Payload Position Tracking and Fractional Control Evaluation for a Drone-based Ground Penetrating Radar System

by

Mitesh Patel

A thesis
submitted to the University of Manitoba
in fulfilment of the
thesis requirement for the degree of
Master of Science
in
Mechanical Engineering

Winnipeg, Manitoba, Canada 2023

Abstract

Remote sensing technology is becoming a standard tool for Arctic studies to understand and preserve vulnerable ecosystems. A Ground Penetrating Radar (GPR) is a common tool used to study the change in ice properties, which can help reduce climate change effects. However, due to the GPR's large size and the Arctic's remote terrain, it is difficult to navigate GPRs in the Arctic. Flying the GPR using a drone provides a solution, but it requires the GPR to be suspended from the drone which can affect the drone's stability through the non-linear sway of the GPR. The non-linear effects may not be well captured by linear controllers and therefore require the implementation of a non-linear controller on the drone. In this thesis, I use a Light Detection and Ranging (LiDAR) sensor and an Extended Kalman filter to measure the position of the payload relative to the drone with an accuracy of 2 cm. Additionally, I design a feedback control system for the drone carrying a payload to compare the performance of a feedback integer-order Proportional-Integral-Derivative (IOPID) controller and fractional-order PID (FOPID) controller to minimize the sway of the payload. Results from simulation and experimental tests indicate that if optimally tuned, both the IOPID controller and FOPID controller (using the Oustaloup recursive filter) had a comparable performance for a non-linear drone-based cable-suspended system. Neither feedback controller provided the optimal performance to control the sway of the payload and a model-based control technique may be a better solution.

Acknowledgements

I would like to thank my advisor, Dr. Philip Ferguson, for his guidance and support throughout the years. You have been a great mentor and friend to me, and I greatly appreciate the role you have played in my personal and professional development. I would also like to thank my friends and colleagues at STARLab, especially Aref Asgari, Ali Barari, Andrew Bowman, Brendan Pachal, Jaime Campos, Morgan May, and Yujia Huang. Thank you for your help and support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation[1]

Remote sensing research for civilian and industrial applications is gaining attention as drones provide access to remote and inaccessible terrain [3] such as forest regions and the Arctic. Understanding the soil moisture content in forests provides insight into the temporal and spatial variations of climate, hydrological and ecological models [4]. This facilitates improved weather forecasting, agricultural practices, and crop yield. As for Arctic or ice research, understanding changes in ice properties and quantity can help preserve the most vulnerable ecosystems impacted by changes in climate [5, 6]. Furthermore, Arctic studies provide Inuit and Indigenous communities with data to determine ice safety as they rely on ice for transportation and hunting [7].

[1] This section has been reproduced with permission from:

[1] Patel, M., and Ferguson, P., "Tracking and Estimation of a Swaying Payload Using a LiDAR and an Extended Kalman Filter," 2021 IEEE ROSE. ©2021 Mitesh Patel and Philip Ferguson.

[2] Mitesh S. Patel and Philip A. Ferguson. "Drone-Based Cable-Suspended Payload Tracking and Estimation Using Simulated LiDAR Measurements and an Extended Kalman Filter," AIAA 2022-2290. AIAA SCITECH 2022 Forum. ©2022 Mitesh Patel and Philip Ferguson.

A Ground Penetrating Radar (GPR) is a key instrument in soil moisture research (Figure 1.1) and Arctic research (Figure 1.2) when applied to climate research. A GPR is a tool that uses radio frequencies to image deep under a surface, therefore mapping underground features. The recorded data can be used for feature detection or ice property determination by retrieving information on brine content in the ice [8]. Moreover, with the high transparency of ice to radio wave signals, GPRs can be used to find metallic objects under the ice, image ice sheets, make avalanche predictions, and measure the snow to ground boundary which can be used to forecast freshwater supply [9, 10].

Figure 1.1: GPR used for agricultural surveys [11].

Figure 1.2: Measuring ice thickness for roads [9].

Despite their capabilities, GPRs are difficult to use for such studies since they are bulky and need multiple components to operate, as shown in Figure 1.3. This may not be an issue for flat open ground, but in less hospitable environments such as the Arctic, navigating the equipment through the terrain can be slow and dangerous.

Advances in drone technology provide a simpler and more flexible solution to accessing remote conditions [13]. However, integrating GPRs and drones introduces new challenges. Notably, GPRs need to be less than one meter off the ground for proper functionality, since drones do not fly well close to the ground surface due to In-Ground Effect (IGE) turbulence, which introduces instability in the drone [14]. Therefore, the GPR needs to be suspended from a long cable attached to the drone, as shown in Figure 1.4.

Figure 1.3: Human pulling GPR in the cold and difficult Arctic terrain [12].

Large sway angles of the suspended GPR can result in a non-linear motion of the GPR that may lead to an unstable flight. Additionally, the unstable motion of the GPR may impact the quality of the collected data [15]. Thus, maintaining a stable flight is key from a flight stability standpoint as well as the quality of the data collected by the GPR.

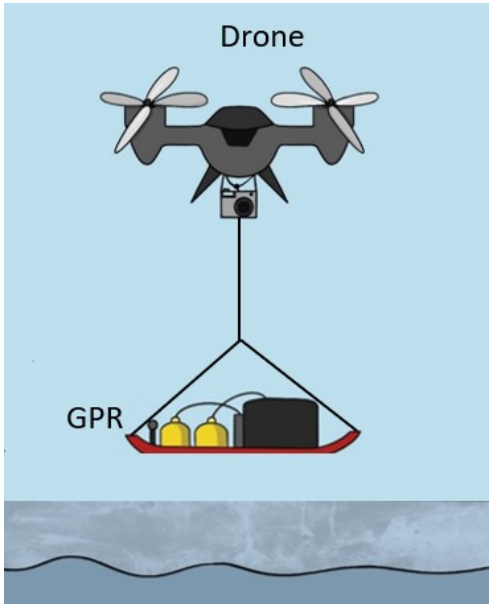


Figure 1.4: Schematic of a GPR suspended via a cable attached to a drone.

1.2 Research Objective

The objective of this research was to evaluate the effectiveness of fractional control for controlling drones with a swaying payload. To achieve this goal, this thesis is divided into two key components: an onboard tracking mechanism to track the location of the suspended payload, and a robust and quick-to-respond controller that can handle the non-linearity of the swaying payload.

Stable flight and accurate control of the payload require a tracking mechanism to monitor the suspended payload's position. This research explores tracking the payload's position using a Light Detection and Ranging (LiDAR) sensor and an Extended Kalman Filter (EKF). Several drones used for remote sensing studies are fitted with a LiDAR sensor [13, 16] avoiding the need for installing additional sensors to track the position of the suspended payload.

Furthermore, a robust drone controller that reduces the motion of the suspended payload and results in a stable flight is needed. Fractional calculus can be applied to control theory to provide superior performance over integer-order controllers [17]. Conventional drone controllers utilize integer-order derivatives and integrals to control the system's response to user input or external perturbations. For a suspended payload experiencing non-linear swing, a quick response is required. Fractional calculus may be preferred as it applies non-integer order derivatives and integers in both real and complex number powers providing more flexibility when tuning the performance of the controller. This could result in better controller response properties such as reduced time delay, overshoot, steady-state error and transient response which can be key to stabilizing non-linear dynamics of drone-payload system [18–20]. For this thesis, only feedback control was considered and model-based approaches were not investigated since model-based approaches are arduous and error-prone, especially in cases where external disturbances are not completely known.

Beyond my research focus on Arctic remote sensing, the application of such a system can be further extended to other drone applications including firefighting, delivery services, search and rescue missions, and geographic mapping. Moreover, the fractional control aspect from this research could be applied to space applications including fine pointing of spacecraft, especially for those in low altitude orbits (approximately < 400 km) where they experience high drag scenarios or to control spacecraft with fuel tanks that may experience slosh [20–22].

1.3 Hypotheses

My proposed hypotheses are as follows:

H1. A LiDAR sensor, in combination with a Kalman filter, can be used as a feedback sensor to track and estimate the position of a drone payload to an accuracy of 3 cm relative to the drone position.

H2. A fractional-order PID (FOPID) controller, with both fractional differential and integral terms, provides better response to stabilizing the non-linear dynamics of a drone payload swinging in 3D when compared to the traditional integer-order PID (IOPID) controller through reducing the percent overshoot by at least 80%, settling time by 75% and steady-state error by 10%.

1.4 Methodology Overview

To test my hypothesis H1, I developed a mathematical model of a drone carrying a payload via a cable. Then, I developed an algorithm that uses LiDAR point cloud data to locate the position of the payload. Finally, I used an Extended Kalman Filter

to combine my mathematical model and LiDAR measurements to obtain an accurate estimate of the position of the suspended payload. I verified my results using Vicon motion capture system.

For hypothesis H2, I developed a drone flight control software with both IOPID and FOPID controllers. The flight software integrated with the indoor Vicon motion capture system and the Quanser QDrone. Next, I added drone attitude dynamics to the mathematical model from H1 and used Particle Swarm Optimization (PSO) to automatically tune the controller parameters for both IOPID and FOPID controllers. Finally, I simulated and performed hardware tests to compare the results of each controller.

1.5 Research Contribution

This thesis evaluates the feasibility and value of a fractional feedback controller for handling non-linear plants when compared to linear feedback controllers by developing, simulating, testing, and comparing a fractional-order PID controller to an integer-order PID controller on a drone carrying a swaying payload (referred to as the plant). To aid the controller evaluation, I also developed a position tracking algorithm that estimates the position of the suspended payload.

Major contributions resulting from this thesis include:

1. A payload position tracking algorithm using a Velodyne VLP-16$^{\text{TM}}$ LiDAR sensor's point cloud data and an Extended Kalman Filter.

2. A non-linear mathematical model representing the dynamics of a drone-based cable-suspended payload system used for simulation and controller parameter tuning.

3. A closed-loop flight control law for a drone using a fractional-order PID controller with real time Vicon position feedback (used to emulate LiDAR sensor data).

4. An evaluation of the performance of an IOPID versus a FOPID using a non-linear plant (Quanser QDrone carrying a cable-suspended payload).

1.6 Thesis Summary

The remainder of this thesis is outlined as follows. Chapter 2 provides the literature review on previously implemented payload carrying strategies. Moreover, various mechanics for tracking cable-suspended payloads and non-linear control strategies, used for similar systems, are compared.

Chapter 3 derives the mathematical models that relate the drone and payload dynamics. It introduces a simple 3D pendulum model to replicate a payload suspended from a stationary drone. Then, it extends the 3D model to a non-stationary drone and studies the effects of the sway of the payload on the drone and the motion of the drone on the sway of the payload.

Chapter 4 introduces the LiDAR hardware, tracking algorithm and the Extended Kalman Filter used to estimate the location of the suspended payload. Further, experimental results testing the accuracy of the position estimate are presented and discussed.

Chapter 5 provides background on the two controllers studied in this thesis: IOPID and FOPID. This chapter outlines the working principle of each controller type and evaluates the various approximations of the fractional order integrators and derivatives for application on a drone.

Chapter 6 describes a closed-loop drone control law that I developed for a drone carrying a payload. It describes the controller parameter tuning using Particle Swarm Optimization and discusses the simulation results of the system's performance. Then, the test setup and experimental results of both controller types are compared and discussed.

Chapter 7 summarizes the main research objectives, contributions, and areas for consideration in future studies.

Chapter 2

Literature Review[1]

This chapter provides background information on drone-based payload transportation. A literature review on methods for carrying payloads on a drone, payload tracking, and non-linear controller strategies for drone-based plants is provided.

2.1 Drone-based Payload Carrying Strategies

Payload transportation using aerial vehicles has been used on helicopters for a long time [23–27]. While the swaying of the payload can affect the control and obstacle avoidance of a helicopter, the larger mass and size of the helicopter minimize the effect of the sway on the helicopter's flight stability.

[1]Portions of this section have been reproduced with permission from:

[1] Patel, M., and Ferguson, P., "Tracking and Estimation of a Swaying Payload Using a LiDAR and an Extended Kalman Filter," 2021 IEEE ROSE. ©2021 Mitesh Patel and Philip Ferguson.

[2] Mitesh S. Patel and Philip A. Ferguson. "Drone-Based Cable-Suspended Payload Tracking and Estimation Using Simulated LiDAR Measurements and an Extended Kalman Filter," AIAA 2022-2290. AIAA SCITECH 2022 Forum. ©2022 Mitesh Patel and Philip Ferguson.

On the contrary, the smaller size and mass of modern load-carrying drones are affected more by the payload's mass, which can lead to flight instability. With drone-based transportation increasingly gaining attention in civilian and industrial applications [3, 28, 29], minimizing flight instabilities is becoming a major focus. If the drone is not properly designed for payload carrying, or if the payload is not attached to the drone as per the manufacturer's requirements, it could lead to a dangerous crash.

The most common way of attaching a payload to a drone is by mounting the payload to the drone via a rigid connection. For smaller payloads, especially vision-based sensors, a gimbal is used while larger payloads are mounted directly to the drone's bottom chassis [30, 31].

Extending the rigid connection idea, one way to prevent the suspended payload's motion is to use a rigid connection between the payload and the drone, as demonstrated on micro aerial vehicles by Loianno et al. [32]. Also, Lee. H et al. [33] used rigid connections to demonstrate payload handling and transportation using multiple aerial vehicles. They conducted a successful flight with two hexacopters mounted with end effectors to transport a common load along a desired trajectory. However, using a rigid connection to the payload will result in a less agile drone affecting the drone's maneuverability and not all applications can accommodate a rigid connection. Simpler solutions exist to counteract the swaying motion and maintain the payload stable relative to the drone [34], such as using a robust closed-loop controller alongside a single cable that is used to suspend the payload.

Previous studies considered using a single vehicle [35–38], as well as a swarm of drones to carry a payload, [39–41] to transport a cable suspended payload. For single drone applications with single cable-suspended payloads, a closed-loop feedback system that measures the position of the payload has been investigated in some studies. An indoor motion capture system was used to provide the attitude and

position of the aerial vehicle and payload in order to study a lift maneuver of a cable-suspended payload in [42]. Using an indoor motion capture system to provide the swing angle of the slung payload, Liang X. et al. [43] developed a trajectory planning algorithm for a cable-suspended payload. Similarly, other studies have used indoor motion capture systems [44] or an onboard camera [45] to provide position or attitude feedback of the aerial vehicle and the suspended payload. In [45], the authors performed flight trials using only a visual camera with a marker detection algorithm to detect the payload's position, however, the visual object detection algorithm was limited to a frequency of 25 Hz update rate.

Other solutions to reduce the sway of the suspended payload exist. These solutions do not require sensors to track the position of a suspended payload, but instead, require the introduction of dynamic coupling between the payload and the drone or data fusion techniques. Angelis [46] utilized signal processing techniques to estimate the cable's swing angle. The swing angle of the payload is autonomously measured using data fusion of the Inertial Measurement Unit (IMU) on the drone, control inputs to the drone, and the dynamics model of a slung load system, but the system performance was not tested when subjected to unknown disturbances. Chen [34] developed a cable-suspended payload with four cables by studying the forces and torques on all four actuators of a quadcopter. Chen showed that the use of four cables to suspend the payload from a quadrotor increases the coupling between the quadrotor and the suspended payload, thereby avoiding the use of additional sensors. However, a four cable system is not always an option and many applications may require the payload to be suspended via a single cable. Further, a multi-cable system may perform as a single-cable system when the cable length is significantly larger than the distance between the cable attachment points on the drone (*i.e.,* the drone's span).

Therefore, my research considered a simple method of suspending the GPR using a single cable attached to the drone which will result in a swaying motion of the payload. Furthermore, this research was aimed at aiding Arctic researchers who already utilize a LiDAR sensor for remote sensing. This provides an opportunity for me to utilize a LiDAR sensor to track the suspended payload and implement a non-linear feedback controller on a drone with a single cable-suspended payload.

2.2 Non-Linear Control Strategies

A quadrotor is an underactuated non-linear system. A quadrotor is designed with four actuators, all of which are located in a single plane, thereby preventing the quadrotor from changing its position without changing its attitude. Adding a cable-suspended payload to the quadrotor affects the stability and controllability of the underactuated system. Numerous studies have implemented either path planning techniques or control algorithms to minimize the sway of the payload. Some of the controllers were linear like the integer-order PID controller [47–49], while others implemented non-linear controllers like sliding mode controller [50], fuzzy control [51], and backstepping controllers [52] for better performance over linear controllers [53].

Researchers have designed algorithms that utilized trajectory planning to generate a desired path for a swing-free motion of the payload [40, 54–56]. In [55] and [56], the authors tested path planning and control of a quadrotor with a cable-suspended payload for both planar and three-dimensional trajectories. Moreover, Sreenath and Kumar [57] tested feasible trajectory generation for both the quadrotor and slung payload using multiple quadrotors. However, the overall complexity of the controller increased since model dynamics needed to be added to the system controller. Furthermore, model-driven approaches can be tedious to derive and can be sensitive to unmodeled dynamics.

Some researchers have opted to implement non-linear feedback controllers over path-planning algorithms. Different non-linear feedback controllers have been tested in studies related to a suspended payload. Angelis et al. [46] utilized an artificial two-time-scale separation to develop a non-linear controller that could simultaneously perform payload swing damping and trajectory control. Liang et al. [58] developed a non-linear hierarchical energy-based controller for an underactuated quadrotor with a suspended payload while Yang and Xian [59] designed a non-linear adaptive energy-based controller to control the payload's swing angle. Nicotra et al. [60] proposed a nested saturation controller with an inner loop to control the roll and pitch of the vehicle and an outer loop to control the position of the vehicle and the swing angle of the payload. Multiple researchers have also looked at using reinforcement learning on aerial vehicles to learn the dynamics of the slung payload to perform trajectory tracking for the suspended payload [35, 61–63]. However, all these controllers require more complicated algorithms and have been tested with small payload swing angles with no external disturbances.

Sliding Mode Control (SMC) is another extensively researched non-linear feedback controller for slung payload applications [64–69]. SMC leverages the Lyapunov stability method to keep the non-linear system under control. The controller applies discontinuous control signals that force the dynamic system to follow a particular surface (sliding function) in state space while remaining bounded within the control structures. With SMC, the system behaviour can be adjusted by the choice of the sliding function and the closed-loop system becomes insensitive to particular model uncertainties [70]. However, practical implementations of SMC can result in undesirable high frequency oscillations of the system (chattering) that could damage the hardware [71]. Numerous solutions to SMC chattering have been proposed, including the use of Higher-Order SMC (HOSMC) that considers the integral of the discontinuous term, thereby suppressing the chatter [72–75]. Generally, control laws for an

SMC controller for a drone-based slung payload are complex and tedious to derive. The inclusion of chatter-suppressing algorithms to SMC makes the controller more complex, and simpler controllers are preferred.

Overall, most of these studies tend to address either position tracking, altitude control, or attitude control but not all three together. My research looks at a different approach to designing a non-linear feedback controller by implementing fractional calculus to maintain position, altitude, and attitude control for a drone-based cable-suspended GPR.

Fractional calculus is a generalization of integer-order calculus that has been studied for many years but has gained the attention of researchers in recent times [76]. It has been applied to several applications including thermoelastic systems, viscoelastic systems, diffusion systems, biological systems, and spacecraft control systems [20, 77–80]. Moreover, while fractional calculus has been implemented in drone control systems [81–86], the focus area of these studies has not been on drone-based slung payload systems.

In general, previous studies have shown that fractional control performs better than traditional integer control [20,76,87], but the improvement varies with the target system and the definition of the fractional integral and derivative used. Regardless, previous studies on FOPID controllers for attitude control of small satellites have shown improvements of approximately 88% reduction in percent overshoot and a 74% reduction in settling time with a trade-off in the steady-state error increasing from 2% to 7% [20]. Other studies have shown similar results with an 84% - 88% reduction in percent overshoot and an almost negligible reduction in the system response rise time [76, 87]. Therefore, previous literature seems to suggest that implementing a fractional-order controller could provide a good opportunity to control the sway of a single cable-suspended payload and stabilize the drone's flight.

This thesis considers implementing an effective controller for a drone carrying a slung payload using a FOPID controller. A fractional controller could provide a more responsive and accurate control solution to counteract the sway of the payload. Model dynamics and path planning algorithms are not included in the controller, thereby making the controller a simple feedback controller that can be used to compare the performance of a FOPID controller to an IOPID controller for drone-based slung payload systems.

2.3 Chapter Summary

This chapter provided a review of studies conducted for a drone-payload system, including payload attachment strategies, payload tracking, and control algorithms. A rigid connection or a multi-cable system have been used to attach the payload from the drone, but these methods reduce the agility of the drone and a simpler and widely used alternative is to suspend the payload using a single cable. However, a single cable has been shown to induce payload sway and sensors such as an optical camera have been used to track the sway of the payload. To control the sway of the payload, several model-based control strategies, such as command shaping or sliding mode control, have been used but these methods can be tedious to derive and are error-prone. Hence, to bridge the research gap between drone technology and Arctic remote sensing, in this thesis, I proposed using a single cable to suspend the payload and a LiDAR sensor to track the payload's position. Further, I considered a feedback fractional control (with no model knowledge) to evaluate its performance for a drone-payload system.

Chapter 3

Drone-Payload Dynamics Modeling[1]

In this chapter, the two dynamics model are derived: a mathematical model of a three-dimensional pendulum is derived in Section 3.1, and a dynamics model of a slung payload attached to a non-stationary drone is derived in Section 3.2. Then, each model is verified by applying a simple force function and simulating it in MATLAB. The resulting output from each model is graphed and presented.

[1]Portions of this section have been reproduced with permission from:

[1] Patel, M., and Ferguson, P., "Tracking and Estimation of a Swaying Payload Using a LiDAR and an Extended Kalman Filter," 2021 IEEE ROSE. ©2021 Mitesh Patel and Philip Ferguson.

[2] Mitesh S. Patel and Philip A. Ferguson. "Drone-Based Cable-Suspended Payload Tracking and Estimation Using Simulated LiDAR Measurements and an Extended Kalman Filter," AIAA 2022-2290. AIAA SCITECH 2022 Forum. ©2022 Mitesh Patel and Philip Ferguson.

3.1 3D Pendulum Dynamics

In this section, a mathematical model of a 3D pendulum is derived in Cartesian coordinates. Cartesian coordinates are preferred over spherical coordinates since spherical coordinates suffer from angle wrapping as shown in [1]. A schematic showing the coordinate system and key model parameters is shown in Figure 3.1.

I modeled the payload as a point-mass pendulum, with mass m. A cable of length L is attached to the origin of an Earth-fixed inertial reference frame ($\vec{\boldsymbol{\mathcal{F}}}_I$) and goes to the center of mass of the payload. It is assumed that the cable is massless, rigid, and inelastic. The rigid cable assumption is only valid when the cable is in tension and not when the cable is in compression or slack.

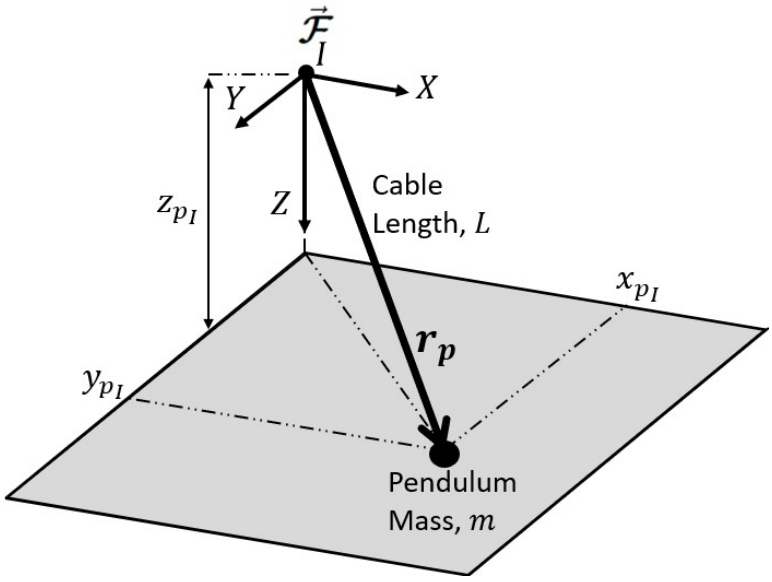


Figure 3.1: Single cable-suspended payload schematic.

Then, in the $\vec{\boldsymbol{\mathcal{F}}}_I$ frame, $\mathbf{r_p} = [x_{p_I}, y_{p_I}, z_{p_I}]^T$ is the position of the payload and $\mathbf{v_p}$ $= [\dot{x}_{p_I}, \dot{y}_{p_I}, \dot{z}_{p_I}]^T$ is the velocity of the payload. The pendulum has only 2 Degrees of Freedom (DOF) since the cable is assumed to be rigid. This can also be demonstrated if the pendulum was modeled in spherical coordinates. In spherical coordinates,

the pendulum can only swing in two directions which can be represented by the angle between the cable and the Z axis and the angle between the X axis and the pendulum's projection in the $X-Y$ plane. Therefore, to constrain the dynamics in Cartesian coordinates, the pendulum can only be modeled using two of the three position variables $[x_{p_I}, y_{p_I}, z_{p_I}]$, with the third being dictated by the constraint that the cable remains taught and rigid.

For this model, I modeled the pendulum dynamics using the x and y position of the pendulum where the z position was derived using the Euclidean distance in Equation 3.1.

$$z_{p_I} = \sqrt{L^2 - x_{p_I}^2 - y_{p_I}^2} \tag{3.1}$$

where L is the total pendulum length. Then, the velocity of the pendulum in the z direction can be found by differentiating Equation 3.1 as:

$$\dot{z}_{p_I} = -\frac{x_{p_I}\dot{x}_{p_I} + y_{p_I}\dot{y}_{p_I}}{\sqrt{L^2 - x_{p_I}^2 - y_{p_I}^2}} \tag{3.2}$$

To derive the full non-linear 3D pendulum model, large swing angles, up to a maximum angular deflection of 90 degrees, are considered. Lagrange's Equation [88], shown in Equation 3.3, is used to derive the equations of motion of the 3D pendulum without any linearization.

$$\frac{d}{dt}\left(\frac{d\Lambda}{d\dot{x}_i}\right) - \frac{d\Lambda}{dx_i} + \frac{dR}{d\dot{x}_i} = \mathbf{F} \tag{3.3}$$

where Λ is the Lagrangian and is defined as the difference between kinetic energy (T) and potential energy (U), as in Equation 3.4. R is a function representing the energy dissipation of the system, and $\mathbf{F}$ consists of the force vector in the reference frame of the generalized variables x_i.

$$\Lambda = T - U \tag{3.4}$$

For this model, it was assumed that no external forces other than gravity were applied to the pendulum and that energy dissipation through viscous damping from aerodynamic drag was negligible when compared to the magnitude of both kinetic and potential energy of the 3D pendulum. Therefore, Equation 3.3 reduces to:

$$\frac{d}{dt}\left(\frac{d\Lambda}{d\dot{x}_i}\right) - \frac{d\Lambda}{dx_i} = 0 \tag{3.5}$$

Then, the kinetic energy and potential energy of the pendulum can be calculated using Equation 3.6 and Equation 3.7, respectively.

$$T = \frac{1}{2}m(\dot{x}_{p_I}^2 + \dot{y}_{p_I}^2 + \dot{z}_{p_I}^2) \tag{3.6}$$

$$U = -mgz_{p_I} \tag{3.7}$$

where $g = 9.81\ m/s^2$ represents the acceleration due to gravity. Substituting Equation 3.6 and Equation 3.7 into Equation 3.4, and replacing z_{p_I} and $\dot{z}_{p_I}$ with Equation 3.1 and Equation 3.2 results in:

$$\Lambda = \frac{1}{2}m\left(\dot{x}_{p_I}^2 + \dot{y}_{p_I}^2 + \left[-\frac{x_{p_I}\dot{x}_{p_I} + y_{p_I}\dot{y}_{p_I}}{\sqrt{L^2 - x_{p_I}^2 - y_{p_I}^2}}\right]^2\right) + mg\sqrt{L^2 - x_{p_I}^2 - y_{p_I}^2} \tag{3.8}$$

Using Equation 3.5 with the generalized variables as the pendulum's x_{p_I} and y_{p_I}, two equations of motions are derived. The Lagrange's equation that was solved for each generalized variable is shown in Equation 3.9 and Equation 3.10.

$$\frac{d}{dt}\left(\frac{d\Lambda}{d\dot{x}_{p_I}}\right) - \frac{d\Lambda}{dx_{p_I}} = 0 \tag{3.9}$$

$$\frac{d}{dt}\left(\frac{d\Lambda}{d\dot{y}_{p_I}}\right) - \frac{d\Lambda}{dy_{p_I}} = 0 \tag{3.10}$$

I used MATLAB's Symbolic Toolbox to solve each Lagrange's equation for the equations of motion of the 3D pendulum. See Appendix B for the MATLAB code that was used to solve for the equations of motion for this 3D pendulum.

As an example for expanding the terms in Lagrange's Equation, consider the generalized variable x_{p_I} (Equation 3.9). The derivative of the Lagrangian with respect to the $\dot{x}_{p_I}$ and the derivative of the Lagrangian with respect to the x_{p_I} are shown in Equation 3.11 and Equation 3.12, respectively.

$$\frac{d\Lambda}{d\dot{x}_{p_I}} = m\left(\dot{x}_{p_I} + \left[\frac{x_{p_I}\dot{x}_{p_I} + y_{p_I}\dot{y}_{p_I}}{\sqrt{L^2 - x_{p_I}^2 - y_{p_I}^2}}\right]\frac{x_{p_I}}{\sqrt{L^2 - x_{p_I}^2 - y_{p_I}^2}}\right) \tag{3.11}$$

$$\begin{aligned}\frac{d\Lambda}{dx_{p_I}} =& \left\{ m\left[\frac{x_{p_I}\dot{x}_{p_I} + y_{p_I}\dot{y}_{p_I}}{\sqrt{L^2 - x_{p_I}^2 - y_{p_I}^2}}\right]\left[\frac{\dot{x}_{p_I}(L^2 - x_{p_I}^2 - y_{p_I}^2) + (x_{p_I}\dot{x}_{p_I} + y_{p_I}\dot{y}_{p_I})x_{p_I}}{(L^2 - x_{p_I}^2 - y_{p_I})^{3/2}}\right]\right. \\ & \left. - g\frac{x_{p_I}}{\sqrt{L^2 - x_{p_I}^2 - y_{p_I}^2}}\right\}\end{aligned} \tag{3.12}$$

Given the complexity of the calculations, the derivative $\frac{d}{dt}\left(\frac{d\Lambda}{d\dot{x}_{p_I}}\right)$ term has not been shown here. However, once this term has been calculated, the result can be substituted into Equation 3.5. The resulting equation can be solved for $\ddot{x}_{p_I}$ to obtain the equation of motion of the pendulum's position in the x-direction.

With variable σ defined by Equation 3.13, the equations of motion representing the 3D pendulum dynamics are shown in Equation 3.14 and Equation 3.15. The z-position of the pendulum can be derived using Equation 3.1.

$$\sigma_1 = \sqrt{L^2 - x_{p_I}^2 - y_{p_I}^2} \tag{3.13}$$

$$\begin{aligned}\ddot{x}_{p_I} = {} & g x_{p_I}^5 + 2g x_{p_I}^3 y_{p_I}^2 - \left(\sigma \dot{y}_{p_I}^2 - 2gL^2\right) x_{p_I}^3 + g x_{p_I} y_{p_I}^4 - \sigma \dot{x}_{p_I}^2 x_{p_I} y_{p_I}^2 \\ & - 2gL^2 x_{p_I} y_{p_I}^2 + \left(\sigma \dot{x}_{p_I}^2 L^2 + \sigma \dot{y}_{p_I}^2 L^2 + gL^4\right) x_{p_I} + 2\sigma x_{p_I}^2 y_{p_I} \dot{x}_{p_I} \dot{y}_{p_I}\end{aligned} \tag{3.14}$$

$$\begin{aligned}\ddot{y}_{p_I} = {} & g y_{p_I}^5 + 2g x_{p_I}^2 y_{p_I}^3 - \left(\sigma \dot{x}_{p_I}^2 - 2gL^2\right) y_{p_I}^3 + g x_{p_I}^4 y_{p_I} - \sigma \dot{y}_{p_I}^2 x_{p_I}^2 y_{p_I} \\ & - 2gL^2 x_{p_I}^2 y_{p_I} + \left(\sigma \dot{x}_{p_I}^2 L^2 + \sigma \dot{y}_{p_I}^2 L^2 + gL^4\right) y_{p_I} + 2\sigma x_{p_I} y_{p_I}^2 \dot{x}_{p_I} \dot{y}_{p_I}\end{aligned} \tag{3.15}$$

3.2 Drone-based Cable-Suspended Payload Dynamics

In this section, I develop a mathematical model of the dynamics of a payload attached to a non-stationary drone. Expanding the 3D pendulum in Section 3.1, I used Lagrange dynamics to develop the full non-linear model to study the effects of the sway of the payload on the drone and the motion of the drone on the sway of the payload. Energy dissipation, control forces and external forces on the payload are also included in the model. Figure 3.1 shows a schematic of a payload suspended from a drone using an inertial Earth-fixed reference frame ($\vec{\boldsymbol{\mathcal{F}}}_I$) and a drone-fixed body frame ($\vec{\boldsymbol{\mathcal{F}}}_d$).

The payload is modeled as a point-mass pendulum attached to the drone via a cable of length L. It is assumed that the cable is attached to the center of mass of the drone and the center of mass of the payload. Additionally, the cable is assumed to be mass-less, rigid, and inelastic. Therefore, a drawback of the rigid cable assumption is that the drone will only accelerate the payload when the cable is under tension. Compression of the cable can be ignored if any downward acceleration from the drone is maintained to be slower than the acceleration due to gravity, a reasonable assumption for stable flight.

I modeled the drone as a point mass of mass M attached to the payload of mass

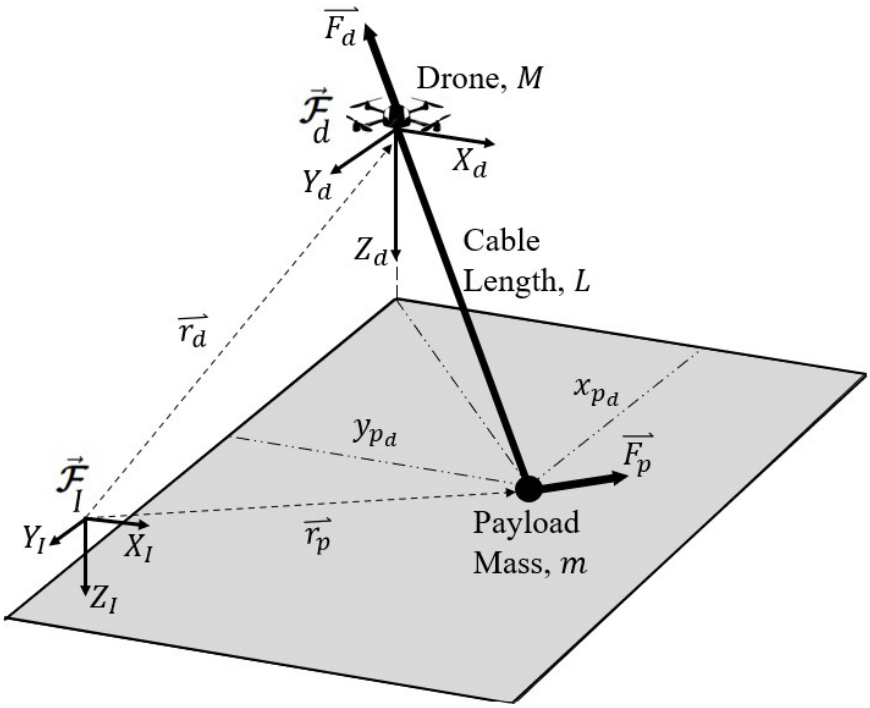


Figure 3.2: Drone with a cable-suspended payload schematic.

m via a rigid cable. Then, using the vectrix notation, the position of the drone ($\mathbf{r_d}$) and the velocity of the drone ($\mathbf{v_d}$) can be expressed using Equation 3.16 and Equation 3.17.

$$\vec{r}_d = \vec{\boldsymbol{\mathcal{F}}}_I^T \vec{r}_{d_I} = \vec{\boldsymbol{\mathcal{F}}}_d^T \vec{r}_{d_d} \tag{3.16}$$

$$\vec{v}_d = \vec{\boldsymbol{\mathcal{F}}}_I^T \vec{v}_{d_I} = \vec{\boldsymbol{\mathcal{F}}}_d^T \vec{v}_{d_d} \tag{3.17}$$

where $\vec{\boldsymbol{\mathcal{F}}}_I$ and $\vec{\boldsymbol{\mathcal{F}}}_d$ represents the unit vector of the inertial frame and the drone's frame, respectively. The position of the payload ($\mathbf{r_p}$) and the velocity of the payload ($\mathbf{v_p}$) can be expressed using Equation 3.18 and Equation 3.19.

$$\vec{r}_p = \vec{\boldsymbol{\mathcal{F}}}_I^T \vec{r}_{p_I} = \vec{\boldsymbol{\mathcal{F}}}_d^T \vec{r}_{p_d} \tag{3.18}$$

$$\vec{v}_p = \vec{\boldsymbol{\mathcal{F}}}_I^T \vec{v}_{p_I} = \vec{\boldsymbol{\mathcal{F}}}_d^T \vec{v}_{p_d} \tag{3.19}$$

Then, $\mathbf{r_{d_I}} = [x_{d_I}, y_{d_I}, z_{d_I}]^T$ is the position of the drone and $\mathbf{v_{d_I}} = [\dot{x}_{d_I}, \dot{y}_{d_I}, \dot{z}_{d_I}]^T$ is the velocity of the drone in $\vec{\boldsymbol{\mathcal{F}}}_I$. Similarly, $\mathbf{r_{p_I}} = [x_{p_I}, y_{p_I}, z_{p_I}]^T$ is the position of the payload and $\mathbf{v_{p_I}} = [\dot{x}_{p_I}, \dot{y}_{p_I}, \dot{z}_{p_I}]^T$ is the velocity of the payload, in $\vec{\boldsymbol{\mathcal{F}}}_I$.

