SEVİNÇ GÜNSEL

M.S. Thesis

2020

OPTIMAL AND MODEL PREDICTIVE CONTROL OF A QUADROTOR

SEVİNÇ GÜNSEL

IŞIK UNIVERSITY
2020

OPTIMAL AND MODEL PREDICTIVE CONTROL OF A QUADROTOR

SEVİNÇ GÜNSEL
B.S., Mechatronics Engineering, IŞIK UNIVERSITY, 2017

Submitted to the Graduate School of Science and Engineering
in partial fulfillment of the requirements for the degree of
Master of Science
in
Electronics Engineering

IŞIK UNIVERSITY
2020

IŞIK UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

OPTIMAL AND MODEL PREDICTIVE CONTROL OF A QUADROTOR

SEVİNÇ GÜNSEL

APPROVED BY:

Prof. Yorgo Istefanopulos (Thesis Supervisor)	Işık University	
Assoc. Prof. Erkin Dinçmen (Thesis Co-Supervisor)	Işık University	
Prof. Ümit Güz	Işık University	
Prof. Mustafa Doğan	ITU	

APPROVAL DATE: ..../..../....

OPTIMAL AND MODEL PREDICTIVE CONTROL OF A QUADROTOR

Abstract

Quadrotors are unmanned aerial vehicles (UAVs) used widely and these vehicles have been controlled with different controllers, such as PID, Linear Quadratic Regulator (LQR) and, Model Predictive Control (MPC). In this study, the linear state-space model was developed according to quadrotor's kinematic and dynamic equations, and LQR and MPC controllers were designed for tracking a given trajectory. The simulation results of the controller performances were compared.

Keywords: Quadrotor, MPC, LQR, trajectory tracking, control

QUADROTOR'UN OPTİMAL VE MODEL ÖNGÖRÜLÜ KONTROLÜ

Özet

Quadrotorlar yaygın kullanılan insansız hava araçlarıdır (İHA) ve PID, Lineer Kuadratik Regülatör (LQR) ve Model Öngörülü Kontrol (MPC) gibi farklı kontrolörler ile kontrol edilmiştir. Bu çalışmada, quadrotorun kinematik ve dinamik denklemlerine göre lineer durum uzay modeli geliştirildi ve verilen bir yörüngeyi izlemek için LQR ve MPC kontrolörleri tasarlandı. Kontrolör performanslarının simülasyon sonuçları karşılaştırıldı.

Anahtar kelimeler: Quadtoror, MPC, LQR, yörünge izleme, kontrol

Acknowledgements

This study was completed thanks to my thesis supervisors Prof. Dr. Yorgo Istefanopulos and Assoc Prof. Dr. Erkin Dinçmen and Prof. Dr. Mustafa Doğan who helped with their extensive knowledge and experiences. I owe them a debt of gratitude.

I would like to thank my friends Ece Yener and Ege Sarı who never made me feel lonely on this rocky road. They encouraged me whenever I wanted to give up. I thank other friends Berk Kenan Mataracı who gave me suggestions to find a thesis subject and for his endurance in all steps of the thesis and Oğuzhan Ulusoy who shared his knowledge about coding with me and always tried to be helpful to me.

I also want to thank my lovely cousin Güldamla Özsema for her supports.

Finally, I would like to express my special thanks to my dear family, my mom and dad for their love and patience.

To The Students of The Future

Table of Contents

List of Figures

List of Abbreviations

UAV	**U**nmanned **A**erial **V**ehicle
VTOL	**V**ertical **T**ake-**O**ff **L**anding
MIMO	**M**ultiple **I**nput **M**ultiple **O**utput
PD	**P**roportional **D**erivative
PID	**P**roportional **I**ntegral **D**erivative
LQR	**L**inear **Q**uadratic **R**egulator
6-DOF	**6** **D**egree **o**f **F**reedom
MPC	**M**odel **P**redictive **C**ontrol
RHC	**R**eceding **H**orizon **C**ontrol

Chapter 1

Introduction

In the last ten years, Unmanned Aerial Vehicles (UAVs) have become the focus of research and development. In particular, Quadrotors are more preferred due to their Vertical Take-Off and Landing (VTOL) capability and movement flexibility. Moreover, Quadrotors have importance in the field of Aeronautic Engineering, military, and civilian applications. Such as environmental monitoring, lifesaving, battlefield assessment, and various industrial applications [1].

Quadrotors have a simple and symmetric structure in virtue of four equally spaced rotors. This structure provides six degrees of freedom as coupled with translational and rotational motion. With the aerodynamic effects, the dynamics of the Quadrotors have become considerable nonlinear and causing the stability more complicated. Depending on these factors, controlling the Quadrotor becomes harder [2].

A considerable amount of control techniques have been tested to deal with multiple inputs and multiple outputs (MIMO) nonlinear systems in both classical and modern control theories. The linear model of the Quadrotor is attained by applying a linearization procedure around an equilibrium point [3]. In general, the linear model would rather be preferred due to its ease of implementation and simple design.

Feedback linearization method, backstepping controller, PD, PID, sliding mode controller, and Linear Quadratic Regulator (LQR) controller were used to achieve a proper stability [4].In accordance with references [3] and [4], the feedback linearization method has been applied to obtain a proper trajectory tracking and the backstepping technique was used to obtain a more flexible controller with its interconnected subsystems. As a classical approach, PID presents a solution for an attitude control problem with a simplified dynamic model [4].]. Another control technique LQR is an optimal control method which aspires to achieve minimum objective function, in other words, a cost function is obtained from the system dynamics described in the state-space model. This function is in quadratic form and consists of differential Riccati Equations. The state feedback gain of the system is obtained by solving the Riccati Equations. Therefore, states are evolved with minimum energy consumption [1]. In addition to these, Model Predictive Control (MPC) method has been used in recent researches. When comparing with the other control methods, MPC is designed with robust control law for position control. Also, MPC provides stable rotational movements. Solves quadratic problem as LQR. However, it solves by making iteration periodically as distinct from the LQR controller [4].

In this study, the dynamic model of the system has been established in both continuous-time and discrete-time state-space models to track a given trajectory. Then, the LQR controller and MPC controller have been designed. All the system modellings and simulations have been done by using Matlab / Simulink. Finally, evaluated simulation results have been compared for LQR and MPC controllers

Chapter 2

Literature Survey

2.1 Working Principle

Quadrotors are simple unmanned aerial vehicles with symmetric structure. Configuration can be in two different types, ”cross” and ”plus” configurations, as shown in Figure 2.1. below. There are four aligned rotors in a quadrotor. While one pair of rotors, placed opposite to each other, rotate in the same direction, the rest one pair of rotors rotate in opposite direction with respect to the other pair.

Quadrotors have three transitional and three rotational movements. Therefore, we can understand that it has six degrees of freedom (6-DOF).

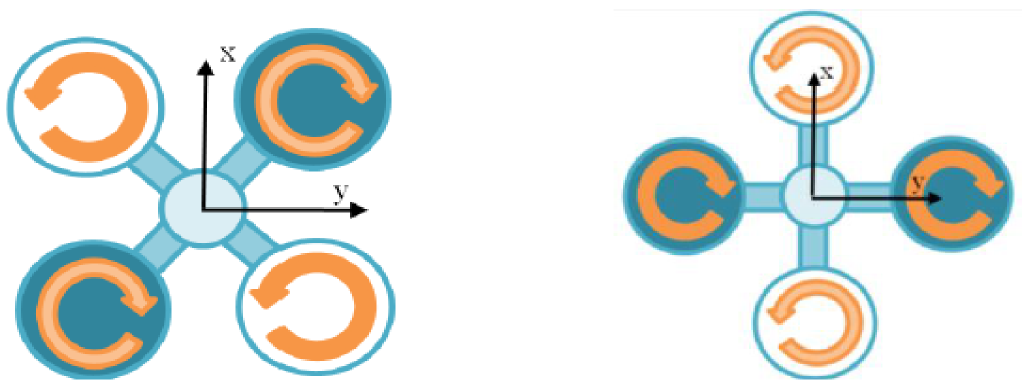


Figure 2.1: Cross and Plus Configurations.

By adjusting the angular velocity of each rotor, rotational movements on three axes are provided. The speed difference of rotor pairs creates roll, pitch, and yaw movements according to x, y, and z axes respectively [5].

2.2 Mathematical Model

According to reference [6], the system dynamics are modeled using Newton's Laws and Euler's Laws considering the following assumptions:

- The body is rigid.
- The tensor of inertia (ToI) of the Quadrotor is assumed as zero.
- The center of gravity (CoG) and the center of Mass (CoM) assumed to coincide with the geometrical center of the Quadrotor.
- The moment of inertia (MoI) of the propellers is neglected.
- The drag force is neglected.

2.2.1 Reference Frame

There are two frames to describe the linear and angular positions of the quadrotor. The axes on the quadrotor are called as body frame and the axes on the Earth are called as inertial frame. The linear position of the quadrotor in the inertial frame is defined as x, y, z. In the body frame, the axes are defined as x_b, y_b, z_b.

Angular position is called Euler angles and the Euler angles are defined as ϕ, θ, ψ. ϕ is the roll angle which determines rotations around the x-axis, θ is the pitch angle which determines rotations around the y-axis, and ψ is the yaw angle which determines rotations around the z-axis [7].

The orientation between the inertial frame and body frame is provided by an orthogonal rotations matrix R, which will be explained and derived in the next section.

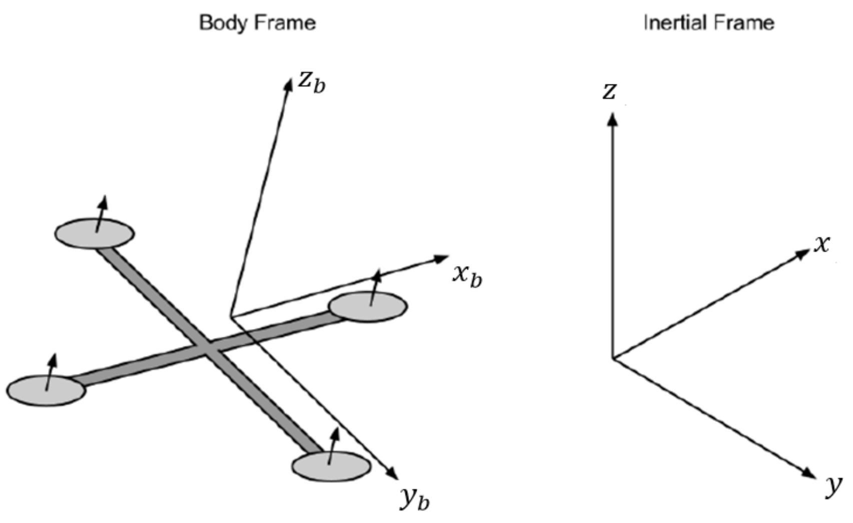


Figure 2.2: Inertial & Body Frames.

