# TASK – 1

**OBJECTIVE** – To develop a fuzzy logic controller for a robot to reach its target in the given graphical user interface. We need to build rules based on the fuzzy controls using fuzzy inference system and compare it with different cases to obtain the data for the application in the ANFIS.

In our first step, we build the circuit for the simulation block to provide an environment suitable for the robot to reach the destination in the graphical user interface. We are provided with a five-block and fourteen-block environment for the movement and testing of robots to achieve the results.

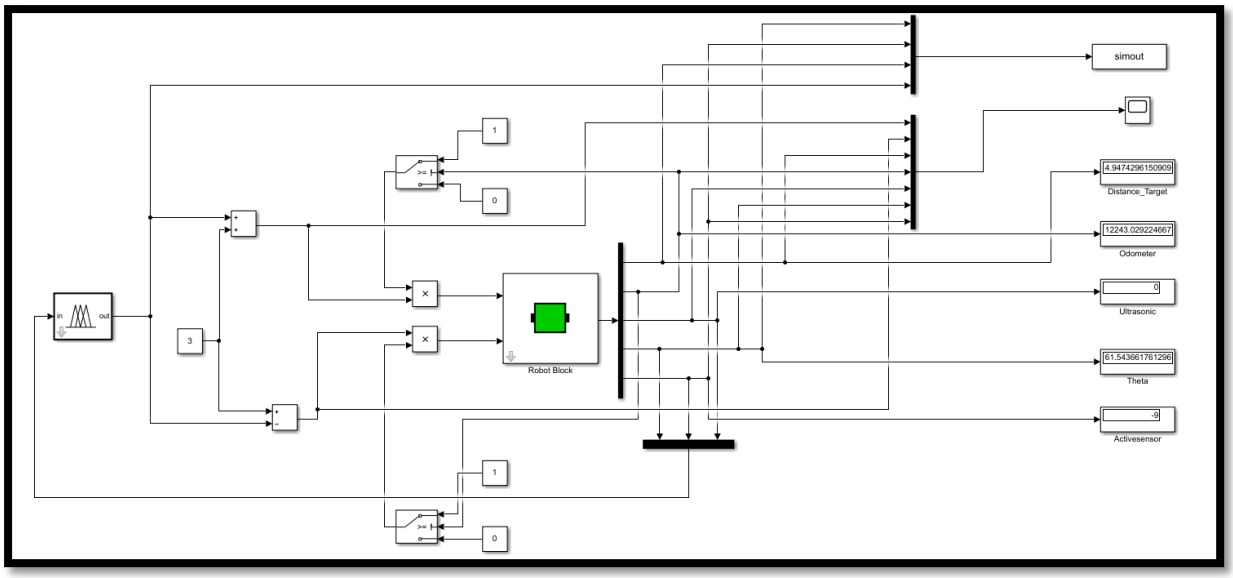



***Figure 1: Robot blocks for simulink model***

Figure 1 shows the robot block for the Simulink model. It consists of a fuzzy inference system, robot block, switches for the wheel of the robot. The output of the system consists of Odometer to obtain the value of the distance travelled by the robot, value to obtain the distance between the robot and the target, theta angle to obtain the direction and position of the movement of the robot. It also consists of active Sensor and ultrasonic Sensor to find the distance between the obstacle and the robot. This helps in maintaining the distance between robot and obstacle. The input consists of two wheels, and a constant block is attached to the wheel to set the velocity of the robot. The switches help to maintain the operation a threshold of the wheel. The simulation output is

obtained through the block *To workspace*, which directs the data to the Matlab workspace. This data is then used for generating rules for the ANFIS inference system.

