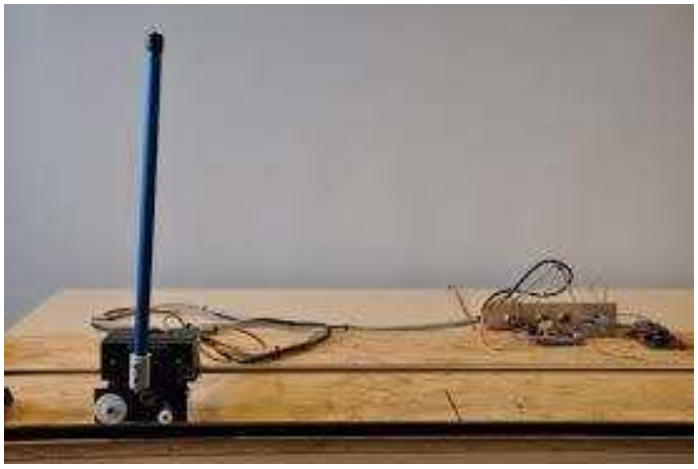
Inverted Pendulum



Researched By:
Mohamed Emad El-Din Anwar

Contents of the Research:

i. Introduction

The Inverted Pendulum (IP) is an inherently unstable system. Force must be properly applied to keep the system intact. To achieve this, proper control theory is required. The Inverted Pendulum is essential in the evaluating and comparing of various control theories.

The reason for selecting the Inverted Pendulum as the system is:
· It is a nonlinear system, which can be treated to be linear, without much error, for quite a wide range of variation.

The different stages of the work for accomplishing the task of controlling the Inverted Pendulum are as follows:
· Modeling the IP and linearizing the model for the operating range.
· Analyzing the uncompensated closed loop response with the help of a root locus.
· Designing the PID controller and simulating it in Mat-Lab for proper tuning and verification.
.Designing the State Space controller and simulating it in Mat-Lab for proper tuning and verification.

Problem Definition:

The system consists of an inverted pendulum mounted to a motorized carriage, the pendulum will simply fall over if the carriage isn't moved to balance it. This Carriage Balanced Inverted Pendulum (CBIP) provides the control force to the carriage by means of a DC servo-motor through a belt drive system. The outputs from the CBIP rig can be carriage position, carriage velocity and pendulum angle, the pendulum angle is fed back to an Analog Controller which controls the servo-motor, ensuring consistent and continuous traction

The AIM OF THE STUDY is to stabilize the pendulum such that the position of the carriage on the track is controlled quickly and accurately and that the pendulum is always maintained tightly in its inverted position during such movements.

The problem involves a cart, able to move backwards and forwards, and a pendulum, hinged to the cart at the bottom of its length such that the pendulum can move in the same plane as the cart, shown below. That is, the pendulum mounted on the cart is free to fall along the cart's axis of motion. The system is to be controlled so that the pendulum remains balanced and upright, and is resistant to a step disturbance.

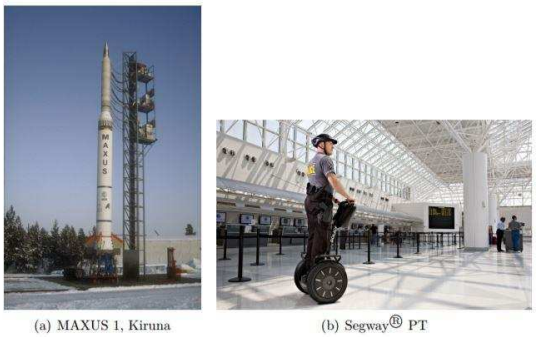
The inverted pendulum cart runs along a track and is pulled by a belt connected to an electric motor. A potentiometer measures the cart position from its rotation and another potentiometer measures the angle of the pendulum, since the pendulum will fall down if we release it with a small angle. To stabilize the system, a feedback control system must be used.

So briefly, the Inverted Pendulum system is made up of a cart and a pendulum. The goal of the controller is to move the cart to its commanded position without causing the pendulum to tip over.

For the PID sections of this problem, we will be interested only in the control of the pendulum's position. This is because the technique used in this section is best-suited for single-input, single-output (SISO) systems. Therefore, the design criteria will not deal with the cart's position. However we will investigate the controller's effect on the cart's position, we will design a controller to restore the pendulum to a vertically upward position after it has experienced an impulsive "bump" to the cart.

Employing state-space design techniques, we are more readily able to address a multi-output system. In our case, the inverted pendulum system is single-input, multi-output (SIMO). Therefore, for the state-space section of the Inverted Pendulum example, we will attempt to control both the pendulum's angle and the cart's position.

Applications of Inverted Pendulum:



(a) MAXUS 1, Kiruna (b) Segway® PT

(a): Rocket MAXUS 1 in starting position, Esrange Space Center, Kiruna.
(b): Segway R Personal Transporter (PT), a self-balancing vehicle

ii. System Modeling

Inverted pendulum is a classic control problem. The process is nonlinear and unstable with one input signal and several output signals. The aim is to balance a pendulum vertically.

Mathematical Modeling:

For this example, the physical parameters of the system:-

(M)	mass of the cart	0.5 kg
(m)	mass of the pendulum	0.2 kg
(b)	coefficient of friction for cart	0.1 N/m/sec
(l)	length to pendulum center of mass	0.3 m
(I)	mass moment of inertia of the pendulum	0.006 kg.m^2
(F)	force applied to the cart	
(x)	cart position coordinate	
(theta)	pendulum angle from vertical (down)	

1. Summing the forces in the free-body diagram of the cart in the horizontal direction, you get the following equation of motion.

$$M\ddot{x} + b\dot{x} + N = F$$

2. Summing the forces in the free-body diagram of the pendulum in the horizontal direction, you get the following expression for the reaction force N.

$$N = m\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta$$

3. substituting (2) in (1) we get one of the two governing equations for this system.

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

4. To get the second equation of motion for this system, sum the forces perpendicular to the pendulum. You should get the following equation.

$$P\sin\theta + N\cos\theta - mg\sin\theta = ml\ddot{\theta} + m\ddot{x}\cos\theta$$

5. To get rid of the P and N terms in the equation above, sum the moments about the centroid of the pendulum to get the following equation.

$$-Pl\sin\theta - Nl\cos\theta = I\ddot{\theta}$$

6. Summing (4) and (5), you get the second governing equation.

$$(I + ml^2)\ddot{\theta} + mgl\sin\theta = -ml\ddot{x}\cos\theta$$

Since the analysis and control design techniques we will be employing in this example apply only to linear systems, this set of equations needs to be linearized. Specifically, we will linearize the equations about the vertically upward equilibrium position, $\theta = \pi$, and will assume that the system stays within a small neighborhood of this equilibrium. This assumption should be reasonably valid since under control we desire that the pendulum not deviate more than 20 degrees from the vertically upward position.

Let ϕ represent the deviation of the pendulum's position from equilibrium, that is, $\theta = \pi + \phi$. Again presuming a small deviation (ϕ) from equilibrium.

7, 8, 9. Using the following small angle approximations of the nonlinear functions in our system equations:

$$\cos\theta = \cos(\pi + \phi) \approx -1$$
$$\sin\theta = \sin(\pi + \phi) \approx -\phi$$
$$\dot{\theta}^2 = \dot{\phi}^2 \approx 0$$

10, 11. Substituting the above approximations into our nonlinear governing equations, we get the two linearized equations of motion. Note u has been substituted for the input F.

$$(I + ml^2)\ddot{\phi} - mgl\phi = ml\ddot{x}$$

$$(M + m)\ddot{x} + b\dot{x} - ml\ddot{\phi} = u$$

Transfer Function:

To obtain the transfer functions of the linearized system equations, we must first take the Laplace transform of the system equations assuming zero initial conditions.

12, 13.The resulting Laplace transforms are shown below.

$$(I + ml^2)\Phi(s)s^2 - mgl\Phi(s) = mlX(s)s^2$$
$$(M + m)X(s)s^2 + bX(s)s - ml\Phi(s)s^2 = U(s)$$

Since that the transfer function represents the relationship between a single input and a single output (SISO) at a time.

To get the first transfer function for the output $\Phi(s)$ and an input of U(s) we need to eliminate X(s) from the above equations.

14. Solving the first equation for X(s).

$$X(s) = \left[\frac{I + ml^2}{ml} - \frac{g}{s^2}\right]\Phi(s)$$

15. Then substitute the above into the second equation.

$$(M + m)\left[\frac{I + ml^2}{ml} - \frac{g}{s^2}\right]\Phi(s)s^2 + b\left[\frac{I + ml^2}{ml} - \frac{g}{s^2}\right]\Phi(s)s - ml\Phi(s)s^2 = U(s)$$

16. Therefore we get this transfer function.

$$\frac{\Phi(s)}{U(s)} = \frac{\frac{ml}{q}s^2}{s^4 + \frac{b(I+ml^2)}{q}s^3 - \frac{(M+m)mgl}{q}s^2 - \frac{bmgl}{q}s}$$

17. Where

$$q = [(M + m)(I + ml^2) - (ml)^2]$$

From the transfer function above it can be seen that there is both a pole and a zero at the origin.

