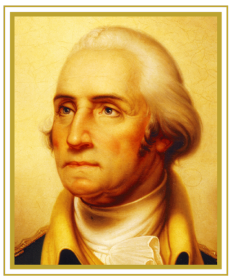
Dynamics of quadcopter and its control system

Peiji Tang, David Phelps *

A project report presented for MAE 6245



THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON DC

Department of Mechanical and Aerospace Engineering

The George Washington University

Washington, DC

April 25, 2017

*Nikhil Nigam

Contents

1 Dynamics of quadcopter **1**
1.1 Introduction 1
1.2 State variables and Euler angle sets 1
1.3 Thrust and drag force 2
1.4 Torques 3
1.5 Equation of motion 3
1.6 Parameters 7

2 Linearlization **8**
2.1 System matrix 8
2.2 Controllability 10
2.3 Stability 11

3 Controller design **12**
3.1 Linear-quadratic regulator 12
3.2 Numerical simulation of system 13
3.2.1 Simulation of uncontrolled system 13
3.2.2 Simulation of controlled system 16

Appendices **19**

Appendix A LQR control via MATLAB **19**

Appendix B Algebra Calculation via Python **23**

1 Dynamics of quadcopter

1.1 Introduction

A quadcopter is a multirotor helicopter which is lifted and propelled by four rotors. The set of vertically oriented propellers made quadcopter as a rotorcraft which is opposed to fixed-wing aircraft. Quadcopters generally utilize two pairs of identical fixed pitched propellers: two clockwise(CW) and two counter-clockwise(CCW). The control can be achieved through independent variation of speed of each rotor.By changing the speed it is possible to generate a desired thrust; to locate for the center of thrust both laterally and longitudinally; and to create a desired total torque, or tuning force[1].

The quadcopers has higher payload than other aircrafts, simplicity of control and has great manoeuvring attitude which can help in going in several areas cannot be accessed by traditional airplanes nor helicopters[2]. The symmetry of the quadcopter body gives simplicity to the controller design as it can be controlled through changing speed of each propellers. Compared with helicopters, quadcopters craft has further efficiency and mechanical advantages[3]. Their smaller blades are also advantageous because they process less kinetic energy, reducing their ability to cause damage.

The development of quadcopters has stalled until 2000s, since the incredible difficulty without electronic assistance[4]. With an decreasing cost of modern microprocessors has made electronic and even completely autonomous control of quadcopters feasible for application in Drone-delivery, Photography, Sport-Racing and Freestyle, Military and law enforcement, research platform etc.

With Six degree of freedom(DOM) which included: three translational and three rotational and four independent inputs(rotor speed), quadcopters are severally under-actuated. The resulting of dynamics are highly nonlinear. The complex aerodynamics properties has been introduced in *G.M. Hoffmann et al* paper[5;6]

1.2 State variables and Euler angle sets

In order to describe the dynamics of quadcopter, it is necessary to construct adequate state variables to accomplish this task.The state variables consists of linear position (3D displacement of quadcopter under inertial frame), the velocity of quadcopter, three euler angles, and angular velocity possessed by quadcopter. The definition of position and velocity are given below:

$$Position = \mathbf{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad Velocity = \dot{\mathbf{p}} = \mathbf{v} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}$$

Three Euler angles included:

- Roll angle(ϕ) - rotation of of quadcopter around x-axis
- Pitch angle(θ) - rotation of quadcopter around y-axis

- Yaw angle(ψ) - rotation of of quadcopter around z-axis

Thus, the euler angle vector and its derivatives can be written as:

$$Euler\ angles = \eta = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}, \quad Euler\ angles\ derivatives = \dot{\eta} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

However, the euler angles derivative is not equal to the angular velocity of quadcopter. The angular velocity is a vector pointing along the axis of rotation. Conversion of angle derivative to angular velocity (ω):

$$\omega = C \cdot \dot{\eta} = \begin{bmatrix} 1 & 0 & -sin(\theta) \\ 0 & cos(\phi) & cos(\theta)sin(\phi) \\ 0 & -sin\phi & cos(\theta)cos(\phi) \end{bmatrix} \cdot \dot{\eta} = \begin{bmatrix} 1 & 0 & -sin(\theta) \\ 0 & cos(\phi) & cos(\theta)sin(\phi) \\ 0 & -sin\phi & cos(\theta)cos(\phi) \end{bmatrix} \cdot \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

There are 12 types of Euler angle sets to reflect the different rotating status of quadcopter. To reduce the repeating work, This project paper will choose 3-1-3 Euler angles, so the rotation matrix can be given as below[7]:

$$R = \begin{bmatrix} cos(\phi)cos(\psi) - cos(\theta)sin(\phi)sin(\psi) & -cos(\psi)sin(\phi) - cos(\phi)cos(\theta)sin(\psi) & sin(\theta)sin(\psi) \\ cos(\theta)cos(\psi)sin(\phi) & cos(\phi)cos(\theta)cos(\psi) - sin(\phi)sin(\psi) & -cos(\psi)sin(\theta) \\ sin(\phi)sin(\theta) & cos(\phi)sin(\theta) & cos(\theta) \end{bmatrix}$$

1.3 Thrust and drag force

According to *Gibiansky* paper[4], the thrust of quadcopter is:

$$T_b = k \begin{bmatrix} 0 \\ 0 \\ \sum_{i=1}^{4} {\Omega_i}^2 \end{bmatrix}$$

It is clearly to notice that thrust is in the z-axis. T_b indicates the thrust under body-fixed-frame and k is a dimensionless constant called lift constant. $\Omega_i, i = 1, 2, 3, 4$ shows the angular velocity of each motor.

In addition to the thrust force, quadcopter will sustain drag force which is proportional to the linear velocity in each direction. So the drag force can be expressed as below:

$$F_d = \begin{bmatrix} -k_d\dot{x} \\ -k_d\dot{y} \\ -k_d\dot{z} \end{bmatrix}$$

Where k_d indicates drag coefficient.

1.4 Torques

Each rotor contributes some torque about the z-axis.This torque is to keep the propeller spinning and providing thrust.It creates the instantaneous angular acceleration and overcomes the frictional drag forces.Drag depends on the properties of the fluid and on the size,shape,and speed of the object. One way to express this is by means of drag equation:

$$F_d = \frac{1}{2}\rho C_d A v^2$$

where F_d is drag force, ρ is the density of air, C_d is the drag coefficient, v is the speed of quadcopter relative to air, A - propeller cross-section area.

