ENME489C/ENME808M: Problem Set 5
Inverse Kinematics

Prof. Axel Krieger
Fall 2017
Due Date: Wednesday 10/18 5 PM

October 6, 2017

In this problem set, we will study the Inverse Kinematics using a numerical approach for the same Fanuc manipulator that we studied in the forward kinematics problem set #4. For the numerical approach you will use the inbuilt numerical solvers in the Robotics System Toolbox to achieve a straight line trajectory and trajectory of your choice in the task space.

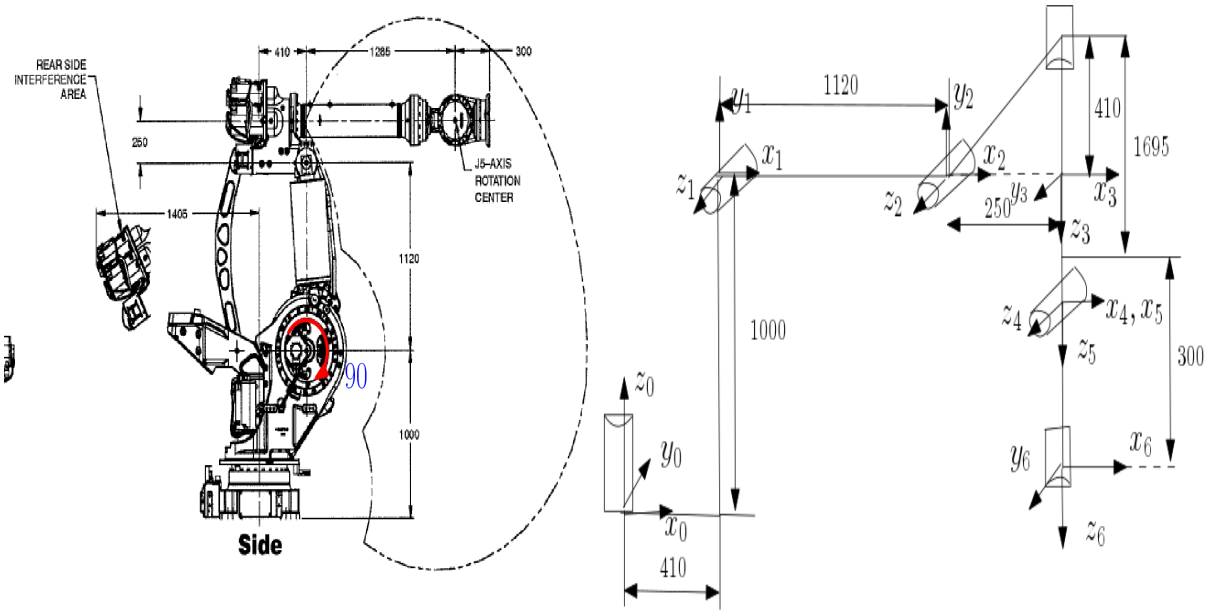
1 MATLAB Assignment

Figure 1 shows the manipulator diagram of the Fanuc M-900 and the corresponding frame assignment on the stick figure. As described earlier, the manipulator is a 6 DOF, Articulated Manipulator (R-R-R) with a spherical wrist (Z-Y-Z).

In the forward kinematics problem, the end-effector location is determined uniquely for any given set of joint displacements. On the other hand, the inverse kinematics is more complex in the sense that multiple solutions may exist for the same end-effector location. Also, solutions may not always exist for a particular range of end-effector locations and arm structures. With just the planar configuration the equations can get complicated and because the pertaining kinematic equation is comprised of nonlinear simultaneous equations with many trigonometric functions, it is not always possible to derive a closed-form solution, which is the explicit inverse function of the kinematic equation. When the kinematic equation cannot be solved analytically, numerical methods are used in order to derive the desired joint displacements.

