Contents


```
%{
The following code is used to design an optimal controller and estimator for 2 pendulem of
different length and mass on a single cart. The only acutator on this car
the motor that drives the linear position of the cart.

We are attempting to swing up both the pendulem from the inital position of
all pendulem down to both pendulem up and stabilize

%}
```

Determine u_k and state

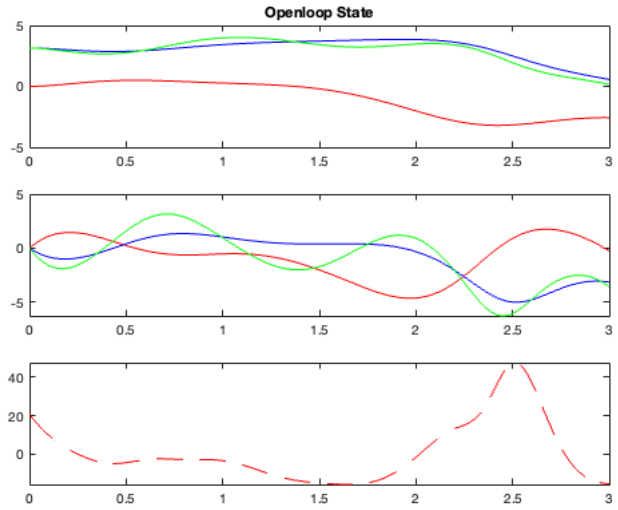
```
clear all; clc; close all
load_prev = 1; %set to 1 if a previous trajectory has been generated

%{
Use code provided in NR Ch21 to generate x_k state, u_k, using adjoint
method for the swing up phase. This phase last about T = 3 seconds because

0) Within this code, tune QT to drive error at T close to zero
1) Warm start: use an inital guess of u_k being all zeros
2) iterate the provided function with a new u_k until the state reach near
desired value of [0 0 0 0 0 0]
3)save x_k and u_k

%}
```

Uncomment the Following Code if you need an x_k and u_k using Adjoint method to generate swingup states trajectory

```
% T = 3; %approx sqrt(g/l)
%
% %initial guess of u_k being all zero
% [u_k(:,1),x_k] = OptimizeTraj(T);
%
% state_at_T(:,1) = x_k(1:6, end);
%
% number_of_trial = 1;
% trial_number = 1:number_of_trial
% for i = trial_number
%     [u_k(:,i+1), x_k] = OptimizeTraj(T, u_k(:,i))
%     state_at_T(:,i+1) = x_k(1:6, end)
% end
```

If Previous Trajectory has been generated, run this code

```
%Trjectory can be manually optimzed by tuning Q iteration by interation by
%using the file "AdjointPathManualIteration.m"

%load previous trajectory is better

if load_prev == 1
    clear all; clc;
    load("SavePoint3.mat")
    figure(1); clf; subplot(3,1,1); plot(t,x_k(1,:),'r-',t,x_k(2,:),'b-',t,x_k(3,:),'g-'); hold on; title("Openloop State");
              subplot(3,1,2); plot(t,x_k(4,:),'r-',t,x_k(5,:),'b-',t,x_k(6,:),'g-'); hold on;
              subplot(3,1,3); plot(t,u_k,'r--'); hold on;

end
```



Find A(t), E(t), B(t) for Each x_k in the swing up phase

```
%From generated state trajectory, compute for the state matrix E_t, A_t,
%B_t, C_t, D_t wrt time for every timestep h since the system is not
%linearizable during the swing up case

s.h=0.01; s.N=T/s.h; s.mc=10; t=[0:s.N]*s.h; % STEP 0: initialize simulation
s.m1=1; s.L1=1;    s.ell1=s.L1; s.I1=s.m1*s.ell1^2/3; % system, & derived parameters
s.m2=0.5; s.L2=0.5;  s.ell2=s.L2; s.I2=s.m2*s.ell2^2/3; alpha=0.1;
s.B=[0; 0; 0; 1; 0; 0]; s.Q=diag([0 0 0 0 0 0]); s.R=0; s.QT=diag([5 40 10 .1 60 10]);
g = 9.81;

for i = 1:length(x_k)
    A_temp = Compute_A(x_k(:,i) ,s); %NR funtion
    A_t(i) = {A_temp};
end

for i = 1:length(x_k)
    E_temp = Compute_E(x_k(1:6,i),s); %NR funtion
    E_t(i) = {E_temp};
end

for i = 1:length(x_k)
    B_temp = [0; 0; 0; 1; 0; 0]; %NR funtion
    B_t(i) = {B_temp};
end

for i = 1:length(x_k)
    C_temp = [[1 0 0  0   0   0];... %Sending x, theta1, theta2
              [0 1 0  0   0   0];...
              [0 0 1  0   0   0];...
              [0 0 0  0   0   0];...
              [0 0 0  0   0   0];...
              [0 0 0  0   0   0]];
    C_t(i) = {C_temp};
end
```

