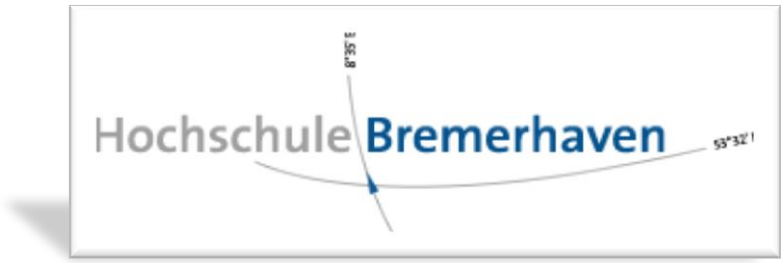


MASTER OF SCIENCE IN EMBEDDED SYSTEM DESIGN

DISCRETE CONTROL SYSTEM LAB REPORT

Under the Guidance of

Prof. Dr. Karsten Peter

Submitted by

Name	***Matriculation number***
Dharmesh Patel	**38073**

Discrete Control Systems Laboratory

1. Make a nonlinear model in Matlab Simulink for the seesaw system.

2. Linearize the seesaw system and make a linear state-space model for the working point zero in Matlab Simulink

3. Compare the linear model and the nonlinear model. When does the linear model become imprecise? How far can the working point be away from the initial working point (in state-space), upon which the linear system was derived from?

4. Make a state-feedback control based on the linear seesaw system. Place all poles at -1. Calculate state feedback vector k and preamplifier p. Verify the control on the linear seesaw system. Test the linear state-feedback vector k and the preamplifier p on the nonlinear seesaw system. Test it for reference value steps and disturbance torque steps.

5. Make a PI-state-feedback control based on the linear seesaw system. Place all poles at -1. Calculate the state feedback vector k based on the augmented system (5th order). Verify the control on the linear seesaw system. Test the linear PI-state-feedback control for reference value steps and disturbance torque steps.

6. Make the PI-part time discrete with a sample rate 8 kHz and test the linear PI-state-feedback on the nonlinear seesaw system.

7. Make a linear full-state observer based on the linear seesaw system. Place all poles at -3. Calculate the error feedback vector h. Validate the observer on the linear seesaw system for different initial conditions. Make the parallel model time discrete (zero-order-hold) with a sample rate of 8 kHz. Test the linear full-state observer on the nonlinear seesaw system with different initial conditions.

8. Use the observed states from 7. for the controls 4. – 6. and test them on the linear and non-linear system for different initial conditions and reference value steps and disturbance torque steps.

Contents

INTRODUCTION

In this Lab we analysing the behavioural differences between linear and non-linear systems. Linear control system applies to systems made of linear devices; which means they obey the superposition principle; the output of the device is proportional to its input. Nonlinear control system covers a wider class of systems that do not obey the superposition principle. Nonlinear control theory is the area of control theory which deals with systems that are nonlinear, time-variant, or both. Control theory is an interdisciplinary branch of engineering and mathematics that is concerned with the behaviour of dynamical systems with inputs, and how to modify the output by changes in the input using feedback. The system to be controlled is called the "plant". In order to make the output of a system follow a desired reference signal a controller is designed which compares the output of the plant to the desired output, and provides feedback to the plant to modify the output to bring it closer to the desired output.

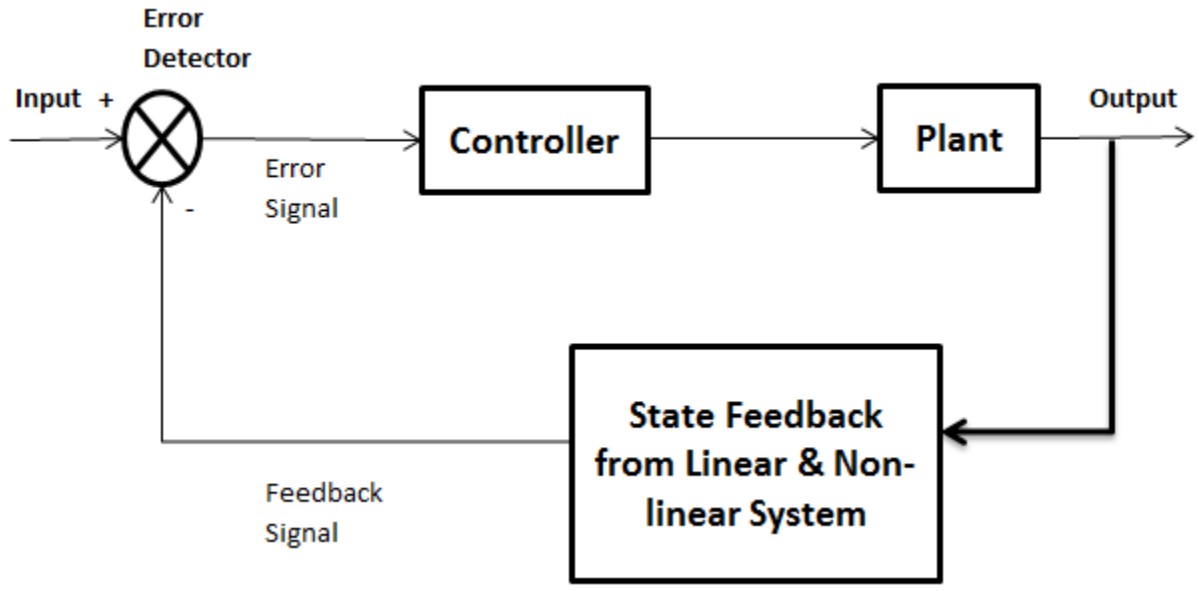


Figure 1. Closed loop feedback control system block diagram

In this report we take a nonlinear seesaw system and analyse its behaviour. We then linearize the system by converting it into a state- space representation. We compare the behaviour of Linear and non-linear system when we give an input.

We will have a look that both the linear and nonlinear systems are not stable, so we will try to stabilize it. We care for BIBS stability of these systems (Bounded Input, Bounded State). Hence we adopt state feedback method which will shift poles of system to desired location (left half of S-plane). We design state-feedback gain using Ackermann's formula. With this new additional feature, i.e., State-feedback we analyse the behaviour of the Linear and nonlinear system.

Though, these systems exhibit stable behaviour after implementing state-feedback, they are not immune to distortion. Stability conditions fail once the distortions are introduced to the system. Hence our next attempt will be to design a PI-Controller which will make systems much more immune to the distortions acting at the input of the system. In most of the applications, PI-Controller is implemented on a processor. Hence we repeat our PI-Controller design for discrete time domain. We analyse the system behaviour with PI controller.

Disadvantage of state-feedback controller is that it requires full information about all states. It implies we need to measure all states of the system which is not possible for many reasons in practical scenarios. Alternative to measuring all states is reconstructing all states with a help of just one state. For this purpose we need an Observer (Parallel model) which reconstructs states of our plant / original system. Therefore, next we make an attempt to design an Observer. We analyse the behaviour of Observer with the given system. Purpose of reconstructing states will be served only if we can use observed states for our state-feedback controller. Hence we analyse the system behaviour using observed states for feed-back controller.

System Overview

Our system can be termed as See-Saw system. It consists of a beam and a cart. On this system a cart is placed at a distance x from the centre as shown in Fig.2. The mass of the cart is m_F and the height from the beam up to the cart is given as h. The torque τ_B is applied at the other end of the seesaw system. Using these variables we build mathematical model for the given system.

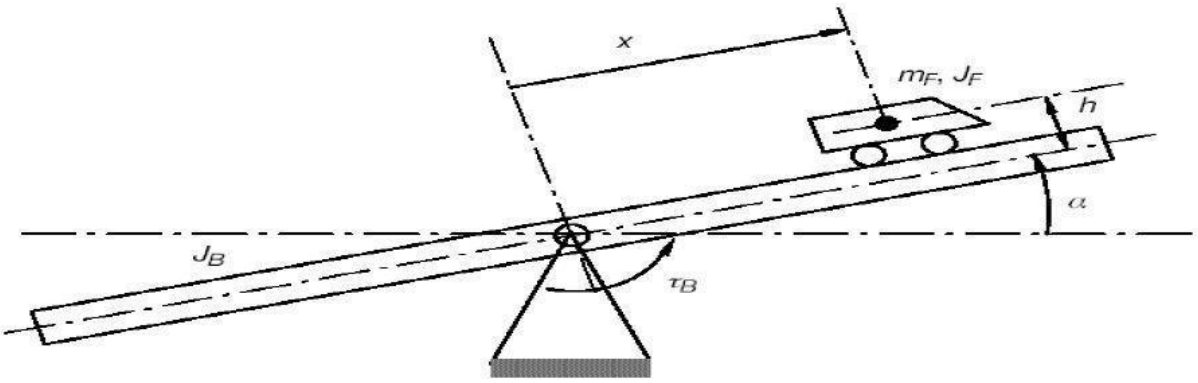


Figure 2. See-Saw System

Using Lagrange-equation: $\frac{\partial}{\partial t}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \left(\frac{\partial L}{\partial q_i}\right) = Q_i$ with $\mathbf{q} = \begin{bmatrix} x \\ \alpha \end{bmatrix}$ and $\mathbf{Q} = \begin{bmatrix} 0 \\ \tau_B \end{bmatrix}$

Nonlinear differential equations of the see-saw system are as follows:

$$\ddot{x} = x \cdot \dot{\alpha}^2 - g.sin\alpha \qquad \text{...Equation (1)}$$

$$\ddot{\alpha} = \frac{\tau_B - m_F \cdot g \cdot x \cdot c(\alpha) + m_F \cdot g \cdot h \cdot \sin(\alpha) - 2 \cdot m_F \cdot x \cdot \dot{x} \cdot \alpha}{J_B + m_F \cdot h^2 + m_F x^2} \qquad \text{...Equation (2)}$$

Where,

α is the tilt angle,

J_B Is the moment of inertia of the beam,

τ_B is the driving torque which is our manipulating variable, and

$J_{const} = J_B + m_F \cdot h^2$

Implementing the system in MATLAB Simulink we are going to perform various tasks to linearize the nonlinear model.

TASK-1

Make a nonlinear model in Matlab Simulink for the seesaw system.

Non-Linear Model

Using nonlinear system equations 1 and 2, the nonlinear model is built in MATLAB Simulink for our given see-saw system.

The following script file with the mentioned constants is used for our system.

Following Equations are related to this Task:

$$J_{const} = J_b + m_F \cdot h^2$$
$$m_F = 0.1\ Kg$$
$$J_b = 0.5\ \text{Kg} \cdot m^2$$

MATLAB Script:

```
% Task-01

Jb = 0.5 % Jb=0.5 Kg· m^2
mf=0.1 %mf = 0.1 KG
h=0.1 % h=10cm
g=9.81 % 9.81 m· s^2

Jconst=Jb+mf*h^2 % J_B + m_F · h^2
```

