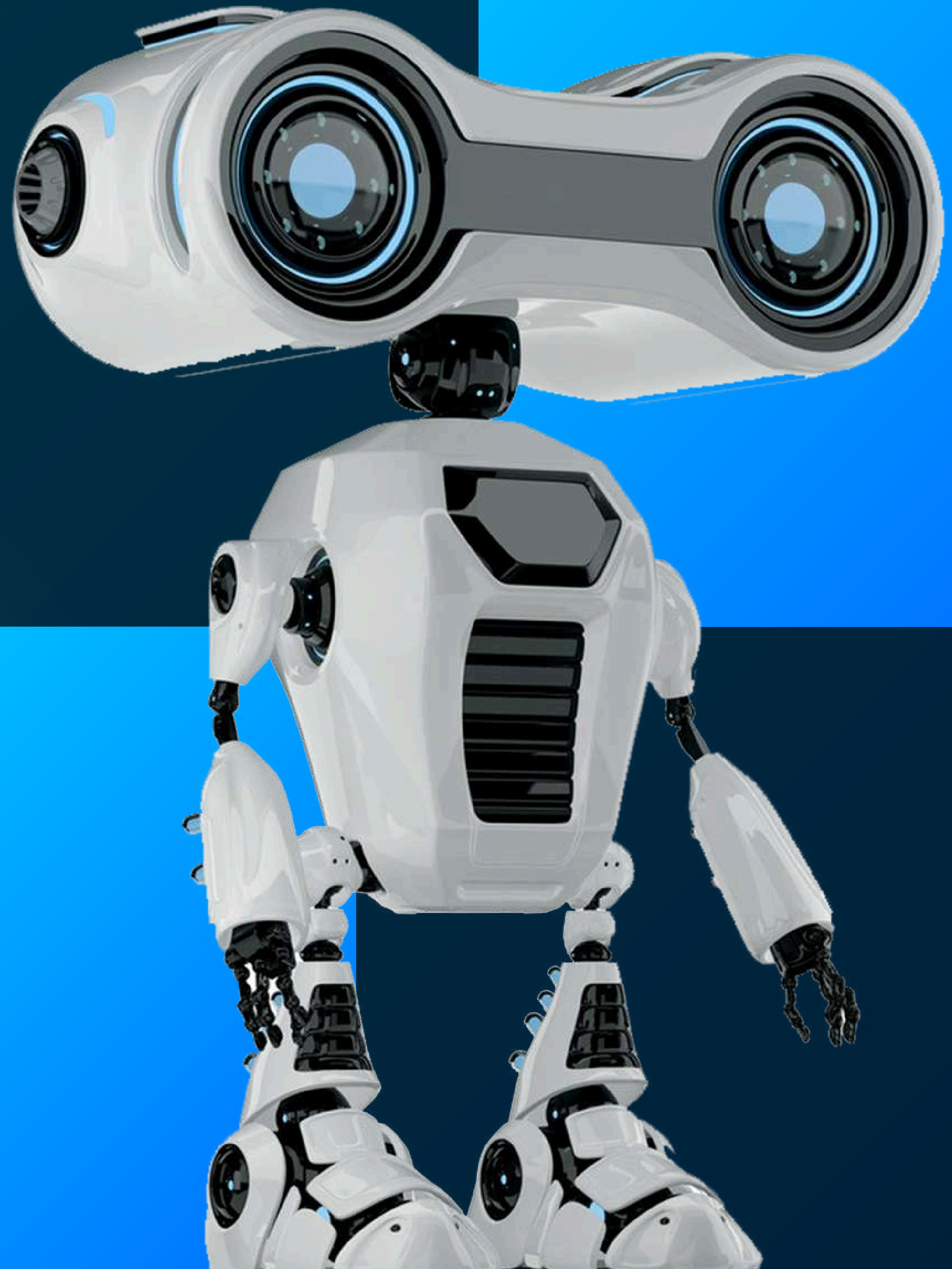


simulation of 3D robotic arm

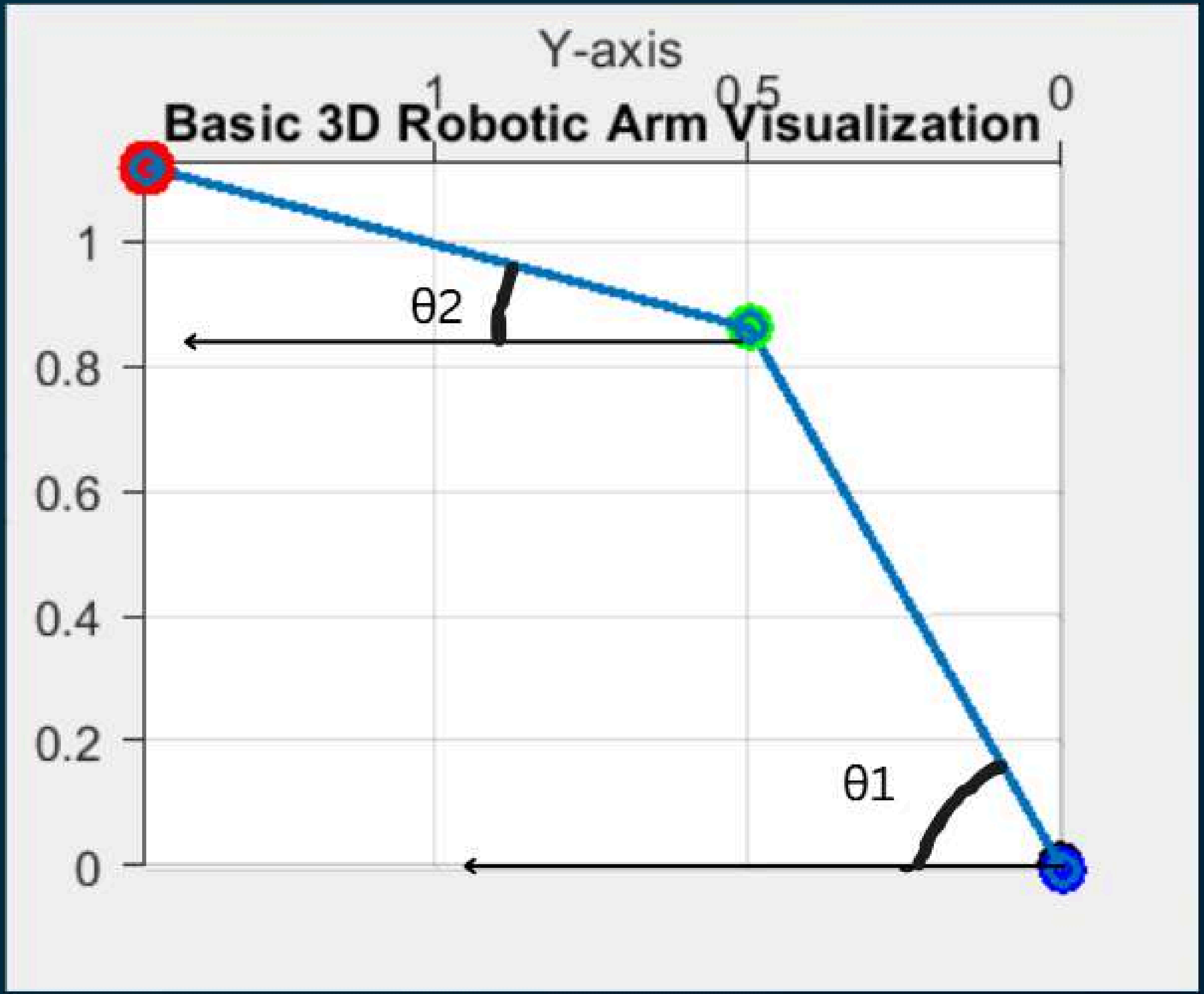
Forward kinematics

Forward Kinematics

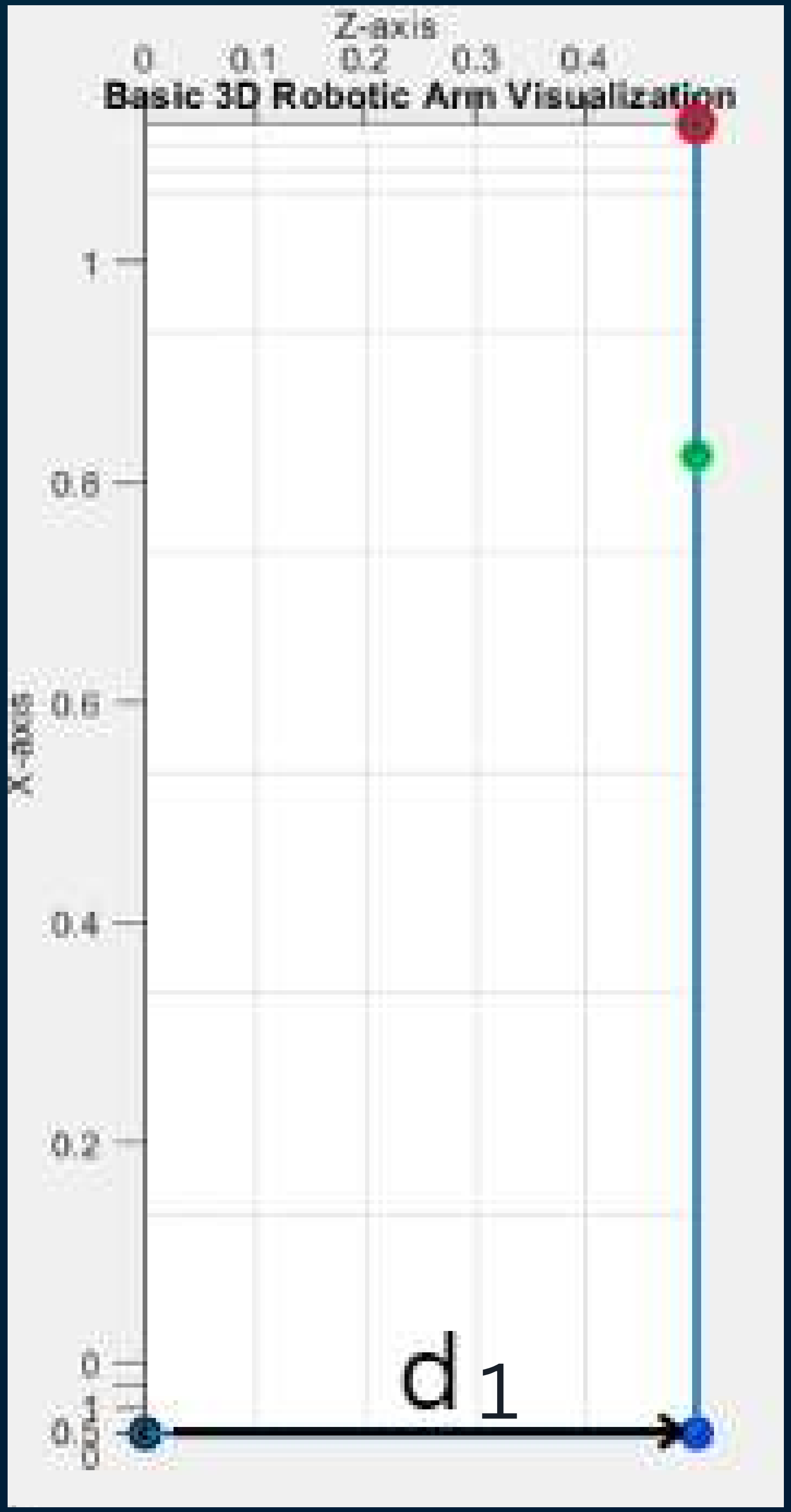
Forward kinematics involves calculating the position and orientation of the end-effector of a robotic arm based on its joint parameters and link configurations.

In our case, the robotic arm has:

1. A prismatic joint that moves along the Z-axis (joint displacement d_1).
2. Two revolute joints, where θ_1 and θ_2 define their respective angles.



side view

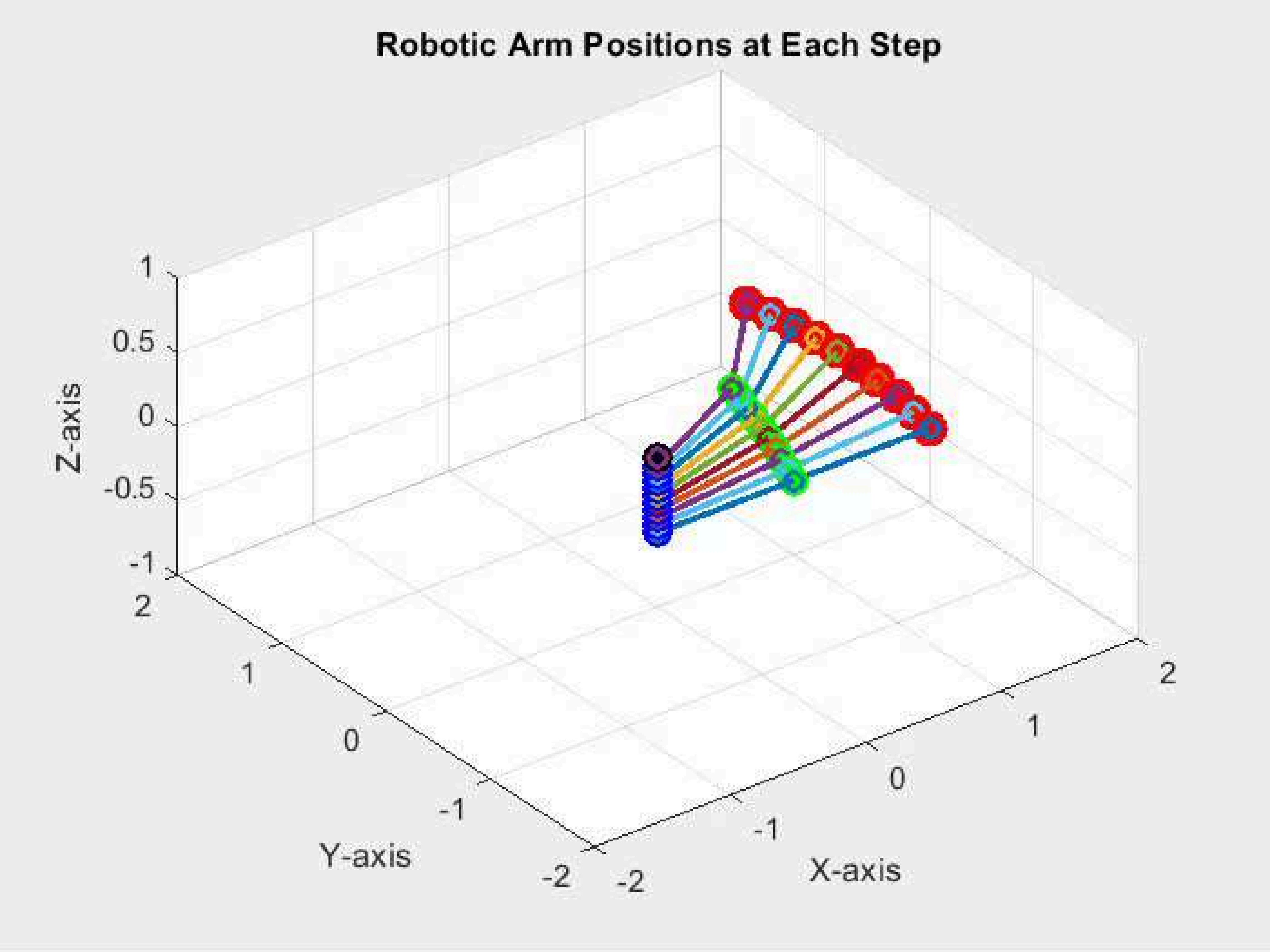


front view

matlab code for forward kinematics:

```
function [xa,ya,za] = forward_kinematics(theta1_a,theta2_a,Z)
l1=1;
l2=1;
xa=l1*cos(theta1_a)+l2*cos(theta2_a+theta1_a);
ya=l1*sin(theta1_a)+l2*sin(theta2_a+theta1_a);
za=Z;
```

