ME446 Lab 1: Kinematic Transformations

Kevin Gim and Jasvir Virdi

February 22, 2019

Abstract

In this report, we present forward and inverse kinematic analysis of a CRS robot arm. The forward kinematic analysis has been performed by first, assigning coordinate frames to each link using the Denavit-Hartenberg (DH) convention, and then computing the end effector position given the input motor angles. The inverse kinematic analysis deals with computing the inverse problem, i.e, calculation of motor angles given the end effector position. We then present equations for the Jacobian, which are computed using Robotica and MATLAB. Finally, we present some of the applications where these derived equations could be used.

1 Introduction

The problem of kinematics can be described as finding motion of a kinematic chain conforming to geometric constrains. Specifically, forward kinematics is a problem of finding the end effector position given the joint angles of a serial link. The other problem which is of specific interest is the inverse kinematics problem, which deals with finding the joint angles given the end effector

Figure 1: CAD Drawing of CRS Robot Arm

position. This analysis is very essential to robotics, as the kinematic analysis must be verified to move the robot to a desired position or velocity. In the first lab session of ME 446 "Robotic Dynamics and Controls" course, we conducted forward and inverse kinematic analysis on a real 3-axis robot manipulator.

The manipulator used for the lab is "Catalyst-5 Robot Arm" manufactured by Thermo Electron corporation. Originally it comprises of 5 Degrees of Freedom, however the last two motors at the wrist have been deactivated to restrict the motion to 3-DOF. Figure 1 presents a CAD model of the CRS robot arm. The CAD model is generated from the 2D drawings provided in the lab manual [1], and because of that it has almost identical dimension with the actual robot arm. The robot arm consists of three serial links, one base, and 3 revolute joints. All the link lengths are $254mm$.

In this report, we conduct forward and inverse kinematics of the CRS robot arm. In Chapter 2, the DH parameter table is defined and homogeneous transformation matrices are obtained to express the end effector position. Chapter 3 demonstrates the procedure to solve the inverse kinematic problem in a geometric way. We have validated the result of forward and inverse kinematics using Robotica, MATLAB and Hardware implementation. The verification results are shown in Chapter 4.

2 Forward Kinematics

In this section, we present the result of the forward kinematic analysis on the CRS Robot arm. By solving the forward kinematics problem, the orientation and position of end effector of the robot arm can be derived from the joint angles. In order to express the relationship between the links of the kinematic chain, Denavit-Hartenberg(DH) convention has been used. DH parameters are the standard methodology for defining reference frames of a spatial kinematic chain and are used to find the homogeneous transformation relation among the frames.

2.1 DH Parameter

The DH parameter is the common convention to express the geometric constraint of a spatial kinematic chain. The first step of DH parameter is assigning reference frames to the spatial kinematic chain that represents the robot arm. The reference frames are assigned with the following rules:

1. The z_i axis is the direction of $(i+1)^{th}$ joint axis.
2. The x_i axis is parallel to the common normal of z_i and z_{i-1}.
3. The origin of ith frame is located at the intersection of the i^{th} link and the $(i+1)^{th}$ link.
4. The y_i axis is determined by following a right-handed coordinate rule with x_i and z_i axis.

In the DH parameters, the homogeneous transformation matrix between $(i-1)^{th}$ link and i^{th} link, T_i^{i-1} is represented as following [2]:

$$T_i^{i-1} = Rot_{z,\theta} \cdot Trans_{z,d} \cdot Trans_{x,a} \cdot Rot_{x,\alpha} \tag{1}$$

$$= \begin{bmatrix} c\theta & -s\theta c\alpha & s\theta s\alpha & ac\theta \\ s\theta & c\theta c\alpha & -c\theta s\alpha & as\theta \\ 0 & s\alpha & c\alpha & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

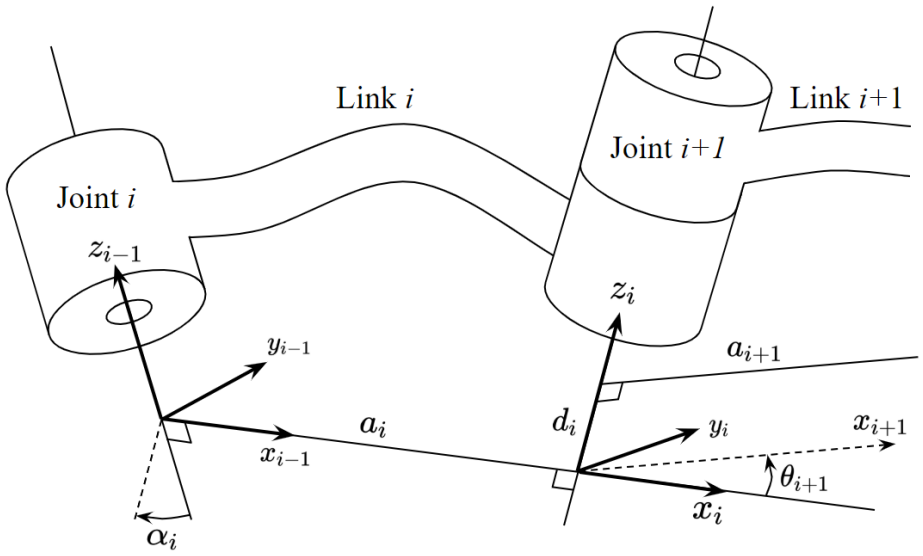


Figure 2: Denavit-Hatenberg frame assignment

Where c and s are the abbreviation of *cos* and *sin* and the four quantities θ_i, d_i, a_i, α_i are link length, link twist, link offset, and joint angle respectively. Specifically, θ_i is the angle difference between x_i and x_{i+1} along z_i, d_i is the distance between x_i and x_{i+1} along z_i, a_i is the distance between z_{i-1} and z_i along x_{i-1}, and α_i is the angle difference between z_{i-1} and z_i along x_{i-1}. In the case of a revolute joint, θ_i become the variable that indicates the joint angle position of the joint, while the other parameters are constant value that are determined by the geometric constraint of the kinematic chain. Figure 2 shows the frame location assignment and parameter identification for the DH parameter.