Reusing your implementation from the Problem Set #4 your task in this problem set is to write three more functions, **ik_fanuc_m900.m**, **line_trajectory_generator.m**, and **my_trajectory_generator.m**. Using your implementation of the inverse kinematics solution, you will make the manipulator first sketch a straight line trajectory and later a more complex shape. Details of these functions are as follows:

1. **line_trajectory_generator.m**: This function creates a discretized cartesian line trajectory. It returns an array of points along the line. This function takes start and end points of the line segment as the arguments, and computes points on this line with a given resolution or point spacing. [10 points]



(a) Manipulator diagram of the Fanuc M-900 (b) Kinematic diagram of the manipulator

Figure 1: Manipulator

2. **my_trajectory_generator.m**: This function creates your own discretized cartesian trajectory. You can choose to call **line_Trajectory_generator.m** for different line segments or write your own curved trajectory generator. The design of this function is left up to you. But, remember that it should return an array of points on your chosen trajectory with a given resolution or point spacing. [30 points]

3. **ik_fanuc_m900.m**: This function performs the inverse kinematics using inbuilt numerical solvers in Robotics System Toolbox. The function calls **line_trajectory_generator.m** (or **my_trajectory_generator.m**) and computes the joint positions for reaching the points on this trajectory with a given gripper orientation. The function also computes the time it takes for our inverse kinematics solver to arrive at the solution for the entire trajectory segment and the root mean square error between our actual path and ideal path. This function uses our fanuc_fk_m900.m file from the last problem set to calculate the gripper origin using the joint values returned by the inverse kinematic solver. You are free to use your own implementation from PS4, or use the solution files already provided to you with this problem set. **Note that, you can neglect the six gripper points from PS4 for this problem set**. In other words, we will take origin O_6 as our end-effector point we are trying to control. You can implement your solver in a subfunction called **solve_ik** at the bottom of this file. See the comments in the file for more information. [40 points]

You are supposed to experiment and write your observations for the following problems:

Line Trajectory:

- Put a plot in your report for the sketched line trajectory by the Fanuc robot.

- Report the root mean square error between the ideal line trajectory and your robot trajectory as output from your inverse kinematic solver. Also report the time for computing the entire line trajectory.
- Experiment with changing your initial guess from home configuration to the last joint configuration (the robot configuration at the previous point). Does this impact your solution (time and error) in any way?
- Change the solver algorithm from BFGS (default) to LM and compare your results.
- Experiment with a couple different values of weights on orientation and position and report the impact of this value on your position accuracy and computation time.

Your Own Trajectory:

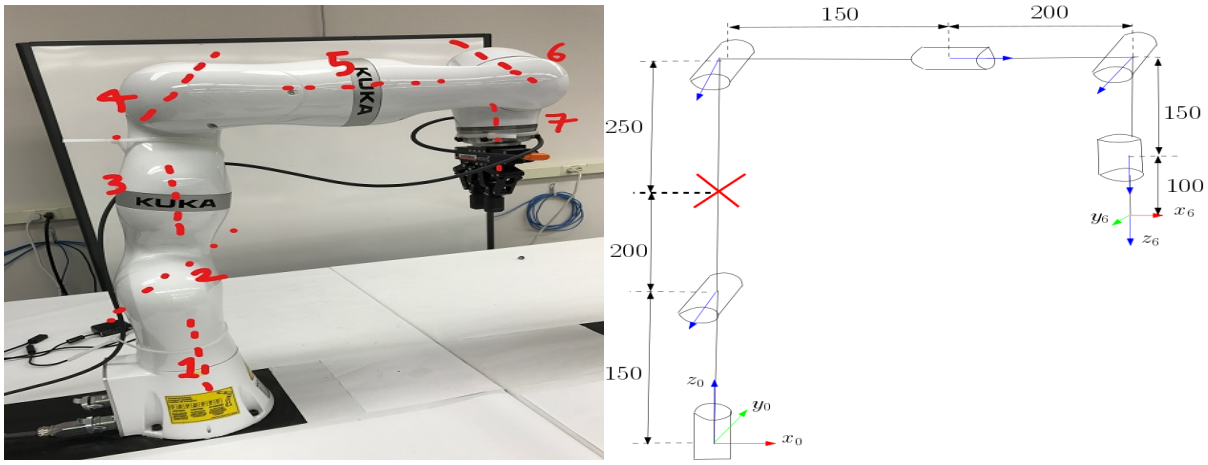
- Compute your own custom trajectory. In this task you can choose a planar trajectory, e.g. the first letter of your name on a plane. Put a plot in your report for the sketched custom trajectory by the Fanuc robot.
- Use your inverse kinematic function to compute the robot configuration on each point of the trajectory and plot your robot traversing your trajectory.
- Up to 5 extra points are given for traversing original, complex trajectories such as curved 3D shapes.

