**ECEn 483 / ME 431**

# Final - Winter 2021

## Time Limit: 4 HOURS

Name: _____
Start Time: _____
End Time: _____

Open book, open notes/slides, open J drive, open MATLAB/Python/MS Word.

Closed email, internet (except for access to Python or MATLAB documentation, and Learning Suite), and other forms of communication. All helper code provided is in the Python language only. However, you may use MATLAB for certain aspects if you find it helpful.

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

**Draw a box around your final answer.**

Note that Alt-Printscreen copies the contents of the selected window to the clipboard.

| | |
|---|---|
| Part 1 | _____/ 20 |
| Part 2 | _____/ 20 |
| Part 3 | _____/ 20 |
| Part 4 | _____/ 20 |
| Part 5 | _____/ 20 |
| Total | _____/ 100 |

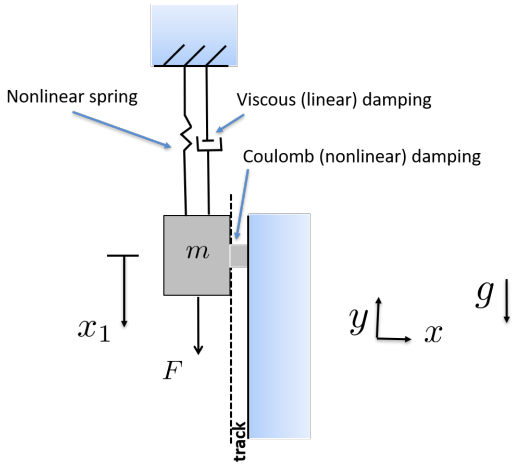
# Part 1. Equations of Motion for Simulating Dynamic Systems

The figure below shows a mass attached to the top surface with a nonlinear spring and linear damper and its initial position (described by $x_1 = 0$) is when the spring is unstretched. Additionally, the mass rubs against the dotted horizontal surface, but has a component inside the shown track so it will not come away from the surface. The potential energy for the nonlinear spring is given by

$$V_{\text{spring}} = \frac{1}{2}k_1 x_1^2 + \frac{1}{4}k_2 x_1^4.$$

The nonlinear damping force (Coulumb friction) due to the mass sliding on the surface is given by

$$f_{nonlinear,friction} = c * sign(\dot{x}_1).$$




**1.1** Using the configuration variable $q = x_1$, find the kinetic energy of the system.

**1.2** Find the potential energy for the system.

**1.3** Find the Lagrangian $L = K - P$.

**1.4** Find the generalized forces.

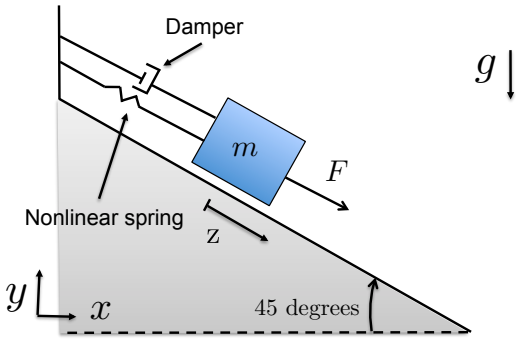
**1.5** Derive the equations of motion using the Euler-Lagrange equations.

# Part 2. Models for Control Design

The figure below shows a mass on a fixed inclined plane connected to a nonlinear spring and a linear damper and the equation of motion is the following:

$$m\ddot{z} + k_1 z + k_2 z^3 - \frac{1}{\sqrt{2}} mg = F - b\dot{z} \tag{1}$$

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 $\pm 5.0$ Newtons.




For this section, use the following files to implement the simulation: `finalSim2.py, massDynamics.py`. The objective of this part is to use the equations of motion to find the appropriate models that will be used to design the feedback control strategies. If you CANNOT get the dynamics to work, you can instead use the file "massDynamicsCompiled.py," but you will take a 10 point deduction by using that file instead of writing your own dynamics for the rest of the exam.

**2.1** Suppose that the objective is to linearize the system around the equilibrium position $z_e$. Find the associated equilibrium force $F_e$ so that $z_e$ is an equilibrium of the system.

**2.2** Create a "controller" (at this point it is mostly just a simulation of the equilibrium conditions) that places a constant force of $F_e$ on the physical system. In the simulation files, set the initial conditions to $z(0) = z_e$ and $\dot{z}(0) = 0$ to verify that the equilibrium force is correct. For this problem you may assume that $z_e = 0$. **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.**

**2.3** Linearize the model around the equilibrium $(z_e, F_e)$, for $z_e \neq 0$ and report your linearized model.

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

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

**2.6** Did you write your own dynamics, or did you use the complied file? **Please insert your dynamics file that you used in the associated Word document.**

# Part 3. PID Control

For this section, use the following files to implement the simulation: `finalSim3.py`, `controllerPID.py`. The sampling rate for the controller is $T_s = 0.01$. Use a dirty derivative gain of $\sigma = 0.005$.