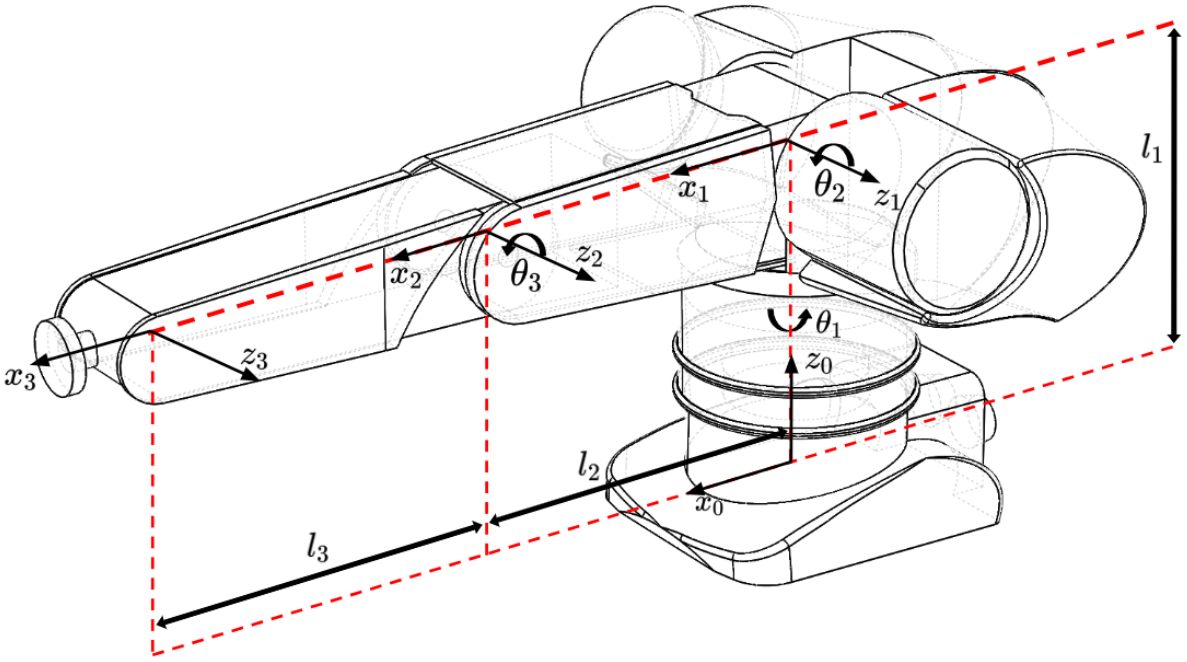


Figure 3: DH Coordinate Frame of the CRS Robot Arm

The assigned DH coordinate frame are presented in Figure 3. All the DH parameter joint angles of the robot arm in this configuration are considered to be zero. This is done in order to simplify the DH parameters although the actual motor angles of the manipulator in this configuration won't be zero. Note that the link length l_1, l_2, and l_3 are each $254mm$. In order to find the DH parameters of the robot arm, z axes of each frame needs to be defined first. The direction of each z_i axis is assigned to coincide with $(i+1)^{th}$ joint axis. For instance, z_0 is pointing along the axis of the first joint at the base. Next, z_1 is assigned to coincide with the axis of second joint. Similarly, z_2 lies along with the joint 3 axis and z_3 is attached to the end of the third link while pointing in the same direction as z_2. Note that the direction of each axis at the joint is assigned to be counter-clockwise by considering the positive torque direction of each motor.

Now, x axes can be derived using the relationship between z axes of the frames. x_{i-1} is assigned to point along the common normal of z_{i-1} and z_i intersecting z_{i-1} axis. In general, it is recommended to follow right hand rule to determine the direction of x axis, however we decided to set x_0 to point in the opposite direction of the outcome of right hand rule. This helps in getting rid of the angle offset between following x axis, hence simplifying calculations. Since z_1 and z_2 are parallel, x_1 is assigned to point along z_3 . Likewise, the direction of x_2 and x_3 are defined as shown in the figure. As x and z axes are determined, the location of the frame can be obtained. The origin of frame 0, the base frame is attached to the point where the rotation axis of the first joint meets the ground base. Then the origin of frame 1 is attached to the point where the joint axis z_0 and z_1 intersect. At last, the origin of frame 2 and 3 lie at the location where z_i axis coincides with x_{i-1} axis.

i	a_i	α_i	d_i	θ_i
1	0	$-\pi/2$	l_1	θ_{1DH}
2	l_2	0	0	θ_{2DH}
3	l_3	0	0	θ_{3DH}

Table 1: DH Parameter table of the CRS Robot arm

After defining the coordinate frames, θ_i, d_i, a_i, α_i needs to be defined to complete the DH parameter table. As a_i is the distance between z_{i-1} and z_i along x_{i-1}, a_i can be easily obtained as a_1 becomes 0 and a_2, a_3 are l_2 and l_3 respectively. Similarly, as d_i is the offset between x_{i-1} and x_i along z_i, only d_1 has non-zero value of l_1. α_i is the angle offset between z_{i-1} and z_i along x_{i-1} axis and since z_1, z_2, and z_3 are parallel, a_2 and a_3 are all 0 and only a_1 is $-\pi/2$. Lastly, since all the x_i are pointing in the same direction, θ_i becomes θ_{iDH} where θ_i denotes the angular position of the i^{th} joint. The DH parameter table is shown in Table 1. Note that θ_i has subscribe DH to distinguish the DH parameter joint angle from the actual motor encoder readings. As shown in figure 4, by applying equation 2, the homogeneous transformation matrix of each link can be obtained as follows:

$$T_1^0 = \begin{bmatrix} c\theta_{1_{DH}} & 0 & s\theta_{1_{DH}} & 0 \\ s\theta_{1_{DH}} & 0 & -c\theta_{1_{DH}} & 0 \\ 0 & -1 & 0 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2^1 = \begin{bmatrix} c\theta_{2_{DH}} & -s\theta_{2_{DH}} & 0 & l_2\ c\theta_{2_{DH}} \\ s\theta_{2_{DH}} & c\theta_{2_{DH}} & 0 & l_2\ s\theta_{2_{DH}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3^2 = \begin{bmatrix} c\theta_{3_{DH}} & -s\theta_{3_{DH}} & 0 & l_3\ c\theta_{3_{DH}} \\ s\theta_{3_{DH}} & c\theta_{3_{DH}} & 0 & l_3\ s\theta_{3_{DH}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Lastly, by multiplying the three transformation matrices, the complete transformation matrix for the forward kinematics is obtained as follows:

$$T_3^0 = \begin{bmatrix} c_1c_{23} & -c_1s_{23} & -s_1 & c_1(l_2c_2 + l_3\ c_{23}) \\ s_1c_{23} & -s_1s_{23} & c_1 & s_1(l_2c_2 + l_3\ c_{23}) \\ -s_{23} & -c_{23} & 0 & l_1 - l_3s_{23} - l_2s_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Where c_{ij} and s_{ij} are abbreviation of $cos(i+j)$ and $sin(i+j)$ respectively.

