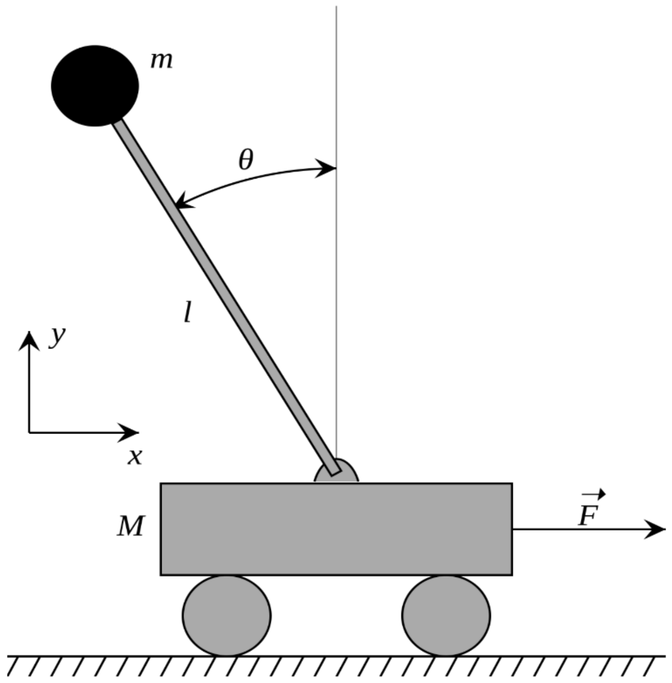
University of Science and Technology in Zewail City

2022/2023

LINEAR INVERTED PENDULUM

REE 310 - Final Report



UNDER SUPERVISION

DR. MOHAMED LOTFI EID SHALTOUT

Presented By

Aya Atya 202000247
Beshoy Bahaa 202001123
Marwa Elbadry 202001052
Phelopater Ramsis 202001171
Shrouk Emara 202001004
Yara Abdullah 202000629

Table of Contents

Abstract……………………………………………………………………………..3

Introduction…………………………………………………………………………4

Differential equations derivation..........................……………………………………………..5

System parameters...7

System modeling and transfer function 8

Investigating system stability 11

Design objectives 13

PID controller design using the Root Locus technique 14

Designed PID controller in SIMULINK 17

Conclusion 18

Appendix...18

References...19

Abstract

The project attempts to analyze and control a linear time-invariant single-input single-output system using the fundamental control system concepts and apply using MATLAB/SIMULINK. The inverted pendulum has been chosen as the system for this project, and it needs to be controlled by an electric actuator. To meet particular performance requirements, the project requires the mathematical modeling and simulation of the system in MATLAB/SIMULINK, followed by the design and implementation of a control system. The project's outcomes include competence and the capability of designing and implementing a control system for a physical system, as well as a comprehensive understanding of control systems theory and its practical applications.

Introduction

In this project, we use the MATLAB/SIMULINK environment to control inverted pendulum system. The system consists of an inverted pendulum, with the motion of the pendulum being controlled by an electric actuator. The system's goal is to keep the pendulum in place and stabilize it against external disturbances.

To model the system, we started with the fundamental physical laws and relations and derived the differential equations that describe the system dynamics. The driving force, friction force, and weight are all forces that are included in the system. We analyzed the forces affecting the cart and pendulum and the differential equations that characterize the motion of the system. Using the resulting differential equations, we described the system's parameters, such as the cart's mass, the pendulum's mass, the friction coefficient, and the moment of inertia, and then constructed a model of the system in SIMULINK.

We checked the stability of the system's open-loop transfer function, which was created and verified manually using SIMULINK's Linearization tool, which creates a linear approximation of a nonlinear system by calculating its open-loop transfer function based on small perturbations around an operating point. The analysis showed that the system is unstable as its natural response grows infinitely with time.

To achieve the desired closed-loop response, we specified control design objectives such as settling time and gain are control design objectives used to create a PID controller using the Root Locus approach, which is a method for designing feedback systems by plotting the locations of a system's closed-loop poles as a function of a feedback gain parameter. We assumed values for gain, settling time, and damping ratio to create a PID controller with starting zeroes and poles. The controller was iteratively enhanced by adding more zeroes and poles until the desired closed-loop response was achieved. We achieved so using SIMULINK's building blocks which are a set of pre-built components used to model and simulate dynamic systems in a visual programming environment.

Differential equations derivation

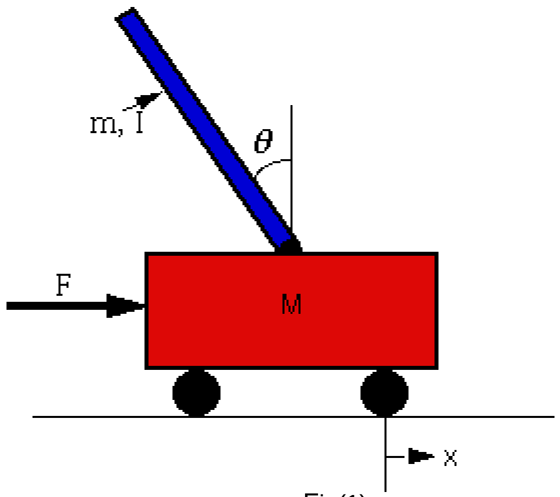
-Forces included in the system are as follows:

-Force acting on the system(driving force)

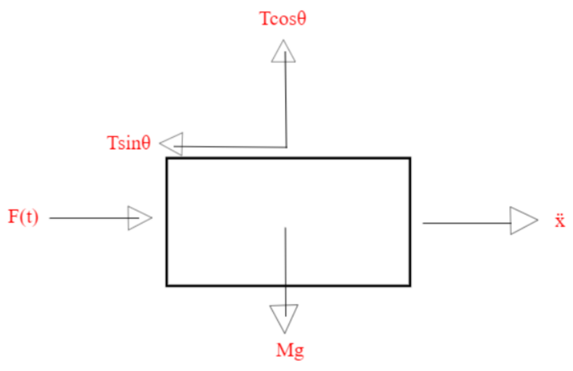
-Friction force

-Weight

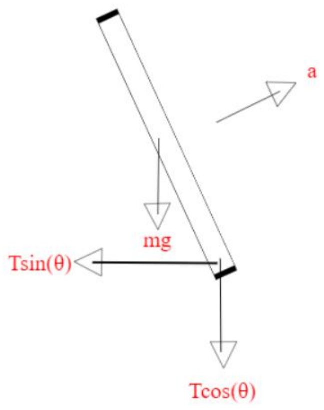
The following photo represents forces analysis of both the cart and the pendulum:.



