DESIGN OF A LINEAR PARAMETER VARYING CONTROL SYSTEM FOR A DELIVERY QUADROTOR

A Project

Presented to

The Faculty of the Department of Aerospace Engineering

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Hussam Okasha

May 2020

© 2020

Hussam Okasha

ALL RIGHTS RESERVED

The Designated Project Advisor(s) Approves the Project Titled

DESIGN OF A LINEAR PARAMETER VARYING CONTROL SYSTEM FOR A DELIVERY QUADROTOR

by

Hussam Okasha

APPROVED FOR THE DEPARTMENT OF AEROSPACE ENGINEERING

SAN JOSÉ STATE UNIVERSITY

May 2020

Dr. Sean Swei	NASA Ames Research Center	Advisor

ABSTRACT

DESIGN OF A LINEAR PARAMETER VARYING CONTROL SYSTEM FOR A DELIVERY QUADROTOR

by Hussam Okasha

Developing flight control systems for quadrotors capable of grasping, carrying, and dropping payloads is an active research area. Applications include package delivery, post-disaster relief and rescue, and firefighting. The purpose of this project is to propose a suitable controller for a quadrotor capable of delivery of small packages up to 2.3 kg. The action of picking up or dropping off payloads can significantly affect the dynamic response of a quadrotor, possibly preventing the successful completion of a mission. Furthermore, during flight the battery voltage decreases leading to varying propeller speeds with losses in the control effectiveness of thrust and torque factors of the propellers. A linear parameter varying (LPV) control solution is proposed with a focus on the implementation of its adaptive structure and investigating its stability, performance, and robustness in controlling the quadrotor and counteracting adverse effects. The quadrotor is modeled as an LPV system and the LPV controller is designed utilizing an $\mathcal{H}_\infty$ self-scheduling technique in which the payload mass is treated as a scheduling parameter. To estimate the mass online, an adaptive estimator based on the gradient descent method is developed. The controller gains are updated automatically based on the convex constructions of fixed controllers at the vertices of a parameter box. These controllers are determined by solving a system of linear matrix inequalities (LMIs) which synthesize gain-scheduled $\mathcal{H}_\infty$ controllers that act within a bounded parameter space. A two-degrees-of-freedom control structure, with reference and error signals fed into the LPV controller, is developed to counteract system variations while providing tracking control. Finally, the LPV control system with added modifications is tested against the nonlinear system to validate the control algorithm meets requirements by picking up an unknown payload and tracking a reference trajectory subject to actuator dynamics and disturbances.

ACKNOWLEDGEMENTS

I would like to thank Dr. Swei for his guidance on this project and introducing me to the topic of LPV control during AE 173 and other advanced control methods in AE 246. His teaching of control theory from a researcher’s point of view with a focus on the fundamentals while addressing practical issues were so helpful for a student of controls. I also thank Professor Lu for his excellent teaching of AE 245 which improved my skills in MATLAB/Simulink and controls design. I kept their lessons in mind when working on the project. I would also like to thank my brother Samer Okasha for sharing his adaptive control course notes which were indispensable for developing the mass estimator and for our discussions on how to properly add modifications to the system for control of the nonlinear system. These modifications resolved stability issues with the nonlinear simulations. Finally, my gratitude to my family for supporting my decision to pursue a MS degree after nearly a decade removed from my undergraduate education.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

K_F	=	thrust factor of the propeller
K_M	=	drag factor of the propeller
g	=	acceleration due to gravity
J_x	=	moment of inertia along the x direction
J_y	=	moment of inertia along the y direction
J_z	=	moment of inertia along the z direction
l	=	distance from center of quadrotor to center of each propeller
m	=	total mass of the quadrotor
m_p	=	mass of the package payload
p	=	roll rate
q	=	pitch rate
r	=	yaw rate
F_z	=	lift thrust factor in z direction
τ_θ	=	pitching torque factor in θ direction
τ_φ	=	rolling torque factor in φ direction
τ_ψ	=	yawing torque factor in ψ direction
u	=	velocity in the x-axis direction
v	=	velocity in the y-axis direction
w	=	velocity in the z-axis direction
X	=	x inertial coordinate of quadrotor at the center of mass
Y	=	y inertial coordinate of quadrotor at the center of mass
Z	=	z inertial coordinate of quadrotor at the center of mass
θ	=	pitch angle
φ	=	roll angle
ψ	=	yaw angle
Ω_f	=	front propeller speed
Ω_r	=	right propeller speed
Ω_b	=	rear propeller speed
Ω_l	=	left propeller speed
c_m	=	motor constant
τ	=	motor time constant

ω_n	=	natural frequency
ζ	=	damping ratio
$\boldsymbol{u}$	=	control input vector
$\boldsymbol{w_d}$	=	disturbance input vector
$\boldsymbol{e}$	=	error between reference set point and real output of plant
$\boldsymbol{r}$	=	reference set point
$\boldsymbol{K}$	=	control gains
$\boldsymbol{R}, \boldsymbol{S}, \boldsymbol{X}$	=	LPV synthesis separation parameters
$\boldsymbol{P}$	=	positive definite matrix
$\boldsymbol{\rho}$	=	scheduling parameter vector
$\boldsymbol{x}$	=	state vector
$\boldsymbol{y}$	=	measurement vector
$\boldsymbol{z}$	=	performance output vector
k_r	=	reference input scaling factor
k_u	=	control input scaling factor
G	=	plant model
W_d	=	disturbance weight
W_p	=	performance or sensitivity weight
W_r	=	setpoint weight
W_u	=	control or robustness weight
γ	=	closed-loop $\mathcal{H}_\infty$ norm
λ	=	rate of convergence constant
τ_{ref}	=	model reference signal time constant
α	=	convex coordinate
Ω	=	polytope
$\overline{\sigma}$	=	maximum singular value
$\underline{\sigma}$	=	minimum singular value
K_{Π}	=	vertex controller
A_K, B_K, C_K, D_K	=	LPV controller matrices
$\mathcal{A}, \mathfrak{B}, \mathcal{C}, \mathcal{D}$	=	closed-loop state space matrices
$(\cdot)_B$	=	expressed in body frame
$(\cdot)_E$	=	expressed in inertial frame

1. CHAPTER 1

INTRODUCTION

1.1 Motivation

A quadrotor is an aerospace vehicle with four propellers in a cross configuration. The front and rear rotors rotate counterclockwise while the left and right rotors rotate clockwise, as shown in Figure 1.1 with a quadrotor in hovering condition where all four propeller speeds have the same magnitude. Adjacent propellers counter rotates to remove the need for a tail rotor. The quadrotor's motion can be controlled by changing the speed of the rotors [1].

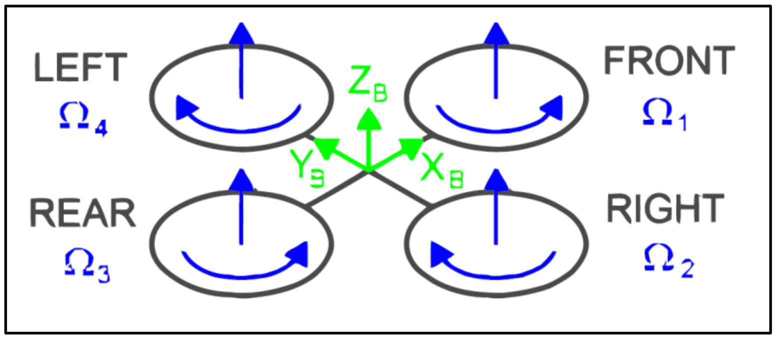


Figure 1.1 - Propeller speeds in hovering condition [1]

Compared to fixed-wing and rotary-wing aircraft, a quadrotor provides engineering advantages such as energy efficiency, a simple mechanical structure, vertical takeoff and landing (VTOL) ability, and lower cost and maintenance requirements. However, there are three main challenges involving quadrotor control: under-actuation, model uncertainty, and actuator failure [2]. Despite its advantages, the physical consequence of under-actuation means a quadrotor cannot follow an arbitrary trajectory due to the limits imposed by the number of system configurations that can be directly controlled [1]. A quadrotor has six degrees of freedom and four independent control inputs which results in two degrees of under-actuation. This means the vehicle can reach a desired set-point in four degrees [2]. Therefore, there are four basic movements that allow a quadrotor to reach a desired altitude and attitude:

- Throttle (F_z)
- Pitch (τ_θ)
- Roll (τ_φ)
- Yaw (τ_ψ)

These control actions are graphically represented in Figure 1.2. Throttle control is achieved by increasing or decreasing all propeller speeds by the same amount. Roll control about x_B is provided by increasing the left propeller speed while decreasing the right one, or the opposite configuration. Similarly, pitch control about y_B is achieved by increasing the rear propeller speed while decreasing the front one. By increasing the paired front-rear propeller speed while decreasing the paired left-right propeller speed, yaw control is achieved about z_B [1].

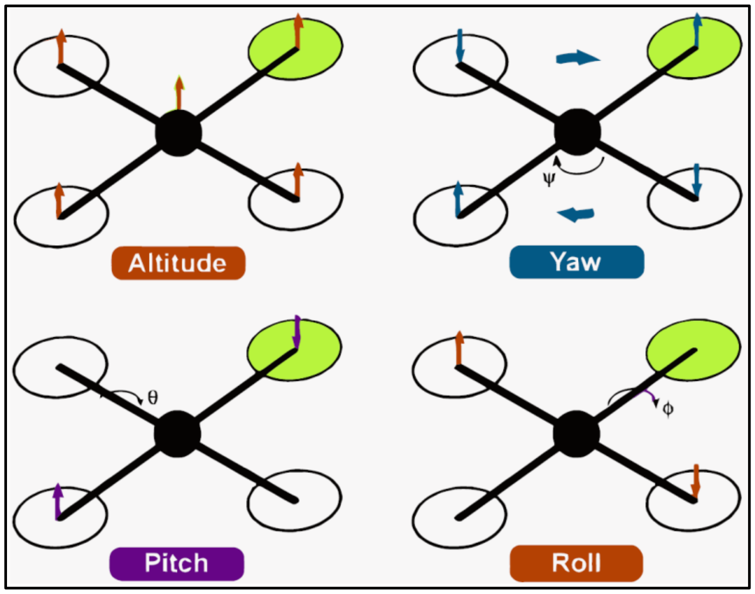


Figure 1.2 - Basic quadrotor control actions [3]

Due to the under-actuation problem, to reach a desired trajectory in all coordinates, tracking control for a quadrotor requires more modern strategies than classical control techniques which were developed for fully actuated systems [2]. Quadrotors also experience uncertainties in the plant and external disturbances during flight. Model uncertainty corresponds to two types: unmodeled plant dynamics caused by high-frequency or nonlinear behavior and parametric uncertainty resulting from physical parameters being inaccurately measured or from variations of these values during operation [4]. Developing flight control systems for underactuated systems and quadrotors capable of grasping, carrying, and dropping payloads is an active research area. The action of picking up or dropping off payloads can significantly affect the dynamic response of the quadrotor [5]. According to [2], a UAV carrying unknown payloads is a famous example of parametric uncertainty. It requires a “nontraditional control” strategy to compensate for the varying mass [6].

Furthermore, during flight the battery voltage decreases leading to varying propeller speeds with losses in the control effectiveness of thrust and torque factors of the propellers [5]. Unexpected failure of an actuator can also lead to a complete loss of an independent control input. This led to research in fault tolerant control (FTC) which combines fault diagnosis detection and a reconfigurable controller [2]. The goal of FTC is to provide "graceful degradation" of the system's performance when a fault is detected, and its adverse effect accurately estimated and compensated for by the controller [5].

Applications for this research include commercial package delivery, medical supply delivery, post-disaster relief and rescue, environmental sampling, and firefighting [5]. For example, in late 2019, Amazon is expected to begin its drone delivery service *Prime Air* in select cities. The service utilizes autonomous, hybrid aircrafts to deliver packages less than 5 pounds within a 15-mile radius of a participating fulfillment center [7]. Amazon promises delivery within 30 minutes of a customer order. Current FAA regulations require drones to fly no higher than 400 ft with a 100-mph maximum speed constraint [8]. Amazon's planned mission requirements are summarized in Table 1.1.

Table 1.1 – Summary of mission requirements for Amazon Prime Air

Altitude Range	**Package Weight**	**Flight Speed**	**Flight Radius**	**Flight Time**
$200\,ft < h < 500\,ft$	$< 5\,lb$	$< 50\,mph$	$< 15\,mi$	$< 30\,min$

UPS Flight Forward announced on October 1st, 2019 they were awarded the first drone airline certificate from the FAA allowing them to fly multiple drones beyond line of sight during deliveries [9]. In March of 2019, they began using quadcopters to deliver blood and medical samples to a North Carolina hospital as part of a test program. The quadcopters are built by California-based *Matternet*. They fly along predetermined flight paths and have a maximum range of 12.5 miles before they need to be recharged [9].

A package delivery UAV is subject to disturbances and uncertainties in its plant dynamics and operating environment that requires an adaptive-robust approach in control [2]. Most quadrotor controllers proposed in the literature focus on a narrow aspect related to tracking or system stability. Furthermore, control systems designed to handle the three main challenges discussed in this section have not been well investigated in the literature. The purpose of this project therefore is to develop a suitable control system for a delivery quadrotor model that can counteract system variations while tracking a desired trajectory and satisfying stability and performance criteria.

1.2 Literature Review

This section is a survey of the literature describing solutions proposed to tackle the three main challenges addressed in *Section 1.1* as it pertains to control of a delivery quadrotor:

- Tracking control subject to under-actuation constraints
- The mass variation and disturbance problem causing instability and performance losses
- The battery drainage problem causing losses in control effectiveness

As an underactuated system with nonlinear dynamics, many control strategies have been proposed in the literature. These include PID control, model reference adaptive control (MRAC), LQR and LQG control, nonlinear dynamic inversion (NDI), model predictive control (MPC), and linear parameter varying (LPV) control.

A common approach for tracking control is to divide the overall quadrotor dynamics into an inner loop and outer loop representing the attitude and position dynamics, respectively [2]. Utilizing a cascade feedback structure for each loop, the overall closed loop system provides attitude and position control. Typically, the inner loop runs at a frequency 5-10 times faster than the outer loop [10]. A sample cascade structure for a quadrotor is shown in Figure 1.3.

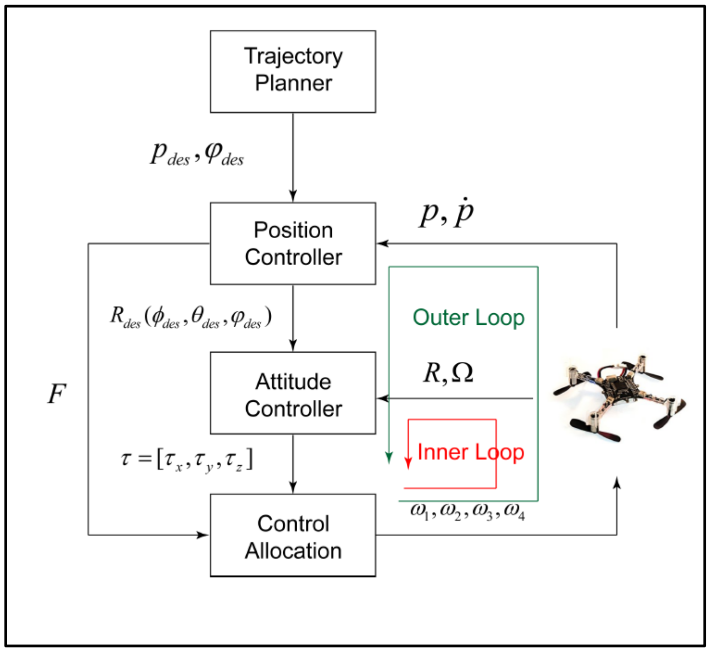


Figure 1.3 - Successive loop closure for a quadrotor [10]

A two-degrees-of-freedom controls structure for a quadrotor is shown in Figure 1.4. Here, the command signals and the feedback signals are independently processed by the controller [11]. For many tracking problems, a one-degree-of-freedom controller may not be sufficient to meet time domain specifications set on the output response [11]. The advantage of this structure is the response to command signals and disturbances are decoupled [10]. This allows the designer flexibility in meeting multiple control objectives by designing feedback and feedforward paths to handle disturbance rejection and provide tracking control.

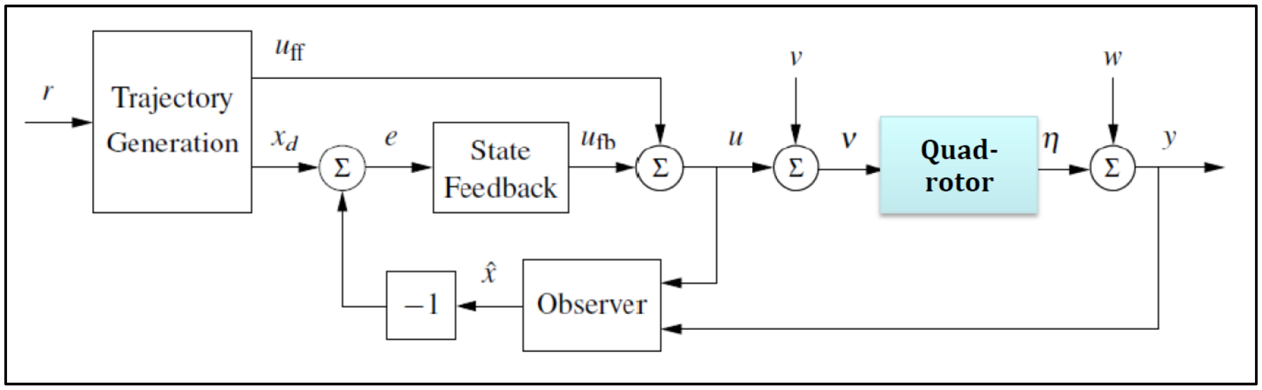


Figure 1.4 - Two-degrees-of-freedom quadrotor control structure [10]

Several strategies have been proposed for the mass variation problem. To demonstrate the inadequacy of a fixed gain PID controller in handling payload changes, the authors in [12] designed an experiment where a 200g payload was added to a quadrotor frame in level flight. The PID controller was not able to compensate for the change in the overall weight and dropped to ground within 3 seconds. The authors then proposed an adaptive control scheme that estimates the system mass in real time which feeds into the control law to adapt to the new system weight. The same test was repeated, and the resultant response is greatly improved. The authors in [13] used gain scheduled PID control and MPC algorithms to control the vertical position of a quadrotor while carrying or dropping a payload. During an experiment with a laboratory quadrotor, a fixed-gain PID controller was not able to eliminate unwanted overshoot at the instant of a payload drop. The control setup was replaced with a gain scheduled PID controller and then an MPC algorithm. Both controllers produced improved system reaction and reduced overshoot of the vertical position. The authors conclude that although the two controllers met some performance criteria, there was no guarantee of stability based on their control structure and suggested applying LPV theory to address the stability question and to improve system performance. An adaptive command-filtered backstepping controller for a quadrotor was developed in [14] to compensate for changes in uncertain parameters of mass, inertia, actuator efficiency, and thruster misalignment. The controller was able to track commands subject to physical constraints and parameter uncertainty.

In [15], a nonlinear adaptive-robust controller (ARC) developed with an LMI-based approach is used to provide attitude and altitude control of a quadrotor subject to disturbances due to wind gusts and uncertain parameters. The quadrotor's mass and moments of inertia were treated as uncertain parameters subject to changes in flight. In an illustrative example to demonstrate the robustness of the controller, the quadrotor follows a reference trajectory subject to wind gusts and delivers a package of unknown weight in midflight. The ARC controller was able to track the desired trajectory in the presence of wind gusts and maintain performance while subject to abrupt changes in the mass dependent parameters of the dynamics model.

1.2.1 Introduction to LPV Systems

Linear parameter varying (LPV) systems are plants that can be described by the state-space model (1.1) where $\boldsymbol{\rho(t)}$ is a vector of time-varying parameters representing the range of possible plant dynamics and the matrices $A(\cdot), B(\cdot), C(\cdot), D(\cdot)$ are fixed functions of those parameters [4]. LPV systems can be interpreted as a model of a more general linear time varying (LTV) system or as a result of linearization of a nonlinear system along the variation of the parameters [16].

$$\begin{aligned} \dot{\boldsymbol{x}}(\boldsymbol{t}) &= \boldsymbol{A}(\boldsymbol{\rho}(\boldsymbol{t}))\boldsymbol{x}(\boldsymbol{t}) + \boldsymbol{B}(\boldsymbol{\rho}(\boldsymbol{t}))\boldsymbol{u}(\boldsymbol{t}) \\ \boldsymbol{y}(\boldsymbol{t}) &= \boldsymbol{C}(\boldsymbol{\rho}(\boldsymbol{t}))\boldsymbol{x}(\boldsymbol{t}) + \boldsymbol{D}(\boldsymbol{\rho}(\boldsymbol{t}))\boldsymbol{u}(\boldsymbol{t}) \end{aligned} \tag{1.1}$$

In many linear robust control problems, a single controller is designed for some defined parametric uncertainty associated with an LPV system. This is considered a conservative approach and can result in poor performance if the system parameters change rapidly or abruptly during operation [4]. Furthermore, a single LTI controller might not even be able to stabilize an LPV system [4]. Another approach to handle model uncertainties is gain scheduling techniques in the field of adaptive control. In gain scheduling, controllers are designed for different equilibrium points within a flight envelope and interpolated in between based on the flight condition using look-up tables. However, stability is not guaranteed in such a setup other than at the design points [17]. LPV control methodologies were developed as an alternative approach to resolve shortcomings of fixed-gain robust controllers and classical gain-scheduling.

In LPV control, the parameters are used as scheduling variables to develop automatically gain-scheduled controllers that update based on weighting functions or convex constructions. The scheduling parameter vector $\boldsymbol{\rho(t)}$ is assumed to be measurable and restricted to a set of admissible trajectories based on operating conditions of the system. In an aerospace context, the parameter $\boldsymbol{\rho}$ can include variables such as airspeed, altitude, mass, center of gravity, or angle of attack. There are several approaches to LPV system representation and controls design. Generally, the state feedback controller may take the form of $\boldsymbol{u} = \boldsymbol{K}[\boldsymbol{\rho}|_{[\boldsymbol{0},\boldsymbol{t}]}]\boldsymbol{x}(\boldsymbol{t})$ where the feedback gain $\boldsymbol{K}$ is a function of the current parameter values or the entire history of parameter measurements [18]. The range of parameter variations is specified by the control designer, but no other *a priori* knowledge is necessary [4].

A specific methodology for LPV controls design is LPV $\mathcal{H}_\infty$ synthesis. It is a robust-adaptive technique that has found applications in air-breathing hypersonic vehicle control [19], control of aeroelastic effects for a flexible wing [20], missile autopilot design [16], control of robotic manipulators [21], and flutter suppression of UAVs [22]. The control law is determined by solving a system of linear matrix inequalities (LMIs) to synthesize the controllers that would act within a bounded parameter space. The methodology for this procedure is described in references [4], [16], and [17]. In such a control strategy, the controller takes advantage of the available parameter information to adjust to the current dynamics of the plant [16].

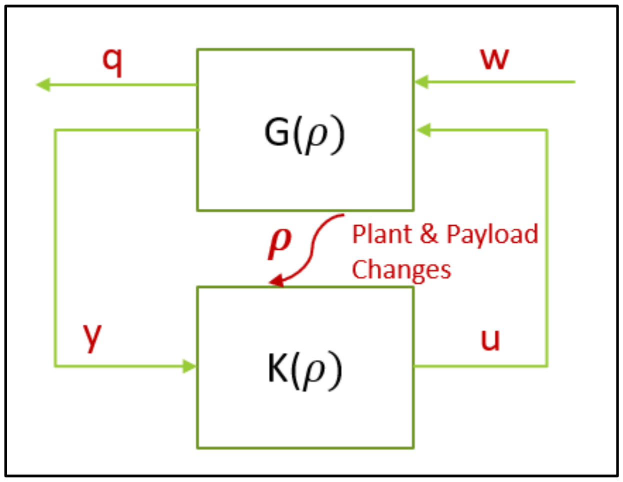


Figure 1.5 - LPV control of an LPV system [16]

LPV control of an LPV system is illustrated in Figure 1.5. The plant $G(\cdot)$ and the controller $K(\cdot)$ are parameter dependent. The exogenous input w includes reference signals and disturbances into the plant. The input u is the control under the designer's authority. The output q is the performance output used to assess the controller and y is the measured output available to the designer to develop the controller K which calculates the control signals u. The LPV structure provides automatic gain scheduling with respect to the measured parameters and the closed-loop system guarantees a prescribed quadratic $\mathcal{H}_\infty$ performance level [16]. Therefore, LPV theory offers an appealing solution to the stability issues of gain scheduling and the performance issues of a fixed-gain robust controller, but it comes at the cost of the higher complexity required to produce the controller.

In recent years, the theory has been applied to control quadrotor UAVs. In reference [5], the authors propose a fault tolerant LPV controller that can compensate for mass variation and battery drainage. The authors in [23] use LPV $\mathcal{H}_\infty$ synthesis to develop a hybrid fault tolerant controller which was robust under fault occurrence. In their approach, the parameter vector was used to schedule between uncertain linear time invariant (LTI) systems, and a reference model was used to generate the desired trajectories to track. An LPV control strategy is proposed in [24] to compensate for a complete actuator loss of a quadrotor by using the yaw rate as a scheduling parameter to produce a controller to align the thrust axis for safe recovery and continuation of flight.

1.3 Proposal

Developing robust, adaptive controllers for quadrotors that experience model variations ensure their safe and efficient operation in the airspace. The goal of this research is to develop a control system for a quadrotor model capable of delivery of small packages in the range of 0.3 kg and 2.3 kg that can compensate for the adverse effects caused by mass variation, actuator effectiveness losses, and disturbance rejection while providing attitude and position tracking. Based on the literature review, it is surmised that LPV control is well suited to provide a solution to the tracking and uncertainty challenges for a delivery quadrotor. Using LPV theory in which the payload mass is used as a scheduling parameter, estimated online using an adaptive estimator, this project proposes a control strategy to achieve the control objectives utilizing a 2DOF LPV controller and a 2DOF PI actuator controller, as shown in Figure 1.6.

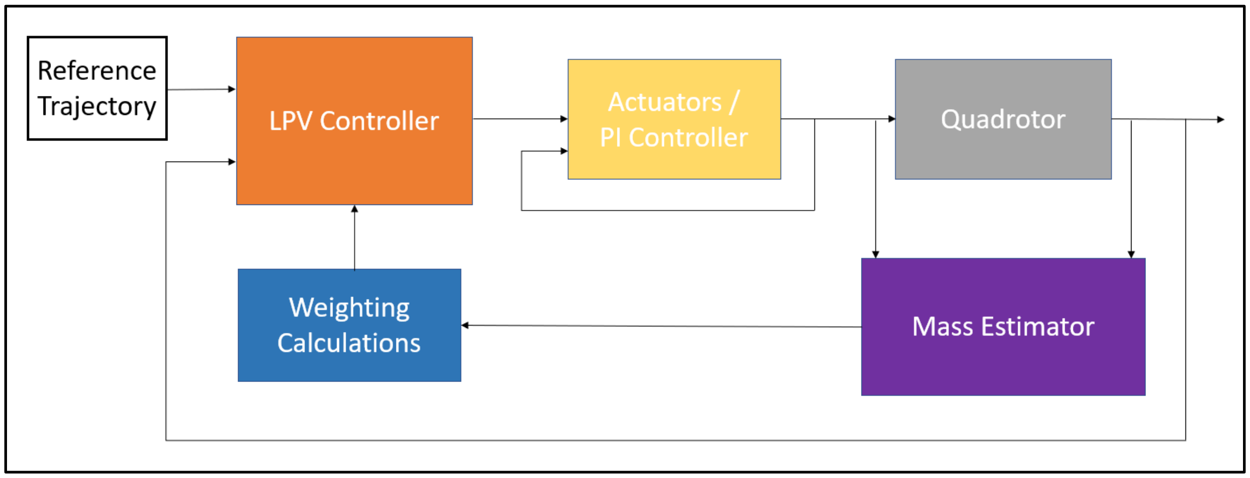


Figure 1.6 - Proposed two-degrees-of-freedom control structure

The automatic gain scheduling policy and LPV controller is obtained through the $\mathcal{H}_\infty$ self-scheduling technique. To demonstrate the stability, performance, and robustness of the proposed LPV control system, nonlinear simulations with actuator dynamics and disturbances applied to the actuator and quadrotor outputs will be performed. To validate the control algorithm meets requirements, several payload masses within the design range are tested.

1.4 Methodology

The research project will be organized into two major parts. The goal of Part I is to develop the control strategy to solve the mass variation problem, disturbance rejection, and provide tracking control. An adaptive estimator is developed to estimate the mass online. For Part II, linear and nonlinear simulations are completed to validate the proposed LPV control strategy meets system requirements A detailed breakdown of the tasks to achieve these goals is presented below.

Part I: LPV Modeling and Control

1.) Specify mission requirements for "last mile" deliveries and design quadrotor's geometry and actuator requirements based on desired flight time and mass range.
2.) Develop nonlinear model representing the equations of motion and actuator dynamics. Linearize about hovering conditions to derive linear, parameter dependent model. Develop a Simulink model of the quadrotor dynamics for 6DOF simulation.
3.) Literature review of LPV theory and $\mathcal{H}_\infty$ self-scheduling techniques.
4.) Develop a mass estimation scheme to determine the payload mass online.
5.) In a MATLAB script, develop the LPV representation of the model.
6.) In a MATLAB script, determine the two-degrees-of-freedom control structure using weighting functions and LMI's with the aid of MATLAB's *Robust Control Toolbox*.
7.) Design PI controller for actuator compensation due to battery drainage and augment to LPV control system.

Part II: Linear and Nonlinear Simulation

1.) Test the LPV controller against the linear model at a set payload mass value and track a reference trajectory. Assess its stability, performance, and robustness.
2.) If satisfactory, test the LPV controller against the nonlinear system without actuator dynamics using the desired forces and torques as the control inputs into the quadrotor plant. Adjust the weights of LPV controller, iterate as necessary, until the desired performance is achieved for a given reference trajectory.
3.) If satisfactory, simulate the control system against the nonlinear system with actuator dynamics using a propeller speed based LPV controller.
4.) Stress the control system by adding disturbance sources on the voltage input, actuator outputs, and velocity states to test the performance and robustness of the LPV controller when tracking the reference trajectory.
5.) Validate the control algorithm meets performance requirements by testing multiple payload masses in the design range.

1.5 Chapters Overview

In Chapter 1, a literature review of the challenges of quadrotor control given its nonlinear, underactuated dynamics and its operation in an uncertain environment is described. The review includes a discussion of parameter variations and its adverse effects on the performance of a control system. Approaches to control a quadrotor with uncertain parameters are described as well. This chapter also includes an introduction to LPV systems.

In Chapter 2, the actuator and rigid body dynamics of the delivery quadrotor are developed with design parameters consistent with mission requirements for a delivery system. The linear, parameter dependent model is also developed. In Chapter 3, an overview of the mathematics required for LPV control theory is presented. The process to develop the LPV controller using the $\mathcal{H}_\infty$ self-scheduling technique and LMIs is also outlined.

In Chapter 4, an adaptive estimator based on the gradient descent method is developed to estimate the payload mass online. The mass estimate is later fed into the LPV controller for automatic gain scheduling. This chapter also includes a hover state conditioning system to control the switching operation of the mass estimator and control commands.

In Chapter 5, the linear parameter dependent model developed in Chapter 2 is extended to build a generalized $\mathcal{H}_\infty$ plant that is structured for gain-scheduled $\mathcal{H}_\infty$ control. In Chapter 6, the LPV controller is developed including a linear simulation to demonstrate the tracking quality of the controller. This chapter also includes a description of the process to build the reference trajectory. In Chapter 7, a 2DOF PI controller is designed to regulate the propeller speeds subject to changes in the input voltage using a first-order motor model.

In Chapter 8, the LPV controller is tested against the nonlinear system subject to actuator dynamics and disturbance sources. A propeller speed based LPV controller is developed to control the nonlinear system with actuator dynamics. A control conditioning subsystem is introduced to compensate for the equilibrium point and the hover state conditioning subsystem is modified to remove control input switching so that the LPV commands are used for the entire operation.

In Chapter 9, the advantages of the LPV controller and its limitations is discussed. Possible solutions to address the limitations are proposed. A qualitative comparison between multivariable and SISO control methods is also discussed. Finally, an assessment of the LPV controller function in an overall GNC system is presented along with future research to improve the performance of the controller and expand the system to include optimal guidance.

2. CHAPTER 2

DELIVERY QUADROTOR MODEL

2.1 Mission Requirements

A typical flight profile for a package delivery drone is shown in Figure 2.1 [25]. The main profile consists of two critical segments for the purposes of this study:

- two cruise segments at level flight
- two hover segments, one with a payload and one without a payload

This profile will be simulated as the reference trajectory to test the efficacy of the proposed controller.

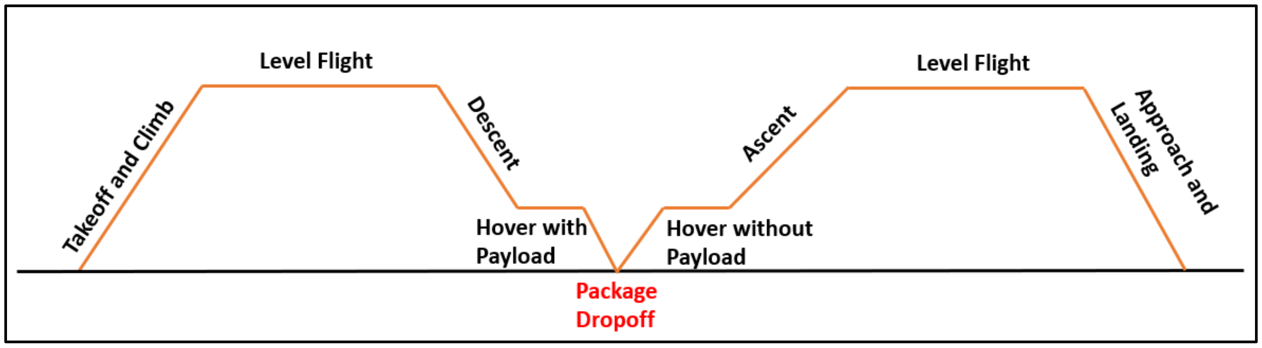


Figure 2.1 - Package delivery mission profile

2.2 Actuator Model

2.2.1 Motor Mixing

The relationship between the thrust and drag factors $[F_z \quad \tau_\phi \quad \tau_\theta \quad \tau_\psi]$ and the motor commands $[\delta_f \quad \delta_r \quad \delta_b \quad \delta_l]$ are determined by (2.1) where k_1 and k_2 are constants that can be determined experimentally [10] and l is the moment arm from the center of mass to the center of the rotor shown in Figure 2.4.

$$\begin{pmatrix} F_z \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{pmatrix} = \begin{pmatrix} k_1 & k_1 & k_1 & k_1 \\ 0 & -lk_1 & 0 & lk_1 \\ lk_1 & 0 & -lk_1 & 0 \\ -k_2 & k_2 & -k_2 & k_2 \end{pmatrix} \begin{pmatrix} \delta_f \\ \delta_r \\ \delta_b \\ \delta_l \end{pmatrix} \tag{2.1}$$

A rotor with an angular speed Ω produced a vertical force $F_i = k_F {\Omega_i}^2$ and a moment $M_i = k_M {\Omega_i}^2$, where k_F and k_M are the rotor drag and motor thrust factors, respectively [10]. Figure 2.2 illustrates the rotor vertical forces and moment reactions rotations for the front, right, back, and left rotors of the quadrotor. Note that the reaction moments are opposite to the direction the propellers are rotating.

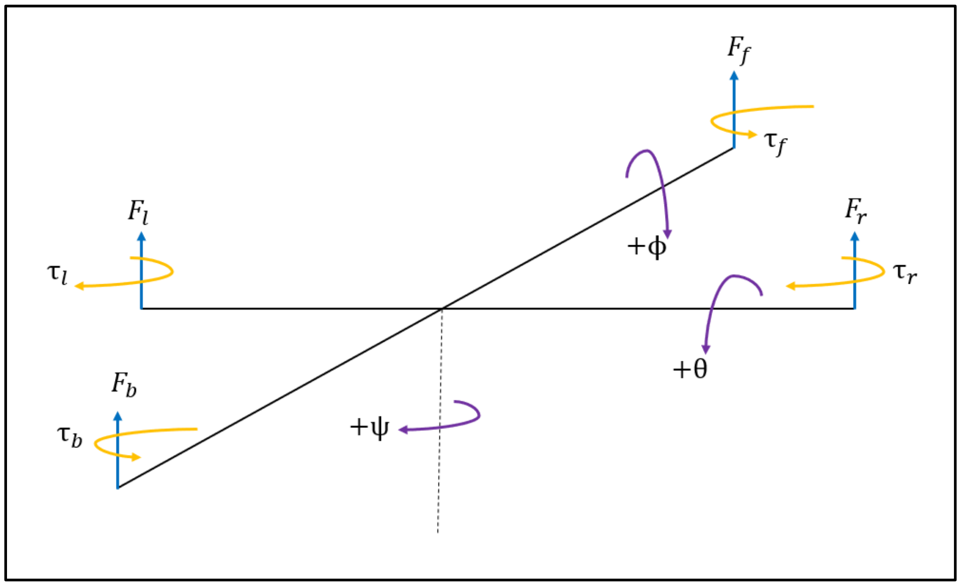


Figure 2.2 – Schematic for rotor and rigid body rotations

The net thrust force F_z and the torque factors τ_ϕ, τ_θ, and τ_ψ are defined in (2.2).

$$\begin{gathered} F_z = F_f + F_r + F_b + F_l \\ \tau_\phi = l(F_l - F_r) \\ \tau_\theta = l\left(F_f - F_b\right) \\ \tau_\psi = \tau_r + \tau_l - \tau_f - \tau_b \end{gathered} \tag{2.2}$$

The equations in (2.3) combined with the results that the thrust and drag factors are related to the square of the propeller speeds can be rewritten in matrix form.

$$\begin{pmatrix} F_z \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{pmatrix} = \begin{pmatrix} k_F & k_F & k_F & k_F \\ 0 & -lk_F & 0 & lk_F \\ lk_F & 0 & -lk_F & 0 \\ -k_M & k_M & -k_M & k_M \end{pmatrix} \begin{pmatrix} {\Omega_f}^2 \\ {\Omega_r}^2 \\ {\Omega_b}^2 \\ {\Omega_l}^2 \end{pmatrix} \tag{2.3}$$

Inverting (2.3) results in the determination of the squared propeller speeds.

$$\begin{pmatrix} {\Omega_f}^2 \\ {\Omega_r}^2 \\ {\Omega_b}^2 \\ {\Omega_l}^2 \end{pmatrix} = \frac{1}{4lk_F k_M} \begin{pmatrix} lk_M & 0 & 2k_M & -lk_F \\ lK_M & -2k_M & 0 & lk_F \\ lk_M & 0 & -2k_M & lk_F \\ lk_M & 2k_M & 0 & lk_F \end{pmatrix} \begin{pmatrix} F_z \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{pmatrix} \tag{2.4}$$

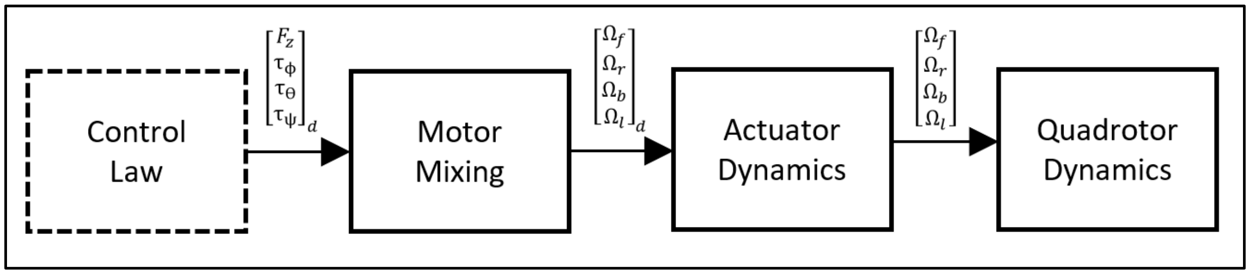
2.2.2 Actuator Dynamics

Actuator dynamics can be described by a first order model [26]

$$\dot{\Omega}_i = c_m(\Omega_{i,d} - \Omega_i), \tag{2.5}$$

where Ω_i are the actual propeller speeds of the four quadrotor motors, $\Omega_{i,d}$ are the desired speeds, and c_m is the motor gain constant taken from reference [26] to be $20\ s^{-1}$.

The block diagram representation of the actuator model is shown in Figure 2.3. The desired thrust and torque factors commanded by the control law are translated to desired propeller speeds by the motor mixing block. However, the motors will not be able to ramp up to this desired angular speed instantly, therefore, the time delay to reach the desired levels is accounted for in the actuator dynamics block. The real speeds $\Omega_f, \Omega_r, \Omega_b, \Omega_l$ are the inputs to the quadrotor dynamics and used to control the quadrotor motion.



Control Law
$[F_z, \tau_\phi, \tau_\theta, \tau_\psi]^T_d$
Motor Mixing
$[\Omega_f, \Omega_r, \Omega_b, \Omega_l]^T_d$
Actuator Dynamics
$[\Omega_f, \Omega_r, \Omega_b, \Omega_l]^T$
Quadrotor Dynamics

Figure 2.3 – Motor Mixing and Actuator Dynamics

2.3 Delivery Quadrotor Parameters

The quadrotors parameters are selected based on the mission requirements for small package delivery. Figure 2.4 is a schematic showing the design parameters to be determined for the delivery quadrotor.

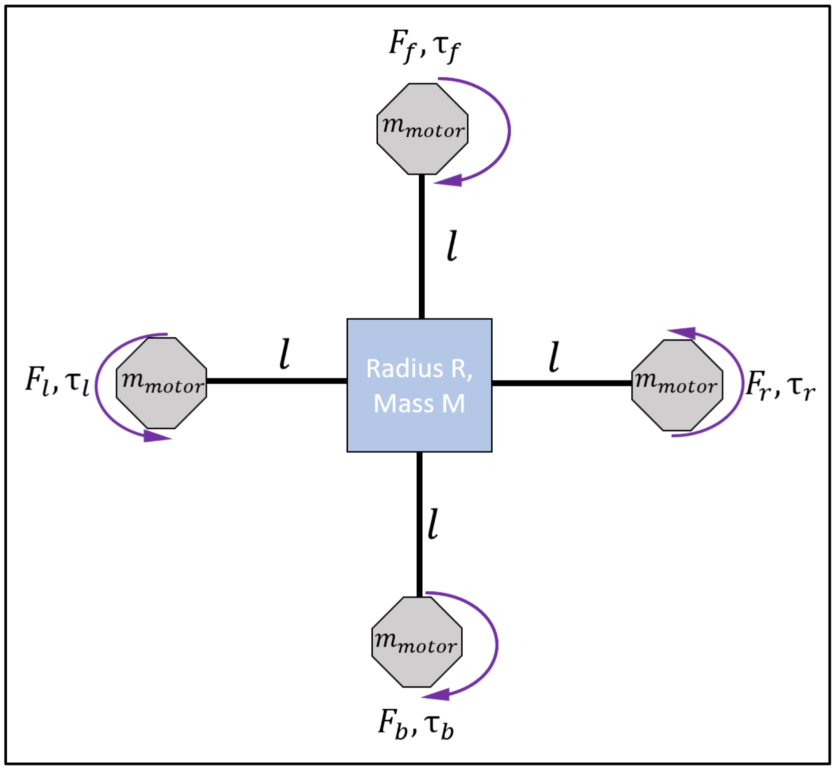


Figure 2.4 – Quadrotor top-view schematic for design parameters

To estimate the moments of inertia, the (2.6) simplification is used to calculate the inertial parameters [10]

$$
\begin{aligned}
J_x &= \frac{2MR^2}{5} + 2l^2 m_{motor} \\
J_y &= \frac{2MR^2}{5} + 2l^2 m_{motor} \\
J_z &= \frac{2MR^2}{5} + 4l^2 m_{motor}
\end{aligned}
\tag{2.6}
$$

where M is defined by (2.7) and R is the estimated radius of the mass center. The parameter m_{quad} is the mass of the quadrotor not including the payload mass, motor mass, and battery mass. It is a design choice, chosen to be 3.8 kg based on similarly sized quadrotors for delivery purposes.

$$
\begin{gathered}
M = m_{quad} + m_{batt} + m_p \\
m_{base} = m_{quad} + 4 * m_{motor} + m_{batt} \\
m = m_{base} + m_p
\end{gathered}
\tag{2.7}
$$

Since J_x, J_y, J_z, m are functions of the package mass m_p, an abrupt change in the payload mass results in a change in the total mass $m = m_{base} + m_p$ which alters these system parameters during flight when a package is picked up or dropped off. As documented in the literature review, this can have adverse effects on the performance of a conventional controller and can lead to instability issues. The design parameters defined in Section 2.2 and Section 2.3 are summarized in Table 2.1. For the purposes of the simulation model, the actuator parameters K_F and K_M are taken from a previous design [27].

Table 2.1 – Summary of design parameters

Parameter	Description	Value	Unit
m_{batt}	battery mass	3.673	kg
m_{motor}	single motor mass	0.325	kg
m_{quad}	base quadrotor mass	3.8	kg
l	length	0.6	m
R	radius of mass center	0.15	m
K_F	rotor drag factor	$4.5x10^{-4}$	$kg \cdot m^2$
K_M	motor thrust factor	$0.45x10^{-5}$	$kg \cdot m$
c_m	motor gain	$0.2x10^2$	$1/s$

2.4 Rigid Body Dynamics of Delivery Quadrotor

The equations of motion for the quadrotor are derived assuming the structure connecting the four rotors is a rigid body and the body axes coincides with the principal axes of rotation so that the moment of inertia tensor

$$\boldsymbol{J} = \begin{pmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{pmatrix} \tag{2.8}$$

is diagonal. The origin of the body axes is set to the center of mass. It is assumed the mass is distributed symmetrically along the x and y axes so that the moments of inertia $J_x = J_y$. The package mass is assumed to be rigidly attached to the base of the quadrotor. The equations for the rigid body dynamics of a quadrotor are taken from references [10], [26] and summarized from (2.9) to (2.12). The equations were developed with the rotation conventions shown in Figure 2.2. A summary of the variables used to describe the motion of the quadrotor is summarized in Table 2.2.

Table 2.2 – Summary of variables for rigid body quadrotor dynamics

Variable	Description
X	inertial north position along $\widehat{I_E}$
Y	inertial east position $\widehat{J_E}$
Z	altitude measured along $-\widehat{K_E}$
u	body frame velocity measured along $\widehat{\iota_B}$
v	body frame velocity measured along $\widehat{\jmath_B}$
w	body frame velocity measured along $\widehat{k_B}$
ϕ	Euler defined roll angle
θ	Euler defined pitch angle
ψ	Euler defined yaw angle
p	roll rate measured along $\widehat{\iota_B}$
q	pitch rate measured along $\widehat{\jmath_B}$
r	yaw rate measured along $\widehat{k_B}$

Applying the motor dynamics summarized in (2.2) and (2.3) yields the control input equations in (2.13).

Table 2.3 – Summary of equations of motion for rigid body quadrotor dynamics

KINEMATIC EQUATIONS – TRANSLATION
$\dot{X} = (cos\psi cos\theta)u + (-sin\psi cos\phi + cos\psi sin\theta sin\phi)v + (sin\psi sin\phi + cos\psi sin\theta cos\phi)w$ (2.9) $\dot{Y} = (sin\psi cos\theta)u + (cos\psi cos\phi + sin\psi sin\theta sin\phi)v + (-cos\psi sin\phi + sin\psi sin\theta cos\phi)w$ $\dot{Z} = (sin\theta)u + (-cos\theta sin\phi)v + (-cos\theta cos\phi)w$
FORCE EQUATIONS
$\dot{u} = (vr - wq) - gsin\theta$ (2.10) $\dot{v} = (wp - ur) + gcos\theta sin\phi$ $\dot{w} = (uq - vp) + gcos\theta cos\phi - \frac{U_z}{m}$
KINEMATIC EQUATIONS – ROTATION
$\dot{\phi} = p + (sin\phi tan\theta)q + (cos\phi tan\theta)r$ (2.11) $\dot{\theta} = (cos\phi)q + (-sin\phi)r$ $\dot{\psi} = (sin\phi/cos\theta)q + (cos\phi/cos\theta)r$
MOMENT EQUATIONS
$\dot{p} = \frac{J_y - J_z}{J_x}qr + \frac{U_\phi}{J_x}$ $\dot{q} = \frac{J_z - J_x}{J_y}pr + \frac{U_\theta}{J_y}$ (2.12) $\dot{r} = \frac{J_x - J_y}{J_z}pq + \frac{U_\psi}{J_z}$
CONTROL INPUT EQUATIONS
$F_z = K_F\left({\Omega_f}^2 + {\Omega_r}^2 + {\Omega_b}^2 + {\Omega_l}^2\right)$ $\tau_\phi = lK_F\left(-{\Omega_r}^2 + {\Omega_l}^2\right)$ (2.13) $\tau_\theta = lK_F\left({\Omega_f}^2 - {\Omega_b}^2\right)$ $\tau_\psi = K_M\left(-{\Omega_f}^2 + {\Omega_r}^2 - {\Omega_b}^2 + {\Omega_l}^2\right)$

2.5 State Variable Representation of Nonlinear Model

The equations of motion can be represented in a state variable representation by defining the state vector $\boldsymbol{x}$ as $\underline{x} = [XYZuvw\phi\theta\psi pqr]^T = [x_1\ x_2 \cdots x_{12}]^T$ and the control vector $\boldsymbol{u}$ as $\underline{u} = \left[F_z\ \tau_\phi\ \tau_\theta\ \tau_\psi\right]^T = [u_1 u_2 u_3 u_4]^T$. The equations from (2.9) to (2.13) can be written compactly as a system of differential equations $\dot{\underline{x}} = f\left(\underline{x}\,, \underline{u}\right)$. These equations are summarized in Appendix B.1 and implemented in Simulink in the subsystem structures shown in Appendix B.2 for a complete nonlinear model of the quadrotor system.

2.6 Simulink Model and Simulation

2.6.1 Quadrotor and Actuator Dynamics

Ultimately, the controller developed in the next chapters must be tested against the complete nonlinear model of the quadrotor dynamics. Therefore, a 6DOF Simulink model was developed using the actuator dynamics and the equations of motions summarized in Sections 2.2 through 2.6. The equations were developed using Simulink block structures shown in Appendix B.2 and condensed into the system shown in Figure 2.5 with subsystems representing the motor mixing, actuator dynamics, quadrotor dynamics, and data logging. This comprises the "open-loop system" of the quadrotor.