2.2.2 System Kinematics

Transitional Kinematics: Kinematic equations are obtained by geometric relationships between axes on both inertial and body frames. Roll movement is described by the relationship between y and z axes. Pitch movement is described by the relationship between x and z axes. Yaw movement is described by the relationship between x and y axes.

The transitional kinematics equations and matrix forms are as below [8]:

Roll Movement

$\mathrm{x}_b = x$

$\mathrm{y}_b = y.\cos\phi + z.\sin\phi$

$\mathrm{z}_b = -y.\sin\phi + z.\cos\phi$

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & sin\phi & \cos\phi \end{bmatrix} \tag{2.1}$$

Pitch Movement

$\mathrm{x}_b = x.\cos\theta - z.\sin\theta$

$\mathrm{y}_b = y$

$\mathrm{z}_b = x.\sin\theta + z.\cos\theta$

$$R_y = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.2}$$

Yaw Movement

$\mathrm{x}_b = x.\cos\psi + y.\sin\psi$

$\mathrm{y}_b = -x.\sin\psi + y.\cos\psi$

$\mathrm{z}_b = z$

$$R_z = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

The rotation matrix from the body frame to the inertial frame is found below [9]:

$\mathrm{R}=\mathrm{R}_z \times R_y \times R_x$

$$R = \begin{bmatrix} \cos\theta\cos\psi & \sin\phi sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \tag{2.4}$$

Rotational Kinematics: Angles on the body frame are denoted as p, q, and r according to the x, y, and z axes respectively. Trigonometric relationships between the Euler rates $\dot{\phi}$, $\dot{\theta}$, $\dot{\psi}$ and angular rates of body $\dot{p}$, $\dot{q}$, $\dot{r}$ are as follows [9]:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = R(\dot{\phi}) \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + R(\phi)R(\dot{\theta}) \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + R(\phi)R(\theta)R(\dot{\psi}) \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \tag{2.5}$$

To provide the transformation between the Euler rates and body rates, $\dot{\phi}$, $\dot{\theta}$, $\dot{\psi}$ need to be assumed small. Thus,

$$R(\dot{\phi}) = R(\dot{\theta}) = R(\dot{\psi}) = I \tag{2.6}$$

Then the equation becomes

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \tag{2.7}$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{2.8}$$

The transfer matrix is obtained by inverting

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\theta \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} \tag{2.9}$$

2.2.3 System Dynamics

Dynamic equations describe the quadrotor attitude. The angle of Roll, Pitch, and Yaw movement is procured by the rotational velocity of rotors. This angle is also affected by the weight of the quadrotor, wind and gravity.

Thrust force is produced through the rotor torques of brushless DC motors [8].

Motor Dynamics and Thrust In order to meet the power requirement of the system, motor power is determined. Motor torque is calculated before determining the motor power [10].

$$\tau = K_t I \tag{2.10}$$

where τ (Nm) is the rotor torque, K_t (Nm/A) is the torque constant, and I (A) is the input current. For DC power sources Kirchoff's Voltage Law can be applied. Motor resistance can be neglected because it is too small. According to the Kirchoff's Voltage Law, the voltage that generated by motor is given as:

$$V = (I.R) + V_{EMF} \tag{2.11}$$

R represents the motor resistance and if it is negligible, then the voltage that generated by motor equals to Beck-EMF voltage.

$$V = V_{EMF} \tag{2.12}$$

$$V_{EMF} = \omega . K_e \tag{2.13}$$

where V (volts) is the developed voltage, V_{EMF} (volts) is the voltage drop of the Back-EMF, R (ohm) is the motor resistance, I (A) is the current, ω (rpm)) is the angular velocity of the motor, and K_e (volts/(radian/sec)) is the Back-EMF constant.

If the voltage and the current values are known, the required electrical power of the motor can be calculated as follows [10]:

$$P = I.V \tag{2.14}$$

In accordance with the previous equations (2.10) and (2.12) τ / K_t can be written instead of I and $\omega . K_e$ can be written instead of V. Therefore the power requirement P (watt) becomes:

$$P = \frac{\tau \omega . K_e}{K_t} \tag{2.15}$$

Power requirement can also be described as

$$P = T.v_i \tag{2.16}$$

where T (N) is the thrust and v_i (m/s) is the inlet air velocity.

In hover case, the air is sucked in from all directions and passes through the rotor disk. The air passed through the disk is accelerated. Then, according to the Momentum Theorem accelerated mass flow of air is

$$\dot{m} = \rho A v_i \tag{2.17}$$

where A (m^2) is the disk area and ρ (kg/m^3) is the air density.

Thrust is produced by momentum change of the stream. Therefore,

$$T = \rho A v_i v_s \tag{2.18}$$

where v_s (m/s) is the stream velocity.

From Bernoulli Equation, the energy change per unit time of stream can be written as

$$\Delta \dot{E} = \frac{1}{2}\dot{m} v_s^2 \tag{2.19}$$

Energy change per unit time describes work done per unit time. Therefore, the work done per unit time of the thrust is

$$\dot{W} = T v_i = \Delta \dot{E} \tag{2.20}$$

Here from following equations are obtained

$$T v_i = \rho A v_i^2 v_s = \frac{1}{2}\rho A v_i v_s^2 \tag{2.21}$$

$$v_s = 2 v_i \tag{2.22}$$

Then the thrust becomes

$$T = 2\rho A v_i^2 \tag{2.23}$$

Rewriting the power requirement as [11]

$$P = \frac{T^{3/2}}{\sqrt{2\rho A}} \tag{2.24}$$

The power requirement of the motor should be equal to the power requirement of hover.

$$P = \frac{\tau \omega K_e}{K_t} = \frac{T^{3/2}}{\sqrt{2\rho A}} \tag{2.25}$$

Thus,

$$T = \Big(\frac{K_e r\sqrt{2\rho A}}{K_t}\omega\Big)^2 = \Big(\frac{K_e r\sqrt{2\rho A}}{K_t}\Big)^2(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \tag{2.26}$$

where r (mm) is the length of propeller from its center to the end.

Drag Force Effect on Thrust In accordance with the magnitude of thrust and size of the quadrotor, the size of the propeller and its rotational speed changes. Propellers generate thrust by rotational motion. Thrust is affected by drag. Thrust and drag effect relation is as follows [12]:

$$T = C_d \frac{1}{2}\rho A_p v_i^2 \tag{2.27}$$

where C_d is the drag coefficient and A_p is the propeller cross-section area.

Torques on Roll, Pitch, and Yaw Motion The rolling moment is computed by taking into account the moments of the second and fourth rotor. The moment of the first and third rotors is zero because of the fact that these rotors are on the *x-axis*

$$\tau_x = dT_2 - dT_4 \tag{2.28}$$

where d (mm) is the moment arm, T_2 and T_4 are thrust generated by second and fourth rotors.

The pitch moment is the moment on the *y-axis* and computed by taking into account the first and third rotor moments. Therefore,

$$\tau_y = dT_1 - dT_3 \tag{2.29}$$

To compute *z-axis* torque, unbalance between rotors is used. However, this time total thrust is multiplied by the drag force coefficient.

$$\tau_z = C_d(-T_1 + T_2 - T_3 + T_4) \tag{2.30}$$

where C_d is the drag coefficient mentioned in previous section [13].

T_1, T_2, T_3, T_4 are the thrust of each rotor. Therefore, the total thrust is computed as:

$$\sum_{i=1}^{4} T_i \tag{2.31}$$

2.2.3.1 Transitional Dynamics

Transitional motion equations are described by Newton's Second Law [13]. The equations for *x, y,* and *z* directions are as follows [14]:

$$\ddot{x} = \frac{1}{m}[-k_x\dot{x} + T(\cos\phi\sin\theta c\cos\psi + \sin\phi sin\psi)] \tag{2.32}$$

$$\ddot{y} = \frac{1}{m}[-k_y\dot{y} + T(\cos\phi\sin\theta\sin\psi - \sin\phi cos\psi)] \tag{2.33}$$

$$\ddot{z} = \frac{1}{m}[-k_z\dot{z} - mg + T(\cos\phi\cos\theta)] \tag{2.34}$$

where m (kg) is mass of the quadrotor and k_x, k_y, and k_z are friction coefficients. In this study, the friction force will be neglected.

These equations can be represented in the form of a matrix as

$$\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = \frac{1}{m}\left(\begin{bmatrix}-k_x\dot{x}\\ -k_y\dot{y}\\ -k_z\dot{z}\end{bmatrix} + \begin{bmatrix}0\\ 0\\ -mg\end{bmatrix}\right.$$
$$\left. + \begin{bmatrix}\cos\theta\cos\psi & \sin\phi sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi\\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi\\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta\end{bmatrix}\begin{bmatrix}0\\ 0\\ T\end{bmatrix}\right) \tag{2.35}$$

2.2.3.2 Rotational Dynamics

Rotational dynamic equations are described by the Euler's equation that gives total torque:

$$J\dot{\omega} + \omega \times (J\omega) \tag{2.36}$$

J is the inertia matrix.

$$\begin{bmatrix}J_x & 0 & 0\\ 0 & J_y & 0\\ 0 & 0 & J_z\end{bmatrix} \tag{2.37}$$

Inertia products are neglected because of the quadrotor symmetry. There are external moments acting on the quadrotor body which are gyroscopic and wind.

$$M = (\tau_x + \tau_y + \tau_z) - g_a + \tau_w \tag{2.38}$$

M represents total moments in body, g_a is the gyroscopic moment, and τ_w is the wind torque. According to [15] moment of inertia of rotor is found to be small, so that the gyroscopic moment is not taken into account for system modeling. Also, the wind effect is neglected for this study. Therefore, rotational dynamic equations become [15]:

$$\ddot{p} = \frac{1}{J_x}[\tau_x + \dot{q}\dot{r}(J_y - J_z)] \tag{2.39}$$

$$\ddot{q} = \frac{1}{J_y}[\tau_y + \dot{p}\dot{r}(J_x - J_z)] \tag{2.40}$$

$$\ddot{r} = \frac{1}{J_z}[\tau_z + \dot{p}\dot{q}(J_x - J_y)] \tag{2.41}$$

2.2.4 State-Space Model

A state-space model describes the behavior of a physical system mathematically. State variables, inputs, and outputs are related to each other and describe by differential equations. There are twelve state variables to describe the quadrotor dynamics [13].

$$[x\ \dot{x}\ y\ \dot{y}\ z\ \dot{z}\ \phi\ \dot{\phi}\ \theta\ \dot{\theta}\ \psi\ \dot{\psi}]^T \tag{2.42}$$