The control forces acting on the drone are modeled as $\mathbf{F_{d_I}} = [F_{d_x}, F_{d_y}, F_{d_z}]^T$, where $\mathbf{F_{d_I}}$ are the forces generated by the drone's propellers, expressed in the $\vec{\boldsymbol{\mathcal{F}}}_I$ frame. Similar to the 3D pendulum in Section 3.1, a payload attached to the drone using a single cable has two DOF and can only be modeled using any two of the three coordinates. I used the x and y position of the payload as the generalized coordinates in Lagrange dynamics. Hence, the direct forces applied to the payload (forces applied to the payload that do not result from the cable tension) are modeled as $\mathbf{F_{p_I}} = [F_{p_x}, F_{p_y}]^T$ in $\vec{\boldsymbol{\mathcal{F}}}_I$. Note that the combination of $\mathbf{F_d}$ and $\mathbf{F_p}$ must not violate the rigid cable assumption for the model to be valid.

The dynamics of the payload is modeled using only the x and y position of the payload. Applying the constraint that the cable is rigid, the z position of the payload in the Earth-fixed reference frame is derived using Equation 3.20.

$$z_{p_I} = z_{d_I} + \sqrt{L^2 - (x_{d_I} - x_{p_I})^2 - (y_{d_I} - y_{p_I})^2} \tag{3.20}$$

Then, the velocity of the payload in the z-direction can be calculated as:

$$\dot{z}_{p_I} = \dot{z}_{d_I} - \frac{(\dot{x}_{d_I} - \dot{x}_{p_I})(x_{d_I} - x_{p_I}) + (\dot{y}_{d_I} - \dot{y}_{p_I})(y_{d_I} - y_{p_I})}{\sqrt{L^2 - (x_{d_I} - x_{p_I})^2 - (y_{d_I} - y_{p_I})^2}} \tag{3.21}$$

Using Langrange dynamics with dissipation and external forces, Equation 3.3, I derived the equations of motion for this system. I modeled energy dissipation using Rayleigh's dissipation function that assumes frictional dissipation is proportional to the velocity of an object, as shown in Equation 3.22.

$$R = \frac{1}{2}b\left[(\dot{x}_{d_I} - \dot{x}_{p_I})^2 + (\dot{y}_{d_I} - \dot{y}_{p_I})^2 + (\dot{z}_{d_I} - \dot{z}_{p_I})^2\right] \tag{3.22}$$

where b is the dissipation constant and is negative due to energy loss. Then, the kinetic energy and potential energy of the system can be derived as the sums of the kinetic energies and potential energies of the swinging payload and the drone respectively, using Equation 3.23 and Equation 3.24.

$$T = \frac{1}{2}M\left(\dot{x}_{d_I}^2 + \dot{y}_{d_I}^2 + \dot{z}_{d_I}^2\right) + \frac{1}{2}m\left(\dot{x}_{p_I}^2 + \dot{y}_{p_I}^2 + \dot{z}_{p_I}^2\right) \tag{3.23}$$

$$U = -Mgz_{d_I} - mgz_{p_I} \tag{3.24}$$

Substituting Equation 3.23 and Equation 3.24 into Equation 3.3, and replacing z_{p_I} and $\dot{z}_{p_I}$ with Equation 3.20 and Equation 3.21 results in:

$$\begin{aligned}\Lambda = {} & \frac{1}{2}M\left(\dot{x}_{d_I}^2 + \dot{y}_{d_I}^2 + \dot{z}_{d_I}^2\right) \\ & + \frac{1}{2}m\left(\dot{x}_{p_I}^2 + \dot{y}_{p_I}^2 + \left[\dot{z}_{d_I} - \frac{(\dot{x}_{d_I} - \dot{x}_{p_I})(x_{d_I} - x_{p_I}) + (\dot{y}_{d_I} - \dot{y}_{p_I})(y_{d_I} - y_{p_I})}{\sqrt{L^2 - (x_{d_I} - x_{p_I})^2 - (y_{d_I} - y_{p_I})^2}}\right]^2\right) \\ & + Mgz_{d_I} + mg\left[z_{d_I} + \sqrt{L^2 - (x_{d_I} - x_{p_I})^2 - (y_{d_I} - y_{p_I})^2}\right]\end{aligned} \tag{3.25}$$

The equations of motion can then be derived using Equation 3.3) with the generalized variable (x_i) as x_{d_I} for the x-position of the drone, y_{d_I} for the y-position of the drone, z_{d_I} for the z-position of the drone, and x_{p_I} for the x-position of the payload and y_{p_I} for the y-position of the payload in $\vec{\boldsymbol{\mathcal{F}}}_I$. The Lagrangian that was solved for each generalized variable is shown in Equation 3.26 to Equation 3.30.

$$\frac{d}{dt}\left(\frac{d\Lambda}{d\dot{x}_{d_I}}\right) - \frac{d\Lambda}{dx_{d_I}} + \frac{dR}{d\dot{x}_{d_I}} = F_{d_x} \tag{3.26}$$

$$\frac{d}{dt}\left(\frac{d\Lambda}{d\dot{y}_{d_I}}\right) - \frac{d\Lambda}{dy_{d_I}} + \frac{dR}{d\dot{y}_{d_I}} = F_{d_y} \tag{3.27}$$

$$\frac{d}{dt}\left(\frac{d\Lambda}{d\dot{z}_{d_I}}\right) - \frac{d\Lambda}{dz_{d_I}} + \frac{dR}{d\dot{z}_{d_I}} = F_{d_z} \tag{3.28}$$

$$\frac{d}{dt}\left(\frac{d\Lambda}{d\dot{x}_{p_I}}\right) - \frac{d\Lambda}{dx_{p_I}} + \frac{dR}{d\dot{x}_{p_I}} = F_{p_x} \tag{3.29}$$

$$\frac{d}{dt}\left(\frac{d\Lambda}{d\dot{y}_{p_I}}\right) - \frac{d\Lambda}{dy_{p_I}} + \frac{dR}{d\dot{y}_{p_I}} = F_{p_y} \tag{3.30}$$

I used MATLAB's Symbolic Toolbox to solve for the equations of motion. Since the equations involve much more terms than the 3D pendulum model, the equations are much longer and are therefore not presented here in text. See Appendix B for the MATLAB code that was used to solve for the equations of motion for this drone-based cable-suspended payload.

The position of the payload relative to the drone (*i.e.*, position of the payload in $\vec{\boldsymbol{\mathcal{F}}}_d$) is important since it provides the relative displacement between the drone and the suspended payload. This can be calculated using Equation 3.31 and Equation 3.32 for the x and y position of the payload, respectively.

$$x_{p_d} = x_{p_I} - x_{d_I} \tag{3.31}$$

$$y_{p_d} = y_{p_I} - y_{d_I} \tag{3.32}$$

3.3 Dynamics Simulation

I tested each model in simulation to verify they were derived correctly. For the 3D pendulum model, the pendulum mass was given an initial displacement with zero initial velocity. For the drone-payload model, a forcing function in the form of a square wave was applied to the drone.

3.3.1 3D Pendulum Simulation

The 3D pendulum dynamics are defined by Equation 3.14 and Equation 3.15. Integrating these equations twice results in the position of the pendulum on the x-axis and y-axis. Then, Equation 3.1 can be used to determine the z-position of the pendulum.

To initiate the motion of the pendulum, I used the following initial conditions and model parameters.

$$\begin{aligned} L = 1\ m \qquad & x_{p_I} = 0.2\ m \qquad & \dot{x}_{p_I} = 0\ m/s \\ m = 0.5\ kg \qquad & y_{p_I} = 0.3\ m \qquad & \dot{y}_{p_I} = 0\ m/s \end{aligned} \tag{3.33}$$

I simulated the model for 10 s using the Runge-Kutta integrator with a time step of 0.001 s. The resulting position and velocity of the pendulum in 3D space are shown in Figure 3.3.

The pendulum swings between $\pm 20\ cm$ in the x-axis and $\pm 30\ cm$ in the y-axis. Since damping was not included in the model, the pendulum motion is expected to oscillate between the maximum displacements (defined by the initial conditions). The z-position of the pendulum is maximum when the x and y positions are zero as expected. Further, the velocity of the pendulum in the x-axis and y-axis is maximum when the cable is vertical and where the z-velocity of the pendulum is zero. This verifies that the 3D pendulum model is derived correctly and performs as expected.

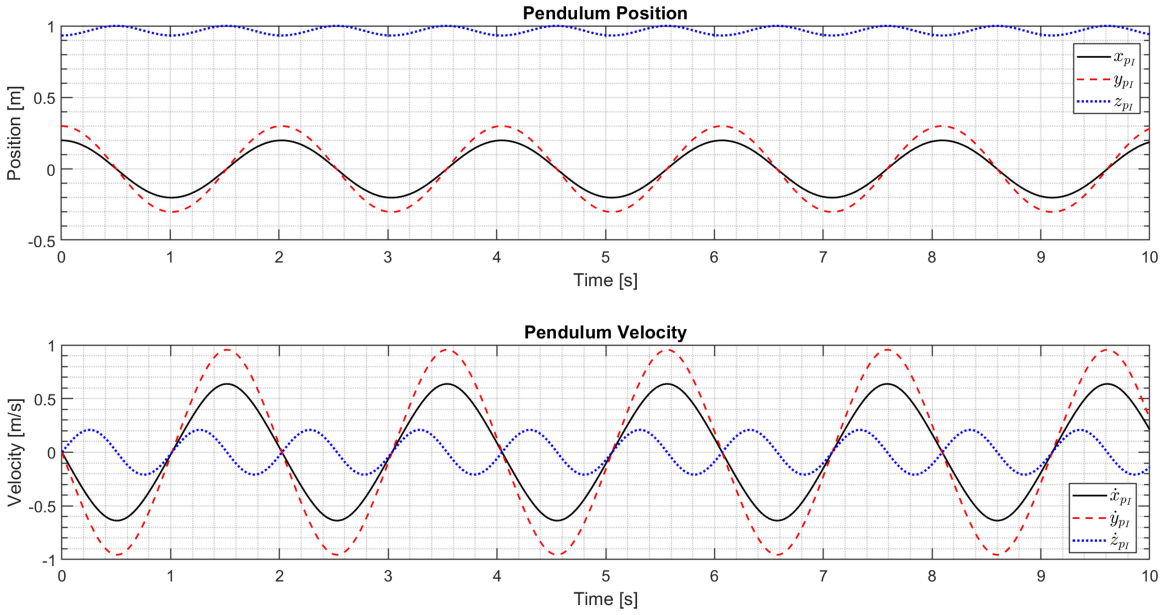


Figure 3.3: Simulated position and velocity output of the 3D pendulum model. (Note: + z-axis is down)

3.3.2 Drone-Payload Simulation

I simulated the drone-based cable-suspended model using five equations of motion that result from solving Equation 3.26 to Equation 3.30. I used the following initial conditions and model parameters.

$$\begin{aligned} M &= 1\ kg & m &= 0.1\ kg & & & L &= 1\ m \\ x_{d_I} &= 0\ m & \dot{x}_{d_I} &= 0\ m/s & x_{p_I} &= 0\ m, & \dot{x}_{p_I} &= 0\ m/s \\ y_{d_I} &= 0\ m & \dot{y}_{d_I} &= 0\ m/s & y_{p_I} &= 0\ m, & \dot{y}_{p_I} &= 0\ m/s \\ z_{d_I} &= 0\ m & \dot{z}_{d_I} &= 0\ m/s & & & & \end{aligned} \tag{3.34}$$

The drone was assumed to be hovering at its initial state and a force was applied to the drone along the x-axis and y-axis. The forcing function is shown in Figure 3.4. The direct forces applied to the payload (F_{p_x}, F_{p_y}) were zero.

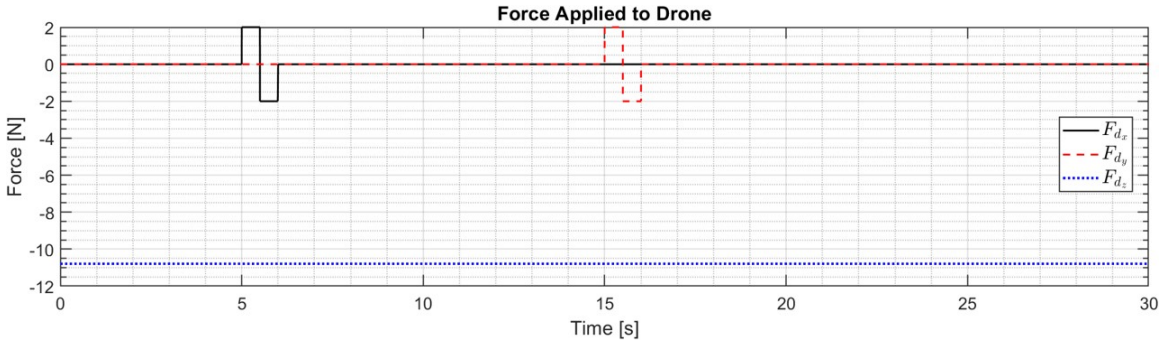


Figure 3.4: Force applied the drone-payload model. (Note: + z-axis is down)

The resulting position and velocity of the drone and payload are shown in Figure 3.5 and Figure 3.6, respectively. When the positive force is applied to the drone in its x-axis, the drone's velocity increases in the x-axis linearly and the drone comes to a stop when the same magnitude force is applied in the negative x direction. During this motion, and even after the drone is stationary, the payload continues to oscillate in the x-axis. The oscillation decreased over time since I added viscous damping to the model. The same observations are made when a force is applied to the drone along its y-axis. The viscous damping coefficient (b), from Equation 3.22, was experimentally approximated to be 0.0073. I set up an experiment using a plumb bob and suspended it as a pendulum. Then, I recorded and analyzed its motion using the Vicon motion capture system to determine the damping constant.

Moreover, the swaying motion of the payload can be seen to induce oscillatory motion on the drone in all three axes. Since a constant thrust was applied to the drone (along the z-axis) the drone's z-position experiences small oscillations. This is expected since the tension in the cable is maximum when the payload is directly underneath the drone and minimum when the payload is at its maximum swing angle.

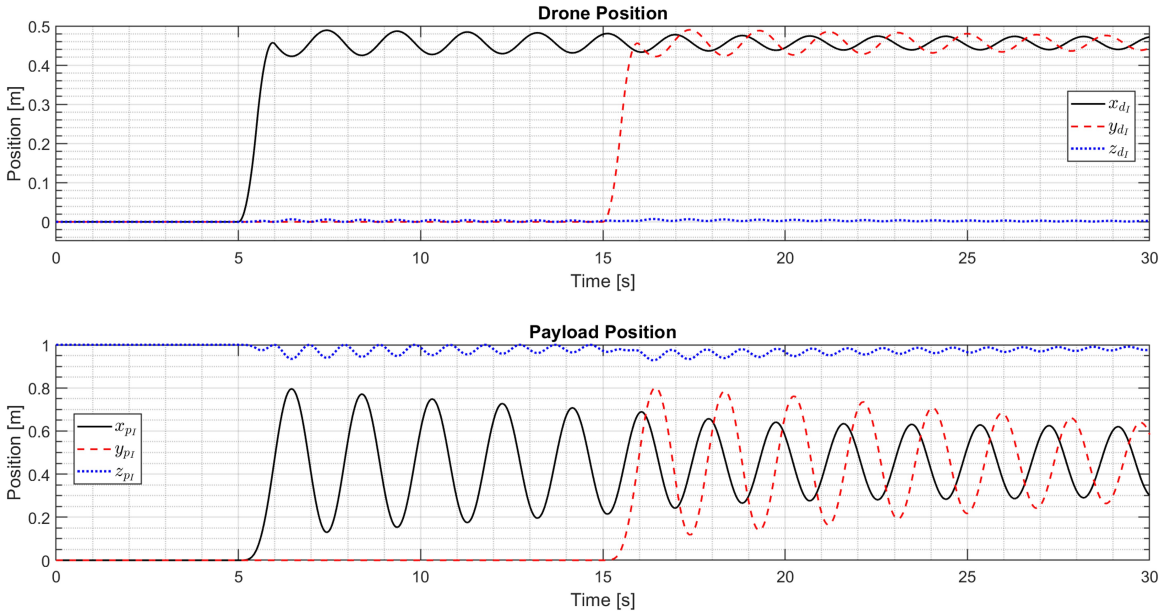


Figure 3.5: Simulated position of the drone-payload model. (Note: + z-axis is down)

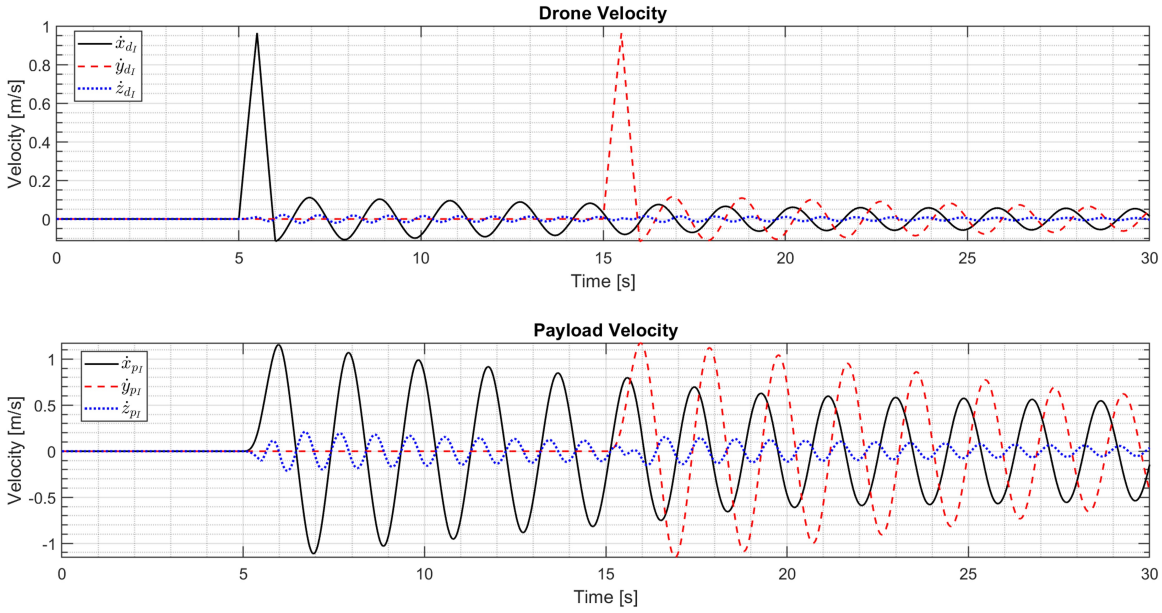


Figure 3.6: Simulated velocity of the drone-payload model. (Note: + z-axis is down)

Similar observations are made when I applied a force to the payload. Using initial conditions where the drone is hovering and both the drone and payload are stationary, I applied the forcing function shown in Figure 3.7 to the payload. The direct forces applied to the drone along the x-axis and y-axis (F_{d_x}, F_{d_y}) were zero. A constant hovering force was applied to the drone (similar to F_{d_z} in Figure 3.4).

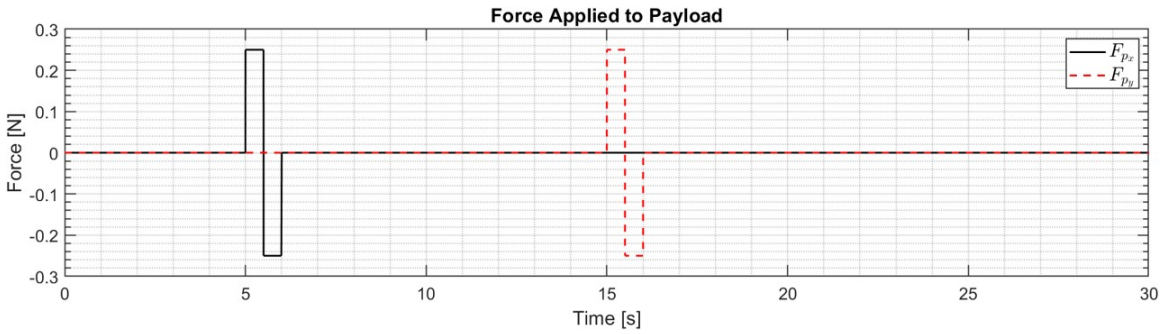


Figure 3.7: Force applied directly to the payload. (Note: + z-axis is down)

The resulting position and velocity of the drone and payload are shown in Figure 3.8 and Figure 3.9, respectively. The swaying motion of the payload mimics a pendulum's dynamics and can be seen to decay over time as a result of viscous damping. Also, the motion of the payload can be seen to affect the motion of the drone showing the coupled dynamics of the drone-payload model. This verifies that the drone-based cable-suspended model is derived correctly and performs as expected.

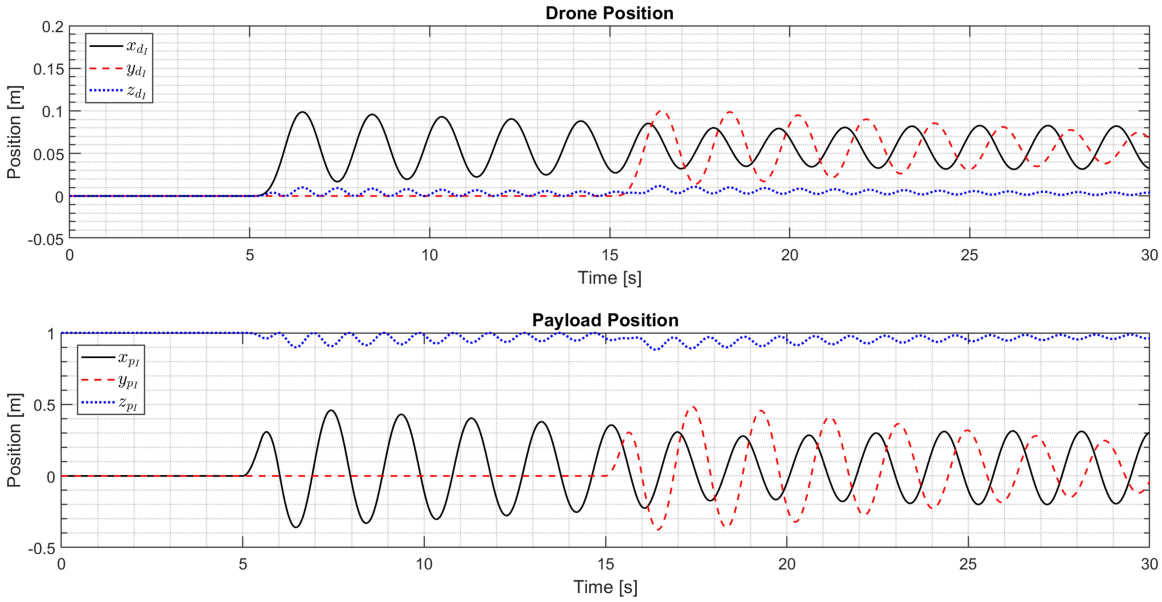


Figure 3.8: Simulated position of the drone-payload model with direct forces applied to the payload. (Note: + z-axis is down)

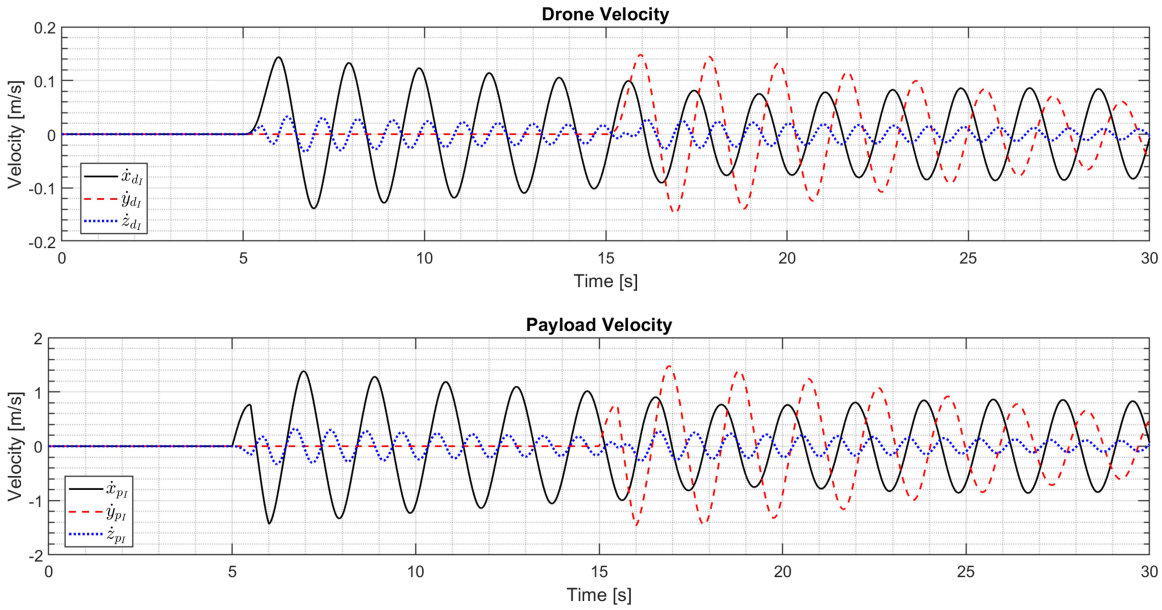


Figure 3.9: Simulated velocity of the drone-payload model with direct forces applied to the payload. (Note: + z-axis is down)

3.4 Chapter Summary

To summarize this chapter, I developed a mathematical model of a 3D pendulum and a mathematical model of a payload attached to a drone via a single cable using Lagrangian dynamics. I used MATLAB's Symbolic Toolbox to derive the equations of motion for both models. Simulation results of both models showed that both models performed as expected. Further, I also used the drone-payload model to briefly study the coupled effects between the drone and the payload. I used both mathematical models in Chapter 4 to develop the LiDAR and Extended Kalman Filter tracking algorithm. In Chapter 6, I used the drone-payload mathematical model to tune the candidate controllers and simulate the response of the drone-payload system.

Chapter 4

LiDAR Tracking and Payload Position Estimation[1]

This chapter provides details on the Light Detection and Ranging (LiDAR) sensor hardware and the algorithm used to locate the centroid of an object using the LiDAR's point cloud. First, the algorithm is tested using a simple 3D pendulum experimental setup and the 3D pendulum model from Section 3.1. Then, a LiDAR noise model is developed that is used to simulate LiDAR measurements. The simulated measurements and an Extended Kalman Filter (EKF) are used to test the dynamics model developed in Section 3.2. Experimental setup and test results are presented and discussed.

[1]Portions of this section have been reproduced with permission from:

[1] Patel, M., and Ferguson, P., "Tracking and Estimation of a Swaying Payload Using a LiDAR and an Extended Kalman Filter," 2021 IEEE ROSE. ©2021 Mitesh Patel and Philip Ferguson.

[2] Mitesh S. Patel and Philip A. Ferguson. "Drone-Based Cable-Suspended Payload Tracking and Estimation Using Simulated LiDAR Measurements and an Extended Kalman Filter," AIAA 2022-2290. AIAA SCITECH 2022 Forum. ©2022 Mitesh Patel and Philip Ferguson.

4.1 LiDAR Tracking

For this thesis, LiDAR tracking is needed to locate the position of the GPR relative to the drone, which can be used to geo-rectify the GPR data. Also, while not used in this thesis, the position of the GPR can serve as a control feedback for model-based controllers. The purpose of this research was to evaluate the performance of different feedback controllers (model-less methods) in the presence of a non-linear and unmodeled disturbance coming from the swaying GPR payload (per hypothesis H2). Therefore, LiDAR tracking is only required for geo-referencing the GPR data and not for control purposes.

4.1.1 LiDAR Hardware

I used the Velodyne VLP-16$^{\text{TM}}$ LiDAR [89] sensor, shown in Figure 4.1, in this study. The LiDAR uses an array of 16 offset laser beams to create a 3D point cloud of objects. The 16 laser beams are aligned across the x-axis of the LiDAR, which results in the resolution of the LiDAR in the x-direction being significantly less than that in the y-direction and z-direction. The lasers spin about the x-axis at 600 rpm resulting in a higher resolution in the y-direction and z-direction. A 360^o rotation of the LiDAR's laser firings represents one LiDAR frame. The lasers on the LiDAR are angled between $\pm 15^o$ along the x-axis of the LiDAR which results in a further loss of resolution in the x-axis as the distance from the LiDAR increases.

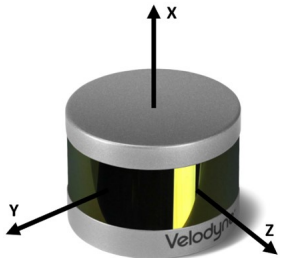


Figure 4.1: Velodyne VLP-16 LiDAR sensor [89].

Figure 4.2 shows a sample of the point cloud generated by the VLP-16 LiDAR. The figure shows one LiDAR frame of a box suspended as a simple 3D pendulum. The 16 lines represent the 16 lasers while the different colours represent the relative intensity of the reflected rays.

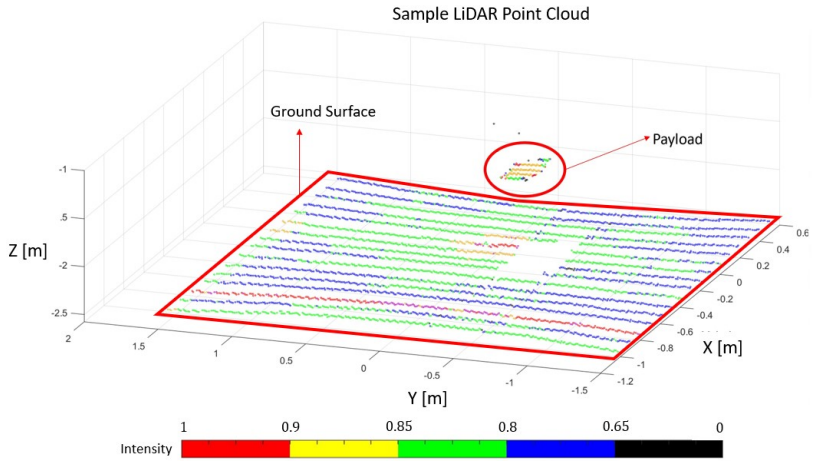


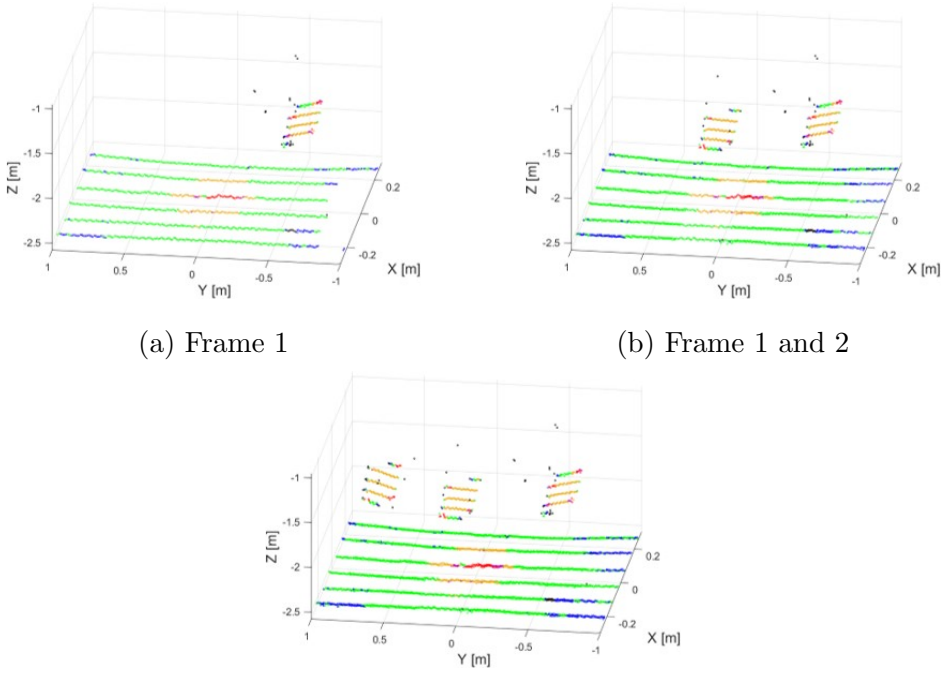
Figure 4.2: Sample LiDAR point cloud from VLP-16 LiDAR sensor.

4.1.2 LiDAR Tracking Algorithm

As shown in Figure 4.2, the LiDAR data contains the point cloud of the ground below the payload as well as the payload itself. Therefore, the tracking algorithm I developed first identifies the point cloud representing the payload and uses that data to estimate the centroid of the payload in 3D space.

Figure 4.3 shows three LiDAR frames at three separate instances of a box swinging as a pendulum, with the LiDAR attached at the base of the pendulum (the origin of the inertial frame shown in Figure 3.1). To isolate the point cloud representing the payload, all point cloud data representing the ground need to be eliminated. Since the maximum z-distance that the LiDAR should measure is when the box is directly underneath the LiDAR (*i.e.*, the cable length), I eliminated all data points with a z-measurement greater than the length of the cable. This reduced the total point

cloud to the point cloud of the payload, with a small amount of noise introduced by the footprint of the cable. Figure 4.4 shows the point cloud data representing the payload after elimination of the ground data from Figure 4.3.



(a) Frame 1

(b) Frame 1 and 2

(c) Frame 1, 2, and 3

Figure 4.3: Three LiDAR frames along the trajectory of a swinging pendulum.

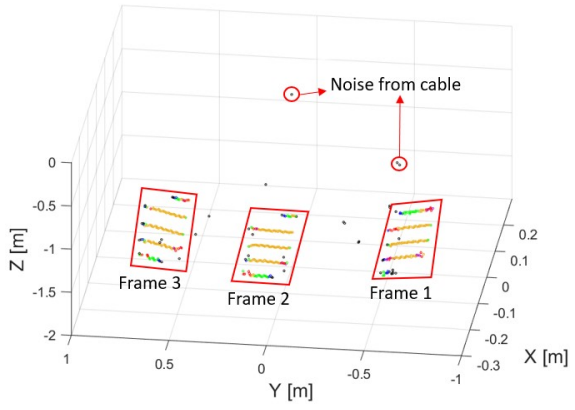


Figure 4.4: Payload-only point cloud.

With the LiDAR data reduced to contain only the point cloud of the payload, I determined the centroid of the box using k-means clustering, per Equation 4.1. I separated the payload point cloud into three vectors with one vector containing the x-coordinates of all data points, one vector containing the y-coordinates of all data points, and one vector containing the z-coordinates of all data points. Then, I used k-means clustering to determine the centroid in all three axes by partitioning N observations into k data sets (S) such that $S = (S_1,\ S_2,\ ...,\ S_k)$, and minimizing the variance. I used MATLAB's *kmeans* function to determine the centroid of the payload.

$$\min_{S} \sum_{i=1}^{k} \sum_{X \in S_i} |\mathbf{X} - \boldsymbol{\mu_i}|^2 \tag{4.1}$$

where $\boldsymbol{\mu_i}$ is a vector containing the mean of all points in the x, y, and z coordinates and $\mathbf{X}$ is a vector containing the x, y, and z coordinates of all points representing the box in a given LiDAR frame. Figure 4.5 shows the corresponding location of the centroid for each of the sample frames in 3D space.

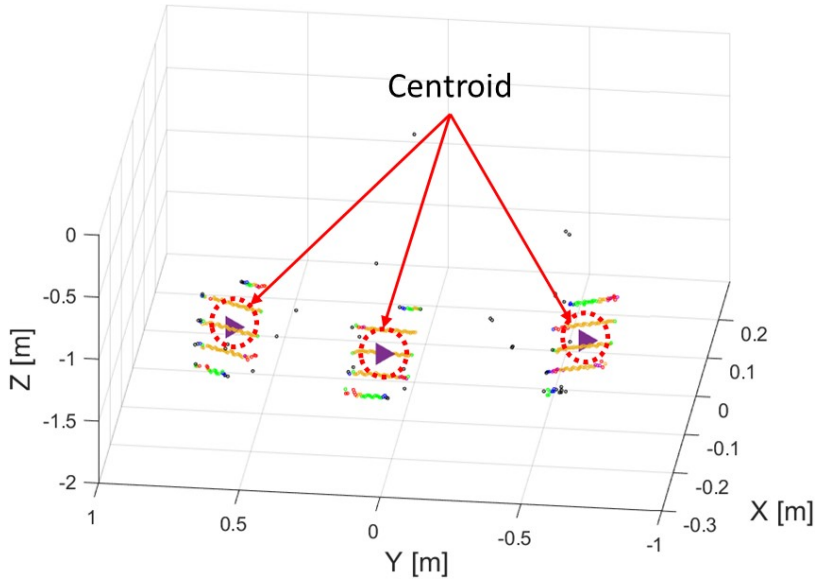


Figure 4.5: Payload point cloud with centroid estimates using k-means clustering.

4.2 Discrete Extended Kalman Filter

A Kalman Filter is a powerful mathematical estimator that has been extensively researched and used for estimation from noisy sensor measurements for a linear system [90]. An extension of the Kalman Filter is the discrete Extended Kalman Filter (EKF) which is used for state estimation of non-linear discrete-time controlled processes [91].

For this thesis, I proposed a discrete EKF since the drone-payload dynamics are non-linear. I used the non-linear drone-based cable-suspended payload dynamics I developed in Section 3.2 to predict the theoretical position of the payload, and I used the LiDAR measurement to correct the theoretical prediction.

