```
clc
clear
close all

% Sys J
A = @(t) -(1+t); B = 1; C = 1; D = 0;
% Weight
Q = 1; R = 1; F = Q*0;
% Initial
x0 = 5;
ti = 0; tf = 2;

% Kalman gain
[tK, K] = Psolve(A, B, Q, R, F, ti, tf);
% Plot K(t)
figure(1)
plot(tK, K, 'r', 'LineWidth', 3)
title('Kalman Gain K(t)')
xlabel('Time')
ylabel('Gain K')
set(gca, 'FontSize', 20)

% ODE45
options = odeset('RelTol', 1e-10);
[t, x] = ode45(@(t, x) xdiff(t, x, flag, A, B, tK, K),[0 (tf - ti)], x0,
 options);

% Plot x(t)
figure(2)
plot(t, x, 'b', 'LineWidth', 3)
title('State x(t)')
xlabel('Time')
ylabel('State x')
set(gca, 'FontSize', 20)

% Plot u(t)
[m, n] = size(x);
[mR, nR] = size(R);
Kt = interp1(tK, K, t);
u = zeros(m, mR);
for jj = 1:1:m
    u(jj) = -Kt(jj, :)*(x(jj, :))';
end
figure(3)
plot(t, u, 'r', 'LineWidth', 3)
title('Input u(t)')
xlabel('Time')
ylabel('Input u')
set(gca, 'FontSize', 20)

% x(2)
index = find(t == 2);
```

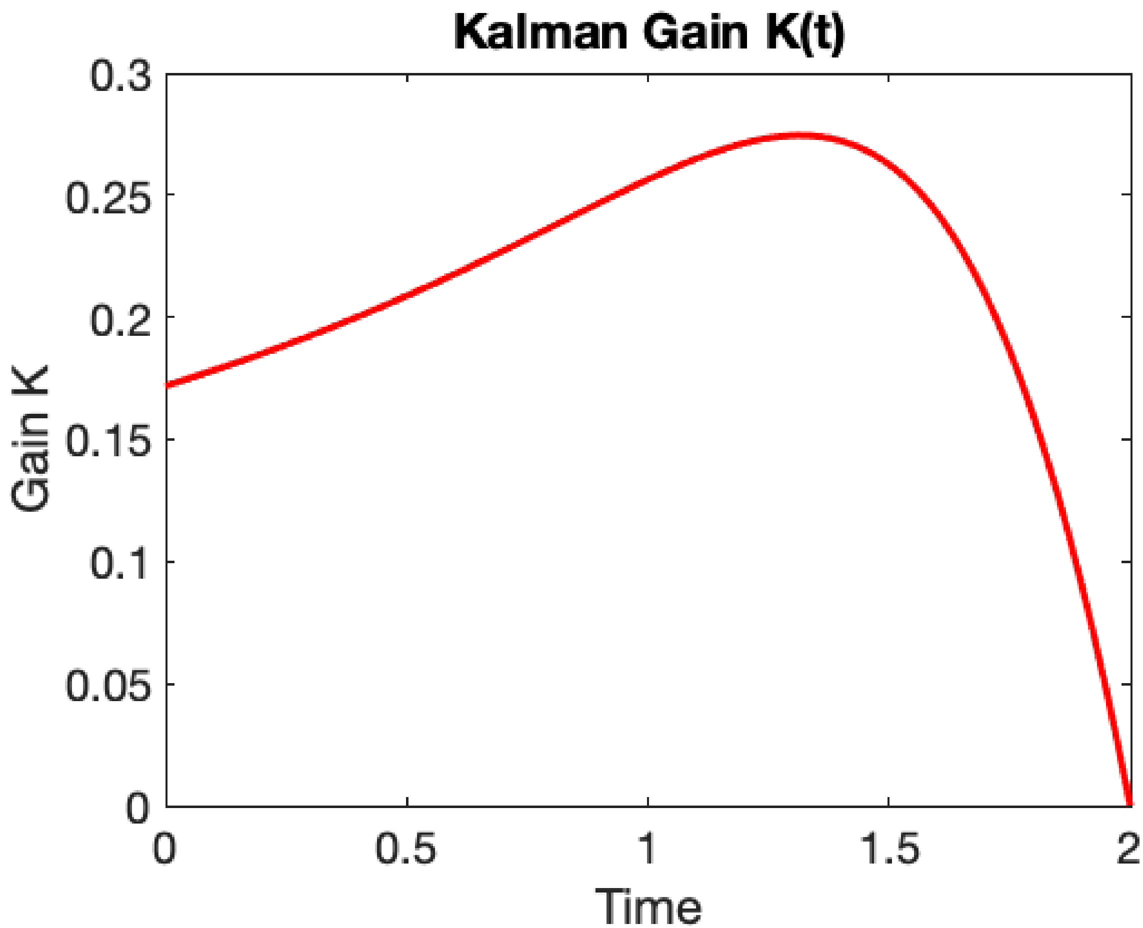
```
