FINALS PROJECT REPORT

SELF BALANCING ROBOT

Abstract

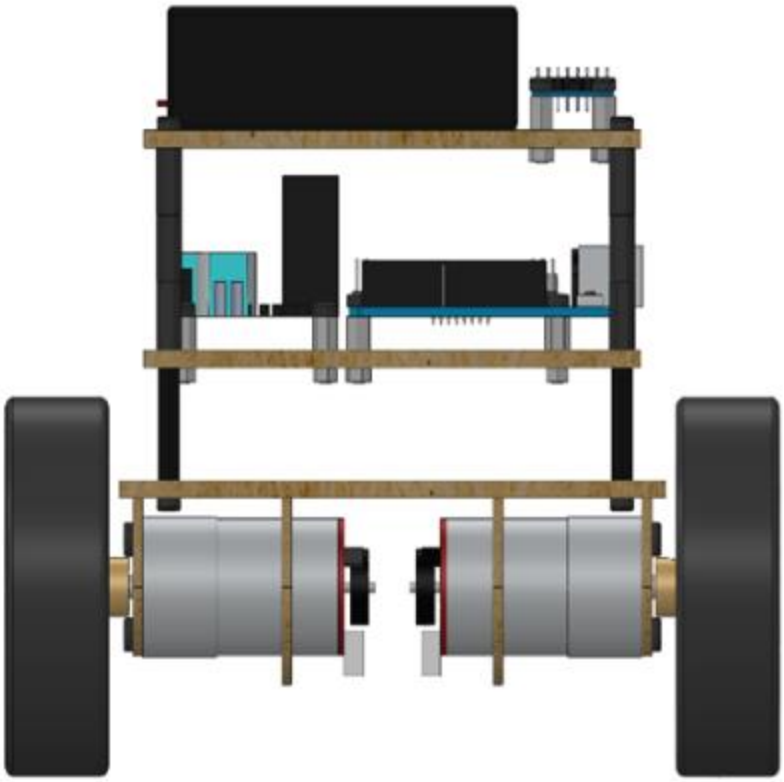
Control System Design Algorithm for Linear
Control System of Self-Balancing Robot

Arpit Savarkar (SID:109561054)
Sagar Eligar (SID:109544149)

Linear Control Systems – MCEN 5228

Course Project – Finals

Self-Balancing Robot



Self – Balancing Robot

Answer Q1:

Objective: The selected system is a self-balancing robot with a set defined parameters. This system in theory behaves similar to an inverted pendulum as self-balancing systems aim to balance in vertical orientation, i.e. there needs to be a constant force to keep system to keep in upright position. All technical details are elaborated in Table 1.

Following are the set of desired behavioral characteristics of the system after the implementation of Linear Control System tools:

- Bounded Input and Bounded Output (BIBO) stable system with a suitable control system.
- Single Input, multi-output (SIMO) system for Real time balancing with smooth and dynamic response.
- To move and maintain desired pose on ground with vertical balanced system.
- Controlled Maneuverability with an optimized K gain matrix.

Control of the system:

- Constant force needs to be applied in the required direction to keep the system balanced.
- This force is applied in the direction of the fall to get an inertial force in the opposite direction and balance the robot in vertical position.
- The dependent parameters of the system decide the controls. Characteristics include stability, controllability, overshoot, observability, rise time and settling time.

Assumptions:

- The system is considered as an inverted pendulum. Masses above and below the center of mass (COM) is considered as single mass namely mass **'M'** and **'m'**.
- The rod connecting the masses is considered to be very light almost weightless.
- The weight is distributed symmetrically in y-axis.
- No air friction and other disturbances.

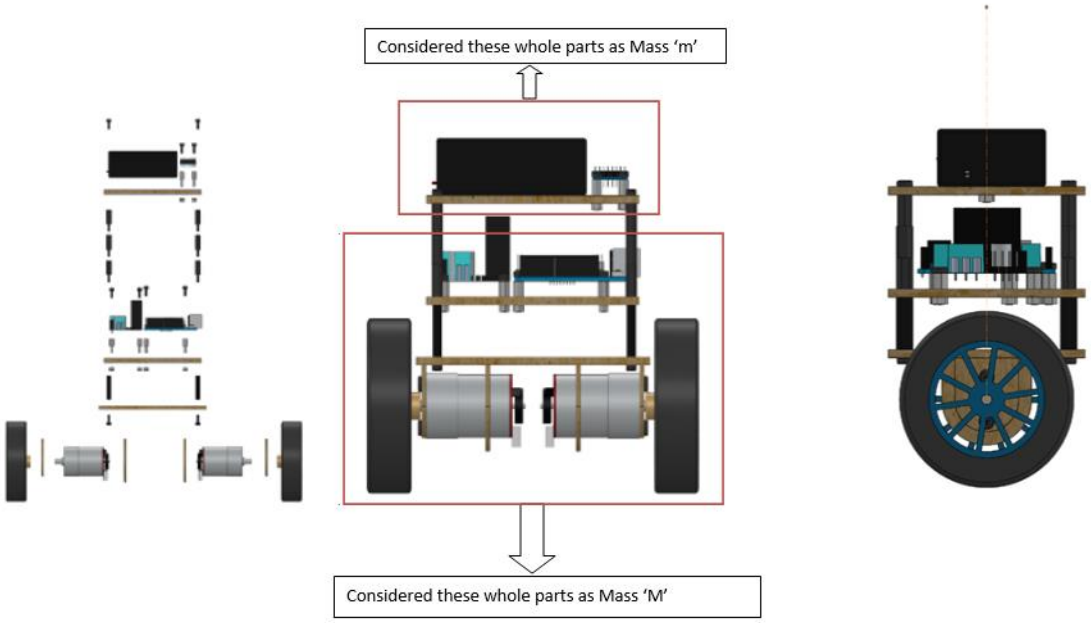
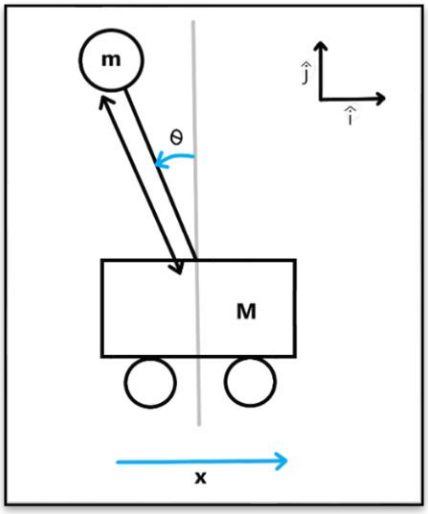


Figure 1 Self-Balancing Robot Model

State variables:

$$\begin{bmatrix} x \\ \dot{x} \\ \Theta \\ \dot{\Theta} \end{bmatrix}$$



Physical meaning of the state variables:

$\boldsymbol{x} = The\ position\ of\ the\ mass\ M\ in\ x\ direction$

$\boldsymbol{\dot{x}} = The\ velocity\ of\ mass\ M\ in\ x\ direction$

$\Phi = \ \ The\ angle\ of\ mass\ m\ from\ vertical$

$\dot{\Phi} = The\ angular\ velocity\ of\ mass\ m$

Other parameters:

***I** = moment of inertia of mass m*

***L**= Distance between the mass 'M' and mass 'm'*

***F**= force applied on mass M to balance mass m at vertical angle = control input of the system('u')*

***b** = coefficient of friction between mass M and ground*

***y**= output of the system. Both Mass M x position and mass m Φ angle along vertical*

State Space Equation:

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

With the help of FBD (Free body diagram) and Newton second law, matrix A, B, C and D are derived.

