Robot Dynamics and Control

Lab Report

Kinematic Transformations

Amith Ramdas Achari
amithr3
Peng Han
penghan2
Section: Friday 9PM
Submitted on February 18, 2022

1 Introduction

Forward Kinematics of a robot refers to finding the end effectors position and orientation with respect to the robot based on the joint angles of each joint theta. The end effectors position can be found out using two methods, namely Denavit Hartenberg and Product of Exponentials parameters. In this lab we used the Denavit Hartenberg method to find the forward kinematics of the CRS Robot.

Denavit Hartenberg method mainly comprises of determining the frames of each joint using a particular convention and then finding the Denavit–Hartenberg parameters (also known as DH parameters) which are four parameters that are associated with a specific convention for attaching reference frames to the links of a spatial kinematic chain, or robot manipulator.

Inverse Kinematics of a robot refers to finding the joint angles of each joint based on the end effectors position and orientation with respect to the robot. These joint angles can be found out using two methods, analytical approach, also known as graphical approach and numerical approach. In this lab we used the analytical approach to find the inverse kinematics of the CRS Robot.

The analytical approach or graphical approach basically uses a sequence of equations which can be used to solve for the desired joint angles. This can potentially be faster than the numerical approach and it will always provide the correct joint angles given that the sequence of equations is solvable. Whereas the iterative solution, finds the desired joint angles after iterating through a number of iterations using the Newton-Raphson Method. The advantage of this method is that it is easier to think and gives more freedom for incremental displacements and control.

2 Tasks

- Introduce the TMS320F28335 DSP controller for the CRS robot arm and using Code Composer Studio IDE to program the TMS320F28335 DSP with C.
- Derive the forward kinematic equations for CRS robot arm following the Denavit-Hartenberg (DH) convention
- Derive inverse kinematic equations for the CRS robot arm
- Use the provided Code Composer Studio project to verify your solution

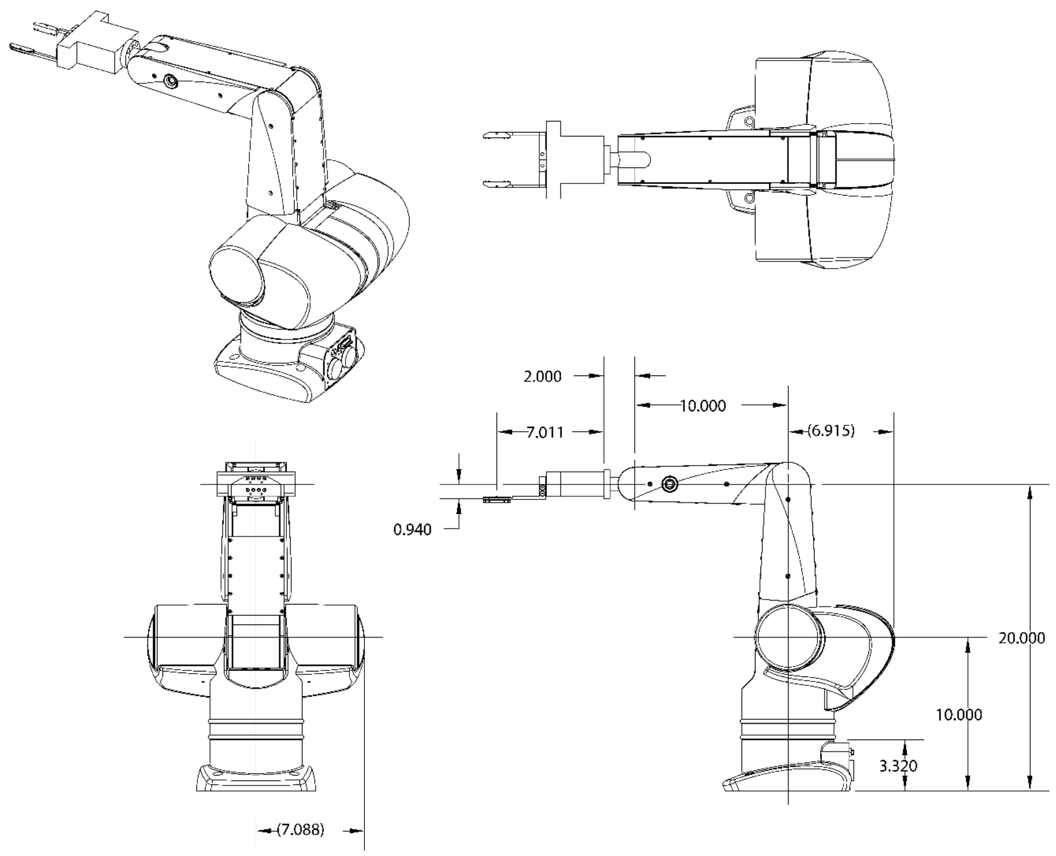


Figure 1. CRS Robot with dimensions in inches

3 Forward Kinematics

3.1 Frame Construction

For rotary joints, the zi axis is built along the axis of rotation, while for prismatic joints, the axis of translation is used. The xi axis is then positioned to be perpendicular to the zi-1 axis while also intersecting it. Following that, the yi axis is added using right-handed conventions. Figure 1 shows the dimensions of the robot in inches in front, top and isometric view. The position direction of rotation is shown in Figure 2, marked by the arrows, which is then used to assign the frames. Figure 4 shows the assigned frame of the first three joints of the CRS robot arm.

