ECEn 483 / ME 431

Practice Final - Winter 2020

Author: Professor R. W. Beard

Name: _____

Open book, open notes, open J drive, open Matlab/Simulink/Python/MS Word.
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.
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

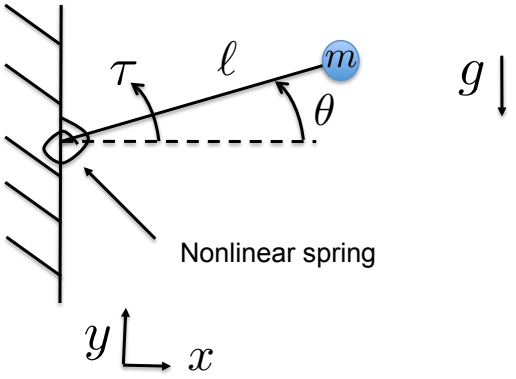
Part 1. Equations of Motion - Simulation Model

The figure below shows a point mass connected to a massless rod which is connected to the wall with a nonlinear spring and damper. The potential energy for the nonlinear spring is given by

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

The joint has friction which we will model as viscous friction with damping coefficient of $b = 0.1$. The physical parameters of the system are $g = 9.8$ meters per second, $\ell = 0.25$ meters, $m = 0.1$ kg, $k_1 = 0.02$, and $k_2 = 0.01$.

The input torque τ is limited to ± 3.0 Newton-meters.



1.1 Using the configuration variable $q = \theta$, 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. Design Models

For this section, use one of the following files to implement the simulation: `practice_final_part_2.slx` (Simulink), `practice_final_part_2.m` (Matlab), `practice_final_part_2.py` (Python).

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.

2.1 Suppose that the objective is to linearize the system around the equilibrium angle θ_e, which may or may not be zero. Find the associated equilibrium torque τ_e so that θ_e is an equilibrium of the system.

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

2.3 Linearize the model found in Part I around the equilibrium (θ_e, τ_e).

2.4 Find the transfer function of the linearized model when $\theta_e = 0$.

2.5 Find a state-space model for the system linearized around θ_e, u_e when $\theta_e = 0$. For your states use $x = (\tilde{\theta}, \dot{\tilde{\theta}})^\top$.

Part 3. PID Control

For this section, use the following files to implement the simulation: `practice_final_part_3.slx`, `ctrl_pid.m` (Simulink), `practice_final_part_3.m`, `controllerPID.m` (Matlab), `practice_final_part_3.py`, `controllerPID.py` (Python). 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 multiples the angular rate and not the derivative of the error.

3.2 Derive the transfer function from the reference input θ_r to the angle θ.

3.3 Find the proportional gain k_p such that the control input τ saturates at $\tau_{\max} = 3$ Newton-Meters when a step of 20 degrees is placed on the system.

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 ω_n and the derivative gain k_d so that the actual transfer function equals the desired transfer function, when $\zeta = 0.707$.

3.5 Using a dirty derivative, implement PD control where the input θ_r is a square wave with an amplitude of ± 20 degrees and a frequency of $\omega = 0.1$ Hertz. **Insert a plot in the Word file that shows both θ_r and θ for 20 seconds of simulation.**

3.6 Add an integrator to remove the steady state error. **Insert a plot in the Word file that shows both θ_r and θ for 20 seconds of simulation.**

3.7 Insert a copy of the control code (`ctrl_pid.m`, `controllerPID.m`, or `controllerPID.py`) in the Word document.

Part 4. Observer-based Control

For this section, use the following files to implement the simulation: `practice_final_part_4.slx`, `ctrl_obsv.m` (Simulink), `practice_final_part_4.m`, `controllerObsv.m` (Matlab), `practice_final_part_4.py`, `controllerObsv.py` (Python). 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

$$u = u_e - K\hat{x} - k_i \int (\theta_r - \hat{\theta}) d\sigma - \hat{d}$$

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

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

4.1 Find the feedback gain K that places the poles at the locations found in Part 3 and the integrator gain k_i so that the pole of the integrator is at -10.

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

4.3 Add a disturbance observer, where the pole of the disturbance observer is $p_{dist} = -1$.

4.4 Implement the observer based control and tune the controller and estimator.

4.5 Insert a plot of the step response of the system (θ and θ_r), for the complete observer based controller in the Word document.

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

4.7 Insert a copy of the control code (`ctrl_obsv.m`, `controllerObsv.m`, or `controllerObsv.py`) in the Word document.

Part 5. Loopshaping

For this section, use the following files to implement the simulation: `practice_final_part_5.slx`, `ctrl_loop.m` (Simulink), `practice_final_part_5.m`, `controller_loop.m` (Matlab), `practice_final_part_5.py`, `controller_loop.py` (Python). The sampling rate for the controller is $T_s = 0.01$.

5.1 Using the PID controller derived in the 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. Using PID control, what is the attenuation on the noise for noise with frequency content above 100 rad/sec? Using PID control, what is the tracking accuracy for reference signals with frequency content below 0.001 rad/sec.

5.2 Starting with the PID controlled plant, add a lag filter to improve the tracking accuracy for signals with frequency content below $\omega_r = .02$ rad/sec by a factor of 10.

5.3 Add a low pass filter to increase noise attenuation by a factor of 10 for noise with frequency content above $\omega_n = 2000$ rad/sec. What is the augmented controller $C(s)$?

5.4 Add a prefilter to remove any overshoot in the response.

5.5 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.

5.6 Implement the controller in simulation. **Insert a plot in the Word file that shows both θ_r and θ for 20 seconds of simulation.**

5.7 Insert a copy of the control code (`ctrl_loop.m`, `controller_loop.m`, or `controller_loop.py`) in the Word document.