Compute K(t) using A(t), E(t), B(t), C(t)

```
%{
 compute controller gain K(t) using differential ricatti equation (DRE),
 which can be solved using RK4 marching. This K(t) controls the the system from
 one timestep to another

%}
h = s.h; %timestep

R = .1; %controller cost
Q = [[1 0 0  0   0   0];... %penilize theta anlges more than other states
     [0 7 0  0   0   0];...
     [0 0 7  0   0   0];...
     [0 0 0  1   0   0];...
     [0 0 0  0   1   0];...
     [0 0 0  0   0   1]];

X_t = {}; %allocate cell frame to store X at every time frame
X_t(length(u_k)) = {eye(6)}; %X at T
```

```
%Backward March to find X(t)using RK4 + DRE
for i = length(u_k):-1:2

f1 = KRiccati(E_t{i},A_t{i},B_t{i},X_t{i},R,Q);
f2 = KRiccati(E_t{i},A_t{i},B_t{i},X_t{i} - f1*h/2,R,Q);
f3 = KRiccati(E_t{i},A_t{i},B_t{i},X_t{i} - f2*h/2,R,Q);
f4 = KRiccati(E_t{i},A_t{i},B_t{i},X_t{i} - f3*h,R,Q);

X_t{i-1} = X_t{i} - h*(f1/6 + f2/3 + f3/3 + f4/6);

end

%find K(t) using X(t) using NR22.13a
for i = 1:length(u_k)
    K_t(i) = {-inv(R)*B_t{i}'*X_t{i}};
end
```

Compute L(t) using A(t), E(t), B(t), C(t)

```
%{
 compute observer gain L(t) using differential ricatti equation, which
 can be solved using RK4 forward marching
%}

Q1 = eye(6); %covariance of state disturbance
Q2 = eye(6); %convariance of sensor noise

P_t = {}; %allocate space for storage of P

P_t(1) = {eye(6)}; %intial estimation covariance

%forward march to find L(t) using RK4 and DRE NR22.30
for i = 1:1:length(u_k)

f1 = KRiccati(E_t{i},A_t{i},B_t{i},X_t{i},R,Q);
f2 = KRiccati(E_t{i},A_t{i},B_t{i},X_t{i} + f1*h/2,R,Q);
f3 = KRiccati(E_t{i},A_t{i},B_t{i},X_t{i} + f2*h/2,R,Q);
f4 = KRiccati(E_t{i},A_t{i},B_t{i},X_t{i} + f3*h,R,Q);

P_t{i+1} = P_t{i} + h*(f1/6 + f2/3 + f3/3 + f4/6);

end

%find L(t) using P(t) by appying NR22.30
for i = 1:length(u_k)
    L_t(i) = {-P_t{i}*C_t{i}'*inv(Q2)};
end
```

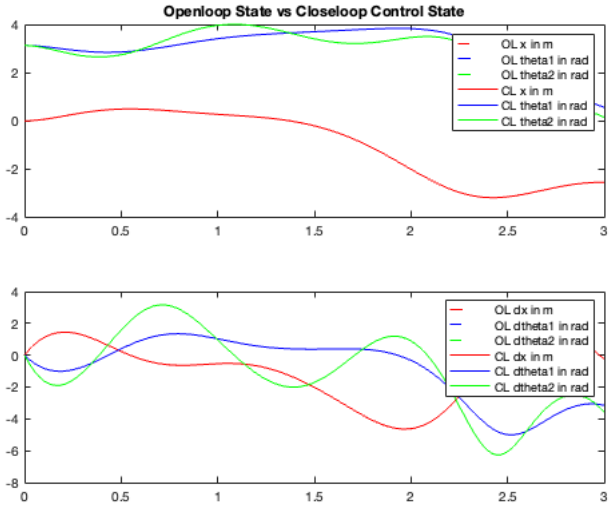
Uncomment this section to check eigenvalues of the control + observation system

```
% eigControl = {};
% ControlSys = {};
% for i = 1:length(u_k)
%     ControlSys(i) = {inv(E_t{i})*A_t{i} + inv(E_t{i})* B_t{i}*K_t{i}};
%    [evecK, eigControl{i}] = eig(ControlSys{i});
% end


% eigEstimate = {};
% EstimateSys = {};
% for i = 1:length(u_k)
%     EstimateSys(i) = {A_t{i} + L_t{i}*C_t{i}};
%    [evecL, eigEstimate{i}] = eig(EstimateSys{i});
% end
```