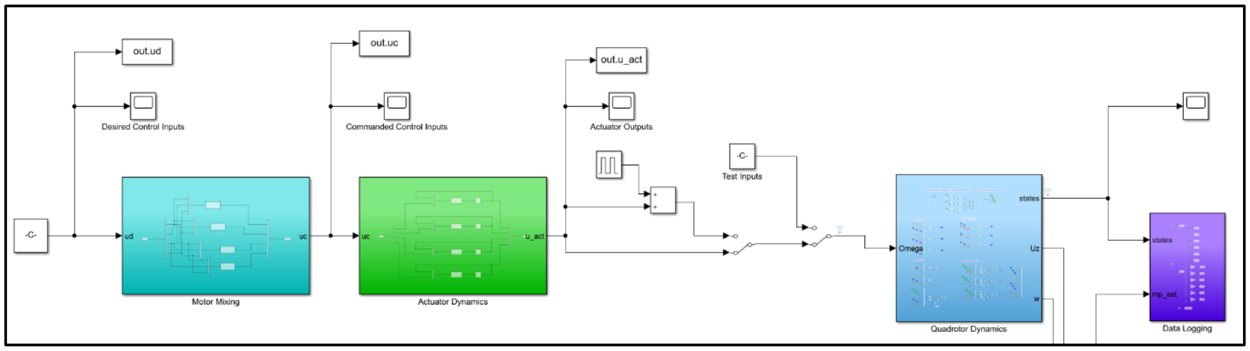


Figure 2.5 – Simulink model of actuator and quadrotor dynamics

A test nonlinear simulation is completed using the system parameters in Table 2.4. The output of the motor mixing and actuator dynamics to the commanded thrust and torque factors summarized in Table 2.5 are shown in Section 2.7.3. The output of the payload estimator is shown in Section 2.7.2 with a convergence of the estimated payload mass $\widehat{m_p}$ to the real mass m_p. The open-loop, *unstable* state responses to the propeller speeds listed in Table 2.5 are shown in Section 2.7.4.

Table 2.4 – Simulation system parameters

Parameter	**Description**	**Value**	**Unit**
m	quadrotor mass without payload	8.733	kg
m_p	payload mass	1.0	kg
g	gravitational acceleration	9.81	m/s^2
J_x	moment of inertia	0.3103	$kg \cdot m^2$
J_y	moment of inertia	0.3103	$kg \cdot m^2$
J_z	moment of inertia	0.5443	$kg \cdot m^2$

Table 2.5 – Input parameters

Parameter	**Description**	**Value**	**Unit**
F_z	net thrust force	95.87	N
τ_ϕ	rolling torque	0	$N \cdot m$
τ_θ	pitching torque	0	$N \cdot m$
τ_ψ	yawing torque	0	$N \cdot m$
Ω_f	front propeller speed	230.79	rad/s
Ω_r	right propeller speed	461.57	rad/s
Ω_b	rear propeller speed	230.79	rad/s
Ω_l	left propeller speed	923.14	rad/s

2.6.2 Actuator Responses

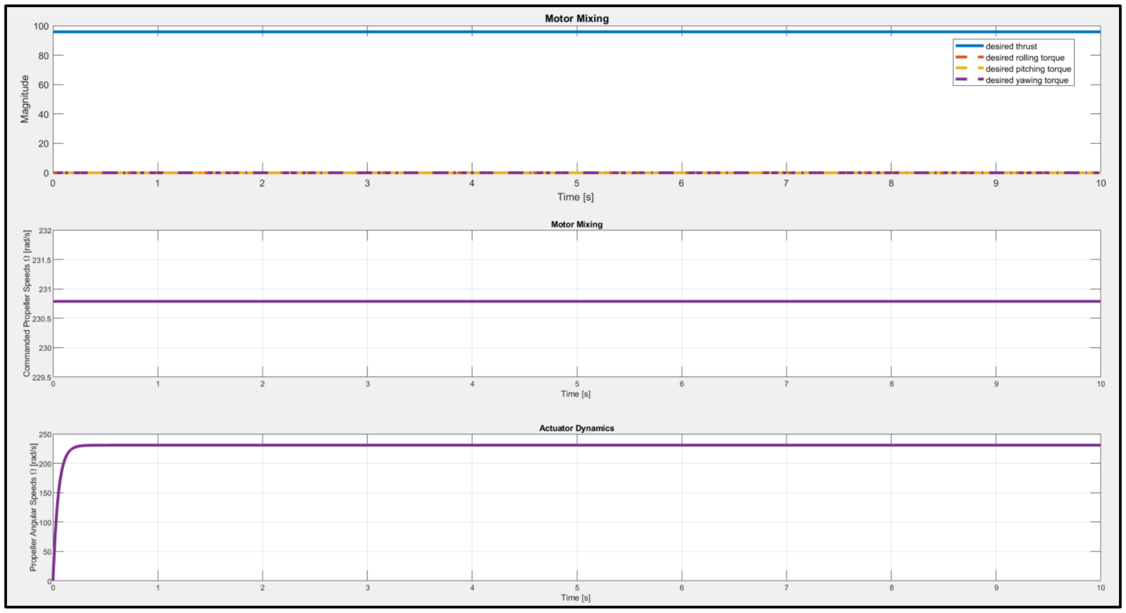


Figure 2.6 – Motor mixing

2.6.3 Open-Loop State Responses

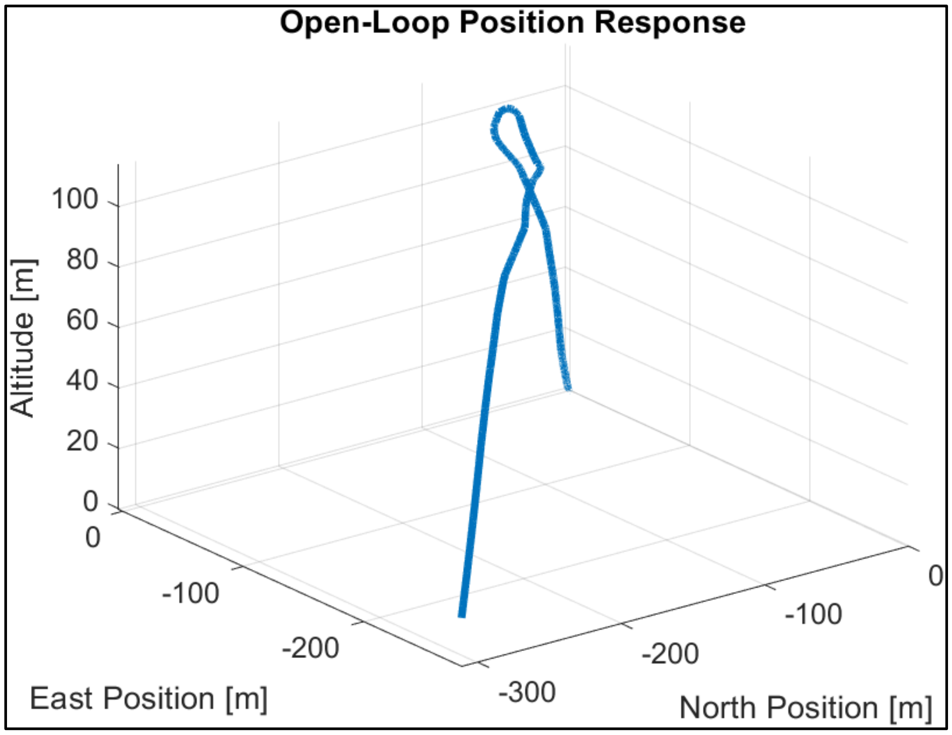


Figure 2.7 – Open-loop response of inertial positions

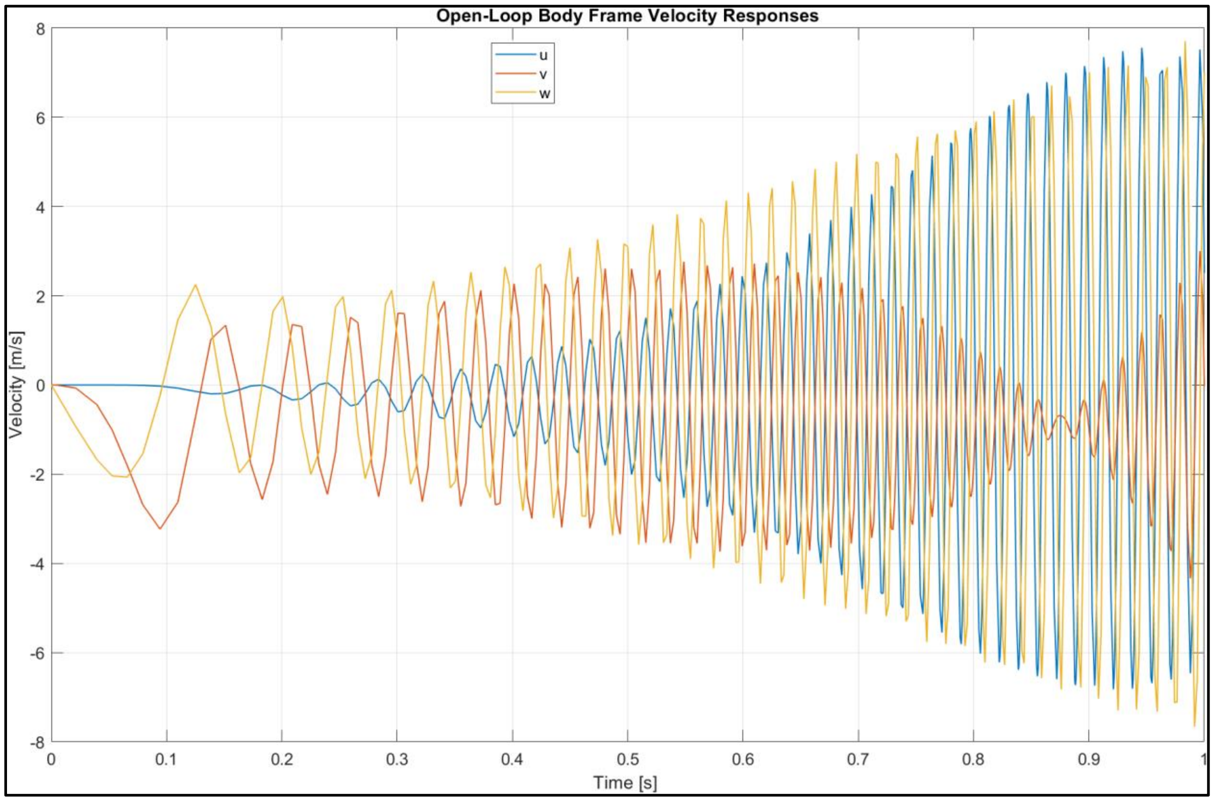


Figure 2.8 – Body frame velocity open-loop response

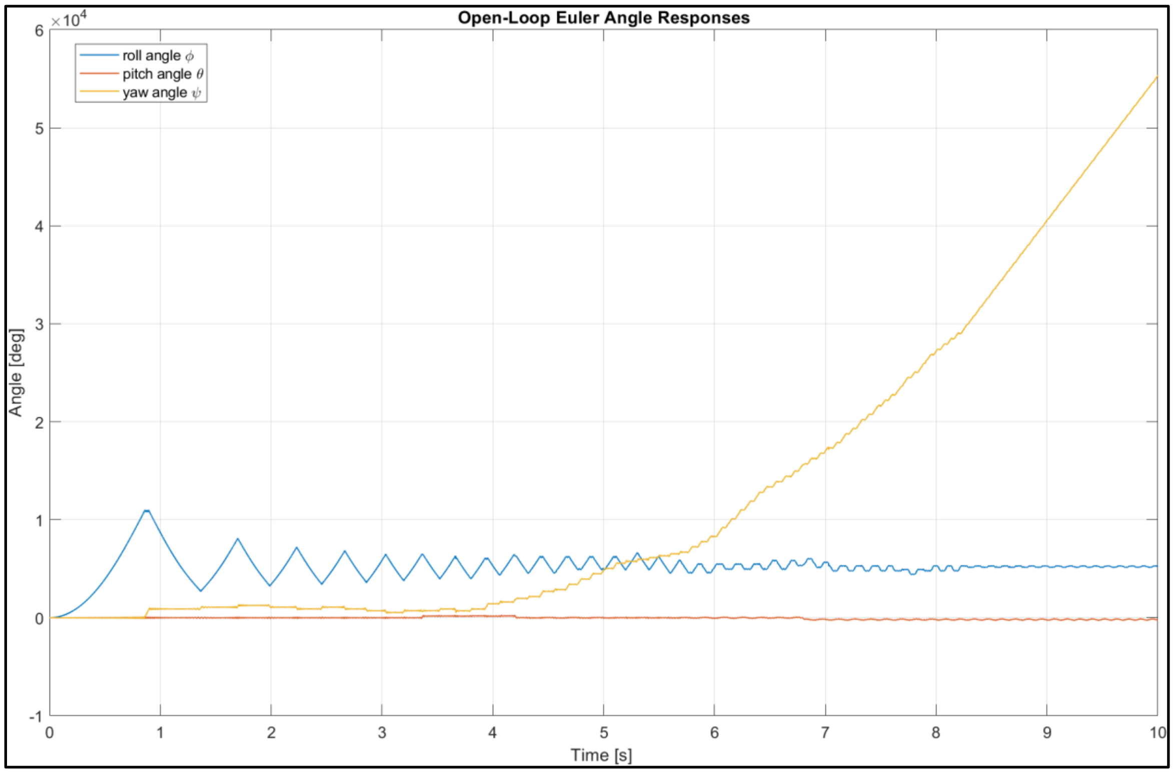


Figure 2.9 – Euler angle open-loop responses

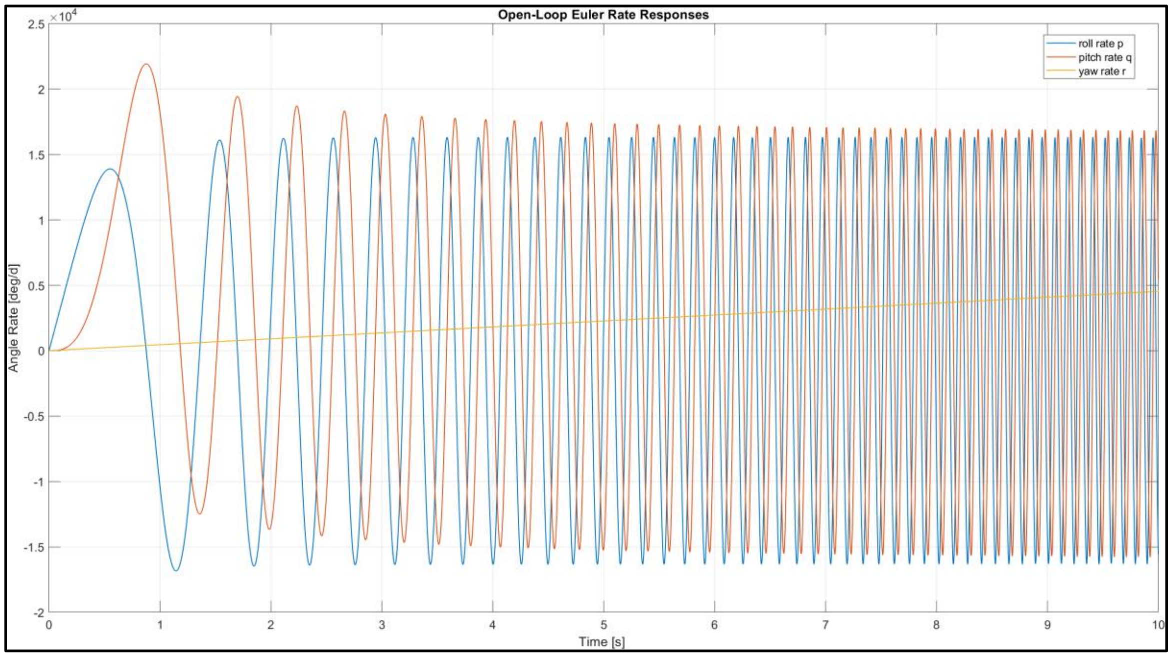


Figure 2.10 – Euler rate open-loop responses

2.7 Linear, Parameter Dependent Model

To develop a controller using methods from linear systems theory, the nonlinear equations of motion are linearized using Jacobian linearization at hovering conditions. At this state, the quadrotor must counteract the downward force of gravity. Therefore, the motor angular speeds Ω_i are equal in magnitude and the vertical forces F_i from each propeller must produce a thrust equal to $mg/4$ yielding $F_z = mg$. From equation (2.13), the motor speeds at hover are given by

$$\Omega_{hov} = \sqrt{\frac{mg}{4K_F}}. \tag{2.14}$$

Also, the position $r_0 = [X_0\ Y_o\ Z_0]$ and the heading (yaw) angle $\psi = \psi_0$ are fixed in a nominal hover state. Since the roll and pitch angles are assumed to be small in this state, small angle approximations are applied, $cos\phi \cong 1$, $cos\theta \cong 1$, $sin\phi \cong \phi$, $sin\theta \cong \theta$.

Note that in a hover state, from equation (2.11), the body angular speeds are roughly equal to the Euler angle rates, $[pqr] \cong [\dot{\phi}\dot{\theta}\dot{\psi}]$.

With these assumptions, the nonlinear equations are linearized with an equilibrium point at $x^* = [X_0 \quad Y_o \quad Z_0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad \psi_0 \quad 0 \quad 0 \quad 0]$ and an operating point at $u^* = mg \cdot [1 \quad 0 \quad 0 \quad 0]$. This is accomplished using the linearization function at the end of the MALTAB script in Appendix A.1 applied to the nonlinear system derive the linear, parameter dependent model given by (2.15).

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u} \tag{2.15}$$

where $A = \begin{bmatrix} 0_{3x3} & S^1_{3x3} & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & S^2_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & 0_{3x3} & S^3_{3x3} \\ 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} \end{bmatrix}$ and $B = \begin{bmatrix} 0_{5x1} & 0_{5x1} & 0_{5x1} & 0_{5x1} \\ -1/m & 0 & 0 & 0 \\ 0_{3x1} & 0_{3x1} & 0_{3x1} & 0_{3x1} \\ 0 & 1/J_x & 0 & 0 \\ 0 & 0 & 1/J_y & 0 \\ 0 & 0 & 0 & 1/J_z \end{bmatrix}$. The smaller matrix entries are defined by (2.16). Recall the moment of inertias are also mass dependent. Also note that x^* and u^* satisfy the equilibrium condition $f(x,u)|_{x^*,u^*} = 0$.

$$S^1_{3x3} = \begin{pmatrix} cos\psi_0 & -sin\psi_0 & 0 \\ sin\psi_0 & cos\psi_0 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

$$S^2_{3x3} = \begin{pmatrix} 0 & -g & 0 \\ g & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{2.16}$$

$$S^3_{3x3} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

This model is extended in Chapter 5 to derive an LPV model specified for a parameter space as the parameter ρ varies in a bounded parameter box. The LPV model is then used to derive the automatically gain scheduled controller.

2.8 Controllability and Observability

A check on the controllability and observability of the state space model at a sample parameter is completed. Using the MATLAB functions $ctrb$ and $obsv$, the rank of both the controllability and observability matrix is full rank, $n = 12$. Therefore, the system (2.15) is fully controllable and fully observable. However, for the LPV system, quadratic stabilizability and quadratic detectability must be satisfied, which is discussed in Chapter 3 and 6.

3. CHAPTER 3

LPV CONTROL THEORY

3.1 Introduction

This chapter presents a linear parameter varying control strategy to control the general LPV system (1.1) based on the LPV $\mathcal{H}_\infty$ self-scheduling technique introduced in the literature review. The mathematical tools required to develop this controller are summarized. For the systems discussed, the closed-loop linear system

$$\begin{aligned} \dot{x} &= Ax + Bw \\ y &= Cx + Dw \end{aligned} \tag{3.1}$$

and closed loop LPV system with state space matrices

$$A(\rho), B(\rho), C(\rho), D(\rho) \triangleq \mathcal{A}, \mathfrak{B}, \mathcal{C}, \mathcal{D} \tag{3.2}$$

are considered in the formulations presented in the next sections. Linear matrix inequalities (LMIs) are utilized to transform suitable control problems into tractable formats which can be solved using optimization solvers. The development of numerical methods and availability of greater computing power has made it possible to efficiently solve LMIs. An important advantage of LMIs in control theory is they can be formulated as convex optimization problems that are computationally tractable [28]. In such a formulation, many types of control problems for which no analytical solution has been found can be solved using LMI methods [28]. LMI problems can be solved with tools such as the MATLAB Robust Control Toolbox [29] or the software package CVX, a MATLAB-based modeling system for convex optimization [30]. Many controls problems can be formulated in terms of LMIs [11]. For this study, the $\mathcal{H}_\infty$ controller design is utilized to develop the proposed LPV controller. It was shown by Apkarian and Gahinet in [4] this control problem can be solved by posing it as an LMI. Therefore, a brief overview of the mathematical properties and techniques of LMIs is presented.

3.2 Linear Matrix Inequalities

Matrix inequalities that are linear or affine in a set of matrix variables are called linear matrix inequalities. The basic form of an LMI can be written as

$$F(x) \triangleq F_0 + \sum_{i=1}^{m} x_i F_i > 0, \tag{3.3}$$

where $x \in \mathcal{R}^m$ is the variable of interest and F_i, F_0 are constant, symmetric, and real matrices. The variable x_i is composed of a single vector composed by stacking column vectors from one or many matrices [11]. Therefore, the function $F(x)$ is expanded

$$F(x) = F(X_1, X_2, \dots, X_n) = F_0 + \sum_{i=1}^{m} G_i X_i H_i > 0, \tag{3.4}$$

where $X_i \in \mathcal{R}^{q_i x p_i}$ are the matrix variables of interest and G_i, H_i are given matrices. Several types of LMI problems are categorized in the literature. For this study, the LMI feasibility and linear objective minimization problems are applicable and summarized below [11].

- The feasibility problem seeks to find a solution $X_1, X_2, \dots, X_n$ such that the inequality (3.4) $F(x) > 0$ holds with no consideration of the optimality of the solution and no guarantee of the uniqueness of the solution.
- The goal of the linear objective minimization problem is to minimize or maximize a linear scalar function $\alpha(X_i)$ subject to satisfying the LMI constraints $F(X_i) > 0$.

3.3 Assumptions of the LPV Plant

A general LPV plant can be described by the model (3.5), where exogeneous inputs w and control inputs u are mapped to controlled outputs z and measured outputs y [16].

$$\Sigma_\rho \begin{cases} \dot{x}(t) = A(\rho)x(t) + B_1(\rho)w(t) + B_2(\rho)u(t) \\ z(t) = C_1(\rho)x(t) + D_{11}(\rho)w(t) + D_{12}(\rho)u(t) \\ y(t) = C_2(\rho)x(t) + D_{21}(\rho)w(t) + D_{22}(\rho)u(t) \end{cases} \tag{3.5}$$

The goal is to develop an LPV controller of the form (3.6) that guarantees quadratic $\mathcal{H}_\infty$ performance for the closed-loop system in Figure 1.5

$$\begin{aligned} \dot{x}_K &= A_k(\rho)x + B_k(\rho)y \\ u &= C_k(\rho)x + D_k(\rho)y \end{aligned} \tag{3.6}$$

The methodology for the self-scheduling $\mathcal{H}_\infty$ technique given by [16] is used to control the quadrotor. This technique is restricted to LPV plants whose matrix descriptions depend affinely on the parameter $\rho \in \mathcal{R}^k$ which varies in a polytope Ω of vertices 2^k. Furthermore, the following assumptions of the plant (3.5) must be satisfied.

$D_{22}(\rho) = 0$	*(A1)*
$B_2(\rho), C_2(\rho), D_{12}(\rho), D_{21}(\rho)$ are parameter independent	*(A2)*
The pairs $(A(\rho), B_2)$ and $(A(\rho), C_2)$ are quadratically stabilizable and quadratically detectable over the polytope Ω	*(A3)*

The assumption (A2) can be alleviated by filtering the control inputs and/or measurement outputs [16]. This is applied to the control inputs in Chapter 5 to remove the parameter dependence of the B matrix.

3.4 Bounded Real Lemma and $\mathcal{H}_\infty$ Norm

To synthesize an LPV controller, the bounded-real lemma (BRL) is used. Following the description of the BRL in [28], the LMI

$$\begin{gathered} \begin{pmatrix} A^TP + PA + C^TC & PB + C^TD \\ B^TP + D^TC & D^TD - I \end{pmatrix} \leq 0 \\ P > 0 \end{gathered} \tag{3.7}$$

where P is positive definite and symmetric is considered. The LMI is feasible if and only the linear system (3.1) satisfies the "non-expansive" condition

$$\int_0^\infty y^Ty\,dt \ \leq \ \int_0^\infty u^Tu\,dt \tag{3.8}$$

for all solutions of (3.1) with zero initial conditions. An equivalent form is the bounded-real condition applied to the transfer function matrix of the linear system (3.1)

$$G(s) = C(sI - A)^{-1}B + D \tag{3.9}$$

expressed as $\|G\|_\infty \leq 1$ where $\|G\|_\infty = sup\{\|G(s)\| \mid Re(s) > 0\}$. This is called the $\mathcal{H}_\infty$ norm of $G(s)$.

To calculate the $\mathcal{H}_\infty$ norm of the transfer function from w to z for the system (3.1) is equivalent to solving the optimization problem

$$\begin{gathered} min\,\gamma \\ subject\ to \begin{pmatrix} A^T + PA & PB & C^T \\ B^T P & -\gamma I & D^T \\ C & D & -\gamma I \end{pmatrix} < 0 \end{gathered} \tag{3.10}$$

in $P > 0$ [11]. The uniqueness of $P > 0$ is not guaranteed, but $\gamma > 0$ is unique.

For a closed-loop LPV system (3.2), the system has quadratic $\mathcal{H}_\infty$ performance γ if and only if there exists a positive definite matrix P such that (3.11) holds for the operating range of the parameter vector ρ [17].

$$\begin{pmatrix} A^T(\rho) + PA(\rho) & PB(\rho) & C^T(\rho) \\ B^T(\rho)P & -\gamma I & D^T(\rho) \\ C(\rho) & D(\rho) & -\gamma I \end{pmatrix} < 0 \tag{3.11}$$

3.5 LPV Control Using $\mathcal{H}_\infty$ Self-Scheduling Technique

Recall the parameter dependent LPV controller expressed in the form

$$\begin{aligned} \dot{x}_K &= A_k(\rho)x + B_k(\rho)y \\ u &= C_k(\rho)x + D_k(\rho)y \end{aligned} \tag{3.12}$$

where K represents the controller and y represents the measurement vector. After finding the matrix P, the existence of a quadratic Lyapunov function $V(x) = x^T Px$ is required to guarantee $\mathcal{H}_\infty$ performance and asymptotic stability for the entire parameter space of ρ [17]. Using the methodology in [31], the parameter dependent controller gains $K(\rho)$ are obtained by expressing LTI vertices of a parameter space in an affine polytopic form and finding the resultant gains

$$K(\rho) = \sum_{i=1}^{q} \sum_{j=1}^{q} \sigma_{1,i}\sigma_{2,j}\, K_r, \qquad (3.13)$$

where q is the number of LTI vertices, $K_r = \begin{pmatrix} A_{K_{i,j}} & B_{K_{i,j}} \\ C_{K_{i,j}} & D_{K_{i,j}} \end{pmatrix}$, and $\sigma_{1,i}\sigma_{2,j}$ are two weighting functions used in the interpolation of the controller gains. With the $\mathcal{H}_\infty$ self-scheduling technique, the LPV system and LPV controller interpolate automatically based on the weighting functions to produce the gains and update based on the condition of the parameter ρ. The parameter dependent gains (3.13) and the controller form (3.12) can be computed using the Robust Control Toolbox's *hinfgs* function. A summary for the characteristic LMI system calculation is provided below.

Characteristic LMI System Calculation

1. LPV controller guarantees some quadratic $\mathcal{H}_\infty$ performance level γ for the closed loop system acting in the polytope Ω if and only if there exist two symmetric matrices R and S satisfying the following LMI's:

$$\begin{pmatrix} R & I \\ I & S \end{pmatrix} \geq 0$$

$$\begin{pmatrix} \mathcal{N}_R & 0 \\ 0 & I \end{pmatrix}^T \begin{pmatrix} A_i R + R A_i^T & R C_{1i}^T & B_{1i} \\ C_{1i} R & -\gamma I & D_{1i} \\ {B_{1i}}^T & {D_{1i}}^T & -\gamma I \end{pmatrix} \begin{pmatrix} \mathcal{N}_S & 0 \\ 0 & I \end{pmatrix} < 0$$

$$\begin{pmatrix} \mathcal{N}_S & 0 \\ 0 & I \end{pmatrix}^T \begin{pmatrix} {A_{1i}}^T S + S A_i & S B_{1i} & {C_{1i}}^T \\ {B_{1i}}^T S & -\gamma I & {D_{1i}}^T \\ C_{1i} & D_{1i} & -\gamma I \end{pmatrix} \begin{pmatrix} \mathcal{N}_S & 0 \\ 0 & I \end{pmatrix} < 0$$

$$i = 1 \ldots q$$

where $\mathcal{N}_R$ and $\mathcal{N}_S$ are the bases of the null spaces of $(B_2^T, 0)$ and (C_2, D_2).

2. From the R and S matrices found in Step 1:

a. Compute full-rank matrices M, N such that $\mathrm{MN^T} = \mathrm{I} - \mathrm{RS}$, where I is the identity matrix.
b. Compute $\mathrm{X_{CL}}$ as the unique solution of the matrix equation $\Pi_2 = \mathrm{X_{CL}}\Pi_1$,

   where $\Pi_2 = \begin{pmatrix} \mathrm{S} & \mathrm{I} \\ \mathrm{N^T} & 0 \end{pmatrix}$ and $\Pi_1 = \begin{pmatrix} \mathrm{I} & \mathrm{R} \\ 0 & \mathrm{M^T} \end{pmatrix}$.

3. With $\mathrm{X_{CL}}$, a possible controller $\Omega_\mathrm{i} = \begin{pmatrix} \mathrm{A_{ki}} & \mathrm{B_{ki}} \\ \mathrm{C_{ki}} & \mathrm{D_{ki}} \end{pmatrix}$ is any solution of the matrix inequality

$$\begin{pmatrix} \mathrm{A_{CL}(w_i)^T X_{CL} + X_{CL} A_{CL}(w_i)} & \mathrm{X_{CL} B_{CL}(w_i)} & \mathrm{C_{CL}(w_i)^T} \\ \mathrm{B_{CL}(w_i)^T X_{CL}} & -\gamma \mathrm{I} & \mathrm{D_{CL}(w_i)^T} \\ \mathrm{C_{CL}(w_i)} & \mathrm{D_{CL}(w_i)} & -\gamma \mathrm{I} \end{pmatrix} < 0$$

This process is internally computed by the *hinfgs* function. The LMI approach to LPV controls design can be summarized as follows [32].

1. For a desired closed-loop system property, derive a sufficiency analysis condition.
2. Evaluate this condition on the LPV closed loop system with the generalized structure of the plant and controller in feedback.
3. Find the control parameters using a convex search with LMIs.
4. If the search is successful, extract the controller parameters.

Note this process is based on a sufficiency condition, meaning if the search is unsuccessful, the LMI constraints applied to the system can be adjusted in an iterative process until a controller candidate is found. The LPV representation of the quadrotor using affine and polytopic forms is described in Chapter 5 and the LPV control is developed in Chapter 6.

4 CHAPTER 4

ONLINE ADAPTIVE PARAMETER ESTIMATION

4.1 Introduction

For the control law to be developed, LPV theory depends on the parameters to be measured or estimated in real-time. Estimation of unknown parameters linear in the equations of motion is a common problem in robotics and control applications [26]. To estimate the mass, an adaptive estimator based on the gradient descent method is utilized. The estimate is determined online and fed into convex constructions which determine the controller gains.

4.2 Gradient Descent Adaptive Law

The adaptive estimator based on the gradient descent method is described in this section. A process to develop this estimator is taken from [33], a set of adaptive control notes, and summarized as follows. Define a transfer function $G(s)$ as

$$G(s) = \frac{z_m s^m + z_{m-1} s^{m-1} + \ldots + z_0}{s^n + p_{n-1} s^{n-1} + \ldots + p_0}, \tag{4.1}$$

where $G(s)$ is strictly proper with $m = n + 1$ and represents a SISO system. Let $G(s) = \frac{Y(s)}{U(s)}$ and rearrange,

$$s^n Y(s) + p_{n-1} s^{n-1} Y(s) + \ldots + p_0 Y(s) = z_m s^m U(s) + z_{m-1} s^{m-1} U(s) + \ldots + z_0 \tag{4.2}$$

To ensure stable parameter convergence, define a Hurwitz characteristic polynomial $\Lambda(s)$

$$\Lambda(s) = s^n + \lambda_{n-1} s^{n-1} + \lambda_{n-2} s^{n-2} + \cdots + \lambda_0. \tag{4.3}$$

In a parameter estimation problem, for any $G(s)$ there is a set of known and unknown coefficients for p_i and z_i. The unknown coefficients are the parameters to be estimated. Rearrange (4.2) so that the known terms are on the left-hand side and the unknown terms are on the right-hand side. Dividing each side by $\Lambda(s)$ and rearranging results in the equations (4.4)

$$\bar{Y}(s) = Y(s) + \bar{Y_f}(s) + \overline{U_f}(s)$$
$$\bar{y}(t) = y(t) + y_f(t) + u_f(t) \tag{4.4}$$
$$\bar{Y}(s) = \Phi^T(s)\Theta,$$

where $\Phi(s)$ and Θ are defined by (4.5).

$$\Theta = \begin{bmatrix} z_0 \\ \vdots \\ z_m \\ \hline \lambda_0 - p_0 \\ \vdots \\ \lambda_{n-1} - p_{n-1} \end{bmatrix} \qquad \Phi(s) = \begin{bmatrix} \frac{1}{\Lambda(s)}[U(s)] \\ \vdots \\ \frac{s^m}{\Lambda(s)}[U(s)] \\ \hline \frac{1}{\Lambda(s)}[Y(s)] \\ \vdots \\ \frac{s^{n-1}}{\Lambda(s)}[Y(s)] \end{bmatrix} \tag{4.5}$$

Input and output filter generators are defined by (4.6)

$$\dot{\xi} = A\xi + Bu(t) \tag{4.6}$$
$$\dot{\eta} = \mathrm{A}\eta + \mathrm{By(t)}$$

where A and B are written in control canonical form,

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -\lambda_0 & -\lambda_1 & -\lambda_2 & \cdots & -\lambda_{n-1} \end{bmatrix} \text{ and } B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}.$$

These state equations guarantee stability of the adaptive process through $\Lambda(s)$. The filtered input and output defined in (4.7) correspond to the known coefficients in the expressions in (4.4) for $\bar{Y}(s)$ where C_u and C_y are vectors whose entries are the known values based on the order of s.

$$u_f(t) = C_u\xi \tag{4.7}$$
$$y_f(t) = C_y\eta$$

Similarly, the state selectors C_I and C_0 in (4.8) select the transfer functions defined in $\Phi(s)$ based on the order of s.

$$\begin{gathered} R = C_I \xi \\ Q = C_0 \eta \\ \phi(\mathrm{t}) = \begin{bmatrix} R \\ Q \end{bmatrix} \end{gathered} \tag{4.8}$$

With $\phi(\mathrm{t})$ and normalizing function m^2, the gradient descent adaptive law is given by (4.9)

$$\begin{gathered} m^2 = 1 + \phi^T \phi \\ \upsilon = \bar{y} - \phi^T \hat{\Theta} \\ \hat{\Theta} = [\widehat{z_0} \ldots \widehat{z_m} \mid \lambda_0 - \widehat{p_0} \ldots \lambda_{n-1} - \widehat{p_{n-1}}]^T \\ \dot{\hat{\Theta}} = \frac{\phi \upsilon}{m^2} \\ \widehat{\Theta_0} = initial\ estimate. \end{gathered} \tag{4.9}$$

The gradient law updates the estimate $\hat{\Theta}$ based on information of $y(t)$ and $\phi(t)$ [34]. An important requirement for the adaptive estimator is it must satisfy the "persistency of excitation" condition to ensure parameter convergence [34]. A bounded signal x(t) is persistently exciting if there exist $\delta > 0$ and $\alpha_0 > 0$ such that

$$\int_{\sigma}^{\sigma+\delta} \mathrm{x}(\mathrm{t})\mathrm{x}^T(t)dt \geq \alpha_0 I, \forall\, \sigma \geq t_0. \tag{4.10}$$

This means the control input must have at least an equal number of frequencies and number of unknown parameters.

4.3 Application to Quadrotor Mass Estimation

The linearized model derived in Section 2.8 is used to derive the transfer function $G(s)$ for the estimator design. In pure translation in the $\widehat{E_z}$ direction, only the control input F_z is of interest. The first column of B is set to B_{SI}. The transfer function from F_z to each state are then found by

$$\begin{gathered} G(s) = C(sI - A)^{-1} B_{SI} \\ C = I_{12x12} \\ B_{SI} = [0_{1x5} \quad -1/m \quad 0_{1x6}]^T. \end{gathered} \tag{4.11}$$

The resultant vector of transfer functions is given by (4.12) with $G_{F_z \to Z}(s) = \frac{1}{ms^2}$ and $G_{F_z \to w}(s) = \frac{-1}{ms}$.

$$G(s) = \begin{bmatrix} 0 & 0 & \dfrac{1}{ms^2} & 0 & 0 & \dfrac{-1}{ms} & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T. \qquad (4.12)$$

Since only a single parameter is to be estimated and the transfer functions are first and second order, the gradient descent method procedure is greatly simplified. Also, since there is only one unknown, the input only needs a single frequency to satisfy the persistency of excitation condition. Choosing $G_{F_z \to w}(s)$ to be the simplest transfer function to implement the gradient descent algorithm yields the gradient law

$$\begin{gathered} \dot{\xi} = -10\xi + F_z \\ \dot{\eta} = -10\eta + w \\ \hat{\Theta} = \widehat{1/m} \\ \dot{\hat{\Theta}} = \gamma \frac{\xi(w - 10\eta - \xi\hat{\Theta})}{1 + \xi^2}, \widehat{\Theta_0} = \Theta(0), \gamma > 0 \end{gathered} \qquad (4.13)$$

Derivation

Let $\Lambda(s) = s + \lambda_0$ be a stable characteristic polynomial and define $\frac{Y(s)}{U(s)} = \frac{-1}{ms}$.

Then $s = \Lambda(s) - \lambda_0$ and $-sY(s) = \frac{1}{m}U(s)$. To avoid a zero crossing of the parameter, define $z_0 = 1/m$.

The negative sign is neglected in the subsequent derivation, but it is noted that the sign on the transfer function is accounted for by negating the time domain output in implementation. Dividing $\Lambda(s)$ on both sides,

$$\frac{sY(s)}{\Lambda(s)} = \frac{z_0 U(s)}{\Lambda(s)} \rightarrow \frac{(\Lambda(s) - \lambda_0)Y(s)}{\Lambda(s)} = \frac{z_0 U(s)}{\Lambda(s)} \rightarrow \mathrm{Y(s)} - \lambda_0 \frac{Y(s)}{\Lambda(s)} = \frac{z_0 U(s)}{\Lambda(s)}$$

Define $\bar{Y}(s) = Y(s) - \lambda_0 \frac{Y(s)}{\Lambda(s)} \rightarrow \bar{y}(t) = y(t) - y_f(t)$

For the right-hand side, set $\Phi(s) = \frac{U(s)}{\Lambda(s)}$ and $\Theta = z_0 \rightarrow \bar{Y}(s) = \Phi(s)\Theta$

- Input Filter Generator

$$\dot{\xi} = -\lambda_0 \xi + u \text{ and } R = 1 \cdot \xi$$

- Output Filter Generator

$$\dot{\eta} = -\lambda_0 \eta + y \text{ and } y_f(t) = \lambda_0 \eta$$

Set $\phi(t) = R$ and choose -10 as a stable pole: $\Lambda(s) = s + 10$

- Gradient Law

$$\dot{\xi} = -10\xi + F_z$$
$$\dot{\eta} = -10\eta + -w$$
$$m^2 = 1 + \phi^2 = 1 + \xi^2$$
$$\hat{\Theta} = \widehat{z_0} = \widehat{1/m}$$
$$\bar{y} = w - 10\eta$$
$$\upsilon = \bar{y} - \phi^T \hat{\Theta} = w - 10\eta - \xi\hat{\Theta}$$
$$\dot{\hat{\Theta}} = \frac{\phi\upsilon}{m^2} \rightarrow \dot{\hat{\Theta}} = \frac{\xi(w - 10\eta - \xi\hat{\Theta})}{1 + \xi^2}$$

Adding a rate of convergence gain γ, $\gamma > 0$, to be tuned in simulation, results in the final equations

$$\dot{\xi} = -10\xi + F_z$$
$$\dot{\eta} = -10\eta + w$$
$$\hat{\Theta} = \widehat{1/m}$$
$$\dot{\hat{\Theta}} = \gamma \frac{\xi(w - 10\eta - \xi\hat{\Theta})}{1 + \xi^2}, \hat{\Theta} = \Theta(0), \gamma > 0$$

If necessary, λ_0 can be treated as a design variable along with γ. There is a tradeoff between convergence time and estimation error. The gain γ is tuned so that a desired convergence time is achieved without causing the parameter trajectory to diverge. In general, allowing more energy into the system can result in fast convergence and reduced error, but with the consequence of increased computational demands.

The above gradient law is applicable to the linear system (2.16). Applying the estimator against the nonlinear system requires a further modification. Recall the equilibrium point of the control input used to linearize the plant is $u_e = [mg \quad 0 \quad 0 \quad 0]$. Neglecting the rise time due to the actuator dynamics, the linear input $U_L = C$ is a constant signal. Therefore, with respect to the input F_z into the nonlinear system,

$$\begin{gathered} U_L = F_z - u_e = F_z - mg \\ F_z = U_L + mg \end{gathered} \tag{4.14}$$

This shows applying F_z directly would require knowledge of the unknown mass to be estimated, defeating the purpose of the estimator. The following modification is made.

Modification

Let $\frac{Y_L(s)}{U_L(s)} = \frac{1}{ms}$ and $F_z = U_L + mg$.

It follows $Y(s) = \left(\frac{Y_L}{U_L}\right) F_z \rightarrow Y(s) = \left(\frac{1}{ms}\right)(C + mg)$

$Y(s) = \frac{C}{ms} + \frac{mg}{ms} \rightarrow Y(s) = \frac{C}{ms} + \frac{g}{s}$

Note the mass in the additional term in $Y(s)$ cancels out for this formulation. Therefore, the output $y(t)$ is adjusted by adding in the time domain $\int g dt$ to the state -w.

The implementation of the mass estimator in Simulink is shown in *Figure 4.1*. Note that w is negated in an outside subsystem and the $\int g dt$ is added to account for the operating point used to linearize the nonlinear system. The estimate $\hat{\Theta}$ is inverted at the output of the estimator to obtain the estimate total mass $\hat{m}$. Subtracting m_{base} from $\hat{m}$ yields the estimate package mass $\widehat{m_p}$.

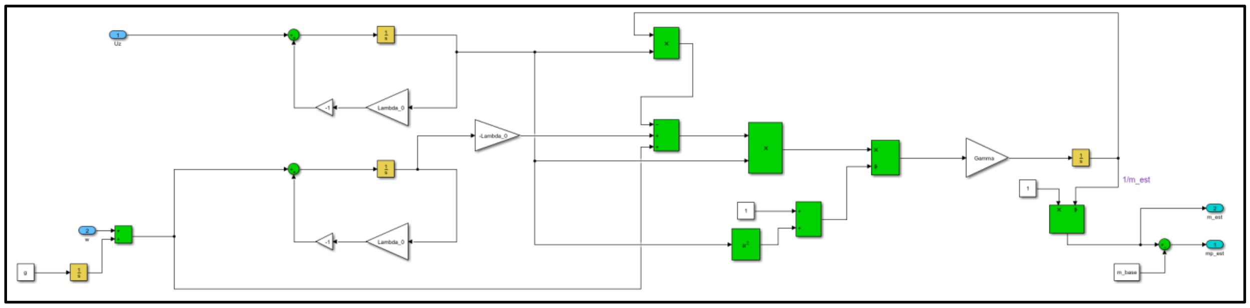


Figure 4.1 – Simulink implementation of mass estimator based on gradient descent law

4.4 Hover State Conditioning

The mass estimation is engaged only when the quadrotor is in a hover state at the point of payload pickup or drop off, typically done at launch. In a hover state, the mass cannot be estimated using the estimator as this will not satisfy the persistency of excitation condition. Therefore, a “hover up” control input needs to be applied with enough force to translate the quadrotor in vertical flight. This action allows the estimator to function as the persistency of excitation condition is satisfied. A hover trigger signal is introduced to control when this control input is activated. An enable signal is defined by HOV_EN in (4.15) and the trigger action is illustrated in Figure 4.2.

$$HOV\_EN = \begin{cases} 1, & 0 < t < \tau \\ 0, & t > \tau \end{cases} \tag{4.15}$$

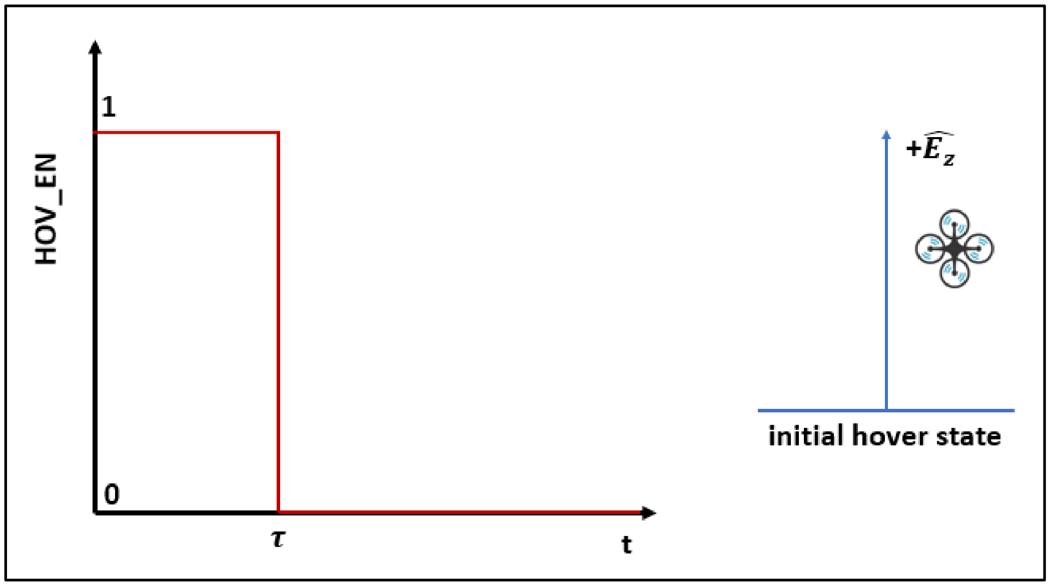


Figure 4.2 – Hover trigger enable signal and “persistency of excitation” requirement

When HOV_EN is true, the control commands switch to the lifting input and the estimator produces a new mass estimate. It is assumed this signal is triggered by a latching, gripping mechanism when a package is attached or dropped off. The quadrotor needs to be able to carry the unknown weight and fly vertically to estimate the mass. Therefore, an additional mass of 0.1 kg and gain of 1.05 is included to U_p to provide a factor of safety and the requisite force needed to transition from hover to vertical flight. This input also satisfies the persistency of excitation condition. The input U_p is applied when the hover trigger is enabled.

$$\begin{aligned} max(m) &= m_{base} + max(m_p) + 0.1 \\ &= 11.173\,kg \\ F_{z_p} &= 1.05 * max(m) * g \end{aligned} \tag{4.16}$$

The remaining three torque inputs $\tau_\phi, \tau_\theta, \tau_\psi$ are zeroed out. Therefore, the control signal when the quadrotor is in the hover state for package pick up or drop-off is $U_p = [F_{z_p} \quad 0 \quad 0 \quad 0]^T$. Applying this input guarantees that the quadrotor will be able to pick up and lift any payload within its design specifications.

The subsystem for hover state conditioning is detailed in Figure 4.3. Its implementation in Simulink is shown in Figure 4.4. As shown, the mass estimation gives a new estimate of the mass when the hover trigger is enabled. Otherwise, it uses the previous mass estimate. Additionally, the control signal is switched from the LPV commands to the input U_p.