Figure 4: Three Configuration for finding DH-Motor Angle Relationship

Next, we find the relation between the DH motor angles and the actual motor angles which are measured by encoders. The DH parameter joint angles θ_{iDH} needs to be converted to the motor angles in order to apply the derived transformation matrix for manipulating the robot arm. Since there are three variables for motor angle θ_{1M}, θ_{1M}, and θ_{3M}, it is required to have three different configurations to find the relation between the DH parameter angle and the actual motor angle. Moreover, due to the chain-driven mechanism of the robot arm, we need to define the effect of θ_{2M} to θ_{3M}. We measured the motor position angles of the three configurations shown in Figure 4 using encoders and derived the following equations:

$$\theta_{1_{DH}} = \theta_{1_M} \tag{4}$$

$$\theta_{2_{DH}} = \theta_{2_M} - \frac{\pi}{2} \tag{5}$$

$$\theta_{3_{DH}} = \theta_{3_M} - \theta_{2_M} + \frac{\pi}{2} \tag{6}$$

It is obvious that $\theta_{1_{DH}}$ and θ_{1_M} are identical. Meanwhile, $\theta_{2_{DH}}$ and θ_{2_M} have an offset of $-\frac{\pi}{2}$. Equation 5 shows that $\theta_{3_{DH}}$ is combination of second motor and third motor which movements are oppose to each other.

i	a_i	α_i	d_i	θ_i
1	0	$-\pi/2$	l_1	θ_{1_M}
2	l_2	0	0	$\theta_{2_M} - \pi/2$
3	l_3	0	0	$\theta_{3_M} - \theta_{2_M} + \pi/2$

Table 2: DH Parameter table of the CRS Robot arm with Motor angle

Finally, Table 2 shows the final representation of the DH parameter table.

3 Inverse Kinematics

In this section, we analyze the problem of finding the motor input angles from a given end effector position. In the following diagram, the position of the end effector is marked as (x, y, z) and joint angle relations are computed based on geometry. Note that the angles mentioned on the figures are all DH angles.

After assigning the coordinate frames, we calculate the inverse kinematic relations (i.e the DH motor angles) using geometry. Figure 3 is used as a reference to derive these relations. In order to

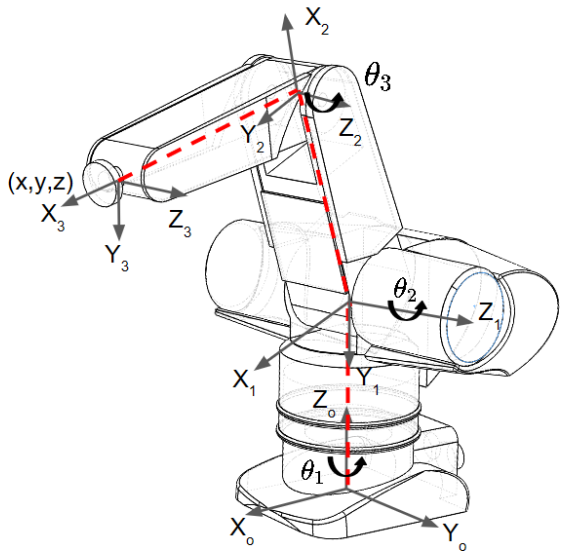


Figure 5: Coordinate Frame assignment for each link

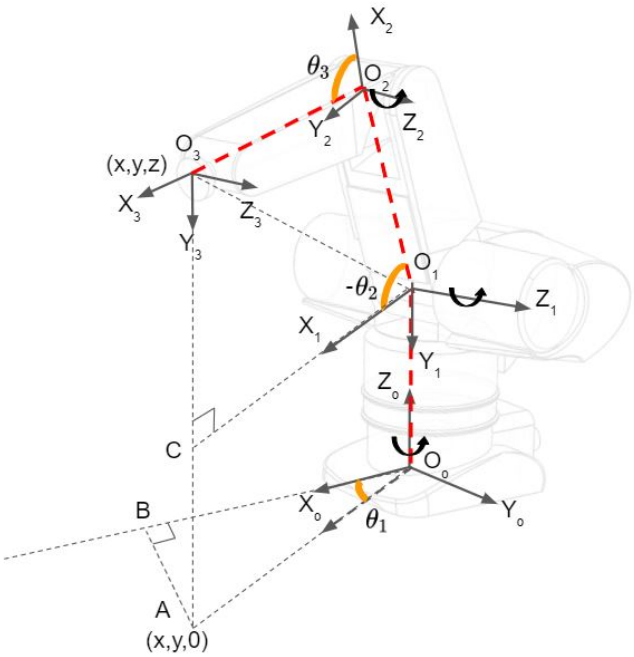


Figure 6: Motor angle relations derived using Geometry

derive θ_1, we will first focus on $\Delta O_o AB$. Using simple geometry we can easily obtain the following relation:

$$\theta_1 = atan2\left[\frac{y}{x}\right] \tag{7}$$

Next, we derive θ_3. We first calculate distance $\overline{O_1O_3}$ which can be computed by using Pythagoras theorem in ΔO_1O_3C.

$$\overline{O_1O_3}^2 = \overline{O_3C}^2 + \overline{O_1C}^2 \tag{8}$$

Here $\overline{O_3C}$ is given simply as $(z - \overline{O_1O_5})$ where $\overline{O_1O_5}$ indicates length of link 1. Also, since $\overline{O_1C} = \overline{O_1A}$, we can be compute $\overline{O_1A}$ by applying Pythagoras theorem to $\Delta O_o AB$.

$$\overline{O_oA}^2 = x^2 + y^2 = \overline{O_1C}^2 \tag{9}$$

Hence rewriting the relation for $\overline{O_1O_3}$ gives us:

$$\overline{O_1O_3}^2 = x^2 + y^2 + (z - \overline{O_1O_5})^2 \tag{10}$$

From the above relation, we can easily compute O_1O_3 given the end effector position. We now compute θ_3 by applying cosine rule to $\Delta O_1O_2O_3$. Since the link lengths are already known, we already have lengths of O_1O_2 & O_2O_3. Hence θ_3 is given by:

$$\theta_3 = \pi - acos\left(\frac{\overline{O_1O_2}^2 + \overline{O_2O_3}^2 - \overline{O_1O_3}^2}{2\overline{O_1O_2} \cdot \overline{O_2O_3}}\right) \tag{11}$$

Lastly, θ_2 is derived by adding $\angle O_3O_1O_2$ and $\angle O_3O_1C$. However, an important thing to note is that the sign of θ_2 will be negative according to sign convention. Hence θ_2 is given by

$$\theta_2 = -(\angle O_3O_1O_2 + \angle O_3O_1C) \tag{12}$$

where $\angle O_3O_1O_2$ and $\angle O_3O_1C$ is given as follows

$$\angle O_3O_1O_2 = \frac{\theta_3}{2} \tag{13}$$

$$\angle O_3O_1C = asin\Big(\frac{z - \overline{O_1O_5}}{\overline{O_1O_3}}\Big) \tag{14}$$

4 Result

This section demonstrates the results for validating both the forward and inverse kinematics through software and hardware implementation. In order to verify the forward kinematics result, the DH table derived earlier is plugged into Robotica and a formula for end effector position is computed with given motor angles. This computed formula is used in our main C program to print the end effector positions which are then compared with the actual end effector position. The actual end effector positions were measured using scale. The inverse kinematics part was verified by comparing the computed motor angles, given an end effector position, with the encoder values.