18. These are canceled by.

$$P_{pend}(s) = \frac{\Phi(s)}{U(s)} = \frac{\frac{ml}{q}s}{s^3 + \frac{b(I+ml^2)}{q}s^2 - \frac{(M+m)mgl}{q}s - \frac{bmgl}{q}} \qquad [\frac{rad}{N}]$$

19. The transfer function with the cart position X(s) as the output can be derived in a similar manner to arrive at the following.

$$P_{cart}(s) = \frac{X(s)}{U(s)} = \frac{\frac{(I+ml^2)s^2 - gml}{q}}{s^4 + \frac{b(I+ml^2)}{q}s^3 - \frac{(M+m)mgl}{q}s^2 - \frac{bmgl}{q}s} \qquad [\frac{m}{N}]$$

State Space Representation:

The linearized equations of motion from above can also be represented in state-space form if they are rearranged into a series of first order differential equations. Since the equations are linear, they can then be put into the standard matrix form.

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-(I+ml^2)b}{I(M+m)+Mml^2} & \frac{m^2gl^2}{I(M+m)+Mml^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-mlb}{I(M+m)+Mml^2} & \frac{mgl(M+m)}{I(M+m)+Mml^2} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{I+ml^2}{I(M+m)+Mml^2} \\ 0 \\ \frac{ml}{I(M+m)+Mml^2} \end{bmatrix} u$$

$$\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u$$

:

Note:

The C matrix has 2 rows because both the cart's position and the pendulum's position are part of the output.

iii. Mat-Lab System Modeling

Transfer Function:

We can represent the transfer functions derived above for the inverted pendulum system within MATLAB employing the following commands.

Mat-Lab Input:-

```
M = 0.5;
m = 0.2;
b = 0.1;
I = 0.006;
g = 9.8;
l = 0.3;
q = (M+m)*(I+m*l^2)-(m*l)^2;
s = tf('s');

P_cart = (((I+m*l^2)/q)*s^2 - (m*g*l/q))/(s^4 +
(b*(I + m*l^2))*s^3/q - ((M + m)*m*g*l)*s^2/q -
b*m*g*l*s/q);

P_pend = (m*l*s/q)/(s^3 + (b*(I + m*l^2))*s^2/q -
((M + m)*m*g*l)*s/q - b*m*g*l/q);

sys_tf = [P_cart ; P_pend];

inputs = {'u'};
outputs = {'x'; 'phi'};

set(sys_tf,'InputName',inputs)
set(sys_tf,'OutputName',outputs)

sys_tf
```

Mat-Lab Output:-

```
sys_tf =

  From input "u" to output...
                      4.182e-06 s^2 - 0.0001025
   x:  ---------------------------------------------------------
       2.3e-06 s^4 + 4.182e-07 s^3 - 7.172e-05 s^2 - 1.025e-05 s

                               1.045e-05 s
   phi:  ------------------------------------------------------
         2.3e-06 s^3 + 4.182e-07 s^2 - 7.172e-05 s - 1.025e-05

Continuous-time transfer function.
```

State Space:-

We can represent the system using the state-space equations.

Mat-Lab Input:-

```
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 matrices

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 = [1 0 0 0;
     0 0 1 0];
D = [0;
     0];

states = {'x' 'x_dot' 'phi' 'phi_dot'};
inputs = {'u'};
outputs = {'x'; 'phi'};

sys_ss =
ss(A,B,C,D,'statename',states,'inputname',inputs,'outputname',out
puts)
```

Mat-Lab Output:-

```
Trial>> StateSpace

sys_ss =

  A =
                x     x_dot       phi   phi_dot
   x            0         1         0         0
   x_dot        0   -0.1818     2.673         0
   phi          0         0         0         1
   phi_dot      0   -0.4545     31.18         0

  B =
                u
   x            0
   x_dot    1.818
   phi          0
   phi_dot  4.545

  C =
              x     x_dot       phi   phi_dot
   x          1         0         0         0
   phi        0         0         1         0

  D =
        u
   x    0
   phi  0

Continuous-time state-space model.
```

State Space to Transfer Function:-

The above state-space model can also be converted into transfer function.

Mat-Lab Input:-

```
sys_tf = tf(sys_ss)
```

Mat-Lab Output:-

```
sys_tf =

  From input "u" to output...
          1.818 s^2 + 1.615e-15 s - 44.55
   x:  ---------------------------------------
       s^4 + 0.1818 s^3 - 31.18 s^2 - 4.455 s

                 4.545 s - 1.277e-16
   phi:  ---------------------------------
         s^3 + 0.1818 s^2 - 31.18 s - 4.455

Continuous-time transfer function.
```

iv. System Analysis

Main Problem and Design Requirements :-

The open-loop transfer functions of the inverted pendulum system as the following.

$$P_{pend}(s) = \frac{\Phi(s)}{U(s)} = \frac{\frac{ml}{q}s}{s^3 + \frac{b(I+ml^2)}{q}s^2 - \frac{(M+m)mgl}{q}s - \frac{bmgl}{q}} \quad [\frac{rad}{N}]$$

$$P_{cart}(s) = \frac{X(s)}{U(s)} = \frac{\frac{(I+ml^2)s^2 - gml}{q}}{s^4 + \frac{b(I+ml^2)}{q}s^3 - \frac{(M+m)mgl}{q}s^2 - \frac{bmgl}{q}s} \quad [\frac{m}{N}]$$

Where:

$$q = (M+m)(I+ml^2) - (ml)^2$$

The above two transfer functions are valid only for small values of the angle *phi* where ϕ is the deviation of the pendulum from the vertically upward position. Also, the absolute pendulum angle θ is equal to $\pi + \phi$.

Considering the response of the pendulum at 1-Nsec impulse applied to the cart, the design requirements for the pendulum are:

- Settling time for θ of less than 5 seconds
- Pendulum angle θ never more than 0.05 radians from the vertical

Considering the response of the system at a 0.2-meter step the design requirements for the cart position are:

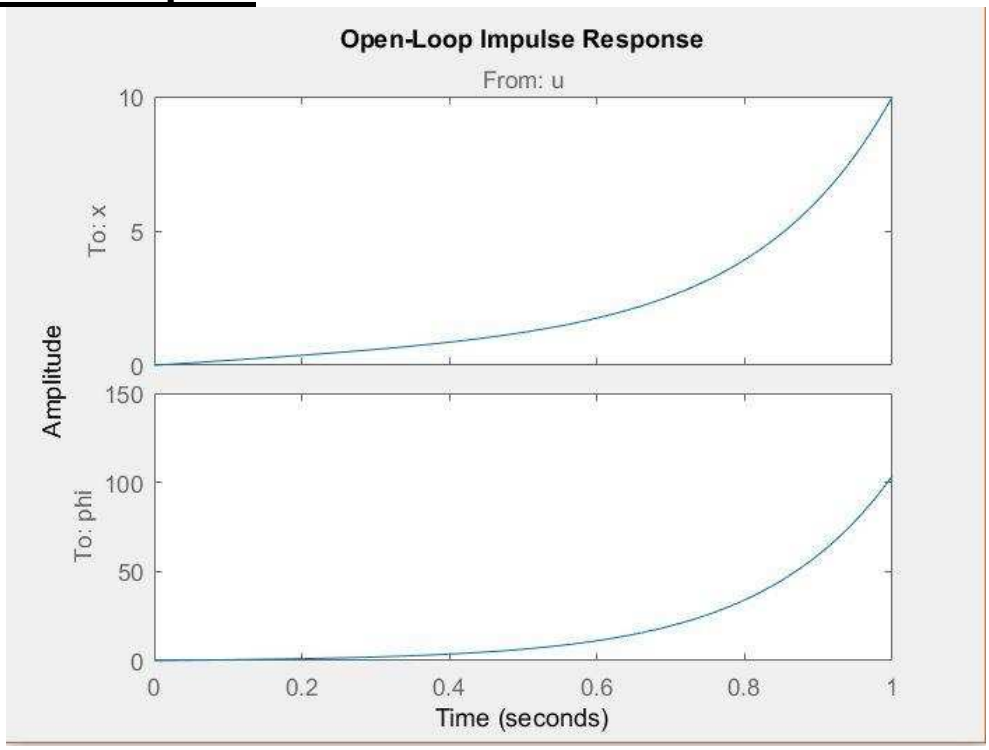
- Settling time for x and θ of less than 5 seconds
- Rise time for x of less than 0.5 seconds
- Pendulum angle θ never more than 20 degrees (0.35 radians) from the vertical

Open-Loop Impuls Response:-

Mat-Lab Input:-

```
M = 0.5;
m = 0.2;
b = 0.1;
I = 0.006;
g = 9.8;
l = 0.3;
q = (M+m)*(I+m*l^2)-(m*l)^2;
s = tf('s');
P_cart = (((I+m*l^2)/q)*s^2 - (m*g*l/q))/(s^4 + (b*(I +
m*l^2))*s^3/q - ((M + m)*m*g*l)*s^2/q - b*m*g*l*s/q);
P_pend = (m*l*s/q)/(s^3 + (b*(I + m*l^2))*s^2/q - ((M +
m)*m*g*l)*s/q - b*m*g*l/q);
sys_tf = [P_cart ; P_pend];
inputs = {'u'};
outputs = {'x'; 'phi'};
set(sys_tf,'InputName',inputs)
set(sys_tf,'OutputName',outputs)
t=0:0.01:1;
impulse(sys_tf,t);
title('Open-Loop Impulse Response')
```

Mat-Lab Output:-



As seen from the plot, the system response is un-stable in open loop. Although the pendulum's position is shown to increase past 100 radians (15 revolutions), the model is only valid for small (phi). Also observe that the cart's position moves very far to the right, though there is no requirement on cart position for an impulsive force input.

