Table of Contents

Question 8 1
a 2
b 3
c) 6
d 7

Question 8

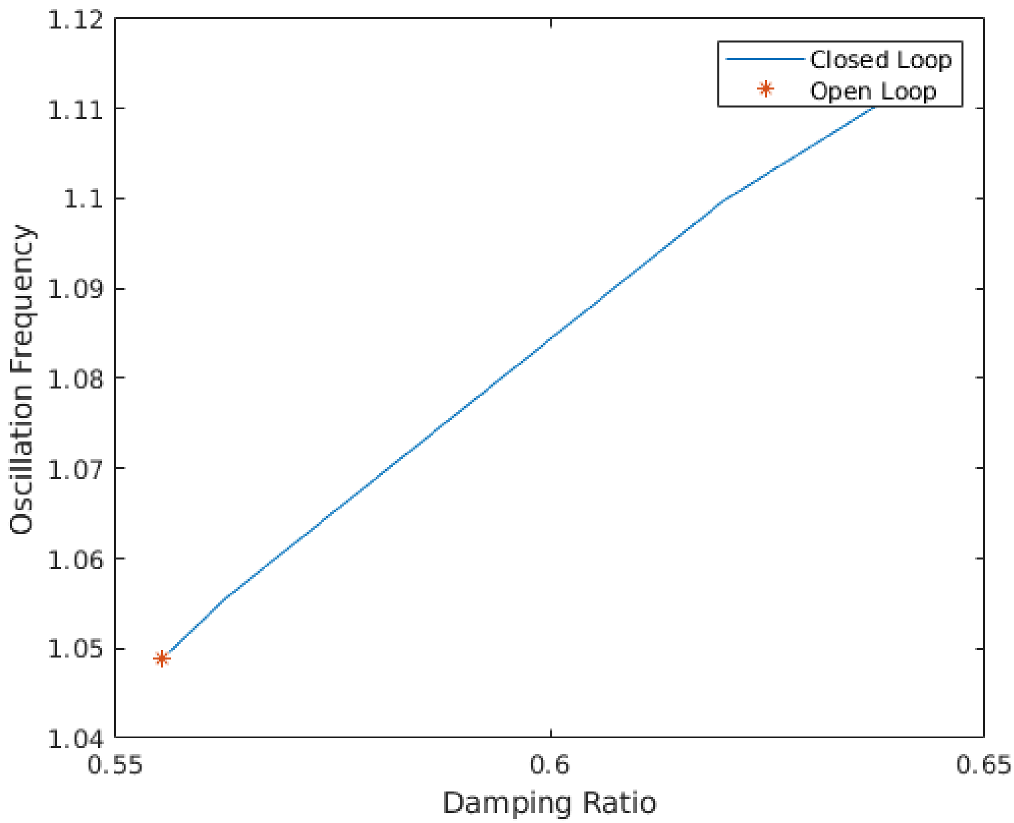
Xo = [250,0,0,0]';

```
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.
```

