**WELDING ROBOT**

A PROJECT REPORT

*Submitted by GROUP E3*

**B Sai Abhishek BL.EN.U4AIE21015**

**B Ruchith Balaji BL.EN.U4AIE21017**

**Chillakuru Hari BL.EN.U4AIE21038**

***for the course***

***21AIE205- Introduction to Robotics***

***Guided and evaluated by***

***Nidhin Prabhakar T V Sir***

***Dept. of CSE,***



**AMRITA SCHOOL OF ENGINEERING, BANGALORE**

**AMRITA VISHWA VIDHYAPEETHAM**

**BANGALORE-560 035**

January 2023

**INDEX**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **TOPIC** | **Page No** |
| 1 | Chapter-1 (Problem Explanation) | 3 |
| 2 | Cgapter-2 (Implementation) | 5 |
| 3 | Chapter-3 (Result) | 10 |
| 4 | Chapter-4 (Reference) | 12 |

**CHAPTER 1**

**PROBLEM STATEMENT**

***Design and implement a welding robot with a suitable degree of freedom. The input for the system is the welding area. The welding area can be as simple as a line or some basic shape. Given the input, the robot should be able to weld the location.***

This problem involves designing and implementing a welding robot that can weld a specific area according to the input provided. The input for the system is the welding area, which can be a simple shape such as a line or a more complex shape. The robot should have a suitable degree of freedom, meaning it should be able to move and manipulate itself in a way that allows it to reach and weld all parts of the designated area. The robot needs to have a suitable degree of freedom, meaning it needs to be able to move and rotate in multiple directions to reach and weld the designated areas. The goal is for the robot to be able to take the input and then weld the designated location.

First, the robot must be able to move and position itself in a way that allows it to effectively weld a given area. Second, the robot must have the necessary sensors and control systems to accurately and consistently weld the desired location. Finally, the robot must be able to adapt to different welding areas and shapes, as the input for the system can vary widely.

One of the key challenges in this project is determining the degree of freedom that is necessary for the robot to effectively weld the designated area. Degree of freedom refers to the number of axes that a robot can move along. For a welding robot, the main axes of movement will likely include the x, y, and z axes, as well as the rotational axes for the welding arm and tool.

Another major challenge is creating a suitable control system for the robot. This system must be able to accurately and efficiently move the robot in the designated areas, taking into account factors such as the size and shape of the welding area and the properties of the materials being welded. Additionally, the control system must also be able to account for any variations in the welding area, such as changes in the position of the welding material or any obstacles that may be present.

The first step in designing a welding robot for this problem would be to determine the degree of freedom that is required. This will depend on the complexity of the welding area and the type of welding that needs to be done. For example, if the welding area is a simple line, a robot with a single degree of freedom, such as a linear actuator, may be sufficient. However, if the welding area is a more complex shape, a robot with multiple degrees of freedom would be needed. Once the control system is in place, the next step is to design and implement the welding arm and tool. This arm and tool must be able to withstand the high temperatures and forces associated with welding, and must be able to move in the designated areas with a high degree of precision and accuracy. Additionally, the arm and tool must also be able to handle different types of welding materials and be able to switch between different welding techniques, such as arc welding and gas tungsten arc welding.

Next, the robot's mechanical design would need to be created. This would involve designing the robot's kinematics, which determine how the robot will move, and the dynamics, which determine how the robot will respond to forces and torques. The mechanical design should also take into account the size and weight of the robot, as well as its environment, including the workspace and any potential obstacles.

Once the mechanical design is complete, the robot's control system would need to be implemented. This would involve programming the robot's control algorithms, which will govern the robot's motion and how it responds to input. The control system should be able to handle the robot's different degree of freedom and manage its welding process.

Finally, the robot should be tested and calibrated to ensure that it can weld the designated area accurately and efficiently. This could involve using simulations or physical prototypes, as well as conducting tests to verify that the robot can weld the area within an acceptable degree of precision.

