

# **ROBOT CONFIGURATIONS AND WORK ENVELOPES USING MATLAB'S ROBOTICS SYSTEM TOOLBOX**

Assignment 2

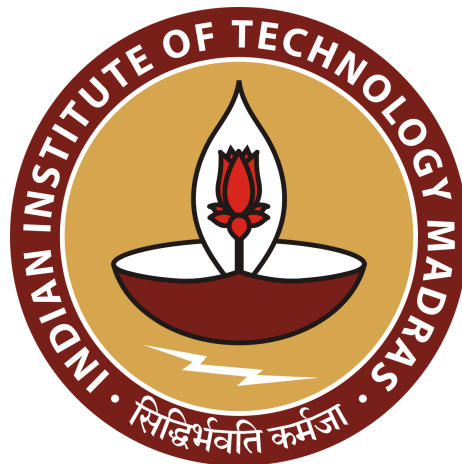
Course: ME3302 (Automation in Manufacturing)

Submitted by

**VISHAL V  
(ME20B204)**

*Under the guidance of*

DR. SOMASHEKHAR S. HIREMATH



**DEPARTMENT OF MECHANICAL ENGINEERING  
INDIAN INSTITUTE OF TECHNOLOGY, MADRAS**

**APRIL 2023**

## **INTRODUCTION TO ROBOTICS**

Robotics is an interdisciplinary field that integrates elements from mechanical engineering, electrical engineering, computer science, and other disciplines. It deals with the design, construction, operation, and application of robots, which are machines capable of performing complex tasks autonomously or semi-autonomously. Robotics has made significant advancements over the years, impacting a variety of industries, including manufacturing, medical, agriculture, service, and logistics, among others.

## **DIFFERENT ROBOT CONFIGURATIONS**

### **1. Cartesian Robot**

The Cartesian robot, also known as a linear or gantry robot, employs three linear actuators along the X, Y, and Z axes to achieve motion. This configuration is characterized by its rectangular work envelope, with sides parallel to the X, Y, and Z axes. Cartesian robots are often used for tasks that require precise linear movements, such as pick-and-place operations or 3D printing.

### **2. Cylindrical Robot**

The cylindrical robot utilizes a rotary actuator for rotation about a vertical axis and a linear actuator for motion along the vertical axis. It operates within a cylindrical work envelope with a circular base and a height defined by the linear actuator's range. Cylindrical robots are commonly used in assembly lines, material handling, and packaging applications.

### **3. Spherical Robot**

The spherical robot employs three rotary actuators to achieve motion, creating a spherical work envelope defined by the range of motion of these actuators. Spherical robots, also known as polar robots, can access points within a ball-shaped volume in three-dimensional space. These robots are often used for tasks that require a wide range of motion, such as welding or painting.

### **4. SCARA Robot**

The SCARA (Selective Compliance Assembly Robot Arm) robot features two rotary joints for motion in the X-Y plane and a linear joint for motion along the Z-axis. The work envelope of a SCARA robot is crescent-shaped, defined by the range of motion of its rotary and linear joints. SCARA robots excel in high-speed, high-precision assembly and pick-and-place operations.

### **5. Articulated Robot**

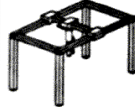
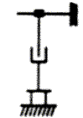

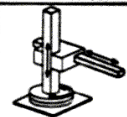


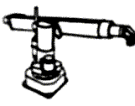


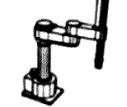
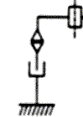




The articulated robot uses multiple rotary joints to achieve motion, resulting in a complex work envelope defined by the range of motion of its joints. The envelope varies depending on the number of joints and their arrangement. Articulated robots, also known as serial manipulators, are highly versatile and can be found in applications ranging from automotive assembly to medical surgeries.

## WORK ENVELOPES

A work envelope, or robot's workspace, is the spatial region a robot can reach with its end-effector. Understanding a robot's work envelope is vital for design, assessing capabilities, and determining suitability for specific tasks. Work envelopes differ among robot configurations, each with unique shapes and ranges of motion for specific applications.

