Mark Peterson HW #6 ECEN 773

1) 23.3

Verify that the LQG/LQR controller makes the closed loop system asymptotically stable

$\dot{\hat{x}} = (A - LC)\hat{x} + Bu + Ly \qquad u = -Kx \qquad \dot{x} = Ax + Bu \qquad e = x - \hat{x}$

$\dot{\hat{x}} = (A - LC - BK)\hat{x} + LCx \qquad \dot{x} = Ax - BK\hat{x}$

$$\dot{e} = \dot{x} - \dot{\hat{x}} = Ax - \cancelto{0}{BK\hat{x}} - A\hat{x} + LC\hat{x} - LCx + \cancelto{0}{BK\hat{x}}$$
$$= (A - LC)e$$ ✓

↑ This is stable so the estimation will go to zero &

? $\dot{x} = Ax - BK\hat{x} = Ax - BKx = (A - BK)x$ which is stable, so the system is stable

2) 23.4

verify that the solution to (23.17) is given by (23.18)

$$\begin{cases} Ax_{eq} + Bu_{eq} = 0 \\ r = Gx_{eq} + Hu_{eq} \end{cases} \iff \begin{bmatrix} \overset{n\times n}{-A} & \overset{n\times k}{B} \\ \overset{\ell\times n}{-G} & \overset{\ell\times k}{H} \end{bmatrix} \begin{bmatrix} -x_{eq} \\ u_{eq} \end{bmatrix} = \begin{bmatrix} 0 \\ r \end{bmatrix}$$

Let $P(S) = \underset{(n+\ell)\times(n+k)}{\begin{bmatrix} SI - A & B \\ -G & H \end{bmatrix}}$ be the rosenbrock matrix then

$P(0) = \begin{bmatrix} -A & B \\ -G & H \end{bmatrix}$ if $P(0)$ is full rank & number of inputs is larger than the number of inputs then

$\begin{bmatrix} -x_{eq} \\ u_{eq} \end{bmatrix} = P(0)'\left(P(0)P(0)'\right)^{-1}\begin{bmatrix} 0 \\ r \end{bmatrix}$ proof. sub into equation 23.17.

$$\cancelto{I}{P(0)\,P(0)'\left(P(0)P(0)'\right)^{-1}}\begin{bmatrix} 0 \\ r \end{bmatrix} = \begin{bmatrix} 0 \\ r \end{bmatrix} \Rightarrow \begin{bmatrix} 0 \\ r \end{bmatrix} = \begin{bmatrix} 0 \\ r \end{bmatrix}$$

3) 23.5

Verify that the LQG/LQR set point controller (23.21) makes the closed loop system asymptotically stable.

$\dot{\bar{x}} = (A - LC - BK)\bar{x} - L(y - \overset{y_{eq}}{Cx_{eq}})$ $\quad u = K\bar{x} + u_{eq}$ $\quad u_{eq} = Nr$ $\quad y_{eq} = rCF$

$\bar{x} = x_{eq} - \hat{x}$ $\quad \tilde{x} = x - x_{eq}$ $\quad e = x - \hat{x}$

$$\dot{e} = \dot{x} - \dot{\hat{x}} = Ax + Bu + (A - LC - BK)\bar{x} - L(y - Cx_{eq})$$
$$= Ax + \cancel{BK\bar{x}} + Bu_{eq} + A\bar{x} - LC\bar{x} - \cancel{BK\bar{x}} - L(y - Cx_{eq})$$
$$= AX + A\bar{x} + Bu_{eq} - LC\bar{x} - LCX + LCX_{eq}$$
$$= AX + A(x_{eq} - \hat{x}) + Bu_{eq} - LC(x_{eq} - \hat{x}) - LCX + LCx_{eq}$$
$$= (A - LC)(x - \hat{x}) + \underbrace{\cancel{Ax_{eq}} + \overset{0}{\cancel{Bu_{eq}}}}_{\text{at equilibrium this is zero}}$$

?

$$= (A - LC)\, e$$

this shows that $\hat{x} \to x$ as $t \to \infty$ & (23.21) shows that $\bar{x} = x_{eq} - \hat{x} \to 0$ as $t \to \infty$ thus the system is stable.

