Exercise 1a: Forward Kinematics of the ABB IRB 120

Prof. Marco Hutter*

Teaching Assistants:
Jan Carius, Jean-Pierre Sleiman, Jan Preisig, Maria Vittoria Minniti

September 23, 2020

Figure 1: The ABB IRW 120 robot arm.

Abstract

The aim of this exercise is to calculate the forward and inverse kinematics of an ABB robot arm. In doing so, you will practice the use of different representations of the end-effector's orientation as well as how to check whether your implementations are correct. Essentially, the task is to implement the functions for computing the forward and inverse kinematics using symbolic and numerical computations in MATLAB. A separate MATLAB script will be provided for the 3D visualization of the robot arm.

*original contributors include Michael Blösch, Dario Bellicoso, and Samuel Bachmann

1 Introduction

The following exercise is based on an ABB IRB 120 depicted in Figure 2. It is a 6-link robotic manipulator with a fixed base. During the exercise you will implement several different MATLAB functions, which you should test carefully since the following tasks are often dependent on them. To help you with this, use the provided script prototypes (download from Piazza).

2 Forward Kinematics

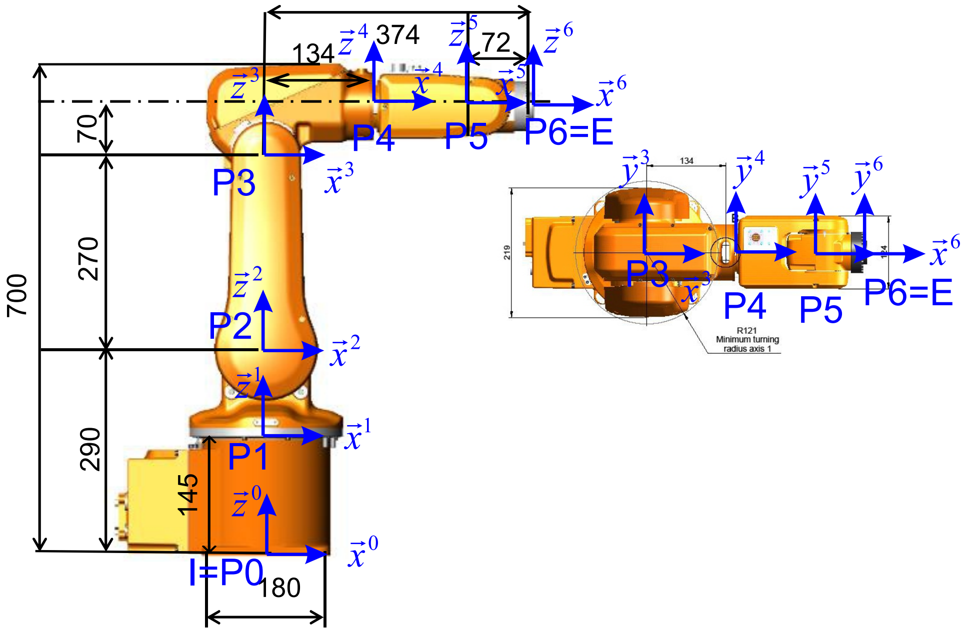

Figure 2: ABB IRB 120 with coordinate systems and joints. The units are mm.

Throughout this document, we will employ I for denoting the inertial world coordinate system (coordinate system P0 in Figure 2) and E for the coordinate system attached to the end-effector (coordinate system P6 in Figure 2).
You should always check your solutions with the provided script `evaluate_problems.m`. This script compares your implementation with our solution on random data points.

Exercise 2.1

Define a vector $\mathbf{q}$ of generalized coordinates to describe the configuration of the ABB IRB120. Recall that generalized coordinates should be *complete* and *independent*. The former property means that they should fully described the configuration of the robot while at the same time comprising a minimal set of coordinates. The latter property refers to the fact that the each generalized coordinate must not be a function of any of the others.

