Control of 3-Link Manipulator

Md Ali Azam, Sandesh Acharya

Final Project of
EE 592: Linear System Theory
Submitted to
Dr. Randy C. Hoover
in Fall 2018

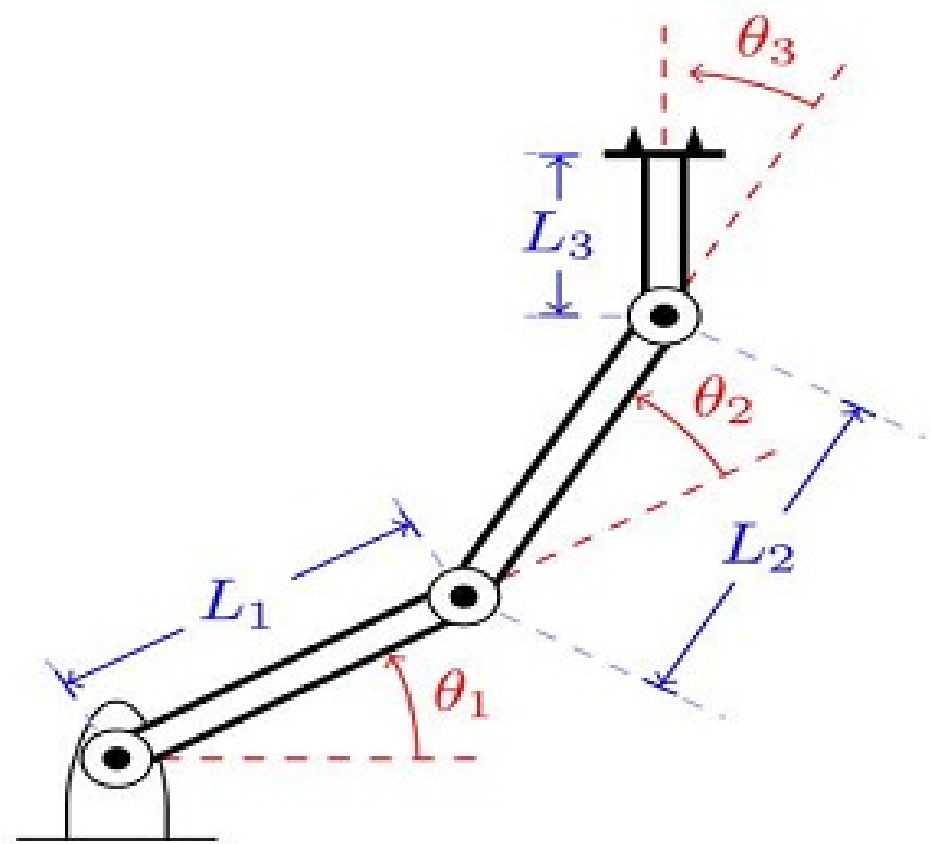
Electrical Engineering
South Dakota School of Mines and Technology

Overview

Introduction

- We derive an expression for the manipulator Jacobian
- We simulate a pick and place motion from point A to B
- We use a damped least square to avoid vibration
- Out of scopes:
  - We do not control orientation of the end-effector
  - Rotations at each link is not considered

Geometric of Three Link Manipulator



Courtesy: Dr. Randy C. Hoover

Forward Kinematic Equations

- The forward kinematic equations derived from graphical solution are given below

$$f_1(q) = L_1 cos(\theta_1) + L_2 cos(\theta_1 + \theta_2) + L_3 cos(\theta_1 + \theta_2 + \theta_3)$$
$$f_2(q) = L_1 sin(\theta_1) + L_2 sin(\theta_1 + \theta_2) + L_3 sin(\theta_1 + \theta_2 + \theta_3)$$

  - q is the vector containing angles shown in the Figure 1; $q = [\theta_1, \theta_2, \theta_3]^T$
  - $[L_1, L_2, L_3]$ are the lengths of the manipulator arms
  - $f_1(.)$ and $f_2(.)$ represent the global x and y positions of the manipulator end-effector respectively

Jacobian Matrix

- Jacobian matrix is the matrix of all first-order partial derivatives of the kinematics
- Jacobian matrix of the forward kinematic equations is given below

$$J = \frac{df(q)}{dq} = \begin{bmatrix} \frac{df_1}{d\theta_1} & \frac{df_1}{d\theta_2} & \frac{df_1}{d\theta_3} \\ \frac{df_2}{d\theta_1} & \frac{df_2}{d\theta_2} & \frac{df_2}{d\theta_3} \end{bmatrix} \in \mathbb{R}^{2\times 3}$$

Obtaining Angles

We want to figure out the angles associated with robot arms. Once we have the angles q, we know the exact location and orientation of the end-effector.