4.1 Robotica

Robotica is a useful tool package for solving robotics problems in Mathematica. Mathematica has numerous advantages when using symbolic expressions to handle complex calculations. Robotica can produce the forward kinematic solution by taking the DH parameter table as an input. We have validated the result of Section 2 by comparing it with the output from Robotica.

Joint	Type	r	α	d	θ
1	revolute	0	$-\frac{\pi}{2}$	254	θ_1^*
2	revolute	254	0	0	θ_2^*
3	revolute	254	0	0	θ_3^*

Figure 7: The DH parameters Table Input to Robotica

Figure 7 shows the input table to Robotica and Figure 8 presents the output of executing Robotica, the transformation matrices. Since l_1 and l_2 is $254mm$, The forward kinematics solution in equation 3 is identical with $T(0,3)$ as seen in Figure 8. In addition, the Jacobian matrix of the robot arm computed by Robotica is shown in Figure 9.

In[85]:= TPrint[] (*Run to show all the T matrices*)

$$T[0,1] = \begin{pmatrix} c_1 & 0 & -s_1 & 0 \\ s_1 & 0 & c_1 & 0 \\ 0 & -1 & 0 & 254 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T[0,2] = \begin{pmatrix} c_1 c_2 & -c_1 s_2 & -s_1 & 254\, c_1 c_2 \\ c_2 s_1 & -s_1 s_2 & c_1 & 254\, c_2 s_1 \\ -s_2 & -c_2 & 0 & 508 \sin\left[\frac{\pi}{4} - \frac{q2}{2}\right]^2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T[0,3] = \begin{pmatrix} c_1 c_{23} & -c_1 s_{23} & -s_1 & 508\, c_1 \cos\left[q2 + \frac{q3}{2}\right] \cos\left[\frac{q3}{2}\right] \\ c_{23} s_1 & -s_1 s_{23} & c_1 & 508 \cos\left[q2 + \frac{q3}{2}\right] \cos\left[\frac{q3}{2}\right] s_1 \\ -s_{23} & -c_{23} & 0 & -254\,(-1 + s_2 + s_{23}) \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T[1,2] = \begin{pmatrix} c_2 & -s_2 & 0 & 254\, c_2 \\ s_2 & c_2 & 0 & 254\, s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T[1,3] = \begin{pmatrix} c_{23} & -s_{23} & 0 & 508 \cos\left[q2 + \frac{q3}{2}\right] \cos\left[\frac{q3}{2}\right] \\ s_{23} & c_{23} & 0 & 508 \cos\left[\frac{q3}{2}\right] \sin\left[q2 + \frac{q3}{2}\right] \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T[2,3] = \begin{pmatrix} c_3 & -s_3 & 0 & 254\, c_3 \\ s_3 & c_3 & 0 & 254\, s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure 8: The Transformation Matrix Computed by Robotica

In[86]:= Mprint[J, "Jacobian="]

Out[86]= Mprint[{{-508 Cos[q2 + q3/2] Cos[q3/2] s1, c1 (-254 - 254 (-1 + s2 + s23)), c1 (-508 Sin[π/4 - q2/2]^2 - 254 (-1 + s2 + s23))},
{508 c1 Cos[q2 + q3/2] Cos[q3/2], s1 (-254 - 254 (-1 + s2 + s23)), s1 (-508 Sin[π/4 - q2/2]^2 - 254 (-1 + s2 + s23))},
{0, -508 c1^2 Cos[q2 + q3/2] Cos[q3/2] - 508 Cos[q2 + q3/2] Cos[q3/2] s1^2, -c1 (-254 c1 c2 + 508 c1 Cos[q2 + q3/2] Cos[q3/2]) -
s1 (-254 c2 s1 + 508 Cos[q2 + q3/2] Cos[q3/2] s1)}, {0, -s1, -s1}, {0, c1, c1}, {1, 0, 0}}, Jacobian=]

Figure 9: The Jacobian Matrix Computed by Robotica

4.2 MATLAB

Additional verification has been conducted by using MATLAB's Robotics Toolbox. Robotics Toolbox [3] made by Peter Corke provides useful functions to deal with kinematics and dynamics problems related to robotics. Similar to Robotica, Robotic Toolbox also takes DH parameters as input to define geometric constraint of each link. Figure 10 shows the result of calculating the transformation matrix with MATLAB. In addition, Figure 11 presents the jacobian matrix as well. The MATLAB code and more detailed results are presented in Appendix.

```
Robot =

noname:: 3 axis, RRR, stdDH, fastRNE, Symbolic
+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|         l1|          0|      -pi/2|          0|
|  2|         q2|          0|         l2|          0|          0|
|  3|         q3|          0|         l3|          0|          0|
+---+-----------+-----------+-----------+-----------+-----------+


[ cos(q2 + q3)*cos(q1), -sin(q2 + q3)*cos(q1), -sin(q1), cos(q1)*(l3*cos(q2 + q3) + l2*cos(q2))]
[ cos(q2 + q3)*sin(q1), -sin(q2 + q3)*sin(q1),  cos(q1), sin(q1)*(l3*cos(q2 + q3) + l2*cos(q2))]
[        -sin(q2 + q3),         -cos(q2 + q3),        0,      l1 - l3*sin(q2 + q3) - l2*sin(q2)]
[                    0,                     0,        0,                                      1]
```

Figure 10: The transformation matrix computed by MATLAB script

```
J =

[                                                -sin(q1)*(l3*cos(q2 + q3) + l2*cos(q2)), l2*cos(q2 + q3)*cos(q1)*sin(q3) - sin(q2 + q3)*cos(q1)*(l3 + l2*cos(q3)), -l3*sin(q2 + q3)*cos(q1)]
[                                                cos(q1)*(l3*cos(q2 + q3) + l2*cos(q2)), l2*cos(q2 + q3)*sin(q1)*sin(q3) - sin(q2 + q3)*sin(q1)*(l3 + l2*cos(q3)), -l3*sin(q2 + q3)*sin(q1)]
[                                                                                    0,                 - cos(q2 + q3)*(l3 + l2*cos(q3)) - l2*sin(q2 + q3)*sin(q3),         -l3*cos(q2 + q3)]
[ cos(conj(q2) + conj(q3))*sin(q2 + q3)*cos(q1) - sin(conj(q2) + conj(q3))*cos(q2 + q3)*cos(q1),                                                                -sin(q1),                -sin(q1)]
[ cos(conj(q2) + conj(q3))*sin(q2 + q3)*sin(q1) - sin(conj(q2) + conj(q3))*cos(q2 + q3)*sin(q1),                                                                 cos(q1),                 cos(q1)]
[               cos(conj(q2) + conj(q3))*cos(q2 + q3) + sin(conj(q2) + conj(q3))*sin(q2 + q3),                                                                       0,                       0]
```

Figure 11: The Jacobian matrix computed by MATLAB script

4.3 Hardware Validation

The forward and inverse kinematics results are also validated using the robot arm hardware. The end effector position is computed in C code using the equation we derived in the forward kinematics analysis. By comparing the computed values with actual measurement, the solution can be verified. The C code used for hardware implementation and serial monitor results are attached to Appendix.

