Nonlinear Control of Aerial Manipulators

by

Zifei Jiang

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Control Systems

Department of Electrical and Computer Engineering
University of Alberta

Abstract

With advancements in the autonomy and capabilities of Unmanned Aerial Vehicles (UAVs), their potential for complex operations such as infrastructure inspection, maintenance, and transportation has significantly increased. This thesis delves into innovative methods to enhance UAV autonomy, focusing on the motion control and force control of specialized UAV configurations. The study primarily revolves around the dynamics and control of slung load systems, the implementation of quasi-static feedback linearizing control, immersion and invariance adaptive control for fully-actuated aerial manipulation, and the application of reinforcement learning for improved UAV performance.

One key area of this research is the intricate dynamics associated with slung load systems in UAVs, which are integral for complex transportation and deployment operations. These systems demand precise control for managing the dynamics of their cargo. The thesis introduces a quasi-static feedback linearizing algorithm, universally applicable to differential flat systems. Based on this algorithm, quasi-static feedback controllers for a slung load system are developed, making the slung load system precisely linearized. The quasi-static feedback controller is adept at managing both the outer-loop and the full system dynamics of the slung load system. The controller's effectiveness, robustness, and accuracy in trajectory tracking are validated through extensive experiments in simulated and real-world scenarios, significantly enhancing UAVs' ability to handle heavy loads, a key aspect in diverse operational settings. Additionally, the research contributes to the development of a Maple-Matlab Software-In-The-Loop pipeline, streamlining the creation of nonlinear UAV controllers.

Another pivotal aspect of this thesis is the formulation and application of immersion and invariance adaptive control in fully-actuated aerial manipulators. Fully-actuated aerial manipulators, equipped with rigidly connected sticks, are becoming vital in tasks that necessitate direct environmental interaction, such as maintenance or disaster recovery. Integrating immersion and invariance adaptive control with these aerial manipulators addresses the complexities of interacting with unpredictable environments. The newly developed immersion and invariance adaptive hybrid force-motion controller demonstrates global asymptotically stable results, surpassing classical adaptive control theories by not requiring persistent excitation conditions or linear parameterization assumptions. This innovative control strategy ensures stable, robust, and efficient manipulation, unlocking new possibilities for sophisticated and autonomous aerial manipulators.

Furthermore, the thesis explores the application of reinforcement learning to enhance UAV autonomy. This segment focuses on developing learning algorithms that enable UAVs to adapt to diverse environments and tasks autonomously. The use of reinforcement learning allows UAVs to learn from their flight data, leading to continuous improvement in performance and adaptability. The effectiveness of these learning strategies is demonstrated through various simulations, showcasing their potential in advancing UAV capabilities.

In summary, this thesis presents a comprehensive study on advancing UAV technology through the control of slung load systems, the application of quasi-static feedback linearizing and immersion and invariance adaptive control for aerial manipulation, and the innovative use of reinforcement learning. These contributions mark a significant step forward in the field of UAV technology, expanding their applications in complex and dynamic environments.

Preface

- An earlier version of Section 3.1 and Section 3.2 was published as [Z. Jiang, M. A. Lawati, A. Mohammadhasani, and A. F. Lynch, “Flatness-based Motion Control of a UAV Slung Load System Using Quasi-static Feedback Linearization”, *in Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS)*, 2022], and [Z. Jiang and A. F. Lynch, “Quasi-static State Feedback Output Tracking for a Slung Load System”, *2023 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*]. I was responsible for the scientific content in the two papers and for writing the manuscripts. Professor Lynch acted as supervisor for this work and helped with concept formation and the manuscript writing.

- Section 3.1, 3.2 and 3.3 was published as [Z. Jiang Z , M. A. Lawati, and A. F. Lynch, “Quasi-Static State Feedback Output Tracking for a Slung Load System with Rotor Drag Compensation: PX4-SITL Validation.” *Journal of Intelligent & Robotic Systems*, vol. 109, no. 42, 2023]. I was responsible for theoretical derivation, experiments and manuscript composition. Professor Lynch supervised the work and helped with the manuscript writing.

- Section 4 has been accepted for publication as [Z. Jiang, A. F. Lynch, “Nonlinear Motion Control of a Multirotor Slung Load System: Experimental Results,” *2024 American Control Conference*]. I was responsible for theoretical derivation, software implementation, simulation, performance analysis and manuscript composition. Professor Lynch acted as supervisors and helped with concept formation and the manuscript writing.

- Section 5 is under preparation as [Z. Jiang, A. F. Lynch, “I&I Adaptive Hybrid Force-Motion Control of Fully Actuated Hexarotors,” *Automatica*], I was responsible for theoretical derivation, simulation and manuscript composition. Dr. Lynch provided a supervisory role and helped with the manuscript writing.

- Section 6 has been accepted for publication as [Z. Jiang, A. F. Lynch, “Quadrotor Motion Control Using Deep Reinforcement Learning,” *Journal of Unmanned Vehicle Systems*, vol. 9, no. 4, pp. 234- 201] and [Z. Jiang, A. F. Lynch, “Quadrotor Motion Control Using Deep Reinforcement Learning,” *Unmanned Systems Canada Annual Conference*, 8 pages, 2020], I was responsible for theoretical derivation, software implementation, simulation, performance analysis and manuscript composition. Professor Lynch acted as supervisors and helped with concept formation and the manuscript writing.

Contents

List of Tables

List of Figures

List of Acronyms

AM	Aerial Manipulator
ANCL	Applied Nonlinear Control Lab
CoM	Center of Mass
CPU	Central Processing Unit
DDPG	Deep Deterministic Policy Gradient
DoF	Degree of Freedom
DQN	Deep Q-Learning
EKF	Extended Kalman Filter
ESC	Electronic Speed Controller
FA AM	Fully Actuated Aerial Manipulatior
FMU	Flight Management Unit
FPV	First-Person View
FTDI	Future Technology Devices International Limited
GC	Geometric Control
GPS	Global Positioning System
I&I	Immersion and Invariance
I&M	Inspection and Maintainance
IMU	Inertial Measurement Unit
LTI	Linear Time-Invariant
MCU	Micro Control Unit
MDP	Markov Decision Process
MPC	Model Predictive Control
NNs	Neural Nets
ODE	Open Dynamics Engine
PE	Persistent Excitation

PPO	Proximal Policy Optimization
PWM	Pulse Width Modulation
PX4	Pixhawk Autopilot
QSF	Quasi-Static Feedback
QSFA	Quasi-Static Feedback Algorithm
RL	Reinforcement Learning
ROS	Robot Operating System
RPM	Revolutions Per Minute
SITL	Software-In-The-Loop
SLS	Slung Load System
TRPO	Trust Region Policy Optimization
UART	Universal Asynchronous Receiver-Transmitter
UAV	Unmanned Aerial Vehicle
UDT	Uni-directional Thrust

Chapter 1

Introduction

1.1 Background

The integration of robotics and artificial intelligence into our daily lives represents a significant paradigm shift in our interaction with technology. From the early stages of digital technology, robotic systems have evolved to extend and enhance human capabilities across a spectrum of tasks, whether complex, challenging, or routine. These mobile robots, operating on land, underwater, or in the air, have become indispensable tools, acting as our extended sensors and amplifiers of efficiency. Their deployment has expanded our reach to remote locations on Earth and beyond, as we imbue these systems with greater autonomy, enabling them to process sensory information and execute tasks without direct human oversight. Robots are increasingly working alongside or replacing humans in performing tasks such as the inspection of critical infrastructure, characterized by increased efficiency, precision, repeatability, lower cost, and reduced environmental impact. As humans and robots focus on tasks for which they are best suited, with humans excelling in tasks requiring higher cognitive abilities and complex decision-making, the quality of life improves. The current technology restricts so-called industrial robots, i.e., heavy, rigid, fixed-base manipulator arms, to structured environments, such as the automotive sector, where they are used for high-volume assembly line tasks. These arms follow preprogrammed trajectories in static, known environments and are not designed to deviate from their tasks or adapt to environment change. However, increasing pressures to improve sustainable economic growth require new capabilities in mobile and arm robotics. Over the next decades, robotics will undergo a transformation where machines leave the structured factory floor and operate cooperatively and safely in day-to-day unstructured environments alongside humans. This emerging class of service robots will work intelligently alongside humans, employing their sensors to carry out highly complex tasks in unanticipated situations.

Experts agree that dominant growth in robotics will be in service robots, as evidenced by the significant increase in the sale of professional service robots used in non-domestic natural environments to assist humans, such as in inspection and maintainance (I&M), package delivery and medical applications. The functioning of society depends on the integrity of infrastructure, which must remain operational 24/7 and can be geographically widespread over rugged terrain, aging, or deteriorating due to a severe climate. Regular I&M are often mandated but can be inefficient, costly, require downtime, and be dangerous when performed by manned crews.

In particular, rotary-wing unmanned aerial vehicle (UAV) have found diverse applications in commercial and recreational activities. Equipped with cameras and specialized sensors, UAVs are adept at surveilling areas for security monitoring and structural inspection. In the realm of aerial transport and delivery, they represent a burgeoning field, offering solutions to circumvent ground-level traffic congestion. Moreover, they have captured the imagination of the public, with amateur pilots navigating drones using first-person view (FPV) goggles along soaring mountain ridges, extending their vision and experiencing exhilaration without physical risk. As commercial UAVs' advance, the focus of aerial robotics research is shifting from tools for observation and navigation to more interactive roles involving aerial manipulation. This transition towards extending autonomous mobile manipulation into unrestricted workspaces has necessitated the development of fully actuated aerial robots equipped with novel control methods, crucial for refining strategies for aerial interaction and precise dynamic tracking of end effectors from a flying base.

The subsequent sections of the literature review delve into slung load system (SLS), fully actuated aerial manipulatior (FA AM), and the application of reinforcement learning (RL) to UAV motion control. The SLS, with its broad applications in load transportation, construction, and search and rescue, exemplifies the practical utility of these developments. The advancement of fully actuated and omnidirectional UAVs aims to enhance the capabilities of conventional underactuated flying systems, crucial for effective aerial manipulation. Aerial interaction with fully actuated UAVs emerges as a recent research focus, presenting numerous opportunities for advancing the control of aerial robots. Additionally, the application of RL to the control of UAVs has gained traction in recent years, with the potential to overcome the limitations of traditional control methods. UAVs have broader societal implications of robotic integration, where robots are increasingly performing tasks that are dull, dirty, or dangerous, thus improving efficiency, precision, and reducing environmental impacts. This is not limited to structured environments like factories but extends to unstructured settings where service robots operate cooperatively and safely alongside humans. The International Federation of Robotics notes a significant rise in the sale of professional service robots, particularly for I&M tasks , highlighting the growing reliance on robotics in maintainingt essential infrastructure.

1.2 Literature Review

1.2.1 Slung Load System

Recently, interest has increased in using drones for load transport. Developments are summarized in [1]. A popular method of transport is a multirotor drone SLS, in which a cable attached to the underside of the drone suspends the load. The ability to maneuver the system and maintain a safe distance between the payload and the vehicle are some of the obvious advantages of SLSs and has led to their use for obstacle avoidance applications [2]. Also, SLSs are lightweight and do not require powered computer control as with gripper-based solutions. Despite SLSs being a desirable option for load transportation, creating a high-performance and rigorous motion control system is difficult due to the underactuated nonlinear dynamics involved. Much of the research on SLS motion focuses on developing new feedback laws to ensure some form of tracking error stability is achieved. It should be also noted that motion planning or open-loop control for SLS is another important area of work, e.g., [2]–[4] optimize for obstacle avoidance. Both problems are important to achieve high-performance motion control. It is interesting to note that a single drone SLS can be generalized to a multiple drone SLS that can control the 6D pose of a rigid body payload, e.g., [1], [5]–[7]. Drone SLSs are part of a larger recent trend of "unmanned aerial manipulation" that focuses on aerial robots that interact with their environment. In the case of the SLS, the "environment" is the payload. See [1], [8], [9] for surveys on aerial manipulation. Figure 1.1 shows the SLS platform built and tested by the author in Applied Nonlinear Control Lab (ANCL).

The concept of differential flatness was first presented in [10] and a survey of results is in [11]. The so-called flat outputs differentially parameterize the input and state. In other words, the state and input can be expressed in terms of the time derivatives of the flat output. Open-loop motion planning frequently employs this parameterization. The nonlinear SLS model has been proven flat in [3] under common modelling assumptions. That paper uses flatness to perform motion planning and does not explicitly use flatness in the feedback design. Every flat system is known to be linearizable by endogenous dynamic feedback [10] that is a type of dynamic state feedback where controller state and auxiliary input can be expressed as a function of the system state, system input, and their time derivatives. Flatness is preserved under this form of feedback [12]. Any endogenous dynamic feedback controller can be transformed into a quasi-static feedback (QSF) [13]. The advantage of QSF is that it achieves exact linearization without appending controller states to the closed-loop, unlike dynamic state feedback. We exploit this static linearization

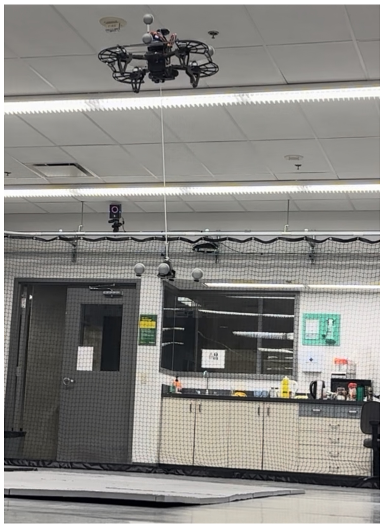


Figure 1.1: ANCL (University of Alberta) SLS Platform.

property in the proposed design to ensure implementation is possible on common autopilot hardware.

Recent theoretical work on flatness focuses on the open problem of deriving complete necessary and sufficient conditions which can be used to construct a flat output. Only partial results are available, and examples of recent work include [14] where the system dynamics is taken to be flat after one-fold prolongation. Two-input systems and endogenous dynamic feedback are considered in [15]. Another line of work relating to flatness is more practically oriented. For example, flatness is used for open-loop trajectory planning for drone swarms and wheeled mobile robots in [16], [17], single quadrotors [18], [19], and the SLS [2], [3]. For traditional quadrotors, flatness-based feedback control is developed in [20], [21]. This work provides convincing experimental results for aggressive trajectory tracking. Flatness-based control provides high performance with low computation compared to model predictive controllers (MPC) [22]. For SLSs, although flatness was used for open-loop trajectory planning [3], there is no previous work on using it for *feedback* control to achieve linear tracking error dynamics. Furthermore, the SLS was proven flat in [3] ignoring rotor drag and flapping, and our work [23] extends this result to prove flatness with these forces present and directly compensates them with the QSF. Since tracking error dynamics can be linearized, adopting flatness for closed-loop control has the advantage of a simple and immediate stability result. This should be compared to results such as [3], [24], [25] where stability analysis is complicated by nonlinear multi-loop dynamics. We further elaborate on this difference in more detail below.

Due to the practical and theoretical interest in SLS motion control, there have been a significant number of results on this topic. A standard nonlinear rigid-body multirotor drone model coupled with pendulum dynamics, which describes the suspended load, is the foundation of the majority of the work. Multi-loop designs are common where the coupled translational drone/pendulum dynamics are controlled at the outer loop. Using a reference from the outer loop, an inner loop regulates the rotational degree of freedom (DoF) of the drone. This approach is in [3], [25]–[27]. In particular, a 3-loop geometric control (GC) structure is proposed in [3], [27] with the innermost loop tracking drone attitude. Load attitude and drone yaw are controlled by the middle loop. The outer loop tracks load position. This work proved the entire tracking error dynamics is almost globally exponentially attractive. The nonlinear error dynamics achieve exponential stability in a difficult-to-describe local region dependent on controller gain.

The nature of the exponential stability result of the proposed QSF design is an important aspect of the thesis's contribution, and to show this we contrast our result with the GC [23]. A comparison with this work is logical, given that both methods are nonlinear and track payload position and UAV yaw. We show the GC is harder to tune and yields very complicated controller expressions that must be approximated for implementation. Even assuming exact control law expressions are available, the GC provides almost global exponential *attractiveness*, but only *local* exponential stability on some hard-to-compute region of attraction. In comparison, our result is exponential *stable* over a wide and precisely described subset of state space where control law singularities do not occur. Simulation results show the GC's tuning challenges and sensitivity to unmodelled drag and flapping force and constant disturbance.

Existing work on nonlinear control of SLS, which uses backstepping includes [24], [28]–[30]. This work inherently performs design subsystems-at-a-time. Work [30] uses a Lyapunov function-based motion control law for a bare multirotor without pendulum. Backstepping is then used to derive the SLS control law. Only local exponential stability is proven. Since the control law is specified in a general form, the complexity and conditions for stability of the particular multirotor control law used in simulation and experiments are difficult to determine. In [24], a backstepping control is derived by dividing the SLS into 4 subsystems involving (1) translational load variables, (2) load direction, (3) quadrotor attitude, and (4) quadrotor angular velocity. A complex local asymptotic stability proof is required that includes conditions on initial conditions and controller gain. A similar backstepping approach is taken in [29] that includes adaptation to unknown payload mass. As in [24], the stability result is local asymptotic, and the stability proof is relatively complicated. The closed-loop backstepping design in [24] is used in [2] where the main focus is aggressive open-loop trajectory generation. Here, flatness relations and constrained quadratic programming are used to minimize angular velocity actuation through the fifth-order time derivative of load position. Similarly, load acceleration is designed to satisfy load angle constraints. The convex optimization problem can be efficiently solved. Another example of backstepping is in [31] where motion control for a bare multirotor is used as the first step in the design. Independent of backstepping, bare multirotor motion controllers which are robust to external disturbance, e.g., [32], [33], could potentially be used for SLS control where the pendulum acts as a disturbance.

A standard nonlinear rigid-body multirotor drone model coupled with a single rigid pendulum, which describes the suspended load, is the foundation for most of the literature on nonlinear motion control. Our work models and compensates parasitic rotor drag. This force arises from the difference in air velocity seen by the advancing and retreating blades when the UAV has non-zero linear velocity [34]. The importance of compensating parasitic drag has been shown in recent work on motion control for bare (i.e., no slung load attached) multirotors [21]. We can classify the large body of work on feedback motion control as multi- or single-loop. Multi-loop approaches divide the design into two or more loops or subsystems. The most common is a two-loop structure where the coupled translational UAV/pendulum dynamics are controlled in an outer-loop. This outer-loop feeds the inner-loop a reference trajectory, and the inner-loop tracks the attitude error of the drone.

Examples of the multi-loop method are [3], [25]–[27], [35]. A nested control law is presented in [35] where air resistance of the payload is accounted for. Here three loops are designed: (1) an attitude control for the quadrotor, (2) a swing angle control, and (3) a linear velocity controller for the payload. A lengthy local exponential stability proof assumes zero desired load linear acceleration, which limits the method to stabilization of payload velocity. Hence, for general time-varying payload reference positions, tracking error of the payload position will not converge. This limitation is also found in related work by the same authors [36], [37].

Another nested control result is in [25]. Here, the outer loop is partially feedback linearized, whereas the inner loop is fully feedback linearized. The load angle is stabilized due to the exponential regulation of drone position error. The disadvantage of the approach is its inability to track time-varying load positions. Work [38] extends the geometric control in [3], [27] by considering a pendulum with offset. The approach proposes a 3-loop controller which requires small quadrotor acceleration.

Examples of single-loop methods are [39], [40]. Here, the entire SLS dynamics is compensated as a whole, translating to more complex control laws in general. However, single-loop has the benefit of a complete stability result, which accounts for interaction between subsystems often neglected in multi-loop designs.

Slung loads are simple, lightweight, passive, mechanical solutions for transporting a payload at a safe distance from the drone. The advantages of SLSs are that they are based on almost stock drone platforms, they keep the payload at a distance to reduce interference with thrust generation and provide improved safety during loading/unloading. High maneuverability of the payload is another important benefit which allows for package delivery in congested urban environments. The maneuverability of the payload's position (and full pose more generally for multi-drone SLS [7]) is due to the additional degrees of freedom provided by suspending the payload. However, reliably achieving complex maneuvers is challenging as it requires a rigorous compensation of nonlinear and underactuated dynamics. A conventional quadrotor has four inputs, and even a simplified rigid body SLS dynamics has eight degrees of freedom with nonlinear coupling between the drone and payload subsystems. Hence, it is unsurprising there is significant attention on developing nonlinear closed-loop motion control for the SLS in the literature.

1.2.2 Fully-Actuated Aerial Manipulator

I&M is dangerous, mundane, and resource intensive when directly performed by humans. Given their vast workspace, traditional UAVs have been proven useful for such applications [41]. They improve efficiency (e.g., lower cost, time, inaccuracy) by accessing difficult areas while removing humans from danger. Up until now UAVs have been limited to passive operation where they "see" but cannot "touch". However, I&M requires contact and manipulation to perform testing, cleaning, or sensor deployment. Hence, the second part of the thesis tackles the challenge of expanding the multirotor UAV's capability to an innovative new generation of so-called fully actuated aerial manipulatior (FA AM) which can "push" and "feel" their environment in a robust and accurate manner.

There are two typical design in AMs, fixed rotor design [42] and tiltrotor design [43]. A tiltrotor AM provides advantages over a fixed tilt AM for generating a range of force vectors while maintaining hover efficiency. However, despite these advantages, tiltrotor systems also come with drawbacks such as additional actuation mass and system complexity, finite arm rotation due to cable windup, and not encountered in a fixed rotor system. Therefore, morphology design is important to ensure that the resulting platform meets performance requirements. These requirements are defined by tasks, e.g., point inspection, rolling inspection, aerial drawing and grasping object.

There are general three design goals introduced [43]. Firstly, the system is full actuated in any hover orientation. This increase the versatility of the platforms to different and constraint environments. Secondly, the system is able to achieve high force and torque capabilities in all directions. In addition to fully actuation, increasing the max achievable force and torque is desired to perform tasks, e.g., heavy load transport or inspections that required high contact force and torque. Thirdly, high-efficiency hover in at least one orientation. The most obvious advantage of AMs over ground robot arm is the maneuverability, where some objects are only reachable by flying. However, the trade-off between capabilities and platform complexity needs to be addressed carefully in practical platform design. Work [44] optimized the ratio of the desired body force magnitude to the sum of the individual rotor group thrust magnitudes. Work [45] provided a comprehensive discussion on the design of FA AMs and provided advantages and disadvantages of each design.

For any kind of AM design, core aerial manipulation capability relies on a stable force/motion control which does not require knowledge of a contact model. Environment uncertainty makes force control difficult compared to an already challenging problem of free motion control in a structured space [46], [47]. New AM force controls with rigourous stability results and precise conditions for robustness to environmental uncertainty is essential to achieve force/motion control adaptive to unknown environments.

Force control for fixed-based arm robotics has enjoyed significant attention for about 5 decades [48]. Interestingly, most industrial robots do not provide force control and rely on structured environments or passive compliance between the robot and the environment. Two main frameworks for force control are: hybrid force/position (thereafter “hybrid control”) and impedance control [48], [49]. The AM-specific work is relatively sparse with the first papers appearing about 10 years ago [50]. Most existing work suffers from limiting assumptions about the environment.

Hybrid control uses knowledge of motion constraints to decompose the workspace into constrained and unconstrained directions. The method is favored in tasks where accurate force control is needed since it uses feedback of estimated or measured force (and position). In its ideal form, hybrid control assumptions include: the end-effector remains in contact, friction is negligible, geometry and pose of the environment is known, the robot model is ideal, and contact force is perfectly estimated or measured. To deal with these restrictive assumptions, hybrid control has been extended to robust and adaptive versions for fixed-based arms. For AMs, there has been little work which removes the above assumptions [50]–[63]. Work in [64] proposed a rudimentary hybrid control for a unit-direction-thrust· AM for point contact on a planar surface with known orientation. A more advanced FA AM platform uses hybrid force control for push-slide on planar surfaces [65]. However, these controllers rely on most of the abovementioned assumptions and effectively assume a known environment. An example of AM impedance control is in [42]. Force is indirectly regulated without the added complexity of contact detection and controller switching which is required in hybrid control. However, the geometry of the environment must be integrated into the impedance model and reference position trajectory. As well, as the method is indirect, force is not accurately controlled.

Recently, impedance and hybrid control are combined for a simple FA AM for point contact [56]. The virtual mass of an impedance control is varied along a single contact axis to provide compliance as required. This assumes the operator has aligned the AM to the surface normal and the distance to contact is measured. Impedance control provides some robustness to environment uncertainty by eliminating controller switching and contact detection, however, choosing a suitable impedance model implicitly assumes knowledge of the position and geometry of the surface. A direct force feedback is smoothly mixed with impedance control as a function of distance in the contact axis. Although initial tests are encouraging, a significant *drawback is the implicit assumption about the environment (i.e., the surface normal and position of surface is known, the vehicle model known, friction neglected)* which ultimately makes the approach difficult to use outside an academic lab.

The problem of designing AM force control robust to useful a range of environment uncertainty is clearly an open problem. Hence, our approach takes a hybrid force control framework which is adaptive to AM model uncertainty and environment geometry. Work [66] gave results on curved surface for fixed-base robots on push-slide tasks where the methods of nonlinear adaptive control can be applied assuming a general algebraic form for the surface. However, the adaptation law is based on least square method, which relies on persistent excitation (PE) condition and leads to complicated stability proof. As hyrbid methods can suffer from transition chattering, we adopt new methods of contact detection to switch between free and contact controllers [67].

The last decade has shown that AM vehicle design (or morphology) is critical and must match the manipulation task considered. Equally important is the accuracy and robustness of the force/motion control and perception algorithms used. Early work involves simple tasks and traditional uni-directional thrust (UDT) multirotors, sometimes with a rudimentary arm mounted to the vehicle [68]. As the vehicle is underactuated, even simple manipulation (e.g., point contact on a vertical wall with a sustained practical level of force) is difficult given the limited lateral forces UDT vehicles apply and that force is generated by rotating the vehicle. The latest generation of AMs is fully actuated (i.e., can apply any 3D force and torque). Since 6 DoFs are independently controlled, a wide range of wrench can be stably generated. Here, hexrotor fixed-tilt with non-collinear rotor axes [69], and multirotor variable-tilt [70] designs are common. Fixed-tilt designs are popular as they can be created from traditional off-the-shelf multirotors. However, energy efficiency is traded off with lateral force [71] since internal force not compensating gravity is required to generate arbitrary wrench on the environment. On the other hand, variable-tilt AM are difficult to engineer, heavier, and have finite tilt dynamics bandwidth, but create no waste in wrench generation. Our research focus on force control of fixed-tilt fully actuated platforms and derive a new adaptive controller with rigorous and clean stability proof based on so called immersion and invariance (I&I) adaptive control theory.

Our innovative adaptive hybrid control method for FA hexrotors builds upon the extensive development of adaptive control theory, which began in the early 1950s with the design of autopilots for high-performance aircraft. These aircraft required sophisticated controllers to adapt to dynamic changes, leading to the development of model reference adaptive control [72]. However, early efforts lacked stability proof, causing issues in flight tests [73]. Advances in stability and control theory in the 1960s and 1970s, including breakthroughs in model reference adaptive control using the Lyapunov design approach, significantly enhanced understanding and application in adaptive control.

Recent adaptive control literature focuses on nonlinear dynamical systems with parametric uncertainties, particularly feedback linearisable systems reliant on unknown parameters, exemplified by adaptive control in robot arms [74]. The predominant method for these systems involves the certainty equivalence principle and a parameter update law to transform a quadratic function of states and parameter estimation errors into a Lyapunov function, ensuring global stability and boundedness of closed-loop signals. This method removes parameter-dependent terms from the Lyapunov function's derivative. While the work in [74], [75] was foundational, it overlooked environmental uncertainties, an aspect addressed in thesis.

Begin with [76], introduced the I&I control theory, emphasizing the concepts of system immersion and manifold invariance. It highlights the method's ability to design stabilizing control laws without requiring a Lyapunov function, similar to nonlinear regulator theory. Researchers expanded the I&I framework to various applications and began exploring its application in adaptive control as an alternative to adaptive backstepping and sliding mode control [77]–[79]. This approach, different from the classic adaptive method, does not require the linear parameterization condition, nor does it invoke certainty equivalence. It also simplifies the stability analysis by providing cross terms in the Lyapunov function [79]. In [80], an I&I method is used to estimate the unknown mass of a VTOL vehicle, and it guarantees that the estimation converges to its true value. In [81], to control a mini quadrotor UAV and overcome the uncertainties related with the thrust and drag coefficients, Fujimoto et al. simplified the dynamic model and developed an adaptive controller via the I&I methodology. Work in [82] design a outerloop I&I adaptive controller for quadrotor free motion. Apart from the control of UAV, I&I is also applied in visual servoing, aerospace control of pendulum on cart, and many other mechatronic systems [83], [84]. Recent work [85] shows that I&I adaptive control can achieve asymptotic stability without PE. Work [86] applied I&I to path following of underactuated mechanical systems to achievable orbital stabilization.

1.2.3 Reinforcement Learning for UAV Motion Control

The problem of motion control for UAVs is a primary area of research which continues to generate interest. A survey of the area is given in [47]. The interest in motion control comes from a practical need for high performance trajectory tracking which is robust to external force disturbances and changing dynamics. For example, external disturbance forces arise in load transportation [87] or during non-destructive contact inspection [9]. UAV dynamics can change when rotor thrust is affected by nearby obstacles [88]. Other sources of challenge include underactuated nonlinear vehicle dynamics [46], input bounds [89], noise/delay in measurements and state estimates [90]–[92], and limited sensing with autonomous motion control, e.g., position tracking in GPS denied environments or motion control relative to visual targets [93]. Much of the work on UAV motion control is traditionally model-based and the design procedure relies on a physics-based dynamics. Traditionally, a control law has a fixed structure influenced by the model equations and which has adjustable gains that affect the closed-loop performance. Tuning these gains provides a practical implementation challenge. Although a rigourous statement regarding convergence is typical in traditional control, it often holds under ideal modelling assumptions. Such shortcomings of traditional methods has led to interest in RL being applied to UAV motion control. RL has the advantage of requiring less domain knowledge in that the design method can be applied to a generic system. For example, the proposed method uses only simulation data to train the control law or policy which maximizes an accumulated reward to minimize regulation error. No particular controller structure is needed in advance other than it be a static state feedback. As well, controller tuning is performed automatically during training.

RL is concerned with how agents act within an environment in order to maximize a scalar cumulative reward which measures long term performance. RL enables a robot to autonomously discover optimal behaviour through trial and error interactions with its environment. Recent research has shown impressive performance from so-called *Deep RL*, where “deep” relates to the use of neural nets (NNs) with multiple layers. Deep RL has been shown to outperform human experts in complex tasks. For example, playing Go [94], Atari games [95], and solving a Rubik’s Cube [96]. These examples involve action and state spaces which are discrete. Although RL has been used in robotics for decades, it has mainly been confined to the high-level decisions (e.g., trajectory planning [97]) rather than control of low-level actuators. This is because high-level decisions are easier to discretize and thus admit tabular methods. Recently, it has been shown that deep RL can also be applied to continuous action and state spaces in robotics [98] due to improved computational power.

Existing work on RL applied to UAVs can be divided into model-based and model-free methods. Model-based refers to learning an optimal controller indirectly by learning a model which can predict the future system state given the current state and input. Model-free methods learn an optimal controller directly by observing the reward and system state for a given input. Both model-based and model-free methods have attracted much attention in recently year. We begin by surveying some of the relevant model-based approaches. Earlier work [99] considers a design in three stages where initially an apprenticeship learning algorithm extracts a reference trajectory for the UAV from suboptimal expert pilot demonstrations. Secondly, a full nonlinear dynamic model for a helicopter UAV is identified. Thirdly, a receding horizon variation of linear quadratic control based on a linearized model is applied to the identified dynamics. Experimental results are provided. Work [100] focuses on inner loop control which maps Inertial Measurement Unit (IMU) measurements to Pulse Width Modulation (PWM) waveforms for the Electronic Speed Controller (ESC). During training, a NN identifies a model of the rotational dynamics which is combined with a Random Shooter Model Predictive Control (MPC) which finds an optimal control by simulation. Experimental results are provided on the Crazyflie platform and short term hover performance is demonstrated in experiment. No position loop control is considered. Work [101] uses the method in [102] to design two separate Incremental Dual Heuristic Programming (IDHP) controllers for collective and longitudinal cyclic pitch inputs of a full-sized helicopter. These inputs control the altitude and pitch DoF. The outer lateral position DoF loops and roll/yaw control are PID controllers. The reduced number of DoF controlled by RL control leads to faster learning of simpler dynamics with reduced coupling. An "incremental" linear model is estimated during training which corresponds to the linearization of the system dynamics about its trajectory. A detailed simulation model is used to train the controller and test it's performance. Work [103] presents an end-to-end RL method for the full system dynamics with only IMU and a single monocular camera as sensor input. It uses an abstraction of camera images and IMU measurements in order to improve transfer from simulation to the real-world. These processed measurements are fed to a NN which is trained in a simulator to imitate MPC expert demonstrations with privileged simulation data (i.e., the full UAV state). Experimental results are demonstrated for aggressive trajectories. From the above we observe model-based RL has evolved to eliminate the need for expert demonstration. The adoption of NNs in model-based RL has allowed the solution of more complex problems such as end-to-end vision based control [103].

Model-free RL has been investigated recently to develop NN-based controllers. These approaches are *model-free* as they do not try to estimate the transition dynamics associated with the Markov Decision Process (MDP). Representative work in this area is deep Q-learning (DQN) [104], deep deterministic policy gradient methods (DDPG) [105], trust region policy optimization (TRPO) [106], and proximal policy optimization (PPO) [95]. Often model-free RL is a *policy-gradient method*. This means a parameterized policy (or control law) is optimized using gradient descent. The PPO method is a policy-gradient method and is used in this paper. This method was chosen based on recent demonstrated performance [95].

A number of researchers have applied model-free RL to UAV motion control. Model-free approaches are often considered less computational efficient than model-based methods. However, model-free methods are generally easier to implement and currently receive more attention in the RL community. Work [107] applies Q-learning to the UAV landing problem. The controller outputs high-level commands (i.e., forward, backward, left, right, down) as opposed to low level physical inputs (i.e., ESC PWM). Work [108] uses the PPO method to solve the attitude control of fixed-wing UAV. The attitude dynamics are inherently stable which makes the problem less challenging than quadrotor control. Work [109] improves the DDPG algorithm by using a double experience replay buffer (DERB) for training. This DERB-DDPG control is trained using a linearized outer loop system dynamics. Simulation results are presented using an inner-outer loop control. Work [110] proposes an improvement on the DDPG algorithm by accurately estimating the gradient of the objective function w.r.t. action. The authors use a Singular Value Decomposition (SVD) method to find the exact solution of the Hessian matrix inverse. This value is needed to update the actor NN parameters. Motion control is provided for the full quadrotor dynamics. A PD controller for attitude control is used to assist the learning and testing performance. Experiments show hover stabilization for a wide range of initial states.

1.3 Outline

This document is organized as follows:

Chapter 2 starts with modelling for an SLS, fully actuated hexrotor platform. The modelling is presented using Euler-Newton method. The rotor drag force is considered in SLS for fast maneuvers. The chapter also includes the presentation of the hardware and software components of the quadrotor UAV platforms developed, upgraded and maintained during the course of this thesis. It briefly explains the specifications and purpose of each component and highlights the important interconnections between them.

Chapter 3 presents the QSFA and its application to SLS. The QSFA can be applied to general nonlinear systems. This chapter also describes how QSFA is applied to SLS. SITL results is presented to validate the QSFA controller performance and its comparison with the state-of-the-art geometric controller.

Chapter 4 first presents the outer-loop dynamics of SLS. It describes applying QSFA to the outer-loop dynamics of the SLS. Both SITL and real-world experiments are presented to validate the performance of the proposed controller.

Chapter 5 presents preliminary knowledge of I&I adaptive control. It also presents the I&I adaptive hybrid force-motion control for a simple explanatory 2 DoF Cartesian robot and UAM interacting with a rigid plane surface. The chapter also presents the simulation results of I&I adaptive controller.

Chapter 6 presents preliminary knowledge of RL. It also presents a detailed description of PPO algorithm and how training processes is conducted in finding the controller for UAV free motion. The chapter also presents the python simulation environment and the results of the training process and the trained controller.

The document concludes with Chapter 7 where it summarizes the whole thesis and provides a conclusion. Also, the limitations of the work presented and future directions for research are outlined.

1.4 Contributions

The contributions made in this thesis are listed below:

- A novel general procedure, called the quasi-static feedback algorithm (QSFA), is invented for computing a QSF for nonlinear control affine systems. Such an algorithm does not appear in the literature to date. Rather, existing work described QSF using specific examples [111]–[113]. Another contribution is to apply the QSFA to the SLS. Similar to the Dynamic Extension Algorithm (DEA), the QSFA provides a straightforward and systematic procedure for testing whether a system is flat relative to a given output. Hence, our work [40], [114] is a practical method for testing flatness which contrasts with the intricate theory such as [14], [15] which applies to a restricted system class.

- This thesis presents an implementation of the proposed QSF in a PX4 software-in-the-loop (SITL) environment to validate performance. This is an important contribution as it shows the developed control law can be implemented on typical autopilots and is robust to model error introduced in the SITL simulation environment. For example, the SITL simulator models drag forces at each rotor whereas the model used for control design lumps the effect of individual rotors together. Both models are based on the same underlying physical reasoning given in [115]. A video shows the practical performance of the proposed control in a SITL simulation `https://www.youtube.com/watch?v=Js6GVBOfXts`. The code for the PX4-SITL-Gazebo simulation is provided at `https://github.com/ANCL/SLS_PX4_SITL`.

- A significant benefit of multi-loop control is that it is generally simpler to implement than single-loop approaches. Based on our past work [40], [116], we observed an exact QSF linearization of the entire dynamics led to complex controller expressions that are unwieldy for onboard implementation. Although it is possible to implement such a single-loop QSF control, the reduced complexity multi-loop design proposed in the work is more efficient. We still adopt a QSF approach but apply it to the outer-loop dynamics. The static QSF linearization of the outer-loop is a contribution, as this design has not appeared to date. The QSF yields the benefits of exact static state linearizability, such as ease-of-tuning and a simple stability result as the error dynamics are linear-time-invariant (LTI).

- A main contribution of Chapter 4 is its experimental validation of a reduced complexity trajectory-tracking flatness-based QSF using an open-source drone platform. The code used to generate this paper's experimental and simulation results can be downloaded, verified, and extended. The drone hardware and software are state-of-the-art, open-source, inexpensive, and readily available today at [117], [118]. This is a key point as much of the existing work has no experimental validation, e.g., [119], or is based on closed-source hardware and software platforms with no published code making results impossible to reproduce or extend. For example, [24], [30] where MATLAB/Simulink is used and no code provided or [27] where unavailable Astec drones are used and no source code is provided. The importance of open platforms and reproducible results is echoed in [120]. A video of the experimental results described in Section 4.3.2 is at `https://www.youtube.com/watch?v=wyEOHNX4Rf8` and the code used to perform these experiments is at `https://github.com/ANCL/QuasiSLSExp`.

- In this thesis, we present the adaptive hybrid force-motion controller for UAM interacting with a rigid plane surface. We first introduce the I&I control framework and necessary assumptions. Then, an explanatory example is given to explain how the I&I adaptive control techniques can be applied in the hybrid force-motion control problem. We also compare the I&I adaptive control with the conventional adaptive control based on Lyapunov design. The simple example shows that the conventional adaptive control is not able to give asymptotically stable result and has no guarantee on parameter convergence. In contrast, I&I adaptive control overcomes these challenges by adding parameter observer states. The I&I adaptive control gives global asymptotically stabilisability results. We apply the I&I adaptive control to the UAM hybrid force-motion control problem. Specifically, we consider the interaction problem with two common force sensors, 1 DoF and 3 DoF force sensors. The I&I adaptive hybrid control guarantees the exponential convergence of both system states, reaction forces and parameter estimations. This is verified in simulation in Section. 5.5.

- Chapter 6 proposed PPO-based method is unique relative to existing approaches in that solves the motion control problem for the full dynamics with the control output being low-level system inputs: total thrust and torque. Actuating low level inputs and controlling all six UAV DoF is more challenging than controlling a subset of DoF as in [100], [109]. As well, no traditional control is used to assist the RL during training or testing. This should be compared to work [110] where PID assists the RL. Our approach investigates the choice of reward function on time-domain tracking performance. Previous RL work does not focus on this design parameter which we show is useful in improving actual closed-loop performance. Simulation results show hover and trajectory tracking performance which is comparable to a manually tuned inner-outer loop PD control. Hence, the method benefits from not requiring a tuning stage as in traditional control. The code and simulation data for this project are available at `https://github.com/ANCL/QuadPPO`.

- The author has enhanced the ANCL flight platform, transitioning from the ANCL Q3 to the more advanced ANCL Gen2 platform. This upgrade marks a significant departure from the earlier ANCL Q2 and Q3 models, as referenced in previous works [121], [122]. The ANCL Gen2 showcases substantial improvements in usability, maneuverability, and stability. It features an compactly designed on-board and off-board system structure, which is highly conducive to executing complex control algorithms and handling computationally intensive tasks. Furthermore, there has been a noteworthy update in the PX4 autopilot system used in the ANCL Gen2, advancing from version 1.5.5, released in 2016, to version 1.13.3, launched in 2023. The author's modular Vicon data transmission framework streamlines the integration process with the PX4 autopilot, significantly reducing the need for extensive modifications. This efficient design approach enables the ANCL lab to easily stay in sync with the latest PX4 firmware updates from the community with minimal effort.

Chapter 2

Platform & Modelling

This chapter introduces the dynamic models of SLS and UAM. The notation is presented here and will be used throughout the thesis. In Section 2.1, we present the SLS model. In Section 2.2, we present the UAM model. In addition to dynamics, Section 2.3 present the hardware platform and software architecture used in this thesis.

2.1 Sung Load System Dynamics

The suspended load is modeled as a universal joint pendulum with two rotation DoF attached to the drone's center of mass (CoM). Fig. 2.1 summarize the notation. Two reference frames are used: a navigation frame $\mathcal{N}$ fixed to the earth and a body frame $\mathcal{B}$ attached to the drone. We assume that $\mathcal{N}$ is inertial and has an orthonormal basis $\{n_1, n_2, n_3\}$ of vectors oriented north, east, and down, respectively. The origin of $\mathcal{B}$ is the drone's CoM, and its basis $\{b_1, b_2, b_3\}$ has vectors oriented forward, right, and downward, respectively. We denote pendulum position p_L, pendulum attitude q, and drone attitude R. The SLS configuration variable $[p_L, R, q] \in SE(3) \times \mathbb{S}^2$. The unit vector q is expressed in $\mathcal{N}$ and parameterized with angles α and β where α is a rotation about n_1 and β is about n_2. We have

$$q = R_{n_1}(\alpha) R_{n_2}(\beta) n_3 = [s_\beta, -s_\alpha c_\beta, c_\alpha c_\beta]^T$$

where R_{n_1} and R_{n_2} are elementary rotation matrices about n_1 and n_2 axis, respectively.

