



Quadcopter Unmanned Aerial Vehicle: State-Space Modelling, Control Design, and Simulation

Muhammad Abdelsallam 201801356

Mariam Shafik 201-801-585

University of Science and Technology - Zewail City
REE 409 Advanced Control

Dr. Gamal El-Bayoumi

Eng. Somaia Talaat



Table of Contents

Table of Contents	1
Introduction	2
Working Principle	2
System Modeling	3
QuadCopter Dynamics	3
Roll Motion	3
Pitch Motion	4
Yaw Motion	4
Vertical Motion	5
Horizontal Motion	5
State-Space Model	6
Quadcopter Modelling	7
Numerical Values	9
Controllability & Observability	10
Designing a Controller	11
Control System Architecture	11
Linear Quadratic Regulator	11
System Full State Variable	13
System Transfer Function	14
System Stability	15
Simulation	17
Matlab Code	19
References	23
Table of Contents	1
Introduction	2
Working Principle	2
System Modeling	3
QuadCopter Dynamics	3
Roll Motion	3
Pitch Motion	4

Yaw Motion	4
Vertical Motion	5
Horizontal Motion	5
State-Space Model	6
Quadcopter Modelling	7
Numerical Values	9
Controllability & Observability	10
Designing a Controller	11
Control System Architecture	11
Linear Quadratic Regulator	11
System Full State Variable	13
System Transfer Function	14
System Stability	15
Simulation	17
Matlab Code	19
References	23

Introduction

Recently, small unmanned aerial vehicles (UAVs) have witnessed fast improvement in multi-mission proficiency. Their capacity to replace manned aerial vehicles in standard and, most importantly, hazardous missions diminish the expenses of numerous aerial operations. UAVs are being utilized in different nonmilitary missions, for example, news agencies, climate checking, and by-law authorization offices. Besides a wide utilization in military operations, for example, insight information gathering, observation and surveillance missions, and aerial targeting. These flexible applications have prompted propelled research for expanding the level of self-sufficiency of these unmanned aerial vehicles.

On the premise of their configuration, UAVs are separated into two primary classes: fixed-wing and rotary-wing UAVs. Fixed-wing UAVs are the most widely recognized sort.

Their configuration is straightforward when contrasted with different sorts and they can fly for long periods of time with high speeds. Nonetheless, these UAVs need runways or other recovery and launch frameworks like sling or parachute for takeoff and landing. Fixed-wing UAVs are not suitable to be flown indoors at low altitudes. Rotary wing UAVs are better as they do not require runways for takeoff and landing which makes them perfect in tight and congested regions.

In the following sections, a mathematical model is formulated for six degrees of freedom UAVs and used to construct a state-space model. Following that, controllability and observability checks are performed to design an appropriate controller. After that, the system full state variables are restated then the system transfer function is extracted. Finally, a stability check is carried out and the driven model is simulated.

Working Principle

A quadcopter is a four-rotor helicopter. It is an under-actuated, dynamic vehicle with four input forces (one for each rotor) and six degrees of freedom (6DOF). Unlike regular helicopters that have variable pitch angle rotors, a quadcopter has four fixed-pitch and fixed-angle rotors. As illustrated in figure 1, the motion of a quadcopter in 6DOF is controlled by varying the rpm of the four rotors individually, thereby changing the lift & rotational forces. Quadcopter tilts toward the direction of a slow spinning motor, which enables it to roll and pitch. Roll and pitch angles divide the thrust into two directions due to which linear motion is achieved. The rotors rotate in clockwise-anticlockwise pairs, to control the yaw produced due to the drag force on propellers. The center of gravity (CG) lies almost at the same plane which contains all the rotors. Also, all four motors of the same class differ in efficiency. This differentiates it from helicopters, and it is very difficult to stabilize a quadcopter by human control. Therefore, a sophisticated control is essential for a balanced flight of quadcopter

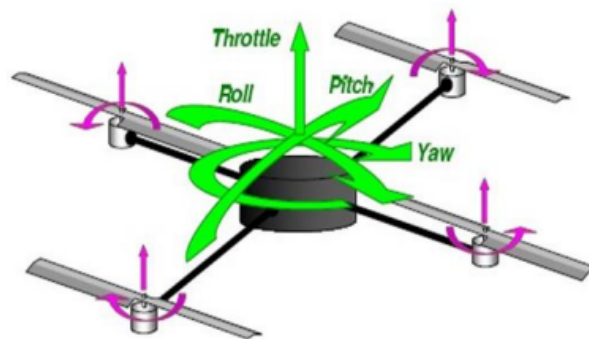


Figure 1. Quadcopter schematic

System Modeling

QuadCopter Dynamics

The motion of QuadCopter in 6DOF is controlled by varying the rpm of four rotors individually, thereby changing the vertical, horizontal, and rotational forces as depicted in figure 2.

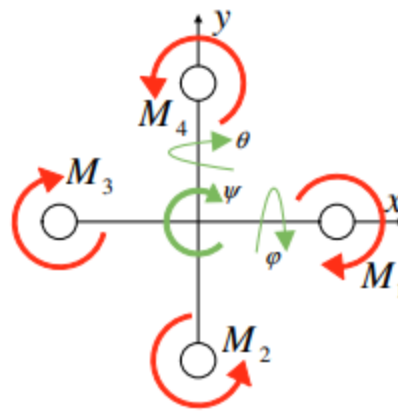


Figure 2. Quadcopter structure

Roll Motion

The torque about the x-axis is modeled as the rolling moment τ_x . Now r (moment arm) is the distance of individual forces from the x-axis.

$$\tau_x = r_1 f_1 + r_2 f_2 + r_3 f_3 + r_4 f_4 \quad (2)$$

The rotors 1 and 3 are placed along the body x-axis so their moment arm is zero and they do not contribute to the rolling moment as shown in figure 2.

$$r_1 = r_3 = 0$$

The moment arm for rotor 2 is positive as it will create a positive moment (counterclockwise), while for rotor 4 is negative because it will tend to rotate the system in a clockwise fashion and as a convention clockwise torque is taken as negative.

$$r_2 = d, r_4 = -d$$

So equation 2 becomes

$$\tau_x = d(f_2 - f_4) \quad (3)$$

Thus when rotor 4 applies force it tends to rotate the system about x-axis opposite to curl of fingers (figure 2) so using the right-hand rule (RHR) its torque contribution in the x-axis is negative. Similarly, rotor 2 produces positive torque about the x-axis according to RHR.