The complete dynamic equations of the system is as follows:

$$\dot{x} = \dot{z_b}[\sin\phi\sin\psi{+}\cos\phi\cos\psi\sin\theta]{-}\dot{y_b}[\cos\phi\sin\psi{-}\cos\psi\sin\phi\sin\theta]{+}\dot{x_b}[|\cos\psi\cos\theta]$$

$$\ddot{x} = \frac{1}{m}(\cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi)T$$

$$\dot{y} = \dot{y_b}[\cos\phi\cos\psi{+}\sin\phi\sin\psi\sin\theta]{-}\dot{z_b}[\cos\psi\sin\phi{-}\cos\phi\sin\psi\sin\theta]{+}\dot{x_b}[\cos\theta\sin\psi]$$

$$\ddot{y} = \frac{1}{m}(\sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi)T$$

$$\dot{z} = \dot{z_b}[\cos\phi\cos\theta] - \dot{x_b}[\sin\theta] + \dot{y_b}[\cos\theta\sin\phi]$$

$$\ddot{z} = -g + \frac{1}{m}(\cos\theta\cos\phi)T$$

$$\dot{\phi} = \dot{p} + \dot{r}[\cos\phi\tan\theta] + \dot{q}[\sin\phi\tan\theta]$$

$$\ddot{p} = \frac{1}{J_x}[\tau_x + \dot{q}\dot{r}(J_y - J_z)]$$

$$\dot{\theta} = \dot{q}[\cos\phi] - \dot{r}[\sin\phi]$$

$$\ddot{q} = \frac{1}{J_y}[\tau_y + \dot{p}\dot{r}(J_x - J_z)]$$

$$\dot{\psi} = \dot{r}\frac{\cos\phi}{\cos\theta} + \dot{q}\frac{\sin\phi}{\cos\theta}$$

$$\ddot{r} = \frac{1}{J_z}[\tau_z + \dot{p}\dot{q}(J_x - J_y)] \tag{2.43}$$

According to the dynamic equations, the whole system can be represented in the form of state-space.

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{2.44}$$

$$y(t) = Cx(t) + Du(t) \tag{2.45}$$

where A is the system matrix, B is the input matrix, C is the output matrix, and D is the feed forward matrix. D matrix is accepted as zero matrix. $x(t)$ represents state vector and $u(t)$ represents control input vector.

In this study, the control input vector is determined as:

$$[\phi \ \theta \ T \ \tau_x \ \tau_y \ \tau_z]^T \tag{2.46}$$

2.2.5 Linearization

Linearization procedure is applied around an equilibrium point for fixed constant input. The dynamic equations of the whole system written in the previous section are nonlinear. Finding the solution of the algebraic system becomes difficult because of the trigonometric functions in the dynamic equations. Hence, the mathematical model should be linearized by simplification. The simplification is made by small-angle assumptions. According to the small-angle assumption, the sine function is equal to its argument, and the cosine function is equal to one [15] and also the thrust is almost equal to the weight of the quadrotor when it is in hovering position. Consequently, the linear equations can be written as follows [13]:

$$
\begin{aligned}
\dot{x} &= \dot{x}_b \\
\ddot{x} &= g\theta \\
\dot{y} &= \dot{y}_b \\
\ddot{y} &= -g\theta \\
\dot{z} &= \dot{z}_b \\
\dot{\phi} &= \dot{p} \\
\ddot{p} &= \frac{1}{J_x}\tau_x \\
\dot{\theta} &= \dot{q} \\
\ddot{q} &= \frac{1}{J_y}\tau_y \\
\dot{\psi} &= \dot{r} \\
\ddot{r} &= \frac{1}{J_z}\tau_z
\end{aligned}
\tag{2.47}
$$

The system equations can be written in state-space matrix form as following:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{2.48}$$

$$\begin{bmatrix}\dot{x}\\ \ddot{x}\\ \dot{y}\\ \ddot{y}\\ \dot{z}\\ \ddot{z}\\ \dot{\phi}\\ \ddot{\phi}\\ \dot{\theta}\\ \ddot{\theta}\\ \dot{\psi}\\ \ddot{\psi}\end{bmatrix} = \begin{bmatrix}0&1&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&1&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0&0&0&0&0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ y\\ \dot{y}\\ z\\ \dot{z}\\ \phi\\ \dot{\phi}\\ \theta\\ \dot{\theta}\\ \psi\\ \dot{\psi}\end{bmatrix} + \begin{bmatrix}0&0&0&0&0&0\\g&0&0&0&0&0\\0&0&0&0&0&0\\0&-g&0&0&0&0\\0&0&0&0&0&0\\0&0&g-\frac{1}{m}&0&0&0\\0&0&0&0&0&0\\0&0&0&\frac{1}{J_x}&0&0\\0&0&0&0&0&0\\0&0&0&0&\frac{1}{J_y}&0\\0&0&0&0&0&0\\0&0&0&0&0&\frac{1}{J_z}\end{bmatrix}\begin{bmatrix}\phi\\ \theta\\ T\\ \tau_x\\ \tau_y\\ \tau_z\end{bmatrix} \tag{2.49}$$

$$y(t) = C(t) \tag{2.50}$$

$$\begin{bmatrix}x\\ y\\ z\\ \phi\\ \theta\\ \psi\end{bmatrix} = \begin{bmatrix}1&0&0&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0&0&0&0\\0&0&0&0&0&0&1&0&0&0&0&0\\0&0&0&0&0&0&0&0&1&0&0&0\\0&0&0&0&0&0&0&0&0&0&1&0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ y\\ \dot{y}\\ z\\ \dot{z}\\ \phi\\ \dot{\phi}\\ \theta\\ \dot{\theta}\\ \psi\\ \dot{\psi}\end{bmatrix} \tag{2.51}$$

2.3 Controller Design

2.3.1 Linear Quadratic Optimal Control

Linear quadratic optimal control is a general name of optimal control problems for linear systems. The optimal control includes state, input, output, and cost function, in other words, performance index. The aim of optimal control is to minimize the cost function [16]. The cost function is in quadratic form and it is given as:

$$J = \int_0^\infty (x^T Q x + u^T R u) dt \tag{2.52}$$

Where Q is the state weighting matrix and R is the control weighting matrix. $u(t)$ is the control vector and found as follows:

$$u(t) = -Kx(t) \tag{2.53}$$

The optimal controller is a closed-loop system with full-state feedback and determines the gain matrix denoted by K. The optimal gain K is computed through:

$$K = R^{-1} B^T P \tag{2.54}$$

P is the optimal matrix and it is positive-definite. The P matrix is found from algebraic Riccati Equation [17]:

$$A^T P + PA - PBR^{-1}B^T P + Q = 0 \tag{2.55}$$

The closed-loop system is asymptotically stable with the feedback gain matrix K [18].

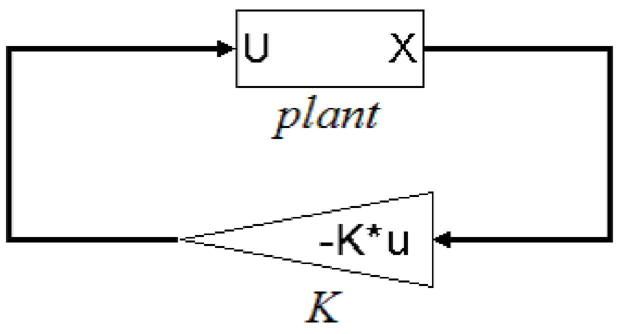


Figure 2.3: Closed Loop LQR Controller System.

2.3.2 Optimal Control with Estimator

The optimal control technique is one of the control methods that can be applied for UAVs. There are twelve states determined for the quadrotor. Some of them cannot be measured or measurements of the states can be noisy. Hence, a filtering method named Kalman filter is applied with the optimal controller. Kalman filter estimates the unmeasurable states. Kalman filter uses state equations like LQR controller. First, calculates gain by using initial states, then, estimates the states of the system for each sampling period. The estimation process is done by a discretized linear state-space model.

The Kalman filter described as follows:

$$x(k+1) = Ax(k) + Bu(k) + Gw(k) \tag{2.56}$$

$$y(k) = Cx(k) + v(k) \tag{2.57}$$

$x(k)$, $u(k)$, and $y(k)$ vectors represent state, control input and output of the system respectively. A is the system matrix, B is the control input matrix, and C is the output matrix. $w(k)$ is the noise effecting the states and accepted as Gaussian noise with zero mean. $v(k)$ is the noise caused by measurements with zero mean. Both $w(k)$ and $v(k)$vectors are known by their covariance structure. There is no correlation between the noise vectors [19].

The covariance matrices for the noise vectors are given by [20]:

$$E[w(k)^T w(k)] = Q \tag{2.58}$$

$$E[v(k)^T v(k)] = R \tag{2.59}$$

The Kalman filter equation to obtain optimal states and outputs is given by [2]:

$$\hat{x}(k+1|k) = A\hat{x}(k-1|k) + Bu(k|k-1) + K\big(y(k) - \hat{y}(k|k-1)\big) \tag{2.60}$$

$$\hat{y}(k|k-1) = C\hat{x}(k|k-1) \tag{2.61}$$

where K is the Kalman gain and it is computed by solving algebraic Riccati equation.

$$K(k) = P(k)C^T R^{-1}(k) \tag{2.62}$$

$P(k)$ represents the covariance matrix and R represents the weight matrix [20].

2.3.3 Linear Model Predictive Control

Model predictive control (MPC) is the most popular research topic in recent years. Predictive control is a kind of feedback control. MPC solves the quadratic optimal control problem at each sampling period and predicts the future dynamic behavior of the system with finite or infinite horizon [21].

Receding horizon control (RHC) is a widely used method for MPC. In accordance with the RHC, the optimization problem is solved for each time step and future inputs and states are determined over a fixed time horizon. To determine the future behavior of the system, RHC uses initial inputs. For the next time step, the process is repeated and a new optimization problem is solved by shifting the time horizon one step forward. The estimation process depends on measurable data and optimization of the system. This process provides feedback [22].

In this study, state feedback receding horizon control method was used.

The general form of the predictive representation is given by [2]:

$$\hat{x}(k+j|k) = A^{j-1}\hat{x}(k+1|k) + \sum_{n=1}^{j-1} A^{j-n-1}Bu(k+n|k) \tag{2.63}$$

$$\hat{y}(k+j|k) = CAj - 1\hat{x}(k+1|) + C\hat{x}(k+j|k) \tag{2.64}$$

where k represents current time, j represents time distance. In this manner, the state feedback RHC is in the form of tracking. According to the terminal equality constraint, the cost function of RHC for tracking case is written as:

$$J\big(\hat{x}(k), x^r, u(k+j|k)\big) = \sum_{j=0}^{N-1} [\hat{x}(k+j|k) - x^r(k+j|k)]^T Q[\hat{x}(k+j|k) - x^r(k+j|k)] + [u(k+j|k)^T Ru(k+j|k)] \tag{2.65}$$

where N represents horizon size.

