North Carolina A&T State University

Final project For ECEN-865 Optimal Control Theory

Development of LQ Tracker for Inverted Pendulum

Submitted To:
Dr. Ali Karimoddini
Asst. Professor
ECE Department

Submitted By :
Mrinmoy Sarkar
950363260
PhD
msarkar@aggies.ncat.edu
Spring-2018

1 Abstract

In this project a linear quadratic(LQ) tracker is developed for a linear time invariant(LTI) system. The selected LTI system is an inverted pendulum which have four states. The vertical angle of the pendulum is controlled using the LQ tracker. The LQ tracker is developed in MATLAB programming environment and simulated with different reference signal.The controller gain is adjusted in such a way that the overall absolute error between the reference signal and the output signal is less than 0.1. In LQ tracker design the trade off between the input energy and tracking error is shown.

Contents

2 Introduction

The linear quadratic tracker is a very efficient tracker in the context of gain margin and phase margin. The LQ tracker has infinite gain margin and at least 60° phase margin. So, the tracker is quite robust for stability of the system. In this project a LQ tracker is developed for an inverted pendulum. In the design step, first the mathematical model of the system is developed then the state space representation is formulated with the developed model. In next step, a cost function is defined for the tracking problem and then using the calculus of variation the cost function is minimized. After this formulation, we get two differential riccati equation(DRE) which is solved using Runge–Kutta method. At last the system is simulated with two different reference signal(step and sinusoidal). The following sections are organized as section 3: System model of inverted pendulum and State space representation, section 4: LQ tracker formulation, section 5: Simulation, section 6: Conclusion.

3 System model of inverted pendulum and State space representation [1]

The cart with an inverted pendulum, shown in figure 1, is "bumped" with an impulse force, F. Determine the dynamic equations of motion for the system, and linearize about the pendulum's angle, $\theta = \pi$ (in other words, assume that pendulum does not move more than a few degrees away from the vertical, chosen to be at an angle of π).

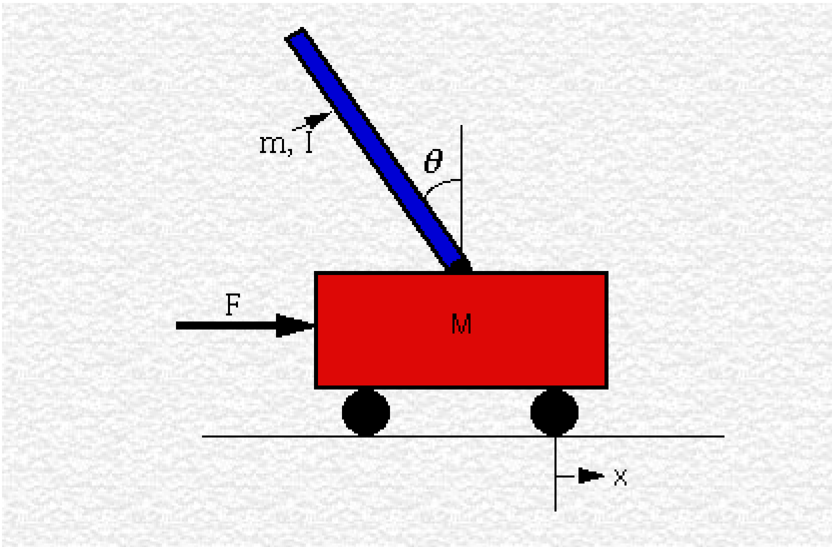

Figure 1: An inverted pendulum

here,
M = mass of the cart
m = mass of the pendulum
b = friction of the cart
I = inertia of the pendulum
L = length of the pendulum's center of mass
F = impulse force applied to cart

3.1 Force analysis and system equations

In figure 2 the free body diagram of the system is shown.