A =

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -(I+ml^2)b/(Mml_2 + I(M+m)) & m^2gl^2/(Mml_2 + I(M+m) & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -mlb/(Mml_2 + I(M+m) & mgl(M+m)/(Mml_2 + I(M+m) & 0 \end{bmatrix}$$

B =

$$\begin{bmatrix} 0 \\ I + ml^2/(I(M+m) + Mml^2) \\ 0 \\ ml/(I(M+m) + Mml^2) \end{bmatrix}$$

C=

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

D=

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Numerical Details of system parameters:

Table 1: System Technical Data

Parameters	Data
M	**1.175 Kg**
m	**0.2 Kg**
b	**0.1 Kg**
I	$\frac{ml^2}{3} = .0004\ Kg.m^2$
g	**9.8** m/s^2
L	**.08m**

```
A =
```

Plant under Inspection

```
         0    1.0000         0         0
         0   -0.0818    1.2214         0
         0         0         0    1.0000
         0   -0.7790  104.9659         0
```

```
>> B

B =
```

Actuators under Work

```
         0
    0.8179
         0
    7.7897
```

STATE SPACE

MODEL VARIABLES

```
>> C

C =
```

Sensors in Action

```
     1     0     0     0
     0     0     1     0
```

```
>> D

D =

     0
     0
```

Answer Q2:

***Desired Closed Loop Characteristics*:**

- Settling time less than 10 seconds
- Rise time less than 5 seconds.
- The equilibrium angle between 15 degrees (0.28 radians) from the vertical.
- Steady state error less than 1% for all the states of the system.
- A stable system.

Open loop system:

Analyzing the open loop poles (without compensation) of the system shows to determine system stability and dynamic characteristics. Since the system is assumed to be *Linear Time Invariant* (LTI). Eigenvalues of the matrix A represent the open loop poles of the system.

MATLAB command to calculate eigen values:

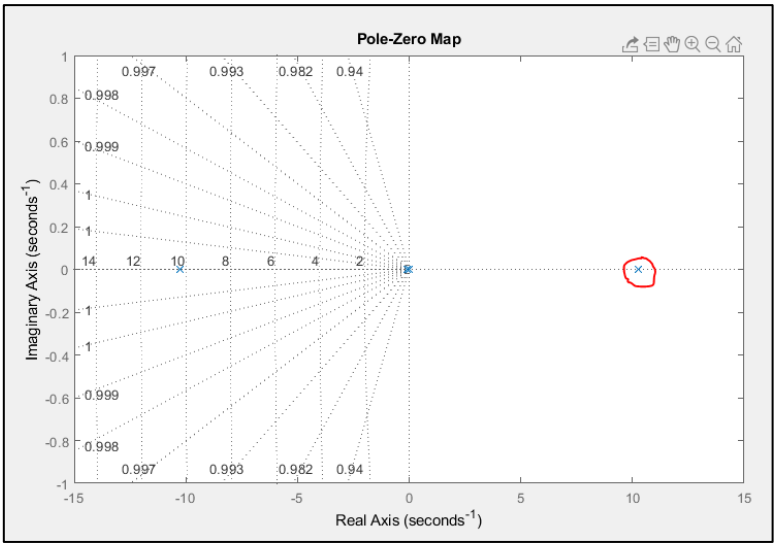
```
>> eigs(A)

ans =

  -10.2499
   10.2408
   -0.0727
         0
```

Since one of the poles is in the Right Half Plane on the Real-Imaginary axis, which indicates the system is unstable in open loop. **As it can be seen from the figure, the system blows up for a step input in open loop**. Therefore, we design a controller to meet our desired requirements.

Figure 1: Pole-Zero Map for Open loop system



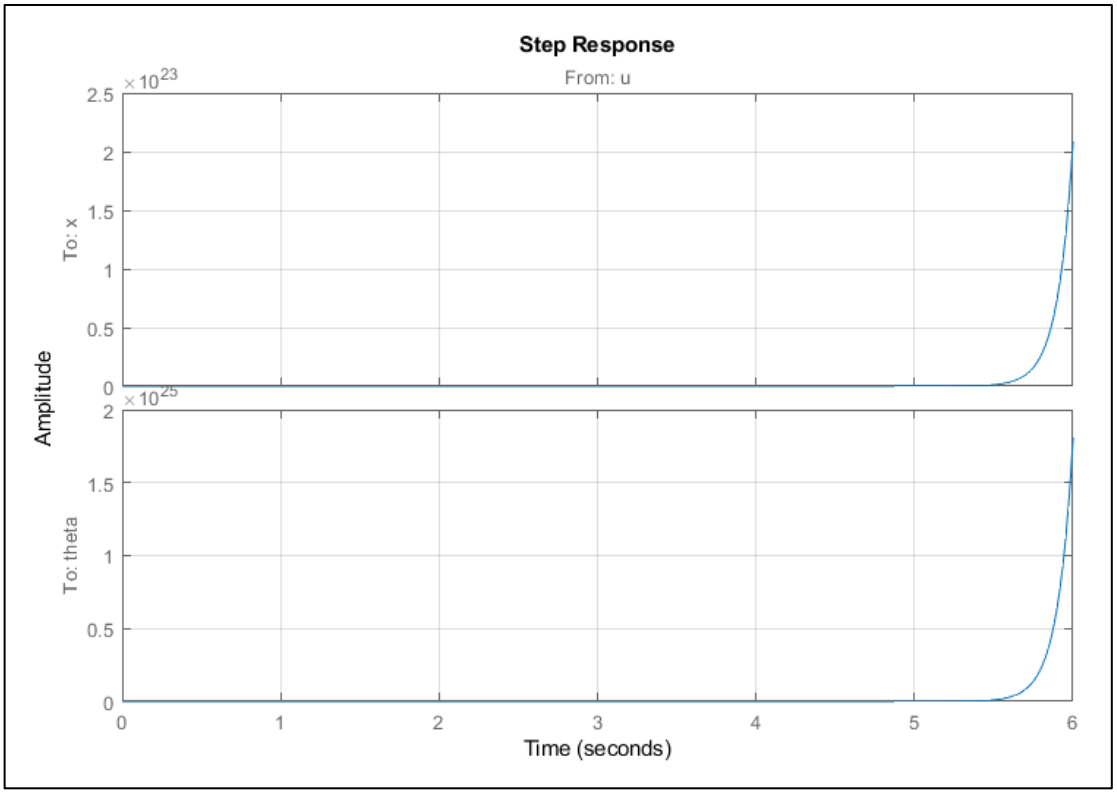


Figure 2: Step Response of the open loop system

Conclusion:

1) **The step response in Figure 2 with initial static conditions shows the immediate instability of the robot for both balance and position tracking with single input of 0.2N step force input which is confirmed with the positive eigenvalue. Since this is a regulator problem architecture. As predicted, the poles of the system are identical at 10.2409 (Figure 1) growing unboundedly, which is also confirmed by the indication the robot's vertical orientation shoot past 180 degrees for a small input. A pole situated at 0 never settle's with increment in time.**
2) **As shown below, MATLAB indicates the instability with respect to the Rise time, Settling time and Overshoot.**

```
>> open_loop = stepinfo(sys_ss);
>> s=open_loop.SettlingTime

s =

   NaN

>> r=open_loop.RiseTime

r =

   NaN

>> o=open_loop.Overshoot

o =

   NaN
```

Answer Q3:

Controllability:

Implementing the concepts of control system to design the controller to enhance performance we check if the system is controllable. This can be known using controllability matrix (€). If the matrix € is a full rank matrix or equal to number of linear independent rows or columns, it means the system is completely controllable.

