State Feedback and Observer-Based Feedback Control of Linearized Attitude Quadrotor Model

Stephen Brutch
Embry-Riddle Aeronautical University
Daytona Beach, United States
BRUTCHS@my.erau.edu

***Abstract*—Unmanned aerial vehicles are becoming a more widely used technology for different set of applications. Quadcopters have applications to construction, photography and security areas for their vertical take-off and land and their ability to hover. Quadcopter control in general faces challenges due to it being an under actuated system. For this paper two feedback control architectures are designed to handle the under actuation.**

I. INTRODUCTION

Quadcopter control in general faces challenges due to it being an under actuated system in that 6 states are studied in this paper yet only 4 control actuators are used to control those 6 states. For this paper a nonlinear model for a quadcopter will be linearized about its hover equilibrium point. A state feedback and observer based feedback control system architectures will be developed to control the quadcopter's attitude from a set of non-zero initial condition to its attitude and angular rates to regulate the system to hover conditions

II. PROBLEM STATEMENT

For this project a linear model of the quadcopter must be found first before the two control architectures could be studies. A nonlinear model of a quadcopter was first obtained, and ignoring gyroscope perturbations they are, [3]

$$\begin{cases} \ddot{\phi} = \dot{\theta}\dot{\Psi}\left(\frac{I_y - I_z}{I_x}\right) - \frac{l}{I_x} b(\Omega_4^2 - \Omega_2^2) \\ \ddot{\theta} = \dot{\phi}\dot{\Psi}\left(\frac{I_z - I_x}{I_y}\right) + \frac{l}{I_y} b(\Omega_3^2 - \Omega_1^2) \\ \ddot{\Psi} = \dot{\phi}\dot{\theta}\left(\frac{I_x - I_y}{z}\right) + \frac{1}{I_z} d(\Omega_2^2 + +\Omega_4^2 + -\Omega_1^2 + -\Omega_3^2) \end{cases} \quad (1)$$

Where:

Symbol	Definition	Value(s)
ϕ	Roll	
θ	Pitch	
Ψ	Yaw	
I_x	Moment of Inertia	0.001617 $kg * m^2$
I_y	Moment of Inertia	0.001345 $kg * m^2$
I_z	Moment of Inertia	0.002755 $kg * m^2$
l	Arm Length	0.125 m
b	Thrust Factor	
d	Drag Factor	
$\Omega_{1,2,3,4}$	Rotor Speeds	

For this project an actual quadcopter was not obtained, thus the physical characteristics used for the purposes of completing this project are from [1].

A simplification to the nonlinear dynamics equations before linearizing is to label the input vector, U, as such to avoid coupling in the equations [2],

$$U = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} \quad (2)$$

The state vector, X, can also be labeled as,

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} \phi \\ \theta \\ \Psi \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \quad (3)$$

A. Nonlinear Model

The original nonlinear equations (1) now becomes,

$$\begin{cases} \ddot{\phi} = x_5 x_6 \left(\frac{I_y - I_z}{I_x}\right) - \frac{l}{I_x} b(U_4 - U_2) \\ \ddot{\theta} = x_4 x_6 \left(\frac{I_z - I_x}{I_y}\right) + \frac{l}{I_y} b(U_3 - U_1) \\ \ddot{\Psi} = x_4 x_5 \left(\frac{I_x - I_y}{z}\right) + \frac{1}{I_z} d(U_2 + U_4 - U_1 - U_3) \end{cases} \quad (4)$$

The F function which will be used for the Jacobian linearization will be,

$$F(x,u) = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{bmatrix} = \begin{bmatrix} x_4 \\ x_5 \\ x_6 \\ x_5 x_6 \left(\frac{I_y - I_z}{I_x}\right) - \frac{l}{I_x} b(U_4 - U_2) \\ x_4 x_6 \left(\frac{I_z - I_x}{I_y}\right) + \frac{l}{I_y} b(U_3 - U_1) \\ x_4 x_5 \left(\frac{I_x - I_y}{z}\right) + \frac{1}{I_z} d(U_2 + U_4 - U_1 - U_3) \end{bmatrix} \quad (5)$$

The method of forming the linearized state-space matrices from the Jacobian is,

$$A = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_6} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_6}{\partial x_1} & \cdots & \frac{\partial f_6}{\partial x_6} \end{bmatrix} \quad (6)$$

$$B = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_1}{\partial u_4} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_6}{\partial u_1} & \cdots & \frac{\partial f_6}{\partial u_4} \end{bmatrix} \quad (7)$$

$$C = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \frac{\partial h_1}{\partial x_6} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_6}{\partial x_1} & \cdots & \frac{\partial h_6}{\partial x_6} \end{bmatrix} \quad (8)$$

$$D = \begin{bmatrix} \frac{\partial h_1}{\partial u_1} & \cdots & \frac{\partial h_1}{\partial u_4} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_6}{\partial u_1} & \cdots & \frac{\partial h_6}{\partial u_4} \end{bmatrix} \quad (9)$$

The output that was selected for this model was made under the assumption that onboard the quadcopter there are sensors that can directly track the 3 angles but no sensors for the angular rates,

$$h = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad (10)$$

Each of the A, B, C and D matrices once formed from taking the partial derivatives are then evaluated at equilibrium conditions which for this study are perfect hover conditions,

$$X_{eq} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \qquad U_{eq} = \begin{bmatrix} U_{1,eq} \\ U_{2,eq} \\ U_{3,eq} \\ U_{4,eq} \end{bmatrix} \quad (11)$$

B. Linearized Model

The final linearized state-space matrices are as follows,

$$A = \begin{bmatrix} 0_3 & I_3 \\ 0_3 & 0_3 \end{bmatrix} \quad (12)$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -77.30 & 0 & 77.30 \\ -92.94 & 0 & 92.94 & 0 \\ -362.97 & 362.97 & -362.98 & 362.97 \end{bmatrix} \quad (14)$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (15)$$

$$D = [0_{6x4}] \quad (16)$$

The transfer function can be obtained by,

$$H(s) = C(SI_6 - A)^{-1}B + D \quad (17)$$

The matrix of transfer functions are attached in the appendix.

C. Controllability and Observability

From the linearized state-space matrices, the controllability and observability matrix can be obtained. The controllability matrix for this system can be found by,

$$P_c = [B \quad AB \quad A^2B \quad A^3B \quad A^4B] \quad (18)$$

To be able to study state-feedback control, the controllability matrix must be full rank, invertible or non-zero determinant to ensure a K gain exists such that $u(t) = -Kx(t)$. For this model the controllability matrix was found to be full rank. The observability matrix can be formed by,

$$P_o = \begin{bmatrix} C \\ CA \\ CA^2 \\ CA^3 \\ CA^4 \\ CA^5 \end{bmatrix} \quad (19)$$

To be able to study observer based feedback, the observability matrix must be full rank to ensure an L (observer) gain exists. For this model the observability matrix was found to be full rank. Both matrices are attached in the appendix.