Key factors to consider when evaluating a robot's work envelope include:

1. Reach: The maximum distance the end-effector can extend from the robot's base.
2. Payload capacity: The maximum weight the robot can handle, including the tool or gripper and the manipulated object.
3. Accuracy and repeatability: The robot's ability to position its end-effector at a specific point and return to the same position consistently.
4. Obstacles and safety considerations: The presence of obstacles or equipment within the robot's workspace and maintaining a safe distance from human operators.
5. Kinematic singularities: Certain joint configurations may result in loss of degrees of freedom, affecting the robot's motion capabilities.

Principle	Kinematic Structure	Workspace
 Cartesian Robot		
 Cylindrical Robot		
 Spherical Robot		
 SCARA Robot		
 Articulated Robot		

By analyzing and optimizing the work envelope, engineers can ensure that a robot is well-suited for its intended application and performs tasks effectively and safely.

## MATLAB ROBOTICS SYSTEM TOOLBOX

MATLAB, developed by MathWorks, is a powerful numerical computing environment widely used for developing robotic applications. Its strength lies in its ability to perform complex mathematical calculations, offer a user-friendly interface, and provide extensive libraries for various domains.

The Robotics System Toolbox is a collection of functions and tools for MATLAB that allows users to model, simulate, and analyze robotic systems. By providing support for various robotic manipulators and tools for kinematics, dynamics, path planning, and trajectory generation, the toolbox plays a crucial role in the design and analysis of robotic systems, including work envelope analysis.