x_2 = x(index);
disp(['x(2) = ' num2str(x_2)])

% Sys J with penalty
A_ = @(t) -(1+t); B_ = 1; C_ = 1; D_ = 0;
% Weight
Q_ = 1; R_ = 1; F_ = 10;
% Initial
x0_ = 5;
ti_ = 0; tf_ = 2;

% Kalman gain
[tK_, K_] = Psolve(A_, B_, Q_, R_, F_, ti_, tf_);
% Plot K(t)
figure(4)
plot(tK, K, 'b', tK_, K_, 'r', 'LineWidth', 3)
title('Kalman Gain K(t) vs K_p(t)')
xlabel('Time')
ylabel('Gain K')
set(gca, 'FontSize', 20)

% ODE45
options = odeset('RelTol', 1e-10);
[t_, x_] = ode45(@(t_, x_) xdiff(t_,x_,flag,A_,B_,tK_,K_),[0 (tf_ - ti_)],
 x0_, options);

% Plot x(t) vs x_p(t)
figure(5)
plot(t, x,'b', t_, x_, 'r', 'LineWidth', 3)
title('State x(t) vs x_p(t)')
xlabel('Time')
ylabel('State x')
set(gca, 'FontSize', 20)

% Plot u(t) vs u_p(t)
[m_, n_] = size(x_);
[mR_, nR_] = size(R_);
Kt_ = interp1(tK_, K_, t_);
u_ = zeros(m_, mR_);
for jj_=1:1:m_
    u_(jj_) = -Kt_(jj_, :)*(x_(jj_, :))';
end
figure(6)
plot(t, u, 'b', t_, u_, 'r', 'LineWidth', 3)
