ECE4560 - Homework #13

Due: Dec. 1, 2017

Problem 1. (30 or 35 pts) Let's continue working with the kinematically sufficient manipulator shown in Figure 1. We will consider only the positioning of the end-effector and not concern ourselves with the orientation. As such, the manipulator becomes a redundant manipulator because there are three joint angles for the two degrees of freedom of planar position. Furthermore, we can consider the position part of the manipulator Jacobian only, without requiring a transformation to body or spatial coordinates (since rotations are ignored).

The following trajectory

$$(x(t), y(t)) = (1 - \frac{1}{2}\cos(t), -\frac{3}{4} + 0.49\sin(t))$$

comes close to the boundary of the manipulator workspace, hence it approaches a singularity. In fact, increasing the major axes of the ellipsoid by the right amount will cause the system to cross the singularity. The initial condition of the joint configuration for this trajectory is $\alpha_i = (-0.2987, -2.4189, 1.1468)$.

(a) Using resolved rate trajectory generation with the pseudo-inverse, solve for the joint configuration trajectory for $t \in [0, 2\pi]$. Plot the joint configuration as a function of time. Also plot the (x,y) position of the end-effector as a function of time.

(b) Do the same as above, but with the damped pseudo-inverse, for two values of the damping (say $\rho^2 = 0.05, 0.005$).

(c) Compare the three trajectories. Where is the final end-effector position for each of the trajectories?

(*) For extra credit, tell me where in the trajectory the curve is closest to the singularity. How did you find it?

(x) If you want to see what purpose the damped pseudo-inverse plays, try changing the trajectory to have 0.5 instead 0.49 and integrate using resolved rate trajectory generation. It should crap out if the standard pseudo-inverse is used (at least it does for my code), whereas it won't if the damped pseudo-inverse is used. You don't have to turn this in, it's just a little note of exploration for your personal satisfaction or not.

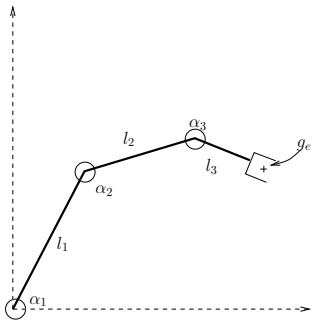

Figure 1: Planar 3R Manipulator.

Problem 2. (35 pts) Let's work here with the planar RRRP manipulator, whose constant link lengths are $l_1 = 1$ and $l_2 = \frac{2}{3}$. This manipulator may have been explored in class, with the main difference over the main homework planar manipulator is the addition of the prismatic joint. Within the dextrous workspace, this manipulator is redundant, which is what this homework will explore. Here, trajectories with different joint histories will be generated though they will have the same end-effector history. Imagine that you are interested in actually using the manipulator to perform some task (in some virtual manner).

(a) What is the manipulator body Jacobian as a function of $\alpha = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)$?

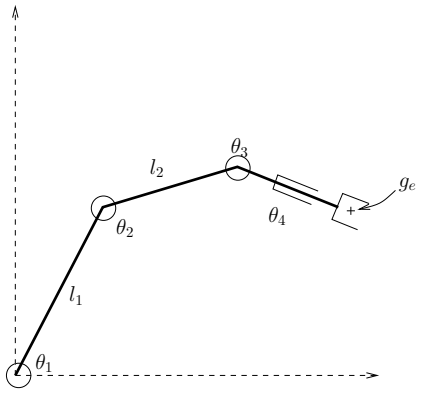

Figure 2: Kinematically redundant manipulator.

(b) Suppose that the initial joint configuration is $\theta_i = (\pi/4, 5\pi/6, -\pi/4, 1.5)^T$ and that the manipulator will flow according to the body velocity $\xi = (-0.415, -2.263, -1)$ for $\tau = 2.618$ seconds. Using resolved rate trajectory generation with the pseudo-inverse, what is the resulting end-effector configuration at the end of the trajectory? What is the associated final joint-configuration?

