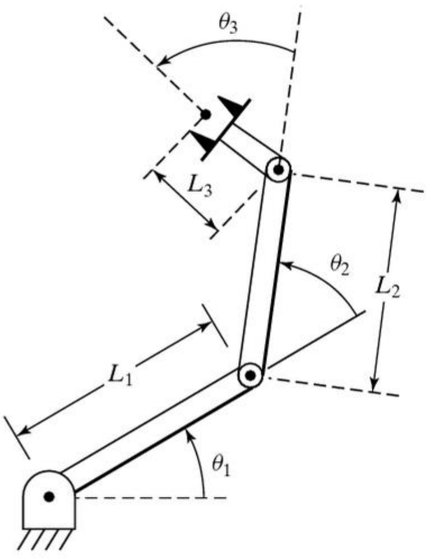
ENN 581 – Semester 2 – 2023

Week 3 – Inverse Kinematics of Manipulator Arms

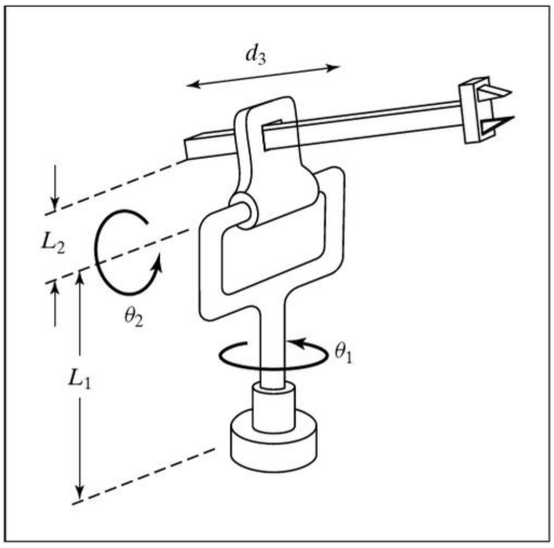
Krishna Manaswi Digumarti

Theory

1. Derive the inverse kinematics of the following 3DoF manipulator.



2. Consider the following manipulator. How many solutions do the (position) kinematic equations possess?

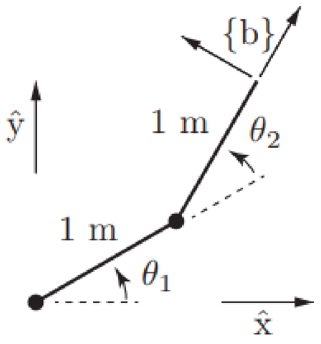


3. Name two reasons for which analytical solutions are preferred over iterative solutions.

4. Consider a planar 2R robot as shown below. Each link is 1m in length. We want to find the joint angles that place the tip at (x, y) = (0.366 m, 1.366 m), which corresponds to $q^* = (30^o, 90^o)$ and

$$^{0}T_3 \text{ OR } \xi_3 = \begin{bmatrix} -0.5 & -0.866 & 0 & 0.366 \\ 0.866 & -0.5 & 0 & 1.366 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Starting with an initial guess of $q^0 = (0^\circ, 30^\circ)$, write down 4 iterations of using the Newton-Raphson method to solve for the correct joint angles.



Programming – robotics toolbox for python

5. Using the robotics toolbox for python, plot all the configurations of the Puma 560 robot. How many are there? Use the skeleton code below.

```
import roboticstoolbox as rtb
import matplotlib.pyplot as plt
from spatialmath.base import *

# import the puma robot arm model
puma = rtb.models.DH.Puma560()

# plot the initial configuration of the robot
figure1 = plt.figure()
puma.plot(puma.qz, 'pyplot', fig=figure1)

# set a desired target configuration
T = transl(0.5, 0.1, 0.0) @ rpy2tr(0, 180, 0, 'deg')
print(T)

# solve the inverse kinematics for the desired target
# <== change the parameter below to see other variations ==>
sol = puma.ikine_NR(T, 'r') # we are forcing the robot to go into a right handed configuration
puma.plot(sol.q, 'pyplot', fig=figure1)

trplot(T) # plot the desired target to confirm

plt.draw() # draw the plot
plt.pause(10) # show it for 5 seconds
```

Simulation – CoppeliaSim

Use the same robot scene as from the previous tutorial. Set the robot into kinematic mode and adjust the joints and links as described last week.

The DH parameter table for the robot that we used in the previous practical is given as follows:

	Theta [deg]	d [mm]	alpha [deg]	A [mm]
Link 1	0	243.3	-90	0
Link 2	-90	0	180	200
Link 3	-90	0	90	87
Link 4	0	227.6	90	0
Link 5	0	0	-90	0
Link 6	0	61.5	0	0

Step 1: Define a robot with these parameters using the robotics toolbox. Refer to the rtb.DHRobot() function.

Step 2: Given a target position, solve the inverse kinematics problem using the robotics toolbox. The ikine_NR() function uses a Newton-Raphson solver.

Step 3: Verify that the solution is correct, by using the Joint tool in CoppeliaSim. Access this tool via modules -> Kinematics -> Joint tool. Input the joint parameters that the above solution gave you and verify the position of the end effector.