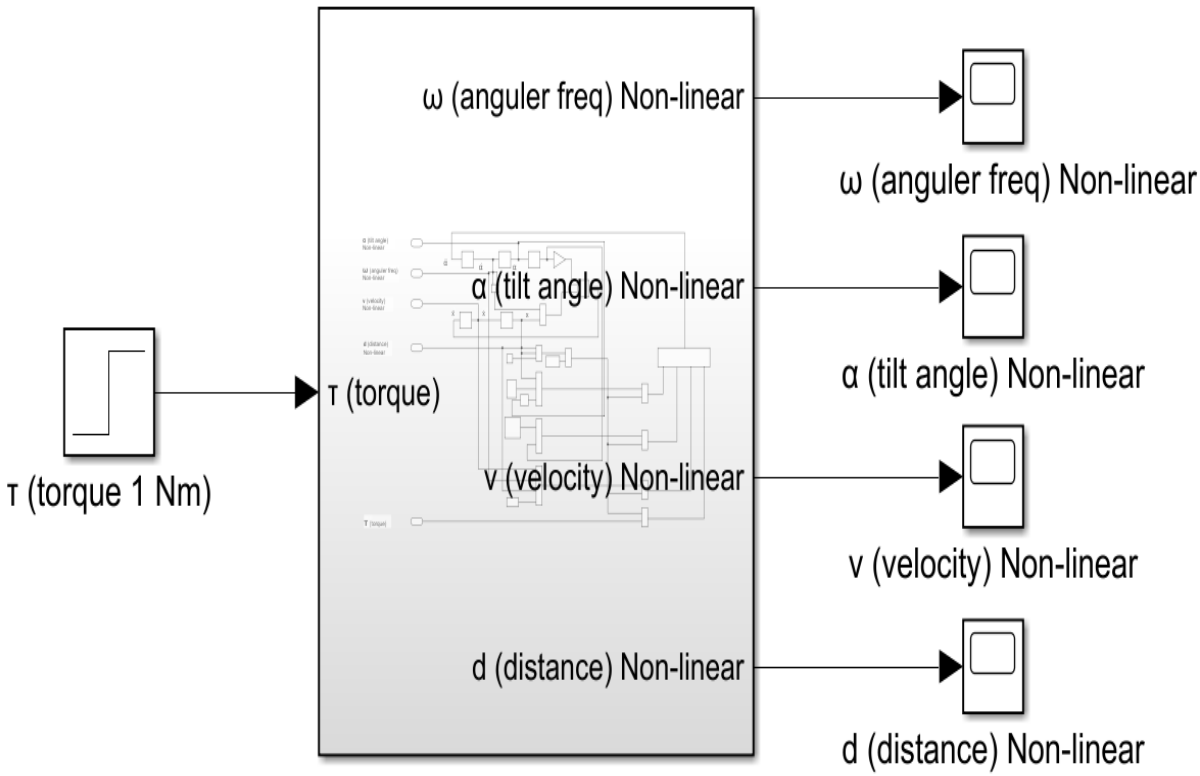


Figure 3. MATLAB Simulink model for non-linear System

Non-linear Subsystem

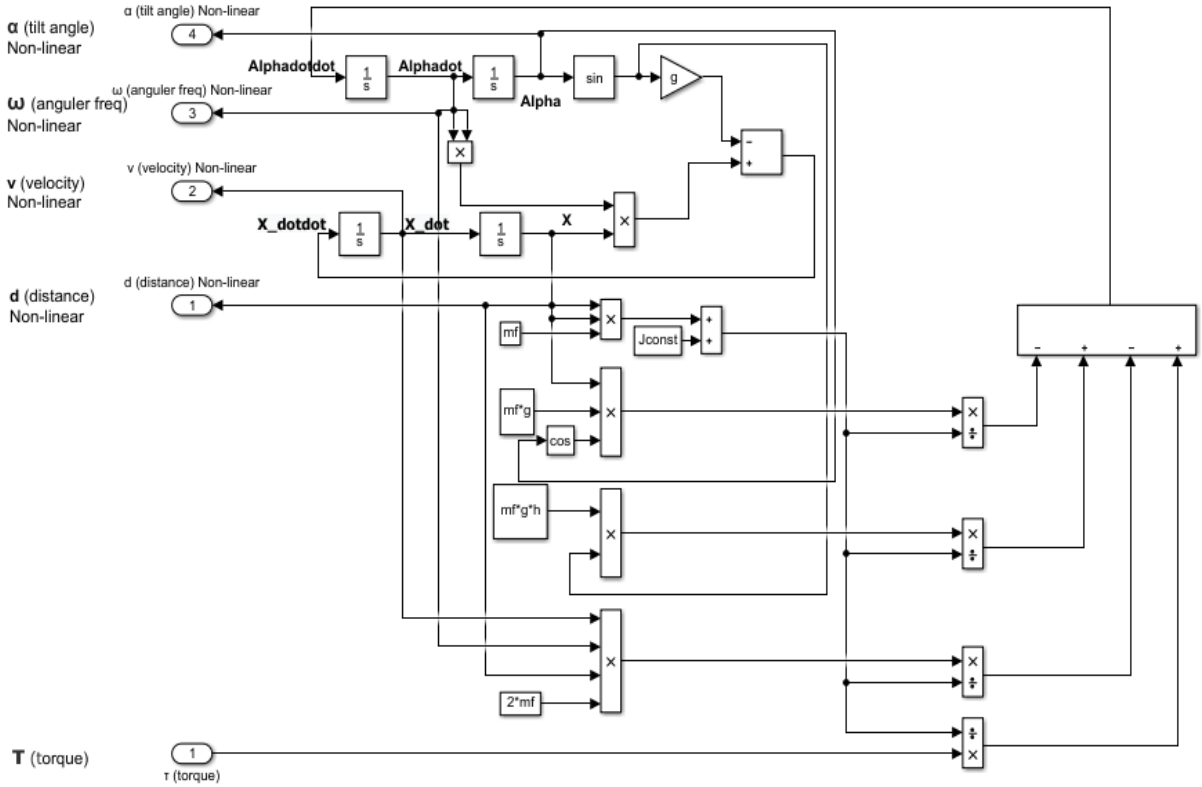
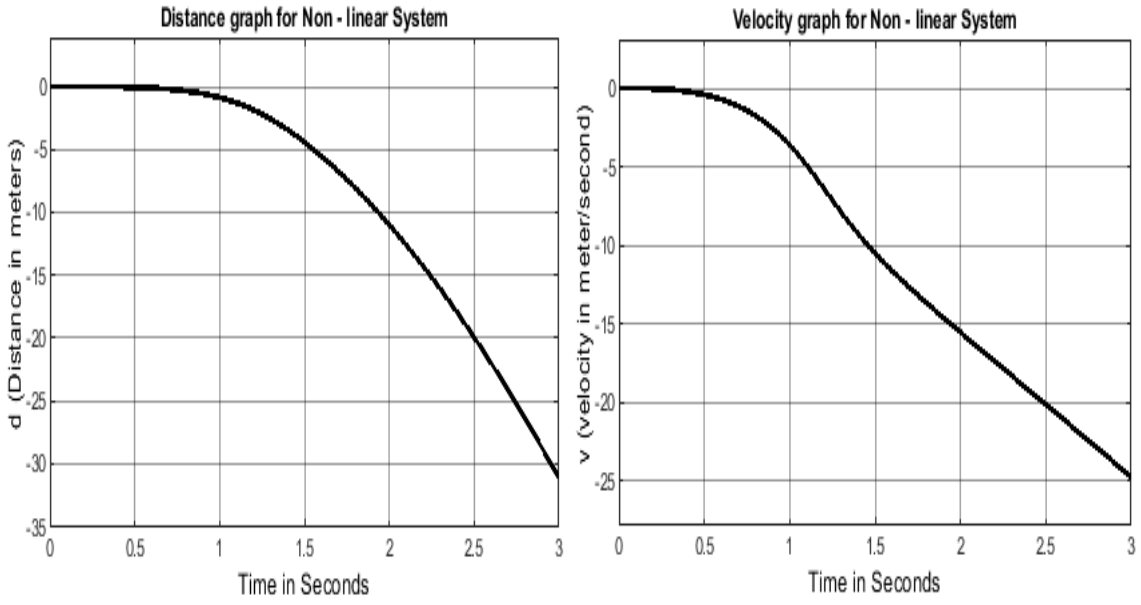
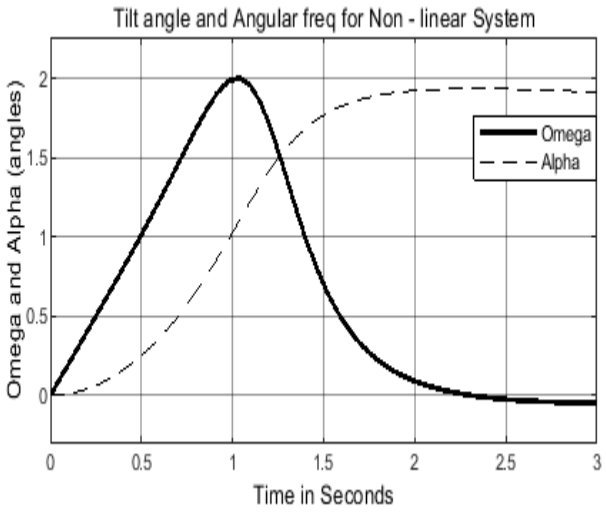


Figure 4. Non-linear Sea-Saw Subsystem

Non-linear Subsystem Simulation output





Graph Description: We could see that the input torque of 1 Nm has been applied to the Non-linear see-saw system. We could notice that distance and velocity increased with respect to the time in negative direction (which means that cart is not steady and moving to downward). Additionally, alpha and omega values are increase initially but after 3 seconds it is constant and it's within the boundary.

TASK-2

Linearize the seesaw system and make a linear state-space model for the working point zero in MATLAB Simulink.

Linear State Space Model

It is known that, nonlinear systems can be linearized around a working point, provided the working point is a solution of the system. In order to make our study of dynamics of the given system easier, we linearize our seesaw system at working point zero which is also known as the trivial solution.

Linearizing around the unstable resting position, we consider

($\alpha = 0, \dot{\alpha} = 0, x = 0, \dot{x} = 0, \tau_B = 0$)

$\ddot{x} = x \cdot \dot{\alpha}^2 - g \cdot sin\alpha$ (nonlinear system equation)

$\ddot{x} = 0 \cdot 0^2 - g \cdot \sin(0)$ Thus, the linearized differential equation will be

$\ddot{x} = -g \cdot \alpha$ ...Equation (3)

and

$$\ddot{\alpha} = \frac{\tau_B - m_F \cdot g \cdot x \cdot c(\alpha) + m_F \cdot g \cdot h \cdot \sin(\alpha) - 2 \cdot m_F \cdot x \cdot \dot{x} \cdot \alpha}{J_B + m_F \cdot h^2 + m_F x^2}$$ (nonlinear system equation)

$$\ddot{\alpha} = \frac{\tau_B - m_F \cdot g \cdot x \cdot c(0) + m_F \cdot g \cdot h \cdot \sin(0) - 2 \cdot m_F \cdot 0 \cdot 0 \cdot 0}{J_B + m_F \cdot h^2 + m_F \cdot 0^2}$$ Thus nonlinear equation will be

$$\ddot{\alpha} = \frac{\tau_B - m_F \cdot g + m_F \cdot g \cdot h \cdot \alpha}{J_{const}}$$ ...Equation (4)

State Space Representation of our system

Generally, following LTI state-space equations is:

$$\underline{\dot{x}} = \underline{A} \cdot \underline{x} + \underline{B} \cdot \underline{u}$$
$$\underline{y} = \underline{C} \cdot \underline{x} + \underline{D} \cdot \underline{u}$$

Vehicle has only one input, **u** = u here is a scalar function and **B** = **b** is a column vector. Because the sample system has only one output, **y** = y here is actually a scalar function, and C = c^T a row vector. For **D** is then a scalar **d**.

$$\underline{\dot{x}} = \underline{A} \cdot \underline{x} + \underline{b} \cdot \underline{u}$$
$$\underline{y} = \underline{c}^T \cdot \underline{x} + d \cdot \underline{u}$$

$$\underbrace{\begin{bmatrix} \ddot{\alpha}(t) \\ \dot{\alpha}(t) \\ \ddot{x}_F(t) \\ \dot{x}_F(t) \end{bmatrix}}_{\dot{\underline{x}}} = \underbrace{\begin{bmatrix} 0 & \frac{m_F g h}{J_{konst}} & 0 & -\frac{m_F g}{J_{konst}} \\ 1 & 0 & 0 & 0 \\ 0 & -g & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\underline{A}} \cdot \underbrace{\begin{bmatrix} \dot{\alpha}(t) \\ \alpha(t) \\ \dot{x}_F(t) \\ x_F(t) \end{bmatrix}}_{\underline{x}} + \underbrace{\begin{bmatrix} \frac{1}{J_{konst}} \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{\underline{B}} \cdot \underline{u}$$

$$\underline{y} = \underbrace{[0 \quad 0 \quad 0 \quad 1]}_{\underline{c}^T} \cdot \underline{x} + \underbrace{0}_{d} \cdot \underline{u}$$

The script file of the above representation using MATLAB is as follows:

```
% Task-02
A = [0,mf*g*h/Jconst,0,-(mf*g)/Jconst;1,0,0,0;0,-g,0,0;0,0,1,0];
B = [1/Jconst;0;0;0];
c = [0,0,0,1];
C = eye(4)
D = [0;0;0;0];
```

MATLAB-Linear Model

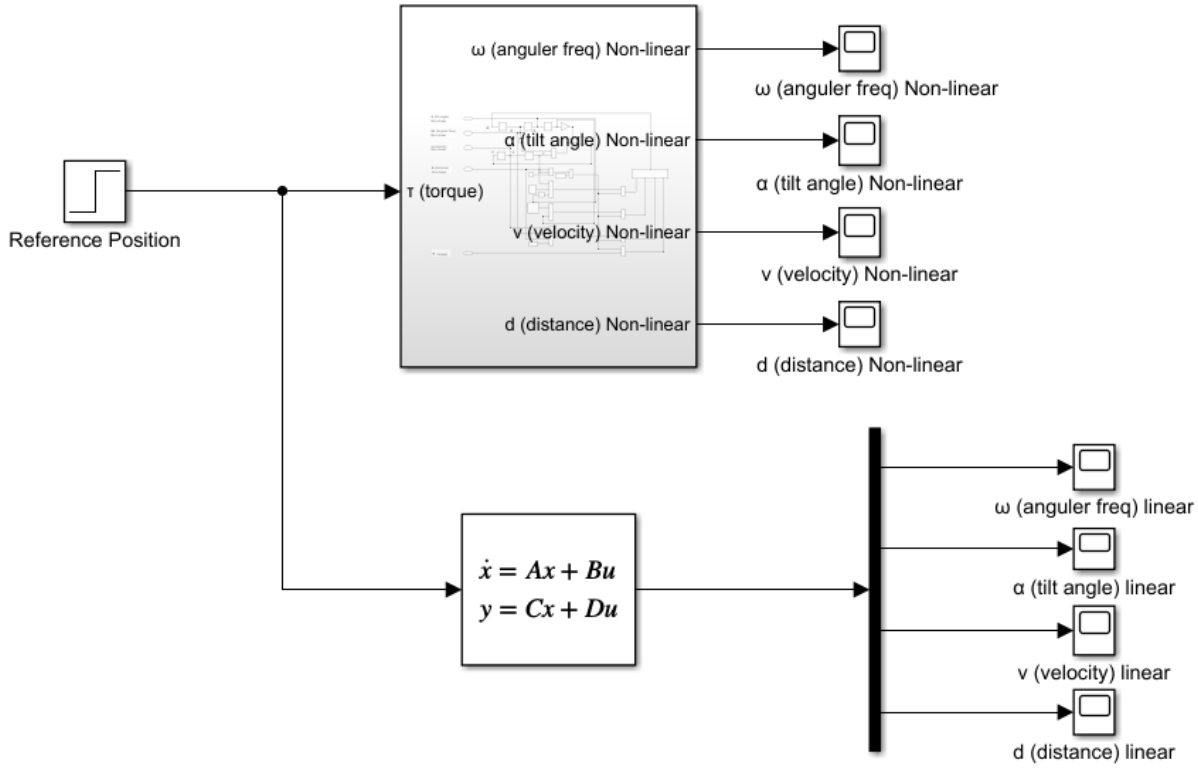


Figure 5. MATLAB See-Saw Liner Model

TASK-3

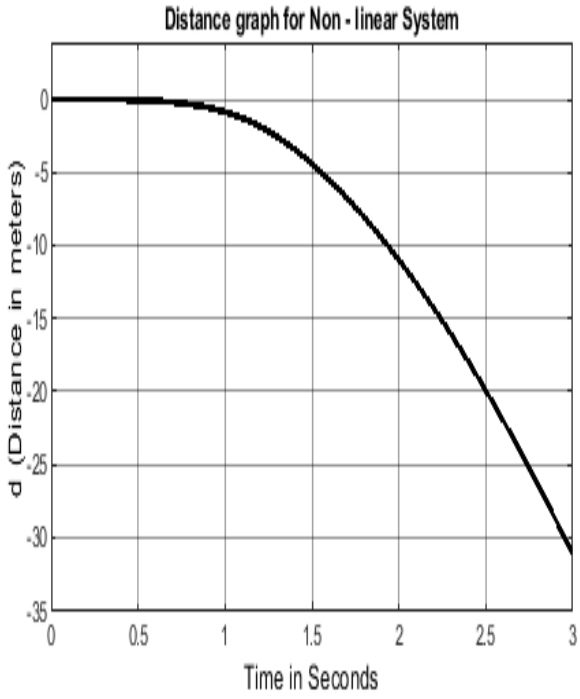
Compare the linear model and the nonlinear model. When does the linear model become imprecise? How far can the working point be away from the initial working point (in state-space), upon which the linear system was derived from?