title('Input u(t) vs u_p(t)')
xlabel('Time')
ylabel('Input u')
set(gca, 'FontSize', 20)

% x(2) for penalty case
index_ = find(t_ == 2);
x_2_ = x_(index_);
disp(['x_p(2) = ' num2str(x_2_)])
```

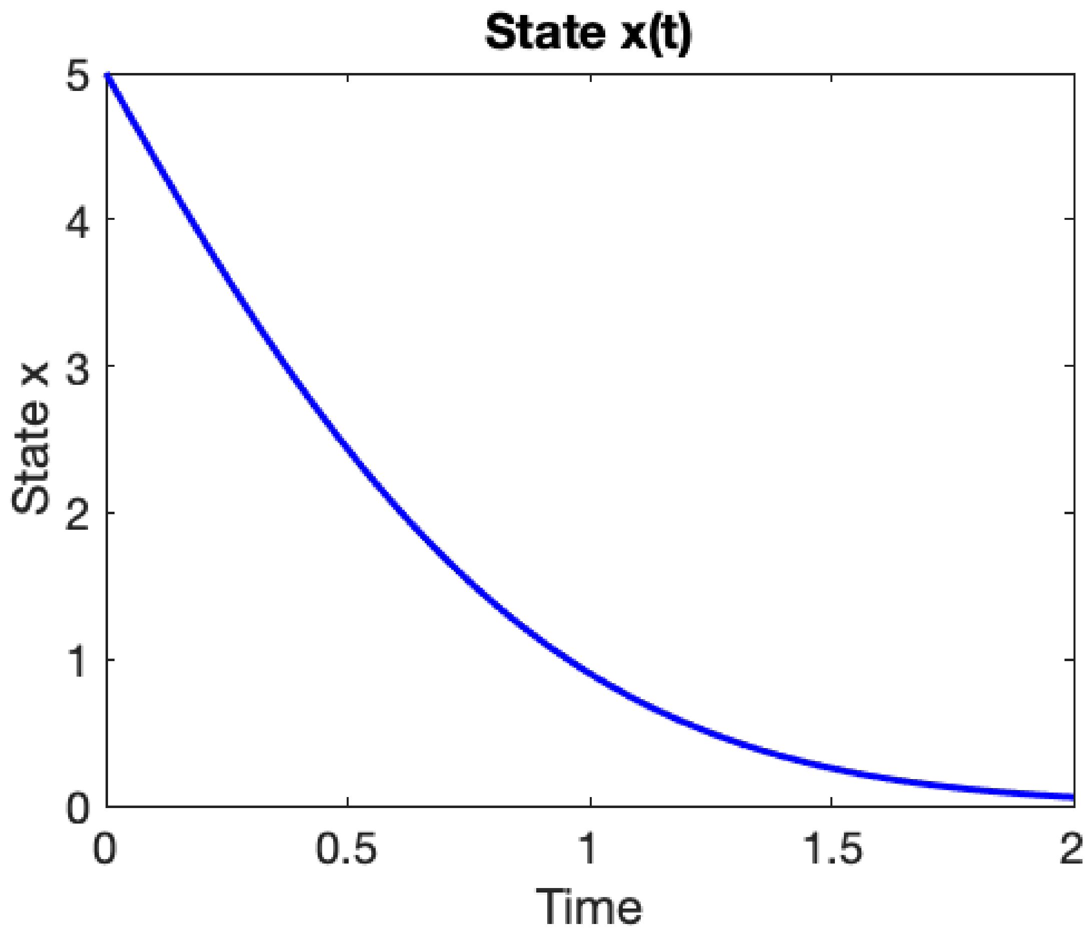
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Function to find the gain matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [t,K] = Psolve(A,B,Q,R,F,ti,tf)
%PSolve - Compute continuous-time solution to the Riccati Equation solved
 backward in time.
%
%   [t,K] = Psolve(A,B,Q,R,ti,tf) computes the gain matrix K
%   K = -R^(1) B' P
%   by solving for the symmetric Riccati matrix P
%   from the Riccati equation
%       .
%       P = - PA  -A'P  -Q + PBR^(-1)B'P
%   with final condition, i.e. P(tf) = F.
%   usig Backward integration
%       .
%       P = + PA  +A'P  +Q - PEP
%   where E = BR^(-1)B'
%   and  initial Condition P(0) = F
%   then needs to be reversed in time
%    See also Pdiff.
