Summer 8-2017

Cyber-Physical System Characterization and Co-Regulation of a Quadrotor UAS

Seth E. Doebbeling
University of Nebraska - Lincoln, sdoebbeling2@unl.edu

CYBER-PHYSICAL SYSTEM CHARACTERIZATION AND CO-REGULATION OF A QUADROTOR UAS

by

Seth Doebbeling

A THESIS

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfilment of Requirements

For the Degree of Master of Science

Major: Mechanical Engineering and Applied Mechanics

Under the Supervision of Professor Justin Bradley

Lincoln, Nebraska

August, 2017

CYBER-PHYSICAL SYSTEM CHARACTERIZATION AND CO-REGULATION OF A QUADROTOR UAS

Seth Doebbeling, M.S.

University of Nebraska, 2017

Adviser: Justin Bradley

An Unmanned Aircraft System (UAS) is a Cyber-Physical System (CPS) in which a host of real-time computational tasks contending for shared resources must be cooperatively managed to obtain mission objectives. Traditionally, control of the UAS is designed assuming a fixed, high sampling rate in order to maintain reliable performance and margins of stability. But emerging methods challenge this design by dynamically allocating resources to computational tasks, thereby affecting control and mission performance. To apply these emerging strategies, a characterization and understanding of the effects of timing on control and trajectory following performance is required. Going beyond traditional control evaluation techniques, in this work we characterize the trajectory following performance, timing, and control of a quadrotor UAS under Discrete Linear Quadratic Regulator control (DLQR) designed at various sampling rates. We introduce new metrics for characterizing cyber-physical quadrotor performance, and provide empirical evidence that high-sampling-rate control strategies over 50 Hz may not significantly improve control performance for our quadrotor platform and hence may not effectively allocate resources that could be used to improve other (non-control related) mission objectives. We then propose a strategy in which a model representing the sampling rate is augmented to the state-space model of a quadrotor UAS, controllers are designed for this holistic system to more effectively allocate these resources. We develop a full nonlinear equation co-regulation simulation suite in MATLAB and provide analysis of the UAS in following a trajectory under traditional control

design as well as our proposed co-regulation design for comparison. Under co-regulation we are able to reduce the maximum power consumption by ~12% and the time averaged normalized state error by ~75% for a unit step in x, y, and z, while maintaining relatively good cross-tracking performance. Results illustrate the need for a higher level trajectory planning and generation technique capable of translating mission tasks into smooth trajectories and providing both physical and cyber reference commands suitable for our variable rate co-regulation architecture. Therefore, we design a low level, kinematic trajectory generator capable of easily adjusting timing constraints which provides a first step toward such a motion tracking architecture.

DEDICATION

To my parents. Thank you, you are my inspiration and my foundation. I would not be who I am today without your guidance.

ACKNOWLEDGMENTS

This work was supported in part by NSF award #1638099.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The design and development of robotic control systems, with advances in autonomy and associated increasing demands on computation, can no longer rely on a "black box" view of the computational system to adequately meet the requirements of next generation smart autonomous vehicles. Unmanned Aircraft Systems (UAS), where computational resources are scarce due to size and weight restrictions, are one example of such a system. The common assumptions of plentiful resources, instantaneous computation, and model emulation accuracy become decreasingly valid as systems grow in complexity and try to meet increasing demands. Under these scenarios, the consideration of computational and physical demands in control design can lead to improved overall system and mission performance by dynamically allocating computational and physical resources according to system performance.

As a result, control strategies that consider physical and computational limitations and performance are gaining popularity [1]. The design, consideration, and intersection of computational resources with communication and physical resources is the objective of Cyber-Physical Systems (CPS) research [2]. We make the assumption that "cyber" and "physical" components are defined as in [3] where "cyber" refers to the intangible, computational, algorithmic, and software components of the system and "physical" refers to components consisting of mass and/or capable of generating a physical force.

Control strategies like event-triggered control [4], optimal sampling instances [5], and time-varying control [6] all consider the regulation of computational resources alongside physical actuation and control. This gives a system the ability to dynamically use resources for control as needed. The potential benefits of such a control strategy to a robotic vehicle include: more timely and targeted resource allocation, allowing the vehicle to accomplish more with the same resources; more computation available for accomplishing mission objectives like data collection; and reduced control effort. While hybrid strategies like event-triggered, optimal sampling, and time-varying control adjust computational resources reactively, they do not consider how to plan for this dynamic allocation to meet mission performance objectives.

In this work we focus on the development and application of a CPS co-regulation mechanism [3, 7] that simultaneously, and in response to system performance, adjusts control inputs alongside the sampling period of the real-time computing task executing the control software to realize the benefits described for a quadrotor UAS system.

Consider a quadrotor UAS hosting an on-board board camera and global positioning system (GPS) and carrying a small, light-weight survival kit on a search and rescue mission. The UAS is tasked with surveying an area of terrain too difficult for human rescuers to navigate using the on-board camera and locate distressed hikers. Once the hikers are located, the UAS must drop the survival kit, indicate to a ground station that the distressed hikers have been found, and transmit their GPS location as well as images of their condition. In addition, the UAS is tasked with performing periodic flight maneuvers as gestures to indicate to the hikers which direction to seek help, stay put, or otherwise. Given a control design in which cyber and physical resources are fixed, the UAS may easily be limited in its performance while executing one or more of its mission tasks. Consider Figure 1.1,

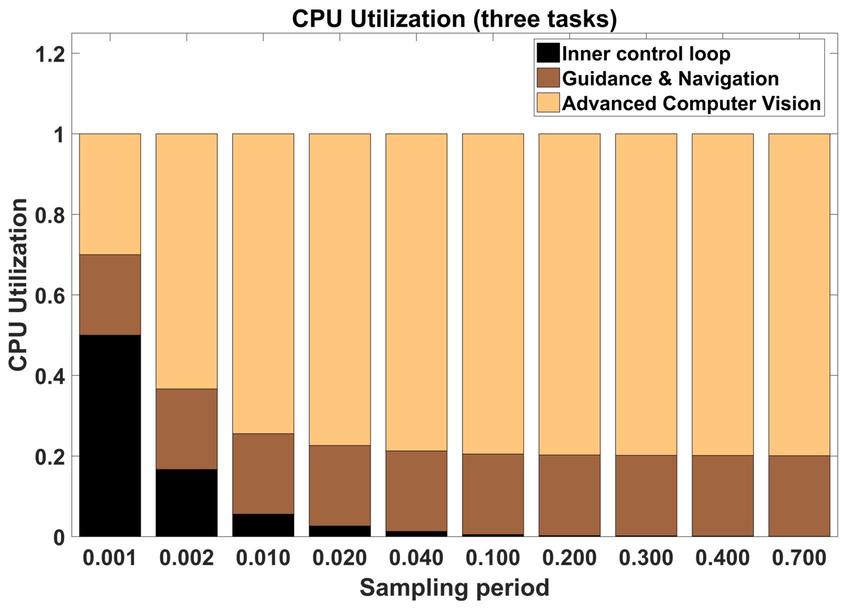


Figure 1.1: CPU utilization and availability for different tasks, assuming that the attitude controller has a worst case execution time of 0.5 ms.

which represents resource allocation on a real-time system for an inner flight control loop, guidance and navigation, and advanced computer vision. In a fixed-rate design, the design engineer must choose a single sampling period for the system to operate at. If the sampling rate (inverse of sampling period) is high, the UAS may be better able to efficiently cover the search area as well as perform flight gestures once the hikers are located; however, there are less resources for image processing and transmission to indicate the state of the hikers and the severity of the terrain surrounding them to nearby rescuers. Design at lower rates may provide for better evaluation of the situation for rescuers, but the UAS may face limitations in surveying the area effectively with the given payload and may not be able to accurately converge on the hikers position or gesture information to them. Under a variable rate control such as co-regulation, the UAS may be able to transition between the entire range of sampling periods in Figure 1.1 under a single control design. The UAS may then dynamically allocate resources to each of the real-time system task to most effectively execute the mission, using lower control rates when vision processing is prioritized and higher rates when aggressive flight maneuvers are required.

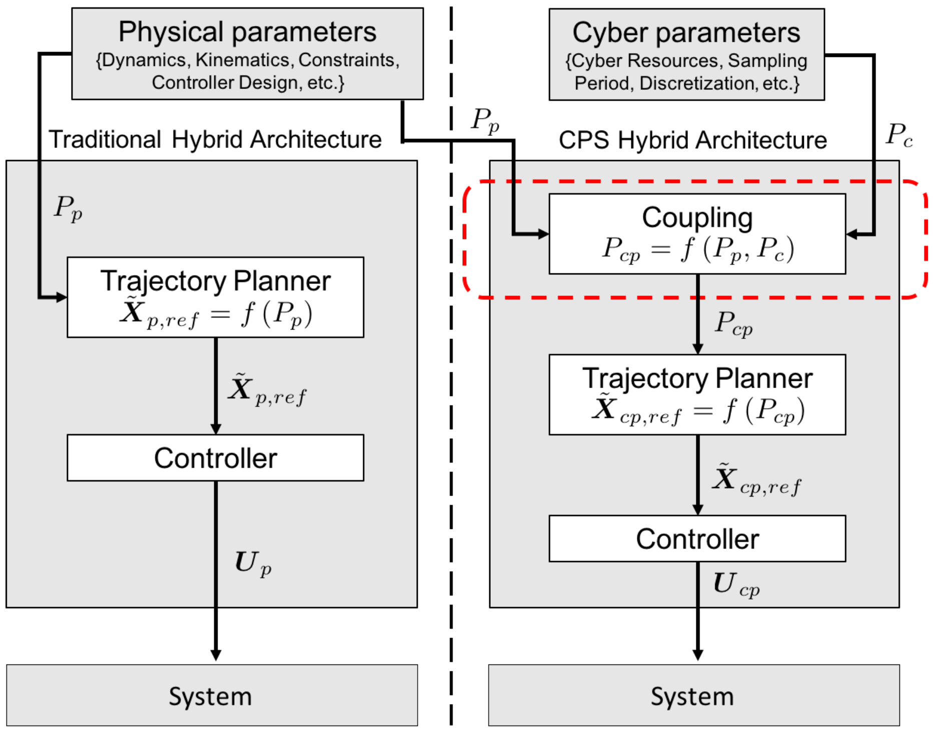

Physical parameters
{Dynamics, Kinematics, Constraints, Controller Design, etc.}
Cyber parameters
{Cyber Resources, Sampling Period, Discretization, etc.}
P_p
P_c
Traditional Hybrid Architecture
CPS Hybrid Architecture
P_p
Coupling
$P_{cp} = f(P_p, P_c)$
Trajectory Planner
$\tilde{\boldsymbol{X}}_{p,ref} = f(P_p)$
P_{cp}
Trajectory Planner
$\tilde{\boldsymbol{X}}_{cp,ref} = f(P_{cp})$
$\tilde{\boldsymbol{X}}_{p,ref}$
$\tilde{\boldsymbol{X}}_{cp,ref}$
Controller
Controller
$\boldsymbol{U}_p$
$\boldsymbol{U}_{cp}$
System
System

Figure 1.2: Traditional Hybrid Architecture (left) and CPS Hybrid Architecture (right). Subscript "p, c" represent "physical" and "cyber" respectively.

In Figure 1.2 we show a traditional hybrid architecture alongside a potential CPS hybrid architecture. In the traditional hybrid robotic architecture (left in Figure 1.2), a reactive controller provides an input, U_p, to the system to command small movements of the controlled object through space and time [8]. To command larger movements, a guidance and deliberative planning layer uses the explicit mathematical relationships between actuators and movement (e.g. kinematics, dynamics, constraints, etc.), P_p, to discretize the desired larger movement into accomplishable reference commands, $\tilde{X}_{p,ref}$, for the reactive controller [9]. Similarly, to develop a CPS hybrid architecture (right in Figure 1.2), where cyber effectors (i.e. sampling rate) are controlled alongside physical ones, a cyber-physical control input, U_{cp}, is given to the combined cyber-physical system to move the physical object through space and time simultaneously with adjustments to the sampling rate. Commanding larger adjustments is accomplished by issuing cyber-physical reference commands, $\tilde{X}_{cp,ref}$, from a trajectory planner that requires a corresponding relationship between both cyber and physical effectors and the movement through physical and cyber space and time, P_{cp} (red dashed line around "Coupling" block in Figure 1.2). This enables a cyber-physical guidance and planning layer to optimize physical and cyber trajectories where physical and cyber performance is coupled. In this work we focus on characterization of the cyber-physical system, followed by design of a hybrid control scheme, and an important first step toward a CPS trajectory planner as we work toward development of such a CPS hybrid architecture as described in Figure 1.2.

Toward the goal of directly coupling cyber and physical resources to mission performance for a quadrotor UAS, we develop a CPS co-regulation mechanism through three stages of characterization, design, and development.

We first characterize the relationship between control task period (or sampling period), control gain, and reference trajectory following performance of a quadrotor UAS. A careful characterization of the imposed sampling rate of the controller influences stability margins [10], and schedulability [11]. Although the relationship between sampling rate and control performance is understood [12], we take the important next step of characterizing the relationship between sampling rate and trajectory following performance - the relationship required to more optimally trade off cyber and physical resources at the planning layer. We also introduce new metrics that explicitly measure trajectory following performance of a cyber-physical vehicle system, going beyond traditional controller performance metrics.

We then propose a hybrid control design called "co-regulation" in which both computational and physical resources are allocated in real-time under a unifying, closed loop control architecture. We evaluate the controller under metrics that build off of those defined in characterization of trajectory following performance as well as previous work to compare our variable rate co-regulation control design to standard fixed rate controllers.

Finally we design and analyze a low level trajectory generator employing kinematic equations and physical constraints on velocity, acceleration, and time suitable for the designed co-regulation controller. The trajectory generator is a first step toward a higher level motion tracking architecture which optimally commands resource allocation at a mission planning level.

1.1 Contributions

1.1.1 Co-regulation of a Quadrotor UAS

In this work we propose a new Riemannian/Lebesgue hybrid method for a quadrotor UAS we call "co-regulation" wherein computational and physical resources are allocated and controlled according to feedback from the computational and physical system.

Similar formulations have been pursued in [7, 3] for different systems and scenarios. Here we build on that work in 3 distinct ways. First, we apply our strategy to a quadrotor UAS representing the first discrete-time-varying control strategy for a quadrotor UAS. Next, we evaluate our system by assessing performance of the UAS in following a trajectory in addition to a common step-response as in most controller assessments. This gives a much better indication of how the UAS will behave in a real mission and gives us a strategy for developing a high-level planning algorithm that leverages our co-regulation strategy. Finally, we demonstrate that common strategies of over-design using time redundancy in sampling to improve safety margins [13] offer very little improvement in performance and, in fact, almost no improvement is seen for our quadrotor in sampling over 50Hz. Since some commercial quadrotors sample at 1000Hz, this implies that ~95% of control resources may be poorly utilized and could possibly be put to use in accomplishing other mission objectives.

1.1.2 CPS Trajectory Generation

Toward optimal allocation of cyber and physical resources for a quadrotor UAS executing mission tasks, kinematic trajectory generation technique is adapted to three-dimensional Cartesian coordinates and used to generate smooth trajectories under physical constraints on velocity, acceleration, and time. The generator designed forms a series of discrete trajectory points with associated information for estimations of state variables at given time instances. The period at which these points are generated can be made to correspond to the sampling period of control tasks under a higher level motion planning architecture with the aim of effectively allocating cyber and physical resources at a lower level. This trajectory generator may be a useful tool for increasing precision in trajectory following control for a UAS as well as cyber-physical efficiency.

1.2 Innovations

The application of a cyber-physical co-regulation control architecture to a quadrotor UAS model is a novel contribution. That is, for a quadrotor UAS, this is the first application of a real-time, varying sampling rate control scheme. Coupled with this design, evaluation of the control scheme in regard to trajectory following through several newly defined metrics is an extension of traditional controller performance analysis. We also make an important first step toward the development of a cyber-physical motion planning architecture through the design of a trajectory generator suitable for such a system. When coupled with estimates of cyber and physical effectors the trajectory generator may be used to adjust trajectory points to more efficiently manage cyber and physical resources.

Chapter 2

Related Work

The area of Cyber-Physical Systems research arose out of the area known as “cybernetics,” an ancestor of research areas such as control, real-time systems, optimization, autonomy, artificial intelligence, and others [14].

One specific CPS research aim is to investigate new methods, models, and integrations that bridge the divide between discrete computation and continuous control and movement through space [15]. We briefly discuss the role of computation in control and present related work that considers computation and control simultaneously, including within UAS. This is followed by review of several motion planning schemes related to autonomous robotics as we aim to provide insight into computational and physical trade-offs in their design.

2.1 Computer Control

Generally, computer control strategies can be divided up into Riemannian, Lebesgue, and Hybrid sampling techniques [16]. Riemannian sampling represents traditional constant sample-and-hold periods seen in digital/computational control techniques [17, 18]. This approach is nearly universal due to the strong foundation of digital control, and because periodic sampling matches traditional real-time task scheduling algorithms that primarily consider time-triggered periodic task sets [11].

The implementation of the chosen sampling rate is typically represented by assigning a periodicity value to the real-time computational control task. Ideally, either an offline real-time task schedule is then designed, or an online scheduling algorithm is selected [11] and implemented on a Real-Time Operating System (RTOS) to guarantee timing deadlines will be met. Typically, however, because of the complexity of implementation on an RTOS and since certifications and performance guarantees may not be strictly regulated for applications of certain CPS, a best-effort round-robin architecture (or similar) on a Linux distribution or other micro-controller may be relied upon to provide timing with significant variability. In this scenario, where a processor hosts a multitude of tasks and the execution of one or more tasks may become aperiodic, time redundancy is employed to try and mitigate the consequences of missed deadlines [13]. That is, to ensure tasks meet execution deadlines, redundancies across multiple processors may be employed, consuming computational resources.

Under Lebesgue sampling, in contrast, control inputs are obtained as needed in irregular intervals and result in the flexible and dynamic allocation of both control and computational resources that could lead to improved efficiency and performance for a CPS. One strategy for Lebesgue sampling, event-triggered control [19, 20], issues control inputs based on limits of output deviation from nominal conditions, also called "control by exception" [21]. Historically, event-triggered control has been applied to relatively simple systems, but recently has been applied to a quadrotor UAS, though with limitations in applicability [22]. The primary difficulty with this strategy is the lack of a mature theoretical framework for designing and analyzing these control strategies [16].

Another Lebesgue sampling strategy uses optimal control techniques to solve for the optimal control input and sampling instant simultaneously. In [5] optimality is shown for a narrow class of systems and a quantization strategy is used to reduce computational complexity. A similar approach in a receding horizon framework is proposed in [6]. These strategies provide the dynamic resource allocation desired, but depend on precise execution and timing of the computational system. Also, since no feedback is used, they may be susceptible to noise, disturbances, and environmental changes.

Finally, hybrid strategies combine Riemannian and Lebesgue sampling approaches to reap the benefits of predictability of Riemannian sampling with the resource savings and efficiency of Lebesgue sampling. In [23] an adaptive hybrid switching strategy switches between time-triggered and event-triggered controllers, and the controller is shown to be stable for a class of disturbances. The implications of this cooperation between software execution, real-time progression, and control performance are highly consequential. For system performance and margins of stability, the on-board control software must meet deadlines, usually over-designed for worst-case contingency management. For cyber performance, devoting fewer resources to control implies available resources for other computing activities. As a result, on a constrained system the design is a resource allocation and performance trade off.

Several related areas have investigated and leveraged this tradeoff. In [12] an exploration of the impact of sampling rate and control gain on step response is given for a system of inverted pendulums. Very high sampling rates typically result in better performance as the discrete controller approaches its continuous counterpart, though with some caveats [24]. Quality of Service (QoS) research investigates tradeoffs between cyber resource allocation and system performance (including controllers) for various discrete "service" intervals in the cyber system. This research confirms the trend that allocating more cyber resources generally results in better performance [25, 26]. Networked Control Systems has traditionally sought to identify conditions under which stability and performance can be guaranteed for a system wherein sensing, control, and actuation occur on networked computers [27, 28]. More recently, in [29], an optimization strategy is used to identify communication and control inputs simultaneously while taking into account packet loss. Event-triggered control research seeks to maximize cyber resource allocation [30, 31] while maintaining control performance guarantees. While successful in some instances, a fully-developed theory similar to digital control has still eluded the community [16]. A few mechanisms utilizing optimal control techniques to generate control trajectories and sampling instants form a time-varying sampling rate controller in [6, 5], and have produced successful theoretical results, although with increased computational complexity.

2.1.1 Sensor Scheduling

In feedback control an estimator is often used to make estimates of state variables based off of measurements made by one or more sensors. Sensors may be required to measure multiple states and may have limitations on the rate in which they are able to sample a given system. For a real-time system, energy, control rate, and computational constraints on the system as a whole may limit an estimator's ability to process all sensor measurements for a given time-step. In such scenarios, a process referred to as sensor scheduling is often used to optimize a pattern for either a set of sensors making measurements or which states a single sensor measures for a given time-step to reduce system error [32, 33, 34]. A high rate of sensor sampling may quickly become energy consuming and computationally intensive, where in contrast, the absence of state measurements for relatively long periods of time may result in a large accumulation of state error as the controller is unaware of deviations in the system state [32]. For systems with high sensitivity the latter may even result in instability.

For simple linear systems with Gaussian noise profiles, sensor scheduling solutions are well understood and can be obtained a priori as the noise profile is independent of the system [33]. However, for hybrid systems which may be adjusting physical and computational resources resulting in dynamic constraints on sensor scheduling, a more complex problem arises. As our co-regulation technique falls into an extreme case of such a hybrid system in which the sampling rate of the control task has the potential to vary at each discrete time-step, a real-time optimal sensor scheduling algorithm may be required. If the sensor update rate deviates largely from the control rate, large amounts of state error may accumulate. While operating at low control rates which may approach the limits on stability as the controller cannot respond as quickly to large deviations, poor sensor scheduling may be catastrophic. Conversely, limitations on sensor update rates and sensor scheduling constraints may constrict bounds on the region of stability for controllers designed outside of such sensor constraints.

Nevertheless, in this work we will assume ideal sensors and full state feedback such that sensors may sample infinitely fast while consuming infinitesimal amounts of energy or computational resources, and we leave sensor scheduling for future work.

2.2 Motion Planning

Autonomous robots are rapidly expanding their ability to perform a variety of tasks, requiring higher level motion planning techniques. For mobile autonomous robots, these motion planning techniques may vary widely in regard to the mission task or ability to assess the environment. In surveying structures or otherwise, coverage trajectories are often sought after where the robot is tasked with sweeping through an entire area to provide complete visual inspection or environment mapping [35][36][37]. In contrast, target interception/rendezvous trajectories are often concerned with optimal path calculation [38][39] and obstacle avoidance [40]. GPS and vision based navigation vary still in that the robot is better equipped to analyze feedback on a static or dynamic environment [41, 42][43].

In the field of autonomous robotic manipulators, a robot is typically given a series of waypoints to which a trajectory planner assigns target velocity and accelerations given the kinematics of the robot and limitations on velocity and acceleration specified by the manufacturer or through experimental results. At run-time a trajectory generator generates a continuous function which satisfies the requirements set forth by the trajectory planner via either a higher order polynomial spline or a linear approximation with higher order (parabolic) blends between linear segments [44]. Although linear functions with parabolic blends can better approximate a series of linear segments between waypoints (perhaps the desired result), this technique can only generate continuous functions for position and velocity. Therefore, for a system with inertia, this may result in extreme values of jerk (i.e. the first time derivative of acceleration) when switching between zero acceleration in linear segments and large accelerations in the parabolic blends, making it difficult for a robot to execute the generated trajectory and causing wear on actuators. As a result several techniques which optimize or place bounds on jerk have been developed using higher order polynomials, which provide for smoother transitions between values of acceleration [45, 46, 47].

The authors of [9] expand on several different approaches similar to a linear approximation with parabolic blends in which lower order dynamic constraints are enforced in motion planning. In the context of small fixed wing UAS, algorithms for following a series of straight-line segments with methods for waypoint switching including a sphere of acceptance and half plane penetration are explored. Note that this is similar to trajectory following techniques we employ in Chapters 3 and 4 with the caveat that we switch reference waypoints when the simulation reaches an associated time, rather than when the UAS reaches a certain point in space. This points to the differences in path following, which is independent of time, and trajectory following in which the vehicle is expected to adhere to time requirements as well as cross-tracking. It should also be noted that motion planning in a partially known or unknown environment based on behavioral methods and on-board sensor information, i.e. reactive motion planning, is subjugated here by a deliberative motion planning approach as outlined in [9] where explicit paths and trajectories are computed based on global knowledge of the environment. That is, waypoints are specified by a higher level planner (Voronoi graphs, Rapid Exploration of Random Trees (RRT) [9], A* [38], the user, etc.) in such a way that the vehicle will avoid any objects given it follows the provided waypoints to some degree of accuracy. However, as also stated in [9], deliberative motion planning has a strong dependence on the precision of the given dynamic model, which, in the case of UAS, given unpredictable environmental and feedback disturbances, is never sufficient; therefore, the motion planning algorithm must be constantly recomputed. As a result, simple lower order models must be used to ensure computational efficiency.

With the addition of circular orbit following algorithms for smoother transitions between straight-line segments [48], vector field path following techniques are outlined in [49, 50] which allow for path following even under large environmental disturbances. However, straight-line and orbital path following computation is performed under the constraint of a constant speed and therefore may result in a variety of paths depending on whether conservation of path length, minimum execution time, or waypoint interception is desired [48][49]. Alternatively, optimizing the combination of straight-line and orbital paths for the shortest path lengths, known as the Dubins path [51], is a common technique for fixed-wing aircraft [52].

Multirotor UAS, however, are much more agile than fixed-wing UAVs and have less demanding constraints on motion and velocity. A method exploiting the agility of a multirotor UAS is presented in [53] which relies on nonlinear modeling and control as well as higher order, optimization constraints on snap, the second derivative of acceleration which provides smooth accelerations and precise timing. Other aggressive maneuvers are achieved through event-triggered or scheduled trajectory generation to construct a trajectory as a series of sub-trajectories for which separate controllers have been designed [54]. Although such methods allow for impressive maneuverability, they may become computationally expensive in scenarios where precision performance or aggressive maneuvering is not prioritized. Other approaches, like that of [55], seek to minimize time in trajectory planning using optimal control coupled with nonlinear modeling and constraints as well as higher order polynomial approximations. In [56] a time optimal control method formulated in [57] is applied to a quadrotor UAS which employs limitations on control effort and is computationally inexpensive (up to 100 iterations of algorithm for each control input generated at 50 Hz). Similarly in [58] kinematic equations with maximum velocity and acceleration constraints are imposed to generate a sub-optimal, computationally simplistic trajectory while maintaining sufficient physical performance. A series of sweeps over the generated trajectory under acceleration and velocity constraints are used to optimize the trajectory with respect to time. Therefore, lower order optimal control techniques are feasible given model simplifications and fixed limitations on acceleration.

2.3 Application to UAS

Unmanned Air Systems (UAS) are a compelling platform on which to apply these methodologies as they may have very limited physical and computational resources due to size and weight restrictions in conjunction with increasing autonomy and mission demands. For example, an Ascending Technologies Hummingbird quadrotor system, a small to mid-sized research grade quadrotor UAS commonly used in the NIMBUS Lab[1], has a rotor-to-rotor diameter of 340 mm and a max payload of only ~200 g. A three cell 2200 mAh LiPo battery may supply ~20 min of flight time without payload [59]. The Hummingbird operates via a two level processor framework in which the low level processor (LLP) is mainly re-

[1]`http://nimbus.unl.edu`

sponsible for maintaining attitudes required for stable flight, and the high level processor (HLP) runs additional algorithms for processes such as path planning and sensor fusion. In the NIMBUS Lab, control software is used to autonomously drive the vehicle through a series of waypoints, or otherwise, to execute various missions. As an example, consider a quadrotor UAS tasked with surveying a complex environment containing many obstacles and boundaries using a suite of small, lightweight sensors. Computational tasks for the mission, such as video collection, image processing, storage, and communication of data, must contend for resources while the computer maintains a high quality of service [25] for basic autonomy tasks (e.g. control, planning, localization). Because sensors must be lightweight they may have reduced quality requiring more computation to compensate (e.g. advanced estimation schemes). Although high control authority may not be needed in all phases of flight, controllers are typically designed for worst-case noise, disturbances, and portion of the flight envelope, and time redundancy is often used in selecting a sampling rate to improve safety margins [13]. That sampling rate is enforced by the real-time computer system in which resource allocation is done through scheduling CPU cycles according to fixed, worst-case execution time (WCET) estimates and task periods of many tasks competing for execution, which (ideally) provides timing guarantees for each task [11]. However, because WCET and task periods are constant, and typically over-designed using time redundancy as a safety margin, tasks are allocated fixed resources regardless of their performance or needs at run-time.

This fixed design for the worst case scenario has consequences. For a surveillance quadrotor UAS during more quiescent periods of flight, it means over-allocated resources in flight control are wasted while other processes such as data collection may be limited by fixed resources. Similarly, in executing an aggressive maneuver, flight control tasks may be at a loss for resources while data collection resources are idly wasted. In contrast, in a UAS that dynamically allocates resources according to control and mission performance, those resources can be diverted to improving image processing, sensing, communication or other surveillance tasks when appropriate, or towards increasing control authority if desired.

In cyber-physical control of a UAS, this trade off between physical performance and computational resources is capitalized on. However, in dynamically allocating resources, the impacts of variability on performance, while maintaining stability, must first be understood. Typically, a tight feedback control loop is used to provide reactive behavior to the vehicle [9]. This loop is composed of physical components and cyber components. Initially, physical sensors representing system properties are read, and translated into the digital signals fed into a computer. A controller, modeled in software and executed as a real-time computational task, reads these sensed values and computes a digital control input [11]. The digital control input is then converted into a continuous signal and fed to actuators. This control input is “held” (a zero-order hold) until the control task executes again, restarting the cycle. The question of how often the control task should be executed is governed by the sampled-data assumption, and chosen by the control engineer according to various rules of thumb typically involving noise bandwidth, eigenvalues, and the Nyquist frequency [18, 60].