D. Stability of the Open Loop System

To study the stability of the open loop system the eigenvalues of the linearized A matrix will be found. This can be done by finding the characteristic equation and setting it equal to zero by,

$$|A - \lambda I| = 0 \ \rightarrow \lambda^6 = 0 \rightarrow \lambda = 0,0,0,0,0,0 \quad (20)$$

The algebraic multiplicity is 6 for the zero eigenvalue because it is repeated 6 times. The geometric multiplicity can be found by

$$G_i = n - rank(\lambda I - A) \quad (21)$$

For this problem λ is equal to zero and n is the size of the A matrix, so the geometric multiplicity can be simplified down to,

$$G_i = n - rank(A) = 6 - 3 = 3 \quad (22)$$

The geometric multiplicity is less than the algebraic multiplicity meaning the zero eigenvalue is non semi simple thus the system is unstable. Seen in Section III the open loop linearized system will be seen to be neutrally stable in that the system does not return to its equilibrium state but its response will continue to act based off of its initial conditions.

E. Stabilizability and Detectability

The model is completely controllable and completely observable thus the system is also stabilizable and detectable.

III. Control and Observer Design

The control objective is to regulate the quadcopter to stable hover conditions from a set of non-zero initial conditions.

A. State Feedback Control

The general form of the equation to solve for a state feedback gain is,

$$[\lambda_{di} I - A \quad B]\bar{\phi} = \bar{0} \tag{20}$$

The amount of columns in the matrix above is based off of the linearized A and B state-space matrices. The matrix has 10 columns thus $\bar{\phi}$ will have 10 components labeled $\phi_1, \phi_2, \ldots \phi_{10}$ in a 10x1 column vector. These two matrices can be multiplied together and the systems of equations obtained are,

$$\lambda_{di}\phi_1 - \phi_4 = 0 \tag{21}$$

$$\lambda_{di}\phi_2 - \phi_5 = 0 \tag{22}$$

$$\lambda_{di}\phi_3 - \phi_6 = 0 \tag{23}$$

$$77.304\phi_{10} - \phi_8 + \lambda_{di}\phi_4 = 0 \tag{24}$$

$$92.937\phi_{10} - 92.937\phi_7 + \lambda_{di}\phi_5 = 0 \tag{25}$$

$$362.98\phi_{10} - 362.98\phi_7 - 362.98\phi_9 + 362.98\phi_{10} + \lambda_{di}\phi_6 \tag{26}$$

There are above a set of 6 equations and 10 variables, thus there are 4 free variables. $\phi_1, \phi_2, \phi_3, \phi_7$ will be labeled as a, b, c, d. The general form of the solution can be seen below,

$$\bar{\phi} = \begin{bmatrix} a_i \\ b_i \\ c_i \\ a_i\lambda_{di} \\ b_i\lambda_{di} \\ c_i\lambda_{di} \\ d_i \\ d_i + 0.0065a_i\lambda_{di}^2 - 0.0054b_i\lambda_{di}^2 - 0.0014c\lambda_{di}^2 \\ d - 0.0108b\lambda_{di}^2 \\ d - 0.0065a_i\lambda_{di}^2 - 0.0054b_i\lambda_{di}^2 - 0.0014c\lambda_{di}^2 \end{bmatrix} \tag{27}$$

This selection of variables was selected to result in such a way a solution can be obtained. λ_{di} are each of the desired eigenvalues, where they were selected as $\lambda_{di} = [-1, -1, -2, -3, -4, -6]$. 6 iterations were performed, were each iteration was done with each individual eigenvalue. Subsequently each a,b,c,d in each iteration were labeled as a1,…,a6 and b1,…,b6 and c1,…,c6 and d1,…,d6 were the subscript number are related to each iteration. Lastly for each iteration $\phi_1, \ldots \phi_6$ are labeled as Ψ_i and $\phi_7, \ldots \phi_{10}$ are labeled as $K\Psi_i$. The details are seen in the appendix. At the end of 6 iterations, the Ψ matrix can be constructed as,

$$\Psi = [\Psi_1 \quad \Psi_2 \quad \Psi_3 \quad \Psi_4 \quad \Psi_5 \quad \Psi_6] \tag{27}$$

$$\Psi = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \\ b_1 & b_2 & b_3 & b_4 & b_5 & b_6 \\ c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \\ -a_1 & -2a_2 & -3a_3 & -4a_4 & -a_5 & -6a_6 \\ -b_1 & -2b_2 & -3b_3 & -4b_4 & -b_5 & -6b_6 \\ -c_1 & -2c_2 & -3c_3 & -4c_4 & -c_5 & -6c_6 \end{bmatrix} \tag{28}$$

A state feedback gain, K, will exist such that the above Ψ matrix is invertible. To do such coefficient values for each of the letters are to be selected such that the matrix is invertible. Due to the amount of numbers to hand pick select, a MATLAB code was developed where a set of random integers from -10 to 10 were selected and substituted into the Ψ matrix until the condition of invertibility was reached. That final matrix is,

$$\Psi = \begin{bmatrix} 5 & 1 & -6 & 9 & -5 & 8 \\ -6 & -3 & -8 & 3 & -6 & -9 \\ 4 & -3 & 3 & -2 & 3 & -10 \\ -5 & -2 & 18 & -36 & 5 & -48 \\ 6 & 6 & 24 & -12 & 6 & 54 \\ -4 & 6 & -9 & -8 & -3 & 60 \end{bmatrix} \tag{29}$$

The same process to construct the Ψ matrix was done to construct the KΨ matrix as seen in equation 27. The simplified KΨ matrix with the coefficients previously solved for can be seen below,

$$\text{K}\Psi = \begin{bmatrix} 8 & 6 & 5 & 6 & -2 & 2 \\ 8.06 & 6.10 & 5.0 & 6.72 & -2.0 & 6.10 \\ 8.06 & 6.13 & 8.77 & 5.43 & -1.93 & 5.48 \\ 7.99 & 6.05 & 5.69 & 4.85 & -1.93 & 2.38 \end{bmatrix} \tag{30}$$

The matrix K gain can be found by,

$$K = \text{K}\Psi * \Psi^{-1} = \begin{bmatrix} 0.98 & -0.98 & -1.90 & -0.13 & 0.15 & -0.80 \\ 0.95 & -0.93 & -1.85 & -0.16 & 0.19 & -0.77 \\ 0.87 & -0.76 & -2.12 & -0.23 & 0.38 & -1.03 \\ 1.00 & -0.96 & -1.87 & -0.09 & 0.17 & -0.78 \end{bmatrix} \tag{31}$$

