ROBOTICS AND AUTOMATION

EC 6090

MINI PROJECT

GROUP 10

2020/E/100

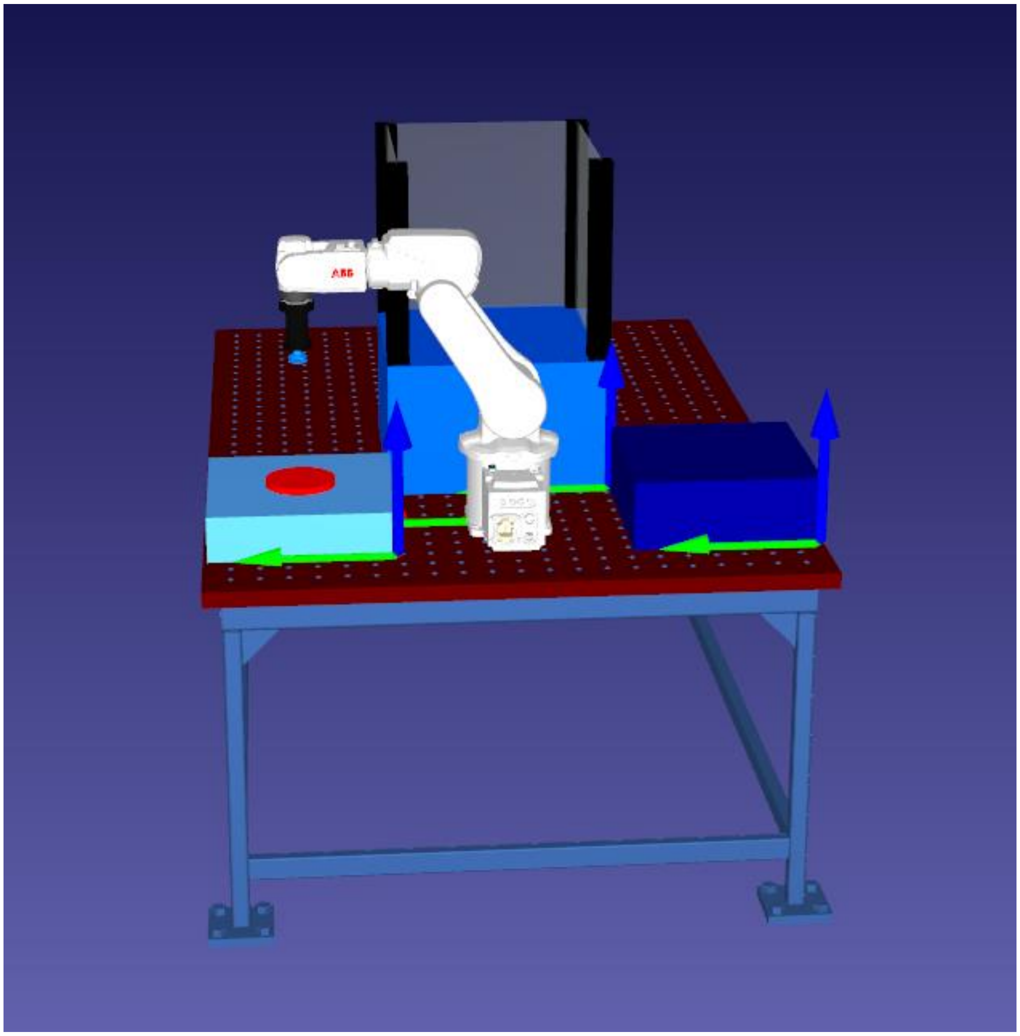
2020/E/105

2020/E/106

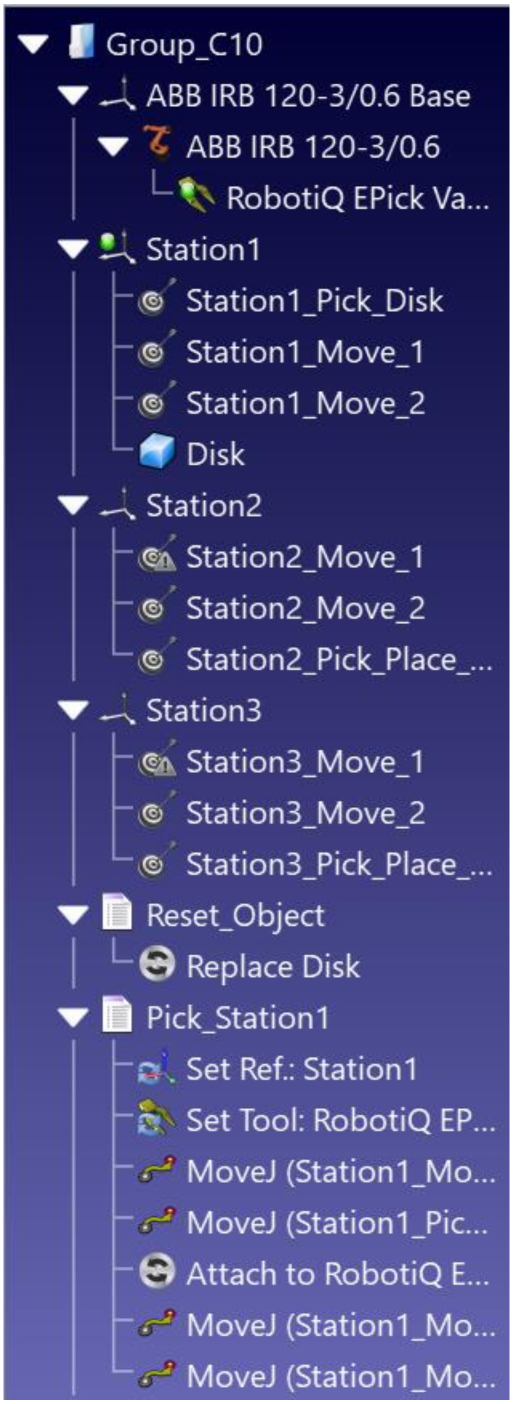
2020/E/114

2020/E/117

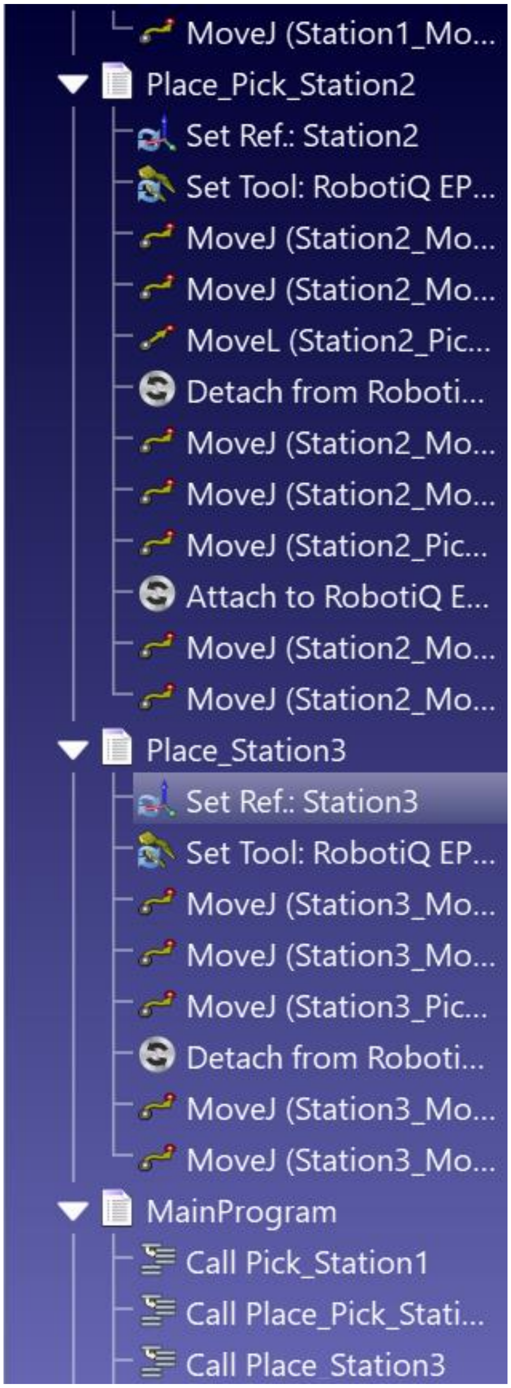
28 AUGUST 2024



ABB



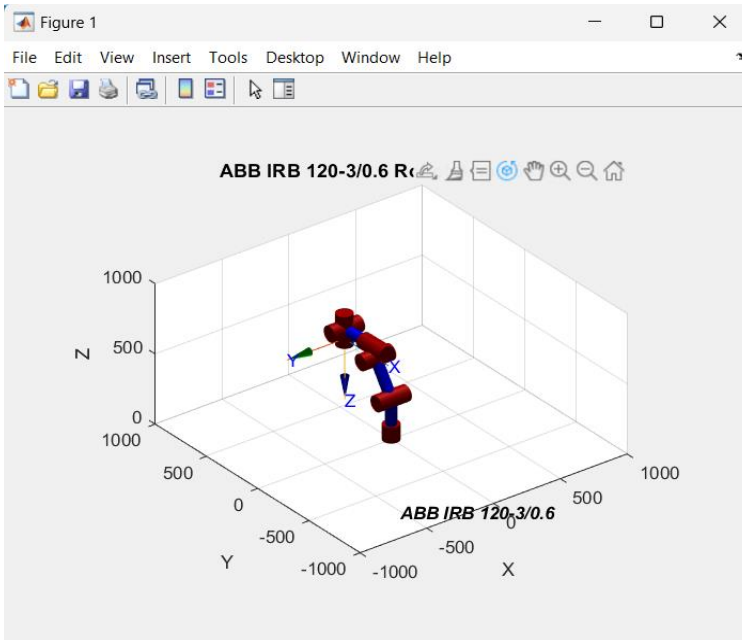
```
Group_C10
  ABB IRB 120-3/0.6 Base
    ABB IRB 120-3/0.6
      RobotiQ EPick Va...
  Station1
    Station1_Pick_Disk
    Station1_Move_1
    Station1_Move_2
    Disk
  Station2
    Station2_Move_1
    Station2_Move_2
    Station2_Pick_Place_...
  Station3
    Station3_Move_1
    Station3_Move_2
    Station3_Pick_Place_...
  Reset_Object
    Replace Disk
  Pick_Station1
    Set Ref.: Station1
    Set Tool: RobotiQ EP...
    MoveJ (Station1_Mo...
    MoveJ (Station1_Pic...
