# ME 417 - Num Assignment #3

**Control of Mechanical Systems - Fall 2020**

Num Assignment Due: Sun, 07 Feb 2021 23:59

Complete the following problems and submit your work as a working notebook and a saved pdf copy. *You can complete the numerical assignment using **Julia**, **Python** or **MATLAB**, and submit your work as a Jupyter Notebook (or MATLAB Livescript) + a pdf export*

Provide response plots as relevant, ensure that you label the figures, the axes, title plots and legends. Any controller design specifications given, should be met by observing the time response of the system. The Numerical Lessons provided will aid greatly in carrying out this assignment.

Collaboration is only allowed within the group members.

## Problem 1

**Rooting the Locus (25pts)**

In this problem you will design function a basic root-locus visualizer.

a. (100%) Design a function *rootlocus(G)* that takes an open-loop transfer function and returns a plot of the root-locus, showing the zeros and the open-loop poles.

By definition, the root-locus gives all the solutions of the characteristics equation $1 + KG(s) = 0$ on the s-plane for $0 \leq K < \infty$, so to plot the root-locus, you can solve for the roots of the characteristics polynomial numerically by varing the value of K and plotting the result.
You can use MATLAB's built-in pole() and zero() commands.
Helpful questions:
- How many root-locus segments do you have?
- How many arrays do you need to store the results of pole locations, as you vary K?
- Compare your result with MATLAB's built-in rlocus()

## Problem 2

**Controller Design over a Transfer Function (25pts)**

We wish to design a controller for the following system.

- $G(s) = \dfrac{30.0}{(s - 5.0)\,(s - 3.0)}$

a. Using Control System Designer, design a continuos domain PID controller to achieve the following performance specifications:

- $\%OS < 25.0\%$
- $T_s < 0.15s$
- Zero Steady-State Error

Provide a screenshot of the Control System Designer setup.

b. Test your controller by simulating the closed loop response using step()

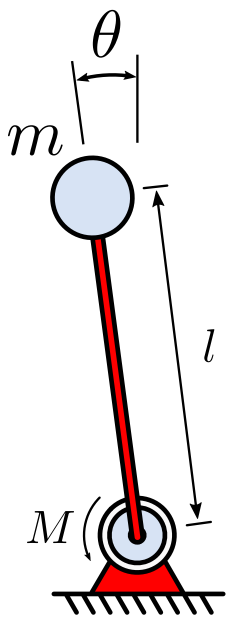
c. Discretize your controller and simulate the closed-loop system using basic numerical integration. Integrate using $\Delta t = 0.001s$

Note that you can retrieve the differential equation from the transfer function and can define a model (@xdot) from the differential equation.

## Problem 3

**Nonlinear System Control (25pts)**

Given the following inverted pendulum. Modeled as a point mass of negligible size connected to a massless rod. With a motor connected at the base, providing a moment torque.



With $m = 3kg, l = 500cm$

a. (10%) Derive the equations of motion of the system. Then linearize and derive the transfer function of the system, relating torque to angular position: $\frac{\Theta(s)}{M(s)}$.

b. (30%) Using the transfer function, design a PID controller to achieve the following specifications:

- $T_s = 1s$
- $\omega_d = 2rad/s$

c. (30%) Simulate the system with your controller, on the nonlinear model (numerical integration), and test it by using it as a regulator $(r = 0)$ for each of the following intial conditions

$\theta_o = 30^o, 60^o, 90^o, 145^o, 180^o$

Plot all the responses on the same subplot.

Plot the torque applied for each case, in a different subplot.

d. (30%) In practice, you are bounded by the actuation effort. Repeat the simulation in part. c, but limit the magnitude of $u$ (which is the moment torque) to $200\,Nm$. This is called a saturation limit. Discuss how you would change your PID gains after you've added this saturation limit.

## Problem 4

**Controller Robustness (25pts)**

In this problem you will design a controller for a given system, then you will test the controller on the system with varying parameters. To assess the robustness of the controller.

Given the following DC Motor transfer function, relating voltage to angular velocity:

$$G(s) = \frac{\Omega}{E} = \frac{K_t s}{K_b K_t s + (Ds + Js^2)(L_a s + R_a)}$$

With $L_a = 1H, R_a = 9\Omega, K_t = 2, K_b = 1, D = 0.5N - s - m/rad, J = 3kg \cdot m^2$

a. (30%) Design a PID position controller to achieve the following performance specifications to a step input:

- $T_s = 0.75s$
- $\omega_d = 6rad/s$
- Zero Steady-State Error

b. (30%) Run a "Monte Carlo" simulation with $n = 100$ samples (simulations) while varying some parameters. Specifically, add Gaussian, zero-mean, random noise with specific standard deviations to some parameters:

- Add Gaussian noise with $\sigma = 0.2$ to $L_a$
- Add Gaussian noise with $\sigma = 1$ to $J$
- Add Gaussian noise with $\sigma = 0.25$ to $D$
- To add normally distributed random noise with standard deviation $\sigma$ in MATLAB, you can do : $\sigma * randn()$
- Remember that the noise values need to be regenerated on every iteration.

c. (20%) Measure the average error in overshoot by comparing between the base case and the simulations. Also capture the maxium overshoot deviation.

d. (20%) Tune your initial PID controller gains, such that the average overshoot deviation in the simulation in (b) does not exceed 30%