Overall, designing and implementing a welding robot with a suitable degree of freedom is a complex task that requires a thorough understanding of robotics, welding technology, and control systems. By effectively addressing the challenges of determining the appropriate degree of freedom, creating a suitable control system, designing and implementing a robust welding arm and tool, and addressing safety concerns, it is possible to create a robot that can accurately and efficiently weld designated areas.

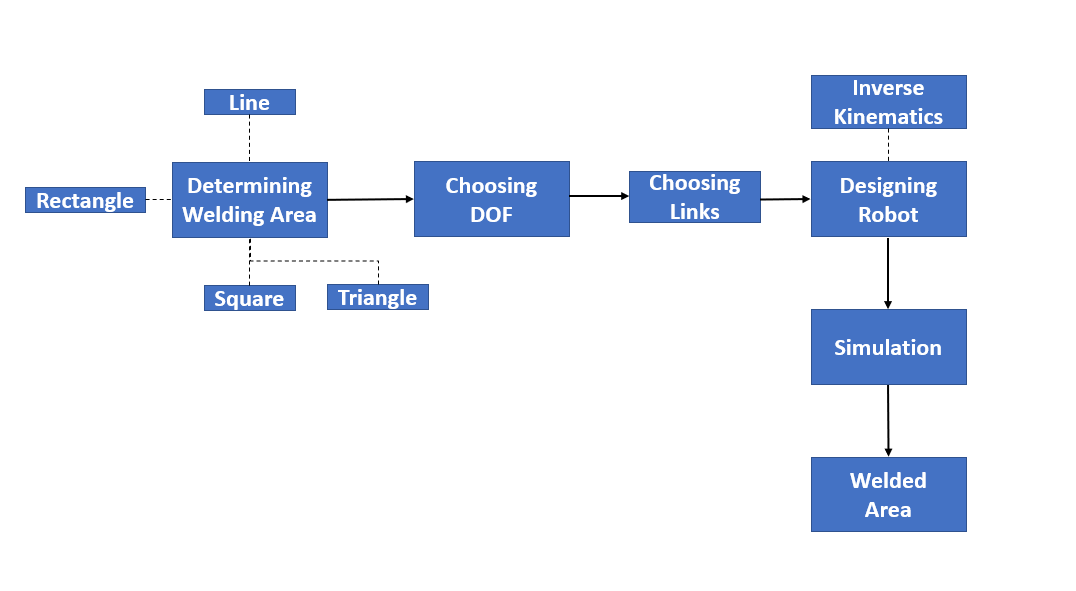
**CHAPTER 2**

**IMPLEMENTATION**

A welding robot with 6 degrees of freedom can be designed and implemented by following a specific set of steps. The first step is to determine the welding area, which can be as simple as a line or some basic shape. The next step is to design the robot itself, including its joints, motors, and sensors. The final step is to implement the software and control system that will be used to control the robot.

The first step in designing a welding robot is to determine the welding area. This will typically involve working with a customer or other stakeholder to understand their needs and requirements. This could include a simple line or more complex shapes such as a rectangle, square, or even more complex geometric shapes. Once the welding area has been determined, it can be used as the input for the robot.

The next step in designing the welding robot is to design the robot itself. This will typically involve working with a mechanical engineer or other expert to design the robot's joints, motors, and sensors. The robot's joints are typically composed of actuators, such as motors or pneumatic cylinders, that allow the robot to move in different directions. The robot's motors are typically used to drive the robot's movement, and sensors are used to detect the robot's position and orientation.

 The final step in designing and implementing a welding robot is to implement the software and control system that will be used to control the robot. This typically involves working with a software engineer or other expert to develop the control system. The control system is typically composed of a set of instructions or algorithms that the robot uses to move and weld. These instructions can be written in a Matlab or Python, and they can be stored on the robot's onboard computer or on a remote server.

*Fig-1*

Once the welding robot has been designed and implemented, it will be ready to be used in the welding area. The robot will typically be operated by a human operator who will use the control system to move the robot and perform the welding. The robot will typically be equipped with cameras and other sensors that allow the operator to see the welding area and monitor the robot's movements.

