```
clc
clear
close all

nfig = 0;

A = @(t) [0 1; -(2+t) -3];
B = [0;1]; C = [1 0];
Q = [0.5]*1000;
r = 1; R = 0.5*r;
F = 0;
ti =0; tf=10;
x0  =[0;0];


%desired trajectory
td = ti:(tf-ti)/100:tf;
zd = ones(size(td)); % each row is desired output at that time
zd = 2*(td)'; tf =1; % smaller time for this

nfig = nfig+1; figure(nfig)
plot(td,zd,'r','LineWidth',3)
xlabel('time')
ylabel('zd')
set(gca,'FontSize',20)




% Find the controller gain K and G
[tK,K,G,nfig] = Psolve(A,B,C,Q,R,F,ti,tf,td,zd,nfig);

% plot the control gain K
nfig = nfig+1; figure(nfig)
plot(tK,K,'r','LineWidth',3)
xlabel('time')
ylabel('Kalman gain K')
set(gca,'FontSize',20)

% simulate the sytem response using ode45
options=odeset('RelTol',1e-10);
[t,x]=ode45(@(t,x) xdiff(t,x,flag,A,B,R,tK,K,G),[0 (tf-ti)],x0,options);

% plot the state x
nfig = nfig+1; figure(nfig)
plot(t,x,'b','LineWidth',3)
xlabel('time')
ylabel('State x')
set(gca,'FontSize',20)

% plot the state x
nfig = nfig+1; figure(nfig)
```

```
plot(t,x(:,1),'b',t,x(:,2),'r','LineWidth',3)
xlabel('time')
ylabel('State x')
legend('x_1','x_2')
set(gca,'FontSize',20)

%return
% find and plot the input u

[m,n]=size(x);     % Length of time, x is m
[mR,nR] = size(R); % number of inputs = mR
Kt = interp1(tK,K,t); % interpolate the K matrix
Gt = interp1(tK,G,t); % interpolate the G matrix
%
u = zeros(m,mR);    % initialize the input
for jj=1:1:m
    u(jj) = -Kt(jj,:)*(x(jj,:))' +inv(R)*B'*(Gt(jj,:))';
end
nfig = nfig+1; figure(nfig); clf
plot(t,u,'r','LineWidth',3)
xlabel('time')
ylabel('Input u')
set(gca,'FontSize',20)



return









%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Function to find the gain matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [tK,K,G,nfig] = Psolve(A,B,C,Q,R,F,ti,tf,td,zd,nfig)
%PSolve - Compute continuous-time solution to the Riccati Equation solved
 backward in time.
%
%   [t,K] = Psolve(A,B,Q,R,ti,tf) computes the gain matrix K
%   K = -R^(1) B' P
%   and G(t)
%   by solving for the symmetric Riccati matrix P
%   from the Riccati equation
%       .
%       P = - PA  -A'P  -V + PBR^(-1)B'P
%   with final condition, i.e. P(tf) = F.
%   using Backward integration
```

```
%       .
%       P = + PA  +A'P  +V - PEP
%   where E = BR^(-1)B'; V = C'QC;
%   and  initial Condition P(0) = F
%   then needs to be reversed in time
%   Also compute g(t)
%       .
%       g = -[A  -EP]'g - C'Q z(t)
%    See also Pdiff.