Pitch Motion

The torque about the y-axis is modeled as the pitching moment τ_y . Similarly, as the roll motion, the pitching moment is

$$\tau_y = d(f_1 - f_3) \quad (5)$$

Yaw Motion

Finally the yawing moment is modeled as and is the torque about the z-axis (figure 2 & 3). As the motors rotate the rotors encounter a reactive torque due to air drag on the rotor blade, this reactive torque is modeled as Q . The yaw is accomplished by rotor rpm imbalance between clockwise and counter-clockwise rotating propellers. The reactive torque as a convention is taken positively for counterclockwise rotating motors/rotors and negatively for clockwise rotating ones. All four rotors contribute to the yawing moment.

$$\tau_z = \sum_{i=1}^4 Q_i \quad (6)$$

Whereas $Q = cF_i$

$$\tau_z = c(-F_1 + F_2 - F_3 + F_4) \quad (7)$$

Where 'c' in Equation 7 is the force to moment scaling factor for a fixed motor-propeller set, and it is determined experimentally.

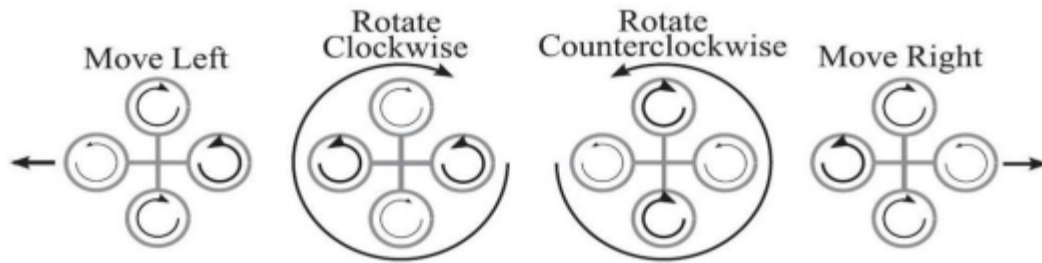


Figure 3. Yaw & linear motion due to rotor forces

Vertical Motion

The vertical motion of the quadcopter is governed by the total thrust force 'T' and weight 'w' along the z-axis (figure 2). The rpms of the four rotating propellers are changed simultaneously in order to accomplish vertical motion.

$$T = \sum_{i=1}^4 F_i \quad (8)$$

The vertical motion is described by the equation (Newton's 2nd law):

$$\ddot{z} = (T - mg)/m \quad (9)$$

Horizontal Motion

The horizontal motion in the x & y-axes is accomplished by decreasing the rpm of the rotor in whose direction it is intended to move, and subsequently increasing the rpm of the rotor on the opposite side of the same arm as depicted in figure 3. The force imbalance will cause the quadcopter to tilt to one side and the horizontal component of the thrust force will impart horizontal linear motion to the quadcopter (figure 4). If the mass of the quadcopter is 'm', then referring to (figure 4), the linear acceleration in the horizontal x-direction can be written as:

$$\ddot{x} = -T \cos \theta / m \quad (10)$$

Using small angle approximation ($\theta < 0.5$ rad):

$$\sin \theta \simeq \theta \quad (11)$$

Since θ is small

$$T \simeq mg \quad (12)$$

Thus, from equation 11 & 12, equation 10 becomes:

$$\ddot{x} = -g \theta \quad (13)$$

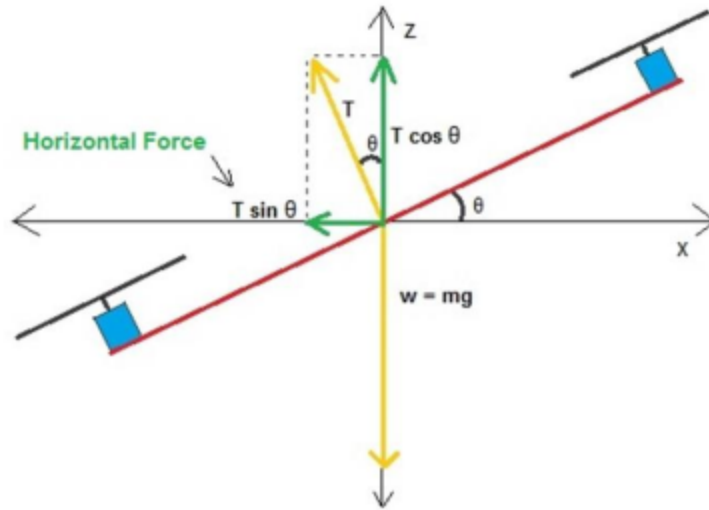


Figure 4. Horizontal motion

Similarly, can be derived out to be:

$$\ddot{y} = g\phi \quad (14)$$

Thus equations 3, 5, 7, 9, 13 & 14 give us the dynamics of the quadcopter in all of the 6 degrees of freedom. After establishing the system's dynamical equations, the state-space can be modeled easily.

State-Space Model

A state-space representation is a mathematical model of a physical system as a set of inputs, outputs, and state variables related by first-order differential equations. "State-space" refers to the space whose axes are the state variables. The state of the system can be represented as a vector within that space. The most general state-space representation of a linear system with p inputs, q outputs, and n state variables is written in the following form

$$\dot{x}(t) = Ax(t) + Bu(t) \quad (15)$$

$$y(t) = Cx(t) + Du(t) \quad (16)$$

Where

$x(t)$ is called the 'State Vector'

$y(t)$ is called the 'Output Vector'

$u(t)$ is called the 'Input (or control) Vector'

A is the 'System Matrix'

B is the 'Input Matrix'

C is the 'Output Matrix'

D is the 'Feedforward Matrix'

Quadcopter Modelling

We start by selecting the system states. The following twelve states are practically suitable for a quadcopter in 6DOF.