The relation between load position p_L and quadrotor position p_Q is

$$p_L = p_Q + Lq = p_Q + L[s_\beta, -s_\alpha c_\beta, c_\alpha c_\beta]^T \tag{2.1}$$

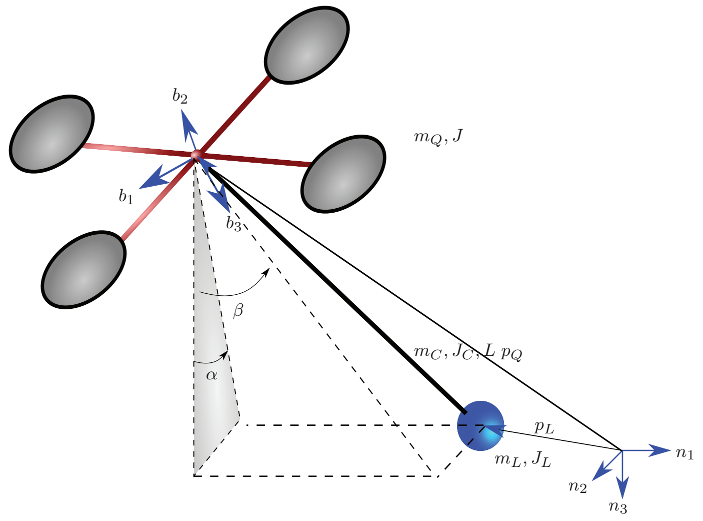


Figure 2.1: The SLS is modelled as a spherical pendulum and multirotor drone.

where L is rod length. Differentiating (2.1) gives

$$\begin{aligned} v_L &= v_Q + L\dot{q} \\ \dot{v}_L &= \dot{v}_Q + L\ddot{q} \end{aligned} \tag{2.2}$$

We extend the model used in [114] to include rod mass and inertia. The relation between the rod CoM p_p and drone position p_Q is

$$p_p = p_Q + \frac{1}{2}Lq$$

Hence, differentiating gives

$$v_p = v_Q + \frac{1}{2}L\dot{q}, \quad \dot{v}_p = \dot{v}_Q + \frac{1}{2}L\ddot{q}$$

Drag forces have been shown to be important in recent aggressive quadrotor flying experiments such as [20], [22], [123]. Hence, we adopt the quadrotor dynamics, which models rotor drag under the assumption of no wind in [123]:

$$\dot{p}_Q = v_Q \tag{2.3a}$$
$$m_Q\dot{v}_Q = m_Q g n_3 - R\bar{u}n_3 + Tq - RDR^T v_Q \tag{2.3b}$$
$$\dot{R} = RS(\omega) \tag{2.3c}$$
$$J\dot{\omega} = -\omega \times J\omega + \tau - AR^T v_Q - B\omega \tag{2.3d}$$

where $J = \mathrm{diag}(J_1, J_2, J_3)$ is the inertia the drone about its CoM, $T \geq 0$ denotes rod tension, $\bar{u} \geq 0$ is total propeller thrust, τ is propeller torque expressed in $\mathcal{B}$, g is the acceleration of gravity, and m_Q is drone mass. The rotor drag parameters are constant matrix $D = \text{diag}(d_x, d_y, d_z) > 0$ for drag force coefficients, and constant matrix A and B for drag moment coefficients. See [115] for the details on rotor drag modelling. The skew operator $S(\cdot) : \mathbb{R}^3 \to so(3)$ in (2.3c) is

$$S(x) = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}, \quad \text{where } x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}.$$

We assume that $T \geq 0$ which is typical of normal SLS operation. We assume J is diagonal for simplicity of presentation.

The rotational kinematics (2.3c) is parameterized with the ZYX Euler angles $\eta = [\phi, \theta, \psi]^T \in \mathbb{R}^3$:

$$\dot{\eta} = W(\eta)\omega \tag{2.4}$$

with

$$W(\eta) = \begin{bmatrix} 1 & s_\phi t_\theta & c_\phi t_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi / c_\theta & c_\phi / c_\theta \end{bmatrix}$$

where $t_\theta = \tan\theta, s_\theta = \sin\theta, c_\theta = \cos\theta$.

We assume the rod has mass m_C and an inertia matrix J_C relative to its CoM given by

$$J_C = \text{diag}(\frac{1}{12} m_C L^2, \frac{1}{12} m_C L^2, 0)$$

That is, we assume a thin rod with no radial dimension.

The translational pendulum dynamics is

$$\dot{p}_L = v_L \tag{2.5a}$$

$$m_L \dot{v}_L = -Tq + (m_L + m_C) g n_3 \tag{2.5b}$$

The rotational pendulum dynamics is

$$\dot{q} = \omega_L \times q \tag{2.6a}$$

$$J_L \dot{\omega}_L = -\omega_L \times J_L \omega_L + Lq \times (m_L g n_3 - m_L \dot{v}_Q) + \frac{1}{2} Lq \times (m_C g n_3 - m_C \dot{v}_Q) \tag{2.6b}$$

where J_L is the inertia of m_L about the drone's CoM. Using Steiner's Theorem [124], we can express J_L as a function of the relative position between the load and drone:

$$J_L = m_L L^2 (I - qq^T) + J_C + m_C \frac{L^2}{4} (I - qq^T) \tag{2.7}$$

Eliminating rod tension T in (2.5b) and (2.3b) gives

$$m_Q\dot{v}_Q + m_L\dot{v}_L = (m_Q + m_L + m_C)gn_3 - R\bar{u}n_3 - RDR^T v_Q \tag{2.8}$$

Substituting for $\dot{v}_Q$ in (2.8) using (2.2), we have the translational dynamics for the load

$$(m_L + m_Q)\dot{v}_L = (m_L + m_Q + m_C)gn_3 - R\bar{u}n_3 + m_Q L\ddot{q} - RDR^T v_Q \tag{2.9}$$

where we have substituted $\ddot{q} = \dot{\omega}_L \times q + \omega_L \times \dot{q}$ from (2.6a). Combining (2.9) and (2.2), we get

$$\dot{v}_Q = -\frac{m_L}{m_Q + m_L}L\ddot{q} + (1 + \frac{m_C}{m_L + m_Q})gn_3 - \frac{R\bar{u}n_3 + RDR^T v_Q}{m_Q + m_L} \tag{2.10}$$

Substituting (2.7), and (2.10) into (2.6b), we obtain the rotational dynamics of the load expressed in $\mathcal{N}$:

$$\begin{aligned} J_L\dot{\omega}_L = -\omega_L \times J_L\omega_L + Lq \times \big((m_L + \frac{m_C}{2}) \\ (gn_3 + \frac{m_L}{m_Q + m_L}L\ddot{q} - (1 + \frac{m_C}{m_L + m_Q})gn_3 + \frac{R\bar{u}n_3 + RDR^T v_Q}{m_Q + m_L})\big) \end{aligned} \tag{2.11}$$

In addition, we have the relation between ω_L and $\gamma_\alpha, \gamma_\beta$

$$\omega_L = \begin{bmatrix}1\\0\\0\end{bmatrix}\gamma_\alpha + \begin{bmatrix}0\\c_\alpha\\s_\alpha\end{bmatrix}\gamma_\beta \tag{2.12}$$

and

$$\begin{aligned} \dot{\omega}_L &= \begin{bmatrix}1\\0\\0\end{bmatrix}\dot{\gamma}_\alpha + \begin{bmatrix}0\\c_\alpha\\s_\alpha\end{bmatrix}\dot{\gamma}_\beta + \begin{bmatrix}0\\-s_\alpha\\c_\alpha\end{bmatrix}\gamma_\alpha\gamma_\beta \\ &= R_{n_1}(\alpha)\begin{bmatrix}\dot{\gamma}_\alpha\\ \dot{\gamma}_\beta\\ \gamma_\alpha\gamma_\beta\end{bmatrix} \end{aligned} \tag{2.13}$$

Substituting (2.12) and (2.13) into (2.9) and (2.11), and solving for $\dot{v}_L, \dot{\gamma}_\alpha, \dot{\gamma}_\beta$, we obtain the SLS dynamics

$$
\begin{aligned}
\dot{x} &= f(x) + g(x)u \qquad (2.14)\\
f(x) &= \tilde{f}(x) + d_f(x)RDR^T(v_L - L\dot{q}) + d_\tau(AR^T(v_L - L\dot{q}) + B\omega)\\
\tilde{f}(x) &= \begin{bmatrix} v \\ \gamma_\alpha \\ \gamma_\beta \\ W(\eta)\omega \\ -\mathrm{s}_\beta(\gamma_\alpha^2\mathrm{c}_\beta^2 + \gamma_\beta^2)LM_0 \\ \mathrm{s}_\alpha\mathrm{c}_\beta(\gamma_\alpha^2\mathrm{c}_\beta^2 + \gamma_\beta^2)LM_0 \\ g - \mathrm{c}_\alpha\mathrm{c}_\beta(\gamma_\alpha^2\mathrm{c}_\beta^2 + \gamma_\beta^2)LM_0 \\ 2\gamma_\alpha\gamma_\beta\mathrm{t}_\beta \\ -\gamma_\alpha^2\mathrm{c}_\beta\mathrm{s}_\beta \\ -J^{-1}S(\omega)J\omega \end{bmatrix}, g(x) = \begin{bmatrix} 0_{8\times1} & 0_{8\times3} \\ \bar{g}(x) & 0_{5\times3} \\ 0_{3\times1} & J^{-1} \end{bmatrix}
\end{aligned}
$$

with $x = [p_L^T, \alpha, \beta, \eta^T, v_L^T, \gamma_\alpha, \gamma_\beta, \omega^T]^T \in \mathbb{R}^{16}$, $u = [\bar{u}, \tau^T]^T \in \mathbb{R}^4$, $M_0 = (2m_Q + m_L)/(2(m_Q + m_L + m_C))$ and $d_\tau \in \mathbb{R}^{16\times3} = [0_{3\times13}, J^{-T}]^T$. $d_f(x), \bar{g}$ is given by

$$
\bar{g}(x) = -d_f(x)Rn_3\, , d_f(x) = \begin{bmatrix} & 0_{8\times3} & \\ \frac{\mathrm{s}_\beta^2}{M_1} & -\frac{\mathrm{s}_\alpha\mathrm{s}_\beta\mathrm{c}_\beta}{M_1} & \frac{\mathrm{c}_\alpha\mathrm{s}_\beta\mathrm{c}_\beta}{M_1} \\ -\frac{\mathrm{s}_\alpha\mathrm{s}_\beta\mathrm{c}_\beta}{M_1} & -\frac{\mathrm{s}_\alpha^2\mathrm{c}_\beta^2}{M_1} & -\frac{\mathrm{s}_\alpha\mathrm{c}_\alpha\mathrm{c}_\beta^2}{M_1} \\ \frac{\mathrm{c}_\alpha\mathrm{s}_\beta\mathrm{c}_\beta}{M_1} & -\frac{\mathrm{s}_\alpha\mathrm{c}_\alpha\mathrm{c}_\beta^2}{M_1} & \frac{\mathrm{s}_\alpha^2\mathrm{c}_\beta^2}{M_1} \\ 0 & -\frac{\mathrm{c}_\alpha}{Lm_Q\mathrm{c}_\beta} & \frac{\mathrm{s}_\alpha}{Lm_Q\mathrm{c}_\beta} \\ -\frac{\mathrm{c}_\beta}{Lm_Q} & -\frac{\mathrm{s}_\alpha\mathrm{s}_\beta}{Lm_Q} & \frac{\mathrm{c}_\alpha\mathrm{s}_\beta}{Lm_Q} \\ & 0_{3\times3} & \end{bmatrix} \qquad (2.15)
$$

where $M_1 = m_Q + m_L + m_p$. The drift vector field of (2.14) has singularities at $\mathrm{c}_\beta = \mathrm{c}_\theta = 0$ due to parametrizations used for the orientation of the drone and pendulum. Hence, domain $\mathcal{M}$ of x is taken to be a subset of $\mathbb{R}^{16}$ including 0 but excluding these points. Below we show these singularities appear in the controller. From a practical point of view, these points are not encountered in typical SLS motion. As well, $\mathrm{c}_\beta = 0$ is not physically possible as it means the rod collides with the frame of the drone.

In order to improve robustness to unmodelled effects, we introduce a force disturbance $f_e \in \mathbb{R}^3$ acting on the UAV CoM, and a torque disturbance $\tau_e \in \mathbb{R}^3$ acting about the UAV's CoM. We have

$$m_Q \dot{v}_Q = m_Q g n_3 - R\bar{u} n_3 + Tq - f_e - RDR^T v_Q \tag{2.16a}$$

$$J\dot{\omega} = J\omega + \tau - \tau_e - AR^T v_Q - B\omega \tag{2.16b}$$

This leads to a state space form

$$\dot{x} = f(x) + g(x)u + d_f(x) f_e + d_\tau(x)\tau_e \tag{2.17}$$

2.2 Unmanned Aerial Manipulator Dynamics

We denote the world inertia frame with $\mathcal{N}$, the body frame $\mathcal{B}$, attached to the hexrotor frame, where origin coincides with the hexrotor CoM. Let the frame associated with the ith propeller be defined as $\mathcal{P}_i$. The origin O_{P_i} is the center of the spinning and the CoM of the ith propeller. The axes X_{P_i} is the axis along the rotor arm and Z_{P_i} is the axis about which the propeller spins and coincides with the thrust direction. The Y_{P_i} axis is defined such that the frame $\mathcal{P}_i$ is right-handed. The coordinate definitions is shown in Figure 2.2 and the real-world platform picture is shown in Figure 2.3. The propeller thrust is denoted by T_{thrust_i} and drag torque to the body is denoted by τ_{drag}. The thrust and torque are related to the propeller angular velocity ω_i by the following equations:

$$T_{thrust_i} = [0, 0, C_T \omega_i^2], \quad \tau_{drag_i} = [0, 0, C_D (-1)^i \omega_i^2] \tag{2.18}$$

We define $p_i^B \in \mathbb{R}^3$ the position O_{P_i} of the ith propeller in the body frame $\mathcal{B}$. $p_i^B \in \mathbb{R}^3$ is given by

$$p_i^B = R_Z(\lambda_i) \begin{bmatrix} L_{x_i} \\ 0 \\ 0 \end{bmatrix}, \forall i \in \{1, \ldots, 6\} \tag{2.19}$$

where $R_Z(\lambda_i)$ is the rotation matrix about the Z axis by λ_i and L_{x_i} is the distance between the ith propeller and the CoM of the hexrotor.

The parameters λ_i and L_{x_i} represent the geometric perspective of the hexrotor. To get the total thrust and torque, we sum up the thrust and torque from each propeller. Each rotor generates a thrust and torque in z_{p_i} axis. The rotation matrix $R_{p_i}^B$ is defined as the rotation matrix from $\mathcal{P}_i$ to $\mathcal{B}$. In order to have clear geometric perspective, we decompose the rotation matrix $R_{p_i}^B$ into three rotation matrices about x_{p_i}, y_{p_i} and z_{p_i} axis

$$R_{P_i}^B = R_z(\lambda_i) R_x(\alpha_i) R_y(\beta_i) \tag{2.20}$$

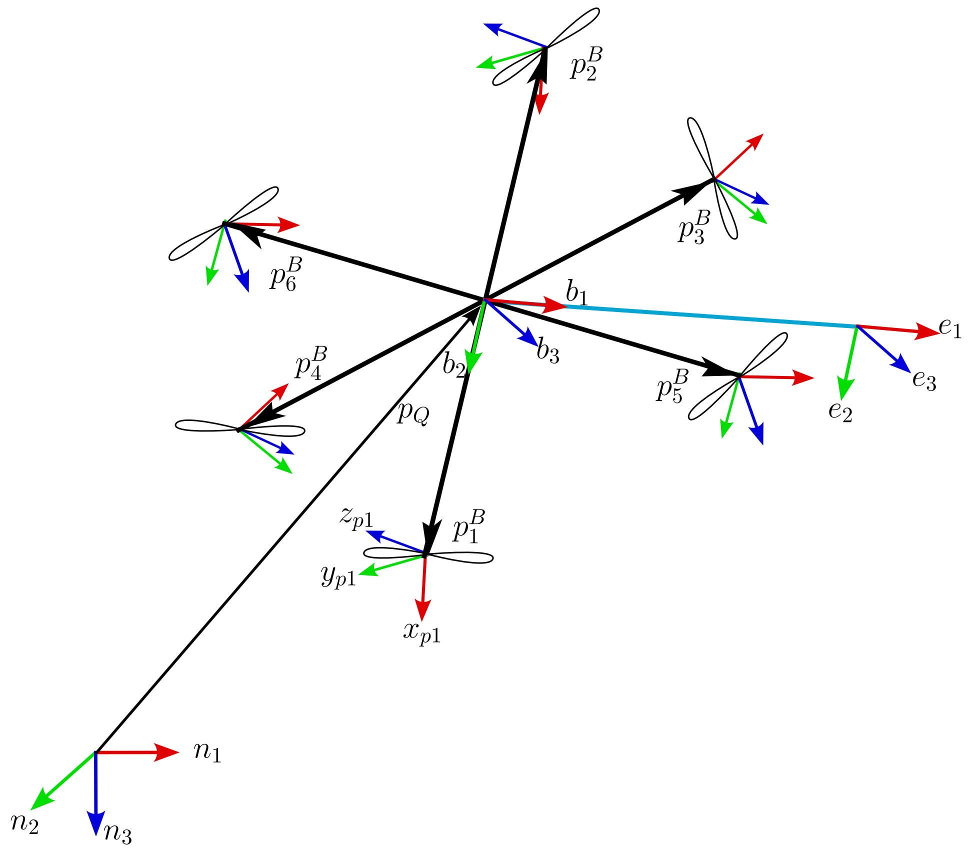


Figure 2.2: The FA hexrotor

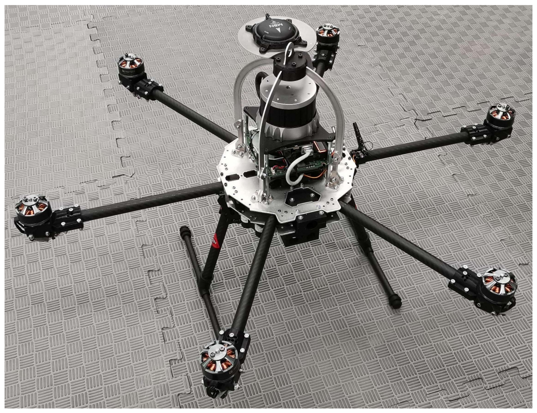


Figure 2.3: The FA hexrotor rotor platform at ANCL.

where

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix},$$

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix},$$

$$R_z(\lambda) = \begin{bmatrix} \cos\lambda & -\sin\lambda & 0 \\ \sin\lambda & \cos\lambda & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

$R^B_{P_i}$ can be interpreted by first applying a rotation about z_{p_i} axis by λ_i, then a rotation about x_{p_i} axis by α_i and finally a rotation about y_{p_i} axis by β_i.

The total thrust T_{sum} and torque τ_{sum} are given by

$$T_{sum} = \sum_{i=1}^{6} R^B_{P_i} T_{thrust_i} \tag{2.21}$$

$$\tau_{sum} = \sum_{i=1}^{6} R^B_{P_i} \tau_{drag_i} + p^B_i \times R^B_{P_i} T_{thrust_i} \tag{2.22}$$

The dynamics of the hexrotor is given by

$$m_Q \dot{v}_Q = m_Q g n_3 - R T_{sum} - f_e \tag{2.23}$$

$$J\omega = -\omega \times J\omega + \tau_{sum} - \tau_e \tag{2.24}$$

where $T_{sum} = [T_1, T_2, T_3]$ is the total thrust and $\tau_{sum} = [\tau_1, \tau_2, \tau_3]$ is the total torque.

The end-effector position p_E is given by

$$p_E = p_Q + R^B [x_e, y_e, z_e]^T \tag{2.25}$$

where R^B is the rotation matrix from $\mathcal{B}$ to the end-effector frame $\mathcal{E}$. The rotation matrix R^B is given by$R^B = I_3$ as Figure 2.2.

2.3 Platform Hardware and Software Architecture

Our ANCL Gen2 platforms are based on Holybro vision kit. The common components of a ANCL Gen2 are shown in Figure 2.4 and their interconnection is shown in Figure 2.5. Compare the previous general platform ANCL Q2 and Q3 [121], [122], ANCL Gen2 has improved significantly in terms of usability, maneuverability and stability. ANCL Gen2 also features a compact on-board and off-board system structure, which are suitable for complex control algorithm implementation and more computational heavy tasks. The ANCL Gen2 upgraded PX4 version from v1.5.5 released in 2016 to v1.13.3 released in 2023. The author designed a modular Vicon data transmit framework, which minimizes the changes needed to be done in the

PX4 autopilot side and this makes ANCL lab is able to keep up with the latest firmware version with the PX4 community with minimum efforts.

2.3.1 Hardware Components

The essential components of a quadrotor UAV and the specifications of those used in the ANCL platforms are described below.

Flight Controller

A flight controller is a mini-computer onboard a UAV that has built-in sensors and communication ports. The onboard central processing unit (CPU) on a flight controller runs the autopilot software that contains the estimation, control and safety-related decision-making algorithms to fly the UAV. Built-in sensors provide an various measurements that are used by the estimation algorithms to determine system states. The communication ports allow connection with external sensors, actuators, ground station computers, and remote controllers.

ANCL Gen2 use Pixhawk 6C from Holybro. The Pixhawk 6C is the latest update to the successful family of Pixhawk flight controllers, based on the Pixhawk flight management unit (FMU) v6C Open Standard and Connector Standard. Inside the Pixhawk 6C, you can find an STMicroelectronics based STM32H743, paired with sensor technology from Bosch & InvenSense, giving user flexibility and reliability for controlling any autonomous vehicle, suitable for both academic and commercial applications. The Pixhawk 6C's H7 microcontroller contain the Arm Cortex-M7 core running up to 480 MHz, has 2MB flash memory and 1MB RAM. Thanks to the updated processing power, the author can be more productive and efficient with the development work, allowing for complex algorithms and models. The FMUv6C open standard includes high-performance, low-noise IMUs (ICM-42688-P and BMI055) on board, designed to be cost effective while having IMU redundancy. A vibration isolation System to filter out high-frequency vibration and reduce noise to ensure accurate readings, allowing vehicles to reach better overall flight performances.

Motion Capture System (MCS)

The UAVs can obtain a position estimate from a GPS during outdoor flights; however, GPS is unavailable for indoor flight operations. Therefore, ANCL has a Motion Capture System in the lab that provides UAV pose estimates with an accuracy down to 1 mm in translational and 1 degree in the rotational. When testing control algorithms, this pose estimate is used either as a ground truth or a system state. ANCL motion capture system consists of 10 Vicon Vero Cameras with a resolution of 3.3 megapixels and a frame rate of 330 frames per second (fps). The

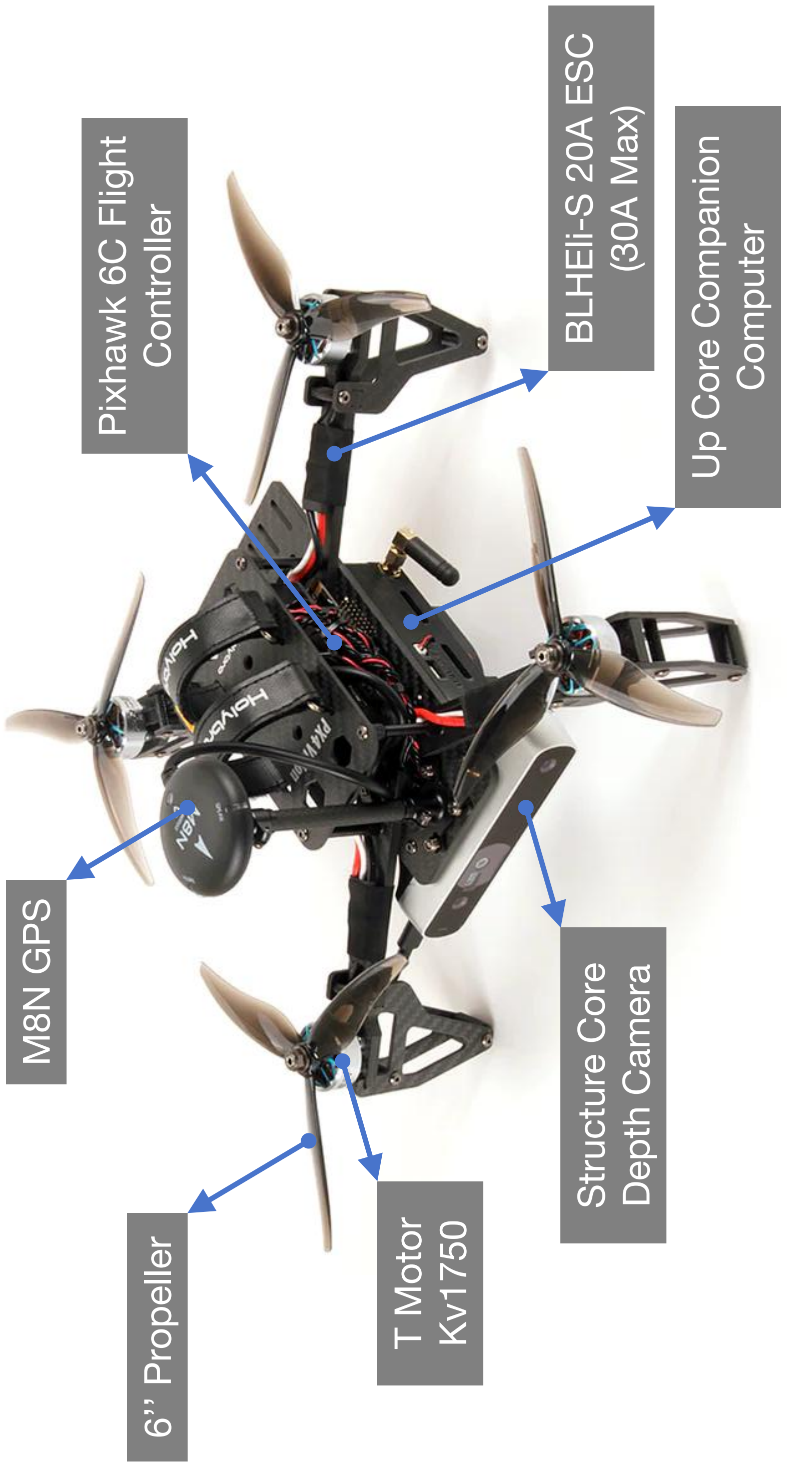

Pixhawk 6C Flight Controller
BLHEli-S 20A ESC (30A Max)
Up Core Companion Computer
M8N GPS
Structure Core Depth Camera
6'' Propeller
T Motor Kv1750

Figure 2.4: ANCL Gen2 components (Holybro PX4 Vision Kit).

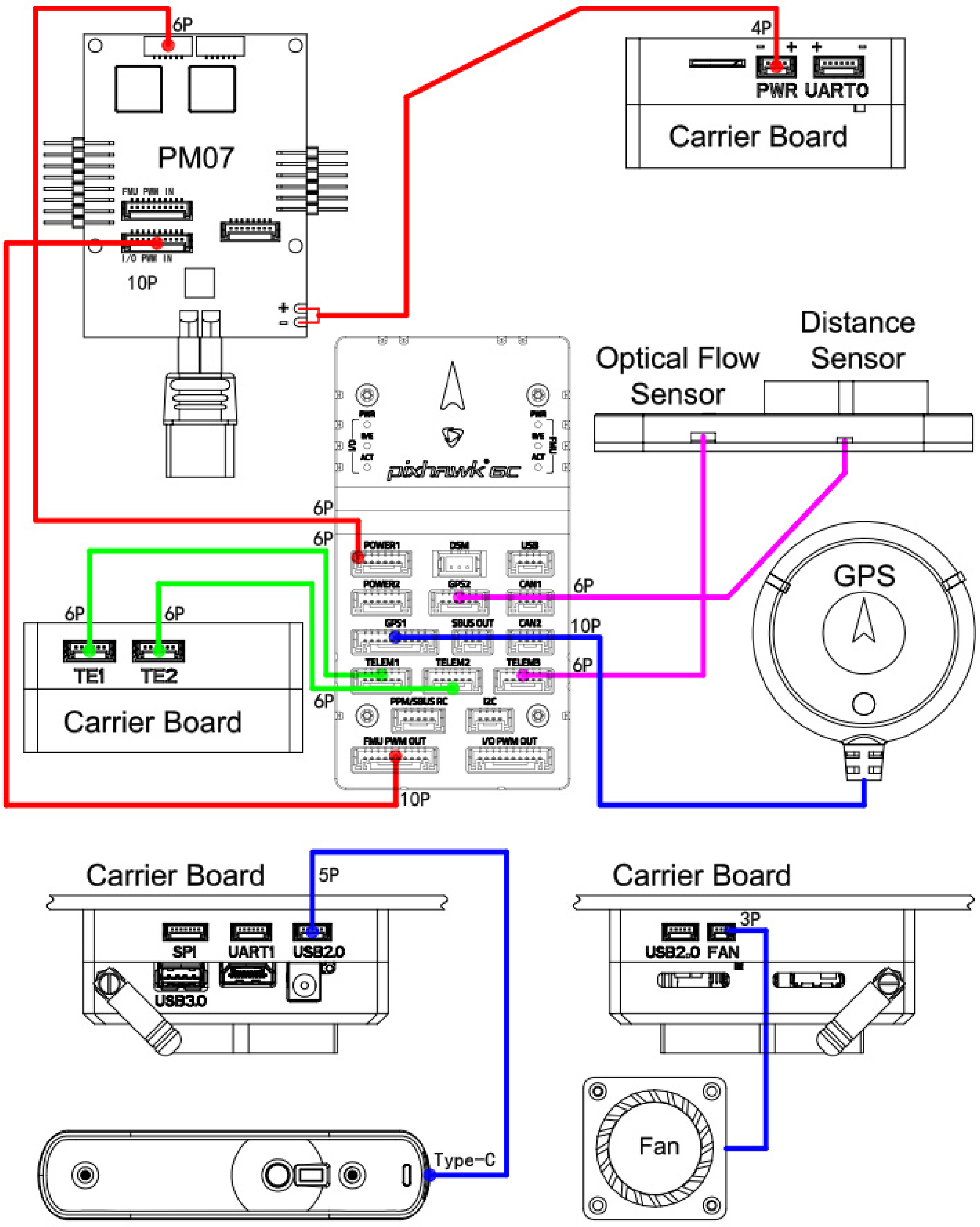


Figure 2.5: Hardware interconnections of ANCL Gen2 platforms [125].

cameras emit infrared light, which is reflected by the infrared markers in the room, and projected on cameras to track the location of markers. The system can detect passive markers of diameter as low as 9 mm, which can be uniquely arranged in a 3D pattern onboard a UAV for identification and tracking. For better reliability, we use 38 mm diameter markers onboard the ANCL Gen2 platforms. The cameras are connected to the lab network using two switches which also provide power to the cameras i.e., Power over Ethernet.

Radios

Communication radios are essential components of a UAV platform. They allow a wireless connection between the UAV and its peripherals. The first radio is a Spektrum receiver and transmitter pair. The transmitter serves as the remote control for the quadrotor. Its channels are programmed in the autopilot firmware to operate the quadrotor in specific modes and fly manually. This is essential from a safety perspective and allows the user to intervene immediately if an accident is about to happen. In ANCL, we use Spektrum DX8 remote transmitter. This remote control has safety functions and therefore has the highest priory and uses the most sophisticated equipment.

The second radio is a Roving Networks ESP8266 802.11b radio, which essentially serves as a Wi-Fi Adapter for the flight controller computer, connecting it to the Wi-Fi Network of the QGroundControl PC. This allows us to wirelessly control the UAV and monitor the flight data online during a flight.

Motors and Propellers

Electric-powered actuators have become favourable for UAVs due to their high power-to-weight ratio compared to gas-powered internal combustion engines. These electric actuators consist of a motor-propeller pair. In a quadrotor UAV, there are four motor-propeller pairs. The motor is a brushless DC outrunner with three inputs. The motor speed and torque depend upon the voltage levels and phase shift between the input pulses. The motor speeds are rated in revolutions per minute (RPM) per volt at no load, often referred to as Kv rating. The motors used in ANCL Gen2 are T-MOTOR P2207 V3.0 with a Kv rating of 1750 RPM per volt. Their rated voltage is 7.4 - 16.8V, and their maximum current specification is 38.4 A. The propellers used in ANCL Gen2 is Gemfan Freestyle 6030-3 with 6-inch diameter and a pitch of 3 inches. A propeller pitch is defined as the measure of forward distance moved through a soft solid due to one complete rotation of the propeller.

Electronic Speed Controllers

The ESC used in the quadrotor receive a PWM signal from the flight controller and produce DC pulses at its three outputs. The ESCs vary the magnitude and phase of these pulses with the variation in the PWM input to control the speed of brushless DC motors. The ESCs used in ANCL Gen2 are BLHeli-S 20A based on EFM8BB21F16G micro control unit (MCU) 8-bit C8051 core with 50 MHz maximum operating frequency. BLHeli-S is an open source hardware and BLHeli-S 20A used the version A-H-30. It supports Dshot150, Dshot300 and Dshot600. Dshot is digital signal, anti-interference ability is stronger and do not need throttle calibration.

Onboard Computer

The onboard computer is UP Core computer with Intel Atom x5-z8350 (4 cores up to 1.92 GHz), 4GB memory and 64GB eMMC storage. It support off-the-shelf Ubuntu operating system. We choose Ubuntu 20.04 LTS as the OS onboard. The UP Core computer connected to Pixhawk 6C by universal asynchronous receiver-transmitter (UART) connection. The onboard PC has two USB 2.0 and one USB 3.0 ports for peripheral device. As 2.4G WiFi channel is fully congested by the university WiFi broadcast, we used a Netgear wireless AC1200 Wi-Fi dual band USB adapter to connect onboard computer to local WiFi network to get the Vicon datastream. As the Vicon datastream provide the absolute position and yaw angle of the ANCL Gen2, which is essential to a stable flight, it is important to test the WiFi connection before taking off.

2.3.2 Software Components

Autopilot Firmware

The flight controller's autopilot firmware is responsible for a multitude of tasks. It operates drivers for a range of sensors and gathers data from these sensors. Utilizing estimation algorithms, it calculates the UAV's state, communicates with external sensors, and processes user commands, including the ability to switch between different flight modes. Additionally, control algorithms within the autopilot generate inputs that are transformed into PWM signals. These signals are then relayed to the ESCs, which in turn drive the UAV's motors and propellers. Beyond these functions, the autopilot is also tasked with numerous safety and control-related operations.

At ANCL, version 1.13.3 release of PX4 [117] is currently used. PX4 is based on a publish-subscribe structure and has several built-in applications for various UAV models. PX4's primary programming language is C++. It also provides many libraries and tools for developers to implement their algorithms. PX4 uses a MAVLINK protocol [126] for its communication with onboard components e.g., CVS as well as off-board components e.g., Ground station.

The ANCL Gen2 platform fully utilizes the off-board control capabilities of the PX4 Autopilot. Thus, most outer-loop controller can be implemented on UP Core computer using ROS. This improves the implementation efficiency.

PX4 computes the attitude using the `attitude_estimator_q` module, which is a quaternion-based attitude estimator and fuses raw IMU measurements. The IMU sensors are initially calibrated using the QGroundControl software. Attitude estimates are provided at 250 Hz.

Ground Station Software

QGroundControl (QGC) is a ground station software for MAVLINK enabled UAVs, which provides graphical user interface (GUI) based user-friendly option for complete setup and configuration of PX4 autopilot. It also provides in-flight support and mission planning. The Q-Ground Control displays flight maps and UAV trajectories. The QGC can be used to modify UAV parameters, switch flight modes, download flight data, monitor instruments online during flight, calibrate UAV sensors, configure control, and several other functions to support UAV flight. QGC software is a valuable tool for tuning and monitoring. ANCL Gen2 is compatible with the latest version of QGroundControl and should be compatible to future releases.

ESC Firmware

ESC firmware runs inside the ESC microcontroller and, depending upon the PWM input, generates an output signal to provide electronic commutation to the brushless DC motor. The most commonly used open-source firmwares for UAV ESCs are BLHeli [127].

Offboard Control System

The computer vision system's software component utilizes the Robot Operating System (ROS) running on an UP Core computer with Ubuntu 20.04 as the operating system. ROS is a versatile, open-source framework for robotics, essentially acting as a meta-operating system layered on top of an existing OS. It offers an array of tools, libraries, and conventions aimed at easing the task of developing software for robots. ROS provides the flexibility to work at different levels. At a lower level, it can be employed for creating drivers for sensors and actuators, enabling direct control of robotic devices. On a higher level, it is adept at handling more complex tasks such as robot mapping, localization, control, and estimation algorithms.

The architecture of ROS is inherently modular, consisting of discrete programs known as nodes, each designed for specific functions. These nodes communicate via a process called message passing, exchanging data through streams known as topics. Typically, a node will subscribe to one or more of these topics, process the incoming data, and then publish its output to another topic. For instance, a robot control node in ROS might subscribe to a topic that provides system states or sensor data. It then processes this data to generate control inputs, which are subsequently published on a different topic. A driver node would then subscribe to this latter topic to operate the UAV's actuators. For ANCL Gen2, the Noetic Ninjemys release of ROS has been employed for these purposes.

2.3.3 Maple-Matlab-SITL Simulation Pipeline

The Software-In-The-Loop (SITL) creates a virtual autopilot environment, enabling the validation of performance in a controlled yet realistic setting. This approach is critical for ensuring that the design remains resilient against real-world challenges such as controller saturation, multi-rate sampling, and computational delays. These factors could potentially compromise the performance or feasibility of theoretical designs due to limitations in processing power or memory. For our simulation framework, we have selected the open-source PX4-SITL, coupled with the Gazebo simulator [128], which utilizes the Open Dynamics Engine (ODE) for physics simulation. Gazebo was the preferred choice for its straightforward multibody model format, open-source nature, and its recommendation by the PX4 development community for SITL applications. Unlike the Matlab simulations, where the model is based directly on differential equations, the Gazebo model eschews the need for differential equation modeling, relying instead on the geometry and inertial properties of the system's components. The PX4-Gazebo SITL simulation incorporates the RotorS simulation plugins [129], which are based on a ROS/Gazebo framework. We have made the simulation source code available to the research community, enabling independent result reproduction or design modifications on standard desktop PCs.

The simulation environment's architecture comprises three main components: the Gazebo simulator, the PX4 autopilot software, and the ROS Offboard Control. Gazebo incorporates a multibody dynamics engine, a 3D graphics interface, an SLS model description file based on the 3DR Iris quadrotor, external disturbances, and RotorS-derived plugins for simulating onboard sensors such as the GPS, IMU, and magnetometer. The PX4 autopilot software is emulated to operate on a desktop environment. The ROS Offboard Control integrates our QSF and coordinate transformations, convertin g the Gazebo world frame into the navigation frame used in the QSF.

Model-based controllers such as the QSF benefit from precise theoretical statements about their stability and performance. Their mathematical derivation means they can be extended systematically to different problems. However, they can suffer from complex expressions. The QSF proposed in Chapter 3 uses Maple to compute the high-order Lie derivatives required. The resulting controller expression is very complex and must be automatically moved from Maple to C++ to avoid errors and streamline debugging. Therefore, we developed a Maple-Matlab-SITL pipeline for controller development. First, we do all symbolic calculations, including system modeling and the QSFA in Maple. Secondly, the resulting symbolic expression for the system model and controller is exported to Matlab for efficient simulation. Finally, the Matlab controller is exported to C++ for SITL using Matlab's `Coder Toolbox`. The full framework code which was used to generate the simulations in Section 3.3.1 is at `https://github.com/ANCL/SLS_PX4_SITL`.

SLS Rigid Body Dynamics

The Gazebo simulator uses a Simulation Description Format (SDF) model object to describe the SLS model. An SDF file contains information on model links, joints, collision objects, visuals, and plugins. The SDF file used in this work is at `https://github.com/ANCL/SLS_PX4_SITL/blob/ancl-sls/ancl_sls/iris_pendulum/iris_pendulum.sdf`. The model is built on top of the default 3DR IRIS model provided in the stock PX4-SITL [117]. Gazebo's internal dynamics engine is set to the ODE [130]. ODE provides a convenient way to simulate the SLS as no differential equation model such as (2.14) is needed explicitly. This should be compared to typical Matlab simulations, where simulation uses differential equations which must be derived.

The SDF file for the SLS consists of 8 links which are rigid bodies with specified mass and inertia matrix. The 8 links are denoted *base_link, imu_link, rotor1_link, rotor2_link, rotor3_link, rotor4_link, pendulum_link, load_link.* Fig. 2.6 shows a subset of these links. The SLS's links are interconnected by 7 joints of various types. Each joint has a parent and a child link. The 7 joints are *imu_joint, rotor1_joint, rotor2_joint, rotor3_joint, rotor4_joint, pendulum_joint, load_joint.*

As shown in Fig. 2.6, a Universal Joint connects the *pendulum_link* to the quadrotor *base_link.* A Fixed Joint connects the *load_link* to the *pendulum_link.* A Universal Joint provides 2 rotational DoF between the parent and child links. A full description of joint types can be found in `http://sdformat.org/spec?ver=1.6&elem=joint`. All Gazebo links must be defined as 3D rigid bodies. Thus, we define a pendulum rod with a mass 1.6 kg and inertia diag(5×10^{-5} ,8.3×10^{-4} ,8.3×10^{-4}) kgm^2 and spherical load with a mass 0.16 kg and inertia diag(4×10^{-3} ,4×10^{-3} ,4×10^{-3}) kgm^2.

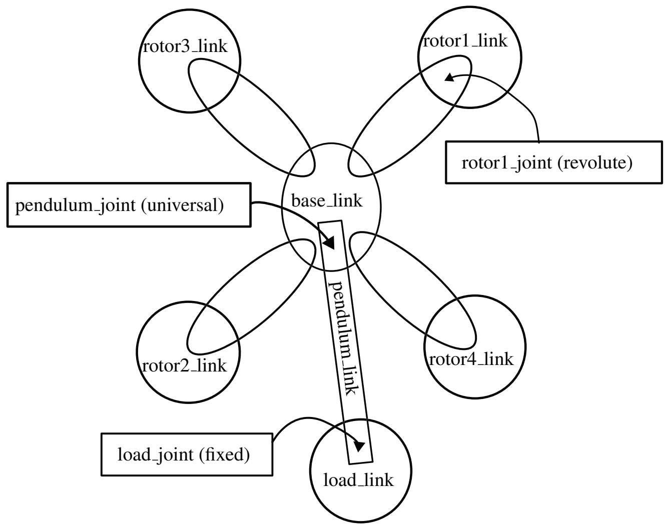


Figure 2.6: SLS SDF Structure.