This control law will satisfy $u(t) = -Kx(t)$ where x(t) is from equation 3. U(t) will become,

$$U(t) = \begin{bmatrix} 0.98 & -0.98 & -1.90 & -0.13 & 0.15 & -0.80 \\ 0.95 & -0.93 & -1.85 & -0.16 & 0.19 & -0.77 \\ 0.87 & -0.76 & -2.12 & -0.23 & 0.38 & -1.03 \\ 1.00 & -0.96 & -1.87 & -0.09 & 0.17 & -0.78 \end{bmatrix} \begin{bmatrix} \phi \\ \theta \\ \Psi \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\Psi} \end{bmatrix} \tag{32}$$

B. Observer Based Feedback

The general form to solve for a state observer feedback gain is,

$$\left[\mu_{\text{di}} - \text{A}^{\text{T}} \quad C^T\right]\bar{\phi} = \bar{0} \tag{33}$$

The amount of columns in the matrix above is 12, thus the ϕ matrix will be a 12x1 column vector of 12 components labeled $\phi_1, \ldots \phi_{12}$.The first 6 components are labeled as $\bar{\Psi}_i$ and the last 6 components will be labeled as $L^T\Psi_i$. The formation of these matrices and solving for them takes the same form as the process from equation (21) to (30). The details can be seen in the appendix. The observer gain (L) can be found by,

$$L = \left((L^T \Psi)(\Psi)\right)^T = \begin{bmatrix} 26.20 & 8.70 & -50.15 & 0 & 0 & 0 \\ 21.43 & 14.51 & -49.23 & 0 & 0 & 0 \\ -17.15 & -6.87 & 39.13 & 0 & 0 & 0 \\ 50.62 & 19.34 & -106.64 & 0 & 0 & 0 \\ 80.74 & 39.39 & -183.90 & 0 & 0 & 0 \\ -0.21 & -0.81 & 5.36 & 0 & 0 & 0 \end{bmatrix} \quad (34)$$

IV. Results and Discussion

The solution obtained in section III, their results in the form of simulations performed in MATLAB and Simulink solved with ODE45. The linearized system's controller will be applied to the state feedback controller as well as the observer based feedback controller. Lastly the same controller will be applied to the nonlinear model. For the following controllers the initial nonzero initial conditions were set as,

$$\bar{x}_0 = \begin{bmatrix} 0.09 \\ 0.09 \\ 0.09 \\ 3 \\ 3 \\ 3 \end{bmatrix}, \hat{x}_0 = \begin{bmatrix} 0.05 \\ 0.05 \\ 0.05 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (35)$$

Where the first 3 components of the initial states and initial estimated states relate to the angle in radians and the last 3 relate to the angular rate in rad/s. The results can be seen below, see appendix for MATLAB code and Simulink models,

