delta_1} & \cdots & k_{z,\delta_{n_A}} \end{bmatrix}}_{\boldsymbol{B}_A} \begin{bmatrix} \delta_{A,1} \\ \vdots \\ \delta_{A,N_A} \end{bmatrix} \\ &= \boldsymbol{\tau}'_A + \boldsymbol{B}_A\boldsymbol{u}_A. \end{aligned} \tag{3.16}$$

3.2 Unified Control Architecture

By substituting (3.7), (3.13), and (3.16) into (3.3), we can rewrite $\boldsymbol{f}_b$ and $\boldsymbol{\tau}_b$ as,

$$\begin{bmatrix} \boldsymbol{f}_b \\ \boldsymbol{\tau}_b \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_A(\boldsymbol{v}_i) \\ \boldsymbol{\tau}'_A(\boldsymbol{v}_i, \boldsymbol{\omega}) \end{bmatrix} + \begin{bmatrix} 0 \\ \boldsymbol{B}_A \boldsymbol{u}_A \end{bmatrix} + \boldsymbol{B}_T \boldsymbol{u}_T. \tag{3.17}$$

We see that $\boldsymbol{B}_T \boldsymbol{u}_T$ and $\boldsymbol{B}_A \boldsymbol{u}_A$ indicate that direct inversion can be applied to generate forces and moments, while $\boldsymbol{f}_A$ presents a difficulty in determining the desired attitude. We intend to use $\boldsymbol{R}\boldsymbol{f}_b$ and $\boldsymbol{\tau}_b$ as control inputs in our position and attitude dynamics, then $\bar{\boldsymbol{f}}$ and $\bar{\boldsymbol{\tau}}$ can be solved as desired forces and moments. Then we use a novel *force allocation* method that outputs a desired attitude which ensures $\boldsymbol{R}\boldsymbol{f}_b \to \bar{\boldsymbol{f}}$. The proposed control system architecture is shown in Figure 3.3.

3.2.1 Position Tracking Controller

Given a desired position trajectory $\boldsymbol{p}_d(t)$ to track, a position error is defined as $\tilde{\boldsymbol{p}} = \boldsymbol{p} - \boldsymbol{p}_d$. We intend to design a manifold $\tilde{\boldsymbol{v}}$ on which the position error converges to zero exponentially:

$$\tilde{\boldsymbol{v}} = \dot{\tilde{\boldsymbol{p}}} + \boldsymbol{\Lambda}_p \tilde{\boldsymbol{p}} = \boldsymbol{v} - \left(\dot{\boldsymbol{p}}_d - \boldsymbol{\Lambda}_p \tilde{\boldsymbol{p}}\right), \tag{3.18}$$

where $\boldsymbol{\Lambda}_p$ is a positive definite gain matrix for the position error. We propose the following controller using a required net force $\bar{\boldsymbol{f}}$ as an input:

Proposition 3.1. *The position controller is defined as*

$$\bar{\boldsymbol{f}} = -\boldsymbol{g} + \dot{\boldsymbol{v}}_r - \boldsymbol{K}_v \tilde{\boldsymbol{v}} - \boldsymbol{K}_p \tilde{\boldsymbol{p}}, \tag{3.19}$$

$$\textit{with} \quad \tilde{\boldsymbol{v}} = \boldsymbol{v} - \boldsymbol{v}_r, \quad \boldsymbol{v}_r = \dot{\boldsymbol{p}}_d - \boldsymbol{\Lambda}_p \tilde{\boldsymbol{p}}, \tag{3.20}$$

where $\boldsymbol{K}_v$ and $\boldsymbol{K}_p$ are positive definite gain matrices. Suppose the difference between $\bar{\boldsymbol{f}}$ and its realization is $\tilde{\boldsymbol{f}}$. If it is bounded by some $\|\tilde{\boldsymbol{f}}\| \le \epsilon$, then $\boldsymbol{v} \to \boldsymbol{v}_r$ and $\boldsymbol{p} \to \boldsymbol{p}_d$ exponentially within a ball of radius r, denoted as b_r, controlled by ϵ and gain matrices $\boldsymbol{\Lambda}_p$, $\boldsymbol{K}_p$, and $\boldsymbol{K}_v$.

Proof. Using (3.1), (3.19), and (3.20), we get the closed-loop dynamics of $\tilde{\boldsymbol{v}}$:

$$\dot{\tilde{\boldsymbol{v}}} + \boldsymbol{K}_v \tilde{\boldsymbol{v}} + \boldsymbol{K}_p \tilde{\boldsymbol{p}} = \tilde{\boldsymbol{f}}. \tag{3.21}$$

Differentiating the Lyapunov function $\mathcal{V}(\tilde{\boldsymbol{v}}, \tilde{\boldsymbol{p}}) = (1/2)\left(\|\tilde{\boldsymbol{v}}\|^2 + \tilde{\boldsymbol{p}}^\top K_p \tilde{\boldsymbol{p}}\right)$, it yields

$$\begin{aligned} \dot{\mathcal{V}} &= \tilde{\boldsymbol{v}}^\top(-\boldsymbol{K}_v \tilde{\boldsymbol{v}} - \boldsymbol{K}_p \tilde{\boldsymbol{p}} + \tilde{\boldsymbol{f}}) + \tilde{\boldsymbol{p}}^\top \boldsymbol{K}_p \left(\tilde{\boldsymbol{v}} - \boldsymbol{\Lambda}_p \tilde{\boldsymbol{p}}\right) \\ &= -\tilde{\boldsymbol{v}}^\top \boldsymbol{K}_v \tilde{\boldsymbol{v}} - \tilde{\boldsymbol{p}} \boldsymbol{K}_p \boldsymbol{\Lambda}_p \tilde{\boldsymbol{p}} + \tilde{\boldsymbol{v}}^\top \tilde{\boldsymbol{f}}. \end{aligned}$$

By letting $\boldsymbol{\zeta} = [\tilde{\boldsymbol{v}}^\top, \tilde{\boldsymbol{p}}^\top]^\top$ and using the comparison method from [95], we get

$$\|\boldsymbol{\zeta}(t)\| \le \sqrt{\frac{c_2}{c_1}}\|\boldsymbol{\zeta}(t_0)\| \exp\left(\frac{c_3}{c_2}(t-t_0)\right) + \frac{c_2}{c_1c_3}\sup_{t\ge t_0}\|\tilde{\boldsymbol{f}}\|,$$

with

$$\begin{aligned} c_1 &= \min\{1, \lambda_{\min}(\boldsymbol{K}_p)\} \\ c_2 &= \max\{1, \lambda_{\max}(\boldsymbol{K}_p)\} \\ c_3 &= \min\{\lambda_{\min}(\boldsymbol{K}_v), \lambda_{\min}(\boldsymbol{K}_p)\lambda_{\min}(\boldsymbol{\Lambda}_p)\}. \end{aligned}$$

Since $\|\tilde{\boldsymbol{f}}\| \le \epsilon$ on a compact set, $\therefore\ \sqrt{\|\tilde{\boldsymbol{v}}\|^2 + \|\tilde{\boldsymbol{p}}\|^2} \to b_r$ with $r = (\epsilon c_2)/(c_1c_3)$. ■

3.2.2 Attitude Tracking Controller on $SO(3)$ and Quaternion

We design a controller that provides global tracking of any attitude trajectory. Assume that the desired time trajectories of attitude $\boldsymbol{R}_d(t)$ and its associated angular velocity $\boldsymbol{\omega}_d(t)$ are given by the force allocation block, and define an error rotation matrix which is defined as $\tilde{\boldsymbol{R}} = \boldsymbol{R}_d^\top\boldsymbol{R}$. Following [96], it can be shown that the error function used in $SO(3)$ is equivalent to the vector part of the error quaternion $\tilde{\boldsymbol{q}} = [\tilde{q}_0, \tilde{\boldsymbol{q}}_v^\top]^\top$ from $\tilde{\boldsymbol{R}}$ with $\tilde{q}_0 \ge 0$:

$$\tilde{q}_0 = \frac{1}{2}\sqrt{1+\mathrm{tr}(\tilde{\boldsymbol{R}})}, \qquad \tilde{\boldsymbol{q}}_v = \frac{1}{4\tilde{q}_0}(\tilde{\boldsymbol{R}} - \tilde{\boldsymbol{R}}^\top)^\vee, \tag{3.22}$$

where the $\vee$ symbol here denotes the *vee* map which is the inverse operation of $\boldsymbol{S}(\cdot)$. We use singularity-free formulas when $\mathrm{tr}(\tilde{\boldsymbol{R}}) = -1$ [97], [98]. The angular rate error $\boldsymbol{e}_\omega$ can be obtained by differentiating $\tilde{\boldsymbol{R}}$ with respect to time as

$$\begin{aligned} \dot{\tilde{\boldsymbol{R}}} &= \boldsymbol{R}_d^\top\dot{\boldsymbol{R}} + \dot{\boldsymbol{R}}_d^\top\boldsymbol{R} &(3.23)\\ &= \tilde{\boldsymbol{R}}\boldsymbol{S}\left(\boldsymbol{\omega} - \tilde{\boldsymbol{R}}^\top\boldsymbol{\omega}_d\right), &(3.24) \end{aligned}$$

which gives $\boldsymbol{e}_\omega = \boldsymbol{\omega} - \tilde{\boldsymbol{R}}^\top\boldsymbol{\omega}_d$ in the current body frame. The kinematic relation between $\dot{\tilde{\boldsymbol{q}}}$ and $\boldsymbol{e}_\omega$ is given as

$$\dot{\tilde{\boldsymbol{q}}} = \frac{1}{2}\boldsymbol{W}(\tilde{\boldsymbol{q}})\boldsymbol{e}_\omega, \quad \boldsymbol{W}(\tilde{\boldsymbol{q}}) = \begin{bmatrix} -\tilde{\boldsymbol{q}}_v^\top \\ \tilde{q}_0\boldsymbol{I} + \boldsymbol{S}(\tilde{\boldsymbol{q}}_v) \end{bmatrix}, \tag{3.25}$$

with $\boldsymbol{W}(\tilde{\boldsymbol{q}})^\top\boldsymbol{W}(\tilde{\boldsymbol{q}}) = \boldsymbol{I}_{3\times3}$. We augment the quaternion attitude error with $\tilde{q}_0 - 1$. Similarly to (3.19), we design a manifold on which error will converge exponentially:

$$\tilde{\boldsymbol{\omega}} = 2\boldsymbol{W}(\tilde{\boldsymbol{q}})^\top\left(\begin{bmatrix}\dot{\tilde{q}}_0 \\ \dot{\tilde{\boldsymbol{q}}}_v\end{bmatrix} + \boldsymbol{W}(\tilde{\boldsymbol{q}})\boldsymbol{\Lambda}_\omega\boldsymbol{W}(\tilde{\boldsymbol{q}})^\top\begin{bmatrix}\tilde{q}_0 - 1 \\ \tilde{\boldsymbol{q}}_v\end{bmatrix}\right), \tag{3.26}$$

where $\boldsymbol{\Lambda}_\omega$ is a 3×3 positive definite gain matrix. Suppose $\boldsymbol{\Lambda}_\omega$ has positive eigenvalues $\{\lambda_1, \lambda_2, \lambda_3\}$, then $\boldsymbol{W}(\tilde{\boldsymbol{q}})\boldsymbol{\Lambda}_\omega \boldsymbol{W}(\tilde{\boldsymbol{q}})^\top$ will have eigenvalues $\{0, \lambda_1, \lambda_2, \lambda_3\}$. Since the fourth element of quaternion is redundant, it is expected that $\tilde{\boldsymbol{q}}_v \to \boldsymbol{0}$ exponentially. It is also easy to verify that

$$\boldsymbol{W}(\tilde{\boldsymbol{q}})^\top \begin{bmatrix} \tilde{q}_0 - 1 \\ \tilde{\boldsymbol{q}}_v \end{bmatrix} = \tilde{\boldsymbol{q}}_v. \tag{3.27}$$

Our attitude control law is inspired by [99] with an additional back-stepping term.

Proposition 3.2. *Suppose the control law is defined as*

$$\bar{\boldsymbol{\tau}} = \boldsymbol{J}\dot{\boldsymbol{\omega}}_r - \boldsymbol{S}(\boldsymbol{J}\boldsymbol{\omega})\boldsymbol{\omega}_r - \boldsymbol{K}_\omega \tilde{\boldsymbol{\omega}} - k_q \tilde{\boldsymbol{q}}_v, \tag{3.28}$$

$$\textit{with} \quad \tilde{\boldsymbol{\omega}} = \boldsymbol{\omega} - \boldsymbol{\omega}_r, \quad \boldsymbol{\omega}_r = \tilde{\boldsymbol{R}}\boldsymbol{\omega}_d - 2\boldsymbol{\Lambda}_\omega \tilde{\boldsymbol{q}}_v \tag{3.29}$$

where $\boldsymbol{K}_\omega$ is a positive definite matrix and $k_q > 0$. Suppose the difference between $\bar{\boldsymbol{\tau}}$ and its realization is $\tilde{\boldsymbol{\tau}}$. If it is bounded by some $\|\tilde{\boldsymbol{\tau}}\| \le \epsilon$, then $\tilde{\boldsymbol{q}}_v(t) \to \boldsymbol{0}$ and $\boldsymbol{\omega} \to \boldsymbol{\omega}_r$ exponentially within b_r controlled by ϵ and the gain matrices $\boldsymbol{\Lambda}_q$, k_q, and $\boldsymbol{K}_\omega$.

Proof. Select the candidate Lyapunov function as

$$\mathcal{V}(\tilde{\boldsymbol{\omega}}, \tilde{\boldsymbol{q}}_v) = (1/2)\tilde{\boldsymbol{\omega}}^\top \boldsymbol{J}\tilde{\boldsymbol{\omega}} + k_q \tilde{\boldsymbol{q}}_v^\top \tilde{\boldsymbol{q}}_v + k_q(\tilde{q}_0 - 1)^2.$$

$\tilde{q}_0 = \sqrt{1 - \|\tilde{\boldsymbol{q}}_v\|^2}$ is a redundant variable here. Combining dynamics from (3.2) and controller (3.28), we get the closed-loop dynamics for $\tilde{\boldsymbol{\omega}}$ as

$$\boldsymbol{J}\dot{\tilde{\boldsymbol{\omega}}} + \Big(\boldsymbol{K}_\omega - \boldsymbol{S}(\boldsymbol{J}\boldsymbol{\omega})\Big)\tilde{\boldsymbol{\omega}} + k_q \tilde{\boldsymbol{q}}_v = \tilde{\boldsymbol{\tau}}. \tag{3.30}$$

Using (3.25), (3.29), and (3.30), we compute the time derivative of Lyapunov function as,

$$\begin{aligned}
\dot{\mathcal{V}} &= \tilde{\boldsymbol{\omega}}^\top \boldsymbol{J}\dot{\tilde{\boldsymbol{\omega}}} + 2k_q \tilde{\boldsymbol{q}}_v^\top \dot{\tilde{\boldsymbol{q}}}_v + 2k_q(\tilde{q}_0 - 1)\dot{\tilde{q}}_0 \\
&= \tilde{\boldsymbol{\omega}}^\top \big(\boldsymbol{S}(\boldsymbol{J}\boldsymbol{\omega}) - \boldsymbol{K}_\omega\big)\tilde{\boldsymbol{\omega}} - k_q \tilde{\boldsymbol{\omega}}^\top \tilde{\boldsymbol{q}}_v + \tilde{\boldsymbol{\omega}}^\top \tilde{\boldsymbol{\tau}} \\
&\qquad + k_q \tilde{\boldsymbol{q}}_v^\top \big(\tilde{q}_0 \boldsymbol{I}_{3\times3} + \boldsymbol{S}(\tilde{\boldsymbol{q}}_v)\big)\boldsymbol{e}_\omega - k_q(\tilde{q}_0 - 1)\tilde{\boldsymbol{q}}_v^\top \boldsymbol{e}_\omega \\
&= -\tilde{\boldsymbol{\omega}}^\top \boldsymbol{K}_\omega \tilde{\boldsymbol{\omega}} - 2k_q \tilde{\boldsymbol{q}}_v^\top \boldsymbol{\Lambda}_q \tilde{\boldsymbol{q}}_v + \tilde{\boldsymbol{\omega}}^\top \tilde{\boldsymbol{\tau}}.
\end{aligned}$$

By letting $\boldsymbol{\eta} = [\tilde{\boldsymbol{\omega}}^\top, \tilde{\boldsymbol{q}}_v^\top]^\top$ and using the comparison method from [95], we get

$$\|\boldsymbol{\eta}(t)\| \le \sqrt{\frac{c_2}{c_1}} \|\boldsymbol{\eta}(t_0)\| \exp\left(\frac{c_3}{c_2}(t - t_0)\right) + \frac{c_2}{c_1 c_3} \sup_{t \ge t_0} \|\tilde{\boldsymbol{\tau}}\|,$$

with

$$
\begin{aligned}
c_1 &= \min\{\lambda_{\min}(\boldsymbol{J}), 2k_q\} \\
c_2 &= \max\{\lambda_{\max}(\boldsymbol{J}), 4k_q\} \\
c_3 &= \min\{\lambda_{\min}(\boldsymbol{K}_\omega), 2k_q\lambda_{\min}(\boldsymbol{\Lambda}_q)\}.
\end{aligned}
$$

Since $\|\tilde{\boldsymbol{\tau}}\| \leq \epsilon$ on a compact set, $\therefore \sqrt{\|\tilde{\boldsymbol{\omega}}\|^2 + \|\tilde{\boldsymbol{q}}_v\|^2} \to b_r$ with $r = (\epsilon c_2)/(c_1 c_3)$. ∎

3.3 Force Allocation

The controllers in (3.19) and (3.28) are written in general form for any aircraft type, but each different type relies on $\bar{\boldsymbol{f}}$ and $\bar{\boldsymbol{\tau}}$ achieved through varying actuator commands or vehicle attitudes. We name such processes as force allocation [44]. The realization of $\bar{\boldsymbol{f}}$ within a small error is essential to minimizing position tracking errors.

The reference force command is achieved through a two-step process, solving for desired attitude and thrust commands, respectively. First, desired attitude $\boldsymbol{R}_d$ is obtained through

$$
\boldsymbol{R}_d \cdot \Big(\boldsymbol{f}_T(\boldsymbol{u}_T) + \boldsymbol{f}_A(\boldsymbol{v}_i)\Big)_d = \bar{\boldsymbol{f}} \tag{3.31}
$$

with $\boldsymbol{v}_i$ also a function of $\boldsymbol{R}_d$ as seen from (3.8). Second, we solve for the thrust commands by minimizing the force residue $\left\|\boldsymbol{R}\left(\boldsymbol{f}_T + \boldsymbol{f}_A\right) - \bar{\boldsymbol{f}}\right\|$ in the current frame.

Note that $\boldsymbol{f}_T$ is directly controlled by fast input dynamics $\boldsymbol{u}_T$ while $\boldsymbol{f}_A$ is dependent on both $\boldsymbol{R}$ and $\boldsymbol{v}_i$. For a forward flying VTOL with wings, we intend to utilize as much wing-lift as possible. Thus $\boldsymbol{f}_A$ will be prioritized to reach $\bar{\boldsymbol{f}}$, with any residue taken care of by $\boldsymbol{f}_T$.

3.3.1 Desired Attitude Determination

A solution to (3.31) for a multicopter relies on the differential flatness property, with $\boldsymbol{f}_A = 0$ or considering only the drag effect [52], [100]. For a fixed-wing aircraft, a similar property can be derived by constraining the vehicle in the coordinated flight frame [101]. We propose a simplified solution following the same procedure in [44] for fast and slow flight speeds.

Proposition 3.3 (Low-Speed *Force Allocation*)**.** *Suppose $\hat{\boldsymbol{k}}_f$ is a unit vector in the body frame indicating the favorable thruster force direction. Then we let*

$$
\boldsymbol{f}_T(\boldsymbol{u}_T) = \left(\boldsymbol{R}^\top \bar{\boldsymbol{f}} - \boldsymbol{f}_A(\boldsymbol{v}_i)\right) \cdot \hat{\boldsymbol{k}}_f, \tag{3.32}
$$

as the current thruster force output. $\boldsymbol{R}_d$ *is obtained as any rotation that satisfies*

$$\boldsymbol{R}_d \hat{\boldsymbol{k}}_f = \frac{\bar{\boldsymbol{f}}}{\|\bar{\boldsymbol{f}}\|}. \tag{3.33}$$

To fully determine $\boldsymbol{R}_d$*, we can constrain the body x-axis to be in certain pre-defined heading direction as normally done for multirotors [100], [102].*

For high-speed flight, we limit $\beta = 0$ and α to be small within a linear region of the aerodynamic model, as commonly done for fixed-wing aircraft.

Proposition 3.4 (High-Speed *Force Allocation*)**.** *Let* $\bar{\boldsymbol{f}}_b = \boldsymbol{R}^\top \bar{\boldsymbol{f}}$ *denote the desired body frame force. Given measured* $\boldsymbol{v}_i$*, the incident wind frame axes are defined as*

$$\hat{\boldsymbol{x}}_i = \frac{\boldsymbol{v}_i}{\|\boldsymbol{v}_i\|}, \quad \hat{\boldsymbol{y}}_i = \frac{\boldsymbol{v}_i \times \bar{\boldsymbol{f}}_b}{\|\boldsymbol{v}_i \times \bar{\boldsymbol{f}}_b\|}, \quad \hat{\boldsymbol{z}}_i = \hat{\boldsymbol{x}}_i \times \hat{\boldsymbol{y}}_i.$$

Incident wind frame is then represented in body frame as

$$\boldsymbol{R}_i = [\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{y}}_i, \hat{\boldsymbol{z}}_i],$$

in which no side force is required (i.e $\bar{\boldsymbol{f}}_b \cdot \hat{\boldsymbol{y}}_i = 0$*). Wing lift is then prioritized to produce* $\bar{\boldsymbol{f}}_b$ *normal to* $\boldsymbol{v}_i$ *by setting a desired angle of attack* α_d *according to the lift model*

$$\alpha_d = \begin{cases} \frac{-\bar{\boldsymbol{f}}_b \cdot \hat{\boldsymbol{z}}_i - L_0}{L_1} & \textit{if} \quad -\bar{\boldsymbol{f}}_b \cdot \hat{\boldsymbol{z}}_i \leq L_{\max} \\ \alpha_{\max} & \textit{if} \quad -\bar{\boldsymbol{f}}_b \cdot \hat{\boldsymbol{z}}_i > L_{\max} \end{cases} \tag{3.34}$$

where $L_0 = \frac{1}{2}\rho V_\infty^2 S_{ref} C_{L_0}$ *and* $L_1 = \frac{1}{2}\rho V_\infty^2 S_{ref} C_{L_1}$*. The incident wind frame is then rotated around* $\hat{\boldsymbol{y}}_i$ *by* α_d *denoted by rotation matrix* $\boldsymbol{R}_\alpha$*. The desired attitude can be calculated via consecutive rotations*

$$\boldsymbol{R}_d = \boldsymbol{R}\boldsymbol{R}_i\boldsymbol{R}_\alpha. \tag{3.35}$$

The small angle approximation is often used implicitly for the fixed-wing aircraft control problems. In contrast, our approach explicitly uses the lift model in (3.12) to output the desired attitude, and relies on accurate estimation of the incident velocity $\boldsymbol{v}_i$ with a dedicated sensor described in Section 4.3.

3.3.2 Thruster Force Allocation

At the current attitude, we calculate thruster force through

$$\boldsymbol{M}\boldsymbol{f}_T = \boldsymbol{M}\left(\boldsymbol{R}^\top \bar{\boldsymbol{f}} - \boldsymbol{f}_A(\boldsymbol{v}_i)\right), \tag{3.36}$$

where $\boldsymbol{M}$ is the matrix masking out the component of force in the body axis that is unable to achieve by thruster input. $\boldsymbol{f}_A$ is estimated from (3.9) and (3.13) using current $\boldsymbol{v}_i$ estimation. Thus the desired thruster forces can be solved easily.

3.3.3 Boundedness of Force Error

Before proving the boundedness of $\tilde{\boldsymbol{f}}$, we state the following assumptions:

Assumption 3.1. *The coefficients C_L, C_Y, and C_D from (3.12) admit a lower bounded error ϵ_1 on a linear model than ϵ_2 on a full-range model.*

Assumption 3.2. *Vehicle contained entirely in its flight envelope. The attitude error satisfies $\|\tilde{\boldsymbol{q}}_v\| \leq \sigma_1$ globally. Within the linear aerodynamic region, $\|\tilde{\boldsymbol{q}}_v\| \leq \sigma_1 < \sigma_1$ determined by cruise condition.*

Assumption 3.3. *The total thruster force $\boldsymbol{f}_T$ can be achieved arbitrarily close to a desired value.*

Theorem 3.1. *Suppose we feed $\boldsymbol{R}_d$ into the attitude controller (3.28). In quaternion form, this is $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$, from (3.32) and (3.35) using $\boldsymbol{q}_d = \mathrm{Slerp}(\boldsymbol{q}_1, \boldsymbol{q}_2; \gamma)$, an interpolation scheme for quaternions [103] with a parameter γ as a* tanh *mixing function varying with V_∞ and transition around a threshold speed V_{tr}. Assuming $V_{\max}$ is the maximum vehicle speed associated in the trajectory, the overall force allocation guarantees a bound on $\tilde{\boldsymbol{f}}$ controlled by* $\max\{2V_{tr}^2\epsilon_2, V_{\max}^2\epsilon_1\}$.

Proof. From (3.32) and (3.36), denote $\boldsymbol{f}_d = \bar{\boldsymbol{f}} - \boldsymbol{R}\boldsymbol{f}_A(\boldsymbol{v}_i)$ and $\hat{\boldsymbol{f}}_d$ as the estimated value based on approximate aerodynamics (3.12). Using Assumptions 3.1 and 3.2, it can be shown

$$\begin{aligned}\|\tilde{\boldsymbol{f}}\| &\leq \|\boldsymbol{f}_d - \hat{\boldsymbol{f}}_d \cdot \hat{\boldsymbol{k}}_f\| \leq \|\hat{\boldsymbol{f}}_d - \hat{\boldsymbol{f}}_d \cdot \hat{\boldsymbol{k}}_f\| + \frac{\rho S_{\mathrm{ref}}}{2} V_\infty^2 \epsilon_2 \\ &\leq \epsilon_3 \|\hat{\boldsymbol{f}}_d\| + \frac{\rho S_{\mathrm{ref}}}{2} V_\infty^2 \epsilon_2,\end{aligned}$$

where $\epsilon_3 < 1$ is determined by the global attitude error bound from Assumption 3.2. Similarly from (3.35), we have

$$\|\tilde{\boldsymbol{f}}\| \leq \epsilon_4 \|\hat{\boldsymbol{f}}_d\| + \frac{\rho S_{\mathrm{ref}}}{2} V_\infty^2 \epsilon_1,$$

where $\epsilon_4 < \epsilon_3$ is satisfied from small angle approximation and $\epsilon_4 \ll 1$. During the transitioning phase, when $0 < \gamma < 1$,

$$\|\tilde{\boldsymbol{f}}\| \leq \epsilon_3 \|\hat{\boldsymbol{f}}_d\| + \rho S_{\mathrm{ref}} V_{\mathrm{tr}}^2 \epsilon_2.$$

Therefore, the supremum of the norm $\|\tilde{\boldsymbol{f}}\|$ is bounded as follows

$$\sup_{t \geq t_0} \|\tilde{\boldsymbol{f}}\| \leq \epsilon_3 \|\hat{\boldsymbol{f}}_d\| + \frac{\rho S_{\mathrm{ref}}}{2} \max\{2V_{\mathrm{tr}}^2 \epsilon_2,\ V_{\max}^2 \epsilon_1\}. \tag{3.37}$$

■

It is interesting to note the intuition behind this bound. $\epsilon_3 \left\|\hat{\boldsymbol{f}}_d\right\|$ is limited by desired force and attitude, respectively, which is achieved internally inside the controller. The first term in $\max\{\cdot\}$ represents a bound with the full-range aerodynamic model having larger uncertainties, limited by V_{tr}. This is a common implicit assumption among multicopter controls. The second term in $\max\{\cdot\}$ represents a tighter bound from the high accuracy linear model.

3.4 Control Allocation

From (3.28) and (3.36), we eventually arrive at the required thruster forces and moments

$$\begin{bmatrix} \boldsymbol{f}_T \\ \boldsymbol{\tau}_T \end{bmatrix} = \begin{bmatrix} \boldsymbol{M}\left(\boldsymbol{R}^\top \bar{\boldsymbol{f}} - \boldsymbol{f}_A(\boldsymbol{v}_i)\right) \\ \bar{\boldsymbol{\tau}} - \boldsymbol{B}_A \boldsymbol{u}_A \end{bmatrix}. \tag{3.38}$$

We can redefine wrench vector $\boldsymbol{w}_T$ as a combination of thruster forces and moments. Given a feasible control space $\mathcal{U}_T \subset \mathbb{R}^{N_T}$ of the distributed propulsion, control allocation aims to find $\boldsymbol{u}_T \in \mathcal{U}_T$ such that

$$\boldsymbol{w}_T = \boldsymbol{B}_T \boldsymbol{u}_T \quad \text{with} \quad \boldsymbol{w}_T = \begin{bmatrix} \boldsymbol{f}_T \\ \boldsymbol{\tau}_T \end{bmatrix} \in \mathbb{R}^{N_w}. \tag{3.39}$$

However, [104] shows that there exists no single inverse mapping that can recover a full set of feasible $\boldsymbol{w}_T$ without violating some control constraints for an over-actuated system when $N_T > N_w$. Nonetheless, since $\boldsymbol{B}_T$ has multiple right inverses, there still exist more than one $\boldsymbol{B}_T$ satisfying (3.39).

Control allocation in practice is executed with very high frequency, thus we would like to validate (3.39) within the same time-scale of the controllers. We achieve this by pre-computing a full attainable wrench space $\mathcal{W}_T \subset \mathbb{R}^{N_w}$ for all elements in which it is guaranteed to find $\boldsymbol{u}_T \in \mathcal{U}_T$ under some inverse mapping of $\boldsymbol{B}_T$. If $\boldsymbol{w}_T \in \mathcal{W}_T$, then a method proposed in Section 3.4.2 is used to solve for $\boldsymbol{u}_T$. Otherwise, the magnitude of $\boldsymbol{w}_T$ may have to be scaled down until it becomes an element of $\mathcal{W}_T$ with its direction maintained, as is commonly done in the literature [104].

3.4.1 Attainable Thruster Wrench Space

Let $\mathcal{U}_T = \{\boldsymbol{u}_T \mid \boldsymbol{u}_{T,\min} \le \boldsymbol{u}_T \le \boldsymbol{u}_{T,\max}\}$ and its boundary $\partial\mathcal{U}_T = \{\boldsymbol{u}_{T,\min}, \boldsymbol{u}_{T,\max}\}$. A feasible wrench $\boldsymbol{w}_T$ lies under the column space of $\boldsymbol{B}_T$ with $\boldsymbol{u}_T \in \mathcal{U}_T$. We define the attainable wrench space as $\mathcal{W}_T = \{\boldsymbol{w}_T = \sum_{j=1}^{N_T} u_{T,j}\boldsymbol{b}_j \mid u_{T,\min} \le u_{T,j} \le u_{T,\max}, \forall j\}$, where $u_{T,j}$ and $\boldsymbol{b}_j$ represent the j-th element and the column of $\boldsymbol{u}_T$ and $\boldsymbol{B}_T$, respectively. Without loss of generality, let us assume that the individual thrusters are identical, with $u_{T,\min} = 0$ and $u_{T,\max} = 1$. Then, by definition

$$\mathcal{W}_T = \mathrm{Conv}\left(\bigoplus_{j=1}^{N_T}\{\mathbf{0}, \boldsymbol{b}_j\}\right), \tag{3.40}$$

where Conv$(\cdot)$ represents a convex hull of a given set and $\bigoplus$ is the Minkowski sum. We check whether $\boldsymbol{w}_T \in \mathcal{W}_T$ can be done via linear programming [105]. Even though linear programs run reasonably fast on modern computers, we argue that for many low-power control units on aerial vehicles, the method still poses difficulty. In such a case, we propose that $\mathcal{W}_T$ may be approximated by a hypercuboid $\bar{\mathcal{W}}_T$ that is maximally fitted within $\mathcal{W}_T$

$$\bar{\mathcal{W}}_T = \{\boldsymbol{w}_T \,|\, \boldsymbol{w}_{T,\min} \leq \boldsymbol{w}_T \leq \boldsymbol{w}_{T,\max}, \} \subset \mathcal{W}_T, \tag{3.41}$$

where $\boldsymbol{w}_{T,\min}$ and $\boldsymbol{w}_{T,\max}$ are constants that define the boundary of hypercuboid. Although there is a potential loss of space in $\mathcal{W}_T$, checking $\boldsymbol{w}_T \in \bar{\mathcal{W}}_T$ will be much faster than that of $\boldsymbol{w}_T \in \mathcal{W}_T$, involving only a finite number of comparison operations.

In order to find the reduced wrench space $\bar{\mathcal{W}}_T$, we recall that $\mathcal{W}_t$ is a convex hull constructed with a finite set of points, and thus it is an N_w-dimensional polyhedron that can be written as $\mathcal{W}_t = \{\boldsymbol{w}_T \,|\, \mathbf{A}\boldsymbol{w}_T \leq \mathbf{c}, \mathbf{A} = [a_{ij}] \in \mathbb{R}^{p\times N_w}, \mathbf{c} = [c_i] \in \mathbb{R}^p\}$, where p is the number of polyhedron faces. We assume that the edge lengths of $\bar{\mathcal{W}}_T$ are $\phi(\bar{v}_i - \underline{v}_i)$, where ϕ is a scaling constant to be optimized and $\bar{v}_i$ and $\underline{v}_i$ are given coordinates of an edge of $\bar{\mathcal{W}}_T$ along the i-th dimension. Then the following optimization problem solves for a maximum ϕ, which is equivalent to finding $\bar{\mathcal{W}}_T$ maximally fitted in $\mathcal{W}_T$:

$$\begin{aligned} \phi^* = \ & \underset{\phi}{\mathrm{argmax}} \sum_{i=1}^{m} \log\big(\phi(\bar{v}_i - \underline{v}_i)\big), \qquad (3.42)\\ \text{subject to} \quad & a_{ij}^+ = \max\{a_{ij}, 0\}, \quad a_{ij}^- = \max\{-a_{ij}, 0\}, \\ & \sum_{j=1}^{m} a_{ij}^+\bar{v}_j - a_{ij}^-\underline{v}_j \leq c_i, \quad i = 1, \cdots, p, \\ & \underline{v}_k < \bar{v}_k, \quad k = 1, \cdots, m. \end{aligned}$$

3.4.2 Static and Recursive Control Allocation

A right pseudoinverse $\boldsymbol{B}_T^+ = \boldsymbol{B}_T^\top(\boldsymbol{B}_T\boldsymbol{B}_T^\top)^{-1}$ of $\boldsymbol{B}_T$ is a common saturation-prune method for control allocation often considered in the literature and practice. A straightforward alternative is to solve an optimization,

$$\begin{aligned} \min_{\boldsymbol{u}_T} \quad & \|\boldsymbol{u}_T\|_2, \\ \text{subject to} \quad & \boldsymbol{B}_T \boldsymbol{u}_T = \boldsymbol{w}_T, \\ & \boldsymbol{u}_{T,\min} \le \boldsymbol{u}_T \le \boldsymbol{u}_{T,\max}, \end{aligned} \tag{3.43}$$

which we deemed it not ideal for a real-time application for the abovementioned practical reasons. Note that $\boldsymbol{u}_T$ computed with the right pseudoinverse is a solution to (3.43) if the inequality constraint is ignored. Therefore, the method of cascading inverse [106] can be promising in computing $\boldsymbol{u}_T \in \mathcal{U}_T$, which computes pseudoinverse or generalized inverse solutions in an iterative manner.

Algorithm 1 Recursive Control Allocation

```
Input: Control effective matrix $\boldsymbol{B}_T$, desired wrench $\boldsymbol{w}_T$.
Output: Distributed propulsion control input $\boldsymbol{u}_T \in \mathcal{U}_T$.
 1: while A new wrench vector $\boldsymbol{w}_T$ is available do
 2:     if $\boldsymbol{w}_T \notin \mathcal{W}_T$ or $\boldsymbol{w}_T \notin \bar{\mathcal{W}}_T$ then
 3:         Scale down $\boldsymbol{w}_T$ until $\boldsymbol{w}_T \in \mathcal{W}_T$ or $\boldsymbol{w}_T \in \bar{\mathcal{W}}_T$.
 4:     end if
 5:     if $\boldsymbol{B}_T$ is square then
 6:         $\boldsymbol{u}_T = \boldsymbol{B}_T^{-1}\boldsymbol{w}_T$.
 7:     else
 8:         $\boldsymbol{u}_T = \boldsymbol{B}_T^\top(\boldsymbol{B}_T\boldsymbol{B}_T^\top)^{-1}\boldsymbol{w}_T$.
 9:         if $\exists j$ such that $u_{T,\max} < u_{T,j}$ then
10:             $\boldsymbol{w}_T \leftarrow \boldsymbol{w}_T - u_{T,\max}\boldsymbol{b}_j$.
11:             $\boldsymbol{B}_T \leftarrow \left[\boldsymbol{b}_1, \cdots, \boldsymbol{b}_{j-1}, \boldsymbol{b}_{j+1}, \cdots, \boldsymbol{b}_n\right]$.
12:             $\boldsymbol{u}_T$ = Recursive Control Allocation$(\boldsymbol{B}_T, \boldsymbol{w}_T)$.
13:             Insert $u_{T,\max}$ as $j$-th element of $\boldsymbol{u}_T$.
14:         else if $\exists j$ such that $u_{T,j} < u_{T,\min}$ then
15:             $\boldsymbol{w}_T \leftarrow \boldsymbol{w}_T - u_{T,\min}\boldsymbol{b}_j$.
16:             $\boldsymbol{B}_T \leftarrow \left[\boldsymbol{b}_1, \cdots, \boldsymbol{b}_{j-1}, \boldsymbol{b}_{j+1}, \cdots, \boldsymbol{b}_n\right]$.
17:             $\boldsymbol{u}_T$ = Recursive Control Allocation$(\boldsymbol{B}_T, \boldsymbol{w}_T)$.
18:             Insert $u_{T,\min}$ as $j$-th element of $\boldsymbol{u}_T$.
19:         end if
20:     end if
21:     return $\boldsymbol{u}_T$.
22: end while
```

When a pseudoinverse solution does not violate the constraints on $\boldsymbol{u}_T$, it is admitted as a solution. Otherwise, the saturated element $u_{t,j}$ of $\boldsymbol{u}_T$ is either set to $u_{t,\max}$ if $u_{t,j} > u_{t,\max}$, or $u_{t,\min}$ if $u_{t,j} < u_{t,\min}$. This action determines the contribution of the saturated element $u_{t,j}$ to the wrench $\boldsymbol{w}_T$ as either $u_{t,\max}\boldsymbol{b}_j$ or $u_{t,\min}\boldsymbol{b}_j$. The wrench is updated by subtracting this amount, and the j-th column $\boldsymbol{b}_j$ of $\boldsymbol{B}_T$ is removed from the matrix because its contribution to $\boldsymbol{w}_T$ is already considered. Then Algorithm 1 is executed recursively to solve for the remaining elements of $\boldsymbol{u}_T$ with the updated $\boldsymbol{w}_T$ and $\boldsymbol{B}_T$ until a non-saturated solution is found. For each given $\boldsymbol{w}_T$, the method runs at most $(N_T - N_w)$ iterations until $\boldsymbol{B}_T$ reduces to a square matrix, and each iteration involves one pseudoinverse (or inverse in the last iteration) operation. Therefore, the worst case time complexity of the algorithm is $O(N_T - N_w)$.

For flying aircraft with distributed propulsion, a major portion of control effort goes to lift generation to counter-balance the weight. Based on this observation, we propose an allocation method that computes an inverse mapping $\boldsymbol{B}'_T$ of $\boldsymbol{B}_T$ that avoids saturation of $\boldsymbol{u}_T$ by evenly distributing the effort to generate lift across the thrusters. Furthermore, the allocation prevents thrusters from generating downward forces. The resultant $\boldsymbol{B}'_T$ can be implemented onboard as a static mapping and its execution speed would be much faster than any online optimization method.

