Practical 3

Resource:

https://users.cs.duke.edu/~brd/Teaching/Bio/asmb/current/Papers/chap3-forward-kinematics.pdf

Note 1: To do the first 3 exercises, you might need a pen and paper.

Note 2: The robotics toolbox needs to be re-installed before you use it. The installer files can be found on Canvas in the following section:

"PART 2 - Unit 2 - Locomotion Control Strategies of Robots with/without locomotion (T9-22/3/2023)"

Forward Kinematics: The Denavit-Hartenberg (DH) Convention

In this practical we develop the forward or configuration kinematic equations for rigid robots. The forward kinematics problem is concerned with the relationship between the individual joints of the robot manipulator and the position and orientation of the tool or end-effector. Stated more formally, the forward kinematics problem is to determine the position and orientation of the end-effector, given the values for the joint variables of the robot. The joint variables are the angles between the links in the case of revolute or rotational joints, and the link extension in the case of prismatic or sliding joints. The forward kinematics problem is to be contrasted with the inverse kinematics problem, which will be studied in the next practical, and which is concerned with determining values for the joint variables that achieve a desired position and orientation for the end-effector of the robot.

Summary of the DH method steps to solve the forward Kinematic of a robot manipulator:

We may summarize the above procedure based on the D-H convention in the following algorithm for deriving the forward kinematics for any manipulator.

Step l: Locate and label the joint axes $z_0, \ldots, z_{n-1}$.

Step 2: Establish the base frame. Set the origin anywhere on the z_0-axis. The x_0 and y_0 axes are chosen conveniently to form a right-hand frame. For $i = 1, \ldots, n - 1$, perform Steps 3 to 5.

Step 3: Locate the origin O_i where the common normal to z_i and z_{i-1} intersects z_i. If z_i intersects z_{i-1} locate O_i at this intersection. If z_i and z_{i-1} are parallel, locate O_i in any convenient position along z_i.

Step 4: Establish x_i along the common normal between z_{i-1} and z_i through O_i, or in the direction normal to the $z_{i-1} - z_i$ plane if z_{i-1} and z_i intersect.

Step 5: Establish yi to complete a right-hand frame.

Step 6: Establish the end-effector frame on $x_n y_n z_n$. Assuming the n-th joint is revolute, set zn = a along the direction zn−1. Establish the origin On conveniently along zn, preferably at the center of the gripper or at the tip of any tool the manipulator may be carrying. Set yn = s in the direction of the gripper closure and set xn = n as s $\times$ a. If the tool is not a simple gripper set xn and yn conveniently to form a right-hand frame.

Step 7: Create a table of link parameters ai, di, αi, θi.

Joint angle	θ_j	the angle between the x_{j-1} and x_j axes about the z_{j-1} axis	revolute joint variable
Link offset	d_j	the distance from the origin of frame $j-1$ to the x_j axis along the z_{j-1} axis	prismatic joint variable
Link length	a_j	the distance between the z_{j-1} and z_j axes along the x_j axis; for intersecting axes is parallel to $\hat{z}_{j-1} \times \hat{z}_j$	constant
Link twist	α_j	the angle from the z_{j-1} axis to the z_j axis about the x_j axis	constant
Joint type	σ_j	$\sigma = R$ for a revolute joint, $\sigma = P$ for a prismatic joint	constant

Step 8: Form the homogeneous transformation matrices Ai by substituting the above parameters into the following:

$$^{j-1}\mathbf{A}_j = \begin{pmatrix} \cos\theta_j & -\sin\theta_j \cos\alpha_j & \sin\theta_j \sin\alpha_j & a_j \cos\theta_j \\ \sin\theta_j & \cos\theta_j \cos\alpha_j & -\cos\theta_j \sin\alpha_j & a_j \sin\theta_j \\ 0 & \sin\alpha_j & \cos\alpha_j & d_j \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

You need to ensure that you are following the definitions to calculate alpha and theta in the positive direction, as shown below:

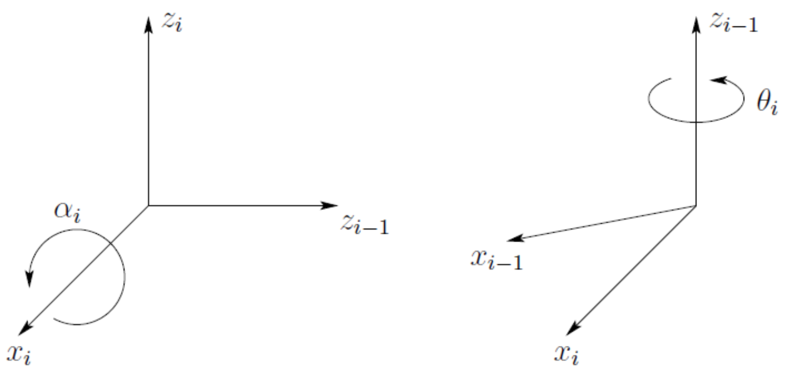


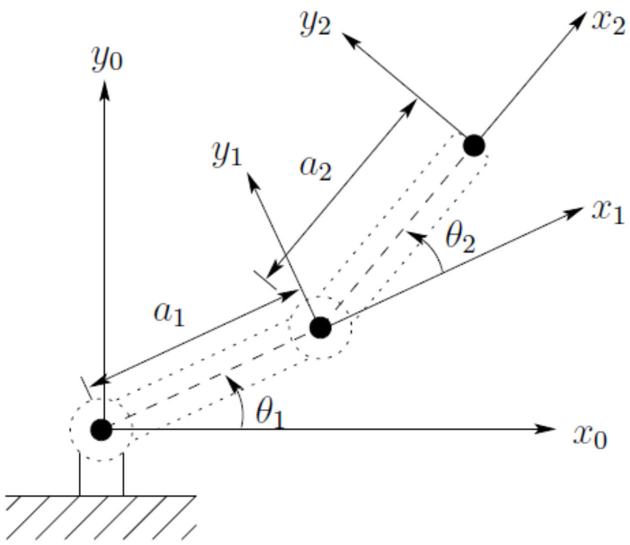
Figure 1 Positive sense for αi and Θi.

Exercise 1: Follow the DH method procedure and solve the forward Kinematics of the following serial manipulator.

Once you calculated the F.K. of this robot, model the robot in MATLAB toolbox and verify your solution by substituting theta1=thata2=45 deg.

Note 1: Derive the DH table in the parametric form, calculate A1, A2, and T. assuming a1=a2=1, and verify your solution using “trMatrixfull.m” MATLAB file. What is the 2D position (x,y) of the end effector?

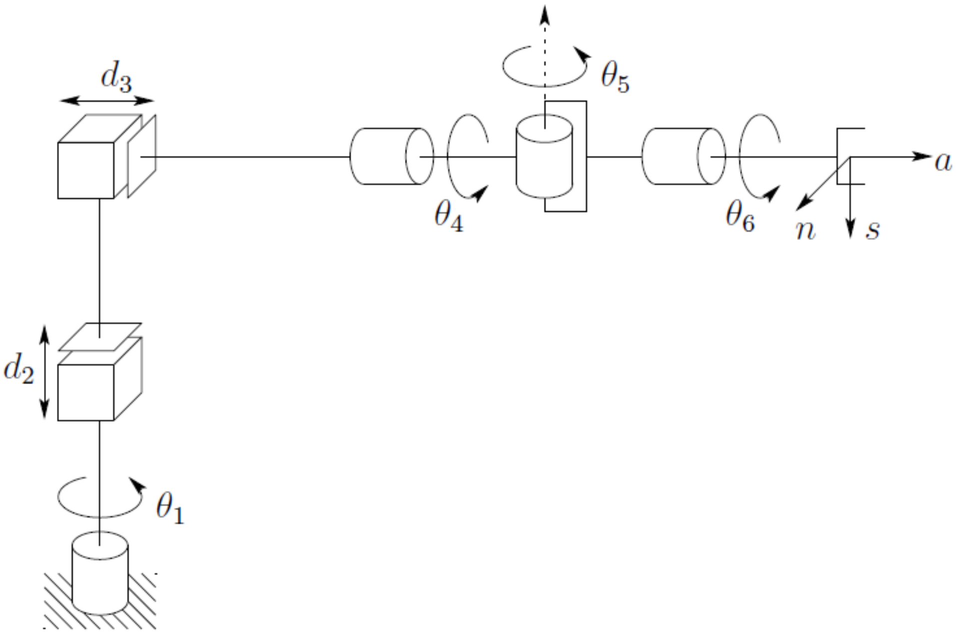
Note 2: “trMatrixfull.m” MATLAB file receives as input the “n: joint number” as well as “DH table” as a matrix where we substitute theta column with 0. It can calculate the “An” matrix as the output.



