CONTROL OF QUAD-ROTOR UAVS USING SWITCHED-SYSTEM SYNTHESIS METHODS

BY

DI FAN

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2015

Urbana, Illinois

Adviser:

Professor Geir E. Dullerud

ABSTRACT

This thesis applies switched systems synthesis and linear quadratic regulator (LQR) theory to control of a quad-rotor unmanned aerial vehicle (UAV). The thesis presents the development of the system dynamics, the theory of LQR and its implementation, the synthesis and simulation results of switched control of the UAV, which consists of a central rigid body and four propellers in a cross configuration. Since first introduced in 1917, UAVs have been extensively studied and utilized in various circumstances that prefer no human pilots aboard, due to safety, expenses, etc. Stability is crucial in controller design, while other parameters also draw great concerns, depending on the environment.

The methodologies of LQR control and semidefinite programming (SDP) are discussed to provide preliminary knowledge of the switched control. Benefits of the LQR control include tracking of reference trajectories and cost function minimization. The core of switched control methods is the design and analysis of systems whose dynamical models and performance specifications are governed by the modes of an automaton. By assigning the weights properly on the performance states, the controller allows transitions between modes with stability guaranteed. The model of the UAV was established by analyzing the equations of motions based on kinemics and dynamics, then linearized and discretized for design purposes. Both the LQR and switched controllers were generated and simulated using MATLAB, and the LQR controller was transferred to the physical UAV for test and data collection. To incorporate with reality, lags to commands and saturation of the motors were taken into consideration.

To my parents and friends, for their love and support.

ACKNOWLEDGMENTS

I want to express sincerest thanks to Professor Geir E. Dullerud, for the support and guidance during the past two years, for the inspiring course and access to the hardware resources. He also demonstrated an example of a successful researcher and offered deep insights into the control industries. Thanks to Ray Essick, who shared his profound knowledge about the switched control theory and tutored me on CVX, an SDP solver based on MATLAB. I also want to thank Jan Vervoorst for providing the Simulink model and his studies of the quad-rotor drone. Thanks to Seungho Lee and Bicheng Zhang, who helped debug codes and troubleshoot the hardware.

I would like to thank my friends, Yuqi Li and Chen Luo, for keeping me company through these years. Thanks for creating so much precious memory with me, and I know our friendship will continue despite distance and age.

Last but not least, thanks to my parents who have always been encouraging and supportive. Thanks for loving me as I am. Thanks for being by my side.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 A.R. Drone

An unmanned aerial vehicle (UAV), also known as a drone, is an aircraft without presence of human pilots aboard that can be controlled either manually or autonomously. Beyond applications in military areas, UAVs also participate in the fields of search and rescue, filmmaking and scientific research. Various sensors and cameras can be attached from which data are collected for controllers and external devices. All work presented in this thesis is based on one common class of UAVs – the quad-rotor drone. System parameters are extracted from a Parrot A.R. Drone 1.0 with PX4 control platform, as shown in Fig. 1.1.

Figure 1.1: A.R. Drone 1.0

Technical specifications that are of interest include [1]:

- carbon fiber tubes: total weight 420 g with indoor hull
- 4 brushless in-runner motors with 14.5 W and 28500 RPM

- Linux 2.6.32 system
- 3 axis gyroscope 2000 °/second precision
- 3 axis accelerometer ±50 mg precision
- HD camera with 720p resolution

The drone consists of a rigid central body with four beams to which the four motors are attached. Each motor has two rotating wings at the far extreme. Two propellers rotate clockwise and the other two counterclockwise, in order to prevent the potential horizontal movements. A quad-rotor drone has six degrees of freedoms (DOFs): roll angle, pitch angle, yaw angle, x position, y position and z position. The system states are composed of these six variables and their derivatives. The model also takes the dynamics of the wings into consideration and therefore results in a nonlinear system.

1.2 CVX

Although both the design of the LQR control and switched control was performed in the environment of MATLAB, the switched controller required a special solver for semidefinite programming – CVX. CVX is a modeling system that constructs and solves disciplined convex programs, including SDPs. It is implemented in MATLAB, which transfers MATLAB into a language for optimization models using common MATLAB functions and operations [2].

In this thesis, CVX was set in SDP mode using solver SDP3 that evaluated linear matrix inequalities (LMIs) to test stability of the system given certain weights on the performance states, then generated controllers if the system could be stabilized. Unfortunately, CVX is not designed to determine if the optimization problem is convex or not. But this is not a great concern since linearized systems are automatically convex.

1.3 Overview

Following this brief introduction chapter, this thesis continues to Chapter 2, which discusses the preliminaries and notations used in the development of

the controllers. The theory of LQR control and semidefinite programming are presented in this chapter.

Chapter 3 focuses on the switched system: problem synthesis, design model, synthesis results and illustrative examples. It includes the conditions for the existence of stabilizing controllers and derivations of the control gains.

Chapter 4 establishes the dynamic model of the quad-rotor drone based on differential equations of motion. Since both the implementation of LQR and switched control require linear system, this chapter further explains linearization around equilibrium points.

Chapter 5 provides the design procedure and simulation results of the LQR controller and switched controller, and the implementation of the LQR controller. Simulations of the switched controller are analyzed to compare with expectations. Practical factors that differ from the ideal model are also discussed here.

Chapter 6 concludes the thesis and suggests future work.

CHAPTER 2

PRELIMINARIES AND NOTATIONS

In this chapter, the statement and solution of the LQR problem are introduced. The original model is extended to incorporate with the reference signals. The theory of semidefinite programming, which is used to solve for the switched controller, is discussed in the second section.

2.1 LQR

LQR control is a method to determine the optimum solution for a minimization problem that guarantees stability of a closed-loop system. By solving the Algebraic Riccati Equation (ARE), the solution exists uniquely if the system is stabilizable and detectable. Consider the general form of a linear, time-invariant (LTI) system:

$$\begin{aligned} \dot{x}(t) = Ax(t) + Bu(t),\ x(t_0) = x_0 \\ y(t) = Cx(t) \end{aligned} \tag{2.1}$$

with $x(t) \in R^n$ and $u(t) \in R^m$.

The design goal is to find a state feedback controller $u(t) = -Kx(t)$ such that the system is stabilized, and in the mean time, minimizing the quadratic cost function that is given by

$$J = \int_{t_0}^{t_f} (x^T(t)Qx(t) + u^T(t)Ru(t))dt \tag{2.2}$$

where $Q = M^T M \succeq 0$ and $R \succeq 0$. Apparently, the design of the feedback is a tradeoff between the transient response and the control effort.

Then, given that (A, B) is stabilizable and (A, M) is detectable, the optimal stabilizing control is calculated as

$$u(t) = -R^{-1}B^T P x(t) \tag{2.3}$$

where P is the positive semidefinite solution ($P \succeq 0$) of the Algebraic Riccati Equation (ARE):

$$PA + A^T P + Q - PBR^{-1}B^T P = 0 \tag{2.4}$$

The value of K can be determined using the MATLAB command $K = lqr(A, B, Q, R)$. Since matrices A and B are fixed system dynamics, the weights on the state and input (Q and R) need to be chosen appropriately for the optimal solution to exist.

However, tacking certain signals requires more than just convergence to the origin. Additional state $x_I(t)$ is defined to accommodate the reference $r(t)$. Therefore, the original system model is generalized as [3]

$$\begin{bmatrix} \dot{x}(t) \\ \dot{x}_I(t) \end{bmatrix} = \begin{bmatrix} A & 0 \\ -C & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ x_I(t) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u(t) + \begin{bmatrix} 0 \\ I \end{bmatrix} r(t) \tag{2.5}$$

with

$$x_I(t) = \int_{t_o}^{t_f} (r(t) - Cx(t))dt = \int_{t_o}^{t_f} (r(t) - y(t))dt \tag{2.6}$$

$$\bar{x}(t) = \begin{bmatrix} x(t) \\ x_I(t) \end{bmatrix} \tag{2.7}$$

and the new cost function is given by

$$\bar{J} = \int_{t_o}^{t_f} (\bar{x}^T(t)Q\bar{x}(t) + u^T(t)Ru(t))dt \tag{2.8}$$

Thus, the controller gain becomes

$$u(t) = -\begin{bmatrix} K & K_I \end{bmatrix} \begin{bmatrix} x(t) \\ x_I(t) \end{bmatrix} = -\bar{K}\bar{x}(t) \tag{2.9}$$

Once the optimal $u(t)$ is found, the loop is closed by following the diagram in Fig. 2.1. Detailed design of the LQR controller is presented in Chapter 4.

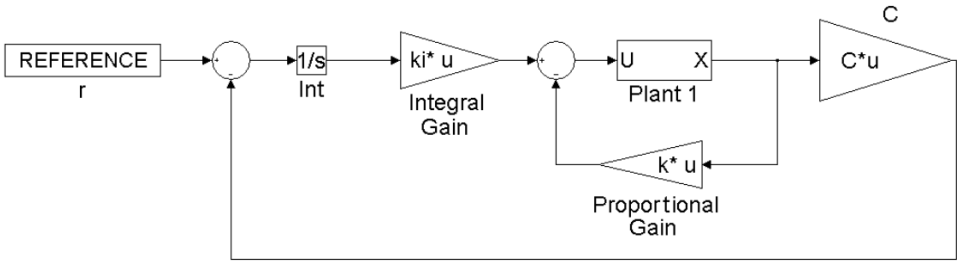

Figure 2.1: Diagram of LQR control with integral effects

2.2 SDP

Semidefinite programming (SDP) is a subfield of convex optimization that minimizes a linear function subject to the constraints of an affine combination of positive semidefinite matrices. Such constraint may be nonlinear or non-smooth, but convex. SDPs are more general than linear programming (LP), but add little computational complexity. SDP has applications in various fields, such as convex constrained optimization, control theory, and combinatorial optimization [4].

Consider minimizing a linear function of $x(t) \in R^m$:

$$\begin{aligned} &minimize \quad c^T x \\ &subject\,to \quad F(x) \geq 0 \end{aligned} \tag{2.10}$$

where

$$F(x) = F_0 + \sum_{i=1}^{m} x_i F_i \tag{2.11}$$

Define the feasible region as $\{x \mid F(x) \geq 0\}$, which consists of a boundary curve along with the region it encloses. In Fig. 2.2, it shows an example of SDP for $x \in R^2$. Roughly speaking, the SDP problem is to move as far as possible in the direction of $-c$.

Although SDP may appear quite specialized, it proves to be a generalized version of many critical optimization problems. Take the linear programming (LP) as an example:

$$\begin{aligned} &minimize \quad c^T x \\ &subject\,to \quad Ax + b \geq 0 \end{aligned} \tag{2.12}$$

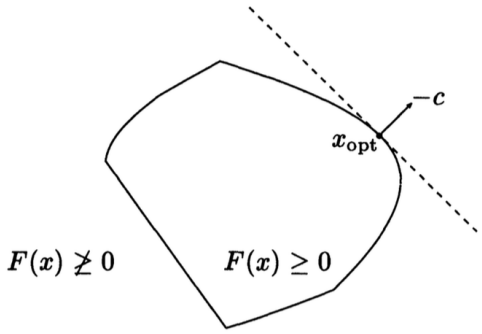

Figure 2.2: A simple SDP

A vector $v \succeq 0$ if and only if the matrix $diag(v) \succeq 0$ where $diag(v)$ is the diagonal matrix with the components of v on its diagonal; thus, we can define $F(x) = diag(Ax + b)$.