By definition, $\boldsymbol{B}'_T\boldsymbol{w}_T = \boldsymbol{u}_T$. The energy consumption, $\|\boldsymbol{u}_T\|_2^2 = (\boldsymbol{w}_T)^\top(\boldsymbol{B}'_T)^\top\boldsymbol{B}'_T\boldsymbol{w}_T$, is minimized for a given feasible $\boldsymbol{w}_T$ by minimizing the Frobenius norm of $\boldsymbol{B}'_T$,

$$\|\boldsymbol{B}'_T\|_F = \sqrt{\mathrm{tr}(\boldsymbol{B}'^\top_T\boldsymbol{B}'_T)}.$$

The following optimization is formulated, which not only minimizes the energy consumption, but also guarantees evenly distributed lift generation from all thrusters:

$$\begin{aligned} \min_{\boldsymbol{B}'_T} \quad & \|\boldsymbol{B}'_T\|_F + f_{\mathrm{aad}}(\boldsymbol{b}'), \\ \text{subject to} \quad & \boldsymbol{B}_T\boldsymbol{B}'_T = I, \\ & \boldsymbol{b}' \leq \boldsymbol{0}, \end{aligned} \tag{3.44}$$

where $\boldsymbol{b}'$ corresponds to the column of $\boldsymbol{B}'_T$ governing the allocation of a vertical force. Note that the positive body z-direction is downwards in Figure 3.1, and to generate an upward vertical lift, it is desirable to have all elements of $\boldsymbol{b}'$ less than zero. In addition, to avoid any individual rotor reaching maximum speed, we distribute the vertical force evenly across thrusters. In particular, the difference between individual thruster forces are minimized using the average absolute deviation $f_{\mathrm{aad}}(\boldsymbol{b}')$ about the mean of $\boldsymbol{b}'$, where

$$f_{\mathrm{aad}}(\boldsymbol{b}') = \frac{1}{N_T}\sum_{i=1}^{N_T}|b'_i - \mathrm{mean}(\boldsymbol{b}')|.$$

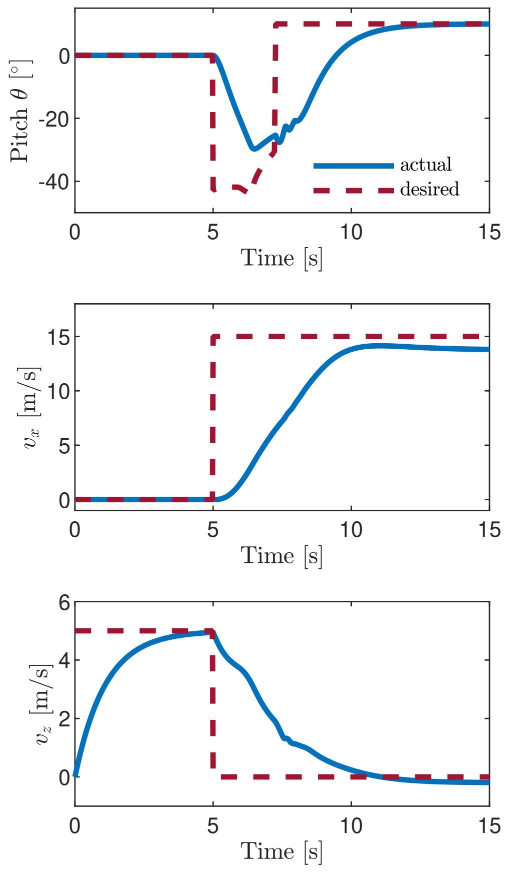
Note that (3.44) is convex and can be easily solved through a convex optimization solver.

3.5 Results on Tilt-Rotor Fixed-wing VTOL

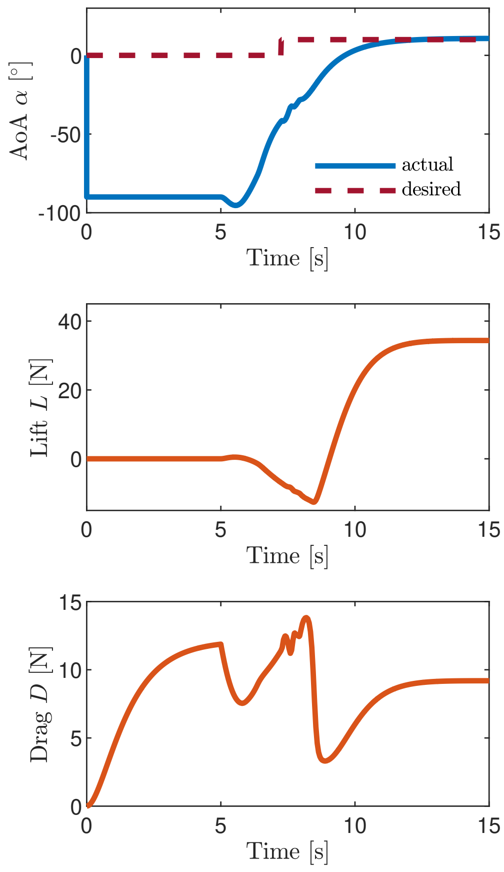
We model a vehicle similar to Figure 3.1 with tiltable rear thrusters to demonstrate as test platform for the overall control architecture explained in this chapter. The physical details of the test vehicles are described in Appendix B.1.

3.5.1 Flight Simulations of Take-off, Transition, and Cruise

Given crude velocity commands without any sophisticated trajectory design. Figure 3.4a shows the vehicle's longitudinal states trajectory. Starting from rest, the vehicle is commanded with a positive climbing speed $v_z = 5\text{m/s}$ from $t = 0\,\text{s}$ to $t = 5\,\text{s}$. Starting at $t = 5\text{s}$, a forward cruising speed $v_x = 15\text{m/s}$ is given. Lift, drag, and angle-of-attack are shown in Figure 3.4b. The transitional behavior of the vehicle can be clearly seen. During the slow climbing stage, the *force allocator* will use upward force from vertical thrusters. Starting forward flight, the vehicle will first pitch forward as a conventional multicopter. After gaining some forward speed, the controller uses positive lift production from the wings during fast forward flight.



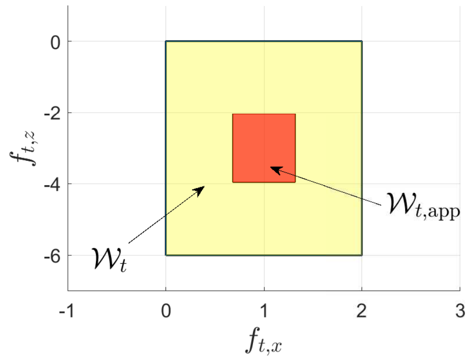
(a) Velocity and Pitch



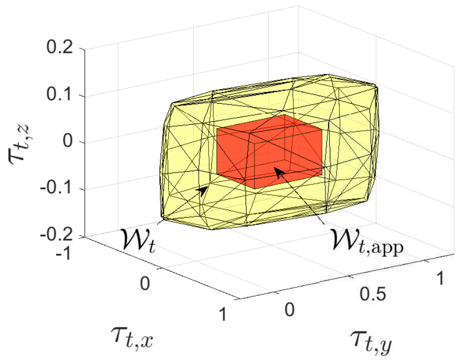
(b) α and Aerodynamic Forces

Figure 3.4: State and commands related to a transition-to-cruise trajectory.

3.5.2 Numerical Simulations of Control Allocation Methods



(a) 2D force space $\mathcal{F}_t$



(b) 3D moment space $\mathcal{M}_t$

Figure 3.5: Projections of the attainable wrench space $\mathcal{W}_T$ and its approximation $\bar{\mathcal{W}}_T$ of the vehicle in Figure 3.1 onto (a) 2D force space and (b) 3D moment space. The force space is two-dimensional because the vehicle does not have an ability to generate thrust in its y-axis.

Figure 3.5 shows the five-dimensional spaces $\mathcal{W}_T$ and $\bar{\mathcal{W}}_T$ of the vehicle in Figure 3.1 projected onto a two-dimensional force space and three-dimensional moment space for visualization. Note that the vehicle does not have an ability to generate a thrust in its y-axis, hence the force space is only two-dimensional.

Table 3.1: Comparison of pseudoinverse (Pinv), optimization (Opt), and recursive control allocation (RCA) methods.

	Pinv	Opt	RCA
% of $\boldsymbol{u}_T \in \partial\mathcal{U}$	10.8%	0%	0%
$\bar{t}/\bar{t}_{\text{pinv}}$	1	7292	1.16
$\text{cost}/\text{cost}_{\text{opt}}$	0.9995	1	1

A total of 100,000 $\boldsymbol{w}_T$ from $\bar{\mathcal{W}}_T$ were sampled, and their associated $\boldsymbol{u}_T$ were computed using the conventional pseudoinverse, cascading inverse, and optimization (3.43). The result is presented in Table 3.1, where we show the percentage of $\boldsymbol{u}_T \notin \mathcal{U}_T$, average time spent to compute each $\boldsymbol{u}_T$ normalized by the average time taken by the pseudoinverse approach ($\bar{t}$ / $\bar{t}_{\text{pinv}}$), and the average cost of the obtained $\boldsymbol{u}_T$ normalized by the average cost of the optimal solutions obtained by solving (3.43) ($\text{cost}/\text{cost}_{\text{opt}}$). While the conventional pseudoinverse approach returned $\boldsymbol{u}_T \notin \mathcal{U}_T$ for 10.8% of the sampled wrenches, the cascading inverse method returned no saturated control commands for all the test cases, with only about 20% increase in average computation time. The optimization also returned no $\boldsymbol{u}_T \notin \mathcal{U}_T$, but took much longer to find solutions. In addition, the costs of $\boldsymbol{u}_T$ obtained by the cascading inverse were nearly the same as the optimal costs obtained by solving (3.43).

3.5.3 Flight Experiments on AFA

We conducted flight experiments on AFA 1.0 in Caltech CAST's fan array wind tunnel as shown in Figure 3.6. The wind tunnel can generate a wind field up to 17 m/s and should sustain the fast flight of our vehicle.

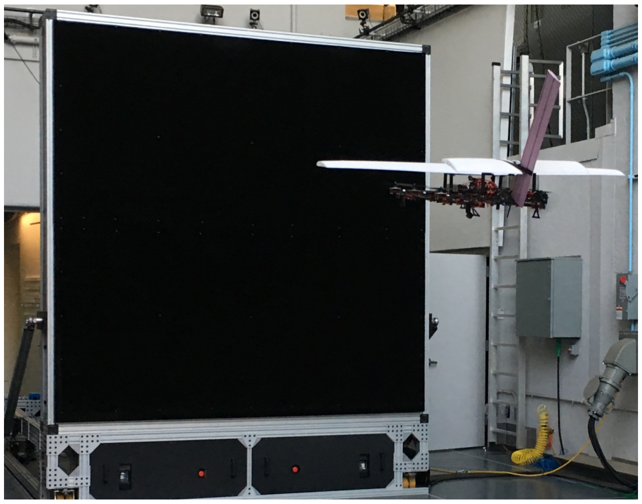
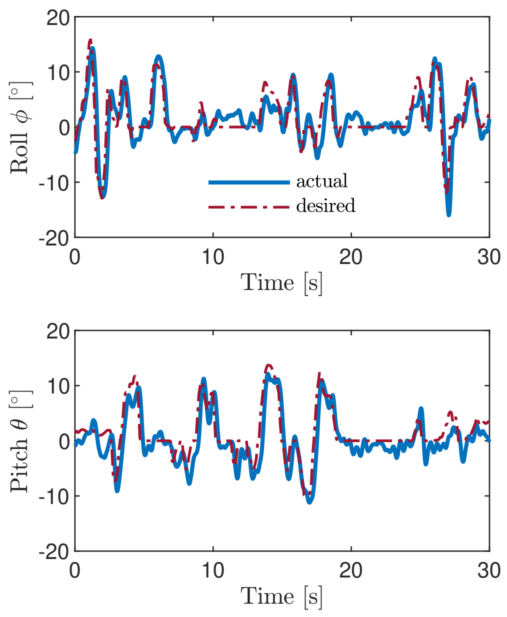


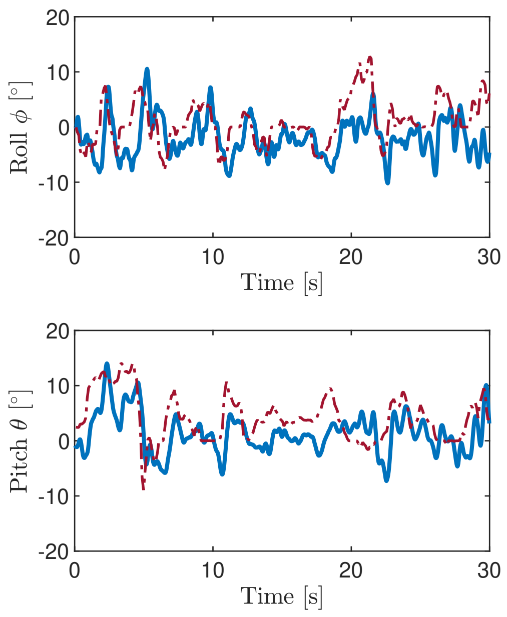
Figure 3.6: AFA 1.0 flying in wind field generated by a fan-array wind tunnel.

We implemented our controllers and force allocation method from Section 3.2, as well as the control allocation scheme in (3.44), using modified PX4 firmware and Pixhawk flight controller [107] on our functional prototype shown in Figure B.1. The prototype had the same configuration in terms of thrusters and wing compared to Appendix B.1. Tests on attitude tracking performance were performed. We conducted experiments with and without uniform wind field generated by the fan-array wind tunnel.

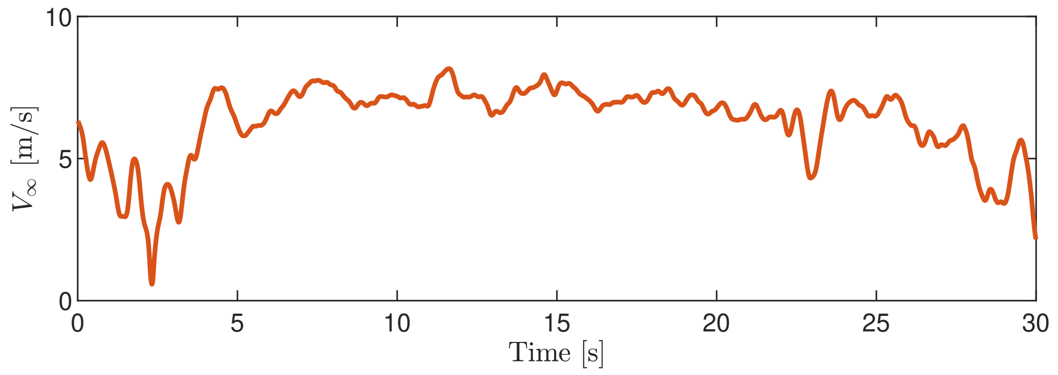
The vehicle has asymmetric rotor placement and aerodynamic surfaces of significant size. Even during normal free flight, we observed noticeable non-vanishing disturbances from wing, ground, and rotor interactions. In spite of that, our proposed controller was able to swiftly track the attitude trajectory given by an operator, as shown in Figure 3.7a. During the fan-array wind tunnel test shown in Figure 3.6, the vehicle was switched to a forward flying mode with a tilted back rotor. Tracking error increased due to a larger wind disturbance, but the controller still maintained stability. The onboard pitot-tube measured an airspeed up to 8 m/s.



(a) Attitude without wind



(b) Attitude with steady wind



(c) Measured airspeed from onboard pitot-tube during fan-array test

Figure 3.7: State and commands related to a transition-to-cruise trajectory.

3.6 Chapter Summary

We presented a novel framework for designing controllers for a VTOL aircraft with wing. The control design objective was split into: (a) designing nonlinear position/velocity and attitude/rate controllers using net forces and moments as input; and (b) force, moment, and control allocations to generate the desired wrench. We identified the problem of force allocation as complex and vital to a winged VTOL aircraft under substantial aerodynamic forces, and proposed a general solution for closed-loop flight in varying speed regimes. Furthermore, the attainable force, moments, and control spaces were analyzed to give a realtime verifiable set to avoid control saturation. A prototype hybrid VTOL vehicle was constructed. Realistic data were used to model and simulate the behavior of the proposed framework in commanding a transitional maneuver of the vehicle without a explicit trajectory design. Experiments were carried out to demonstrate the stability and robustness of the proposed controller and allocation method.

Chapter 4

PHYSICS-BASED MODEL ADAPTIVE FLIGHT CONTROL

The proposed unified control architecture, force allocation, and control allocation schemes from Chapter 3 improve the performance of fixed-wing VTOL in all flight stages. However, the accuracy of force tracking depends highly on the validity of the force model being used, as well as a precise estimate of induced wind velocity $\boldsymbol{v}_i$. Thus in this chapter, adapted from [108], we develop adaptive methods to update force models for any unmodeled transient and unsteady behaviors; and incorporate novel sensors to have realtime measurements of $\boldsymbol{v}_i$.

4.1 Linear-in-Parameter Force Model

The general dynamics for fixed-wing VTOL is described in great detail in Chapter 3. The important point for us to note is that the body force is comprised of thruster and aerodynamic forces $\boldsymbol{f}_b = \boldsymbol{f}_T + \boldsymbol{f}_A$. To further evolve the performance for postion tracking control, a more detailed and adaptation-friendly model needs to be introduced.

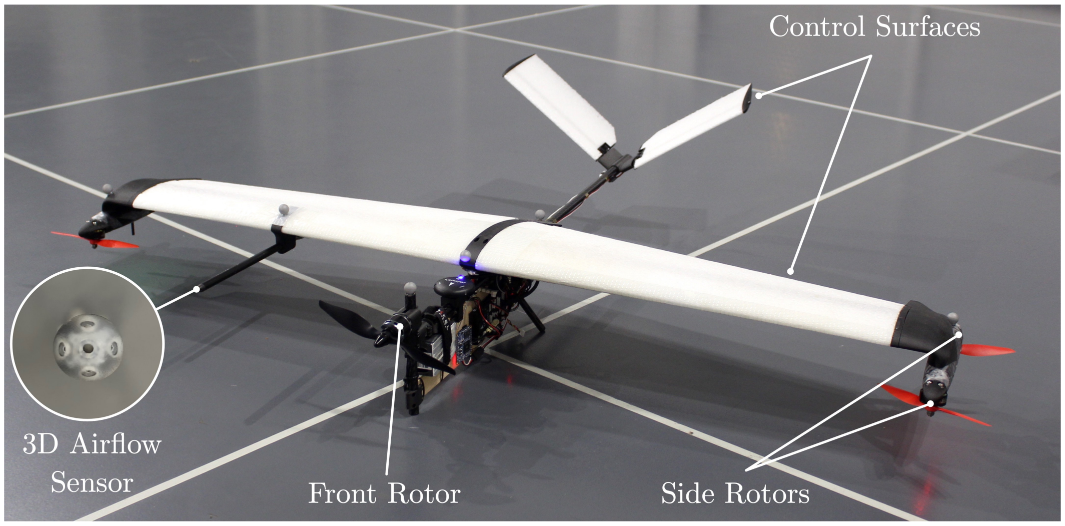
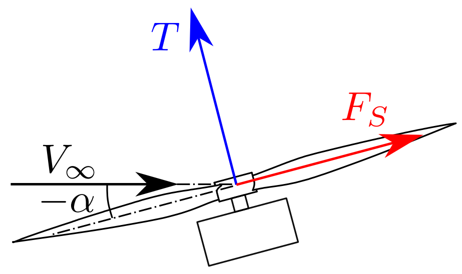
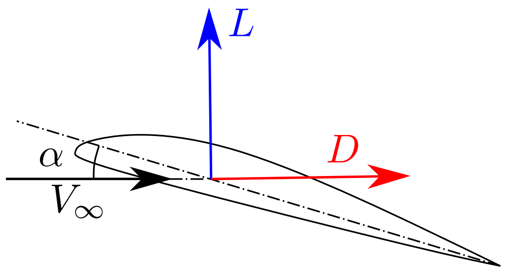


Figure 4.1: Fixed-wing VTOL prototype with copter-plane configuration.

Instead of developing a general model for any vehicle type, we will focus on the copter-plane configuration as shown in Figure 4.1. Details of the vehicle including its physical parameters are included in Appendix B.2.



(a) Propeller Force Diagram



(b) Wing Force Diagram

Figure 4.2: Diagrams with definition of forces on different propellers and wings.

As shown in Figure 4.2a, typical propellers generate an axial thrust as well as a side force tangential to the rotor plane. As stated before in (3.4), a common model for static propeller thrust is $T = C_T \rho d^4 n^2$ The rotor side force F_S however lacks simplified models, even though the measurements indicate that the side forces produced by vertical thrusters often incur a significant drag increase during forward flight. Hence, we derive a canonical model from wind tunnel test data and dimensional analysis. Details about the derivation of the model are described in Appendix A, here we include (A.1) again for completeness:

$$F_S = C_S \rho n^{k_1} V_\infty^{2-k_1} d^{2+k_1} \left(\left(\frac{\pi}{2}\right)^2 - \alpha^2 \right) (\alpha + k_2) \tag{4.1}$$

where α is in radians. By assuming a linear relationship between n and the throttle signal u, and that only the vertical thrusters produce significant side forces, we get the following force estimate

$$T_x = \bar{C}_{Tx} u_x^2, \quad T_z = \bar{C}_{Tz} u_z^2, \tag{4.2}$$

$$F_S = \bar{C}_S \cdot G(\alpha, V_\infty, u_z). \tag{4.3}$$

The coefficients $\bar{C}_{Tx}$, $\bar{C}_{Tz}$, and $\bar{C}_S$ are non-dimensional coefficients combined with constants in (3.4) and (4.1). $G(\alpha, V_\infty, u_z)$ is the basis from (4.1) using fixed k_1 and k_2 fitted from data. u_x and u_z represents the collective thrust signals in the x-axis and z-axis, respectively, which is specific to our type of fixed-wing VTOL.

The aerodynamic lift and drag on wing as shown in Figure 4.2b is the same as defined in (3.11) and (3.12). For the purpose of adaptation, we will focus on the linear range of aerodynamics forces. In practice, we will operate our vehicle in this region for a smooth flight. Thus the linear model for coefficients C_L and C_D is

$$C_L = C_{L_0} + C_{L_1} \alpha \tag{4.4}$$

$$C_D = C_{D_0} + C_{D_1} \alpha + C_{D_2} \alpha^2, \tag{4.5}$$

while the side force C_Y remains unmodeled with the assumption that $C_Y = 0$ at $\beta = 0$. This assumption is valid for most aircraft symmetric in the xz-plane. In Section 4.2, we use this fact to constrain $Y = 0$ by ensuring $\beta = 0$.

From (4.2) to (4.5), the estimated thruster and aerodynamic forces can be expressed as

$$\hat{\boldsymbol{f}}_T = \begin{bmatrix} \bar{C}_{Tx}u_x^2 \\ 0 \\ -\bar{C}_{Tz}u_z^2 \end{bmatrix} + \begin{bmatrix} -\bar{C}_S G_z(\alpha, V_\infty, u_z)\cos(\bar{\beta}) \\ -\bar{C}_S G_z(\alpha, V_\infty, u_z)\sin(\bar{\beta}) \\ 0 \end{bmatrix} \tag{4.6}$$

$$\hat{\boldsymbol{f}}_A = \frac{1}{2}\rho S_{\text{ref}} V_\infty^2 \begin{bmatrix} C_L(\alpha)\sin\alpha - C_D(\alpha)\cos\alpha \\ -C_Y(\beta) \\ -C_L(\alpha)\cos\alpha - C_D(\alpha)\sin\alpha \end{bmatrix}. \tag{4.7}$$

Note that $\bar{\beta} = \arctan\left(v_{iy}/v_{iz}\right)$ is slightly different from the sideslip. We can express the overall body force estimate

$$\hat{\boldsymbol{f}}_b = \hat{\boldsymbol{f}}_T + \hat{\boldsymbol{f}}_A = \boldsymbol{\Phi}(\boldsymbol{v}_i, u_x, u_z)\hat{\boldsymbol{\theta}}$$

where $\boldsymbol{\Phi}(\cdots)$ denotes the model basis collected from (4.4) to (4.7). The model parameter vector is

$$\hat{\boldsymbol{\theta}} = \left[\bar{C}_{Tx},\ \bar{C}_{Tz},\ \bar{C}_S,\ C_{D_0},\ C_{D_1},\ C_{D_2},\ C_{L_0},\ C_{L_1}\right]^\top.$$

4.2 Adaptive Force Allocation

4.2.1 Force Model Composite Adaptation

With reference force realized through allocation methods (3.32), (3.35), and (3.36) and position controller (3.19), the closed-loop dynamics of the velocity tracking error $\tilde{\boldsymbol{v}}$ is

$$\begin{aligned} \dot{\tilde{\boldsymbol{v}}} &= \boldsymbol{g} + \boldsymbol{R}\boldsymbol{f}_b - \dot{\boldsymbol{v}}_r + \left(\bar{\boldsymbol{f}} - \bar{\boldsymbol{f}}\right) + (\boldsymbol{R}\hat{\boldsymbol{f}}_b - \boldsymbol{R}\hat{\boldsymbol{f}}_b) \\ &= -\boldsymbol{K}_v\tilde{\boldsymbol{v}} + \left(\boldsymbol{R}\hat{\boldsymbol{f}}_b - \bar{\boldsymbol{f}}\right) - \boldsymbol{R}(\hat{\boldsymbol{f}}_b - \boldsymbol{f}_b) \\ &= -\boldsymbol{K}_v\tilde{\boldsymbol{v}} + \boldsymbol{\zeta} - \boldsymbol{R}\boldsymbol{\Phi}\tilde{\boldsymbol{\theta}}. \end{aligned} \tag{4.8}$$

The force tracking error $\boldsymbol{\zeta} = \left(\boldsymbol{R}\hat{\boldsymbol{f}}_b - \bar{\boldsymbol{f}}\right)$ is affected by the attitude difference $\tilde{\boldsymbol{R}}$ and the residual side force after projection in (3.36). Any force error in the body y-axis cannot be directly compensated for by thrusters and has to be achieved through an attitude change.

The composite adaptation technique is employed to facilitate convergence of both tracking and prediction errors [109]. We compute prediction error via a low-pass filtered onboard accelerometer measurement $\boldsymbol{a}_m = [\boldsymbol{a} - \boldsymbol{g}]_f$ and basis calculation $\boldsymbol{W}_f = [\boldsymbol{\Phi}]_f$, where $[\cdot]_f$ here is a strictly-positive-real (SPR) filter [35], [36]. Then the prediction error $\boldsymbol{e}$ is simply the difference between the two

$$\boldsymbol{e} = \boldsymbol{W}_f \hat{\boldsymbol{\theta}} - \boldsymbol{a}_m. \tag{4.9}$$

We are now ready to define tjhe parameter update law:

$$\dot{\hat{\boldsymbol{\theta}}} = \boldsymbol{P}\left(\boldsymbol{\Phi}^\top \boldsymbol{R}^\top \tilde{\boldsymbol{v}} - \boldsymbol{W}_f^\top \boldsymbol{e}\right) - \sigma \boldsymbol{P}\left(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_0\right) \tag{4.10}$$

$$\dot{\boldsymbol{P}} = -\boldsymbol{P}\boldsymbol{W}_f^\top \boldsymbol{W}_f \boldsymbol{P} + \lambda \boldsymbol{P}. \tag{4.11}$$

To improve the robustness, a damping term σ is added to pull updates toward initial estimates $\boldsymbol{\theta}_0$, which is obtained through model fitting described in Appendix B.2. λ makes (4.11) exponentially forget data for continual adaptation [35].

4.2.2 Stability Property

From (4.8), it can be seen that the tracking of the velocity profile relies on the force prediction as well as attitude control. Due to the realistic aerodynamic effects on a fixed-wing VTOL being nonlinear and unsteady, we make the following assumptions on $\boldsymbol{\zeta}$ and $\tilde{\boldsymbol{q}}_v$

Assumption 4.1. *The vehicle is confined to operate within a linear force model range and* $\|\boldsymbol{\zeta}\| \leq \mathcal{F}\, \|\tilde{\boldsymbol{q}}_v\|$, *with* $\mathcal{F}$ *being a constant related to maximum aerodynamic force.*

Assumption 4.2. *The controller in (3.28) enables exponential tracking of* $\boldsymbol{\omega}$. *The time-scale of* $\tilde{\boldsymbol{\omega}}$ *convergence is much shorter than* $\tilde{\boldsymbol{v}}$ *and* $\tilde{\boldsymbol{q}}_v$ *thus can be treated as* $\boldsymbol{\omega} = \boldsymbol{\omega}_r$.

Given exponential convergence proved in Proposition 3.2, Assumption 4.2 can be easily validated through gain tuning based on the vehicle's hardware. With the composite adaptation on the force model and tracking controller defined previously, the convergence of $\tilde{\boldsymbol{v}}$, $\tilde{\boldsymbol{q}}_v$, and $\tilde{\boldsymbol{\theta}}$ is given below.

Theorem 4.1. *By applying the controller, force allocation, and model adaptation from (3.19), (3.28), (3.35), (3.36), (4.10), and (4.11), the tracking errors* $\|\tilde{\boldsymbol{v}}\|$, $\|\tilde{\boldsymbol{q}}_v\|$ *and parameter error* $\left\|\tilde{\boldsymbol{\theta}}\right\|$ *will exponentially converge to a bounded error ball.*

Proof. A Lyapunov function consisting of tracking error and prediction error terms is selected

$$\mathcal{V} = \frac{1}{2}\tilde{\boldsymbol{v}}^\top \tilde{\boldsymbol{v}} + \gamma^{-1}(2 - 2\tilde{q}_0) + \frac{1}{2}\tilde{\boldsymbol{\theta}}^\top \boldsymbol{P}^{-1}\tilde{\boldsymbol{\theta}} \tag{4.12}$$

where the global attitude tracking error $2 - 2\tilde{q}_0$ is used, similar to [96] for $SO(3)$, with its time derivative

$$\dot{\tilde{q}}_0 = -\frac{1}{2}\tilde{\boldsymbol{q}}_v^\top \left(\boldsymbol{\omega} - \tilde{\boldsymbol{R}}^\top \boldsymbol{\omega}_d\right).$$

Taking the time derivative of $\mathcal{V}$ and substituting in (4.8), (4.10), and (4.11) with Assumption 4.2, we get

$$\begin{aligned}\dot{\mathcal{V}} = &-\tilde{\boldsymbol{v}}^\top \boldsymbol{K}_v \tilde{\boldsymbol{v}} + \tilde{\boldsymbol{v}}^\top \boldsymbol{\zeta} - \gamma^{-1}\tilde{\boldsymbol{q}}_v^\top \boldsymbol{\Lambda}_q \tilde{\boldsymbol{q}}_v \\ &- \frac{1}{2}\tilde{\boldsymbol{\theta}}^\top \left(\boldsymbol{W}_f^\top \boldsymbol{W}_f + \lambda \boldsymbol{P}\right)\tilde{\boldsymbol{\theta}} \\ &- \frac{\sigma}{2}\left(\|\tilde{\boldsymbol{\theta}}\|^2 + \|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_0\|^2 - \|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|^2\right).\end{aligned}$$

We define the minimum eigenvalues of positive definite matrices, $c_v = \lambda_{\min}(\boldsymbol{K}_v)$, $c_q = \lambda_{\min}(\boldsymbol{\Lambda}_q)$, and $c_P = \lambda_{\min}(\boldsymbol{P})$. Applying Assumption 4.1 and using the fact that $\boldsymbol{W}_f^\top \boldsymbol{W}_f \geq 0$, we arrive at

$$\begin{aligned}\dot{\mathcal{V}} \leq &-c_v \|\tilde{\boldsymbol{v}}\|^2 - \frac{c_q}{\gamma}\|\tilde{\boldsymbol{q}}_v\|^2 - \frac{\lambda c_P + \sigma}{2}\|\tilde{\boldsymbol{\theta}}\|^2 + \frac{\sigma}{2}\|\boldsymbol{\theta}_0\|^2 \\ &+ \mathcal{F}\|\tilde{\boldsymbol{v}}\|\,\|\tilde{\boldsymbol{q}}_v\| \\ = &-\begin{bmatrix}\|\tilde{\boldsymbol{v}}\| \\ \|\tilde{\boldsymbol{q}}_v\|\end{bmatrix}^\top \begin{bmatrix} c_v & \mathcal{F}/2 \\ \mathcal{F}/2 & c_q/\gamma\end{bmatrix}\begin{bmatrix}\|\tilde{\boldsymbol{v}}\| \\ \|\tilde{\boldsymbol{q}}_v\|\end{bmatrix} - \frac{\lambda c_P + \sigma}{2}\|\tilde{\boldsymbol{\theta}}\|^2 \\ &+ \frac{\sigma}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|^2.\end{aligned}$$

By selecting a $\gamma < (4c_v c_p)/\mathcal{F}^2$, we can find a constant c, such that $\dot{\mathcal{V}} < -c\mathcal{V} + \frac{\sigma}{2}\|\boldsymbol{\theta}_0\|^2$. This proves that $\|\tilde{\boldsymbol{v}}\|$, $\|\tilde{\boldsymbol{q}}_v\|$, and $\|\tilde{\boldsymbol{\theta}}\|$ will exponentially converge to an error ball bounded by initial parameter error $\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|$. ■

4.3 3D Airflow Sensing

To measure the angle of attack (α) and the sideslip angle (β) in flight, a custom 3D airflow sensor (Figure B.3) containing three differential pressure sensors and a conic tip [110] was developed. The sensor uses the SDP33 chip from Sensirion, a pressure sensor based on thermal mass flow [111]. These sensors have almost no zero-pressure offset and drift (0.2 Pa), which makes them ideal for sensing flow angles. In addition, their fast response (< 3 ms) allows them to be used in the attitude control loop. The conic tip has four orifices placed symmetrically relative to a central orifice. The difference in pressure for each pair of holes varies with α and β. The relationship between the differential pressures ($q_\infty, q_\alpha, q_\beta$) and the incident velocity

$\boldsymbol{v}_i$ in the body frame is modeled as:

$$\boldsymbol{v}_i = \sqrt{\frac{2q_\infty/\rho}{q_\infty^2 + (kq_\beta)^2 + (kq_\alpha)^2}} \begin{bmatrix} q_\infty \\ kq_\beta \\ kq_\alpha \end{bmatrix} \tag{4.13}$$

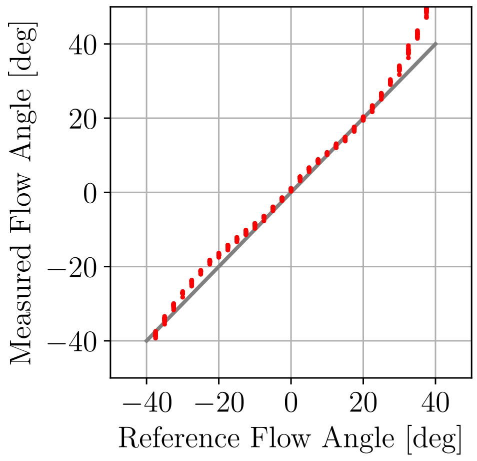


Figure 4.3: Linearity of flow angle measurement and model fit.

A wind tunnel was used to calibrate the sensor and fit the coefficient k from the above equation. The linearity of estimated flow angle is shown in Figure 4.3.

The feedback from the sensor provides an accurate estimate of the incident airspeed vector $\boldsymbol{v}_i$, which enables the aircraft to fly at desired aerodynamic conditions. The airspeed vector is used in the force model described by (4.6) and (4.7) in order to compensate for the aerodynamic forces and to provide an adaptation basis $\boldsymbol{\Phi}$.

4.4 Experiments on The Prototype Vehicle

We used a custom fixed-wing VTOL aircraft (Appendix B.2) running a modified PX4 firmware on a Pixhawk flight controller [107]. We conducted indoor experiments using the Real Weather Wind Tunnel from the Center for Autonomous Systems and Technologies (CAST). The fan array based wind tunnel can generate uniform wind fields up to 12.9 m/s and produces wind speeds linearly with an input throttle. The facility is equipped with motion capture cameras which allow closed-loop position control in front of the fan array.

During the experiment, the VTOL keeps its position fixed in front of the fan array, as shown in Figure 4.4. The control loop as well as the force allocation and parameter adaptation is running at 250 Hz. Given our estimate of $\hat{\boldsymbol{\theta}}$'s posterior distribution

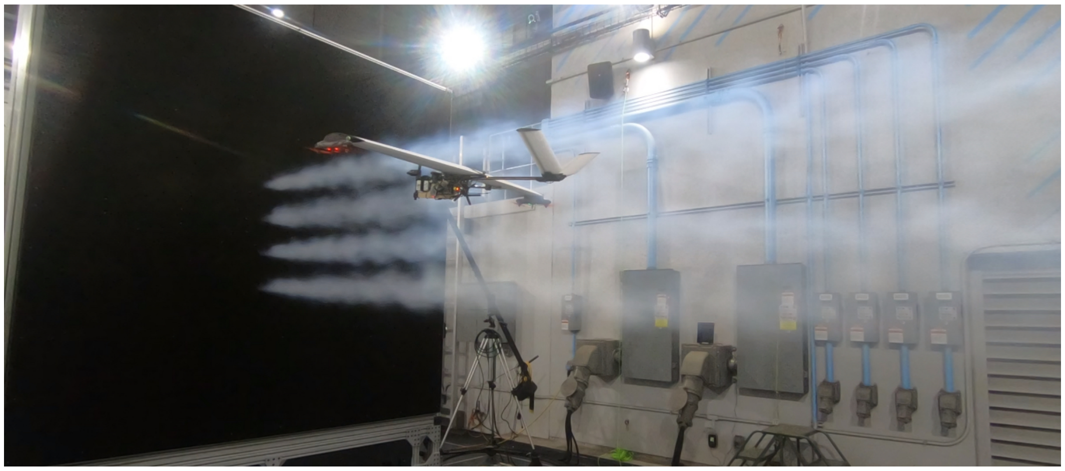


Figure 4.4: The VTOL flying in position control in front of the CAST Fan Array, with smoke for flow visualization.

from aerodynamic testing shown in Table B.5, we limit the maximum deviation of $\hat{\boldsymbol{\theta}}$ to stay within fixed bounds around the initial parameter estimate. The filter used for the acceleration measurement and prediction (4.9) is a first order low-pass filter with a cutoff frequency of 10 Hz. In all experiments, $\boldsymbol{P}$ is initialized to a diagonal matrix ($\boldsymbol{P}_0$) with elements listed in Table 4.1.

Table 4.1: Initial adaptation gains: diagonal of $\boldsymbol{P}_0$

$\bar{C}_{Tx}$	$\bar{C}_{Tz}$	$\bar{C}_S$	C_{D_0}	C_{D_1}	C_{D_2}	C_{L_0}	C_{L_1}
200	200	0.1	1	5	20	0.1	0.1

4.4.1 Aerodynamic Parameters Convergence

Figure 4.5 shows parameter convergence for seven runs of the composite adaptation controller from (4.10) and (4.11), each with randomly sampled initial parameters. Each experiment was started with the vehicle hovering in still air. The fan array was then sequentially set to 30%, 50%, and 70% throttle, which roughly corresponds to 4 m/s, 6.5 m/s, and 9 m/s wind speeds, as the vehicle continued to maintain its position.

As the test conditions stay the same across different runs and only the initial parameters are varied, it can be seen from Figure 4.5 that the components of $\hat{\boldsymbol{\theta}}$ converge to some patterns, indicated by the lower variance as time progresses. It is also interesting to note that C_{L_0}, C_{L_1}, and $\bar{C}_{Tz}$ have cleaner trends than the other variables, indicating steady and accurate force estimates in the vertical direction. There seem to be bigger variations in C_{D_1} estimates. Overall, the parameter convergence is consistent with varying initial conditions, which reflects the robustness of the method.

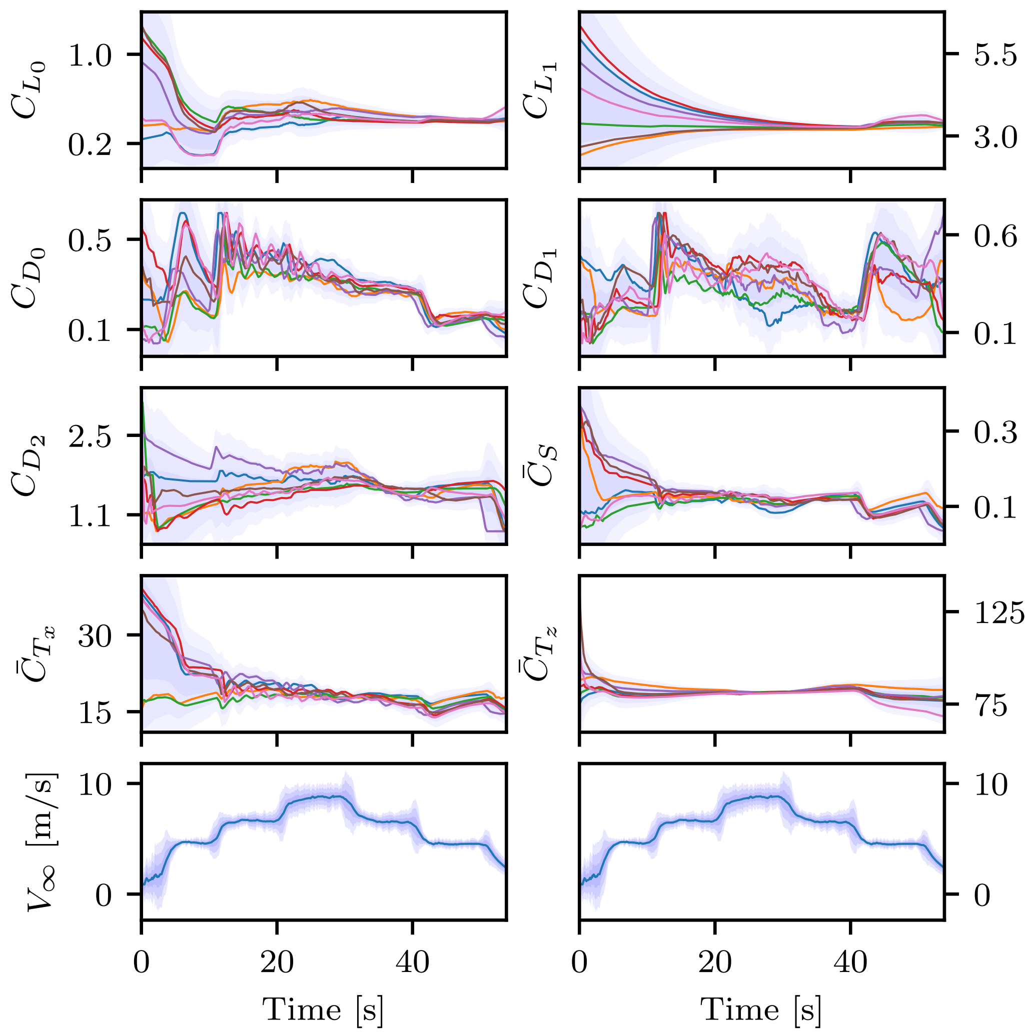


Figure 4.5: Convergence of aerodynamic coefficient estimates for seven runs with random initial $\hat{\boldsymbol{\theta}}$. The filled region represents 1σ to 3σ deviations. Freestream velocity is duplicated for visualization.