This is implies that torque due to drag is given by:

$$\tau_d = \frac{1}{2}R\rho C_d A v^2 = \frac{1}{2}R\rho C_d A(\Omega R)^2 = b\Omega^2$$

where R is the radius of the propeller, b is a dimensionless constant. Then write the complete torque about the z-axis for ith motor:

$$\tau_z = b\Omega^2 + I_z\dot{\Omega}$$

where I_z is the moment of inertia about the motor z axis, $\dot{\Omega}$ is the angular acceleration of the propeller. Note that in steady flight $\dot{\Omega} = 0$. Thus ignore this term,simplifying the entire expression into:

$$\tau_z = (-1)^{i+1} b{\Omega_i}^2$$

Hence, the roll,pitch,yaw torque and total torque in body-fixed-frame:

$$\begin{aligned}
\tau_\phi &= lk({\Omega_1}^2 - {\Omega_3}^2) \\
\tau_\theta &= lk({\Omega_2}^2 - {\Omega_4}^2) \\
\tau_\psi &= b({\Omega_1}^2 - {\Omega_2}^2 + {\Omega_3}^2 - {\Omega_4}^2) \\
\tau_b &= \begin{bmatrix} lk({\Omega_1}^2 - {\Omega_3}^2) \\ lk({\Omega_2}^2 - {\Omega_4}^2) \\ b({\Omega_1}^2 - {\Omega_2}^2 + {\Omega_3}^2 - {\Omega_4}^2) \end{bmatrix}
\end{aligned}$$

where l is the distance from the center of the quadcopter to any of the propellers.

1.5 Equation of motion

Under the inertia frame, the quadcopter suffered thrust, drag(linear), gravity, so the resultant force can be separated into three parts. Recall that the derived thrust is under body-fixed-frame, so in the linear equation of motion thrust should be transformed to

corresponding force under inertia frame. Thus, the linear equation of motion can be generalized as:

$$m\ddot{p} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \cdot T_b + F_d \tag{1}$$

The second part of deriving equation of motion is to find the rotational equations of motion under body-fixed-frame. Let's recall the Euler equation for rigid body dynamics:

$$\begin{aligned} I\dot{\omega} + \omega \times (I\omega) &= \tau \\ \implies \dot{\omega} &= I^{-1}(\tau - \omega \times (I\omega)) \end{aligned} \tag{2}$$

where $\dot{\omega} = \begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix}$ and I indicates the inertia matrix since the quadcopter rotates about the principal axes, so the inertia matrix can be written as $I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$. To simplify the cross product, a algorithm called hat map introduced. Assume that $x, y \in R^{3\times 1}$.

$$x \times y = \hat{x} \cdot y = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix} \cdot \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

Then $\dot{\omega} = I^{-1}(\tau - \hat{\omega} \cdot I\omega)$. Using Python to calculate this equation see in Appendix 1:

$$\dot{\omega} = \begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} = \begin{bmatrix} \frac{1}{I_{xx}}(I_{yy}\omega_y\omega_z - I_{zz}\omega_y\omega_z + \tau_\phi) \\ \frac{1}{I_{yy}}(-I_{xx}\omega_x\omega_z + I_{zz}\omega_x\omega_z + \tau_\theta) \\ \frac{1}{I_{zz}}(I_{xx}\omega_x\omega_y - I_{yy}\omega_x\omega_y + \tau_\psi) \end{bmatrix} \tag{3}$$

With Eq (1) and (3), the total equation of motion obtained, but it may looks too complicated. In order to make these equations more readable. Let us redefine state

variables $X = \begin{bmatrix} x1 \\ x2 \\ x3 \\ x4 \end{bmatrix} \in R^{12}$:

$$x1 = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} - \text{linear position of quadcopter} \tag{4}$$

$$x2 = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} x_4 \\ x_5 \\ x_6 \end{bmatrix} - \text{linear velocity of quadcopter} \tag{5}$$

$$x3 = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} x_7 \\ x_8 \\ x_9 \end{bmatrix} - \text{three euler angles} \tag{6}$$

$$x4 = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} x_{10} \\ x_{11} \\ x_{12} \end{bmatrix} - \text{angular velocity of quadcopter} \tag{7}$$

Then equation of motion $\dot{X} = \begin{bmatrix} \dot{x}1 \\ \dot{x}2 \\ \dot{x}3 \\ \dot{x}4 \end{bmatrix}$ should be(Obtained via **Python** see in Appendix):

$$\dot{x}1 = x2 = \begin{bmatrix} x_4 \\ x_5 \\ x_6 \end{bmatrix} \tag{8}$$

$$\begin{aligned} \dot{x}2 &= \begin{bmatrix} -\frac{k_d\dot{x}}{m} + \frac{1}{m}\left({\Omega_1}^2 + {\Omega_2}^2 + {\Omega_3}^2 + {\Omega_4}^2\right) sin(\psi)sin(\theta) \\ -\frac{k_d\dot{y}}{m} - \frac{1}{m}\left({\Omega_1}^2 + {\Omega_2}^2 + {\Omega_3}^2 + {\Omega_4}^2\right) sin(\theta)cos(\psi) \\ -g - \frac{k_d\dot{z}}{m} + \frac{1}{m}\left({\Omega_1}^2 + {\Omega_2}^2 + {\Omega_3}^2 + {\Omega_4}^2\right) cos(\theta) \end{bmatrix} \\ &= \begin{bmatrix} -\frac{k_d x_4}{m} + \frac{1}{m}\left({\Omega_1}^2 + {\Omega_2}^2 + {\Omega_3}^2 + {\Omega_4}^2\right) sin(x_9)sin(x_8) \\ -\frac{k_d x_5}{m} - \frac{1}{m}\left({\Omega_1}^2 + {\Omega_2}^2 + {\Omega_3}^2 + {\Omega_4}^2\right) sin(x_8)cos(x_9) \\ -g - \frac{k_d x_6}{m} + \frac{1}{m}\left({\Omega_1}^2 + {\Omega_2}^2 + {\Omega_3}^2 + {\Omega_4}^2\right) cos(x_8) \end{bmatrix} \end{aligned} \tag{9}$$

$$\dot{x}3 = C^{-1}\begin{bmatrix} x_{10} \\ x_{11} \\ x_{12} \end{bmatrix}$$

$$= \begin{bmatrix} x_{10} + \dfrac{x_{11}sin(\phi)sin(\theta)}{\left(\dfrac{sin^2(\phi)cos(\theta)}{cos(\phi)} + cos(\phi)cos(\theta)\right)cos(\phi)} + \dfrac{x_{12}sin(\theta)}{\dfrac{sin^2(\phi)cos(\theta)}{cos(\phi)} + cos(\phi)cos(\theta)} \\ \left(\dfrac{x_{11}}{cos(\phi)}\left(1 - \dfrac{sin^2(\phi)cos(\theta)}{\left(\dfrac{sin^2(\phi)cos(\theta)}{cos(\phi)} + cos(\phi)cos(\theta)\right)cos(\phi)}\right) - \right. \\ \left. \dfrac{x_{12}sin(\phi)cos(\theta)}{\left(\dfrac{sin^2(\phi)cos(\theta)}{cos(\phi)} + cos(\phi)cos(\theta)\right)cos(\phi)}\right) \\ \dfrac{x_{11}sin(\phi)}{\left(\dfrac{sin^2(\phi)cos(\theta)}{cos(\phi)} + cos(\phi)cos(\theta)\right)cos(\phi)} + \dfrac{x_{12}}{\dfrac{sin^2(\phi)cos(\theta)}{cos(\phi)}cos(\phi)cos(\theta)} \end{bmatrix}$$

