ECEn 483 / MeEn 431

Final - Winter 2025

Time Limit: 3 Hours 40 Minutes (including uploading files)

Name: _____
Start Time: _____
End Time: _____

Open book, open notes/slides, open code that was written by you, open J drive, open MATLAB/Python/MS Word. **In terms of AI, you may not use ChatGPT, Co-Pilot or any other similar tools.**

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, then scan and submit this in addition to your code and the Word file. You MUST submit a zipped folder with ALL of your code in it (in addition to the code requested for the Word file).

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	_____/ 25
Part 4	_____/ 35
Total	_____/ 100

Part 1. Equations of Motion - Simulation Model (20 points)

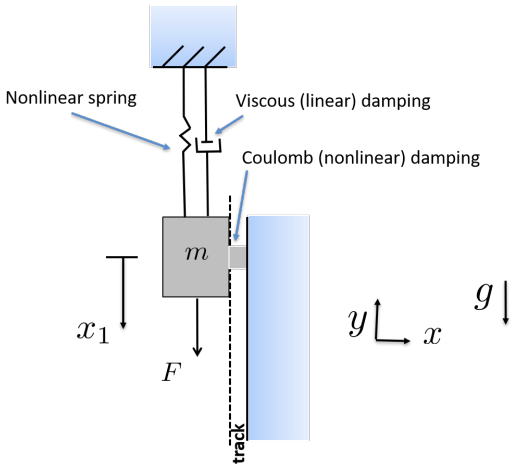
The figure below shows a mass attached to the top surface with a nonlinear spring and linear damper and its initial position (described by $x_1 = 0$) is when the spring is unstretched. Additionally, the mass rubs against the dotted horizontal surface experiencing a Coulomb friction force, and has a component 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 (Coulomb friction) due to the mass sliding on the surface is given by

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

where the *sign* function just returns a plus one or minus one, depending on the sign of $\dot{x}_1$.



1.1 (3 points) Identify the configuration variable (or generalized coordinate) and find the kinetic energy of the system.

1.2 (3 points) Find the potential energy for the system.

1.3 (3 points) Find the Lagrangian $L = K - P$.

1.4 (3 points) Find the generalized forces.

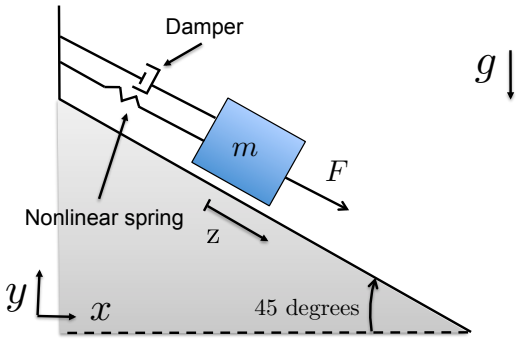
1.5 (8 points) Derive the equations of motion using the Euler-Lagrange equations.

Part 2. Models for Control Design (20 points)

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 ± 5.0 Newtons.



For this problem, 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 (i.e. 10% of entire test) by using that file instead of writing your own dynamics for the rest of the exam.

2.1 (3 points) 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 and report it below:

$F_e =$

2.2 (6 points) 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 (5 points) Linearize the model around the equilibrium (z_e, F_e), for $z_e \neq 0$ and report your linearized model:

2.4 (3 points) Find the transfer function of the linearized model when $z_e = 0$, and report that model too:

2.5 (3 points) 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*** and report the model below:

2.6 Did you write your own dynamics, or did you use the compiled file? **Please insert the dynamics file that you used (if you wrote it) in the associated Word document.**

Part 3. PID Control (25 points)

For this problem, use the following files to implement the simulation: `finalSim3.py`, `controllerPID.py`. The sampling rate for the controller is $T_s = 0.01$.

3.1 (2 points) 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 (3 points) Derive and report the closed-loop transfer function from the reference input z_r to the position z.

3.3 (2 points) 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 (3 points) 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 ω_n and the derivative gain k_d so that the actual closed loop characteristic equation equals the desired closed loop characteristic equation when $\zeta = 0.95$. Show your work.

$\omega_n =$

$k_d =$

3.5 (5 points) 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 0.05 Hertz. **Insert a plot in the Word file that shows both z_r and z for 40 seconds of simulation.**

3.6 (5 points) 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 (5 points) **Insert a copy of the control code (`controllerPID.py`) in the Word document.**

Part 4. State Space Control (35 points)

For this section, use the following files to implement the simulation: `finalSim4.py`, `controllerStateSpace.py`. The sampling rate for the controller is $T_s = 0.01$. The objective of this part is to design a state feedback controller in stages (first with only full state feedback and an integrator state, then with an observer and disturbance observer, and then using LQR to find gains for the controller).

4.1 (5 points) 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 (i.e. in the left half plane). What are these gains?

$K =$

$k_i =$

4.2 (4 points) For the same reference input as in Part 3, tune the controller to get good performance in code, assuming you have access to the full state. Also, make sure to include the disturbance that is defined in the template code as part of your input to the dynamics. **Then insert a plot of the step response of the system (z and z_r), for the state space controller with an integrator.**

4.3 (5 points) Find the observer gains so that the poles of the observer are five times *faster* (not necessarily five times larger in magnitude) than the controller poles from problem **4.1**. What are those gains?

$L =$

4.4 (4 points) Now add a disturbance observer, where the pole of the disturbance observer is $p_{dist} = -2$, and report the new L matrix (called L_2 in our notes).

$L_2 =$

4.5 (4 points) Now implement the observer-based controller (estimating $\hat{x}$ and $\hat{d}$ and using *both* for control) and tune for good performance. **Insert a plot of the response of the system (z and z_r), for the complete observer-based controller in the Word document.**

4.6 (4 points) **Also, insert a plot of the estimation error and disturbance estimation error in the Word document.**

4.7 (4 points) Finally, find a new set of gains K and k_i using the LQR method. Keep your old pole-placement code, but comment it out. Tune the LQR controller to use half as much maximum input force ($\tilde{F}$) as in 4.2. **Insert a copy of the response in the Word document.** Report the values for Q and R that you used below:

$Q =$

$R =$

4.8 (5 points) **Insert a copy of the final control code (`controllerStateSpace.py`) in the Word document.**