ECSE 4480/6470, MANE 4560/6120, CSCI 4480 October 18, 2021

Robotics I

Mini-Project 4: Jacobian, Singularity

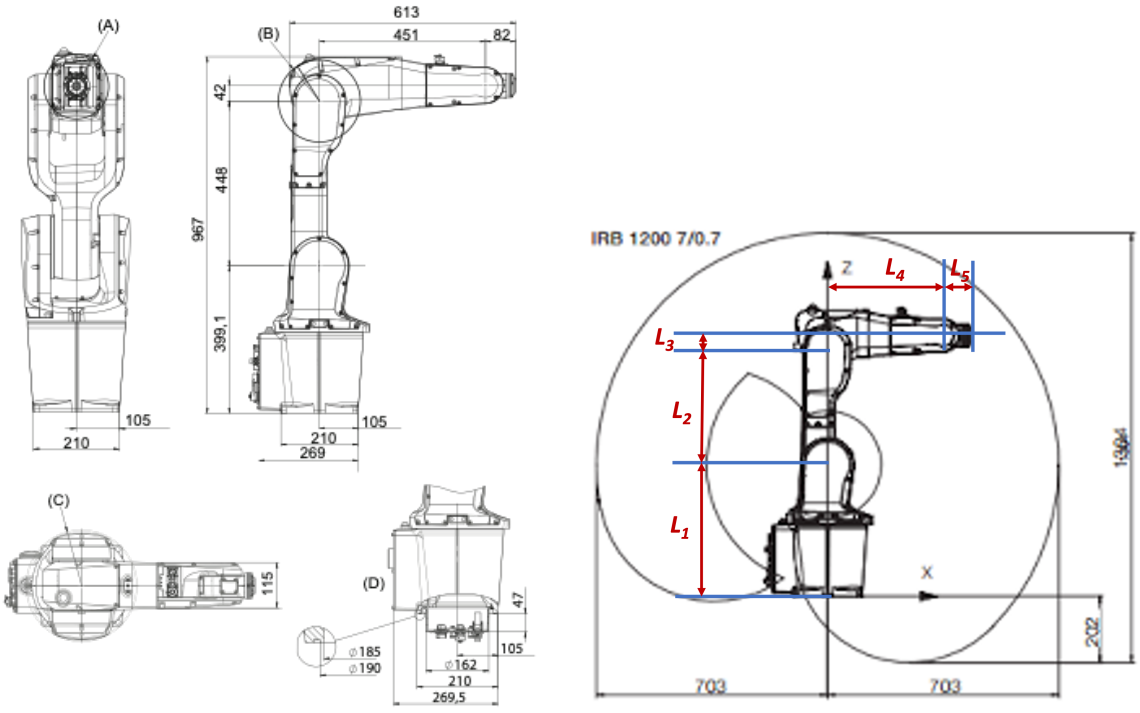
Assigned: October 18, 2021
Due: November 2, 2021

Check back here regularly for the latest update
Submit your project report to Gradescope and your code through your private GitHub repository. Discussion with your peers is encouraged (particularly on WebEx Team and Piazza), but you must hand in your own work and be able to explain what you have done in the project. **Verbatim copying (whether you copy from someone or let someone copy your work) of derivation, writing, software code, etc., is considered cheating and will result in zero for the project grade and notification to the Class Dean and Dean of Students. Multiple instances of cheating will result in failing of the course.**

Things to Do

- Post weekly update of your project progress on your team's WebEx space.

Task Description Consider the ABB IRB 1200-5/0.9 robot as shown in its zero configuration (the manual of the robot is posted on Piazza).



1. **Jacobian**

   (a) (10 points) Use the iterative method covered in class to modify your forward kinematics routine to compute the 6×6 Jacobian, $(J_T)_0(q)$, in addition to T_{0T}.

   (b) (10 points) Find the analytical expression of $(J_4)_3(q)$, and describe the procedure to compute $(J_T)_0(q)$ from $(J_4)_3(q)$.

