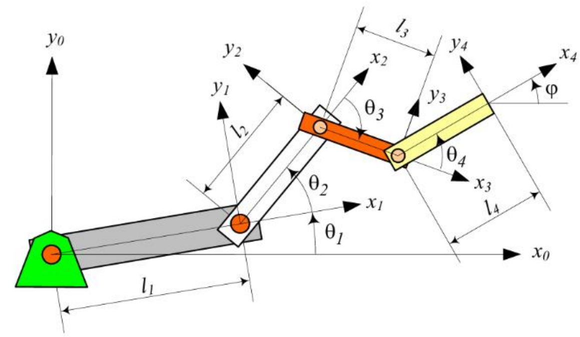
Robotic-Arm Mathematical Model Calculations

By:

Amr Osama Abd-Allah

Calculations:



L1 = 54.00 mm

L2 = 109.990 mm

L3 = 69.607 mm

L4 = 116.826 mm

```
function [theta1, theta2, theta3, theta4] = inverse_kinematics(x, y, z, phi)
  % Link lengths in mm
  L1 = 54.00;
  L2 = 109.990;
  L3 = 69.607;
  L4 = 116.826;

  % Calculate theta1
  theta1 = atan2(y, x);

  % Calculate the wrist center position

  wx = x - L4 * cos(phi) * cos(theta1);
  wy = y - L4 * cos(phi) * sin(theta1);
  wz = z - L4 * sin(phi);

  % Calculate the distance from the base to the wrist center
  r = sqrt(wx^2 + wy^2);
  s = wz - L1;

  % Calculate theta2 and theta3 using the law of cosines
  D = (r^2 + s^2 - L2^2 - L3^2) / (2 * L2 * L3);

  % Clamp D to the range [-1, 1] to avoid errors due to numerical issues
  D = min(1, max(-1, D));
  theta3 = atan2(sqrt(1 - D^2), D);

  % Intermediate angle for theta2 calculation
  beta = atan2(L3 * sin(theta3), L2 + L3 * cos(theta3));
  theta2 = atan2(s, r) - beta;

  % Calculate theta4 based on the desired orientation phi
  theta4 = phi - theta2 - theta3;
end
```

$$
{}^{i-1}T_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^0T_1 = \begin{bmatrix} \cos\theta_1 & 0 & -\sin\theta_1 & 0 \\ \sin\theta_1 & 0 & \cos\theta_1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^1T_2 = \begin{bmatrix} \cos\theta_2 & 0 & \sin\theta_2 & 0 \\ \sin\theta_2 & 0 & -\cos\theta_2 & 0 \\ 0 & 1 & 0 & l_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^2T_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

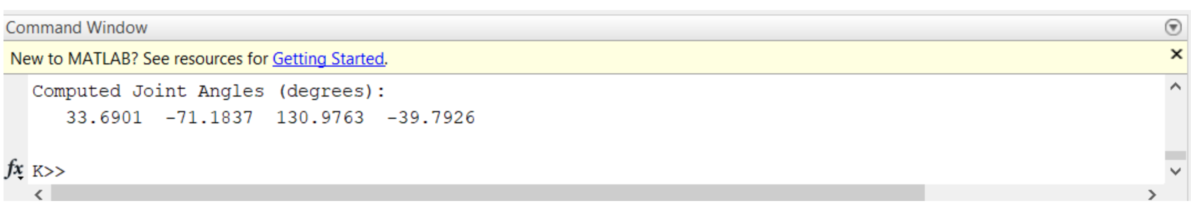
$$
{}^3T_4 = \begin{bmatrix} \cos\theta_4 & 0 & -\sin\theta_4 & 0 \\ \sin\theta_4 & 0 & \cos\theta_4 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