A. *Open Loop system Response*

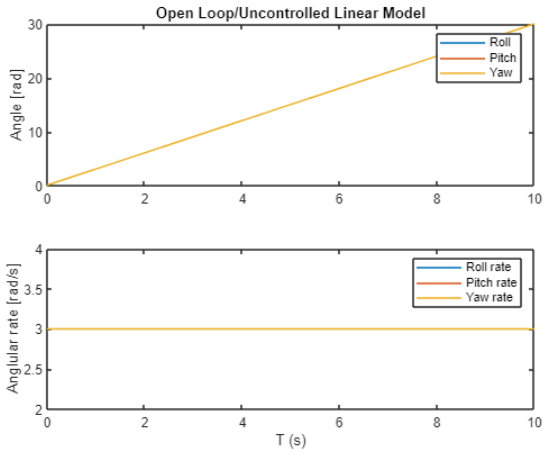


Figure 1Uncontrolled system Response

B. *State Feedback Control*

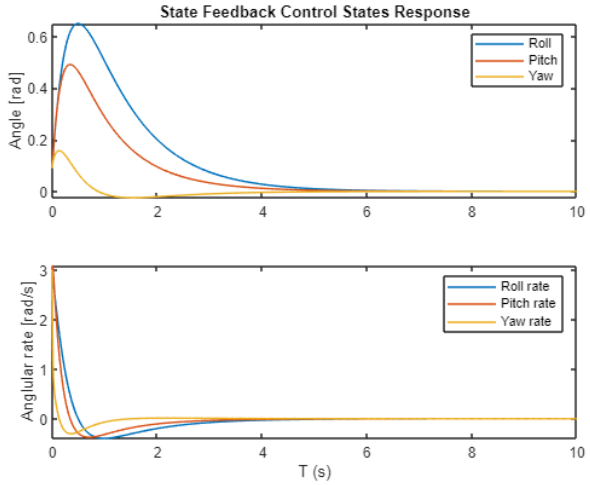


Figure 2 State Response of state feedback controller

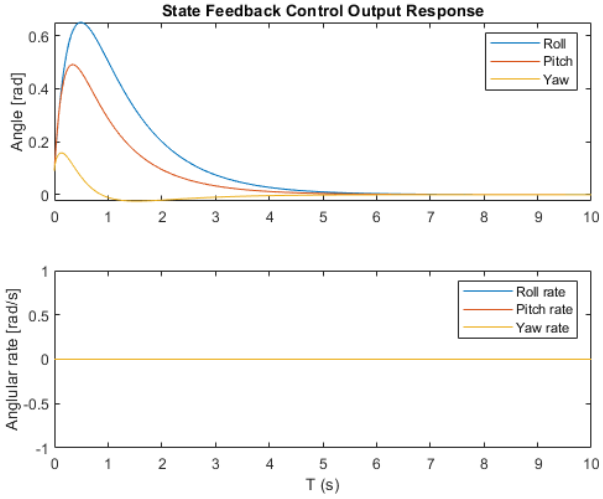


Figure 3 Output Response of state feedback controller

C. Observer Based Feedback Control

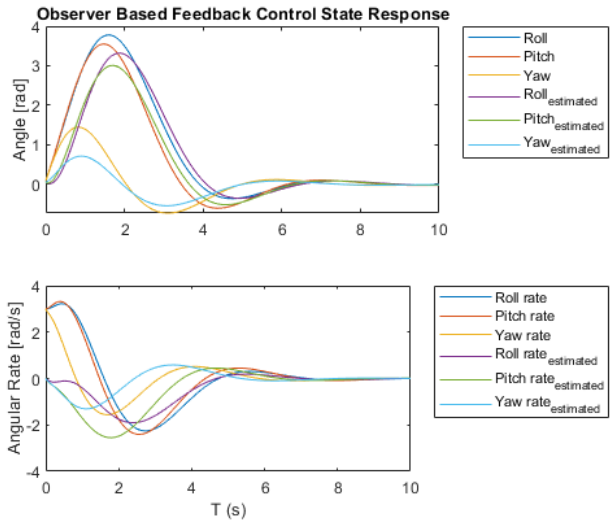


Figure 4 State response of observer based feedback control

D. Same Controller applied to the Nonlinear Model

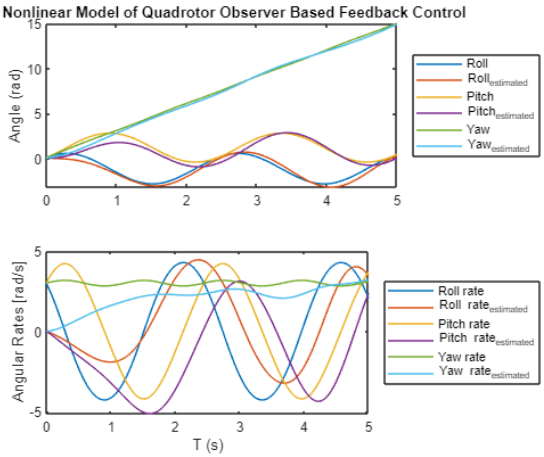


Figure 5 Nonlinear model response with linear controller

The linearized system's response is improved in that the system's uncontrolled response will continue to grow based off of its initial angular rates whereas the state feedback controller and observer based feedback control is able to regulate the system to zero. This is can be seen by,

$$A - BK = Hurwitz \tag{35}$$

$$A - LC = Hurwitz \tag{36}$$

V. Conclusions

The open loop uncontrolled system was found to be neutrally stable in that the systems response would continue to grow based off of its initial conditions. The use of the 2 controllers, state feedback controller and observer based feedback control was able to regulate the system to zero due to A-BK and A-LC are Hurwitz/stable matrix. This was done through pole placement where the desired poles were selected to be stable eigenvalues and the pole placement method would push the unstable eigenvalues of the A matrix to the stable eigenvalues. The controller was then applied to the nonlinear model where the controller for any set of small initial conditions was not able to control the system. The estimated states were found to converge to the actual states.

Acknowledgment

The author would like to thank Dr. Nazari for his teachings of the class and assistance on the project.

References

[1] A. Cuenca, "Design and construction of a Quadcopter and implementation of a stability controller in an autonomous flight algorithm application" Bogota,Colombia

[2] P. Wang, Z. Man, Z. Cao, J. Zheng and Y. Zhao, "Dynamics modelling and linear control of quadcopter," 2016 International Conference on Advanced Mechatronic Systems (ICAMechS), Melbourne, VIC, Australia, 2016, pp. 498-503, doi: 10.1109/ICAMechS.2016.7813499.

[3] S. Bouabdallah, P. Murrieri and R. Siegwart, "Design and control of an indoor micro quadrotor," IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004, New Orleans, LA, USA, 2004, pp. 4393-4398 Vol.5, doi: 10.1109/ROBOT.2004.1302409.

```
clear;
clc;
close all;
```

Define equations of motion, X vector and X dot vector

```
syms x1 x2 x3 x4 x5 x6 u1 u2 u3 u4  Omega  s lambda u1eq u2eq u3eq u4eq
ix=0.001617;
iy=0.001345;
iz=0.002755;
m=0.525;
L=0.125;
Jr=1.8E-04;
g=9.81;
b=9.8E-07;
d=1.14E-07;

xdot1=x4;
xdot2=x5;
xdot3=x6;
xdot4=x5*x6*((iy-iz)/ix)+(L/ix)*(u4-u2);
xdot5=x4*x6*((iz-ix)/iy)+(L/iy)*(u3-u1);
xdot6=x4*x5*((ix-iy)/iz)+(1/iz)*(u2+u4-u1-u3);

f=[xdot1,xdot2,xdot3,xdot4,xdot5,xdot6];
xvector=[x1,x2,x3,x4,x5,x6];
uvector=[u1,u2,u3,u4];
```

Take partials for Jacobians and define A, B, C, D matrices

```
for i=1:6
    for j=1:6
        Amatrix(i,j)=diff(f(i),xvector(j));
    end
end
Amatrix
```

Amatrix =

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -\frac{470\,x_6}{539} & -\frac{470\,x_5}{539} \\ 0 & 0 & 0 & \frac{1138\,x_6}{1345} & 0 & \frac{1138\,x_4}{1345} \\ 0 & 0 & 0 & \frac{272\,x_5}{2755} & \frac{272\,x_4}{2755} & 0 \end{pmatrix}$$

```
for i=1:6
    for j=1:4
        Bmatrix(i,j)=diff(f(i),uvector(j));
    end
end
Bmatrix;
```

Evaluate Jacobians at star conditions

```
x_star=[0 0 0 0 0 0];
u_star=[u1eq u2eq u3eq u4eq];
Amatrix_star=vpa(subs(Amatrix,[xvector,uvector],[x_star,u_star]))
```

Amatrix_star =

$$\begin{pmatrix} 0 & 0 & 0 & 1.0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

```
Amatrix_star =[0, 0, 0, 1.0,   0,   0;
0, 0, 0,   0, 1.0,   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];
Bmatrix_star=vpa(subs(Bmatrix,[xvector,uvector],[x_star,u_star]))
```

Bmatrix_star =

$$\begin{pmatrix} 0 & 0 & \\ 0 & 0 & \\ 0 & 0 & \\ 0 & -77.303648732220167971718183252 96 & \\ -92.936802973977695167286245353 16 & 0 & 92.9368029739 7 \\ -362.976406533575300272786989808 08 & 362.976406533575300272786989808 08 & -362.976406533 \end{pmatrix}$$

```
Bmatrix_star = [0,       0,       0,      0;
      0,       0,       0,      0;
     0,       0,       0,      0;
      0, -77.304,       0, 77.304;
-92.937,       0,  92.937,      0;
-362.98,  362.98, -362.98, 362.98];
Cmatrix=[
    1 0 0 0 0 0;
    0 1 0 0 0 0;
    0 0 1 0 0 0;zeros(3,6)];
```

```
Dmatrix=zeros(6,4);
```

Define Controllability and Observability matrices

```
ctrbmatrix=[Bmatrix_star,Amatrix_star*Bmatrix_star,Amatrix_star^2*Bmatrix_star,Amatrix_star^3*B
    Amatrix_star^4*Bmatrix_star,Amatrix_star^5*Bmatrix_star];
rank_ctrbmatrix=rank(ctrbmatrix)
```

```
rank_ctrbmatrix = 6
```

```
obsvmatrix=[Cmatrix;Cmatrix*Amatrix_star;Cmatrix*Amatrix_star^2;Cmatrix*Amatrix_star^3;Cmatrix*
rank_obsvmatrix=rank(obsvmatrix)
```

```
rank_obsvmatrix = 6
```

