Assignment 4: Introduction to Tracking Controllers

Tufts ME 149: Optimal Control For Robotics

Assigned: Feb 6 — Due: Feb 15

Part One: Cubic Spline Math

Suppose that you are given a single segment of a cubic spline, $x(t)$, shown below.

$$x(t) = \sum_{i=0}^{3} c_i t^i = c_0 + c_1 t + c_2 t^2 + c_3 t^3 \tag{1}$$

1a: Compute the spline coefficients c_i in terms of the boundary conditions shown below.

$$x(0) = x_0 \qquad x(h) = x_h \tag{2}$$

$$\dot{x}(0) = v_0 \qquad \dot{x}(h) = v_h \tag{3}$$

1b: Compute the acceleration $\ddot{x}(t)$ at each boundary ($t = 0$ and $t = h$) in terms of the boundary conditions.
1c: Compute the value of the "minimum-jerk" objective function for this segment in terms of the boundary conditions. Note that "jerk" is the technical term for the time-derivative of acceleration.

$$J(x(t)) = \int_0^h \dddot{x}^2(\tau)\, d\tau \tag{4}$$

Note: You may perform these calculations either by hand or using the Matlab symbolic toolbox. If you do the calculations by hand, then please scan[1] your solution and upload it as a pdf. If you choose to use the Matlab symbolic toolbox then upload your derivation script. The script should print the solution to the command prompt; please copy this output directly into the comments of the script before submitting. Your file should be named either:
`hw_04_studentName_splineMath.pdf`
`hw_04_studentName_splineMath.m`

Part Two: Tracking Controller Design

For this part of the assignment you will design a reference trajectory and implement a tracking controller for two model systems: a simple pendulum and a double pendulum. In both cases you will construct a reference trajectory that performs a swing-up maneuver, moving the pendulum from the state of minimum potential energy to maximum potential energy.

kinematic reference trajectory design

For both systems you will be given boundary conditions: the state of the system at the initial and final time. Your job is to construct a piece-wise polynomial (PP) reference trajectory that satisfies those boundary conditions. The easiest way to do this is to construct a clamped cubic spline for the position (angle) reference using the Matlab `spline` command (see the help file for how to specify boundary slopes). You can then use the `ME149/codeLibrary/splines/ppDer` function in the code library to construct the velocity (anglular rate) and acceleration (angular acceleration) reference splines by differentiating the position spline.

[1] There are several phone apps that will easily scan a document and save to pdf.

Use at least two segments in your spline (minimum of three knot points). Later in the class we will learn to select the position at the intermediate knot points using trajectory optimization. For now, you can just select them by hand.

Later in the assignment you will compute an objective function to evaluate the integral of the feed-forward torque squared along the reference trajectory. Experiment with your reference trajectory to see if you can modify the intermediate knot points to reduce the overall trajectory cost. The student who achieves the lowest overall objective function value (and has a correct solution) will receive two bonus points on this homework assignment. There will be one winner for the simple pendulum and another for the double pendulum. The maximum score on the assignment will be capped at 50 points.

feed-forward torque calculation

Both the simple and double pendulum models include a function to copmute the inverse dynamics for the system. You can call the inverse dynamics inside of your tracking controller to compute a feed-forward torque term.

feed-back controller design

The form of your controller will look something like:

$$u_{ref} = \text{invDyn}\big(x_{ref}(t),\, \dot{x}_{ref}(t),\, \ddot{x}_{ref}(t)\big) \tag{5}$$

$$u = u_{ref} + K_p \cdot \big(x - x_{ref}(t)\big) + K_d \cdot \big(\dot{x} - \dot{x}_{ref}(t)\big) \tag{6}$$

This type of controller is called an *open-loop* trajectory tracking controller, because the reference trajectory does not depend on the state of the system. Notice that there is a still a feed-back term in the controller — it is used to stabilize the system about the current reference. You may use any method to select the gains K_p and K_d. Please include (as comments in your code) a description of how you selected the gains. Hand-tuning is an acceptable solution. The function `ME149/codeLibrary/control/secondOrderSystemGains` in the code library might be useful, although you are not required to use it.

