Name: Shaunak Deshpande

Branch: Instrumentation and Control Engineering

Year: TY

Division: C

Batch: 2

Roll Number: 39

Gr Number: 11911180

**Course Project 2: Inverted Pendulum Control using MPC**

1. Use the Process Plant in Course Project 1

2. Implement one of the following advanced controllers in software like MATLAB or Python

(a) State Feedback Controller

(b) Optimal Controller

(c) Adaptive Controller

**(d) MPC Controller**

3. Get the Step response of the closed-loop control system

4. Analyze the response of the system

5. Make a comparative analysis of the performance of classical PI / PD / PID controller and advanced controller used

**Python Code:**

**Imports:**

import matplotlib.animation as animation

import numpy as np

from gekko import GEKKO

#Defining a model

m = GEKKO()

#Weight of item

m2 = 1

#Defining the time, we will go beyond the 6.2s to check if the objective was achieved

m.time = np.linspace(0,8,100)

end\_loc = int(100.0\*6.2/8.0)

#Parameters

m1a = m.Param(value=10)

m2a = m.Param(value=m2)

final = np.zeros(len(m.time))

for i in range(len(m.time)):

if m.time[i] < 6.2:

final[i] = 0

else:

final[i] = 1

final = m.Param(value=final)

#MV

ua = m.Var(value=0)

#State Variables

theta\_a = m.Var(value=0)

qa = m.Var(value=0)

ya = m.Var(value=-1)

va = m.Var(value=0)

#Intermediates

epsilon = m.Intermediate(m2a/(m1a+m2a))

#Defining the State Space Model

m.Equation(ya.dt() == va)

m.Equation(va.dt() == -epsilon\*theta\_a + ua)

m.Equation(theta\_a.dt() == qa)

m.Equation(qa.dt() == theta\_a -ua)

#Definine the Objectives

#Make all the state variables be zero at time >= 6.2

m.Obj(final\*ya\*\*2)

m.Obj(final\*va\*\*2)

m.Obj(final\*theta\_a\*\*2)

m.Obj(final\*qa\*\*2)

m.fix(ya,pos=end\_loc,val=0.0)

m.fix(va,pos=end\_loc,val=0.0)

m.fix(theta\_a,pos=end\_loc,val=0.0)

m.fix(qa,pos=end\_loc,val=0.0)

#Try to minimize change of MV over all horizon

m.Obj(0.001\*ua\*\*2)

m.options.IMODE = 6 #MPC

m.solve() #(disp=False)

#Plotting the results

import matplotlib.pyplot as plt

plt.figure(figsize=(12,10))

plt.subplot(221)

plt.plot(m.time,ua.value,'m',lw=2)

plt.legend([r'$u$'],loc=1)

plt.ylabel('Force')

plt.xlabel('Time')

plt.xlim(m.time[0],m.time[-1])

plt.subplot(222)

plt.plot(m.time,va.value,'g',lw=2)

plt.legend([r'$v$'],loc=1)

plt.ylabel('Velocity')

plt.xlabel('Time')

plt.xlim(m.time[0],m.time[-1])

plt.subplot(223)

plt.plot(m.time,ya.value,'r',lw=2)

plt.legend([r'$y$'],loc=1)

plt.ylabel('Position')

plt.xlabel('Time')

plt.xlim(m.time[0],m.time[-1])

plt.subplot(224)

plt.plot(m.time,theta\_a.value,'y',lw=2)

plt.plot(m.time,qa.value,'c',lw=2)

plt.legend([r'$\theta$',r'$q$'],loc=1)

plt.ylabel('Angle')

plt.xlabel('Time')

plt.xlim(m.time[0],m.time[-1])

plt.show()

**Output:**

Chart, line chart

Description automatically generated