€ = [A AB A^2B A^3B] {for our system}

MATLAB command for controllability test:

```
>> rank(ctrb(A,B))

ans =

     4
```

Since the rank of € matrix is 4 equals to column rank depicting the system is fully controllable and poles can be placed as per requirement. Thus, the system can be transferred to any desired state in a finite time.

Stabilizability:

Condition for stabilizability is:

Matrix: [(A - **λ***I) B] for all Re{**λ**} ≥ 0

```
>> lambda=10.2408;
>> X=lambda*[1 0 0 0;0 1 0 0;0 0 1 0;0 0 0 1];
>> [A-X B]

ans =

  -10.2408    1.0000         0         0         0
         0  -10.3226    1.2214         0    0.8179
         0         0  -10.2408    1.0000         0
         0   -0.7790  104.9659  -10.2408    7.7897

>> rank(ans)

ans =

     4
```

Self-Balancing robot has 2 negative poles, 1 pole at 0, which is seen in the concatenated matrix with full row rank 4 which indicates that the system is stabilizable for [(A - **λ***I) B] of dimension 4*5

Observability:

```
>> obserability_matrix = obsv(A,C)

obserability_matrix =

    1.0000         0         0         0
         0         0    1.0000         0
         0    1.0000         0         0
         0         0         0    1.0000
         0   -0.0818    1.2214         0
         0   -0.7790  104.9659         0
         0    0.0067   -0.0999    1.2214
         0    0.0637   -0.9514  104.9659

>> observability = rank(obserability_matrix)

observability =

     4
```

A system or equivalently the pair (C,A) is said to be observable if for any t>0 , the initial state can be determined from the time histories of y(t) & u(t) on the interval. If the map is invertible, this implies its null space N(O) is trivial.

For the current system since it is a complete column rank in the concatenated matrix, system is observable for needed input.

Detectability

Detectability is a weaker condition than observability. That is if the system has full column rank of observability matrix is it detectable for all eigenvalues that are equal or greater or equal to zero.

```
>> [A-X;C]

ans =

  -10.2408    1.0000         0         0
         0  -10.3226    1.2214         0
         0         0  -10.2408    1.0000
         0   -0.7790  104.9659  -10.2408
    1.0000         0         0         0
         0         0    1.0000         0

>> rank(ans)

ans =

     4
```

Matrix: [(A-λ*I) C] for all Re{λ} ≥ 0

Since, all the eigenvalues of the matrix have full column rank the, that is **4**, the system is detectable.

Closed loop system

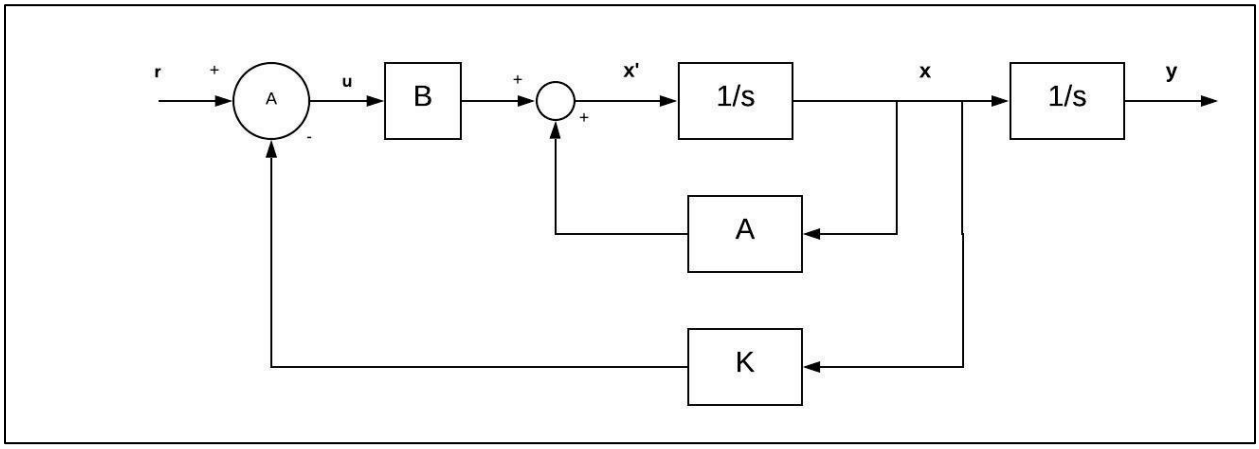


Figure3: State Feedback System in Close Loop

State feedback Control law:

The block diagram of full state feedback is shown above. Poles placement full state feedback is method of placing desired poles (Eigen values) anywhere in the s plane. For this to be possible the system should be controllable. Since our system is controllable, we can use this full sate feedback to make system stable and meet out desired requirements.

Effects of poles location on system response:

- Stability
- Convergence rate
- disturbance rejection
- command following, etc.

From block diagram (Figure 3):

$u = -K * x$

$\dot{x} = (A - B * K) * x$

$y = (C - DK) * x$

With Pre-compensation:

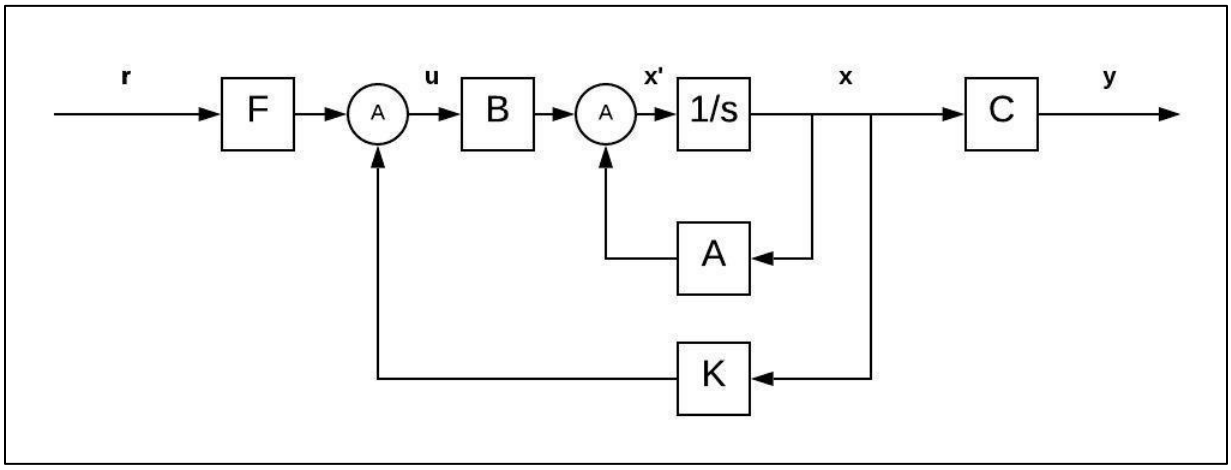


Figure 4: System Architecture with Pre-Compensation

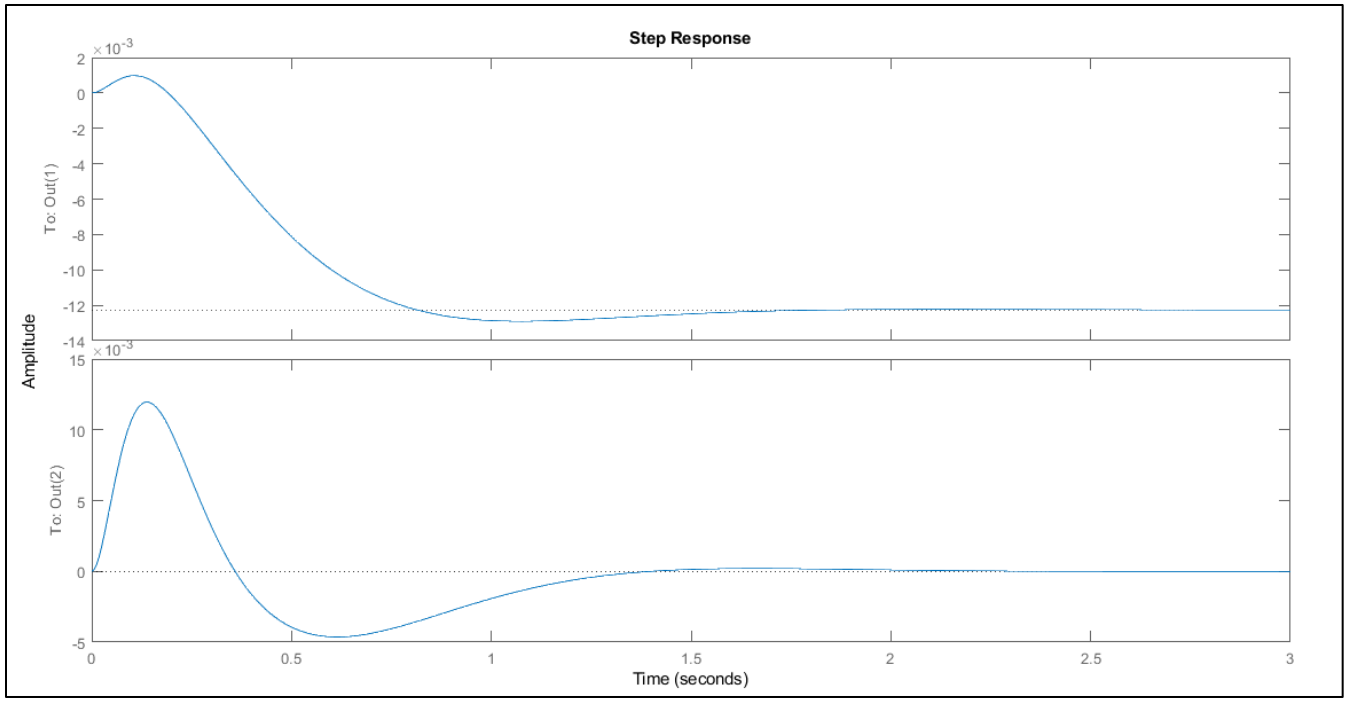


Figure 5 step Response without Precompensation

The simulation from above in Figure 5 indicates the system response incapability with respect to position tracking which steady state error after transients.

Answer Q4:

Pole Placement method:

Using the pole placement methods considering the criteria set according to

- A stable system.
- A settling time of less than 10 seconds
- Rise time less than 5 seconds.
- The equilibrium angle between 15 degrees (0.28 radians) from the vertical.
- Steady state error less than 1% for all the states of the system.

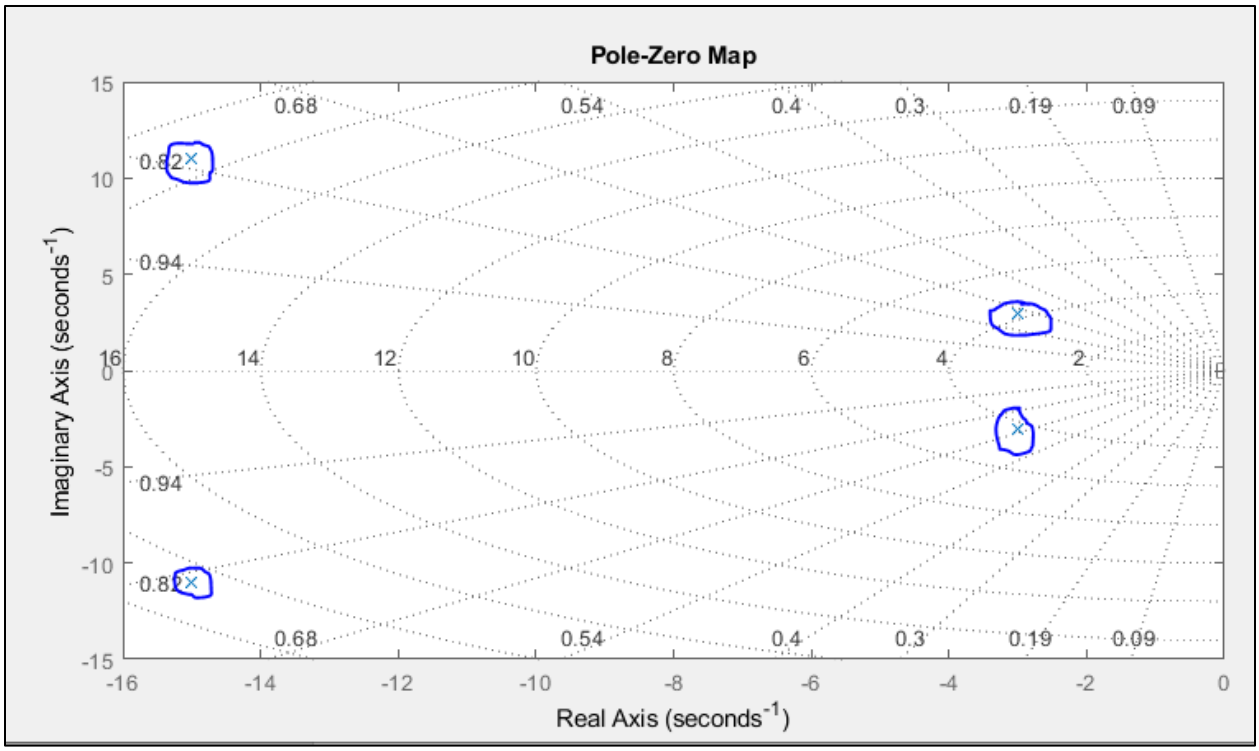


Figure 6 Map of Closed Loop Poles

The Poles in the above diagram indicates the shift into the left half plane which was initially had 1 pole in RHP and 1 pole at the origin. The more the poles towards the left half planes the more the stability of the system.