%
[m,n] = size(A); NT = n*(n+1)/2;
E = B*inv(R)*B';
options=odeset('RelTol',1e-10);

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

% Solving for P in a vector form PV

[t,PV]=ode45(@(t,p) PVdiff(t,p,flag,A,B,Q,R,E,F),[0 (tf-ti)],Ptf,options);
%
%
% PV is in vector form, each row corresponds to row in time t
% flip the PV vector
PV = flipud(PV);
% redefine the time vector
t = flipud(t);
t = -t +(tf)*ones(size(t));
%
%
% computing the gain matrix K(t) as row vector
%
[mP,nP] = size(PV);
K = zeros(mP,n);
%
```

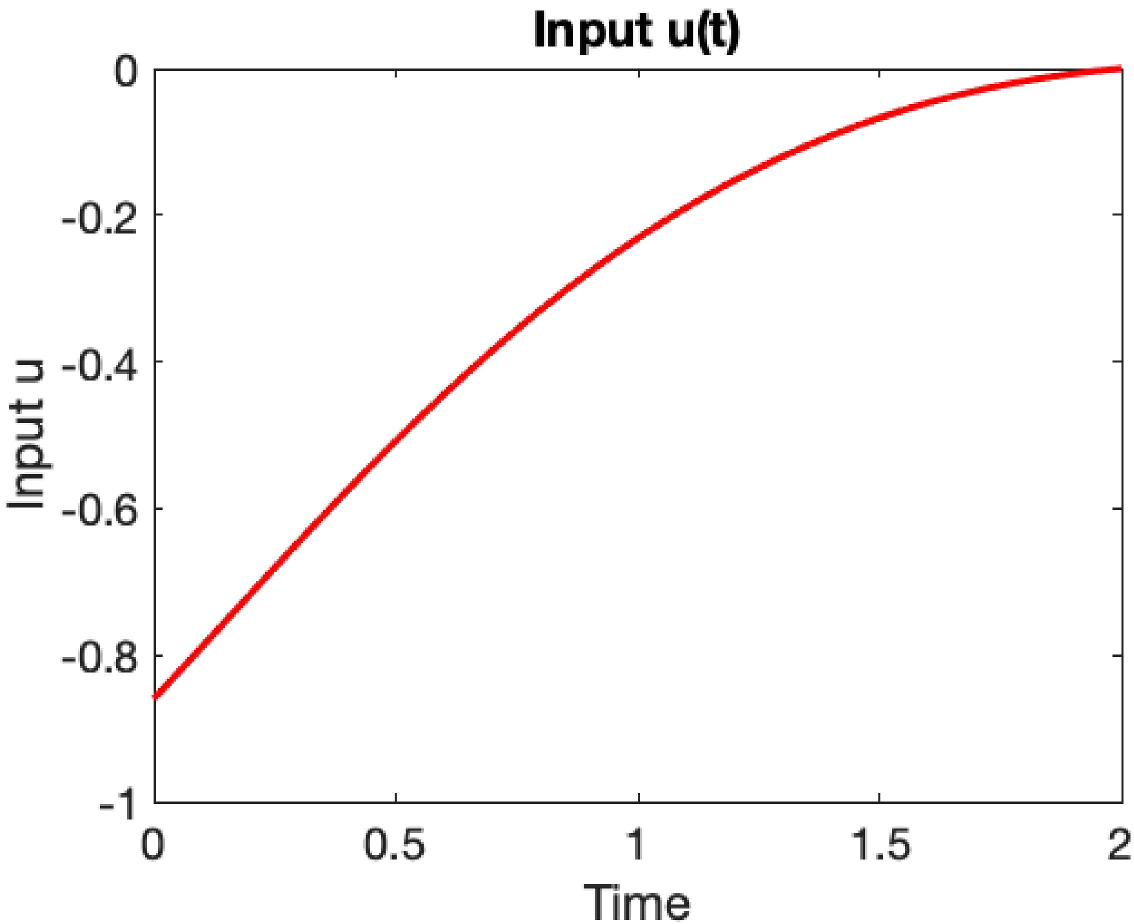
```
for jj = 1:1:mP
    % find P matrix at time jj
    Pjj = zeros(n);
    for i=1:n
        for j=i:n
            k = i*n - i*(i-1)/2 - (n-j);
            Pjj(i,j) = PV(jj,k);
            Pjj(j,i) = Pjj(i,j);
        end
    end
    % computing the feedback gain matrix
K(jj,:) = inv(R)*B'*Pjj;
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Function to find the solution to Riccati Eq
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function PVdiff = PVdiff(t,p,flag,At,B,Q,R,E,F)
%
%    Called by PSolve to compute the Riccati matrix P.