Figure 12 shows one configuration where we verified our results for the forward kinematics part. In this case, all the DH joint angles are 0, and the robotic arm is straight along the x axis. In this configuration, measuring the end effector positions with a scale is straigtforward, and is about $(508mm, 0mm, 254mm)$. The computed end-effector position is displayed by the serial monitor presented in Figure 12 which shows nearly an exact match for the measured end effector position. Note that the DH joint angles were computed from the motor angles by the relation specified earlier.

Figure 13 shows the second case which we used to confirm the forward kinematics part. In this configuration, the first motor angle is kept at 0, and the other two arms are rotated each by $\frac{\pi}{2}$. The end effector position in this configuration is easy to measure using scale and is about $(254mm, 0mm, 508mm)$. Figure 13 also shows our computed result for the tested configuration. The end effector position again comes out to be an exact match to values we measured using scale.

At last, we conduct the validation for the case where only the first joint is moved to $\frac{\pi}{2}$ as presented in Figure 14. The result in the serial monitor verifies that the calculated position is matches well the actual measured position as well. In conclusion, the hardware validation clearly demonstrates that our forward kinematics solution works well for all three joints.

Next, we validate our inverse kinematics solution. We did this by comparing our calculated motor angles from inverse kinematics equation with the actual motor angles measured by the encoders. Figure 15 shows the results for the home configuration. It is apparent that there is a clear match between the computed angles and measured angles. Another configuration is tested to verify the angles computed by the inverse kinematics solution. Figure 16 demonstrates when the motor angles are set to zero. The angles match fairly, however, there is a slight difference observed for θ_2 and θ_3. The reason for this is primarily due to a specific range in which trigonometric inverses are valid and here, we are exceeding those ranges for θ_2 slightly. Hence, values for θ_2 become marginally incorrect which in turn causes impact on θ_3.

5 Conclusion

From the above results, we can conclude that the forward and inverse kinematic relations are computed correctly and these relations can be further used for other applications like in dynamics, control etc.

References

[1] ME 446 Laboratory 1 Kinematic Transformations http://coecsl.ece.illinois.edu/me446/ME446Lab_1.pdf

[2] Spong, M.W., Hutchinson, S. and Vidyasagar, M., Robot Modeling and Control.

[3] Peter Croke, Robotics Toolbox, http://petercorke.com/wordpress/toolboxes/robotics-toolbox

6 Appendix

Kinematics Analysis Matlab Script

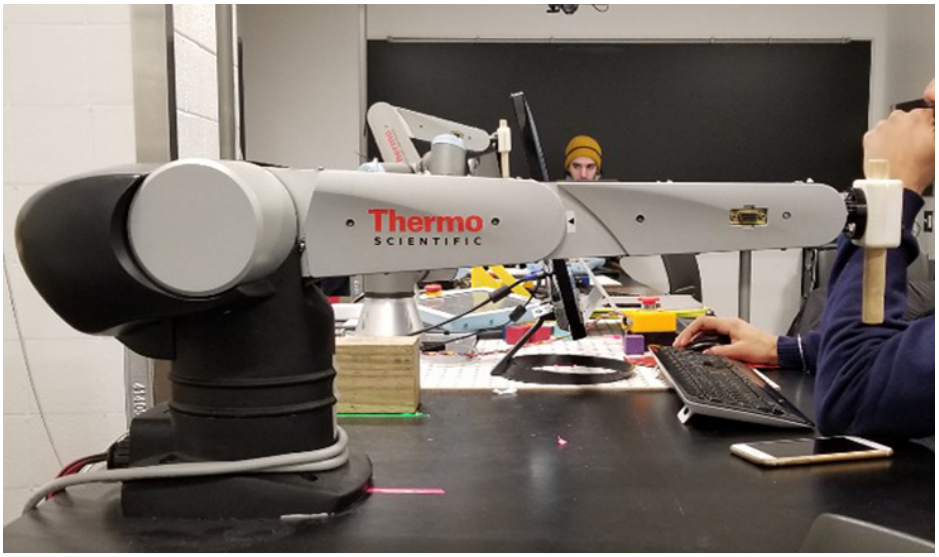
```
startup_rvc

%%

% Define Sympolic letters
syms l1 l2 l3 q1 q2 q3 t1 t2 t3

% Making DH Parameter Table
     % a ahpha d theta
DH = [0, -pi/2, l1, q1;
      l2,  0,   0,  q2;
      l3,  0,   0,  q3;];

 % Generate Link structure to define each link
for i= 1:size(DH,1)
     a = DH(i,1);
    al = DH(i,2);
     d = DH(i,3);
    th = DH(i,4);

    L(i) = Link([th, d, a, al,], 'standard'); % Define link using the DH parameters
                                              % Link structure can also
                                              % contains additional
                                              % information about link such
                                              % as inertia, mass, geometry,
                                              % which is useful features to
                                              % calculate robot dynamics.

end

% Define Robot by using SerialLink function. L
Robot = SerialLink(L)

% Transfer Function
T = Robot.fkine([q1; q2; q3])

% Jacobian Matrix
J = Robot.jacob0([q1; q2; q3])

% Symbolic Inverse Kinematics
sol = Robot.ikine_sym(3)
q1_sol = simplify(sol{1})
q2_sol = simplify(sol{2})
q3_sol = simplify(sol{3})
```