The discrete EKF consists of two steps, a time update step, and a measurement update step. Figure 4.6 shows a flow chart summarizing the operation of an EKF. The following summarizes the flow of an EKF.

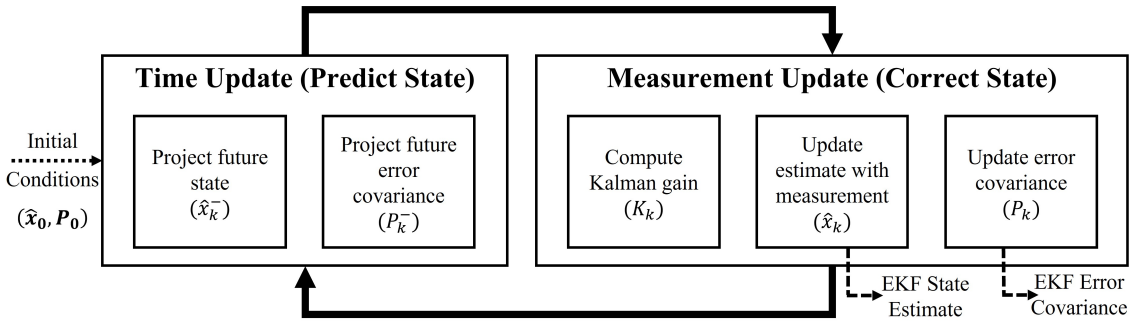


Figure 4.6: Extended Kalman Filter (EKF) working principle.

During the update stage, the state and covariance are propagated from the previous time step $(k-1)$ to the current time step (k) to obtain the *a priori* estimates. To calculate the *a priori* estimates at each time step, the EKF first linearizes the state dynamics about the previous time step $(k-1)$. Initial conditions are used for the first time step of the EKF, after which the *a posteriori* estimates are used to predict a new *a priori* state [90].

For a given system, the non-linear continuous dynamics can be described as a function of the state vector ($\mathbf{x}$), the control inputs ($\mathbf{u}$), and the process noise ($\mathbf{W}$), per Equation 4.2.

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{W}) \tag{4.2}$$

I assumed the control inputs ($\mathbf{u}$) to be constant over the duration of time step k. The process noise ($\mathbf{W}$) is a vector that contains the standard deviation of the disturbances that result from unmodeled dynamics for each state and can be represented as a continuous process noise covariance ($\mathbf{Q}$) using Equation 4.3. Note that $\mathbf{Q}$ is a diagonal matrix that contains the variance of the process noise $\mathbf{W}$.

$$\mathbf{Q} = \mathbf{W}\mathbf{W}^T \tag{4.3}$$

The linearized form of the dynamics can be expressed using Equation 4.4.

$$\dot{\mathbf{x}} = \mathbf{G}\mathbf{x} + \mathbf{B}(\mathbf{u} + \mathbf{W}) \tag{4.4}$$

where the matrix $\mathbf{G}$ represents the linearized dynamics and is determined using the Jacobian of the non-linear dynamics (f) with respect to the state vector (x), as shown in Equation 4.5. The matrix $\mathbf{B}$, which relates the control input to the state, is linearized using the Jacobian of the non-linear dynamics with respect to the control input vector (u), per Equation 4.6.

$$\mathbf{G}[i,j] = \frac{\partial f_{[i]}}{\partial x_{[j]}}(\hat{x}_{k-1}, u_k, 0) \tag{4.5}$$

$$\mathbf{B}[i,j] = \frac{\partial f_{[i]}}{\partial u_{[j]}}(\hat{x}_{k-1}, u_k, 0) \tag{4.6}$$

Then, as a result of the linearization step, the linearized state transition matrix ($\mathbf{G}$) and the continuous process noise covariance ($\mathbf{Q}$) need to be discretized and

recomputed at each time step during the time update phase. Since the linearized process is time-varying, numerical methods are required. The van Loan numerical method [92] is used for the discretization step, as follows:

1. Form the matrix $\mathbf{E}$ as shown in Equation 4.7.

$$\mathbf{E} = \begin{bmatrix} -\mathbf{G} & | & \mathbf{B}\mathbf{Q}\mathbf{B}^T \triangle t_s \\ - - - - & | & - - - - \\ 0 & | & \mathbf{G}^T \end{bmatrix} \triangle t_s \tag{4.7}$$

   where $\triangle t_s$ is the time step. The constant process noise results from the uncertainty in the dynamics model such as the effects of neglecting aerodynamic drag, downwash from the drone's propellers, and effects of using a flexible cable.

2. Perform the following operation, $e^{\mathbf{E}}$. In MATLAB, use the *expm* function to perform matrix exponential.

$$\mathbf{C} = \text{expm}(\mathbf{E}) = \begin{bmatrix} ... & | & \mathbf{A}_k^{-1}\mathbf{Q}_k \\ - - - - & | & - - - - \\ 0 & | & \mathbf{A}_k^T \end{bmatrix} \tag{4.8}$$

3. The transpose of the lower-right portion of $\mathbf{C}$ is $\mathbf{A}_k$, which represents the linearized and discretized state transition matrix.

4. The discretized process noise covariance matrix, $\mathbf{Q}_k$, can be obtained using the upper-right portion of matrix $\mathbf{C}$.

$$\mathbf{Q}_k = \mathbf{A}_k * (\text{upper right part of } \mathbf{C}) \tag{4.9}$$

After the discretization step, the linearized state transition matrix ($\mathbf{G}$) is discretized into matrix $\mathbf{A}_k$. Then, $\mathbf{A}_k$ and $\mathbf{Q}_k$ are used to calculate the *a priori* estimate error covariance ($\mathbf{P_k^-}$) in Equation 4.10.

$$\mathbf{P}_k^- = \mathbf{A}_k \mathbf{P}_{k-1} \mathbf{A}_k^T + \mathbf{Q}_{k-1} \tag{4.10}$$

where $\mathbf{A}_k$ is the discretized and linearized state transition matrix that relates the previous state at time $k-1$ to the current state at time k. $\mathbf{Q}_{k-1}$ is the discretized and linearized process noise covariance matrix which is modeled as a zero mean Gaussian uncertainty for this filter.

During the measurement update phase, the *a priori* estimate is updated using the drone's and payload's position measurements ($\mathbf{Z}_k$). The Kalman gain ($\mathbf{K}_k$), that performs a weighted average between the measurement and the *a priori* estimate, is calculated using Equation 4.11. Lastly, the *a posteriori* state estimates and error covariance are calculated using Equation 4.12 and Equation 4.13, respectively [90].

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k)^{-1} \tag{4.11}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{Z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-) \tag{4.12}$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^- (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)^T + \mathbf{K}_k \mathbf{R}_k \mathbf{K}_k^T \tag{4.13}$$

where $\mathbf{P}_k$ is the *a posteriori* estimate error covariance matrix calculated using the Joseph form [93], and $\mathbf{I}$ is an identity matrix. $\mathbf{R}$ is the measurement noise covariance, representing the error in measuring the position of the payload and the drone. $\mathbf{H}$ is the linearized form of the measurement equation h which relates the *a posteriori*

state estimates ($\hat{\mathbf{x}}_k$) to the measurements ($\mathbf{Z}_k$). The non-linear h is defined such that:

$$\mathbf{Z}_k = h(\mathbf{x}_k) \tag{4.14}$$

Then, the linearized measurement matrix $\mathbf{H}$ can be calculated using the Jacobian of h as shown in Equation 4.15.

$$\mathbf{H}_{[i,j]} = \frac{\partial h_{[i]}}{\partial x_{[j]}}(\hat{x}_k, 0) \tag{4.15}$$

4.3 LiDAR Tracking Results and Discussion

This section describes the experimental setup used to test and validate the LiDAR tracking and EKF algorithm. Two tests were conducted. The first test uses a 3D pendulum setup to test the k-means clustering algorithm with an EKF. The second test uses a Quanser QDrone with a cable-suspended payload to test the drone-based cable-suspended payload dynamics with an EKF.

4.3.1 LiDAR Tracking Experimental Setup and Results for a 3D Pendulum

4.3.1.1 LiDAR Centroid Measurement Verification

To validate the tracking algorithm's performance for a simple 3D pendulum, I conducted an experiment in an indoor environment equipped with a Vicon motion capture system. The Vicon measurements were captured at a frequency of $\sim 140\ Hz$. I used the Vicon measurements as the truth measurements for this experiment to verify the tracking algorithm results. The test setup is shown in Figure 4.7. Note that I used four cables for this test to prevent the box from spinning to get accurate Vicon data. While the setup did not use a single cable, given that all four cables attach to the

same location on the drone and that the box was approximately 20 cm wide, the system dynamics should be comparable to a single cable.

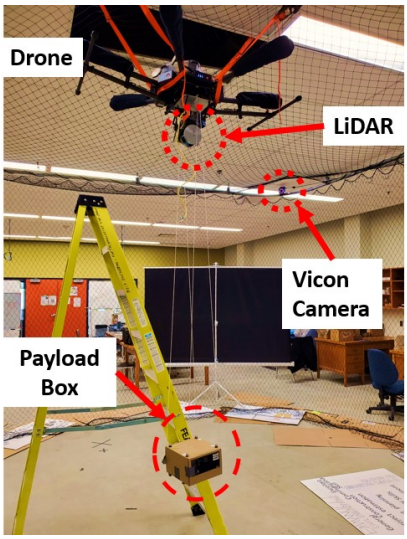


Figure 4.7: Experimental setup used to test the LiDAR tracking algorithm.

A drone carrying the LiDAR was fixed to the ceiling and a box was suspended from the drone using a cable. The LiDAR was attached to the drone in the same orientation as the inertial reference frame ($\vec{\mathcal{F}}_I$) used to derive the 3D pendulum dynamics (in Section 3.1). The parameters of the payload and initial conditions to initiate the motion of the box were as follows:

$$\begin{aligned} L &= 1.71\ m & x_{p_I} &= 0\ m & \dot{x}_{p_I} &= 0\ m/s \\ m &= 0.65\ kg & y_{p_I} &= 0.65\ m & \dot{y}_{p_I} &= 0\ m/s \end{aligned} \tag{4.16}$$

Then, I collected LiDAR point cloud and Vicon data for 22 seconds. I ran the LiDAR point cloud data through the tracking algorithm presented in Section 4.1.2 and verified the LiDAR centroid measurements using the Vicon measurements. Figure 4.8, Figure 4.9, and Figure 4.10 show an overlay of the LiDAR centroid measurement using k-means clustering and the position of the payload as recorded by the Vicon motion capture system.

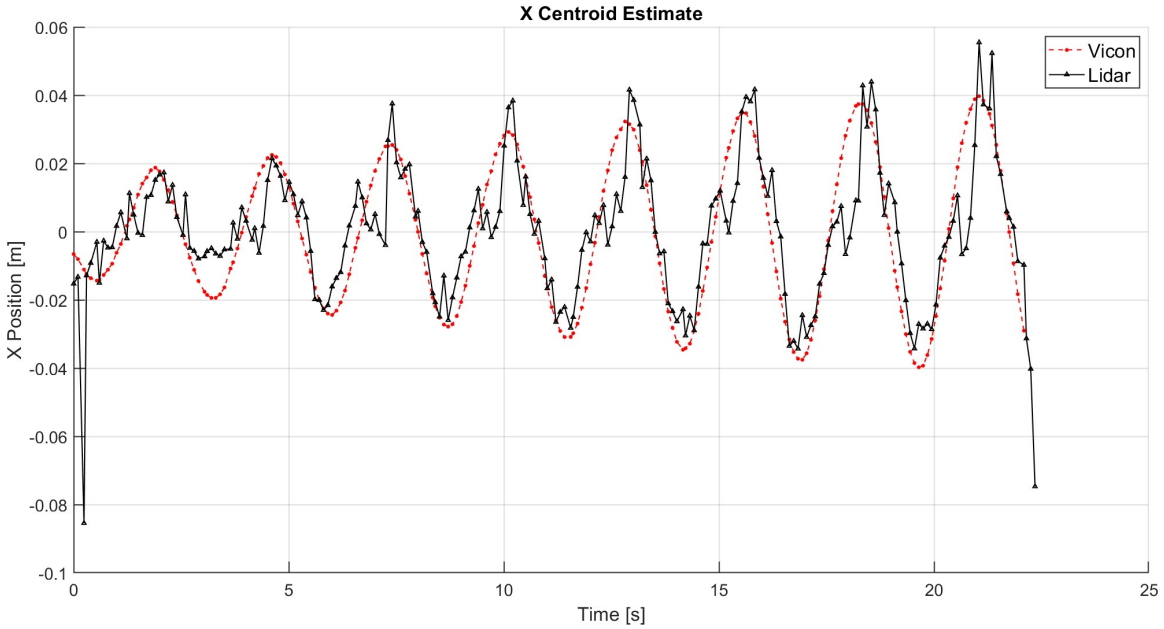


Figure 4.8: X-centroid position as measured by LiDAR and Vicon.

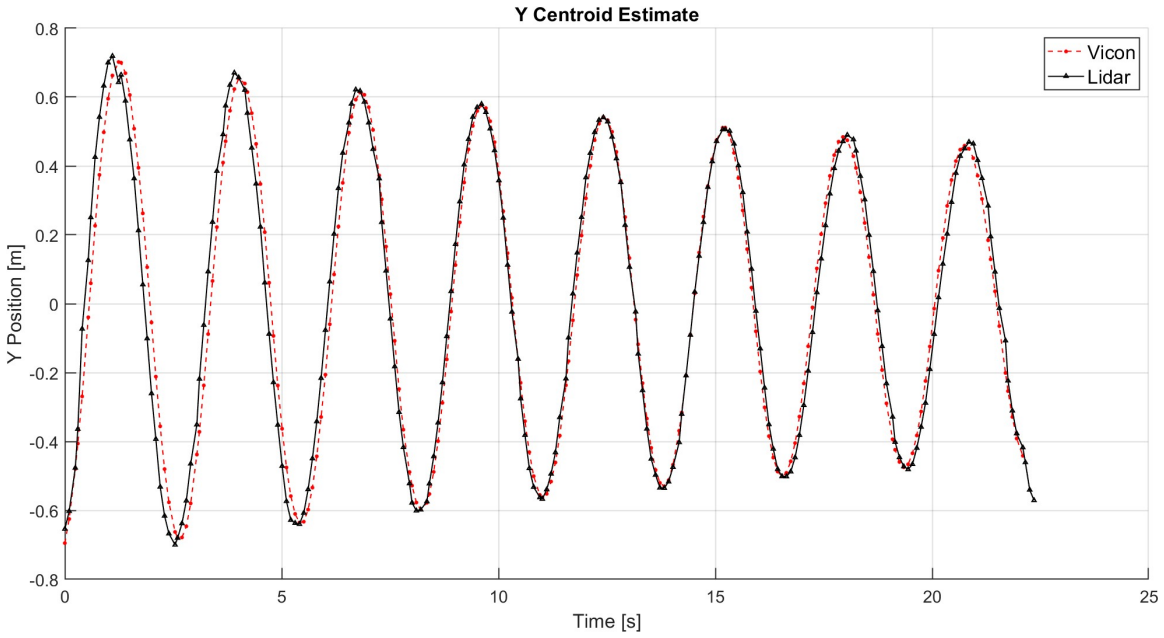


Figure 4.9: Y-centroid position as measured by LiDAR and Vicon.

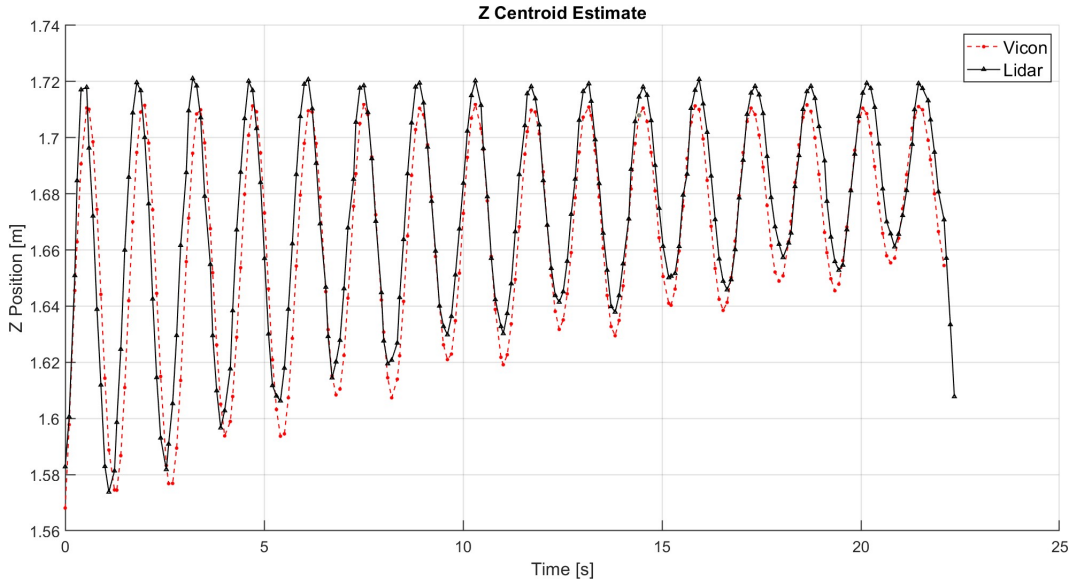


Figure 4.10: Z-centroid position as measured by LiDAR and Vicon.

For the measurements in all three coordinates (x, y, and z), the variation in Vicon frequency results in a varying time step. The Vicon system does not capture data at a constant frequency, but rather tries to maintain a set capture frequency. The fluctuations in the capture frequency result from the lack of a real-time operating system on the base computer and a finite data transfer time from the Vicon cameras to MATLAB's interface. Further, since the Vicon data is not timestamped, it is difficult to determine the exact time interval between successive measurements, which results in a time synchronization issue. To reduce the effect of the synchronization issue, I interpolated the Vicon data and refitted the data using a fixed time step. This did not solve the synchronization completely but it did result in more consistent Vicon data.

Despite the synchronization issue, the LiDAR results agree with Vicon measurements since fluctuations in the frequency are small. The variation in Vicon's frequency can be overcome in future experiments using Vicon's Lock Lab synchronization box [94].

Overall, the LiDAR tracking algorithm using k-means clustering performed well for the given experimental setup. The centroid estimates in the y-coordinates and z-coordinates closely matched the Vicon measurements, taking into consideration Vicon's time synchronization issue. The error in the x-centroid measurement was higher than the y-centroid and z-centroid measurements which I expected due to quantization of the x-centroid measurement.

4.3.1.2 EKF Estimate for 3D Pendulum Test Setup

I incorporated the LiDAR measurements from Section 4.3.1.1, and the 3D pendulum dynamic model from Section 3.1, into the EKF. I used the EKF to estimate four states of the pendulum mass: two positions (x_{p_I} and y_{p_I}), and two velocities ($\dot{x}_{p_I}$ and $\dot{y}_{p_I}$).

$$\mathbf{x} = \begin{bmatrix} x_{p_I} \\ y_{p_I} \\ \dot{x}_{p_I} \\ \dot{y}_{p_I} \end{bmatrix} \tag{4.17}$$

The initial state estimates for the EKF were from Equation 4.16. I determined the linearized state transition matrix using the Jacobian, as shown in Equation 4.18. I linearized the dynamics by evaluating $\mathbf{G}$ using the state estimate from the previous time step.

$$\mathbf{G} = \begin{bmatrix} \frac{\partial \dot{x}_{p_I}}{\partial x_{p_I}} & \frac{\partial \dot{x}_{p_I}}{\partial y_{p_I}} & \frac{\partial \dot{x}_{p_I}}{\partial \dot{x}_{p_I}} & \frac{\partial \dot{x}_{p_I}}{\partial \dot{y}_{p_I}} \\ \frac{\partial \dot{y}_{p_I}}{\partial x_{p_I}} & \frac{\partial \dot{y}_{p_I}}{\partial y_{p_I}} & \frac{\partial \dot{y}_{p_I}}{\partial \dot{x}_{p_I}} & \frac{\partial \dot{y}_{p_I}}{\partial \dot{y}_{p_I}} \\ \frac{\partial \ddot{x}_{p_I}}{\partial x_{p_I}} & \frac{\partial \ddot{x}_{p_I}}{\partial y_{p_I}} & \frac{\partial \ddot{x}_{p_I}}{\partial \dot{x}_{p_I}} & \frac{\partial \ddot{x}_{p_I}}{\partial \dot{y}_{p_I}} \\ \frac{\partial \ddot{y}_{p_I}}{\partial x_{p_I}} & \frac{\partial \ddot{y}_{p_I}}{\partial y_{p_I}} & \frac{\partial \ddot{y}_{p_I}}{\partial \dot{x}_{p_I}} & \frac{\partial \ddot{y}_{p_I}}{\partial \dot{y}_{p_I}} \end{bmatrix}_{\mathbf{x}_{k-1}} \tag{4.18}$$

I determined the measurement vector, containing the LiDAR measurements, using Equation 4.19 where x_L and y_L are the LiDAR's x_{p_I} and y_{p_I} measurements, respectively.

$$\mathbf{Z}_k = \mathbf{H}\left[x_L,\ y_L,\ \dot{x}_L,\ \dot{y}_L\right]^T \tag{4.19}$$

Since h was linear and I measured only x_{p_I} and y_{p_I} (the LiDAR does not measure the velocity of the payload), the matrix $\mathbf{H}_k$ is given by:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{4.20}$$

I approximated the variance in the process noise ($\mathbf{Q}$) using Equation 4.33, which represents the drag force acting on the payload.

$$F_{noise} = \frac{\rho_a}{2} C_d A_f V_a^2 \tag{4.21}$$

where ρ_a is the density of air at room temperature, V_a is the velocity of the air and A_f is the frontal area of the object. The constant C_d is the drag coefficient of the object. I assumed the payload to have a drag coefficient similar to a cube. The resulting continuous process noise covariance ($\mathbf{Q}$) is shown in Equation 4.22. The bounded process noise also accounts for minor drafts that result from the room's ventilation system. The matrix $\mathbf{B}$ relating the process noise to the state vector is given by Equation 4.23.

$$\mathbf{Q} = diag\left(\begin{bmatrix} 0 \\ 0 \\ F_{x_{noise}} \\ F_{y_{noise}} \end{bmatrix}\right) = diag\left(\begin{bmatrix} 0 \\ 0 \\ 9 \times 10^{-2} \\ 9 \times 10^{-2} \end{bmatrix}\right) N^2 \tag{4.22}$$

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{m} & 0 \\ 0 & 0 & 0 & \frac{1}{m} \end{bmatrix} \tag{4.23}$$

I approximated the measurement noise variance ($\mathbf{R}$) using Equation 4.24 and assumed it to be constant. x_{noise} and y_{noise} represent the variance of the x_{p_I} and y_{p_I} measurements from the LiDAR. I used a larger error for the x_{p_I} measurement due to the low resolution of the LiDAR in the x-axis. I approximated the measurement noise as the distance between two successive data points in each axis of the LiDAR's point cloud when the LiDAR rotation speed was set to 600 rpm.

$$\mathbf{R} = diag\left(\begin{bmatrix} x_{noise} \\ y_{noise} \end{bmatrix}\right) = diag\left(\begin{bmatrix} 1 \times 10^{-2} \\ 2.5 \times 10^{-3} \end{bmatrix}\right) m^2 \tag{4.24}$$

Then, I ran the EKF using the LiDAR's time step. The resulting EKF estimates for the payload's x_{p_I} and y_{p_I} positions, and the Vicon truth measurements, are shown in Figure 4.11 and Figure 4.12, respectively.

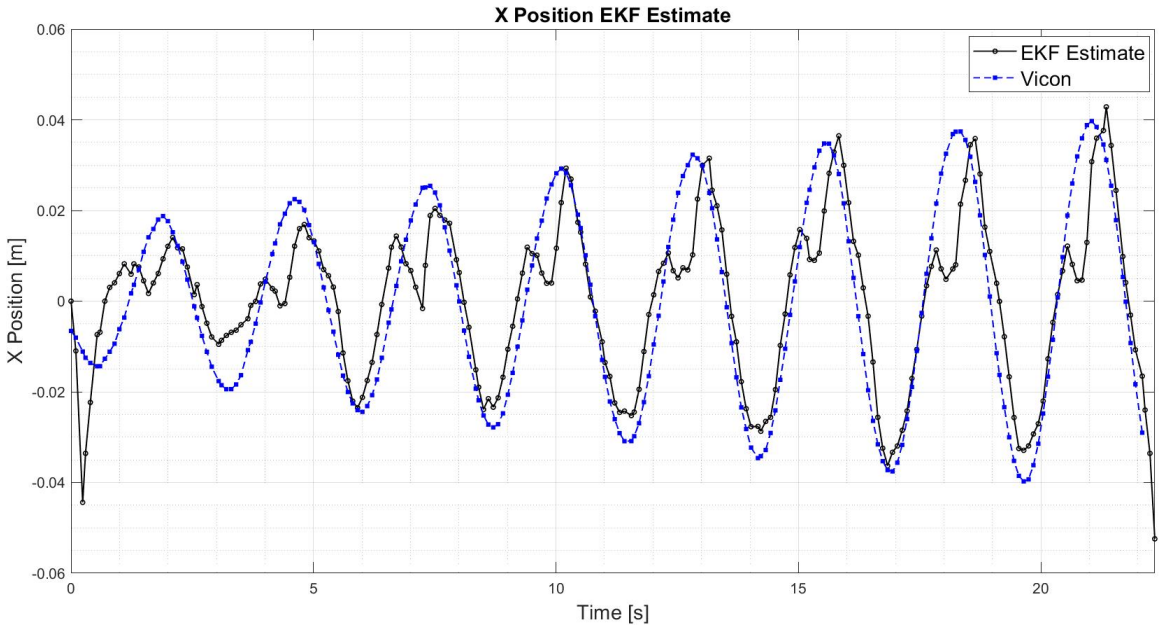


Figure 4.11: EKF x_{p_I} position estimate with Vicon truth measurement.

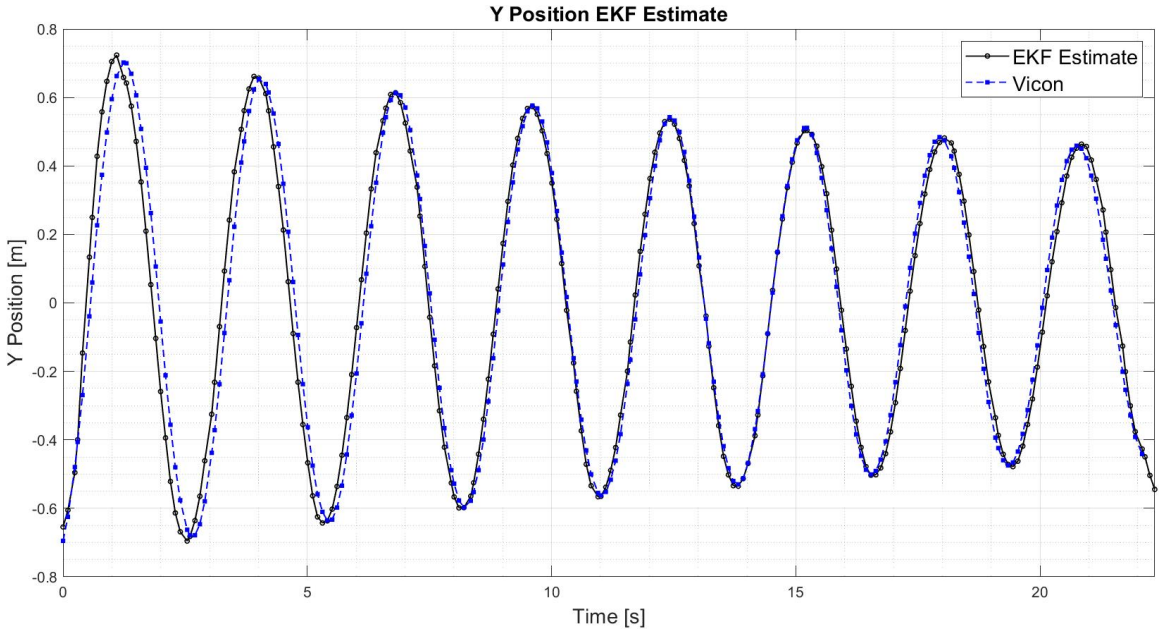


Figure 4.12: EKF y_{p_I} position estimate with Vicon truth measurement.

The EKF, using the LiDAR measurements and the 3D pendulum dynamics model, performed well for the given experimental setup. Comparing the EKF estimate to the Vicon truth measurements, it can be seen that the filter was able to track the position of the pendulum accurately. Furthermore, the filter reduced the impact of quantization in the x_{p_I} estimate.

The error between the EKF estimate and the Vicon truth measurement, for both x_{p_I} and y_{p_I}, is shown in Figure 4.13 and Figure 4.14, respectively. The positive and negative square roots of the diagonals of the *a posteriori* covariance estimate are plotted as well.

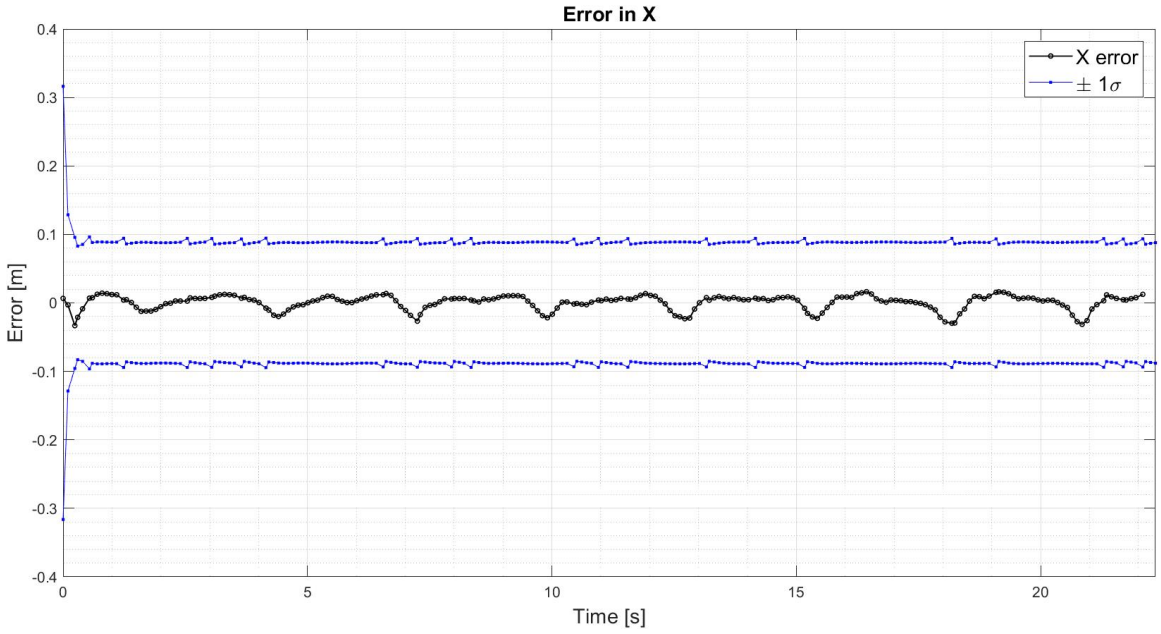


Figure 4.13: Error in EKF x_{p_I} estimate (using Vicon as truth measurement).

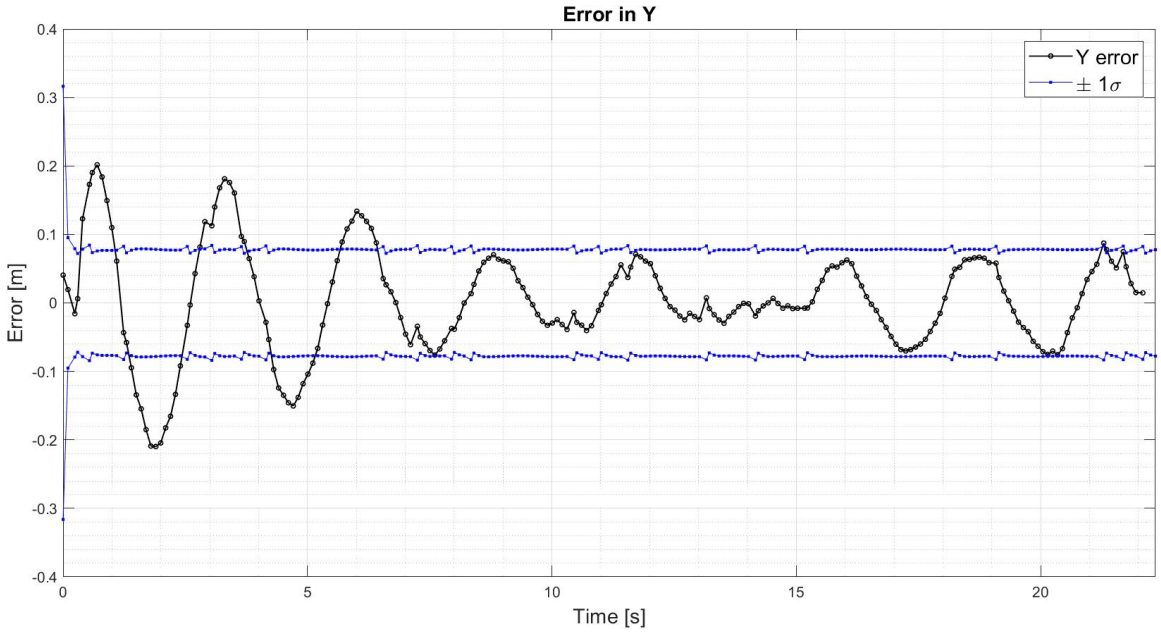


Figure 4.14: Error in EKF y_{p_I} estimate (using Vicon as truth measurement).

The majority of the estimated error in x_{p_I} is bound between $\pm 0.06\ m$ while the majority of the estimated error in y_{p_I} is bound between $\pm 0.035\ m$. The larger error in the y_{p_I} estimate at the start of the experiment results from the variation in the Vicon frame rate. For the x estimate, the filter was overestimating the error, while for the y estimate, the filter was sometimes overestimating and sometimes underestimating the error. This is an inherent issue for an EKF due to the linearization of the dynamics at each time step [95].

I derived the z-measurement (z_{p_I}) using Equation 3.1. Figure 4.15 shows the pendulum's z-position estimate computed using the pendulum's x_{p_I} and y_{p_I} estimates from the EKF. The pendulum's z-position estimate agreed with the measurements showing the efficacy of the proposed tracking and estimation system.

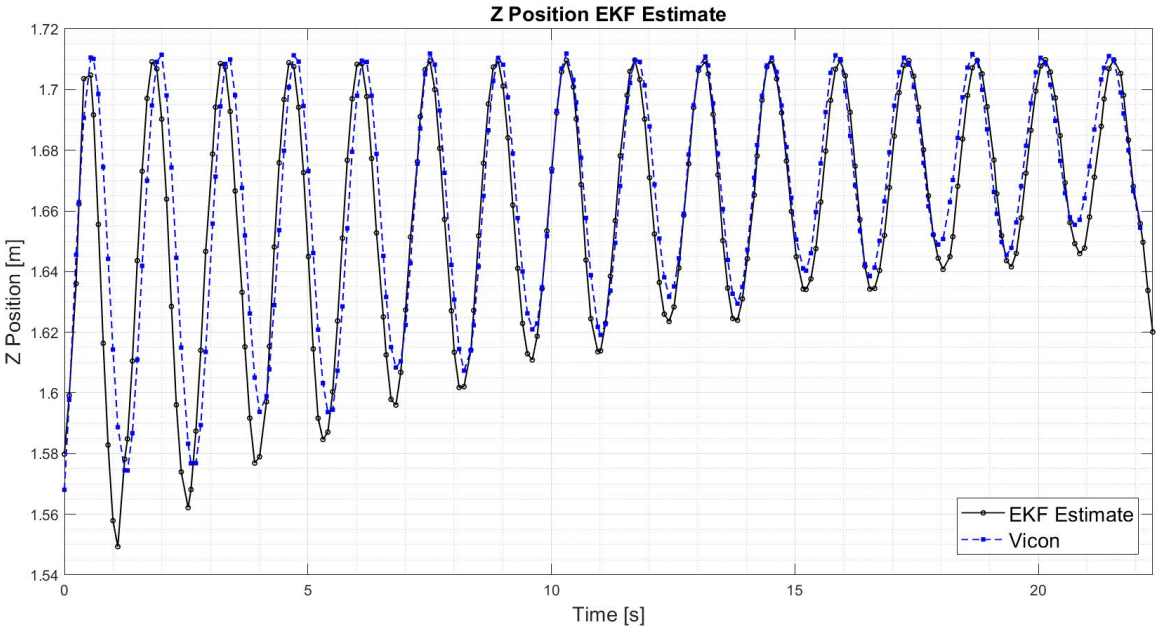


Figure 4.15: z_{p_I} position estimate with Vicon truth measurement.

4.3.2 LiDAR Tracking Experimental Setup and Results for a Drone-based Slung Payload

I conducted an experiment to validate the tracking algorithm's performance for a drone-based cable-suspended payload using a Quanser QDrone in an indoor environment equipped with a Vicon motion capture system. However, due to the low payload carrying capacity of the QDrone ($\sim 200\ g$), I could not mount the Velodyne VLP-16$^{\text{TM}}$ LiDAR sensor ($\sim 850\ g$) onto the drone. Hence, I used Vicon's motion capture system to measure the position of the payload, and added noise to the Vicon measurements to simulate the position measurements as if they were collected by the VLP-16$^{\text{TM}}$ LiDAR sensor. Then, I fed the simulated LiDAR measurements, alongside the dynamics model developed in Section 3.2, to an EKF to determine the viability of the LiDAR tracking system for a non-linear drone-based cable-suspended system.