Rotor Aerodynamics

The PX4-SITL environment uses RotorS [129] to model the aerodynamic forces and torques of the rotor. This modelling is based on [131], [132]. As shown in Fig. 2.7, we denote the thrust of Rotor i ($1 \leq i \leq 4$) as $\bar{u}_i$, drag force as F_{Di}, rolling moment as M_{Ri}, and rotor torque as $\bar{\tau}_i$. We have the expression

$$\bar{u}_i = -\Omega_i^2 C_T b_3 \tag{2.26a}$$

$$F_{Di} = -\Omega_i C_D (I_3 - b_3 b_3^T) v_{Ai} \tag{2.26b}$$

$$M_{Ri} = \Omega_i C_R (I_3 - b_3 b_3^T) v_{Ai} \tag{2.26c}$$

$$\bar{\tau}_i = (-1)^i C_M \bar{u}_i \tag{2.26d}$$

where $\Omega_i \geq 0$ is rotor speed, C_T is thrust constant, C_D is rotor drag constant, C_R is rolling moment constant, and C_M is the rotor torque constant. All constants are positive. The linear velocity v_{Ai} is the velocity of the ith rotor w.r.t. air velocity expressed in $\mathcal{B}$. Hence, denoting wind velocity v_W expressed in $\mathcal{N}$ we have

$$v_{Ai} = R^T (v_Q + \omega \times \ell_i - v_W) \tag{2.27}$$

where ℓ_i is the arm length from the quadrotor CoM to the rotor's axis of rotation.

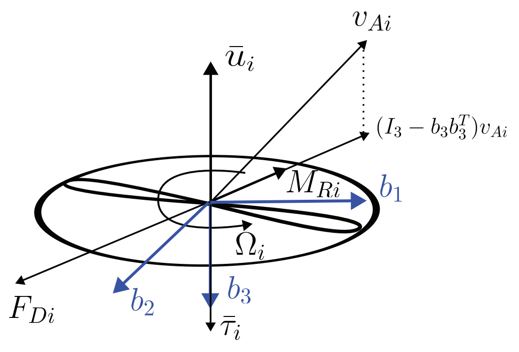


Figure 2.7: Forces and moments on a single rotor

Actuation Thrust and Torque

The output of the QSF is total thrust $\bar{u}$ and torque τ in units of N and $\mathrm{N} \cdot \mathrm{m}$, respectively. From (2.26) the relation between $[\bar{u}, \tau]$ and rotor speed Ω is

$$\begin{bmatrix} \bar{u} \\ \tau \end{bmatrix} = C_T \begin{bmatrix} 1 & 1 & 1 & 1 \\ -\ell_1 & \ell_3 & \ell_1 & -\ell_3 \\ \ell_2 & -\ell_4 & \ell_2 & -\ell_4 \\ C_M & C_M & -C_M & -C_M \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} \tag{2.28}$$

We take $C_T = 5{\times}10^{-6}\mathrm{Ns}^2$ and $C_M = 0.05\,\mathrm{m}$. The 3DR Iris has $\ell = [0.22, 0.13, 0.2, 0.13]$ m.

PX4 expects a normalized thrust $\tilde{u} \in [0,1]$ and torque $\tilde{\tau} \in [-1,1]$. Using its mixer, PX4 converts $\tilde{u}, \tilde{\tau}$ into normalized rotor speed commands $\tilde{\Omega}_i, 1 \leq i \leq 4$, using parameters describing the geometry of the multirotor's frame. In PX4, this mapping is given by $[\tilde{\Omega}_1^2, \tilde{\Omega}_2^2, \tilde{\Omega}_3^2, \tilde{\Omega}_4^2]^T = K_P[\tilde{u}, \tilde{\tau}]^T$, where K_P is given by

$$K_P = \begin{bmatrix} 1 & -0.4377 & 0.7071 & 0.9091 \\ 1 & 0.4377 & -0.7071 & 1 \\ 1 & 0.4377 & 0.7071 & -0.9091 \\ 1 & -0.4377 & -0.7071 & -1 \end{bmatrix} \tag{2.29}$$

Gazebo receives a normalized rotor speed command $[\tilde{\Omega}_1, \tilde{\Omega}_2, \tilde{\Omega}_3, \tilde{\Omega}_4]$ which it scales to obtain a rotor speed command

$$\Omega_i = C_\Omega \tilde{\Omega}_i, \quad 1 \leq i \leq 4 \tag{2.30}$$

where $C_\Omega = 1000$ rad/s is the scaling constant.

In the last step, Gazebo uses the traditional quadratic rotor model (2.26a) and (2.26d) to generate individual propeller thrust and torque from rotor speed command Ω_i. With the assumption the above modelling is known exactly, we can scale the

QSF output before feeding it to PX4 so that Gazebo applies the desired values of $\bar{u}$ and τ. Given $\bar{u}, \tau = [\tau_1, \tau_2, \tau_3]^T$ output from the QSF in (4.24) we scale $[\tilde{u}, \tilde{\tau}^T]$ as $[\bar{u}/S_1, \tau_1/S_2, \tau_2/S_3, \tau_3/S_4]$. We can summarize the above discussion with

$$\begin{bmatrix} \bar{u}_G \\ \tau_G \end{bmatrix} = C_T C_\Omega^2 \begin{bmatrix} 1 & 1 & 1 & 1 \\ -\ell_1 & \ell_3 & \ell_1 & -\ell_3 \\ \ell_2 & -\ell_4 & \ell_2 & -\ell_4 \\ C_M & C_M & -C_M & -C_M \end{bmatrix} K_P \begin{bmatrix} \bar{u}/S_1 \\ \tau_1/S_2 \\ \tau_2/S_3 \\ \tau_3/S_4 \end{bmatrix} \tag{2.31}$$

where $[\bar{u}_G, \tau_G]$ is the total thrust and torque in Gazebo. Substituting parameter values into (2.31) gives scaling parameters $S_1 = 20, S_2 = 1.8383, S_3 = 1.8383, S_4 = 0.9546$ so that $\bar{u}_G = \bar{u}, \tau_G = \tau$.

2.4 Conclusion

In this section, we present a detailed examination of UAV slung load systems and FA hexarotors. Our focus includes various frame definitions, rotation matrices, and both kinematic and dynamic modeling. We introduce a UAV slung load model that incorporates drag force for enhanced maneuverability. Additionally, the modeling of FA hexarotors, including end-effector dynamics, is explored. Constraint dynamics specific to FA hexarotors will be further discussed in Chapter 5. This section also covers the ANCL Gen2 platforms, detailing their crucial hardware components, functions, and the integration of essential software like PX4, ROS, and PX4-SITL simulation. This comprehensive approach bridges theoretical models with practical applications, underscoring the synergy between advanced research and real-world implementation in robotics.

Chapter 3

Quasi-static State Feedback Output Tracking for a Slung Load System

3.1 Quasi-Static Feedback (QSF) Linearization

We consider the control-affine dynamics

$$\dot{x} = f(x) + \sum_{i=1}^{m} g_i(x)u_i \tag{3.1a}$$

$$y_i = h_i(x), \quad 1 \leq i \leq m \tag{3.1b}$$

with vector fields $f, g_i : \mathcal{M} \to \mathbb{R}^n$, and output functions $h_i : \mathcal{M} \to \mathbb{R}$ defined on open subset $\mathcal{M} \subseteq \mathbb{R}^n$. A QSF has the form $u = u(x, v, \dot{v}, \ddot{v}, \dots, v^{(\rho)})$, where v is an auxiliary input of dimension m. QSF assigns input v to be a time derivative of y:

$$y^{(r_i)} = v_i, \quad 1 \leq i \leq m \tag{3.2}$$

where $y^{(i)}$ denotes the ith time derivative of y. To achieve output tracking, v is taken to be a linear function of the output tracking error and its time derivatives. The goal of QSF is to statically linearize flat systems that do not satisfy the conditions for static state feedback linearization. Compared with dynamic state feedback, QSF is a static function of state, i.e., it requires no state augmentation. Having a simpler controller structure is practically important when implementing the autopilot on-board where computing resources are limited. In Section 3.1.1 we present the QSFA which generates a linearizing QSF for a flat system of the form (3.1). Section 3.1.2 describes the application of the QSFA to the SLS.

3.1.1 Quasi-Static Feedback Algorithm (QSFA)

The Lie derivative of a function $\lambda : \mathcal{M} \to \mathbb{R}$ along the vector field f is defined by $L_f\lambda(x) = \frac{\partial\lambda}{\partial x} f(x)$. The QSFA makes use of a vector of indices $r = [r_1, \dots r_m]$ where r_i is the largest integer satisfying $L_{g_j} L_f^k h_i(x) = 0, 1 \le j \le m, k < r_i - 1$, about some $x_0 \in \mathcal{M}$. The existence of these indices does not imply a well-defined relative degree since that requires the decoupling matrix

$$D(x) = \begin{bmatrix} L_{g_1} L_f^{r_1-1} h_1(x) & \dots & L_{g_m} L_f^{r_1-1} h_1(x) \\ \vdots & \dots & \vdots \\ L_{g_1} L_f^{r_m-1} h_m(x) & \dots & L_{g_m} L_f^{r_m-1} h_m(x) \end{bmatrix} \tag{3.3}$$

is nonsingular at x_0.

Variables have superscript $^{\langle i \rangle}$ to keep track of the algorithm iteration. We begin the QSFA at Step 0 and assume the decoupling matrix $D^{\langle 0 \rangle} = D$ given by (3.3) has a constant rank less than m about $x_0 \in \mathcal{M}$ where $r^{\langle 0 \rangle} = [r_1^{\langle 0 \rangle}, \dots, r_m^{\langle 0 \rangle}] = r$. Define $s^{\langle i \rangle} = \text{rank}(D^{\langle i \rangle}(x_0))$, and $y^{\langle 0 \rangle} = y^{(r^{\langle 0 \rangle})} = [y_1^{(r_1^{\langle 0 \rangle})}, \dots, y_m^{(r_m^{\langle 0 \rangle})}]^T$.

Step 0**:** From the definition of $r^{\langle 0 \rangle}$ we can express $y^{(r^{\langle 0 \rangle})}$ as

$$y^{\langle 0 \rangle} = y^{(r^{\langle 0 \rangle})} = a_0(x) + D^{\langle 0 \rangle} u \tag{3.4}$$

We reorder and decompose $y^{\langle 0 \rangle}$ as

$$y^{\langle 0 \rangle} = \begin{bmatrix} \tilde{y}^{\langle 0 \rangle} \\ \hat{y}^{\langle 0 \rangle} \end{bmatrix} \tag{3.5}$$

where $\tilde{y}^{\langle 0 \rangle}$ corresponds to the first $s^{\langle 0 \rangle}$ independent rows of $D^{\langle 0 \rangle}$, and $\hat{y}^{\langle 0 \rangle}$ corresponds to the dependent rows of $D^{\langle 0 \rangle}$. We introduce auxiliary input $v_1 = \tilde{y}^{\langle 0 \rangle}$. Since the last $m - s^{\langle 0 \rangle}$ rows of $D^{\langle 0 \rangle}$ are linearly dependent on the first $s^{\langle 0 \rangle}$ rows, we have

$$\tilde{y}^{\langle 0 \rangle} = \tilde{a}_0(x) + \tilde{b}_0(x)u = v_1 \tag{3.6}$$

$$\hat{y}^{\langle 0 \rangle} = \hat{y}^{\langle 0 \rangle}(x, v_1) \tag{3.7}$$

where $\hat{y}^{\langle 0 \rangle}$ is affine in v_1 and $\tilde{b}_0(x)$ are the first $s^{\langle 0 \rangle}$ independent rows of $D^{\langle 0 \rangle}$. We remark that in (3.7) we have eliminated u using (3.6). This ensures time derivatives of u do not appear $\dot{\hat{y}}^{\langle 0 \rangle}$ below and we have a static relation between u and v and its time derivatives.

Next, we compute index $r^{\langle 1 \rangle}$ using the definition of r but with output $\hat{y}^{\langle 0 \rangle}$. We have

$$\begin{aligned}\dot{\hat{y}}^{\langle 0\rangle} = \hat{y}^{(r^{\langle 0\rangle}+1)} &= \frac{\partial \hat{y}^{\langle 0\rangle}}{\partial x}[f(x) + \sum_{i=1}^{m} g_i(x)u_i] + \frac{\partial \hat{y}^{\langle 0\rangle}}{\partial v_1}\dot{v}_1 \qquad (3.8)\\ &= L_f\hat{y}^{\langle 0\rangle} + \frac{\partial \hat{y}^{\langle 0\rangle}}{\partial v_1}\dot{v}_1 + \sum_{i=1}^{m} L_{g_i}\hat{y}^{\langle 0\rangle}u_i\\ &= a_1(x,\dot{v}_1) + b_1(x)u \qquad (3.9)\end{aligned}$$

where

$$b_1 = \begin{bmatrix} L_{g_1}\hat{y}_1^{\langle 0\rangle} & \dots & L_{g_m}\hat{y}_1^{\langle 0\rangle} \\ \vdots & \dots & \vdots \\ L_{g_1}\hat{y}_{m-s^{\langle 0\rangle}}^{\langle 0\rangle} & \dots & L_{g_m}\hat{y}_{m-s^{\langle 0\rangle}}^{\langle 0\rangle} \end{bmatrix} \in \mathbb{R}^{(m-s^{\langle 0\rangle})\times m} \qquad (3.10)$$

Three cases are possible depending on the value of b_1. If b_1 is identically zero, we must continue taking time derivatives of $\dot{\hat{y}}^{\langle 0\rangle}$. If any row of b_1 is linearly dependent on the rows of $\tilde{b}_0(x)$ in (3.6), it should be expressed as a function of (x, v_1) as in (3.7), and then time derivatives of these rows are computed. If all rows of b_1 are linearly independent of the rows of $\tilde{b}_0(x)$ in (3.6), we define $r^{\langle 1\rangle} = [r_1^{\langle 1\rangle}, \dots, r_{m-s^{\langle 0\rangle}}^{\langle 1\rangle}]$, so that $y^{\langle 1\rangle} = (\hat{y}^{\langle 0\rangle})^{(r^{\langle 1\rangle})}$. Every row of the corresponding decoupling matrix $D^{\langle 1\rangle}$ is linearly independent of the rows of $\tilde{b}_0$. We decompose $y^{\langle 1\rangle}$ as

$$y^{\langle 1\rangle} = \begin{bmatrix} \tilde{y}^{\langle 1\rangle} \\ \hat{y}^{\langle 1\rangle} \end{bmatrix} \qquad (3.11)$$

where $\tilde{y}^{\langle 1\rangle}$ corresponds to the first $s^{\langle 1\rangle}$ independent rows of $D^{\langle 1\rangle}$. We introduce auxiliary input $v_2 = \tilde{y}^{\langle 1\rangle}$.

Similar to (3.6) and (3.7), $y^{\langle 1\rangle}$ can be written as

$$\tilde{y}^{\langle 1\rangle} = \tilde{a}_1(x, v_1, \dots, v_1^{(r^{\langle 1\rangle})}) + \tilde{b}_1(x, v_1, \dots, v_1^{(r^{\langle 1\rangle}-1)})u = v_2 \qquad (3.12)$$

$$\hat{y}^{\langle 1\rangle} = \hat{y}^{\langle 1\rangle}(x, v_1, \dots, v_1^{(r^{\langle 1\rangle})}, v_2) \qquad (3.13)$$

As shown in (3.12), $\tilde{a}_1, \tilde{b}_1$ and $\hat{y}^{\langle 1\rangle}$ are functions of x and auxiliary input v. To track this dependence conveniently, we introduce set S_n^T which contains components of the auxiliary input and their time derivatives. The largest component of auxillary input is denoted n and $T = [T_1, \dots, T_n] \in \mathbb{N}^n$ is the highest order of derivatives for each input component. That is,

$$S_n^T = \{v_1, \dots, v_1^{(T_1)}, v_2, \dots, v_2^{(T_2)}, \dots, v_n, \dots, v_n^{(T_n)}\} \qquad (3.14)$$

Hence, (3.12) and (3.13) can be written more compactly as

$$\tilde{y}^{\langle 1\rangle} = \tilde{a}_1(x, S_1^{T^{\langle 1\rangle}}) + \tilde{b}_1(x, S_1^{T^{\langle 1\rangle}-1})u = v_2 \tag{3.15}$$

$$\hat{y}^{\langle 1\rangle} = \hat{y}^{\langle 1\rangle}(x, S_1^{T^{\langle 1\rangle}}, v_2) \tag{3.16}$$

Step k. Suppose that in Steps 0 through k, $y^{\langle 0\rangle}, \dots, y^{\langle k\rangle}$ have been defined so that

$$\begin{aligned}
\tilde{y}^{\langle 0\rangle} &= \tilde{a}_0(x) + \tilde{b}_0(x)u \\
&\vdots \\
\tilde{y}^{\langle k\rangle} &= \tilde{a}_k(x, S_k^{T^{\langle k\rangle}}) + \tilde{b}_k(x, S_k^{T^{\langle k\rangle}-1})u = v_{k+2} \\
\hat{y}^{\langle k\rangle} &= \hat{y}_k^{\langle k\rangle}(x, S_k^{T^{\langle k\rangle}}, v_{k+1})
\end{aligned}$$

where $\hat{y}^{\langle k\rangle}$ is affine in v_{k+1}. Define $y^{\langle k+1\rangle} = (\hat{y}^{\langle k\rangle})^{(r^{\langle k+1\rangle})}$ which can be decomposed as before:

$$y^{\langle k+1\rangle} = \begin{bmatrix} \tilde{y}^{\langle k+1\rangle} \\ \hat{y}^{\langle k+1\rangle} \end{bmatrix} \tag{3.17}$$

where $\tilde{y}^{\langle k+1\rangle}$ are the first $s^{\langle k+1\rangle}$ independent rows of $D^{\langle k+1\rangle}$. Introducing auxiliary inputs $v_{k+2} = \tilde{y}^{\langle k+1\rangle}$, (3.17) can be written as

$$\begin{aligned}
\tilde{y}^{\langle k+1\rangle} &= \tilde{a}_{k+1}(x, S_{k+1}^{T^{\langle k+1\rangle}}) + \tilde{b}_k(x, S_{k+1}^{T^{\langle k+1\rangle}-1})u \\
\hat{y}^{\langle k+1\rangle} &= \hat{y}_{k+1}^{\langle k\rangle}(x, S_{k+1}^{T^{\langle k+1\rangle}}, v_{k+2})
\end{aligned}$$

If $s^{\langle 0\rangle} + s^{\langle 1\rangle} + \cdots + s^{\langle k+1\rangle} = m$, the algorithm terminates. Otherwise, we take the time derivative of $\hat{y}^{\langle k+1\rangle}$ and perform the next iteration.

Defining $[\tilde{y}^{\langle 0\rangle}, \dots, \tilde{y}^{\langle k+1\rangle}]^T = [v_1, \dots, v_{k+2}]^T = v \in \mathbb{R}^m$, and when the algorithm terminates, we have an invertible relation between the original input u and auxiliary input v:

$$v = \begin{bmatrix} \tilde{a}_0(x) \\ \vdots \\ \tilde{a}_{k+1}(x, S_{k+1}^{T^{\langle k+1\rangle}}) \end{bmatrix} + D^\circ u \tag{3.18}$$

where D° is the final decoupling matrix with rank $(D^\circ) = m$.

3.1.2 SLS QSF

For controller design, we assume $f_e = 0$ and $\tau_e = 0$ in (2.17). This assumption is true for indoor environments. Moreover, the QSF will be augmented with integral control to compensate f_e, τ_e.

Differential Flatness

In this section, we show that the SLS dynamics including rotor drag (2.16) is differentially flat, as is the model without drag [3]. To prove the flatness property, we will show that the system state $[p_L, v_L, \alpha, \beta, \gamma_\alpha, \gamma_\beta, R, \omega]$ and inputs $[\bar{u}, \tau]$ can be written as a function of the flat output and a finite number of their derivatives.

Theorem 3.1.1. *System*

$$m_Q \dot{v}_Q = m_Q g n_3 - R\bar{u}n_3 + Tq - RDR^T v_Q \tag{3.19a}$$

$$\dot{\omega} = J^{-1}(-\omega \times J\omega + \tau - AR^T v_Q - B\omega) \tag{3.19b}$$

$$m_L \dot{v}_L = -Tq + (m_L + m_C) g n_3 \tag{3.19c}$$

has a flat output $y = [p_L^T, \psi]^T$.

Proof. The expression for Tq in terms of $\dot{y}$ is obtained immediately from (3.19c). We obtain expressions for the unit vector $q = Tq/\|Tq\|$ and tension $T = Tq \cdot q$.

To show R and $\bar{u}$ are functions of y and it derivatives, we reformulate (3.19a) as

$$m_Q g n_3 - \bar{u} b_3 + Tq - (d_x b_1^T v_Q) b_1 - (d_y b_2^T v_Q) b_2 - (d_z b_3^T v_Q) b_3 = m_Q \dot{v}_Q \tag{3.20}$$

Left multiplying (3.20) by b_1^T we have

$$b_1^T \hat{X} = 0, \text{ with } \hat{X} = m_Q g n_3 + Tq - d_x v_Q - m_Q \dot{v}_Q \tag{3.21}$$

Left multiplying (3.20) by b_2^T we have

$$b_2^T \tilde{X} = 0, \text{ with } \tilde{X} = m_Q g n_3 + Tq - d_y v_Q - m_Q \dot{v}_Q \tag{3.22}$$

For our SLS, b_1 is defined as the forward direction of the quadrotor and can be calculated from ψ as

$$b_1 = \frac{y_\psi \times \hat{X}}{\|y_\psi \times \hat{X}\|}, \text{ with } y_\psi = [-\mathrm{s}_\psi, \mathrm{c}_\psi, 0]^T \tag{3.23}$$

Since b_2 is orthogonal to both b_1 and $\tilde{X}$, we have

$$b_2 = \frac{\tilde{X} \times b_1}{\|\tilde{X} \times b_1\|} \tag{3.24}$$

Since $b_3 = b_1 \times b_2$. In the above relations for $R = [b_1, b_2, b_3]$ we express $v_Q, \dot{v}_Q$ in terms of $v_L, \dot{v}_L$ using (2.1). This yields the flatness relation for R.

The flatness relation for $\bar{u}$ comes from left multiplying (3.20) by b_3^T to get

$$\bar{u} = b_3^T(m_Q g n_3 + Tq - d_z v_Q - m_Q \dot{v}_Q) \tag{3.25}$$

and eliminating $\dot{v}_Q$.

For the flatness relation for ω, we take the derivative of (3.19a)

$$\begin{aligned} m_Q \ddot{v}_Q = &-RS(\omega)\bar{u}n_3 - R\dot{\bar{u}}n_3 + \dot{T}q + T\dot{q} \\ &- R((S(\omega)D + DS^T(\omega))R^T v_Q + DR^T \dot{v}_Q) \end{aligned} \tag{3.26}$$

Left mutiplying (3.26) by b_1^T we obtain

$$\begin{aligned} b_1^T(m_Q \ddot{v}_Q + d_x v_Q) = &-\bar{u}\omega_2 + b_1^T(\dot{T}q + T\dot{q}) - \omega_3(d_x - d_y)(b_2^T v_Q) \\ &- w_2(d_z - d_x)(b_3^T v_Q) \end{aligned} \tag{3.27}$$

Left mutiplying (3.26) by b_2^T we obtain

$$\begin{aligned} b_2^T(m_Q \ddot{v}_Q + d_y v_Q) = &\bar{u}\omega_1 + b_2^T(\dot{T}q + T\dot{q}) - \omega_3(d_x - d_y)(b_1^T v_Q) \\ &- \omega_1(d_y - d_z)(b_3^T v_Q) \end{aligned} \tag{3.28}$$

Computing $b_2^T \dot{R}$ we have

$$\omega_3 = b_2^T \dot{b}_1 \tag{3.29}$$

Combining (3.27) (3.28) and (3.29), we can solve for ω.

To compute $\dot{\omega}$ from y, we can take the time derivative of (3.27) (3.28) and (3.29). Next, three independent linear equations can be solved for $\dot{\omega}$. After obtaining $\dot{\omega}$, (3.19b) can be used to compute an expression for τ.

In the above, the expression for $\dddot{v}_Q$ in terms of y and its derivatives is needed to compute τ. Since $\dddot{v}_Q = \dddot{v}_L + Lq^{(4)}$ and recalling

$$q = \frac{Tq}{\|Tq\|} = \frac{(m_L + m_C)gn_3 - m_L \dot{v}_L}{\|(m_L + m_C)gn_3 - m_L \dot{v}_L\|} \tag{3.30}$$

$$T = \|(m_L + m_C)gn_3 - m_L \dot{v}_L\| \tag{3.31}$$

the time derivatives of T and q can be calculated from (3.19c), (3.30), and (3.31). Hence we observe $q^{(4)}$ depends on $v_L^{(5)}$ and a flatness relation for τ follows.

□

QSFA on SLS

In this section, the QSFA is applied to the SLS model (2.14) and flat output

$$y = [p_L^T, \psi]^T \tag{3.32}$$

As expected, the QSFA yields a QSF which exactly linearizes the SLS dynamics.
Step 0**.** According to (3.3), we have

$$D^{\langle 0\rangle} = \begin{bmatrix} d_{11}^{\langle 0\rangle} & 0 & 0 & 0 \\ d_{21}^{\langle 0\rangle} & 0 & 0 & 0 \\ d_{31}^{\langle 0\rangle} & 0 & 0 & 0 \\ 0 & 0 & \frac{\mathrm{s}_\phi}{J_2 \mathrm{c}_\theta} & \frac{\mathrm{c}_\phi}{J_3 \mathrm{c}_\theta} \end{bmatrix} \tag{3.33}$$

where $d_{11}^{\langle 0\rangle}, d_{21}^{\langle 0\rangle}, d_{31}^{\langle 0\rangle}$ are functions of state and we have $r^{\langle 0\rangle} = [2, 2, 2, 2]$ and $\mathrm{rank}(D^{\langle 0\rangle}) = 2$ on a subset of $\mathcal{M}$ where

$$d_{31}^{\langle 0\rangle}(x) = -\frac{[\mathrm{s}_\beta, -\mathrm{s}_\alpha \mathrm{c}_\beta, \mathrm{c}_\alpha \mathrm{c}_\beta] \cdot R n_3}{m_Q + m_L + m_C} \neq 0 \tag{3.34}$$

As in (3.4), we obtain

$$y^{\langle 0\rangle} = a_0(x) + D^{\langle 0\rangle} u \tag{3.35}$$

Hence, $y^{\langle 0\rangle}$ is decomposed as $\tilde{y}^{\langle 0\rangle} = [\ddot{y}_3, \ddot{y}_4]^T$, and $\hat{y}^{\langle 0\rangle} = [\ddot{y}_1, \ddot{y}_2]^T$. We introduce auxiliary input $v_1 = [\ddot{y}_3, \ddot{y}_4]^T$ so that

$$v_1 = \tilde{y}^{\langle 0\rangle} = \tilde{a}_0(x) + \tilde{b}_0(x) u \tag{3.36}$$

Then, $\hat{y}^{\langle 0\rangle}$ can be written as

$$\hat{y}^{\langle 0\rangle} = \begin{bmatrix} -\left(g - [1, 0]^T v_1\right) \mathrm{t}_\beta / \mathrm{c}_\alpha \\ \left(g - [1, 0]^T v_1\right) \mathrm{t}_\alpha \end{bmatrix} \tag{3.37}$$

Step 1**.** Taking a time derivative of $\hat{y}^{\langle 0\rangle}$ we obtain

$$\dot{\hat{y}}^{\langle 0\rangle} = \begin{bmatrix} \frac{\dot{v}_1 \mathrm{s}_\beta}{\mathrm{c}_\beta \mathrm{c}_\alpha^2} - \frac{\gamma_\beta (g - v_1)}{\mathrm{c}_\beta^2 \mathrm{c}_\alpha} - \frac{\gamma_\alpha \mathrm{s}_\alpha \mathrm{s}_\beta (g - v_1)}{\mathrm{c}_\beta \mathrm{c}_\alpha} \\ -\dot{v}_1 \mathrm{t}_\alpha + \frac{\gamma_\alpha (g - v_1)}{\mathrm{c}_\alpha^2} \end{bmatrix} \tag{3.38}$$

Since the input does not appear in (3.38), we take another time derivative of $\hat{y}^{\langle 0\rangle}$ to get

$$\ddot{\hat{y}}^{\langle 0\rangle} = a_1(x, v_1, \dot{v}_1, \ddot{v}_1) + b_1(x, v_1, \dot{v}_1) u \tag{3.39}$$

where $a_1(x, v_1, \dot{v}_1, \ddot{v}_1) \in \mathbb{R}^{2\times 1}$, $b_1(x, v_1, \dot{v}_1) \in \mathbb{R}^{2\times 4}$ are functions of x, auxiliary input

v, and its time derivative. Matrix b_1 has the form

$$b_1(x, v_1, \dot{v}_1) = \begin{bmatrix} b_{11} & 0 & 0 & 0 \\ b_{21} & 0 & 0 & 0 \end{bmatrix} \tag{3.40}$$

We observe that both rows of (3.40) are linearly dependent on the third row of $D^{\langle 0 \rangle}$. Hence, $\ddot{\hat{y}}^{\langle 0 \rangle}$ can be expressed using v_1. As a result, (3.39) can be written as $\ddot{\hat{y}}^{\langle 0 \rangle} = \ddot{\hat{y}}^{\langle 0 \rangle}(x, v_1, \dot{v}_1, \ddot{v}_1)$ with u eliminated. Similarly we eliminate u in $\left(\hat{y}^{\langle 0 \rangle}\right)^{(3)}$ to obtain a function $\left(\hat{y}^{\langle 0 \rangle}\right)^{(3)}(x, v_1, \dot{v}_1, \ddot{v}_1, v_1^{(3)})$. Calculating $\left(\hat{y}^{\langle 0 \rangle}\right)^{(4)}$ gives the decoupling matrix $D^{\langle 1 \rangle}$

$$D^{\langle 1 \rangle} = \begin{bmatrix} d_{11}^{\langle 1 \rangle} & d_{12}^{\langle 1 \rangle} & d_{13}^{\langle 1 \rangle} & 0 \\ d_{21}^{\langle 1 \rangle} & d_{22}^{\langle 1 \rangle} & d_{23}^{\langle 1 \rangle} & 0 \end{bmatrix} \tag{3.41}$$

where $d_{ij}^{\langle 1 \rangle}$ are functions of $(x, v_1, \dot{v}_1, \ldots, v_1^{(3)})$ with $\text{rank}(D^{\langle 1 \rangle}) = 2$ and its rows are linear independent of all rows of $D^{\langle 0 \rangle}$. Hence, $r^{\langle 1 \rangle} = [4, 4]$. Defining $v_2 = \left(\hat{y}^{\langle 0 \rangle}\right)^{(r^{\langle 1 \rangle})} = y^{\langle 1 \rangle}$, we have

$$v_2 = y^{\langle 1 \rangle} = \tilde{a}_1(x, \dot{v}_1, \ldots, v^{(4)}) + D^{\langle 1 \rangle} u \tag{3.42}$$

Combining (3.36), (3.42), we have a invertible relation between u and v

$$v = \begin{bmatrix} \tilde{a}_0(x) \\ \tilde{a}_1(x, v_1, \dot{v}_1, \ldots, v_1^{(4)}) \end{bmatrix} + D^{\circ} u \tag{3.43}$$

where

$$D^{\circ} = \begin{bmatrix} d_{31}^{\langle 0 \rangle} & 0 & 0 & 0 \\ 0 & 0 & \frac{s_\phi}{J_2 c_\theta} & \frac{c_\phi}{J_3 c_\theta} \\ d_{11}^{\langle 1 \rangle} & d_{12}^{\langle 1 \rangle} & d_{13}^{\langle 1 \rangle} & 0 \\ d_{21}^{\langle 1 \rangle} & d_{22}^{\langle 1 \rangle} & d_{23}^{\langle 1 \rangle} & 0 \end{bmatrix} \tag{3.44}$$

is the final decoupling matrix with rank $(D^{\circ}) = 4$ on a suitable subset of $\mathcal{M}$.

3.1.3 SLS Output Tracking using QSF

By setting $v = [v_1, v_2]^T = [y_3^{(2)}, y_4^{(2)}, y_1^{(6)}, y_2^{(6)}]^T$ and using (4.24), a linearizing QSF is obtained. The tracking error is defined as

$$\begin{aligned} \tilde{z} = [\tilde{z}_1, \ldots, \tilde{z}_{16}] = [&y_1 - y_{d1}, \ldots, y_1^{(5)} - y_{d1}^{(5)}, y_2 - y_{d2}, \ldots, y_2^{(5)} - y_{d2}^{(5)}, \\ &y_3 - y_{d3}, \dot{y}_3 - \dot{y}_{d3}, y_4 - y_{d4}, \dot{y}_4 - \dot{y}_{d4}]^T \end{aligned} \tag{3.45}$$

where $y_{di}, 1 \leq i \leq 4$ are desired outputs. Variables $y_1^{(3)}, \ldots, y_1^{(5)}$, and $y_2^{(3)}, \ldots, y_2^{(5)}$ are functions of $x, v_1, \dot{v}_1, \ldots, v_1^{(4)}$, while the remaining outputs and their derivatives in (4.27) can be expressed as a function of x. We integrate the tracking error $C\tilde{z} = [\tilde{z}_1, \tilde{z}_7, \tilde{z}_{13}, \tilde{z}_{15}]$.

We can express the dynamics in the $\tilde{z}$-coordinates as

$$\dot{\tilde{z}} = A_c\tilde{z} + B_c v, \ \text{with} \ v = \tilde{a} + D^\circ u \tag{3.46}$$

$$\dot{\sigma} = C\tilde{z} \tag{3.47}$$

where

$$\begin{aligned} A_c &= \text{blockdiag}(A_1, A_2, A_3, A_4) \\ B_c &= \begin{bmatrix} e_6 & e_{12} & e_{14} & e_{16} \end{bmatrix} \\ C &= \begin{bmatrix} e_1 & e_7 & e_{13} & e_{15} \end{bmatrix} \\ \tilde{a} &= [\tilde{a}_0(x), \tilde{a}_1(x, v_1, \dot{v}_1, \dots, v_1^{(4)})]^T \end{aligned}$$

$e_i \in \mathbb{R}^{16}$ denotes the ith unit vector, and

$$A_1 = A_2 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \ \text{and} \ A_j = \begin{bmatrix} 0_{5\times1} & I_5 \\ 0 & 0_{1\times5} \end{bmatrix} \in \mathbb{R}^{6\times6}$$

with $j = 3, 4$. Applying the linearizing control $u = D^{\circ-1}(K\tilde{z} + K_\sigma\sigma - \tilde{a} + y_d^{(\bar{r})})$ with $y_d^{(\bar{r})} = [y_{d1}^{(6)}, y_{d2}^{(6)}, y_{d3}^{(2)}, y_{d4}^{(2)}]^T$ to (3.46) we obtain

$$\dot{\tilde{z}} = (A_c + B_c K)\tilde{z} + B_c K_\sigma \sigma \tag{3.48}$$

$$\dot{\sigma} = C\tilde{z} \tag{3.49}$$

where $K \in \mathbb{R}^{4\times16}, K_\sigma \in \mathbb{R}^{4\times4}$ are a control gain chosen so that $A_c + B_c K$ is Hurwitz and transient error tracking is satisfactory. Because $\dot{v}_1, \dots, v_1^{(4)}$ are calculated using $y_3 - y_{d3}, \dot{y}_3 - \dot{y}_{d3}, y_4 - y_{d4}, \dot{y}_4 - \dot{y}_{d4}$ and their time derivatives, the controller depends only on x and the reference trajectory. Hence, it is a static state feedback. The expressions for $v_1, \dot{v}_1, \ddot{v}_1$ are

$$v_1 = [\ddot{y}_{d3}, \ddot{y}_{d4}]^T - k_1[\tilde{z}_{13}, \tilde{z}_{15}]^T - k_2[\tilde{z}_{14}, \tilde{z}_{16}]^T \tag{3.50a}$$

$$\dot{v}_1 = [y_{d3}^{(3)}, y_{d4}^{(3)}]^T - k_1[\tilde{z}_{14}, \tilde{z}_{16}]^T - k_2(v_1 - [\ddot{y}_{d3}, \ddot{y}_{d4}]^T) \tag{3.50b}$$

$$\ddot{v}_1 = [y_{d3}^{(4)}, y_{d4}^{(4)}]^T - k_1(v_1 - [\ddot{y}_{d3}, \ddot{y}_{d4}]^T) - k_2(\dot{v}_1 - [y_{d3}^{(3)}, y_{d4}^{(3)}]^T) \tag{3.50c}$$

where k_1, k_2 are control gains from K. Similar expressions can be obtained for $v_1^{(3)}, v_1^{(4)}$.

3.1.4 QSF Domain

In this section, we describe the domain where the QSF is well-defined. The control is singular at $\theta = \pm 90^\circ$ and $\beta = \pm 90^\circ$ as these model singularities which occur in (2.14) due to Euler angles. At points where the Jacobian matrix of the $\tilde{z}$-coordinates is singular, the QSF cannot be evaluated. These points can be obtained from the singularities of D° in (3.44). We obtain

$$\phi = \pm 90^\circ \tag{3.51a}$$

$$[\mathrm{s}_\beta, -\mathrm{s}_\alpha \mathrm{c}_\beta, \mathrm{c}_\alpha \mathrm{c}_\beta] \cdot R n_3 = 0 \tag{3.51b}$$

$$\ddot{p}_3 = g - \frac{\mathrm{c}_\alpha \mathrm{c}_\beta (\gamma_\alpha^2 \mathrm{c}_\beta^2 + \gamma_\beta^2) L m}{m_Q + m_L} \tag{3.51c}$$

$$\ddot{p}_3 = g \tag{3.51d}$$

The LHS of (3.51b) (which from (3.34) is d_{31} scaled) can be geometrically interpreted as the inner product of q and the direction of thrust Rn_3. Thus, when the pendulum is perpendicular to b_3, a physical singularity occurs. Condition (3.51c) corresponds to $\bar{u} = 0$. When the pendulum's downward linear acceleration $\ddot{p}_3 = g$, we obtain (3.51d). This is another physical singularity that appears in drone motion control. Based on the above, we can conclude that the controller's domain is a practical subset of $\mathcal{M}$ since the above points are atypical of SLS operation.

3.2 Matlab Simulation

In this section, Matlab simulations validate QSF performance. Simulations demonstrate the importance of compensating for drag force/torque, and disturbance force/torque. We consider simulations for output stabilization and time-varying output tracking and compare performance with the geometric control (GC) in [3]. The model parameters used are in Table 4.1. The drag force coefficient D is from [123]. The values of drag moments coefficients A, B are from [115].

3.2.1 Stabilization Performance Comparison

This section considers the stabilization of the SLS at $x = 0$ with no external disturbance. Since the error dynamics of the QSF (3.48) is LTI, designing transient performance for y is straightforward. We use an LQR gain with $Q = \text{blockdiag}\,(Q_1, Q_2, Q_3, Q_4), R = 0.1 \cdot I_4$, where $Q_1 = Q_2 = \text{diag}\,(100, 100, 100, 1, 1, 1)$, $Q_3 = Q_4 = I_2$. The integral control gains are $K_\sigma = [2, 2, 0.5, 0.5]$. We denote QSF_0 as the QSF with integral control and drag compensation, QSF_1 is without integral control but with drag compensation, and QSF_2 is without either integral control or

Table 3.1: Model parameters.

m	$1.6\,\mathrm{kg}$
m_p	$0.16\,\mathrm{kg}$
m_c	$0.05\,\mathrm{kg}$
L	$1\,\mathrm{m}$
J_1	$0.03\,\mathrm{kg}\cdot\mathrm{m}^2$
J_2	$0.03\,\mathrm{kg}\cdot\mathrm{m}^2$
J_3	$0.05\,\mathrm{kg}\cdot\mathrm{m}^2$
D	$\mathrm{diag}(0.49, 0.24, 0)\ \mathrm{s}^{-1}$
A	$S([0.01, 0, 0]^T)\ \mathrm{N}\cdot\mathrm{s}$
B	$\mathrm{diag}(0.9, 0.9, 0.01)\mathrm{N}\cdot\mathrm{s}\cdot\mathrm{m}$

drag compensation. This notation is used throughout Sections 4 and 5. Due to the three-loop nested controller design and the nonlinear error dynamics of the GC, its gain tuning is challenging, and we resorted to trial and error in an effort to match the transient performance of the QSF.

We consider two initial conditions with nonzero position $p_L(0) = [p_{x0}, p_{y0}, p_{z0}]^T\mathrm{m}$ and remaining states zero (i.e., the SLS is at rest and yaw is zero). We observe from Fig. 3.1, where $p_L(0) = [2, 2, 2]^T$, and Fig. 3.3, where $p_L(0) = [-0.5, 1.5, 0]^T$, that for the same control gains, the QSF maintains good transient performance for different initial condition. This convergence is expected, given the LTI error dynamics. On the other hand, the GC's transient performance varies dramatically with initial condition. That is, in Fig. 3.3, the GC has a large $0.8\,\mathrm{m}$ peak deviation from its setpoint for p_{L3}. In Fig. 3.1 the peak errors are smaller, but trajectories of the GC are oscillatory relative to the QSF. For both initial condition, the GC's yaw transient response is erratic. Fig. 3.2 shows the control input $\bar{u}, \tau$ for initial condition $p_L(0) = [2, 2, 2]^T$. The control input of the QSF is smooth and bounded to practical levels, while the GC's input has large amplitudes and high-frequency transients. Comparing QSF_0,1,2 we observe that drag compensation provides a similar transient response since in hover no drag force is present. Further, integral action does not affect the response since no disturbance is present.