There are several types of algorithms for solving SDPs. The SDP3 solver in CVX is based on the interior point method, which is a robust and efficient approach for general linear SDPs.

CHAPTER 3

SWITCHED SYSTEMS

This chapter presents the switched control problem, model establishment and synthesis results of a discrete-time linear system with finite memory and foreknowledge. The system is subject to exponential stability with disturbance attenuation and windowed variance minimization. The resulted SDP is solved using CVX to arrive at a suitable controller. Simple examples of the switched problem are also discussed in this chapter.

3.1 Theory and Model

A switched system is defined to be a multi-model system that allows transitions among operation modes, where each mode corresponds to a distinct state-space model [5]. A possible trajectory of the modes is called an admissible sequence. We consider the system dynamics that are in the form:

$$\begin{aligned} x_{t+1} &= A_{\theta(t)}x_t + B_{\theta(t)}u_t \\ y_t &= C_{\theta(t)}x_t + D_{\theta(t)}u_t \end{aligned} \tag{3.1}$$

where $\theta(t)$ denotes the mode at which the system is operated at time t. Note that a nonlinear, continuous-time system, like the drone dynamics, needs to be linearized and discretized first in order to apply the switched control theory. An illustration of the switched system and its possible trajectories is shown in Fig. 3.1.

Let L be the length of past system parameters and H be the length of future parameters. Thus, at current time t, the controller has access to the system model from time $t - L$ to time $t + H$, represented by $\theta_{(t-L:t+H)}$. In this thesis, all stability refers to uniformly exponentially stability.

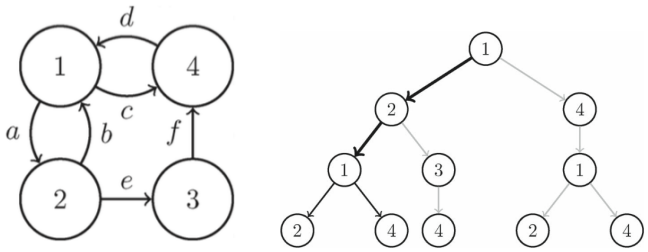

Figure 3.1: Switched system and the corresponding switching sequences

It can be proved that for a simple switched system of the form

$$x_{t+1} = A_{\theta(t)} x_t \tag{3.2}$$

with $H \geq 0$, $L \geq 0$, it is uniformly exponentially stable if and only if there exist an integer $M \geq 0$ and matrices $X_j \succ 0$ for $j \in [N]^{L+M+H}$, such that for all admissible $i_{(-L-M:H)}$ and ϕ, we have

$$A^T_{\phi(i_{(-L:H)})} X_{i_{(-L-M+1:H)}} A_{\phi(i_{(-L:H)})} - X_{i_{(-L-M:H-1)}} \prec 0 \tag{3.3}$$

where $\phi(i_{(-L:H)}) = i_0$ and $[N]$ denotes the set of indices $\{1, \ldots, N\}$.

We now apply this result to the system in Eq. 3.1 with a feedback controller given by

$$\begin{aligned} \hat{x}_{t+1} &= \hat{A}_t \hat{x}_t + \hat{B}_t y_t \\ u_t &= \hat{C}_t \hat{x}_t + \hat{D}_t y_t \end{aligned} \tag{3.4}$$

Define

$$\tilde{A}_i = \begin{bmatrix} A_i & 0 \\ 0 & 0 \end{bmatrix}; \quad \tilde{B}_i = \begin{bmatrix} 0 & B_i \\ I & 0 \end{bmatrix}; \quad \tilde{C}_i = \begin{bmatrix} 0 & I \\ C_i & 0 \end{bmatrix} \tag{3.5}$$

for $i \in [N]$ and

$$K_t = \begin{bmatrix} \hat{A}_t & \hat{B}_t \\ \hat{C}_t & \hat{D}_t \end{bmatrix} \tag{3.6}$$

Then the closed-loop system is represented by

$$x_C(t+1) = A_C(t) x_C(t) \tag{3.7}$$

where

$$x_C(t) = \begin{bmatrix} x_t \\ \hat{x}_t \end{bmatrix} \tag{3.8}$$

and

$$A_C(t) = \tilde{A}_{\theta(t)} + \tilde{B}_{\theta(t)} K_t \tilde{C}_{\theta(t)} \tag{3.9}$$

By applying Eq. 3.3 to the closed-loop system, we have

$$A_{C(i_{(-L:H)})}^T X_{i_{(-L-M+1:H)}} A_{C(i_{(-L:H)})} - X_{i_{(-L-M:H-1)}} \prec 0 \tag{3.10}$$

Now the original system is generalized by introducing the disturbance $w(t)$ and performance $z(t)$:

$$\begin{aligned} x_{t+1} &= A_{\theta(t)} x_t + B_{1,\theta(t)} w_t + B_{2,\theta(t)} u_t \\ z_t &= C_{1,\theta(t)} x_t + D_{11,\theta(t)} w_t + D_{12,\theta(t)} u_t \\ y_t &= C_{2,\theta(t)} x_t + D_{21,\theta(t)} w_t \end{aligned} \tag{3.11}$$

Note that $D_{22,\theta(t)} = 0$. Otherwise, the system cannot be uniformly exponentially stable.

The feedback controller keeps the form in Eq. 3.4. Then the system is closed using:

$$\tilde{A}_i = \begin{bmatrix} A_i & 0 \\ 0 & 0 \end{bmatrix}; \quad \tilde{B}_{1,i} = \begin{bmatrix} B_{1,i} \\ 0 \end{bmatrix}; \quad \tilde{B}_{2,i} = \begin{bmatrix} 0 & B_{2,i} \\ I & 0 \end{bmatrix} \tag{3.12}$$

$$\tilde{C}_{1,i} = \begin{bmatrix} C_{1,i} & 0 \end{bmatrix}; \quad \tilde{D}_{12,i} = \begin{bmatrix} 0 & D_{12,i} \end{bmatrix} \tag{3.13}$$

$$\tilde{C}_{2,i} = \begin{bmatrix} 0 & I \\ C_{2,i} & 0 \end{bmatrix}; \quad \tilde{D}_{21,i} = \begin{bmatrix} 0 \\ D_{21,i} \end{bmatrix} \tag{3.14}$$

and

$$\begin{aligned} A_C(i_{(-L:H)}) &= \tilde{A}_{i_0} + \tilde{B}_{2,i_0} K_{i_{(-L:H)}} \tilde{C}_{2,i_0} \\ B_C(i_{(-L:H)}) &= \tilde{B}_{1,i_0} + \tilde{B}_{2,i_0} K_{i_{(-L:H)}} \tilde{D}_{21,i_0} \\ C_C(i_{(-L:H)}) &= \tilde{C}_{1,i_0} + \tilde{D}_{12,i_0} K_{i_{(-L:H)}} \tilde{C}_{2,i_0} \\ D_C(i_{(-L:H)}) &= D_{11,i_0} + \tilde{D}_{12,i_0} K_{i_{(-L:H)}} \tilde{D}_{21,i_0} \end{aligned} \tag{3.15}$$

which gives

$$\begin{aligned} x_C(t+1) &= A_C(\theta_{(t-L:t+H)})x_C(t) + B_C(\theta_{(t-L:t+H)})w(t) \\ z(t) &= C_C(\theta_{(t-L:t+H)})x_C(t) + D_C(\theta_{(t-L:t+H)})w(t) \end{aligned} \tag{3.16}$$

Consider the T-step uniform performance level γ for a finite forward window $T \geq 0$: for some $\gamma > 0$, $x(0) = 0$ and $t \geq 0$, we have

$$\frac{1}{T+1} \sum_{s=t}^{t+T} \parallel z(s) \parallel^2 < \gamma^2 \tag{3.17}$$

For systems with stochastic disturbance, instead of averaging $\parallel z(s) \parallel^2$, we need to average $E[\parallel z(s) \parallel^2]$.

Analogous to Eq. 3.3, for $H \geq 0$, $L \geq 0$, the system is uniformly exponentially stable and satisfies the T-step uniform performance level γ if and only if there exists an integer $M \geq 0$ and matrices $Y_j \succ 0$ for $j \in [N]^{L+M+H}$ such that for all admissible $i_{(-L-M:H)}$ and $\hat{\imath}_{(-L-M:H+T)}$

$$A_{\phi(i_{(-L:H)})} Y_{i_{(-L-M:H-1)}} A^T_{\phi(i_{(-L:H)})} - Y_{i_{(-L-M+1:H)}} \prec -B_{\phi(i_{(-L:H)})} B^T_{\phi(i_{(-L:H)})} \tag{3.18}$$

$$\begin{aligned} \frac{1}{T+1} \sum_{t=0}^{T} Tr(& C_{\phi(\hat{\imath}_{(t-L:t+H)})} Y_{\hat{\imath}_{(t-L-M:t+H-1)}} C^T_{\phi(\hat{\imath}_{(t-L:t+H)})} \\ & + D_{\phi(\hat{\imath}_{(t-L:t+H)})} D^T_{\phi(\hat{\imath}_{(t-L:t+H)})}) < \gamma^2 \end{aligned} \tag{3.19}$$

Equivalently,

$$A_C(i_{(-L:H)}) Y_{i_{(-L-M:H-1)}} A^T_C(i_{(-L:H)}) - Y_{i_{(-L-M+1:H)}} \prec -B_C(i_{(-L:H)}) B^T_C(i_{(-L:H)}) \tag{3.20}$$

$$\begin{aligned} \frac{1}{T+1} \sum_{t=0}^{T} Tr(& C_C(\hat{\imath}_{(t-L:t+H)}) Y_{\hat{\imath}_{(t-L-M:t+H-1)}} C^T_C(\hat{\imath}_{(t-L:t+H)}) \\ & + D_C(\hat{\imath}_{(t-L:t+H)}) D^T_C(\hat{\imath}_{(t-L:t+H)})) < \gamma^2 \end{aligned} \tag{3.21}$$

for the closed-loop system in Eq. 3.16.