- a. Position along x-axis - x
- b. Position along y-axis - y
- c. Position along z-axis (height) - z
- d. Velocity along x-axis - \dot{x}
- e. Velocity along y-axis - \dot{y}
- f. Velocity along z-axis - \dot{z}
- g. Roll angle - ϕ
- h. Pitch angle θ
- i. Yaw angle - ψ
- j. Roll rate - $\dot{\phi}$
- k. Pitch rate - $\dot{\theta}$
- l. Yaw rate - $\dot{\psi}$

Thus, we will get our state vector x as follows:

$$x^T = [x \ y \ z \ \dot{x} \ \dot{y} \ \dot{z} \ \phi \ \theta \ \psi \ \dot{\phi} \ \dot{\theta} \ \dot{\psi}] \quad (17)$$

The input matrix is given as follows:

$$u^T = [U_1 \ U_2 \ U_3 \ U_4] \quad (18)$$

Where,

U_1 is the Total Upward Force on the quadcopter along z-axis

U_2 is the Pitch Torque (about the x-axis)

U_3 is the Roll Torque (about the y-axis)

U_4 is the Yaw Torque (about the z-axis)

The output y vector is given as follows:

$$y^T = [x \ y \ z \ \phi \ \theta \ \psi] \quad (19)$$

From equations 3, 5, 7, 9, 13, and 14, the state differential equations can be written as:

$$\begin{aligned}
 \dot{x} &= \dot{x} \\
 \dot{y} &= \dot{y} \\
 \dot{z} &= \dot{z} \\
 \ddot{x} &= -g\theta \\
 \ddot{y} &= g\phi \\
 \ddot{z} &= -U_1/m \\
 \dot{\phi} &= \dot{\phi} \\
 \dot{\theta} &= \dot{\theta} \\
 \dot{\psi} &= \dot{\psi} \\
 \ddot{\phi} &= U_2/I_x \\
 \ddot{\theta} &= U_3/I_y \\
 \ddot{\psi} &= U_4/I_z
 \end{aligned}$$

Where I_x, I_y, I_z are the moment of inertia of the quadcopter in the x, y, & z axes respectively.

The quadcopter's inputs are given as Pham, Ichalal, & Mammar (2019) stated:

$$\begin{aligned}
 U_1 &= b(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \\
 U_2 &= b(\omega_3^2 - \omega_1^2) \\
 U_3 &= b(\omega_4^2 - \omega_2^2) \\
 U_4 &= d(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2)
 \end{aligned}$$

Where ω_i , $i = 1, \dots, 4$ are the angular velocity of the i^{th} rotor, b, and d are thrust and drag coefficients.

The state differential equations are written in a matrix as follows:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \ddot{\psi} \\ \ddot{\theta} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \psi \\ \theta \\ \phi \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{1}{I_x} & 0 & 0 \\ 0 & 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & 0 & \frac{1}{I_z} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ z \\ \psi \\ \theta \\ \phi \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \psi \\ \theta \\ \phi \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix}$$

Numerical Values

All quadcopter parameters for simulation, taken from Bouabdallah, & Siegwart (2007), are mentioned in table 1.

Table 1. Quadcopter parameters definition

Parameter	Name	Value	Units
m	Quadcopter mass	0.65	Kg
r	Arm length	0.23	m
b	Thrust Coefficient	$3.13 * 10^5$	$N.m.s^2$

d	Drag Coefficient	$7.5 * 10^{-7}$	$N.m.s^2$
I_x, I_y	x,y Inertia	$7.5 * 10^{-3}$	$Kg.m^2$
I_z	z Inertia	$1.3 * 10^{-2}$	$Kg.m^2$
ω_i	Rotors speed	$[0, 400]$	$rad.s^{-1}$

Assuming that the quadcopter is symmetric, which means that $I_x = I_y$.

Controllability & Observability

Controllability and observability represent two major concepts of modern control system theory. These concepts were introduced by R. Kalman in 1960. They can be roughly defined as follows.

- Controllability: In order to be able to do whatever we want with the given dynamic system under control input, the system must be controllable.
- Observability: In order to see what is going on inside the system, the system must be observable.

The concepts of controllability and observability for a linear time-invariant dynamical system can be related to suitable linear systems of algebraic equations. It is well known that a solvable system of linear algebraic equations has a solution if and only if the rank of the system matrix is full. Observability and controllability tests will be connected to the rank tests of certain matrices: the controllability and observability matrices.

The observability matrix is given by:

$$\mathcal{O} = \begin{bmatrix} \mathbf{C} \\ \mathbf{C} \cdot \mathbf{A} \\ \mathbf{C} \cdot \mathbf{A}^2 \\ \vdots \\ \mathbf{C} \cdot \mathbf{A}^{11} \end{bmatrix} \in \mathbb{R}^{144 \times 6}$$

The linear continuous-time system is observable if and only if the observability matrix has full rank.

The controllability matrix is given by:

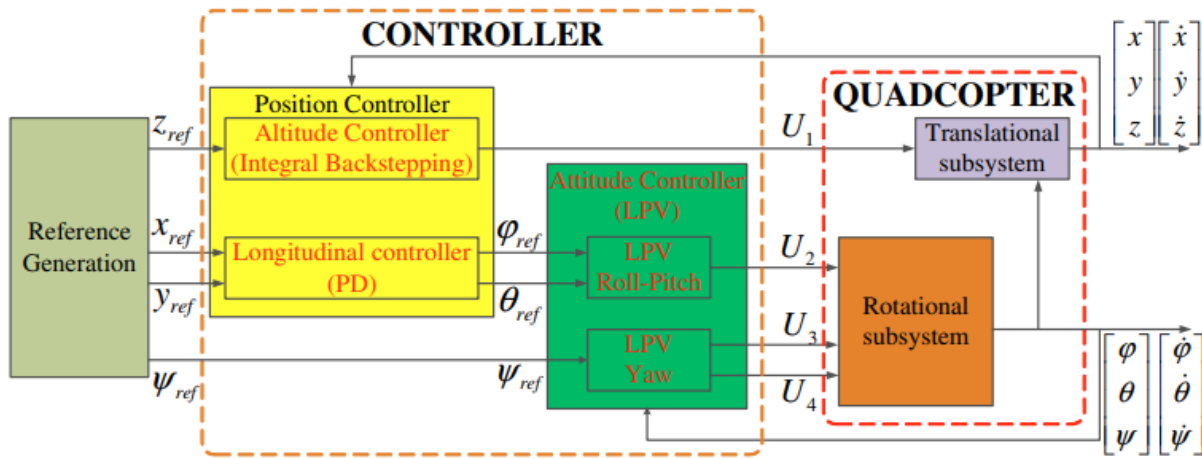
$$\mathcal{C} = [\mathbf{B} \quad \mathbf{A} \cdot \mathbf{B} \quad \mathbf{A}^2 \cdot \mathbf{B} \quad \dots \quad \mathbf{A}^{11} \cdot \mathbf{B}] \in \mathbb{R}^{12 \times 48}$$

The linear continuous-time system is controllable if and only if the controllability matrix has full rank. To check its observability and controllability, Matlab is used.