In conclusion, a welding robot with 6 degrees of freedom can be designed and implemented by following a specific set of steps. The first step is to determine the welding area, which can be as simple as a line or more complex shapes. The next step is to design the robot itself, including its joints, motors, and sensors. The final step is to implement the software and control system that will be used to control the robot. Once the robot has been designed and implemented, it will be ready to be used in the welding area, and it will typically be operated by a human operator who will use the control system to move the robot and perform the welding.

In this project we have given the welding are as:

1. 3-dimesional line.
2. Square.
3. Rectangle.
4. Triangle.

We have taken DOF as 6 and 6 links are taken.

Each link is taken on the basis of DH-parameters.

***DH-Parameters:***

The Denavit-Hartenberg (DH) parameters are a set of conventions for describing the kinematic characteristics of a serial link robot. They provide a standardized way of defining the relative locations and orientations of the robot's links and joints.

The DH parameters consist of four main parameters for each link:

* **'theta':** the angle of the joint between the current link and the next link, measured in the plane of the current link and with respect to a fixed reference frame (often called the world frame)
* **'d':** the distance along the common normal of the current link and the next link, between the surfaces of the two links where they meet at the joint.
* **'a':** the distance along the x-axis of the current link, between the origin of the current link's frame and the origin of the next link's frame
* **'alpha':** the angle between the z-axes of the current link's frame and the next link's frame, measured in the plane of the joint and with respect to the common normal.
* These parameters can be used to represent the forward and inverse kinematics of a robot.

In this Project we are using INVERSE KINEMATICS

In MATLAB Robotics Toolbox, the 'SerialLink' object, the class for creating robots with a fixed number of links and joints, use these parameters to create the robot model. The 'Link' object is constructed with DH parameters as input, which is then used to create 'SerialLink' object.

**CODE:**

clear all;

clc;

close all;

L1 = Link('revolute','d',0,'a',0,'alpha',-pi/2);

L2 = Link('revolute','d',0,'a',1000,'alpha',0);

L3 = Link('revolute','d',0,'a',0,'alpha',pi/2);

L4 = Link('revolute','d',1330,'a',0,'alpha',-pi/2);

L5 = Link('revolute','d',0,'a',0,'alpha',pi/2);

L6 = Link('revolute','d',470,'a',0,'alpha',0);

Robot = SerialLink([L1 L2 L3 L4 L5 L6],'name','Welding Manipulator');

Robot.teach

shape = 10;

while shape ~= 0

clc;

disp("Choose a shape that you would like weld");

disp("1. 3-Lines in 3 planes");

disp("2. Square");

disp("3. Rectangle")

disp("4. Triangle")

disp("0. Exit, please :)");

shape = input("Enter your choice:");

switch shape

case 1

T = transl(2000, 0, 0)\*rpy2tr(0,90,0,'deg');

T1 = transl(2000, -1500, 0)\*rpy2tr(-164.6,90,163,'deg');

T2 = transl(1000, -1500, 0)\*rpy2tr(-90,90,150,'deg');

T3 = transl(1000, -1500, 1000)\*rpy2tr(-90,90,150,'deg');

T\_in = ctraj(T, T1, 10);

plot3([2000 2000], [0 -1500],[0 0] ,'o-')

hold on

plot3([2000 1000], [-1500 -1500],[0 0] ,'o-')

plot3([1000 1000], [-1500 -1500],[0 1000] ,'o-')

T\_in1 = ctraj(T1,T2,10);

T\_in2 = ctraj(T2,T3,10);

q = Robot.ikine(T\_in);

q1= Robot.ikine(T\_in1);

q2 = Robot.ikine(T\_in2);

q\_total = [q;q1;q2];

Robot.teach(q\_total)

case 2

T = transl(2000, 0, 0)\*rpy2tr(0,90,0,'deg');

T1 = transl(2000, -1000, 0)\*rpy2tr(-164.6,90,163,'deg');

