Fuzzy Logic Final Project

Experiment for Fuzzy Logic Based Control System for AGV Systems

Ahmed Faris Bingöl

BLU 513E

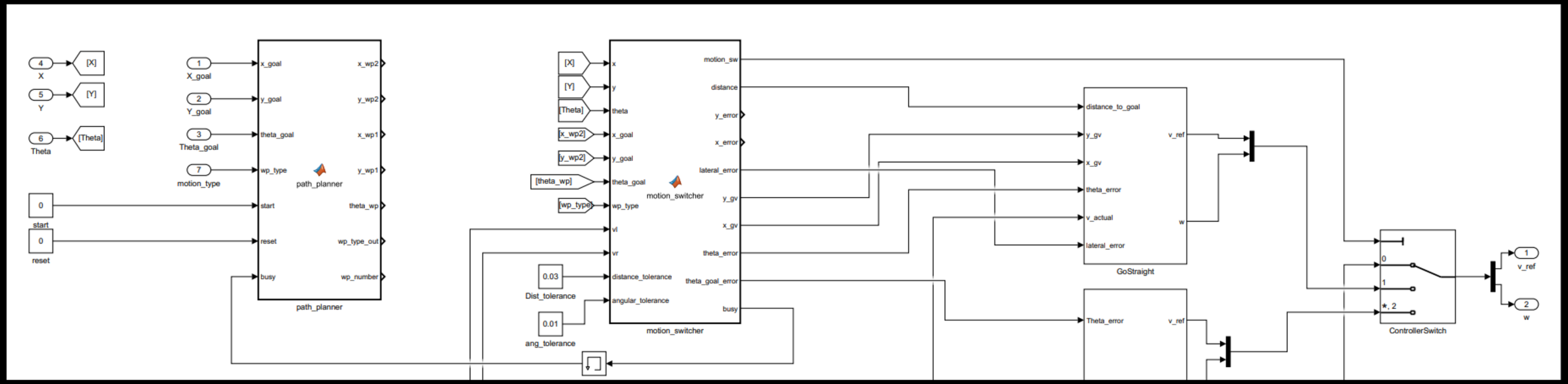
What is AGV?

Robee (Makhina Robotics)

Classic Control

Current Method in Product

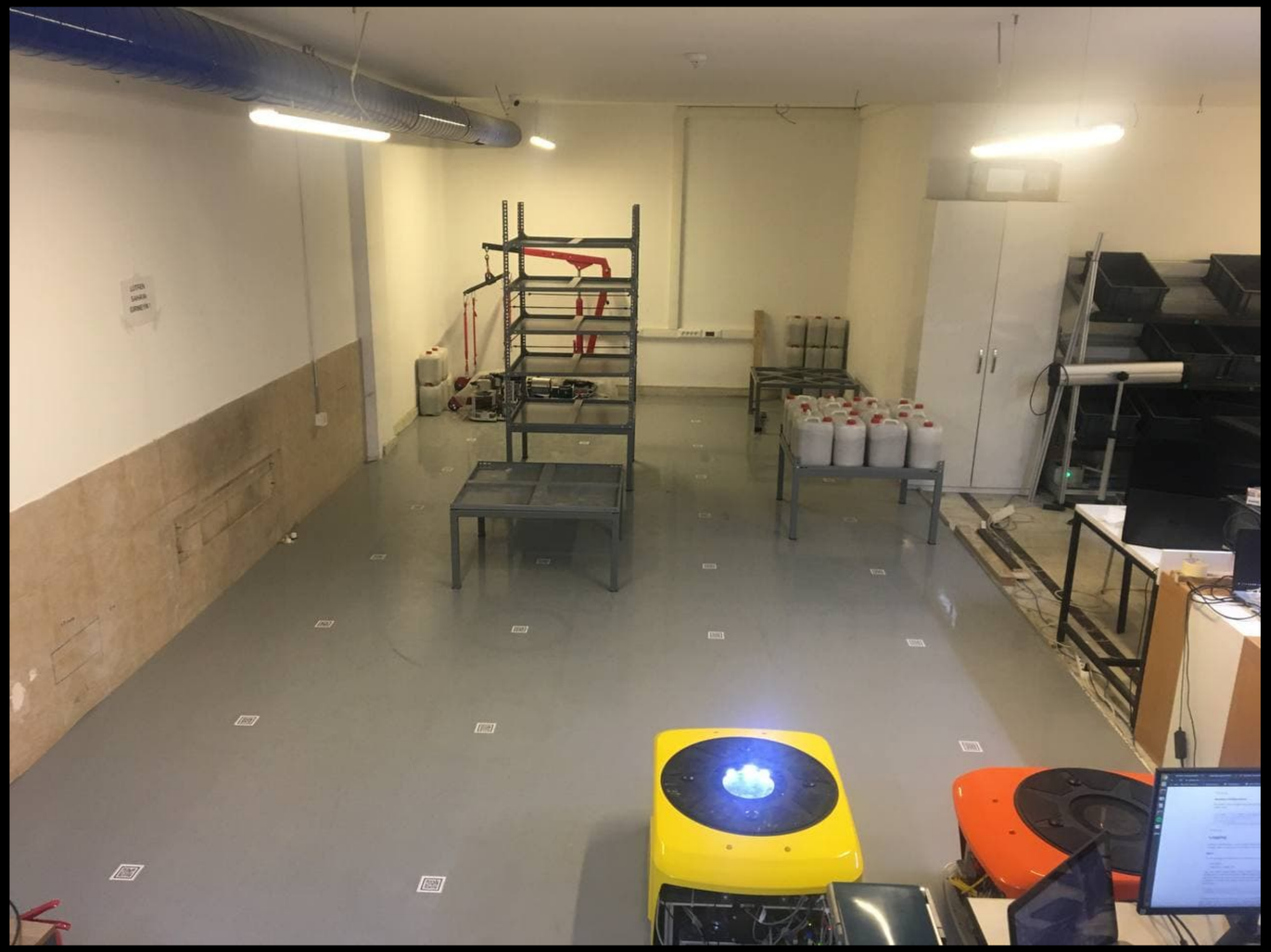
- 5 Input Parameters: X_Position, Y_Position, Theta Angle, X_Destination, Y_Destination
- 2 Output: Linear Velocity, Angular Velocity