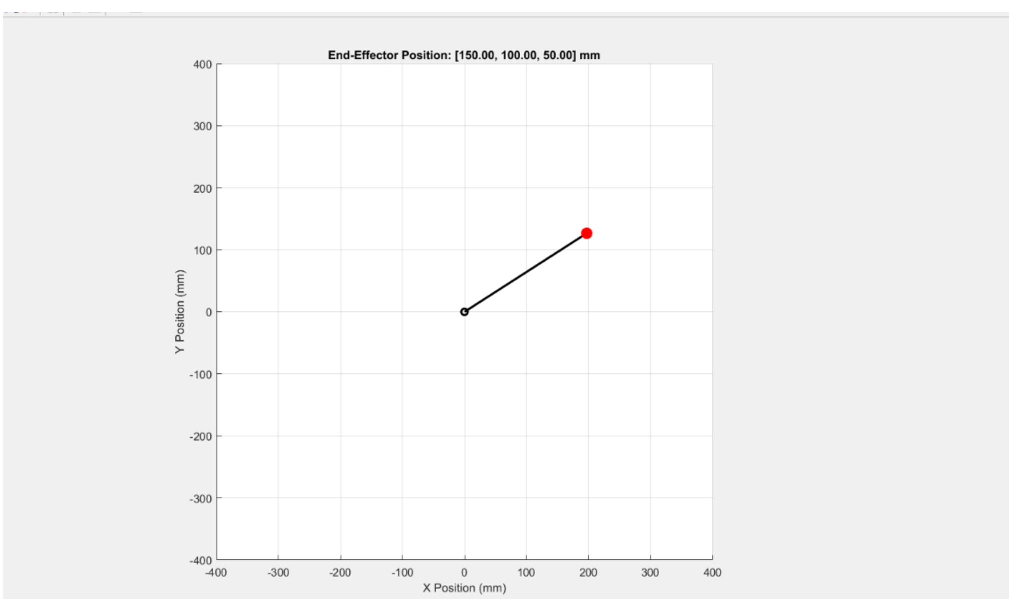
#Then call Function and Apply it

```
% Define the link lengths
a1 = 54.00;   % in mm
a2 = 109.990; % in mm
a3 = 69.607;  % in mm
a4 = 116.826; % in mm

% Define symbolic joint angles
theta = sym('theta', [1 4]);  % symbolic joint angles
a = [a1, a2, a3, a4];        % link lengths

% Construct transformation matrices using DH parameters
T = cell(1, 4);
for i = 1:4
  T{i} = [cos(theta(i)), -sin(theta(i)), 0, a(i)*cos(theta(i));
        sin(theta(i)), cos(theta(i)), 0, a(i)*sin(theta(i));
        0, 0, 1, 0;
        0, 0, 0, 1];
end

% Define a function for the full transformation from base to end-effector
full_transformation = @(th) double(subs(T{1}, theta(1), th(1)) * ...
                         subs(T{2}, theta(2), th(2)) * ...
                         subs(T{3}, theta(3), th(3)) * ...
                         subs(T{4}, theta(4), th(4)));

% Predefined end-effector position (X, Y, Z) and orientation (phi in degrees)
positions = [
  150, 100, 50, 20;
];

num_movements = 1;  % Number of movements

for move = 1:num_movements
  % Use predefined end-effector positions and solve for joint angles
  target_position = positions(move, 1:3);
  phi = deg2rad(positions(move, 4));  % Convert to radians
  [theta1, theta2, theta3, theta4] = inverse_kinematics(target_position(1), target_position(2),
target_position(3), phi);
  theta_rad = [theta1, theta2, theta3, theta4];
  theta_deg = rad2deg(theta_rad);  % Convert to degrees

  % Display results
  disp('Computed Joint Angles (degrees):');
  disp(theta_deg);
```

```
    % Plot the manipulator
    T_full = full_transformation(theta_rad);
    position_values = T_full(1:3, 4);
    orientation_value = atan2d(T_full(2, 1), T_full(1, 1));

    figure(1); clf; hold on;
    plot3([0, T_full(1, 4)], [0, T_full(2, 4)], [0, T_full(3, 4)], 'k-o', 'LineWidth', 2);
    plot3(T_full(1, 4), T_full(2, 4), T_full(3, 4), 'ro', 'MarkerSize', 10, 'MarkerFaceColor', 'r');
    title(sprintf('End-Effector Position: [%.2f, %.2f, %.2f] mm', target_position));
    xlabel('X Position (mm)');
    ylabel('Y Position (mm)');
    zlabel('Z Position (mm)');
    grid on;
    axis equal;
    xlim([-400, 400]);
    ylim([-400, 400]);
    zlim([-400, 400]);

    pause(1);  % Pause for 1 second between movements
end
```