The poles of a system can also tell us about its time response. Since our system has two outputs and one input, it is described by two transfer functions. We will specifically examine the poles and zeros of the system using the Mat-Lab. The parameter 'v' shown below returns the poles and zeros as column vectors rather than as cell arrays.

Pendulum Poles and Zeros:

Mat-Lab Input:-

```
[zeros poles] = zpkdata(P_pend,'v')
```

Mat-Lab Output:-

```
zeros =

     0


poles =

    5.5651
   -5.6041
   -0.1428
```

Cart Poles and Zeros:

Mat-Lab Input:-

```
[zeros poles] = zpkdata(P_cart,'v')
```

Mat-Lab Output:-

```
zeros =

    4.9497
   -4.9497


poles =

         0
    5.5651
   -5.6041
   -0.1428
```

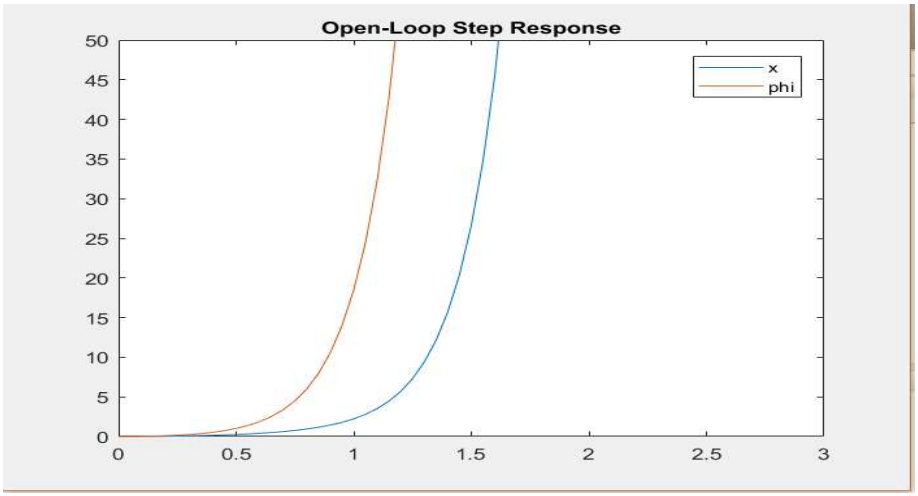
The pole at 5.5651 indicates that the system is unstable since the pole has positive real part. In other words, the pole is in the right half of the complex s-plane, therefore the response to a step input will also grow unbounded

Open-Loop Step Response:-

Mat-Lab Input:-

```
t = 0:0.05:10;
u = ones(size(t));
[y,t] = lsim(sys_tf,u,t);
plot(t,y)
title('Open-Loop Step Response')
axis([0 3 0 50])
legend('x','phi')
```

Mat-Lab Input:-



Characteristics of the response:-

Mat-Lab Input:-

```
step_info = lsiminfo(y,t);
cart_info = step_info(1)
pend_info = step_info(2)
```

Mat-Lab Output:-

```
cart_info =

  struct with fields:

    SettlingTime: 9.9959
             Min: 0
         MinTime: 0
             Max: 8.7918e+21
         MaxTime: 10


pend_info =

  struct with fields:

    SettlingTime: 9.9959
             Min: 0
         MinTime: 0
             Max: 1.0520e+23
         MaxTime: 10
```

The above results confirm our expectation that the system's response to a step input is unstable. It is apparent from the analysis above that some sort of control will need to be designed to improve the response of the system.

v. PID Control Design

Objective:

PID controller will be designed for the inverted pendulum system. In the design process we will assume a single-input, single-output system, with attempt to control the pendulum's angle without regard for the cart's position, subject to the following Transfer Function.

$$P_{pend}(s) = \frac{\Phi(s)}{U(s)} = \frac{\frac{ml}{q}s}{s^3 + \frac{b(I+ml^2)}{q}s^2 - \frac{(M+m)mgl}{q}s - \frac{bmgl}{q}} \qquad [\frac{rad}{N}]$$

Where

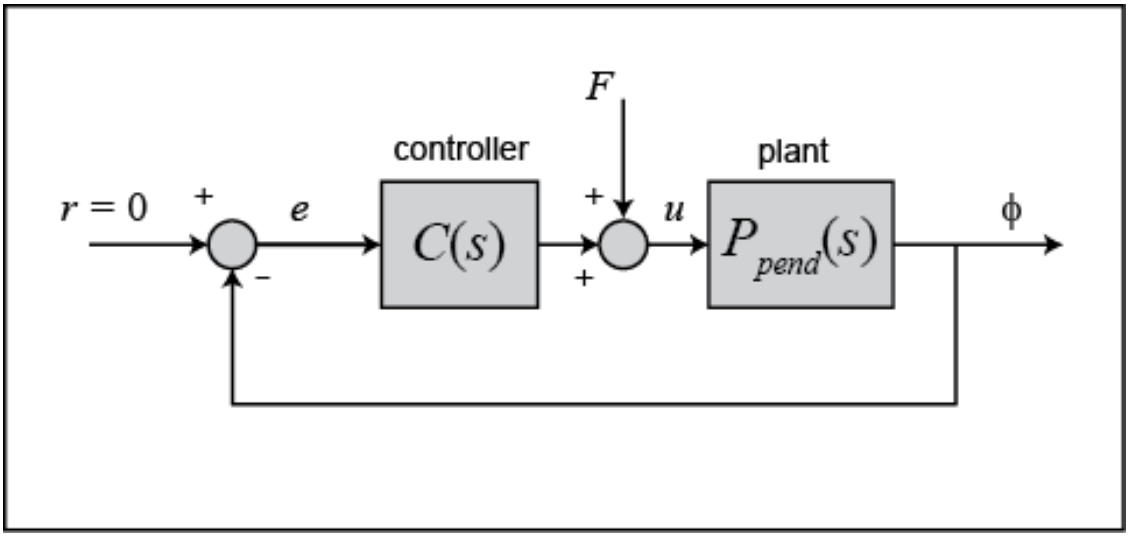
$$q = (M+m)(I+ml^2) - (ml)^2$$

The controller will attempt to maintain the pendulum vertically upward when the cart is subjected to a 1-Nsec impulse. Under these conditions, the design criteria are:

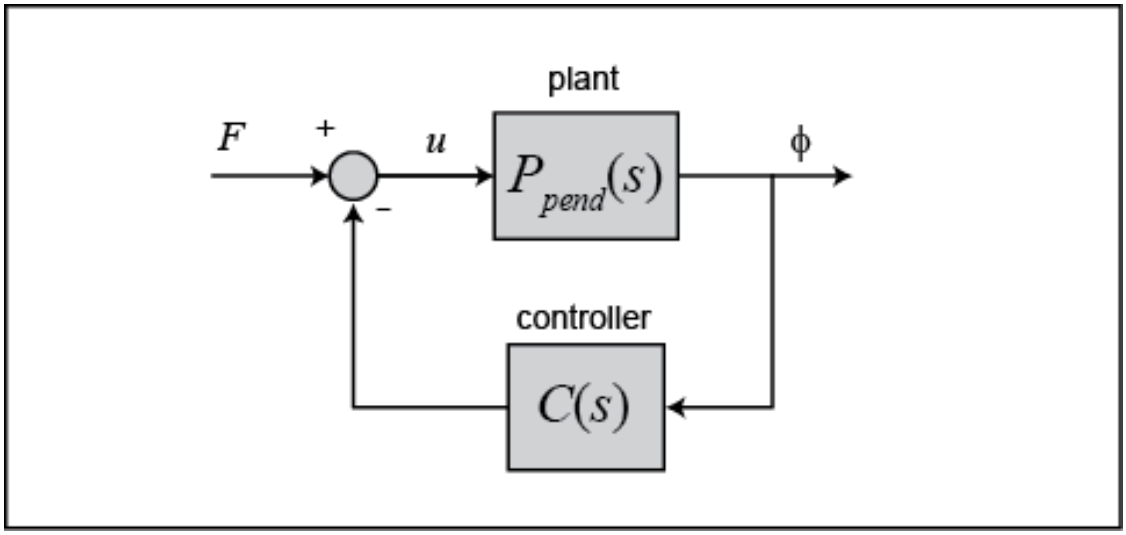
- Settling time of less than 5 seconds
- Pendulum should not move more than 0.05 radians away from the vertical

System Structure:

We are attempting to control the pendulum's position, which should return to the vertical after the initial disturbance, the reference signal we are tracking should be zero. This type of situation is often referred to as a regulator problem. The external force applied to the cart can be considered as an impulsive disturbance. The Block Diagram for this problem is depicted below.



To ease the design, the Block Diagram will be arranged to the following form.



The resulting transfer function for the closed-loop system from an input of force to an output of pendulum angle (PHI) is then determined to be the following.