4.4.2 Comparison of Different Feedback Control Schemes

The tests started with the vehicle hovering and the fan array turned off, the throttle is increased to 30% (4 m/s wind) for 10 seconds and then to 70% (9 m/s wind) in 10% increments every 5 seconds. The fan array was switched off after 10 seconds at top throttle. The same experiment profile was used to test five different control and adaptation schemes: **(I)** The full implementation based on (3.19), (4.10), and (4.11), (C. Adapt); **(II)** Same method, but with $\boldsymbol{P}$ update disabled (C. Adapt $\dot{\boldsymbol{P}} = 0$); **(III)**. Tracking error only adaptation, with $\dot{\hat{\boldsymbol{\theta}}} = \boldsymbol{P}_0 \left(\boldsymbol{\Phi}^\top \boldsymbol{R}^\top \tilde{v}\right)$; **(IV)**

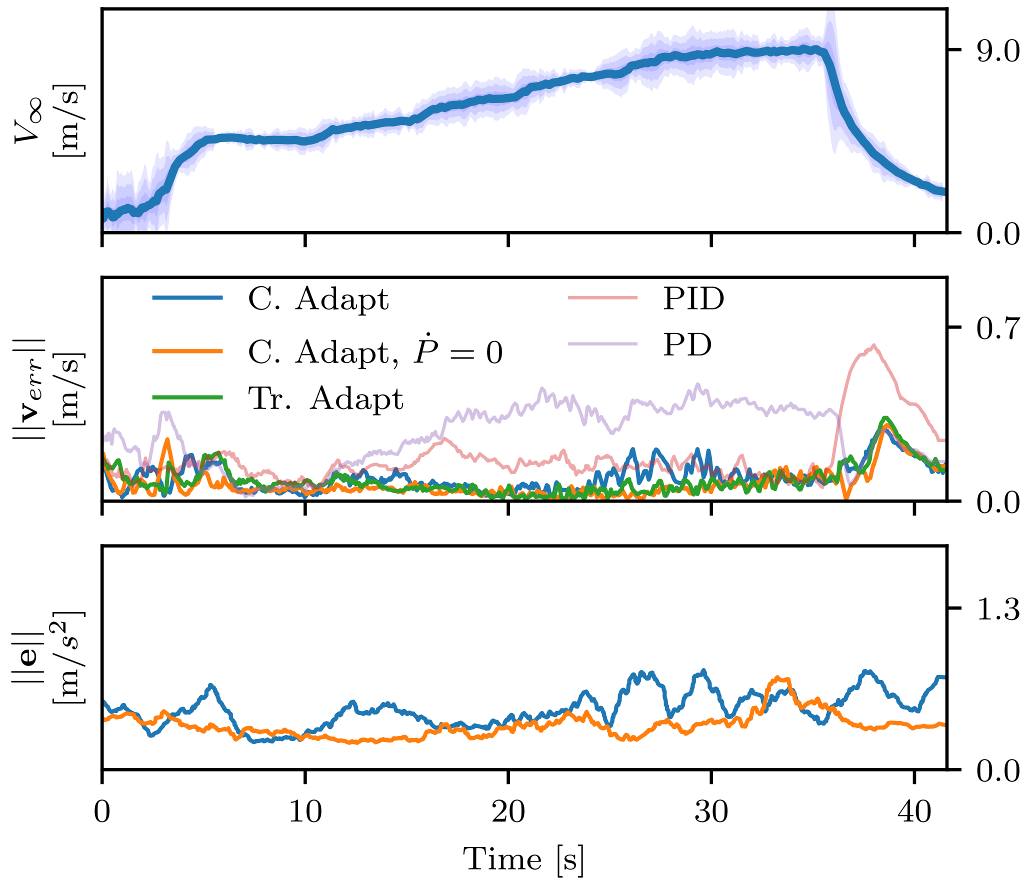


Figure 4.6: From top to bottom, figures show measured total airspeed from 3D airflow sensor, norm of velocity tracking errors for all five controllers, and norm of filtered (for visualization) prediction errors for the two composite adaptation controllers.

Proportional-Integral-Derivative (PID) controller; and **(V)** Proportional-Derivative (PD) controller.

The adaptive controllers **(I)** to **(III)** use the controller from (3.19). The PID and PD controller are based on Pixhawk's default implementation of multirotor control

$$\bar{\boldsymbol{f}} = -\boldsymbol{g} - \boldsymbol{K}_P\tilde{\boldsymbol{v}} - \boldsymbol{K}_D\dot{\tilde{\boldsymbol{v}}} - \boldsymbol{K}_I \int_0^t \tilde{\boldsymbol{v}}(\tau)\, d\tau$$

with positive definite gain matrices $\boldsymbol{K}_P$, $\boldsymbol{K}_I$, $\boldsymbol{K}_D$. The force allocation technique described in Chapter 3 is still active with constant $\hat{\boldsymbol{\theta}}$ using values from Table B.5. The equivalent gains used in all five controllers are held the same for consistent comparison.

It is clear to see from Figure 4.6 that as airspeed increases, the PD controller starts to accumulate unrecoverably large tracking errors. The PID controller can account for some model uncertainties and is almost on par with the adaptive controllers, however, we can see its slow response to changing disturbance as the vehicle overshoots significantly when airspeed ramps down.

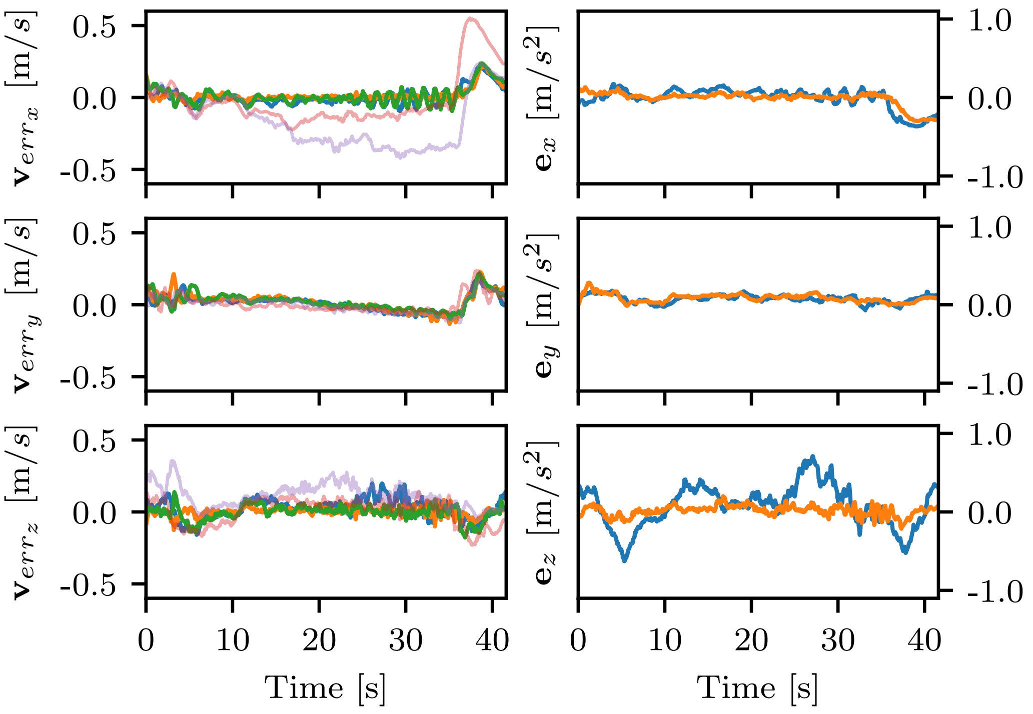


Figure 4.7: Velocity tracking error and filtered (for visualization) acceleration prediction error of the same experiment as in Figure 4.6, for each axis individually. Velocity error is in Earth fixed North-East-Down frame with the vehicle flying north; acceleration is in body frame.

All versions of the adaptive controllers have better overall velocity tracking and do not suffer from the large overshoot that the PID has during sudden changes of the environment. For the two composite controllers, the prediction is kept as low as $0.05\boldsymbol{g}$ in error consistently. Interestingly, the constant gain composite controller actually outperforms the full implementation in terms of prediction accuracy. This is more prominent in Figure 4.7, which shows that the $\boldsymbol{e}$ for C. Adapt lies mostly in the body z-axis. The result video is also available online.[1]

4.5 Chapter Summary

In this chapter, we introduced a set of linear-in-parameter force models with good steady-state accuracy appropriate for fixed-wing VTOL. The 3D airflow sensor provided crucial realtime information related to the aerodynamic forces on the vehicle. The adaptation law, together with the controllers, was proven to possess

[1]Experiment video: `https://youtu.be/2d2Gy-PjgpA`

the stability and robustness needed for high speed transition flight. After being integrated into our custom hybrid VTOL, the composite adaptive controller with 3D airflow sensor feedback greatly improved the tracking and prediction performance over baseline methods.

Chapter 5

FAULT-TOLERANT DESIGN

One of the benefits for distributed electric rotors as propulsion systems is their robustness against failures of some of the rotors. In this chapter, we will first lay the mathematical foundation needed, then discuss how rotor configuration can be optimized for a specific type of fixed-wing VTOL aircraft. The content in this chapter is based on the published prior results from [112].

5.1 Controllability with Rotor Failure

The robustness of distributed multirotor aircraft ties closely to the controllability of such dynamics systems. We start from the same dynamics (3.1) and (3.2) introduced in Chapter 3, with some variations on state representation $\boldsymbol{x} = [\boldsymbol{p}; \boldsymbol{v}; \boldsymbol{q}; \boldsymbol{\omega}]$:

$$\dot{\boldsymbol{p}} = \boldsymbol{v} \qquad \dot{\boldsymbol{v}} = \boldsymbol{g} + \boldsymbol{R}\boldsymbol{f}_T \tag{5.1a}$$

$$\dot{\boldsymbol{q}} = \boldsymbol{Q}(\boldsymbol{q})\boldsymbol{\omega} \qquad \boldsymbol{J}\dot{\boldsymbol{\omega}} = \boldsymbol{S}(\boldsymbol{J}\boldsymbol{\omega})\boldsymbol{\omega} + \boldsymbol{\tau}_T. \tag{5.1b}$$

We elect to use ZYX Euler angles $\boldsymbol{q} = [\phi; \theta; \psi]$ as attitude representation describing the orientation of the aircraft with respect to the inertial frame. $\boldsymbol{R}(\boldsymbol{q})$ that transforms a vector from the body frame to the inertial frame and $\boldsymbol{Q}(\boldsymbol{q})$ that transforms $\boldsymbol{\omega}$ into $\dot{\boldsymbol{q}}$ are defined as a function of $\boldsymbol{q}$:

$$\boldsymbol{R}(\boldsymbol{q}) = \begin{bmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix}, \tag{5.2}$$

$$\boldsymbol{Q}(\boldsymbol{q}) = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{bmatrix}. \tag{5.3}$$

Furthermore, we choose to neglect the contribution of $\boldsymbol{f}_A$ and $\boldsymbol{\tau}_A$ in multirotor dynamics as their contributions are typically small in the rotor failure case we are interested in. Namely, all body forces and moments come from rotor thrusts and torques:

$$\boldsymbol{f}_b = \boldsymbol{f}_T \quad \text{and} \quad \boldsymbol{\tau}_b = \boldsymbol{\tau}_T.$$

We remind ourselves that both $\boldsymbol{f}_T = [f_{T,x}; f_{T,y}; f_{T,z}]$ and $\boldsymbol{\tau}_T = [\tau_{T,x}; \tau_{T,y}; \tau_{T,z}]$ are in the body frame. For simplicity, we define control signal $\boldsymbol{u} = [\boldsymbol{f}_T; \boldsymbol{\tau}_T]$.

5.1.1 Static Hover

When a multirotor aircraft detects rotor failure, it is desirable to steer the aircraft to static hover as soon as possible to cope with the emergency and land safely if necessary. If the aircraft has an ability to generate the force $\boldsymbol{f}_T$ and the moment $\boldsymbol{\tau}_T$ in all directions unconstrained, then one can design a controller such that $\boldsymbol{f}_T$ and $\boldsymbol{\tau}_T$ stabilize the position and attitude dynamics, respectively. However, many practical multirotor aircraft can only generate a limited $\boldsymbol{f}_T$, and the position control often relies on the attitude control because body fixed $\boldsymbol{f}_T$ can only be directed through change in attitude $\boldsymbol{q}$, as shown in the hierarchical control architecture from Figure 3.3.

A multirotor may achieve static hover in two steps: first, it finds a desired attitude $\boldsymbol{q}_d$ such that the constrained $\boldsymbol{f}_T$ can stabilize the position dynamics towards $\dot{\boldsymbol{p}} = \dot{\boldsymbol{v}} = \boldsymbol{0}$, second, it stabilizes the attitude about $\dot{\boldsymbol{q}} = \dot{\boldsymbol{\omega}} = \boldsymbol{0}$ and $\boldsymbol{q} = \boldsymbol{q}_d$. We make a mild assumption that the aircraft can always generate thrust large enough to counterbalance its own weight after the failure of any combination of rotors of interest. Obviously, the remaining portion of $\boldsymbol{f}_T$ after canceling the gravity, if any, can be used for position control. Hereafter, we mainly focus on the attitude controllability of and moments exerted on multirotor aircraft and let $\boldsymbol{q}_d = \boldsymbol{0}$.

5.1.2 Null Controllability

Controllability analysis has been a classical problem in the controls community and many seminal works exist in the literature [95], [113], [114]. Especially for nonlinear systems, different concepts of controllability have been developed; among them, we are interested in *null controllability* of a multirotor, which is related to its ability to reach static hover.

Definition 5.1. *Consider a nonlinear system $\dot{\boldsymbol{x}} = \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u})$, where $\boldsymbol{x}(t) \in \mathbb{R}^n$ is a state vector, $\boldsymbol{u}(t) \in \Omega \subset \mathbb{R}^m$ is a control input bounded by a restraint set Ω, and $\boldsymbol{h}$ is C^1. The system is locally Ω-null controllable if there exists an open set $\mathcal{X} \subset \mathbb{R}^n$ containing the origin such that any $\boldsymbol{x}_0 \in \mathcal{X}$ at time t_0 can be controlled to $\boldsymbol{x}_f = \boldsymbol{0}$ in finite time $t_f < \infty$ by some controller satisfying $\boldsymbol{u}(t) \in \Omega$ for all time $t \in [t_0, t_f]$. The system is globally Ω-null controllable if $\mathcal{X} = \mathbb{R}^n$.*

The following theorem and corollary from [115] provide conditions for a system to be globally Ω-null controllable. We refer the readers to [115] for their proofs.

Theorem 5.1. *Consider a system $\dot{\boldsymbol{x}} = \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u})$. Suppose there exist a scalar function $V(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$ and a vector function $U(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}^m$ in C^1 such that*

(a) $V(\boldsymbol{x}) = 0$ *if and only if* $\boldsymbol{x} = \boldsymbol{0}$, *and* $V(\boldsymbol{x}) > 0$ *otherwise;*

(b) $\lim_{\|\boldsymbol{x}\|\to\infty} V(\boldsymbol{x}) = +\infty$*;*

(c) $U(\boldsymbol{x}) \in \Omega \quad \forall \boldsymbol{x} \in \mathbb{R}^n$*;*

(d) $\dot{V} < 0 \, \forall \boldsymbol{x} \neq \boldsymbol{0}$, *and* $\dot{V} = 0$ *when* $\boldsymbol{x} = \boldsymbol{0}$.

Then the system is globally asymptotically stable about the origin with the controller $\boldsymbol{u}(t) = U(\boldsymbol{x}(t)) \subset \Omega$ *for* $0 \leq t < \infty$.

In addition to (d), if we also have that $\dot{V} \leq -\alpha V$ holds for some $\alpha > 0$, then the system is globally exponentially stable.

Corollary 5.1.1. *Consider the system in Theorem 5.1 and assume that* $V(\boldsymbol{x})$ *and* $U(\boldsymbol{x})$ *exist satisfying the conditions therein. If the followings also hold*

(e) $\boldsymbol{h}(\boldsymbol{0}, \boldsymbol{0}) = \boldsymbol{0}$*;*

(f) $\boldsymbol{u} = \boldsymbol{0}$ *is in the interior of* Ω*;*

(g) *rank* $\left[\boldsymbol{B}, \boldsymbol{A}\boldsymbol{B}, \cdots, \boldsymbol{A}^{n-1}\boldsymbol{B}\right] = n$, *where* $\boldsymbol{A} = \boldsymbol{h}_{\boldsymbol{x}}(\boldsymbol{0}, \boldsymbol{0})$ *and* $\boldsymbol{B} = \boldsymbol{h}_{\boldsymbol{u}}(\boldsymbol{0}, \boldsymbol{0})$,

then the domain of null controllability for the system is $\mathcal{X} = \mathbb{R}^n$.

Define attitude state vector $\boldsymbol{y} = [\boldsymbol{q}; \boldsymbol{\omega}]$. For static hover, we want to arrive at $\boldsymbol{y}(t_f) = \dot{\boldsymbol{y}}(t_f) = \boldsymbol{0}$ in finite time $t_f < \infty$ from a given initial condition at time t_0 by some controller $\boldsymbol{\tau}_T$ bounded by a control input set $\mathcal{M}$ for all $t \in [t_0, t_f]$. In other words, we want the aircraft to be $\mathcal{M}$-null controllable. Let us denote the attitude dynamics in (5.1) as $\dot{\boldsymbol{y}} = \boldsymbol{g}(\boldsymbol{y}, \boldsymbol{\tau}_T)$, and let $\mathcal{Y}$ be the domain of $\boldsymbol{y}$ where the Euler angles are valid and $\boldsymbol{\omega} \in \mathbb{R}^3$.

Corollary 5.1.2. *The system* $\dot{\boldsymbol{y}} = \boldsymbol{g}(\boldsymbol{y}, \boldsymbol{\tau}_T)$ *is locally* $\mathcal{M}$*-null controllable if* $\boldsymbol{\tau}_T = \boldsymbol{0}$ *is in the interior of* $\mathcal{M}$.

Proof. We check the conditions presented in Theorem 5.1 and Corollary 5.1.1. The local result is due to (c) because $\boldsymbol{\tau}_T(\boldsymbol{y}) \in \mathcal{M}$ is not guaranteed for all $\boldsymbol{y} \in \mathcal{Y}$ with a state feedback controller $\boldsymbol{\tau}_T$, especially when $\mathcal{M}$ is restricted due to rotor failure. However, it is possible to show that there exists a non-empty domain of null controllability. If $\mathcal{M} = \mathbb{R}^3$, then the system is $\mathcal{M}$-null controllable on $\mathcal{Y}$.

Similar to the attitude controller from (3.28) for $SO(3)$, an Euler angle based controller can be defined with

$$\boldsymbol{\omega}_r = \boldsymbol{\omega}_d + \boldsymbol{Q}^{-1}\boldsymbol{K}_1(\boldsymbol{q}_d - \boldsymbol{q}) \quad \text{and} \quad \boldsymbol{\tau}_T = \boldsymbol{J}\dot{\boldsymbol{\omega}}_r - \boldsymbol{J}\boldsymbol{\omega} \times \boldsymbol{\omega} - \boldsymbol{J}\boldsymbol{K}_2(\boldsymbol{\omega} - \boldsymbol{\omega}_r).$$

Here, $\boldsymbol{K}_1$ and $\boldsymbol{K}_2$ are positive definite gain matrices. For static hover $\boldsymbol{q}_d = \boldsymbol{\omega}_d = \boldsymbol{0}$, then we have

$$\boldsymbol{\omega}_r = -\boldsymbol{Q}^{-1}\boldsymbol{K}_1\boldsymbol{q} \quad \text{and} \quad \dot{\boldsymbol{\omega}}_r = -\dot{\boldsymbol{Q}}^{-1}\boldsymbol{K}_1\boldsymbol{q} - \boldsymbol{Q}^{-1}\boldsymbol{K}_1\dot{\boldsymbol{q}}.$$

Define $\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega} - \boldsymbol{\omega}_r$ and $\tilde{\boldsymbol{y}} = [\boldsymbol{q}; \tilde{\boldsymbol{\omega}}]$. Notice that $\tilde{\boldsymbol{y}} = \boldsymbol{0}$ if and only if $\boldsymbol{y} = \boldsymbol{0}$, so we can select a Lyapunov function

$$V = \frac{1}{2}\boldsymbol{q}^\top\boldsymbol{q} + \frac{1}{2}\tilde{\boldsymbol{\omega}}^\top\tilde{\boldsymbol{\omega}} = \frac{1}{2}\left\|\tilde{\boldsymbol{y}}\right\|^2,$$

with $V = 0$ if and only if $\boldsymbol{y} = \boldsymbol{0}$ and $V > 0$ when $\boldsymbol{y} \neq \boldsymbol{0}$, thereby satisfying (a). Furthermore, V is radially unbounded and (b) is satisfied.

With the above controller $\boldsymbol{\tau}_T$, it can be shown that

$$\dot{V} = -\tilde{\boldsymbol{y}}^\top\boldsymbol{P}\tilde{\boldsymbol{y}}, \quad \text{where} \quad \boldsymbol{P} = \begin{bmatrix} \boldsymbol{K}_1 & -\frac{1}{2}\boldsymbol{Q} \\ -\frac{1}{2}\boldsymbol{Q}^\top & \boldsymbol{K}_2 \end{bmatrix}.$$

Also P is a positive definite matrix when we properly select gain matrices $\boldsymbol{K}_1$ and $\boldsymbol{K}_2$. Hence, $\dot{V} = 0$ if and only if $\boldsymbol{y} = \boldsymbol{0}$; and $\dot{V} < 0$ otherwise, satisfying (d). In fact,

$$\dot{V} \leq -\lambda_{\min}(\boldsymbol{P})\left\|\tilde{\boldsymbol{y}}\right\|^2 = -2c_1V$$

with $c_1 = \lambda_{\min}(\boldsymbol{P}) = \min\{\lambda_{\min}(\boldsymbol{K}_1), \lambda_{\min}(\boldsymbol{K}_2)\} > 0$, and $V(\tilde{\boldsymbol{y}}) \leq e^{-2c_1t}V(\tilde{\boldsymbol{y}}_0)$. Therefore, the closed loop system is exponentially stable and

$$\left\|\boldsymbol{q}\right\| \leq \left\|\tilde{\boldsymbol{y}}\right\| \leq e^{-c_1t}\left\|\tilde{\boldsymbol{y}}_0\right\|.$$

Regarding (c), we show that there exists a non-empty subset $\mathcal{U}$ within the domain of null controllability, which guarantees $\boldsymbol{\tau}_T$ to be within a subset of $\mathcal{M}$ for all times. Define $\mathcal{B}_\epsilon = \{\boldsymbol{\tau}_T \in \mathbb{R}^3 \mid \left\|\boldsymbol{\tau}_T\right\| \leq \epsilon\}$ and let $\bar{\epsilon} > 0$ be the maximum radius possible within $\mathcal{M}$. Such an $\bar{\epsilon}$ always exists because the origin is in the interior of $\mathcal{M}$. We show that ϵ is determined by an initial state, and $\mathcal{U}$ is a set of $\boldsymbol{y}_0$ that satisfies $\epsilon(\boldsymbol{y}_0) \leq \bar{\epsilon}$. For this, notice that

$$\begin{aligned}\left\|\boldsymbol{\tau}_T\right\| &\leq \left\|\boldsymbol{J}\dot{\boldsymbol{\omega}}_r\right\| + \left\|\boldsymbol{J}\boldsymbol{\omega}\right\|\left\|\boldsymbol{\omega}\right\| + \left\|\boldsymbol{J}\boldsymbol{K}_2\tilde{\boldsymbol{\omega}}\right\| \\ &\leq \lambda_{\max}(\boldsymbol{J})\left\|\dot{\boldsymbol{\omega}}_r\right\| + \lambda_{\max}(\boldsymbol{J})\left\|\boldsymbol{\omega}\right\|^2 + \lambda_{\max}(\boldsymbol{J}\boldsymbol{K}_2)\left\|\tilde{\boldsymbol{\omega}}\right\|.\end{aligned}$$

With the controller $\boldsymbol{\tau}_T$, the closed loop attitude dynamics become $\boldsymbol{J}\dot{\tilde{\boldsymbol{\omega}}} + \boldsymbol{J}\boldsymbol{K}_2\tilde{\boldsymbol{\omega}} = \boldsymbol{0}$ whose solution is $\tilde{\boldsymbol{\omega}} = e^{-\boldsymbol{K}_2 t}\tilde{\boldsymbol{\omega}}_0$, and thus, $\|\tilde{\boldsymbol{\omega}}\| \leq e^{-c_2 t}\|\tilde{\boldsymbol{\omega}}_0\|$ with $c_2 = \lambda_{\min}(\boldsymbol{K}_2) \geq c_1 > 0$. Furthermore, we have

$$\begin{aligned} \|\boldsymbol{\omega}\| &\leq \|\tilde{\boldsymbol{\omega}}\| + \|\boldsymbol{\omega}_r\| \leq \|\tilde{\boldsymbol{\omega}}\| + c_3\|\boldsymbol{q}\| \\ &\leq e^{-c_2 t}\|\tilde{\boldsymbol{\omega}}_0\| + c_3 e^{-c_1 t}\|\tilde{\boldsymbol{y}}_0\|, \end{aligned}$$

with $c_3 = \sqrt{2}\lambda_{\max}(\boldsymbol{K}_1) > 0$. Also we can simplify

$$\begin{aligned} \|\dot{\boldsymbol{\omega}}_r\| &\leq \left\|\dot{\boldsymbol{Q}}^{-1}\boldsymbol{K}_1\boldsymbol{q}\right\| + \left\|\boldsymbol{Q}^{-1}\boldsymbol{K}_1\dot{\boldsymbol{q}}\right\| \\ &\leq \lambda_{\max}(\boldsymbol{K}_1)\left(\|\boldsymbol{q}\| + \sqrt{2}\right)\|\dot{\boldsymbol{q}}\| \leq \lambda_{\max}(\boldsymbol{K}_1)\lambda_{\max}(\boldsymbol{Q})\left(\|\boldsymbol{q}\| + \sqrt{2}\right)\|\boldsymbol{\omega}\| \\ &\leq \lambda_{\max}(\boldsymbol{K}_1)\lambda_{\max}(\boldsymbol{Q})\left(e^{-c_1 t}\|\tilde{\boldsymbol{y}}_0\| + \sqrt{2})(e^{-c_2 t}\|\tilde{\boldsymbol{\omega}}_0\| + c_3 e^{-c_1 t}\|\tilde{\boldsymbol{y}}_0\|\right). \end{aligned}$$

Consequently, $\|\boldsymbol{\tau}_T\|$ is upper bounded by the sum of terms that are in turn upper bounded by the terms that are exponentially decaying from the norms of initial states; we let this final upper bound of $\|\boldsymbol{\tau}_T\|$ be ϵ. Therefore, for the initial states that result in $\epsilon \leq \bar{\epsilon}$, $\boldsymbol{\tau}_T \in \mathcal{B}_\epsilon \subset \mathcal{B}_{\bar{\epsilon}} \subset \mathcal{M}$ for all times and (c) is satisfied. Since $\epsilon > 0$ always exist, such initial states also exist, and $\mathcal{U}$ is not empty. Furthermore, $\mathcal{U}$ is larger with a greater $\bar{\epsilon}$.

The remaining conditions are straightforward. (e) is checked by noting that $\boldsymbol{g}(\boldsymbol{0}, \boldsymbol{0}) = \boldsymbol{0}$, and (f) is satisfied by the assumption. The controllability matrix constructed with

$$\boldsymbol{A} = \boldsymbol{g}_{\boldsymbol{y}}(\boldsymbol{0}, \boldsymbol{0}) = \begin{bmatrix} 0 & \boldsymbol{I} \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad \boldsymbol{B} = \boldsymbol{g}_{\boldsymbol{\tau}_T}(\boldsymbol{0}, \boldsymbol{0}) = \begin{bmatrix} 0 \\ \boldsymbol{J}^{-1} \end{bmatrix}$$

has a full rank, thereby satisfying (g). ■

According to Corollary 5.1.2, one can quickly inspect if the aircraft is null controllable when its rotors are failed by checking if the resultant moment set $\mathcal{M}$ contains a neighborhood of the origin. In the proof, we also noted that it is desirable to have a greater $\bar{\epsilon}$ that will result in larger $\mathcal{U}$. Physically, $\bar{\epsilon}$ measures the magnitude of the admissible moment in the weakest direction. If the evolution of attitude dynamics is too slow due to small $\bar{\epsilon}$, the position dynamics may become undesirable or too much energy may be drained from the power source before reaching static hover. Therefore, we want the aircraft to maintain as great $\bar{\epsilon}$ as possible after rotor failure. This may be done from the initial design stage of the aircraft using, for instance, the optimization presented in Section 5.2. The aircraft may also be designed in such a way that the position and orientation of its rotors are actively changed to reshape the moment set $\mathcal{M}$ and to increase $\bar{\epsilon}$. In Section 5.3, we illustrate how rotor failure affects controllability and demonstrate how aircraft design can help preserve controllability with a concrete example.

5.2 Control-Centric Design Optimization

Let us denote the force and moment developed by the i-th rotor as $\boldsymbol{f}_{T,i}$ and $\boldsymbol{\tau}_{T,i}$, respectively. We inherit the rotor force and torque models from Section 3.1.1 with

$$\boldsymbol{f}_{T,i} = T_i \hat{\boldsymbol{z}}_i \quad \text{and} \quad \boldsymbol{\tau}_{T,i} = T_i \boldsymbol{\mu}_i$$

where $\hat{\boldsymbol{z}}_i$ and $\boldsymbol{\mu}_i$ are defined by the configuration and physical properties of the rotors. Here we define rotor-i's maximum thrust $\bar{T}_i > 0$, and consequently the max force and torque vectors:

$$\bar{\boldsymbol{f}}_{T,i} = \bar{T}_i \hat{\boldsymbol{z}}_i \quad \text{and} \bar{\boldsymbol{\tau}}_{T,i} = \bar{T}_i \boldsymbol{\mu}_i .$$

Without loss of generality, we will assume that all rotors have the same physical properties to simplify analysis, thus dropping subscript i when needed.

5.2.1 Moment Set and Quality Measure

Let us denote the set of all possible moments that the rotors can exert on the aircraft as $\mathcal{M} \subset \mathbb{R}^3$ and define it as

$$\mathcal{M} = \left\{ \boldsymbol{\tau}_T \in \mathbb{R}^3 \ \middle| \ \boldsymbol{\tau}_T = \sum_{i=1}^{n_T} a_i \bar{\boldsymbol{\tau}}_{T,i}, \ \ 0 \le a_i \le 1, \ \ \forall i = 1, \cdots, n_T \right\}$$

where n_T is the total number of rotors. Note that $\mathcal{M}$ is a convex set defined by the vectors $\hat{\boldsymbol{z}}_i$ and $\boldsymbol{\mu}_i$, which in turn are characterized by a set of aircraft design parameters $\boldsymbol{r}_i$, C_T, C_Q, etc. If we denote the set of possible design parameters of interest as $\mathcal{D}$, then for each $\boldsymbol{d} \in \mathcal{D}$, one can construct the associated set $\mathcal{M}(\boldsymbol{d})$.

To achieve our goal of design optimization, we define a quality measure $\bar{\kappa} : \mathcal{D} \to \mathbb{R}_{\ge 0}$ based on some characterization of $\mathcal{M}(\boldsymbol{d})$, which enables comparison of the quality of different aircraft designs in $\mathcal{D}$. Recall from the discussion in Section 5.1.2 that, for null controllability, we prefer to have a large radius $\bar{\epsilon}$ of a maximal ball inscribed in $\mathcal{M}(\boldsymbol{d})$. As a further generalization of the concept, one can also consider fitting geometric objects other than a ball inside of $\mathcal{M}(\boldsymbol{d})$. For example, an ellipsoid would replace the ball if weighted control authority is desired for each dimension of $\mathcal{M}(\boldsymbol{d})$. If independence between different moment directions is desired, a rectangular cuboid may be considered. Regardless of which geometric object is chosen, we define the *maximal scale* of the object within $\mathcal{M}(\boldsymbol{d})$ to be our quality measure function $\bar{\kappa}$. In all cases, the objects describe preferred moment directions and amounts in some manner, and $\bar{\kappa}$ indicates the aircraft's largest ability to generate moments according to this preference. Note that the origin must be always in $\mathcal{M}(\boldsymbol{d})$ in order for $\bar{\kappa}$ to be properly defined; otherwise, we let $\bar{\kappa} = 0$. Also, this measure corresponds to the worst case analysis in the sense that there normally exists extra control authority available beyond the geometric objects considered.

5.2.2 Design Optimization with Rotor Failure

We present a design procedure to obtain an optimal $\boldsymbol{d}^* \in \mathcal{D}$ that results in $\mathcal{M}(\boldsymbol{d}^*)$ maximizing $\bar{\kappa}$. Here, we consider a rectangular cuboid with a fixed ratio representing the desired relative control authority between dimensions of $\mathcal{M}(\boldsymbol{d})$, in order to ensure that certain level of control authority is independently available for each dimension of $\mathcal{M}(\boldsymbol{d})$ even after rotor failure. However, the same approach may be taken with a sphere or an ellipsoid by using the methods in [105] instead of the optimization problem formulated below.

Suppose the rectangular cuboid is defined as

$$C = \left\{ \boldsymbol{\tau}_T \in \mathbb{R}^3 \;\middle|\; \underline{\boldsymbol{\zeta}} \leq \boldsymbol{\tau}_T \leq \bar{\boldsymbol{\zeta}};\;\; \underline{\boldsymbol{\zeta}}, \bar{\boldsymbol{\zeta}} \in \mathbb{R}^3 \right\}$$

with a pre-specified ratio given as $\boldsymbol{\beta} = [\beta_1, \beta_2, \beta_3]^\top$ representing the desired relative control authority for each dimension of $\mathcal{M}(\boldsymbol{d})$. Let $\bar{\kappa}C$ be the scaled cuboid of C with lower and upper bounds of $\boldsymbol{\tau}_T$ replaced by $\bar{\kappa}\underline{\boldsymbol{\zeta}}$ and $\bar{\kappa}\bar{\boldsymbol{\zeta}}$, respectively. Note that the convex set $\mathcal{M}(\boldsymbol{d})$ is a polyhedron, which can be written as $\mathcal{M}(\boldsymbol{d}) = \{\boldsymbol{\tau}_T \in \mathbb{R}^3 \,|\, \boldsymbol{A}\boldsymbol{\tau}_T \leq \boldsymbol{b},\, \boldsymbol{A} = [a_{ij}] \in \mathbb{R}^{p\times 3},\, \boldsymbol{b} = [b_i] \in \mathbb{R}^p\}$, where p is the number of polyhedron facets. Then the following convex optimization problem solves for $\bar{\kappa}^*$, which is the maximal scale possible while ensuring $\bar{\kappa}^*C$ is within $\mathcal{M}(\boldsymbol{d})$ and while considering the relative importance $\boldsymbol{\beta}$:

$$\begin{aligned}
\bar{\kappa}^* = \underset{\bar{\kappa}}{\operatorname{argmax}} & \sum_{i=1}^{3} \log\left(\bar{\kappa}(\bar{\zeta}_i - \underline{\zeta}_i)\right), \\
\text{subject to} \quad & \bar{\kappa} \geq 0, \\
& {\beta_1}^{-1}(\bar{\zeta}_1 - \underline{\zeta}_1) = {\beta_2}^{-1}(\bar{\zeta}_2 - \underline{\zeta}_2) = {\beta_3}^{-1}(\bar{\zeta}_3 - \underline{\zeta}_3), \\
& \underline{\zeta}_k < \bar{\zeta}_k, \quad k = 1, 2, 3, \\
& a_{ij}^+ = \max\{a_{ij}, 0\}, \qquad a_{ij}^- = \max\{-a_{ij}, 0\}, \\
& \sum_{j=1}^{3} a_{ij}^+ \bar{\zeta}_j - a_{ij}^- \underline{\zeta}_j \leq b_i, \quad i = 1, \cdots, p.
\end{aligned} \tag{5.4}$$

In (5.4), the quality measure function is the log of a volume of $\bar{\kappa}C$. The equality constraint enforces the relative ratio of $\bar{\kappa}C$, and the last inequality constraint ensures that $\bar{\kappa}C$ remains within $\mathcal{M}(\boldsymbol{d})$. The optimal set of design parameters $\boldsymbol{d}^*$ is then computed by solving (5.4) over the domain $\mathcal{D}$, that is, $\boldsymbol{d}^* = \operatorname{argmax}_{\boldsymbol{d} \in \mathcal{D}} \bar{\kappa}^*$.

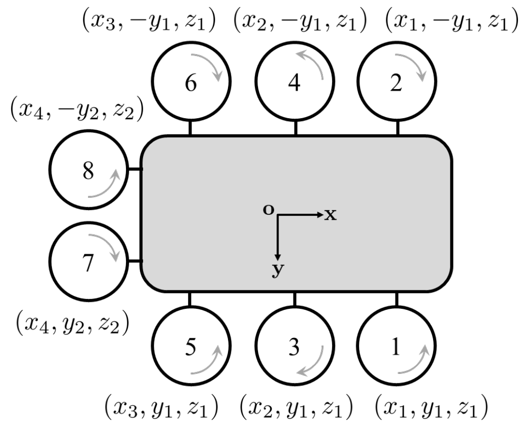
The above formulation, which assumed all rotors are fully functional, can be extended to handle the case with rotor failures. Let $\mathcal{R}$ be a set whose elements are combinations of rotor failures. For each $r \in \mathcal{R}$ and $\boldsymbol{d} \in \mathcal{D}$, there is an associated $\mathcal{M}_r(\boldsymbol{d})$, where the subscript represents the loss of rotors in r, and also $\bar{\kappa}_r^*$ obtained by solving (5.4) with $\mathcal{M}_r(\boldsymbol{d})$. Because we want to maintain at least some control authority even in the worst case scenario, we let $\bar{\kappa}_{\mathcal{R}}^* = \min_{r \in \mathcal{R}} \bar{\kappa}_r^*$ and $\boldsymbol{d}^* = \operatorname{argmax}_{\boldsymbol{d} \in \mathcal{D}} \bar{\kappa}_{\mathcal{R}}^*$.

5.3 Example Design Optimization for AFA

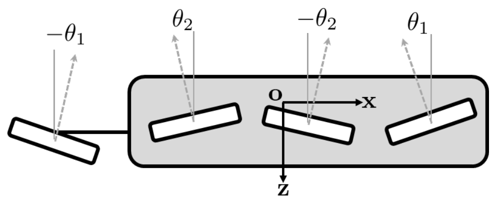
We apply the optimization procedure from Section 5.2.2 to a prototype VTOL vehicle described in Appendix B.1. The schematic showing the placement of rotors are included in Figure 5.1. Specifically, the prototype has a total of eight rotors around its main body, which are placed symmetrically about the body longitudinal axis and are driven by electric brushless DC motors. Among them, six rotors are placed on the sides of the main body, all at the same height and equidistant from the main body to minimize aerodynamic drag when in cruise mode. The last two are located in the back and are also tiltable for forward flight.