    Attach to RobotiQ E...
    MoveJ (Station1_Mo...
    MoveJ (Station1_Mo...
```

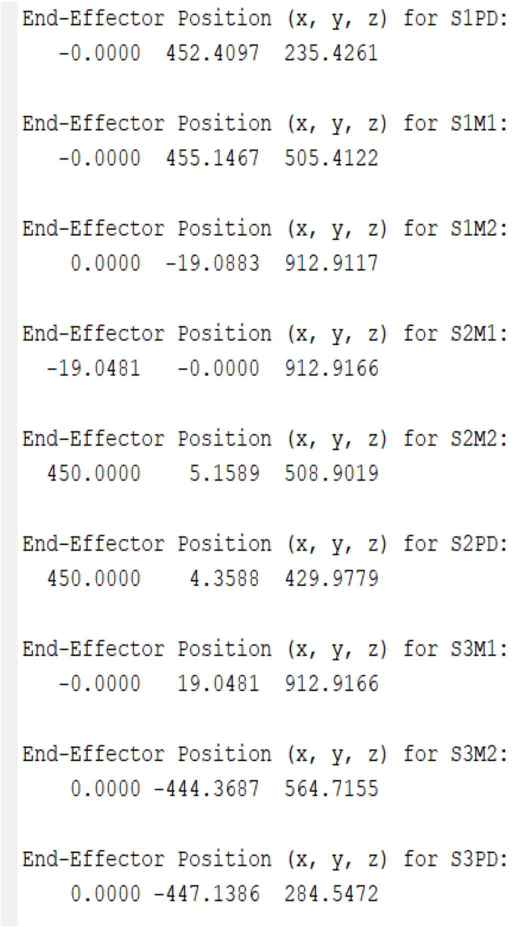


```
    MoveJ (Station1_Mo...
  Place_Pick_Station2
    Set Ref.: Station2
    Set Tool: RobotiQ EP...
    MoveJ (Station2_Mo...
    MoveJ (Station2_Mo...
    MoveL (Station2_Pic...
    Detach from Roboti...
    MoveJ (Station2_Mo...
    MoveJ (Station2_Mo...
    MoveJ (Station2_Pic...
    Attach to RobotiQ E...
    MoveJ (Station2_Mo...
    MoveJ (Station2_Mo...
  Place_Station3
    Set Ref.: Station3
    Set Tool: RobotiQ EP...
    MoveJ (Station3_Mo...
    MoveJ (Station3_Mo...
    MoveJ (Station3_Pic...
    Detach from Roboti...
    MoveJ (Station3_Mo...
    MoveJ (Station3_Mo...
  MainProgram
    Call Pick_Station1
    Call Place_Pick_Stati...
    Call Place_Station3
```

