MODELLING A BAXTER ROBOTIC ARM TO DROP A BALL INTO A BASKET

ROBOT MODELLING

ENPM - 662 Project

- ***Pradeep Gopal***

116885027

December 2019

Table of Contents

Abstract

The idea is to model a Baxter robotic arm to pick up a ball from a fixed point (maybe from a table or from a conveyor belt) with the help of a two finger robotic gripper and drop it successfully inside a box placed at different positions each time by formulating the forward and inverse kinematics of the arm along with trajectory planning for the ball from the robotic arm to the box.

Motivation

Robots are widely used in applications including manufacturing, assembly, packing, transport, space etc. Robots are not considered ideal for most of the applications mainly because of the costs incurred in its installation and maintenance. One such application were robots have recently made an impact is the sports industry. Robots such as 'Cue' have already found its place in the sports industry. It was developed by Toyota as a fun project to simulate the throwing of a ball into the basket [1]. We can employ robots that mimics the action of players and hence use these results to simulate the moves required by the player. This way we can calculate the moves precisely and thus the efficiency of the player can be improvised to a great extent. This mechanism not only can be limited to sports, but also can be extended to a variety of applications including Manufacturing industries. Being a human friendly robot, this robot can sense potential collisions early and can reduce the force before the impact and it is considered as one of the best robots to be used in crowded environments.

Approach to performing the task

The robot arm picks the ball from the table using its 2-finger spherical wrist and drops it successfully inside a box placed at different positions each time by formulating the forward and inverse kinematics of the arm along with trajectory planning for the ball from the robotic arm to the box. The 6 Degrees of Freedom (DOF) robotic arm facilitates the dropping of the ball into the basket even if the basket is placed at a numerous position inside the workspace of this robot. In order to achieve this objective, the robot's kinematics and its full range of abilities is studied.

This report follows every work sequentially and helps the reader understand the challenges faced in each step. Forward kinematic modeling of the robot's left arm is done. A custom GUI is created in MATLAB for both the 7 DOF and 6 DOF case to visualize the robot motions in 3D for different input joint angles. The inverse kinematics equations of the robot are derived in detail for the 6 DOF left arm and validated. Finally, the trajectory is planned, and the joint angles are calculated for different positions of the end effector to successfully pick and drop the ball into the basket through inverse kinematics equations. These joint angles are then fed into the Vrep scene containing the Baxter robot to visualize the whole process. Analysis and future work are also discussed towards the end of this report.

Robot description

The Baxter Left arm with 7 DOF is chosen for this operation. The 2 DOF in the shoulder, 2 DOF in the elbow and the 3 DOF in the wrist makes it a suitable choice for performing this task. All the joints are revolute joints and hence makes this operation smooth. The different joint configurations of the Baxter arm can be noted from Table.1

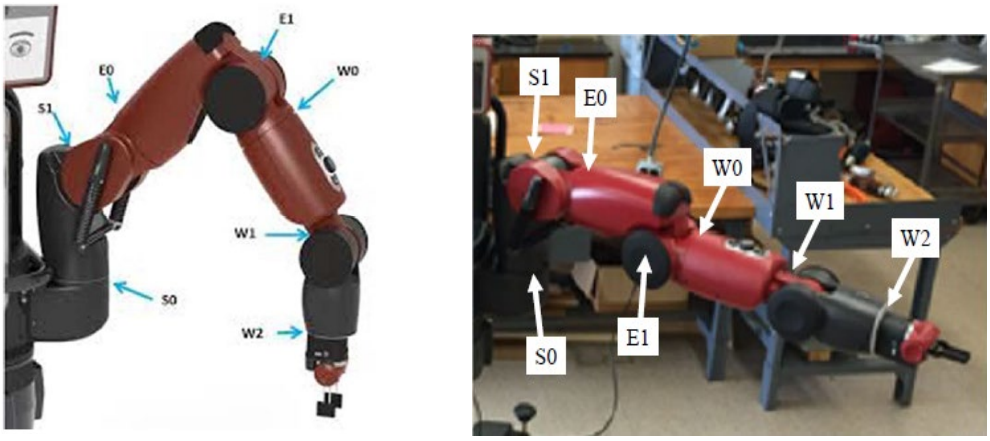


Fig 1 : Baxter Robot Arm

Joint Name	Joint Motion
S0	Shoulder roll
S1	Shoulder pitch
E0	Elbow roll
E1	Elbow pitch
W1	Wrist roll
W2	Wrist pitch
W3	Wrist roll

Table 1: Seven DOF left arm naming conventions

The link lengths of the Baxter left arm is shown in Table 2. The minimum and maximum joint angle constraints of the original Baxter arm is given in Table 3 [2]. But for simulations purposes, these constraints are not considered and a range of -180 to 180 degrees is allowed for each revolute joint.

Length	Value (mm)
L_0	270.35
L_1	69.00
L_2	364.35
L_3	69.00
L_4	374.29
L_5	10.00
L_6	368.30

Table 2: Link lengths of the Baxter arm

Joint Name	Joint Variable	θ_i min	θ_i max	θ_i range
S_0	θ_1	+51°	−141°	192°
S_1	θ_2	+60°	−123°	183°
E_0	θ_3	+173°	−173°	346°
E_1	θ_4	+150°	−3°	153°
W_0	θ_5	+175°	−175°	350°
W_1	θ_6	+120°	−90°	210°
W_2	θ_7	+175°	−175°	350°

Table 3: Seven DOF Left arm Joint limits

Robot appropriateness and modifications made

The Baxter Robot System is a human-sized humanoid robot with dual 7-degree-of-freedom (DOF) arms with stationary pedestal, torso, and 2-dof head, a vision system, a robot control system, a safety system, and an optional gravity-offload controller and collision detection routine. One main advantage in choosing this arm is that, all the joints are revolute, and it has a spherical wrist which can be used to drop the ball into the basket more efficiently.

The Baxter arm design is rather traditional, like many previous 7-dof robot arms. 7-dof robot arms are classified as kinematically-redundant, i.e. possessing more joint freedoms than necessary to operate fully in the desired Cartesian space. Specifically, Baxter has $n = 7$ single-DOF revolute (R) joints, which is one greater than the $m = 6$ Cartesian DOF (3 translations and 3 rotations) for general trajectories. When $n > m$, the robot qualifies as kinematically-redundant, which means that in addition to reaching general desired 6-dof Cartesian trajectories, a 7-dof robot arm can also be used for optimizing robot arm performance at the same time.

For avoiding the infinite solution problem and kinematically redundant case, we will be setting one degree of freedom (Elbow roll E0) to zero all the time. Now the resultant Baxter arm is a 6 DOF robot arm.

According to Pieper's solution, for an analytic Inverse Kinematic solution to exist, the robot must have a minimum of three consecutive frames with the same origin. But in the case of the Baxter arm, the presence of the link offset $L_5 \neq 0$ does not allow the

system to have more than two consecutive frames with same origin. Therefore, in order to calculate the inverse kinematic solutions, the link length L_5 is set to 0. The link length being the smallest link of all the other links considered, the final error in computing the joint angles will be relatively small. Therefore, to calculate the analytic solution to the Baxter arm, the joint θ_3 is locked to 0 and the link length L_5 is assumed to be zero.

Gripping Mechanism

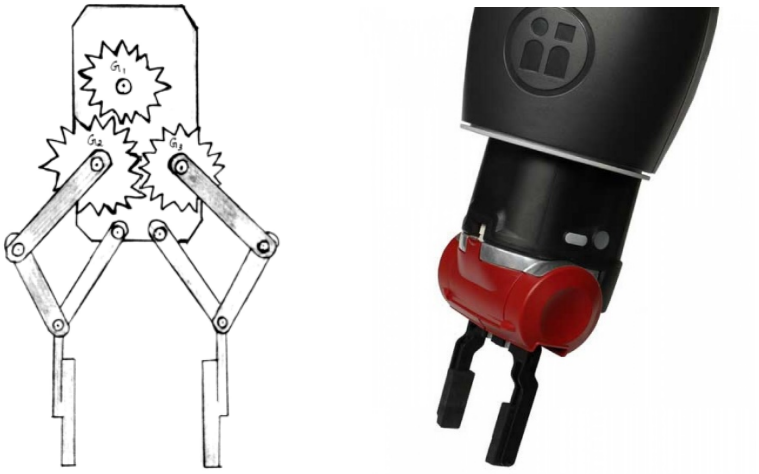


Figure 2 - 2 finger robotic gripper

Fig. 2 represents the 2-finger robotic gripper for the end effector of the Baxter robot to pick up the ball from the table. There are three gears namely G1, G2 and G3. The G1 gear is connected to the motor and it drives the other two gears G2 and G3 which in turns controls the opening and closing of the gripper.

The base plate of the robot can be changed respective to the type of object we want to pick up with the help of the gripper. In our case, the gripper base plate can be rectangular and it is efficient to hold the ball inside the fingers without possible slippage.

Due to time constraints, the Baxter's electric parallel gripper which is similar to the above designed gripper is used in this project for Simulation purposes. Electric parallel grippers provide one degree of freedom with multiple grasp widths, deliver an actuation time of less than one second and allow Baxter to pick up from the outside or inside of an object. The clips which come along with the parallel gripper provide various grasp widths in order to adapt to the types of objects that Baxter must handle. Baxter is able to pick up from the outside or inside of an object. The maximum size of the clamp opening is 151 mm for the largest pliers [3]. The clips and the fingers of the robot can be changed with ease.

Assumptions

- The third joint of the Baxter left arm is locked at zero degree to overcome the kinematically redundant scenario, i.e. $\theta_3 = 0$
- The link length is set to zero, while calculating the analytical inverse kinematic solutions, i.e. $L_5 = 0$
- Only one solution of the inverse kinematic problem is considered
- There is no slippage of the ball from the robotic gripper.
- Maximum and minimum joint limits of the robot is not considered
- The basket where the ball must be dropped is always kept inside the workspace of the robot.
- Since the pose of the ball and the basket is already known, the force of gravity is taken into consideration while dropping the ball into the basket. i.e., The end effector moves slowly and orients the ball at the top of the basket before dropping
- The Right-hand coordination system is used to draw the frame diagram for the robot.
- The rotation of the joints in clockwise direction is assumed to be the positive direction about the Z – axis and the rotation of the joints in anti – clockwise direction is assumed to the negative direction rotation about the Z – axis.

Forward Kinematics

The DH table is provided for the 7 DOF and 6DOF versions of the Left Baxter arm. From the table, the A matrices can be computed as the product of 4 basic transformations.