Summing the forces in the Free Body Diagram of the cart in the horizontal direction,

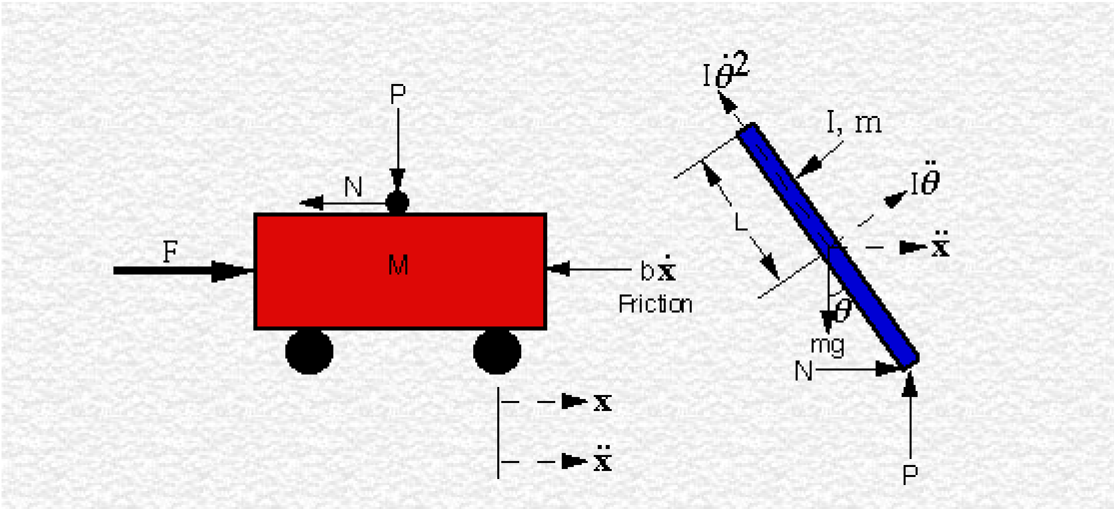

Figure 2: Free body diagrams of the system

we get the following equation of motion:

$$M\ddot{X} + b\dot{X} + N = F \tag{1}$$

Summing the forces in the Free Body Diagram of the pendulum in the horizontal direction, we get an equation for N:

$$N = m\ddot{X} + mL\ddot{\theta}cos\theta - mL\dot{\theta}^2 sin\theta \tag{2}$$

If we substitute this equation into the first equation, we get the first equation of motion for this system:

$$(M + m)\ddot{X} + b\dot{x} + mL\ddot{\theta}cos\theta - mL\dot{\theta}^2 sin\theta = F \tag{3}$$

To get the second equation of motion, sum the forces perpendicular to the pendulum. Solving the system along this axis ends up saving you a lot of algebra. You should get the following equation:

$$Psin\theta + Ncos\theta - mgsin\theta = mL\ddot{\theta} + m\ddot{X}cos\theta \tag{4}$$

To get rid of the P and N terms in the equation above, sum the moments around the centroid of the pendulum to get the following equation:

$$-PIsin\theta - NIcos\theta = I\ddot{\theta} \tag{5}$$

Combining these last two equations, you get the second dynamic equation:

$$(I + mL^2)\ddot{\theta} + mgLsin\theta = -mL\ddot{X}cos\theta \tag{6}$$

This set of equations should be linearized about $\theta = \pi$. Assume that $\theta = \pi + \phi$ (ϕ represents a small angle from the vertical upward direction). Therefore, $cos(\theta) = -1, sin(\theta) = -\phi,, \ddot{\theta}^2 = 0$ and $F = u$. After linearization the two equations of motion become:

$$(I + mL^2)\ddot{\phi} - mgL\phi = ml\ddot{X} \tag{7}$$

$$(M + m)\ddot{X} + b\dot{X} - mL\ddot{\phi} = u \tag{8}$$

3.2 State space representation of the system

From equation 7 and 8, we get the state space representation of the system as below:

$$\begin{bmatrix} \dot{X} \\ \ddot{X} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-(I+ML^2)b}{I(M+m)+MmL^2} & \frac{m^2gL^2}{I(M+m)+MmL^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-mLb}{I(M+m)+MmL^2} & \frac{mgL(M+m)}{I(M+m)+MmL^2} & 0 \end{bmatrix} \begin{bmatrix} X \\ \dot{X} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{I+mL^2}{I(M+m)+MmL^2} \\ 0 \\ \frac{mL}{I(M+m)+MmL^2} \end{bmatrix} u \tag{9}$$

$$y = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ \dot{X} \\ \phi \\ \dot{\phi} \end{bmatrix} \tag{10}$$