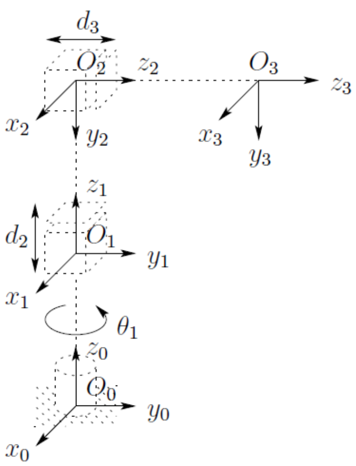
Exercise 2: Follow the DH method procedure and solve the forward Kinematics of the following serial manipulator. Model the overall robot structure in the robotic toolbox and visualise your solution using "robot.teach" function.

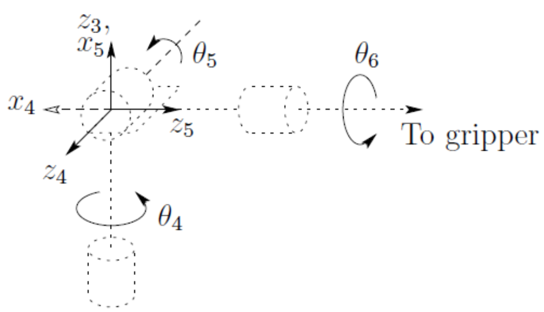
Note 1: Make the required assumption and derive the DH table and then the homogeneous transformation matrix. Use "trMatrixfull.m" Matlab file as well as the Matlab toolbox to verify your solution.

Note 2: To make it easier to solve you can break the robot into 2 sub-components, for example, first 3 joints as one robot and the next 3 joints as well as the end effector as the second robot. The overall T can be derived by multiplying the T1 and T2 of the subcomponents.

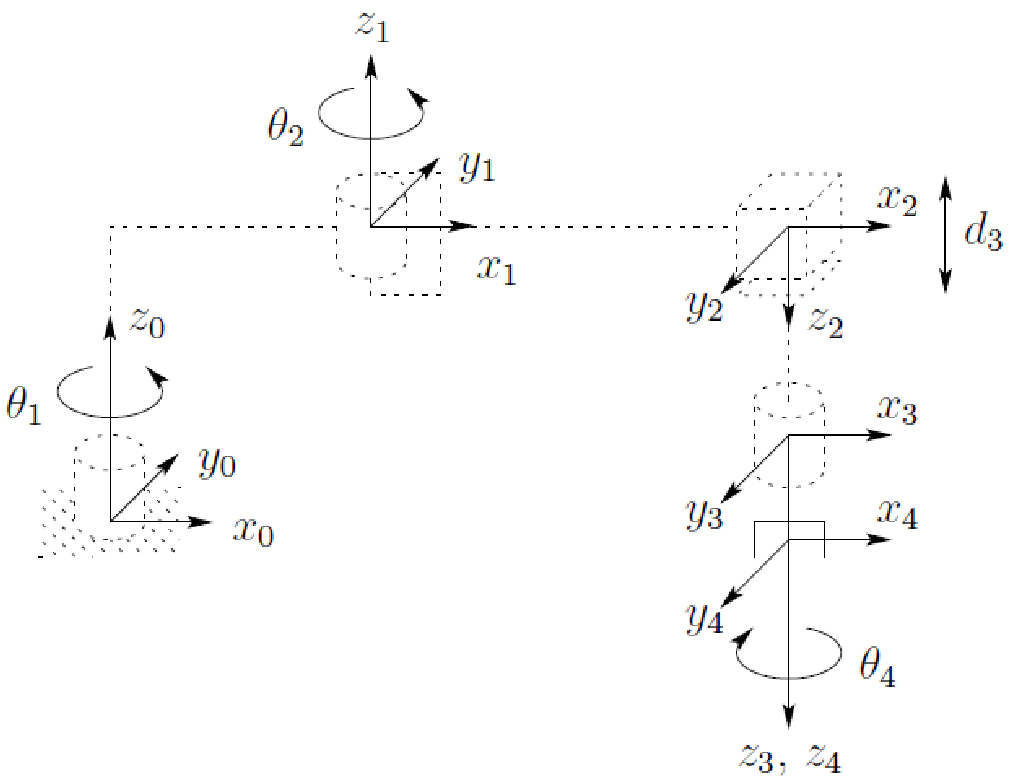


The sub-components are as follows:





Exercise 3: The SCARA robot consists of an RRP arm and a one degree-of-freedom wrist, whose motion is a roll about the vertical axis. Solve the forward Kinematic for this robot using DH method. Verify your implementation and visualise this robot in MATLAB toolbox.



