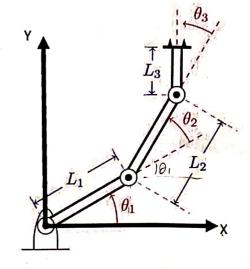
## **ICP 8: Robot Trajectory Generation**

#### 1. The robot:

a. The forward kinematics problem (suppose the robot has N joints)

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} L_{1} (0501) \\ L_{1} \sin \theta_{1} \end{bmatrix} + \begin{bmatrix} L_{2} (0501 + \theta_{2}) \\ L_{2} \sin (0102) \end{bmatrix} + \begin{bmatrix} L_{3} \cos (0102 + \theta_{3}) \\ L_{3} \sin (0102 + \theta_{3}) \end{bmatrix}$$

$$= \begin{bmatrix} L_{1} C_{1} + L_{2} C_{12} + L_{3} C_{123} \\ L_{1} S_{1} + L_{2} S_{12} + L_{3} S_{113} \end{bmatrix}$$



b. Jacobian matrix

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}_{2x} = \int \times \begin{bmatrix} \dot{\theta} \mathbf{1} \\ \vdots \\ \dot{\theta} \mathbf{N} \end{bmatrix} N^{x}$$
Matrix  $J$  is of dimension (rows:  $2 \times \text{columns}$ :  $N$ )
$$J = \underbrace{\frac{\partial x}{\partial t}}_{2x} = \underbrace{\frac{\partial \theta}{\partial t}}_{2x} = \underbrace{\frac{\partial x}{\partial t}}_{2x} = \underbrace{\frac$$

#### c. Inverse kinematics

efbtfc0042

Explain the relationship between Jacobian matrices and IK:

$$\Delta X = J \times \Delta \theta$$

$$\Delta \theta = \left[ J \right] \Delta X$$

$$\Delta \theta = \left[$$

2. Trajectory generation:

What is your plan for generating the trajectories?

### Given:

(1) joint\_angles: the current robot joint angles

(2) joint goal angles: the final robot joint angles when the goal is reached.

Our plan to generate trajectory is curve fitting a sinusoidal function

 $\dot{\theta} \rightarrow \dot{\theta}_{max}$   $\dot{\theta} = \sin(wt)$   $\dot{\theta} = w\cos(wt)$   $\dot{\theta} = w\cos(wt)$ 

In your design, is the EE speed, acceleration and the amount of joint angle changes regulated or capped?

I collaborated with Alex, in our design, we set up the maximum speed and acceleration to get a sin function to follow.

List two benefits of your design:

1 I can control the Gmax which means I can control my torque.

decause 7=10, thus make my motor can perform in their capability.

I can control the Gmax which makes the robot be safer.

- 3. Python Hands-On Task:
  - a. Download the code from here to your virtual machine
    - i. Unzip it and put it under a directory of your choice
    - ii. Go to the directory: \$ cd ~/director\_of\_your\_choice
  - b. Make files executable
    - \$ chmod +x NLinkArm.py
    - \$ chmod +x RobotTrajGen.py
  - c. Make edits to the python file: RobotTrajGen.py
    - Find all the places with keyword "TODO"
    - ii. Implment your own trajectory generation algorithm

lomam : 2D

nxn

#### d. Run the code:

i. Command:

\$ cd ~/director\_of\_your\_choice

\$ python RobotTrajGen.py

- ii. show\_animation flag: This is a flag on top of the "RobotTrajGen.py" file. Toggle the Boolean variable to change between:
  - 1. Randomly generated goals & numeric outputs on the terminal
  - 2. User defined goals (by right mouse click) & visualization

#### 4. What to submit:

- a. Questions:
  - i. Answer the questions in 1. And 2.
  - ii. Answer the three questions below:

When designing the trajectory, we consider the goal and current joint angles instead of the cartesian position difference. List the downsides or risks if you do it in the cartesian space.

Ans:
In cartesion space, may be some joints other
than the EE will collide with obstacle

When EE is safe.