5.3.1 Optimization Result

Given fixed rotor locations, we intend to find their optimal orientations to maximize controllability and robustness against failures. The optimization procedure in



(a) Top view



(b) Side view

Figure 5.1: Diagrams of prototype VTOL denoted with design parameters and rotor numbers. The origin of the body frame coincides with the COM. Rotor positions and orientations are defined with respect to the body frame.

Section 5.2.2 is solved for this specific aircraft. The design parameters for the optimization and its domain are chosen as

$$\mathcal{D} = \{\boldsymbol{d} \,|\, 0^\circ \le \theta_i \le 20^\circ,\, i = 1, 2\} \quad \text{with} \quad \boldsymbol{d} = [\theta_1; \theta_2].$$

Figure 5.1b shows that rotors alternate tilt angles θ_1 and θ_2 with respect to body y-axis. The positive tilt angles are chosen such that each rotor's thrust and torque would generate body z-axis yaw moments in the same directions. We select set $\mathcal{R}$ of all possible two-rotor failure cases. Regarding $\bar{\kappa}$, it is assumed that the rectangular cuboid is centered at the origin, implying that an equal amount of control authority is desired in both positive and negative directions for each dimension of $\mathcal{M}(\boldsymbol{d})$. As a result, additional constraints are added to the vanilla optimization problem (5.4):

$$\underline{\zeta}_i = -\bar{\zeta}_i, \quad \bar{\zeta}_i > 0, \quad i = 1, 2, 3.$$

Furthermore, we assume that the ability to control roll, pitch, and yaw is of equal importance and let $\beta_1 = \beta_2 = \beta_3 = 1$.

The optimal quality measure $\bar{\kappa}^*_{\mathcal{R}}$ on $\mathcal{D}$ is depicted in Figure 5.2, which also shows the optimal design parameter as

$$\boldsymbol{d}^* = [\theta_1^*; \theta_2^*] = [19^\circ; 13^\circ].$$

When there is no rotor failure, the loss in maximum vertical thrust in the body z-direction due to the tilt angles is relatively small; only 4% loss compared to the case of rotors without any tilt.

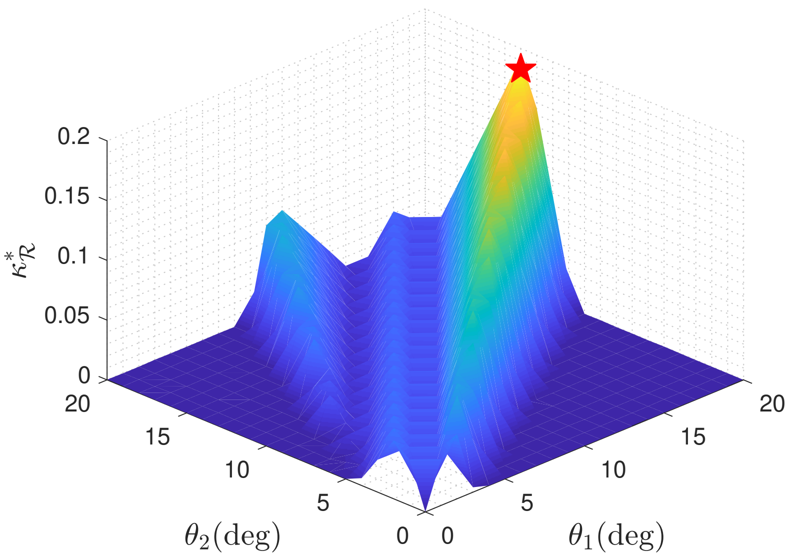


Figure 5.2: Quality measure $\bar{\kappa}^*_{\mathcal{R}}$ on $\mathcal{D}$. The optimal design parameters are marked with a red star.

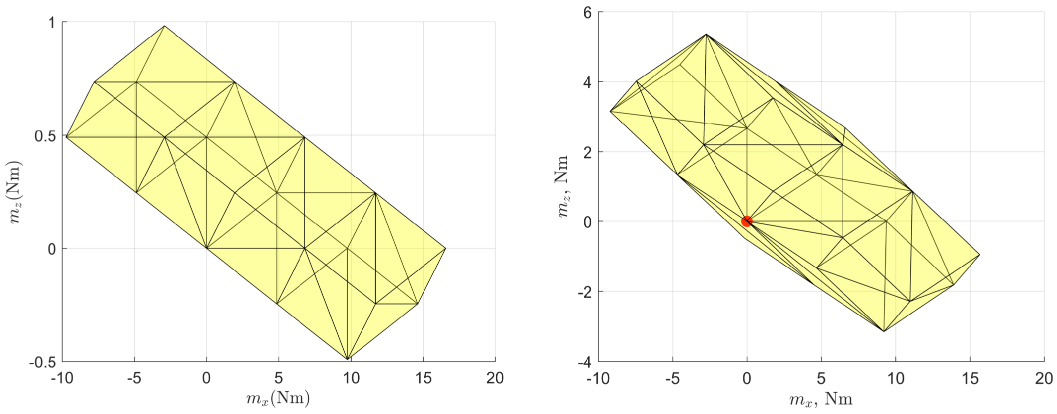
Table 5.1: Comparison of quality measures $\bar{\kappa}^*_r(\boldsymbol{d}^*)$ and $\bar{\kappa}^*_r(\boldsymbol{d}_0)$ over failure set $\mathcal{R}$

$r \in \mathcal{R}$	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)	(1,8)	(2,3)	(2,4)	(2,5)
$\bar{\kappa}^*_r(\boldsymbol{d}^*)$	**2.04**	**1.01**	**0.20**	**0.20**	**2.09**	**1.89**	**1.89**	**0.20**	**1.01**	**2.09**
$\bar{\kappa}^*_r(\boldsymbol{d}_0)$	0.48	0.23	0.16	0.09	0.48	0.44	0.44	0.16	0.23	0.48
$r \in \mathcal{R}$	(2,6)	(2,7)	(2,8)	(3,4)	(3,5)	(3,6)	(3,7)	(3,8)	(4,5)	(4,6)
$\bar{\kappa}^*_r(\boldsymbol{d}^*)$	**0.20**	**1.89**	**1.89**	**1.34**	**1.12**	**1.34**	**0.19**	**1.34**	**1.34**	**1.12**
$\bar{\kappa}^*_r(\boldsymbol{d}_0)$	0.09	0.44	0.44	0.33	0.33	0.33	0.00	0.33	0.33	0.33
$r \in \mathcal{R}$	(4,7)	(4,8)	(5,6)	(5,7)	(5,8)	(6,7)	(6,8)	(7,8)		
$\bar{\kappa}^*_r(\boldsymbol{d}^*)$	**1.34**	**0.19**	**2.58**	**1.59**	**1.08**	**1.08**	**1.59**	**1.58**		
$\bar{\kappa}^*_r(\boldsymbol{d}_0)$	0.33	0.00	0.57	0.41	0.15	0.15	0.41	0.40		

5.3.2 Comparison of Controllability with Rotor Failure

A comparison between the optimized design $\boldsymbol{d}^*$ and a baseline design $\boldsymbol{d}_0 = [0, 0]^\top$ over the fail set $\mathcal{R}$ is shown in Table 5.1. One can see that $\bar{\kappa}^*_r(\boldsymbol{d}^*) > \bar{\kappa}^*_r(\boldsymbol{d}_0)$ for all $r \in \mathcal{R}$.

Two notable cases are $r = (3, 7)$ and $r = (4, 8)$, where the optimal quality measures are 0.0 for $\boldsymbol{d}_0$. In these cases, $\boldsymbol{\tau}_T = \boldsymbol{0}$ is on the boundary of $\mathcal{M}_r(\boldsymbol{d}_0)$, and yaw moment is coupled with roll moment as shown in Figure 5.3a. For instance, it is not possible to generate a negative yaw moment and a negative roll moment at the



(a) $\mathcal{M}_r(\boldsymbol{d}_0)$, $r = (3, 7)$ (b) $\mathcal{M}_r(\boldsymbol{d}^*)$, $r = (3, 7)$

Figure 5.3: Moment authorities for the baseline and the optimized designs during rotor failure $r = (3, 7)$. (a) Note that the origin (marked with a red dot) is on the boundary. Because of this, generation of roll and yaw moments is coupled. (b) Independent generation of roll and yaw moments is possible in this case and the aircraft is null controllable. The axes are not drawn to scale.

same time for the case of $r = (3, 7)$. Indeed, the linear analysis shows that these cases are not controllable around static hover [116]. In such cases, one may give up yaw authority and focus on roll control, as is commonly done in the literature, since roll directly affects position control. However, this may not be a desirable behavior for some multirotor aircraft, such as in UAM applications. This difficulty stems from the fact that the two failed rotors have the same rotation direction, and the aircraft suddenly loses a large amount of its capability to generate yaw moment in one direction when they fail simultaneously. Furthermore, these rotors are located on the same side with respect to the body x-axis, and the failure of these rotors significantly reduces the amount of achievable roll moment in one direction.

The optimized design $\boldsymbol{d}^*$, on the other hand, ensures that $\boldsymbol{\tau}_T = \boldsymbol{0}$ is in the interior of $\mathcal{M}_r(\boldsymbol{d}^*)$ and maintains null controllability of the aircraft after the failure of the same sets of rotors shown in Figure 5.3b. In this case, tilted rotors generate additional yaw moment from the body x-axis forces, which compensates for the loss of yaw moment from rotor failure. As a result, the coupling between roll and yaw moments is resolved near the origin, and both moments can independently be generated in positive and negative directions, without having to give up yaw control authority.

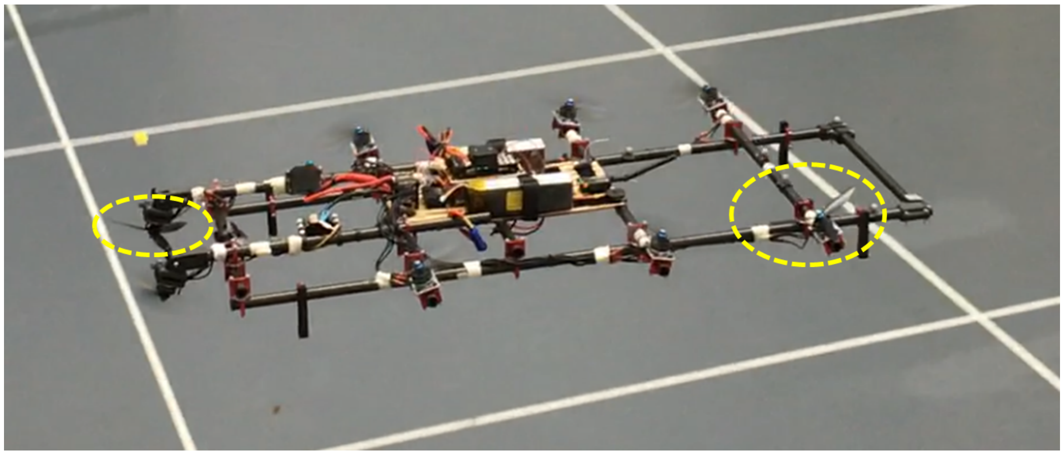
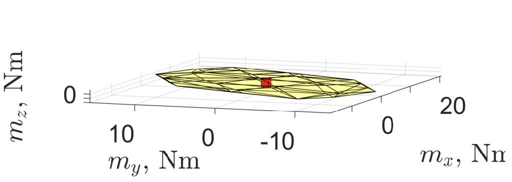


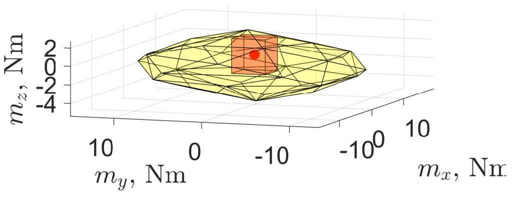
Figure 5.4: The prototype in flight. The two rotors indicated by dotted circles are intentionally failed.

5.3.3 Experiments on Prototype Hardware

Experiments are conducted on the prototype aircraft to test the aircraft's controllability after rotor failure. We use a Pixhawk flight controller [107] to control the prototype, with changes made to the open-source firmware to enable custom control allocation functions obtained by solving (3.44) from Section 3.4.2.



(a) $\mathcal{M}_r(\boldsymbol{d}_0)$, $r = (1, 8)$

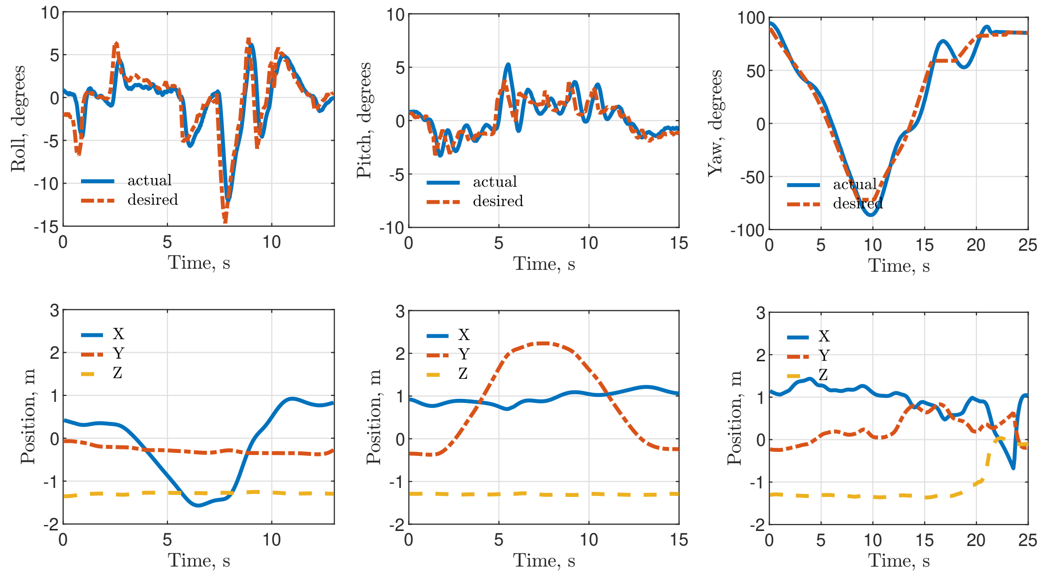


(b) $\mathcal{M}_r(\boldsymbol{d}^*)$, $r = (1, 8)$

Figure 5.5: Moment authorities for the baseline and the optimized designs during rotor failure $r = (1, 8)$. Red boxes represent the corresponding $\bar{\kappa}_r^*$.

In the experiments, we intentionally fail rotors No.1 and No.8 of the aircraft (Figure 5.4). The resulting moment sets for $\boldsymbol{d}_0$ and $\boldsymbol{d}^*$ are shown in Figure 5.5, which shows that the aircraft has little yaw authority with $\boldsymbol{d}_0$, although it is able to generate a reasonable amount of roll and pitch moments after the rotor failure. On the other hand, the aircraft with $\boldsymbol{d}^*$ is expected to generate a reasonable amount of roll, pitch, and yaw moments.

In order to compare the null controllability of the prototype between $\boldsymbol{d}_0$ and $\boldsymbol{d}^*$, we configured the prototype with both settings across tests. During flight, Euler angles are perturbed individually through manual remote control. The perturbation is made in both directions of each Euler angle axis to check the directional dependency of moment generation. We record the evolution of the vehicle's angular states as well as its position to see how well the prototype reaches static hover. The experiments are conducted without changing position and attitude controllers so that the effect of design change on the flight performance is solely verified.

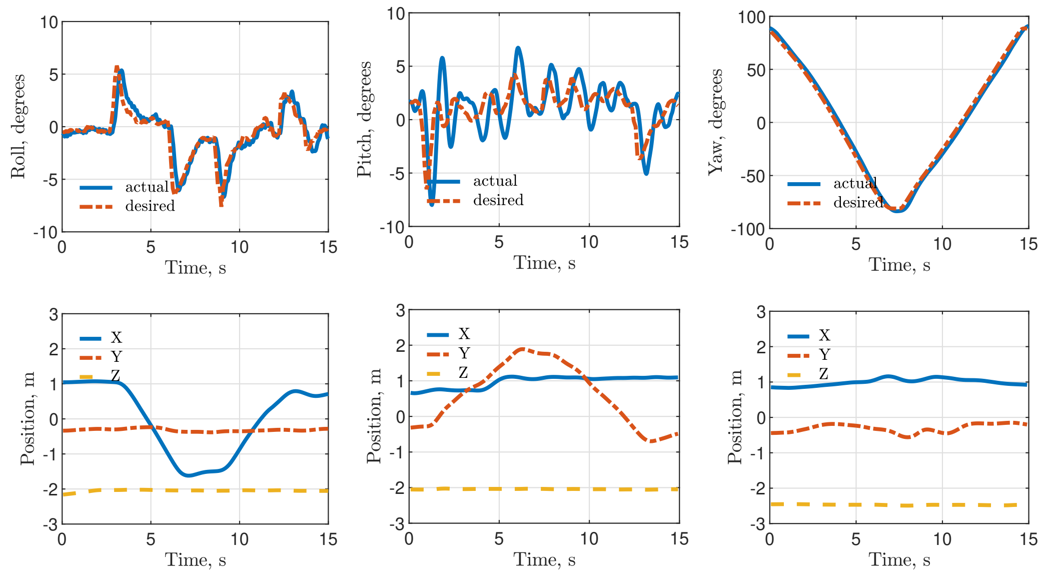


(a) Roll Perturbation (b) Pitch Perturbation (c) Yaw Perturbation

Figure 5.6: Euler angles and inertial position of the prototype with $\boldsymbol{d}_0$ after perturbations are made in each axis. The prototype is able to track roll and pitch commands as well as negative yaw commands. However, it started to become unstable at around $t = 10$ s in (c) while yawing in a positive direction.

The result of $\boldsymbol{d}_0$ is shown in Figure 5.6. Figures 5.6a and 5.6b show that the prototype is able to track roll and pitch commands and arrive at static hover once the perturbations are removed. Regarding yaw, the prototype is able to track negative yaw command while holding its position; however, yawing in positive direction caused instability to the prototype due to its limited capability to generate positive yaw moment, and the prototype is not able to hold its position anymore. This observation agrees with our discussion in Section 5.3.2, both rotors No. 1 and No.8 were in charge of generating positive yaw moment and their failure made it difficult for the prototype to yaw in positive direction.

The result of $\boldsymbol{d}^*$ is shown in Figure 5.7. As expected, the prototype now remains null controllable after rotor failure and is able to track all of roll, pitch, and yaw commands, reaching static hover when the perturbations are removed, drastically improving over the baseline design.



(a) Roll Perturbation (b) Pitch Perturbation (c) Yaw Perturbation

Figure 5.7: Euler angles and inertial position of the prototype with $\boldsymbol{d}^*$ after perturbations are made in each axis. The prototype is able to track roll, pitch, and yaw commands in both directions.

5.4 Chapter Summary

This chapter presented a novel design optimization method for multirotor aircraft robust to rotor failures, with the goal of maximizing a quality measure derived from the notion of null controllability that is related to the aircraft's ability to reach static hover. The design procedure was illustrated in detail with the Autonomous Flying Ambulance model being developed at Caltech's Center for Autonomous Systems and Technologies. We compared the controllability of the optimized design with a baseline configuration and showed that the optimized design was able to maintain null controllability for the failure cases with which the baseline could not. We also validated our results with a set of hardware experiments using the prototype aircraft, and showed that the prototype configured with the optimal design parameters stabilized its attitude against perturbations in all angular states even after losing two rotors.

Chapter 6

DEEP LEARNING FOR FLIGHT CONTROL

To capture complex aerodynamic interactions without being overly-constrained by conventional modeling assumptions, we take a machine-learning (ML) approach to build a black-box force model using Deep Neural Networks (DNNs). However, incorporating such models into a flight controller faces three key challenges. First, collecting sufficient real-world training data is difficult, as DNNs are notoriously data-hungry. Second, due to high-dimensionality, DNNs can be unstable to train and generate an unpredictable output, which makes the system susceptible to instability during control loop. Third, DNNs are often difficult to analyze for designing provably stable DNN-based controllers.

Consider the same dynamics as in Chapter 3 for fixed-wing VTOL. The dynamics from (3.1) and (3.2) is valid for any 6 DOF rigid body vehicle. High precision position tracking for aerial vehicles require accurate estimation of forces and moments. In particular, $\boldsymbol{f}_T$ and $\boldsymbol{\tau}_T$ are typically easy to model and identify beforehand, such as $\boldsymbol{w}_T = \boldsymbol{B}_T\boldsymbol{u}_T$ (3.39). However, $\boldsymbol{f}_A$ and $\boldsymbol{\tau}_A$ become particularly hard to model for fixed-wing VTOL aircraft. Complex interactions among rotors, wing surfaces, and the environments pose challenges for conventional formulation using physics principles. We will first present prior result from [117] that incorporates DNN with known vehicle dynamics for control design, then introduce methods to train controllers for general DNN based dynamical models.

6.1 Dynamics Learning using DNN

We elect to learn the unknown force model using a DNN with Rectified Linear Units (ReLU) activation. In general, DNNs equipped with ReLU converge faster during training, demonstrate more robust behavior with respect to changes in hyperparameters, and have fewer vanishing gradient problems compared to other activation functions such as *sigmoid* [118].

6.1.1 ReLU Deep Neural Networks

A ReLU deep neural network with L hidden layers represents the functional mapping from the input $\boldsymbol{x}$ to the output $f(\boldsymbol{x}, \boldsymbol{\theta})$:

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{W}^{L+1}\phi_{\text{act}}\left(\boldsymbol{W}^{L}\left(\phi_{\text{act}}\left(\boldsymbol{W}^{L-1}\left(\cdots\phi_{\text{act}}(\boldsymbol{W}^{1}\boldsymbol{x})\cdots\right)\right)\right)\right), \tag{6.1}$$

parameterized by DNN weights $\boldsymbol{\theta} = \{\boldsymbol{W}^1, \cdots, \boldsymbol{W}^{L+1}\}$, where the activation function $\phi_{\text{act}}(\cdot) = \max(\cdot, 0)$ is called the element-wise ReLU function. ReLU is less computationally expensive than *tanh* and *sigmoid* because it involves simpler mathematical operations. However, deep neural networks are usually trained by first-order gradient based optimization, which is highly sensitive on the curvature of the training objective and can be unstable [119]. To alleviate this issue, we apply the spectral normalization technique [70].

6.1.2 Spectral Normalization with Specified Lipschitz Constant

Spectral normalization stabilizes DNN training by constraining the Lipschitz constant of the objective function. Spectrally normalized DNNs have also been shown to generalize well [120], which is an indication of stability in machine learning. Mathematically, the Lipschitz constant of a function $\|f\|_{\text{Lip}}$ is defined as the smallest value such that

$$\forall\, \boldsymbol{x}, \boldsymbol{x}' : \ \|f(\boldsymbol{x}) - f(\boldsymbol{x}')\| \leq \|f\|_{\text{Lip}}\, \|\boldsymbol{x} - \boldsymbol{x}'\|\,.$$

It is known that the Lipschitz constant of a general differentiable function f is the maximum spectral norm (maximum singular value) of its gradient over its domain $\|f\|_{\text{Lip}} = \sup_{\boldsymbol{x}} \sigma_{\text{sn}}(\nabla f(\boldsymbol{x}))$.

The ReLU DNN in (6.1) is a composition of functions. Thus we can bound the Lipschitz constant of the network by constraining the spectral norm of each layer. For linear maps $g(\boldsymbol{x}) = \boldsymbol{W}^l\boldsymbol{x}$, the spectral norm is given by

$$\|g\|_{\text{Lip}} = \sup_{\boldsymbol{x}} \sigma_{\text{sn}}(\nabla g(\boldsymbol{x})) = \sup_{\boldsymbol{x}} \sigma_{\text{sn}}(\boldsymbol{W}^l) = \sigma_{\text{sn}}(\boldsymbol{W}^l).$$

For ReLU activation function $\phi_{\text{act}}(\cdot)$, the Lipschitz constant is equal to 1. With inequality $\|g_1 \circ g_2\|_{\text{Lip}} \leq \|g_1\|_{\text{Lip}} \cdot \|g_2\|_{\text{Lip}}$, we can find the following bound:

$$\|f\|_{\text{Lip}} \leq \prod_{l=1}^{L+1} \sigma_{\text{sn}}(\boldsymbol{W}^l). \tag{6.2}$$

In order to regulate the Lipschitz constant, we can apply spectral normalization to the weight matrices in each layer l during training as

$$\bar{\boldsymbol{W}}^l = \boldsymbol{W}^l/\sigma_{\text{sn}}(\boldsymbol{W}^l) \cdot \gamma^{\frac{1}{L+1}}, \tag{6.3}$$

where γ is the intended Lipschitz constant for the DNN. The following lemma bounds the Lipschitz constant of a ReLU DNN with spectral normalization.

Lemma 6.1. *For a multi-layer ReLU network $f(\boldsymbol{x}, \boldsymbol{\theta})$, defined in (6.1) without an activation function on the output layer. Using spectral normalization, the Lipschitz constant of the entire network satisfies:*

$$\left\|f(\boldsymbol{x}, \bar{\boldsymbol{\theta}})\right\|_{\text{Lip}} \leq \gamma,$$

with spectrally-normalized parameters $\bar{\boldsymbol{\theta}} = \bar{\boldsymbol{W}}^1, \cdots, \bar{\boldsymbol{W}}^{L+1}$.

Proof. From (6.2), the Lipschitz constant can be written as the product of spectral norms over all layers. The proof follows from (6.3). ■

6.1.3 Constrained Training

Estimating $\boldsymbol{f}_A$ boils down to optimizing the parameters $\boldsymbol{\theta}$ in (6.1) with a constrained Lipschitz constant, given a collected data of N entries: input feature $\boldsymbol{x}^{(i)}$ and output $\boldsymbol{y}^{(i)}$ with $i = 1, \cdots, N$.

The optimization problem is stated as

$$\begin{aligned} \min_{\boldsymbol{\theta}} \quad & \sum_{i=1}^{N} \frac{1}{N} \left\|\boldsymbol{y}^{(i)} - \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{\theta})\right\| \\ \text{subject to} \quad & \|\boldsymbol{f}\|_{\text{Lip}} \leq \gamma. \end{aligned} \tag{6.4}$$

In our case, $\boldsymbol{y}^{(i)}$ is the observed disturbance forces and $\boldsymbol{x}^{(i)}$ is the observed states and control inputs. According to the upper bound in (6.2), we can satisfy the constraint by regulating the spectral norm of the weights in each layer. We use stochastic gradient descent (SGD) to optimize (6.4) and apply spectral normalization to regulate the weights. From Lemma 6.1, the trained ReLU DNN has a Lipschitz constant γ.

6.2 Flight Control for partial DNN Dynamics

Different from before, let us represent $\boldsymbol{f}_A$ in inertia frame here for simplicity. Thus the position dynamics is now

$$\dot{\boldsymbol{p}} = \boldsymbol{v}, \qquad \dot{\boldsymbol{v}} = \boldsymbol{g} + \boldsymbol{R}\boldsymbol{f}_T + \boldsymbol{f}_A \tag{6.5}$$

Our controller for 3D trajectory tracking is constructed as a nonlinear feedback linearization controller whose stability guarantees are obtained using the spectral normalization of the DNN-based dynamics model. We exploit the Lipschitz property of the DNN to solve for the resulting control input using fixed-point iteration.

6.2.1 Nonlinear Controller with Fixed-point Iteration on DNN

We start from our baseline nonlinear position controller (3.19) in Chapter 3. Using the methods described in Section 6.1, we define $\hat{\boldsymbol{f}}_A(\boldsymbol{x}, \boldsymbol{u}_T)$ as the DNN approximation to the disturbance aerodynamic forces, with $\boldsymbol{x}$ being the partial states used as input features to the network. We design the total desired rotor force $\boldsymbol{f}_d$ as

$$\boldsymbol{f}_d = \bar{\boldsymbol{f}} - \hat{\boldsymbol{f}}_A, \quad \text{with} \quad \bar{\boldsymbol{f}} = -\boldsymbol{g} + \dot{\boldsymbol{v}}_r - \boldsymbol{K}_v \tilde{\boldsymbol{v}}. \tag{6.6}$$

Substituting (6.6) into (3.1), the closed-loop dynamics would simply become

$$\dot{\tilde{\boldsymbol{v}}} + \boldsymbol{K}_v \dot{\tilde{\boldsymbol{v}}} = \boldsymbol{\epsilon}_A,$$

with approximation error $\boldsymbol{\epsilon}_A = \boldsymbol{f}_A - \hat{\boldsymbol{f}}_A$. Hence from Proposition 3.1, $\tilde{\boldsymbol{p}}(t) \to \boldsymbol{0}$ globally and exponentially with bounded error, as long as $\|\boldsymbol{\epsilon}_A\|$ is bounded.

Consequently, desired total thrust $\bar{T}$ and force direction $\hat{\boldsymbol{k}}_f$ can be computed as

$$\bar{T} = \boldsymbol{f}_d \cdot \hat{\boldsymbol{k}}_f, \quad \text{and} \quad \hat{\boldsymbol{k}}_f = \boldsymbol{f}_d / \|\boldsymbol{f}_d\|, \tag{6.7}$$

with $\hat{\boldsymbol{k}}_f$ being the unit vector of rotor thrust direction (typically z-axis in multirotors). This is similar to (3.32) and (3.36), with $\boldsymbol{f}_d$ defined differently based on DNN approximation $\hat{\boldsymbol{f}}_A$.

We assume the same nonlinear attitude controller (3.28) that uses desired torque $\bar{\boldsymbol{\tau}}$ to track $\boldsymbol{R}_d(t)$. In this case, $\boldsymbol{\tau}_A$ is ignored and we can use solely rotor torque $\boldsymbol{\tau}_T = \bar{\boldsymbol{\tau}}$. Thus from Proposition 3.2, exponential trajectory tracking of a desired attitude $\boldsymbol{R}_d(t)$ is guaranteed within some bounded error in the presence of bounded disturbance torques.

From (3.28) and (3.39), we can relate the desired wrench $\boldsymbol{w}_T = [\bar{T}, \boldsymbol{\tau}_T^\top]^\top$ with the control signal $\boldsymbol{u}_T$ through

$$\boldsymbol{B}_T \boldsymbol{u}_T = \boldsymbol{w}_T = \begin{bmatrix} \left(\bar{\boldsymbol{f}} - \hat{\boldsymbol{f}}_A(\boldsymbol{x}, \boldsymbol{u}_T)\right) \cdot \hat{\boldsymbol{k}}_f \\ \boldsymbol{\tau}_T \end{bmatrix}. \tag{6.8}$$

Because of the dependency of $\hat{\boldsymbol{f}}_A$ on $\boldsymbol{u}_T$, the control synthesis problem here is nonaffine. Therefore, we propose the following fixed-point iteration method for solving (6.8):

$$\boldsymbol{u}_{T,k} = \boldsymbol{B}_T^+ \begin{bmatrix} \left(\bar{\boldsymbol{f}} - \hat{\boldsymbol{f}}_A(\boldsymbol{x}, \boldsymbol{u}_{T,k-1})\right) \cdot \hat{\boldsymbol{k}}_f \\ \boldsymbol{\tau}_T \end{bmatrix} \tag{6.9}$$

where $\boldsymbol{u}_{T,k}$ and $\boldsymbol{u}_{T,k-1}$ are the control input for current and previous time-step in the discrete-time controller. $\boldsymbol{B}_T^+$ denotes either the inverse or right pseudoinverse depending on whether $\boldsymbol{B}_T$ is fully or over actuated. Next, we prove the stability of the system and the convergence of the control inputs in (6.9).

6.2.2 Stability Analysis

In order to properly prove the stability of the closed-loop system, we have to incorporate a discrete-time controller with continuous dynamics. The closed-loop tracking error analysis provides a direct correlation on how to tune the neural network and controller parameter to improve control performance and robustness. We first show that the control input $\boldsymbol{u}_{T,k}$ converges to the solution of (6.8) when all states are fixed.

Lemma 6.2. *Define mapping $\boldsymbol{u}_{T,k} = \mathcal{F}(\boldsymbol{u}_{T,k-1})$ based on (6.9) and fix $\boldsymbol{x}$:*

$$\mathcal{F}(\boldsymbol{u}) = \boldsymbol{B}_T^{+} \begin{bmatrix} \left(\bar{\boldsymbol{f}} - \hat{\boldsymbol{f}}_A(\boldsymbol{x}, \boldsymbol{u})\right) \cdot \hat{\boldsymbol{k}}_f \\ \boldsymbol{\tau}_T \end{bmatrix}. \tag{6.10}$$

If $\hat{\boldsymbol{f}}_A(\boldsymbol{x}, \boldsymbol{u}_T)$ is L_A-Lipschitz continuous, and $\sigma_{\text{sn}}(\boldsymbol{B}_T^{+}) \cdot L_A < 1$; then $\mathcal{F}(\cdot)$ is a contraction mapping, and $\boldsymbol{u}_{T,k}$ converges to a unique solution of $\boldsymbol{u}^* = \mathcal{F}(\boldsymbol{u}^*)$.

Proof. $\forall\, \boldsymbol{u}_1, \boldsymbol{u}_2 \in \mathcal{U}_T$ with $\mathcal{U}_T$ being a compact set of feasible control inputs; and given the fixed states as $\bar{\boldsymbol{f}}$, $\boldsymbol{\tau}_T$ and $\hat{\boldsymbol{k}}_f$, then:

$$\begin{aligned} \|\mathcal{F}(\boldsymbol{u}_1) - \mathcal{F}(\boldsymbol{u}_2)\| &= \left\|\boldsymbol{B}_T^{+}\left(\hat{\boldsymbol{f}}_A(\boldsymbol{x}, \boldsymbol{u}_1) - \hat{\boldsymbol{f}}_A(\boldsymbol{x}, \boldsymbol{u}_2)\right)\right\| \\ &\leq \sigma_{\text{sn}}(\boldsymbol{B}_T^{+}) \cdot L_A \left\|\boldsymbol{u}_1 - \boldsymbol{u}_2\right\|. \end{aligned}$$

Thus, $\exists\, \alpha < 1,$ s.t $\|\mathcal{F}(\boldsymbol{u}_1) - \mathcal{F}(\boldsymbol{u}_2)\| < \alpha \|\boldsymbol{u}_1 - \boldsymbol{u}_2\|$. Hence, $\mathcal{F}(\cdot)$ is a contraction mapping. ■

Before continuing to prove the stability of the full system, we make the following assumptions.

Assumption 6.1. *The desired states along the position trajectory $\boldsymbol{p}_d(t)$, $\dot{\boldsymbol{p}}_d(t)$, and $\ddot{\boldsymbol{p}}_d(t)$ are bounded.*

Assumption 6.2. *One-step difference of control signal satisfies $\left\|\boldsymbol{u}_{T,k} - \boldsymbol{u}_{T,k-1}\right\| \leq \kappa \|\tilde{\boldsymbol{v}}\|$ with a small positive κ.*

Here we provide the intuition behind this assumption. From (6.10), we can derive the following approximate relation with $\Delta(\cdot)_k = \|(\cdot)_k - (\cdot)_{k-1}\|$:

$$\begin{aligned} \Delta \boldsymbol{u}_{T,k} \leq \sigma_{\text{sn}}(\boldsymbol{B}_T^{+})\Big(&L_A \Delta \boldsymbol{u}_{T,k-1} + L_A \Delta \boldsymbol{x}_k \\ &+ \Delta \dot{\boldsymbol{v}}_{r,k} + \lambda_{\max}(\boldsymbol{K}_v) \Delta \tilde{\boldsymbol{v}}_k + \Delta \bar{\boldsymbol{\tau}}_k\Big). \end{aligned}$$

Because the update rate of attitude controller ($> 100\,\mathrm{Hz}$) and motor speed control ($> 5\,\mathrm{kHz}$) are much higher than that of the position controller ($\approx 10\,\mathrm{Hz}$) in practice, we neglect $\Delta\tilde{\boldsymbol{v}}_k$, $\Delta\dot{\boldsymbol{v}}_{r,k}$, and $\Delta\boldsymbol{x}_k$ in one update (Theorem 11.1 [95]). Furthermore, $\Delta\bar{\boldsymbol{\tau}}_k$ can be limited internally by the attitude controller. It leads to:

$$\Delta\boldsymbol{u}_{T,k} \le \sigma_{\mathrm{sn}}(\boldsymbol{B}_T^{+})\big(L_A \Delta\boldsymbol{u}_{T,k-1} + c\big),$$

with c being a small constant and $\sigma_{\mathrm{sn}}(\boldsymbol{B}_T^{+}) \cdot L_A < 1$ from Lemma 6.2. From here, we can deduce that $\Delta\boldsymbol{u}_T$ rapidly converges to a small ultimate bound between each position controller update.

Assumption 6.3. *The learning error of $\hat{\boldsymbol{f}}_A(\boldsymbol{x}, \boldsymbol{u}_T)$ over the compact sets $\boldsymbol{x} \in \mathcal{X}$, $\boldsymbol{u}_T \in \mathcal{U}_T$ is upper bounded by $\epsilon_m = \sup_{\boldsymbol{x}\in\mathcal{X}, \boldsymbol{u}_T\in\mathcal{U}_T} \|\boldsymbol{\epsilon}_A(\boldsymbol{x}, \boldsymbol{u}_T)\|$, where $\boldsymbol{\epsilon}_A(\boldsymbol{x}, \boldsymbol{u}_T) = \boldsymbol{f}_A(\boldsymbol{x}, \boldsymbol{u}_T) - \hat{\boldsymbol{f}}_A(\boldsymbol{x}, \boldsymbol{u}_T)$.*

DNNs have been shown to generalize well to sets of unseen data that are from almost the same distribution as the training set [121], [122]. This empirical observation is also theoretically studied in order to shed more light on an understanding of the complexity of these models [120], [123]–[125]. Based on the above assumptions, we can now present our overall stability and robustness result.

Theorem 6.3. *Under Assumptions 6.1 to 6.3, for a time-varying $\boldsymbol{p}_d(t)$, the controller defined in (6.6) and (6.9) with $\lambda_{\min}(\boldsymbol{K}_v) > \kappa L_A$ achieves exponential convergence of $\tilde{\boldsymbol{v}}$ to error ball*

$$\lim_{t\to\infty} \|\tilde{\boldsymbol{v}}(t)\| = \frac{\epsilon_m}{\lambda_{\min}(\boldsymbol{K}_v) - \kappa L_A}$$

with rate $\lambda_{\min}(\boldsymbol{K}_v) - \kappa L_A$. $\tilde{\boldsymbol{p}}$ exponentially converges to error ball

$$\lim_{t\to\infty} \|\tilde{\boldsymbol{p}}(t)\| = \frac{\epsilon_m}{\lambda_{\min}(\boldsymbol{\Lambda})(\lambda_{\min}(\boldsymbol{K}_v) - \kappa L_A)} \tag{6.11}$$

with rate $\lambda_{\min}(\boldsymbol{\Lambda})$.

Proof. We begin the proof by selecting a Lyapunov function as $\mathcal{V}(\tilde{\boldsymbol{v}}) = \frac{1}{2}\|\tilde{\boldsymbol{v}}\|^2$, then by applying the controller (6.6), we get the time-derivative of $\mathcal{V}$:

$$\begin{aligned}\dot{\mathcal{V}} &= \tilde{\boldsymbol{v}}^\top\Big(-\boldsymbol{K}_v\tilde{\boldsymbol{v}} + \hat{\boldsymbol{f}}_A(\boldsymbol{x}_k, \boldsymbol{u}_{T,k}) - \hat{\boldsymbol{f}}_A(\boldsymbol{x}_k, \boldsymbol{u}_{T,k-1}) + \boldsymbol{\epsilon}_A(\boldsymbol{x}_k, \boldsymbol{u}_{T,k})\Big)\\ &\le -\tilde{\boldsymbol{v}}^\top\boldsymbol{K}_v\tilde{\boldsymbol{v}} + \|\tilde{\boldsymbol{v}}\|\left(\left\|\hat{\boldsymbol{f}}_A(\boldsymbol{x}_k, \boldsymbol{u}_{T,k}) - \hat{\boldsymbol{f}}_A(\boldsymbol{x}_k, \boldsymbol{u}_{T,k-1})\right\| + \epsilon_m\right).\end{aligned}$$

Let $\lambda = \lambda_{\min}(\boldsymbol{K}_v)$ denote the minimum eigenvalue of the positive-definite matrix $\boldsymbol{K}_v$. By applying the Lipschitz property of $\hat{\boldsymbol{f}}_A$ Lemma 6.1 and Assumption 6.2, we obtain

$$\dot{\mathcal{V}} \leq -2\left(\lambda - \kappa L_A\right)\mathcal{V} + \sqrt{2\mathcal{V}}\epsilon_m$$

Using the Comparison Lemma [95], we define $\mathcal{W}(t) = \sqrt{\mathcal{V}(t)} = \sqrt{1/2}\|\tilde{\boldsymbol{v}}\|$ and $\dot{\mathcal{W}} = \dot{\mathcal{V}}/\left(2\sqrt{\mathcal{V}}\right)$ to obtain

$$\|\tilde{\boldsymbol{v}}(t)\| \leq \|\tilde{\boldsymbol{v}}(t_0)\| \exp\left(-(\lambda - \kappa L_A)(t - t_0)\right) + \frac{\epsilon_m}{\lambda - \kappa L_A}.$$

It can be shown that this leads to finite-gain $\mathcal{L}_p$ stability and input-to-state stability (ISS) [126]. Furthermore, the hierarchical combination between $\tilde{\boldsymbol{v}}$ and $\tilde{\boldsymbol{p}}$ results in $\lim_{t\to\infty}\|\tilde{\boldsymbol{p}}(t)\| = \lim_{t\to\infty}\|\tilde{\boldsymbol{v}}(t)\|/\lambda_{\min}(\boldsymbol{\Lambda})$, yielding (6.11). ■

By designing the controller gain $\boldsymbol{K}_v$ and Lipschitz constant L_A of the DNN, we can ensure that $\lambda - \kappa L_A > 0$ and achieve exponential tracking within bound.