Solution 2.1

The generalized coordinates can be chosen as the single joint angles between subsequent links. Any other *complete* and *independent* linear combination of the single joint angles is also a valid solution.

$$\mathbf{q} = (q_1, \ldots, q_6)^T \in \mathbb{R}^6 \tag{1}$$

Exercise 2.2

Assume from here on that the generalized coordinates $\mathbf{q}$ are the joint angles of the robot arm numbered according to Figure 2. Positive angles imply rotations around the positive coordinate axis.
Compute the homogeneous transformations matrices $\mathbf{T}_{k-1,k}(q_k)$, $\forall\, k = 1, \ldots, 6$. Additionally, find the constant homogeneous transformations between the inertial frame and frame 0 ($\mathbf{T}_{I0}$) and between frame 6 and the end-effector frame ($\mathbf{T}_{6E}$). Please implement the following functions (i.e., replace the zero assignments with your solution):

```
function TI0 = getTransformI0()
  % Input: void
  % Output: homogeneous transformation Matrix from frame 0 to the ...
      inertial frame I. T_I0

  % PLACEHOLDER FOR OUTPUT -> REPLACE WITH SOLUTION
  TI0 = zeros(4);
end

function T01 = jointToTransform01(q)
  % Input: joint angles
  % Output: homogeneous transformation Matrix from frame 1 to frame ...
      0. T_01

  % PLACEHOLDER FOR OUTPUT -> REPLACE WITH SOLUTION
  T01 = zeros(4);
end

function T12 = jointToTransform12(q)
  % Input: joint angles
  % Output: homogeneous transformation Matrix from frame 2 to frame ...
      1. T_12

  % PLACEHOLDER FOR OUTPUT -> REPLACE WITH SOLUTION
  T12 = zeros(4);
end

function T23 = jointToTransform23(q)
  % Input: joint angles
  % Output: homogeneous transformation Matrix from frame 3 to frame ...
      2. T_23

  % PLACEHOLDER FOR OUTPUT -> REPLACE WITH SOLUTION
  T23 = zeros(4);
end

function T34 = jointToTransform34(q)
  % Input: joint angles
  % Output: homogeneous transformation Matrix from frame 4 to frame ...
      3. T_34

  % PLACEHOLDER FOR OUTPUT -> REPLACE WITH SOLUTION
  T34 = zeros(4);
end


function T45 = jointToTransform45(q)
  % Input: joint angles
  % Output: homogeneous transformation Matrix from frame 5 to frame ...
      4. T_45

  % PLACEHOLDER FOR OUTPUT -> REPLACE WITH SOLUTION
  T45 = zeros(4);
```

```
end

function T56 = jointToTransform56(q)
  % Input: joint angles
  % Output: homogeneous transformation Matrix from frame 6 to frame ...
      5. T_56

  % PLACEHOLDER FOR OUTPUT -> REPLACE WITH SOLUTION
  T56 = zeros(4);
end

function T6E = getTransform6E()
  % Input: void
  % Output: homogeneous transformation Matrix from the end-effector ...
      frame E to frame 6. T_6E

  % PLACEHOLDER FOR OUTPUT -> REPLACE WITH SOLUTION
  T6E = zeros(4);
end
```

Solution 2.2

Remember that a homogeneous transformation matrix is expressed in the form

$$\mathbf{T}_{hk}(q_k) = \begin{bmatrix} \mathbf{C}_{hk}(q_k) & {}_h\mathbf{r}_{hk}(q_k) \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix}. \tag{2}$$

