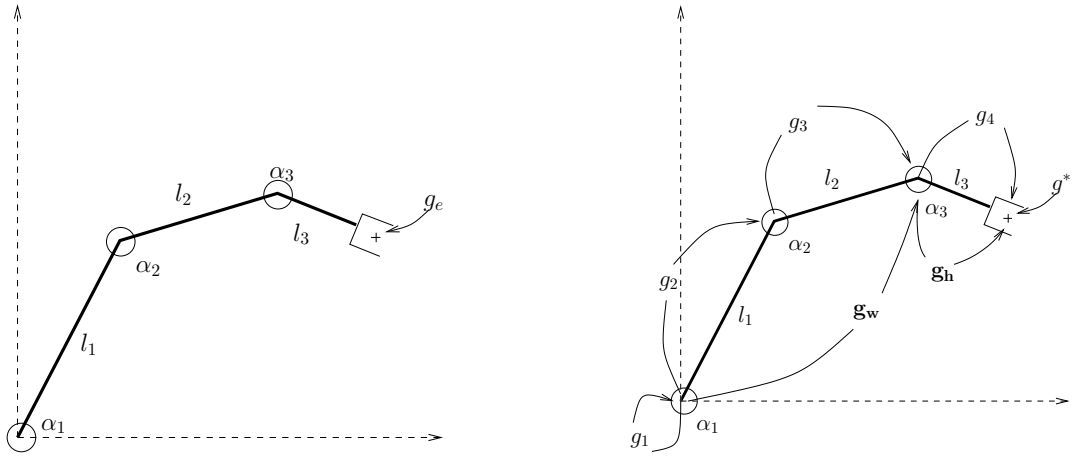
ECE4560 - Homework #9

Due: Nov. 30, 2017

Problem 1. [30 pts] Work out the inverse kinematics for the three-link rotational planar manipulator depicted in Figure 2. Since it is fully controllable, we can use a version of Pieper's solution for the plane, but use the geometric method (from the homework) for the position part.

The last joint/link combination could be considered the hand, with the first two joint/link combinations being the base to wrist transformation. If the forward kinematics are given by $g_e(\alpha) = g_1(\alpha_1)g_2(\alpha_2)g_3(\alpha_3)g_4$, then use the first two joints to solve for the position of the wrist, given by the position of the following wrist configuration $g_w = g_e^* \, g_4^{-1}$. Once this position has been solved, you can solve for α_3 in order to get the orientation to work out properly for the end-effector (don't forget that $g_3(\alpha_3)$ gets broken up into $\bar{g}_3\tilde{g}_3(\alpha_3)$).

Hopefully the drawing in Figure 1(b) helps a little bit, since it contains additional details to help you visualize what is going on. The whole point of Pieper's idea was to use the orientation control of the wrist-hand connection to define the hand joints using the desired orientation and for the base to wrist portion of the kinematic chain to be used to find appropriate position for the wrist so that the hand could actually achieve the orientation task.



(a) Manipulator

(b) Pieper's solution visualization.

Figure 1: Planar 3R Manipulator.

So, in short

(a) For link lengths $l_1 > l_2 > l_3$, what are the inverse kinematics for the manipulator depicted in Figure 2.

(b) Supposing that $l_1 = 1$, $l_2 = \frac{1}{2}$, and $l_3 = \frac{1}{4}$, what joint angles would be used to get to the configuration $g_e^* = (1.5560, 0.7288, 0.7854)$? (the angle is in radians, which would be 45.00 degrees).

Because of the way that angles work, I could also add or subtract 2π to the desired orientation and get a valid solution. You may see such behavior pop up as you work out the inverse kinematics, so don't be confused about that. This goes along with my mentioning that sometimes 2π should be added/subtracted to the angle as part of the inverse kinematics solution in order for things to work out. If you'd like to visualize and verify things, then there is a function called `planarR3_display` on the class wiki page (`pvela.gatech.edu/classes`). Follow the *Display/Plotting* link.