$$= \begin{bmatrix} x_{10} + \dfrac{x_{11}sin(x_7)sin(x_8)}{\left(\dfrac{sin^2(x_7)cos(x_8)}{cos(x_7)} + cos(\phi)cos(x_8)\right)cos(x_7)} + \dfrac{x_{12}sin(x_8)}{\dfrac{sin^2(x_7)cos(x_8)}{cos(x_7)} + cos(\phi)cos(\theta)} \\ \left(\dfrac{x_{11}}{cos(x_7)}\left(1 - \dfrac{sin^2(x_7)cos(x_8)}{\left(\dfrac{sin^2(x_7)cos(x_8)}{cos(x_7)} + cos(x_7)cos(x_8)\right)cos(x_7)}\right) - \right. \\ \left. \dfrac{x_{12}sin(x_7)cos(x_8)}{\left(\dfrac{sin^2(x_7)cos(x_8)}{cos(x_7)} + cos(x_7)cos(x_8)\right)cos(x_7)}\right) \\ \dfrac{x_{11}sin(x_7)}{\left(\dfrac{sin^2(x_7)cos(x_8)}{cos(x_7)} + cos(x_7)cos(x_8)\right)cos(x_7)} + \dfrac{x_{12}}{\dfrac{sin^2(x_7)cos(x_8)}{cos(x_7)}cos(x_7)cos(x_8)} \end{bmatrix} \tag{10}$$

$$\dot{x}4 = \dot{\omega}$$
$$= \begin{bmatrix} \frac{1}{I_{xx}}\left(I_{yy}x_{11}x_{12} - I_{zz}x_{11}x_{12} + lk(\Omega_1^2 - \Omega_3^2)\right) \\ \frac{1}{I_{yy}}\left(-I_{xx}x_{10}x_{12} + I_{zz}x_{10}x_{12} + lk(\Omega_2^2 - \Omega_4^2)\right) \\ \frac{1}{I_{zz}}\left(I_{xx}x_{10}x_{11} - I_{yy}x_{10}x_{11} + b(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2)\right) \end{bmatrix} \tag{11}$$

1.6 Parameters

Eq(8)-(11) indicates the final equation of motion of quadcopter. It is easy to recongnize that this system is a complicated nonlinear system. Parameters shown in above equations are given below[8]:

Parameter	Value	Units
m	0.468	kg
g	9.81	m/s^2
l	0.225	m
k	$2.980 \cdot 10^{-6}$	
b	$1.140 \cdot 10^{-7}$	
k_d	0.25	s^{-1}
I_{xx}	$4.856 \cdot 10^{-3}$	$kg \cdot m^2$
I_{yy}	$4.856 \cdot 10^{-3}$	$kg \cdot m^2$
I_{zz}	$8.801 \cdot 10^{-3}$	$kg \cdot m^2$

Table 1: Parameters shown in equations of motion

2 Linearlization

2.1 System matrix

For equations of derived in first chapter is complex which may obstruct fundamental research in its system.So In order to explain the dynamics of quadcopter in a less precise but more efficient way. Linearlizing above system is acceptable and necessary since analyzing this system in a nonlinear way required to construct a lyapunov function to measure the stability of the system. Generally, the lyaounov function modeled on energy however the above derivation ignored the energy loss of each rotor which will affected the accuracy of lyapunov function. Besides that one certain lyapunov function fails to prove the stability of system does not indicate there is no other lyapunov function could test system is stable. Thus, based on these two reasons, This project attempted to take this system into consideration from a linear point which is able to avoid above deficiencies.

For origin nonlinear system of quadcopter, the expression is $\dot{X} = f(X, u)$ Recall the general expression for linear system is $\dot{X} = AX + Bu$ where $X - state\ variables,\ u - input,\ A, B - system\ matrix$.*Khail* in his book mentioned the method linearlized the nonlinear system to obtain the general linear form[9]. The calculation of system matrix are given below:

$$A = \frac{\partial f}{\partial X}(X, u)|_{X=X^*, u=u^*},\ B = \frac{\partial f}{\partial u}(X, u)|_{X=X^*, u=u^*} \tag{12}$$

Where the X^*, u^u* indicates the equilibrium of the quadcopter which means hover status. For hover status,the linear position should all be zeros except displacement in z-direction and all linear velocity should be zeros as well as pitch,roll,yaw angles. The four input of rotor angular speed should maintain a constant value which produced enough thrust to counter drag and gravity and angular velocity for quadcopter should be zero.To simplify the input, define $\Omega_i^2 = u_i, i = 1, 2, 3, 4$. With adequate condition, equilibrium should be

$$X^* = \begin{bmatrix} (x1)^* \\ (x2)^* \\ (x3)^* \\ (x4)^* \end{bmatrix},\ (x1)^* = \begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix},\ (x2)^* = (x3)^* = (x4)^* = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix},\ u^* = \begin{bmatrix} u_1^* \\ u_2^* \\ u_3^* \\ u_4^* \end{bmatrix}$$

Where C_1, C_2, C_3 can be arbitrary constants. f indicates the 12 equations showed from Eq(8) to Eq(11):

$$\begin{aligned} f_1 &= x_4,\ f_2 = x_5,\ f_3 = x_6 \\ &\vdots \\ &\vdots \\ f_{12} &= \frac{1}{I_{zz}}\left(I_{xx}x_{10}x_{11} - I_{yy}x_{10}x_{11} + b(u_1 - u_2 + u_3 - u_4)\right) \end{aligned}$$

Using Jacobian matrix to obtain matrices A, B:

$$A = \left[\begin{array}{cccccc} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \cdots & \cdots & \frac{\partial f_1}{\partial x_{12}} \\ \frac{\partial f_2}{\partial x_1} & \cdots & & & & \\ \frac{\partial f_3}{\partial x_1} & \cdots & & & & \\ \vdots & & & & & \\ \vdots & & & & & \\ \frac{\partial f_{12}}{\partial x_1} & \cdots & & & & \frac{\partial f_{12}}{\partial x_{12}} \end{array}\right]\Bigg|_{X=X^*,u=u^*} \in R^{12\times 12} \tag{13}$$

$$B = \left[\begin{array}{cccc} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} & \frac{\partial f_1}{\partial u_3} & \frac{\partial f_1}{\partial u_4} \\ \vdots & & & \\ \vdots & & & \\ \frac{\partial f_{12}}{\partial u_1} & \cdots & & \frac{\partial f_{12}}{\partial u_4} \end{array}\right]\Bigg|_{X=X^*,u=u^*} \in R^{12\times 4} \tag{14}$$

Then let the **Python** to finish the laborious computing work(See code in Appendix).

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.5342 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.5342 & 0 & 0 & -9.81 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.5342 & -2.4525 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.0637 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.0637 & 0.0637 & 0.0637 & 0.0637 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.14 & 0 & -0.14 & 0 \\ 0 & 0.14 & 0 & -0.14 \\ 0.529 & -0.529 & 0.529 & -0.529 \end{bmatrix}$$

2.2 Controllability

Quadcopter system can be written as a Linear Time Invariant system(LTI):

$$\dot{X} = A \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \end{bmatrix} + B \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{15}$$

Controllability is concerned with whether or not the state of a state-space equation can be controlled from the input[10].Controllability is an essential part in discussing the internal structure of linear system.It is also needed in studying control and filtering problems.