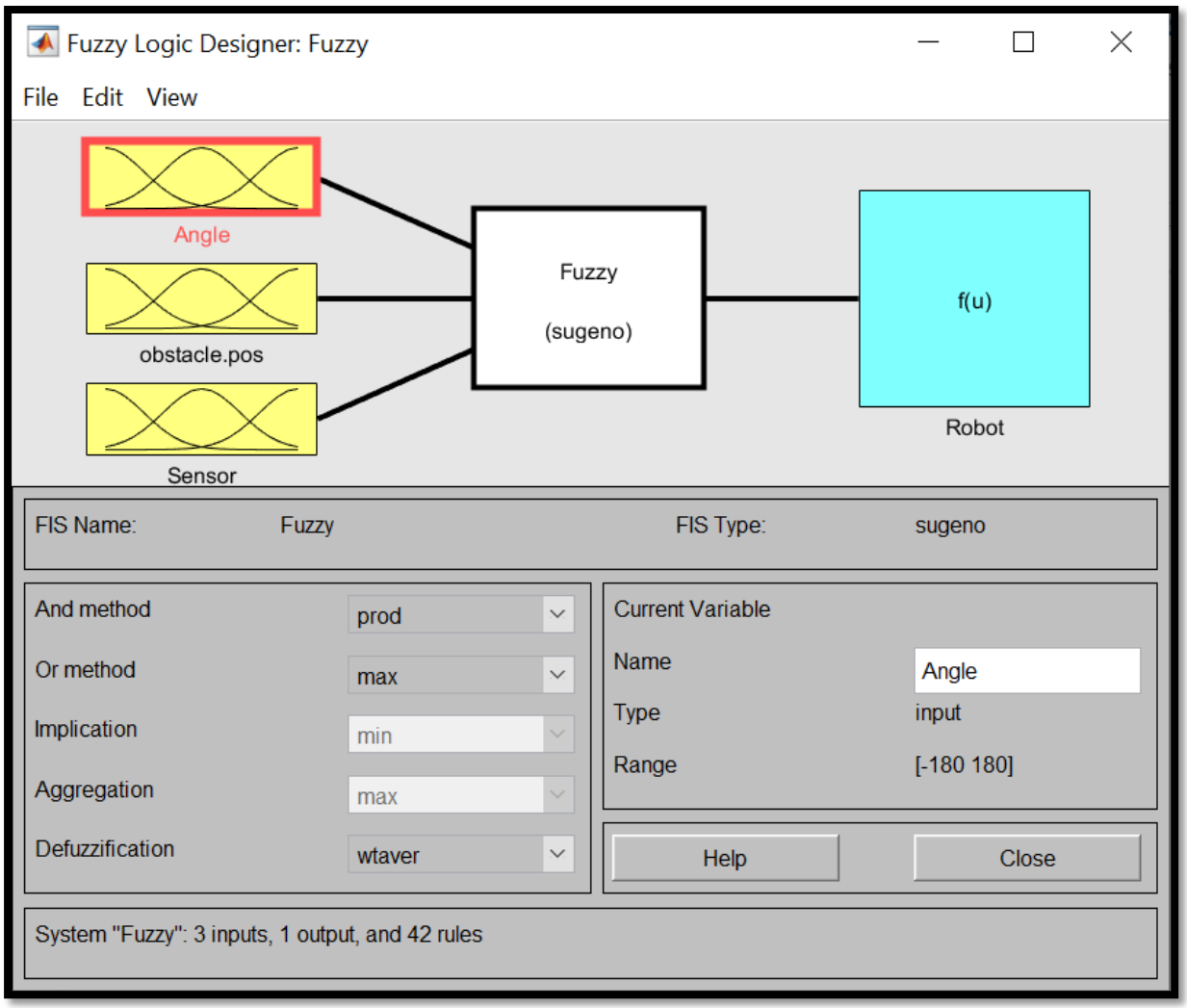



***Figure 2: Fuzzy inference system for the simulation***

Figure 2 shows the FIS system for fuzzy logic. It consists of three inputs for the robot. The angle refers to the direction and movement of the robot, obstacle.pos refers to the position of the obstacles in the graphical user interface. The Sensor is used to predict the distance from robot to obstacle to avoid clashing with the obstacles. The output consists of odometer reading and range between the target and the robot. We set the suitable member function and its value in the Fuzzy FIS. We have established a total number of 42 rules for the ideal functioning of the robot using the FIS.


***Figure 3: Rules for Fuzzy inference systems***

The rules provided in the Fuzzy FIS is suitable for the robot to move in the graphical user interface to reach the target. The output(Robot) member function consists of left slow, right slow, left fast, right fast and None to control the movements of the robots based on the factors acting in parallel in the GUI.

The input member function Sensor consists of Nearby_obstacle, Faraway_obstacle and None to predict the status of the robot from the obstacles. The angle is one of the input functions to carry the member values such as positive, negative, alpha, beta and zero. The obstacle.pos consist of values of position of the obstacle in the GUI environment.

All such factors will be responsible for obtaining the ideal result for the simulation.