4.3.2.1 LiDAR Measurement Simulation

As presented in Section 4.1.1, the resolution of the VLP-16$^{\text{TM}}$ LiDAR sensor varies in the x-direction and the y-direction of the sensor. The resolution of the LiDAR point cloud in the y-direction is significantly higher than in the x-direction, which results in a larger measurement noise in the LiDAR's x-centroid measurement.

I used experimental data from Section 4.3.1 and applied a line of best fit to the experimental data to superimpose representative LiDAR noise data onto the Vicon measurements. I fitted the x and y experimental data separately to get an accurate representation of the LiDAR's noise in different axes. I expect the dynamics of a lightly damped pendulum in 2D to follow harmonic motion. Hence, the line of best fit that I used of the generalized form shown in Equation 4.25.

$$\mathbf{X} = Ce^{-\alpha t}sin(At + B) \tag{4.25}$$

Using MATLAB's *fmincon* function, I determined the optimal parameters for α, A, B, and C by minimizing the variance of the difference between the experimental data and the line of best fit for all data points, as in Equation 4.26.

$$\min \sum_{i=1}^{k} (\mathbf{X} - \boldsymbol{\mu})^2 \tag{4.26}$$

where $\boldsymbol{\mu}$ is a vector containing the x and y position measurements from the LiDAR, and $\mathbf{X}$ is a vector containing the x and y estimates from the line of best fit for a given LiDAR measurement. Figure 4.16 shows the experimental data and the curve fit on the experimental data in the LiDAR's x-axis and y-axis.

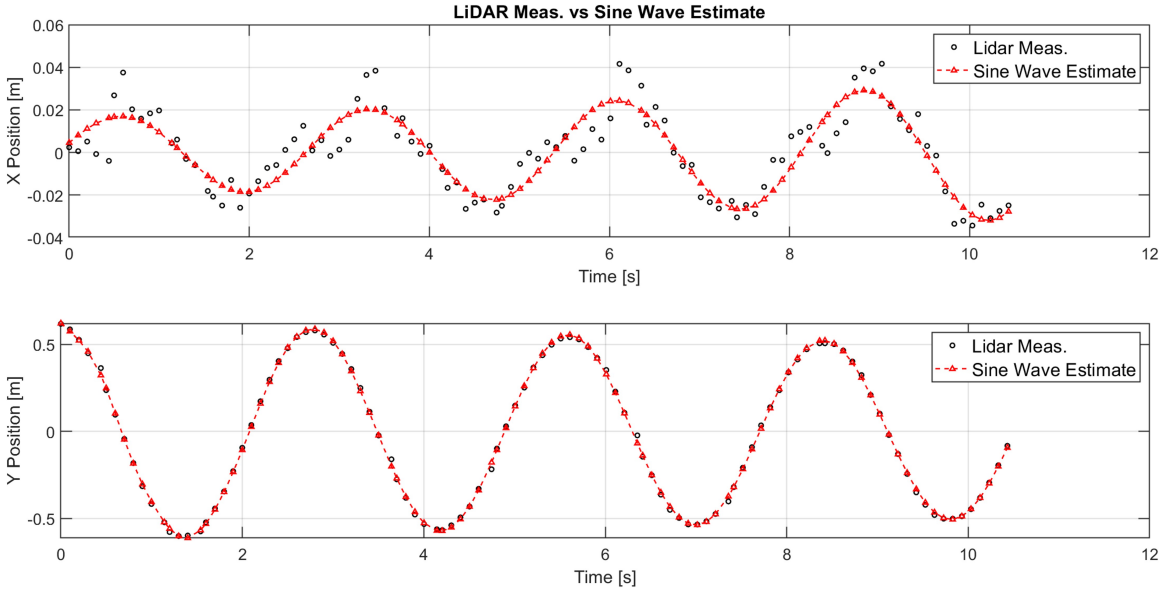


Figure 4.16: LiDAR measurement curve fitting.

The resulting LiDAR noise model in the x and y positions are shown in Figure 4.17. I repeated this noise model over time and applied it to the Vicon measurements to simulate the LiDAR measurements as if they were collected by a VLP-16$^{\text{TM}}$ LiDAR. Since the LiDAR's resolution in the z-direction matches that in the y-direction, I applied the noise model in the y-direction to the z-measurements.

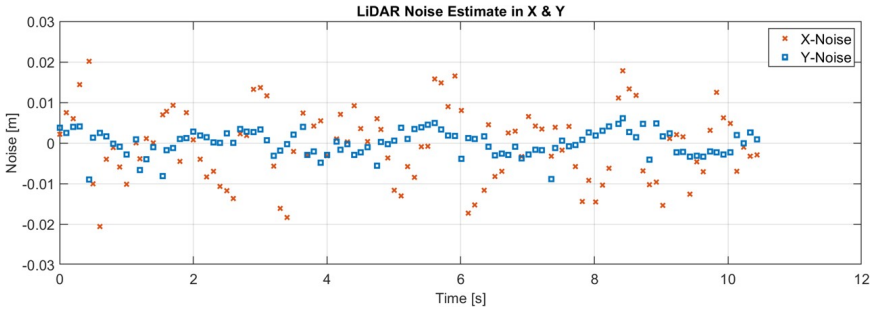


Figure 4.17: LiDAR noise estimate in x-measurement and y-measurement.

The first, second, and third moment for the LiDAR noise is summarized in Table 4.1. The results show that the skewness of the data is close to zero. This means that while not perfectly Gaussian, the LiDAR noise is close to Gaussian. Additionally, I used MATLAB's *kstest* and *adtest* functions to perform the statistical Anderson-Darling test [96] and Kolmogorov-Smirnov test [97–99] and verified that the noise was normally distributed. Hence, for the purpose of designing the EKF, the LiDAR noise was assumed to be Gaussian.

Table 4.1: LiDAR noise model statistical moments.

	x-axis	**y-axis**
First moment (mean)	5.78×10^{-4} $[m]$	4.33×10^{-8} $[m]$
Second moment (variance)	8.08×10^{-5} $[m^2]$	9.49×10^{-6} $[m^2]$
Third moment (skewness)	0.0362	-0.53

4.3.2.2 EKF Estimate for Drone-based Slung Payload Test Setup

I conducted an experiment using a Quanser QDrone, a plumb bob to represent the suspended payload, and a Vicon motion capture system to track the position of the drone and the suspended payload. Note that for this test, I did not implement a custom controller on the QDrone. The goal of this test was to validate the tracking system, hence, I used the default control system pre-designed by Quanser [100].

The experimental setup is shown in Figure 4.18. For the drone position measurements, I used raw measurements from the Vicon system. However, to simulate the position measurements of the payload as if they were captured by the Velodyne VLP-16$^{\text{TM}}$ LiDAR, I added the noise models, shown in Figure 4.17, to the raw payload position measurements recorded from Vicon.

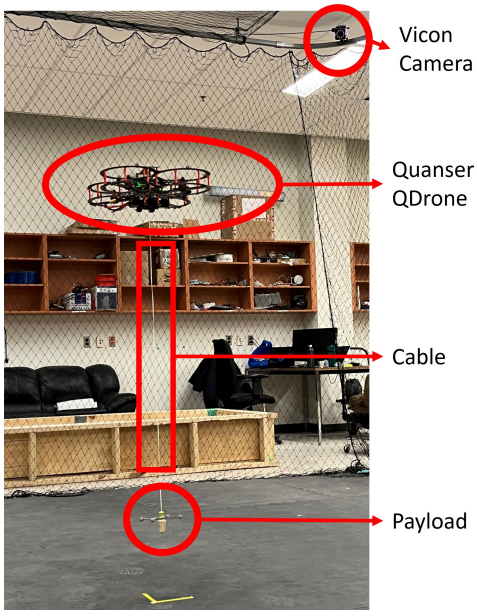


Figure 4.18: Experimental test setup for the drone-based slung payload tracking.

The experimental parameters and the initial conditions for the positions and velocities of the drone and payload are outlined below.

$$\begin{aligned} M &= 1.101\ kg & m &= 0.174\ kg & L &= 1.1\ m & \\ x_{d_I 0} &= 0\ m & \dot{x}_{d_I 0} &= 0\ m/s & x_{p_I 0} &= 0\ m, & \dot{x}_{p_I 0} &= 0\ m/s \\ y_{d_I 0} &= 0\ m & \dot{y}_{d_I 0} &= 0\ m/s & y_{p_I 0} &= 0\ m, & \dot{y}_{p_I 0} &= 0\ m/s \\ z_{d_I 0} &= -1.4\ m & \dot{z}_{d_I 0} &= 0\ m/s & & & & \end{aligned} \tag{4.27}$$

I started collecting the experimental data for this study when the drone was hovering steadily and had just lifted the payload off the ground surface. The position plots of the drone, in $\vec{\mathcal{F}}_I$, are shown in Figure 4.19.

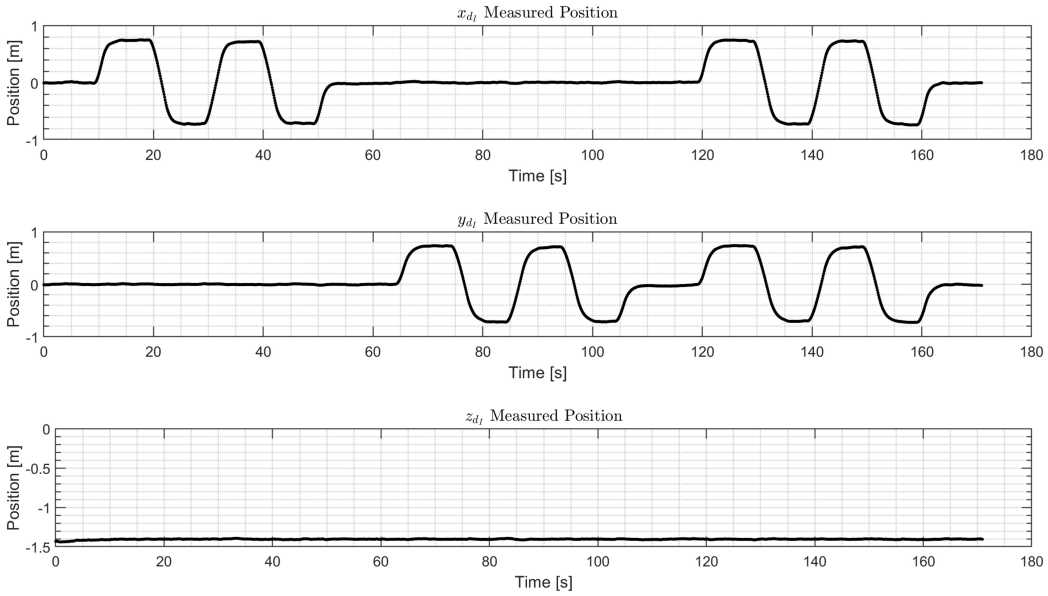


Figure 4.19: Drone position measurement (in the $\vec{\mathcal{F}}_I$ frame).

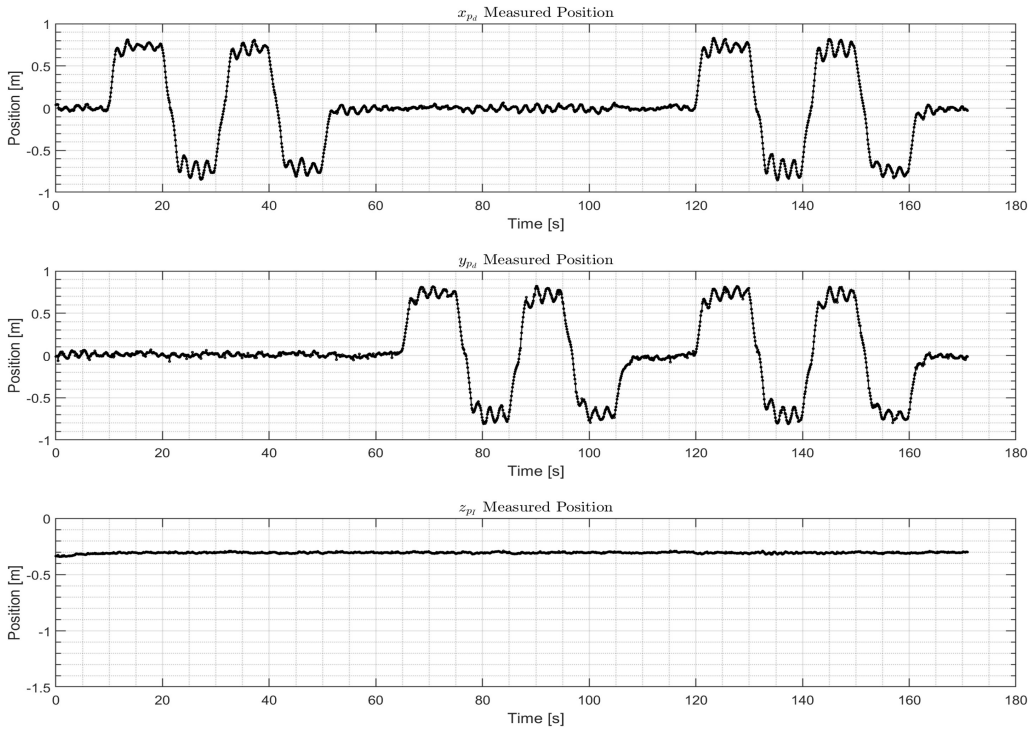


Figure 4.20: Payload position measurement (in $\vec{\mathcal{F}}_I$ frame).

I commanded the drone to move in the x-direction, then in the y-direction, and finally to move simultaneously in both the x-direction and y-direction. I commanded the drone to maintain a constant z-position throughout the experiment. Note that the drone's z-position decreases at the start of the experiment to indicate payload liftoff. The corresponding position measurements of the payload, with the added LiDAR noise, are shown in Figure 4.20.

Then, I used the drone's position measurements and the payload's position measurements in the EKF to estimate the ten states shown in Equation 4.28.

$$\mathbf{x} = \begin{bmatrix} x_{d_I},\ y_{d_I},\ z_{d_I},\ x_{p_I},\ y_{p_I},\ \dot{x}_{d_I},\ \dot{y}_{d_I},\ \dot{z}_{d_I},\ \dot{x}_{p_I},\ \dot{y}_{p_I} \end{bmatrix}^T \tag{4.28}$$

I determined the Jacobian matrix $\mathbf{G}$ using Equation 4.5. I used the control inputs, shown in Equation 4.29, to determine the linearized $\mathbf{B}$ matrix, per Equation 4.6.

$$\mathbf{u} = \begin{bmatrix} F_{d_x}, & F_{d_y}, & F_{d_z}, & F_{p_x}, & F_{p_y} \end{bmatrix}^T \tag{4.29}$$

I determined the position measurement vector ($\mathbf{Z}_k$) using Equation 4.30, where $\mathbf{H}$ is given by Equation 4.31 such that the measurement vector ($\mathbf{Z}_k$) only maps the position measurements of the drone and the payload since neither the velocity of the drone nor the velocity of the payload was measured. In matrix $\mathbf{H}$, the $\mathbf{I}_{5\times5}$ and $\mathbf{0}_{5\times5}$ represent a 5×5 identity matrix and zeroes matrix, respectively.

$$\mathbf{Z}_k = \mathbf{H} \begin{bmatrix} x_{d_I},\ y_{d_I},\ z_{d_I},\ x_{p_I},\ y_{p_I},\ \dot{x}_{d_I},\ \dot{y}_{d_I},\ \dot{z}_{d_I},\ \dot{x}_{p_I},\ \dot{y}_{p_I} \end{bmatrix}^T \tag{4.30}$$

$$\mathbf{H} = \begin{bmatrix} \mathbf{I}_{5\times5}\ ,\ \mathbf{0}_{5\times5} \end{bmatrix} \tag{4.31}$$

I approximated the measurement noise variance in the drone's position measurements and the payload's position measurements as in Equation 4.32. I used a larger noise variance for the payload position measurements than for the drone's position

measurements. This accounted for the larger noise in the simulated LiDAR measurements that would have resulted if the LiDAR sensor was used to measure the position of the payload. Further, for the payload position measurements, I applied a larger noise to the x-position measurements than the y-position measurements to account for the poor resolution in the LiDAR's x measurement. I estimated the variance in the drone's position approximated from the Vicon system's datasheet, while I estimated the variance in the payload's position as the variance of the noise model from Figure 4.17

$$\mathbf{R} = diag\left(\begin{bmatrix} x_{d_I\ variance} \\ y_{d_I\ variance} \\ z_{d_I\ variance} \\ x_{p_d\ variance} \\ y_{p_d\ variance} \end{bmatrix}\right) = diag\left(\begin{bmatrix} 1\times10^{-6} \\ 1\times10^{-6} \\ 1\times10^{-6} \\ 8\times10^{-5} \\ 9\times10^{-6} \end{bmatrix}\right) m^2 \tag{4.32}$$

I approximated the variance in the process noise ($\mathbf{W}$) using Equation 4.33 which represents the drag force resulting from shear and pressure forces acting on the surface of the drone and payload.

$$F_{noise} = \frac{\rho_a}{2} C_d A_f V_a^2 \tag{4.33}$$

where ρ_a is the density of air at room temperature, V_a is the velocity of the air and A_f is the frontal area of the object. The constant C_d is the drag coefficient of the object. I assumed the drone had a drag coefficient similar to a cuboid and the plumb bob was assumed to have a drag coefficient similar to a cylinder. The resulting process noise for the drone and the payload is shown in Equation 4.34. I expected the drone and payload to experience different magnitudes of disturbance from external forces in the indoor experimental setup since the payload was attached underneath the drone and the downwash from the drone added additional wind disturbances onto the payload.

$$\mathbf{W} = diag\left(\begin{bmatrix} F_{d_x\ variance} \\ F_{d_y\ variance} \\ F_{d_z\ variance} \\ F_{p_x\ variance} \\ F_{p_y\ variance} \end{bmatrix}\right) = diag\left(\begin{bmatrix} 4.9 \times 10^{-3} \\ 4.9 \times 10^{-3} \\ 4.9 \times 10^{-3} \\ 1.44 \times 10^{-2} \\ 1.44 \times 10^{-2} \end{bmatrix}\right) N^2 \tag{4.34}$$

I extracted the control forces applied to the drone ($\mathbf{F}_d$) from the drone control system. I modeled the control forces applied to the payload ($\mathbf{F}_p$) as zero at each time step in the EKF since no direct forces were applied to the payload.

Using the input control forces, the drone and payload position measurements, and the payload dynamics model, the EKF ran at a constant time step (10 Hz) which was the capture rate of the VLP-16$^{\text{TM}}$ LiDAR sensor. The resulting error between the EKF estimate and the position measurement for the drone's x, y, and z-position are shown in Figure 4.21.

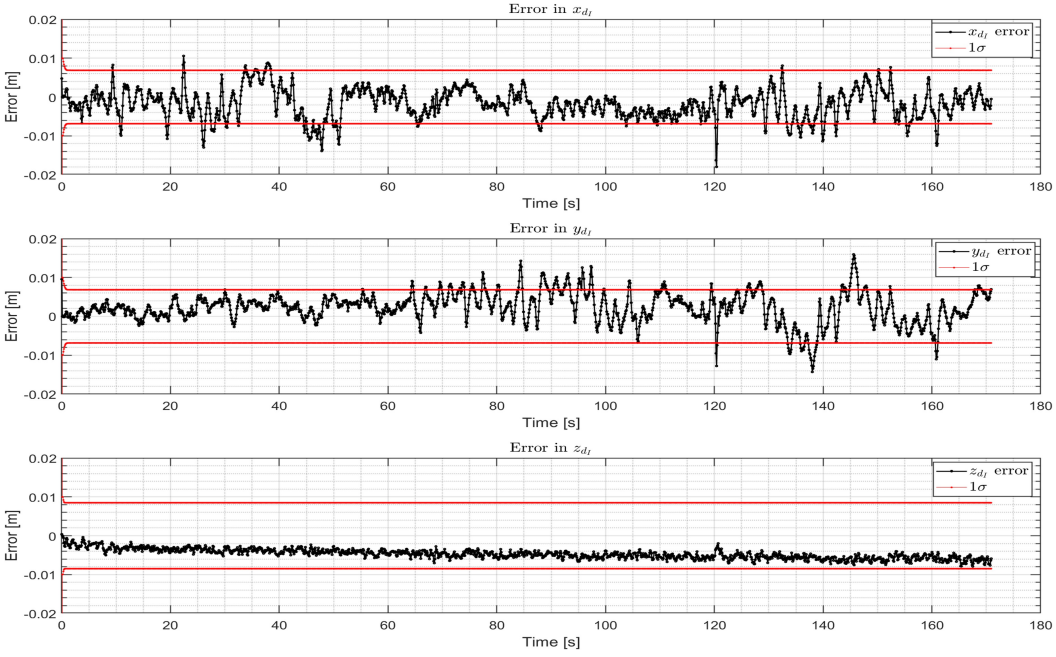


Figure 4.21: Error in EKF estimate of drone's position.

The filter performed well in estimating the position of the drone since the majority of the error is bound between $\pm 1\sigma$ ($\pm 0.008\ m$) in the x, y, and z drone position estimates. The EKF estimates for the x and y position of the payload relative to the drone are shown in Figure 4.22. I calculated the payload's position relative to the drone using Equation 3.31 and Equation 3.32. Figure 4.23 shows the error overlayed onto the positive and negative square roots of the diagonals of the *a posteriori* covariance matrix which represent the expected payload position 1σ error bounds.

The filter performed well in estimating the position of the payload relative to the drone since the majority of the error is contained within the $\pm 1\sigma$ bounds of $\pm 0.02\ m$ and $\pm 0.017\ m$ in the x and y payload position estimates respectively. A higher error in the x-position estimate than the y-position estimate was expected due to the LiDAR's poor resolution in the x-direction.

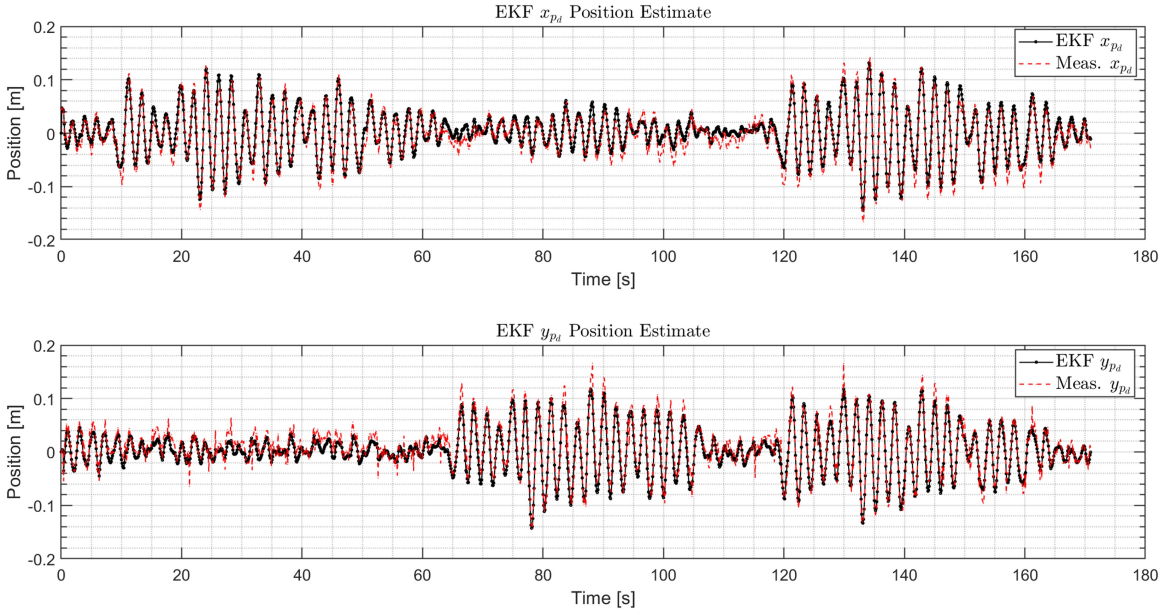


Figure 4.22: EKF estimate of payload's x and y-position in the $\vec{\mathcal{F}}_d$ frame.

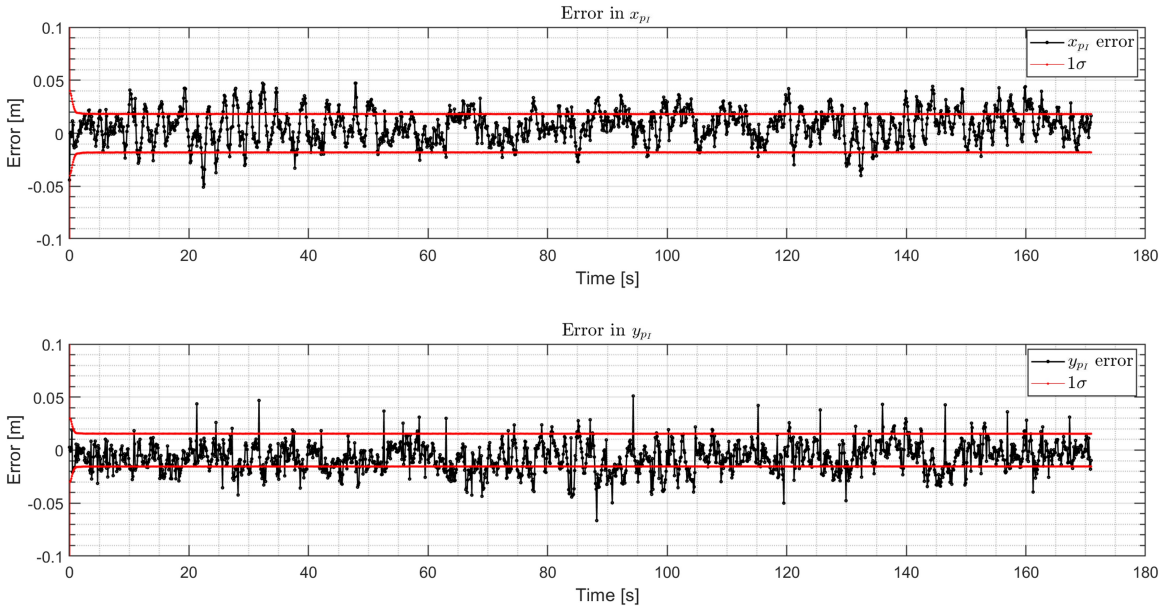


Figure 4.23: Error in EKF estimate of payload's x and y-position.

Finally, I computed the error between the z-position of the payload in the Earth-fixed inertial frame (calculated using Equation 3.20) and the Vicon measurement to verify the filter performance. Figure 4.24 shows the payload's z-position estimate error computed using the payload's x and y estimates from the EKF, and the Vicon truth measurement. The payload's z-position estimate agreed with the Vicon truth measurements, as shown by the small error magnitude ($\sim \pm 6\ mm$), indicating the efficacy of the proposed tracking and estimation system.

Overall, the error in the EKF estimates of the drone's and payload's position did not decay to zero but the majority of the error was contained within the $\pm 1\sigma$ error bounds showing that the filter had converged.

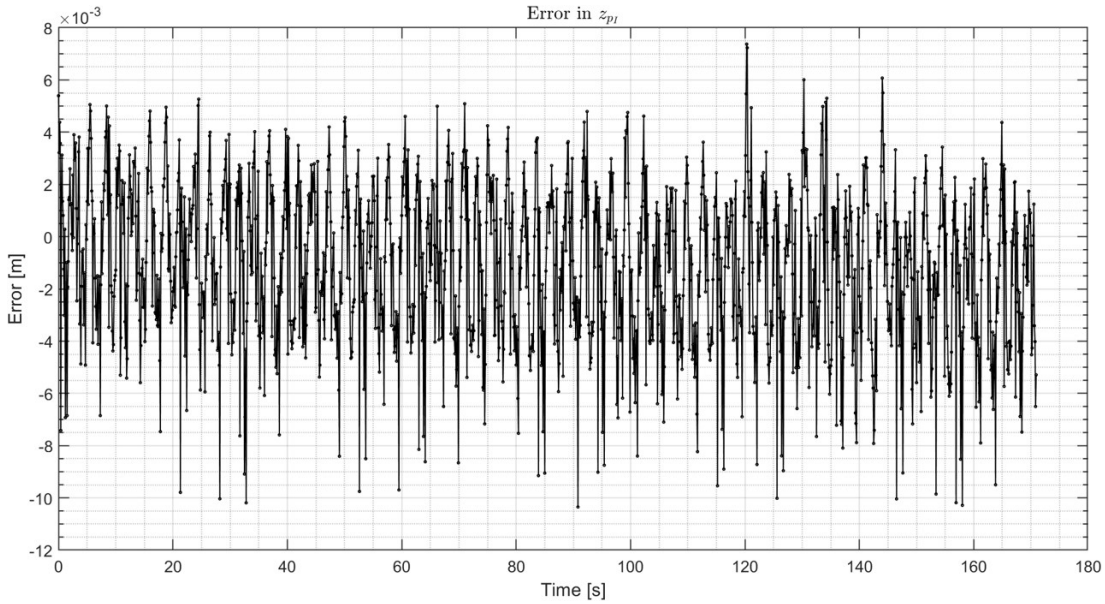


Figure 4.24: EKF estimate of payload's z-position in the $\vec{\mathcal{F}}_I$ frame.

4.4 Chapter Summary

In this chapter, I developed a payload tracking algorithm using a LiDAR sensor and an EKF. First, I used k-means clustering to determine the centroid of the payload using the LiDAR's point cloud. Experimental results showed quantization in the payload's x-centroid estimate due to the LiDAR's poor resolution in its x-axis. Hence, to provide a better estimate of the payload's position relative to the drone, I designed and experimentally tested an EKF. Experimental results showed the feasibility of the tracking algorithm since the EKF's estimated error in the drone's position was less 8 mm (in all three axes), and the EKF's estimated error in the payload's position was less 2 cm (in all three axes).

4.5 Hypothesis H1 Evaluation

Recall hypothesis H1:

> *A LiDAR sensor, in combination with a Kalman filter, can be used as a feedback sensor to track and estimate the position of a drone payload to an accuracy of 3 cm relative to the drone's position.*

The results of my research have found this hypothesis to be **true**. The combination of using an EKF with a LiDAR sensor has resulted in a position estimate error of less than 2 *cm* in the x-direction, 1.7 *cm* in the y-direction, and 0.5 *cm* in the z-direction, meeting the requirements of hypothesis H1. Note that these error values represent the error in the payload's position relative to the drone.

Chapter 5

Controller Design Background

This chapter provides a theoretical background on integer-order and fractional-order calculus and their applicability to PID controllers. First, a comparison of integer-order and fractional-order calculus is presented. Then, the discretization of fractional integrals and derivatives in the time domain, and approximations of fractional integrals and derivatives in the Laplace domain are reviewed from an application standpoint to real-time systems. Algorithms for computing fractional-order integrals and derivatives, in both discrete and continuous forms, are presented and compared.

5.1 Integer-Order PID Control

In any control application, a Proportional-Integral-Derivative (PID) controller is the most widely implemented controller in the industry. More than 95% of the control loops are of PID type of which 90% are integer PID controllers [19]. This is because Integer-Order PID (IOPID) controllers are simple to design and implement with the availability of several auto-tuning methods including real-time auto-tuning [101] and genetic algorithms [102] for auto-tuning PID controllers. However, from a control

point of view, improving the system behaviour is a major concern, specifically when considering systems with highly non-linear behaviour [87].

An IOPID controller aims to maintain the output of the system as close as possible to the targeted output using three tunable parameters shown in Figure 5.1.

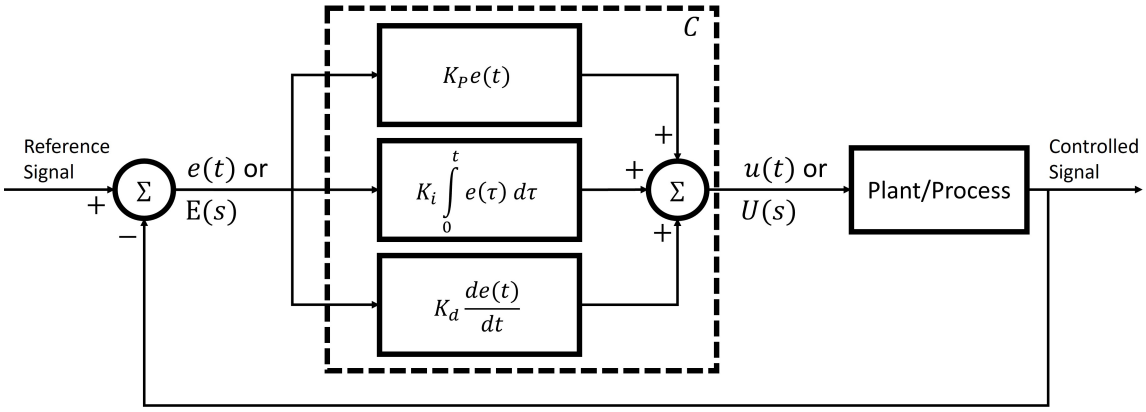


Figure 5.1: Integer-Order PID control block diagram.

The IOPID controller transfer function ($C(s)$) can be written as Equation 5.1 in the frequency domain, while in the time domain, the controller output ($u(t)$) can be written as in Equation 5.2.

$$C(s) = \frac{U(s)}{E(s)} = K_p + K_i s^{-1} + K_d s \tag{5.1}$$

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt} \tag{5.2}$$

- The Proportional gain (K_p) creates a control response that is proportional to the control error.
- The Integral gain (K_i) creates a control response that is related to the accumulated control error based on the past values.

- The Derivative gain (K_d) provides a damping effect on the measurement based on the rate of change of the error.
- $e(t)$ is the error between the reference signal and plant output as a function of time.

This type of controller is referred to as an IOPID controller because the order of integration and differentiation is always real integer values. Since the system is linear, it can be realized by the exponents on the Laplace variable s in Equation 5.1 (-1 for the integrator and 1 for the derivative). Similarly, in the time domain (Equation 5.2), the first integral and first derivative of the error signal are used to determine the control output.

5.2 Fractional-Order PID Control

While integer-order calculus has been used for a long time, fractional calculus has gained attention since the 1900's. Fractional calculus is a generalization of integer calculus that performs non-integer order integrals and derivatives. This means that rather than calculating the third integral or the second derivative per traditional integer calculus, one could determine the one-eighth integral or the one-third derivative of a given function.

Several fractional integral and differentiation definitions have been developed [18], with each definition having its pros and cons. This makes fractional-order controllers harder to develop as compared to an integer-order controller which has a fixed definition. However, with the ever-growing drone industry, there is a need to improve the controller's performance to stabilize and control the non-linear motion of a suspended payload. In general, linear controllers, such as the IOPID controller, do not

perform well when tasked with controlling non-linear dynamics, thus requiring the development of non-linear controllers [103].

5.2.1 Fractional Calculus Background

A Fractional-Order PID controller (FOPID) may improve the performance of a closed-loop feedback system when compared to an IOPID controller by implementing fractional-I and fractional-D actions. This results in two additional tunable parameters (when compared to IOPID controllers) arising from the fractional-order of integration (ξ) and differentiation (ζ), as shown in Figure 5.2.

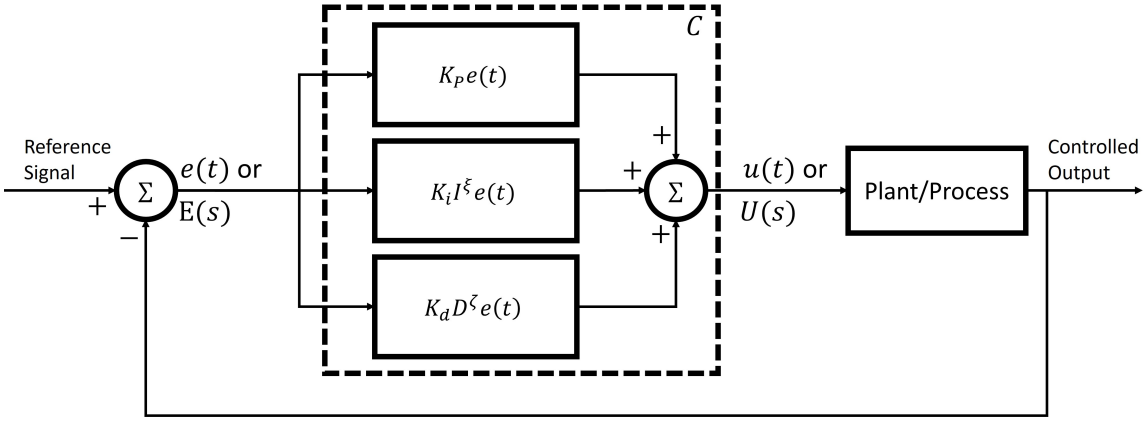


Figure 5.2: Fractional-Order PID control block diagram.

The FOPID controller transfer function ($C(s)$) can be written as Equation 5.3 in the frequency domain, while in the time domain, the controller output ($u(t)$) can be written as in Equation 5.4 [104].

$$C(s) = \frac{U(s)}{E(s)} = K_p + K_i s^{-\xi} + K_d s^{\zeta} \tag{5.3}$$

$$u(t) = K_p e(t) + K_i I^{\xi} e(t) + K_d D^{\zeta} e(t) \tag{5.4}$$

- The Proportional action is identical to the IOPID controller.
- The Integration order term responds with selective memory of the past error values.
- The Differential order term predicts the future error by extrapolating the error using a straight line that is not tangent to the process curve (when $\zeta \neq 1$) but dependent on the order of differentiation.
- I^{ξ} is the fractional integral of order ξ and D^{ζ} is the fractional derivative of order ζ. The process of calculating the fractional integral and derivative depends on the definition of fractional integration and differentiation used.

The fractional integral is not the area under a curve but the area under a projection of the curve. This arises from the definition of fractional integration that implements a scaling property of $(t - \tau)$, as shown in Equation 5.5 and Equation 5.6. This results in a dynamic change in the geometric interpretation of the fractional integral (the projection of the curve experiences a change in size and shape) rather than the simple increase in area under the curve in integer calculus [104].