Collecting Data

Ground Truth

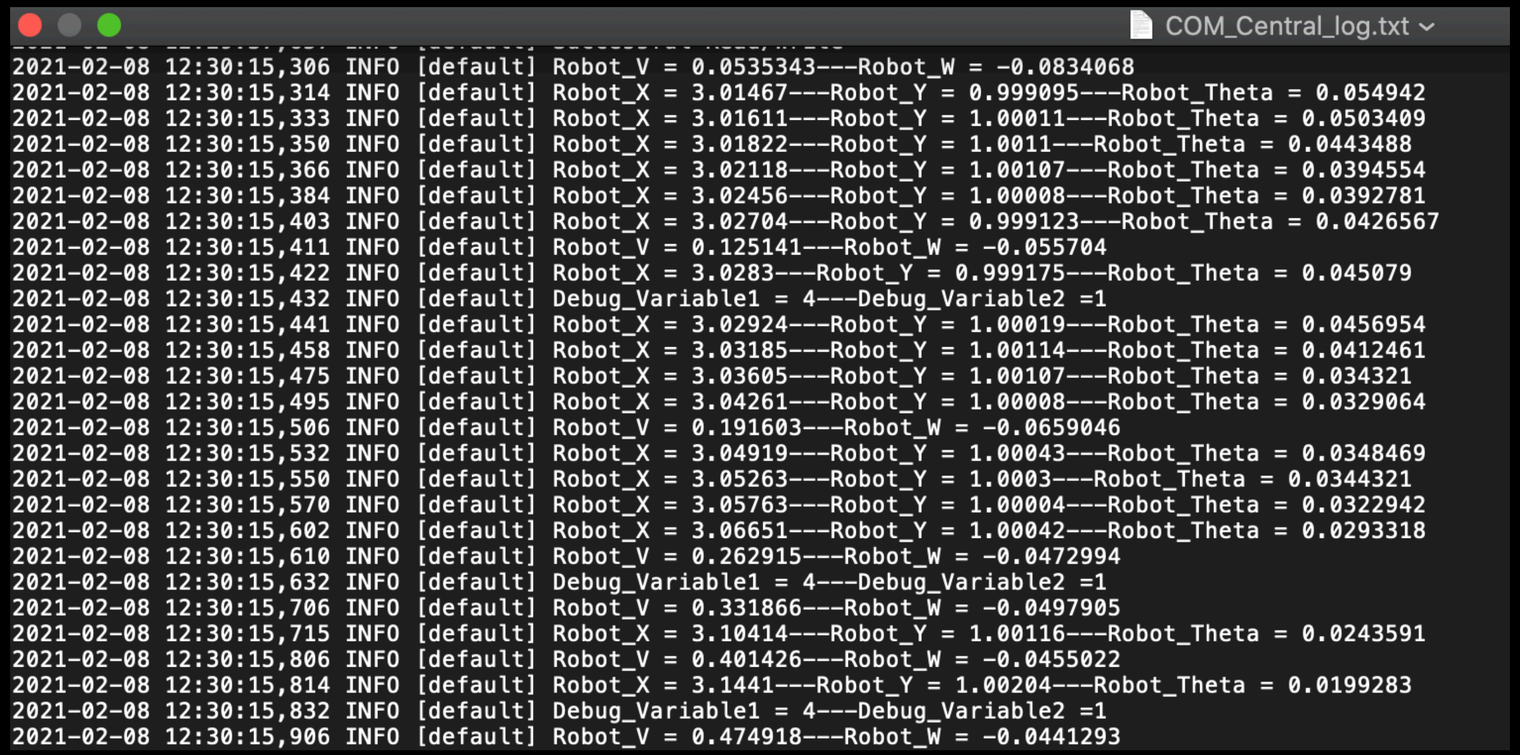
In this project, i collected to ground truth variables from our current controller system which is dynamically calculating “ Distance To Goal” and “Angle Error” for every time step. We calculate odometry value from encoders and updated with processing QR code from ground to update it is position variables