Problem 2. [10 pts] Complete the adjoint code for $SE(3)$ for the case of computing the adjoint of a Lie group element g and a twist ξ (e.g., Lie algebra element). Verify against the following Matlab run:

```
>> g1 = SE3([-5;6;-1] , [1 0 0;0 0 -1;0 1 0]);
>> g2 = SE3([ 0;0; 0] , [ 1/sqrt(2) 0 1/sqrt(2);0 1 0;-1/sqrt(2) 0 1/sqrt(2)]);
```

```
>> adjoint(g1, g2)

ans =

    0.7071   -0.7071         0    2.7782
    0.7071    0.7071         0    5.2929
         0         0    1.0000         0
         0         0         0    1.0000


>>g = SE3([1;0;-1],[sqrt(2)/2 0 -sqrt(2)/2 ; 0 1 0; sqrt(2)/2 0 sqrt(2)/2]);
>>xi = [0; 1; 0; pi/10; 0; pi/12];

>>adjoint(g, xi)

ans =

         0
    0.5557
         0
    0.0370
         0
    0.4073
```

If $\xi = (1, 0, 1, 0, \pi/6, -\pi/12)^T$, then what results from the adjoint operation?

The Adjoint. Recall that the adjoint operation is a linear transformation of the body or spatial velocities. It can be performed in the vector space representation or the homogeneous (matrix) representation. The homogeneous representation is easy to do, since it is just matrix multiplication. In vector representation, the left and right multiplication is not as simple, but given that the adjoint is a linear operation on vectors, it is possible to define the adjoint as a linear transformation.

The $SE(2)/\mathfrak{se}(2)$ Adjoint. In the vector representation, it has the following form:

$$\mathrm{Ad}_h\xi = \left[\begin{array}{c|c} R_h & \mathbb{J}\overrightarrow{d_h} \\ \hline 0 & 1 \end{array}\right] \left\{ \begin{array}{c} v \\ \omega \end{array} \right\} = \left\{ \begin{array}{c} R_h v + \mathbb{J}\overrightarrow{d_h}\,\omega \\ \omega \end{array} \right\}, \qquad \text{where} \qquad h = (\overrightarrow{d_h}, R_h).$$

The $SE(3)/\mathfrak{se}(3)$ Adjoint. In vector representation, it has the the following form:

$$\mathrm{Ad}_h\xi = \left[\begin{array}{c|c} R_h & \hat{d}_h R_h \\ \hline 0 & R_h \end{array}\right] \left\{ \begin{array}{c} v \\ \omega \end{array} \right\} = \left\{ \begin{array}{c} R_h v + \hat{d}_h R_h \omega \\ R\omega \end{array} \right\}, \qquad \text{where} \qquad h = (\overrightarrow{d_h}, R_h).$$

When given in vector form, the inverse of the adjoint applied to a vector is simply the inverse of the adjoint matrix applied the same vector.

Problem 3. [40 pts] Let's consider the three-link rotational planar manipulator from the previous homeworks, c.f. Figure 2. The link lengths were specified to be $l_1 = 1$, $l_2 = \frac{1}{2}$, and $l_3 = \frac{1}{4}$. Let's work out how to get trajectories in the α space given a desired trajectory in the (x, y)-position space. Here we will be ignoring the orientation. That makes our robot a kinematically redundant manipulator since the input space is three dimensions and the output space is only two.

The task here is to perform resolved rate trajectory generation. As hinted in earlier homeworks, the Jacobian is a useful little beast. It lets us maps joint velocities to position velocities:

$$\dot{p} = J(\alpha)\dot{\alpha}$$

assuming that α is a function of time. Resolve rate trajectory generation takes the Jacobian $J(\alpha)$ and inverts it. However, here the Jacobian matrix is not square, it is 2×3. There actually is still an inverse to this particular problem. It is called the pseudo-inverse and is actually implemented by Matlab (the function is `pinv`). I believe that Matlab can also use the back-slash or forward-slash (I don't use either, so I don't know which one is correct) to also solve for $\dot{\alpha}$ given $\dot{p}$. Anyhow, performing the pseudo-inverse gives:

$$\dot{\alpha} = J^{\dagger}(\alpha)\dot{p}$$

where $\cdot^{\dagger}$ is the symbol for the pseudo-inverse. Now, given an initial α and a desired end-effector velocity, we can integrate the above using `ode45` to get a trajectory for α that takes use from one point to another. Put succintly, you will have to integrate the following differential equation

$$\dot{\alpha}(t) = J^{\dagger}(\alpha(t))\dot{p}_{des}(t), \quad \alpha(0) = \alpha_0,$$

with a known initial condition (the initial joint configuration) and known end-effector position velocity (here it is constant). The actual function that computes the derivative $\dot{\alpha}$ will need to invoke code associated to the manipulator Jacobian. It should also have access to the desired end-effector body velocity (again, constant so that's not a problem).