The **observability rank is 6** and the **controllability rank is 12**, hence the linear system results to be **controllable** and **observable**.

Designing a Controller

Control System Architecture



Linear Quadratic Regulator

The objective of the optimal control is to determine control signals so that the system to be controlled can meet physical constraints and minimize/maximize a cost/performance function. Namely, the solution of an optimization problem is supposed to bring the system's state $x(t)$ to the desired trajectory x_d minimizing some cost. Furthermore, it minimizes the use of the control inputs, thus reducing the use of actuators.

The control input $u(t)$ which minimizes the functional is a state linear feedback as

$$u(t) = -Kx(t)$$

Where:

$$K = R^{-1}B^TP$$

The P matrix is a solution of the reduced-matrix Riccati equation:

$$A^TP + PA - PBR^{-1}B^TP + Q = 0$$

Being P , R , & Q such matrices that:

- P is a positive definite matrix
- R is the cost of actuators ($R = R^T$ positive definite matrix; $R \in \mathbb{R}^{m \times m}$)
- Q is the cost of the state ($Q = Q^T$ positive semi-definite matrix; $Q \in \mathbb{R}^{r \times r}$)

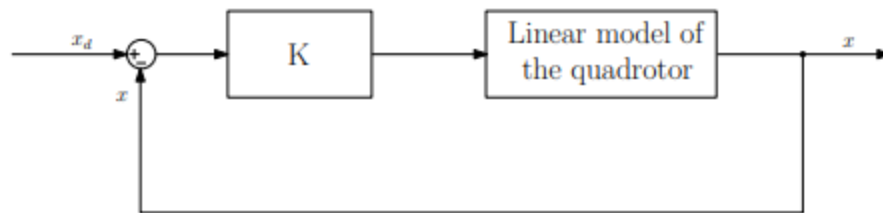


Figure 5. A simplified model of the LQR control.

Choosing $Q = \text{diag}(10, 10, 10, 10, 10, 10, 100, 100, 100, 100, 1)$, $R = \rho I_{4 \times 4}$

Where ρ is the control weight and has a value of 0.2.

Applying the LQR control using the LQR function from Matlab: $[K, P, e] = \text{lqr}(A, B, Q, R)$

The gain matrix K (4 rows for each input)

```
K =
    0.0000    -0.0000    -0.0000     0.0000    -0.0000    -0.0000     0.0000     0.0000    22.4256     0.0000    -0.0000     2.2361
   57.0850    -0.0000     0.0000     7.1314    -0.0000    -0.0000     0.0000    22.9352    -0.0000     0.0000     2.2361     0.0000
   -0.0000    63.0596     0.0000    -0.0000     7.1376     0.0000   -28.0620    -0.0000     0.0000   -22.3607    -0.0000     0.0000
   -0.0000     0.0000     7.0711    -0.0000     0.0000     7.0841    -0.0000    -0.0000     0.0000    -0.0000    -0.0000     0.0000
```

The P matrix

```
P =
   81.0813     0.0000     0.0000     0.0856     0.0000     0.0000    -0.0000    32.7085     0.0000     0.0000     3.1892    -0.0000
     0.0000    89.6065     0.0000     0.0000     0.0946    -0.0000   -40.0257    -0.0000    -0.0000   -31.9205    -0.0000    -0.0000
     0.0000     0.0000    10.0184     0.0000     0.0000     0.0184    -0.0000     0.0000    -0.0000    -0.0000    -0.0000     0.0000
     0.0856     0.0000     0.0000     0.0107     0.0000     0.0000    -0.0000     0.0344     0.0000    -0.0000     0.0034    -0.0000
     0.0000     0.0946     0.0000     0.0000     0.0107    -0.0000    -0.0421     0.0000     0.0000    -0.0335    -0.0000    -0.0000
     0.0000    -0.0000     0.0184     0.0000    -0.0000     0.0184     0.0000     0.0000    -0.0000     0.0000    -0.0000     0.0000
    -0.0000   -40.0257    -0.0000    -0.0000    -0.0421     0.0000    32.8231     0.0000     0.0000    28.7473     0.0000     0.0000
   32.7085    -0.0000     0.0000     0.0344     0.0000     0.0000     0.0000    26.3672     0.0000     0.0000     2.6024    -0.0000
     0.0000    -0.0000    -0.0000     0.0000     0.0000    -0.0000     0.0000     0.0000     2.9153     0.0000    -0.0000     0.2907
     0.0000   -31.9205    -0.0000    -0.0000    -0.0335     0.0000    28.7473     0.0000     0.0000    125.4969     0.0000     0.0000
     3.1892    -0.0000    -0.0000     0.0034    -0.0000    -0.0000     0.0000     2.6024    -0.0000     0.0000    10.2569    -0.0000
    -0.0000    -0.0000     0.0000    -0.0000    -0.0000     0.0000     0.0000    -0.0000     0.2907     0.0000    -0.0000    10.0290
```

```
e =  
  
1.0e+02 *  
  
-0.0010 + 0.0000i  
-0.0010 + 0.0000i  
-0.0100 + 0.0000i  
-0.0100 + 0.0000i  
-0.0394 + 0.0394i  
-0.0394 - 0.0394i  
-0.0397 + 0.0391i  
-0.0397 - 0.0391i  
-0.3440 + 0.0000i  
-5.4393 + 0.0000i  
-9.4281 + 0.0000i  
-9.4281 + 0.0000i
```

[illegible]

System Transfer Function

Since the model is a Multiple Input Multiple Output (MIMO) system (12 outputs to 4 inputs), there is twelve transfer function for each input i.e. $Y_1/U_1, Y_2/U_1, Y_3/U_1, \dots, Y_{12}/U_1$.