$$A_i = \mathbf{Rot}_{z,\theta_i}\mathbf{Trans}_{z,d_i}\mathbf{Trans}_{x,a_i}\mathbf{Rot}_{x,\alpha_i}$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

a) 7 DOF Baxter Left Arm

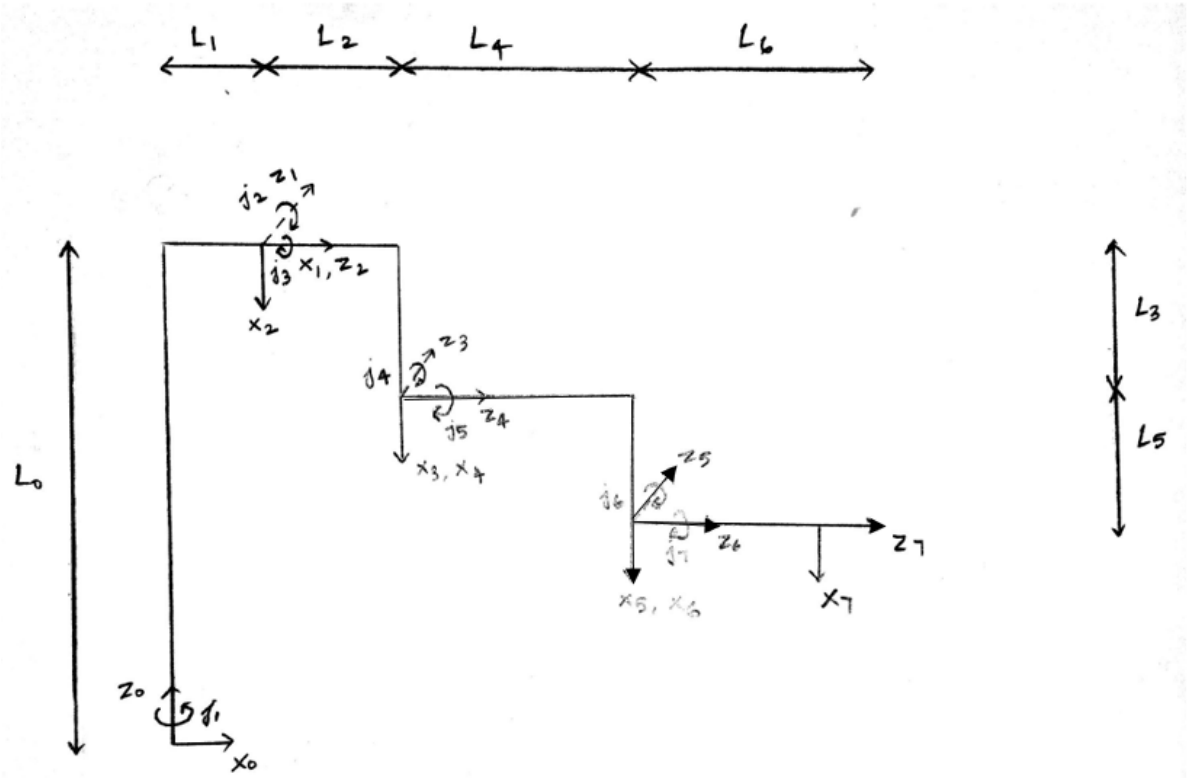


Figure 3: Frame Diagram for 7 DOF Baxter Left arm

i	d_i	θ_i	a_i	α_i
1	L0	J1 $=0^\circ$	L1	-90°
2	0	J2 $=90^\circ$	0	90°
3	L2	J3 $=0^\circ$	L3	-90°
4	0	J4 $=0^\circ$	0	90°
5	L4	J5 $=0^\circ$	L5	-90°
6	0	J6 $=0^\circ$	0	90°
7	L6	J7 $=0^\circ$	0	0°

Table 4: DH Parameters for the Left arm with 7 DOF

b) 7 DOF Baxter Left Arm – Altered DH parameters

The Frame diagram shown in the Fig.3 was broken down into furthermore frames which is shown below in Fig.5. This is because, some frame transformation in

the above table involved two translations. For example, moving from frame 0 → 1 involves two translations L0 and L1 in z - axis and x – axis respectively. Because of this, while simulating, the joints of the manipulator were not shown precisely. Below frame diagram shows the frame transformations with each frame involving only one translation either in the x – axis or in the z – axis.

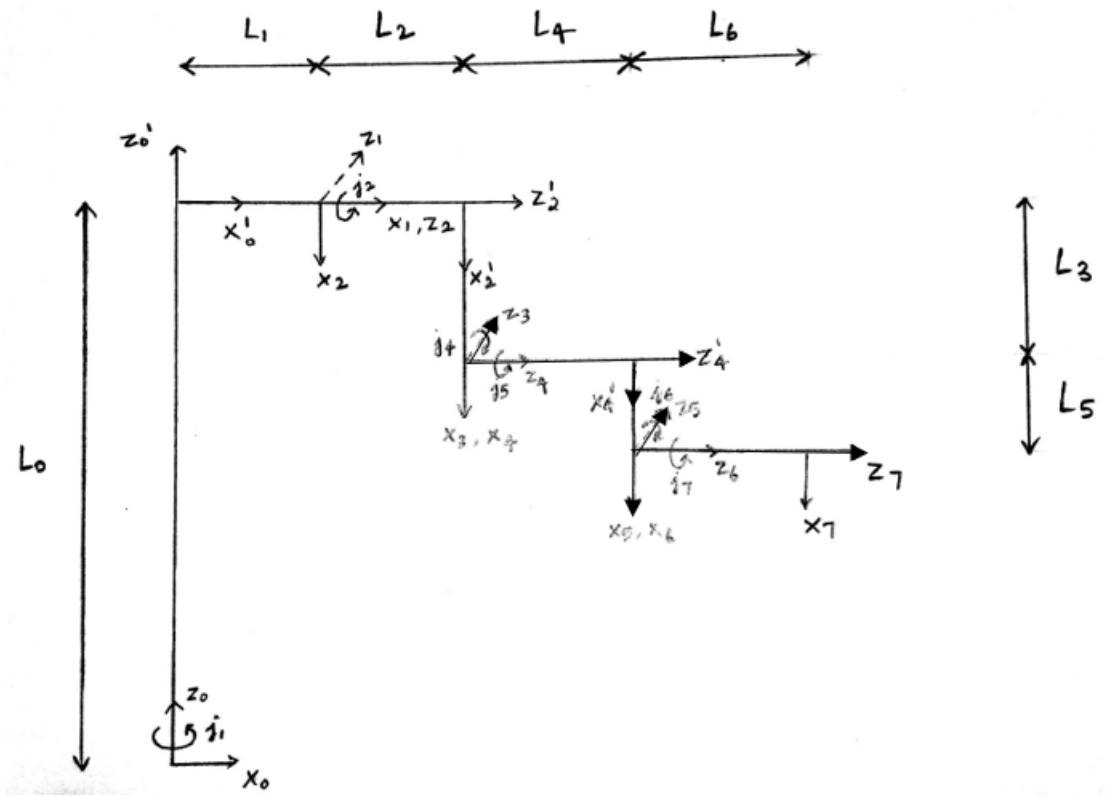


Figure 4: Altered Frame Diagram for 7 DOF Baxter Left arm

i	d_i	θ_i	a_i	α_i
0′	L0	J1=0	0	0
1	0	0	L1	-90
2	0	J2+90	0	90
2′	L2	J3	0	0
3	0	0	L3	-90
4	0	J4	0	90
4′	L4	J5	0	0
5	0	0	L5	-90
6	0	J6	0	90
7	L6	J7	0	0

Table 5: Altered DH Parameters for the Left arm with 7 DOF

c) 6 DOF Baxter Left arm

The Frame diagram and the DH table for the 6 DOF Baxter left arm is given in the Fig. 5 and Table.6 respectively. It can be noted that the joint angle j3 and frame 3 is not included and the distance between the frame 3 and frame 4 is given by simple hypotenuse calculation.

$$L_h = \sqrt{{L_2}^2 + {L_3}^2}$$

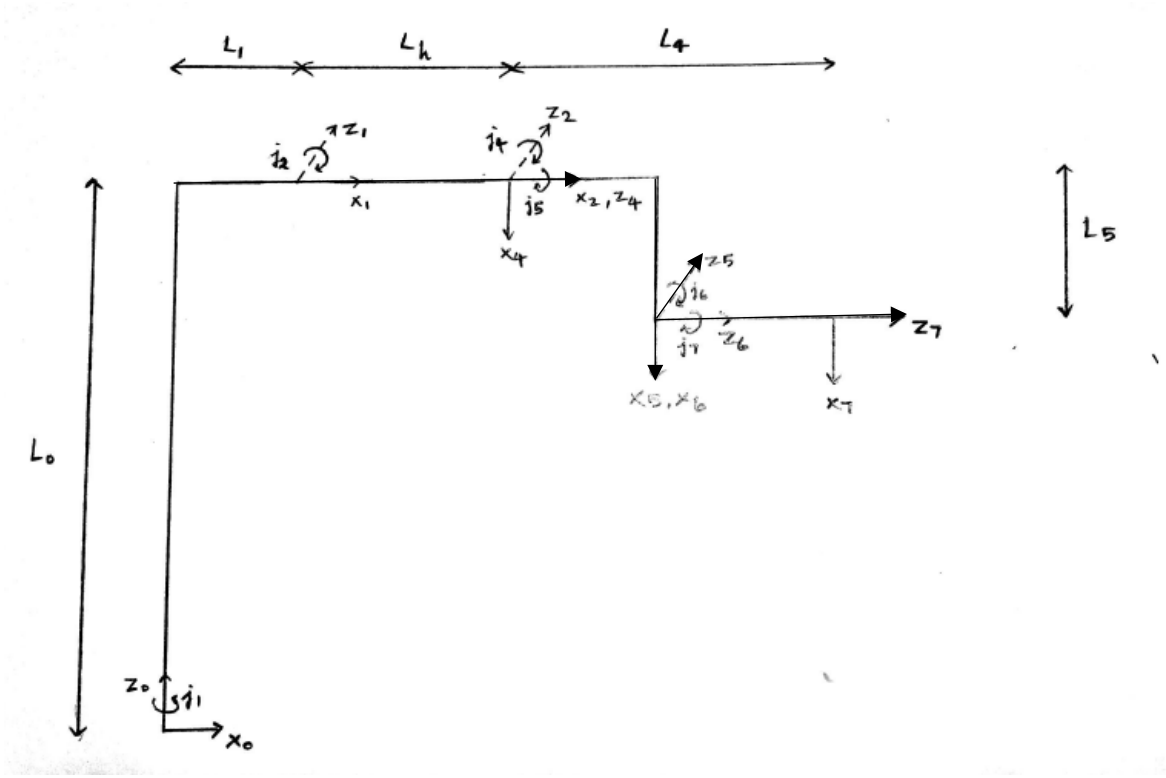


Figure 5: Frame Diagram for 6 DOF Baxter Left arm

i	d_i	θ_i	a_i	α_i
1	L0	J1=0	L1	-90
2	0	J2=0	L_h	0
4	0	J4+90	0	90
5	L4	J5	L5	-90
6	0	J6	0	90
7	L6	J7	0	0

Table 5: DH Parameters for the Left arm with 6 DOF

The transformation matrix T_0^7 is calculated from Matlab as follows,

$$T_0^7 = \begin{pmatrix} a11 & a12 & a13 & a14 \\ a21 & a22 & a23 & a24 \\ a31 & a32 & a33 & a34 \\ a41 & a42 & a43 & a44 \end{pmatrix}$$