The state equation is said to be controllable if for any initial state $X(t_i) = X_i$ and final state X_f, there exists an input that transfer X_i to X_f in a finite time. Checking the controllability of linear system can be achieved through computing the rank of controllability matrix,if it is full rank then the system can be concluded as controllable. Controllability matrix is given below:

$$\mathcal{C} = \begin{bmatrix} B & AB & \dots & A^{n-1}B \end{bmatrix}$$

In MATLAB, function *ctrb* aims at computing the controllability matrix. So the controllability proved through two line code: $C = ctrb(A, B); rank(C);$.According to MATLAB results, this system is controllable.

2.3 Stability

System are designed to perform some tasks or some or to process signals. If a system is not stable, the system may burn out, disintegrate,or saturate when a signal, no matter how small, is applied.Therefore,an unstable system is useless in practice and stability is a basic requirement for all systems. In addition to stability, system must meet other requirements, such as to track desired signals or to suppress noise, to be really useful in practice.

The Eq(15) is marginally stable if and only if all eigenvalues of A have zero or negative real parts and those with zero or negative real parts and those with zero real parts are simple roots of the minimal polynominal of A. With MATLAB, the eigenvalues of A shown below:

$$Eigenvalues = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -0.5342 \\ -0.5342 \\ -0.5342 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\text{Minimal Polynominal: } x^4 + \frac{2671x^3}{5000}$$

It is clearly to see that this system is not stable.In the next chapter, a designed controller will stabilize the system.

3 Controller design

3.1 Linear-quadratic regulator

To stabilize the quadcopter system, all eigenvalues should be negative or zero.One may wonder at this point how to select a set of desired eigenvalues.This depends on the performance criteria, such as rise time, setting time, overshoot,and others.

The theory of optimal control is concerned with operating a dynamic system at minimum cost. The case where the system dynamics are described by a set of linear differential equations and the cost is described by a quadratic function is called the LQ problem. One of the main results in the theory is that the solution is provided by the linear–quadratic regulator (LQR), a feedback controller whose equations are given below:

$$\begin{aligned} \dot{X} &= AX + Bu \, , u = -KX \\ \dot{X} &= AX - BKX = (A - BK)X \end{aligned} \tag{16}$$

where K indicates the feedback gain, so implemented as $u = -KX$. The bandwidth of the feedback system will be larger and the resulting system will be more susceptible to noise.One way to find the state feedback gain $K$to minimize the quadratic performance.

$$J = \int_0^\infty (X^T Q X + u^T R u + 2X^T N u) dt \tag{17}$$

The feedback control law that minimizes the value of the cost is$u = -KX$ where $K = R^{-1}(B^T P + N^T)$, and P is found by solving the continuous time algebraic Riccati equation:

$$A^T P + PA - (PB + N)R^{-1}(B^T P + N^T) + Q = 0 \tag{18}$$

To simplify the research, let me choose $Q = I_{12\times 12}, R = I_{4\times 4}$. Recall that one basic standard to choose Q, R is that all of them should be positive definite. With MATLAB build-in function **lqr**, feedback gain K attained and then it is time to check the stability

of modified system via eigenvalues:

$$Eigenvalues\ of\ (A-BK) = \begin{bmatrix} -0.4060+0.7499i \\ -0.4060-0.7499i \\ -0.0599+0.0000i \\ -0.8997+0.4991i \\ -0.8997-0.4991i \\ -0.5310+0.0000i \\ -0.7864+0.2218i \\ -0.7864-0.2218i \\ -0.6126+1.1457i \\ -0.6126-1.1457i \\ -1.0607+0.1598i \\ -1.0607-0.1598i \end{bmatrix}$$

All real parts of eigenvalues is negative, so this system is asymptotically stable.

3.2 Numerical simulation of system

3.2.1 Simulation of uncontrolled system

First of all, let us present the simulation of the system without any controller.

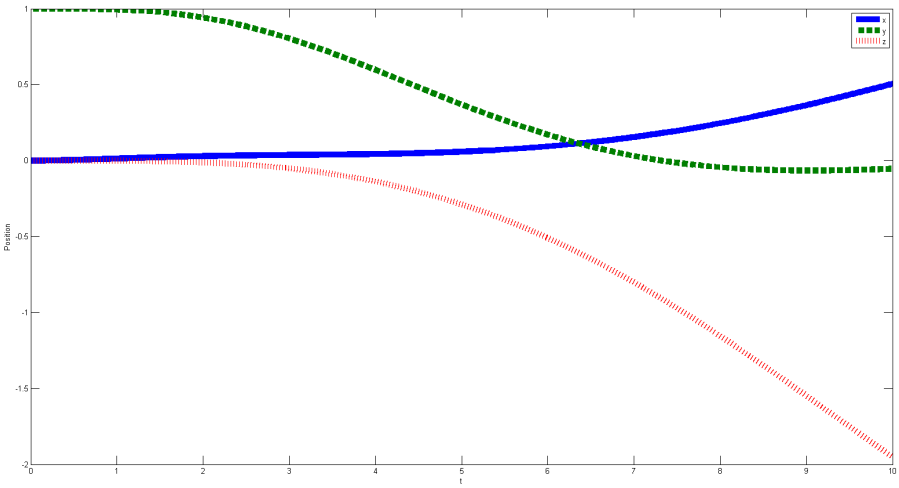


Figure 1: Position of quadcopter(without controller)
Blue-x,Green-y,Red-z

Position of quadcopter grows undoubtedly which is out of control.

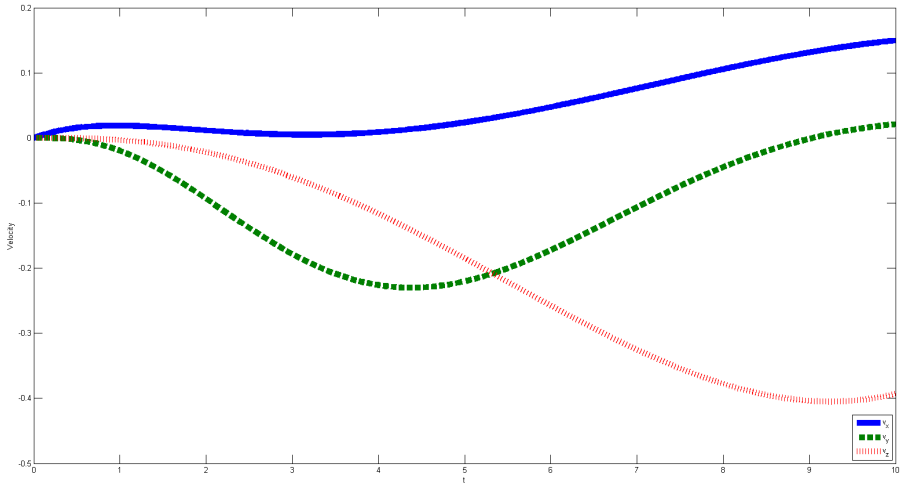


Figure 2: Velocity of quadcopter(without controller)
Blue-v_x,Green-v_y,Red-v_z

Similarly the velocity of quadcopter is impossible to stabilized in a specified time which is complete failure for system.Besides that increasing of velocity means higher needs for input which indirectly augmented the consumption of energy.

