**Python PID Control System for Mass-Spring-Damper EGR 598 Project Code Explanation**

**CODE**

“# Importing necessary libraries

import numpy as np

import matplotlib.pyplot as plt”

**Explanation:** The code starts by importing two essential libraries, NumPy and Matplotlib. NumPy is used for numerical operations, and Matplotlib is used for data visualization, specifically for plotting graphs.

“# Mass-Spring-Damper System Parameters

m = 1.0 # Mass (kg)

k = 10.0 # Spring constant (N/m)

c = 1.0 # Damping coefficient (Ns/m)”

**Explanation:** These lines define the parameters of the mass-spring-damper system. “m” represents the mass, “k” is the spring constant, and “c” is the damping coefficient. These values are typical parameters used in modeling mechanical systems.

“# PID Controller Parameters

Kp = 20.0 # Proportional gain

Ki = 10.0 # Integral gain

Kd = 2.0 # Derivative gain”

**Explanation:** Here, the code sets the parameters for the PID (Proportional-Integral-Derivative) controller. “Kp”, “Ki”, and “Kd” are the proportional, integral, and derivative gains, respectively. These gains determine how much the controller responds to the current error, accumulated error over time, and the rate of change of error.

“# Simulation Parameters

dt = 0.01 # Time step (seconds)

total\_time = 10.0 # Total simulation time (seconds)

num\_steps = int(total\_time / dt)”

**Explanation:** This section defines the parameters for the simulation. dt is the time step used in the simulation. total\_time is the duration of the simulation, and num\_steps calculates the total number of steps needed based on the time step and total simulation time.

“# Initial Conditions

x0 = 1.0 # Initial position (m)

v0 = 0.0 # Initial velocity (m/s)”

**Explanation:** These lines set the initial conditions of the system. “x0” is the initial position of the mass, and “v0” is the initial velocity. These values determine the starting state of the system.

“# Reference Position (Setpoint)

setpoint = 0.0 # The desired position (m)”

**Explanation:** setpoint represents the desired or target position that the PID controller will try to achieve. In this case, it's set to 0.0 meters.

“# Lists to store data for plotting

time = [0.0]

position = [x0]”

**Explanation:** Two lists, time, and position are initialized. These lists will be used to store data during the simulation for later plotting.

“# Initialize PID controller error terms

integral = 0.0

previous\_error = 0.0”

**Explanation:** integral and previous\_error are variables used to store the integral of the error (cumulative error) and the previous error, respectively. These are crucial for the PID control calculations.

“# Simulation Loop

x = x0

v = v0

for \_ in range(num\_steps):

# Calculate control input using PID control

error = setpoint - x

integral += error \* dt

derivative = (error - previous\_error) / dt

control\_input = Kp \* error + Ki \* integral + Kd \* derivative

# Equations of motion (Newton's second law)

acceleration = (control\_input - c \* v - k \* x) / m

v += acceleration \* dt

x += v \* dt

# Store data for plotting

time.append(time[-1] + dt)

position.append(x)

previous\_error = error”

**Explanation:** This section represents the main simulation loop. It iterates through the time steps and calculates the control input using the PID control formula. It then updates the position and velocity of the system based on the equations of motion (Newton's second law). The results are stored in the “time” and “position” lists for later plotting.

“# Plot the results

plt.figure()

plt.plot(time, position, label='Position')

plt.plot([0, total\_time], [setpoint, setpoint], 'r--', label='Setpoint')

plt.xlabel('Time (s)')

plt.ylabel('Position (m)')

plt.title('Mass-Spring-Damper System with PID Control')

plt.legend()

plt.grid(True)

plt.show()”

**Explanation:** This section uses Matplotlib to plot the simulation results. It creates a plot of position over time and includes a red dashed line representing the setpoint. This visualization helps in understanding how well the PID controller achieves the desired position.

**OUTPUT**

The output is a plot that visually represents the simulation results of a mass-spring-damper system controlled by a PID controller. The key components of the output are as follows:

A graph of a line graph

Description automatically generated

1. **Position vs. Time Plot:**

The primary plot shows the position of the mass over time. The x-axis represents time in seconds, and the y-axis represents the position of the mass in meters.

1. **Setpoint Reference Line:**

A red dashed line on the plot represents the setpoint or the desired position that the PID controller aims to achieve. In the provided code, the setpoint is set to 0.0 meters.

1. **Legend:**

The plot includes a legend that distinguishes between the actual position ("Position") and the setpoint reference line.

1. **Axes Labels:**

The x-axis is labeled "Time (s)," and the y-axis is labeled "Position (m)."

1. **Visualization of System Response:**

The plot visually illustrates how the position of the mass changes over time in response to the PID controller. The goal is for the system to reach and maintain the setpoint, demonstrating the effectiveness of the PID control in regulating the position.

The output provides a clear representation of the dynamic behavior of the simulated system under the influence of the PID controller. Users can observe how the system responds to disturbances and changes in the setpoint, gaining insights into the principles of control systems and the impact of PID gains on system performance.