Where,

```
a11 =

- cos(q7)*(sin(q6)*(cos(q4 + 90)*cos(q1)*sin(q2) + sin(q4 +
  90)*cos(q1)*cos(q2)) + cos(q6)*(sin(q1)*sin(q5) - cos(q5)*(cos(q4
  + 90)*cos(q1)*cos(q2) - sin(q4 + 90)*cos(q1)*sin(q2)))) -
  sin(q7)*(cos(q5)*sin(q1) + sin(q5)*(cos(q4 + 90)*cos(q1)*cos(q2) -
  sin(q4 + 90)*cos(q1)*sin(q2)))


a12 =

sin(q7)*(sin(q6)*(cos(q4 + 90)*cos(q1)*sin(q2) + sin(q4 +
  90)*cos(q1)*cos(q2)) + cos(q6)*(sin(q1)*sin(q5) - cos(q5)*(cos(q4
  + 90)*cos(q1)*cos(q2) - sin(q4 + 90)*cos(q1)*sin(q2)))) -
  cos(q7)*(cos(q5)*sin(q1) + sin(q5)*(cos(q4 + 90)*cos(q1)*cos(q2) -
  sin(q4 + 90)*cos(q1)*sin(q2)))


a13 =

cos(q6)*(cos(q4 + 90)*cos(q1)*sin(q2) + sin(q4 + 90)*cos(q1)*cos(q2))
  - sin(q6)*(sin(q1)*sin(q5) - cos(q5)*(cos(q4 + 90)*cos(q1)*cos(q2) -
  sin(q4 + 90)*cos(q1)*sin(q2)))


 a14 =

 L1*cos(q1) + L4*sin(q2 + q4 + 90)*cos(q1) + Lh*cos(q1)*cos(q2) +
   L6*sin(q2 + q4 + 90)*cos(q1)*cos(q6) - L6*sin(q1)*sin(q5)*sin(q6)
   + L6*cos(q4 + 90)*cos(q1)*cos(q2)*cos(q5)*sin(q6) - L6*sin(q4 +
   90)*cos(q1)*cos(q5)*sin(q2)*sin(q6)


a21 =

sin(q7)*(cos(q1)*cos(q5) - sin(q5)*(cos(q4 + 90)*cos(q2)*sin(q1)
  - sin(q4 + 90)*sin(q1)*sin(q2))) - cos(q7)*(sin(q6)*(cos(q4
  + 90)*sin(q1)*sin(q2) + sin(q4 + 90)*cos(q2)*sin(q1)) -
  cos(q6)*(cos(q1)*sin(q5) + cos(q5)*(cos(q4 + 90)*cos(q2)*sin(q1) -
  sin(q4 + 90)*sin(q1)*sin(q2))))


a22 =

sin(q7)*(sin(q6)*(cos(q4 + 90)*sin(q1)*sin(q2) + sin(q4 +
  90)*cos(q2)*sin(q1)) - cos(q6)*(cos(q1)*sin(q5) + cos(q5)*(cos(q4
  + 90)*cos(q2)*sin(q1) - sin(q4 + 90)*sin(q1)*sin(q2)))) +
  cos(q7)*(cos(q1)*cos(q5) - sin(q5)*(cos(q4 + 90)*cos(q2)*sin(q1) -
  sin(q4 + 90)*sin(q1)*sin(q2)))


a23 =

sin(q6)*(cos(q1)*sin(q5) + cos(q5)*(cos(q4 + 90)*cos(q2)*sin(q1)
  - sin(q4 + 90)*sin(q1)*sin(q2))) + cos(q6)*(cos(q4 +
  90)*sin(q1)*sin(q2) + sin(q4 + 90)*cos(q2)*sin(q1))
```

```
a24 =

L1*sin(q1) + L4*sin(q2 + q4 + 90)*sin(q1) + Lh*cos(q2)*sin(q1) +
 L6*sin(q2 + q4 + 90)*cos(q6)*sin(q1) + L6*cos(q1)*sin(q5)*sin(q6)
+ L6*cos(q4 + 90)*cos(q2)*cos(q5)*sin(q1)*sin(q6) - L6*sin(q4 +
90)*cos(q5)*sin(q1)*sin(q2)*sin(q6)

a31 =

sin(q2 + q4 + 90)*sin(q5)*sin(q7) - cos(q7)*(cos(q2 + q4 + 90)*sin(q6)
 + sin(q2 + q4 + 90)*cos(q5)*cos(q6))


 a32 =

 sin(q7)*(cos(q2 + q4 + 90)*sin(q6) + sin(q2 + q4 +
  90)*cos(q5)*cos(q6)) + sin(q2 + q4 + 90)*cos(q7)*sin(q5)


 a33 =

 cos(q2 + q4 + 90)*cos(q6) - sin(q2 + q4 + 90)*cos(q5)*sin(q6)


a34 =

L0 + L4*cos(q2 + q4 + 90) - Lh*sin(q2) + L6*(cos(q2 + q4 + 90)*cos(q6)
 - sin(q2 + q4 + 90)*cos(q5)*sin(q6))


a41 =     a42 =     a43 =     a44 =

0         0         0         1
```

d) Base Frame

The base frame for the Baxter's left arm is not fixed exactly near the shoulder of the robot. It is fixed near the base of the actual robot. The transformation from the base frame to the shoulder frame of the robot is given below based on the offset in three coordinate axes as per the Fig. 6 and Table 6. shown below.

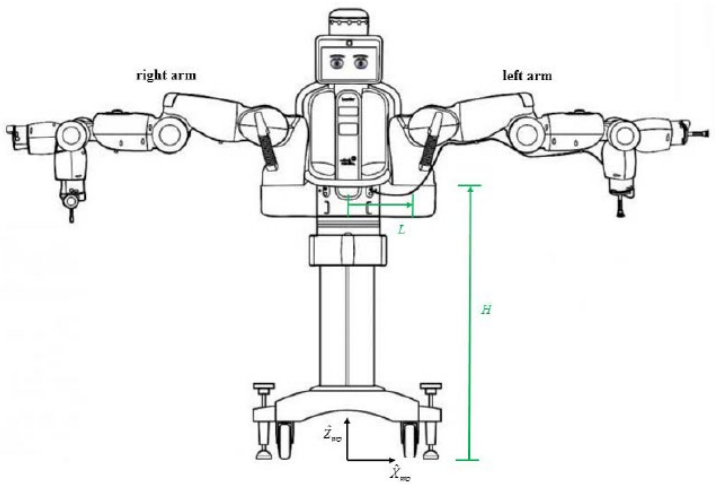


Figure 6: Offset from shoulder frame to the base frame

Length	Value (mm)
L	278
h	64
H	1104

Table 6: Offset values in millimetre

The transformation matrix from base frame to the left shoulder frame is given by,

TB_Ls =

$$\begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & L \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & -h \\ 0 & 0 & 1 & H \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The transformation from the Baxter's base frame to the end effector frame of the robot is given by,

$$T_{LeftArm} = [TB\_Ls][T_0^7]$$

Inverse Kinematics

The inverse kinematic solution is found out analytically for the 6 DOF Baxter left arm where the joint angle j_3 is set to zero and the link length L_5 is also assumed to be zero.

Given: The DH parameters of the manipulator and the required end effector pose.

$$T_0^7 = \begin{pmatrix} a11 & a12 & a13 & a14 \\ a21 & a22 & a23 & a24 \\ a31 & a32 & a33 & a34 \\ a41 & a42 & a43 & a44 \end{pmatrix}$$

To Calculate: The joint angles $(j1, j2, j4, j5, j6, j7)$ to achieve this pose

With j_3 set to zero, the rotation axis for j_2 and j_4 are parallel to each other. Since, L_5 is also set to zero, the resulting wrist of the Baxter robot is now spherical with three

consecutive frames sharing the same origin. The translational vector $d_0^7 = d_0^5$ is a function of (j1, j2 and j4) only.

Therefore, the solution approach is to solve (j1, j2, j4) given values for d_0^5 and then solve for (j5, j6, j7) from R_0^7

With j3 set to 0, the frame 3 and j3 has been removed from the table. The frames are now numbered as 1, 2, 4, 5, 6 and 7.

a) Translational joints solution (j1, j2, j4)

Joint angle j1 is calculated from the ratio of the Y to X translational equations:

```
x05 =

cos(q1)*(L1 + L4*sin(q2 + q4 + 90) + Lh*cos(q2))


y05 =

sin(q1)*(L1 + L4*sin(q2 + q4 + 90) + Lh*cos(q2))


z05 =

L0 + L4*cos(q2 + q4 + 90) - Lh*sin(q2)
```

From the above equations, dividing y05 by x05 gives the joint angle 1,

$$\frac{y05}{x05} = \frac{\sin(q1) * (L1 + L4 * \sin(q2 + q4 + 90) + Lh * \cos(q2))}{\cos(q1) * (L1 + L4 * \sin(q2 + q4 + 90) + Lh * \cos(q2))}$$

$$\tan(q1) = \frac{y05}{x05}$$

$$\boxed{j_1 = q_1 = atan2(y05, x05)}$$

From the above equations for x05, y05 and z05, the values for j2 and j4 can be calculated as follows,

x05 and z05 can be written as,

$$L_4 C_{24} = \frac{x05}{c1} - L_1 - L_h c_2$$

$$L_4 S_{24} = L_0 - z_{05} - L_h s_2$$

These two equations can be squared and added to eliminate the C_{24} and S_{24} terms. We get,

$$L_4^2 = \left(\frac{x05}{c1}\right)^2 + {L_1}^2 + {L_h}^2{c_2}^2 - \frac{2xL_1}{c_1} + 2L_1L_hc_2 - \frac{2x05L_hc_2}{c_1} + {L_0}^2 + {z_{05}}^2 + {L_h}^2{s_2}^2 - 2L_0z05 + 2z05L_hs_2 - 2L_0L_hs_2$$

This equation can be written in terms of the sines and cosines,

$$E\cos q_2 + F\sin q_2 + G = 0$$

Where,

$$E = 2L_1L_h - \frac{2x05L_hc_2}{c_1} = 2L_h\left(L_1 - \frac{x05}{c1}\right)$$

$$F = 2z05L_h - 2L_0L_h = 2L_h(z05 - L_0)$$

$$G = \left(\frac{x05}{c1}\right)^2 + {L_1}^2 + {L_h}^2 - \frac{2xL_1}{c_1} + {L_0}^2 + {z_{05}}^2 - 2L_0z05 - L_4^2$$

We can solve this equation for q_2 by using the Tangent Half – Angle subtraction, by defining as follows,

$$t = tan\frac{q_2}{2}, then\ \cos q_2 = \frac{1-t^2}{1+t^2}\ and\ \sin q_2 = \frac{2t}{1+t^2}$$

$$E\left(\frac{1-t^2}{1+t^2}\right) + F\left(\frac{2t}{1+t^2}\right) + G = 0$$

$$(G-E)t^2 + (2F)t + (G+E) = 0$$

We can solve for t by using the quadratic formula,

$$t = \frac{-F \pm \sqrt{E^2 + F^2 - G^2}}{G - E}$$

Now that the variable 't' is found, q_2 can be found from the equation $t = \tan\frac{q_2}{2}$

$$j_2 = q_2 = 2 * atan2(t)$$

Note that, we have two values for q2 based on $\pm$ in the quadratic formula.

Joint angle q_4 can be found by computing the ratio of x05 and z05,

$$j_4 = q_4 = atan2\left(L_0 - z_{05} - L_h\sin q_2, \frac{x05}{c1} - L_1 - L_hc_2\right) - q_2$$

b) Wrist joints solutions (q_5, q_6 and q_7)

The joint angles q_5, q_6 and q_7 can be calculated from the rotation matrix R_0^7.

The Transformation matrix T_0^7 can be written as,

$$T_0^4 T_4^7 = T_0^7$$

$$T_4^7 = (T_0^4)^{-1} T_0^7$$

The inverse of the matrix can be found by using the following formula,

$$H^{-1} = \begin{bmatrix} R^T & -R^T d \\ 0 & 1 \end{bmatrix}$$

The T_4^7 matrix after simplification in Matlab is given by,

```
T47 =

[ cos(q5)*cos(q6)*cos(q7) - sin(q5)*sin(q7), - cos(q7)*sin(q5) - cos(q5)*cos(q6)*sin(q7), cos(q5)*sin(q6), L6*cos(q5)*sin(q6)]
[ cos(q5)*sin(q7) + cos(q6)*cos(q7)*sin(q5),   cos(q5)*cos(q7) - cos(q6)*sin(q5)*sin(q7), sin(q5)*sin(q6), L6*sin(q5)*sin(q6)]
[                         -cos(q7)*sin(q6),                           sin(q6)*sin(q7),         cos(q6),    L4 + L6*cos(q6)]
[                                        0,                                        0,               0,                 1]
```

q_5, q_6 and q_7 can be calculated from the above matrix as follows,

$$\frac{r_{23}}{r_{13}} = \frac{s_5 s_6}{c_5 s_6}$$

$$\tan q_5 = \frac{r_{23}}{r_{13}}$$

$$\boxed{j_5 = q_5 = atan2(r_{23}, r_{13})}$$

$$\frac{r_{32}}{-r_{31}} = \frac{s_6 s_7}{c_7 s_6}$$

$$\boxed{j_7 = q_7 = atan2(r_{32}, -r_{31})}$$

Similarly,

$$\boxed{j_6 = q_6 = atan2(-r_{31}/\cos q_7, -r_{33})}$$

Validation

a) Forward Kinematics

Random angles for the 6 DOF Baxter arm are given in the Matlab script to compute the Transformation matrices. It can be noted that, the joint variables only contain 6 joint angles, i.e., $j_3 = 0$ is considered as a solid link with no motion around its axis.

The calculated transformation matrices corresponding to the joint angles are given below,

Joint Angles	Transformation Matrix
[50,0,35,45,35,45]	T07 = -0.8767 0.4801 -0.0297 460.6955 0.3699 0.7124 0.5964 793.0550 0.3075 0.5119 -0.8021 -246.1575 0 0 0 1.0000
[45,45,45,45,45,45]	T07 = -0.5000 0.5000 -0.7071 -26.1849 0.7071 0.7071 0 234.0304 0.5000 -0.5000 -0.7071 -627.1842 0 0 0 1.0000
[35,45,95,45,95,45]	T07 = 0.4664 -0.5135 -0.7202 -229.2530 0.8838 0.3040 0.3556 155.9307 0.0364 -0.8024 0.5956 -13.8200 0 0 0 1.0000
[45,10,45,22,45,10]	T07 = -0.9170 -0.2837 -0.2803 355.7334 -0.3204 0.9426 0.0943 493.5887 0.2374 0.1763 -0.9553 -453.0564 0 0 0 1.0000

b) Inverse Kinematics

The Transformation matrices calculated for different joint angles of the manipulator are now fed as input to the Inverse kinematics equations computed earlier and the corresponding joint angles are found. It can be found that the joint angles which were used to calculate the Transformation matrices earlier matches exactly with the joint angles found using the Inverse kinematic equations. This validates both our Forward kinematics model and the computed Inverse Kinematic equations.