1. FORWARD KINEMATICS

```
% Define the DH parameters for the ABB IRB 120-3/0.6
L(1) = Link('d', 290,  'a', 0,     'alpha', pi/2);  % Joint
1
L(2) = Link('d', 0,    'a', -270,  'alpha', 0);     % Joint
2
L(3) = Link('d', 0,    'a', -70,   'alpha', pi/2);  % Joint
3
L(4) = Link('d', 302,  'a', 0,     'alpha', pi/2);  % Joint
4
L(5) = Link('d', 0,    'a', 0,     'alpha', pi/2);  % Joint
5
L(6) = Link('d', 130,  'a', 0,     'alpha', 0);     %
Joint 6

% Create the robot model
IRB120 = SerialLink(L, 'name', 'ABB IRB 120-3/0.6');

% Define joint angle adjustment (if any)
adjustment = [180 -90 0 180 180 -0];  % No adjustment
needed based on provided DH parameters

% Define joint angles for different positions
joint_angles = [
    90.000000, 45.958063, 0.408965, 0.000000, 44.213781, -
0.000000;  % S1PD
    90.000000, 34.442298, -33.446424, 0.000000, 89.584935,
-0.000000;  % S1M1
   90.000000, -13.954059, -63.899943, -0.000000, 32.854001,
0.000000;  % S1M2
    0.000000, -13.950000, -63.900000, -0.000000, 32.850000,
-0.000000;  % S2M1
    0.749740, 32.928022, -31.843716, -0.580862, 88.923349,
0.760693;  % S2M2
    0.647886, 31.376326, -14.897182, -0.605628, 73.528294,
0.819646;  % S2PD
    -90.000000, -13.950000, -63.900000, -0.000000,
32.850000, -0.000000;  % S3M1
    -90.000000, 37.105744, -49.130470, 0.000000,
101.474917, -0.000000;  % S3M2
    -90.000000, 39.305162, 1.085147, 0.000000, 49.028883, -
0.000000;  % S3PD
];

% Define corresponding position names
position_names = {'S1PD', 'S1M1', 'S1M2', 'S2M1', 'S2M2',
'S2PD', 'S3M1', 'S3M2', 'S3PD'};

% Plot and display each position with a pause
for i = 1:size(joint_angles, 1)
    plot_robot(IRB120, joint_angles(i, :),
position_names{i}, adjustment);
    pause(2);  % Pause for 2 seconds between each plot
end

% Function to plot robot and display end-effector position
function plot_robot(robot, joint_angles, position_name,
adjustment)
    % Adjust joint angles
    q = deg2rad(joint_angles + adjustment);
    % Compute forward kinematics
    P = robot.fkine(q);
    % Display the translation part of the transformation
matrix
    disp(['End-Effector Position (x, y, z) for ',
position_name, ':']);
    disp(P.t');
    % Plot the robot in the specified pose
    robot.plot(q);
    % Set axis limits dynamically based on the robot's
workspace
    axis([-1000 1000 -1000 1000 0 1000]);
    title(['ABB IRB 120-3/0.6 Robot Arm - ',
position_name]);
end
```





```
End-Effector Position (x, y, z) for S1PD:
   -0.0000  452.4097  235.4261

End-Effector Position (x, y, z) for S1M1:
   -0.0000  455.1467  505.4122

End-Effector Position (x, y, z) for S1M2:
    0.0000  -19.0883  912.9117

End-Effector Position (x, y, z) for S2M1:
  -19.0481   -0.0000  912.9166

End-Effector Position (x, y, z) for S2M2:
  450.0000    5.1589  508.9019

End-Effector Position (x, y, z) for S2PD:
  450.0000    4.3588  429.9779

End-Effector Position (x, y, z) for S3M1:
   -0.0000   19.0481  912.9166

End-Effector Position (x, y, z) for S3M2:
    0.0000 -444.3687  564.7155

End-Effector Position (x, y, z) for S3PD:
    0.0000 -447.1386  284.5472
```

2. INVERSE KINEMATICS

```
clc
clear
% Define the DH parameters for the ABB IRB 120-3/0.6
L(1) = Link('d', 290,  'a', 0,     'alpha', pi/2);  % Joint
1
L(2) = Link('d', 0,    'a', -270,  'alpha', 0);     % Joint
2
L(3) = Link('d', 0,    'a', -70,   'alpha', pi/2);  % Joint
3
L(4) = Link('d', 302,  'a', 0,     'alpha', pi/2);  % Joint
4
L(5) = Link('d', 0,    'a', 0,     'alpha', pi/2);  % Joint
5
L(6) = Link('d', 130,   'a', 0,     'alpha', 0);     %
Joint 6
```

```
% Create the robot model
CRB1300 = SerialLink(L, 'name', 'ABB CRB 1300-11/0.9');

% Define target end-effector positions and orientations
targets = {
    transl(0, 450.99, 235.433) * trotz(-pi/2) * troty(0) *
trotx(pi);  % S1PD
    transl(450, 3.041, 429.985) * trotx(pi) * troty(0) *
trotz(pi);    % S2PD
    transl(0, -448.456, 284.554) * trotx(pi) * troty(0) *
trotz(pi);   % S3PD
    transl(0, 450.991, 505.433) * trotx(-pi/2) * troty(0) *
trotz(pi);   % S1M1
};

% Define corresponding target names
target_names = {'S1PD', 'S2PD', 'S3PD', 'S1M1'};

% Initialize array to store the end-effector positions
end_effector_positions = [];

% Solve inverse kinematics for each target
for i = 1:length(targets)
    % Perform inverse kinematics
    q_solution = CRB1300.ikine(targets{i}, 'mask', [1 1 1 0
0 0]);

    % Check if a valid solution is found
    if isempty(q_solution)
        warning(['No valid IK solution found for ',
target_names{i}]);
        continue;
    end

    % Plot the robot in the solution pose and display joint
angles
    plot_robot(CRB1300, q_solution, target_names{i});

    % Store the end-effector position for 3D trajectory
plotting
    T_computed = CRB1300.fkine(q_solution);
    end_effector_positions = [end_effector_positions;
T_computed.t'];
```

```
    % Display the computed end-effector position
    disp(['Computed End-Effector Position (x, y, z) for ',
target_names{i}, ':']);
    disp(T_computed.t');

    % Pause for 2 seconds to visualize
    pause(2);
end

% Plot the 3D trajectory of the end effector
figure;
plot3(end_effector_positions(:, 1),
end_effector_positions(:, 2), end_effector_positions(:, 3),
'b-o', 'LineWidth', 1.5);
grid on;
xlabel('X (mm)');
ylabel('Y (mm)');
zlabel('Z (mm)');
title('3D Trajectory of the ABB CRB 1300-11/0.9 End-
Effector');
axis([-1000 1000 -1000 1000 0 1000]);
view(3);  % Set view to 3D

% Function to plot robot and display joint angles
function plot_robot(robot, q_solution, target_name)
    % Convert joint angles to degrees
    q_deg = rad2deg(q_solution);

    % Display the joint angles in degrees
    disp(['Joint Angles (degrees) for ', target_name,
':']);
    disp(q_deg);

    % Plot the robot in the specified pose
    robot.plot(q_solution);

    % Set axis limits dynamically based on the robot's
workspace
    axis([-1000 1000 -1000 1000 0 1000]);
    title(['ABB CRB 1300-11/0.9 Robot Arm - ',
target_name]);
end
```