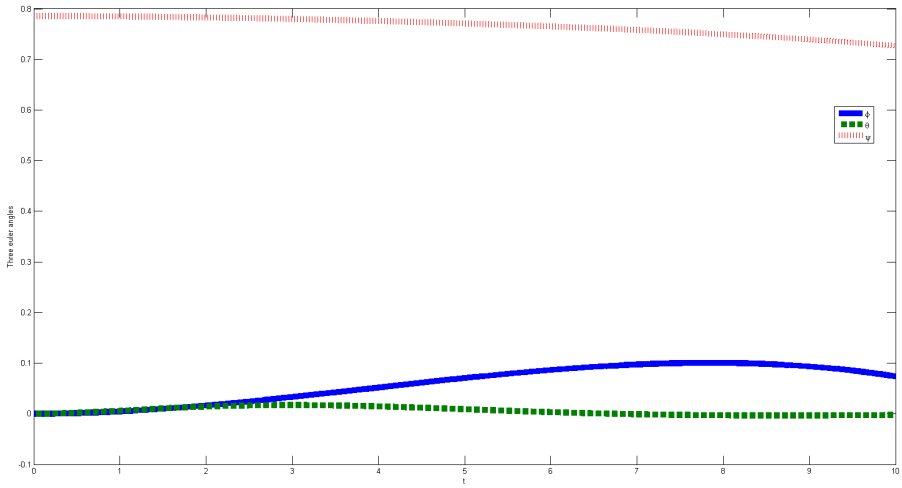


Figure 3: Three Euler angles of quadcopter(without controller)
Blue-ϕ,Green-θ,Red-ψ

To reach the equilibrium of quadcopter which means hover status of quadcopter, all three Euler angles should be zeros which means there is no roll,pitch,yaw for quadcopter.

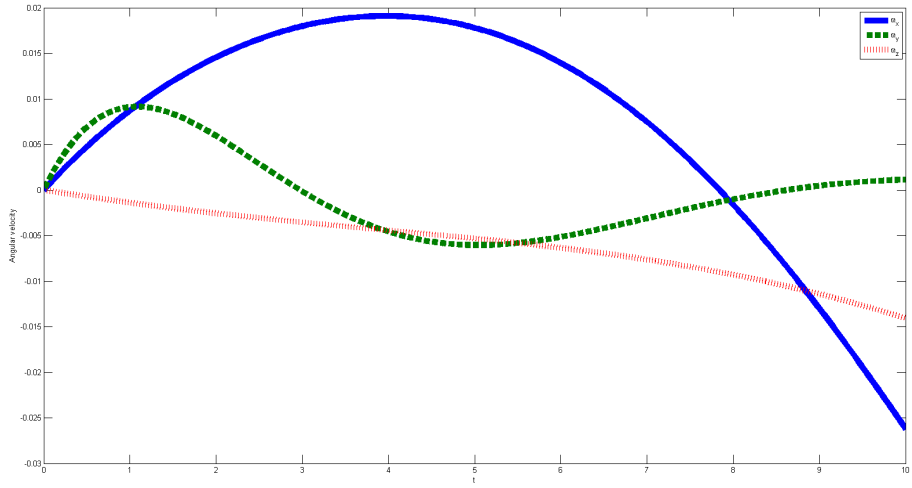


Figure 4: Angular velocity of quadcopter(without controller)
Blue-ω_x,Green-ω_y,Red-ω_z

Like equilibrium requirements for velocity, once the quadcopter reaches stable status the angular velocity should jumped in a specified boundary. And above figure apparently does not satisfy this criteria.

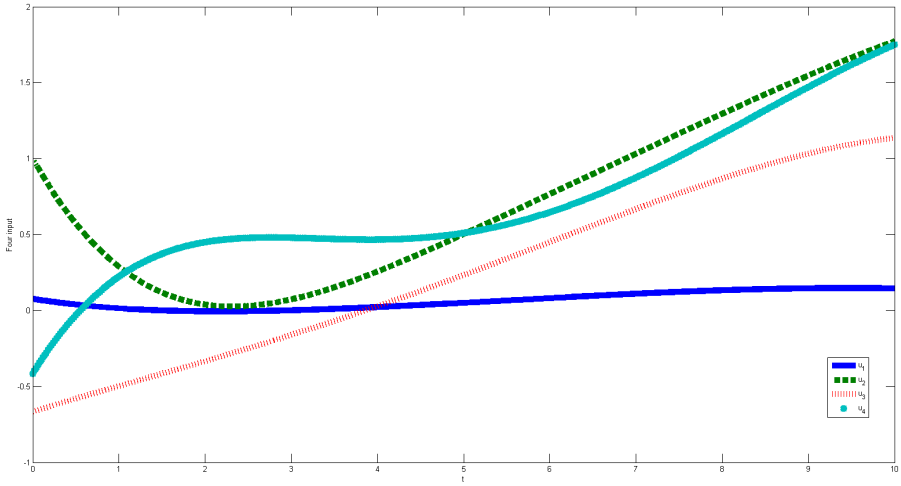


Figure 5: Inputs of quadcopter(without controller)
Blue-u_1,Green-u_2,Red-u_3,Cyan-u_4

It is easy to observe that, this system requires more inputs as times goes infinity. With this tendency, the system is extremely difficult to achieve the stables status.

3.2.2 Simulation of controlled system

Designing a controller through Linear-quadratic regulator and then simulates this controlled system.

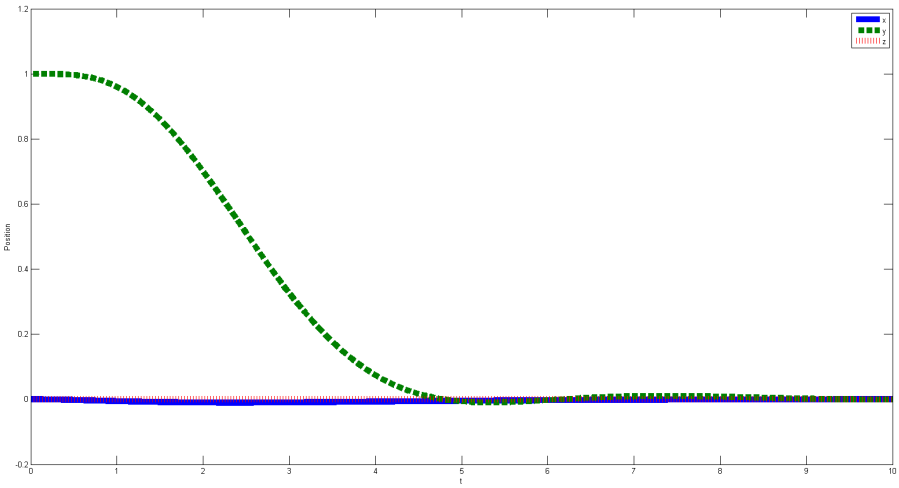


Figure 6: Position of quadcopter(LQR)
Blue-x,Green-y,Red-z

After applied LQR for quadcopter system, the position of quadcopter reaches original point after 6 seconds which can be seen as quadcopter hovers over the original points.