3.2 Synthesis Results

To further evaluate the stability and performance level conditions, we take advantage of the Schur complement [6]. Consider a portioned matrix given by

$$X = \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \tag{3.22}$$

Then the followings are equivalent:

- $X \succ 0$
- $X_{22} \succ 0 \, and \, X_{11} - X_{12}X_{22}^{-1}X_{21} \succ 0$
- $X_{11} \succ 0 \, and \, X_{22} - X_{21}X_{11}^{-1}X_{12} \succ 0$

Applying the Schur complement to Eq. 3.1 and Eq. 3.21, we have [7]

$$\begin{bmatrix} -Y^{-1}_{i_{(-L-M:H-1)}} & A_C^T(i_{(-L:H)}) & 0 \\ A_C(i_{(-L:H)}) & -Y_{i_{(-L-M+1:H)}} & B_C(i_{(-L:H)}) \\ 0 & B_C^T(i_{(-L:H)}) & -I \end{bmatrix} \prec 0 \tag{3.23}$$

$$\begin{bmatrix} -Y^{-1}_{\hat{\imath}_{(-L-M:H-1)}} & C_C^T(\hat{\imath}_{(L:H)}) & 0 \\ C_C(\hat{\imath}_{(L:H)}) & -Z_{\hat{\imath}_{(-L-M:H-1)}} & D_C(\hat{\imath}_{(L:H)}) \\ 0 & D_C^T(\hat{\imath}_{(L:H)}) & -I \end{bmatrix} \prec 0 \tag{3.24}$$

$$\frac{1}{T+1}\sum_{t=0}^{T} Tr(Z_{\hat{\imath}_{(t-L-M:t+H-1)}}) < \gamma^2 \tag{3.25}$$

where Y_j has the form

$$Y_j = \begin{bmatrix} R_j & T_j \\ T_j^T & \cdot \end{bmatrix}; \quad Y_j^{-1} = \begin{bmatrix} S_j & U_j \\ U_j^T & \cdot \end{bmatrix} \tag{3.26}$$

and it can be proved that

$$U_j = (S_j - R_j^{-1})^{\frac{1}{2}} \tag{3.27}$$

$$T_j = -R_j U_j \tag{3.28}$$

Define

$$W_i = \begin{bmatrix} S_{i_{(-L-M+1:H)}} A_{j_0} R_{i_{(-L-M:H-1)}} & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} U_{i_{(-L-M+1:H)}} & S_{i_{(-L-M+1:H)}} B_{2,i_0} \\ 0 & I \end{bmatrix} K_{i_{(-L:H)}} \begin{bmatrix} T^T_{i_{(-L-M:H-1)}} & 0 \\ C_{2,i_0} R_{i_{(-L-M+1:H)}} & I \end{bmatrix} \tag{3.29}$$

for all $i_{(-L-M:H)}$. Based on the value for W_i's, a stabilizing controller $K_{i_{(-L:H)}}$ can be determined for each switching sequence.

Finally, we arrived at the solution to the switched control synthesis: there exists a path-dependent controller with horizon $H \geq 0$ such that the system in Eq. 3.11 is uniformly exponentially stable and satisfies the T-step uniform performance level γ if and only if there exists an integer $\bar{L} \geq 0$, matrices $R_j \succ 0$, $S_j \succ 0$ for $j \in [N]^{\bar{L}+H}$, and matrices Z_i, W_i for $i \in [N]^{\bar{L}+H+1}$ such that for all admissible $i_{(-\bar{L}:H)}$ and $\hat{\imath}_{(-\bar{L}:H+T)}$ [8]

$$H_i + F_{i_0}^T W_i G_{i_0} + G_{i_0}^T W_i^T F_{i_0} \prec 0 \tag{3.30}$$

$$\hat{H}_i + \hat{F}_{i_0}^T W_i \hat{G}_{i_0} + \hat{G}_{i_0}^T W_i^T \hat{F}_{i_0} \prec 0 \tag{3.31}$$

$$\frac{1}{T+1} \sum_{t=0}^{T} Tr(Z_{\hat{\imath}_{(t-\bar{L}:t+H)}}) < \gamma^2 \tag{3.32}$$

with $i_- = i_{(-\bar{L}:H-1)}$, $i_+ = i_{(-\bar{L}+1:H)}$ and

$$G_{i_0} = \begin{bmatrix} 0 & I & 0 & 0 & 0 \\ C_{2,i_0} & 0 & 0 & 0 & D_{21,i_0} \end{bmatrix}; \quad \hat{G}_{i_0} = \begin{bmatrix} 0 & I & 0 & 0 \\ C_{2,i_0} & 0 & 0 & D_{21,i_0} \end{bmatrix} \tag{3.33}$$

$$H_i = \begin{bmatrix} -S_{i_-} & -I & A_{i_0}^T & A_{i_0}^T S_{i_+} & 0 \\ -I & -R_{i_-} & R_{i_-} A_{i_0}^T & 0 & 0 \\ A_{i_0} & A_{i_0} R_{i_-} & -R_{i_+} & -I & B_{1,i_0} \\ S_{i_+} A_{i_0} & 0 & -I & -S_{i_+} & S_{i_+} B_{1,i_0} \\ 0 & 0 & B_{1,i_0}^T & B_{1,i_0}^T S_{i_+} & -I \end{bmatrix} \tag{3.34}$$

$$\hat{H}_i = \begin{bmatrix} -S_{i_-} & -I & C_{1,i_0}^T & 0 \\ -I & -R_{i_-} & R_{i_-}C_{1,i_0}^T & 0 \\ C_{1,i_0} & C_{1,i_0}R_{i_-} & -Z_i & D_{11,i_0} \\ 0 & 0 & D_{11,i_0}^T & -I \end{bmatrix} \tag{3.35}$$

Here $i_{(-\bar{L}:H)}$ refers to the modes that the controller have access to and $\hat{\imath}_{(-\bar{L}:H+T)}$ is the path on which performance is evaluated. Given all conditions are satisfied, a controller can be constructed with $L \leq \bar{L}$. For simplicity, the controller in this thesis is designed with $L = \bar{L}$.

3.3 Examples

Two examples are given in this section to demonstrate how to establish the switched control model, and each corresponding controller is calculated based on the performance level, length of memory and horizon.

Consider the following system [9]:

$$Mode\,1 : \begin{cases} x(t+1) = 0.3x(t) \\ z(t) = x(t) + u(t) \\ y(t) = x(t) \end{cases} \tag{3.36}$$

$$Mode\,2 : \begin{cases} x(t+1) = 3x(t) + 0.5w(t) + u(t) \\ z(t) = x(t) + u(t) \\ y(t) = x(t) \end{cases} \tag{3.37}$$

Assume the transition diagram is complete, which means all modes are connected and the system can continuously stay in one mode. The controller is designed with knowledge of the past one mode ($M = 1$), zero horizon ($H = 0$), zero performance window ($T = 0$) and a performance level (γ) of 0.8 and 0.9. We can see that in this model, the system dynamics vary between modes while the performance states remain the same.

The result showed that a performance level of 0.8 is too small for a stabilizing controller to exist, which is reasonable since γ characterizes how bounded the performance $z(t)$ is with respect to the disturbance $w(t)$.

However, after increasing the limit to 0.9, the system became stabilizable and the four controllers are given by

$$K_{11}: \begin{cases} \hat{x}_{t+1} = 0.0059\hat{x}_t - 0.4076y_t \\ u_t = -0.0001\hat{x}_t - 1.0002y_t \end{cases} \tag{3.38}$$

$$K_{12}: \begin{cases} \hat{x}_{t+1} = 0.0107\hat{x}_t - 1.3794y_t \\ u_t = -0.0028\hat{x}_t - 2.6344y_t \end{cases} \tag{3.39}$$

$$K_{21}: \begin{cases} \hat{x}_{t+1} = -0.0021\hat{x}_t - 0.4043y_t \\ u_t = -0.0000\hat{x}_t - 0.9999y_t \end{cases} \tag{3.40}$$

$$K_{22}: \begin{cases} \hat{x}_{t+1} = 0.0000\hat{x}_t - 1.8971y_t \\ u_t = -0.0000\hat{x}_t - 2.4972y_t \end{cases} \tag{3.41}$$

where K_{ij} represents the controller corresponding to a transition from the past mode i to the current mode j.

Consider a continuous-time system that is a simplified model for a small spacecraft. Based on the location of the aircraft in the environment, three modes are defined: the unobstructed mode, obstacle-in-x-direction mode and obstacle-in-y-direction mode, which are characterized by

$$\begin{cases} \dot{x} = v_x \\ \dot{v}_x = -0.5v_x + u_x + 0.1u_y \\ \dot{y} = v_y \\ \dot{v}_y = -0.5v_y + 0.1u_x + u_y \end{cases} \tag{3.42}$$

where

$$z_1 = \begin{bmatrix} x & y & 0.5u_x & 0.5u_y \end{bmatrix}^T \tag{3.43}$$

$$z_2 = \begin{bmatrix} 5x & 0.5y & 0.5u_x & 0.5u_y \end{bmatrix}^T \tag{3.44}$$

$$z_3 = \begin{bmatrix} 0.5x & 5y & 0.5u_x & 0.5u_y \end{bmatrix}^T \tag{3.45}$$

In this model, universal system dynamics are constructed for all modes, while the performance is evaluated inconsistently in each case. The state x is penalized much more heavily than state y in mode 2, but much less in mode 3.

In order to apply the switched control, the system was first discretized with time interval 0.1 s. Then the controllers were designed with $L = H = T = 1$ and $\gamma = 1$. This setup resulted in 17 different paths, 2 of which are shown in Fig. 3.2 and Fig. 3.3.

```
A_111 =

   1.0e-03 *

    0.0014    0.0037    0.0004   -0.0001
    0.1036    0.0879    0.0082   -0.0026
    0.0004   -0.0001    0.0014    0.0037
    0.0082   -0.0026    0.1036    0.0879


B_111 =

   1.0e+04 *

   -1.6391   -0.1372    0.0048    0.0011
    1.0733   -1.0560    0.0954    0.0225
    0.0048    0.0011   -1.6391   -0.1372
    0.0954    0.0225    1.0733   -1.0560
C_111 =

   1.0e-07 *

   -0.6312   -0.5411    0.0136    0.0697
    0.0136    0.0697   -0.6312   -0.5411


D_111 =

   -6.5062   -3.3782    0.0728    0.2013
    0.0728    0.2013   -6.5062   -3.3782
```

Figure 3.2: Switched controller for path 111

```
A_131 =

   1.0e-04 *

   -0.1013   -0.0421    0.0094    0.0069
   -0.6086   -0.3398    0.0909    0.0666
   -0.0014   -0.0014   -0.0076    0.0037
   -0.0224   -0.0216    0.4241    0.1771


B_131 =

   1.0e+04 *

   -1.8223   -0.2393   -0.0043    0.0065
    0.6359   -1.4138    0.0001    0.0451
    0.0045    0.0004   -1.7356   -0.1645
    0.1367    0.0220    0.1283   -1.2634
C_131 =

   1.0e-07 *

    0.3577    0.1989   -0.0286   -0.0288
   -0.0218   -0.0066   -0.2517   -0.1047


D_131 =

   -4.1978   -1.4601    0.0839   -0.0396
   -0.3857    0.0255   -0.8838   -2.1795
```

Figure 3.3: Switched controller for path 131

CHAPTER 4

DYNAMICS AND MODELING

The derivation of the system dynamics of the quad-rotor drone, using the body and earth reference systems, is presented in this chapter. Due to the property of the LQR control and switched control, the model needs to be linearized with respect to the equilibrium positions.

4.1 Drone Dynamics

Several assumptions are required in order to construct the model: the inertia matrix should be time-invariant. The origin and axes of the body frame coincide with the center of mass and principal axes of inertia, respectively, so that the inertia matrix is diagonal.

As shown in Fig. 4.1, the A.R. drone used for this thesis has an 'X' configuration, which means the axes of body frame lie between motors. Motor 1 and 3 rotate clockwise while 2 and 4 counterclockwise to balance the movements in the xy plane.