✓

4) 23.6

Show that for a single controlled output ($\ell = 1$), we can take $u_{eq} = 0$ in (23.17) when the matrix A has an eigenvalue at the origin and this mode is observable through z.

If this mode is observable through z then $\operatorname{rank}\begin{bmatrix} A - \lambda I \\ G \end{bmatrix} = n$

$$\begin{bmatrix} -A & B \\ -G & H \end{bmatrix}\begin{bmatrix} -x_{eq} \\ u_{eq} \end{bmatrix} = \begin{bmatrix} 0 \\ r \end{bmatrix}$$ let $u_{eq} = 0$ then $$\begin{bmatrix} -A\,x_{eq} \\ -G\,x_{eq} \end{bmatrix} = \begin{bmatrix} 0 \\ r \end{bmatrix}$$

this means $Ax_{eq} = 0$ & $-Gx_{eq} = r$.
Since A has an eigen value at the origin, then A is singular & X_{eq} must be in the null space of A. Since this mode is observable through G s.t. $Gx_{eq} \neq r$, then X_{eq} can be chosen such that $-Gx_{eq} = r$

remember $x_{eq} \in N(A) = \alpha v_0$ where α is a scalar and v_0 is the eigen vector corresponding to the 0 eigen value.

$-Gx_{eq} = \alpha(-Gv_0) = r$ where α can be a tuning value to get any value r desired.

✓

7) show that $\det\begin{bmatrix} z_i - A & B \\ -C & D \end{bmatrix} = \det(z_i - A)\det(G(z_i))$

$$\begin{bmatrix} z_i - A & B \\ -C & D \end{bmatrix} = \begin{bmatrix} z_i I - A & 0 \\ -C & I \end{bmatrix}\begin{bmatrix} I & (z_i I - A)^{-1}B \\ 0 & G(z_i) \end{bmatrix}$$

$$\det\left(\begin{bmatrix} z_i I - A & 0 \\ -c & I \end{bmatrix}\begin{bmatrix} I & (z_i I - A)^{-1}B \\ 0 & G(z_i) \end{bmatrix}\right) = \det\left(\begin{bmatrix} zI - A & 0 \\ -c & I \end{bmatrix}\right)\det\left(\begin{bmatrix} I & (I - A)^{-1}B \\ 0 & G\, z_i \end{bmatrix}\right)$$

$$= \det(z_i I - A)\det(G(z_i))$$

✓

Table of Contents

5)

```
A = [2 0 0 ; 0 -1 0; 0 0 -1];
B = [1 0 ; 1 0; 0 1];
C = [1 0 2; 0 -1 0];
D = [1 0; 1 0];

sys = ss(A,B,C,D);
```

a)

Whare are the poles of the system? What is the multiplicity of each pole?

```
syms s

G_s = C*inv(s*eye(3)-A)*B +D;
% Poles of the system are 2,-1,-1
```

b)

What are the invariant zeros of the system(finite and infinite)?

```
P = [s*eye(3)-A, B; -C D]

% An invariant zero will cause P to lose rank.
rank_2 = rank(subs(P,2))
rank_1 = rank(subs(P,-1))
rank_0 = rank(subs(P,0))

% Invariant zeros are 0, inf and 2
```

c) Transmission zeros are the invariant zeros that are not eigenvalues

of A.

```
% Since 2 is an eigen value of A, 0 is the only transmission zero.
```

```
P =

[ s - 2,     0,     0, 1, 0]
[     0, s + 1,     0, 1, 0]
[     0,     0, s + 1, 0, 1]
[    -1,     0,    -2, 1, 0]
[     0,     1,     0, 1, 0]


rank_2 =

     4


rank_1 =

     5


rank_0 =

     4
```