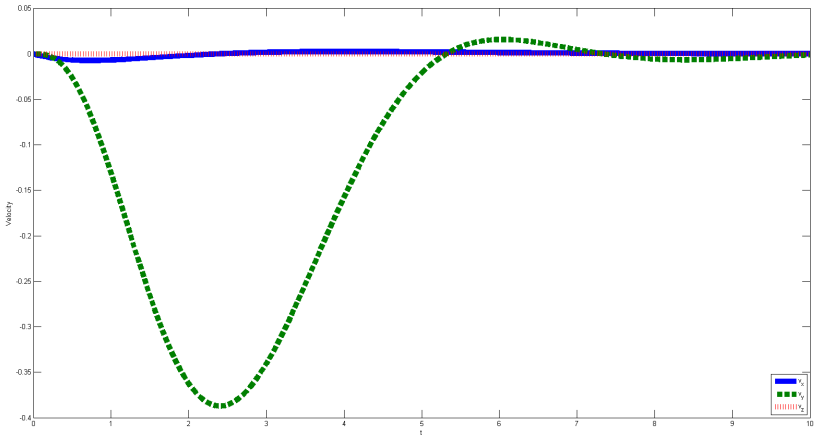


Figure 7: Velocity of quadcopter(LQR)
Blue-v_x,Green-v_y,Red-v_z

The time period between 5 second and 6 second, the quadcopter can be regraded as sustained a marginally stable status since the velocity oscillation is so small which can be ignored. After 8 seconds, the velocity of quadcopter becomes zeros,so the system goes asymptotically stable.

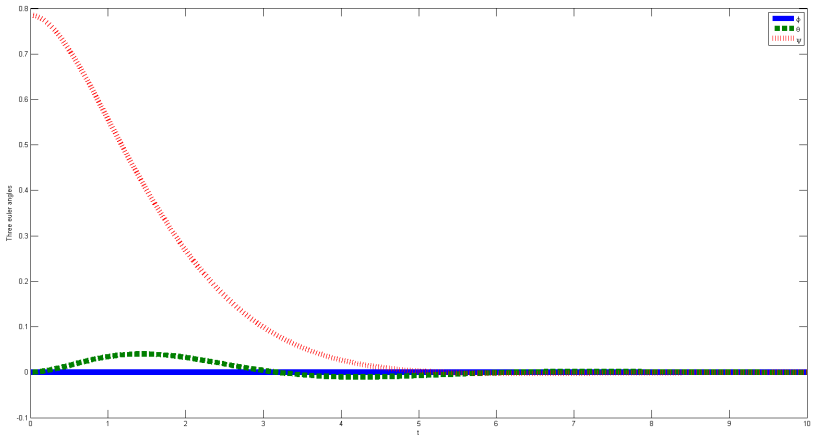


Figure 8: Three Euler angles of quadcopter(LQR)
Blue-ϕ,Green-θ,Red-ψ

After 6 seconds, all of roll angle, pitch angle, and yaw angle goes to zero which means equilibrium achieved. And this numerical result proved our assertion that during the equilibrium all Euler angles will become zero.

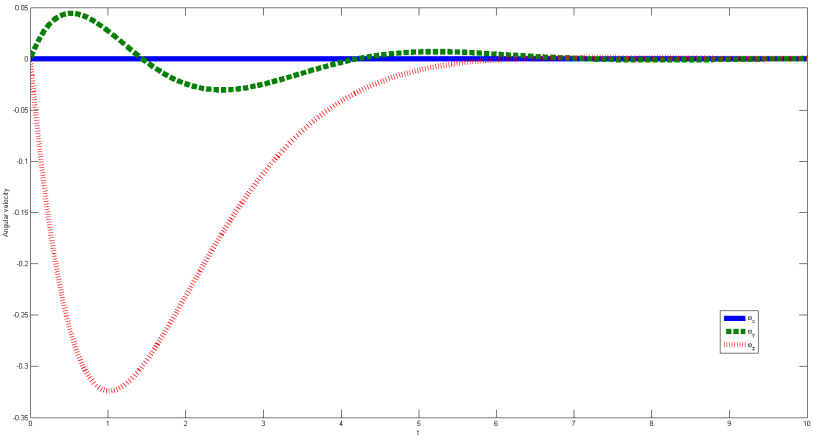


Figure 9: Angular velocity of quadcopter(LQR)
Blue-ω_x,Green-ω_y,Red-ω_z

Same as our physical analysis, during the equilibrium the angular velocity of quadcopter should be zeros which indicates that quadcopter is free of inclination of rolling,pitching,yawing.

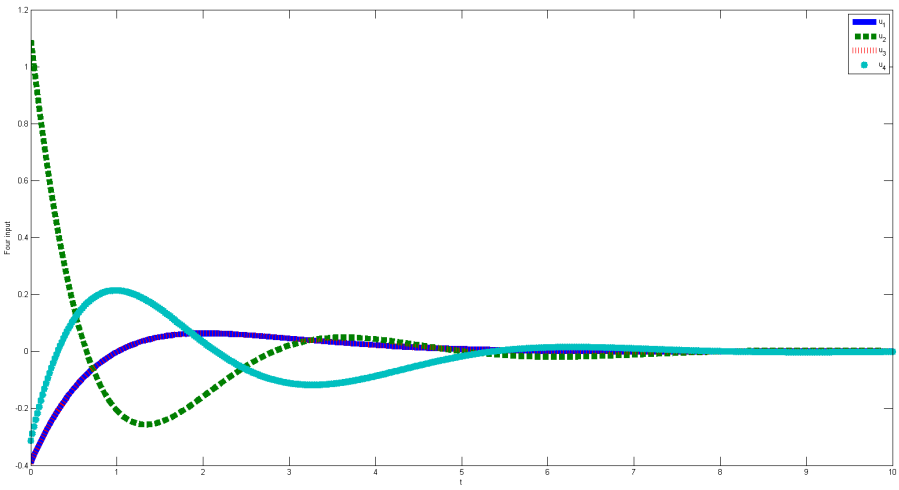


Figure 10: Inputs of quadcopter(LQR)
Blue-u_1,Green-u_2,Red-u_3,Cyan-u_4

After reached equilibrium, the input system goes to a constant column vector. Notice that change rate of u_2, u_3 almost identical to each other, so the combination of these two color lines become a "purple" line.