For quadrotors, control systems and their real-time requirements have been studied [61, 62]. Others have implemented an event-triggered control system for attitude stabilization, which is more resource aware [22]. In related work, the authors examined the response-time constraints for a real-time controller implemented onboard the quadrotor [63]. They analyze the response rate of actuators at different operating conditions in order to design a controller that has an update rate of at least as much as the sampling rate of the various sensors. Seghour et al. [64] implemented a real-time embedded control system for stabilizing a quadrotor; however, they do not reason about the response time or the control rate chosen.

These methods demonstrate and leverage traditional control analysis techniques by assessing controller response to step inputs – the generally accepted strategy in controller design [18]. However, assessing trajectory following performance as a function of cyber resource allocation provides another trade-off to exploit in the pursuit of dynamic resource allocation for the holistic cyber-physical system. Here, we extend traditional controller analysis by investigating trajectory following performance and providing the mathematical relationship needed to apply a full cyber-physical control and planning architecture for a UAS. This architecture will enable a more dynamic UAS that can adjust computation in response to performance at both a low, reactive control level, as well as a higher, deliberative planning level.

2.4 Our Work

The culmination of the state of the art discussed above illustrates that cyber and physical effectors in quadrotor UAS are not typically allocated efficiently. With knowledge of the cyber and physical characteristics and limitations, real-time allocation of resources in a CPS like a quadrotor UAS may then increase performance and functionality. The rest of this work is structured as follows. In the following chapter we seek to characterize the effects of fixed rate (Riemannian sampling) control at different rates as the UAS executes a trajectory (without a higher level generation technique). We begin by defining nonlinear and linear physical models for the physical system followed by design of the physical control law, a discrete linear quadratic regulator, by which control inputs will be generated for the physical system. Analysis is conducted to illustrate the characteristics of the physical system across a broad range of sampling rates. Then, in Chapter 4 the CPS design is completed with the addition of a cyber control law. We implement our hybrid Riemannian/Lebesgue co-regulation technique and conduct analysis of cyber and physical system effectors. Our variable rate co-regulation implementation is compared to fixed rate techniques in driving the quadrotor UAS through a trajectory (again without a higher level generation technique). In Chapter 5, we present a suitable trajectory generation technique as a first step toward a higher level motion planning architecture which may be used to provide better CPS performance and deliberation. Finally conclusions and a brief outline of future research aims are provided.

Chapter 3

Characterization of UAS Performance Under Discrete Control

In this chapter we will examine the affects of sampling period in discrete control of a quadrotor UAS system. We seek to characterize the relationship between sampling rate and physical performance of the system in execution mission trajectories. In order to do so we define a system model and provide analysis of trajectory following performance for controllers designed across a wide range of sampling rates. Effective analysis is achieved through the definition of several CPS metrics designed to measure cyber-physical performance. This provides insight into a suitable range of sampling rates for which dynamic allocation of cyber-physical resources may be most effective.

Much of the work presented in Chapter 3 was conducted in collaboration with Ajay Shankar, a graduate student researcher in the NIMBUS Lab[1] at the University of Nebraska-Lincoln, USA, and published in the International Conference on Unmanned Aircraft Systems [65].

[1]`http://nimbus.unl.edu`

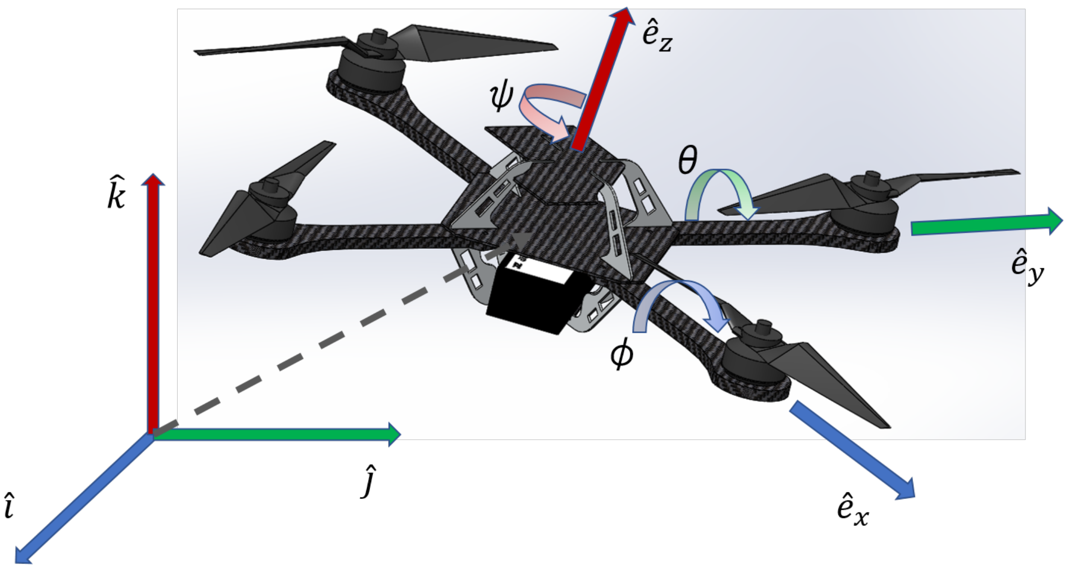


Figure 3.1: Quadrotor frame orientations.

3.1 Quadrotor UAS System Model

3.1.1 Nonlinear Physical Model

A quadrotor is a six degree of freedom system in which translational movements are achieved by rotational displacements generated by combinations of individual rotor thrusts. The system is under actuated as its six degrees of freedom must be controlled by four inputs: either individual rotor commands or a net upward thrust generated collectively by all four motors, and pitch, roll, and yaw moments generated by thrust imbalances between pairs of rotors [66, 67, 68]. Being underactuated, the system requires active control, often via autopilot software, to retain stable flight. The flight dynamics of a quadrotor UAS are nonlinear, and although several methods for nonlinear control of quadrotor vehicles exist [69, 70, 71, 72], typically a linear system is used for control design. In this work we leverage nonlinear equations for high-fidelity simulation, and use a linearized system model for control design as it is effective and potentially less computationally intensive than nonlinear control.

The system state consists of the vehicle's position $\tilde{X}_p = (x, y, z)^T$ in an inertial frame $\{i, j, k\}$, orientation in roll (ϕ), pitch (θ), and yaw (ψ) angles of the vehicle frame $\{\hat{e}_x, \hat{e}_y, \hat{e}_z\}$ with respect to the inertial frame (see Figure 3.1), velocity in $\mathbb{R}^3$, and angular rate of change in roll, pitch, and yaw,

$$X_p = \left(x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}\right)^T .$$

We derive the equations of motion via Newton and Euler's equations in the body frame of the UAS with respect to an inertial world frame similar to the procedures described in [67, 72]. The nonlinear equations in $\mathbb{R}^3$ are

$$\begin{aligned} \boldsymbol{a} &= \boldsymbol{n}/m + \boldsymbol{f}_{drag}/m - \boldsymbol{g} \\ \boldsymbol{\alpha} &= \boldsymbol{I}^{-1}\left[\boldsymbol{\tau} - (\boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega})\right], \end{aligned} \tag{3.1}$$

where $\boldsymbol{a}$ is the linear acceleration of the UAS, $\boldsymbol{n}$ is the net thrust vector, m is the total mass, $\boldsymbol{f}_{drag}$ is the force due to drag, $\boldsymbol{g}$ is the gravitational force vector, α is the angular acceleration, $\boldsymbol{I}$ is the moment of inertia with respect to the center of the vehicle (also assumed to be center of mass), $\boldsymbol{\tau}$ is a vector of torques produced by the vehicle's rotors, and $\boldsymbol{\omega}$ represents angular velocity. Here, $\boldsymbol{\alpha}$, $\boldsymbol{\omega}$, $\boldsymbol{\tau}$, and $\boldsymbol{I}$ are calculated about principal axes of the UAS, and corrections are imposed relating $\boldsymbol{\alpha}$ and $\boldsymbol{\omega}$ to state variables $\dot{\phi}, \dot{\theta}, \dot{\psi}$, and their derivatives $\ddot{\phi}$, etc. (see Appendix A).

3.1.2 Linear Physical Model

For controller design we linearize the nonlinear Equations (3.1) about a stable hover (see Appendix B). From this procedure we derive the system matrices $\boldsymbol{A}_p$ and $\boldsymbol{B}_p$ [67, 72]

relative to the physical system state X_p. This provides a traditional linear state space model

$$\dot{X}_p = \boldsymbol{A}_p X_p + \boldsymbol{B}_p U_p. \tag{3.2}$$

$$\boldsymbol{A} = \begin{bmatrix} & & \mathbf{0}_{6\times6} & & & \mathbb{I}_{6\times6} & & \\ & 0 & g & 0 & \frac{-D_x}{m} & 0 & 0 & \\ \mathbf{0}_{3\times3} & -g & 0 & 0 & 0 & \frac{-D_y}{m} & 0 & \mathbf{0}_{3\times3} \\ & 0 & 0 & 0 & 0 & 0 & \frac{-D_z}{m} & \\ & & \mathbf{0}_{3\times6} & & & \mathbf{0}_{3\times6} & & \end{bmatrix}$$

$$\boldsymbol{B} = \begin{bmatrix} & \mathbf{0}_{8\times4} & & \\ 0 & 0 & 0 & 1/m \\ I_{xx}^{-1} & 0 & 0 & 0 \\ 0 & I_{yy}^{-1} & 0 & 0 \\ 0 & 0 & I_{zz}^{-1} & 0 \end{bmatrix}$$

where $\mathbf{0}_{a\times b}$ is a sub-matrix of a by b dimensions consisting of all zeros, $\mathbb{I}_{a\times b}$ is an identity sub-matrix of a by b dimensions, m and g are the mass of the vehicle and acceleration due to gravity, D_x, D_y, and D_z are the coefficients of linear drag forces acting in each of the coordinate axes, and I_{xx}, I_{yy} and I_{zz} are the mass moments of inertia of the quadrotor's body about the pitch, roll and yaw axes respectively. The input vector, U_p, consists of independent torques in roll (τ_ϕ), pitch (τ_θ), and yaw (τ_ψ), and the magnitude of the net thrust (N)

$$U_p = \left(\tau_\phi, \tau_\theta, \tau_\psi, N\right)^T.$$

Note that in linearizing about an operating point in which the vehicle is in a stationary hover, the net thrust input N is biased for a vertical component of gravity (i.e. z-axis of the body frame) which does not appear in the system matrix, $\boldsymbol{A}$, (see Appendix B). In our linear model all inputs are considered independent of each other so long as motor saturation is not reached. As a result, the input values are each constrained to physical specifications provided by the manufacturer [59] so that the motors operate safely in a range of 1-70 percent of max power.

3.2 LQR Formulation

3.2.1 Physical System Control

To allow the UAS to follow a trajectory we implement non-zero position reference tracking by adding three integrator states to the state vector and subsequently augment the system matrices [60]. The reference state is a position vector in $\mathbb{R}^3$ represented by $\tilde{X}_{p,ref} = (x, y, z)^T$. In our trajectory guidance algorithm we enable following of an arbitrary trajectory by submitting a series of reference points, $\tilde{X}_{p,ref}$, to the controller at the appropriate times. The augmented system is

$$\begin{aligned}
\dot{X}_{p,aug} &= \boldsymbol{A}_{p,aug} X_{p,aug} + \boldsymbol{B}_{p,aug} U_p + \boldsymbol{B}_{p,r} \tilde{X}_{p,ref} \\
X_{p,aug} &= \left(x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}, 0, 0, 0\right)^T \\
U_p &= (\tau_\phi, \tau_\theta, \tau_\psi, N)^T \\
\boldsymbol{A}_{p,aug} &= \begin{bmatrix} \boldsymbol{A}_{p,12\times12} & \mathbf{0}_{12\times3} \\ \mathbb{I}_{3\times3} & \mathbf{0}_{3\times12} \end{bmatrix} \\
\boldsymbol{B}_{p,aug} &= \begin{bmatrix} \boldsymbol{B}_{p,12\times4} \\ \mathbf{0}_{3\times4} \end{bmatrix}, \quad \text{and} \quad \boldsymbol{B}_{p,r} = \begin{bmatrix} \mathbf{0}_{12\times3} \\ -\mathbb{I}_{3\times3} \end{bmatrix}.
\end{aligned} \tag{3.3}$$

Equations (3.2) and (3.3) are expressed in the continuous time domain. However, since they will be implemented in software in an embedded, real-time system, we employ digital control techniques to transform the system into the discrete domain for control design. For this, the plant is discretized at a fixed sampling rate using a zero order hold (ZOH) approximation, in which the control inputs are held constant between updates. The discrete system can be obtained by

$$\mathbf{\Phi}_p = e^{\boldsymbol{A}_{p,aug}T_d}, \text{ and } \mathbf{\Gamma}_p = \int_0^{T_d} e^{\boldsymbol{A}_{p,aug}\eta}\boldsymbol{B}_{p,aug}d\eta$$

where $\mathbf{\Phi}_p$ and $\mathbf{\Gamma}_p$ are the discrete time counterparts of system matrices $\boldsymbol{A}_{p,aug}$ and $\boldsymbol{B}_{p,aug}$ respectively, T_d is the fixed sampling period of the control task, and η is an integration variable. The discrete time linear control law with respect to time step k is then,

$$X_{p,aug}[k+1] = \mathbf{\Phi}_p X_{p,aug}[k] + \mathbf{\Gamma}_p U_p[k] + \boldsymbol{B}_{p,r}\tilde{X}_{p,ref}. \tag{3.4}$$

as described in [18, 17].

Once the system matrices are computed for a given sampling period, we design a DLQR controller to maintain a stable hover at a given reference. Note that in the DLQR, the gain matrix, $\boldsymbol{K}_{T_d}$, is specific to the sampling period T_d used to generate the $\mathbf{\Phi}_{T_d}$ and $\mathbf{\Gamma}_{T_d}$ matrices [18]. This implies that a real-time system designed with the control task executed at a different periodicity, or under jitter or missed deadline conditions, will result in poor performance of the controller.

3.3 Real-time Requirements

If the control signals are not generated in a timely sequence, the quadrotor may become unstable [60]. As a result, flight control code must be executed correctly and completely before a specified deadline. A controller and planner implemented in software must consider the dynamics and limitations of the vehicle and on-board sensors as well as the timing and scheduling of software tasks in the computer. A low control task period (high sampling rate) can better approximate a continuous model, potentially offering better performance at the expense of computation. Conversely, a high task period (low sampling rate) is easier to achieve computationally amongst many competing autonomy-related tasks, but this may be detrimental to performance. This is compounded in a digital system by sensor values and control inputs that are "sampled and held" until the next time the control task is executed [18]. In this duration, the vehicle continues to react based on its dynamics, possibly becoming unstable.

In computer-based control, the sampled-data assumption is often employed to construct digital controllers that are executed periodically according to a real-time schedule. The sampled-data assumption presumes that sensors are read, and the control input is calculated and sent to the actuators at a single, and periodically recurring, instant of time, t. It assumes there is no delay in the states or control input, only that the control input is then held by the actuators for the entire sampling period, T_d, called a zero-order hold (ZOH) [18]. The sampling period of the discrete system is, ideally, matched by the control task execution period enforced by the real-time computing schedule [11]. However, in a real-time system, the only timing guarantee is that deadlines will be met, not that the period between task completion times is consistent. This means that control inputs may not be given at regular time intervals, and there will be state delay in the control input calculation (thus violating the sampled-data assumption) [3].

Consider an implication following this assumption. From the simplified model in Equation (3.2), the angular roll acceleration at a given time t can be written as:

$$\ddot{\phi}(t) = \frac{\tau_\phi(t)}{I_{xx}}.$$

We compute the rotation angle by integrating $\ddot{\phi}$ twice over the $(k+1)^{th}$ discrete time-interval, where $k \in [0, t/T_d]$, as follows:

$$\phi = \frac{1}{I_{xx}} \int_{kT_d}^{(k+1)T_d} \left(\int_{kT_d}^{(k+1)T_d} \tau_\phi(t) dt \right) dt.$$

In this case the input torque is held constant throughout each sampling period and is only recomputed at the end of the step. Therefore,

$$\tau_\phi(t) = K_{T_d,\phi} \tau_\phi^{max}, \quad kT_d \le t < (k+1)T_d,$$

where τ_ϕ^{max} is the maximum torque that can be applied, and $K_{T_d,\phi}$ is the roll gain constant chosen suitably for a given T_d. Plugging $\tau_\phi(t)$ into the relationship for ϕ, we have,

$$\begin{aligned} \phi &= \frac{1}{I_{xx}} \int_{kT_d}^{(k+1)T_d} \left(\int_{kT_d}^{(k+1)T_d} K_{T_d,\phi} \tau_\phi^{max} dt \right) dt \\ &= K_{T_d,\phi} \frac{\tau_\phi^{max}}{I_{xx}} T_d^2, \end{aligned} \tag{3.5}$$

which implies a quadratic relationship between the angular displacement and the amount of time the input is applied for. Using Equation (3.5) we can compute the maximum amount of time the full input can be applied to the system while keeping the angular rotations within vehicle limitations.

Because of the zero-order hold behavior of discrete systems, Equation (3.5) has significant implications. The control system has a strong real-time dependency - if a deadline for a new commanded input is missed, and since $K_{T_d,\phi}$ is a constant, the rotation angle may become unbounded. It follows that if the discrete sampling period, T_d, is too large, then the gain, $K_{T_d,\phi}$ needs to be reduced to prevent input saturation.

In a cyber-physical UAS, the same processor executes a multitude of computing tasks, and CPU cycles may become a contended resource that must be allocated appropriately. Small and infrequent delays in meeting the deadlines imposed by the choice of sampling period may lead to a degraded quality of service and instability, but also may be accounted for by time redundancy [25]. While repeated or consistent timing misses may cause an unbounded response on the UAS, committing a larger amount of CPU-time to UAS flight performance may degrade the performance of another important service task (e.g. sensing, data collection). As a result, understanding the limits and implications of sampling rate for the holistic cyber-physical system allows us to balance these resources over the course of a mission.

Finally, it is critical to note that a given digital control strategy is a function of both controller design and selected sampling period. That is, changing the sampling period, even while holding control design variables constant, results in a different value of $\boldsymbol{K}_{T_d}$, effectively re-characterizing the controller itself [18]. The implication is that intelligently trading cyber and physical resources requires us to develop new control techniques that account for the nonlinear relationship between digital, linear control design and sampling rate.

3.4 Experimental Setup

Our simulation experiments are designed to demonstrate the effect of varying the software control task period, or sampling period, of flight control tasks as the UAS executes a mission trajectory.

The nonlinear equations discussed in Section 3.1 are used to model and simulate the flight of the quadrotor. The controller utilizes a discrete-time linear quadratic regulator (DLQR) strategy on the augmented *linear* system in Equation (3.3), allowing it to drive the vehicle to a given reference point in space and time. By varying the reference with respect to time, a trajectory is generated for the UAS to follow as the simulation progresses.

We choose LQR control for two reasons: 1) LQR has a closed-form solution and is a stabilizing optimal control algorithm with good margins of stability, thus making it easier to compare controllers with similar performance at different sampling rates; and 2) in finding the optimal gain, LQR minimizes the error on the state vector, $X_{p,aug}$, and the control input, U_p. This is particularly useful for UAS applications where a large control input may be undesirable. Because our objective is to isolate the relationship between sampling rate and trajectory following performance, we use a MATLAB-based nonlinear equation simulation with full state feedback and do not model external disturbances or sensor noise. Under the sampled data assumption we assume ideal sensors are read and control inputs are calculated and transmitted to actuators without delay or uncertainty.

In order to study the effect of a changing sampling period, we perform a sweep over a range of values for T_d, generating a corresponding set of $\mathbf{\Phi}_{T_d}, \mathbf{\Gamma}_{T_d}$, and $\boldsymbol{K}_{T_d}$ matrices. Because we are interested in a highly accurate relationship between sampling period and trajectory following performance, we use a fourth-order Runge-Kutta ordinary differential equation solver, `ode45` in MATLAB, to simulate the system using the nonlinear equations in Equation (3.1). However, `ode45` is a continuous-time solver, unsuited to the zero-order hold paradigm. As a result, we leverage it as part of a larger simulation technique designed to simulate both the correct zero-order hold behavior and corresponding transients of the system response for each time period during which the input is held, $0 \leq j < m$ where j represents an internal `ode45` time step on $kT_d \leq t < (k+1)T_d$. These modifications are described as follows.

Algorithm 1 Algorithm to simulate the control of quadrotor flight at one discrete sampling rate.

```
Data: Nonlinear system model, $f(X)$, $sampling\_period$
// initialize system constants
$T_d \longleftarrow sampling\_period$
$lin\_model \longleftarrow$ linearize($f(X), T_d$)
$\boldsymbol{Q} \longleftarrow 10 \cdot \mathbb{I}_{15\times15} \cdot q$
$\boldsymbol{R} \longleftarrow 2 \cdot \mathbb{I}_{4\times4}$
begin
    $\boldsymbol{\Phi}_{T_d}, \boldsymbol{\Gamma}_{T_d} \longleftarrow$ discretize($lin\_model, T_d$)
    $\boldsymbol{K}_{gain} \longleftarrow$ dlqr($\boldsymbol{\Phi}_{T_d}, \boldsymbol{\Gamma}_{T_d}$)
    $k \longleftarrow 0$
    $X_{all} = [\,]$
    $X_{init} \longleftarrow$ initial_state()
    while $kT_d \leq simulation\_length$ do
        $U_k \longleftarrow$ input_vector($X_{init}, \boldsymbol{K}_{gain}$)
        $[X_1 \ldots X_m]$ = ode45($f(X), T_d, U_k\ X_{init}$)          // Simulate
        $X_{all} = [X_{all}; X_1 \ldots X_m]$
        $k \longleftarrow k + 1$                    // propagate discrete time step
        $X_{init} \longleftarrow X_m$                            // new initial state
    end
end
```

An outer loop iterates over discrete time steps, k, computing and holding $U_p[k] = -\boldsymbol{K}_{T_d} X_p[k]$ for the sampling period duration T_d. That is, $U_p(t) = U_p[k]$, where $kT_d \leq t < (k+1)T_d$. Within each sampling period, $U_p[k]$ is passed and held as an input to the nonlinear system model, which is simulated using `ode45`. The initial system state for each discrete step is the final state propagated by `ode45` in the previous iteration. Because `ode45` is a one-step solver, the output from each execution of `ode45` can be appended to the previous one to put together a complete continuous system response. This ensures that the control system follows a discrete-time sample and hold behavior, but we also obtain the transient response for each sampling interval. A pseudo code for the algorithm that simulates the system for a specific sampling period is shown in Algorithm 1.

This simulation strategy was previously outlined in [3]; however, here we improve it in several key ways. First, we increase the fidelity of the simulation by using full nonlinear equations to simulate the movement of the vehicle through space and time. Second, we developed a trajectory generation, or guidance algorithm which deconstructs high-level plans into a series of waypoints which are passed to the co-regulation framework at the appropriate time. This gives us the ability to assess trajectory-following performance providing a better indication of how a controller affects overall mission performance.

The simulation is run with the UAS initially in a stable hover at the origin of the inertial frame, and with the parameters listed in Table 3.1 which are specific to the Ascending Technologies Hummingbird [59], a medium-sized, general purpose research UAV used in the NIMBUS Lab. Uniform and manually tuned $\boldsymbol{Q}$ and $\boldsymbol{R}$ values (also in Table 3.1) were used in designing each of the DLQR controllers.

Parameter	Value	Parameter	Value
g	$9.80665\,\mathrm{m/s^2}$	m	$0.515\,\mathrm{kg}$
D_x	$0.0075\,\mathrm{kg/s}$	I_{xx}	$0.0040\,\mathrm{kg\,m^2}$
D_y	$0.0075\,\mathrm{kg/s}$	I_{yy}	$0.0040\,\mathrm{kg\,m^2}$
D_z	$0.015\,\mathrm{kg/s}$	I_{zz}	$0.0044\,\mathrm{kg\,m^2}$
$\boldsymbol{Q}$	$10\mathbb{I}_{15\times15}\boldsymbol{q}$	$\boldsymbol{R}$	$2\mathbb{I}_{4\times4}$
$\boldsymbol{q} = [100\ 100\ 100\ 100\ 100\ 100\ 10\ 10\ 10\ 1\ 1\ 1\ 50\ 50\ 50]^T$			

Table 3.1: System Constants

3.5 CPS Metrics

The ability of a UAS to follow a given path is often expressed through a controller's ability to minimize cross-track error [9, 73]. This analysis may be coupled with a measurement of control effort to gain some understanding of the amount of energy expended by the controller through inputs to the system [73, 74]. Additionally, analysis of different trajectories in relation to one another may involve estimates of power or energy consumption [75]. To quantify the relationship between sampling rate and trajectory following performance, we

Name	Abbreviation	Description
Cumulative State Error	CSE	*cumulative time weighted average of positional state error*
Maximum State Error	MSE	*maximum deviation from straight line path*
Control Effort	CE	*time weighted squared average of control input*

Table 3.2: CPS metrics

introduce several different CPS metrics [65]. We design these metrics to capture the performance of a controller in tracking a given trajectory and minimizing physical state error as well as control effort. The metrics also place an emphasis on the controller's ability to reduce error in a timely manner. Each of these metrics can be computed for a specific sampling rate and simulation, which can then be combined to form an explicit mathematical relationship between sampling rate and trajectory following performance. Although we characterize these metrics in the context of a quadrotor UAS, they are, in principle, more generally applicable to a broad class of cyber-physical vehicle systems. These metrics are listed in Table 3.2 and described below.

3.5.1 Cumulative Time-Weighted State Error

In our experiments, the reference point, $\tilde{X}_{p,ref}$, is given at specific time intervals throughout the simulation. Therefore, by changing the position of the reference waypoint with respect to time, the UAS can be commanded through a desired trajectory. We define the state error at any given time step, k, as the squared Euclidean distance between $\tilde{X}_p[k]$ (the position states from the current state vector $X_p[k]$) and the corresponding reference state vector at that time, $\tilde{X}_{p,ref}[k]$. Given the reference state to which the controller must drive the system, a cumulative time-averaged state error (CSE) can be defined for the position state vector over the entire mission. Weighting each state error by the simulation time yields the following equation for cumulative time weighted average of state error,

$$\text{CSE} = \frac{1}{t_{tot}} \sum_{i=0}^{n} t_i \left(\tilde{X}_p(t_i) - \tilde{X}_{p,ref}(t_i) \right)^2 \tag{3.6}$$

where $\tilde{X}_p$ is the vehicle's position in $\mathbb{R}^3$, t_i is `ode45`'s discretization of continuous time (t), t_{tot} is the total amount of simulation time, n is the total number of internal simulation steps, i.e. $i \in [0, n]$. Weighting by time has the advantage of more aggressively penalizing the state error as time progresses, while having a smaller weight associated with initial offsets in the system. This definition of state error as a metric for system design captures the effectiveness of a cyber control system in driving the state of the physical system to the reference state within a short amount of time, and with minimal overshoot.

3.5.2 Translational Bounds

The ability to place bounds on maximum offsets from a desired trajectory is useful in planning mission objectives and helps to identify worst case flight envelopes and failure states. As the controller responds to commanded target waypoints, this metric of maximum state error (MSE) determines the farthest point the UAS reached from the ideal desired trajectory line connecting two successive target waypoints, $\boldsymbol{l} = \tilde{X}_{p,ref,next}(t_i) - \tilde{X}_{p,ref,prev}(t_i)$,

$$\text{MSE} = \max \left(\frac{||\boldsymbol{l} \times (\tilde{X}_{p,ref,prev}(t_i) - \tilde{X}_p(t_i))||}{||\boldsymbol{l}||} \right). \tag{3.7}$$

where $|| \cdot ||$ represents the magnitude of a vector quantity. This metric is used to set a standard for mission success which hinges on whether or not the UAS remained within a desired maximum distance from the given path throughout execution and could be used to determine possible failure states in order to invoke a contingency control strategy.

3.5.3 Control Effort

For a quadrotor UAS, minimization of control effort is essential for decreasing power and energy demands thereby preventing possible damage to components and potentially increasing vehicle endurance.

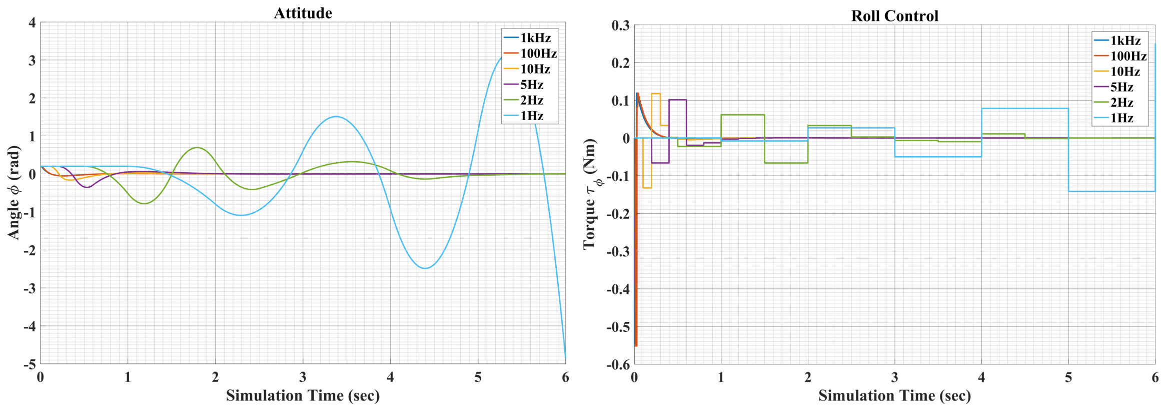
To analyze the control effort of the system, we compute the time-averaged control effort (CE) over all simulation time as follows:

$$\text{CE} = \frac{1}{t_{tot}} \sum_{i=0}^{n} U_p(t_i)^2 t_i, \tag{3.8}$$

where $U_p(t_i) = U_p[k] = \text{const. on } kT_d \leq t_i < (k+1)T_d$. As before, weighting the value of the control effort with the simulation time rewards the natural response of the system to a reference step, which, generally would require less control effort as error is reduced. This metric is also proportional to the energy consumed for propulsion and is approximately proportional to total energy consumed in a system where propulsion dominates energy resources.

