Dynamic pick and place using ABB IRB 140

Siddhesh Rane - 116147286

December 19, 2018

Abstract

In this paper, I have derived a kinematic model of the ABB IRB 140 industrial robot arm to perform the dynamic pick and place operation. First, forward kinematics is derived in which I have constructed the forward transformation matrices using DH parameter tables. Then I have shown the validity of the model by constructing a replica of the ABB IRB140 robot in VREP simulation software and compared it's results with the results of the official ABB IRB140 robot given in the instructions manual of the product. Then I have derived the Jacobian matrices for the robot.

1 Introduction

For past few decades, it has been observed that the industrial automation is on the constant upsurge owing to advancements in the robotics technology. Due to ever increasing demands for various industrial products, many industries have adopted high automation and robotics to increase their production rate. Industrial automation is about the high quality, fast speed and increased productivity, hence it is not surprising that human workers are being replaced by industrial robot arms to carry out the tasks at faster rates and with more precision.These robotic arms cut down on worker error and labor costs. For this project, I have used the IRB140 industrial robotic arm by ABB. The ABB IRB 140 is a compact, 6 axis multipurpose robot with a payload of 6 kg and a reach of 810 mm. It offers 360 degrees of rotation, fast acceleration, a large working envelope, and the flexibility to be floor, suspended, or wall mounted at any angle. [1].

2 Motivation

2.1 Why industrial robot arm?

As explained in the introduction section, industrial robots are taking over the conventional human work and as a result demand for industrial robotics products is increasing at a very high rate. Thus, as a robotics engineer it is crucial to familiarize oneself with the workings and intricacies of the industrial robotic arms. Hence I wanted to do a project on an industrial robot arm and study its properties and working.

2.2 Issues with the current system.

Most commonly used industrial robots are the robotic arms which are used for repetitive tasks such as pick and place operations.Since robots are exceptionally good at monotonous repetitive tasks, these robot arms bring numerous benefits to the manufacturer due to their speed and consistency. There are two types of the pick and place systems used in the industry. First is **Fixed pick and place (FPP)** and another one is **Dynamic pick and place (DPP)**. FPP system has the following drawbacks:
1. Most important drawback of the FPP robot arm is that it has a large idle time i.e. it has to wait until the target object reaches the desired location before it can pick it up. As a result it delays the production process.
2. It is not reliable as it cannot handle objects which have different velocity than it is programmed for. i.e any mismatch in the timing of object motion and robot arm can cause an error.
3. Overall production cost is increased as one FPP robot arm can only do one specific job at a time, total number of robots for different tasks is increased which is responsible for the raise in the production cost.

For an example, most widely used application of the pick and place robots is the assembly of the PCB. Due to huge demand of the electronics devices, timing is the most crucial parameter in the PCB production. In case of FPP assembly, the component magazine and PCB board both move to the fixed predefined location before the robot arm can actually start assembling it. This increases the overall robotic assembly time. Experimental results showed that the proposed dynamic pick-and-place (DPP) approach is superior to the fixed pick-and-place (FPP) approach in nearly all cases.

2.3 Why dynamic pick and place?

As explained in the previous section, to avoid wasting of the assembly time, DPP approach is used that enables the robot to make dynamic decisions of the pick-and-place locations for retrieving and inserting components DPP models make the robots more flexible. In DPP, robot arms dynamically decide best pick up location and go to that location before the carrier can bring a desired component to the predefined ideal pick up location. Compared with the FPP it is observed that this approach increases the total assembly time by 25 percent. Also another most important advantage of this method is that it increases the scope of the robot usage. i.e. not only DPP robots can perform tasks faster, they need not to be hard coded for a particular application involving fixed timings and fixed locations. Also any mismatch in the object's location or velocity does not affect the performance of the DPP robot.

3 Assumptions

Following assumptions were made during modeling of the project:
1.CAD model used has same dimensions as the real ABB IR140 model.
2.Dynamic properties and velocity constraints on the joints are not considered.
3.Energy consumption is not considered.
4.Simulation environment is free from humans and hence human safety factors

Figure 1: ABB IRB140 model. [2].

are completely ignored in this project.
5.All target objects have same dimensions. Only orientation and location on conveyor belt are unknown
6.This project does not account for obstacles in the path while calculating trajectory. Hence it can only be applied in obstacle free scenarios.
7. End effector forces and joint torques are not used while creating the models. As a result inertial tensors for each link is also ignored.
8. Links in the simulation are not collidable.

4 Robot Modeling

4.1 ABB IRB140 model creation.

Robot model was created by referring to the official documentation of the IRB140 manipulator [2]. Following figure shows the actual arm structure along with the axes.

For replicating this model, a basic robot arm model was created using VREP. Figure 2 shows the basic model of the ABB IRB140 which included solid joints and revolute joints. Dimensions of the joints are exactly equal to the original robot arm. Dimensions of the IRB140 are shown below in figure 3.

4.2 Frame Assignment

Right handed coordinates were used for the axis assignment of the frames of the robot. And frames were assigned in such way that they satisfy the following DH parameter conditions:
1. Axis x_n intersects z_{n-1} axis.
2. Axis x_n is perpendicular to z_{n-1} axis. Following figure 4 shows the frame assignment used to create the IRB 140 model.

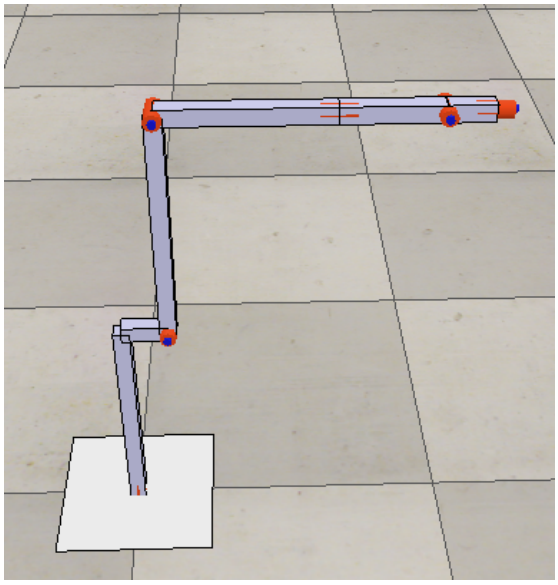

Figure 2: ABB IRB140 VREP model. [2].

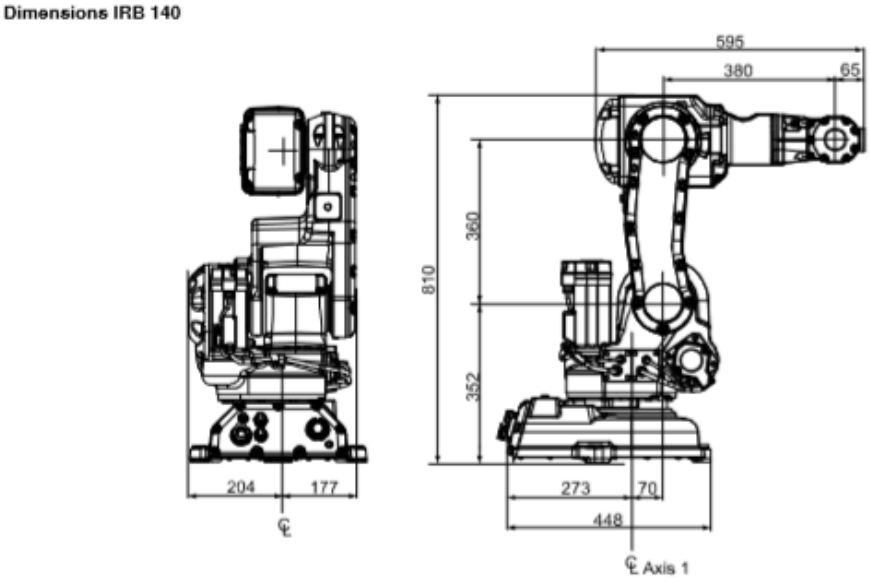

Figure 3: ABB IRB 140 dimensions. [2].