**3.1** Using the transfer function derived in Problem 2.4, draw the block diagram for the system using PD control, where the derivative gain multiplies the derivative of the output, and not the derivative of the error.

**3.2** Derive and report the closed-loop transfer function from the reference input $z^r$ to the position $z$.

**3.3** Find the proportional gain $k_p$ so that the control input $F$ saturates at $F_{\max} = 5$ Newtons when a step of 1.0 meter is commanded.

**3.4** If the desired closed loop characteristic polynomial is given by

$$\Delta_{cl}^d(s) = s^2 + 2\zeta\omega_n s + \omega_n^2,$$

find the natural frequency $\omega_n$ and the derivative gain $k_d$ so that the actual closed loop characteristic equation equals the desired closed loop characteristic equation when $\zeta = 0.707$.

**3.5** Using a dirty derivative, implement PD control, where the input $z^r$ is a square wave with an amplitude of $\pm 0.5$ meters and a frequency of 0.05 Hertz. **Insert a plot in the Word file that shows both $z^r$ and $z$ for 40 seconds of simulation.**

**3.6** Add an integrator to remove the steady state error and tune to obtain good transient performance. **Insert a plot in the Word file that shows both $z^r$ and $z$ for 40 seconds of simulation.** What is the integrator gain?

$k_i =$

**3.7 Insert a copy of the control code (`controllerPID.py`) in the Word document.**

# Part 4. Observer-based Control

For this section, use the following files to implement the simulation: `finalSim4.py`, `controllerObsv.py`. 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

$$u = u_e - K\hat{x} - k_i \int (z^r - \hat{z}) d\sigma$$

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

$$\dot{\hat{x}} = A(\hat{x} - x_e) + B(u - u_e) + L(y - C\hat{x}).$$

**4.1** Find the feedback gain $K$ that places the poles at $s = -2 \pm 2j$ and the integrator gain $k_i$ so that the pole of the integrator is at -5.0 (left half plane).

**4.2** Find the observer gains so that the poles of the observation error, i.e., the eigenvalues of $A - LC$, are roughly five times the eigenvalues of $A - BK$.

**4.3** Implement the observer based control and tune the controller and estimator to get good performance.

**4.4 Insert a plot of the step response of the system ($z$ and $z^r$) in the Word document for the complete observer based controller.**

**4.5 Insert a plot of the estimation error in the Word document.**

**4.6 Insert a copy of the control code (`controllerObsv.py`) in the Word document.**

# Part 5. Loopshaping

For this section, use the following files to implement the simulation: `finalSim5.py`, `controllerLoop.py`. The sampling rate for the controller is $T_s = 0.01$. For problems 5.2 through 5.5 you can either start with the given $C_{pid}(s)$ in problem 5.1 and add compensators, or you can start without any controller. However, your final controller needs to still have adequate performance (good rise time and phase margin), in addition to satisfying the constraints listed below.

**5.1** Using a new PID controller (with requirements that were different than specified in part 3), graph the Bode Plot of the loop gain of the original open loop system without any control, and the Bode Plot of the loop gain using PID control. The gains are as follows: $k_p = 5$, $k_d = 2.9$, and $k_i = 0.1$. What is the attenuation on noise with frequency content above 20 rad/sec?

Using PID control, what is the tracking accuracy for reference signals with frequency content below 0.2 rad/sec?

**5.2** Add a lag filter to improve the tracking accuracy for signals with frequency content below 0.001 rad/sec such that $\gamma_r = 10^{-4}$ or -80 dB.

**5.3** Add a low-pass filter to increase noise attenuation such that for frequency content about 1000 rad/sec, $\gamma_n = 10^{-5}$ or -100 dB (i.e., the magnitude ratio for $\frac{E}{N}$ is less than or equal to $10^{-5}$). Write the augmented controller $C(s)$ (for parts 5.2 and 5.3) below:

**5.4** Add a prefilter to remove any overshoot in the response. Write the form and values for your prefilter $F(s)$ below:

**5.5** Implement the loopshaping-based controller in code along with adding some noise (above 1,000 rad/sec) to your measurement signal in order to test the low-pass filter portion of your loopshaping controller. Adjust your loopshaping additions if necessary.

**5.6 Insert a graph in the Word file that simultaneously (in the same plot) shows Bode plots for the original plant ($P(s)$), the PID controlled plant ($P(s)C_{pid}(s)$), and the plant augmented with loopshaping ($P(s)C(s)$, where $C(s)$ may or may not include the PID controller, it is up to you).**

**5.7 Insert a plot in the Word file that shows both $z^r$ and $z$ for 40 seconds of simulation.**

**5.8 Insert a copy of the control code (`controllerLoop.py`) AND loopshaping code in the Word document.**