Plot State Trajectory using K(t) Controller

```
%From derivation, dx'/dt = inv(E) * [A + Bk - Bk*x_bar + Bu_bar]
%Note that u_k != K(t)*x_k because u_k is designed from openloop adjoint
%methods


% March protabation x_prime using RK4 and dx'/dt derivation
dx_prime = {};
x_prime(:,1) = [0 0 0 0 0 0]';
for i = 1:1:length(u_k)-1

    f1 = inv(E_t{i})*(A_t{i} + B_t{i})* x_prime(:,i) - inv(E_t{i})*B_t{i}*K_t{i}*x_k(1:6,i) + inv(E_t{i})*B_t{i}*u_k(i);
    f2 = inv(E_t{i})*(A_t{i} + B_t{i})* (x_prime(:,i) + f1*h/2) - inv(E_t{i})*B_t{i}*K_t{i}*x_k(1:6,i) + inv(E_t{i})*B_t{i}*u_k(i);
    f3 = inv(E_t{i})*(A_t{i} + B_t{i})* (x_prime(:,i) + f2*h/2) - inv(E_t{i})*B_t{i}*K_t{i}*x_k(1:6,i) + inv(E_t{i})*B_t{i}*u_k(i);
    f4 = inv(E_t{i})*(A_t{i} + B_t{i})* (x_prime(:,i) + f3*h) - inv(E_t{i})*B_t{i}*K_t{i}*x_k(1:6,i) + inv(E_t{i})*B_t{i}*u_k(i);
```

```
    x_prime(:,i+1) = x_prime(i) + h*(f1/6 + f2/3 + f3/3 + f4/6);
end

% Determine state simulation by adding nominal state and the proptabation states
% ideally, protabation tends to zero if controller is designed properly
% x_sim should ideally be very close to x_k
x_sim = x_k(1:6, :) + x_prime*h;


figure(2); subplot(2,1,1); plot(t,x_k(1,:),'r--',t,x_k(2,:),'b--',t,x_k(3,:),'g--'); hold on; title("Openloop State vs Closeloop Control State")
                        legend("OL x in m", "OL theta1 in rad", "OL theta2 in rad")
               subplot(2,1,2); plot(t,x_k(4,:),'r--',t,x_k(5,:),'b--',t,x_k(6,:),'g--'); hold on;
                        legend("OL dx in m", "OL dtheta1 in rad", "OL dtheta2 in rad")

figure(2); subplot(2,1,1); plot(t,x_sim(1,:),'r',t,x_sim(2,:),'b',t,x_sim(3,:),'g');
                        legend("OL x in m", "OL theta1 in rad", "OL theta2 in rad", "CL x in m", "CL theta1 in rad", "CL theta2 in rad")
                    subplot(2,1,2); plot(t,x_sim(4,:),'r-',t,x_sim(5,:),'b-',t,x_sim(6,:),'g-');
                        legend("OL dx in m", "OL dtheta1 in rad", "OL dtheta2 in rad", "CL dx in m", "CL dtheta1 in rad", "CL dtheta2 in rad")


%plots shows convergence of simulated close loop x and open loop x
```

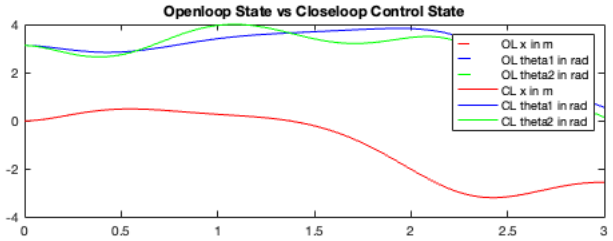


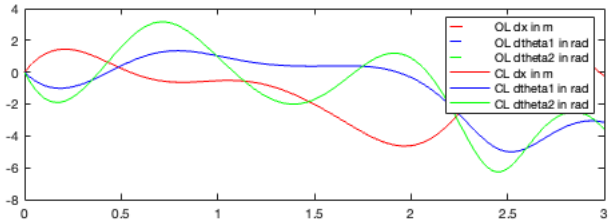
Add Estimation Technique to system using L(t)