T2 = transl(2000, -1000, 1000)\*rpy2tr(-164.6,90,163,'deg');

T3 = transl(2000, 0, 1000)\*rpy2tr(-164.6,90,163,'deg');

plot3([2000 2000], [0 -1000],[0 0] ,'o-')

hold on

plot3([2000 2000], [-1000 -1000],[1000 0] ,'o-')

plot3([2000 2000], [-1000 -0],[1000 1000] ,'o-')

plot3([2000 2000], [-0 -0],[1000 0] ,'o-')

T\_in = ctraj(T,T1,10);

T\_in1 = ctraj(T1,T2,10);

T\_in2 = ctraj(T2,T3,10);

T\_in3 = ctraj(T3,T,10);

q = Robot.ikine(T\_in);

q1= Robot.ikine(T\_in1);

q2 = Robot.ikine(T\_in2);

q3 = Robot.ikine(T\_in3);

q\_total = [q;q1;q2;q3];

Robot.teach(q\_total)

case 3

T = transl(2000, 0, 0)\*rpy2tr(0,90,0,'deg');

T1 = transl(2000, -1500, 0)\*rpy2tr(-164.6,90,163,'deg');

T2 = transl(2000, -1500, 500)\*rpy2tr(-164.6,90,163,'deg');

T3 = transl(2000, 0, 500)\*rpy2tr(-164.6,90,163,'deg');

plot3([2000 2000], [0 -1500],[0 0] ,'o-')

hold on

plot3([2000 2000], [-1500 -1500],[500 0] ,'o-')

plot3([2000 2000], [-1500 -0],[500 500] ,'o-')

plot3([2000 2000], [-0 -0],[500 0] ,'o-')

T\_in = ctraj(T,T1,10);

T\_in1 = ctraj(T1,T2,10);

T\_in2 = ctraj(T2,T3,10);

T\_in3 = ctraj(T3,T,10);

q = Robot.ikine(T\_in);

q1= Robot.ikine(T\_in1);

q2 = Robot.ikine(T\_in2);

q3 = Robot.ikine(T\_in3);

q\_total = [q;q1;q2;q3];

Robot.teach(q\_total)

case 4

T = transl(2000, 0, 0)\*rpy2tr(0,90,0,'deg');

T1 = transl(2000, -1500, 0)\*rpy2tr(-164.6,90,163,'deg');

T2 = transl(2000, -750, 500)\*rpy2tr(-164.6,90,163,'deg');

%T3 = transl(2000, 0, 500)\*rpy2tr(-164.6,90,163,'deg');

plot3([2000 2000], [0 -1500],[0 0] ,'o-')

hold on

plot3([2000 2000], [-1500 -750],[0 500] ,'o-')

plot3([2000 2000], [-750 -0],[500 0] ,'o-')

T\_in = ctraj(T,T1,10);

T\_in1 = ctraj(T1,T2,10);

T\_in2 = ctraj(T2,T,10);

%T\_in3 = ctraj(T3,T,10);

q = Robot.ikine(T\_in);

q1= Robot.ikine(T\_in1);

q2 = Robot.ikine(T\_in2);

%q3 = Robot.ikine(T\_in3);

q\_total = [q;q1;q2];

Robot.teach(q\_total)

case 0

return

otherwise

disp('Invalid choice! Try again.');

pause(0);

end

end

**CODE EXPLANATION**

This script defines a welding manipulator robot using the Robotics Toolbox for MATLAB, allows the user to select a shape to weld, and then generates an animation of the robot moving through the path required to "weld" the selected shape.

The robot is defined as a 6-DOF serial link manipulator with revolute joints, using the Link class from the Robotics Toolbox. The dimensions and orientations of the links are defined by the 'd', 'a', and 'alpha' parameters in the Link class constructor for each link.