6)

```
clear all;
A = [-1 0 0; 0 -2 0; 0 0 -2];
B = [2 -2; -2 4; -4 2];
C = [1 1 0; 1 0 1];
D = [0 0; 0 0];
sys = ss(A,B,C,D);
```

a) Use the transmission zero to fine an input u(t) and the initial condition x(0) that will result in y(t) = 0 for all time.

```
lambda_A = eig(A);      % Eigen values are -2, -2, -1
inv_zeros = tzero(sys); % Invariant zeros are 1
                        % The transmission zero must be 1 since 1 is
 not an
                        % eigen value of A.

% Now that we have a transmission zero, we can find an x(s) and u(s)
 that
% are in the null space of P(s=transmission_zero).
```

```
syms s
P = [s*eye(3)-A, B; -C D];
Pz = [1*eye(3) - A,B; -C D];
nu = null(double(subs(P,1)));
nu = null(Pz)

% Thus the solution is x(0) = -[2 -2 -2]' and u(t) = [-1 1]'exp(t)'
```

b) Verify your solution by calculating the output for the given input

```
xo = nu(1:3);
uo = -nu(4:5);

% opts = odeset('RelTol',1e-2,'AbsTol',1e-8);

[t,x] = ode45(@(t,x) prob6(t,x,A,B,C,D,uo), [0 2],xo);

y = C*x';

sum(y,2)

syms t
u = uo.*exp(0:0.01:2);

[y,x] = lsim(ss(A,B,C,0),uo.*exp(0:0.01:5),0:0.01:5,xo);
sum(y)



function  dx = prob6(t,x,A,B,C,D,uo)

u = uo*exp(t);

dx = A*x+B*u;


end
```

```
nu =

    0.5345
   -0.5345
   -0.5345
   -0.2673
    0.2673


ans =

   1.0e-04 *
```

```
    0.2060
    0.2060


ans =

   1.0e-03 *

   -0.2190   -0.2190
```

