ECSE 4480/6470, MANE 4560/6120, CSCI 4480 September 13, 2021

Robotics I

Mini-Project 2: Planar Robot Arm Motion

Assigned: September 13, 2021
Due: September 28, 2021

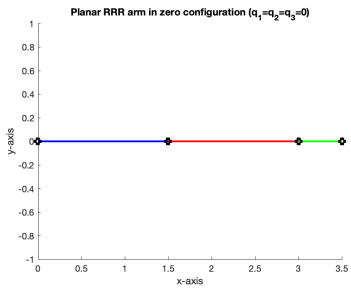
Check back here regularly for the latest update
Submit your project report to `Gradescope` and your code through your private GitHub repository. Discussion with your peers is encouraged (particularly on WebEx Team and Piazza), but you must hand in your own work and be able to explain what you have done 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.**
This course will use the MATLAB Robotics Toolbox. You are welcome to use other simulation platforms such as ROS/Gazebo, MATLAB ROS Toolbox, Webot, Unity, and others. Your are encouraged (but not required) to show the robot motion (in parts 3 and 4) as a movie file (take a look of the MATLAB `movie` command).

Things to Do

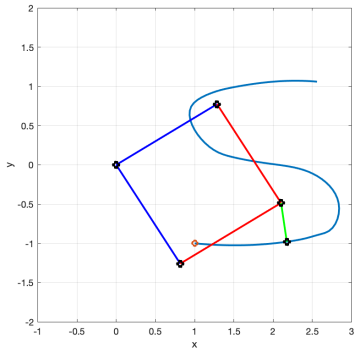
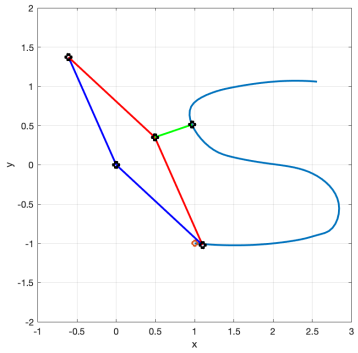
- Complete course project survey https://forms.gle/NS5pVsZ8gBMtdK5B8

Task Description

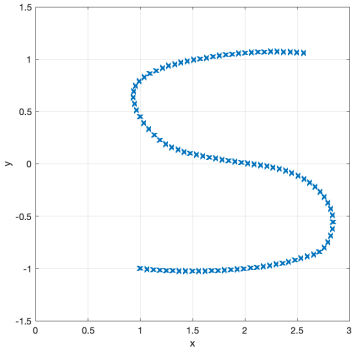
1. Consider a three-link planar robot arm with link lengths (1.5,1.5,.5) m. The zero configuration of the arm is as shown.



The goal is to plan the motion of the arm so the tip of the arm traces out the letter **S** with the robot orientation normal to the letter. Examples at two different locations are shown with the two poses of the robot (elbow up and elbow down).

The data file `S_letter_path.mat` containing the locations of the path points will be posted on Piazza.



Plot the letter S and its outward normal (towards left part of the page) and generate the path length array, λ, corresponding to the letter S.

2. Write down the forward kinematics of the three-link arm. Explain the algebraic, geometric, and iterative solution methods to inverse kinematics.

3. Write the forward and inverse kinematics routines for the three-link arm using the following specification.

   Define an object `robot` that has the following attributes:
   • q: $n \times 1$ joint displacement vector
   • P: a $3 \times n + 1$ link displacement vector in the zero configuration.
   • T: a 4×4 homogeneous transform for the pose (orientation and position) of the robot.
   • J: a $3 \times n$ matrix relating joint velocity to the end effector angular and linear velocities.

   The forward kinematics should take the robot with joint angle q and compute T and J. The inverse kinematics (only for a 3-dof planar arm) should take the end effector pose T and return all possible solutions q.

   Verify that your code as follows:

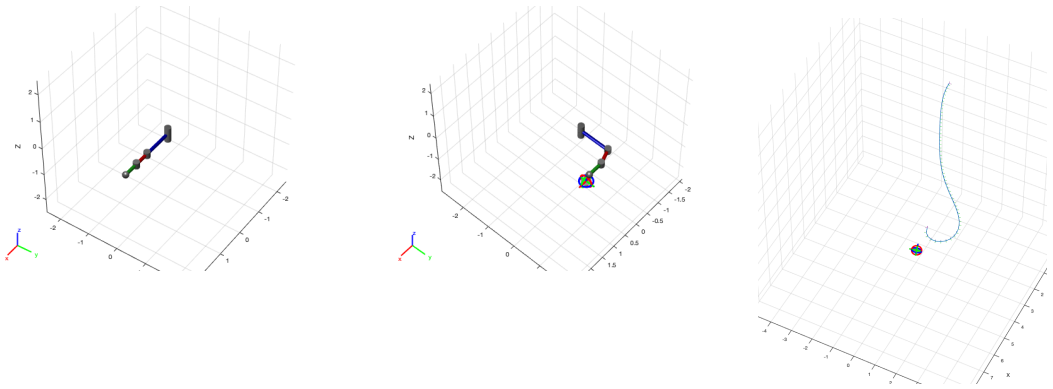
   (a) Randomly generate a joint vector q with each element $q_i \in [-\pi, \pi]$.

(b) Use your forward kinematics code to compute T corresponding to q.

(c) Use your inverse kinematics code to compute all solutions q_{sol}.

(d) Check that one of the q_{sol} is your original q.

4. Find the desired pose for each point on the letter S path. Note that the end point position is just the path point itself, you need to determine the orientation so that the last link is normal to the curve as shown. I chose $\vec{e}_{2_T}$ to be the outward normal and therefore $\vec{e}_{1_T}$ is in the direction of motion (from the bottom of the letter S). Find the robot motion (q as a function of λ) that will trace out the letter S. Visualize the robot motion either using the plot command (e.g., using the simple `plotarm.m` program) or use URDF (Universal Robot Description Format) and MATLAB's `importRobot` command to construct a three-link planar arm and use `show` to visualize. You can also use the `interactiveRigidBodyTree` command to drag the robot end effector around (it uses the MATLAB iterative solver `inversekinematics`). You can also construct a MATLAB rigidbody tree directly (e.g., see my sample code in `planarNlink.m`). You can then use interactive visualization as well.



5. Suppose the maximum robot joint velocities are all 1 rad/sec, and the robot is supposed to trace out the letter in constant *path speed*, i.e., $\dot{\lambda}$. Find a solution that gives you the largest path speed subject to the robot joint velocity constraint. How does the linkage lengths and the positioning of the letter S affect the maximum path speed?

6. We will cover several Jacobian-based robot motion control methods. Implement one of them (subject to the same joint velocity constraint). Compare your Jacobian-based method with the inverse kinematics approach above (in terms of the relative advantages and performance). Explain your choice of the Jacobian-based method among the several available alternatives.

7. (Graduate Section) Consider a 10-link planar arm with each link length 0.35 m. Apply either a Jacobian-based method or Jacobian-based inverse kinematics method for the arm end effector to track the letter S (while keeping the last link normal to the curve).

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).