Collecting Data

From the Bottom

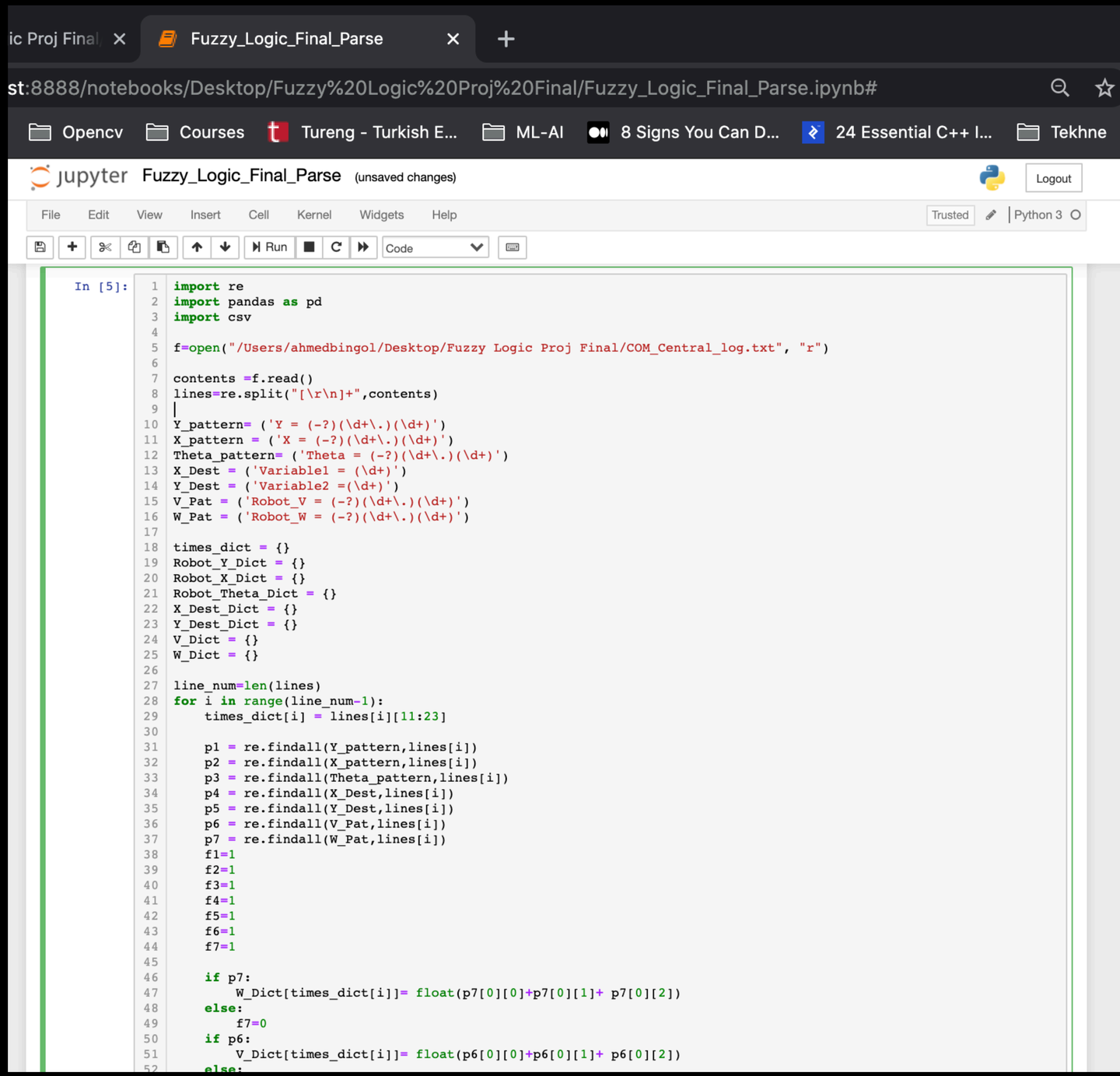
I collected raw data with driving Robee more than 50 minutes. This raw data can be shown at COM_Central_log.txt file. The data sample taken in every 10 ms.



Collecting Data

Process Raw Data

- This raw data is converted to a .csv file data which can be later be used at MatLab ANFIS editor to generate Fuzzy Inference System.
- The Resulting csv file shown at next slide.



Collecting Data

Process Raw Data

- Obtained Data from Robot:
- At every time step, we obtain some part of our 7 Input+Output variable
- We had NaN values there.
- Total we had around 31000 data samples for 7 Input-Output and Time= 31000 x 8 Matrix !

