

EN3563 Robotics Laboratory Experiment 03

Title: DH Parameters, Forward Kinematics and Inverse Kinematics using Robotics Toolbox

1. Introduction

Denavit-Hartenberg (DH) notation is a systematic way of describing the geometry of a serial link manipulators as portrayed in Fig. 1. Forward kinematics is the mapping from joint coordinates, or robot configuration, to end-effector pose. On the other hand, if we know the pose of an object, finding what joint coordinates the robot needs to reach it, is the inverse kinematics problem.

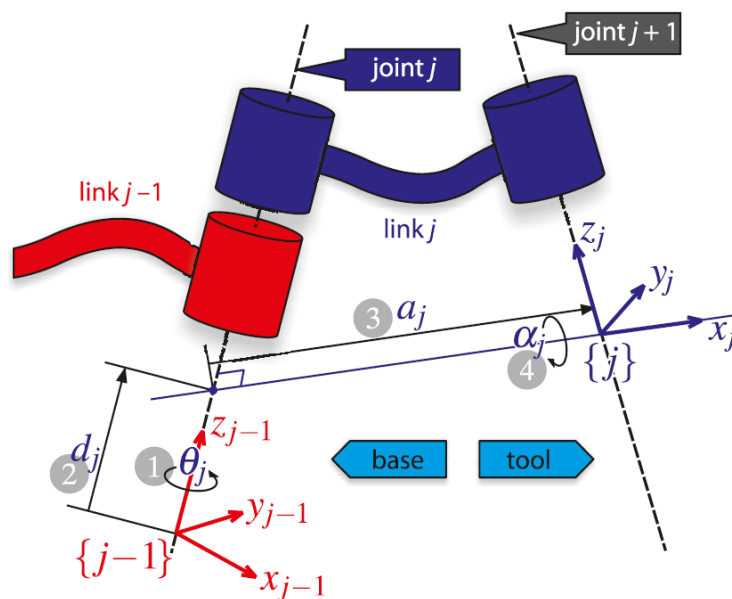


Figure 1: Definition of standard Denavit and Hartenberg link parameters

The Robotics Toolbox is a MATLAB toolbox software that supports research and teaching into arm-type and mobile robotics. It contains functions and classes to represent orientation and pose in 2D and 3D ($SO(2)$, $SE(2)$, $SO(3)$, $SE(3)$) as matrices, quaternions, twists, triple angles, and matrix exponentials. You are expected to have correctly configured the Robotics Toolbox to proceed with the remainder of this laboratory experiment.

From this experiment you will learn the following:

- Usage of MATLAB Robotics Toolbox
- Reinforce the understanding of DH Parameters, Forward Kinematics and Inverse Kinematics.

2. Theory

This section describes the underlying theories associated with DH parameters, forward kinematics and inverse kinematics of serial link manipulators.

2.1. Denavit and Hartenberg (DH) Parameters

The DH convention introduces two constraints.

1. Axis x_i is perpendicular to axis z_{i-1}
2. Axis x_i intersects axis z_{i-1}

Table 1 defines the DH parameters for the link j depicted in Fig. 1.

Table 1: Definitions of DH parameters

Parameter	Name	Definition
a	Link length	Distance between axes z_{i-1} and z_i measured along axis x_i
α	Link twist	Angle between axes z_{i-1} and z_i measured about axis x_i
d	Link offset	Distance from o_{i-1} to the intersection of x_i and z_{i-1} measured along axis z_{i-1} axis
θ	Joint angle	Angle between axes x_{i-1} and x_i measured about axis z_{i-1}

The transformation from link frame $\{j-1\}$ to frame $\{j\}$ can be given in terms of elementary rotations and translation as,

$$A_j^{j-1} = Rot_{z,\theta} \cdot Trans_{z,d} \cdot Trans_{x,a} \cdot Rot_{x,\alpha} \quad (1)$$

Equation (1) can be expanded in homogeneous matrix form as,

$$A_j^{j-1} = \begin{bmatrix} \cos \theta & -\sin \theta \cos \alpha & \sin \theta \sin \alpha & a \cos \theta \\ \sin \theta & \cos \theta \cos \alpha & -\cos \theta \sin \alpha & a \sin \theta \\ 0 & \sin \alpha & \cos \alpha & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

2.2. Spherical Wrist

A Spherical wrist is a combination of three degree-of-freedom revolute joints with all three axes of rotation crossing at a common point. Inverse kinematics of a 6-DoF robot manipulator with a spherical wrist can be simplified, and computed using a closed-form solution.

3. Procedure

Follow the subsequent procedure to carry out kinematics of the well-known Puma 560 6-axis industrial robot manipulator.