In previous lectures we were using geometric methods and algebraic methods to solve for IK, but here using Jacobians seems like a very systematic way. Does IK using inverse jacobian always work for every robotic application? Explain your reasons.

Ans:

Actually, doing inverse jacobian is a really time consuming procedure, while using geometric or algebraic method to get the closed form solution is less complicated to get the joint value. Thus in real-time manipulation, inverse jacobian cannot handle.

In our application, the trajectory is purely a smooth transition from initial to target joint angles. What if there are obstacles in the workspace? Explain an idea to perform trajectory generation while taking into consideration of the obstacles in the environment.

Ans: TCP8 -> HW4.

If there are obstacles in workspace, I'll sample some points in joint space to check the range of collision point and make my trajectory generation don't pass through the vange / region, thus make the trajectory be safe.

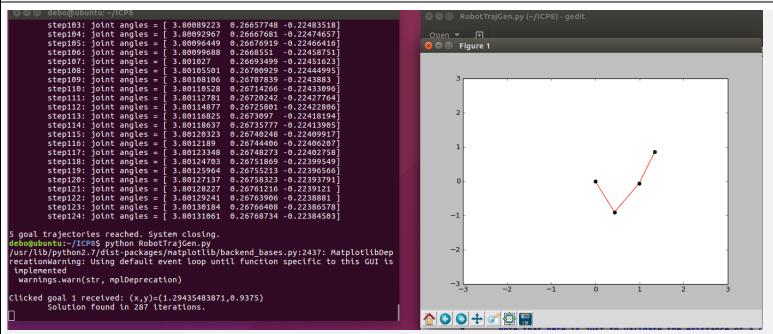
And also I will consider the distance between Joints and

Obstacle as a cost, to make it safe.

And also the speed when approaching obstacles, I'll also slow it down.

# Default link # and link length:

Show\_animation = True:



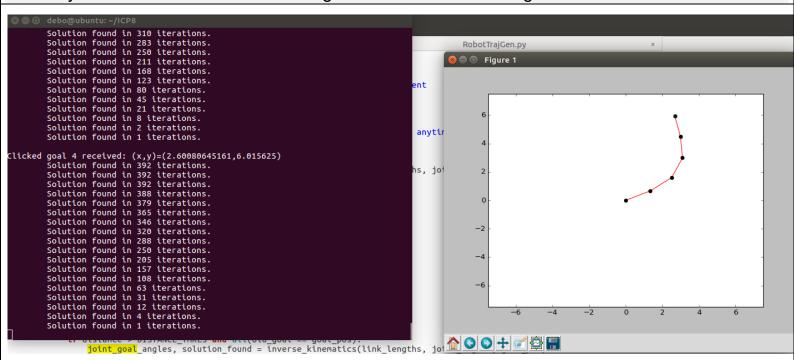
## Default link # and link length:

Show animation =False:

```
🔕 🖨 📵
        debo@ubuntu: ~/ICP8
         step95: joint angles = [
                                  3.80047122
                                               0.26546068 -0.22583153]
         step96: joint angles =
                                  3.80053814
                                               0.26563819 -0.22567317
         step97: joint angles =
                                  3.80060037
                                               0.26580327 -0.2255259
         step98: joint angles =
                                  3.80065824
                                               0.26595679 -0.22538893]
         step99: joint angles = [ 3.80071206
                                               0.26609957 -0.22526155]
         step100: joint angles = [3.80076212]
                                                0.26623235 -0.22514309]
         step101: joint angles =
                                   3.80080867
                                                0.26635584 -0.22503292]
         step102: joint angles =
                                   3.80085196
                                                0.26647068 -0.22493047]
         step103: joint angles =
                                   3.80089223
                                                0.26657748 -0.22483518]
                                   3.80092967
         step104: joint angles =
                                                0.26667681 -0.22474657]
         step105: joint angles =
                                   3.80096449
                                                0.26676919
                                                           -0.22466416]
         step106: joint angles =
                                   3.80099688
                                                0.2668551
                                                            -0.22458751]
                                   3.801027
                                                0.26693499 -0.22451623
         step107: joint angles =
         step108: joint angles =
                                                           -0.22444995]
                                   3.80105501
                                                0.26700929
         step109: joint angles =
                                   3.80108106
                                                0.26707839 -0.2243883
                                   3.80110528
         step110: joint angles =
                                                0.26714266 -0.22433096]
         step111: joint angles =
                                   3.80112781
                                                0.26720242 -0.22427764]
         step112: joint angles =
                                   3.80114877
                                                0.26725801 -0.22422806]
         step113: joint angles =
                                   3.80116825
                                                0.2673097
                                                           -0.22418194]
         step114: joint angles =
                                   3.80118637
                                                0.26735777 -0.22413905]
         step115: joint angles =
                                   3.80120323
                                                0.26740248 -0.22409917
         step116: joint angles =
                                   3.8012189
                                                0.26744406 -0.22406207]
                                   3.80123348
         step117: joint angles =
                                                0.26748273 -0.22402758]
         step118: joint angles =
                                   3.80124703
                                                0.26751869 -0.22399549]
         step119: joint angles =
                                   3.80125964
                                                0.26755213
                                                           -0.22396566]
         step120: joint angles =
                                                0.26758323 -0.22393791]
                                   3.80127137
         step121: joint angles =
                                   3.80128227
                                                0.26761216 -0.2239121
                  joint angles =
                                   3.80129241
         step122:
                                                0.26763906 -0.2238881
         step123: joint angles =
                                   3.80130184
                                                0.26766408 -0.22386578]
         step124: joint angles = [ 3.80131061
                                                0.26768734 -0.22384503]
¶5 goal trajectories reached. System closing.
debo@ubuntu:~/ICP8$
```

## Show animation = True:

What's your choice of link # and link length? 5 links and link length 1.5



## Modifiedlink # and link length:

Show\_animation = False:

What's your choice of link # and link length? 5 links and link length 2.5

```
🔞 🖨 🗊 debo@ubuntu: ~/ICP8
         Solution found in 1 iterations.
step326: joint angles = [-54.52112207
                                                    -1.6054765
                                                                   2.13918889 -38.3605535
                                                                                              41.44651381
  10.54553217]
        Solution found in 1 iterations.
         step327: joint angles = [-54.5405866
                                                    -1.54117123
                                                                   2.11569053 -38.33047219 41.44610675
  10.55729008]
Random goal 5 generated: (x,y)=(3.81820154696,3.00713901424)
Solution found in 39 iterations.
Solution found in 39 iterations.
         step1: joint angles = [-55.22565235
                                                                 2.06709019 -38.06673715 42.13158419
                                                 -1.69029447
  11.10318794]
         Solution found in 17 iterations.
         step2: joint angles = [-55.44759872
                                                 -1.70874335
                                                                 2.05628443 -37.97952196 42.30803237
  11.27036521]
         Solution found in 31 iterations.
         step3: joint angles = [-55.45786658
                                                 -1.68916279
                                                                 2.15324107 -37.96217493 42.23663624
  11.25757777]
         Solution found in 29 iterations.
         step4: joint angles = [-55.41721812]
                                                 -1.65587281
                                                                 2.28186081 -37.95105086 42.11656782
  11.202746991
         Solution found in 24 iterations.
         step5: joint angles = [-55.39305635]
                                                 -1.62229571
                                                                 2.40350725 -37.92802884
                                                                                            42.02468864
  11.16385882]
         Solution found in 14 iterations.
         step6: joint angles = [-55.4015844]
                                                  -1.59362974
                                                                 2.5066647 -37.89354109 41.97469863
  11.15424874]
         Solution found in 1 iterations.
         step7: joint angles = [-55.43320286]
                                                -1.57058876
                                                                 2.5969315 -37.85504713 41.94699326
  11.16328115]
5 goal trajectories reached. System closing.
debo@ubuntu:~/ICP8$
```