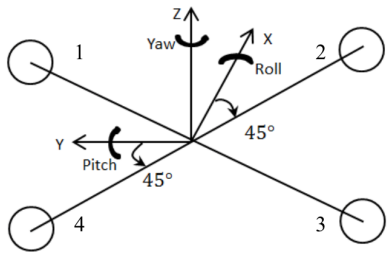

Figure 4.1: Reference frame of quad-rotor drone

For a rigid body with 6 degrees of freedom (DOFs), it is common to use two reference systems to describe its motion: body frame and earth frame. The kinemics of a generic 6 DOF system are determined by [10]

$$\dot{\xi} = J_\theta \nu \tag{4.1}$$

ξ consists of the linear and angular position vectors in the earth frame, where the angles stand for roll, pitch, and yaw, respectively.

$$\xi = \begin{bmatrix} \Gamma^E \\ \Theta^E \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{bmatrix} \tag{4.2}$$

ν is composed of the corresponding linear and angular velocity vectors in the body frame.

$$\nu = \begin{bmatrix} v^B \\ \omega^B \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \\ p \\ q \\ r \end{bmatrix} \tag{4.3}$$

Due to the inconsistency in the reference frames, a generalized matrix J_θ is introduced, which is given by

$$J_\theta = \begin{bmatrix} & R_\theta & & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & & T_\theta & \end{bmatrix} \tag{4.4}$$

with rotational matrix

$$R_\theta =$$

$$\begin{bmatrix} \cos\psi\cos\theta & \cos\psi\sin\theta\sin\psi - \sin\psi\cos\phi & \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ \sin\psi\cos\theta & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix} \tag{4.5}$$

and translational matrix

$$T_\theta = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \tag{4.6}$$

Therefore, the linear velocities in the earth frame and body frame are related by

$$v^E = \dot{\Gamma}^E = R_\theta v^B \tag{4.7}$$

Similarly, we have

$$\dot{\Theta}^E = T_\theta \omega^B \tag{4.8}$$

Next, the dynamics of the drone is studied. From Newton's second law:

$$F^E = m\ddot{\Gamma}^E = m(R_\theta \dot{v}^B) \tag{4.9}$$

$$\Rightarrow R_\theta F^B = m(R_\theta \dot{v}^B + \dot{R}_\theta v^B) = mR_\theta(\dot{v}^B + \omega^B \times v^B) \tag{4.10}$$

$$\Rightarrow F^B = m(\dot{v}^B + \omega^B \times v^B) = m\dot{v}^B + \omega^B \times (mv^B) \tag{4.11}$$

where F^B represents the force vector in body frame.

For the angular components of body motion:

$$\tau^E = I\ddot{\Theta}^E = I(T_\theta \dot{\omega}^B) \tag{4.12}$$

$$\Rightarrow T_\theta \tau^B = I(T_\theta \dot{\omega}^B + \dot{T}_\theta \omega^B) = IT_\theta(\dot{\omega}^B + \omega^B \times \omega^B) \tag{4.13}$$

$$\Rightarrow \tau^B = I\dot{\omega}^B + \omega^B \times (I\omega^B) \tag{4.14}$$

where τ^B stands for the torque vector in body frame.

Combining Eq. 4.11 and Eq. 4.14 leads to

$$\Lambda = \begin{bmatrix} F_x \\ F_y \\ F_z \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{xx} & 0 & 0 \\ 0 & 0 & 0 & 0 & I_{yy} & 0 \\ 0 & 0 & 0 & 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix}$$
$$+ \begin{bmatrix} 0 & 0 & 0 & 0 & mw & -mv \\ 0 & 0 & 0 & -mw & 0 & mu \\ 0 & 0 & 0 & mv & -mu & 0 \\ 0 & 0 & 0 & 0 & I_{zz}r & -I_{yy}q \\ 0 & 0 & 0 & -I_{zz}r & 0 & I_{xx}p \\ 0 & 0 & 0 & I_{yy}q & -I_{xx}p & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ p \\ q \\ r \end{bmatrix} \tag{4.15}$$

where

$$\Lambda = G_B(\xi) + O_B(\nu)\overrightarrow{\Omega} + E_B\overrightarrow{\Omega}^2 \tag{4.16}$$

The generalized force vector Λ is decomposed into three parts: the gravitational vector G_B, gyroscopic propeller matrix O_B and moment matrix E_B [11].

The gravitational vector takes into account the acceleration due to gravity so it only affects the linear equations.

$$G_B(\xi) = \begin{bmatrix} -mg\sin\theta \\ mg\cos\theta\sin\phi \\ mg\cos\theta\cos\phi \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.17}$$

The gyroscopic propeller matrix is responsible for the gyroscopic effects caused by propeller rotation. When the drone is in a perfect hovering state, the matrix should be zero. But for nonzero roll or pitch rates, the quad-rotor drone experiences a gyroscopic torque.

$$O_B(\nu)\overrightarrow{\Omega} = J_{TP} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ q & -q & q & -q \\ -p & p & -p & p \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \\ \Omega_4 \end{bmatrix} \tag{4.18}$$

where J_{TP} represents the overall motor rotational moment of inertia, and

$$\overrightarrow{\Omega} = \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \\ \Omega_4 \end{bmatrix} \tag{4.19}$$

is the propeller speed vector in rad/s. And denote the overall propeller speed by

$$\Omega = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4 \tag{4.20}$$

Finally the moment matrix considers the forces and torques produced directly by the movement inputs throttle, roll, pitch and yaw. From aerodynamics, the forces and moments are proportional to the squared propeller speed.

$$E_B \overrightarrow{\Omega}^2 = \begin{bmatrix} 0 \\ 0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ \frac{1}{\sqrt{2}} bl(\Omega_1^2 - \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \\ \frac{1}{\sqrt{2}} bl(\Omega_1^2 + \Omega_2^2 - \Omega_3^2 - \Omega_4^2) \\ d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \end{bmatrix} \tag{4.21}$$

with aerodynamic drag d in Nms^2, aerodynamic thrust b in Ns^2, and l being the distance between the center of the drone and the center of the propeller. And the u's denote the throttle, roll, pitch and yaw inputs, respectively.

Now the complete quad-rotor drone model can be written as

$$\begin{cases} \dot{u} = (vr - wq) - g\sin\theta \\ \dot{v} = (wp - ru) + g\cos\theta\sin\phi \\ \dot{w} = (uq - vp) + g\cos\theta\cos\phi + \frac{u_1}{m} \\ \dot{p} = \frac{I_{yy}-I_{zz}}{I_{xx}}qr - \frac{J_{TP}}{I_{xx}}q\Omega + \frac{u_2}{I_{xx}} \\ \dot{q} = \frac{I_{zz}-I_{xx}}{I_{yy}}pr + \frac{J_{TP}}{I_{yy}}p\Omega + \frac{u_3}{I_{yy}} \\ \dot{r} = \frac{I_{xx}-I_{yy}}{I_{zz}}pq + \frac{u_4}{I_{zz}} \end{cases} \tag{4.22}$$

Alternatively, the drone model can be constructed using a hybrid system of linear states in the earth frame and angular states in the body frame. Thus, the dynamics are characterized by

$$\begin{cases} \ddot{x} = (\sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi)\frac{u_1}{m} \\ \ddot{y} = (-\cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi)\frac{u_1}{m} \\ \ddot{z} = g + \cos\theta\cos\phi\frac{u_1}{m} \\ \dot{p} = \frac{I_{yy}-I_{zz}}{I_{xx}}qr - \frac{J_{TP}}{I_{xx}}q\Omega + \frac{u_2}{I_{xx}} \\ \dot{q} = \frac{I_{zz}-I_{xx}}{I_{yy}}pr + \frac{J_{TP}}{I_{yy}}p\Omega + \frac{u_3}{I_{yy}} \\ \dot{r} = \frac{I_{xx}-I_{yy}}{I_{zz}}pq + \frac{u_4}{I_{zz}} \end{cases} \tag{4.23}$$

with

$$\begin{cases} u_1 = -b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ u_2 = \frac{1}{\sqrt{2}}bl(\Omega_1^2 - \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \\ u_3 = \frac{1}{\sqrt{2}}bl(\Omega_1^2 + \Omega_2^2 - \Omega_3^2 - \Omega_4^2) \\ u_4 = d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \end{cases} \tag{4.24}$$

Then the control inputs are distributed to the motors following the relation given by

$$\begin{gathered} front\ left\ motor : u_1 + \frac{1}{\sqrt{2}}(u_2 + u_3) - u_4 \\ front\ right\ motor : u_1 + \frac{1}{\sqrt{2}}(-u_2 + u_3) + u_4 \\ rear\ right\ motor : u_1 - \frac{1}{\sqrt{2}}(u_2 + u_3) - u_4 \\ rear\ left\ motor : u_1 + \frac{1}{\sqrt{2}}(u_2 - u_3) + u_4 \end{gathered} \tag{4.25}$$

From control theory, we know that a system with four independent inputs can fully control no more than four independent states. In the design of the LQR and switched controller, the x position, y position, z position and yaw

angle are selected to follow the reference trajectory.

4.2 Linearization

Based on the model established in the previous section, it is obvious that linearity does not exist in the drone dynamics. To convert the system into the following form:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{4.26}$$

the original system needs to be linearized with respect to the equilibrium points.

Define the system state as

$$x = [\ p \quad q \quad r \quad \dot{x} \quad \dot{y} \quad \dot{z} \quad \phi \quad \theta \quad \psi \quad x \quad y \quad z\ ]^T \tag{4.27}$$

By setting all velocities and accelerations to zero, the equilibrium point is given by

$$x_e = [\ 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad \psi_{ref} \quad x_{ref} \quad y_{ref} \quad z_{ref}\ ]^T \tag{4.28}$$

and

$$u_e = \begin{bmatrix} -mg & 0 & 0 & 0 \end{bmatrix}^T \tag{4.29}$$

which means the drone can reach equilibrium at any position with arbitrary yaw angle ψ_{ref}.

According to the theory of nonlinear system, given that

$$x = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}^T \tag{4.30}$$

and

$$\dot{x}_i = f_i(x) \tag{4.31}$$

where f is a nonlinear function in x, then the linearized matrix is determined by [12]

$$A_{ij} = \frac{\partial f_i}{\partial x_j} \mid_{x=x_e} \tag{4.32}$$

where A_{ij} denotes the ij-th entry in matrix A, such that

$$\dot{x} \approx Ax \tag{4.33}$$

in a small neighborhood of the equilibrium point.

Following this method, the state matrices in Eq. 4.26 are written as

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -g\cos\psi_{ref} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & g\cos\psi_{ref} & -g\sin\psi_{ref} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.34}$$

$$B = \begin{bmatrix} 0 & \frac{1}{I_{xx}} & 0 & 0 \\ 0 & 0 & \frac{1}{I_{yy}} & 0 \\ 0 & 0 & 0 & \frac{1}{I_{zz}} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.35}$$

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.36}$$

and D is a matrix of zeros with 12 rows and 4 columns.

With this linear state-space model, we are ready to generate the LQR controller and switched controller. Since the linearized dynamics are valid approximations only at a bounded area around the initial point, the transit response of the original nonlinear model is expected to differ from the linear one. But in the long term, they should converge to the same reference trajectory.