Then the Riccati equation is solved by forward computation and control input of the system is obtained.

$$u(k) = -R^{-1}B^T P_N^{-1} Ax(k) \tag{2.66}$$

P_N is computed by the following equation:

$$P(j+1) = A^{-1}P(j)[I + A^{-T}QA^{-1}P(j)]^{-1}A^{-T} + BR^{-1}B^T \tag{2.67}$$

According to [23] assume A $P(1) = 0$.

Chapter 3

Approach

In this study, there are six different models. Each model was modeled by using Matlab/Simulink and observed individually. The reference trajectory was determined as lifting the quadrotor with a ramp, then going straight along the x-axis and returning down to the ground with a ramp. Whether the system tracks the given trajectory or not was observed. Then the results of each model was compared. Also, the development was preferred as step by step instead of direct application.

First, the transitional movement on the x and y axes, vertical take-off, and the rotational movement were observed separately modeled by the linear state-space model. The linear state-space model for the transitional movement on the x and y axes is given as:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{y} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ -g & 0 \\ 0 & 0 \\ 0 & g \end{bmatrix} \begin{bmatrix} \phi \\ \theta \end{bmatrix} \tag{3.1}$$

The vertical take-off of the quadrotor was observed by constituting another linear state-space model.

$$\begin{bmatrix} \dot{z} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} z \\ \dot{z} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} g - \frac{T}{m} \tag{3.2}$$

The other state-space model was constituted for observing the angular position of the quadrotor.

$$\begin{bmatrix} \dot{\phi} \\ \ddot{\phi} \\ \dot{\theta} \\ \ddot{\theta} \\ \dot{\psi} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \phi \\ \dot{\phi} \\ \theta \\ \dot{\theta} \\ \psi \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ \frac{1}{J_x} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \frac{1}{J_y} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{1}{J_z} \end{bmatrix} \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} \tag{3.3}$$

Second, the system was modeled by combining all transitional positions in one state-space model. The rotational movement of the quadrotor was observed from another state-space model like in the first model. Third, the system was modeled by combining all states. Then, by discretizing the third model, the fourth model was obtained. So far for four models, the system is closed-loop full state feedback. Therefore, the outputs of all states can be observed. The linear-quadratic optimal control method was applied to the models. The optimal gain matrices were computed by using lqr(A,B,Q,R) command in Matlab for each model.

In the fifth model, an estimator was applied to the discrete-time system. Matlab function block was used as a Kalman filter. Kalman filter algorithm was written in the Matlab function block. The output states were estimated iteratively by using for loops in the algorithm.

In the sixth model, the discrete-time system was modeled with the model predictive controller by using the RHC method. Matlab function block was used as controller. In the Matlab function block, the Kalman filter algorithm was used and the predictive part was added to the algorithm. The predicted states were obtained iteratively. The Riccati equation was solved in for loop algorithm. Thus, the optimal inputs could be obtained.

Chapter 4

Results and Discussion

In this study, more than one Simulink model were analyzed and the output states were observed individually for each model.

In Model 1, the system was modeled in three parts which are x-y position, z-position, and angular position. This model is in continuous time. The simulation run time was set as 100 seconds. The controller of the system is LQR. References were given for the states x, z, and phi. According to the output results of the Model 1, the system tracks the desired trajectory well. The LQR controller performs optimally and the system is stable.

The Simulink model and the results of the system are given in the following figures:

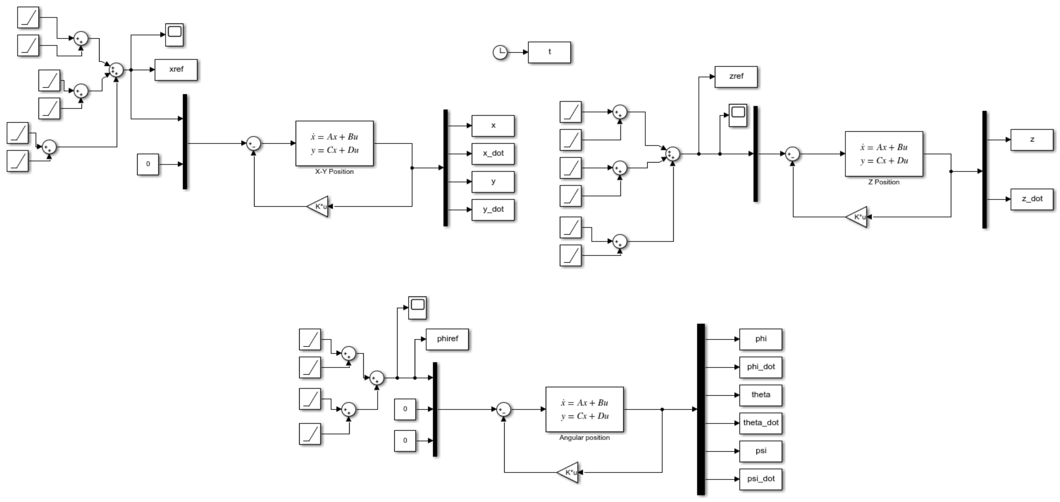


Figure 4.1: Model 1.

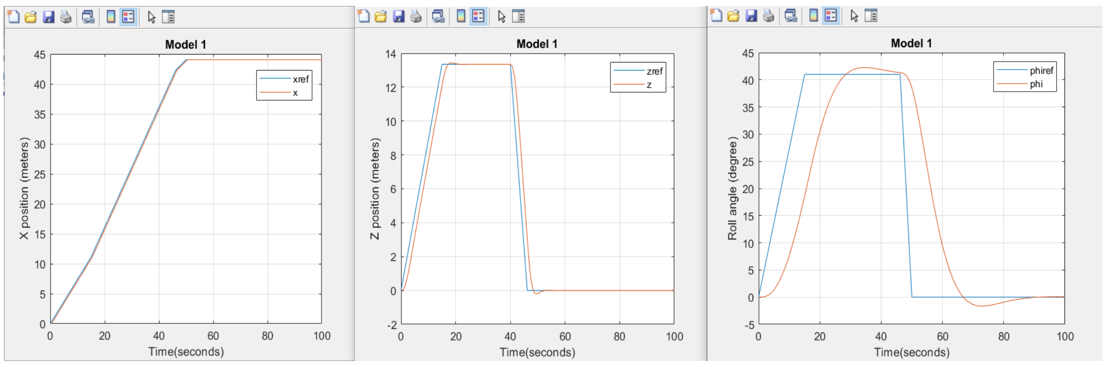


Figure 4.2: Results of Model 1.

In Model 2, all transitional positions were combined in one part and the angular positions were the other part of the model. This model is in continuous time. The simulation run time was set as 100 seconds. The controller of the system is LQR. References were given for the states x, z, and phi (ϕ). According to the output results of the Model 2, the system tracks the desired trajectory well. The LQR controller performs optimally and the system is stable.

The Simulink model and the results of the system are given in the following figures:

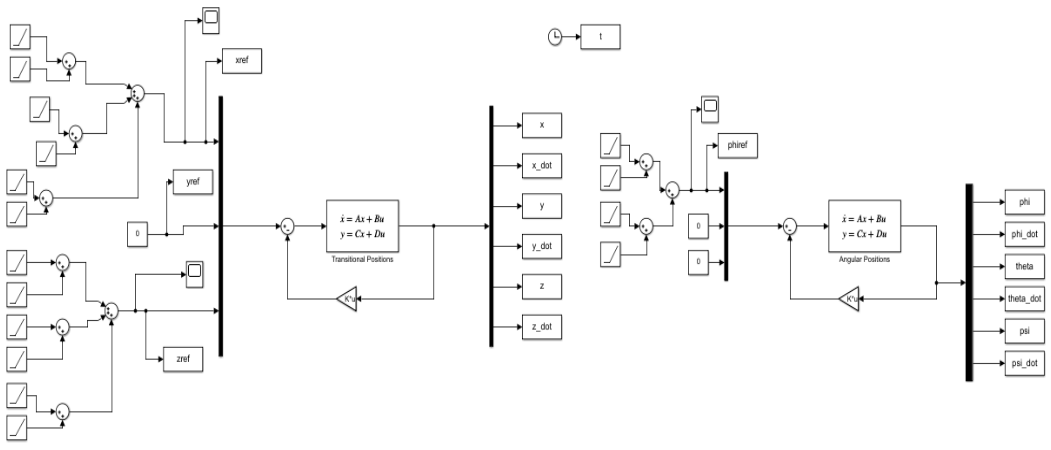


Figure 4.3: Model 2.

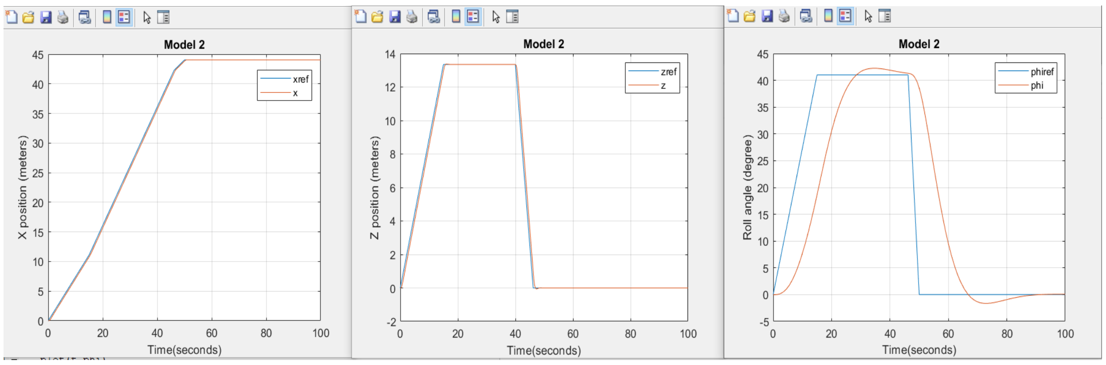


Figure 4.4: Results of Model 2.

In Model 3, the system was modeled by combining all states. This model is in continuous time. The simulation run time was set as 100 seconds. The controller of the system is LQR. References were given for the states x, z, and phi. According to the output results of the Model 3, the system tracks the desired trajectory well. The LQR controller performs optimally and the system is stable. In addition, we can say that the Model 3 is more robust than the Model 1 and the Model 2.