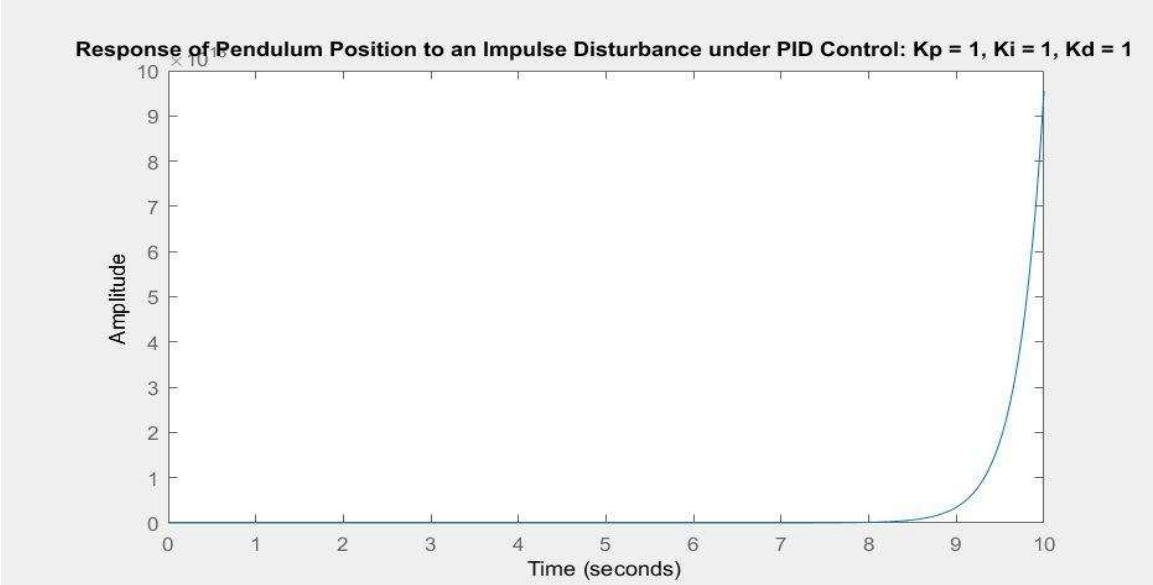
$$T(s) = \frac{\Phi(s)}{F(s)} = \frac{P_{pend}(s)}{1 + C(s)}$$

Pendulum Control via PID Control:-

Mat-Lab Input:

```
M = 0.5;
m = 0.2;
b = 0.1;
I = 0.006;
g = 9.8;
l = 0.3;
q = (M+m)*(I+m*l^2)-(m*l)^2;
s = tf('s');
P_pend = (m*l*s/q)/(s^3 + (b*(I + m*l^2))*s^2/q - ((M +
m)*m*g*l)*s/q - b*m*g*l/q);
Kp = 1;
Ki = 1;
Kd = 1;
C = pid(Kp,Ki,Kd);
t=0:0.01:10;
impulse(T,t)
title('Response of Pendulum Position to an Impulse Disturbance
under PID Control: Kp = 1, Ki = 1, Kd = 1');
T = feedback(P_pend,C);
```

Mat-Lab Output:-



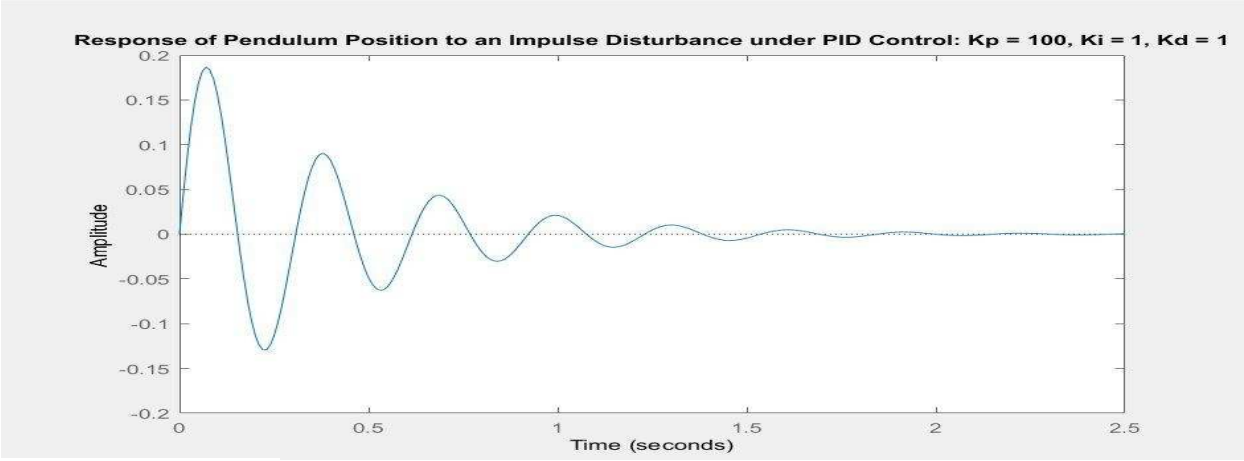
At the response of the closed-loop system at an impulse disturbance for this initial set of control gains this response is still Un-stable.

So we try to modify the response by increasing the proportional gain by increasing the Kp variable to see what effect it has on the response.

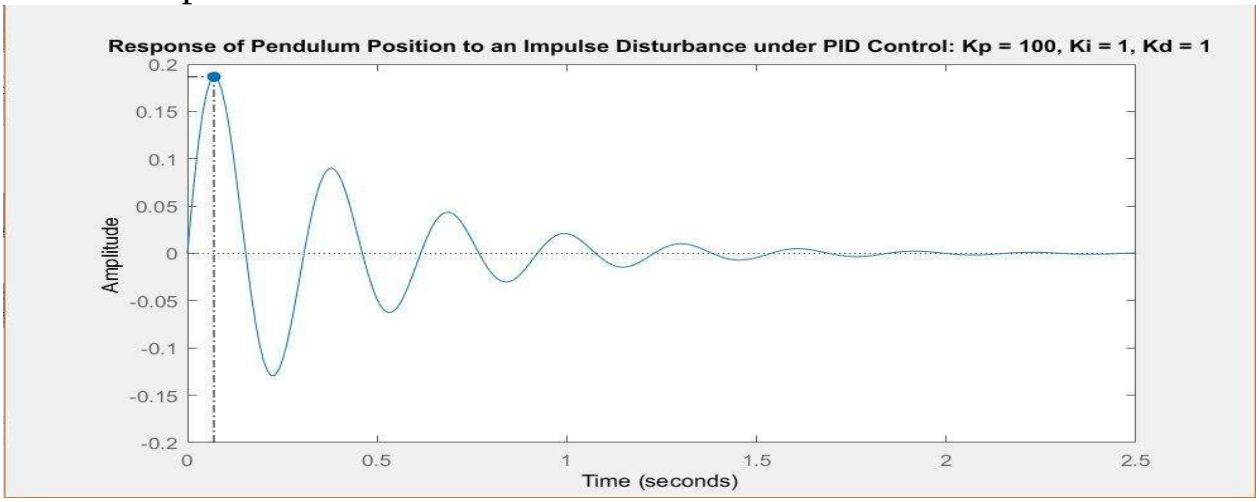
Mat-Lab Input:-

```
Kp = 100;
Ki = 1;
Kd = 1;
C = pid(Kp,Ki,Kd);
T = feedback(P_pend,C);
t=0:0.01:10;
impulse(T,t)
axis([0, 2.5, -0.2, 0.2]);
title('Response of Pendulum Position to an Impulse Disturbance
under PID Control: Kp = 100, Ki = 1, Kd = 1');
```