Fig(1)



Fig(2)



Fig(3)

By summing the forces in the x-axis for the cart:

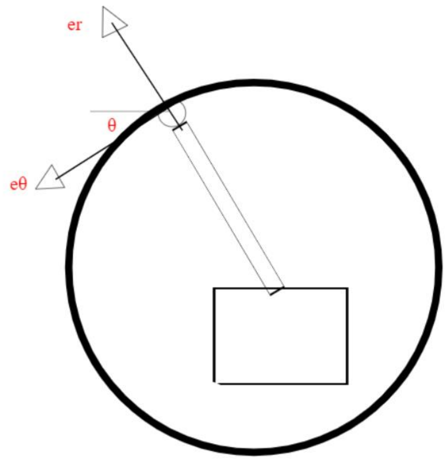
$F - \mu\dot{x} - M\ddot{x} = T\sin(\theta) \rightarrow (1)$

By summing the forces in x-axis for the pendulum:

$T\sin\theta = ma_x \rightarrow (2)$

By summing the forces in the y-axis for the pendulum:

$-T\cos\theta - mg = ma_y \rightarrow (3)$



Fig(4)

We know that:

$$a_{pendulum} = a_{cart} + a_{pendulum/cart}$$
$$= \ddot{x}\, i + [l\ddot{\theta}\, e_\theta - l(\dot{\theta})^2 e_r]$$

Therefore

$$a_x = \ddot{x} + l\ddot{\theta}(-\cos\theta) - l(\dot{\theta})^2(-\sin\theta) = \ddot{x} - l\ddot{\theta}\cos\theta + l(\dot{\theta})^2\sin\theta$$
$$a_y = l\ddot{\theta}(-\sin\theta) - l(\dot{\theta})^2\cos\theta = -l\ddot{\theta}\sin\theta - l(\dot{\theta})^2\cos\theta$$

Now, by substituting in (2) and (3)

$T\sin\theta = m(\ddot{x} - l\ddot{\theta}\cos\theta + l(\dot{\theta})^2\sin\theta) \rightarrow (2)'$

$-T\cos\theta = mg + m(-l\ddot{\theta}\sin\theta - l(\dot{\theta})^2\cos\theta) \rightarrow (3)'$

By taking $\sum M_0 = I\ddot{\theta}$ for the pendulum:

$l\,T\sin\theta\cos\theta - l\cos\theta\sin\theta = I\ddot{\theta} \rightarrow (4)$

Now substitute with (2)' and (3)' in (4)

$$l\,m(\ddot{x} - l\ddot{\theta}\cos\theta + l(\dot{\theta})^2\sin\theta)\cos\theta + l(mg + m(-l\ddot{\theta}\sin\theta - l(\dot{\theta})^2\cos\theta))\sin\theta = I\ddot{\theta}$$

Simplify to get the first equation of motion:

$l\,m\ddot{x}\cos\theta - ml^2\ddot{\theta}(\cos^2\theta + \sin^2\theta) + mg\sin\theta = I\ddot{\theta}$

we know that: $\sin^2\theta + \cos^2\theta = 1$

$l\,m\ddot{x}\cos\theta - ml^2\ddot{\theta} + mg\sin\theta = I\ddot{\theta} \rightarrow$ ***differential eq 1***

To get the second equation of motion, substitute with (2)' in (1) and let $f(t) = u(t)$:

$u(t) - \mu\dot{x} - M\ddot{x} = m(\ddot{x} - l\ddot{\theta}\cos\theta + l(\dot{\theta})^2\sin\theta)$

$u(t) = \mu\dot{x} + (M+m)\ddot{x} - ml\ddot{\theta}\cos\theta + ml(\dot{\theta})^2\sin\theta \rightarrow$ ***differential eq 2***

By using small angle approximation in both equations

$\sin\theta = \theta$, $\cos\theta = 1$, $(\dot{\theta})^2 = 0$, and $\sin^2\theta = \theta^2 = 0$

First differential equation: $u(t) = \mu\dot{x} + (M+m)\ddot{x} - ml\ddot{\theta}$

Second differential equation: $l\,m\ddot{x} + mgl\,\theta = (I + ml^2)\ddot{\theta}$