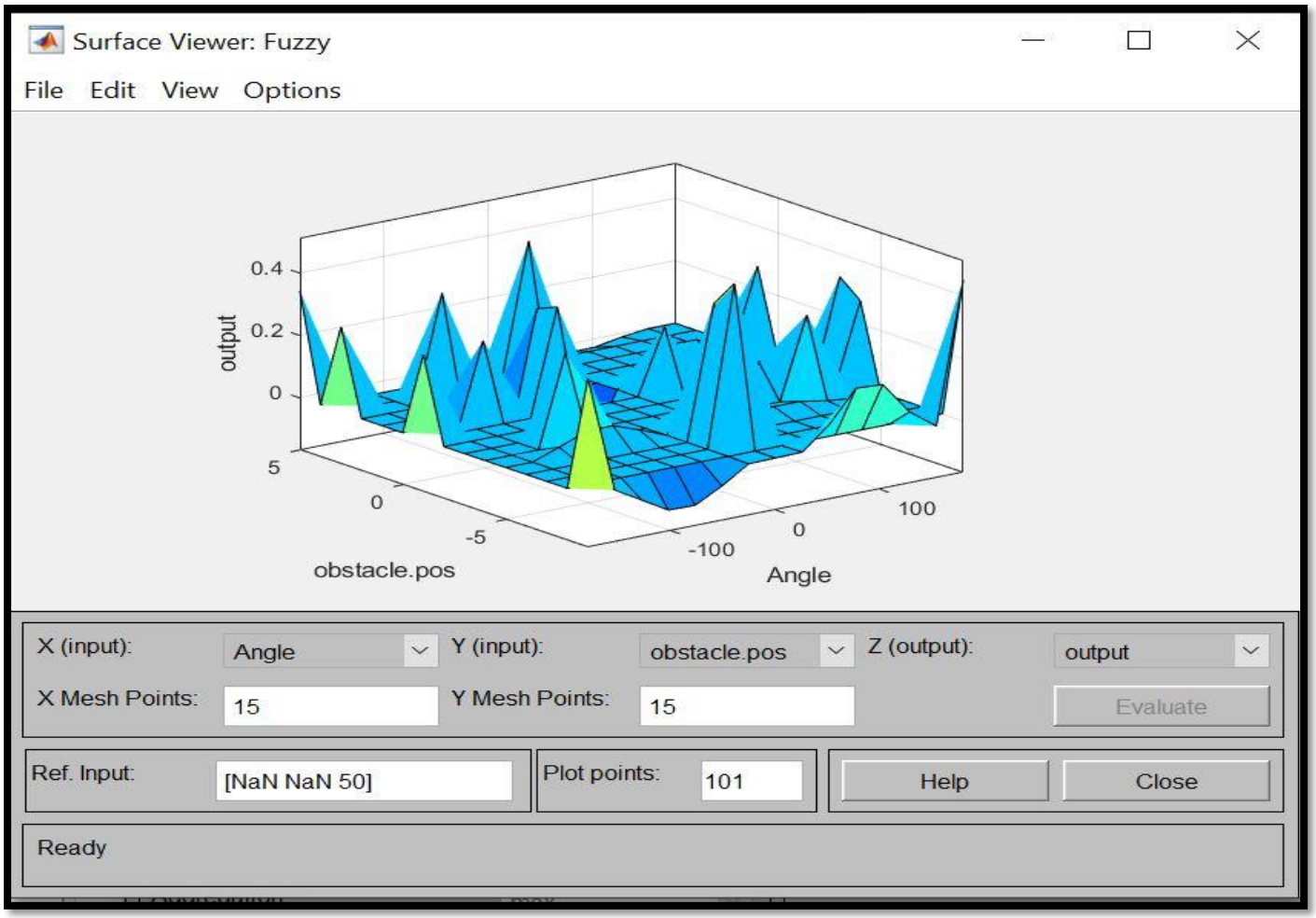



***Figure 4: Fuzzy FIS rule viewer map***

Figure 3 shows the map view of the FIS rules based on the inputs and the output. We use three input and one output for our Fuzzy FIS. As we have built our Fuzzy FIS, we upload the *.fis* file to the fuzzy inference system controller in the robot simulation file. We set a value for the constant block that is responsible for the velocity of the robot. Hence, the robot block consists of three inputs, the left wheel, the right wheel and the velocity of the robot.

As we simulate a specific time interval, the robot reaches the target twice at a particular angle and velocity. We record these values to obtain the data for the performance enhancement in the next task. Similarly, we take the record of simulations for four different cases that would set the input parameters to the robot for successful reach to the target.

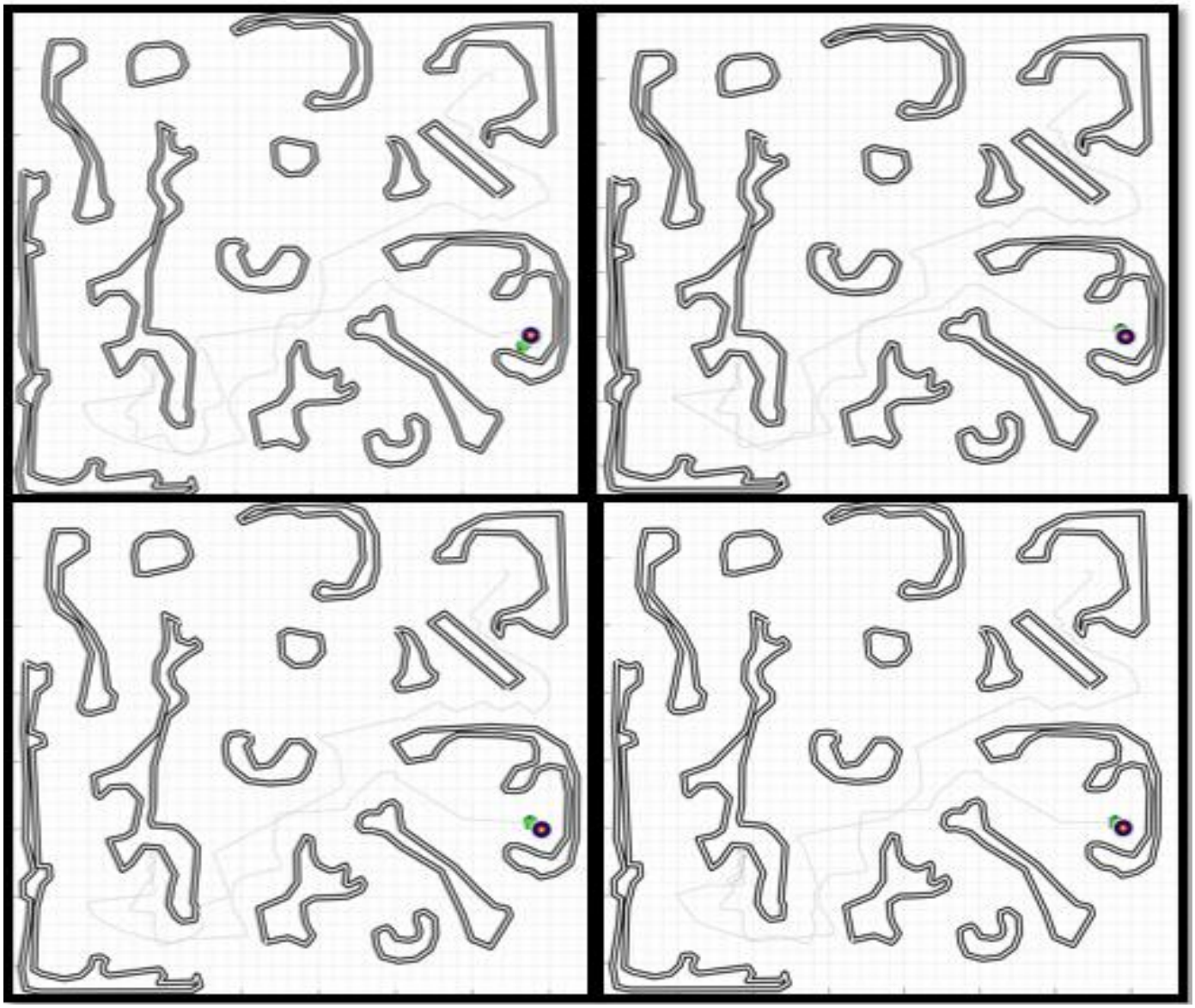


**Figure 5: Robot trajectory using Fuzzy inference system shows the output of case 1 to case 4 from top-most left in the clockwise direction**

The obtained outputs show that the robot takes different paths based on the velocity and the starting angle given to the system for four different cases.