In part, this metric is motivated by the mathematical realization that a controller with higher gains may more quickly converge to the reference state by generating larger control inputs for a shorter amount of time (without saturating). This metric favors such a controller, as compared to one which applies smaller inputs for a longer duration, thus taking longer to converge.

Intuitively, we expect each of these metrics to increase as the sampling period is increased. That is, with longer sampling periods there should be higher state error, a higher control effort (CE), and a typically larger maximum deviation from the ideal trajectory.



(a) Rejecting an initial roll angle of $\phi = 0.2$ radians for different sampling rates.

(b) Rolling torque, τ_ϕ, generated by the controller at different sampling rates in order to bring the vehicle to a stable hover.

Figure 3.2: Disturbance rejection on the roll axis at different sampling rates. A higher sampling rate results in higher controller gains and more aggressive control. Lower sampling rates result in a more narrow control input operating range, but the response also takes longer to converge to the reference state. Note that the response for 1 kHz (blue) is nearly identical to the 100 Hz plot, and is obscured by it.

3.6 Results

We now present the results of several important test cases representing various scenarios we regularly find in our UAS missions.

3.6.1 Traditional Disturbance Rejection Experiment

We begin by first characterizing the performance of the controller in rejecting a disturbance represented by an initial non-zero attitude angle. This represents a traditional control system performance metric - evaluating a step response. At time $t = 0$, the system is initialized at the origin of the inertial space $\boldsymbol{p} = (0, 0, 0)$ with $\phi = 0.2$ rad and other components of the state vector set to zeros.

Figure 3.2a shows the progression of ϕ for different sampling rates as the controller brings the system to a stable hover at the origin. As expected, a higher sampling period results in a longer settling time and larger overshoot.

Although DLQR is a stabilizing controller, it stabilizes the linear approximation of the nonlinear system. However, the stable region of a closed-loop nonlinear system changes with the sampling period [76]. As a result, there are states that may exceed the bounds on disturbances from which the DLQR controller can recover. We hypothesize this is the case for the DLQR controller designed and operated at a 1.0 s sampling period in Figure 3.2a. The initial condition 0.2 rad exceeds the region of stability for the DLQR controlled nonlinear system. The control input required to reject the disturbance is shown with different sampling rates in Figure 3.2b. We note that for high sampling rates, as anticipated, the control input changes in a much smoother fashion, but the maximum control effort required is higher. For lower sampling rates, however, the maximum control effort is smaller in magnitude, and the system takes longer to settle.

3.6.2 Trajectory Following Experiments

We now assess the controller's performance in following a single, straight line trajectory by driving the vehicle to a point $\boldsymbol{p} = (x_1, y_1, z_1)$ in space, starting from a stable hover at the origin. The commanded reference is held constant throughout the length of the simulation so that the controller causes the vehicle to go to, and hover at, $\boldsymbol{p}$. In the following subsections, we use this test to analyze the various metrics defined previously.

Finally, to assess complex trajectory following performance, we develop trajectories consisting of reference waypoints and issue commands to the vehicle to follow. We design the framework such that a new waypoint might be made available at any time instant, whether the vehicle has reached its current waypoint or not. Therefore, if several distant waypoints arrive in quick succession, it is not necessary that the vehicle would ever reach any single one of them. This design decision was based on a cyber-physically co-regulated and co-optimized UAS with a mission planner that could decide whether reaching each waypoint in a complex trajectory may be subjugated by the desire to conserve resources.

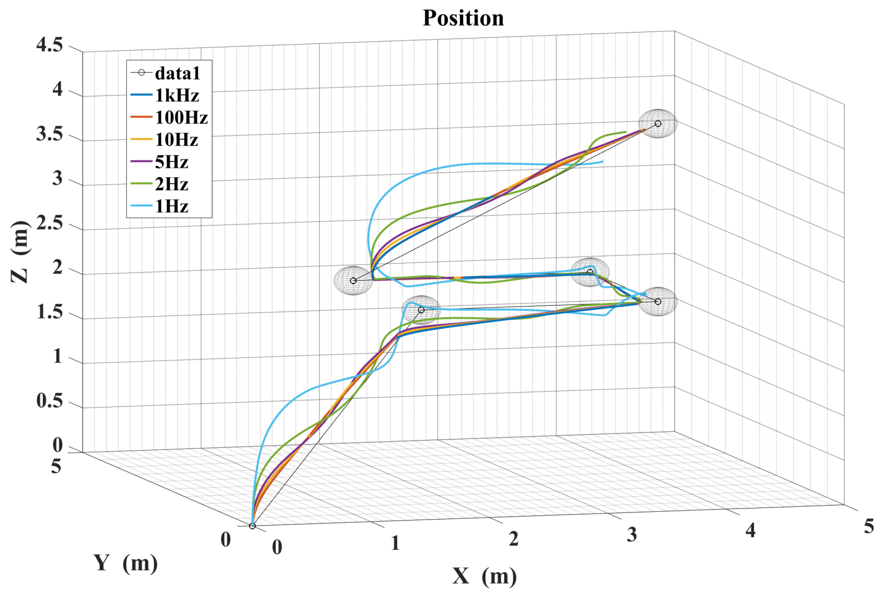


Figure 3.3: The paths taken by the UAS as it follows five commanded waypoints in space at different sampling rates of the controller.

This may then be achieved by allowing for less aggressive control (conserving computational resources) at the expense of precision tracking.

Figure 3.3 shows the path taken by the UAS as it follows five commanded waypoints in space. The effect of a low sampling rate is clear for certain course legs, most notably the first and the last ones. This becomes less predictable as the simulation progresses. For instance, for three consecutive waypoints $\boldsymbol{p}_i$, $\boldsymbol{p}_j$ and $\boldsymbol{p}_k$, if the angle between the two consecutive waypoints, $\boldsymbol{p}_i\boldsymbol{p}_j$ and $\boldsymbol{p}_j\boldsymbol{p}_k$, is obtuse, then it is possible that the vehicle undershoots the waypoint $\boldsymbol{p}_j$ and is then better poised to reach $\boldsymbol{p}_k$. Because DLQR is a stabilizing controller, as long as the states of the vehicle remain within the region of stability of the closed-loop nonlinear system, the controller will always recover. As a result, contrary to the step response in Figure 3.2a, where initial conditions were beyond the disturbance limits that ensure stability, in Figure 3.3 the vehicle successfully navigates the trajectory, although with reduced performance. If design of the control system includes similarly large sampling period, a more rigorous mathematical characterization of the bounds on disturbances and regions of stability, similar to [76], is needed.

3.6.3 Characterizing the Relationship Between Trajectory Following Performance and Sampling Period

We now demonstrate the relationship between trajectory following performance of our UAS and the sampling period of the software control task using the metrics we discussed in Section 3.5.

3.6.3.1 Variation of Gain and H_2-norm

We noted in Equation (3.5) that for a larger sampling period, the gain of the system should decrease. Since we have multiple elements in the input vector, we quantify the control gain here as the L2-norm of the $\boldsymbol{K}_{T_d}$ matrix. Another useful analysis tool is the H_2-norm, which represents the energy of the output of the system [24]. This tool can be used to identify potentially destabilizing intermediate sampling periods of the system if the H_2-norm is infinite at a given sampling period. We perform a sweep on a wide range of sampling periods and plot the variation in the controller gain and the H_2-norm of the system in Figure 3.4. Much like the analysis in [12], gain decreases with sampling period.

In our case, where the DLQR design parameters remain constant as we change sampling period, analysis of the gain vs. sampling period curve in Figure 3.4 can be used to select the lowest feasible sampling period of control as long as motor saturation is not reached. In practice, however, sampling rate will most likely be limited by constraints in the cyber system (i.e. how much processing time can be devoted to control computation).

The high-gain operation of the controller at small sampling periods can potentially saturate the actuators, thereby violating the assumption that the thrust and torques on each

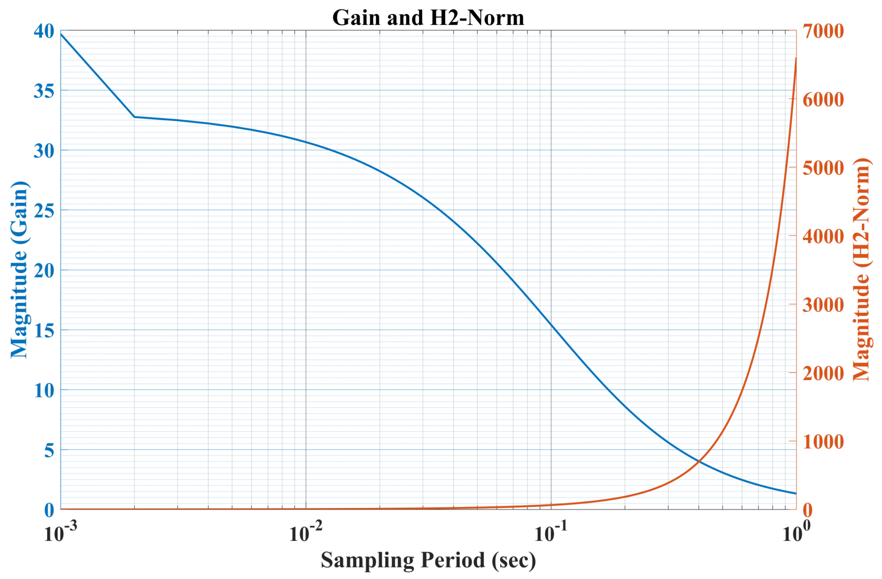


Figure 3.4: Variation in the controller gain and the H_2 norm of the discretized system across different sampling periods.

of the axes are independently controllable. Knowing the matrix $\boldsymbol{K}_{T_d}$ designed for a specific T_d, and a given state vector $X_p(t)$ at time t, we can check for saturation:

$$U(t) = -\boldsymbol{K}_{T_d} X_p(t) \ \leq U_{p,max}$$

where $U_{p,max}$ is determined appropriately using maximum rotor thrust from system specifications [59].

3.6.3.2 State Error

We introduced the cumulative time-weighted state error (CSE) and defined it as a metric to characterize the performance of a controller over a trajectory leg. Observing the traversed paths in Figure 3.3, we expect this metric to increase in magnitude as the sampling period increases. The maximum deviation from the trajectory leg is also expected to increase, as the sampling period becomes longer, due to the sample and hold nature of control. The trend in these two metrics is captured first in Figure 3.5 representing a single trajectory

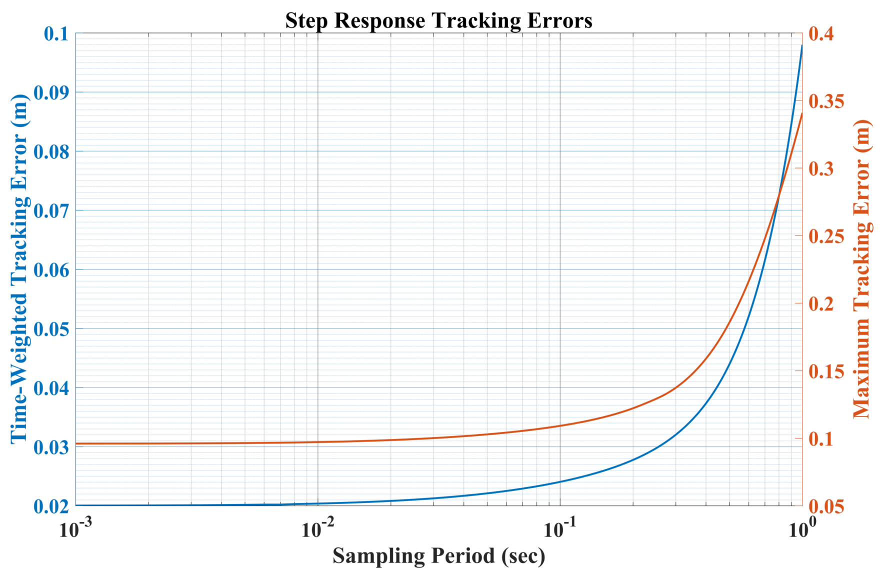


Figure 3.5: The change in average trajectory tracking error and the maximum deviation from the trajectory for a single leg (step response) as the sampling period changes.

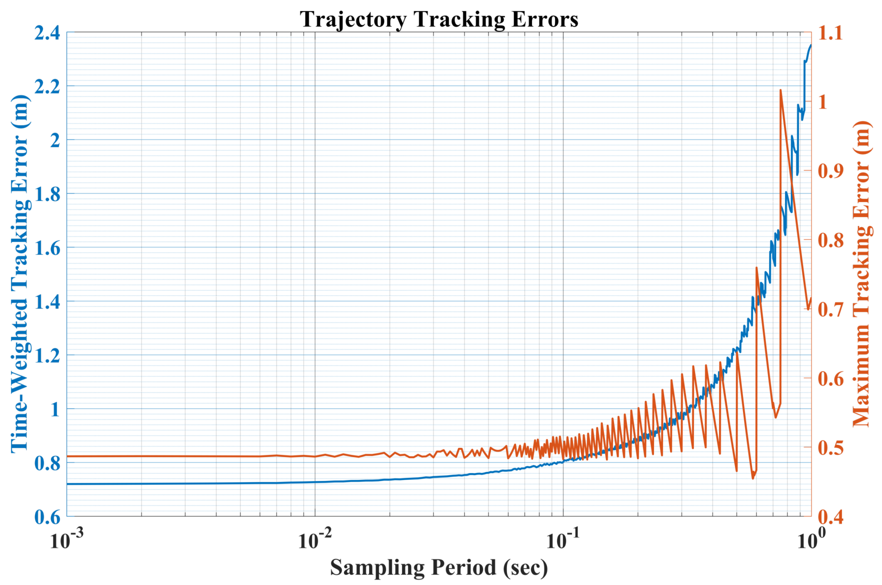


Figure 3.6: The change in average trajectory tracking error and the maximum deviation from the trajectory in Figure 3.3 as the sampling period changes.

leg from the origin to a waypoint (i.e. a step response), as a function of sampling period. In Figure 3.6 we again show the trend in these two metrics, but this time for the entire trajectory shown in Figure 3.3. While the tracking errors for a step response follow a smooth trend as sampling period varies, the tracking errors for the trajectory (and similarly

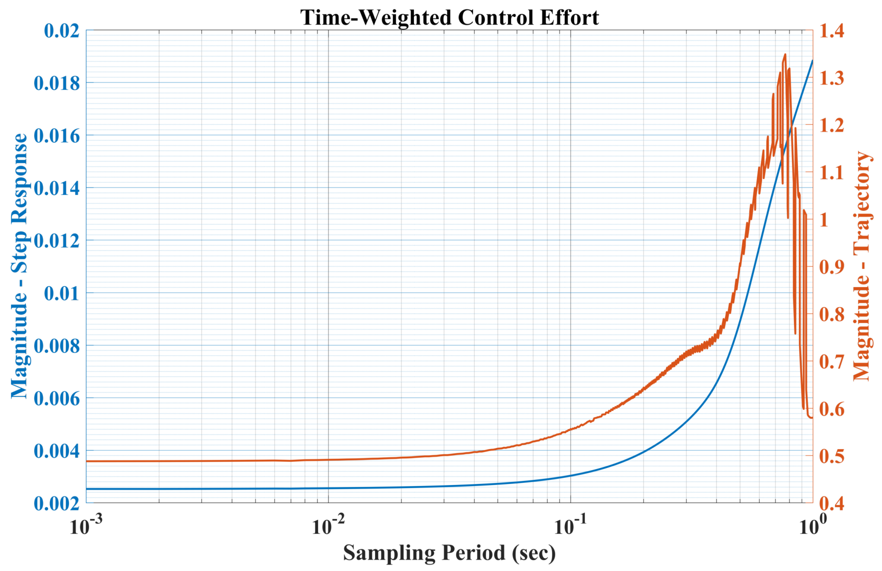


Figure 3.7: The increase in time-weighted average control effort metric against increasing sampling periods for a step response and for the trajectory in Figure 3.3.

control effort) do not. We speculate this is a result of internal resolution changes and numerical error in MATLAB's `ode45` solver coupled with effects previously discussed in regard to path geometry in which the vehicle may find itself poised differently in regards to reaching a newly generated target waypoint.

3.6.3.3 Control Effort

Using our control effort metric, CE (Equation (3.8)), as the sampling period increases, we expect the value of CE to increase since the controller will operate at a lower gain, but for a longer amount of time. However, it is important to note that the magnitude of the maximum control input generated by the controller will now be smaller as in Figure 3.2b.

We accumulate the time-weighted control effort expended over a given mission for various sampling rates and plot the trend, in Figure 3.7, against the sampling period for the controller as it drives the vehicle to a stable hover at a single waypoint (blue plot) and also as it drives the vehicle through the entire trajectory (red plot).

3.7 Discussion of Results

To develop a cyber-physical UAS, we must consider both limitations in cyber resources and performance expectations of the physical system. Our results capture this trade-off and imply that control analysis must go beyond traditional controller performance assessment and include trajectory following performance in order to trade off resources at each level of the autonomy architecture.

Intuitively, a controller designed to operate at a higher sampling period may cause undesired overshoots in the system state because the dynamics of the system act faster than appropriate control signals are generated. Additionally, the control effort (CE metric) increases, implying the system may need to spend more energy over a longer period of time, though with smaller power requirements. The benefit, however, is the increased availability of computing resources for other tasks (vision, data collection, sensing, etc.).

Choosing lower sampling periods allows for the selection of a higher-gain controller resulting in increased precision and the ability to conduct more aggressive maneuvers. Unfortunately, this trade-off results in large control inputs which may adversely affect mechanical actuators. For the cyber system, a smaller sampling period adversely affects the schedulability of additional tasks that the system must perform, particularly aperiodic tasks which are often scheduled in available slack time in the cyber system [11].

However, from the above results, the state error and control effort of the system follow a relatively flat curve as sampling period increases up to a range of approximately 0.02 s - 0.1 s. Therefore, in our idealized, no-noise simulation environment, the sampling rate of control can be lowered to this range without incurring a significant cost in state error or control effort. This illustrates the opportunity for savings in cyber resources while sampling at 0.02 s versus 0.002 s provided we can design appropriate controllers that are robust to noise and disturbances.

	Circle						Square						Spiral					
Rate(Hz)	1k	500	100	50	10	2	1k	500	100	50	10	2	1k	500	100	50	10	2
CSE	0.681	0.682	0.686	0.691	0.734	1.057	0.517	0.517	0.520	0.523	0.554	0.827	3.582	3.590	3.608	3.632	3.850	5.433
MSE	0.375	0.375	0.374	0.374	0.373	0.390	0.310	0.310	0.310	0.309	0.303	0.296	0.985	0.985	0.985	0.985	0.985	0.985
CE	0.0021	0.0021	0.0021	0.0022	0.0027	0.0203	0.0247	0.0247	0.0251	0.0255	0.0309	0.2368	0.1200	0.1201	0.1204	0.1216	0.1548	0.1570

Table 3.3: Table summarizing the trend in the proposed metrics for different trajectories.

Table 3.3 summarizes the trend in the above metrics for select sampling periods as the vehicle moves along several more complex trajectories. We consider three additional trajectories, a circle, a square, and a spiral, and compute the same metrics across the entire mission. The circular trajectory consists of eight equally spaced (in space and time) reference points about the origin at a radius of one and height of 0.5. Similarly the spiral trajectory is centered about the origin with a radius of one and an increasing height of 0.5 every second. The spiral consists of twenty-one equally spaced waypoints. The square trajectory is defined by four sides of length two at a height of two. All simulations begin with the vehicle initialized at the origin. In the cases of the circle and the spiral, the first waypoint is located at $\boldsymbol{p}_1 = (0, 1, 0)$, and in the case of the square the first waypoint is $\boldsymbol{p}_1 = (0, 0, 2)$. The square is defined in the first quadrant with the first side along the x-axis and the last side along the y-axis. Once again, large changes in error do not occur until the sampling rate reaches a range of approximately 0.02 s - 0.1 s. Also note that the values calculated for each of these evaluation metrics depend largely on the geometry of the trajectory and how it is defined. That is, following a more complex trajectory, or one defined by a higher number of waypoints, especially those consisting of smooth curves, may result in unique results. This suggests the importance of a high level CPS trajectory planner and CPS controller that is able to dynamically adjust resources, thus enabling higher precision following of complex trajectories and reducing resources for following simpler ones.

3.7.1 Utilization Metric

In a UAS there exists a set of computational tasks to which a scheduler must allocate appropriate resources to ensure computing deadlines are met. This task set may contain tasks with non-deterministic execution times, varying logical priority, sporadic, aperiodic, and other periodic tasks [11]. For example, a UAS executing a camera-based surveillance mission might have computationally intensive vision processing algorithms, guidance and navigation tasks, and a top-level planner in addition to the on-board state-estimation, sensor fusion, and attitude stabilization algorithms. To complicate this further, there may also be aperiodic tasks with quick deadlines that are triggered by a user input from a ground station. In such a scenario, it is critical to ensure that task priorities and deadlines in the real-time schedule be set correctly and perform predictably, but it is also an opportunity to dynamically adjust task priorities and deadlines depending on the environment, system performance, and mission context.

In this context, it is useful to examine resource utilization of the control task in the real-time system as a metric for cyber performance analysis. The utilization of the i^{th} real-time task is computed as $util_i = e_i/p_i$, where e_i and p_i are execution time and the period of the task [11]. The total resource utilization is then the summation over all tasks. Since e_i is difficult to know beforehand, it is usually substituted with the worst-case execution time (WCET) [77]. Given two processes with similar CPU requirements, the one with a larger period will have smaller CPU utilization. This drives efforts towards developing on-demand controllers that can guarantee performance even at higher sampling periods. This frees up cyber resources, which the scheduler might allocate to other tasks in the system. As an example, in a hover, the likelihood of running into a stationary object is low. This may be an opportunity to turn off a laser scanner and reduce the priority and task period of the corresponding sensor task, thus freeing up cyber resources for communicating collected

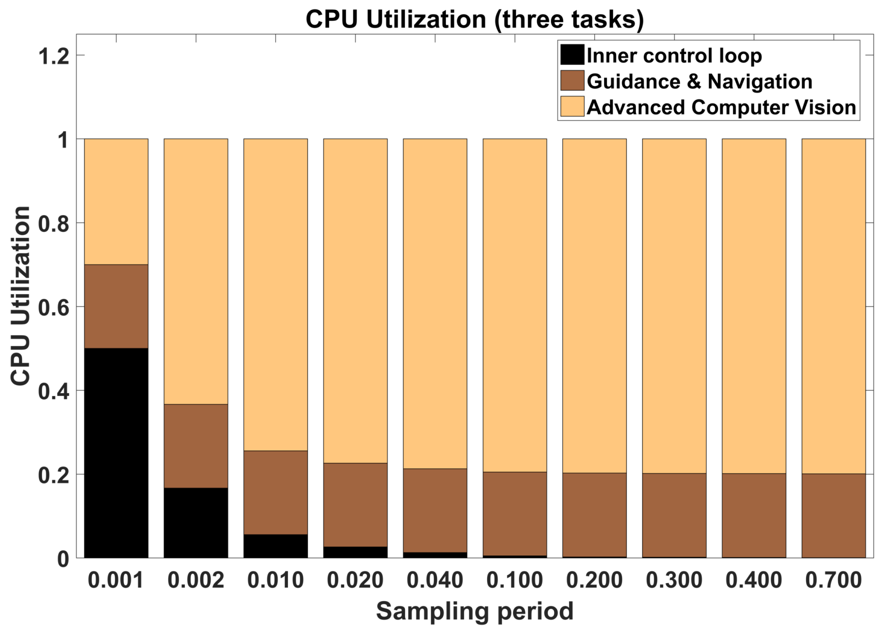


Figure 3.8: CPU utilization and availability for different tasks, assuming that the attitude controller has a worst case execution time of 0.5 ms.

data to a ground station. Figure 3.8 shows the decrease in CPU utilization for the attitude control loop of our UAS as sampling period increases, thereby accommodating other tasks which may have larger WCETs.

Chapter 4

Cyber-Physical UAS Co-regulation

With a better understanding of the affects of sampling rate in discrete LQR control of a quadrotor UAS in regard to trajectory following performance, we may now better formulate and analyze our new methods of CPS co-regulation[1]. In this chapter we formulate our cyber-physical co-regulation technique and compare its performance through CPS metrics to a discrete LQR control scheme in order to illustrate the potential for savings of cyber and physical resources through dynamic allocation versus a fixed rate control design scheme.

4.1 CPS Model and Control

We employ two types of controllers for our simulation experiments. The first is a fixed-rate discrete linear quadratic regulator (DLQR) which we will use as the baseline for comparisons against our co-regulation strategy. After discretizing the system, DLQR controller can then be designed by choosing appropriate $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrices [18]. In this type of control design a controller is typically designed assuming timing guarantees will be met by

[1]The work presented in this chapter as well as portions of Chapters 1 and 2 give rise to a paper in preparation titled *Co-Regulation of Computational and Physical Effectors in a Quadrotor UAS* and co-authored by Seth Doebbeling and Justin Bradley.

the computer system. In the real-time system a well designed control task will read sensor values, compute a control input, and send the output to actuators quickly and with minimal delay. In practice, however, particularly at slower sampling rates, due to preemption and nondeterminism in task executions there can be significant delay between each phase of the control task resulting in stale data, or irregular control inputs to actuators [78, 3]. To mitigate these effects, control designers select a sampling rate much faster than the system dynamics and rely on oversampling and time redundancy to improve safety margins. In co-regulation, however, we are exploring the low end of the sampling rate spectrum to conserve these wasted resourced, and hence, seek a discrete-time-varying control strategy.

This leads to the second type of controller which we will use in our co-regulation strategy. Since we will change the sampling rate dynamically at discrete intervals in response to system performance we need a discrete-time-varying control strategy [79]. In this case the system matrices $\mathbf{\Phi}_p$ and $\mathbf{\Gamma}_p$ formulated in Chapter 3 become functions of the time step k as they must be recalculated as the sampling rate varies. Therefore, Equation (3.4) becomes

$$X_{p,aug}[k+1] = \mathbf{\Phi}_p[k]X_{p,aug}[k] + \mathbf{\Gamma}_p[k]U_p[k] + \boldsymbol{B}_{p,r}\tilde{X}_{p,ref}.$$

We employ an emerging class of controllers, a forward-propagation discrete Riccati-based (FPRB) controller. These controllers are built upon the same optimal control foundation as other algebraic Riccati equation (ARE) based controllers (e.g. LQR) but rather than propagating the ARE backward in time, or finding a steady state solution to the ARE, it is propagated forward in time. Although research is still needed to provide performance guarantees for forward-propagation techniques we have found success with it in other systems [3, 80].

Although other standard control techniques are often implemented on quadrotor UAS, such as PID, we find the optimization characteristics of the FPRB important for co-regulation purposes. That is, FPRB is rooted in LQR optimal control and seeks to minimize the cost of state error and control input to the system–an objective of co-regulation. This complements parallel work in which we are developing an architecture for planning, guidance, and control wherein resource allocation is determined by trade-offs represented by cost metrics. LQR is, in a sense, sensitive to the energy used to generate large inputs for the system and regulates this cost to compute an optimal controller gain. Although some level of aggressive control capability tends to be lost in LQR as opposed to PID, an LQR provides improved stability guarantees [81]. This realization is beneficial to a co-regulation framework in which margins of stability may be approached.

4.1.1 Computational Model and Control

For co-regulation we model the computational system as a set of task execution rates (inverse of traditional task period) of mission critical tasks. In a complete co-regulation framework,

$$\dot{X}_c = \boldsymbol{A}_c X_c + \boldsymbol{B}_c U_c$$

would consist of task rates for the complete set of mission critical tasks (e.g. navigation, image processing, communication, control, sensing, planning, etc.) and each of these would be co-regulated alongside and in response to system performance. In this paper we focus on just the control task sampling rate and model it as we have in other work [3] where the computational system consists of a single state, the sampling rate x_c of the control task, and a single input u_c modeled as

$$\dot{x}_c = u_c.$$

A second controller is now needed to calculate the computational control input u_c, which adjusts the sampling rate, in real time, as the dynamics of the system change. This control law consists of two components. The first component scales the error between the current sampling rate and a desired reference rate. This has the effect of pushing the sampling rate toward the desired reference rate. The second component scales the difference between the current physical state to the reference state. This pushes the sampling rate toward a faster rate to provide better control authority when needed. The computational control law is represented as

$$u_c = \boldsymbol{k}_{cp}(X_p - X_{p,ref}) - k_c(x_c - x_{c,ref})$$

where $X_{p,ref}$ is a reference vector containing the three components of $\tilde{X}_{p,ref}$ and twelve zeros and x_c is the current sampling rate. The coupling gain, $\boldsymbol{k}_{c,p}$, is used to increase the sampling rate of the system in response to physical state error. The gain, k_c, drives the system toward the desired reference sampling rate $x_{c,ref}$.