DH Table:

T	θ_i	i	i	α_i
0 to 1	θ_1	L_1	L_2	-90
1 to 2	$\theta_2 - 90$	0	L_3	0
2 to 3	$\theta_3 + 180$	0	0	90
3 to 4	θ_4	L4	0	-90
4 to 5	θ_5	0	0	90
5 to 6	θ_6	L5	0	0

Transformation Matrices:
Using standard form of DH matrix and DH table calculated above, transformation matrices for all frames are calculated as shown below: For simplicity of writing lets take:$\cos\theta_1 = c1, \sin\theta_1 = s1, \cos\theta_2 = c2, \sin\theta_2 = s2, \cos\theta_3 = c3, \sin\theta_3 = s3, \cos\theta_4 = c4, \sin\theta_4 = s4, \cos\theta_5 = c5, \sin\theta_5 = s5, \cos\theta_6 = c6, \sin\theta_6 = s6$

$$A_1^0 = \begin{bmatrix} c_1 & 0 & -s_1 & L_2c_1 \\ s_1 & 0 & c_1 & L_2s_1 \\ 0 & -1 & 0 & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

$$A_2^1 = \begin{bmatrix} s_2 & c_2 & 0 & L_3s_2 \\ -c_2 & s_2 & 0 & -L_3c_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

$$A_3^2 = \begin{bmatrix} -c_3 & 0 & -s_3 & 0 \\ -s_3 & 0 & c_3 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$A_4^3 = \begin{bmatrix} c_4 & 0 & -s_4 & 0 \\ s_4 & 0 & c_4 & 0 \\ 0 & -1 & 0 & L_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$A_5^4 = \begin{bmatrix} c_5 & 0 & s_5 & 0 \\ s_5 & 0 & -c_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

$$A_6^5 = \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 1 & L_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

After getting transformation matrices between consecutive frames, finally each frame is expressed in zeroth frame using following relation: $T_n^0 = T_{n-1}^0 T_n^{n-1}$ Hence calculation of transformation matrix of each frame w.r.t. 0 frame is given in MATLAB code given below.

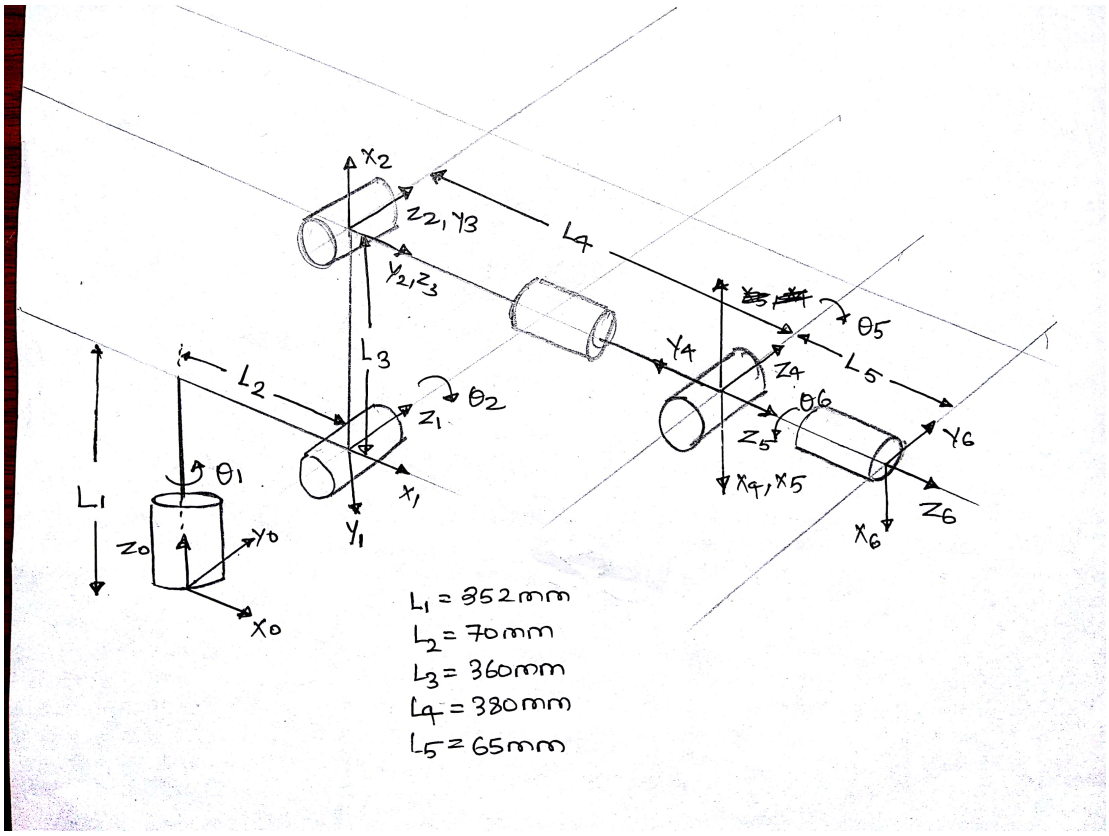

Figure 4: ABB IRB140 frame assignment. [2].

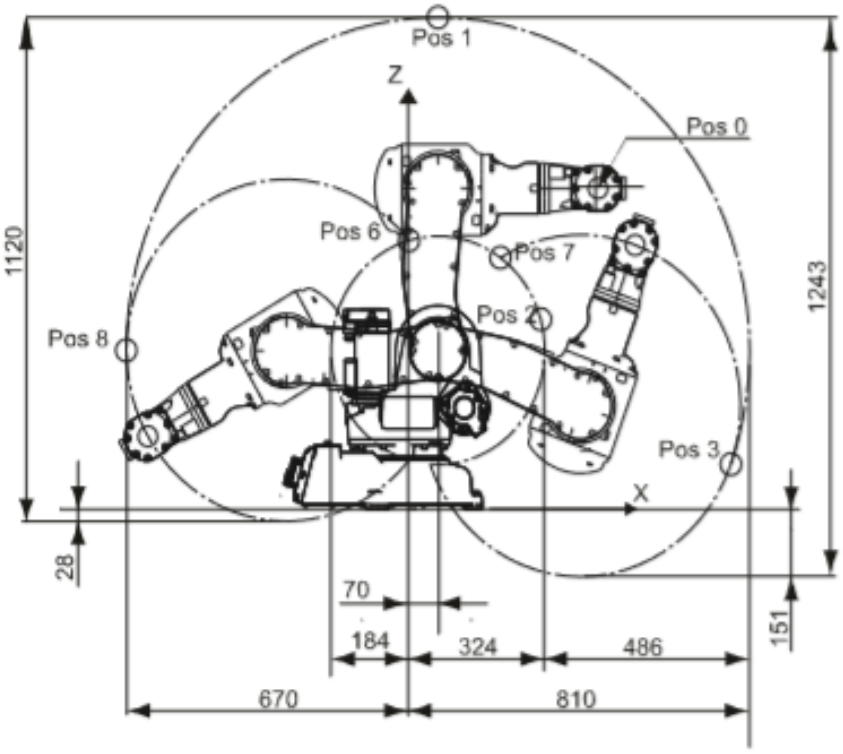

Figure 5: Workspace positions given for ABB IRB 140. [2].