Out:



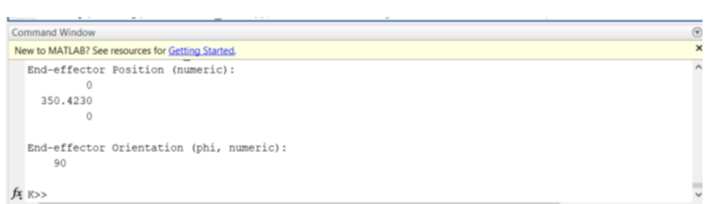


First let's Assume that we know the Angles

We can adjust angles at the code to find End Effector at the space with matrix using MATLAB Code:

```
% Define the link lengths
a1 = 54.00;   % in mm
a2 = 109.990; % in mm
a3 = 69.607;  % in mm
a4 = 116.826; % in mm

% Symbolic joint angles
theta = sym('theta', [1 4]);  % symbolic joint angles
d = [0, 0, 0, 0];  % for planar manipulator
a = [a1, a2, a3, a4];  % link lengths
alpha = [0, 0, 0, 0];  % for planar manipulator

% Construct transformation matrices using DH parameters
T = cell(1, 4);
for i = 1:4

  T{i} = [cos(theta(i)), -sin(theta(i)), 0, a(i)*cos(theta(i));
        sin(theta(i)), cos(theta(i)), 0, a(i)*sin(theta(i));
        0, 0, 1, d(i);
        0, 0, 0, 1];
end

% Multiply the matrices to get the transformation from base to end-effector
T_0_4 = T{1} * T{2} * T{3} * T{4};

% Extract end-effector position and orientation
end_effector_position = T_0_4(1:3, 4);
end_effector_orientation = atan2(T_0_4(2, 1), T_0_4(1, 1));

% Substitute specific values for theta1, theta2, theta3, and theta4
theta_values = [deg2rad(90), deg2rad(0), deg2rad(0), deg2rad(0)];  % example values in radians
position_values = double(subs(end_effector_position, theta, theta_values));
orientation_value = double(subs(end_effector_orientation, theta, theta_values));

disp('End-effector Position (numeric):');
disp(position_values);

disp('End-effector Orientation (phi, numeric):');
disp(rad2deg(orientation_value));  % convert to degrees
```

So at the last code we adjust theta1=90 and the rest of angles = 0 that means that the movement at Y axis and both of X,Z will be Zero as you see the output →