Published with MATLAB® R2018a

Table of Contents

Question 8

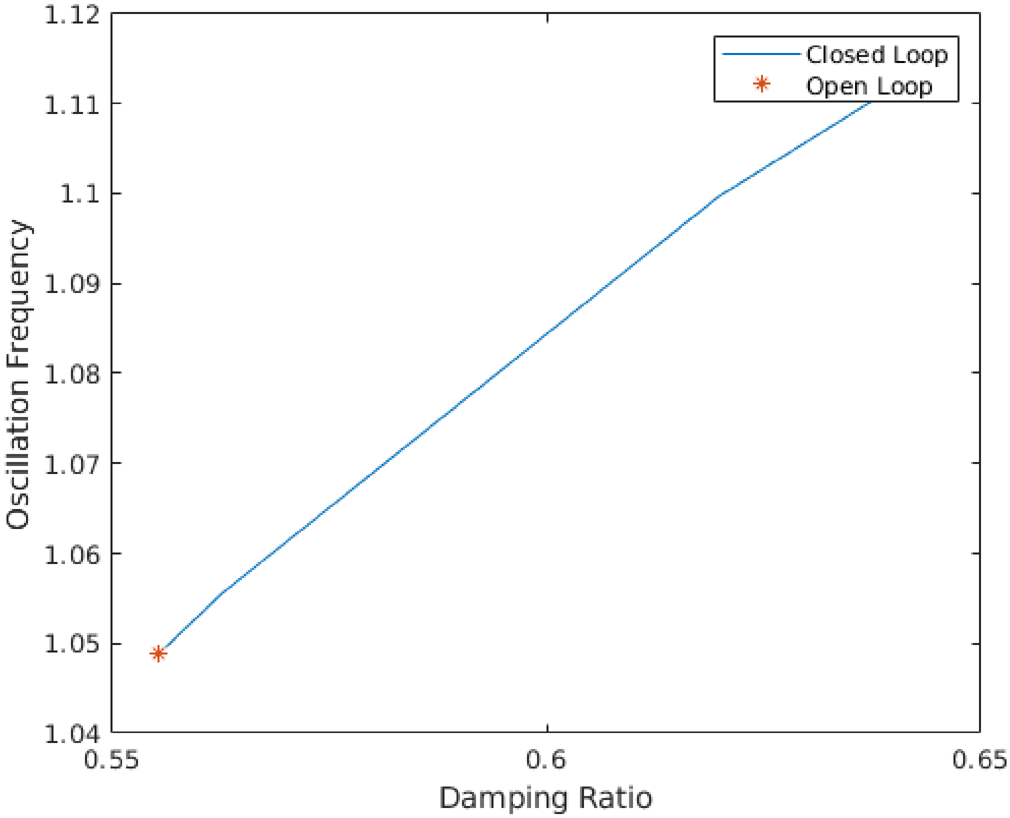
Xo = [250,0,0,0]';

```
th_max = 0.5;        % Throttle max
el_max =  25*pi/180; % Elevator max


%
% x = [Airspeed, Angle of attack, Pitch Rate, Pitch Angel]'
% u = [Throttle, elevator]'

% State Transition Model
A = [-0.038   18.984   0      -32.174;...
     -0.001  -0.632    1        0;...
      0       -0.759   -0.518   0;...
      0        0        1       0];

  % Control-Input Model
B =   [10.1       0;...
        0        -0.0086;...
        0.025    -0.011;...
        0         0];

C = zeros(2,4);
C(1:2,1:2) = eye(2);
D = 0;

SYS = ss(A,B,C,D);

 % Input Weights
 R1 = diag([1/th_max^2; 1/el_max^2]);  % Bryson's method
 r = 1;                                % Scaling Factor
 R = r*R1;

 % State Weights
 max_angle_dev = 0.5*pi/180;
 Q1 = diag([0.1, 1/max_angle_dev^2,1/max_angle_dev^2, 1/
max_angle_dev^2 ]); % Bryson's method
 q = 1;
    % Scaling Factor
 Q = q*Q1;
```

a

Set q = 1 and let r = [1000, 100, 10, 7]. Plot the damping ratio vs oscilation frequency for the closed-loop short period mode obtained using LQR optimal. Include the open loop short period modes too.

```
% Openloop
[Wn,Z,P] = damp(SYS);

% Close loop
Wnc = zeros(4,4);
Zc = zeros(4,4);
Pc = zeros(4,4);
ii = 1;
for r = [7,10,100,1000]

    R = r*R1;
    [K,S,E] = lqr(A,B,Q,R,0);             % Compute K gain
    Ac = A-B*K;
    Bc = zeros(4,2);
    SYSc = ss(Ac,Bc,C,0);
    [Wnc(:,ii),Zc(:,ii),Pc(:,ii)] = damp(SYSc);
    ii = ii +1;
end

figure(1), clf;
plot(Zc(4,:),Wnc(4,:));
hold on
plot(Z(4),Wn(4),'*');
xlabel("Damping Ratio");
ylabel("Oscillation Frequency");
legend("Closed Loop","Open Loop")

% It seems that as r becomes smaller, the closed loop system short
period
% modes converge to the open loop system short period modes.
```