4 LQ tracker formulation

If we have a LTI system like equation 11 and 12 and if we define the cost function as equation 13, then the control input of the system would be like equation 14. But to get this solution we neet to solve the two differential riccati equation given in equations 15 and 16 [3]. The block diagram of the system is shown in figure 3. The two DREs are solved simultaneously using Runge–Kutta method [2]. The formula needed to use Runge–Kutta methods is given in equation 17.

$$\dot{X} = AX + Bu \tag{11}$$

$$y = CX \tag{12}$$

Cost function:

$$J = \frac{1}{2}(y(t_f) - r(t_f)^T P(t_f)(y(t_f) - r(t_f) + \frac{1}{2}\int_{t_0}^{t_f} [(y-r)^T Q(y-r) + u^T Ru]dt \tag{13}$$

here, $P \geq 0, Q \geq 0, R > 0$ and all are symmetric.

Solution:

$$u = -kX + R^{-1}B^T\nu \tag{14}$$

here, $k(t) = R^{-1}B^T S(t)$

DREs:

$$-\dot{S} = A^T S + SA - SBR^{-1}B^T S + C^T QC; S(t_f) = C^T P(t_f)C \tag{15}$$

$$-\dot{\nu} = (A - Bk)^T\nu + C^T Qr; \nu(t_f) = C^T P(t_f)r(t_f) \tag{16}$$

here, $r(t)$ is the reference signal and given over time range $[t_0, t_f]$.

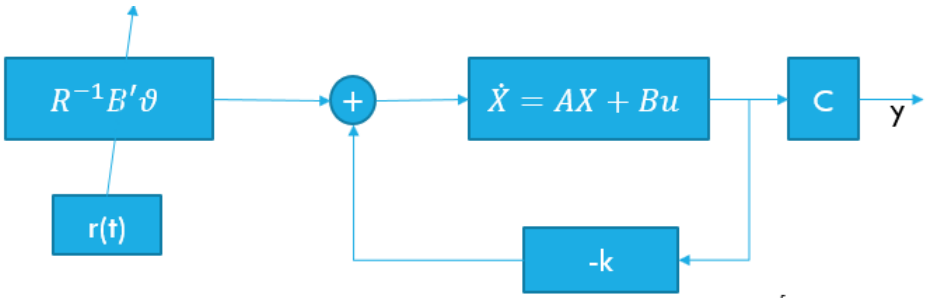

Figure 3: Block diagram of the system with LQ tracker

Equations for Runge–Kutta method:

$$\begin{aligned}
&\dot{y} = f(t, y), \\
&y(t_0) = y_0, \\
&y_{n+1} = y_n + \frac{dt}{6}(k_1 + 2k_2 + 2k_3 + k_4), \\
&t_{n+1} = t_n + dt, \\
&k_1 = f(t_n, y_n), \\
&k_2 = f(t_n + \frac{dt}{2}, y_n + dt\frac{k_1}{2}), \\
&k_3 = f(t_n + \frac{dt}{2}, y_n + dt\frac{k_2}{2}), \\
&k_4 = f(t_n + dt, y_n + dtk_3).
\end{aligned} \tag{17}$$

5 Simulation

Matlab programming language is used for the simulation of the system. The parameters of the simulation is given in table 1. Reference tracking signals are shown in figure 4 and 7. Corresponding output of the system are shown in figure 5 and 8 and the absolute error in tracking are shown in figure 6 and 9 respectively.