4.3 Forward Kinematics Validation

Validation of the VREP model is done in following way:
1. Using dimensions of ABB IRB given in the official document, a simple model of the robot was created in the VREP.
2. Following **figure 4** is from official documentation of the IRB140 robot. It shows various positions of the robot to define the workspace of the robot (please check paper [2] page 32).

Also, following **figure 5** shows the angles entered to get these positions.

Hence I validated my model by entering the same angles and checked the position of the end-effector and matched the results with the official documentation. Coordinates for the end effectors of my model matched with the coordinates of the original product given in the documentation. Following table shows the outputs I got for each position. **DH Table:**

Position (see Figure 12)	Position (mm) X	Position (mm) Z	Angle (degrees) Axis 2	Angle (degrees) Axis 3
0	450	712	0	0
1	70	1092	0	-90
2	314	421	0	+50
3	765	99	110	-90
6	1	596	-90	+50
7	218	558	110	-230
8	-670	352	-90	-90

Figure 6: Angles given in the documentation to get each position along with it's coordinates. [2].

Pos 0 z=0.7140	q2=0	q3=0	x=0.4500m	y=0.0000m
Pos 1 z=1.0940	q2=0	q3=-90	x=0.0700	y=0.0000m
Pos 2 z=0.4229	q2=0	q3=50	x=0.3143m	y=0.0000m
Pos 3 z=0.1009	q2=100	q3=-90	x=0.7654m	y=0.0000m
Pos 6 z=0.5983	q2=-90	q3=50	x=0.0011	y=0.0000m
Pos 7 z=0.5600m	q2=100	q3=-230	x=0.2183m	y=0.0000m
Pos 8 z=0.3540	q2=-90	q=-90	x=-0.6700m	y=0.0000m

Hence it can be concluded that the frame assignments and link lengths given to the model is valid. Please check out IRB-Kinematics-Final.mlx code for calculations. And run it along with the matlab-vrep.ttt to see the simulation.

5 Inverse Kinematics

In forward kinematics, we calculated the end effector location given the joint angles. In inverse kinematics we will do the exactly opposite process i.e calculate the joint angles given the end effector position. To calculate the inverse kinematics of the model, **kinematic decoupling** is used. **Process followed to calculate the inverse kinematics:**

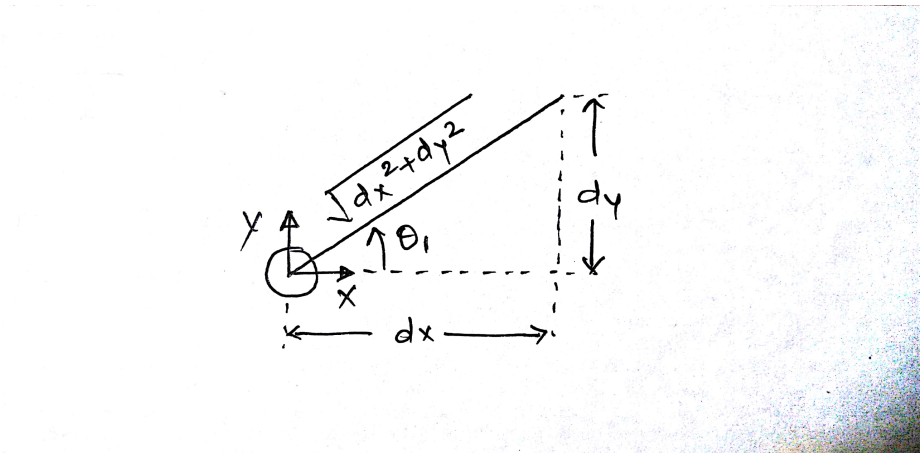

Figure 7: Top View. [2].

5.1 First a wrist center is located at the origin of the $frame_4$

.

2.Given the end effector location x_{des},y_{des},z_{des}, location of the wrist position $P_c = [dx, dy, dz]$ can be calculated as follows:

$$O_6^0 = P_c + l_6 z_5^0 \tag{7}$$

Where z_6^0 is the unit vector along the z6 axis expressed in the frame 0. Since wrist center always remains on the z6 axis, its position wrt. end effector is given by multiplying $l_5 z_6^0$ and subtracting from O_6^0

$$P_c = O_6^0 - + l_5 z_6^0 \tag{8}$$

Note than in our case $l_5 = 65mm$

5.2 Top view

Consider the top view of the robot given in the figure. From the top view we get the θ_1

$$\theta_1 = atan2(dy, dx) \tag{9}$$

5.3 Side view

Consider the side view of the robot. Using it, we can write the following equations to get the value of θ_2 and θ_3

$$\alpha = atan2(b, a) \tag{10}$$

$$c = \sqrt{a^2 + b^2} \tag{11}$$

$$L_4^2 = L_3^2 + c^2 - 2L_3 c \cos\beta \tag{12}$$

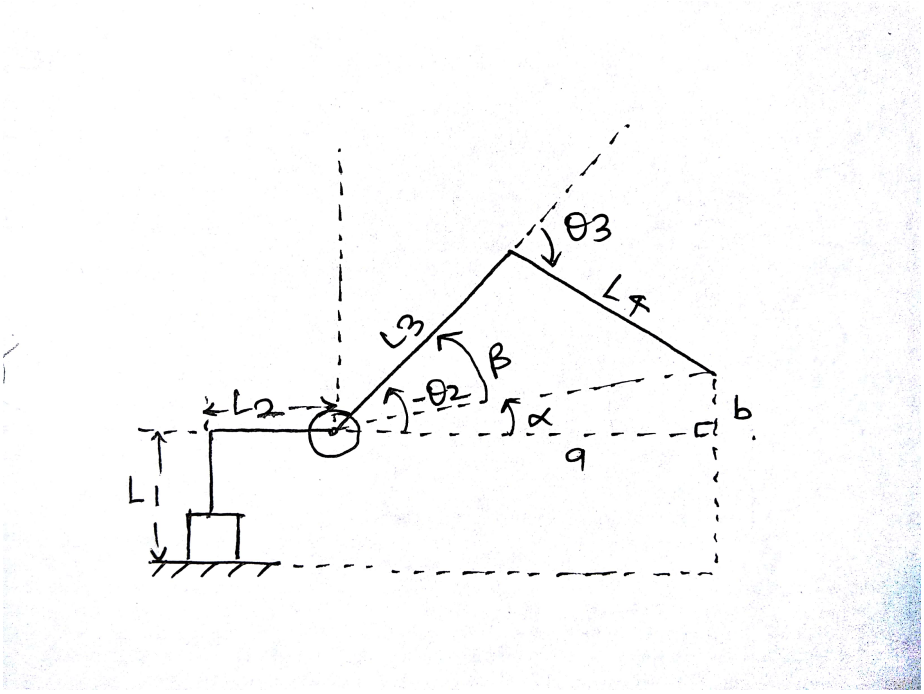

Figure 8: Side View. [2].