Next, we study the effects of constant disturbance on the QSF and GC for a stabilization task. The setpoint is $x = 0$, external force disturbance $f_e = [5, 5, 10]^T\mathrm{N}$, and torque disturbance $\tau_e = [1, 0, 0]^T\mathrm{N}\cdot\mathrm{m}$. The IC $p_L(0) = [2, 2, 2]^T$ m and the remaining states are zero. Fig. 3.4 shows the output response. We observe that the QSF_0, thanks to integral action, is the only controller which converges to the setpoint. QSF_1 and QSF_2 do not have integration and have constant steady-state tracking error. The GC response is highly oscillatory and not convergent. Fig. 3.5 shows the corresponding control input. As in the previous simulation, drag compensation in the QSF provides a similar transient response as there is drag in

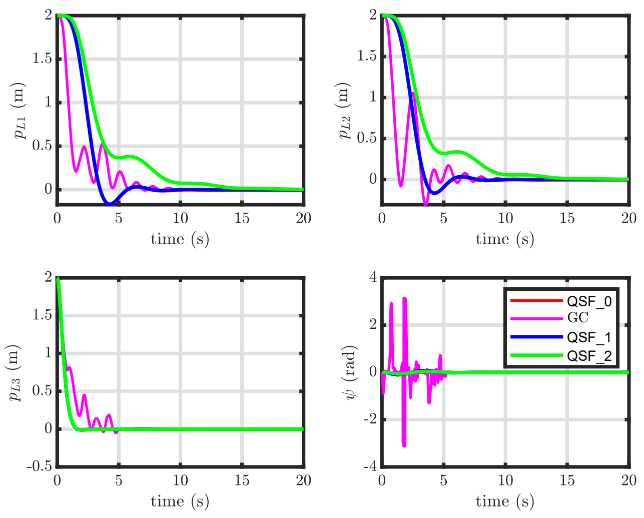


Figure 3.1: Stabilization with IC $p_L(0) = [2, 2, 2]^T$ m: System states p_L, ψ. No disturbance.

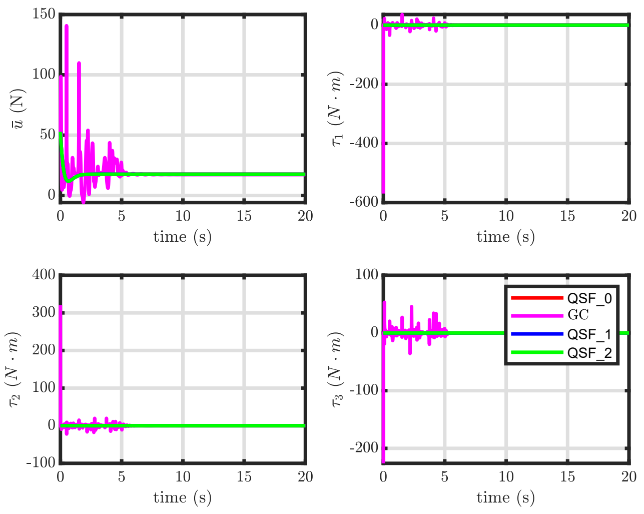


Figure 3.2: Stabilization with IC $p_L(0) = [2, 2, 2]^T$ m: Control input $\bar{u}, \tau$. No disturbance.

hover.

3.2.2 Trajectory Tracking Performance Comparison

The proposed QSF is capable of tracking complex reference trajectories. We consider a "figure-8" reference

$$y_d(t) = \begin{bmatrix} 3\sin(\pi t/8) \text{ m} \\ 1.5\sin(\pi t/4) \text{ m} \\ 0.5\sin(\pi t/8) - 9 \text{ m} \\ 0.02t \text{ rad} \end{bmatrix} \tag{3.52}$$

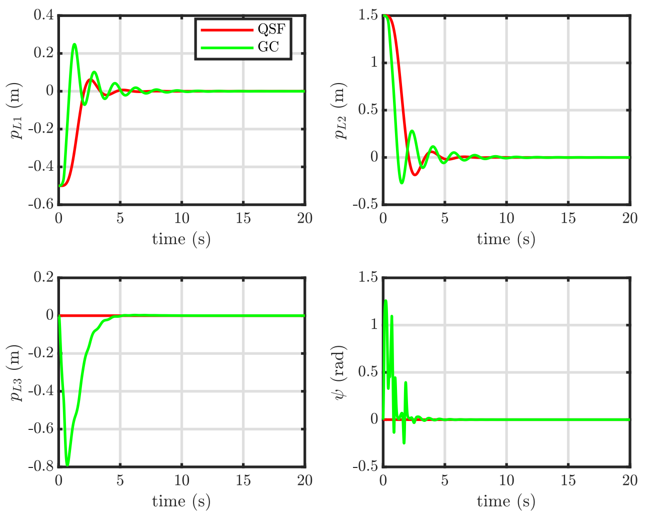


Figure 3.3: Stabilization with IC $p_L(0) = [-0.5, 1.5, 0]^T$ m: System states p_L, ψ. No disturbance.

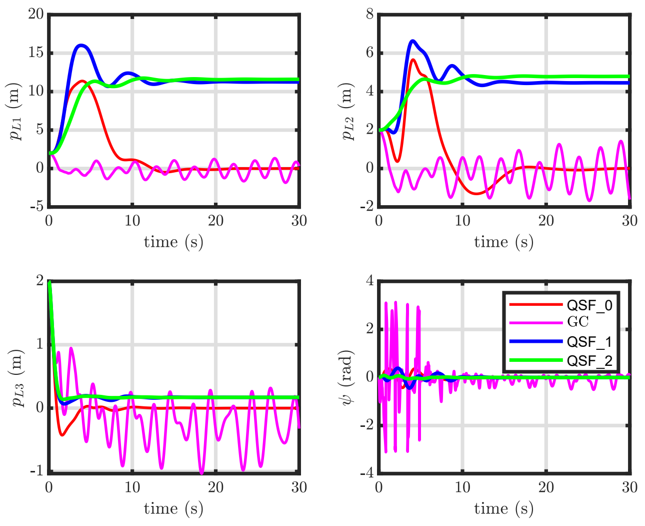


Figure 3.4: Stabilization with IC $p_L(0) = [2, 2, 2]^T$ m: System states p_L, ψ. Constant disturbance.

Initially, no disturbance is considered. The gain for the QSF was obtained using LQR with $Q = \text{blockdiag}(Q_1, Q_2, Q_3, Q_4), R = 0.1 \cdot I_4$, where $Q_1 = Q_2 = \text{diag}(100, 100, 100, 1, 1, 1), Q_3 = Q_4 = I_2$. The integral control gains are $K_\sigma = [500, 500, 30, 30]$.

Tracking error is shown in Fig. 3.6, and configuration variables are in Fig. 3.7. The input trajectories are in Fig. 3.8. We observe that for QSF_0 and QSF_1 the tracking error converges to 0 with an acceptable transient. Further, QSF_2 has large steady-state tracking error in p_{L1} and p_{L2}. Hence, we conclude the LTI error dynamics of QSF_0 and QSF_1, which include rotor drag compensation, provide accurate high-performance time-varying trajectory tracking. The importance of

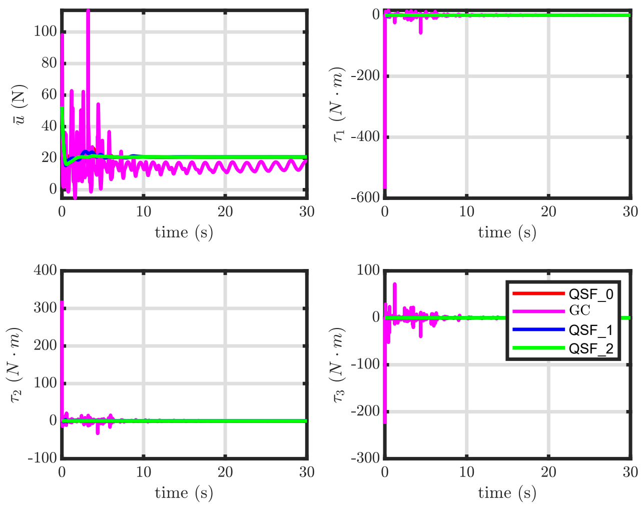


Figure 3.5: Stabilization with IC $p_L(0) = [2, 2, 2]^T$ m: Control input $\bar{u}, \tau$. Constant disturbance.

drag compensation is consistent with recent results [20], [21]. Input trajectories remain within a practical range for all QSF controllers. The GC trajectories have oscillation or diverge due to unmodelled rotor drag.

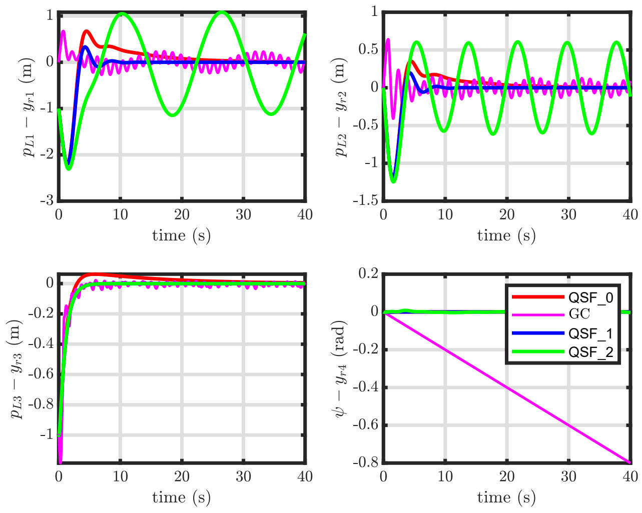


Figure 3.6: Output tracking: tracking errors for p_L, ψ. No disturbance.

Finally we test the QSF and GC for trajectory tracking in the presence of constant disturbance. Control gains are the same as in the previous simulation. Tracking error is shown in Fig. 3.9, and the configuration variables are in Fig. 3.10. The input trajectories are in Fig. 3.11. We observe that tracking error of QSF_0 converges to a practically small bounded error, while QSF_1, which has no integration, has large steady-state tracking errors. The trajectories for QSF_2 and GC do not converge and exhibit dangerous oscillations and divergence. Hence, we conclude

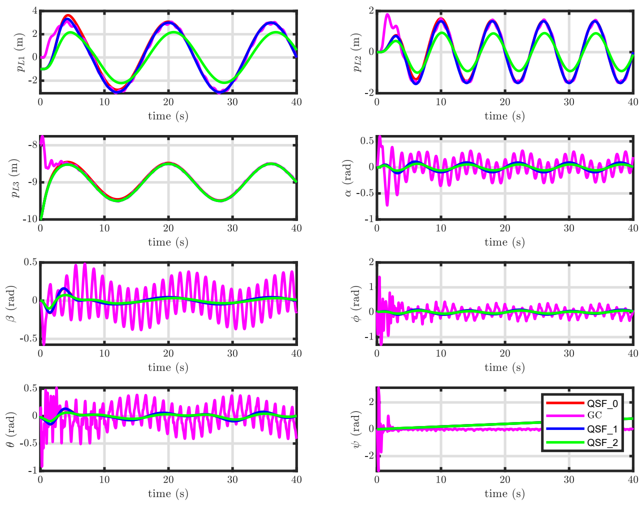


Figure 3.7: Output tracking: System states p, α, β, η. No disturbance.

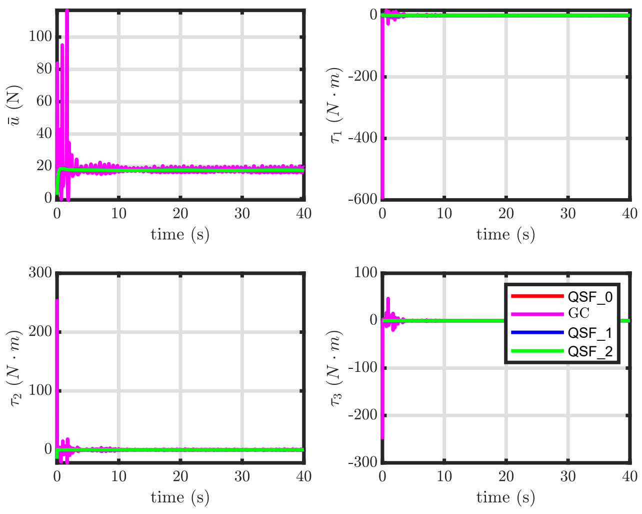


Figure 3.8: Output tracking: Control inputs $\bar{u}, \tau$. No disturbance.

integral action and drag compensation are important factors to ensure trajectory tracking performance in the presence of disturbance.

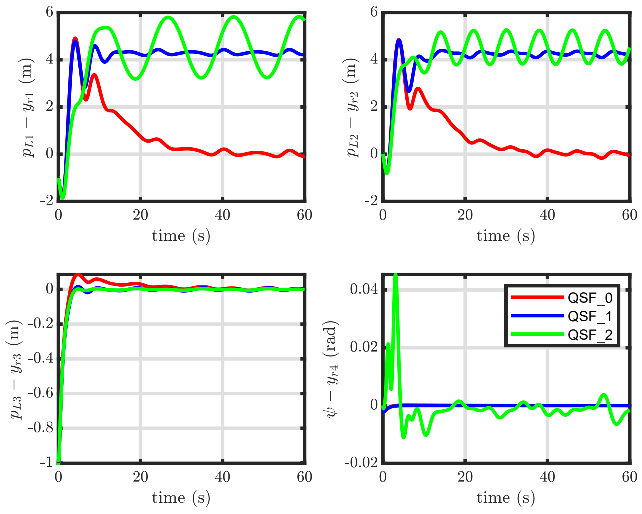


Figure 3.9: Output tracking: tracking errors for p_L, ψ. Constant disturbance.

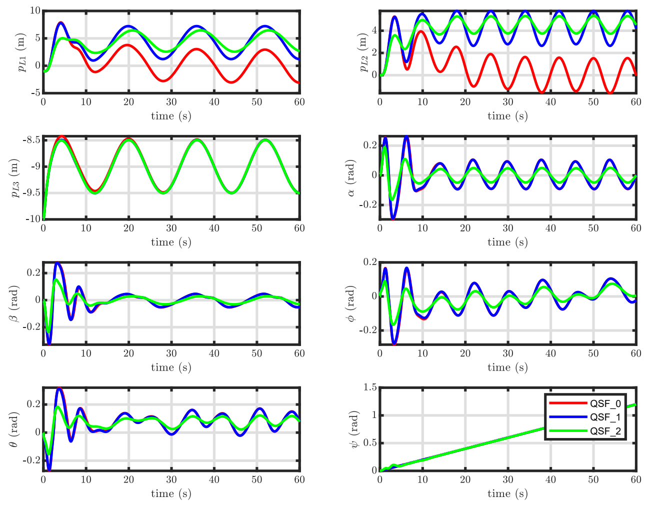


Figure 3.10: Output tracking: System states p, α, β, η. Constant disturbance.

3.3 PX4-SITL Simulation

This section presents Software-In-The-Loop (SITL) simulation of the QSF. SITL emulates an autopilot environment so that performance can be verified in a safe and realistic setting. For instance, it ensures the design is robust to unmodeled effects such as controller saturation, multi-rate sampling, and computational delay. Such real-world effects can degrade performance of the design or make theory impossible to implement (e.g., there may not be enough processing power or memory). We

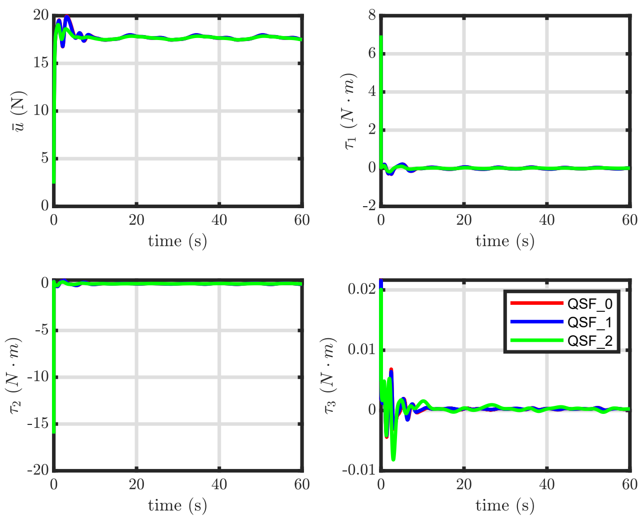


Figure 3.11: Output tracking: Control inputs $\bar{u}, \tau$. Constant disturbance.

choose the open source PX4-SITL framework [117] with the Gazebo simulator using the ODE as the physics engine [128]. Gazebo is chosen for the simulator because of its simple multi-body model format and since it is open-source and currently recommended by the PX4 developers for SITL. Unlike the Matlab simulation in Section 3.1 where the SLS model directly uses the differential equations (2.14), a Gazebo model requires no differential equation model but rather is based on the geometry and inertial properties of the system's multiple bodies. The PX4-Gazebo SITL simulation leverages the RotorS simulation plugins [129]. RotorS is based on a ROS/Gazebo framework. We provide the simulation source code so that the research community can reproduce the results independently or modify the design on a standard desktop PC.

The structure of the simulation environment is shown in Fig 3.12. There are three components: the Gazebo simulator, the PX4 autopilot code, and the ROS Offboard Control. Gazebo contains a multibody dynamics engine, a 3D graphics interface, a 3DR Iris quadrotor-based SLS model description file, external disturbances, and plugins adopted from RotorS that simulate onboard sensors, e.g., GPS, IMU, and magnetometer. The PX4 autopilot software is emulated to run on a desktop. The ROS Offboard Control contains our QSF and coordinate transformations that transform the Gazebo world frame to the navigation frame $\mathcal{N}$ used in the QSF.

3.3.1 SITL Results

A video of the stabilization and time-varying tracking discussed in this subsection is at `https://www.youtube.com/watch?v=Js6GVBOfXts&t=1s`.

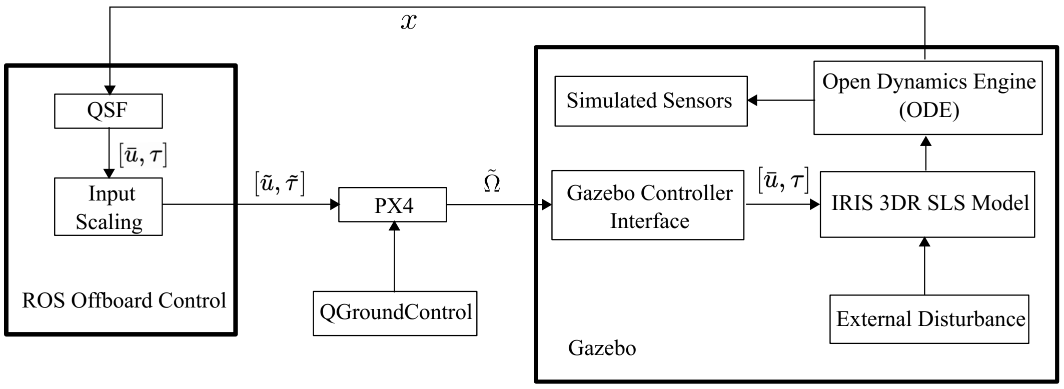


Figure 3.12: PX4-SITL System.

Stabilization

For QSF stabilization, we test the controller first without disturbance and then with a wind disturbance applied. The set point for the QSF is $y_d = [0, 0, -10, 0]^T$. The quadrotor takes off with the PX4 built-in motion controller and commanded to a position away from the set point. The configuration variables are in Fig. 3.13 with the QSF_0 activated at $t = 55$ s. QSF_0 is enabled in the region shaded in blue. As can be seen from Fig. 3.14, the built-in PX4 controller (which uses the MC position control module) has weakly decaying oscillations in all configuration variables. On the other hand, QSF_0 achieves good stabilization performance with a well-damped transition to the setpoint in about 5 s. This performance is similar to the stabilization in Matlab given in Section 3.2. The corresponding input trajectories are given in Fig. 3.14 which are physically realizable and unsaturated.

Next, we study the effect of wind disturbance on stabilization performance. Testing this in Gazebo is relatively simple thanks to the open-source wind disturbance plugins included [129]. The main source of disturbance caused by wind is rotor drag [133]. The rotor drag coefficient is set as $C_D = 0.000\,175\,\mathrm{kg}$ [129]. The wind velocity is set to 5 $\mathrm{m\,s^{-1}}$ in the n_2 direction. The set point is $y_d = [0, 0, -10, 0]^T$. Simulation results for the configuration variables are in Fig. 3.15 where the QSF is enabled at $t = 55$ s. QSF_0 can achieve exponential convergence while QSF_1 has steady-state error in the n_2-direction.

Trajectory Tracking

One of the advantages of the QSF is its guaranteed tracking performance for any bounded smooth trajectory for the flat output. We reuse reference trajectory (3.52) and control gains from Section 3.2.2.

As shown in Fig. 3.16 and Fig. 3.17, the tracking error converges near to zero exponentially with good transient performance. The small steady state errors seen

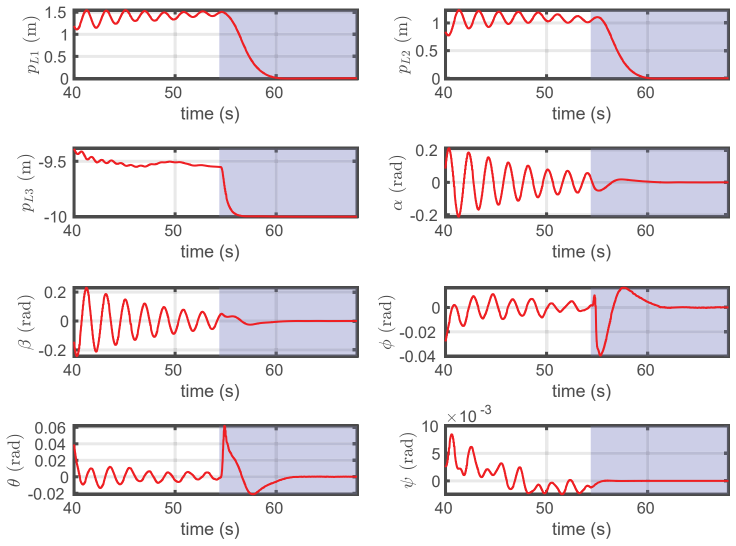


Figure 3.13: SITL stabilization simulation: system states p, α, β, η. QSF_0 is enabled in the blue region. No wind disturbance.

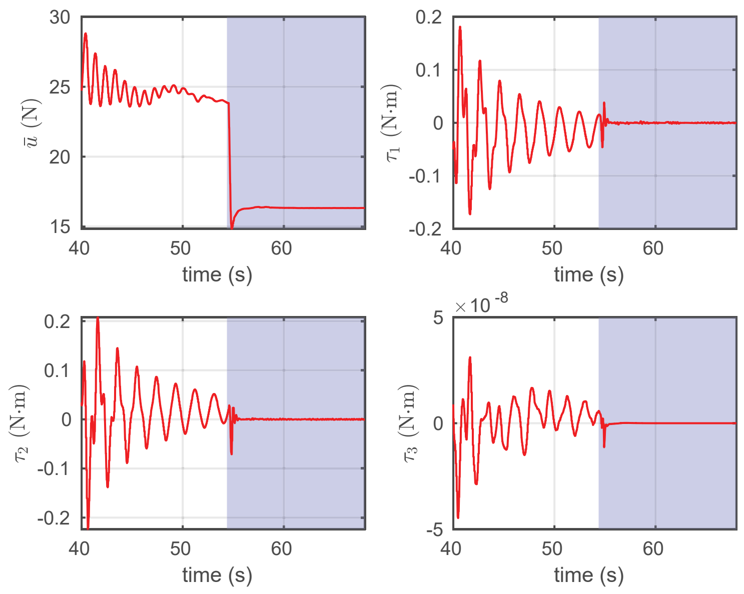


Figure 3.14: SITL stabilization simulation: inputs $\bar{u}, \tau$. QSF_0 is enabled in the blue region. No wind disturbance.

in Fig. 3.16 are due to the rotor drag approximation made in the lumped model (4.1). This should be compared to the Gazebo rotor drag model which acts at individual rotors as described in Section 2.3.3. The control input remains unsaturated, as shown in Fig. 3.18. No wind disturbance is applied.

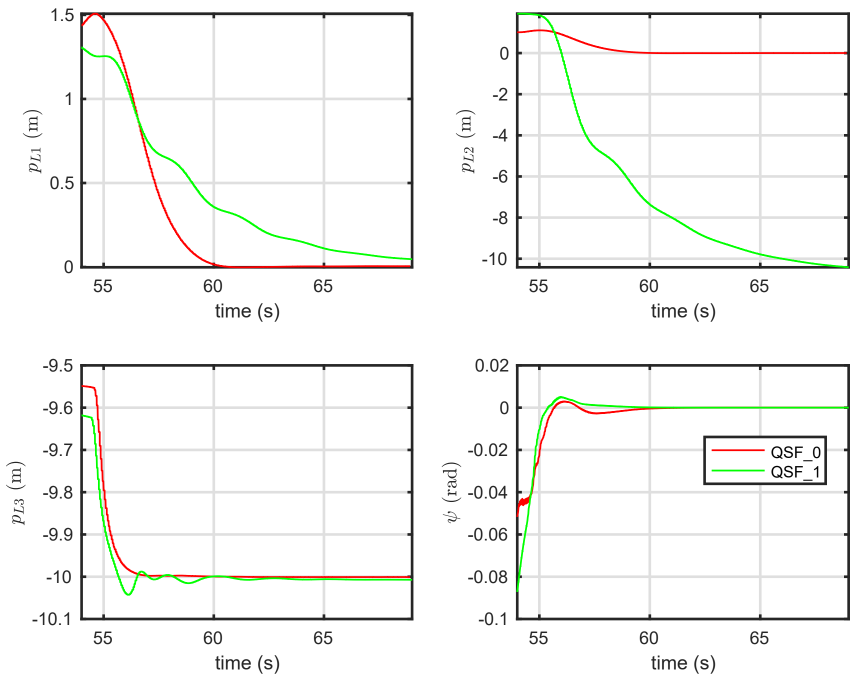


Figure 3.15: Stabilization SITL simulation with wind disturbance: system states p, ψ

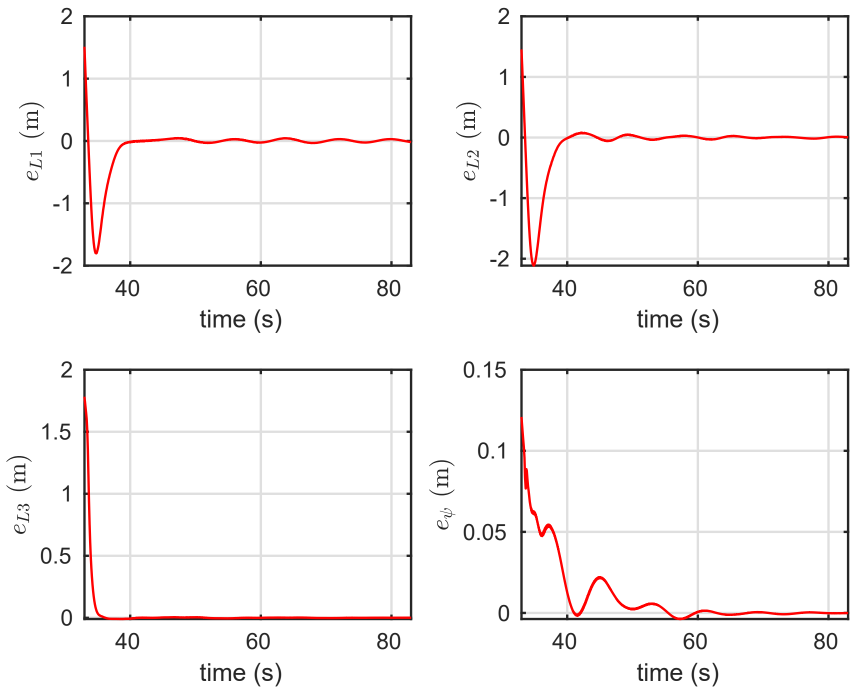


Figure 3.16: Time-varying output tracking SITL simulation: output tracking error $y - y_d$.

Lastly, we study the importance of compensating rotor drag for trajectory tracking by comparing QSF_0 with QSF_2. No wind velocity is applied. Since rotor drag

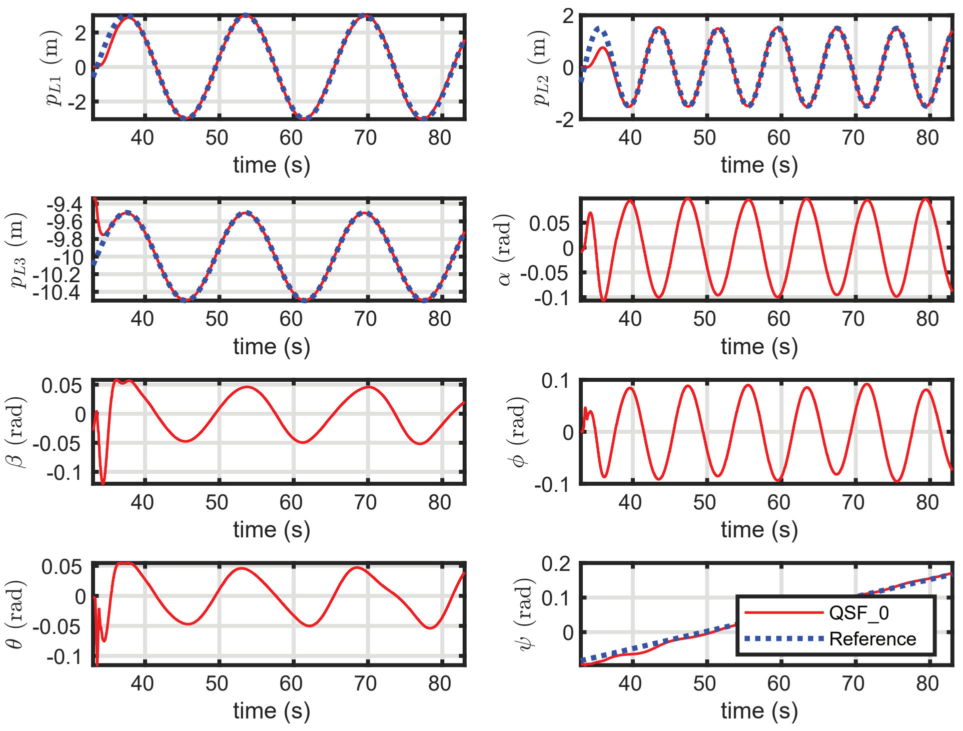


Figure 3.17: Time-varying output tracking SITL simulation: system states p, α, β, η.

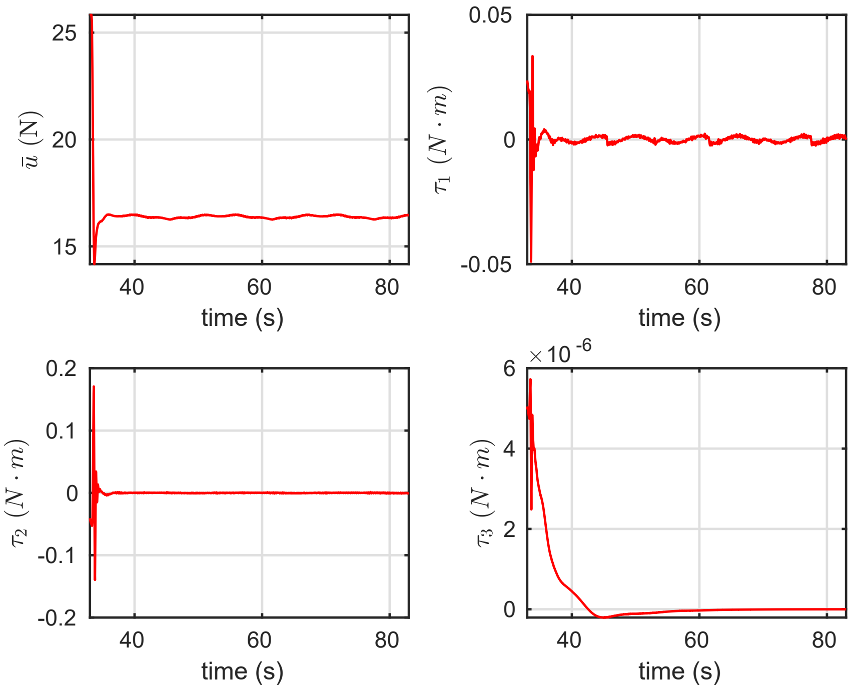


Figure 3.18: Time-varying output tracking SITL simulation: inputs $\bar{u}, \tau$.

is a function of quadrotor velocity, we expect high-speed flight benefits from rotor drag compensation. The output tracking error results are shown in Fig. 3.19. We can see from Fig. 3.19 that without rotor drag compensation (QSF_2), the tracking performance degrades significantly, especially in p_{L1}, p_{L2} direction. These results confirm those obtained in Matlab and are in line with recent work which stresses the

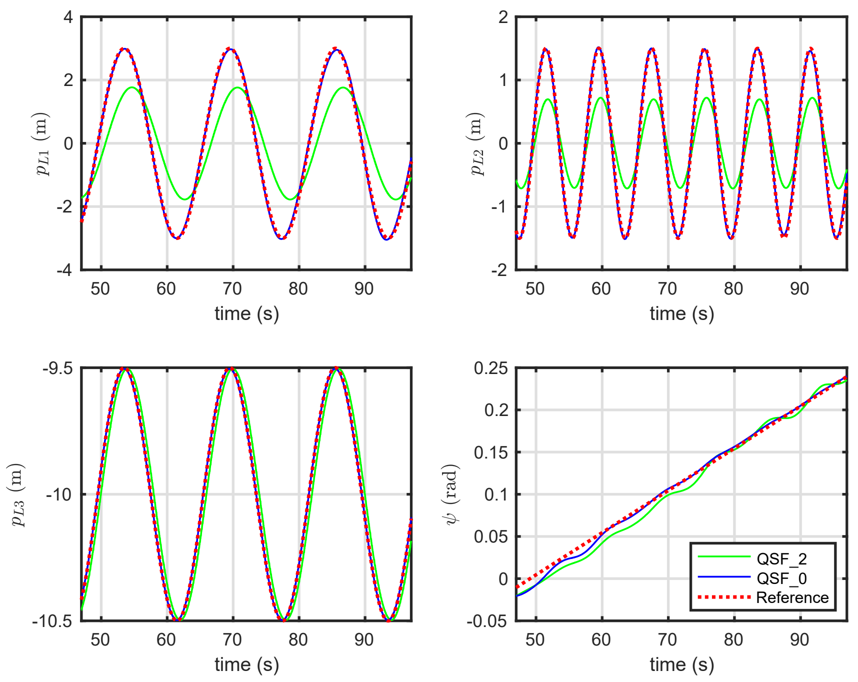


Figure 3.19: Rotor drag effect on time-varying output tracking SITL simulation: output tracking error $y - y_d$.

3.4 Conclusions

A novel algorithm for designing a QSF for a general flat system is presented in this paper. The QSF achieves LTI exponentially stable tracking error dynamics on a well-defined and practical region of state space. Although QSF is investigated in [111]–[113], the design procedure is only demonstrated by specific examples. This paper is the first to provide a systematic algorithm for QSF. The procedure for applying the QSFA is described in detail for the SLS. The resulting tracking controller can be readily tuned thanks to its LTI error dynamics. Compared to dynamic feedback linearization, the QSF benefits from a simpler controller expression with static dependence on state and reference trajectory and its derivatives. We have included rotor drag in the SLS model and directly compensated these forces/torques in the QSF. This improves the performance of QSF trajectory tracking, as shown in the SITL and Matlab simulations. An external disturbance force/torque is added to the model, and integral control is introduced for its compensation. Matlab simulated results show enhanced performance in comparison to a popular geometric control. In order to demonstrate the proposed QSF can be implemented on a real autopilot, we present a software pipeline for implementing a PX4/Gazebo SITL simulation. SITL simulation is a critical step in developing flyable controllers as it proves the design is robust to unmodelled effects. The developed SITL framework code is available publicly and can be used to further develop advanced model-based control.

Chapter 4

Quasi-static Inner-Outer Loop Control of SLS and Experimental Results

4.1 SLS Modelling

This section is based on Section 2.1. We present the outer-loop dynamics here, which will be used in this Chapter.

Compensation for parasitic rotor drag forces has been shown to be important in recent work on aggressive quadrotor motion control such as [20], [22], [123]. Rotor drag force depends strongly on the linear velocity of the UAV and significantly dominates other drag forces such as air resistance due to the fuselage at practical drone speeds (about $5\,\mathrm{m\,s^{-1}}$ or less). Hence, it is therefore natural to consider drag force compensation for the more complex SLS dynamics. Following [115], we include drag forces and torques in the UAV dynamics

$$\begin{aligned}
\dot{p}_Q &= v_Q && (4.1\text{a})\\
m_Q\dot{v}_Q &= m_Q g n_3 - R\bar{u}n_3 + Tq - RDR^T v_Q && (4.1\text{b})\\
\dot{R} &= RS(\omega) && (4.1\text{c})\\
J\dot{\omega} &= -\omega \times J\omega + \tau - AR^T v_Q - B\omega && (4.1\text{d})
\end{aligned}$$

where we denote m_Q as drone mass, g is the acceleration of gravity, $T \geq 0$ denotes tension in the pendulum, $\bar{u} \geq 0$ is total propeller thrust, $J = \mathrm{diag}(J_1, J_2, J_3)$ is the inertia the drone about its CoM, and τ is propeller torque expressed in $\mathcal{B}$. The rotor drag parameters are $D = \mathrm{diag}(d_1, d_2, 0), d_1, d_2 > 0$ for drag force, and constant matrices A and B for drag torque. Further details on rotor drag modelling can be found in [115]. The skew operator $S : \mathbb{R}^3 \to so(3)$ in (4.1c) is

$$S(x) = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}, \quad \text{where } x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}.$$

The rotation matrix R is parameterized with the ZYX Euler angles $\eta = [\phi, \theta, \psi]^T \in \mathbb{R}^3$.

The translational pendulum dynamics is

$$\dot{p}_L = v_L \tag{4.2a}$$

$$m_L \dot{v}_L = -Tq + m_L g n_3 \tag{4.2b}$$

where m_L is payload mass. The rotational pendulum dynamics is

$$\dot{q} = \omega_L \times q \tag{4.3a}$$

$$J_L \dot{\omega}_L = -\omega_L \times J_L \omega_L + Lq \times (m_L g n_3 - m_L \dot{v}_Q) \tag{4.3b}$$

where J_L is the inertia of m_L about the drone's CoM, and ω_L is the load's angular velocity in $\mathcal{N}$. Using the Parallel Axis Theorem, J_L can be expressed in terms of q:

$$J_L = m_L L^2 (I - qq^T) \tag{4.4}$$

We want to write the SLS dynamics using p_Q and v_Q in the state. Hence, we first eliminate T in (4.2b) and (4.1b) to get

$$m_Q \dot{v}_Q + m_L \dot{v}_L = (m_Q + m_L) g n_3 - R \bar{u} n_3 - RDR^T v_Q \tag{4.5}$$

The second derivative of (2.1) gives

$$\dot{v}_L = \dot{v}_Q + L\ddot{q} \tag{4.6}$$

and substituting for $\dot{v}_L$ in (4.5) and using (4.6) gives the payload's translational dynamics

$$(m_L + m_Q)\dot{v}_Q = (m_L + m_Q) g n_3 - R\bar{u} n_3 - m_Q L \ddot{q} - RDR^T v_Q \tag{4.7}$$

where we have substituted $\ddot{q} = \dot{\omega}_L \times q + \omega_L \times \dot{q}$ from (4.3a). Combining (4.7) and (4.6), we get

$$\dot{v}_Q = -\frac{m_L}{m_Q + m_L} L\ddot{q} + g n_3 - \frac{R\bar{u} n_3 + RDR^T v_Q}{m_Q + m_L} \tag{4.8}$$

Substituting (4.4), and (4.8) into (4.3b), we obtain the rotational dynamics of the load expressed in $\mathcal{N}$:

$$m_Q L(I - qq^T)\dot{\omega}_L = q \times (R\bar{u}n_3 + RDR^T v_Q) \tag{4.9}$$

In addition, we have the relation between $\dot{\omega}_L$ and $\dot{\gamma}_\alpha, \dot{\gamma}_\beta$

$$\omega_L = \begin{bmatrix}1\\0\\0\end{bmatrix}\gamma_\alpha + \begin{bmatrix}0\\ \mathrm{c}_\alpha\\ \mathrm{s}_\alpha\end{bmatrix}\gamma_\beta, \dot{\omega}_L = \begin{bmatrix}1 & 0 & 0\\ 0 & \mathrm{c}_\alpha & -\mathrm{s}_\alpha\\ 0 & \mathrm{s}_\alpha & \mathrm{c}_\alpha\end{bmatrix}\begin{bmatrix}\dot{\gamma}_\alpha\\ \dot{\gamma}_\beta\\ \gamma_\alpha\gamma_\beta\end{bmatrix} \tag{4.10}$$

which can be substituted into (4.9) to obtain dynamics for $\gamma_\alpha, \gamma_\beta$.

Given the above, the system dynamics can be separated into inner- and outer-loop dynamics and written in control-affine form. The outer-loop dynamics is

$$\dot{x} = f(x) + g(x)\underbrace{(u_0 - RDR^T v_Q)}_{u} \tag{4.11}$$

$$f(x) = \begin{bmatrix} v_Q \\ \gamma_\alpha \\ \gamma_\beta \\ \mathrm{s}_\beta(\gamma_\alpha^2\mathrm{c}_\beta^2 + \gamma_\beta^2)LM_0 \\ -\mathrm{s}_\alpha\mathrm{c}_\beta(\gamma_\alpha^2\mathrm{c}_\beta^2 + \gamma_\beta^2)LM_0 \\ g + \mathrm{c}_\alpha\mathrm{c}_\beta(\gamma_\alpha^2\mathrm{c}_\beta^2 + \gamma_\beta^2)LM_0 \\ 2\gamma_\alpha\gamma_\beta\mathrm{t}_\beta \\ -\gamma_\alpha^2\mathrm{c}_\beta\mathrm{s}_\beta \end{bmatrix}, g(x) = \begin{bmatrix} 0_{5\times 3} \\ \bar{g}(x) \end{bmatrix}$$

with $x = [p_Q^T, \alpha, \beta, \eta^T, v_Q^T, \gamma_\alpha, \gamma_\beta]^T \in \mathbb{R}^{10}$, $u_0 = R\bar{u}n_3$, $t_\beta = \tan\beta$, $M_0 = m_L/(m_Q + m_L)$,

$$\bar{g}(x) = \begin{bmatrix} & 0_{5\times 3} & \\ \frac{m_L\mathrm{c}_\beta^2 + m_Q}{M_1} & \frac{m_L\mathrm{s}_\alpha\mathrm{s}_\beta\mathrm{c}_\beta}{M_1} & -\frac{m_L\mathrm{c}_\alpha\mathrm{s}_\beta\mathrm{c}_\beta}{M_1} \\ \frac{m_L\mathrm{s}_\alpha\mathrm{s}_\beta\mathrm{c}_\beta}{M_1} & \frac{m_L + m_Q - m_L\mathrm{s}_\alpha^2\mathrm{c}_\beta^2}{M_1} & \frac{m_L\mathrm{s}_\alpha\mathrm{c}_\alpha\mathrm{c}_\beta^2}{M_1} \\ -\frac{m_L\mathrm{c}_\alpha\mathrm{s}_\beta\mathrm{c}_\beta}{M_1} & \frac{m_L\mathrm{s}_\alpha\mathrm{c}_\alpha\mathrm{c}_\beta^2}{M_1} & \frac{m_L + m_Q - m_L\mathrm{c}_\alpha^2\mathrm{c}_\beta^2}{M_1} \\ 0 & \frac{\mathrm{c}_\alpha}{Lm_Q\mathrm{c}_\beta} & \frac{\mathrm{s}_\alpha}{Lm_Q\mathrm{c}_\beta} \\ -\frac{\mathrm{c}_\beta}{Lm_Q} & -\frac{\mathrm{s}_\alpha\mathrm{s}_\beta}{Lm_Q} & \frac{\mathrm{c}_\alpha\mathrm{s}_\beta}{Lm_Q} \end{bmatrix} \tag{4.12}$$

where $M_1 = (m_Q + m_L)m_Q$. Input u is related to η and $\bar{u}$ by

$$u = [F_1, F_2, F_3]^T = -\begin{bmatrix} \mathrm{c}_\phi\mathrm{s}_\theta\mathrm{c}_\psi + \mathrm{s}_\phi\mathrm{s}_\psi \\ \mathrm{c}_\phi\mathrm{s}_\theta\mathrm{s}_\psi - \mathrm{s}_\phi\mathrm{c}_\psi \\ \mathrm{c}_\phi\mathrm{c}_\theta \end{bmatrix}\bar{u} - RDR^T v_Q \tag{4.13}$$

which is an invertible relation between $(\phi, \theta, \bar{u})$ and u (for a given ψ) that is often used in inner-outer loop control [134]. The inner-loop dynamics is given by (4.1c) and (4.1d).