System parameters

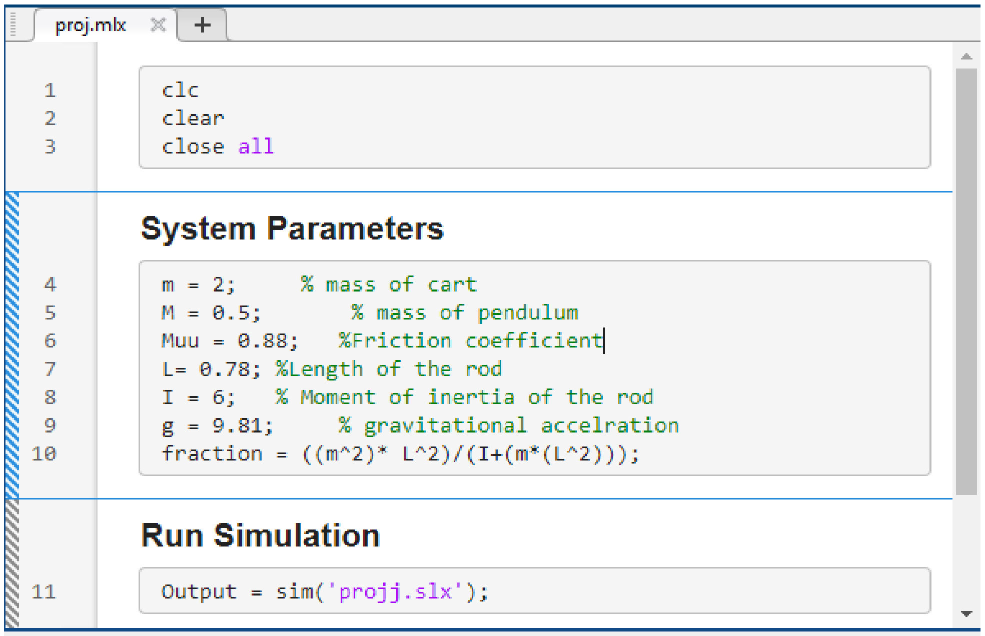
Parameter Name	Unit	Assumed values
Cart mass	kg	2
Pendulum mass	kg	0.5
Applied force	N	Function of time
Gravitational acceleration	m^2/s	9.81
Angle of deviation	θ	Function of time
Friction coefficient	-	0.88
Length of the rod	m	0.78
Moment of inertia of the rod	$kg \cdot m^2$	6

Table(1)

System modeling and transfer function:

To make it easier to model in Simulink, we made an equation named fraction that is used for around 3 times in modeling.

We coded the equations and assumptions in MATLAB and connected its script with Simulink.



```
clc
clear
close all
```

System Parameters

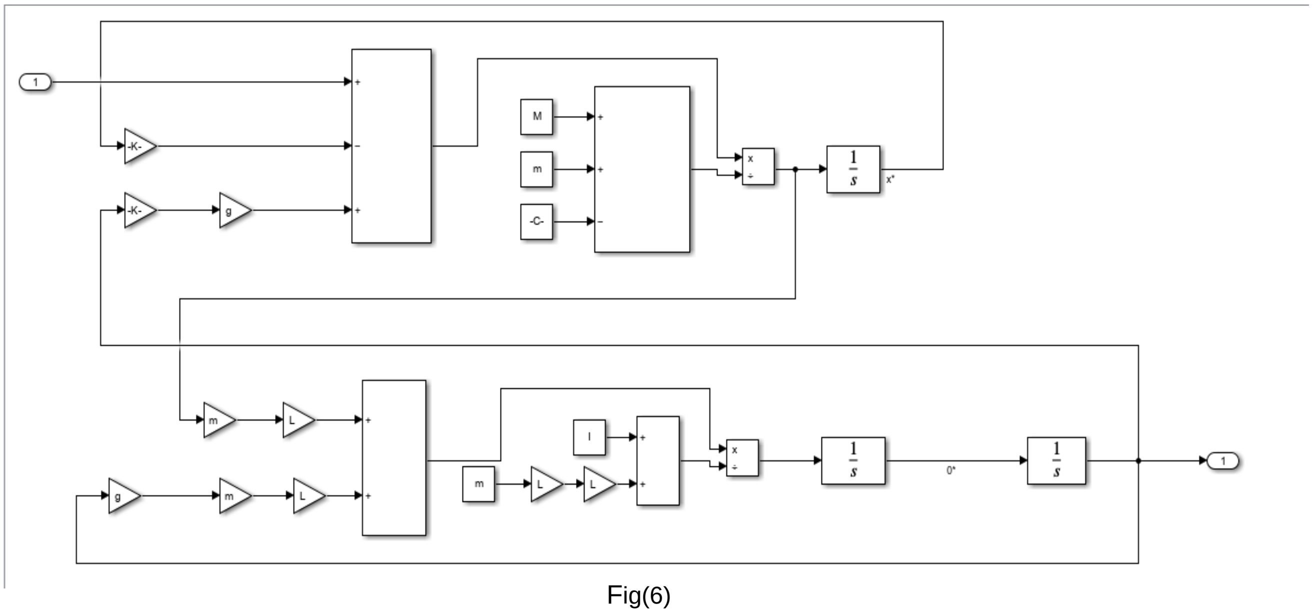
```
m = 2;     % mass of cart
M = 0.5;       % mass of pendulum
Muu = 0.88;   %Friction coefficient
L= 0.78; %Length of the rod
I = 6;   % Moment of inertia of the rod
g = 9.81;     % gravitational accelration
fraction = ((m^2)* L^2)/(I+(m*(L^2)));
```

Run Simulation

```
Output = sim('projj.slx');
```

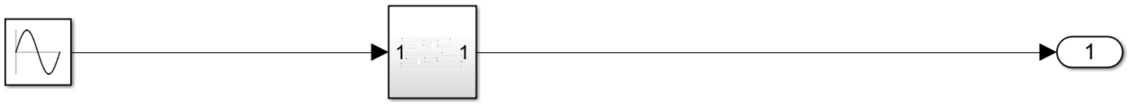
Fig(5)

We modeled the 2 differential equations as follows:



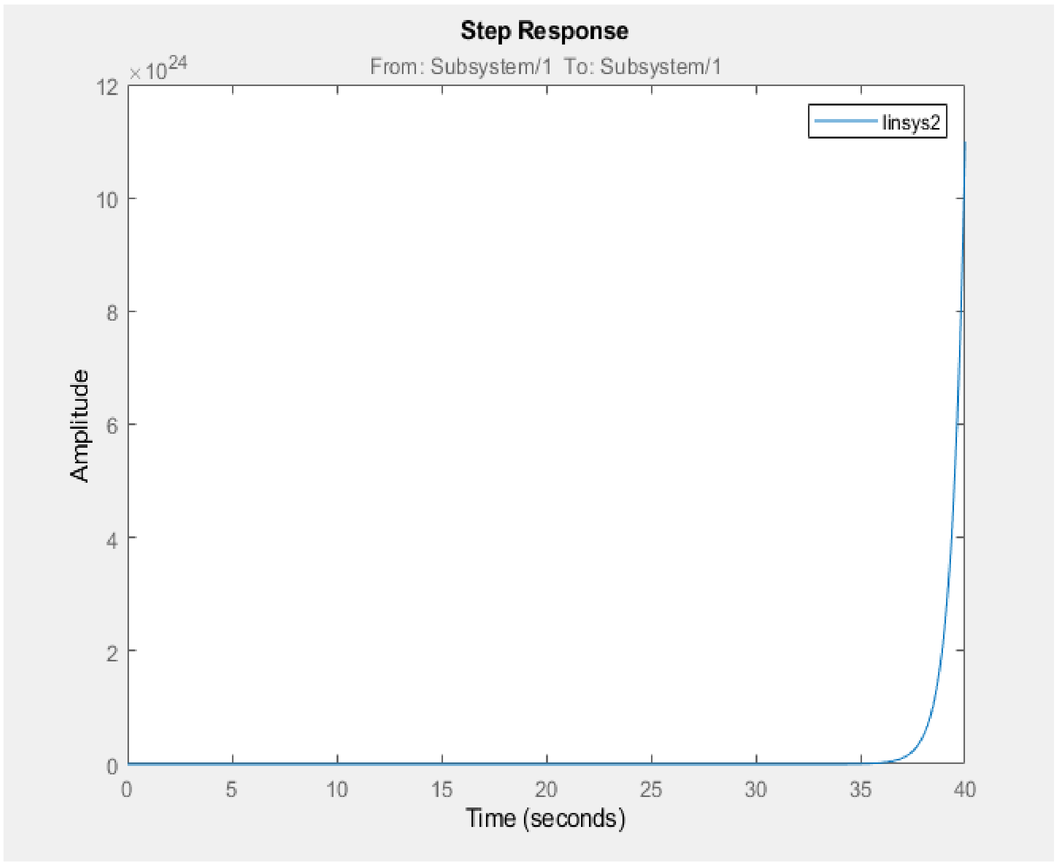
Fig(6)

Then, we created a subsystem and used the Linearization tool.



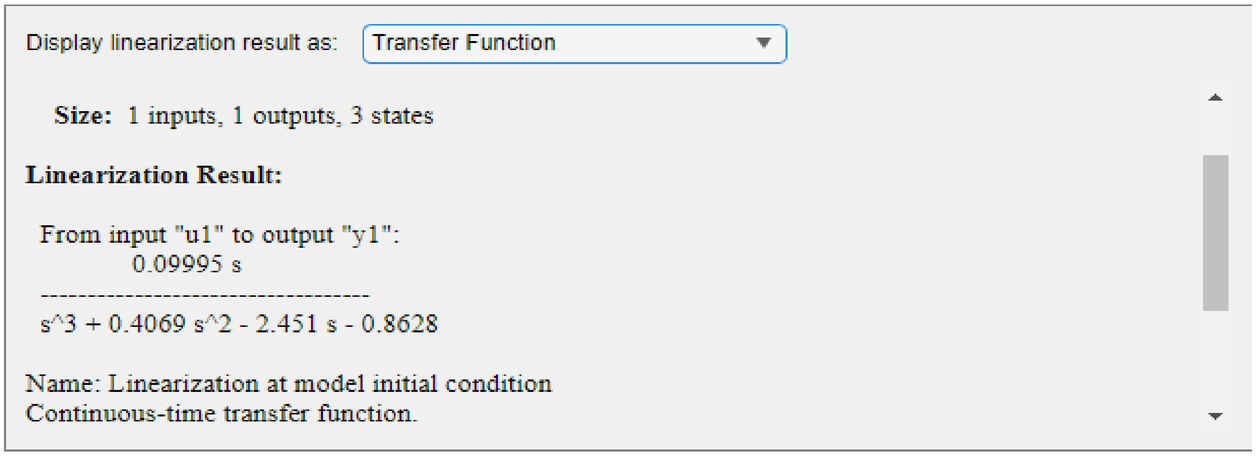
Fig(7)

The step response is not oscillating because the system is an inverted pendulum, and it is not stable:



Fig(8)

The transfer function generated by Simulink was:



```
  Size:  1 inputs, 1 outputs, 3 states

Linearization Result:

  From input "u1" to output "y1":
               0.09995 s
  ------------------------------------
  s^3 + 0.4069 s^2 - 2.451 s - 0.8628

Name: Linearization at model initial condition
Continuous-time transfer function.
```

Fig(9)

To validate our results we also calculated the transfer function manually and compared it to the above one:

Performing Laplace transform for both equations:

First equation: $U(s) = \mu xs \ + (M+m)xs^2 \ - ml\theta s^2$

Second equation: $l\, mxs^2 + mgl\,\theta = (I+ml^2)\theta s^2$

From the second equation: $x = \frac{(I+ml^2)\theta s^2 - mgl\theta}{lms^2}$

Substituting into the first equation, we get:

$$U(s) = \mu \frac{(I+ml^2)\theta s^2 - mgl\theta}{lms} + (M+m)\frac{(I+ml^2)\theta s^2 - mgl\theta}{lm} - ml\theta s^2$$

Dividing both sides on θ and substituting with the assumed values:

$$\frac{U(s)}{\theta(s)} = \mu \frac{(I+ml^2)s^2 - mgl}{lms} + (M+m)\frac{(I+ml^2)s^2 - mgl}{lm} - mls^2$$

$$\frac{U(s)}{\theta(s)} = \frac{s^3 + 0.4069s^2 - 2.451s - 0.8628}{0.09995s}$$

Therefore, our transfer function is:

$$\boxed{\frac{\theta(s)}{U(s)} = \frac{0.09995s}{s^3 + 0.4069s^2 - 2.451s - 0.8628}}$$

Investigating system stability:

When considering the natural response, a system is deemed stable if it tends towards zero as time progresses towards infinity. This stability is reflected in the presence of closed-loop transfer functions with poles located solely in the left half-plane.
Conversely, a system is considered unstable if its natural response grows infinitely as time approaches infinity. An unstable system is characterized by the existence of at least one pole in the right half-plane and/or poles with multiplicity greater than 1 on the imaginary axis within its closed-loop transfer functions.

