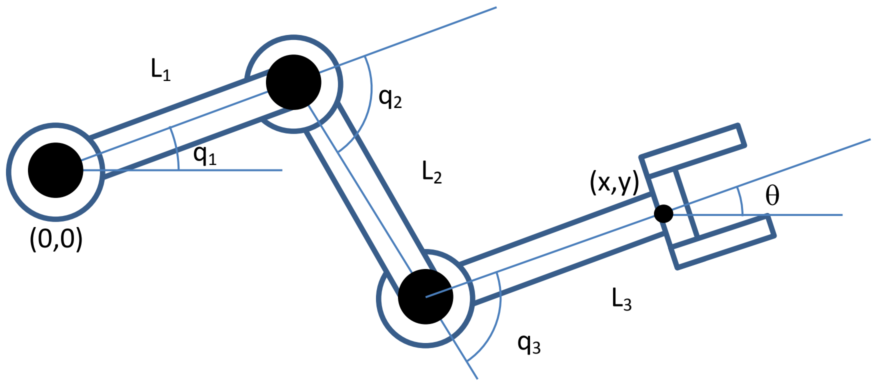
Exercise 1: Analytical IK (ex1.py)

Consider a planar, 3R manipulator as shown in the following figure.



1. Write a mathematical expression giving (x,y,θ) as a function of q_1, q_2, q_3, L_1, L_2, and L_3. You should use the shortcuts s_i=sin(q_i) and c_i=sin(c_i) to make your expression less cumbersome.
2. Devise and code a method for solving for the inverse kinematics solutions this manipulator for a given (x,y,θ). The print statements in run_ex1() will help you verify that your method works properly.

Exercise 2: Numerical IK (ex2.py)

The GUI program shows a robot with a gripper, and a white dot traveling around in a circle. This question asks you to call the appropriate numerical routines for solving the IK constraints such that the robot touches the point. Consult the documentation of the IKObjective class and IKSolver class in the robot module, or examine the API interfaces in robot/src/robotik.h.

1. Implement the GLIKProgram.solve_ik() method so that the point localpos on the given RobotLink robotlink touches the point worldpos.
2. Try initializing the IK solver with the robot's zero configuration as suggested in the comments. Qualitatively, how do the solutions differ from the basic implementation (where you use the robot's prior configuration)?
3. Implement solve_ik() so that the orientation of the given link is set to the identity matrix. How often do you observe that the technique is successful? Why? Does changing the number of iterations or the tolerance help? Why or why not?
4. Compare initializing the IK solver with the zero configuration against using the prior configuration.