The Gain Matrix(M) for the closed loop system with controller is:

```
Gain_matrix =

  -81.5836  -34.3683   91.8773    8.2197
```

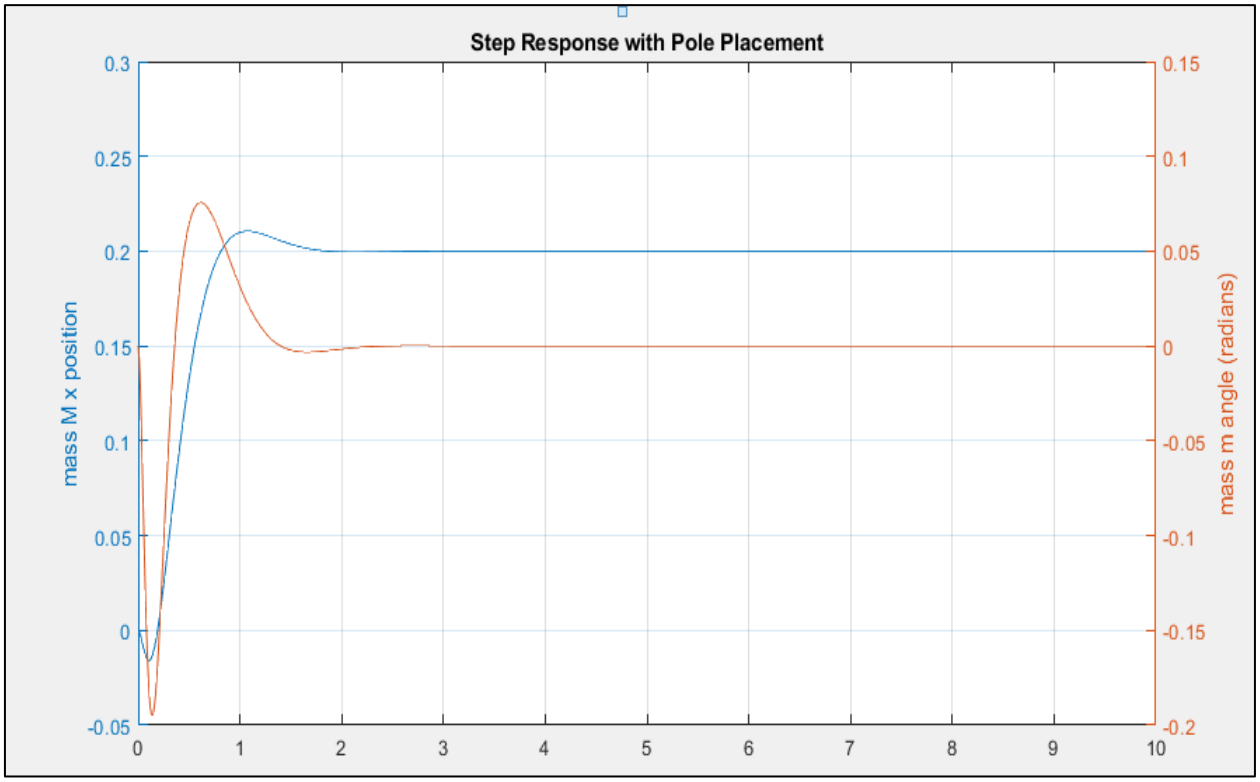


Figure 7 Step Response for Closed Loop with Precompensation

Remarks:

From Figure 5, Figure 6, Figure 7, it can be observed that

- The poles of the system are placed in the LHP due to which the settling time is lower and within limits.
- All poles of closed loop system have imaginary part due to which the oscillatory behavior and overshoot exists in the system.
- A more robust control over the settling time with respect to position tracking is observed.

In process of design, we first set the poles in a method by which to maintain the settling time for position tracking to be within limits, for reference tracking of input position. Accordingly, we get the required poles shown above such that settling time is within 15 seconds with optimized control over overshoot.

We now, move the closed loop poles to target rise time, we add a reference input r and drive along the precompensation gain F (~-80). Toggling with F and rise time requirement to the set desired characteristics of Settling time of less than 10 seconds, Rise time less than 5 seconds, and less than steady state error less than 1%. The Following were derived from the system characteristics.

```
>> S2 = stepinfo(pes,'RiseTimeThreshold',[0.05 0.95]);
>> rt2 = S2.RiseTime

rt2 =

    0.5150

>> rt2 = S2.SettlingTime

rt2 =

    1.4632

>> rt2 = S2.Overshoot

rt2 =

    5.2666
```

Table 2 Closed Loop Parameters

Parameters	Closed Loop Characteristics with Precompensation	Objective
Rise Time	0.515 Sec	5 Sec
Settling Time	1.4632 Sec	10 Sec
Percentage Overshoot	5.266 Sec	-
Steady State Error	0.05 %	1%

Thus, the pole placement is successful as all the objectives are met.

Thus during the design process, the following was established:

- Moving Further into the LHP makes the system settle faster.
- Higher the Force input, higher the overshoot. Approximately a factor of 1.5 times for overshoot.
- Higher Pole makes the system unrealizable for physical robot to achieve.

Answer Q5:

Based on the assumption of full-state feedback, we design full state estimator

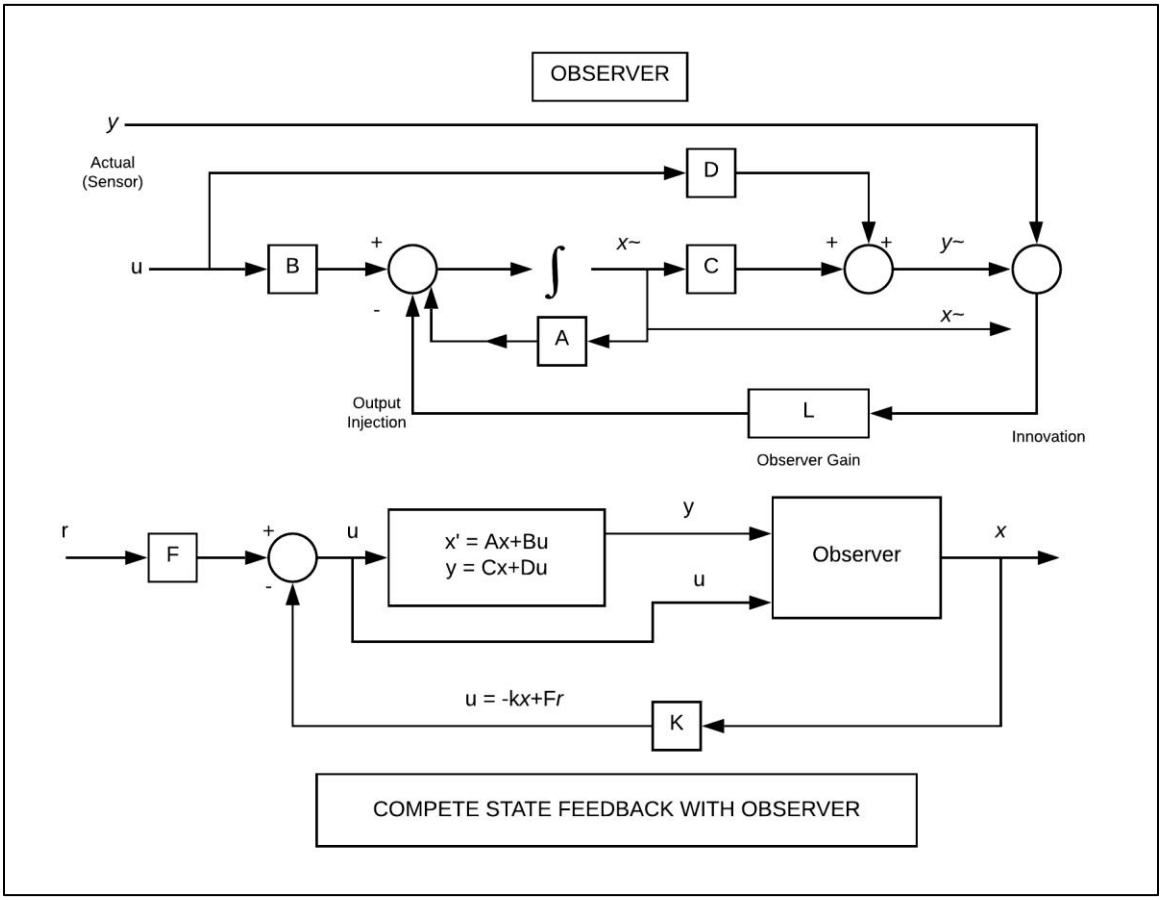


Figure 8 Full State Feedback with Observer

```
>> eig(A-B*M)

ans =

 -15.0000 +11.0000i
 -15.0000 -11.0000i
  -3.0000 + 3.0000i
  -3.0000 - 3.0000i

>> PP = [-30 -32 -34 -36];
>> L = place(A',C',PP)'

L =

   1.0e+03 *

    0.0661   -0.0020
    1.0884   -0.0662
   -0.0027    0.0658
   -0.1150    1.1848
```

The system is **observable**. Since the observability matrix is 8x4 and has rank 4, it is full row rank and thus it is observable. The dynamics of the observer should be much faster than the system itself, we keep it at almost 10 times. The slowest pole is equal to -3, therefore we have kept the observer poles at -30.

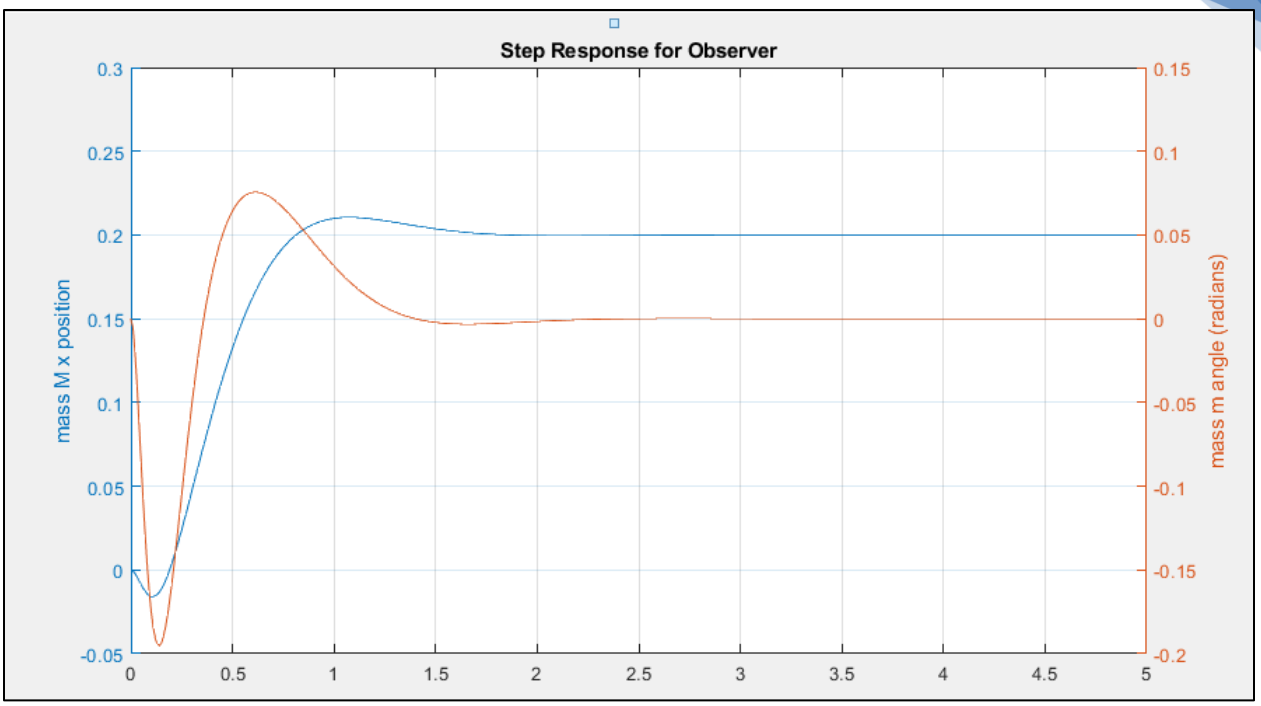


Figure 9 Simulated Response of the observer system

This response is almost identical to the response achieved when it was assumed that we had full access to the state variables. This is because the observer poles are fast, and because the model we assumed for the observer is identical to the model of the actual robot (including the same initial conditions). Therefore, all of the design requirements have been met with the minimal control effort expended.