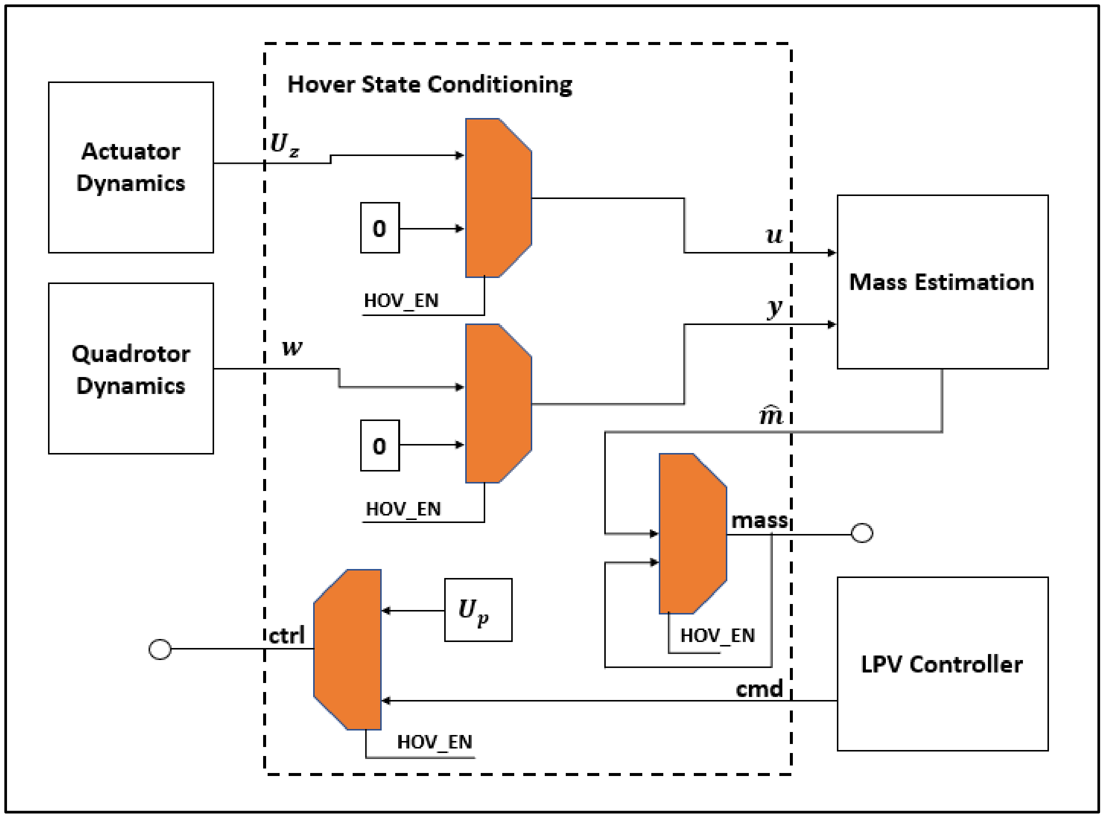


Figure 4.3 – Hover state conditioning subsystem

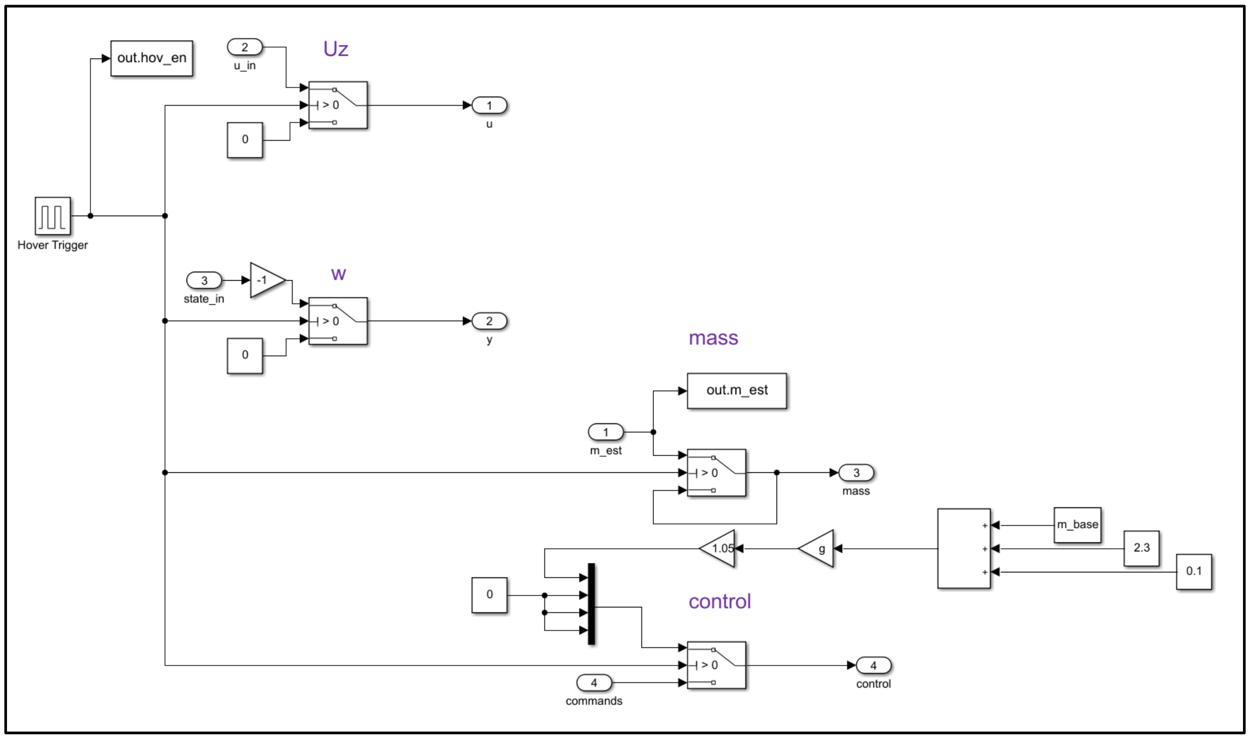


Figure 4.4 – Simulink implementation of hover state conditioning

4.5 Simulation

The mass estimator and hover state conditioning subsystems are added to the nonlinear system in Simulink as shown in Figure 4.5. A simulation was run for an initial estimate $\widehat{\Theta_0} = 1/m_{base}$, a convergence rate gain $\gamma = 10$, and the true package mass is set to $m_p = 2\ kg$. The hover trigger signal (4.15) is modeled in Simulink by a pulse generator with an amplitude of 1, pulse width of 50%, and $\tau = 10\ s$. The input U_p is applied to the system. Simulated against the nonlinear system, the estimator converges to the true system mass $m = 10.773\ kg$ as seen in Figure 4.6.

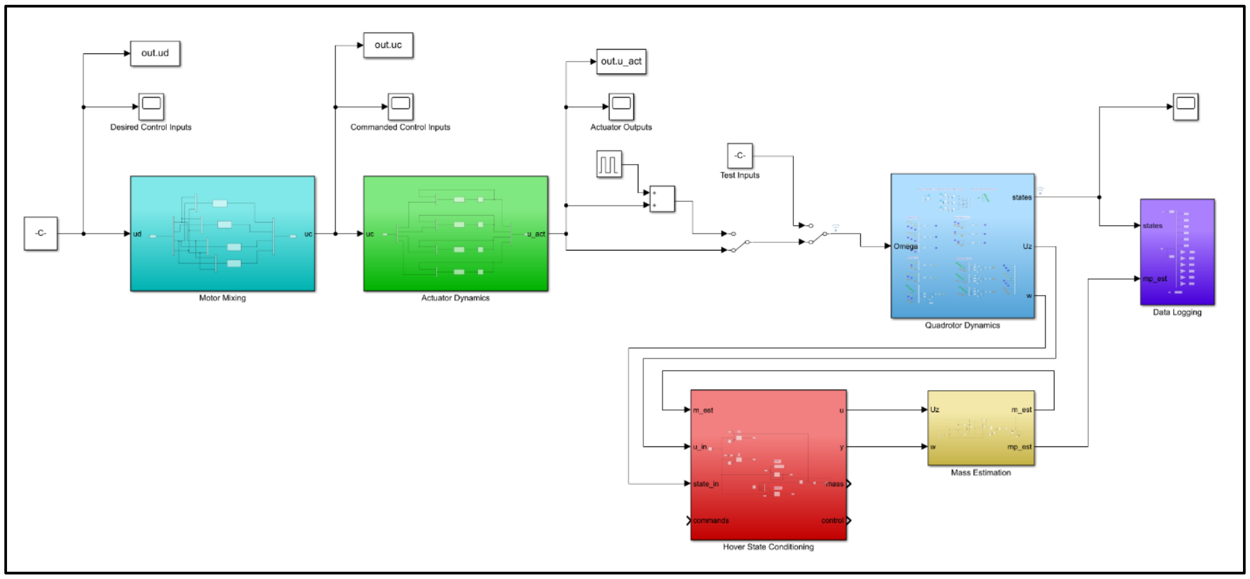


Figure 4.5 – Simulink system with mass estimation and hover state conditioning subsystems

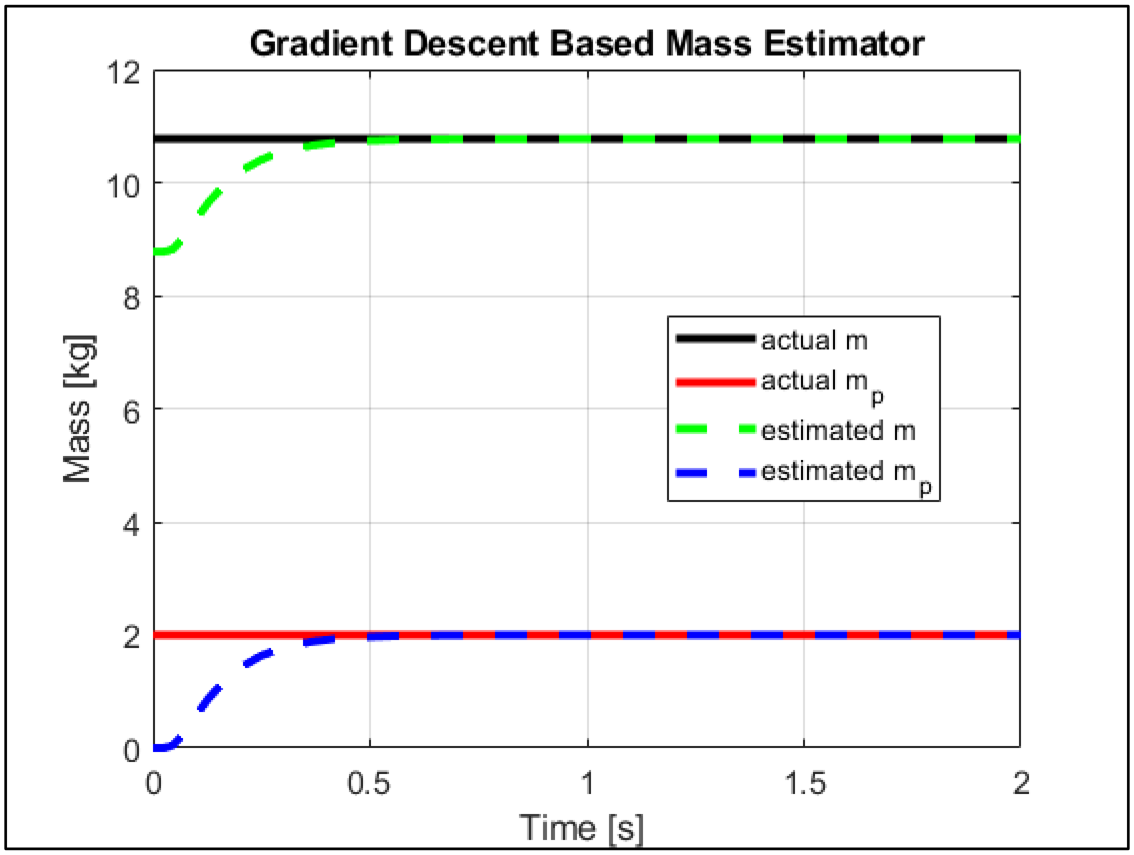


Figure 4.6 – Estimation of unknown mass with information from F_z and w

4.6 Discussion

Several methods exist in the literature for mass estimation. Therefore, justification for choosing an adaptive approach is necessary. As an example, consider an equilibrium search algorithm which enables the mass to be calculated directly when the acceleration $\dot{w} = 0$ is sensed after a sequence of quadrotor maneuvers. Such an approach might cause instability issues as the quadrotor can move towards the ground as it is searching for this point. The gradient law allows the mass estimation to be completed safely in the air and safeguards the estimation process, guaranteeing $\hat{\Theta}$ is bounded by estimating over a time period rather than computing at a single point in time. It also provides flexibility in the mission operations; the quadrotor can fly up while the mass estimation takes place and continues its planned course without having to run through a separate mass determination phase. It should also be noted that the hover state conditioning subsystem for a new mass estimate can be applied during midflight, providing a measure of security during mission operations in case the trigger mechanism is suddenly enabled. The mass can be verified or re-estimated if the package was lost.

The convergence rate gain γ was tuned in simulation to a desired convergence time with zero error. The time to convergence shown in Figure 4.6 is around $0.5\ s$. This is a reasonable time since the quadrotor will be in the hover state for a longer time period and represents a worst-case scenario since an upper bound package mass was used as the test mass. Convergence time and error can be decreased by adjusting the parameters of the gradient law, thereby allowing more energy into the system, but that offers no performance advantages in the hover state. The estimator also has robustness qualities. If a disturbance occurred on the input, such as from a wind gust, resulting in a perturbed input $U = F_z + u_{\Delta}$, the mass can still be estimated accurately because the estimator is designed to adapt to changes in the input. The mass $\hat{m}_p$ is fed into convex constructions which give the parameter dependent gains determined online for the LPV controller. This is the subject of Chapter 5 and 6.

5 CHAPTER 5

LPV SYSTEM REPRESENTATION OF QUADROTOR

5.1 Introduction

The purpose of this chapter is to extend the linear, parameter dependent model described in Chapter 2 to develop a general LPV system that is structured for gain-scheduled $\mathcal{H}_\infty$ control. Functions from the MATLAB Robust Toolbox are utilized to facilitate this process. The generalized LPV system is parametrized by a parameter vector that varies within a bounded parameter box. The reader can refer to Appendix A.1 for the code developed using the functions referenced in this chapter.

5.2 Parameter Space

Substituting the inertia equations (2.6) into the quadrotor equations of motion and applying the same linearization process in Section 2.7 results in a linear, parameter dependent model (5.1) in terms of the payload mass m_p.

$$\dot{\boldsymbol{x}} = A(\rho)\boldsymbol{x} + B(\rho)\boldsymbol{u},$$

$$A = \begin{bmatrix} 0_{3x3} & S^1_{3x3} & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & S^2_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & 0_{3x3} & S^3_{3x3} \\ 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} \end{bmatrix} \text{ and } B = \begin{bmatrix} 0_{5x1} & 0_{5x1} & 0_{5x1} & 0_{5x1} \\ \rho_1 & 0 & 0 & 0 \\ 0_{3x1} & 0_{3x1} & 0_{3x1} & 0_{3x1} \\ 0 & \rho_2 & 0 & 0 \\ 0 & 0 & \rho_2 & 0 \\ 0 & 0 & 0 & \rho_3 \end{bmatrix} \quad (5.1)$$

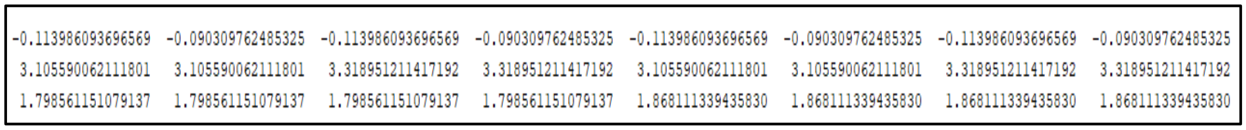
The parameters ρ_1, ρ_2, ρ_3 are fixed functions of m_p, defined by (5.2). From the design specifications, m_p can vary according to $0 \leq m_p \leq 2.3$. The parameter vector $\boldsymbol{\rho}$ is then defined as $\boldsymbol{\rho} = [\rho_1 \quad \rho_2 \quad \rho_3]^T$.

$$\rho_1 = \frac{-1}{m_p + m_{base}}$$
$$\rho_2 = \frac{1}{0.009 * m_p + 0.3013} \tag{5.2}$$
$$\rho_3 = \frac{1}{0.009 * m_p + 0.5353}$$

To specify a parameter-dependent system, a parameter box is defined by finding the lower and upper bounds for each parameter function in $\boldsymbol{\rho}$. These parameters are defined in (5.3) and collected in matrix P.

$$p_{L1} = \rho_1(m_p)|_{m_p=0}\ ,\ \ p_{U1} = \rho_1(m_p)|_{m_p=2.3}$$
$$p_{L2} = \rho_2(m_p)|_{m_p=2.3}\ ,\ p_{U2} = \rho_2(m_p)|_{m_p=0}$$
$$p_{L3} = \rho_3(m_p)|_{m_p=2.3}\ ,\ p_{U3} = \rho_3(m_p)|_{m_p=0} \tag{5.3}$$
$$P = \begin{pmatrix} p_{L1} & p_{U1} \\ p_{L2} & p_{U2} \\ p_{L3} & p_{U3} \end{pmatrix}$$

The parameter box takes on values in a hyperrectangle of $\mathcal{R}^k$ defined by 2^k vertices. Using the Robust Control Toolbox function *pvec* with input P, the parameter box for this system is shown in Figure 5.1. The columns give the coordinates of the 8 corners of the box. The parameter vector $\boldsymbol{\rho}$ ranges within the box and is bounded by the vertices.



```
-0.113986093696569  -0.090309762485325  -0.113986093696569  -0.090309762485325  -0.113986093696569  -0.090309762485325  -0.113986093696569  -0.090309762485325
 3.105590062111801   3.105590062111801   3.318951211417192   3.318951211417192   3.105590062111801   3.105590062111801   3.318951211417192   3.318951211417192
 1.798561151079137   1.798561151079137   1.798561151079137   1.798561151079137   1.868111339435830   1.868111339435830   1.868111339435830   1.868111339435830
```

Figure 5.1 – Vertices of parameter box

Let Π_i denote the column vector of the i^{th} vertex of the parameter box and $\boldsymbol{\rho}_*$ the measurement of the parameter vector online. The measured parameter is then expressed as a convex decomposition over the set of vertices of the parameter box [29].

$$\begin{bmatrix}\rho_1\\ \rho_2\\ \rho_3\end{bmatrix}_* = \alpha_1 \Pi_1 + \ldots + \alpha_8 \Pi_8, \alpha_i \geq 0, \sum_{i=1}^{8} \alpha_i = 1 \tag{5.4}$$

This is generated by the *polydec* function. The parameter box vertices are stored offline while the convex coordinates α_i are calculated online as $\boldsymbol{\rho}_*$ is measured at time t. Note that if the parameter vector matched a vertex's coordinates, representing a bound on the measurement, the convex set $\{\alpha_1 \ldots \alpha_8\}$ would have entry 1 corresponding to the vertex column with all other entries equal to 0. This function is used to compute the controller state-space matrices as the convex combination of controllers at the vertices, as discussed in Chapter 3 and implemented in Chapter 6.

5.3 Control Input Filter

The LPV control theory outlined in Chapter 3 assumes the control and measurement matrices are parameter independent (A2). Therefore, the control inputs are filtered to transform (5.1) and remove the parameter dependence of the matrix B, offloading the parameter functions to the transformed A matrix. The input filter (5.5) is proposed by [16] with $\tilde{u}$ defined as the new control input, A_u is Hurwitz with fast poles compared to the quadrotor poles, and $B_u = C_u = I_{4x4}$.

$$\begin{gathered}\dot{x}_u(t) = A_u x_u(t) + B_u \tilde{u}(t)\\ u(t) = C_u x_u(t)\end{gathered} \tag{5.5}$$

With entries $A(\rho) \in \mathcal{R}^{12x12}$, $B(\rho) \in \mathcal{R}^{12x4}$, $A_u \in \mathcal{R}^{4x4}$, and $B_u \in \mathcal{R}^{4x4}$, the transformed system is given by (5.6).

$$\begin{gathered}\begin{pmatrix}\dot{x}\\ \dot{x}_u\end{pmatrix} = \begin{pmatrix}A(\rho) & B(\rho)C_u\\ 0_{4x12} & A_u\end{pmatrix}\begin{pmatrix}x\\ x_u\end{pmatrix} + \begin{pmatrix}0_{12x4}\\ B_u\end{pmatrix}\tilde{u}\\ A_u = -100 * I_{4x4}\end{gathered} \tag{5.6}$$

The system can be written compactly as (5.7) with $A_p(\rho) \in \mathcal{R}^{16x16}$ and $B_p(\rho) \in \mathcal{R}^{16x4}$.

$$\dot{x}_p(t) = A_p(\rho)x_p(t) + B_p(\rho)\tilde{u}(t) \tag{5.7}$$

Note the dimension of the transformed system has increased from the original system due to the applied filter and $\tilde{u}(t)$ are still the control forces defined as $\left[F_z\ \tau_\phi\ \tau_\theta\ \tau_\psi\right]^T = [u_1 u_2 u_3 u_4]^T$. The controlled outputs $z(t)$ is given by (5.8) where $C_1(\rho) = I_{12x12}$ and $D_{12}(\rho) = 0_{12x4}$.

$$z = \begin{pmatrix} C_1(\rho) & D_{12}(\rho)C_u \end{pmatrix} \begin{pmatrix} x \\ x_u \end{pmatrix} \tag{5.8}$$

5.4 Disturbance Model

Matched disturbances are added to the model by assuming they occur on each state equation containing $F_z, \tau_\varphi, \tau_\theta, \tau_\psi$. This can be due to wind gusts or disturbances in a dynamic environment. Estimation of the actual disturbance is not utilized, instead the uncertainty associated with the disturbance inputs is modeled into the LPV model and the controller synthesis takes this information into account.

$$B_1 = \begin{bmatrix} 0_{5x1} & 0_{5x1} & 0_{5x1} & 0_{5x1} \\ 1 & 0 & 0 & 0 \\ 0_{3x1} & 0_{3x1} & 0_{3x1} & 0_{3x1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.9}$$

The addition of the disturbance matrix $B_1 \in \mathcal{R}^{12x4}$ to the system (5.6) yields (5.10) where $D_{11}(\rho) = D_{12} = 0_{12x4}$.

$$\begin{aligned} \begin{pmatrix} \dot{x} \\ \dot{x}_u \end{pmatrix} &= \begin{pmatrix} A(\rho) & B(\rho)C_u \\ 0_{4x12} & A_u \end{pmatrix} \begin{pmatrix} x \\ x_u \end{pmatrix} + \begin{pmatrix} 0_{12x4} \\ B_u \end{pmatrix} \tilde{u} + \begin{pmatrix} B_1 \\ 0_{4x4} \end{pmatrix} w_d \\ z &= \begin{pmatrix} C_1(\rho) & D_{12}(\rho)C_u \end{pmatrix} \begin{pmatrix} x \\ x_u \end{pmatrix} + D_{11}(\rho)w_d + D_{12}\tilde{u} \end{aligned} \tag{5.10}$$

5.5 Augmented Model

The general parameter dependent model (3.5) is written as (5.11) for consistency with the LPV assumptions (A2).

$$\Sigma_\rho \begin{cases} \dot{x}(t) = A(\rho)x(t) + B_1(\rho)w(t) + B_2u(t) \\ z(t) = C_1(\rho)x(t) + D_{11}(\rho)w(t) + D_{12}u(t) \\ y(t) = C_2x(t) + D_{21}w(t) + D_{22}u(t) \end{cases} \tag{5.11}$$

The system can also be written as the structured system matrix $S(\rho)$.

$$S(\rho) = \left[\begin{array}{c|cc} A(\rho) & B_1(\rho) & B_2 \\ \hline C_1(\rho) & D_{11}(\rho) & D_{12} \\ C_2 & D_{21} & D_{22} \end{array}\right] \tag{5.12}$$

These forms are useful to visualize the model, but for implementation in MATLAB, they must be modified so that the system is in the basic state space form, $\dot{x} = Ax + Bu$ and $y = Cx + Du$. Define a new matrix H_p and input $\widetilde{w}$.

$$\begin{gathered} H_p = [B_p(\rho) \quad B_1(\rho)] \\ \widetilde{w} = \begin{bmatrix} \tilde{u} \\ w_d \end{bmatrix} \end{gathered} \tag{5.13}$$

Applying the definitions (5.13) to the system (5.10) results in the equations

$$\begin{gathered} \begin{pmatrix} \dot{x} \\ \dot{x}_u \end{pmatrix} = \begin{pmatrix} A(\rho) & B(\rho)C_u \\ 0_{4x12} & A_u \end{pmatrix}\begin{pmatrix} x \\ x_u \end{pmatrix} + (B_p(\rho) \quad B_1(\rho))\begin{pmatrix} \tilde{u} \\ w_d \end{pmatrix} \\ z = (C_1(\rho) \quad D_{12}(\rho)C_u)\begin{pmatrix} x \\ x_u \end{pmatrix} + (D_{12} \quad D_{11}(\rho))\widetilde{w}, \end{gathered} \tag{5.14}$$

where $B_p(\rho) = [0_{12x4} \quad B_u]^T$ and $B_1(\rho) = [B_1 \quad 0_{4x4}]^T$. Note the dimensions $H_p \in \mathcal{R}^{16x8}$ and $\widetilde{w} \in \mathcal{R}^8$ are consistent with the overall system dimensions. For the controlled outputs $z(t)$, let $C_p(\rho) = (C_1(\rho) \quad D_{12}(\rho)C_u)$ and $D_p(\rho) = [D_{12} \quad D_{11}(\rho)]$. The resultant system is written compactly as,

$$\begin{aligned}\dot{x}_p(t) &= A_p(\rho)x_p(t) + H_p(\rho)\widetilde{w}(t)\\ z_p(t) &= C_p(\rho)x_p(t) + D_p(\rho)\widetilde{w}(t)\end{aligned} \tag{5.15}$$

To keep the notation easier for the rest of the report, the parameter dependence is dropped on all matrices except for $A_p(\rho)$. Let $H_p(\rho) = B_p$ and $D_p(\rho) = D_p$. For this study, all states are assumed to be measurable, therefore $y(t) = z_p(t)$. The LPV controller needs the x states vector only, the x_u states constitute a $4 - pole$ pre-filter before the LPV controller. Therefore, the measurement matrix $C_p(\rho)$ is redefined to $C_p = (I_{12x12}|0_{12x4})$ to extract only the x states. Redefining $\tilde{u} = u$, the final system is described by (5.16).

$$\begin{gathered}\dot{x}_p(t) = A_p(\rho)x_p(t) + B_p\widetilde{w}(t)\\ y(t) = C_p x_p(t) + D_p\widetilde{w}(t)\\ x_p = \begin{bmatrix} x \\ x_u \end{bmatrix}, \widetilde{w} = \begin{bmatrix} u \\ w_d \end{bmatrix}\end{gathered} \tag{5.16}$$

5.6 LPV Model

To obtain the overall LPV system, the parameter dependent model (5.16) is parametrized by the vector $\boldsymbol{\rho}$ to produce an affine model. The system matrices $A(\rho)$, $B(\rho)$, $C(\rho)$, $D(\rho)$ can be rewritten as an affine combination of $\boldsymbol{\rho}$ and parameter independent system matrices. The number of matrices necessary is $k + 1$. These equations are summarized in Table 5.1. Note that matrix A of (2.15) is evaluated at $\psi_0 = 0^{\circ}$.

Table 5.1 – Summary of matrices for LPV model

$$A_p(\rho) = A_0 + \rho_1 A_1 + \rho_2 A_2 + \rho_3 A_3$$

$$A_0 = \begin{pmatrix} 0_{3x3} & M_1 & 0_{3x3} & 0_{3x3} & 0_{3x4} \\ 0_{3x3} & 0_{3x3} & M_2 & 0_{3x3} & 0_{3x4} \\ 0_{3x3} & 0_{3x3} & 0_{3x3} & M_3 & 0_{3x4} \\ 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{6x3} & 0_{3x4} \\ 0_{4x3} & 0_{4x3} & 0_{4x3} & 0_{1x3} & A_u \end{pmatrix}$$

$$A_1 = \begin{pmatrix} 0_{12x12} & 0_{5x1} & 0_{12x3} \\ 0_{2x12} & 1 & 0_{2x3} \\ 0_{2x12} & 0_{10x1} & 0_{2x3} \end{pmatrix}$$

$$A_2 = \begin{pmatrix} 0_{12x13} & 0_{9x3} \\ 0_{2x13} & M_4 \\ 0_{2x13} & 0_{4x3} \end{pmatrix}$$

$A_3 = \begin{pmatrix} 0_{12x15} & 0_{11x1} \\ 0_{2x15} & 1 \\ 0_{2x15} & 0_{4x1} \end{pmatrix}$ $M_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}, M_2 = \begin{pmatrix} 0 & -g & 0 \\ g & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$ $M_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad M_4 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$
$B_p(\rho) = B_0 + \rho_1 B_1 + \rho_2 B_2 + \rho_3 B_3$ $B_0 = B_p$ $B_1 = B_2 = B_3 = 0_{16x8}$
$C_p(\rho) = C_0 + \rho_1 C_1 + \rho_2 C_2 + \rho_3 C_3$ $C_0 = C_p$ $C_1 = C_2 = C_3 = 0_{12x16}$
$D_p(\rho) = D_0 + \rho_1 D_1 + \rho_2 D_2 + \rho_3 D_3$ $D_0 = D_p$ $D_1 = D_2 = D_3 = 0_{12x8}$

The matrices $(\cdot)_i$, $i = 0, \ldots, 3$, summarized in Table 5.1 can be interpreted as "frozen" LTI system matrices whose affine combination translate into $A(\cdot), B(\cdot), C(\cdot), D(\cdot)$ which are fixed functions of the uncertain parameter vector $\boldsymbol{\rho}$. Now, each system $\{A_i, B_i, C_i, D_i\}$ has a state space representation S_i defined by a structured matrix using the MATLAB function *ltisys*. The overall LPV system is represented by $S(\rho)$ defined by (5.17).

$$\underbrace{\begin{bmatrix} A_p(\rho) & B_p(\rho) \\ C_p(\rho) & D_p(\rho) \end{bmatrix}}_{S(\rho)} = \underbrace{\begin{bmatrix} A_0 & B_0 \\ C_0 & D_0 \end{bmatrix}}_{S_0} + \rho_1 \underbrace{\begin{bmatrix} A_1 & B_1 \\ C_1 & D_1 \end{bmatrix}}_{S_1} + \rho_2 \underbrace{\begin{bmatrix} A_2 & B_2 \\ C_2 & D_2 \end{bmatrix}}_{S_2} + \rho_3 \underbrace{\begin{bmatrix} A_3 & B_3 \\ C_3 & D_3 \end{bmatrix}}_{S_3} \quad (5.17)$$

With the parameter box and the $S(\rho)$ representation, the MATLAB function *psys* specifies an affine parameter-dependent system with 4 systems and 3 parameters as a structured matrix of dimension $29x110$. Each system has 16 states, 8 inputs, and 12 outputs. The MATLAB code in Appendix A.1 was used to produce this LPV system representation. To obtain the polytopic representation of the affine parameter-dependent system, the function *aff2pol* is used to build the vertex system. In this case, the polytopic model has 8 vertex systems of dimension $29x214$ representing the instances of the LPV system at the corners of the box defined previously by *pvec*. Let these vertex systems be $S(\Pi_i)$. Therefore, $S(\rho)$ over the corners Π_i can be expressed by

$$S(\rho) = \alpha_1 S(\Pi_1) + \ldots + \alpha_8(\Pi_8), \alpha_i \geq 0, \sum_{i=1}^{8} \alpha_i = 1 \qquad (5.18)$$

5.7 Interconnected LPV System

The design of the gain-scheduled $\mathcal{H}_\infty$ controller is developed according to Figure 5.2, modified from reference [35]. The open-loop plant $G(s)$ is defined by the system matrix $S(\rho)$. This structure utilizes a two-degrees-of-freedom controller $C(s)$ to achieve better performance. The blocks $W_r(s)$ and $W_d(s)$ are weighting functions applied to the reference input r and disturbance inputs w_d, respectively, used to properly scale the magnitudes. The weights $W_u(s)$ and $W_p(s)$ are applied to the control input u and the system error e to penalize control effort while achieving the desired performance in terms of transient response and stability margins.

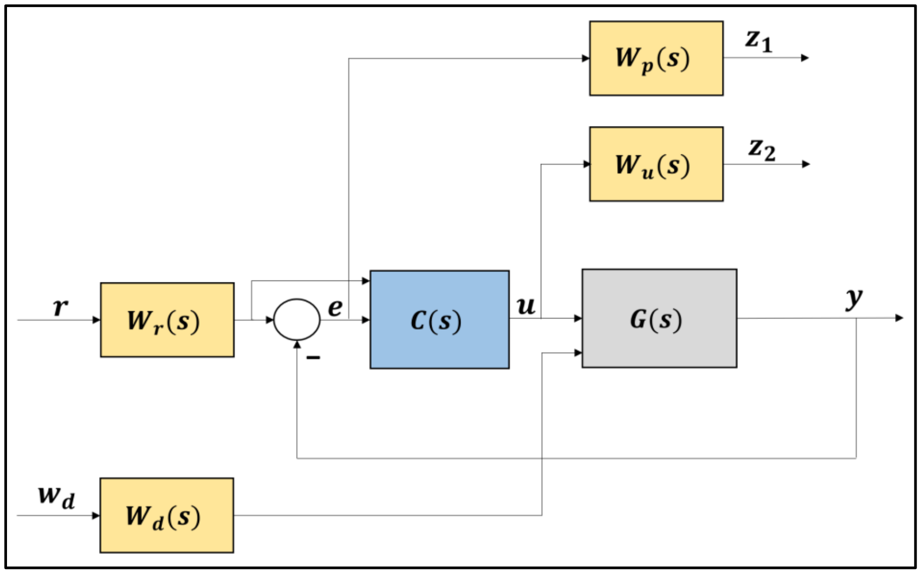


Figure 5.2 – Simplified interconnected LPV System with 2DOF controller

6 CHAPTER 6

LPV CONTROL OF QUADROTOR

6.1 Problem Definition and Control Objectives

The goal of the delivery quadrotor is to track a given reference trajectory and deliver a package of unknown mass. Due to the quadrotor's uncertain environment and unmodeled dynamics, the controller is to handle tracking, disturbance rejection, and to compensate for uncertainty due to modeling errors and parameter variations. Therefore, the following objectives must be achieved by the stabilizing controller,

- The closed-loop system is quadratically stable over the polytope Ω.
- Minimize closed-loop quadratic $\mathcal{H}_\infty$ norm, $\|F(P,C)\|_\infty < \gamma_{min}$, where P is the generalized plant containing the LPV plant G, C is the LPV controller, and $F(P,C)$ is the closed-loop system, subject to actuator dynamics and disturbances.

The $\mathcal{H}_\infty$ norm can be interpreted as the peak gain of the plant across all frequencies and input directions [36]. Therefore, the objective of the gain-scheduled $\mathcal{H}_\infty$ controller is to "push down" the peaks of the frequency response of $F(P,C)$. The performance level γ_{min} is set to 1, however, this is a design goal rather than a requirement. As long the controller achieves adequate performance [37], a $\gamma > 1$ is acceptable, but the result is a more conservative controller. Once the controller is synthesized with minimum norm, analysis of whether the controller met the objectives is completed in Section 6.4. The reader can refer to Appendix A.1 for the LPV code developed using the functions referenced in this chapter.

6.2 Weights Selection and $\mathcal{H}_\infty$ Mixed-Sensitivity Design

The purpose of the tracking, disturbance, performance, and control weights selection is to meet specifications over the plant operating domain. Weights selection is not an exact science, but an LTI closed-loop analysis can be performed at sample parameters of m_p to give starting weights for controls synthesis. The weights can then be tuned in simulation until the desired performance and robustness is achieved. A guide to weights selection for $\mathcal{H}_\infty$ control is provided by the helicopter control case study in reference [11] and design examples in reference [32]. The filters chosen were taken from these examples.

The setpoint weight, $W_r(s)$, is used to scale the reference states. These will be static gains on a diagonal matrix. The purpose of this is to normalize the magnitudes of the reference states based on the units of the measured states. Similarly, a common scaling factor is applied to the disturbance inputs through $W_d(s)$. For the purposes of simulation to demonstrate the disturbance rejection, a scalar value $\beta \geq 1$, is applied on each entry of the diagonal matrix.

The performance weight, also called the sensitivity weight, is applied to the error of the closed-loop system. The entries $W_{p_{x_i}}(s)$ are applied to each state in the controlled output which can be a mix of low-pass filters and static gains. The weight is expressed as a diagonal matrix of transfer functions or static gains. Similarly, the control or robustness weight, is applied to the control input. Typically, it is a high-pass filter applied to each input. The main idea of the performance and robustness weights is to shape the response so that the gain of the loop transfer function $L = GK$ is high at a lower frequency range and low at a higher frequency range. The resultant system operates within this space. The frequency ranges are set by disturbance rejection, command following at the lower frequencies and set by noise and plant uncertainty at the higher frequencies [38]. The goal of the weight selection is then to properly identify the cutoff frequencies for these two ranges as it pertains to the quadrotor's dynamics and operating environment. This analysis is done in the frequency domain. One approach is to first determine the $W_p(s)$ that achieves the best performance with no robustness weight applied, then adjust $W_u(s)$ and iterate until a suitable γ_{min} is achieved.

The weights are summarized by (6.1) and (6.2).

$$W_r(s) = \begin{pmatrix} W_{r_{x_1}}(s) & \cdots & 0 \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & W_{r_{x_{12}}}(s) \end{pmatrix} \qquad W_p(s) = \begin{pmatrix} W_{p_{x_1}}(s) & \cdots & 0 \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & W_{p_{x_{12}}}(s) \end{pmatrix} \tag{6.1}$$

$$W_d(s) = \begin{pmatrix} W_{d_{w_1}}(s) & \cdots & 0 \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & W_{d_{w_4}}(s) \end{pmatrix} \quad W_u(s) = \begin{pmatrix} W_{u_{u_1}}(s) & \cdots & 0 \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & W_{u_{u_4}}(s) \end{pmatrix} \tag{6.2}$$

The performance weights (6.3) are applied to the system

$$W_{p_j}(s) = \frac{2.01}{s + 0.201}$$
$$W_{p_{rates}}(s) = \frac{2s}{s^2 + 8.5s + 18} \tag{6.3}$$

where $W_{p_j}(s)$ is a low-pass filter applied to the position, velocity, and Euler angle states with cutoff frequency $\omega_c = 20\ rad/s$. The filter $W_{p_{rates}}(s)$ is applied to the attitude rates. The high-pass filter with a cutoff frequency of $\omega_c = 1000\ rad/s$ was selected for the control weight and applied for each control input.

$$W_{u_i}(s) = \frac{9.678\ s^3 + 0.029\ s^2}{s^3 + 1.206e4\ s^2 + 1.136e7\ s + 1.066e10} \tag{6.4}$$

W_r, W_d, W_p, and W_u are specified as LTI systems using the function *ltisys*. Their respective singular values plot are shown in Figure 6.2 and Figure 6.3. The weights are inputs to the function *sconnect* which constructs the $\mathcal{H}_\infty$ plant $P(s)$ associated with the control structure shown in Figure 5.2. The control structure is expanded in Figure 6.1 to reflect the feedback and feedforward control as well as the disturbance inputs into the plant. Recall the augmented input $\widetilde{w}$ from Chapter 5. The output of the unweighted plant $G(s)$ can be rewritten in terms of $G_1(s)$ and $G_2(s)$ and the inputs $u(s), w(s)$.

$$Y(s) = G(s)\widetilde{w}(s)$$
$$Y(s) = [G_1(s) \quad G_2(s)] \begin{bmatrix} u(s) \\ w(s) \end{bmatrix} \tag{6.5}$$

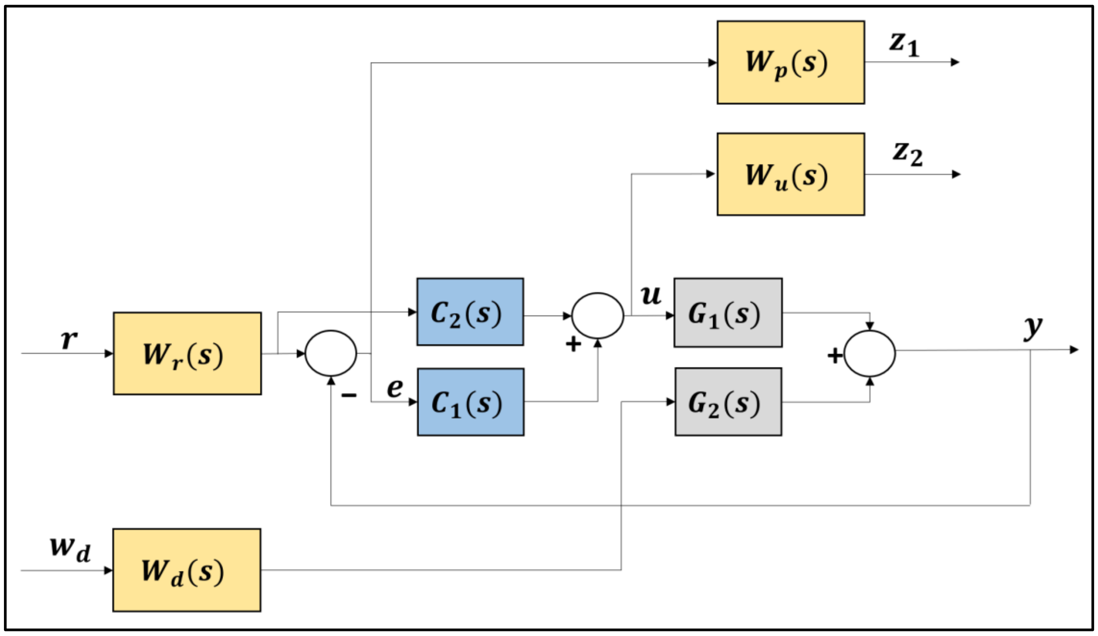


Figure 6.1 – Expanded 2DOF control structure

The weights are then formulated as an $\mathcal{H}_\infty$ mixed sensitivity problem. The advantage of mixed sensitivity is it allows the designer to simultaneously shape the frequency responses for tracking and disturbance rejection, noise reduction and robustness, and controller effort [39]. The objective is to minimize the cost function,

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} r \\ w_d \end{bmatrix}$$

$$H = \begin{bmatrix} W_r W_p - W_r W_p (C_1 + C_2) G_1 S_1 & -W_d W_p G_2 S_1 \\ W_r W_u (C_1 + C_2) S_1 & -W_d W_u C_1 G_2 S_1 \end{bmatrix} \tag{6.6}$$

$$\|H\|_\infty < \gamma_{min},$$

where $\|H\|_\infty$ is the $\mathcal{H}_\infty$ norm of the MIMO transfer function H from $\begin{bmatrix} r \\ w_d \end{bmatrix}$ to $\begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$.

Derivation of (6.6) Transfer Function H

From Figure 6.1, the following equations are written from the block diagram relationships. The s notation is dropped for convenience.

$$y = G_1 u + W_d G_2 w_d$$
$$e = W_r r - y$$
$$u = W_r C_2 r + C_1 e$$

Substituting the error e and output y into the control equation u,

$$u = W_r C_2 r + C_1(W_r r - y) \rightarrow$$
$$(1 + G_1 C_1)y = (W_r C_2 G_1 + W_r C_1 G_1)r + (W_d G_2)w_d$$

Let $S_1 = (1 + C_1 G_1)^{-1}$ be the sensitivity function. Then,

$$y = [W_r(C_1 + C_2)G_1 S_1]r + (W_d G_2 S_1)w_d$$

Also, from Figure 6.1,

$$z_1 = W_p e$$
$$z_2 = W_u u$$

Substituting the results from above into z_1, z_2 and rearranging until z_1 and z_2 are expressed in terms of only the inputs r and w_d yields,

$$z_1 = \left[W_r W_p - W_r W_p(C_1 + C_2)G_1 S_1\right]r - (W_d W_p G_2 S_1)w_d$$
$$z_2 = [W_r W_u(C_1 + C_2)S_1]r - (W_d W_u C_1 G_2 S_1)w_d$$

Written in matrix form,

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} W_r W_p - W_r W_p(C_1 + C_2)G_1 S_1 & -W_d W_p G_2 S_1 \\ W_r W_u(C_1 + C_2)S_1 & -W_d W_u C_1 G_2 S_1 \end{bmatrix} \begin{bmatrix} r \\ w_d \end{bmatrix}$$

With respect to the closed-loop system, the interpretation of $\mathcal{H}_\infty$ norm minimization is interpreted as minimizing the effects of the reference demands and disturbances on the system error and control effort in the mapping from $\begin{bmatrix} r \\ w_d \end{bmatrix}$ to $\begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$.

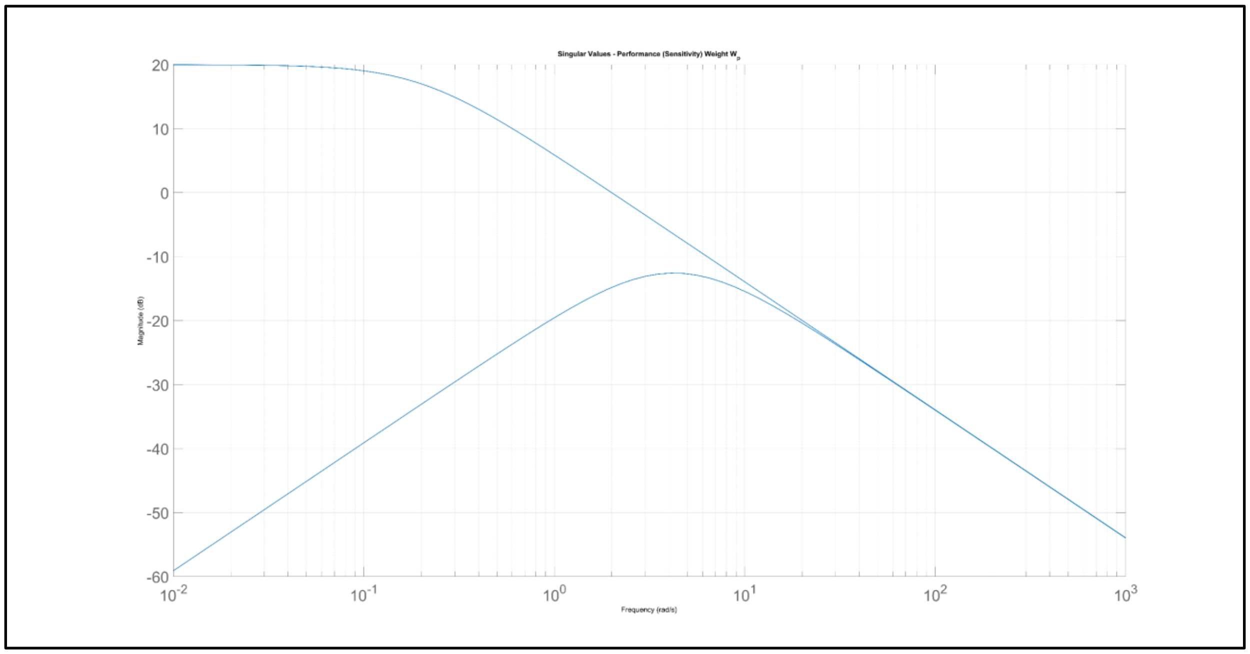


Figure 6.2 – Singular values of performance (sensitivity) weight $W_p(s)$

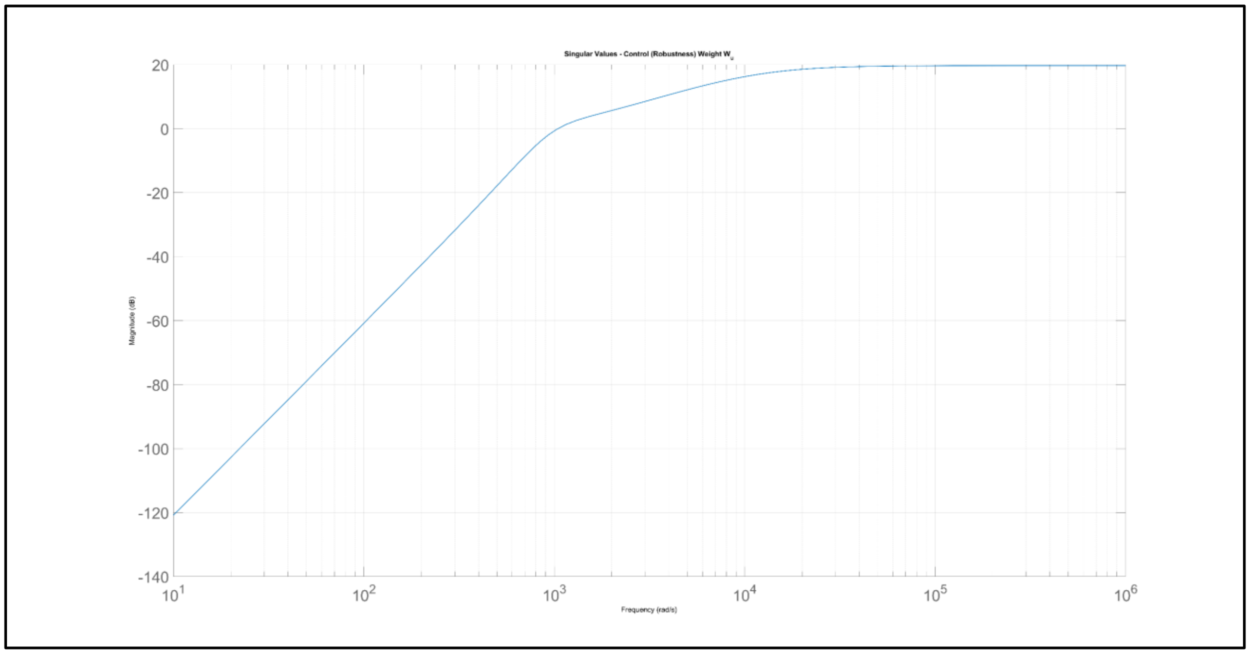


Figure 6.3 – Singular values of control (robustness) weight $W_u(s)$

With the weights selected, the augmented plant $P(s)$ is expressed in polytopic form with eight vertex systems. Each system has 43 states, 20 inputs, and 56 outputs.

6.3 Synthesis of Gain-Scheduled $\mathcal{H}_\infty$ Controllers

The general procedure to produce the gain-scheduled controller using the MATLAB Robust Control Toolbox is provided by references [35] and [32]. The function *sconnect* is first used to construct the $\mathcal{H}_\infty$ plant $P(s)$. Then two-degrees-of-freedom LPV controller C(s) consisting of feedback and feedforward gains acting on the closed-loop error and reference.

$$\begin{gathered} K(\rho) = [K_1(\rho) \quad K_2(\rho)] \\ z = \begin{bmatrix} e \\ r \end{bmatrix} \end{gathered} \tag{6.7}$$

The reference states $\boldsymbol{r}$ and error $\boldsymbol{e} = \boldsymbol{r} - \boldsymbol{y}$ are each of dimension $12x1$ and the control inputs $\boldsymbol{u}$ are of dimension $4x1$. Therefore, the gains K_1, K_1 are dimension $4x12$. The control $\boldsymbol{u}$ is expressed as (6.8) or written compactly as $\boldsymbol{u} = K(\rho)\boldsymbol{z}$, $K \in \mathcal{R}^{4x24}, z \in \mathcal{R}^{24}$.

$$\boldsymbol{u} = [K_1(\rho) \quad K_2(\rho)] \begin{bmatrix} e \\ r \end{bmatrix} = K_1(\rho)\boldsymbol{e} + K_2(\rho)\boldsymbol{r} \tag{6.8}$$

As described in Chapter 3, the control commands u are generated from the controller state space system (6.9).

$$K_\rho \begin{cases} \dot{\zeta} = A_K(\rho)\zeta + B_K(\rho)z \\ u = C_K(\rho)\zeta + D_K(\rho)z \end{cases} \tag{6.9}$$

The vertex controllers K_{Π_i} given by (6.10) in terms of the controller matrices are determined offline with the functions *hinfgs* and *psinfo*.

$$K_{\Pi_i} = \begin{pmatrix} A_K(\Pi_i) & B_K(\Pi_i) \\ C_K(\Pi_i) & D_K(\Pi_i) \end{pmatrix} \tag{6.10}$$

Recall the parameter vector measurement $\boldsymbol{\rho}_*$ at time t is expressed as a convex combination with α_i. The controller state space matrices are then computed as the convex interpolation of the vertex controllers K_{Π_i} as shown in (6.11). Note the gain-scheduled $\mathcal{H}_\infty$ controller (6.11) is specified in polytopic form.

$$\sum_{i=1}^{q} \alpha_i K_{\Pi_i} = \begin{pmatrix} A_K(\rho) & B_K(\rho) \\ C_K(\rho) & D_K(\rho) \end{pmatrix} \tag{6.11}$$

Now, the controller matrices $A_K \in \mathcal{R}^{43x43}, B_K \in \mathcal{R}^{43x24}, C_K \in \mathcal{R}^{4x43}, D_K \in \mathcal{R}^{4x24}$ of (6.11) can be updated with the new controller as $\boldsymbol{\rho}_*$ is calculated from the estimate $\widehat{m_p}$. Stated another way, the vertex controllers K_{Π_i} are stored offline. As the convex decomposition coefficients α_i are calculated online, the combination of α_i and K_{Π_i} give the current controller matrices at $\boldsymbol{\rho}_*$. These matrices update the controller equations (6.9). With the state error and reference fed into the control law, the control commands $\boldsymbol{u}$ are generated. The process to obtain the controller matrices is shown in Figure 6.4 to visualize the process more clearly.