The input parameters for the four cases are as follows –

| CASE | STARTING ANGLE | VELOCITY | SIMULATION STEP |
|---|---|---|---|
| I | 3.14 | 1 | 0.050 |
| II | 1.696 | 3 | 0.050 |
| III | (-0.126) | 1.5 | 0.050 |
| IV | 0.785 | 1.5 | 0.050 |

The starting angle for the robot block varies from (-pi) to (pi).

As we have seen from figure 1, a Simulink block to move the obtained data as per the simulation time is sent to the workspace in the Matlab command window. The data obtained are stored in the names of *case1.dat, case2.dat, case3.dat* and *case4.dat*. All the dataset consists of time, distance to target value, angle and sensor values.

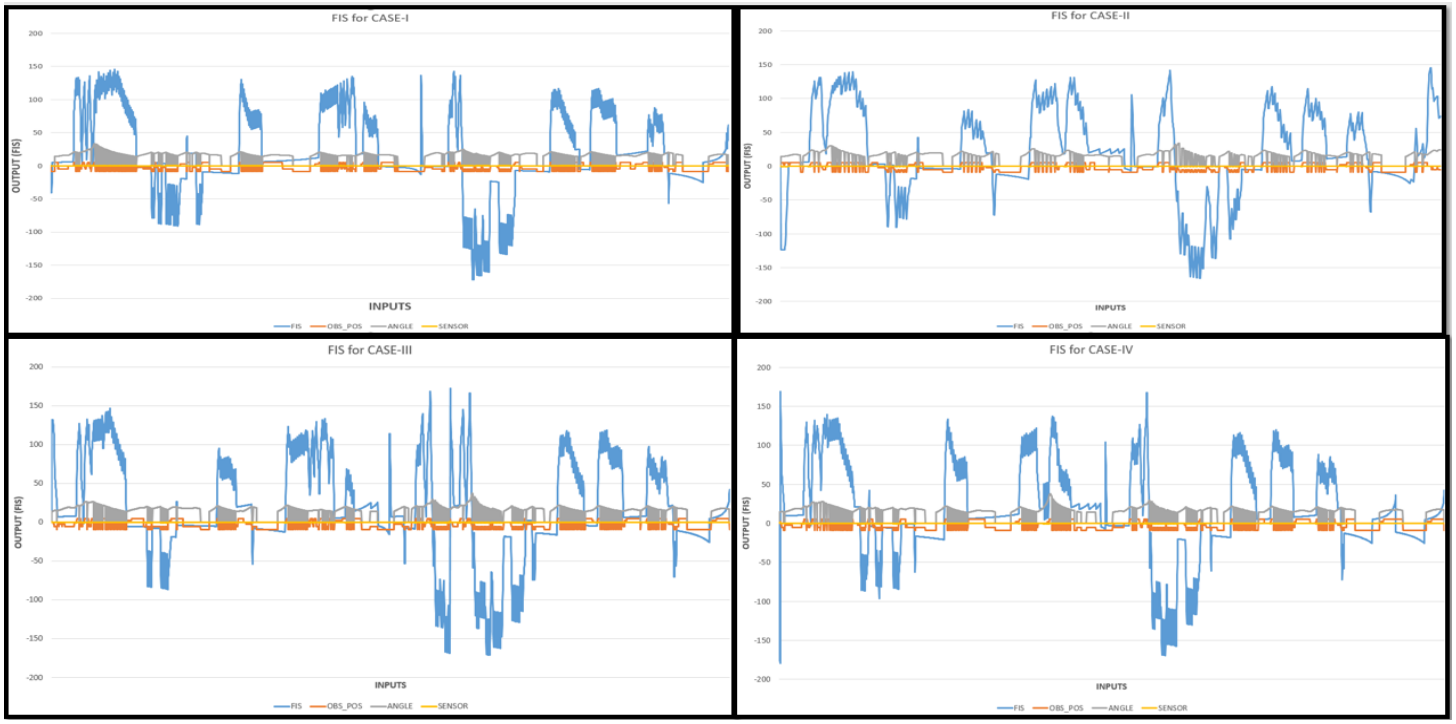



*Figure 6: The above graph shows the performance criteria for case 1 to case 4 in clock-wise direction*

The above figure shows the performance criteria for the inputs in the fuzzy logic FIS system. By combining all the data of four cases, we obtain the data for 12484 units. The data preparation takes place by converting the data(.txt) file to (.dat) file. The combined data is named as *trainingdata.dat,* which is then shuffled randomly to training with the ANSIS neural networks in the next task.

**SUMMARY** - So, we complete this task by creating a fuzzy control FIS system for the robot control in the Matlab simulation environment. We vary the starting angles and the velocity of the robot to adjust the object in such a way that it hits the target twice in different target locations. The data obtained will be used for training the data in ANFIS fuzzy inference system.