Exercise 4 – Simulate 2-dimensional **one-link** and **two-link** robotic manipulators in MATLAB Simulink and solve the forward Kinematic for theta1 = 30, and theta2=40 to calculate the pose of the end effectors.

To simulate a 2D robot we need to import ETS.*

*>> import ETS2.**

The length of the link is assumed to be one

>> a1 = 1;

Calculate the transformation matrix that is equal to multiplication of rotation matric and translation matrix as follows

*>> E = Rz('q1') * Tx(a1)*

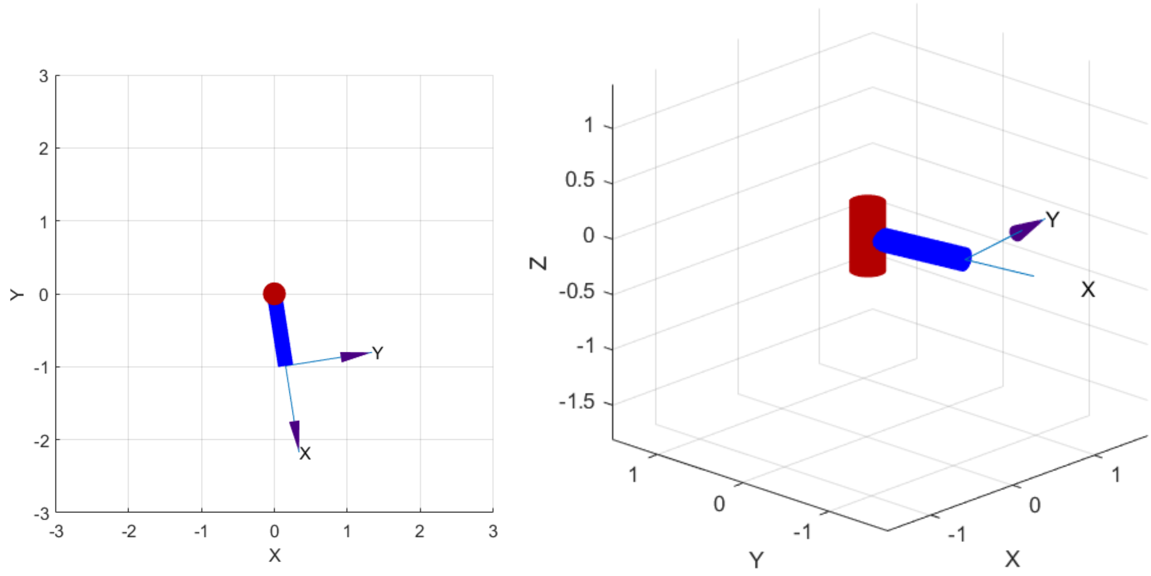
The forward Kinematic matrix is a method of E that can be calculated as follows

>>E.fkine(30, 'deg')

```
ans =
    0.8660   -0.5000     0.866
    0.5000    0.8660       0.5
         0         0         1
```

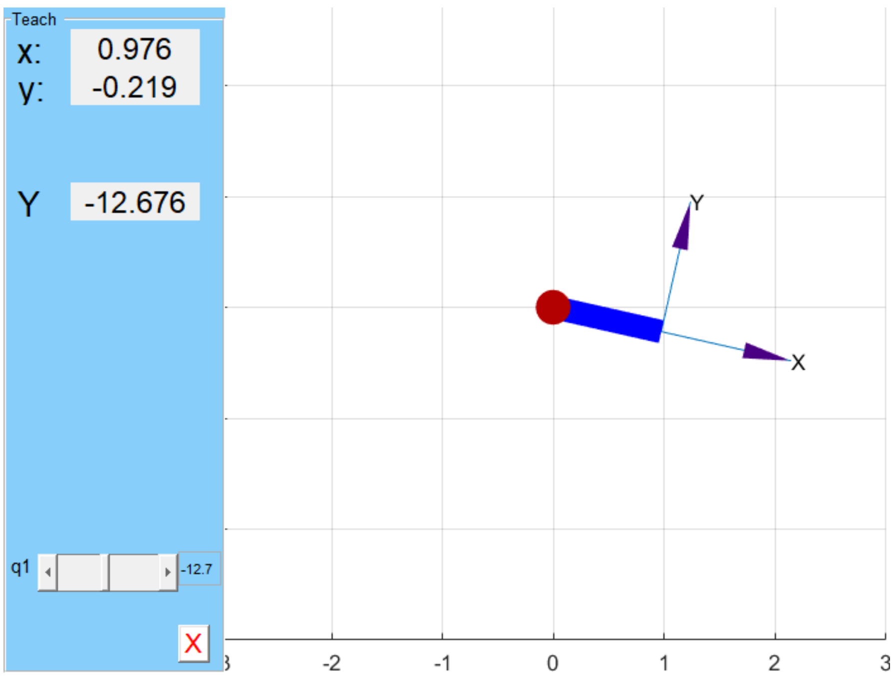
After designing the robot we need to visualise it using the plot command

E.plot([30, 'deg'])



There is another useful method called “teach” that will allow us to obtain the end-effector position while we change the joint angles.