Define transfer functions

```
transferfunctions=Cmatrix*inv((s*eye(6)-Amatrix_star))*Bmatrix_star
```

```
transferfunctions =
```

$$\begin{pmatrix} 0 & -\frac{9663}{125\,s^2} & 0 & \frac{9663}{125\,s^2} \\ -\frac{92937}{1000\,s^2} & 0 & \frac{92937}{1000\,s^2} & 0 \\ -\sigma_1 & \sigma_1 & -\sigma_1 & \sigma_1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

where

$$\sigma_1 = \frac{18149}{50\,s^2}$$

Solve for stability (eigenvalues) and characteristic equation of open loop system

```
characteristic_eq=expand(det(Amatrix_star-lambda*eye(6)))==0
```

```
characteristic_eq = λ^6 = 0
```

```
actual_eigenvalues=solve(characteristic_eq,lambda)
```

```
actual_eigenvalues =
```

$$\begin{pmatrix}0\\0\\0\\0\\0\\0\end{pmatrix}$$

Geometric multiplicity check

```
mi=6;
rankAmatrix=rank(Amatrix_star);
g=6-rankAmatrix;
```

MIMO State Feedback Design

```
syms phi1 phi2 phi3 phi4 phi5 phi6 phi7 phi8 phi9 phi10 lambda_di a b c d
partition_matrix=[lambda_di*eye(6)-Amatrix_star,Bmatrix_star]
```

partition_matrix =

$$\begin{pmatrix}\lambda_{di} & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & \lambda_{di} & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & \lambda_{di} & 0 & 0 & -1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & \lambda_{di} & 0 & 0 & 0 & -\frac{9663}{125} & 0 & \frac{9663}{125}\\ 0 & 0 & 0 & 0 & \lambda_{di} & 0 & -\frac{92937}{1000} & 0 & \frac{92937}{1000} & 0\\ 0 & 0 & 0 & 0 & 0 & \lambda_{di} & -\frac{18149}{50} & \frac{18149}{50} & -\frac{18149}{50} & \frac{18149}{50}\end{pmatrix}$$

```
phimatrix=[phi1;phi2;phi3;phi4;phi5;phi6;phi7;phi8;phi9;phi10];
FBD_design=partition_matrix*phimatrix==0
```

FBD_design =

$$\begin{pmatrix}\lambda_{di}\,\phi_1-\phi_4=0\\ \lambda_{di}\,\phi_2-\phi_5=0\\ \lambda_{di}\,\phi_3-\phi_6=0\\ \frac{9663\,\phi_{10}}{125}-\frac{9663\,\phi_8}{125}+\lambda_{di}\,\phi_4=0\\ \frac{92937\,\phi_9}{1000}-\frac{92937\,\phi_7}{1000}+\lambda_{di}\,\phi_5=0\\ \frac{18149\,\phi_8}{50}-\frac{18149\,\phi_7}{50}-\frac{18149\,\phi_9}{50}+\frac{18149\,\phi_{10}}{50}+\lambda_{di}\,\phi_6=0\end{pmatrix}$$

```
FBD_design=subs(FBD_design,[phi1,phi2,phi3,phi7],[a,b,c,d])
```

FBD_design =

$$\begin{pmatrix} a\,\lambda_{di} - \phi_4 = 0 \\ b\,\lambda_{di} - \phi_5 = 0 \\ c\,\lambda_{di} - \phi_6 = 0 \\ \frac{9663\,\phi_{10}}{125} - \frac{9663\,\phi_8}{125} + \lambda_{di}\,\phi_4 = 0 \\ \frac{92937\,\phi_9}{1000} - \frac{92937\,d}{1000} + \lambda_{di}\,\phi_5 = 0 \\ \frac{18149\,\phi_8}{50} - \frac{18149\,d}{50} - \frac{18149\,\phi_9}{50} + \frac{18149\,\phi_{10}}{50} + \lambda_{di}\,\phi_6 = 0 \end{pmatrix}$$

```
FBD_sys_eqs=solve([FBD_design(1),FBD_design(2),FBD_design(3),FBD_design(4),FBD_design(5),FBD_de
```

```
FBD_sys_eqs = struct with fields:
     phi4: a*lambda_di
     phi6: c*lambda_di
     phi5: b*lambda_di
     phi8: d + (125*a*lambda_di^2)/19326 - (500*b*lambda_di^2)/92937 - (25*c*lambda_di^2)/18149
     phi9: d - (1000*b*lambda_di^2)/92937
    phi10: d - (125*a*lambda_di^2)/19326 - (500*b*lambda_di^2)/92937 - (25*c*lambda_di^2)/18149
```

Create Psi matrices

```
syms a1 a2 a3 a4 a5 a6 a7 a8 a9 a10 b1 b2 b3 b4 b5 b6 b7 b8 b9 b10 c1 c2 c3 c4 c5 c6 c7 c8 c9 c
    d1 d2 d3 d4 d5 d6 d7 d8 d9 d10
lambda_di=[-1 -2 -3 -4 -1 -6];
Psi_1=transpose(vpa(([a1 b1 c1 a1*lambda_di(1)  b1*lambda_di(1) c1*lambda_di(1)])));
Psi_2=transpose(vpa(([a2 b2 c2 a2*lambda_di(2)  b2*lambda_di(2) c2*lambda_di(2)])));
Psi_3=transpose(vpa(([a3 b3 c3 a3*lambda_di(3)  b3*lambda_di(3) c3*lambda_di(3)])));
Psi_4=transpose(vpa(([a4 b4 c4 a4*lambda_di(4)  b4*lambda_di(4) c4*lambda_di(4)])));
Psi_5=transpose(vpa(([a5 b5 c5 a5*lambda_di(5)  b5*lambda_di(5) c5*lambda_di(5)])));
Psi_6=transpose(vpa(([a6 b6 c6 a6*lambda_di(6)  b6*lambda_di(6) c6*lambda_di(6)])));
Psi_total=[Psi_1 Psi_2 Psi_3 Psi_4 Psi_5 Psi_6]
```

```
Psi_total =
```

$$\begin{pmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \\ b_1 & b_2 & b_3 & b_4 & b_5 & b_6 \\ c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \\ -1.0\,a_1 & -2.0\,a_2 & -3.0\,a_3 & -4.0\,a_4 & -1.0\,a_5 & -6.0\,a_6 \\ -1.0\,b_1 & -2.0\,b_2 & -3.0\,b_3 & -4.0\,b_4 & -1.0\,b_5 & -6.0\,b_6 \\ -1.0\,c_1 & -2.0\,c_2 & -3.0\,c_3 & -4.0\,c_4 & -1.0\,c_5 & -6.0\,c_6 \end{pmatrix}$$

Obtain a_i,b_i,c_i,d_i values for Psi matrix until rank=full=invertible then manually save that matrix.

```
min=-10;
max=10;
for w=1:24
    random_num_matrix(w)=round((max-min).*rand(1)+min);
```

```
end
% random_num_matrix=[-5    -8     2    -1    -1     3     5    -3     3 -2 ...
%     7     7    -5     2     2     1     7    -5 4 8 9 3 0 3];
random_num_matrix=[5     1    -6     9    -5     8    -6    -3    -8     3 -6    -9     4    -3
Psi_total=(vpa(subs(Psi_total,[a1,a2,a3,a4,a5,a6,b1,b2,b3,b4,b5,b6,c1,c2,c3,c4,c5,c6],random_nu
rank(Psi_total)
```

```
ans = 6
```