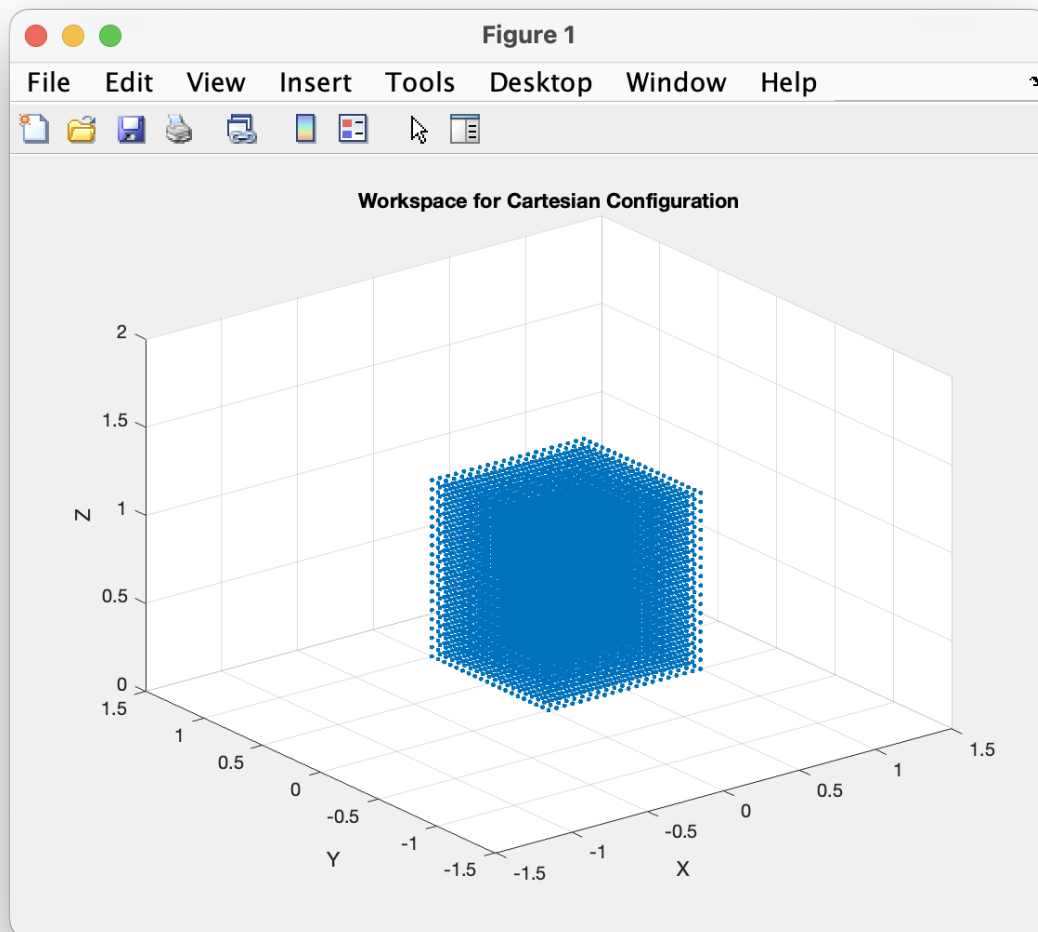
## 1. Cartesian Robot Configuration

### MATLAB Code:

```
% Import robotics system toolbox
import robotics.*;
% Create a rigid body tree
robot = robotics.RigidBodyTree;
% Add the first link (link1) and joint (joint1)
link1 = robotics.RigidBody('link1');
joint1 = robotics.Joint('joint1', 'prismatic');
joint1.HomePosition = 0;
setFixedTransform(joint1, trvec2tform([0, 0, 0]));
link1.Joint = joint1;
addBody(robot, link1, 'base');
% Add the second link (link2) and joint (joint2)
link2 = robotics.RigidBody('link2');
joint2 = robotics.Joint('joint2', 'prismatic');
joint2.HomePosition = 0;
setFixedTransform(joint2, trvec2tform([0, 0, 0])); % Fixed transformation corrected
link2.Joint = joint2;
addBody(robot, link2, 'link1');
% Add the third link (link3) and joint (joint3)
link3 = robotics.RigidBody('link3');
joint3 = robotics.Joint('joint3', 'prismatic');
joint3.HomePosition = 0;
setFixedTransform(joint3, trvec2tform([0, 0, 0])); % Fixed transformation corrected
link3.Joint = joint3;
addBody(robot, link3, 'link2');
% Set joint limits
joint1_range = linspace(0, 1, 20);
joint2_range = linspace(0, 1, 20);
joint3_range = linspace(0, 1, 20);
% Initialize an empty matrix to store reachable points
reachable_points = [];
% Loop through all possible joint configurations
for q1 = joint1_range
    for q2 = joint2_range
        for q3 = joint3_range
            % Compute forward kinematics for Cartesian configuration
            position = [q1, q2, q3];
            % Store the reachable points
            reachable_points = [reachable_points; position];
        end
    end
end
% Plot the reachable points in 3D
scatter3(reachable_points(:, 1), reachable_points(:, 2), reachable_points(:, 3),
```

```
'.'');  
xlabel('X');  
ylabel('Y');  
zlabel('Z');  
title('Workspace for Cartesian Configuration');  
axis([-1.5 1.5 -1.5 1.5 0 2]); % Set axis limits for X, Y, and Z
```