# TASK - 2

**OBJECTIVE** – To develop a Neural controller or adaptive neural fuzzy inference system based on the data received from the task one and optimise the performance of the robot in terms of the path travelled to the target.

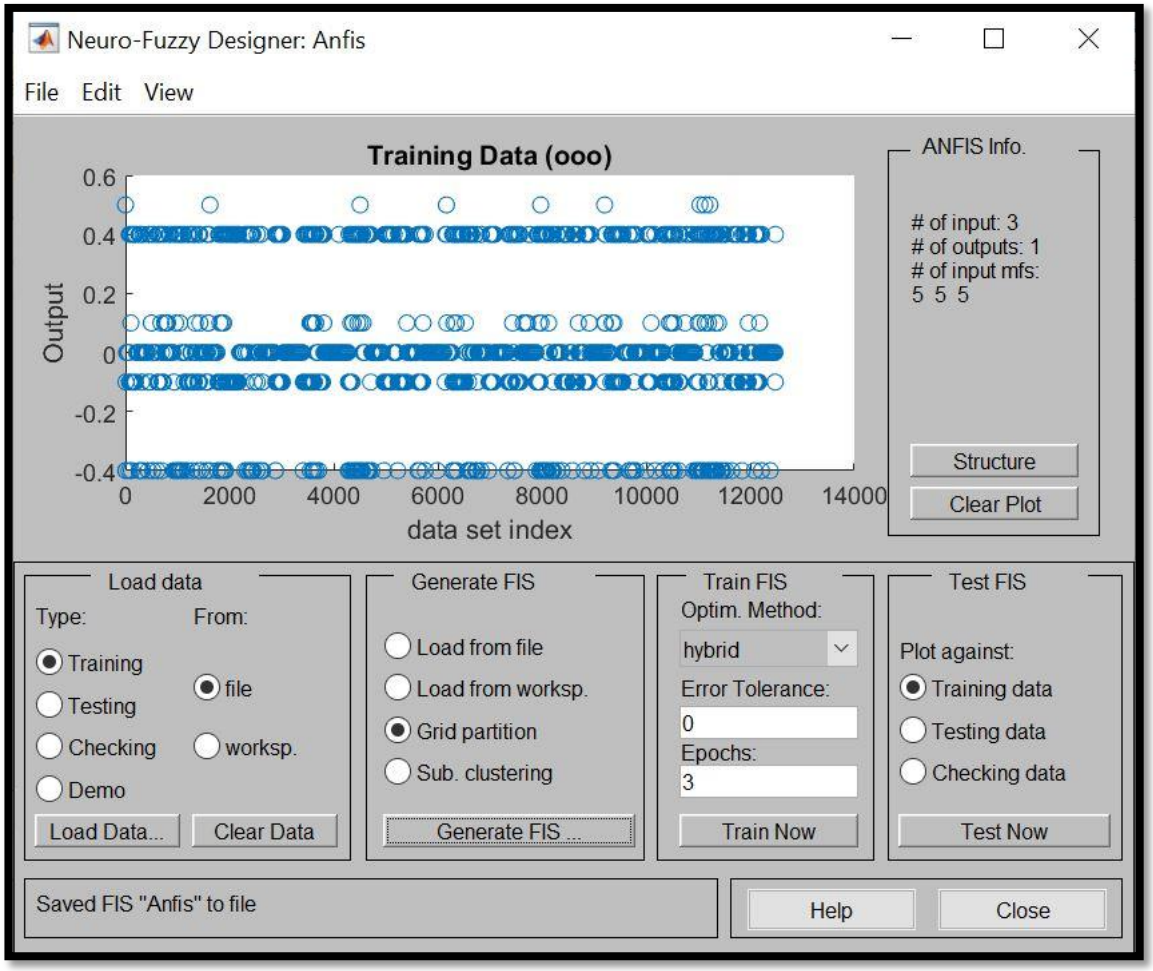



*Figure 7: Anfis FIS to load the training data.*

In this task, we will use the data obtained from the task 1 and upload the data to train the Neural Network. We upload the *trainingdata.dat* into the training section of *anfisedit* in the Matlab command window.

We use the number of member functions [5 5 5] or alternatively [5 6 3] for the training the neural networks with the available data. We use 100 epochs for training the data which is ideal to generate the network that either would not overfit or underfit the data. We have used the linear mode of member functions

in the output. We use the option *Train Now* to start training the data. The Figure 7 shows the model of the available training data.