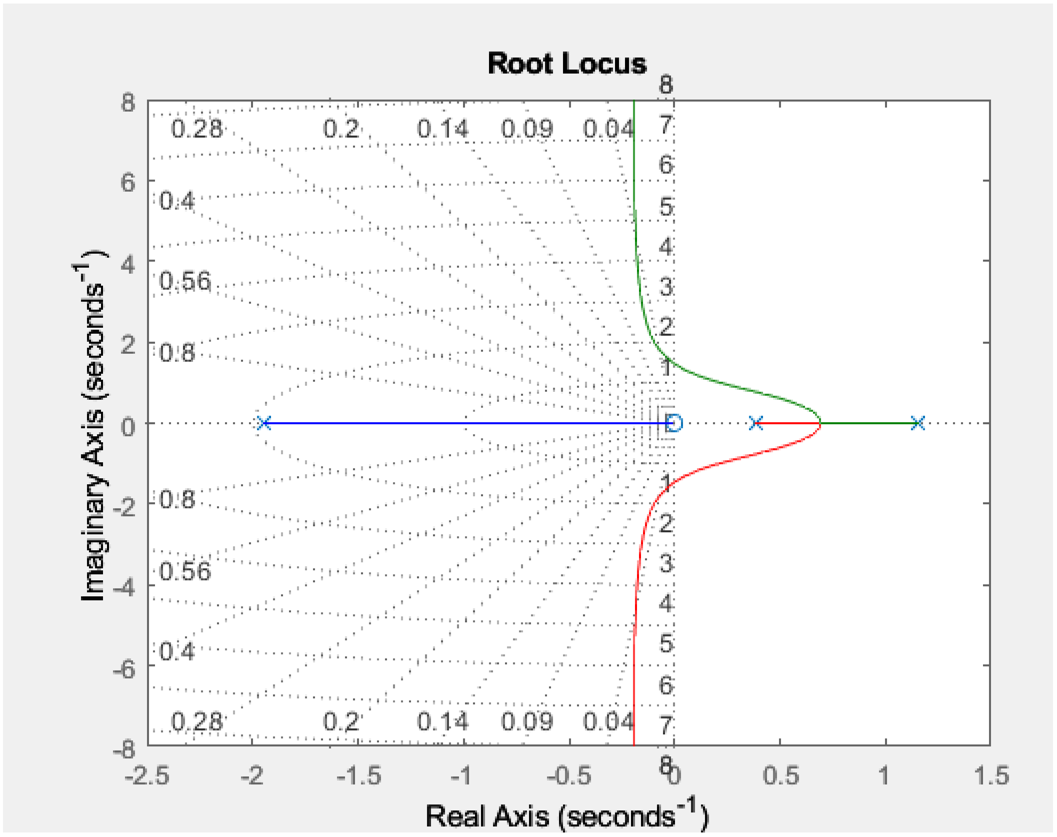
In the case of marginal stability, the natural response of a system does not decay or grow, but instead remains constant or oscillates.

Given that the closed-loop transfer function: $C(s) = \frac{0.09995\, s}{s^3+0.4069s^2-2.451s-0.8628}$

Also given that: $C(s) = \frac{T(s)}{1+T(s)}$, where T(s) is the open-loop transfer function.

Therefore, $T(s) = \frac{0.09995\, s}{s^3+0.4069\, s^2-2.55095\, s-0.8628}$ “by solving for T(s).

The root locus graph from MATLAB is:



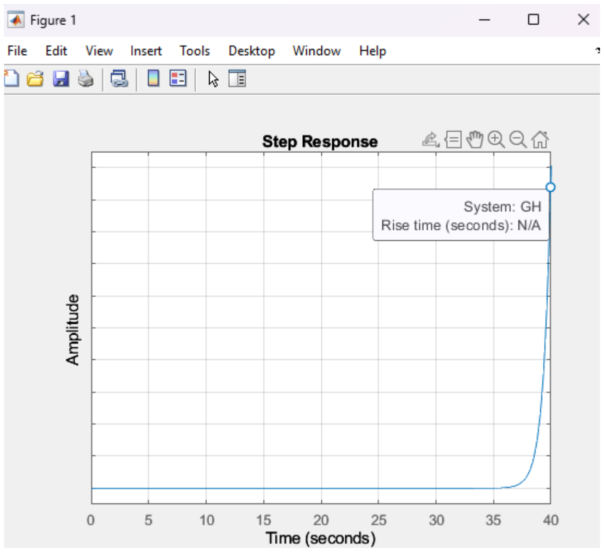
Fig(10)

Since the system has only one pole at the right, then the system is **unstable**.

Design objectives:

Since the system is unstable. Thus, the open loop parameters can't be determined.