### Work Envelope:



## 2. Cylindrical Robot Configuration

### MATLAB Code:

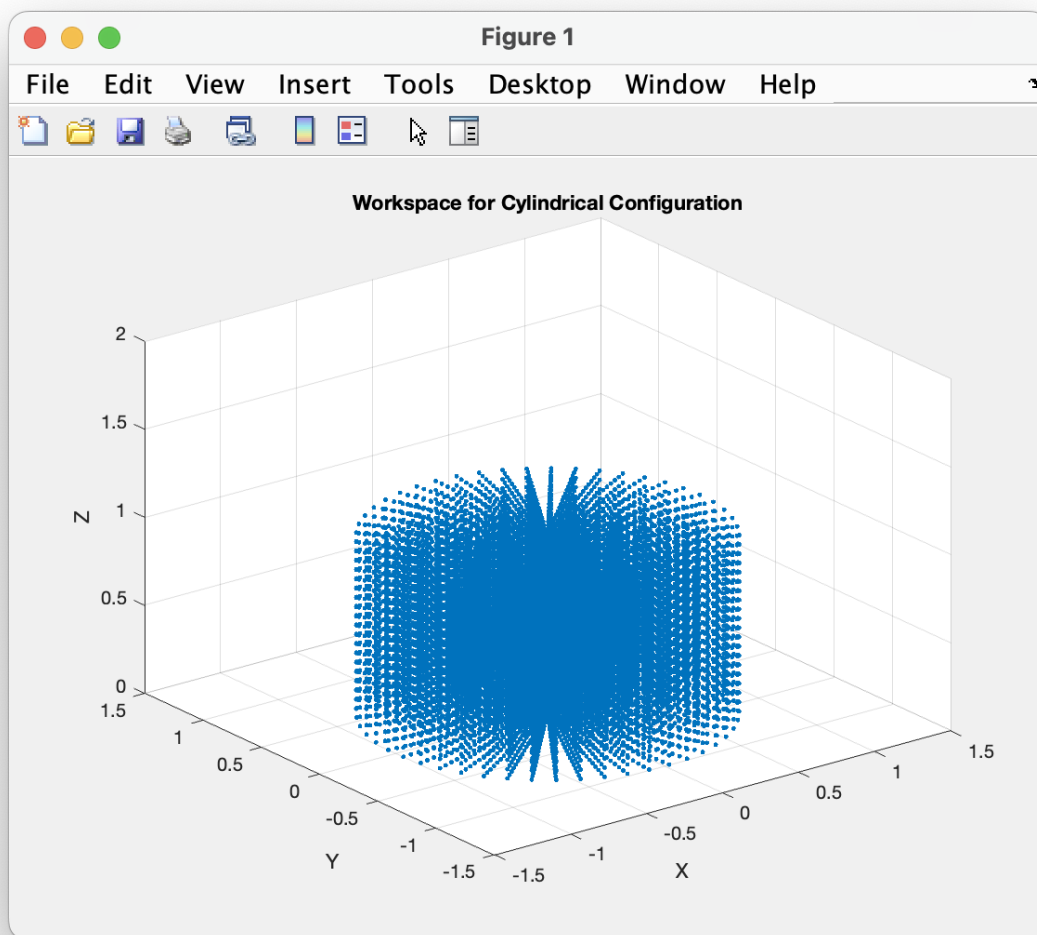
```
% Import robotics system toolbox
import robotics.*;
% Create a rigid body tree
robot = robotics.RigidBodyTree;
% Add the first link (link1) and joint (joint1)
link1 = robotics.RigidBody('link1');
joint1 = robotics.Joint('joint1', 'revolute');
joint1.HomePosition = 0;
setFixedTransform(joint1, eye(4));
link1.Joint = joint1;
addBody(robot, link1, 'base');
% Add the second link (link2) and joint (joint2)
link2 = robotics.RigidBody('link2');
joint2 = robotics.Joint('joint2', 'prismatic');
joint2.HomePosition = 0;
setFixedTransform(joint2, eye(4));
link2.Joint = joint2;
addBody(robot, link2, 'link1');
% Add the third link (link3) and joint (joint3)
link3 = robotics.RigidBody('link3');
joint3 = robotics.Joint('joint3', 'prismatic');
joint3.HomePosition = 0;
setFixedTransform(joint3, eye(4));
link3.Joint = joint3;
addBody(robot, link3, 'link2');
% Set joint limits
theta_range = linspace(0, 2*pi, 50); % Revolute joint angle (0 to 2*pi)
z_range = linspace(0, 1, 20); % Prismatic joint position (0 to 1)
r_range = linspace(0, 1, 20); % Radial prismatic joint position (0 to 1)
% Initialize an empty matrix to store reachable points
reachable_points = [];
% Loop through all possible joint configurations
for theta = theta_range
    for z = z_range
        for r = r_range
            % Compute forward kinematics for cylindrical configuration
            x = r * cos(theta);
            y = r * sin(theta);
            position = [x, y, z];
            % Store the reachable points
            reachable_points = [reachable_points; position];
        end
    end
end
```

```

% Plot the reachable points in 3D
scatter3(reachable_points(:, 1), reachable_points(:, 2), reachable_points(:, 3),
'.'');
xlabel('X');
ylabel('Y');
zlabel('Z');
title('Workspace for Cylindrical Configuration');
axis([-1.5 1.5 -1.5 1.5 0 2]); % Set axis limits for X, Y, and Z