%
%    [t,GP] = ode45('gramdiff',[ti tf],zeros(NT,1),[],A,B,Q,R,E).
%
%    See also PSOLVE.
%
%

A = At(t);

% Finding the P matrix PM
[m,n] = size(A);
NT = n*(n+1)/2;
PM = zeros(n);
%
%
% 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 +Q;
%
%
% computing P vector derivative
PVdiff = zeros(NT,1);
k = 1;
```

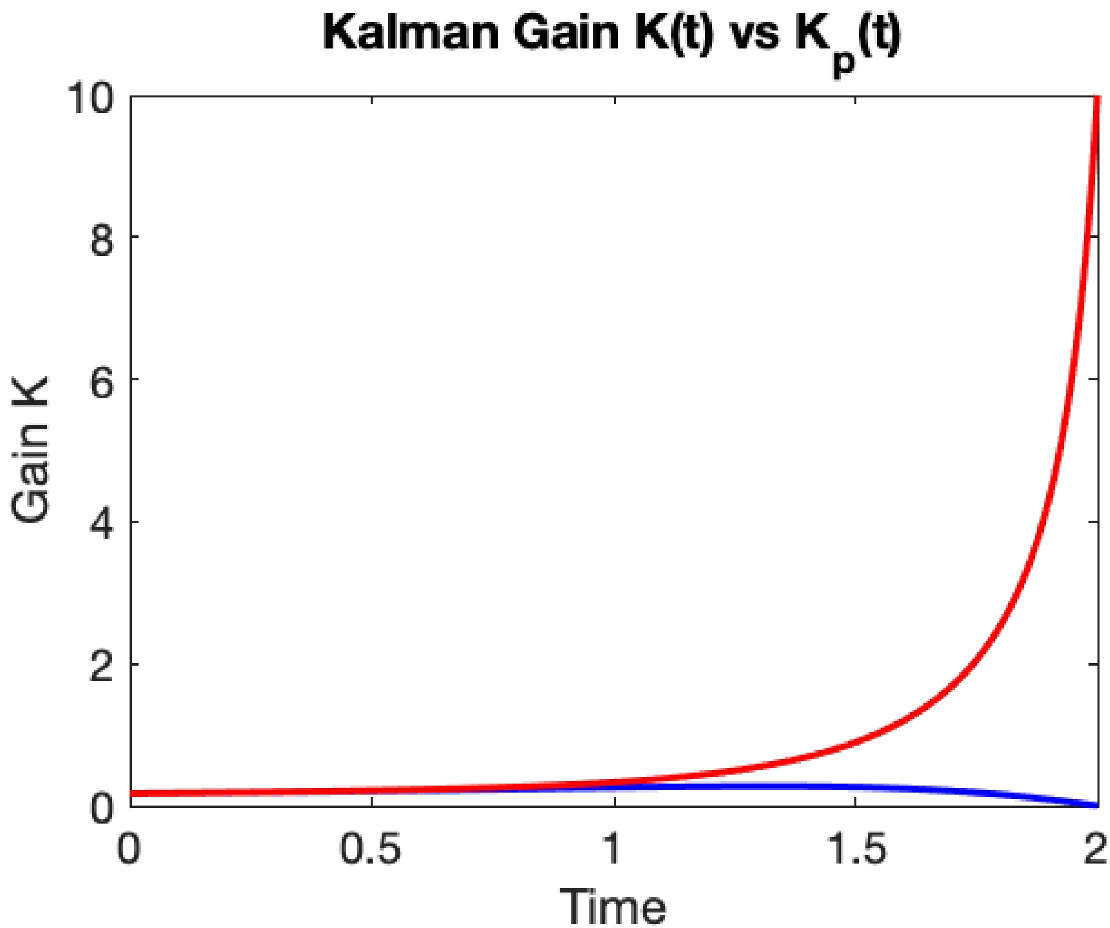
```
for i=1:n
    for j=i:n
        PVdiff(k) = PM_diff(i,j);