Table 1: Simulation parameters

Name	Value
M	0.5 kg
m	0.2 kg
b	0.1
I	0.006 $kg.m^2$
g	9.8 m/s^2
L	0.3 m

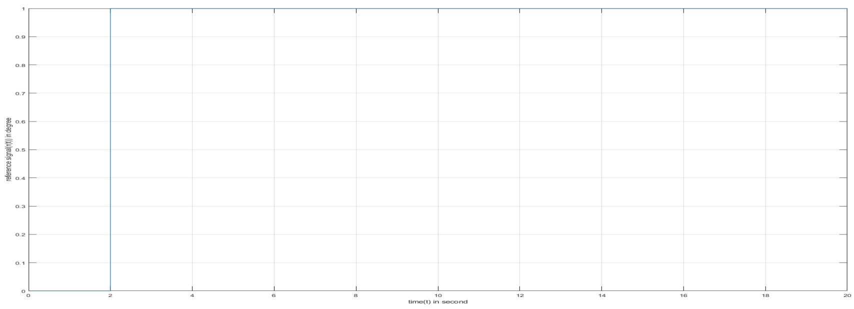

Figure 4: Step reference signal

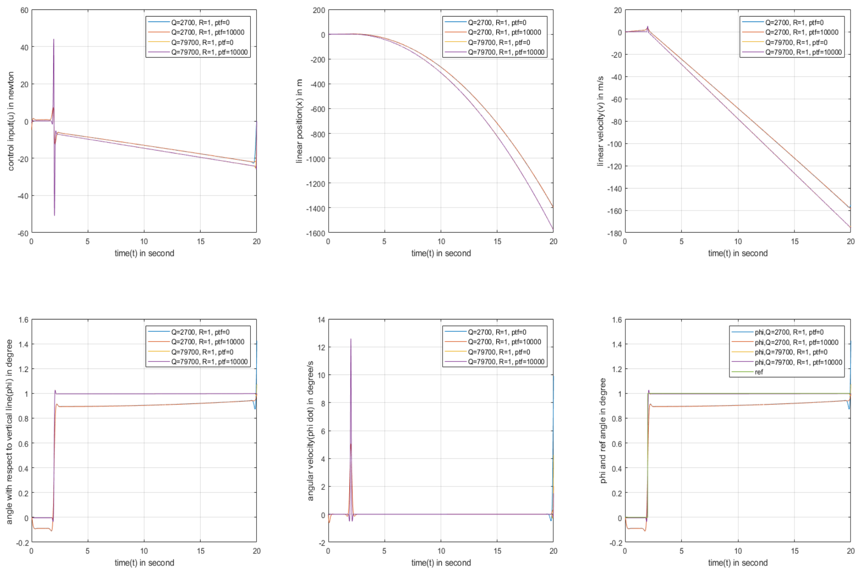

Figure 5: Output of the system for step reference signal

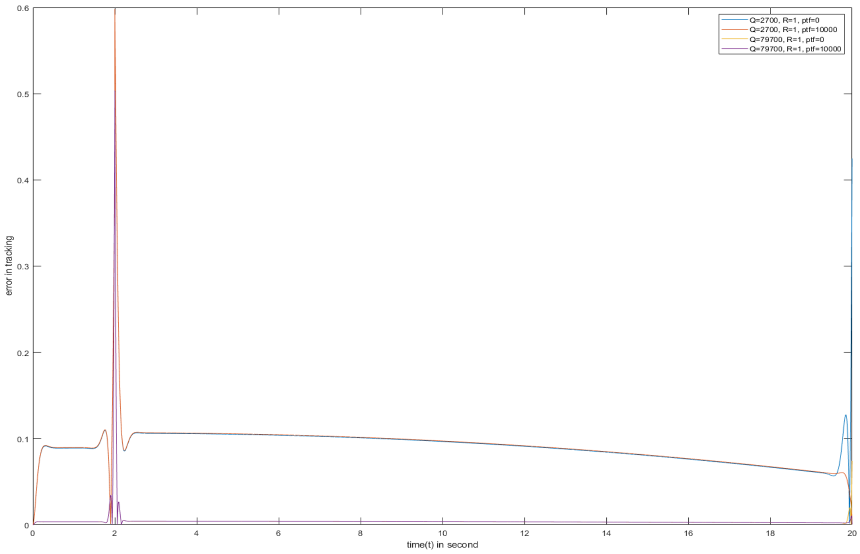

Figure 6: Absolute error in tracking for step reference signal

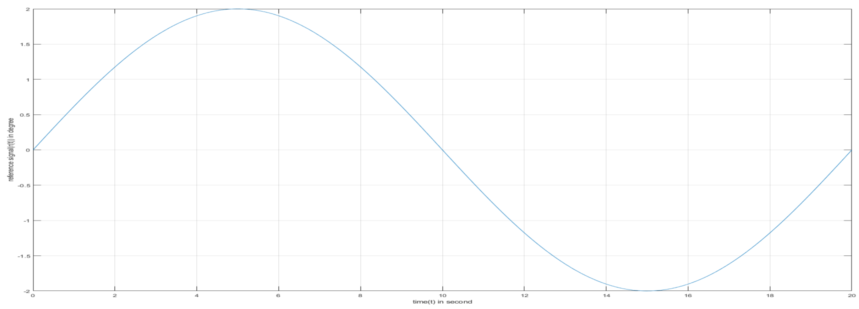

Figure 7: Sine reference signal