Comparison between Linear and Non-Linear Model

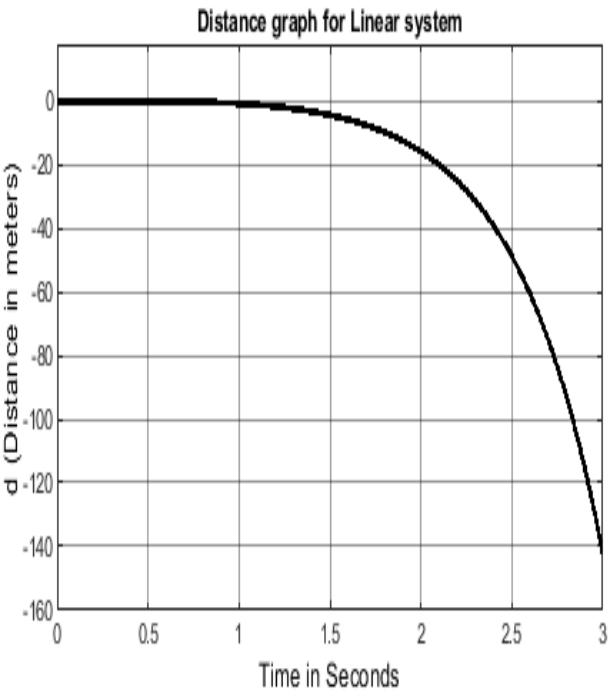
In the last task we linearized the system using state space representation, but we observe that as soon as the driving torque is applied the system becomes unstable again. The possible reason for this is that the input shifts the system away from the operating point. We can observe from our system that small deviation of the angle of the beam makes the system unstable. Hence we can shift working point only to the region in which the linearity is obeyed or the region around the working point in which the non-linear system behaves like linear system.

The Linear and the Non-Linear system can be compared by looking at the graphs of the different state vectors, which are as follows:

Distance of the Cart (x):

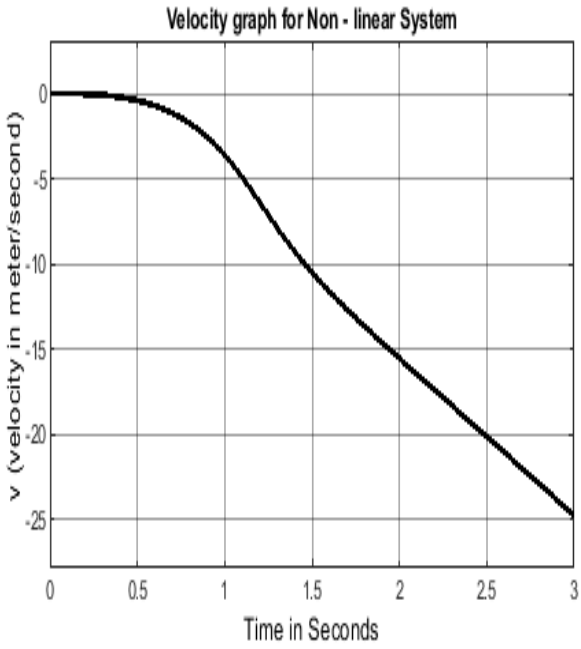


Non-Linear Model

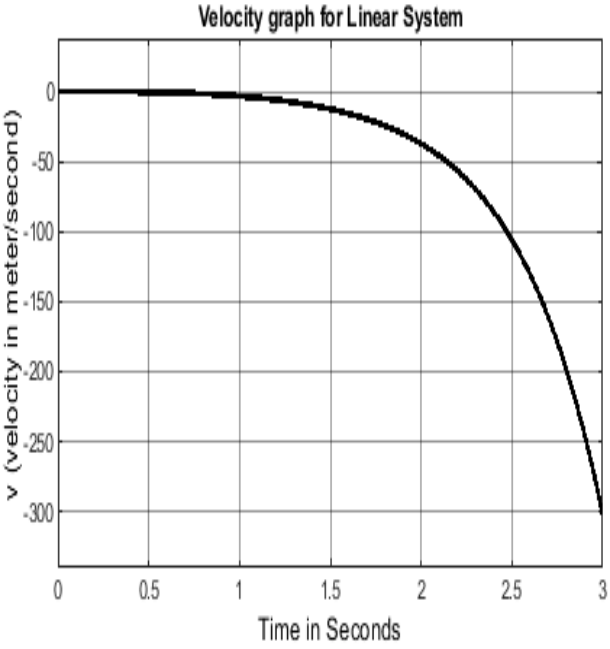


Linear Model

Velocity of the Cart ($\dot{x}$):

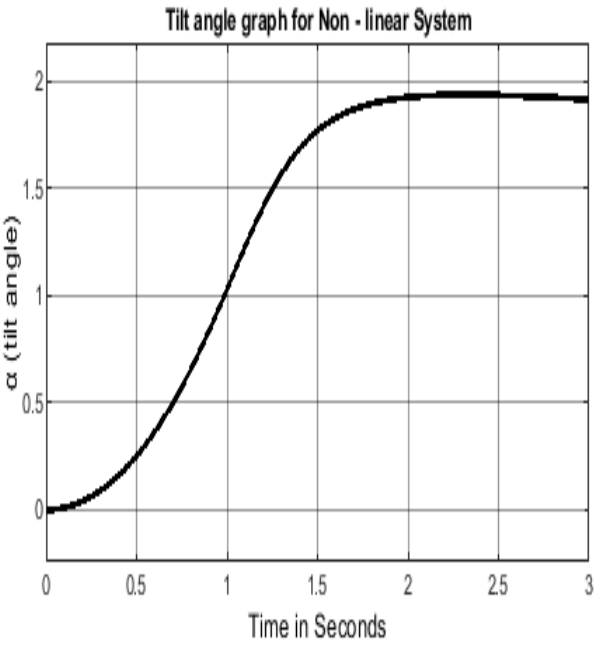


Non-Linear Model

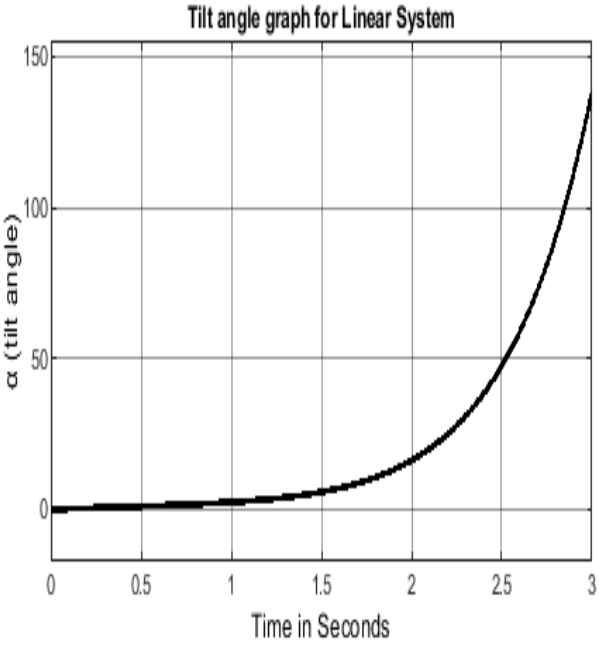


Linear Model

Tilt Angle from the beam (α):

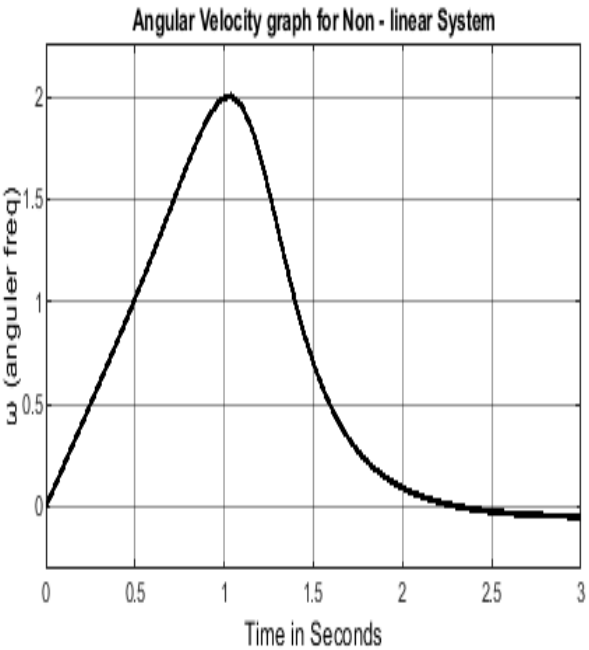


Non-Linear Model

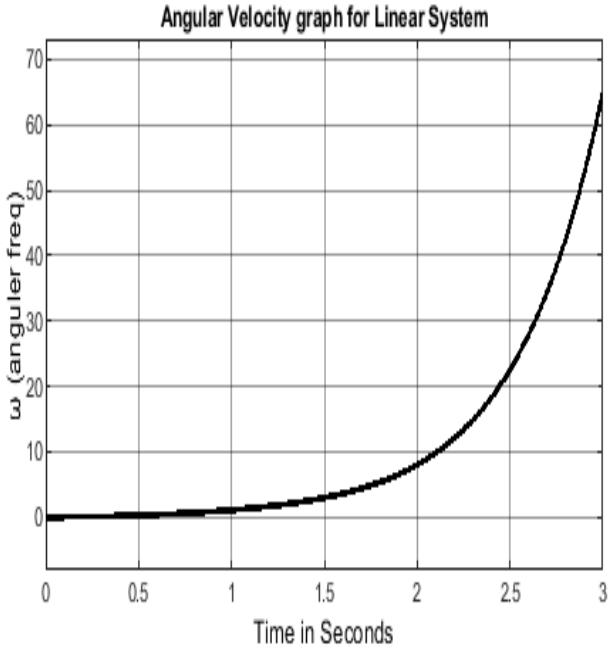


Linear Model

Angular velocity of the beam ($\dot{\alpha}$):



Non-Linear Model



Linear Model

Graph Description: We could see from above graph result that the distance and velocity is almost same for liner and non-linear model. However, alpha and omega are not within the boundary for a linear system. Also, we can notice that none of the attribute is reaching to Reference value as 1 meter for both linear and non-linear system.

TASK-4

Make a state-feedback control based on the linear seesaw system. Place all poles at -1. Calculate state feedback vector k and preamplifier p. Verify the control on the linear seesaw system. Test the linear state-feedback vector k and the preamplifier p on the nonlinear seesaw system. Test it for reference value steps and disturbance torque steps.

State feedback Controller

Since both systems are not stable, our next attempt is to stabilize it. We care for BIBS stability of these systems (Bounded Input, Bounded State). Hence we adopt state feedback method which will shift the poles of system to desired location (left half of S-plane). We design state- feedback gain using Ackermann's formula. With this new additional feature, i.e., State-feedback we analyse the behaviour of the linear and nonlinear system.

The open loop poles of the linear system can be found out by calculating the eigen values of the A matrix. We have used MATLAB to calculate the poles.

The open loop poles are as follows:

```
>> open_loop_poles=eig(A)

open_loop_poles =

   2.1170 + 0.0000i
  -2.1170 + 0.0000i
   0.0000 + 2.0703i
   0.0000 - 2.0703i
```

As we can see that the poles lie on the right hand side of the s-plane which makes the system unstable. We can place the poles according to our choice using the state feedback method by checking that the system is controllable. In this case we are placing the poles at (-1, -1, -1, -1) which

makes the system stable. We design state feedback and suitable pre-amplifier using Ackermann's pole placement approach. Pre- amplifier is needed in order to achieve steady state accuracy.