Transformation Matrix

```
T07 =

   -0.8767    0.4801   -0.0297  460.6955
    0.3699    0.7124    0.5964  793.0550
    0.3075    0.5119   -0.8021 -246.1575
         0         0         0    1.0000
```

Joint Angles

```
Joint_Angles =

   50.0254   -0.0000   35.0178   45.0228   35.0178   45.0228
```

```
T07 =

   -0.5000    0.5000   -0.7071  -26.1849
    0.7071    0.7071         0  234.0304
    0.5000   -0.5000   -0.7071 -627.1842
         0         0         0    1.0000
```

```
Joint_Angles =

   45.0228   45.0228   45.0228   45.0228   45.0228   45.0228
```

```
T07 =

    0.4664   -0.5135   -0.7202 -229.2530
    0.8838    0.3040    0.3556  155.9307
    0.0364   -0.8024    0.5956  -13.8200
         0         0         0    1.0000
```

```
Joint_Angles =

   35.0178   45.0228   95.0482   45.0228   95.0482   45.0228
```

```
T07 =

   -0.9170   -0.2837   -0.2803  355.7334
   -0.3204    0.9426    0.0943  493.5887
    0.2374    0.1763   -0.9553 -453.0564
         0         0         0    1.0000
```

```
Joint_Angles =

   45.0228   10.0051   45.0228   22.0112   45.0228   10.0051
```

c) *Matlab Verification of the Kinematics of the Baxter Left arm for both 6 DOF and 7 DOF cases*

A custom GUI was created in Matlab using the uicontrol and the plot3 function. Configurable sliders with a range of -180° to +180° were provided in the GUI to manually vary the joint angles of the manipulator and validate the motion of the arm with respect to the corresponding change in the joint angle. Whenever the slider for a particular joint angle is changed, the joint angle is updated with the new angle and the corresponding Transformation matrix is computed. The newly computed Transformation matrix is then used to plot the new arm orientation in 3D space to visualize the changes made. The position coordinates of the end effector in the 3D coordinate system is plotted to show the values in real time whenever a change in the joint angle is made.

7 DOF Baxter Left Arm Simulation

Joint Angle [0, 0, 0, 0, 0, 0, 0]

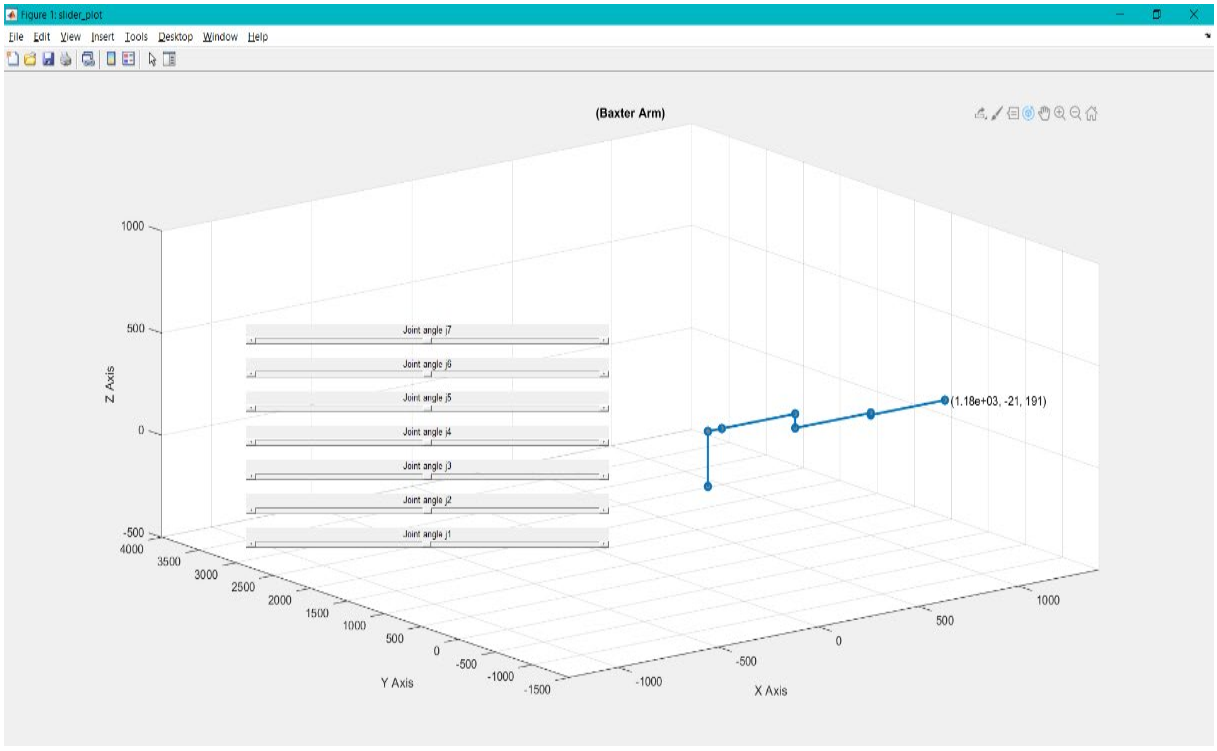


Figure 7 a: Matlab Simulation of 7 DOF Baxter Left Arm

Joint Angle [90, 0, 0, 0, 0, 0, 0]

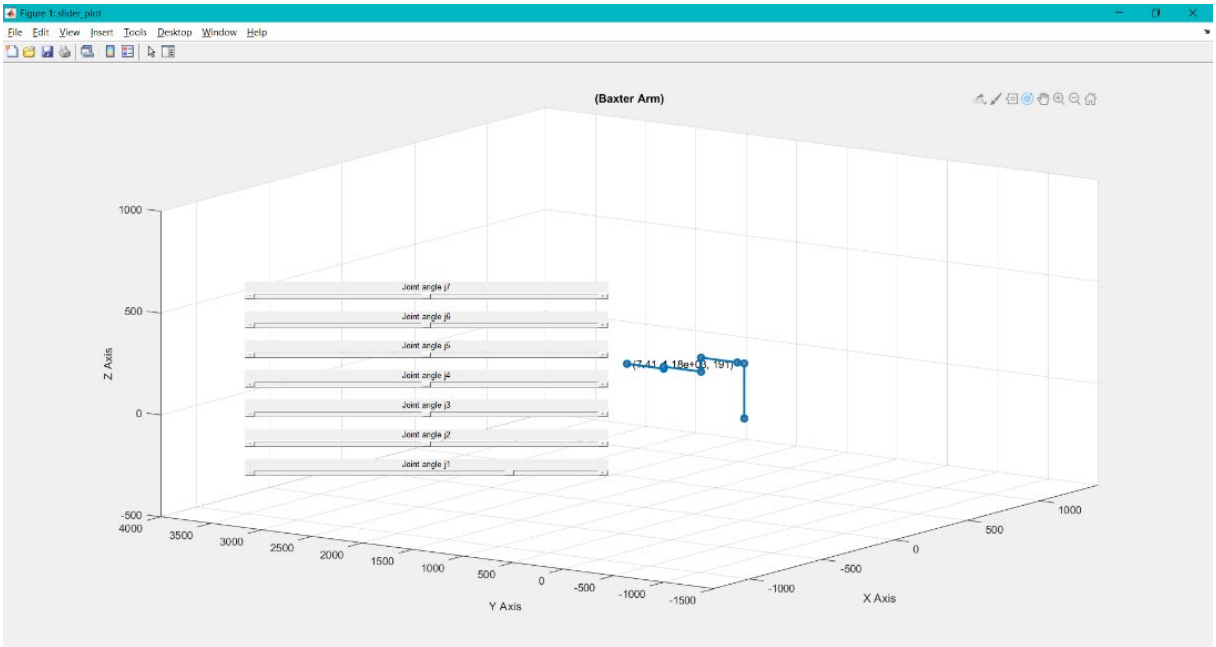


Figure 7 b: Matlab Simulation of 7 DOF Baxter Left Arm

Joint Angle [-20, 15, 0, -15, 0, 0, 0]

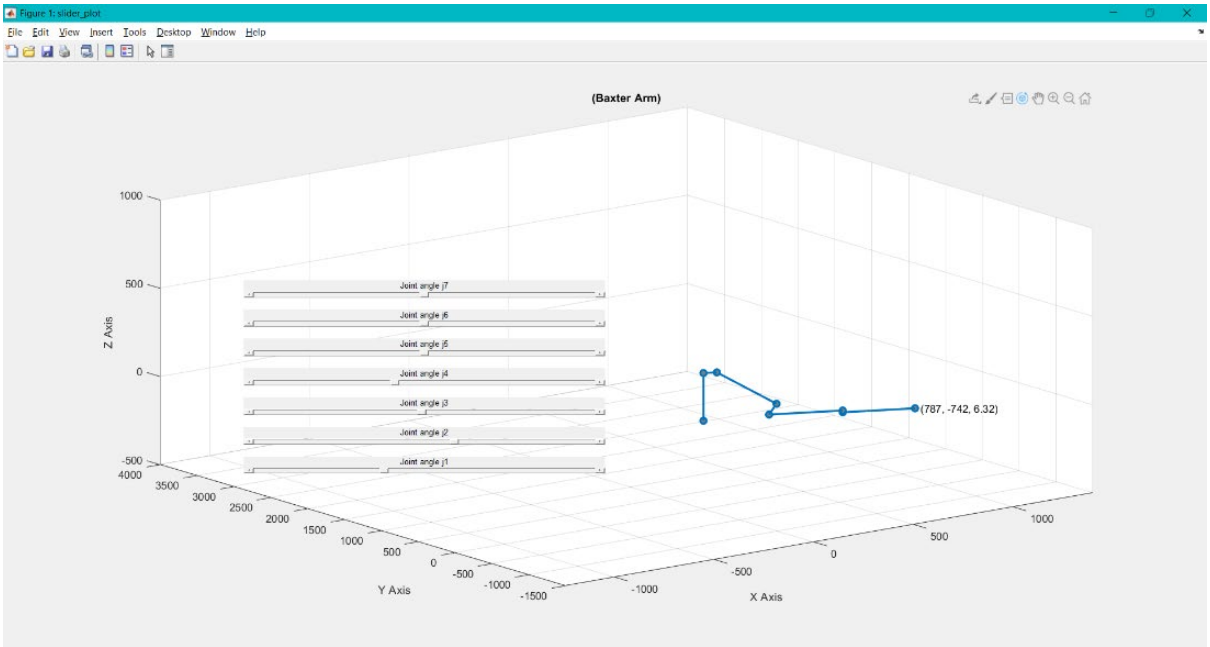


Figure 7 c: Matlab Simulation of 7 DOF Baxter Left Arm

6 DOF Baxter Left Arm Simulation

Joint Angle [0, 0, 0, 0, 0, 0]