Simulation on matlab



matlab code:

```
% Clear workspace
clear; clc; close all;

% Define link lengths
L1 = 1; % Length of first rotational link
L2 = 1; % Length of second rotational link

% Define joint variables and range
d1_range = -0.5:0.05:0.5; % Prismatic joint displacement range (along Z-axis)
theta2_range = linspace(0, pi/3, length(d1_range)); % Joint 2 angle range (in radians)
theta3_range = linspace(0, pi/4, length(d1_range)); % Joint 3 angle range (in radians)

% Initialize figure
figure;
hold on;
grid on;
xlabel('X-axis'); ylabel('Y-axis'); zlabel('Z-axis');
axis equal;
axis([-2 2 -2 2 -1 1]); % Adjust axes limits for better visualization
view(3); % Set 3D view
title('Robotic Arm Positions at Each Step');

% Create a video writer object
videoFileName = 'robotic_arm_simulation_very_slow.mp4'; % Video file name
videoWriter = VideoWriter(videoFileName, 'MPEG-4'); % Create VideoWriter object
videoWriter.FrameRate = 2; % Set a very slow frame rate (e.g., 2 frames per second)
open(videoWriter); % Open the video file for writing

% Loop through the range of displacements and angles
for i = 1:length(d1_range)
    % Current joint variables
    d1 = d1_range(i);        % Current prismatic joint displacement
    theta2 = theta2_range(i); % Current Joint 2 angle
    theta3 = theta3_range(i); % Current Joint 3 angle

    % Define joint positions
    joint1 = [0, 0, 0];                          % Base position
    joint2 = [0, 0, d1];                         % After prismatic joint
    joint3 = joint2 + [L1*cos(theta2), L1*sin(theta2), 0]; % After second joint
    end_effector = joint3 + [L2*cos(theta2 + theta3), L2*sin(theta2 + theta3), 0]; % End-effector
```

```
    % Plot the robotic arm in 3D
    plot3([joint1(1), joint2(1), joint3(1), end_effector(1)], ...
          [joint1(2), joint2(2), joint3(2), end_effector(2)], ...
          [joint1(3), joint2(3), joint3(3), end_effector(3)], 'o-', 'LineWidth', 2);

    % Mark joints and end-effector
    scatter3(joint1(1), joint1(2), joint1(3), 100, 'k', 'filled'); % Base
    scatter3(joint2(1), joint2(2), joint2(3), 100, 'b', 'filled'); % Joint 2
    scatter3(joint3(1), joint3(2), joint3(3), 100, 'g', 'filled'); % Joint 3
    scatter3(end_effector(1), end_effector(2), end_effector(3), 150, 'r', 'filled'); % End-effector

    % Capture the current frame
    frame = getframe(gcf); % Get the current figure as a frame
    writeVideo(videoWriter, frame); % Write the frame to the video

    % Pause for visualization (longer pause for very slow simulation)
    pause(.4); % Set to 1 second for an extremely slow simulation
end

% Close the video file
close(videoWriter);
disp(['Simulation saved as ', videoFileName]);

hold off;
```

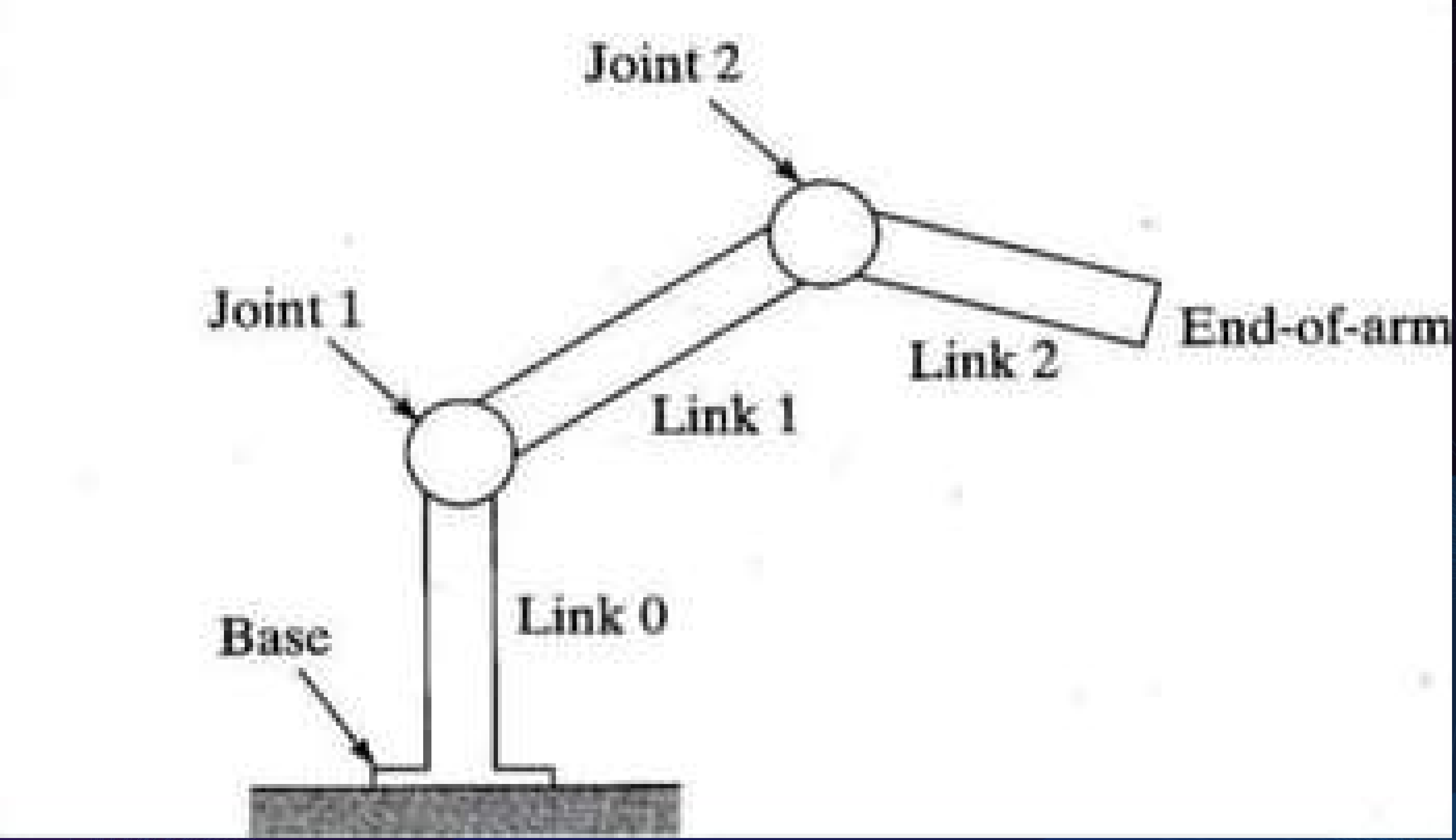
DH Parameters
Robotics

SANDEEP KUMAR GUPTA (Y24).
PRITAM PRIYADARSHI (Y23)
SOUMYA JHUNJHUNWALA (Y24)

What are DH Parameters?