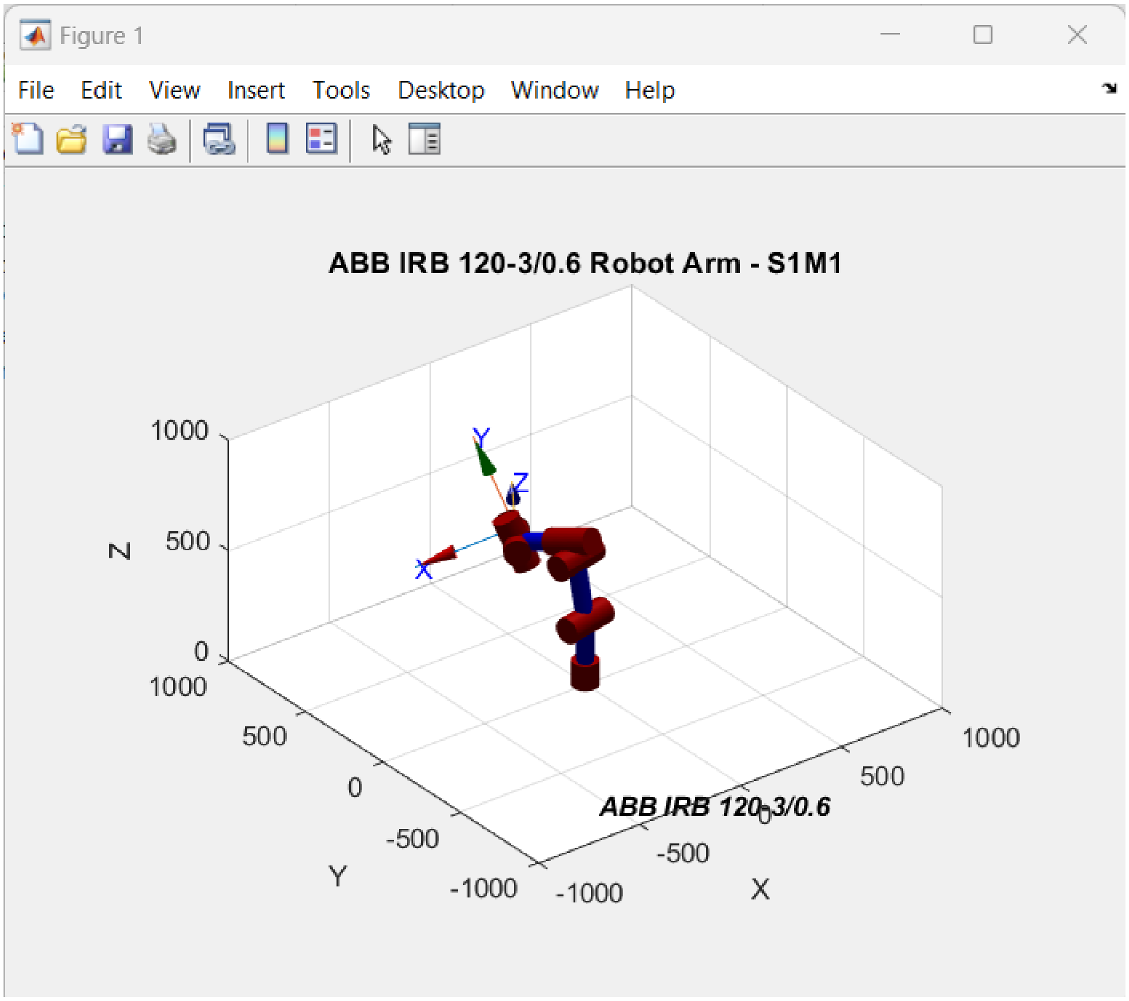


Figure 1
File Edit View Insert Tools Desktop Window Help
ABB IRB 120-3/0.6 Robot Arm - S1M1
Z
1000
500
0
Y
1000
500
0
-500
-1000
X
-1000
-500
0
500
1000
ABB IRB 120-3/0.6
Y
Z
X

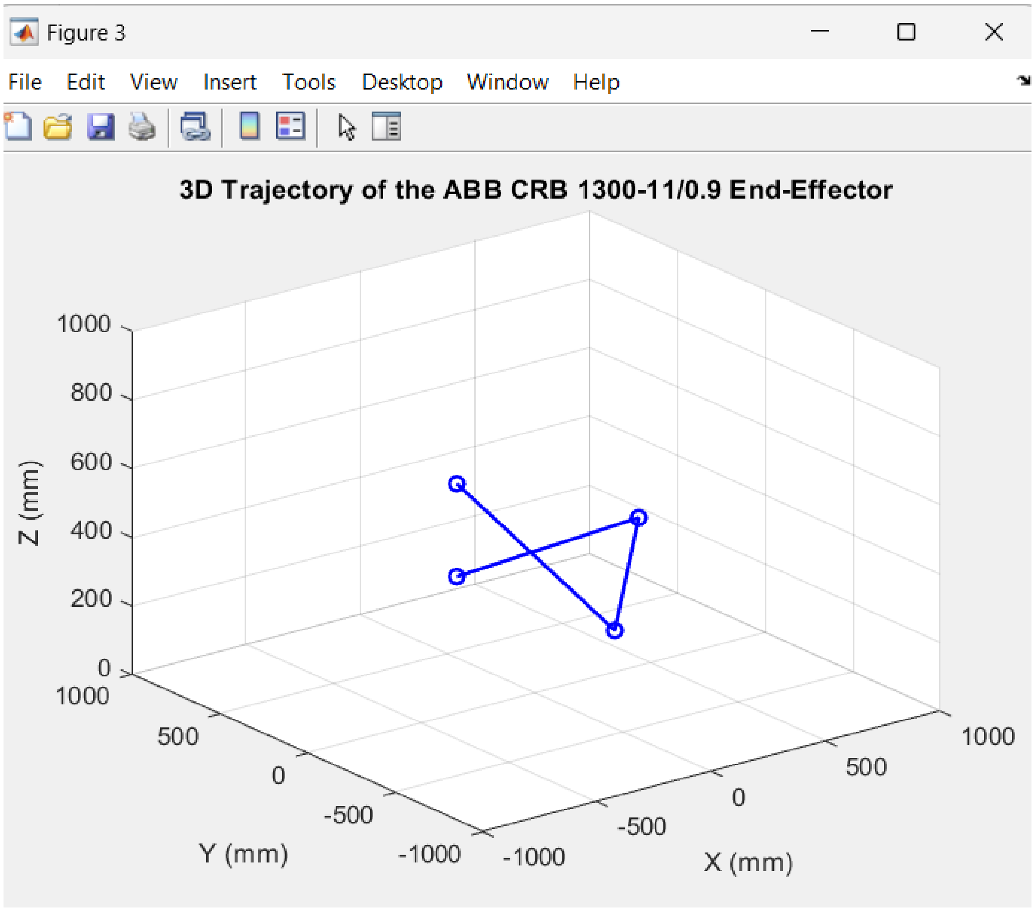
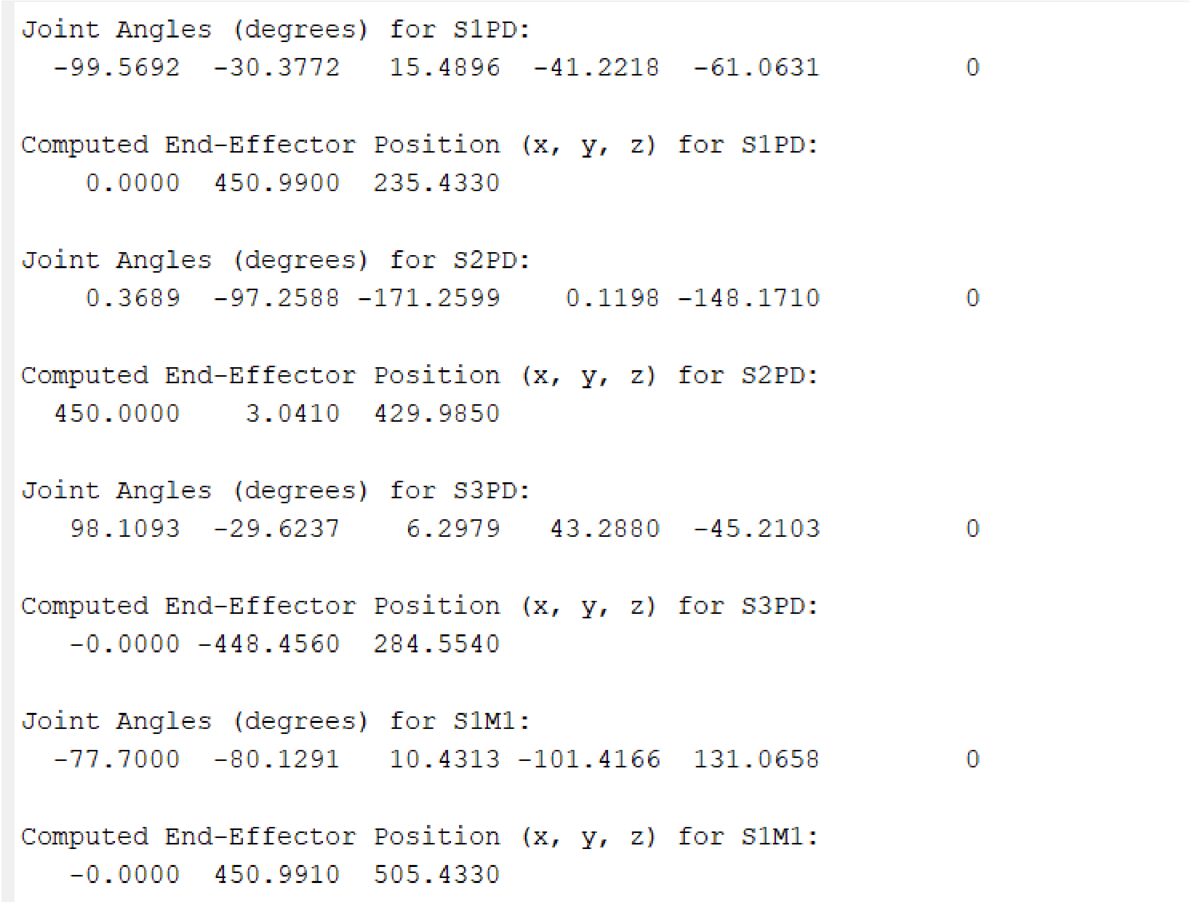