For the ABB IRB 120, each $\mathbf{T}_{hk}(q_k)$ is composed by an elementary rotation a single joint axis and a translation defined by the manipulator kinematic parameters. By defining the elementary rotations matrices about each axis as

$$\mathbf{C}_z(\varphi) = \begin{bmatrix} \cos(\varphi) & -\sin(\varphi) & 0 \\ \sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$\mathbf{C}_y(\varphi) = \begin{bmatrix} \cos(\varphi) & 0 & \sin(\varphi) \\ 0 & 1 & 0 \\ -\sin(\varphi) & 0 & \cos(\varphi) \end{bmatrix} \tag{4}$$

$$\mathbf{C}_x(\varphi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\varphi) & -\sin(\varphi) \\ 0 & \sin(\varphi) & \cos(\varphi) \end{bmatrix}, \tag{5}$$

one can write

$$\mathbf{T}_{01}(q_1) = \begin{bmatrix} \mathbf{C}_z(q_1) & {}_0\mathbf{r}_{01}(q_1) \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \tag{6}$$

$$\mathbf{T}_{12}(\theta_2) = \begin{bmatrix} \mathbf{C}_y(q_2) & {}_1\mathbf{r}_{12}(q_2) \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \tag{7}$$

$$\mathbf{T}_{23}(q_3) = \begin{bmatrix} \mathbf{C}_y(q_3) & {}_2\mathbf{r}_{23}(q_3) \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \tag{8}$$

$$\mathbf{T}_{34}(q_4) = \begin{bmatrix} \mathbf{C}_x(q_4) & {}_3\mathbf{r}_{34}(q_4) \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \tag{9}$$

$$\mathbf{T}_{45}(q_5) = \begin{bmatrix} \mathbf{C}_y(q_5) & {}_4\mathbf{r}_{45}(q_5) \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \tag{10}$$

$$\mathbf{T}_{56}(q_6) = \begin{bmatrix} \mathbf{C}_x(q_6) & {}_5\mathbf{r}_{56}(q_6) \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix}. \tag{11}$$

Finally, the constant homogeneous transformations $\mathbf{T}_{I0}$ and $\mathbf{T}_{6E}$ are simply the identity matrix $\mathbf{I}_{4\times4}$.

Exercise 2.3
Find the end-effector position vector ${}_I\mathbf{r}_{IE} = {}_I\mathbf{r}_{IE}(\mathbf{q})$. Please implement the following function:

```
function I_r_IE = jointToPosition(q)
  % Input: joint angles
  % Output: position of end-effector w.r.t. inertial frame. I_r_IE

  % PLACEHOLDER FOR OUTPUT -> REPLACE WITH SOLUTION
  I_r_IE = zeros(3,1);
end
```

Solution 2.3
The end-effector position is given by the direct kinematics, represented in matrix form by the homogeneous transformation $\mathbf{T}_{IE}(\mathbf{q})$, which can be found by successive concatenation of coordinate frame transformations.

$$\mathbf{T}_{IE}(\mathbf{q}) = \mathbf{T}_{I0} \cdot \left(\prod_{k=1}^{6} \mathbf{T}_{k-1,k} \right) \cdot \mathbf{T}_{6E} = \begin{bmatrix} \mathbf{C}_{IE} & {}_I\mathbf{r}_{IE}\,(\mathbf{q}) \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{12}$$

The end-effector position can then be found by selecting the fourth column of $\mathbf{T}_{IE}(\mathbf{q})$.

$$\begin{bmatrix} \mathbf{r}\,(\mathbf{q}) \\ 1 \end{bmatrix} = \mathbf{T}_{IE}(\mathbf{q}) \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{13}$$

Exercise 2.4

What is the end-effector position for $\mathbf{q} = \begin{pmatrix} \pi/6 \\ \pi/6 \\ \pi/6 \\ \pi/6 \\ \pi/6 \\ \pi/6 \end{pmatrix}$?
Use Matlab (`abbRobot.setJointPositions(q)`) to visualize it.