>>E.teach



Now we can simulate a two-link manipulator by extending the one-link one.

>> a1 = 1; a2 = 1;

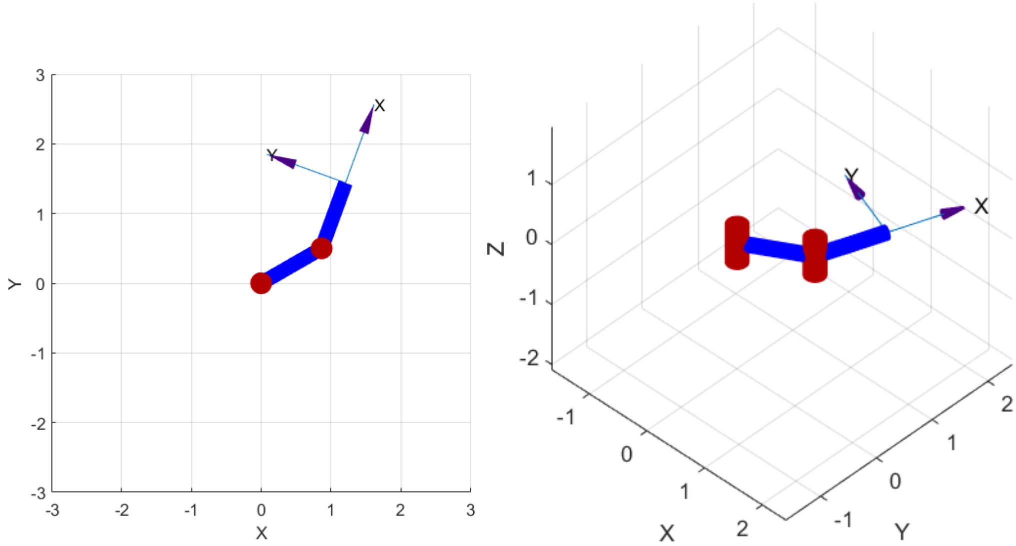
*>> E = Rz('q1') * Tx(a1) * Rz('q2') * Tx(a2)*

E.fkine([30, 40], 'deg')

```
ans =
      0.3420    -0.9397      1.208
      0.9397     0.3420       1.44
           0          0          1
```

And similarly for the visualisation purpose we use the plot command

E.plot([30, 40], 'deg')



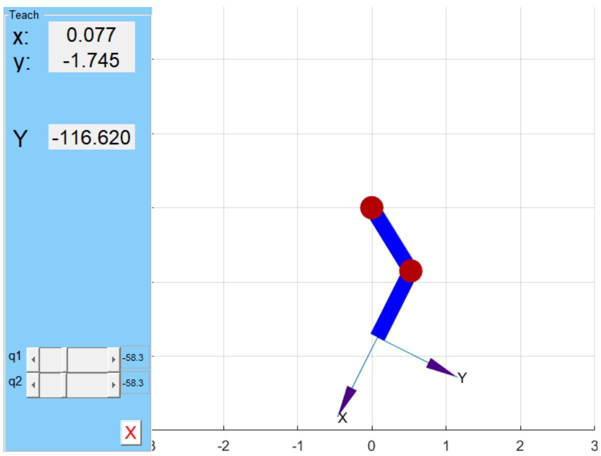
The joint structure of a robot is often referred to by a shorthand comprising the letters R (for revolute) or P (for prismatic) to indicate the number and types of its joints. For this robot

>> E.structure

ans =

RR

Once again, we try the teach “function” for this two-link robot to manipulate its joints angles



Exercise 5 – Visualise a six-DOF robot in 3D space and solve its forward Kinematic problem while all angles are set to be zero. Try the "teach" function as well.