The full CPS co-regulatiom model can now be realized by augmenting the physical state-space control model of the quadrotor UAS with the state-space model of the computational control task. This results in the combined model

$$\begin{bmatrix} \dot{X}_p \\ \dot{X}_c \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_p & \mathbf{0} \\ \mathbf{0} & \boldsymbol{A}_c \end{bmatrix} \begin{bmatrix} X_p \\ X_c \end{bmatrix} + \begin{bmatrix} \boldsymbol{B}_p & \mathbf{0} \\ \mathbf{0} & \boldsymbol{B}_c \end{bmatrix} \begin{bmatrix} U_p\left(X_p, U_c\right) \\ U_c\left(X_p, X_c\right) \end{bmatrix}. \tag{4.1}$$

where subscripts p and c represent “physical” and “computational” respectively. Again, by “physical” we mean the vehicle occupying space, i.e. its sensors, actuators, and dynamics of motion. By “computational” we mean the the algorithms and tasks executed by the computer. The model is also represented in block diagram form in Figure 4.1. Because our co-regulation strategy utilizes feedback and linear control, we get fast, low complexity, reactive control to respond to disturbances and noise where other related methods cannot. Additionally, our formulation allows a designer to leverage the rich theory of linear state-space control in the design of a more holistic system controller. Because our method is a hybrid Riemannian/Lebesgue method, we reap the benefits of both. In our formulation, X_c is the time-varying sampling rate of the physical UAS control task giving us the benefit

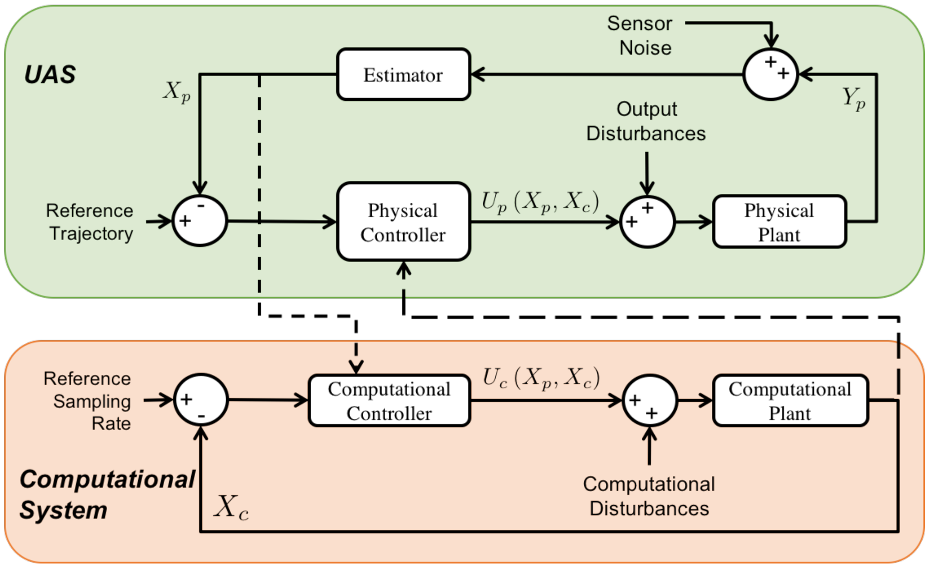


Figure 4.1: Co-Regulation Block Diagram

of retaining a periodic control task and thereby leveraging traditional real-time scheduling algorithms. However, because that sampling rate can change periodically, our sampling instants are not equidistant, and therefore, will save computation where appropriate.

4.2 Evaluation Metrics

4.2.1 State Error Metrics

To quantitatively evaluate the performance of the proposed co-regulation control strategy, we introduce several evaluation metrics. We begin with a metric set forth in our previous work [3] and outlined in Section 3.5, namely, the discrete time weighted average of the square of the control input (i.e. the control effort (CE)). On the computational side, we also reuse the time averaged percent of maximum sampling rate (i.e. the computational rate metric (CR)), defined in [3] as

$$\mathrm{CR} = \frac{1}{t_{tot}} \sum_{i=0}^{n} t_i \, x_{c,i} = \frac{n(n+1)}{2 t_{tot} x_{c,max}} \tag{4.2}$$

where t_{tot} is the total simulation time (in seconds) and $x_{c,max}$ is the maximum allowable sampling rate set for the controller. We use a maximum allowable rate of 1 kHz in our simulation experiments as this is the sampling rate of our commercial platform [13].

We also make use of the time averaged square of physical state error, i.e. the physical state error (PSE), in [3]

$$\text{PSE} = \left|\left| \left[\frac{1}{t_{tot} X^2_{p,j,max}} \sum_{i=0}^{n} \left(X_{p,j}(t_i) - X_{p,j,ref}(t_i) \right)^2 \ t_i \right] \right|\right| . \tag{4.3}$$

where $j \in [1, m]$ and m is the number of states in X_p (excluding integrator states) and the physical state error is the norm of a dimensionless vector consisting of the normalized, time-weighted average of each state error. Note that here we examine the entire state, in contrast to the cumulative state error (CSE) defined in the previous chapter (Section 3.5) where only the positional states were considered. A complementary metric for measuring trajectory-following error (TSE) is also defined in which we consider only the positional states $\tilde{X}_p = (x, y, z)^T$ and examine their deviation from the line connecting successive reference way-points, $\boldsymbol{l} = \tilde{X}_{p,ref,next}(t_i) - \tilde{X}_{p,ref,prev}(t_i)$.

$$\text{TE} = \frac{1}{t_{tot}} \sum_{i=0}^{n} \frac{||\boldsymbol{l} \times \left(\tilde{X}_{p,ref,prev}(t_i) - \tilde{X}_p(t_i) \right) ||}{||\boldsymbol{l}||} \ t_i . \tag{4.4}$$

This metric assumes the ideal path is one defined by a series of straight lines between successive way-points. This metric gives a measure of how close to an ideal straight line path the control strategy can provide.

We also reuse the maximum state error (MSE) defined in Section 3.5 to capture the maximum deviation from the ideal trajectory. These metrics are listed in Table 4.1.

Name	Abbreviation	Description
Control Rate Metric	CR	*time averaged percent of maximum sampling rate*
Physical State Error	PSE	*norm of time averaged normalized state error*
Tracking Error	TE	*time averaged deviation from straight line path*
Maximum State Error	MSE	*maximum deviation from straight line path*
Control Effort	CE	*time weighted squared average of control input*

Table 4.1: CPS co-regulation metrics

4.2.2 Power and Energy Estimates

By examining the mechanics of a single stationary rotor system consisting of a motor and propeller we can calculate an approximation of the power usage by a multi-rotor vehicle during operation [68]. Assuming there is no free stream movement of the surrounding air (i.e. no wind), it follows that the majority of displaced air moves with a velocity v parallel to the rotor thrust $\mathbb{T}$. Therefore, the power for a given rotor may be approximated as $P = \mathbb{T}v$.

Momentum theory tells us that for a thin actuator disk (i.e. a propeller spinning at sufficient speed) of area A pushing a fluid with density ρ, the power required to produce a given thrust $\mathbb{T}$ [82] is

$$P = \sqrt{\frac{\mathbb{T}^3}{2\rho A}}. \tag{4.5}$$

We approximate the average power usage,

$$\text{PWR} = \frac{1}{t_{tot}} \sum_{i=0}^{n} P_i \, t_i, \tag{4.6}$$

as well as the maximum power drawn from the system for propulsion,

$$\text{MXP} = \max_i \left(P_i\right), \tag{4.7}$$

as an additional means of evaluating the efficiency of our co-regulation design. These metrics are important for indicating the nature of the control effort over time and the role that plays in energy consumption and maximum power draw.

4.3 Results

4.3.1 Simulation Setup

Co-regulation, as described here, is not realizable given built-in MATLAB functionality, as functions like `c2d`, and `dlqr` require linearized models with a fixed sampling rate employing Riemannian sampling techniques, and therefore yielding a static gain. Instead, our co-regulation leverages these functions at discrete intervals when the sampling rate is updated yielding a hybrid Riemannian/Lebesgue sampling technique. That is, an initial physical control input is calculated at time $t = 0$ with an initial physical state and sampling rate, and is held constant (i.e. a zero-order hold (ZOH)) for one sampling period (T_d). After one period, a new sampling rate may be calculated via the computational control law. The new rate is then used to re-discretize the system and generate a new control input for the physical system to be held for the length of the new sampling period. Via this process, both the sampling rate and physical system gain become dynamic throughout the entire simulation. During each period, T_d, in which there is a constant control input, the non-linear dynamics from Equation (3.1) are used to simulate the motion of the UAS using `ode45`. This gives us response characteristics of the system in the transients between ZOH samples. We now reconsider the algorithm in Section 3.4 (shown here as Algorithm 2) in which at this point the sampling period T_d changes according to the cyber control law for each propagation of the discrete time-step k.

Simulations are run with the parameters listed in Table 3.1 and the same $\boldsymbol{Q}$ and $\boldsymbol{R}$ values are used in designing both the DLQR controllers and FPRB controllers used. We manually

Algorithm 2 Algorithm to simulate the control of quadrotor flight under varying discrete sampling rate.

```
Data: Nonlinear system model, $f(X)$, $sampling\_period$
// initialize system constants
$T_d \longleftarrow sampling\_period$
$lin\_model \longleftarrow$ linearize$(f(X), T_d)$
$\boldsymbol{Q} \longleftarrow 10 \cdot \mathbb{I}_{15\times 15} \cdot q$
$\boldsymbol{R} \longleftarrow 2 \cdot \mathbb{I}_{4\times 4}$
begin
    $\boldsymbol{\Phi}_{T_d}, \boldsymbol{\Gamma}_{T_d} \longleftarrow$ discretize$(lin\_model, T_d)$
    $\boldsymbol{K}_{gain} \longleftarrow$ dlqr$(\boldsymbol{\Phi}_{T_d}, \boldsymbol{\Gamma}_{T_d})$
    $k \longleftarrow 0$
    $X_{all} = [\,]$
    $X_{init} \longleftarrow$ initial_state()
    while $kT_d \leq simulation\_length$ do
        $U_k \longleftarrow$ input_vector$(X_{init}, \boldsymbol{K}_{gain})$
        $[X_1 \ldots X_m]$ = ode45$(f(X), T_d, U_k\ X_{init})$                // Simulate
        $X_{all} = [X_{all}; X_1 \ldots X_m]$
        $k \longleftarrow k + 1$                  // propagate discrete time step
        $X_{init} \longleftarrow X_m$                         // new initial state
    end
end
```

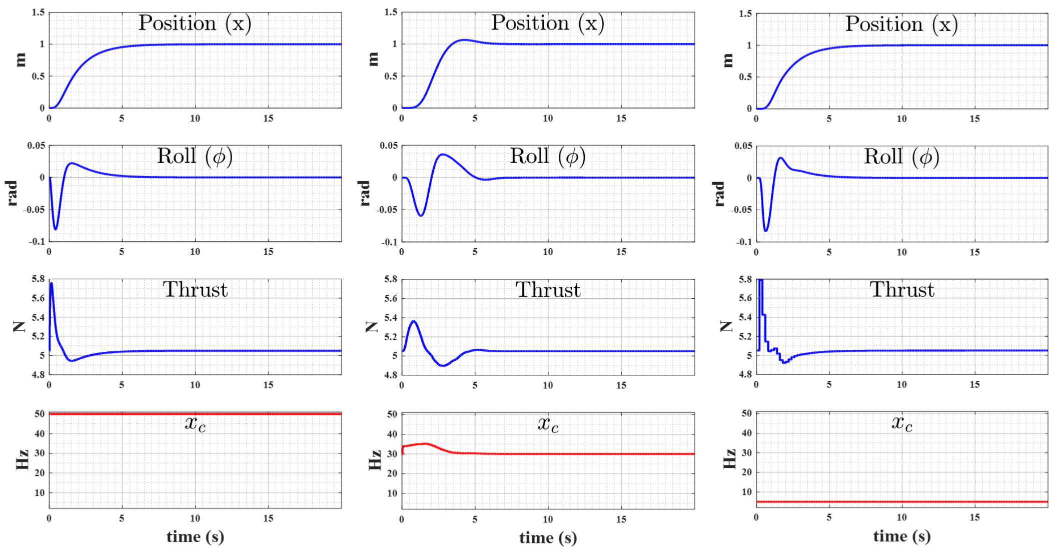
tune the computation control gain $\boldsymbol{k}_{cp}$ and k_c heuristically through visual inspection of step response characteristics such as rise time and settling time for several simulations using a broad range of values. The gain values used in our experiments are

$$\boldsymbol{k}_{cp} = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0]$$

$$k_c = 0.75\ .$$

4.3.2 Results

Variations in the system state, inputs, and sampling rate are shown for a step response in Figure 4.2. There we show a comparison of a traditional DLQR controller employing fixed sampling rates of 50 Hz and 5 Hz, as well as the proposed co-regulation at a reference sampling rate of 30 Hz. We select 50 Hz and 5 Hz for DLQR control as the former is considered sufficient for positional control (30 Hz being a typical control rate in the NIMBUS Lab) and the latter a minimum for reliable control.



(a) Traditional DLQR Control @ 50 Hz (b) Co-Regulation @ $x_{c,0}, x_{c,r}$ = 30 Hz (c) Traditional DLQR Control @ 5 Hz

Figure 4.2: The quadrotor's physical performance under co-regulation suffers an approximate 1 percent overshoot in position but generates control inputs which are significantly smaller in magnitude than the traditional DLQR controller.

The step responses in Figure 4.2 show that significant computational resources can be saved with nominal loss of physical performance using co-regulation (Figure 4.2b. That is, the system can operate at a significantly lower rate using smaller inputs over a longer period of time, while incurring a small (approximately 1%) overshoot in position.

However, in the context of UAS it is much more valuable to examine the effects of co-regulation as the UAS executes a trajectory. Because we seek to conserve as many computational resources as possible, for use by other processes, throughout the entire course of a mission, we expand our controller analysis beyond a traditional step response and apply the above co-regulation techniques and evaluation metrics to more advanced trajectories.

Figure 4.3 shows the UAS executing a commanded trajectory under traditional DLQR

	Step							Trajectory						
Control Strategy	**TE**	**MSE**	**PSE**	**CE**	**CR**	**PWR**	**MXP**	**TE**	**MSE**	**PSE**	**CE**	**CR**	**PWR**	**MXP**
DLQR @ 1 kHz	1.0000	1.0000	4.1315	1.4468	1.000	1.0001	1.1184	1.0000	1.0000	1.6058	1.0000	1.0000	1.0000	1.0000
DLQR @ 50 Hz	1.0288	1.0282	4.0527	1.4468	0.0500	1.0002	1.1216	2.0346	2.0276	1.3732	4.4964	0.0500	1.0233	1.2035
Co-Reg @ 30 Hz	1.7404	1.8415	1.0000	1.0000	0.0307	1.0009	1.0036	3.1404	1.4259	1.0058	5.4015	0.0340	1.0415	1.0777
Co-Reg @ 15 Hz	1.9038	1.8863	1.0238	1.0213	0.0157	1.0012	1.0000	4.6199	2.2813	1.0000	9.8102	0.0194	1.0638	1.1762
DLQR @ 5 Hz	1.2404	1.2742	4.0582	1.6383	0.0050	1.0000	1.1477	2.3715	2.3130	1.4728	5.6715	0.0050	1.2089	1.2445

Table 4.2: Evaluation metrics of co-regulation and DLQR at different rates for step response and trajectory following.

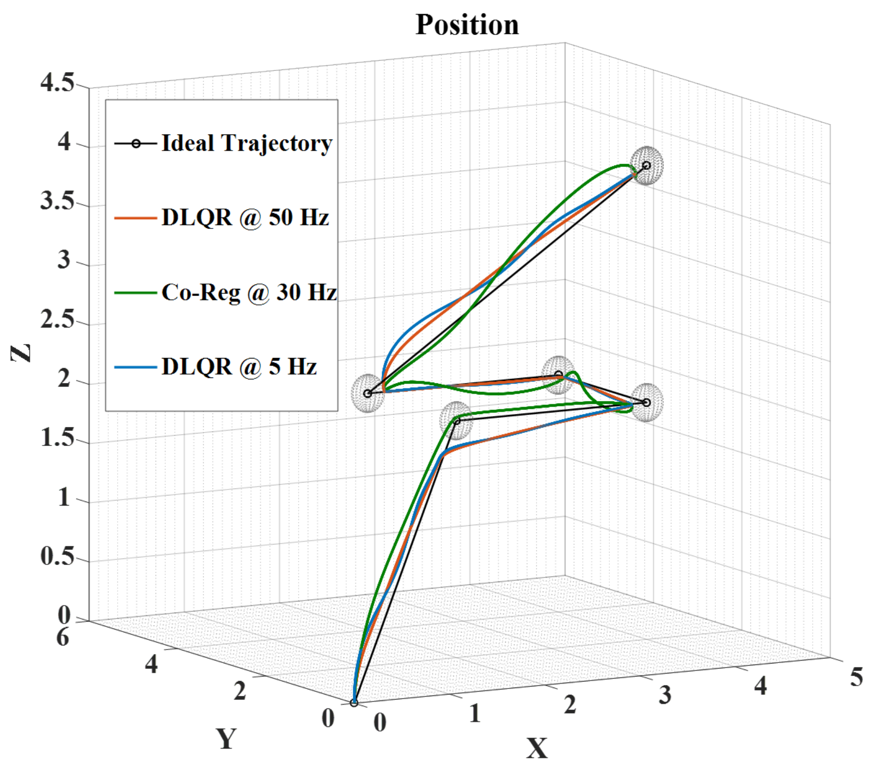


Figure 4.3: Position of traditional DLQR response at 50Hz and 5Hz in comparison to co-regulation at $x_{c,ref}$= 30Hz

control at 50 Hz, and 5 Hz, in comparison with co-regulation using a reference sampling rate of 30 Hz. Table 4.2 illustrates the trends in the metrics described in Section 4.2 for both step response and the trajectory in Figure 4.3 simulated under different DLQR and co-regulation controllers.

For comparison we select frequencies at both extremes of the sampling rate spectrum as well as intermediate frequencies that represent more typical implementations. We are most interested in frequencies for which the quadrotor system approaches lower bounds on stability as this is where the largest trade-offs in physical and computational resources are likely to occur, but also where minimal control performance can be anticipated. The difference in trends in TE and PSE for both the step response and trajectory data convey an interesting characteristic of our co-regulation framework. DLQR controllers more aggressively apply control inputs (larger inputs for shorter amount of time) in comparison to our co-regulation technique. As a result, the DLQR controller is able to retain altitude by quickly applying large control inputs, whereas under co-regulation and longer, smaller control inputs, the UAS experiences a brief loss in altitude when traversing laterally. This allows DLQR to more closely follow a strait line trajectory, as indicated by the TE metric. In contrast, by applying inputs for a longer period of time, the UAS under co-regulation builds more momentum resulting in a faster rise time and under-damped system characteristics, whereas under DLQR control the system exhibits an over-damped response causing it to slow much more drastically as it approaches a target waypoint. This may be desired in a single step response, but for successive trajectory legs this behavior results in a larger PSE as the DLQR control departs from the first trajectory leg as a new reference command is made before reaching the first target waypoint. This implies that co-regulation is better able to minimize state error but less capable of making more precise movements. The amount of control effort required for a step response highlights our co-regulation strategy but proves less promising in terms of trajectory execution. However, it is intuitive that as trajectories become more complex and sampling rate minimums are approached more control authority will be required to maintain stable flight and trajectory tracking. Computational savings are best illustrated in Table 4.2 by the CR metric and MXP. In all cases as sampling rate decreases the savings in CR increase, and as we switch from DLQR to co-regulation maximum power requirements decrease.

These results highlight the complex trade-offs between high and low sampling rates and demonstrate the performance of a strategy that tries to take advantage of both. Co-regulation utilizes significantly fewer computational resources compared with high-rate DLQR control, and much lower state error compared with low-rate DLQR control. On the downside, trajectory following is less robust for co-regulation. This suggests that future work is needed to design the physical FPRB controller and the computational controller to improve performance.

We point out that in our simulations, controllers with sampling rates greater than 50 Hz yield results with negligible performance improvements, as shown in Chapter 3. This illustrates the strong need for an analysis of trajectory-following performance for any controller, particularly if a high sampling rate is assumed to be better without a thorough analysis. It also indicates the computational savings that could be realized by employing a Lebesgue or hybrid Riemannian/Lebesgue control strategy such as co-regulation.

Finally, examining the effects of co-regulation over trajectories is crucial as it informs the generation of more efficient trajectories by identifying maneuvers or paths which require fewer resources to complete. A co-regulation strategy complete with both a physical and computational trajectory planner would trade off prioritizing physical performance and computational performance as the mission requires. It could do this by setting target way-points appropriately as well as associating target sampling rates for different mission segments. The result would be a high-level planner that could optimally allocate physical and computational resources as a plan that would be executed by a low-level reactive co-regulation layer that leverages the advantages of feedback to improve robustness.

Chapter 5

Toward Cyber-Physical UAS Trajectory Generation

5.1 Introduction

Although set-point or point-to-point trajectory tracking may be a sufficient method in some quadrotor UAS missions, a quadrotor cannot perfectly execute a series of piecewise straight lines connecting successive waypoints. Therefore, in order to better evaluate the physical performance of our rate varying control (and others for that matter) a higher level motion planning scheme which can generate a smooth path trajectory based off of physical limitations of the system. Consequentially, as our co-regulation technique showed better results for reducing state error than cross track error we may seen improvements when evaluating physical performance with respect to a feasible trajectory. Higher level motion planning will also allow for time constraint to be enforced more effectively. That is, instead of providing the next target waypoint to the controller at arbitrary times, whether the vehicle has reached the current point or not, a trajectory generated with smooth functions of position and velocity can provide better timing guarantees on passing through successive waypoints. Finally, in the experiments conducted above, the cyber control law is always driven to a constant minimum reference when physical state error is low. Ideally a higher level motion planner would associate a potentially different target sampling rate to each generated trajectory point to optimize the allocation of both cyber and physical resources.

5.2 Background

Terminology in varying robotics communities used to describe how a robot moves through space may vary slightly. Figure 5.1 provides a hierarchical view of the motion planning structure as defined in this work. At some level a mission is defined as a series of tasks for the robot to complete in which failure or success may be defined. Autonomous robots must generate a series of sequential actions to execute mission tasks [83]. A motion planning architecture, along with environmental and dynamic system models, provides deliberation as to how the robot should traverse a given space. Paths may be calculated by search algorithms such as A* [83][38], deterministic or stochastic processes [84], or some other heuristic. For each mission task, a path, consisting of some combination of curves and waypoints, is generated which, if followed to some degree of accuracy, will result in successful execution. A trajectory differs from a path in that it must also include a dimension in time. That is, a path, or portion thereof, must also include constraints on execution time, velocity, etc. This is achieved via a model of system dynamics and kinematics which describes how the system reacts to actuation with respect to space and time (i.e. the physics of the system).

For a UAS, nonlinear three dimensional dynamics as well as performance and computational limitations make guidance non-trivial, and environmental disturbances along with feedback uncertainty can result in poor execution of even sophisticated planning strategies. Consequentially, waypoint path following is often implemented to ensure the UAS remains

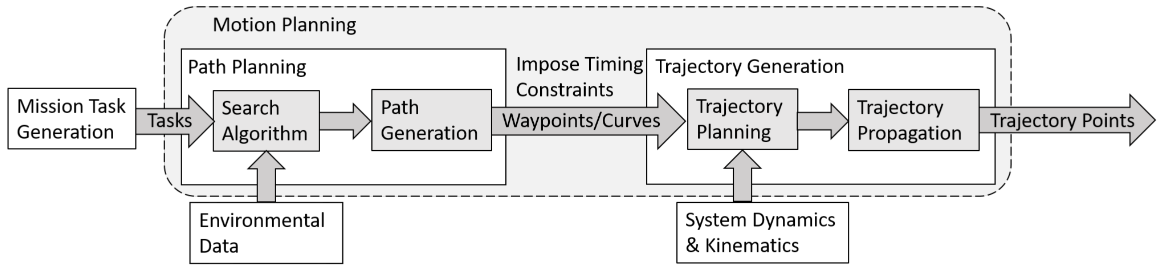


Figure 5.1: Motion planning overview

on the desired path with no guarantees on timing [9]. Waypoint tracking coupled with high gain, high sampling rate control may then be used to achieve execution of a path with loose, sub-optimal timing guarantees. According to a survey by Goerzen et al. [85], UAS are typically modeled in three dimensional space with environmental and feedback sensor disturbances and are subject to speed and acceleration constraints or in more complex cases, typically involving aggressive maneuvers [54, 53] or nonlinear aerodynamic effects [86, 68], higher order constraints derived from derivatives of the equations of motion.

The latter may quickly become computationally expensive given higher order optimization functions and nonlinear dynamics and control, whereas kinematic approaches with lower order dynamic models and constraints that are more computationally conservative, may, even in the absence of external disturbances, result in poor performance as discontinuities in higher order dynamics are ignored. It follows that in a cyber-physical system (CPS) control architecture which seeks to trade off computational and physical resources depending on mission objectives, motion planning algorithms must be carefully chosen to complement that strategy. A cyber-physical trajectory generator would be responsible for allocating cyber and physical resources at a low level while being computationally efficient and sufficiently precise in generating the standard for physical performance evaluation.

5.3 Method

Traditional guidance navigation and control for UAS prioritize path following over time dependent trajectory tracking for several reasons. Environmental disturbances and modeling approximations result in large uncertainties in defining timing guarantees [9], plentiful computational resources provide for high gain, high rate control, and sequential mission planning provides deliberative waypoint tracking which yields satisfactory timing in common applications. However, at low, varying rate control, careful consideration of where to move as well as how to move must be taken to effectively manage physical and computational resources. Two controllers tasked with converging to the same value in the same amount of time while operating at two different control rates will generate unique system inputs consuming differing amounts of computational and physical resources. In our varying rate co-regulation technique we wish to directly manipulate the differences in these generated inputs so as to efficiently manage resources.

For our CPS control architecture we desire a trajectory generation technique which can generate trajectories with bounds on control effort and sufficient physical performance for execution by a discrete, varying sampling rate controller. We desire a trajectory generator which can calculate the necessary physical resources required for execution while balancing computational resources required for generating control inputs sufficient for cross-tracking requirements. Therefore, an aggressive or computationally expensive trajectory generator is undesirable as it may require a large amount of cyber resources in and of itself. We also desire a generator which produces points at discrete time steps corresponding to the sampling rate of control so as to calculate necessary physical inputs and choose the most effective time-step interval and physical input combination to continue the trajectory. Still, the generated trajectory must be feasible given the dynamic constraints of the system. We expect the UAS to intercept intermediate waypoints without stopping, and as a result, the trajectory generator must produce a continuous function in position. Therefore, some type of smoothing function for straight line transitions must be implemented. In evaluating trajectory following performance it is expected that this will prevent excessive punishment in such transitions. Here, as we intend to trade off computational efficiency with precision cross-tracking, we prefer a computationally inexpensive, real-time trajectory generation technique which provides a continuous function the vehicle may be able to execute (i.e. within cross-tracking bounds) based on a series of discretely spaced trajectory points. Therefore, motion planning algorithms with higher order polynomials are feasible so long as saturation is not reached. It follows that, given typically high thrust to weight ratios and very agile rotational dynamics due to rotor geometry and low moments of inertia (i.e. capability for high angular acceleration), one can assume that roll and pitch dynamics can tolerate large amounts of jerk, and may be neglected in trajectory generation so long as saturation is not reached [56]. Note this assumption comes with the caveat that rotational dynamics in yaw are significantly slower but do not have a large influence on trajectory tracking so long as heading is not specified or is held constant. However, this kinematic approach does not provide for more aggressive maneuvers or precise execution. We adhere to the assumptions made in [56] in which constraints on maximum velocity and acceleration ensure the prevention of saturation. As a result a numerical kinematic based approach in $\mathbb{R}^3$ using linear segments with parabolic blends (outlined in one dimension in [44]) is implemented to generate a continuous trajectory. We begin by defining the maximum velocity and accelerations attainable by the vehicle based on system specifications. The maximum velocity is set to $3\,{}^{\mathrm{m}}\!/\!_{\mathrm{s}}$ as specified in documentation and maximum accelerations set based on maximum thrust limitations [59]. Deliberate path planning is considered to be done a priori by the user or some higher level path planner which specifies a series of waypoints outlining an obstacle free piecewise straight-line path.