Create KPsi matrrices

```
K_Psi_1=transpose(vpa([d1, d1 + 0.006467999999999999399238276964752 1*a1*lambda_di(1)^2 - 0.0053
    d1 - (35184372088832*a1*lambda_di(1)^2)/5439760681637601 - (269*b1*lambda_di(1)^2)/50000 -
K_Psi_2=transpose(vpa([d2, d2 + 0.006467999999999999399238276964752 1*a2*lambda_di(2)^2 - 0.0053
    d2 - (35184372088832*a2*lambda_di(2)^2)/5439760681637601 - (269*b2*lambda_di(2)^2)/50000 -
K_Psi_3=transpose(vpa([d3, d3 + 0.006467999999999999399238276964752 1*a3*lambda_di(3)^2 - 0.0053
    d3 - (35184372088832*a3*lambda_di(3)^2)/5439760681637601 - (269*b3*lambda_di(3)^2)/50000 -
K_Psi_4=transpose(vpa([d4, d4 + 0.006467999999999999399238276964752 1*a4*lambda_di(4)^2 - 0.0053
    d4 - (35184372088832*a4*lambda_di(4)^2)/5439760681637601 - (269*b4*lambda_di(4)^2)/50000 -
K_Psi_5=transpose(vpa([d5, d5 + 0.006467999999999999399238276964752 1*a5*lambda_di(1)^2 - 0.0053
    d5 - (35184372088832*a5*lambda_di(5)^2)/5439760681637601 - (269*b5*lambda_di(5)^2)/50000 -
K_Psi_6=transpose(vpa([d6, d6 + 0.006467999999999999399238276964752 1*a6*lambda_di(6)^2 - 0.0053
    d6 - (35184372088832*a6*lambda_di(6)^2)/5439760681637601 - (269*b6*lambda_di(6)^2)/50000 -
K_Psi_total=[K_Psi_1 K_Psi_2 K_Psi_3 K_Psi_4 K_Psi_5 K_Psi_6];
K_Psi_total=(vpa(subs(K_Psi_total,[a1,a2,a3,a4,a5,a6,b1,b2,b3,b4,b5,b6,c1,c2,c3,c4,c5,c6,d1,d2,
```

Obtain K gain for SFB

```
K_gain=double((K_Psi_total*inv(Psi_total)))
```

```
K_gain = 4×6
    0.9793   -0.9814   -1.9067   -0.1305    0.1587   -0.8094
    0.9504   -0.9302   -1.8452   -0.1625    0.1964   -0.7789
    0.8707   -0.7600   -2.1227   -0.2384    0.3872   -1.0316
    1.0097   -0.9611   -1.8748   -0.0926    0.1785   -0.7863
```

Check SFB pole placement returns desired eigenvalues

```
syms s
abk=Amatrix_star-Bmatrix*K_gain;
sia=s*eye(6)-abk;
eq1=det(sia)==0;
vpasolve(eq1,s)
```

```
ans =
```

$$\begin{pmatrix} -112.771508394240170143435916715 62 \\ -6.000000000000004101350226379373 4 \\ -3.999999999999994783692240424942 6 \\ -1.999999999999995953345708932037 8 \\ -1.000000000000002761641273622066 2 \\ -0.999999999999999990098257404480 65 \end{pmatrix}$$

Control Law u(t)

```
SFB_U=-K_gain*transpose(xvector)
```

SFB_U =

$$\begin{pmatrix} \frac{4419807287162387\,x_2}{4503599627370496} - \frac{4410187027301669\,x_1}{4503599627370496} + \frac{536677852118741\,x_3}{281474976710656} + \frac{1175149847928559\,x_4}{9007199254740992} - \frac{1429}{900} \\ \frac{1047361705254531\,x_2}{1125899906842624} - \frac{2140201565646423\,x_1}{2251799813685248} + \frac{8310105332274255\,x_3}{4503599627370496} + \frac{5855597257035685\,x_4}{36028797018963968} - \frac{176}{90} \\ \frac{855695081803823\,x_2}{1125899906842624} - \frac{7842996086916003\,x_1}{9007199254740992} + \frac{1194975918500847\,x_3}{562949953421312} + \frac{4293916153729885\,x_4}{18014398509481984} - \frac{6975}{180} \\ \frac{4328257772762875\,x_2}{4503599627370496} - \frac{4547069177172593\,x_1}{4503599627370496} + \frac{4221677748161219\,x_3}{2251799813685248} + \frac{52123404344451\,x_4}{562949953421312} - \frac{6429}{3602} \end{pmatrix}$$

```
clc;
close all;
```

Linear state observer

```
Amatrix_star =[0, 0, 0, 1.0,   0,   0;
0, 0, 0,   0, 1.0,   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]
```

```
Amatrix_star = 6×6
     0     0     0     1     0     0
     0     0     0     0     1     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
```

```
Cmatrix=[1 0 0 0 0 0;
    0 1 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];

Bmatrix_star = [0,       0,       0,      0;
      0,       0,       0,      0;
     0,       0,       0,      0;
      0, -77.304,       0, 77.304;
-92.937,       0,  92.937,      0;
-362.98,  362.98, -362.98, 362.98];

Dmatrix=zeros(6,4);

syms mu_di phi1 phi2 phi3 phi4 phi5 phi6 phi7 phi8 phi9 phi10 phi11 phi12 a b c...
    a1 a2 a3 a4 a5 a6 b1 b2 b3 b4 b5 b6  c1 c2 c3 c4 c5 c6 ...

partition_matrix=[mu_di*eye(6)-transpose(Amatrix_star),transpose(Cmatrix)]
```

```
partition_matrix =
```

$$\begin{pmatrix} \mu_{di} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mu_{di} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mu_{di} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & \mu_{di} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & \mu_{di} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & \mu_{di} & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

```
phimatrix=[phi1;phi2;phi3;phi4;phi5;phi6;phi7;phi8;phi9;phi10;phi11;phi12];
SOD_design=partition_matrix*phimatrix==0
```