6.3 Experiments on Quadrotor Drone

We evaluate both the generalization performance of our DNN as well as the overall control performance of our controller. We perform experiment on the Intel Aero Drone platform. The physical details of the drone is described in Appendix B.3. Figure 6.1 shows snapshots of the quadrotor in a landing experiment.

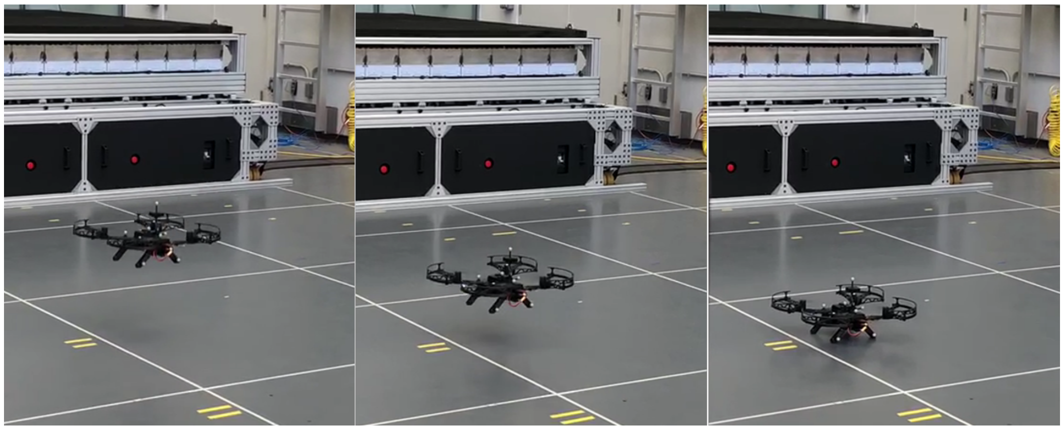


Figure 6.1: Snapshots of Intel Aero Drone during a landing task.

6.3.1 Flight Data Collection and Pre-processing

We collect data by having an expert pilot flying the drone at different heights and speeds. The collected data consists of sequences of relevant states and control inputs

$\{\boldsymbol{p}, \boldsymbol{v}, \boldsymbol{q}, \boldsymbol{\omega}, \boldsymbol{u}_T\}$. We then parse the data to get labels for $\boldsymbol{f}_A$ by using the relationship $\boldsymbol{f}_A = \dot{\boldsymbol{v}} - \boldsymbol{g} - R\boldsymbol{f}_T$ from (3.1) and (3.2), where $\boldsymbol{f}_T$ is calculated based on nominal C_T values.

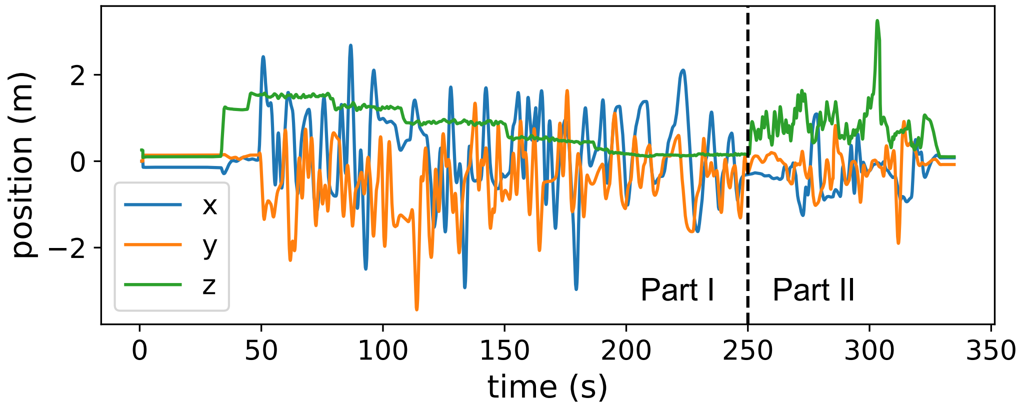


Figure 6.2: Position trajectory during data collection. Part I (0 to 250 s) contains constant height maneuvers (0.05 m to 1.50 m). Part II (250 s to 350 s) is dedicated to random free flight for maximum state-space coverage.

Our dataset is a single continuous trajectory with varying heights and velocities. The trajectory has two parts shown in Figure 6.2. Intuitively, we would like the DNN to learn height dependent ground effect on Part I of the trajectory, and other aerodynamics forces such as drag and rotor interactions on Part II.

6.3.2 Prediction Performance of Spectrally Normalized DNN

From the collected data, we construct a parsed dataset of N total entries

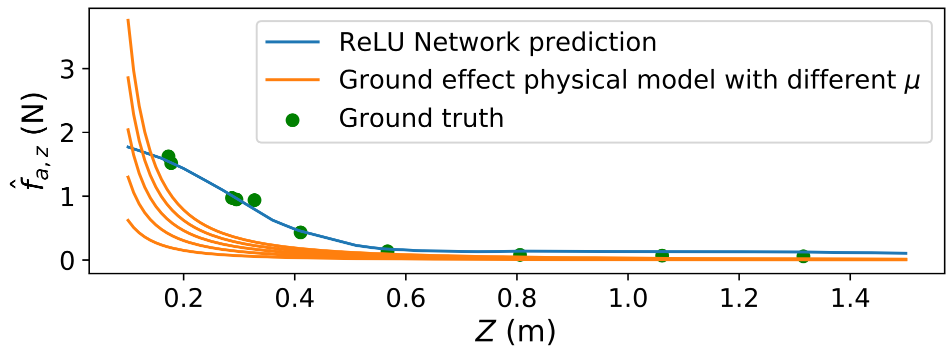
$$\mathcal{D} = \left\{[z; \boldsymbol{v}; \boldsymbol{q}; \boldsymbol{u}_T]^{(i)}, \boldsymbol{f}_A^{(i)}\right\} \quad \text{with} \quad i = 1, \cdots, N. \tag{6.12}$$

The input feature vector $\boldsymbol{x} = [z; \boldsymbol{v}; \boldsymbol{q}; \boldsymbol{u}_T] \in \mathbb{R}^{12}$, with z, $\boldsymbol{v}$, $\boldsymbol{q}$, $\boldsymbol{u}_T$ corresponds with the vehicle's height above ground-plane, global velocity, quaternion attitude, and control input. The output is the calculated force $\boldsymbol{f}_A \in \mathbb{R}^3$. The entire dataset is split into training (60%), test (20%), and validation set (20%) for hyper-parameter tuning. We train a deep ReLU network $\hat{\boldsymbol{f}}_A(\boldsymbol{x}, \boldsymbol{u}_T)$ with four fully-connected hidden layers using PyTorch [127], and use spectral normalization (6.3) to constrain its Lipschitz constant.

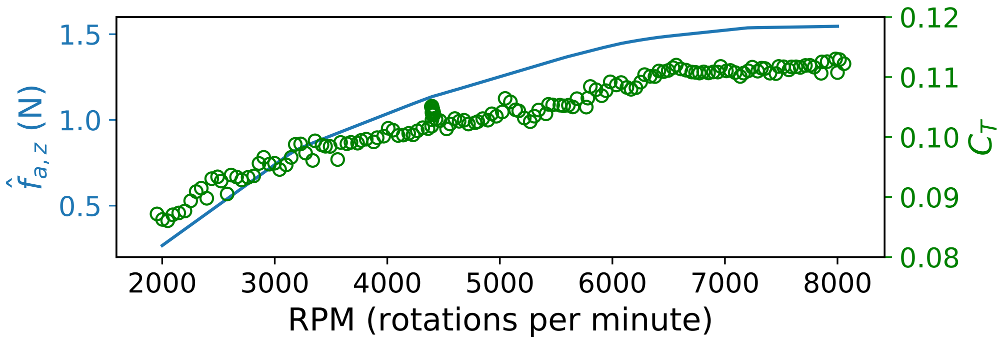
We first compare the near-ground estimation accuracy of our DNN model with an existing 1D steady-state ground effect model from [56], [128]. The physics-based model predicts the thrust force T of rotors when close to the ground:

$$T_{\text{ge}}(n,z) = \frac{n^2}{1-\mu(\frac{d}{8z})^2}k_T(n). \tag{6.13}$$

Here n is the rotor RPM, and μ depends on the number and the arrangement of rotors (i.e $\mu = 1$ for a single rotor, but must be tuned for multiple rotors). $k_T = C_T\rho d^4$ is the lumped thrust constant incorporating rotor diameter d and air density ρ. We can subtract our nominal thrust model $T_{\text{nom}} = k_T(n_0)n^2$ from the total force prediction to compare it with equivalent DNN prediction $\hat{f}_{A,z}$, with n_0 being the idle RPM.



(a) DNN $\hat{f}_{A,z}$ compared to the ground effect model vs. height z during steady hover. Ground truth is from hovering data at different heights.



(b) DNN $\hat{f}_{A,z}$ vs. RPM at $z = 0.2$ m and $v_z = 0$ m/s), compared to measured C_T from bench test.

Figure 6.3: Comparison of DNN $\hat{f}_{A,z}$ with physics-based ground effect model, and C_T from bench test.

Figure 6.3a shows the comparison between the DNN estimated $\boldsymbol{f}_A$ and physics-based ground effect model (6.13) at different heights. We can see that our network achieves much better prediction accuracy. We further investigate the trend of $\hat{f}_{A,z}$ with respect to n, and see a similar trend as bench tested C_T shown in Figure 6.3b.

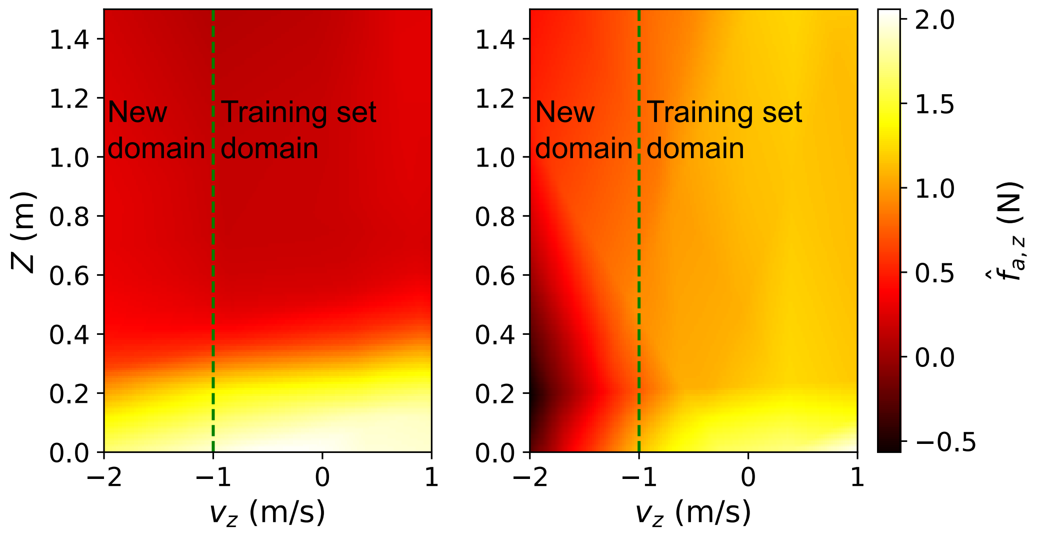


Figure 6.4: Heatmaps of learned $\hat{f}_{A,z}$ versus z and v_z. (Left) ReLU network with spectral normalization. (Right) ReLU network without spectral normalization.

To understand the benefits of spectral normalization, we compared $\hat{f}_{A,z}$ trained both with and without spectral normalization as shown in Figure 6.4. Note that $\{v_z \in [-1.0, 1.0]\ \mathrm{m/s}\}$ is covered in our training set, but $\{v_z \in [-2.0, 1.0]\ \mathrm{m/s}\}$ is not. We observe that: (**I**) Ground effect: $\hat{f}_{A,z}$ increases as z decreases, which is also shown in Figure 6.4(a); (**II**) Air drag: $\hat{f}_{A,z}$ increases as the drone speeds up and vice versa; (**III**) Generalization: the spectrally normalized DNN is smoother and can generalize to unseen feature domains not included in the training set.

In [120], the authors theoretically show that spectral normalization can provide tighter generalization guarantees on unseen data, which is consistent with our empirical observation. We will connect the generalization theory more tightly with our robustness guarantees in the future.

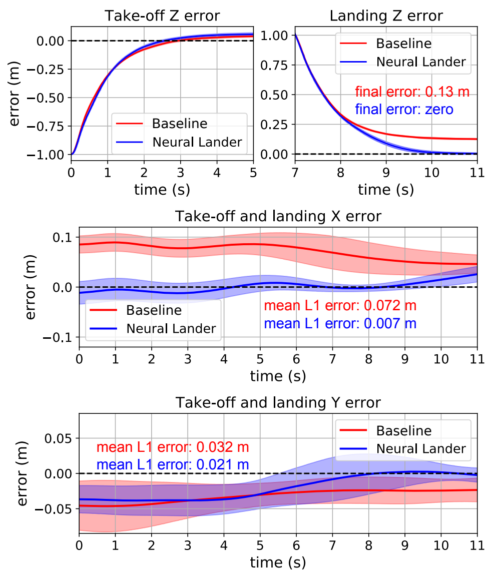
6.3.3 Near-ground Controller Performance

We implement the controller described in (6.6) and (6.9). For trajectory tracking tasks near-ground, we denote this method the *Neural-Lander*. We compared the *Neural-Lander* with a Baseline Nonlinear Tracking Controller, similar to (6.6) but with $\hat{\boldsymbol{f}}_A \equiv 0$, i.e $\boldsymbol{f}_d = \bar{\boldsymbol{f}}$. We also implement an integral feedback variation with

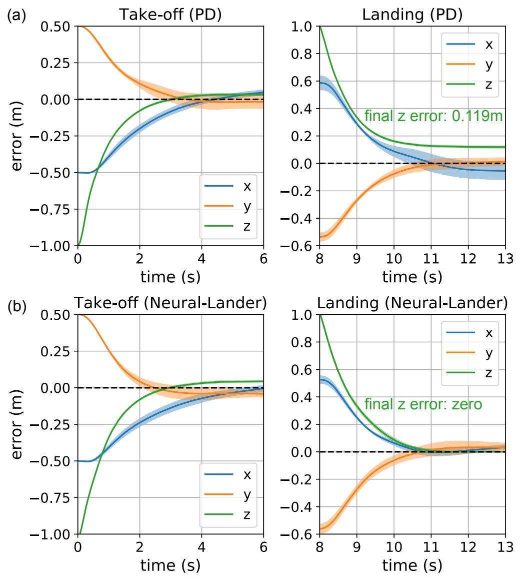
$$\boldsymbol{v}_r = \dot{\boldsymbol{p}}_d - 2\boldsymbol{\Lambda}\tilde{\boldsymbol{p}} - \boldsymbol{\Lambda}^2 \int_0^t \tilde{\boldsymbol{p}}(s)ds.$$

Though an integral gain can cancel steady-state error during set-point regulation, our flight results show that the performance can be sensitive to the magnitude of gain, especially during trajectory tracking. This can be seen in the demo video.[1]

First, we test the two controllers' performance in take-off and landing tasks. A 1D trajectory is constructed by commanding position setpoint $\boldsymbol{p}_d$ between $[0, 0, 0]$ and $[0, 0, 1]$. Similarly, a 3D trajectory is constructed with some lateral motions by having setpoints $[0, 0, 0]$ and $[0.5, -0.5, 1]$. We repeat the experiment 10 times and



(a) 1D Landing Trajectory



(b) 3D Landing Trajectory

Figure 6.5: Baseline Controller and *Neural-Lander* performance in take-off and landing. Means (solid curves) and standard deviations (shaded areas) of 10 trajectories.

compute the means and the standard deviations of the take-off/landing trajectories. From Figure 6.5, we conclude that there are two main benefits of our *Neural-Lander*: (**I**) *Neural-Lander* can control the drone to precisely and smoothly land on the ground surface while the Baseline Controller struggles to achieve 0 terminal height due to the ground effect. (**II**) *Neural-Lander* can mitigate drifts in the xy-plane, as it also learns additional aerodynamics effects such as air drag.

Second, we test the *Neural-Lander* performance with different DNN capacities. Figure 6.6 shows the necessity of having a deep neural network with more than two layers, as baseline, zero, and one layer models generated significant errors during take-off and landing.

Moreover, we observe that *Neural-Lander* without spectral normalization can even result in unexpected controller outputs leading to crash, which empirically implies

[1]Demo video: `https://youtu.be/FLLsG0S78ik`

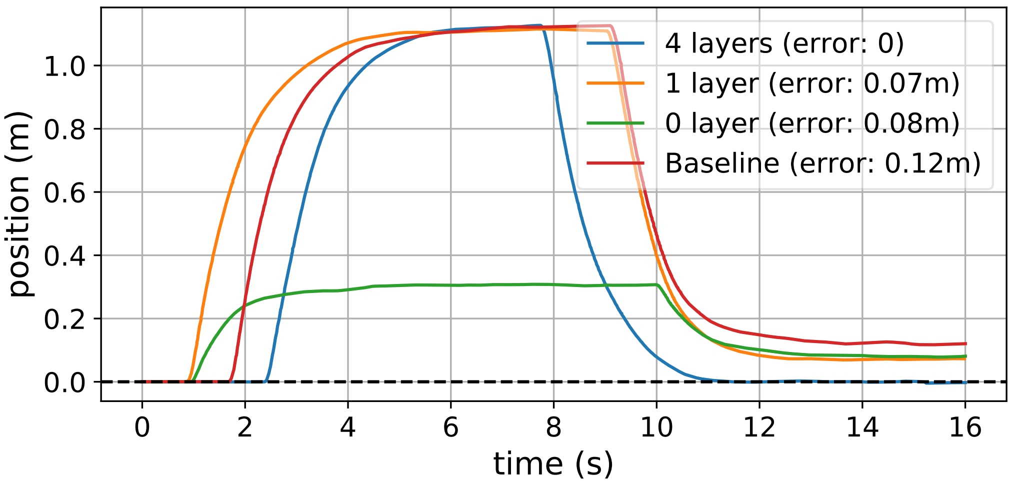


Figure 6.6: *Neural-Lander* performance in take-off and landing with different DNN capacities. 1 layer means $\hat{\boldsymbol{f}}_A = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$; 0 layer means $\hat{\boldsymbol{f}}_A = \boldsymbol{b}$.

that spectral normalization is essential in designing the DNN-based controller.

6.3.4 Near-object Controller Performance

To show that our algorithm can handle more complicated environments where physics-based modelling of dynamics would be substantially more difficult, we devise a task of tracking an elliptic trajectory very close to a table with a period of 10 seconds shown in Figure 6.7. The trajectory is partially over the table with significant ground effects, and a sharp transition to free space at the edge of the table.

We manually pilot the drone in regions close to the table to collect another dataset. We train a different ReLU DNN model with additional input features: $\hat{\boldsymbol{f}}_A(\boldsymbol{p}, \boldsymbol{v}, \boldsymbol{q}, \boldsymbol{u}_T)$. Similarly to the set-point experiment, the benefit of spectral normalization can be seen in Figure 6.7(a), where only the spectrally-normalized DNN exhibits a clear table boundary. Figure 6.7(b) shows that *Neural-Lander* outperforms Baseline Controller for tracking the desired position trajectory in all axes. Additionally, *Neural-Lander* exhibits a lower variance in height, even at the edge of the table, as the controller captures the changes in ground effects when the drone flies over the table.

6.4 General Control for DNN-based Dynamic Model

So far, our main focus has been on dynamical systems of the form

$$\dot{\boldsymbol{x}} = \boldsymbol{f}_0(\boldsymbol{x}) + \boldsymbol{B}_0(\boldsymbol{x})\boldsymbol{u} + \boldsymbol{f}_U(\boldsymbol{x}, \boldsymbol{u}), \tag{6.14}$$

with a known affine-in-control component $\boldsymbol{f}_0(\boldsymbol{x}) + \boldsymbol{B}_0(\boldsymbol{x})\boldsymbol{u}$ and an unknown general disturbance $\boldsymbol{f}_U(\boldsymbol{x}, \boldsymbol{u})$. For the flight control described in Section 6.2, the known inertial dynamics of position and attitude is affine with respect to the thrust signals $\boldsymbol{u}_T$; the unknown aerodynamics is approximated via a DNN $\hat{\boldsymbol{f}}_U(\boldsymbol{x}, \boldsymbol{u}) = \hat{\boldsymbol{f}}_A(\boldsymbol{x}, \boldsymbol{u})$. In general, a nonlinear and nonaffine-in-control dynamics system of the form

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \tag{6.15}$$

would be more appropriate for systems of unknown structures.

6.4.1 General Control for Partial DNN Dynamics

In the case of (6.14), a similar controller to (6.9) for dynamics (6.14) can be derived to drive state $\boldsymbol{x}$ to reference state $\boldsymbol{r}(t)$. Using a DNN to approximate the unknown dynamics as $\hat{\boldsymbol{f}}(\boldsymbol{x}, \boldsymbol{u})$, we can write the controller as

$$\boldsymbol{u}_k = \boldsymbol{B}_0(\boldsymbol{x})^+ \Big(\dot{\boldsymbol{r}}(t) - \boldsymbol{f}_0(\boldsymbol{x}) - \boldsymbol{K}\tilde{\boldsymbol{x}} - \hat{\boldsymbol{f}}_U(\boldsymbol{x}, \boldsymbol{u}_{k-1})\Big). \tag{6.16}$$

For fully-actuated or over-actuated systems, we can always find $\boldsymbol{B}_0(\boldsymbol{x})\boldsymbol{B}_0(\boldsymbol{x})^+ = \boldsymbol{I}$. Substitute (6.16) into (6.14), we can easily show that the closed-loop dynamics are written as

$$\dot{\tilde{\boldsymbol{x}}} = -\boldsymbol{K}\tilde{\boldsymbol{x}} + \Big(\hat{\boldsymbol{f}}_U(\boldsymbol{x}, \boldsymbol{u}_k) - \hat{\boldsymbol{f}}_U(\boldsymbol{x}, \boldsymbol{u}_{k-1})\Big) + \boldsymbol{\epsilon}_U, \tag{6.17}$$

where $\boldsymbol{\epsilon}_U = \boldsymbol{f}_U(\boldsymbol{x}, \boldsymbol{u}) - \hat{\boldsymbol{f}}_U(\boldsymbol{x}, \boldsymbol{u})$ is the approximation error of $\boldsymbol{f}_U(\cdot)$. We can state the stability of the closed-loop system in the following theorem.

Theorem 6.4. *Applying controller (6.16) to dynamics (6.14), the state tracking error $\tilde{\boldsymbol{x}}$ converges exponentially to error ball*

$$\lim_{t\to\infty} \|\tilde{\boldsymbol{x}}(t)\| = \frac{\sup_{\boldsymbol{x},\boldsymbol{u}} \|\boldsymbol{\epsilon}_U(\boldsymbol{x}, \boldsymbol{u})\|}{\lambda_k - \kappa \left\|\hat{\boldsymbol{f}}_U\right\|_{\mathrm{Lip}}}, \tag{6.18}$$

with rate $\lambda_k - \kappa \left\|\hat{\boldsymbol{f}}_U\right\|_{\mathrm{Lip}}$, and $\lambda_k = \lambda_{\min}(\boldsymbol{K})$.

Proof. Similar to Theorem 6.3, by making similar assumptions that

$$\|\boldsymbol{u}_k - \boldsymbol{u}_{k-1}\| \le \kappa \|\tilde{\boldsymbol{x}}\|,$$

and approximation error is ultimately bounded by $\sup_{\boldsymbol{x},\boldsymbol{u}} \|\boldsymbol{\epsilon}_U(\boldsymbol{x}, \boldsymbol{u})\|$. Using closed-loop equation (6.17), we can get the exponential convergence behavior of $\tilde{\boldsymbol{x}}$:

$$\|\boldsymbol{x}(t)\| \le \|\boldsymbol{x}(t_0)\| \exp\left(-\left(\lambda_k - \kappa \left\|\hat{\boldsymbol{f}}_U\right\|_{\mathrm{Lip}}\right)(t - t_0)\right) + \frac{\sup_{\boldsymbol{x},\boldsymbol{u}} \|\boldsymbol{\epsilon}_U(\boldsymbol{x}, \boldsymbol{u})\|}{\lambda_k - \kappa \left\|\hat{\boldsymbol{f}}_U\right\|_{\mathrm{Lip}}}. \tag{6.19}$$

■

From Lemma 6.2, we know that the fixed-point iteration used in (6.16) relies on the contraction mapping property, that the Lipschitz constant of the function composition must satisfy

$$\left\|\boldsymbol{B}_0(\boldsymbol{x})^+\right\|_{\text{Lip}} \cdot \left\|\hat{\boldsymbol{f}}_U\right\|_{\text{Lip}} < 1. \tag{6.20}$$

Since $\boldsymbol{B}_0(\boldsymbol{x})$ is part of the pre-determined dynamics, (6.20) restricts the Lipschitz of the unknown dynamics function to be

$$\left\|\hat{\boldsymbol{f}}_U\right\|_{\text{Lip}} < \frac{1}{\|\boldsymbol{B}_0(\boldsymbol{x})^+\|_{\text{Lip}}}. \tag{6.21}$$

This is acceptable when the disturbance function happens to possess the same Lipschitz property $\left\|\hat{\boldsymbol{f}}_U\right\|_{\text{Lip}} = \|\boldsymbol{f}_U\|_{\text{Lip}}$. If the nonaffine-in-control term $\boldsymbol{f}_U(\boldsymbol{x}, \boldsymbol{u})$ exceeds the requirement $\left\|\hat{\boldsymbol{f}}_U\right\|_{\text{Lip}} < \|\boldsymbol{f}_U\|_{\text{Lip}}$, then it is likely that $\sup_{\boldsymbol{x},\boldsymbol{u}} \|\boldsymbol{\epsilon}_U(\boldsymbol{x}, \boldsymbol{u})\|$ becomes significant and poses challenge for trajectory tracking quality.

6.4.2 Learning to Control Full DNN Dynamics

For general dynamic systems $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ from (6.15), we can learn a $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ through data of $\{\boldsymbol{x}, \boldsymbol{u}\}$, and represent it with a DNN approximator $\hat{\boldsymbol{f}}(\boldsymbol{x}, \boldsymbol{u})$ following the discussion from Section 6.1. Without the assumptions of linear structures in dynamics, the nonaffine-in-control problem can be challenging to solve. In [129], a general solution was proposed by treating $\dot{\boldsymbol{u}}$ as control input. For a scalar single-input single-output (SISO) system with $x,\ u \in \mathbb{R}$, the controller that drives the system to behave as $\dot{x} = -ax$ is of the form

$$\epsilon \dot{u} = -\text{sign}\left(\frac{\partial f}{\partial u}\right)\big(f(x, u) + ax\big), \qquad \text{with} \qquad \epsilon \ll 1. \tag{6.22}$$

However, the resultant controller requires that (a) $\dot{\boldsymbol{u}}$ changes continuously; (b) $\epsilon \ll 1$ is a timescale factor that prompts $\dot{\boldsymbol{u}}$ to evolve arbitrarily fast; and (c) $\boldsymbol{u}$ is closer to the true solution to equation $f(x, u) + ax = 0$. For a discrete controller implemented on a digital computer, a large change rate would require the update frequency to increase substantially. If the dynamics approximation $\hat{\boldsymbol{f}}(\boldsymbol{x}, \boldsymbol{u})$ is done via a DNN, then the evaluation of $\partial \boldsymbol{f}/\partial \boldsymbol{u}$ at a fast rate would require significant computation resources.

For general control design, we elect to use a DNN $\hat{\boldsymbol{f}}(\cdot, \hat{\boldsymbol{\theta}}_f)$ to approximate dynamics $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ and another DNN $\hat{\boldsymbol{c}}(\cdot, \hat{\boldsymbol{\theta}}_c)$ to represent the controller:

$$\boldsymbol{u}(t) = \hat{\boldsymbol{c}}\big(\boldsymbol{x}(t), \boldsymbol{r}(t), \dot{\boldsymbol{r}}(t), \hat{\boldsymbol{\theta}}_c\big). \tag{6.23}$$

$\hat{\boldsymbol{\theta}}_f$ and $\hat{\boldsymbol{\theta}}_c$ are parameters of the respective networks. The goal for $\hat{\boldsymbol{c}}(\cdot)$ is such that when applied to the original dynamics $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$, the closed-loop dynamics

$$\dot{\boldsymbol{x}} = \boldsymbol{f}\big(\boldsymbol{x}, \hat{\boldsymbol{c}}(\boldsymbol{x}, \boldsymbol{r}, \dot{\boldsymbol{r}})\big) \tag{6.24}$$

will guarantee convergence $\boldsymbol{x}(t) \to \boldsymbol{r}(t)$.

For learning the dynamics network $\hat{\boldsymbol{f}}(\cdot, \hat{\boldsymbol{\theta}}_f)$, we record state and control input data, and process them into sets of feature and label

$$\mathcal{D}_f = \left\{ \left[\boldsymbol{x}^{(i)}; \boldsymbol{u}^{(i)} \right], \dot{\boldsymbol{x}}^{(i)} \right\} \quad \text{with} \quad i = 1, \cdots, N. \tag{6.25}$$

We then define the following supervised learning problem with a loss function $\mathcal{L}_f$:

$$\min_{\hat{\boldsymbol{\theta}}_f} \quad \mathcal{L}_f = \frac{1}{N} \sum_{i=1}^{N} \left\| \dot{\boldsymbol{x}}^{(i)} - \hat{\boldsymbol{f}}(\boldsymbol{x}^{(i)}, \boldsymbol{u}^{(i)}, \hat{\boldsymbol{\theta}}_f) \right\|. \tag{6.26}$$

For learning the controller network $\hat{\boldsymbol{c}}(\cdot, \hat{\boldsymbol{\theta}}_c)$, we need it follow a prescribed exponential convergence using Lyapunov analysis. For tracking error $\tilde{\boldsymbol{x}}(t) = \boldsymbol{x}(t) - \boldsymbol{r}(t)$, we select a Lyapunov function $\mathcal{V}(\tilde{\boldsymbol{x}})$, and derive its estimated derivative as

$$\hat{\dot{\mathcal{V}}}(\boldsymbol{x}, \boldsymbol{r}, \dot{\boldsymbol{r}}, \hat{\boldsymbol{\theta}}_f, \hat{\boldsymbol{\theta}}_c) = \frac{\partial \mathcal{V}}{\partial \tilde{\boldsymbol{x}}} \left[\hat{\boldsymbol{f}}\Big(\boldsymbol{x}, \hat{\boldsymbol{c}}(\boldsymbol{x}, \boldsymbol{r}, \dot{\boldsymbol{r}}, \hat{\boldsymbol{\theta}}_c), \hat{\boldsymbol{\theta}}_f\Big) - \dot{\boldsymbol{r}}(t) \right]. \tag{6.27}$$

For exponential convergence with rate $\alpha > 0$, we want the inequality $\dot{\mathcal{V}} \le -\alpha \mathcal{V}$ to hold. To fit the network parameter $\hat{\boldsymbol{\theta}}_c$, we construct another dataset from the record as

$$\mathcal{D}_c = \left\{ \left[\boldsymbol{x}^{(i)}; \boldsymbol{r}^{(i)}; \dot{\boldsymbol{r}}^{(i)} \right] \right\} \quad \text{with} \quad i = 1, \cdots, N. \tag{6.28}$$

In this case, we will fix the dynamics network parameter $\hat{\boldsymbol{\theta}}_f$ and define a second optimization problem:

$$\min_{\hat{\boldsymbol{\theta}}_c} \quad \mathcal{L}_c = \frac{1}{N} \sum_{i=1}^{N} \phi_{\text{lyap}} \left(\hat{\dot{\mathcal{V}}}\big(\boldsymbol{x}^{(i)}, \boldsymbol{r}^{(i)}, \dot{\boldsymbol{r}}^{(i)}, \hat{\boldsymbol{\theta}}_f, \hat{\boldsymbol{\theta}}_c\big) + \alpha \mathcal{V}\big(\boldsymbol{x}^{(i)} - \boldsymbol{r}^{(i)}\big) \right), \tag{6.29}$$

where $\phi_{\text{lyap}}(\cdot)$ is a scalar function that asymmetrically penalizes $\hat{\dot{\mathcal{V}}}^{(i)} + \alpha \mathcal{V}^{(i)} > 0$. Note that (6.29) does not use recorded control inputs $\boldsymbol{u}^{(i)}$ in training, this is similar to a off-policy method in reinforcement learning settings.

In practice, we would first train the dynamics network by optimizing the cost from (6.26) and obtain the parameters $\hat{\boldsymbol{\theta}}_f^*$, then use the learned $\hat{\boldsymbol{f}}(\cdot, \hat{\boldsymbol{\theta}}_f^*)$ to get the controller parameter $\hat{\boldsymbol{\theta}}_c^*$ through (6.29). The resultant controller $\hat{\boldsymbol{c}}(\cdot, \hat{\boldsymbol{\theta}}_c^*)$ can be applied to the actual system thereafter. The exact composition of training schedules will be determined by different use cases. An episodic version of such process is listed in Algorithm 2, which alternates the training of $\hat{\boldsymbol{\theta}}_f$ and $\hat{\boldsymbol{\theta}}_c$ after each trial episodes. The data can be randomly sampled and do not have to be sequential because the networks contain no recurrent formulation.

Algorithm 2 Episodic Learning for Dynamics and Control DNNs

```
Input: state $\boldsymbol{x}$, control input $\boldsymbol{u}$, desired state trajectory $\boldsymbol{r}(t)$ and $\dot{\boldsymbol{r}}(t)$, Lyapunov
    function $\mathcal{V}(\cdot)$, replay buffer size $N_r$, number of episodes $N_e$, episode length $N_t$,
    training iterations $N_l$, training batch size $N_b$
Output: network parameters $\hat{\boldsymbol{\theta}}_f$ and $\hat{\boldsymbol{\theta}}_c$
 1: Randomly initialize $\hat{\boldsymbol{\theta}}_f$ and $\hat{\boldsymbol{\theta}}_c$
 2: Initialize replay buffer $\mathcal{R}$ with size $N_r$
 3: for episode $i = 1, \cdots, N_e$ do
 4:     Randomize initial state $\boldsymbol{x}(t_1)$
 5:     for time $t = 1, \cdots, N_t$ do
 6:         Obtain current $\boldsymbol{x}^{(t)}$, $\dot{\boldsymbol{x}}^{(t)}$, $\boldsymbol{r}^{(t)}$ and $\dot{\boldsymbol{r}}^{(t)}$
 7:         Execute control $\boldsymbol{u}^{(t)} = \hat{\boldsymbol{c}}\left(\boldsymbol{x}^{(t)}, \boldsymbol{r}^{(t)}, \dot{\boldsymbol{r}}^{(t)}, \hat{\boldsymbol{\theta}}_c\right)$
 8:         Record $\left[\boldsymbol{x}^{(t)}; \dot{\boldsymbol{x}}^{(t)}; \boldsymbol{r}^{(t)}, \dot{\boldsymbol{r}}^{(t)}, \boldsymbol{u}^{(t)}\right]$ in $\mathcal{R}$
 9:     end for
10:     for iteration $j = 1, \cdots, N_l$ do
11:         Sample batch $\mathcal{D}_f^{(j)}$ from $\mathcal{R}$ as in (6.25)
12:         Run gradient update on $\hat{\boldsymbol{\theta}}_f$ with loss $\mathcal{L}_f\left(\mathcal{D}_f^{(j)}, \hat{\boldsymbol{\theta}}_f\right)$ as in (6.26)
13:     end for
14:     for iteration $j = 1, \cdots, N_l$ do
15:         Sample batch $\mathcal{D}_c^{(j)}$ from $\mathcal{R}$ as in (6.28)
16:         Run gradient update on $\hat{\boldsymbol{\theta}}_c$ with loss $\mathcal{L}_c\left(\mathcal{D}_c^{(j)}, \hat{\boldsymbol{\theta}}_c\right)$ as in (6.29)
17:     end for
18: end for
19: return learned parameters $\hat{\boldsymbol{\theta}}_f$ and $\hat{\boldsymbol{\theta}}_c$
```

6.4.3 Learning to Control from DNN Observer

For realistic settings, it is possible that the actual states $\boldsymbol{x}$, and the structure of dynamics $\boldsymbol{f}(\cdot)$ are unknown to us. Instead, we have access to a observed output vector $\boldsymbol{y}$. The transformation from $\boldsymbol{x}$ to $\boldsymbol{y}$ can often be represented by an observation model

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}). \tag{6.30}$$

One approach for learning a controller under this setting would be treating $\boldsymbol{x} \equiv \boldsymbol{y}$, and the dynamics would simply becomes $\boldsymbol{f}(\boldsymbol{y}, \boldsymbol{u})$. When $\boldsymbol{y}$ is high dimensional (e.g images), it is often advisable to encode it to a lower dimensional representation instead

$$\boldsymbol{x} = \hat{o}_{\text{ENC}}(\boldsymbol{y}), \tag{6.31}$$

given that $\boldsymbol{y}$ contains all information necessary to recover $\boldsymbol{x}$ for proper dynamics propagation. However, a general formulation is to use an observer to obtain a state estimate $\hat{\boldsymbol{x}}$. As an example, a PD observer for a general nonlinear system from [130] has the form

$$\begin{aligned}\dot{\hat{\boldsymbol{x}}} &= \boldsymbol{f}(\hat{\boldsymbol{x}}, \boldsymbol{u}) - \boldsymbol{K}_P(\hat{\boldsymbol{y}} - \boldsymbol{y}) - \boldsymbol{K}_D(\dot{\hat{\boldsymbol{y}}} - \dot{\boldsymbol{y}}) \\ \hat{\boldsymbol{y}} &= \boldsymbol{h}(\hat{\boldsymbol{x}}, \boldsymbol{u}), \quad \dot{\hat{\boldsymbol{y}}} = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} \boldsymbol{f}(\hat{\boldsymbol{x}}, \boldsymbol{u})\end{aligned} \tag{6.32}$$

Similar to state trajectory tracking problems, our objective is to track a output trajectory by enforcing $\boldsymbol{y} \to \boldsymbol{y}_d$ with the tracking error defined as $\tilde{\boldsymbol{y}} = \boldsymbol{y} - \boldsymbol{y}_d$. Following the same philosophy of using neural network approximation, we will train an observer neural network $\hat{\boldsymbol{o}}(\cdot, \hat{\boldsymbol{\theta}}_o)$, an observation model network $\hat{\boldsymbol{h}}(\cdot, \hat{\boldsymbol{\theta}}_h)$, and a slightly different controller network $\hat{\boldsymbol{c}}'(\cdot, \hat{\boldsymbol{\theta}}_c)$ to handle output tracking and state estimation:

$$\dot{\hat{\boldsymbol{x}}} = \hat{\boldsymbol{f}}\,(\hat{\boldsymbol{x}}, \boldsymbol{u}) + \hat{\boldsymbol{o}}\left(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \boldsymbol{y}, \hat{\boldsymbol{\theta}}_o\right) \tag{6.33}$$

$$\hat{\boldsymbol{y}} = \hat{\boldsymbol{h}}\left(\hat{\boldsymbol{x}}, \boldsymbol{u}, \hat{\boldsymbol{\theta}}_h\right) \tag{6.34}$$

$$\boldsymbol{u} = \hat{\boldsymbol{c}}'(\boldsymbol{y}, \boldsymbol{y}_d, \dot{\boldsymbol{y}}_d, \hat{\boldsymbol{x}}, \hat{\boldsymbol{\theta}}_c). \tag{6.35}$$

Given the recurrent nature of an observer (6.33), we have to utilize sequential data from continuous trajectory and back-propagation through-time (BPTT) during training process. We convert the continuous equations from (6.33) to discrete updates by introducing prior and posterior state estimates $\hat{\boldsymbol{x}}_-$, $\hat{\boldsymbol{x}}_+$. Along a trajectory with time length N_t, we can construct a sequential dataset

$$\mathcal{D}_f' = \left\{\left[\boldsymbol{y}^{(t)}; \boldsymbol{u}^{(t)}\right]\right\} \quad \text{with} \quad t = 1, \cdots, N_t, \tag{6.36}$$

and optimize the following cost function in a recurrent fashion to learn the dynamics and the observer models:

$$\begin{aligned}\min_{\hat{\boldsymbol{\theta}}_f, \hat{\boldsymbol{\theta}}_o, \hat{\boldsymbol{\theta}}_h} \quad & \mathcal{L}_f' = \frac{1}{N_t} \sum_{t=1}^{N_t} \left\| \boldsymbol{y}^{(t)} - \hat{\boldsymbol{y}}^{(t)}\left(\cdots, \hat{\boldsymbol{\theta}}_f, \hat{\boldsymbol{\theta}}_o, \hat{\boldsymbol{\theta}}_h\right) \right\|, \\ \text{where} \quad & \hat{\boldsymbol{y}}^{(t)} = \hat{\boldsymbol{h}}\left(\hat{\boldsymbol{x}}_-^{(t)}, \boldsymbol{u}^{(t)}, \hat{\boldsymbol{\theta}}_h\right), \\ & \hat{\boldsymbol{x}}_+^{(t)} = \hat{\boldsymbol{o}}\left(\hat{\boldsymbol{x}}_-^{(t)}, \hat{\boldsymbol{y}}^{(t)}, \boldsymbol{y}^{(t)}, \hat{\boldsymbol{\theta}}_o\right) \Delta t, \\ & \hat{\boldsymbol{x}}_-^{(t+1)} = \hat{\boldsymbol{x}}_+^{(t)} + \hat{\boldsymbol{f}}\left(\hat{\boldsymbol{x}}_+^{(t)}, \boldsymbol{u}^{(t)}\right) \Delta t.\end{aligned} \tag{6.37}$$

For learning the controller network, we admit a Lyapunov function $\mathcal{V}(\tilde{\boldsymbol{y}})$. A discrete version of exponential convergence can be represented as

$$\mathcal{V}\left(\tilde{\boldsymbol{y}}^{(t+1)}\right) - (1 - \alpha \Delta t)\, \mathcal{V}\left(\tilde{\boldsymbol{y}}^{(t)}\right) \leq 0 \tag{6.38}$$

At each timestep t, the current $\tilde{\boldsymbol{y}}^{(t)} = \boldsymbol{y}^{(t)} - \boldsymbol{y}_d^{(t)}$ is given but the future $\tilde{\boldsymbol{y}}^{(t+1)}$ has to be estimated. Again, a sequential dataset is assembled for recurrent learning process

$$\mathcal{D}'_c = \left\{\left[\boldsymbol{y}^{(t)}; \boldsymbol{y}_d^{(t)}; \dot{\boldsymbol{y}}_d{}^{(t)}; \boldsymbol{u}^{(t)}\right]\right\} \quad \text{with} \quad t = 1, \cdots, N_t. \tag{6.39}$$

We define the learning problem for controller network on trajectory data as

$$\begin{aligned}
\min_{\hat{\boldsymbol{\theta}}_c} \quad & \mathcal{L}'_c = \frac{1}{N_t}\sum_{t=1}^{N_t} \phi_{\text{lyap}}\left(\hat{\mathcal{V}}^{(t+1)} + (\alpha\Delta t - 1)\,\mathcal{V}\left(\tilde{\boldsymbol{y}}^{(t)}\right)\right),\\
\text{where} \quad & \hat{\boldsymbol{x}}_+^{(t)} = \hat{\boldsymbol{o}}\left(\hat{\boldsymbol{x}}_-^{(t)}, \hat{\boldsymbol{y}}^{(t)}, \boldsymbol{y}^{(t)}, \hat{\boldsymbol{\theta}}_o\right)\Delta t,\\
& \hat{\boldsymbol{u}}^{(t)} = \hat{\boldsymbol{c}}'\left(\boldsymbol{y}^{(t)}, \boldsymbol{y}_d^{(t)}, \dot{\boldsymbol{y}}_d{}^{(t)}, \hat{\boldsymbol{x}}_+^{(t)}, \hat{\boldsymbol{\theta}}_c\right),\\
& \bar{\boldsymbol{x}}_-^{(t+1)} = \hat{\boldsymbol{x}}_+^{(t)} + \hat{\boldsymbol{f}}\left(\hat{\boldsymbol{x}}_+^{(t)}, \hat{\boldsymbol{u}}^{(t)}\right)\Delta t,\\
& \hat{\mathcal{V}}^{(t+1)} = \mathcal{V}\left(\hat{\boldsymbol{h}}\left(\bar{\boldsymbol{x}}_-^{(t+1)}, \hat{\boldsymbol{u}}^{(t)}, \hat{\boldsymbol{\theta}}_h\right) - \boldsymbol{y}_d^{(t+1)}\right),\\
& \hat{\boldsymbol{x}}_-^{(t+1)} = \hat{\boldsymbol{x}}_+^{(t)} + \hat{\boldsymbol{f}}\left(\hat{\boldsymbol{x}}_+^{(t)}, \boldsymbol{u}^{(t)}\right)\Delta t,\\
& \hat{\boldsymbol{y}}^{(t+1)} = \hat{\boldsymbol{h}}\left(\hat{\boldsymbol{x}}_-^{(t+1)}, \hat{\boldsymbol{u}}^{(t)}, \hat{\boldsymbol{\theta}}_h\right).
\end{aligned} \tag{6.40}$$

In essence, along the trajectory we use the same propagation equations in (6.37) in order to have proper recurrent estimates $\hat{\boldsymbol{x}}_-^{(t+1)}$ and $\hat{\boldsymbol{y}}^{(t+1)}$ with recorded control inputs $\boldsymbol{u}^{(t)}$. On the other hand, we also calculate the actions from controller network as $\hat{\boldsymbol{u}}^{(t)}$, and propagate forward to get the estimated future state $\bar{\boldsymbol{x}}_-^{(t+1)}$ and $\hat{\mathcal{V}}^{(t+1)}$.