As the robot should work in 3D space we need to import ETS3.*

*>> import ETS3.**

The length of the links are given as follows and can be implemented in MATLAB

>> L1 = 0; L2 = -0.2337; L3 = 0.4318; L4 = 0.0203; L5 = 0.0837; L6 = 0.4318;

And finally, we calculate the transformation matrix of the by multiplying the internal transformation matrices for rotation and translation

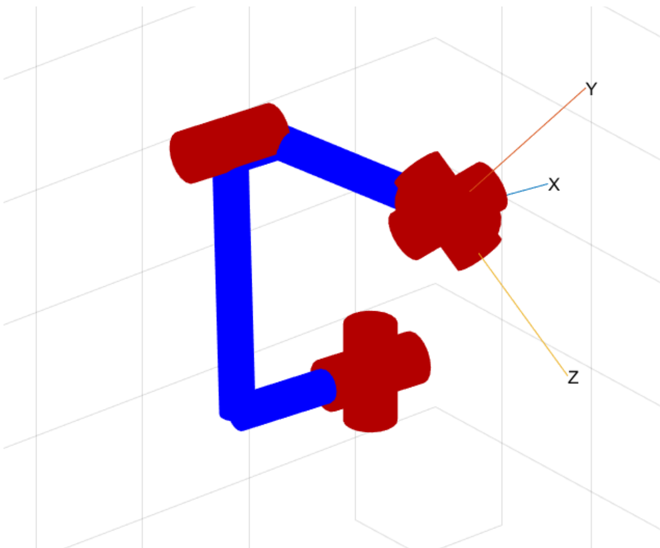
*E3 = Tz(L1) * Rz('q1') * Ry('q2') * Ty(L2) * Tz(L3) * Ry('q3') * Tx(L4) * Ty(L5) * Tz(L6) * Rz('q4') * Ry('q5') * Rz('q6');*

Once the root's transformation matric has been implemented, we can calculate the forward Kinematic *E3.fkine([0 0 0 0 0 0])*

```
ans =
     1          0          0     0.0203
     0          1          0      -0.15
     0          0          1     0.8636
     0          0          0          1
```

As can be seen the forward Kinematic matrix in 3D space is a 4x4 matrix.

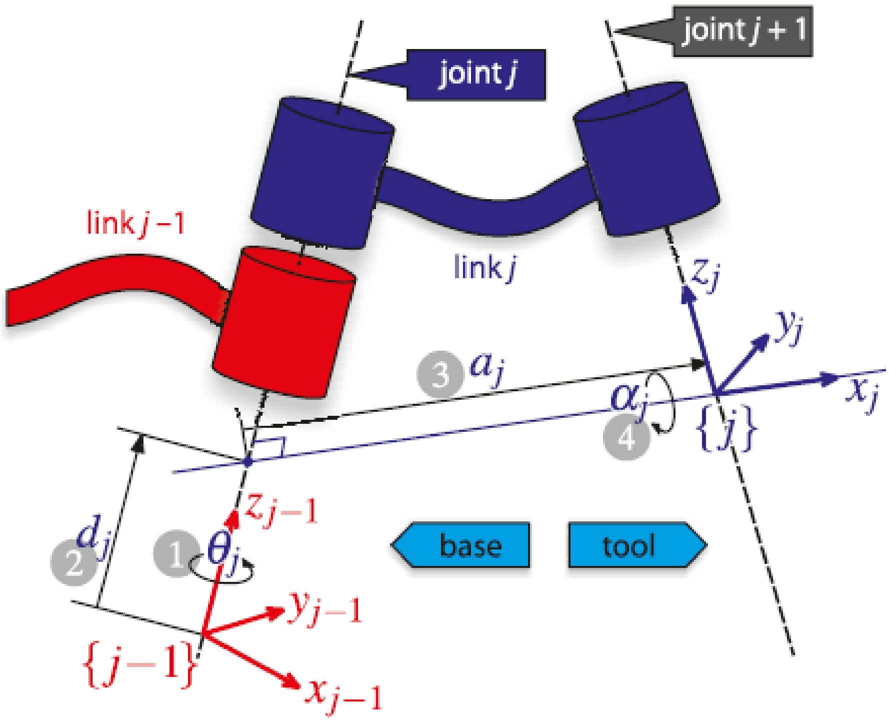
E3.teach



Exercise 6 – When the number of joints increases it is not always easy to derive the robot's transformation matrix by multiplying the internal matrices. The alternative method is to use DH method. In DH method the transformation from link coordinate frame {j − 1} to frame {j} is defined in terms of elementary rotations and translations as

$$^{j-1}\xi_j\left(\theta_j, d_j, a_j, \alpha_j\right) = \mathscr{R}_z\left(\theta_j\right) \oplus \mathscr{T}_z\left(d_j\right) \oplus \mathscr{T}_x\left(a_j\right) \oplus \mathscr{R}_x\left(\alpha_j\right)$$