First let's Assume that we Don't know the Angles

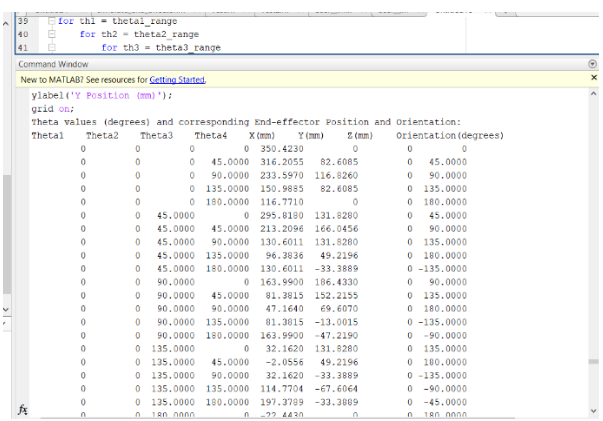
Code:

```
% Define the link lengths
a1 = 54.00;   % in mm
a2 = 109.990; % in mm
a3 = 69.607;  % in mm
a4 = 116.826; % in mm

% Symbolic joint angles
theta = sym('theta', [1 4]);  % symbolic joint angles
d = [0, 0, 0, 0];  % for planar manipulator
a = [a1, a2, a3, a4];  % link lengths
alpha = [0, 0, 0, 0];  % for planar manipulator

% Construct transformation matrices using DH parameters
T = cell(1, 4);
for i = 1:4
  T{i} = [cos(theta(i)), -sin(theta(i)), 0, a(i)*cos(theta(i));
        sin(theta(i)), cos(theta(i)), 0, a(i)*sin(theta(i));
        0, 0, 1, d(i);
        0, 0, 0, 1];
end

% Multiply the matrices to get the transformation from base to end-effector
T_0_4 = T{1} * T{2} * T{3} * T{4};

% Extract end-effector position and orientation
end_effector_position = T_0_4(1:3, 4);
end_effector_orientation = atan2(T_0_4(2, 1), T_0_4(1, 1));

% Define a range of angles for simulation (0 to 180 degrees -> 0 to pi radians)
theta1_range = linspace(0, pi, 5);  % range for theta1
theta2_range = linspace(0, pi, 5);  % range for theta2
theta3_range = linspace(0, pi, 5);  % range for theta3
theta4_range = linspace(0, pi, 5);  % range for theta4

% Initialize a matrix to store end-effector positions
positions = [];

% Iterate through the angle ranges and compute positions
for th1 = theta1_range
  for th2 = theta2_range
    for th3 = theta3_range
      for th4 = theta4_range
        theta_values_rad = [th1, th2, th3, th4];
```

```
            theta_values_deg = rad2deg(theta_values_rad);  % Convert to degrees
            position_values = double(subs(end_effector_position, theta, theta_values_rad));
            orientation_value = double(subs(end_effector_orientation, theta, theta_values_rad));


            % Store the positions and orientations
            positions = [positions; theta_values_deg, position_values', rad2deg(orientation_value)];
         end
      end
   end
end

% Display the results
disp('Theta values (degrees) and corresponding End-effector Position and Orientation:');
disp('Theta1   Theta2   Theta3   Theta4   X(mm)   Y(mm)   Z(mm)   Orientation(degrees)');
disp(positions);

% Plot the end-effector positions
figure;
scatter(positions(:, 5), positions(:, 6), 'filled');
title('End-Effector Positions (0 to 180 Degrees)');
xlabel('X Position (mm)');
ylabel('Y Position (mm)');
grid on;
```

The code here depends on the Lengths we have adjust of Our robot and apply DH matrix to calculate the Theta and X,Y,Z positions so we can determine the End Effector positions at space, but will all possibilities Of Theta and the axis

Here's the out example:

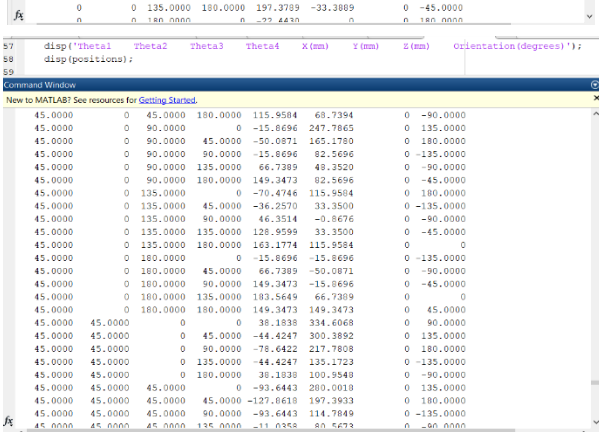


```
for th1 = theta1_range
    for th2 = theta2_range
        for th3 = theta3_range
```