Following equations and matrices value used for state-feedback control:

Desired poles:

alpha = poly([-1 -1 -1 -1]) = [1 4 6 4 1]

$$\text{eye}(4) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Controllability matrix:

$$\underline{S} = [\underline{B} \quad \underline{A} \cdot \underline{B} \quad \underline{A}^2 \cdot \underline{B} \quad \underline{A}^3 \cdot \underline{B}]$$

Ackermann's formula:

$$K = q \cdot (alpha(5) \cdot eye(4) + alpha(4) \cdot \underline{A} + alpha(3) \cdot \underline{A}^2 + alpha(2) \cdot \underline{A}^3 + \underline{A}^4)$$

Pre-amplifier gain:

$$P = \frac{1}{Ct \cdot (B \cdot K - A)^{-1} \cdot B}$$

MATLAB script is as follows:

```
% Task 4

S = [B A*B A^2*B A^3*B]   % Controllability matrix
R = rank(S)               % Rank of Controllability matrix
Si = inv(S)               % Inverse of Controllability matrix
q = Si(4, 1:4)            % Last Row Of Inverse Controllability matrix
alpha = poly([-1 -1 -1 -1]) % desired poles
Ct = [0 0 0 1]            % output position
K = q * (alpha(5)*eye(4) + alpha(4)*A + alpha(3)*A^2 + alpha(2)*A^3 + A^4)
% Ackermann's formula
P = 1/(Ct*inv((B*K)-A)*B) % pre-amplifier gain
```

MATLAB Simulink State Feedback Model without Disturbance Torque

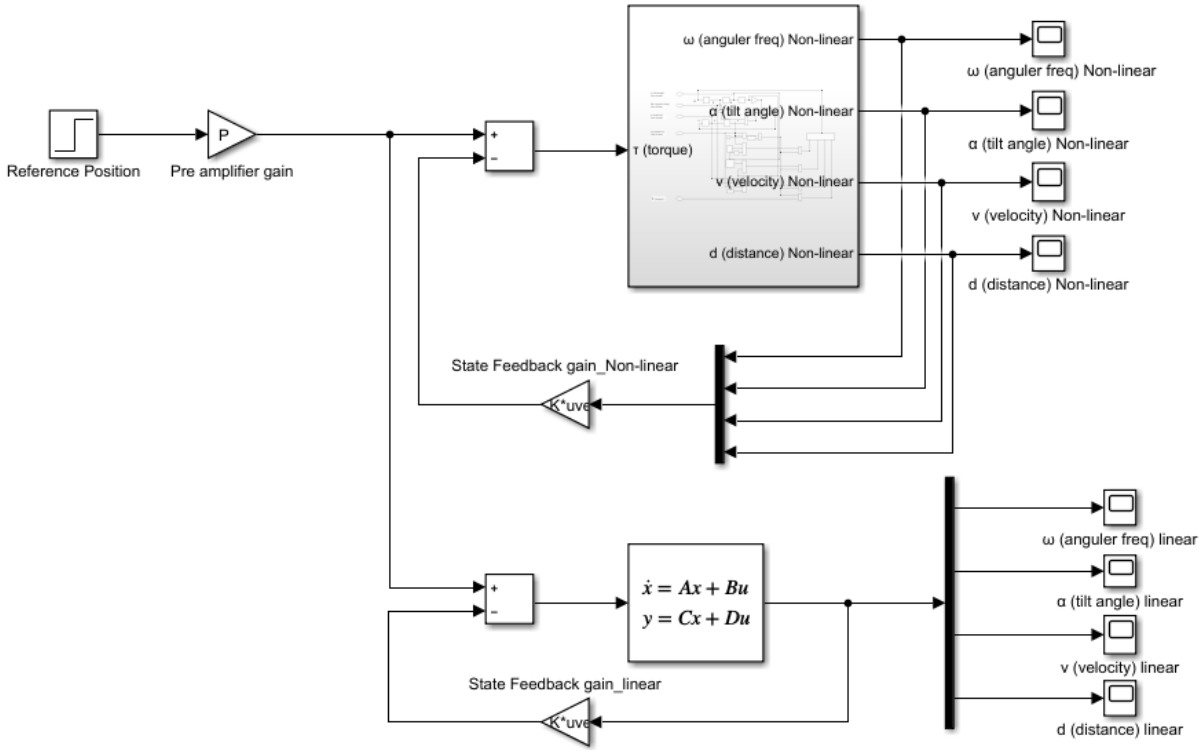
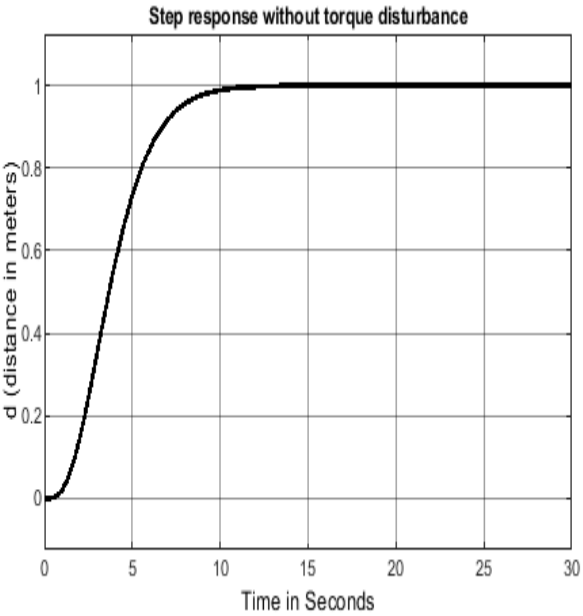
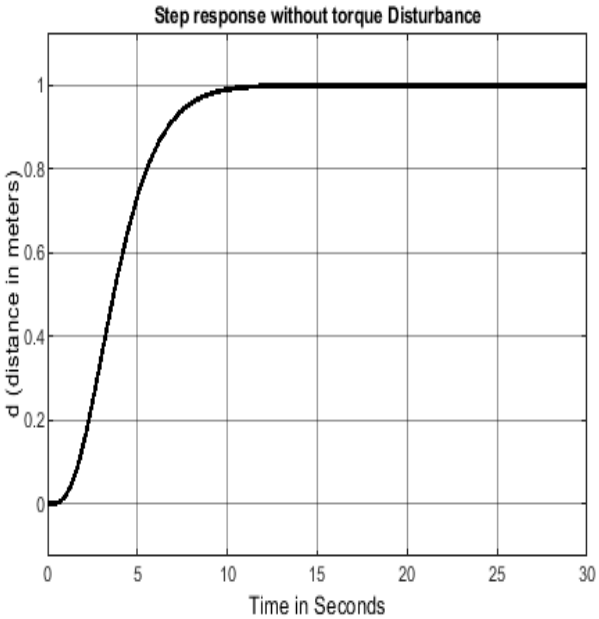


Figure 6. State Feedback Model without Disturbance Torque

Without Disturbance Torque:



Non-Linear Model



Linear Model

MATLAB Simulink State Feedback Model with Disturbance Torque

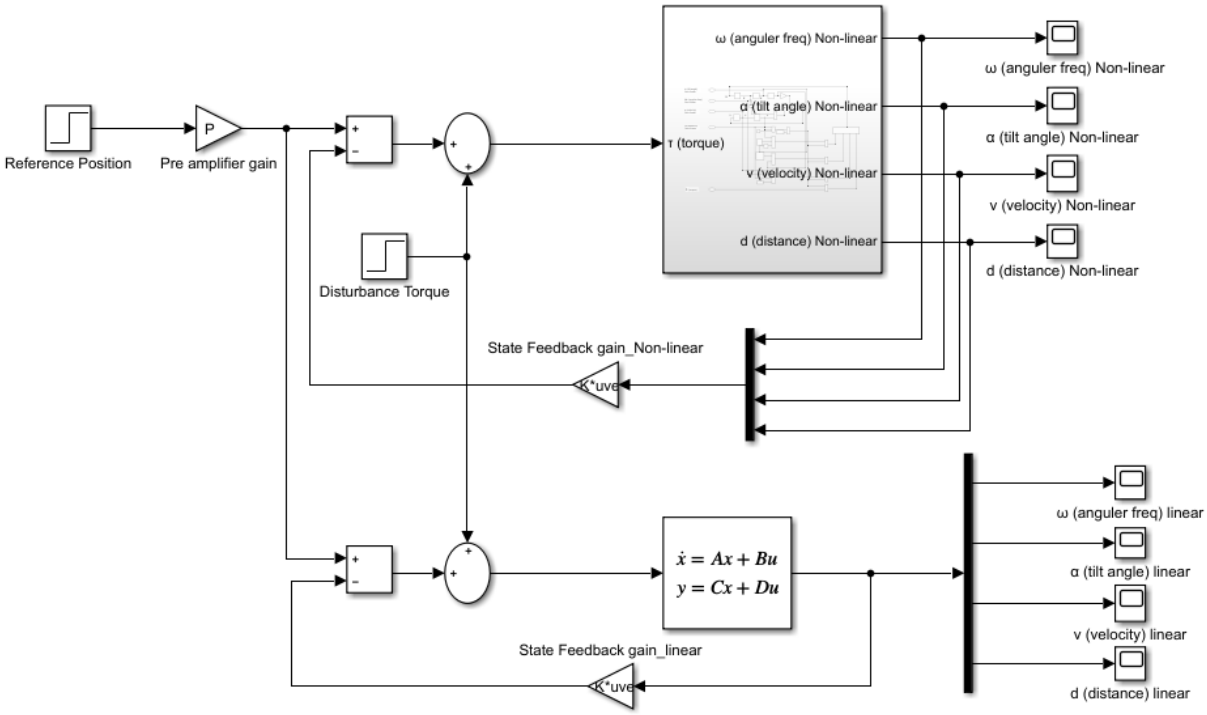
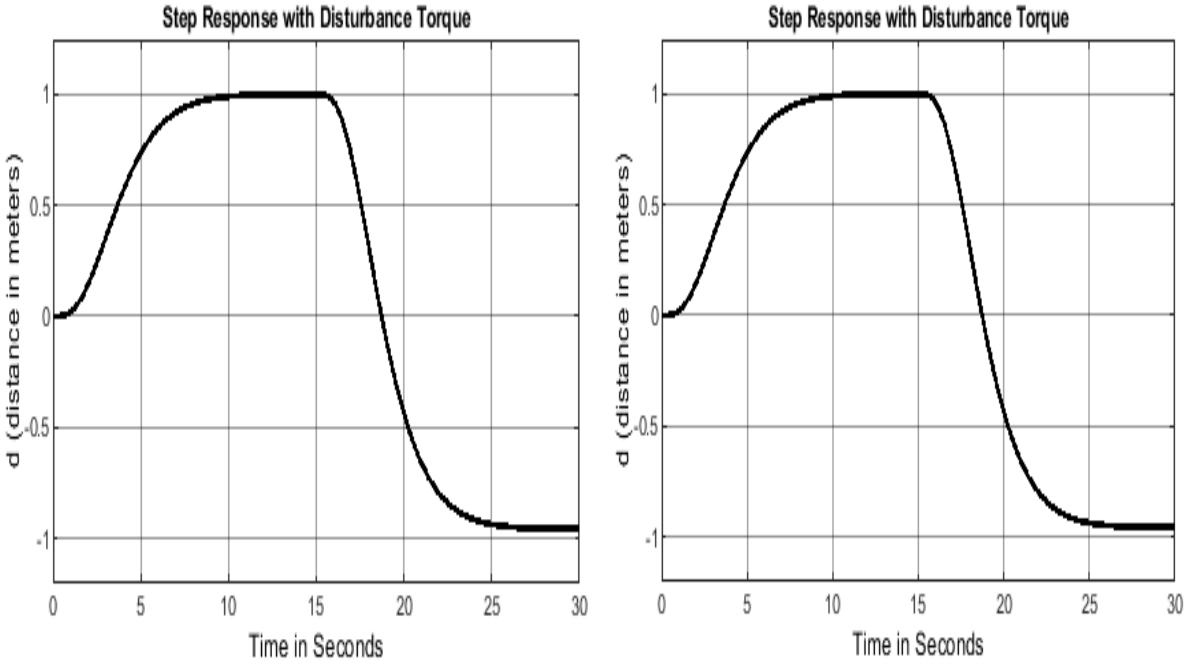


Figure 7. State Feedback Model with Disturbance Torque

With Disturbance Torque:



Non-Linear Model **Linear Model**

Graph Description: We can observe that the system is stable if there is no disturbance torque. However, the feedback fails to make non-linear system stable if the reference input is too large. We have applied the disturbance torque after 15 seconds which we can see from the above graphs. Thus we can conclude state feedback control on non-linear model is not effective for large manipulating range.

TASK-5

Make a PI-state-feedback control based on the linear seesaw system. Place all poles at -1. Calculate the state feedback vector k based on the augmented system (5th order). Verify the control on the linear seesaw system. Test the linear PI-state-feedback control for reference value steps and disturbance torque steps.

PI Controller:

A Proportional-Integral controller (PI controller) is a control loop feedback mechanism used in industrial control systems. This helps in calculating an error value e (t), which is the difference between the desired set-point and a measured process variable and applies for a correction based on the proportional and integral controller.

In the previous task, we designed the state-feedback control which made see-saw system stable. Because of the disturbance torque we have a distortion in the system then we need to make a comparison between actual value and control value to determine control error. For this reason we need Proportional Controller (P- Controller).

If we have a distortion and if this distortion is acting at the output of the system then P- Controller is enough to make sure the error vanishes. But in cases where distortion acts at the input of the system then P- Controller fails to reduce the error hence we need additional integrator. This type of controller is called PI- Controller.

Due to the additional integrator introduced through PI Controller, we get an additional state. Hence to account for this additional state, we convert Seesaw system (4th order system) into 5th order system as shown below.