4.2 QSF Outer Loop Controller Design

In order to apply QSF, the system must be flat. Hence, we prove the flatness of the outer-loop (4.11). That is, we show that system states $p_L, v_L, \alpha, \beta, \gamma_\alpha, \gamma_\beta$ and input u can be expressed as a function of the flat output and a finite number of their derivatives.

Theorem 4.2.1. *System*

$$m_Q \dot{v}_Q = m_Q g n_3 - u + Tq \tag{4.14a}$$

$$m_L \dot{v}_L = -Tq + (m_L + m_C) g n_3 \tag{4.14b}$$

has a flat output $y = p_L$.

Proof. The expression for Tq in terms of $\dot{y}$ is obtained immediately from (4.14b). We obtain expressions for the unit vector $q = Tq/\|Tq\|$ and tension $T = Tq \cdot q$.

$$q = \frac{Tq}{\|Tq\|} = \frac{(m_L + m_C) g n_3 - m_L \dot{v}_L}{\|(m_L + m_C) g n_3 - m_L \dot{v}_L\|} \tag{4.15}$$

$$T = \|(m_L + m_C) g n_3 - m_L \dot{v}_L\| \tag{4.16}$$

It is easy to see that α, β can be calculated from (4.15). Finally, $\gamma_\alpha, \gamma_\beta$ can be obtained from the derivative of (4.15) and u can be calculated by adding (4.14a) and (4.14b). □

4.2.1 Quasi-Static Feedback Linearization

In this section, a quasi-static feedback (QSF) linearizing controller is designed with flat output

$$y = [y_1, y_2, y_3]^T = p_L \tag{4.17}$$

We follow the Quasi-static Feedback Algorithm (QSFA) in [40] to design a QSF static state feedback with linear error dynamics.

Following the QSFA we first take enough time derivatives of y so that at least one component of u first appears in every component of y. This occurs with the second-order time derivative of y:

$$\ddot{y} = a^0(x) + D^0(x) u \tag{4.18}$$

where

$$a^0 = \begin{bmatrix} a_1^0 \\ a_2^0 \\ a_3^0 \end{bmatrix} = \begin{bmatrix} -\frac{L s_\beta m_Q (c_\beta^2 \gamma_\alpha^2 + \gamma_\beta^2)}{m_Q + m_L} \\ \frac{L c_\beta s_\alpha m_Q (c_\beta^2 \gamma_\alpha^2 + \gamma_\beta^2)}{m_Q + m_L} \\ \frac{-L c_\alpha c_\beta m_Q (c_\beta^2 \gamma_\alpha^2 + \gamma_\beta^2)}{m_Q + m_L} + g \end{bmatrix}$$

$$D^0 = \begin{bmatrix} D_1^0 \\ D_2^0 \\ D_3^0 \end{bmatrix} = \frac{1}{m_Q + m_L} \begin{bmatrix} s_\beta^2 & -c_\beta s_\alpha s_\beta & c_\beta c_\alpha s_\beta \\ -c_\beta s_\alpha s_\beta & c_\beta^2 s_\alpha^2 & -c_\alpha s_\alpha c_\beta^2 \\ c_\beta c_\alpha s_\beta & -c_\alpha s_\alpha c_\beta^2 & c_\beta^2 c_\alpha^2 \end{bmatrix}$$

and D_i^0 denotes the ith row of D^0. We require a feedback design to stabilize UAV hover as a special case for practical reasons. Hence, we evaluate D^0 at $\alpha = \beta = 0$ and inspect the linear independence of $D_i^0, i = 1, 2, 3$. We obtain

$$D^0(0) = \frac{1}{m_Q + m_L} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.19}$$

Hence, rank $(D^0)(0) = 1$. Since only the last row of $D^0(0)$ is linearly independent, we perform a linearization by assigning auxiliary input $v_1 = \ddot{y}_3$. The first two rows of D^0 are linearly dependent on the third row for all α, β except when $c_\alpha = 0$ where $D_3^0 = 0$, i.e., $D_1^0 = t_\beta / c_\alpha D_3^0$ and $D_2^0 = -t_\alpha D_3^0$. Using these relations we can eliminate u from $\ddot{y}_1, \ddot{y}_2$ to obtain their expressions in terms of x and v_1:

$$\ddot{y}_1 = \frac{t_\beta}{c_\alpha}(g - v_1) \tag{4.20a}$$

$$\ddot{y}_2 = t_\alpha (g - v_1) \tag{4.20b}$$

We remark that since rank $(D^0(x)) = 1$ for all x, the system cannot be static state input-output feedback linearized (w.r.t. y) as relative degree, which requires a non-singular D^0 about some value of x, is not well-defined. Since the system is flat, it can be dynamically linearized. However, the QSF provides a simpler exact linearization with *static* state dependence. Continuing with the QSFA, we take the third time derivative of y_1, y_2:

$$y_1^{(3)} = \frac{(\dot{v}_1 c_\beta s_\beta - \gamma_\beta (g - v_1)) c_\alpha - s_\alpha \gamma_\alpha c_\beta s_\beta (g - v_1)}{c_\beta^2 c_\alpha^2} \tag{4.21a}$$

$$y_2^{(3)} = \frac{\dot{v}_1 s_\alpha c_\alpha + \gamma_\alpha (g - v_1)}{c_\alpha^2} \tag{4.21b}$$

Since u does not appear in (4.21) we must take a fourth time derivative to obtain

$$\begin{bmatrix} y_1^{(4)} \\ y_2^{(4)} \end{bmatrix} = a^1(x, v_1, \dot{v}_1, \ddot{v}_1) + D^1(x, v_1)u \tag{4.22}$$

where $a^1(x, v_1, \dot{v}_1, \ddot{v}_1) \in \mathbb{R}^{2\times 1}$, $D^1(x, v_1) \in \mathbb{R}^{2\times 4}$ are functions of x, auxiliary input v_1, and its time derivatives. Matrix D^1 is

$$D^1(x, v_1) = \frac{g - v_1}{Lm_Q \mathrm{c}_\alpha \mathrm{c}_\beta} \begin{bmatrix} 1 & 0 & -\mathrm{t}_\beta/\mathrm{c}_\alpha \\ 0 & 1 & \mathrm{t}_\alpha \end{bmatrix} \tag{4.23}$$

Defining a second linearization with auxiliary input $v_2 = [y_1^{(4)}, y_2^{(4)}]^T$, and combining it with $v_1 = \ddot{y}_3$, we have the relation between u and $v = [v_2^T, v_1]^T$:

$$v = \begin{bmatrix} v_2 \\ v_1 \end{bmatrix} = \bar{a}(x, v_1, \dot{v}_1, \ddot{v}_1) + \bar{D}(x, v_1)u \tag{4.24}$$

where

$$\bar{a}(x, v_1, \dot{v}_1, \ddot{v}_1) = \begin{bmatrix} a^1(x, v_1, \dot{v}_1, \ddot{v}_1) \\ a_3^0(x) \end{bmatrix}, \ \bar{D}(x, v_1) = \begin{bmatrix} D^1(x, v_1) \\ D_3^0(x) \end{bmatrix} \tag{4.25}$$

The determinant of $\bar{D}$ is given by

$$\det(\bar{D}) = \frac{(g - v_1)^2}{(m_Q + m_L)L^2 m_Q^2 \mathrm{c}_\beta^2 \mathrm{c}_\alpha^2} \tag{4.26}$$

and hence we can solve (4.24) for u as a function of x and $v_1, \dot{v}_1, \ddot{v}_1$ provided $v_1 \neq g$, $\mathrm{c}_\beta \neq 0$ and $\mathrm{c}_\alpha \neq 0$. Since these conditions hold for safe flight conditions, they can be ignored in practice.

By setting $v = [v_2^T, v_1]^T = [y_1^{(4)}, y_2^{(4)}, y_3^{(2)}]^T$ and using (4.24), a linearizing QSF is obtained. The tracking error coordinates are defined as

$$\begin{aligned} \tilde{z} &= [\tilde{z}_1, \ldots, \tilde{z}_{10}]^T \\ &= [y_1 - y_{d1}, \ldots, y_1^{(4)} - y_{d1}^{(4)}, y_2 - y_{d2}, \ldots, y_2^{(4)} - y_{d2}^{(4)}, \\ &\quad y_3 - y_{d3}, \dot{y}_3 - \dot{y}_{d3}]^T \end{aligned} \tag{4.27}$$

where $y_{di}, 1 \leq i \leq 3$ are desired outputs. Applying the linearizing control

$$u = \bar{D}(x, v_1)^{-1}(K\tilde{z} - \bar{a}(x, v_1, \dot{v}_1, \ddot{v}_1) + y_d^{(\bar{r})}) \tag{4.28}$$

with $y_d^{(\bar{r})} = [y_{d1}^{(4)}, y_{d2}^{(4)}, y_{d3}^{(2)}]^T$ we obtain linear error dynamics

$$\dot{\tilde{z}} = (A_c + B_c K)\tilde{z} \tag{4.29}$$

where

$$\begin{aligned} A_c &= \text{blockdiag}(A_1, A_2, A_3) \\ B_c &= \begin{bmatrix} e_4 & e_8 & e_{10} \end{bmatrix} \end{aligned}$$

$e_i \in \mathbb{R}^{10}$ denotes the ith unit vector, and

$$A_3 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \text{ and } A_1 = A_2 = \begin{bmatrix} 0_{3\times1} & I_3 \\ 0 & 0_{1\times3} \end{bmatrix} \in \mathbb{R}^{4\times4}$$

where $A_c \in \mathbb{R}^{10\times10}, B_c \in \mathbb{R}^{10\times3}$ are in Brunovsky Controller form [135], and $K \in \mathbb{R}^{3\times10}$ is a control gain chosen so that $A_c + B_cK$ is Hurwitz and the closed-loop system has appropriate transient performance. An important benefit of LTI error dynamics is the ease of gain tuning. The $\tilde{z}$ coordinates have physical significance of position, linear velocity, linear acceleration, and linear jerk. Hence, choosing the Q and R weights in an LQR gain to achieve suitable transient performance is intuitive. Because $\dot{v}_1, \ddot{v}_1$ in (4.24) are calculated using $y_3 - y_{d3}, \dot{y}_3 - \dot{y}_{d3}$ and their time derivatives, the control law is a *static function* of $x, y_d, \dot{y}_d, \ddot{y}_d, y_d^{(3)}$, and $y_d^{(4)}$. The static nature of the feedback law makes it simpler to implement onboard relative to a dynamic linearization. The expressions for $v_1, \dot{v}_1, \ddot{v}_1$ are linear functions $\tilde{z}_9, \tilde{z}_{10}$ required to evaluate (4.28) and given by

$$\begin{aligned} v_1 &= \ddot{y}_{d3} - k_1\tilde{z}_9 - k_2\tilde{z}_{10} \\ \dot{v}_1 &= y_{d3}^{(3)} - k_1\tilde{z}_{10} - k_2(v_1 - \ddot{y}_{d3}) \\ \ddot{v}_1 &= y_{d3}^{(4)} - k_1(v_1 - \ddot{y}_{d3}) - k_2(\dot{v}_1 - y_{d3}^{(3)}) \end{aligned}$$

where k_1, k_2 are component of gain K.

4.3 Results

4.3.1 Simulation Results

This section presents Software-In-The-Loop (SITL) simulation of the proposed design. We choose the open-source PX4-SITL framework [117] with the Gazebo simulator using the open dynamics engine (ODE) as the physics engine [128]. The PX4 project currently recommends this environment for development, and it is a logical choice since we use Pixhawk/PX4 for the experimental testing in Section 4.3.2. Relative to Matlab, SITL simulation is a useful tool to check controller performance before actual flight tests. It ensures that control designs can be implemented within the constraints of an actual autopilot platform (e.g., controller saturation and sampling frequency effects). Also, the simulation environment uses the RotorS plugin which incorporates an accurate drag model [129], i.e., it models drag at individual rotors and does not assume equal rotor speed as does the model in (4.1b). Hence, our simulation also tests robustness to model error. The standard 3DR Iris quadrotor was modified to include a pendulum. The built-in PX4 module `mc_att_control` was used for the inner-loop.

We test the QSF controller for stabilization and compare it with the PX4's built-in PID outer-loop control `mc_pos_control`. (The stock PX4 firmware uses an inner-outer loop control with inner loop `mc_att_control`.) The set point for the QSF is $y_d = [0, 0, -10, 0]^T$ m. The quadrotor takes off with `mc_pos_control` and then commanded away from y_d. The configuration variables are shown in Fig. 4.1 with the QSF active at $t \geq 138$ s. We observe from Fig. 4.2 that `mc_pos_control` has weakly decaying oscillatory behaviour in all configuration variables. On the other hand, QSF converges quickly to its setpoint in about 5 s and without oscillation. Input trajectories are shown in Fig. 4.3 which are unsaturated and practically realizable.

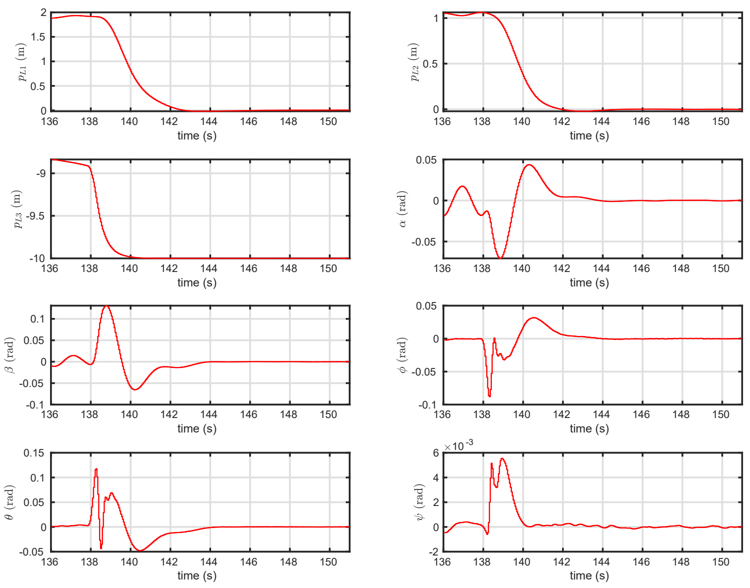


Figure 4.1: SITL simulation: system states p, α, β, η. QSF is active in the blue area.

4.3.2 Experimental Results

The experiments were conducted in the Applied Nonlinear Control Lab (ANCL), University of Alberta. The indoor ANCL flying arena is a $10 \times 5 \times 3$ m^3 volume. The video demonstration of the stabilization and time-varying output tracking is at `https://www.youtube.com/watch?v=wyEOHNX4Rf8&t=1s` and the code used to perform these experiments is at `https://github.com/ANCL/QuasiSLSExp`. The quadrotor platform is a currently available and relatively inexpensive open-source

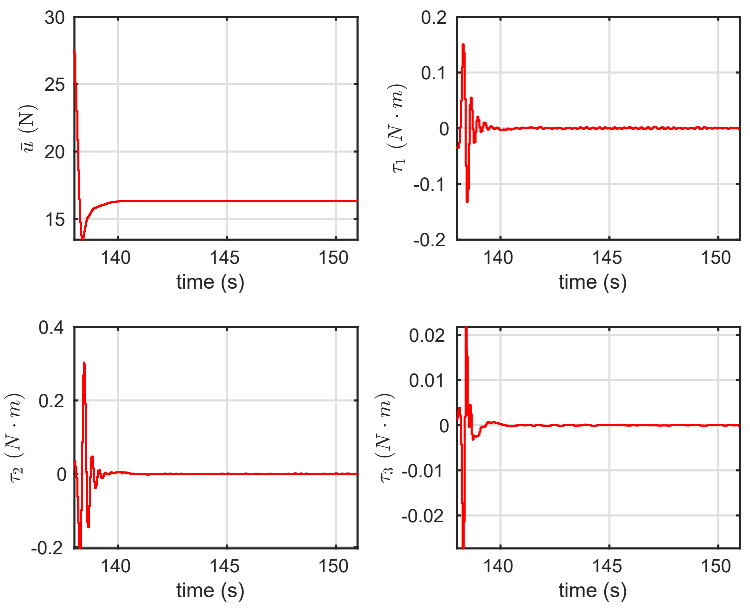


Figure 4.2: SITL simulation: inputs $\bar{u}, \tau$. QSF is enabled in the blue region.

Holybro Vision Dev Kit V1.5. Specifications can be found at [118]. The only modification to the Holybro was a 3D-printed bracket to suspend a rope attached to a 3D-printed payload. A 10 camera Vicon Vantage 5 motion capture system collects the pose trajectories of the Holybro and load. The block diagram of the system is shown in Fig. 4.3. The frequency of the Vicon data is 100 Hz and transmitted to the companion computer using 5 GHz Wi-Fi. Knowing p_L and p_Q allows us to compute q from (2.1). Knowing v_L and v_Q we can determine γ_α and γ_β. Yaw ψ is extracted from the Vicon UAV pose since magnetic disturbance prevented the use of the Pixhawk's onboard magnetometer. Velocity states were obtained with low-pass filtered finite difference on the companion computer. The companion computer sends p_Q, ψ to PX4 at 100 Hz. The QSF runs at 50 Hz on the companion computer and gets p_Q, v_Q from the `EKF2` PX4 module, and p_L, v_L directly from Vicon. The QSF computes the desired force in 3D and computes the desired attitude setpoint and total thrust. The desired attitude is tracked by the built-in PID controller `mc_att_control`. Thrust commands are compensated for decreasing battery voltage. The model parameters used in the control are in 4.1.

Table 4.1: Model parameters.

m_Q	1.412 kg
m_L	0.08 kg
L	1 m
J_1	$0.03\,\mathrm{kg \cdot m^2}$
J_2	$0.03\,\mathrm{kg \cdot m^2}$
J_3	$0.05\,\mathrm{kg \cdot m^2}$

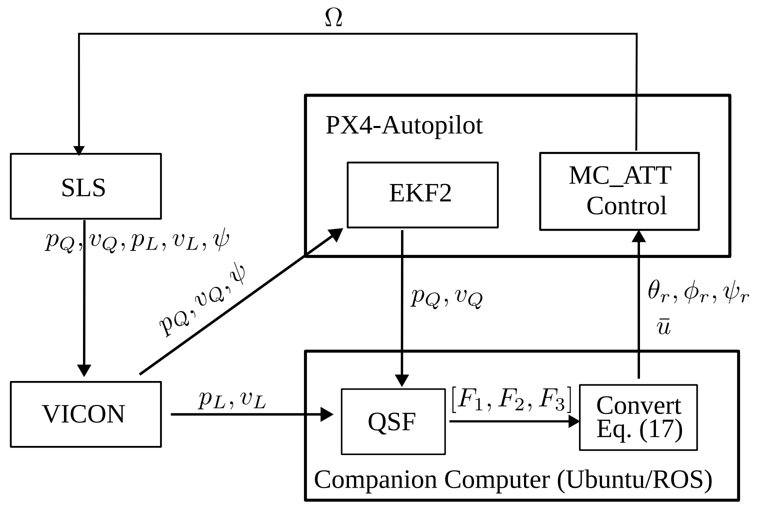


Figure 4.3: Block diagram of closed-loop used in the experiment.

4.3.3 Set-point stabilization

This section presents stabilization performance. Since the error dynamics (4.29) of the QSF is LTI, designing transient performance for y is straightforward. A LQR gain is obtained using $Q = \text{blockdiag}\,(Q_1, Q_2, Q_3), R = 0.1 \cdot I_3$, where $Q_1 = Q_2 = 10 \cdot I_4, Q_3 = I_2$. The QSF is enabled in the coloured areas in Fig. 4.4. The three set points for three coloured areas are $[1, 0.5, -0.6]$ m, $[-1, 0, -0.3]$ m, $[0, 0, -0.3]$ m respectively. QSF achieves stabilization in about 5 s with a reasonable transient. As no integration was used in the outer loop, a small steady-state error is observed.

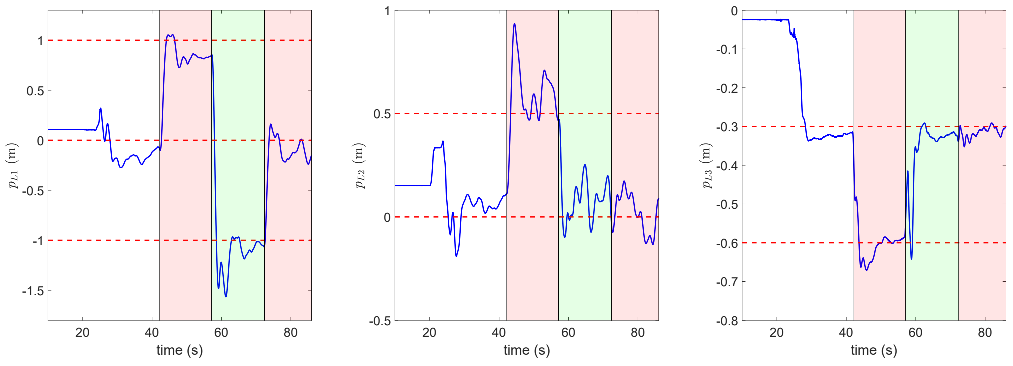


Figure 4.4: Stabilization experiment: Components of payload position p_L

4.3.4 Time-varying Trajectory Tracking

Next, we test the time-varying tracking performance of the QSF. We choose the demanding Figure-8 trajectory

$$y_d(t) = \begin{bmatrix} 1.5\sin(\pi t/8)\ \text{m} \\ 0.75\sin(\pi t/4)\ \text{m} \\ 0.5\sin(\pi t/8) - 0.6\ \text{m} \end{bmatrix} \tag{4.30}$$

We observe good error tracking in Fig. 4.5 which shows the reference in red and the actual position in blue. The corresponding speed $\|v_L\|$ is shown in Fig. 4.6.

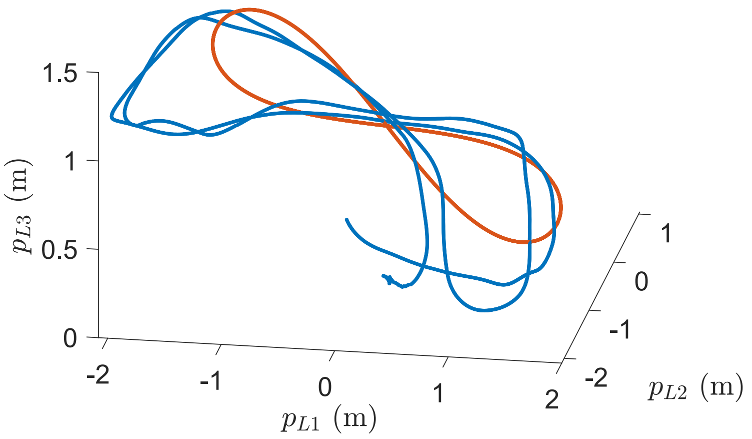


Figure 4.5: Time-varying output tracking experiment: Components of payload position p_L

4.4 Conclusion

This paper presented a novel nonlinear motion control for a SLS. A quasi-static feedback linearization method was applied to the outer-loop of the SLS. The result was a static state feedback law depending on state and reference trajectory. As the error dynamics are LTI, gain tuning and the stability proof were straightforward. The proposed design was tested in a PX4-SITL simulation framework which demonstrated excellent stabilization performance. The same controller was moved to the experimental platform where stabilization and time-varying output tracking for a demanding reference were demonstrated.

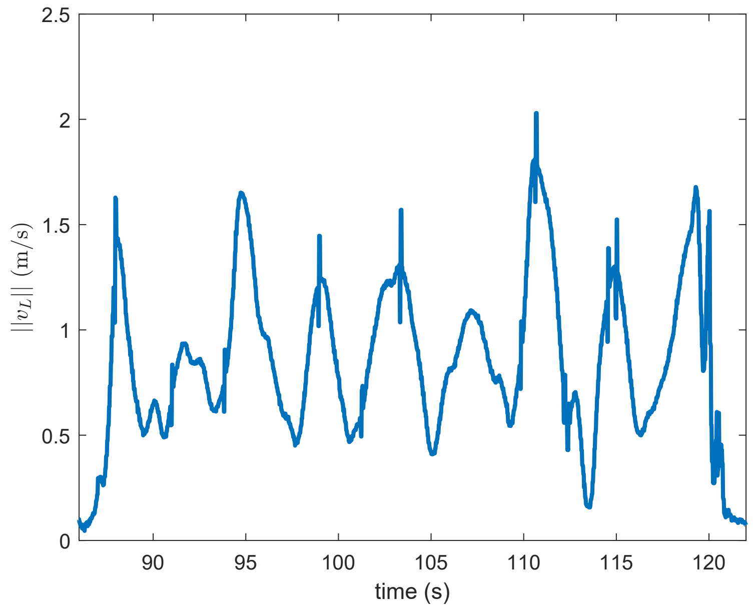


Figure 4.6: Time-varying output tracking experiment: Speed trajectory

Chapter 5

Contact Manipulation with Fully-Actuated Aerial Platform

5.1 Preliminaries on I&I Adaptive Control

Consider the system dynamics

$$\dot{x} = f(x) + g(x)u \tag{5.1}$$

with $x \in \mathbb{R}^n, u \in \mathbb{R}^m$ and an equilibrium $x^\star$ to be stabilized, where the function $f(\cdot)$ and $g(\cdot)$ depend on an unknown parameter vector $\theta \in \mathbb{R}^q$, and the problem of finding an adaptive state feedback control law of the form

$$\dot{\hat{\theta}} = w(x, \hat{\theta}) \tag{5.2}$$

$$u = v(x, \hat{\theta}) \tag{5.3}$$

such that all trajectories of the closed-loop system (5.1), (5.2) and (5.3) are bounded and the tracking error $e = x - x^\star$ converges to zero, is considered.

Since the proposition of *immersion and invariance* (I&I) control [76], it has been adopted widely in nonlinear adaptive control design problems. The basic idea of the *immersion and invariance* (I&I) approach is to design the control by immersing the plant dynamics into a lower-order asymptotic stable target system $\dot{\xi} = \alpha(\xi)$ that captures the desired closed-loop behaviour. We have $x \in \mathbb{R}^n, \xi \in \mathbb{R}^p$ and $p < n$. A smooth mapping $x = \pi(\xi)$ connects the original and target systems. We also require the target system to satisfy the following conditions

$$\pi(\xi(0)) = x(0) \tag{5.4}$$

$$\pi(0) = 0 \tag{5.5}$$

$$f(\pi(\xi)) + g(\pi(\xi))\mu(\pi(\xi)) = \frac{\partial \pi}{\partial \xi}\alpha(\xi) \tag{5.6}$$

where $u = \mu(x)$ is the control law. From (5.4), we conclude that the systems x and ξ share starting point. Condition (5.5) guarantees those two systems have the same equilibrium point and (5.6) is equivalent to $\dot{x} = \dot{\pi}(\xi)$. Then any trajectory $x(t)$ is the image of a trajectory $\xi(t)$ of the target system. Thus the stabilisation problem for the zero equilibrium of the original system can be recast as the problem of solving the partial differential equation (5.6) with the boundary conditions (5.4) and (5.5).

Define a manifold in n-dimensional state space as

$$\mathcal{M} = \{x \in \mathbb{R}^n | x = \pi(\xi), \xi \in \mathbb{R}^p\} \tag{5.7}$$

with (5.4) and (5.5) hold. From (5.6), the manifold $\mathcal{M}$ is invariant. Hence all trajectories $x(t)$ that start on the manifold remain there and asymptotically converge to the point $\pi(0)$, which is the origin. However, from (5.4)-(5.6), the mapping π and the control law v depend on initial condition $x(0)$. By modifying the control law $u = \mu(x)$ so that, for all initial conditions, the trajectories of the original system remain bounded and asymptotically converge to the manifold $\mathcal{M}$, then $\mathcal{M}$ is rendered attractive. The attractivity of the manifold $\mathcal{M}$ can be expressed in terms of the distance

$$z = \text{dist}(x, \mathcal{M}) \tag{5.8}$$

which should be driven to zero.

A natural application of the I&I approach is adaptive control. We define $x' = [x^T, \hat{\theta}]^T$ as the original system state and the target system state is $\xi = x$. The smooth mapping $\pi(\xi)$ connecting x' and x can be defined as

$$\begin{bmatrix} x \\ \hat{\theta} \end{bmatrix} = \begin{bmatrix} x \\ \theta - \beta(x) \end{bmatrix} \tag{5.9}$$

where $\beta(x)$ is a continuous function yet to be specified. Hence the invariant manifold $\mathcal{M}$ is defined as

$$\mathcal{M} = \{[x, \hat{\theta}] \in \mathbb{R}^n \times \mathbb{R}^q | \hat{\theta} - \theta + \beta(x) = 0\} \tag{5.10}$$

And the distance term z is defined as

$$z = \hat{\theta} - \theta + \beta(x) \tag{5.11}$$

Thus we transform the adaptive control designed problems into finding the nominal controller that stabilizes the target system and finding a parameter updating law that drives z to zero. The main theorem of I&I stabilization is presented below.

Theorem 5.1.1. *Consider the system* (5.1). *Assume that there exist smooth mapping* $\alpha : \mathbb{R}^p \to \mathbb{R}^p, \pi : \mathbb{R}^p \to \mathbb{R}^n, \varphi : \mathbb{R}^n \to \mathbb{R}^{n-p}, c : \mathbb{R}^p \to \mathbb{R}^m$ *and* $v : \mathbb{R}^{n\times(n-p)} \to \mathbb{R}^m$*, with* $p < n$ *such that the following conditions hold:*

1. *The target system*
$$\dot{\xi} = \alpha(\xi) \tag{5.12}$$
with $\xi \in \mathbb{R}^p$ *has a globally asymptotically stable equilibrium at* $\xi^\star \in \mathbb{R}^p$ *and* $\pi(\xi^\star) = x^\star$.

2. *For all* $\xi \in \mathbb{R}^p$,
$$f(\pi(\xi)) + g(\pi(\xi))c(\pi(\xi)) = \frac{\partial \pi}{\partial \xi}\alpha(\xi) \tag{5.13}$$

3. *The set identity*
$$\{x \in \mathbb{R}^n | \varphi(x) = 0\} = \{x \in \mathbb{R}^n | x = \pi(\xi), \xi \in \mathbb{R}^p\} \tag{5.14}$$
holds.

4. *All trajectories of the system*
$$\dot{z} = \frac{\partial \varphi}{\partial x}(f(x) + g(x)v(x, z)) \tag{5.15}$$
$$\dot{x} = f(x) + g(x)v(x, z) \tag{5.16}$$
are bounded and (5.12) *has a uniformly globally asymptotically stable equilibrium at* $z = 0$.

Then $x^\star$ *is a globally asymptotically stable equilibrium of the closed-loop system*

$$\dot{x} = f(x) + g(x)v(x, \varphi(x)) \tag{5.17}$$

The Theorem 5.1.1 can be interpreted as follows. From (5.12) and the fact that $\xi^\star$ is a globally asymptotically stable equilibrium of (5.12), it follows that $\xi(t)$ converges to $\xi^\star$ as $t \to \infty$. From (5.14), it follows that $x(t) = \pi(\xi(t))$ converges to $x^\star$ as $t \to \infty$. Then, the control input $u = v(x, z)$ is designed to drive the off-the-manifold coordinates z to zero and keeps the system trajectories bounded.

To illustrate Theorem 5.1.1, consider the two-dimensional system as an example

$$\dot{x}_1 = -x_1 + \theta x_1^3 x_2 \tag{5.18}$$

$$\dot{x}_2 = u \tag{5.19}$$

First, we can observe that the target system is $\dot{\xi} = -\xi$. The mapping $\pi(\xi)$ is defined as

$$\pi(\xi) = \begin{bmatrix} \xi \\ 0 \end{bmatrix}, \quad c(\pi(\xi)) = 0, \quad \varphi(x_1, x_2) = x_2 \tag{5.20}$$

With (5.20) defined, condition 2 and 3 in Theorem 5.1.1 are satisfied. Now we need to show that it is possible to select u such that the trajectories of the closed-loop system are bounded and $z = x_2$ converge to zero.

Let $u = -K(x_1, x_2)x_2$, with $K(x_1, x_2) \geq k \geq 0$ for any (x_1, x_2) and for some k. We consider the Lyapunov candidate function $V = 1/2(x_1^2 + \theta^2 x_2^2)$. Then its derivative is shown as

$$\begin{aligned} \dot{V} &= x_1\dot{x}_1 + \theta^2 x_2 \dot{x}_2 \\ &= -x_1^2 + \theta x_1^4 x_2 - \theta^2 x_2 u \end{aligned} \tag{5.21}$$

For the second term in (5.21), we have

$$(\theta x_1^3 x_2 - \frac{1}{2}x_1)^2 = \theta^2 x_1^6 x_2^2 + \frac{1}{4}x_1^2 - \theta x_1^4 x_2 \geq 0 \tag{5.22}$$

Thus (5.21) becomes

$$\dot{V} \leq -x_1^2 + \theta^2 x_1^6 x_2^2 + \frac{1}{4}x_1^2 - \theta^2 x_2 u \tag{5.23}$$

Design u as

$$u = -(2 + x_1^8)x_2 \tag{5.24}$$

we have

$$\begin{aligned} \dot{V} &\leq -\frac{3}{4}x_1^2 + \theta^2 x_1^6 x_2^2 - \theta^2 x_2^2 (2 + x_1^8) \\ &\leq -\frac{3}{4}x_1^2 - \theta^2 x_2^2 (2 + x_1^8 - x_1^6) \end{aligned} \tag{5.25}$$

It is easily to verify that $(2 + x_1^8 - x_1^6) \geq 0$. Thus, we can conclude that $\dot{V} \leq 0$. With the above discussion, the condition 4 in Theorem 5.1.1 is satisfied. It is worth noticing that the controller u dose not depend on the parameter θ. This should be compared to the backstepping-based controller $u = -\theta x_1^4 - x_2$.

An important application of the I&I approach is the adaptive control. The adaptive control is a control method that uses the information about the unknown parameters of the system to be controlled to adapt the control law to the system. The I&I adaptive control is a special case of the I&I control, where the target system is designed to be the closed-loop system with nominal parameters. To this end, it is natural to assume that a full-information control law (that depends on θ) is known, i.e., that the following stabilisability condition holds.

Assumption 5.1.1. *There exists a function $v(x, \theta)$, where θ is the unknown parameter vector, such that the system*

$$\dot{x} = f(x) + g(x)v(x, \theta) \tag{5.26}$$

has a globally asymptotically stable equilibrium $x^\star$.

The I&I adaptive control problem is then formulated as follows.

Definition 5.1.1. *The system* (5.1) *with 5.1.1 is adaptive I&I stabilizable if there exists a function $\beta(\cdot), w(\cdot)$ such that all trajectories of the extended system*

$$\dot{x} = f(x) + g(x)v(x, \hat{\theta} + \beta(x)) \tag{5.27}$$

$$\dot{\hat{\theta}} = w(x, \hat{\theta}) \tag{5.28}$$

are bounded and satisfy

$$\lim_{t\to\infty}[g(x(t))v(x(t), \hat{\theta}(t) + \beta(x(t))) - g(x(t))v(x(t), \theta)] = 0 \tag{5.29}$$

The main theorem of I&I adaptive control been applied in this section is presented below.

Theorem 5.1.2. *Consider the system below*

$$\dot{x}_1 = f(x_1) + g(x_1)x_2 \tag{5.30}$$

$$\dot{x}_2 = \theta_2 u + \Phi(x_1, x_2)\theta_1 \tag{5.31}$$

where $x_1, x_2 \in \mathbb{R}$, $\theta_1, \theta_2 \in \mathbb{R}$ are unknown parameters. Without loss of generality, we assume $\theta_2 > 0$. The adaptive I&I state feedback control law

$$\dot{\hat{\theta}} = -(I + \frac{\partial \beta}{\partial \hat{\theta}})^{-1}(\frac{\partial \beta}{\partial x_1}(f(x_1) + g(x_1)x_2) + \frac{\partial \beta}{\partial x_2}(-kx_2 - \epsilon\frac{\partial V(x_1)}{\partial x_1}g(x_1))) \tag{5.32}$$

$$u = -(\hat{\theta}_2 + \beta_2(x, \theta_1))\left(kx_2 + \epsilon\frac{\partial V(x_1)}{\partial x_1}g(x_1) + \Phi(x)(\hat{\theta}_1 + \beta_1(x))\right) \tag{5.33}$$

where $\beta(\cdot) = [\beta_1, \beta_2]^T$*, with*

$$\beta_1(x) = \gamma_1 \int_0^{x_2} \Phi(x_1, \mathcal{X}) d\mathcal{X} \tag{5.34}$$

$$\beta_2(x, \hat{\theta}_1) = \gamma_2 \left(k\frac{x_2^2}{2} + \epsilon \frac{\partial V(x_1)}{\partial x_1} g(x_1) x_2 \right) + \gamma_2 \int_0^{x_2} \Phi(x_1, \mathcal{X})(\hat{\theta}_1 + \beta_1) d\mathcal{X} \tag{5.35}$$

and $k > 0, \epsilon > 0, \gamma_1 > 0, \gamma_2 > 0$ *are constants,* $V(x_1)$ *is a positive function and satisfy the condition*

$$\frac{\partial V(x_1)}{\partial x_1} f(x_1) \leq 0 \tag{5.36}$$

Then the update law (5.32) *is well-defined and the closed-loop system* (5.30)*,* (5.31)*,* (5.32) *and* (5.33) *has a globally stable equilibrium at* $(x, \hat{\theta}) = (0, \theta)$*, and* $\lim_{t\to\infty} x = 0$*.*

The proof is omitted here and can be referred to [79]. In the next section, we will show how to apply the I&I adaptive control to a simple planar hybrid force-motion control and compare the result to the classical adaptive controller based on Lyapunov design.

5.2 An Explanatory Example of I&I Adaptive Hybrid Force-Motion Control

Consider a simple example of a planar Cartesian manipulator constrained so that the end effector follows an inclined line with a constant slope. The system dynamics are given by

$$\ddot{p}_1 = u_1 + f_1 \tag{5.37}$$

$$\ddot{p}_2 = u_2 + f_2 \tag{5.38}$$

with scalar constraint $p_1 + Ap_2 = 0$, with A are unknown parameters.

Thus, the forces of the constraint are

$$f_1 = \lambda \tag{5.39}$$

$$f_2 = A\lambda \tag{5.40}$$

where λ is the constraint multiplier. λ can be solved from the constraint equation's time derivatives.

$$\lambda = -\frac{u_1 + Au_2}{A^2 + 1} \tag{5.41}$$

The control objective is to track a possibly time-varying desired joint position p_{2d} and force component f_{1d}. Note that since we assume the manipulator remains in contact with the line, these references determine corresponding p_{1d} and f_{2d}.

We have a coordinate transformation defined as

$$q = \begin{bmatrix} p_1 + Ap_2 + b \\ p_2 \end{bmatrix} \tag{5.42}$$

and the inverse transformation is

$$p = \begin{bmatrix} q_1 - Aq_2 - b \\ q_2 \end{bmatrix} \tag{5.43}$$

The dynamics in the new coordinates are

$$-A\ddot{q}_2 = u_1 + f_1 \tag{5.44a}$$
$$(A^2+1)\ddot{q}_2 = -Au_1 + u_2 \tag{5.44b}$$
$$q_1 = 0 \tag{5.44c}$$

Assuming the constraint is exactly known, the hybrid force-motion controller is designed as

$$u_1 = -A\left(\ddot{q}_{2d} + k_v(\dot{q}_{2d} - \dot{q}_2) + k_p(q_{2d} - q_2)\right) - k_f(f_d - f_1) - f_1 \tag{5.45a}$$
$$u_2 = \left(\ddot{q}_{2d} + k_v(\dot{q}_{2d} - \dot{q}_2) + k_p(q_{2d} - x_2)\right) - Ak_f(f_d - f_1) - Af_1 \tag{5.45b}$$

With this controller, the closed-loop dynamics are

$$A(\ddot{e}_2 + k_v\dot{e}_2 + k_pe_2) = -k_f(f_d - f_1) \tag{5.46a}$$
$$(A^2+1)(\ddot{e}_2 + k_v\dot{e}_2 + k_pe_2) = 0 \tag{5.46b}$$

where $e_2 = q_{2d} - q_2$. We can conclude from (5.46) that

$$\lim_{t\to\infty} e_2 = 0 \tag{5.47a}$$
$$\lim_{t\to\infty} f_1 = f_d \tag{5.47b}$$

5.2.1 Lyapunov based adaptive control

Same as the previous case, the dynamics in q coordinates are

$$-A\ddot{q}_2 = u_1 + f_1 \tag{5.48a}$$
$$(A^2+1)\ddot{q}_2 = -Au_1 + u_2 \tag{5.48b}$$
$$q_1 = 0 \tag{5.48c}$$

Since the constraint is not exactly known, the control input is based on the estimated constraint parameters.

$$u_1 = -\hat{A}\left(\ddot{x}_{2d} + k_v\dot{e} + k_p e_2\right) - k_f(f_d - f_1) - f_1 \tag{5.49a}$$

$$u_2 = \left(\ddot{x}_{2d} + k_v\dot{e} + k_p e_2\right) - \hat{A}k_f(f_d - f_1) - \hat{A}f_1 \tag{5.49b}$$

Substitute (5.49) into (5.48), we have the closed-loop dynamics

$$A(\ddot{e}_2 + k_v\dot{e}_2 + k_p e_2) = \tilde{A}v - k_f(f_d - f_1) \tag{5.50a}$$

$$(A^2+1)(\ddot{e}_2 + k_v\dot{e}_2 + k_p e_2) = A\tilde{A}v - \tilde{A}(k_f(f_d - f_1) + f_1) \tag{5.50b}$$

where

$$v = \ddot{q}_{2d} + k_v\dot{e}_2 + k_p e_2$$
$$\tilde{A} = A - \hat{A}$$

Define Lyapunov function candidate as

$$V = \frac{A^2+1}{2}k_p e_2^2 + \frac{A^2+1}{2}\dot{e}_2^2 + \frac{1}{2}\tilde{A}^2 \tag{5.51}$$

The time derivative of V is

$$\begin{aligned}\dot{V} &= (A^2+1)k_v e_2\dot{e}_2 + (A^2+1)\dot{e}_2\ddot{e}_2 + \tilde{A}\dot{\tilde{A}} \\ &= (A^2+1)\dot{e}_2(k_p e_2 + \ddot{e}_2) + \tilde{A}\dot{\tilde{A}} \\ &= -(A^2+1)k_v\dot{e}_2^2 + \tilde{A}(A\dot{e}_2 v - \dot{e}_2(k_f(f_d - f_1) + f_1) + \dot{\tilde{A}})\end{aligned} \tag{5.52}$$

The update law of $\hat{A}$ is

$$\dot{\hat{A}} = -\dot{\tilde{A}} = A\dot{e}_2 v - \dot{e}_2(k_f(f_d - f_1) + f_1) \tag{5.53}$$

(5.52) becomes

$$\dot{V} = -(A^2+1)k_v\dot{e}_2^2 \le 0 \tag{5.54}$$

From (5.54), the time derivative Lyapunov function $V(q)$ is semi-negative definite. We can conclude that the closed-loop system is stable and $\lim_{t\to\infty}\dot{e}_2 = 0$. But we cannot conclude that $\lim_{t\to\infty}\tilde{A} = 0$. This is usually appeared in the adaptive control. The adaptive control based on Lyapunov is not able to guarantee the convergence of the estimated parameters. Then from (5.50b), we have if $\lim_{t\to\infty}\tilde{A} \neq 0$, then $\lim_{t\to\infty} e_2 \neq 0$.