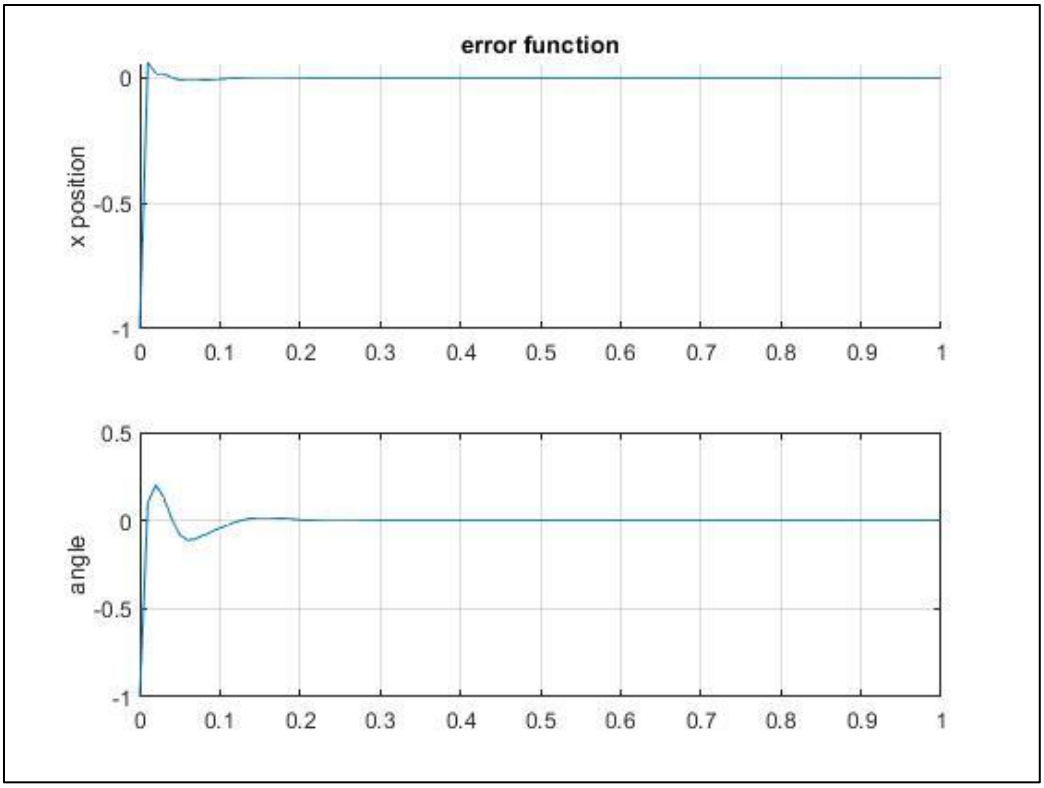


Figure 10: Error in the estimation of the observer system

Conclusion:

- The estimate of the system as seen from Figure 9 almost identical to real value.
- The Error as seen from figure 10, observer follows the robot closely and therefore the errors in the position and angle estimates are very small. The error closes to zero in less than one second. Hence the error rate is sufficient.
- Observer is designed successfully.

Linear Quadratic Regulator (LQR):

To overcome the demerits of the pole placement full state feedback method LQR was developed. It is difficult to know the exact eigen values to meet the desired requirements therefore in LQR with the help of cost function K gain matrix is computed.

LQR design is given by:

$u = -K * (x - x_d)$

$x_d = \ desired\ state\ of\ the\ system$

$\dot{x} = (A - B * K) * x$

$y = (C - DK) * x$

Cost function is given by:

$$\tilde{J} = \frac{1}{2}\int_0^T \left(x^T Q x + u^T R u\right)\, dt + \frac{1}{2}x^T(T) P_1 x(T)$$

Q ≥0, R>0 are symmetric and positive (semi-) definite matrix. These two matrices balance the relative importance of control input(u) and state variables output errors. Here state variables output are mass M x position and mass m vertical angle.

P_1 - is a symmetric, positive (semi-)definite matrix.

As we vary the Q and R matrix K gain matrix varies. The objective is to minimize the cost function to get the best response for our desired requirements and obtain a robust performance. Choosing Q and R matric is an iterative process till you get the best performance. The following three approaches show the difference in the response on varying the Q, R matrices.

Approach 1:

Simplest choice:

R= 1

Q= I_{4x4}

With the help of MATLAB command ***lqr (A, B, Q, R)*** we get the K gain matrix.

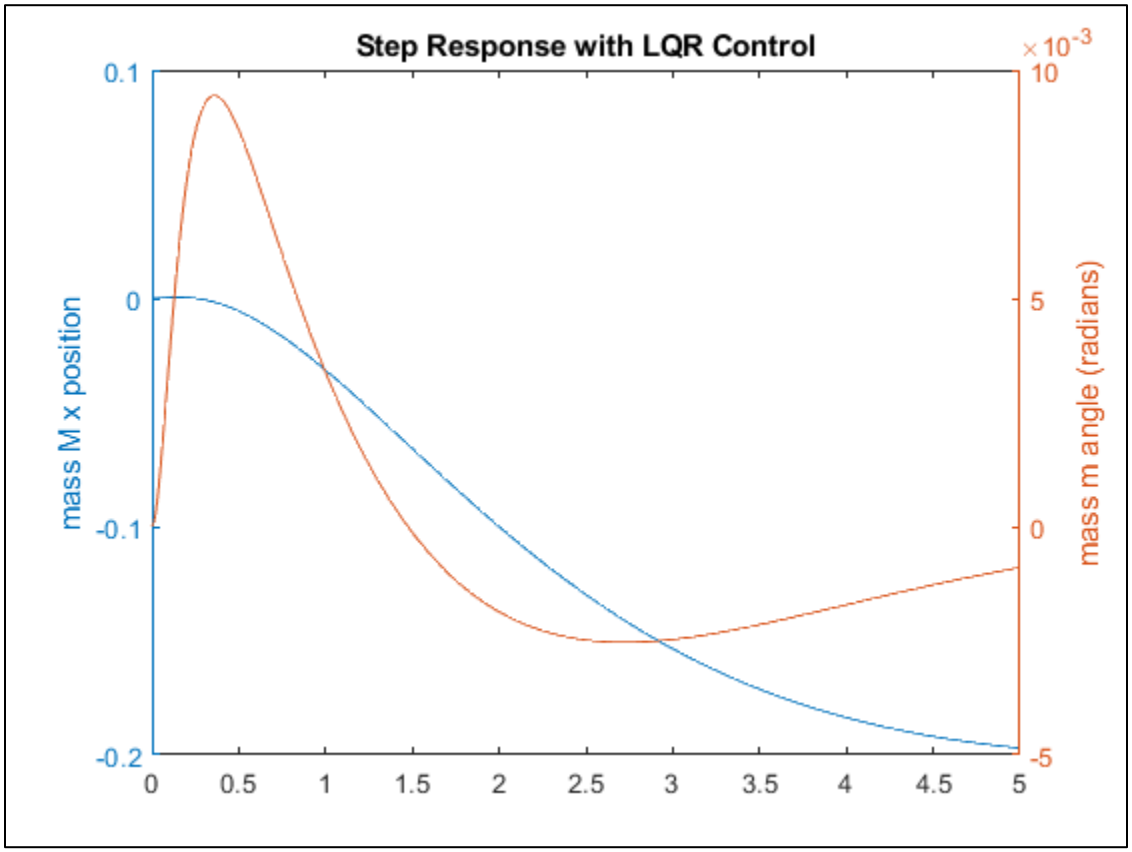


Figure 11 Simulation of LQR control

As we can see from the plot the system is performing well enough and but doesn't meet out required objectives.

Approach 2:

Diagonal weights:

Desired objectives:

- The system to be stable and mass m moves vertical angle within the error of .1 m distance.