5.3.1 Trajectory Planning

Once the performance limitations have been set, the trajectory generator takes in the pre-specified waypoints, and the amount of time that will be spent traversing each straight line between consecutive waypoints is calculated using either the maximum velocity and distances or some user specified time. If the user specifies a time that is not possible to execute while touching each leg of the trajectory during execution, a trajectory will be generated for the minimum amount of time in which each leg is touched. Under the assumption that the vehicle will always start from rest at the origin and end at rest, the trajectory generator calculates velocities for the linear segments and accelerations for the quadratic blend regions as well as corresponding linear segment times and blend region times based on the specified or calculated leg times. These values are calculated for intermediate trajectory legs as in [44][87] with some slight modification into $\mathbb{R}^3$ as,

$$\boldsymbol{v}_j = \frac{\boldsymbol{p}_{j+1} - \boldsymbol{p}_j}{t_{leg,j}} \tag{5.1}$$

$$\boldsymbol{a}_{max,j} = sign(\boldsymbol{v}_j - \boldsymbol{v}_{j-1}) \cdot \boldsymbol{a}_{max} \tag{5.2}$$

$$t_{blend,j} = max\left(\frac{\boldsymbol{v}_j - \boldsymbol{v}_{j-1}}{\boldsymbol{a}_{max,j}}\right) \tag{5.3}$$

$$t_{linear,j} = t_{leg} - \frac{1}{2}t_{blend,j} - \frac{1}{2}t_{blend,j-1} \tag{5.4}$$

where $\boldsymbol{p}_j$ is the position of the j^{th} waypoint $\boldsymbol{v}_j$ is the velocity of the j^{th} leg, $\boldsymbol{a}_{max,j}$ is the maximum acceleration of the j^{th} blend, $t_{leg,j}$ is the j^{th} leg time, $t_{blend,j}$ is the time for the j^{th} blend and $t_{linear,j}$ is the time for the j^{th} linear segment. Note that in Equation 5.3 each component of the velocity difference is divided by its corresponding component of $\boldsymbol{a}_{max,j}$. The first and last legs of the trajectory however, must take into account the fact that the entire blend regions while starting from rest and coming to a stop reside in their respective leg times. Therefore, the values for the first and last legs are calculated as follows [44]. For the first,

$$\boldsymbol{a}_{max,1} = sign(\boldsymbol{p}_2 - \boldsymbol{p}_1) \cdot \boldsymbol{a}_{max} \tag{5.5}$$

$$t_{blend,1} = t_{leg,1} - \sqrt{t_{leg,j} - \frac{2(\boldsymbol{p}_2 - \boldsymbol{p}_1)}{a_{max,1}}} \tag{5.6}$$

$$\boldsymbol{v}_1 = \frac{\boldsymbol{p}_2 - \boldsymbol{p}_1}{t_{leg,1} - \frac{1}{2}t_{blend,1}} \tag{5.7}$$

$$t_{linear,1} = t_{leg,1} - t_{blend,1} - \frac{1}{2}t_{blend,2} \tag{5.8}$$

and for the last,

$$\boldsymbol{a}_{max,n+1} = sign(\boldsymbol{p}_{n+1} - \boldsymbol{p}_n) \cdot \boldsymbol{a}_{max} \tag{5.9}$$

$$t_{blend,n+1} = t_{leg,n} - \sqrt{t_{leg,n} - \frac{2(\boldsymbol{p}_{n+1} - \boldsymbol{p}_n)}{a_{max,n+1}}} \tag{5.10}$$

$$\boldsymbol{v}_n = \frac{\boldsymbol{p}_{n+1} - \boldsymbol{p}_n}{t_{leg,n} - \frac{1}{2}t_{blend,n+1}} \tag{5.11}$$

$$t_{linear,n} = t_{leg,n} - t_{blend,n+1} - \frac{1}{2}t_{blend,n}. \tag{5.12}$$

Where n is the total number of trajectory legs. Note that Equations 5.6 and 5.10 may result in complex numbers. Therefore, if a complex number arises, an appropriate amount of time is allocated to the first or last leg to ensure all calculated values remain real. At this point the generated trajectory will approach a waypoint on a straight-line path and curve away from the point before reaching it in order to transition to the next linear segment, essentially rounding corners. If the user prefers the trajectory to pass through the specified waypoints, a series of calculated pseudo waypoints can be placed relative to the original waypoints in such a way that the original waypoints become the point of inflection on the blend regions. This results in linear segments that differ from the straight-line trajectory defined by the original waypoints. This formulation is illustrated in Algorithm 1. Based on

Algorithm 3 Converting waypoints to through points by introducing pseudo waypoints

```
if $want\ pseudo\ points$
  $pseudo\_points \leftarrow first\_waypoint$;
    for $n = 2 : number\_waypoints$ - 1
      $pseudo\_points \leftarrow [pseudo\_points,\ waypoints(n) \pm \frac{vel_{j-1}+vel_j}{||vel_{j-1}+vel_j||} \cdot \frac{1}{2} t_{blend,j}]$;
    end
  $pseudo\_points \leftarrow last\_waypoint$;
end
```

the new, pseudo waypoints, new leg times, velocities, accelerations, linear times, and blend times are calculated using Equations 5.1-5.4. Finally the points at which blend regions begin and end are specified as,

$$\boldsymbol{p}_{blend,j} = \boldsymbol{p}_j + \boldsymbol{v}_j \cdot \frac{1}{2} t_{blend,j}$$

$$\boldsymbol{p}_{blend,j-1} = \boldsymbol{p}_j - \boldsymbol{v}_{j-1} \cdot \frac{1}{2} t_{blend,j}$$

where $\boldsymbol{p}_{blend,j}$ is the j^{th} blend point. Note that for the first and last waypoints $(\boldsymbol{p}_1,\ \boldsymbol{p}_{last})$, blend points $\boldsymbol{p}_{blend,j-1}$and $\boldsymbol{p}_{blend,j}$ are replaced by $\boldsymbol{p}_j$ and $\boldsymbol{p}_{last}$ respectively.

5.3.2 Trajectory Generation

Once velocities and accelerations are calculated for linear segments and blend regions, trajectory points are calculated using the basic kinematic equations of motion for a point mass. Note that this means the trajectory is unaware of any inertia the system may have differing from that of a point mass and that any system with obscure dynamics like that of a quadrotor can still be expected to deviate from the trajectory depending on control. During the linear segments the trajectory points are calculated with a constant velocity (i.e. zero acceleration)

$$\boldsymbol{p}_{traj} = \boldsymbol{p}_{blend,j} + \boldsymbol{v}_j \cdot t_{linear,j} \tag{5.13}$$

and in the blend region an acceleration blends the preceding velocity into the velocity of the next trajectory leg as

$$\boldsymbol{p}_{traj} = \boldsymbol{p}_{blend,j-1} + \boldsymbol{v}_{j-1} \cdot t_{blend} + \frac{1}{2}\boldsymbol{a}_{blend} \cdot t_{blend,j}^2 \tag{5.14}$$

where $\boldsymbol{a}_{blend} = \left(\frac{\boldsymbol{v}_j - \boldsymbol{v}_{j-1}}{t_{blend,j}}\right)$. If the dimensions of the original straight-line path are such that the computed velocities and accelerations along with linear and blend times are insufficient to execute the trajectory (e.g. a 'sharp' turn) the linear segment between blends may be lost (becomes negative) resulting in an overlap of adjacent blend regions. In such a case the velocity of the lost linear segment, as well as corresponding times, are adjusted to facilitate the generation of a continuous path according to,

$$t_{leg}^* = t_{leg} - t_{linear}$$

$$\boldsymbol{v}_j^* = \frac{\boldsymbol{p}_{j+1} - \boldsymbol{p}_j}{t_{leg}}$$

$$t_{linear}^* = 0$$

$$\boldsymbol{a}_{blend}^* = \left(\frac{\boldsymbol{v}_j - \boldsymbol{v}_{j-1}}{t_{blend,j}}\right)$$

$$\boldsymbol{p}_{blend,j}^* = \boldsymbol{p}_{blend,j-1} + \boldsymbol{v}_j^* \cdot t_{blend,j} + \frac{1}{2}\boldsymbol{a}_{blend}^* \cdot t_{blend}^2$$

$$\boldsymbol{p}_{blend,j+1}^* = \boldsymbol{p}_{blend,j}^*$$

where the notation * represents a corrected value.

5.4 Experimental Setup

The method outlined in the above section was implemented in a MATLAB function which takes in a series of waypoints, a time-step value for spacing between generated trajectory points, and a trajectory execution time and returns a series of generated trajectory points as well as the corresponding time and velocities associated with reaching each point. For several different trajectories, each generated under two different execution time constraints, we evaluate several characteristics of the generated trajectories in comparison to each other.

First we evaluate the deviation of the generated trajectory from the original straight-line path connecting the specified waypoints,

$$\text{cross-track error} = \frac{||(\boldsymbol{p}_{traj} - \boldsymbol{p}_j) \times (\boldsymbol{p}_{traj} - \boldsymbol{p}_{j+1})||}{||\boldsymbol{p}_{j+1} - \boldsymbol{p}_j||}$$

which we will call the cross-track error. This deviation occurs primarily in rounding corners in the straight line path, or in generating trajectories through the addition of pseudo waypoints. We also examine the ratios of time spent in both the blend regions and linear segments of the trajectories, as we expect this to be an indicator of both physical and computational performance of the eventual CPS control implementation. Correspondingly we evaluate generated velocities and accelerations for each trajectory and resulting estimates of power consumption by an ideal vehicle which could execute the generated trajectory given the enforced constraints on time, velocity, and acceleration. The power is estimated at each time-step using Equation 4.5, and the maximum and average power for each mission is recorded.

5.4.1 Experimental Results

Four test trajectories were generated with complete mission execution times of five seconds, three seconds (Figure 5.2), and a sub-optimal minimum time (Figure 5.3).

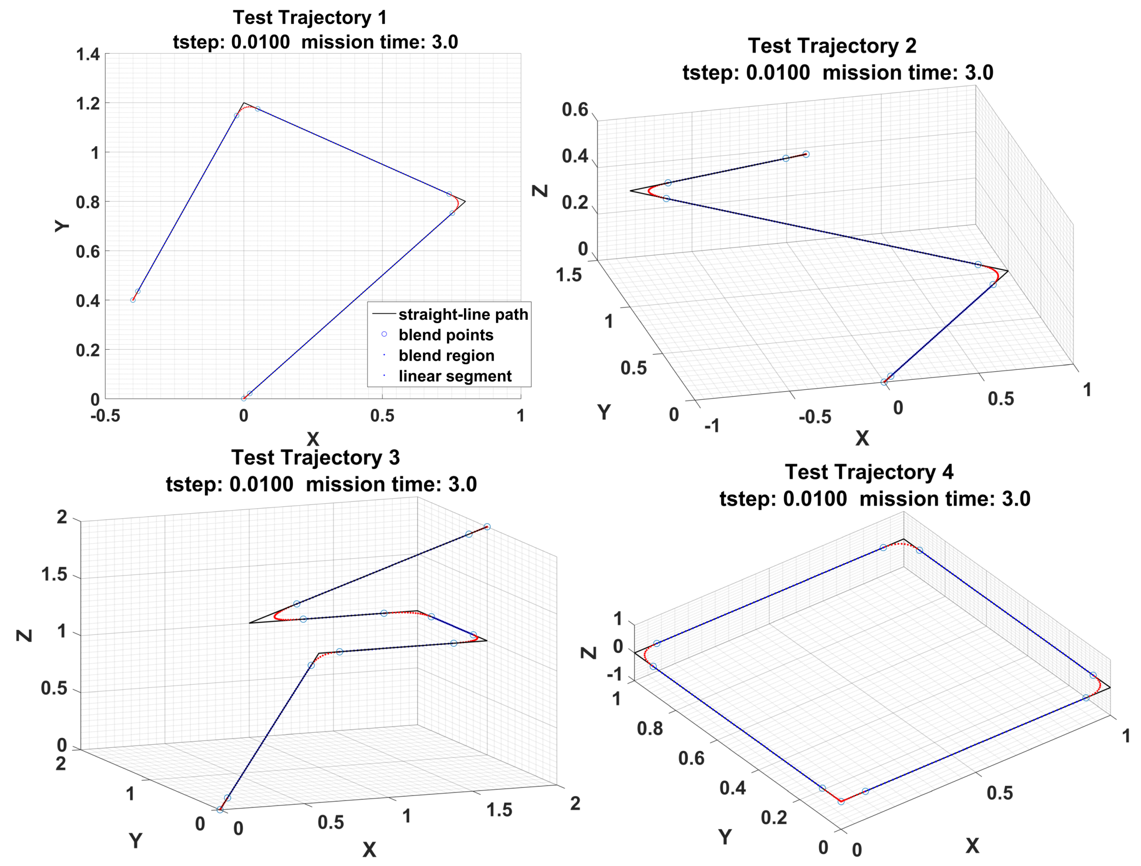


Figure 5.2: Test trajectories generated with a mission execution time of three seconds and time-step resolution of 0.01 seconds

Portions of the trajectory shown in red illustrate the blend regions for which the state of the vehicle is propagated via Equation 5.14 under a constant acceleration value, $\boldsymbol{a}_{blend}$. Portions indicated in blue represent linear segments of the trajectory connecting successive blend regions calculated via Equation 5.13 under a constant velocity. Trajectories were chosen to examine varying degree of turns and motion in three dimensions. The trajectories analyzed are believed to capture maneuvers that may be a part of a typical mission for a quadrotor UAS. Additional trajectories were tested less extensively to ensure robustness.

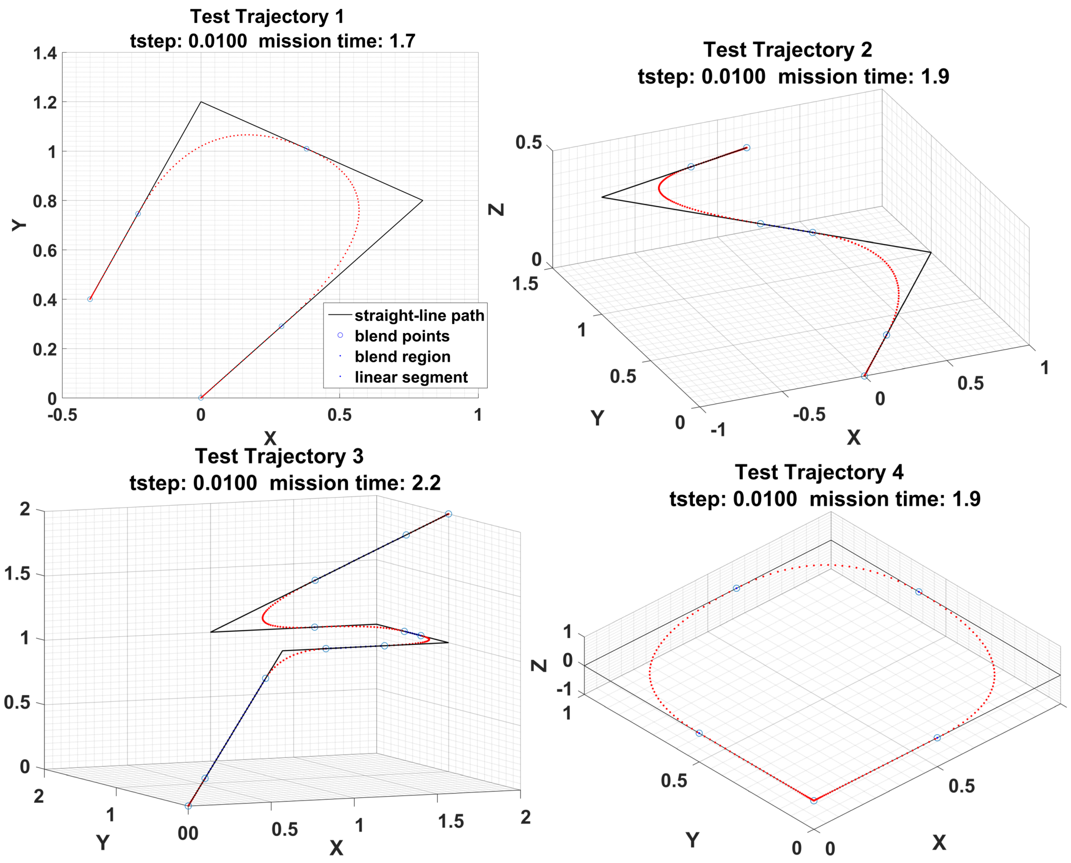


Figure 5.3: Test trajectories generated with a minimum execution time (sec) which results in a continuous path and time-step resolution of 0.01 seconds

It is also important to note that although the trajectory planner assumes an obstacle free path provided by a higher level path planner, there are scenarios where the vehicle may deviate significantly from the planned path to meet timing requirements. In these scenarios, an additional check of maximum deviation by the trajectory generator which, if violated, raises the minimum mission time by adjusting constraints on velocity and acceleration accordingly may be useful to ensure obstacles are avoided. Calculated values for cross-track error, power estimates, and timing ratios are listed in Table 5.1 for each generated trajectory.

	Trajectory 1			Trajectory 2			Trajectory 3			Trajectory 4		
Mission Time:	5sec	3sec	Min: 1.7sec	5sec	3sec	Min: 1.9sec	5sec	3sec	Min: 2.2sec	5sec	3sec	Min: 1.9sec
Cross-Track Error:	0.0064	0.0304	1.1370	0.0120	0.0699	0.9493	0.0857	0.3731	1.2050	0.0141	0.0644	1.3614
Avg. Power Est. (J/s):	86.10	94.13	109.69	90.03	105.19	137.10	102.36	135.41	184.76	91.39	107.64	138.64
Max Power Est. (J/s):	224.59	224.59	128.37	224.59	224.59	210.74	376.35	376.35	376.35	224.59	224.59	223.27
$t_{linear} : t_{blend}$	20.831	6.730	0	11.015	3.168	0.060	7.424	1.967	0.475	13.283	3.906	0
$\%t_{linear}$	95.4	87.1	0	91.7	76.0	5.7	88.1	66.3	32.2	93.0	79.6	0
$\%t_{blend}$	4.6	12.9	100	8.3	24.0	94.3	11.9	33.7	67.8	7.0	20.4	100

Table 5.1: Evaluation metrics of test trajectories generated with time-step resolution of 0.01 seconds

5.5 Discussion

From Table 5.1 we see that, for the same trajectory, as the constraint on mission time approaches the minimum feasible time, the cross-track error increases. Execution of the trajectory in less time requires faster velocities and more blending (rounding) of corners of the straight-line path, therefore, higher deviation from the straight-line path. This may seem intuitive, but consider a vehicle under active control attempting to track the generated trajectory. Higher velocities and sharper (less rounded) corners will likely be more difficult for the controller/vehicle to execute, resulting in a larger tracking error. That is, whereas here, the cross-track error is calculated as the difference between the straight-line path and the generated trajectory and results in an increase in error as mission time is decreased, actual tracking of the generated trajectory by a controller/vehicle may produce the opposite trend as the vehicle cannot perfectly track a piecewise straight-line trajectory without stopping. Shorter times and increased velocities also result in larger accelerations and thus larger power requirements. In contrast, the maximum power usage data in Table 5.1 is more consistent, implying at least some saturation of thrust (max acceleration reached in one or multiple directions) was achieved during each mission. An examination of the accelerations produced by the trajectory generator confirms this is the case. It is also interesting to note that in several cases of minimum mission time the maximum power usage is reduced as average power increases likely due to larger turning radii. Evaluation of time ratios for linear segments and blend regions supports intuition wherein more time spent in blend regions, where larger accelerations occur, results in higher power consumption by the system.

5.5.1 Pseudo Trajectories

Trajectories are also generated with the use of pseudo points to convert the pre-specified waypoints into through points in the trajectory. The same test trajectories shown in Figures 5.2 and 5.3 are shown with pseudo points in Figures 5.4 and 5.5.

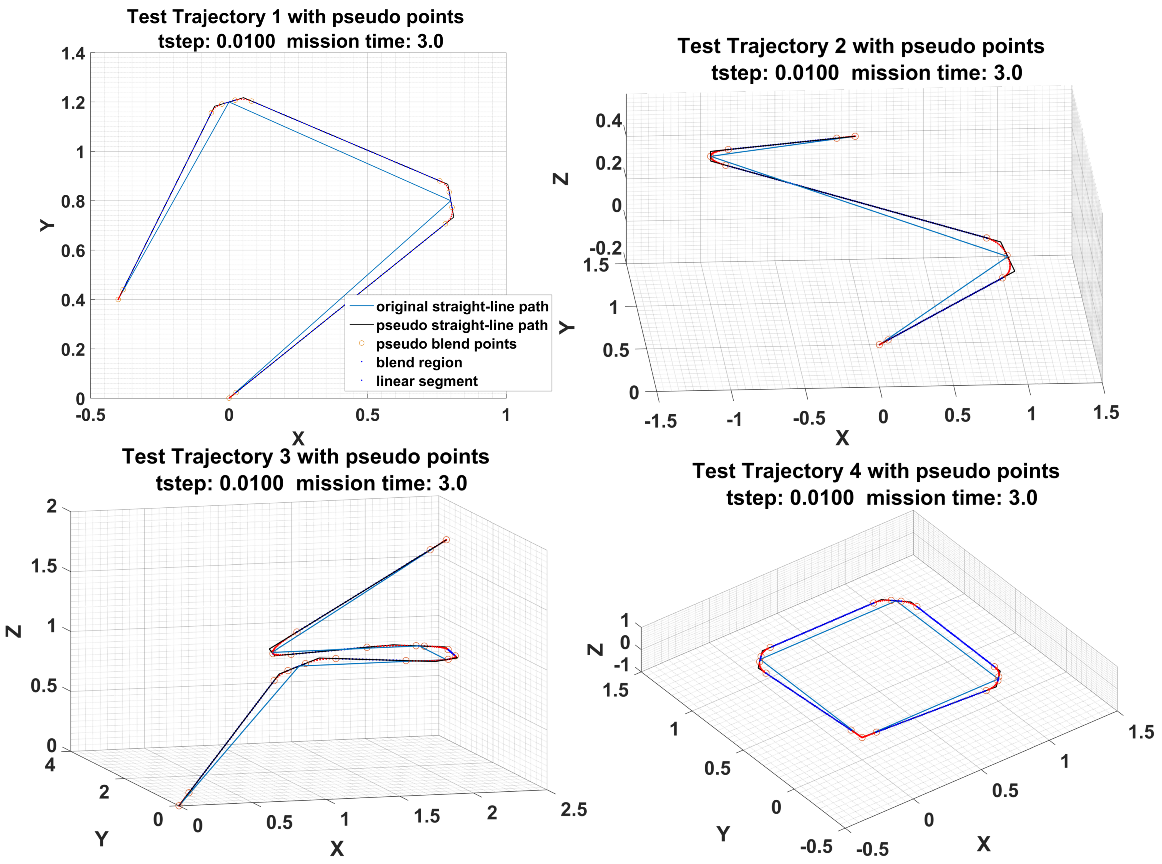


Figure 5.4: Test trajectories with pseudo points generated with a mission execution time of three seconds and time-step resolution of 0.01 seconds

Similar analysis of the pseudo trajectories is conducted and illustrates the same trends as that of Table 5.1. The data for the pseudo trajectories is shown in Table 5.2. Comparison

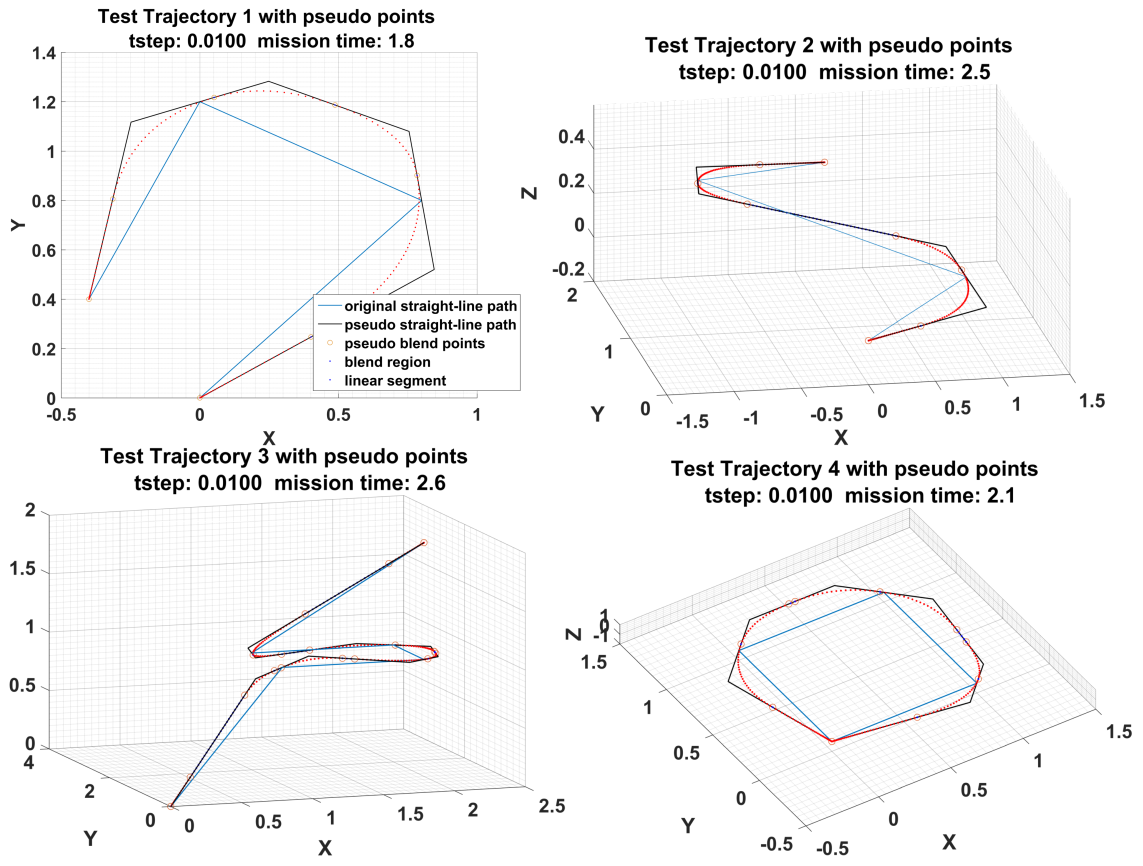


Figure 5.5: Test trajectories with pseudo points generated with a minimum execution time (sec) which results in a continuous path and time-step resolution of 0.01 seconds

	Trajectory 1			Trajectory 2			Trajectory 3			Trajectory 4		
Mission Time:	5sec	3sec	Min: 1.8	5sec	3sec	Min: 2.5	5sec	3sec	Min: 2.6sec	5sec	3sec	Min: 2.1sec
Cross-Track Error:	0.0084	0.0338	0.9020	0.0424	0.1593	1.0857	0.1833	0.4583	1.0532	0.0275	0.1318	1.0342
Avg. Power Est. (J/s):	86.36	95.80	124.82	89.47	104.70	118.17	99.86	133.28	168.29	88.98	105.37	140.27
Max Power Est. (J/s):	217.73	213.20	172.55	213.90	206.68	158.15	370.51	366.60	362.74	171.92	176.90	171.27
$t_{linear} : t_{blend}$	20.160	6.308	0	10.167	2.858	0.188	5.747	1.339	0.336	10.209	2.843	0.028
$\%t_{linear}$	95.3	86.3	0	91.0	74.1	15.8	85.2	57.2	25.1	91.1	74.0	2.7
$\%t_{blend}$	4.7	13.7	100	9.0	25.9	84.2	14.8	42.8	74.9	8.9	26.0	97.3

Table 5.2: Evaluation metrics of test trajectories with pseudo points generated with time-step resolution of 0.01 seconds

of Table 5.1 and Table 5.2 shows that with the addition of pseudo points a larger cross-track error occurs as there are more blend regions. Note that the cross-track here is calculated as the difference between the generated pseudo trajectory and the *pseudo* straight-line path. Power estimates are comparable, though slightly lower with the use of pseudo points despite the increase in the amount of time spent in blend regions. This is likely a result of the fact that given the pseudo points the trajectory produces larger turns (less ’sharp’ corners) than the trajectory without pseudo points, thus requiring lower velocities and accelerations, and consequently less power.

5.5.2 Trajectory Tracking

In using the trajectory generation strategy outlined above in conjunction with the co-regulation design described in Chapter 4, future work is required. The discrete LQR scheme described above requires modification in order to track both positions and velocities without generating conflicting control signals between the two. That is, under positional control the controller will attempt to bring the vehicle to a stop after driving it to the next waypoint; however, using integrator states on velocity will also try to bring the vehicle to a non-zero velocity at the next waypoint as specified by the trajectory generator. This results in poor tracking of the generated trajectory.

Trajectory following for UAS is an area of high research interest and proves to be non-trivial. The under-actuated nature of a quadrotor UAS categorizes it in a unique class of mechanical systems [88] for which trajectory tracking is not well understood. Classical control techniques can be used to track trajectories if the system model is decoupled and linearized around local operating points in such a way that the system is divided into several fully actuated systems or the system becomes linearly time-invariant through design of several scheduled controllers for different portions of the flight envelope [89]. One example of this methodology is shown in [58] in which a UAS is commanded to a reference position while achieving a specified velocity (i.e. set-point tracking). However, these methods are unable to accommodate for singularities resulting from excessive angles of attack, thus constraining their application to more fundamental trajectories and, although beneficial in enforcing timing constraints on waypoint following, they cannot be effectively used to follow the trajectories described here due to the relatively short spacing between successive waypoints. Amongst the robotics community, perhaps the most promising trajectory tracking technique for this work is that of a supervisory back-stepping nonlinear control technique [89, 90, 91, 92]. In [89] a Lyapunov-based back-stepping control law is formulated and an estimator-based supervisory control law discretely switches between controllers to solve the trajectory tracking problem for an under-actuated system. Similarly in [92] a quadrotor UAS system is divided into three subsystems (under-actuated, fully-actuated, and rotor subsystems) and nonlinear back-stepping control techniques are also used. A geometric (coordinate independent) approach to the integrator back-stepping formulation is also used in [91] for a helicopter model and in [93] a purely geometric approach with a hierarchical tracking control scheme is implemented on a quadrotor UAS. Finally, in [90] a back-stepping trajectory tracking controller is implemented along with numerical feed-forward differentiation and filter compensation to decouple linear and rotational dynamics without the use of an inner/outer loop structure.

In addition to trajectory tracking, a planner which can deliberate and assign target sampling rates to the generated trajectory to optimize the allocation of cyber and physical resources is needed for full functionality with the co-regulation design. Although, the described trajectory generation is suitable for such a planner in that each trajectory point is calculated at a discrete time-step which may be easily varied during trajectory generation, a planner which can capitalize on this feature has yet to be designed.

Chapter 6

Conclusions

Control and real-time computing are coupled by implementing control laws on a digital device requiring the periodic execution of a task. Characterizing this coupling and the performance of the system allows us to design planning algorithms that trade off cyber and physical resources and ensure predictable performance. In Chapter 3, we have investigated and quantified the effects of varying sampling periods of a controller on a quadrotor UAS as it follows various trajectories. This provides a mathematical relationship for developing a cyber-physical planning algorithm that trades off cyber and physical resources for improved mission performance.

We also introduced new metrics that quantify both the physical and cyber performance of a quadrotor UAS following a reference trajectory. The results provide us with a means for developing a higher-level CPS planner that computes coupled cyber-physical trajectories and reference commands for a low-level reactive cyber-physical control strategy. The results also serve as a pointer to the awareness for considering timing requirements while designing control laws.