The Simulink model and the results of the system are given in the following figures:

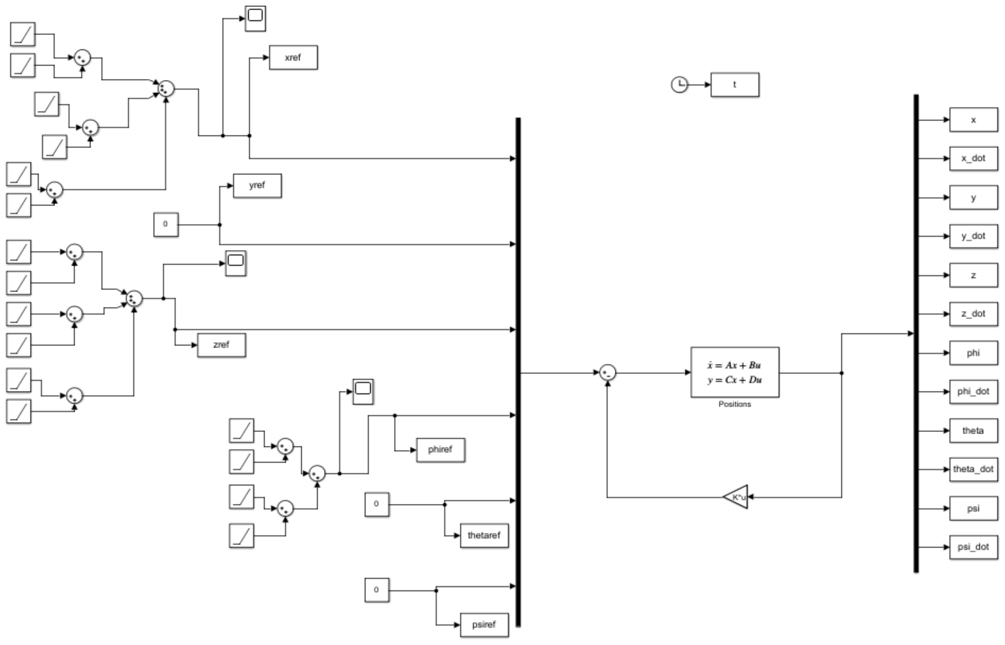


Figure 4.5: Model 3.

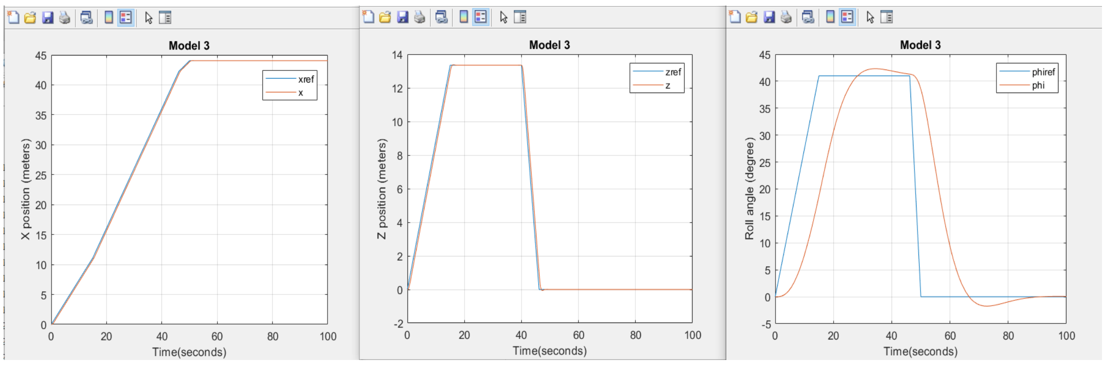


Figure 4.6: Results of Model 3.

In Model 4, the system is in discrete-time. The sampling period was set at 0.1 seconds. A and B matrices were discretized. The simulation run time was set as 100 seconds. The controller of the system is LQR. References were given for the states x, z, and phi. According to the output results of the Model 4, the system tracks the desired trajectory well. The LQR controller performs optimally and the system is stable. There is a little scaling factor between the reference and output lines. This scaling factor is caused by discretization. Also, the output line is smoother than the reference line with sharp corners.

The Simulink model and the results of the system are given in the following figures:

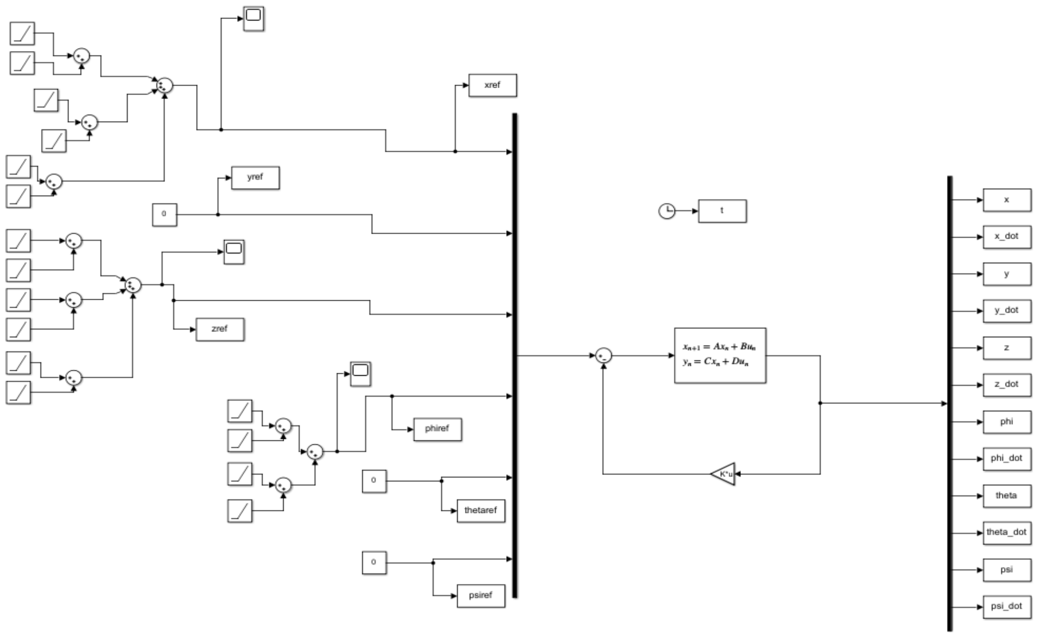


Figure 4.7: Model 4.

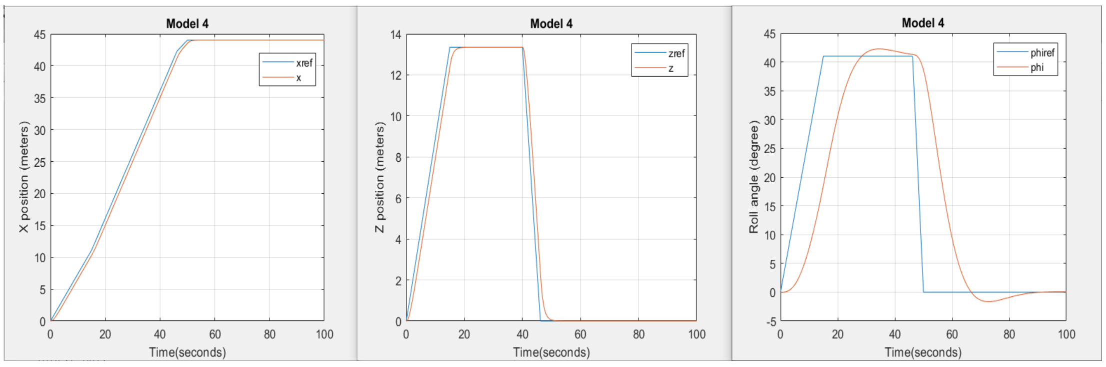


Figure 4.8: Results of Model 4.

Model 5 was designed by adding an estimator to the LQR controller. This system is in discrete-time. The sampling period was set at 0.1 seconds. . The simulation run time was set as 100 seconds. References were given for the states x, z, and phi. According to the output results of the Model 5, the Kalman filter estimates the states correctly. The optimal Kalman gain values were found the same as the LQR gain values after many trials. The system tracks the given trajectory perfectly. Using the Kalman estimator with the optimal controller got better than the first four models.

The Simulink model and the results of the system are given in the following figures:

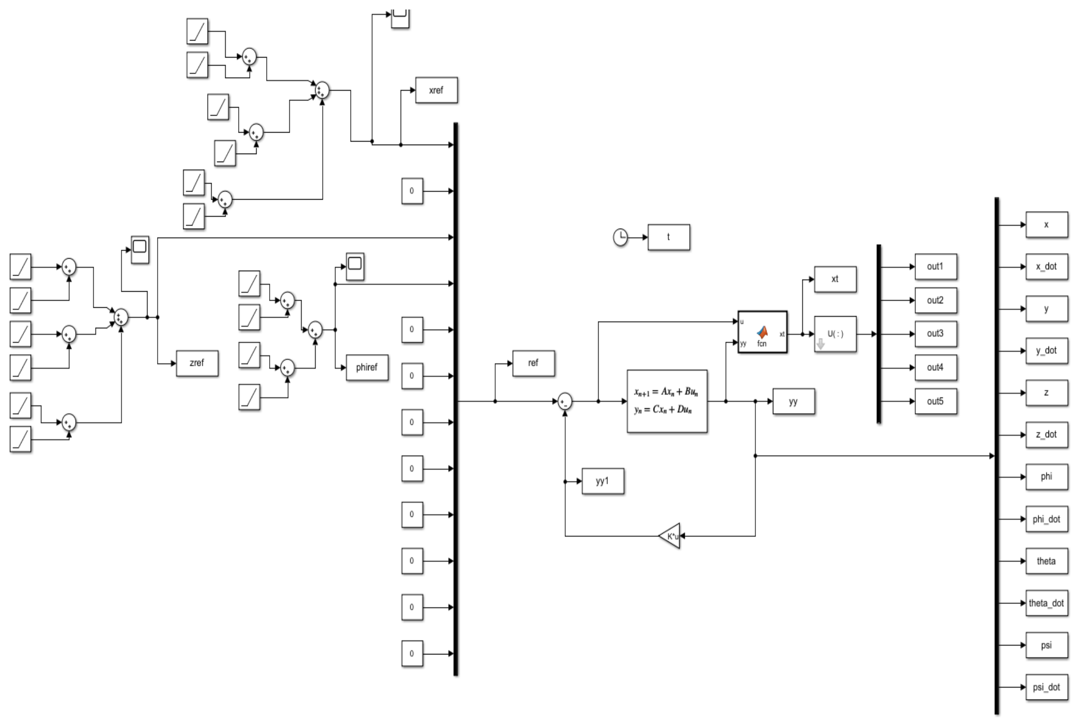


Figure 4.9: Model 5.