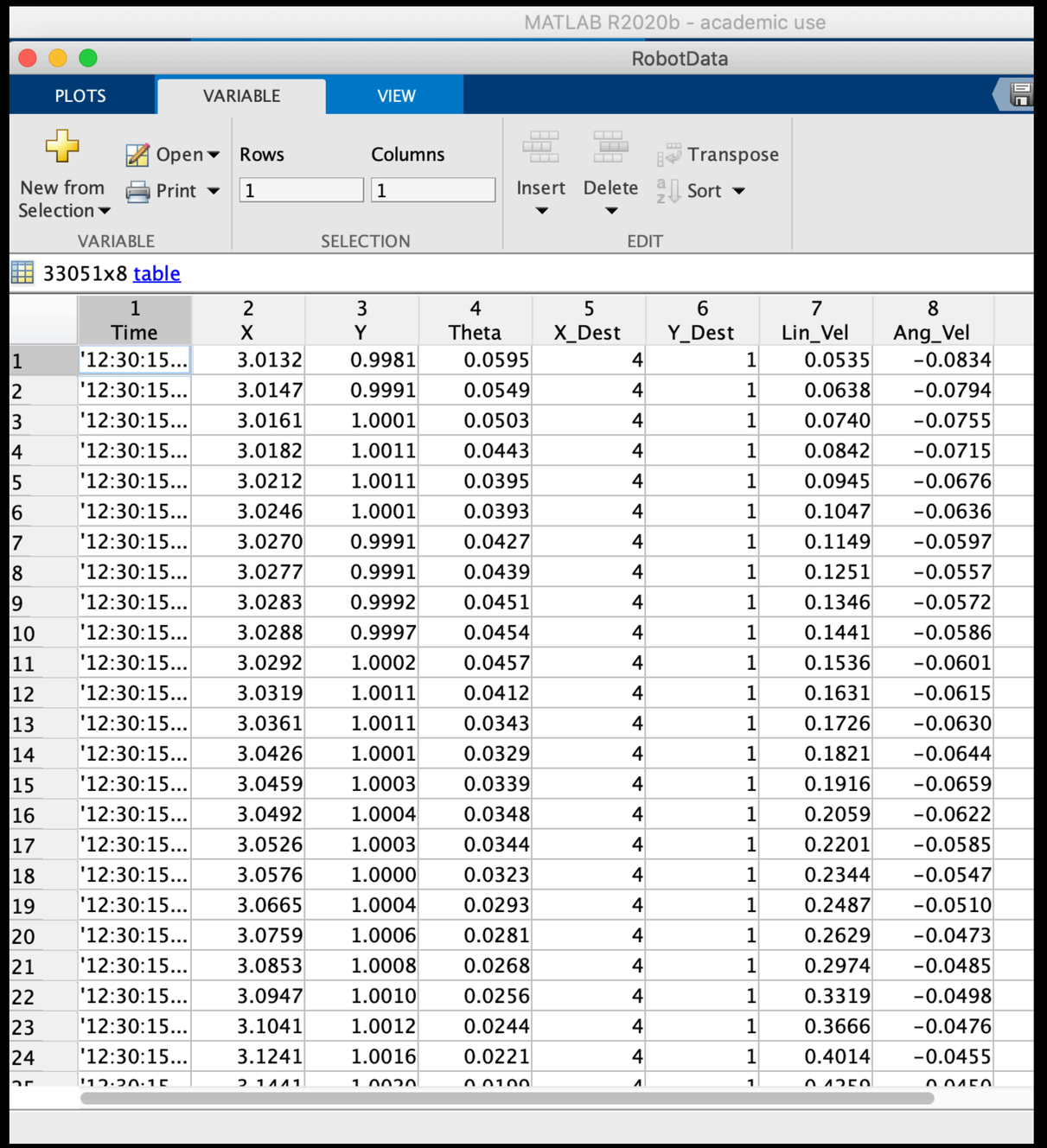
Robot_Data.csv — Open with Numbers

Time	X	Y	Theta	X_Dest	Y_Dest	Lin_Vel	Ang_Vel
12:30:15,306	NaN	NaN	NaN	NaN	NaN	0.0535343	-0.0834068
12:30:15,314	3.01467	0.999095	0.054942	NaN	NaN	NaN	NaN
12:30:15,333	3.01611	1.00011	0.0503409	NaN	NaN	NaN	NaN
12:30:15,350	3.01822	1.0011	0.0443488	NaN	NaN	NaN	NaN
12:30:15,366	3.02118	1.00107	0.0394554	NaN	NaN	NaN	NaN
12:30:15,384	3.02456	1.00008	0.0392781	NaN	NaN	NaN	NaN
12:30:15,403	3.02704	0.999123	0.0426567	NaN	NaN	NaN	NaN
12:30:15,411	NaN	NaN	NaN	NaN	NaN	0.125141	-0.055704
12:30:15,422	3.0283	0.999175	0.045079	NaN	NaN	NaN	NaN
12:30:15,432	NaN	NaN	NaN	4.0	1.0	NaN	NaN
12:30:15,441	3.02924	1.00019	0.0456954	NaN	NaN	NaN	NaN
12:30:15,458	3.03185	1.00114	0.0412461	NaN	NaN	NaN	NaN
12:30:15,475	3.03605	1.00107	0.034321	NaN	NaN	NaN	NaN
12:30:15,495	3.04261	1.00008	0.0329064	NaN	NaN	NaN	NaN
12:30:15,506	NaN	NaN	NaN	NaN	NaN	0.191603	-0.0659046
12:30:15,532	3.04919	1.00043	0.0348469	NaN	NaN	NaN	NaN
12:30:15,550	3.05263	1.0003	0.0344321	NaN	NaN	NaN	NaN
12:30:15,570	3.05763	1.00004	0.0322942	NaN	NaN	NaN	NaN
12:30:15,602	3.06651	1.00042	0.0293318	NaN	NaN	NaN	NaN
12:30:15,610	NaN	NaN	NaN	NaN	NaN	0.262915	-0.0472994
12:30:15,632	NaN	NaN	NaN	4.0	1.0	NaN	NaN
12:30:15,706	NaN	NaN	NaN	NaN	NaN	0.331866	-0.0497905
12:30:15,715	3.10414	1.00116	0.0243591	NaN	NaN	NaN	NaN
12:30:15,806	NaN	NaN	NaN	NaN	NaN	0.401426	-0.0455022
12:30:15,814	3.1441	1.00204	0.0199283	NaN	NaN	NaN	NaN
12:30:15,832	NaN	NaN	NaN	4.0	1.0	NaN	NaN
12:30:15,906	NaN	NaN	NaN	NaN	NaN	0.474918	-0.0441293

Collecting Data

Process Raw Data

- Second step written and operated at Matlab. In Fuzzy_Final_Data_Prep.m,
- Scaled the variables and filled empty timestep variables with linear filling missing points.
- Data is 31000x8. First column shows timestep and next 5 column showed to input of our ANFIS model.
- Last 2 column showed output of our classical control system results: Linear Velocity and Angular Velocity



	1 Time	2 X	3 Y	4 Theta	5 X_Dest	6 Y_Dest	7 Lin_Vel	8 Ang_Vel
1	'12:30:15...	3.0132	0.9981	0.0595	4	1	0.0535	-0.0834
2	'12:30:15...	3.0147	0.9991	0.0549	4	1	0.0638	-0.0794
3	'12:30:15...	3.0161	1.0001	0.0503	4	1	0.0740	-0.0755
4	'12:30:15...	3.0182	1.0011	0.0443	4	1	0.0842	-0.0715
5	'12:30:15...	3.0212	1.0011	0.0395	4	1	0.0945	-0.0676
6	'12:30:15...	3.0246	1.0001	0.0393	4	1	0.1047	-0.0636
7	'12:30:15...	3.0270	0.9991	0.0427	4	1	0.1149	-0.0597
8	'12:30:15...	3.0277	0.9991	0.0439	4	1	0.1251	-0.0557
9	'12:30:15...	3.0283	0.9992	0.0451	4	1	0.1346	-0.0572
10	'12:30:15...	3.0288	0.9997	0.0454	4	1	0.1441	-0.0586
11	'12:30:15...	3.0292	1.0002	0.0457	4	1	0.1536	-0.0601
12	'12:30:15...	3.0319	1.0011	0.0412	4	1	0.1631	-0.0615
13	'12:30:15...	3.0361	1.0011	0.0343	4	1	0.1726	-0.0630
14	'12:30:15...	3.0426	1.0001	0.0329	4	1	0.1821	-0.0644
15	'12:30:15...	3.0459	1.0003	0.0339	4	1	0.1916	-0.0659
16	'12:30:15...	3.0492	1.0004	0.0348	4	1	0.2059	-0.0622
17	'12:30:15...	3.0526	1.0003	0.0344	4	1	0.2201	-0.0585
18	'12:30:15...	3.0576	1.0000	0.0323	4	1	0.2344	-0.0547
19	'12:30:15...	3.0665	1.0004	0.0293	4	1	0.2487	-0.0510
20	'12:30:15...	3.0759	1.0006	0.0281	4	1	0.2629	-0.0473
21	'12:30:15...	3.0853	1.0008	0.0268	4	1	0.2974	-0.0485
22	'12:30:15...	3.0947	1.0010	0.0256	4	1	0.3319	-0.0498
23	'12:30:15...	3.1041	1.0012	0.0244	4	1	0.3666	-0.0476
24	'12:30:15...	3.1241	1.0016	0.0221	4	1	0.4014	-0.0455