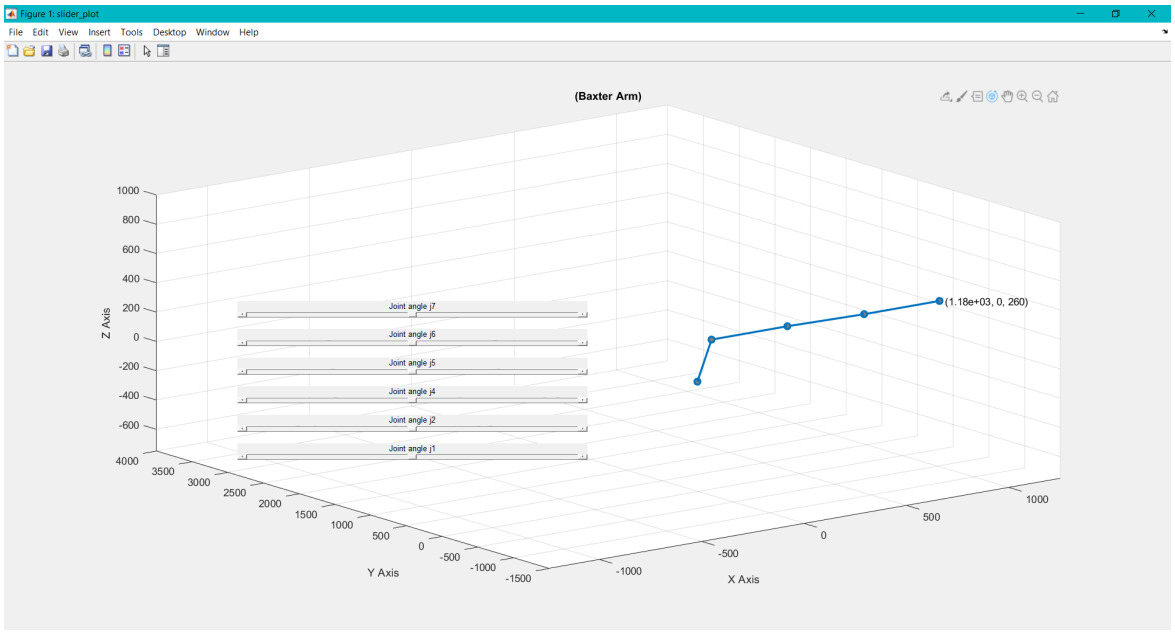


Figure 8 a: Matlab Simulation of 6 DOF Baxter Left Arm

Joint Angle [90, 0, 0, 0, 0, 0]

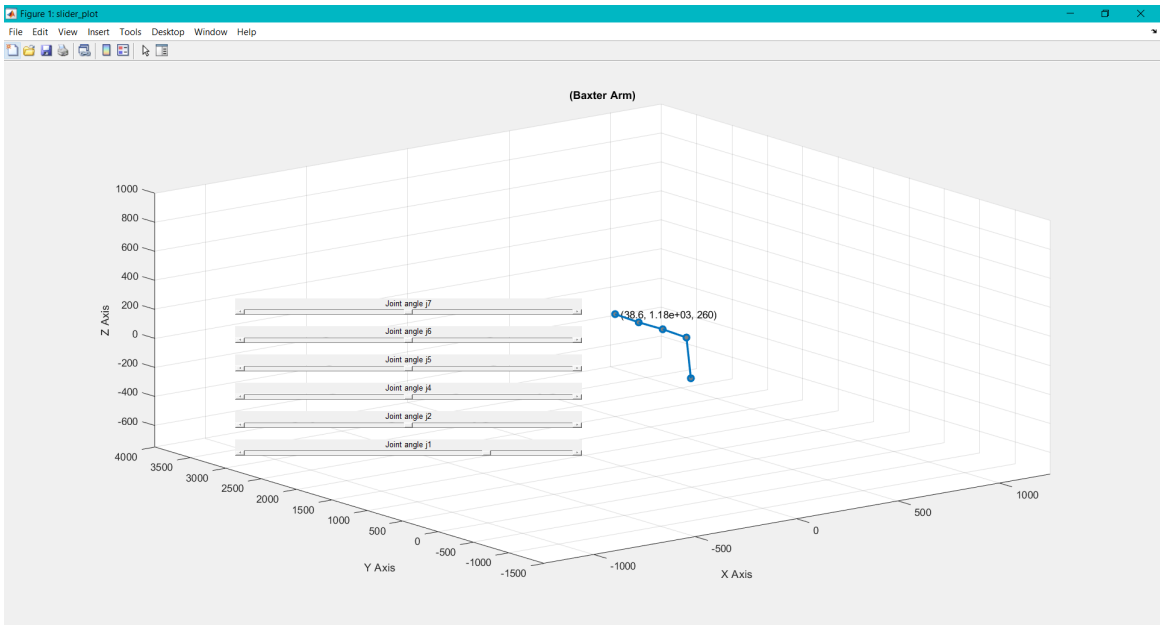


Figure 8 b: Matlab Simulation of 6 DOF Baxter Left Arm

Joint Angle [-90, 30, -90, 0, 0, 0]

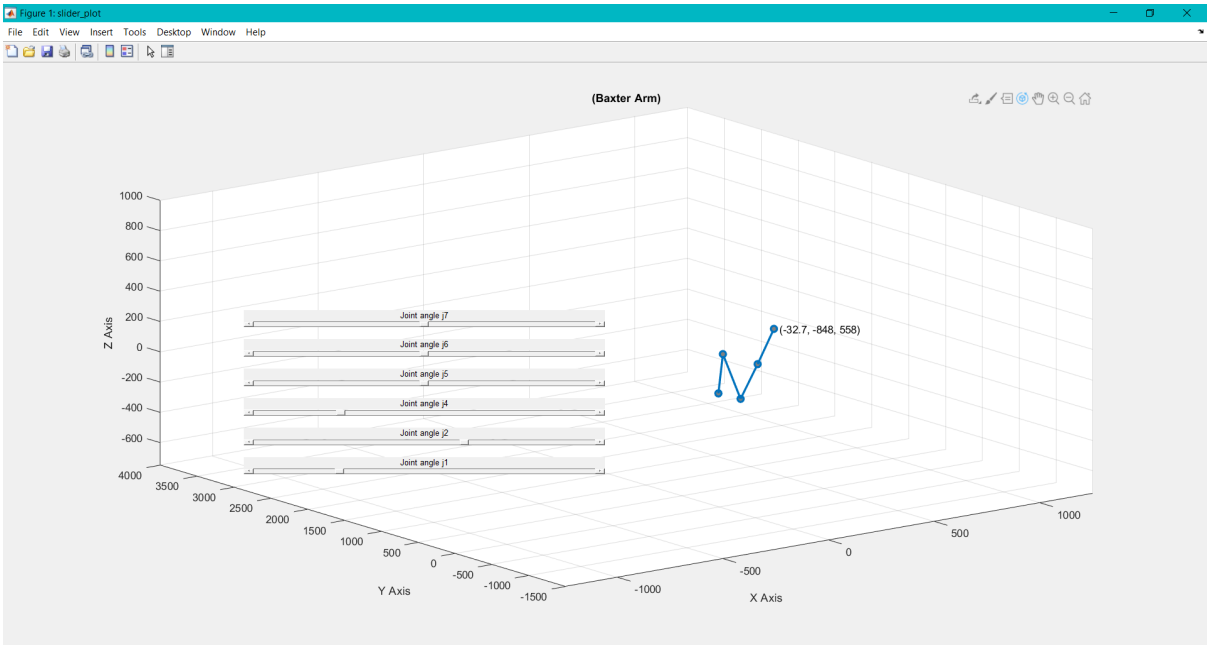


Figure 8 c: Matlab Simulation of 6 DOF Baxter Left Arm

d) Path Planning and Vrep Validation of Forward and Inverse Kinematics of the Baxter Left Arm

Since, the pose of the end effector and the basket is already known, a path is planned between the two points in 3D space. The path is nothing but a sequence of continuous points in 3D space. This path is broken down into a number of discrete points. The coordinates of these points are known and the end effector orientation at these points are calculated such that there is minimum slippage of the ball from the end effector. The end effector gets to the ball, opens the gripper, grips the ball, moves to the next set of discrete point in its pre-planned path and then finally drops the ball by opening the gripper when it reaches the final discrete point in its path.

The joint angles needed to position the end effector when it moves from one point to another is calculated in Matlab from the Inverse kinematics equations shown above. A set of 6 points are chosen for the path of the arm from the table to the basket and the corresponding joint angles are calculated.

The path planned and the corresponding Lua script is as follows,

- Open the Gripper (Giving a '0' in this script, opens the gripper)
  sim.setIntegerSignal (gripperName..'_close', 0)
- Move to the first location (Moving from Origin to the table)

Config = {-69*math.pi/180, 0*math.pi/180, 0*math.pi/180, 0*math.pi/180, 0*math.pi/180, 0*math.pi/180, 0*math.pi/180}

- Move to the second point (Bend down on the table to reach the ball)
  Config = {-69*math.pi/180, 13*math.pi/180, 0*math.pi/180, 0*math.pi/180, 0*math.pi/180, 80*math.pi/180, 0*math.pi/180}
- Close the Gripper (Giving a '1' in this script, closes the Gripper)
  sim.setIntegerSignal(gripperName..'_close',1)
- Move to the third location (Orient the end effector such that the ball doesn't slip)
  Config = {-60*math.pi/180, 0*math.pi/180, 0*math.pi/180, 0*math.pi/180, 0*math.pi/180, 0*math.pi/180, 0*math.pi/180}
- Move to the fourth location (Moving from Table to the top of box)
  Config = {47*math.pi/180, 0*math.pi/180, 0*math.pi/180, 0*math.pi/180, 0*math.pi/180, 0*math.pi/180, 0*math.pi/180}
- Move to the fifth location (Bend the manipulator to reach the box)
  Config = {47*math.pi/180, 10*math.pi/180, -5*math.pi/180, -2.8*math.pi/180, 6*math.pi/180, 85*math.pi/180, 0*math.pi/180}
- Open the Gripper (To drop the ball into the box)
  sim.setIntegerSignal(gripperName..'_close',0)

Note: In the actual script, a delay script is given after each operation based on what the operation is. This is mainly done to ensure smooth transition from one operation to the other.

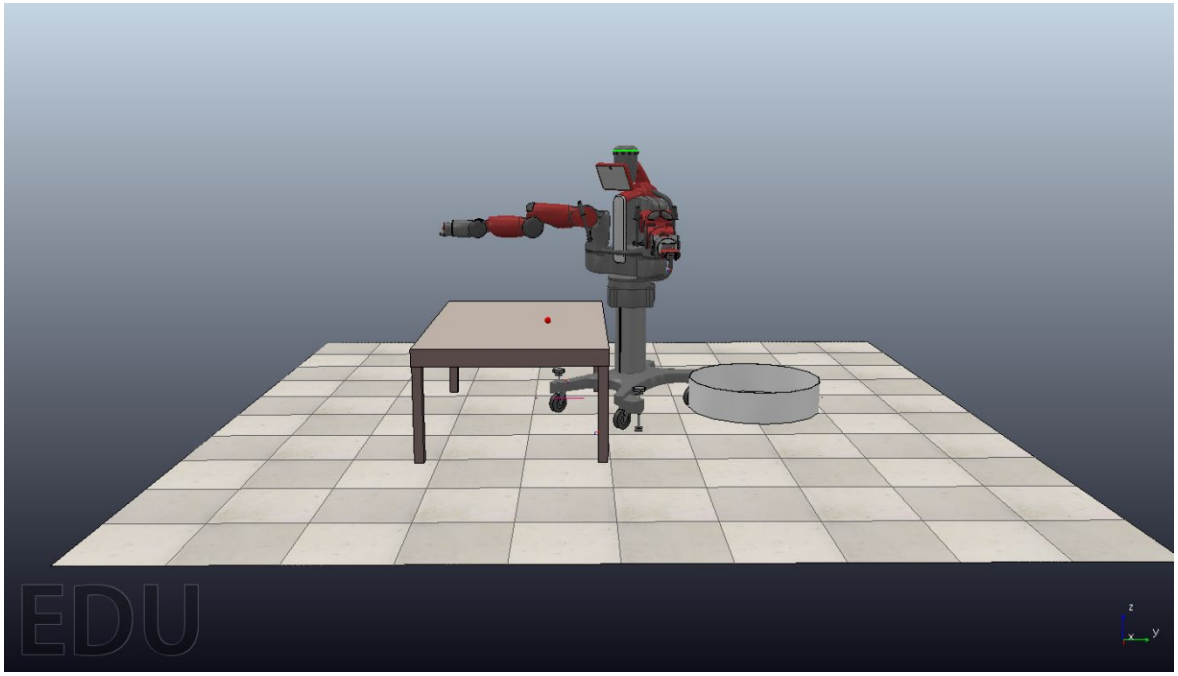


Figure 9: Vrep simulation of the pick and drop operation

Tools and Software used

a) Matlab for Modelling and Simulation

MATLAB is a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs easier.

The symbolic toolbox helps in solving equations with multiple variables and allows us to substitute the numerical values later.