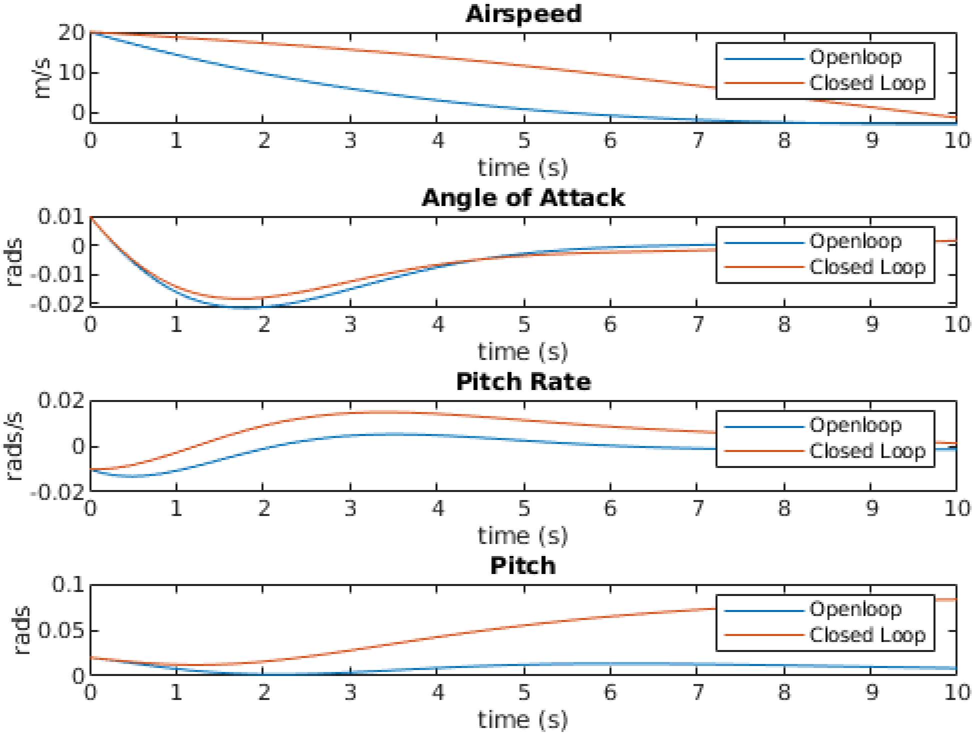


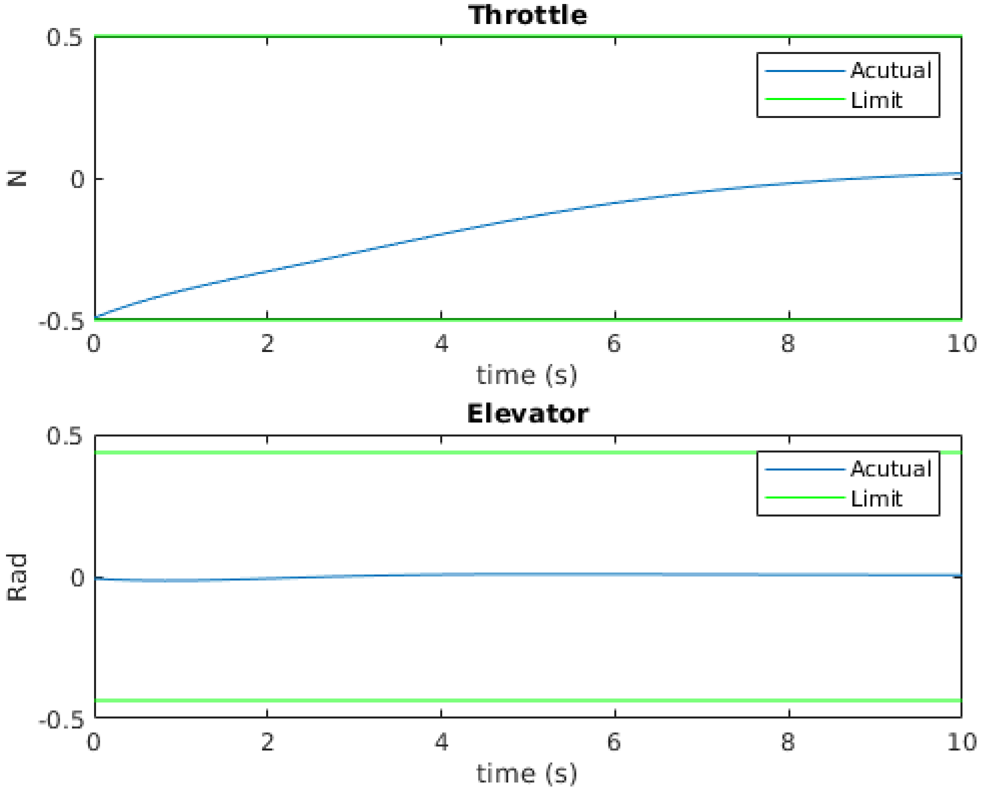
b

Choose an R matrix that maximizes use of the elevator and throttle deflection without exceeding their bounds given an initial perturbation of xo = [20, 0.01, -0.01, 0.02]'. Make plots of the uncontrolled and controlled responses to verify your design.

```
r =54;     % Scale
R = r*R1;
xo = [20, 0.01, -0.01, 0.02]'; % Initial Conditions
xo = [xo;xo];
[K,S,E] = lqr(A,B,Q,R,0);

[t,x] = ode45(@(t,x) aircraftDynamics(t,x,A,B,C,K,[]),[0 10],xo);
  % closed loop
[t_ol,x_ol] = ode45(@(t,x) aircraftDynamics(t,x,A,B,C,[],[]),[0
10],xo); % Open loop
u = -K*x(:,1:4)';

figure(2),clf;

subplot(4,1,1);

plot(t,x(:,1));
hold on
plot(t_ol,x_ol(:,1));
```

```
 title("Airspeed")
 xlabel("time (s)");
 ylabel("m/s");
 legend("Openloop", "Closed Loop")

 subplot(4,1,2);
 plot(t,x(:,2));
 hold on
 plot(t_ol,x_ol(:,2));
 title("Angle of Attack")
 xlabel("time (s)");
 ylabel("rads");
 legend("Openloop", "Closed Loop")

