Homework 1

EECS/BioE C106B/206B
Robotic Manipulation and Interaction

Due: February 3, 2021

Note: This problem set includes a programming component where you will be required to use the Matlab Symbolic Toolbox. Specifically, you will answer problems 2 and 3 by filling in the provided Matlab files `SCARA.m`, `elbow.m`, and `SEA_dynamics.m`. Running these files we produce Matlab function files that implement the symbolic expressions that you define. These generated function files are then used by the autograder to verify your solution.

The autograder is provided to you in the form of an encrypted matlab file `HW1_ag.p`. You will be unable to examine the contents of this file, but can run it like any other Matlab function. It takes no arguments, and looks for your generated function files. The autograder will ask for your SID and will generate a text file called `submitThis.txt`, which will summarize the autograder results in an encrypted format. You should **submit this text file to the Gradescope assignment** called `Homework 1 (MATLAB)`. Your score for problems 2 and 3 will come from this autograder.

You should also submit a PDF with your solutions to the other problems to the Gradescope assignment called `Homework 1`.

Problem 1. Cartesian Control of a Robot Manipulator

Note: This question will help you with project 1.

(a) Let's define a robot arm with spatial frame S and tool frame T. We can measure the end effector configuration at given time as g_{ST}. We want the end effector to track a trajectory $g_{SD}(t)$, where D is the "desired frame", or the ideal location of T. We consider g_{TD} the tracking error and want to design a controller to minimize it.

  (i) How do you define the transformation g_{TD} as a function of g_{ST} and g_{SD}?

  (ii) Let ξ_{TD} be the twist corresponding to the error configuration g_{TD}. In other words, $e^{\hat{\xi}_{TD}} = g_{TD}$. How would you go about computing ξ_{TD}?

  (iii) Interpret this twist as a rigid body velocity. In which reference frame is this velocity expressed?

  (iv) This twist is the direction which will reduce the end effector error. However, we need to convert it to a velocity in the S frame, V_{ST}^{S}. How can we do so?

  (v) We would like the end effector to perform this velocity, V_{ST}^{S}. How can we compute the set of joint velocities that achieve this?

(b) Let's examine the problem of discretizing smooth trajectories in $SE(3)$.

  (a) First, imagine you have a smooth point-trajectory $p(t) \in \mathbb{R}^3$ that we wish to track with some control law. Say we are given this trajectory in discrete form, with discretization time-step Δt. In other words, we are given a sequence of points $\tilde{p}(n) = p(n\Delta t)$. We would like associate a velocity vector $\tilde{v}(n)$ with each timstep. If we had a symbolic expression for the original smooth trajectory $p(t)$, then we would simply differentiate it and set $\tilde{v}(n) = \dot{p}(n\Delta t)$. In the absence of such an expression, given only $\tilde{p}$, how would you go about approximating $\tilde{v}(n)$?

  (b) Now, imagine you have a smooth trajectory of rigid body configurations $g(t) \in SE(3)$. Once again, we only have access to a discrete version of the trajectory, $\tilde{g}(n) = g(n\Delta t)$. We wish to associate a *body frame velocity* $\tilde{V}^b(n) \in \mathfrak{se}(3)$ with each timestep. Again, if we had access to the derivative of g, we would simply compute the true body velocity $V^b(t) = g^{-1}\dot{g}$ and then discretize it to get $\tilde{V}^b(n) = V^b(n\Delta t)$. In the absence of this derivative, how would you go about approximating this velocity to generate $\tilde{V}^b(n)$? Make sure the approximation you use generates a valid member of $\mathfrak{se}(3)$.

  Hint: Re-visit the formulation of rigid body velocities presented in 106A homework 6 problem 3.

Problem 2. Kinematics and Jacobians

For this question you will be editing `elbow.m` and `SCARA.m`. For each of the manipulators shown in figures 1 and 2:

(a) Calculate the forward kinematics map $g(\theta_1 \cdots \theta_n)$ of each manipulator as a function of θ using the Matlab Symbolic Toolbox.

(b) Calculate the spatial and body Jacobians of each manipulator as a function of θ using the Matlab Symbolic Toolbox. (*Hint: The* ***`jacobian`*** *function won't work for this*)

(c) Use Matlab's `matlabFunction` utility to turn all three of these symbolic expressions into `.m` files for each manipulator and then run the provided autograder. To run the autograder run `HW1_ag.p` and submit the .txt file it creates to Gradescope (the generated file should be called `submitThis.txt`).