- Second step written and operated at Matlab. In Fuzzy_Final_Data_Prep.m,
- LinearVelocity Matrix used for train LinearVelocity resulted FIS model 30000x6 data
- AngularVelocity Matrix used for train AngularVelocity resulted FIS model 30000x6 data

Editor - /Users/ahmedbingol/Desktop/Fuzzy Logic Proj Final/Fuzzy_Final_Data_Prep.m*

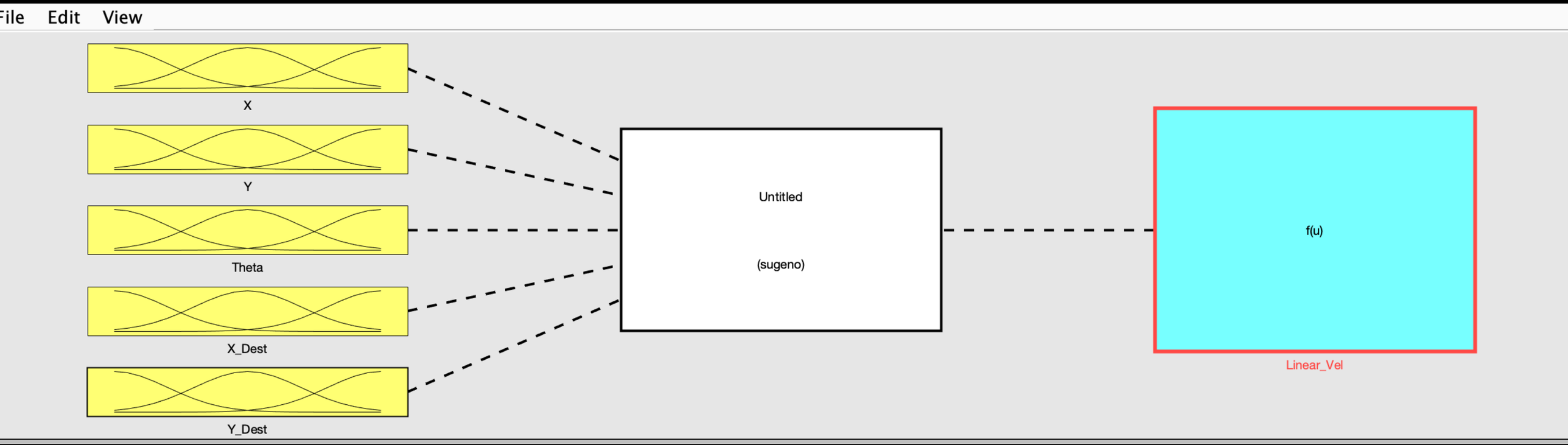
Fuzzy_Final_Data_Prep.m*

```
clear all;
clc;
RobotData = readtable('Robot_Data.csv');
%Filling Empty NaN Value for missing data
%Before feed to ANFIS
FilledX = fillmissing(RobotData.X,'linear');
FilledY= fillmissing(RobotData.Y,'linear');
FilledTheta = fillmissing(RobotData.Theta , 'linear');

Filled_Dest_X = fillmissing(RobotData.X_Dest , 'previous');
Filled_Dest_X = fillmissing(Filled_Dest_X , 'next');
Filled_Dest_Y = fillmissing(RobotData.Y_Dest , 'previous');
Filled_Dest_Y = fillmissing(Filled_Dest_Y , 'next');
Filled_Lin_Vel = fillmissing(RobotData.Lin_Vel,'linear');
Filled_Ang_Vel = fillmissing(RobotData.Ang_Vel,'linear');

RobotData.X = FilledX;
RobotData.Y = FilledY;
RobotData.Theta = FilledTheta;
RobotData.X_Dest = Filled_Dest_X;
RobotData.Y_Dest = Filled_Dest_Y;
RobotData.Lin_Vel = Filled_Lin_Vel;
RobotData.Ang_Vel = Filled_Ang_Vel;

RobotTable1 = RobotData(1:30000,2:7);
LinearVelocity= RobotTable1{:,:};
RobotTable2 = [RobotData(1:30000,2:6) , RobotData(1:30000,7)];
AngularVelocity= RobotTable2{:,:};