- *Denavit - Hartenberg (DH) parameters simplify the representation of robotic arms.*
- *Define relationship between consecutive coordinate frames.*
- *Used for forward and inverse kinematics.*

LINKS AND JOINTS



Understanding the DH Parameters

1. *Link Length(ai-1):*

- *The distance between the zi-1 and zi axes, measured along the xi-1 axis.*
- *Represents the physical length of the link*

2. *Link Twist(αi-1):*

- *The angle between the zi-1 and the zi axes, measured around the xi-1 axis.*
- *Represents the twist or tilt of one link relative to other.*

3. *Link Offset(di):*

- *The distance between the xi−1 and xi axes, measured along the zi−1 axis.*
- *Represents the displacement along the previous z-axis.*

3. *Joint Angle(θi)*

- *The angle between the xi−1 and xi axes, measured around the zi−1 axis.*
- *Represents the rotation about the joint axis.*

Representing a DH TABLE

- *Tabular representation of DH parameters for a manipulator:*

Axis	a_{i-1}	α_{i-1}	d_i	θ_i
0 - 1	0	0°	d_1	$\Theta_1=0^\circ$
1 – 2	L_1	0°	0	Θ_2
2 - 3	L_2	0°	0	Θ_3

Using DH Parameters in Kinematics

- *Example: 3-DOF manipulator.*
- *Steps:*
- *1. Assign coordinate frames.*
- *2. Identify ai-1, αi-1, di, θi.*
- *3. Construct transformation matrices.*
- *4. Multiply to get end-effector pose.*

Transformation Matrix

$$
{}^{i-1}_{\ \ i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}\, d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}\, d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$T^{O}n = T^{O}_{1} * T^{1}_{2} * \ldots * T^{n-1}n$