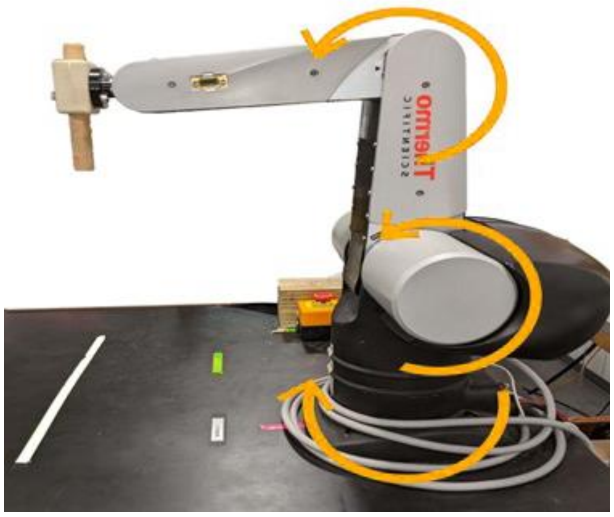


Figure 2. Zero Configuration for Motor Angles

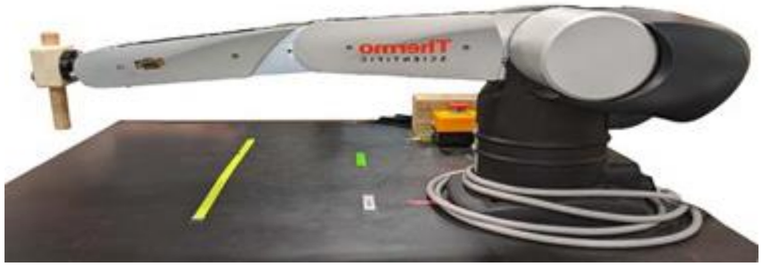


Figure 3. Zero Configuration for DH angles

3.2 DH Table

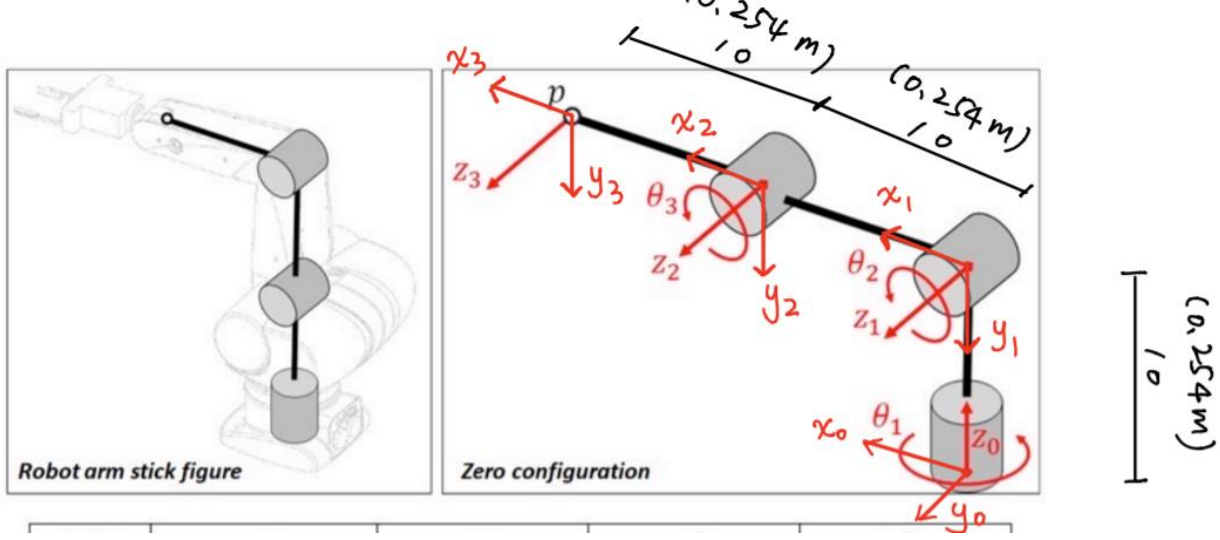


Figure 4. Frames assigned to each joint according to DH parameter convention

Joint	a_i (inch)	α_i (deg)	d_i (inch)	θ_i (deg)
1	0	-90	10	Θ1
2	10	0	0	Θ2
3	10	0	0	Θ3

Table 1. DH Table for 3 joints

The reference frames are laid out as follows:

- The z-axis is in the direction of the joint axis
- The x-axis is parallel to the common normal: $x_n = z_n*z_{n-1}$ (or away from zn-1)
- If there is no unique common normal (parallel z axes), then d is a free parameter. The direction of x_n is from z_{n-1} to z_n.
- The y-axis follows from the x and z-axis by choosing it to be a right-handed coordinate system.

Take Joint 1 as an example, a1 is the distance between z0 and z1 measured along x1, as we can see from Figure 4, the distance is 0, so a1 = 0. α1 is the angle between z0 and z1 measured in a plane normal to x1. So we can get the angle is -90 degree from Figure 4. d1 is the perpendicular distance from the origin o0 to the intersection of the x1 axis with z0 measured along the z0 axis. From Figure 4, the distance is 10 inches. $\theta 1$ is the angle between x0 and x1 measured in a plane normal to z0. As Figure 4 shows, the angle is Θ1.

After determining the DH parameters, the Homogenous Tranformation Matrix can be determined by using the following equations, which is then used to find the position p0 with respect to the zero frame.

$$Rot(z0,\theta 1) = \begin{bmatrix} \cos\theta 1 & -\sin\theta 1 & 0 & 0 \\ \sin\theta 1 & \cos\theta 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Trans(z0,d1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Trans(x1,a1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Rot(x1, \alpha 1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\frac{-\pi}{2} & -\sin\frac{-\pi}{2} & 0 \\ 0 & \sin\frac{-\pi}{2} & \cos\frac{-\pi}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Rot(x1, \alpha 1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A1(\theta 1) = \ Rot(z0, \theta 1) Trans(z0, d1) Trans(x1, a1) Rot(x1, \alpha 1)$$

$$A1(\theta 1) = \begin{bmatrix} \cos\theta 1 & -\sin\theta 1 \cos\alpha 1 & -\sin\theta 1 \sin\alpha 1 & a1cos\theta 1 \\ \sin\theta 1 & \cos\theta 1 \cos\alpha 1 & -\cos\theta 1 \sin\alpha 1 & a1sin\theta 1 \\ 0 & \sin\alpha 1 & \cos\alpha 1 & d1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A2(\theta 2) = \begin{bmatrix} \cos\theta 2 & -\sin\theta 2 \cos\alpha 2 & -\sin\theta 2 \sin\alpha 2 & a2cos\theta 2 \\ \sin\theta 2 & \cos\theta 2 \cos\alpha 2 & -\cos\theta 2 \sin\alpha 2 & a2sin\theta 2 \\ 0 & \sin\alpha 2 & \cos\alpha 2 & d2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A3(\theta 3) = \begin{bmatrix} \cos\theta 3 & -\sin\theta 3 \cos\alpha 3 & -\sin\theta 3 \sin\alpha 3 & a3cos\theta 3 \\ \sin\theta 3 & \cos\theta 3 \cos\alpha 3 & -\cos\theta 3 \sin\alpha 3 & a3sin\theta 3 \\ 0 & \sin\alpha 3 & \cos\alpha 3 & d3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_n^0 = A_1(q_1) A_2(q_2) \ldots A_n(q_n)$$