```

ANFIS

Training Model

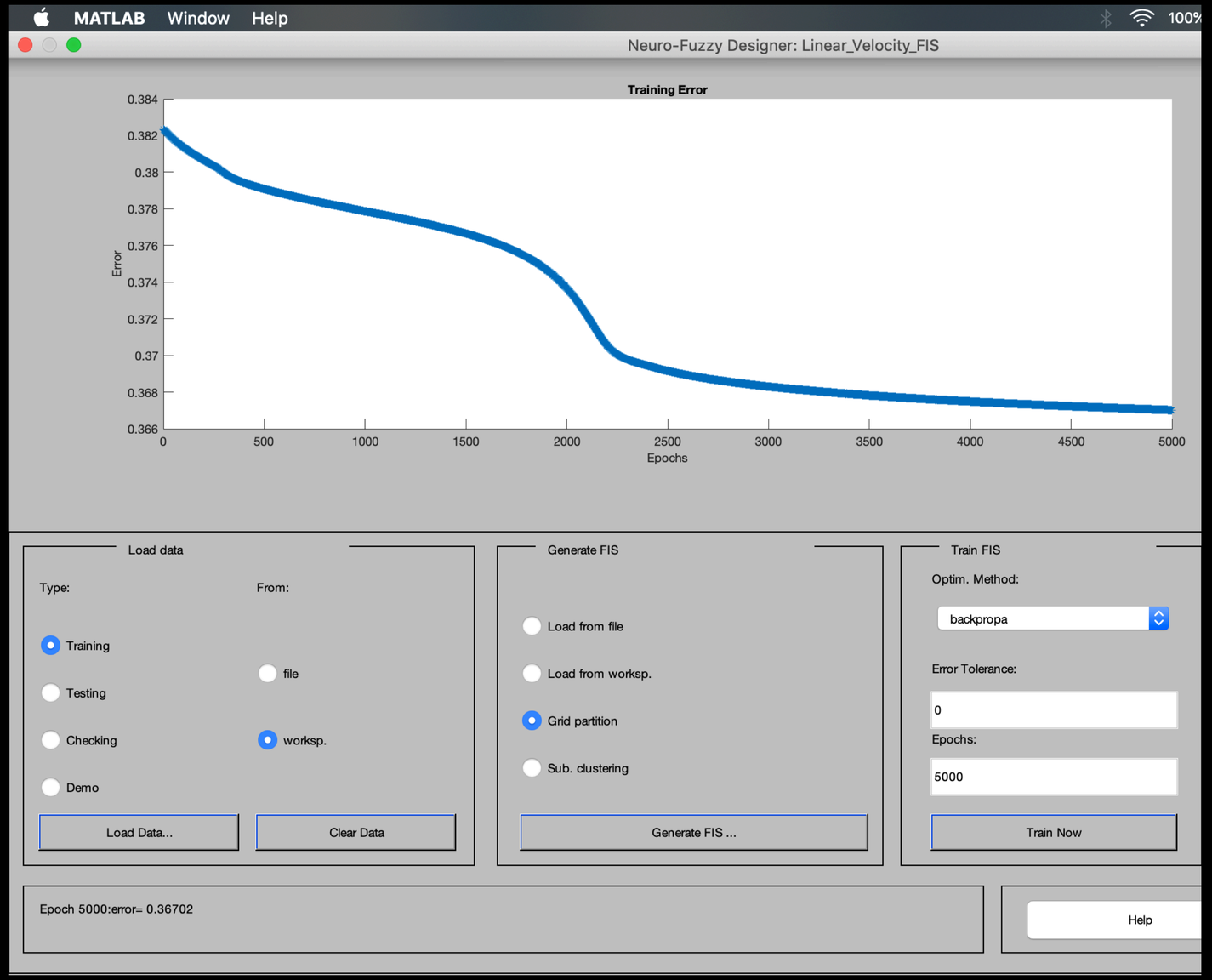
The structure of resulted Input Output relation is shown below for Linear Velocity part. Then i defined the ranges of inputs and MF functions of our system at here.



ANFIS

Training

I used back-propagation as an optimization method and trained the 30 000 different sample point for 5 input 1 output system at 5000 Epoch. The system is converged to Error as 0.36702