There are functions to,

Sym - allows creation and use of symbolic expressions in the script

Simplify() - condense an algebraic expression by grouping and combining similar terms

Solve() – solves a system of equations

Uicontrol() – creates a custom UI

Plot3() - displays a three-dimensional plot of a set of data points

b) Simulation in Vrep

Vrep is one of the most versatile and powerful robot simulation platforms available. Vrep contains CAD models of most of the commonly used Robots available in the market and also allows the user to add his custom-made robot into the Vrep scene for simulation. Apart from the robot, Vrep also allows us to configure the robot environment by adding different real-world elements into the scene. Baxter being a commonly known versatile robot, the Baxter robot model was readily available in Vrep and was imported to the scene. Other elements including the ball, table and the baskets were easily created in Vrep.

The joint angles for the pre-planned path were calculated in Matlab with the Inverse Kinematics equations and were directly used in the Vrep scene to pick and drop the ball from the table into the basket.

Future Work

- Dynamic modelling of the robot to better understand the velocity and torque constraints.
- Extend the Forward and Inverse kinematics to both the arms of the robot
- Trajectory planning allowing the robot to throw the ball into the basket placed outside of the workspace of the robot.
- Modelling of a Gripper which is capable of picking up a variety of commonly used items
- Effective path planning considering the obstacles in real world environment.

Conclusion

Any point in the 3D space can be reached using the 6 DOF of the robot. The Baxter robot arm with 7 DOF was modelled as a 6 DOF robot by locking one of its joint angles to zero to avoid redundancy even though it allows the robot to reach hard to access points easily.

The Forward Kinematics of the Baxter Left arm were obtained using the D-H parameters. The inverse kinematics of the Baxter arm was calculated analytically by modifying the Baxter arm to obtain a spherical wrist. The kinematics of the robot were formulated and validated in Matlab. The Baxter with 6 DOF left arm was successfully made to pick and drop the ball into a basket in the Vrep simulation. Future work and possible improvements were also discussed.

References

[1] CORPORATION., T. (2019). The Development Diary of CUE, the AI Basketball Robot: Second story | Corporate | Global Newsroom | Toyota Motor Corporation Official Global Website. [online] Toyota Motor Corporation Official Global Website. Available at: https://global.toyota/en/newsroom/corporate/28595150.html [Accessed 6 Oct. 2019].

[2] Ohio.edu. (2019). [online] Available at: https://www.ohio.edu/mechanical/faculty/Williams/html/pdf/BaxterKinematics.pdf [Accessed 8 Nov. 2019].

[3] Sawyer (2019). [online] Available at: https://www.generationrobots.com/en /401 - 517 electric-parallel-gripper-kit-for-baxter-research-robot.html [Accessed 13 Nov. 2019].

[4] Mathworks.com. (2019). Create symbolic variables, expressions, functions, matrices - MATLAB sym. [online] Available at: https://www.mathworks.com /help/symbolic/sym.html [Accessed 9 Dec. 2019].

[5] Coppeliarobotics.com. (2019). *Robot simulator CoppeliaSim: create, compose, simulate, any robot. - Coppelia Robotics*. [online] Available at: http://www.coppeliarobotics.com/ [Accessed 9 Dec. 2019].

[6] Sdk.rethinkrobotics.com. (2019). *Hardware Specifications - sdk-wiki*. [online] Available at: https://sdk.rethinkrobotics.com/wiki/Hardware_Specifications [Accessed 9 Dec. 2019].

[7] GitHub. (2019). *RethinkRobotics/sdk-docs*. [online] Available at: https://github.com/RethinkRobotics/sdk-docs/wiki/What-kind-of-grippers-works-with-the-baxter-research-robot [Accessed 9 Dec. 2019].

[8] Coppeliarobotics.com. (2019). *Inverse kinematics tutorial*. [online] Available at: http://www.coppeliarobotics.com/helpFiles/en/inverseKinematicsTutorial.htm [Accessed 9 Dec. 2019].

[9] SPONG, M. (2012). *ROBOT MODELING AND CONTROL*. JOHN WILEY & Sons.

[10] J. Denavit and R.S. Hartenberg, 1955, A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices, Journal of Applied Mechanics: 215-221.

Definitions and Abbreviations

Symbols	Definitions
α_i	Twist angle along x - axis
d_i	Linear Displacement in z - axis
a_i	Linear Displacement in x - axis
θ_i	Twist angle along z - axis
s_0	Shoulder 0 (Baxter joint)
s_1	Shoulder 1 (Baxter joint)
e_0	Elbow 0 (Baxter joint)
e_1	Elbow 1 (Baxter joint)
w_0	Wrist 0 (Baxter joint)
w_1	Wrist 1 (Baxter joint)
w_2	Wrist 2 (Baxter joint)

Appendix

Appendix A: Matlab code for 6 DOF Baxter Left arm – FK and IK Validation

```
clc;
clear all;

%%Baxter 6 DOF Forward and Inverse Kinematics Validation%%

syms q1 q2 q3 q4 q5 q6 q7 alpha1 alpha2 alpha3 alpha4 alpha5
syms alpha6 alpha7 L0 L1 L2 L3 L4 L5 L6 L7 Lh a1 a2 a3 a4 a5 d1 d2 d3 d4
syms d5 d6 d7

%%Input Joint Variables

disp('Input Joint Angles to compute the Transformation matrix')

q1=45;
q2=10;
q4=45;
q5=22;
q6=45;
q7=10;

Q=[q1,q2,q4,q5,q6,q7]

%%Link Lengths
L0=270;
L1=69;
L2=364;
L3=69;
L4=375;
L5=0;
L6=368;
Lh=370.48;

q1=q1;
q2=q2;
q4=q4+90;
q5=q5;
q6=q6;
q7=q7;

alpha1=-90;
alpha2=0;
alpha4=90;
alpha5=-90;
alpha6=90;
alpha7=0;

a1=L1;
a2=Lh;
a4=0;
a5=L5;
a6=0;
a7=0;
```

```
d1=L0;
d2=0;
d4=0;
d5=L4;
d6=0;
d7=L6;


%%0-1 matrix
Rz_theta1=[cosd(q1) -sind(q1) 0 0;sind(q1) cosd(q1) 0 0;0 0 1 0;0 0 0
 1];
Rx_alpha1=[1 0 0 0;0 cosd(alpha1) -sind(alpha1) 0;0 sind(alpha1)
 cosd(alpha1) 0;0 0 0 1];
Tz_d1=[1 0 0 0;0 1 0 0;0 0 1 d1;0 0 0 1];
Tx_a1=[1 0 0 a1;0 1 0 0;0 0 1 0;0 0 0 1];

%%1-2 matrix
Rz_theta2=[cosd(q2) -sind(q2) 0 0;sind(q2) cosd(q2) 0 0;0 0 1 0;0 0 0
 1];
Rx_alpha2=[1 0 0 0;0 cosd(alpha2) -sind(alpha2) 0;0 sind(alpha2)
 cosd(alpha2) 0;0 0 0 1];
Tz_d2=[1 0 0 0;0 1 0 0;0 0 1 d2;0 0 0 1];
Tx_a2=[1 0 0 a2;0 1 0 0;0 0 1 0;0 0 0 1];

%%2-4 matrix
Rz_theta4=[cosd(q4) -sind(q4) 0 0;sind(q4) cosd(q4) 0 0;0 0 1 0;0 0 0
 1];
Rx_alpha4=[1 0 0 0;0 cosd(alpha4) -sind(alpha4) 0;0 sind(alpha4)
 cosd(alpha4) 0;0 0 0 1];
Tz_d4=[1 0 0 0;0 1 0 0;0 0 1 d4;0 0 0 1];
Tx_a4=[1 0 0 a4;0 1 0 0;0 0 1 0;0 0 0 1];

%%4-5 matrix
Rz_theta5=[cosd(q5) -sind(q5) 0 0;sind(q5) cosd(q5) 0 0;0 0 1 0;0 0 0
 1];
Rx_alpha5=[1 0 0 0;0 cosd(alpha5) -sind(alpha5) 0;0 sind(alpha5)
 cosd(alpha5) 0;0 0 0 1];
Tz_d5=[1 0 0 0;0 1 0 0;0 0 1 d5;0 0 0 1];
Tx_a5=[1 0 0 a5;0 1 0 0;0 0 1 0;0 0 0 1];

%%5-6 matrix
Rz_theta6=[cosd(q6) -sind(q6) 0 0;sind(q6) cosd(q6) 0 0;0 0 1 0;0 0 0
 1];
Rx_alpha6=[1 0 0 0;0 cosd(alpha6) -sind(alpha6) 0;0 sind(alpha6)
 cosd(alpha6) 0;0 0 0 1];
Tz_d6=[1 0 0 0;0 1 0 0;0 0 1 d6;0 0 0 1];
Tx_a6=[1 0 0 a6;0 1 0 0;0 0 1 0;0 0 0 1];

%%6-7 matrix
Rz_theta7=[cosd(q7) -sind(q7) 0 0;sind(q7) cosd(q7) 0 0;0 0 1 0;0 0 0
 1];
Rx_alpha7=[1 0 0 0;0 cosd(alpha7) -sind(alpha7) 0;0 sind(alpha7)
 cosd(alpha7) 0;0 0 0 1];
```

```
Tz_d7=[1 0 0 0;0 1 0 0;0 0 1 d7;0 0 0 1];
Tx_a7=[1 0 0 a7;0 1 0 0;0 0 1 0;0 0 0 1];



%%multiplying to get homogeneous matrices
A_1 = (Rz_theta1*Tz_d1*Tx_a1*Rx_alpha1);
A_2 =(Rz_theta2*Tz_d2*Tx_a2*Rx_alpha2);
A_4 =(Rz_theta4*Tz_d4*Tx_a4*Rx_alpha4);
A_5 =(Rz_theta5*Tz_d5*Tx_a5*Rx_alpha5);
A_6 =(Rz_theta6*Tz_d6*Tx_a6*Rx_alpha6);
A_7 =(Rz_theta7*Tz_d7*Tx_a7*Rx_alpha7);


%Transformation matrices
T01=A_1;
T02=A_1*A_2;
T04=A_1*A_2*A_4;
%T04=simplify(T04)
T05=A_1*A_2*A_4*A_5;
%T05=simplify(T05)
T06=A_1*A_2*A_4*A_5*A_6;
disp('Final Transformation matrix for the input joint angles')
T07=A_1*A_2*A_4*A_5*A_6*A_7

%Computing Theta5, Theta6, Theta7
R04=T04(1:3,1:3);
D04=T04(1:3,4);
R04T=transpose(R04);
R04TD04= R04T*D04;
T04inv=[R04T -R04T*D04;0 0 0 1];
%T04inv=simplify(T04inv);
T47=T04inv*T07;
%T47=simplify(T04inv*T07)

%Computing theta5
r23=T47(2,3);
r13=T47(1,3);
theta5=(atan2(r23,r13))*180/3.14;

%Computing theta 7
r32=T47(3,2);
r31=T47(3,1);
theta7=(atan2(r32,-r31))*180/3.14;

%Computing theta6
r33=T47(3,3);
theta6=(atan2((-r31/cosd(q7)),r33))*180/3.14;

%Computing theta1
x05=T05(1,4);
%x05=simplify(x05)
y05=T05(2,4);
%y05=simplify(y05)
```

```
z05=T05(3,4);

%z05=simplify(z05)
theta1=(atan2(y05,x05))*180/3.14;

%Computing theta2
E = 2*Lh*(L1-(x05/cosd(q1)));
F = 2*Lh*(z05-L0);
G = (x05/cosd(q1))^2+(L1^2)+(Lh^2)+(z05^2)+(L0^2)-(L4^2)-((2*x05*L1)/
cosd(q1))-2*z05*L0;
t = (-F-sqrt((E^2)+(F^2)-(G^2)))/(G-E);
theta2 = 2*atan(t);
teta2=theta2*180/3.14;


%Computing theta4
theta4 = (atan2(L0-z05-Lh*sind(q2),(x05/cosd(q1))-L1-(Lh*cosd(q2)))-
theta2)*180/3.14;


%Final joint variables
j1=theta1;
j2=teta2;
j4=theta4;
j5=theta5;
j6=theta6;
j7=theta7;

disp('Joint angles calculated from the Transformation matrix above')
Joint_Angles = [j1,j2,j4,j5,j6,j7]

Input Joint Angles to compute the Transformation matrix

Q =

    45    10    45    22    45    10

Final Transformation matrix for the input joint angles

T07 =

   -0.9170   -0.2837   -0.2803  355.7334
   -0.3204    0.9426    0.0943  493.5887
    0.2374    0.1763   -0.9553 -453.0564
         0         0         0    1.0000

Joint angles calculated from the Transformation matrix above

Joint_Angles =

   45.0228   10.0051   45.0228   22.0112   45.0228   10.0051
```