For example, Rising time as shown in the opposite figure (can't be calculated):



Fig(11)

Steady state Error of the open-loop transfer function:

$$T(s) = \frac{0.09995\, s}{s^3 + 0.4069\, s^2 - 2.55095\, s - 0.8628}$$

To calculate steady state, the next formulas will be used:

- For a step input: $e(\infty) = e_{step}(\infty) = \frac{1}{1+K_P}$, Where $K_P = T(s)$

- For a ramp input: $e(\infty) = e_{ramp}(\infty) = \frac{1}{K_V}$, Where $K_V = sT(s)$

- For a parabolic input: $e(\infty) = e_{parabola}(\infty) = \frac{1}{K_a}$, Where $K_a = s^2T(s)$

- $K_P = T(s) = 0$, Then $e_{step}(\infty) = \frac{1}{1+0} = 1$

-$K_V = sT(s) = 0$, Then $e_{ramp}(\infty) = \infty$

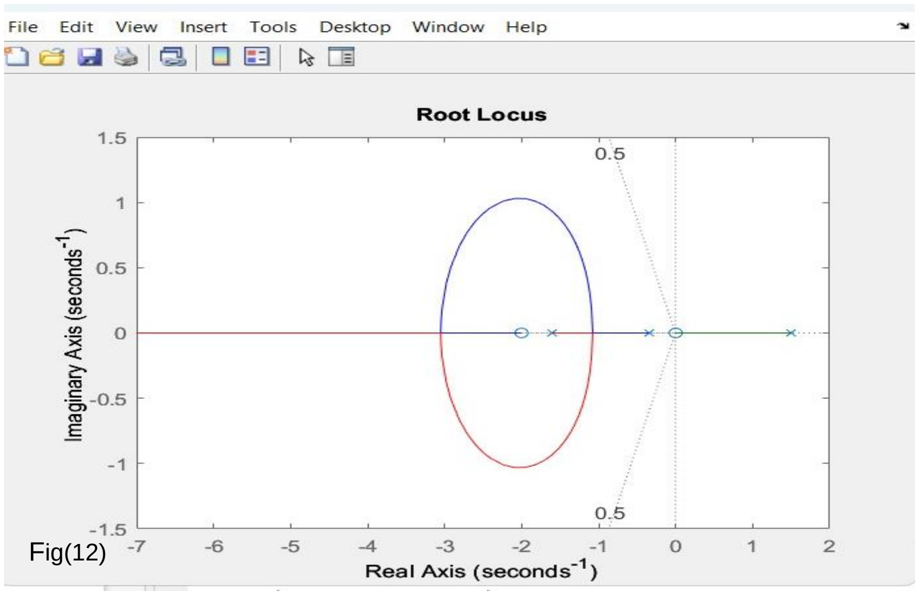
-$K_a = sT(s) = 0$, Then $e_{parabola}(\infty) = \infty$

Thus, we need a zero steady state error for step input for the closed-loop system.

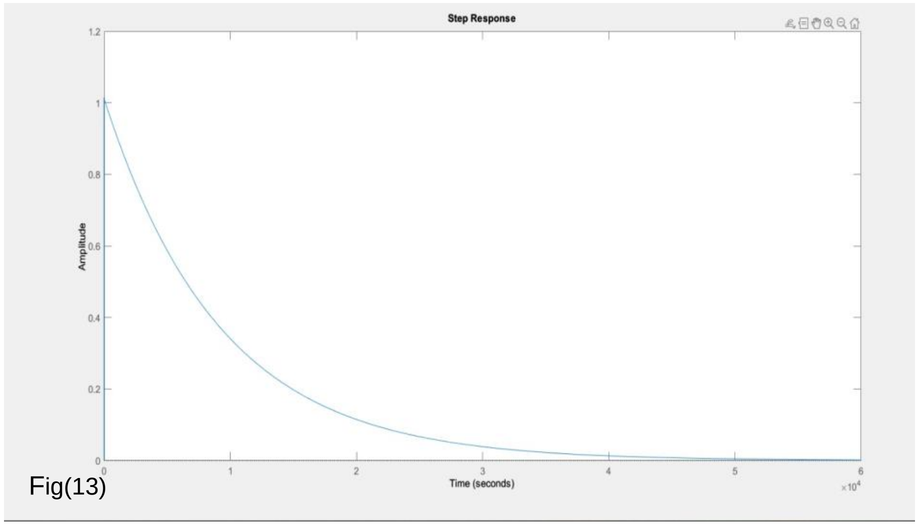
PID controller design using the Root Locus technique:

We assumed:

1. zeta $(\zeta = .5)$
2. Settling time (Ts=.4)
3. Gain (k=23)
4. The initial zeroes [0,-2]
5. Poles [1.5,-3.49,-1.6]



Fig(12)

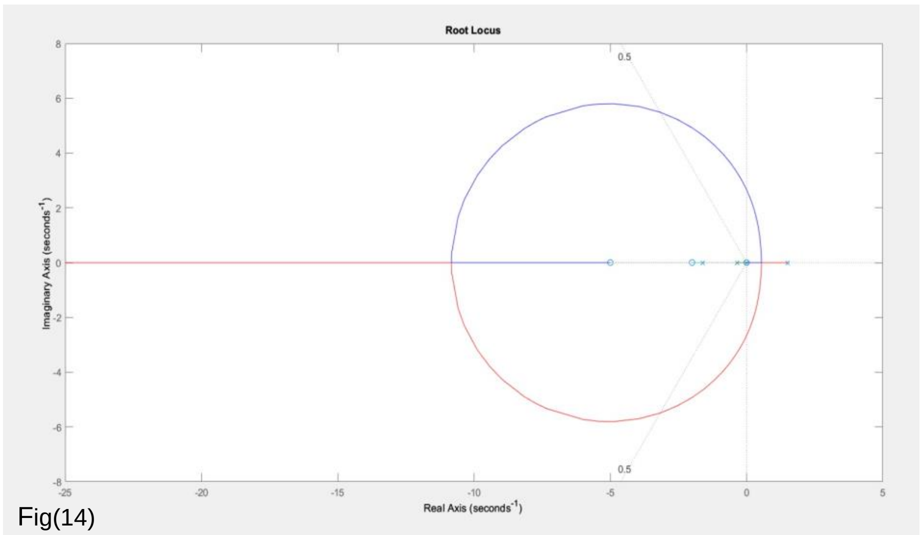


Fig(13)