Suppose that your initial configuration is $\alpha(0) = (-\pi/4, 2\pi/3, -\pi/5)$ and that your desired velocity is $\dot{p} = (-0.5, 0.5)$ which will be applied for 2 seconds. Perform resolved rate trajectory generation to get the $\alpha(t)$ trajectory. Plot **(a)** the resulting joint angles as a function of time, **(b)** the end-effector position as a function of time (so, you will have to map the α values through the forward kinematics for position the plot), and **(c)** a parametric plot of the manipulator's position. In the parametric plot, you should also overlay the plot of the manipulator in its initial configuration, in it's final configuration, and one to two intermediate configurations.

You can always view it as a movie using the hints and code from prior homeworks. Sample code should be on the class wiki page for ECE4560 (located at `pvela.gatech.edu/classes` and following the Display/Plotting link). The plot code for the manipulator is here too.

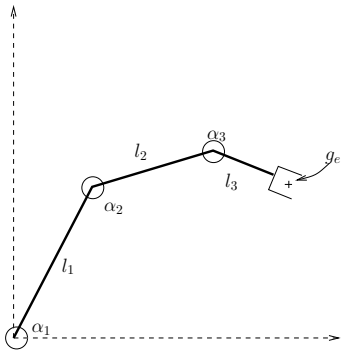


Figure 2: Planar 3R Manipulator.

Problem 4: Manipulator. [30 pts] The instructions for this lab assignment are on the class wiki in Module Set 1, Buller 4. This lab thread will continue with the piktul and look at trajectory generation using the Jacobian. The earlier resolved rate trajectory generation homework problem should serve as a template/example for proceeding here.

The trajectory should start at the initial joint configuration of:

$$\alpha_i = (1.5, -45, 60, 0, 0.5)^T$$

and move at the linear velocity

$$v_{des} = \left\{ \begin{array}{c} -0.15 \\ 0.7 \\ -0.25 \end{array} \right\} \text{inches/sec}$$

for 3 seconds. Tips, hints, and advice are provided at the wiki page for this lab module.

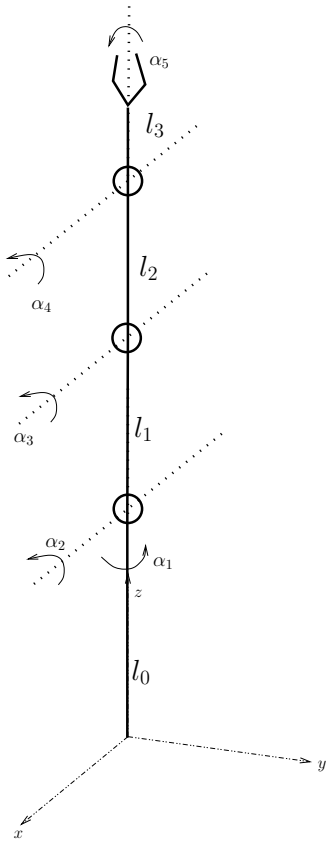


Figure 3: Lynx6 manipulator, straight up home configuration.

Problem 4: Manipulator. [20 pts] It is time to calibrate the second manipulator, the Lynx-6 manipulator. It is a more typical manipulator, being able to move about in $SE(3)$. The learning module and instructions for calibration can be found at the class wiki, as part of Module Set 0 (both bullets).

To make sure that all the manipulators are calibrated, the demo will involve a verification exercise. The code file should have been found with the homework document. If you wonder what the base coordinate frame should be, just look at Figure 3. That is what I, and therefore you, will be using. Code approriately.

Problem 4: Turtlebot. [40 pts] For the turtlebot groups, work out enumerated items 3 and 4 in the "Turtlebot: Sensing Part 1" module. In addition, to the "turn in place" script, create a modified version that has a subscriber and will turn to the angle defined by the subscriber. Then create a python script that takes in user input for the angle and publishes it for use by the modified turn script.

Problem 4: Biped. [40 pts] Work out the next step in the biped project.

Problem 4: Custom Lab. [40 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 as opposed to the lab problem.