Inverse kinematics and Jacobian

Exercise 1:

Modify the Matlab function “CrustInvKin.m” to be able to calculate inverse kinematics for your own CrustCrawler model.

Verify the function with your own forward kinematics function – i.e. it should do the “inverse” of your function (apart from the last rotation angle, q4).

Exercise 2:

Try to test *Resolved-rate motion control* for your own CrustCrawler model. You may use the “Crustcrawler_test.m” file.

Test the motion control on different trajectories – e.g. where the robot moves across its upright position and where it is close to the limits of its workspace.

Which positions of Crustcrawler are singularities ? Test / verify with calculation of the manipulability measure.

Exercise 3 : Inverse Kinematics on CrustCrawler

(MANDATORY EXERCISE – SEE JOURNAL REQUIREMENTS BELOW..)

Use the https://github.com/au-crustcrawler/au_crustcrawler_base/blob/master/nodes/au_dynamixel_invkin_test_node.py file as base for implementing an inverse kinematic algorithm for the Crustcrawler robot in Python and ROS. The algorithm has partly been implemented, but need some math at the point marked "# Insert code here!!!"

Test the algorithm on the real CrustCrawler robot trying both the elbow up and elbow down solutions.

Journal requirements :

(This is what you have to upload to "Crustcrawler Inverse kinematics" on Blackboard)

Basically, you should be able to understand what is going on (both in theory and practice), if you look at the journal in 5-10 years from now.

1. Drawing of Crustcrawler robot kinematic structure.
2. Description of the mathematics behind the inverse kinematics implementation.
3. Description of the most important implementation details.
4. Test results.
5. Python code attached