Which in our case is given by

$$T_3^0 = A_1(\theta 1) A_2(\theta 2)\ A3(\theta 3)$$

```
T03 =
```

$$\begin{pmatrix} \cos(\theta_2+\theta_3)\cos(\theta_1) & -\sin(\theta_2+\theta_3)\cos(\theta_1) & -\sin(\theta_1) & 10\cos(\theta_1)\,\sigma_1 \\ \cos(\theta_2+\theta_3)\sin(\theta_1) & -\sin(\theta_2+\theta_3)\sin(\theta_1) & \cos(\theta_1) & 10\sin(\theta_1)\,\sigma_1 \\ -\sin(\theta_2+\theta_3) & -\cos(\theta_2+\theta_3) & 0 & 10 - 10\sin(\theta_2) - 10\sin(\theta_2+\theta_3) \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

where

$$\sigma_1 = \cos(\theta_2+\theta_3) + \cos(\theta_2)$$

Using the above equations, we solved for T_3^0 and found the position vector p_3^0 as a function of $\theta 1, \theta 2$ and $\theta 3$.

$$p^0 = \begin{bmatrix} x_w(\theta_1, \theta_2, \theta_3) \\ y_w(\theta_1, \theta_2, \theta_3) \\ z_w(\theta_1, \theta_2, \theta_3) \end{bmatrix}$$

$$p^0 = \begin{bmatrix} f_x(\theta_1, \theta_2, \theta_3) \\ f_y(\theta_1, \theta_2, \theta_3) \\ f_z(\theta_1, \theta_2, \theta_3) \end{bmatrix} = \begin{bmatrix} L\cos(\theta_1)\,(\cos(\theta_2 + \theta_3) + \cos(\theta_2)) \\ L\sin(\theta_1)\,(\cos(\theta_2 + \theta_3) + \cos(\theta_2)) \\ L(1 - \sin(\theta_2) - \sin(\theta_2 + \theta_3)) \end{bmatrix} \quad (1)$$

Relationship between the DH thetas and the motor thetas.

We choose three configurations:

Configuration	Configuration 1	Configuration 2	Configuration 3
DH angles	[0, 0, 0]	[0, -90, 0]	[0, -90, 90]
Motor angles	[0, 90, 0]	[0, 0, -90]	[0, 0, 0]

Table 2. Configurations chosen to calculate the relation relationship between the DH and the motor angles

From the three equations below, we can solve for c, c1, c2, c3 and get the relationship between the DH thetas and the motor thetas shown above.

theta1DH = theta1Motor
theta2DH = theta2Motor + c
theta3DH = c1 + c2*theta2Motor + c3* theta3Motor

After substituting the values of theta motor in the position equation 1, we get the value of x,y,z as a function of $\theta_{1M}, \theta_{2M}, \theta_{3M}$ which is given the following equation. It is convenient for us to work with DH parameters, as we are well aware of how to determine its forward and inverse kinematics. However, the commands sent to the motor from code composer must be in motor angles, and these are related to the motor based on the above equations. Two of motor joints are coupled and hence the motor angles are related to one another. So, one experiment to determine the relationship between DH angles and motor angles is to move the robot in three different configurations and measure the motor angles. Substituting both angles in equations and solving for DH angles give us the following results.

$$\theta_{1DH} = \theta_{1M}$$

$$\theta_{2DH} = \theta_{2M} - 90^{o}$$

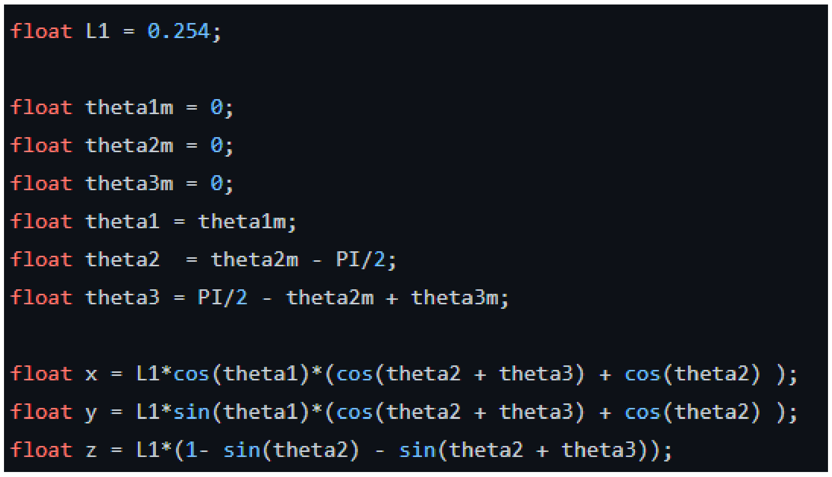
$$\theta_{3DH} = -\theta_{2M} + \theta_{3M} + 90^{o}$$

p0 =

$$\begin{pmatrix} 10\cos(\theta_1)\ (\cos(\theta_2+\theta_3)+\cos(\theta_2)) \\ 10\sin(\theta_1)\ (\cos(\theta_2+\theta_3)+\cos(\theta_2)) \\ 10-10\sin(\theta_2)-10\sin(\theta_2+\theta_3) \\ 1 \end{pmatrix}$$

P0 is the fourth column of T03 HTM, and the forward kinematics of the end effector position is given by the first three rows of p0. This is used in both C Code and MATLAB code to find forward kinematics.