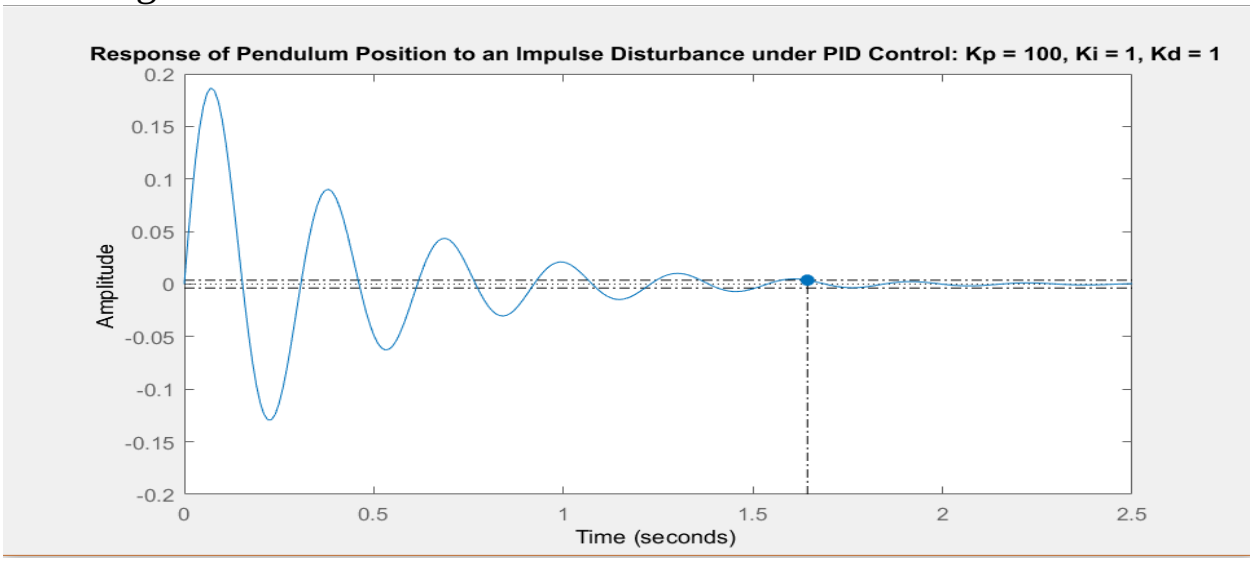
Mat-Lab Output:-



*Peak Response



*Settling Time



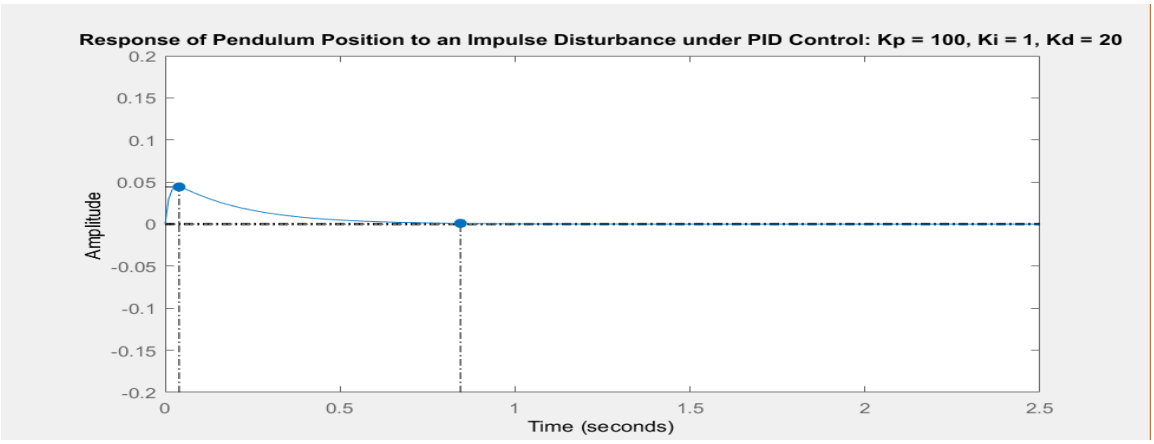
The Results allows you to identify important characteristics of the response. Specifically, the settling time of the response is determined to be 1.64 seconds, which is less than the requirement of 5 seconds. Since the steady-state error approaches zero in a sufficiently fast manner, no additional integral action is needed

Setting the integral gain constant Ki to zero to observe that some integral control is needed. The peak response, however, is larger than the requirement of 0.05 radians. The overshoot often can be reduced by increasing the amount of derivative control. After some trial and error it is found that a derivative gains of Kd = 20 provide a satisfactory response.

Mat-Lab Input:-

```
Kp = 100;
Ki = 1;
Kd = 20;
C = pid(Kp,Ki,Kd);
T = feedback(P_pend,C);
t=0:0.01:10;
impulse(T,t)
axis([0, 2.5, -0.2, 0.2]);
title('Response of Pendulum Position to an Impulse Disturbance
under PID Control: Kp = 100, Ki = 1, Kd = 20')
```

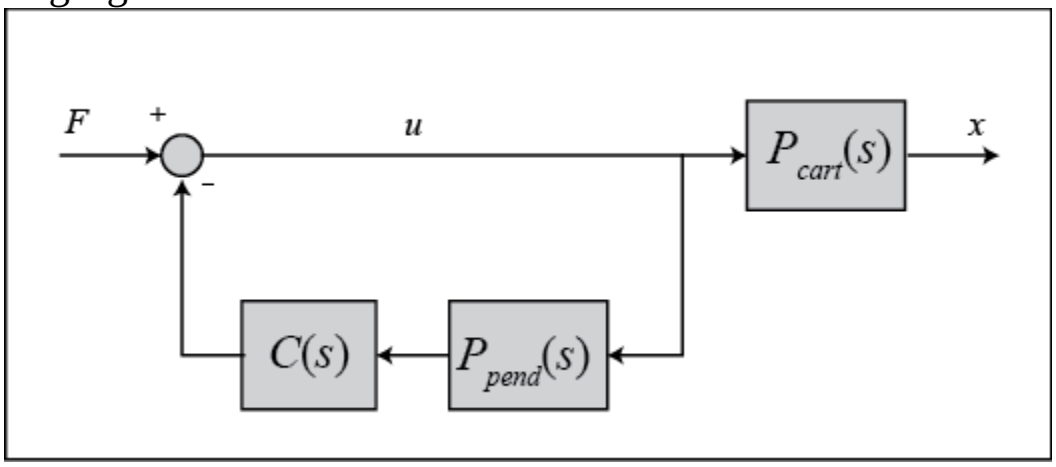
Mat-Lab Output:-



Since the overshoot has been reduced the pendulum does not move more than 0.05 radians away from the vertical. Since all of the given design requirements have been met, no further iteration is needed.

Integrating the Cart Into The System:

In the designed system the block representing the response of the cart's position was not included because that variable is not being controlled. To see what is happening to the cart's position when the controller for the pendulum's angle (PHI) is in place. We need to consider the full system block diagram as shown in the following figure.



The block C(s) is the controller designed for maintaining the pendulum vertical. The closed-loop transfer function T2(s) from an input force applied to the cart to an output of cart position is, therefore, given by the following.

$$T_2(s) = \frac{X(s)}{F(s)} = \frac{P_{cart}(s)}{1 + P_{pend}(s)C(s)}$$

And the transfer function for Pcart(s) is defined as follows.

$$P_{cart}(s) = \frac{X(s)}{U(s)} = \frac{\frac{(I+ml^2)s^2 - gml}{q}}{s^4 + \frac{b(I+ml^2)}{q}s^3 - \frac{(M+m)mgl}{q}s^2 - \frac{bmgl}{q}s} \quad [\frac{m}{N}]$$

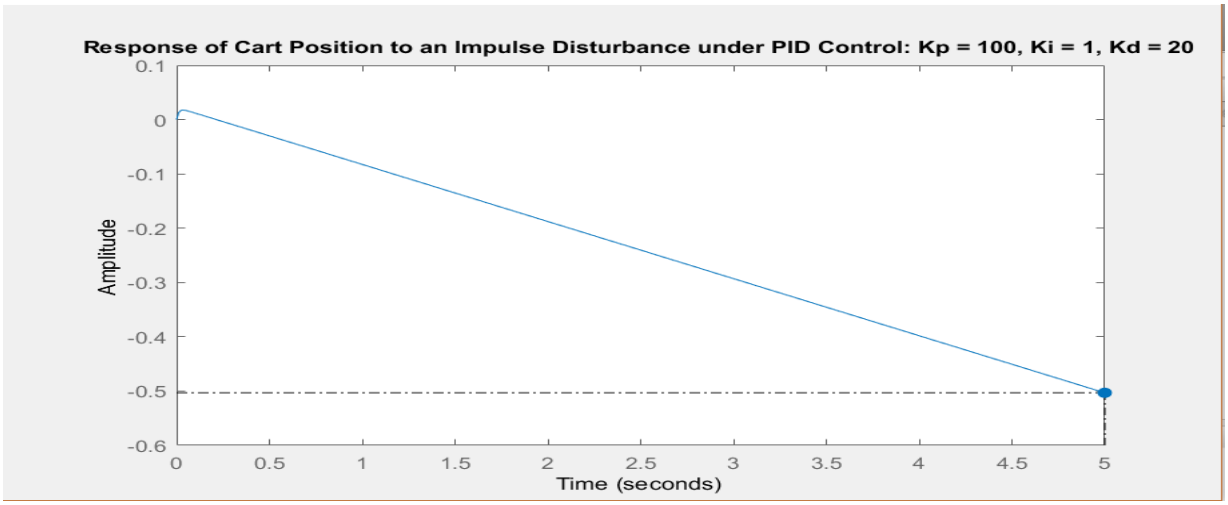
Where

$$q = [(M+m)(I+ml^2) - (ml)^2]$$

Mat-Lab Input:-

```
P_cart = (((I+m*l^2)/q)*s^2 - (m*g*l/q))/(s^4 + (b*(I +
m*l^2))*s^3/q - ((M + m)*m*g*l)*s^2/q - b*m*g*l*s/q);
T2 = feedback(1,P_pend*C)*P_cart;
t = 0:0.01:5;
impulse(T2, t);
title('Response of Cart Position to an Impulse Disturbance under
PID Control: Kp = 100, Ki = 1, Kd = 20');
```

Mat-Lab Output:



As you can see, the cart moves in the negative direction with approximately constant velocity. Therefore, although the PID controller stabilizes the angle of the pendulum, this design would not be feasible to implement on an actual physical system

vi. State Space Control Design

Objective:

After finding the shortcomings of the PID Controller in controlling the system, a new topology of design is tried in order to sufficiently control all aspects of the system.