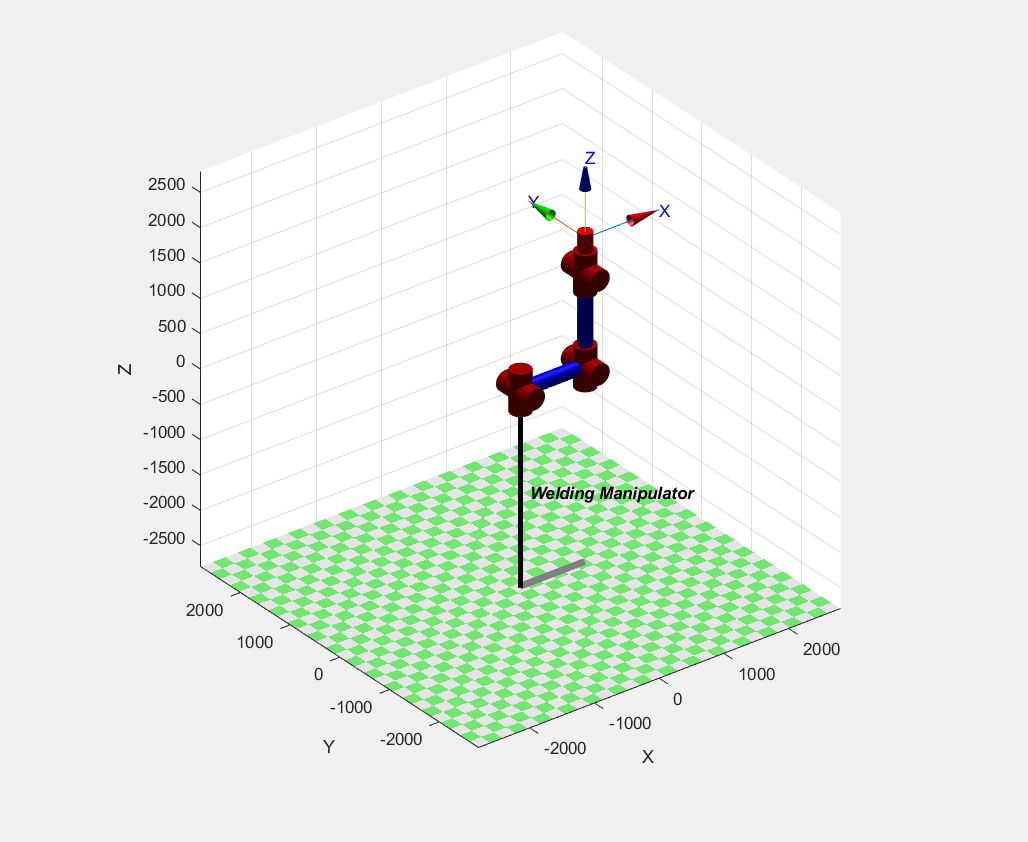
Once the robot is defined, the script allows the user to select a shape from a menu and then generates an array of joint configurations(q) corresponding to the path that the robot's end-effector should take to "weld" the selected shape. The script then uses the 'ikine' method of the robot object to inverse kinematically solve for the necessary joint angles, and finally, the 'teach' function is used to animate the robot moving through the generated path.

It should be noted that, this script is only a simulation, it doesn't account for the dynamic constraint, collision and other safety measures that would need to be considered when building a real robotic welder. Also, the specific values used in this script (such as the distance, angles and orientation) are arbitrary.

