ECE 383/MEMS 442 Lab 2: Forward and Inverse Kinematics

Due date: 10/5/2015

Instructions: Submit your lab2[x].py files on Sakai.

Problem 1: Forward kinematics

Lab 2a displays several configurations of a fixed-base robot and a target sphere in 3D space. The center of the robot's *end effector* is at position (0.17,0,0) relative to link 7. The target is animated using the target_motion(t) function.

The configurations should be drawn with opacity growing when the end effector is closer to the target, but currently the distance isn't being calculated correctly. Your job is to properly calculate these values.

Complete the function `lab2a(robot,q,ee_link,ee_local_position,target)` which should return the distance in workspace between the target and the world position of the end effector when the robot takes on configuration `q`. The end effector is attached to link `ee_link` and has local position `ee_local_position`.

[You will need to understand the `RobotModel` class and the `RobotModelLink` class. Please consult the Intro to Klamp't slides and the Python API documentation]

Problem 2: Analytical inverse kinematics

In Lab 2b, will see a 3R robot at its zero position, with joint axes ZYY. You will also see a point that denotes a target position that should be reached by the robot's end effector. Your job is to compute **all** the IK solutions of the robot so that the robot's end effector position lies at the target.

Your code will go into the function `lab2b(L1,L2,L3,point)`. Please consult the lecture notes and the solution for the 2R manipulator covered in Lecture 7 as part of your solution. The `math.atan2(y,x)` function will be useful here.

Make sure your solution follows the designated formatting of the output pair. The GUI will draw your solutions, which will help you debug. You will typically see either 0, 2, or 4 solutions.

Problem 3: Optimizing position and orientation

In Lab 2c you will see a single configuration of a fixed-base robot and a target cylinder in 3D space. As the target position moves, the robot's configuration should be optimized to place the end effector closer to the target, and it should also be oriented toward the target's orientation (as though it were picking it up).

We've chosen the end effector local position and axis for you, and it is your job to find a configuration that matches this to the target world position and axis. Complete the function `lab2c(robot,qseed,ee_link,ee_local_position,` `ee_local_axis,target,target_axis)` which should use the Klamp't inverse kinematics functions to compute the IK solution for this problem.

It should return a triple:

- `q`: the solution configuration
- `errpos`: The distance between the end effector's world position and the target position at the solution.
- `erraxis`: The distance between the end effector's world axis and the target axis at the solution.

[Note: the current implementation for finding `errpos` and `erraxis` is incorrect]

For more help, please consult the Klamp't IK tutorial and the documentation of the IK module in http://motion.pratt.duke.edu/klampt/pyklampt_docs/namespaceklampt_1_1ik.html

You will notice that even if your implementation is correct, many IK solve calls will fail. Why? Try to start from a random configuration by uncommenting the three lines marked "TODO:" under the `optimize` method. How does this affect the ability to find a solution? What could you do to make your IK solver more reliable?