Command Window

New to MATLAB? See resources for Getting Started.

```
ylabel('Y Position (mm)');
grid on;
Theta values (degrees) and corresponding End-effector Position and Orientation:
Theta1    Theta2    Theta3    Theta4    X(mm)     Y(mm)     Z(mm)    Orientation(degrees)
         0         0         0         0  350.4230         0         0         0
         0         0         0   45.0000  316.2055   82.6085         0   45.0000
         0         0         0   90.0000  233.5970  116.8260         0   90.0000
         0         0         0  135.0000  150.9885   82.6085         0  135.0000
         0         0         0  180.0000  116.7710         0         0  180.0000
         0         0   45.0000         0  295.8180  131.8280         0   45.0000
         0         0   45.0000   45.0000  213.2096  166.0456         0   90.0000
         0         0   45.0000   90.0000  130.6011  131.8280         0  135.0000
         0         0   45.0000  135.0000   96.3836   49.2196         0  180.0000
         0         0   45.0000  180.0000  130.6011  -33.3889         0 -135.0000
         0         0   90.0000         0  163.9900  186.4330         0   90.0000
         0         0   90.0000   45.0000   81.3815  152.2155         0  135.0000
         0         0   90.0000   90.0000   47.1640   69.6070         0  180.0000
         0         0   90.0000  135.0000   81.3815  -13.0015         0 -135.0000
         0         0   90.0000  180.0000  163.9900  -47.2190         0  -90.0000
         0         0  135.0000         0   32.1620  131.8280         0  135.0000
         0         0  135.0000   45.0000   -2.0556   49.2196         0  180.0000
         0         0  135.0000   90.0000   32.1620  -33.3889         0 -135.0000
         0         0  135.0000  135.0000  114.7704  -67.6064         0  -90.0000
         0         0  135.0000  180.0000  197.3789  -33.3889         0  -45.0000
```

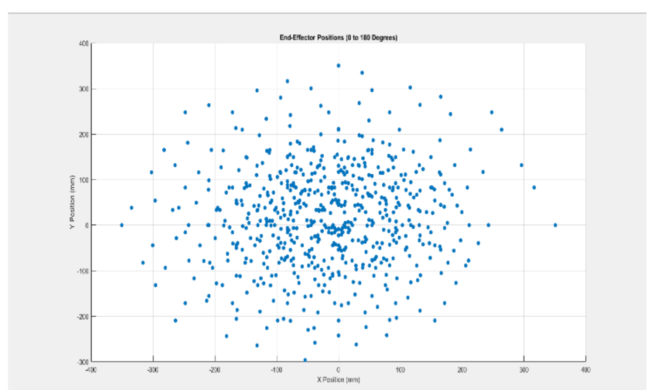


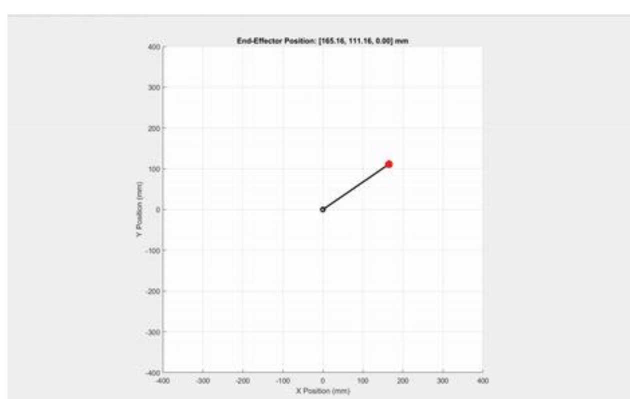
```
disp('Theta1    Theta2    Theta3    Theta4    X(mm)    Y(mm)    Z(mm)    Orientation(degrees)');
disp(positions);
```

Command Window

New to MATLAB? See resources for Getting Started.

```
   45.0000         0   45.0000  180.0000  115.9584   68.7394         0  -90.0000
   45.0000         0   90.0000         0  -15.8696  247.7865         0  135.0000
   45.0000         0   90.0000   45.0000  -50.0871  165.1780         0  180.0000
   45.0000         0   90.0000   90.0000  -15.8696   82.5696         0 -135.0000
   45.0000         0   90.0000  135.0000   66.7389   48.3520         0  -90.0000
   45.0000         0   90.0000  180.0000  149.3473   82.5696         0  -45.0000
   45.0000         0  135.0000         0  -70.4746  115.9584         0  180.0000
   45.0000         0  135.0000   45.0000  -36.2570   33.3500         0 -135.0000
   45.0000         0  135.0000   90.0000   46.3514   -0.8676         0  -90.0000
   45.0000         0  135.0000  135.0000  128.9599   33.3500         0  -45.0000
   45.0000         0  135.0000  180.0000  163.1774  115.9584         0         0
   45.0000         0  180.0000         0  -15.8696  -15.8696         0 -135.0000
   45.0000         0  180.0000   45.0000   66.7389  -50.0871         0  -90.0000
   45.0000         0  180.0000   90.0000  149.3473  -15.8696         0  -45.0000
   45.0000         0  180.0000  135.0000  183.5649   66.7389         0         0
   45.0000         0  180.0000  180.0000  149.3473  149.3473         0   45.0000
   45.0000   45.0000         0         0   38.1838  334.6068         0   90.0000
   45.0000   45.0000         0   45.0000  -44.4247  300.3892         0  135.0000
   45.0000   45.0000         0   90.0000  -78.6422  217.7808         0  180.0000
   45.0000   45.0000         0  135.0000  -44.4247  135.1723         0 -135.0000
   45.0000   45.0000         0  180.0000   38.1838  100.9548         0  -90.0000
   45.0000   45.0000   45.0000         0  -93.6443  280.0018         0  135.0000
   45.0000   45.0000   45.0000   45.0000 -127.8618  197.3933         0  180.0000
   45.0000   45.0000   45.0000   90.0000  -93.6443  114.7849         0 -135.0000
```