CHAPTER 5

CONTROLLER DESIGN AND SIMULATION

In this chapter, the detailed design procedure of the LQR controller and switched controller is provided. For LQR, the controller was simulated on both the linear and nonlinear systems. Furthermore, the saturation of motors and lag of commands were taken into account to better construct the physical model. We also included an observer to estimate the immeasurable states. Experimental data was collected to compare with the simulation results and we analyzed the problems that occurred during implementation. For switched control, all admissible paths were generated as well as their corresponding controllers, based on the method discussed in Chapter 3. Simulation results displayed how the performance varied on each path, which perfectly matched with expectations.

5.1 LQR Controller

As presented in Chapter 2, the design of a LQR controller depends on the system dynamics and proper weights (Q and R) on the states and control inputs. For better tracking of the reference signals, the integral states ($x_I(t)$) were penalized most heavily, since they account for the difference between current states and desired trajectories. We also applied large weights to the control inputs, with an aim to avoid motor saturation. But a tradeoff was made due to the fact that the system may not be stabilizable for large R. Since we are most interested in the drone position, all derivative states (linear velocities and angular velocities) were designed to have little contribution in the cost function. Based on this methodology, the weighting matrices are given by

$$R = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix} \tag{5.1}$$

and

$$Q_{tot} = \begin{bmatrix} Q & 0 \\ 0 & Q_I \end{bmatrix} \tag{5.2}$$

where

$$Q = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 20 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{bmatrix} \tag{5.3}$$

$$Q_I = \begin{bmatrix} 100 & 0 & 0 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \end{bmatrix} \tag{5.4}$$

which result in a stabilizing controller in the form:

$$K = \begin{bmatrix} -0.0000 & 0.0000 & -0.0000 & -0.0000 & -0.0000 & 3.5350 \\ 1.8401 & -0.0000 & 0.0000 & 0.0000 & 4.5766 & -0.0000 \\ -0.0000 & 1.8368 & -0.0000 & -4.5747 & -0.0000 & 0.0000 \\ 0.0000 & -0.0000 & 1.8349 & 0.0000 & -0.0000 & -0.0000 \end{bmatrix}$$

$$\begin{bmatrix} -0.0000 & -0.0000 & -0.0000 & -0.0000 & 0.0000 & 6.5181 \\ 12.9052 & -0.0000 & 0.0000 & 0.0000 & 7.3833 & 0.0000 \\ -0.0000 & 12.8936 & -0.0000 & -7.3818 & 0.0000 & -0.0000 \\ -0.0000 & -0.0000 & 4.7806 & -0.0000 & 0.0000 & 0.0000 \end{bmatrix} \tag{5.5}$$

and

$$K_I = \begin{bmatrix} 0.0000 & -0.0000 & -0.0000 & 0.0000 & 0.0000 & 5.7735 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 5.7735 & -0.0000 \\ 0.0000 & -0.0000 & -0.0000 & 5.7735 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 5.7735 & -0.0000 & -0.0000 & 0.0000 \end{bmatrix} \tag{5.6}$$

Now the feedback loop has been completed, but not all states are accessible since there are no sensors on the drone directly measuring the linear velocities in the x, y and z directions. Thus, an observer is required to estimate the immeasurable states based on the output y. An estimator $\hat{x}(t)$ is introduced which follows the same dynamics of the original system, and also includes the difference between the estimated output and the measured output [13] [14].

$$\begin{aligned} \dot{\hat{x}}(t) &= A\hat{x}(t) + Bu(t) + L(y(t) - \hat{y}(t)) \\ \hat{y}(t) &= C\hat{x}(t) + Du(t) \end{aligned} \tag{5.7}$$

Define the error state as

$$e(t) = \hat{x}(t) - x(t) \tag{5.8}$$

$$\Rightarrow \dot{e}(t) = (A - LC)e(t) \tag{5.9}$$

If the system in Eq. 5.9 is stabilizable, the error $e(t)$ will converge to zero exponentially fast, so that the estimator provides an accurate approximation for the actual state. Then the observer gain L was determined by placing the poles about five times larger than the poles of the original system, which all lie in the left half of the complex plane.

$$L = \begin{bmatrix} 1.2210 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.0920 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.1227 & 0 & 0 & 0 \\ 0 & -0.0010 & 0 & 1.1016 & 0 & 0 \\ 0.0010 & 0 & 0 & 0 & 1.0807 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.0600 \\ 0.0221 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.0209 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0212 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0210 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0208 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0206 \end{bmatrix} \times 10^4 \quad (5.10)$$

After acquiring all gains, the controller and observer were simulated on both the linear system and original nonlinear system, as shown in Fig. 5.1 and Fig. 5.2. The initial values for all states are zero, and some system parameters of the drone are listed below:

$$\begin{cases} m = 0.4472\, kg \\ I_{xx} = 0.0020\, Nms^2 \\ I_{yy} = 0.0016\, Nms^2 \\ I_{zz} = 0.0035\, Nms^2 \\ l = 0.1778\, m \\ d = 1 \times 10^{-7}\, Nms^2 \\ b = 192.32 \times 10^{-7}\, Ns^2 \end{cases} \quad (5.11)$$

Saturation blocks were included in the linear model and the motors in the nonlinear model. The reference signals fed into the system are given by

$$r(t) = \begin{bmatrix} 0 & 0 & \frac{\pi}{6} & 0.2 & 0.5 & 0.8 \end{bmatrix} \quad (5.12)$$

equivalently, point $(0.2, 0.5, 0.8)$ in the xyz space with a yaw angle of $30\,^\circ$.

In a simulation of 10 s, the positions and angles of the linear and nonlinear systems are presented in Fig. 5.3.

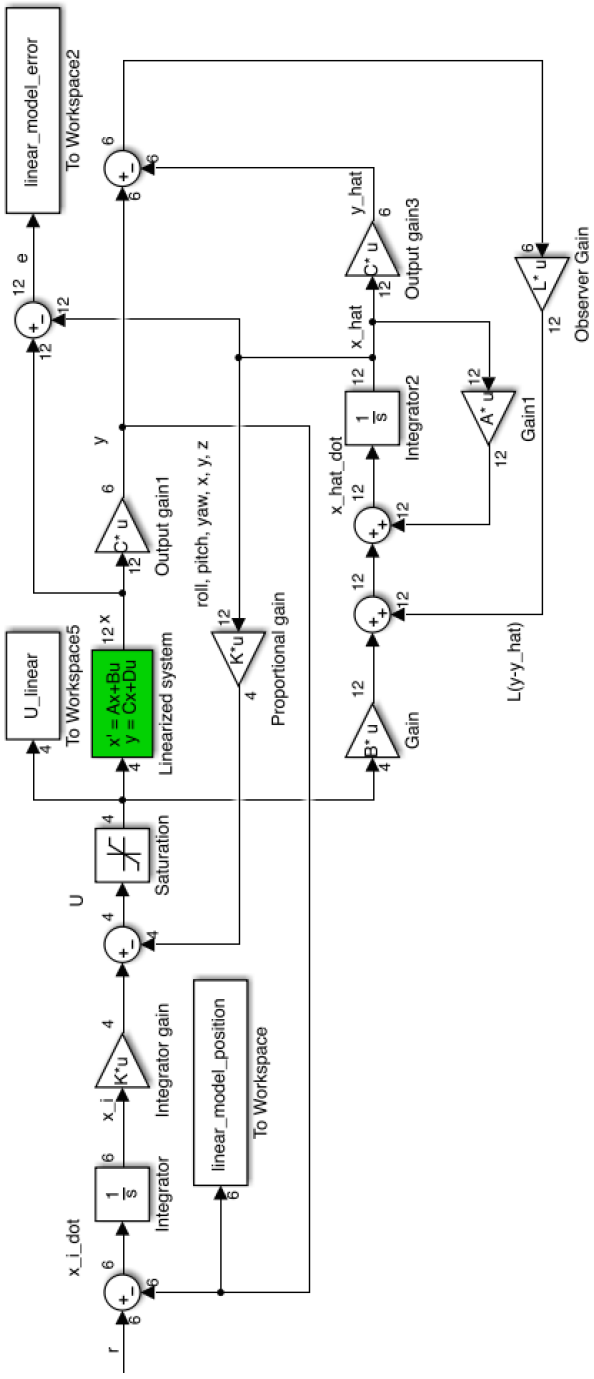

Figure 5.1: Block diagram of the linear system with LQR controller and observer

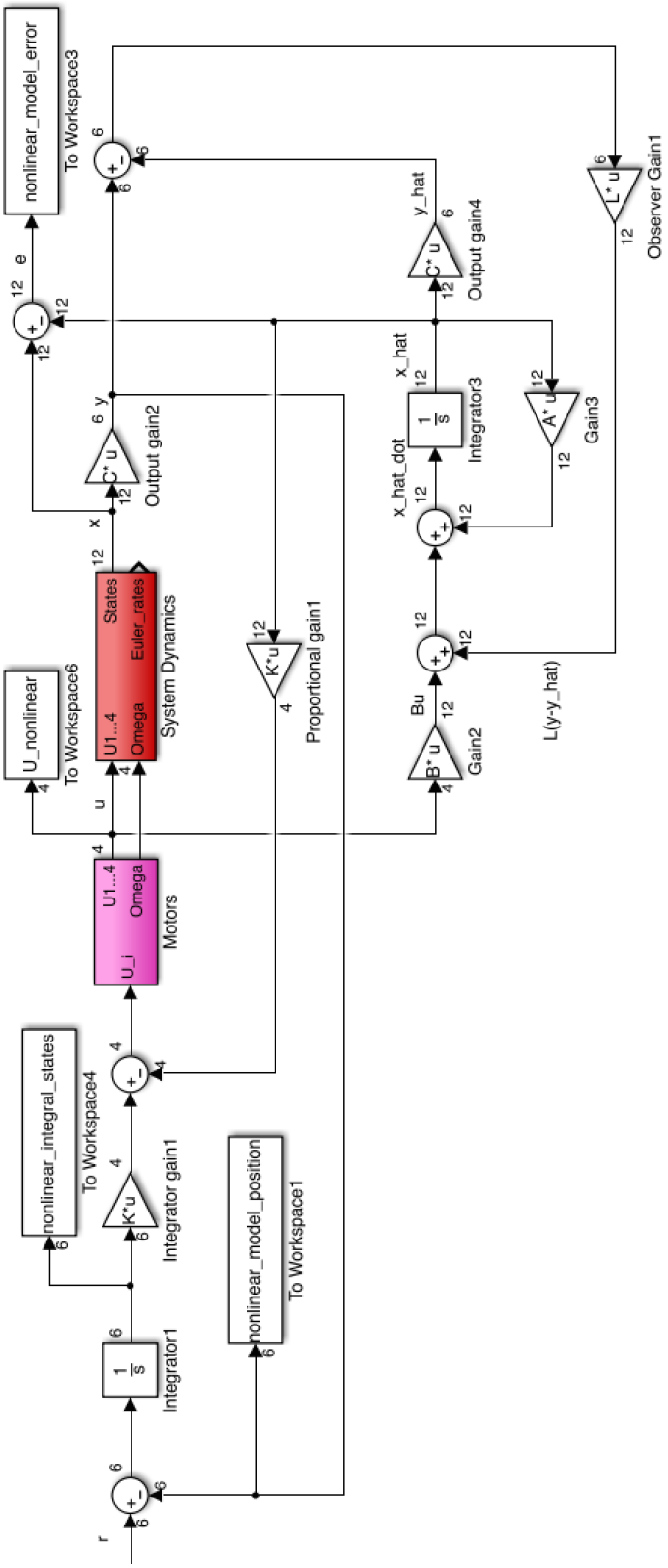

Figure 5.2: Block diagram of the nonlinear system with LQR controller and observer