```

## Work Envelope:



### 3. Spherical Robot Configuration

#### MATLAB Code:

```
% Import robotics system toolbox
import robotics.*;
% Create a rigid body tree
robot = robotics.RigidBodyTree;
% Add the first link (link1) and joint (joint1)
link1 = robotics.RigidBody('link1');
joint1 = robotics.Joint('joint1', 'revolute');
joint1.HomePosition = 0;
setFixedTransform(joint1, eye(4));
link1.Joint = joint1;
addBody(robot, link1, 'base');
% Add the second link (link2) and joint (joint2)
link2 = robotics.RigidBody('link2');
joint2 = robotics.Joint('joint2', 'revolute');
joint2.HomePosition = 0;
setFixedTransform(joint2, axang2tform([1 0 0 pi/2]));
link2.Joint = joint2;
addBody(robot, link2, 'link1');
% Add the third link (link3) and joint (joint3)
link3 = robotics.RigidBody('link3');
joint3 = robotics.Joint('joint3', 'prismatic');
joint3.HomePosition = 0;
setFixedTransform(joint3, eye(4));
link3.Joint = joint3;
addBody(robot, link3, 'link2');
% Set joint limits
theta1_range = linspace(0, 2*pi, 50); % Revolute joint angle (0 to 2*pi)
theta2_range = linspace(-pi/2, pi/2, 50); % Revolute joint angle (-pi/2 to pi/2)
z_range = linspace(0, 1, 20); % Prismatic joint position (0 to 1)
% Initialize an empty matrix to store reachable points
reachable_points = [];
% Loop through all possible joint configurations
for theta1 = theta1_range
    for theta2 = theta2_range
        for z = z_range
            % Compute forward kinematics for RRP configuration
            x = cos(theta1) * (1 + z * cos(theta2));
            y = sin(theta1) * (1 + z * cos(theta2));
            position = [x, y, z * sin(theta2)];
            % Store the reachable points
            reachable_points = [reachable_points; position];
        end
    end
end
```

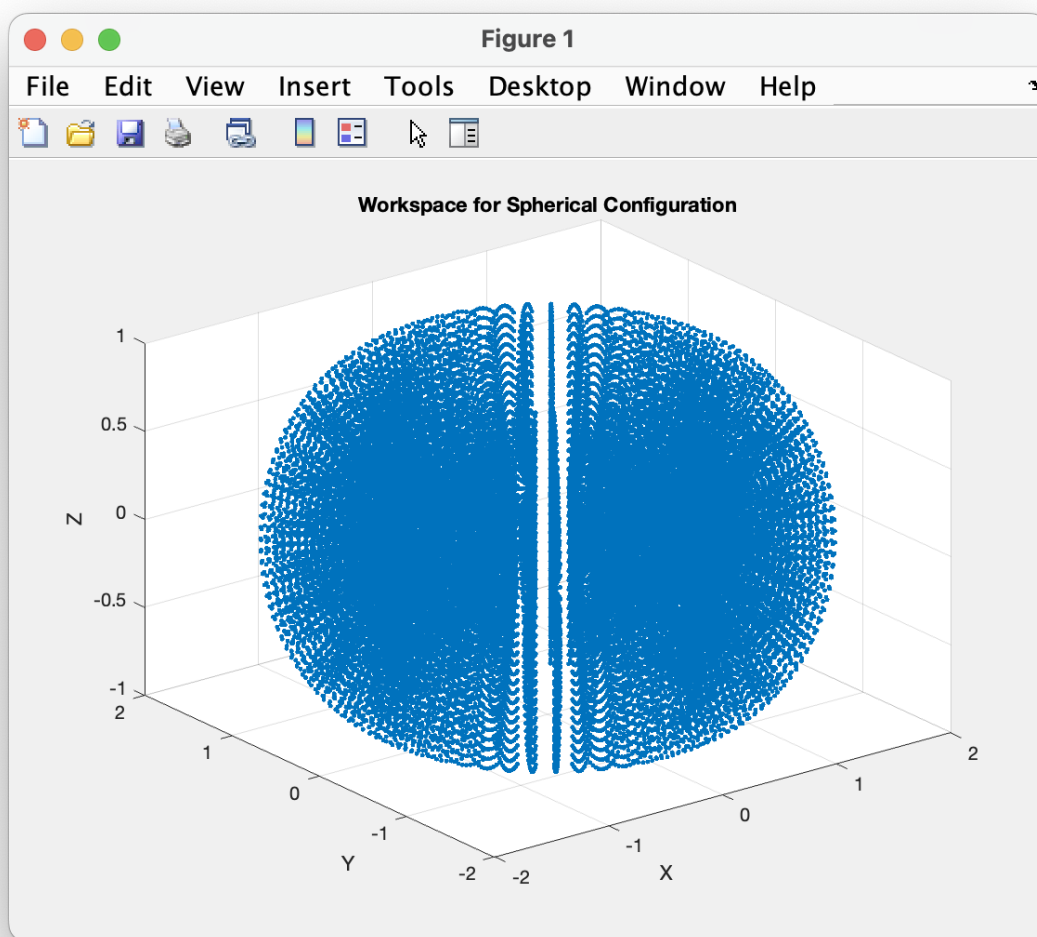


```

% Plot the reachable points in 3D
scatter3(reachable_points(:, 1), reachable_points(:, 2), reachable_points(:, 3),
'.'');
xlabel('X');
ylabel('Y');
zlabel('Z');
title('Workspace for Spherical Configuration');
axis([-2 2 -2 2 -1 1]); % Set axis limits for X, Y, and Z