Algorithm 3 summarizes the same episodic learning process as Algorithm 2, except with recurrent structures that require continuous trajectory data. The dimension of $\boldsymbol{x}$ can be tuned in this case, since we have no knowledge of it beforehand. In principle, the recurrent structure of the observer network $\hat{\boldsymbol{o}}(\cdot, \hat{\boldsymbol{\theta}}_o)$ can have states $\boldsymbol{x}$ resemble not only dynamics evolution but also adaptation parameters.

6.4.4 Connection to Reinforcement Learning

Off-policy reinforcement learning methods on continuous action space has some similarities to our proposed framework. In particular, actor-critic methods such as Deep Deterministic Policy Gradient (DDPG) from [131] uses neural networks to approximate Q-functions $Q(\boldsymbol{x}, \boldsymbol{u}, \hat{\boldsymbol{\theta}}_Q)$ and optimal policies $\hat{\boldsymbol{c}}(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_c)$. Using our

Algorithm 3 Recurrent Learning for DNN dynamics with Observer

```
Input: output $\boldsymbol{y}$, control input $\boldsymbol{u}$, desired output trajectory $\boldsymbol{y}_d(t)$ and $\dot{\boldsymbol{y}}_d(t)$, Lyapunov
    function $\mathcal{V}(\cdot)$, replay buffer size $N_r$, number of episodes $N_e$, episode length $N_t$,
    training trajectory length $N_l$, training batch size $N_b$
Output: network parameters $\hat{\boldsymbol{\theta}}_f$, $\hat{\boldsymbol{\theta}}_c$, $\hat{\boldsymbol{\theta}}_o$ and $\hat{\boldsymbol{\theta}}_h$
 1: Randomly initialize $\hat{\boldsymbol{\theta}}_f$, $\hat{\boldsymbol{\theta}}_c$, $\hat{\boldsymbol{\theta}}_o$ and $\hat{\boldsymbol{\theta}}_h$
 2: Initialize replay buffer $\mathcal{R}$ with size $N_r$
 3: for episode $i = 1, \ldots, N_e$ do
 4:     Select random initial output $\boldsymbol{y}^{(1)}$
 5:     Randomly initialize state estimate $\hat{\boldsymbol{x}}_-^{(1)}$ and output estimate $\hat{\boldsymbol{y}}^{(1)}$
 6:     for time $t = 1, \ldots, N_t$ do
 7:         Obtain current $\boldsymbol{y}^{(t)}$,$\boldsymbol{y}_d^{(t)}$, and $\dot{\boldsymbol{y}}_d{}^{(t)}$
 8:         Run observer $\hat{\boldsymbol{x}}_+^{(t)} = \hat{\boldsymbol{o}}\left(\hat{\boldsymbol{x}}_-^{(t)}, \hat{\boldsymbol{y}}^{(t)}, \boldsymbol{y}^{(t)}, \hat{\boldsymbol{\theta}}_o\right)\Delta t$
 9:         Execute control $\hat{\boldsymbol{u}}^{(t)} = \hat{\boldsymbol{c}}'\left(\boldsymbol{y}^{(t)}, \boldsymbol{y}_d^{(t)}, \dot{\boldsymbol{y}}_d{}^{(t)}, \hat{\boldsymbol{x}}_+^{(t)}, \hat{\boldsymbol{\theta}}_c\right)$
10:         Propagate state $\hat{\boldsymbol{x}}_-^{(t+1)} = \hat{\boldsymbol{x}}_+^{(t)} + \hat{\boldsymbol{f}}\left(\hat{\boldsymbol{x}}_+^{(t)}, \hat{\boldsymbol{u}}^{(t)}\right)\Delta t$
11:         Update output estimate $\hat{\boldsymbol{y}}^{(t+1)} = \hat{\boldsymbol{h}}\left(\hat{\boldsymbol{x}}_-^{(t+1)}, \hat{\boldsymbol{u}}^{(t)}, \hat{\boldsymbol{\theta}}_h\right)$
12:         Record $\left[\boldsymbol{y}^{(t)}; \boldsymbol{y}_d^{(t)}; \dot{\boldsymbol{y}}_d{}^{(t)}; \boldsymbol{u}^{(t)}\right]$ in $\mathcal{R}$
13:     end for
14:     Sample batch of trajectories $\mathcal{D}_f'^{(i)}$ of $N_l \times N_b$ from $\mathcal{R}$ as in (6.36)
15:     Run gradient update on $\hat{\boldsymbol{\theta}}_f, \hat{\boldsymbol{\theta}}_o, \hat{\boldsymbol{\theta}}_h$ with loss $\mathcal{L}_f'\left(\mathcal{D}_f'^{(i)}, \hat{\boldsymbol{\theta}}_f, \hat{\boldsymbol{\theta}}_o, \hat{\boldsymbol{\theta}}_h\right)$ as in (6.37)
16:     Sample batch of trajectories $\mathcal{D}_c'^{(i)}$ of $N_l \times N_b$ from $\mathcal{R}$ as in (6.39)
17:     Run gradient update on $\hat{\boldsymbol{\theta}}_c$with loss $\mathcal{L}_c'\left(\mathcal{D}_c'^{(i)}, \hat{\boldsymbol{\theta}}_c\right)$ as in (6.40)
18: end for
19: return learned parameters $\hat{\boldsymbol{\theta}}_f$, $\hat{\boldsymbol{\theta}}_c$, $\hat{\boldsymbol{\theta}}_o$, and $\hat{\boldsymbol{\theta}}_h$
```

notation, DDPG alternates between the following optimizations:

$$\min_{\hat{\boldsymbol{\theta}}_Q} \ \mathcal{L}_Q = \frac{1}{N}\sum_i \left(r^{(i)} + \gamma Q\left(\boldsymbol{x}^{(i+1)}, \hat{\boldsymbol{c}}(\boldsymbol{x}^{(i+1)}, \hat{\boldsymbol{\theta}}_c'), \hat{\boldsymbol{\theta}}_Q'\right) - Q\left(\boldsymbol{x}^{(i)}, \boldsymbol{u}^{(i)}, \hat{\boldsymbol{\theta}}_Q\right)\right)^2, \tag{6.41}$$

$$\max_{\hat{\boldsymbol{\theta}}_c} \ \mathcal{J}_u = \frac{1}{N}\sum_i Q\left(\boldsymbol{x}^{(i)}, \hat{\boldsymbol{c}}(\boldsymbol{x}^{(i)}, \hat{\boldsymbol{\theta}}_c), \hat{\boldsymbol{\theta}}_Q\right). \tag{6.42}$$

DDPG and other actor-critic algorithms first use (6.41) to learn optimal action value functions from past data, then optimize policies (i.e controllers). In contrast, our method learns the dynamics of state transition functions, and relies on a pre-selected Lyapunov function $\mathcal{V}(\cdot)$ to replace optimality.

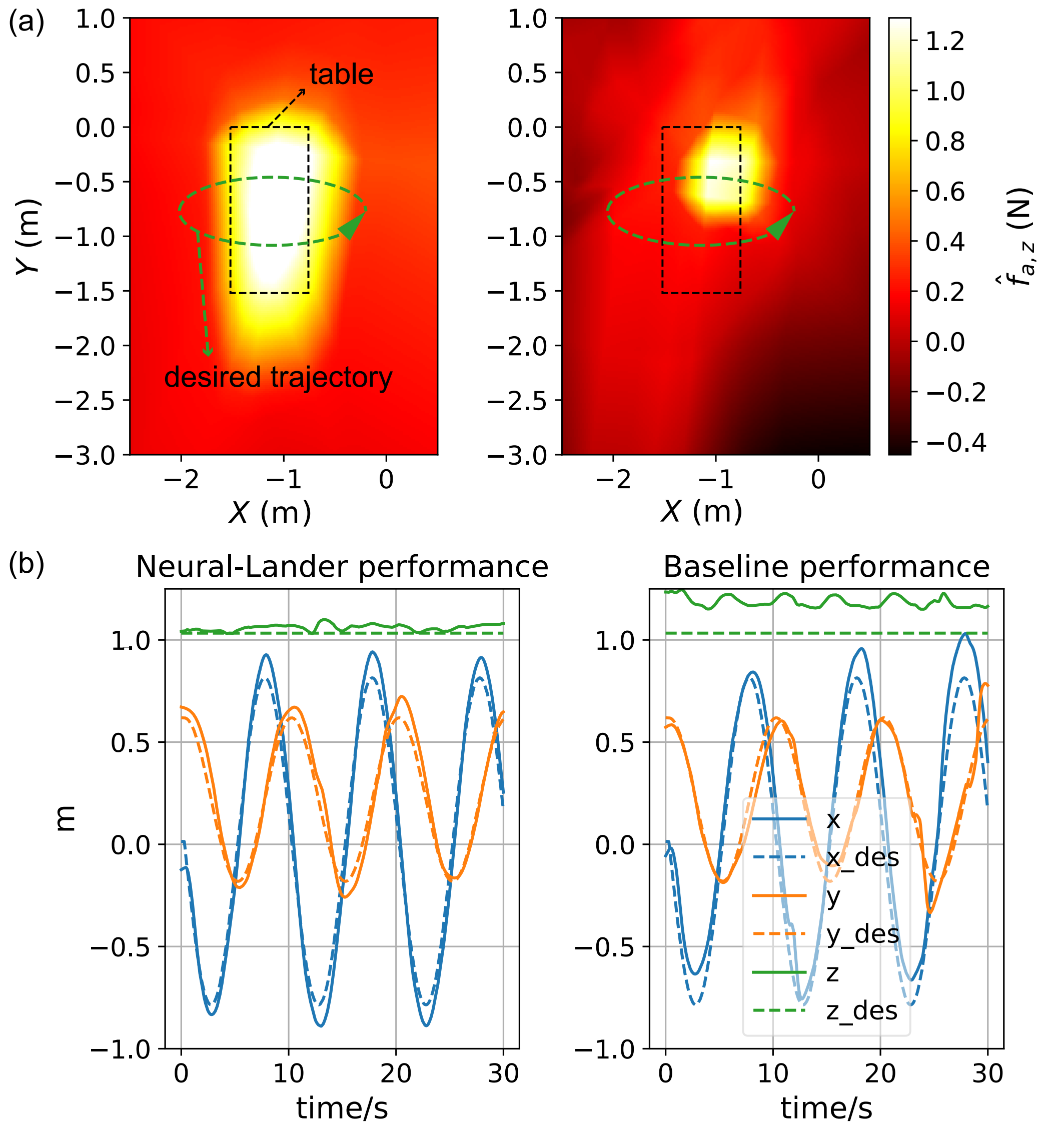
For example, the Q-function for linear quadratic regulator (LQR) problems is a particular Control Lyapunov Function (CLF). Loosely speaking, we are fixing the structure of our Q-function by selecting $\mathcal{V}(\cdot)$ using knowledge of control theory, and adapt a CLF through learning the dynamics $\hat{\boldsymbol{f}}(\boldsymbol{x}, \boldsymbol{u})$. Both our method and DDPG employ controller in the forms of neural networks $\hat{\boldsymbol{c}}(\cdot, \hat{\boldsymbol{\theta}}_c)$. Without the asymmetric penalty function $\phi_{\text{lyap}}(\cdot)$, (6.29) and (6.40) is equivalent to (6.42) if we treat

$$Q\left(\boldsymbol{x}^{(i)}, \hat{\boldsymbol{c}}(\boldsymbol{x}^{(i)}, \hat{\boldsymbol{\theta}}_c), \hat{\boldsymbol{\theta}}_Q\right) \equiv \mathcal{V}\left(\boldsymbol{x}^{(i)} + \Delta t \hat{\boldsymbol{f}}(\boldsymbol{x}^{(i)}, \hat{\boldsymbol{c}}(\boldsymbol{x}^{(i)}, \hat{\boldsymbol{\theta}}_c, \hat{\boldsymbol{\theta}}_f))\right). \tag{6.43}$$

Without loss of generality, we set $\boldsymbol{r}(t) \equiv \boldsymbol{0}$. In summary, a Q-function learns a general value function from reward data; and a Lyapunov function is tailored toward the cost of driving $\boldsymbol{x}(t) \to \boldsymbol{r}(t)$ or $\boldsymbol{y}(t) \to \boldsymbol{y}_d(t)$.

6.5 Chapter Summary

In this chapter, we presented a deep learning-based nonlinear controller with guaranteed stability for improved flight performance, and elaborated to general DNN controller and observer formulation for any nonlinear dynamics. With pre-collected flight data, the spectrally normalized ReLU DNN was able to estimate unknown residual forces with a specified Lipschitz constant. Compared to previous physics-based methods, our DNN learned from coupled aerodynamics and vehicle dynamics to provide more accurate estimates. We also provided rigorous theoretical analysis of our method and guarantee the stability of the controller, which implied generalization to unseen domains. Compared to baseline control methods, our DNN-based controller was able to significantly improve tracking accuracy in previously challenging tasks such as take-off, landing, and flying near large objects. When residual disturbance within dynamics became substantial, we proposed a general deep learning framework that incorporated controller and observer networks. The method used DNNs for all function approximators. It could potentially regulate any nonaffine-in-control systems on feasible reference trajectories. This novel approach shared resemblance to off-policy RL methods, but was tailored particularly toward control settings.



Mean X error	**0.079m**	0.126m
Mean Y error	**0.052m**	0.061m
Mean Z error	**0.027m**	0.153m
Z variance	**0.014m**	0.026m

Figure 6.7: Generalization and control performance during near-object tasks. (a) Heatmaps of learned $\hat{f}_{A,z}$ vs. x and y, with other inputs fixed. (Left) ReLU network with spectral normalization. (Right) ReLU network without spectral normalization. (b) Tracking performance and statistics.

Chapter 7

ACTUATION DELAY COMPENSATION

When actuator measurements are available, it is straightforward to include the actuator dynamics in the full system control design and adjust for the additional transport delay. In cases where such measurements are inaccessible, actuator observers can be constructed. This is common in applications such as multirotor control when delays exist in motor speed, but output rotation may not be available [132], [133]. Without the assumption of a continuous control signal, we resolve to use hybrid stability analysis in place of Lyapunov-Krasvoskii approach. Similar methods have been employed to show that input-to-state (ISS) stable systems inherit robustness against effects of discrete sampling or reasonable actuation delays [134]–[136]. We present methods from [137] in this chapter to account for various types of delay during actuation.

7.1 Problem Formulation

7.1.1 Nonautonomous Dynamics of Trajectory Tracking

Consider the system described by nonlinear and nonautonomous dynamics of the form

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\eta}, t) \tag{7.1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the n-dimensional state, and $\boldsymbol{\eta} \in \mathbb{R}^m$ is the m-dimensional actuator input. Given a smooth, time-prescribed, feasible reference trajectory $\boldsymbol{r}(t)$, along with the corresponding reference control $\boldsymbol{\eta}^*(t)$, we define the state error as $\tilde{\boldsymbol{x}}(t) = \boldsymbol{x}(t) - \boldsymbol{r}(t)$ corresponding dynamics

$$\dot{\tilde{\boldsymbol{x}}} = \boldsymbol{g}(\tilde{\boldsymbol{x}}, \boldsymbol{\eta}, t). \tag{7.2}$$

$\boldsymbol{g}(\tilde{\boldsymbol{x}}, \boldsymbol{\eta}, t) = \boldsymbol{f}(\tilde{\boldsymbol{x}} + \boldsymbol{r}(t), \boldsymbol{\eta}, t) - \dot{\boldsymbol{r}}(t)$ is transformed from (7.1). Without loss of generality, we will focus our analysis on system (7.2) in this paper. We also assume that $\boldsymbol{r}(t)$ is feasible for (7.1) with $\boldsymbol{\eta}^*(t)$ that guarantees $\boldsymbol{f}(\boldsymbol{r}(t), \boldsymbol{\eta}^*(t), t) = \dot{\boldsymbol{r}}(t)$. Therefore, along $\boldsymbol{r}(t)$ we have

$$0 = \boldsymbol{g}(0, \boldsymbol{\eta}^*(t), t). \tag{7.3}$$

Additionally, we make the following assumptions:

Assumption 7.1. *The function $\boldsymbol{f}(\cdot)$ is Lipschitz continuous on compact sets with constant L_f. The trajectory $\boldsymbol{r}(t)$ is C^2 smooth with bounded derivatives. Thus it follows that $\boldsymbol{g}(\cdot)$ is Lipschitz continuous on compact sets with constant L_g.*

Assumption 7.2. *The full state vector $\boldsymbol{x}$ is observable, the analytical form of $\boldsymbol{r}(t)$ and its derivatives are known, but $\boldsymbol{\eta}$ cannot be measured directly.*

Assumptions 7.1 and 7.2 are not overly restrictive. A wide class of dynamic systems possess these properties. The unavailability of measuring $\boldsymbol{\eta}$ is intentional, and that variation of our method can compensate for delay in $\boldsymbol{\eta}$ without its feedback.

7.1.2 Exponentially Stabilizing Control for Undelayed System

Suppose a feedback controller of the form $\boldsymbol{\eta} = \bar{\boldsymbol{\eta}}\left(\tilde{\boldsymbol{x}}(t), t\right)$ has been designed, such that when applied to (7.2), the closed-loop system $\dot{\tilde{\boldsymbol{x}}} = \boldsymbol{g}\big(\tilde{\boldsymbol{x}}, \bar{\boldsymbol{\eta}}(\tilde{\boldsymbol{x}}, t), t\big)$ is exponentially stable. By the Converse Lyapunov Theorem [95], there exists a smooth Lyapunov function $\mathcal{V}(\tilde{\boldsymbol{x}}, t)$ such that

$$c_1 \left\|\tilde{\boldsymbol{x}}\right\|^2 \le \mathcal{V}(\tilde{\boldsymbol{x}}, t) \le c_2 \left\|\tilde{\boldsymbol{x}}\right\|^2 \tag{7.4a}$$

$$\frac{\partial \mathcal{V}}{\partial t} + \frac{\partial \mathcal{V}}{\partial \tilde{\boldsymbol{x}}}\boldsymbol{g}(\tilde{\boldsymbol{x}}, \bar{\boldsymbol{\eta}}, t) \le -c_3 \left\|\tilde{\boldsymbol{x}}\right\|^2 \tag{7.4b}$$

$$\left\|\frac{\partial \mathcal{V}}{\partial \tilde{\boldsymbol{x}}}\right\| \le c_4 \left\|\tilde{\boldsymbol{x}}\right\|. \tag{7.4c}$$

Likewise, we assume the smoothness of the controller function:

Assumption 7.3. *The function $\bar{\boldsymbol{\eta}}(\cdot)$ is Lipschitz continuous on compact sets with constant $L_{\bar{\eta}}$.*

We can differentiate $\bar{\boldsymbol{\eta}}(\tilde{\boldsymbol{x}}, t)$ and use (7.2) to get

$$\dot{\bar{\boldsymbol{\eta}}}(\tilde{\boldsymbol{x}}, \boldsymbol{\eta}, t) = \frac{\partial \bar{\boldsymbol{\eta}}}{\partial \tilde{\boldsymbol{x}}}\boldsymbol{g}(\tilde{\boldsymbol{x}}, \boldsymbol{\eta}, t) + \frac{\partial \bar{\boldsymbol{\eta}}}{\partial t}. \tag{7.5}$$

Based on Assumption 7.3, it can be shown that $\dot{\bar{\boldsymbol{\eta}}}(\tilde{\boldsymbol{x}}, \boldsymbol{\eta}, t)$ is also Lipschitz continuous on compact sets, and we define its Lipschitz constant as $L_{\dot{\bar{\eta}}}$.

7.1.3 Delay in Systems with Sample-based Control

In practice, control input $\boldsymbol{\eta}(t)$ lags behind the actual command signal $u(t)$ generated by a sample-based control system. We choose to describe the combined delay as a sample-based first-order plus dead time (FOPDT) model defined between sample interval $t \in [t_i', t_{i+1}')$:

$$\dot{\boldsymbol{\eta}}(t) = -\boldsymbol{\Lambda}\boldsymbol{\eta}(t) + \boldsymbol{\Lambda}\boldsymbol{u}'(t), \quad \boldsymbol{u}'(t) = \boldsymbol{u}(t_i' - \Delta) \tag{7.6}$$

Table 7.1: Summary of proposed control methods for delay compensation

Baseline (7.4)	$\bar{\boldsymbol{\eta}}\left(\tilde{\boldsymbol{x}}(t), t\right)$
Actuator Delay (7.9)	$\bar{\boldsymbol{\eta}}'(\tilde{\boldsymbol{x}}, \boldsymbol{\eta}, t) = \bar{\boldsymbol{\eta}}(\tilde{\boldsymbol{x}}, t) + \boldsymbol{\Lambda}^{-1}\dot{\bar{\boldsymbol{\eta}}}(\tilde{\boldsymbol{x}}, \boldsymbol{\eta}, t)$
Observer-based (7.13)	$\bar{\boldsymbol{\eta}}''(\tilde{\boldsymbol{x}}, \hat{\boldsymbol{\eta}}, t) = (\boldsymbol{I} - \boldsymbol{\Lambda}^{-1}\Gamma)\hat{\boldsymbol{\eta}} + \boldsymbol{\Lambda}^{-1}\Gamma\bar{\boldsymbol{\eta}}(\tilde{\boldsymbol{x}}, t) + \boldsymbol{\Lambda}^{-1}\dot{\bar{\boldsymbol{\eta}}}(\tilde{\boldsymbol{x}}, \hat{\boldsymbol{\eta}}, t)$
Predictive (7.21)	$\bar{\boldsymbol{\eta}}''\left(\hat{\boldsymbol{z}}_{\text{RK}}(t_i + \Delta), t_i + \Delta\right)$
Truncated (7.30)	$\bar{\boldsymbol{\eta}}''_{\text{FO}}(t_i + \Delta) = \bar{\boldsymbol{\eta}}(t_i) + (\boldsymbol{\Lambda}^{-1} + \Delta)\frac{\bar{\boldsymbol{\eta}}(t_i) - \bar{\boldsymbol{\eta}}(t_{i-1})}{T_s}$

with $t_i' = t_i + \Delta$ being the time at which the actuator received the control signal computed from samples at t_i, and $\boldsymbol{\Lambda} \succ 0$ is a diagonal matrix whose entries are rates of convergence of $\boldsymbol{\eta}$. The signal generated by the controller $\boldsymbol{u}(t - \Delta)$ is delayed by Δ when it is received by the actuator as $\boldsymbol{u}'(t)$. Figure 7.1 illustrates such a process at sample time t_i: Δ_c is the computation delay, which is the time needed to compute a control signal; Δ_s is the combined system delay in other parallel processes (e.g network latency, downstream process, etc.). We express total transport delay as $\Delta = \Delta_c + \Delta_s$.

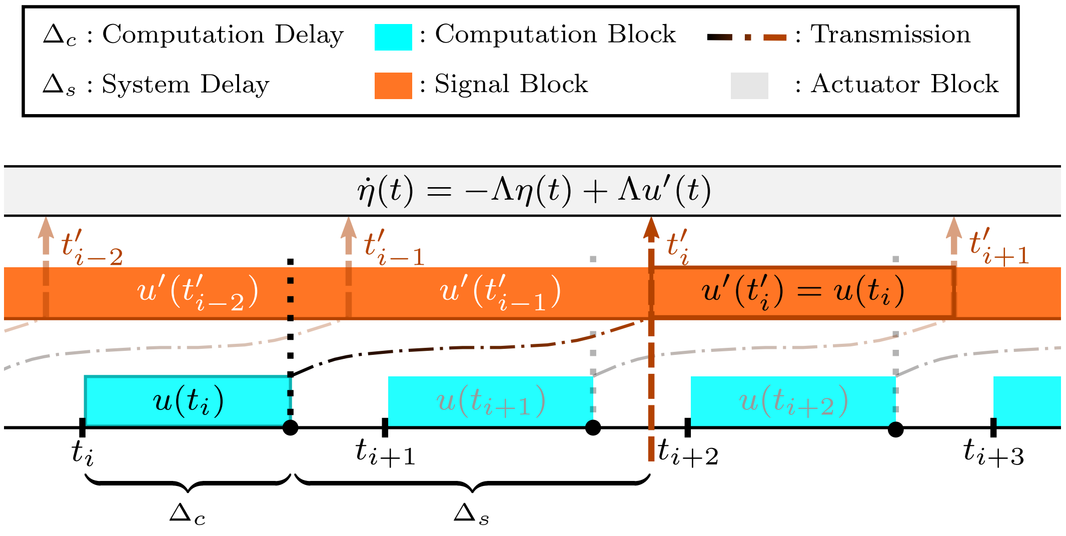


Figure 7.1: Timeline of periodic control with computation, system, and actuator delays. At every t_i, the controller begins computing a new command, $u(t_i)$, which takes Δ_c to calculate and an additional Δ_s to be received and applied by the actuators.

7.2 Delay Compensation Control

We first devise a control that compensates for first-order dynamic delay; then we introduce a general class of predictive controllers with a numerical integration scheme to account for large transport delays. The stability of the combined method will be analyzed under discrete sampling and integration. A summary of the proposed methods is shown in Table 7.1.

7.2.1 Derivative Compensation for First-order Delay

Consider the case with only first-order delay when $\bar{\boldsymbol{\eta}}(\tilde{\boldsymbol{x}}, t)$ is naively applied to $\boldsymbol{u}' = \bar{\boldsymbol{\eta}}(\tilde{\boldsymbol{x}}, t)$ in (7.6), the combined closed-loop system for actuation error $\tilde{\boldsymbol{\eta}} = \boldsymbol{\eta} - \bar{\boldsymbol{\eta}}$ becomes

$$\dot{\tilde{\boldsymbol{\eta}}} = -\boldsymbol{\Lambda}\tilde{\boldsymbol{\eta}} - \dot{\bar{\boldsymbol{\eta}}}(\tilde{\boldsymbol{x}}, t), \tag{7.7}$$

which can be shown using the Comparison Lemma [95] that

$$\|\tilde{\boldsymbol{\eta}}(t)\| \leq \|\tilde{\boldsymbol{\eta}}(t_0)\| \, e^{-\underline{\lambda}(t-t_0)} + \frac{1}{\underline{\lambda}} \sup_{\tilde{\boldsymbol{x}}, t} \left\|\dot{\bar{\boldsymbol{\eta}}}(\tilde{\boldsymbol{x}}, t)\right\| \tag{7.8}$$

with $\underline{\lambda} = \lambda_{\min}(\boldsymbol{\Lambda})$ being the minimum first-order gain of the actuators. Thus the actuation error $\|\tilde{\boldsymbol{\eta}}(t)\|$ converges exponentially to a bounded region determined by $\left\|\dot{\bar{\boldsymbol{\eta}}}(\tilde{\boldsymbol{x}}, t)\right\|$, which is affected by the smoothness of trajectory as seen in (7.5). We propose to extend the original controller with command derivative feedback to overcome such deficiency.

Theorem 7.1. *With system defined in (7.2) and (7.6), and controller $\bar{\boldsymbol{\eta}}(\tilde{\boldsymbol{x}}, t)$ that satisfies (7.4), the augmented controller*

$$\boldsymbol{u}' = \bar{\boldsymbol{\eta}}'(\tilde{\boldsymbol{x}}, \boldsymbol{\eta}, t) = \bar{\boldsymbol{\eta}}(\tilde{\boldsymbol{x}}, t) + \boldsymbol{\Lambda}^{-1}\dot{\bar{\boldsymbol{\eta}}}(\tilde{\boldsymbol{x}}, \boldsymbol{\eta}, t) \tag{7.9}$$

exponentially stabilizes the closed-loop systems (7.2) and (7.7).

Proof. We choose a candidate Lyapunov function $\mathcal{V}_1 = \mathcal{V} + \alpha \|\tilde{\boldsymbol{\eta}}\|^2$, where $\mathcal{V}$ is from (7.4) and $\alpha > (c_4^2 L_g^2)/(8c_3\underline{\lambda})$. Using (7.4), (7.6), and (7.9), we differentiate $\mathcal{V}_1$ with respect to t and obtain

$$\begin{aligned}
\dot{\mathcal{V}}_1 &= \frac{\partial \mathcal{V}}{\partial t} + \frac{\partial \mathcal{V}}{\partial \tilde{\boldsymbol{x}}}\big(\boldsymbol{g}(\tilde{\boldsymbol{x}}, \boldsymbol{\eta}, t) \pm \boldsymbol{g}(\tilde{\boldsymbol{x}}, \bar{\boldsymbol{\eta}}, t)\big) + 2\alpha\tilde{\boldsymbol{\eta}}^\top \dot{\tilde{\boldsymbol{\eta}}} \\
&\leq -c_3 \|\tilde{\boldsymbol{x}}\|^2 + \frac{\partial \mathcal{V}}{\partial \tilde{\boldsymbol{x}}}\big(\boldsymbol{g}(\tilde{\boldsymbol{x}}, \boldsymbol{\eta}, t) - \boldsymbol{g}(\tilde{\boldsymbol{x}}, \bar{\boldsymbol{\eta}}, t)\big) \\
&\qquad + 2\alpha\tilde{\boldsymbol{\eta}}^\top\big(- \boldsymbol{\Lambda}\boldsymbol{\eta} + \boldsymbol{\Lambda}\boldsymbol{u}' - \dot{\bar{\boldsymbol{\eta}}}\big) \\
&\leq -c_3 \|\tilde{\boldsymbol{x}}\|^2 + c_4 L_g \|\tilde{\boldsymbol{x}}\| \|\tilde{\boldsymbol{\eta}}\| - 2\alpha\underline{\lambda} \|\tilde{\boldsymbol{\eta}}\|^2 \\
&\leq - \begin{bmatrix} \|\tilde{\boldsymbol{x}}\| \\ \|\tilde{\boldsymbol{\eta}}\| \end{bmatrix}^\top \underbrace{\begin{bmatrix} c_3 & -c_4 L_g/2 \\ -c_4 L_g/2 & 2\alpha\underline{\lambda} \end{bmatrix}}_{\boldsymbol{K}_1} \begin{bmatrix} \|\tilde{\boldsymbol{x}}\| \\ \|\tilde{\boldsymbol{\eta}}\| \end{bmatrix} \\
&\leq -c_3' \|\boldsymbol{\theta}\|^2
\end{aligned} \tag{7.10}$$

with $\boldsymbol{\theta} = [\tilde{\boldsymbol{x}}; \tilde{\boldsymbol{\eta}}]$ being the combined error vector, $\boldsymbol{K}_1 > 0$, if $\alpha > (c_4^2 L_g^2)/(8c_3\underline{\lambda})$, and $c_3' = \lambda_{\min}(\boldsymbol{K}_1)$. Furthermore, let $c_1' = \min\{c_1, \alpha\}$ and $c_2' = \max\{c_2, \alpha\}$, so we can get $c_1' \|\boldsymbol{\theta}\|^2 \leq \mathcal{V}_1 \leq c_2' \|\boldsymbol{\theta}\|^2$. Consequently,

$$\|\boldsymbol{\theta}(t)\| \leq \sqrt{\frac{c_2'}{c_1'}} \|\boldsymbol{\theta}(t_0)\| \exp\left(-\frac{c_3'}{2c_2'}(t - t_0)\right),$$

which proves that $[\tilde{\boldsymbol{x}}; \tilde{\boldsymbol{\eta}}]$ converges exponentially with rate $c_3'/(2c_2')$. ■

Remark 7.1. *Equivalently, if $\dot{\tilde{\boldsymbol{x}}}$ is available through direct measurement or numerical differentiation, then*

$$\dot{\bar{\boldsymbol{\eta}}}(\tilde{\boldsymbol{x}}, \dot{\tilde{\boldsymbol{x}}}, t) = \frac{\partial \bar{\boldsymbol{\eta}}}{\partial \tilde{\boldsymbol{x}}} \dot{\tilde{\boldsymbol{x}}} + \frac{\partial \bar{\boldsymbol{\eta}}}{\partial t} \tag{7.11}$$

and controller (7.9) can be implemented without $\boldsymbol{\eta}$ feedback. Nevertheless, the rate of convergence is limited by $\underline{\lambda}$ of the underlying actuators.

7.2.2 Improved Delay Compensation with Actuator Observer

An actuator-observer is needed if we were to increase the convergence rate on $\tilde{\boldsymbol{\eta}}$ beyond $\boldsymbol{\Lambda}$. We define $\hat{\boldsymbol{\eta}} \in \mathbb{R}^m$ to be the estimation of $\boldsymbol{\eta}$, and the observer error is their difference $\boldsymbol{\eta}_e = \hat{\boldsymbol{\eta}} - \boldsymbol{\eta}$. In this work, we assume that an observer with the following property is available.

Assumption 7.4. *An $\boldsymbol{\eta}$ observer can be designed such that the closed-loop dynamics of estimation error satisfies*

$$\dot{\boldsymbol{\eta}}_e = -\boldsymbol{\Omega}(\tilde{\boldsymbol{x}}, t)\boldsymbol{\eta}_e, \tag{7.12}$$

where $\boldsymbol{\Omega}(\tilde{\boldsymbol{x}}, t)$ is always positive definite. We can define its minimum and maximum eigenvalues as $\underline{\omega} = \inf_{\tilde{\boldsymbol{x}},t} \lambda_{\min}\boldsymbol{\Omega}(\tilde{\boldsymbol{x}}, t)$, $\overline{\omega} = \sup_{\tilde{\boldsymbol{x}},t} \lambda_{\max}\boldsymbol{\Omega}(\tilde{\boldsymbol{x}}, t)$.

A trivial observer of such type is $\dot{\hat{\boldsymbol{\eta}}} = -\boldsymbol{\Lambda}\hat{\boldsymbol{\eta}} + \boldsymbol{\Lambda}\boldsymbol{u}'$, since the first-order delay is a stable system. However, if we want to increase the rate of convergence of $\tilde{\boldsymbol{\eta}}$, it would be favorable to have $\underline{\omega} > \underline{\lambda}$. With the availability of the measurement stated in Assumption 7.2, a reduced-order Luenberger observer for a linear system or a contraction-based PD observer for a nonlinear system [130] can be utilized.

The observer-based delay compensation controller that increases the overall rate of convergence is stated as follows.

Theorem 7.2. *With the system defined in (7.2) and (7.6), and controller $\bar{\boldsymbol{\eta}}(\tilde{\boldsymbol{x}}, t)$ that satisfies (7.4), the augmented controller that incorporates the estimated actuator* *input*

$$\begin{aligned} \boldsymbol{u}' &= \bar{\boldsymbol{\eta}}''(\tilde{\boldsymbol{x}}, \hat{\boldsymbol{\eta}}, t) \\ &= (I - \boldsymbol{\Lambda}^{-1}\Gamma)\hat{\boldsymbol{\eta}} + \boldsymbol{\Lambda}^{-1}\Gamma\bar{\boldsymbol{\eta}}(\tilde{\boldsymbol{x}}, t) + \boldsymbol{\Lambda}^{-1}\dot{\bar{\boldsymbol{\eta}}}(\tilde{\boldsymbol{x}}, \hat{\boldsymbol{\eta}}, t) \end{aligned} \tag{7.13}$$

exponentially stabilizes the closed-loop systems (7.2) and (7.7) with an increased rate of convergence than the controller (7.9).