Appendix A LQR control via MATLAB

```
function quadcopter
%-------------------------------------------------------------------------
close all;
clear all;
global a b c K
%-------------------------------------------------------------------------
%system matrix A,B, since the complicated expression for A
%there will be row vectors of A first, and then whole matrix
a1 = [0 0 0 1 0 0 0 0 0 0 0 0];
a2 = [0 0 0 0 1 0 0 0 0 0 0 0];
a3 = [0 0 0 0 0 1 0 0 0 0 0 0];
a4 = [0 0 0 -0.5342 0 0 0 0 0 0 0 0];
a5 = [0 0 0 0 -0.5342 0 0 -9.81 0 0 0 0];
a6 = [0 0 0 0 0 -0.5342 -2.45 0 0 0 0 0];
a7 = [0 0 0 0 0 0 0 0 0 1 0 0];
a8 = [0 0 0 0 0 0 0 0 0 0 1 0];
a9 = [0 0 0 0 0 0 0 0 0 0 0 1];
a10 = [0 0 0 0 0 0 0 0 0 0 0 0];
a11 = [0 0 0 0 0 0 0 0 0 0 0 0];
a12 = [0 0 0 0 0 0 0 0 0 0 0 0];
A = [a1;a2;a3;a4;a5;a6;a7;a8;a9;a10;a11;a12];

a = 0.0637;
b = 0.14;
c = 0.529;

B = [0 0 0 0;0 0 0 0;0 0 0 0;
     a 0 0 0;0 0 0 0;a a a a;
     0 0 0 0;0 0 0 0;0 0 0 0;
     b 0 -b 0; %line 10
     0 b 0 -b; %line 11
     c -c c -c]; %line 12
%-------------------------------------------------------------------------
%check the controllability of quadcopter system:
C = ctrb(A,B);      %define the controllability matrix
r = rank(C);        %check the rank controllability matrix
if r >= 12
    disp('This system is controllable.')
end
%-------------------------------------------------------------------------
%check the stability of system through eigenvalues:
```

```
eig_A = eig(A);
syms s
minpoly(A,s);
%-------------------------------------------------------------------------
%quadratic cost function parameter:
Q = eye(12);
R = eye(4);

K = lqr(A,B,Q,R);       %obtaining optimal feedback gain through LQR
%-------------------------------------------------------------------------
% check the stability after lqr applied:
eig_A_BK = eig(A-B*K);
if real(eig_A_BK) < 0
   disp('After LQR applied,the system is stabilized');
end

%-------------------------------------------------------------------------
%initial condition:
x1_0 = [0 1 0]';
x2_0 = [0 0 0]';
x3_0 = [0 0 pi/4]';
x4_0 = [0 0 0]';
X0 = [x1_0;x2_0;x3_0;x4_0];
%-------------------------------------------------------------------------
%using ode45 to obtain
N=501;
t=linspace(0,10,N);
[t X] = ode45(@eom,t,X0);

%-------------------------------------------------------------------------
%data processing:
for i = 1:N
    x(i) = X(i,1);              %x-position
    y(i) = X(i,2);              %y-position
    z(i) = X(i,3);              %z-position
    vx(i) = X(i,4);             %velocity in x-direction
    vy(i) = X(i,5);             %velocity in y-direction
    vz(i) = X(i,6);             %velocity in z-direction
    phi(i) = X(i,7);            %roll angle
    theta(i) = X(i,8);          %pitch angle
    psi(i) = X(i,9);            %yaw angle
    wx(i) = X(i,10);            %angular velocity in x-direction
    wy(i) = X(i,11);            %angular velocity in y-direction
```

```
    wz(i) = X(i,12);              %angular velocity in z-direction
    u(i,1:4) = -K*X(i,1:12)';     %four input: Omega_i^2
end

%-------------------------------------------------------------------------
%mapping for each state variables:
figure;
plot(t,x,t,y,'--',t,z,':','linewidth',8);
xlabel('t');
ylabel('Position');
legend('x','y','z','location','northeast');

figure;
plot(t,vx,t,vy,'--',t,vz,':','linewidth',8);
xlabel('t');
ylabel('Velocity');
legend('v_x','v_y','v_z','location','southeast');


figure;
plot(t,phi,t,theta,'--',t,psi,':','linewidth',8);
xlabel('t');
ylabel('Three euler angles');
legend('\phi','\theta','\psi','location','northeast');


figure;
plot(t,wx,t,wy,'--',t,wz,':','linewidth',8);
xlabel('t');
ylabel('Angular velocity');
legend('\omega_x','\omega_y','\omega_z','location','northeast');

figure;
plot(t,u(1:501,1),t,u(1:501,2),'--',t,u(1:501,3),':',t,u(1:501,4),'*'...
    ,'linewidth',8);
xlabel('t');
ylabel('Four input');
legend('u_1','u_2','u_3',...
    'u_4','location','northeast');


save quadcopter;
evalin('base','load quadcopter');
```

```
end

%-------------------------------------------------------------------------
%constructing the equation of motion:
function X_dot = eom(t,X)
global a b c
x1 = X(1);x2 = X(2);x3 = X(3);x4 = X(4);x5 = X(5);x6 = X(6);
x7 = X(7);x8 = X(8);x9 = X(9);x10 = X(10);x11 = X(11);x12 = X(12);

u = control(t,X);

x1_dot = x4;x2_dot = x5;x3_dot = x6;x4_dot = -0.5342*x4+a*u(1);
x5_dot = -0.5342*x5-9.81*x8;x6_dot = -0.5432*x6-2.4525*x7;
x7_dot = x10;x8_dot = x11;x9_dot = x12;
x10_dot = b*u(1)-b*u(3);
x11_dot = b*u(2)-b*u(4);
x12_dot = c*u(1)-c*u(2)+c*u(3)-c*u(4);

X_dot = [x1_dot;x2_dot;x3_dot;x4_dot;x5_dot;x6_dot;x7_dot;x8_dot;x9_dot;...
    x10_dot;x11_dot;x12_dot];

end

%-------------------------------------------------------------------------
%constructing input u:
function u = control(t,X)
global K
u = -K*X;
end
```