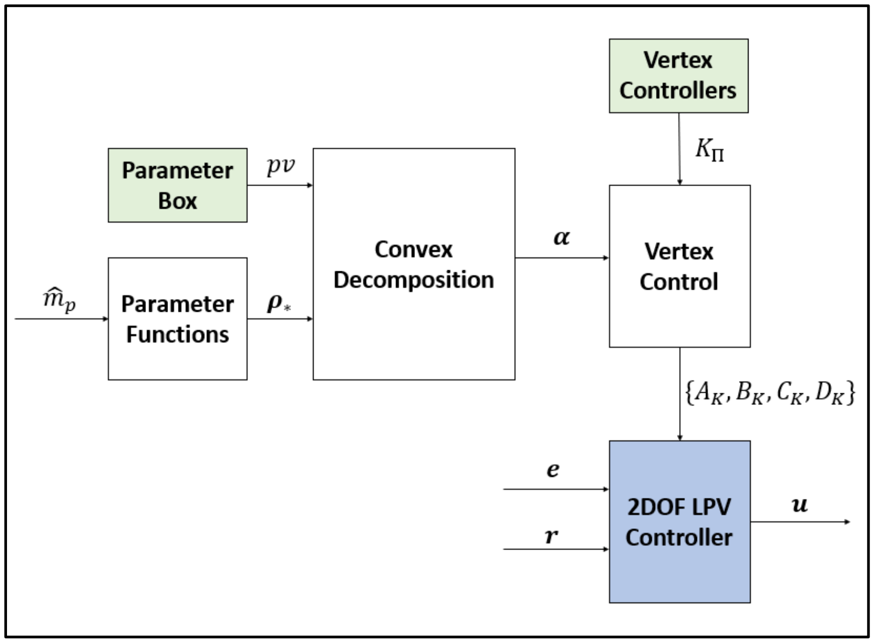
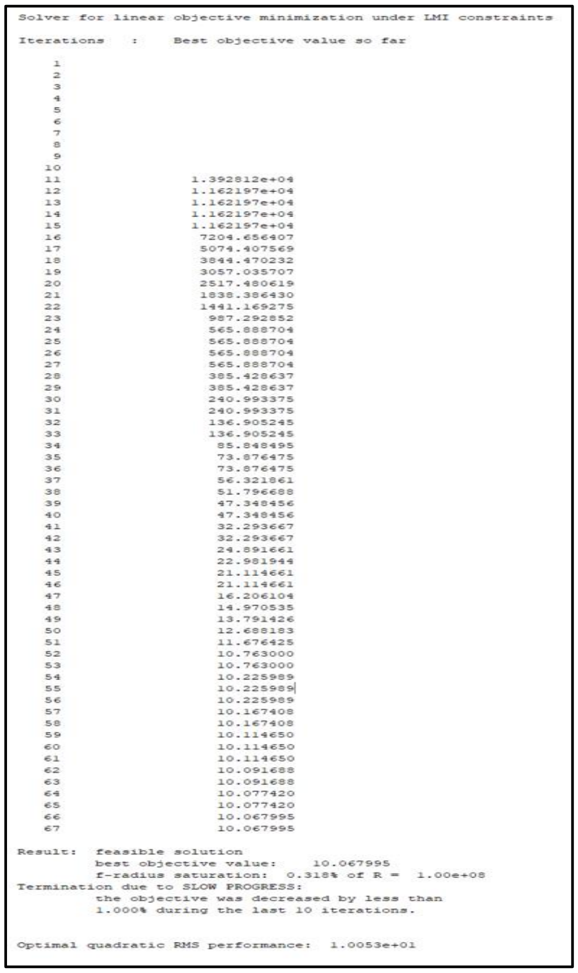


Figure 6.4 – High-level representation of controller matrices generation

After several iterations adjusting the weights and testing the controller against the LPV system using the functions *pdsimul* and *sigma*, the final LPV control system with eight vertex controllers is chosen, reflected in the code provided in Appendix A.1 and shown in the subsequent analysis plots in Section 6.4. The linear objective minimization feasibility problem under LMI constraints, discussed in Chapter 3, is completed using the function *hinfgs*, returning the quadratic performance $\gamma = 10.068$ for the closed-loop system.



```
Solver for linear objective minimization under LMI constraints

Iterations   :    Best objective value so far

     1
     2
     3
     4
     5
     6
     7
     8
     9
    10
    11               1.392812e+04
    12               1.162197e+04
    13               1.162197e+04
    14               1.162197e+04
    15               1.162197e+04
    16                7204.656407
    17                5074.407569
    18                3844.470232
    19                3057.035707
    20                2517.480619
    21                1838.386430
    22                1441.169275
    23                 987.292852
    24                 565.888704
    25                 565.888704
    26                 565.888704
    27                 565.888704
    28                 385.428637
    29                 385.428637
    30                 240.993375
    31                 240.993375
    32                 136.905245
    33                 136.905245
    34                  85.848495
    35                  73.876475
    36                  73.876475
    37                  56.321861
    38                  51.796688
    39                  47.348456
    40                  47.348456
    41                  32.293667
    42                  32.293667
    43                  24.891661
    44                  22.981944
    45                  21.114661
    46                  21.114661
    47                  16.206104
    48                  14.970535
    49                  13.791426
    50                  12.688183
    51                  11.676425
    52                  10.763000
    53                  10.763000
    54                  10.225989
    55                  10.225989
    56                  10.225989
    57                  10.167408
    58                  10.167408
    59                  10.114650
    60                  10.114650
    61                  10.114650
    62                  10.091688
    63                  10.091688
    64                  10.077420
    65                  10.077420
    66                  10.067995
    67                  10.067995

Result:  feasible solution
         best objective value:    10.067995
         f-radius saturation:  0.318% of R =  1.00e+08
Termination due to SLOW PROGRESS:
         the objective was decreased by less than
         1.000% during the last 10 iterations.


Optimal quadratic RMS performance:  1.0053e+01
```

Figure 6.5 – Closed-loop quadratic performance

A check on the location of the eigenvalues of the A_K controller matrix of each vertex controller to demonstrate the stability of the LPV controller is confirmed using the code provided in Appendix A.1. Note the LPV controller is a dynamic controller with a mix of shaping filters and $\mathcal{H}_\infty$ controllers.

6.4 Control Analysis – Assessment of Controller

6.4.1 Lyapunov Stability Analysis

For the closed-loop system determined from the generalized plant and the LPV controller, a Lyapunov matrix Q is sought such that

$$A_{cl}(\rho)Q + QA_{cl}(\rho)^T < 0 \tag{6.12}$$

for all values of the parameter vector ρ within the parameter box. If Q is found through an LMI optimization, the existence of a Lyapunov function $V(x)$,

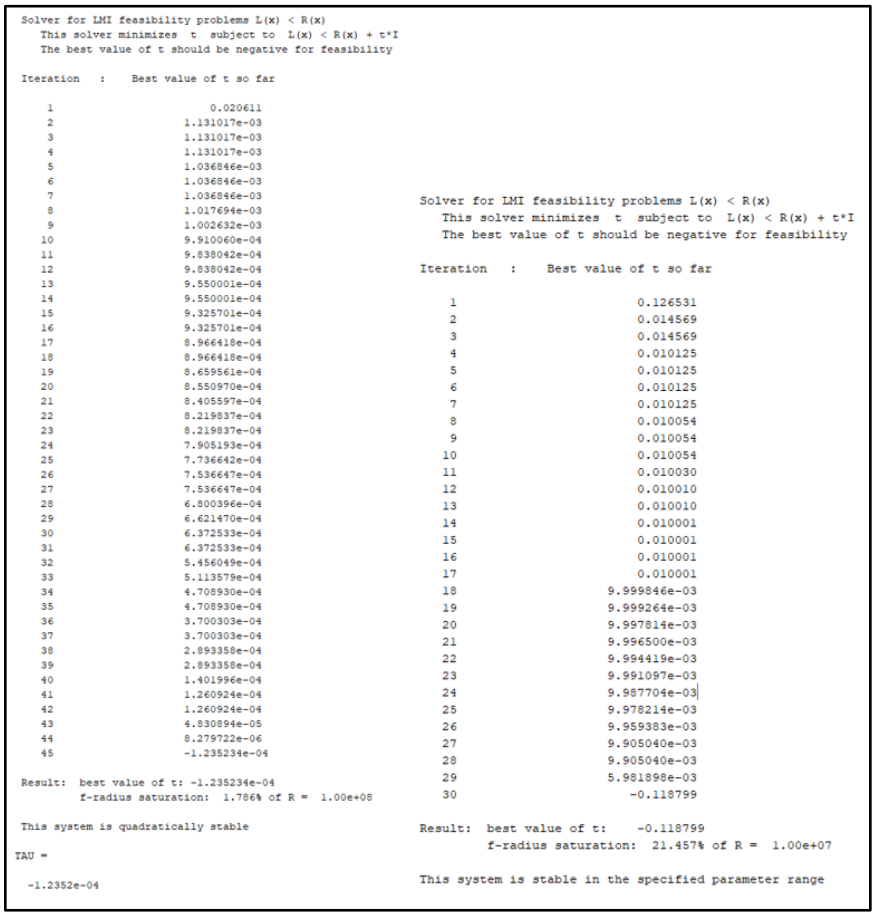
$$\begin{gathered} V(x) = x^T P x \\ P = Q^{-1} \\ \dot{V}\big(x(t)\big) < 0 \end{gathered} \tag{6.13}$$

establishes quadratic stability over the entire parameter range and for arbitrarily fast parameter variations [29]. This process is computed by the function *quadstab* with the result demonstrating quadratic stability for the closed system.

An additional stability check assesses robust stability of the closed-loop system via Lyapunov matrices Q_i at the vertices of the parameter box such that the Lyapunov function is of the form,

$$\begin{gathered} V(x, \alpha) = x^T (Q(\alpha))^{-1} x \\ Q(\alpha) = \alpha_1 Q_1 + \cdots + \alpha_q Q_q \end{gathered} \tag{6.14}$$

This establishes stability for the entire polytope of systems [29]. The results from *quadstab* and *pdlstab* shown in Figure 6.6 demonstrates robust stability for the closed-loop system.



```
Solver for LMI feasibility problems L(x) < R(x)
   This solver minimizes  t  subject to  L(x) < R(x) + t*I
   The best value of t should be negative for feasibility

Iteration   :    Best value of t so far

    1                    0.020611
    2                1.131017e-03
    3                1.131017e-03
    4                1.131017e-03
    5                1.036846e-03
    6                1.036846e-03
    7                1.036846e-03
    8                1.017694e-03
    9                1.002632e-03
   10                9.910060e-04
   11                9.838042e-04
   12                9.838042e-04
   13                9.550001e-04
   14                9.550001e-04
   15                9.325701e-04
   16                9.325701e-04
   17                8.966418e-04
   18                8.966418e-04
   19                8.659561e-04
   20                8.550970e-04
   21                8.405597e-04
   22                8.219837e-04
   23                8.219837e-04
   24                7.905193e-04
   25                7.736642e-04
   26                7.536647e-04
   27                7.536647e-04
   28                6.800396e-04
   29                6.621470e-04
   30                6.372533e-04
   31                6.372533e-04
   32                5.456049e-04
   33                5.113579e-04
   34                4.708930e-04
   35                4.708930e-04
   36                3.700303e-04
   37                3.700303e-04
   38                2.893358e-04
   39                2.893358e-04
   40                1.401996e-04
   41                1.260924e-04
   42                1.260924e-04
   43                4.830894e-05
   44                8.279722e-06
   45               -1.235234e-04

Result:  best value of t: -1.235234e-04
         f-radius saturation:  1.786% of R =  1.00e+08

This system is quadratically stable

TAU =

  -1.2352e-04
```

```
Solver for LMI feasibility problems L(x) < R(x)
   This solver minimizes  t  subject to  L(x) < R(x) + t*I
   The best value of t should be negative for feasibility

Iteration   :    Best value of t so far

    1                    0.126531
    2                    0.014569
    3                    0.014569
    4                    0.010125
    5                    0.010125
    6                    0.010125
    7                    0.010125
    8                    0.010054
    9                    0.010054
   10                    0.010054
   11                    0.010030
   12                    0.010010
   13                    0.010010
   14                    0.010001
   15                    0.010001
   16                    0.010001
   17                    0.010001
   18                9.999846e-03
   19                9.999264e-03
   20                9.997814e-03
   21                9.996500e-03
   22                9.994419e-03
   23                9.991097e-03
   24                9.987704e-03
   25                9.978214e-03
   26                9.959383e-03
   27                9.905040e-03
   28                9.905040e-03
   29                5.981898e-03
   30                   -0.118799

Result:  best value of t:    -0.118799
         f-radius saturation:  21.457% of R =  1.00e+07

This system is stable in the specified parameter range
```

Figure 6.6 – Quadratic and robust stability results

6.4.2 Time Domain Analysis

To assess the controller, a random set of polytopic coordinates are generated to evaluate controller and closed-loop system. The function *pdsimul* is used to generate output and state trajectories for the specified parameter trajectory shown in Appendix A.1.

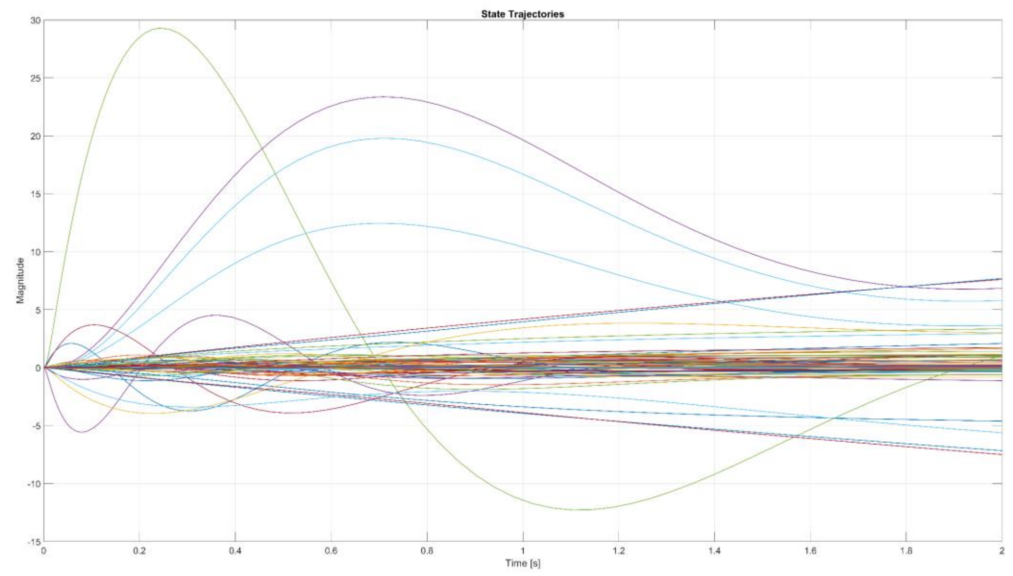


Figure 6.7 – State trajectories of closed-loop system for sample trajectory

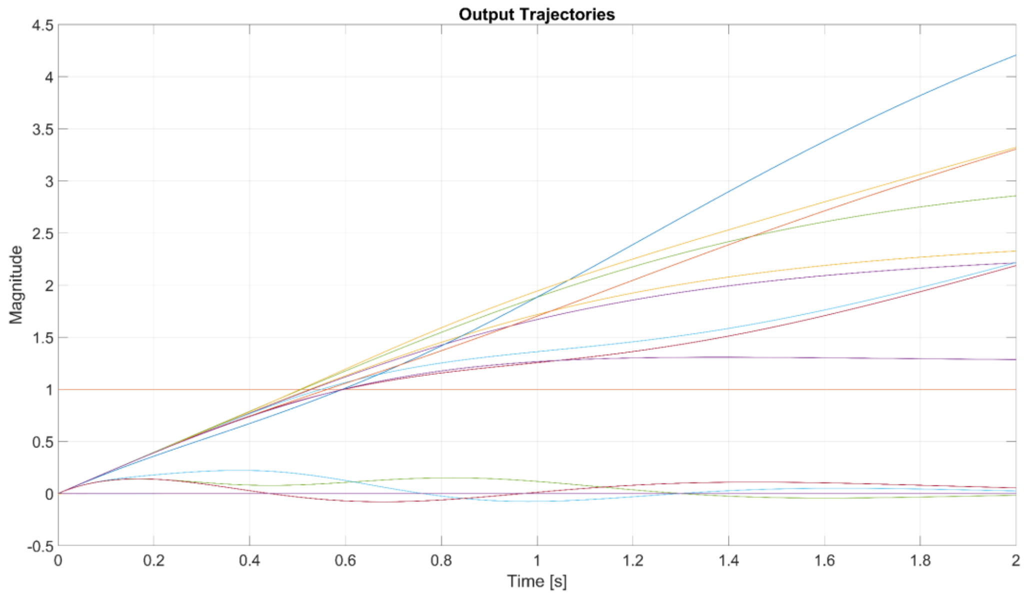


Figure 6.8 – Output trajectories of closed-loop system for sample trajectory

6.4.3 Frequency Domain Analysis – Singular Values

The function *sigma* plots of the maximum and minimum singular values, $\overline{\sigma}(j\omega)$ and $\underline{\sigma}(j\omega)$, at the specified frequency range. These plots show the MIMO frequency responses in terms of singular values and the function returns these values as a vector. Singular values give a measure of how a system acts on an input at a particular frequency. Therefore, $\overline{\sigma}$ and $\underline{\sigma}$ can be analyzed to determine amplification and attenuation of input signals acting on the system [37].

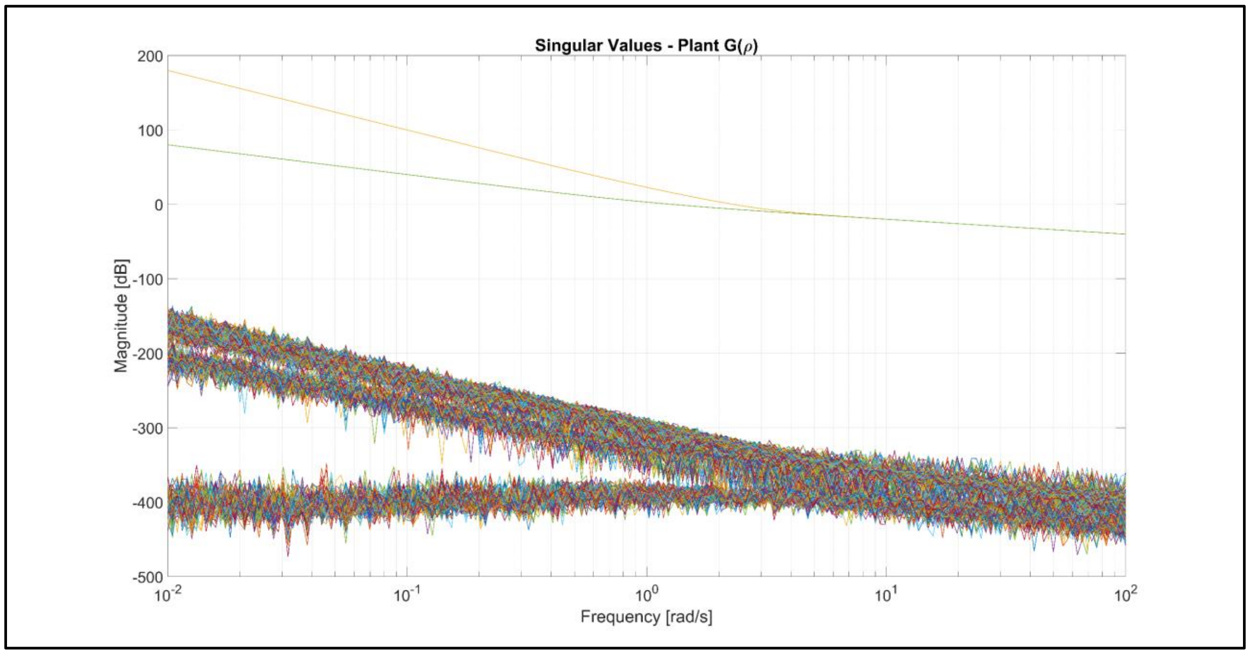


Figure 6.9 – Singular values of polytopic plant $G(\rho)$

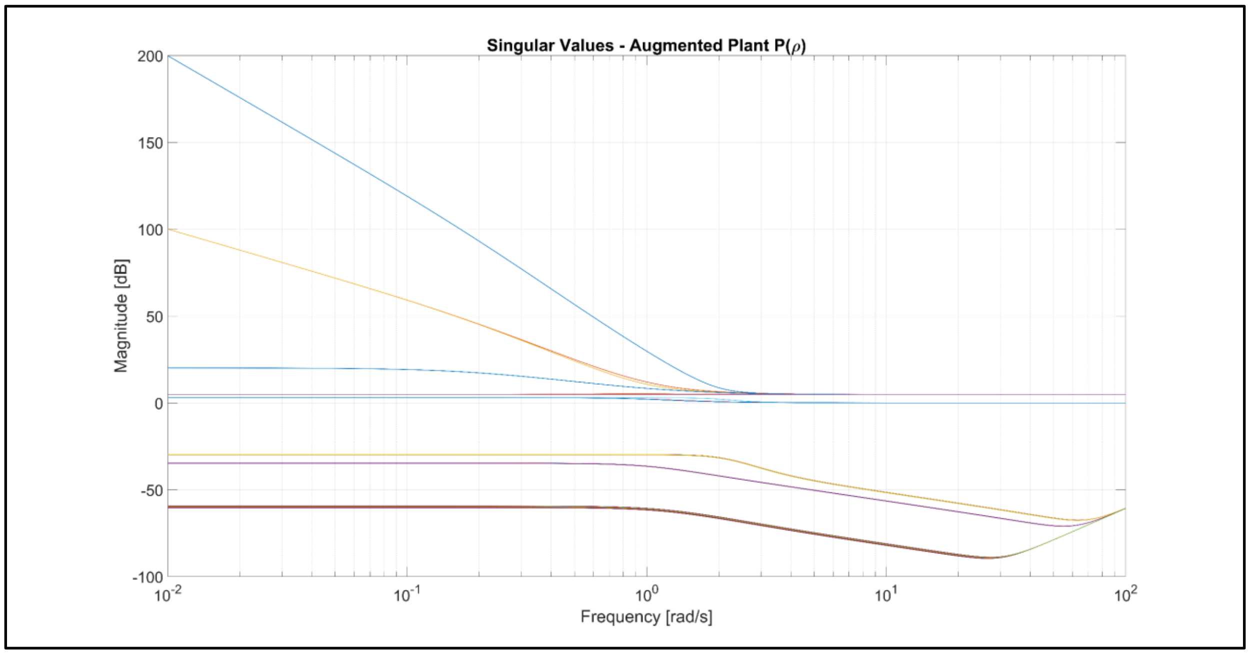


Figure 6.10 – Singular values of augemented $\mathcal{H}_\infty$ plant P(ρ)

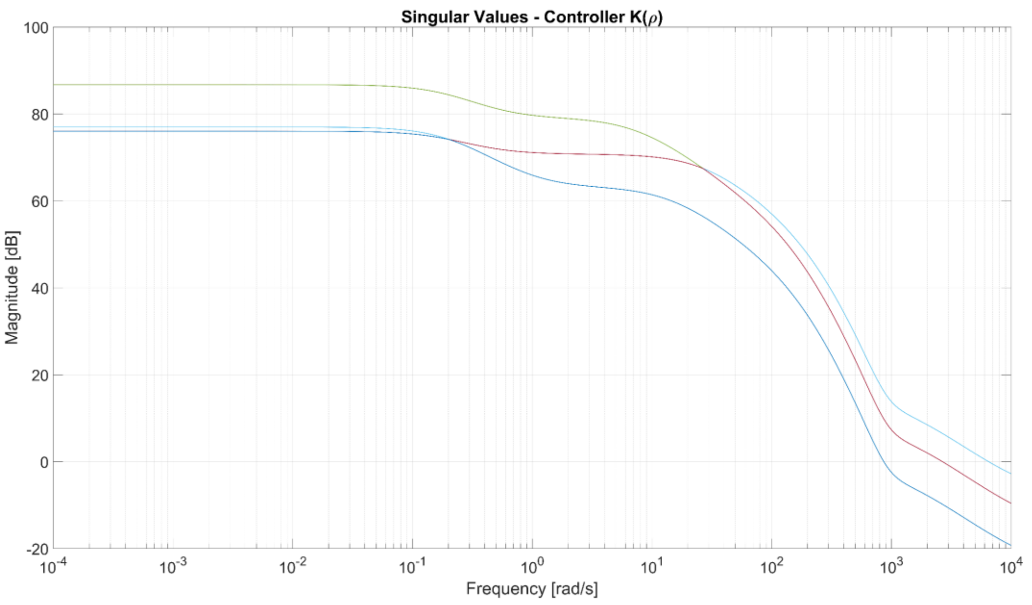


Figure 6.11 – Singular values of polytopic LPV controller $K(\rho)$

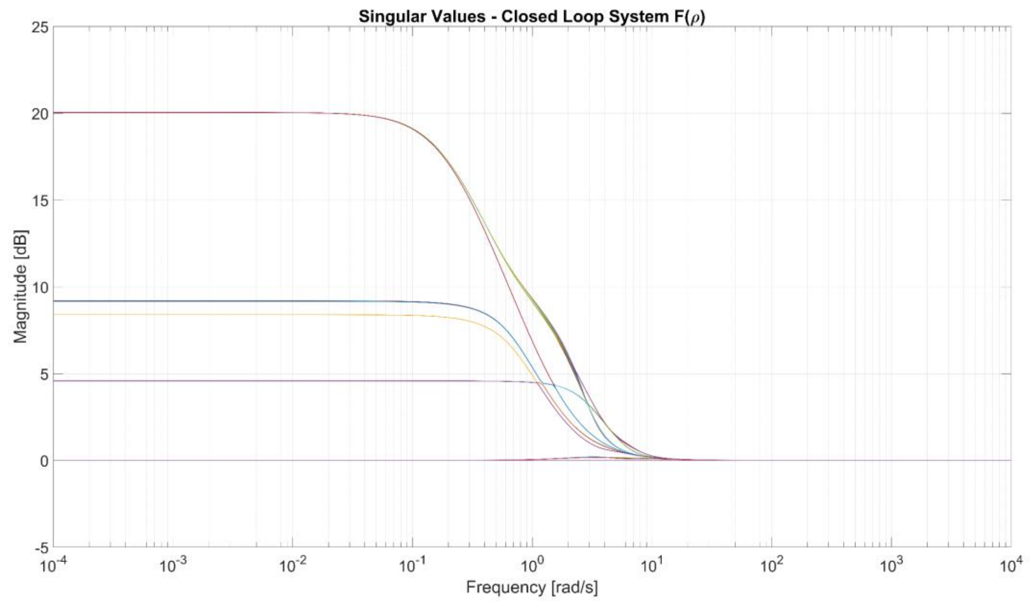


Figure 6.12 – Singular values of polytopic closed-loop system $F(\rho)$

6.5 Linear Simulation

6.5.1 LPV Controller Implementation in Simulink

The Simulink implementation of the gain-scheduled controller is shown in Figure 6.13 and Figure 6.14. The convex decomposition and the computation of controller matrices $\{A_K(\rho),$ $B_K(\rho), C_K(\rho), D_K(\rho)\}$ are determined online by the functions *polydec*, *psinfo,* and *ltiss* through the Simulink MATLAB functions *ConvexDecomp* and *VertexControl* whose code is provided in Appendix A.6.

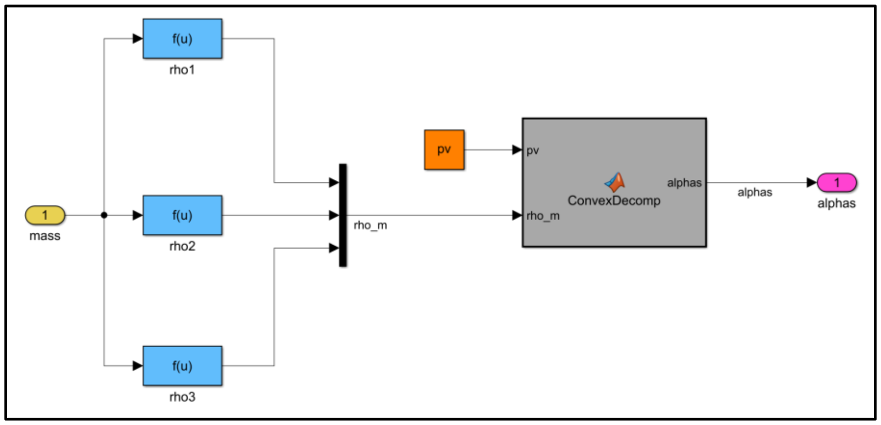


Figure 6.13 – Convex decomposition determined online

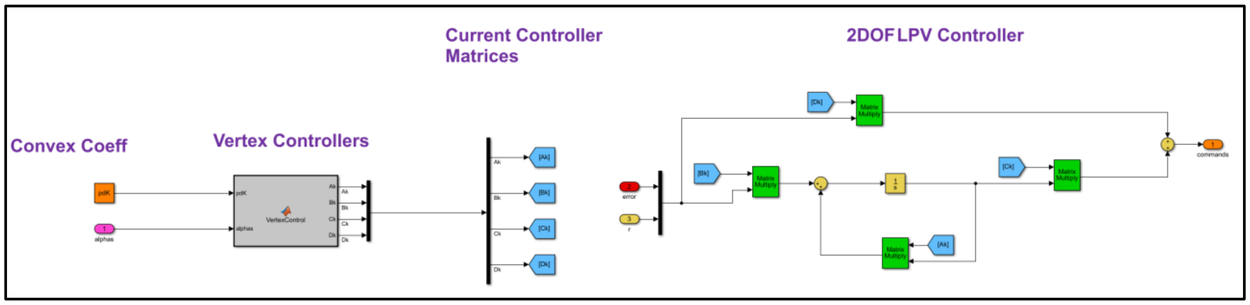


Figure 6.14 – Simulink implementation of LPV controller

With the gain-scheduled Simulink implementation, the LPV controller is tested against the linear system and the complete nonlinear system with disturbance sources added. The latter is the subject of Chapter 8.

6.5.2 Reference Trajectory

For the purpose of testing the tracking quality of the LPV controller, a reference trajectory is designed. A sample mission scenario where the quadrotor takes off at initial point [0 0 0], picking up a payload of an unknown mass, and delivers to final position [1000 1000 40] is considered. This scenario is illustrated in Figure 6.15 where the final position can be visualized as the top of an office or residential building.

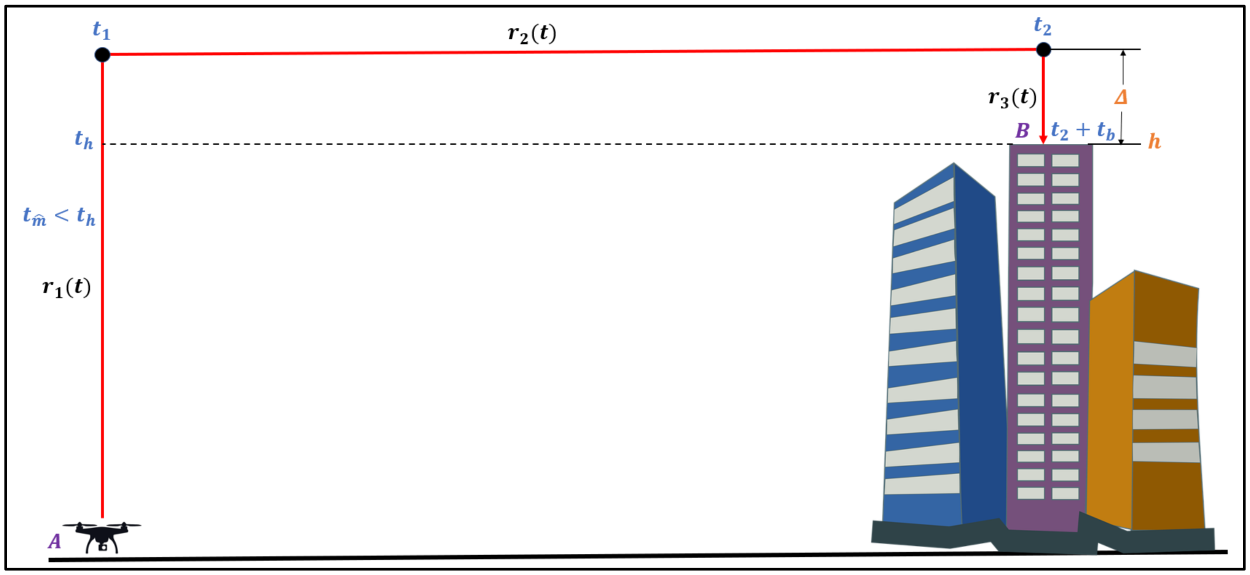


Figure 6.15 – Reference trajectory schematic

The parameters shown in the schematic are summarized in Table 6.1. The time to estimate the mass $t_{\hat{m}}$ is designed such that it is less than the time it takes the quadrotor to reach h. The geometry between point A and point B is setup as shown Figure 6.16. This is used to determine the horizontal and vertical components of the velocity.

Table 6.1 – Summary of trajectory parameters

Parameter	Description	Value
h	Height of building	$40\ m$
Δ	Height of quadrotor above h	$10\ m$
v_p	Lifting velocity	$10\ m/s$
v_m	Cruise velocity	$40\ m/s$
v_h	Landing velocity	$5\ m/s$

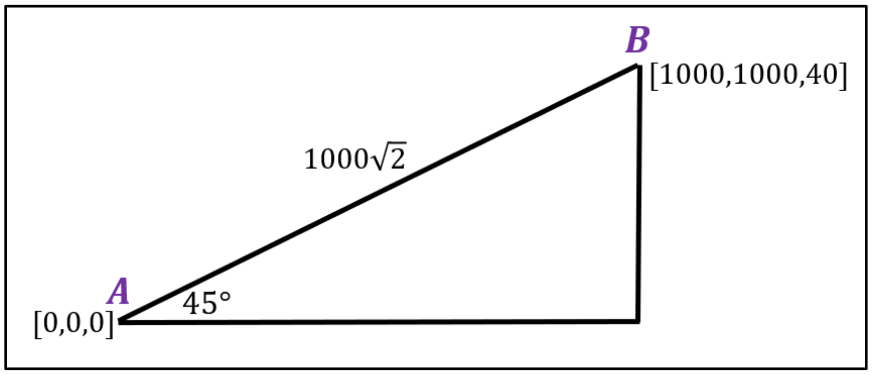


Figure 6.16 – Geometry of reference path

To ensure a safe transition from cruise into the landing at the final position, a linear decrease in the cruise velocity to w at the final position is setup according to Figure 6.17, where t_b is the desired time from $h + \Delta$ to the final position from time t_2.

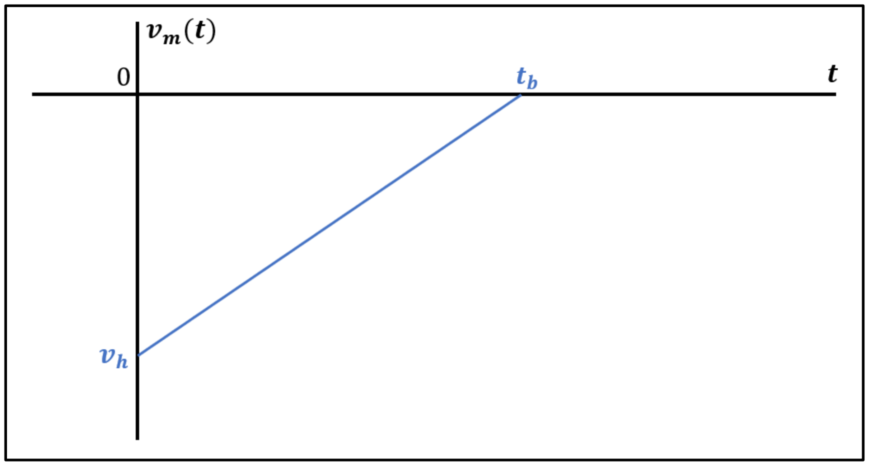


Figure 6.17 – Linear decrease of velocity to 0 at landing position

The state trajectory $\boldsymbol{z(t)}$ is split up into paths $\boldsymbol{z_o(t), z_1(t), z_2(t), z_3(t), z_4(t)}$ as piecewise linear functions. The reference states for the Euler angles and attitude rates are taken to be $\boldsymbol{0}$. For each path, the reference trajectory $z_r(t)$ is defined. The derivation for each path is detailed in Table 6.2.

$$z_r(t) = [X_r \quad Y_r \quad Z_r \quad u_r \quad v_r \quad w_r \quad \varphi_r \quad \theta_r \quad \psi_r \quad p_r \quad q_r \quad r_r]^T \qquad (6.15)$$

Table 6.2 – Summary of reference paths

$\boldsymbol{z_o(t)}$ path – initial position

Parameters: $h = 40\ m$, $\Delta = 10\ m$
State Equations:

$$X = Y = Z = 0$$
$$u = v = w = 0$$

$\boldsymbol{z_1(t)}$ path

Parameters: $V_p = 10\ m/s$, $t_1 = \frac{h+\Delta}{V_p} = 5\ s$, $t_h = \frac{h}{V_p} = 4\ s$
State Equations:

$$u = v = 0$$
$$w = V_p$$
$$X = Y = 0$$
$$Z = V_p t$$

$\boldsymbol{z_2(t)}$ path

Parameters: $V_m = 40\ m/s$, $\theta = 45°$, $t_2 = t_1 + \frac{1000\sqrt{2}}{V_m} = 40.4\ s$
State Equations:

$$Z = h + \Delta \rightarrow w = 0$$
$$u = V_m cos\theta = V_m/\sqrt{2}$$
$$v = V_m sin\theta = V_m/\sqrt{2}$$
$$X = Y = \int_{t_1}^{t} V_m(t)dt$$
$$\Rightarrow\ X = Y = \frac{V_m}{\sqrt{2}}(t - t_1)$$

$\boldsymbol{z_3(t)}$ path

Parameters: $t_b = 4\ s$
$-\Delta = \int_{t_a=0}^{t_b}\left(-v_h + \frac{v_h}{t_b}t\right)dt \rightarrow v_h = \frac{2\Delta}{t_b} = 5\ m/s$
State Equations:

$$X = Y = 1000 \rightarrow u = v = 0$$
$$w = -v_h + \frac{v_h}{t_b}(t - t_2)$$
$$\Rightarrow\ Z = \int_{t_2}^{t}\left[\left(-v_h + \frac{v_h}{t_b}(t - t_2)\right)\right]dt,\ Z(t_2) = h + \Delta$$

$$Z = -v_h(t - t_2) + \frac{v_h}{2t_b}(t - t_2)^2 + (h + \Delta)$$

$z_4(t)$ path – final position

Parameters: $t_f = t_2 + t_b = 45.4\ s$
State Equations:

$$X = Y = 1000$$
$$Z = 40$$
$$u = v = w = 0$$

The implementation of the reference trajectory is generated by the Simulink MATLAB function *RefTraj* provided in Appendix A.5 and shown in Figure 6.18.

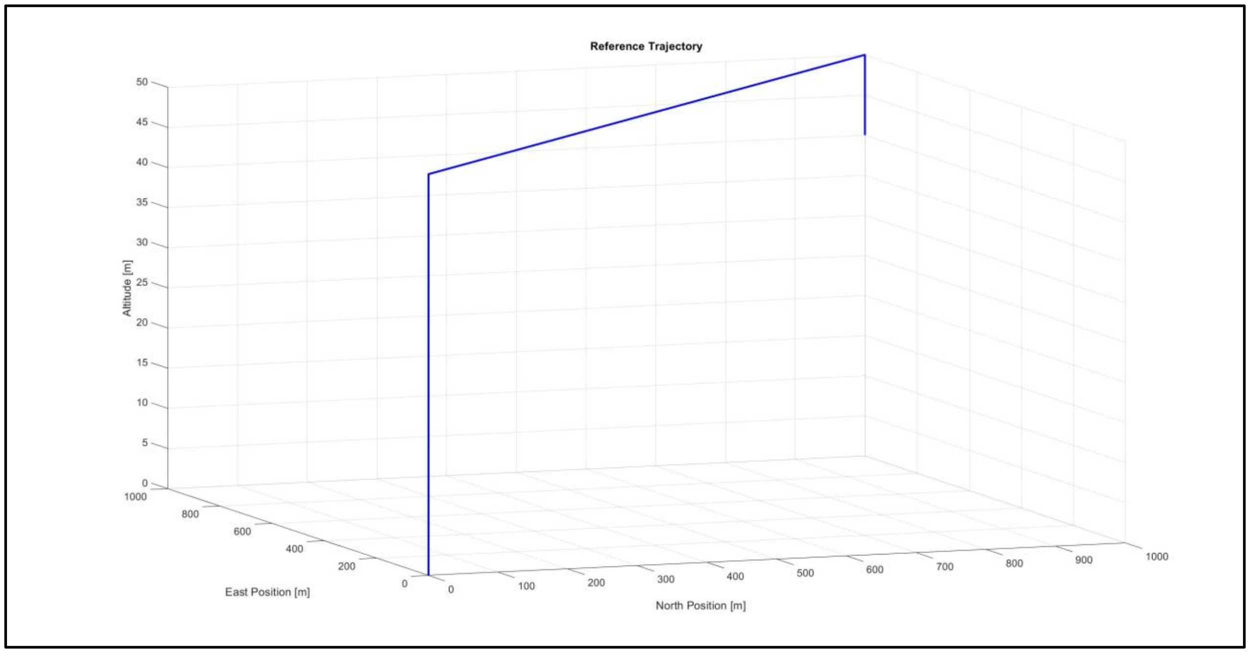


Figure 6.18 – 3D reference trajectory to assess controller

6.5.3 Simulation

For an initial test, the simplest setup is considered where the mass is known and fed directly into the LPV system and simulated against the linear system given by (2.15) evaluated at $m_p = 2$. Details on how to integrate the hover state conditioning and mass estimator subsystems, with modifications from the Chapter 4 design, to control the complete nonlinear system is detailed in Chapter 8.

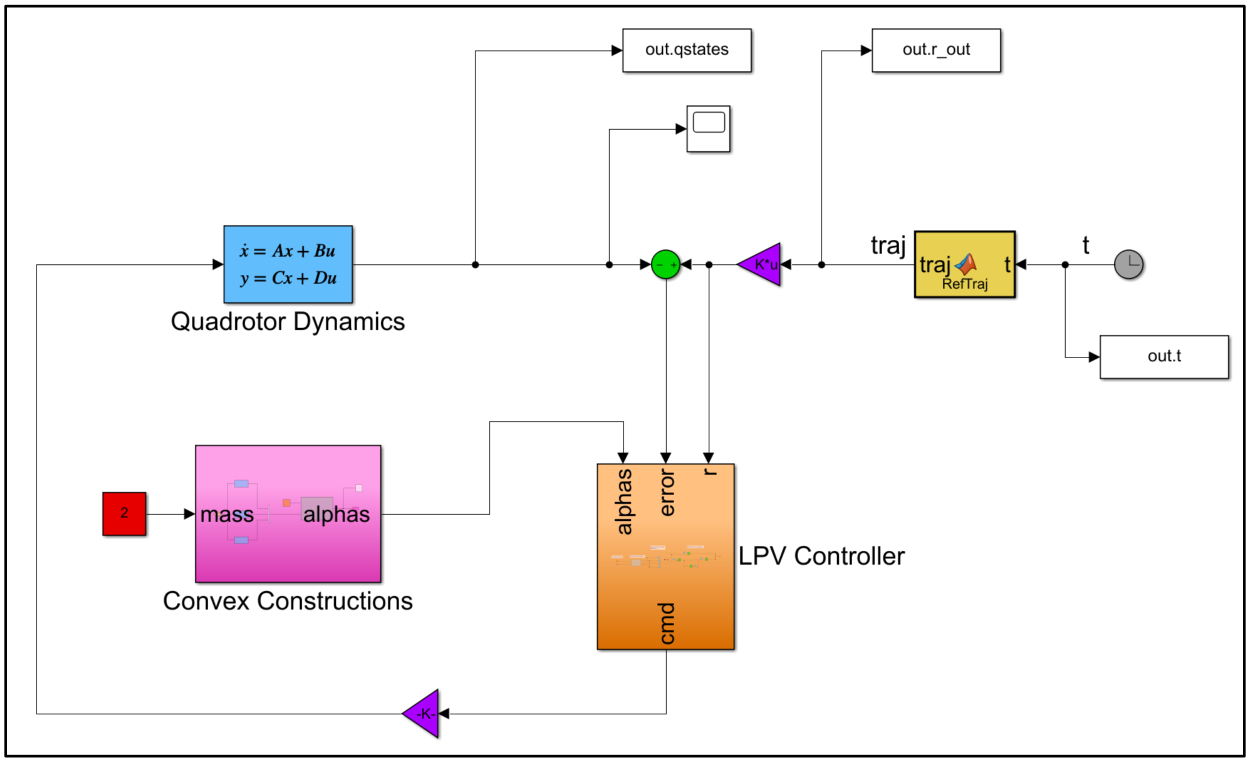


Figure 6.19 – Simulink setup for linear simulation

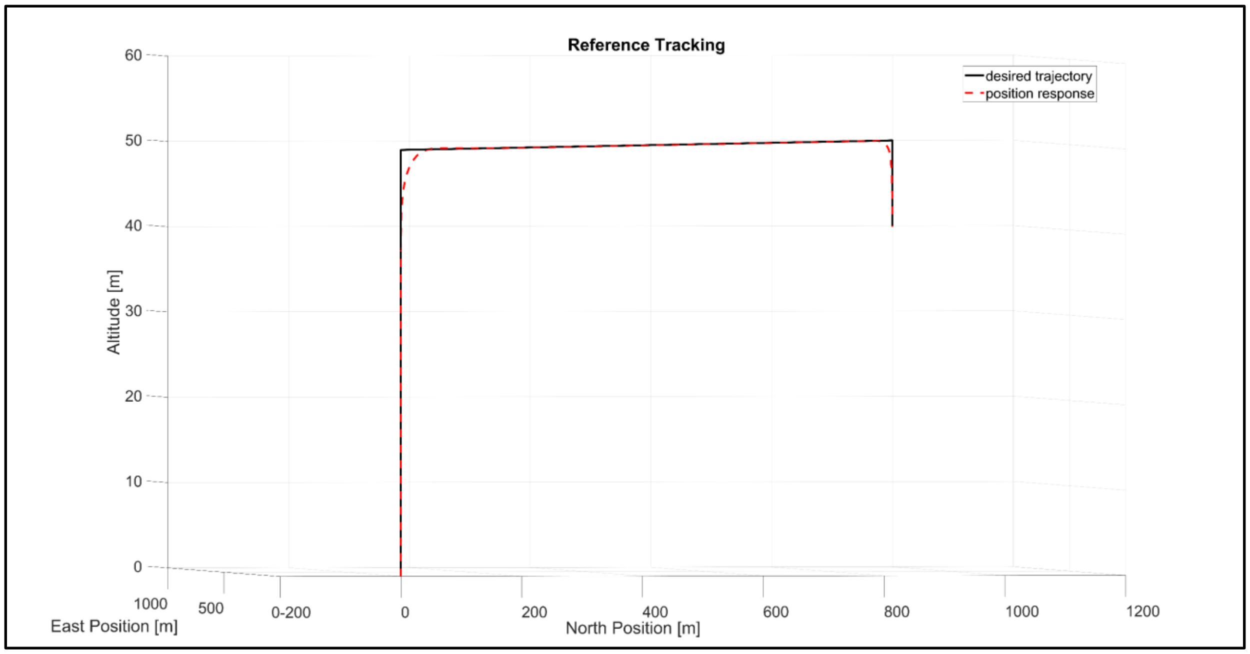


Figure 6.20 – Reference tracking for linear simulation

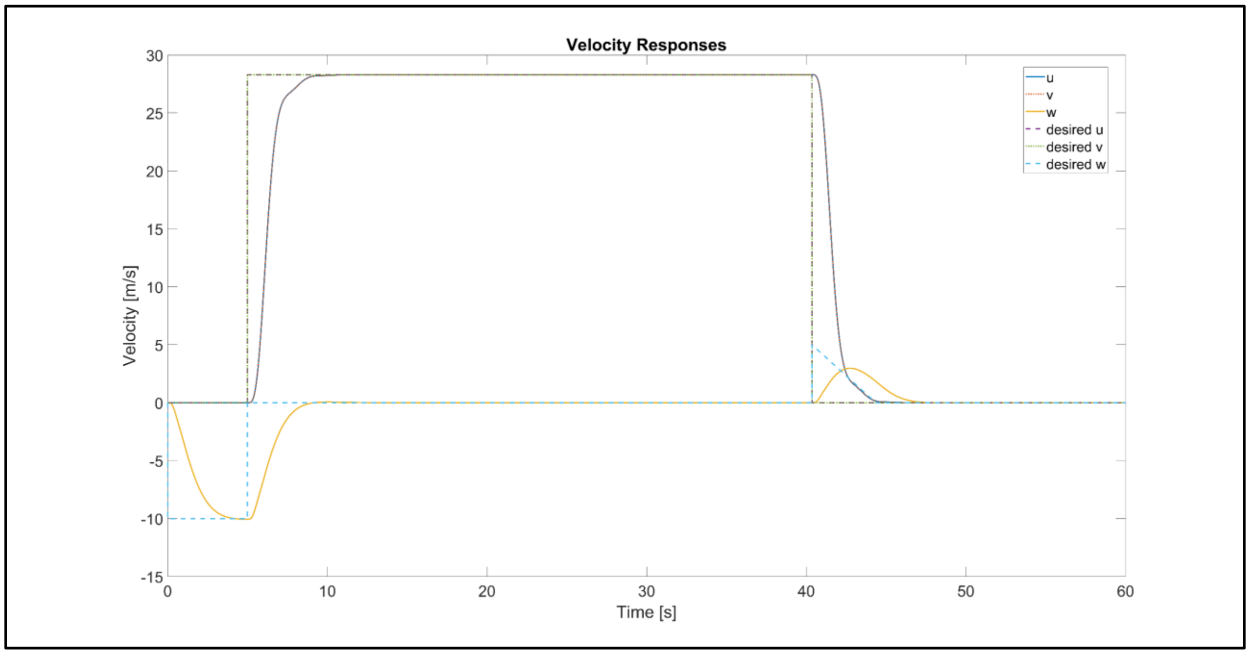


Figure 6.21 – Velocity tracking for linear simulation

6.5.4 Discussion

The LPV controller was able to track the position and velocity including the reference states for the attitude. In simulation, it was found the output of the reference trajectory and LPV commands needed to be scaled by two gains for good tracking performance. After testing several LPV controllers with different control weights, there were no significant changes in the tracking quality of the LPV controller in the linear simulation. The controller was able to handle higher demanded accelerations when tested against the linear system. However, this was an issue for control of the nonlinear system, as seen in Chapter 8, limitations are imposed on the controller and proper selection of the weights is necessary to achieve good performance.