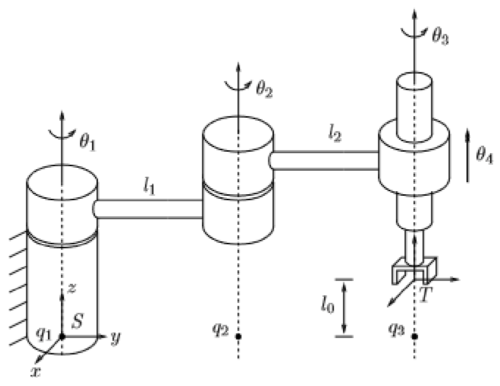

Figure 1: SCARA Manipulator

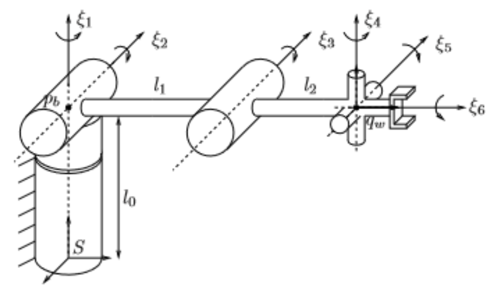

Figure 2: Elbow manipulator

Problem 3. Dynamics of Serial Elastic Actuators

For this question you will be editing `SEA_dynamics.m`. Figure 3 depicts a two DOF manipulator with Series-Elastic Actuators (the same actuators in Baxter and Sawyer's joints). Each joint contains a motor (θ_1 and θ_3) and a connected link (θ_2 and θ_4). In rigid robots, the motor is mechanically attached to the arm so their states are the same. In series elastic actuators however, the arm is connected to the motor output with a *torsional* spring, and thus they can move independently of one another. The motors have inertias J_1 and J_2. The arms have mass m_1 and m_2, moments of inertia I_1 and I_2, and lengths L_1 and L_2. The springs have torsional spring constants K_1 and K_2 respectively. Solve the following problems:

(a) Derive the equations of motion the system shown below using the Euler-Lagrange equations and the Matlab Symbolic Toolbox.

(b) Now imagine that there is a damper connecting each motor to its arm, in addition to a spring. The dampers have viscosity B_1 and B_2. Derive the equations of motion for this modified system.

When you're done, use Matlab's `matlabFunction` utility to turn these symbolic expressions into `.m` files and then run the provided autograder. To run the autograder run `HW1_ag.p` and submit the .txt file it creates to Gradescope (the generated file should be called `submitThis.txt`).

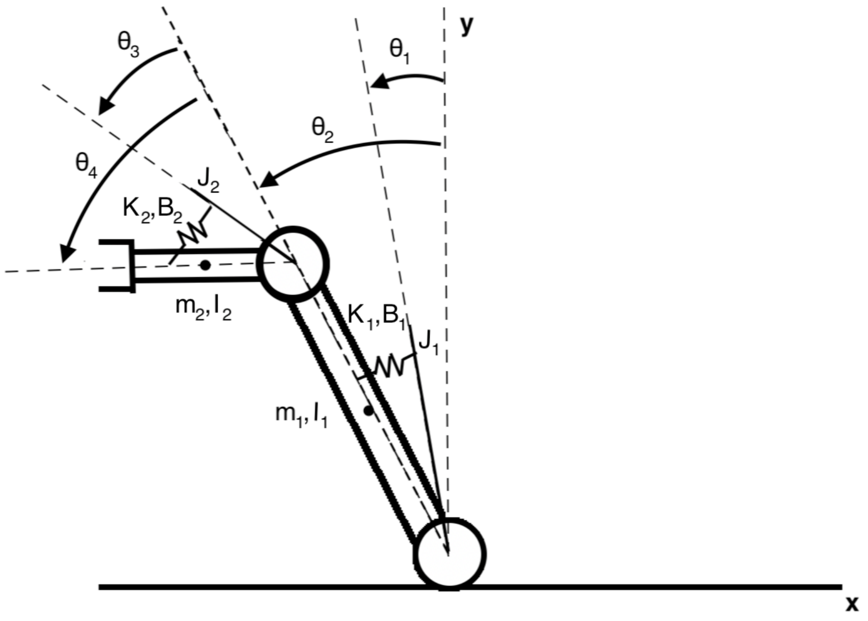

Figure 3: SEA Diagram

Problem 4. Inverse Kinematics

Read *Kinematic Control of Redundant Robot Manipulators*, by Bruno Siciliano, which is posted on the website and Piazza, and answer the following questions:

(a) Describe Task Space Augmentation in 2-3 paragraphs. Describe a case in which it would be useful, and describe what secondary task the algorithm would solve. Do not choose an example from the paper.

(b) Sciavicco and Siciliano as well as Slotline and Yoerger independently proposed using the transpose of the Jacobian matrix rather than the pseudoinverse. Here, we assume that the workspace configuration of the robot is given as a vector $x \in \mathbb{R}^6$ (in some minimal representation). Additionally, the jacobian we consider here will be the jacobian relative to this minimal representation. In other words, assume you have efficient access to the jacobian $J(q)$, which relates jointspace and workspace velocities (in this minimal representation) as

$$\dot{x} = J(q)\dot{q}$$

Let the forward kinamtics map of the manipulator be f. Imagine that you're trying to find an inverse kinematics solution $\hat{q}$, where $f(\hat{q}) = \hat{x}$. You seed your algorithm with some $q = q_0$, and evolve q using the following equation:

$$\dot{q} = AJ^T(q)(\hat{x} - x) \tag{1}$$

where A is a positive definite Hermitian matrix and $x = f(q)$.

(i) Define the workspace error to be $e = \hat{x} - x$. Show that under the evolution of equation 1, the error is governed by the dynamics

$$\dot{e} = -J^T AJe$$

(ii) Using an appropriately chosen quadratic Lyapunov function, show that under appropriate assumptions on the Jacobian, the workspace error converges asymptotically to zero. What assumptions must be made about the Jacobian, and why are they reasonable?

(iii) Hence argue that by allowing the evolution in equation 1 to continue till convergence, we can find an inverse kinematics solution for the desired state $\hat{x}$.

(c) Siciliano defines x as a parameterization of your task space. If you're doing inverse kinematics with the end effector configuration as your task space, what's the problem with using this algorithm?