Kinematics Analysis C code

```
1
2  // ME446 Robot Dynamics and Controls
3  // Lab 1: Kinematic Transformation
4  // Kevin Gim & Jasvir Virdi
5
6  #include <tistdtypes.h>
7  #include <coecsl.h>
8  #include <user_includes.h>
9  #include <math.h>
10
11 // These two offset angles are adjusted for our robot arm.
12 float offset_Enc2_rad = -0.36;
13 float offset_Enc3_rad = 0.27;
14
15 // Your global varialbes.
16
17 long mycount = 0;
18
19 float whattoprint = 0.0;
20
21 float theta1array[100];
22
23 long arrayindex = 0;
24
25 float printtheta1motor = 0;
26 float printtheta2motor = 0;
27 float printtheta3motor = 0;
28
29 float DHtheta1 = 0;
30 float DHtheta2 = 0;
31 float DHtheta3 = 0;
32
33 float x = 0;
34 float y = 0;
35 float z = 0;
36
37 float IKtheta1DH = 0;
38 float IKtheta2DH = 0;
39 float IKtheta3DH = 0;
40
41 float IKthetam1 = 0;
42 float IKthetam2 = 0;
43 float IKthetam3 = 0;
44
45 float r1 = 0;
46 float r2 = 0;
47
48 float l1 = 254;
49 float l2 = 254;
50 float l3 = 254;
51
52 // Assign these float to the values you would like to plot in Simulink
53 float Simulink_PlotVar1 = 0;
54 float Simulink_PlotVar2 = 0;
55 float Simulink_PlotVar3 = 0;
56 float Simulink_PlotVar4 = 0;
```

```
57
58
59  // This function is called every 1 ms
60  void lab(float theta1motor,float theta2motor,float theta3motor,float *tau1,
        float *tau2,float *tau3, int error) {
61
62
63          *tau1 = 1;
64          *tau2 = 0;
65          *tau3 = 0;
66
67          //Motor torque limitation(Max: 5 Min: -5)
68
69          // save past states
70          if ((mycount%50)==0) {
71
72                  theta1array[arrayindex] = theta1motor;
73
74                  if (arrayindex >= 100) {
75                          arrayindex = 0;
76                  } else {
77                          arrayindex++;
78                  }
79
80          }
81
82          if ((mycount%500)==0) {
83                  if (whattoprint > 0.5) {
84                          serial_printf(&SerialA, I love robotics);
85                  } else {
86                          printtheta1motor = theta1motor;
87                          printtheta2motor = theta2motor;
88                          printtheta3motor = theta3motor;
89
90                          // Converts Motor angle to DH joint angles
91                          DHtheta1 = theta1motor;
92                          DHtheta2 = theta2motor-PI*0.5;
93                          DHtheta3 = theta3motor-theta2motor+PI*0.5;
94                          SWI_post(&SWI_printf); //Using a SWI to fix SPI issue
                               from sending too many floats.
95                  }
96                  GpioDataRegs.GPBTOGGLE.bit.GPIO34 = 1; // Blink LED on Control
                       Card
97                  GpioDataRegs.GPBTOGGLE.bit.GPIO60 = 1; // Blink LED on
                       Emergency Stop Box
98          }
99
100      // Which value to plot with Simulink
101          Simulink_PlotVar1 = theta1motor;
102          Simulink_PlotVar2 = theta2motor;
103          Simulink_PlotVar3 = theta3motor;
104          Simulink_PlotVar4 = 0;
105
106
107          //Forward Kinematics
108
109          x = 508*cos(DHtheta1)*cos(DHtheta2+DHtheta3/2)*cos(DHtheta3/2);
```

```
        y = 508*sin(DHtheta1)*cos(DHtheta2+DHtheta3/2)*cos(DHtheta3/2);
        z = -254*(-1+sin(DHtheta2)+sin(DHtheta2+DHtheta3));


        //Inverse Kinematics

        IKtheta1DH = atan2(y,x);
        r1 = z-l1;
        r2 = sqrt(r1*r1 + x*x + y*y);
        IKtheta3DH = PI - acos((l2*l2+l3*l3-r2*r2)/(2*l2*l3));
        IKtheta2DH = -(IKtheta3DH)/2 - asin((r1/r2));

        IKthetam1= IKtheta1DH;                  //Joint 1 DH Angle
        IKthetam2=IKtheta2DH +(PI/2);           //Joint 2 DH Angle
        IKthetam3=IKtheta3DH + IKtheta2DH;      //Joint 3 DH Angle

        mycount++;
}

// Display computed value through a serial monitor
void printing(void){

    //serial_printf(&SerialA, (%.2f
       %.2f,%.2f),(%.2f,%.2f,%.2f), DHtheta1, DHtheta2, DHtheta3,x,y,z);
    //serial_printf(&SerialA, (%.2f,%.2f,%.2f), IKtheta1DH,IKtheta2DH,IKtheta3DH);
    serial_printf(&SerialA, (%.2f,%.2f,%.2f),(%.2f,%.2f,%.2f),printtheta1motor*180/
       PI,printtheta2motor*180/PI,printtheta3motor*180/PI, IKthetam1*180/PI,
       IKthetam2*180/PI, IKthetam3*180/PI );}
```



COM4 - Tera Term VT

File Edit Setup Control Window Help

```
DH Joint Angle: (0.09 1.34,1.31),EE Position(507.65,0.81,236.26)
DH Joint Angle: (0.09 1.20,1.49),EE Position(507.66,0.81,236.67)
DH Joint Angle: (0.11 0.91,1.78),EE Position(507.68,1.01,237.98)
DH Joint Angle: (0.14 0.88,1.80),EE Position(507.68,1.30,238.10)
DH Joint Angle: (0.14 0.89,1.80),EE Position(507.68,1.30,238.09)
DH Joint Angle: (0.14 0.89,1.80),EE Position(507.68,1.30,238.09)
DH Joint Angle: (0.14 0.88,1.80),EE Position(507.68,1.30,238.10)
DH Joint Angle: (0.14 0.88,1.81),EE Position(507.68,1.30,238.10)
DH Joint Angle: (0.14 0.90,1.79),EE Position(507.68,1.29,238.03)
DH Joint Angle: (0.14 1.09,1.71),EE Position(507.64,1.27,236.69)
DH Joint Angle: (0.14 1.39,1.50),EE Position(507.59,1.27,234.92)
DH Joint Angle: (0.14 1.71,1.29),EE Position(507.53,1.26,233.08)
DH Joint Angle: (0.14 1.71,1.29),EE Position(507.53,1.27,233.08)
DH Joint Angle: (0.14 1.71,1.29),EE Position(507.53,1.27,233.07)
DH Joint Angle: (0.14 1.71,1.29),EE Position(507.53,1.27,233.07)
DH Joint Angle: (0.14 1.71,1.28),EE Position(507.53,1.27,233.06)
DH Joint Angle: (0.14 1.71,1.28),EE Position(507.53,1.27,233.05)
DH Joint Angle: (0.14 1.72,1.27),EE Position(507.53,1.26,233.01)
DH Joint Angle: (0.14 1.78,1.22),EE Position(507.52,1.26,232.75)
DH Joint Angle: (0.14 1.79,1.20),EE Position(507.52,1.26,232.70)
DH Joint Angle: (0.14 1.85,1.15),EE Position(507.51,1.26,232.46)
DH Joint Angle: (0.14 1.94,1.06),EE Position(507.50,1.26,232.05)
DH Joint Angle: (0.14 1.98,1.02),EE Position(507.49,1.26,231.86)
```

Figure 12: Configuration with $\theta_{1_{DH}} = 0$, $\theta_{2_{DH}} = 0$, $\theta_{3_{DH}} = 0$