A new hybrid method is proposed in Chapter 4, where physical and computational effectors are co-regulated simultaneously. Our method leverages the benefits of feedback control to vary, in discrete-time, the sampling period of the controller according to physical system performance, which subsequently is used to calculate the control law for the physical effectors.

We implemented our co-regulation strategy in a full nonlinear simulation environment, and, going beyond traditional control analysis, we explored trajectory-following performance for our UAS under several fixed-rate control strategies as well as our co-regulation strategy. We have shown that significant computational resource savings can be realized while still maintaining reasonable control performance. The co-regulation strategy less aggressively applies physical control inputs (i.e. smaller thrusts) for longer periods of time, which results in less precise flight performance but with significant savings in computational processing. Finally, we have provided strong evidence that control strategies that execute at $> 50\,\mathrm{Hz}$ most likely provide little to no trajectory following performance improvement, but have high computational cost. Those resources could be used to improve performance on other tasks, particularly those involving data collection and transmission.

Toward designing a cyber-physical trajectory generator, in Chapter 5 we have used a computationally simple kinematic trajectory generator with constraints on maximum velocities and accelerations. The generator uses linear approximations with parabolic blends to turn a straight-line path connecting pre-specified waypoints into a continuous function through space with associated velocities and accelerations. The generator also ensures the mission is executed in the desired mission time so long as saturation is not reached. The generated trajectory can be augmented with the addition of pseudo waypoints to provide a trajectory that passes through the pre-specified waypoints, instead of deviating from them to better track the straight-line path. The implications of estimated power and time requirements were examined and show that a reduction in mission execution time results in larger blends, and as a result, larger power requirements and cross-tracking error when comparing the generated trajectory to the straight-line path. Implementation of the trajectory tracker in a control system architecture is needed to better asses the effectiveness of the designed trajectory generator and how it may best serve a cyber-physical system.

6.1 Future Work

Future work is most immediately focused on obtaining experimental flight results on our AscTec Hummingbird platform. The proposed co-regulation is non-trivial for implementation due to lack of an on-board real-time system, but empirical results on a physical system will provide much more insight into the magnitude of computational savings and performance limitations. Toward implementation on a physical vehicle it may also be useful to explore the effects of external disturbances and noise on the system. We did not explore these effects in this work because we were interested only in the relationship between different sampling rates and control and as such wanted to isolate those effects. However, in Appendix C we present a possible approach to acquiring a meaningful noise profile to apply to our control technique. Prior to implementation on a physical system we also desire formal guarantees on stability of the system. As a formal stability analysis of rate varying control techniques is of current research interest [94, 95, 96], a standard technique has yet to be accepted by the community. Still a formal stability analysis for the described co-regulation technique is an aim for future work; though, an approach to such analysis remains unclear as conducting stability analysis for each discrete controller over a range of sampling rates from 1 Hz to 1000 Hz may be highly inefficient. In addition, complications arising from non-ideal sensors will likely require investigation into real-time optimal sensor scheduling techniques as mentioned in Chapter 2. As the controllers in this work were manually tuned during experiments, an optimal tuning technique for co-regulation is also desired. Toward the goal of a higher level motion planning architecture, future work will also include investigation into trajectory tracking and adapting successful techniques to account for cyber trajectory generation as well as physical trajectory tracking. This points to an all encompassing analytical model for which optimal execution with regard to CPS effectors can be achieved at a mission level.

Appendix A

Quadrotor Dynamics

In the following discussion we will derive a dynamic model for a small quadrotor vehicle for further use in simulation and control environments. We begin by first defining an inertial reference frame $\{\hat{i},\hat{j},\hat{k}\}$. This frame is considered fixed with the origin at the point of takeoff for the vehicle and the positive $\hat{k}$ direction pointing upward. Our physical model of the quadrotor vehicle will consist of a solid sphere of mass M and radius r with four thin

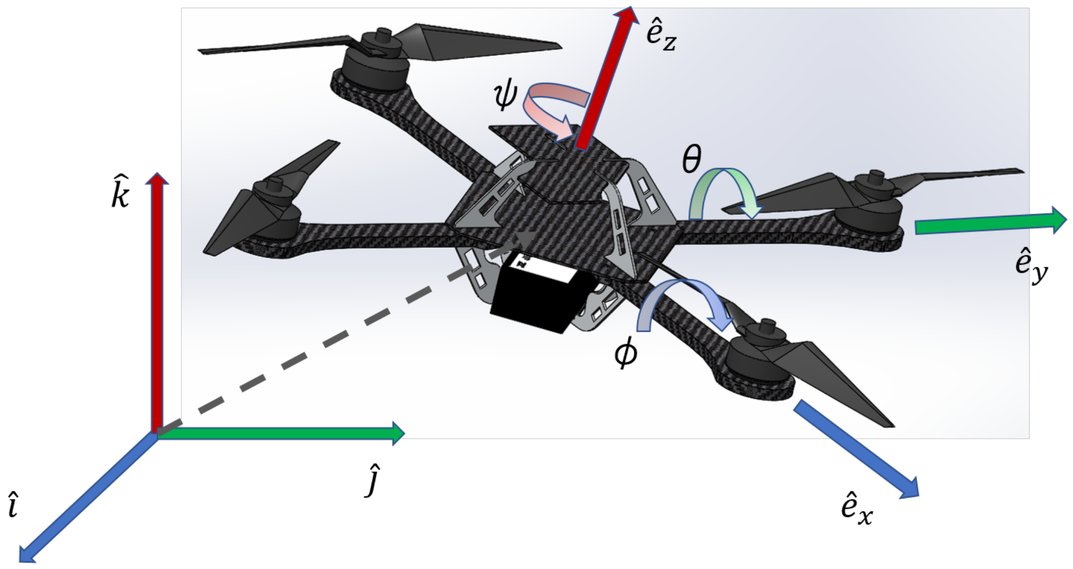


Figure A.1: Quadrotor frame orientations.

rods extending from its center at right angles from each other (all in the same plane) with a point mass m_r at the end of each of them. The mass of each thin rod is noted as m_{rod}. We then define a right handed body (or vehicle) frame $\{\hat{e}_x,\hat{e}_y,\hat{e}_z\}$ with the origin fixed at the center of the vehicle (assumed to be the center of mass), the positive $\hat{e}_x$ axis pointing out of what will be considered the front of the vehicle, the positive $\hat{e}_z$ axis pointing upward, and the $\hat{e}_x$ and $\hat{e}_y$ axis aligned with the arms of the quadrotor. The position of the body frame is simply calculated by the vector connecting the origins of each frame, and the orientation of the body frame (along with any vector or matrix in the body frame) can be described with the application of a rotation matrix $\boldsymbol{R}$. This rotation matrix is constructed using three intrinsic[1] rotations about the inertial axes through respective Tait-Bryan angles[2] $\{\phi,\theta,\psi\}$ as follows:

$$\boldsymbol{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & sin(\phi) \\ 0 & -sin(\phi) & cos(\phi) \end{bmatrix}$$

$$\boldsymbol{R}_{y'} = \begin{bmatrix} cos(\theta) & 0 & -sin(\theta) \\ 0 & 1 & 0 \\ sin(\theta) & 0 & cos(\theta) \end{bmatrix}$$

[1]Intrinsic rotations are are described in terms of the moving (rotated) frame (i.e. $\boldsymbol{R}_x\boldsymbol{R}_{y'}\boldsymbol{R}_{z''}$). The second and third rotations are about 'new' or non-inertial axis which are defined by the previous rotation. Successive extrinsic rotations in contrast are defined as rotations about only the inertial axes or original (fixed) frame no matter the orientation of the moving frame (the frame being rotated) (i.e. $\boldsymbol{R}_x\boldsymbol{R}_y\boldsymbol{R}_z$) .

[2]Classical (proper) Euler angles include rotations about first an axis in the fixed frame, followed by a rotation about an intermediate (nodal) axis, and then a rotation about the same axis as the first rotation (i.e. $\boldsymbol{R}_z\boldsymbol{R}_y\boldsymbol{R}_z$). On the contrary Tait-Bryan angles include rotations about three separate axes (i.e. $\boldsymbol{R}_x\boldsymbol{R}_y\boldsymbol{R}_z$). The number of possible rotation combinations including both the classical Euler and Tait-Bryan angles totals at 12. Including both extrinsic and intrinsic combinations doubles the total possibilities. Intrinsic Tait-Bryan angles are the convention in aeronautics as they are intuitive when describing the roll, pitch, and yaw of a vehicle

$$\boldsymbol{R}_{z''} = \begin{bmatrix} cos(\psi) & sin(\psi) & 0 \\ -sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{R}_{xyz} = \boldsymbol{R}_{z''}\boldsymbol{R}_{y'}\boldsymbol{R}_{x} \tag{A.1}$$

$$= \begin{bmatrix} c(\psi)cos(\theta) & c(\theta)sin(\psi) & -s(\theta) \\ -c(\phi)s(\psi) + c(\psi)s(\phi)s(\theta) & c(\phi)c(\psi) + s(\phi)s(\psi)s(\theta) & c(\theta)s(\phi) \\ s(\phi)s(\psi) + c(\phi)c(\psi)s(\theta) & -c(\psi)s(\phi) + c(\phi)s(\psi)s(\theta) & c(\phi)c(\theta) \end{bmatrix}$$

where $c(\theta)$ and $s(\theta)$ are short-hand notation for $cos(\theta)$ and $sin(\theta)$ respectively. Converting from the body frame to the inertial frame is achieved by applying the same rotations in the reverse order as:

$$\boldsymbol{R}_{xyz}^{T} = \boldsymbol{R}_{zyx}(-\phi, -\theta, -\psi)$$

Note here that the correct mathematical derivation for reversing the transform includes multiplying an equation by the inverse of each rotation matrix in the correct order, but for rotations about coordinate axes the inverse is equal to the transpose. Also note that here $\boldsymbol{R}_{x}, \boldsymbol{R}_{y'}, \boldsymbol{R}_{z''}$are written in terms of counter-clockwise rotations as viewed looking at the origin down the axis of rotation. In terms of aviation, these rotations are known as roll (ϕ), pitch (θ), and yaw (ψ), and the body frame orientation may be referred to as the ENU (East North Up) convention[3].

Using Newtonian mechanics, the motion of the center of mass of a quadrotor, relative to the inertial frame, can be modeled by calculating the sum of the forces on the vehi-

[3]ENU along with NED (North East Down) are standard axes conventions for depicting aerial vehicle orientation. ENU has been selected because it is consistent with the software (ROS) we utilize for autonomous flight (see http://www.ros.org/reps/rep-0103.html#coordinate-frame-conventions).

cle. In particular there exists a gravitational force, a thrust produced by each of the four rotors analogous to a net thrust at the center of mass, and a force due to drag. There is most certainly an amount of selectivity when considering the level of accuracy at which to approximate these forces as may become clear in later discussion. Each of the four independent motor thrusts produces a moment about the center of the vehicle in either the $\hat{e}_x$ or $\hat{e}_y$ axis of the body frame (roll and pitch respectively). As a result, unequal motor thrusts result in a rotation of the quadrotor in space. A reactive moment is also produced in the $\hat{e}_z$ axis (yaw) as a result of drag induced as each rotor travels through the air along with conservation of momentum. In order to achieve zero yaw, rotors adjacent to one another are rotated in opposite directions canceling opposing moments, when operated at equal speeds. The resulting equations of motion are then:

$$\sum \boldsymbol{F}_{cm} = m\boldsymbol{a}_{cm} = \boldsymbol{T}_{net} + \boldsymbol{F}_{drag} - m\boldsymbol{g} \tag{A.2}$$

$$\sum \boldsymbol{M}_{cm} = \boldsymbol{\tau} = \boldsymbol{I}_{cm}\boldsymbol{\alpha} + \boldsymbol{\omega} \times \boldsymbol{I}_{cm}\boldsymbol{\omega} \tag{A.3}$$

where $m = M + 4m_r + 4m_{rod}$ is the total mass of the vehicle, $\boldsymbol{a}_{cm}$ is the acceleration of the center of mass, $\boldsymbol{T}_{net}$ is the net thrust, $\boldsymbol{F}_{drag}$ is the drag force, $g = 9.8066\frac{m}{s^2}$, $\boldsymbol{I}_{cm}$ is the moment of inertia tensor, $\boldsymbol{\alpha}$ is the angular acceleration, $\boldsymbol{\tau}$ is the torques exerted on the vehicle by the motors, and $\boldsymbol{\omega}$ is the angular velocity. Alternatively the above moment equation may be written using the skew symmetric matrix[4] form of the angular velocity vector, $\boldsymbol{\Omega}$, as follows:

$$\boldsymbol{\tau} = \boldsymbol{I}_{cm}\boldsymbol{\alpha} + \boldsymbol{\Omega}\boldsymbol{I}_{cm}\boldsymbol{\omega}$$

[4]A skew symmetric matrix is one in which $-A = A^T$. The skew symmetric form of a vector involves filling the skew symmetric matrix with the components of the vector such that multiplying an arbitrary vector by the skew symmetric matrix is the equivalent of performing the outer (cross) product of the two vectors (i.e. $A_a b = a \times b$ where a and b are vectors and A is the skew symmetric form of vector a).

where for any vector of constant magnitude $\boldsymbol{\lambda}$,

$$\boldsymbol{\Omega}\boldsymbol{\lambda} = \boldsymbol{\omega} \times \boldsymbol{\lambda}$$

$\boldsymbol{\Omega}$ may be derived in the following manner [44]:

$$\boldsymbol{\lambda}_f = \boldsymbol{R}\boldsymbol{\lambda}_0$$

$$\implies \boldsymbol{\lambda}_0 = \boldsymbol{R}^T\boldsymbol{\lambda}_f$$

$$\dot{\boldsymbol{\lambda}}_f = \dot{\boldsymbol{R}}\boldsymbol{\lambda}_0 = \dot{\boldsymbol{R}}(\boldsymbol{R}^T\boldsymbol{\lambda}_f)$$

The vector representation of angular velocity gives us [97, 44]:

$$\dot{\boldsymbol{\lambda}} = \boldsymbol{\omega} \times \boldsymbol{\lambda}$$

$$\therefore \dot{\boldsymbol{\lambda}}_f = \boldsymbol{\omega} \times \boldsymbol{\lambda}_f = \dot{\boldsymbol{R}}\boldsymbol{R}^T\boldsymbol{\lambda}_f = \boldsymbol{\Omega}\boldsymbol{\lambda}_f$$

$$\implies \boldsymbol{\Omega} = \dot{\boldsymbol{R}}\boldsymbol{R}^T$$

However, because $\dot{\boldsymbol{R}}$ is non-trivial to calculate, another approach may be used. This method essentially corrects for each intermediate rotation of the transform $\boldsymbol{R}$ leaving the residual components of angular velocity at each stage in terms of the Tait-Bryan angles via the equation [67]:

$$\boldsymbol{\omega} = \dot{\theta}_3\hat{e}_3 + \boldsymbol{R}_3\dot{\theta}_2\hat{e}_2 + \boldsymbol{R}_3\boldsymbol{R}_2\dot{\theta}_1\hat{e}_1$$

for the transform listed above, $\boldsymbol{R}_{xyz}$, this equation yields:

$$\boldsymbol{\omega} = \dot{\psi}\hat{k} + \boldsymbol{R}_z\dot{\theta}\hat{j} + \boldsymbol{R}_z\boldsymbol{R}_y\dot{\phi}\hat{i}$$

$$\boldsymbol{\omega} = \begin{bmatrix} cos(\theta)cos(\psi) & sin(\psi) & 0 \\ -cos(\theta)sin(\psi) & cos(\psi) & 0 \\ sin(\theta) & 0 & 1 \end{bmatrix} \begin{Bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{Bmatrix} \tag{A.4}$$

The components of ω may then be used to construct the skew symmetric form Ω as:

$$\boldsymbol{\Omega}_{xyz} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

Due to the simplicity of geometry used to describe the physical model of the quadrotor, the moment of inertia tensor becomes a trivial sum of commonly known inertia tensors for basic shapes when considering rotations about the principal axes. We will start with the sphere of mass M and radius r centered at the origin of the body frame. The inertial tensor for a uniform sphere about an axis coinciding with its diameter is [98]:

$$\boldsymbol{I}_s = \begin{bmatrix} \frac{2}{5}Mr^2 & 0 & 0 \\ 0 & \frac{2}{5}Mr^2 & 0 \\ 0 & 0 & \frac{2}{5}Mr^2 \end{bmatrix}$$

Next for a point mass m at a distance of l away from the origin along the $\hat{e}_x$ axis the moment of inertia is:

$$\boldsymbol{I}_p = \begin{bmatrix} ml^2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

For modeling each rotor as a point mass positioned on each of the $\hat{e}_x$, $-\hat{e}_x$, $\hat{e}_y$, and$-\hat{e}_y$ axes we get:

$$\boldsymbol{I}_r = \begin{bmatrix} 2ml^2 & 0 & 0 \\ 0 & 2ml^2 & 0 \\ 0 & 0 & 4ml^2 \end{bmatrix}$$

The moment of inertia tensor for a uniform thin rod of mass m_{rod} and length l about an axis perpendicular to the bar, and at the end of the bar, is used to model the arms of the vehicle. This inertial moment is $I = \frac{1}{3}m_{rod}l^2$[98]. Following the same symmetry as used for calculating moment of inertia tensor for the network of point masses above, the inertial tensor for four rods each with one end at the origin and aligned with the $\hat{e}_x$, $-\hat{e}_x$, $\hat{e}_y$, and$-\hat{e}_y$ axes the tensor becomes:

$$\boldsymbol{I}_{rod} = \begin{bmatrix} \frac{2}{3}m_{rod}l^2 & 0 & 0 \\ 0 & \frac{2}{3}m_{rod}l^2 & 0 \\ 0 & 0 & \frac{4}{3}m_{rod}l^2 \end{bmatrix}$$

The total mass moment of inertia tensor for our physical model is then simply the sum of each of these three tensors, $I_{cm} = I_s + I_r + I_{rod}$.

By examining the mechanics of a single stationary rotor system consisting of a motor and propeller one can calculate a thrust approximation for a flying single or multi-rotor vehicle at a state of hover. By conservation of energy we have that:

$$KE_0 + \sum W_{0-f} = KE_f$$

where KE is kinetic energy and W is work done in the system. For a stationary rotor system the work done by the rotor on the air is:

$$W \equiv \int P dt = \int \boldsymbol{F} \cdot \boldsymbol{v} dt$$

where P is the power expended by the system, $\boldsymbol{F}$ is the force the propeller exerts on the air also known as the thrust, and v is the velocity of the air displaced by the propeller. When considering a stationary system and assuming there is no free stream movement of the surrounding air (i.e. no wind), we can assume that the majority of displaced air moves with a velocity parallel to the force $\boldsymbol{F}$. Replacing $\boldsymbol{F}$ with $\boldsymbol{T}$ for thrust and differentiating, it follows that:

$$P = Tv$$

Momentum theory tells us that for a thin actuator disk (i.e. a propeller spinning at sufficient speed) of area A pushing a fluid with density ρ, the power required to produce a given thrust is:

$$P = \sqrt{\frac{T^3}{2\rho A}}$$

therefore,

$$v = \sqrt{\frac{T}{2\rho A}}$$

Solving for T and substituting $v = \varpi r'$ where ϖ is the angular velocity of the rotor and r' is the radius of the rotor, gives us a usable equation for thrust:

$$T = 2\rho A r'^2 \varpi^2$$

Noting that the rotating propeller is, in fact, not a thin disk, better results may be attained by experimentally determining a thrust coefficient C_T to be used in the previous equation yielding [66]:

$$T = C_T \rho A r'^2 \varpi^2$$

Turning to a electro-mechanical view of the system and examination of the motor used to drive the propeller (a DC brushless motor in this discussion), the power expended by the system may also be modeled as:

$$P = i_m V$$

Characteristic of a DC brushless motor, the torque generated is related to the current by a motor torque coefficient κ_τ by:

$$\tau_m = \kappa_\tau \left(i_m - i_{NL} \right) = \kappa_\tau i_L$$

where i_{NL} is the no load current of the motor, τ_m is the motor torque, and i_m is the amount of current drawn by the motor. Using Ohm's law we also know that the voltage across the motor is:

$$V - V_{EMF} = i_m R_m$$

where V is the nominal voltage, R_m is the impedance of the motor, and $V_{EMF} = \kappa_v \varpi$ is the back-emf of the motor. It follows that:

$$P = i_m^2 R_m + i_m \kappa_v \varpi$$

$$= (\frac{\tau_m}{\kappa_\tau} + i_{NL})^2 R_m + (\frac{\tau_m}{\kappa_\tau} + i_{NL}) \kappa_v \varpi$$

Assuming that $i_{NL} \ll i_L$and i_{NL} is therefore negligible, and that our motor is sufficiently efficient (implying resistive losses are also negligible) the power of the system may be approximated as:

$$P \approx \frac{\tau_m}{\kappa_\tau} \kappa_v \varpi$$

Noting that the torque on the motor is linearly proportional to the amount of thrust produced by the rotor by some constant κ_T we can write [99]:

$$P \approx \frac{\kappa_v \kappa_T}{\kappa_\tau} T \varpi$$

$$\implies \frac{\kappa_v \kappa_T}{\kappa_\tau} T\varpi = \sqrt{\frac{T^3}{2\rho A}}$$

$$T = 2\rho A(\frac{\kappa_v \kappa_T}{\kappa_\tau})^2 \varpi^2$$

In either case the thrust may be modeled as proportional to the square of the angular velocity of the propeller:

$$T = k\varpi^2$$

where k is a physical parameter of the system.

Given the geometry of our physical model, the torques (moments) in the $\hat{e}_x$ and $\hat{e}_y$ direction, as mentioned, are generated by the thrust of each individual rotor by:

$$\boldsymbol{\tau} = \sum \boldsymbol{r} \times \boldsymbol{F} = \sum \boldsymbol{l} \times \boldsymbol{T}$$

more specifically, the torques about the pitch and roll axes, τ_θ and τ_ϕ respectively, become:

$$\boldsymbol{\tau}_\phi = l\hat{e}_y \times k\varpi_2^2 \hat{e}_z - l\hat{e}_y \times k\varpi_4^2 \hat{e}_z$$

$$= lk\varpi_2^2 \hat{e}_x - lk\varpi_4^2 \hat{e}_x$$

$$\boldsymbol{\tau}_\theta = l\hat{e}_x \times k\varpi_1^2 \hat{e}_z - l\hat{e}_x \times k\varpi_3^2 \hat{e}_z$$

$$= lk\varpi_1^2 \hat{e}_y - lk\varpi_3^2 \hat{e}_y$$

Forces due to drag on the vehicle are also important to consider when dealing with aerodynamics. As mentioned above one important artifact of drag on the rotors is the generation of a reactive torque in the $\hat{e}_z$ direction formerly known as yaw. The equation for drag on an object transversing a fluid (in this case air) is given as [99]:

$$F'_{drag} = \frac{1}{2} C_{drag} \rho A_c \nu^2$$

where F'_{drag} is the force due to drag on the propeller, C_{drag} is the drag coefficient, ρ is again the density of air, A_c is the cross-sectional area of the propeller blade, and $\nu = \varpi r'$ is the translational velocity of the blade. This equation assumes the propeller blade is moving at a high relative velocity, reflected in the square of the term ν. This assumption along with the value of C_{drag} are dependent on the Reynolds[5] number of the system. Making the substitution $\nu = \varpi r'$ the reactive torque on the vehicle generated by the drag may then be represented as [99]:

$$\tau_{drag} = \frac{1}{2} C_{drag} \rho A_c r'^3 \varpi^2$$

Once again this may also be simply modeled as:

$$\tau_{drag} = b\varpi^2$$

where b is a physical parameter of the system. Summing τ_{drag} for all of the rotors while taking into consideration the direction of drag, we can then write an equation for the net yaw torque:

$$\tau_\psi = b\varpi_1^2 - b\varpi_2^2 + b\varpi_3^2 - b\varpi_4^2$$

There are several other affects of drag to be considered in modeling a quadrotor system in flight. For the drag force F_{drag} seen in Equation A.2 above, we will assume the vehicle moves with relatively low velocity in near laminar conditions (low Reynolds number)

[5]Reynolds number is a dimensionless ratio of inertial forces to viscous forces for an object transversing a fluid. This ratio is used to characterize the nature of flow of the fluid in which the object is immersed. High Reynolds numbers (on the order of 10^3 and above) are associated with aggressive or turbulent flow while low Reynolds numbers (on the order of $10^{-1}$and below) are associated with passive (laminar) or negligble flow.

therefore inducing Stokes's drag, or linear drag, modeled as:

$$F_{drag} = -D\upsilon$$

where D is drag coefficient dependent on physical geometry of the vehicle and properties of the fluid it transverses (air), and υ is the velocity of the vehicle. Some other retarding affects not represented here include blade flapping, interference drag, non-laminar drag, and so on. Note that in the following equations D becomes a matrix containing drag coefficiants for each of the three principal axes of the quadrotor along the diagonal. These effects, although important in some scenarios, are not considered dominant in the model described in this discussion.

Equations A.2 and A.3 now become [67]:

$$m\left\{\begin{array}{c} a_x \\ a_y \\ a_z \end{array}\right\} = \boldsymbol{R}^T_{xyz}\left\{\begin{array}{c} 0 \\ 0 \\ (\sum k\varpi_i^2)_{z'} \end{array}\right\} - \boldsymbol{D}_{xyz}\left\{\begin{array}{c} \upsilon_x \\ \upsilon_y \\ \upsilon_z \end{array}\right\} - m\left\{\begin{array}{c} 0 \\ 0 \\ g_z \end{array}\right\}$$

$$\left\{\begin{array}{c} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{array}\right\} = \boldsymbol{I}_{cm}\left\{\begin{array}{c} \alpha_\phi \\ \alpha_\theta \\ \alpha_\psi \end{array}\right\}_{b/i} + \boldsymbol{\Omega}_{x'y'z'}\boldsymbol{I}_{cm}\left\{\begin{array}{c} \omega_\phi \\ \omega_\theta \\ \omega_\psi \end{array}\right\}_{b/i}$$

where $\{\hat{i}, \hat{j}, \hat{k}\} \Longrightarrow \left\{\begin{array}{c} x \\ y \\ z \end{array}\right\}$ and $\{\hat{e}_x, \hat{e}_y, \hat{e}_z\} \Longrightarrow \left\{\begin{array}{c} x' \\ y' \\ z' \end{array}\right\}$. The notation b/i represents the derivatives of angles in the body frame with respect to the inertial frame. Note that the equation of rotational dynamics is expressed in the body frame of reference while the equation of translational dynamics is expressed in the inertial frame of reference. Solving for accelerations and expressing both in the body frame yields [67]:

$$\begin{Bmatrix} a_{x'} \\ a_{y'} \\ a_{z'} \end{Bmatrix} = \boldsymbol{R}_{xyz} \begin{Bmatrix} a_x \\ a_y \\ a_z \end{Bmatrix} = \frac{1}{m} \begin{Bmatrix} 0 \\ 0 \\ \sum k\varpi_i^2 \end{Bmatrix} - \frac{1}{m} \boldsymbol{D}_{x'y'z'} \boldsymbol{R}_{xyz} \begin{Bmatrix} \upsilon_x \\ \upsilon_y \\ \upsilon_z \end{Bmatrix} - \boldsymbol{R}_{xyz} \begin{Bmatrix} 0 \\ 0 \\ g \end{Bmatrix} - \begin{Bmatrix} \omega_\theta \upsilon_{z'} - \omega_\psi \upsilon_{y'} \\ \omega_\psi \upsilon_{x'} - \omega_\phi \upsilon_{z'} \\ \omega_\phi \upsilon_{y'} - \omega_\theta \upsilon_{x'} \end{Bmatrix} \tag{A.5}$$

$$\begin{Bmatrix} a_{x'} \\ a_{y'} \\ a_{z'} \end{Bmatrix} = \frac{1}{m} \begin{Bmatrix} 0 \\ 0 \\ \sum k\varpi_i^2 \end{Bmatrix} - \frac{1}{m} \begin{Bmatrix} D_{x'} \upsilon_{x'} \\ D_{y'} \upsilon_{y'} \\ D_{z'} \upsilon_{z'} \end{Bmatrix} - \begin{Bmatrix} -gsin(\theta) \\ gcos(\theta)sin(\phi) \\ gcos(\theta)cos(\phi) \end{Bmatrix} - \begin{Bmatrix} \omega_\theta \upsilon_{z'} - \omega_\psi \upsilon_{y'} \\ \omega_\psi \upsilon_{x'} - \omega_\phi \upsilon_{z'} \\ \omega_\phi \upsilon_{y'} - \omega_\theta \upsilon_{x'} \end{Bmatrix}$$

$$\begin{Bmatrix} \alpha_\phi \\ \alpha_\theta \\ \alpha_\psi \end{Bmatrix}_{b/i} = \boldsymbol{I}_{cm}^{-1} \begin{Bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{Bmatrix} - \boldsymbol{I}_{cm}^{-1} \boldsymbol{\Omega}_{x'y'z'} \boldsymbol{I}_{cm} \begin{Bmatrix} \omega_\phi \\ \omega_\theta \\ \omega_\psi \end{Bmatrix}_{b/i} \tag{A.6}$$

$$\begin{Bmatrix} \alpha_\phi \\ \alpha_\theta \\ \alpha_\psi \end{Bmatrix}_{b/i} = \begin{Bmatrix} \frac{\tau_\phi}{I_x} \\ \frac{\tau_\theta}{I_y} \\ \frac{\tau_\psi}{I_z} \end{Bmatrix} - \begin{Bmatrix} \frac{\omega_\theta \omega_\psi I_z - \omega_\theta \omega_\psi I_y}{I_x} \\ \frac{\omega_\phi \omega_\psi I_x - \omega_\phi \omega_\psi I_z}{I_y} \\ \frac{\omega_\phi \omega_\theta I_y - \omega_\phi \omega_\theta I_x}{I_z} \end{Bmatrix}$$

where $\boldsymbol{I}_{cm}^{-1} = \begin{bmatrix} \frac{1}{I_{xx}} & 0 & 0 \\ 0 & \frac{1}{I_{yy}} & 0 \\ 0 & 0 & \frac{1}{I_{zz}} \end{bmatrix}$. Here again the vector representation of angular velocity generates an additional term for rotational correction in the translational equation. Recall that from Equation A.4 we may also write Equations A.5 and A.6 in terms of the Tait-Bryan angles. Expressing Equations A.5 and A.6 in the inertial frame gives us:

$$\begin{Bmatrix} a_x \\ a_y \\ a_z \end{Bmatrix} = \frac{1}{m}\boldsymbol{R}_{xyz}^T \begin{Bmatrix} 0 \\ 0 \\ \sum k\varpi_i^2 \end{Bmatrix} - \frac{1}{m}\boldsymbol{D}_{xyz} \begin{Bmatrix} \upsilon_x \\ \upsilon_y \\ \upsilon_z \end{Bmatrix} - \begin{Bmatrix} 0 \\ 0 \\ g \end{Bmatrix} \tag{A.7}$$

$$\begin{Bmatrix} a_x \\ a_y \\ a_z \end{Bmatrix} = \frac{1}{m} \begin{Bmatrix} [sin(\phi)sin(\psi) + cos(\phi)cos(\psi)sin(\theta)]\sum k\varpi_i^2 \\ [-cos(\psi)sin(\phi) + cos(\phi)sin(\psi)sin(\theta)]\sum k\varpi_i^2 \\ [cos(\phi)cos(\theta)]\sum k\varpi_i^2 \end{Bmatrix}$$
$$- \frac{1}{m}\boldsymbol{R}_{xyz}^T\boldsymbol{D}_{x'y'z'} \begin{Bmatrix} \upsilon_x \\ \upsilon_y \\ \upsilon_z \end{Bmatrix} - \begin{Bmatrix} 0 \\ 0 \\ g \end{Bmatrix}$$

$$\boldsymbol{R}_{xyz}^T \left[\begin{Bmatrix} \alpha_\phi \\ \alpha_\theta \\ \alpha_\psi \end{Bmatrix}_{b/i} = \boldsymbol{I}_{cm}^{-1} \begin{Bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{Bmatrix} - \boldsymbol{I}_{cm}^{-1}\boldsymbol{\Omega}_{x'y'z'}\boldsymbol{I}_{cm} \begin{Bmatrix} \omega_\phi \\ \omega_\theta \\ \omega_\psi \end{Bmatrix}_{b/i} \right] \tag{A.8}$$

Note here that (in Equation A.8 especially) calculations are made in the body frame of reference and then transformed (through $\boldsymbol{R}_{xyz}^T$) for realization in the inertial frame. This is due to the nature of rotations not being realizable without two different frames expressed in relation to each other. As a result, the following control will be implemented in the body frame of reference and a transform to the inertial frame will be enforced before generating graphical representations of the vehicle, providing the perspective of an observer in the inertial frame.