Compared to integer-order calculus that is performed over an infinitesimal range (*i.e.*, integer-order calculus uses local properties), fractional differentiation is non-local and can be attributed to having memory. Therefore, the behaviour of a process is not only determined by the state at a given time t, but also by all previous states over a time interval τ (τ ranges from the initial time to the current time). In theory, this makes fractional controllers more predictable, allowing the creation of a control algorithm that can deliver precise future control signals to follow the reference signals, thereby enabling them to operate closer to the system constraints which may lead to better system performance.

To compute fractional integrals or derivatives, several time domain definitions have been derived some of which have been presented in [18], including Caputo, Riemann-Liouville (RL), Grunwald-Letnikov, Weyl, Marchaud, Hadamard, and Cossar. Each definition is best applied to certain applications based on the nature of the function undergoing fractional differentiation or integration.

For example, RL definitions are well-suited for modeling viscoelastic deformation and advantageous since they do not require a function to be differentiable. Further, the RL derivative removes the singularity at the origin for certain functions. However, if a function is constant at the origin, the fractional derivative has a singularity at the origin reducing the range of applications for the RL derivative [20]. For the Caputo definitions, initial and boundary conditions can be included in the formulation of the Caputo derivative. However, the Caputo definitions are only valid for differentiable functions [20, 105].

The RL integral and the RL derivative are given in Equation 5.5 and Equation 5.6 respectively.

$$I^{\xi} f(t) = {}_{a}D_{t}^{-\xi} = \frac{1}{\Gamma(\xi)} \int_{a}^{t} (t-\tau)^{\xi-1} f(\tau) d\tau \tag{5.5}$$

$${}_{a}D_{t}^{\zeta} f(t) = \frac{d^n}{dt^n} I^{n-\zeta} f(t) = \frac{1}{\Gamma(n-\zeta)} \frac{d^n}{dt^n} \int_{a}^{t} (t-\tau)^{n-\zeta-1} f(\tau) d\tau \tag{5.6}$$

- The notation ${}_{a}D_{t}^{\zeta}$ represents the differentiation of a function with respect to t with a lower integration bound of a and differentiation order of ζ.
- $f(t)$ is the function for which the integral or derivative will be performed.
- ξ is the fractional integral order and $\xi > 0$ while ζ is the fractional differential order and $n - 1 < \zeta < n$.

- $n-1$ and n represent that the integer numbers that bound the differentiation order (ζ), when ζ is rounded down and rounded up respectively.
- Γ represents the "Gamma Function" and is determined using Equation 5.7.

$$\Gamma(m) = (m-1)! \tag{5.7}$$

Note that the fractional integral notation can also be denoted as a derivative with negative power, as in Equation 5.5, and that RL fractional derivative is calculated by first performing the RL integral $n-\zeta$ times and then differentiating the result n times. To explain this further, consider Figure 5.3.

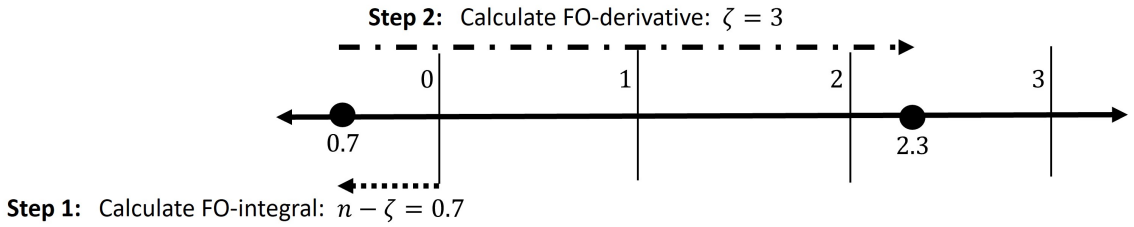


Figure 5.3: Riemann-Liouville derivative schematic.

Suppose I want to calculate the fractional derivative of order 2.3 ($\zeta = 2.3$) for any given function. Then, $n = 3$. Therefore, I would first calculate the RL integral of order 0.7 ($n - \zeta = 3 - 2.3 = 0.7$) and then differentiate with an order of 3 ($n = 3$), to achieve my overall goal of differentiating the function with an order of $\zeta = 2.3$.

The RL definition in Equation 5.5 and Equation 5.6 are given in the time domain, but of importance to control systems engineering are transfer functions in the Laplace domain. The Laplace transform of the RL fractional integral and derivative can be determined using Equation 5.8 and Equation 5.9, respectively.

$$\mathcal{L}\{I^{\xi} f(t)\} = s^{\xi} F(s) \tag{5.8}$$

$$\mathcal{L}\{D^{\zeta} f(t)\} = s^{\zeta} F(s) \tag{5.9}$$

Now consider a fractional-order plant. A fractional-order dynamic system can be modeled as a linear sum of fractional-order differential equations [106, 107], as shown in Equation 5.10, where a_n and b_m are constants, ζ_n and β_m are arbitrary real numbers, and $u(t)$ is the control input.

$$a_n D_t^{\zeta_n} f(t) + a_{n-1} D_t^{\zeta_{n-1}} f(t) + ... + a_0 D_t^{\zeta_0} f(t) = b_m D_t^{\beta_m} u(t) + ... + b_0 D_t^{\beta_0} u(t) \quad (5.10)$$

Applying the Laplace transform to Equation 5.10 results in the generalized continuous transfer function (of a fractional-order system) shown in Equation 5.11.

$$G(s) = \frac{b_m s^{\beta_m} + b_{m-1} s^{\beta_{m-1}} + ... + b_0 s^{\beta_0}}{a_n s^{\zeta_n} + a_{n-1} s^{\zeta_{n-1}} + ... + a_0 s^{\zeta_0}} \quad (5.11)$$

A similar approach can be used to derive a fractional controller transfer function. Hence, in theory, control systems can have both fractional-order plants and fractional-order controllers. But, in most cases, it is common to use integer-order dynamics that are derived in the classical sense (similar to the dynamics derived in Chapter 3) and consider designing only a fractional-order controller.

However, fractional controllers cannot be implemented directly into a digital control system using such definitions. Equation 5.11 shows that a fractional-order continuous transfer function is an irrational transfer function since it results in an infinite number of poles and zeros. While it is a benefit of fractional-order systems to have infinite memory when compared to integer-order systems, it limits their implementation to real-life problems since infinite memory is unfeasible. Hence, it is only possible to generate an exact transfer function for an integer-order system using conventional methods, and some form of approximation is required to implement fractional controllers into physical systems. Then, the controller parameters (K_p, K_i, K_d, ξ, and ζ) can be tuned to optimize the trade-off between percent overshoot, steady-state error, and settling time.

5.2.2 Approximations for Fractional Order Operators

In general, numerical methods have been used to generate both continuous and discrete time approximations of fractional-order transfer functions to obtain closed-loop solutions for real-time applications.

For discrete models, the Grunwald-Letnikov definition is a well-established definition in fractional calculus [108]. The discretized version of the Grunwald-Letnikov derivative and integral is defined in Equation 5.12 and Equation 5.13, for a constant time step T.

$$D_T^{\zeta} f(nT) = T^{-\zeta} \sum_{k=0}^{\infty} (-1)^k \binom{\zeta}{k} f((n-k)T) \tag{5.12}$$

$$I_T^{\xi} f(nT) = T^{\xi} \sum_{k=0}^{\infty} (-1)^k \binom{-\xi}{k} f((n-k)T) \tag{5.13}$$

Both equations show that even for a discretized fractional-order integral and derivative model, full memory is needed (seen by the summation from 0 to ∞). Therefore, to approximate the Grunwald-Letnikov discretized definitions using finite memory, power series expansion and continued fraction expansion are needed. Further, either the backward difference, forward difference, or trapezoidal rule is required.

Discrete models with fixed memory have been tested and proven to show reliable performance in [108]. However, they only performed well for smooth periodic functions. Furthermore, as shown by the authors in [109,110], discretized fractional-order definitions with fixed memory do not perform well when compared to continuous approximations. The step size has a large impact on the performance of discrete models as does the memory length. Since the function value needs to be known and stored for the calculation of integrals and derivatives at a future time, a larger memory length is preferred. This means that more storage and computational power are required which is not ideal for real-time applications.

On the other hand, continuous models for approximating a fractional-order transfer function are widely used approaches for application purposes [109, 111, 112]. Continuous models either use continuous fractional expansion and/or curve fitting techniques to generate an integer-order rational (finite-term) approximation of an irrational (infinite-term) transfer function.

Among many algorithms, Continuous Fraction Expansion (CFE), Matsuda's method, and Oustaloup's Filter are commonly used approximation methods [111, 113]. These algorithms aim to approximate the magnitude and phase response of the irrational transfer function $G(s) = s^{\zeta}$ or $G(s) = s^{\xi}$ within a given frequency range, using a finite number of poles and zeros (referred to as the order of approximation).

For this thesis, I only considered continuous models. I evaluated CFE, Matsuda's method, and Oustaloup's Filter in simulation to compare the performance of each method.

5.2.2.1 Continuous Fraction Expansion Approximation of s^{ζ} or s^{ξ}

For a given transfer function $G(s) = s^{\xi}$ (fractional integrator) or $G(s) = s^{\zeta}$ (fractional derivative), the rational transfer function ($H(s)$) can be determined by performing a CFE of Equation 5.14 [111]. CFE should be truncated to obtain an n-order finite calculation where $x = s - 1$ can be substituted into Equation 5.15 to obtain the CFE of s^{ζ}.

$$CFE\{s^{\zeta}\} = CFE\{(1+x)^{\zeta}\} \tag{5.14}$$

$$(1+x)^{\zeta} = 1 + \cfrac{\zeta x}{1 + \cfrac{(1-\zeta)x}{2 + \cfrac{(1+\zeta)x}{3 + \cfrac{(2-\zeta)x}{2+\ldots}}}} \tag{5.15}$$

Overall, a CFE approximation can be simple to calculate, however, one drawback of CFE is that the frequency range of the approximation is not configurable. A larger

frequency range can be achieved using higher order approximations, but the overall increase in the frequency bandwidth by using higher order approximations is still limited [111].

5.2.2.2 Matsuda's Method

Matsuda's method [111] is a combination of curve fitting and CFE. This method approximates the irrational fractional-order transfer function ($G(s)$) using CFE and fits a set of integer-order approximations of $G(s)$ using a set of logarithmically sampled frequencies.

For a given frequency range $[\omega_l, \omega_h]$, the sampled frequencies (ω_k) used to curve-fit the approximation are calculated using Equation 5.16 and the corresponding gain values are determined using Equation 5.17.

$$\omega_k = \omega_l \left(\frac{\omega_h}{\omega_l}\right)^{\frac{k-1}{n-1}} \tag{5.16}$$

$$gain = 20 \cdot \log_{10}\left(|\omega_k|^{\zeta}\right) \tag{5.17}$$

where n is the order of the approximation ($n > 0$). For an n^{th}-order approximation, $2n+1$ frequencies are selected for curve fitting, hence, $k = 0, 1, 2, ..., 2n$. Then, using the gain value at each of the sampled frequencies (ω_k), the following interpolation process is solved recursively to fit a curve to the sampled frequencies.

$$\begin{aligned} d_0(\omega_k) &= |gain(j\omega_k)| \\ d_1(\omega_k) &= \frac{\omega_k - \omega_0}{d_0(\omega_k) - d_0(\omega_0)} \\ &\vdots \\ d_n(\omega_k) &= \frac{\omega_k - \omega_{n-1}}{d_{n-1}(\omega_k) - d_{n-1}(\omega_{n-1})} \end{aligned} \tag{5.18}$$

Then, using the curve-fitted model and CFE, the rational transfer function ($H(s)$) can be determined using Equation 5.19.

$$H(s) = a_0 + \frac{s - \omega_0}{a_1 + \frac{s-\omega_1}{a_2 + \frac{s-\omega_2}{a_3 + \ldots}}} \tag{5.19}$$

where $a_0 = |gain(j\omega_0)|$ for $k = 0$. For $k = 1, 2, ..., 2n$:

$$a_k = d_k(\omega_k) = \frac{\omega_k - \omega_{k-1}}{d_{k-1}(\omega_k) - d_{k-1}(\omega_{k-1})} \tag{5.20}$$

A low-order Matsuda approximation can result in poor frequency response due to the presence of unwanted ripples in the magnitude and phase response. A higher-order Matsuda approximation results in more frequent sampling for interpolation which results in a better phase response approximation.

5.2.2.3 Oustaloup's Recursive Filter

Among other filter approximations for fractional-order transfer functions presented in the literature, the Oustaloup recursive filter has shown good fitting for several applications [114, 115]. The Oustaloup filter approximates a transfer function of the form $G(s) = s^{\xi}$ (fractional integral) or $G(s) = s^{\zeta}$ (fractional derivative) such that the frequency response of the approximated rational function $H(s)$ fits the frequency response of $G(s)$.

The Oustaloup filter approximates the transfer function of the fractional derivative (s^{ζ}) using an integer-order transfer function, as per Equation 5.21. The approximation is valid in a frequency range $[\omega_l, \omega_h]$. The number of poles and zeros (N) is chosen beforehand. A lower value of N results in a simpler approximation, but may result in ripples in the frequency response of the approximation [109, 116, 117].

$$H(s) = K_O \prod_{k=-N}^{N} \frac{s + \omega_k^{'}}{s + \omega_k} \tag{5.21}$$

where the poles and zeros of the transfer function can be determined using Equation 5.22 and Equation 5.23, respectively. The gain K_O can be evaluated using Equation 5.24.

$$\omega_k^{'} = \omega_l \left(\frac{\omega_h}{\omega_l}\right)^{\frac{k+N+0.5(1-\zeta)}{2N+1}} \tag{5.22}$$

$$\omega_k = \omega_l \left(\frac{\omega_h}{\omega_l}\right)^{\frac{k+N+0.5(1+\zeta)}{2N+1}} \tag{5.23}$$

$$K_O = \omega_h^{\zeta} \tag{5.24}$$

To determine an approximation of a fractional-order integral s^{ξ}, Equation 5.21 needs to be inverted. The Oustaloup filter approximation is valid for $0 < \zeta < 1$ and $0 < \xi < 1$. To determine the transfer function for integral and derivative orders greater than 1, integer-order integrals and derivatives can be performed, leaving only the fractional value for the Oustaloup filter, as shown in Equation 5.25.

$$s^{\zeta} = s^{floor(\zeta)}\ s^{(\zeta - floor(\zeta))} \tag{5.25}$$

The Oustaloup filter is simple to implement. The only drawback to this method is the limitation of the number of poles and zeros to odd numbers (the product from $-N$ to N in Equation 5.21 results in $2N + 1$ poles and zeros) [111].

5.2.3 Frequency Response Comparison of CFE, Matsuda, and Oustaloup Approximations

I compared the three approximation methods discussed above by comparing the magnitude and phase response of each method to determine the best-performing approximation. I simulated each method to approximate the frequency response of the irrational transfer function $G(s) = s^{0.5}$ within the frequency range of $[0.001,\ 1000]\ rad/s$.

The better approximation is one which can accurately match the magnitude and phase response to the actual response of $G(s)$. Moreover, to make a fair comparison, I tested all three methods using varying orders of approximation. A lower-order approximation is better since it contains fewer terms to compute, which is preferred for real-time embedded computer systems such as the flight controller onboard the Quanser QDrone (given its limited computational capability).

Figure 5.4 and Figure 5.5 show the magnitude and phase response of the transfer function $G(s)$ as approximated by CFE, Matsuda's method, and Oustaloup's method using a first, third, and seventh-order approximation.

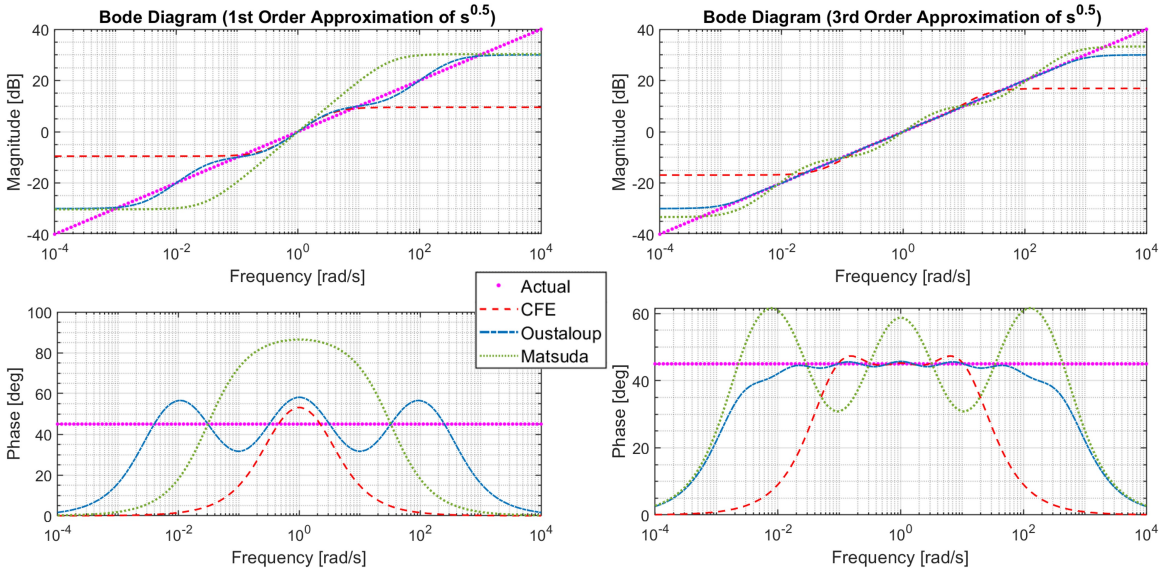


Figure 5.4: Magnitude and phase response comparison of $s^{0.5}$ using 1st and 3rd order approximations.

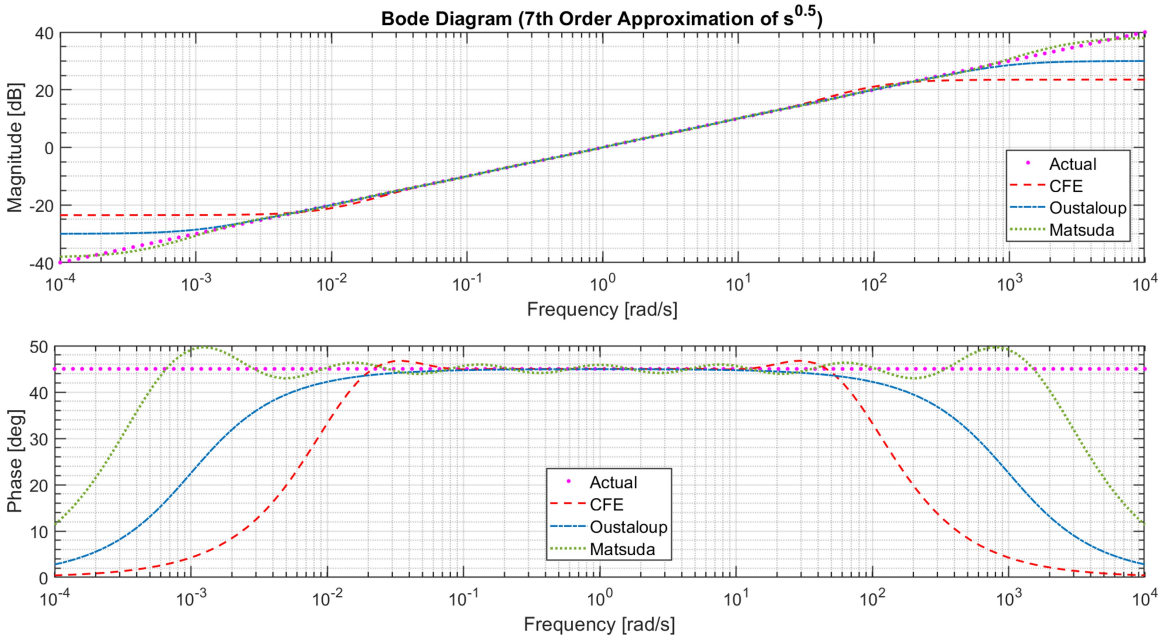


Figure 5.5: Magnitude and phase response comparison of $s^{0.5}$ using a 7th order approximation.

The first-order approximations from all three methods do not perform well at estimating the fractional order derivative of order 0.5. Increasing the approximation order to three shows that the Oustaloup filter provides the most accurate response of all three methods. The CFE approximation suffers from a very small bandwidth while the Matsuda approximation experiences large magnitude ripples in the phase response. Further increasing the approximation order to seven shows that the Oustaloup filter still provides the most accurate approximation of all three methods with no ripples.

The Oustaloup filter does suffer from a decaying phase response at the edges of the upper and lower frequency range ($[0.001,\ 1000]\ rad/s$ for this example), but this can be overcome by specifying a larger frequency range when using the filter. Both the Matsuda and CFE approximations improved with increasing order of approximation, however, even with a seventh-order approximation both suffer from a poorer phase

response when compared to the Oustaloup filter. Furthermore, both Figure 5.4 and Figure 5.5 show that the CFE method suffers from limited bandwidth, while the Matsuda approximation suffers from ripples in the phase response.

Increasing the order of approximation will increase the bandwidth of the CFE approximation and result in a better fitting phase response for the Matsuda approximation. However, this increases the complexity of the transfer function which may result in a lag in real-time systems, such as a real-time drone feedback system running on an embedded flight computer. Therefore, I determined the Oustaloup filter to be the best approximation method and thereby used it to create the FOPID controller for the drone-based slung payload closed-loop control software discussed in Chapter 6.

5.3 Chapter Summary

This chapter provided an overview of integer-order control and fractional-order control and their application to feedback PID controllers. Fractional calculus is a generalization of integer-order calculus where the integrals and derivatives can be determined for any real number. Further, integer-order calculus is local while fractional calculus is associated with having memory. Applying this to PID controllers, the IOPID controller uses three tunable parameters while the FOPID controller uses five tunable parameters, which provides more flexibility. However, having infinite memory prevents the implementation of FOPID controllers in real-time systems using direct definitions of fractional integrals and derivatives. Therefore, I evaluated three continuous methods for approximating the frequency response of the fractional integral and derivative using a finite number of zeros and poles to limit the memory of the fractional controller. The Oustaloup recursive filter performed better than the other approximation methods (CFE and Matsuda's approximation), therefore, I used it to create the feedback FOPID controller for the drone-payload in the following chapter.

Chapter 6

Closed-Loop Control Design, Simulation, Testing, and Discussion

This chapter provides an overview of the closed-loop control software I implemented on the Quanser QDrone to compare the performance of the IOPID and FOPID controllers. This chapter develops and presents a simulation model, which includes the closed-loop drone control software and the drone-based cable-suspended dynamics. This simulation model and Particle Swarm Optimization were used to tune the position and attitude controller gains for both the IOPID and FOPID controllers. Finally, in this chapter, I present the closed-loop drone control software, with the tuned IOPID and FOPID gains, describing how I experimentally tested them on the QDrone. Lastly, I present and discuss the simulation and experimental results.

6.1 Drone Closed-Loop Control

As shown in Chapter 3, the sway of a cable-suspended payload creates a non-linear unmodeled disturbance on the drone. A drone control system that could suppress such disturbances would result in a smoother path for the suspended GPR, which may result in better GPR data. Therefore, I designed a closed-loop feedback control system that controls the drone's position to evaluate how well an integer-order controller and a fractional-order controller could reject the non-linear disturbance.

Further, to evaluate the performance of each candidate controller, I used the overshoot, steady-state error, and settling time since these time response characteristics generally demonstrate the quality of a tracking controller. However, from an operational perspective, the settling time was the most important since a faster settling time would imply that the slung GPR motion would dampen faster, hence, for a given survey area, the GPR data could be collected over a shorter period of time.

6.1.1 Closed-Loop Hardware

The hardware I used to design the closed-loop controller included a Quanser QDrone, a Vicon motion capture system, and a Wi-Fi module. Figure 6.1 shows a schematic of the hardware and how the systems interacted with each other. The Vicon motion capture system measured the position of the QDrone and transferred the data to the ground station computer. The ground station connected to a Wi-Fi router that communicated between the drone and the ground station.

The flight software on the QDrone flight controller and the ground station software ran at a frequency of 1000 Hz, while the Vicon measurements were recorded at 100 Hz. Note that the control and measurement frequencies were not equal, but separating these frequencies by an order of magnitude ensured there was no aliasing between the control and measurements. The Wi-Fi communication, Vicon hardware,

and computational time between successive time steps resulted in a data transportation delay of approximately 0.01 s between the Vicon system and the flight controller.

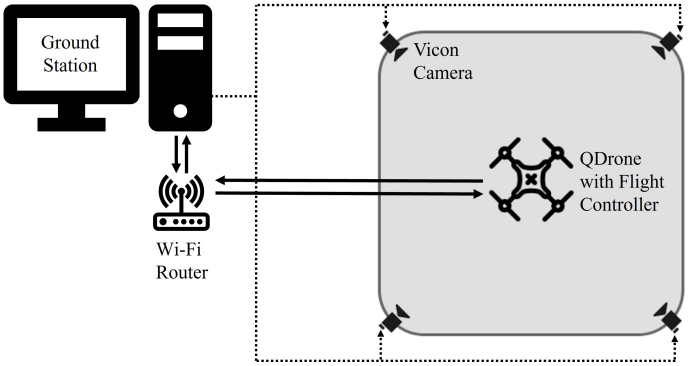


Figure 6.1: Closed-loop hardware setup.

6.1.2 Closed-Loop Software

The overall closed-loop control schematic I designed is shown in Figure 6.2. The system consisted of a ground station and a flight control software. The ground station ran on a local computer while the flight control software ran onboard the QDrone.

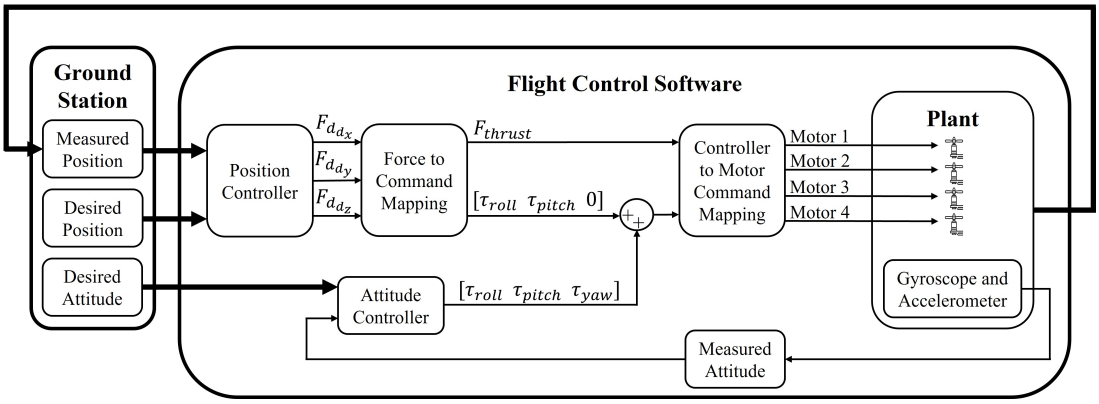


Figure 6.2: Closed-loop control software block diagram designed for QDrone.

I set the desired position and attitude of the drone within the ground station. The ground station also measured the position of the drone using the Vicon system and transferred three-axis position data to the drone flight control software.

The flight control software I designed for the Quanser QDrone had three key components: the position controller, the attitude controller, and the controller command to motor command mapping. The position and attitude controllers were responsible for maneuvering the drone along a desired trajectory. The controller command to motor command mapping was responsible for transforming the overall forces and torques commanded by the controllers to motor-specific commands.

6.1.2.1 Position Controller

A schematic of the position controller is shown in Figure 6.3. The position controller took the desired and measured position as inputs and output a three-axis force vector.

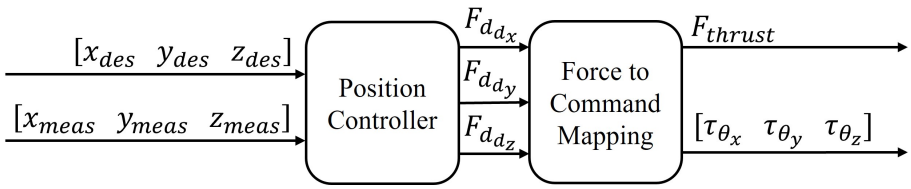


Figure 6.3: Position controller schematic.

The position controller used the desired and measured position of the drone to calculate the position control error ($e_x,\ e_y,\ e_z$), per in Equation 6.1 through Equation 6.3. The subscript *des* represents the desired position and the subscript *meas* represents the measured position.

$$e_x(t) = x_{des} - x_{meas} \tag{6.1}$$

$$e_y(t) = y_{des} - y_{meas} \tag{6.2}$$

$$e_z(t) = z_{des} - z_{meas} \tag{6.3}$$

Then, the position error in each axis was fed into an IOPID or a FOPID controller to generate a control command separate in each axis (*i.e.*, I used three position

controllers, one for each axis). Equation 6.4, Equation 6.5, and Equation 6.6 show the resulting IOPID controller transfer function in the time domain for position control in the x-axis, y-axis, and z-axis respectively. The superscript io on the controller gains implies that the gains are for an integer-order PID controller.

$$C_x^{io}(t) = K_{p_x}^{io} e_x(t) + K_{i_x}^{io} \int_0^t e_x(\tau) d\tau + K_{d_x}^{io} \frac{de_x(t)}{dt} \tag{6.4}$$

$$C_y^{io}(t) = K_{p_y}^{io} e_y(t) + K_{i_y}^{io} \int_0^t e_y(\tau) d\tau + K_{d_y}^{io} \frac{de_y(t)}{dt} \tag{6.5}$$

$$C_z^{io}(t) = K_{p_z}^{io} e_z(t) + K_{i_z}^{io} \int_0^t e_z(\tau) d\tau + K_{d_z}^{io} \frac{de_z(t)}{dt} \tag{6.6}$$

Since the IOPID controller used integer-order derivatives, the derivative action in the IOPID transfer functions represented the velocity error of the drone. However, calculating the derivative of noisy measurement error data ($e_x,\ e_y,\ e_z$) is challenging since it results in an inaccurate and noisy velocity error estimate. Therefore, when designing the IOPID controller, it was beneficial to first estimate the velocity of the drone and then use the velocity estimate to calculate the velocity error.

I estimated the velocity of the drone using its noisy position measurement by calculating a filtered derivative using a low-pass filter. This resulted in a smoother and more accurate estimate of the drone's velocity. I used the transfer function, shown in Equation 6.7, to calculate the velocity of the drone in all three axes, where T_v is the sampling period.

$$V(s) = \frac{s}{T_v s + 1} \tag{6.7}$$

Using the measured velocity estimate (v_{meas}) and a desired velocity (v_{des}) of zero, I calculated the velocity control error ($\dot{e}_x,\ \dot{e}_y,\ \dot{e}_z$) using Equation 6.8 through Equation

6.10. I set the desired velocity in each axis to zero so that the controller held the drone stationary unless a position change was requested.

$$\dot{e}_x(t) = v_{x_{des}} - v_{x_{meas}} \tag{6.8}$$

$$\dot{e}_y(t) = v_{y_{des}} - v_{y_{meas}} \tag{6.9}$$

$$\dot{e}_z(t) = v_{z_{des}} - v_{z_{meas}} \tag{6.10}$$

Then, the controller transfer function from Equation 6.4 to Equation 6.6 can be re-written as:

$$C_x^{io}(t) = K_{p_x}^{io} e_x(t) + K_{i_x}^{io} \int_0^t e_x(\tau) d\tau + K_{d_x}^{io} \dot{e}_x(t) \tag{6.11}$$

$$C_y^{io}(t) = K_{p_y}^{io} e_y(t) + K_{i_y}^{io} \int_0^t e_y(\tau) d\tau + K_{d_y}^{io} \dot{e}_y(t) \tag{6.12}$$

$$C_z^{io}(t) = K_{p_z}^{io} e_z(t) + K_{i_z}^{io} \int_0^t e_z(\tau) d\tau + K_{d_z}^{io} \dot{e}_z(t) \tag{6.13}$$

The FOPID controller did not experience the issue of calculating the derivative of the noisy position error since it performed non-integer derivatives, which accounted for the noisy position measurement. The FOPID controller transfer function in the Laplace form for position control in the x-axis, y-axis, and z-axis is shown in Equation 6.14, Equation 6.15, and Equation 6.16 respectively.

$$C_x^{fo}(s) = K_{p_x}^{fo} + K_{i_x}^{fo} s^{-\xi_x} + K_{d_x}^{fo} s^{\zeta_x} \tag{6.14}$$

$$C_y^{fo}(s) = K_{p_y}^{fo} + K_{i_y}^{fo} s^{-\xi_y} + K_{d_y}^{fo} s^{\zeta_y} \tag{6.15}$$

$$C_z^{fo}(s) = K_{p_z}^{fo} + K_{i_z}^{fo} s^{-\xi_z} + K_{d_z}^{fo} s^{\zeta_z} \tag{6.16}$$

The superscript fo on the controller gains implies that the gains are for a fractional-order PID controller. I computed the fractional-order integral and derivative using Oustaloup's recursive filter approximation, per Equation 5.21.

The outputs of the position controller were three controller commanded forces: $F_{d_{d_x}}$, $F_{d_{d_y}}$, and $F_{d_{d_z}}$, which represent the forces required to maintain the desired position and velocity. The force to command mapping block converted these forces to thrust force (F_{thrust}), roll torque (τ_{θ_x}), pitch torque (τ_{θ_y}), and yaw torque (τ_{θ_y}), using Equation 6.17 through Equation 6.20.

$$F_{thrust} = F_{d_{d_z}} \tag{6.17}$$

$$\tau_{\theta_x} = -F_{d_{d_y}} \left(\frac{L_{roll}}{2} \right) \tag{6.18}$$

$$\tau_{\theta_y} = F_{d_{d_x}} \left(\frac{L_{pitch}}{2} \right) \tag{6.19}$$

$$\tau_{\theta_z} = 0 \tag{6.20}$$

where L_{roll} is the center-to-center distance between the motors along the y-axis of the drone and L_{pitch} is the center-to-center distance between the motors along the x-axis of the drone.

6.1.2.2 Attitude Controller

A schematic of the attitude controller is shown in Figure 6.4. The attitude controller took as input the desired and measured attitude ($\theta_x,\ \theta_y,\ \theta_z$) and attitude rates ($\dot{\theta}_x,\ \dot{\theta}_y,\ \dot{\theta}_z$) to output a three-axis torque vector. I set the desired attitude and

attitude rates for all tests and simulations to zero ($\theta_{x_{des}} = \theta_{y_{des}} = \theta_{z_{des}} = \dot{\theta}_{x_{des}} = \dot{\theta}_{y_{des}} = \dot{\theta}_{z_{des}} = 0$). This ensured that the drone always maintained a stable attitude unless a position change was requested. Both the attitude and attitude rates, about all three axes, were measured using a gyroscope and an accelerometer onboard the drone.

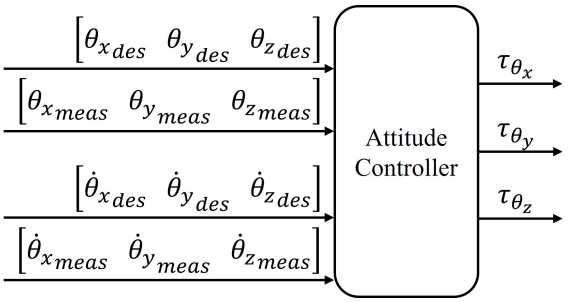


Figure 6.4: Attitude controller schematic.

Similar to the position controller, the attitude control error ($e_{\theta_x},\ e_{\theta_y},\ e_{\theta_z}$) was determined using the desired and measured attitude of the drone, per Equation 6.21 through Equation 6.23. The angles θ_x, θ_y, and θ_z represent the roll, pitch, and yaw angles respectively.

$$e_{\theta_x}(t) = \theta_{x_{des}} - \theta_{x_{meas}} \tag{6.21}$$

$$e_{\theta_y}(t) = \theta_{y_{des}} - \theta_{y_{meas}} \tag{6.22}$$

$$e_{\theta_z}(t) = \theta_{z_{des}} - \theta_{z_{meas}} \tag{6.23}$$

The attitude rate control error ($\dot{e}_{\theta_x},\ \dot{e}_{\theta_y},\ \dot{e}_{\theta_z}$) was determined using the desired and measured attitude rates, per Equation 6.24 through Equation 6.26.

$$\dot{e}_{\theta_x}(t) = \dot{\theta}_{x_{des}} - \dot{\theta}_{x_{meas}} \tag{6.24}$$

$$\dot{e}_{\theta_y}(t) = \dot{\theta}_{y_{des}} - \dot{\theta}_{y_{meas}} \tag{6.25}$$

$$\dot{e}_{\theta_z}(t) = \dot{\theta}_{z_{des}} - \dot{\theta}_{z_{meas}} \tag{6.26}$$

I used an integer-order PD (IOPD) controller to control the attitude of the drone when using an IOPID position controller. Equation 6.27, Equation 6.28, and Equation 6.29 show the resulting IOPD controller transfer function in the time domain for attitude control about the x-axis (roll), y-axis (pitch), and z-axis (yaw), respectively.

$$C^{io}_{\theta_x}(t) = K^{io}_{p_{\theta_x}} e_{\theta_x}(t) + K^{io}_{d_{\theta_x}} \dot{e}_{\theta_x}(t) \tag{6.27}$$

$$C^{io}_{\theta_y}(t) = K^{io}_{p_{\theta_y}} e_{\theta_y}(t) + K^{io}_{d_{\theta_y}} \dot{e}_{\theta_y}(t) \tag{6.28}$$

$$C^{io}_{\theta_z}(t) = K^{io}_{p_{\theta_z}} e_{\theta_z}(t) + K^{io}_{d_{\theta_z}} \dot{e}_{\theta_z}(t) \tag{6.29}$$

I used an IOPD attitude controller (an IOPID without an integral term) since the integrator action in an IOPID controller resulted in high frequency oscillations in the drone's attitude. The oscillations resulted from a combination of sensitive attitude measurements from the onboard sensors and a command that forces the drone to always maintain zero attitude about all three axes. Therefore, adding the integrator action to the IOPD attitude controller resulted in chattering about the zero roll, pitch, and yaw angles.

