ECEn 483 / ME 431

Final - Winter 2023

Professor R. W. Beard

Name: ______________________________

Open book, open notes, open J drive, open Python. You may only use files that were provided by the instructor, or that you developed yourself.

Closed email, internet, and other forms of communication.

Work all problems. Unless directed otherwise, write solutions on this document.

Draw a box around your final answer.

Part 1	________/ 25
Part 2	________/ 25
Part 3	________/ 25
Part 4	________/ 25
Total	________/ 100

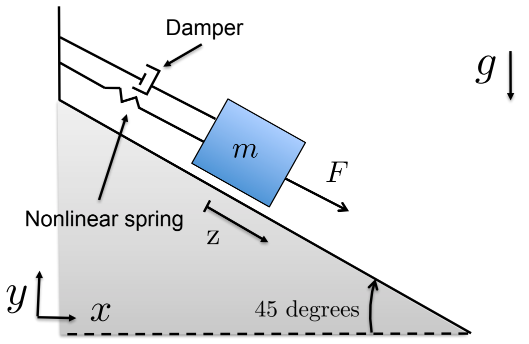
Equations of Motion - Simulation Model

The figure below shows a mass on a **fixed** inclined plane connected to a nonlinear spring and a linear damper. The potential energy for the nonlinear spring is given by

$$V_{\text{spring}} = \frac{1}{2}k_1 z^2 + \frac{1}{4}k_2 z^4.$$

The damping coefficient is $b = 0.1$. The physical parameters of the system are $g = 9.8$ meters per second, $\theta = 45$ degrees, $m = 0.5$ kg, $k_1 = 0.05$, and $k_2 = 0.02$.

The input force F is limited to ± 25.0 Newtons.



The nonlinear equations of motion for this system are

$$m\ddot{z} + b\dot{z} + k_1 z + k_2 z^3 - mg\sin\left(\frac{\pi}{4}\right) = F.$$

A Python implementation of this system is in `slopedMassDynamics.py`, with parameters in `slopedMassParameters.py`, and animation in `slopedMassAnimation.py`.

Part 1. Design Models

For this section, use `part1_slopedMassSim.py` to implement the simulation.

The objective of this part is to use the equations of motion to find the appropriate design models that will be used to design the feedback control strategies.

1.1 Suppose that the objective is to linearize the system around the equilibrium position z_e, which may or may not be zero. Find the associated equilibrium force F_e so that z_e is an equilibrium of the system.

1.2 Create a controller that places a constant force of F_e on the physical system. Set the initial conditions to $z(0) = \dot{z}(0) = 0$ to verify that the equilibrium force is correct, assuming that $z_e = 0$ degrees. **Insert a plot of the output of the system with initial condition $z(0) = z_e = 0$ and an input of F_e in the associated Word document.**

1.3 Using Jacobian linearization, linearize the nonlinear model around the equilibrium (z_e, F_e).

1.4 Find the transfer function of the linearized model when $z_e = 0$.

1.5 Find a state-space model for the system linearized around z_e, F_e when $z_e = 0$. For your states use $x = (\tilde{z}, \dot{\tilde{z}})^\top$.

Part 2. PID Control

For this section, use `part2_slopedMassSim.py` to implement the simulation. An input disturbance of $d = 1.0$ should be included in the simulation.

The sampling rate for the controller is $T_s = 0.01$. Use a dirty derivative gain of $\sigma = 0.005$.

2.1 Using the transfer function derived in Problem 1.4, draw the block diagram for the system using PD control, where the derivative gain multiples the velocity and not the derivative of the error.

2.2 Derive the transfer function from the reference input z_r to the position z.

2.3 Find the gain k_p and k_d so that the rise time of the system is $t_r = 0.5$ and the damping ratio is $\zeta = 2.0$. Write down your final gains in this booklet.

2.4 Using a dirty derivative, implement PD control where the input z_r is a square wave with an amplitude of ± 0.5 meters and a frequency of $\omega = 0.05$ Hertz. **Insert a plot in the Word file that shows both z_r and z for 50 seconds of simulation.**

2.5 Add an integrator to remove the steady state error. What is the value of the integrator gain k_i? **Insert a plot in the Word file that shows both z_r and z for 50 seconds of simulation.**

2.6 Insert a copy of the control code `ctrlPID.py` in the Word document.

Part 3. Observer-based Control

For this section, use `part3_slopedMassSim.py` to implement the simulation. An input disturbance of $d = 1.0$ should be included in the simulation. The sampling rate for the controller is $T_s = 0.01$.

The objective of this part is to design a state feedback controller of the form

$$F = F_e - K\hat{x} - k_i \int (z_r - C_r\hat{x})d\sigma - \hat{d}$$

to regulate the position to a commanded input, and where the estimated state $\hat{x}$ and the estimated disturbance $\hat{d}$ is produced by the observer

$$\dot{\hat{x}}_2 = A_2(\hat{x}_2 - x_{2e}) + B_2(u - u_e) + L_2(y - C_2\hat{x}_2).$$

3.1 Find the feedback gain K that places the poles so that the rise time is $t_r = 0.5$ s and the damping ratio is $\zeta = 2.0$, and the integrator gain k_i so that the pole of the integrator is at $p_{int} = -2$. Write the gains in this booklet.

3.2 Design a disturbance observer, so that the poles of the observation error, i.e., the eigenvalues of $A - LC$, are five times the eigenvalues of $A - BK$, and the pole of the disturbance observer is $p_{dist} = -20$. Write the observer gains L_2 in this booklet.

3.3 Insert a plot of the step response of the system (z and z_r), for the complete observer based controller in the Word document.

3.4 Insert a plot of the estimation error and disturbance estimation error in the Word document.

3.5 Insert a copy of the control code `ctrlObsv.py` in the Word document.

Part 4. Loopshaping

For this section, use `part4_slopedMassSim.py` to implement the simulation. An input disturbance of $d = 1.0$ should be included in the simulation. The sampling rate for the controller is $T_s = 0.01$.

4.1 Using the PID controller derived in the Part 2, graph the Bode Plot of the loop gain of the original open loop system using PID control. Using PID control, what is the attenuation on the noise for noise with frequency content above 1000 rad/sec? Using PID control, what is the tracking accuracy for reference signals with frequency content below 0.01 rad/sec.

4.2 Starting with the PID controlled plant, add a lead filter so that the phase margin is $PM = 60°$.

4.3 Add a lag filter to improve the tracking accuracy for signals with frequency content below $\omega_r = 0.1$ rad/sec by a factor of 10.

4.4 Add a low pass filter to increase noise attenuation by a factor of 10 for noise with frequency content above $\omega_n = 1000$ rad/sec. What is the final controller $C(s)$? Write the transfer function in this booklet. *Hint: Use* `print('C(s)=', C)` *in the loopshaping file.*

4.5 Add a prefilter to remove any overshoot in the response. Write the transfer function of the prefilter in this booklet. *Hint: Use* `print('F(s)=', F)` *in the loopshaping file.*

4.6 Insert a graph in the Word file that simultaneously show Bode plots for the original plant, the PID controlled plant, and the plant augmented with loopshaping.

4.7 Implement the controller in simulation. **Insert a plot in the Word file that shows both z_r and z for 50 seconds of simulation.**

4.8 Insert a copy of the loopshaping code `loopshapeslopedMass.py` in the Word document.