Appendix B

Linearization

The equations of motion derived for a quadrotor vehicle (Equations A.5-A.6 and A.7-A.8) form two systems of non-linear functions. In order to autonomously operate the vehicle, either system must be organized in a manner in which it is possible to implement active control due to the fact that the system is inherently unstable. Although several methods for nonlinear control of quadrotor vehicles exist [100, 70, 69], linear controllers such as PID and LQR may be more common [101]. In order to implement a linear controller, the equations of motion must be linearized about a defined operating point. In this discussion the Taylor series expansion of the equations of motion will be used as the primary method of linearization. The general form of the Taylor series expansion is shown here:

$$\begin{aligned} F(x_1, x_2, x_3, ...) = F(a_1, a_2, a_3, ...) + \frac{\partial F(x_1, x_2, x_3, ...)}{\partial x_1} |_{x_1=a_1, x_2=a_2, ...} (x_1 - a_1) \\ + \frac{\partial F(x_1, x_2, x_3, ...)}{\partial x_2} |_{x_1=a_1, x_2=a_2, ...} (x_2 - a_2) + ... + HOT \end{aligned}$$

where a_i are the values of the function variables x_i at the operating point, and F is the function being linearized. In order to effectively linearize the equation the higher order terms (HOT) are neglected and the Taylor expansion becomes an approximation of F.

$$\begin{aligned} F(x_1,x_2,x_3,...) \approx F(a_1,a_2,a_3,...) &+ \frac{\partial F(x_1,x_2,x_3,...)}{\partial x_1} \mid_{x_1=a_1,x_2=a_2,...} (x_1-a_1) \\ &+ \frac{\partial F(x_1,x_2,x_3,...)}{\partial x_2} \mid_{x_1=a_1,x_2=a_2,...} (x_2-a_2) + ... \end{aligned} \tag{B.1}$$

$$\boldsymbol{F}(\boldsymbol{x}) \approx \boldsymbol{F}(\boldsymbol{a}) + \nabla \boldsymbol{F}(\boldsymbol{x}) \mid_{\boldsymbol{x}=\boldsymbol{a}} (\boldsymbol{x}-\boldsymbol{a})$$

For the quadrotor system an operating point will be chosen as a point when the vehicle is in a hovering state. In the body frame, the vehicle is motionless at the origin while in a hover. Therefore, assuming full state feedback on position, velocity, Tait-Bryan angles, and Tait-Bryan angular velocities, $\boldsymbol{x} = (x_1,...,x_{12}) = \left(x,y,z,\phi,\theta,\psi,\dot{x},\dot{y},\dot{z},\dot{\phi},\dot{\theta},\dot{\psi}\right)$ and $\boldsymbol{a} = (a_1,...,a_{12}) = 0$. As mention above, both Equations A.5 and A.6 in the body frame and Equations A.7 and A.8 in the inertial frame consist of six independent functions of the state variables, three translational and three rotational. Therefore,

$$\left\{\begin{array}{c} a_{x'} \\ a_{y'} \\ a_{z'} \\ \alpha_\phi \\ \alpha_\theta \\ \alpha_\psi \end{array}\right\} = \left\{\begin{array}{c} F_1(x,y,z,\phi,\theta,\psi,\dot{x},\dot{y},\dot{z},\dot{\phi},\dot{\theta},\dot{\psi}) \\ F_2(x,y,z,\phi,\theta,\psi,\dot{x},\dot{y},\dot{z},\dot{\phi},\dot{\theta},\dot{\psi}) \\ F_3(x,y,z,\phi,\theta,\psi,\dot{x},\dot{y},\dot{z},\dot{\phi},\dot{\theta},\dot{\psi}) \\ F_4(x,y,z,\phi,\theta,\psi,\dot{x},\dot{y},\dot{z},\dot{\phi},\dot{\theta},\dot{\psi}) \\ F_5(x,y,z,\phi,\theta,\psi,\dot{x},\dot{y},\dot{z},\dot{\phi},\dot{\theta},\dot{\psi}) \\ F_6(x,y,z,\phi,\theta,\psi,\dot{x},\dot{y},\dot{z},\dot{\phi},\dot{\theta},\dot{\psi}) \end{array}\right\} = \boldsymbol{F}(\boldsymbol{x})$$

the equations of motion may then be linearized using Equation B.1 one function at a time yielding six linear equations of motion for the system model. Using a linear algebra approach,

$$\begin{Bmatrix} a_{x'} \\ a_{y'} \\ a_{z'} \\ \alpha_{\phi} \\ \alpha_{\theta} \\ \alpha_{\psi} \end{Bmatrix} = \begin{bmatrix} \frac{\partial F_1(x,y,z,...)}{\partial x}\Big|_{x=0,y=0,...} & \frac{\partial F_1(x,y,z,...)}{\partial y}\Big|_{x=0,y=0,...} & \cdots \\ \frac{\partial F_2(x,y,z,...)}{\partial x}\Big|_{x=0,y=0,...} & \ddots & \\ \vdots & & \\ & & \\ & & \\ & & \end{bmatrix} \begin{Bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{Bmatrix} + \boldsymbol{F}(\mathbf{0})$$

$$\boldsymbol{F}(\mathbf{0}) = \begin{bmatrix} F_1(0,0,0,...) \\ F_2(0,0,0,...) \\ F_3(0,0,0,...) \\ F_4(0,0,0,...) \\ F_5(0,0,0,...) \\ F_6(0,0,0,...) \end{bmatrix}$$

Note that this representation is realized if the inputs are considered to be constant at the operating point (a hover in this case). However, a more technical representation in which this assumption is not necessary is:

$$\boldsymbol{F}(\boldsymbol{x},\boldsymbol{u}) \approx \boldsymbol{F}(\boldsymbol{a},\boldsymbol{u_a}) + \nabla\boldsymbol{F}(\boldsymbol{x},\boldsymbol{u})\mid_{\boldsymbol{x}=\boldsymbol{a},\boldsymbol{u}=\boldsymbol{u_a}} \left\{ \begin{array}{c} (\boldsymbol{x}-\boldsymbol{a}) \\ (\boldsymbol{u}-\boldsymbol{u_a}) \end{array} \right\}$$

In the body frame then via Equation B.1,

$$F_1(x',y',z',\phi,\theta,\psi,\upsilon_{x'},\upsilon_{y'},\upsilon_{z'},\dot{\phi},\dot{\theta},\dot{\psi}) = -\frac{D_{x'}\upsilon_{x'}}{m} + gsin(\theta)$$
$$-(-\dot{\phi}cos(\theta)sin(\psi)+\dot{\theta}cos(\psi))\upsilon_{z'} + (\dot{\phi}sin(\theta)+\dot{\psi})\upsilon_{y'}$$

$$\approx 0+0+0+0+0+(gcos(\theta)-\dot{\phi}sin(\theta)sin(\psi)\upsilon_{z'}+\dot{\phi}cos(\theta)\upsilon_{y'})\mid_{\boldsymbol{x}=0}(\theta)$$
$$+(\dot{\phi}cos(\theta)cos(\psi)\upsilon_{z'}-\dot{\theta}sin(\psi)\upsilon_{z'})\mid_{\boldsymbol{x}=0}(\psi)+(-\frac{D_{x'}}{m})\mid_{\boldsymbol{x}=0}(\upsilon_{x'})$$
$$+(\dot{\phi}sin(\theta)+\dot{\psi})\mid_{\boldsymbol{x}=0}(\upsilon_{y'})+(-\dot{\phi}cos(\theta)sin(\psi)+\dot{\theta}cos(\psi))\mid_{\boldsymbol{x}=0}(\upsilon_{z'})$$
$$+(cos(\theta)sin(\psi)\upsilon_{z'}+sin(\theta)\upsilon_{y'})\mid_{\boldsymbol{x}=0}(\dot{\phi})+(cos(\psi)\upsilon_{z'})\mid_{\boldsymbol{x}=0}(\dot{\theta})$$
$$+(\upsilon_{y'})\mid_{\boldsymbol{x}=0}(\dot{\psi})$$

$$\approx g\theta - \frac{D_{x'}}{m}\upsilon_{x'}$$

$$F_2(x',y',z',\phi,\theta,\psi,\upsilon_{x'},\upsilon_{y'},\upsilon_{z'},\dot{\phi},\dot{\theta},\dot{\psi}) = -\frac{D_{y'}\upsilon_{y'}}{m} - gcos(\theta)sin(\phi)$$
$$-(\dot{\phi}sin(\theta)+\dot{\psi})\upsilon_{x'}+(\dot{\phi}cos(\theta)cos(\psi)+\dot{\theta}sin(\psi))\upsilon_{z'}$$

$$\approx 0 + 0 + 0 + 0 + (-gcos(\theta)cos(\phi)) \mid_{\boldsymbol{x}=0} (\phi)$$
$$+ (gsin(\theta)sin(\phi) - \dot{\phi}cos(\theta)\upsilon_{x'} - \dot{\phi}sin(\theta)cos(\psi)\upsilon_{z'}) \mid_{\boldsymbol{x}=0} (\theta)$$
$$+ (-\dot{\phi}cos(\theta)sin(\psi)\upsilon_{z'} + \dot{\theta}cos(\psi)\upsilon_{z'}) \mid_{\boldsymbol{x}=0} (\psi)$$
$$+ (-\dot{\phi}sin(\theta) - \dot{\psi}) \mid_{\boldsymbol{x}=0} (\upsilon_{x'})$$
$$+ (-\frac{D_{y'}}{m}) \mid_{\boldsymbol{x}=0} (\upsilon_{y'}) + (\dot{\phi}cos(\theta)cos(\psi) + \dot{\theta}sin(\psi)) \mid_{\boldsymbol{x}=0} (\upsilon_{z'})$$
$$+ (-sin(\theta)\upsilon_{x'} + cos(\theta)cos(\psi)\upsilon_{z'}) \mid_{\boldsymbol{x}=0} (\dot{\phi})$$
$$+ (sin(\psi)\upsilon_{z'}) \mid_{\boldsymbol{x}=0} (\dot{\theta}) + (\upsilon_{x'}) \mid_{\boldsymbol{x}=0} (\dot{\psi})$$

$$\approx -g\phi - \frac{D_{y'}}{m}\upsilon_{y'}$$

$$F_3(x', y', z', \phi, \theta, \psi, \upsilon_{x'}, \upsilon_{y'}, \upsilon_{z'}, \dot{\phi}, \dot{\theta}, \dot{\psi}) = \frac{\sum k\varpi_i^2}{m} - \frac{D_{z'}\upsilon_{z'}}{m} - gcos(\theta)cos(\phi)$$
$$-(\dot{\phi}cos(\theta)cos(\psi) + \dot{\theta}sin(\psi))\upsilon_{y'} + (-\dot{\phi}cos(\theta)sin(\psi) + \dot{\theta}cos(\psi))\upsilon_{x'}$$

$$\approx \frac{\sum k\varpi_i^2}{m} - g + 0 + 0 + 0 + (gcos(\theta)sin(\phi)) \mid_{\boldsymbol{x}=0} (\phi)$$

$$+ (gsin(\theta)cos(\phi) + \dot{\phi}sin(\theta)cos(\psi)\upsilon_{y'} + \dot{\phi}sin(\theta)sin(\psi)\upsilon_{x'}) \mid_{\boldsymbol{x}=0} (\theta)$$

$$+ (\dot{\phi}cos(\theta)sin(\psi)\upsilon_{y'} - \dot{\theta}cos(\psi)\upsilon_{y'} + \dot{\phi}cos(\theta)cos(\psi)\upsilon_{z'}$$

$$- \dot{\theta}sin(\psi)\upsilon_{z'}) \mid_{\boldsymbol{x}=0} (\psi)$$

$$+ (\dot{\phi}cos(\theta)sin(\psi) + \dot{\theta}cos(\psi)) \mid_{\boldsymbol{x}=0} (\upsilon_{x'})$$

$$+ (-\dot{\phi}cos(\theta)cos(\psi) - \dot{\theta}sin(\psi)) \mid_{\boldsymbol{x}=0} (\upsilon_{y'})$$

$$+ (-\frac{D_{z'}}{m}) \mid_{\boldsymbol{x}=0} (\upsilon_{z'}) + (-cos(\theta)cos(\psi)\upsilon_{y'} + cos(\theta)sin(\psi)\upsilon_{x'}) \mid_{\boldsymbol{x}=0} (\dot{\phi})$$

$$+ (sin(\psi)\upsilon_{y'} + cos(\psi)\upsilon_{x'}) \mid_{\boldsymbol{x}=0} (\dot{\theta})$$

$$+ 0$$

$$\approx \frac{\sum k\varpi_i^2}{m} - g - \frac{D_{z'}}{m}\upsilon_{z'}$$

$$\begin{aligned} F_4(x', y', z', \phi, \theta, \psi, \upsilon_{x'}, \upsilon_{y'}, \upsilon_{z'}, \dot{\phi}, \dot{\theta}, \dot{\psi}) &= \frac{\tau_\phi}{I_x} - \left(\frac{I_z - I_y}{I_x}\right)\omega_\theta\omega_\psi \\ &= \frac{\tau_\phi}{I_x} + \left(\frac{I_z - I_y}{I_x}\right)(\dot{\phi}^2 cos(\theta)sin(\theta)sin(\psi) \\ &\quad -\dot{\phi}\dot{\theta}sin(\theta)cos(\psi) + \dot{\phi}\dot{\psi}cos(\theta)sin(\psi) \\ &\quad -\dot{\theta}\dot{\psi}cos(\psi)) \end{aligned}$$

$$\approx \frac{\tau_\phi}{I_x} + 0 + 0 + 0 + 0 + \left(\frac{I_z - I_y}{I_x}\right)(-\dot{\phi}^2 sin(\theta)sin(\theta)sin(\psi)$$

$$+ \dot{\phi}^2 cos(\theta)cos(\theta)sin(\psi) - \dot{\phi}\dot{\theta}cos(\theta)cos(\psi) - \dot{\phi}\dot{\psi}sin(\theta)sin(\psi)) \mid_{\boldsymbol{x}=0} (\theta)$$

$$+ \left(\frac{I_z - I_y}{I_x}\right)(\dot{\phi}^2 cos(\theta)sin(\theta)cos(\psi) + \dot{\phi}\dot{\theta}sin(\theta)sin(\psi)$$

$$+ \dot{\phi}\dot{\psi}cos(\theta)cos(\psi) + \dot{\theta}\dot{\psi}sin(\psi)) \mid_{\boldsymbol{x}=0} (\psi)$$

$$+ 0 + 0 + 0$$

$$+ \left(\frac{I_z - I_y}{I_x}\right)(2\dot{\phi}cos(\theta)sin(\theta)sin(\psi) - \dot{\theta}sin(\theta)cos(\psi)$$

$$+ \dot{\psi}cos(\theta)sin(\psi)) \mid_{\boldsymbol{x}=0} (\dot{\phi})$$

$$+ \left(\frac{I_z - I_y}{I_x}\right)(-\dot{\phi}sin(\theta)cos(\psi) - \dot{\psi}cos(\psi)) \mid_{\boldsymbol{x}=0} (\dot{\theta})$$

$$+ \left(\frac{I_z - I_y}{I_x}\right)(\dot{\phi}cos(\theta)sin(\psi) - \dot{\theta}cos(\psi)) \mid_{\boldsymbol{x}=0} (\dot{\psi})$$

$$\approx \frac{\tau_\phi}{I_x}$$

$$F_5(x', y', z', \phi, \theta, \psi, \upsilon_{x'}, \upsilon_{y'}, \upsilon_{z'}, \dot{\phi}, \dot{\theta}, \dot{\psi}) = \frac{\tau_\theta}{I_y} - \left(\frac{I_x - I_z}{I_y}\right)\omega_\phi \omega_\psi$$

$$= \frac{\tau_\theta}{I_y} + \left(\frac{I_x - I_z}{I_y}\right)(-\dot{\phi}^2 cos(\theta)sin(\theta)cos(\psi)$$

$$-\dot{\phi}\dot{\theta}sin(\theta)sin(\psi) - \dot{\phi}\dot{\psi}cos(\theta)cos(\psi)$$

$$-\dot{\theta}\dot{\psi}sin(\psi))$$

$$
\begin{aligned}
&\approx \frac{\tau_\theta}{I_y} + 0 + 0 + 0 + 0 + \left(\frac{I_x - I_z}{I_y}\right)(\dot{\phi}^2 sin(\theta)sin(\theta)cos(\psi) \\
&\quad - \dot{\phi}^2 cos(\theta)cos(\theta)cos(\psi) - \dot{\phi}\dot{\theta}cos(\theta)sin(\psi) + \dot{\phi}\dot{\psi}sin(\theta)cos(\psi))\ |_{\boldsymbol{x}=0}\ (\theta) \\
&\quad + \left(\frac{I_x - I_z}{I_y}\right)(\dot{\phi}^2 cos(\theta)sin(\theta)sin(\psi) - \dot{\phi}\dot{\theta}sin(\theta)cos(\psi) \\
&\quad + \dot{\phi}\dot{\psi}cos(\theta)sin(\psi) - \dot{\theta}\dot{\psi}cos(\psi))\ |_{\boldsymbol{x}=0}\ (\psi) \\
&\quad + 0 + 0 + 0 \\
&\quad + \left(\frac{I_x - I_z}{I_y}\right)(-2\dot{\phi}cos(\theta)sin(\theta)cos(\psi) - \dot{\theta}sin(\theta)sin(\psi) \\
&\quad - \dot{\psi}cos(\theta)cos(\psi))\ |_{\boldsymbol{x}=0}\ (\dot{\phi}) \\
&\quad + \left(\frac{I_x - I_z}{I_y}\right)(-\dot{\phi}sin(\theta)sin(\psi) - \dot{\psi}sin(\psi))\ |_{\boldsymbol{x}=0}\ (\dot{\theta}) \\
&\quad + \left(\frac{I_x - I_z}{I_y}\right)(-\dot{\phi}cos(\theta)cos(\psi) - \dot{\theta}sin(\psi))\ |_{\boldsymbol{x}=0}\ (\dot{\psi})
\end{aligned}
$$

$$
\approx \frac{\tau_\theta}{I_y}
$$

$$
\begin{aligned}
F_6(x', y', z', \phi, \theta, \psi, \upsilon_{x'}, \upsilon_{y'}, \upsilon_{z'}, \dot{\phi}, \dot{\theta}, \dot{\psi}) &= \frac{\tau_\psi}{I_z} - \left(\frac{I_y - I_x}{I_z}\right)\omega_\phi \omega_\psi \\
&= \frac{\tau_\psi}{I_z} + \left(\frac{I_y - I_x}{I_z}\right)(-\dot{\phi}^2 cos(\theta)cos(\theta)sin(\psi)cos(\psi) \\
&\quad -\dot{\phi}\dot{\theta}cos(\theta)sin(\psi)sin(\psi) + \dot{\phi}\dot{\theta}cos(\theta)cos(\psi)cos(\psi) \\
&\quad +\dot{\theta}^2 cos(\psi)sin(\psi))
\end{aligned}
$$

$$\approx \frac{\tau_\psi}{I_z} + 0 + 0 + 0 + 0 + \left(\frac{I_y - I_x}{I_z}\right)(2\dot{\phi}^2 sin(\theta)cos(\theta)sin(\psi)cos(\psi)$$

$$+ \dot{\phi}\dot{\theta}sin(\theta)sin(\psi)sin(\psi) - \dot{\phi}\dot{\theta}sin(\theta)sin(\psi)sin(\psi))\ |_{\boldsymbol{x}=0}\ (\theta)$$

$$+ \left(\frac{I_y - I_x}{I_z}\right)(-\dot{\phi}^2 cos(\theta)cos(\theta)cos(\psi)cos(\psi)$$

$$+ \dot{\phi}^2 cos(\theta)cos(\theta)sin(\psi)sin(\psi) + \dot{\theta}^2 cos(\psi)cos(\psi) - \dot{\theta}^2 sin(\psi)sin(\psi)$$

$$- 2\dot{\phi}\dot{\theta}cos(\theta)cos(\psi)sin(\psi) + 2\dot{\phi}\dot{\theta}cos(\theta)cos(\psi)sin(\psi))\ |_{\boldsymbol{x}=0}\ (\psi)$$

$$+ 0 + 0 + 0$$

$$+ \left(\frac{I_y - I_x}{I_z}\right)(-2\dot{\phi}cos(\theta)cos(\theta)cos(\psi)sin(\psi)$$

$$- \dot{\theta}cos(\theta)sin(\psi)sin(\psi) + \dot{\theta}cos(\theta)cos(\psi)cos(\psi))\ |_{\boldsymbol{x}=0}\ (\dot{\phi})$$

$$+ \left(\frac{I_y - I_x}{I_z}\right)(2\dot{\theta}cos(\psi)sin(\psi) - \dot{\phi}cos(\theta)sin(\psi)sin(\psi)$$

$$+ \dot{\phi}cos(\theta)cos(\psi)cos(\psi))\ |_{\boldsymbol{x}=0}\ (\dot{\theta})$$

$$+ 0$$

$$\approx \frac{\tau_\psi}{I_z}$$

Therefore, after linearizing, Equations A.5 and A.6 become:

$$\begin{Bmatrix} a_{x'} \\ a_{y'} \\ a_{z'} \end{Bmatrix} = \frac{1}{m}\begin{Bmatrix} 0 \\ 0 \\ \sum k\varpi_i^2 - mg \end{Bmatrix} - \frac{1}{m}\begin{Bmatrix} D_{x'}\upsilon_{x'} \\ D_{y'}\upsilon_{y'} \\ D_{z'}\upsilon_{z'} \end{Bmatrix} - \begin{Bmatrix} -g\theta \\ g\phi \\ 0 \end{Bmatrix} \tag{B.2}$$

$$\begin{Bmatrix} \alpha_\phi \\ \alpha_\theta \\ \alpha_\psi \end{Bmatrix}_{b/i} = \boldsymbol{I}_{cm}^{-1} \begin{Bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{Bmatrix} \tag{B.3}$$

A control strategy may now be formulated.

Appendix C

Investigation of Noise Model

In a closed loop control system, sensors are often used to generate feedback from the output as a means of calculating error in the system's state. A typical design process involves simulating control techniques through implementation on a digital system and injecting some type of noise profile into the system as random disturbances on the system plant or feedback sensor(s). In our work as we are interested in comparing different control techniques, noise is omitted from simulation as a means to more accurately compare the controllers themselves. However, in testing a simulated controller for implementation, perhaps the most simple case involves injecting a Gaussian noise profile into the system [102]. Here we investigate the noise profile for an indoor motion tracking system used for small UAS in an effort to create a statistical model which more accurately represents the noise profile for a specific physical system during flight. A more specific and physically related model may be beneficial in improving simulation accuracy and consequently control design via improved filtering and tuning.

When flying the quadrotor UAS indoors, a motion capture system may be used to track reflective markers mounted in a unique configuration on the vehicle itself. Alternatively, when flying outdoors the vehicle relies on GPS in relation to satellites. This is less accurate,

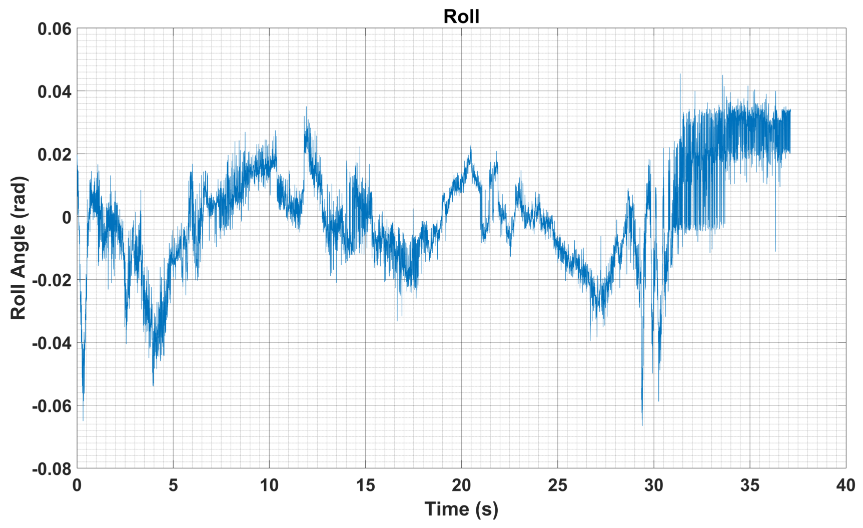


Figure C.1: Roll Angle motion capture system data for a flight with minimum control input of ~37sec in duration

within ~3 meters, and subject to random walk disturbances and otherwise. Here, we will be investigating data collected from a motion capture system as it is much more accurate and easily accessible. Data describing the vehicle's roll angle (rotation about the y-axis) relative to the world frame is collected from the motion capture system for a flight of approximately 37 seconds in duration with minimum control inputs and plotted in Figure C.1. Note that the data have been calibrated for sensor drift. Although only roll data are analyzed here, similar data relating to pitch and yaw angles as well as positions in x, y, and z may be analyzed in a similar fashion.

C.1 Data Dependent System Modeling

Modeling analysis often follows an approach similar to that of the previous section in which a mathematical formulation of differential equations (or similar discrete formulation) is conducted, almost always with the use of simplifying assumptions, to approximate the dynamics and response of the system. The level of accuracy, in comparison to nature, the derived model achieves is directly related to the assumptions made, and it follows that the derived response can only be an approximation of the natural one. Data Dependent System (DDS) modeling approaches a system with a methodology in the reverse direction so to speak. That is, DDS begins with data from the natural response of the system and statistically fits a difference/differential equation with a white noise (i.e. random disturbance) forcing function to the response. The order and number of parameters in the DDS model are determined by the least squares method in which the order of the difference/differential fit is increased until the residual sum of squares of errors (RSS) is minimized at some confidence interval [103, 104]. For the purposes of this report, the data will only be fit to an auto-regressive moving average (ARMA) model of second order regression and first order moving average (i.e. ARMA(2,1)); therefore, the reduction in the RSS may not be optimized. A DDS modeling and analysis program[1] was used to fit the ARMA(2,1) model and provide information for the following analysis.

