Name: Shaunak Deshpande

Branch: Instrumentation and Control Engineering

Year: TY

Division: C

Batch: 2

Roll Number: 39

Gr Number: 11911180

**Course Project 1: Inverted Pendulum Control**

2. Get the Step response for the Plant transfer function

3. Get the root locus for the Process Plant transfer function Gp(s) assuming feedback H(s) = 1

4. Design PI or PD or PID Controller Gc(s)

5. Cascade the Controller Gc(s) with the process plant transfer function Gp(s)

6. Get the root locus of the cascade transfer function Gc(s)Gp(s)

7. Get the closed-loop transfer function Y (s) R(s)

8. Get the Step response of the Closed-loop Control system

9. Analyze the best performance of the controller for the process plant in Python

10. Use the designed controller to be implemented real-time in the lab

The system in this example consists of an inverted pendulum mounted to a motorized cart. In this case we will consider a two-dimensional problem where the pendulum is constrained to move in the vertical plane shown in the figure below. For this system, the control input is the force 𝐹F that moves the cart horizontally and the outputs are the angular position of the pendulum 𝜃 and the horizontal position of the cart 𝑥.

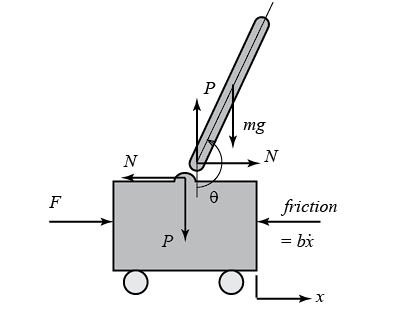
For this example, let's assume the following quantities:

(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)

The transfer function:

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

Assumed design requirements:  
Settling time for 𝜃 of less than 5 seconds  
Pendulum angle 𝜃 never more than 0.05 radians from the vertical



**Imports:**

import numpy as np

import control

from matplotlib import pyplot as plt

**Transfer function:**

M = 0.5;

m = 0.2;

b = 0.1;

I = 0.006;

g = 9.8;

l = 0.3;

q = (M+m)\*(I+ (m\*l\*l))-((m\*l) \* (m\*l));

pend\_tf\_num=[m\*l/q,0]

pend\_tf\_den=[1, b\*(I+ (m\*l\*l))/q, -(M+m)\*m\*g\*l/q, -(b\*m\*g\*l/q)]

Ppend=control.tf(pend\_tf\_num,pend\_tf\_den)

print("Pendulum TF: ", Ppend)

Pendulum TF:

4.545 s

----------------------------------

s^3 + 0.1818 s^2 - 31.18 s - 4.455

**Get the Step response for the Plant transfer function**

# t=np.arange(0,100.1,0.1)

T2, yout2 = control.step\_response(Ppend)

plt.plot(T2,yout2)

plt.title("Step Response of Pendulum")

plt.show()

Chart

Description automatically generated

For a bounded input, we are getting unbounded output. This is showing that the system is unstable. This is easily verified by observation as it is evident that even a small impulse will lead to inverted pendulum falling down.

**Get the root locus for the Process Plant transfer function Gp(s)**

control.rlocus(Ppend, print\_gain=True)

plt.title("Root locus of the system")

plt.show()

Chart

Description automatically generated

Since there is a pole in the right half plane, this again confirms that the system is unstable

**Design PI or PD or PID Controller Gc(s)**

**Cascade the Controller Gc(s) with the process plant transfer function**

# Kp = 1;

# Ki = 1;

# Kd = 1;

#Thus, the tf of PID will be:

pid=control.tf([1,1,1],[1,0])

print(pid)

ser = control.series(pid,Ppend)

output=control.feedback(ser,1)

t=np.arange(0,10,0.01)

a, b = control.step\_response(output,T=t)

plt.plot(a,b)

plt.xlabel("Time in seconds")

plt.ylabel("Amplitude")

plt.title("PID with Kp=1, Ki=1 & Kd=20")

plt.show()

s^2 + s + 1

-----------

s

Chart

Description automatically generated

Note how the system still has unbounded input. This means we have to tune the controller

# Kp = 100;

# Ki = 1;

# Kd = 1;

#Thus, the tf of PID will be:

pid2=control.tf([1,100,1],[1,0])

print(pid2)

ser2 = control.series(pid2, Ppend)

output2=control.feedback(ser2,1)

a2, b2 = control.step\_response(output2,T=t)

plt.plot(a2,b2)

plt.xlabel("Time in seconds")

plt.ylabel("Amplitude")

plt.title("PID with Kp=100, Ki=1 & Kd=1")

plt.show()

s^2 + 100 s + 1

---------------

s

Chart

Description automatically generated

Note now that the settling time is less than 5s, which is one of the design criteria. However, the peak response is still higher.

# Kp = 100;

# Ki = 1;

# Kd = 20;

#Thus, the tf of PID will be:

pid3=control.tf([20,100,1],[1,0])

print(pid3)

# T = control.feedback(Ppend,C);

ser3 = control.series(pid3,Ppend)

output3=control.feedback(ser3,1)

t=np.arange(0,10,0.01)

a3, b3 = control.step\_response(output3,T=t)

plt.plot(a3,b3)

plt.xlabel("Time in seconds")

plt.ylabel("Amplitude")

plt.title("PID with Kp=100, Ki=1 & Kd=20")

plt.show()

20 s^2 + 100 s + 1

------------------

s

Chart

Description automatically generated with medium confidence

Note how now the ripples are absent, the peak overshoot is low. This means no more tuning is required

**Get the root locus of the cascade transfer function**

FinalOutput=output3

print(output3)

control.rlocus(FinalOutput, print\_gain=True)

plt.title("Root locus of the system")

plt.show()

90.91 s^3 + 454.5 s^2 + 4.545 s

---------------------------------------

s^4 + 91.09 s^3 + 423.4 s^2 + 0.09091 s

Chart

Description automatically generated

**Get the closed loop transfer function Y (s)/R(s)**

**Get the Step response of the Closed loop Control system**

#Closed loop control system

Hs=1

Clcs= control.feedback(FinalOutput, Hs)

t=np.arange(0,10,0.01)

a4, b4 = control.step\_response(Clcs,T=t)

plt.plot(a4,b4)

plt.show()

Chart

Description automatically generated with medium confidence

Learning Outcomes:

1. Learn how to model a physical system
2. Learn how to get the transfer function of a physical system
3. Model said system in python
4. Find Step response of the model to learn if system is stable or unstable
5. Find Root locus of the model to learn if system is stable or unstable
6. Tune a PID controller