Proof. Similarly to Theorem 7.1, we select a candidate Lyapunov function $\mathcal{V}_2 = \mathcal{V} + \alpha \|\tilde{\boldsymbol{\eta}}\|^2 + \beta \|\boldsymbol{\eta}_e\|^2$. Taking time-derivative and substituting in (7.4), (7.6), (7.12), and (7.13), we get the following relationship after some simplifications:

$$\begin{aligned} \dot{\mathcal{V}}_2 &= \dot{\mathcal{V}} + 2\alpha\tilde{\boldsymbol{\eta}}^\top\dot{\tilde{\boldsymbol{\eta}}} + 2\beta\boldsymbol{\eta}_e^\top\dot{\boldsymbol{\eta}}_e \\ &\le - \begin{bmatrix} \|\tilde{\boldsymbol{x}}\| \\ \|\tilde{\boldsymbol{\eta}}\| \\ \|\boldsymbol{\eta}_e\| \end{bmatrix}^\top \underbrace{\begin{bmatrix} c_3 & -c_4L_g/2 & 0 \\ -c_4L_g/2 & 2\alpha\underline{\gamma} & -\alpha\rho \\ 0 & -\alpha\rho & 2\beta\underline{\omega} \end{bmatrix}}_{K_2} \begin{bmatrix} \|\tilde{\boldsymbol{x}}\| \\ \|\tilde{\boldsymbol{\eta}}\| \\ \|\boldsymbol{\eta}_e\| \end{bmatrix} \\ &\le -c_3'' \|\boldsymbol{z}\|^2 . \end{aligned} \tag{7.14}$$

We define the combined error vector $\boldsymbol{z} = [\tilde{\boldsymbol{x}}; \tilde{\boldsymbol{\eta}}; \boldsymbol{\eta}_e]$, constants $\underline{\gamma} = \lambda_{\min}(\Gamma)$ and $\rho = \lambda_{\max}(\Gamma - \boldsymbol{\Lambda})$. If we choose α and β such that

$$\begin{aligned} \alpha &> (c_4^2L_g^2)/(8c_3\underline{\lambda}) \\ \beta &> \frac{2c_3\alpha^2\rho^2}{\underline{\omega}(8c_3\alpha\underline{\gamma} - c_4^2L_g^2)}, \end{aligned}$$

then we can guarantee $K_2 \succ 0$ and define $c_3'' = \lambda_{\min}(K_2)$. Letting $c_1'' = \min\{c_1, \alpha, \beta\}$, $c_2'' = \max\{c_2, \alpha, \beta\}$, and consequently $c_1'' \|\boldsymbol{z}\|^2 \le \mathcal{V}_2 \le c_2'' \|\boldsymbol{z}\|^2$, we obtain

$$\|\boldsymbol{z}(t)\| \le \sqrt{\frac{c_2''}{c_1''}} \|\boldsymbol{z}(t_0)\| \exp\left(-\frac{c_3''}{2c_2''}(t - t_0)\right),$$

which proves that $[\tilde{\boldsymbol{x}}; \tilde{\boldsymbol{\eta}}; \boldsymbol{\eta}_e]$ converges exponentially with rate $c_3''/(2c_2'')$ ■

Remark 7.2. *Although the overall rate of convergence is improved with the introduction of the observer (7.12), tracking performance is now tied with the estimation error $\boldsymbol{\eta}_e$, which will be affected by sensor noise or model error in practice.*

Remark 7.3. *When setting $\Gamma = \Lambda$, (7.13) reduces to (7.9), and the dependence on $\hat{\boldsymbol{\eta}}$ is dropped. Thus we can treat Theorem 7.1 as a special case of Theorem 7.2.*

7.2.3 Numerical Predictive Control under Periodic Sampling

Starting with the continuous time formulation from Theorem 7.2, we propose to extend the controller with predicted future states to account for transport delays. In the literature (e.g, [74]), predictors are often treated as continuous integration of dynamics from the current state:

$$\hat{\boldsymbol{x}}(t+\Delta) = \boldsymbol{x}(t) + \int_t^{t+\Delta} \boldsymbol{f}(\hat{\boldsymbol{x}}(s), \boldsymbol{u}(s-\Delta), s)ds.$$

Instead, we consider a predictor in the form of discrete numerical integration. Our controller is activated periodically at sample times t_i. A general fixed step-size Runge-Kutta (RK) integration method is then used to predict state and actuator input at $t_i' = t_i + \Delta$

$$\begin{bmatrix} \hat{\boldsymbol{x}}_{\mathrm{RK}}(t_i') \\ \hat{\boldsymbol{\eta}}_{\mathrm{RK}}(t_i') \end{bmatrix} = \mathcal{F}_{\mathrm{RK}}\left(\boldsymbol{x}(t_i), \hat{\boldsymbol{\eta}}(t_i), t_i, \Delta, h, p\right). \tag{7.15}$$

We denote $\mathcal{F}_{\mathrm{RK}}(\cdot)$ as the integration scheme, with accuracy of order p, stepsize h, and time horizon Δ.

For ease of analysis, we vertically stack $\boldsymbol{z} = [\tilde{\boldsymbol{x}}; \tilde{\boldsymbol{\eta}}; \boldsymbol{\eta}_e]$, and rewrite $\boldsymbol{\eta} = \bar{\boldsymbol{\eta}}(\tilde{\boldsymbol{x}}, t) + \tilde{\boldsymbol{\eta}}$. Then we have

$$\dot{\boldsymbol{z}} = \begin{bmatrix} \boldsymbol{g}\left(\tilde{\boldsymbol{x}}, \boldsymbol{\eta}, t\right) \\ -\boldsymbol{\Lambda}\bar{\boldsymbol{\eta}}(\tilde{\boldsymbol{x}}, t) - \boldsymbol{\Lambda}\tilde{\boldsymbol{\eta}} - \dot{\bar{\boldsymbol{\eta}}}(\tilde{\boldsymbol{x}}, \boldsymbol{\eta}, t) + \boldsymbol{\Lambda}\boldsymbol{u} \\ -\boldsymbol{\Omega}(\tilde{\boldsymbol{x}}, t)\boldsymbol{\eta}_e \end{bmatrix} = \xi(\boldsymbol{z}, \boldsymbol{u}, t). \tag{7.16}$$

Furthermore, we limit $p \in \{1, 2, 3, 4\}$ and make the following assumption about the bound on the integration error.

Assumption 7.5. *The integration error from t_i to $t_i + \Delta$ is bounded by E_{RK} as*

$$\|\hat{\boldsymbol{z}}_{\mathrm{RK}} - \boldsymbol{z}\|_{t_i'} \leq E_{\mathrm{RK}} = \frac{Mh^p + w}{L_{\mathrm{RK}}}\left(e^{L_{\mathrm{RK}}\Delta} - 1\right). \tag{7.17}$$

L_{RK} is the Lipschitz constant of the one-step RK function [138]; w is the upper bound on model error; and M is a constant related to the smoothness of $\xi(\cdot)$.

Before stating the result for the predictive controller, we define the following useful quantities based on the Lipschitz constants of $\boldsymbol{g}(\cdot)$, $\bar{\boldsymbol{\eta}}(\cdot)$ and $\dot{\bar{\boldsymbol{\eta}}}(\cdot)$:

$$\mu = \sqrt{3}\max\left\{\rho + L_{\dot{\bar{\eta}}},\ \overline{\lambda}L_{\bar{\eta}} + L_{\dot{\bar{\eta}}}(1 + L_{\bar{\eta}})\right\} \tag{7.18}$$

$$\nu = \sqrt{3}\max\left\{\overline{\lambda}L_{\bar{\eta}} + (L_g + L_{\dot{\bar{\eta}}})(1 + L_{\bar{\eta}}),\right. \tag{7.19}$$

$$\left.\overline{\lambda} + L_{\dot{\bar{\eta}}} + L_g,\ \overline{\omega} + L_{\dot{\bar{\eta}}}\right\}$$

$$\nu_0 = \sqrt{3}\max\left\{L_g(1 + L_{\bar{\eta}}),\ \overline{\gamma} + L_g,\ \rho + \overline{\omega}\right\}. \tag{7.20}$$

The numerical predictive controller under periodic sampling can be stated as follows.

Theorem 7.3. *At $t = t_i$, prediction $\hat{z}_{\text{RK}}(t_i + \Delta)$ can be estimated from numerical integration with (7.15). The predictive controller is defined from (7.13) as*

$$u(t_i) = \bar{\eta}''\big(\hat{z}_{\text{RK}}(t_i + \Delta), t_i + \Delta\big). \tag{7.21}$$

Suppose the sampling period satisfies

$$T_s < \frac{1}{\nu} \ln\left[1 + \left(\frac{\nu}{\nu_0}\right)\frac{c_3''}{2\alpha\mu}\right]. \tag{7.22}$$

Then $\exists\ 0 < \delta_1 \le \delta_2$ such that the overall system (7.16) is exponentially stable for $\delta_1 \le \|z\| \le \delta_2$ under (7.21).

Proof. We start from the same Lyapunov candidate $\mathcal{V}_2(z)$ as in Theorem 7.2. Differentiating $\mathcal{V}_2$ with respect to time and substituting in (7.16) and (7.21), we get the following inequalities after simplification

$$\begin{aligned}\dot{\mathcal{V}}_2 &= \dot{\mathcal{V}} + 2\alpha\tilde{\eta}^\top\dot{\tilde{\eta}} + 2\beta\eta_e^\top\dot{\eta}_e \\ &\le -c_3''\,\|z\|^2 + 2\alpha\mu\,\|z\|\left\{\left\|\hat{z}_{\text{RK}}(t_i') - z(t_i')\right\|\right. \\ &\qquad\qquad \left. + \left\|z(t) - z(t_i')\right\| + (1/\sqrt{3})\left\|t - t_i'\right\|\right\}.\end{aligned}$$

We can express $z(t) = z(t_i') + \int_{t_i'}^{t} \xi\big(z(s), u(t_i), s\big)ds$ using (7.16) and (7.21). The inequality can be reduced to

$$\begin{aligned}\left\|z(t) - z(t_i')\right\| \le\ & \mu\left\|\hat{z}_{\text{RK}}(t_i') - z(t_i')\right\|(t - t_i') \\ & + \nu_0\left\|z(t_i')\right\|(t - t_i') + \frac{1}{2\sqrt{3}}(t - t_i')^2 \\ & + \int_{t_i'}^{t} \nu\left\|z(s) - z(t_i')\right\|ds\end{aligned}$$

with (7.3), (7.20), and Assumption 7.1. We can apply Grönwall's lemma to the above inequality; and (7.17) to $\left\|\hat{z}_{\text{RK}}(t_i') - z(t_i')\right\|$:

$$\begin{aligned}\dot{\mathcal{V}}_2 \le\ & -c_3''\,\|z\|^2 + 2\alpha\mu\,\|z\|\left\{E_{\text{RK}}\right. \\ & \left. + \left(\frac{\nu_0}{\nu}\left\|z(t_i')\right\| + \frac{\mu}{\nu}E_{\text{RK}} + \frac{1}{\sqrt{3}\nu}\right)\left(e^{\nu(t - t_i')} - 1\right)\right\}.\end{aligned}$$

Thus, for any sampling period that satisfies (7.22), the following equation holds

$$T_s = \frac{1}{\nu} \ln \left[1 + \left(\frac{\nu}{\nu_0} \right) \frac{\phi c_3''}{2\alpha\mu} \right] \tag{7.23}$$

with $\phi \in (0, 1)$. We can define ϵ such that $0 < \phi < \sqrt{\phi} < \epsilon < 1$. Therefore, for any

$$\delta \geq \frac{\frac{2\alpha\mu\nu_0}{c_3''} E_{\text{RK}} + (\mu E_{\text{RK}} + \frac{1}{\sqrt{3}})\phi}{\nu_0(\epsilon^2 - \phi)}, \tag{7.24}$$

it can be shown using (7.23) that

$$\frac{2\alpha\mu}{\epsilon c_3''} \left[E_{\text{RK}} + \frac{1}{\nu} \left(\nu_0 \delta + \mu E_{\text{RK}} + \frac{1}{\sqrt{3}} \right) \left(e^{\nu T_s} - 1 \right) \right] \leq \epsilon \delta$$

and we can state that if $\forall \, \|z\| \in [\epsilon\delta, \delta]$, we have $\dot{\mathcal{V}}_2 \leq -c_3''(1-\epsilon) \, \|z\|^2$, and therefore

$$\|z(t)\| \leq \sqrt{\frac{c_2''}{c_1''}} \, \|z(t_0)\| \exp \left(-\frac{c_3''(1-\epsilon)}{2c_2''} (t - t_0) \right),$$

which guarantees exponential convergence with rate $c_3''(1-\epsilon)/(2c_2'')$. Setting $\delta_1 = \epsilon\delta$ and $\delta_2 = \delta$ completes the proof. ∎

Remark 7.4. *From (7.24), it can be shown that $\epsilon\delta$ is lower bounded by*

$$\epsilon\delta \geq \frac{\frac{2\alpha\mu\nu_0}{c_3''} E_{\text{RK}} + \left(\mu E_{\text{RK}} + \frac{1}{\sqrt{3}} \right) \phi}{\nu_0(1 - \phi)}. \tag{7.25}$$

This gives an asymptotic region within which exponential convergence is not proven sufficiently. As $\phi \to 0$, we get its continuous limit $2\alpha\mu E_{\text{RK}}/c_3''$.

Remark 7.5. *The limit for sampling time in (7.22) is a sufficient condition that considers the worst case of which sampling error $\left\| z(t) - z(t_i') \right\|$ can grow during $t \in [t_i', t_{i+1}']$. In reality, a sampling period higher than the bound can still yield reasonable stability, as seen in event-triggered controllers [134].*

7.2.4 Effects of Numerical Prediction Scheme on Delay

In the case of our proposed predictive controller (7.21), we postulate that Δ_c is mainly affected by computation of the predictor (7.15) and the controller $\bar{\eta}''(\cdot)$. The predictor integrates $(\Delta_c + \Delta_s)/h$ steps of target function using RK method of order $p \in \{1, 2, 3, 4\}$, which requires the evaluation of target function p times. Thus Δ_c can be written as

$$\Delta_c = \frac{\Delta_c + \Delta_s}{h} (pC_f + C_0) + C_\eta,$$

and C_f, $C_{\boldsymbol{\eta}}$, and C_0 are the respective times for evaluating $\boldsymbol{f}(\cdot)$, $\boldsymbol{\eta}''(\cdot)$, and other related numerical operations. In turn, Δ_c can be solved as

$$\Delta_c = \frac{hC_{\boldsymbol{\eta}} + \Delta_s(pC_f + C_0)}{h - pC_f - C_0}. \tag{7.26}$$

It is clear that $h > pC_f + C_0$ is required for feasible Δ_c, and that h cannot exceed the total delay (i.e $h \le \Delta_c + \Delta_s$). Furthermore, Δ_c needs to fit within sampling period T_s. We can derive that h has to fall within the range:

$$h \in \left[\frac{T_s + \Delta_s}{T_s - C_{\boldsymbol{\eta}}}(pC_f + C_0), \quad \Delta_s + C_{\boldsymbol{\eta}} + pC_f + C_0\right]. \tag{7.27}$$

For a feasible h to exist, it follows directly from the above equation that $T_s \ge C_{\boldsymbol{\eta}} + pC_f + C_0$. Δ can thus be represented in terms of h, p, and other pre-determined quantities:

$$\Delta = (\Delta_s + C_{\boldsymbol{\eta}})\frac{h}{h - pC_f - C_0}. \tag{7.28}$$

Combining (7.17) and (7.28), we obtain

$$E_{\mathrm{RK}} = \frac{Mh^p + w}{L_{\mathrm{RK}}}\left(e^{L_{\mathrm{RK}}\frac{h(\Delta_s + C_{\boldsymbol{\eta}})}{h - pC_f - C_0}} - 1\right), \tag{7.29}$$

which admits a minimum within (7.27). Since the prediction error and E_{RK} affect the overall convergence rate of the system, a choice of h and p will directly affect the controller performance.

7.2.5 Truncated Predictive Control with Numerical Derivative

We can also treat $\boldsymbol{\Lambda}^{-1}$ as the diagonal matrix of time constants for the actuator dynamics. In many cases, $\boldsymbol{\Lambda}^{-1}$, Δ, and T_s are on the same small timescale, i.e $O(\boldsymbol{\Lambda}^{-1}) \sim O(\Delta) \sim O(T) \ll 1$. Using the 1st-order RK method (Euler's method) on (7.9) and applying a backward difference method to $\dot{\bar{\boldsymbol{\eta}}}(t_i)$, we can write the RK predictive controller as

$$\begin{aligned}
\bar{\boldsymbol{\eta}}''(t_i') &= \bar{\boldsymbol{\eta}}(t_i) + (\boldsymbol{\Lambda}^{-1} + \Delta)\dot{\bar{\boldsymbol{\eta}}}(t_i) + O(\boldsymbol{\Lambda}^{-1}\Delta) \\
&= \bar{\boldsymbol{\eta}}(t_i) + O(T_s^2) \\
&\qquad + (\boldsymbol{\Lambda}^{-1} + \Delta)\left[\frac{\bar{\boldsymbol{\eta}}(t_i) - \bar{\boldsymbol{\eta}}(t_{i-1})}{T_s} + O(T_s)\right] \\
&= \bar{\boldsymbol{\eta}}(t_i) + (\boldsymbol{\Lambda}^{-1} + \Delta)\left[\frac{\bar{\boldsymbol{\eta}}(t_i) - \bar{\boldsymbol{\eta}}(t_{i-1})}{T_s}\right] + O(T_s^2)
\end{aligned}$$

where for simplicity we denote $\bar{\boldsymbol{\eta}}''(t_i') = \bar{\boldsymbol{\eta}}''\big(\hat{\boldsymbol{z}}_{\mathrm{RK}}(t_i'), t_i'\big)$, $\bar{\boldsymbol{\eta}}(t_i) = \bar{\boldsymbol{\eta}}(\tilde{\boldsymbol{x}}(t_i), t_i)$, and $\dot{\bar{\boldsymbol{\eta}}}(t_i) = \dot{\bar{\boldsymbol{\eta}}}\big(\tilde{\boldsymbol{x}}(t_i), \dot{\tilde{\boldsymbol{x}}}(t_i), t_i\big)$. We can thus define the first-order truncation of the predictive controller:

$$\bar{\boldsymbol{\eta}}''_{\mathrm{FO}}(t_i') = \bar{\boldsymbol{\eta}}(t_i) + (\boldsymbol{\Lambda}^{-1} + \Delta)\frac{\bar{\boldsymbol{\eta}}(t_i) - \bar{\boldsymbol{\eta}}(t_{i-1})}{T_s}, \tag{7.30}$$

which has a truncation error of $O(T^2)$. The truncated controller (7.30) avoids the evaluation of $\dot{\bar{\boldsymbol{\eta}}}(\cdot)$ and in turn $\boldsymbol{g}(\cdot)$. Similarly to (7.9), it also avoids the need for $\hat{\boldsymbol{\eta}}$ and therefore saving computation on the observer as well. As will be seen in later analysis, (7.30) performs favorably compared to more complex methods for a certain class of systems.

7.3 Numerical Analysis

In this section, we conduct numerical experiments on a delayed double integrator example running our proposed control methods described in Section 7.2.

7.3.1 Example: Delayed Double Integrator

We consider trajectory tracking for simple double integrator dynamics, with delayed force actuation:

$$\dot{\boldsymbol{x}}_1 = \boldsymbol{x}_2, \quad \dot{\boldsymbol{x}}_2 = b\boldsymbol{\eta}_1, \quad \dot{\boldsymbol{\eta}}_1 = -\lambda\boldsymbol{\eta}_1 + \lambda\boldsymbol{u}(t-\Delta). \tag{7.31}$$

Let the scalar states $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ denote position and velocity, respectively. b is a known scalar actuation multiplier, and $\boldsymbol{\eta}_1$ is the actuator input with first-order delay λ. The goal is to track $\boldsymbol{x}_1 \to \boldsymbol{r}(t)$ and $\boldsymbol{x}_2 \to \dot{\boldsymbol{r}}(t)$. The error dynamics are

$$\dot{\tilde{\boldsymbol{x}}}_1 = \tilde{\boldsymbol{x}}_2, \qquad \dot{\tilde{\boldsymbol{x}}}_2 = b\boldsymbol{\eta}_1 - \ddot{\boldsymbol{r}}(t). \tag{7.32}$$

We use a baseline feedback linearizing controller of the form:

$$\bar{\boldsymbol{\eta}}_1(\tilde{\boldsymbol{x}}, t) = b^{-1}\Big(\ddot{\boldsymbol{r}}(t) - k_1\tilde{\boldsymbol{x}}_1 - k_2\tilde{\boldsymbol{x}}_2\Big) \tag{7.33}$$

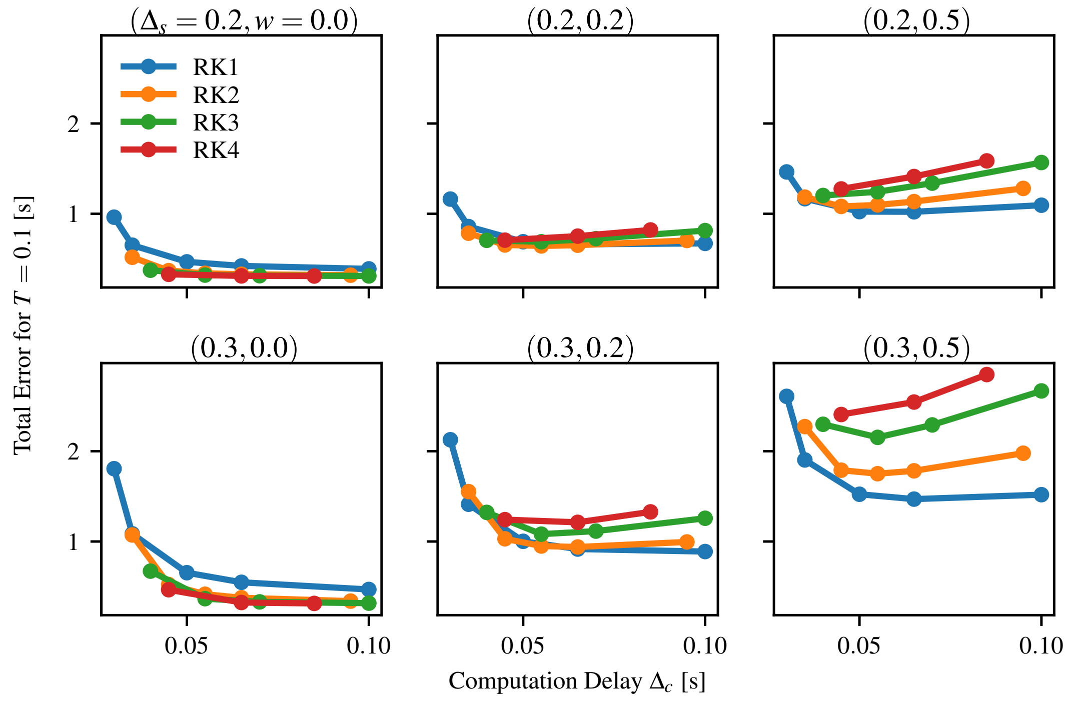
which can be proven to exponentially stabilize the undelayed system. Although the base dynamics are relatively simple, the addition of an aggressive trajectory $\boldsymbol{r}(t)$, large delays, and discrete sampling will pose difficulties for the baseline controller.

Table 7.2: Baseline parameters for delayed double integrator

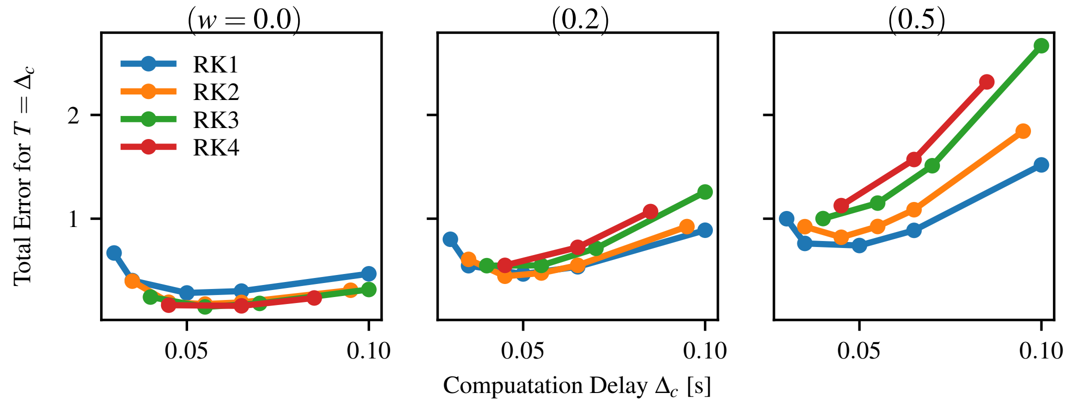
b	λ	k_1	k_2	C_f	C_0	C_η
1.0	5.0	1.0	2.0	0.005	0.0	0.025

We will use this example to study different effects on overall performance from various components of our proposed methods. Table 7.2 lists related parameters for the system that are set or calculated.

7.3.2 Effects of Computation Delay on Control Performance



(a) Sample period fixed at $T_s = 0.1$ s, $\Delta_s \in \{0.2, 0.3\}$s across rows, and $w \in \{0.0, 0.2, 0.5\}$ across columns.



(b) Varying sample period $T_s = \Delta_c$, with $w \in \{0.0, 0.2, 0.5\}$

Figure 7.2: Theoretical Total error bound vs. computation delay Δ_c for different system delay Δ_s and model error w. (Top) Fixed control period $T_s = 0.1$s. (Bottom) Variable control period $T_s = \Delta_c$.

From the Lyapunov analysis in Theorem 7.2, together with (7.29), we can predict trade-offs between integration schemes (p and h) and system stability by plotting

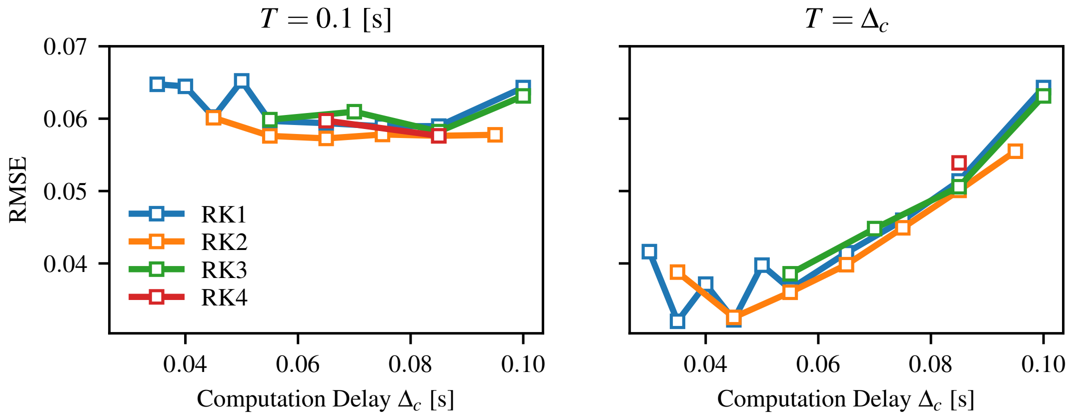


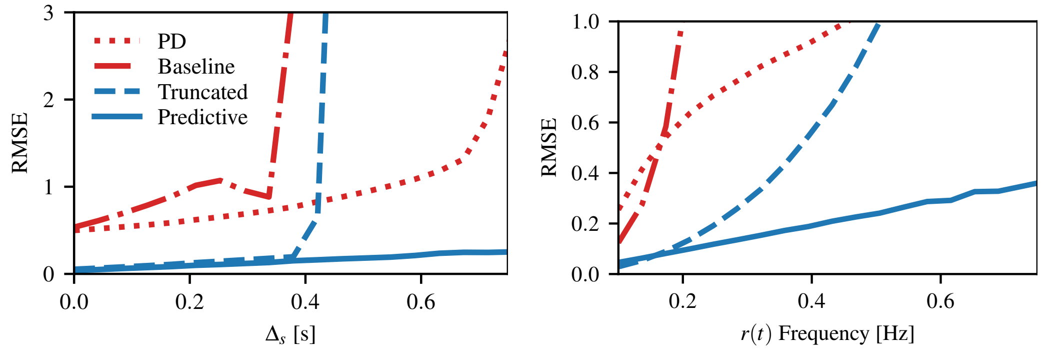
Figure 7.3: Simulation tracking RMSEs for different integration schemes and step sizes. (Left) Fixed sample period $T_s = 0.1$ s. (Right) Variable sample period $T_s = \Delta_c$.

the Lyapunov derivative error bound. In Figure 7.2a, we fix the sample period $T_s = 0.1$ s and examine the variations of system delay Δ_s and model error w. The total error decreases with increasing Δ_c for small w, implying the benefit of maximizing integration accuracy as long as $\Delta_c \leq T_s$. On the other hand, with higher w, we observe a reversal in trend, where increasing numerical complexity no longer decreases total error, and computation delay Δ_c should be minimized for better results. In Figure 7.2b, we adapted the sampling period to the computation delay $T_s = \Delta_c$. These results show that faster computation is strongly favored. Moreover, Euler's method outperforms other higher order RK's when w is high. Results from numerical simulation corroborates our conjecture as shown in Figure 7.3. We observe similar Δ_c versus steady state tracking root-mean-square-error (RMSE) patterns when compared to Figure 7.2.

7.3.3 Control Performance Benchmarks

We conduct comparisons of our proposed controllers, $\bar{\eta}''(\cdot)$ and $\bar{\eta}''_{\text{FO}}(\cdot)$ (given in (7.21) and (7.30), respectively) to the baseline controller $\bar{\eta}(\cdot)$ and a reasonably tuned linear PD controller. The trajectory considered is a sine function of varying frequency. For each test, we let the system run for a horizon of 20 s and then measure the steady-state RMSE. In Figure 7.4a, we simulate the system with different system delays, Δ_s. Throughout the test, the predictive controller $\bar{\eta}''(\cdot)$ maintains a low level of RMSE, even though it is computationally more complex and has a higher Δ_c than the others. As discussed in Section 7.2.5, the truncated control $\bar{\eta}''_{\text{FO}}(\cdot)$ is a first order approximation of the full predictive control scheme. It performs well for $\Delta_s < 0.4$ s, but fails for larger values. The PD control maintains stability for most of the range, but has worse RMSE than $\bar{\eta}''(\cdot)$. Not surprisingly, the naively applied baseline control $\bar{\eta}(\cdot)$ takes high error and becomes unstable even for moderate Δ_s. Figure 7.4b shows almost identical rankings in RMSEs. It is interesting to note that $\bar{\eta}''_{\mathrm{FO}}(\cdot)$ outperforms its more sophisticated counterpart $\bar{\eta}''(\cdot)$ for small delay, likely due to significant reduction in computation cost.

To characterize the performance of $\bar{\eta}''_{\mathrm{FO}}(\cdot)$ on different types of delay, we put it through varying combinations of λ and Δ. Figure 7.5 shows that the truncated controller is delay-type agnostic when $\Delta + 1/\lambda$ is moderate, and has trouble dealing with larger Δ. This is expected since the assumption of $O(1/\lambda) \sim O(\Delta)$ breaks down for large Δ.



(a) RMSE vs. system delay Δ_s. (b) RMSE vs. $r(t)$ frequency.

Figure 7.4: Comparisons of PD, baseline $\bar{\eta}(\cdot)$, truncated $\bar{\eta}''_{\mathrm{FO}}(\cdot)$, and full predictive control $\bar{\eta}''(\cdot)$. RMSE is steady-state root-mean-square-error.

7.4 Chapter Summary

We proposed a control augmentation strategy that transformed exponentially-stabilizing controllers for an undelayed system to a class of sample-based, predictive controllers with numerical integration. The predictive controllers exponentially stabilized the corresponding sample-based system with FOPDT delay, and varied computation complexity for different performance requirements. We performed hybrid stability analysis on the overall system, which provided insights on how features such as sampling period and integration step and order affected output stability. We demonstrated the efficacy of our methods through numerical analysis of our theoretical bounds and simulations of a delayed double integrator system using our proposed control methods. Our analysis demonstrated the often overlooked importance of computation delay in control design. In conclusion, our predictive controller and its truncated variants provided an easily applicable improvement for discrete control tasks in different computation, network, and dynamic environments.

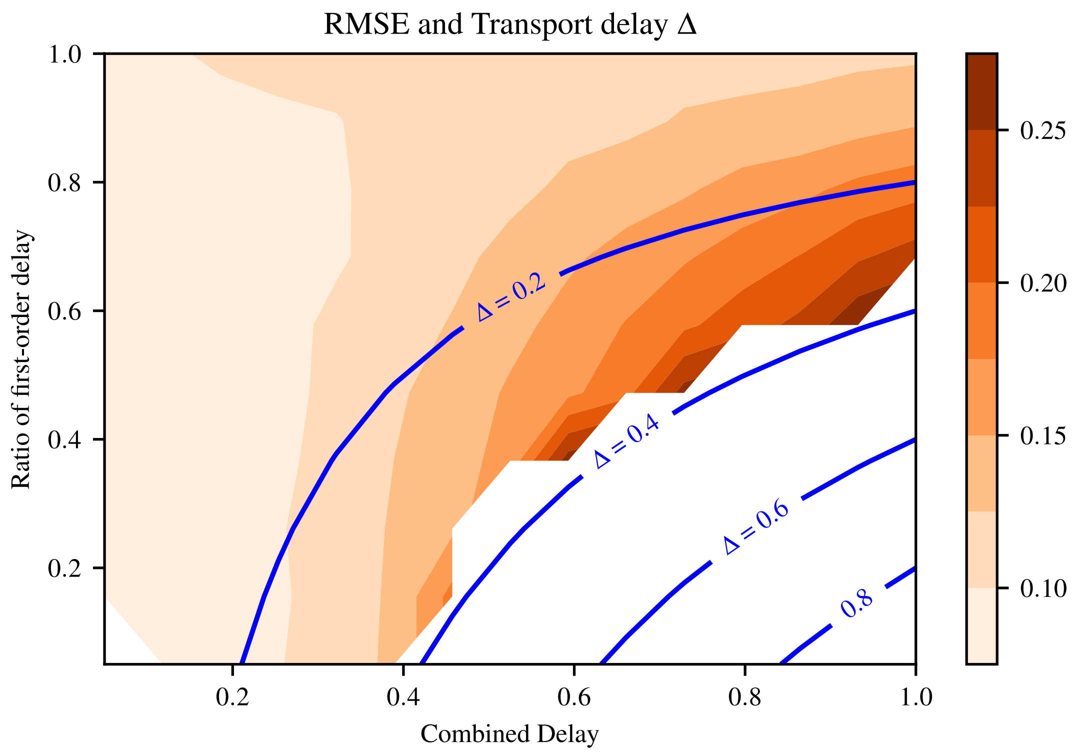


Figure 7.5: Contour of RMSE and transport delay Δ for truncated predictive control $\bar{\eta}''_{\mathrm{FO}}(\cdot)$ (7.30). The horizontal axis is the combined delay $(\Delta + 1/\lambda)$ in seconds. The vertical axis is its ratio of first-order delay $1/(\lambda\Delta + 1)$.

Chapter 8

CONCLUSION

Efficient vertical and long-range flight capabilities were indispensable for Urban Air Mobility (UAM) applications. Although VTOL rotorcraft and fixed-wing aircraft had been thoroughly studied, the combined and hybrid capability of the two different flying mechanisms for a next-generation eVTOL vehicle posed many technological challenges especially in understanding the underlying dynamical behaviors and system-level particularities. In Chapter 2, we used momentum theory and linear aerodynamic models to construct prediction methods for power required during flight for such a hybrid aircraft. When applied to example VTOL designs of our CAST Autonomous Flying Ambulance (AFA) project, the method showed that a fixed-wing flight mode with lift assist from vertical rotors possessed a clear advantage over a multirotor mode. The endurance and range boosts were as much as twice on a $1/5$ scaled eVTOL prototype. It was expected that this superiority would hold for full scale UAM vehicles, consistent with the claim from the Uber Elevate program [1].

The novel control architecture developed in Chapter 3 unified control methods across different types of vehicles and flight modes. As fixed-wing VTOL vehicles would be used in cluttered urban environments, it was expected to frequently encounter transient aerodynamic disturbances, in addition to the already-challenging models of rotor-wing interactions. With traditional control methods for hybrid VTOL focusing on separate design schemes and transition strategies, flight performance could not be fully utilized for mixed modes of vertical and forward flights, and control accuracy was questionable during the transition phase. In contrast, our method split the objective into two parts. First, nonlinear position/velocity and attitude/rate control designs using forces and moments as input were discussed in Section 3.2. The unified control scheme was easily proven to be globally exponentially stable, given a property of robustness to bounded force and moment tracking errors. Second, concepts of force allocation and control allocation to realize desired output wrench were expanded on in Sections 3.3 and 3.4. Attainable force, moments, and control spaces were analyzed to give a realtime verifiable set in order to avoid control saturation. A prototype hybrid VTOL vehicle was constructed to validate the results. The innovative force allocation scheme eliminated the need for separate control designs and significantly simplified future implementation and maintenance. The provably stable and robust method handled the fixed-wing VTOL control with ease, as shown in results of simulation and experimentation from Section 3.5, where the prototype vehicle achieved transition behavior without a explicit trajectory design.

Nevertheless, we identified that accuracy of force allocation was crucial to a fixed-wing VTOL aircraft under substantial aerodynamic forces, which prompted further development of force estimation methods in Chapter 4. In Section 4.1, a set of LiP models derived from basic aerodynamic equations incorporated greater details on rotor side force. It enabled the inclusion of composite adaptation of model parameters based on trajectory tracking errors and force prediction errors. To further improve force prediction accuracy, Section 4.3 introduced a 3D airflow sensor that provides timely information on environmental airflow around the vehicle. We proved the stability and robustness of the adaptive force allocation and control for high speed transition flight in Section 4.2. The method vastly decreased the position tracking error during fast transitions of flight stages. Transient aerodynamic effects that caused baseline non-adaptive methods to temporally drift away from commanded trajectory were completely compensated by the improved controller shown in the results from Section 4.4.

For safety, Chapter 5 derived controllability-based optimization that increased the vehicles' authority during rotor failures. From the definition of null controllability for multirotors introduced in Section 5.1, we established an optimization routine in Section 5.2. The moment space of a multirotor was maximized with respect to combinations of different rotor failure cases, by adjusting physical configuration parameters of the aircraft design. Section 5.3 compared an optimized multirotor design for AFA 1.0 with a baseline configuration. By checking the controllability quality measure for each rotor failure case, we saw substantial improvements in all of them. Especially in some situations when a naive design lose all authority in directional yaw, the optimized design still performed a stabilization task. This was also corroborated by experimental results on a hardware prototype for multirotor position control.

Continued improvement of control performance for flights would require the inclusion of greater amount of sensor information. Since our vehicle already possessed a substantial amount of data collected from its past flights; it was natural to apply data-driven methods for continual learning and improvements. In Chapter 6, the method of using DNNs was studied as a performance multiplier to the existing physics-based approach. The learning capacity of a deep ReLU network was considered in

Section 6.1. By building on prior results on spectral normalization of DNNs [70], we elucidated that the process of spectral normalization by regulating Lipschitz constants of a DNNs presented important benefits to a learning-based control task in addition to its generalization ability. When inputting both state and control in DNN training, its prediction capability was expanded, but at the same time made the system non-affine-in-control. We presented a DNN-based nonlinear controller in Section 6.2 that used the Lipschiz property of the network during training to construct a contraction mapping thereby solving the issue of nonlinearity of the control input. The proposed DNN-based controller had guaranteed stability, making it one of the first provably-stable DNN-based controller. When applied to a quadrotor drone in Section 6.3, our learning and control method set a new record for flight tasks near ground or large objects, achieving zero-speed touchdown during landing and holding an accurate trajectory during a sudden drop in height near the table edges. Furthermore, the spectrally-normalized DNN showed better prediction performance on an unseen dataset compared to the ones trained without the spectral normalization technique. Incorporating partial DNN dynamics required the known physics-based model to be reasonably accurate and that the Lipschitz constant of residual DNN was constrained to an acceptable range for the contraction mapping property to hold. Section 6.4 instead used a DNN for modeling the entire dynamics, thereby further producing controller and observer networks based on episodic and recurrent training. The method could be adapted to any dynamic models at the cost of weaker theoretical guarantees. Nevertheless, it still possessed advantages over deep Reinforcement Learning (RL) approaches for the particular dynamics-controller-observer structure designed specifically for output tracking problems.

In the temporal domain, actuation delay hindered further improvement in control performance. Chapter 7 proposed methods that addressed delay directly. Starting from the existing controllers designed to exponentially stabilize a general nonlinear system, Section 7.2 introduced a novel augmentation scheme with predictive elements to account for FOPDT delays and transport delays. The proposed predictive method relied on numerical integration schemes to turn the overall system to a hybrid one. All the proposed augmented controllers had discrete computations. The resulting sample-based predictive controller was analyzed with hybrid stability analysis to prove its validity under certain time constraints. In addition, the accuracy-computation trade-offs of different RK methods were discussed in Section 7.2.4. Although higher order methods gave better prediction results, they did not always translate to better control performance due to their heavier computation requirement. These effects were useful for all digital control systems, but were rarely studied. In Section 7.3, results on different integration schemes for trajectory tracking tasks were compared, showing that spending more time to compute for better accuracy versus updating the control faster were both beneficial under different scenarios. In particular, higher system delay or modeling uncertainty was the key incentive in using a faster yet less accurate prediction scheme. Section 7.3 illustrated that our proposed augmentation improved over the baseline result drastically, and only incurred small computation burden. This justified the need for including such time-delay prediction techniques in real-time flight control applications.