Following equations and matrices value used for PI-state-feedback control:

Desired poles:

alpha = poly([-1 -1 -1 -1 -1]) = [1 5 10 10 5 1]

$$\text{eye}(5) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Controllability matrix:

$$\underline{S} = \left[\underline{B_{pi}} \quad \underline{B_{pi}} \cdot \underline{A_{pi}} \quad \underline{B_{pi}} \cdot \underline{A_{pi}}^2 \quad \underline{B_{pi}} \cdot \underline{A_{pi}}^3 \quad \underline{B_{pi}} \cdot \underline{A_{pi}}^4\right]$$

Ackermann's formula:

$$K_{pi} = qI \cdot (alpha(6) \cdot eye(5) + alpha(5) \cdot \underline{A_{pi}} + alpha(4) \cdot \underline{A_{pi}}^2 + alpha(3) \cdot \underline{A_{pi}}^3 + alpha(2) \cdot \underline{A_{pi}}^4 + alpha(1) \cdot \underline{A_{pi}}^5$$

Pre-amplifier gain:

Pi = $-Kpi(5)$

MATLAB script is as follows:

```
% Task-05
% State feedback with PI
Api=[A [0;0;0;0]; -Ct 0]
Bpi=[B;0]
Ci=[0,0,0,1,0]
Cpi=eye(5)
Dpi=[0;0;0;0;0]
SI=[Bpi Api*Bpi Api^2*Bpi Api^3*Bpi Api^4*Bpi] %Controllability Matrix
RI=rank(SI) %Rank Of Inverse Controllability Matrix
SI_i=inv(SI) %Inverse Controllability Matrix
qI=SI_i(5, 1:5) % Las Row of Inverse Controllability matrix
alpha = poly([ -1 -1 -1 -1 -1]) % Desired poles
Kpi = qI*(alpha(6)*eye(5) + alpha(5)*Api + alpha(4)*Api^2 + alpha(3)*Api^3 + alpha(2)*Api^4 + alpha(1)*Api^5) % ackermann's formula
Ki = Kpi(1:4)-(P*Ct); %Value of state feedback
```

```
Pi = -Kpi(5) % Gain of Integral
```

In this case we are only considering only one state which is the position (x), so it is a SISO system and the pre-amplifier gain remains the same.

MATLAB Simulink Model:

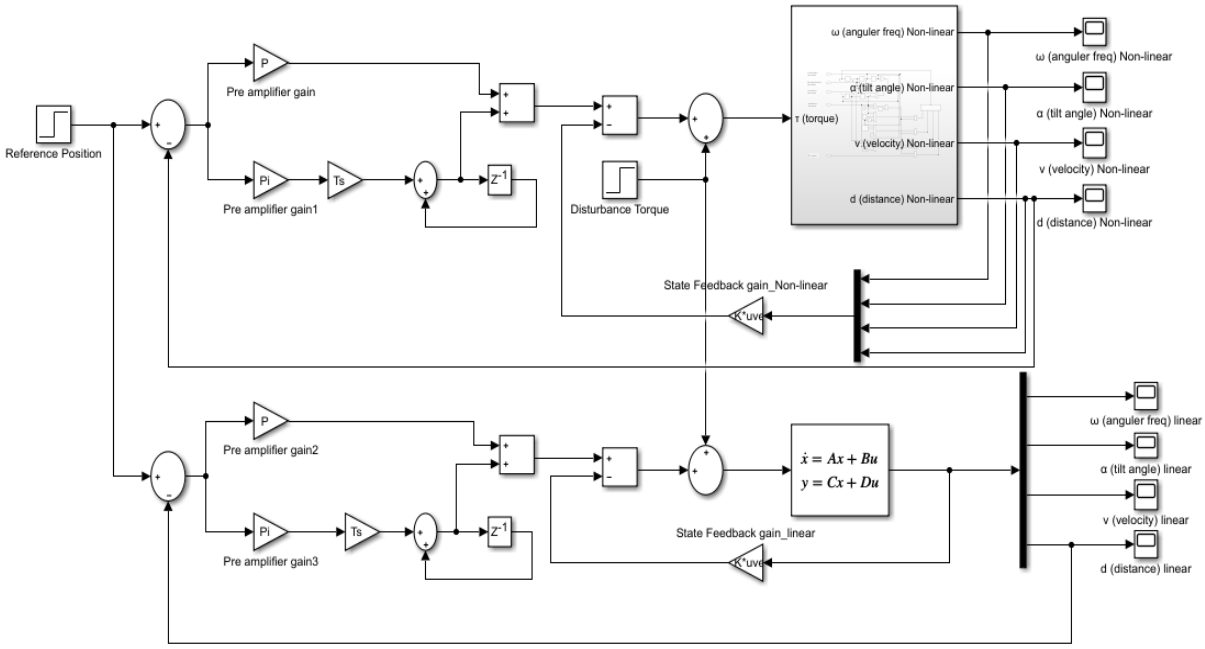
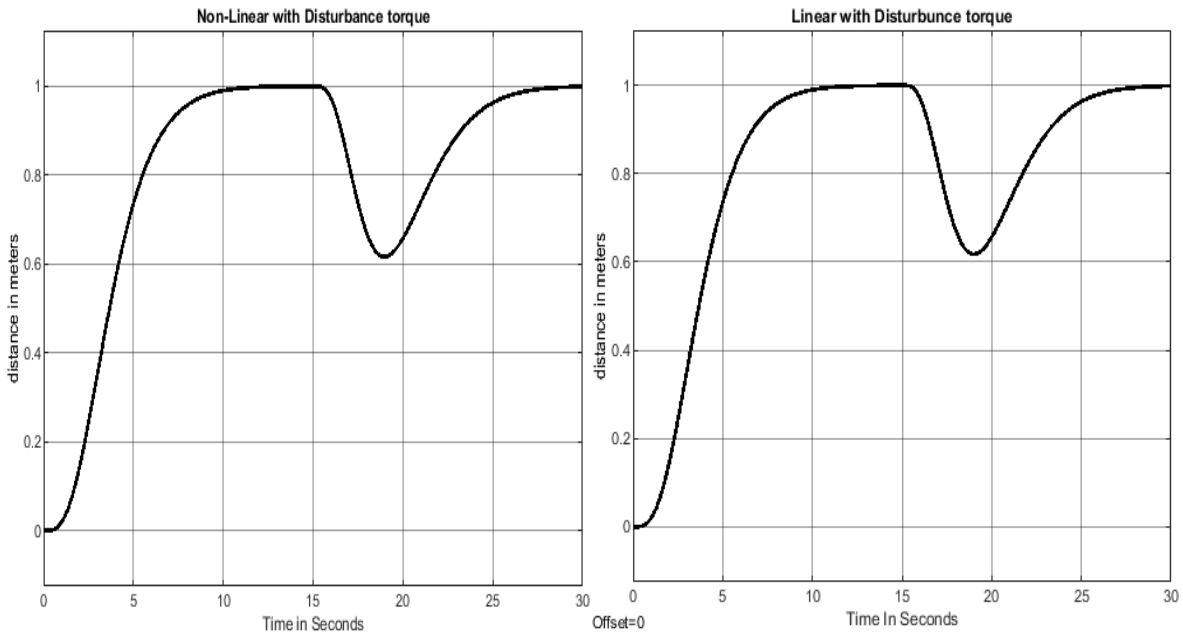


Figure 8. PI Controller with Disturbance Torque (continuous)

The PI controller is applied to both the linear and non-linear system and the behaviour is shown below:



Non-Linear Model **Linear Model**

Graph Description: linear and non-linear model graph shows that when disturbance torque applied as an input to the system after 15 seconds it is not reaching to desire value of 1 and it becomes un-stable. However, with help of PI controller method an error feedback has been calculated and applied to the input. Eventually it has made system stable after 10 seconds of disturbance torque.

TASK-6

Make the PI-part time discrete with a sample rate 8 kHz and test the linear PI-state-feedback on the nonlinear seesaw system.

In previous task we designed PI Controller in continuous time domain. For many applications we require to implement this PI Controller on a micro-controller/processor. Hence we need to design this in time- discrete system. It's important to note, time-discrete system is stable when all the poles are within the unit circle. Hence it's important to convert all time domain poles to z-domain poles. The sample rate given for our system is 8 kHz and thus we need to calculate the sample time (Ts) and provide this sample time to our system.

Sample Time:

$T_s = \frac{1}{8000}$ seconds;

Desired poles:

alpha = poly([-1 -1 -1 -1 -1]) = [1 5 10 10 5 1]

Controllability matrix:

$\underline{S} = [\underline{B}_{pi} \quad \underline{B}_{pi} \cdot \underline{A}_{pi} \quad \underline{B}_{pi} \cdot \underline{A}_{pi}^{2} \quad \underline{B}_{pi} \cdot \underline{A}_{pi}^{3} \quad \underline{B}_{pi} \cdot \underline{A}_{pi}^{4}]$

Ackermann's formula:

$K_{pi} = qI \cdot (alpha(6) \cdot eye(5) + alpha(5) \cdot \underline{A}_{pi} + alpha(4) \cdot \underline{A}_{pi}^{2} + alpha(3) \cdot \underline{A}_{pi}^{3} + alpha(2) \cdot \underline{A}_{pi}^{4} + alpha(1) \cdot \underline{A}_{pi}^{5}$

State Feedback value:

$\mathrm{Kpd} = \mathrm{Ktd}(1{:}4) - (\mathrm{P} \cdot \mathrm{Ct})$

MATLAB script is as follows:

```
% Task-06
% Discrete PI system

Ts=125e-6 % sampling time
sys1=c2d(ss(A,B,C,D),Ts) % continuous system
[AD,BD,CD,DD]=ssdata(sys1) % accessing data of discrete system
sysd=c2d(ss(Api,Bpi,Ci,0),Ts) % discrete system
[Adi,Bdi,cdi,Ddi]=ssdata(sysd) % accessing data of discrete system
Sd=[Bdi Adi*Bdi Adi^2*Bdi Adi^3*Bdi Adi^4*Bdi] % controllability matrix
Rd=rank(Sd) % rank of the controllability matrix
Sd_i=inv(Sd) % inverse of controllability matrix
qdi=Sd_i(5, 1:5) % last row of the inverse controllability matrix
tf_pi=tf([1],poly([-1 -1 -1 -1 -1 ])) % transfer function for desired poles
tfd=c2d(tf_pi,Ts) % converting transfer function to discrete
[zeros_d,poles_d]=tfdata(tfd,'v')
Ktd=qdi*(poles_d(6)*eye(5)+poles_d(5)*Adi+poles_d(4)*Adi^2+
poles_d(3)*Adi^3+poles_d(2)*Adi^4+ poles_d(1)*Adi^5) %Ackermann's formula
pi_d=-Ktd(5) % discrete integrator gain
Kpd=Ktd(1:4)-(P*Ct) %Value of state feedback
```

MATLAB Simulink Model:

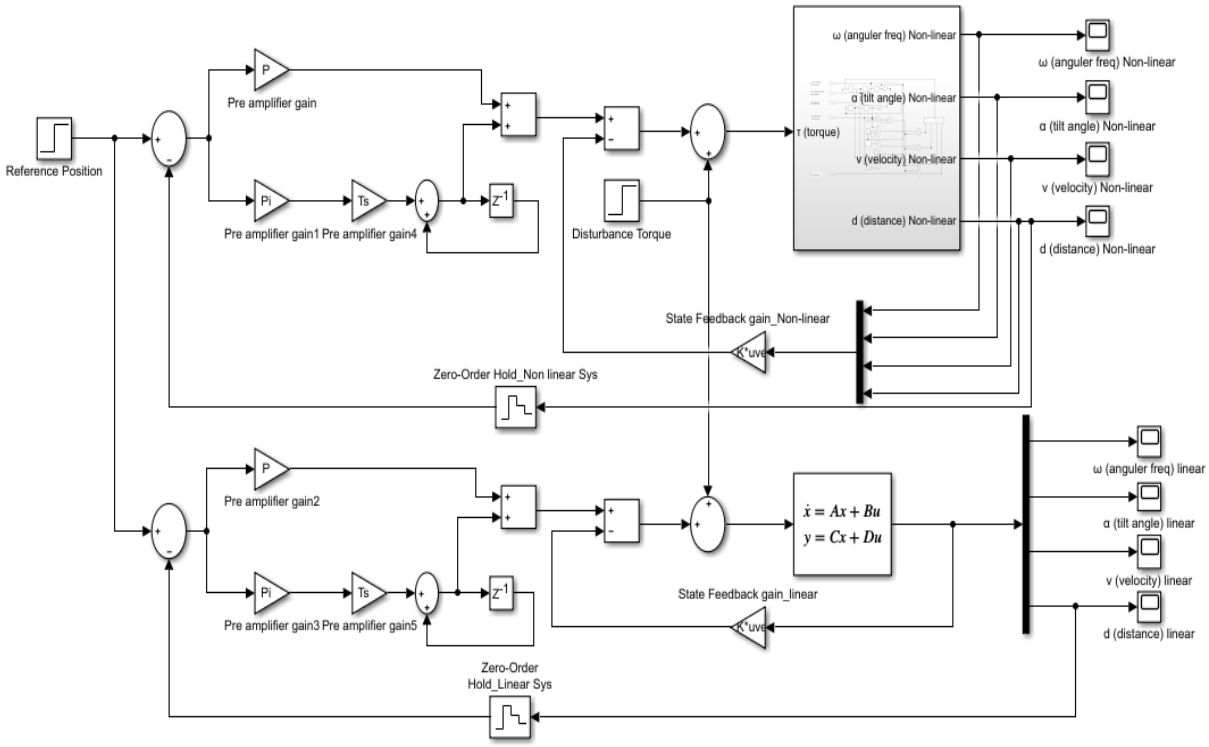
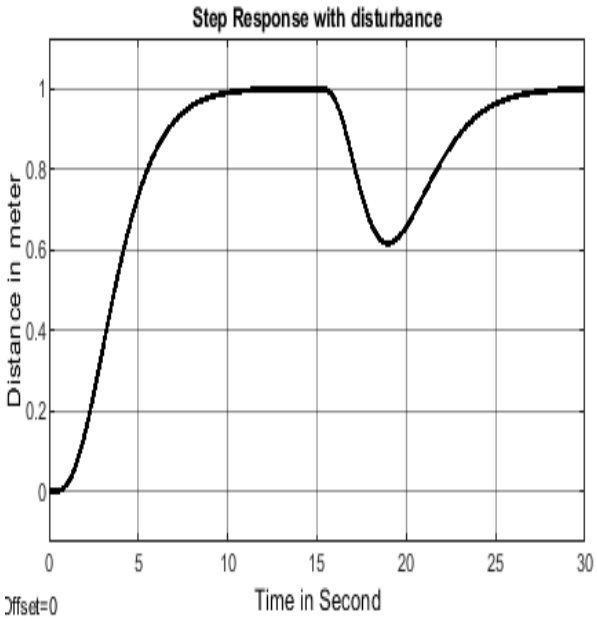
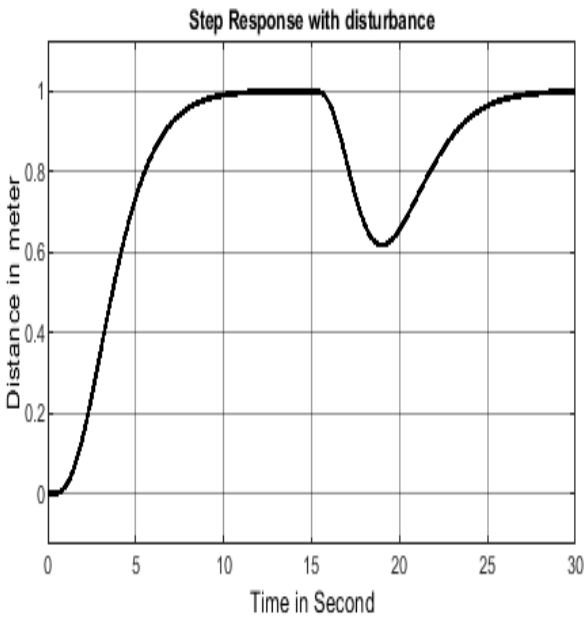