You are also provided with two more MATLAB scripts **ik_lab.m** and **plot_inverse_output.m**. You don't have to change anything in these files. In ik_lab.m, you may change the start and end points of the line, if you want to experiment with some other position values. You should run this file to call your implementation you write in ik_fanuc_m900.m. plot_inverse_output.m will simply take your result and animate the manipulator to demonstrate how well it is performing.

You should submit all your findings in a single page report for this section**[20 Points]**. Check the link provided to the MATLAB documentation in the resources section for more information on using the Inverse kinematics solver.

2 For Grad Students/ Extra Credit for UnderGrad

Figure 2 shows the same Kuka iiwa manipulator and the solved kinematic chain that we used for Problem Set 4. From Figure 2b you can see that we have removed joint 3 (marked with a red cross), which represents the redundant 7th joint of the Kuka iiwa. This is because the Robotics Toolbox does not yet support inverse kinematics of 7 DOF manipulators. You should consider the 200 mm and 250 mm links across joint 3 as one link. Note that we mentioned in PS4 that, using DH notations, we cannot plot this manipulator using the Robotics System Toolbox in MATLAB. However, the robotics system toolbox allows for constructing a robot object using a "rigid body tree". In this section you will explore this functionality of the toolbox to construct the rigid body chain for the kuka iiwa manipulator and then perform inverse kinematics. Check the link on "build a robot step by step" in the resource section for more information.



(a) Kuka iiwa manipulator with axis marked

(b) Kinematic Diagram of the Kuka iiwa with joint 3 removed

Figure 2: Kuka iiwa Manipulator Simplification

For this task, you will need to fill in your code in create_rigid_body_chain.m file and run the ik_lab_extra.m file.

1. **create_rigid_body_chain.m:** Fill in your code to construct the robot using Robotics System Toolbox support [20 points]. Check the link on "build robot step by step" in the resources section. Note that the frame assignment you did for DH convention in PS4 might not be valid for constructing this chain.

2. **ik_kuka_iiwa.m:** This function is similar to ik_fanuc_m900.m and should perform the same inverse kinematic operation for a straight line trajectory now for the KUKA iiwa (with reduced configuration to 6 joints). There is only one difference that, rather than returning position array as a result of forward kinematics we simply return an array of configuration solutions structures as returned by the Inverse Kinematics Solver. This is because we will rely on MATLAB toolbox functions for animating this manipulator.

3. **ik_lab_extra.m:** You do not have to modify this file. Simply execute this file to call your implementation in create_rigid_body_chain.m and ik_kuka_iiwa.m.

3 Resources

- https://www.mathworks.com/help/robotics/ug/inverse-kinematics-algorithms.html
- https://www.mathworks.com/help/robotics/ref/robotics.inversekinematics-system-object.html
- https://www.mathworks.com/help/robotics/ug/build-a-robot-step-by-step.html
- https://www.mathworks.com/help/robotics/coordinate-system-transformations.html

4 Submission

- Your function ik_fanuc_m900.m my_trajectory_generator.m, line_trajectory_generator.m along with your report with plots in a .zip file.
- If you are a graduate student, add create_rigid_body_chain.m, ik_kuka_iiwa.m, and a report for your kuka_iiwa section to your .zip file.