Michael Cummins

Control Systems Learning Element 1

Mob El+Inf.Tech | 22-908-255

Task 1

a) Matlab Code

```
close all
clear all
Clc

% define constants
z_d = 2;
psi_d = pi/4;
m = 0.03;
g = 9.81;
Ix = 1.5e-5;
Iy = Ix;
Iz = 3e-5;
kx = 4.5e-3;
ky = 4.5e-3;
kz = 4.5e-3;
kp = 4.5e-4;
kq = 4.5e-4;
kr = 4.5e-4;

% equilibrium points
sys1_eq = [0 0];
sys2_eq = [0 0 0 0 0 psi_d m*g];
sys3_eq = [0 0 0 0 0 0];

% system 1 linearization
syms phi theta
sys1_states = [phi theta];
sys1_A = [1 sin(phi)*tan(theta) cos(phi)*tan(theta);
      0 cos(phi) -sin(phi);
      0 sin(phi)/cos(theta) cos(phi)*cos(theta)];
sys1_A_lin = subs(sys1_A, sys1_states, sys1_eq);

% system 2 linearization
syms xdot ydot zdot phi theta psi u1
states = [xdot ydot zdot];
euler_angles = [phi theta psi];
input = u1;
xddot = (cos(phi)*sin(theta)*cos(psi) + sin(phi)*sin(psi))*u1 - kx*xdot;
yddot = (cos(phi)*sin(theta)*sin(psi) - sin(phi)*cos(psi))*u1 - ky*ydot;
zddot = (cos(phi)*cos(theta)*u1 - m*g - kz*zdot);
sys2 = [xddot; yddot; zddot]/m;
sys2_Alin = subs(jacobian(sys2, states), [states euler_angles input], sys2_eq);
sys2_Blin = subs(jacobian(sys2, input), [states euler_angles input], sys2_eq);
A2 = double(vpa(sys2_Alin));
```

```
B2 = double(vpa(sys2_Blin));

% system 3 linearization
syms p q r u2 u3 u4
states = [p q r];
inputs = [u2 u3 u4];
pdot = ((Ix - Iz)*q*r + u2 -kp*p)/Ix;
qdot = ((Iz - Ix)*p*r + u3 - kq*q )/Iy;
rdot = ((Ix - Iy)*p*q + u4 - kr*r )/Iz;
sys3 = [pdot; qdot; rdot];
sys3_Alin = subs(jacobian(sys3, states), [states inputs], sys3_eq);
sys3_Blin = subs(jacobian(sys3, inputs), [states inputs], sys3_eq);
A3 = double(vpa(sys3_Alin));
B3 = double(vpa(sys3_Blin));

% Transfer Functions
syms s
sI = eye(12)*s;

% state space matrices
A = [0 0 0 1 zeros([8 1]).';
  0 0 0 0 1 zeros([7 1]).';
  0 0 0 0 0 1 zeros([6 1]).';
  0 0 0 -kx/m 0 0 g*sin(psi_d) g*cos(psi_d) 0 0 0 0;
  0 0 0 0 -ky/m 0 -g*cos(psi_d) g*sin(psi_d) 0 0 0 0;
  0 0 0 0 0 -kz/m 0 0 0 0 0 0;
  zeros([1 9]) 1 0 0;
  zeros([1 10]) 1 0;
  zeros([1 11]) 1;
  zeros([1 9]) -kp/Ix 0 0;
  zeros([1 10]) -kq/Iy 0;
  zeros([1 11]) -kr/Iz;];
B = [zeros([5 4]);
  1/m 0 0 0;
  zeros([3 4]);
  0 1/Ix 0 0;
  0 0 1/Iy 0;
  0 0 0 1/Iz];
C = [eye(3) zeros([3 9]);
  zeros([1 8]) 1 0 0 0];
D = zeros([4 4]);

% compute transfer function of system
tf = tf(ss(A,B,C,D))
```