$$cos\beta = \frac{L_3^2 - L_4^2 + c^2}{2L_3c} = R \tag{13}$$

$$sin\beta = +\sqrt{1-R^2} or -\sqrt{1-R^2} \tag{14}$$

$$\beta = atan2(\sqrt{1-R^2}, R) \tag{15}$$

or

$$\beta = atan2(-\sqrt{1-R^2}, R) \tag{16}$$

similalrly,

$$c^2 = L_3^2 + L_4^2 - 2L_3L_4\cos\gamma \tag{17}$$

$$cos\gamma = \frac{L_3^2 + L_4^2 - c^2}{2L_3L_4} = S \tag{18}$$

Hence following the process used in equation (13),(14) and (15)

$$\gamma = atan2(\sqrt{1-S^2}, S) \tag{19}$$

or

$$\gamma = atan2(-\sqrt{1-S^2}, S) \tag{20}$$

$$\theta_2 = \beta + \alpha \tag{21}$$

and

$$\theta_2 = \pi - \gamma \tag{22}$$

5.4 Using θ_1, θ_2 and θ_3 we can calculate R_3^0

5.5 Note that we have R_0^6 and R_3^0, hence R_6^3 can be calculated as follows:

$$R_6^3 = R_3^{0T} R_0^6 \tag{23}$$

5.6 Finally calculate θ_4,θ_5, θ_6 using analytical method

That is compare the previously calculated R_6^3 with standard ZYZ Euler angle matrix. Let,

$$R_6^3 = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix} \tag{24}$$

Also our wrist resembles EULER ZYZ configuarion. Hence the euler angle matrix for the wrist can be written as,

$$R_{ZYZ} = \begin{bmatrix} c4c5c6 - s4s6 & -c4c5c6 - s4c6 & c4s5 \\ s4c5c6 + c4s6 & -s4c5s6 + c4c6 & s4s5 \\ -s5c6 & s5s6 & c5 \end{bmatrix} \tag{25}$$

Comparing (25) and (24), we get,

$$\theta_5 = atan2(\sqrt{a_7^2 + a_8^2}, a_9^2) \tag{26}$$

or

$$\theta_5 = atan2(-\sqrt{a_7^2 + a_8^2}, a_9^2) \tag{27}$$

$$\theta_6 = atan2(\frac{a8}{s5}, \frac{a7}{s5}) \tag{28}$$

or

$$\theta_4 = atan2(\frac{a6}{s5}, \frac{a3}{s5}) \tag{29}$$

6 Please see Jacobian Matrix derivation in Appendix

References

[1] https://www.robots.com/robots/abb-irb-140 *Kvantefænomener i Nanosystemer*. Niels Bohr Institute & Nano-Science Center, Københavns Universitet

[2] https://library.e.abb.com/public/8566e7a2794a4c2d8a63267d8b6e5697/3HAC041346*Kvantefænomener i Nanosystemer*. Niels Bohr Institute & Nano-Science Center, Københavns Universitet

Appendix

```
% Final project
disp(A01);
```

$$\begin{pmatrix} c_1 & 0 & -s_1 & L_2 c_1 \\ s_1 & 0 & c_1 & L_2 s_1 \\ 0 & -1 & 0 & L_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
disp(A12);
```

$$\begin{pmatrix} s_2 & c_2 & 0 & L_3 s_2 \\ -c_2 & s_2 & 0 & -L_3 c_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
disp(A23);
```

$$\begin{pmatrix} -c_3 & 0 & -s_3 & 0 \\ -s_3 & 0 & c_3 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
disp(A34);
```

$$\begin{pmatrix} c_4 & 0 & -s_4 & 0 \\ s_4 & 0 & c_4 & 0 \\ 0 & -1 & 0 & L_4 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
disp(A45);
```

$$\begin{pmatrix} c_5 & 0 & s_5 & 0 \\ s_5 & 0 & -c_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
disp(A56);
```

$$\begin{pmatrix} c_6 & -s_6 & 0 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 1 & L_5 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
T02 = A01 * A12;
T03 = T02 * A23;
T04 = T03 * A34;
T05 = T04 * A45;
T06 = T05 * A56;
```

```
disp(T02);
```

$$\begin{pmatrix} c_1 s_2 & c_1 c_2 & -s_1 & L_2 c_1 + L_3 c_1 s_2 \\ s_1 s_2 & c_2 s_1 & c_1 & L_2 s_1 + L_3 s_1 s_2 \\ c_2 & -s_2 & 0 & L_1 + L_3 c_2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
disp(T03);
```

$$\begin{pmatrix} -c_1 c_2 s_3 - c_1 c_3 s_2 & -s_1 & c_1 c_2 c_3 - c_1 s_2 s_3 & L_2 c_1 + L_3 c_1 s_2 \\ -c_2 s_1 s_3 - c_3 s_1 s_2 & c_1 & c_2 c_3 s_1 - s_1 s_2 s_3 & L_2 s_1 + L_3 s_1 s_2 \\ s_2 s_3 - c_2 c_3 & 0 & -c_2 s_3 - c_3 s_2 & L_1 + L_3 c_2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
disp(T04);
```

$$\begin{pmatrix} -s_1 s_4 - c_4 \sigma_1 & c_1 s_2 s_3 - c_1 c_2 c_3 & s_4 \sigma_1 - c_4 s_1 & L_2 c_1 + L_4 (c_1 c_2 c_3 - c_1 s_2 s_3) + L_3 c_1 s_2 \\ c_1 s_4 - c_4 \sigma_2 & s_1 s_2 s_3 - c_2 c_3 s_1 & c_1 c_4 + s_4 \sigma_2 & L_2 s_1 + L_4 (c_2 c_3 s_1 - s_1 s_2 s_3) + L_3 s_1 s_2 \\ -c_4 \sigma_3 & \sigma_4 & s_4 \sigma_3 & L_1 + L_3 c_2 - L_4 \sigma_4 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

where

$\sigma_1 = c_1 c_2 s_3 + c_1 c_3 s_2$

$\sigma_2 = c_2 s_1 s_3 + c_3 s_1 s_2$

$\sigma_3 = c_2 c_3 - s_2 s_3$

$\sigma_4 = c_2 s_3 + c_3 s_2$

```
disp(T05);
```

$$\begin{pmatrix} -s_5 \sigma_1 - c_5 \sigma_6 & s_4 \sigma_8 - c_4 s_1 & c_5 \sigma_1 - s_5 \sigma_6 & L_2 c_1 + L_4 \sigma_1 + L_3 c_1 s_2 \\ c_5 \sigma_5 - s_5 \sigma_2 & c_1 c_4 + s_4 \sigma_7 & c_5 \sigma_2 + s_5 \sigma_5 & L_2 s_1 + L_4 \sigma_2 + L_3 s_1 s_2 \\ s_5 \sigma_4 - c_4 c_5 \sigma_3 & s_4 \sigma_3 & -c_5 \sigma_4 - c_4 s_5 \sigma_3 & L_1 + L_3 c_2 - L_4 \sigma_4 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