For similar reasons, I implemented a fractional-order PD (FOPD) controller to control the attitude of the drone when using a FOPID position controller. Equation 6.14, Equation 6.15, and Equation 6.16 show the resulting FOPD controller transfer function in the frequency domain for attitude control in the x-axis (roll), y-axis (pitch), and z-axis (yaw), respectively.

$$C^{fo}_{\theta_x}(s) = K^{fo}_{p_{\theta_x}} + K^{fo}_{d_{\theta_x}} s^{\zeta_{\theta_x}} \tag{6.30}$$

$$C^{fo}_{\theta_y}(s) = K^{fo}_{p_{\theta_y}} + K^{fo}_{d_{\theta_y}} s^{\zeta_{\theta_y}} \tag{6.31}$$

$$C^{fo}_{\theta_z}(s) = K^{fo}_{p_{\theta_z}} + K^{fo}_{d_{\theta_z}} s^{\zeta_{\theta_z}} \tag{6.32}$$

The outputs of the attitude controller were three controller commanded torques: τ_{θ_x}, τ_{θ_y}, and τ_{θ_z}, which represent the torques required in order to maintain the desired

attitude and attitude rates. These torques were added to the torques from the position controller (Equation 6.18 to Equation 6.20) to obtain the total torque required to maintain the desired position and attitude, as shown in Figure 6.2.

6.1.2.3 Controller To Motor Command Mapping

In any drone controller, the total commanded thrust and torques from its position and attitude controller are forces and torques that, in theory, need to be applied to the drone's center of mass. However, since a quadrotor (such as the QDrone) applies forces and torques through each of the four motors, the forces and torques required from each motor need to be computed. Then, these forces and torques need to be converted to motor speed for each motor. This subsection describes the mapping I used for the experiments in this thesis.

After several experimental tests, Quanser [118] developed two empirical constants for the QDrone: $K_t = 0.0487\ Nm$ and $K_f = 5.11\ N$. When a propeller rotates, it generates a torque about the axis of the motor shaft. The constant K_t describes the value of the torque generated by a single propeller when the QDrone is hovering.

The constant K_f describes the maximum thrust one pair of motor and propeller combination on the QDrone can generate. Using the area of the propeller, the motor speed constant, and the nominal full voltage of the QDrone's Lithium-Potasium battery ($12.6\ V$), the motor commands can be calculated as a percentage of the maximum motor speed. This means that if all four motors were running at full throttle, the maximum thrust that could be generated by the QDrone would be $4K_f$ and the motor command would be 1 (or 100%). Using these constants, the controller-commanded thrust force and torques can be converted to desired motor speed. The following describes the process to convert the force and torque commands to motor-specific commands for the QDrone.

Figure 6.5 shows the configuration of the Quanser QDrone which uses a cross-configuration. A cross-configuration quadrotor has counter-rotating propellers on either side of the roll and pitch axes. For example, propellers #2 and #3 rotate counterclockwise while propellers #1 and #4 rotate clockwise.

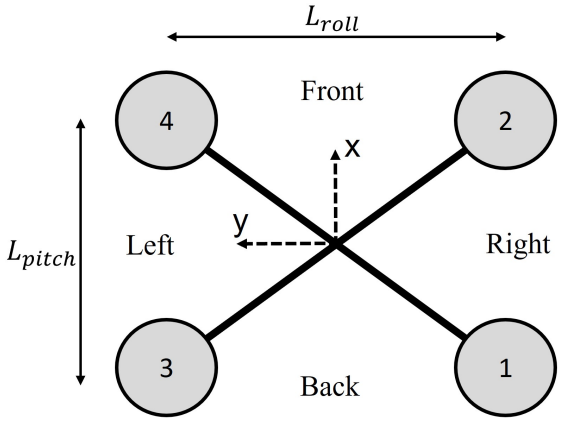


Figure 6.5: Quanser QDrone physical layout.

Using K_f and K_t, and the motor center-to-center distances (L_{roll} and L_{pitch}), the following relations describe the thrust forces and torques generated by a given motor on the quadcopter.

1. To generate a translation motion along the z-axis (vertical motion), the thrust force generated by all motors should be the same. Hence, Equation 6.33 describes the thrust force required from each motor.

$$F_{thrust} = K_f[u_1,\ u_2,\ u_3,\ u_4]^T \tag{6.33}$$

   where u_1, u_2, u_3, and u_4 are the motor commands as a percentage of the maximum motor speed.

2. To generate a positive roll torque (translating towards the negative y-axis), motor 3 and motor 4 would have to rotate faster than motor 1 and motor 2.

This would result in a greater thrust force on the left side than the right side of the drone, thereby resulting in a positive roll torque. Equation 6.34 relates the net roll torque to the motor commands.

$$\tau_{\theta_x} = K_f \left(\frac{L_{roll}}{2} \right) [-1,\ -1,\ 1,\ 1]\, [u_1,\ u_2,\ u_3,\ u_4]^T \tag{6.34}$$

3. To generate a positive pitch torque (translating towards the positive x-axis), motor 1 and motor 3 would have to rotate faster than motor 2 and motor 4. This would result in a greater thrust force on the back side than the front side of the drone, thereby resulting in a positive pitch torque. Equation 6.35 relates the net pitch torque to the motor commands.

$$\tau_{\theta_y} = K_f \left(\frac{L_{pitch}}{2} \right) [1,\ -1,\ 1,\ -1]\, [u_1,\ u_2,\ u_3,\ u_4]^T \tag{6.35}$$

4. To generate a positive yaw torque (counterclockwise rotation about the z-axis), motor 1 and motor 4 would have to rotate faster than motor 2 and motor 3. This would result in a constant thrust force across each diagonal of the drone but a greater yaw moment from motor 1 and motor 4, when compared to motor 2 and motor 3, thereby resulting in a positive yaw torque. Equation 6.36 relates the net yaw torque to the motor commands.

$$\tau_{\theta_z} = K_t\, [1,\ -1,\ -1,\ 1]\, [u_1,\ u_2,\ u_3,\ u_4]^T \tag{6.36}$$

Equation 6.33 to Equation 6.36 can be condensed using matrix notation, as in Equation 6.37 and Equation 6.38 [118].

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = M^{-1} \begin{bmatrix} F_{thrust} \\ \tau_{\theta_x} \\ \tau_{\theta_y} \\ \tau_{\theta_z} \end{bmatrix} \tag{6.37}$$

$$\mathbf{M} = \begin{bmatrix} K_f & K_f & K_f & K_f \\ -K_f\left(\frac{L_{roll}}{2}\right) & -K_f\left(\frac{L_{roll}}{2}\right) & K_f\left(\frac{L_{roll}}{2}\right) & K_f\left(\frac{L_{roll}}{2}\right) \\ K_f\left(\frac{L_{pitch}}{2}\right) & -K_f\left(\frac{L_{pitch}}{2}\right) & K_f\left(\frac{L_{pitch}}{2}\right) & -K_f\left(\frac{L_{pitch}}{2}\right) \\ K_t & -K_t & -K_t & K_t \end{bmatrix} \tag{6.38}$$

where M is the matrix that I used to map the controller-commanded forces and torques to motor commands.

6.2 Controller Tuning and Simulation

In this section, I develop a simulation model with both drone attitude dynamics and drone-based cable-suspended payload dynamics. I used this model to tune the position and attitude controllers from Section 6.1 using Particle Swarm Optimization.

6.2.1 Simulation Model

I used the drone-based cable-suspended payload model, developed in Section 3.2, to tune both the integer-order controllers and fractional-order controllers for a drone carrying a payload. This ensured that the controller parameters for the IOPID and FOPID controllers were tuned to the dynamics that were expected when performing a flight test using the QDrone with a cable-suspended payload.

Recall that I had modeled the drone as a point mass without any attitude dynamics when I derived the drone-based cable-suspended payload dynamics model. This means that I only modeled the drone's translational motion and did not include the drone's rotational dynamics. However, the closed-loop software I developed for the drone in Section 6.1 shows that in addition to position control, the drone also needed to maintain a stable attitude for a stable flight. If only the drone-based cable-suspended payload model was used to tune the controllers, then only the position controllers could have been tuned. In order to tune the attitude controller, I added attitude dynamics to the theoretical model developed for the drone-based cable-suspended payload model. Then, I tuned both the attitude and position controllers simultaneously and accounted for the effect of the attitude controller commands on the position controller and the effect of the position controller commands on the attitude controller.

Figure 6.6 shows the simulation model block diagram. The position controller, force to command mapping, and attitude controller blocks are similar to those presented in Section 6.1.2. The drone-based cable-suspended dynamics are from Section 3.2. I used Euler's equations for rigid body rotational motion to simulate the drone's attitude dynamics,

The full non-linear form of Euler's equations for a rigid body are shown in Equation 6.39, Equation 6.40, and Equation 6.41 for rotation about the x-axis, y-axis, and z-axis, respectively.

$$I_x\ddot{\theta}_x + (I_z - I_y)\dot{\theta}_y\dot{\theta}_x = \tau_{\theta_x} \tag{6.39}$$

$$I_y\ddot{\theta}_y + (I_x - I_z)\dot{\theta}_x\dot{\theta}_z = \tau_{\theta_y} \tag{6.40}$$

$$I_z\ddot{\theta}_z + (I_y - I_x)\dot{\theta}_x\dot{\theta}_y = \tau_{\theta_z} \tag{6.41}$$

where $I_x = 0.01\ kgm^2$, $I_y = 0.0082\ kgm^2$, and $I_z = 0.0148\ kgm^2$ represent the QDrone's inertia in the three principal axes [118]. Then, using the total torque

command from the position and attitude controllers, and the inertia of the QDrone, I simulated the attitude and attitude rate of the drone using the fourth-order Runge-Kutta integrator.

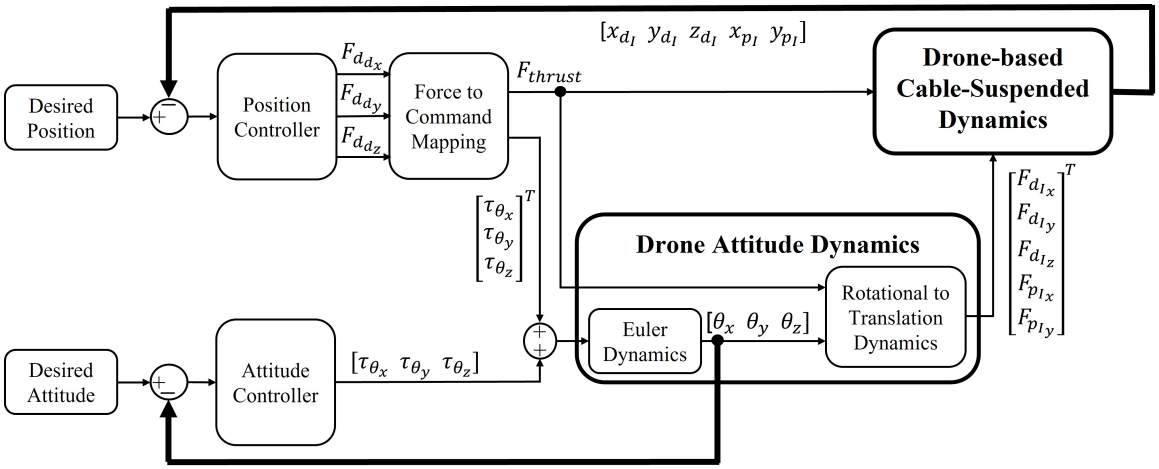


Figure 6.6: Simulation block diagram with both attitude dynamics and drone-payload dynamics.

Next, using the commanded thrust force (F_{thrust}) and the attitude of the drone ($\theta_x,\ \theta_y,\ \theta_z$), I determined the drone's acceleration in the drone's body frame ($\ddot{x}_{d_d},\ \ddot{y}_{d_d},\ \ddot{z}_{d_d}$) using Equation 6.42.

$$\begin{bmatrix}\ddot{x}_{d_d}\\ \ddot{y}_{d_d}\\ \ddot{z}_{d_d}\end{bmatrix} = {C_{zyx}}^T \begin{bmatrix}0\\ 0\\ \frac{F_{thrust}}{M+m}\end{bmatrix} \tag{6.42}$$

where M is the mass of the drone, m is the mass of the payload, and C_{zyx} is a rotation matrix that performs three successive rotations in the yaw-pitch-roll order and can be calculated using Equation 6.43.

$$C_{zyx} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta_x) & sin(\theta_x) \\ 0 & -sin(\theta_x) & cos(\theta_x) \end{bmatrix} \begin{bmatrix} cos(\theta_y) & 0 & -sin(\theta_y) \\ 0 & 1 & 0 \\ sin(\theta_y) & 0 & cos(\theta_y) \end{bmatrix} \begin{bmatrix} cos(\theta_z) & sin(\theta_z) & 0 \\ -sin(\theta_z) & cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.43}$$

I calculated the drone's acceleration in the inertial body frame ($\ddot{x}_{d_I},\ \ddot{y}_{d_I},\ \ddot{z}_{d_I}$) using Equation 6.44, where g is the acceleration due to gravity.

$$\begin{bmatrix} \ddot{x}_{d_I} \\ \ddot{y}_{d_I} \\ \ddot{z}_{d_I} \end{bmatrix} = \begin{bmatrix} \ddot{x}_{d_d} \\ \ddot{y}_{d_d} \\ \ddot{z}_{d_d} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} \tag{6.44}$$

Using the drone's acceleration in the inertial frame and the commanded thrust force, I calculated the net force applied to the drone and payload in the drone-based cable-suspended dynamics model using Equation 6.45.

$$\begin{bmatrix} F_{d_{I_x}} \\ F_{d_{I_y}} \\ F_{d_{I_z}} \\ F_{p_{I_x}} \\ F_{p_{I_y}} \end{bmatrix} = (M+m) \begin{bmatrix} \ddot{x}_{d_I} \\ \ddot{y}_{d_I} \\ \ddot{z}_{d_I} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ F_{thrust} \\ 0 \\ 0 \end{bmatrix} \tag{6.45}$$

Finally, I applied these forces to the drone-based cable-suspended model and simulated the system response using the fourth-order Runge-Kutta integrator, similar to the simulation setup shown in Section 3.3.2.

In summary, I used the drone-based cable-suspended model to simulate the translation motion of the drone and payload, while I used Euler's dynamics to simulate the rotational motion of the drone. I fed the position of the drone and payload to the position controller, and the attitude of the drone to the attitude controller, to form a closed-loop feedback model that simulated the effects of the sway of the payload on the drone and the motion of the drone on the sway of the payload.

I used this simulation model, in combination with Particle Swarm Optimization (PSO), to tune the IO-controller and FO-controller parameters and to simulate the expected response of the system before performing flight tests using the Quanser QDrone. I chose PSO because I wanted an automated method to fairly compare each controller without the implicit bias of manual tuning capabilities. Using a numerical optimizer, such as PSO, ensured that both controllers were tuned optimally per the cost function I designed.

6.2.2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) algorithm is an iterative method to find the optimal solution for a given criterion [119]. The algorithm is initialized using a population of particles with assigned initial velocities. Each particle represents a candidate solution to the optimization problem. The algorithm uses an objective function (sometimes referred to as a cost function) as a criterion to evaluate the candidate solutions. With each iteration, the algorithm updates the velocity and location of the particles using the current velocity of the particles, the best location of each particle, and the best locations of the particles in its neighborhood. As the algorithm progresses, the particles slowly move toward the best solution (global minima of the cost function) within the search space. The algorithm stops when a stopping criterion is reached, often based on either the successive changes in the best position or the number of

iterations. The stopping criterion that I used for this thesis was the relative change between objective function values for a set of 20 iterations (*i.e.*, if the objective function value did not change by more than 0.001 over 20 successive iterations, then PSO would terminate).

Compared to other optimization algorithms, such as the genetic algorithm [120] or gradient descent [121], PSO provides the global optimum solution independent of the initial point with faster convergence towards the global optimum solution. I selected PSO for this thesis since the number of tuning parameters for both the integer-order controllers and the fractional-order controllers were large. With three position PID controllers (for x, y, and z-positions) and three attitude PD controllers (for roll, pitch, and yaw), I needed to tune 15 parameters for the integer-order controllers and 24 parameters for the fractional-order controllers. Using rule-based tuning methods (such as Ziegler-Nichols [122]), analytical tuning methods, or manual tuning for a large number of parameters would have been cumbersome and potentially biased towards a controller that I thought should have been better from the outset of this research project. Therefore, I chose an autonomous tuning algorithm to remove the bias and ensure that each candidate controller was tuned using the same heuristics. Further, rule-based tuning, analytical tuning, and manual tuning are not exhaustive and may not result in the best controller parameters for a given model. With a well-defined objective function, PSO served as a great tool for controller tuning in this thesis.

I used MATLAB's *particleswarm* function to run the PSO algorithm. The objective function I used to evaluate the performance of the integer-order controllers and fractional-order controllers is shown in Equation 6.46.

$$f_{obj} = f_{obj}^{x} + f_{obj}^{y} + f_{obj}^{z} + f_{obj}^{\theta_x} + f_{obj}^{\theta_y} + f_{obj}^{\theta_z} \tag{6.46}$$

where f_{obj}^{x}, f_{obj}^{y}, and f_{obj}^{z} are the objective functions that evaluate the translation performance of the payload's position in the x-axis, y-axis, and z-axis, respectively.

$$f_{obj}^{x} = t_{s_x} \tag{6.47}$$

$$f_{obj}^{y} = t_{s_y} \tag{6.48}$$

$$f_{obj}^{z} = t_{s_z} \tag{6.49}$$

where t_{s_x}, t_{s_y}, and t_{s_z} represent the settling time of the payload's position in each axis. I calculated the settling time using a 2% threshold on the desired position. $f_{obj}^{\theta_x}$, $f_{obj}^{\theta_y}$, and $f_{obj}^{\theta_z}$ are the objective functions that evaluate the attitude performance of the drone in roll, pitch, and yaw, respectively. The rotational motion objective functions are the sum the of squared error (SSE) of the roll, pitch, and yaw response of the drone's attitude.

$$f_{obj}^{\theta_x} = \sum \left(\theta_{xdes} - \theta_{xmeas}\right)^2 \tag{6.50}$$

$$f_{obj}^{\theta_y} = \sum \left(\theta_{y_{des}} - \theta_{y_{meas}}\right)^2 \tag{6.51}$$

$$f_{obj}^{\theta_z} = \sum \left(\theta_{zdes} - \theta_{zmeas}\right)^2 \tag{6.52}$$

I selected this objective function to provide a hybrid and robust performance evaluator of the overall performance of the drone-payload system. Using the settling time of the payload as a criterion to evaluate the performance of the system ensured that the optimizer selected controller parameters that minimized the sway of the payload. Further, from a practical standpoint, a faster settling time would imply that the payload motion would dampen faster, hence, for a given survey area, the GPR data could be collected over a shorter period of time. The SSE of the drone's attitude ensured that oscillations in the drone's attitude were minimized, which resulted in a stable flight for the drone.

Figure 6.7 shows how I implemented the PSO for controller tuning in this thesis. PSO took as input the desired trajectory, desired attitude, and a range for each tunable parameter (upper bound and lower bound) to form a search space.

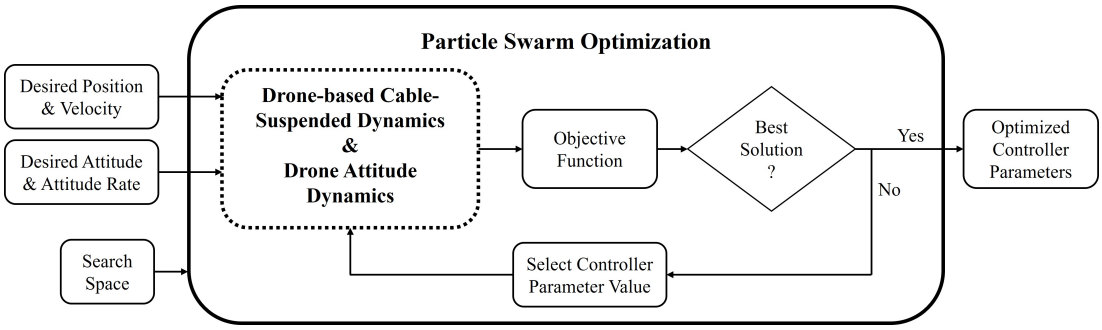


Figure 6.7: PSO for tuning controller parameters.

The PSO algorithm selected a set of gains for the controller being tuned (15 parameters for integer-order controllers, 24 parameters for fractional-order controllers) and ran a simulation for the selected desired trajectory. Next, the output from the simulation was used to evaluate the objective function. If the PSO algorithm determined that the stopping criterion had not been triggered, it selected a new set of controller parameters and re-ran the simulation. Once the stopping criterion was triggered, the controller parameters that resulted in the minimum objective function value were outputted.

To tune both the integer-order controllers and fractional-order controllers, I simulated a drone carrying a payload with a 1 m cable length using a time step of 0.001 s. The mass of the drone was 1.1 kg while the mass of the payload was 0.175 kg. I selected these parameters to match the actual cable length and masses of the QDrone and the payload that I used for my physical tests. The simulation time step also matched that of the QDrone. Then, I used a desired trajectory that first commanded the drone to take off followed by simultaneously providing a step command in the x-axis and y-axis, as shown in Figure 6.8. I set the desired attitude of the drone to zero.

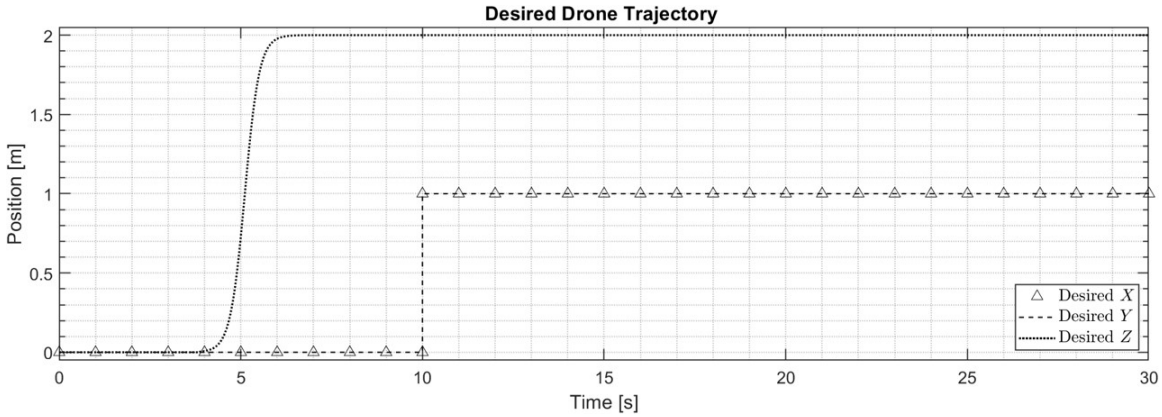


Figure 6.8: The desired trajectory used to tune the controllers using PSO.

The resulting controller parameters for the integer-order position (IOPID) and attitude (IOPD) controller, and the fractional-order position (FOPID) and attitude (FOPD) controller are shown in Table 6.1. I used these controller parameters to simulate the dynamics presented in Section 6.2.1 and performed the hardware tests using the closed-loop software I developed in Section 6.1.2. The following sections provide the results of the simulation and hardware tests.

Note that in the following sections of this thesis, I refer to the combination of the IOPID position controller and IOPD attitude controller as the IO-controller. Using a similar notation, I refer to the combination of the FOPID position controller and FOPD attitude controller as the FO-controller.

Table 6.1: Optimized Controller Parameters

	Parameter	**IO-controller**	**FO-controller**
Position Controller for x-axis	K_{p_x} (proportional gain)	1.1304	1.9620
	K_{i_x} (integral gain)	0.0015	0.0113
	K_{d_x} (derivative gain)	2.7938	3.7783
	ξ_x (integral order)	1	0.3326
	ζ_x (derivative order)	1	1.0108
Position Controller for y-axis	K_{p_y}	1.8656	2.3385
	K_{i_y}	0.0085	0.0072
	K_{d_y}	3.9026	4.1551
	ξ_y	1	1.2246
	ζ_y	1	1.0212
Position Controller for z-axis	K_{p_z}	49.7706	48.4369
	K_{i_z}	5.8050	26.3941
	K_{d_z}	14.8683	48.4726
	ξ_z	1	0.9115
	ζ_z	1	0.5715
Attitude Controller for Roll	$K_{p_{\theta_x}}$	1.9775	1.5186
	$K_{d_{\theta_x}}$	0.4359	0.4170
	ζ_{θ_x}	1	0.9867
Attitude Controller for Pitch	$K_{p_{\theta_y}}$	1.9931	1.8250
	$K_{d_{\theta_y}}$	0.2759	0.2152
	ζ_{θ_y}	1	1.4381
Attitude Controller for Yaw	$K_{p_{\theta_z}}$	0.6598	0.8969
	$K_{d_{\theta_z}}$	0.2866	0.0978
	ζ_{θ_z}	1	0.9004

6.2.3 Simulation Results

I used the desired trajectory shown in Figure 6.8 and the controller parameters from Table 6.1 to simulate the response of the drone and the payload. The resulting position response of the drone, for the IO-controller and FO-controller is shown in Figure 6.9.

Both the optimally tuned IO-controller and FO-controller resulted in comparable performance in the drone's position response in the x-axis and y-axis. In both axes, neither controller resulted in an overshoot and instead caused the drone to approach the desired step command asymptotically. For the z-axis position response, the FO-controller was fast to respond and tracked the desired position accurately. The IO-controller tracked the desired take-off trajectory, but with a delay.

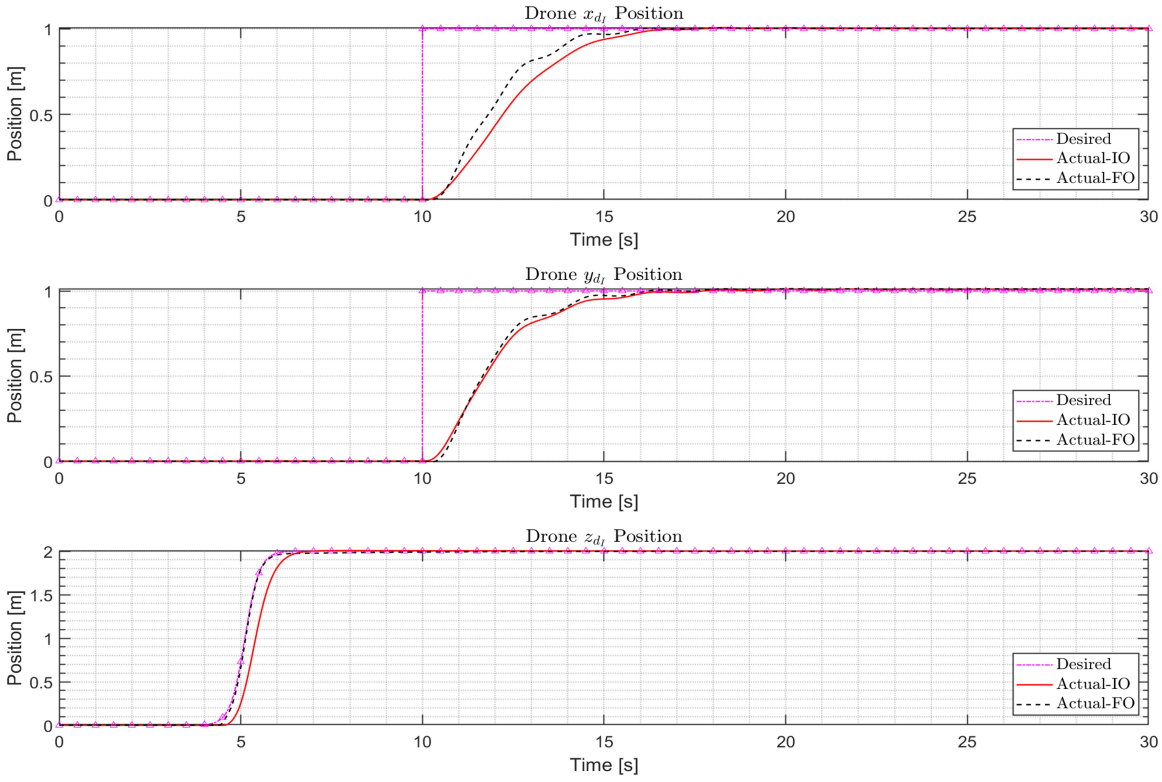


Figure 6.9: Simulation of drone's position response using IO and FO-controller.

The resulting payload position response in the x-axis and y-axis respectively is shown in Figure 6.10 and Figure 6.11. A 2% threshold is also plotted to show the boundary for the settling time.

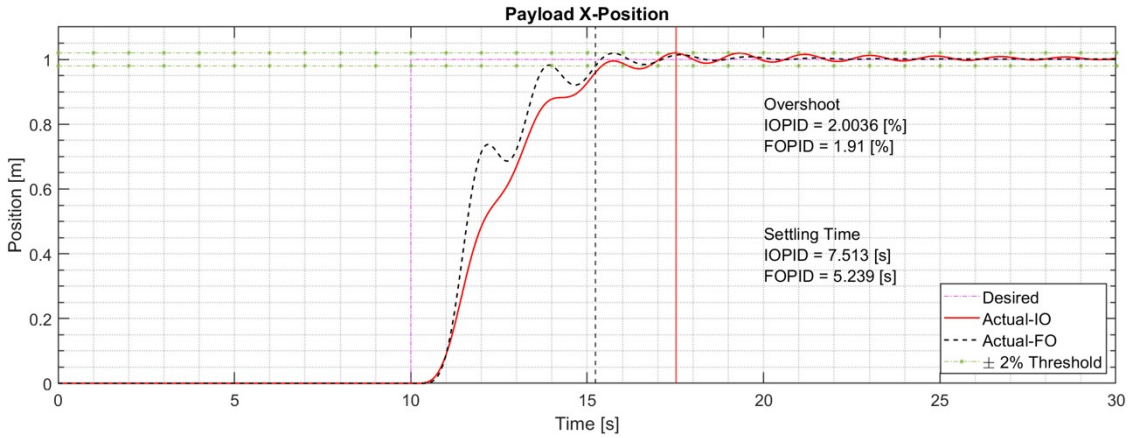


Figure 6.10: Simulation of payload's x-position using IO and FO-controller.

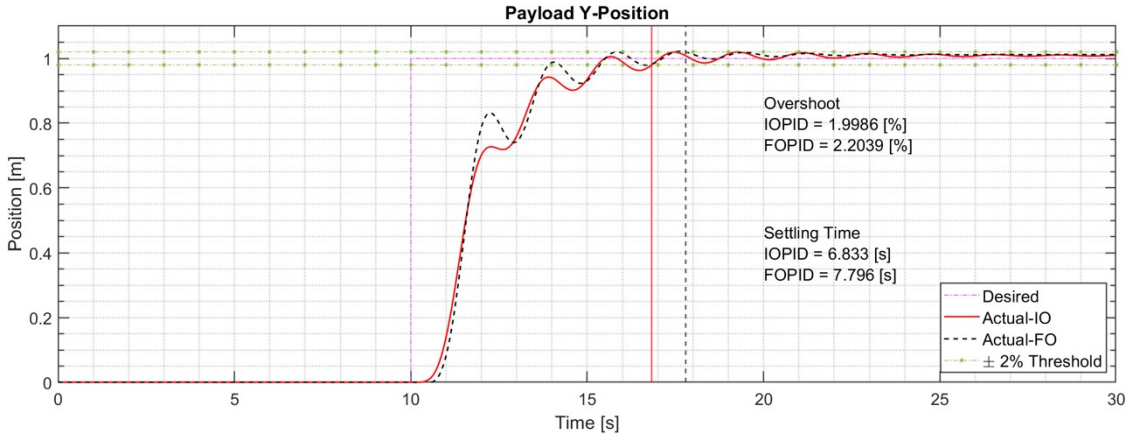


Figure 6.11: Simulation of payload's y-position using IO and FO-controller.

In the x-axis response, the payload settling time was smaller using the FO-controller (5.239 s) as compared to the settling time of the IO-controller (7.513 s). This corresponds to a 30.3% faster settling time using the FO-controller when compared to the IO-controller. However, in the y-axis, the payload settling time was larger using the FO-controller (7.796 s) as compared to the settling time of the IO-controller

(6.833 s). This corresponds to a 14.1% slower settling time using the FO-controller when compared to the IO-controller.

For each controller, I expected a difference in the response between the x-axis and y-axis since the inertia of the drone along the x-axis and y-axis were different. Further, the dynamics between the drone's x-axis and y-axis are coupled, as seen through Euler's rotational dynamics in Equation 6.39 to Equation 6.41. This means that changing the controller parameters in one axis would affect the performance in all three axes of the drone. Therefore, since the PSO algorithm optimized all the position and attitude controller parameters in one simulation, these coupled dynamics are captured in the simulated output, and the performance difference between the x-axis and y-axis was a compromise that needed to be made.

Both controllers resulted in a small but comparable percent overshoot in both the x-axis and y-axis response of the payload. The FO-controller resulted in a 1.91% and 2.20% overshoot in the x-axis and y-axis, respectively. The IO-controller resulted in a 2% overshoot in both the x-axis and y-axis. Also, both controllers resulted in a zero steady-state error in the drone's and payload's position response in all three axes.

The payload sway angle, represented as the angle between the cable and the vertical axis of the drone's reference frame, is shown in Figure 6.12.

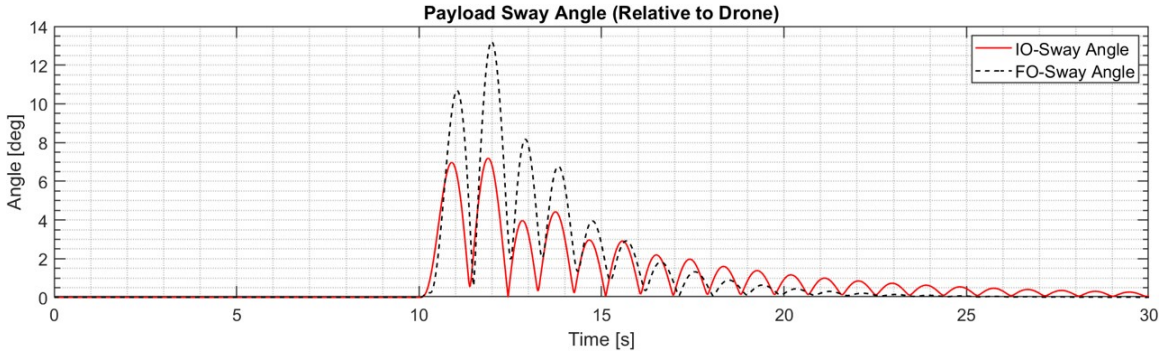


Figure 6.12: Payload's sway angle using IO-controller and FO-controller.

The FO-controller resulted in a large payload sway when compared to the IO-controller. The maximum sway angle using the FO-controller and IO-controller was approximately 13.2 degrees and 7.2 degrees, respectively. This is reflective of the faster rise time in the payload's x-position tracking using the FO-controller, as seen in Figure 6.10. Despite the larger sway of the payload, the FO-controller was able to dampen the motion of the payload faster than the IO-controller. At the end of the 30 s simulation time, the FO-controller had fully dampened the sway of the payload while the payload was still swaying using the IO-controller.

Overall, in simulation, the FO-controller (using the Oustaloup recursive filter) resulted in a marginal improvement over the IO-controller for a drone-payload system. The FO-controller resulted in a 30.3% faster settling time in the x-axis payload response, but it also resulted in a 14.1% slower settling time in the y-axis.

6.3 Experimental Test Results

I experimentally tested the IO-controller and FO-controller using a Quanser QDrone and a plumb bob as my payload. I attached the plumb bob to the QDrone using a cable of length approximately 1.05 m. Note that the cable length was 1 m, but the plumb bob added another 5 cm to the overall length. Additionally, I used the hardware and software described in Section 6.1 for my tests, and the experimental setup was similar to that shown in Figure 4.18.

For the tests, I commanded the drone to hover at a height of 1.3 m at the origin of the Vicon reference frame ($x_{d_I} = 0,\ y_{d_I} = 0$) so that the payload would be 0.25 m off the ground (at $x_{p_I} = 0,\ y_{p_I} = 0$). Then, I provided the drone with a step command in the x-axis, followed by a step command in the y-axis, and then a command to return to the origin of the x-y plane ($x_{d_I} = 0,\ y_{d_I} = 0,\ z_{d_I} = 1.3$). The resulting position of the drone and payload using the IO-controller is shown in Figure 6.13 and Figure 6.14, respectively.