User can choose How Many Movement & Simulate it

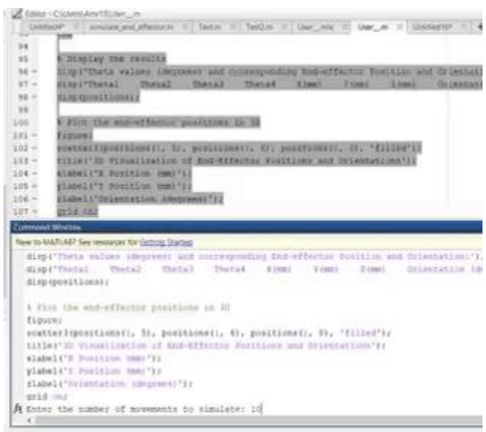
Code:

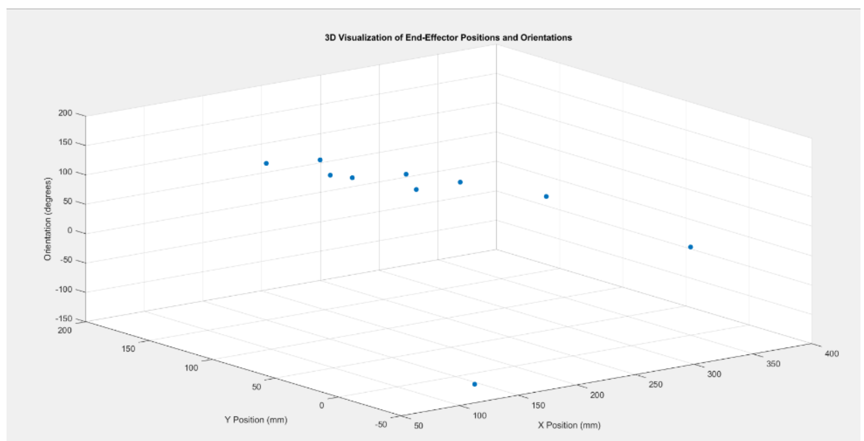
```
% Define the link lengths
a1 = 54.00;   % in mm
a2 = 109.990; % in mm
a3 = 69.607;  % in mm
a4 = 116.826; % in mm

% Symbolic joint angles
theta = sym('theta', [1 4]);  % symbolic joint angles
d = [0, 0, 0, 0];  % for planar manipulator
a = [a1, a2, a3, a4];  % link lengths
alpha = [0, 0, 0, 0];  % for planar manipulator

% Construct transformation matrices using DH parameters
T = cell(1, 4);
for i = 1:4
  T{i} = [cos(theta(i)), -sin(theta(i)), 0, a(i)*cos(theta(i));
       sin(theta(i)), cos(theta(i)), 0, a(i)*sin(theta(i));
       0, 0, 1, d(i);
       0, 0, 0, 1];
end

% Define a function for the full transformation from base to end-effector
full_transformation = @(th) double(subs(T{1}, theta(1), th(1)) * ...
                    subs(T{2}, theta(2), th(2)) * ...
                    subs(T{3}, theta(3), th(3)) * ...
                    subs(T{4}, theta(4), th(4)));

% Get the number of movements to simulate from the user
num_movements = input('Enter the number of movements to simulate: ');

% Define a range of angles for simulation (0 to 180 degrees -> 0 to pi radians)
theta1_range = linspace(0, pi, 5);  % range for theta1
theta2_range = linspace(0, pi, 5);  % range for theta2
theta3_range = linspace(0, pi, 5);  % range for theta3
theta4_range = linspace(0, pi, 5);  % range for theta4

% Initialize a matrix to store end-effector positions and orientations
positions = [];

% Counter for the number of simulated movements
count = 0;

% Iterate through the angle ranges and compute positions
```