where

$\sigma_1 = c_1 c_2 c_3 - c_1 s_2 s_3$

$\sigma_2 = c_2 c_3 s_1 - s_1 s_2 s_3$

$\sigma_3 = c_2 c_3 - s_2 s_3$

```
disp(T06);
```

$$\begin{pmatrix} -c_6 \sigma_4 - s_6 \sigma_7 & s_6 \sigma_4 - c_6 \sigma_7 & \sigma_3 & L_2 c_1 + L_5 \sigma_3 + L_4 \sigma_{10} + L_3 c_1 s_2 \\ s_6 \sigma_6 - c_6 \sigma_1 & s_6 \sigma_1 + c_6 \sigma_6 & \sigma_2 & L_2 s_1 + L_5 \sigma_2 + L_4 \sigma_8 + L_3 s_1 s_2 \\ c_6 \sigma_5 + s_4 s_6 \sigma_{12} & c_6 s_4 \sigma_{12} - s_6 \sigma_5 & -c_5 \sigma_{13} - c_4 s_5 \sigma_{12} & L_1 + L_3 c_2 - L_4 \sigma_{13} - L_5 (c_5 \sigma_{13} + c_4 s_5 \sigma_{12}) \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

where

$\sigma_1 = s_5 \sigma_8 - c_5 \sigma_9$

$\sigma_2 = c_5 \sigma_8 + s_5 \sigma_9$

$\sigma_3 = c_5 \sigma_{10} - s_5 \sigma_{11}$

```
%-------------------------Jacobian calculations --------------------------
Z00 = [0 0 1]';
Z01 = A01(1:3,3);
Z02 = T02(1:3,3);
```

```
Z03 = T03(1:3,3);
Z04 = T04(1:3,3);
Z05 = T05(1:3,3);
disp(Z00);
```

```
     0
     0
     1
```

```
disp(Z01);
```

$$\begin{pmatrix} -s_1 \\ c_1 \\ 0 \end{pmatrix}$$

```
disp(Z02);
```

$$\begin{pmatrix} -s_1 \\ c_1 \\ 0 \end{pmatrix}$$

```
disp(Z03);
```

$$\begin{pmatrix} c_1\,c_2\,c_3 - c_1\,s_2\,s_3 \\ c_2\,c_3\,s_1 - s_1\,s_2\,s_3 \\ -c_2\,s_3 - c_3\,s_2 \end{pmatrix}$$

```
disp(Z04);
```

$$\begin{pmatrix} s_4\,\left(c_1\,c_2\,s_3 + c_1\,c_3\,s_2\right) - c_4\,s_1 \\ c_1\,c_4 + s_4\,\left(c_2\,s_1\,s_3 + c_3\,s_1\,s_2\right) \\ s_4\,\left(c_2\,c_3 - s_2\,s_3\right) \end{pmatrix}$$

```
disp(Z05);
```

$$\begin{pmatrix} c_5\,\left(c_1\,c_2\,c_3 - c_1\,s_2\,s_3\right) - s_5\,\left(s_1\,s_4 + c_4\,\left(c_1\,c_2\,s_3 + c_1\,c_3\,s_2\right)\right) \\ c_5\,\left(c_2\,c_3\,s_1 - s_1\,s_2\,s_3\right) + s_5\,\left(c_1\,s_4 - c_4\,\left(c_2\,s_1\,s_3 + c_3\,s_1\,s_2\right)\right) \\ -c_5\,\left(c_2\,s_3 + c_3\,s_2\right) - c_4\,s_5\,\left(c_2\,c_3 - s_2\,s_3\right) \end{pmatrix}$$

```
Jv1 = cross(Z00,T06(1:3,4)-[0 0 1]');
Jv2 = cross(Z01,T06(1:3,4)-A01(1:3,4));
Jv3 = cross(Z02,T06(1:3,4)-T02(1:3,4));
Jv4 = cross(Z03,T06(1:3,4)-T03(1:3,4));
Jv5 = cross(Z04,T06(1:3,4)-T04(1:3,4));
Jv6 = cross(Z05,T06(1:3,4)-T05(1:3,4));

disp(Jv1);
```

$$\begin{pmatrix} -L_2\,s_1 - L_5\,\left(c_5\,\sigma_2 + s_5\,\left(c_1\,s_4 - c_4\,\left(c_2\,s_1\,s_3 + c_3\,s_1\,s_2\right)\right)\right) - L_4\,\sigma_2 - L_3\,s_1\,s_2 \\ L_2\,c_1 + L_5\,\left(c_5\,\sigma_1 - s_5\,\left(s_1\,s_4 + c_4\,\left(c_1\,c_2\,s_3 + c_1\,c_3\,s_2\right)\right)\right) + L_4\,\sigma_1 + L_3\,c_1\,s_2 \\ 0 \end{pmatrix}$$

where

$$\sigma_1 = c_1\,c_2\,c_3 - c_1\,s_2\,s_3$$

$$\sigma_2 = c_2\,c_3\,s_1 - s_1\,s_2\,s_3$$

```
disp(Jv2);
```

$$\begin{pmatrix} -c_1\,\sigma_3 \\ -s_1\,\sigma_3 \\ -c_1\,\left(L_5\,\left(c_5\,\sigma_1 - s_5\,\left(s_1\,s_4 + c_4\,\left(c_1\,c_2\,s_3 + c_1\,c_3\,s_2\right)\right)\right) + L_4\,\sigma_1 + L_3\,c_1\,s_2\right) - s_1\,\left(L_5\,\left(c_5\,\sigma_2 + s_5\,\left(c_1\,s_4 - c_4\,\left(c_2\,s_1\,s_3 + c_3\,s_1\,s_2\right)\right)\right) + L_4\,\sigma_2 + L_3\,s \end{pmatrix}$$

where

$$\sigma_1 = c_1\,c_2\,c_3 - c_1\,s_2\,s_3$$

$$\sigma_2 = c_2\,c_3\,s_1 - s_1\,s_2\,s_3$$

$$\sigma_3 = L_4\,\sigma_4 - L_3\,c_2 + L_5\,\left(c_5\,\sigma_4 + c_4\,s_5\,\left(c_2\,c_3 - s_2\,s_3\right)\right)$$

$$\sigma_4 = c_2\,s_3 + c_3\,s_2$$

```
disp(Jv3);
```

$$\begin{pmatrix} -c_1\,\sigma_1 \\ -s_1\,\sigma_1 \\ -c_1\,\left(L_5\,\left(c_5\,\sigma_2 - s_5\,\left(s_1\,s_4 + c_4\,\left(c_1\,c_2\,s_3 + c_1\,c_3\,s_2\right)\right)\right) + L_4\,\sigma_2\right) - s_1\,\left(L_5\,\left(c_5\,\sigma_3 + s_5\,\left(c_1\,s_4 - c_4\,\left(c_2\,s_1\,s_3 + c_3\,s_1\,s_2\right)\right)\right) + L_4\,\sigma_3\right) \end{pmatrix}$$

where

$$\sigma_1 = L_4\,\left(c_2\,s_3 + c_3\,s_2\right) + L_5\,\left(c_5\,\left(c_2\,s_3 + c_3\,s_2\right) + c_4\,s_5\,\left(c_2\,c_3 - s_2\,s_3\right)\right)$$