7. CHAPTER 7

ACTUATOR COMPENSTATION

7.1 Introduction

As discussed in the literature review, battery drainage affects the control effectiveness of the propeller speeds which drive the quadrotor dynamics. It is possible to apply the LPV methodology and schedule gains based on the propeller speeds, but with added complexity and expansion of the parameter space. However, for the purpose of regulation, without considering fault tolerant control, it is found that a 2DOF PI controller will be able to regulate the propeller speeds subject to changes to the input voltage due to battery drainage. Many modeling, estimation, and control challenges are involved with battery power management, motor dynamics, and control beyond the scope of the project, therefore only the design aspect of regulation assuming a simple motor model is considered so that actuator compensation is integrated into the control system.

7.2 Updated Actuator Dynamics Model

Each motor is modeled as a system with an input voltage and propeller speed output, with a proportional gain and time constant representative of the gain and time delay effects of the actuator expressed as the transfer function (7.1). Physical modeling of motors is far more complex, but this study considers design of a linear controller for the simplest model to demonstrate the function of the compensator and to provide completeness of the control system.

$$\frac{\Omega(s)}{V_{in}(s)} = G(s) = \frac{c_m}{\tau s + 1} \tag{7.1}$$

From previous designs, a mechanical time constant of $\tau = 1.28\ ms$ and motor constant $c_m = 20\ s^{-1}$ are chosen.

7.3 2-DOF PI Control Design

The objective of the controller is to maintain $\Omega_c = \Omega$ subject to changes in the input voltage V_{in} with the fastest response possible without overshoot, i.e. critically damped with damping ratio $\zeta = 1$. Recall Figure 2.3 where the commanded propeller speeds are produced by a motor mixing block whose inputs are the desired forces and torques from the main control law. The PI controller is taken from reference [40], a set of notes on adaptive control of a DC motor.

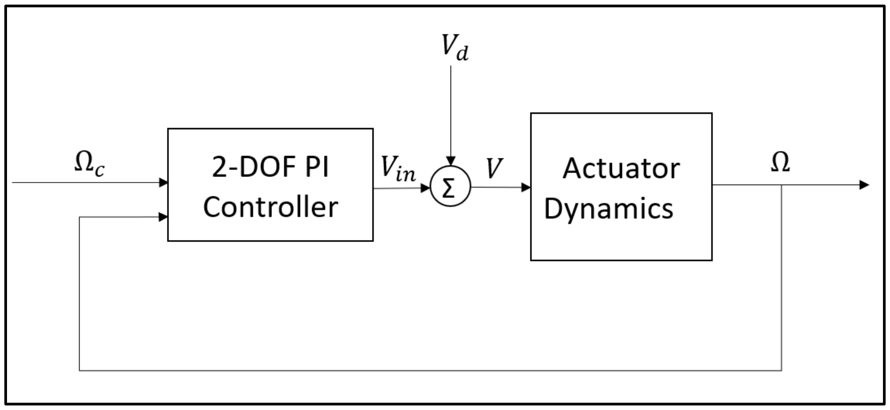


Figure 7.1 – Simplified diagram for propeller speed regulation

A simplified diagram is shown in Figure 7.1 where V_d models an additive change in the input voltage due to battery drainage.

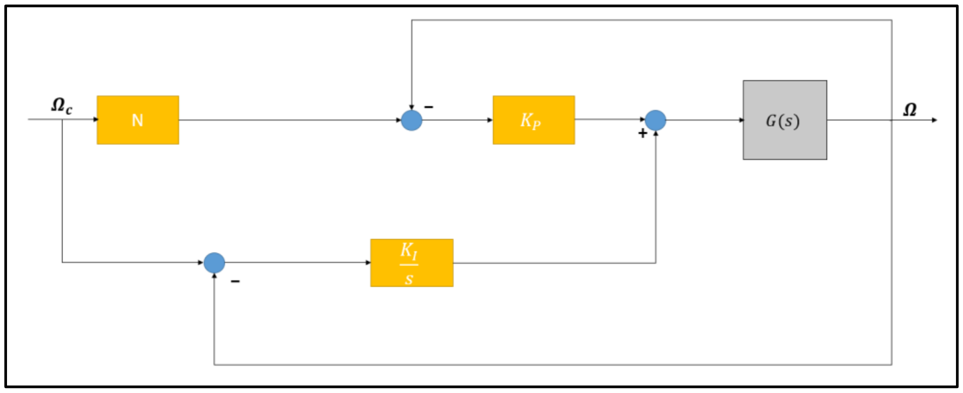


Figure 7.2 – 2DOF PI controller

The closed-loop transfer function of Figure 7.2 from the commanded propeller speed Ω_c to the real propeller speed Ω is given by (7.2).

$$T(s) = \frac{c_m(K_P N s + K_I)}{\tau s^2 + (1 + c_m K_P)s + c_m K_I} \tag{7.2}$$

Derivation of Closed-Loop Transfer Function

Let $r = \Omega_c$ and $y = \Omega$

From the block diagram relationships,

$$P_1(s) = \frac{K_I}{s}(\Omega_c(s) - \Omega(s))$$
$$P_2(s) = K_p(N\Omega_c(s) - \Omega(s))$$

Adding the two results,

$$V_{in}(s) = P_1(s) + P_2(s)$$

Let $U(s) = V_{in}(s)$ and $Y(s) = \Omega(s)$

$$Y(s) = G(s)U(s)$$
$$\Omega(s) = G(s)\left(K_P N\Omega_c(s) - K_P\Omega(s) + \frac{K_I}{s}\Omega_c(s) - \frac{K_I}{s}\Omega(s)\right)$$

Rearranging so that $\Omega(s)$ and $\Omega_c(s)$ are on the left- and right-hand sides of the equation,

$$\Omega(s)[s + G(s)K_P s + G(s)K_I] = \Omega_c(s)\big[G(s)K_p N s + G(s)K_I\big]$$

The closed-loop transfer function from Ω_c to Ω is then,

$$T(s) = \frac{\Omega(s)}{\Omega_c(s)} = \frac{G(s)K_p N s + G(s)K_I}{s + G(s)K_P s + G(s)K_I}$$

Substituting $G(s) = \frac{c_m}{\tau s+1}$ and simplifying yields the transfer function (7.2) ,

$$T(s) = \frac{c_m(K_P N s + K_I)}{\tau s^2 + (1 + c_m K_P)s + c_m K_I}$$

Comparing $T(s)$ to a general second-order system,

$$T(s) = \frac{{\omega_n}^2}{s^2 + 2\zeta\omega_n s + {\omega_n}^2} \tag{7.3}$$

and choosing poles at $-20, -25$ to determine the natural frequency ω_n results in the gains (7.4)

$$K_P = \frac{2\zeta\omega_n\tau}{c_m}$$

$$K_I = \frac{{\omega_n}^2\tau}{c_m} \tag{7.4}$$

The gains K_P, K_I are starting gains to be tuned in simulation along with the feedforward gain N.

7.4 Simulation

The PI controller is implemented in Simulink, shown in Figure 7.3, to determine the tuned gains that achieve fast response while tracking the reference Ω_c subject to changes in V_{in}.

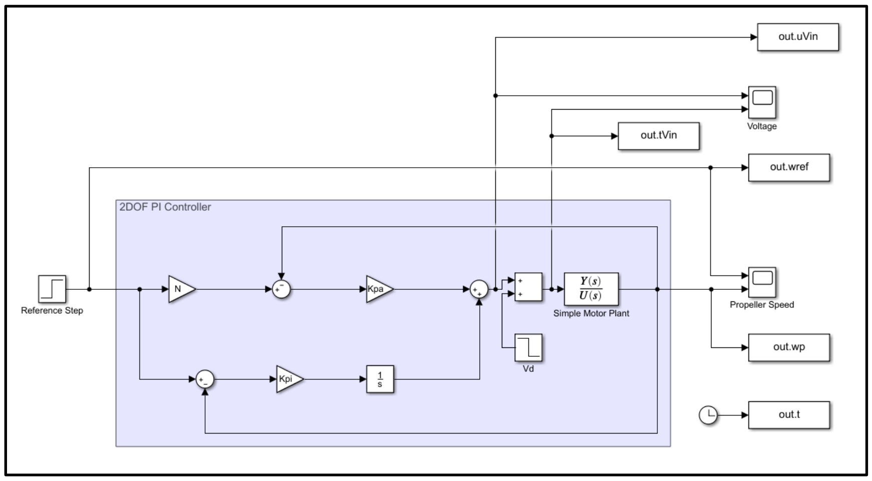


Figure 7.3 – Simulink setup for PI controller

A reference $\Omega_c = 200\ rad/s$ and voltage parameters of $V_d = 0\ V$ and $V_d = -5\ V$ were set for the simulation to test the controller with and without a voltage change. The controller gains were adjusted until the propeller speed response shown in Figure 7.4 was achieved. This is reflected in the Appendix A.3 code. The controller was able to track the reference propeller speed. Note with no change in the voltage input, the rise time of the response is small with a slight overshoot and no steady state error. Adding the voltage change causes a longer rise time, but the controller still tracks the reference.

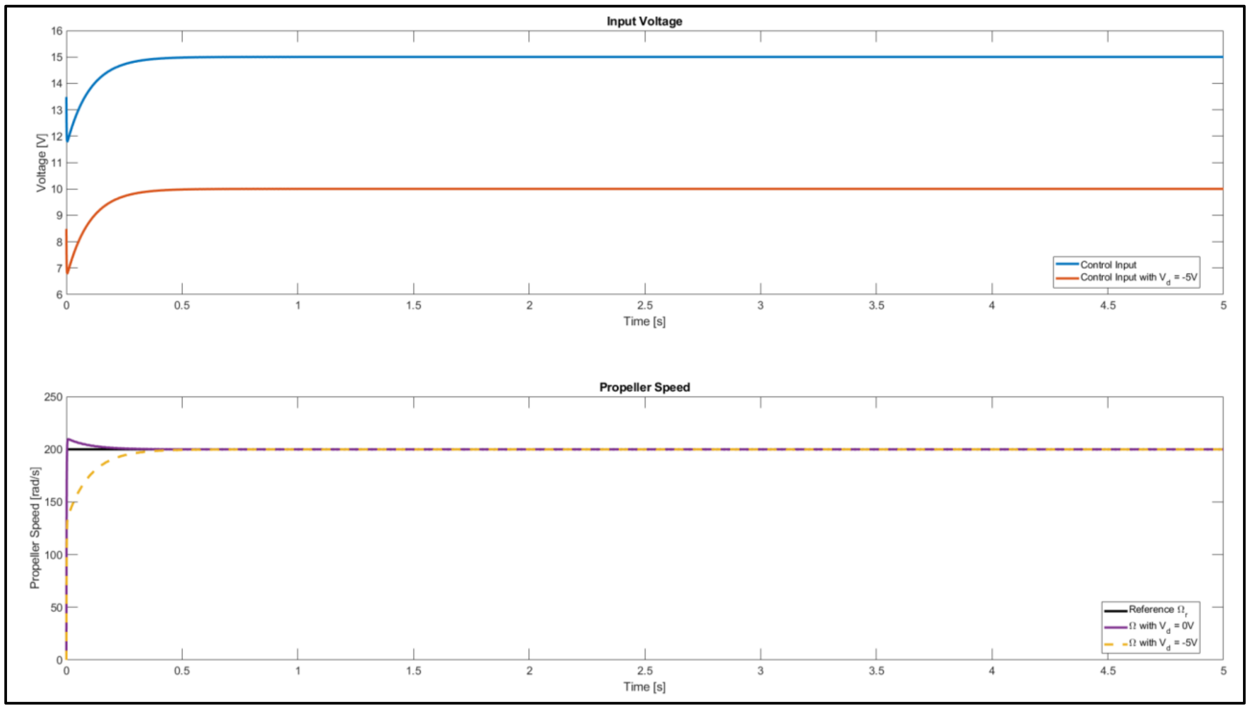


Figure 7.4 – Propeller speed response subject to voltage change

The PI controller shown in Figure 7.3 is applied to each of the four commanded propeller speeds and augmented to the LPV control system. These are part of the nonlinear simulations described in Chapter 9.

8. CHAPTER 8

LPV CONTROL OF NONLINEAR SYSTEM

8.1 System Modifications

LPV control against the nonlinear system expressed by Table 2.3 and the actuator dynamics system described in Chapter 7 was not a straightforward process. Although the LPV controller developed in Chapter 6 had no issues controlling the linear system, in nonlinear simulation, it was found the controller developed could not stabilize the nonlinear system without introducing modifications. Therefore, this chapter describes the modifications added to the control system and their efficacy in controlling the nonlinear system. These modifications are a modification of the LPV commands to account for the equilibrium point, a reference model signal to filter the reference trajectory, and a modification of the hover state conditioning system so that only the LPV commands are used for the control in contrast to the design in Chapter 4 where the commands were switched to a direct input to estimate the mass.

With modifications applied, the LPV controller had issues controlling the system using the desired forces and torques as inputs into the motor mixing and actuator dynamics blocks shown in Figure 2.5. However, applying these inputs directly into the quadrotor dynamics block, neglecting the actuator dynamics, resulted in stabilization and control of the nonlinear system. A propeller speed based LPV controller is developed to control the complete system with actuator dynamics. Therefore, the nonlinear simulations were completed in three steps:

- Nonlinear simulation with the control inputs as the desired forces and torques, not including actuator dynamics.
- Nonlinear simulation with actuator dynamics using an updated LPV controller derived using an LPV model with the propeller speeds as the control inputs.
- Nonlinear simulation with disturbances added to the actuator system and velocity states.

The objective of the control system is to estimate the unknown mass online, determine the controller gains automatically, and track the reference trajectory developed from Chapter 6 subject to disturbances.

8.1.1 Control Conditioning

Since the system was linearized with respect to $\boldsymbol{u_e}$, the control commands $\boldsymbol{u_d}$ from the LPV controller can be applied directly to control linear system. However, to control the linear system, these commands are conditioned to account for the offset caused by the equilibrium point. The new control input $\boldsymbol{u}$ is found by scaling the LPV commands by k_u and adding the equilibrium point used in the linearization.

$$\begin{gathered}\boldsymbol{u} = \boldsymbol{u_e} + \boldsymbol{k_u u_d} \\ u_e = \left[(\hat{m}_p + m_{base})g \quad 0 \quad 0 \quad 0\right]^T \\ u_d = [F_z \quad \tau_\varphi \quad \tau_\theta \quad \tau_\psi]_{des}{}^T\end{gathered} \tag{8.1}$$

Note $\hat{m}_p$ is taken from the mass estimator and k_u was determined in simulation. The hover enable signal is applied for the first second of flight. Within this time, the mass estimator is active producing an estimate at every sample, the LPV commands adjust accordingly until the estimate converges to the actual m_p.

The hover state conditioning system is also modified so that only the LPV commands are used for control. Note the hover enable signal can be modified to enable the estimation during multiple points in the flight path. But for the purposes of simulation, the hover enable signal is applied only at launch to demonstrate the efficacy of the control system. These modifications are shown in Figure 8.1 and implemented in the Simulink structures in Appendix B.4 and B.5.

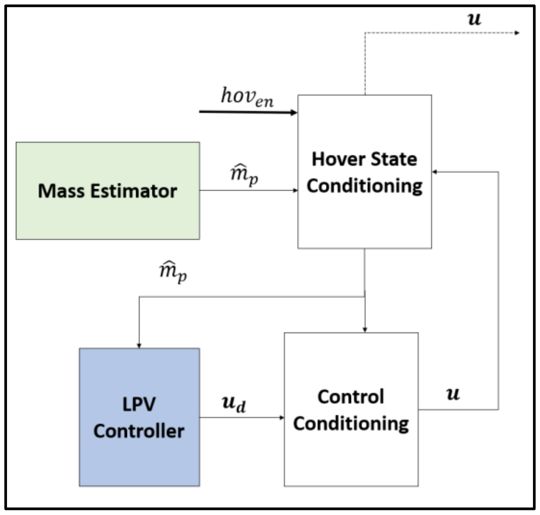


Figure 8.1 – Modifications of LPV controller

8.1.2 Model Reference Signal

In addition to conditioning the control, the reference trajectory is scaled and filtered before being fed into the LPV controller. The filter gradually ramps up the velocity demands from the reference trajectory to prevent instabilities from occurring in the system response due to inputs exceeding a stable limit. This modification was not necessary to control the linear system. The model reference signal (8.2) is proposed for the filter, applied after the scaling gain k_r.

$$\begin{gathered}\dot{\boldsymbol{x}}_{\boldsymbol{ref}} = A_{ref}\boldsymbol{x}_{\boldsymbol{ref}} + B_{ref}\boldsymbol{u}_{\boldsymbol{ref}} \\ \boldsymbol{y}_{\boldsymbol{ref}} = C_{ref}\boldsymbol{x}_{\boldsymbol{ref}} + D_{ref}\boldsymbol{u}_{\boldsymbol{ref}} \\ \\ A_{ref} = -\tau_{ref} * I_{12x12} \\ B_{ref} = I_{12x12} \\ C_{ref} = \tau_{ref} * I_{12x12} \\ D_{ref} = 0_{12x12}\end{gathered} \tag{8.2}$$

A time constant τ_{ref} is used to control the speed of response to each reference input. For the simulation in 8.1.3, $\tau_{ref} = 0.06$ is chosen after several simulations. The model can be expressed as a diagonal matrix of transfer function with $G_r(s) = \frac{3}{50s+3}$ applied to each reference input. Its frequency response is shown in the Bode plot in Figure 8.2. The model reference limits the inputs from the LPV controller, so they do not exceed force demands that cause instability.

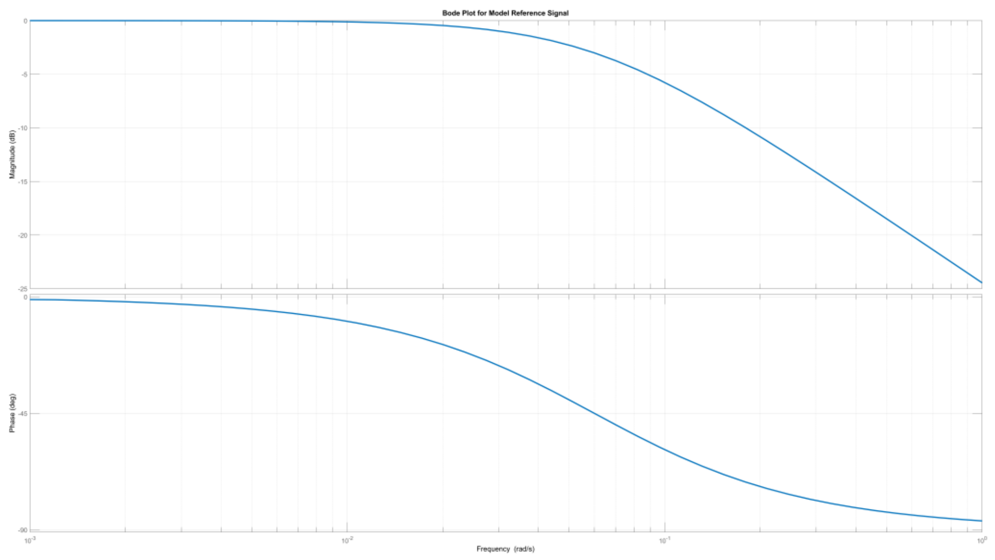


Figure 8.2 – Bode plot of model reference signal $G_r(s)$ with $\tau_{ref} = 0.06$

The modifications for the control and reference can be summarized by the simplified block diagram shown in Figure 8.3 with the mass estimator, LPV controller, scaling gain K_u, control conditioning subsystem, and hover state conditioning subsystem contained within the LPV control system block.

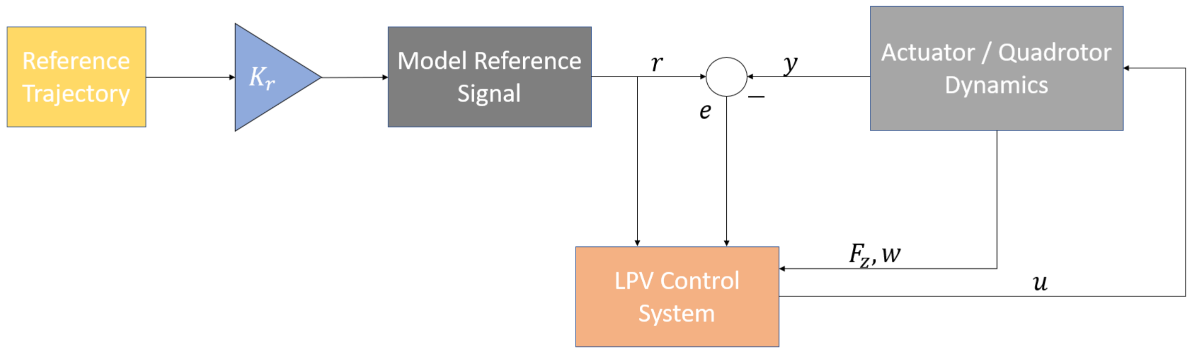


Figure 8.3 – System modifications and overall representation

8.1.3 Nonlinear Simulation

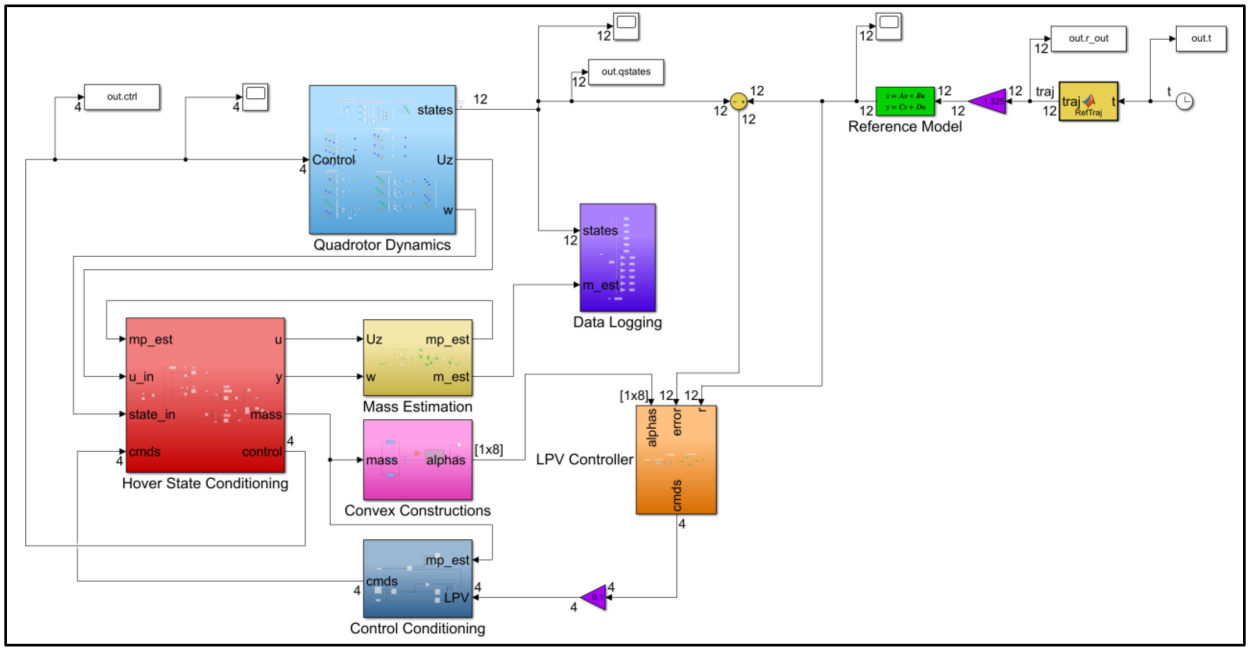
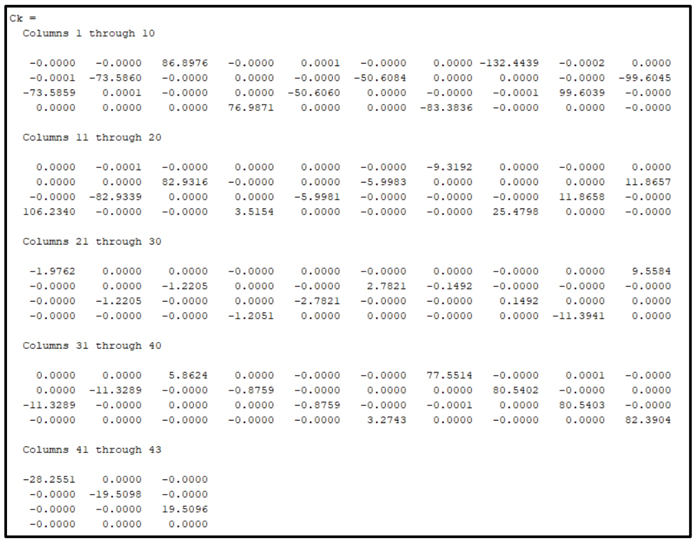


Figure 8.4 – Nonlinear simulation setup with desired forces and torques as inputs

Table 8.1 – Simulation Parameters

Parameters	Description	Value
k_u	Control scaling gain	0.1
k_r	Reference scaling gain	1.325
γ	Rate of convergence gain	15
τ_{ref}	Model reference time constant	0.06 s
t	Simulation time	300 s



```
Ck =
  Columns 1 through 10

   -0.0000   -0.0000   86.8976   -0.0000    0.0001   -0.0000    0.0000 -132.4439   -0.0002    0.0000
   -0.0001  -73.5860   -0.0000    0.0000   -0.0000  -50.6084    0.0000    0.0000   -0.0000  -99.6045
  -73.5859    0.0001   -0.0000    0.0000  -50.6060    0.0000   -0.0000   -0.0001   99.6039   -0.0000
    0.0000    0.0000    0.0000   76.9871    0.0000    0.0000  -83.3836   -0.0000    0.0000   -0.0000

  Columns 11 through 20

    0.0000   -0.0001   -0.0000    0.0000    0.0000   -0.0000   -9.3192    0.0000   -0.0000    0.0000
    0.0000    0.0000   82.9316   -0.0000    0.0000   -5.9983    0.0000    0.0000    0.0000   11.8657
   -0.0000  -82.9339    0.0000    0.0000   -5.9981   -0.0000   -0.0000   -0.0000   11.8658   -0.0000
  106.2340   -0.0000   -0.0000    3.5154    0.0000   -0.0000   -0.0000   25.4798    0.0000   -0.0000

  Columns 21 through 30

   -1.9762    0.0000    0.0000   -0.0000    0.0000   -0.0000    0.0000   -0.0000    0.0000    9.5584
   -0.0000    0.0000   -1.2205    0.0000   -0.0000    2.7821   -0.1492   -0.0000   -0.0000   -0.0000
   -0.0000   -1.2205   -0.0000    0.0000   -2.7821   -0.0000   -0.0000    0.1492    0.0000    0.0000
   -0.0000   -0.0000   -0.0000   -1.2051    0.0000    0.0000   -0.0000    0.0000  -11.3941    0.0000

  Columns 31 through 40

    0.0000    0.0000    5.8624    0.0000   -0.0000   -0.0000   77.5514   -0.0000    0.0001   -0.0000
    0.0000  -11.3289   -0.0000   -0.8759   -0.0000    0.0000    0.0000   80.5402   -0.0000    0.0000
  -11.3289   -0.0000    0.0000    0.0000   -0.8759   -0.0000   -0.0001    0.0000   80.5403   -0.0000
   -0.0000    0.0000   -0.0000   -0.0000   -0.0000    3.2743    0.0000   -0.0000    0.0000   82.3904

  Columns 41 through 43

  -28.2551    0.0000   -0.0000
   -0.0000  -19.5098   -0.0000
   -0.0000   -0.0000   19.5096
   -0.0000    0.0000    0.0000
```

Figure 8.5 – C_K controller matrix at $\widehat{m}_p = 2$

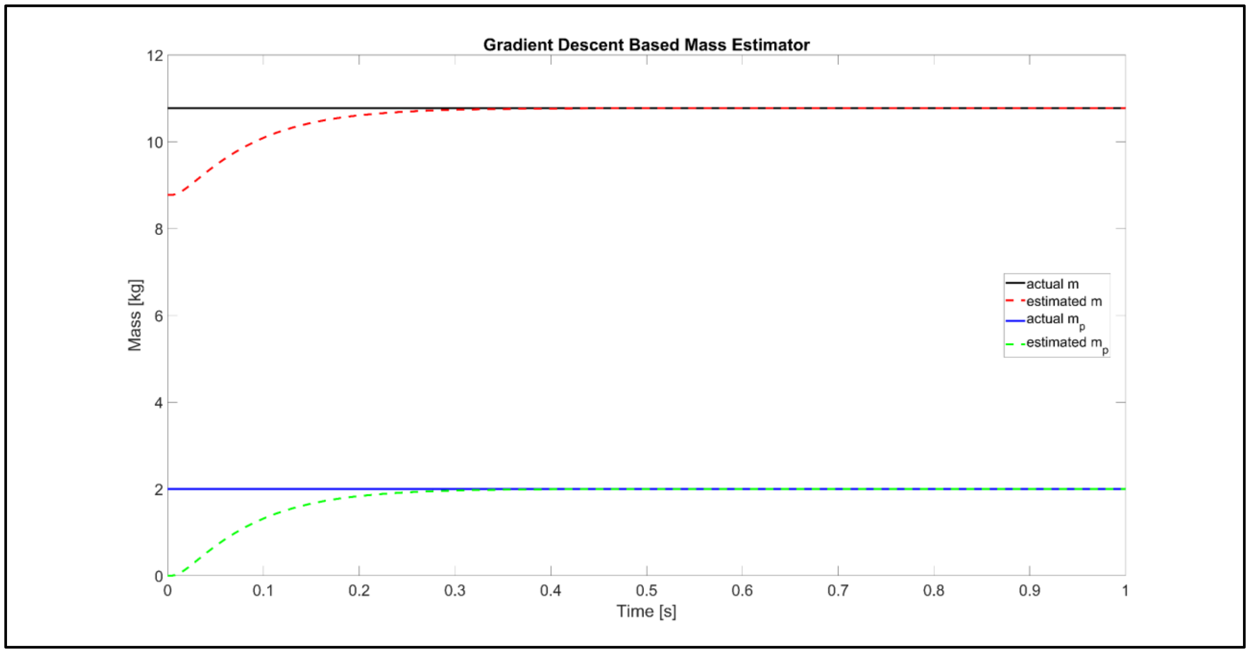


Figure 8.6 – Nonlinear simulation of mass estimation

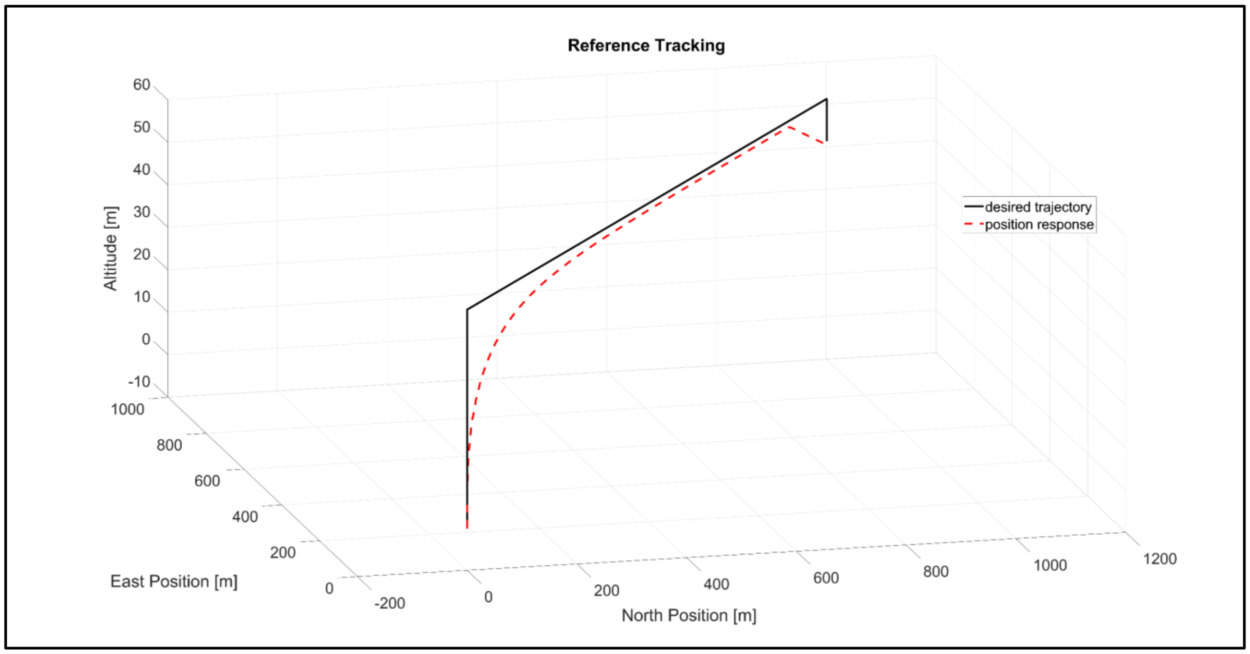


Figure 8.7 – Nonlinear simulation of position tracking control

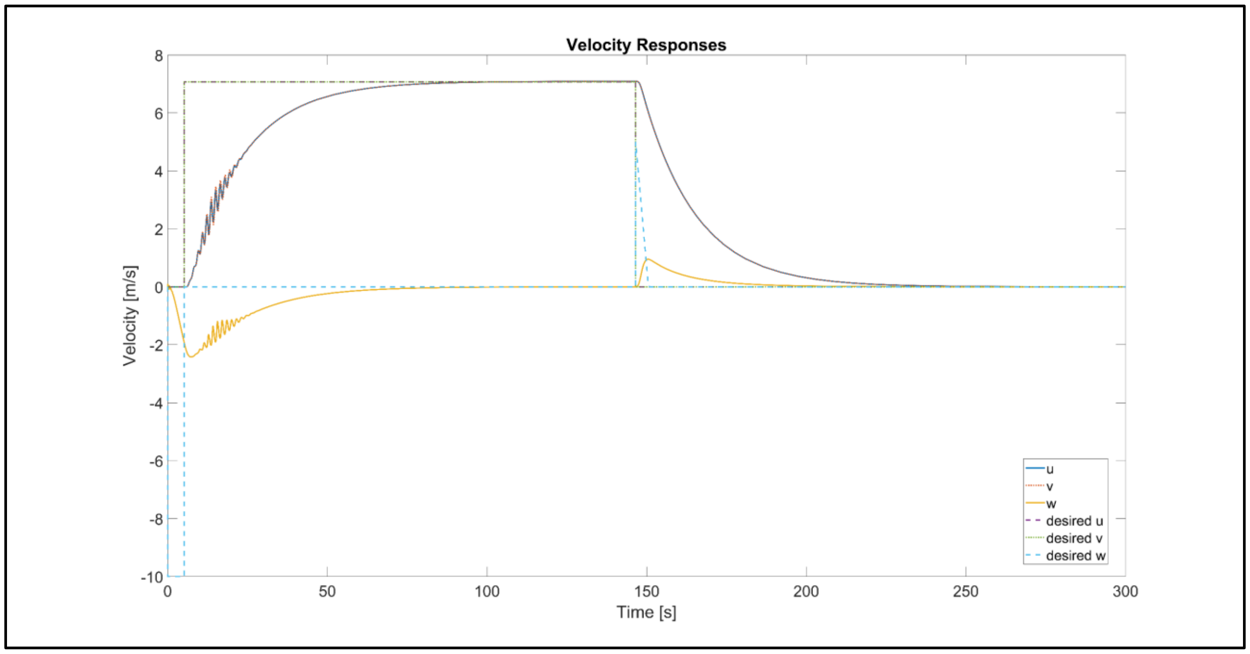


Figure 8.8 – Nonlinear simulation of velocity responses

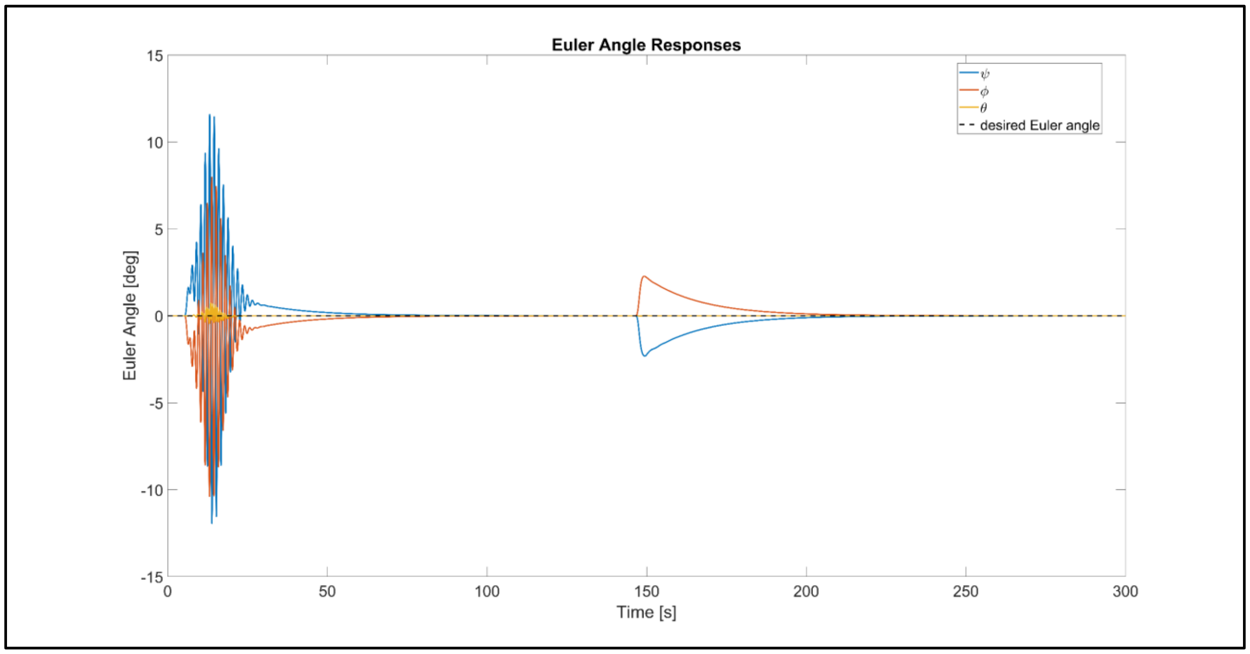


Figure 8.9 – Nonlinear simulation of Euler angle responses

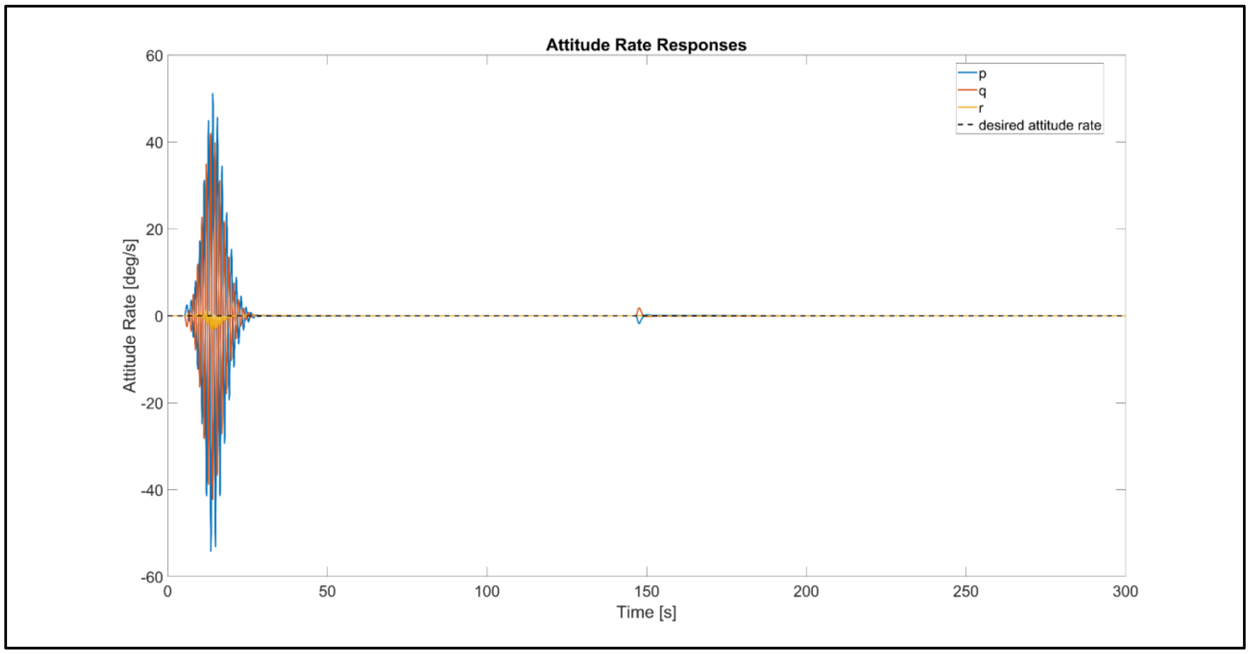


Figure 8.10 – Nonlinear simulation of attitude rate responses

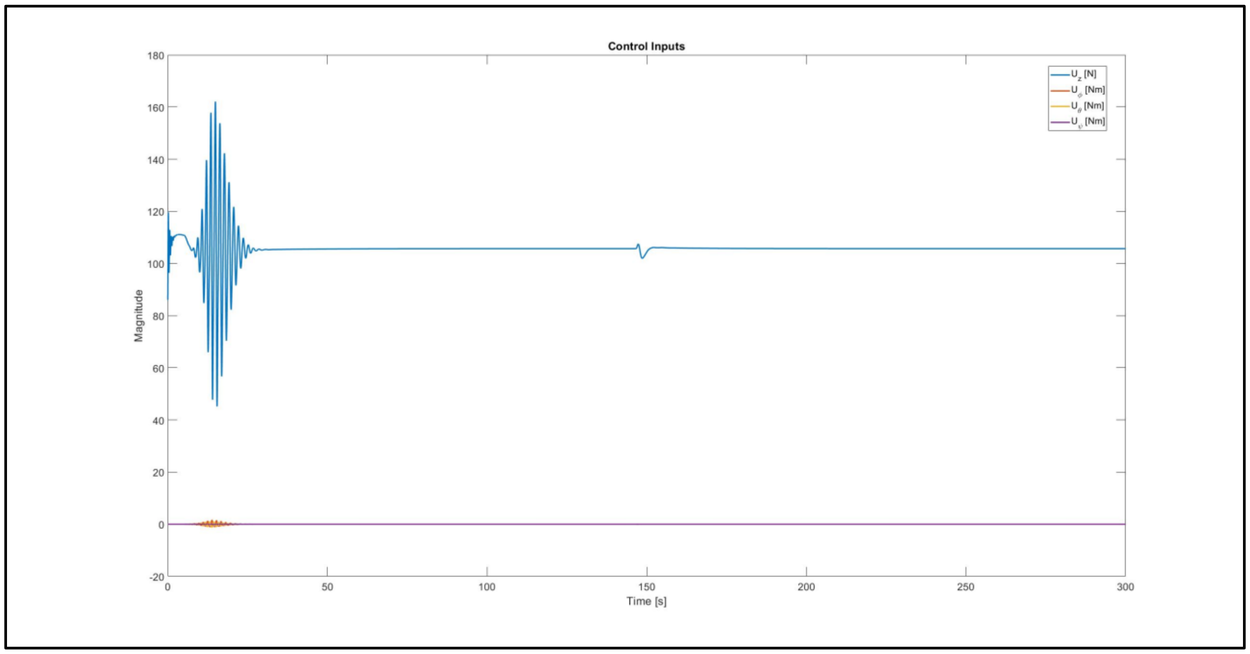


Figure 8.11 – Control inputs

8.2 Updated LPV Controller

8.2.1 Propeller Speed Based LPV Model and Controller

To derive the propeller speed based LPV controller, the linear, parameter dependent model from Chapter 2 is rewritten with the control inputs as functions of the propeller speeds. The model is summarized by (8.3).

$$\begin{gathered}\dot{\boldsymbol{x}} = A(\rho)\boldsymbol{x} + B(\rho)\boldsymbol{u} \\ x = [X\ Y\ Z\ u\ v\ w\ \varphi\ \theta\ \psi\ p\ q\ r]^T \\ u = [\Omega_f \quad \Omega_r \quad \Omega_b \quad \Omega_l]^T\end{gathered} \tag{8.3}$$

where the matrices $A(\rho)$ and $B(\rho)$ are given by (8.4)

$$A(\rho) = \begin{bmatrix} 0_{3x3} & S^1_{3x3} & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & S^2_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & 0_{3x3} & S^3_{3x3} \\ 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} \end{bmatrix} \text{ and } B(\rho) = \begin{bmatrix} 0_{5x1} & 0_{5x1} & 0_{5x1} & 0_{5x1} \\ \rho_1 & \rho_1 & \rho_1 & \rho_1 \\ 0_{3x1} & 0_{3x1} & 0_{3x1} & 0_{3x1} \\ 0 & -\rho_2 & 0 & \rho_2 \\ \rho_2 & 0 & -\rho_2 & 0 \\ -\rho_3 & \rho_3 & -\rho_3 & \rho_3 \end{bmatrix} \tag{8.4}$$

and the matrices S^1_{3x3}, S^2_{3x3}, S^3_{3x3} and the parameter functions ρ_1, ρ_2, ρ_3 are defined by (2.16) and (8.5), respectively. The equilibrium operating point at hover is $u_e = \sqrt{\frac{mg}{4K_F}}[1\ 1\ 1\ 1]$.

$$\begin{gathered}\rho_1 = \frac{-1.222 * 10^{-7}\sqrt{4.014e7 * m_p\ +\ 3.521 * 10^8}}{m_p + m_{base}} \\ \rho_2 = \frac{7.332 * 10^{-8}\sqrt{4.014e7 * m_p\ +\ 3.521 * 10^8}}{0.009 * m_p + 0.3013} \\ \rho_3 = \frac{3.0 * 10^{-9}\sqrt{4.014e7 * m_p\ +\ 3.521 * 10^8}}{0.009 * m_p + 0.5353}\end{gathered} \tag{8.5}$$

The same LPV modeling and controls process described in Chapters 5 and 6 is applied to derive the controller, but with a different affine parameter dependent system. The MATLAB code for the propeller based LPV controller is provided in Appendix A.2 with the singular values plots for the controller and closed-loop system shown in Figure 8.12 and Figure 8.13.