- Rise time less than .7 sec.
- Settling time less than 5 sec.
- Mass m never crosses 15degrees (.26 radians)

$$Q = \begin{bmatrix} q_1 & & \\ & \ddots & \\ & & q_n \end{bmatrix} \qquad R = \rho \begin{bmatrix} r_1 & & \\ & \ddots & \\ & & r_n \end{bmatrix}$$

Since the distance can have .1 m error and angle error of .25 radians which implies,

$$q_1 = (\frac{1}{.1})^2 = 100$$

$$q_3 = (\frac{1}{.25})^2 = 16$$

To get the K gain matrix as per our desired requirements we need to tweak the values as a factor of 10 until we reach our objectives. With some iterations following values were obtained to meet our requirements.

R = 1; Q (1,1) =5000; Q (3,3) =100

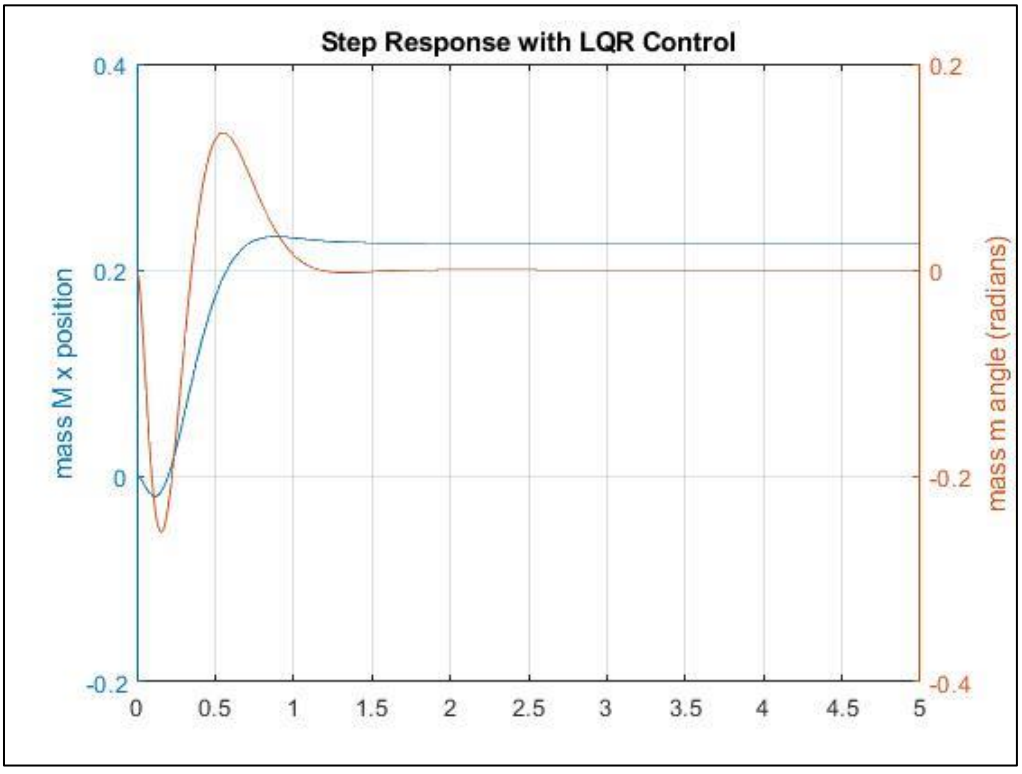


Figure 11 Simulation of LQR control (Diagonal Weights)

Conclusion:

- In all the approach of LQR control **Q >> R,** indicates that the control input is cheap i.e., the cost of input energy is low as the batteries can be recharged and their cycling ability is very high.
- Approach 2 has higher overshoot than approach 1 but it is within the desired limits described by the requirements. It also provides the lowest settling and rise time due to lower penalty on the states of the system as compared to others. Thus, it is the best system.

MATLAB CODE

```
%writing state variables

A = [ 0    1.0000        0        0;
 0   -0.0818    1.2214        0;
 0         0         0    1.0000;
 0   -0.7790   104.9659        0];
B= [0;
0.8179;
0;
7.7897];
C= [ 1     0     0     0;
0     0     1     0];
D= [0;
0];
x0 = [0;0;0;0] % initial condition of the system i.e. for origin
%%x0 = [2;0;0;0]'; % initial state of the system for LQR control
t_final=5;
t = 0:0.01:5; % simulating for 5 seconds
r =0.2*ones(size(t)); % reference step input of 0.2
%open loop system
sys_open_loop = ss(A, B, C, D,'statename',states,'inputname',inputs,'outputname',outputs); %open loop
state space system
figure(1); grid on;
step(sys_open_loop); %Step response of the open loop system
figure(2);grid on;
pzmap(sys_open_loop); % gives the location of poles and zeros in s plane
%closed loop control
p1 = -15 + 11i; % pole placement poles
 p2 = -15 - 11i;
 p3 = -3 + 3i;
 p4 = -3 - 3i;
M = place(A,B,[p1 p2 p3 p4]); % places the poles for closed loop with controller with a gain matrix M
pes = ss(A-B*M, B*-80, C, D); % closed loop system with a reference input
figure(3); grid on; %simulating the closed loop system using pole placement method
lsim(pes,r,t); % simulated step response of closed loop system with controller
%Closed loop system with controller and observer
Ac=A-B*M; %closed loop A matrix
Bc= B*-80; %closed loop B matrix
PP=[ -30; -32; -34; -36]; consolidate observer poles into one, multiplying lowest poles method pole by
factor 10
 L = place(A', C', PP)'; %Placing poles 10 times farther than the closed loop poles
Observer_sys = ss(A-L*C,B*-30,C,D); %Create a new state space system for the observer
figure(6);grid on;
lsim(Observer_sys ,r, t); %simulates closed loop system with controller
```

```
[AX,H1,H2] = plotyy(t,y(:,1),t,y(:,2),'plot'); %naming the plot
set(get(AX(1),'Ylabel'),'String','mass M x position')
set(get(AX(2),'Ylabel'),'String','mass m angle (radians)')
%Error function
closeloop_controller=lsim(pes,r,t);
 closedloop_observer=lsim(sys_est_cl,r,t);
 Xhat = closeloop_controller-closedloop_observer;
 tt = 0:.01:5;
 subplot(2,1,1); grid on;hold on;
 ylabel("x position");
 plot(tt,Xhat(:,2));
subplot(2,1,1); grid on;hold on;
ylabel("x position");
 plot(tt,Xhat(:,1));
 subplot(2,1,2); grid on;
 ylabel("angle error");
plot(tt,Xhat(:,2));
 grid on >> title("Error function") ;
subplot(2,1,1);
 title("Error function") ;
subplot(2,1,2);
 ylabel("angle error");
%LQR control:
%
R = 1;
 Q(1,1)=5000;
Q(3,3)=100;
 Q(2,2)=0;
 Q(4,4)=0;
[K,S,P] = lqr(A,B,Q,R); %calculating the gain matrix using LQR method
Ac = [(A-B*K)]; %closed loop LQR state space matrix
Bc = [B];
 Cc = [C];
Dc = [D];
sys_LQR = ss(Ac,Bc,Cc,Dc,'statename',states,'inputname',inputs,'outputname',outputs); % New state
space system
figure(7);grid on;
lsim(sys_LQR,r,t); %simulating the closed loop control system
```