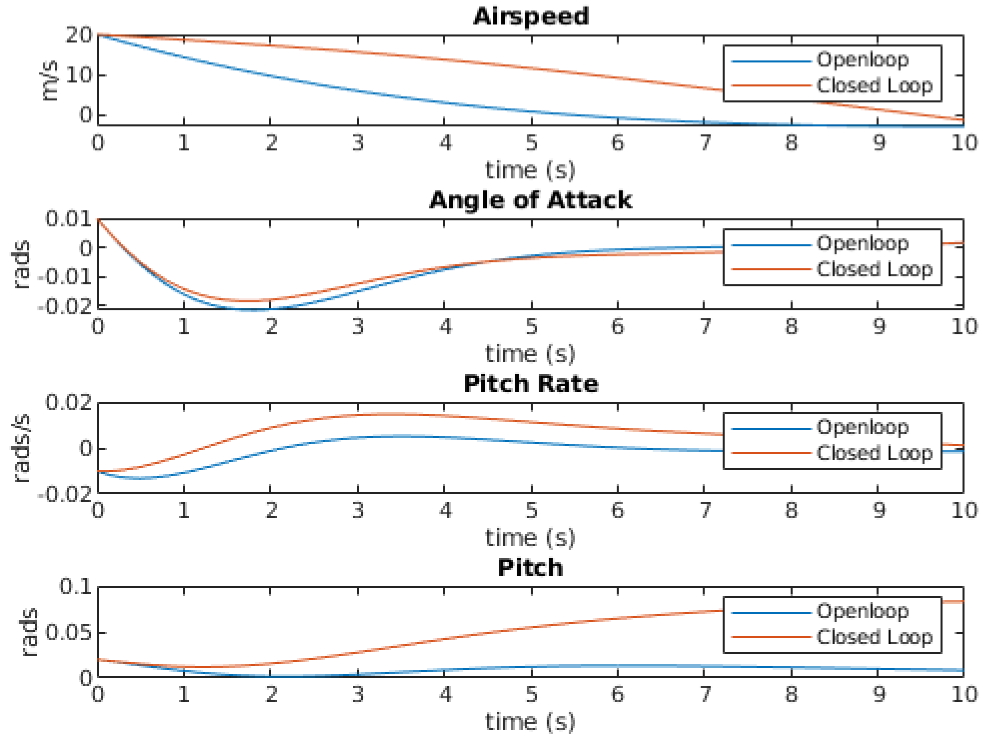


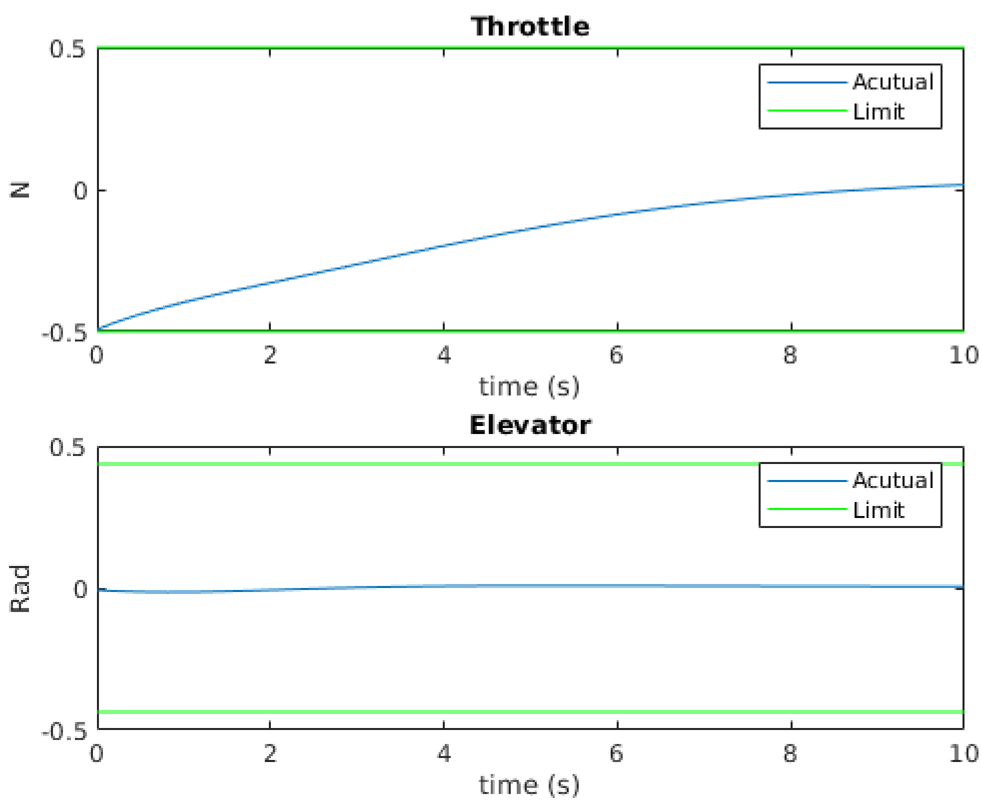
b

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
Openloop
Closed Loop
time (s)
Angle of Attack
rads
Openloop
Closed Loop
time (s)
Pitch Rate
rads/s
Openloop
Closed Loop
time (s)
Pitch
rads
Openloop
Closed Loop
time (s)