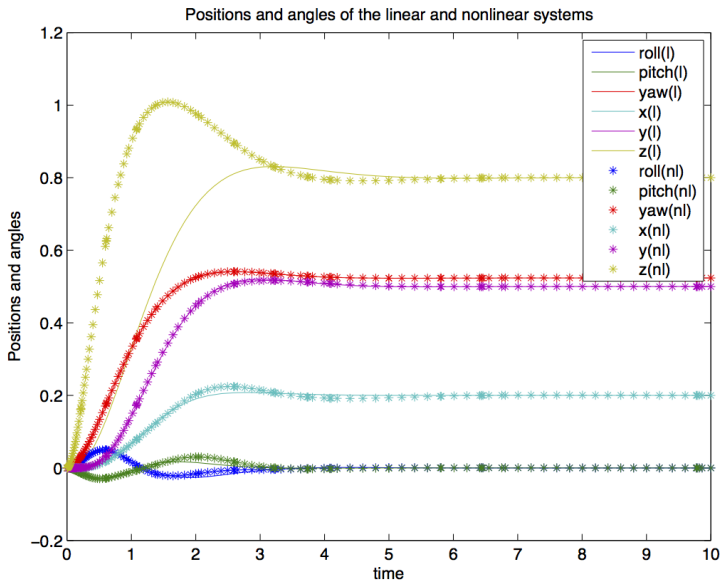

Figure 5.3: Positions and angles of linear (l) and nonlinear (nl) systems with LQR controller and observer

From Fig. 5.3, we can see that the linear system reached the reference level faster and experienced less overshoot, since it is the model used to design the LQR controller. For the nonlinear system, the x and y positions generally matched the linear system; however, the peak values of z exceeded the reference by about 0.2 m, and settles at 0.8 m at 6 s. The control inputs in both systems are shown in Fig. 5.4.

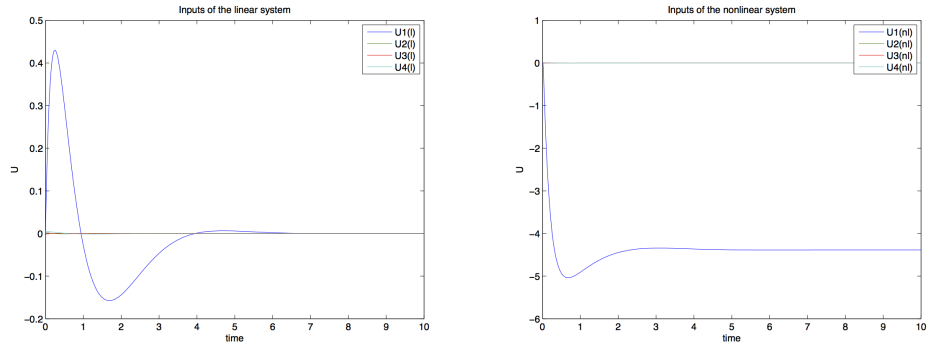

Figure 5.4: Control inputs of linear (l) and nonlinear (nl) systems with LQR controller and observer

Finally, the LQR controller was transferred to the physical drone to test the design quality. Since the motors only take in commands between 0 and

1 [15], the control inputs generated by the controller were normalized such that the equilibrium value corresponded to the motor RPM that stabilizes the drone, which was found to be 0.69 by experiments.

In the tests, the drone was programmed to ascend to a height of 0.5 m while maintaining the same planar position as the initial point. The yaw angle was not controlled but the yaw speed was accounted for in u_4. The drone managed to take off vertically from the starting point and reached about 0.58 m at peak, then started to descend to the desired height. However, due to the inaccuracy of the speed measurement in the z direction, the motor inputs were saturated to the maximum value 1. This saturation resulted in a considerable horizontal drift because of the asymmetry in motors, which failed to stabilize the drone. Future work should be focused on obtaining better estimates of the linear velocities.

5.2 Switched Controller

Before generating the switched controller, we need to design the operation modes and rules for valid transitions between modes. Similar to the second example shown in Chapter 3, three modes were considered: the unobstructed mode (mode 1), obstacle-in-x mode (mode 2) and obstacle- in-y mode (mode 3). In mode 1, all linear positions were penalized equally, so were the corresponding velocities. For mode 2 and 3, more weights were added to the states associated with the x direction and y direction, respectively. Therefore, the performance states are written as

$$z_1 = \Big[\; 3p \quad 3q \quad 3r \quad 3\dot{x} \quad 3\dot{y} \quad 3\dot{z} \quad 5\phi \quad 5\theta \quad 5\psi \quad 10x \quad 10y \quad 10z \quad 3u_1 \quad 3u_2 \quad 3u_3 \quad 3u_4 \; \Big]^T \tag{5.13}$$

$$z_2 = \Big[\; 3p \quad 3q \quad 3r \quad 5\dot{x} \quad \dot{y} \quad 3\dot{z} \quad 5\phi \quad 5\theta \quad 5\psi \quad 30x \quad 3y \quad 10z \quad 3u_1 \quad 3u_2 \quad 3u_3 \quad 3u_4 \; \Big]^T \tag{5.14}$$

$$z_3 = \Big[\; 3p \quad 3q \quad 3r \quad \dot{x} \quad 5\dot{y} \quad 3\dot{z} \quad 5\phi \quad 5\theta \quad 5\psi \quad 3x \quad 30y \quad 10z \quad 3u_1 \quad 3u_2 \quad 3u_3 \quad 3u_4 \; \Big]^T \tag{5.15}$$

It is assumed that the system can remain in the current mode, and mode 1 can jump to either mode 2 or mode 3. But to move between mode 2 and 3, the system is required to pass mode 1. This transit diagram is characterized by matrix Q:

$$Q = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \tag{5.16}$$

where

$$\begin{cases} Q_{ij} = 1 & valid\, transition\, from\, mode\, i\, to\, j \\ Q_{ij} = 0 & invalid\, transition\, from\, mode\, i\, to\, j \end{cases} \tag{5.17}$$

We consider the controllers with $L = H = 1$ and $T = 1$, which means the controller has access to the system parameters in 3 modes, including the current mode, while the performance is evaluated over 4 modes. Based on the above information, a complete transition diagram was first generated. Then the invalid paths were removed by checking the entries in Q.

Table 5.1: Switching sequence for a three-mode system

past	current	future	i_-	i_+	i
1	1	1	11	11	111
1	1	2	11	12	112
1	1	3	11	13	113
1	2	1	12	21	121
1	2	2	12	22	122
1	3	1	13	31	131
1	3	3	13	33	133
2	1	1	21	11	211
2	1	2	21	12	212
2	1	3	21	13	213
2	2	1	22	21	221
2	2	2	22	22	222
3	1	1	31	11	311
3	1	2	31	12	312
3	1	3	31	13	313
3	3	1	33	31	331
3	3	3	33	33	333

As shown in Table. 5.1, this procedure was done by the function *path_search*, which also provided the $i_- = i_{(-\bar{L}:H-1)}$ and $i_+ = i_{(-\bar{L}+1:H)}$ that are used to

test stability.

After discretizing the system at an interval of 0.1 s, controllers were determined with $\gamma = 0.5$ and connected with the linear system as shown in Fig. 5.5.

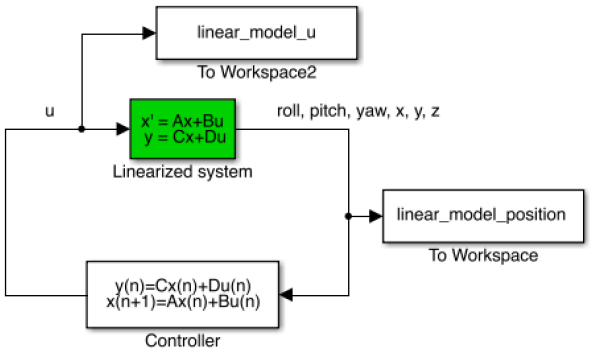

Figure 5.5: Block diagram of the switched controller

The system performance on three different paths is presented in Fig. 5.6. The simulation was run for a period of 15 s with initial value:

$$x_0 = y_0 = 1,\ z_0 = 0.5 \tag{5.18}$$

Unlike path 111, with imbalance on weights of x and y, path 333 and path 131 show clear separations between these two states, and converge to 0 noticeably faster. Comparing path 131 to 333, there are only slight differences between x and y initially and in the long term, while the system behaves similar to 333 in the middle of the simulation. The performance of the system perfectly matches with our expectations and demonstrates the benefits of switched control. Future work includes increasing the memory and horizon, and implementation of the controller on the physical drone.

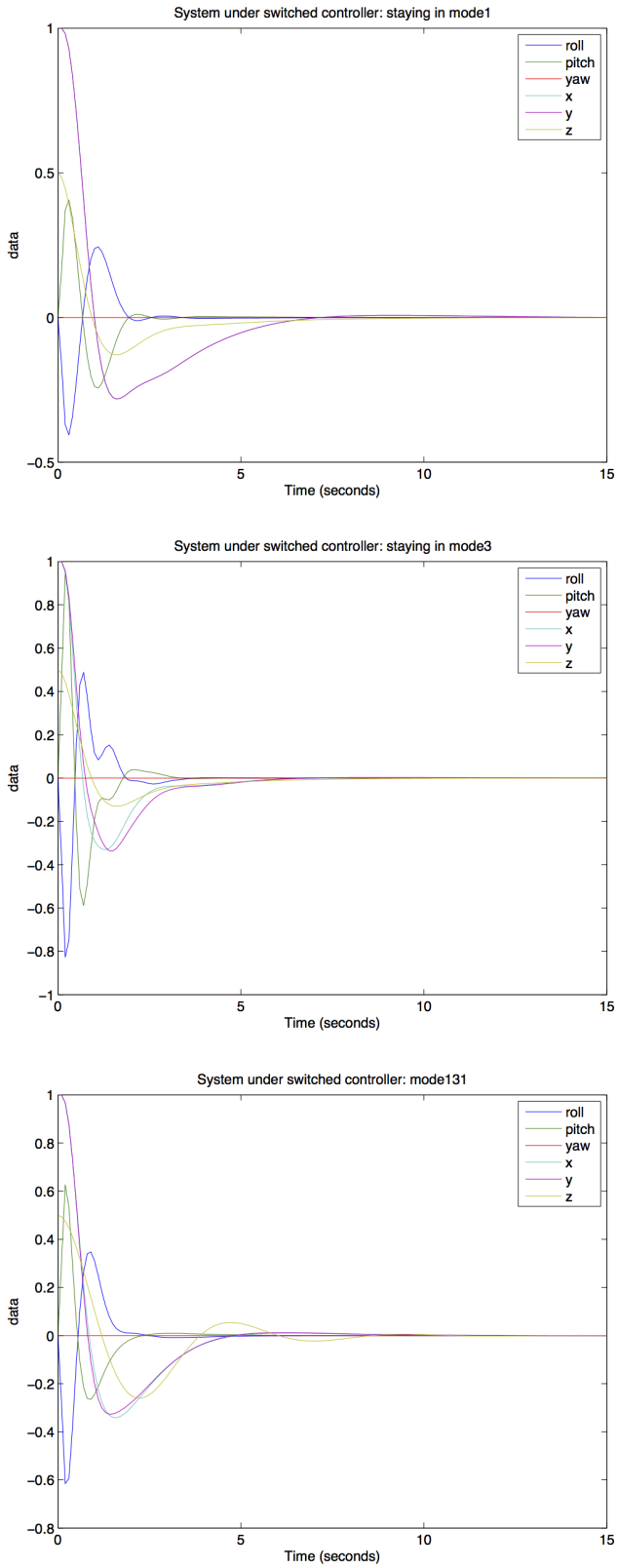

Figure 5.6: System performance on path 111, 333 and 131

CHAPTER 6

CONCLUSIONS

This thesis consists of the theory and synthesis results of the switched control, the dynamics and model of an A.R. Drone, the design and simulation of the LQR controller and switched controller, and the associated preliminaries and notations.