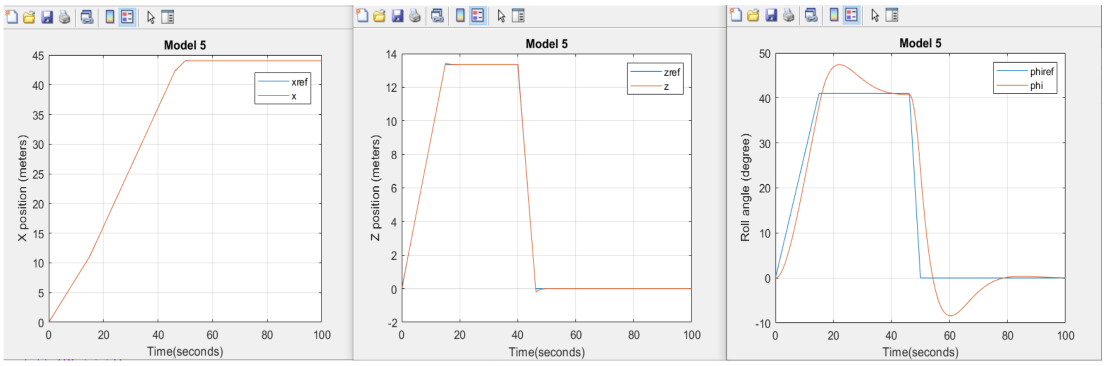


Figure 4.10: Results of Model 5.

In Model 6, the system was designed with the model predictive controller. By comparing the sixth model and the other models, there is a major difference. The controller converges to the reference values exponentially. The MPC output curve stopped by changing the Simulink run time from 100 seconds to 70 seconds. Only the result of state x has been observed for the MPC controller. Because the MPC controller needs more optimization. The MPC controller has not been working well for other states. Nevertheless, we can say that the system is stable.

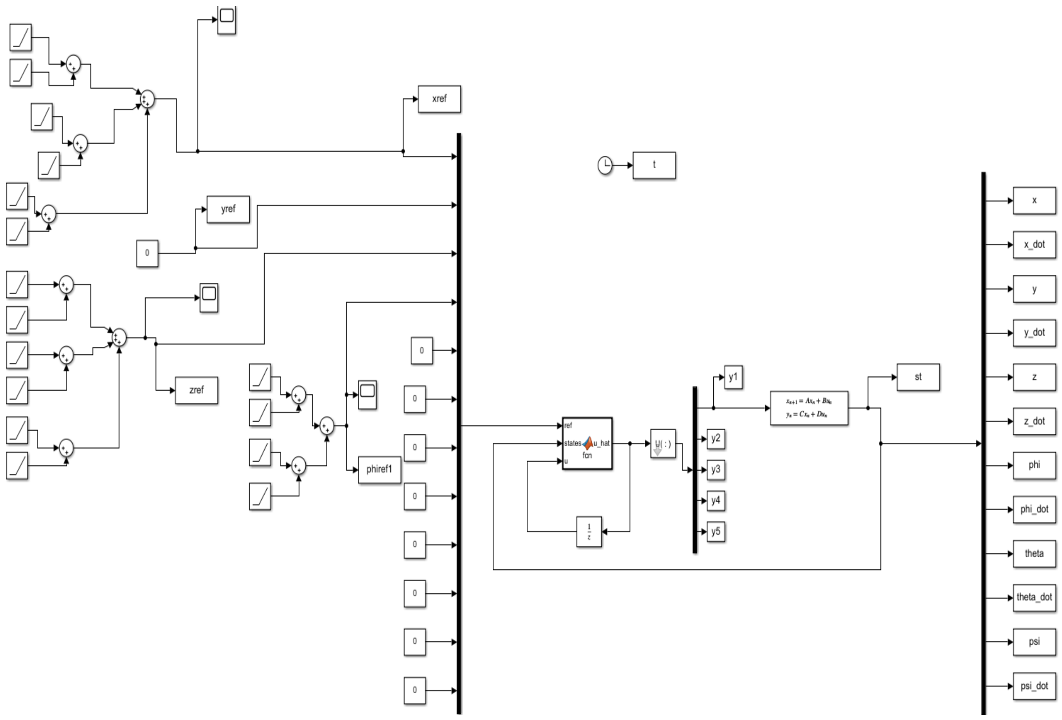


Figure 4.11: Model 6.

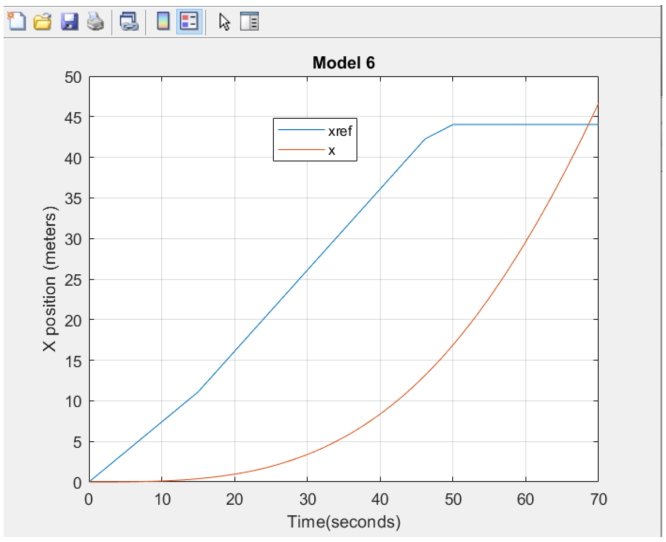


Figure 4.12: Results of Model 6.

As a result, the results of the states and the controller performances were observed for each model. We see that the controllers work well for five models, especially when the system is not under any disturbance. For five models, while the states x and z track the given reference perfectly, the phi () angle tracks smoothly the given angle changes with a minor time delay. For Model 6, we can say that the controller works but the results are not fully satisfactory. As a future study, the controller can be developed to work well as could as previous models.

Chapter 5

Conclusion

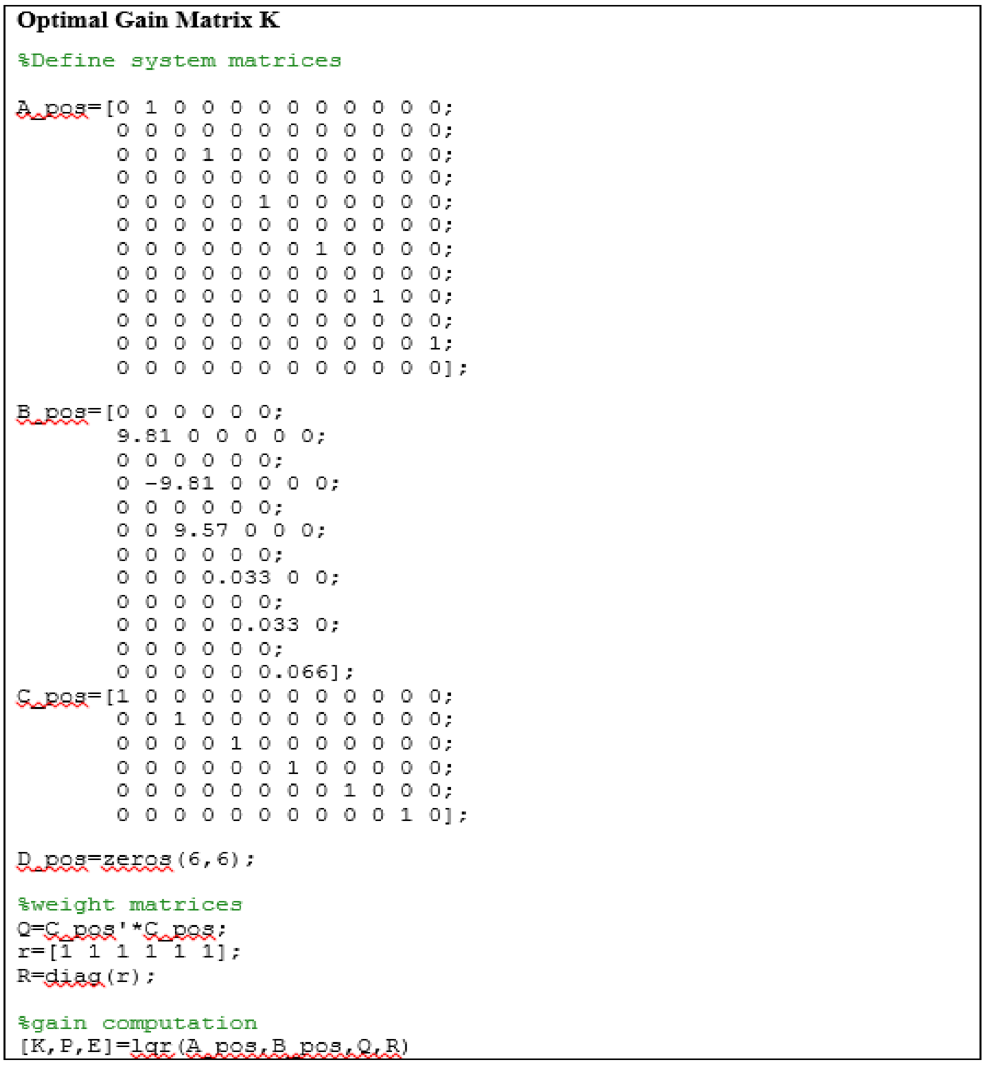
In this thesis project, controllers' performances and different system designs have been examined then, the output results have been compared. At the beginning of the system design, states have been observed by dividing into three groups and gathering them. In compliance with both cases, in continuous time, the LQR controller has the ability to trajectory tracking. After the continuous-time system has been discretized, the LQR controller performed well again. Nevertheless, a slight difference caused by discrete-time coefficients has been observed. With these coefficients, a smoother graphic has been obtained, as if through a low-pass filter. Then, the Kalman estimator has been applied to the system. According to the Kalman estimator results, the outputs have been obtained almost exactly the same as the reference.

Finally, the MPC performance has been observed by using the receding horizon control method. Considering the optimal controller and the Kalman estimator, the MPC controller has achieved to approach the reference values by converging exponentially. The MPC performance has been observed for only the state x. Because for other states, it was not exactly accurate. Hence, the MPC controller can be optimized for future projects.

The RHC is a kind of repetitive LQR control with a fixed period. Therefore, the proceeding has been determined as step by step development.