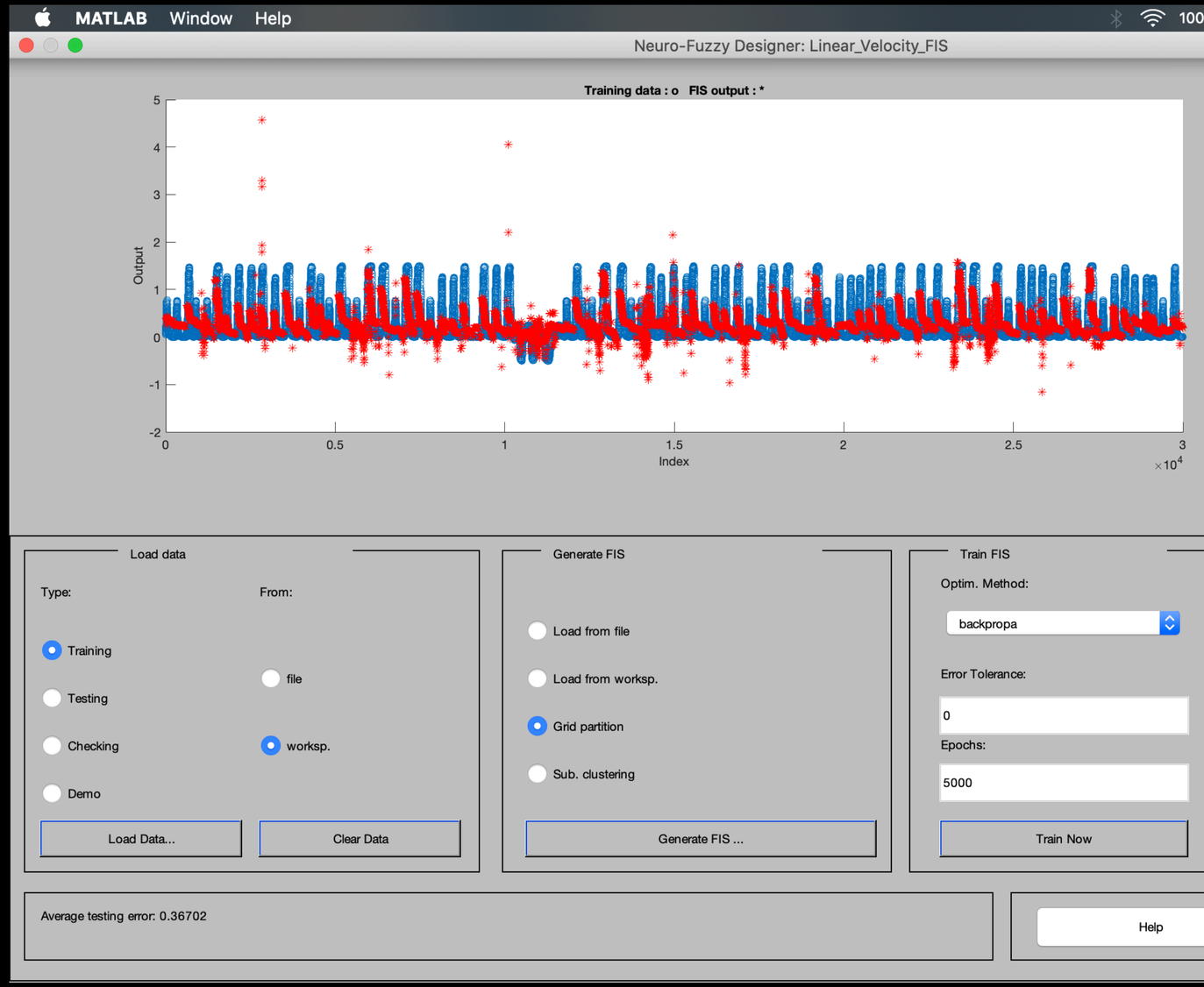


ANFIS

Result

Red points are result of our Model prediction at the graph

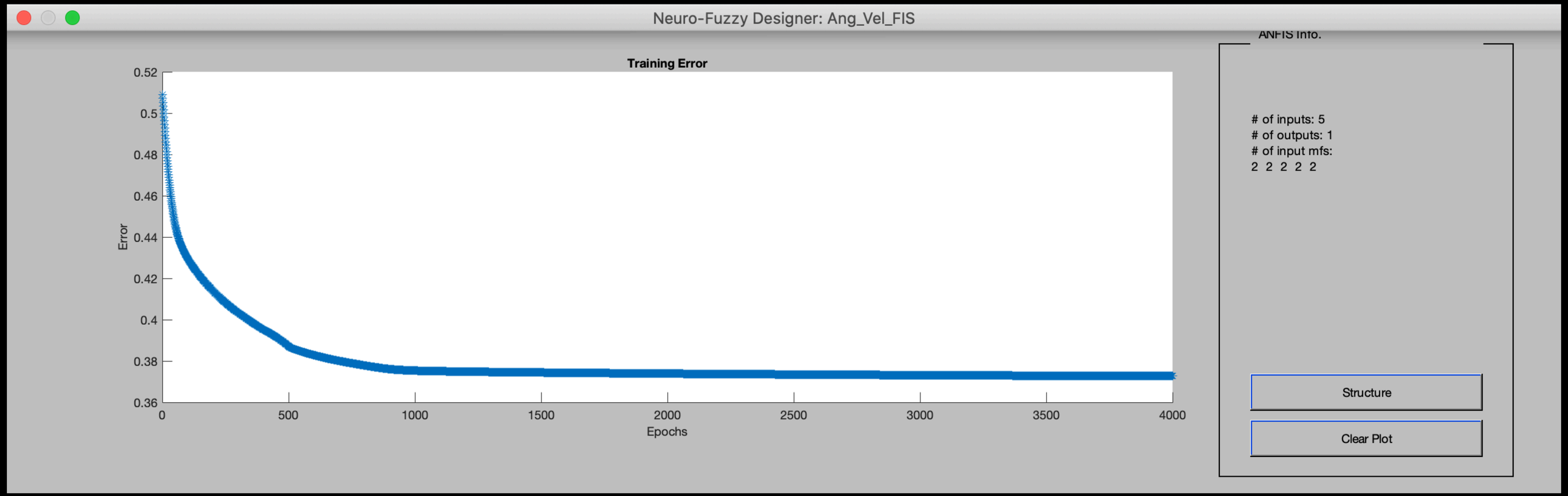
Blue points are our ground truth classical controller output.



ANFIS

Angular Velocity Train

Trained Angular_Velocity.fis with 4000 Epoch with same 30 000 sample points with 5 input 1 output; each has 2 different Gaussian Membership Function at Fuzzy Inference Base

