CS545–Introduction to Robotics

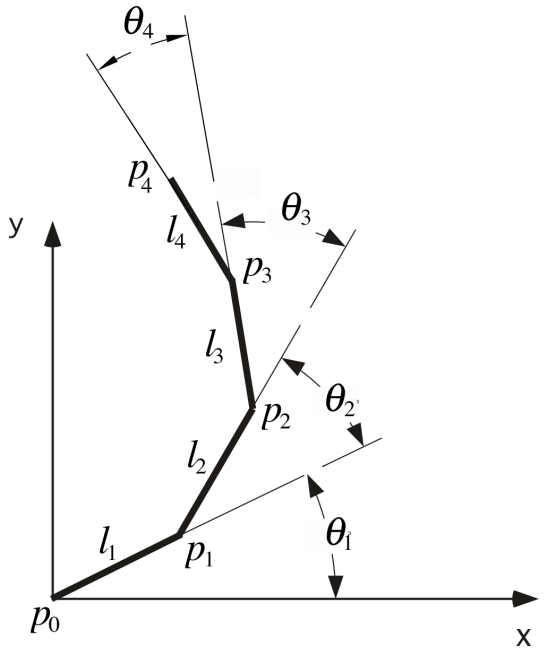
Homework Assignment 2 (Due March 31)

In the following problems, you should use MATLAB to compute numerical results and visualize the data, and Simulink for simulations. A handout about getting started with MATLAB is in

http://www-clmc.usc.edu/Teaching/TeachingIntroductionToRoboticsHomework

This web page also contains all the files needed below. IMPORTANT: In your solutions of the homework, also provide intermediate steps how you derived the solution to a problem.

1. (75 Points) On the homework web page, you find four files: kinematic_arm_simulation.mdl, ArmAnimation.m, forward_kinematics.m, and inverse_kinematics.m. This Simulink simulation is a kinematic simulation of a planar 4-joint-arm (note that a "kinematic simulation" is a simulation where no mass and inertia properties are included – it basically assumes that there is a perfect inverse dynamics model in the control loop such that desired states and current states always coincide). Details of the robot arm are in the figure below. To simplify the inertial properties of the arm, we assume that each link's mass is a point mass located at $\mathbf{p}_i$, which also means that each link's center of mass is at $\mathbf{p}_i$. The goal of this assignment will be to create inverse kinematics control for the center of mass of the entire arm, which prepares controlling the center of mass of the NAO.



a) Write down the formula for computing center of center of mass $\mathbf{p}_{cm}$ of the entire robot arm using your knowledge of the center of mass of the individual links. Assume that the mass m_i of each link is 1kg, i.e., all links weigh the same amount.

b) Write down the generic formula for the geometric Jacobian $\frac{\partial \mathbf{p}_4}{\partial \theta}$ of a system with 4 revolute joints arranged in an open-chain as shown for the 4 degrees-of-freedom system above – just assume that all the n links form an open-loop chain, where each link is of length l_i and the corresponding joint angle is θ_i with origin of the local coordinate system at p_{i-1} and joint axis $\mathbf{z}_i$. Note that you do **not** need to give the part of the Jacobian that deals with orientations. Explain the symbols in the formula. Use the notation in the figure above for this formula.

c) Expand the formula for the Jacobian in b) such that it is only a function of the joint angles and link lengths, i.e., eliminate the p_i variables

d) From your result in c), what are the Jacobians for the points $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$?

e) Given your previous results, what is the center of mass Jacobian $\frac{\partial \mathbf{p}_{cm}}{\partial \theta}$?

f) Implement this center of mass Jacobian for the 4 DOF system above in the Matlab program inverse_kinematics.m at the indicated position by using the (vector) variables "links" and "theta" in this function. Provide a printout of the program.

g) Give the general formula of the Jacobian transpose for inverse kinematics computations, and implement this method in the "inverse_kinematics.m" program. Provide a printout of your program. Run the Simulink simulation (which implements a figure-8 tracking task for the center of mass) and provide print-outs of the x-y-Graph and the joint trajectories in the "scope" block. How well does the method perform, and why does it perform this way?

h) Give the general formula of the pseudo-inverse for inverse kinematics computations, implement this method in the "inverse_kinematics.m" program. Provide a printout of your program. Run the Simulink simulation and provide print-outs of the x-y-Graph and the joint trajectories in the "scope" block. How well does the method perform, and why does it perform this way?

i) Give the general formula of the pseudo-inverse with Null-space optimization for inverse kinematics computations, and implement this method in the "inverse_kinematics.m" program — use all joint angles to be 0.5 as a desired optimization posture (see Slide 13, Lecture-9). Provide a printout of your program. Run the Simulink and provide print-outs of the x-y-Graph and the joint trajectories in the "scope" block. How well does the method perform, and why does it perform this way?

j) Using a weighted pseudo-inverse allows you changing how much each degree-of-freedom should contribute to the inverse kinematics. Assume a weight vector for the four joints of the robot as **w**=[0.1 0.2 0.3 0.5]. Derive a weighted version of the inverse kinematics of h) and repeat subproblem h) for this weighted version. How well does the method perform, and why does it perform this way?

k) You can also use the weights **w** in j) for weighting the null-space optimization criterion. Repeat the subproblem f) with this method. How well does the method perform, and why does it perform this way?

2. (50 Points) On the homework web page, you find the file: draw_task.cpp. This task implements a simple operational space reaching task with minimum jerk trajectories. You need to

add this task to your CMakeLists.txt compilation, and also in naoUser/src/initUserTasks.c. A simple point-to-point movement is currently implemented and should work. Modify this program to create a figure-8 drawing in Cartesian space. Create the figure-8 in the X-Z plane of the workspace, as big as possible, but make sure that the NAO can realize the figure-8 without violating the max/min joint space limits it has. Create phase plots of the figure-8 the same as in Homework 1.i. Provide the phase plot printouts and the printout of your modified draw_task.cpp. Comment on how the performance of accomplishing this task.