- Redefining global position of the end-effector $X = [f_1(q), f_2(q)]$
- We get the following relation from the forward kinematics

$$\dot{X} = J\dot{q}$$

- We use the following equation to obtain the angles at each link

$$\dot{q} = J^{\dagger}\dot{X} \tag{1}$$

Manipulator Arm Speed Limitation

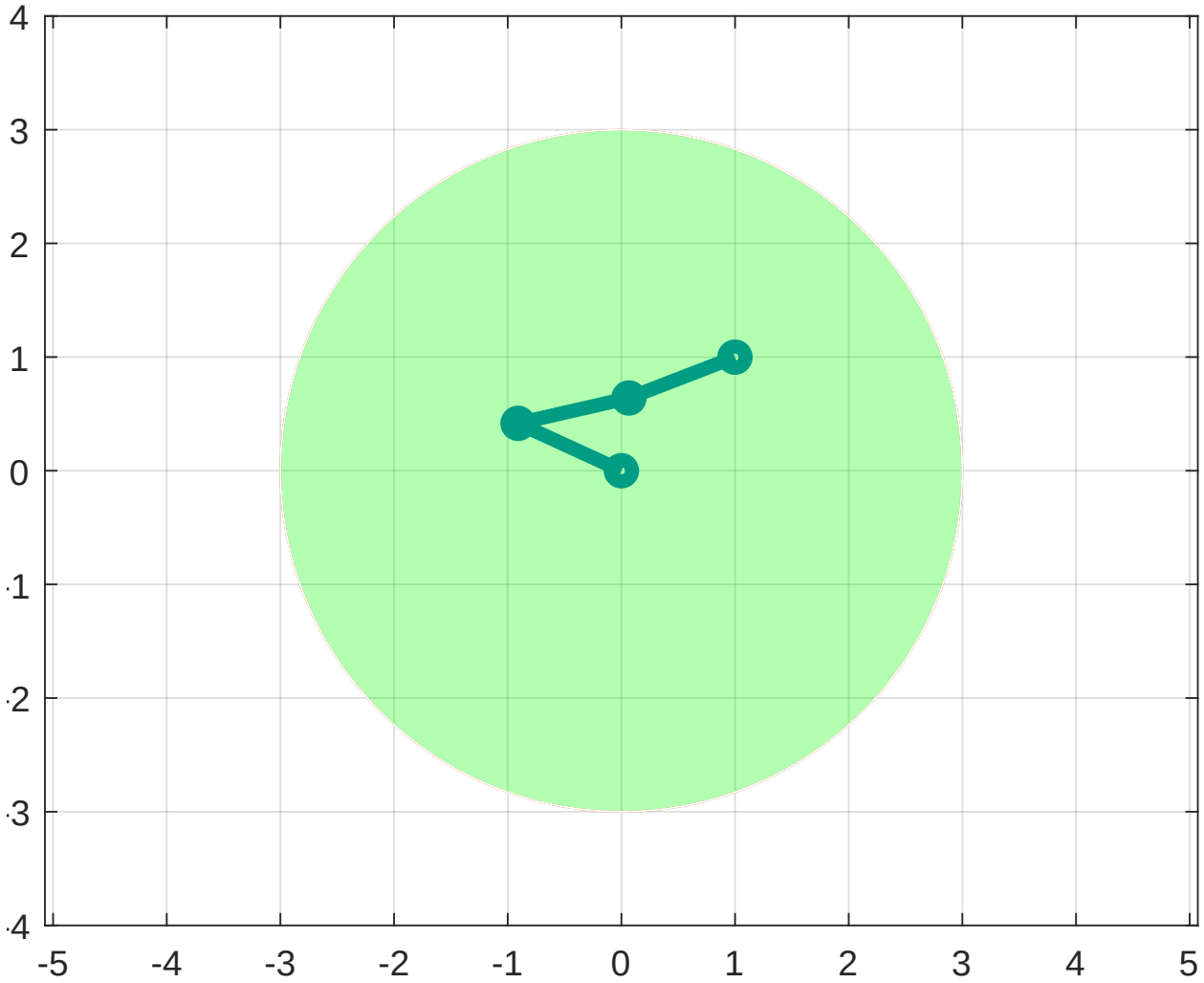
- In real systems, manipulator arm should have speed limitations to avoid injury
- The Equation 1 is modified to avoid a big move in the workspace as follows

$$\Delta q = J^{\dagger} \Delta X \tag{2}$$

  - ΔX is a very small change in the workspace
  - The Equation 2 is valid for small changes in local region

We would like to avoid the manipulator arm to move very fast which is impractical. Instead, we want the end-effector follow a trajectory in the case of large change in local region.

Simulation in MATLAB



Damped Least Square

- So far we used MATLAB function *pinv* for pseudo-inverse which is actually as follows

$$J^{\dagger} = J^T (JJ^T)^{-1}$$

- We make the following adjustment in Equation 1 to stop the vibration

$$\Delta q = J^T \left(JJ^T + \lambda^2 I\right)^{-1} \Delta X \tag{3}$$

  - λ is a damping factor
  - I is an identity matrix

The reason of vibration: when the end-effector tries to reach a point outside its reachable boundary, JJ^T gives a singular matrix. We added a damped least square to JJ^T to prevent the singularity.

Simulation