In this method, we will use the state-space method to design the digital controller. The linearized state-space equations were derived as:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-(I+ml^2)b}{I(M+m)+Ml^2} & \frac{m^2gl^2}{I(M+m)+Ml^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-mlb}{I(M+m)+Ml^2} & \frac{mgl(M+m)}{I(M+m)+Ml^2} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{I+ml^2}{I(M+m)+Ml^2} \\ 0 \\ \frac{ml}{I(M+m)+Ml^2} \end{bmatrix} u$$

$$\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u$$

Where,

The outputs are the cart's displacement (X in meters) and the pendulum angle ((PHI) in radians) where (PHI) represents the deviation of the pedulum's position from equilibrium, that is, $\theta = \pi + \phi$.

The design criteria for this system for a 0.2-m step in desired cart position x are as follows:

- Settling time for x and *theta* of less than 5 seconds
- Rise time for x of less than 0.5 seconds
- Pendulum angle θ never more than 20 degrees (0.35 radians) from the vertical
- Steady-state error of less than 2% for x and θ

Discretization of the System:-

First step in designing a digital controller is to convert the above continuous state-space equations to a discrete form.

Mat-Lab Input:-

```
M = 0.5;
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 matrices

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 = [1 0 0 0;
     0 0 1 0];
D = [0;
     0];

states = {'x' 'x_dot' 'phi' 'phi_dot'};
inputs = {'u'};
outputs = {'x'; 'phi'};

sys_ss =
ss(A,B,C,D,'statename',states,'inputname',inputs,'outputname',out
puts);

Ts = 1/100;

sys_d = c2d(sys_ss,Ts,'zoh')
```

Mat-Lab Output:-

```
sys_d =

  A =
                    x        x_dot          phi      phi_dot
   x                1     0.009991    0.0001336    4.453e-07
   x_dot            0       0.9982      0.02672    0.0001336
   phi              0   -2.272e-05        1.002      0.01001
   phi_dot          0    -0.004544       0.3119        1.002

  B =
                    u
   x         9.086e-05
   x_dot       0.01817
   phi       0.0002272
   phi_dot     0.04544

  C =
              x    x_dot      phi  phi_dot
   x          1        0        0        0
   phi        0        0        1        0

  D =
        u
   x    0
   phi  0

Sample time: 0.01 seconds
Discrete-time state-space model.
```

Controllability and Observability:-

The next step is to check the controllability and the observability of the system.

For the system to be completely state controllable, the controllability matrix must have the rank of n:

$$C = \begin{bmatrix} B & AB & A^2B & \cdots & A^{n-1}B \end{bmatrix}$$

For the system to be completely state observable, the observability matrix must have the rank of n.

$$O = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}$$

These tests for controllability and observability are identical to the situation of continuous control except that now the state space model is discrete.

Mat-Lab Input:-

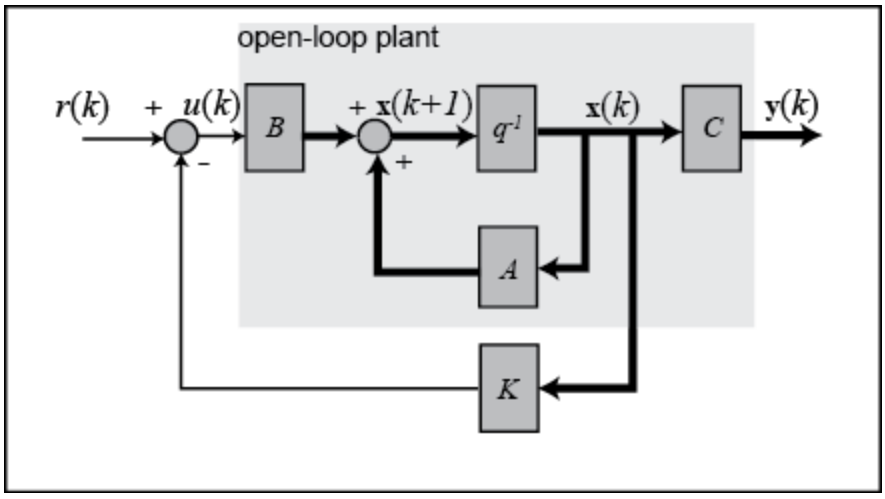
```
co = ctrb(sys_d);
ob = obsv(sys_d);

controllability = rank(co)
observability = rank(ob)
```

Mat-Lab Output:-

```
controllability =

     4


observability =

     4
```

This proves that the system is both completely state controllable and completely state observable.

Control Design via Pole Placement:



Next step is to assume that all four states are measurable and design the control gain matrix K, the Linear Quadratic Regulator (LQR) method was used to find the control gain matrix K that results in the optimal balance between system errors and control effort, To use this LQR method, we need to specify two parameters, the performance index matrix and the state-cost matrix Q. For simplicity, we will initially choose the performance index matrix R equal to 1, and the state-cost matrix Q equal to c'c, the relative weightings of these two matrices will then be tuned by trial and error. The state-cost matrix Q has the following structure.

$$Q = C'C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

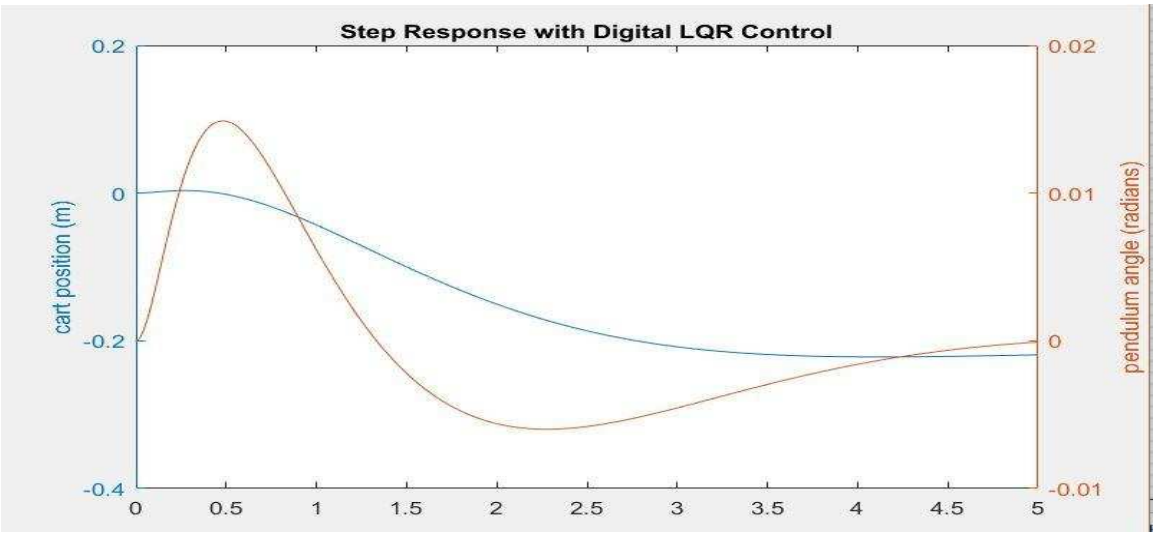
The element in the (1, 1) position of Q represents the weight on the cart's position and the element in the (3, 3) position represents the weight on the pendulum's angle.

Mat-Lab Input:-

```
A = sys_d.a;
B = sys_d.b;
C = sys_d.c;
D = sys_d.d;
Q = C'*C
R = 1;
[K] = dlqr(A,B,Q,R)

Ac = [(A-B*K)];
Bc = [B];
Cc = [C];
Dc = [D];

states = {'x' 'x_dot' 'phi' 'phi_dot'};
inputs = {'r'};
outputs = {'x'; 'phi'};

sys_cl =
ss(Ac,Bc,Cc,Dc,Ts,'statename',states,'inputname',inputs,'outputna
me',outputs);

t = 0:0.01:5;
r =0.2*ones(size(t));
[y,t,x]=lsim(sys_cl,r,t);
[AX,H1,H2] = plotyy(t,y(:,1),t,y(:,2),'plot');
set(get(AX(1),'Ylabel'),'String','cart position (m)')
set(get(AX(2),'Ylabel'),'String','pendulum angle (radians)')
title('Step Response with Digital LQR Control')
```

Mat-Lab Output:-

```
Q =

     1     0     0     0
     0     0     0     0
     0     0     1     0
     0     0     0     0


K =

   -0.9384   -1.5656   18.0351    3.3368
```



The curve in green represents the pendulum's angle in radians, and the curve in blue represents the cart's position in meters. As you can see, this plot is not satisfactory. The pendulum and cart's overshoot appear fine, but their settling times need improvement and the cart's rise time needs to be reduced. the cart's final position is also not near the desired location but has in fact moved in the opposite direction.

Increasing the (1, 1) and (3,3) elements makes the settling and rise times go down, and lowers the angle the pendulum moves. In other words, you are putting more weight on the errors at the cost of increased control effort u.