SOD_design =

$$\begin{pmatrix} \phi_7 + \mu_{di}\,\phi_1 = 0 \\ \phi_8 + \mu_{di}\,\phi_2 = 0 \\ \phi_9 + \mu_{di}\,\phi_3 = 0 \\ \mu_{di}\,\phi_4 - \phi_1 = 0 \\ \mu_{di}\,\phi_5 - \phi_2 = 0 \\ \mu_{di}\,\phi_6 - \phi_3 = 0 \end{pmatrix}$$

```
SOD_design=subs(SOD_design,[phi4,phi5,phi6],[a,b,c]);
SOD_sys_eqs=solve([SOD_design(1),SOD_design(2),SOD_design(3),SOD_design(4),SOD_design(5),SOD_de
```

```
SOD_sys_eqs = struct with fields:
    phi1: a*mu_di
    phi2: b*mu_di
    phi3: c*mu_di
    phi7: -a*mu_di^2
    phi8: -b*mu_di^2
    phi9: -c*mu_di^2
```

```
mu_di=[-1 -2 -2 -3 -4 -5];
Psi_1=transpose(vpa(([a1*mu_di(1) b1*mu_di(1) c1*mu_di(1)  a1 b1 c1])));
Psi_2=transpose(vpa(([a2*mu_di(2) b2*mu_di(2) c1*mu_di(2)   a2 b2 c2])));
Psi_3=transpose(vpa(([a3*mu_di(3) b3*mu_di(3) c3*mu_di(3)   a3 b3 c3])));
Psi_4=transpose(vpa(([a4*mu_di(4) b4*mu_di(4) c4*mu_di(4)   a4 b4 c4])));
Psi_5=transpose(vpa(([a5*mu_di(5) b5*mu_di(5) c5*mu_di(5)   a5 b5 c5])));
Psi_6=transpose(vpa(([a6*mu_di(6) b6*mu_di(6) c6*mu_di(6)   a6 b6 c6])));
Psi_total=[Psi_1 Psi_2 Psi_3 Psi_4 Psi_5 Psi_6]
```

Psi_total =

$$\begin{pmatrix} -1.0\,a_1 & -2.0\,a_2 & -2.0\,a_3 & -3.0\,a_4 & -4.0\,a_5 & -5.0\,a_6 \\ -1.0\,b_1 & -2.0\,b_2 & -2.0\,b_3 & -3.0\,b_4 & -4.0\,b_5 & -5.0\,b_6 \\ -1.0\,c_1 & -2.0\,c_1 & -2.0\,c_3 & -3.0\,c_4 & -4.0\,c_5 & -5.0\,c_6 \\ a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \\ b_1 & b_2 & b_3 & b_4 & b_5 & b_6 \\ c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \end{pmatrix}$$

```
min=-10;
max=10;
for w=1:18
    random_coeff_matrix(w)=round((max-min).*rand(1)+min);
end
random_coeff_matrix=[  3    -2     0     8   -10    -7    -1     1    -9   3     8    -8    -1
Psi_total=(vpa(subs(Psi_total,[a1,a2,a3,a4,a5,a6,b1,b2,b3,b4,b5,b6,c1,c2,c3,c4,c5,c6],random_co
rank(Psi_total);
Psi_total
```

Psi_total =

$$\begin{pmatrix} -3.0 & 4.0 & 0 & -24.0 & 40.0 & 35.0 \\ 1.0 & -2.0 & 18.0 & -9.0 & -32.0 & 40.0 \\ 1.0 & 2.0 & -20.0 & -6.0 & 20.0 & 35.0 \\ 3.0 & -2.0 & 0 & 8.0 & -10.0 & -7.0 \\ -1.0 & 1.0 & -9.0 & 3.0 & 8.0 & -8.0 \\ -1.0 & -4.0 & 10.0 & 2.0 & -5.0 & -7.0 \end{pmatrix}$$

```
K_Psi_1=transpose(vpa(([-a1*mu_di(1)^2 -b1*mu_di(1)^2 c1*mu_di(1)^2 0 0 0])));
K_Psi_2=transpose(vpa(([-a2*mu_di(2)^2 -b2*mu_di(2)^2 c2*mu_di(2)^2 0 0 0])));
K_Psi_3=transpose(vpa(([-a3*mu_di(3)^2 -b3*mu_di(3)^2 c3*mu_di(3)^2 0 0 0])));
K_Psi_4=transpose(vpa(([-a4*mu_di(4)^2 -b4*mu_di(4)^2 c4*mu_di(4)^2 0 0 0])));
K_Psi_5=transpose(vpa(([-a5*mu_di(5)^2 -b5*mu_di(5)^2 c5*mu_di(5)^2 0 0 0])));
K_Psi_6=transpose(vpa(([-a6*mu_di(6)^2 -b6*mu_di(6)^2 c6*mu_di(6)^2 0 0 0])));
K_Psi_total=[K_Psi_1 K_Psi_2 K_Psi_3 K_Psi_4 K_Psi_5 K_Psi_6];
K_Psi_total=(vpa(subs(K_Psi_total,[a1,a2,a3,a4,a5,a6,b1,b2,b3,b4,b5,b6,c1,c2,c3,c4,c5,c6],rando

L_gain=transpose(vpa((K_Psi_total*inv(Psi_total))));
L_gain=[  26.202,   8.7046, -50.158, 0, 0, 0;
21.431,   14.517,  -49.23, 0, 0, 0;
 -17.157,  -6.8695,  39.132, 0, 0, 0;
  50.621,   19.349, -106.64, 0, 0, 0;
  80.749,   39.394,  -183.9, 0, 0, 0;
-0.21553, -0.81432,   5.366, 0, 0, 0]
```

```
L_gain = 6×6
   26.2020    8.7046  -50.1580         0         0         0
   21.4310   14.5170  -49.2300         0         0         0
  -17.1570   -6.8695   39.1320         0         0         0
   50.6210   19.3490 -106.6400         0         0         0
   80.7490   39.3940 -183.9000         0         0         0
   -0.2155   -0.8143    5.3660         0         0         0
```

```
K_gain=[0.9793   -0.9814   -1.9067   -0.1305    0.1587 -0.8094;
    0.9504   -0.9302   -1.8452   -0.1625    0.1964  -0.7789;
    0.8707   -0.7600   -2.1227   -0.2384    0.3872 -1.0316;
    1.0097   -0.9611   -1.8748   -0.0926    0.1785 -0.7863]
```

```
K_gain = 4×6
    0.9793   -0.9814   -1.9067   -0.1305    0.1587   -0.8094
    0.9504   -0.9302   -1.8452   -0.1625    0.1964   -0.7789
    0.8707   -0.7600   -2.1227   -0.2384    0.3872   -1.0316
    1.0097   -0.9611   -1.8748   -0.0926    0.1785   -0.7863
```