```

## Work Envelope:



## 4. SCARA Robot Configuration

### MATLAB Code:

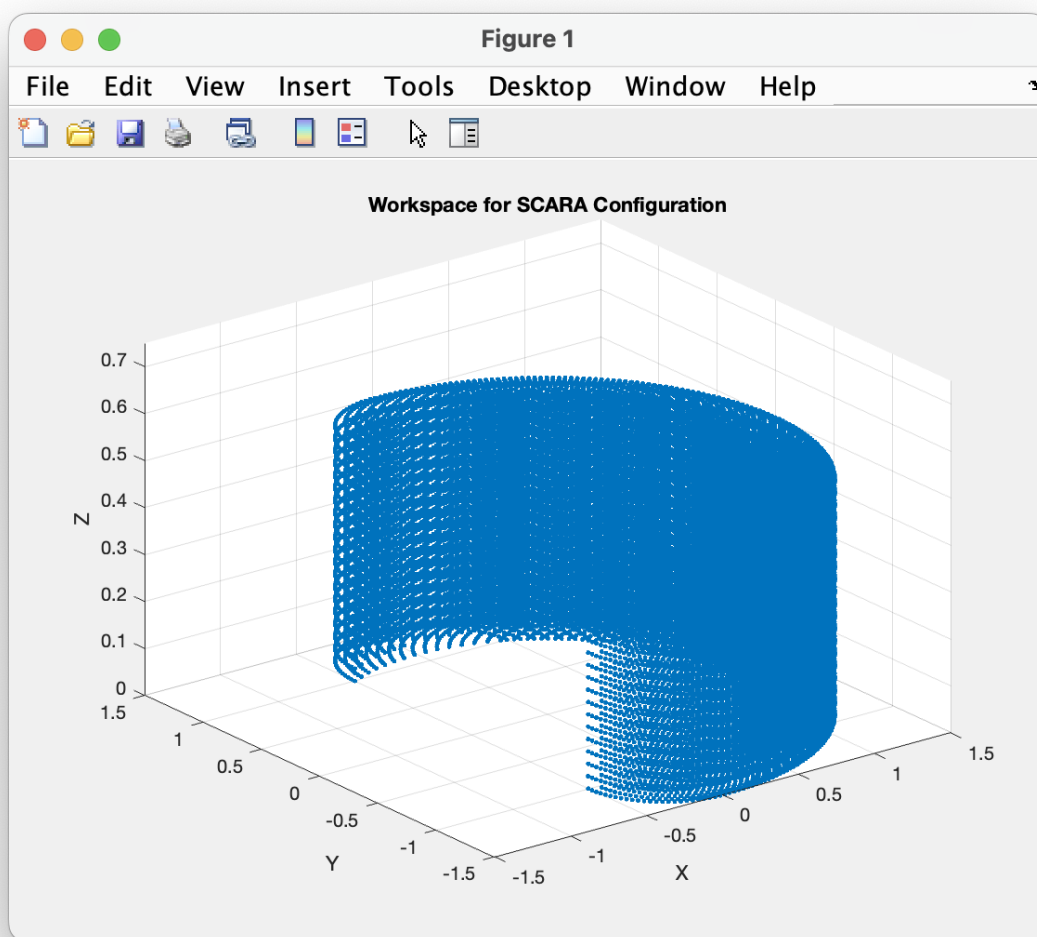
```
% Import robotics system toolbox
import robotics.*;
% Create a rigid body tree
robot = robotics.RigidBodyTree;
% Add the first link (link1) and joint (joint1)
link1 = robotics.RigidBody('link1');
joint1 = robotics.Joint('joint1', 'revolute');
joint1.HomePosition = 0;
setFixedTransform(joint1, eye(4));
link1.Joint = joint1;
addBody(robot, link1, 'base');
% Add the second link (link2) and joint (joint2)
link2 = robotics.RigidBody('link2');
joint2 = robotics.Joint('joint2', 'revolute');
joint2.HomePosition = 0;
setFixedTransform(joint2, eye(4));
link2.Joint = joint2;
addBody(robot, link2, 'link1');
% Add the third link (link3) and joint (joint3)
link3 = robotics.RigidBody('link3');
joint3 = robotics.Joint('joint3', 'prismatic');
joint3.HomePosition = 0;
setFixedTransform(joint3, eye(4));
link3.Joint = joint3;
addBody(robot, link3, 'link2');
% Set joint limits
theta1_range = linspace(-pi/2, pi/2, 50); % Revolute joint angle (-pi/2 to pi/2)
theta2_range = linspace(-pi/2, pi/2, 50); % Revolute joint angle (-pi/2 to pi/2)
z_range = linspace(0, 0.5, 20); % Prismatic joint position (0 to 0.5)
% Initialize an empty matrix to store reachable points
reachable_points = [];
% Loop through all possible joint configurations
for theta1 = theta1_range
    for theta2 = theta2_range
        for z = z_range
            % Compute forward kinematics for SCARA configuration
            x = 1*cos(theta1) + 0.5*cos(theta1 + theta2);
            y = 1*sin(theta1) + 0.5*sin(theta1 + theta2);
            position = [x, y, z];
            % Store the reachable points
            reachable_points = [reachable_points; position];
        end
    end
end
```

```

% Plot the reachable points in 3D
scatter3(reachable_points(:, 1), reachable_points(:, 2), reachable_points(:, 3),
'.');
xlabel('X');
ylabel('Y');
zlabel('Z');
title('Workspace for SCARA Configuration');
axis([-1.5 1.5 -1.5 1.5 0 0.75]); % Set axis limits for X, Y, and Z