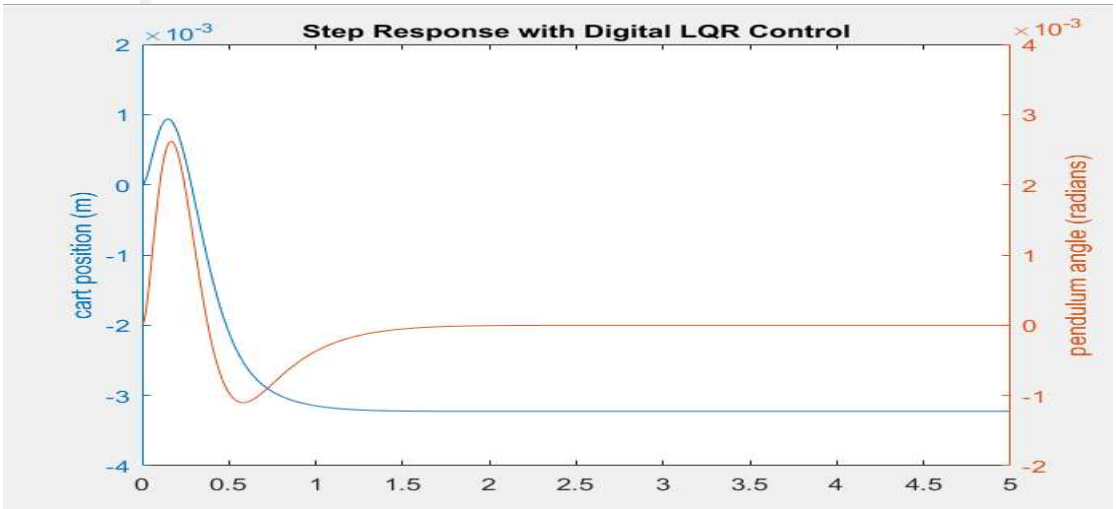
Mat-Lab Input:-

```
A = sys_d.a;
B = sys_d.b;
C = sys_d.c;
D = sys_d.d;
Q = C'*C;
Q(1,1) = 5000;
Q(3,3) = 100
R = 1;
[K] = dlqr(A,B,Q,R)

Ac = [(A-B*K)];
Bc = [B];
```

```
Cc = [C];
Dc = [D];

states = {'x' 'x_dot' 'phi' 'phi_dot'};
inputs = {'r'};
outputs = {'x'; 'phi'};

sys_cl =
ss(Ac,Bc,Cc,Dc,Ts,'statename',states,'inputname',inputs,'outputna
me',outputs);

t = 0:0.01:5;
r =0.2*ones(size(t));
[y,t,x]=lsim(sys_cl,r,t);
[AX,H1,H2] = plotyy(t,y(:,1),t,y(:,2),'plot');
set(get(AX(1),'Ylabel'),'String','cart position (m)')
set(get(AX(2),'Ylabel'),'String','pendulum angle (radians)')
title('Step Response with Digital LQR Control')
```

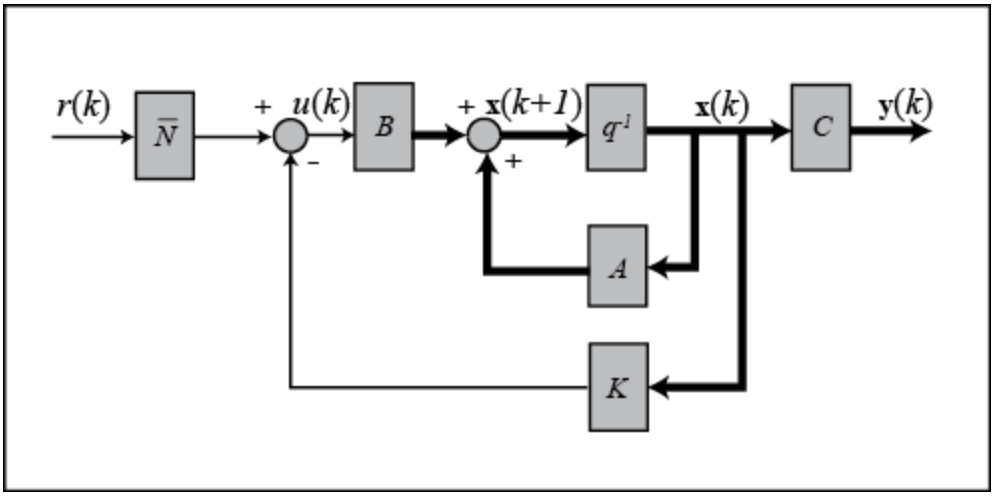
Mat-Lab Output:-

```
Q =

        5000            0            0            0
           0            0            0            0
           0            0          100            0
           0            0            0            0


K =

   -61.9933   -33.5040    95.0597    18.8300
```



From this plot, we see that all design requirements are satisfied except the steady-state error of the cart position x. We can easily correct this by introducing a feed forward scaling factor.

Precompensator Design:-

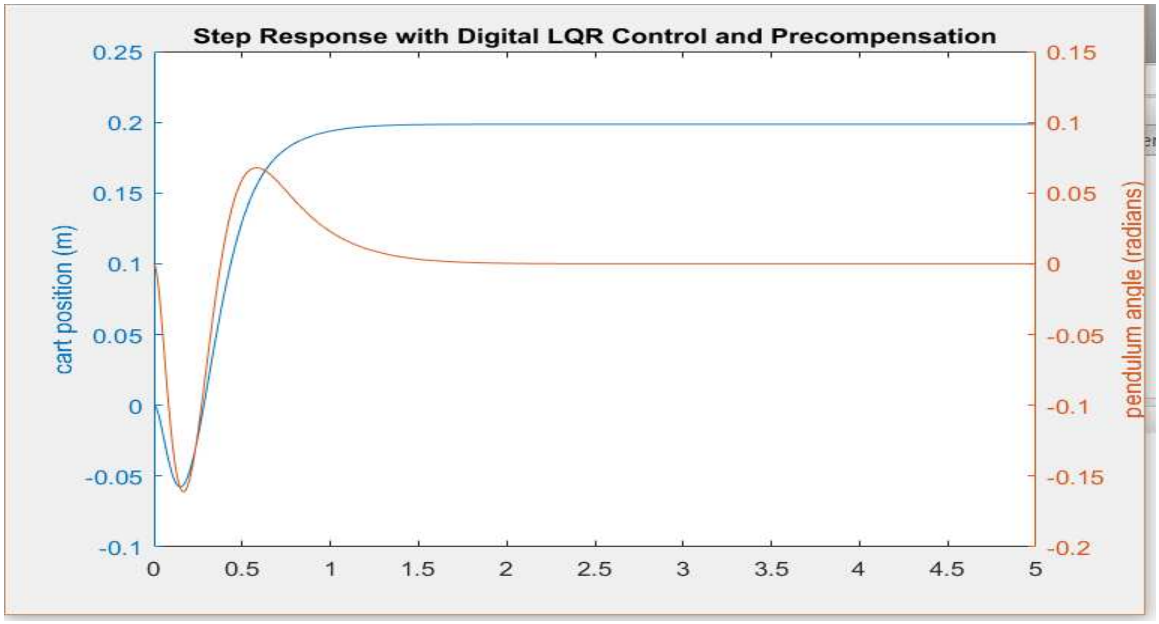


Unlike other design methods, the full-state feedback system does not compare the output directly to the reference, rather, it compares the state vector multiplied by the control matrix (Kx) to the reference. Thus, we should not expect the output to converge to the commanded reference. To obtain the desired output, we need to scale the reference input so that the output equals the reference. This can be easily done by introducing a feed forward scaling factor N. The basic full state-feedback schematic with scaling factor is shown above.

Mat-Lab Input:-

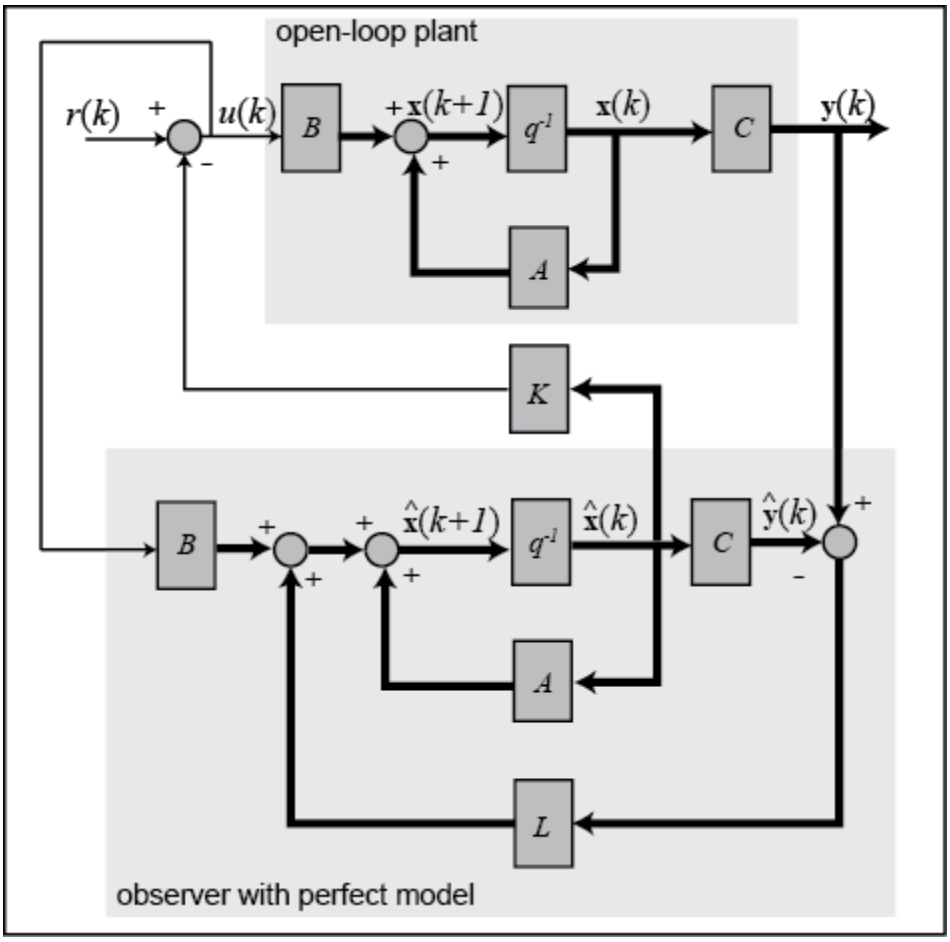
```
Nbar = -61.55;
sys_cl =
ss(Ac,Bc*Nbar,Cc,Dc,Ts,'statename',states,'inputname',inputs,'out
putname',outputs);

t = 0:0.01:5;
r =0.2*ones(size(t));
[y,t,x]=lsim(sys_cl,r,t);
[AX,H1,H2] = plotyy(t,y(:,1),t,y(:,2),'plot');
set(get(AX(1),'Ylabel'),'String','cart position (m)')
set(get(AX(2),'Ylabel'),'String','pendulum angle (radians)')
title('Step Response with Digital LQR Control and
Precompensation')
```

