AuE 8220- Autonomy: Mobility and Manipulation

Homework 5: 3RRR Modeling in CoppeliaSim

Authors: Tanmay Samak, Chinmay Samak, Riccardo Setti, Olamide Akinyele

Problem 1:

1-A

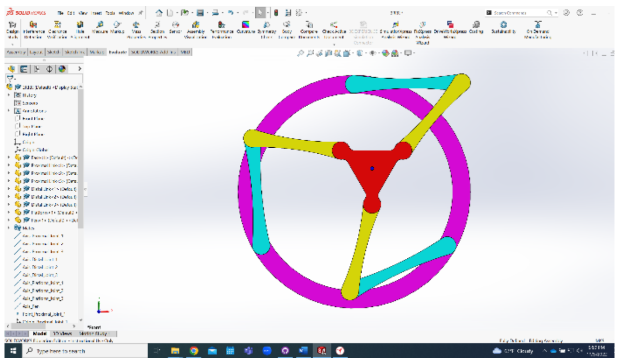
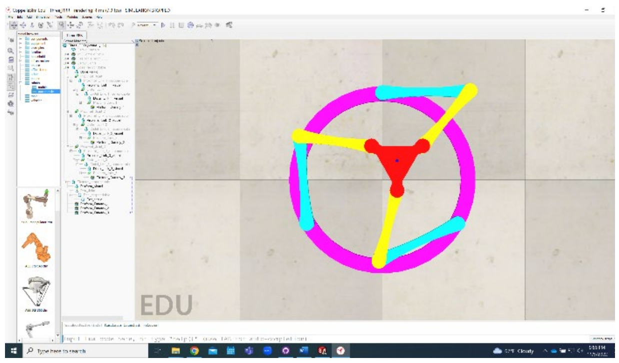
Concept: We used SolidWorks to model and assemble the 3RRR parallel manipulator robot (while adhering to the dimensions (in mm) described in the question) and exported its URDF using the SW2URDF exporter plugin (refer `\3RRR\SW2URDF2CS.txt` for more details). We then imported the URDF into CoppeliaSim and modified it to work as follows:

- The duplicate links for "Platform" and "Pen" were removed.
- A common "Platform" with "Pen" was brought up the parent chain (hierarchy) to match the level of "Base".
- The common "Platform" was linked to all the 3 "Distal Links" using "Dummy" objects which were positioned and oriented exactly at the joints.
- The offset angles w.r.t. global horizontal for each of the joints in "home-configuration" were manually calculated (and verified using SolidWorks).

SolidWorks Parts and Assembly: `\3RRR\SOLIDWORKS`

URDF: `\3RRR\URDF`

CoppeliaSim Scene: `Three_RRR.ttt`

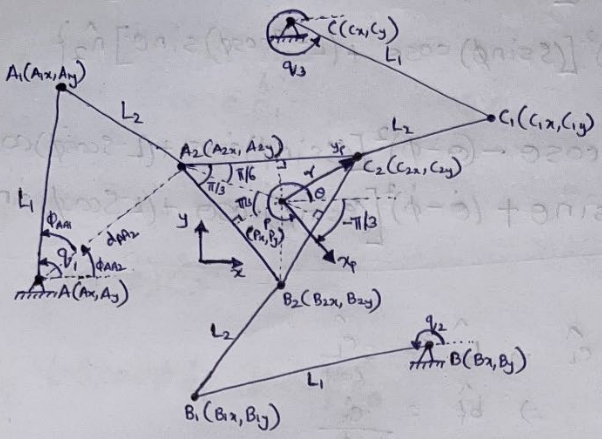



1-B

Concept: We developed the inverse kinematics model of the 3RRR parallel manipulator robot and added it as a function to the MATLAB script. For testing the motion of end effector from (50mm, 65mm, -60deg) to (0mm, 65mm, -60deg), we first planned a quintic polynomial task-space trajectory from start point to end point using the `ctraj()` function from PCRTB. We then computed the IK at each of the interpolated trajectory coordinates to get the joint angles for base/proximal joints, which we then commanded the robot to follow.

IK Manual Derivation:

3<u>R</u>RR Manipulator



$A_{2x} = P_x + r\cos(\theta - 5\pi/6)$
$A_{2y} = P_y + r\sin(\theta - 5\pi/6)$

$B_{2x} = P_x + r\cos(\theta - \pi/6)$
$B_{2y} = P_y + r\sin(\theta - \pi/6)$

$C_{2x} = P_x + r\cos(\theta - 9\pi/6)$
$C_{2y} = P_y + r\sin(\theta - 9\pi/6)$

$(d_{AA_2})^2 = (A_{2x} - A_x)^2 + (A_{2y} - A_y)^2$
$(d_{BB_2})^2 = (B_{2x} - B_x)^2 + (B_{2y} - B_y)^2$
$(d_{CC_2})^2 = (C_{2x} - C_x)^2 + (C_{2y} - C_y)^2$