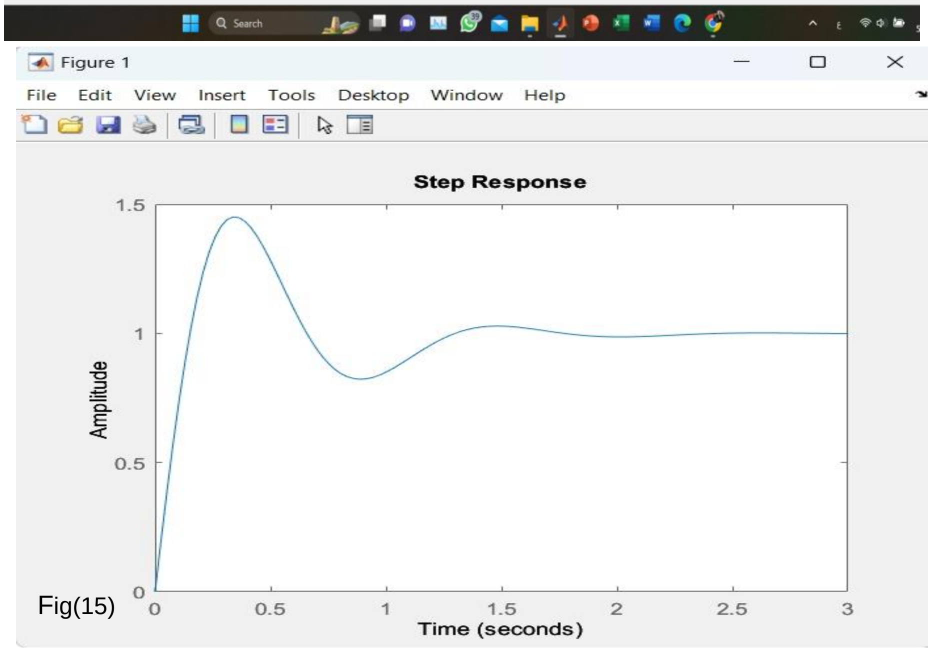
We increased number of zeroes and poles to improve the error.

Assume:

1. zeroes [0 -2 -3]
2. Poles [1.5 ,-3.49 ,-1.6 .01]



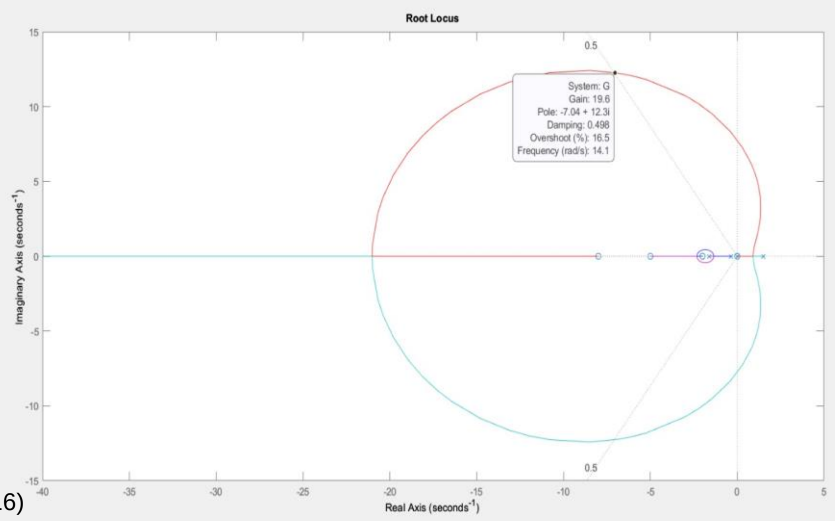
Fig(14)



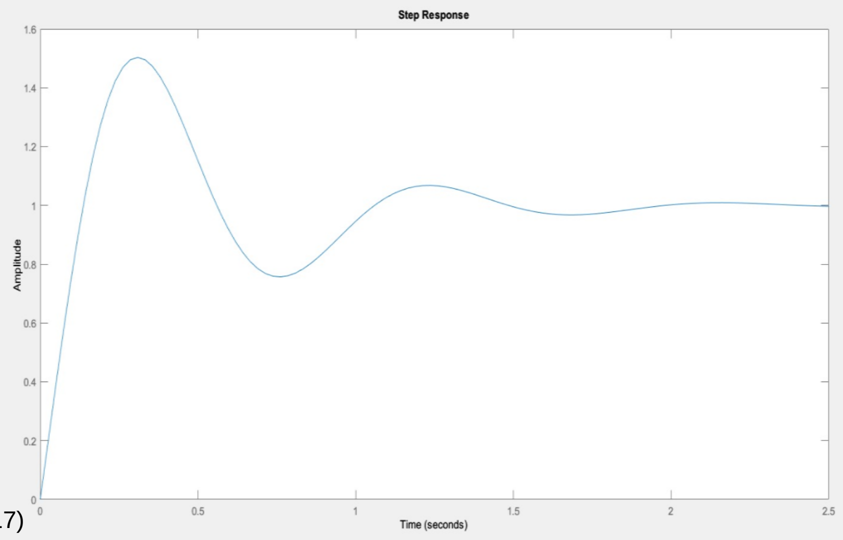
Fig(15)

Finally, Assume

1. zeroes [0 -2 -3 -2.5]
2. Poles [1.5 ,-3.49 ,-1.6 .01 .02]

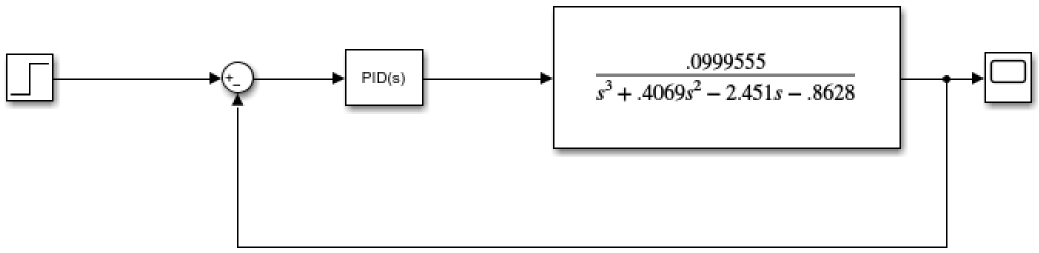


Fig(16)