**CHAPTER 3**

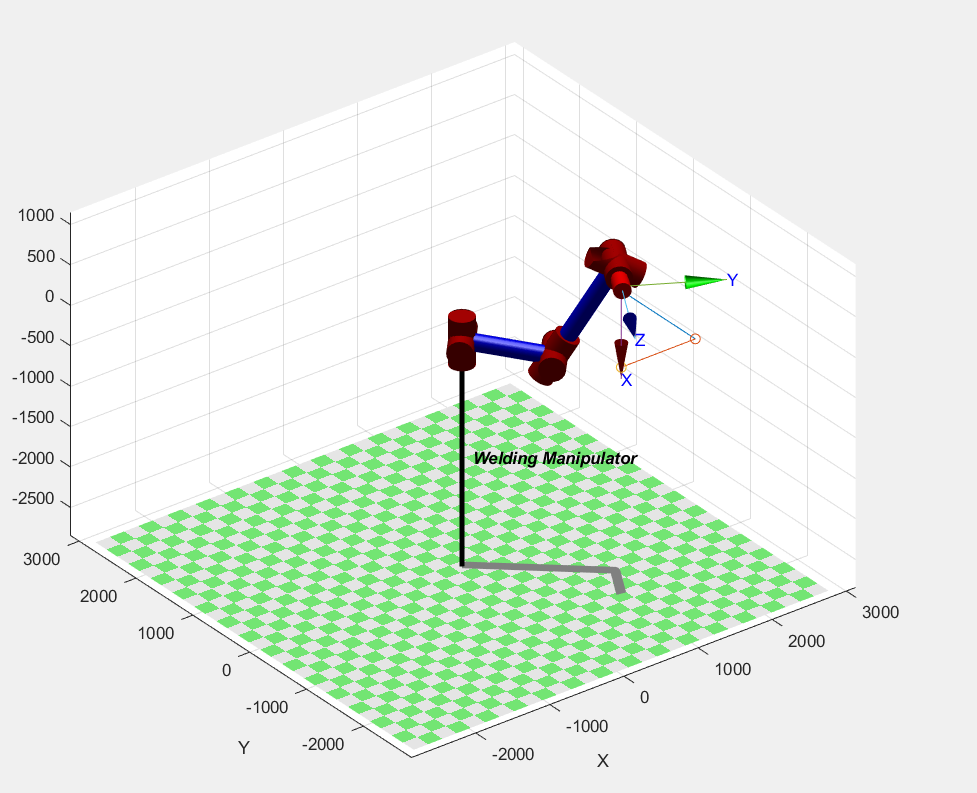
**RESULTS**

* Generating the Welding Robot:

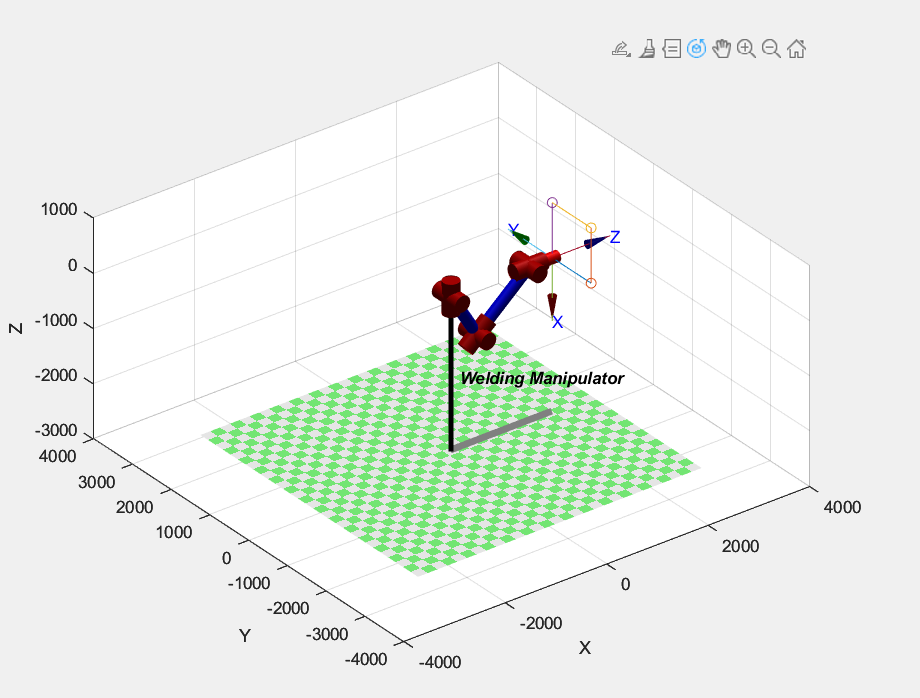


*Fig-2*

* Giving the Input Welding Area

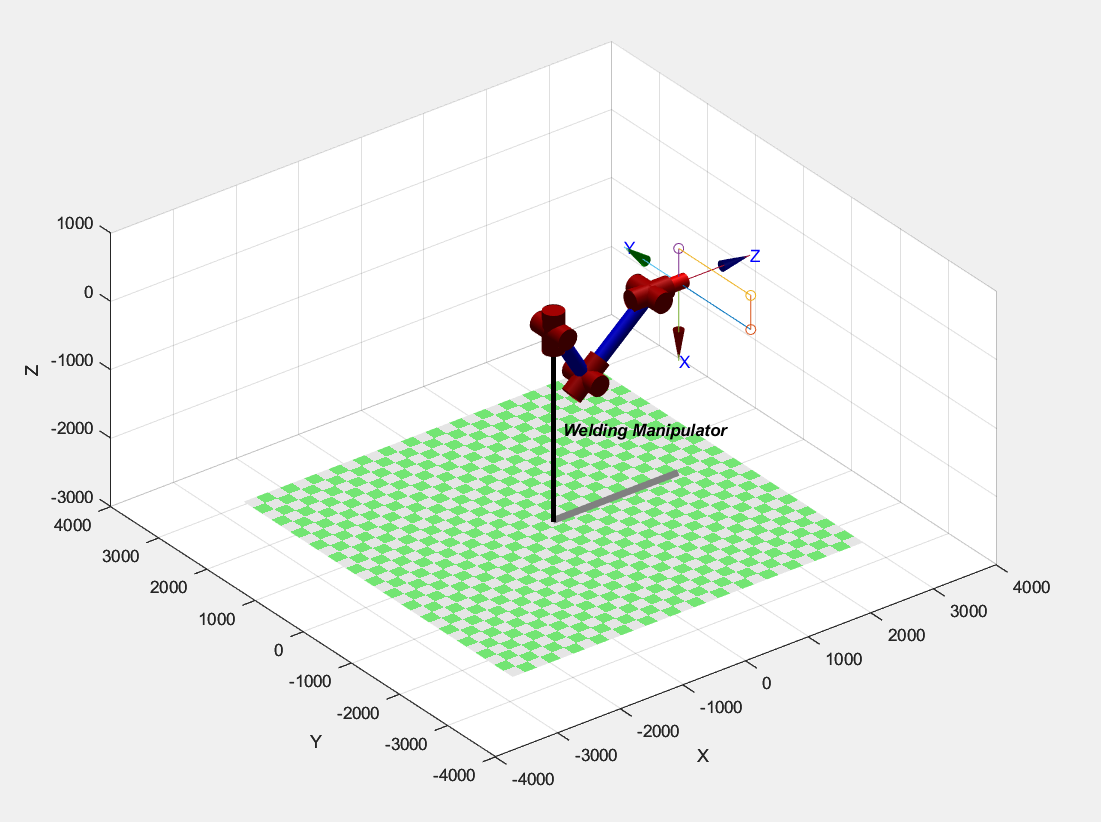
1.  3 lines in Different planes of 1000 units each.