 subplot(4,1,3);
 plot(t,x(:,3));
 hold on
 plot(t_ol,x_ol(:,3));
 title("Pitch Rate")
 xlabel("time (s)");
 ylabel("rads/s");
 legend("Openloop", "Closed Loop")

 subplot(4,1,4);
 plot(t,x(:,4));
 hold on
 plot(t_ol,x_ol(:,4));
 title("Pitch")
 xlabel("time (s)");
 ylabel("rads");
 legend("Openloop", "Closed Loop")

 figure(3),clf;
 subplot(2,1,1);
 plot(t,u(1,:));
 hold on
 plot(t,ones(1,length(t))*th_max,'g');
 plot(t,-ones(1,length(t))*th_max,'g');
 title("Throttle")
 xlabel("time (s)");
 ylabel("N");
 legend("Acutual","Limit")

subplot(2,1,2)
 plot(t,u(2,:));
 hold on
 plot(t,ones(1,length(t))*el_max,'g');
 plot(t,-ones(1,length(t))*el_max,'g');
 title("Elevator")
 xlabel("time (s)");
 ylabel("Rad");
 legend("Acutual","Limit")
```


Airspeed
m/s
20
10
0
Openloop
Closed Loop
Angle of Attack
rads
0.01
0
-0.01
-0.02
Openloop
Closed Loop
Pitch Rate
rads/s
0.02
0
-0.02
Openloop
Closed Loop
Pitch
rads
0.1
0.05
0
Openloop
Closed Loop
0 1 2 3 4 5 6 7 8 9 10
time (s)