Figure 3
File Edit View Insert Tools Desktop Window Help
3D Trajectory of the ABB CRB 1300-11/0.9 End-Effector
Z (mm)
1000
800
600
400
200
0
Y (mm)
1000
500
0
-500
-1000
X (mm)
-1000
-500
0
500
1000



```
Joint Angles (degrees) for S1PD:
  -99.5692  -30.3772   15.4896  -41.2218  -61.0631         0

Computed End-Effector Position (x, y, z) for S1PD:
    0.0000  450.9900  235.4330

Joint Angles (degrees) for S2PD:
    0.3689  -97.2588 -171.2599    0.1198 -148.1710         0

Computed End-Effector Position (x, y, z) for S2PD:
  450.0000    3.0410  429.9850

Joint Angles (degrees) for S3PD:
   98.1093  -29.6237    6.2979   43.2880  -45.2103         0

Computed End-Effector Position (x, y, z) for S3PD:
   -0.0000 -448.4560  284.5540

Joint Angles (degrees) for S1M1:
  -77.7000  -80.1291   10.4313 -101.4166  131.0658         0

Computed End-Effector Position (x, y, z) for S1M1:
   -0.0000  450.9910  505.4330
```

3. TRAJECTORY

```
clc
clear
% Define the DH parameters for the ABB IRB 120-3/0.6
L(1) = Link('d', 290,  'a', 0,      'alpha', pi/2);  % Joint
1
L(2) = Link('d', 0,    'a', -270,  'alpha', 0);      % Joint
2
L(3) = Link('d', 0,    'a', -70,   'alpha', pi/2);  % Joint
3
L(4) = Link('d', 302,  'a', 0,      'alpha', pi/2);  % Joint
4
L(5) = Link('d', 0,    'a', 0,      'alpha', pi/2);  % Joint
5
L(6) = Link('d', 130,   'a', 0,      'alpha', 0);      %
Joint 6

% Create the robot model
```

```
CRB1300 = SerialLink(L, 'name', 'ABB IRB 120-3/0.6');

% Define joint angle adjustment
adjustment = [0 -90 0 0 0 180];

% Define joint angles for different positions
joint_angles = [
    90.000000, 45.958063, 0.408965, 0.000000, 44.213781, -
0.000000;  % S1PD
    90.000000, 34.442298, -33.446424, 0.000000, 89.584935,
-0.000000;  % S1M1
   90.000000, -13.954059, -63.899943, -0.000000, 32.854001,
0.000000;  % S1M2
    0.000000, -13.950000, -63.900000, -0.000000, 32.850000,
-0.000000;  % S2M1
    0.749740, 32.928022, -31.843716, -0.580862, 88.923349,
0.760693;  % S2M2
    0.647886, 31.376326, -14.897182, -0.605628, 73.528294,
0.819646;  % S2PD
    -90.000000, -13.950000, -63.900000, -0.000000,
32.850000, -0.000000;  % S3M1
    -90.000000, 37.105744, -49.130470, 0.000000,
101.474917, -0.000000;  % S3M2
    -90.000000, 39.305162, 1.085147, 0.000000, 49.028883, -
0.000000;  % S3PD
];


% Define corresponding position names
position_names = {'S1PD', 'S1M1', 'S1M2', 'S2M1', 'S2M2',
'S2PD', 'S3M1', 'S3M2', 'S3PD'};
% Calculate joint velocities
joint_velocities = zeros(size(joint_angles, 1) - 1,
size(joint_angles, 2));  % Initialize matrix for velocities
for i = 1:size(joint_angles, 1) - 1
    joint_velocities(i, :) = joint_angles(i+1, :) -
joint_angles(i, :);
end

% Plot and display each position with a pause
for i = 1:size(joint_angles, 1)
    plot_robot(CRB1300, joint_angles(i, :),
position_names{i}, adjustment);
    pause(2);  % Pause for 2 seconds between each plot
```

```
end

% Plot velocity diagram
figure;
num_positions = size(joint_angles, 1) - 1;
t = 1:num_positions;  % Time or position index for
velocities

% Plot for each joint in separate subplots
for j = 1:size(joint_velocities, 2)
    subplot(3, 2, j);  % Adjust the subplot grid if needed
    plot(t, joint_velocities(:, j), 'o-', 'LineWidth', 1.5,
'MarkerSize', 6);
    xlabel('Position Index');
    ylabel('Velocity (degrees/s)');
    title(['Joint ', num2str(j), ' Velocity']);
    grid on;
    legend(['Joint ', num2str(j)], 'Location', 'best');
end

sgtitle('Joint Velocities Over Different Positions');

% Function to plot robot and display end-effector position
function plot_robot(robot, joint_angles, position_name,
adjustment)
    % Adjust joint angles
    q = deg2rad(joint_angles + adjustment);
    % Compute forward kinematics
    P = robot.fkine(q);
    % Display the translation part of the transformation
matrix
    disp(['End-Effector Position (x, y, z) for ',
position_name, ':']);
    disp(P.t');
    % Plot the robot in the specified pose
    robot.plot(q);
    % Set axis limits dynamically based on the robot's
workspace
    axis([-1000 1000 -1000 1000 0 1000]);
    title(['ABB IRB 120-3/0.6 Robot Arm - ',
position_name]);
end
```