- 3.1. Create an instance of a Puma 560 robot.
- 3.2. Find the geometrical structure and DH parameter table of Puma 560 robot.
- 3.3. Find the joint coordinate vectors of Puma 560 robot for the following canonical configurations:
 - a) zero angle b) ready c) stretch d) nominal
- 3.4. Define a tool transform for Puma 560 robot as an extension of 200mm in z-direction in T_6 frame.
- 3.5. Perform forward kinematics for the Puma 560 robot for the canonical configurations in 3.3 and find the position and orientation of tool center point (TCP).
- 3.6. Graphically display realistic plots of Puma 560 robot for the canonical configurations in 3.3.
- 3.7. To perform inverse kinematics, first, reset the tool transform to zero extension in T_6 frame.
- 3.8. For the forward kinematics of “nominal” configuration, perform inverse kinematics.
- 3.9. Experimentally figure out the correct arm geometry that gives you the required nominal configuration out of the following:
 - a) Left hand, elbow up b) Left hand, elbow down
 - c) Right hand, elbow up d) Right hand, elbow down
- 3.10. Give an unreachable point for the Puma 560 inverse kinematics solver, and observe the output.
- 3.11. Run the following MATLAB code and observe Puma 560 motion.

```
close all;
clear all;

mdl_puma560

T1 = SE3(0.8,0,0)*SE3.Ry(pi/2);
T2 = SE3(-0.8,0,0)*SE3.Rx(pi);

q1 = p560.ikine6s(T1);
q2 = p560.ikine6s(T2);

t = [0:0.05:2]';
q = jtraj(q1, q2, t);

p560.plot3d(q);
```

4. MATLAB Robotics Toolbox Reference

4.1. Instantiating Robot Models

`mdl_baxter`: Baxter dual-arm robot model
`mdl_puma560`: Puma 560 robot model
`mdl_stanford`: Stanford arm robot model

4.2. Robot Model Details

Type the `SerialLink` object in MATLAB workspace on the command window to find more details of the instantiated robot model.

e.g. Stanford arm

```
>> mdl_stanford      % instantiates Stanford arm robot model  
>> stanf            % displays more details of instantiated Stanford arm
```

4.3. Robot Model Parameters

You can set/modify certain robot model parameters for the corresponding `SerialLink` object. Double click the `SerialLink` instance to find out available parameters (e.g. base, tool etc.) to be changed in *Property Inspector*.

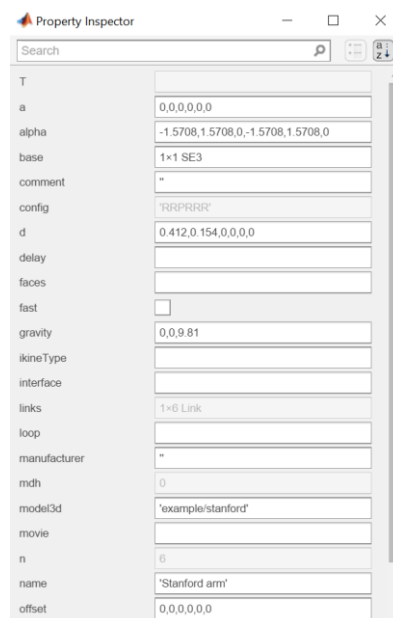


Figure 2: Snapshot of Property Inspector for Stanford arm `SerialLink` instance

e.g. Stanford arm

```
>> mdl_stanford      % instantiates Stanford arm robot model  
>> stanf.base = SE3(0.1, 0.2, 0.3) % shift the base of the Stanford arm  
                                     % using a pure translation
```

4.4. Some Canonical Configurations

`qz`: zero angle
`qr`: ready – the arm is straight and vertical
`qs`: stretch – the arm is straight and horizontal
`qn`: nominal – the arm is in a dexterous working pose

4.5. Robot Kinematics

`fkine`: outputs a homogeneous transformation for the pose of the tool corresponding to the input joint coordinates

e.g. Stanford arm

```
>> mdl_stanford      % instantiates Stanford arm robot model
>> stanf.fkine(q)    % tool forward kinematics for q joint vector
```

`ikine6s`: computes inverse kinematics if the robot is 6-axis with a spherical wrist

`ikine`: computes inverse kinematics numerically

$Q = R.ikine6s(T)$ are the joint coordinates ($1 \times N$) corresponding to the robot tool pose T which is an SE3 object or homogeneous transform matrix (4×4), and N is the number of robot joints.

$Q = R.ikine6s(T, CONFIG)$ specifies the configuration of the arm in the form of a string containing one or more of the configuration codes:

- 'l': arm to the left (default)
- 'r': arm to the right
- 'u': elbow up (default)
- 'd': elbow down
- 'n': wrist not flipped (default)
- 'f': wrist flipped (rotated by 180 deg)

4.6. Visualizing Robot Configurations

e.g. Stanford arm

`stanf.plot(qz)`: visualizes zero angle configuration of Stanford arm

Note:

`plot3d`: creates more realistic looking plots for certain robot models

4.7. Trajectories

`jtraj`: computes a joint space trajectory

e.g. $[q, qd, qdd] = jtraj(q1, q2, t)$ is a joint space trajectory q where the joint coordinates vary from $q1$ to $q2$. A quintic (5th order) polynomial is used with default zero boundary conditions for velocity and acceleration. The number of steps in the trajectory is defined by the length of the time vector t .