Matlab simulation result is given below. We set the nominal parameter value as $A = b = 1$, the system initial point as $p_1 = -1, p_2 = 0$, the set point is set as

$p_{1d} = -2, p_{2d} = 1$. The initial value of line parameter $\hat{A} = 0.8$. The control gain is chosen as $k_v = 4, k_p = 4, k_f = 1$. The simulation result is shown in Figure 5.1, 5.2, 5.3, 5.4. We can observe the steady state error of p_2 from Figure 5.1 and the parameter estimation error $\tilde{A}$ from Figure 5.4. The steady state error and parameter estimation error motivate us to design an innovative adaptive controller based on I&I control. The effects of the persistent excitation condition are shown in Figure **??**. The steady-state errors of the green and cyan lines are smaller than those observed during stabilization. This improvement is due to the addition of an excitation signal to the input reference during trajectory tracking.

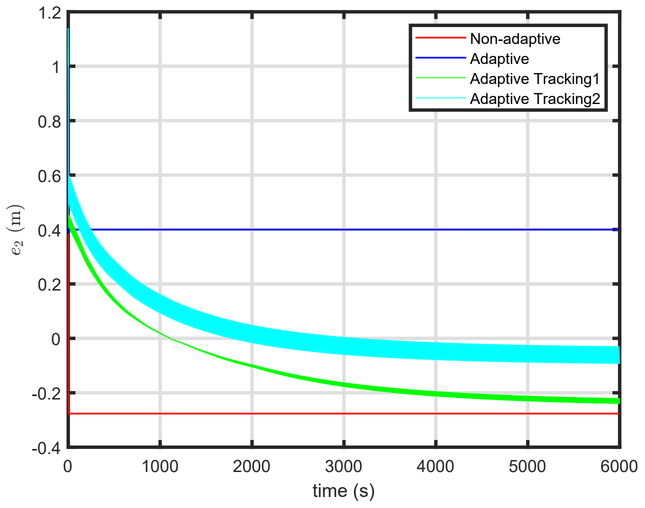


Figure 5.1: Stabilization position error $q_{2d} - q_2$.

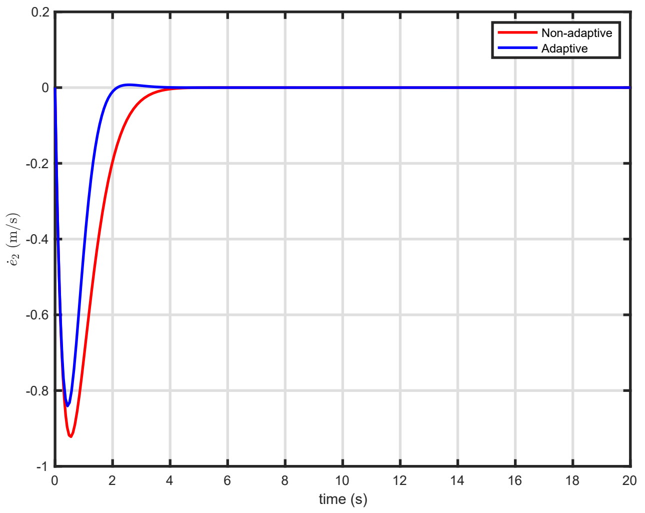


Figure 5.2: Stabilization velocity error $\dot{q}_{2d} - \dot{q}_2$.

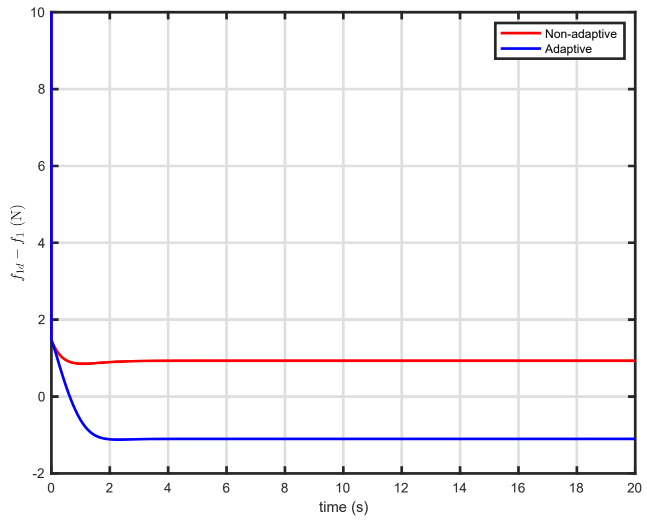


Figure 5.3: Stabilization force error $f_d - f_1$.

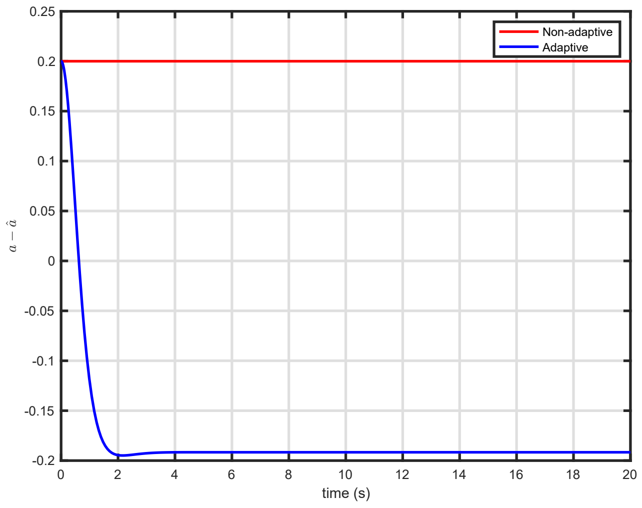


Figure 5.4: Parameter a convergence error $a - \hat{a}$.

5.2.2 I&I adaptive control

The target system is closed-loop system (5.46). Define $z = \hat{A} - A + \beta(q_2, \dot{q}_2)$. Take the time derivative of z, we have

$$\dot{z} = \dot{\hat{A}} + \frac{\partial \beta}{\partial q_2}\dot{q}_2 + \frac{\partial \beta}{\partial \dot{q}_2}\ddot{q}_2 \tag{5.55}$$

We design $\beta(q_2, \dot{q}_2)$ as

$$\beta(q_2, \dot{q}_2) = \gamma(k_p q_2 \dot{q}_2 + \frac{k_v}{2}\dot{q}_2^2) \tag{5.56}$$

Here, we apply the Theorem 5.1.2, specifically the state is $q_2, \dot{q}_2$, $\theta_1 = 0, \theta_2 = A$. We evaluate (5.35) to get $\beta(q_2, \dot{q}_2)$

With $\beta(q_2, \dot{q}_2)$ defined, (5.55) becomes

$$\begin{aligned}\dot{z} &= \dot{\hat{A}} + \gamma(k_p\dot{q}_2^2 + v\ddot{q}_2) \\ &= \dot{\hat{A}} + \gamma\left(k_p\dot{q}_2^2 + v\frac{-Au_1 + u_2}{A^2+1}\right) \qquad (5.57)\end{aligned}$$

where

$$v = k_p q_2 + k_v \dot{q}_2 \qquad (5.58)$$

To make $\dot{z}$ negative definite, we design the update law of $\hat{A}, u_1$ as

$$u_1 = (\hat{A} + \beta(q_2, \dot{q}_2))v - f_d \qquad (5.59)$$

$$u_2 = -v - (\hat{A} + \beta) f_d \qquad (5.60)$$

$$\dot{\hat{A}} = \gamma\left(-k_p\dot{q}_2^2 + v^2 + (f_d - f_1)\right) \qquad (5.61)$$

Then (5.57) becomes

$$\begin{aligned}\dot{z} &= \dot{\hat{A}} + \gamma\left(k_p\dot{q}_2^2 + v\frac{-A((z+A)v - f_d) - v - (z+A)f_d}{A^2+1}\right) \\ &= \dot{\hat{A}} + \gamma\left(k_p\dot{q}_2^2 - v^2 + v\frac{-Azv - zvf_d}{A^2+1}\right) \\ &= \gamma\left(\frac{-Azv^2 - zvf_d}{A^2+1} + f_d(f_d - f_1)\right) \\ &= \gamma\left(\frac{-Azv^2 - zvf_d}{A^2+1} + f_d^2 + f_d\frac{u_1 + au_2}{A^2+1}\right) \\ &= \gamma\left(\frac{-Azv^2 - zvf_d}{A^2+1} + \frac{zvf_d - Azf_d^2}{A^2+1}\right) \\ &= -\frac{\gamma}{A^2+1} z(Av^2 + Af_d^2) \qquad (5.62)\end{aligned}$$

Thus we can conclude that the z subsystem is asymptotically stable.

Apply (5.59), (5.60) and (5.61) to (5.48), we have the closed-loop dynamics as

$$A(\ddot{q}_2 + k_v\dot{q}_2 + k_p q_2) = -zv + f_d - f_1 \qquad (5.63)$$

$$(A^2+1)(\ddot{q}_2 + k_v\dot{q}_2 + k_p q_2) = -Azv - zf_d \qquad (5.64)$$

Defining Lyapunov function candidate as

$$V = \frac{A^2+1}{2}(k_p q_2^2 + \dot{q}_2^2) + \frac{A^2+1}{A\gamma} z^2 \qquad (5.65)$$

Then the first time derivative of V is

$$\begin{aligned}
\dot{V} &= (A^2+1)(k_p x_2 \dot{q}_2 + \dot{q}_2 \ddot{q}_2) + \frac{A^2+1}{A\gamma} z\dot{z} \\
&= \dot{q}_2(-(A^2+1)k_v \dot{q}_2 - Azv - zf_d) - 2z^2(v^2+f_d^2) \\
&= -k_v \dot{q}_2^2 - z^2(v^2+f_d^2) - A^2 k_v \dot{q}_2^2 - Azv\dot{q}_2 - z^2v^2 - f_d^2 z^2 - f_d z \dot{q}_2 - \dot{q}_2^2 \\
&\leq -k_v \dot{q}_2^2 - z^2(v^2+f_d^2) \leq 0 \qquad (5.66)
\end{aligned}$$

In the above inequality equation, we utilize the following inequalities,

$$\begin{aligned}
(A\dot{q}_2 + \frac{1}{2}zv)^2 &= A^2\dot{q}_2^2 + Azv\dot{q}_2 + \frac{1}{4}z^2v^2 \geq 0 \\
(f_d z + \frac{1}{2}\dot{q}_2)^2 &= f_d^2 z^2 + f_d z \dot{q}_2 + \dot{q}_2^2 \geq 0 \qquad (5.67)
\end{aligned}$$

Finally, we establish the global asymptotically stabilisability of the closed-loop system. For the case where $\dot{V} = 0$, we have $z = 0$ and $\dot{q}_2 = 0$. When $z = 0$ and $\dot{q}_2 = 0$, we can get $q_2 = 0$ and $\lim_{t\to\infty} f_1 = f_d$. Thus, according to LaSalle's theorem, we can conclude that the close-loop system is globally asymptotically stable.

The Matlab simulation result is given here. From Fig. 5.5, 5.6, 5.7, 5.8, we can observe that the state stabilization error of e_2 converge to zero and the parameter estimation error $\tilde{A}$ converge to zero. The I&I adaptive controller is able to guarantee the convergence of the estimated parameters and the global convergence of the system states.

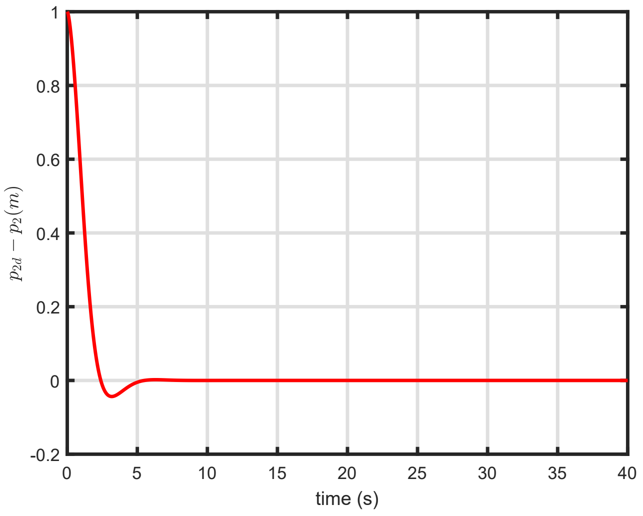


Figure 5.5: Stabilization position error $q_{2d} - q_2$.

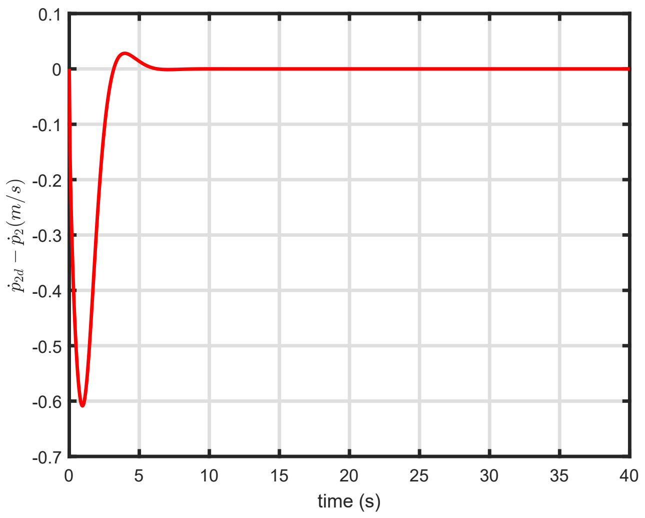


Figure 5.6: Stabilization velocity error $\dot{q}_{2d} - \dot{q}_2$.

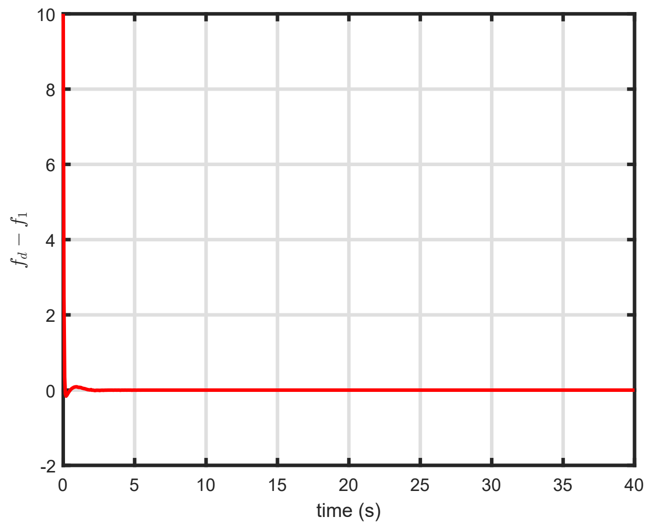


Figure 5.7: Stabilization force error $f_d - f_1$.

5.2.3 I&I Adaptive Hybrid Force-Motion Controller for Robot Arm Manipulator

A practical extension of the previous example is the robot arm manipulator. The dynamics of the robot arm manipulator is given as

$$M(\theta)\ddot{\theta} + c(\theta, \dot{\theta})\dot{\theta} + G(\theta) = \tau + \tau_e \tag{5.68}$$

where $M(\theta)$ is the inertia matrix, $c(\theta, \dot{\theta})$ is the Coriolis and centrifugal matrix, $G(\theta)$ is the gravity vector, τ is the control input, $J(\theta)$ is the Jacobian matrix, f is the

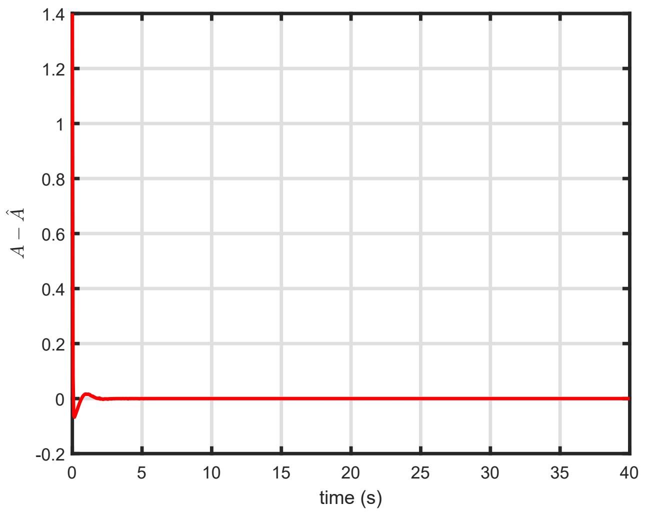


Figure 5.8: Parameter a convergence error $a - \hat{a}$.

constraint force. The dynamics in task space is given as

$$\mathcal{F} = \Lambda(\theta)\dot{\mathcal{V}} + \eta(\theta, \mathcal{V}) + \mathcal{F}_{tip} \tag{5.69}$$

where

$$\Lambda(\theta) = J^{-T} M(\theta) J^{-1} \tag{5.70}$$
$$\eta(\theta, \mathcal{V}) = J^{-T}(c(\theta, \dot{\theta})\dot{\theta} + G(\theta)) - \Lambda \dot{J} J^{-1} \mathcal{V} \tag{5.71}$$
$$\mathcal{F} = J^{-T}\tau \, , \, \mathcal{F}_{tip} = J^{-T}\tau_e \tag{5.72}$$

The end-effector position is given as X_e and the end-effector velocity is given as $\mathcal{V}$. We have $\mathcal{V} = J\dot{\theta}$. The $\Lambda(\theta)$ and $\eta(\theta, \mathcal{V})$ are the inertia matrix and the Coriolis and centrifugal matrix in task space.

By setting $\mathcal{F}_n = \Lambda^{-1}(\mathcal{F} - \eta), \mathcal{F}_{tipn} = \Lambda^{-1}\mathcal{F}_{tip}$, we have

$$\mathcal{F}_n = \dot{\mathcal{V}} + \mathcal{F}_{tipn} \tag{5.73}$$

We should notice that (5.73) is in the same format as (5.37). Thus, the I&I adaptive controller can be directly applied to the plane robot arm manipulator. We set $\mathcal{F}_n$ same as (5.59), (5.60).

$$\mathcal{F}_n = \begin{bmatrix} (\hat{A} + \beta(q_2, \dot{q}_2))v - f_d \\ -v - (\hat{A} + \beta) f_d \end{bmatrix} \tag{5.74}$$

The update law of $\hat{A}$ is same as (5.61). We also have the map between $\mathcal{F}_n$ and τ as

$$\tau = J^T(\Lambda \mathcal{F}_n + \eta) \tag{5.75}$$

Here, we present the detailed expression for a plane 2 DoF robot manipulator. $M(\theta)$, $c(\theta, \dot{\theta})$, $G(\theta)$ are given as

$$M(\theta) = \begin{bmatrix} m_1 L_1^2 + m_2(L_1^2 + 2L_1L_2\mathrm{c}_{\theta_2} + L_2^2) & m_2(L_2^2 + L_1L_2\mathrm{c}_{\theta_2}) \\ m_2(L_2^2 + L_1L_2\mathrm{c}_{\theta_2}) & m_2L_2^2 \end{bmatrix} \tag{5.76}$$

$$c(\theta, \dot{\theta}) = \begin{bmatrix} -m_2L_1L_2\mathrm{s}_{\theta_2}\dot{\theta}_2 & -m_2L_1L_2\mathrm{s}_{\theta_2}(\dot{\theta}_1 + \dot{\theta}_2) \\ m_2L_1L_2\mathrm{s}_{\theta_2}\dot{\theta}_1 & 0 \end{bmatrix} \tag{5.77}$$

$$G(\theta) = \begin{bmatrix} m_2 g(L_1\mathrm{c}_{\theta_1} + L_2\mathrm{c}_{\theta_1+\theta_2}) \\ m_2L_2 g\mathrm{c}_{\theta_1+\theta_2} \end{bmatrix} \tag{5.78}$$

And the external force is given by

$$\tau_e = \lambda \begin{bmatrix} aL_1\mathrm{c}_{\theta_1} + aL_2\mathrm{c}_{\theta_1+\theta_2} - \mathrm{s}_{\theta_1}L_1 - L_2\mathrm{s}_{\theta_1+\theta_2} \\ aL_2\mathrm{c}_{\theta_1+\theta_2} - L_2\mathrm{s}_{\theta_1+\theta_2} \end{bmatrix} \tag{5.79}$$

where a is the line parameter, $\lambda \in \mathbb{R}$ is the magnitude of the external force. The Jacobian matrix is given as

$$J = \begin{bmatrix} -\mathrm{s}_{\theta_1}L_1 - \mathrm{s}_{\theta_1+\theta_2}L_2 & -\mathrm{s}_{\theta_1+\theta_2}L_2 \\ \mathrm{c}_{\theta_1}L_1 + \mathrm{c}_{\theta_1+\theta_2}L_2 & \mathrm{c}_{\theta_1+\theta_2}L_2 \end{bmatrix} \tag{5.80}$$

We set the end-effector position as X_e and the end-effector velocity as $\mathcal{V}$. We have

$$\dot{X}_e = \begin{bmatrix} L_1\mathrm{c}_{\theta_1} + L_2\mathrm{c}_{\theta_1+\theta_2} \\ L_1\mathrm{s}_{\theta_1} + L_2\mathrm{s}_{\theta_1+\theta_2} \end{bmatrix} \tag{5.81}$$

$$\mathcal{V} = J \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} \tag{5.82}$$

The simulation results of I&I adaptive control for the 2 DoF robot manipulator is shown in Figure 5.9. We set the nominal parameter value as $a = 1$, the system initial point as $\theta_1 = \pi/6, \theta_2 = -\pi/3$, the set point is set as $X_{e2d} = 0.5, f_d = 10$. The initial value of line parameter $\hat{a} = 0.1$. The control gain is chosen as $k_v = 4, k_p = 4, \gamma = 0.08$. We can observe that the state stabilization error of X_{e2}, f converge to zero and the parameter estimation error $\tilde{a}$ converge to zero. The I&I adaptive controller is able to guarantee the convergence of the estimated parameters and the global convergence of the system states.

5.3 Free Motion Controller Design for UAM

Since a free motion controller is required for all tasks, a free motion controller is presented here and is inspired by [71] and the inner-loop controller can be referenced

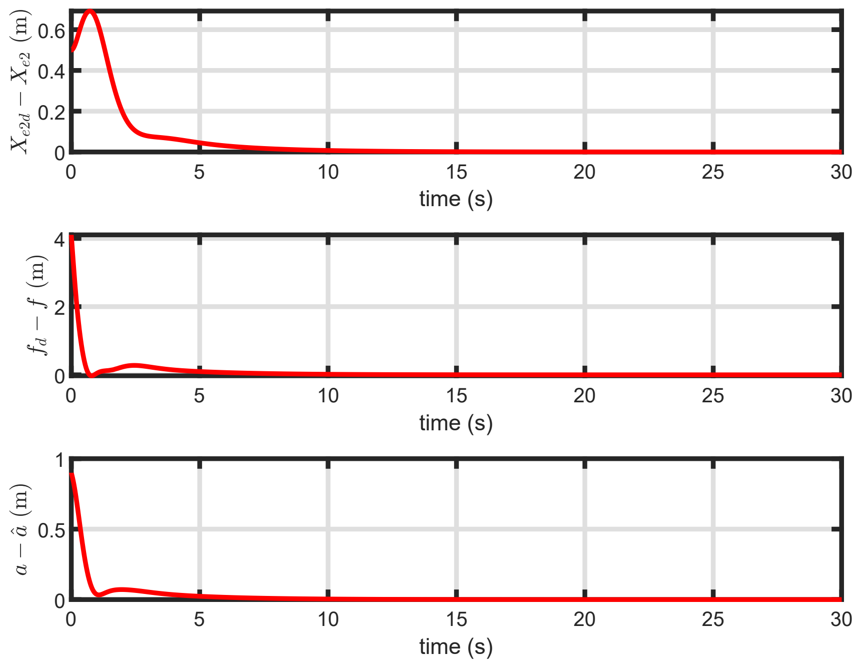


Figure 5.9: Simulation results of 2 DoF robot manipulator I&I adaptive control.

from [136]. The system dynamics is given below.

$$m_Q \dot{v}_Q = m_Q g n_3 - R T_{sum} - f_e \tag{5.83}$$

$$J\dot{\omega} = -\omega \times J\omega + \tau_{sum} - \tau_e \tag{5.84}$$

The controller is designed as

$$T_{sum} = R^T (m_Q \dot{v}_d + m_Q g n_3 - K_p e_p - K_v e_v) \tag{5.85}$$

$$\tau_{sum} = \omega \times J\omega - K_R e_R - K_\omega e_\omega - J(\omega \times R^T R_d \omega_d - R^T R_d \dot{\omega}_d) \tag{5.86}$$

where

$$e_R = \frac{1}{2}(R_d^T R - R^T R_d)^\vee \tag{5.87}$$

$$e_\omega = \omega - R^T R_d \omega_d \tag{5.88}$$

Assuming no external forces and torques $f_e = \tau_e = 0$, we have the closed-loop dynamics as

$$m_Q \dot{e}_v + K_p e_p + K_v e_v = 0 \tag{5.89}$$

$$J\dot{e}_\omega + K_\omega e_\omega + K_R e_R = 0 \tag{5.90}$$

where

$$J\dot{e}_\omega = J\dot{\omega} + J(\omega \times R^T R_d \omega_d - R^T R_d \dot{\omega}_d) \tag{5.91}$$

Then, the zero equilibrium of the tracking errors e_R, e_ω, e_p, e_v is exponentially stable. The region of attraction is characterized by (5.92), (5.93) [136].

$$\frac{1}{2} tr(I - R_d^T(0)R(0)) < 2 \tag{5.92}$$

$$||e_\omega(0)|| < \frac{2}{\lambda_{\min}(J)} \left(1 - \frac{1}{2} tr\left(I - R_d^T(0)R(0)\right)\right) \tag{5.93}$$

5.4 Adaptive Controller Design for UAM

The dynamics of UAM interacting with a rigid surface is given as

$$m_Q \dot{v}_Q = m_Q g n_3 - R T_{sum} - f_e \tag{5.94}$$

$$J\dot{\omega} = -\omega \times J\omega + \tau_{sum} - \tau_e \tag{5.95}$$

To simplify the expression in this section, we define the following variables

$$F = [F_1, F_2, F_3]^T = (R T_{sum} - m_Q g n_3)/m_Q \tag{5.96}$$

$$F_e = [F_{e1}, F_{e2}, F_{e3}]^T = f_e/m_Q \tag{5.97}$$

Thus (5.94) becomes

$$\dot{v}_Q = -F - F_e \tag{5.98}$$

The end-effector position is given by (5.100). The constraint surface is defined as

$$x + A_1 y + A_2 z + b = 0 \tag{5.99}$$

The end-effector position p_E is given by

$$p_E = p_Q +^B R_E [x_e, y_e, z_e]^T \tag{5.100}$$

where ${}^B R_E$ is the rotation matrix from $\mathcal{B}$ to the end-effector frame $\mathcal{E}$. The rotation matrix ${}^B R_E$ is given by${}^B R_E = I_3$. Define $p_E = [p_{E1}, p_{E2}, p_{E3}]$ and substitute into the constraint equation, we have

$$p_{E1} + A_1 p_{E2} + A_2 p_{E3} + b = 0 \tag{5.101}$$

Also from (5.100), we have $v_E = v_Q, \dot{v}_E = \dot{v}_Q$. Define the new coordinate $\mathcal{Q} = [x_1, x_2, x_3]$ as

$$q_1 = p_{E1} + A_1 p_{E2} + A_2 p_{E3} + b \tag{5.102}$$

$$q_2 = p_{E2} \tag{5.103}$$

$$q_3 = p_{E3} \tag{5.104}$$

The F_e is given by $F_e = [1, A_1, A_2]^T \lambda$. And by taking the second time derivative of(5.101) and substitute $\dot{v}_E$ from (5.98), λ can be calculated as

$$\lambda = -\frac{F_1 + A_1 F_2 + A_2 F_3}{1 + A_1^2 + A_2^2} \tag{5.105}$$

The dynamics in $\mathcal{X}$ coordinates are

$$A_1 \ddot{q}_2 + A_2 \ddot{q}_3 = F_1 + \lambda \tag{5.106}$$

$$(1 + A_1^2 + A_2^2)\ddot{q}_2 = A_1 F_1 - (1 + A_2^2) F_2 + A_1 A_2 F_3 \tag{5.107}$$

$$(1 + A_1^2 + A_2^2)\ddot{q}_3 = A_2 F_1 + A_1 A_2 F_2 - (1 + A_1^2) F_3 \tag{5.108}$$

Nominal hybrid force-motion controller

The nominal hybrid force-motion controller is designed as

$$F_1 = A_1 v_1 + A_2 v_2 - \lambda_d \tag{5.109}$$

$$F_2 = -v_1 - A_1 \lambda_d \tag{5.110}$$

$$F_3 = -v_2 - A_2 \lambda_d \tag{5.111}$$

where

$$v_1 = \ddot{q}_{2d} + k_v(\dot{q}_{2d} - \dot{q}_2) + k_p(q_{2d} - q_2) \tag{5.112}$$

$$v_2 = \ddot{q}_{3d} + k_v(\dot{q}_{3d} - \dot{q}_3) + k_p(q_{3d} - q_3) \tag{5.113}$$

5.4.1 I&I adaptive hybrid force-motion controller

We have the adaptive hybrid force-motion controller as

$$F_1 = (\hat{A}_1 + \beta_1) v_1 + (\hat{A}_2 + \beta_2) v_2 - \lambda_d \tag{5.114a}$$

$$F_2 = -v_1 - (\hat{A}_1 + \beta_1)\lambda_d \tag{5.114b}$$

$$F_3 = -v_2 - (\hat{A}_2 + \beta_2)\lambda_d \tag{5.114c}$$

The closed-loop dynamics after applying the controller (5.114) are

$$\begin{aligned}
A_1(\ddot{e}_2 + k_v\dot{e}_2 + k_p e_2) + A_2(\ddot{e}_3 + k_v\dot{e}_3 + k_p e_3) &= -z_1 v_1 - z_2 v_2 + \lambda_d - \lambda && \text{(5.115a)}\\
(1 + A_1^2 + A_2^2)(\ddot{e}_2 + k_v\dot{e}_2 + k_p e_2) &= -A_1 z_1 v_1 - A_1 z_2 v_2 \\
&\quad + z_2 A_2 A_1 \lambda_d - (1 + A_2^2)\lambda_d z_1 && \text{(5.115b)}\\
(1 + A_1^2 + A_2^2)(\ddot{e}_3 + k_v\dot{e}_3 + k_p e_3) &= -A_2 z_1 v_1 - A_2 z_2 v_2 \\
&\quad + z_1 A_2 A_1 \lambda_d - (1 + A_1^2)\lambda_d z_2 && \text{(5.115c)}
\end{aligned}$$

z is defined as

$$z_1 = \hat{A}_1 - A_1 + \gamma\dot{e}_2 \tag{5.116}$$

$$z_2 = \hat{A}_2 - A_2 + \gamma\dot{e}_3 \tag{5.117}$$

Taking the derivative of z_1, we have

$$\begin{aligned}
\dot{z}_1 &= \dot{\hat{A}}_1 + \gamma\ddot{e}_2 \\
&= \dot{\hat{A}}_1 + \gamma\left(\ddot{q}_{2d} - v_1 + \frac{-A_1 z_1 v_1^2 - A_1 z_2 v_2 v_1 + z_2 A_1 A_2 \lambda_d v_1 - (1 + A_2^2)\lambda_d z_1 v_1}{1 + A_1^2 + A_2^2}\right)
\end{aligned}$$

We design the update law of $\hat{A}_1$ as

$$\begin{aligned}
\dot{\hat{A}}_1 &= \gamma(v_1 - \ddot{q}_{2d} + F_{2d} - F_{e2}) \\
&= \gamma(v_1 - \ddot{q}_{2d} + A_1(\lambda_d - \lambda)) && \text{(5.118)}
\end{aligned}$$

Thus $\dot{z}_1$ becomes

$$\begin{aligned}
\dot{z}_1 &= \gamma\left(\frac{-A_1 z_1 v_1^2 - A_1 z_2 v_2 v_1 + z_2 A_1 A_2 \lambda_d v_1 - (1 + A_2^2)\lambda_d z_1 v_1}{1 + A_1^2 + A_2^2} + A_1(\lambda_d - \lambda)\right) \\
&= \gamma\frac{-A_1 z_1 v_1 - A_1 z_2 v_2 + z_2 A_1 A_2 \lambda_d - (1 + A_2^2)\lambda_d z_1}{1 + A_1^2 + A_2^2} \\
&\quad - \gamma A_1 \frac{A_1 z_1 \lambda_d + A_2 z_2 \lambda_d - v_1 z_1 - v_2 z_2}{A_1^2 + A_2^2 + 1} \\
&= -\gamma\lambda_d z_1 && \text{(5.119)}
\end{aligned}$$

Similarly, we design $\dot{\hat{A}}_2$ as

$$\dot{\hat{A}}_2 = \gamma(v_2 - \ddot{q}_{3d} + A_2(\lambda_d - \lambda)) \tag{5.120}$$

Thus $\dot{z}_2$ becomes

$$\dot{z}_2 = -\gamma\lambda_d z_2 \tag{5.121}$$

Based on (5.119) and (5.121), we can conclude that the z subsystem is asymptotically stable. The closed-loop dynamics is presented here for convenience

$$A_1(\ddot{e}_2 + k_v\dot{e}_2 + k_p e_2) + A_2(\ddot{e}_3 + k_v\dot{e}_3 + k_p e_3) = -z_1 v_1 - z_2 v_2 + \lambda_d - \lambda \tag{5.122a}$$

$$(1 + A_1^2 + A_2^2)(\ddot{e}_2 + k_v\dot{e}_2 + k_p e_2) = -A_1 z_1 v_1 - A_1 z_2 v_2 + z_2 A_2 A_1 \lambda_d - (1 + A_2^2)\lambda_d z_1 \tag{5.122b}$$

$$(1 + A_1^2 + A_2^2)(\ddot{e}_3 + k_v\dot{e}_3 + k_p e_3) = -A_2 z_1 v_1 - A_2 z_2 v_2 + z_1 A_2 A_1 \lambda_d - (1 + A_1^2)\lambda_d z_2 \tag{5.122c}$$

Since z subsystem is asymptotically stable, from (5.122b) and (5.122c), we can conclude that

$$\lim_{t\to\infty} e_2 = 0, \quad \lim_{t\to\infty} e_3 = 0 \tag{5.123}$$

Thus, from (5.122a), we can conclude that

$$\lim_{t\to\infty} \lambda = \lambda_d \tag{5.124}$$

Also since z, $\dot{e}_2$ and $\dot{e}_3$ converge to zero, we have

$$\lim_{t\to\infty} \hat{A}_1 = A_1, \quad \lim_{t\to\infty} \hat{A}_2 = A_2 \tag{5.125}$$

Here we establish the adaptive hybrid force-motion controller for UAM interacting with a rigid plane surface. This controller is designed for the case when the UAM has 3 DoF force measurements. The control gain k_p, k_v are easy to tune and the adaptive gain γ can be tuned to change the convergence speed of z subsystem.

5.5 Simulation Results

The hexarotor UAV design used in the UAM simulations is based on Tarot 680 experimental platform, and its parameters are given in Table 5.1. The UAV is equipped with a 3D force sensor at the end-effector. The mass of the UAV is about 2.5 kg. The control gains are given in Table 5.2.

Table 5.1: Tarot UAM parameters.

Parameter	Value	Parameter	Value
m_Q	2.5 kg	J_{xx}	0.05 kg m^2
J_{yy}	0.05 kg m^2	J_{zz}	0.08 kg m^2
J_{xy}	0.0 kg m^2	J_{yz}	0.0 kg m^2
J_{zx}	0.0 kg m^2	$[x_e, y_e, z_e]$	$[0.6, 0, 0]$ m

First, we present the simulation result of free motion of Tarot 680 UAM. The

Table 5.2: Control Gains

Parameter	Value	Parameter	Value
k_v	4	k_p	4
γ	0.95	K_R	$\operatorname{diag}(30,30,30)$
k_ω	$\operatorname{diag}(25,25,25)$		

initial point is $p_Q = (0,0,0), (\phi,\theta,\psi) = (0,0,0)$ and the set point is $p_Q = (1,2,3)$ m, $(\phi,\theta,\psi) = (0.1,0.2,0.3)$. The simulation result is shown in Figure 5.10 and Figure 5.11. The result shows that the free motion controller can stabilize the UAM to the set point in fast and smooth way. The control inputs are also smooth and bounded.

In the second part of simulation, we test the I&I adaptive hybrid force-motion controller for UAM interacting with a rigid plane surface. The initial point is $p_E = (0,0,0)$ m, $(\phi,\theta,\psi) = (0,0,0)$ and the set point is $p_E = (-7,1,2)$ m, $(\phi,\theta,\psi) = (0.1,0.2,0.3)$. The desited force $\lambda_d = 5$ N. The constraint surface is set as

$$p_{E1} + p_{E2} + 3p_{E3} = 0 \tag{5.126}$$

which means $A_1 = 1, A_2 = 3, b = 0$. The simulation result is shown in Figure 5.12, Figure 5.13. The result shows that the I&I adaptive hybrid force-motion controller can stabilize the UAM to the set point in less than 5 seconds. The control inputs are in reasonable range. The reaction force λ also converge to the desired value λ_d. The unknown parameters $\hat{A}_1, \hat{A}_2$ also converge to the true value A_1, A_2 smoothly.

In the third part of simulation, we present the result of a figure-8 tracking task. The initial point is $p_E = (0,0,0)$ m, $(\phi,\theta,\psi) = (0,0,0)$. The desired trajectory is given as

$$p_{E2d} = x_2 = 2\sin t \text{ m} \tag{5.127}$$

$$p_{E3d} = x_3 = 2\sin t\cos t \text{ m} \tag{5.128}$$

The constraint surface is same as the second part. The desired force is set as $\lambda_d = 5$ N. The simulation result is shown in Figure 5.14, Figure 5.15. The 3D view of the tracking result is shown in Figure 5.16. We can conclude that the tracking error, the parameter estimation error, and the force tracking error will converge to zero. The control inputs are within reasonable range.

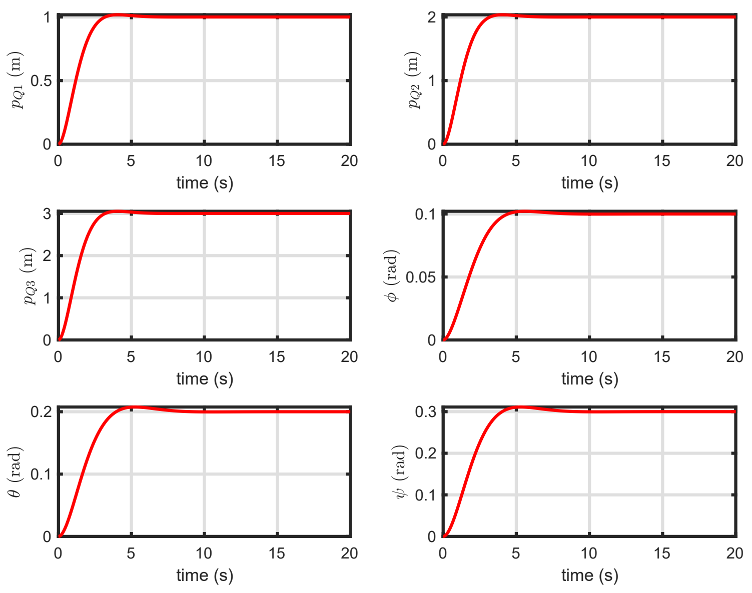


Figure 5.10: Stabilization configuration states of UAM free motion.

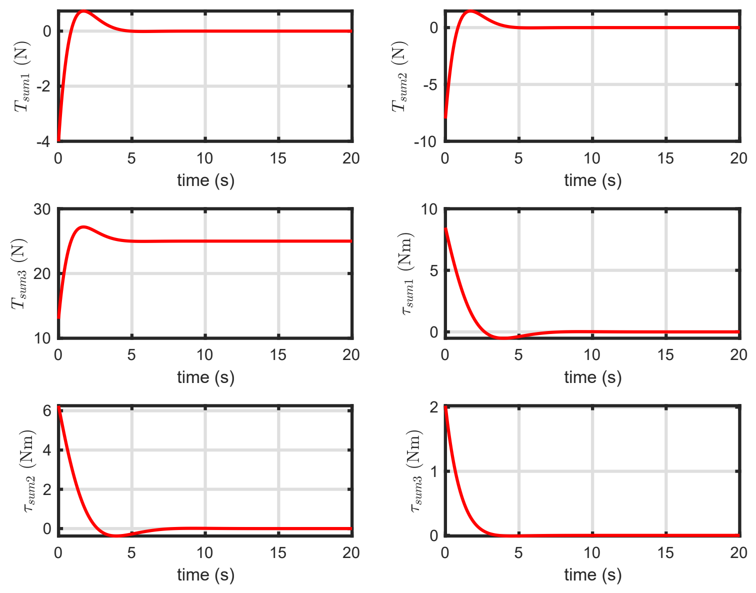


Figure 5.11: Stabilization control input of of UAM free motion.