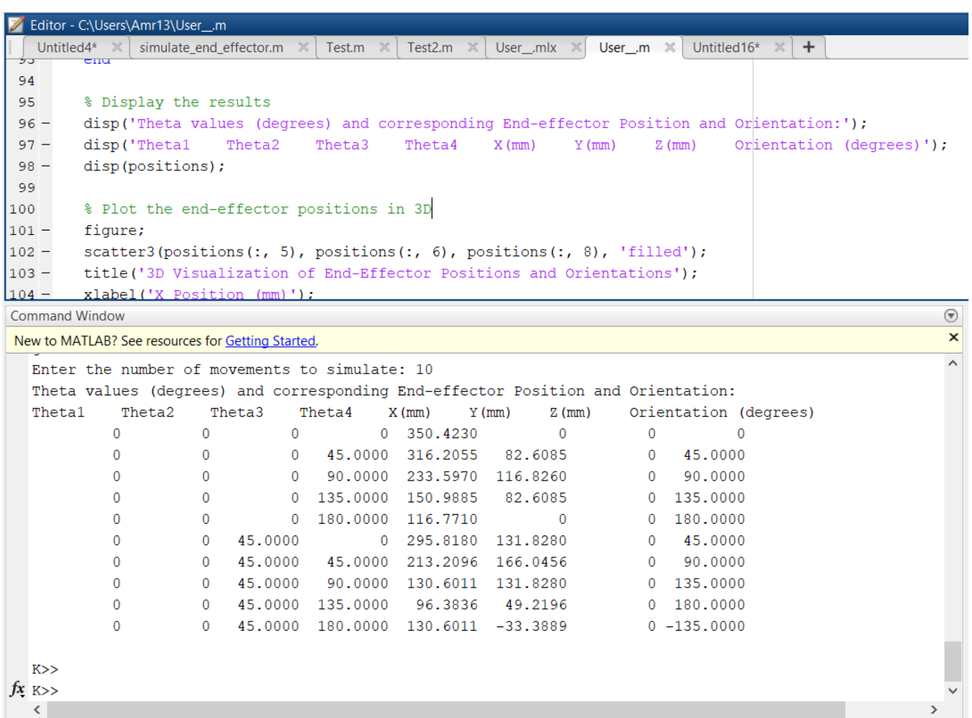
```
for th1 = theta1_range
  for th2 = theta2_range
    for th3 = theta3_range
      for th4 = theta4_range
        if count >= num_movements
          break;  % Exit loop if the desired number of movements is reached
        end

        theta_values_rad = [th1, th2, th3, th4];
        theta_values_deg = rad2deg(theta_values_rad);  % Convert to degrees

        % Compute the full transformation matrix for the current angles
        T_full = full_transformation(theta_values_rad);
        position_values = T_full(1:3, 4);
        orientation_value = atan2d(T_full(2, 1), T_full(1, 1));

        % Store the positions and orientations
        positions = [positions; theta_values_deg, position_values', orientation_value];

        % Plot the manipulator in 3D space
        figure(1); clf; hold on;
        plot3([0, T_full(1, 4)], [0, T_full(2, 4)], [0, T_full(3, 4)], 'k-o', 'LineWidth', 2);
        plot3(T_full(1, 4), T_full(2, 4), T_full(3, 4), 'ro', 'MarkerSize', 10, 'MarkerFaceColor', 'r');
        title(sprintf('End-Effector Position: [%.2f, %.2f, %.2f] mm', position_values));
        xlabel('X Position (mm)');
        ylabel('Y Position (mm)');
        zlabel('Z Position (mm)');
        grid on;
        axis equal;
        xlim([-400, 400]);
        ylim([-400, 400]);
        zlim([-400, 400]);
        drawnow;

        count = count + 1;  % Increment the counter

        pause(1);  % Pause for 1 second between movements
      end
      if count >= num_movements
        break;  % Exit loop if the desired number of movements is reached
      end
    end
    if count >= num_movements
      break;  % Exit loop if the desired number of movements is reached
    end
  end
  if count >= num_movements
    break;  % Exit loop if the desired number of movements is reached
```

```
    end
end

% Display the results
disp('Theta values (degrees) and corresponding End-effector Position and Orientation:');
disp('Theta1   Theta2   Theta3   Theta4   X(mm)   Y(mm)   Z(mm)   Orientation (degrees)');
disp(positions);

% Plot the end-effector positions in 3D
figure;
scatter3(positions(:, 5), positions(:, 6), positions(:, 8), 'filled');
title('3D Visualization of End-Effector Positions and Orientations');
xlabel('X Position (mm)');
ylabel('Y Position (mm)');
zlabel('Orientation (degrees)');
grid on;
```

