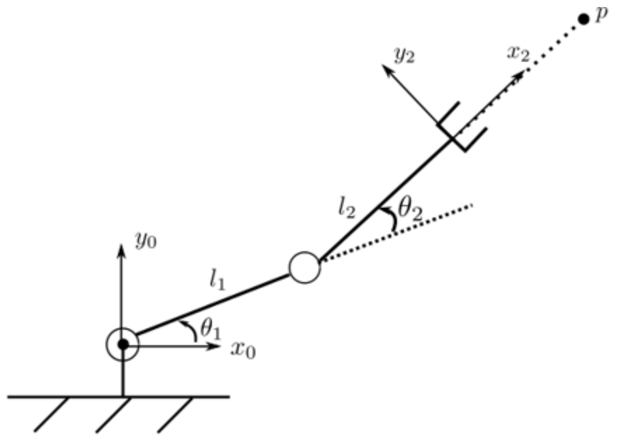
COMS W4733: Computational Aspects of Robotics

Homework 2

Due: February 25, 2019

Problem 1 (15 points)

Suppose we take our planar RR arm and attach a laser to the end effector. The laser points directly outward along the direction of the second link. We want to hit an arbitrary point p in the plane using the laser.



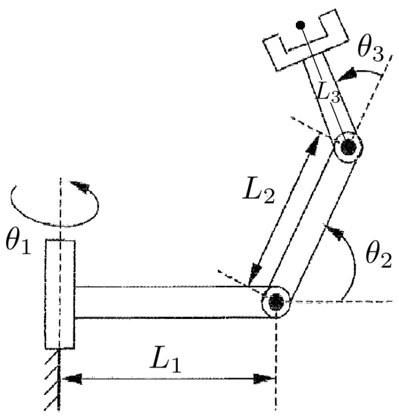
(a) How many solutions exist for the inverse kinematics problem of solving for the two joint angles θ_1 and θ_2 if we do not care about the angle at which the laser hits p? Does this depend on the relative link lengths l_1 and l_2?

(b) Suppose that in addition to making sure that the laser passes through p's coordinates $(p_x, p_y)^T$, we also want it to come in at an angle ϕ. Solve for the solutions of the joint angles assuming that both p and ϕ are reachable.
Hint: Think about modeling the laser with a third joint. You don't need to show all work in deriving any new forward kinematics or the algebra of solving the inverse kinematics, but make sure to justify your steps.

(c) Explain when no solutions exist to the above problem relative to the robot's workspace.

Problem 2 (15 points)

For this problem it may be helpful to review the differential kinematics example of the anthropomorphic arm (3.2.2 and 3.3.3 in SSVO). Consider the modified non-planar RRR arm below.



Assuming a reference configuration where $\theta_1 = \theta_2 = \theta_3 = 0$ and the base x axis points to the right, the position forward kinematics can be found to be the following:

$$
\begin{aligned}
x &= (L_1 + L_2 c_2 + L_3 c_{23}) c_1 \\
y &= (L_1 + L_2 c_2 + L_3 c_{23}) s_1 \\
z &= L_2 s_2 + L_3 s_{23}
\end{aligned}
$$

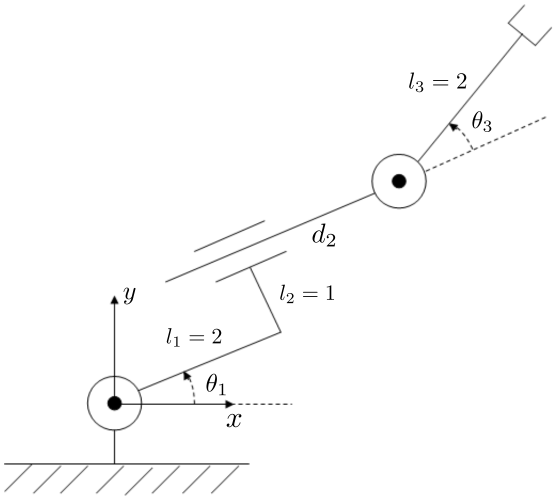
(a) Derive the full 6×3 Jacobian matrix $\mathbf{J}$ for the manipulator.

(b) Compute the determinant of the linear velocity Jacobian $\mathbf{J}_P$ and simplify it to find a general expression for when singularities occur. You may use a program to help with the simplification.

(c) Explain how the expression you derived yields both elbow and shoulder singularities. Draw examples of both for this RRR manipulator.

Problem 3 (25 points)

Consider the planar offset RPR manipulator below.



The forward kinematics are as follows:

$$
\begin{aligned}
x &= (l_1 + d_2) c_1 - l_2 s_1 + l_3 c_{13} \\
y &= (l_1 + d_2) s_1 + l_2 c_1 + l_3 s_{13}
\end{aligned}
$$

(a) Derive the full 6×3 Jacobian matrix $\mathbf{J}$ for the manipulator.

(b) While $\mathbf{J}$ is not square, it should only have three out of six non-zero rows. Find the singularities for this manipulator corresponding to these three directions of motion.

(c) Suppose that the manipulator is at the configuration $\overline{\mathbf{q}} = (\theta_1, d_2, \theta_3)^T = (30°, 2, 30°)^T$. We want to achieve the workspace velocities $(\dot{x}, \dot{y})^T = (-1, 2)^T$. Is this problem overconstrained or underconstrained? Use the appropriate pseudoinverse to numerically find the "best" solution. What criterion does the pseudoinverse optimize?