Mat-Lab Output:-



Notice that the steady-state error of the cart's position has been eliminated. Now we have designed a system that satisfies all of the design requirements, however, that the scaling factor N was designed based on a model of the system. If the model is in error or there are unknown disturbances, then the steady-state error will no longer be driven to zero.

Observer Design:-



Observer is a technique for estimating the state of the system based on the measured outputs and a model of the plant, thus in this section we will design a full-order state observer to estimate all of the system's state variables, including those that are measured.

Designing the observer equates to finding the observer gain matrix L, To accomplish this, we need to first determine the closed-loop poles of the system without the observer (the eigenvalues of A-BK).

Mat-Lab Input:-

```
poles = eig(A-B*K)
```

Mat-Lab Output:-

```
poles =

   0.9157 + 0.0728i
   0.9157 - 0.0728i
   0.9535 + 0.0079i
   0.9535 - 0.0079i
```

Since the observer is attempting to estimate the values of state variables which are themselves changing, it is desired that the dynamics of the observer be significantly faster than the dynamics of the closed-loop system without the observer. A common guideline is to make the estimator poles (eigenvalues of A-LC) 4-10 times faster than the slowest controller pole (eigenvalue of A-BK). Making the estimator poles too fast can be problematic if the measurement is corrupted by noise or there are errors in the sensor measurement in general. Based on the poles found above, we will place the observer poles at [-0.2 -0.21 -0.22 -0.23]. These poles can be modified later, if necessary.

Mat-Lab Input:-

```
P = [-0.2 -0.21 -0.22 -0.23];
L = place(A',C',P)'
```

Mat-Lab Output:-

```
L =

    2.4308   -0.0104
  147.6324   -1.2418
   -0.0131    2.4305
   -1.8079  147.9057
```

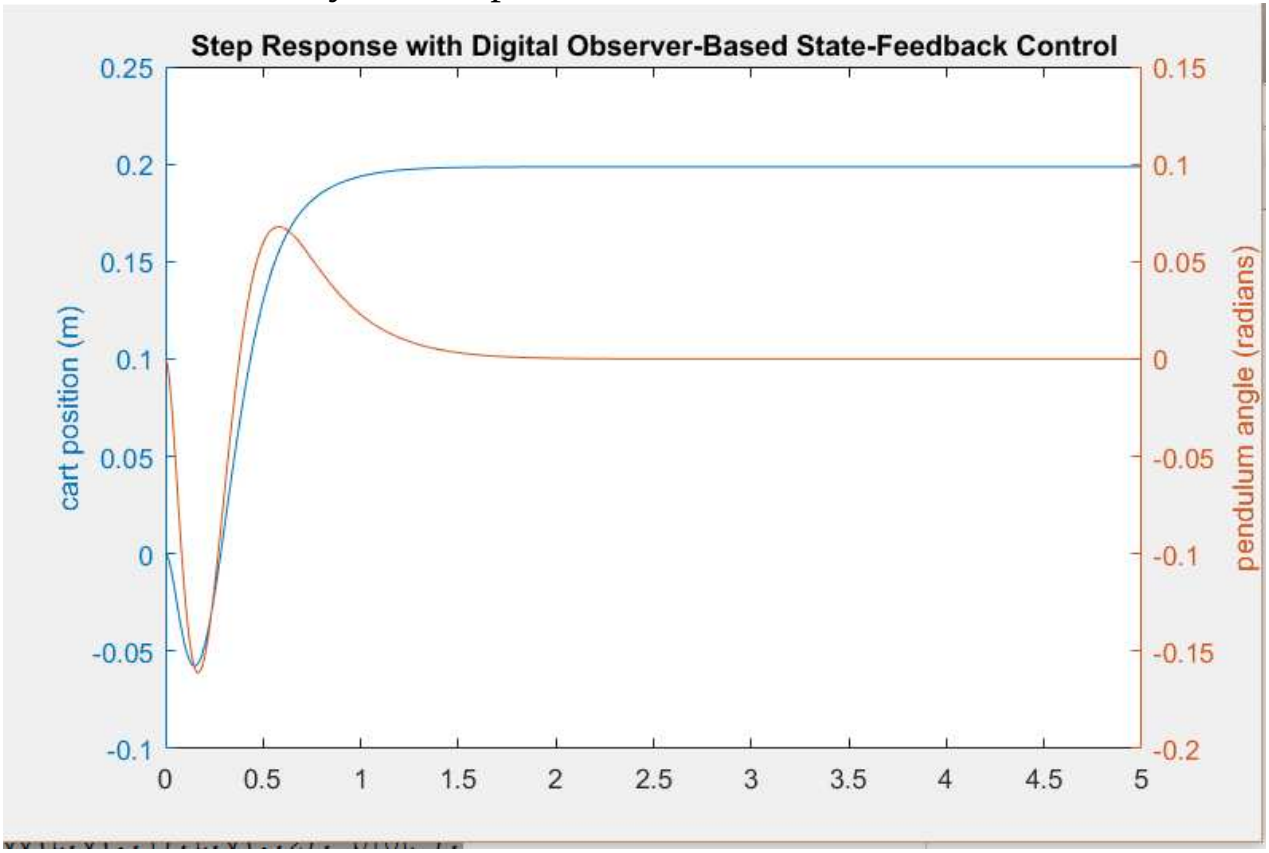
Now we will obtain the overall system response including the observer.

Mat-Lab Input:-

```
Ace = [(A-B*K) (B*K);
       zeros(size(A)) (A-L*C)];
Bce = [B*Nbar;
       zeros(size(B))];
Cce = [Cc zeros(size(Cc))];
Dce = [0;0];

states = {'x' 'x_dot' 'phi' 'phi_dot' 'e1' 'e2' 'e3'
'e4'};
inputs = {'r'};
outputs = {'x'; 'phi'};

sys_est_cl =
ss(Ace,Bce,Cce,Dce,Ts,'statename',states,'inputname',in
puts,'outputname',outputs);

t = 0:0.01:5;
r = 0.2*ones(size(t));
[y,t,x]=lsim(sys_est_cl,r,t);
[AX,H1,H2] = plotyy(t,y(:,1),t,y(:,2),'plot');
set(get(AX(1),'Ylabel'),'String','cart position (m)')
set(get(AX(2),'Ylabel'),'String','pendulum angle
(radians)')
title('Step Response with Digital Observer-Based State-
Feedback Control')
```

Mat-Lab Output:-

obtain the overall system response



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 plant (including the same initial conditions). Therefore, all of the design requirements have been met with the minimal control effort expended. No further iteration is needed.

vii. Conclusion

As we concluded in this research, the Inverted Pendulum can be controlled by PID Controller, but due to the SISO nature of the PID controller, the control of the cart will be neglected, leading to the unfeasibility of introducing the control into a practical application.

The solution was designing a multi-input system, so that the system can be fully controlled effectively.

After the trial of different topologies and after much iteration it was found that the state space observer method would be one of the most effective methods of controlling the system, due to its ability to take all of the system variables into consideration, and effectively controlling them.

viii. References

i. http://s3.amazonaws.com/academia.edu.documents/43005828/paper_furuta.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1495285322&Signature=L0oPw94bikZY856Gm%2BwIXdmpqcI%3D&response-content-disposition=inline%3B%20filename%3DSwing_Up_Control_of_Inverted_Pendulum.pdf

ii. http://s3.amazonaws.com/academia.edu.documents/4133309/p61.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1495285328&Signature=0Ycla9iwMwg540RkbL%2Fihzq4IJc%3D&response-content-disposition=inline%3B%20filename%3DVelocity_and_position_control_of_a_wheel.pdf

iii. https://www.mathworks.com/help/mpc/examples/control-of-an-inverted-pendulum-on-a-cart.html

iv. http://web.mit.edu/klund/www/papers/UNP_pendulum.pdf

v. https://www.control.isy.liu.se/student/tsrt19/ht2/file/invpendpmenglish.pdf