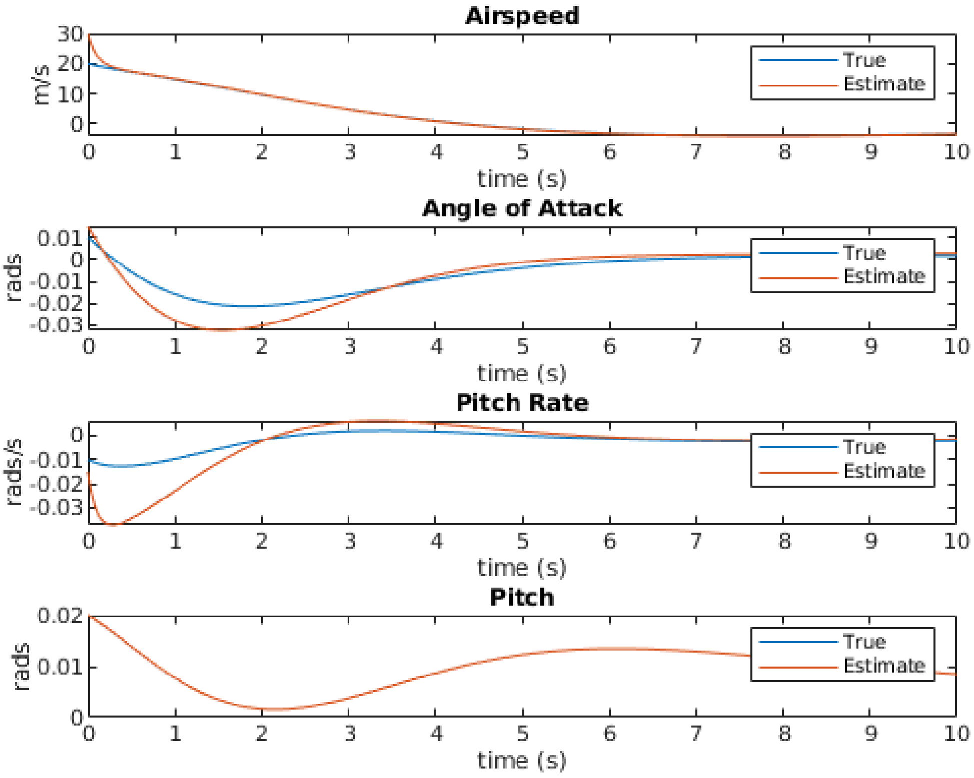

Throttle
N
0.5
0
-0.5
Acutual
Limit
0 2 4 6 8 10
time (s)
Elevator
Rad
0.5
0
-0.5
Acutual
Limit
0 2 4 6 8 10
time (s)

c)

Iplement a kalman filter observer. And we wish to control the airspeed and the flight path angle which are the only variables measured.

```
xeq = [10,0,0,0]';
ueq = -B \ A*xeq;
A*xeq + B*ueq;

SYSk = ss(A, [B B*B'],C,0);

R = diag([1,10^-5]);
Q = B*10^-4*B';
[est,L,P] = kalman(SYSk,Q,R);

xo = [20, 0.01, -0.01, 0.02]'; % Initial Conditions
xo = [xo;xo*1.5];
[tk,xk] = ode45(@(t,x) aircraftDynamics(t,x,A,B,C,K,L),[0 10],xo); %
Open loop

figure(4),clf;

subplot(4,1,1);

plot(tk,xk(:,1));
hold on
plot(tk,xk(:,5));
title("Airspeed")
xlabel("time (s)");
ylabel("m/s");
legend("True", "Estimate")

subplot(4,1,2);
plot(tk,xk(:,2));
hold on
plot(tk,xk(:,6));
title("Angle of Attack")
xlabel("time (s)");
ylabel("rads");
legend("True", "Estimate")

subplot(4,1,3);
plot(tk,xk(:,3));
hold on
plot(tk,xk(:,7));
title("Pitch Rate")
xlabel("time (s)");
ylabel("rads/s");
legend("True", "Estimate")

subplot(4,1,4);
plot(t,x(:,4));
hold on
```

```
plot(t,x(:,8));
title("Pitch")
xlabel("time (s)");
ylabel("rads");
legend("True", "Estimate")
```