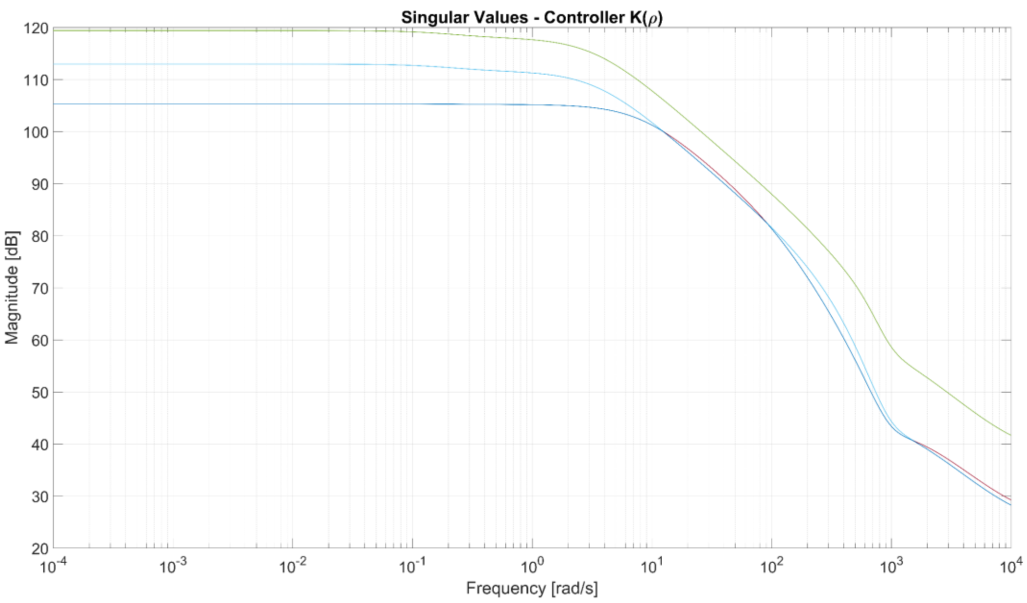


Figure 8.12 – Singular values plot of propeller speed based LPV controller

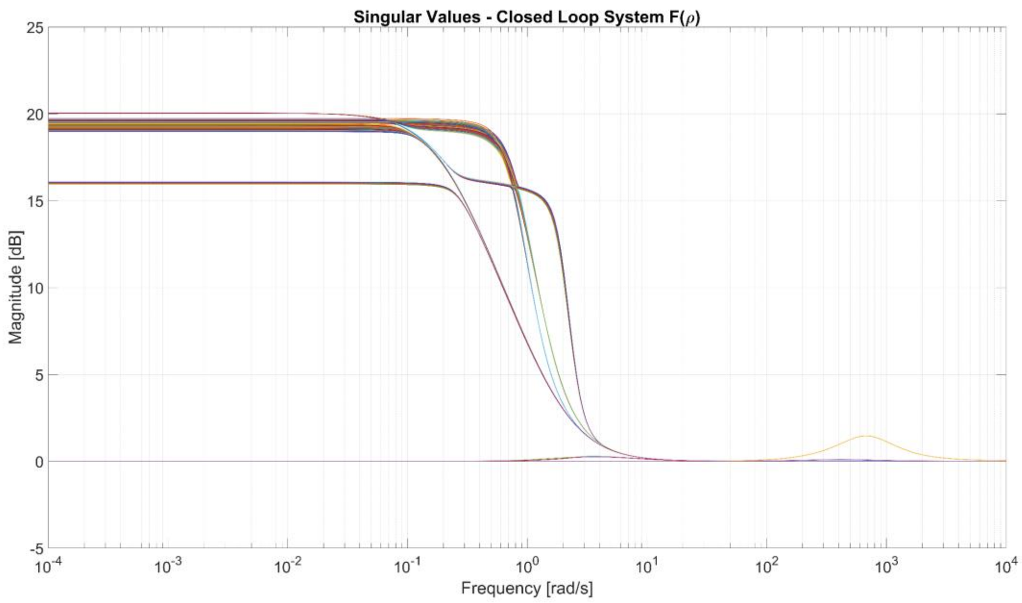


Figure 8.13 – Singular values of propeller speed based closed loop system

8.2.2 Nonlinear Simulation with Actuator Dynamics

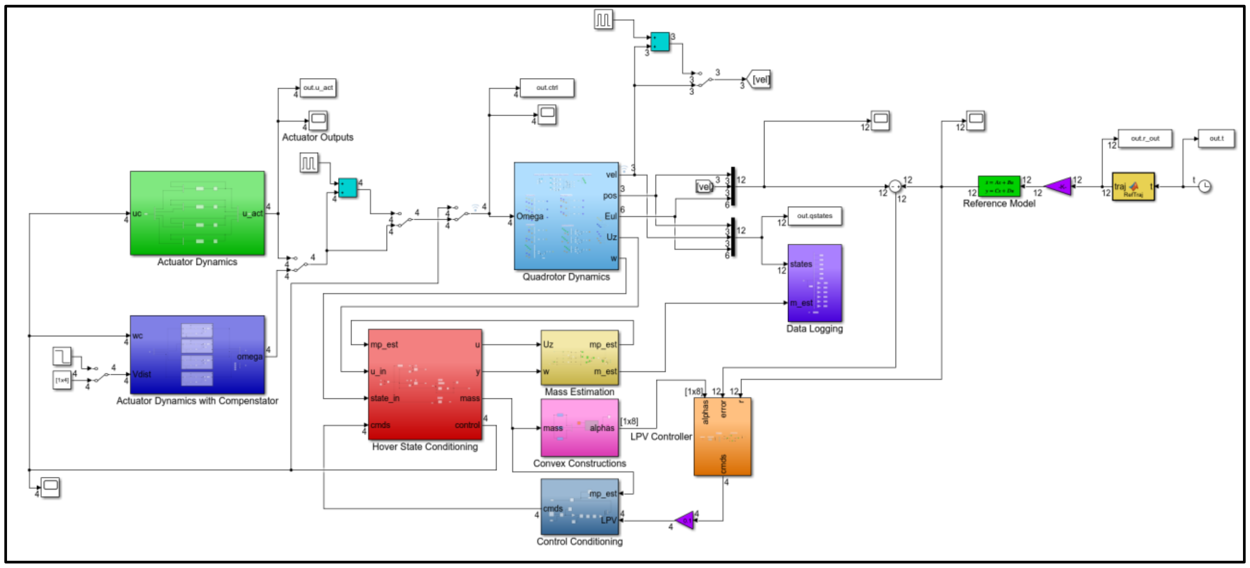
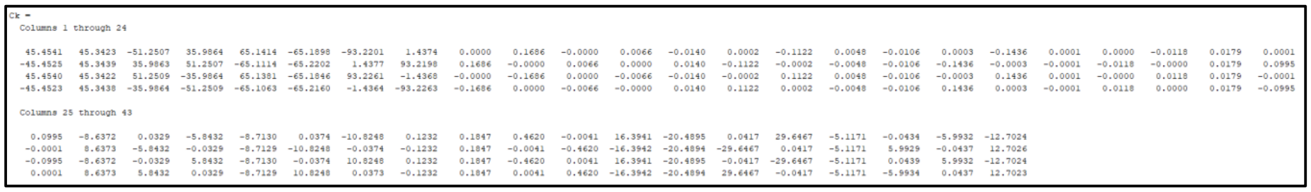


Figure 8.14 – Nonlinear simulation with actuator dynamics

Table 8.2 – Simulation Parameters

Parameters	**Description**	**Value**
K_u	Control scaling gain	0.1
K_r	Reference scaling gain	$[1.685 \quad 1.685 \quad 1.99 \quad I_{1x9}]^T$
γ	Rate of convergence gain	15
τ_{ref}	Model reference time constant	$0.1\ s$
t	Simulation time	$300\ s$



```
Ck =
  Columns 1 through 24

   45.4541   45.3423  -51.2507   35.9864   65.1414  -65.1898  -93.2201    1.4374    0.0000    0.1686   -0.0000    0.0066   -0.0140    0.0002   -0.1122    0.0048   -0.0106    0.0003   -0.1436    0.0001    0.0000   -0.0118    0.0179    0.0001
  -45.4525   45.3439   35.9863   51.2507  -65.1114  -65.2202    1.4377   93.2198    0.1686   -0.0000    0.0066    0.0000    0.0140   -0.1122   -0.0002   -0.0048   -0.0106   -0.1436   -0.0003   -0.0001   -0.0118   -0.0000    0.0179    0.0995
   45.4540   45.3422   51.2509  -35.9864   65.1381  -65.1846   93.2261   -1.4368   -0.0000   -0.1686    0.0000   -0.0066   -0.0140   -0.0002    0.1122    0.0048   -0.0106   -0.0003    0.1436    0.0001   -0.0000    0.0118    0.0179   -0.0001
  -45.4523   45.3438  -35.9864  -51.2509  -65.1063  -65.2160   -1.4364  -93.2263   -0.1686    0.0000   -0.0066   -0.0000    0.0140    0.1122    0.0002   -0.0048   -0.0106    0.1436    0.0003   -0.0001    0.0118    0.0000    0.0179   -0.0995

  Columns 25 through 43

    0.0995   -8.6372    0.0329   -5.8432   -8.7130    0.0374  -10.8248    0.1232    0.1847    0.4620   -0.0041   16.3941  -20.4895    0.0417   29.6467   -5.1171   -0.0434   -5.9932  -12.7024
   -0.0001    8.6373   -5.8432   -0.0329   -8.7129  -10.8248   -0.0374   -0.1232    0.1847   -0.0041   -0.4620  -16.3942  -20.4894  -29.6467    0.0417   -5.1171    5.9929   -0.0437   12.7026
   -0.0995   -8.6372   -0.0329    5.8432   -8.7130   -0.0374   10.8248    0.1232    0.1847   -0.4620    0.0041   16.3941  -20.4895   -0.0417  -29.6467   -5.1171    0.0439    5.9932  -12.7024
    0.0001    8.6373    5.8432    0.0329   -8.7129   10.8248    0.0373   -0.1232    0.1847    0.0041    0.4620  -16.3942  -20.4894   29.6467   -0.0417   -5.1171   -5.9934    0.0437   12.7023
```

Figure 8.15 – C_K controller matrix at $\widehat{m}_p = 2$

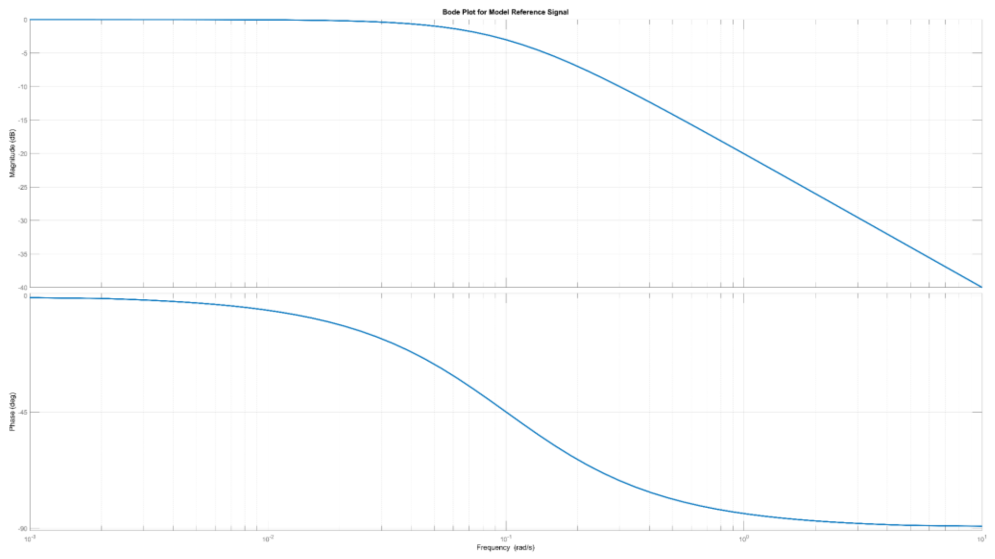


Figure 8.16 – Bode plot for reference signal with $\tau_{ref} = 0.1$

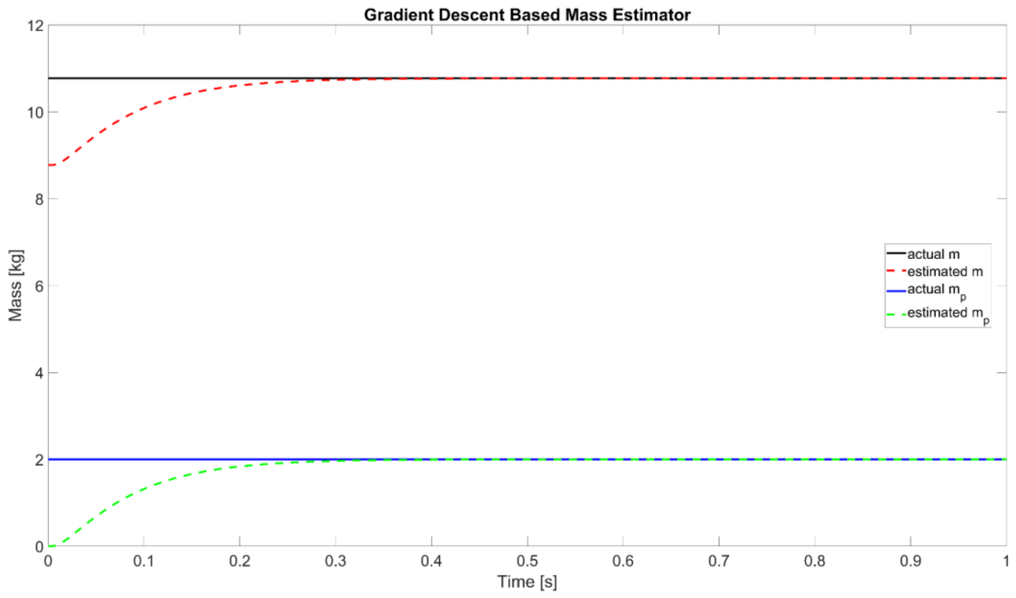


Figure 8.17 – Mass estimation against nonlinear system with actuator dynamics

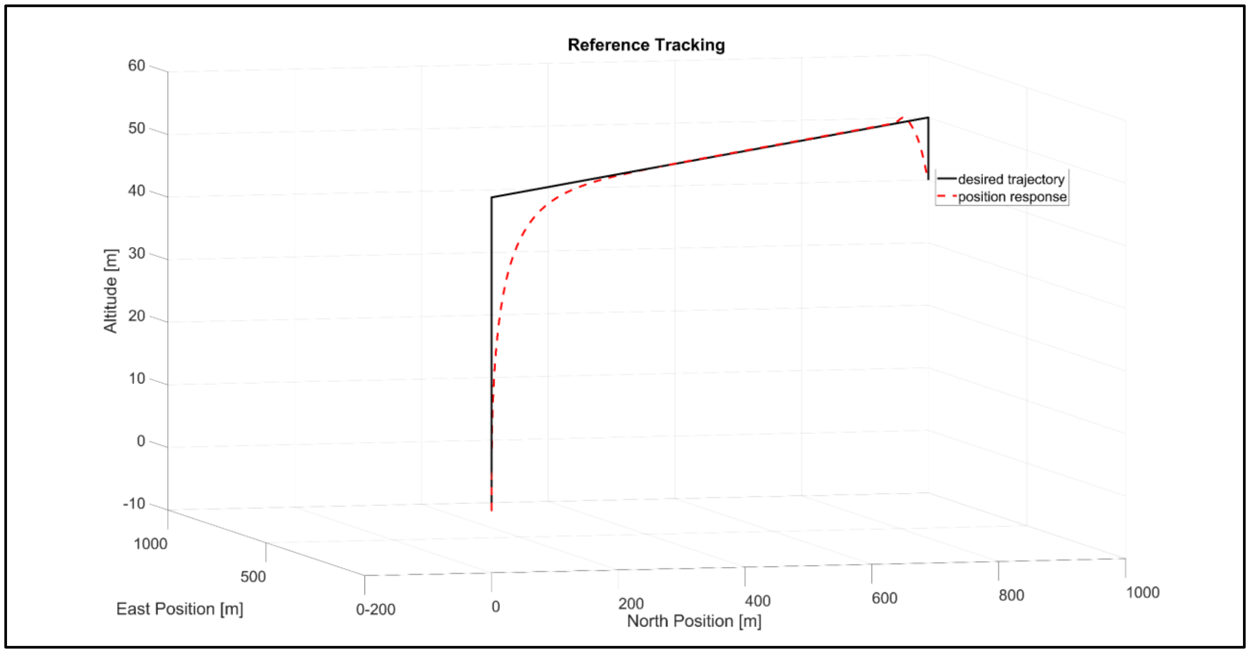


Figure 8.18 – Trajectory tracking against nonlinear system with actuator dynamics

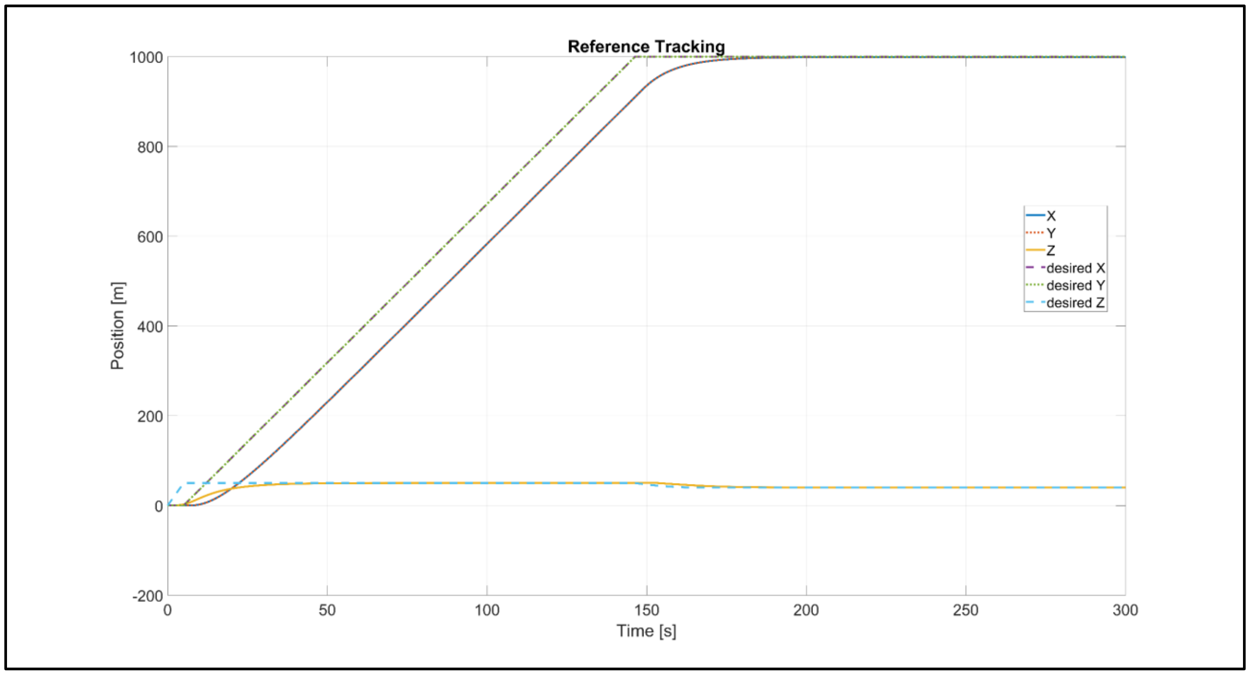


Figure 8.19 – Position state responses of nonlinear system with actuator dynamics

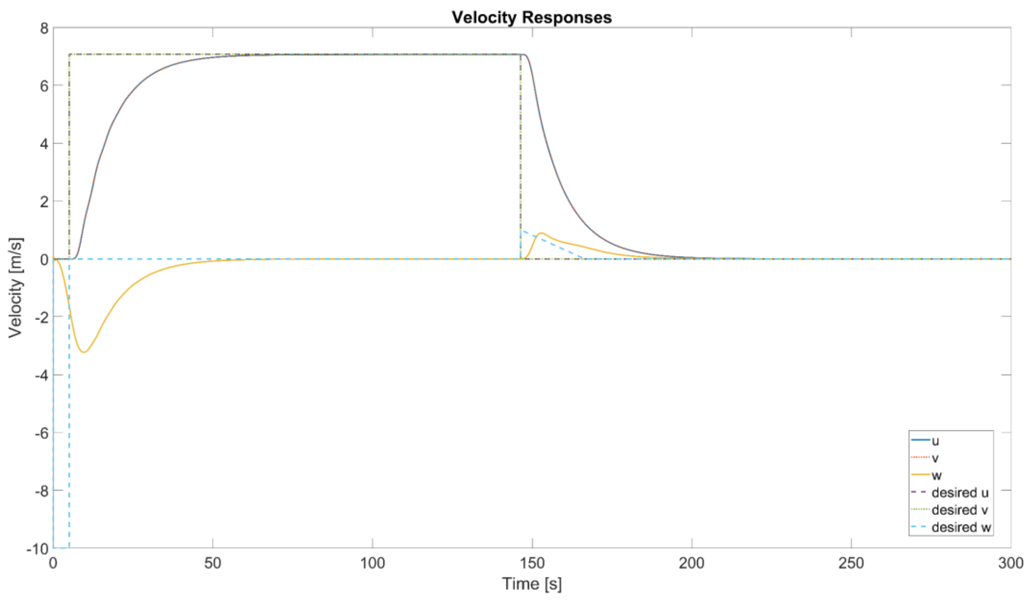


Figure 8.20 – Velocity state responses of nonlinear system with actuator dynamics

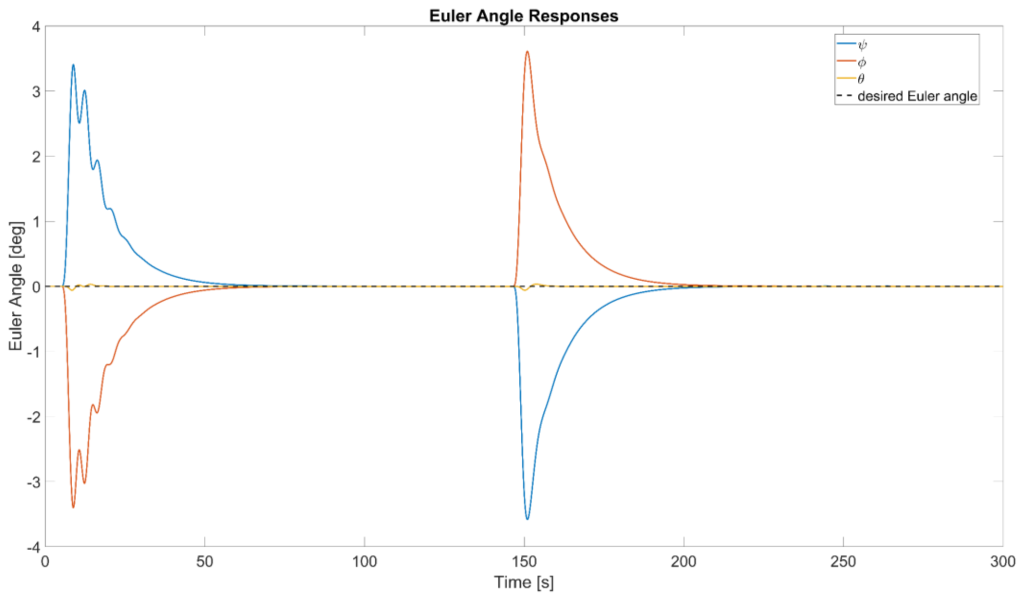


Figure 8.21 – Euler angle state responses of nonlinear system with actuator dynamics

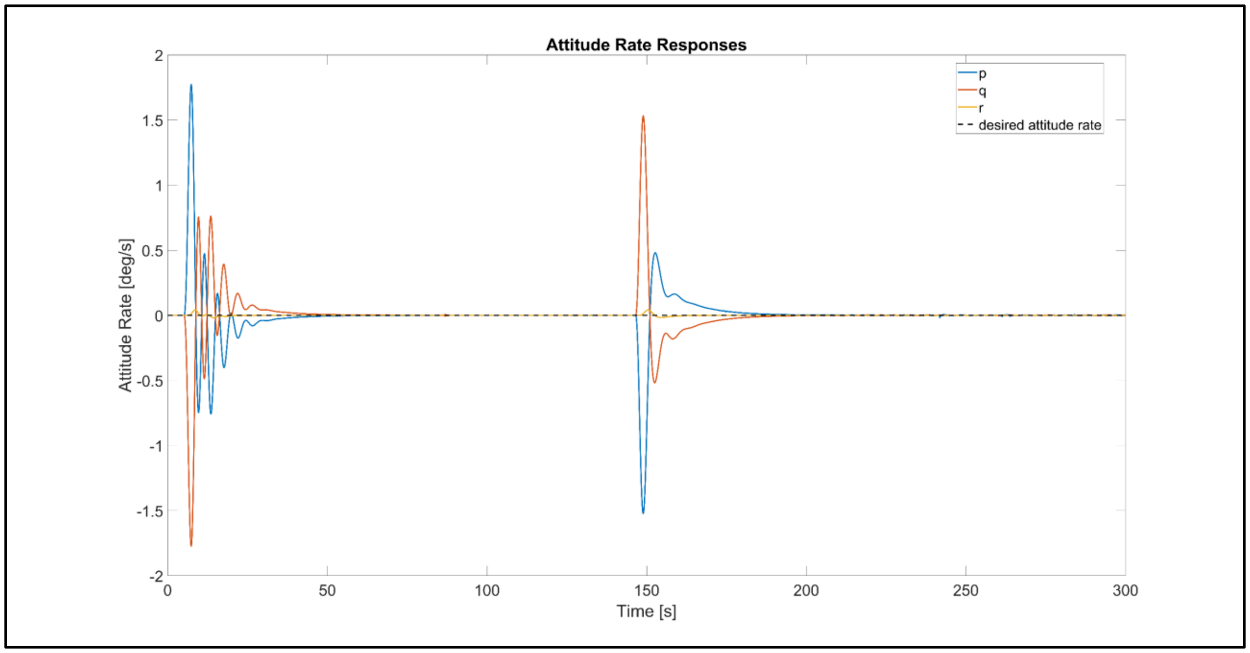


Figure 8.22 – Attitude rate state responses of nonlinear system with actuator dynamics

8.3 Nonlinear Simulation with Disturbances

8.3.1 Setup

To stress the controller, the disturbances sources shown in Figure 8.14 are switched at the times given by Table 8.3. The same simulation parameters of Table 8.2 are used.

Table 8.3 – Disturbance sources in simulation

Disturbance	**Time Applied**
Pulse on Control Inputs	$t = 30\ s$
Pulse on Velocity States	$t = 50\ s$
Step on Actuator Input Voltage	$t = 100\ s$

8.3.2 Results

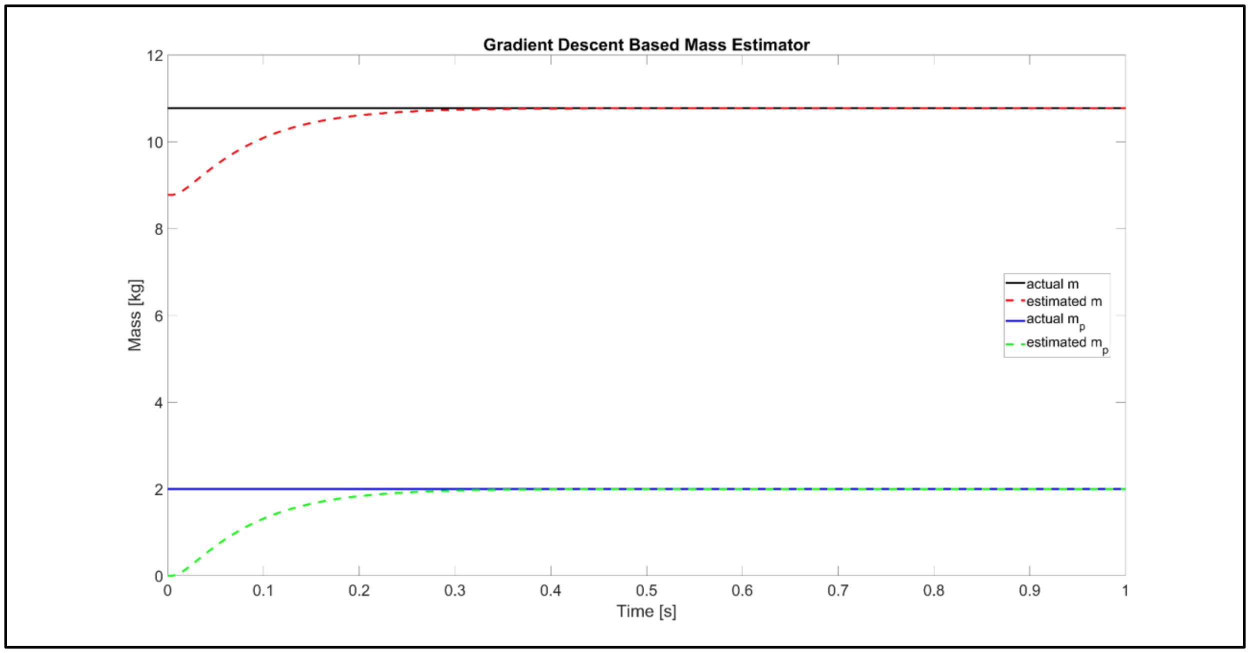


Figure 8.23 – Mass estimation against nonlinear system with disturbances

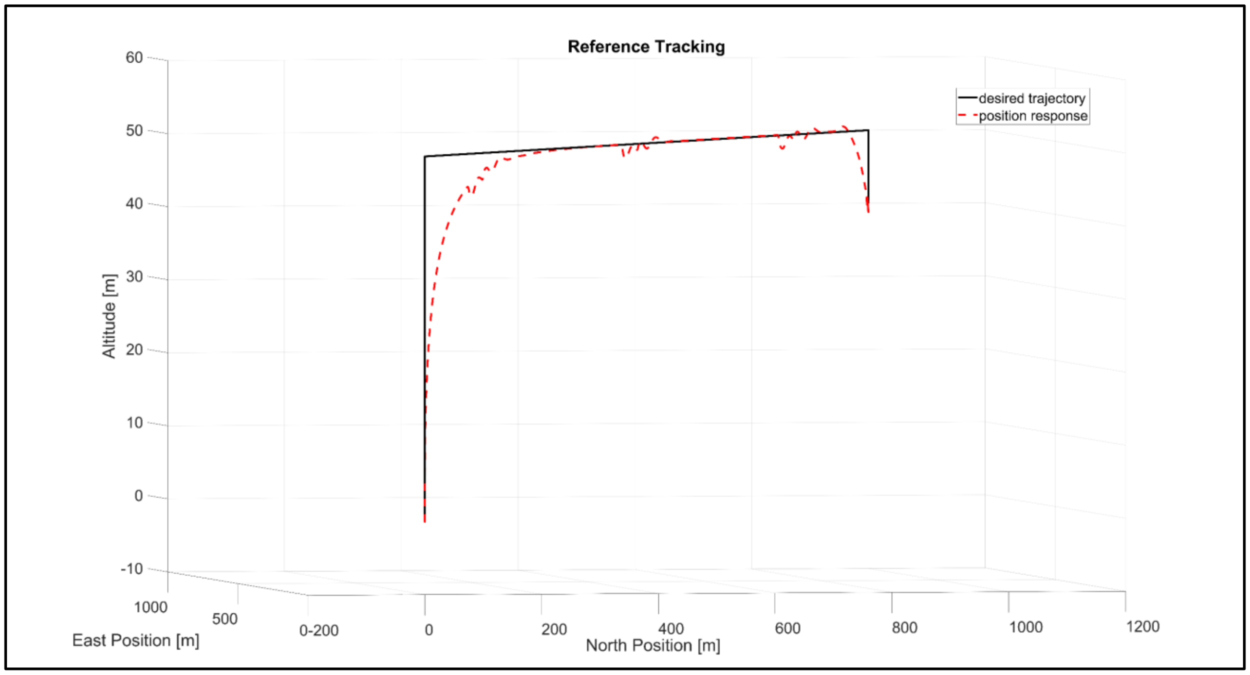


Figure 8.24 – Trajectory tracking against nonlinear system with disturbances

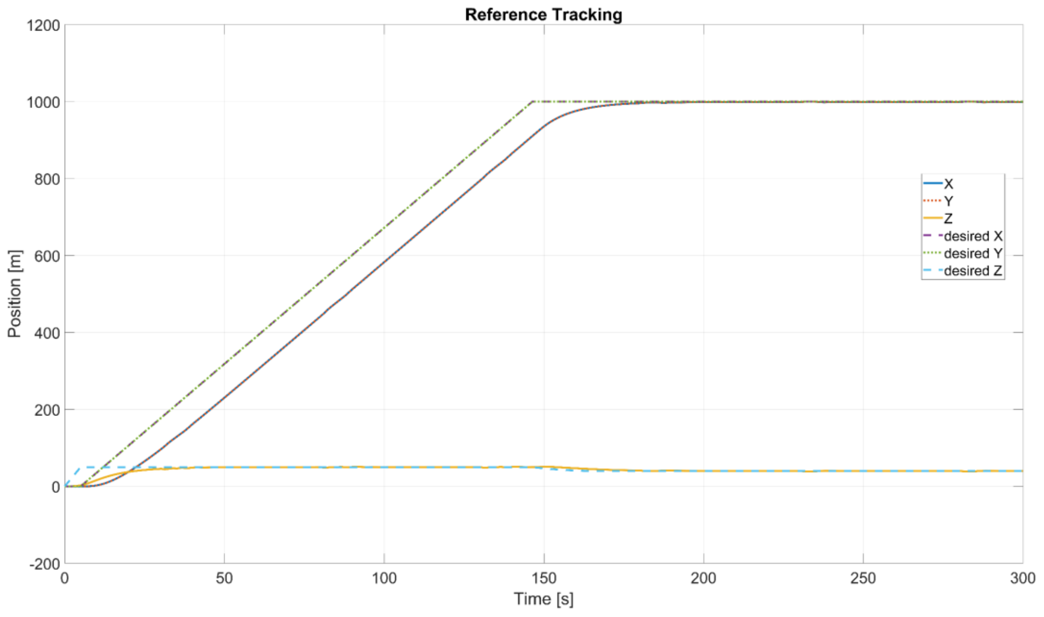


Figure 8.25 – Position state responses of nonlinear system with disturbances

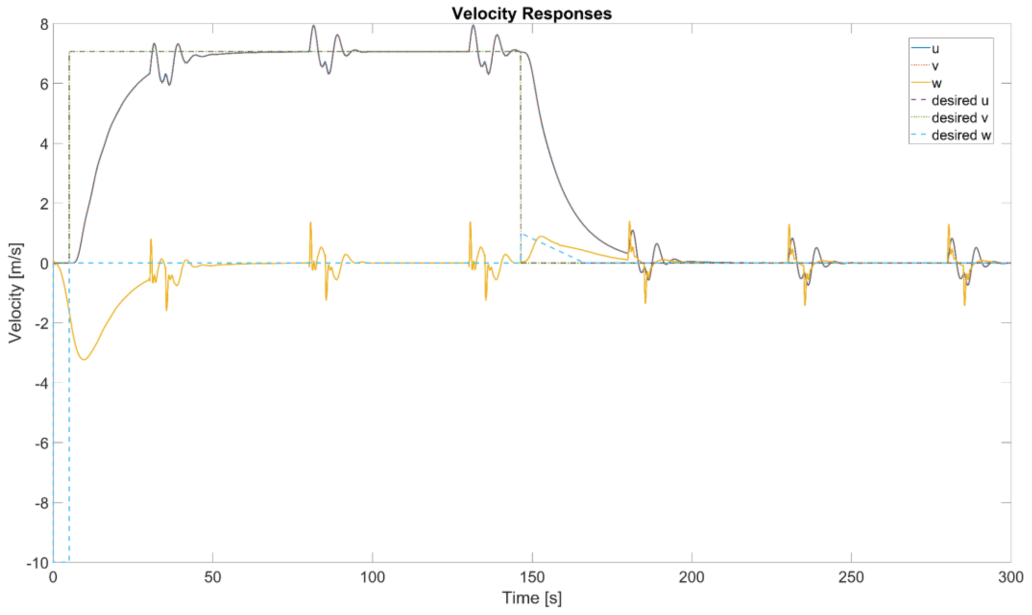


Figure 8.26 – Velocity state responses of nonlinear system with disturbances

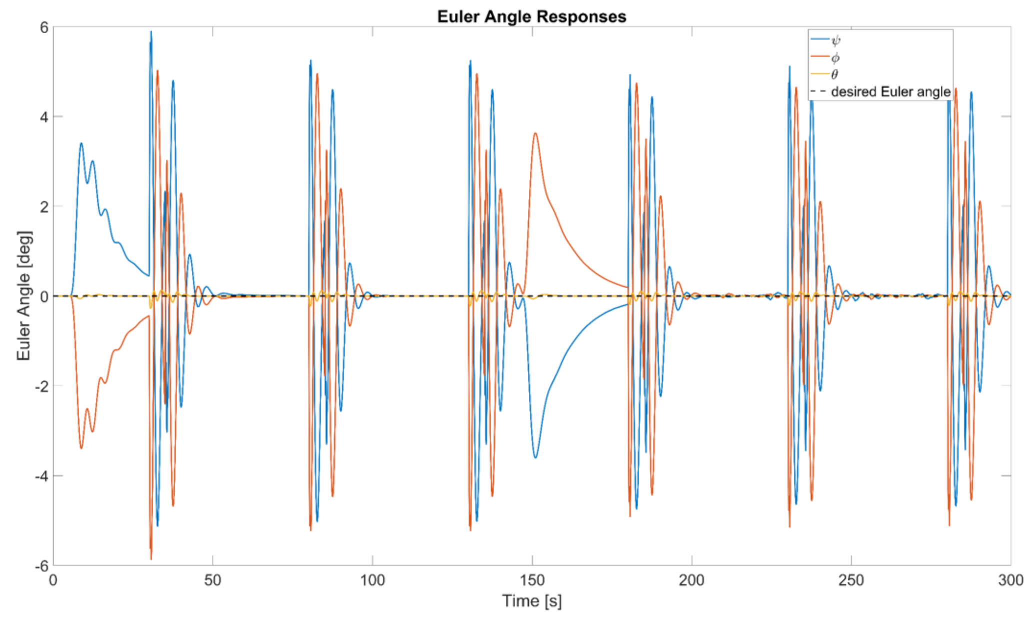


Figure 8.27 – Euler angle state responses of nonlinear system with disturbances

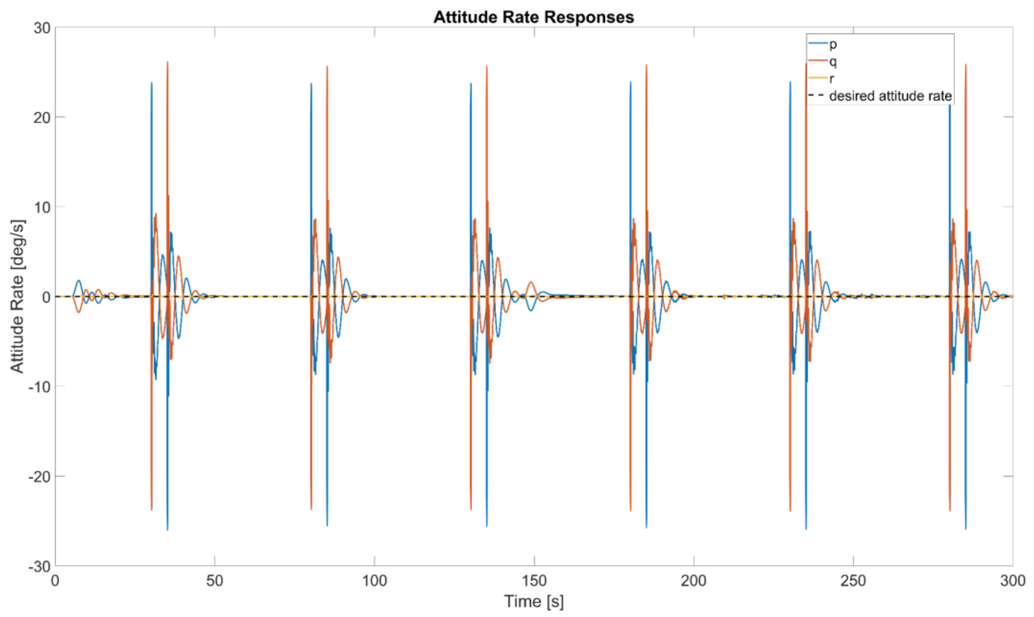


Figure 8.28 – Attitude rate responses of nonlinear system with disturbances

8.4 Discussion

Without the modifications, the LPV controller could not directly stabilize the nonlinear system. Modifying the reference and control was not necessary for the linear simulation in Chapter 6, where with scaling the control commands could control the system directly. The model reference signal filters the reference trajectory to gradually ramp up the velocities to prevent the demanded accelerations from causing instability, however the cruise velocity v_m had to be decreased to $10\ m/s$ with a limit of $20\ m/s$. The simulations show the modifications resolved stability issues and the controller was able to track the reference.

The switching design from Chapter 4 where the direct input U_p is used to lift the mass resulted in stability issues when switching to the LPV commands after the hover signal was disabled. There was significant “wiggle” in the quadrotor motion that produced unstable behavior. Therefore, all control was switched over to be automatically handled by the LPV controller from $t = 0$. The rate of convergence gain for the mass estimator was increased to $\gamma = 15$ to accommodate the system modifications. During the estimation time from $t = 0$ to $t = t_{\hat{m}_p} < 1s$, the LPV controller is continually adjusting its gains to the mass estimate until the hover enable signal is disabled at $t = 1\ s$. It was found the LPV control commands could handle fast variations in the mass estimate and the max payload weight, without having to apply the direct input U_p. The filter time constant had to be increased to 0.1 to control the nonlinear system with actuator dynamics. The time t_b had to be increased to $20\ s$ from $4\ s$ in the linear simulation to decrease the velocity demands.

The main takeaway from the simulations is the nonlinear system imposes physical limitations on the control the linear system did not need to consider. This is also due to the hover assumptions of the LPV model. The quadratic and robust stability results discussed in Chapter 6 only guarantee stability for the LPV system, not the nonlinear system. Small steady state errors are still present in the state responses, but these can be resolve by further tuning the feedforward gains. Finally, multiple payload masses within the design range were tested to validate the control systems meets the requirements.

9 CHAPTER 9

CONCLUSION

9.1 Advantages of LPV Control System

The LPV controller based on the self-scheduling $\mathcal{H}_\infty$ technique was able to track a desired trajectory subject to an unknown mass and disturbance sources, with some performance issues. The modifications enabled LPV control system compatibility for the nonlinear system. Since the parameter box and vertex controllers are determined offline, only the convex coordinates are necessary to determine the online LPV controller matrices. Additionally, the mass estimator proved to be robust when tested against the nonlinear system including actuator dynamics by still converging to the true mass in different scenarios. However, a rigorous proof is necessary to prove parameter convergence for all trajectories as $t \to \infty$. The design of the range of the payload mass can easily be increased to produce a new controller without increasing controller dimensions or changing its structure, adding flexibility to the design requirements for the control system. For instance, the parameter space for the LPV controller could be adjusted to control a larger aerospace delivery vehicle making multiple payload pickups or drop-offs along a flight path. Additionally, the hover state conditioning system switches off the mass estimation when the hover signal is disabled. This saves computational resources online as the LPV controller will use the converged mass estimate until it is necessary to estimate the payload again. By feeding in the payload mass directly, the controller gains adjust automatically to the mass estimates without requiring a switching to a worst-case lifting force input.

9.2 Limitations and Possible Solutions

As demonstrated in Chapter 8, controlling the nonlinear system is not a trivial task. The nonlinear system imposes physical limitations on what can be achieved by the control. For example, the controller could not stabilize the plant when the demanded velocities exceeded more than $20\ m/s$. Relaxing the hover state conditions by expanding the parameter space to include the velocity states and then developing the LPV controller to gain schedule based on both payload mass and velocity could result in a more reactive controller so that the quadrotor that can translate faster while maintaining stability. Furthermore, incorporating actuator constraints for the propeller speed limits into the LMI formulation would allow for a control solution that is physically realizable within the hardware capability of a delivery quadrotor.

9.3 Multivariable vs. SISO Approaches

The main advantage of multivariable control is much of the controls design can be handled in “one shot”. By feeding the reference and error from the state measurements into the controller, the LPV controller generates the four control inputs necessary to affect the motion of the quadrotor. In multivariable control, each controlled output can be dependent on several variables and/or inputs. In contrast to SISO control, as outlined in Appendix C using PID control of a quadrotor as an example, each controlled output is controlled by one input in a successive loop closure structure.

9.4 Controller Function in an Overall GNC System

The function of any feedback controller is to first stabilize the plant and then improve system performance for some tracking quality criteria and to provide robustness in meeting control objectives even in the presence of system variations, unmodeled dynamics, disturbances, or noise. A complete guidance, navigation, and control system could also include state estimation and an optimizer which minimizes or maximizes an objective to produce a reference for the controller to track according to Figure 9.1

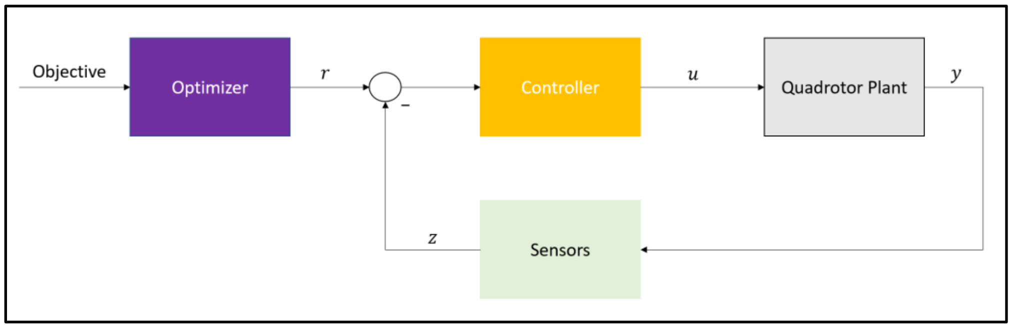


Figure 9.1 – Optimizer and controller

A further breakdown of the optimizer and controller, but still at a high-level representation, is shown in Figure 9.2, based on reference notes [41]. The trajectory can be altered as necessary online based on the control law and state estimation blocks. This project did not consider trajectory generation based on generating an optimal u_{ref} and x_{ref}, as shown in Figure 9.2, for the feedback controller to then track. For example, a model predictive controller (MPC) could generate the optimal trajectory in the outer loop, the LPV controller tracks and rejects any disturbances that occur along the path in the inner loop.

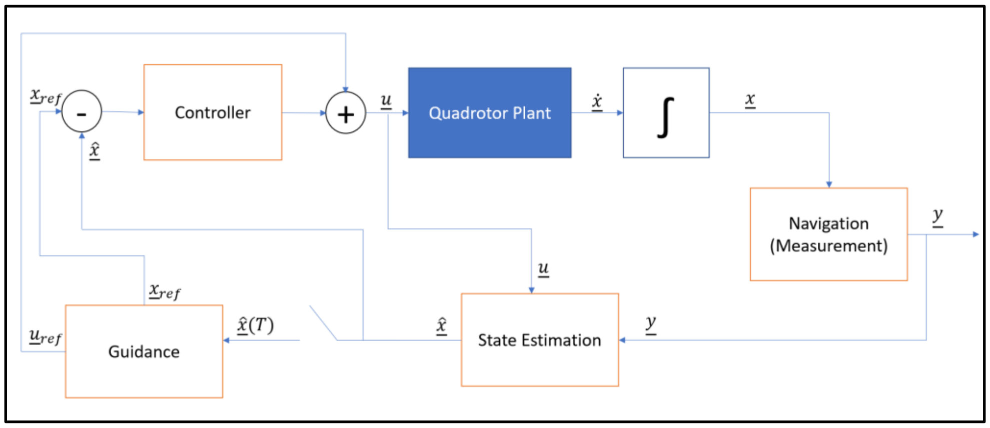


Figure 9.2 – General guidance, navigation, and control system

To narrow down the scope of the study, the project assumed all states were available for measurement, therefore the state estimation block is not considered. In practice, all states are not available for direct feedback due to design limitations and sensor dynamics. The project considered estimation for an unknown system parameter, but not an inaccessible or undesired state measurement. Generally, a controller and state observer can be designed separately and integrated together for an observer-controller system. This separation principle applies to LPV systems.

9.5 Future Research

Future research involves investigating the performance advantages gained by the LPV controller by refining the weights selection used in the $\mathcal{H}_\infty$ mixed sensitivity design and relaxing the hover state assumptions by expanding the parameter space to include velocity states. To add guidance capability, research on integrating a guidance law to produce optimal reference trajectories automatically is considered to complete a guidance and control system for the delivery quadrotor.

REFERENCES

[1] T. Bresciani, "Modelling, Identification and Control of Quadrotor Helicopter," Lund University, Lund, Sweden, 2008.

[2] B. J. Emran and H. Najjaran, "A review of quadrotor: An underactuated mechanical system," *Annual Reviews in Control,* vol. 46, pp. 165-180, 2018.

[3] A. T. Larroya, "LPV Control of a Quadrotor," Universitat Politecnica, Catalonia, Spain, 2015.

[4] P. Apkarian and P. Gahinet, "A Convex Characterization of Gain-Scheduled Hinf Controllers," *IEEE Transaction on Automatic Control,* Vols. AC-40, pp. 853-854, 1995.

[5] Z. Liu, C. Yuan and Y. Zhang, "Active Fault-Tolerant Control of Unmanned Quadrotor Helicopter Using Linear Parameter Varying Technique," *Journal of Intelligent and Robotic Systems,* 2017.

[6] B. J. Emran and H. Najjaran, "Switching Control of Quadrotor with Adaptation Mechanism," in *2016 IEEE International Conference on Systems, Man, and Cybernetics*, Budapest, Hungary, 2016.

[7] J. Vincent and C. Gartenberg, "Here's Amazon's new transforming Prime Air delivery drone," The Verge, 5 June 2019. [Online]. Available: https://www.theverge.com/2019/6/5/18654044/amazon-prime-air-delivery-drone-new-design-safety-transforming-flight-video.

[8] Federal Aviation Administration, "Fact Sheet - Small Unmanned Aircraft Regulations (Part 107)," 23 July 2018. [Online]. Available: https://www.faa.gov/news/fact_sheets/news_story.cfm?newsId=22615.

[9] T. Risen, "FAA grants first drone airline approval for UPS," FlightGlobal, [Online]. Available: https://www.flightglobal.com/news/articles/faa-grants-first-drone-airline-approval-for-ups-461206/. [Accessed 2019].

[10] S. Swei, *Quadrotor Flight Controls,* San Jose, CA: Department of Aerospace Engineering, San Jose State University, Fall 2018.

[11] S. Skogestad and I. Postlethwaite, Multivariable Feedback Control: Analysis and Design, 2nd ed., Wiley, 2005.

[12] C. Wang, M. Nahon and M. Trentini, "Controller Development and Validation for a Small Quadrotor with Compensation for Model Variation," in *2014 International Conference on Unmanned Aircraft Systems*, Orlando, FL, 2014.

[13] I. Sadeghzadeh, M. Abdolhosseini and Y. M. Zhang, "Payload Drop Application of Unmanned Quadrotor Helicopter Using Gain-Scheduled PID and Model Predictive Control Techniques," Department of Mechanical and Industrial Engineering, Concordia University, Montreal, QC, Canada.

[14] I.-H. Choi and H.-C. Bang, "Adaptive command filtered backsteeping tracking controller design for quadrotor unmanned aerial vehicle," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering,* vol. 226, no. 5, 2012.

[15] D. W. Kun and I. Hwang, "Linear Matrix Inequality-Based Nonlinear Adaptive Robust Control of Quadrotor," *Journal of Guidance, Control, and Dynamics,* vol. 39, no. 5, pp. 996-1008, 2016.

[16] P. Apkarian, P. Gahinet and G. Becker, "Self-scheduled Hinf Control of Linear Parameter-Varying Systems: a Design Example," *Automatica,* vol. 31, no. 9, pp. 1251-1261, 1995.

[17] I. Sadeghzadeh, A. Chamseddine, D. Theilliol and Y. Zhang, "Linear Parameter Varying Control Synthesis: State Feedback versus Hinf Technique with Application to Quadrotor UAV," in *2014 International Conference on Unmanned Aircraft Systems*, Orlando, FL, 2014.

[18] J. S. Shamma, "An Overview of LPV Systems," in *Control of Linear Parameter Varying Systems with Applications*, Springer Science+Business Media, 2012.

[19] H. D. Hughes and F. Wu, "Chapter 16: LPV Hinf Control for Flexible Hypersonic Vehicle," in *Control of Linear Parameter Varying Systems with Applications*, Springer Science+Business Media, 2012.

[20] P. Seiler, G. J. Balas and A. Packard, "Chapter 19: Linear Parameter-Varying Control for the X-53 Active Aeroelastic Wing," in *Control of Linear Parameter Varying Systems with Applications*, Springer Science+Business Media, 2012.

[21] Z. Yu, H. Chen and P.-y. Woo, "Gain Scheduled LPV Hinf Control Based on LMI Approach for a Robotic Manipulator," *Journal of Robotic Systems,* 2002.

[22] H. Pfifer, C. P. Moreno, J. Theis, A. Kotikapuldi, A. Gupta, B. Takarics and P. Seiler, "Linear Parameter Varying Techniques Applied to Aeroservoelastic Aircraft: In Memory of Gary Balas," *IFAC-PapersOnLine,* vol. 48, no. 26, pp. 103-108, 2015.

[23] D. Rotondo, F. Nejjari and V. Puig, "Robust Quasi-LPV Model Reference FTC of a Quadrotor UAV Subject to Actuator Faults," *International Journal of Applied Mathematics and Computer Science,* vol. 25, no. 1, pp. 7-22, 2015.

[24] J. Stephan, L. Schmitt and W. Fichter, "Linear Parameter-Varying Control for Quadrotors in Case of Complete Actuator Loss," *Journal of Guidance, Control, and Dynamics,* vol. 41, no. 10, 2018.

[25] J. Xu, "Design Perspectives on Delivery Drones," RAND Corporation, Santa Monica, CA, 2017.

[26] D. W. Mellinger, "Trajectory Generation and Control for Quadrotors," Publicly Accessible Penn Dissertations, 2012.