Appendix B: Matlab code for 7 DOF Baxter Left arm simulation with custom GUI

```
function [] = slider_plot()
% Plot different plots according to slider location.
% Junk data for comparison of adjusted curves in this example

S.fh = figure('units','normalized',...
    'Position', [0.515 0.025 0.415 0.87],... %%%%
              'name','slider_plot');


% Define inital parameter values for curve
S.a = 0;
S.b = 0;
S.c = 0;
S.d = 0;
S.e = 0;
S.f = 0;
S.g = 0;

update(S);

% 1st Slider:
S.aSlider = uicontrol('style','slider',...
              'unit','normalized',...
              'position',[0.2 0.3 0.3 0.01],...
              'min',-180,'max',180,'value', S.a,...
              'sliderstep',[0.003 0.03],...
              'callback', {@SliderCB, 'a'});
% Add a text uicontrol to label the slider.
txta = uicontrol('Style','text',...
     'unit','normalized',...
              'position',[0.2 0.31 0.3 0.02],...
     'String','Joint angle j1');

% 2nd Slider:
S.bSlider = uicontrol('style','slide',...
              'unit','normalized',...
              'position',[0.2 0.35 0.3 0.01],...
              'min',-180,'max',180,'value', S.b,...
              'sliderstep',[0.003 0.03],...
              'callback', {@SliderCB, 'b'});
% Add a text uicontrol to label the slider.
txtb = uicontrol('Style','text',...
```

```
      'unit','normalized',...
                'position',[0.2 0.36 0.3 0.02],...
      'String','Joint angle j2');

% 3rd Slider:
S.cSlider = uicontrol('style','slide',...
                'unit','normalized',...
                'position',[0.2 0.4 0.3 0.01],...
                'min',-180,'max',180,'value', S.c,...
                'sliderstep',[0.003 0.03],...
                'callback', {@SliderCB, 'c'});
% Add a text uicontrol to label the slider.
txtc = uicontrol('Style','text',...
      'unit','normalized',...
                'position',[0.2 0.41 0.3 0.02],...
      'String','Joint angle j3');

% 4th Slider:
S.dSlider = uicontrol('style','slide',...
                'unit','normalized',...
                'position',[0.2 0.45 0.3 0.01],...
                'min',-180,'max',180,'value', S.d,...
                'sliderstep',[0.003 0.03],...
                'callback', {@SliderCB, 'd'});
% Add a text uicontrol to label the slider.
txtd = uicontrol('Style','text',...
      'unit','normalized',...
                'position',[0.2 0.46 0.3 0.02],...
      'String','Joint angle j4');

 % 5th Slider:
S.eSlider = uicontrol('style','slide',...
                'unit','normalized',...
                'position',[0.2 0.5 0.3 0.01],...
                'min',-180,'max',180,'value', S.e,...
                'sliderstep',[0.003 0.03],...
                'callback', {@SliderCB, 'e'});
% Add a text uicontrol to label the slider.
txtd = uicontrol('Style','text',...
      'unit','normalized',...
                'position',[0.2 0.51 0.3 0.02],...
      'String','Joint angle j5');

  % 6th Slider:
S.fSlider = uicontrol('style','slide',...
                'unit','normalized',...
                'position',[0.2 0.55 0.3 0.01],...
                'min',-180,'max',180,'value', S.f,...
                'sliderstep',[0.003 0.03],...
                'callback', {@SliderCB, 'f'});
% Add a text uicontrol to label the slider.
txtd = uicontrol('Style','text',...
      'unit','normalized',...
                'position',[0.2 0.56 0.3 0.02],...
```

```
     'String','Joint angle j6');

  % 7th Slider:
S.eSlider = uicontrol('style','slide',...
                'unit','normalized',...
                'position',[0.2 0.6 0.3 0.01],...
                'min',-180,'max',180,'value', S.g,...
                'sliderstep',[0.003 0.03],...
                'callback', {@SliderCB, 'g'});
% Add a text uicontrol to label the slider.
txtd = uicontrol('Style','text',...
     'unit','normalized',...
                'position',[0.2 0.61 0.3 0.02],...
     'String','Joint angle j7');


guidata(S.fh, S);  % Store S structure in the figure
end

% Callback for all sliders defined above
function SliderCB(aSlider, EventData, Param)
S = guidata(aSlider);  % Get S structure from the figure
S.(Param) = get(aSlider, 'Value');  % Any of the 'a', 'b', etc.
 defined
update(S);  % Update the plot values
guidata(aSlider, S);  % Store modified S in figure
end
% Plot update function, creates new y-vector for plot and replaces the
 plot
% S.p2 with new y-vector
function update(S)
h1=FKdraw(S.a,S.b,S.c,S.d,S.e,S.f,S.g)
end
```

Fk Draw Function (Plots the Links in 3D space)

```
function [ FK ] = FKdraw( j1,j2,j3,j4,j5,j6,j7)

%initial parameter
%[j0 j1 j2 j3;d0 d1 d2 d3;a0 a1 a2 a3;t0 t1 t2 t3]
```

```
% D-H Parameters
a1 = 69; % length of first arm
a2 = 0; % length of second arm
a3 = 69; % length of third arm
a4 = 0; % length of fourth arm
a5 = 10; % length of fifth arm
a6 = 0; % length of sixth arm
a7 = 0; % length of seventh arm
d1 = 270; % offset of first arm
d2 = 0; % offset of second arm
d3 = 364; % offset of third arm
d4 = 0; % offset of fourth arm
d5 = 375; % offset of fifth arm
d6 = 0; % offset of sixth arm
d7 = 368; % offset of seventh arm

j=[j1 0 j2+90 j3 0 j4 j5 0 j6 j7;d1 0 0 d3 0 0 d5 0 0 d7;0 a1 0 0 a3 0
 0 a5 0 0;0 -90 90 0 -90 90 0 -90 90 0];

FK=DHkine(j);
Q=XYZkine(FK);

h=plot3(Q(1,:),Q(2,:),Q(3,:),'-
o','LineWidth',2,'MarkerSize',6,'MarkerFaceColor',[0.5,0.5,0.5]);



grid on;clc
text(Q(1,11),Q(2,11),Q(3,11),['  (', num2str(Q(1,11),3), ', ',
 num2str(Q(2,11),3),', ', num2str(Q(3,11),3), ')']);
title('(Baxter Arm)');
xlabel('X Axis');
ylabel('Y Axis');
zlabel('Z Axis');
axis([-1250 1400 -1500 4000 -500 1000]);
h = rotate3d;
h.Enable = 'on';
h.ActionPostCallback = @mypostcallback;
assignin('base','FK',FK);

end

function mypostcallback(obj,evd)
%disp('A rotation is about to occur.');
ax_properties = get(gca);
assignin('base','pov',ax_properties.View);
end

%use evalin('base',a) to get variable a from workspace
%assignin('base','a_rms',a_rms) to write variable a_rms to workspace
```

DHkine Function (Computes the Transformation Matrices)

```
function [ FK ] = DHkine(j)
%collum 1=joint angle, 2=joint offset, 3=link lenght, 4=twist angle
T01=DHmatrix(j(1,1),j(2,1),j(3,1),j(4,1));
T12=DHmatrix(j(1,2),j(2,2),j(3,2),j(4,2));
T23=DHmatrix(j(1,3),j(2,3),j(3,3),j(4,3));
T34=DHmatrix(j(1,4),j(2,4),j(3,4),j(4,4));
T45=DHmatrix(j(1,5),j(2,5),j(3,5),j(4,5));
T56=DHmatrix(j(1,6),j(2,6),j(3,6),j(4,6));
T67=DHmatrix(j(1,7),j(2,7),j(3,7),j(4,7));
T78=DHmatrix(j(1,8),j(2,8),j(3,8),j(4,8));
T89=DHmatrix(j(1,9),j(2,9),j(3,9),j(4,9));
T910=DHmatrix(j(1,10),j(2,10),j(3,10),j(4,10));


T02=T01*T12;
T03=T02*T23;
T04=T03*T34;
T05=T04*T45;
T06=T05*T56;
T07=T06*T67;
T08=T07*T78;
T09=T08*T89;
T010=T09*T910;


FK=[T01 T02 T03 T04 T05 T06 T07 T08 T09 T010];
end
```

DH matrix Function (Contains the General DH Matrix)

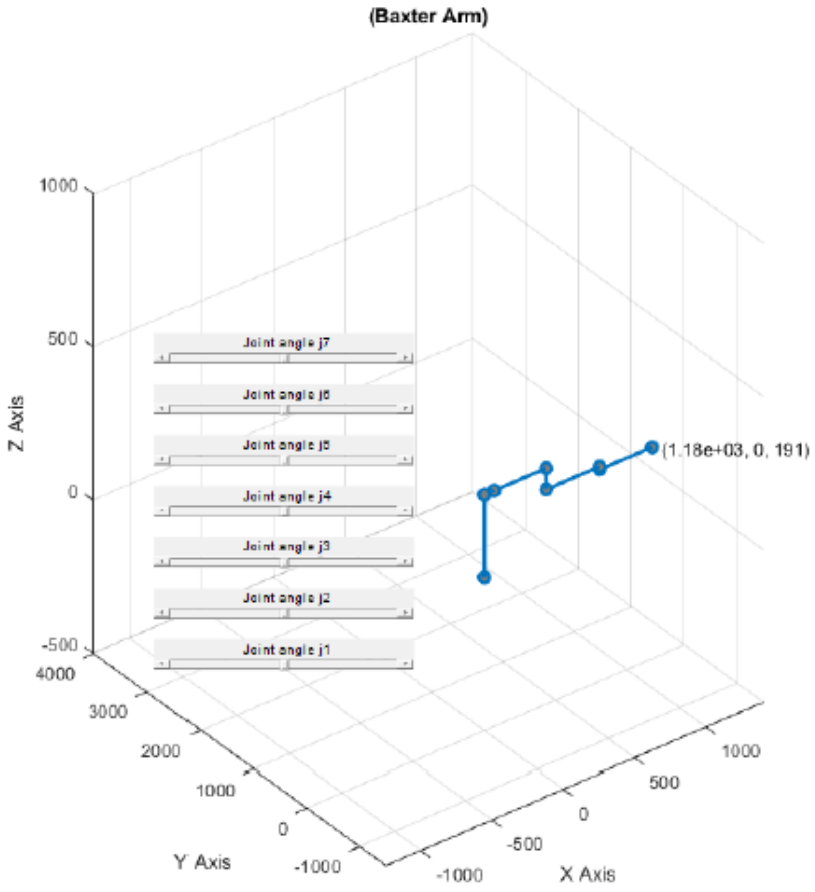
```
function [Mdh] = DHmatrix(theta,d,a,alpha)
%DHMATRIX Summary of this function goes here
%   input DH parameter
Mdh=[cosd(theta) -sind(theta)*cosd(alpha) sind(theta)*sind(alpha)
 a*cosd(theta);
     sind(theta) cosd(theta)*cosd(alpha)  -cosd(theta)*sind(alpha)
 a*sind(theta);
     0,sind(alpha),cosd(alpha),d;
     0,0,0,1];
end
```

XYZkine (To compute the Q matrix to plot)

```
function [Q] = XYZkine(FK)
%DRAWKINE Summary of this function goes here

Q1=[0 FK(1,4) FK(1,8) FK(1,12) FK(1,16) FK(1,20) FK(1,24) FK(1,28)
 FK(1,32) FK(1,36) FK(1,40)];
Q2=[0 FK(2,4) FK(2,8) FK(2,12) FK(2,16) FK(2,20) FK(2,24) FK(2,28)
 FK(2,32) FK(2,36) FK(2,40)];
Q3=[0 FK(3,4) FK(3,8) FK(3,12) FK(3,16) FK(3,20) FK(3,24) FK(3,28)
 FK(3,32) FK(3,36) FK(3,40)];
Q=[Q1;Q2;Q3];
end
```