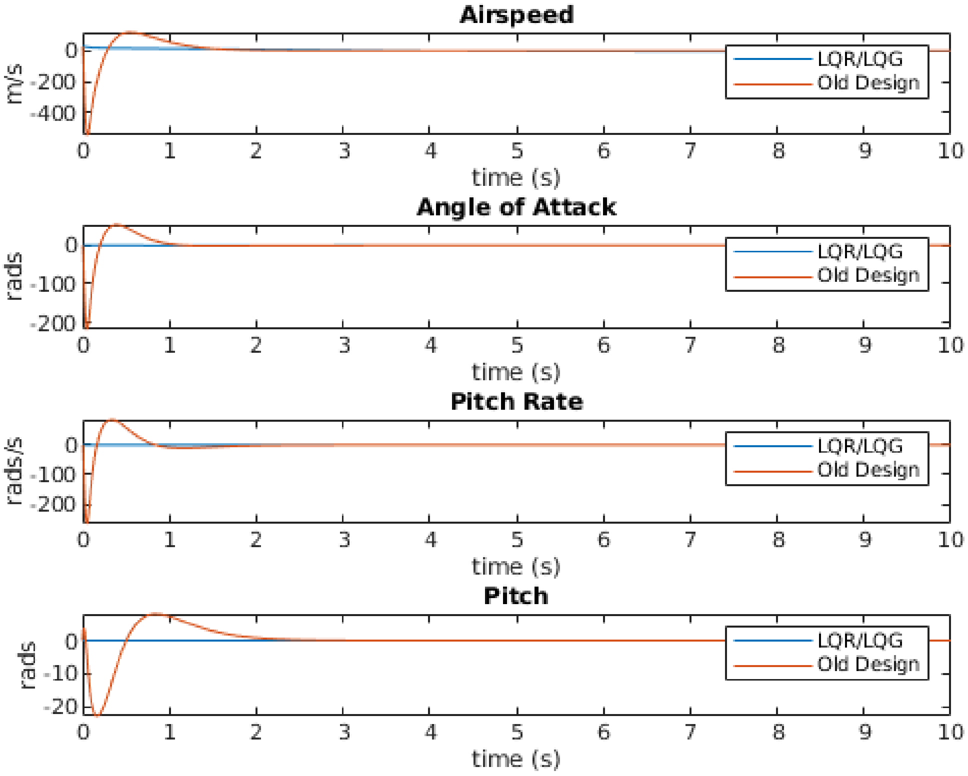
d

Use the observer gains and controller gains derived in parts c) and d) from hw #5 to regulate your system to their trim conditions.

```
p = [-5+j,-5-j,-3+0.14j,-3-0.14j];
K_old = place(A,B,p);
L_old = place(A',C',p*10)';

[t_old,x_old] = ode45(@(t,x) aircraftDynamics(t,x,A,B,C,K_old,L_old),
[0 10],xo); % Using old gains.

% By inspecting the plots, the LQR/LQG design seems to be much more
% efficient.

 figure(5),clf;

 subplot(4,1,1);
```

```
plot(tk,xk(:,5));
hold on
plot(t_old,x_old(:,5));
title("Airspeed")
xlabel("time (s)");
ylabel("m/s");
legend("LQR/LQG", "Old Design")

subplot(4,1,2);
plot(tk,xk(:,6));
hold on
plot(t_old,x_old(:,6));
title("Angle of Attack")
xlabel("time (s)");
ylabel("rads");
 legend("LQR/LQG", "Old Design")

subplot(4,1,3);
plot(tk,xk(:,7));
hold on
plot(t_old,x_old(:,7));
title("Pitch Rate")
xlabel("time (s)");
ylabel("rads/s");
legend("LQR/LQG", "Old Design")

subplot(4,1,4);
plot(tk,xk(:,8));
hold on
plot(t_old,x_old(:,8));
title("Pitch")
xlabel("time (s)");
ylabel("rads");
legend("LQR/LQG", "Old Design")

function dxdt = aircraftDynamics(t,x,A,B,C,K,L)

   z = x(1:4);
   zh = x(5:8);


   if isempty(K)
       u = zeros(2,1);
   else
       u = -K*zh;
   end

   z_dot = A*z + B*u;

   if isempty(L)
       zh_dot = z_dot;
   else
       zh_dot = A*zh + B*u - L*C*(zh -z);
   end
```

```
    dxdt = [z_dot;zh_dot];

end
```



Published with MATLAB® R2018a