   (c) (10 points) Find the Jacobian, $(J_T)_0(q)$, by using the MATLAB `geometricJacobian` function. (You will need to create a rigid body tree object by using `defineRobot.m` as in mini-project 3).

   (d) (Bonus Points) You now have three different ways to compute the Jacobian matrix:

      A. Numerical implementation of the iterative method in Part 1a evaluate $(J_T)_0(q)$ for a given q..

      B. Use the analytical expression in Part 1b to write a function to numerically evaluate $(J_T)_0(q)$ for a given q.

      C. Use the MATLAB Robotics Toolbox function as described in Part 1c.

      Generate 100 random joint angles using the uniform distribution for the following:

      i. (10 points) Verify that the results from the three methods agree.

      ii. (10 points) Compare the computation times. Try to explain the difference. (The MATLAB Robotics Toolbox uses a similar iterative algorithm, by Roy Featherstone, so should have similar performance as our iterative algorithm.)

2. **Singularities**

   (a) (20 points) Use $(J_4)_3$ from Part 1b to find the analytical expression and geometric interpretation of the condition for the three types of singularities: Boundary, Interior, and Wrist.

   (b) (10 points) Draw (e.g., by hand or using powerpoint) or plot (using MATLAB's `show` command) a representative singular pose for each type of singularity condition.

3. **Inverse Kinematics**

   (a) Modify the iterative inverse kinematics solver from Mini-Project 2 for a general n-link arm by using the forward kinematics and Jacobian code from Part 1a. You may use any error metric on $SO(3)$.

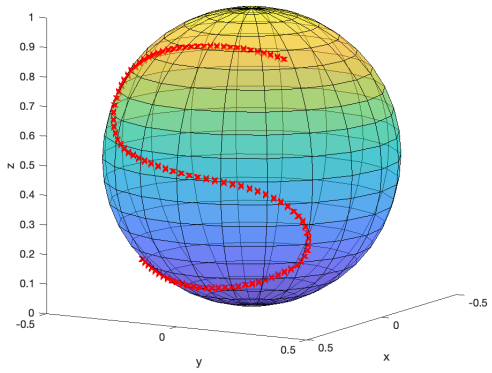
      i. (20 points) Verify your result by testing on 50 randomly generated end effector poses (produced using randomly generated joint angles and forward kinematics). Adjust your parameters (such as step size and number of iterations) so that the Frobenius norm (MATLAB norm function with 'fro' option) of the end effector pose error is less than 10^{-4}.

      ii. (Grad Section) (10 points) Discuss the effect of parameters (such as step size and number of iterations) in the convergence of solution.

iii. (Bonus Points) (10 points) Compare at least two different error metrics on $SO(3)$ and discuss the difference in the results.

(b) (Bonus points) (10 points) Compare with your analytical solution in terms of accuracy and speed.

4. **Jacobian and Trajectory Motion**: Consider the curve S on a sphere as from mini-project 3. The goal is to use Jacobian-based control to follow the curve



(a) (10 points) For one of the joint trajectories from Mini-Project 3, plot the minimum Jacobian singular values (for each point λ grid point) vs. λ.

(b) (10 points) Explain how the Jacobian singularity is related to the path speed $\dot{\lambda}$ when the joint velocity $\dot{q}$ is limited (as in mini-Project 3).

Deliverable

1. Your project report should be structured as follows:

   (a) Cover page with your name, course number, date, project title, and a statement on academic integrity (stating that you did this project by yourself):
   I, [name], certify that the following work is my own and completed in accordance with the academic integrity policy as described in the Robotics I course syllabus.

   (b) Summary: what did you try to do and accomplished.

   (c) Technical Content: containing the following for each section:

      i. Description of the problem

      ii. Derivation of the solution

      iii. Results based on your simulation

      The key is to **show that you understand what you are doing**, not just tweaking the code.

   (d) Conclusion: what you learned and what can be improved.

2. Put your code in your private GitHub repository (shared with the Instructor and TA).

3. If you have any video of your results (e.g., movies of the robot motion), provide a link (e.g., YouTube), or put them in your GitHub repository (in a separate folder under the miniproject1 folder).