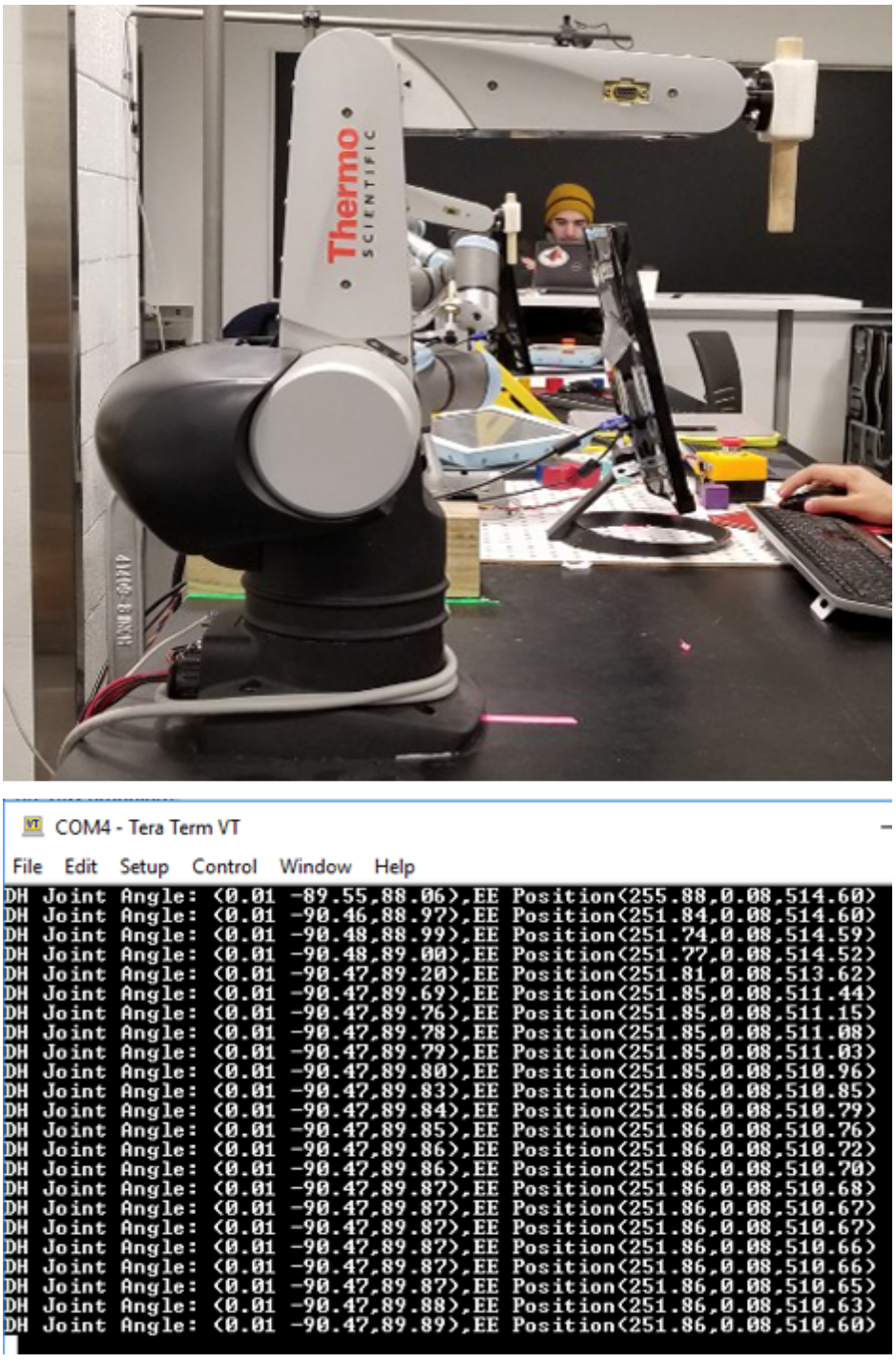
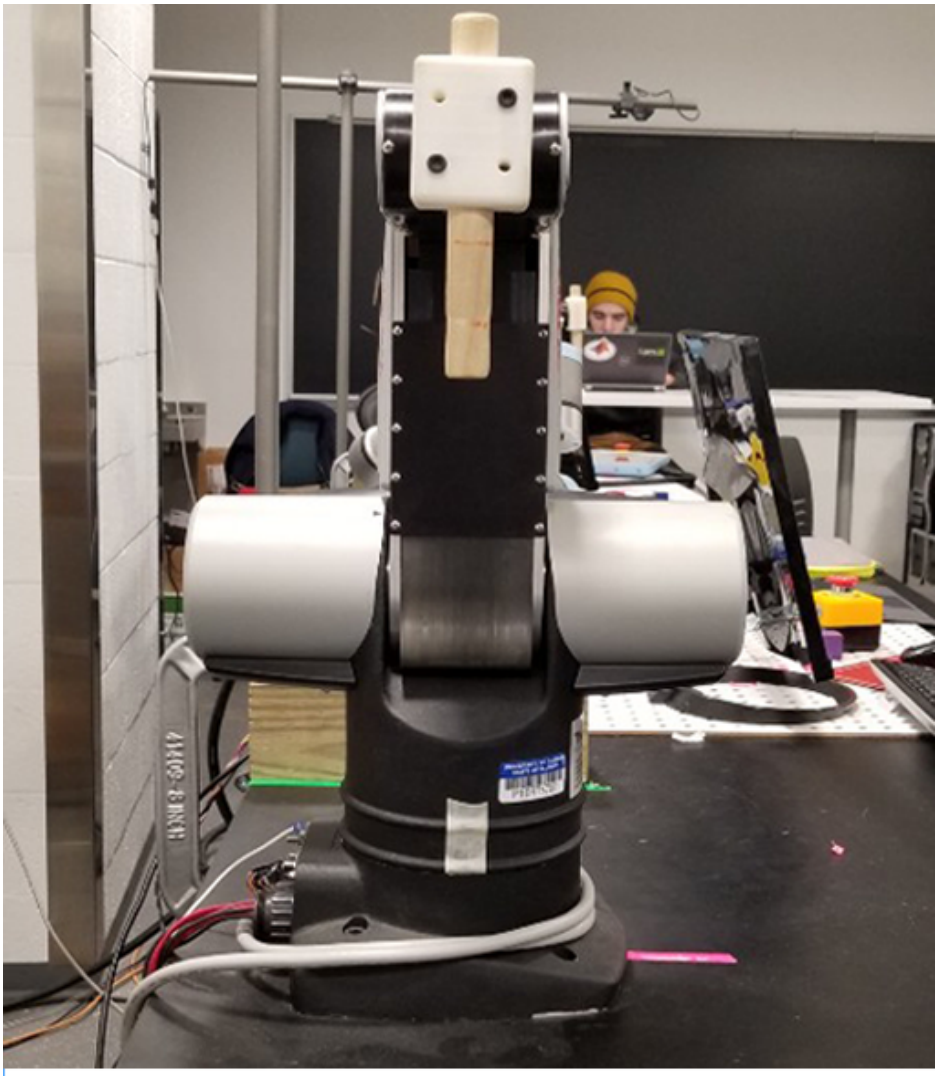


Figure 13: Configuration with $\theta_{1_{DH}} = 0$, $\theta_{2_{DH}} = -\frac{\pi}{2}$, $\theta_{3_{DH}} = \frac{\pi}{2}$