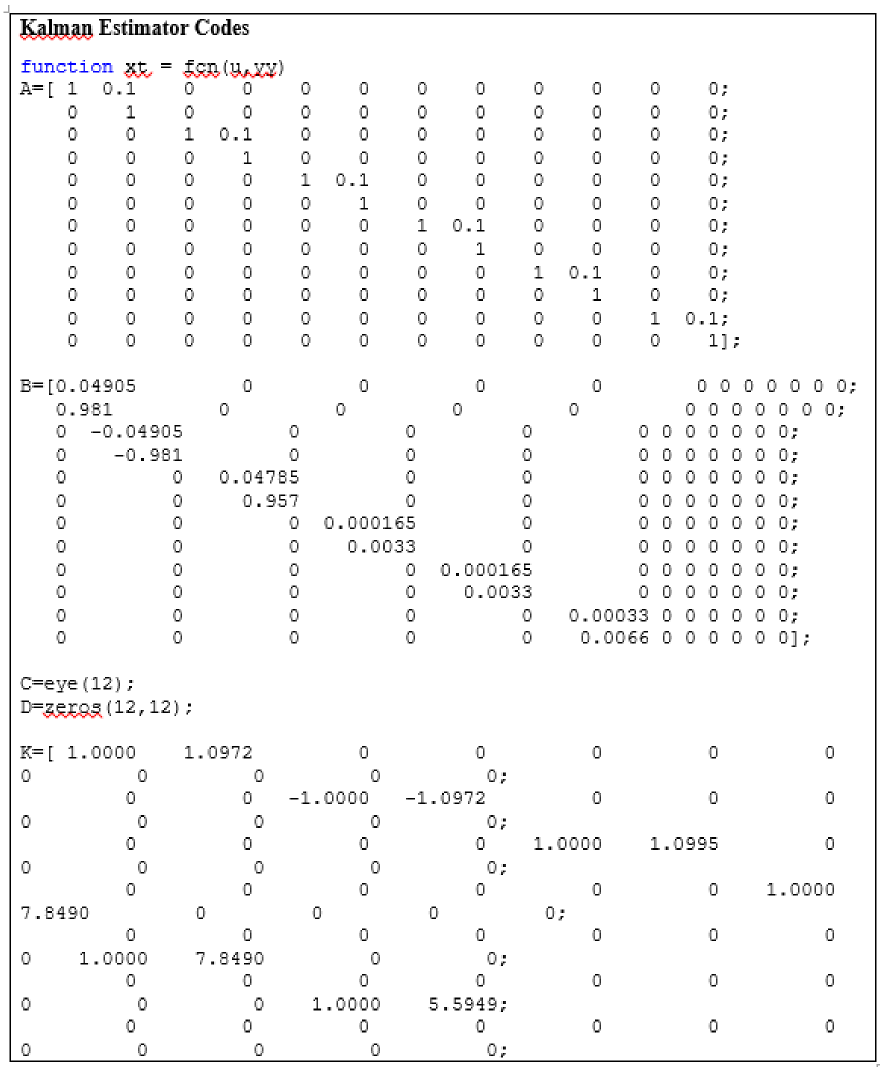
Chapter 6

APPENDİX



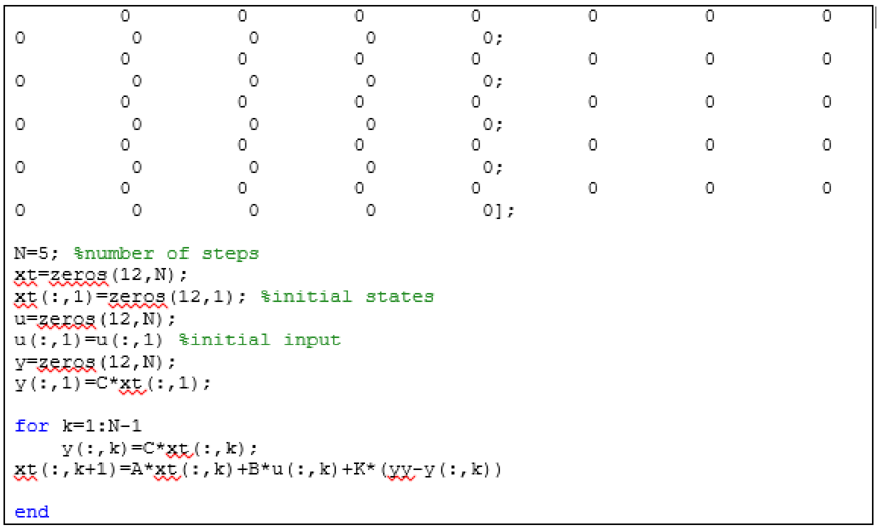
Optimal Gain Matrix K

```
%Define system matrices

A_pos=[0 1 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 1 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 1 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 1 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 1 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 1;
       0 0 0 0 0 0 0 0 0 0 0 0];

B_pos=[0 0 0 0 0 0;
       9.81 0 0 0 0 0;
       0 0 0 0 0 0;
       0 -9.81 0 0 0 0;
       0 0 0 0 0 0;
       0 0 9.57 0 0 0;
       0 0 0 0 0 0;
       0 0 0 0.033 0 0;
       0 0 0 0 0 0;
       0 0 0 0 0.033 0;
       0 0 0 0 0 0;
       0 0 0 0 0 0.066];
C_pos=[1 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 0 1 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 0 1 0 0 0;
       0 0 0 0 0 0 0 0 0 0 1 0];

D_pos=zeros(6,6);

%weight matrices
Q=C_pos'*C_pos;
r=[1 1 1 1 1 1];
R=diag(r);

%gain computation
[K,P,E]=lqr(A_pos,B_pos,Q,R)
```

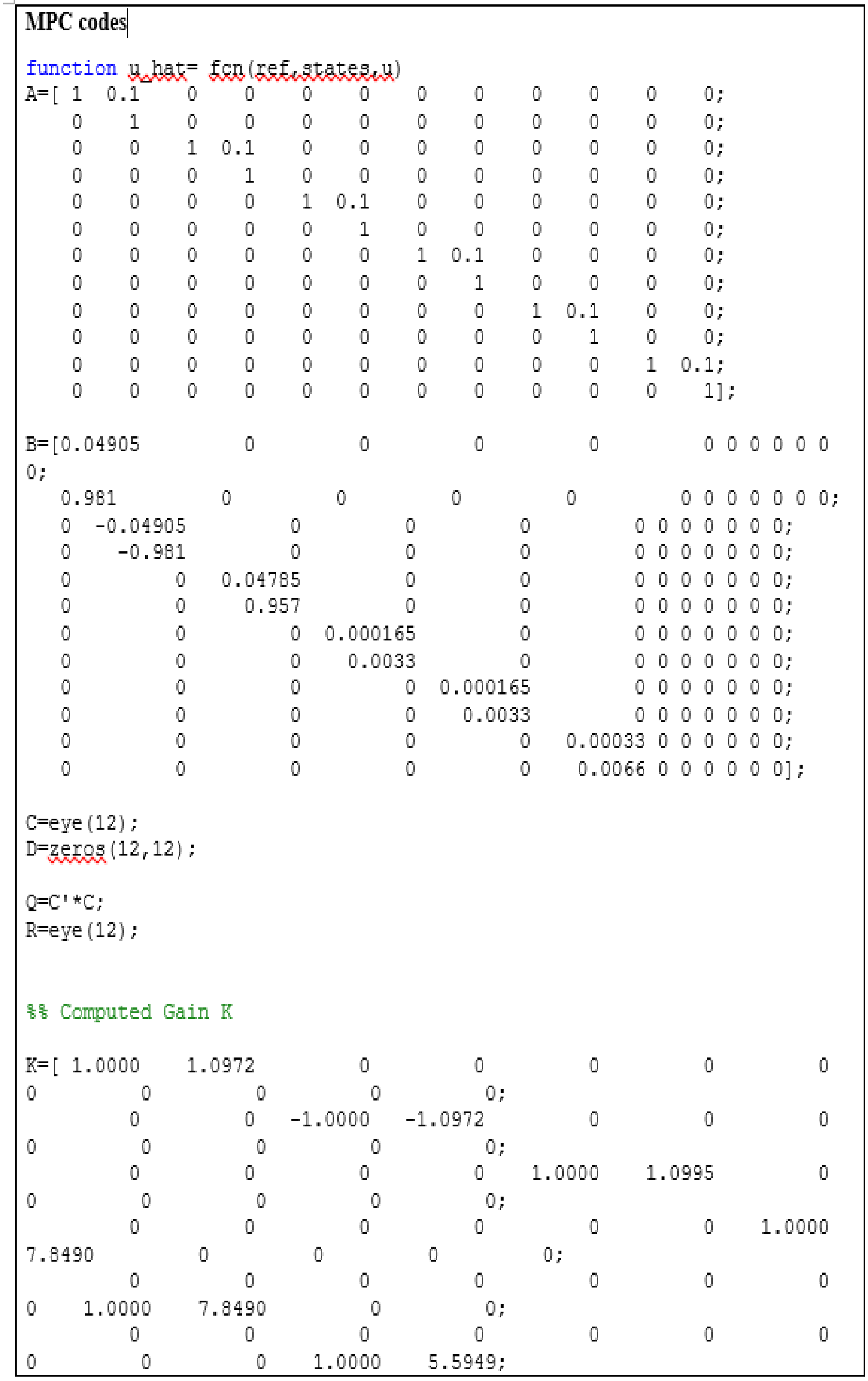


Kalman Estimator Codes

```
function xt = fcn(u,yy)
A=[ 1  0.1    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  0.1    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  0.1    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  0.1    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  0.1    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  0.1;
    0    0    0    0    0    0    0    0    0    0    0    1];

B=[0.04905         0         0         0         0        0 0 0 0 0 0 0;
   0.981         0         0         0         0        0 0 0 0 0 0 0;
   0  -0.04905         0         0         0        0 0 0 0 0 0 0;
   0    -0.981         0         0         0        0 0 0 0 0 0 0;
   0         0   0.04785         0         0        0 0 0 0 0 0 0;
   0         0     0.957         0         0        0 0 0 0 0 0 0;
   0         0         0  0.000165         0        0 0 0 0 0 0 0;
   0         0         0    0.0033         0        0 0 0 0 0 0 0;
   0         0         0         0  0.000165        0 0 0 0 0 0 0;
   0         0         0         0    0.0033        0 0 0 0 0 0 0;
   0         0         0         0         0  0.00033 0 0 0 0 0 0;
   0         0         0         0         0   0.0066 0 0 0 0 0 0];

C=eye(12);
D=zeros(12,12);

K=[ 1.0000    1.0972         0         0         0         0         0
0         0         0         0         0;
         0         0   -1.0000   -1.0972         0         0         0
0         0         0         0         0;
         0         0         0         0    1.0000    1.0995         0
0         0         0         0         0;
         0         0         0         0         0         0    1.0000
7.8490         0         0         0         0;
         0         0         0         0         0         0         0
0    1.0000    7.8490         0         0;
         0         0         0         0         0         0         0
0         0         0    1.0000    5.5949;
         0         0         0         0         0         0         0
0         0         0         0         0;
```