        k = k+1;
    end
end
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Function to find the solution to system with optimal control
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function xdiff = xdiff(t,x,flag,At,B,tK,K)
%
% solving xdot = AX + Bu

A = At(t);

Kt = interp1(tK,K,t);
ut = -Kt*x;
xdiff = A*x +B*ut;
end
```

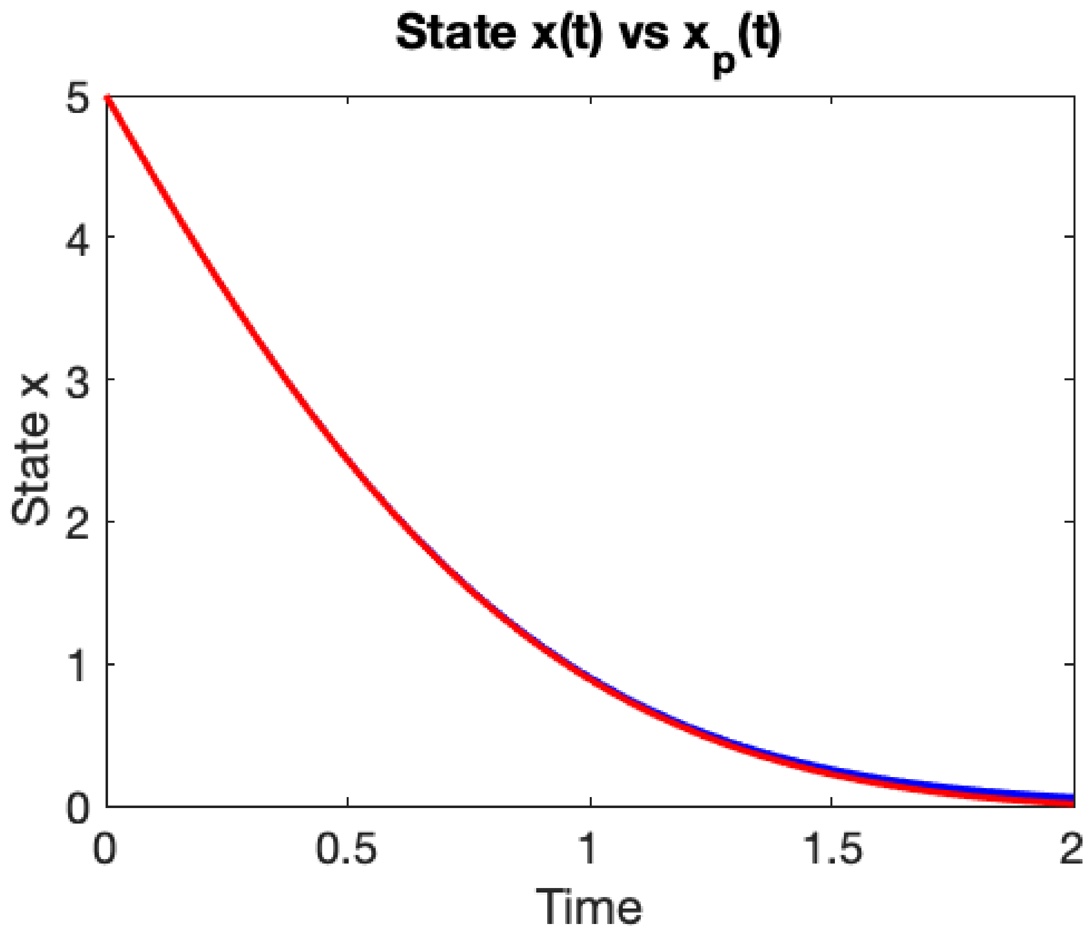
x(2) = 0.059633