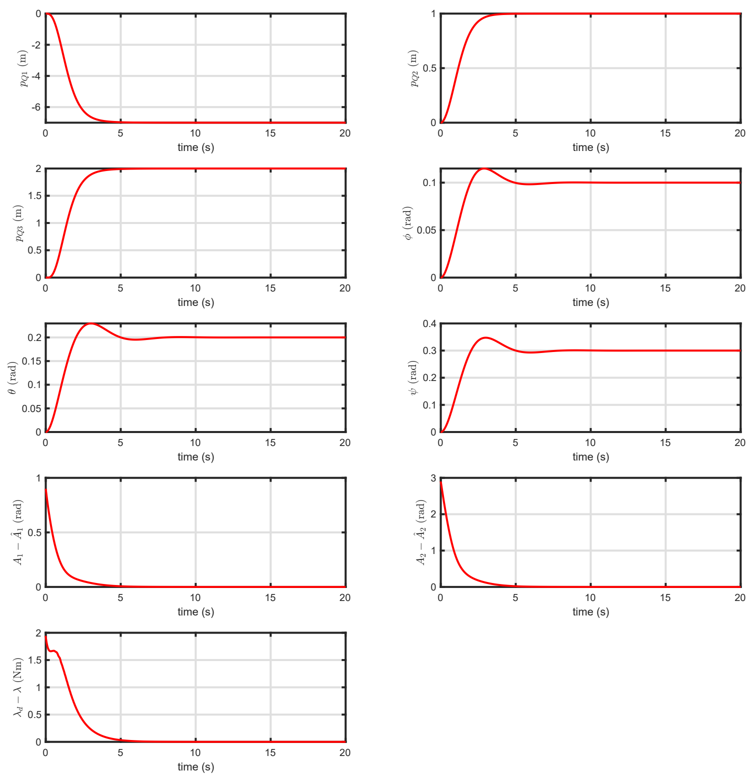


Figure 5.12: Stabilization configuration states.

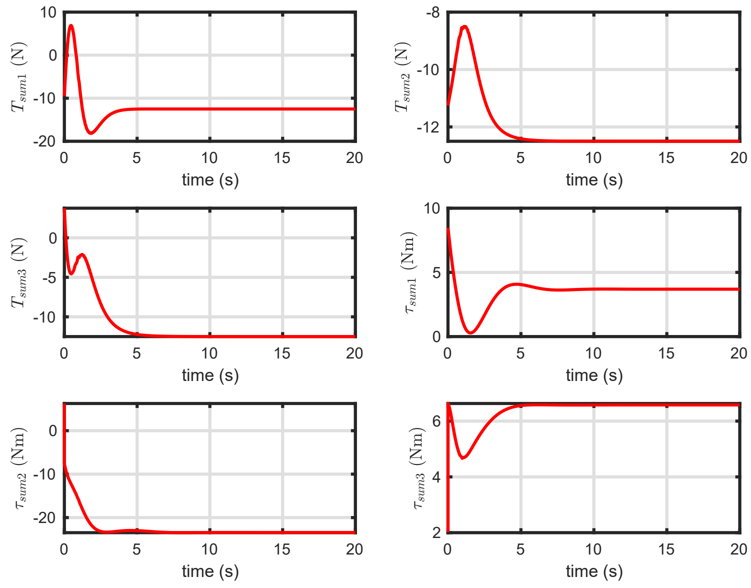


Figure 5.13: Stabilization control input of UAM.

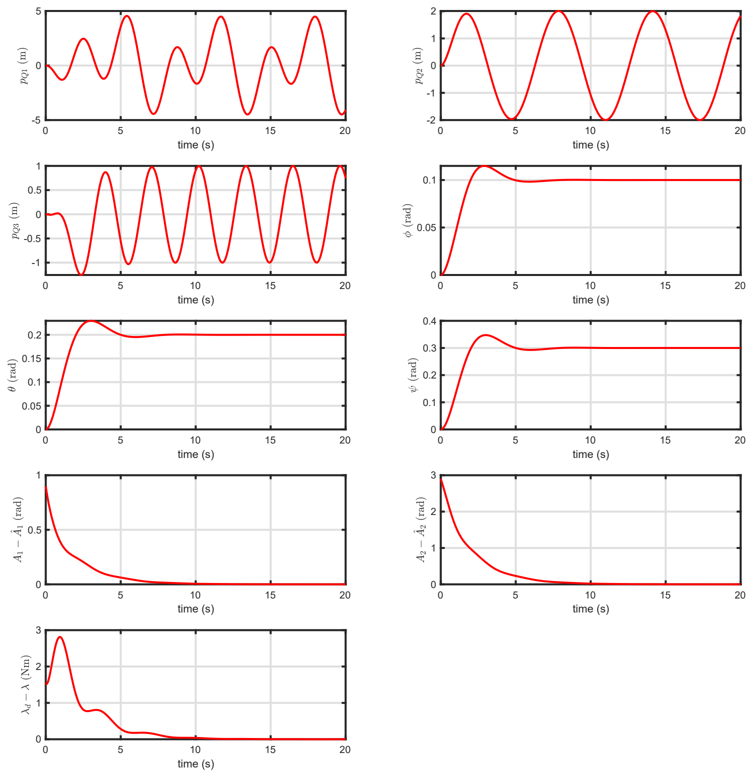


Figure 5.14: Figure-8 tracking configuration states.

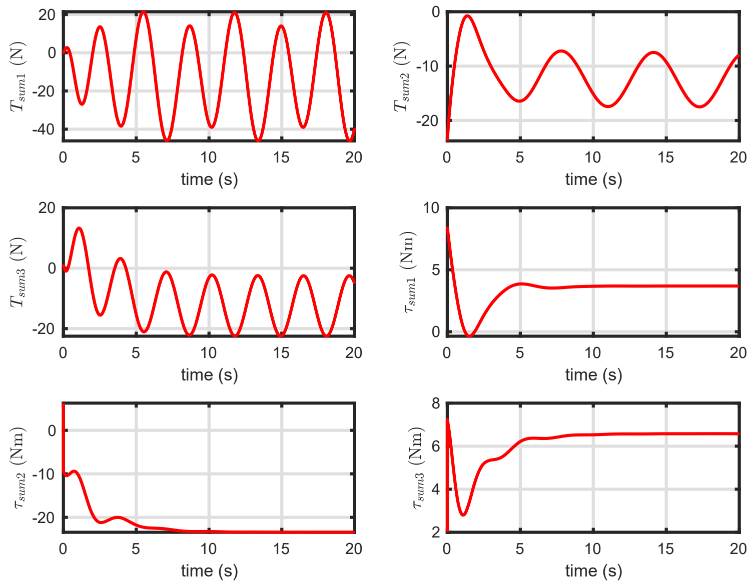


Figure 5.15: Figure-8 tracking control input of UAM.

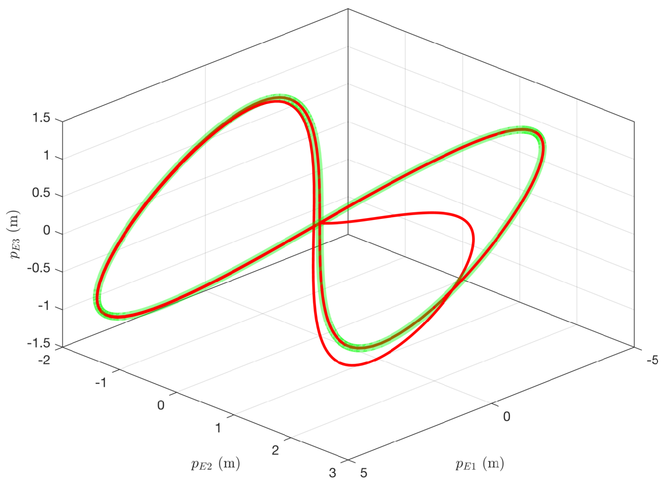


Figure 5.16: Figure-8 tracking 3D view, green line is the desired trajectory, red line is the actual trajectory

5.6 Conclusion

In this section, we present the adaptive hybrid force-motion controller for UAM interacting with a rigid plane surface. We first introduce the I&I control framework and necessary assumptions. Then, an explanatory example is given to explain how the I&I adaptive control techniques can be applied in hybrid force-motion control problem. We also compare the I&I adaptive control with the conventional adaptive control based on Lyapunov design. The simple example shows that the conventional adaptive control is not able to give asymptotically stable result and has no guarantee of parameter convergence. In contrast, I&I adaptive control overcomes these challenges by adding parameter observer states. The I&I adaptive control gives global asymptotically stabilisability results. We apply the I&I adaptive control to the UAM hybrid force-motion control problem. Specifically, we consider the interaction problem with two common force sensors, 1 DoF and 3 DoF force sensor. The I&I adaptive hybrid control guarantees the exponential convergence of both system states, reaction forces and parameter estimations. This is verified in simulation in Section. 5.5.

Chapter 6

Quadrotor Motion Control Using Deep Reinforcement Learning

6.1 Preliminaries

6.1.1 Markov Decision Process

6.1 shows the closed-loop system in an RL framework. We follow the notation of [137]. The learner or decision-maker is called an *agent*, which can be thought of as the controller in control systems. We remark that with the RL method considered here, the system operates in a training mode in which the controller design is performed adaptively based on system measurements. After this design stage is completed normal operation begins where the controller's parameters are fixed. The open-loop system or plant together with any disturbances is called the *environment*. The agent computes an action which influences the environment state. The environment also provides a reward signal which the agent maximizes over time. From a control systems perspective, the reward is included in the reference output which is usually generated in the controller. A discrete-time framework is adopted here as is customary with RL methods. For every time $t = 0, 1, 2, \dots$, the agent measures the environment state $S_t \in \mathcal{S}$ in order to compute an action $A_t \in \mathcal{A}$. Here, S_t, A_t denote random variables for each t. We take the state space $\mathcal{S} = \mathbb{R}^n$, action space $\mathcal{A} = \mathbb{R}^m$, and reward space $\mathcal{R} = \mathbb{R}$ where $n = 12, m = 4$ for the case of the quadrotor. A MDP is normally used to model the dynamics of the environment. The MDP is described by the function $p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ where

$$p(s', r|s, a) = \Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\} \tag{6.1}$$

where $s', s \in \mathcal{S}$, s' is the next state, $r \in \mathcal{R}$, $a \in \mathcal{A}$, and Pr denotes probability. Hence, (6.1) determines the probability a current state and reward occurs given a certain action and state at the previous time. Although quadrotor motion control is normally derived based on a deterministic ODE model, derivation of an RL method is performed in a stochastic framework.

6.1.2 Policy and Value Function

In this paper we use a stochastic policy $\pi(a|s)$ which is the probability that $A_t = a$ if $S_t = s$. In control systems terminology, policy π corresponds to the control law which is derived from the design method. In deep RL, the policy is parameterized using a neural net with a vector of parameters θ. The parameterized policy is $\pi_\theta(a|s)$ and the parameterization is described below in Section 6.2.2. The state-value function $v_\pi(s)$ of a state s under a policy π is the expected accumulated reward when starting in s and following policy π:

$$v_\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R_t | S_0 = s], \text{for all } s \in \mathcal{S} \tag{6.2}$$

where $\gamma < 1$ is the discounting factor and $\mathbb{E}_\pi[\cdot]$ denotes the expected accumulated reward following policy π.

Similar to v_π we introduce the action-value function $q_\pi(s,a)$ for policy π as

$$q_\pi(s,a) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R_t | S_0 = s, A_0 = a] \tag{6.3}$$

This function is the expected accumulated reward when starting in s, taking action a initially, and then following policy π. We remark that policy π is normally derived using q_π.

Solving a RL task means finding π that achieves a large v_π. Policy π, with corresponding v_π, is defined to be better than or equal to policy π', with corresponding v'_π, if $v_\pi(s) \geq v_{\pi'}(s)$ for all $s \in \mathcal{S}$. In this case we say $\pi \geq \pi'$. Policies better than or equal to all other policies are called optimal policies π_*. An optimal policy may not be unique and all optimal policies share the same value function which is called the optimal value function $v_*(s) = \max_\pi v_\pi(s)$.

In practice, the optimal policy and optimal value function are hard to find if the environment dynamics is unknown. The methods for finding the optimal policy without the knowledge of the environment dynamics are called model-free. The method described in this paper is model-free. Also, for continuous or very large state and action space, it is impossible to work in extremely large table settings. Hence, in such situations function approximation techniques are often used.

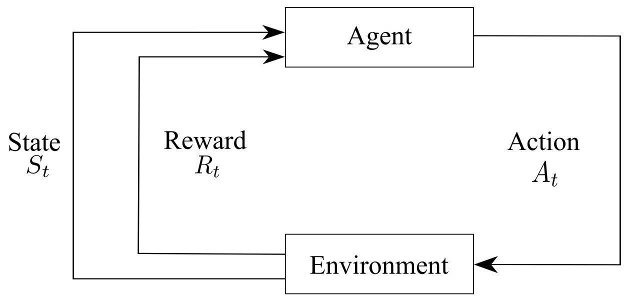


Figure 6.1: The agent-environment interaction

6.2 RL Method

The policy gradient method uses gradient descent optimization of a scalar performance measure $J_{\pi_\theta}(s)$ to determine optimal an optimal policy parameter θ. By introducing $J_{\pi_\theta}(s)$ we generalize the value function $v_\pi(s)$ and this is an important factor in improving convergence speed in optimization-based methods. According to the Policy Gradient Theorem [137, Sec. 13.2], the derivative of J_{π_θ} w.r.t. θ satisfies

$$\triangledown_\theta J_{\pi_\theta} \propto \sum_{a,s} q_\pi(s,a) \triangledown_\theta \pi_\theta(a|s) \tag{6.4}$$

Hence, the gradient ascent update to maximize J_{π_θ} is

$$\theta_{k+1} = \theta_k + \alpha \triangledown_{\theta_k} J_{\pi_{\theta_k}} = \theta_k + \alpha \sum_{a,s} q_\pi(s,a) \triangledown_{\theta_k} \pi_{\theta_k}(a|s) \tag{6.5}$$

where $\alpha > 0$ is the update step size. Although there is no convergence guarantee for the gradient method, it has been shown to perform in various benchmarks [138].

6.2.1 Parameterized Neural Policy

This Section gives the expression for the feedforward neural nets used. Notation is adopted from [139]. We consider the so-called *Actor Neural Net* for approximating policy π. An activation function is used to transform linear features to nonlinear features. We choose the tanh function:

$$g^{(1)}(s) = g^{(2)}(s) = \tanh s \tag{6.6}$$

where $g^{(k)}$ is evaluated component-wise on the quadrotor state s. The actor neural net is given by

$$h^{(1)}(s) = g^{(1)}(\theta^{(1)\top} s + \theta_b^{(1)}) \tag{6.7}$$

$$h^{(2)}(h^{(1)}(s)) = g^{(2)}(\theta^{(2)\top} h^{(1)}(s) + \theta_b^{(2)}) \tag{6.8}$$

$$\mu(s) = \theta^{(3)\top} h^{(2)}(h^{(1)}(s)) + \theta_b^{(3)} \tag{6.9}$$

$$\pi_\theta(a|s) = \frac{1}{\sqrt{2\pi}\sigma} \exp - \frac{(a-\mu(s))^\top (a-\mu(s))}{2\sigma^2} \tag{6.10}$$

The outputs of the first and second layers are denoted $h^{(1)}, h^{(2)}$, respectively. The dimension of $h^{(1)}, h^{(2)}$ is N which is a parameter to be adjusted. Thus, we have $\theta^{(1)} \in \mathbb{R}^{N\times n}$, $\theta^{(2)} \in \mathbb{R}^{N\times N}$, $\theta^{(3)} \in \mathbb{R}^{m\times N}$. The biases are denoted $\theta_b^{(k)}, k = 1, 2, 3$ with their dimension determined from (6.7)–(6.9). The output of the neural net determines a Gausian pdf in (6.10) which is used to evaluate π_θ. The standard deviation σ in (6.10) is taken as a small constant. A widely used σ is $\sigma = \exp(-\omega)$ where ω is usually chosen on $[0.5, 5]$ [138], [140]. Some methods treat σ as a NN parameter which can be tuned during training, e.g., [141].

Similarly, the so-called *Critic Neural Net* which approximates $\hat{v}_\zeta$ is given by

$$h^{(1)}(s) = g^{(1)}(\zeta^{(1)^T} s + \zeta_b^{(1)}) \tag{6.11}$$

$$h^{(2)}(h^{(1)}(s)) = g^{(2)}(\zeta^{(2)^T} h^{(1)}(s) + \zeta_b^{(2)}) \tag{6.12}$$

$$\hat{v}_\zeta(s) = h^{(3)}(h^{(2)}(h^{(1)}(s))) = \zeta^{(3)^T} h^{(2)}(h^{(1)}(s)) + \zeta_b^{(3)} \tag{6.13}$$

where ζ is the parameters for the neural net and $\zeta^{(1)} \in \mathbb{R}^{N\times n}$, $\zeta^{(2)} \in \mathbb{R}^{N\times N}$, $\zeta^{(3)} \in \mathbb{R}^{1\times N}$.

6.2.2 Proximal Policy Optimization (PPO)

PPO is a gradient descent method based on an Actor-Critic model. It is currently considered a state-of-art algorithm in the RL community [95]. It benefits from a relatively simple implementation and has shown promising performance in practice [142]. The innovation of PPO comes from the choice of performance measure

$$J_{\pi_\theta}(s) = \min(k_t(\theta)\hat{A}_t, \sigma(k_t(\theta))\hat{A}_t) \tag{6.14}$$

where $\hat{A}_t$ is the advantageous estimation [106] to the end of an episode of length T:

$$\hat{A}_t = \delta_t + \gamma\lambda\delta_{t+1} + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1}, \quad \lambda \in (0, 1], \tag{6.15}$$

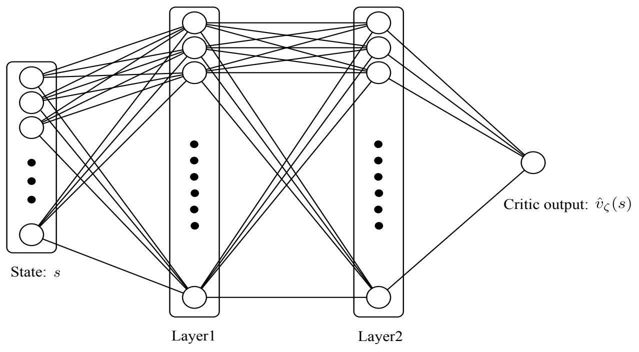


Figure 6.2: Critic Neural Net.

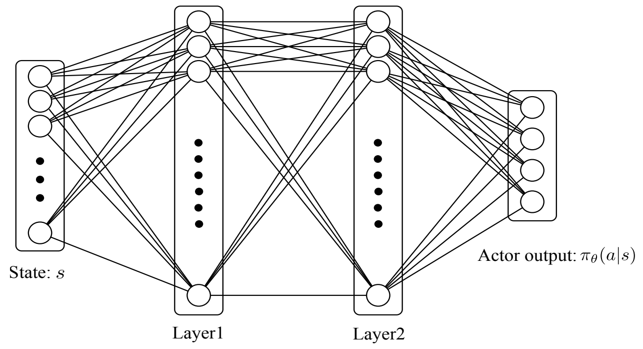


Figure 6.3: Actor Neural Net.

where

$$\delta_t = r_{t+1}(a, s) + \gamma \hat{v}_\zeta(s_{t+1}) - \hat{v}_\zeta(s_t) \tag{6.16}$$

with r_t being a deterministic reward function, and $\hat{v}_\zeta(s)$ denotes the estimated value of v_π. Function

$$k_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \tag{6.17}$$

describes the difference between the previous policy with parameter θ_{old} and the present policy with parameter θ as optimization proceeds. The saturation function $\sigma : \mathbb{R} \to \mathbb{R}$ is

$$\sigma(\xi) = \begin{cases} 1+\varepsilon, & \xi > 1+\varepsilon \\ \xi, & 1-\varepsilon \le \xi \le 1+\varepsilon \\ 1-\varepsilon, & \xi < 1-\varepsilon \end{cases} \tag{6.18}$$

where the clip ratio $\varepsilon > 0$ is a small value.

We remark that strictly speaking quadrotor stabilization is not an episodic task. However, taking a sufficiently large episode length T allows us to approximate a continuing task. A trajectory is a sequence of states and actions, denoted as $\tau = (s_0, a_0, s_1, a_1, \ldots, s_T)$. In (6.16), the estimated state value function $\hat{v}_\zeta(s)$ appears. The reason for introducing $\hat{v}_\zeta(s)$ and using $\hat{A}_t$ is to decrease the variance in the estimation of $\bigtriangledown_\theta J_\theta(s)$ and hence improve the convergence speed of training. Using the estimated value function $\hat{v}_\zeta(s)$ to help the convergence speed of policy function π_θ is called an Actor-Critic method. See [143] for details on the benefits of using an Actor-Critic structure.

Using the Policy Gradient Theorem (i.e., (6.4)), we can take the derivative of J_{π_θ} relative to θ by taking the derivative of $\pi_\theta(a|s)$ relative to θ. Hence, assuming $1 - \varepsilon \leq k_t(\theta) \leq 1 + \varepsilon$ we obtain

$$\bigtriangledown_\theta J_{\pi_\theta} = \sum_{t=0}^{T} \hat{A}_t \bigtriangledown_\theta k_t(\theta) \tag{6.19}$$

which is used to estimate the gradient of J_{π_θ} from sampled action and state data obtained from the agent and environment interaction. The update equation for θ is

$$\theta_{k+1} = \theta_k + \alpha \sum_{t=0}^{T} \hat{A}_t \bigtriangledown_\theta k_t(\theta) \tag{6.20}$$

It has been shown in [95] that PPO is a refinement of TRPO [144] as it improves training efficiency by constraining the difference between π_θ and $\pi_{\theta_{\text{old}}}$. The saturation function σ limits the magnitude of the gradient $\bigtriangledown_\theta k_t(\theta)$ and ensures the updated θ is close to its previous value θ_{old}.

In order to compute the estimated state value function $\hat{v}_\zeta$ in (6.16) we optimize the following loss function for the critic neural net:

$$L_\zeta = \sum_{t=0}^{T} (r(a_t, s_t) + \gamma \hat{v}_\zeta(s_{t+1}) - \hat{v}_\zeta(s_t))^2 \tag{6.21}$$

The loss function minimizes the one-step look ahead error in $\hat{v}_\zeta$. As with (6.19), sampled data (s, a, r) is used in (6.21) to estimate the error of $\hat{v}_\zeta$. The update for ζ is

$$\zeta_{k+1} = \zeta_k - \beta \bigtriangledown_\zeta L_\zeta \tag{6.22}$$

where β is step size.

The pseudocode code of the RL method is in Algorithm 1. Since the update step size α, β in (6.20) and (6.22) are usually set very small, to accelerate the training process, we usually update parameter θ, ζ multiple times for the same set of data. After the training process is finished then the quadrotor's motion can be controlled using the policy $\pi_\theta(a|s)$ which maps the measured state to a four-dimensional Gaussian distribution which can be sampled to determine the physical inputs of quadrotor.

Algorithm 1: Proximal Policy Optimization (PPO)

```
1  Input: a differentiable policy parameterization $\pi_\theta(a|s)$;
2  Input: a differentiable estimated state value function parameterization
     $\hat{v}_\zeta(s)$;
3  for $j = 0, 1, 2, \dots$ do
4      Run policy $\pi_\theta$ for $K$ timesteps and collect $\{s_t, a_t, r_t\}$. Collect
         trajectories $D_j = \{\tau_i\}$. ;
5      Calculate $\delta_t, \hat{A}_t$ using (6.16) ;
6      for $k$ = 1:M do
7          Calculate $\bigtriangledown_\theta J_{\pi_\theta}$ with the collected $\{s, a, r\}$ with (6.19);
8          Update $\theta$ with using (6.20) ;
9          $$\theta_{k+1} = \theta_k + \alpha \frac{1}{|D_j|T} \sum_{\tau \in D_j} \sum_{t=0}^{T} \hat{A}_t \bigtriangledown_\theta k_t(\theta)$$
10     end
11     for $k$ = 1:B do
12         Calculate $\bigtriangledown_\zeta L_\zeta$ with the collected $\{s, a, r\}$ data with (6.21) ;
13         Update $\zeta$ using (6.22) ;
14         $$\zeta_{k+1} = \zeta_k - \beta \frac{1}{|D_j|T} \sum_{\tau \in D_j} \bigtriangledown_\zeta L_\zeta$$
15     end
16 end
```

6.3 Experiment

6.3.1 Simulated Quadrotor Dynamics

We consider a traditional quadrotor UAV as shown in Figure 6.4. Two reference frames are needed for the modelling: a fixed inertial navigation frame $\mathcal{N}$ with orthonormal basis $\{n_1, n_2, n_3\}$ and a body frame $\mathcal{B}$ whose origin is at the vehicle's center of mass (CoM) and with orthonormal basis $\{b_1, b_2, b_3\}$. We define b_1 to point in the forward direction of vehicle, b_2 pointing right, and b_3, n_3 pointing down. The configuration of the quadrotor belongs to the special Euclidean group SE(3), and includes the position $p = [p_1, p_2, p_3]^\top \in \mathbb{R}^3$ of the origin of $\mathcal{B}$ relative to $\mathcal{N}$, and the rotation matrix $R \in \mathrm{SO}(3)$ which describes the orientation of $\mathcal{B}$ and $\mathcal{N}$. $\eta = [\phi, \theta, \psi] \in \mathbb{R}$ is Euler-angles corresponding to the rotation matrix. We assume each propeller generates thrust in the $-b_3$ direction and denote the total thrust due to all propellers by the scalar input $u \geq 0$. Controlling individual propeller speeds creates an input torque denoted $\tau = [\tau_1, \tau_2, \tau_3]^\top \in \mathbb{R}^3$ which is expressed in $\mathcal{B}$. To ease the presentation of the control design we take torque τ and thrust u as system

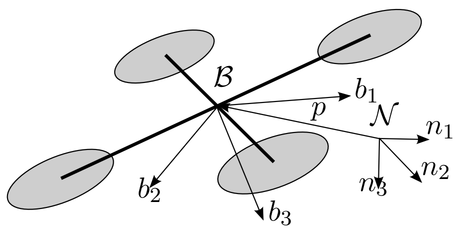


Figure 6.4: Quadrotor modelling and reference frames

inputs. However, in practice the physical input to the UAV is PWM signals to the ESC.

The UAV dynamics is

$$\dot{p} = v \tag{6.23a}$$
$$m\dot{v} = mgn_3 - uRn_3 \tag{6.23b}$$
$$\dot{R} = RS(\omega) \tag{6.23c}$$
$$J\dot{\omega} = -\omega \times J\omega + \tau \tag{6.23d}$$

where $v \in \mathbb{R}^3$ is linear velocity expressed in $\mathcal{N}$, $\omega \in \mathbb{R}^3$ is angular velocity in $\mathcal{B}$, m is mass, J is inertia, g is the gravity constant, and $n_3 = [0, 0, 1]^\top$. The skew operator $S(\cdot) : \mathbb{R}^3 \to so(3)$ is given by

$$S(x) = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}, \quad \text{where } x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}.$$

The simulator uses a discretized system model

$$v_{t+1} = v_t + \dot{v}_t \mathrm{d}t \tag{6.24a}$$
$$\omega_{t+1} = \omega_t + \dot{\omega}_t \mathrm{d}t \tag{6.24b}$$
$$p_{t+1} = p_t + v_t \mathrm{d}t + \frac{1}{2}\dot{v}_t \mathrm{d}t^2 \tag{6.24c}$$
$$\eta_{t+1} = \eta_t + W(\eta_t)\omega_t \mathrm{d}t + \frac{1}{2}W(\eta_t)\dot{\omega}_t \mathrm{d}t^2 \tag{6.24d}$$
$$W(\eta) = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{cos\theta} \end{bmatrix} \tag{6.24e}$$

which has a sampling interval of $\mathrm{d}t$. This model assumes a constant $\dot{\omega}$ and $\dot{v}$ between sampling instants. The simulator runs at $\mathrm{d}t = 1\,\mathrm{ms}$. The parameters for the quadrotor are $m = 0.5\,\mathrm{kg}$, $g = 9.81\,\mathrm{m/s^2}$, $J = \mathrm{diag}(J_1, J_2, J_3)$, $J_1 = 0.0135\,\mathrm{kg\,m^2}$,

$J_2 = 0.0135\,\mathrm{kg\,m^2}$, and $J_3 = 0.024\,\mathrm{kg\,m^2}$.

6.3.2 Reward Design

An important factor in the RL-based control's motion control performance is the choice of dependence of the reward function $r(a, s)$ on control input $a = [u, \tau^\top]^\top$ and $s = [p^\top, \eta^\top, v^\top, \omega^\top]^\top$. In this chapter, we considered three different reward designs to investigate the influence of reward function on closed-loop performance:

$$r_a(a, s) = r_0 - \|v\|_2 - \|\omega\|_2 \tag{6.25}$$

$$r_b(a, s) = r_0 - \|v\|_2 - \|\omega\|_2 - \|p\|_2 \tag{6.26}$$

$$r_c(a, s) = r_0 - \|v\|_2 - \|\omega\|_2 - \|p\|_2 - \|\tau\|_2 \tag{6.27}$$

$$r_0 = \begin{cases} 1, & -0.5 \leq p_3 \leq 0.5 \\ -1, & \text{otherwise} \end{cases} \tag{6.28}$$

r_0 used here is to accelerate the training process. Reward r_a aims to control hover using only velocity. Reward r_b includes an extra term $\|p\|_2$ to reduce drift in position due to unmodelled disturbances. Reward r_c includes $\|\tau\|_2$ to limit control effort. The yaw rate of the vehicle is regulated to zero by including the angular velocity in the reward.

6.3.3 Simulation Results

Training

In order to optimize training, the hyperparameters λ, ε, and learning rates α, β were adjusted for reward function r_c. Parameter λ appears in the generalized advantage estimator (6.15). As $\lambda \to 1$ the bias of the estimator decreases. Although the bias will increase as $\lambda \to 0$, smaller λ might accelerate the training process. Parameter ε is the clip ratio in (6.18). It constrains the difference between the old and new policy. A grid of parameter values was created from $\lambda \in \{0.35, 0.65, 0.95\}$, $\varepsilon \in \{0.1, 0.2, 0.3\}$, and $(\alpha, \beta) \in \{(3 \times 10^{-3}, 1 \times 10^{-2}), (3 \times 10^{-4}, 1 \times 10^{-3}), (3 \times 10^{-5}, 1 \times 10^{-4})\}$. This yielded 27 different parameter combinations. The training was defined to fail if there is convergence to a local maximum which is unable to stabilize the quadrotor. The hyperparameter search led to successful training when $\lambda = 0.95$ for certain values of $\varepsilon, \alpha, \beta$ as indicated in Table 6.1. The average cumulative reward for a single trajectory collected in the training process using 6 different random seeds is denoted V_{avg}, and its value is shown in Figure 6.6 for the 5 cases of successful training. Parameter Set 3 has the fastest convergence speed and achieves a relatively high V_{avg} at around 500 iterations. It is therefore chosen for the final

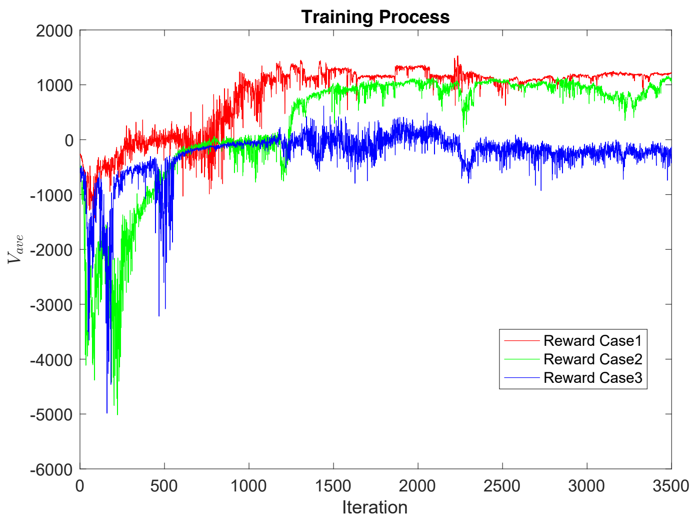


Figure 6.5: Training progress for V_{avg} for different rewards

Hyperparameter Set	$\{\lambda, \varepsilon, \alpha, \beta\}$
1	$\{0.95, 0.3, 3 \times 10^{-4}, 1 \times 10^{-3}\}$
2	$\{0.95, 0.3, 3 \times 10^{-5}, 1 \times 10^{-4}\}$
3	$\{0.95, 0.2, 3 \times 10^{-4}, 1 \times 10^{-3}\}$
4	$\{0.95, 0.2, 3 \times 10^{-5}, 1 \times 10^{-4}\}$
5	$\{0.95, 0.1, 3 \times 10^{-4}, 1 \times 10^{-3}\}$

Table 6.1: Hyperparameters sets corresponding to successful training. Training progress is shown in Figure 6.6

training process. The remaining design parameter values used are in Table 6.2. The training process is about 12 hours and is run on a single core of a E5-2683 v4 Intel Broadwell CPU.

Training progress for the three reward functions is shown in Figure 6.5. The average cumulative reward V_{avg} for a single trajectory using 6 different random seeds. We observe V_{avg} for r_a and r_b converge to a higher value than r_c. This is expected as r_c includes position error and control input as negative reward. The learning rate for r_c is the fastest and steady state is achieved at around 500 iterations. Although the algorithms can achieve convergence for the three reward cases, the time domain performances varies between different reward functions.

Performance Testing

Figures 6.7–6.10 present the simulation results after training using the three reward functions. As well, a traditional manually tuned inner-outer loop PD control

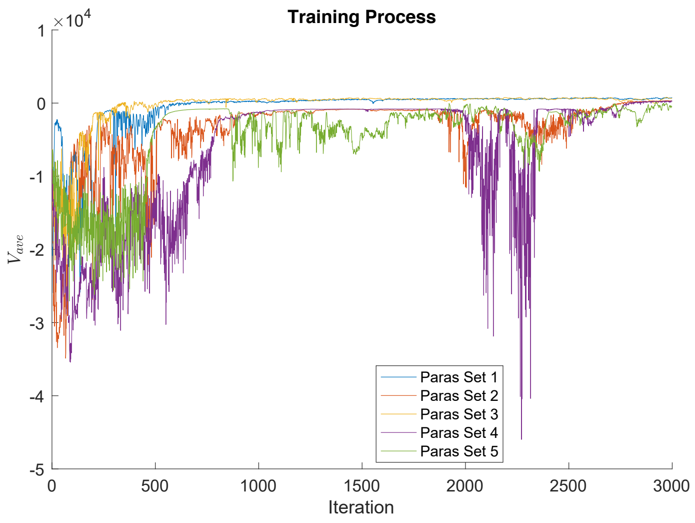


Figure 6.6: Training progress for V_{avg} for different hyperparameters

Parameters	Values
α	3×10^{-4}
β	1×10^{-3}
γ	0.99
λ	0.95
ε	0.2
K	4000
M	80
B	80
T	1000
N	64

Table 6.2: Algorithm Parameters

is provided for comparison. This control is defined by

$$\begin{aligned}
\phi_r &= k^o_{1p}(p^r_1 - p_1) + k^o_{1d}\frac{d}{dt}(p^r_1 - p_1) \\
\theta_r &= k^o_{2p}(p^r_2 - p_2) + k^o_{2d}\frac{d}{dt}(p^r_2 - p_2) \\
\tau_1 &= k^i_{1p}(\phi_r - \phi) + k^i_{1d}\frac{d}{dt}(\phi_r - \phi) \\
\tau_2 &= k^i_{2p}(\theta_r - \theta) + k^i_{2d}\frac{d}{dt}(\theta_r - \theta) \\
\tau_3 &= -k_{3p}\psi \\
u &= k_{hp}(p^r_3 - p_3) + k_{hd}\frac{d}{dt}(p^r_3 - p_3)
\end{aligned} \tag{6.29}$$

where ϕ_r, θ_r are roll and pitch set points from the outer loop. The control gains are chosen as $k^o_{1p} = k^o_{2p} = k^i_{1p} = k^i_{2p} = 0.07, k^o_{1d} = k^o_{2d} = k^i_{1d} = k^i_{2d} = 0.09, k_{3p} = 0.1, k_{hp} = 2, k_{hd} = 8$.

The control objective is to stabilize the position $[0, 0, 0]^\top$ m with the vehicle initialized to $p(0) = [1, 1, 0]^\top$ m. The results are given in Figures 6.7–6.10. We observe for r_a that velocity is regulated to zero and roll and pitch have a small oscillation in steady-state. These oscillations also appear in the inputs. As expected, there is a large steady-state position error as r_a does not measure this error. Using r_b the amplitude of steady-state position error is reduced relative to r_a since it includes position in the reward. The amplitude of steady-state oscillation in input and roll/pitch is increased. This is likely due to position being added to r_b. Reward r_c leads to the best performance with the smallest steady-state input and roll/pitch oscillation. The increased steady-state error position compared to r_b is due to control input added.

A time-varying trajectory tracking is tested using r_c and compared with the PD controller. A figure-8 reference trajectory $p_r = [1.5\sin(2t) + 1,\ 0.75\sin(4t),\ -4 + 2\sin(2t)]^\top$ is used. The tracking error is shown in Figure 6.11. The RL controller performs better than the PD controller designed for hover. Hence, the proposed method exhibits performance in tasks more general than what it was trained for.

6.4 Conclusion

We applied the deep RL method PPO to control the full 6 DoF system dynamics of a quadrotor UAV. Relative to the existing method, the proposed method considers the *full dynamics* of the UAV and this makes the design challenging. The work explored the effect of reward functions on closed-loop performance of the trained neural net controller. We observed that although different rewards functions can achieve a stable hover. Including input effort into the reward makes the closed-loop less sensitive (e.g., reduced impractical oscillation in the input signals). A simulation comparison of the proposed RL-based control and a classical inner-outer loop PD controller was presented. The results demonstrate similar hover stabilization and output tracking performance without requiring manual tuning.

Future work includes testing the control in actual flight, improving the training speed of the algorithm, and developing a systematic procedure for designing the reward function. We investigate methods which provide monotone performance improvement during training. Finally, we plan to develop algorithms which are robust to parameter updates. This will enable on-line training and improve robustness to environment change.