objective function calculation

Compute the torque-squared objective function for you candidate reference trajectory, defined below, where T is the duration of the trajectory, $\boldsymbol{z}(t)$ is the state trajectory (position and velocity reference), and $\boldsymbol{u}(t)$ is the control trajectory (feed-forward torques). Note that n is the number of actuators; $n = 1$ for the simple pendulum and $n = 2$ for the double pendulum.

$$J(\boldsymbol{z}(t), \boldsymbol{u}(t)) = \int_0^T \|\boldsymbol{u}(\tau)\|^2 \, d\tau = \sum_{i=0}^{n} \int_0^T u_i(\tau)^2 \, d\tau \tag{7}$$

This objective function needs to be computed numerically. Unlike the solution to the dynamics, the objective function is purely dependent on time. This means that it can be computed using simple quadrature methods, such as the trapezoid rule or Simpson's rule. In Matlab, it can be computed using the `integral` command.

Once we start doing trajectory optimization, it is important to compute the path integral using the exact same numerical method as was used to evaluate the system dynamics. There is an easy way to do this: just pass the integrand of the objective function to your simulation function! For this assignment you may use either technique to numerically evaluate the objective function.

simulation of the closed-loop system

For the final part of the assignment you will run a simulation of your closed-loop system, using the tracking controller to follow your reference trajectory. You may use any simulation method that you like, such as those from your previous assignments, the solutions in the code library, or Matlab functions such as `ode45`.

Introduce a small tracking error at the start of the trajectory. This will make it easier to see if your tracking controller is effective. Choose the size of the perturbation to be 0.1 radians in each initial angle: `q0 = ppval(qRef, 0) + 0.1;` The initial rate should be zero, both for the simulation and the reference trajectory.

plots

Create one figure for each system, with a set of plots that show the angle(s), rate(s), and torque(s). Make each figure full screen and then save as a pdf with the file name:
`hw_04_studentName_simplePendulum.pdf`
`hw_04_studentName_doublePendulum.pdf`

The top row of plots should show the reference and measured (simulated) angles, the middle row should show the reference and measured angular rates, and the bottom row should show the feed-forward and measured torques. There should be one column of subplots per joint. Include axis labels, titles, and legends for each subplot. The title for the torque plots should include the value of the objective function for your reference trajectory.

simple pendulum model

The code for the simple pendulum model is in the code library: (`ME149/codeLibrary/simplePendulum/`). For this assignment you should set the parameters: `param.freq = 1.0` and `param.damp = 0.0`.

The swing-up maneuver should take $T = 3$ seconds to complete. The pendulum should start at rest in the stable equilibrium and finish at rest in the unstable equilibrium.

$$q(0) = 0 \qquad \dot{q}(0) = 0 \tag{8}$$
$$q(T) = \pi \qquad \dot{q}(T) = 0 \tag{9}$$

double pendulum model

The code for the double pendulum model is in the code library: (`ME149/codeLibrary/doublePendulum/`). For this assignment you should set all of the parameters to one.

The swing-up maneuver should take $T = 5$ seconds to complete. Both pendulums should start at rest in the stable equilibrium and finish at rest in the unstable equilibrium.

$$q_1(0) = 0 \qquad q_2(0) = 0 \qquad \dot{q}_1(0) = 0 \qquad \dot{q}_2(0) = 0 \tag{10}$$
$$q_1(T) = \pi \qquad q_2(T) = \pi \qquad \dot{q}_1(T) = 0 \qquad \dot{q}_2(T) = 0 \tag{11}$$

Deliverables

Upload all of the Matlab code that you wrote for this assignment to Trunk, along with the pdf files for each of the two figures and your solution to the spline math problems.

The entry-point file for your two simulations should be called `hw_04_studentName.m`. Create a header block in this file (see template) that includes your name, a list of collaborators and references, how long the assignment took to complete, and concludes with a brief outline of your code.