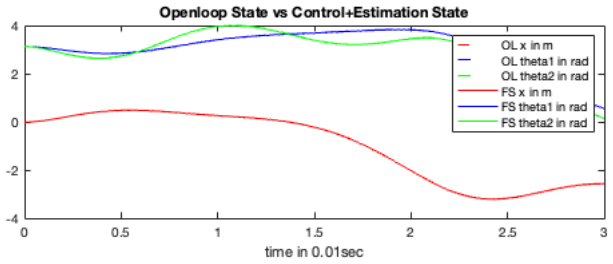
```
% March protabation x_prime using the following derivation
%{
dx_prime = (inv(E_t)*A_t + L_t*C_t)* x_hat_prime(:,i)...
            + (2*L_t*C_t - inv(E_t)*B_t*K_t)*x_k(1:6, i)...
            + (L_t*C_t + inv(E_t)*B_t*K_t)*x_prime(:,i)...
            + inv(E_t)*B_t*u_k(i);
%}

x_hat_prime(:,1) = [0 0 0 0 0 0]';
for i = 1:1:length(u_k)-1

     f1 = (inv(E_t{i})*A_t{i} + L_t{i}*C_t{i})* x_hat_prime(:,i)...
        + (2*L_t{i}*C_t{i} - inv(E_t{i})*B_t{i}*K_t{i})*x_k(1:6, i)...
        + (L_t{i}*C_t{i} + inv(E_t{i})*B_t{i}*K_t{i})*x_prime(:,i)...
        + inv(E_t{i})*B_t{i}*u_k(i);

     f2 = (inv(E_t{i})*A_t{i} + L_t{i}*C_t{i})* (x_hat_prime(:,i) + f1*h/2) ...
        + (2*L_t{i}*C_t{i} - inv(E_t{i})*B_t{i}*K_t{i})*x_k(1:6, i)...
        + (L_t{i}*C_t{i} + inv(E_t{i})*B_t{i}*K_t{i})*x_prime(:,i)...
        + inv(E_t{i})*B_t{i}*u_k(i);

     f3 = (inv(E_t{i})*A_t{i} + L_t{i}*C_t{i})* (x_hat_prime(:,i) + f2*h/2) ...
        + (2*L_t{i}*C_t{i} - inv(E_t{i})*B_t{i}*K_t{i})*x_k(1:6, i)...
        + (L_t{i}*C_t{i} + inv(E_t{i})*B_t{i}*K_t{i})*x_prime(:,i)...
        + inv(E_t{i})*B_t{i}*u_k(i);
     f4 = (inv(E_t{i})*A_t{i} + L_t{i}*C_t{i})* (x_hat_prime(:,i) + f3*h) ...
        + (2*L_t{i}*C_t{i} - inv(E_t{i})*B_t{i}*K_t{i})*x_k(1:6, i)...
        + (L_t{i}*C_t{i} + inv(E_t{i})*B_t{i}*K_t{i})*x_prime(:,i)...
        + inv(E_t{i})*B_t{i}*u_k(i);

    x_hat_prime(:,i+1) = x_hat_prime(i) + h*(f1/6 + f2/3 + f3/3 + f4/6);

end
```

