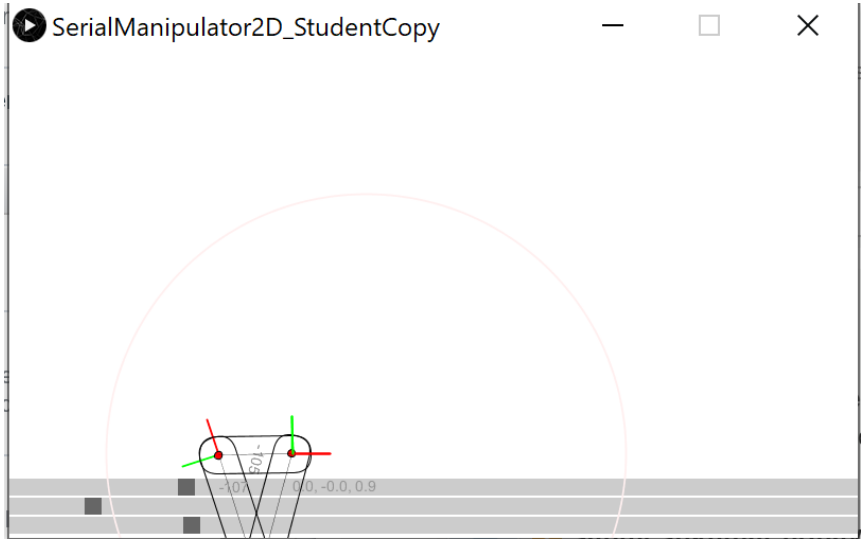
CS 3630 Project 4

Name: Rushabh Varia
GT email: rvaria@gatech.edu
GT username: rvaria3
GTID: 903286713

1. Give a short overview of how the simulator works. (Your explanation does not have to be very detailed at this point)

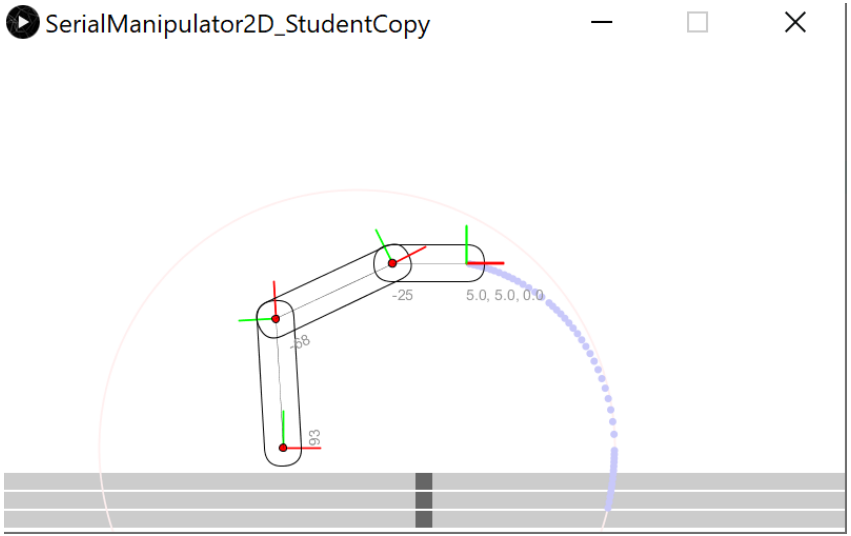
Three sliders are used to change the position of each arm and the axes tells us the orientation of each joint and end effector. There is free movement as there are no obstacles so it is easy to get the end effector to any point within the marked range.

2. Screenshot and paste the robot arm with the end effector frame and base frame aligned. How did you achieve this? What were the main difficulties in achieving the goal?



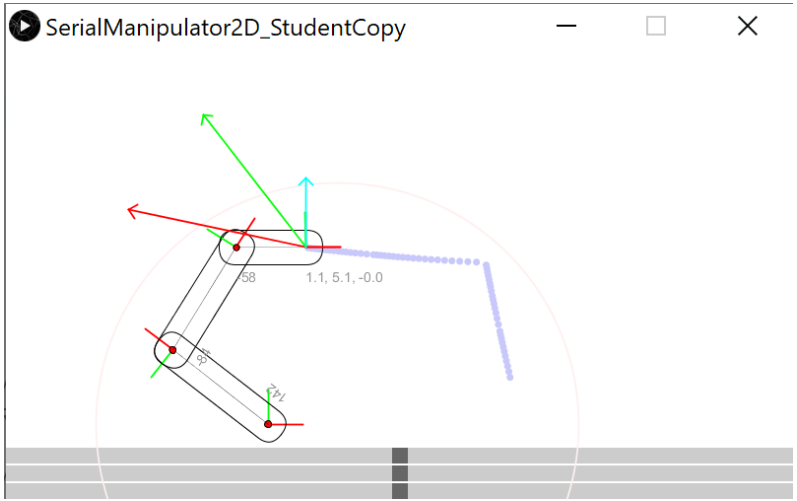
I achieved this by drawing it out on paper first and thinking about how I can achieve this orientation using the fixed distances. The main difficulty was not being able to adjust the length of an arm, and only the angles. Imagining it was hard as I was not sure how to do it without changing the length/distance of each arm.