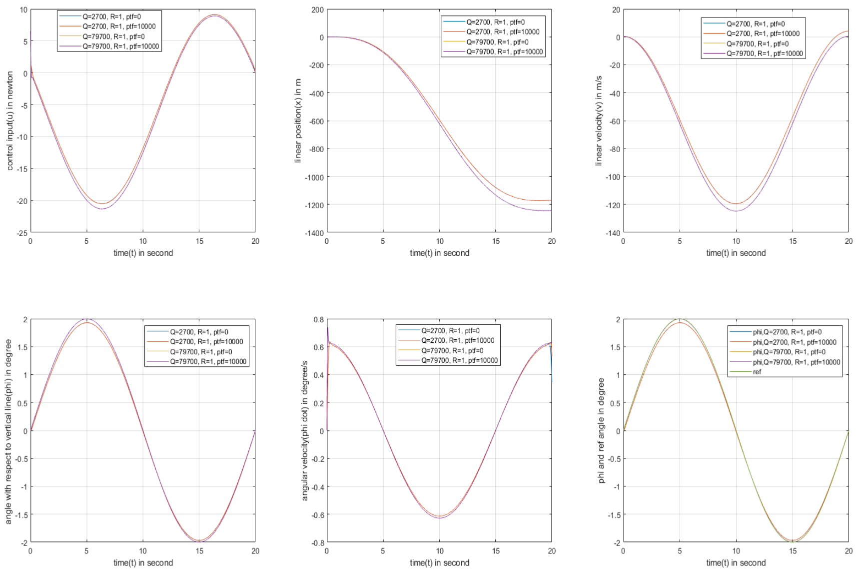

Figure 8: Output of the system for sine reference signal

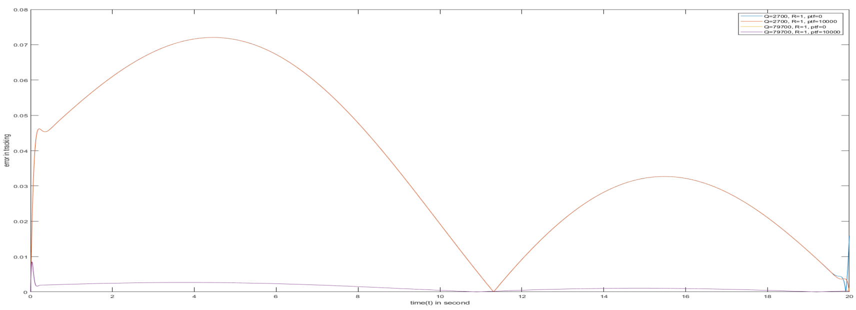

Figure 9: Absolute error in tracking for sine reference signal

6 Conclusion

1. In this project LQ tracker is developed and simulated with two reference signal.
2. The two DRE is solved simultaneously using Runge–Kutta method.
3. The control input is kept in reasonable value.
4. The over all absolute error of the tracker is less than 0.1