Published with MATLAB® R2019b



Appendix C: Matlab code for 6 DOF Baxter Left arm simulation with custom GUI

```
function [] = slider_plot()
% Plot different plots according to slider location.
% Junk data for comparison of adjusted curves in this example

S.fh = figure('units','normalized',...
    'Position', [0.515 0.025 0.415 0.87],... %%%%
              'name','slider_plot');


% Define inital parameter values for curve
S.a = 0;
S.b = 0;
S.d = 0;
S.e = 0;
S.f = 0;
S.g = 0;

update(S);

% Slider for slope parameter:
S.aSlider = uicontrol('style','slider',...
              'unit','normalized',...
              'position',[0.2 0.3 0.3 0.01],...
              'min',-180,'max',180,'value', S.a,...
              'sliderstep',[0.003 0.03],...
              'callback', {@SliderCB, 'a'});
% Add a text uicontrol to label the slider.
txta = uicontrol('Style','text',...
     'unit','normalized',...
              'position',[0.2 0.31 0.3 0.02],...
     'String','Joint angle j1');
% 2nd Slider:
S.bSlider = uicontrol('style','slide',...
              'unit','normalized',...
              'position',[0.2 0.35 0.3 0.01],...
              'min',-180,'max',180,'value', S.b,...
              'sliderstep',[0.003 0.03],...
              'callback', {@SliderCB, 'b'});
% Add a text uicontrol to label the slider.
txtb = uicontrol('Style','text',...
     'unit','normalized',...
              'position',[0.2 0.36 0.3 0.02],...
     'String','Joint angle j2');
```

```
% 4th Slider:
S.dSlider = uicontrol('style','slide',...
                'unit','normalized',...
                'position',[0.2 0.4 0.3 0.01],...
                'min',-180,'max',180,'value', S.d,...
                'sliderstep',[0.003 0.03],...
                'callback', {@SliderCB, 'd'});
% Add a text uicontrol to label the slider.
txtd = uicontrol('Style','text',...
     'unit','normalized',...
                'position',[0.2 0.41 0.3 0.02],...
     'String','Joint angle j4');
 % 5th Slider:
S.eSlider = uicontrol('style','slide',...
                'unit','normalized',...
                'position',[0.2 0.45 0.3 0.01],...
                'min',-180,'max',180,'value', S.e,...
                'sliderstep',[0.003 0.03],...
                'callback', {@SliderCB, 'e'});
% Add a text uicontrol to label the slider.
txtd = uicontrol('Style','text',...
     'unit','normalized',...
                'position',[0.2 0.46 0.3 0.02],...
     'String','Joint angle j5');

  % 6th Slider:
S.fSlider = uicontrol('style','slide',...
                'unit','normalized',...
                'position',[0.2 0.50 0.3 0.01],...
                'min',-180,'max',180,'value', S.f,...
                'sliderstep',[0.003 0.03],...
                'callback', {@SliderCB, 'f'});
% Add a text uicontrol to label the slider.
txtd = uicontrol('Style','text',...
     'unit','normalized',...
                'position',[0.2 0.51 0.3 0.02],...
     'String','Joint angle j6');
  % 7th Slider:
S.eSlider = uicontrol('style','slide',...
                'unit','normalized',...
                'position',[0.2 0.55 0.3 0.01],...
                'min',-180,'max',180,'value', S.g,...
                'sliderstep',[0.003 0.03],...
                'callback', {@SliderCB, 'g'});
% Add a text uicontrol to label the slider.
txtd = uicontrol('Style','text',...
     'unit','normalized',...
                'position',[0.2 0.56 0.3 0.02],...
     'String','Joint angle j7');


guidata(S.fh, S);  % Store S structure in the figure
end
```

```
% Callback for all sliders defined above
function SliderCB(aSlider, EventData, Param)
S = guidata(aSlider);  % Get S structure from the figure
S.(Param) = get(aSlider, 'Value');  % Any of the 'a', 'b', etc.
 defined
update(S);  % Update the plot values
guidata(aSlider, S);  % Store modified S in figure
end
% Plot update function, creates new y-vector for plot and replaces the
 plot
% S.p2 with new y-vector
function update(S)
h1=FKdraw( S.a,S.b,S.d,S.e,S.f,S.g)
```

Fk Draw Function (Plots the Links in 3D space)

```
function [ FK ] = FKdraw( j1,j2,j4,j5,j6,j7)

%initial parameter
%[j0 j1 j2 j3;d0 d1 d2 d3;a0 a1 a2 a3;t0 t1 t2 t3]

% D-H Parameters
a1 = 69; % length of first arm
a2 = 0; % length of second arm
a3 = 69; % length of third arm
a4 = 0; % length of fourth arm
a5 = 10; % length of fifth arm
a6 = 0; % length of sixth arm
a7 = 0; % length of seventh arm
d1 = 270; % offset of first arm
d2 = 0; % offset of second arm
d3 = 364; % offset of third arm
d4 = 0; % offset of fourth arm
d5 = 375; % offset of fifth arm
d6 = 0; % offset of sixth arm
d7 = 368; % offset of seventh arm
Lh=370.48
j=[j1 j2 j4+90 j5 j6 j7;d1 0 0 d5 0 d7;a1 Lh 0 a5 0 0;-90 0 90 -90 90
 0];

FK=DHkine(j);
Q=XYZkine(FK);

h=plot3(Q(1,:),Q(2,:),Q(3,:),'-
o','LineWidth',2,'MarkerSize',6,'MarkerFaceColor',[0.5,0.5,0.5]);



grid on;
text(Q(1,7),Q(2,7),Q(3,7),['  (', num2str(Q(1,7),3), ', ',
 num2str(Q(2,7),3),', ', num2str(Q(3,7),3), ')']);
title('(Baxter Arm)')
xlabel('X Axis');
ylabel('Y Axis');
zlabel('Z Axis');
axis([-1250 1250 -1500 4000 -750 1000]);
h = rotate3d;
h.Enable = 'on';
h.ActionPostCallback = @mypostcallback;
assignin('base','FK',FK);

end

function mypostcallback(obj,evd)
%disp('A rotation is about to occur.');
ax_properties = get(gca);
```

```
assignin('base','pov',ax_properties.View);
end

%use evalin('base',a) to get variable a from workspace
%assignin('base','a_rms',a_rms) to write variable a_rms to workspace
```

DHkine Function (Computes the Transformation Matrices)

```
function [ FK ] = DHkine(j)
%collum 1=joint angle, 2=joint offset, 3=link lenght, 4=twist angle
T01=DHmatrix(j(1,1),j(2,1),j(3,1),j(4,1));
T12=DHmatrix(j(1,2),j(2,2),j(3,2),j(4,2));
T24=DHmatrix(j(1,3),j(2,3),j(3,3),j(4,3));
T45=DHmatrix(j(1,4),j(2,4),j(3,4),j(4,4));
T56=DHmatrix(j(1,5),j(2,5),j(3,5),j(4,5));
T67=DHmatrix(j(1,6),j(2,6),j(3,6),j(4,6));


T02=T01*T12;
T04=T02*T24;
T05=T04*T45;
T06=T05*T56;
T07=T06*T67

FK=[T01 T02 T04 T05 T06 T07];
end
```

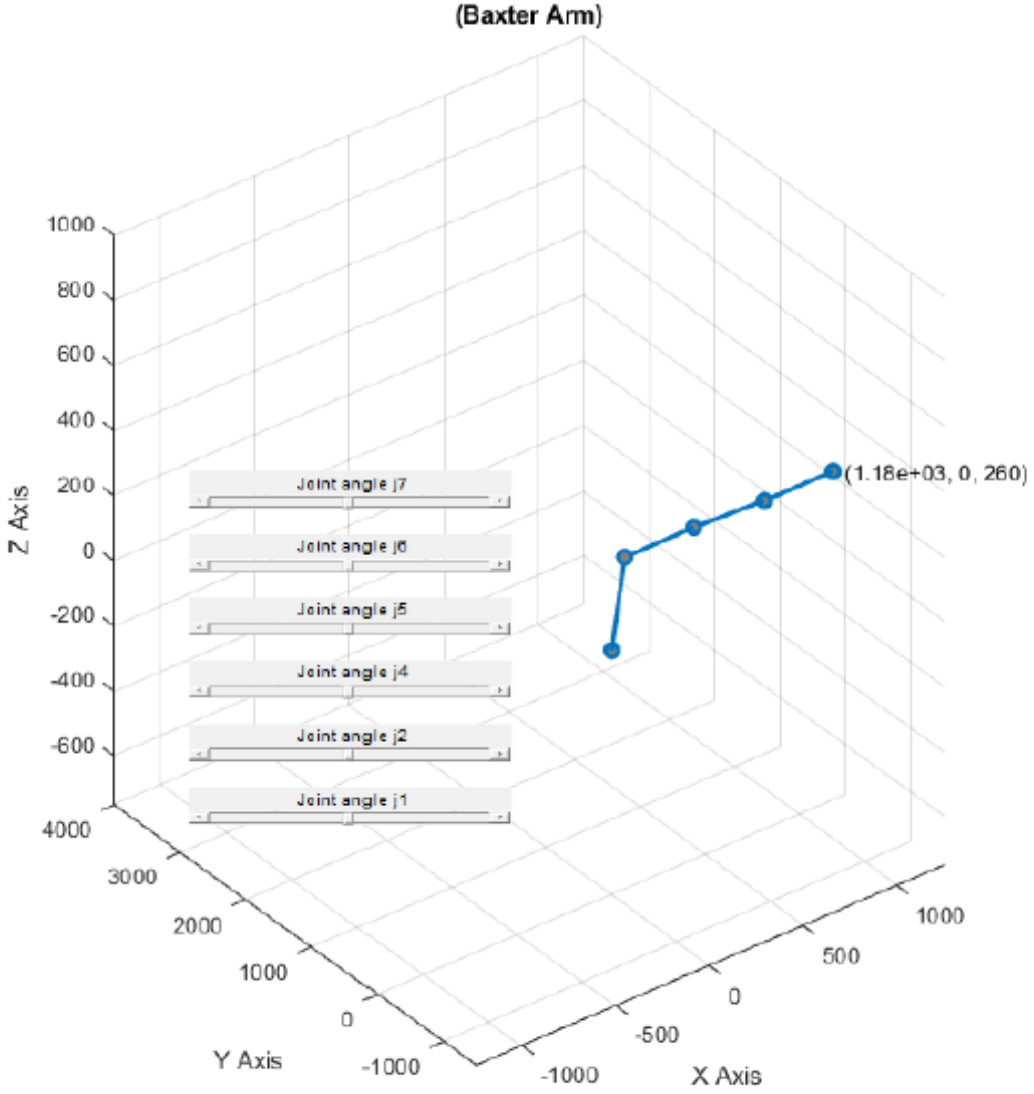
DH matrix Function (Contains the General DH Matrix)

```
function [Mdh] = DHmatrix(theta,d,a,alpha)
%DHMATRIX Summary of this function goes here
%   Detailed explanation goes here
%   inputannya DH parameter yaa
Mdh=[cosd(theta) -sind(theta)*cosd(alpha) sind(theta)*sind(alpha)
 a*cosd(theta);
     sind(theta) cosd(theta)*cosd(alpha)  -cosd(theta)*sind(alpha)
 a*sind(theta);
     0,sind(alpha),cosd(alpha),d;
     0.0.0.1];
```

XYZkine (To compute the Q matrix to plot)

```
function [Q] = XYZkine(FK)
%DRAWKINE Summary of this function goes here
%   Detailed explanation goes here
%P=[FK(1:3,4) FK(1:3,8) FK(1:3,12) FK(1:3,16)];

Q1=[0 FK(1,4) FK(1,8) FK(1,12) FK(1,16) FK(1,20) FK(1,24)];
Q2=[0 FK(2,4) FK(2,8) FK(2,12) FK(2,16) FK(2,20) FK(2,24)];
Q3=[0 FK(3,4) FK(3,8) FK(3,12) FK(3,16) FK(3,20) FK(3,24)];
Q=[Q1;Q2;Q3];
end
```


(Baxter Arm)
Z Axis
1000
800
600
400
200
0
-200
-400
-600
Joint angle j7
Joint angle j6
Joint angle j5
Joint angle j4
Joint angle j2
Joint angle j1
(1.18e+03, 0, 260)
4000
3000
2000
1000
0
-1000
Y Axis
-1000
-500
0
500
1000
X Axis