(d) Find all possible solutions to the above problem by including the homogeneous solution. You should have something of the form $\dot{\mathbf{q}} = (\dot{\theta}_1, \dot{d}_2, \dot{\theta}_3)^T = \dot{\mathbf{q}}^* + \mathbf{P}\dot{\mathbf{q}}_0$, where $\dot{\mathbf{q}}^*$ is the "best" solution from the previous part.

(e) Now suppose that, at the same configuration $\overline{\mathbf{q}}$ from before, we want to achieve the workspace velocities $(\dot{x}, \dot{y}, \dot{z}, \omega_x, \omega_y, \omega z)^T = (-1, 2, 1, -3, 0, -2)^T$. Is this problem overconstrained or underconstrained? Use the appropriate pseudoinverse to numerically find the "best" solution. What criterion does the pseudoinverse optimize?

(f) From the joint solution you found in the previous part, what are the actual end effector velocities? Which velocity components is the manipulator unable to achieve?

Problem 4 (15 points)

We want a joint space trajectory for a revolute joint starting from rest at position $q_i = 0$ rad at time $t_0 = 0$ seconds and reaching position $q_f = 2$ rad at time $t_f = 2$ seconds. However, the joint should have **final velocity** 1 rad/s upon reaching q_f at time t_f.

(a) Suppose we specify the desired acceleration magnitude $\ddot{q}_c = 2$ rad/s^2. Compute a trajectory that contains a linear segment with parabolic blends (LSPD) to satisfy these requirements. In other words, the trajectory should have a trapezoidal velocity profile. Plot, either by hand or a program, the resultant position, velocity, and acceleration profiles.

(b) Suppose we specify the cruise velocity instead: $\dot{q}_c = 1.5$ rad/s. Find the new resultant joint trajectory and plot the position, velocity, and acceleration profiles.

Problem 5 (30 points)

For this problem you'll be working with the FRANKA EMIKA Panda arm. We will also provide a script that will simulate and move the manipulator in ROS. Your goal will be to implement the numerical inverse kinematics algorithm we discussed in class in order to get the arm to move between provided waypoints. For submission, provide both your code implementation of the tasks below as well as any requested writeups or plots in your written submission file.

(a) The robot specifications, and in particular the DH parameters, are provided on FRANKA EMiKA's Github docs (**note that the order of α and d is switched on their website!**). From these you can get the forward kinematics of the arm as you did in the last homework. Use the forward kinematics of the robot to implement a function to compute the Jacobian for the end effector. This function should take in as input seven joint angles and return a numerical

6×3 matrix. As a sanity check, at the joint configuration $\mathbf{q}_i = (0.1, 0.2, 0.3, -0.4, 0.5, 0.6, 0.7)^T$ rad the Jacobian should look like

$$\mathbf{J}(\mathbf{q}_i) = \begin{pmatrix} .0409 & .0409 & .0634 & 0.333 & -.0495 & -.0324 & -.0624 \\ -.0815 & -.0815 & 0.0196 & -.0074 & -.0003 & 0.0571 & -.0203 \\ 0 & 0 & .0900 & -.0978 & -.0385 & -.0147 & 0.0586 \\ 0 & 0 & -.296 & 0.282 & -.0832 & -.210 & 0.481 \\ 0 & 0 & 0.955 & 0.0873 & -.990 & 0.130 & -.849 \\ 1 & 1 & 0 & 0.955 & 0.115 & 0.969 & 0.219 \end{pmatrix}.$$

(b) At the joint configuration $\mathbf{q}_i$, the end effector is in the initial pose

$$\mathbf{T}_e^0(\mathbf{q}_i) = \begin{bmatrix} 0.515 & 0.481 & 0.709 & -.0815 \\ 0.476 & -.849 & 0.230 & -.0409 \\ 0.713 & 0.219 & -.667 & 0.399 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Now we want the arm to move to the following desired pose:

$$\mathbf{T}_e^0(\mathbf{q}_d) = \begin{bmatrix} -.781 & -.474 & 0.407 & -.0206 \\ -.220 & 0.818 & 0.531 & -.147 \\ -.585 & 0.325 & -.743 & 0.620 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Use the simple Jacobian transpose method to compute a sequence of joint angle configurations that would move the arm between the two poses. Save these in a comma-separated text file with seven angle values per line, one for each joint. This file can be fed into the ROS script to visualize the robot's motion.

Provide a snapshot of the robot at the goal pose, a plot of the error (magnitude is sufficient) between the end effector's current pose and the desired pose over time, and a 3D trajectory plot of the robot's end effector position. Briefly discuss the algorithm's performance and any choices you made in terms of weighting parameters or error metrics. How long does it take to converge? How smooth are the resultant trajectories? Does the algorithm run into any singularities?

(c) Try to use the Jacobian pseudoinverse in place of the Jacobian transpose in your implementation. Does the manipulator run into singularities? You should find that the solution is very unstable. Use the damped least-squares inverse instead. Again, provide the same snapshots and plots as indicated in the previous part, and provide a brief discussion of this algorithm's performance compared to the Jacobian transpose.