A Matlab Code

```
%% author: mrinmoy sarkar
% email: msarkar@aggies.ncat.edu

clear all;
close all;

t0 = 0;
tf = 20;
dt = 0.001;
t = t0:dt:tf;

no_of_state = 4;
no_of_input = 1;



M = .5;
m = 0.2;
b = 0.1;
i = 0.006;
g = 9.8;
l = 0.3;

p = i*(M+m)+M*m*l^2; %denominator for the A and B matricies
A = [0      1                0              0;
     0  -(i+m*l^2)*b/p   (m^2*g*l^2)/p     0;
     0      0                0              1;
```

```
      0  -(m*l*b)/p          m*g*l*(M+m)/p    0];
B = [       0;
      (i+m*l^2)/p;
      0;
     m*l/p];
C = [0  0  1  0];
x0 = [0  0  0  0]';

QQ = [2700,79700];
pptf = [0, 10000];

ref1 = zeros(length(t),1);
refdummy = 2*sin(2*pi*t/20);
% for sinusoidal input
ref1(:,1) = refdummy;

% for step input
%ref1(:,1) = 1;
%ref1(1:2000,1)=0;


for qq=1:2
    for pp=1:2

        Q = QQ(qq);%79700;
        R = 1;
        ptf = pptf(pp);% 10000;




        ref1 = flipud(ref1);
        ref = (ref1(1,:))';



        S = zeros(no_of_state,no_of_state);
        nu = zeros(no_of_state,1);