Evaluating the transfer function using Matlab:

For U_1

```
>> [num,den] = ss2tf(A-B*K, eye(12),eye(12),eye(12),1)

num =

1.0e+13 *

    0.0000    0.0000    0.0000    0.0001    0.0027    0.0412    0.3306    1.5179    3.9775    4.8624    2.4556    0.3608    0.0160
         0         0         0         0    -0.0000    -0.0000    -0.0000    -0.0000    -0.0000    -0.0000    -0.0000    -0.0000    -0.0000
         0         0         0         0    -0.0000    -0.0000    -0.0000    -0.0000    -0.0000    -0.0000    -0.0000    0.0000    0.0000
         0         0    -0.0000    -0.0000    -0.0004    -0.0195    -0.2252    -1.3045    -3.7861    -4.7933    -2.4504    -0.3608    -0.0160
         0         0         0    -0.0000    -0.0000    -0.0000    -0.0000    -0.0000    -0.0000    -0.0000    -0.0000    -0.0000    -0.0000
         0         0         0    -0.0000    -0.0000    -0.0000    -0.0000    -0.0000    -0.0000    -0.0000    -0.0000    -0.0000    -0.0000
         0         0         0         0    -0.0000    -0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    -0.0000    -0.0000
         0         0    0.0000    0.0000    0.0000    0.0006    0.0220    0.1908    0.8431    1.2499    0.6272    0.0510         0
         0         0         0    -0.0000    -0.0000    -0.0000    -0.0000    -0.0000    -0.0000    0.0000    0.0000    0.0000    0.0000
         0         0         0         0         0    -0.0000    -0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    -0.0000
         0         0         0    0.0000    0.0000    0.0000    0.0006    0.0220    0.1908    0.8431    1.2499    0.6272    0.0510
         0         0         0         0    -0.0000    -0.0000    -0.0000    -0.0000    -0.0000    -0.0000    0.0000    0.0000    0.0000

den =

1.0e+13 *

    0.0000    0.0000    0.0000    0.0001    0.0027    0.0390    0.3111    1.4319    3.8501    4.7985    2.4504    0.3608    0.0160
```

For U_2

```
>> [num,den] = ss2tf(A-B*K, eye(12),eye(12),eye(12),2)

num =

1.0e+13 *

         0         0         0    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    -0.0000    -0.0000    -0.0000
    0.0000    0.0000    0.0000    0.0001    0.0027    0.0411    0.3287    1.5025    3.9157    4.8096    2.4509    0.3608    0.0160
         0         0         0         0    0.0000    0.0000    0.0000    0.0000    0.0000    -0.0000    -0.0000    -0.0000    -0.0000
         0         0    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    -0.0000    0.0000    0.0000    0.0000    0.0000
         0         0    -0.0000    -0.0000    -0.0005    -0.0213    -0.2405    -1.3663    -3.8390    -4.7980    -2.4504    -0.3608    -0.0160
         0         0         0    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
         0         0    -0.0000    -0.0000    -0.0000    -0.0006    -0.0215    -0.1728    -0.6925    -0.6443    -0.1088    -0.0051         0
         0         0         0         0    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    -0.0000
         0         0         0    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    -0.0000
         0         0         0    -0.0000    -0.0000    -0.0000    -0.0006    -0.0215    -0.1728    -0.6925    -0.6443    -0.1088    -0.0051
         0         0         0         0         0    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
         0         0         0         0    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000

den =

1.0e+13 *

    0.0000    0.0000    0.0000    0.0001    0.0027    0.0390    0.3111    1.4319    3.8501    4.7985    2.4504    0.3608    0.0160
```


For U_3

```
>> [num,den] = ss2tf(A-B*K, eye(12),eye(12),eye(12),3)

num =

1.0e+13 *

    0         0         0 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000 0.0000 -0.0000
    0         0         0 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000
    0.0000    0.0000    0.0000 0.0001 0.0027 0.0416 0.3475 1.7072 5.0093 7.4963 4.5599 0.7063 0.0320
    0         0         0 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
    0         0 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000 0         0         0
    0         0 -0.0000 -0.0000 -0.0001 -0.0026 -0.0358 -0.2727 -1.1522 -2.6890 -2.1050 -0.3448 -0.0160
    0         0         0 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0
    0         0         0 0.0000 0.0000 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000 0
    0         0         0 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 -0.0000
    0         0         0 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
    0         0         0 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
    0         0         0 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
    0         0         0 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

den =

1.0e+13 *

    0.0000    0.0000    0.0000 0.0001 0.0027 0.0390 0.3111 1.4319 3.8501 4.7985 2.4504 0.3608 0.0160
```

For U_4

```
>> [num,den] = ss2tf(A-B*K, eye(12),eye(12),eye(12),4)

num =

1.0e+13 *

    0         0    0.0000 0.0000 0.0000 0.0000 0.0000 0.0001 0.0001 0.0001 0.0000 0         0
    0         0         0 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 -0.0000 -0.0000 -0.0000 -0.0000
    0         0         0 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
    0.0000    0.0000    0.0000 0.0001 0.0027 0.0390 0.3112 1.4321 3.8501 4.7985 2.4504 0.3608 0.0160
    0         0    0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 -0.0000 -0.0000 -0.0000 0.0000
    0         0    0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
    0         0         0 0.0000 0.0000 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000 0.0000
    0         0         0 0.0000 0.0000 0.0000 0.0000 0.0002 0.0009 0.0013 0.0007 0.0001 0
    0         0         0 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
    0         0         0 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
    0         0         0 0.0000 0.0000 0.0000 0.0000 0.0002 0.0009 0.0013 0.0007 0.0001 0.0001
    0         0         0 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

den =

1.0e+13 *

    0.0000    0.0000    0.0000 0.0001 0.0027 0.0390 0.3111 1.4319 3.8501 4.7985 2.4504 0.3608 0.0160
```

System Stability

In order to check the system stability, the matrix P is checked whether it is a positive definite or not. If it is positive definite, then the system is stable.