1. What is the geometrical structure of the Puma 560 robot?
2. Fill the DH parameter table for Puma 560 robot.

j	θ	d	a	α

3. Joint coordinate vectors of Puma 560 robot for the following canonical configurations:
 - a) zero angle:
 - b) ready :
 - c) stretch :
 - d) nominal :
4. Forward kinematics for tool center point (TCP) in Procedure 3.5 for the canonical configurations.

Configuration	Position	Orientation (Rotation Matrix)
Zero angle		
Ready		
Stretch		
Nominal		

5. Visualization of Puma 560 robot for Procedure 3.6.

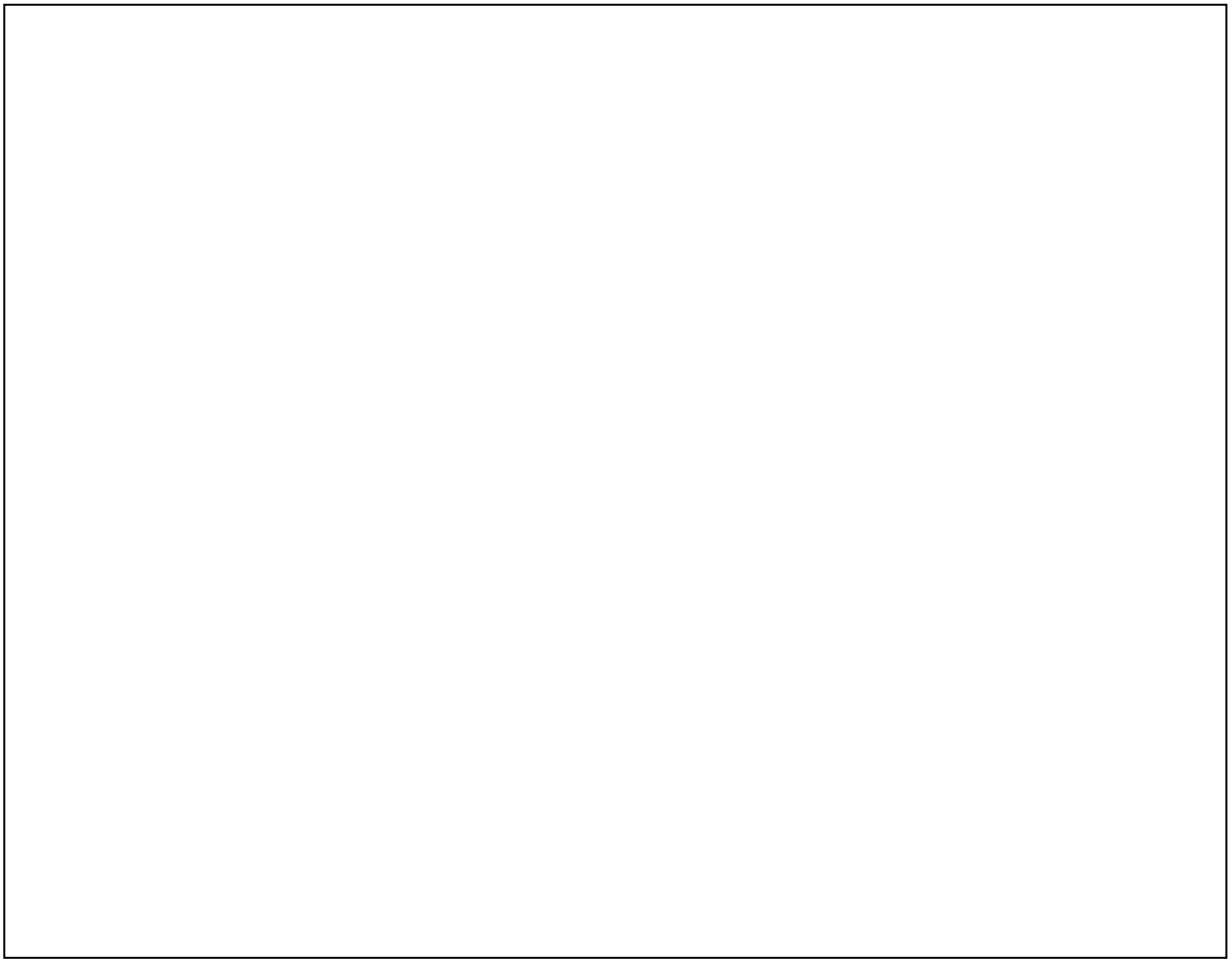
a)	b)
c)	d)

6. Inverse kinematics joint vector for Procedure 3.8. What is your observation?

7. Correct arm configuration for Procedure 3.9.

8. What can be observed for Procedure 3.10?

9. MATLAB code for the entire procedure.

A large, empty rectangular box with a thin black border, intended for the user to provide MATLAB code for the entire procedure.

10. Explain in point form what the MATLAB code in 3.11 does.