For the preliminaries, theories on the LQR control and basic knowledge of SDP are provided. A LQR controller stabilizes a linear continuous-time system with negative feedback, and the control gains are calculated by solving the ARE. SDP serves as an efficient method for solving minimization problems with inequality constraints, on which the SDP3 solver in CVX is based to test stability for a switching system.

We considered switched controllers that are path-dependent and have finite memory of past plant parameters and finite foreknowledge of future parameters. The linear and discrete-time system is designed to have multiple operation modes that vary in the model dynamics or performance evaluations, in presence of disturbances. Convex synthesis problems for each admissible path were expressed as linear matrix inequalities then solved by CVX in MATLAB, in order to generate the controllers depending on the specified performance levels.

When modeling the A.R, Drone, a 6-DOF system was established by analyzing the kinemics and dynamics. Due to the rotational effects of the propellers, the model is nonlinear and needs to be linearized for the design of the LQR and switched controllers. Partial derivatives with respect to each state were taken at equilibrium points to construct the linear system.

For the design of the LQR controller, proper weights were assigned to the

states and control inputs in the cost function to guarantee stability. The original system was extended by some integral states in order to track reference signals. We simulated the controller on both the nonlinear and linear models and compared the results. The closed-loop system successfully followed the desired trajectory but experienced moderate overshoot. The controller was also implemented on the physical drone, but failed to maintain at the specified height due to saturations and asymmetry in motors. The calculated switched controller was also simulated in MATLAB, and the system performance was compared among different switching sequences. The simulation showed satisfactory results, and the implementation should be included in future work.

APPENDIX A

CVX CODES FOR SWITCHED CONTROLLER

In this appendix, the CVX codes used to search for admissible paths, determine minimal performance level and generate the stabilizing controller are provided, based on Kan Chen's contribution.

A.1 *path_search*

```
function [path,pathtmp] = path_search(N,M,H,Q)
%% *********possible path searching******
% *****list all the possible path******
% M = L = L_hat, H = 0 in the original codes
if M+H==0
    % no memory, no knowledge about furture
    pathtmp(1:N)=1:N; % all modes
    path=[(1:N)' (1:N)' (1:N)' (1:N)'];
else
    % list all the path without considering the contraining
        matrix Q
    path=PermsRep(1:N,M+H+1); % size: N^(M+H+1) * (M+H+1)
    % ****filter the path "M+H" and "N" according to Q******
    % if Q(i,j)=0 then it is impossible to jump from mode i to
        mode j
    k=1;
    for i=1:size(path,1) % size(path,1) = M+H+1
        incr=1;
        for j=1:M+H
            if Q(path(k,j),path(k,j+1))==0
                % clear the row that corresponds to the
                    inadmissable path
                path(k,:)=[]; % delete that row
                incr=0;
                break
```

```
23              end
24          end
25          k=k+incr;
26          % if previous row gets removed, the next row has the
                same index
27      end
28      % #rows in path = # of admissiable paths
29
30      % ****store path, path+ and path- into one variable****
31      for i=1:size(path,1) % size(path,1) = # of admissiable paths
32          path(i,M+H+2)=0; % for path-: cascading nodes 1 ~ M+H
33          path(i,M+H+3)=0; % for path+: 2 ~ M+H+1
34          path(i,M+H+4)=0; % for whole path
35          for j=1:M+H
36              path(i,M+H+2)=path(i,M+H+2)+path(i,j)*10^(M+H-j); %
                    scalar
37              path(i,M+H+3)=path(i,M+H+3)+path(i,j+1)*10^(M+H-j);
                    % scalar
38          end
39          for j=1:M+H+1
40              path(i,M+H+4)=path(i,M+H+4)+path(i,j)*10^(M+H-j+1);
                    % scalar
41          end
42          % pathtmp used to avoid overlapping declarance in LMI
43          % odd number of rows for path-
44          pathtmp((i-1)*2+1)=path(i,M+H+2);
45          % even number of rows for path+
46          pathtmp((i-1)*2+2)=path(i,M+H+3);
47      end
48      if exist('pathtmp')~=0
49          % remove the repeated paths
50          % pathtemp = unique admissiable paths with length M+H
51          pathtmp=unique(pathtmp);
52      else
53          exists=0;
54          pathtmp=[];
55      end
56      path=unique(path,'rows');
57  end
58  end
59
60  function res = PermsRep(v,k)
61  %  PERMSREP Permutations with replacement.
```

```
62 %
63 %  PermsRep(v, k) lists all possible ways to permute k elements
       out of
64 %  the vector v, with replacement.
65
66 if nargin<1 || isempty(v)
67     error('v must be non-empty')
68 else
69     n = length(v);
70 end
71
72 if nargin<2 || isempty(k)
73     k = n;
74 end
75
76 v = v(:).'; % Ensure v is a row vector
77 for i = k:-1:1
78     tmp = repmat(v,n^(k-i),n^(i-1));
79     res(:,i) = tmp(:);
80 end
81 end
```