C Code for forward kinematics:

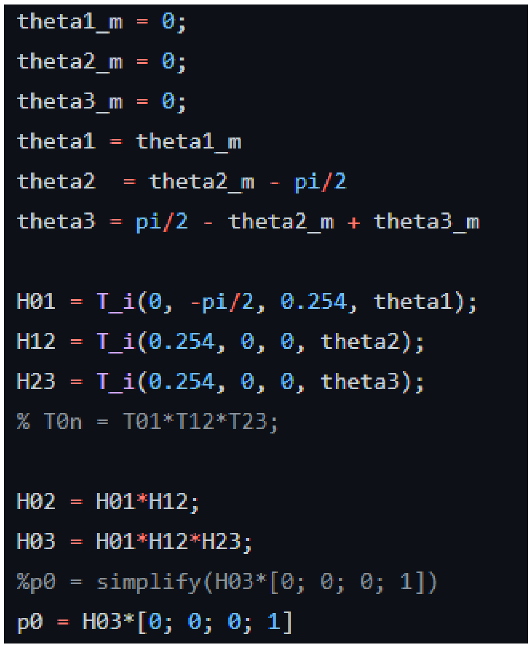


```
float L1 = 0.254;

float theta1m = 0;
float theta2m = 0;
float theta3m = 0;
float theta1 = theta1m;
float theta2  = theta2m - PI/2;
float theta3 = PI/2 - theta2m + theta3m;

float x = L1*cos(theta1)*(cos(theta2 + theta3) + cos(theta2) );
float y = L1*sin(theta1)*(cos(theta2 + theta3) + cos(theta2) );
float z = L1*(1- sin(theta2) - sin(theta2 + theta3));
```

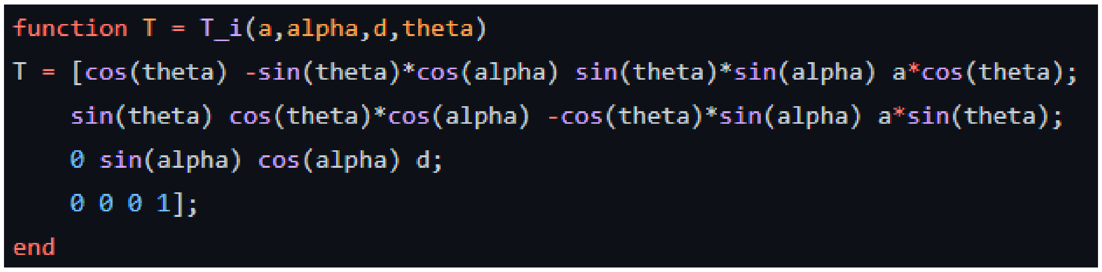
MATLAB Code for forward kinematics



```
theta1_m = 0;
theta2_m = 0;
theta3_m = 0;
theta1 = theta1_m
theta2  = theta2_m - pi/2
theta3 = pi/2 - theta2_m + theta3_m

H01 = T_i(0, -pi/2, 0.254, theta1);
H12 = T_i(0.254, 0, 0, theta2);
H23 = T_i(0.254, 0, 0, theta3);
% T0n = T01*T12*T23;

H02 = H01*H12;
H03 = H01*H12*H23;
%p0 = simplify(H03*[0; 0; 0; 1])
p0 = H03*[0; 0; 0; 1]
```



```
function T = T_i(a,alpha,d,theta)
T = [cos(theta) -sin(theta)*cos(alpha) sin(theta)*sin(alpha) a*cos(theta);
    sin(theta) cos(theta)*cos(alpha) -cos(theta)*sin(alpha) a*sin(theta);
    0 sin(alpha) cos(alpha) d;
    0 0 0 1];
end
```

4 Inverse Kinematics

The CRS robot's inverse kinematics were calculated analytically using simple geometry. The task was to find the joint angles to produce such a configuration given a set of end-effector coordinates. Angle θ1 was determined by viewing the robot from the top, as shown in Figure 5. The robot then forms a simple triangle, where the tangent function can be used to calculate the angle. The robot was viewed from the side for θ2 and θ3, with the z0 axis pointing up. Based on the given coordinates and lengths of the links, we were able to determine values for the angles using the law of cosines and basic geometric properties.

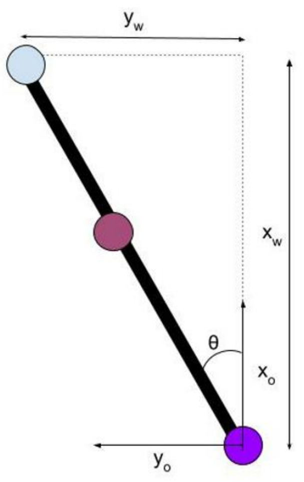


Figure 5. Analytical solution to θ_1, top view

$$atan2(\theta 1) = \frac{yw}{xw}$$

$$\theta 1 = atan2^{-1}\frac{yw}{xw}$$

$$r = \sqrt{xw^2 + yw^2}$$

To find θ2. Θ3 we project the configuration on to the (r,zo) plane, where we get the following figure

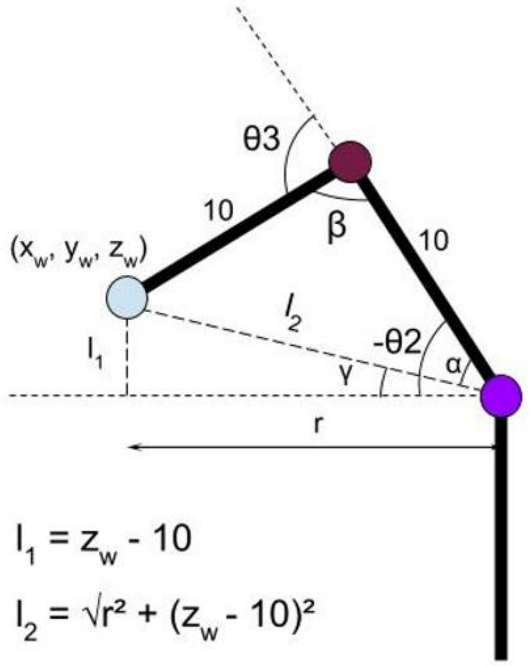


Figure 6. Analytical solution to θ_2 and θ_3.