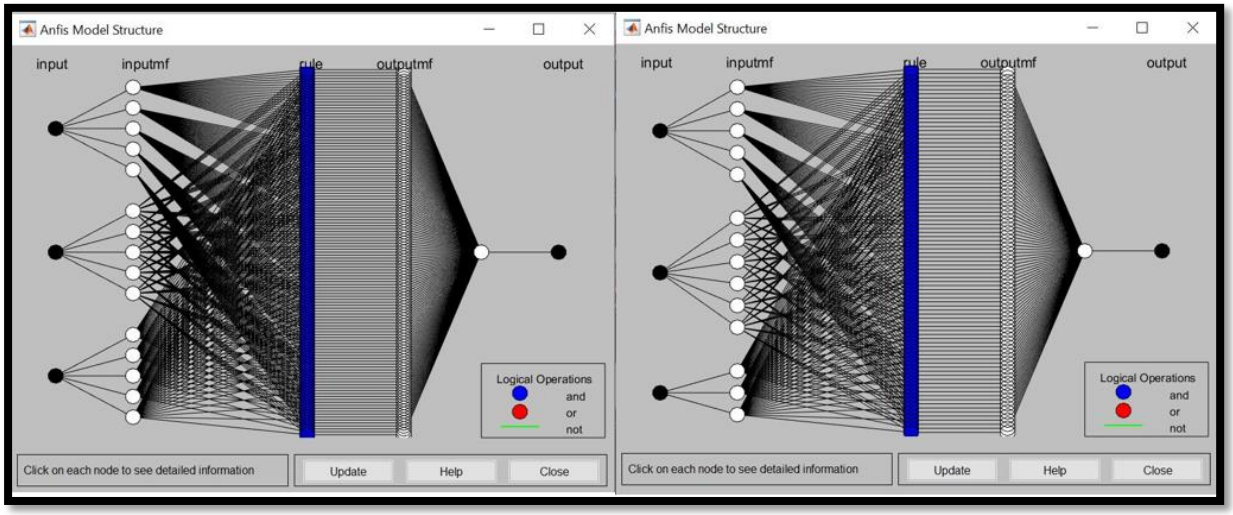



*Figure 8 shows the Anfis Model structure of training data.*

Figure 8 shows the model structure of the training data. The one on the left is the [5 5 5] membership functions to the three inputs and the one on the right is the [5 6 3] membership functions to the three inputs. Let us perform the ANFIS operation for the first task.

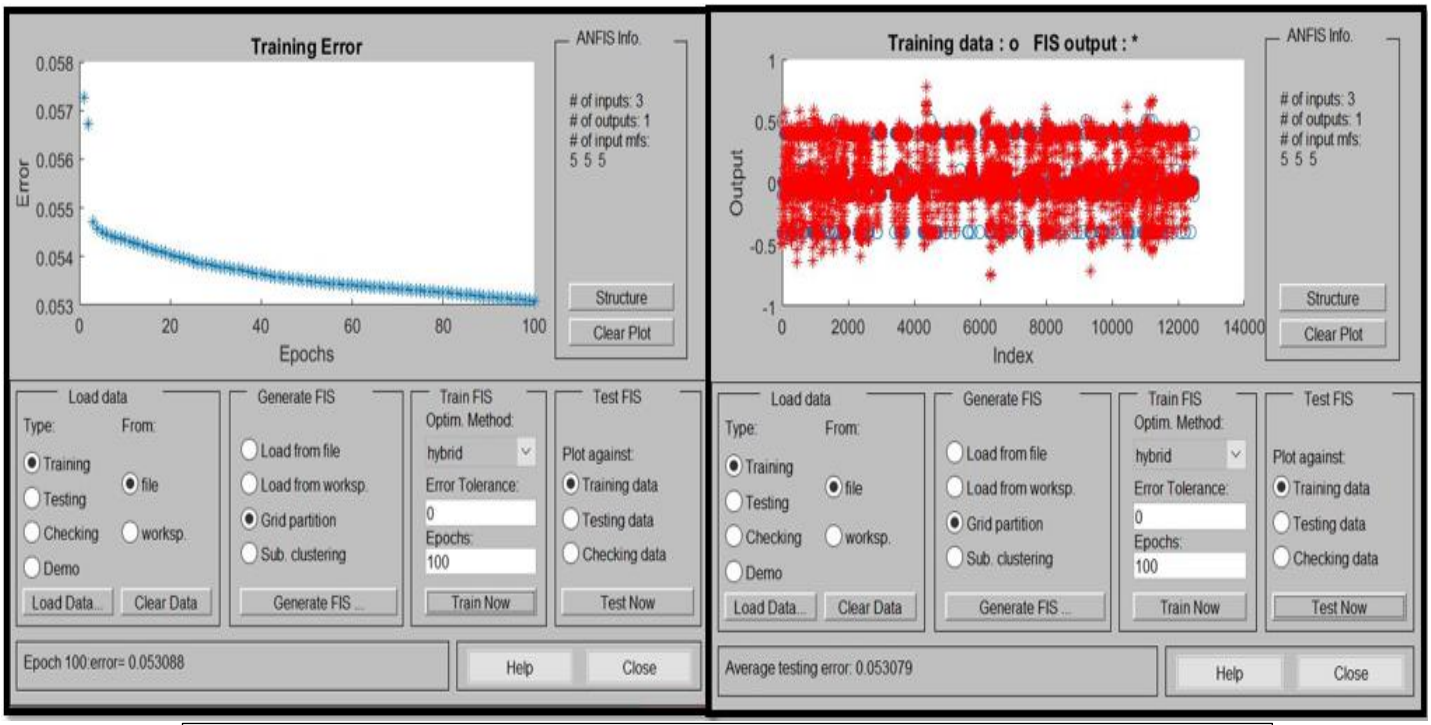
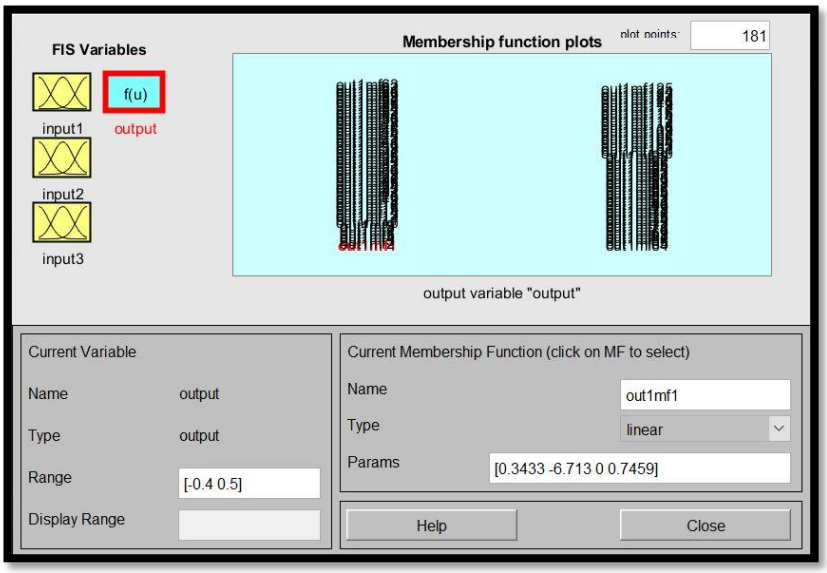