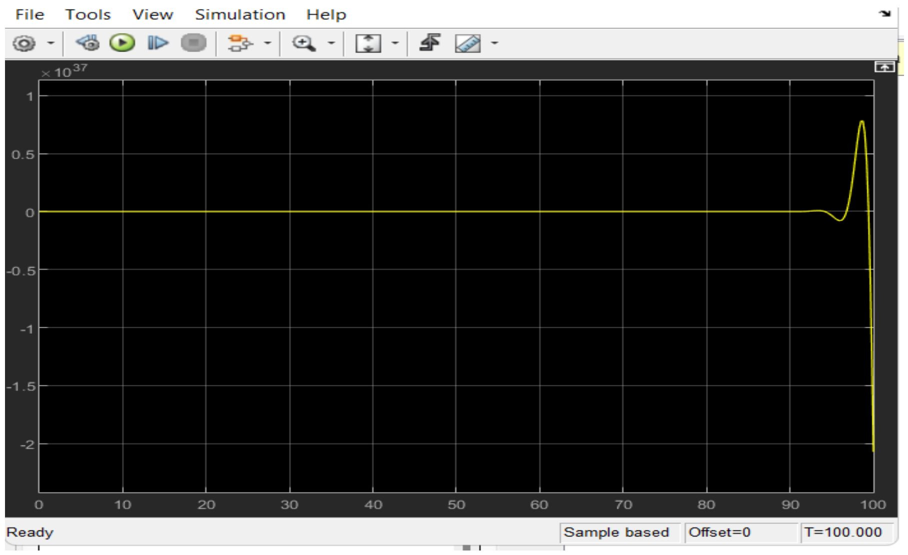


Fig(17)

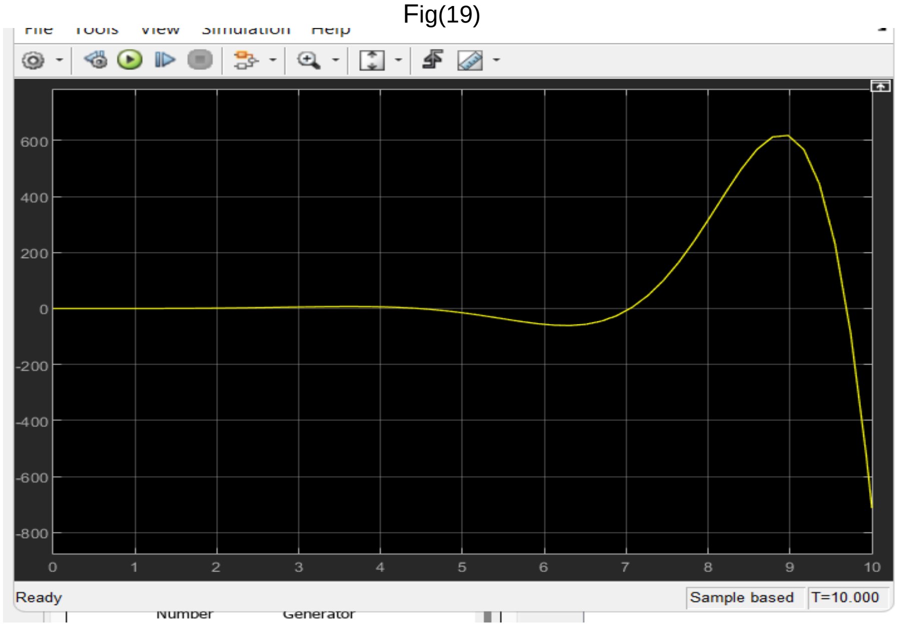
Designed PID controller in SIMULINK:



Fig(18)



Fig(19)



Fig(20)

Conclusion

In conclusion, we checked the stability of the system's open-loop transfer function using SIMULINK's Linearization tool. The analysis showed that the system is unstable as its natural response grows infinitely with time. The desired closed-loop response achieved, as we specified control design using the Root Locus approach. We assumed values for gain, settling time, and damping ratio to create a PID controller with starting zeroes and poles. The controller was iteratively enhanced by adding more zeroes and poles.

Appendix

Code connected to DE modeling.

```
clc
clear
close all
m = 2;     % mass of cart
M = 0.5;       % mass of pendulum
Muu = 0.88;   %Friction coefficient
L= 0.78; %Length of the rod
I = 6;   % Moment of inertia of the rod
g = 9.81;     % gravitational accelration
fraction = ((m^2)* L^2)/(I+(m*(L^2)));
Output = sim('projj.slx');
```

Root locus code:

```
% Define the transfer function numerator and denominator coefficients
num = [0.09995 0];              % Numerator coefficients
den = [1, 0.4069, -2.55095, 0.8628];        % Denominator coefficients
% Create the transfer function object
sys = tf(num, den);
% Plot the root locus
rlocus(sys)
```

PID controller Codes:

```
G = zpk([0 -2  -2.5 -3 -5],[1.5 -.349 -1.601 .01 .02 .03],1); %open-loop TF
figure; rlocus(G); % plot the root locus
zeta = .5; % damping ratio
sgrid(zeta,0) % radial line
figure; step(T); % step response of the CL system

%% System with pid
```

```
%open-loop TF
G = zpk([0 -2 -5 -2.5 -3],[1.5 -.349 -1.6 .01 .02 .03],1);

% PID TF (1/N = 1e-4)
Gc = pid(69,46,23,0);

% closed-loop TF
T = feedback(Gc*G,1);

step(T)
```

References

- Nise, N. S. (2020). *Control systems engineering*. John Wiley & Sons.