(c) Do the same as in part (b), but now consider the weighted pseudo-inverse where

$$W = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 20 \end{bmatrix}.$$

Verify that the final end-effector configuration and joint configuration match the solution in part (b).

(d) Plot the joint configuration trajectories for both solutions and compare them? What about the end-effector trajectories, are they the same?

Problem 3. (15 pts) Consider a task where the planar 3R manipulator has to push on an object as in Figure 3. The end-effector is supposed to apply a certain wrench to the rigid body it is currently touching. The rigid body center of mass is located at $g = (-3, 1, -\pi/4)$, and the end-effector frame is located at $h = (0, 1/8, -\pi/4)$, with respect to the frame g.

(a) Suppose that the wrench applied is a pure force, $F_e = (1, 0, 0)$, located at the origin of the end-effector frame. What is the equivalent wrench, F_c, applied at the origin of the center of mass frame?

(b) What if the task objective were given at the the incorrect point? The objective is to apply the wrench $F_c = \left(-3/(4\sqrt{2}), 1/(4\sqrt{2}), 3/(32\sqrt{2})\right)$ at the center of mass. What should the corresponding wrench be that the end-effector applies?

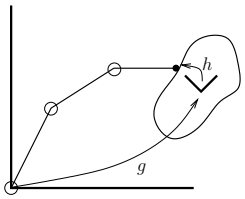

Figure 3: Kinematically sufficient manipulator.

Problem 4: Manipulator. (20 pts) For this problem, attempt to generate a trajectory from the initial g_i from the previous spline problem that moves with a constant body velocity ξ to the final location g_f in the same amount of time as given for the spline problem. Since the initial joint values α were known, the start condition is known. The trajectory should then flow according to the velocity ξ for the allotted time. Turn in the value ξ computed.

Using the past resolved rate trajectory generation code for `piktul` as an example, add the necessary functions that will do so for the `lynx6`. Call the function `genTrajectory` where it is assumed that some kind of $SE(3)$ function of time and body velocity function of time is given in a structure built with `traj.gvec`, `traj.gdot`, and `traj.tspan`. In reality, you will probably only need the body velocity structure field, not the configuration structure field. The output of the system should be a set of joint angles versus time that will move the arm to follow the specified trajectory. It can be in the output format of `ode45` and should be in a structure with the fields as per: `jtraj.alpha` and `jtraj.time`.

You will also need to write a function called `Jacobian` that computes the (linear and angular) body velocity of the end effector as a function of the α coordinates. For completeness, you should include all α coordinates in it (the last one will just have a zero column since it doesn't contribute to movement, it just opens/closes the gripper). For sure you will also reuse the `followJointTrajectory` function to make it happen. The script you wrote for the last homework can be recycled and modified to work for this problem.

Since this manipulator is kinematically insufficient, it most likely won't be able to follow the exponential trajectory. You can see this by printing the original final end-effector configuration $g_e(\alpha_f)$ and the one from the resolved rate trajectory $g_e(\alpha(t_f))$. Compute their difference by doing $g_{err} = (g_e(\alpha(t_f)))^{-1} g_e(\alpha_f)$ and then taking the logarithm to get ξ_{err}. Turn in all of the quantities described here. What is the norm of ξ_{err}? That gives an indication of how close the two are. You will also have to demo the final generated trajectory.

Tips: You can always use the `lynx6_display` function to test out the code. If you can get the resolved rate code working to generate the joint angles versus time, then using `lynx6_display` in a loop will display it as a movie in a Matlab figure. Remember to include the `drawnow` command at the end of the loop to force the figure to refresh.

Note: Probably you will notice we are in the kinematically insufficient case, for which I said one should never try to do the psuedo-inverse since it will be wrong always. That is the case here, regarding being wrong. Probably I should just be doing resolved rate trajectory generation with the end-effect position and ignoring the orientation, or making sure that my desired trajectory is legit. For the sake of this problem, let's just ignore this untidy fact and pretend it's all good. The extra calculations at the end are to see how wrong the final configuration is from the one that it should be (the elbow manipulator would be able to follow the trajectory no problems since it is kinematically sufficient).