```

```
        all_nu = zeros(length(t),size(nu,1)*size(nu,2));
        all_nu(1,:) = C'*ptf*ref;

        all_s = zeros(length(t),size(S,1)*size(S,2));
        stf = C'*ptf*C;
        all_s(1,:) = stf(:);
        all_k = zeros(length(t),no_of_state*no_of_input);
       K = (R^(-1))*B'*stf;
        all_k(1,:) = K(:);

        for i=2:length(t)
            S0 = reshape(all_s(i-1,:),size(S));
            S = S0;
            S_dot = A'*S + S*A - S*B*(R^(-1))*B'*S + C'*Q*C;
            k1 = dt*S_dot;

            S = S0 + k1./2;
            S_dot = A'*S + S*A - S*B*(R^(-1))*B'*S + C'*Q*C;
            k2 = dt*S_dot;

            S = S0 + k2./2;
            S_dot = A'*S + S*A - S*B*(R^(-1))*B'*S + C'*Q*C;
            k3 = dt*S_dot;

            S = S0 + k3;
            S_dot = A'*S + S*A - S*B*(R^(-1))*B'*S + C'*Q*C;
            k4 = dt*S_dot;

            S = S0 + k1./6 + k2./3 + k3./3 + k4./6;

             all_s(i,:) = S(:);

            K = (R^(-1))*B'*S;
             all_k(i,:) = K(:);

            nu0 = reshape(all_nu(i-1,:),size(nu));
            nu = nu0;
            ref = (ref1(i,:))';
            nu_dot = (A-B*K)'*nu + C'*Q*ref;
```

```
            k1 = dt*nu_dot;

            nu = nu0 + k1./2;
            nu_dot = (A-B*K)'*nu + C'*Q*ref;
            k2 = dt*nu_dot;

            nu = nu0 + k2./2;
            nu_dot = (A-B*K)'*nu + C'*Q*ref;
            k3 = dt*nu_dot;

            nu = nu0 + k3;
            nu_dot = (A-B*K)'*nu + C'*Q*ref;
            k4 = dt*nu_dot;

            nu = nu0 + k1./6 + k2./3 + k3./3 + k4./6;

            all_nu(i,:) = nu(:);
        end

        all_k = flipud(all_k);
        all_nu = flipud(all_nu);
        all_s = flipud(all_s);
        ref1 = flipud(ref1);

        u = zeros(length(t), no_of_input);
        x = zeros(length(t), no_of_state);
        x(1,:) = x0';
        F = 1;
        u(1,:) = -(reshape(all_k(1,:),size(K)))*(x(1,:))' +
           F*(R^(-1))*B'*reshape(all_nu(1,:),size(nu));
        for i=2:length(t)

            xx0 = (reshape(x(i-1,:),size(x0)));
            xx = xx0;
            xx_dot = A*xx + B*(u(i-1,:)');
            k1 = dt*xx_dot;

            xx = xx0 + k1./2;
            xx_dot = A*xx + B*(u(i-1,:)');
            k2 = dt*xx_dot;
```

```
            xx = xx0 + k2./2;
            xx_dot = A*xx + B*(u(i-1,:)');
            k3 = dt*xx_dot;

            xx = xx0 + k3;
            xx_dot = A*xx + B*(u(i-1,:)');
            k4 = dt*xx_dot;

            xx = xx0 + k1./6 + k2./3 + k3./3 + k4./6;

            x(i,:) = xx(:);
            %F = pinv(C*pinv(-A+B*(reshape(all_k(i,:),size(K))))*B);
            u(i,:) = -(reshape(all_k(i,:),size(K)))*(x(i,:))' + F*(R^(-1))*B'*reshape(all_nu(i,:),size(nu));
        end
        figure(1)
        plot(t,ref1)
        xlabel('time(t) in second')
        ylabel('reference signal(r(t)) in degree')
        grid on
        figure(2)
        %clf
        subplot(231)
        plot(t,u)
        hold on
        xlabel('time(t) in second')
        ylabel('control input(u) in newton')
        grid on
        subplot(232)
        plot(t,x(:,1))
        hold on
        xlabel('time(t) in second')
        ylabel('linear position(x) in m')
        grid on
        subplot(233)
        plot(t,x(:,2))
        hold on
```

```
            xlabel('time(t) in second')
            ylabel('linear velocity(v) in m/s')
            grid on
            subplot(234)
            plot(t,x(:,3))
            hold on
            xlabel('time(t) in second')
            ylabel('angle with respect to vertical line(phi) in
               degree ')
            grid on
            subplot(235)
            plot(t,x(:,4))
            hold on
            xlabel('time(t) in second')
            ylabel('angular velocity(phi dot) in degree/s')
            grid on
            subplot(236)
            plot(t,x(:,3))
            xlabel('time(t) in second')
            ylabel('phi and ref angle in degree')
            hold on
           %plot(t,ref1)
           %hold on
           %legend('phi','ref')
            grid on
            figure(3)
            plot(t, abs(x(:,3)'-ref1'))
            xlabel('time(t) in second')
            ylabel('error in tracking')
            hold on
        end
end

figure(2)
subplot(231)
legend('Q=2700, R=1, ptf=0', 'Q=2700, R=1, ptf=10000', 'Q
   =79700, R=1, ptf=0', 'Q=79700, R=1, ptf=10000')

subplot(232)
legend('Q=2700, R=1, ptf=0', 'Q=2700, R=1, ptf=10000', 'Q
```

```
=79700, R=1, ptf=0', 'Q=79700, R=1, ptf=10000')
subplot(233)
legend('Q=2700, R=1, ptf=0', 'Q=2700, R=1, ptf=10000', 'Q=79700, R=1, ptf=0', 'Q=79700, R=1, ptf=10000')
subplot(234)
legend('Q=2700, R=1, ptf=0', 'Q=2700, R=1, ptf=10000', 'Q=79700, R=1, ptf=0', 'Q=79700, R=1, ptf=10000')
subplot(235)
legend('Q=2700, R=1, ptf=0', 'Q=2700, R=1, ptf=10000', 'Q=79700, R=1, ptf=0', 'Q=79700, R=1, ptf=10000')
subplot(236)
plot(t,ref1)
legend('phi,Q=2700, R=1, ptf=0', 'phi,Q=2700, R=1, ptf=10000', 'phi,Q=79700, R=1, ptf=0', 'phi,Q=79700, R=1, ptf=10000','ref')
figure(3)
legend('Q=2700, R=1, ptf=0', 'Q=2700, R=1, ptf=10000', 'Q=79700, R=1, ptf=0', 'Q=79700, R=1, ptf=10000')
```

References

[1] https://www.ee.usyd.edu.au/tutorials_online/matlab/examples/pend/invpen.html.

[2] https://en.wikipedia.org/wiki/Runge%E2%80%93Kutta_methods.

[3] Ecen-865 lecture notes.