%
[m,n] = size(A); NT = n*(n+1)/2;
E = B*inv(R)*B';
options=odeset('RelTol',1e-12);

% Fiding final P in a vector form using F
Ptf = zeros(NT,1);
PMtf = C'*F*C;
k = 1;
for i=1:n
    for j=i:n
        Ptf(k) = PMtf(i,j);
        k = k+1;
    end
end

ztf = interp1(td,zd,tf);
gtf = C'*F*(ztf');


% combined vector
PGtf = [Ptf; gtf];

% Solving for P in a vector form PV

[t,PVG]=ode45(@(t,p) PVGdiff(t,p,flag,A,B,C,Q,R,F,tf,td,zd),[0 (tf-
ti)],PGtf,options);
%
%
% PV is in vector form, each row corresponds to row in time t
% flip the PV vector
PVG = flipud(PVG);
% redefine the time vector
t = flipud(t);
t = -t +(tf)*ones(size(t));
%



%
% computing the gain matrix K(t) as row vector
%
PV = PVG(:,1:NT);
G = PVG(:,NT+1:NT+n);
```

```
% plot the riccati matrix terms
nfig = nfig+1; figure(nfig)
plot(t,PV,'r','LineWidth',3)
xlabel('time')
ylabel('Riccati matrix')
set(gca,'FontSize',20)

% plot the riccati matrix terms
nfig = nfig+1; figure(nfig)
plot(t,G,'r','LineWidth',3)
xlabel('time')
ylabel('G matrix')
set(gca,'FontSize',20)


[mP,nP] = size(PVG);
K = zeros(mP,n);
G = zeros(mP,n);
tK =t;


%
for jj = 1:1:mP
    % find P matrix at time jj
    Pjj = zeros(n);
    for i=1:n
        for j=i:n
            k = i*n - i*(i-1)/2 - (n-j);
            Pjj(i,j) = PVG(jj,k);
            Pjj(j,i) = Pjj(i,j);
        end
    end
    % computing the feedback gain matrix
    K(jj,:) = inv(R)*B'*Pjj;
    G(jj,:) = PVG(jj,NT+1:NT+n);
end
end





%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Function to find the solution to Riccati Eq
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function PVGdiff = PVGdiff(t,p,flag,A,B,C,Q,R,F,tf,td,zd)
%
%    Called by PSolve to compute the Riccati matrix P.
%
%    See also PSOLVE.
%
%
% Finding the P matrix PM
```

```
A = A(t);
[m,n] = size(A);
NT = n*(n+1)/2;
PM = zeros(n);
E = B*inv(R)*B';
%
%
% Finding the P matrix PM
for i=1:n
    for j=i:n
        k = i*n - i*(i-1)/2 - (n-j);
        PM(i,j) = p(k);
        PM(j,i) = PM(i,j);
    end
end
%
%
% computing P matrix derivative
% Backward in time
PM_diff = A'*PM + PM*A -PM*E*PM +C'*Q*C;
%
zdt = interp1(td,zd,tf-t);
gt = p(NT+1: NT+n,1);
%
Gdiff = (A -E*PM)'*gt +C'*Q*zdt;
%
%
% computing P vector derivative
PVdiff = zeros(NT,1);
k = 1;
for i=1:n
    for j=i:n
        PVdiff(k) = PM_diff(i,j);
        k = k+1;
    end
end

PVGdiff = [PVdiff; Gdiff];

end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Function to find the solution to system with optimal control
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function xdiff = xdiff(t,x,flag,A,B,R,tK,K,G)
%
% solving xdot = AX + Bu
A = A(t)
Kt = interp1(tK,K,t);
Gt = interp1(tK,G,t);
ut = -Kt*x +inv(R)*(B')*Gt';
xdiff = A*x +B*ut;
end
```

```
Index in position 1 exceeds array bounds. Index must not exceed 3.

Error in HW6_2>PVGdiff (line 235)
gt = p(NT+1: NT+n,1);

Error in HW6_2>@(t,p)PVGdiff(t,p,flag,A,B,C,Q,R,F,tf,td,zd) (line 140)
[t,PVG]=ode45(@(t,p) PVGdiff(t,p,flag,A,B,C,Q,R,F,tf,td,zd),[0 (tf-
ti)],PGtf,options);

Error in odearguments (line 92)
f0 = ode(t0,y0,args{:});   % ODE15I sets args{1} to yp0.

Error in ode45 (line 107)
  odearguments(odeIsFuncHandle,odeTreatAsMFile, solver_name, ode, tspan, y0,
 options, varargin);

Error in HW6_2>Psolve (line 140)
[t,PVG]=ode45(@(t,p) PVGdiff(t,p,flag,A,B,C,Q,R,F,tf,td,zd),[0 (tf-
ti)],PGtf,options);

Error in HW6_2 (line 31)
[tK,K,G,nfig] = Psolve(A,B,C,Q,R,F,ti,tf,td,zd,nfig);
```

Published with MATLAB® R2022b