Joint angle	θ_j	the angle between the x_{j-1} and x_j axes about the z_{j-1} axis	revolute joint variable
Link offset	d_j	the distance from the origin of frame $j-1$ to the x_j axis along the z_{j-1} axis	prismatic joint variable
Link length	a_j	the distance between the z_{j-1} and z_j axes along the x_j axis; for intersecting axes is parallel to $\hat{z}_{j-1} \times \hat{z}_j$	constant
Link twist	α_j	the angle from the z_{j-1} axis to the z_j axis about the x_j axis	constant
Joint type	σ_j	$\sigma = R$ for a revolute joint, $\sigma = P$ for a prismatic joint	constant



which can be expanded in homogeneous matrix form as bellow.

$$
{}^{j-1}\mathbf{A}_j = \begin{pmatrix} \cos\theta_j & -\sin\theta_j\cos\alpha_j & \sin\theta_j\sin\alpha_j & a_j\cos\theta_j \\ \sin\theta_j & \cos\theta_j\cos\alpha_j & -\cos\theta_j\sin\alpha_j & \alpha_j\sin\theta_j \\ 0 & \sin\alpha_j & \cos\alpha_j & d_j \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

Solve the forward Kinematic of the robot using two methods: by calculating T=A01*A12 and comparing your result with the toolbox output as below.

To solve the Kinematics (using the toolbox) using the DH method firstly we need to calculate DH parameters for the robot then use the “SerialLink“ command to create the robot.

robot = SerialLink([Revolute('a', 1) Revolute('a', 1)], 'name', 'my robot')

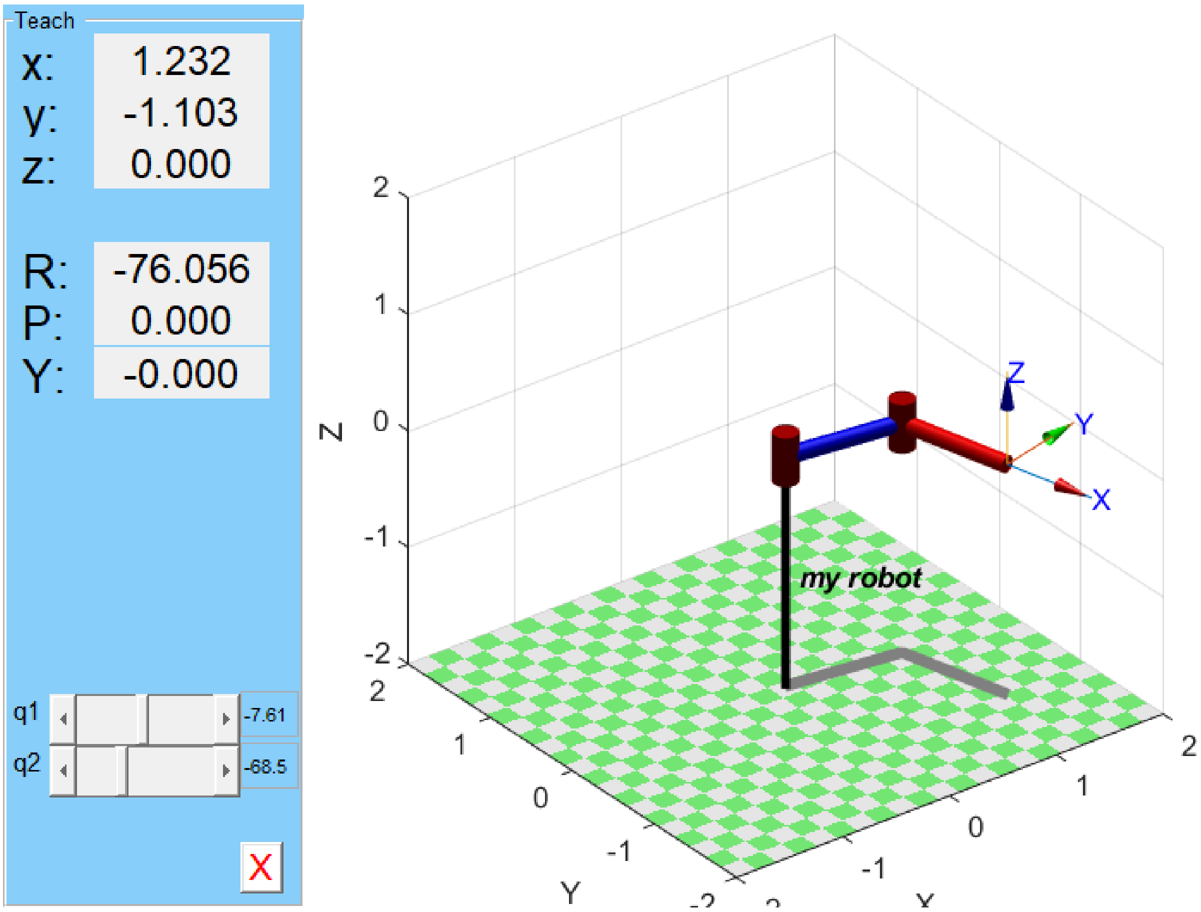
```
robot =
my robot:: 2 axis, RR, stdDH
+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|0          |1          |0          |0          |
|  2|         q2|0          |1          |0          |0          |
+---+-----------+-----------+-----------+-----------+-----------+
```