$$\sigma_2 = c_1\,c_2\,c_3 - c_1\,s_2\,s_3$$

$$\sigma_3 = c_2\,c_3\,s_1 - s_1\,s_2\,s_3$$

```
disp(Jv4);
```

$$\begin{pmatrix} \sigma_1\,\left(c_2\,s_3 + c_3\,s_2\right) - \sigma_3\,\left(c_2\,c_3\,s_1 - s_1\,s_2\,s_3\right) \\ \sigma_3\,\left(c_1\,c_2\,c_3 - c_1\,s_2\,s_3\right) - \sigma_2\,\left(c_2\,s_3 + c_3\,s_2\right) \\ \sigma_1\,\left(c_1\,c_2\,c_3 - c_1\,s_2\,s_3\right) - \left(c_2\,c_3\,s_1 - s_1\,s_2\,s_3\right)\,\sigma_2 \end{pmatrix}$$

where

$$\sigma_1 = L_5\,\left(c_5\,\left(c_2\,c_3\,s_1 - s_1\,s_2\,s_3\right) + s_5\,\left(c_1\,s_4 - c_4\,\left(c_2\,s_1\,s_3 + c_3\,s_1\,s_2\right)\right)\right) + L_4\,\left(c_2\,c_3\,s_1 - s_1\,s_2\,s_3\right)$$

$$\sigma_2 = L_5\,\left(c_5\,\left(c_1\,c_2\,c_3 - c_1\,s_2\,s_3\right) - s_5\,\left(s_1\,s_4 + c_4\,\left(c_1\,c_2\,s_3 + c_1\,c_3\,s_2\right)\right)\right) + L_4\,\left(c_1\,c_2\,c_3 - c_1\,s_2\,s_3\right)$$

$$\sigma_3 = L_4\,\left(c_2\,s_3 + c_3\,s_2\right) + L_5\,\left(c_5\,\left(c_2\,s_3 + c_3\,s_2\right) + c_4\,s_5\,\left(c_2\,c_3 - s_2\,s_3\right)\right)$$

```
disp(Jv5);
```

$$\begin{pmatrix} -L_5\,\sigma_3\,\sigma_5 - L_5\,s_4\,\sigma_1\,\sigma_6 \\ L_5\,s_4\,\sigma_2\,\sigma_6 - L_5\,\sigma_5\,\sigma_4 \\ -L_5\,\sigma_3\,\sigma_2 - L_5\,\sigma_1\,\sigma_4 \end{pmatrix}$$

where

$$\sigma_1 = c_5\,\left(c_2\,c_3\,s_1 - s_1\,s_2\,s_3\right) + s_5\,\left(c_1\,s_4 - c_4\,\left(c_2\,s_1\,s_3 + c_3\,s_1\,s_2\right)\right)$$

$$\sigma_2 = c_5\,\left(c_1\,c_2\,c_3 - c_1\,s_2\,s_3\right) - s_5\,\left(s_1\,s_4 + c_4\,\left(c_1\,c_2\,s_3 + c_1\,c_3\,s_2\right)\right)$$