We find the value of θ2 and θ3 from the Figure 6 above, using the following steps.

$$l1 = zw - 10$$

$$l1^2 + r^2 = L2^2$$

$$l2 = \sqrt{(xw^2 + yw^2) + (zw - 10)^2}$$

Compute β:

$$l2^2 = r^2 + l1^2 = 10^2 + 10^2 - 2 * 10 * 10 * \cos\beta$$

$$\beta = \cos^{-1}\frac{200 - l2^2}{200} = \cos^{-1}\frac{200 - (r^2 + (zw - 10)^2)}{200}$$

Compute α:

$$10^2 = 10^2 + l2^2 - 2 * 10 * l2 * cos\alpha$$

$$\alpha = \cos^{-1}\frac{l2^2}{20 * l2} = \cos^{-1}\frac{(r^2 + (zw - 10)^2)}{20\sqrt{r^2 + (zw - 10)^2}}$$

$$\gamma = a\tan 2^{-1}\frac{zw - 10}{r}$$

Solution:

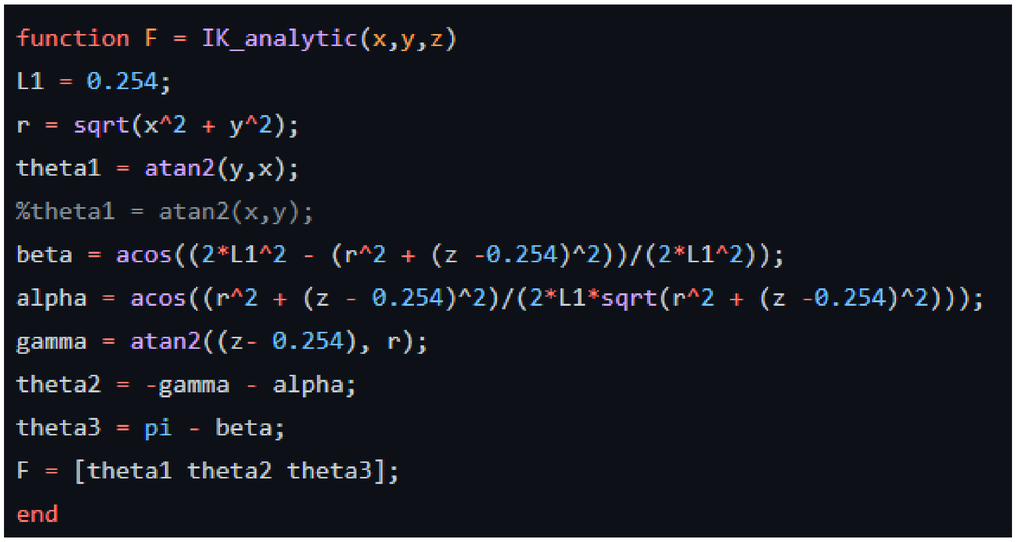
$$\theta 1 = a\tan 2^{-1}\frac{yw}{xw}$$

$$-\theta 2 = \gamma + \alpha$$

$$\theta 2 = -\gamma - \alpha$$

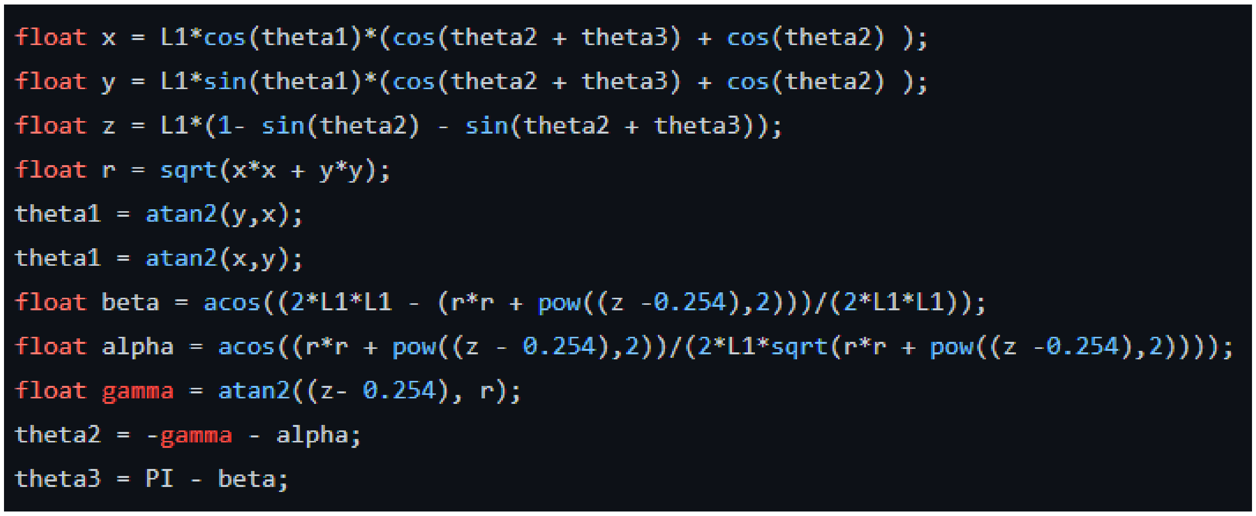
$$\theta 3 = \pi - \beta$$

MATLAB Code for inverse kinematics:



```matlab
function F = IK_analytic(x,y,z)
L1 = 0.254;
r = sqrt(x^2 + y^2);
theta1 = atan2(y,x);
%theta1 = atan2(x,y);
beta = acos((2*L1^2 - (r^2 + (z -0.254)^2))/(2*L1^2));
alpha = acos((r^2 + (z - 0.254)^2)/(2*L1*sqrt(r^2 + (z -0.254)^2)));
gamma = atan2((z- 0.254), r);
theta2 = -gamma - alpha;
theta3 = pi - beta;
F = [theta1 theta2 theta3];
end
```