It is easily checked using Matlab and results that it is a positive definite matrix which means the system is stable.

The matrix P

```
P =
```

81.0813	0.0000	0.0000	0.0856	0.0000	0.0000	-0.0000	32.7085	0.0000	0.0000	3.1892	-0.0000
0.0000	89.6065	0.0000	0.0000	0.0946	-0.0000	-40.0257	-0.0000	-0.0000	-31.9205	-0.0000	-0.0000
0.0000	0.0000	10.0184	0.0000	0.0000	0.0184	-0.0000	0.0000	-0.0000	-0.0000	-0.0000	0.0000
0.0856	0.0000	0.0000	0.0107	0.0000	0.0000	-0.0000	0.0344	0.0000	-0.0000	0.0034	-0.0000
0.0000	0.0946	0.0000	0.0000	0.0107	-0.0000	-0.0421	0.0000	0.0000	-0.0335	-0.0000	-0.0000
0.0000	-0.0000	0.0184	0.0000	-0.0000	0.0184	0.0000	0.0000	-0.0000	0.0000	-0.0000	0.0000
-0.0000	-40.0257	-0.0000	-0.0000	-0.0421	0.0000	32.8231	0.0000	0.0000	28.7473	0.0000	0.0000
32.7085	-0.0000	0.0000	0.0344	0.0000	0.0000	0.0000	26.3672	0.0000	0.0000	2.6024	-0.0000
0.0000	-0.0000	-0.0000	0.0000	0.0000	-0.0000	0.0000	0.0000	2.9153	0.0000	-0.0000	0.2907
0.0000	-31.9205	-0.0000	-0.0000	-0.0335	0.0000	28.7473	0.0000	0.0000	125.4969	0.0000	0.0000
3.1892	-0.0000	-0.0000	0.0034	-0.0000	-0.0000	0.0000	2.6024	-0.0000	0.0000	10.2569	-0.0000
-0.0000	-0.0000	0.0000	-0.0000	-0.0000	0.0000	0.0000	-0.0000	0.2907	0.0000	-0.0000	10.0290

The check

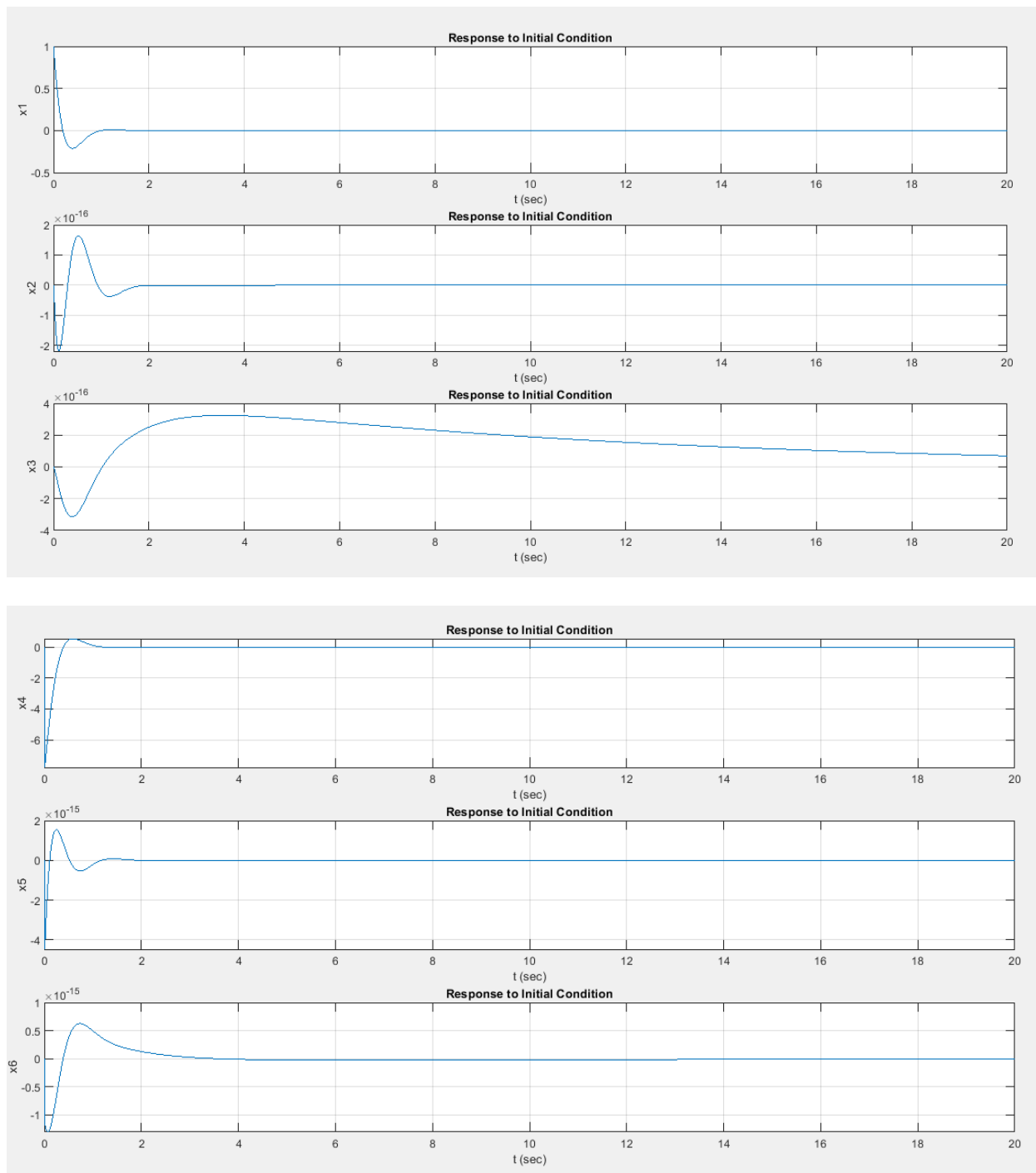
```
>> try chol(P)
    disp('Matrix is symmetric positive definite.')
catch ME
    disp('Matrix is not symmetric positive definite')
end
```