$$\sigma_3 = c_1\,c_4 + s_4\,\left(c_2\,s_1\,s_3 + c_3\,s_1\,s_2\right)$$

$$\sigma_4 = c_4\,s_1 - s_4\,\left(c_1\,c_2\,s_3 + c_1\,c_3\,s_2\right)$$

$$\sigma_5 = c_5\,\left(c_2\,s_3 + c_3\,s_2\right) + c_4\,s_5\,\sigma_6$$

$$\sigma_6 = c_2\,c_3 - s_2\,s_3$$

```
disp(Jv6);
```

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

```
%%%%% Kinematics Test for Pose0 %%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Click Run section to run this section %%%%%%%%%%%%%%%
vrep=remApi('remoteApi');
```

```
Note: always make sure you use the corresponding remoteApi library
(i.e. 32bit Matlab will not work with 64bit remoteApi, and vice-versa)
```

```
vrep.simxFinish(-1);
clientID=vrep.simxStart('127.0.0.1',19999,true,true,5000,5);
if (clientID>-1)
    disp('Connected to remote API server');
    %-------------- Handles -------------%
    [returnCode,joint1]=vrep.simxGetObjectHandle(clientID,'joint1',vrep.simx_opmode_blocking);
    [returnCode,joint2]=vrep.simxGetObjectHandle(clientID,'joint2',vrep.simx_opmode_blocking);
    [returnCode,joint3]=vrep.simxGetObjectHandle(clientID,'joint3',vrep.simx_opmode_blocking);
    [returnCode,joint4]=vrep.simxGetObjectHandle(clientID,'joint4',vrep.simx_opmode_blocking);
    [returnCode,joint5]=vrep.simxGetObjectHandle(clientID,'joint5',vrep.simx_opmode_blocking);
    [returnCode,joint6]=vrep.simxGetObjectHandle(clientID,'joint6',vrep.simx_opmode_blocking);

    % Current positions
    [returnCode,j1_init]=vrep.simxGetJointPosition(clientID,joint1,vrep.simx_opmode_streaming);
    [returnCode,j2_init]=vrep.simxGetJointPosition(clientID,joint2,vrep.simx_opmode_streaming);
    [returnCode,j3_init]=vrep.simxGetJointPosition(clientID,joint3,vrep.simx_opmode_streaming);
    [returnCode,j4_init]=vrep.simxGetJointPosition(clientID,joint4,vrep.simx_opmode_streaming);
    [returnCode,j5_init]=vrep.simxGetJointPosition(clientID,joint5,vrep.simx_opmode_streaming);
    [returnCode,j6_init]=vrep.simxGetJointPosition(clientID,joint6,vrep.simx_opmode_streaming);

    [returnCode]=vrep.simxSetJointPosition(clientID,joint2,degtorad(0),vrep.simx_opmode_oneshot);
    [returnCode]=vrep.simxSetJointPosition(clientID,joint3,degtorad(0),vrep.simx_opmode_oneshot);
    [returnCode,pos0]=vrep.simxGetObjectPosition(clientID,joint5,-1,vrep.simx_opmode_blocking);
    disp(pos0);
    vrep.simxFinish(-1);
end
```

```
Connected to remote API server
    0.4500    0.0000    0.7140
```

```
vrep.delete();
```

```
%%%%% Kinematics Test for Pose1 %%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Click Run section to run this section %%%%%%%%%%%%%%%
vrep=remApi('remoteApi');
```

```
Note: always make sure you use the corresponding remoteApi library
(i.e. 32bit Matlab will not work with 64bit remoteApi, and vice-versa)
```

```
vrep.simxFinish(-1);
clientID=vrep.simxStart('127.0.0.1',19999,true,true,5000,5);
if (clientID>-1)
    disp('Connected to remote API server');
    %-------------- Handles -------------%
    [returnCode,joint1]=vrep.simxGetObjectHandle(clientID,'joint1',vrep.simx_opmode_blocking);
    [returnCode,joint2]=vrep.simxGetObjectHandle(clientID,'joint2',vrep.simx_opmode_blocking);
    [returnCode,joint3]=vrep.simxGetObjectHandle(clientID,'joint3',vrep.simx_opmode_blocking);
    [returnCode,joint4]=vrep.simxGetObjectHandle(clientID,'joint4',vrep.simx_opmode_blocking);
    [returnCode,joint5]=vrep.simxGetObjectHandle(clientID,'joint5',vrep.simx_opmode_blocking);
    [returnCode,joint6]=vrep.simxGetObjectHandle(clientID,'joint6',vrep.simx_opmode_blocking);

    % Current positions
    [returnCode,j1_init]=vrep.simxGetJointPosition(clientID,joint1,vrep.simx_opmode_streaming);
    [returnCode,j2_init]=vrep.simxGetJointPosition(clientID,joint2,vrep.simx_opmode_streaming);
    [returnCode,j3_init]=vrep.simxGetJointPosition(clientID,joint3,vrep.simx_opmode_streaming);
    [returnCode,j4_init]=vrep.simxGetJointPosition(clientID,joint4,vrep.simx_opmode_streaming);
    [returnCode,j5_init]=vrep.simxGetJointPosition(clientID,joint5,vrep.simx_opmode_streaming);
    [returnCode,j6_init]=vrep.simxGetJointPosition(clientID,joint6,vrep.simx_opmode_streaming);

    [returnCode]=vrep.simxSetJointPosition(clientID,joint2,degtorad(0),vrep.simx_opmode_oneshot);
    [returnCode]=vrep.simxSetJointPosition(clientID,joint3,degtorad(-90),vrep.simx_opmode_oneshot);
    [returnCode,pos1]=vrep.simxGetObjectPosition(clientID,joint5,-1,vrep.simx_opmode_blocking);
```

```
    disp(pos1);
    vrep.simxFinish(-1);
end
```

```
Connected to remote API server
    0.0700    0.0000    1.0940
```

```
vrep.delete();
```

```
%%%%% Kinematics Test for Pose 2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Click Run section to run this section %%%%%%%%%%%%%%%
vrep=remApi('remoteApi');
```

```
Note: always make sure you use the corresponding remoteApi library
(i.e. 32bit Matlab will not work with 64bit remoteApi, and vice-versa)
```

```
vrep.simxFinish(-1);
clientID=vrep.simxStart('127.0.0.1',19999,true,true,5000,5);
if (clientID>-1)
    disp('Connected to remote API server');
    %-------------- Handles -------------%
    [returnCode,joint1]=vrep.simxGetObjectHandle(clientID,'joint1',vrep.simx_opmode_blocking);
    [returnCode,joint2]=vrep.simxGetObjectHandle(clientID,'joint2',vrep.simx_opmode_blocking);
    [returnCode,joint3]=vrep.simxGetObjectHandle(clientID,'joint3',vrep.simx_opmode_blocking);
    [returnCode,joint4]=vrep.simxGetObjectHandle(clientID,'joint4',vrep.simx_opmode_blocking);
    [returnCode,joint5]=vrep.simxGetObjectHandle(clientID,'joint5',vrep.simx_opmode_blocking);
    [returnCode,joint6]=vrep.simxGetObjectHandle(clientID,'joint6',vrep.simx_opmode_blocking);

    % Current positions
    [returnCode,j1_init]=vrep.simxGetJointPosition(clientID,joint1,vrep.simx_opmode_streaming);
    [returnCode,j2_init]=vrep.simxGetJointPosition(clientID,joint2,vrep.simx_opmode_streaming);
    [returnCode,j3_init]=vrep.simxGetJointPosition(clientID,joint3,vrep.simx_opmode_streaming);
    [returnCode,j4_init]=vrep.simxGetJointPosition(clientID,joint4,vrep.simx_opmode_streaming);
    [returnCode,j5_init]=vrep.simxGetJointPosition(clientID,joint5,vrep.simx_opmode_streaming);
    [returnCode,j6_init]=vrep.simxGetJointPosition(clientID,joint6,vrep.simx_opmode_streaming);

    [returnCode]=vrep.simxSetJointPosition(clientID,joint2,degtorad(0),vrep.simx_opmode_oneshot);
    [returnCode]=vrep.simxSetJointPosition(clientID,joint3,degtorad(50),vrep.simx_opmode_oneshot);
    [returnCode,pos2]=vrep.simxGetObjectPosition(clientID,joint5,-1,vrep.simx_opmode_blocking);
    disp(pos2);
    vrep.simxFinish(-1);
end
```

```
Connected to remote API server
    0.3143   -0.0000    0.4229
```

```
%%%%% Kinematics Test for Pose 3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Click Run section to run this section %%%%%%%%%%%%%%%
vrep=remApi('remoteApi');
```

```
Note: always make sure you use the corresponding remoteApi library
(i.e. 32bit Matlab will not work with 64bit remoteApi, and vice-versa)
```

```
vrep.simxFinish(-1);
clientID=vrep.simxStart('127.0.0.1',19999,true,true,5000,5);
if (clientID>-1)
    disp('Connected to remote API server');
    %-------------- Handles -------------%
    [returnCode,joint1]=vrep.simxGetObjectHandle(clientID,'joint1',vrep.simx_opmode_blocking);
    [returnCode,joint2]=vrep.simxGetObjectHandle(clientID,'joint2',vrep.simx_opmode_blocking);
    [returnCode,joint3]=vrep.simxGetObjectHandle(clientID,'joint3',vrep.simx_opmode_blocking);
    [returnCode,joint4]=vrep.simxGetObjectHandle(clientID,'joint4',vrep.simx_opmode_blocking);