b) Transfer functions derived from matlab

$$G_z(s) = \frac{33.33}{s^2 + 0.15s}$$

$$G_\Psi(s) = \frac{3.333\times10^{-4}}{s^2 + 15s}$$

Task 2

a) Matlab Code

```
close all
clear all
clc
% define constants
z_d = 2;
psi_d = pi/4;
m = 0.03;
g = 9.81;
Ix = 1.5e-5;
Iy = Ix;
Iz = 3e-5;
kx = 4.5e-3;
ky = 4.5e-3;
kz = 4.5e-3;
kp = 4.5e-4;
kq = 4.5e-4;
kr = 4.5e-4;

% state space matrices
A = [0 0 0 1 zeros([8 1]).';
   0 0 0 0 1 zeros([7 1]).';
   0 0 0 0 0 1 zeros([6 1]).';
   0 0 0 -kx/m 0 0 g*sin(psi_d) g*cos(psi_d) 0 0 0 0;
   0 0 0 0 -ky/m 0 -g*cos(psi_d) g*sin(psi_d) 0 0 0 0;
   0 0 0 0 0 -kz/m 0 0 0 0 0 0;
   zeros([1 9]) 1 0 0;
   zeros([1 10]) 1 0;
   zeros([1 11]) 1;
   zeros([1 9]) -kp/Ix 0 0;
   zeros([1 10]) -kq/Iy 0;
```

```
    zeros([1 11]) -kr/Iz;];
B = [zeros([5 4]);
    1/m 0 0 0;
    zeros([3 4]);
    0 1/Ix 0 0;
    0 0 1/Iy 0;
    0 0 0 1/Iz];
C = [eye(3) zeros([3 9]);
    zeros([1 8]) 1 0 0 0];
D = zeros([4 4]);

% Eigenvalues
eigA = eig(A);
J = jordan(A)
```

b) Jordan normal of A

J =

$$
\begin{pmatrix}
0 & 1.0000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -30.0000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -0.1500 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1.0000 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -30.0000 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.1500 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -15.0000 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.1500 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
$$

c) Accessing stability

The system was originally thought to be stable since all eigenvalues were non-negative. However, since the Jordan for is not diagonal, then not all jordan blocks are of dimension 1x1. Therefore, the system is neither stable or asymptotically stable at the equilibrium point and is concluded to be unstable.