$Pi\text{-}1 = T^{I-1}i * Pi$

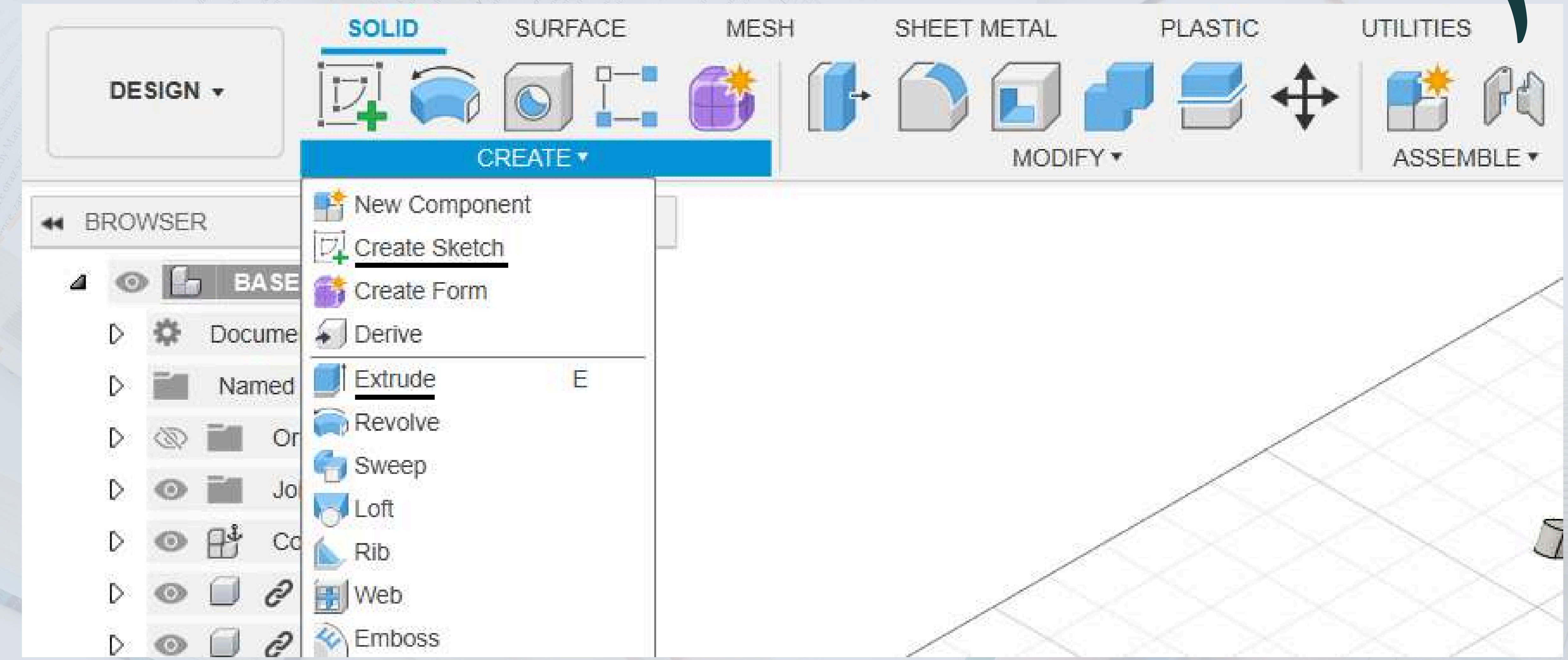
FUSION MODEL

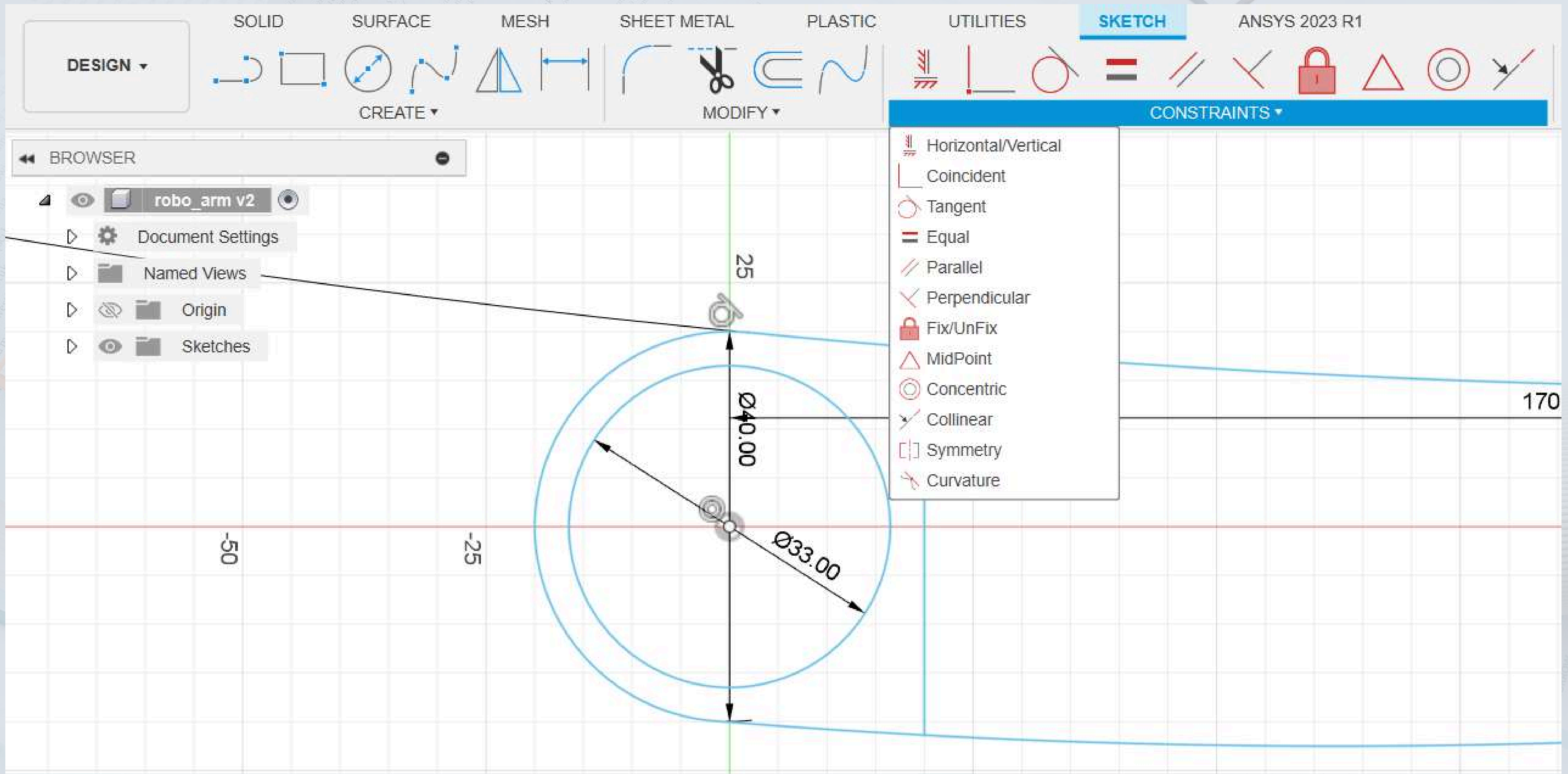
FUSION360

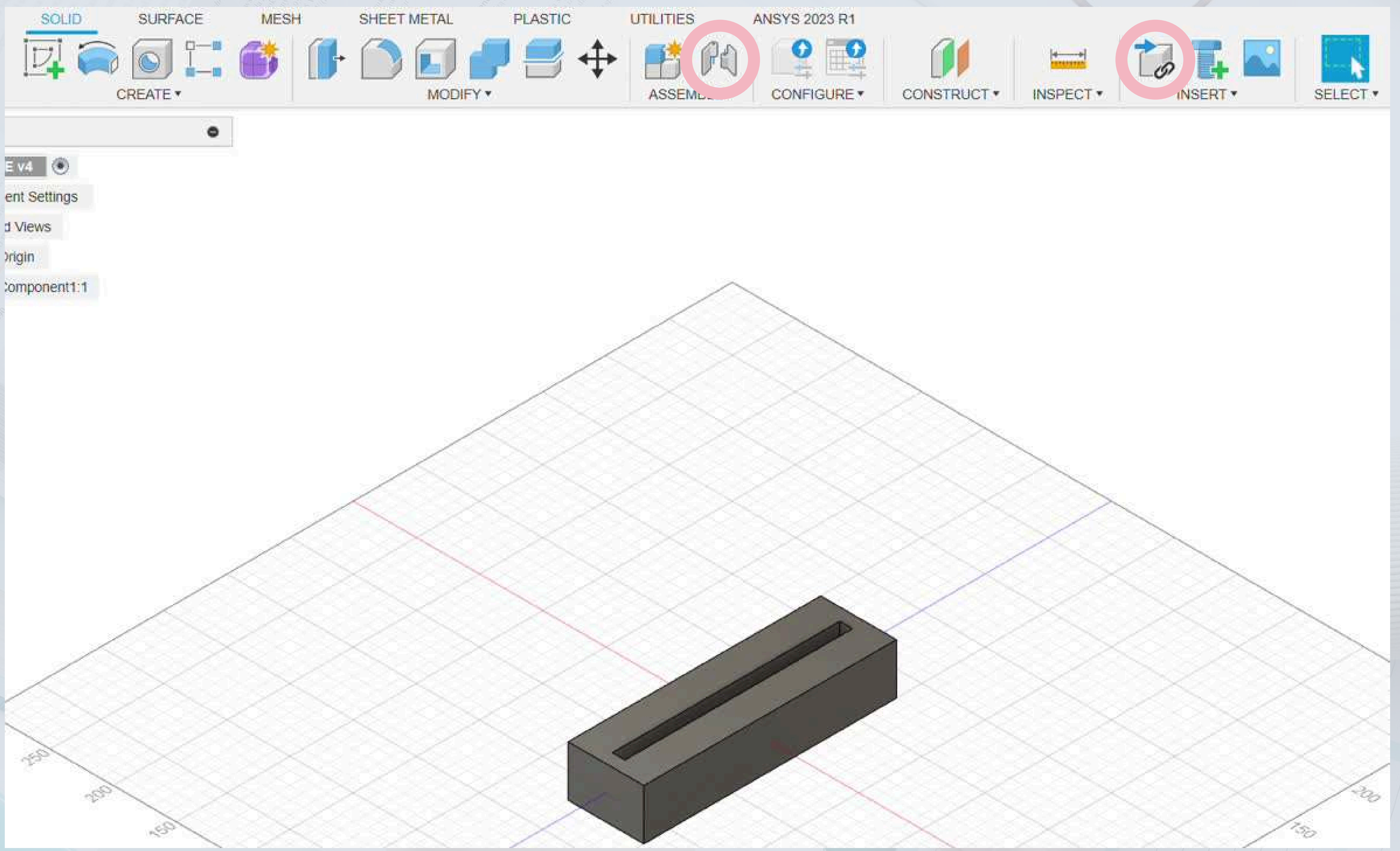
Fusion360 is a commercial software for CAD modelling, animation and simulation.
We learnt how to use Fusion360 for designing and 3D modelling.

Let’s look at the interface of the software.

JOINT


SOLID
SURFACE
MESH
SHEET METAL
PLASTIC
UTILITIES
DESIGN
CREATE
MODIFY
ASSEMBLE
BROWSER
BASE
Docume
Named
New Component
Create Sketch
Create Form
Derive
Extrude E
Revolve
Sweep
Loft
Rib
Web
Emboss


DESIGN
SOLID
SURFACE
MESH
SHEET METAL
PLASTIC
UTILITIES
SKETCH
ANSYS 2023 R1
CREATE
MODIFY
CONSTRAINTS
BROWSER
robo_arm v2
Document Settings
Named Views
Origin
Sketches
Horizontal/Vertical
Coincident
Tangent
Equal
Parallel
Perpendicular
Fix/UnFix
MidPoint
Concentric
Collinear
Symmetry
Curvature
25
Ø40.00
Ø33.00
170
-50
-25


SOLID
SURFACE
MESH
SHEET METAL
PLASTIC
UTILITIES
ANSYS 2023 R1
CREATE ▾
MODIFY ▾
CONFIGURE ▾
CONSTRUCT ▾
INSPECT ▾
INSERT ▾
SELECT ▾
E v4
ent Settings
d Views
Origin
Component1:1
250
200
150
200
150