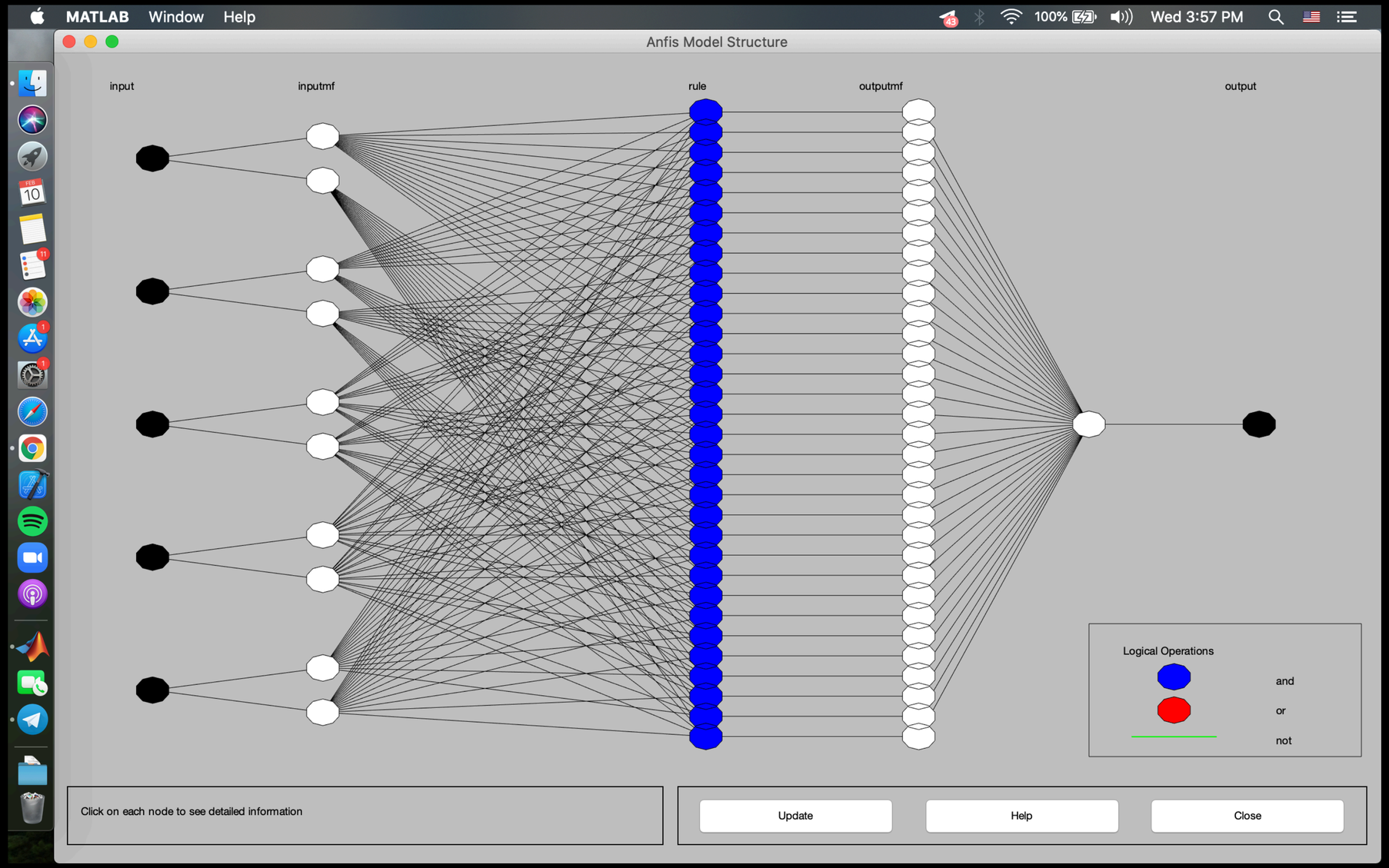


ANFIS

Structure of FIS

- Our 5 Input Variable with 2 MF function. Created 32 different node for Fuzzy Rule

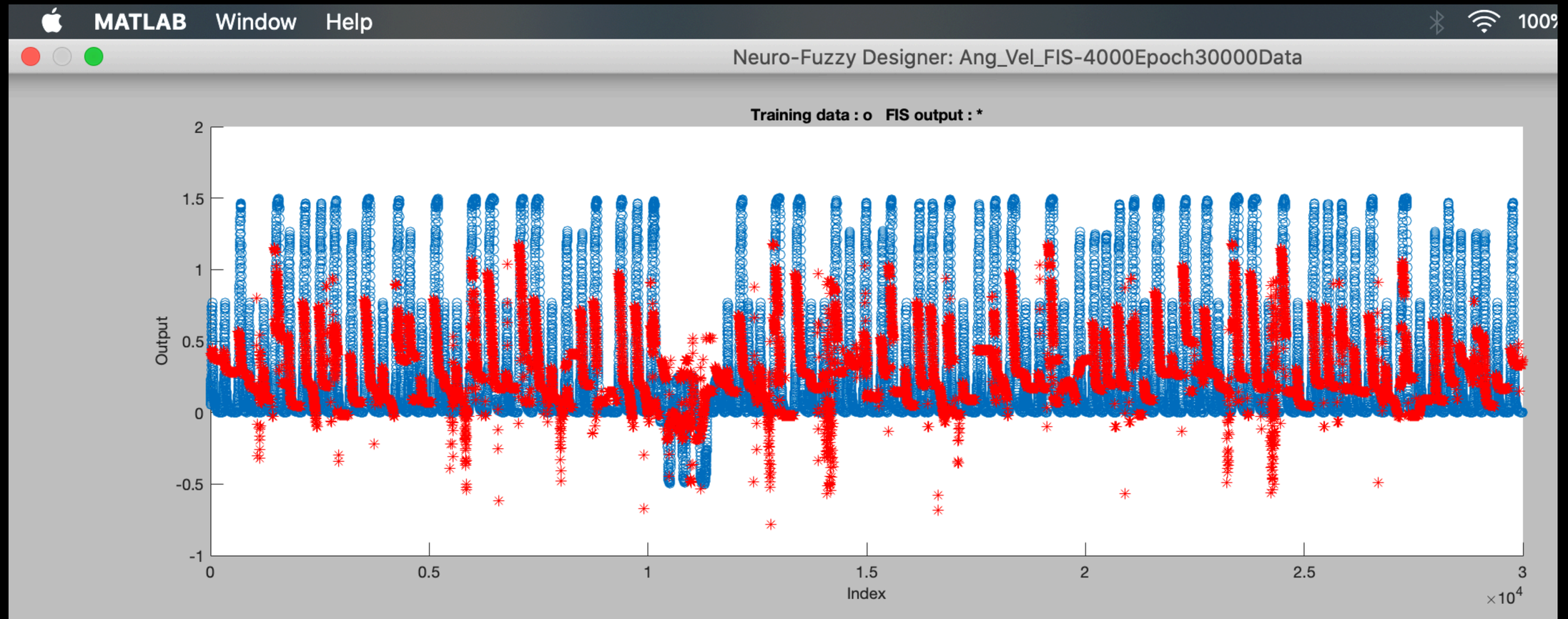
- Aggregator collected from 32 different Fuzzy Rule Output



ANFIS

Result 2

- Result of Angular Velocity Fuzzy Inference System vs Ground Truth



Comments

- Dynamic system of our moving vehicle (Robot) can be well expressed by linear systems, the classic, traditional PID based controller has more advantages in terms of giving more continuous and accelerate friendly result for driving of the system.
- For my project experiment result, if we can interpret the reason of outlier points at our current FIS system, then we can obtain a FIS model which is more accurate, non disturbed with outliers and in the more continuous form.
- This experiments result of Angular Velocity FIS model at shown at **previous slide,** may cause so many distortion at straight line moving. Robot may move to left and right in consecutive time steps with unstable condition which may increase the risk of accident by unstable inertia.

Github Repo: All work (Presentation Video+Codes) is uploaded to

https://github.com/ahmedbing/FuzzyLogicBLU513E

Thank you for semester