[27] K. K. Nermisky and K. Turkoglu, "Simulated Annealing-Based Optimal Control PID Controller Design: A Case Study on Nonlinear Quadrotor Dynamics," in *Proceedings of the ASME 2017 Dynamic Systems and Control Conference*, Tysons, Virginia, 2017.

[28] S. Boyd, L. El Ghaoui, E. Feron and V. Bakakrishnan, Linear Matrix Inequalities in System and Control Theory, Philadelphia: Society for Industrial and Applied Mathematics, 1994.

[29] MathWorks, *Robust Control Toolbox,* 2019.

[30] CVX Research, Inc., *CVX: Matlab Software for Disciplined Convex Programming.*

[31] S. L. Rangajeeva and J. F. Whidborne, "Linear Parameter Varying Control of a Quadrotor," in *6th International Conference on Industrial and Information Systems*, Sri Lanka, 2011.

[32] M. Fujita, *LPV System and Gain Scheduling,* University of Tokyo, Spring 2015.

[33] ECE 792, *Adaptive Control,* Raleigh, NC: North Carolina State University.

[34] G. Tao, Adaptive Control Design and Analysis, Hoboken, NJ: Wiley-Interscience, 2003.

[35] D.-W. Gu, P. H. Petkov and P. H. Konstantinov, Robust Control Design with MATLAB, vol. Advanced Textbooks in Control and Signal Processing, Glasgow, UK: Springer, 2nd Edition.

[36] G. Balas, R. Chiang, A. Packard and M. Safonov, *Robust Control Toolbox: Getting Started Guide,* MathWorks R2020a.

[37] J. How, *16.323 Principles of Optimal Control,* Massachusetts Institute of Technology: MIT OpenCourseWare, Spring 2008.

[38] S. Swei, *Control System Overview,* San Jose , CA: San Jose State University, Fall 2019.

[39] MathWorks, *Mixed-Sensitivity Loop Shaping.*

[40] M. Balas and B. Udrea, *Direct Model-Reference Adaptive Control of a DC Motor,* Daytona Beach, FL : Embry Riddle Aeronautical University, May 2015.

[41] A. Rao, *Lecture 30,* Optimal Control Theory*,* EML 6934, Gainesville, FL: University of Florida, Spring 2012.

[42] P. Apkarian and H. D. Tuan, "Parametrized LMIs in Control Theory," *SIAM Journal on Control and Optimization,* vol. 38, no. 4, pp. 1241-1264, 2000.

[43] S. Shahruz and S. Behtash, "Design of Controllers for Linear Parameter-Varying Systems by the Gain Scheduling Technique," *Journal of Mathematical Analysis and Applications,* vol. 168, pp. 195-217, 1992.

[44] G. Becker and A. Packard, "Robust performance of linear parametrically varying systems using parametrically-dependent linear feedback," *Systems and Control Letters,* vol. 23, pp. 205-215, 1994.

[45] F. Wu, X. H. Yang, A. Packard and G. Becker, "Induced L2-Norm Control for LPV Systems with Bounded Parameter Variation Rates," *International Journal of Robust and Nonlinear Control,* vol. 6, pp. 983-998, 1996.

[46] P. Apkarian and R. J. Adams, "Advanced Gain-Scheduling Techniques for Uncertain Systems," *IEEE Transactions on Control Systems Technology,* vol. 6, no. 1, 1998.

[47] C. Scherer, P. Gahinet and M. Chilali, "Multiobjective Output-Feedback Control via LMI Optimization," *IEEE Transactions on Automatic Control,* vol. 42, no. 7, 1997.

[48] C. W. Scherer, "Structured finite-dimensional controller design by convex optimization," *Linear Algebra and its Applications,* Vols. 351-352, pp. 639-669, 2002.

[49] A. Abdullah and M. Zribi, "Model reference control of LPV systems," *Journal of the Franklin Institute,* vol. 346, pp. 854-871, 2009.

[50] K. P. B. Chandra, H. Alwi and C. Edwards, "Fault Reconstruction for a Quadrotor Using an LPV Sliding Mode Observer," *International Federation of Automatic Control,* vol. 48, no. 21, pp. 374-379, 2015.

[51] H. Alwi and C. Edwards, "Robust fault reconstruction for linear parameter varying systems using sliding mode observers," *International Journal of Robust and Nonlinear Control,* vol. 24, pp. 1947-1968, 2013.

APPENDECIES

A. MATLAB Codes

A.1 LPV System Representation and LPV Control of Quadrotor

```
% MS Project (AE 295B)
% Design of a Linear Parameter Varying Controller for a Delivery Quadrotor
% Description: Script develops the LPV model and LPV controller for the
% delivery quadrotor
% Author: Hussam Okasha

%% Parameters
g = 9.81;
l = 0.6; %m
R = 0.15; %m
m_quadrotor = 3.800; %kg %all mass not including the motors, battery, and
payload
m_motor = 0.325;
m_battery = 3.673;
m_motors = 4*m_motor;
mp = 2;
m_base = m_quadrotor + m_motors + m_battery;

KF = 6.11e-8;
KM = 1.5e-9;

k = sqrt(g*(m_base+mp)/(4*KF));
hov_in = (m_base+mp)*g*[1 0 0 0];
omega_direct = k*[1 1 1 1];

psi0 = 0;

%% Linear, Parameter Dependent Model
syms mp psi0
%State Variable Selection
%[x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13]' = [X Y Z u v w phi theta psi p
q r]'
syms x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 u1 u2 u3 u4
x = [x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12];
%Control Input Selection
%[u1 u2 u3 u4]' = [U_z U_roll U_pitch U_yaw]'
u = [u1 u2 u3 u4];

M = m_quadrotor + m_battery + mp;
Jx = ((2*M*R^2)/5) + 2*(l^2)*m_motor;
Jy = ((2*M*R^2)/5) + 2*(l^2)*m_motor;
Jz = ((2*M*R^2)/5) + 4*(l^2)*m_motor;
[Al, Bl] = Qrotor_Linearization(x,u,m_base,mp,psi0,Jx,Jy,Jz);
C = eye(12);
D = zeros(12,4);
```

```
%Controllability and Observability
A = subs(Al,psi0,0);
A = double(A);
Bcc = subs(Bl,mp,0);
Co = ctrb(A,Bcc);
rankCo = rank(Co)
Ob = obsv(A,C);
rankOb = rank(Ob)

%% LPV Model
%Control Input Filter
Au = -100*eye(4);
Bu = eye(4);
Cu = eye(4);
Ap = [A, Bl*Cu;
      zeros(4,12), Au];
Ap = vpa(Ap,4);
Bp = [zeros(12,4);Bu];

%disturbance matrix
B1 = [zeros(5,4);
      1 0 0 0;
      zeros(3,4);
      zeros(3,1), eye(3)];

B1p = [B1; zeros(4,4)];

%define H
%input defn: w' = [u' w]'
Bpp = [Bp, B1p]; %dim 16x8

%Parameter functions
p1 = -1/(mp+8.773);
p2 = 1/(0.009*mp+0.3013);
p3 = 1/(0.009*mp+0.5353);

%Upper and lower bounds
p1_l = double(subs(p1,mp,0));
p1_u = double(subs(p1,mp,2.3));
p2_l = double(subs(p2,mp,0));
p2_u = double(subs(p2,mp,2.3));
p3_l = double(subs(p3,mp,0));
p3_u = double(subs(p3,mp,2.3));

P = [p1_l p1_u;
     p2_u p2_l;
     p3_u p3_l];

%Parameter Box
pv = pvec('box',P);
VERTX = polydec(pv); %vertex coordinates
pvinfo(pv)
%Vector of 3 parameters ranging in a box

%example of polydec function
```

```
[V_C,VERTX_C] = polydec(pv,[p1_u, p2_u, p3_u]);

%Augmented Model Intermediate Matrices
M1 = [1 0 0;
      0 1 0;
      0 0 -1];

M2 = [0 -g 0;
      g 0 0;
      0 0 0];

M3 = eye(3);

M4 = [1 0 0;
      0 1 0;
      0 0 0];

A0 = [zeros(16,3),[M1;zeros(13,3)],[zeros(3);M2;zeros(10,3)],[zeros(6,3);
M3;zeros(7,3)],[zeros(12,4);Au]];
A1 = [zeros(16,12),[zeros(5,1);1;zeros(10,1)],zeros(16,3)]; %1 corresponds to
rho1
A2 = [zeros(16,13),[zeros(9,3);M4;zeros(4,3)]]; %move over one column in B,
M4 corresponds to rho2
A3 = [zeros(16,15),[zeros(11,1);1;zeros(4,1)]]; %1 corresponds to rho3

B0 = Bpp;
B1 = zeros(16,8);
B2 = zeros(16,8);
B3 = zeros(16,8);

Cp = [eye(12),zeros(12,4)];
C0 = Cp;
C1 = zeros(12,16);
C2 = zeros(12,16);
C3 = zeros(12,16);

Dp = [zeros(12,4),zeros(12,4)];
D0 = Dp;
D1 = zeros(12,8);
D2 = zeros(12,8);
D3 = zeros(12,8);

S0 = ltisys(A0,B0,C0,D0);
S1 = ltisys(A1,B1,C1,D1,0);
S2 = ltisys(A2,B2,C2,D2,0);
S3 = ltisys(A3,B3,C3,D3,0);

pdsys = psys(pv,[S0,S1,S2,S3]); %affine parameter dependent system
psinfo(pdsys)
%Affine parameter-dependent model with 3 parameters (4 systems)
%Each system has 16 state(s), 8 input(s), and 12 output(s)

polysys = aff2pol(pdsys); %polytopic model - instances of pdsys (affine) at
the vertices of the box
psinfo(polysys)
```

```
%Polytopic model with 8 vertex systems
%Each system has 16 state(s), 8 input(s), and 12 output(s)
% [tau,P] = quadstab(polysys)
% open-loop unstable

%% LPV Controls Design
%% Weights Selection
%static gains and filters in TF form
sys_Wr1 = ltisys('tf',[1],[1]);
sys_Wr2 = ltisys('tf',[1],[1]);
sys_Wr3 = ltisys('tf',[1],[1]);
sys_Wr4 = ltisys('tf',[1],[1]);
sys_Wr5 = ltisys('tf',[1],[1]);
sys_Wr6 = ltisys('tf',[1],[1]);
sys_Wr7 = ltisys('tf',[1],[1]);
sys_Wr8 = ltisys('tf',[1],[1]);
sys_Wr9 = ltisys('tf',[1],[1]);
sys_Wr10 = ltisys('tf',[1],[1]);
sys_Wr11 = ltisys('tf',[1],[1]);
sys_Wr12 = ltisys('tf',[1],[1]);

sys_Wd1 = ltisys('tf',[1],[1]);
sys_Wd2 = ltisys('tf',[1],[1]);
sys_Wd3 = ltisys('tf',[1],[1]);
sys_Wd4 = ltisys('tf',[1],[1]);

cWp1 = 2.01;
cWp2 = 0.201;
% cWp1 = 20.1;
% cWp2 = 2.01;
% cWp1 = 0.201;
% cWp2 = 0.0201;
sys_Wp1 = ltisys('tf',[cWp1],[1, cWp2]);
sys_Wp2 = ltisys('tf',[cWp1],[1, cWp2]);
sys_Wp3 = ltisys('tf',[cWp1],[1, cWp2]);
sys_Wp4 = ltisys('tf',[cWp1],[1, cWp2]);
sys_Wp5 = ltisys('tf',[cWp1],[1, cWp2]);
sys_Wp6 = ltisys('tf',[cWp1],[1, cWp2]);
sys_Wp7 = ltisys('tf',[cWp1],[1, cWp2]);
sys_Wp8 = ltisys('tf',[cWp1],[1, cWp2]);
sys_Wp9 = ltisys('tf',[cWp1],[1, cWp2]);
% sys_Wp10 = ltisys('tf',[cWp1],[1, cWp2]);
% sys_Wp11 = ltisys('tf',[cWp1],[1, cWp2]);
% sys_Wp12 = ltisys('tf',[cWp1],[1, cWp2]);

sys_Wp10 = ltisys('tf',[2 0],[1, 8.5, 18]);
sys_Wp11 = ltisys('tf',[2 0],[1, 8.5, 18]);
sys_Wp12 = ltisys('tf',[2 0],[1, 8.5, 18]);

% numWu = [10 0];
% denWu = [1 100];
numWu =[9.678, 0.029, 0, 0];
denWu = [1, 1.206e4, 1.136e7, 1.066e10];
sys_Wu1 = ltisys('tf',numWu,denWu);
sys_Wu2 = ltisys('tf',numWu,denWu);
sys_Wu3 = ltisys('tf',numWu,denWu);
```

```
sys_Wu4 = ltisys('tf',numWu,denWu);

%MIMO TF matrices
sysWrg1 = sdiag(sys_Wr1,sys_Wr2,sys_Wr3,sys_Wr4,sys_Wr5,sys_Wr6);
sysWrg2 = sdiag(sys_Wr7,sys_Wr8,sys_Wr9,sys_Wr10,sys_Wr11,sys_Wr12);
Wr = sdiag(sysWrg1,sysWrg2);
Wd = sdiag(sys_Wd1,sys_Wd2,sys_Wd3,sys_Wd4);
sysWpg1 = sdiag(sys_Wp1,sys_Wp2,sys_Wp3,sys_Wp4,sys_Wp5,sys_Wp6);
sysWpg2 = sdiag(sys_Wp7,sys_Wp8,sys_Wp9,sys_Wp10,sys_Wp11,sys_Wp12);
Wp = sdiag(sysWpg1,sysWpg2);
Wu = sdiag(sys_Wu1,sys_Wu2,sys_Wu3,sys_Wu4);

figure,
splot(Wu,'sv')
grid on;
title('Singular Values - Control (Robustness) Weight W_u')
xlabel('Frequency')
ylabel('Magnitude')
set(findall(gcf,'type','line'),'linewidth',1);
set(gca,'fontsize',24);

figure,
splot(Wp,'sv')
grid on;
title('Singular Values - Performance (Sensitivity) Weight W_p')
xlabel('Frequency')
ylabel('Magnitude')
set(findall(gcf,'type','line'),'linewidth',1);
set(gca,'fontsize',24);
%% Generalized LPV Plant
% inputs = 'r(12);w(4)';
% outputs = 'Wr;Wd;Wp;Wu';
% K_in = 'K:[e=Wr:r-G;Wr:r]'; %controller K with its inputs
% %G:K means the input of G is the output of K
% G1_in = 'G:[K;Wd:w]'; %g1 = pdsys
% G2_in = 'Wr:r'; %g2 = Wr
% G3_in = 'Wd:w'; %g3 = Wd
% G4_in = 'Wp:e'; %g4 = Wp
% G5_in = 'Wu:K'; %g5 = Wu
%
% [P_aug,N_MC] =
sconnect(inputs,outputs,K_in,G1_in,pdsys,G2_in,Wr,G3_in,Wd,G4_in,Wp,G5_in,Wu)
;

inputs = 'r(12);w(4)';
outputs = 'Wr;Wd;Wp;Wu;';
K_in = 'K:e=Wr-G;Wr'; %controller K with its inputs
%G:K means the input of G is the output of K
G1_in = 'G:K;Wd'; %g1 = pdsys
G2_in = 'Wr:r'; %g2 = Wr
G3_in = 'Wd:w'; %g3 = Wd
G4_in = 'Wp:e'; %g4 = Wp
G5_in = 'Wu:K'; %g5 = Wu
```

```
[P_aug,N_MC] =
sconnect(inputs,outputs,K_in,G1_in,pdsys,G2_in,Wr,G3_in,Wd,G4_in,Wp,G5_in,Wu)
;

psinfo(P_aug)
%N_MC = [nbr of measurements (C(s) inputs), nbr of controls (C(s) outputs))
%expectation: [24 4] --> gain K [4 24]
%pdP = Hinf plant P(s) associated with the control structure
%Polytopic model with 8 vertex systems
%Each system has 43 state(s), 20 input(s), and 56 output(s)

%% Gain-Scheduled Hinf Controller
[gopt,pdK,R,S] = hinfgs(P_aug,N_MC,0,1e-2);
psinfo(pdK)
%Polytopic model with 8 vertex systems
%Each system has 43 state(s), 24 input(s), and 4 output(s)

pCL = slft(P_aug,pdK); %closed-loop system
psinfo(pCL)
%Polytopic model with 8 vertex systems
%Each system has 86 state(s), 16 input(s), and 32 output(s)

%Performance Analysis
% [PERF, LP] = quadperf(pCL)

%Stability Analysis
% [TAU, LyP] = quadstab(pCL)
% pdlstab(pCL)
%NOTE: very long computation time for pdlstab, comment out when not needed

%Vertex Controllers
VK1 = psinfo(pdK,'sys',1); VK2 = psinfo(pdK,'sys',2);
VK3 = psinfo(pdK,'sys',3); VK4 = psinfo(pdK,'sys',4);
VK5 = psinfo(pdK,'sys',5); VK6 = psinfo(pdK,'sys',6);
VK7 = psinfo(pdK,'sys',7); VK8 = psinfo(pdK,'sys',8);

%Evaluate the eigenvalues of the Ak matrices of the vertex controllers
VK = {VK1,VK2,VK3,VK4,VK5,VK6,VK7,VK8};
Vertex_Eig = cell(8,1);
for i = 1:8
    VKe = VK{i};
    [Ak, Bk, Ck, Dk] = ltiss(VKe);
    Vertex_Eig{i} = eig(Ak);
end

%check eigenvalues are less than 0
Vertex_Stable = cell(8,1);
for i = 1:8
    Vertex_Eig1 = Vertex_Eig{i};
    Vertex_Stable{i} = Vertex_Eig1 < 0;
end
```

```
[Ak, Bk, Ck, Dk] = ltiss(VK1);
size(Ak); %43x43
size(Bk); %43x24
size(Ck); %4x43
size(Dk); %4x24

%example of psinfo function - to be implemented in Simulink
SKsys = psinfo(pdK,'eval',V_C); %instantiates the polytopic system for the
vertex controllers
[Ak, Bk, Ck, Dk] = ltiss(SKsys);

%% Frequency Domain Analysis - Singular Values Plots
%random polytopic coordinates for control analysis
pNum = 100; polyc = [];
for j = 1:pNum
   poly = rand(1,8);
   poly = poly/sum(poly);
   polyc = [polyc; poly];
end
%singular values plot for polytopic plant
figure,
omega = logspace(-2,2,200);
for j = 1:pNum
    PolySys = psinfo(polysys,'eval',polyc(j,:)); %evaluate at convex
coordinates
    [adp,bdp,cdp,ddp] = ltiss(PolySys);
    sysp = ss(adp,bdp,cdp,ddp);
    [sv] = sigma(sysp, omega);
    semilogx(omega, mag2db(sv));
    hold on; grid on;
    title('Singular Values - Plant G(\rho)')
    xlabel('Frequency [rad/s]')
    ylabel('Magnitude [dB]')
    set(findall(gcf,'type','line'),'linewidth',1);
    set(gca,'fontsize',24);
end

%singular values plot for Hinf plant
figure,
omega = logspace(-2,2,200);
for j = 1:pNum
    Pdg = psinfo(P_aug,'eval',polyc(j,:)); %evaluate at convex coordinates
    [adg,bdg,cdg,ddg] = ltiss(Pdg);
    sysg = ss(adg,bdg,cdg,ddg);
    [sv] = sigma(sysg, omega);
    semilogx(omega, mag2db(sv));
    hold on; grid on;
    title('Singular Values - Augmented Plant P(\rho)')
    xlabel('Frequency [rad/s]')
    ylabel('Magnitude [dB]')
    set(findall(gcf,'type','line'),'linewidth',1);
    set(gca,'fontsize',24);
end

%singular values plot for polytopic controller
figure,
omega = logspace(-4,4,300);
for j = 1:pNum
    Pdk = psinfo(pdK,'eval',polyc(j,:)); %evaluate at convex coordinates
    [adk,bdk,cdk,ddk] = ltiss(Pdk);
    sysk = ss(adk,bdk,cdk,ddk);
    [sv] = sigma(sysk, omega);
    semilogx(omega, mag2db(sv));
    hold on; grid on;
    title('Singular Values - Controller K(\rho)')
    xlabel('Frequency [rad/s]')
    ylabel('Magnitude [dB]')
    set(findall(gcf,'type','line'),'linewidth',1);
    set(gca,'fontsize',24);
end

%singular values plot for closed loop transfer system
figure,
omega = logspace(-4,4,300);
for j = 1:pNum
    Pcl = psinfo(pCL,'eval',polyc(j,:)); %evaluate at convex coordinates
    [adcl,bdcl,cdcl,ddcl] = ltiss(Pcl);
    syscl = ss(adcl,bdcl,cdcl,ddcl);
    [sv] = sigma(syscl, omega);
    semilogx(omega, mag2db(sv));
    hold on; grid on;
    title('Singular Values - Closed Loop System F(\rho)')
    xlabel('Frequency [rad/s]')
    ylabel('Magnitude [dB]')
    set(findall(gcf,'type','line'),'linewidth',1);
    set(gca,'fontsize',24);
end

%% Time Domain Analysis
%Plots output trajectory of closed-loop system along parameter trajectories
[T,X,Y] = pdsimul(pCL,'Mass_Traj',2,'Input_Traj');
figure,
plot(T,X)
title('State Trajectories')
grid on;
xlabel('Time [s]')
ylabel('Magnitude')
set(findall(gcf,'type','line'),'linewidth',1);
set(gca,'fontsize',14);
figure,
plot(T,Y)
title('Output Trajectories')
grid on;
xlabel('Time [s]')
ylabel('Magnitude')
set(findall(gcf,'type','line'),'linewidth',1);
set(gca,'fontsize',24);

Mass = cell(length(T),1);
Pa1 = zeros(length(T),1); Pa2 = zeros(length(T),1); Pa3 = zeros(length(T),1);
for i = 1:length(T)
Mass{i} = Mass_Traj(T(i));
```

```
Mass1 = Mass{i};
Pa1(i) = Mass1(1);
Pa2(i) = Mass1(2);
Pa3(i) = Mass1(3);
end

figure,
plot(T,Pa1)
hold on
plot(T,Pa2)
hold on
plot(T,Pa3)
title('Parameter Function Trajectories')
grid on;
xlabel('Time [s]')
ylabel('Magnitude')
set(findall(gcf,'type','line'),'linewidth',1);
set(gca,'fontsize',14);
size(X); %131x86
size(Y); %131x32
%Comment: further iterations might be necessary when testing against
%nonlinear system

%% Functions
%% Jacobian Linearization
function [Al, Bl] = Qrotor_Linearization(x,u,m_base,mp,psi0,Jx,Jy,Jz)
g = 9.81;
syms x0 y0 z0

xdot = sym(zeros(12,1));
xdot(1) = (cos(x(9))*cos(x(8)))*x(4) + (-
sin(x(9))*cos(x(7))+cos(x(9))*sin(x(8))*sin(x(7)))*x(5) +
(sin(x(9))*sin(x(7))+cos(x(9))*sin(x(8))*cos(x(7)))*x(6);
xdot(2) = (sin(x(9))*cos(x(8)))*x(4) +
(cos(x(9))*cos(x(7))+sin(x(9))*sin(x(8))*sin(x(7)))*x(5) + (-
cos(x(9))*sin(x(7))+sin(x(9))*sin(x(8))*cos(x(7)))*x(6);
xdot(3) = (sin(x(8)))*x(4) + (-cos(x(8))*sin(x(7)))*x(5) + (-
cos(x(8))*cos(x(7)))*x(6);
xdot(4) = (x(5)*x(12)-x(6)*x(11)) - g*sin(x(8));
xdot(5) = (x(6)*x(10)-x(4)*x(12)) + g*cos(x(8))*sin(x(7));
xdot(6) = (x(4)*x(11)-x(5)*x(10)) + g*cos(x(8))*cos(x(7)) -
(1/(m_base+mp))*u(1);
xdot(7) = x(10) + (sin(x(7))*tan(x(8)))*x(11) + (cos(x(7))*tan(x(8)))*x(12);
xdot(8) = cos(x(7))*x(11) + -sin(x(7))*x(12);
xdot(9) = (sin(x(7))/cos(x(8)))*x(11) + (cos(x(7))/cos(x(8)))*x(12);
xdot(10) = ((Jy-Jz)/Jx)*x(11)*x(12) + (1/Jx)*u(2);
xdot(11) = ((Jz-Jx)/Jy)*x(10)*x(12) + (1/Jy)*u(3);
xdot(12) = ((Jx-Jy)/Jz)*x(10)*x(11) + (1/Jz)*u(4);

Ax = jacobian(xdot,x);
Bu = jacobian(xdot,u);
xe =[x0,y0,z0,0,0,0,0,0,psi0,0,0,0]; %operating point at hover condition
ue = (m_base+mp)*g*[1 0 0 0];
Al = subs(Ax,x,xe);
Al = subs(Al,u,ue); %propeller speeds at hover conditions
Al = vpa(Al,4); %linearized A
```

```
Bl = subs(Bu,x,xe);
Bl = subs(Bl,u,ue);
Bl = vpa(Bl,4); %linearized B
end

%% Parameter trajectory for pdsimul function
%sample parameter trajectory
function mass = Mass_Traj(t)
mass = zeros(3,1);
mass(1) = -1/(t+8.773);
mass(2) = 1/(0.009*t+0.3013);
mass(3) = 1/(0.009*t+0.5353);
end

%% Input trajectory for pdsimul function
%Step inputs applied to each input
function UT = Input_Traj(t)
UT = ones(16,1);
end
```

A.2 Propeller Speed Based LPV Model and Controller

```
% MS Project (AE 295B)
% Design of a Linear Parameter Varying Controller for a Delivery Quadrotor
% Description: Script develops the LPV model and LPV controller for the
% delivery quadrotor with the propeller speeds as control inputs
% Author: Hussam Okasha

%% Parameters
g = 9.81;
l = 0.6; %m
R = 0.15; %m
m_quadrotor = 3.800; %kg %all mass not including the motors, battery, and
payload
m_motor = 0.325;
m_battery = 3.673;
m_motors = 4*m_motor;
mp = 2;
m_base = m_quadrotor + m_motors + m_battery;
psi0 = 0;
hov_in = (m_base+mp)*g*[1 0 0 0]; %test input
KF = 6.11e-8;
KM = 1.5e-9;
k = sqrt(g*(m_base+mp)/(4*KF));
omega_direct = k*[1 1 1 1];

%% Linear, Parameter Dependent Model
syms mp psi0
%State Variable Selection
%[x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13]' = [X Y Z u v w phi theta psi p
q r]'
syms x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 u1 u2 u3 u4
x = [x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12];
%Control Input Selection
%[u1 u2 u3 u4]' = [Omega_f Omega_r Omega_b Omega_l]'
u = [u1 u2 u3 u4];

M = m_quadrotor + m_battery + mp;
Jx = ((2*M*R^2)/5) + 2*(l^2)*m_motor;
Jy = ((2*M*R^2)/5) + 2*(l^2)*m_motor;
Jz = ((2*M*R^2)/5) + 4*(l^2)*m_motor;
[Al, Bl] = Qrotor_Linearization(x,u,l,m_base,mp,psi0,Jx,Jy,Jz);
C = eye(12);
D = zeros(12,4);

%Controllability and Observability
A = subs(Al,psi0,0);
A = double(A);
Bcc = subs(Bl,mp,0);
Co = ctrb(A,Bcc);
rankCo = rank(Co)
Ob = obsv(A,C);
rankOb = rank(Ob)
```

```
%% LPV Model
%Control Input Filter
Au = -100*eye(4);
Bu = eye(4);
Cu = eye(4);
Ap = [A, Bl*Cu;
      zeros(4,12), Au];
Ap = vpa(Ap,4);
Bp = [zeros(12,4);Bu];

%disturbance matrix
B1 = [zeros(5,4);
      1 0 0 0;
      zeros(3,4);
      zeros(3,1), eye(3)];

B1p = [B1; zeros(4,4)];

%define H
%input defn: w' = [u' w]'
Bpp = [Bp, B1p]; %dim 16x8

%Parameter functions
p1 = -(1.222e-7*(4.014e+7*mp + 3.521e+8)^(1/2))/(mp + 8.773);
p2 = (7.332e-8*(4.014e+7*mp + 3.521e+8)^(1/2))/(0.009*mp + 0.3013);
p3 = (3.0e-9*(4.014e+7*mp + 3.521e+8)^(1/2))/(0.009*mp + 0.5353);

%Upper and lower bounds
p1_l = double(subs(p1,mp,0));
p1_u = double(subs(p1,mp,2.3));
p2_l = double(subs(p2,mp,0));
p2_u = double(subs(p2,mp,2.3));
p3_l = double(subs(p3,mp,0));
p3_u = double(subs(p3,mp,2.3));

P = [p1_l p1_u;
     p2_l p2_u;
     p3_l p3_u];

%Parameter Box
pv = pvec('box',P);
VERTX = polydec(pv); %vertex coordinates
pvinfo(pv)
%Vector of 3 parameters ranging in a box

%example of polydec function
[V_C,VERTX_C] = polydec(pv,[p1_u, p2_u, p3_u]);

%Augmented Model Intermediate Matrices
M1 = [1 0 0;
      0 1 0;
      0 0 -1];

M2 = [0 -g 0;
      g 0 0;
      0 0 0];

M3 = eye(3);

M4 = [0 -1 0 1;
      1 0 -1 0];

A0 = [zeros(16,3),[M1;zeros(13,3)],[zeros(3);M2;zeros(10,3)],[zeros(6,3);
M3;zeros(7,3)],[zeros(12,4);Au]];
A1 = [zeros(16,12),[zeros(5,4); [1 1 1 1]; zeros(10,4)]]; %corresponds to
rho1
A2 = [zeros(16,12),[zeros(9,4);M4;zeros(5,4)]]; %M4 corresponds to rho2
A3 = [zeros(16,12),[zeros(11,4);[-1 1 -1 1];zeros(4,4)]]; %corresponds to
rho3

B0 = Bpp;
B1 = zeros(16,8);
B2 = zeros(16,8);
B3 = zeros(16,8);

Cp = [eye(12),zeros(12,4)];
C0 = Cp;
C1 = zeros(12,16);
C2 = zeros(12,16);
C3 = zeros(12,16);

Dp = [zeros(12,4),zeros(12,4)];
D0 = Dp;
D1 = zeros(12,8);
D2 = zeros(12,8);
D3 = zeros(12,8);

S0 = ltisys(A0,B0,C0,D0);
S1 = ltisys(A1,B1,C1,D1,0);
S2 = ltisys(A2,B2,C2,D2,0);
S3 = ltisys(A3,B3,C3,D3,0);

pdsys = psys(pv,[S0,S1,S2,S3]); %affine parameter dependent system
psinfo(pdsys)
%Affine parameter-dependent model with 3 parameters (4 systems)
%Each system has 16 state(s), 8 input(s), and 12 output(s)

polysys = aff2pol(pdsys); %polytopic model - instances of pdsys (affine) at
the vertices of the box
psinfo(polysys)
%Polytopic model with 8 vertex systems
%Each system has 16 state(s), 8 input(s), and 12 output(s)
% [tau,P] = quadstab(polysys)
% open-loop unstable

%% LPV Controls Design
%% Weights Selection
%static gains and filters in TF form
sys_Wr1 = ltisys('tf',[1],[1]);
sys_Wr2 = ltisys('tf',[1],[1]);
```

```
sys_Wr3 = ltisys('tf',[1],[1]);
sys_Wr4 = ltisys('tf',[1],[1]);
sys_Wr5 = ltisys('tf',[1],[1]);
sys_Wr6 = ltisys('tf',[1],[1]);
sys_Wr7 = ltisys('tf',[1],[1]);
sys_Wr8 = ltisys('tf',[1],[1]);
sys_Wr9 = ltisys('tf',[1],[1]);
sys_Wr10 = ltisys('tf',[1],[1]);
sys_Wr11 = ltisys('tf',[1],[1]);
sys_Wr12 = ltisys('tf',[1],[1]);

sys_Wd1 = ltisys('tf',[1],[1]);
sys_Wd2 = ltisys('tf',[1],[1]);
sys_Wd3 = ltisys('tf',[1],[1]);
sys_Wd4 = ltisys('tf',[1],[1]);

cWp1 = 2.01;
cWp2 = 0.201;
% cWp1 = 20.1;
% cWp2 = 2.01;
% cWp1 = 0.201;
% cWp2 = 0.0201;
sys_Wp1 = ltisys('tf',[cWp1],[1, cWp2]);
sys_Wp2 = ltisys('tf',[cWp1],[1, cWp2]);
sys_Wp3 = ltisys('tf',[cWp1],[1, cWp2]);
sys_Wp4 = ltisys('tf',[cWp1],[1, cWp2]);
sys_Wp5 = ltisys('tf',[cWp1],[1, cWp2]);
sys_Wp6 = ltisys('tf',[cWp1],[1, cWp2]);
sys_Wp7 = ltisys('tf',[cWp1],[1, cWp2]);
sys_Wp8 = ltisys('tf',[cWp1],[1, cWp2]);
sys_Wp9 = ltisys('tf',[cWp1],[1, cWp2]);
% sys_Wp10 = ltisys('tf',[cWp1],[1, cWp2]);
% sys_Wp11 = ltisys('tf',[cWp1],[1, cWp2]);
% sys_Wp12 = ltisys('tf',[cWp1],[1, cWp2]);

sys_Wp10 = ltisys('tf',[2 0],[1, 8.5, 18]);
sys_Wp11 = ltisys('tf',[2 0],[1, 8.5, 18]);
sys_Wp12 = ltisys('tf',[2 0],[1, 8.5, 18]);

% numWu = [10 0];
% denWu = [1 100];
numWu =[9.678, 0.029, 0, 0];
denWu = [1, 1.206e4, 1.136e7, 1.066e10];
% numWu = [0.5, 0.5*0.0001];
% denWu = [1 10];
sys_Wu1 = ltisys('tf',numWu,denWu);
sys_Wu2 = ltisys('tf',numWu,denWu);
sys_Wu3 = ltisys('tf',numWu,denWu);
sys_Wu4 = ltisys('tf',numWu,denWu);

%MIMO TF matrices
sysWrg1 = sdiag(sys_Wr1,sys_Wr2,sys_Wr3,sys_Wr4,sys_Wr5,sys_Wr6);
sysWrg2 = sdiag(sys_Wr7,sys_Wr8,sys_Wr9,sys_Wr10,sys_Wr11,sys_Wr12);
Wr = sdiag(sysWrg1,sysWrg2);
Wd = sdiag(sys_Wd1,sys_Wd2,sys_Wd3,sys_Wd4);
sysWpg1 = sdiag(sys_Wp1,sys_Wp2,sys_Wp3,sys_Wp4,sys_Wp5,sys_Wp6);
```

```
sysWpg2 = sdiag(sys_Wp7,sys_Wp8,sys_Wp9,sys_Wp10,sys_Wp11,sys_Wp12);
Wp = sdiag(sysWpg1,sysWpg2);
Wu = sdiag(sys_Wu1,sys_Wu2,sys_Wu3,sys_Wu4);

figure,
splot(Wu,'sv')
grid on;
title('Singular Values - Control (Robustness) Weight W_u')
xlabel('Frequency')
ylabel('Magnitude')
set(findall(gcf,'type','line'),'linewidth',1);
set(gca,'fontsize',24);

figure,
splot(Wp,'sv')
grid on;
title('Singular Values - Performance (Sensitivity) Weight W_p')
xlabel('Frequency')
ylabel('Magnitude')
set(findall(gcf,'type','line'),'linewidth',1);
set(gca,'fontsize',24);
%% Generalized LPV Plant
% inputs = 'r(12);w(4)';
% outputs = 'Wr;Wd;Wp;Wu';
% K_in = 'K:[e=Wr:r-G;Wr:r]'; %controller K with its inputs
% %G:K means the input of G is the output of K
% G1_in = 'G:[K;Wd:w]'; %g1 = pdsys
% G2_in = 'Wr:r'; %g2 = Wr
% G3_in = 'Wd:w'; %g3 = Wd
% G4_in = 'Wp:e'; %g4 = Wp
% G5_in = 'Wu:K'; %g5 = Wu
%
% [P_aug,N_MC] =
sconnect(inputs,outputs,K_in,G1_in,pdsys,G2_in,Wr,G3_in,Wd,G4_in,Wp,G5_in,Wu)
;

inputs = 'r(12);w(4)';
outputs = 'Wr;Wd;Wp;Wu;';
K_in = 'K:e=Wr-G;Wr'; %controller K with its inputs
%G:K means the input of G is the output of K
G1_in = 'G:K;Wd'; %g1 = pdsys
G2_in = 'Wr:r'; %g2 = Wr
G3_in = 'Wd:w'; %g3 = Wd
G4_in = 'Wp:e'; %g4 = Wp
G5_in = 'Wu:K'; %g5 = Wu

[P_aug,N_MC] =
sconnect(inputs,outputs,K_in,G1_in,pdsys,G2_in,Wr,G3_in,Wd,G4_in,Wp,G5_in,Wu)
;

psinfo(P_aug)
%N_MC = [nbr of measurements (C(s) inputs), nbr of controls (C(s) outputs))
%expectation: [24 4] --> gain K [4 24]
%pdP = Hinf plant P(s) associated with the control structure
%Polytopic model with 8 vertex systems
%Each system has 40 state(s), 20 input(s), and 56 output(s)
```

```
%% Gain-Scheduled Hinf Controller
[gopt,pdK,R,S] = hinfgs(P_aug,N_MC,0,1e-2);
psinfo(pdK)
%Polytopic model with 8 vertex systems
%Each system has 43 state(s), 24 input(s), and 4 output(s)

pCL = slft(P_aug,pdK); %closed-loop system
psinfo(pCL)
%Polytopic model with 8 vertex systems
%Each system has 86 state(s), 16 input(s), and 32 output(s)

%Performance Analysis
% [PERF, LP] = quadperf(pCL)

%Stability Analysis
% [TAU, LyP] = quadstab(pCL)
% pdlstab(pCL)
%NOTE: very long computation time for pdlstab, comment out when not needed

%Vertex Controllers
VK1 = psinfo(pdK,'sys',1); VK2 = psinfo(pdK,'sys',2);
VK3 = psinfo(pdK,'sys',3); VK4 = psinfo(pdK,'sys',4);
VK5 = psinfo(pdK,'sys',5); VK6 = psinfo(pdK,'sys',6);
VK7 = psinfo(pdK,'sys',7); VK8 = psinfo(pdK,'sys',8);

%Evaluate the eigenvalues of the Ak matrices of the vertex controllers
VK = {VK1,VK2,VK3,VK4,VK5,VK6,VK7,VK8};
Vertex_Eig = cell(8,1);
for i = 1:8
    VKe = VK{i};
    [Ak, Bk, Ck, Dk] = ltiss(VKe);
    Vertex_Eig{i} = eig(Ak);
end

%check eigenvalues are less than 0
Vertex_Stable = cell(8,1);
for i = 1:8
    Vertex_Eig1 = Vertex_Eig{i};
    Vertex_Stable{i} = Vertex_Eig1 < 0;
end


[Ak, Bk, Ck, Dk] = ltiss(VK1);
size(Ak); %43x43
size(Bk); %43x24
size(Ck); %4x43
size(Dk); %4x24

%example of psinfo function - to be implemented in Simulink
SKsys = psinfo(pdK,'eval',V_C); %instantiates the polytopic system for the
vertex controllers
[Ak, Bk, Ck, Dk] = ltiss(SKsys);

%% Frequency Domain Analysis - Singular Values Plots
```

```
%random polytopic coordinates for control analysis
pNum = 100; polyc = [];
for j = 1:pNum
   poly = rand(1,8);
   poly = poly/sum(poly);
   polyc = [polyc; poly];
end

%singular values plot for polytopic plant
figure,
omega = logspace(-2,2,200);
for j = 1:pNum
    PolySys = psinfo(polysys,'eval',polyc(j,:)); %evaluate at convex
coordinates
    [adp,bdp,cdp,ddp] = ltiss(PolySys);
    sysp = ss(adp,bdp,cdp,ddp);
    [sv] = sigma(sysp, omega);
    semilogx(omega, mag2db(sv));
    hold on; grid on;
    title('Singular Values - Plant G(\rho)')
    xlabel('Frequency [rad/s]')
    ylabel('Magnitude [dB]')
    set(findall(gcf,'type','line'),'linewidth',1);
    set(gca,'fontsize',24);
end

%singular values plot for Hinf plant
figure,
omega = logspace(-2,2,200);
for j = 1:pNum
    Pdg = psinfo(P_aug,'eval',polyc(j,:)); %evaluate at convex coordinates
    [adg,bdg,cdg,ddg] = ltiss(Pdg);
    sysg = ss(adg,bdg,cdg,ddg);
    [sv] = sigma(sysg, omega);
    semilogx(omega, mag2db(sv));
    hold on; grid on;
    title('Singular Values - Augmented Plant P(\rho)')
    xlabel('Frequency [rad/s]')
    ylabel('Magnitude [dB]')
    set(findall(gcf,'type','line'),'linewidth',1);
    set(gca,'fontsize',24);
end

%singular values plot for polytopic controller
figure,
omega = logspace(-4,4,300);
for j = 1:pNum
    Pdk = psinfo(pdK,'eval',polyc(j,:)); %evaluate at convex coordinates
    [adk,bdk,cdk,ddk] = ltiss(Pdk);
    sysk = ss(adk,bdk,cdk,ddk);
    [sv] = sigma(sysk, omega);
    semilogx(omega, mag2db(sv));
    hold on; grid on;
    title('Singular Values - Controller K(\rho)')
    xlabel('Frequency [rad/s]')
    ylabel('Magnitude [dB]')
    set(findall(gcf,'type','line'),'linewidth',1);
    set(gca,'fontsize',24);
end

%singular values plot for closed loop transfer system
figure,
omega = logspace(-4,4,300);
for j = 1:pNum
    Pcl = psinfo(pCL,'eval',polyc(j,:)); %evaluate at convex coordinates
    [adcl,bdcl,cdcl,ddcl] = ltiss(Pcl);
    syscl = ss(adcl,bdcl,cdcl,ddcl);
    [sv] = sigma(syscl, omega);
    semilogx(omega, mag2db(sv));
    hold on; grid on;
    title('Singular Values - Closed Loop System F(\rho)')
    xlabel('Frequency [rad/s]')
    ylabel('Magnitude [dB]')
    set(findall(gcf,'type','line'),'linewidth',1);
    set(gca,'fontsize',24);
end

%% Functions
%% Jacobian Linearization
function [Al, Bl] = Qrotor_Linearization(x,u,l,m_base,mp,psi0,Jx,Jy,Jz)
g = 9.81;
KF = 6.11e-8;
KM = 1.5e-9;
k = sqrt(g*(m_base+mp)/(4*KF));

syms x0 y0 z0

U_z = KF*(u(1)^2 +u(2)^2 + u(3)^2 + u(4)^2);
U_roll = l*KF*(-u(2)^2 + u(4)^2);
U_pitch = l*KF*(-u(3)^2 + u(1)^2);
U_yaw = KM*(-u(1)^2 + u(2)^2 - u(3)^2 + u(4)^2);

xdot = sym(zeros(12,1));
xdot(1) = (cos(x(9))*cos(x(8)))*x(4) + (-
sin(x(9))*cos(x(7))+cos(x(9))*sin(x(8))*sin(x(7)))*x(5) +
(sin(x(9))*sin(x(7))+cos(x(9))*sin(x(8))*cos(x(7)))*x(6);
xdot(2) = (sin(x(9))*cos(x(8)))*x(4) +
(cos(x(9))*cos(x(7))+sin(x(9))*sin(x(8))*sin(x(7)))*x(5) + (-
cos(x(9))*sin(x(7))+sin(x(9))*sin(x(8))*cos(x(7)))*x(6);
xdot(3) = (sin(x(8)))*x(4) + (-cos(x(8))*sin(x(7)))*x(5) + (-
cos(x(8))*cos(x(7)))*x(6);
xdot(4) = (x(5)*x(12)-x(6)*x(11)) - g*sin(x(8));
xdot(5) = (x(6)*x(10)-x(4)*x(12)) + g*cos(x(8))*sin(x(7));
xdot(6) = (x(4)*x(11)-x(5)*x(10)) + g*cos(x(8))*cos(x(7)) -
(1/(m_base+mp))*U_z;
xdot(7) = x(10) + (sin(x(7))*tan(x(8)))*x(11) + (cos(x(7))*tan(x(8)))*x(12);
xdot(8) = cos(x(7))*x(11) + -sin(x(7))*x(12);
xdot(9) = (sin(x(7))/cos(x(8)))*x(11) + (cos(x(7))/cos(x(8)))*x(12);
xdot(10) = ((Jy-Jz)/Jx)*x(11)*x(12) + (1/Jx)*U_roll;
xdot(11) = ((Jz-Jx)/Jy)*x(10)*x(12) + (1/Jy)*U_pitch;
xdot(12) = ((Jx-Jy)/Jz)*x(10)*x(11) + (1/Jz)*U_yaw;
```

```
Ax = jacobian(xdot,x);
Bu = jacobian(xdot,u);
xe =[x0,y0,z0,0,0,0,0,0,psi0,0,0,0]; %operating point at hover condition
ue = k*[1 1 1 1];
Al = subs(Ax,x,xe);
Al = subs(Al,u,ue); %propeller speeds at hover conditions
Al = vpa(Al,4); %linearized A
Bl = subs(Bu,x,xe);
Bl = subs(Bl,u,ue);
Bl = vpa(Bl,4); %linearized B
end
```

A.3 2DOF PI Actuator Control

```
% MS Project (AE 295B)
% Design of a Linear Parameter Varying Controller for a Delivery Quadrotor
% Description: MATLAB script for 2DOF PI Actuator Controller
% Author: Hussam Okasha

wref = 200;
tau = 0.00128;
cm = 20;

numa = cm;
dena = [tau 1];

zeta = 1;
wn = 100;

N = 4.58;
Kpi = (wn^2)*tau/cm;
Kpa = 2*wn*zeta*tau/cm;
Kpa = 1.15*Kpa;

Vd = [0, -5];
t = zeros(1519,2); u = zeros(1519,2); Vin = zeros(1519,2); wp =
zeros(1519,2);
outwr = zeros(1519,2);

open_system('PI_Control_Act.slx')

for i = 1:2
    Vdist = Vd(i);
    sim('PI_Control_Act.slx')
    t(:,i) = out.t;
    u(:,i) = out.uVin;
    Vin(:,i) = out.tVin;
    r(:,i) = out.wref;
    wp(:,i) = out.wp;
end

purple = [0.4940 0.1840 0.5560];
gold = [0.9290 0.6940 0.1250];

figure,
subplot(2,1,1)
    plot(t1{1},outu{1})
    hold on
    plot(t1{2},outv{2})
    title('Input Voltage')
    legend('Control Input',...
           'Control Input with V_d = -5V','location','best')
    xlabel('Time [s]');
    ylabel('Voltage [V]');
    set(findall(gcf,'type','line'),'linewidth',3);
    set(gca,'fontsize',24);
subplot(2,1,2)
    plot(t(:,1),r(:,1),'k')
```

```
hold on
plot(t(:,1),wp(:,1),'color',purple)
hold on
plot(t(:,2),wp(:,2),'color',gold,'LineStyle','--')
title('Propeller Speed')
legend('Reference \Omega_r',...
       '\Omega with V_d = 0V',...
       '\Omega with V_d = -5V','location','best')
xlabel('Time [s]');
ylabel('Propeller Speed [rad/s]');
set(findall(gcf,'type','line'),'linewidth',3);
set(gca,'fontsize',24);
```