C Code for forward kinematics:

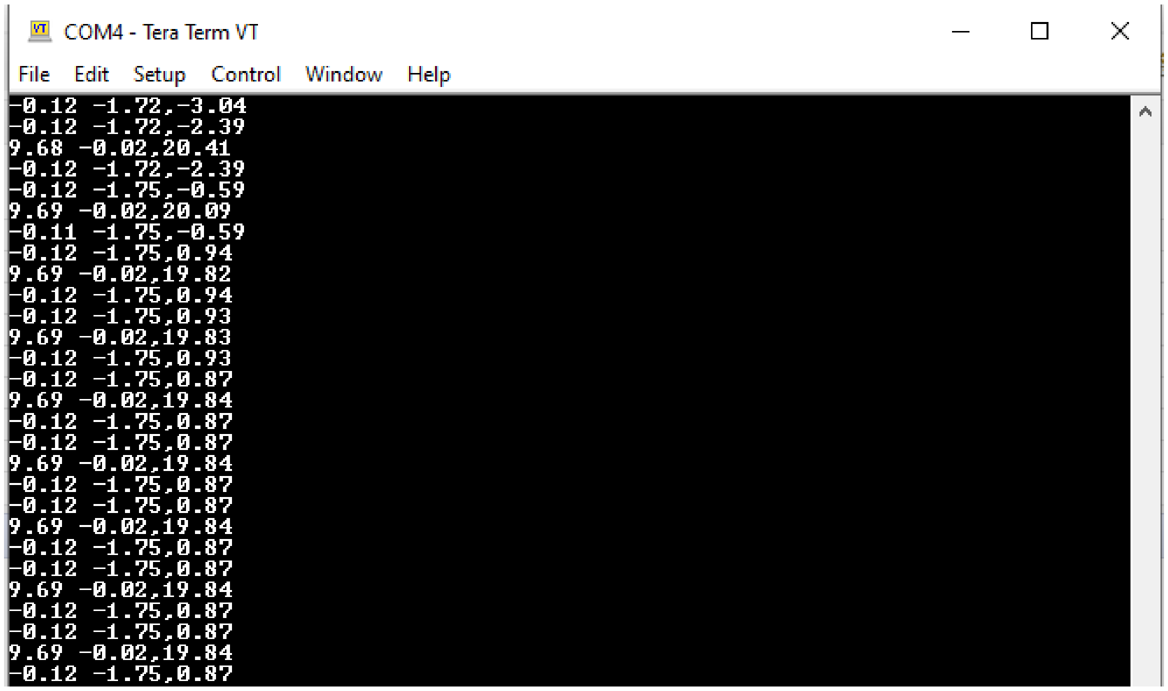


```c
float x = L1*cos(theta1)*(cos(theta2 + theta3) + cos(theta2) );
float y = L1*sin(theta1)*(cos(theta2 + theta3) + cos(theta2) );
float z = L1*(1- sin(theta2) - sin(theta2 + theta3));
float r = sqrt(x*x + y*y);
theta1 = atan2(y,x);
theta1 = atan2(x,y);
float beta = acos((2*L1*L1 - (r*r + pow((z -0.254),2)))/(2*L1*L1));
float alpha = acos((r*r + pow((z - 0.254),2))/(2*L1*sqrt(r*r + pow((z -0.254),2))));
float gamma = atan2((z- 0.254), r);
theta2 = -gamma - alpha;
theta3 = PI - beta;
```

5 Conclusion

Values were printed to Tera Term to verify validity after the forward and inverse kinematics were computed. Forward kinematics takes the three motor angles, converts them to Denavit-Hartenberg thetas, and then uses that to calculate the end effector's position. The forward kinematics should return X = 10, Y = 0, Z = 20 when in the motor zero configuration. The forward kinematics should return X = 20, Y = 0, Z = 10 when in the Denavit-Hartenberg zero configuration. The inverse kinematics solves for the Denavit-Hartenberg angles using the end effector position calculated by the forward kinematics. Finally, the encoder-measured motor angles are printed alongside the converted motor angles for comparison. The motor angles for the motor zero configuration should be (0, 0, 0). The motor angles for the Denavit-Hartenberg zero configuration should be (0, 90, 0). In Figures 7 and 8, you can see the Tera Term outputs for both configurations.

Tera Term Images:

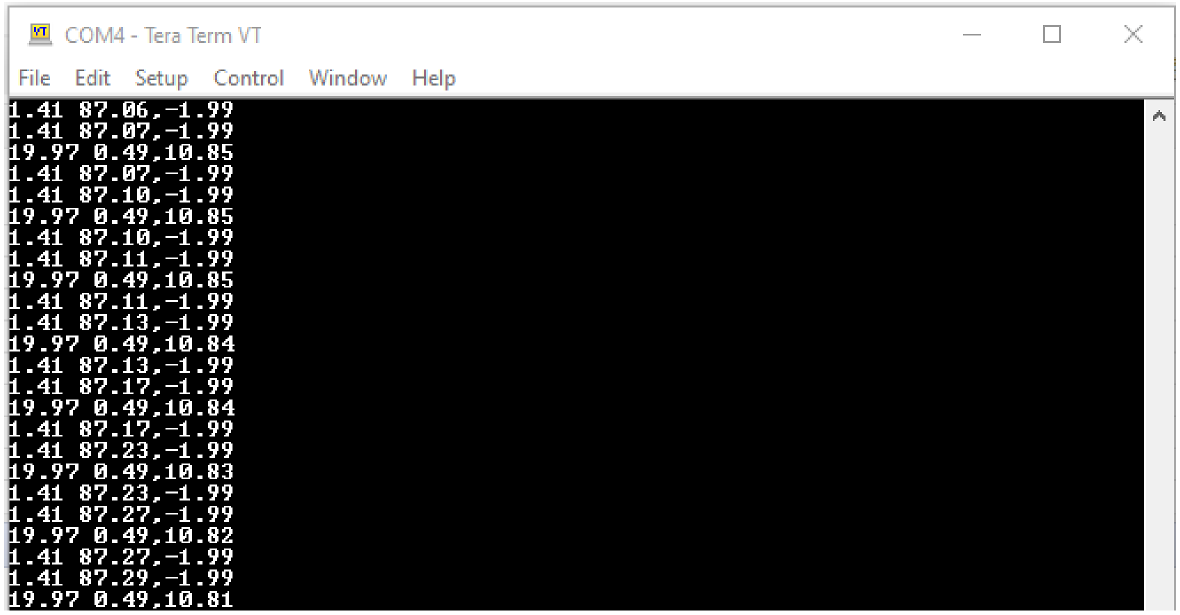


```
-0.12 -1.72,-3.04
-0.12 -1.72,-2.39
9.68 -0.02,20.41
-0.12 -1.72,-2.39
-0.12 -1.75,-0.59
9.69 -0.02,20.09
-0.11 -1.75,-0.59
-0.12 -1.75,0.94
9.69 -0.02,19.82
-0.12 -1.75,0.94
-0.12 -1.75,0.93
9.69 -0.02,19.83
-0.12 -1.75,0.93
-0.12 -1.75,0.87
9.69 -0.02,19.84
-0.12 -1.75,0.87
-0.12 -1.75,0.87
9.69 -0.02,19.84
-0.12 -1.75,0.87
-0.12 -1.75,0.87
9.69 -0.02,19.84
-0.12 -1.75,0.87
-0.12 -1.75,0.87
9.69 -0.02,19.84
-0.12 -1.75,0.87
-0.12 -1.75,0.87
9.69 -0.02,19.84
-0.12 -1.75,0.87
```

Figure 7. Tera Term outputs for motor zero configuration

Motor angles $(\theta 1, \theta 2, \theta 3)$	-0.12	-0.75	0.87
Position of end-effector (x,y,z)	9.69	-0.02	19.84
Motor angles calculated$(\theta 1, \theta 2, \theta 3)$	-0.12	-0.75	0.87

Table 3. Tera Term outputs for motor zero configuration

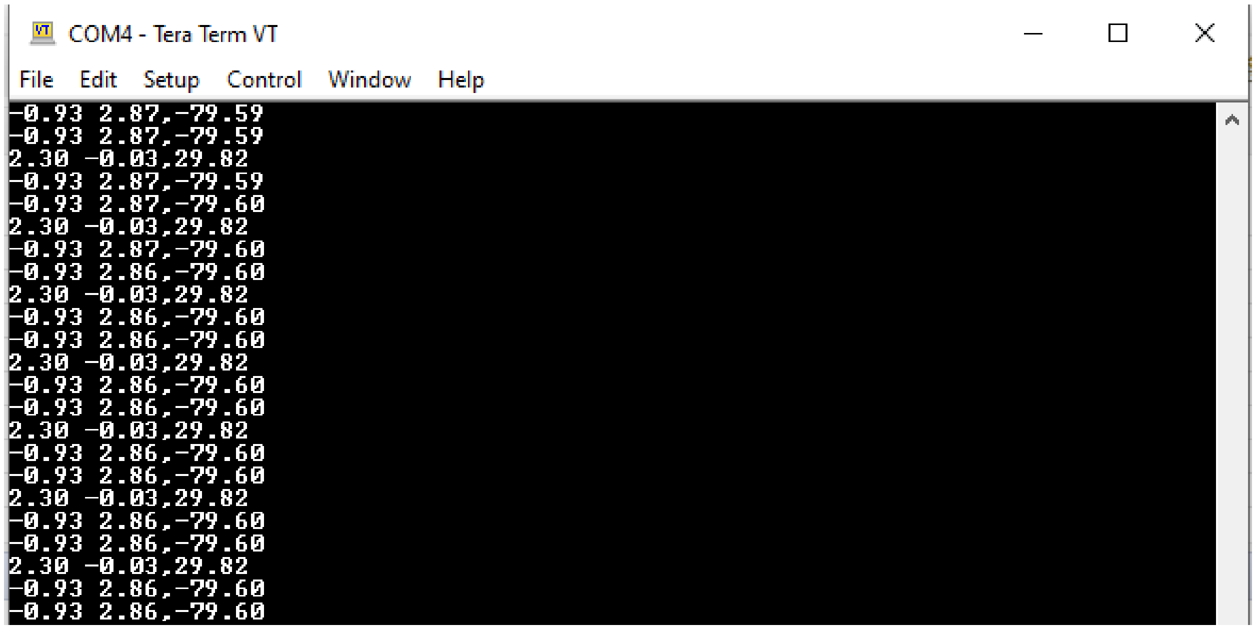


```
1.41 87.06,-1.99
1.41 87.07,-1.99
19.97 0.49,10.85
1.41 87.07,-1.99
1.41 87.10,-1.99
19.97 0.49,10.85
1.41 87.10,-1.99
1.41 87.11,-1.99
19.97 0.49,10.85
1.41 87.11,-1.99
1.41 87.13,-1.99
19.97 0.49,10.84
1.41 87.13,-1.99
1.41 87.17,-1.99
19.97 0.49,10.84
1.41 87.17,-1.99
1.41 87.23,-1.99
19.97 0.49,10.83
1.41 87.23,-1.99
1.41 87.27,-1.99
19.97 0.49,10.82
1.41 87.27,-1.99
1.41 87.29,-1.99
19.97 0.49,10.81
```

Figure 8. Tera Term outputs for Denavit-Hartenberg zero configuration

Motor angles ($\theta 1, \theta 2, \theta 3$)	1.41	-87.06	-1.99
Position of end-effector (x,y,z)	19.97	0.49	10.85
Motor angles calculated($\theta 1, \theta 2, \theta 3$)	1.41	-87.06	-1.99