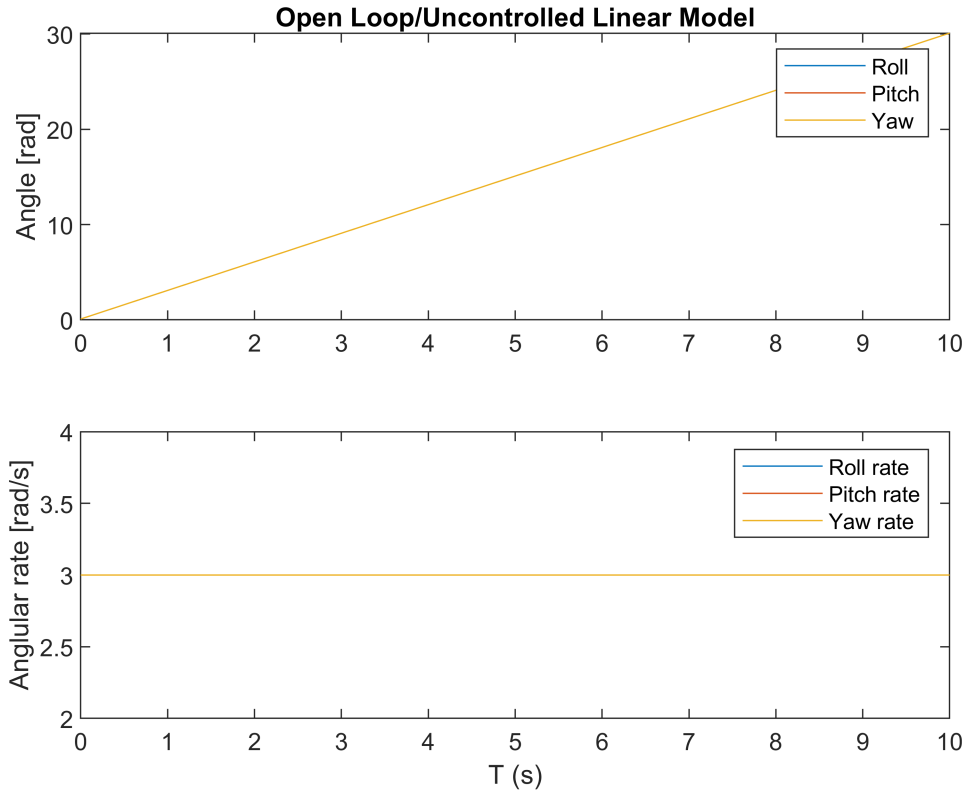
```
%eig(Amatrix_star-L_gain*Cmatrix)
```

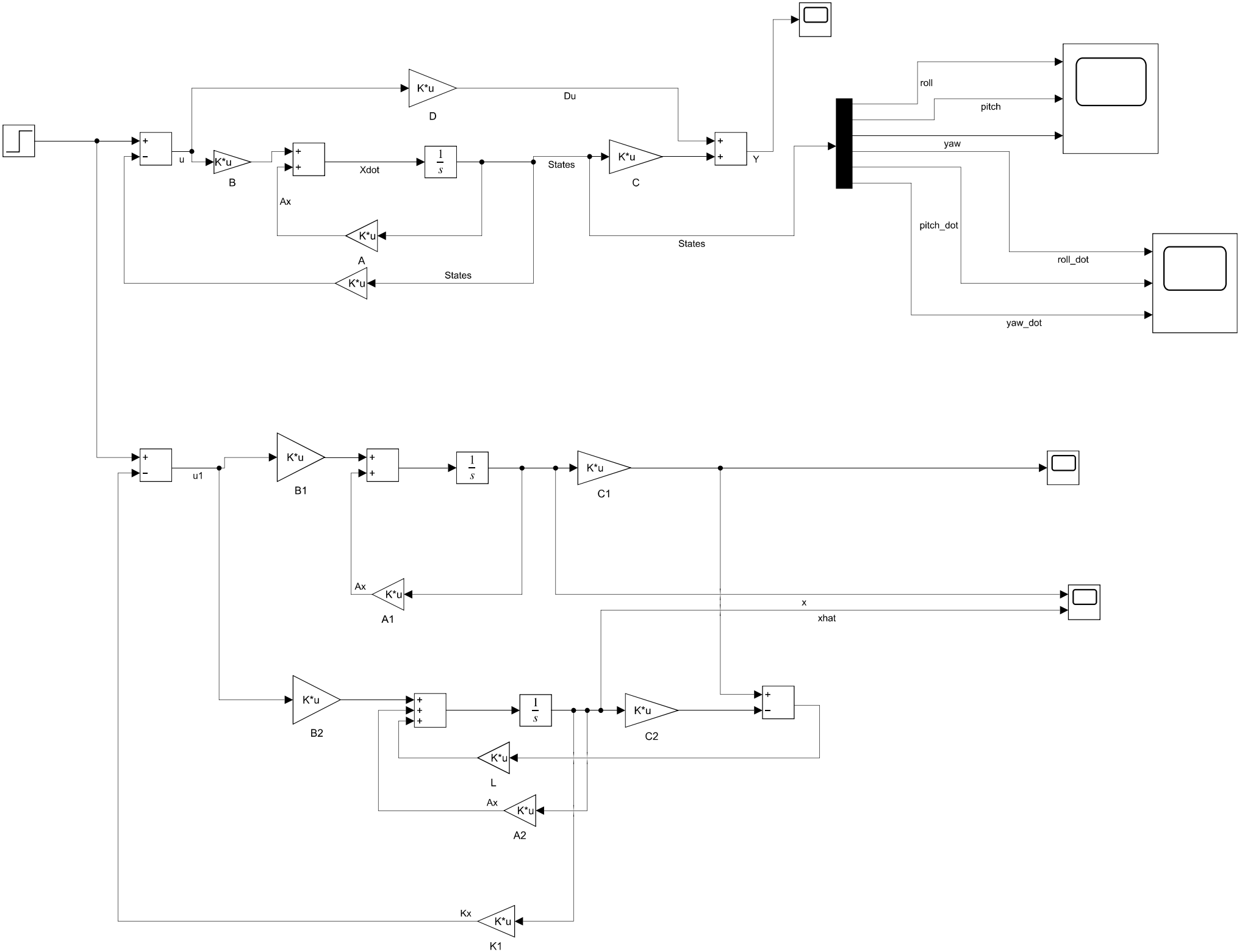
Plotting

```
time=out.tout;

figure()
```

```
subplot(2,1,1)
plot(time,out.states(:,1))
hold on
plot(time,out.states(:,2))
hold on
plot(time,out.states(:,3))
title('Open Loop/Uncontrolled Linear Model')
ylabel('Angle [rad]')
legend('Roll','Pitch','Yaw')
subplot(2,1,2)
plot(time,out.states(:,4))
hold on
 plot(time,out.states(:,5))
 hold on
 plot(time,out.states(:,6))
 ylabel('Anglular rate [rad/s]')
 legend('Roll rate','Pitch rate','Yaw rate')
 xlabel('T (s)')
```




K*u
D
Du
u
B
Xdot
1/s
States
C
Y
Ax
A
States
roll
pitch
yaw
pitch_dot
roll_dot
yaw_dot
u1
B1
A1
C1
x
xhat
B2
L
A2
C2
Kx
K1