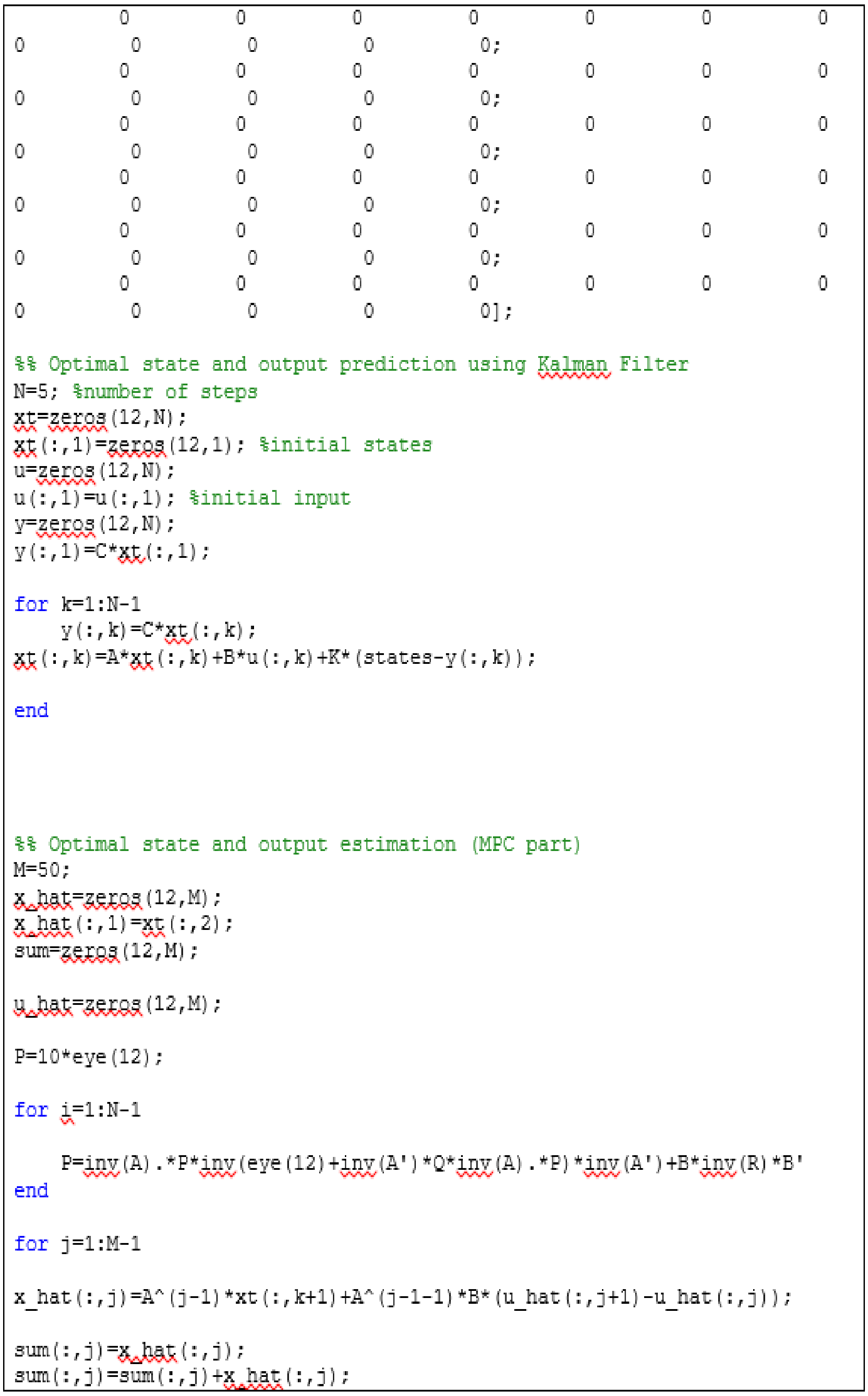


```
         0         0         0         0         0         0         0
0         0         0         0         0;
         0         0         0         0         0         0         0
0         0         0         0         0;
         0         0         0         0         0         0         0
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];

N=5; %number of steps
xt=zeros(12,N);
xt(:,1)=zeros(12,1); %initial states
u=zeros(12,N);
u(:,1)=u(:,1) %initial input
y=zeros(12,N);
y(:,1)=C*xt(:,1);

for k=1:N-1
    y(:,k)=C*xt(:,k);
xt(:,k+1)=A*xt(:,k)+B*u(:,k)+K*(yy-y(:,k))

end
```



MPC codes

```
function u_hat= fcn(ref,states,u)
A=[ 1  0.1    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   0.1     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   0.1     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   0.1     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   0.1     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   0.1;
    0    0    0     0     0     0     0     0     0     0     0     1];

B=[0.04905         0         0         0         0         0 0 0 0 0 0
0;
   0.981         0         0         0         0         0 0 0 0 0 0 0;
   0  -0.04905         0         0         0         0 0 0 0 0 0 0;
   0    -0.981         0         0         0         0 0 0 0 0 0 0;
   0         0   0.04785         0         0         0 0 0 0 0 0 0;
   0         0     0.957         0         0         0 0 0 0 0 0 0;
   0         0         0  0.000165         0         0 0 0 0 0 0 0;
   0         0         0    0.0033         0         0 0 0 0 0 0 0;
   0         0         0         0  0.000165         0 0 0 0 0 0 0;
   0         0         0         0    0.0033         0 0 0 0 0 0 0;
   0         0         0         0         0   0.00033 0 0 0 0 0 0;
   0         0         0         0         0    0.0066 0 0 0 0 0 0];

C=eye(12);
D=zeros(12,12);

Q=C'*C;
R=eye(12);


%% Computed Gain K

K=[ 1.0000    1.0972         0         0         0         0         0
0         0         0         0         0;
         0         0   -1.0000   -1.0972         0         0         0
0         0         0         0         0;
         0         0         0         0    1.0000    1.0995         0
0         0         0         0         0;
         0         0         0         0         0         0    1.0000
7.8490         0         0         0         0;
         0         0         0         0         0         0         0
0    1.0000    7.8490         0         0;
         0         0         0         0         0         0         0
0         0         0    1.0000    5.5949;
```



```
          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
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];

%% Optimal state and output prediction using Kalman Filter
N=5; %number of steps
xt=zeros(12,N);
xt(:,1)=zeros(12,1); %initial states
u=zeros(12,N);
u(:,1)=u(:,1); %initial input
y=zeros(12,N);
y(:,1)=C*xt(:,1);

for k=1:N-1
    y(:,k)=C*xt(:,k);
xt(:,k)=A*xt(:,k)+B*u(:,k)+K*(states-y(:,k));

end




%% Optimal state and output estimation (MPC part)
M=50;
x_hat=zeros(12,M);
x_hat(:,1)=xt(:,2);
sum=zeros(12,M);

u_hat=zeros(12,M);

P=10*eye(12);

for i=1:N-1

    P=inv(A).*P*inv(eye(12)+inv(A')*Q*inv(A).*P)*inv(A')+B*inv(R)*B'
end

for j=1:M-1

x_hat(:,j)=A^(j-1)*xt(:,k+1)+A^(j-1-1)*B*(u_hat(:,j+1)-u_hat(:,j));

sum(:,j)=x_hat(:,j);
sum(:,j)=sum(:,j)+x_hat(:,j);
```

References

[1] C. Liu, J. Pan, and Y. Chang, "Pid and lqr trajectory tracking control for an unmanned quadrotor helicopter: Experimental studies," in *Proceedings of the 35th Chinese Control Conference*, China, Chengdu, July 27-29 2016.

[2] M. Chipofya, D. Lee, and K. Chong, "Trajectory tracking and stabilization of a quadrotor using model predictive control of laguerre functions," *Hindawi Publishing Corperation Abstract and Applied Analysis*, vol. 2015, 2014.

[3] M. Islam, M. Okasha, and M. M. Idres, "Trajectory tracking in quadrotor platform by using pd controller and lqr control approach," in *Proceedings of 6th International Conference on Mechatronics - ICOM'17*, 2017.

[4] R. Lopes, P. Santana, G. Borges, and J. Ishihara, "Model predictive control applied to tracking and attitude stabilization of a vtol quadrotor aircraft," *ABCM Symposium Series in Mechatronics*, vol. 5, p. 176, 2012.

[5] S. Norouzi, Y. Aghli, M. Alimohammadi, and A. Akbari, "Quadrotors unmanned aerial vehicles: A review," *Periodicals of Engineering and Natural Sciences*, vol. 9, no. 1, March,2016.

[6] S. Kurak and M. Hodzig, "Control and estimation of a quadcopter dynamical model," *Periodicals of Engineering and Natural Sciences*, vol. 6, no. 1, pp. 63–75, March,2018.

[7] T. Luukkonen, *Modelling and Control of Quadcopter.* Aalto University, 2011.

[8] E. Kuantama and R. Tarca, "Quadcopter attitude and thrust simulation based on simulink platform," *ResearchGate*, vol. 6, no. 1, December,2015.

[9] R. W. Beard, *Quadrotor Dynamics and Control.* Western New England University, February 19, 2008.

[10] K. Rahnamai, "Quadrotor drones thrust measurement apparatus," in *Western New England University.* Springfield, USA: IEEE, 2016.

[11] "Momentum disk analysis," http://www.aerodynamics4students.com/aircraft-performance/rotor_momentum_analysis.php, accessed:2019-04-20.

[12] A. Al, A. Dewi, and T. Hidayat, "The lifting force relationship of a bldc motor rotor system to the blade rotation speed," *MATEC Web of Conferences*, vol. 215, no. 01013, 2018.

[13] Z. Tahir, M. Jamil, S. Liaqat, L. Mubarak, W. Tahir, and S. Gilani, "State space system modeling of a quad copter uav," *Indian Journal of Science and Technology*, vol. 9, no. 27, July,2016.

[14] M. Islam, M. Okasha, and M. Idres, "Dynamics and control of quadcopter using linear model predictive control approach," *AEROS Conference*, vol. 270, no. 012007, 2017.

[15] F. Sabatino, "Quadrotor control: modeling, nonlinear control design, and simulation," Ph.D. dissertation, KTH Royal Institute of Technology, 2015.

[16] H. Trantelman, "Linear quadratic optimal control," *Encyclopedia of Systems and Control*, vol. 201, no. 3, 2014.

[17] L. Martins, C. Cardeira, and P. Oliveira, "Linear quadratic regulator for trajectory tracking of a quadrotor," in *IFAC PapersOnLine.* ScienceDirect, 2019.

[18] A. Mahturi and H. Wahid, "Optimal tuning of linear quadratic regulator controller using a particle swarm optimization for two-rotor aerodynamical

system," *World Academy of Science, Engineering and Technology International Journal of Electronics and Communication Engineering*, vol. 11, no. 2, 2017.

[19] C. Hajiyev and S. Vural, "Lqr controller with kalman estimator applied to uav longitudinal dynamics," *Scientific Research*, vol. 4, pp. 36–41, 2013.

[20] "Kalman," https://www.mathworks.com/help/control/ref/ss.kalman.html, accessed:2020-06-17.

[21] M. Villanueva, R. Quirynen, M. Diehl, C. B., and B. Houska, "Robust mpc via min–max differential inequalities," *Automatica*, vol. 77, pp. 311–321, 2016.

[22] S. Warthenpfuhl, B. Tibken, and S. Mayer, "An interval arithmetic approach for the estimation of the domain of attraction," in *2010 IEEE International Symposium on Computer-Aided Control System Design*, 2010, pp. 1999–2004.

[23] Kwon, W. Hyun, Han, and S. Hee, *Receding Horizon Control.* Springer, 2005.