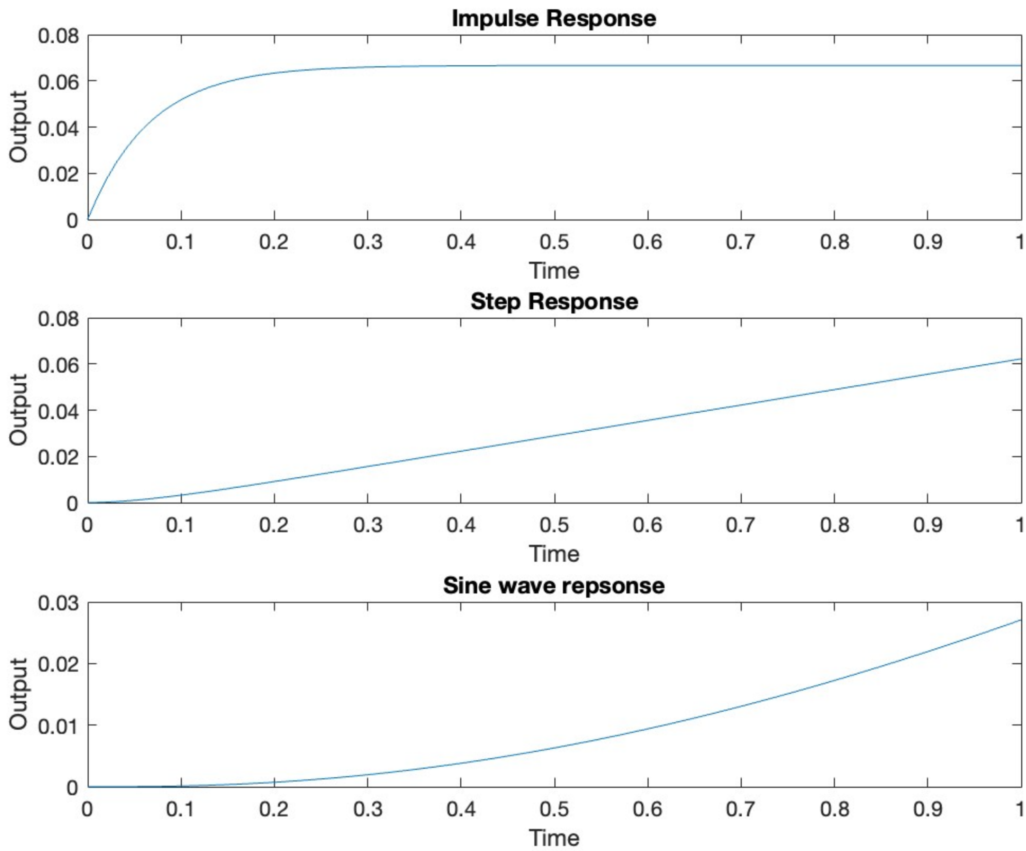
Task 3

a) Matlab Code

```
close all
clear all
clc
% define constants
z_d = 2;
psi_d = pi/4;
m = 0.03;
g = 9.81;
Ix = 1.5e-5;
Iy = 1.5e-5;
Iz = 3e-05;
kx = 4.5e-03;
ky = 4.5e-03;
kz = 4.5e-03;
kp = 4.5e-04;
kq = 4.5e-04;
kr = 4.5e-04;

% Define state space model
A = [0 1; 0 -15];
B = [0; 3.3333e+04];
C = [1 0];
D = 0;
T = 0.01;

% Discretize the system
sys = ss(A, B, C, D);

% Initial Conditions
x0 = [0;0];
t = (0:T:1);

% impulse response evolution
y1 = impulse(sys, t)*3e-05;

% step response evolution
y2 = step(sys, t)*3e-05;

% sine wave response evolution
u4 = sin(t)*3e-05;
y3 = lsim(sys, u4, t, x0);

% Analysing transfer function
Hs = tf(sys);
```

```
% Plotting results
subplot(3,1,1);
plot(t, y1)
title('Impulse Response')
xlabel('Time')
ylabel('Output')
axis([0 1 0 0.08])
subplot(3,1,2)
plot(t, y2)
title('Step Response')
xlabel('Time')
ylabel('Output')
axis([0 1 0 0.08])
subplot(3,1,3)
plot(t, y3)
title('Sine wave repsonse')
axis([0 1 0 0.03])
xlabel('Time')
ylabel('Output')
```

b) Figure showing the heading trajectory vs time for all three control input cases.



c) Verification of steady state error value using final value theorem

From matlab we find the transfer function;

$$H(s) = \frac{33{,}333}{s(s+15)}$$

The laplace transform of $\delta(0)3 \times 10^{-5} = 3 \times 10^{-5}$

By the Final Value Theorem,

$$\lim_{t \to \infty} y(t) = \lim_{y \to 0} sY(s) = \lim_{y \to 0} sH(s)U(s) = \lim_{y \to 0} \frac{33{,}333(3 \times 10^{-5})}{s+15} = \frac{1}{15} = 0.0667$$

Task 4

a) Matlab Code

```
close all
clear all
clc
% define constants
z_d = 2;
psi_d = pi/4;
m = 0.03;
g = 9.81;
Ix = 1.5e-5;
Iy = 1.5e-5;
Iz = 3e-05;
kx = 4.5e-03;
ky = 4.5e-03;
kz = 4.5e-03;
kp = 4.5e-04;
kq = 4.5e-04;
kr = 4.5e-04;
k = [-0.0001 0.005 0.00005 0];
x0 = [-1; 0];
t = (0:0.01:300);

% Convergent system oscillations
Af = [0 1; -k(1)/m -kz/m];
Bf = [0;0];
C = [1 0];
D = 0;
sys_feedback = ss(Af, Bf, C, D);
ya = initial(sys_feedback, x0, t);
```

```
% divergent system
Af = [0 1; -k(2)/m -kz/m];
sys_feedback = ss(Af, Bf, C, D);
yb = initial(sys_feedback, x0, t);

% convergent with oscillations
Af = [0 1; -k(3)/m -kz/m];
sys_feedback = ss(Af, Bf, C, D);
yc = initial(sys_feedback, x0, t);

% Convergent to non-zero value
Af = [0 1; -k(4)/m -kz/m];
sys_feedback = ss(Af, Bf, C, D);
yd = initial(sys_feedback, x0, t);

% Plotting
subplot(2,2,1)
plot(t, ya)
hold on
plot(t, zeros([1, length(t)]), 'r--')
title('Divergent, K = -0.0001')
xlabel('Time')
ylabel('Output')
axis([0 60 -2.5 0.5])
hold off
subplot(2,2,2)
plot(t, yb)
hold on
plot(t, zeros([1, length(t)]), 'r--')
title('Convergent with oscillations, K = 0.005')
xlabel('Time')
ylabel('Output')
axis([0 60 -1.5 1])
hold off
subplot(2,2,4)
plot(t, yc)
hold on
plot(t, zeros([1, length(t)]), 'r--')
title('Convergent without Oscillations, K = 0.00005')
xlabel('Time')
ylabel('Output')
axis([0 300 -1.5 1.5])
hold off
subplot(2,2,3)
plot(t, yd)
hold on
plot(t, zeros([1, length(t)]), 'r--')
title('Convergent to non-zero state, K = 0')
```

```
xlabel('Time')
ylabel('Output')
axis([0 300 -1.5 1.5])
hold off
```

b) Figure of responses for different gain values