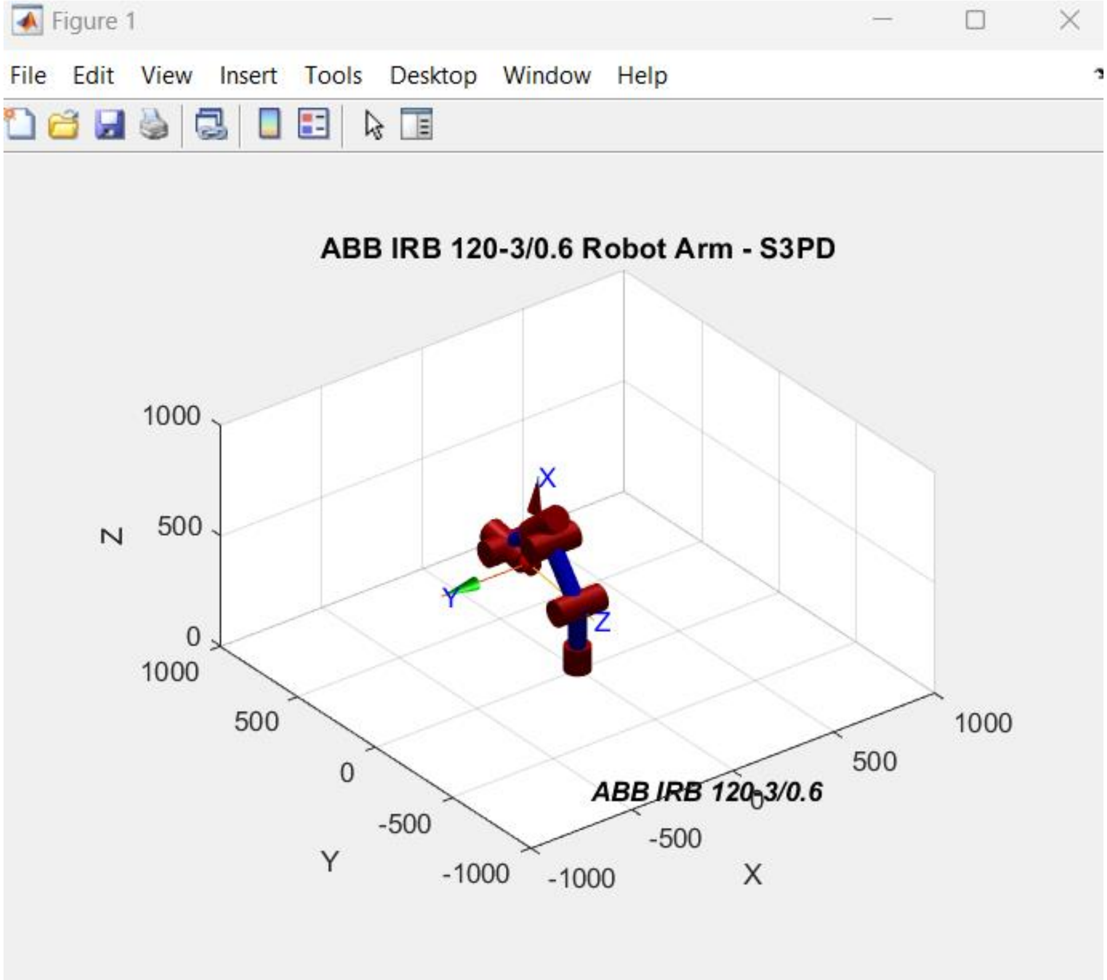


Figure 1
File Edit View Insert Tools Desktop Window Help
ABB IRB 120-3/0.6 Robot Arm - S3PD
Z
1000
500
0
Y
1000
500
0
-500
-1000
X
-1000
-500
0
500
1000
X
Y
Z
ABB IRB 120-3/0.6