Solution 2.4
From the direct kinematics equations found earlier, it is:

$${}_I\mathbf{r}_{IE} = {}_I\mathbf{r}_{IE}\,(\mathbf{q}) = \begin{bmatrix} 0.2948 \\ 0.1910 \\ 0.2277 \end{bmatrix}. \tag{14}$$

Exercise 2.5
Find the end-effector rotation matrix $\mathbf{C}_{IE} = \mathbf{C}_{IE}\,(\mathbf{q})$. Please implement the following function:

```
function C_IE = jointToRotMat(q)
  % Input: joint angles
  % Output: rotation matrix which projects a vector defined in the
  % end-effector frame E to the inertial frame I, C_IE.

  % PLACEHOLDER FOR OUTPUT -> REPLACE WITH SOLUTION
```

```
  C_IE = zeros(3);
end
```

Solution 2.5

From the structure of the direct kinematics equations found earlier, it follows that the end-effector rotation matrix is obtained by extracting the first three rows and the first three columns from $\mathbf{T}_{IE}(\mathbf{q})$. This operation can be compactly written in matrix form:

$$\mathbf{C}_{IE}\left(\mathbf{q}\right) = \begin{bmatrix}\mathbf{I}_{3\times3} & \mathbf{0}_{3\times1}\end{bmatrix} \cdot \mathbf{T}_{IE}(\mathbf{q}) \cdot \begin{bmatrix}\mathbf{I}_{3\times3} \\ \mathbf{0}_{1\times3}\end{bmatrix}. \tag{15}$$

Exercise 2.6

Find the quaternion representing the attitude of the end-effector $\xi_{IE} = \xi_{IE}\left(\mathbf{q}\right)$. Please also implement the following function:

- Two functions for converting from quaternion to rotation matrices and vice-versa. Test these by converting from quaternions to rotation matrices and back to quaternions.
- The quaternion multiplication $\boldsymbol{q} \otimes \boldsymbol{p}$
- The passive rotation of a vector with a given quaternion. This can be implemented in different ways which can be tested with respect to each other. The easiest way is to transform the quaternion to the corresponding rotation matrix (by using the function from above) and then multiply the matrix with the vector to be rotated.

Also check that your two representations for the end-effector orientation match with each other. In total you should write the following five functions:

```
function quat = jointToQuat(q)
  % Input: joint angles
  % Output: quaternion representing the orientation of the end-effector
  % q_IE.

  % PLACEHOLDER FOR OUTPUT -> REPLACE WITH SOLUTION
  quat = zeros(4,1);
end

function R = quatToRotMat(q)
  % Input: quaternion [w x y z]
  % Output: corresponding rotation matrix

  % PLACEHOLDER FOR OUTPUT -> REPLACE WITH SOLUTION
  R = zeros(3);
end

function q = rotMatToQuat(R)
  % Input: rotation matrix
  % Output: corresponding quaternion [w x y z]

  % PLACEHOLDER FOR OUTPUT -> REPLACE WITH SOLUTION
  q = zeros(4,1);
end

function q_AC = quatMult(q_AB,q_BC)
  % Input: two quaternions to be multiplied
  % Output: output of the multiplication

```

```
  % PLACEHOLDER FOR OUTPUT -> REPLACE WITH SOLUTION
  q_AC = zeros(4,1);
end

function B_r = rotVecWithQuat(q_BA,A_r)
  % Input: the orientation quaternion and the coordinate of the ...
      vector to be mapped
  % Output: the coordinates of the vector in the target frame

  % PLACEHOLDER FOR OUTPUT -> REPLACE WITH SOLUTION
  B_r = zeros(3,1);
end
```

Solution 2.6

$$\mathbf{C}_{IE}(\mathbf{q}) = \mathbf{C}_{I0} \cdot \left(\prod_{k=1}^{6} \mathbf{C}_{k-1,k} \right) \cdot \mathbf{C}_{6E} \tag{16}$$

$$\xi(\mathbf{q}) = \frac{1}{2} \begin{pmatrix} \sqrt{c_{11} + c_{22} + c_{33} + 1} \\ sgn(c_{32} - c_{23})\sqrt{c_{11} - c_{22} - c_{33} + 1} \\ sgn(c_{13} - c_{31})\sqrt{c_{22} - c_{33} - c_{11} + 1} \\ sgn(c_{21} - c_{12})\sqrt{c_{33} - c_{11} - c_{22} + 1} \end{pmatrix} \tag{17}$$

where $c_{ij} = \mathbf{C}(i,j)$.