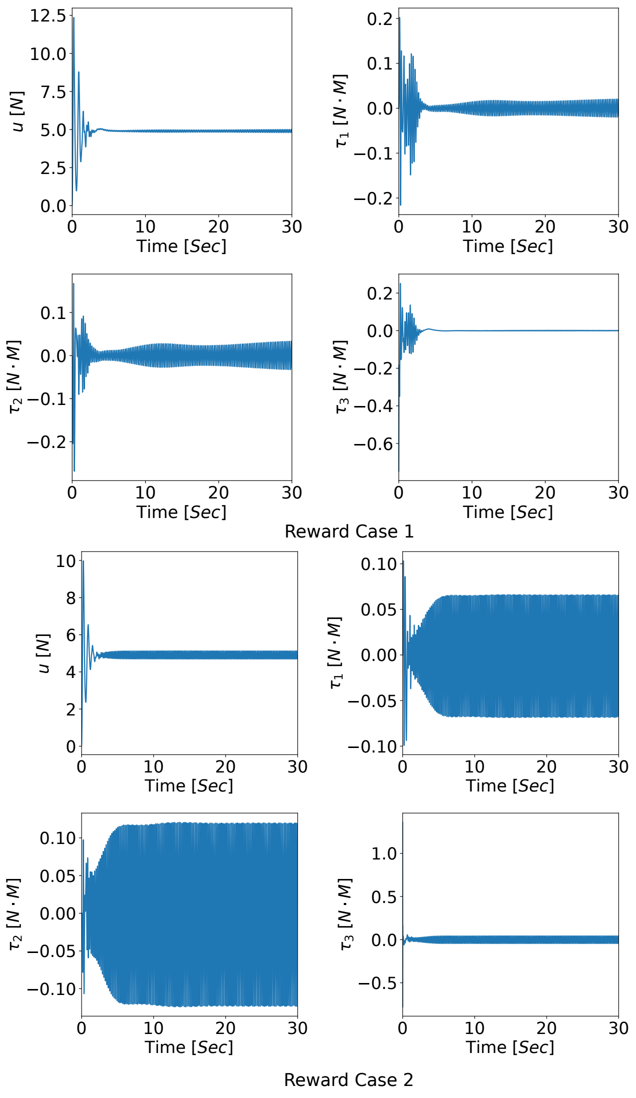
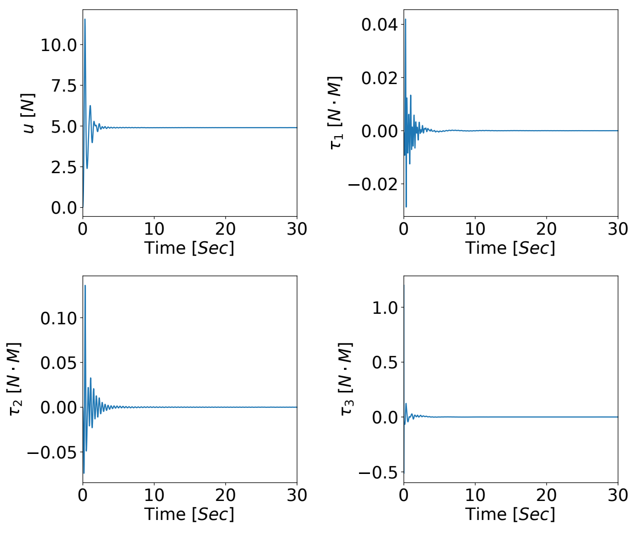
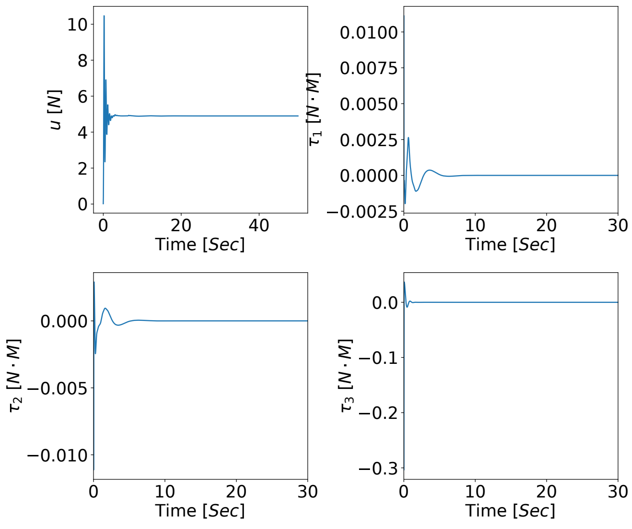


Figure 6.7: Simulation Results for hover stabilization. Input trajectories



Reward Case 3



PD Control

Figure 6.8: Simulation Results for hover stabilization. Input trajectories

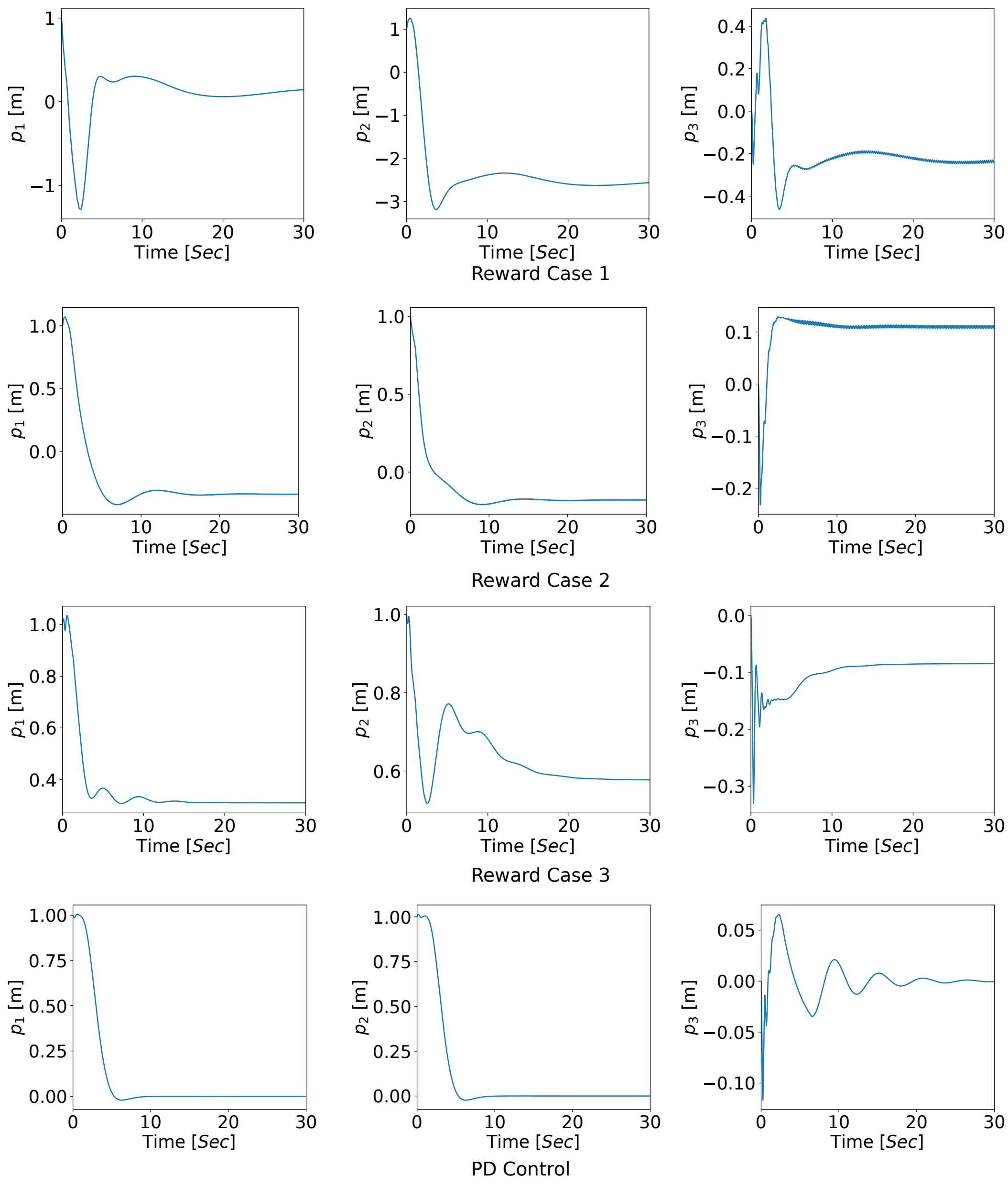


Figure 6.9: Simulation Results for hover stabilization. Position trajectories

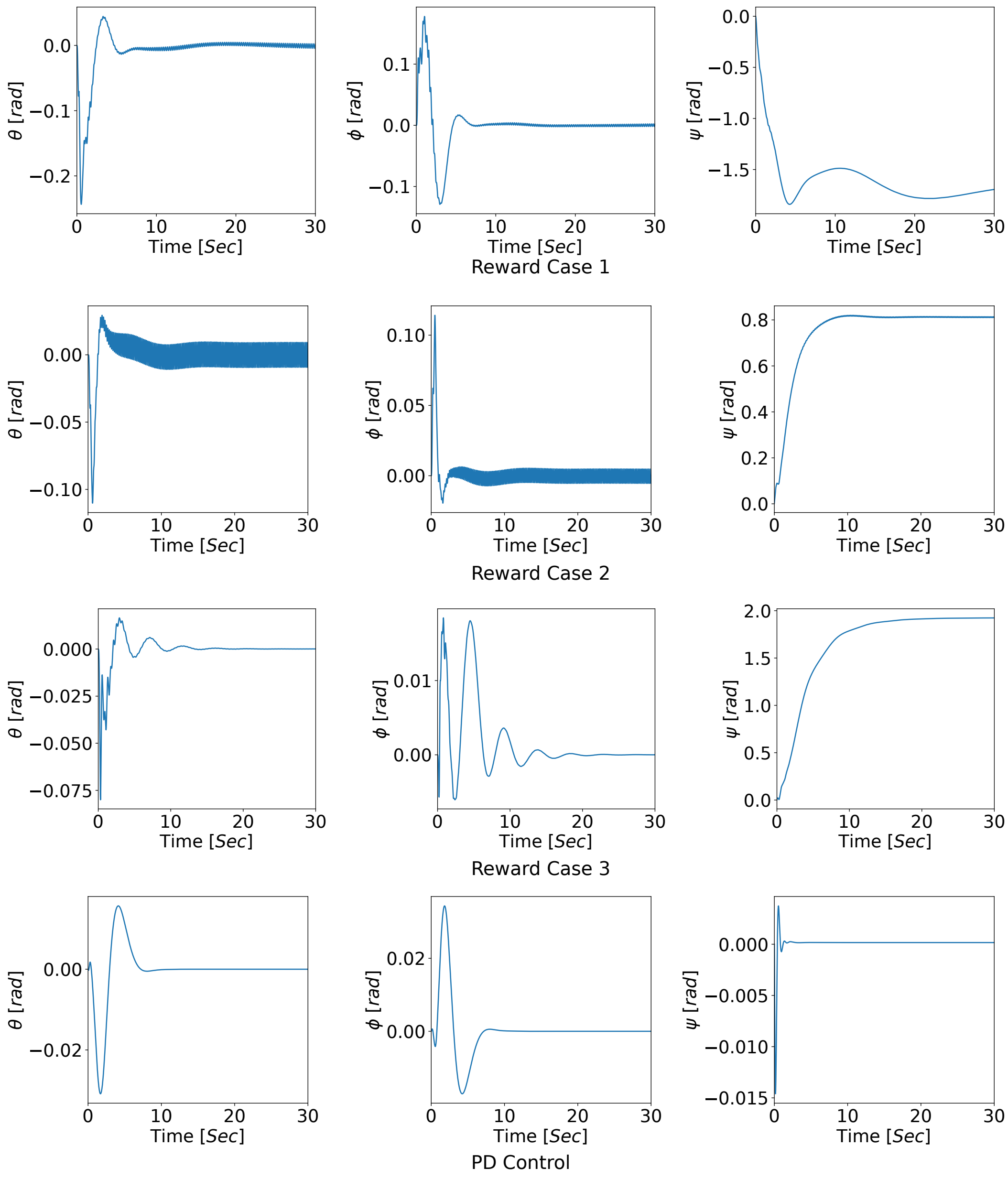


Figure 6.10: Simulation Results for hover stabilization. Attitude trajectories

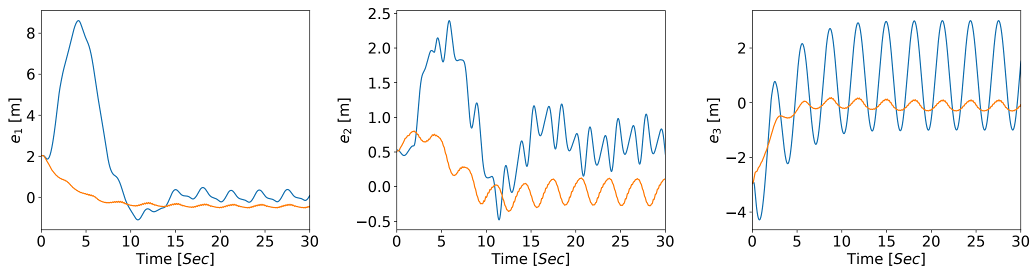


Figure 6.11: Simulation Results for Trajectory Tracking Error(Orange line is for NN control based on r_c, Blue line is for PD Controller).

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This thesis has significantly contributed to the field of unmanned aerial vehicle (UAV) and autonomous navigation systems, focusing on the ANCL Gen2 flight platform and related technologies. The development of the QSFA (QSF Algorithm) for nonlinear control affine systems stands as a notable advancement, addressing a gap in the existing literature. Its application to the SLS, compared with the Dynamic Extension Algorithm, provides a practical method for testing system flatness, diverging from the intricate theoretical approaches prevalent in current research.

The implementation of QSF in the PX4 software-in-the-loop environment demonstrated the robustness of the developed control law, particularly against model errors in simulation environments. This practical application is supported by the availability of the code and a demonstration video, enhancing the reproducibility and transparency of the research.

The thesis introduces a simplified multi-loop control approach, demonstrating its efficiency over more complex single-loop QSF control methods. The experimental validation of this approach using an open-source drone platform highlights its practicality and accessibility.

In the realm of adaptive hybrid force-motion control, the exploration of the I&I adaptive control within the context of UAM interaction with rigid plane surfaces has shown promising results, overcoming limitations found in conventional adaptive control methods.

The novel PPO-based method for motion control challenges traditional approaches by controlling all six UAV DoF with low-level system inputs, without relying on traditional control assistance. This focus on reward function design in RL and its impact on closed-loop performance marks a significant contribution to the field.

Furthermore, the upgrade from ANCL Q3 to the ANCL Gen2 platform, with its enhanced usability, maneuverability, and stability, along with the update in the PX4 autopilot system, exemplifies the practical application of theoretical research in a real-world environment.

7.2 Future Work

Future research should build upon these foundations to further advance UAV and autonomous navigation system capabilities. This includes further development of the QSFA in various control scenarios, expanded use of SITL simulations in complex environments, and advanced studies in multi-loop control approaches.

Broadening the application of I&I adaptive control to various types of UAV interactions, expanding the interaction from unknown planes to general unknown curved surfaces, and deepening research in reinforcement learning with a variety of algorithms and reward structures could lead to more exciting developments.

Moreover, continued development and sharing of open-source tools, along with active community engagement, will ensure that the research remains accessible, reproducible, and relevant, thereby contributing significantly to the advancement of the field and its applications across various domains.

Bibliography

[1] D. K. D. Villa, A. S. Brandão, and M. Sarcinelli-Filho, “A survey on load transportation using multirotor UAVs,” *Journal of Intelligent & Robotic Systems*, vol. 98, no. 2, pp. 267–296, Oct. 2020. DOI: 10.1007/s10846-019-01088-w.

[2] G. Yu, D. Cabecinhas, R. Cunha, and C. Silvestre, “Aggressive maneuvers for a quadrotor-slung-load system through fast trajectory generation and tracking,” *Autonomous Robots*, vol. 46, no. 4, pp. 499–513, Mar. 2022. DOI: 10.1007/s10514-022-10035-y.

[3] K. Sreenath, T. Lee, and V. Kumar, “Geometric control and differential flatness of a quadrotor UAV with a cable-suspended load,” in *Proceedings of the 52nd IEEE Conference on Decision and Control*, Firenze, Italy, Dec. 2013. DOI: 10.1109/cdc.2013.6760219.

[4] H. Li, H. Wang, C. Feng, F. Gao, B. Zhou, and S. Shen, “Autotrans: A complete planning and control framework for autonomous uav payload transportation,” *IEEE Robotics and Automation Letters*, vol. 8, no. 10, pp. 6859–6866, 2023. DOI: 10.1109/LRA.2023.3313010.

[5] T. Lee, “Geometric control of multiple quadrotor uavs transporting a cable-suspended rigid body,” in *53rd IEEE Conference on Decision and Control*, 2014, pp. 6155–6160. DOI: 10.1109/CDC.2014.7040353.

[6] M. Bernard and K. Kondak, “Generic slung load transportation system using small size helicopters,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, Kobe, Japan, 2009, pp. 3258–3264. DOI: 10.1109/ROBOT.2009.5152382.

[7] D. Sanalitro, H. J. Savino, M. Tognon, J. Cortes, and A. Franchi, “Full-pose manipulation control of a cable-suspended load with multiple UAVs under uncertainties,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2185–2191, Apr. 2020. DOI: 10.1109/lra.2020.2969930.

[8] H. B. Khamseh, F. Janabi-Sharifi, and A. Abdessameud, “Aerial manipulation—a literature survey,” *Robotics and Autonomous Systems*, vol. 107, pp. 221–235, Sep. 2018. DOI: 10.1016/j.robot.2018.06.012.

[9] F. Ruggiero, V. Lippiello, and A. Ollero, “Aerial manipulation: A literature review,” *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1957–1964, Jul. 2018. DOI: 10.1109/LRA.2018.2808541.

[10] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, "Flatness and defect of non-linear systems: Introductory theory and examples," *International Journal of Control*, vol. 61, no. 6, pp. 1327–1361, Jun. 1995. DOI: 10.1080/00207179508921959.

[11] P. Martin, R. Murray, and P. Rouchon, "Flat systems," *Plenary Lectures and Mini-Courses 4th European Control Conference*, pp. 1–55, Jan. 1997.

[12] J. Rudolph, *Flatness-based control: an introduction.* Saarbrücken: Shaker Verlag, 2021.

[13] E. Delaleau and J. Rudolph, "Control of flat systems by quasi-static feedback of generalized states," *International Journal of Control*, vol. 71, no. 5, pp. 745–765, 1998. DOI: 10.1080/002071798221551.

[14] F. Nicolau and W. Respondek, "Flatness of multi-input control-affine systems linearizable via one-fold prolongation," *SIAM Journal on Control and Optimization*, vol. 55, no. 5, pp. 3171–3203, Jan. 2017. DOI: 10.1137/140999463.

[15] C. Gstöttner, B. Kolar, and M. Schöberl, "Necessary and sufficient conditions for the linearisability of two-input systems by a two-dimensional endogenous dynamic feedback," *International Journal of Control*, pp. 1–22, Dec. 2021. DOI: 10.1080/00207179.2021.2015542. arXiv: 2106.14722.

[16] X. Zhou, X. Wen, Z. Wang, *et al.*, "Swarm of micro flying robots in the wild," *Science Robotics*, vol. 7, no. 66, eabm5954, 2022. DOI: 10.1126/scirobotics.abq2215.

[17] S. M. LaValle, *Planning algorithms.* Oulu: Cambridge university press, 2006.

[18] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE international conference on robotics and automation*, IEEE, 2011, pp. 2520–2525.

[19] B. Morrell, M. Rigter, G. Merewether, *et al.*, "Differential flatness transformations for aggressive quadrotor flight," in *2018 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2018, pp. 5204–5210.

[20] E. Tal and S. Karaman, "Accurate tracking of aggressive quadrotor trajectories using incremental nonlinear dynamic inversion and differential flatness," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 1203–1218, 2020. DOI: 10.1109/TCST.2020.3001117.

[21] 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, Dec. 2017. DOI: 10.1109/LRA.2017.2776353.

[22] S. Sun, A. Romero, P. Foehn, E. Kaufmann, and D. Scaramuzza, "A comparative study of nonlinear mpc and differential-flatness-based control for quadrotor agile flight," *IEEE Transactions on Robotics*, 2022. DOI: 10.1109/TRO.2022.3177279.

[23] Z. Jiang, M. A. Lawati, A. Mohammadhasani, and A. F. Lynch, "Quasistatic state feedback trajectory tracking controller: Px4 sitl validation," *Journal of Intelligent & Robotic Systems*, Apr. 2022.

[24] G. Yu, D. Cabecinhas, R. Cunha, and C. Silvestre, "Nonlinear backstepping control of a quadrotor-slung load system," *IEEE/ASME Transactions on Mechatronics*, vol. 24, no. 5, pp. 2304–2315, Oct. 2019. DOI: 10.1109/tmech.2019.2930211.

[25] S. Yang and B. Xian, "Exponential regulation control of a quadrotor unmanned aerial vehicle with a suspended payload," *IEEE Transactions on Control System Technology*, vol. 28, no. 6, pp. 2762–2769, Nov. 2020. DOI: 10.1109/tcst.2019.2952826.

[26] K. Klausen, T. I. Fossen, and T. A. Johansen, "Nonlinear control with swing damping of a multirotor UAV with suspended load," *Journal of Intelligent & Robotic Systems*, vol. 88, no. 2-4, pp. 379–394, Mar. 2017. DOI: 10.1007/s10846-017-0509-6.

[27] K. Sreenath, N. Michael, and V. Kumar, "Trajectory generation and control of a quadrotor with a cable-suspended load - a differentially-flat hybrid system," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, May 2013. DOI: 10.1109/icra.2013.6631275.

[28] G. Yu, D. Cabecinhas, R. Cunha, and C. Silvestre, "Nonlinear backstepping control of a quadrotor-slung load system," *IEEE/ASME Transactions on Mechatronics*, vol. 24, no. 5, pp. 2304–2315, 2019.

[29] G. Yu, W. Xie, D. Cabecinhas, R. Cunha, and C. Silvestre, "Adaptive control with unknown mass estimation for a quadrotor-slung-load system," *ISA Transactions*, vol. 133, pp. 412–423, Feb. 2023. DOI: 10.1016/j.isatra.2022.06.036.

[30] G. Yu, J. Reis, D. Cabecinhas, R. Cunha, and C. Silvestre, "Reduced-complexity active disturbance rejection controller for quadrotor-slung-load transportation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–12, 2023. DOI: 10.1109/tsmc.2023.3263881.

[31] D. Cabecinhas, R. Cunha, and C. Silvestre, "A trajectory tracking control law for a quadrotor with slung load," *Automatica*, vol. 106, pp. 384–389, 2019.

[32] G. Flores, A. M. de Oca, and A. Flores, "Robust nonlinear control for the fully actuated hexa-rotor: Theory and experiments," *IEEE Control Systems Letters*, vol. 7, pp. 277–282, 2023. DOI: 10.1109/lcsys.2022.3188517.

[33] N. P. Nguyen, H. Oh, and J. Moon, "Continuous nonsingular terminal sliding-mode control with integral-type sliding surface for disturbed systems: Application to attitude control for quadrotor UAVs under external disturbances," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 58, no. 6, pp. 5635–5660, Dec. 2022. DOI: 10.1109/taes.2022.3177580.

[34] M. Bangura and R. Mahony, "Thrust control for multirotor aerial vehicles," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 390–405, 2017. DOI: 10.1109/TRO.2016.2633562.

[35] Z.-Y. Lv, Y. Wu, and W. Rui, "Nonlinear motion control for a quadrotor transporting a cable-suspended payload," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 8192–8206, Aug. 2020. DOI: 10.1109/tvt.2020.2997733.

[36] Z.-Y. Lv, S. Li, Y. Wu, and Q.-G. Wang, "Adaptive control for a quadrotor transporting a cable-suspended payload with unknown mass in the presence of rotor downwash," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 9, pp. 8505–8518, Sep. 2021. DOI: 10.1109/tvt.2021.3096234.

[37] Z. Lv, Q. Zhao, S. Li, and Y. Wu, "Finite-time control design for a quadrotor transporting a slung load," *Control Engineering Practice*, vol. 122, p. 105 082, May 2022. DOI: 10.1016/j.conengprac.2022.105082.

[38] J. Zeng and K. Sreenath, "Geometric control of a quadrotor with a load suspended from an offset," in *2019 American Control Conference (ACC)*, 2019, pp. 3044–3050. DOI: 10.23919/ACC.2019.8814939.

[39] M. Guerrero, D. Mercado, R. Lozano, and C. García, "Swing-attenuation for a quadrotor transporting a cable suspended payload," *Instrument Society of America*, vol. 68, pp. 433–449, 2017. DOI: 10.1016/j.isatra.2017.01.027.

[40] Z. Jiang, M. Al Lawati, and A. F. Lynch, "Quasi-static state feedback output tracking for a slung load system with rotor drag compensation: Px4-sitl validation," *Journal of Intelligent & Robotic Systems*, 2023.

[41] M. Fumagalli and E. Simetti, "Robotic technologies for predictive maintenance of assets and infrastructure [from the guest editors]," *IEEE Robotics & Automation Magazine*, vol. 25, no. 4, pp. 9–10, 2018.

[42] M. Ryll, G. Muscio, F. Pierri, *et al.*, "6d interaction control with aerial robots: The flying end-effector paradigm," *The International Journal of Robotics Research*, vol. 38, no. 9, pp. 1045–1062, 2019. DOI: 10.1177/0278364919856694.

[43] M. Allenspach, K. Bodie, M. Brunner, *et al.*, "Design and optimal control of a tiltrotor micro-aerial vehicle for efficient omnidirectional flight," *The International Journal of Robotics Research*, vol. 39, no. 10-11, pp. 1305–1325, Aug. 2020. DOI: 10.1177/0278364920943654.

[44] M. Ryll, H. H. Bulthoff, 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, Mar. 2015. DOI: 10.1109/tcst.2014.2330999.

[45] M. Hamandi, F. Usai, Q. Sablé, N. Staub, M. Tognon, and A. Franchi, "Design of multirotor aerial vehicles: A taxonomy based on input allocation," *The International Journal of Robotics Research*, vol. 40, no. 8-9, pp. 1015–1044, Jul. 2021. DOI: 10.1177/02783649211025998.

[46] M.-D. Hua, T. Hamel, P. Morin, and C. Samson, "Introduction to feedback control of underactuated vtol vehicles: A review of basic control design ideas and principles," *IEEE Control systems magazine*, vol. 33, no. 1, pp. 61–75, 2013.

[47] K. P. Valavanis and G. J. Vachtsevanos, *Handbook of unmanned aerial vehicles*. Springer, 2015, vol. 2077.

[48] M. Suomalainen, Y. Karayiannidis, and V. Kyrki, “A survey of robot manipulation in contact,” *Robotics and Autonomous Systems*, vol. 156, p. 104 224, 2022, ISSN: 0921-8890. DOI: `https://doi.org/10.1016/j.robot.2022.104224`.

[49] L. Villani and J. D. Schutter, “Force control,” in M. Ang, O. Khatib, and B. Siciliano, Eds. Springer-Verlag, 2018, ch. 9, pp. 195–219.

[50] H.-N. Nguyen and D. Lee, “Hybrid force/motion control and internal dynamics of quadrotors for tool operation,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Nov. 2013. DOI: `10.1109/iros.2013.6696849`.

[51] H. W. Wopereis, J. J. Hoekstra, T. H. Post, G. A. Folkertsma, S. Stramigioli, and M. Fumagalli, “Application of substantial and sustained force to vertical surfaces using a quadrotor,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, May 2017. DOI: `10.1109/icra.2017.7989314`.

[52] T. Tomic, C. Ott, and S. Haddadin, “External wrench estimation, collision detection, and reflex reaction for flying robots,” *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1467–1482, Dec. 2017. DOI: `10.1109/tro.2017.2750703`.

[53] M. Ryll, G. Muscio, F. Pierri, *et al.*, “6d physical interaction with a fully actuated aerial robot,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, May 2017. DOI: `10.1109/icra.2017.7989608`.

[54] G. Nava, Q. Sable, M. Tognon, D. Pucci, and A. Franchi, “Direct force feedback control and online multi-task optimization for aerial manipulators,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 331–338, Apr. 2020. DOI: `10.1109/lra.2019.2958473`.

[55] D. Tzoumanikas, F. Graule, Q. Yan, D. Shah, M. Popovic, and S. Leutenegger, “Aerial manipulation using hybrid force and position nmpc applied to aerial writing,” in *Robotics: Science and Systems XVI*, ser. RSS2020, Robotics: Science and Systems Foundation, Jun. 2020. DOI: `10.15607/RSS.2020.XVI.046`.

[56] K. Bodie, M. Brunner, M. Pantic, *et al.*, “Active interaction force control for contact-based inspection with a fully actuated aerial vehicle,” *IEEE Transactions on Robotics*, vol. 37, no. 3, pp. 709–722, Jun. 2021. DOI: `10.1109/tro.2020.3036623`.

[57] L. Peric, M. Brunner, K. Bodie, M. Tognon, and R. Siegwart, “Direct force and pose NMPC with multiple interaction modes for aerial push-and-slide operations,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, May 2021. DOI: `10.1109/icra48506.2021.9561990`.

[58] A. Gazar, G. Nava, F. J. A. Chavez, and D. Pucci, “Jerk control of floating base systems with contact-stable parameterized force feedback,” *IEEE Transactions on Robotics*, vol. 37, no. 1, pp. 1–15, Feb. 2021. DOI: `10.1109/tro.2020.3005547`.

[59] R. Watson, M. Kamel, D. Zhang, *et al.*, "Dry coupled ultrasonic non-destructive evaluation using an over-actuated unmanned aerial vehicle," *IEEE Transactions on Automation Science and Engineering*, pp. 1–16, 2021. DOI: `10.1109/tase.2021.3094966`.

[60] M. Allenspach, N. Lawrance, M. Tognon, and R. Siegwart, *Towards 6dof bilateral teleoperation of an omnidirectional aerial vehicle for aerial physical interaction*, 2022. DOI: `10.48550/ARXIV.2203.03177`.

[61] M. Brunner, L. Giacomini, R. Siegwart, and M. Tognon, *Energy tank-based policies for robust aerial physical interaction with moving objects*, 2022. DOI: `10.48550/ARXIV.2202.06755`.

[62] M. Allenspach, Y. Vyas, M. Rubio, R. Siegwart, and M. Tognon, "Human-state-aware controller for a tethered aerial robot guiding a human by physical interaction," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2827–2834, Apr. 2022. DOI: `10.1109/lra.2022.3143574`.

[63] M. Brunner, G. Rizzi, M. Studiger, R. Siegwart, and M. Tognon, "A planning-and-control framework for aerial manipulation of articulated objects," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 689–10 696, Oct. 2022. DOI: `10.1109/lra.2022.3191178`.

[64] S. Bellens, J. De Schutter, and H. Bruyninckx, "A hybrid pose / wrench control framework for quadrotor helicopters," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2012, pp. 2269–2274. DOI: `10.1109/ICRA.2012.6224682`.

[65] S. Park, J. Lee, J. Ahn, *et al.*, "Odar: Aerial manipulation platform enabling omnidirectional wrench generation," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 4, pp. 1907–1918, 2018. DOI: `10.1109/TMECH.2018.2848255`.

[66] J. Pliego-Jiménez and M. A. Arteaga-Pérez, "Adaptive position/force control for robot manipulators in contact with a rigid surface with uncertain parameters," *European Journal of Control*, vol. 22, pp. 1–12, 2015, ISSN: 0947-3580. DOI: `https://doi.org/10.1016/j.ejcon.2015.01.003`.

[67] S. Haddadin, A. De Luca, and A. Albu-Schäffer, "Robot collisions: A survey on detection, isolation, and identification," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1292–1312, 2017. DOI: `10.1109/TRO.2017.2723903`.

[68] A. Albers, S. Trautmann, T. Howard, T. A. Nguyen, M. Frietsch, and C. Sauter, "Semi-autonomous flying robot for physical interaction with environment," in *IEEE Conference on Robotics, Automation and Mechatronics*, 2010, pp. 441–446. DOI: `10.1109/RAMECH.2010.5513152`.

[69] D. Brescianini and R. D'Andrea, "Design, modeling and control of an omni-directional aerial vehicle," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2016, pp. 3261–3266. DOI: `10.1109/ICRA.2016.7487497`.

[70] 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 System Technology*, vol. 23, no. 2, pp. 540–556, 2015. DOI: `10.1109/TCST.2014.2330999`.

[71] 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, Apr. 2018. DOI: 10.1109/tro.2017.2786734.

[72] H. P. Whitaker, J. Yamron, and A. Kezer, *Design of model-reference adaptive control systems for aircraft*. Massachusetts Institute of Technology, Instrumentation Laboratory, 1958.

[73] Z. T. Dydek, A. M. Annaswamy, and E. Lavretsky, “Adaptive control and the nasa x-15-3 flight revisited,” *IEEE Control Systems*, vol. 30, no. 3, pp. 32–48, Jun. 2010, ISSN: 1941-000X. DOI: 10.1109/mcs.2010.936292.

[74] J.-J. E. Slotine and W. Li, “On the adaptive control of robot manipulators,” *The International Journal of Robotics Research*, vol. 6, no. 3, pp. 49–59, Sep. 1987. DOI: 10.1177/027836498700600303.

[75] C. Cheah, Y. Zhao, and J. Slotine, “Adaptive jacobian motion and force tracking control for constrained robots with uncertainties,” in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, IEEE, 2006. DOI: 10.1109/robot.2006.1642034.

[76] A. Astolfi and R. Ortega, “Immersion and invariance: A new tool for stabilization and adaptive control of nonlinear systems,” *IEEE Transactions on Automatic Control*, vol. 48, no. 4, pp. 590–606, Apr. 2003, ISSN: 0018-9286. DOI: 10.1109/tac.2003.809820.

[77] D. Karagiannis and A. Astolfi, “Nonlinear adaptive control of systems in feedback form: An alternative to adaptive backstepping,” *Systems & Control Letters*, vol. 57, no. 9, pp. 733–739, Sep. 2008, ISSN: 0167-6911. DOI: 10.1016/j.sysconle.2008.02.006.

[78] D. Seo and M. R. Akella, “Non-certainty equivalent adaptive control for robot manipulator systems,” *Systems & Control Letters*, vol. 58, no. 4, pp. 304–308, Apr. 2009, ISSN: 0167-6911. DOI: 10.1016/j.sysconle.2008.11.008.

[79] A. Astolfi, D. Karagiannis, and R. Ortega, *Nonlinear and adaptive control with applications*. Springer, 2008, vol. 187.

[80] J. Hu and H. Zhang, “Immersion and invariance based command-filtered adaptive backstepping control of vtol vehicles,” *Automatica*, vol. 49, no. 7, pp. 2160–2167, 2013.

[81] K. Fujimoto, M. Yokoyama, and Y. Tanabe, “I & i -based adaptive control of a four-rotor mini helicopter,” in *IECON 2010-36th Annual Conference on IEEE Industrial Electronics Society*, IEEE, 2010, pp. 144–149.

[82] B. Zhao, B. Xian, Y. Zhang, and X. Zhang, “Nonlinear robust adaptive tracking control of a quadrotor uav via immersion and invariance methodology,” *IEEE Transactions on Industrial Electronics*, vol. 62, no. 5, pp. 2891–2902, May 2015, ISSN: 1557-9948. DOI: 10.1109/tie.2014.2364982.

[83] A. Astolfi, R. Ortega, and A. Venkatraman, “A globally exponentially convergent immersion and invariance speed observer for mechanical systems with non-holonomic constraints,” *Automatica*, vol. 46, no. 1, pp. 182–189, 2010.

[84] D. Thakur, S. Srikant, and M. R. Akella, "Adaptive attitude-tracking control of spacecraft with uncertain time-varying inertia parameters," *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 1, pp. 41–52, Jan. 2015, ISSN: 1533-3884. DOI: `10.2514/1.g000457`.

[85] L. Wang, B. Yi, and H. Su, "An i & i adaptive redesign approach for asymptotic stability without pe," *IFAC-PapersOnLine*, vol. 56, no. 1, pp. 264–269, 2023, ISSN: 2405-8963. DOI: `10.1016/j.ifacol.2023.02.045`.

[86] B. Yi, R. Ortega, I. R. Manchester, and H. Siguerdidjane, "Path following of a class of underactuated mechanical systems via immersion and invariance-based orbital stabilization," *International Journal of Robust and Nonlinear Control*, vol. 30, no. 18, pp. 8521–8544, Oct. 2020, ISSN: 1099-1239. DOI: `10.1002/rnc.5258`.

[87] D. K. Villa, A. S. Brandao, and M. Sarcinelli-Filho, "A survey on load transportation using multirotor uavs," *Journal of Intelligent & Robotic Systems*, pp. 1–30, 2019.

[88] M. Bangura, "Aerodynamics and control of quadrotors," Ph.D. dissertation, College of Engineering and Computer Science, The Australian National University, 2017.

[89] N. Cao and A. F. Lynch, "Inner-outer loop control with constraints for rotary-wing UAVs," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2015, pp. 294–302.

[90] N. Cao and A. Lynch, "Time-delay robustness analysis of a nested saturation control for uav motion control," *Control Strategy for Time-Delay Systems: Part II: Engineering Applications*, p. 69, 2020.

[91] N. Cao and A. Lynch, "Predictor-based control design for uavs: Robust stability analysis and experimental results," *International Journal of Control*, vol. 94, no. 6, pp. 1529–1543, 2021.

[92] W. Dong, G.-Y. Gu, X. Zhu, and H. Ding, "High-performance trajectory tracking control of a quadrotor with disturbance observer," *Sensors and Actuators A: Physical*, vol. 211, pp. 67–77, 2014.

[93] S. Tang and V. Kumar, "Autonomous flight," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 29–52, 2018.

[94] D. Silver, J. Schrittwieser, K. Simonyan, *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017. DOI: `10.1038/nature24270`.

[95] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, Jul. 20, 2017. arXiv: `1707.06347v2 [cs.LG]`.

[96] OpenAI, I. Akkaya, M. Andrychowicz, *et al.*, "Solving rubikś cube with a robot hand," Oct. 16, 2019. arXiv: `1910.07113 [cs.LG]`.

[97] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, Aug. 2013. DOI: `10.1177/0278364913495721`.

[98] T. P. Lillicrap, J. J. Hunt, A. Pritzel, *et al.*, "Continuous control with deep reinforcement learning," *arXiv:1509.02971 [cs, stat]*, Jul. 2015, arXiv: 1509.02971. [Online]. Available: `http://arxiv.org/abs/1509.02971`.

[99] 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, Jun. 2010. DOI: `10.1177/0278364910371999`.

[100] N. O. Lambert, D. S. Drew, J. Yaconelli, S. Levine, R. Calandra, and K. S. Pister, "Low-level control of a quadrotor with deep model-based reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4224–4230, 2019.

[101] B. Helder, E.-J. Van Kampen, and M. Pavel, "Online adaptive helicopter control using incremental dual heuristic programming," in *AIAA Scitech 2021 Forum*, 2021, p. 1118.

[102] Y. Zhou, E.-J. van Kampen, and Q. P. Chu, "Incremental model based online dual heuristic programming for nonlinear adaptive control," *Control Engineering Practice*, vol. 73, pp. 13–25, 2018.

[103] E. Kaufmann, A. Loquercio, R. Ranftl, M. Muller, V. Koltun, and D. Scaramuzza, "Deep drone acrobatics," *arXiv preprint arXiv:2006.05768*, 2020.

[104] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[105] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic Policy Gradient Algorithms," en, in *International Conference on Machine Learning*, Jan. 2014, ch. Machine Learning, pp. 387–395. [Online]. Available: `http://proceedings.mlr.press/v32/silver14.html`.

[106] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proceedings of the International Conference on Learning Representations (ICLR)*, Jun. 8, 2015. arXiv: `1506.02438v6 [cs.LG]`.

[107] R. Polvara, M. Patacchiola, S. Sharma, *et al.*, "Toward end-to-end control for uav autonomous landing via deep reinforcement learning," in *2018 International conference on unmanned aircraft systems (ICUAS)*, IEEE, 2018, pp. 115–123.

[108] E. Bohn, E. M. Coates, S. Moe, and T. A. Johansen, "Deep reinforcement learning attitude control of fixed-wing uavs using proximal policy optimization," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2019, pp. 523–533.

[109] Y. Zhang, Y. Zhang, and Z. Yu, "Path following control for uav using deep reinforcement learning approach," *Guidance, Navigation and Control*, vol. 1, no. 01, p. 2 150 005, 2021.

[110] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2096–2103, Oct. 2017. DOI: `10.1109/lra.2017.2720851`.

[111] J. Rudolph and E. Delaleau, "Some remarks on quasi-static feedback of generalized states," in *IFAC Proceedings Volumes*, vol. 27, 1994, pp. 51–56. DOI: `https://doi.org/10.1016/S1474-6670(17)47621-X`.

[112] E. Delaleau and J. Rudolph, "Some examples and remarks on quasi-static feedback of generalized states," *Automatica*, vol. 34, no. 8, pp. 993–999, Aug. 1998. DOI: `10.1016/s0005-1098(98)00047-8`.

[113] O. Fritsch, P. D. Monte, M. Buhl, and B. Lohmann, "Quasi-static feedback linearization for the translational dynamics of a quadrotor helicopter," in *Proceedings of the American Control Conference*, Montreal, QC, Canada, Jun. 2012. DOI: `10.1109/acc.2012.6314682`.

[114] Z. Jiang, M. Al Lawati, A. Mohammadhasani, and A. F. Lynch, "Flatness-based motion control of a uav slung load system using quasi-static feedback linearization," in *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2022, pp. 361–368.

[115] J.-M. Kai, G. Allibert, M.-D. Hua, and T. Hamel, "Nonlinear feedback control of quadrotors exploiting first-order drag effects," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 8189–8195, 2017.

[116] Z. Jiang and A. F. Lynch, "Quasi-static state feedback output tracking for a slung load system with rotor drag compensation: Px4 sitl validation," in *2023 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, IEEE, 2023, pp. 241–246.

[117] L. Meier, D. Agar, B. Küng, *et al.*, *Px4/px4-autopilot: Stable release v1.13.0*, version v1.13.0, Jun. 2022. DOI: `10.5281/zenodo.6682275`.

[118] *Holybro Vision Dev Kit V1.5*, `https://holybro.com/products/px4-vision-dev-kit-v1-5`, [Online; accessed 29-Sep-2023], 2023.

[119] J. G. Romero and H. Rodríguez-Cortés, "Asymptotic stability for a transformed nonlinear UAV model with a suspended load via energy shaping," *European Journal of Control*, vol. 52, pp. 87–96, Mar. 2020. DOI: `10.1016/j.ejcon.2019.09.002`.

[120] P. Foehn, E. Kaufmann, A. Romero, *et al.*, "Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight," *Science Robotics*, vol. 7, no. 67, eabl6259, 2022. DOI: `10.1126/scirobotics.abl6259`. [Online]. Available: `https://www.science.org/doi/abs/10.1126/scirobotics.abl6259`.

[121] M. A. Rafique, "Adaptive nonlinear control for unmanned aerial vehicles: Visual servoing and aerial manipulation," 2022.

[122] A. Moeini, "Disturbance observer-based motion control and visual-inertial-actuator odometry for uavs," 2021.

[123] 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.

[124] K. M. Lynch and F. C. Park, *Modern robotics.* Evanston: Cambridge University Press, 2017.

[125] *Holybro PX4 vision kit wiring diagram*, https://docs.holybro.com/drone-development-kit/px4-vision-dev-kit-v1.5/wiring-diagram, Accessed: 2024-01-10.

[126] L. Meier, *Mavlink: Micro air vehicle communication protocol*, https://mavlink.io/en/ [accessed 01 Dec 2023], 2023. [Online]. Available: https://mavlink.io/en/.

[127] S. Skaug, *Blheli for brushless esc firmware.* [Online]. Available: https://github.com/bitdump/BLHeli.

[128] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, vol. 3, pp. 2149–2154. DOI: 10.1109/IROS.2004.1389727.

[129] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "Robot operating system (ros): The complete reference (volume 1)," in A. Koubaa, Ed. Cham: Springer International Publishing, 2016, ch. RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625, ISBN: 978-3-319-26054-9. DOI: 10.1007/978-3-319-26054-9_23.

[130] R. Smith *et al.*, *Open dynamics engine*, 2007.

[131] P. Martin and E. Salaün, "The true role of accelerometer feedback in quadrotor control," in *2010 IEEE international conference on robotics and automation*, IEEE, 2010, pp. 1623–1629.

[132] R. Gill and R. D'Andrea, "Computationally efficient force and moment models for propellers in uav forward flight applications," *Drones*, vol. 3, no. 4, p. 77, 2019.

[133] *Dynamics of a Rotor-Pendulum With a Small, Stiff Propeller in Wind*, vol. 1, Dynamic Systems and Control Conference, Oct. 2016. DOI: 10.1115/DSCC2016-9774.

[134] N. Cao and A. F. Lynch, "Inner-outer loop control for quadrotor uavs with input and state constraints," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 5, pp. 1797–1804, Apr. 2016. DOI: 10.1109/TCST.2015.2505642.

[135] R. Marino and P. Tomei, *Nonlinear Control Design: Geometric, Adaptive, and Robust.* Hertfordshire, United Kingdom: Prentice Hall, 1995.

[136] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se(3)," in *49th IEEE Conference on Decision and Control (CDC)*, IEEE, Dec. 2010. DOI: 10.1109/cdc.2010.5717652.

[137] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 2018.

[138] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann, *Stable baselines3*, https://github.com/DLR-RM/stable-baselines3, 2019.

[139] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning.* MIT press Cambridge, 2016, vol. 1.

[140] J. Achiam, "Spinning Up in Deep Reinforcement Learning," 2018.

[141] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, PMLR, 2018, pp. 1861–1870.

[142] N. Heess, D. TB, S. Sriram, *et al.*, "Emergence of locomotion behaviours in rich environments," *arXiv preprint arXiv:1707.02286*, 2017.

[143] V. R. Konda and J. N. Tsitsiklis, "On actor-critic algorithms," *SIAM journal on Control and Optimization*, vol. 42, no. 4, pp. 1143–1166, 2003.

[144] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, PMLR, 2015, pp. 1889–1897.