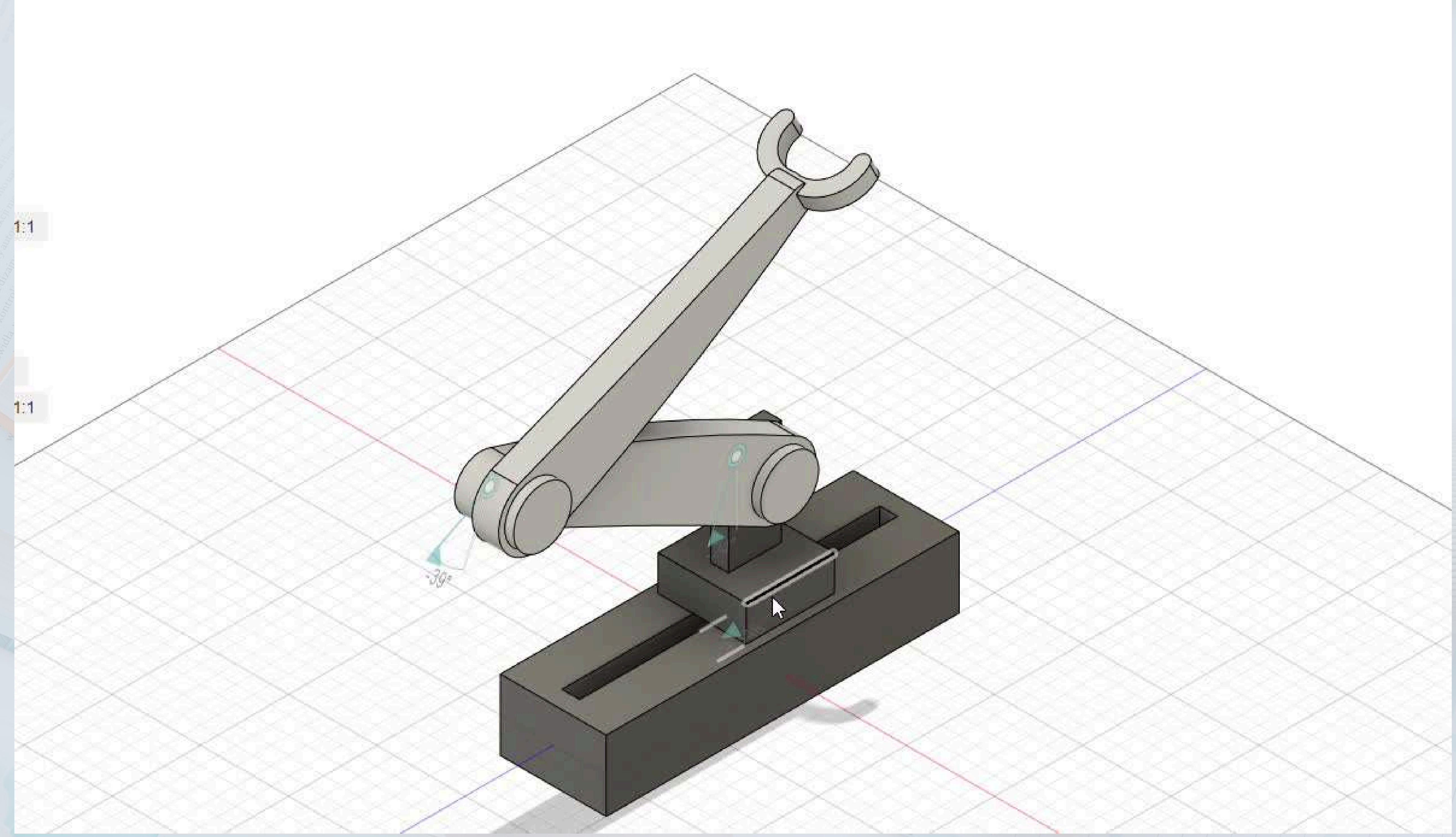
MODELLING OF THE ROBOTIC ARM ON FUSION 360