Figure 9. PI Controller with Time discrete



Non-Linear Model



Linear Model

TASK-7

Make a linear full-state observer based on the linear seesaw system. Place all poles at -3. Calculate the error feedback vector h. Validate the observer on the linear seesaw system for different initial conditions. Make the parallel model time discrete (zero-order-hold) with a sample rate of 8 kHz. Test the linear full-state observer on the nonlinear seesaw system with different initial conditions.

Full State Observer Design for Continuous Time System:

An observer is a dynamic system that is used to estimate the state of a system or some of the states of a system. A full-state observer is used to estimate all the states of the system. The observer can be designed as either a continuous-time system or a discrete-time system. The characteristics are the same, and the design processes are at least very similar and in some cases identical. These notes will focus primarily on continuous-time observers.

In many practical situations it's difficult to measure all internal states. That's where observer comes into picture. Observer will reconstruct all internal states from measured output. Obvious requirement to design an Observer is, given system must be fully Observable (i.e. Observability

matrix should have full Rank). We build a parallel system which is similar to the original system and feed same input to it. However the outputs of both the systems do not match because initial conditions do not match. This means the estimated states will not be accurate resulting in an Observation error (AKA state error). Hence we need to design manipulating variable which makes this Observation error zero. Using Ackermann's formula we design manipulating variable 'h1' which controls the reduction of this observation error. In principle, Observer must be must faster than corresponding Controller. We have designed our controller by placing all poles at -1. In order to make our Observer faster, we design 'h' by placing all poles at -3.

Following equations and matrix value used for Full State Observer for time continuous system:

Desired poles:
alpha_o = poly([-3 -3 -3 -3]) = [1 12 54 108 81]

Observability matrix:

$$M_o = \begin{bmatrix} Ct \\ Ct \cdot A \\ Ct \cdot A^2 \\ Ct \cdot A^3 \end{bmatrix}$$

Ackermann's formula:

$$\text{Kalpha\_o} = \left(alpha\_o(5) \cdot eye(4) + alpha\_o(4) \cdot \underline{A} + alpha\_o(3) \cdot \underline{A}^2 + alpha\_o(2) \cdot \underline{A}^3 + \underline{A}^4\right)$$

Error feedback vector:

$$\text{efed\_v} = \text{Kalpha\_o} \cdot \text{M}_o{}^{-1}(1{:}\,4,4)$$

MATLAB script is as follows:

```
% Task-07
% Observer based on the linear seesaw system.

x1=[0 0 0 1];  % initial condition
Ct1=C(4,1:4)
Mo=[Ct1; Ct1*A; Ct1*(A^2); Ct1*(A^3)] % Observability matrix
```

```
Mo_inv=inv(Mo)  % inverse of Observability matrix
Qt_o=Mo_inv(1:4,4)  % last column of inverse of Observability matrix
alpha_o=poly([-3 -3 -3 -3])  % polynomial
Kalpha_o=(alpha_o(5)*eye(4) + alpha_o(4)*A + alpha_o(3)*(A^2)+ alpha_o(2)*(A^3) +
(A^4)) %Ackermann's formula

efed_v=((Kalpha_o)*(Qt_o))  % error feedback vector

% Make the parallel model time discrete (zero-order-hold)

c_TF_o=tf([1],poly([-3 -3 -3 -3])) % transfer function for desired poles
d_TF_o=c2d(c_TF_o,Ts)  % converting transfer function to discrete
[num_o,den_o]=tfdata(d_TF_o,'v')

Kt_o=acker(Ad.',Cd(4,1:4).',(roots(den_o)).')

Kt_o_new=Kt_o.'
```

MATLAB Simulink Model:

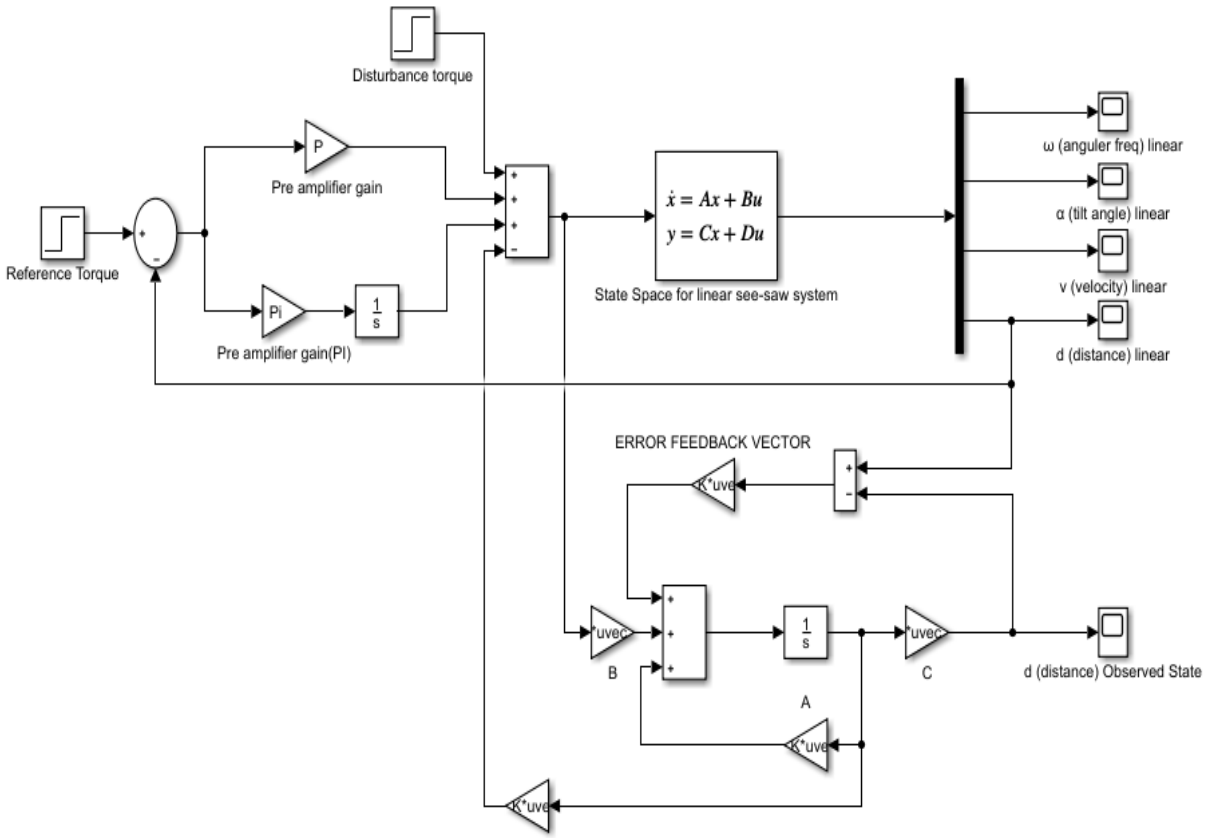


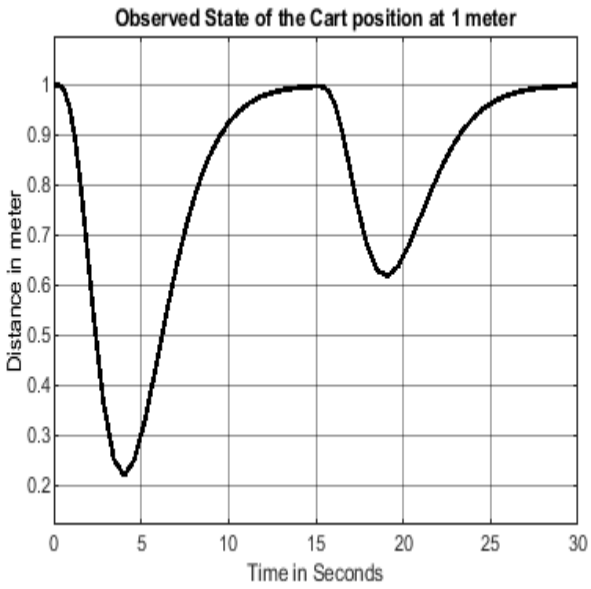
Figure 10. Observer with linear system in time continuous

Observation Graphs:

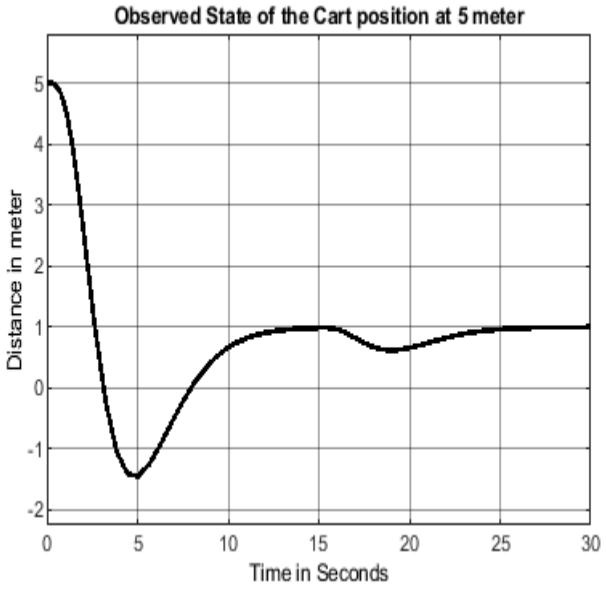
Case 1: Initial condition = 1, Disturbance (step time = 15 / final value = 1)

Case 2: Initial condition = 5, Disturbance (step time = 15 / final value = 1)

Case 1



Case 2



Graph Description: Due to the difference in initial conditions w.r.t system, the observer does some training for a short duration, and then eventually it estimates the exact state. This training period can be shortened by placing the poles more to the left half s-plane; however the manipulating variable shoots to a very high value.

Full State Observer Design for Time Discrete:

Similar to discrete controller, observers too are part of micro-controller, hence we need to discretize observer as well.
We have decided to discretize the observer with the sampling rate of 8 kHz. For designing it in time discrete domain, the desired step response poles considered to be in the left half s-plane needs to be mapped to points within unit circle in z-plane in order for the time-discrete system to be stable. Zero order hold converts continuous signal to discrete with sampling rate of 8 kHz.