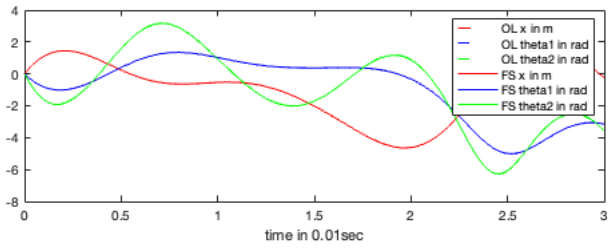
```
x_estimation_sim = x_k(1:6, :) + x_hat_prime*h;

figure(3); subplot(2,1,1); plot(t,x_k(1,:),'r--',t,x_k(2,:),'b--',t,x_k(3,:),'g--'); hold on; title("Openloop State vs Control+Estimation State")
                        legend("OL x in m", "OL theta1 in rad", "OL theta2 in rad")
                        xlabel("time in 0.01sec")
              subplot(2,1,2); plot(t,x_k(4,:),'r--',t,x_k(5,:),'b--',t,x_k(6,:),'g--'); hold on;
                        legend("OL dx in m", "OL dtheta1 in rad", "OL dtheta2 in rad", "OL x in m", "OL theta1 in rad", "OL theta2 in rad")
                        xlabel("time in 0.01sec")


figure(3); subplot(2,1,1); plot(t,x_estimation_sim (1,:),'r',t,x_estimation_sim (2,:),'b',t,x_estimation_sim (3,:),'g');
                legend("OL x in m", "OL theta1 in rad", "OL theta2 in rad", "FS x in m", "FS theta1 in rad", "FS theta2 in rad")
                xlabel("time in 0.01sec")
             subplot(2,1,2); plot(t,x_estimation_sim (4,:),'r',t,x_estimation_sim (5,:),'b',t,x_estimation_sim (6,:),'g');
                legend("OL x in m", "OL theta1 in rad", "OL theta2 in rad", "FS x in m", "FS theta1 in rad", "FS theta2 in rad")
                xlabel("time in 0.01sec")
```

```
Warning: Ignoring extra legend entries.
```









Part LTI Component of Control Problem

```
%{once states are near zero and can be linearizable, we can find a
%constant K and L using algebraic riccatti by calling on icare

%compute state matrix when linearized about [0 0 0 0 0 0]

E44 = s.m1 + s.m2 + s.mc; E45 = -s.m1*s.L1; E46 = -s.m2*s.L2;
E54 = -s.m1*s.L1 ; E55 = s.I1 + s.m1*s.L1^2;
E64 = -s.m2*s.L2 ; E66 = s.I2 + s.m2*s.L2^2;
```

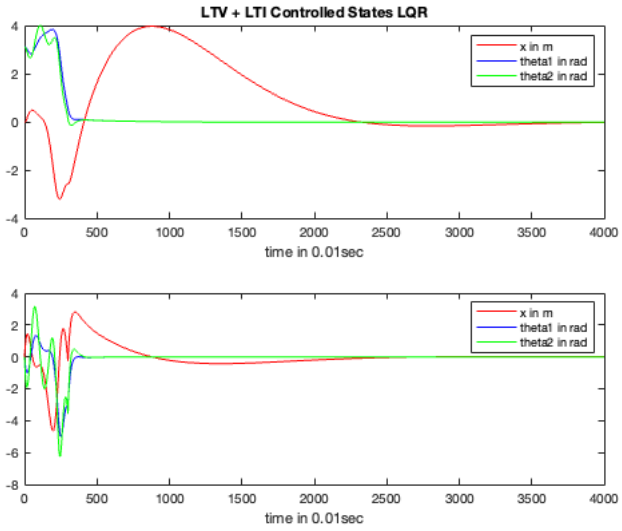
```
E_bar = [[1 0 0  0   0   0];...
     [0 1 0  0   0   0];
     [0 0 1  0   0   0];
     [0 0 0 E44 E45 E46];
     [0 0 0 E54 E55  0 ];
     [0 0 0 E64  0  E66]];

A52 = s.m1*g*s.L1;
A63 = s.m2*g*s.L2;
A_bar = [[0  0  0  1   0   0];...
         [0  0  0  0   1   0];
         [0  0  0  0   0   1];
         [0  0  0  0   0   0];
         [0 A52 0 0   0   0];
         [0  0  A63  0   0   0]];

B_bar = [0 0 0  1  0  0]';


%Compute classical A, B, and C states matrix

A = E_bar^-1 * A_bar;
B = E_bar^-1 * B_bar;
C = [[1 0 0  0   0   0];...
     [0 1 0  0   0   0];...
     [0 0 1  0   0   0];...
     [0 0 0  0   0   0];...
     [0 0 0  0   0   0];...
     [0 0 0  0   0   0]];...


D = [0 0 0 0 0 0]';

E = eye(6);
```