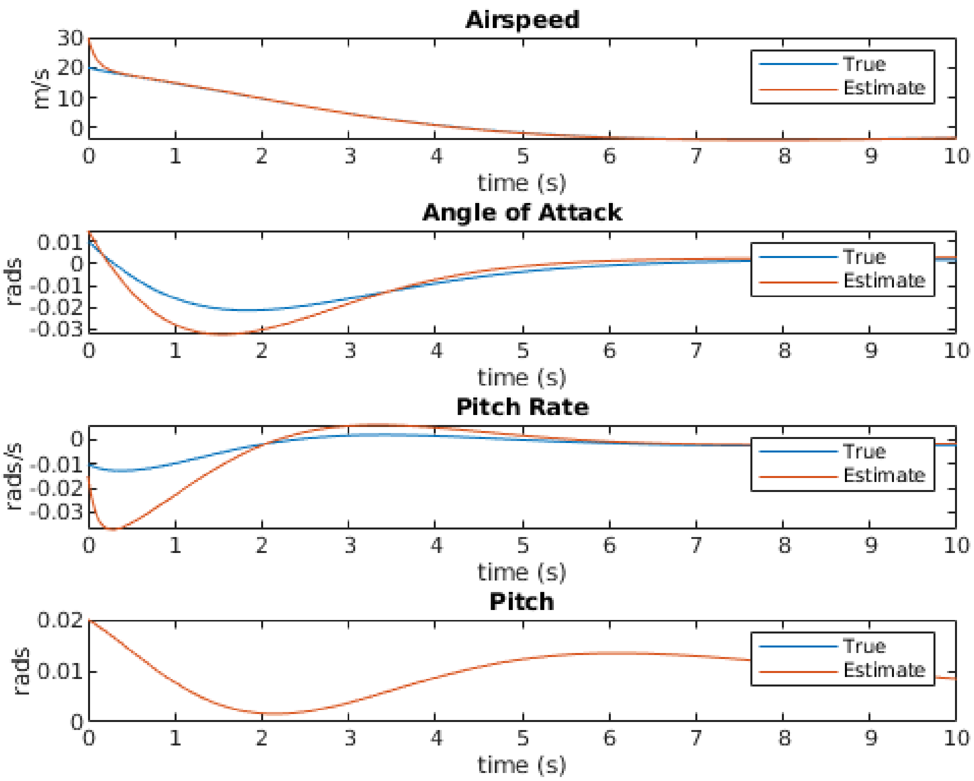

Throttle
N
Acutual
Limit
time (s)
Elevator
Rad
Acutual
Limit
time (s)

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.

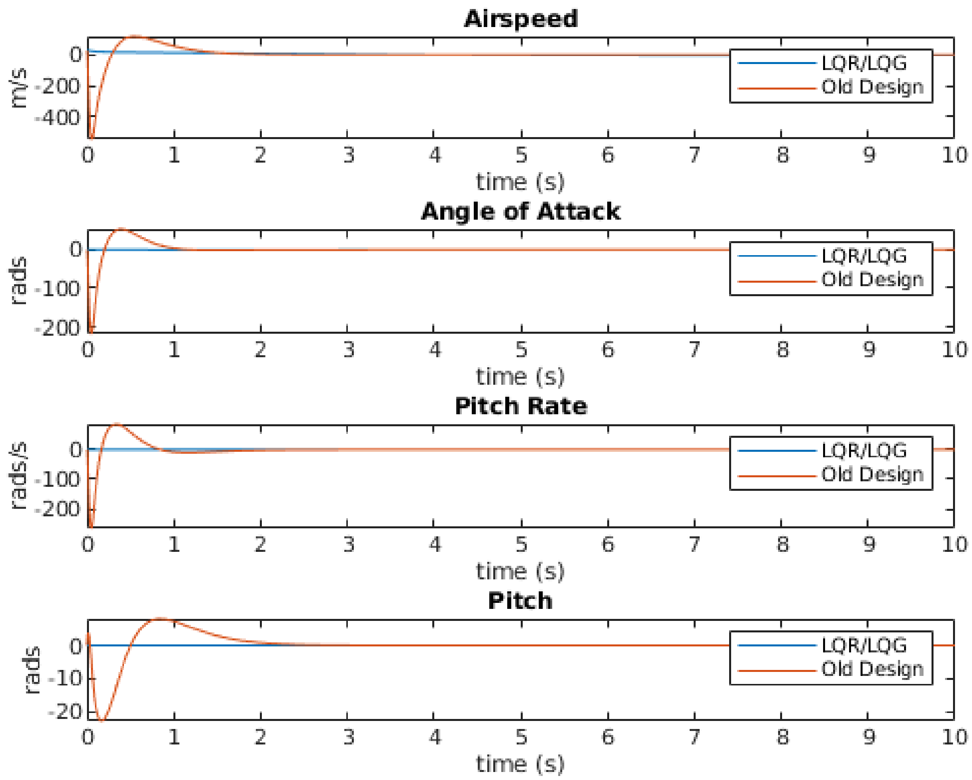
```
p = [-5+j,-5-j,-3+0.14j,-3-0.14j];
K_old = place(A,B,p);
L_old = place(A',C',p*10)';

[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