***Figure 9 shows the trained result and test result of the training data in ANFIS***

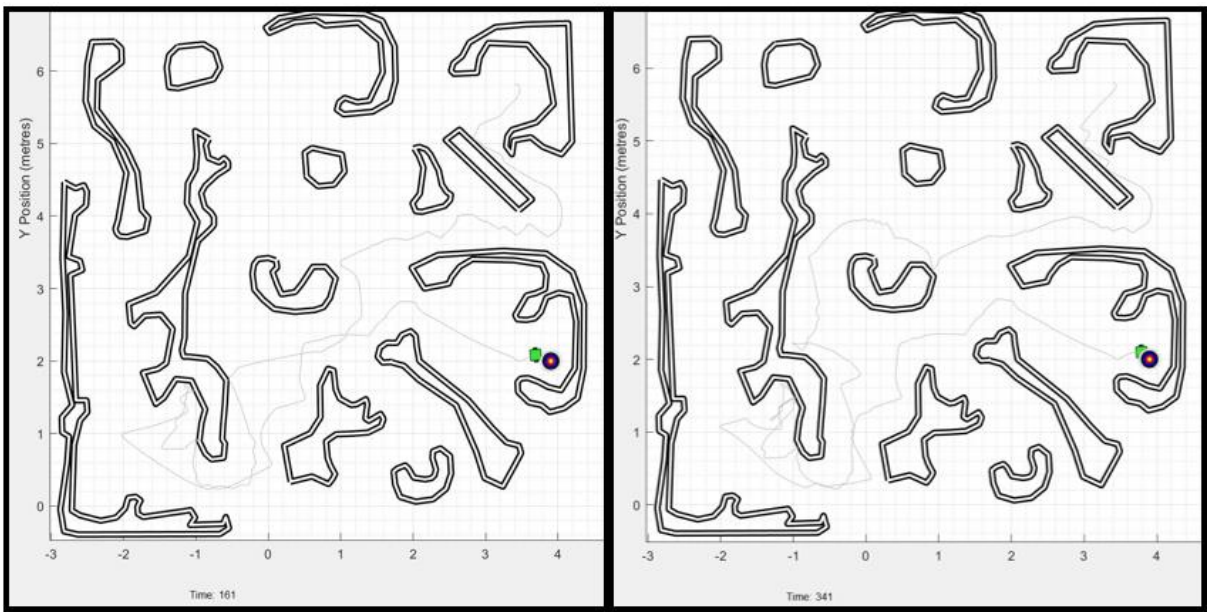
The Figure 9 shows the test result on the right side image which resembles the Figure 7 of training data. This shows that the image has trained well and testing the data provides an complete overlap of the training data in the red colour as provided in the figure.




The above figure shows the output member function of ANFIS fis that is generated throughout the training procedure. The neural networks develops a rule base by itself.

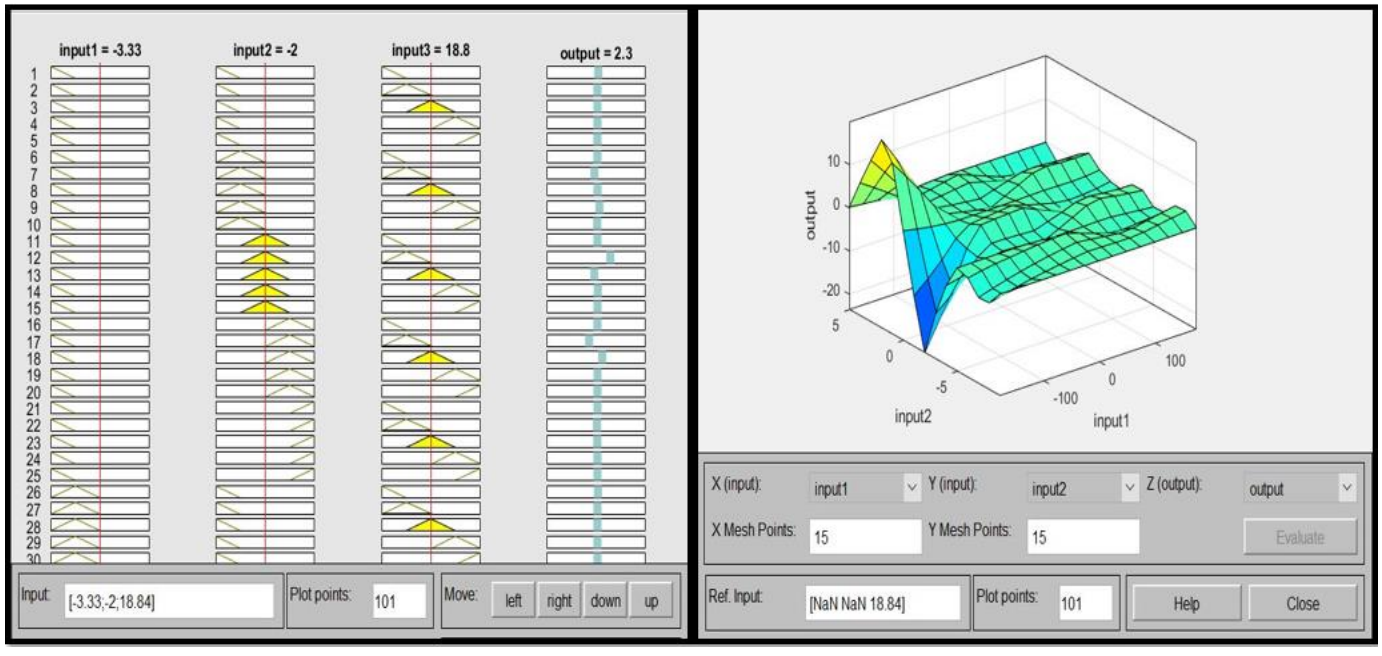
We now save this ANFIS file as *Anfis.fis* and use this data in our simulation model as shown in the figure 1 for the same set or different set of starting angles and variation in the velocity.