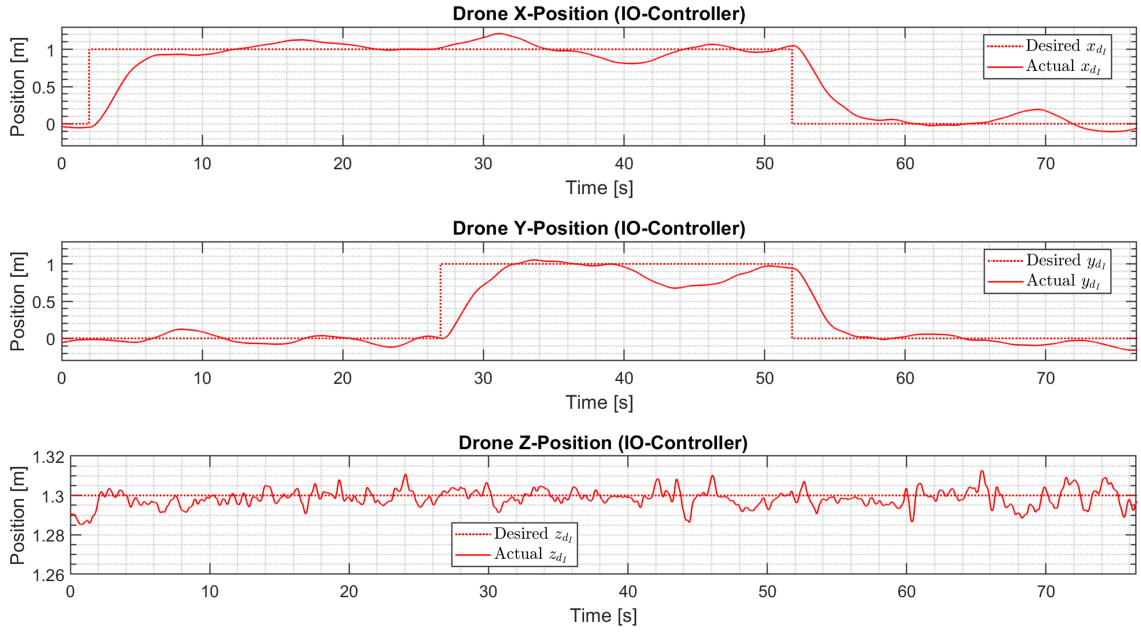


Figure 6.13: Experimental results of drone's position response using IO-controller.

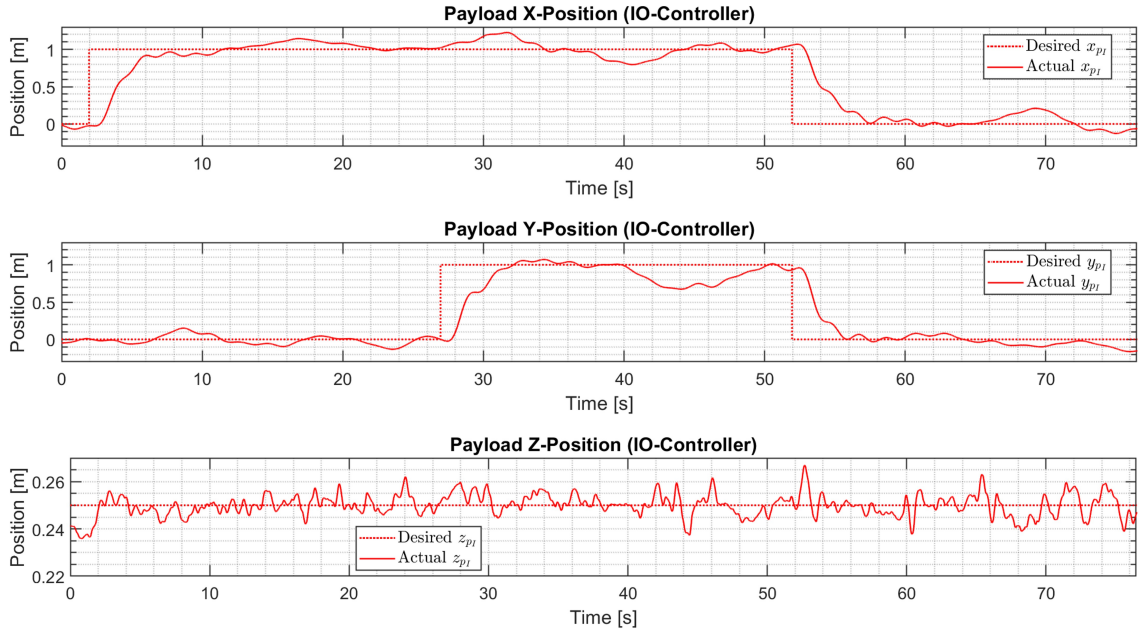


Figure 6.14: Experimental results of payload's position response using IO-controller.

In the z-axis, the drone was able to track the desired position accurately. In the x-axis response, the drone approached the desired position asymptotically as in the simulation, however, the drone suffered from a steady state error after approaching the desired step command. This was reflected in the payload's x-position as well. It took approximately 10 s for both the drone and payload to reach the desired x-position, compared to the ~ 6.5 s seen in the simulation. The drone's and payload's y-position response matched the simulated y-position response more accurately. Both the drone and payload reached the desired y-position in approximately 6 s. The sway of the payload in both the x-axis and y-axis was minimal as seen by the small ripples in Figure 6.14

Both the x-axis and y-axis response of the drone and payload position show that the drone was not able to maintain its desired position. The drone drifted from its desired position. The drift was bounded within approximately ± 20 cm of the desired position, indicating that the drone was not unstable (in a Lyapunov sense). The drift resulted from a combination of low position controller gains, high attitude controller gains, a delay in position feedback (described in Section 6.1.1), and a noisy and phase-delayed velocity estimate. The following describes how each of these contributors likely affected the quadcopter's performance.

As mentioned previously, the QDrone is an underactuated system that cannot independently control its position and attitude simultaneously (*i.e.,* a quadcopter can only change its position by changing its attitude). Therefore, for the drone to maintain a stable flight, the drone's attitude must always remain stable, which requires a minimum set of attitude controller gains.

For the quadcopter to hold its position, the position controller commands need to match or be higher than the attitude controller commands. Higher position controller gains can solve the drifting problem, but it can cause the drone to move toward the

desired position faster, thereby causing the payload to sway more. Since the goal for these experiments was to minimize the sway of the payload, the optimizer chose low position controller gains as they would move the drone slowly towards the desired position, thereby minimizing the payload's sway.

With low position gains, once a position change is commanded, the position error is large which causes the position controller command to be larger than the attitude command, allowing the drone to move toward the desired position. However, once the drone reaches the desired position, the position controller commands become smaller and insignificant compared to the attitude controller commands. Then, if external disturbances are large enough, such as wind resulting from the downwash of the drone's propellers, the disturbances can cause the drone to drift.

Additionally, the delay in transferring position data between the Vicon system and the QDrone's flight controller can affect the drone's ability to maintain its position since the delay induces a lag in the system. The delay also affects the drone's velocity estimate since I determined the drone's velocity using its position, as discussed in Section 6.1.2.1.

Using the same hardware and desired trajectory, the resulting position of the drone and payload using the FO-controller is shown in Figure 6.15 and Figure 6.16, respectively. In the z-axis, the drone was able to track the desired position slightly more accurately when compared to the IO-controller. In both the x-axis and y-axis, the drone approached the desired position asymptotically as in the simulation. However, the drone suffered from a slow response. In both axes, the FO-controller had a longer rise time than the IO-controller. This was reflected in the payload's position response as well. It took approximately 14 s for both the drone and payload to reach the desired x-position, compared to the ~ 6 s seen in the simulation. Further, it took approximately 21 s for both the drone and payload to reach the desired y-position,

compared to the $\sim 6\ s$ seen in the simulation. The longer rise time may have resulted from the delay in position and velocity feedback from the Vicon system.

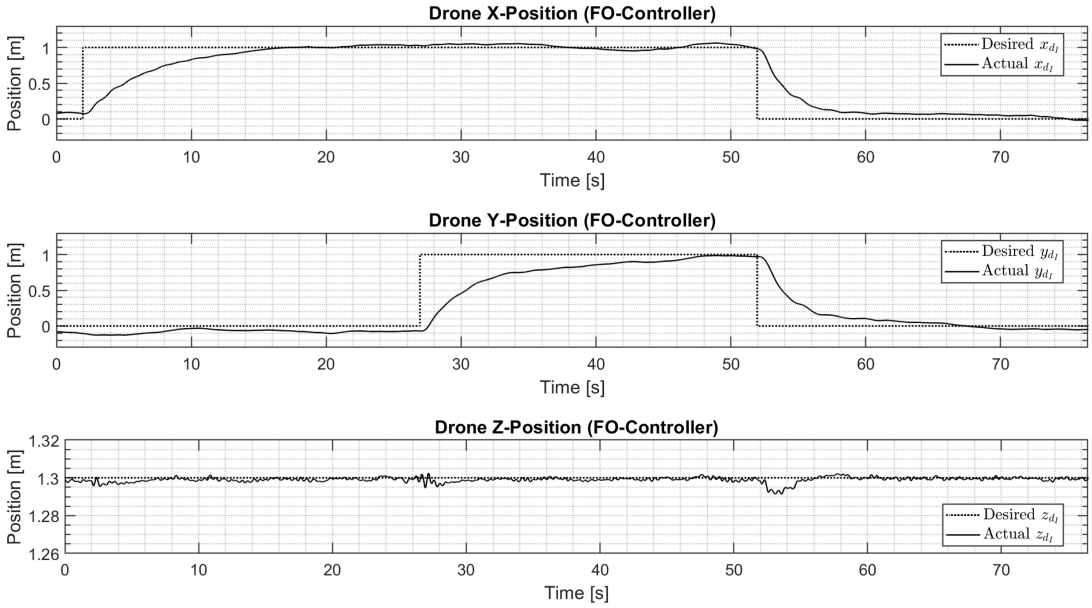


Figure 6.15: Experimental results of drone's position response using FO-controller.

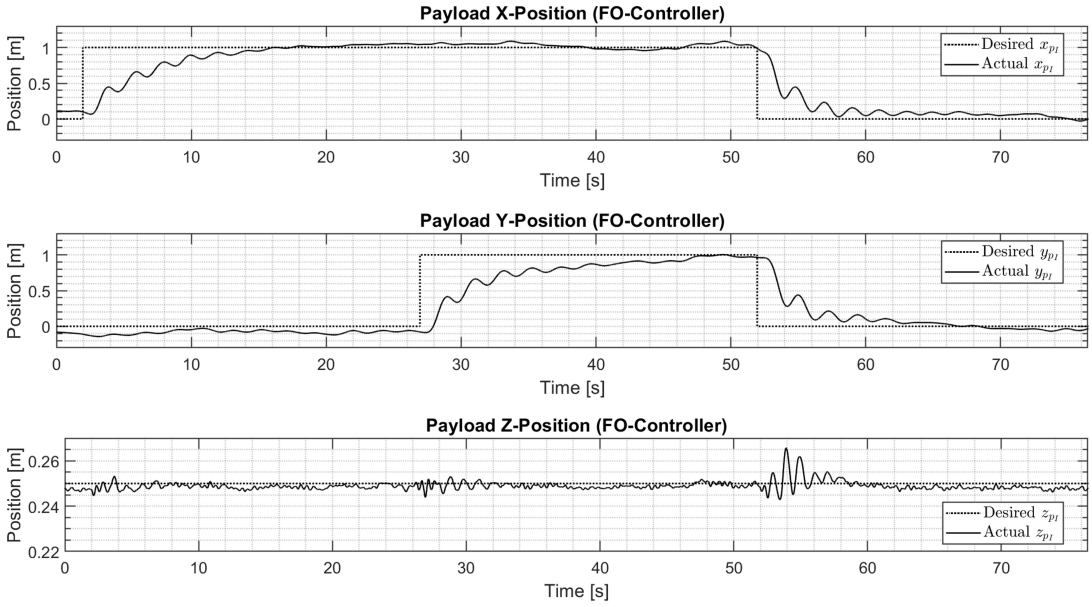


Figure 6.16: Test results of payload's position response using FO-controller.

The sway of the payload in both the x-axis and y-axis was larger compared to the IO-controller response, as seen by the larger ripples in Figure 6.16. The x-axis and y-axis response of the drone and payload position show that the drone was able to maintain its desired position. The drone did drift from its the desired position, but the drift was minimal and was bounded within approximately ± 10 cm of the desired position, thereby showing that the FO-controller (using the Oustaloup recursive filter) was more robust to disturbances when compared with the IO-controller.

Overall, the simulation results showed that the FO-controller (using the Oustaloup recursive filter) had a marginally shorter settling time than the IO-controller, but resulted in larger sway of the payload. Physical test results indicated that the FO-controller resulted in larger payload sway compared to the IO-controller. Physical tests also revealed that the FO-controller had a significantly longer rise time than the IO-controller, showing that the FO-controller was more sensitive to delays in the closed-loop system.

6.4 Discussion

These findings suggest that for a non-linear drone-based cable-suspended payload system, the FO-controller (using the Oustaloup recursive filter) does not significantly improve the performance of the system. In an ideal scenario, such as the simulated scenario in Section 6.2.3, the FO-controller and IO-controller showed comparable time responses in the x-axis and y-axis, with the FO-controller being marginally faster than the IO-controller. In a real-world scenario, such as the physical tests using the QDrone, the FO-controller may be better at handling unknown disturbances, but it may suffer from a slower response that results from lag in the system. Further, the results suggest that using numerical optimizers, such as the PSO algorithm, the IO-controller can be tuned to provide a comparable time response to the FO-controller.

The drawbacks in the IO-controller and FO-controller discussed above can be solved by using higher controller gains for the drone's position controller. However, this may result in larger sway of the payload if only a feedback control law is used. To have a better position control response of the drone and minimize the sway of the payload, system dynamics need to be part of the controller, which can be achieved using model-based techniques [123, 124].

For a non-linear drone-based suspended payload system, model-based controllers or command shaping techniques may be a better option than using IOPID or FOPID feedback controllers. Model-based controllers, such as the Sliding Mode Control (SMC) [70], utilize the system's model to determine the controller commands. Command shaping techniques [124] also use the system's dynamics model to generate an optimal trajectory for the drone to follow in order to minimize the payload sway.

The objective of this thesis was to compare the performance of the feedback IO-controller and the feedback FO-controller for a drone-payload system, and not to design the best controller. Therefore, the scope of this thesis was limited to designing a simple feedback control system, with no model dynamics, using the IO-controller and the FO-controller, and comparing the performance of the two controllers.

But, to briefly compare the performance of a model-based approach for this drone-based cable-suspended payload system to the IO-controller and FO-controller I presented earlier, I designed a sliding mode controller using the translation dynamics model of the drone-payload system from Section 3.2, and the attitude dynamics using Euler's rigid body dynamics (following the methods of [70]). I tuned the sliding mode controller using PSO, similar to the IO-controller and FO-controller. Then, I used the desired trajectory shown in Figure 6.8 to simulate the response of the drone and the payload. The resulting position response of the drone using the sliding mode controller is shown in Figure 6.17.

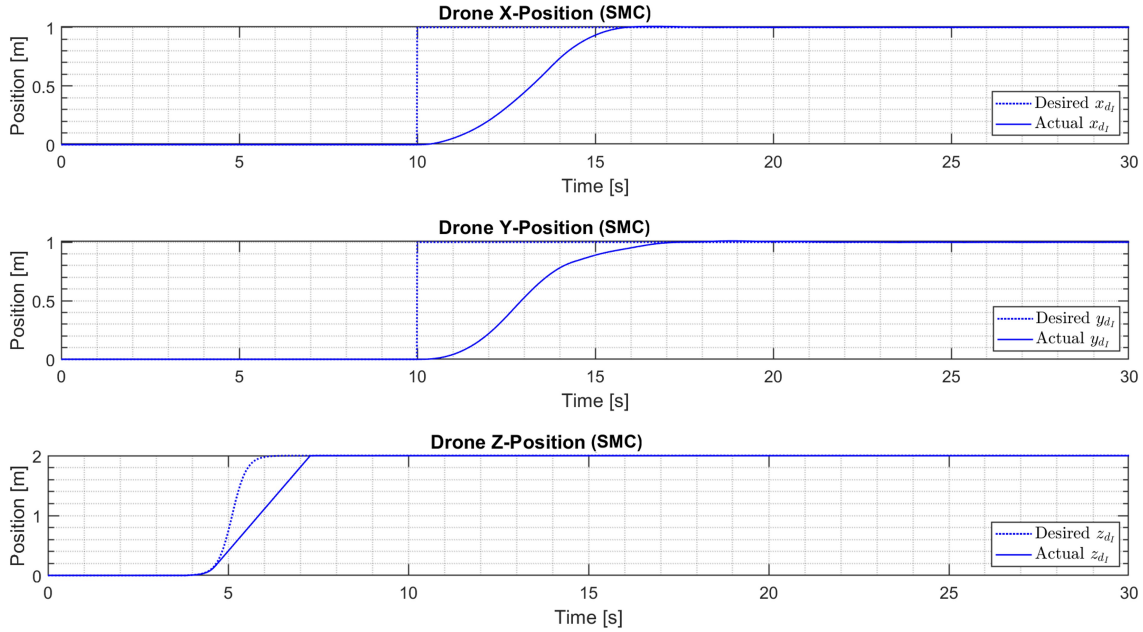


Figure 6.17: Simulation of drone's position response using SMC.

The optimally tuned sliding mode controller resulted in comparable performance in the drone's position response in the x-axis and y-axis. In both axes, the drone did not experience an overshoot but instead approached the desired step command asymptotically. For the z-axis position response, the sliding mode controller resulted in a linear increase in altitude, which caused the z-axis response of the drone's position to be slower than the IO-controller and FO-controller.

The corresponding payload's position in the x-axis and y-axis respectively is shown in Figure 6.18 and Figure 6.19. In both the x-axis and y-axis respectively, the payload settling time using the sliding mode controller was 5.429 s and 6.093 s, which was a shorter settling time than with the IO-controller. Compared to the FO-controller, the sliding mode controller settled $\sim$ 21.8% faster in the y-axis, but was marginally ($\sim$ 3%) slower in the x-axis. The sliding mode controller also resulted in a comparable but small percent overshoot in both the x-axis (1.95%) and y-axis (and 1.96%) position response of the payload.

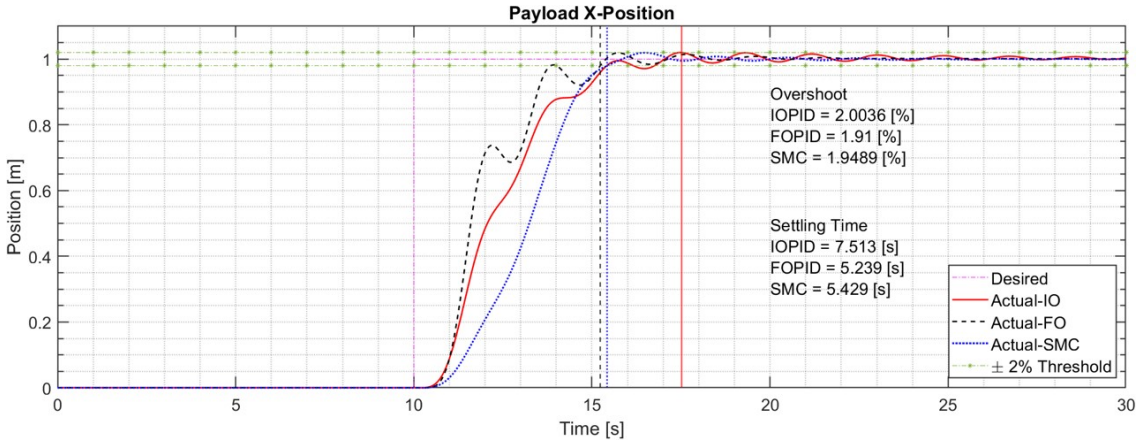


Figure 6.18: Simulation of payload's x-position using IO, FO, and SMC controller.

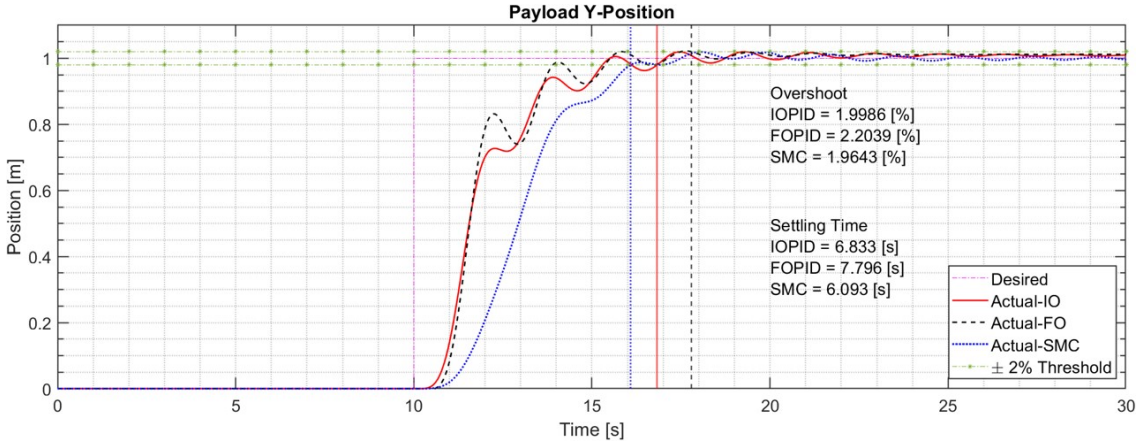


Figure 6.19: Simulation of payload's y-position using IO, FO, and SMC controller.

I experimentally tested the sliding mode controller using the same hardware as the IO-controller and the FO-controller. However, the sliding mode controller resulted in chattering, an inherent issue with sliding mode controllers [70]. For sliding mode controllers, once the actual trajectory reaches the desired sliding surface, the controller often tries to keep the actual trajectory along the sliding surface. This means that if the drone deviates from the sliding surface, the controller tries to pull it back on track. But, this can cause the controller commands to alternate directions (positive and negative), which results in chattering.

Figure 6.20 shows the position of the drone, and Figure 6.21 shows the position of the payload, using the sliding mode controller with the optimally tuned parameters.

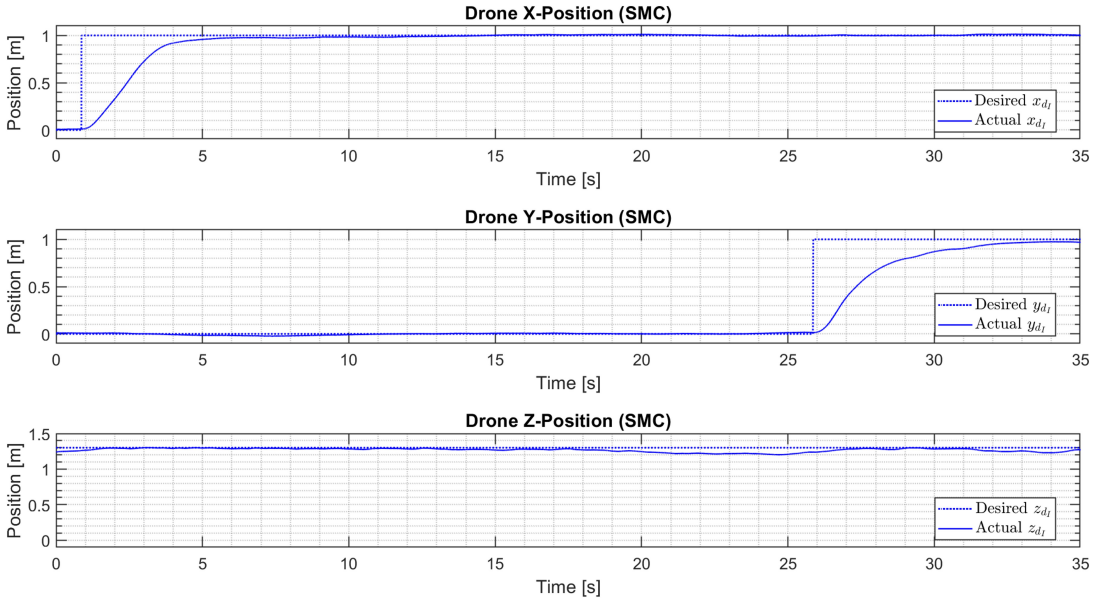


Figure 6.20: Experimental results of the drone's position using SMC.

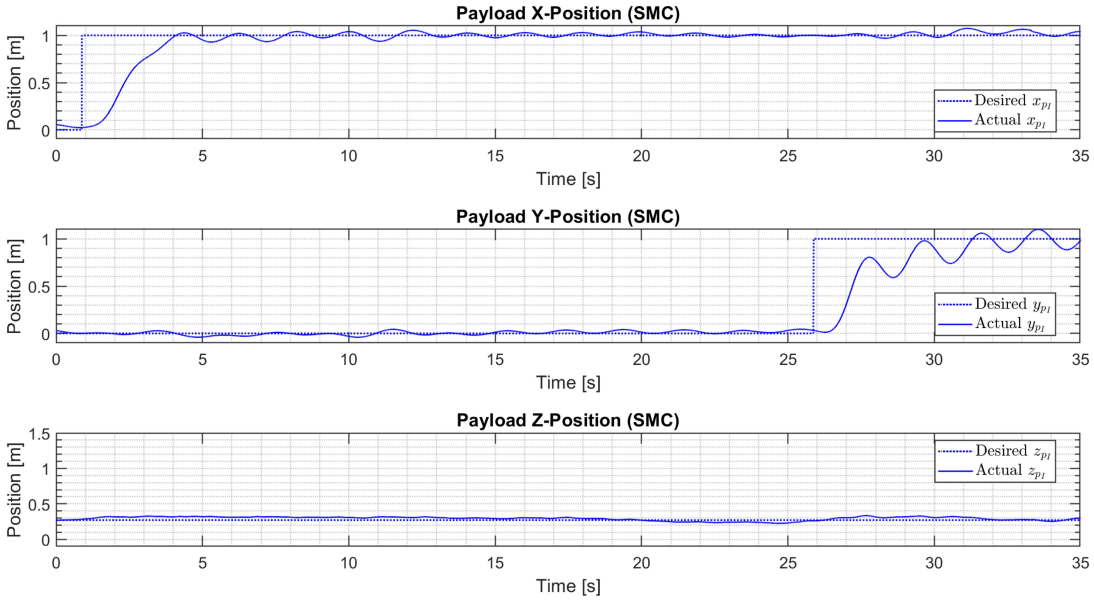


Figure 6.21: Experimental results of the drone's position using SMC.

The drone's position response in the x-axis matched the simulation results. The drone approached the desired position asymptotically and settled without an overshoot. In the x-axis, the drone took $\sim 6\ s$ to reach the desired position, comparable to the simulated response ($\sim 5.5\ s$). In the y-axis, the drone approached the desired position slightly slower ($\sim 8\ s$) than in the simulation ($\sim 6.5\ s$). Further, in all axes, the drone was able to maintain its position with minimal drift. In the x-axis, the payload's position response matched the simulation results with minimal payload sway. In the y-axis, the payload's sway was larger compared to the simulation results, however, this could have resulted from chattering in the drone's roll controller.

Note that during the test, the drone's motors overheated since the chattering caused the motors to cycle on and off at a high frequency. Hence, I had to stop the test before the completion of the y-axis position command. However, since the x-axis response from the experimental test matched the simulation results, I expect the y-axis response would have matched the simulation results as well, with the absence of chatter.

Chattering in my flight test occurred in the drone's attitude, mostly in the drone's roll response (as seen in Figure 6.22), since I commanded the drone to always maintain zero attitude about each axis. Additionally, the noisy measurements from the onboard accelerometer and gyroscope resulted in a noisy error input signal to the attitude controller, which also led to chattering in the drone's attitude controller commands. Chattering was not desired since it could potentially damage the drone's hardware (indeed, during these tests, the drone's motors overheated).

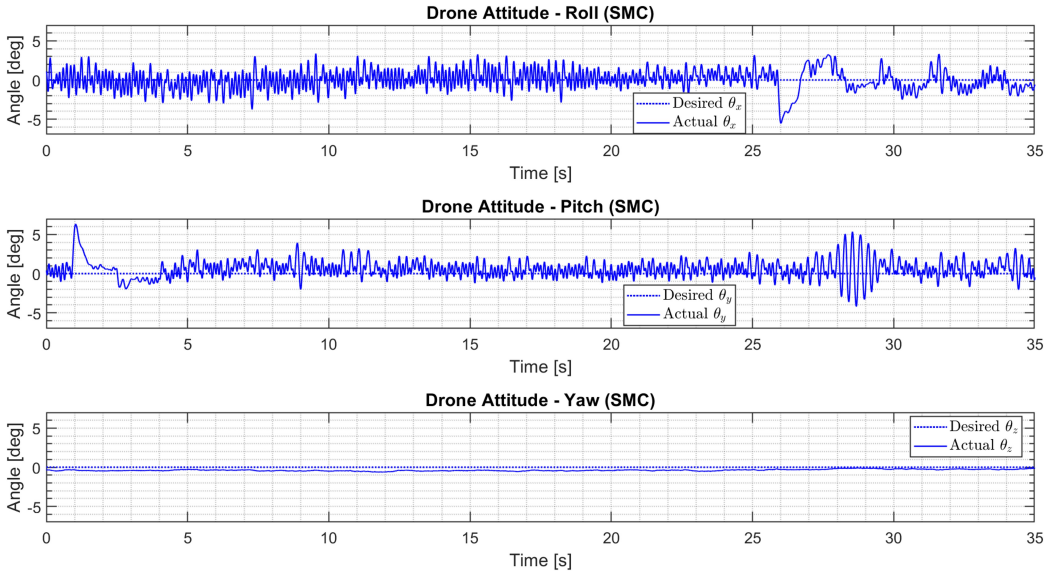


Figure 6.22: Experimental results of the drone's attitude using SMC.

As seen in the attitude response (Figure 6.22), the drone chattered about the y-axis (roll). The roll angle of the drone was oscillating between ±3 degrees at a very high frequency. The pitch response of the drone also experienced chatter, however, the frequency and magnitude of the oscillations in the pitch angle were lower compared to the roll response. The attitude controller commands also revealed that the sliding mode controller caused the drone to chatter. Figure 6.23 shows the attitude controller's torque commands in roll, pitch, and yaw using the sliding mode controller.

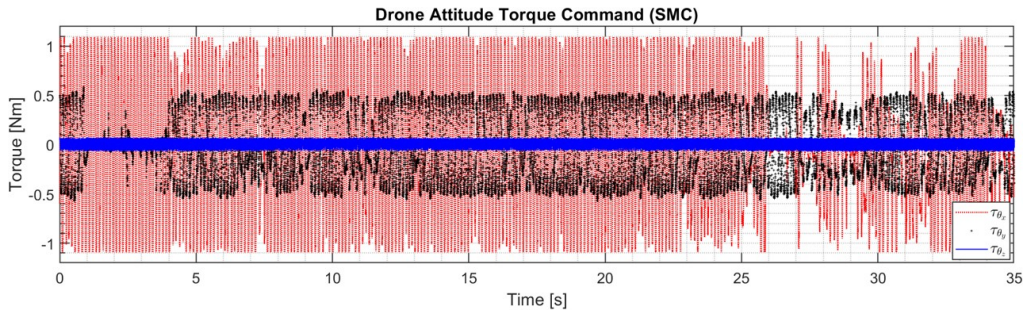


Figure 6.23: Experimental results of the drone's attitude commands using SMC.

Chattering in the roll controller command was evident since the roll controller commanded a maximum positive and maximum negative torque to the drone at a high frequency, as the controller tried to maintain a zero roll angle. In the pitch response, the controller also experienced chatter, however, the controller commands were of smaller magnitude when compared to the roll controller.

Overall, the simulation and experimental results showed that the sliding mode controller had a better performance than both the IO-controller and FO-controller. However, a standard sliding mode controller resulted in chattering in the drone's attitude, and chatter-suppressing algorithms, such as a terminal sliding mode controller [125] or higher-order sliding mode controller [74], would be needed.

6.5 Chapter Summary

To summarize the outcomes of this chapter, I used a Quanser QDrone with a cable-suspended payload to compare the performance of IO-controller and FO-controller. Simulation and experimental test results indicated that both controllers had comparable performance in minimizing the sway of the suspended payload if tuned appropriately. Both controllers were prone to external disturbances, such as the downwash from the propellers, but the IO-controller was more sensitive to disturbances than the FO-controller. Both controllers required higher position controller authority to reduce the drift in their position, but this could result in larger payload sway. This suggested that a simple feedback control system may not be the solution for such a non-linear system. A model-based approach was required to provide a balance between accurately controlling the drone's position and minimizing the sway of the payload. I designed and tested a sliding mode controller to verify the performance of a model-based approach for a drone-based cable-suspended system. Simulation results indicated that the sliding mode controller performed better than both the

IO-controller and FO-controller. Experimental results also indicated that a sliding mode controller performed better than the feedback controllers.

6.6 Hypothesis H2 Evaluation

Recall hypothesis H2:

> *A fractional-order PID (FOPID) controller, with both fractional differential and integral terms, provides better response to stabilizing the nonlinear dynamics of a drone payload swinging in 3D when compared to the traditional integer-order PID (IOPID) controller through reducing the percent overshoot by at least* 80%*, settling time by* 75% *and steady-state error by* 10%*.*

The simulation and experimental results of a feedback fractional-order control system for a drone-based cable-suspended system (using the Oustaloup recursive filter) did not support hypothesis H2. Instead, the results indicated that if a numerical optimizer, such as the particle swarm optimizer, is used to tune the integer-order controllers and the fractional-order controllers with a common objective function, the integer-order controller results in comparable performance to the fractional-order controller in terms of overshoot, settling time, and steady-state error (for the drone-payload system developed in this thesis using the Oustaloup recursive filter).

Chapter 7

Conclusions

This research project aimed to aid Arctic remote sensing studies by gaining dynamics and control system insight into a system that could carry a Ground Penetrating Radar (GPR) on a drone. Such a system would make it easier and safer to collect GPR data in the remote and inhospitable environment of the Arctic. I proposed using a drone and suspending the GPR from the drone via a cable. To achieve this goal, the first objective of this research was to design an onboard tracking algorithm to locate the position of the suspended payload using a Light Detection and Ranging (LiDAR) sensor and an Extended Kalman filter (EKF). The second objective was to evaluate and compare the performance of a closed-loop feedback fractional-order controller and integer-order controller for a non-linear drone-based cable-suspended payload system.

I proposed two hypotheses to achieve these objectives. My hypotheses were:

H1. A LiDAR sensor, in combination with a Kalman filter, can be used as a feedback sensor to track and estimate the position of a drone payload to an accuracy of 3 cm relative to the drone position.

H2. A fractional-order PID (FOPID) controller, with both fractional differential and integral terms, provides a better response to stabilizing the non-linear dynamics

of a drone payload swinging in 3D when compared to the traditional integer-order PID (IOPID) controller through reducing the percent overshoot by at least 80%, settling time by 75% and steady-state error by 10%.

To evaluate my hypothesis H1, I developed a mathematical model of a 3D pendulum. Next, I developed an algorithm that used the 3D pendulum model, a LiDAR sensor, and an EKF to track the position of the suspended payload relative to the drone. Then, I extended my analysis to a drone-based cable-suspended payload and developed a mathematical model of a drone carrying a suspended payload. Finally, I used this model, an EKF, and LiDAR measurements to test the accuracy of my tracking algorithm experimentally. I verified the results using Vicon's motion capture system as my truth measurements. Experimental results indicated that the tracking algorithm, using the LiDAR and EKF, located the position of the suspended payload relative to the drone with an accuracy of 2 cm in the x-direction, 1.7 cm in the y-direction, and 0.5 cm in the z-direction, demonstrating hypothesis H1 to be true.

To evaluate my hypothesis H2, I completed three tasks. First, I compared algorithms that perform fractional-order integration and differentiation suitable for real-time systems. I tested three continuous algorithms, namely Continuous Fraction Expansion (CFE), Matsuda's method, and Oustaloup's recursive filter, for approximating fractional-order transfer functions in the Laplace domain. I simulated each algorithm's response to approximate the magnitude and phase response of a fractional-order transfer function. CFE suffered from limited bandwidth, while Matsuda's approximation resulted in ripples in the phase response. The Oustaloup algorithm provided a better approximation than other algorithms. Therefore, I used the Oustaloup recursive filter when designing the fractional-order controller for the drone-payload system.

Second, I developed a closed-loop feedback flight control software with an integer-order controller and a fractional-order controller and tuned both controllers using Particle Swarm Optimization (PSO). I used the drone-based cable-suspended payload dynamics to simulate the translational motion of the drone and Euler's rigid body dynamics to simulate the rotational motion of the drone. Simulation results indicated that the optimally tuned integer-order controller had comparable performance to the fractional-order controller in terms of the payload position's overshoot, settling time, and steady-state error.

Third, I tested the response of both controllers experimentally in an indoor environment using a Qaunser QDrone. Experimental results demonstrated that the fractional-order controller resulted in a larger payload sway than the integer-order controller. However, the fractional-order controller was more resilient to external disturbances. Both feedback controllers could not hold the drone's position as desired, causing the drone and payload to drift. Higher position controller authority could fix the drift, however, this may cause a larger payload sway.

The experimental results suggested that a feedback control system for a drone-based cable-suspended system may not be sufficient in providing a balance between accurately controlling the drone's position and minimizing the sway of the payload. A model-based approach, such as a sliding mode controller, may be a better choice for such a non-linear application as it allows the controller to use the model dynamics to predict the behaviour of the drone and the payload. To examine this finding, I designed, tuned, simulated, and tested a sliding mode controller for this application. Both simulation and experimental results suggested that the sliding mode controller performed better than the feedback integer-order controller and fractional-order controller, showing that a model-based approach could be a better solution for this application.

Overall, this study indicated that with proper tuning, both the integer-order controller and fractional-order controller (using the Oustaloup recursive filter) achieve comparable performance, particularly for a non-linear drone-based cable-suspended payload, as summarized in Table 7.1. The fractional-order controller (using the Oustaloup recursive filter) may perform marginally better than the integer-order controller, however, the performance improvement is not significant. My study suggests that a simple feedback controller with no knowledge of the system dynamics may not be the best choice for this application and a model-based approach with sufficient mitigations to prevent actuator chattering should be considered for future studies aimed at controlling slung payloads with a drone.

Table 7.1: IO- vs FO-controller payload's simulation response.