BIBLIOGRAPHY

[1] J. Holden and N. Goel, "Uber elevate: Fast-forwarding to a future of on-demand urban air transportation," Uber, Tech. Rep., Oct. 2016.

[2] M. D. Moore, "Distributed electric propulsion (DEP) aircraft," NASA Langley Research Center, 2012.

[3] A. Frank, J. McGrew, M. Valenti, D. Levine, and J. How, "Hover, transition, and level flight control design for a single-propeller indoor airplane," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2007, p. 6318.

[4] R. H. Stone, P. Anderson, C. Hutchison, A. Tsai, P. Gibbens, and K. Wong, "Flight testing of the T-wing tail-sitter unmanned air vehicle," *Journal of Aircraft*, vol. 45, no. 2, pp. 673–685, 2008.

[5] A. B. Chowdhury, A. Kulhare, and G. Raina, "Back-stepping control strategy for stabilization of a tilt-rotor UAV," in *Chinese Control and Decision Conference (CCDC)*, IEEE, 2012, pp. 3475–3480.

[6] S. Park, J. Bae, Y. Kim, and S. Kim, "Fault tolerant flight control system for the tilt-rotor UAV," *Journal of the Franklin Institute*, vol. 350, no. 9, pp. 2535–2559, 2013.

[7] S. Verling, T. Stastny, G. Bättig, K. Alexis, and R. Siegwart, "Model-based transition optimization for a VTOL tailsitter," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 3939–3944.

[8] A. Oosedo, S. Abiko, A. Konno, and M. Uchiyama, "Optimal transition from hovering to level-flight of a quadrotor tail-sitter UAV," *Autonomous Robots*, vol. 41, no. 5, pp. 1143–1159, 2017.

[9] J. Zhou, X. Lyu, Z. Li, S. Shen, and F. Zhang, "A unified control method for quadrotor tail-sitter UAVs in all flight modes: Hover, transition, and level flight," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2017, pp. 4835–4841.

[10] R. Ritz and R. D'Andrea, "A global controller for flying wing tailsitter vehicles," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 2731–2738.

[11] P. Menon, M. Badgett, R. Walker, and E. Duke, "Nonlinear flight test trajectory controllers for aircraft," *Journal of Guidance, Control, and Dynamics*, vol. 10, no. 1, pp. 67–72, 1987.

[12] M.-D. Hua, T. Hamel, P. Morin, and C. Samson, "A control approach for thrust-propelled underactuated vehicles and its application to VTOL drones," *IEEE Transactions on Automatic Control*, vol. 54, no. 8, pp. 1837–1853, 2009.

[13] A. M. Stoll, J. Bevirt, M. D. Moore, W. J. Fredericks, and N. K. Borer, "Drag reduction through distributed electric propulsion," in *14th AIAA Aviation Technology, Integration, and Operations Conference*, 2014, p. 2851.

[14] S. Rajappa, M. Ryll, H. H. Bülthoff, and A. Franchi, "Modeling, control and design optimization for a fully-actuated hexarotor aerial vehicle with tilted propellers," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 4006–4013.

[15] M. Ryll, D. Bicego, and A. Franchi, "Modeling and control of FAST-Hex: A fully-actuated by synchronized-tilting hexarotor," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2016, pp. 1689–1694.

[16] A. Franchi, R. Carli, D. Bicego, and M. Ryll, "Full-pose tracking control for aerial robotic systems with laterally bounded input force," *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 534–541, 2018.

[17] D. Brescianini and R. D'Andrea, "Design, modeling and control of an omni-directional aerial vehicle," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2016, pp. 3261–3266.

[18] M. Kamel, S. Verling, O. Elkhatib, C. Sprecher, P. Wulkop, Z. Taylor, R. Siegwart, and I. Gilitschenski, "Voliro: An omnidirectional hexacopter with tiltable rotors," *arXiv preprint arXiv:1801.04581*, 2018.

[19] S. A. Snell, D. F. Enns, and W. L. Garrard Jr, "Nonlinear inversion flight control for a supermaneuverable aircraft," *Journal of Guidance, Control, and Dynamics*, vol. 15, no. 4, pp. 976–984, 1992.

[20] K. P. Valavanis and G. J. Vachtsevanos, *Handbook of Unmanned Aerial Vehicles*. Springer Publishing Company, Incorporated, 2014.

[21] H. Alwi and C. Edwards, "Fault tolerant control of an octorotor using LPV based sliding mode control allocation," in *American Control Conference (ACC)*, 2013, pp. 6505–6510.

[22] G. P. Falconí and F. Holzapfel, "Adaptive fault tolerant control allocation for a hexacopter system," in *American Control Conference (ACC)*, IEEE, 2016, pp. 6760–6766.

[23] A. Lanzon, A. Freddi, and S. Longhi, "Flight control of a quadrotor vehicle subsequent to a rotor failure," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 2, pp. 580–591, 2014.

[24] M. W. Mueller and R. D'Andrea, "Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2014, pp. 45–52.

[25] G.-X. Du, Q. Quan, and K.-Y. Cai, "Controllability analysis and degraded control for a class of hexacopters subject to rotor failures," *Journal of Intelligent & Robotic Systems*, vol. 78, no. 1, pp. 143–157, 2015.

[26] J. Lee, H. S. Choi, and H. Shim, "Fault tolerant control of hexacopter for actuator faults using time delay control method," *International Journal of Aeronautical and Space Sciences*, vol. 17, pp. 54–63, 2016.

[27] W. Zhang, M. W. Mueller, and R. D'Andrea, "A controllable flying vehicle with a single moving part," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2016, pp. 3275–3281.

[28] G. Michieletto, M. Ryll, and A. Franchi, "Control of statically hoverable multi-rotor aerial vehicles and application to rotor-failure robustness for hexarotors," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 2747–2752.

[29] J. I. Giribet, R. S. Sanchez-Pena, and A. S. Ghersin, "Analysis and design of a tilted rotor hexacopter for fault tolerance," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 4, pp. 1555–1567, 2016.

[30] M. Achtelik, K.-M. Doth, D. Gurdan, and J. Stumpf, "Design of a multi rotor MAV with regard to efficiency, dynamics and redundancy," in *AIAA Guidance, Navigation, and Control Conference*, 2012, pp. 4779–4795.

[31] B. Crowther, A. Lanzon, M. Maya-Gonzalez, and D. Langkamp, "Kinematic analysis and control design for a nonplanar multirotor vehicle," *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 4, pp. 1157–1171, 2011.

[32] E. Kaufman, K. Caldwell, D. Lee, and T. Lee, "Design and development of a free-floating hexrotor UAV for 6-DOF maneuvers," in *IEEE Aerospace Conference*, 2014, pp. 1–10.

[33] M. Ryll, H. H. Bülthoff, and P. R. Giordano, "A novel overactuated quadrotor unmanned aerial vehicle: Modeling, control, and experimental validation," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 2, pp. 540–556, 2015.

[34] H. Efraim, A. Shapiro, and G. Weiss, "Quadrotor with a dihedral angle: On the effects of tilting the rotors inwards," *Journal of Intelligent & Robotic Systems*, vol. 80, no. 2, pp. 313–324, 2015.

[35] J.-J. E. Slotine, W. Li, *et al.*, *Applied Nonlinear Control*, 1. Prentice hall Englewood Cliffs, NJ, 1991, vol. 199.

[36] J. A. Farrell and M. M. Polycarpou, *Adaptive Approximation Based Control: Unifying Neural, Fuzzy and Traditional Adaptive Approximation Approaches*. John Wiley & Sons, 2006, vol. 48.

[37] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4077–4087.

[38] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, *et al.*, “Overcoming catastrophic forgetting in neural networks,” *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[39] F. Zenke, B. Poole, and S. Ganguli, “Continual learning through synaptic intelligence,” in *International Conference on Machine Learning (ICML)*, PMLR, 2017, pp. 3987–3995.

[40] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, “Meta-learning with memory-augmented neural networks,” in *International Conference on Machine Learning (ICML)*, PMLR, 2016, pp. 1842–1850.

[41] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in *International Conference on Machine Learning (ICML)*, PMLR, 2017, pp. 1126–1135.

[42] D. Pucci, “Towards a unified approach for the control of aerial vehicles,” Ph.D. dissertation, INRIA Sophia Antipolis, France, 2013.

[43] E. Bulka and M. Nahon, “A universal controller for unmanned aerial vehicles,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 4171–4176.

[44] X. Shi, K. Kim, S. Rahili, and S.-J. Chung, “Nonlinear control of autonomous flying cars with wings and distributed electric propulsion,” in *IEEE Conference on Decision and Control (CDC)*, 2018, pp. 5326–5333. DOI: 10.1109/CDC.2018.8619578,

[45] T. A. Johansen, A. Cristofaro, K. Sørensen, J. M. Hansen, and T. I. Fossen, “On estimation of wind velocity, angle-of-attack and sideslip angle of small UAVs using standard sensors,” in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2015, pp. 510–519.

[46] P. Tian and H. Chao, “Model aided estimation of angle of attack, sideslip angle, and 3D wind without flow angle measurements,” in *2018 AIAA Guidance, Navigation, and Control Conference*, 2018, p. 1844.

[47] J. Farrell, M. Sharma, and M. Polycarpou, “Backstepping-based flight control with adaptive function approximation,” *Journal of Guidance, Control, and Dynamics*, vol. 28, no. 6, pp. 1089–1102, 2005.

[48] F. Gavilan, R. Vazquez, and J. Á. Acosta, “Adaptive control for aircraft longitudinal dynamics with thrust saturation,” *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 4, pp. 651–661, 2014.

[49] A. J. Calise and R. T. Rysdyk, “Nonlinear adaptive flight control using neural networks,” *IEEE Control Systems Magazine*, vol. 18, no. 6, pp. 14–25, 1998.

[50] T. Lee and Y. Kim, “Nonlinear adaptive flight control using backstepping and neural networks controller,” *Journal of Guidance, Control, and Dynamics*, vol. 24, no. 4, pp. 675–682, 2001.

[51] E. Tal and S. Karaman, "Accurate tracking of aggressive quadrotor trajectories using incremental nonlinear dynamic inversion and differential flatness," in *IEEE Conference on Decision and Control (CDC)*, 2018, pp. 4282–4288.

[52] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 620–626, 2017.

[53] J. Nakanishi, J. A. Farrell, and S. Schaal, "A locally weighted learning composite adaptive controller with structure adaptation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, vol. 1, 2002, pp. 882–889.

[54] M. Bisheban and T. Lee, "Geometric adaptive control with neural networks for a quadrotor UAV in wind fields," *arXiv preprint arXiv:1903.02091*, 2019.

[55] K. Nonaka and H. Sugizaki, "Integral sliding mode altitude control for a small model helicopter with ground effect compensation," in *American Control Conference (ACC)*, IEEE, 2011, pp. 202–207.

[56] L. Danjun, Z. Yan, S. Zongying, and L. Geng, "Autonomous landing of quadrotor based on ground effect modelling," in *Chinese Control Conference (CCC)*, IEEE, 2015, pp. 5647–5652.

[57] F. Berkenkamp, A. P. Schoellig, and A. Krause, "Safe controller optimization for quadrotors with Gaussian processes," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2016, pp. 491–496.

[58] P. Abbeel, A. Coates, and A. Y. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1608–1639, 2010.

[59] A. Punjani and P. Abbeel, "Deep learning helicopter dynamics models," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 3223–3230.

[60] S. Bansal, A. K. Akametalu, F. J. Jiang, F. Laine, and C. J. Tomlin, "Learning quadrotor dynamics using neural network for flight control," in *IEEE Conference on Decision and Control (CDC)*, 2016, pp. 4653–4660.

[61] Q. Li, J. Qian, Z. Zhu, X. Bao, M. K. Helwa, and A. P. Schoellig, "Deep neural networks for improved, impromptu trajectory tracking of quadrotors," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 5183–5189.

[62] S. Zhou, M. K. Helwa, and A. P. Schoellig, "Design of deep neural networks as add-on blocks for improving impromptu trajectory tracking," in *IEEE Conference on Decision and Control (CDC)*, 2017, pp. 5201–5207.

[63] C. Sánchez-Sánchez and D. Izzo, “Real-time optimal control via deep neural networks: Study on landing problems,” *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 5, pp. 1122–1135, 2018.

[64] S. Balakrishnan and R. Weil, “Neurocontrol: A literature survey,” *Mathematical and Computer Modelling*, vol. 23, no. 1-2, pp. 101–117, 1996.

[65] M. T. Frye and R. S. Provence, “Direct inverse control using an artificial neural network for the autonomous hover of a helicopter,” in *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, 2014, pp. 4121–4122.

[66] H. Suprijono and B. Kusumoputro, “Direct inverse control based on neural network for unmanned small helicopter attitude and altitude control,” *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 9, no. 2-2, pp. 99–102, 2017.

[67] F. Berkenkamp, M. Turchetta, A. P. Schoellig, and A. Krause, “Safe model-based reinforcement learning with stability guarantees,” *Advances in Neural Information Processing Systems 30*, vol. 2, pp. 909–919, 2018.

[68] N. Levine, T. Zahavy, D. J. Mankowitz, A. Tamar, and S. Mannor, “Shallow updates for deep reinforcement learning,” in *Advances in Neural Information Processing Systems*, 2017, pp. 3135–3145.

[69] K. Azizzadenesheli, E. Brunskill, and A. Anandkumar, “Efficient exploration through bayesian deep q-networks,” in *2018 Information Theory and Applications Workshop (ITA)*, IEEE, 2018, pp. 1–9.

[70] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, “Spectral normalization for generative adversarial networks,” *arXiv preprint arXiv:1802.05957*, 2018.

[71] Y. Z. Tsypkin, “The systems with delayed feedback,” *Avtomathika i Telemech*, vol. 7, pp. 107–129, 1946.

[72] O. J. Smith, “A controller to overcome dead time,” *ISA Journal*, vol. 6, pp. 28–33, 1959.

[73] J.-P. Richard, “Time-delay systems: An overview of some recent advances and open problems,” *Automatica*, vol. 39, no. 10, pp. 1667–1694, 2003.

[74] M. Krstic, *Delay Compensation for Nonlinear, Adaptive, and PDE Systems*. Springer, 2009.

[75] H. Gao, T. Chen, and J. Lam, “A new delay system approach to network-based control,” *Automatica*, vol. 44, no. 1, pp. 39–52, 2008.

[76] R. A. Gupta and M.-Y. Chow, “Networked control system: Overview and research trends,” *IEEE Transactions on Industrial Electronics*, vol. 57, no. 7, pp. 2527–2535, 2009.

[77] P. Cortes, J. Rodriguez, C. Silva, and A. Flores, "Delay compensation in model predictive current control of a three-phase inverter," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 2, pp. 1323–1325, 2011.

[78] M. Lu, X. Wang, P. C. Loh, F. Blaabjerg, and T. Dragicevic, "Graphical evaluation of time-delay compensation techniques for digitally controlled converters," *IEEE Transactions on Power Electronics*, vol. 33, no. 3, pp. 2601–2614, 2017.

[79] E. Schuitema, L. Buşoniu, R. Babuška, and P. Jonker, "Control delay in reinforcement learning for real-time dynamic systems: A memoryless approach," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 3226–3231.

[80] A. Visioli, *Practical PID Control*. Springer Science & Business Media, 2006.

[81] P. K. Padhy and S. Majhi, "Relay based PI–PD design for stable and unstable FOPDT processes," *Computers & Chemical Engineering*, vol. 30, no. 5, pp. 790–796, 2006.

[82] S. Majhi and D. Atherton, "Online tuning of controllers for an unstable FOPDT process," *IEE Proceedings-Control Theory and Applications*, vol. 147, no. 4, pp. 421–427, 2000.

[83] V. L. Kharitonov and A. P. Zhabko, "Lyapunov–Krasovskii approach to the robust stability analysis of time-delay systems," *Automatica*, vol. 39, no. 1, pp. 15–20, 2003.

[84] F. Mazenc, S.-I. Niculescu, and M. Krstic, "Lyapunov–Krasovskii functionals and application to input delay compensation for linear time-invariant systems," *Automatica*, vol. 48, no. 7, pp. 1317–1323, 2012.

[85] M. A. Henson and D. E. Seborg, "Time delay compensation for nonlinear processes," *Industrial & Engineering Chemistry Research*, vol. 33, no. 6, pp. 1493–1500, 1994.

[86] Y.-H. Roh and J.-H. Oh, "Robust stabilization of uncertain input-delay systems by sliding mode control with delay compensation," *Automatica*, vol. 35, no. 11, pp. 1861–1865, 1999.

[87] D. Bresch-Pietri and M. Krstic, "Adaptive trajectory tracking despite unknown input delay and plant parameters," *Automatica*, vol. 45, no. 9, pp. 2074–2081, 2009.

[88] M. Krstic, "Input delay compensation for forward complete and strict-feedforward nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 55, no. 2, pp. 287–303, 2009.

[89] G. J. Leishman, *Principles of Helicopter Aerodynamics*. Cambridge University Press, 2006.

[90] B. W. McCormick, *Aerodynamics, Aeronautics, and Flight Mechanics*, 2nd ed. Wiley New York, 1995.

[91] L. W. Traub, "Range and endurance estimates for battery-powered aircraft," *Journal of Aircraft*, vol. 48, no. 2, pp. 703–707, 2011.

[92] B. Etkin, *Dynamics of Atmospheric Flight*. Courier Corporation, 2012.

[93] C. Rumsey and V. Vatsa, "A comparison of the predictive capabilities of several turbulence models using upwind and central-difference computer codes," in *31st Aerospace Sciences Meeting*, 1993, p. 192.

[94] R. E. Sheldahl and P. C. Klimas, "Aerodynamic characteristics of seven symmetrical airfoil sections through 180-degree angle of attack for use in aerodynamic analysis of vertical axis wind turbines," Sandia National Labs., Albuquerque, NM (USA), Tech. Rep., 1981.

[95] H. Khalil, *Nonlinear Systems*. Prentice Hall, 2002.

[96] T. Lee, "Exponential stability of an attitude tracking control system on SO(3) for large-angle rotational maneuvers," *Systems & Control Letters*, vol. 61, no. 1, pp. 231–237, 2012.

[97] S. W. Shepperd, "Quaternion from rotation matrix," *Journal of Guidance and Control*, vol. 1, no. 3, pp. 223–224, 1978.

[98] A. R. Klumpp, "Singularity-free extraction of a quaternion from a direction-cosine matrix," *Journal of Spacecraft and Rockets*, vol. 13, no. 12, pp. 754–755, 1976.

[99] S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Nonlinear attitude control of spacecraft with a large captured object," *Journal of Guidance, Control, and Dynamics*, vol. 39, no. 4, pp. 754–769, 2016.

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

[101] J. Hauser and R. Hindman, "Aggressive flight maneuvers," in *IEEE Conference on Decision and Control*, vol. 5, 1997, pp. 4186–4191.

[102] T. Lee, M. Leok, and N. H. McClamroch, "Nonlinear robust tracking control of a quadrotor UAV on SE (3)," *Asian Journal of Control*, vol. 15, no. 2, pp. 391–408, 2013.

[103] K. Shoemake, "Animating rotation with quaternion curves," in *ACM SIGGRAPH Computer Graphics*, ACM, vol. 19, 1985, pp. 245–254.

[104] W. C. Durham, "Constrained control allocation," *Journal of Guidance, Control, and Dynamics*, vol. 16, no. 4, pp. 717–725, 1993.

[105] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[106] K. A. Bordignon, "Constrained control allocation for systems with redundant control effectors," Ph.D. dissertation, Virginia Tech, 1996.

[107] L. Meier, P. Tanskanen, L. Heng, G. H. Lee, F. Fraundorfer, and M. Pollefeys, "Pixhawk: A micro aerial vehicle design for autonomous flight using onboard computer vision," *Autonomous Robots*, vol. 33, no. 1-2, pp. 21–39, 2012.

[108] X. Shi, P. Spieler, E. Tang, E.-S. Lupu, P. Tokumaru, and S.-J. Chung, "Adaptive nonlinear control of fixed-wing VTOL with airflow vector sensing," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 5321–5327. DOI: `10.1109/ICRA40945.2020.9197344`,

[109] J.-J. E. Slotine and W. Li, "Composite adaptive control of robot manipulators," *Automatica*, vol. 25, no. 4, pp. 509–519, 1989. DOI: `10.1016/0005-1098(89)90094-0`.

[110] S. Popowski and W. Dabrowski, "Measurement and estimation of the angle of attack and the angle of sideslip," *Aviation*, vol. 19, no. 1, pp. 19–24, 2015.

[111] *Preliminary datasheet sdp33*, SDP33, Version 0.1, Sensirion, Aug. 2017.

[112] K. Kim, S. Rahili, X. Shi, S.-J. Chung, and M. Gharib, "Controllability and design of unmanned multirotor aircraft robust to rotor failure," in *AIAA Scitech Forum*. 2019. DOI: `10.2514/6.2019-1787`. [Online]. Available: `https://arc.aiaa.org/doi/abs/10.2514/6.2019-1787`,

[113] R. E. Kalman, "Contributions to the theory of optimal control," *Boletin de la Sociedad Matematica Mexicana*, vol. 5, no. 2, pp. 102–119, 1960.

[114] R. Hermann and A. Krener, "Nonlinear controllability and observability," *IEEE Transactions on Automatic Control*, vol. 22, no. 5, pp. 728–740, 1977.

[115] E. B. Lee and L. Markus, *Foundations of Optimal Control Theory*. John Wiley & Sons, 1967.

[116] G.-X. Du, Q. Quan, B. Yang, and K.-Y. Cai, "Controllability analysis for multirotor helicopter rotor degradation and failure," *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 5, pp. 978–985, 2015.

[117] G. Shi, X. Shi, M. O'Connell, R. Yu, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S.-J. Chung, "Neural lander: Stable drone landing control using learned dynamics," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 9784–9790. DOI: `10.1109/ICRA.2019.8794351`,

[118] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[119] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 901–909.

[120] P. L. Bartlett, D. J. Foster, and M. J. Telgarsky, “Spectrally-normalized margin bounds for neural networks,” in *Advances in Neural Information Processing Systems*, 2017, pp. 6240–6249.

[121] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, “Understanding deep learning requires rethinking generalization,” *arXiv preprint arXiv:1611.03530*, 2016.

[122] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2016, pp. 770–778.

[123] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro, “A Pac-Bayesian approach to spectrally-normalized margin bounds for neural networks,” *arXiv preprint arXiv:1707.09564*, 2017.

[124] G. K. Dziugaite and D. M. Roy, “Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data,” *arXiv preprint arXiv:1703.11008*, 2017.

[125] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro, “Exploring generalization in deep learning,” in *Advances in Neural Information Processing Systems*, 2017, pp. 5947–5956.

[126] S.-J. Chung, S. Bandyopadhyay, I. Chang, and F. Y. Hadaegh, “Phase synchronization control of complex networks of Lagrangian systems on adaptive digraphs,” *Automatica*, vol. 49, no. 5, pp. 1148–1161, 2013.

[127] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in PyTorch,” 2017.

[128] I. Cheeseman and W. Bennett, “The effect of ground on a helicopter rotor in forward flight,” *Aeronautical Research Council Reports & Memoranda*, 1955.

[129] N. Hovakimyan, E. Lavretsky, and A. Sasane, “Dynamic inversion for nonaffine-in-control systems via time-scale separation. part i,” *Journal of Dynamical and Control Systems*, vol. 13, no. 4, pp. 451–465, 2007.

[130] W. Lohmiller and J.-J. E. Slotine, “On contraction analysis for non-linear systems,” *Automatica*, vol. 34, no. 6, pp. 683–696, 1998.

[131] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” *arXiv preprint arXiv:1509.02971*, 2015.

[132] M. Faessler, D. Falanga, and D. Scaramuzza, “Thrust mixing, saturation, and body-rate control for accurate aggressive quadrotor flight,” *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 476–482, 2016.

[133] Y. Chen and N. O. Pérez-Arancibia, "Adaptive control of aerobatic quadrotor maneuvers in the presence of propeller-aerodynamic-coefficient and torque-latency time-variations," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 6447–6453.

[134] P. Tabuada, "Event-triggered real-time scheduling of stabilizing control tasks," *IEEE Transactions on Automatic Control*, vol. 52, no. 9, pp. 1680–1685, 2007.

[135] M. Mazo and P. Tabuada, "Input-to-state stability of self-triggered control systems," in *48th IEEE Conference on Decision and Control (CDC) held jointly with 28th Chinese Control Conference*, IEEE, 2009, pp. 928–933.

[136] D. Theodosis and D. V. Dimarogonas, "Self-triggered control under actuator delays," in *IEEE Conference on Decision and Control*, 2018, pp. 1524–1529.

[137] X. Shi, M. O'Connell, and S.-J. Chung, "Numerical predictive control for delay compensation," *arXiv preprint arXiv:2009.14450*, 2020,

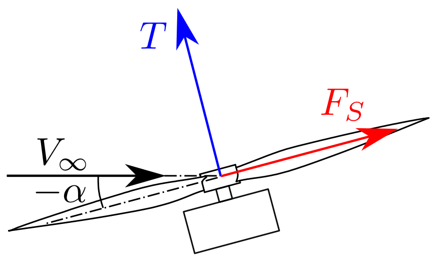
[138] K. Atkinson, W. Han, and D. E. Stewart, *Numerical Solution of Ordinary Differential Equations*. John Wiley & Sons, 2011, vol. 108.

Appendix A

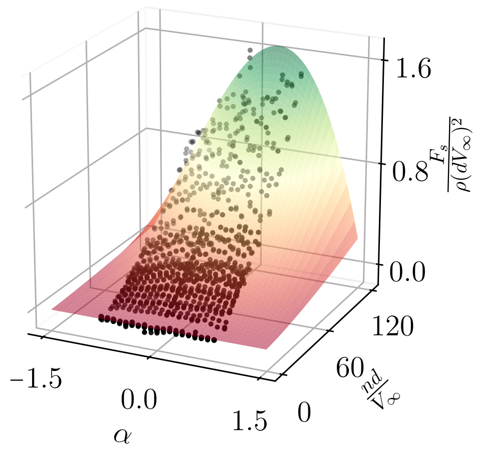
ROTOR SIDE FORCE MODEL

Although rotor thrust T is well studied and has a simple model that works well with feedback controllers [90], rotor side force F_S lacks its counterpart. Measurements from wind tunnel and flight tests indicate that F_S produced by rotors often incurs a significant drag increase during flight. Hence, we cannot ignore its contribution in order to have high accuracy position tracking. We use dimensional analysis to derive a canonical model that is consistent with wind tunnel test data:

$$F_S = C_S \rho n^{k_1} V_\infty^{2-k_1} d^{2+k_1} \left(\left(\frac{\pi}{2}\right)^2 - \alpha^2 \right) (\alpha + k_2)\,. \tag{A.1}$$



(a) Illustrative diagram



(b) Test data with numerical fit

Figure A.1: Rotor side force F_S illustration with non-dimensionalized data.

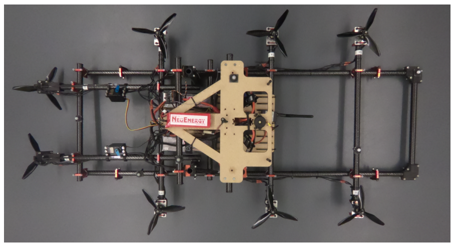
To construct an empirical model for rotor side force F_S, force data was collected for a propeller in forward flight at various angles of attack ranging from $-45°$ to $45°$ as shown in Figure A.1b. We assume that the side force depends on the freestream velocity V_∞, the air density ρ, the propeller's angular velocity n, and the propeller diameter d. Additionally, we expect some variation with α, which is a dimensionless quantity. Per symmetry, there should be no side force at angles of attack of $\pm 90°$. Data also indicates that F_S achieves a maximum at an angle other than $0°$. Hence, we use a third order polynomial of α with zeros at $\pm 90°$ to construct the model. These criteria combine to form the expression in (A.1).

Appendix B

EXPERIMENTAL FIXED-WING VTOL PROTOTYPES

B.1 Caltech Autonomous Flying Ambulance (AFA)

At Caltech's Center for Autonomous Systems and Technologies, research and development have been conducted on the feasibility of a fixed-wing VTOL aircraft for UAM. The project is named autonomous flying ambulance (AFA), and focuses on introducing an autonomous eVTOL that can transport patients with urgent medical needs in urban areas. Two prototypes have been developed for control system and aerodynamics research.


(a) Top view of AFA 1.0.

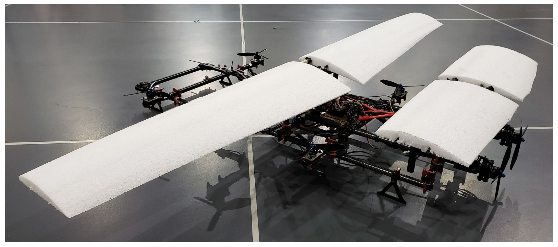

(b) AFA 1.0 with add-on wing and tail surfaces.

Figure B.1: AFA 1.0 prototypes.

Table B.1: AFA 1.0 rotor configuration parameters

Location [cm]								Tilt [°]	
x_1	x_2	x_3	x_4	y_1	y_2	z_1	z_2	θ_1^*	θ_2^*
44	18	−14	−46	25	10	0	7	19°	13°

Table B.2: AFA 1.0 rotor properties

C_T	C_P	d	n_{max}	T_{max}
0.1753	0.0911	15.24 cm	24 600 RPM	19.47 N

The first prototype AFA 1.0 is a 1/5 scale vehicle that combines multirotor and fixed wing surface. There will be two main flight modes for the aircraft: a VTOL mode for multirotor slow speed flight and a fixed-wing mode for high speed cruise flight. A bare-bone prototype of the aircraft is shown in Figure B.1a. Specifically, the prototype has a total of eight rotors around its main body, which are placed symmetrically about the body's longitudinal axis and are driven by electric brushless DC motors. Among them, six rotors are placed on the sides of the main body, all at the same height and equidistant from the main body to minimize aerodynamic drag when in cruise mode. The last two are located in the back, and the exact locations and orientation of the rotors are shown in Table B.1. Table B.2 shows measured rotor properties from bench tests.

The prototype is 112 cm × 56 cm × 16 cm in size with a mass of 3.65 kg. As shown in Figure B.1b, it can also incorporate an add-on wing surface with a span of 160 cm and a pair of horizontal tail. For fast forward flight, two servo motors are added to the two rear rotors to achieve thrust vectoring through the tilting of the entire drive mechanism. This will allow the prototype to gain forward acceleration without pitching the main body too much until it flies fast enough for the wing to be effective.

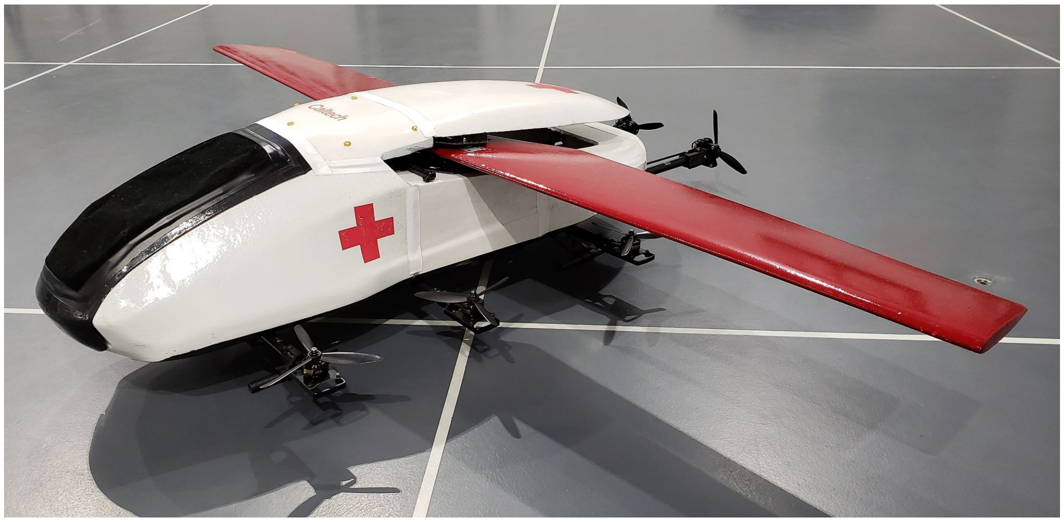


Figure B.2: Caltech's 1/5 scale AFA 2.0 fixed-wing VTOL with tiltable rear rotors.

Table B.3: AFA 2.0 physical parameters

m [kg]	J_{xx}	J_{yy}	J_{zz}	J_{xz}	[kg m^2]
5.5	0.134	0.252	0.346	−0.004	
C_T	C_Q	C_{L_0}	C_{L_1} [rad^{-1}]	C_{D_0}	k_{C_L}
0.182	0.0143	0.216	1.622	0.0651	0.273

AFA 2.0 shown in Appendix B.1 is built upon the same specs of the previous version. It has the same rotor placement, thrust vectoring, and wing size. The main improvements are the elimination of the rear tail surface and the inclusion of a

streamlined fuselage for reduced drag. The physical properties are listed in Table B.3, with $J_{xy} \approx J_{yz} \approx 0$ due to symmetry. C_T and C_Q are experimentally obtained for a six-inch propeller. The aerodynamic coefficients are calculated through combining both wind-tunnel test data as well as theoretical values of a finite wing and blunt body.

B.2 Experimental Prototype for Adaptive Flight Control

A copter-plane configuration fixed-wing VTOL prototype is designed and built for adaptive control research discussed in Chapter 4. The plane is shown in Figure B.3. It has four vertical lift rotors on wingtips and a forward cruise motor mounted in the front. The main wing is roughly 1 m in span, and a pair of all moving vee-tail is responsible for pitch and yaw control during fixed-wing flight.

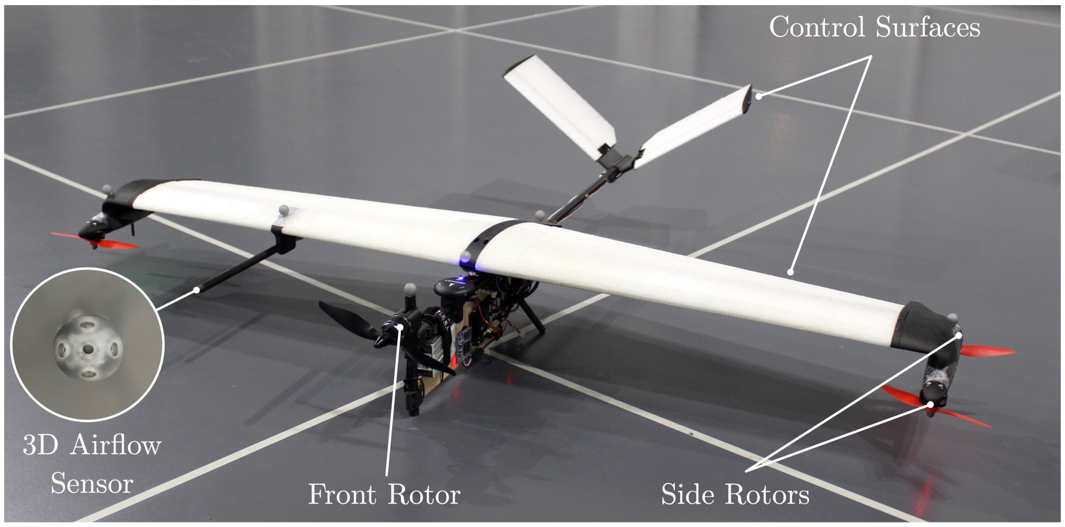


Figure B.3: Fixed-wing VTOL platform with wing and multi-rotor

Table B.4: Fixed-wing VTOL platform physical properties

Weight	1.70 kg	Lift Motor	T-motor F80-2200
Wingspan	1.08 m	Front Motor	T-motor AT2312-1150
Wing Area	0.223 m^2	Lift Prop.	King Kong 6"x4"
Battery	6S 1.3 Ah	Front Prop.	APC 7"x4"

The physical properties of the aircraft are summarized in Table B.4. For flight control computation, it incorporates a Pixhawk flight controller with PX4 firmware [107] and a small single board computer (SBC). The SBC is included to handle some high level trajectory planning and input-output tasks. In addition, the novel 3D airflow sensor described in Section 4.3 is mounted on the main wing. With all the hardware, the platform achieves high accuracy position control in varying wind conditions.

Table B.5: Fixed-wing VTOL platform aerodynamic parameters

Thruster	mean ± std	Aero	mean ± std
C_{Tx}	3.02e−3 ± 2.25e−5	C_{L_0}	0.3705 ± 0.06523
C_{Tz}	2.87e−3 ± 5.00e−6	C_{L_1}	3.2502 ± 0.04434
C_S	2.31e−5 ± 1.18e−5	C_{D_0}	0.1551 ± 0.00394
k_1	1.425 ± 0.010	C_{D_1}	0.1782 ± 0.06518
k_2	3.126 ± 0.087	C_{D_2}	1.6000 ± 0.10820

To accurately model the aerodynamic characteristics, the aircraft was mounted on a force sensor with all control surfaces in a neutral position. The data collected was for free-stream velocities up to 9 m/s, and parameters were fit using Bayesian linear regression for the model described in (4.4) and (4.5). Similarly, single motor and propeller combinations were tested to fit the model from (3.4) and (4.3). The results of the numerical fits are summarized in Table B.5, split by thruster parameters and aerodynamic parameters. The standard deviation is from the posterior distribution of parameters after performing Bayesian linear regression using the Markov Chain Monte-Carlo (MCMC) method.

B.3 Intel Aero Drone Testbed for Learning-based Control

To evaluate the performance of a DNN-based controller, we use an off-the-shelf commercial quadrotor drone that has the required computation and sensing capabilities.

The Intel Aero Drone shown in Figure B.4 weighs 1.47 kg with rotors of 23 cm diameters. The thrust coefficient C_T is measured to be 0.089 from the bench test. It includes an onboard Linux computer (2.56 GHz Intel Atom x7 processor, 4 GB DDR3 RAM) as well as a flight controller in parallel running PX4 firmware [107]. We retrofitted the drone with eight reflective infrared markers for accurate position, attitude, and velocity estimation at 100Hz.

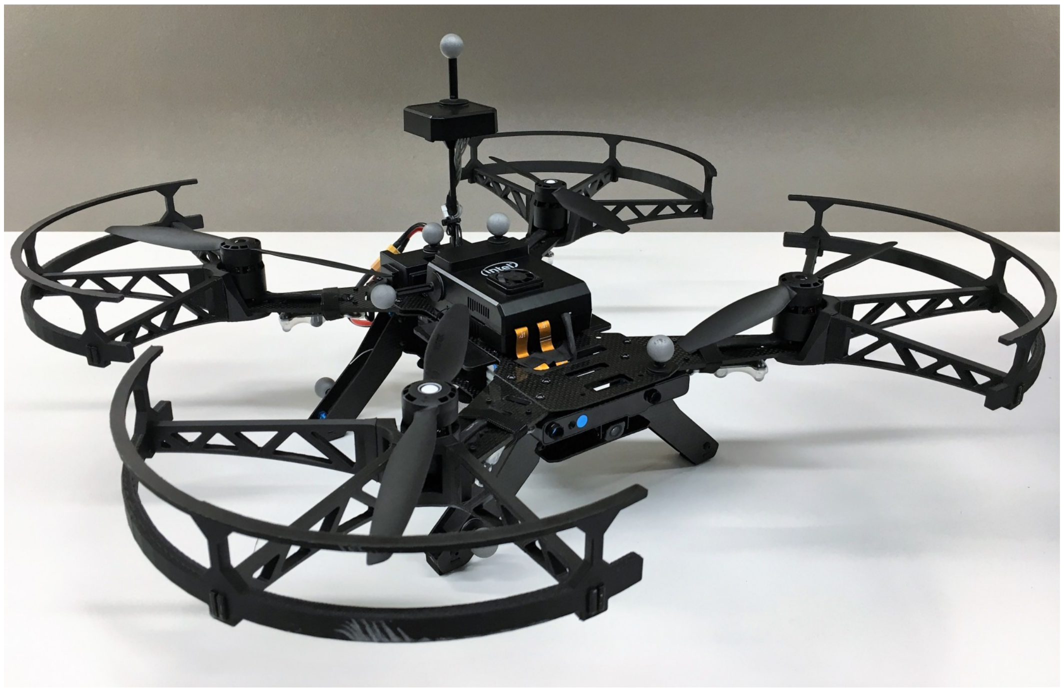


Figure B.4: Intel Aero Drone quadrotor platform