2D Model

3D Model

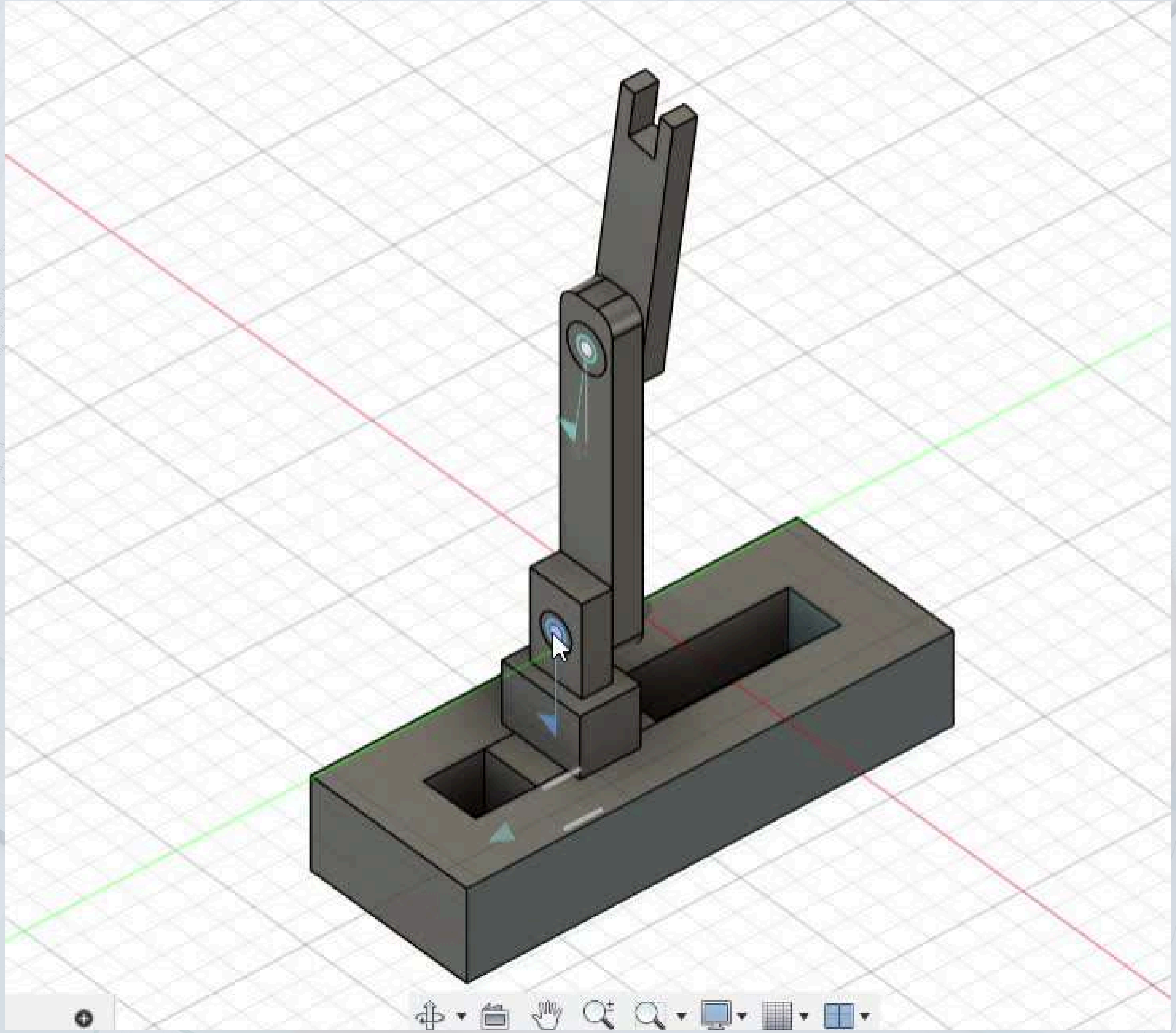
2D MODEL

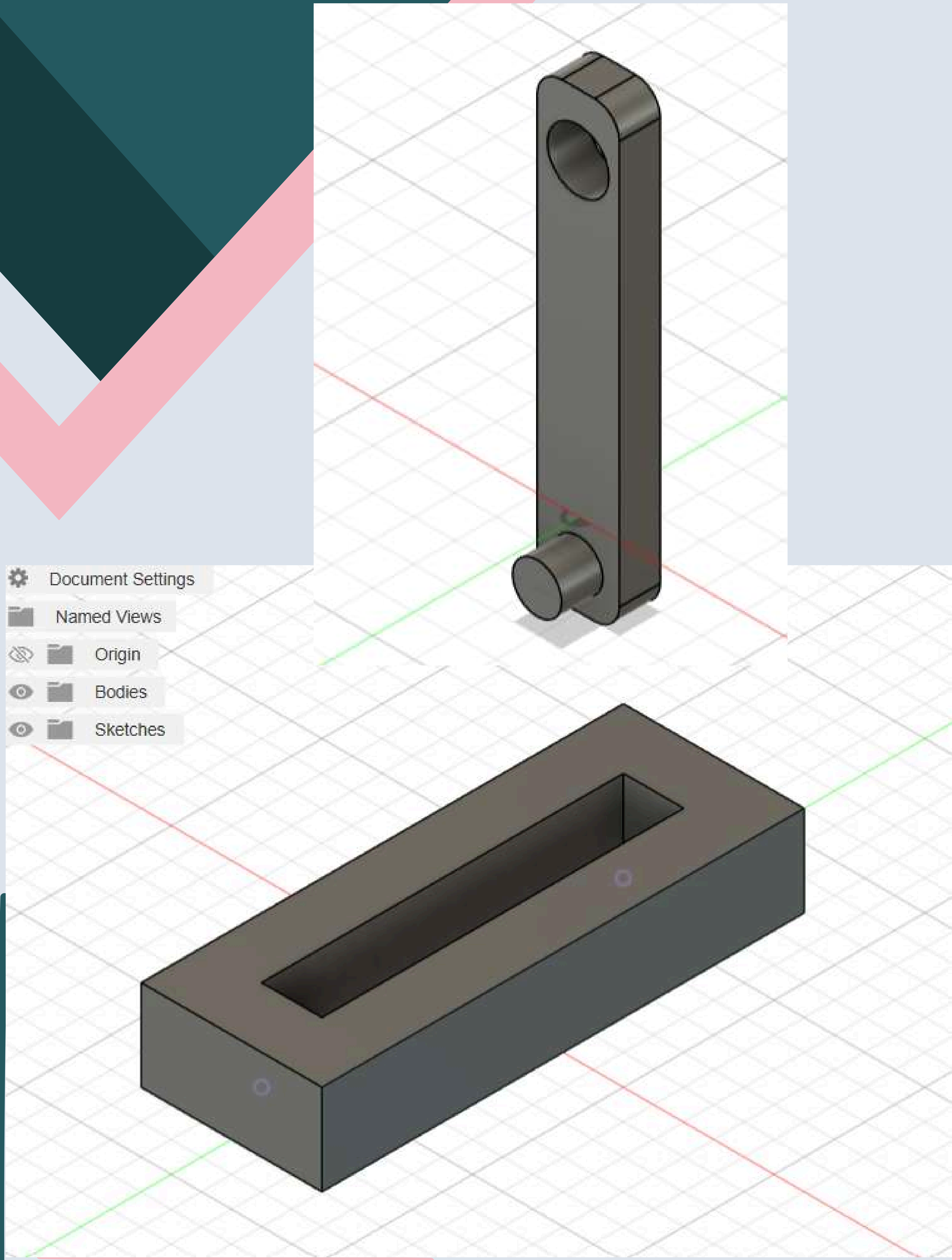
The 2D robotic arm functions along the YZ plane (since the model is oriented along the Z axis) only. The workspace of the arm is a semicircle of radius l1+l2 (where l1 and l2 are the lengths of the links).
There are 3 joints in the model: one prismatic (along Z) and two revolute (axes normal to YZ plane).

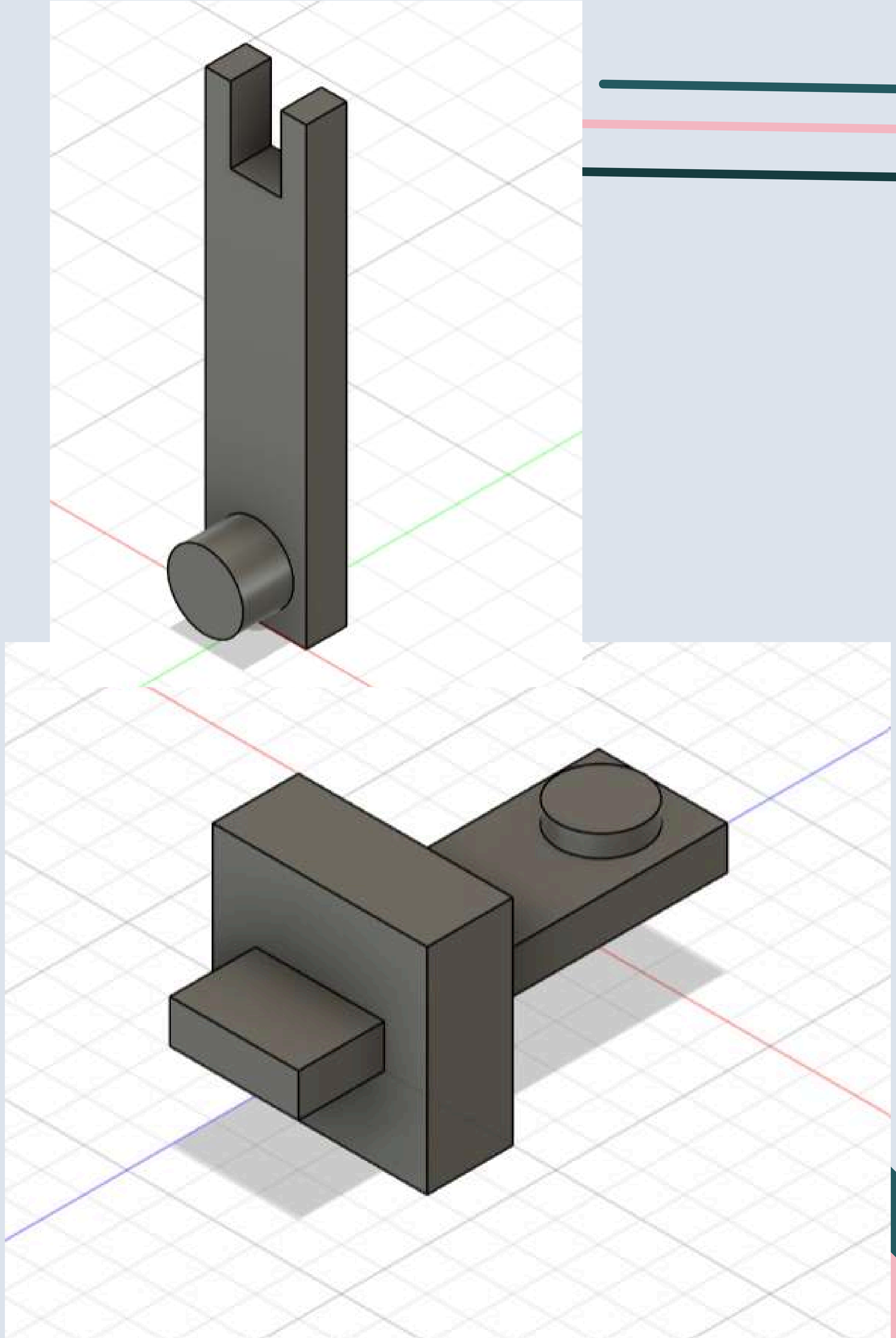

1:1
1:1
-39°

3D MODEL

The 3D robotic arm functions in a cylindrical work volume around 210 degrees for the arm to reach.