3. Screenshot and paste the robot arm with the joint-space control scheme. Make sure to include the trail. What did you do to implement this? Why does the trail look “arcsy”?



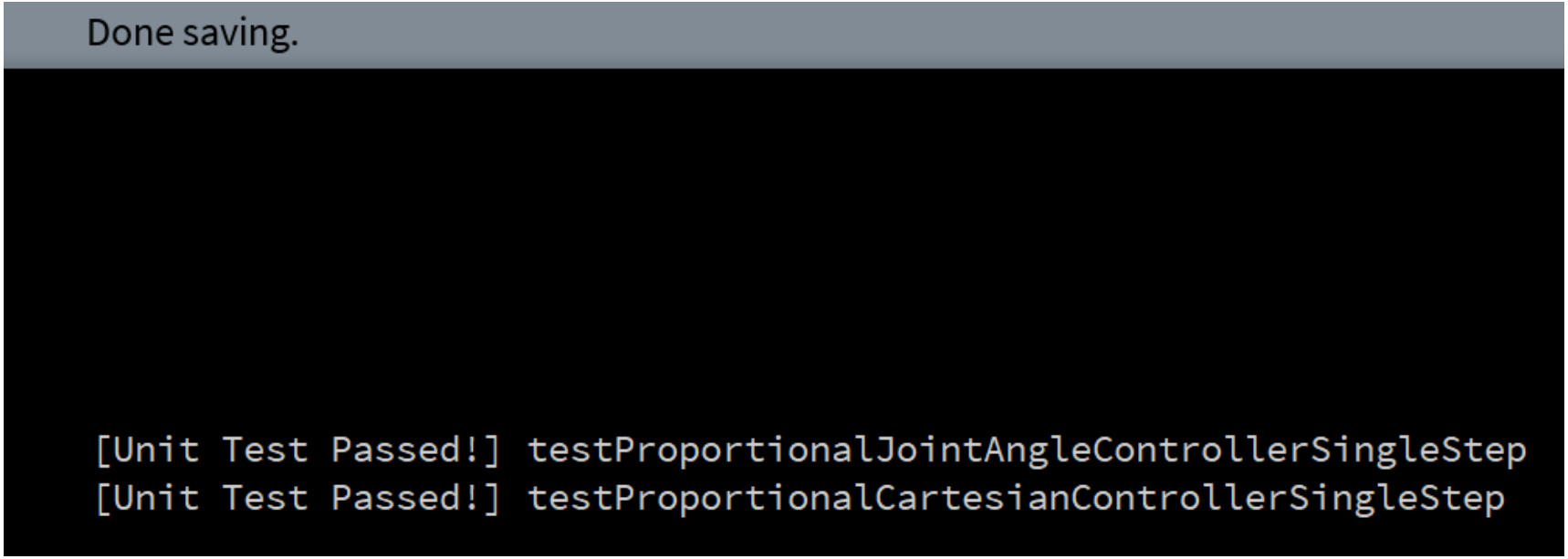
I implemented the joint angle controller where the input was the current joint angles, desired joint angles and gain factor. The aim was to create a set of waypoints for the end effector to follow to reach the desired end point in the same orientation/alignment as the base. The inverse kinematic equations were already implemented to give us the desired and current angles, so we simply had to use qt+1 = qt + Kp(qd-qt) to show the linear interpolation or simple control law in joint space to move from one waypoint to other. The trail looks arcsy as the desired angles have an inverse proportionality and that is how the error is calculated (difference). Also as the angles were changed, it was along a curve axis.

4. Screenshot and paste the robot arm with the cartesian control scheme. Make sure to include the trail. What did you do to implement this? What is the role of th e inverse Jacobian here?



I implemented the cartesian controller where the input was the desired x, y and theta position, current positions, angles and Kp (gain factor). The aim was to create a set of waypoints for the end effector to follow to reach the desired end point in the same orientation/alignment as the base. The inverse jacobian was already given, so we simply had to use qt+1 = qt + Kp*InvJacobianMatrix*error to show the linear interpolation or simple control law in cartesian space to move from one waypoint to other. Jacobian represents the change of the end effector's component position with respect to small changes in each joint angle. The inverse of it multiplied with the error gives us the required individual joint positions.

5. Screenshot and paste the result of the unit tests



6. Discuss what you have learned from this project.

I learned that it can be very challenging to work with a two link robot even though it might seem it is easy to move it around and achieve the point where the end effector needs to be. Also learned that there are various methods to implement this and which one can be more efficient and easier to use.