Table 4. Tera Term outputs for Denavit-Hartenberg zero configuration



```
-0.93 2.87,-79.59
-0.93 2.87,-79.59
2.30 -0.03,29.82
-0.93 2.87,-79.59
-0.93 2.87,-79.60
2.30 -0.03,29.82
-0.93 2.87,-79.60
-0.93 2.86,-79.60
2.30 -0.03,29.82
-0.93 2.86,-79.60
-0.93 2.86,-79.60
2.30 -0.03,29.82
-0.93 2.86,-79.60
-0.93 2.86,-79.60
2.30 -0.03,29.82
-0.93 2.86,-79.60
-0.93 2.86,-79.60
2.30 -0.03,29.82
-0.93 2.86,-79.60
-0.93 2.86,-79.60
2.30 -0.03,29.82
-0.93 2.86,-79.60
-0.93 2.86,-79.60
```

Figure 9. Tera Term outputs when the CRS robot is near vertical position (DH thetas: 0, -90, 0)

Motor angles ($\theta 1, \theta 2, \theta 3$)	-0.93	2.86	-79.60
Position of end-effector (x,y,z)	2.30	-0.03	29.82
Motor angles calculated($\theta 1, \theta 2, \theta 3$)	-0.93	2.86	-79.60

Table 5. Tera Term outputs when the CRS robot is near vertical position (DH thetas: 0, -90, 0)

From the tables we got from the Tera Term outputs, we can conclude that the values of motor angles are the same as the values of the motor angles calculated by us. So, we verify that our calculation is correct.

6 C Code

```
#include <tistdtypes.h>
#include <coecsl.h>
#include "user_includes.h"
#include "math.h"
const double pi= PI;

// These two offsets are only used in the main file user_CRSRobot.c
You just need to create them here and find the correct offset and then
these offset will adjust the encoder readings
float offset_Enc2_rad = -0.4389;//0;//-0.43244; //-0.37;0.479/-0.43244
float offset_Enc3_rad = 0.22689;//03; //0.27;-0.188/0.203

// Your global varialbes.
long mycount = 0;

#pragma DATA_SECTION(whattoprint, ".my_vars")
float whattoprint = 0.0;
#pragma DATA_SECTION(theta1array, ".my_arrs")
float theta1array[100];
#pragma DATA_SECTION(robo, ".my_vars")
float robo = 10.0;
#pragma DATA_SECTION(theta2array, ".my_arrs")
float theta2array[100];
long arrayindex = 0;

//Values to be printed
float printtheta1motor = 0;
float printtheta2motor = 0;
float printtheta3motor = 0;
float theta1_dh2 = 0;
float theta2_dh2 = 0;
float theta3_dh2 = 0;
float t1m = 0;
float t2m = 0;
float t3m = 0;
float x = 0;
float y = 0;
float z = 0;

float r = 0;
float gamma = 0;
float alpha = 0;
float beta = 0;

// Assign these float to the values you would like to plot in Simulink
float Simulink_PlotVar1 = 0;
float Simulink_PlotVar2 = 0;
float Simulink_PlotVar3 = 0;
float Simulink_PlotVar4 = 0;

// This function is called every 1 ms
void lab(float theta1motor,float theta2motor,float theta3motor,float
*tau1,float *tau2,float *tau3, int error) {
      *tau1 = 0;
      *tau2 = 0;
      *tau3 = 0;

//Motor torque limitation (Max: 5 Min: -5)
// save past states
if ((mycount%50)==0) {

      theta1array[arrayindex] = theta1motor;
      theta2array[arrayindex] = theta2motor;
      if (arrayindex >= 100) {
            arrayindex = 0;
      } else {
            arrayindex++;
      }
}
if ((mycount%500)==0) {
      if (whattoprint > 0.5) {
            serial_printf(&SerialA, "I love robotics\n\r");
      } else {
            printtheta1motor = theta1motor;
            printtheta2motor = theta2motor;
            printtheta3motor = theta3motor;
//Motor Thetas to DH Thetas
            float L = 10;
            float theta1 = theta1motor;
            float theta2  = theta2motor - pi/2;
            float theta3 = pi/2 - theta2motor + theta3motor;
      //Forward Kinematics Calculation
            x = L*cos(theta1)*(cos(theta2 + theta3) + cos(theta2) );
            y = L*sin(theta1)*(cos(theta2 + theta3) + cos(theta2) );
            z = L*(1- sin(theta2) - sin(theta2 + theta3));

      //Inverse Kinematics Calculation
            r = sqrt(x*x + y*y);
            gamma = atan2(z-10,r);
            alpha = acos((r*r + (z-10)*(z-10))/(20*sqrt(r*r + (z-
            10)*(z-10))));
            beta = acos((200-(r*r+(z-10)*(z-10)))/200);
            theta1_dh2 = atan2(y,x);
            theta2_dh2 = -gamma - alpha;
```

```
           theta3_dh2 = pi - beta;
      //Convert IK angles from DH to Motor Angles for output
           t1m = theta1_dh2;
           t2m = theta2_dh2 + PI/2;
           t3m = theta3_dh2 + theta2_dh2;

           SWI_post(&SWI_printf); //Using a SWI to fix SPI issue from
sending too many floats.
           }
           GpioDataRegs.GPBTOGGLE.bit.GPIO34 = 1; // Blink LED on
Control Card
           GpioDataRegs.GPBTOGGLE.bit.GPIO60 = 1; // Blink LED on
Emergency Stop Box
      }
      Simulink_PlotVar1 = theta1motor;
      Simulink_PlotVar2 = theta2motor;
      Simulink_PlotVar3 = theta3motor;
      Simulink_PlotVar4 = 0;
      mycount++;
}


void printing(void){
      serial_printf(&SerialA, "%.2f %.2f,%.2f
\n\r",printtheta1motor*180/PI,printtheta2motor*180/PI,printtheta3motor
*180/PI);
    serial_printf(&SerialA, "%.2f %.2f,%.2f   \n\r",x,y,z);
    serial_printf(&SerialA, "%.2f %.2f,%.2f
\n\r",t1m*180/PI,t2m*180/PI,t3m*180/PI);

}
```