C.1.1 ARMA Model Fit

Fitting the finite set of data points for roll angle consisting of N sampled data points, X_t, where t is time (a nominal index) and X_{t-j} refers to the previous point j time steps before X_t to the ARMA(2,1) difference equation yields

$$X_t - \phi_1 X_{t-1} - \phi_2 X_{t-2} = a_t - \theta_1 a_{t-1} \tag{C.1}$$

where a_t is the random error for measurement X_t, ϕ_j is the j^{th} auto-regressive parameter, and θ_j is the j^{th} moving average parameter. Note that as the motion capture system collects data at $200Hz$ the units on time steps t is 0.005 seconds. The DDS program internally fits the model in Equation C.1 to the data and calculates the ARMA parameters as well

[1] 'DDS Toolbox 2007' by Jason Dreyer available at https://www.mathworks.com/matlabcentral/fileexchange/19462-dds-toolbox-2007

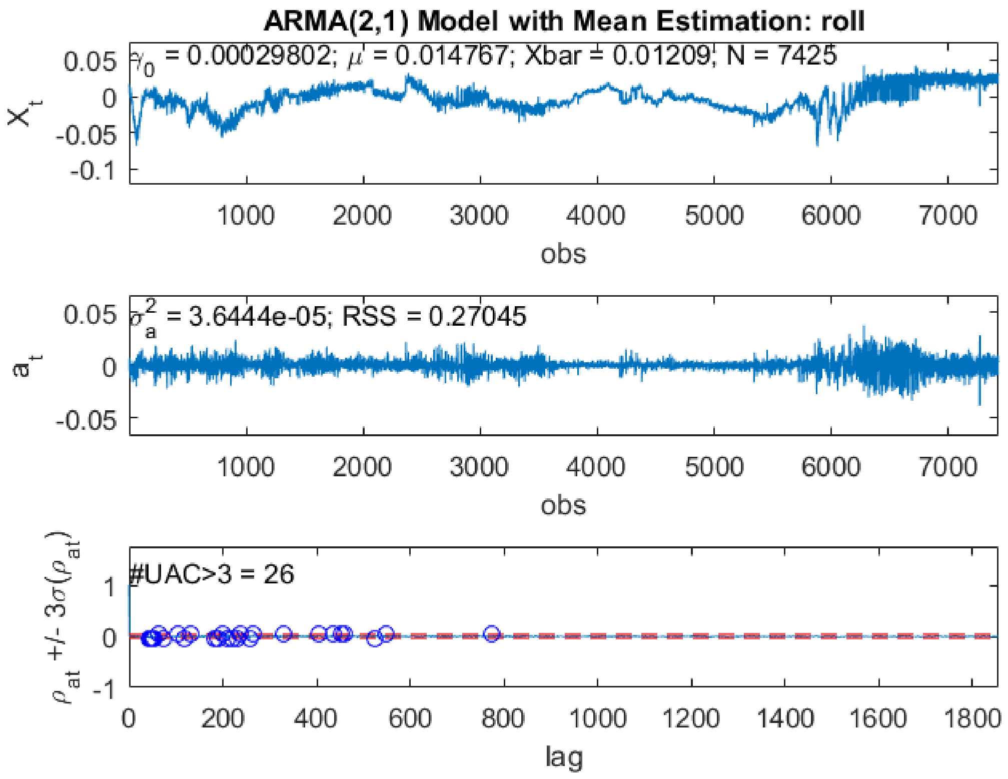


Figure C.2: DDS program results for ARMA(2,1) model fit to roll data. The results list a co-variance $\gamma_0 = 2.98 \times 10^{-4}$, mean $\mu = 0.0148$, average $\bar{X} = 0.0121$, variance $\sigma_a^2 = 3.64 \times 10^{-5}$, residual sum of squares RSS = 0.270, and the number of unified auto-correlations greater than 3 $\#UAC > 3 = 26$ for $N = 7425$ samples.

as several statistical properties of the data like mean, variance, etc. The results generated by the DDS program for the ARMA(2,1) model fit to the roll data shown in Figure C.1 are shown in Figure C.2. The parameters calculated by the DDS program are listed in Table C.1. Note that the number of unified auto-correlations greater than three $(\#UAC > 3)$ is not equal to zero, implying that the residuals of the ARMA(2,1) model are correlated in some way. This implication that the random error (a_t) for the model is not a linearly independent series alludes to the inaccuracy of the fit. Nonetheless for the purposes of this discussion we will continue with ARMA(2,1).

ARMA(2,1) Model Parameters

Number of samples (N):	7425	Mean (μ) :	0.0148
Degrees of freedom (dof):	7421	Average ($\bar{X}$):	0.0121
ϕ_1:	1.1432	Variance (σ_a^2):	3.644E-5
ϕ_2 :	-0.1464	Variance (γ_0):	2.980E-4
θ_1:	0.7946	Residual Sum of Squares (RSS):	0.2705
Number of auto-correlations greater than 3 ($\#UAC > 3$) :			26

Table C.1: Model and statistical parameters calculated by DDS program for ARMA(2,1) model of roll data.

C.1.2 Green's Function

By introducing the back-stepping operator, B, where $B^n X_t = X_{t-n}$, to Equation C.1,

$$(1 - \phi_1 B - \phi_2 B^2)X_t = (1 - \theta_1 B)a_t$$

and factoring and solving for X_t, we arrive at,

$$X_t = \frac{(1 - \theta_1 B)}{(1 - \lambda_1 B)(1 - \lambda_2 B)} a_t = \sum_{j=0}^{\infty} G_j a_{t-j} \tag{C.2}$$

where λ_i is the i^{th} characteristic root of the difference equation (Equation C.1) and G_j is the Green's function. The Green's function at discrete sample intervals, j, represents the weight in the system response given to the random disturbance a_t. That is, the Green's function captures the effect of a discrete unit impulse function at $j = 0$ on the response of the system as time progresses.

C.1.3 Frequency Analysis

The DDS program also estimates properties of a given difference equation's continuous counterpart. That is, a second order difference equation with a non-zero, first order forcing function (i.e. ARMA(2,1)) can be used to also estimate properties like natural frequency, damping ratio, and power contribution (co-variance) for a corresponding continuous system model (i.e. a spring, mass, damper system).

$$\frac{d^2x}{dt^2} + 2\zeta\omega_n \frac{dx}{dt} + \omega_n^2 x = f(t) \tag{C.3}$$

This is accomplished by using the characteristic roots of the difference equation, λ_1 and λ_2 from Equation C.2, to approximate the characteristic roots of the homogeneous solution to the corresponding differential equation. This conversion from the discrete to continuous time domain is made using the principle of co-variance which states that because co-variance is the difference in values of a regression model at two points in time, it is the same whether the model is a discrete or continuous one. Therefore, we can directly compare the discrete co-variance, γ_k, and the continuous co-variance, $\gamma(s)$, where $s = \triangle k$, and s and $\triangle k$ are the continuous lag and the discrete lag respectively. Since the co-variance in both cases is a function of the characteristic roots of their corresponding discrete and continuous models, the continuous roots can be calculated and used to find the damping ratio, ζ, and natural frequency, ω_n, of the system. For the ARMA(2,1) model fit to the roll data discussed above, the DDS program calculates a damping ratio of $\zeta = 1$ and a natural frequency of $\omega_n = 5.98 \times 10^{-4}$. It is again important to note that although the ARMA(2,1) is fit for the purposes of this paper it may not be the best fit as the RSS was not optimized.

C.2 Noise Modeling

Now that we have a differential model approximated from the roll data itself, which includes system noise, we can compare it to our derivation in Appendix A and B. In terms of roll angle (ϕ) we have,

$$\frac{d^2\phi}{dt^2} = \frac{\tau_\phi}{I_{cm}} \tag{C.4}$$

Note that according to the analytical derivation, in which there is no noise, $\omega_n = 0$ and we have no information on the damping ratio, ζ. In comparison, (note for the following comparison x in Equation C.3 is equal to roll angle, ϕ) the DDS model coefficients on the $\frac{dx}{dt}$ and x terms are significantly small, 1.2×10^{-3} and 3.58×10^{-7} respectively. Perhaps the fact that the coefficient of the first derivative may not be of a negligible order of magnitude, but is still significantly small, alludes to the effects of linearizing the systems equations of motion. That is, the nonlinear dynamics may not be negligible to the motion of the vehicle but are dominated by linear effects. By adding a noise profile, $\eta(t)$, to Equation C.4,

$$\frac{d^2\phi}{dt^2} + \eta(t) = f(t)$$

and equating it with Equation C.3 we get the difference between our control law and empirical measurements as

$$\eta(t) = 2\zeta\omega_n \frac{d\phi}{dt} + \omega_n^2 \phi$$

which we can then inject into our control framework as noise. Noise profiles for other state feedback variables in our control architecture may also be realized in a similar fashion. DDS modeling analysis of the motion capture system positional data along the x-axis of the world frame yields $\zeta_x = -1$ and $\omega_{n,x} = 1.17 \times 10^{-6}$ and from Equation A.7,

$$\frac{d^2x}{dt^2} + \frac{D_x}{m}\frac{dx}{dt} = \frac{T_{net,x}}{m}$$

or

$$\frac{d^2x}{dt^2} + \frac{D_x}{m}\frac{dx}{dt} + \eta_x(t) = f(t).$$

In comparison with Equation C.3, we see that

$$\eta_x(t) = (2\zeta_x\omega_{n,x} - \frac{D_x}{m})\frac{dx}{dt} + \omega_{n,x}^2 x.$$

Results of this comparison show a very small coefficient on the x term (i.e. very little noise, $\sim 10^{-12}$meters) but can be used to approximate the drag coefficient of the vehicle along the x-axis as $D_x = 2m\zeta_x\omega_{n,x} \approx -1.2 \times 10^{-6}\frac{kg}{s}$.

Here we have modeled each control variable and its corresponding linear dynamics as a single input single output model, and in doing so, we were unable to reach a model in which the residual error was a linearly independent function of time. It follows that since the quadrotor system is more accurately a nonlinear, multiple input - multiple output system, multiple regression analysis coupled with a nonlinear analytical model may provide more accurate results.

Bibliography

[1] W. Wolf, “Cyber-physical Systems,” Computer, vol. 42, no. 3, pp. 88–89, 2009.

[2] R. R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, “Cyber-physical systems: the next computing revolution,” in Proceedings of the 47th Design Automation Conference. ACM, 2010, pp. 731–736.

[3] J. M. Bradley and E. M. Atkins, “Coupled Cyber-Physical System Modeling and Coregulation of a CubeSat,” IEEE Transactions on Robotics, vol. 31, no. 2, pp. 443–456, Apr 2015.

[4] W. Heemels, K. H. Johansson, and P. Tabuada, “An introduction to event-triggered and self-triggered control,” in Decision and Control (CDC), 2012 IEEE 51st Annual Conference on. IEEE, 2012, p. 3270âĂŞ3285.

[5] E. Bini and G. M. Buttazzo, “The optimal sampling pattern for linear control systems,” Automatic Control, IEEE Transactions on, vol. 59, no. 1, p. 78âĂŞ90, Jan. 2014.

[6] K. Kowalska and M. Mohrenschildt, “An approach to variable time receding horizon control,” Optimal Control Applications and Methods, vol. 33, no. 4, p. 401âĂŞ414, 2012.

[7] J. M. Bradley and E. M. Atkins, “Toward Continuous State-Space Regulation of Coupled Cyber-Physical Systems,” Proceedings of the IEEE, vol. 100, no. 1, pp. 60–74, Jan 2012.

[8] R. Murphy, Introduction to AI Robotics. MIT press, 2000.

[9] R. W. Beard and T. W. McLain, Small Unmanned Aircraft: Theory and Practice. Princeton University Press, 2012.

[10] L. H. Keel and S. P. Bhattacharyya, “Stability margins and digital implementation of controllers,” in Digital Controller Implementation and Fragility. Springer, 2001, pp. 13–24. [Online]. Available: http://link.springer.com/chapter/10.1007/978-1-4471-0265-6_2

[11] J. Liu, Real-Time Systems. Prentice Hall, 2000, lCCB: 99051522.

[12] F. Zhang, K. Szwaykowska, W. Wolf, and V. Mooney, “Task Scheduling for Control Oriented Requirements for Cyber-Physical Systems,” in Real-Time Systems Symposium, 2008. IEEE, 2008, pp. 47–56.

[13] D. Gurdan, J. Stumpf, M. Achtelik, K. M. Doth, G. Hirzinger, and D. Rus, “Energy-efficient Autonomous Four-rotor Flying Robot Controlled at 1 kHz,” in Proceedings 2007 IEEE International Conference on Robotics and Automation, Apr. 2007, pp. 361–366.

[14] N. Wiener, Cybernetics or Control and Communication in the Animal and the Machine. MIT press, 1965, vol. 25.

[15] “Cyber-Physical Systems (CPS) (nsf17529) | NSF - National Science Foundation.” [Online]. Available: https://www.nsf.gov/pubs/2017/nsf17529/nsf17529.htm

[16] K. J. Åström and B. M. Bernhardsson, “Comparison of riemann and lebesgue sampling for first order stochastic systems,” in Decision and Control, 2002, Proceedings of the 41st IEEE Conference on, vol. 2. IEEE, 2002, p. 2011âĂŞ2016.

[17] K. J. Åström and B. Wittenmark, Computer-controlled systems: theory and design. Prentice-Hall New York, 1984.

[18] G. F. Franklin, M. L. Workman, and D. Powell, Digital Control of Dynamic Systems. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1998.

[19] P. Tabuada, “Event-triggered real-time scheduling of stabilizing control tasks,” Automatic Control, IEEE Transactions on, vol. 52, no. 9, p. 1680âĂŞ1685, 2007.

[20] C. S. Draper, Inertial Guidance. Pergamon Press, 1960.

[21] H. Kopetz, “Should responsive systems be event-triggered or time-triggered?” IEICE Transactions on Information and systems, vol. 76, no. 11, pp. 1325–1332, 1993. [Online]. Available: http://search.ieice.org/bin/summary.php?id=e76-d_11_1325

[22] J. F. Guerrero-Castellanos, J. J. Téllez-Guzmán, S. Durand, N. Marchand, and J. U. Alvarez-Muñoz, “Event-triggered nonlinear control for attitude stabilization of a quadrotor,” in Unmanned Aircraft Systems (ICUAS), 2013 International Conference on, May 2013, pp. 584–591.

[23] H. Voit, A. Annaswamy, R. Schneider, D. Goswami, and S. Chakraborty, “Adaptive switching controllers for systems with hybrid communication protocols,” in American Control Conference (ACC), 2012. IEEE, 2012, p. 4921âĂŞ4926.

[24] S. L. Osburn and D. S. Bernstein, “An exact treatment of the achievable closed-loop H_2 performance of sampled-data controllers: From continuous-time to open-loop,” Automatica, vol. 31, no. 4, p. 617âĂŞ620, 1995.

[25] T. F. Abdelzaher, E. M. Atkins, and K. G. Shin, “QoS Negotiation in Real-Time Systems and Its Application to Automated Flight Control,” Computers, IEEE Transactions on, vol. 49, no. 11, pp. 1170–1183, 2000.

[26] F. Xia, L. Ma, J. Dong, and Y. Sun, “Network QoS management in cyber-physical systems,” in Embedded Software and Systems Symposia, 2008. ICESS Symposia’08. International Conference on. IEEE, 2008, p. 302âĂŞ307.

[27] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu, “A survey of recent results in networked control systems,” Proc. of the IEEE, vol. 95, no. 1, p. 138âĂŞ162, 2007.

[28] W. Zhang, M. S. Branicky, and S. M. Phillips, “Stability of networked control systems,” IEEE Control Systems Magazine, vol. 21, no. 1, p. 84âĂŞ99, 2001.

[29] X. Cao, P. Cheng, J. Chen, and Y. Sun, “An online optimization approach for control and communication codesign in networked cyber-physical systems,” Industrial Informatics, IEEE Transactions on, vol. 9, no. 1, pp. 439–450, 2013.

[30] D. V. Dimarogonas, E. Frazzoli, and K. H. Johansson, “Distributed event-triggered control for multi-agent systems,” Automatic Control, IEEE Transactions on, vol. 57, no. 5, pp. 1291–1297, 2012.

[31] A. Eqtami, D. V. Dimarogonas, and K. J. Kyriakopoulos, “Event-triggered control for discrete-time systems,” in American Control Conference (ACC), 2010. IEEE, 2010, pp. 4719–4724.

[32] L. Shi, P. Cheng, and J. Chen, "Sensor data scheduling for optimal state estimation with communication energy constraint," Automatica, vol. 47, no. 8, pp. 1693–1698, 2011.

[33] A. V. Savkin, R. J. Evans, and E. Skafidas, "The problem of optimal robust sensor scheduling," Systems & Control Letters, vol. 43, no. 2, pp. 149–157, 2001.

[34] M. P. Vitus, W. Zhang, A. Abate, J. Hu, and C. J. Tomlin, "On efficient sensor scheduling for linear dynamical systems," Automatica, vol. 48, no. 10, pp. 2482–2493, 2012.

[35] P. Cheng, J. Keller, and V. Kumar, "Time-optimal uav trajectory planning for 3d urban structure coverage," in Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on. IEEE, 2008, pp. 2750–2757.

[36] R. N. De Carvalho, H. Vidal, P. Vieira, and M. Ribeiro, "Complete coverage path planning and guidance for cleaning robots," in Industrial Electronics, 1997. ISIE'97., Proceedings of the IEEE International Symposium on, vol. 2. IEEE, 1997, pp. 677–682.

[37] F. Bonin-Font, A. Ortiz, and G. Oliver, "Visual navigation for mobile robots: A survey," Journal of intelligent and robotic systems, vol. 53, no. 3, p. 263, 2008.

[38] J. J. Ruz, G. Pajares, M. Jesus, and O. Arevalo, UAV trajectory planning for static and dynamic environments. INTECH Open Access Publisher, 2009.

[39] T. McLain and R. Beard, "Trajectory planning for coordinated rendezvous of unmanned air vehicles," in AIAA Guidance, Navigation, and Control Conference and Exhibit, 2000, p. 4369.

[40] F. Kunwar and B. Benhabib, "Rendezvous-guidance trajectory planning for robotic dynamic obstacle avoidance and interception," IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 36, no. 6, pp. 1432–1441, 2006.

[41] R. C. Arkin, "Motor schema-based mobile robot navigation," The International journal of robotics research, vol. 8, no. 4, pp. 92–112, 1989.

[42] I. Ohya, A. Kosaka, and A. Kak, "Vision-based navigation by a mobile robot with obstacle avoidance using single-camera vision and ultrasonic sensing," IEEE Transactions on Robotics and Automation, vol. 14, no. 6, pp. 969–978, 1998.

[43] Y. Watanabe, A. Calise, and E. Johnson, "Vision-based obstacle avoidance for uavs," in AIAA Guidance, Navigation and Control Conference and Exhibit, 2007, p. 6829.

[44] J. J. Craig, Introduction to robotics: mechanics and control. Pearson Prentice Hall Upper Saddle River, 2005, vol. 3.

[45] A. Piazzi and A. Visioli, "Global minimum-jerk trajectory planning of robot manipulators," IEEE transactions on industrial electronics, vol. 47, no. 1, pp. 140–149, 2000.

[46] S. Macfarlane and E. A. Croft, "Jerk-bounded manipulator trajectory planning: design for real-time applications," IEEE Transactions on Robotics and Automation, vol. 19, no. 1, pp. 42–52, 2003.

[47] A. Gasparetto and V. Zanotto, "A new method for smooth trajectory planning of robot manipulators," Mechanism and machine theory, vol. 42, no. 4, pp. 455–471, 2007.

[48] E. P. Anderson, R. W. Beard, and T. W. McLain, "Real-time dynamic trajectory smoothing for unmanned air vehicles," IEEE Transactions on Control Systems Technology, vol. 13, no. 3, pp. 471–477, 2005.

[49] D. R. Nelson, D. B. Barber, T. W. McLain, and R. W. Beard, "Vector field path following for miniature air vehicles," IEEE Transactions on Robotics, vol. 23, no. 3, pp. 519–529, 2007.

[50] ——, "Vector field path following for small unmanned air vehicles," in American Control Conference, 2006. IEEE, 2006, pp. 7–pp.

[51] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," American Journal of mathematics, vol. 79, no. 3, pp. 497–516, 1957.

[52] M. Owen, R. W. Beard, and T. W. McLain, "Implementing dubins airplane paths on fixed-wing uavs," in Handbook of Unmanned Aerial Vehicles. Springer, 2015, pp. 1677–1701.

[53] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in Robotics and Automation (ICRA), 2011 IEEE International Conference on. IEEE, 2011, pp. 2520–2525.

[54] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," The International Journal of Robotics Research, vol. 31, no. 5, pp. 664–674, 2012.

[55] Y. Bouktir, M. Haddad, and T. Chettibi, "Trajectory planning for a quadrotor helicopter," in Control and Automation, 2008 16th Mediterranean Conference on. Ieee, 2008, pp. 1258–1263.

[56] M. Hehn and R. D'Andrea, "Quadrocopter trajectory generation and control," IFAC Proceedings Volumes, vol. 44, no. 1, pp. 1485–1491, 2011.

[57] O. Purwin and R. D'Andrea, "Trajectory generation and control for four wheeled omnidirectional vehicles," Robotics and Autonomous Systems, vol. 54, no. 1, pp. 13–22, 2006.

[58] G. Hoffmann, S. Waslander, and C. Tomlin, "Quadrotor helicopter trajectory tracking control," in AIAA guidance, navigation and control conference and exhibit, 2008, p. 7410.

[59] "AscTec Research UAVs." [Online]. Available: http://www.asctec.de/en/uav-uas-drones-rpas-roav/asctec-hummingbird/

[60] N. S. Nise, Control systems engineering, 6th ed. John Wiley and Sons, 2011.

[61] P. Castillo, P. Albertos, P. Garcia, and R. Lozano, "Simple real-time attitude stabilization of a quad-rotor aircraft with bounded signals," in Proceedings of the 45th IEEE Conference on Decision and Control, Dec. 2006, pp. 1533–1538.

[62] J. F. Guerrero-Castellanos, N. Marchand, A. Hably, S. Lesecq, and J. Delamare, "Bounded attitude control of rigid bodies: Real-time experimentation to a quadrotor mini-helicopter," Control Engineering Practice, vol. 19, no. 8, pp. 790 – 797, 2011.

[63] Corona-Sánchez, J. J. and Rodríguez-Cortés, H., "Experimental real-time validation of an attitude nonlinear controller for the quadrotor vehicle," in Unmanned Aircraft Systems (ICUAS), 2013 International Conference on, May 2013, pp. 453–460.

[64] S. Seghour, M. Bouchoucha, and H. Osmani, "From integral backstepping to integral sliding mode attitude stabilization of a quadrotor system: Real time implementation

on an embedded control system based on a dspic μc," in Mechatronics (ICM), 2011 IEEE International Conference on, Apr. 2011, pp. 154–161.

[65] A. Shankar, S. Doebbeling, and J. Bradley, "Toward a cyber-physical quadrotor: Characterizing trajectory following performance," in International Conference on Unmanned Aircraft Systems. IEEE, June 2017.

[66] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," IEEE robotics & automation magazine, vol. 19, no. 3, pp. 20–32, 2012.

[67] R. Beard, "Quadrotor dynamics and control rev 0.1," 2008.

[68] G. Hoffmann, H. Huang, S. Waslander, and C. Tomlin, "Quadrotor helicopter flight dynamics and control: Theory and experiment," in AIAA Guidance, Navigation and Control Conference and Exhibit, 2007, p. 6461.

[69] S. Bouabdallah and R. Siegwart, "Backstepping and sliding-mode techniques applied to an indoor micro quadrotor," in Proceedings of the 2005 IEEE international conference on robotics and automation. IEEE, 2005, pp. 2247–2252.

[70] R. Xu and U. Ozguner, "Sliding mode control of a quadrotor helicopter," in Proceedings of the 45th IEEE Conference on Decision and Control. IEEE, 2006, pp. 4957–4962.

[71] A. A. Mian and W. Daobo, "Modeling and backstepping-based nonlinear control strategy for a 6 DOF quadrotor helicopter," Chinese Journal of Aeronautics, vol. 21, no. 3, pp. 261–268, 2008.

[72] F. Sabatino, "Quadrotor control: modeling, nonlinearcontrol design, and simulation," 2015.

[73] P. Sujit, S. Saripalli, and J. B. Sousa, "An evaluation of uav path following algorithms," in Control Conference (ECC), 2013 European. IEEE, 2013, pp. 3332–3337.

[74] J. M. Bradley and E. M. Atkins, "Coupled cyber-physical system modeling and coregulation of a CubeSat," IEEE Transactions on Robotics, vol. 31, no. 2, p. 443âĂŞ456, Apr. 2015.

[75] N. Kreciglowa, V. KaKumar, and V. Kumar, "Energy efficiency of trajectory generation methods for stop-and-go aerial robot navigation," in Interational Conference on Unmanned Aircraft Systems, IEEE, Ed., 2017.

[76] T.-T. Lee and S.-H. Lee, "Discrete optimal control with eigenvalue assigned inside a circular region," IEEE Transactions on Automatic Control, vol. 31, no. 10, pp. 958–962, October 1986.

[77] R. Wilhelm, T. Mitra, F. Mueller, I. Puaut, P. Puschner, J. Staschulat, P. Stenström, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, and R. Heckmann, "The worst-case execution-time problem - overview of methods and survey of tools," ACM Transactions on Embedded Computing Systems, vol. 7, no. 3, pp. 1–53, Apr. 2008.

[78] K. G. Shin and X. Cui, "Computing time delay and its effects on real-time control systems," IEEE Transactions on control systems technology, vol. 3, no. 2, pp. 218–224, 1995.

[79] F. Yang, Z. Wang, and Y. S. Hung, "Robust Kalman filtering for discrete time-varying uncertain systems with multiplicative noises," IEEE Transactions on Automatic Control, vol. 47, no. 7, pp. 1179–1183, 2002. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/1017567/

[80] A. Weiss, I. V. Kolmanovsky, M. Baldwin, R. S. Erwin, and D. S. Bernstein, “Forward-integration riccati-based feedback control for spacecraft rendezvous maneuvers on elliptic orbits,” in CDC, 2012, p. 1752âĂŞ1757.

[81] M. Faessler, D. Falanga, and D. Scaramuzza, “Thrust Mixing, Saturation, and Body-Rate Control for Accurate Aggressive Quadrotor Flight,” IEEE Robotics and Automation Letters, 2016.

[82] G. J. Leishman, Principles of helicopter aerodynamics with CD extra. Cambridge university press, 2006.

[83] R. D. Eubank, J. M. Bradley, and E. M. Atkins, “Energy-aware multiflight planning for an unattended seaplane: Flying fish,” Journal of Aerospace Information Systems, pp. 1–19, 2016.

[84] M. L. Puterman, Markov decision processes: discrete stochastic dynamic programming. John Wiley & Sons, 2014.

[85] C. Goerzen, Z. Kong, and B. Mettler, “A survey of motion planning algorithms from the perspective of autonomous uav guidance,” in Selected papers from the 2nd International Symposium on UAVs, Reno, Nevada, USA June 8–10, 2009. Springer, 2009, pp. 65–100.

[86] H. Huang, G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin, “Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering,” in Robotics and Automation, 2009. ICRA’09. IEEE International Conference on. IEEE, 2009, pp. 3277–3282.

[87] T. Kunz and M. Stilman, “Turning paths into trajectories using parabolic blends,” Georgia Institute of Technology, Tech. Rep., 2011.

[88] M. Reyhanoglu, A. van der Schaft, N. H. McClamroch, and I. Kolmanovsky, “Dynamics and control of a class of underactuated mechanical systems,” IEEE Transactions on Automatic Control, vol. 44, no. 9, pp. 1663–1671, 1999.

[89] A. P. Aguiar and J. P. Hespanha, “Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty,” IEEE Transactions on Automatic Control, vol. 52, no. 8, pp. 1362–1379, 2007.

[90] Z. Zuo, “Trajectory tracking control design with command-filtered compensation for a quadrotor,” IET control theory & applications, vol. 4, no. 11, pp. 2343–2355, 2010.

[91] E. Frazzoli, M. A. Dahleh, and E. Feron, “Trajectory tracking control design for autonomous helicopters using a backstepping algorithm,” in American Control Conference, 2000. Proceedings of the 2000, vol. 6. IEEE, 2000, pp. 4102–4107.

[92] T. Madani and A. Benallegue, “Control of a quadrotor mini-helicopter via full state backstepping technique,” in Decision and Control, 2006 45th IEEE Conference on. IEEE, 2006, pp. 1515–1520.

[93] T. Lee, M. Leoky, and N. H. McClamroch, “Geometric tracking control of a quadrotor uav on se (3),” in Decision and Control (CDC), 2010 49th IEEE Conference on. IEEE, 2010, pp. 5420–5425.

[94] D. Liberzon and A. S. Morse, “Basic problems in stability and design of switched systems,” IEEE Control systems, vol. 19, no. 5, pp. 59–70, 1999.

[95] J. C. Geromel and P. Colaneri, “Stability and stabilization of discrete time switched systems,” International Journal of Control, vol. 79, no. 07, pp. 719–728, 2006.

[96] J. Daafouz, P. Riedinger, and C. Iung, "Stability analysis and control synthesis for switched systems: a switched lyapunov function approach," IEEE transactions on automatic control, vol. 47, no. 11, pp. 1883–1887, 2002.

[97] S. Thornton and J. Marion, Classical Dynamics of Particles and Systems 5th edn (Belmont, CA: Brooks/Cole), 2004.

[98] M. R. Spiegel, S. Lipschutz, and J. Liu, "Mathematical handbook of formulas and tables," 1968.

[99] A. Gibiansky, "Quadcopter dynamics, simulation, and control," 2010.

[100] A. A. Mian and W. Daobo, "Nonlinear flight control strategy for an underactuated quadrotor aerial robot," in Networking, Sensing and Control, 2008. ICNSC 2008. IEEE International Conference on. IEEE, 2008, pp. 938–942.

[101] J. Li and Y. Li, "Dynamic analysis and pid control for a quadrotor," in 2011 IEEE International Conference on Mechatronics and Automation. IEEE, 2011, pp. 573–578.

[102] S. Bouabdallah, P. Murrieri, and R. Siegwart, "Design and control of an indoor micro quadrotor," in Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on, vol. 5. IEEE, 2004, pp. 4393–4398.

[103] K. P. Rajurkar and J. Nissen, "Data-dependent systems approach to short-term load forecasting," IEEE transactions on systems, man, and cybernetics, no. 4, pp. 532–536, 1985.

[104] K. Rajurkar, S. Pandit, and W. Wittig, "Pulse current signal as a sensor for on-line computer control of edm." in Manufacturing Engineering Transactions. North American Manufacturing Research Inst of SME, 00001400, 1983.