$\phi_{AA_2} = \tan^{-1}\left(\frac{A_{2y} - A_y}{A_{2x} - A_x}\right)$
$\phi_{BB_2} = \tan^{-1}\left(\frac{B_{2y} - B_y}{B_{2x} - B_x}\right)$
$\phi_{CC_2} = \tan^{-1}\left(\frac{C_{2y} - C_y}{C_{2x} - C_x}\right)$

$\phi_{AA_1} = \cos^{-1}\left(\frac{(d_{AA_2})^2 + (L_1)^2 - (L_2)^2}{2L_1 d_{AA_2}}\right)$
$\phi_{BB_1} = \cos^{-1}\left(\frac{(d_{BB_2})^2 + (L_1)^2 - (L_2)^2}{2L_1 d_{BB_2}}\right)$
$\phi_{CC_1} = \cos^{-1}\left(\frac{(d_{CC_2})^2 + (L_1)^2 - (L_2)^2}{2L_1 d_{CC_2}}\right)$

$q_1 = \phi_{AA_2} + \phi_{AA_1}$
$q_2 = \phi_{BB_2} + \phi_{BB_1}$
$q_3 = \phi_{CC_2} + \phi_{CC_1}$

IK MATLAB Function: `Three_RRR.m` lines 319-374

Trajectory Generation & Tracking Code: `Three_RRR.m` lines 110-161

1-C

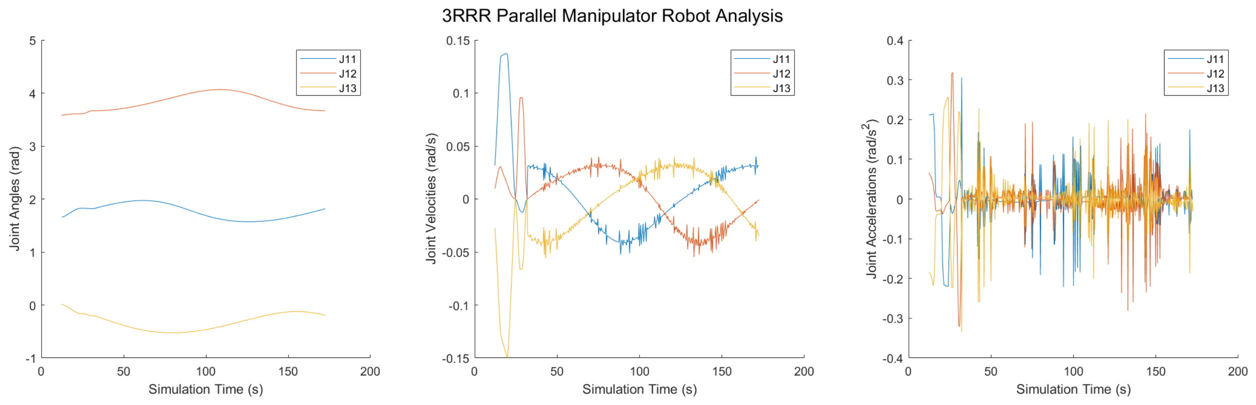
Concept: We first moved the end effector from current position (0mm, 65mm) to the starting point on the circular trajectory (0mm, 50mm). For this, we first planned a quintic polynomial task-space trajectory from start point to end point using the `ctraj()` function from PCRTB. We then computed the IK at each of the interpolated trajectory coordinates to get the joint angles for base/proximal joints, which we then commanded the robot to follow. Next, for moving the end effector in a circle of radius 50mm, centered around (0,0) mm, we first planned a task-space polar-coordinate trajectory from 0 to 360 deg. We then converted the trajectory into cartesian coordinates and computed the IK at each of the interpolated trajectory coordinates to get the joint angles for base/proximal joints, which we then commanded the robot to follow.

***Note:** We noticed that achieving end effector orientation of 0 deg required the parallel manipulator robot to be assembled in an elbow-down configuration (as opposed to the elbow-up configuration required to achieve end effector orientation of -60 deg in 1-B). Hence, we proceeded (as mentioned in the question) to execute the circular trajectory with end effector orientation of 0 deg.*

IK MATLAB Function: `Three_RRR.m` lines 319-374

Trajectory Generation & Tracking Code: `Three_RRR.m` lines 162-213, 214-268

Results



The plot shows joint positions (angles), velocities and accelerations w.r.t. time for the entire simulation where the end effector first moves from (50mm, 65mm, -60deg) to (0mm, 65mm, -60deg) as described in ***1-B***, then moves to starting point on the circular trajectory (0mm, 50mm) as described in ***1-C***, and finally moves in a circular trajectory of radius 50mm, centered around (0,0) mm, also as described in ***1-C***.

***Note:** CoppeliaSim does NOT provide joint velocity and acceleration data. Hence, we had to compute these values by taking first and second derivatives of the joint positions (angles) w.r.t. a fixed timestep (dt) although the actual value of timestep varies continuously depending on the processing capacity of the host machine running the simulation.*