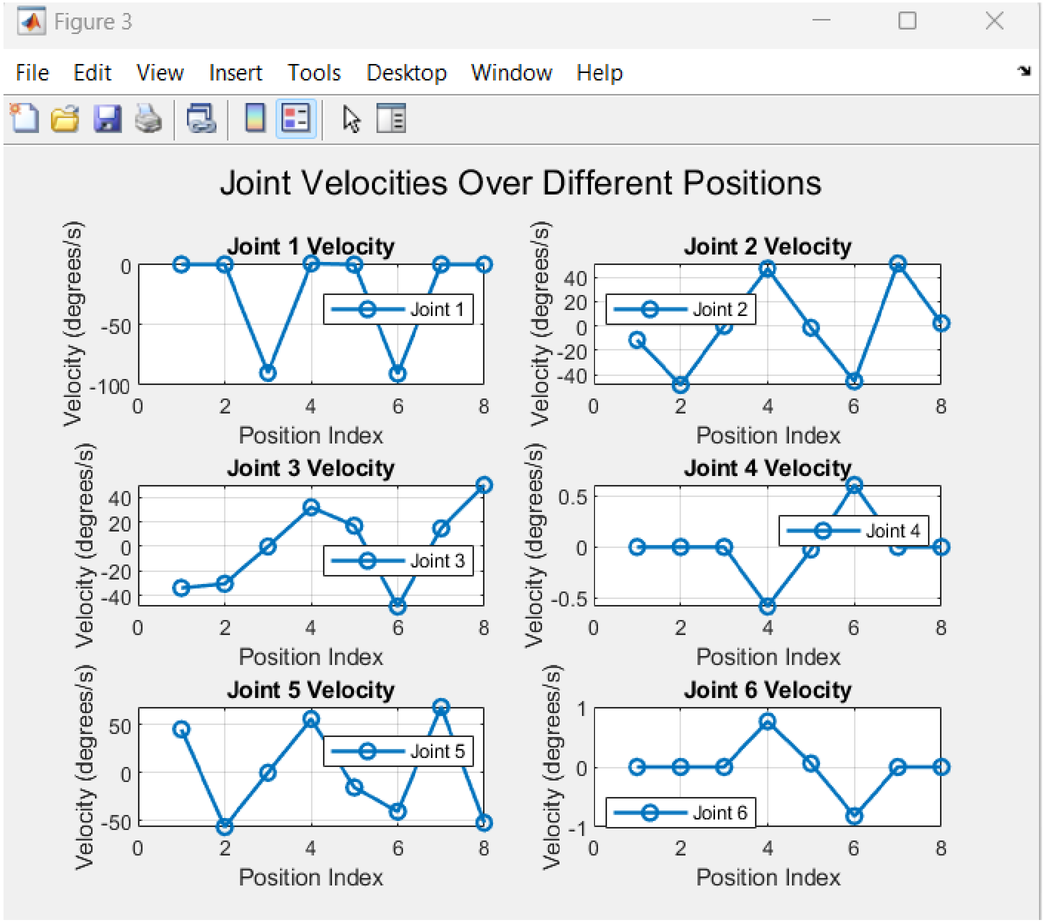


Figure 3
File Edit View Insert Tools Desktop Window Help
Joint Velocities Over Different Positions
Joint 1 Velocity
Velocity (degrees/s)
0
-50
-100
Joint 1
Position Index
0 2 4 6 8
Joint 2 Velocity
Velocity (degrees/s)
40
20
0
-20
-40
Joint 2
Position Index
0 2 4 6 8
Joint 3 Velocity
Velocity (degrees/s)
40
20
0
-20
-40
Joint 3
Position Index
0 2 4 6 8
Joint 4 Velocity
Velocity (degrees/s)
0.5
0
-0.5
Joint 4
Position Index
0 2 4 6 8
Joint 5 Velocity
Velocity (degrees/s)
50
0
-50
Joint 5
Position Index
0 2 4 6 8
Joint 6 Velocity
Velocity (degrees/s)
1
0
-1
Joint 6
Position Index
0 2 4 6 8

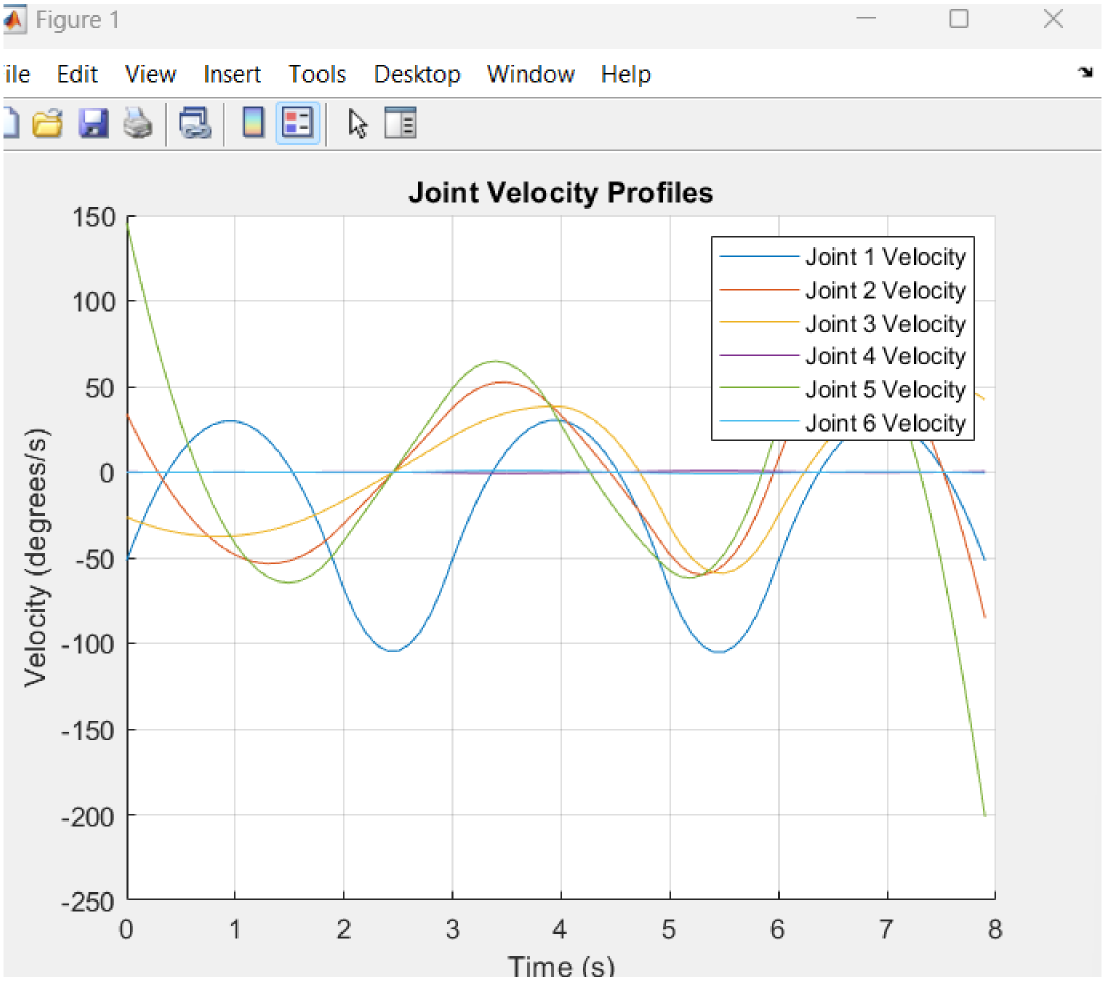
4. VELOCITY PROFILE

```
clc
clear
import robotics.*

% Define waypoints based on provided joint angles
waypoints = [
    90.000000, 45.958063, 0.408965, 0.000000, 44.213781, -
0.000000;  % S1PD
    90.000000, 34.442298, -33.446424, 0.000000, 89.584935,
-0.000000;  % S1M1
   90.000000, -13.954059, -63.899943, -0.000000, 32.854001,
0.000000;  % S1M2
    0.000000, -13.950000, -63.900000, -0.000000, 32.850000,
-0.000000;  % S2M1
    0.749740, 32.928022, -31.843716, -0.580862, 88.923349,
0.760693;  % S2M2
    0.647886, 31.376326, -14.897182, -0.605628, 73.528294,
0.819646;  % S2PD
    -90.000000, -13.950000, -63.900000, -0.000000,
32.850000, -0.000000;  % S3M1
    -90.000000, 37.105744, -49.130470, 0.000000,
101.474917, -0.000000;  % S3M2
    -90.000000, 39.305162, 1.085147, 0.000000, 49.028883, -
0.000000;  % S3PD
];


% Time vector (assuming 1 second between each waypoint)
t = 0:size(waypoints, 1) - 1;

% Generate trajectory using spline interpolation
tq = 0:0.1:t(end);
trajectory = zeros(numel(tq), 6);
for i = 1:6
    trajectory(:,i) = spline(t, waypoints(:,i), tq);
end

% Calculate velocities
dt = 0.1; % Time step
velocities = diff(trajectory) / dt;
```

```
% Adjust time vector for velocity
time_velocity = tq(1:end-1);

% Plot velocity profiles
figure;
hold on;
plot(time_velocity, velocities(:,1), 'DisplayName', 'Joint
1 Velocity');
plot(time_velocity, velocities(:,2), 'DisplayName', 'Joint
2 Velocity');
plot(time_velocity, velocities(:,3), 'DisplayName', 'Joint
3 Velocity');
plot(time_velocity, velocities(:,4), 'DisplayName', 'Joint
4 Velocity');
plot(time_velocity, velocities(:,5), 'DisplayName', 'Joint
5 Velocity');
plot(time_velocity, velocities(:,6), 'DisplayName', 'Joint
6 Velocity');
title('Joint Velocity Profiles');
xlabel('Time (s)');
ylabel('Velocity (degrees/s)');
legend('show');
grid on;

% Define DH parameters for the robot
% Replace these with the actual parameters for your robot
% Example parameters (not accurate, replace with real DH
parameters)
a = [0 0 0 0 0 0];
d = [0 0 0 0 0 0];
alpha = [0 0 0 0 0 0];

% Forward Kinematics Calculation
x_fk = zeros(size(tq));
y_fk = zeros(size(tq));
z_fk = zeros(size(tq));

for i = 1:length(tq)
    theta = trajectory(i, :);
    T = eye(4); % Initialize the transformation matrix

    % Compute the transformation matrix for each joint
    for j = 1:6
```

```
        A = [cosd(theta(j)) -sind(theta(j)) 0 a(j);
             sind(theta(j))*cosd(alpha(j))
cosd(theta(j))*cosd(alpha(j)) -sind(alpha(j)) -
sind(alpha(j))*d(j);
             sind(theta(j))*sind(alpha(j))
cosd(theta(j))*sind(alpha(j)) cosd(alpha(j))
cosd(alpha(j))*d(j);
             0 0 0 1];
        T = T * A; % Update the transformation matrix
    end

    % Extract position from transformation matrix
    x_fk(i) = T(1, 4);
    y_fk(i) = T(2, 4);
    z_fk(i) = T(3, 4);
end

% Inverse Kinematics (Example, replace with real IK
calculations)
x_inv = x_fk; % Placeholder, replace with actual inverse
kinematics calculations
y_inv = y_fk; % Placeholder
z_inv = z_fk; % Placeholder

% Plot joint angles
figure;
plot(tq, trajectory(:,1), tq, trajectory(:,2), tq,
trajectory(:,3), ...
     tq, trajectory(:,4), tq, trajectory(:,5), tq,
trajectory(:,6));
title('Joint Angles');
xlabel('Time (s)');
ylabel('Angle (degrees)');
legend('?1', '?2', '?3', '?4', '?5', '?6');
grid on;

% Compare forward and inverse kinematics results
figure;
subplot(3,1,1);
plot(tq, x_fk, tq, x_inv);
title('X Position');
xlabel('Time (s)');
ylabel('X (mm)');
legend('Forward', 'Inverse');
```

```
grid on;

subplot(3,1,2);
plot(tq, y_fk, tq, y_inv);
title('Y Position');
xlabel('Time (s)');
ylabel('Y (mm)');
legend('Forward', 'Inverse');
grid on;

subplot(3,1,3);
plot(tq, z_fk, tq, z_inv);
title('Z Position');
xlabel('Time (s)');
ylabel('Z (mm)');
legend('Forward', 'Inverse');
grid on;
```