Below equations and matrix value used for Full State Observer for time discrete system:

Initial Condition:

$$x2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Desired poles:

poly([-3 -3 -3 -3]) = [1 12 54 108 81]

```
% Discrete State Observer based on linear seesaw system

x2=[0;0;0;1]  % initial condition
sys=ss(A,B,C,D)
sysd=c2d(sys,Ts)  % Bi-linear transformation
[Ad_obs,Bd_obs,Cd_obs,Dd_obs]=ssdata(sysd)
cTd_obs=Cd_obs(4,1:4)
desys=tf([1],poly([-3 -3 -3 -3]))  % desired polynomials of step response transfer function
desysd=c2d(desys,Ts)  % Converting the tf into Discrete
[num_poly,den_poly]=tfdata(desysd,'v')
poly_space = roots(den_poly)
k_obs_d = acker(Ad_obs.',cTd_obs.',poly_space.') % Ackermann's formula
h_ons_d = k_obs_d.'
```

MATLAB Simulink Model:

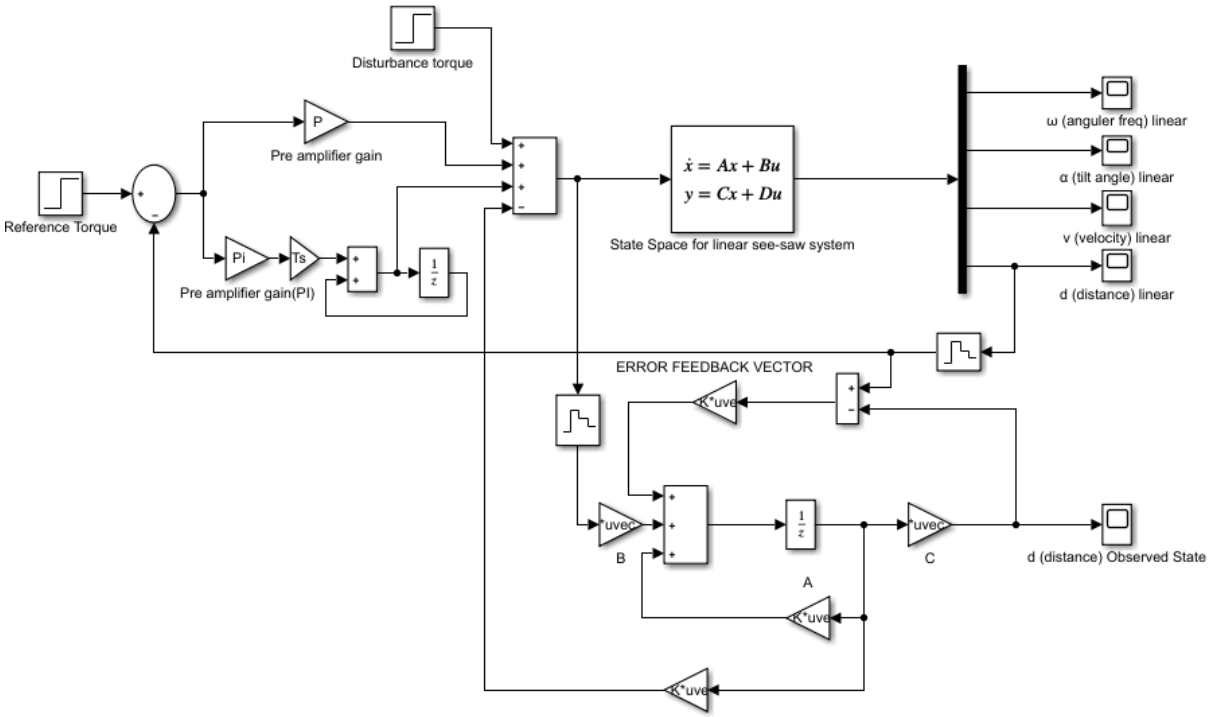


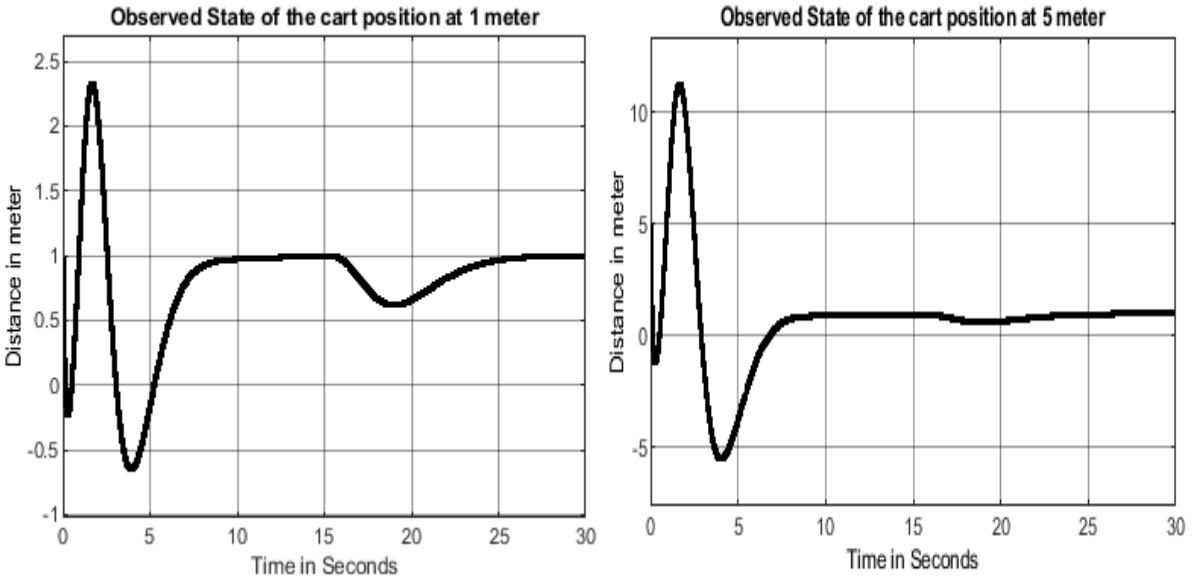
Figure 11. Observer with linear system in time discrete

Observation Graphs:

Case 1: Initial condition = 1, Disturbance (step time = 15 / final value = 1)

Case 2: Initial condition = 5, Disturbance (step time = 15 / final value = 1)

Case 1 Case 2



MATLAB Simulink Model:

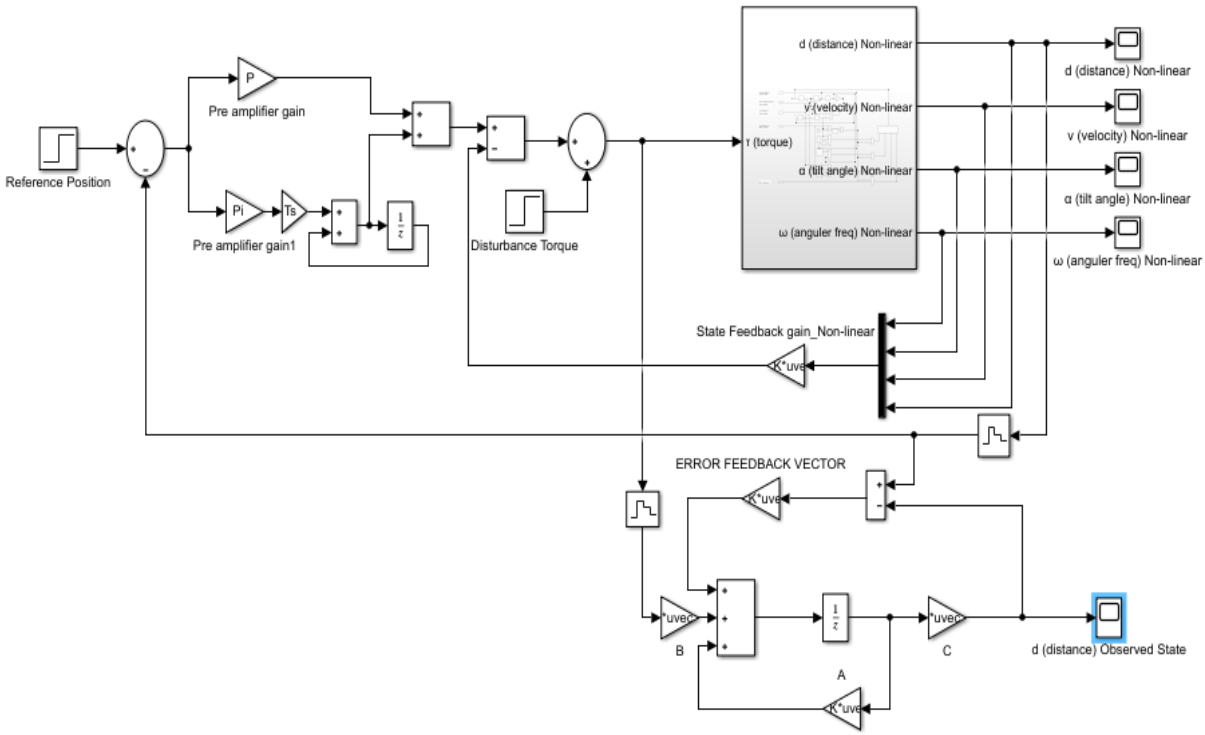


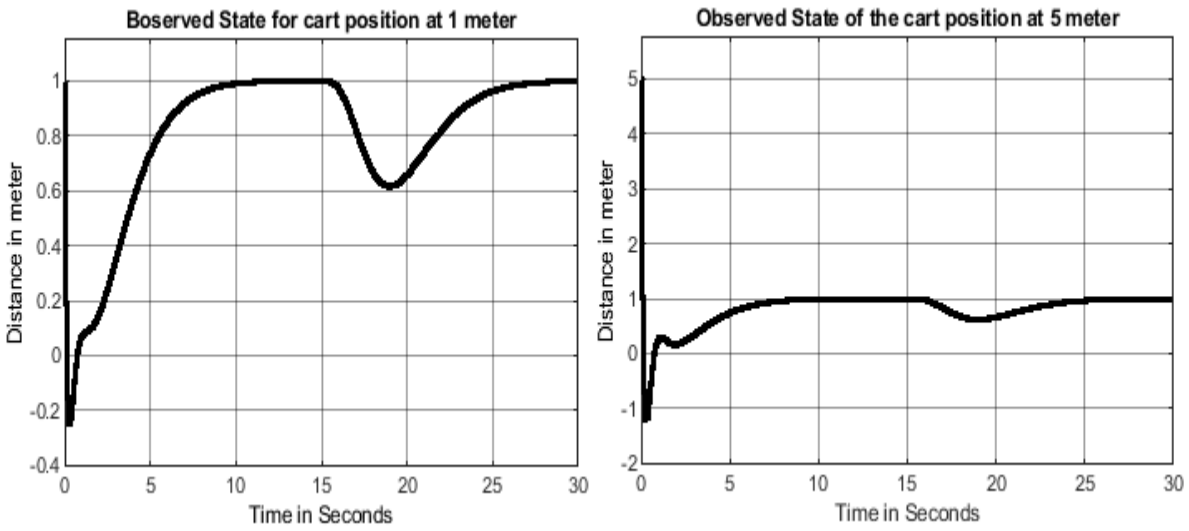
Figure 11. Discrete observer with non-linear system

Observation Graphs:

Case 1: Initial condition = 1, Disturbance (step time = 15 / final value = 1)

Case 2: Initial condition = 5, Disturbance (step time = 15 / final value = 1)

Case 1 Case 2



Graph Description: Due to the difference in initial conditions w.r.t system, the observer does some training for a short duration in non-linear system as well, and then eventually it estimates the exact state. Thus, we can now use observed states as input to PI controller, instead of reading values from sensor.

TASK-8

Use the observed states from 7 for the controls 4 – 6 and test them on the linear and nonlinear system for different initial conditions and reference value steps and disturbance torque steps.

State Feedback Observer using PI Controller:

In practical, all the parts are coded in C language and hence we need discrete observer with discrete PI control. But in this, we have done for both in continuous and discrete observer model just for comparison.

MATLAB Simulink Model:

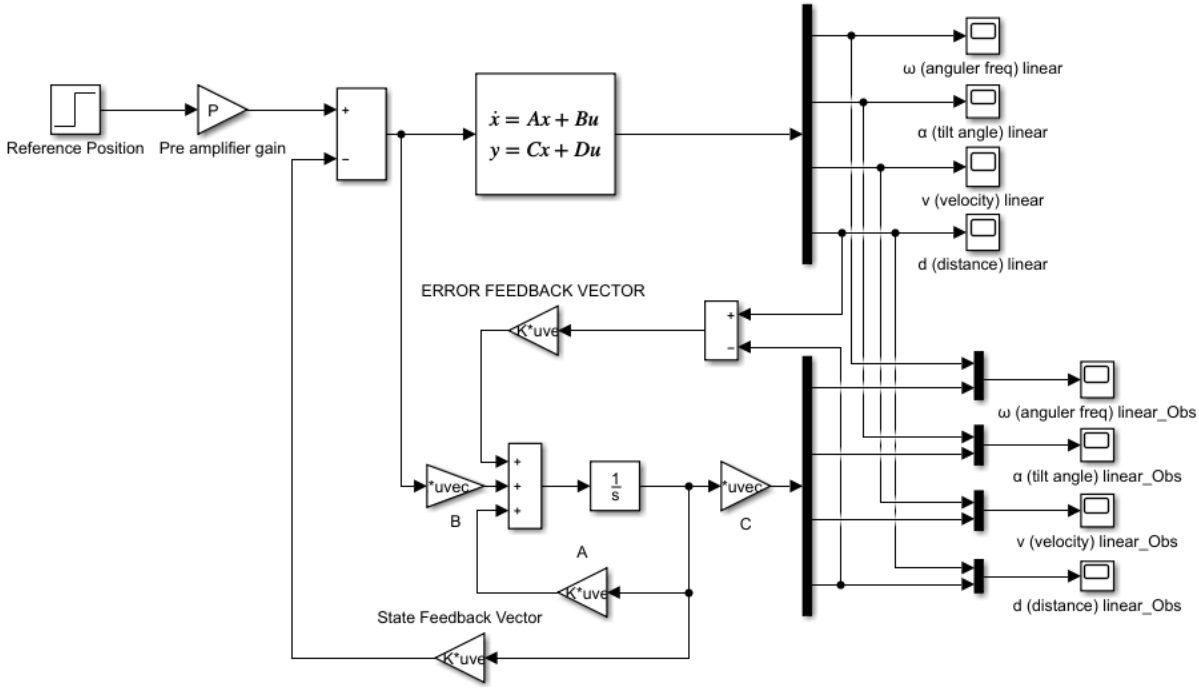
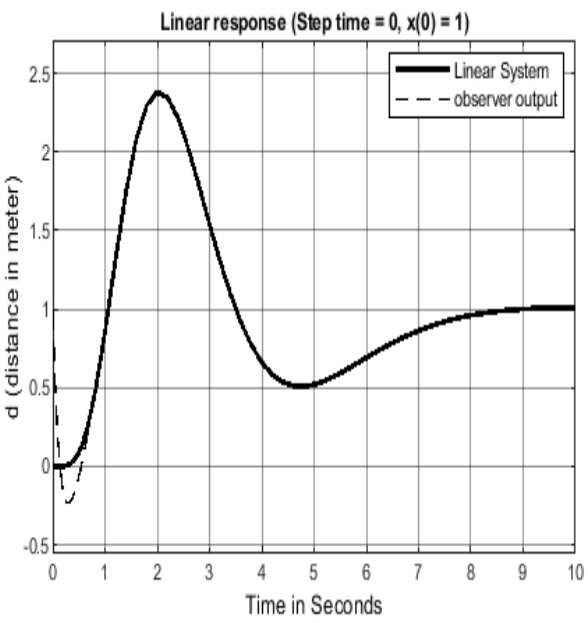