A.4 Nonlinear Simulation of LPV Control System

```
% MS Project (AE 295B)
% Design of a Linear Parameter Varying Controller for a Delivery Quadrotor
% Description: Nonlinear simulations of LPV control systems
% Author: Hussam Okasha

%% Parameters

% Quadrotor Parameters
g = 9.81;
l = 0.6; %m
R = 0.15; %m
m_quadrotor = 3.800; %kg %all mass not including the motors, battery, and
payload
m_motor = 0.325;
m_battery = 3.673;
m_motors = 4*m_motor;
mp = 2; %actual payload mass for simulation purposes
m_base = m_quadrotor + m_motors + m_battery;
psi0 = 0;

% Actuator Parameters
KF = 6.11e-8;
KM = 1.5e-9;
cm = 20; %s^-1
tau = 0.00128;
% Vdist = 0;
% Vdist = -5;
[numa, dena, N, Kpi, Kpa] = ActuatorPI(cm,tau);

% Mass Estimator Parameters
Lambda_0 = 10;
Gamma = 15;
Est_IC = 1/m_base;

%System Parameters for propeller based model
% LinModel = load('sysmodels.mat');
% Alin = LinModel.Al;
% Blin = LinModel.Bl;

% LPV Controller Parameters
%lpvDFT.mat - Forces and Torques
%lpvPropeller.mat - Propeller Speeds
LPVcontrol = load('lpvPropeller.mat');
pv = LPVcontrol.pv; %parameter box
pdK = LPVcontrol.pdK; %polytopic vertex controllers

% Test Inputs
k = sqrt(g*(m_base+mp)/(4*KF));
hov_in = (m_base+mp)*g*[1 0 0 0];
omega_direct = k*[1 1 1 1];
```

```
% Linearized A and B for linear simulation
% run LPV_Control_Quadrotor.m or LPV_Control_Quadrotor_2.m first
syms psi0 mp
Alin = subs(Al,psi0,0);
Blin = subs(Bl,mp,2);
Alin = double(Alin);
Blin = double(Blin);
psi = 0;
mp = 2;

% Trajectory gain for linear simulation
Cr = [1.321, zeros(1,11);
      0, 1.321 zeros(1,10);
      0, 0, 1.361, zeros(1,9);
      zeros(9,3), eye(9)];

% Reference Model (Trajectory Filter)
tau_f = 0.1;
% tau_f = 0.06;
Aref = -tau_f*eye(12);
Bref = eye(12);
Cref = tau_f*eye(12);
Dref = zeros(12);
x0ref = zeros(1,12);

syms s
Gr = Cref*inv(s*eye(12)-Aref)*Bref+Dref;
Gr_sys = ltisys('tf',[3],[50 3]);
Gr_sysA = ltisys('tf',[1],[10 1]);
figure,
splot(Gr_sys,'bode')
grid on;
title('Bode Plot for Model Reference Signal')
set(findall(gcf,'type','line'),'linewidth',2);

figure,
splot(Gr_sysA,'bode')
grid on;
title('Bode Plot for Model Reference Signal')
set(findall(gcf,'type','line'),'linewidth',2);

%% Plots
% open_system('PayloadQuadrotor_Nonlinear2019b2.slx')
% sim('PayloadQuadrotor_Nonlinear2019b2.slx')

% open_system('PayloadQuadrotor_Nonlinear.slx')
% sim('PayloadQuadrotor_Nonlinear.slx')

%Mass Estimator Plot
massp = [(m_base+mp)*ones(length(out.t),1), mp*ones(length(out.t),1),
out.m_est, out.mp_est];
QrotorMassEstPlot(out.t,massp);
% print(gcf,'dist_massest.png','-dpng','-r600')
```

```
%Position Plot
QrotorPosPlot(out.t,out.r_out,out.qstates);
% print(gcf,'dist_3Dtraj.png','-dpng','-r600')
% print(gcf,'dist_poststates.png','-dpng','-r600')

%State Plots
QrotorPlotStates(out.t,out.r_out,out.qstates);
% print(gcf,'dist_vel.png','-dpng','-r600')
% print(gcf,'dist_Eul.png','-dpng','-r600')
% print(gcf,'dist_attr.png','-dpng','-r600')

%Control Plots
QrotorPlotControl(out.t,out.ctrl,out.lpvcmd,out.lpvcmds);
% print(gcf,'dist_LPV.png','-dpng','-r600')
% print(gcf,'dist_scaledLPV.png','-dpng','-r600')
% print(gcf,'dist_actinputs.png','-dpng','-r600')

%to save high res images, with the figure of interest open, type in command
%prompt: print(gcf,'filename.png','-dpng','-r600') %600 = dots per inch

%% Functions

function [] = QrotorPosPlot(t,r,states)
r1 = r(:,1);
r2 = r(:,2);
r3 = r(:,3);
X = states(:,1);
Y = states(:,2);
Z = states(:,3);

figure('Position', get(0, 'Screensize'));
plot3(r1,r2,r3,'k');
hold on
plot3(X,Y,Z,'r--')
grid on
xlabel('North Position [m]');
ylabel('East Position [m]');
zlabel('Altitude [m]');
title('Reference Tracking');
legend('desired trajectory',...
       'position response','location','best')
set(findall(gcf,'type','line'),'linewidth',3);
set(gca,'fontsize',24);

figure('Position', get(0, 'Screensize'));
plot(t,X,t,Y,':',t,Z);
hold on
plot(t,r1,'--',t,r2,':',t,r3,'--')
grid on
xlabel('Time [s]');
ylabel('Position [m]');
title('Reference Tracking');
legend('X',...
       'Y',...
       'Z',...
```

```
        'desired X',...
        'desired Y',...
        'desired Z','location','best')
set(findall(gcf,'type','line'),'linewidth',3);
set(gca,'fontsize',24);
end

function [] = QrotorPlotStates(t,r,states)
udes = r(:,4);
vdes = r(:,5);
wdes = r(:,6);
Edes = r(:,7); %common to phi, theta, psi
pdes = r(:,10); %common to p q r

u = states(:,4);
v = states(:,5);
w = states(:,6);
phi = states(:,7)*180/pi;
theta = states(:,8)*180/pi;
psi = states(:,9)*180/pi;
p = states(:,10)*180/pi;
q = states(:,11)*180/pi;
r = states(:,12)*180/pi;

figure('Position', get(0, 'Screensize'));
plot(t,u,t,v,':',t,w)
hold on
plot(t,udes,'--',t,vdes,':',t,wdes,'--')
title('Velocity Responses')
legend('u',...
       'v',...
       'w',...
       'desired u',...
       'desired v',...
       'desired w','location','best')
xlabel('Time [s]')
ylabel('Velocity [m/s]')
set(findall(gcf,'type','line'),'linewidth',2);
set(gca,'fontsize',24);

figure('Position', get(0, 'Screensize'));
plot(t,phi,t,theta,t,psi)
hold on
plot(t,Edes,'k--')
title('Euler Angle Responses')
legend('\psi',...
       '\phi',...
       '\theta',...
       'desired Euler angle','location','best')
xlabel('Time [s]')
ylabel('Euler Angle [deg]')
set(findall(gcf,'type','line'),'linewidth',2);
set(gca,'fontsize',24);

figure('Position', get(0, 'Screensize'));
plot(t,p,t,q,t,r)
```

```
hold on
plot(t,pdes,'k--')
title('Attitude Rate Responses')
legend('p',...
       'q',...
       'r',...
       'desired attitude rate','location','best')
xlabel('Time [s]')
ylabel('Attitude Rate [deg/s]')
set(findall(gcf,'type','line'),'linewidth',2);
set(gca,'fontsize',24);
end

function [] = QrotorMassEstPlot(t,y)
y1 = y(:,1);
y2 = y(:,2);
y3 = y(:,3);
y4 = y(:,4);

figure('Position', get(0, 'Screensize'));
plot(t,y1,'k')
hold on
plot(t,y3,'r--')
hold on
plot(t,y2,'b')
hold on
plot(t,y4,'g--')
title('Gradient Descent Based Mass Estimator')
legend('actual m',...
       'estimated m',....
       'actual m_p',...
       'estimated m_p','location','best')
xlabel('Time [s]')
ylabel('Mass [kg]')
set(findall(gcf,'type','line'),'linewidth',3);
set(gca,'fontsize',24);
xlim([0, 1])
end

function [] = QrotorPlotControl(t,control,lpv,cmd)
u1 = control(:,1);
u2 = control(:,2);
u3 = control(:,3);
u4 = control(:,4);

lpv1 = lpv(:,1);
lpv2 = lpv(:,2);
lpv3 = lpv(:,3);
lpv4 = lpv(:,4);

cmd1 = cmd(:,1);
cmd2 = cmd(:,2);
cmd3 = cmd(:,3);
cmd4 = cmd(:,4);

figure('Position', get(0, 'Screensize'));
```

```
plot(t,u1,t,u2,t,u3,t,u4)
title('Control Inputs')
% legend('F_z [N]',...
%         '\tau_\phi [Nm]',...
%         '\tau_\theta [Nm]',...
%         '\tau_\psi [Nm]','location','best')
legend('\Omega_f',...
       '\Omega_r',...
       '\Omega_b',...
       '\Omega_l','location','best')
xlabel('Time [s]')
% ylabel('Magnitude')
ylabel('Angular Speed [rad/s]')
set(findall(gcf,'type','line'),'linewidth',2);
set(gca,'fontsize',24);

figure('Position', get(0, 'Screensize'));
plot(t,lpv1,t,lpv2,t,lpv3,t,lpv4)
title('LPV Commands')
% legend('F_z [N]',...
%         '\tau_\phi [Nm]',...
%         '\tau_\theta [Nm]',...
%         '\tau_\psi [Nm]','location','best')
legend('\Omega_f',...
       '\Omega_r',...
       '\Omega_b',...
       '\Omega_l','location','best')
xlabel('Time [s]')
% ylabel('Magnitude')
ylabel('Angular Speed [rad/s]')
set(findall(gcf,'type','line'),'linewidth',2);
set(gca,'fontsize',24);

figure('Position', get(0, 'Screensize'));
plot(t,cmd1,t,cmd2,t,cmd3,t,cmd4)
title('Scaled LPV Commands')
% legend('F_z [N]',...
%         '\tau_\phi [Nm]',...
%         '\tau_\theta [Nm]',...
%         '\tau_\psi [Nm]','location','best')
legend('\Omega_f',...
       '\Omega_r',...
       '\Omega_b',...
       '\Omega_l','location','best')
xlabel('Time [s]')
% ylabel('Magnitude')
ylabel('Angular Speed [rad/s]')
set(findall(gcf,'type','line'),'linewidth',2);
set(gca,'fontsize',24);
end

% Actuator PI Controller Parameters
function [numa, dena, N, Kpi, Kpa] = ActuatorPI(cm,tau)
numa = cm;
dena = [tau 1];
zeta = 1;
```

```
wn = 100;
N = 4.58;
Kpi = (wn^2)*tau/cm;
Kpa = 2*wn*zeta*tau/cm;
Kpa = 1.15*Kpa;
end
```

A.5 Reference Trajectory Build

```
%MATLAB Simulink function builds the reference trajectory
function traj = RefTraj(t)
%Parameters
h = 40; %heigh of the building [m]
delta = 10; %height of quadrotor above h [m]
Vp = 10; %lifting velocity [m/s]
Vm = 10; %cruise velocity [m/s]
t1 = (h+delta)/Vp; %time at waypoint 1 [s]
t2 = t1 + (1000*sqrt(2)/Vm); %time at waypoint 2 [s]
theta = 45; %angle between point AB and ground [deg]
tb = 20; %desired time to deliver payload from z=h+delta to z=40
Vh = 2*delta/tb; %see derivation

%r0 path
if t <= 0
   X = 0; Y = 0; Z = 0;
   r0 = [X, Y, Z];
   u = 0; v = 0; w =0;
   v0 = [u, v, w];
   Eul0 = [0, 0, 0];
   rates0 = [0, 0, 0];
   traj = [r0, v0, Eul0, rates0]';

%r1 path
elseif t > 0 && t <= t1
   X = 0; Y = 0;
   Z = Vp*t+0;
   r1 = [X, Y, Z];
   u = 0; v = 0;
   w = Vp;
   v1 = [u, v, w];
   Eul1 = [0, 0, 0];
   rates1 = [0, 0, 0];
   traj = [r1, v1, Eul1, rates1]';

%r2 path
elseif t > t1 && t <= t2
   X = (Vm/sqrt(2))*(t - t1);
   Y = (Vm/sqrt(2))*(t - t1);
   Z = h + delta;
   r2 = [X, Y, Z];
   u = Vm*cosd(theta);
   v = Vm*sind(theta);
   w = 0;
   v2 = [u, v, w];
   Eul2 = [0, 0, 0];
   rates2 = [0, 0, 0];
   traj = [r2, v2, Eul2, rates2]';

%r3 path
elseif t > t2 && t <= (t2 + tb)
   X = 1000; Y = 1000;
   Z = -Vh*(t - t2) + (Vh/(2*tb))*(t - t2)^2 + (h + delta);
```

```
    r3 = [X, Y, Z];
    u = 0; v = 0;
    w = -Vh + (Vh/tb)*(t - t2);
    v3 = [u, v, w];
    Eul3 = [0, 0, 0];
    rates3 = [0, 0, 0];
    traj = [r3, v3, Eul3, rates3]';

%r4 path
else % t == (t2 + tb)
    X = 1000; Y = 1000;
    Z = h;
    r4 = [X, Y, Z];
    u = 0 ; v = 0; w = 0;
    v4 = [u, v, w];
    Eul4 = [0, 0, 0];
    rates4 = [0, 0, 0];
    traj = [r4, v4, Eul4, rates4]';
end
traj(6) = -traj(6);
end
```

A.6 LPV Control Simulink MATLAB Functions

```
%Function for ConvexDecomp
%Workaround to avoid the code generation feature of the Simulink MATLAB
function
function alphas = myWrapper(pv,rho_m)
alphas = polydec(pv,[rho_m(1), rho_m(2), rho_m(3)]); %vertex coordinates
end
```

```
function alphas = ConvexDecomp(pv,rho_m)
alphas = zeros(1,8);
coder.extrinsic('myWrapper')
alphas = myWrapper(pv,rho_m);
end
```

```
%Function for VertexControl
%Workaround to avoid the code generation feature of the Simulink MATLAB
function
function [Ak,Bk,Ck,Dk] = myWrapper2(pdK,alphas)
SKsys = psinfo(pdK,'eval',alphas); %instantiates the polytopic system for
the vertex controllers
[Ak, Bk, Ck, Dk] = ltiss(SKsys);
end
```

```
function [Ak,Bk,Ck,Dk] = VertexControl(pdK,alphas)
Ak = zeros(43,43);
Bk = zeros(43,24);
Ck = zeros(4,43);
Dk = zeros(4,24);
coder.extrinsic('myWrapper2')
[Ak,Bk,Ck,Dk] = myWrapper2(pdK,alphas);
end
```

B. Simulink Structures for Quadrotor Simulation

B.1 State Variable Representation for Nonlinear System

STATE VARIABLE DEFINITIONS

$$\underline{x} = [X\; Y\; Z\; u\; v\; w\; \phi\; \theta\; \psi\; p\; q\; r]^T = [x_1\; x_2 \cdots x_{12}]^T$$
$$\underline{u} = \left[F_z\; \tau_\phi\; \tau_\theta\; \tau_\psi\right]^T = [u_1\; u_2\; u_3\; u_4]^T$$

KINEMATIC EQUATIONS – TRANSLATION

$$\dot{x_1} = (cosx_9cosx_8)x_4 + (-sinx_9cosx_7 + cosx_9sinx_8sinx_7)x_5 + (sinx_9sinx_7 + cosx_9sinx_8cosx_7)x_6$$
$$\dot{x_2} = (sinx_9cosx_8)x_4 + (cosx_9cosx_7 + sinx_9sinx_8sinx_7)x_5 + (-cosx_9sinx_7 + sinx_9sinx_8cosx_7)x_6$$
$$\dot{x_3} = (-sinx_8)x_4 + (cosx_8sinx_7)x_5 + (cosx_8cosx_7)x_6$$

FORCE EQUATIONS

$$\dot{x_4} = (x_5x_{12} - x_6x_{11}) - gsinx_8$$
$$\dot{x_5} = (x_6x_{10} - x_4x_{12}) + gcosx_8sinx_7$$
$$\dot{x_6} = (x_4x_{11} - x_5x_{10}) + gcosx_8cosx_7 - \frac{u_1}{m}$$

KINEMATIC EQUATIONS – ROTATION

$$\dot{x_7} = x_{10} + (sinx_7tanx_8)x_{11} + (cosx_7tanx_8)x_{12}$$
$$\dot{x_8} = (cosx_7)x_{10} + (-sinx_7)x_{12}$$
$$\dot{x_9} = (sinx_7/cosx_8)x_{11} + (cosx_7/cosx_8)x_{12}$$

MOMENT EQUATIONS

$$\dot{x_{10}} = \frac{J_y - J_z}{J_x}x_{11}x_{12} + \frac{u_2}{J_x}$$
$$\dot{x_{11}} = \frac{J_z - J_x}{J_y}x_{10}x_{12} + \frac{u_3}{J_y}$$
$$\dot{x_{12}} = \frac{J_x - J_y}{J_z}x_{10}x_{11} + \frac{u_4}{J_z}$$

CONTROL INPUT EQUATIONS

$$u_1 = K_F\left(\Omega_f{}^2 + \Omega_r{}^2 + \Omega_b{}^2 + \Omega_l{}^2\right)$$
$$u_2 = lK_F\left(-\Omega_r{}^2 + \Omega_l{}^2\right)$$
$$u_3 = lK_F\left(\Omega_f{}^2 - \Omega_b{}^2\right)$$
$$u_4 = K_M\left(-\Omega_f{}^2 + \Omega_r{}^2 - \Omega_b{}^2 + \Omega_l{}^2\right)$$

B.2 Nonlinear Simulink Subsystems

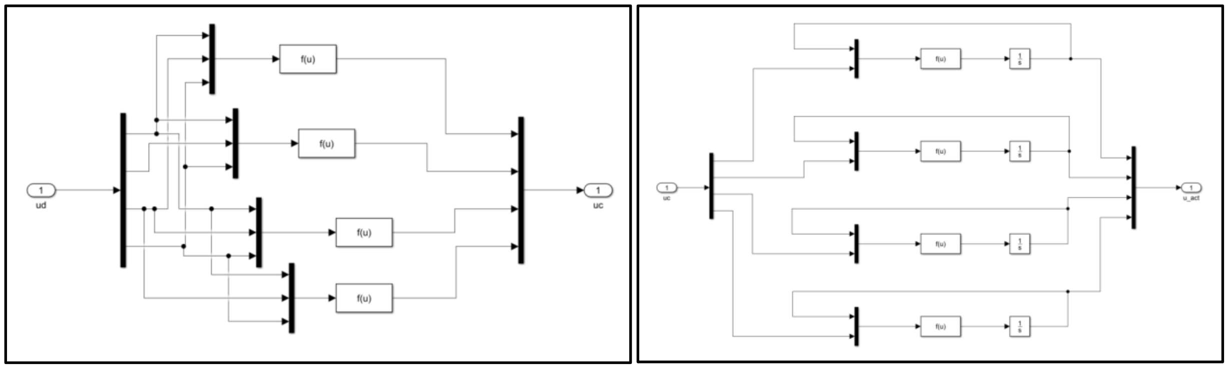
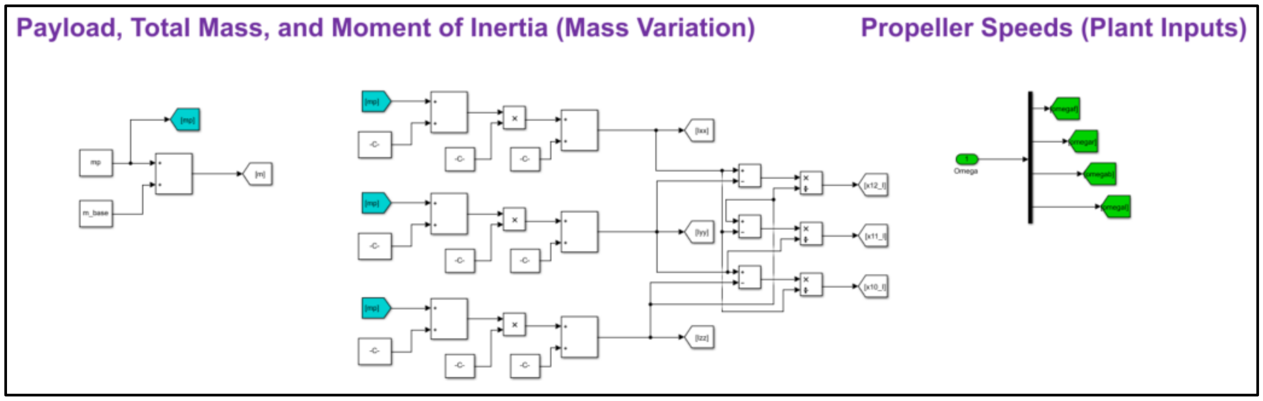


Figure B.1 – Motor mixing and actuator dynamics


Payload, Total Mass, and Moment of Inertia (Mass Variation)
Propeller Speeds (Plant Inputs)
[mp]
mp
m_base
[m]
-C-
[Ixx]
[Iyy]
[Izz]
[x12_I]
[x11_I]
[x10_I]
Omega
Figure B.2 – Mass characteristics and control inputs

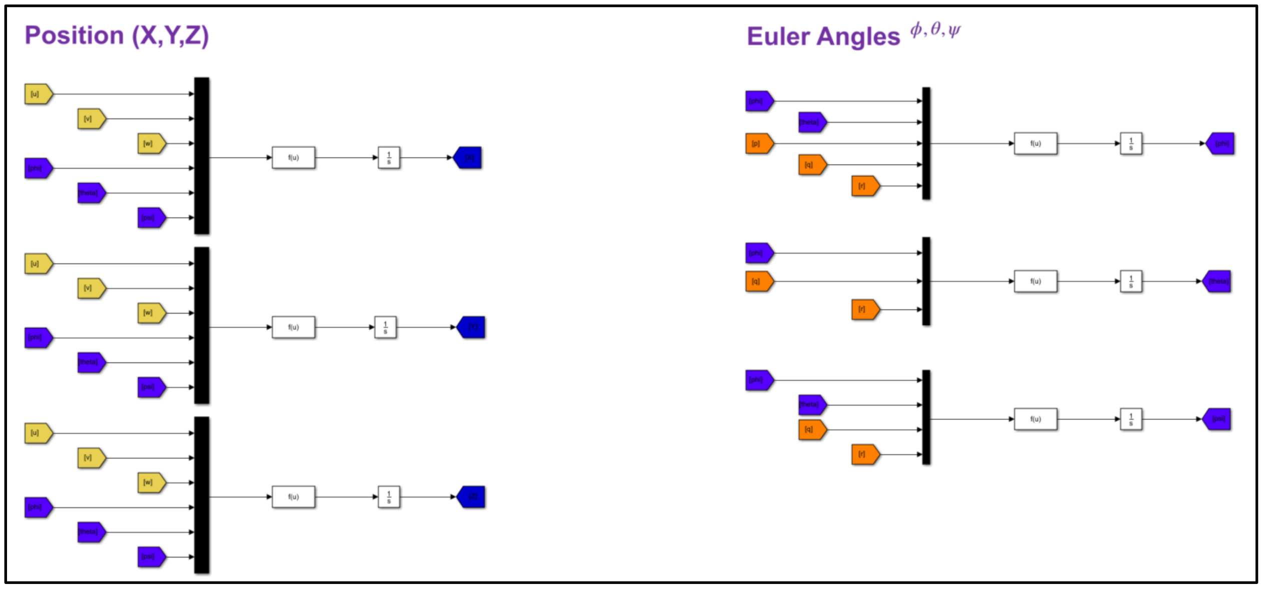


Figure B.3 – Position and Euler angles EOM

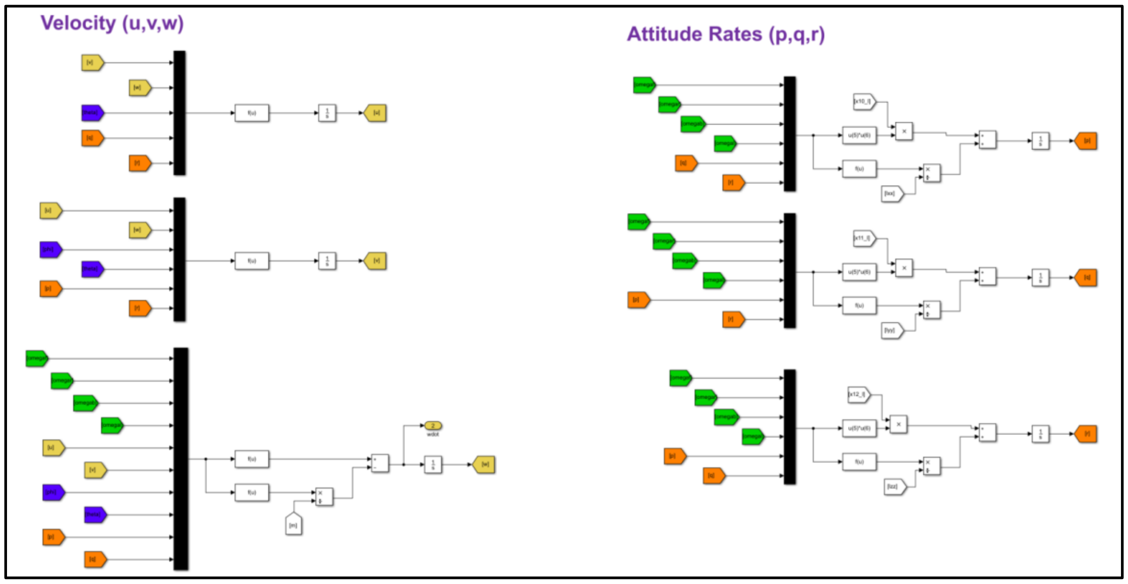


Figure B.4 – Velocity and attitude rates EOM

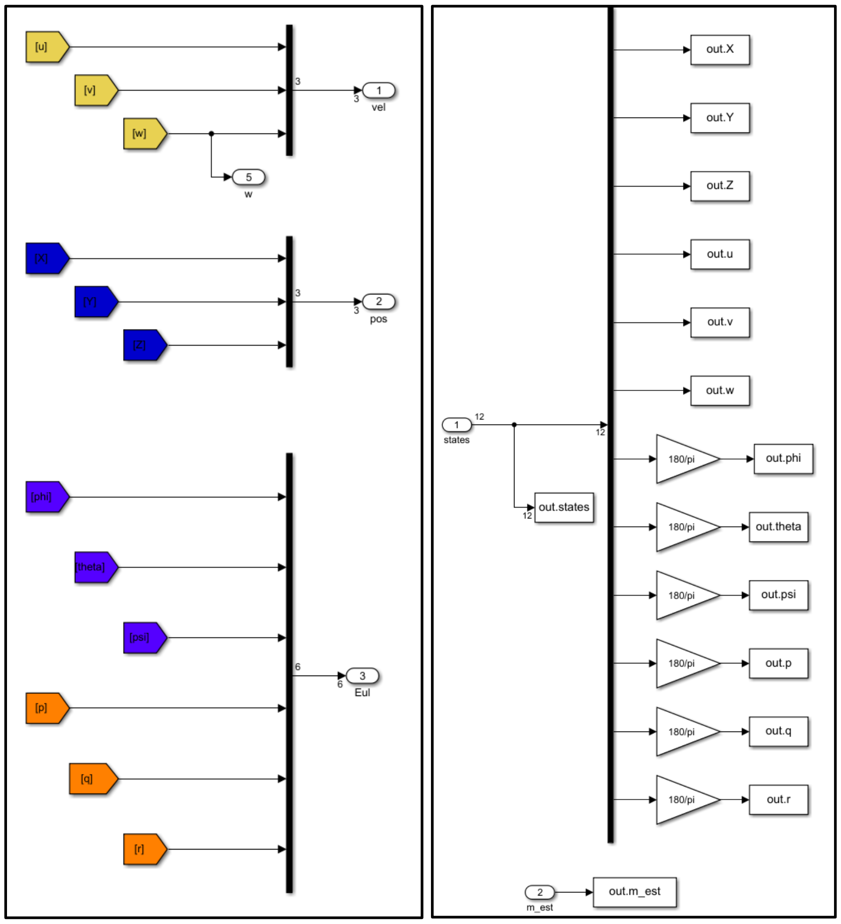


Figure B.5 – Model outputs and data logging

B.3 Actuator Controller

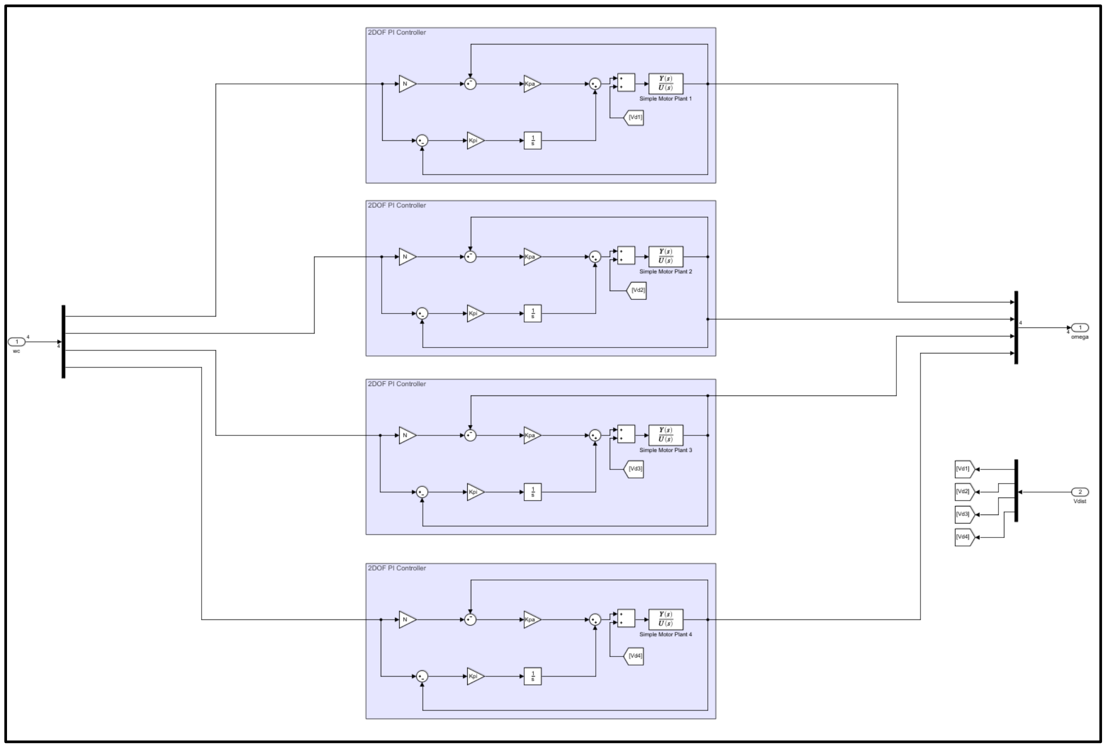


Figure B.6 – 2DOF PI Actuator Controller

B.4 Mass Estimator and Hover State Conditioning

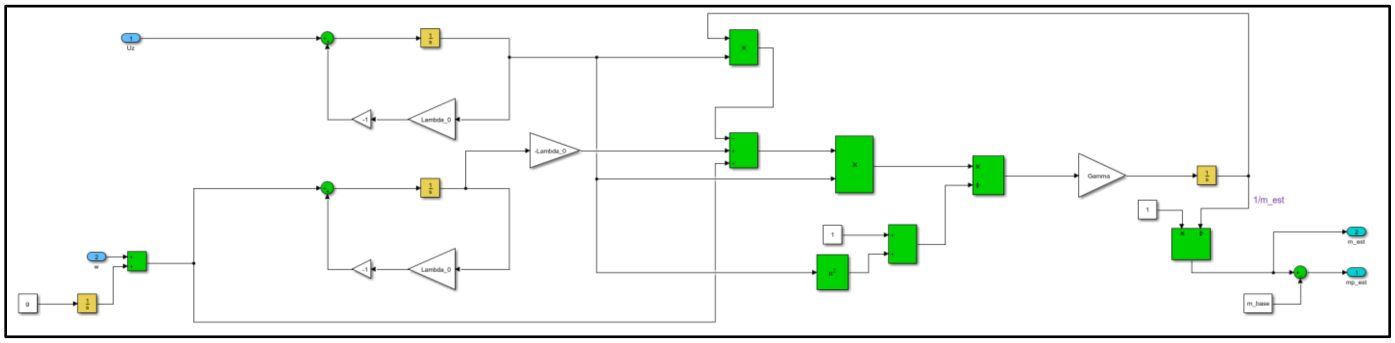


Figure B.7 – Adaptive mass estimator based on gradient descent law

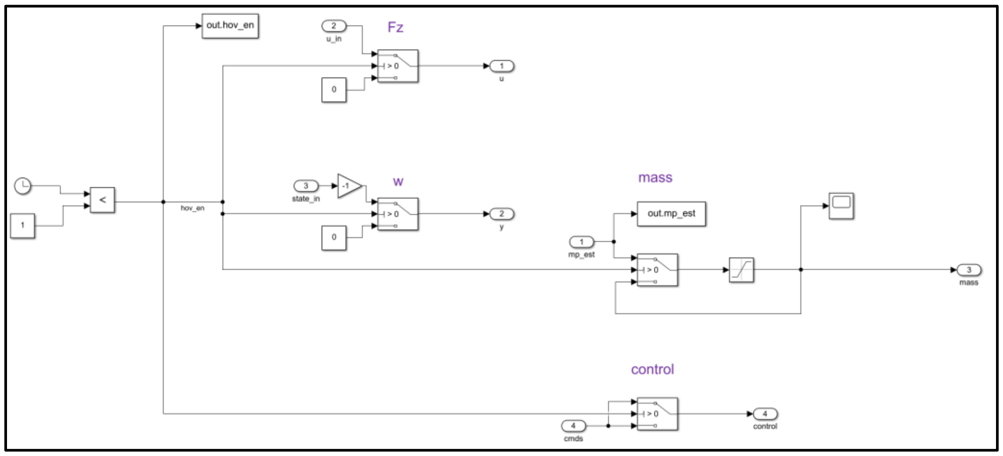


Figure B.8 – Hover state conditioning subsystem

B.5 LPV Control Structures

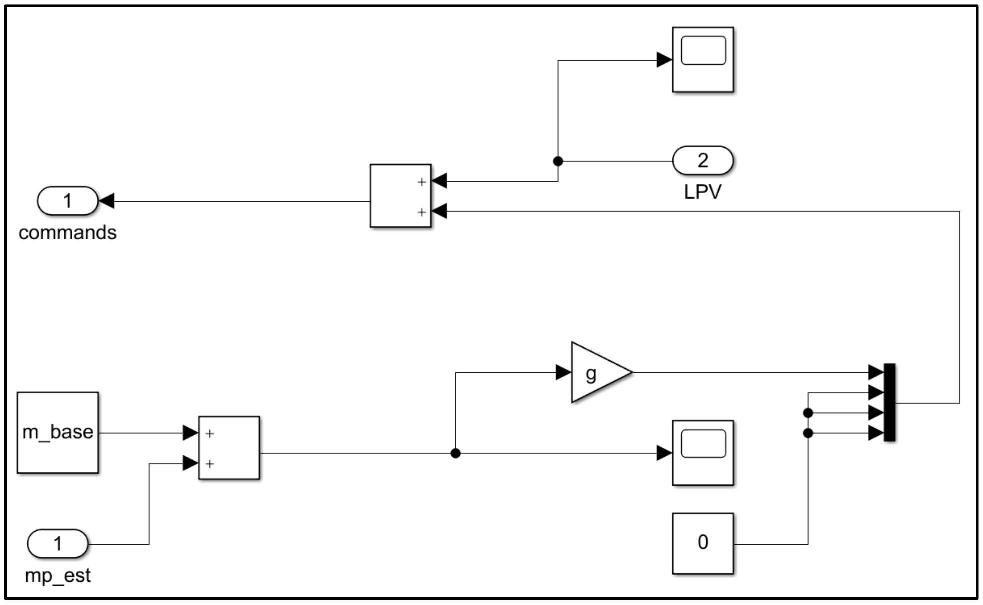


Figure B.9 – Control conditioning for LPV controller

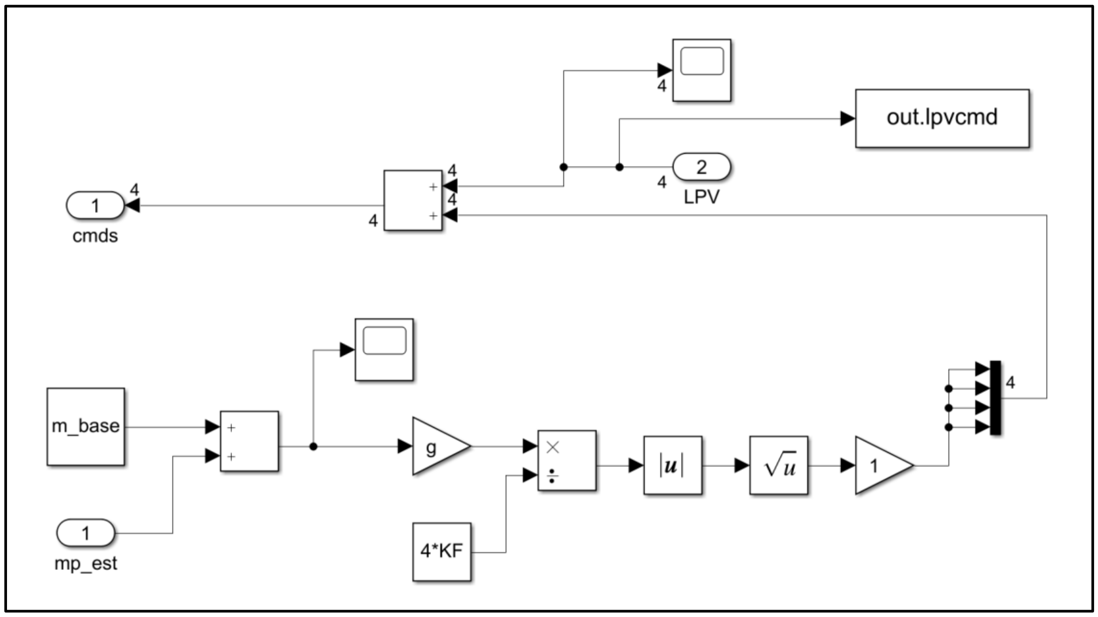


Figure B.10 – Control conditioning for propeller based LPV controller

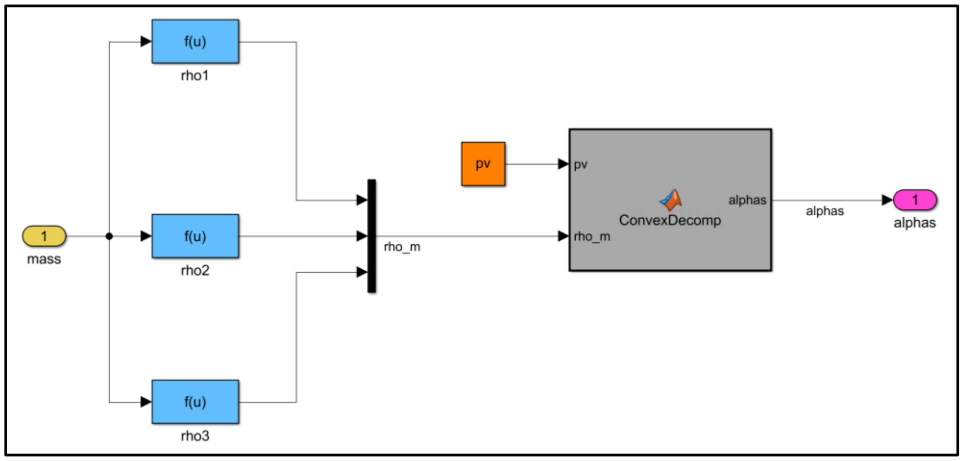


Figure B.11 – Convex decomposition function

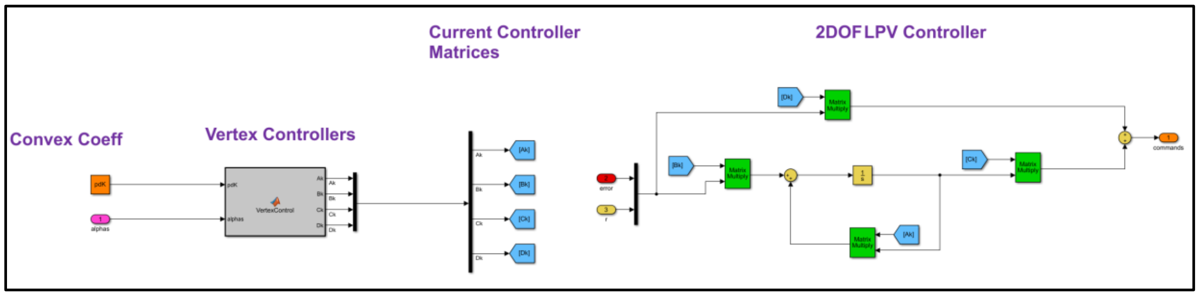


Figure B.12 – LPV controller

C. PID Control of Quadrotor

C.1 Introduction

PID control of quadrotors is a common method and its design methodology is outlined by references [10] and [26]. The methodologies described for attitude and position control in a successive loop closure structure can be used to develop the controller. In principle, the online adaptive mass estimator from Chapter 4 can be integrated into the PID control system. Here the total mass estimate $\hat{m}$ is fed directly into the PID position control equations where applicable. Since PID control relies on SISO relationships and a control strategy utilizing successive loop closure, it will be informative to compare the implementation process to the multivariable approach of LPV control.

C.2 Attitude Control

The linearized model (2.15) is used to design the attitude and position controllers. Considering only small deviations from the nominal hover state, the attitude controller tracks trajectories in 3DOF. The linear accelerations given by (C.1) are derived from the state equations of the linearized model.

$$\begin{gathered}\ddot{X} = -g(\theta\cos(\psi_0) + \varphi sin(\psi_0)) \\ \ddot{Y} = -g(\theta\sin(\psi_0) - \varphi cos(\psi_0)) \\ \ddot{Z} = \frac{U_z}{m} = \frac{U_L}{m} + g\end{gathered} \tag{C.1}$$

Derivation of (C.1)

From the linear model (2.15),

$$\begin{gathered}\dot{X} = \cos(\psi_0)u - sin(\psi_0)v \\ \dot{Y} = si\,n(\psi_0)\,u + cos(\psi_0)v \\ \dot{Z} = -\mathrm{w}\end{gathered} \quad \text{and} \quad \begin{gathered}\dot{u} = -g\theta \\ \dot{v} = g\varphi \\ \dot{w} = \frac{-U_z}{m}\end{gathered}$$

Differentiating $\dot{X}$ yields,

$$\ddot{X} = \cos(\psi_0)\dot{u} - sin(\psi_0)\dot{v}$$

Substituting $\dot{u}$ and $\dot{v}$,

$\ddot{X} = -g\theta\cos(\psi_0) - g\varphi sin(\psi_0) \rightarrow \ddot{X} = -g(\theta\cos(\psi_0) + \varphi sin(\psi_0))$

The same process is applied to the $\dot{Y}$ and $\dot{Z}$ equations. For the $\ddot{Z}$ equation, recall the equilibrium control input used for linearization is $u_e = [mg \quad 0 \quad 0 \quad 0]$. For the vertical direction near hover, $U_L = U_z - mg \rightarrow U_z = U_L + mg$. Therefore, the linear acceleration $\ddot{Z}$ can also be written as $\ddot{Z} = \frac{U_L}{m} + g$ with respect to U_z.

The angular accelerations given by are derived from the state equations for $\dot{p}, \dot{q}, \dot{r}$ and the force and torque equations (2.2). Based on the assumptions in Section 2.4 for the derivation of the equations of motion, the products of inertia are zero, and $J_x = J_y$ due to symmetry.

$$\dot{p} = \frac{U_\varphi}{J_x} = \frac{l}{J_x}(F_l - F_r)$$
$$\dot{q} = \frac{U_\theta}{J_y} = \frac{l}{J_y}\left(F_f - F_b\right) \tag{C.2}$$
$$\dot{r} = \frac{U_\psi}{J_z} = \frac{1}{J_z}(\tau_r + \tau_l - \tau_f - \tau_b)$$

The errors are defined by (C.3).

$$\begin{array}{ll} e_\varphi = \varphi_{des} - \varphi & e_p = p_{des} - p \\ e_\theta = \theta_{des} - \theta & e_q = q_{des} - q \\ e_\psi = \psi_{des} - \psi & e_r = r_{des} - r \end{array} \tag{C.3}$$

The desired torques are expressed of the form shown in (C.4), with proportional and derivative gains.

$$U_{\varphi_{des}} = K_P^\varphi e_\varphi + K_D^\varphi e_p$$
$$U_{\theta_{des}} = K_P^\theta e_\theta + K_D^\theta e_q \tag{C.4}$$
$$U_{\psi_{des}} = K_P^\psi e_\psi + K_D^\psi e_r$$

Since PID control is a SISO method, the attitude controller is developed according to the structure shown in Figure C.1, with outer-loop proportional gains and inner-loop rate gains for each controlled variable.

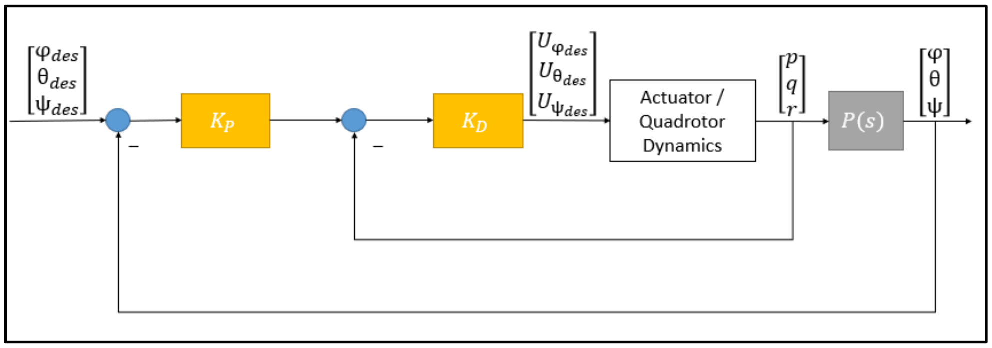

$\begin{bmatrix}\varphi_{des}\\ \theta_{des}\\ \psi_{des}\end{bmatrix}$
K_P
K_D
$\begin{bmatrix}U_{\varphi_{des}}\\ U_{\theta_{des}}\\ U_{\psi_{des}}\end{bmatrix}$
Actuator / Quadrotor Dynamics
$\begin{bmatrix}p\\ q\\ r\end{bmatrix}$
$P(s)$
$\begin{bmatrix}\varphi\\ \theta\\ \psi\end{bmatrix}$
−
−

Figure C.1 – PD control structure for attitude control

Note $[p_{des} \quad q_{des} \quad r_{des}]^T$ is taken to be $\mathbf{0}$ and each gain block consists of three gains corresponding to the errors (C.3).

C.3 Position Control

The goal of the position controller is to determine the desired roll and pitch angles, φ_{des} and θ_{des}, which are fed into the attitude controller, and to maintain a desired position at $\mathbf{r}$. The idea is to drive the position of the quadrotor using the roll and pitch angles as inputs [10]. By adjusting these angles, the position can be controlled in three dimensions. The design objective is to track a desired trajectory $z_{des}(t)$. The time varying desired position $r_T(t)$ and heading $\psi_T(t)$ are specified independently. In the hover state, the position and heading trajectories are fixed, $r_T(t) = r_0$ and $\psi_T(t) = \psi_0$, respectively.

$$z_{des}(t) = \begin{bmatrix} r_T(t) \\ \psi_T(t) \end{bmatrix} \tag{C.5}$$

The desired accelerations $\ddot{\boldsymbol{r}}_{\boldsymbol{i}}^{\boldsymbol{des}}$ are computed from a PID controller. Let e_p be the position error

$$e_p = r_T - r \tag{C.6}$$

The desired attitude for the attitude controller is then computed according to (C.7).

$$\ddot{r}_i^{des}(t) = \ddot{r}_{i,T}(t) + K_{P,i}\big(r_{i,T}(t) - r_i(t)\big) + K_{D,i}\big(\dot{r}_{i,T}(t) - \dot{r}_i(t)\big) + K_{I,i}\int \big(r_{i,T}(t) - r_i(t)\big)\,dt \tag{C.7}$$

Note at hover, $\dot{r}_T(t) = \ddot{r}_T(t) = 0$.

From the linear accelerations (C.1),

$$\begin{aligned} \ddot{r}_X^{des} &= -g(\theta_{des}\cos(\psi_T) + \varphi_{des}\sin(\psi_T)) \\ \ddot{r}_Y^{des} &= -g(\theta_{des}\sin(\psi_T) - \varphi_{des}\cos(\psi_T)) \\ \ddot{r}_Z^{des} &= \left(\frac{U_L}{m} + g\right) \end{aligned} \tag{C.8}$$

Therefore, the desired force is given by (C.9).

$$U_L \triangleq U_{z_{des}} = m\ddot{r}_Z^{des} - mg \tag{C.9}$$

Inverting the (C.8) result and replacing m with the mass estimate $\hat{m}$ produced by the estimator developed in Chapter 4 yields the desired roll and pitch angles and desired force.

$$\begin{aligned} \varphi_{des} &= \frac{-1}{g}\big(\ddot{r}_X^{des}\sin(\psi_T) - \ddot{r}_Y^{des}\cos(\psi_T)\big) \\ \theta_{des} &= \frac{-1}{g}\big(\ddot{r}_X^{des}\cos(\psi_T) + \ddot{r}_Y^{des}\sin(\psi_T)\big) \\ \psi_{des} &= \psi_T \\ U_{z_{des}} &= \hat{m}(\ddot{r}_Z^{des} - g) \end{aligned} \tag{C.10}$$

The function of the position controller within the overall control system is shown in Figure C.2.

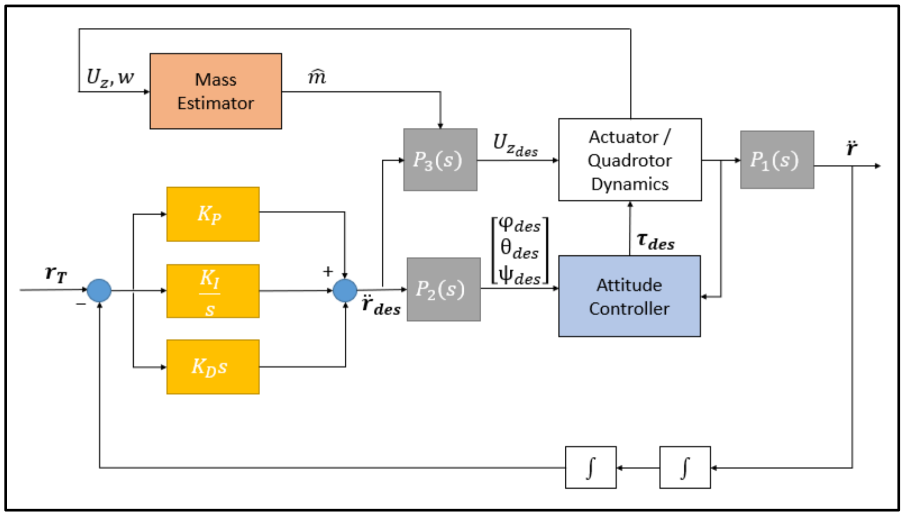


Figure C.2 – Position control integrated into the control system

C.4 Successive Loop Closure

The overall PID control system is shown in Figure C.3, including the actuator PI controller developed in Chapter 8. The same hover state conditioning system from Chapter 4 or 8 can be applied to the PID control system. The control system utilizes single loop designs for each controlled variable in a successive closure structure, a contrast to the multivariable approach of LPV control.

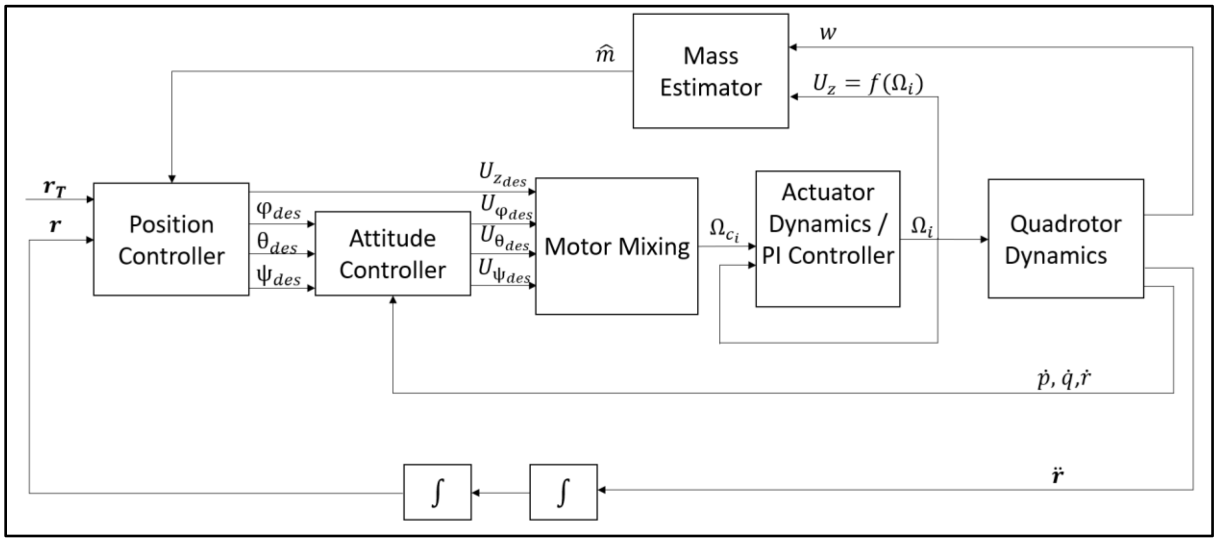


Figure C.3 – Successive loop closure with mass estimator for quadrotor control