A.2 *receding_horizon*

```
function [var_m,K] = receding_horizon(A,B1,B2,C1,C2,D11,D12,D21,
    M,H,gamma,path,pathtmp,pathT)
K=[];
B = struct('B1',B1,'B2',B2);
C = struct('C1',C1,'C2',C2);
D = struct('D11',D11,'D12',D12,'D21',D21);
N=size(A,3); % the number of modes
m1=size(B.B1,2); % size of w (column vector)
m2=size(B.B2,2); % size of u
l1=size(C.C1,1); % size of z
l2=size(C.C2,1); % size of y
n1=size(A,1); % number of states in each mode
T=size(pathT(1).value,2)-1; % performance window length
%% ***preprocess of data before LMI solver****
% ****F, G, F_hat, G_hat*****
% define for each mode
for i=1:N
    F(:,:,i)=[zeros(n1) zeros(n1) zeros(n1) eye(n1) zeros(n1,m1)
        ;
        zeros(m2,n1) zeros(m2,n1) B.B2(:,:,i)' zeros(m2,n1)
            zeros(m2,m1)];
    G(:,:,i)=[zeros(n1) eye(n1) zeros(n1) zeros(n1) zeros(n1,m1)
        ;
        C.C2(:,:,i) zeros(l2,n1) zeros(l2,n1) zeros(l2,n1) D.D21
            (:,:,i)];
    F_hat(:,:,i)=[zeros(n1) zeros(n1) zeros(n1,l1) zeros(n1,m1)
        zeros(m2,n1) zeros(m2,n1) D.D12(:,:,i)' zeros(m2,m1)];
    G_hat(:,:,i)=[zeros(n1) eye(n1) zeros(n1,l1) zeros(n1,m1);
        C.C2(:,:,i) zeros(l2,n1) zeros(l2,l1) D.D21(:,:,i)];
end
% break
%% ****start calculating LMI with cvx*****
%****for the case that M+H=0*****
% stay at one mode
if M+H==0
    cvxq=cvx_quiet(true);
    cvx_begin sdp
    variable var_m
    % dimension check
    variable R_1(n1,n1) symmetric
    variable S_1(n1,n1) symmetric
```

```
37      variable Z_1(l1,l1) symmetric
38      variable W_1(n1+m2,n1+l2)
39      minimize (var_m)
40      % define H and H_hat for all modes: i- = i+ = current mode
41      for i=1:N
42          % H and H_hat are symmetric
43          H=[-S_1 -eye(n1) A(:,:,i)' A(:,:,i)'*S_1 zeros(n1,m1);
44              -eye(n1) -R_1 R_1*A(:,:,i)' zeros(n1) zeros(n1,m1);
45              A(:,:,i) A(:,:,i)*R_1 -R_1 -eye(n1) B.B1(:,:,i);
46              S_1*A(:,:,i) zeros(n1) -eye(n1) -S_1 S_1*B.B1(:,:,i)
                    ;
47              zeros(m1,n1) zeros(m1,n1) B.B1(:,:,i)' B.B1(:,:,i)'*
                    S_1 -eye(m1);];
48          H_hat=[-S_1 -eye(n1) C.C1(:,:,i)' zeros(n1,m1);
49              -eye(n1) -R_1 R_1*C.C1(:,:,i)' zeros(n1,m1);
50              C.C1(:,:,i) C.C1(:,:,i)*R_1 -Z_1 D.D11(:,:,i);
51              zeros(m1,n1) zeros(m1,n1) D.D11(:,:,i)' -eye(m1)];
52          trace(Z_1) - gamma^2 < var_m; % only one Z_1, no need to
                average
53          H+F(:,:,i)'*W_1*G(:,:,i)+G(:,:,i)'*W_1'*F(:,:,i) < var_m
                *eye(size(H,1));
54          H_hat+F_hat(:,:,i)'*W_1*G_hat(:,:,i)+G_hat(:,:,i)'*W_1'*
                F_hat(:,:,i) < var_m*eye(size(H_hat,1));
55      end
56      cvx_end
57  end
58  %****for the case that M+H>0*****
59  if M+H>=1;
60      cvx_clear;
61      cvx_precision high
62      cvx_solver SDPT3
63      cvx_quiet(true)
64      cvx_begin sdp
65      variable var_m
66      % ******define the variables*******
67      % define R and S
68      for i=1:length(pathtmp) % number of valid one-step
            transitions
69          s1= ['variable R_'];
70          s2= ['variable S_'];
71          s3= [int2str(pathtmp(i))];
72          s4= [s1 s3 '(n1,n1) symmetric'];
73          eval(s4)
```

```
74          s5= [s2 s3 '(n1,n1) symmetric'];
75          eval(s5)
76      end
77      % define Z and W
78      for i=1:length(path(:,M+H+4)) % number of valid (M+H)-step
            transitions
79          s1= ['variable Z_'];
80          s2= ['variable W_'];
81          s3= [int2str(path(i,M+H+4))];
82          s4= [s1 s3 '(l1,l1) symmetric'];
83          eval(s4)
84          s5= [s2 s3 '(n1+m2,n1+l2)'];
85          eval(s5)
86      end
87      minimize (var_m)
88      clear s1 s2 s3 s4 s5
89      for i=1:size(path,1) % number of valid (M+H)-step
            transitions
90          % equation for H_i
91          % path(i,M+1) is the current mode!
92          s1=['H_' int2str(path(i,M+H+4)) '=[-S_' int2str(path(i,M
                +H+2)) ' -eye(n1) A(:,:,' int2str(path(i,M+1)) ')'' A
                (:,:,' int2str(path(i,M+1))];
93          s1=[s1 ')''*S_' int2str(path(i,M+H+3)) ' zeros(n1,m1);-
                eye(n1) -R_' int2str(path(i,M+H+2))];
94          s1=[s1 ' R_' int2str(path(i,M+H+2)) '*A(:,:,' int2str(
                path(i,M+1)) ')'' zeros(n1) zeros(n1,m1);'];
95          s1=[s1 'A(:,:,' int2str(path(i,M+1)) ') A(:,:,' int2str(
                path(i,M+1)) ')*R_' int2str(path(i,M+H+2))];
96          s1=[s1 ' -R_' int2str(path(i,M+H+3)) ' -eye(n1) B.B1
                (:,:,' int2str(path(i,M+1)) ');'];
97          s1=[s1 'S_' int2str(path(i,M+H+3)) '*A(:,:,' int2str(
                path(i,M+1)) ') zeros(n1) -eye(n1) -S_' int2str(path(
                i,M+H+3)) ' S_' int2str(path(i,M+H+3))];
98          s1=[s1 '*B.B1(:,:,' int2str(path(i,M+1)) ');zeros(m1,n1)
                 zeros(m1,n1) B.B1(:,:,' int2str(path(i,M+1)) ')'' B.
                B1(:,:,'];
99          s1=[s1 int2str(path(i,M+1)) ')''*S_' int2str(path(i,M+H
                +3)) ' -eye(m1)];'];
100         eval(s1)
101         s1=['H_' int2str(path(i,M+H+4)) '+F(:,:,' int2str(path(i
                ,M+1)) ')''*W_' int2str(path(i,M+H+4))];
```

```
102         s1=[s1 '*G(:,:,' int2str(path(i,M+1)) ')+G(:,:,' int2str
                (path(i,M+1)) ')''*W_' int2str(path(i,M+H+4))];
103         s1=[s1 ''')*F(:,:,' int2str(path(i,M+1)) ') <= var_m*eye(
                size(H_' int2str(path(i,M+H+4)) ',1));'];
104         eval(s1)
105         % equation for H_hat_i
106         s1=['H_hat_' int2str(path(i,M+H+4)) '=[-S_' int2str(path
                (i,M+H+2)) ' -eye(n1) C.C1(:,:,' int2str(path(i,M+1))
                 ')'' zeros(n1,m1);'];
107         s1=[s1 '-eye(n1) -R_' int2str(path(i,M+H+2)) ' R_'
                int2str(path(i,M+H+2)) '*C.C1(:,:,' int2str(path(i,M
                +1)) ')'' zeros(n1,m1);'];
108         s1=[s1 'C.C1(:,:,' int2str(path(i,M+1)) ') C.C1(:,:,'
                int2str(path(i,M+1)) ')*R_' int2str(path(i,M+H+2))];
109         s1=[s1 ' -Z_' int2str(path(i,M+H+4)) ' D.D11(:,:,'
                int2str(path(i,M+1)) ');zeros(m1,n1) zeros(m1,n1) D.
                D11(:,:,' int2str(path(i,M+1)) ')'' -eye(m1)];'];
110         eval(s1)
111         s1=['H_hat_' int2str(path(i,M+H+4)) '+F_hat(:,:,'
                int2str(path(i,M+1)) ')''*W_' int2str(path(i,M+H+4))
                ];
112         s1=[s1 '*G_hat(:,:,' int2str(path(i,M+1)) ')+G_hat(:,:,'
                 int2str(path(i,M+1)) ')''*W_' int2str(path(i,M+H+4))
                ];
113         s1=[s1 ''')*F_hat(:,:,' int2str(path(i,M+1)) ') <= var_m*
                eye(size(H_hat_' int2str(path(i,M+H+4)) ',1));'];
114         eval(s1)
115     end
116     for i=1:size(pathT,2) % size(pathT,2) = # of control
            sequence from each performance sequence
117         s1=['1/(T+1)*('];
118         for j=1:T+1
119             s1=[s1 '+trace(Z_' int2str(pathT(i).value(j)) ')'];
120         end
121         s1=[s1 ') - gamma^2 <= var_m'];
122         eval(s1)
123     end
124     cvx_end
125 end
126
127 %*******Solve for Controller if system is stabilizable
        ********************
```

```
128 if var_m < 0 && nargout == 2 % number of function output
    arguments
129     % define U and T
130     for i=1:length(pathtmp) % number of valid one-step
        transitions
131         if n1==1 % only one state in each mode
132             s1=['U_' int2str(pathtmp(i)) '=sqrt(S_' int2str(
                pathtmp(i))];
133         else
134             s1=['U_' int2str(pathtmp(i)) '=sqrtm(S_' int2str(
                pathtmp(i))];
135         end
136         s1=[s1 '-inv(R_' int2str(pathtmp(i)) ')*eye(n1));']; %
            U = (S - inv(R))^(1/2) * eye(n1)
137         eval(s1)
138         s1=['T_' int2str(pathtmp(i)) '=-R_' int2str(pathtmp(i))
              '*U_' int2str(pathtmp(i)) ';'];
139         % T = -R * U
140         eval(s1)
141     end
142     if M+H>=1 % more than more mode
143         for i=1:size(path,1)
144             s1=['W_' int2str(path(i,M+H+4))];
145             s1=[s1 '-blkdiag(S_' int2str(path(i,M+H+3)) '*A
                (:,:,' int2str(path(i,M+1)) ')'];
146             s1=[s1 '*R_' int2str(path(i,M+H+2)) ',zeros(m2,l2))
                '];
147             s2=['[U_' int2str(path(i,M+H+3)) ',S_' int2str(path
                (i,M+H+3)) '*B.B2(:,:,' int2str(path(i,M+1)) ')'
                ];
148             s2=[s2 ';zeros(m2,n1),eye(m2)]'];
149             s3=['[T_' int2str(path(i,M+H+2)) ''',zeros(n1,l2);C
                .C2(:,:,' int2str(path(i,M+1)) ')*R_' int2str(
                path(i,M+H+2)) ',eye(l2)]'];
150             s1=['K_' int2str(path(i,M+H+4)) '=inv(' s2 ')*(' s1
                  ')*inv(' s3 ');'];
151             eval(s1)
152         end
153     else % no transition
154         for i=1:size(path,1)
155             s1=['W_1-blkdiag(S_1*A(:,:,' int2str(path(i,M+1)) '
                )'];
156             s1=[s1 '*R_1,zeros(m2,l2))'];
```

```
157                 s2=['[U_1,S_1*B.B2(:,:,' int2str(path(i,M+1)) ')'];
158                 s2=[s2 ';zeros(m2,n1),eye(m2)]'];
159                 s3=['[T_1'',zeros(n1,l2);C.C2(:,:,' int2str(path(i,
                        M+1)) ')*R_1,eye(l2)]'];
160                 s1=['K_' int2str(path(i,M+H+4)) '=inv(' s2 ')*(' s1
                        ')*inv(' s3 ');'];
161                 eval(s1)
162             end
163         end
164         for i=1:length(path(:,M+H+4))
165             s1=['K.K_' int2str(path(i,M+H+4)) '=K_' int2str(path(i,
                    M+H+4)) ';'];
166             eval(s1)
167         end
168     end
```

A.3 *rh_test*

```
function [status,K] = rh_test(A,B1,B2,C1,C2,D11,D12,D21,Q,M,H,T,
    gamma)
% M = L = L_hat; H = 0 in the original codes
% T is the performance window
N=size(A,3); % # of modes
[path,pathtmp]=path_search(N,M,H,Q); % search for control
    sequence
[pathTtmp,pathtmpT]=path_search(N,M,H+T,Q); % search for
    performance sequence
for i=1:size(pathTtmp,1) % size(pathTtmp,1) = # of admissiable
    performance sequence
    for j=0:T
        tmp=[];
        for k=1:M+H+1
            tmp=[tmp int2str(pathTtmp(i,k+j))];
        end
        % length of tmp = M+H+1
        pathT(i).value(j+1)=str2num(tmp);
        % taking a string of M+H+1 successive modes (control
            sequence) from the first M+H+T+1
        % modes in each performance sequence
    end
end
clear tmp pathtmpT pathTtmp i j k
K=[];
if isempty(path)
    error('invalid Q, no path exists');
else
    [var_m,K] = receding_horizon(A,B1,B2,C1,C2,D11,D12,D21,M,H,
        gamma,path,pathtmp,pathT);
    if var_m < 0
        status='solved, the system is stabilizable';
    else
        status='solved, the system is not stabilizable';
    end
end
disp(status)
end
```

REFERENCES

[1] "The A.R. Drone website." [Online]. Available: http://cdn.ardrone2.parrot.com

[2] M. C. Grant and S. P. Boyd, *The CVX Users' Guide*, 2nd ed., CVX Research, Inc.

[3] M. Pena, E. Vivas, and C. Rodriguez, "Simulation of the quadrotor controlled with LQR with integral effect," *ABCM Symposium Series in Mechatronics*, vol. 5, p. 390, 2012.

[4] L.Vandenberghe and S.Boyd, "Semidefinite programming," *SIAM Review*, vol. 38, no. 1, pp. 49–95, March 1996.

[5] D. Liberzon and A. Morse, "Basic problems in stability and design of switched systems," *IEEE Control Syst*, vol. 19, pp. 59–70, 1999.

[6] G. E. Dullerud and F. Paganini, *A Course in Robust Control Theory.* Springer, 1999.

[7] J.-W. Lee, G. Dullerud, and P. Khargonekar, "Path-by-path optimal control of switched and Markovian jump linear systems," *Proc. 46th IEEE Conf. Decision Control*, 2008.

[8] R. Essick, J.-W. Lee, and G. E. Dullerud, "Control of linear switched systems with receding horizon modal information," *IEEE Transactions on Automatic Control*, vol. 59, no. 9, Septermber 2014.

[9] K. Chen, "Switched linear stabilization synthesis tools manual," 2009, unpublished.

[10] R. W. Beard, "Quadrotor dynamics and control," unpublished.

[11] J. Vervoorst, "Quadrotor model," unpublished.

[12] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Prentice Hall, 2001.

[13] C.-C. Tsui, "Observer design — a survey," *International Journal of Automation and Computing*, vol. 12, pp. 50–61, 2015.

[14] B. P. K. Higuchi and F. lix Mora-Camino, "Attitude control of a quadrotor aircraft using LQR state feedback controller with full order state observer," *SICE Annual Conference*, 2013.

[15] Y. Sun, "Modeling, identification and control of a quad-rotor drone using low-resolution sensing," M.S. thesis, University of Illinois at Urbana-Champaign, 2012.