Appendix B Algebra Calculation via Python

```
import sympy    #using sympy library to symbols calculation
#turn on Latex to express the answer more clearly:
sympy.init_printing()
from sympy.matrices import Matrix, eye, zeros, ones, diag, GramSchmidt
#numerical implementation of algebra function:
from sympy.utilities.lambdify import lambdify

#variables for computing:
(tau_phi,tau_theta,tau_psi,I_xx,I_yy,I_zz,omega_x,omega_y,omega_z,phi,theta,p
sympy.symbols('tau_phi tau_theta \
tau_psi I_xx I_yy \ I_zz omega_x omega_y omega_z phi theta psi')

#variables for input:
(Omega_1,Omega_2,Omega_3,Omega_4,g,m,k_d,x,y,z,l,k,b) =
sympy.symbols('Omega_1 Omega_2 Omega_3 Omega_4 g m k_d x y z l k b')

(xdot,ydot,zdot)=sympy.symbols('xdot ydot zdot')

def hat(x):
    return Matrix([[0,-x[2],x[1]],[x[2],0,-x[0]],[-x[1],x[0],0]])

#inertia matrix:
In_Ma = Matrix([[I_xx,0,0],[0,I_yy,0],[0,0,I_zz]])
#torque:
Tau = Matrix([[tau_phi],[tau_theta],[tau_psi]])
#angular velocity:
Omega=Matrix([[omega_x],[omega_y],[omega_z]])

#angular acceleration:
dot_Omega = In_Ma.inv('LU')*( Tau - hat(Omega)*(In_Ma *Omega) )

dot_Omega

# conversion matrix from derivativesi of euler angles to angular velocity
C = Matrix([[1,0,-sympy.sin(theta)],[0,sympy.cos(phi),\
sympy.cos(theta)*sympy.sin(phi)],[0,-sympy.sin(phi),sympy.cos(theta)*sympy.cos
C

# thrust under body-fixed-frame
T_b = k*Matrix([[0],[0],[(Omega_1)**2+(Omega_2)**2+(Omega_3)**2\
                         +(Omega_4)**2]])
```

```
T_b

# constructing rotation matrix (3-1-3)
R = zeros(3,3)
R[0,0] = sympy.cos(phi)*sympy.cos(psi)-sympy.cos(theta)*\
sympy.sin(phi)*sympy.sin(psi)
R[0,1] = -sympy.cos(psi)*sympy.sin(phi)-sympy.cos(phi)*\
sympy.cos(theta)*sympy.sin(psi)
R[0,2] = sympy.sin(theta)*sympy.sin(psi)
R[1,0] = sympy.cos(theta)*sympy.cos(psi)*sympy.sin(phi)+\
sympy.cos(phi)*sympy.sin(psi)
R[1,1] = sympy.cos(phi)*sympy.cos(theta)*sympy.cos(psi)-\
sympy.sin(phi)*sympy.sin(psi)
R[1,2] = -sympy.cos(psi)*sympy.sin(theta)
R[2,0] = sympy.sin(phi)*sympy.sin(theta)
R[2,1] = sympy.cos(phi)*sympy.sin(theta)
R[2,2] = sympy.cos(theta)
R

#gravity:
G = Matrix([[0],[0],[-g]])
G

# drag force:
F_d = -k_d*Matrix([[xdot],[ydot],[zdot]])
F_d

# calculate the dot_x2:
dot_x2 = G+R*T_b/m+F_d/m
dot_x2

(x_10,x_11,x_12) = sympy.symbols('x_10 x_11 x_12')
x4 = Matrix([[x_10],[x_11],[x_12]])
# calculate dot_x3
dot_x3 = C.inv('LU')*x4
dot_x3

dot_x4 = zeros(3,1)
tau_phi = l*k*((Omega_1)**2-(Omega_3)**2)
tau_theta = l*k*((Omega_2)**2-(Omega_4)**2)
tau_psi = b*((Omega_1)**2-(Omega_2)**2+(Omega_3)**2-(Omega_4)**2)

dot_x4[0] = 1/I_xx*(-I_zz*omega_y*omega_z+I_yy*omega_y*omega_z +tau_phi )
```

```
dot_x4[1] = 1/I_yy*(I_zz*omega_x*omega_z - I_xx*omega_x*omega_z+tau_theta)
dot_x4[2] = 1/I_zz*(I_xx*omega_x*omega_y-I_yy*omega_x*omega_y+tau_psi)
#calculate the dot_x4:
dot_x4

f4 = dot_x2[0] #f4
f5 = dot_x2[1] #f5
f6 = dot_x2[2] #f6
# for some obvious value,directly put into matrix A
f4_4 = -0.534/0.468 # paritial f4 w.r.t x4
f4_7 = f4_8 = f4_9 =f4_10 = f4_11 = f4_12 = 0

f5_8 = sympy.diff(f_5,theta) # paritial f5 w.r.t x8(theta)
f5_9 = sympy.diff(f_5,psi) # paritial f5 w.r.t x9(psi)

f7 = dot_x3[0]
f8 = dot_x3[1]
f9 = dot_x3[2]

f7_7 = sympy.diff(f7,phi) # paritial f7 w.r.t x7(phi)
f7_8 = sympy.diff(f7,theta) # paritial f7 w.r.t x8(theta)
f7_9 = sympy.diff(f7,psi) # paritial f7 w.r.t x9(psi)

f7_10 = sympy.diff(f7,x_10)
f7_11 = sympy.diff(f7,x_11)
f7_12 = sympy.diff(f7,x_12)

f7_7_l = lambdify((x_11,x_12,phi,theta),f7_7)
f7_7_l(0,0,0,0) # substitute x_11,x_12,phi,theta with equalibrium

f7_8_l = lambdify((x_11,x_12,phi,theta),f7_8)
f7_8_l(0,0,0,0)

f8_7 = sympy.diff(f8,phi) # paritial f8 w.r.t x7(phi)
f8_8 = sympy.diff(f8,theta) # paritial f8 w.r.t x8(theta)
f8_9= sympy.diff(f8,psi) # paritial f8 w.r.t x9(psi)

f8_7_l = lambdify((x_10,x_11,x_12,phi,theta),f8_7)
f8_7_l(0,0,0,0,0)

f8_8_l = lambdify((x_10,x_11,x_12,phi,theta),f8_8)
f8_8_l(0,0,0,0,0)
```

```
f8_10 = sympy.diff(f8,x_10)
f8_11 = sympy.diff(f8,x_11)
f8_11_l  = lambdify((x_10,x_11,x_12,phi,theta),f8_11)
f8_11_l(0,0,0,0,0)
f8_12 = sympy.diff(f8,x_12)
f8_12

f9_7 = sympy.diff(f9,phi)
f9_8 = sympy.diff(f9,theta)
f9_9 = sympy.diff(f9,psi)

f9_7_l = lambdify((x_10,x_11,x_12,phi,theta),f9_7)
f9_7_l(0,0,0,0,0)

f9_8_l = lambdify((x_10,x_11,x_12,phi,theta),f9_8)
f9_8_l(0,0,0,0,0)

f9_10 = sympy.diff(f9,x_10)
f9_11 = sympy.diff(f9,x_11)
f9_12 = sympy.diff(f9,x_12)
f9_12_l = lambdify((x_10,x_11,x_12,phi,theta),f9_12)
f9_12_l(0,0,0,0,0)

f10 = dot_x4[0]
f11 = dot_x4[1]
f12 = dot_x4[2]

f10_10 = sympy.diff(f10,omega_x) #partial deriative w.r.t x10
f10_11 = sympy.diff(f10,omega_y)
f10_12 = sympy.diff(f10,omega_z)

f11_10 = sympy.diff(f11,omega_x)
f11_11 = sympy.diff(f11,omega_y)
f11_12 = sympy.diff(f11,omega_z)

f12_10 = sympy.diff(f12,omega_x)
f12_11 = sympy.diff(f12,omega_y)
f12_12 = sympy.diff(f12,omega_z)
```

References

[1] S. Bouabdallah, A. Noth, and R. Siegwart, "PID vs LQ control techniques applied to an indoor micro quadrotor," IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3, pp. 2451–2456, 2004.

[2] T. Bresciani, "Modelling, Identification and Control of a Quadrotor Helicopter," in Automatic Control. vol. Master of Science Sweden: Lund, 2008.

[3] Elruby, A. Y., et al. "Dynamic modeling and control of quadrotor vehicle." Proceedings of the 15th Int. AMME Conference. Vol. 29. 2012.

[4] Gibiansky, Andrew. "Quadcopter dynamics, simulation, and control." Andrew Gibiansky:: Math[Code] 21 (2012).

[5] G. M. Hoffmann, H. Huang, S. L. Waslander, and C. J. Tomlin, "Quadrotor helicopter flight dynamics and control: Theory and experiment," Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit, Aug. 2007.

[6] H. Huang, G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin, "Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering," IEEE International Conference on Robotics and Automation, pp. 3277–3282, May 2009.

[7] Diebel, James. "Representing attitude: Euler angles, unit quaternions, and rotation vectors." Matrix 58.15-16 (2006): 1-35.

[8] Luukkonen, Teppo. "Modelling and control of quadcopter." Independent research project in applied mathematics, Espoo (2011).

[9] Khalil, Hassan K. Noninear Systems. Prentice-Hall, New Jersey, 1996.

[10] Chen, Chi-Tsong. Linear system theory and design. Oxford University Press, Inc., 1995.