*Fig-3*

1. Square (L = 1000 units)

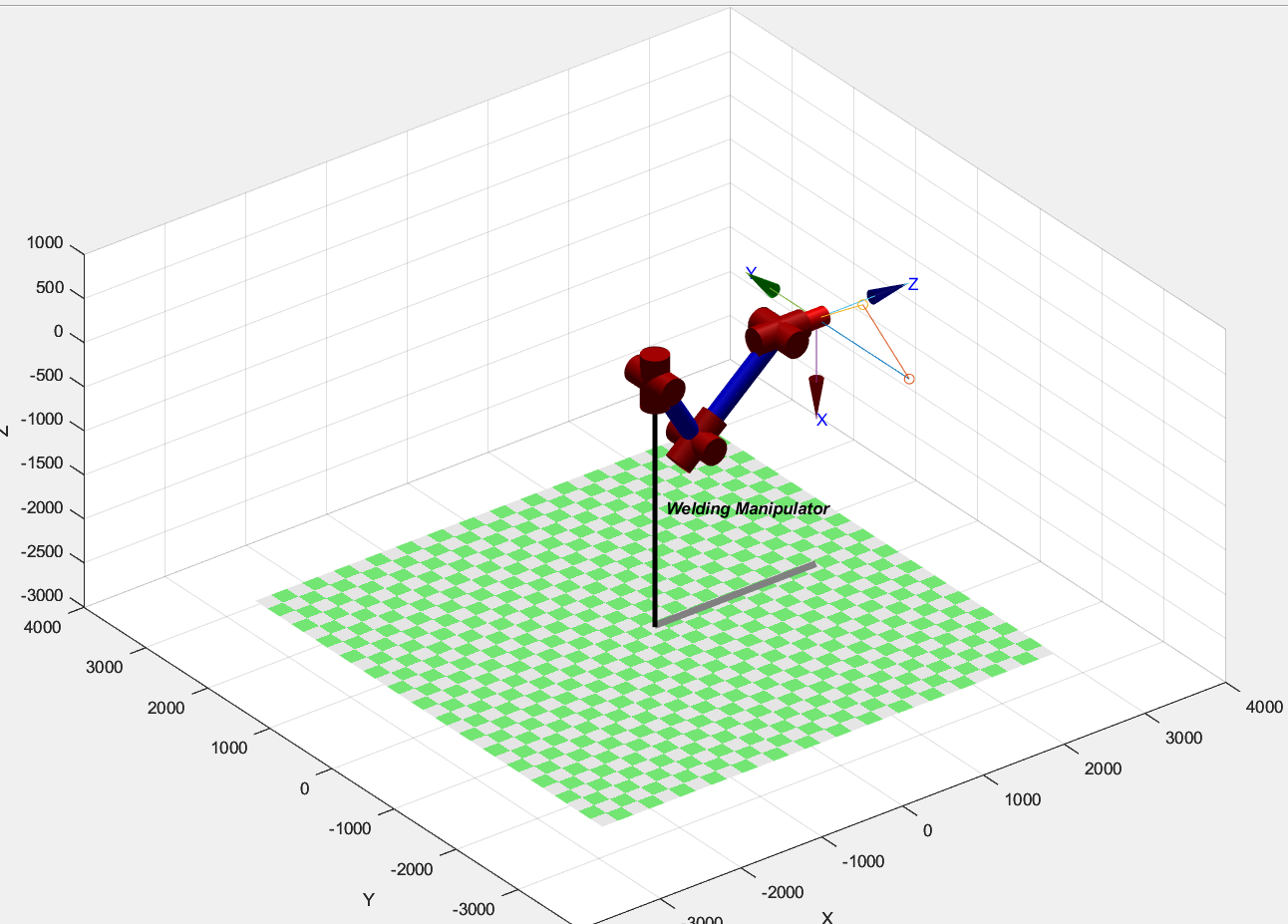
*Fig-4*

1. Rectangle (L = 1500, B = 500)



*Fig-5*

1. Triangle (B = 1500, H = 500 Units)



*Fig-6*

**References:**

[1] Design and Analysis of six DOF Robotic Manipulator V G Pratheep1, M Chinnathambi2, E B Priyanka3, P Ponmurugan4 and Pridhar Thiagarajan [IOP ConferenceSeries: Materials Science and Engineering](https://iopscience.iop.org/journal/1757-899X), [Volume 1055](https://iopscience.iop.org/volume/1757-899X/1055),

[International Virtual Conference onRobotics, Automation, Intelligent Systems and Energy (IVC RAISE 2020) 15th December 2020, Erode, India](https://iopscience.iop.org/issue/1757-899X/1055/1)

[2] Referred to Slides which were taught in class.