    [returnCode,joint5]=vrep.simxGetObjectHandle(clientID,'joint5',vrep.simx_opmode_blocking);
    [returnCode,joint6]=vrep.simxGetObjectHandle(clientID,'joint6',vrep.simx_opmode_blocking);

    % Current positions
    [returnCode,j1_init]=vrep.simxGetJointPosition(clientID,joint1,vrep.simx_opmode_streaming);
    [returnCode,j2_init]=vrep.simxGetJointPosition(clientID,joint2,vrep.simx_opmode_streaming);
    [returnCode,j3_init]=vrep.simxGetJointPosition(clientID,joint3,vrep.simx_opmode_streaming);
    [returnCode,j4_init]=vrep.simxGetJointPosition(clientID,joint4,vrep.simx_opmode_streaming);
    [returnCode,j5_init]=vrep.simxGetJointPosition(clientID,joint5,vrep.simx_opmode_streaming);
    [returnCode,j6_init]=vrep.simxGetJointPosition(clientID,joint6,vrep.simx_opmode_streaming);
```

```
    [returnCode]=vrep.simxSetJointPosition(clientID,joint2,degtorad(110),vrep.simx_opmode_oneshot);
    [returnCode]=vrep.simxSetJointPosition(clientID,joint3,degtorad(-90),vrep.simx_opmode_oneshot);
    [returnCode,pos3]=vrep.simxGetObjectPosition(clientID,joint5,-1,vrep.simx_opmode_blocking);
    disp(pos3);
    vrep.simxFinish(-1);
end
```

```
Connected to remote API server
    0.7654   -0.0000    0.1009
```

```
%%%%% Kinematics Test for Pose 6 %%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Click Run section to run this section %%%%%%%%%%%%%%%
vrep=remApi('remoteApi');
```

```
Note: always make sure you use the corresponding remoteApi library
(i.e. 32bit Matlab will not work with 64bit remoteApi, and vice-versa)
```

```
vrep.simxFinish(-1);
clientID=vrep.simxStart('127.0.0.1',19999,true,true,5000,5);
if (clientID>-1)
    disp('Connected to remote API server');
    %-------------- Handles -------------%
    [returnCode,joint1]=vrep.simxGetObjectHandle(clientID,'joint1',vrep.simx_opmode_blocking);
    [returnCode,joint2]=vrep.simxGetObjectHandle(clientID,'joint2',vrep.simx_opmode_blocking);
    [returnCode,joint3]=vrep.simxGetObjectHandle(clientID,'joint3',vrep.simx_opmode_blocking);
    [returnCode,joint4]=vrep.simxGetObjectHandle(clientID,'joint4',vrep.simx_opmode_blocking);
    [returnCode,joint5]=vrep.simxGetObjectHandle(clientID,'joint5',vrep.simx_opmode_blocking);
    [returnCode,joint6]=vrep.simxGetObjectHandle(clientID,'joint6',vrep.simx_opmode_blocking);

    % Current positions
    [returnCode,j1_init]=vrep.simxGetJointPosition(clientID,joint1,vrep.simx_opmode_streaming);
    [returnCode,j2_init]=vrep.simxGetJointPosition(clientID,joint2,vrep.simx_opmode_streaming);
    [returnCode,j3_init]=vrep.simxGetJointPosition(clientID,joint3,vrep.simx_opmode_streaming);
    [returnCode,j4_init]=vrep.simxGetJointPosition(clientID,joint4,vrep.simx_opmode_streaming);
    [returnCode,j5_init]=vrep.simxGetJointPosition(clientID,joint5,vrep.simx_opmode_streaming);
    [returnCode,j6_init]=vrep.simxGetJointPosition(clientID,joint6,vrep.simx_opmode_streaming);

    [returnCode]=vrep.simxSetJointPosition(clientID,joint2,degtorad(-90),vrep.simx_opmode_oneshot);
    [returnCode]=vrep.simxSetJointPosition(clientID,joint3,degtorad(50),vrep.simx_opmode_oneshot);
    [returnCode,pos6]=vrep.simxGetObjectPosition(clientID,joint5,-1,vrep.simx_opmode_blocking);
    disp(pos6);
    vrep.simxFinish(-1);
end
```

```
Connected to remote API server
    0.0011   -0.0000    0.5983
```

```
%%%%% Kinematics Test for Pose 7 %%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Click Run section to run this section %%%%%%%%%%%%%%%
vrep=remApi('remoteApi');
```

```
Note: always make sure you use the corresponding remoteApi library
(i.e. 32bit Matlab will not work with 64bit remoteApi, and vice-versa)
```

```
vrep.simxFinish(-1);
clientID=vrep.simxStart('127.0.0.1',19999,true,true,5000,5);
if (clientID>-1)
    disp('Connected to remote API server');
    %-------------- Handles -------------%
    [returnCode,joint1]=vrep.simxGetObjectHandle(clientID,'joint1',vrep.simx_opmode_blocking);
    [returnCode,joint2]=vrep.simxGetObjectHandle(clientID,'joint2',vrep.simx_opmode_blocking);
    [returnCode,joint3]=vrep.simxGetObjectHandle(clientID,'joint3',vrep.simx_opmode_blocking);
    [returnCode,joint4]=vrep.simxGetObjectHandle(clientID,'joint4',vrep.simx_opmode_blocking);
    [returnCode,joint5]=vrep.simxGetObjectHandle(clientID,'joint5',vrep.simx_opmode_blocking);
    [returnCode,joint6]=vrep.simxGetObjectHandle(clientID,'joint6',vrep.simx_opmode_blocking);

    % Current positions
    [returnCode,j1_init]=vrep.simxGetJointPosition(clientID,joint1,vrep.simx_opmode_streaming);
    [returnCode,j2_init]=vrep.simxGetJointPosition(clientID,joint2,vrep.simx_opmode_streaming);
    [returnCode,j3_init]=vrep.simxGetJointPosition(clientID,joint3,vrep.simx_opmode_streaming);
    [returnCode,j4_init]=vrep.simxGetJointPosition(clientID,joint4,vrep.simx_opmode_streaming);
    [returnCode,j5_init]=vrep.simxGetJointPosition(clientID,joint5,vrep.simx_opmode_streaming);
```

```
    [returnCode,j6_init]=vrep.simxGetJointPosition(clientID,joint6,vrep.simx_opmode_streaming);

    [returnCode]=vrep.simxSetJointPosition(clientID,joint2,degtorad(110),vrep.simx_opmode_oneshot);
    [returnCode]=vrep.simxSetJointPosition(clientID,joint3,degtorad(-230),vrep.simx_opmode_oneshot);
    [returnCode,pos7]=vrep.simxGetObjectPosition(clientID,joint5,-1,vrep.simx_opmode_blocking);
    disp(pos7);
    vrep.simxFinish(-1);
end
```

```
Connected to remote API server
    0.2183   -0.0000    0.5600
```

```
%%%%% Kinematics Test for Pose 8 %%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Click Run section to run this section %%%%%%%%%%%%%%%
vrep=remApi('remoteApi');
```

```
Note: always make sure you use the corresponding remoteApi library
(i.e. 32bit Matlab will not work with 64bit remoteApi, and vice-versa)
```

```
vrep.simxFinish(-1);
clientID=vrep.simxStart('127.0.0.1',19999,true,true,5000,5);
if (clientID>-1)
    disp('Connected to remote API server');
    %-------------- Handles -------------%
    [returnCode,joint1]=vrep.simxGetObjectHandle(clientID,'joint1',vrep.simx_opmode_blocking);
    [returnCode,joint2]=vrep.simxGetObjectHandle(clientID,'joint2',vrep.simx_opmode_blocking);
    [returnCode,joint3]=vrep.simxGetObjectHandle(clientID,'joint3',vrep.simx_opmode_blocking);
    [returnCode,joint4]=vrep.simxGetObjectHandle(clientID,'joint4',vrep.simx_opmode_blocking);
    [returnCode,joint5]=vrep.simxGetObjectHandle(clientID,'joint5',vrep.simx_opmode_blocking);
    [returnCode,joint6]=vrep.simxGetObjectHandle(clientID,'joint6',vrep.simx_opmode_blocking);

    % Current positions
    [returnCode,j1_init]=vrep.simxGetJointPosition(clientID,joint1,vrep.simx_opmode_streaming);
    [returnCode,j2_init]=vrep.simxGetJointPosition(clientID,joint2,vrep.simx_opmode_streaming);
    [returnCode,j3_init]=vrep.simxGetJointPosition(clientID,joint3,vrep.simx_opmode_streaming);
    [returnCode,j4_init]=vrep.simxGetJointPosition(clientID,joint4,vrep.simx_opmode_streaming);
    [returnCode,j5_init]=vrep.simxGetJointPosition(clientID,joint5,vrep.simx_opmode_streaming);
    [returnCode,j6_init]=vrep.simxGetJointPosition(clientID,joint6,vrep.simx_opmode_streaming);

    [returnCode]=vrep.simxSetJointPosition(clientID,joint2,degtorad(-90),vrep.simx_opmode_oneshot);
    [returnCode]=vrep.simxSetJointPosition(clientID,joint3,degtorad(-90),vrep.simx_opmode_oneshot);
    [returnCode,pos8]=vrep.simxGetObjectPosition(clientID,joint5,-1,vrep.simx_opmode_blocking);
    disp(pos8);
    vrep.simxFinish(-1);
end
```

```
Connected to remote API server
   -0.6700   -0.0000    0.3540
```