Problem 4: Turtlebot. (30 pts) Moving along in the Sensing Part 2 module, work out the remaining bullets (5 and 6). These have the Turtlebot using the registered depth image to snag the actual distance to the target in the camera's (or maybe the turtlebot's) frame. It should be not the distance to the target, but rather the coordinate value of the target along the coordinate axis that aligns with the optical axis of the camera. Demonstrate that you can follow the blob around the halls or room and properly regulate the range.

Problem 4: Biped. (40 pts) Work out the next step in the biped project.

Problem 4: Custom Project. (50 pts)**.** If you are working on a group project, then your group contact will review the submissions associated to the prior tasks and provide a new set of tasks. You should work out whatever baby steps this group contact assigns the group.

The Manipulator Jacobian, and Pseudo-Inverse Formulas. The manipulator body Jacobian related the joint velocities to the resulting body velocity,

$$\xi^b = J_{body}(\alpha)\dot{\alpha}.$$

To get the joint velocities from a desired body velocity, the equality must be inverted. If it is full rank and invertible, then this does not pose a problem. When the matrix is not square, then there is not a unique inverse. One possible choice of inverse is the pseudo-inverse, whose form varies according to the matrix dimensions. For redundant manipulators, the pseudo-inverse is:

$$J^{\#}_{body}(\alpha) = J^T_{body}(\alpha)\left(J_{body}(\alpha)J^T_{body}(\alpha)\right)^{-1}.$$

If the manipulator must go near a singularity, then the damped pseudo-inverse can avoid blow-up near the singularity,

$$J^{\#}_{body}(\alpha) = J^T_{body}(\alpha)\left(J_{body}(\alpha)J^T_{body}(\alpha) + \rho^2 I\right)^{-1}.$$

Finally, if priority is to be given to different joint angles, then the inverse that will minimize $\dot{\alpha}^T W \dot{\alpha}$ is given by

$$J^{\#}_{body}(\alpha) = W^{-1}J^T_{body}(\alpha)\left(J_{body}(\alpha)W^{-1}J^T_{body}(\alpha)\right)^{-1},$$

for W a positive-definite symmetric matrix. Using these equations, one could extrapolate a damped, weighted pseudo-inverse which would give priority to certain joints and also avoid blow-up at singularities.

Matlab's `pinv` function can implement the damped pseudo-inverse with the proper optional arguments.

Dealing with Positions Only. When dealing with position only and not the orientation of the end-effector, then one can use only the position part of the forward kinematics, denoted by $p_e(\alpha)$. The Jacobian only needs to be the standard manipulator Jacobian for the position forward kinematics,

$$J(\alpha) = D_\alpha p_e(\alpha) \quad \text{so that} \quad \frac{\mathrm{d}}{\mathrm{d}t}p_e(\alpha(t)) = J(\alpha(t))\dot{\alpha}(t).$$

The rest follows from this. If you have any questions, feel free to come by my office.

SE(2) and the Adjoint. The adjoint operation in $SE(2)$ applied to the vector form of the twist, ξ is

$$\mathrm{Ad}_g\xi = \left[\begin{array}{c|c} R & \mathbb{J}p \\ \hline 0 & 1 \end{array}\right], \qquad \text{for } g = (p, R).$$

Generating the Trajectory. Recall that after (pseudo)inversion, you will have a differential equation

$$\dot{\alpha}(t) = J^{\#}_{body}(\alpha(t))\xi^b_{des}(t), \quad \alpha(0) = \alpha_0,$$

with a known initial condition (the current joint configuration). Resolved rate trajectory generation takes this and numerically integrates it to solve to the joint trajectory. The numerical integration can be done with `ode45` or some equivalent method. The actual function that computes the derivative $\dot{\alpha}$ will need to invoke code associated to the pseudo-inverse and to the manipulator Jacobian. It should also have access to the desired end-effector body velocity.