COM4 - Tera Term VT

File Edit Setup Control Window Help

```
DH Joint Angle: (-89.67 -90.33,89.62),EE Position(1.42,-252.47,511.14)
DH Joint Angle: (-89.69 -90.33,89.63),EE Position(1.32,-252.47,511.13)
DH Joint Angle: (-89.69 -90.33,89.63),EE Position(1.32,-252.48,511.11)
DH Joint Angle: (-89.69 -90.33,89.63),EE Position(1.33,-252.48,511.10)
DH Joint Angle: (-89.69 -90.33,89.63),EE Position(1.33,-252.48,511.09)
DH Joint Angle: (-89.69 -90.33,89.64),EE Position(1.33,-252.48,511.08)
DH Joint Angle: (-89.69 -90.33,89.64),EE Position(1.33,-252.48,511.08)
DH Joint Angle: (-89.69 -90.33,89.64),EE Position(1.33,-252.48,511.06)
DH Joint Angle: (-89.69 -90.33,89.65),EE Position(1.33,-252.48,511.03)
DH Joint Angle: (-89.69 -90.33,89.65),EE Position(1.33,-252.48,511.03)
DH Joint Angle: (-89.69 -90.33,89.65),EE Position(1.33,-252.48,511.02)
DH Joint Angle: (-89.69 -90.33,89.65),EE Position(1.33,-252.48,511.02)
DH Joint Angle: (-89.69 -90.33,89.65),EE Position(1.33,-252.48,511.01)
DH Joint Angle: (-89.69 -90.33,89.65),EE Position(1.33,-252.48,511.00)
DH Joint Angle: (-89.69 -90.33,89.66),EE Position(1.33,-252.48,510.98)
DH Joint Angle: (-89.69 -90.33,89.66),EE Position(1.33,-252.48,510.96)
DH Joint Angle: (-89.69 -90.33,89.67),EE Position(1.33,-252.48,510.93)
DH Joint Angle: (-89.69 -90.33,89.68),EE Position(1.33,-252.48,510.91)
DH Joint Angle: (-89.69 -90.33,89.68),EE Position(1.33,-252.48,510.87)
DH Joint Angle: (-89.69 -90.33,89.69),EE Position(1.33,-252.48,510.85)
DH Joint Angle: (-89.69 -90.33,89.69),EE Position(1.33,-252.48,510.84)
DH Joint Angle: (-89.69 -90.33,89.69),EE Position(1.33,-252.48,510.83)
DH Joint Angle: (-89.69 -90.33,89.69),EE Position(1.33,-252.48,510.83)
```

Figure 14: Configuration with $\theta_{1_{DH}} = -\frac{\pi}{2}$, $\theta_{2_{DH}} = -\frac{\pi}{2}$, $\theta_{3_{DH}} = \frac{\pi}{2}$

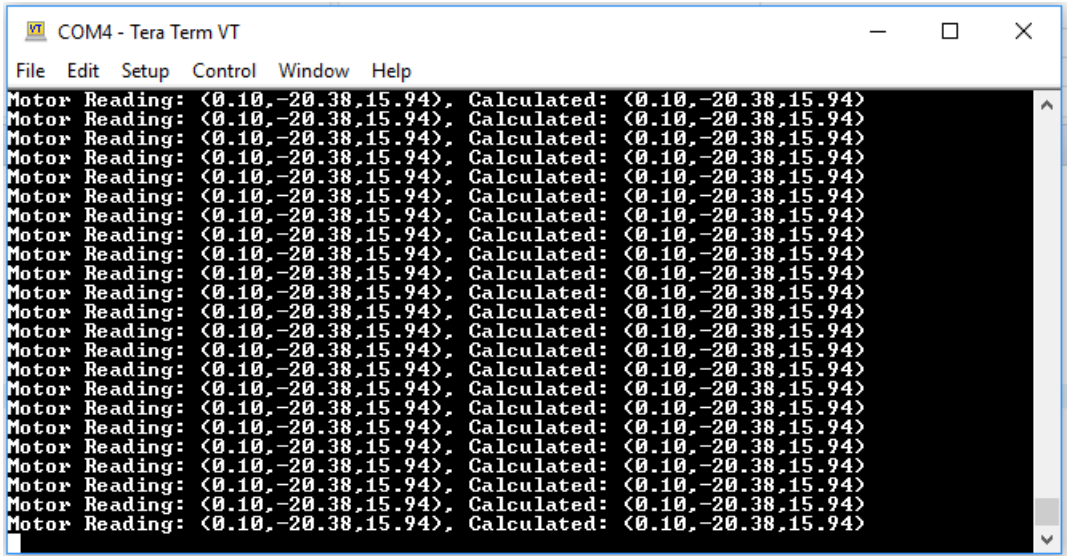


Figure 15: Actual motor angles vs computed motor angles for the first configuration

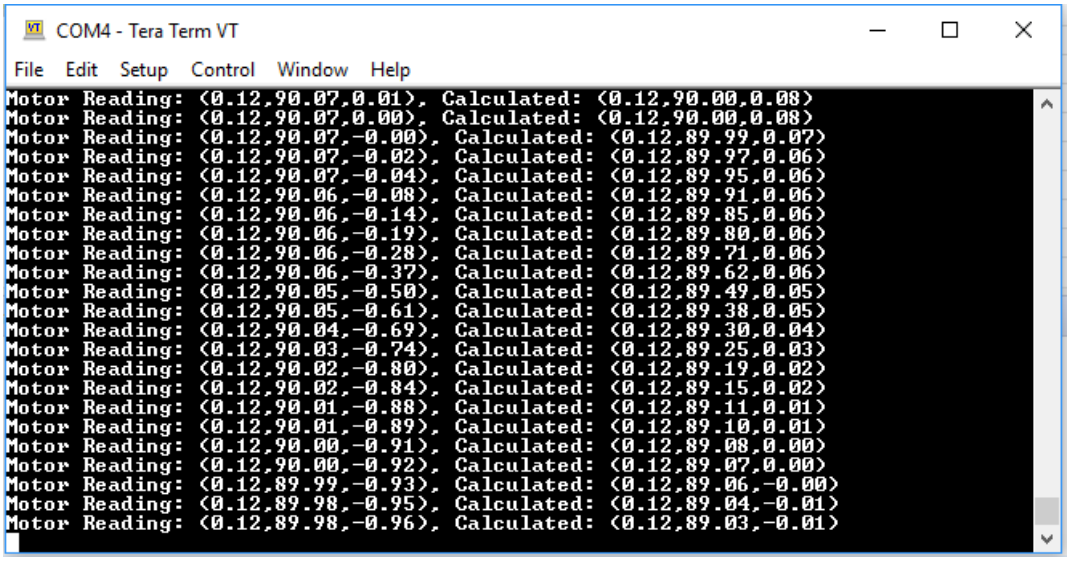


Figure 16: Actual motor angles vs computed motor angles for the second configuration