| CASE | STARTING ANGLE | VELOCITY | SIMULATION STEP |
|---|---|---|---|
| I | 0.785 | 1.5 | 0.050 |
| II | (-0.126) | 0.8 | 0.050 |

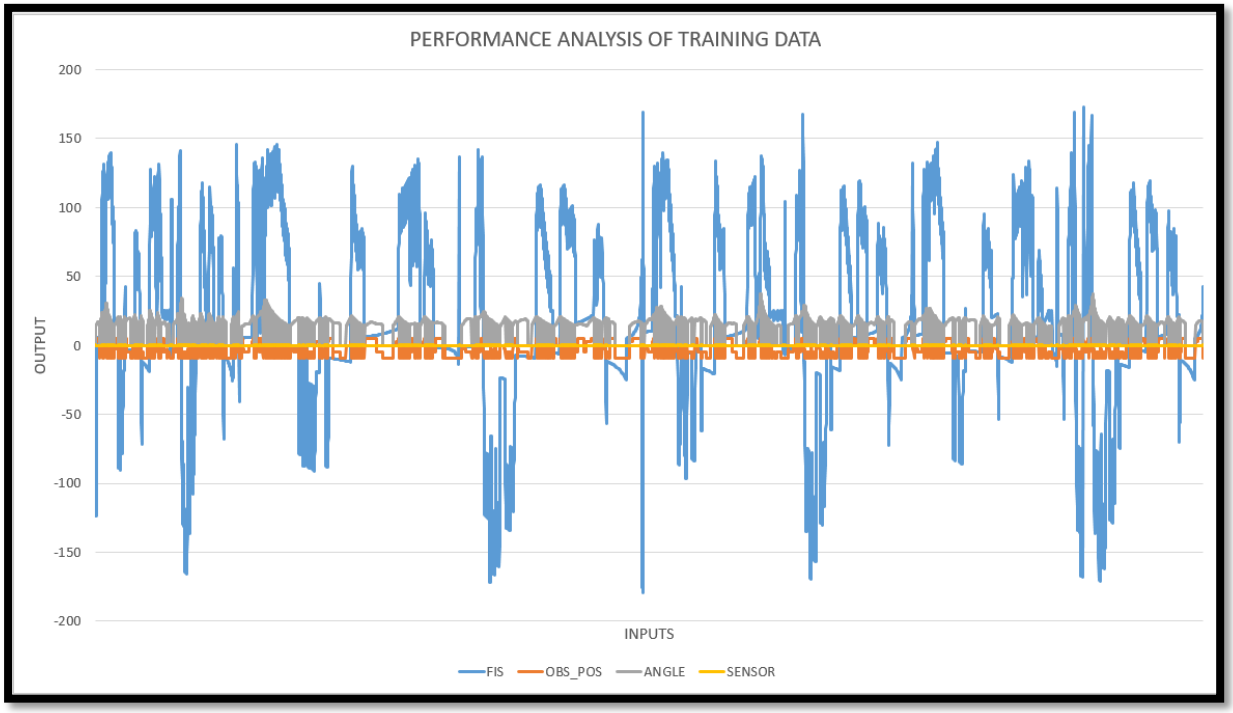



The above figure shows the result obtained for the starting angles and velocity of the robot. We use a fourteen obstacle graphical user interface. It reaches the target twice with high efficiency. This is also applicable for any angle or velocity. The movement of robot is highly optimized without coming in contact with any obstacle in the environment.

Hence, we can see that the ANFIS has generated a lot more base rules for the FIS model and so the robot is optimized with high efficiency. More number of rule base is created for the inputs from the task 1 in the current task.



The above figure shows the rules generated by the ANFIS for the training data. The image on the right shows the rule map for the ANFIS generated data.




Similarly, we get the outputs for starting angles 1.696 and 3.14 with common simulation step of 0.050 and velocities of 1 unit.

| STARTING ANGLE | VELOCITY | ODOMETER READINGS |
|---|---|---|
| 1.6 | 1 | 13154 |
| 3.14 | 1 | 10568 |
| 0.785 | 1.5 | 12665 |
| (-0.126) | 0.8 | 17294 |

SUMMARY – In our Task-2, we have developed a neural network using the data obtained from task 1 and applied the training procedure. The images obtained proved that the training performed well.