Document Settings
Named Views
Origin
Bodies
Sketches



SLIDER MOVES IN Y AXIS ONLY

ROD 1 AND ROD 2 MOVES IN X-Z PLANE

CONTRAINTS ON ROD 1 IS 105 DEGREES BOTH SIDES

NO CONTRAINTS ON ROD 2

MATLAB IMPLEMENTATION

Simulation of 3D robotic arm

ABOUT MATLAB

MATLAB, short for "MATrix LABoratory," is a proprietary programming language and numeric computing platform created by MathWorks. It supports various paradigms and offers capabilities such as matrix manipulations, function and data visualization, algorithm development, user interface design, and integration with programs written in other languages.

Simulink is a graphical programming environment built on MATLAB, designed for modeling, simulating, and analyzing multidomain dynamic systems. Its main interface features a graphical block diagramming tool and a flexible library of blocks. Simulink integrates seamlessly with MATLAB, allowing it to either control MATLAB operations or be operated through MATLAB scripts.

MATLAB CODE

Initialising all the variables

```
% Clear workspace
clear; clc; close all;

% Define link lengths (fixed)
L1 = 1; % Length of the first rotational link
L2 = 1; % Length of the second rotational link

% Define initial joint variables
d1 = 0.5;           % Prismatic joint displacement (along Z-axis)
theta2 = 0;         % Initial angle for Joint 2
theta3 = 0;         % Initial angle for Joint 3
```

Initialising the plot for animation

```
% Create figure
figure;
axis([-3 3 -3 3 0 3]); % Set axis limits for X, Y, Z
grid on; hold on;
xlabel('X-axis'); ylabel('Y-axis'); zlabel('Z-axis');
title('3D Robotic Arm Simulation ');
view(3); % 3D view
```

ANIMATION

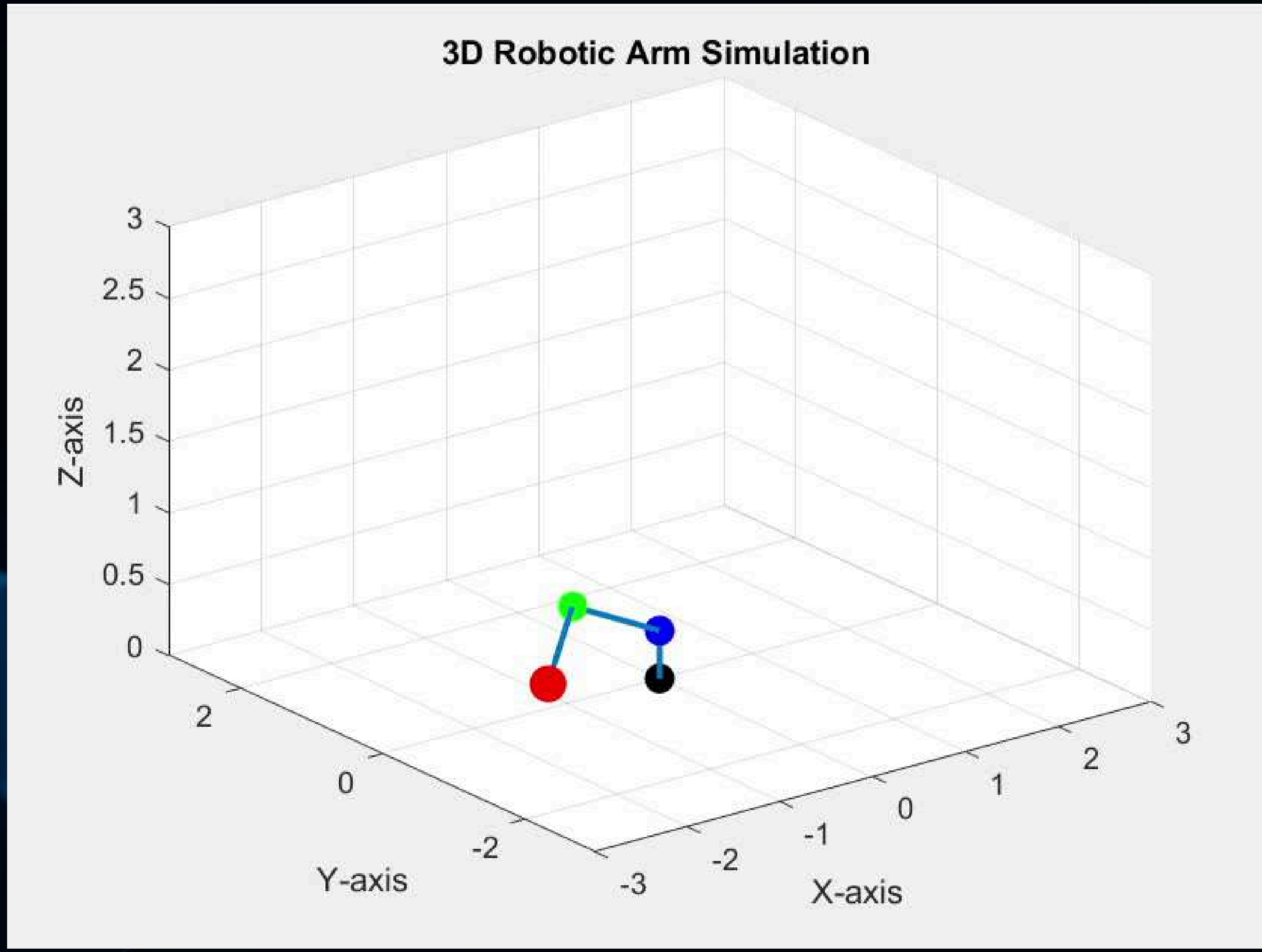
For loop to change the position variables with time

```
% Animation loop
for t = 1:100
    % Update joint variables (simulating motion)
    %base = sin(0.1 * t); % Base oscillates left and right
    d1 = sin(0.1 * t);               % Prismatic joint displacement (fixed in this simulation)
    theta2 = 0.1 * t;    % Joint 2 rotates over time
    theta3 = 0.2 * sin(0.1 * t); % Joint 3 oscillates

    % Define joint positions
    joint1 = [0, 0, 0];                          % Base position
    joint2 = joint1 + [0, 0, d1];                   % After prismatic joint (along Z-axis)
    joint3 = joint2 + [L1 * cos(theta2),0, abs(L1 * sin(theta2))]; % After first rotational link
```

ANIMATION

```matlab
    % Nested loop for end-effector rotation around Joint 3's axis
    for phi = linspace(0, 2 * pi, 10) % 20 steps for a full rotation
        % End-effector position with rotation around Joint 3's local axis
        end_effector = joint3 + ...
            [L2 * cos(theta2 + theta3) * cos(phi), ...
             L2 * cos(theta2 + theta3) * sin(phi), ...
             -abs(L2 * sin(theta2 + theta3))];

        % Clear previous plot
        cla;

        % Plot the robotic arm
        plot3([joint1(1), joint2(1), joint3(1), end_effector(1)], ...
              [joint1(2), joint2(2), joint3(2), end_effector(2)], ...
              [joint1(3), joint2(3), joint3(3), end_effector(3)], 'o-', 'LineWidth', 2);

        % Mark joints and end-effector
        scatter3(joint1(1), joint1(2), joint1(3), 100, 'k', 'filled'); % Base
        scatter3(joint2(1), joint2(2), joint2(3), 100, 'b', 'filled'); % Joint 2
        scatter3(joint3(1), joint3(2), joint3(3), 100, 'g', 'filled'); % Joint 3
        scatter3(end_effector(1), end_effector(2), end_effector(3), 150, 'r', 'filled'); % End-effector

        % Pause to create animation effect
        pause(0.01);
    end
end
```

PLOT



The animation of the moving robot arm appears in a Figure window

THANK YOU!