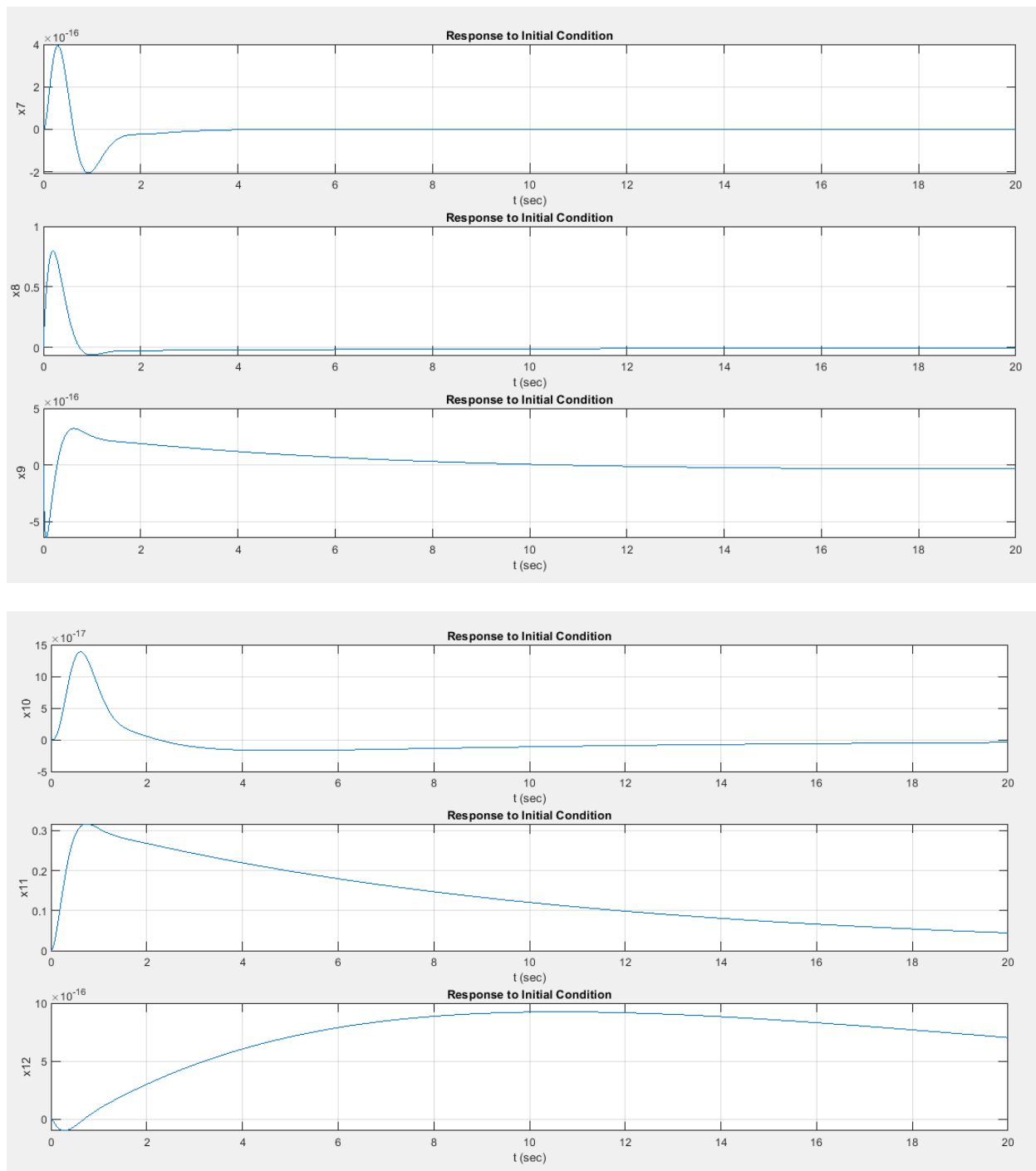
```
ans =
```

9.0045	0.0000	0.0000	0.0095	0.0000	0.0000	-0.0000	3.6325	0.0000	0.0000	0.3542	-0.0000
0	9.4661	0.0000	0.0000	0.0100	-0.0000	-4.2283	-0.0000	-0.0000	-3.3721	-0.0000	-0.0000
0	0	3.1652	0.0000	0.0000	0.0058	-0.0000	-0.0000	-0.0000	-0.0000	-0.0000	0.0000
0	0	0	0.1030	0.0000	0.0000	-0.0000	-0.0014	-0.0000	-0.0000	-0.0001	-0.0000
0	0	0	0	0.1030	-0.0000	0.0015	0.0000	0.0000	0.0015	-0.0000	-0.0000
0	0	0	0	0	0.1356	0.0000	0.0000	0.0000	0.0000	-0.0000	0.0000
0	0	0	0	0	0	3.8658	-0.0000	0.0000	3.7480	0.0000	0.0000
0	0	0	0	0	0	0	3.6294	0.0000	0.0000	0.3625	-0.0000
0	0	0	0	0	0	0	0	1.7074	0.0000	-0.0000	0.1702
0	0	0	0	0	0	0	0	0	10.0039	0.0000	0.0000
0	0	0	0	0	0	0	0	0	0	3.1623	-0.0000
0	0	0	0	0	0	0	0	0	0	0	3.1623

Matrix is symmetric positive definite.