x_p(2) = 0.013395

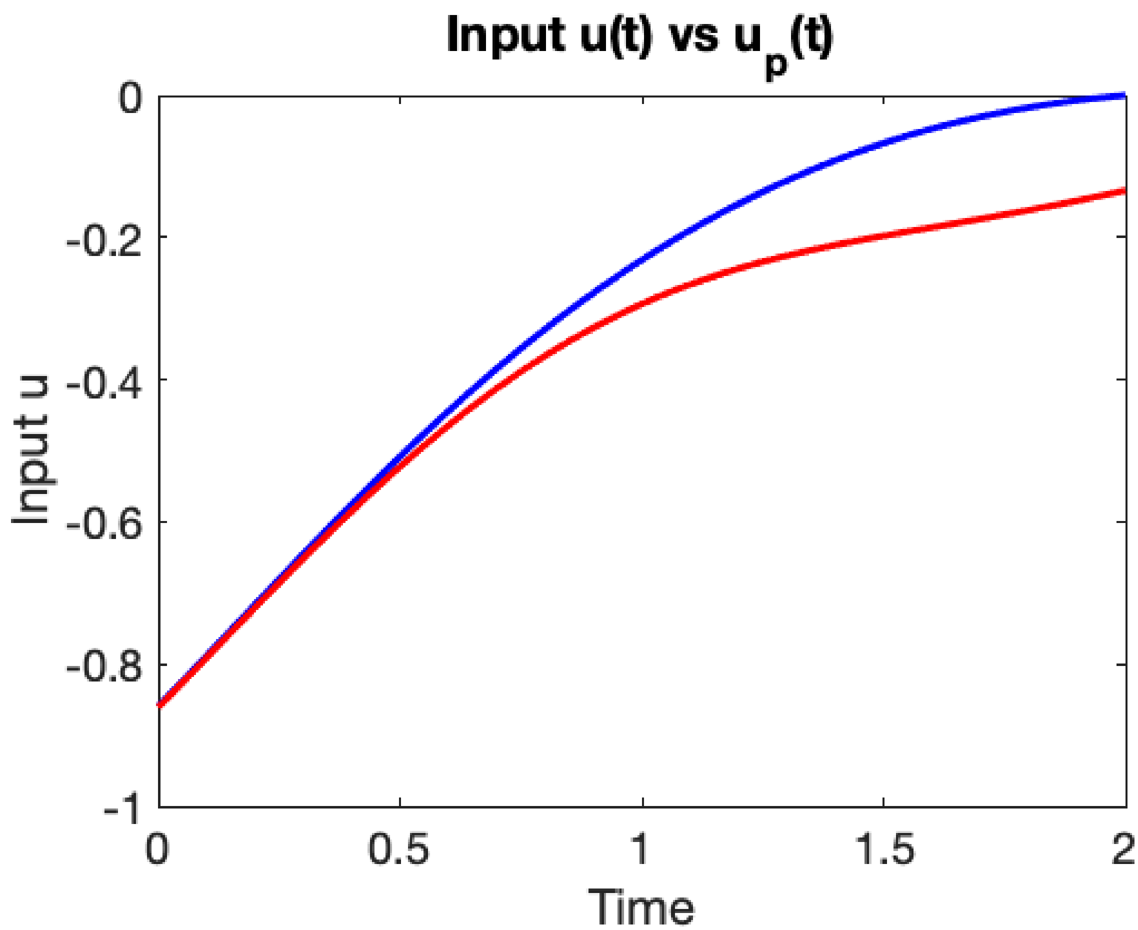

Kalman Gain K(t)
Gain K
Time
0.3
0.25
0.2
0.15
0.1
0.05
0
0
0.5
1
1.5
2


State x(t)
State x
Time
0
0.5
1
1.5
2
0
1
2
3
4
5


Input u(t)
0
-0.2
-0.4
-0.6
-0.8
-1
Input u
0
0.5
1
1.5
2
Time


Kalman Gain K(t) vs K_p(t)
Gain K
Time
10
8
6
4
2
0
0
0.5
1
1.5
2


State x(t) vs x_p(t)
State x
Time
5
4
3
2
1
0
0
0.5
1
1.5
2



Published with MATLAB® R2022b