Solve for LTI Controller Gain K and Observer Gain L

```
% Solve algebraic raccatti K
Q = diag([1.2 1 1 1 80 80]);
R = 3;
[X,K_inf, nouseeig] = icare(A,B,Q,R);
%note, the K here is negative of the one from the notaion of NR22


% Solve for raccatti L using iCARE (use a-bk)
Q1 = eye(6);
Q2 = eye(6);
[P, L_inf, ect] = icare(A', C', Q1, Q2) ;


%note, the L here is negative of the one from the notaion of NR22
```

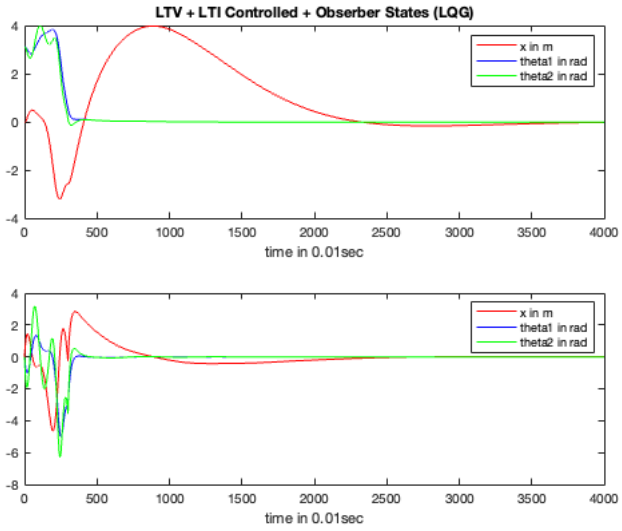
March States Starting from time T on the Estimated Simulated LQR States using RK4

```
T_inf = 40;
for i = 301:1:T_inf/h

    f1 = (A-B*K_inf)*x_sim(:,i);
    f2 = (A-B*K_inf)*(x_sim(:,i) + f1*h/2) ;
    f3 = (A-B*K_inf)*(x_sim(:,i) + f2*h/2) ;
    f4 = (A-B*K_inf)*(x_sim(:,i) + f3*h) ;
    x_sim(:,i+1) = x_sim(:,i) + h*(f1/6 + f2/3 + f3/3 + f4/6);

end

t_inf = 0:T_inf/h;
figure(4); subplot(2,1,1); plot(t_inf,x_sim(1,:),'r',t_inf,x_sim(2,:),'b',t_inf,x_sim(3,:),'g'); title("LTV + LTI Controlled States LQR")
               legend("x in m", "theta1 in rad", "theta2 in rad"); xlabel("time in 0.01sec")
           subplot(2,1,2); plot(t_inf,x_sim(4,:),'r',t_inf,x_sim(5,:),'b',t_inf,x_sim(6,:),'g');
               legend("x in m", "theta1 in rad", "theta2 in rad");xlabel("time in 0.01sec")
```



March States Starting from time T on the Estimated LQG States using RK4

```
for i = 301:1:T_inf/h

    f1 = (A-L_inf'*C)*x_estimation_sim(:,i) - B*K_inf*x_estimation_sim(:,i) + L_inf'*C*x_sim(:,i);
    f2 = (A-L_inf'*C)*x_estimation_sim(:,i) - B*K_inf*x_estimation_sim(:,i) + L_inf'*C*x_sim(:,i);
    f3 = (A-L_inf'*C)*x_estimation_sim(:,i) - B*K_inf*x_estimation_sim(:,i) + L_inf'*C*x_sim(:,i);
    f4 = (A-L_inf'*C)*x_estimation_sim(:,i) - B*K_inf*x_estimation_sim(:,i) + L_inf'*C*x_sim(:,i);

    x_estimation_sim(:,i+1) = x_estimation_sim(:,i) + h*(f1/6 + f2/3 + f3/3 + f4/6);

end

figure(5); subplot(2,1,1); plot(t_inf,x_estimation_sim(1,:),'r',t_inf,x_estimation_sim(2,:),'b',t_inf,x_estimation_sim(3,:),'g'); title("LTV + LTI Con
                    legend("x in m", "theta1 in rad", "theta2 in rad"); xlabel("time in 0.01sec");
                    subplot(2,1,2); plot(t_inf,x_estimation_sim(4,:),'r',t_inf,x_estimation_sim(5,:),'b',t_inf,x_estimation_sim(6,:),'g');
                    legend("x in m", "theta1 in rad", "theta2 in rad"); xlabel("time in 0.01sec");
```



Published with MATLAB® R2022b