Simulation





As the time response to an impulse initial condition graphs show, **all the states converging are stable.**

Matlab Code

%% Entering the system matrices

```
A = [0 0 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 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 0 0; 0 0 0 0 0 0 0 0 0 0 0; 0 -9.81 0 0 0 0 0 0 0 0 0; 9.81 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 1 0 0 0 0; 0 0 0 0 0 0 0 1 0 0 0; 0 0 0 0 0 0 0 0 1 0 0];
```

```
B = [0 0 0 0; 0 0 0 0; 0 0 0 0; 0 1/0.0075 0 0; 0 0 1/0.0075 0; 0 0 0 1/0.0130; 0 0 0 0; 0 0 0 0; 1/0.65 0 0 0; 0 0 0 0; 0 0 0 0];
```

```
C = [1 0 0 0 0 0 0 0 0 0 0; 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 1 0 0 0 0 0 0 0; 0 0 0 0 1 0 0 0 0 0 0; 0 0 0 0 0 1 0 0 0 0 0];
```

%% Checking controllability & observability

```
M = [B A*B A^2*B A^3*B A^4*B A^5*B A^6*B A^7*B A^8*B A^9*B A^10*B A^11*B];
```

```
N = [C C*A C*A^2 C*A^3 C*A^4 C*A^5 C*A^6 C*A^7 C*A^8 C*A^9 C*A^10 C*A^11];
```

```
Mrank = rank(M);
```

```
Nrank = rank(N);
```

```
if Mrank == 12
```

```
    disp('the system is completely state controllable');
```

```
else
```

```
    disp('the system is NOT completely state controllable');
```

```
end
```

```
if Nrank == 6
```

```
    disp('the system is completely observable');
```

```
else
```

```
    disp('the system is NOT completely observable');
```

```
end
```

%% Linear Quadratic Regulator

```
v=[10, 10, 10, 10, 10, 10, 100, 100, 100, 100, 1, 1];
```

```
Q = diag(v);
```

```

R = 0.2 * eye(4,4);
[K,P,e] = lqr(A,B,Q,R);

%% Simulating the system
% Intializing the system with an impulse input
sys = ss(A-B*K, eye(12),eye(12),eye(12));
t = 0:0.01:20;
x = initial(sys,[1 0 0 0 0 0 0 0 0 0 0],t);
x1 = [1 0 0 0 0 0 0 0 0 0 0 0]*x';
x2 = [0 1 0 0 0 0 0 0 0 0 0 0]*x';
x3 = [0 0 1 0 0 0 0 0 0 0 0 0]*x';
x4 = [0 0 0 1 0 0 0 0 0 0 0 0]*x';
x5 = [0 0 0 0 1 0 0 0 0 0 0 0]*x';
x6 = [0 0 0 0 0 1 0 0 0 0 0 0]*x';
x7 = [0 0 0 0 0 0 1 0 0 0 0 0]*x';
x8 = [0 0 0 0 0 0 0 1 0 0 0 0]*x';
x9 = [0 0 0 0 0 0 0 0 1 0 0 0]*x';
x10 = [0 0 0 0 0 0 0 0 0 1 0 0]*x';
x11 = [0 0 0 0 0 0 0 0 0 0 1 0]*x';
x12 = [0 0 0 0 0 0 0 0 0 0 0 1]*x';

% Ploting the states response
subplot(3,1,1); plot(t,x1), grid
title('Response to Initial Condition')
ylabel('x1')
xlabel('t (sec)')
subplot(3,1,2); plot(t,x2),grid
title('Response to Initial Condition')
ylabel('x2')
xlabel('t (sec)')
subplot(3,1,3); plot(t,x3),grid
title('Response to Initial Condition')

```



```
xlabel('t (sec)')
```

```
ylabel('x3')
```

```
figure
```

```
subplot(3,1,1); plot(t,x4),grid
```

```
title('Response to Initial Condition')
```

```
ylabel('x4')
```

```
xlabel('t (sec)')
```

```
subplot(3,1,2); plot(t,x5),grid
```

```
title('Response to Initial Condition')
```

```
ylabel('x5')
```

```
xlabel('t (sec)')
```

```
subplot(3,1,3); plot(t,x6),grid
```

```
ylabel('x6')
```

```
title('Response to Initial Condition')
```

```
xlabel('t (sec)')
```

```
figure
```

```
subplot(3,1,1); plot(t,x7),grid
```

```
title('Response to Initial Condition')
```

```
ylabel('x7')
```

```
xlabel('t (sec)')
```

```
subplot(3,1,2); plot(t,x8),grid
```

```
title('Response to Initial Condition')
```

```
ylabel('x8')
```

```
xlabel('t (sec)')
```


```
subplot(3,1,3); plot(t,x9),grid
```

```
title('Response to Initial Condition')
```

```
ylabel('x9')
```

```
xlabel('t (sec)')
```

```
figure
```



```
subplot(3,1,1); plot(t,x10),grid
title('Response to Initial Condition')
ylabel('x10')
xlabel('t (sec)')
subplot(3,1,2); plot(t,x11),grid
title('Response to Initial Condition')
ylabel('x11')
xlabel('t (sec)')
subplot(3,1,3); plot(t,x12),grid
title('Response to Initial Condition')
ylabel('x12')
xlabel('t (sec)')
```


References

1. Pham, T. H., Ichalal, D., & Mammar, S. (2019). LPV and Nonlinear-based control of an Autonomous Quadcopter under variations of mass and moment of inertia. *IFAC-PapersOnLine*, 52(28), 176–183. doi:10.1016/j.ifacol.2019.12.371
2. Samir Bouabdallah. Design and control of quadrotors with application to autonomous flying. 01 2007. doi: 10.5075/epfl-thesis-3727
3. Samir Bouabdallah and Roland Siegwart. Backstepping and Sliding-mode Techniques Applied to an Indoor Micro Quadrotor. *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference*, pages 153–158, 2007
4. Tahir, Z., Jamil, M., Liaqat, S. A., Mubarak, L., Tahir, W., & Gilani, S. O. (2016). State Space System Modeling of a Quad Copter UAV. *Indian Journal of Science and Technology*, 9(27). doi:10.17485/ijst/2016/v9i27/96613
5. Srinivasan A., (May 2021). Modeling, Design and Control of a 6 D-O-F Quadcopter Fleet With Platooning Control. [Master's Thesis, Arizona State University]
6. SABATINO F., (Jun. 2015). Quadrotor control: modeling, nonlinear control design, and simulation. [Master's Degree Project, KTH electrical engineering]. XR-EE-RT 2015:XXX
7. R. M. Murray, (Mar., 2009). Optimization-Based Control [manuscript]. *Control and Dynamical Systems California Institute of Technology*.

controllability & Observability

Controllability and observability represent two major concepts of modern control system theory. These concepts were introduced by R. Kalman in 1960. They can be roughly defined as follows.