Controller Type	**Settling Time**	**Overshoot**	**Steady-state Error**
IO-controller (x-axis)	7.51 s	2%	0 m
FO-controller (x-axis)	5.24 s	1.91%	0 m
SMC (x-axis)	5.43 s	1.95%	0 m
IO-controller (y-axis)	6.83 s	2%	0 m
FO-controller (y-axis)	7.80 s	2.20%	0 m
SMC (y-axis)	6.09 s	1.96%	0 m

Future work for this research could include the following:

- Perform outdoor tests using the LiDAR tracking algorithm and test the effect of cable length on the accuracy of the tracking algorithm.
- Analyze the response of various non-linear plants using both integer-order and fractional-order controllers tuned using numerical optimizers. This may provide a better understanding of scenarios where fractional-order control is better than integer-order control.

- Develop and test a chatter-free sliding mode controller for a drone-based cable-suspended system using chatter-suppressing algorithms and a more representative drone attitude dynamics model.
- Develop a custom drone capable of carrying the LiDAR and GPR system and perform a full-scale test in an outdoor environment.

References

[1] M. Patel and P. Ferguson, "Tracking and estimation of a swaying payload using a lidar and an extended kalman filter," in *2021 IEEE International Symposium on Robotic and Sensors Environments (ROSE)*, pp. 1–7, 2021.

[2] M. S. Patel and P. A. Ferguson, "Drone-based cable-suspended payload tracking and estimation using simulated lidar measurements and an extended kalman filter," in *AIAA SCITECH 2022 Forum*, 2022.

[3] I. Palunko, P. Cruz, and R. Fierro, "Agile load transportation : Safe and efficient load manipulation with aerial robots," *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 69–79, 2012.

[4] N. R. Council, *Issues in the Integration of Research and Operational Satellite Systems for Climate Research: Part I. Science and Design.* Washington, DC: The National Academies Press, 2000.

[5] D. Barber, "Microwave remote sensing, sea ice and arctic climate," *Phys. Can.*, vol. 61, January 2005.

[6] M. L. Harasyn, D. Isleifson, W. Chan, and D. G. Barber, "Multi-scale observations of the co-evolution of sea ice thermophysical properties and microwave brightness temperatures during the summer melt period in Hudson Bay," *Elementa: Science of the Anthropocene*, vol. 8, 2020.

[7] *National Resources Conservation Service*, 20 April 2021. [Online]. Available: www.nrcs. usda.gov/wps/portal/nrcs/detail/national/climatechange/research/?cid=stelprdb1048127.

[8] E. Mattei, F. Di Paolo, B. Cosciotti, S. E. Lauro, E. Pettinelli, S. E. Beaubien, and D. Barber, "Young sea ice electric properties estimation under non-optimal conditions," in *2017 9th International Workshop on Advanced Ground Penetrating Radar (IWAGPR)*, pp. 1–4, 2017.

[9] "Ice and snow," *Sensors and Software*, 19 October 2021. [Online]. Available: https://www.sensoft.ca/georadar/ice-road-bridge-thickness/.

[10] H. Liu, "New ground penetrating radar system for quantitative characterization of snow and sea ice," *IET Conference Proceedings*, p. 705, January 2013.

[11] "Agriculture and forestry," *Sensors and Software*, 10 May 2021. [Online]. Available: https://www.sensoft.ca/ground-penetrating-radar/agriculture-forestry/.

[12] V. Maria-Jose, *Notes from the Field - Radar Days on the Greenland Ice Sheet*. NASA Earth Observatory, [Online]. Available: https://earthobservatory.nasa.gov/blogs/fromthefield/2013/04/16/radar-days-on-the-greenland-ice-sheet/ [Accessed 5 December, 2022].

[13] C. Gaffey and A. Bhardwaj, "Applications of unmanned aerial vehicles in cryosphere: Latest advances and prospects," *Remote Sensing*, vol. 12, no. 6, 2020.

[14] X. He, G. Kou, M. Calaf, and K. K. Leang, "In-Ground-Effect Modeling and Nonlinear-Disturbance Observer for Multirotor Unmanned Aerial Vehicle Control," *Journal of Dynamic Systems, Measurement, and Control*, vol. 141, May 2019.

[15] R. A. Cabrera and G. Bekic, "Drone-borne ground-penetrating radar suitability for specific surveys: a comparative study of feature sizes versus antenna frequency and elevation over the ground," *First Break*, vol. 36, no. 8, pp. 83–89, 2018.

[16] J. C. Landy, D. Isleifson, A. S. Komarov, and D. G. Barber, "Parameterization of centimeter-scale sea ice surface roughness using terrestrial lidar," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 3, pp. 1271–1286, 2015.

[17] I. Kostial, L. Dorcak, and I. Petras, "Control quality enhancement by fractional order controllers," *Acta Montanistica Slovaca*, vol. 2, no. 3, pp. 143–148, 1998.

[18] E. C. de Oliveira and J. A. T. Machado, "A review of definitions for fractional derivatives and integral," *Mathematical Problems in Engineering*, vol. 2014, no. 6, 2014.

[19] I. Tejado, B. M. Vinagre, J. E. Traver, J. Prieto-Arranz, and C. Nuevo-Gallardo, "Back to basics: Meaning of the parameters of fractional order pid controllers," *Mathematics*, vol. 7, no. 6, 2019.

[20] M. T. Nasri and W. Kinsner, "An evaluation of integer- and fractional-order controllers for small satellites," in *2014 IEEE 13th International Conference on Cognitive Informatics and Cognitive Computing*, pp. 30–38, 2014.

[21] R. Cajo, C. Copot, C. M. Ionescu, R. De Keyser, and D. Plaza, "Fractional order pd path-following control of an ar. drone quadrotor," in *2018 IEEE 12th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, pp. 291–296, 2018.

[22] R. Caponetto, G. Dongola, L. Fortuna, and I. Petráš, *Fractional Order Systems: Modeling and Control Applications*, vol. 72 of *Series A*. Singapore: World Scientific, 2010.

[23] K. K. Dhiman, Abhishek, and M. Kotharic, "Flight dynamics and control of an unmanned helicopter with underslung double pendulum," *Journal of Aircraft*, vol. 59, no. 1, pp. 137–153, 2022.

[24] D. Scaramuzza, M. Achtelik, L. Doitsidis, F. Friedrich, E. Kosmatopoulos, A. Martinelli, M. W. Achtelik, M. Chli, S. Chatzichristofis, L. Kneip, D. Gurdan, L. Heng, G. H. Lee, S. Lynen, M. Pollefeys, A. Renzaglia, R. Siegwart, J. C. Stumpf, P. Tanskanen, C. Troiani, S. Weiss, and L. Meier, "Vision-controlled micro flying robots: From system design to autonomous navigation and mapping in gps-denied environments," *IEEE Robotics Automation Magazine*, vol. 21, no. 3, pp. 26–40, 2014.

[25] B. Nagabhushan, "Dynamic stability of a helicopter carrying a suspended payload," in *AIAA 4th Atmospheric Flight Mechanics Conference*, 1978.

[26] C. Adams, J. Potter, and W. Singhose, "Input-shaping and model-following control of a helicopter carrying a suspended load," *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 1, pp. 94–105, 2015.

[27] D. Fusato, G. Guglieri, and R. Celi, "Flight dynamics of an articulated rotor helicopter with an external slung load," *Journal of The American Helicopter Society*, vol. 46, pp. 3–13, 2001.

[28] M. Luis, C. Fernando, M.-d.-D. Ramiro, M. Iván, and O. Aníbal, "An unmanned aircraft system for automatic forest fire monitoring and measurement," *Journal of Intelligent and Robotic Systems*, vol. 65, pp. 533–548, 2012.

[29] N. A. Johnson and W. Singhose, "Dynamics and modeling of a quadrotor with a suspended payload," in *2018 Applied Aerodynamics Conference*, 2018.

[30] A. Altan and R. Hacıoğlu, "Model predictive control of three-axis gimbal system mounted on uav for real-time target tracking under external disturbances," *Mechanical Systems and Signal Processing*, vol. 138, p. 106548, 2020.

[31] B. Govindarajan and A. Sridharan, "Conceptual sizing of vertical lift package delivery platforms," *Journal of Aircraft*, vol. 57, no. 6, pp. 1170–1188, 2020.

[32] G. Loianno and V. Kumar, "Cooperative transportation using small quadrotors using monocular vision and inertial sensing," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 680–687, 2018.

[33] H. Lee and H. J. Kim, "Constraint-based cooperative control of multiple aerial manipulators for handling an unknown payload," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 6, pp. 2780–2790, 2017.

[34] T. Chen and J. Shan, "A novel cable-suspended quadrotor transportation system: From theory to experiment," *Aerospace Science and Technology*, vol. 104, 2020.

[35] A. Faust, I. Palunko, P. Cruz, R. Fierro, and L. Tapia, "Automated aerial suspended cargo delivery through reinforcement learning," *Artificial Intelligence*, vol. 247, pp. 381–398, 2017. Special Issue on AI and Robotics.

[36] S. Dai, T. Lee, and D. S. Bernstein, "Adaptive control of a quadrotor uav transporting a cable-suspended load with unknown mass," in *53rd IEEE Conference on Decision and Control*, pp. 6149–6154, 2014.

[37] F. A. Goodarzi, D. Lee, and T. Lee, "Geometric control of a quadrotor uav transporting a payload connected via flexible cable," *International Journal of Control, Automation and Systems*, vol. 13, no. 6, pp. 1486–1498, 2015.

[38] H. Sayyaadi and A. Soltani, "Modeling and control for cooperative transport of a slung fluid container using quadrotors," *Chinese Journal of Aeronautics*, vol. 31, no. 2, pp. 262–272, 2018.

[39] V. Kumar, Y. S. Saharawat, M. K. Gathala, A. S. Jat, S. K. Singh, N. Chaudhary, and M. Jat, "Effect of different tillage and seeding methods on energy use efficiency and productivity of wheat in the indo-gangetic plains," *Field Crops Research*, vol. 142, pp. 1–8, 2013.

[40] T. Lee, K. Sreenath, and V. Kumar, "Geometric control of cooperating multiple quadrotor uavs with a suspended payload," in *52nd IEEE Conference on Decision and Control*, pp. 5510–5515, 2013.

[41] V. Parra-Vega, A. Sanchez, C. Izaguirre, O. Garcia, and F. Ruiz-Sanchez, "Toward aerial grasping and manipulation with multiple uavs," *Journal of Intelligent & Robotic Systems*, vol. 70, no. 1, pp. 575–593, 2013.

[42] P. J. Cruz and R. Fierro, "Cable-suspended load lifting by a quadrotor uav: hybrid model, trajectory generation, and control," *Autonomous Robots*, vol. 41, no. 8, p. 1629–1643, 2017.

[43] X. Liang, Y. Fang, N. Sun, and H. Lin, "Dynamics analysis and time-optimal motion planning for unmanned quadrotor transportation systems," *Mechatronics*, vol. 50, pp. 16–29, 2018.

[44] S. Lupashin, M. Hehn, M. W. Mueller, A. P. Schoellig, M. Sherback, and R. D'Andrea, "A platform for aerial robotics research and demonstration: The flying machine arena," *Mechatronics*, vol. 24, no. 1, pp. 41–54, 2014.

[45] M. Zürn, K. Morton, A. Heckmann, A. McFadyen, S. Notter, and F. Gonzalez, "Mpc controlled multirotor with suspended slung load: System architecture and visual load detection," in *2016 IEEE Aerospace Conference*, pp. 1–11, 2016.

[46] E. L. de Angelis, "Swing angle estimation for multicopter slung load applications," *Aerospace Science and Technology*, vol. 89, pp. 264–274, 2019.

[47] C. Raimúndez and J. L. Camaño, "Transporting hanging loads using a scale quad-rotor," in *CONTROLO'2014 – Proceedings of the 11th Portuguese Conference on Automatic Control* (A. P. Moreira, A. Matos, and G. Veiga, eds.), (Cham), pp. 471–482, Springer International Publishing, 2015.

[48] Y. Alothman, W. Jasim, and D. Gu, "Quad-rotor lifting-transporting cable-suspended payloads control," in *2015 21st International Conference on Automation and Computing (ICAC)*, pp. 1–6, 2015.

[49] J. Estevez, J. M. Lopez-Guede, G. Garate, and M. Graña, "A hybrid control approach for the swing free transportation of a double pendulum with a quadrotor," *Applied Sciences*, vol. 11, no. 12, 2021.

[50] P. Adigbli, C. Grand, J.-B. Mouret, and S. Doncieux, "Nonlinear attitude and position control of a micro quadrotor using sliding mode and backstepping techniques," in *7th European Micro Air Vehicle Conference (MAV07)*, pp. 1–9, 2007.

[51] M. Santos, V. López, and F. Morata, "Intelligent fuzzy controller of a quadrotor," in *2010 IEEE International Conference on Intelligent Systems and Knowledge Engineering*, pp. 141–146, 2010.

[52] A. Basri, A. Husain, and K. A. Danapalasingam, "Backstepping controller with intelligent parameters selection for stabilization of quadrotor helicopter," *Journal of Engineering Science and Technology Review*, vol. 7, pp. 66–74, 2014.

[53] B. Prabhakaran, M. Kothari, and Abhishek, "Nonlinear control design for quadrotors," in *2015 IEEE Workshop on Computational Intelligence: Theories, Applications and Future Directions (WCI)*, pp. 1–6, 2015.

[54] D. Mellinger, M. Shomin, N. Michael, and V. Kumar, "Cooperative grasping and transport using multiple quadrotors," in *Distributed autonomous robotic systems*, pp. 545–558, Springer, 2013.

[55] K. Sreenath, N. Michael, and V. Kumar, "Trajectory generation and control of a quadrotor with a cable-suspended load - a differentially-flat hybrid system," in *2013 IEEE International Conference on Robotics and Automation*, pp. 4888–4895, 2013.

[56] I. Palunko, R. Fierro, and P. Cruz, "Trajectory generation for swing-free maneuvers of a quadrotor with suspended payload: A dynamic programming approach," in *2012 IEEE International Conference on Robotics and Automation*, pp. 2691–2697, 2012.

[57] K. Sreenath and V. Kumar, "Dynamics, control and planning for cooperative manipulation of payloads suspended by cables from multiple quadrotor robots," 2013.

[58] X. Liang, Y. Fang, N. Sun, and H. Lin, "Nonlinear hierarchical control for unmanned quadrotor transportation systems," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 4, pp. 3395–3405, 2018.

[59] S. Yang and B. Xian, "Energy-based nonlinear adaptive control design for the quadrotor uav system with a suspended payload," *IEEE Transactions on Industrial Electronics*, vol. 67, pp. 2054–2064, March 2020.

[60] M. M. Nicotra, E. Garone, R. Naldi, and L. Marconi, "Nested saturation control of an uav carrying a suspended load," in *2014 American Control Conference*, pp. 3585–3590, 2014.

[61] I. Palunko, A. Faust, P. Cruz, L. Tapia, and R. Fierro, "A reinforcement learning approach towards autonomous suspended load manipulation using aerial robots," in *2013 IEEE International Conference on Robotics and Automation*, pp. 4896–4901, 2013.

[62] A. K. Shastry, A. Pattanaik, and M. Kothari, "Neuro-adaptive augmented dynamic inversion controller for quadrotors," *IFAC-PapersOnLine*, vol. 49, no. 1, pp. 302–307, 2016. 4th IFAC Conference on Advances in Control and Optimization of Dynamical Systems ACODS 2016.

[63] A. Faust, I. Palunko, P. Cruz, R. Fierro, and L. Tapia, "Learning swing-free trajectories for uavs with a suspended load," in *2013 IEEE International Conference on Robotics and Automation*, pp. 4902–4909, 2013.

[64] A. Martinez-Vasquez, R. Castro-Linares, and A. Rodriguez-Mata, "Sliding mode control of a quadrotor with suspended payload: a differential flatness approach," in *2020 17th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, pp. 1–6, 2020.

[65] Y. Kui, G. Feng, Y. Liying, H. Yuqing, and H. Jian-da, "Sliding mode control for a quadrotor slung load system," *2017 36th Chinese Control Conference (CCC)*, pp. 3697–3703, 2017.

[66] A. Chandra and P. P. Lal, "Higher order sliding mode controller for a quadrotor uav with a suspended load," *International Federation of Automatic Control*, vol. 55, no. 1, pp. 610–615, 2022.

[67] Bingöl and H. M. Güzey, "Finite-time neuro-sliding-mode controller design for quadrotor uavs carrying suspended payload," *Drones*, vol. 6, no. 10, 2022.

[68] Ö. Bingöl and H. M. Guzey, "Sliding mode control for a quadrotor uav transporting a cable-suspended payload," *Proceedings of 14th International Conference on Electromechanics and Robotics "Zavalishin's Readings"*, 2019.

[69] A. Mosco-Luciano, R. Castro-Linares, and H. Rodríguez-Cortés, "Trajectory tracking control for a quadrotor with a slung load," *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 322–328, 2020.

[70] J. Slotine and W. Li, *Applied Nonlinear Control*. Prentice Hall, 1991.

[71] V. Utkin and H. Lee, "Chattering problem in sliding mode control systems," in *International Workshop on Variable Structure Systems, 2006. VSS'06.*, pp. 346–350, 2006.

[72] L. ARIE, "Sliding order and sliding accuracy in sliding mode control," *International Journal of Control*, vol. 58, no. 6, pp. 1247–1263, 1993.

[73] Y. Shtessel, M. Taleb, and F. Plestan, "A novel adaptive-gain supertwisting sliding mode controller: Methodology and application," *Automatica*, vol. 48, no. 5, pp. 759–769, 2012.

[74] S. Emel'Yanov, S. Korovin, and A. Levant, "High-order sliding modes in control systems," *Computational mathematics and modeling*, vol. 7, no. 3, pp. 294–318, 1996.

[75] C. Edwards and Y. B. Shtessel, "Adaptive continuous higher order sliding mode control," *Automatica*, vol. 65, pp. 183–190, 2016.

[76] M. Dulău, A. Gligor, and T.-M. Dulău, "Fractional order controllers versus integer order controllers," in *10th International Conference Interdisciplinarity in Engineering*, vol. 181, pp. 538–545, 2017.

[77] V. S. Ertürk, Z. M. Odibat, and S. Momani, "An approximate solution of a fractional order differential equation model of human t-cell lymphotropic virus i (htlv-i) infection of cd4+ t-cells," *Computers Mathematics with Applications*, vol. 62, no. 3, pp. 996–1002, 2011. Special Issue on Advances in Fractional Differential Equations II.

[78] V. Gafiychuk, B. Datsko, and V. Meleshko, "Mathematical modeling of time fractional reaction–diffusion systems," *Journal of Computational and Applied Mathematics*, vol. 220, no. 1, pp. 215–225, 2008.

[79] M. A. Ezzat, "Theory of fractional order in generalized thermoelectric mhd," *Applied Mathematical Modelling*, vol. 35, no. 10, pp. 4965–4978, 2011.

[80] S. Müller, M. Kästner, J. Brummund, and V. Ulbricht, "A nonlinear fractional viscoelastic material model for polymers," *Computational Materials Science*, vol. 50, no. 10, pp. 2938–2949, 2011.

[81] F. Jiahe and L. Rui, "Fractional pid and backstepping control for a small quadrotor helicopter," in *2015 34th Chinese Control Conference (CCC)*, pp. 5701–5706, 2015.

[82] M. Efe, "Fractional fuzzy adaptive sliding-mode control of a 2-dof direct-drive robot arm," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 6, pp. 1561–1570, 2008.

[83] M. Efe, "Battery power loss compensated fractional order sliding mode control of a quadrotor uav," *Asian Journal of Control*, vol. 14, no. 2, pp. 413–425, 2011.

[84] H. Chao, Y. Luo, L. Di, and Y. Chen, "Roll-channel fractional order controller design for a small fixed-wing unmanned aerial vehicle," *Control Engineering Practice*, vol. 18, pp. 761–772, 2010.

[85] R. Cajo, T. T. Mac, D. Plaza, C. Copot, R. De Keyser, and C. Ionescu, "A survey on fractional order control techniques for unmanned aerial and ground vehicles," *IEEE Access*, vol. 7, pp. 66864–66878, 2019.

[86] M. Vahdanipour and M. Khodabandeh, "Adaptive fractional order sliding mode control for a quadrotor with a varying load," *Aerospace Science and Technology*, vol. 86, pp. 737–747, 2019.

[87] D. Maiti, S. Biswas, and A. Konar, "Design of a fractional order PID controller using particle swarm optimization technique," *CoRR*, 2008.

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

[89] "Vlp-16 user manual," Rev.D, Velodyne LiDAR Inc. [Online]. Available: https://greenvalleyintl.com/wp-content/uploads/2019/02/Velodyne-LiDAR-VLP-16-User-Manual.pdf. [Accessed 28 April 2021].

[90] G. Welch and G. Bishop, *An Introduction to the Kalman Filter*. USA: University of North Carolina at Chapel Hill, 2001.

[91] A. Gelb, *Applied optimal estimation.* M.I.T. Press, 1974.

[92] C. Van Loan, "Computing integrals involving the matrix exponential," *IEEE Transactions on Automatic Control*, vol. 23, no. 3, pp. 395–404, 1978.

[93] F. De Vivo, A. Brandl, M. Battipede, and P. Gili, "Joseph covariance formula adaptation to square-root sigma-point kalman filters," *Nonlinear dynamics*, vol. 88, no. 3, pp. 1969–1986, 2017.

[94] *Lock Studio and Lock Lab: Lock Sync Box*. Vicon, [Online]. Available: https://www.vicon.com/hardware/devices/lock/ [Accessed 29 January 2023].

[95] M. Driedger and P. Ferguson, "Feasibility study of an orbital navigation filter using resident space object observations," *Journal of Guidance, Control, and Dynamics*, vol. 44, no. 3, pp. 622–628, 2021.

[96] T. W. Anderson and D. A. Darling, "Asymptotic theory of certain "goodness of fit" criteria based on stochastic processes," *The Annals of mathematical statistics*, vol. 23, no. 2, pp. 193–212, 1952.

[97] F. J. Massey, "The kolmogorov-smirnov test for goodness of fit," *Journal of the American Statistical Association*, vol. 46, no. 253, pp. 68–78, 1951.

[98] L. H. Miller, "Table of percentage points of kolmogorov statistics," *Journal of the American Statistical Association*, vol. 51, no. 273, pp. 111–121, 1956.

[99] W. W. Tsang, G. Marsaglia, and J. Wang, "Evaluating kolmogorov's distribution," *Journal of statistical software*, vol. 8, no. 18, 2003.

[100] *QDrone*. Quanser, [Online]. Available: https://www.quanser.com/products/qdrone/ [Accessed 10 May 2021].

[101] D. Vrančić, J. Petrovčič, and Y. Peng, "A new simple auto-tuning method for pid controllers," *IFAC Proceedings Volumes*, vol. 30, no. 21, pp. 463–468, 1997. 2nd IFAC Workshop on New Trends in Design of Control Systems 1997, Smolenice, Slovak Republic, 7-10 September.

[102] L. Fan and E. M. Joo, "Design for auto-tuning pid controller based on genetic algorithms," in *2009 4th IEEE Conference on Industrial Electronics and Applications*, pp. 1924–1928, 2009.

[103] R. Zhai, Z. Zhou, W. Zhang, S. Sang, and P. Li, "Control and navigation system for a fixed-wing unmanned aerial vehicle," *AIP Advances*, vol. 4, no. 3, p. 031306, 2014.

[104] I. Podlubny, "Geometric and physical interpretation of fractional integration and fractional differentiation," *Fractional Calculus and Applied Analysis*, vol. 5, June 2004.

[105] A. Atangana, "Chapter 5 - fractional operators and their applications," in *Fractional Operators with Constant and Variable Order with Application to Geo-Hydrology*, pp. 79–112, Academic Press, 2018.

[106] I. Podlubny, *Fractional Differential Equations: An Introduction to Fractional Derivatives, Fractional Differential Equations, to Methods of Their Solution and Some of Their Applications*. Mathematics in Science and Engineering, Elsevier Science, 1998.

[107] I. Podlubny, "Fractional-order systems and pi/sup /spl lambda//d/sup /spl mu//-controllers," *IEEE Transactions on Automatic Control*, vol. 44, no. 1, pp. 208–214, 1999.

[108] M.-S. Abdelouahab and N.-E. Hamri, "The grünwald–letnikov fractional-order derivative with fixed memory length," *Mediterranean Journal of Mathematics*, vol. 13, no. 2, pp. 557–572, 2016.

[109] B. Vinagre, I. Podlubny, A. Hernandez, and V. Feliu, "Some approximations of fractional order operators used in control theory and applications," *Fractional calculus and applied analysis*, vol. 3, no. 3, pp. 231–248, 2000.

[110] Z. Li, L. Liu, S. Dehghan, Y. Chen, and D. Xue, "A review and evaluation of numerical tools for fractional calculus and fractional order controls," *International Journal of Control*, vol. 90, no. 6, pp. 1165–1181, 2017.

[111] F. N. Deniz, B. B. Alagoz, N. Tan, and M. Koseoglu, "Revisiting four approximation methods for fractional order transfer function implementations: Stability preservation, time and frequency response matching analyses," *Annual Reviews in Control*, vol. 49, pp. 239–257, 2020.

[112] P. Shah and S. Agashe, "Review of fractional pid controller," *Mechatronics*, vol. 38, pp. 29–41, 2016.

[113] I. Podlubny, I. Petráš, B. Vinagre, P. O'Leary, and L. Dorčák, "Analogue realizations of fractional-order controllers," *Nonlinear dynamics*, vol. 29, no. 1-4, pp. 281–296, 2002.

[114] D. Xue, Y. Chen, and D. Atherton, *Linear Feedback Control: Analysis and Design with MATLAB*. Advances in Design and Control, Society for Industrial and Applied Mathematics, 2007.

[115] S. Das, I. Pan, S. Saha, A. Kumar, S. Das, and A. Gupta, "Revisiting oustaloup's recursive filter for analog realization of fractional order differintegrators," in *2011 International Conference on Energy, Automation and Signal*, pp. 1–6, 2011.

[116] S. B. Chiranjeevi *et al.*, "Implementation of fractional order pid controller for an avr system using ga and aco optimization techniques," *IFAC-PapersOnLine*, vol. 49, no. 1, pp. 456–461, 2016.

[117] M. R. Faieghi and A. Nemati, "On fractional-order pid design," in *Applications of MATLAB in Science and Engineering* (T. Micha322;owski, ed.), ch. 13, Rijeka: IntechOpen, 2011.

[118] "Physical system parameters, drone parametrization," *Quanser QDrone*, 13 July 2022. [Online]. Available: https://www.quanser.com/courseware-resources/?fwpresourcerelatedproducts=6269.

[119] D. Wang, D. Tan, and L. Liu, "Particle swarm optimization algorithm: an overview," *Soft computing (Berlin, Germany)*, vol. 22, pp. 387–408, 2018.

[120] R. Hassan, B. Cohanim, O. de Weck, and G. Venter, "A comparison of particle swarm optimization and the genetic algorithm," in *46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, 2005.

[121] E. K. P. Chong and S. H. Żak, "Gradient methods," in *An Introduction to Optimization*, ch. 8, pp. 125–153, John Wiley Sons, Ltd, 2008.

[122] G. Ellis, "Chapter 6 - four types of controllers," in *Control System Design Guide*, pp. 97–119, Butterworth-Heinemann, fourth ed., 2012.

[123] P. Homolka, M. Hromčík, and T. Vyhlídal, "Input shaping solutions for drones with suspended load: First results," in *2017 21st International Conference on Process Control (PC)*, pp. 30–35, 2017.

[124] S. Ichikawa, A. Castro, N. Johnson, H. Kojima, and W. Singhose, "Dynamics and command shaping control of quadcopters carrying suspended loads," *IFAC-*

PapersOnLine, vol. 51, no. 14, pp. 84–88, 2018. 14th IFAC Workshop on Time Delay Systems TDS 2018.

[125] Y. Guo, S.-M. Song, and X.-H. Li, "Terminal sliding mode control for attitude tracking of spacecraft based on rotation matrix," *Mathematical Problems in Engineering*, vol. 2015, pp. 1–9, March 2015.

Appendix A

Copyright Permission

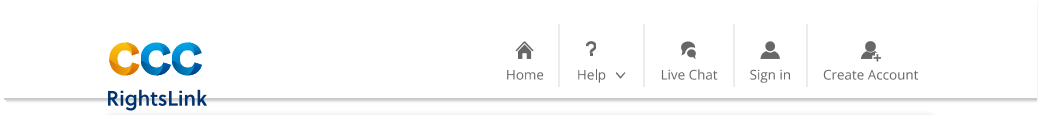
This appendix provides copyright permissions to re-use the content from previously published papers.

A.0.1 Copyright from IEEE

In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of the University of Manitoba's products or services. Internal or personal use of this material is permitted.

The following page summarizes the permission from IEEE to re-use the content from a previously published paper titled "Tracking and Estimation of a Swaying Payload Using a LiDAR and an Extended Kalman Filter"

APPENDIX A. COPYRIGHT PERMISSION



IEEE Requesting permission to reuse content from an IEEE publication

Tracking and Estimation of a Swaying Payload Using a LiDAR and an Extended Kalman Filter

Conference Proceedings:
2021 IEEE International Symposium on Robotic and Sensors Environments (ROSE)

Author: Mitesh Patel

Publisher: IEEE

Date: 28 October 2021

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

A.0.2 Copyright from AIAA

Re: Requesting Information On Reusing Contents Of AIAA SciTech 2022 Conference Paper in Thesis Publication

Mitesh Patel <patelm14@myumanitoba.ca>
Wed 8/24/2022 11:01 AM
To: Katrina Buckley <KatB@aiaa.org>

Hi Katrina,

Thank you for the quick response and clarifying on my question. I will follow your guidelines when reusing postings of the paper.

I appreciate your help. Thanks again!

Mitesh Patel

From: Katrina Buckley <KatB@aiaa.org>
Sent: Wednesday, August 24, 2022 10:57:34 AM
To: Ann Ames <AnnA@aiaa.org>; Mitesh Sunilkumar Patel <patelm14@myumanitoba.ca>
Subject: RE: Requesting Information On Reusing Contents Of AIAA SciTech 2022 Conference Paper in Thesis Publication

Caution: This message was sent from outside the University of Manitoba.

Hi Mitesh,
Thank you for your inquiry. It looks like you and your co-author retained copyright of this paper:

https://arc.aiaa.org/doi/10.2514/6.2022-2290
Copyright © 2022 by Mitesh Patel and Philip Ferguson.

So, you do not need AIAA's permission to reuse portions of it in your thesis; you can do so as you wish. That said, as a general rule, you will want to acknowledge within the main text or a footnote that relevant sections/chapters/figures from your thesis are reprinted from another source (e.g., "From [paper title and authors]). Note that the original source should be cited in full in the reference list.

Best,
Kat

Katrina Buckley
Managing Editor, Books

American Institute of Aeronautics and Astronautics www.aiaa.org
12700 Sunrise Valley Drive, Suite 200
Reston, VA 20191-5807
800-639-AIAA (2422)
katb@aiaa.org 703.264.7566 (direct)

From: Ann Ames <AnnA@aiaa.org>
Sent: Wednesday, August 24, 2022 11:01 AM
To: Mitesh Sunilkumar Patel <patelm14@myumanitoba.ca>
Cc: Katrina Buckley <KatB@aiaa.org>
Subject: RE: Requesting Information On Reusing Contents Of AIAA SciTech 2022 Conference Paper in Thesis Publication

Katrina Buckley, copied, will assist you.

Ann Ames
Manager, Technical Program

American Institute of Aeronautics and Astronautics www.aiaa.org
12700 Sunrise Valley Drive, Suite 200
Reston, VA 20191-5807
800-639-AIAA (2422)
AnnA@aiaa.org 703-264-7549 (direct)

From: Mitesh Sunilkumar Patel <patelm14@myumanitoba.ca>
Sent: Tuesday, August 23, 2022 4:15 PM
To: Ann Ames <AnnA@aiaa.org>
Subject: Requesting Information On Reusing Contents Of AIAA SciTech 2022 Conference Paper in Thesis Publication

Hello Ann,

I was an author and presenter at the AIAA SciTech Forum 2022 in San Diego. I am about to graduate and would like to use parts of my publication from the 2022 SciTech forum in my thesis. I have researched online how to go about getting permission to do so but to no avail.

I would appreciate it if you could provide me with any information on this matter or provide me with contact information for someone who may be able to help me.

Thank you,
Mitesh Patel

Appendix B

Solving Lagrange Dynamics Using Matlab's Symbolic Toolbox

This appendix provides the Matlab code used to solve the equations of motion for the 3D pendulum model and the drone-based cable suspended model.

B.0.1 3D Pendulum Model

```
clear all; close all; clc;
% Create variables using syms
syms x(t) y(t) z(t) L M m g D_x D_y
% Position in inertial frame
x_p = x
y_p = y
z_p = sqrt(L^2 - x_p^2 - y_p^2)
% Velocity in inertial frame
x_dot_p = diff(x_p)
y_dot_p = diff(y_p)
```

```
z_dot_p = diff(z_p)
% Kinetic energy, potential energy, and Lagrangian
T = 1/2*m*(x_dot_p^2 + y_dot_p^2 + z_dot_p^2);
U = -m*g*z_p;
Lag = T - U;

% Solve Lagrangian for xi = x-payload
dLdx = functionalDerivative(Lag,x)==0;
% Solve Lagrangian for xi = y-payload
dLdy = functionalDerivative(Lag,y)==0;

% Substitute D_x and D_y for x and y accelerations
dLdx = subs(dLdx, [diff(diff(x)), diff(diff(y))], [D_x, D_y]);
dLdy = subs(dLdy, [diff(diff(x)), diff(diff(y))], [D_x, D_y]);

xddd = isolate(dLdx,D_x);
yddd = isolate(dLdy,D_y);

% Solve 2 equations for 2 variables (equations of motion)
eq_sol = solve([xddd, yddd], [D_x, D_y]);
```

B.0.2 Drone-Based Cable-Suspended Model

```
clear all; close all; clc;
% Create variables using syms
syms x_d(t) y_d(t) z_d(t) x_p(t) y_p(t) z_p(t) L b Md m g F_x F_y ...
    F_z D_x D_y D_z P_x P_y F_xp F_yp
% Drone Position in inertial
x_d = x_d;
y_d = y_d;
```

```
z_d = z_d;
% Drone Velocity in inertial
xd_dot = diff(x_d);
yd_dot = diff(y_d);
zd_dot = diff(z_d);
% Payload Position in inertial
x_p = x_p;
y_p = y_p;
z_p = z_d + sqrt(L^2 - (x_p-x_d)^2 - (y_p-y_d)^2)
% Payload Velocity in inertial
x_dot_p = diff(x_p);
y_dot_p = diff(y_p);
z_dot_p = diff(z_p)

% Total Kinetic Energy
Td = 1/2*Md*(xd_dot^2 + yd_dot^2 + zd_dot^2);
Tp = 1/2*m*(x_dot_p^2 + y_dot_p^2 + z_dot_p^2);
T = Td + Tp;
% Total Potential Energy
Ud = -Md*g*z_d;
Up = -m*g*z_p;
U = Ud + Up;
% Lagrangian
Lag = T - U;
% Reigls Dissipation Energy
R = 1/2*b*((xd_dot - x_dot_p)^2 + (yd_dot - y_dot_p)^2 + (zd_dot ...
    - z_dot_p)^2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%    Solve Lagrangian for Drone Variables   %%%%%%%%%%%
% For xi = xd
dLdxd = functionalDerivative(Lag,x_d) + diff(R, xd_dot) == -F_x;
% For xi = yd
```

```
dLdyd = functionalDerivative(Lag,y_d) + diff(R, yd_dot) == -F_y;
% For xi = zd
dLdzd = functionalDerivative(Lag,z_d) + diff(R, zd_dot) == -F_z;
%%%%%%%%%%%  Solve Lagrangian for Payload Variables   %%%%%%%%%%%
% For xi = xp
dLdxp = functionalDerivative(Lag,x_p) + diff(R, x_dot_p) == -F_xp;
% For xi = yp
dLdyp = functionalDerivative(Lag,y_p) + diff(R, y_dot_p) == -F_yp;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Substitute D_x, D_y, D_z, P_x, P_y for x, y, z drone ...
    accelerations, and x and y payload acceleration
dLdxd = subs(dLdxd, [diff(xd_dot), diff(yd_dot), diff(zd_dot), ...
    diff(diff(x_p)), diff(diff(y_p))], [D_x, D_y, D_z, P_x, P_y]);
dLdyd = subs(dLdyd, [diff(xd_dot), diff(yd_dot), diff(zd_dot), ...
    diff(diff(x_p)), diff(diff(y_p))], [D_x, D_y, D_z, P_x, P_y]);
dLdzd = subs(dLdzd, [diff(xd_dot), diff(yd_dot), diff(zd_dot), ...
    diff(diff(x_p)), diff(diff(y_p))], [D_x, D_y, D_z, P_x, P_y]);
dLdxp = subs(dLdxp, [diff(xd_dot), diff(yd_dot), diff(zd_dot), ...
    diff(diff(x_p)), diff(diff(y_p))], [D_x, D_y, D_z, P_x, P_y]);
dLdyp = subs(dLdyp, [diff(xd_dot), diff(yd_dot), diff(zd_dot), ...
    diff(diff(x_p)), diff(diff(y_p))], [D_x, D_y, D_z, P_x, P_y]);

xddd = isolate(dLdxd,D_x);
yddd = isolate(dLdyd,D_y);
zddd = isolate(dLdzd,D_z);
xpdd = isolate(dLdxp,P_x);
ypdd = isolate(dLdyp,P_y);

% Solve 5 equations for 5 variables (equations of motion)
eq_sol = solve([xddd, yddd, zddd, xpdd, ypdd], [D_x, D_y, D_z, ...
    P_x, P_y])
```