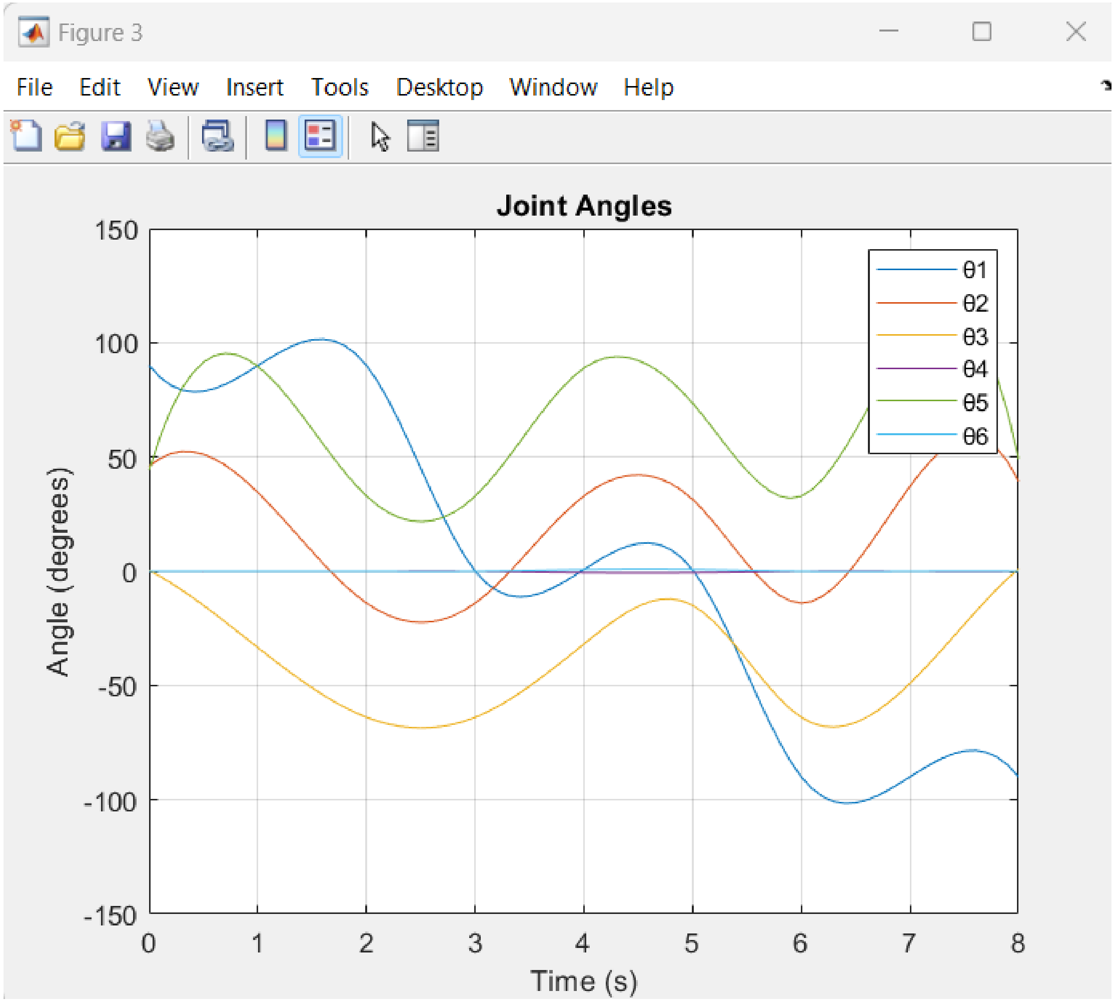


Figure 3
File Edit View Insert Tools Desktop Window Help
Joint Angles
θ1
θ2
θ3
θ4
θ5
θ6
150
100
50
0
-50
-100
-150
Angle (degrees)
0
1
2
3
4
5
6
7
8
Time (s)