EX: As we see 10 movements Simulation with Theta and X,Y,Z for End Effector

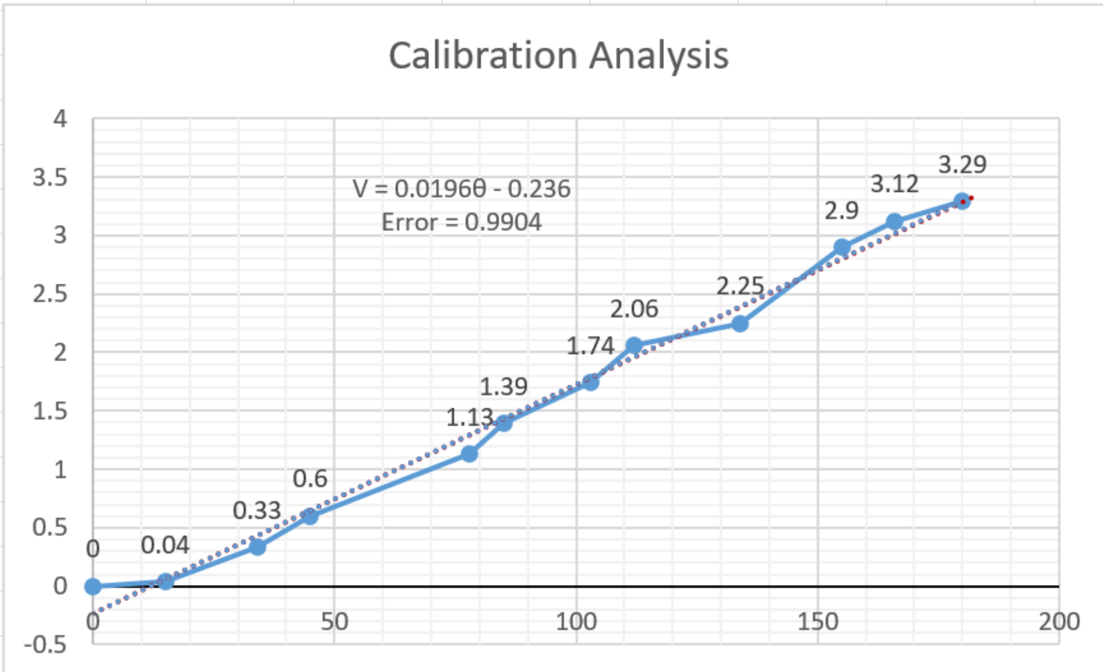






Calibration Analysis

Angel	Volte
0	0
15	0.04
34	0.33
45	0.6
78	1.13
85	1.39
103	1.74
112	2.06
134	2.25
155	2.9
166	3.12
180	3.29



So, we get a relationship between voltage and angel as you see in case we adjust a volte then we can calculate the angle of its movement.

- Error: 0.9904