We have just recreated the 2-robot robot we looked at earlier, but now it is embedded in SE(3). The forward kinematics are

>> robot.fkine([30 40], 'deg')

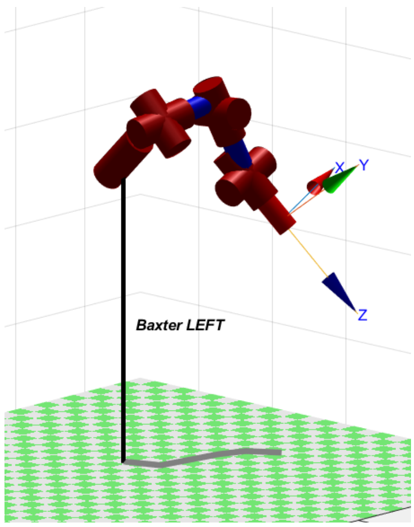
```
ans =
    0.3420   -0.9397         0     1.208
    0.9397    0.3420         0      1.44
         0         0         1         0
         0         0         0         1
```

>>robot.teach



Exercise 4 – There are several pre-defined robotic models that have been already implemented in the toolbox. Use "models" command to load some of them and try to plot them and use the "teach" command to visualise them. For example, load "mdl_nao", "mdl_baxter", "mdl_irb140" and "mdl_puma560", and try to solve the forward Kinematic of this robot with different angle sets.

Use robot.edit command to manipulate the 3D robot and visualise it again.



Exercise 7 – Practice inverse kinematic solution on Puma robot.

First of all, we need to load the predefined model of puma robot and find the name of the robot object in work space

mdl_puma560

p560

```
Puma 560 [Unimation]:: 6 axis, RRRRRR, stdDH, slowRNE
 - viscous friction; params of 8/95;
+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|          0|          0|     1.5708|          0|
|  2|         q2|          0|     0.4318|          0|          0|
|  3|         q3|    0.15005|     0.0203|    -1.5708|          0|
|  4|         q4|     0.4318|          0|     1.5708|          0|
|  5|         q5|          0|          0|    -1.5708|          0|
|  6|         q6|          0|          0|          0|          0|
+---+-----------+-----------+-----------+-----------+-----------+
```

As it is shown puma has 6 revolute joins that we need to initialise as following

qn = [0 0.7854 3.1416 0 0.7854]

and the end effector pose can be calculated using the forward kinematic command

T = p560.fkine(qn)

```
T =
   -0.0000    0.0000    1.0000    0.5963
   -0.0000    1.0000   -0.0000   -0.1501
   -1.0000   -0.0000   -0.0000   -0.0144
         0         0         0    1.0000
```

Since the Puma 560 is a 6-axis robot arm with a spherical wrist we use the method “ikine6s” to compute the inverse kinematics using a closed-form solution. The required joint coordinates to achieve the pose T are

qi = p560.ikine6s(T)

```
qi =
    2.6486   -3.9270    0.0940    2.5326    0.9743    0.3734
```

Surprisingly, these are quite different to the joint coordinates we started with. However if we investigate a little further

p560.fkine(qi)

```
ans =
   -0.0000    0.0000    1.0000    0.5963
    0.0000    1.0000   -0.0000   -0.1500
   -1.0000    0.0000   -0.0000   -0.0144
         0         0         0    1.0000
```

we see that these two different sets of joint coordinates result in the same end-effector pose. In general, there are eight sets of joint coordinates that give the same end-effector pose – as mentioned earlier the inverse solution is not unique.