```

## Work Envelope:



## 5. Articulated Robot Configuration

### MATLAB Code:

```
% Import robotics system toolbox
import robotics.*;
% Create a rigid body tree
robot = robotics.RigidBodyTree;
% Add the first link (link1) and joint (joint1)
link1 = robotics.RigidBody('link1');
joint1 = robotics.Joint('joint1', 'revolute');
joint1.HomePosition = 0;
setFixedTransform(joint1, eye(4));
link1.Joint = joint1;
addBody(robot, link1, 'base');
% Add the second link (link2) and joint (joint2)
link2 = robotics.RigidBody('link2');
joint2 = robotics.Joint('joint2', 'revolute');
joint2.HomePosition = 0;
setFixedTransform(joint2, eye(4));
link2.Joint = joint2;
addBody(robot, link2, 'link1');
% Add the third link (link3) and joint (joint3)
link3 = robotics.RigidBody('link3');
joint3 = robotics.Joint('joint3', 'revolute');
joint3.HomePosition = 0;
setFixedTransform(joint3, eye(4));
link3.Joint = joint3;
addBody(robot, link3, 'link2');
% Set joint limits
theta1_range = linspace(-pi, pi, 20); % Revolute joint angle (-pi to pi)
theta2_range = linspace(-5*pi/6, 5*pi/6, 50); % Revolute joint angle (-pi to pi)
theta3_range = linspace(-5*pi/6, 5*pi/6, 20); % Revolute joint angle (-pi to pi)
% Set link lengths
L1 = 1; L2 = 1; L3 = 0.5;
% Initialize an empty matrix to store reachable points
reachable_points = [];
% Loop through all possible joint configurations
for theta1 = theta1_range
    for theta2 = theta2_range
        for theta3 = theta3_range
            % Compute forward kinematics for the articulated robot configuration
            x = L1 * cos(theta1) + L2 * cos(theta2) * cos(theta1) + L3 * cos(theta2 +
theta3) * cos(theta1);
            y = L1 * sin(theta1) + L2 * cos(theta2) * sin(theta1) + L3 * cos(theta2 +
theta3) * sin(theta1);
            z = L2 * sin(theta2) + L3 * sin(theta2 + theta3);
```

```

    % Store the reachable points
    position = [x, y, z];
    reachable_points = [reachable_points; position];
end
end
end
% Plot the reachable points in 3D
scatter3(reachable_points(:, 1), reachable_points(:, 2), reachable_points(:, 3),
'.'');
xlabel('X'); ylabel('Y'); zlabel('Z');
title('Workspace for Articulated Configuration');
axis([-3 3 -3 3 -3 3]); % Set axis limits for X, Y, and Z

```

### Work Envelope:

