ME 449 Assignment 5
Due 1:30 PM, November 6, 2019

1. (Ungraded problem; you don't have to turn anything in.) Finish working through Practice Exercise 8.1, which we started in class. Make sure the concepts are all clear, that you can derive the potential and kinetic energy, that you can take derivatives to get the equations of motion from the Lagrangian, and that you understand the concepts behind the mass matrix and the end-effector mass matrix and their representations as ellipsoids. I suggest you do this problem before looking at the solution, as a question such as this is likely to appear on a quiz.

2. At a particular configuration θ, the mass matrix $M(\theta)$ of a 2-dof robot arm is

$$M(\theta) = \begin{bmatrix} a & b \\ c & d \end{bmatrix}.$$

The determinant of M is $ad - bc$ and the eigenvalues of M are

$$\frac{1}{2}(a + d \pm \sqrt{a^2 + 4bc - 2ad + d^2}).$$

For this to be a valid mass matrix, what constraints must be satisfied by a, b, c, and d?

3. The UR5 is a popular 6-DOF industrial robot arm. The robot has geared motors at each joint, but in this project, we ignore the effects of gearing, such as friction and the increased apparent inertia of the rotor.

The relevant kinematic and inertial parameters of the UR5 are:

$$M_{01} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.089159 \\ 0 & 0 & 0 & 1 \end{bmatrix}, M_{12} = \begin{bmatrix} 0 & 0 & 1 & 0.28 \\ 0 & 1 & 0 & 0.13585 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, M_{23} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -0.1197 \\ 0 & 0 & 1 & 0.395 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$M_{34} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0.14225 \\ 0 & 0 & 0 & 1 \end{bmatrix}, M_{45} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0.093 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, M_{56} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.09465 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$M_{67} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.0823 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, G_1 = \texttt{diag}([0.010267495893, 0.010267495893, 0.00666, 3.7, 3.7, 3.7]),$$

$G_2 = \texttt{diag}([0.22689067591, 0.22689067591, 0.0151074, 8.393, 8.393, 8.393]),$
$G_3 = \texttt{diag}([0.049443313556, 0.049443313556, 0.004095, 2.275, 2.275, 2.275]),$
$G_4 = \texttt{diag}([0.111172755531, 0.111172755531, 0.21942, 1.219, 1.219, 1.219]),$
$G_5 = \texttt{diag}([0.111172755531, 0.111172755531, 0.21942, 1.219, 1.219, 1.219]),$
$G_6 = \texttt{diag}([0.0171364731454, 0.0171364731454, 0.033822, 0.1879, 0.1879, 0.1879]),$

$$\texttt{Slist} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & -0.089159 & -0.089159 & -0.089159 & -0.10915 & 0.005491 \\ 0 & 0 & 0 & 0 & 0.81725 & 0 \\ 0 & 0 & 0.425 & 0.81725 & 0 & 0.81725 \end{bmatrix}.$$

For your convenience, these parameters are given in Python, Mathematica, and MATLAB at http://hades.mech.northwestern.edu/index.php/Modern_Robotics#Supplemental_Information.

Your job is to write code that simulates the motion of the UR5 for a specified amount of time (in seconds), from a specified initial configuration (at zero velocity), when zero torques are applied to the joints. In other words, the robot simply falls in gravity. Gravity is $g = 9.81\text{m/s}^2$ in the $-\hat{z}_s$-direction, i.e., gravity acts downward. The motion should be simulated with at least 100 integration steps per second. Your program should calculate and record the robot joint angles at each step. This data should be saved as a .csv file, where each row has six numbers separated by commas. This .csv file is suitable for animation with the V-REP UR5 csv animation scene.

You will perform two simulations and make videos of each:

(a) The robot falling from the zero (home) configuration for 3 seconds.

(b) The robot falling from a configuration where all joints are at their zero position, except for joint 2, which is at -1 radian. This simulation should last 5 seconds.

Important: Since the simulated robot has no friction and zero motor torques, no energy is added or subtracted during the simulated motion. Therefore, the total energy of the robot (kinetic plus potential) must remain constant during the simulation. If you see the robot swinging higher and higher, or noticeably losing energy, something is wrong wih your simulation.

Also important: You are welcome to talk to classmates about concepts at the development phase, but you are not allowed to share or look at each others' code. You cannot ask to look at someone else's code, and they cannot ask to look at yours. Sophisticated AI-based software easily detects shared and altered code with common origins, so do not do it.

What exactly to turn in will be coming soon.