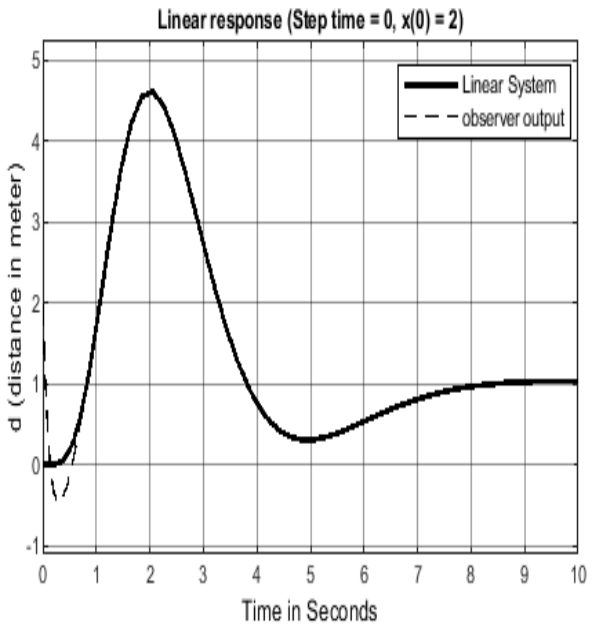
Figure 12. State feedback observer with linear system

Observation Graphs:

Case 1



Case 2



The observer does some training for a short duration in non-linear system as well, and then eventually it estimates the exact state. And its feedback tends to state can be seen in the graph below with two conditions.

MATLAB Simulink Model:

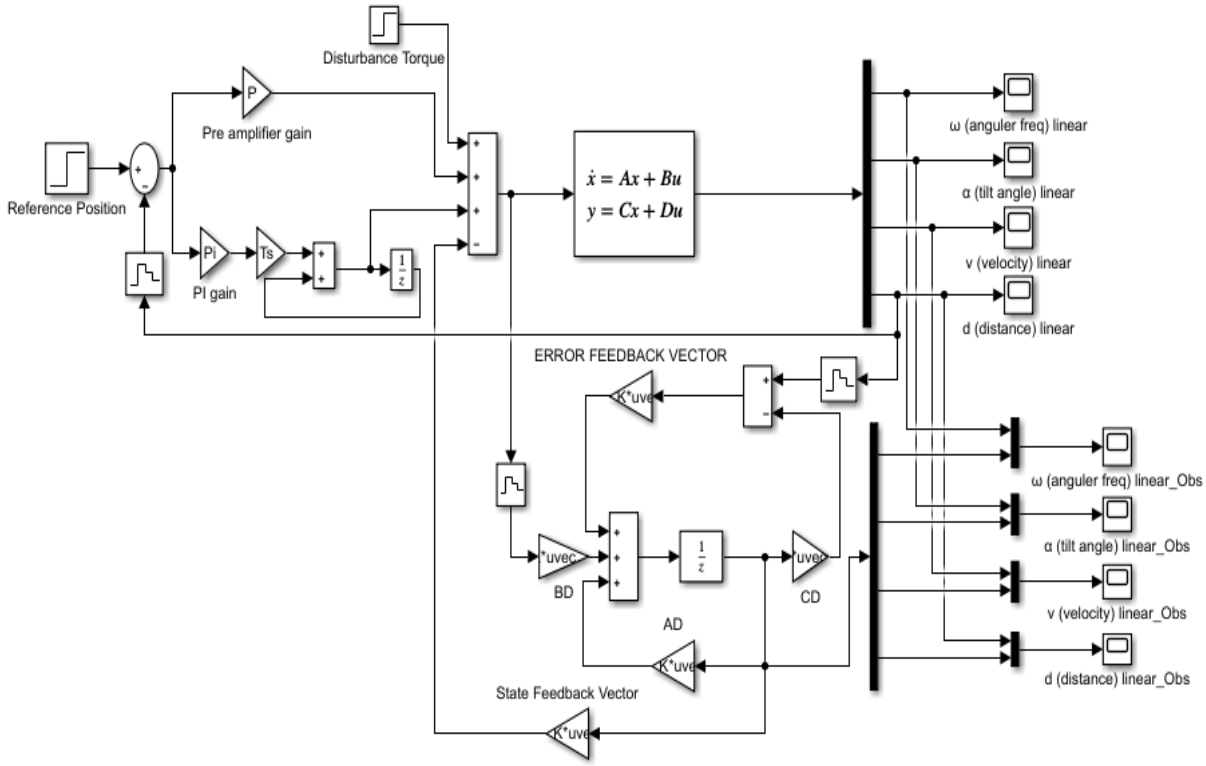
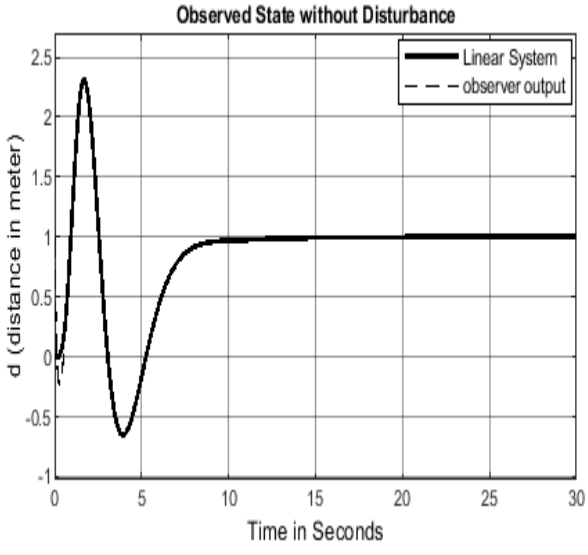
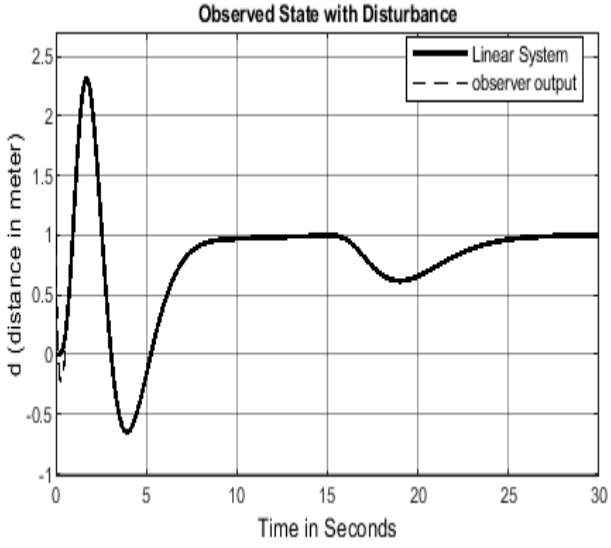


Figure 13. PI Controller linear system discrete observer

Observation Graphs:



Without Disturbance



With Disturbance (step time = 15)

MATLAB Simulink Model:

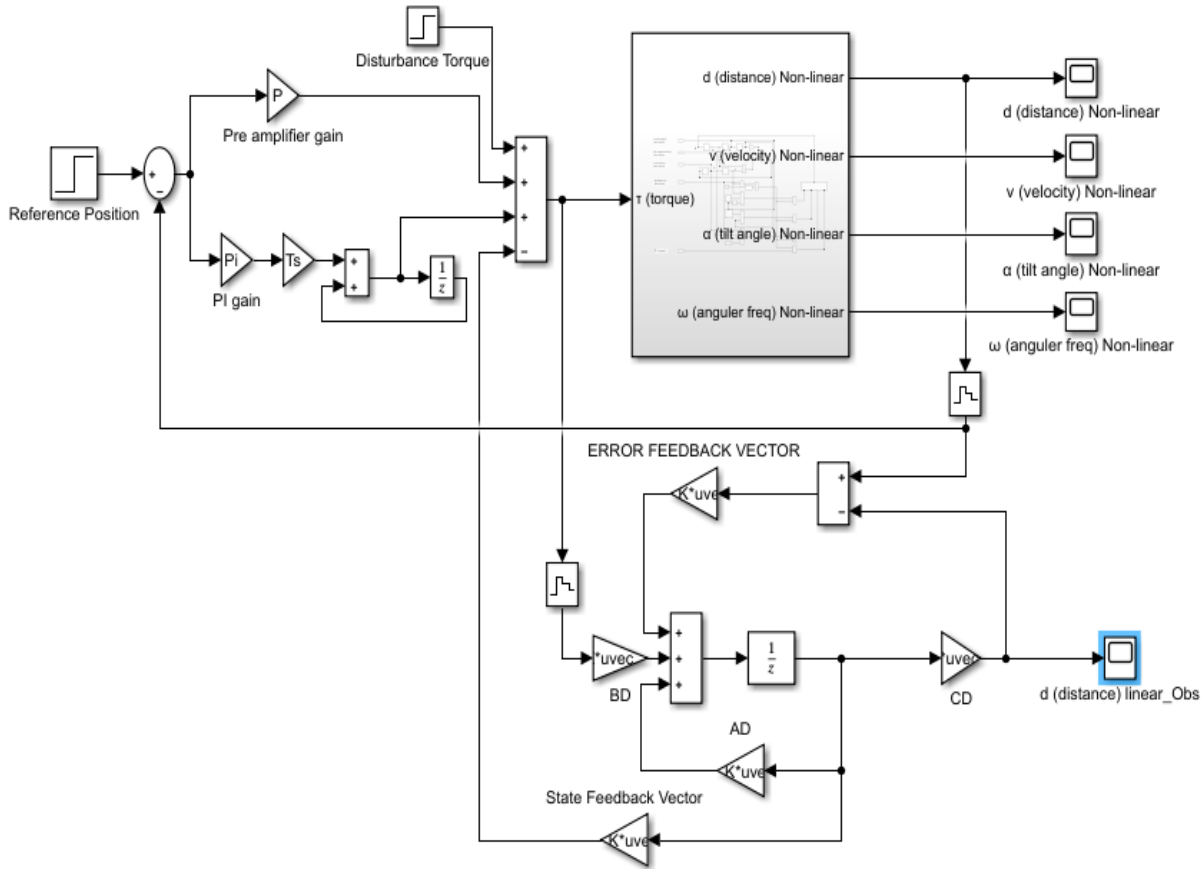
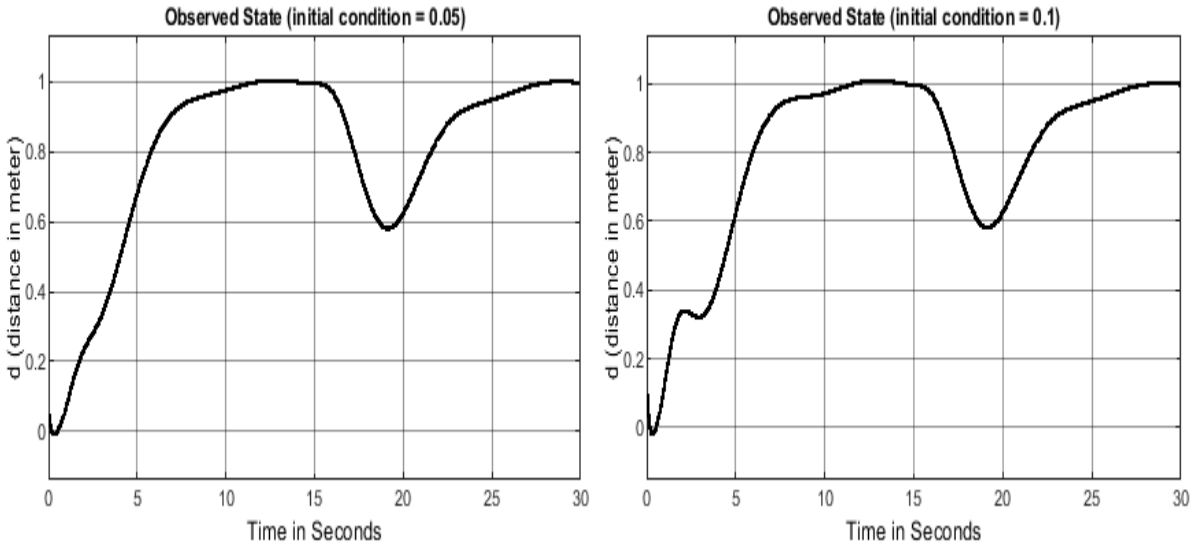


Figure 14. PI Controller non-linear system discrete observer

Observation Graphs:



Initial Condition=0.05 Initial Condition=0.1

Hence we can conclude that, Observed states/ Observer can be combined with Non-linear system only when there is no much difference in initial

conditions. In other words, it is applicable only in the small region around our working point.

CONCLUSION

By performing the entire lab exercise we come to a conclusion that all the controls that we learnt works much as expected on the linear model of the original system. But when we apply these control techniques to the nonlinear counterpart, they work well only when the operating conditions are near around our working point (the point around which the system is linearized). If we deviate large, then we see some undesired behaviour of the original system. We developed both discrete and continuous observer and also we have seen the error that we get due to the initial state difference between the observer and the real system, after training the output of both system becomes the same.