ASSIGNMENT COVERSHEET

UNIVERSITY OF TECHNOLOGY SYDNEY

UTS: ENGINEERING & INFORMATION TECHNOLOGY

	NAME OF STUDENT(s) (PRINT CLEARLY)	STUDENT ID(s).
	Joel Cappelli	12137384

STUDENT EMAIL	STUDENT CONTACT NUMBER
joel.cappell@gmail.com	0422 384 087

NAME OF TUTOR	TUTORIAL GROUP	DUE DATE
Hung Nguyen		19 May 2016

ASSESSMENT ITEM NUMBER/ TITLE
49275 NEURAL NETWORKS AND FUZZY LOGIC Assignment 2

☒ I confirm that I have read, understood and followed the guidelines for assignment submission and presentation on page 2 of this cover sheet.

☒ I confirm that I have read, understood and followed the advice in my Subject Outline about assessment requirements.

☒ I understand that if this assignment is submitted after the due date it may incur a penalty for lateness unless I have previously had an extension of time approved and have attached the written confirmation of this extension.

Declaration of Originality: The work contained in this assignment, other than that specifically attributed to another source, is that of the author(s) and has not been previously submitted for assessment. I understand that, should this declaration be found to be false, disciplinary action could be taken and penalties imposed in accordance with University policy and rules. In the statement below, I have indicated the extent to which I have collaborated with others, whom I have named.

Statement of Collaboration:

Joel Cappelli

Signature of Student(s) ______________________ Date 19 May 2016

ASSIGNMENT RECEIPT

To be completed by the student if a receipt is required

SUBJECT NAME/NUMBER	NAME OF TUTOR
49275 NEURAL NETWORKS AND FUZZY LOGIC	

SIGNATURE OF TUTOR	RECEIVED DATE
	19 May 2016

49275 NEURAL NETWORKS AND FUZZY LOGIC

Assignment 2

Joel Cappelli

12137384 19 May 2016

Qu 1.1

Generalised XOR Problem

A neural network model with three input neurons (one augmented input), four hidden neurons (one augmented) in a single hidden layer, and an output neuron is used to learn the decision surface of the well-known generalised XOR problem. Weights were updated using error backpropagation training with a constant learning rate of 0.2 and augmented inputs of -1. All continuous perceptrons use the bipolar logistic function.

Initial weights (set at random):

$$W(1) = \begin{bmatrix} 0.3443 & 0.6762 & -0.9607 & 0.3626 \end{bmatrix}$$

$$\bar{W}(1) = \begin{bmatrix} -0.2410 & 0.4189 & -0.6207 \\ 0.6636 & -0.1422 & -0.6131 \\ 0.0056 & -0.3908 & 0.3644 \end{bmatrix}$$

Weight update after 1 step (first training pattern):

$$W(2) = \begin{bmatrix} 0.3698 & 0.7154 & -0.9830 & 0.2817 \end{bmatrix}$$

$$\bar{W}(2) = \begin{bmatrix} -0.2312 & 0.4255 & -0.6333 \\ 0.6800 & -0.1312 & -0.6340 \\ -0.0225 & -0.4096 & 0.4003 \end{bmatrix}$$

Qu 1.2

The following formulae were used to compute cycle and pattern error.

$$Pattern\ error = E_p = \frac{1}{2}(d_p - z_p)^{\boldsymbol{T}}(d_p - z_p) \quad Cycle\ error = E_c = \frac{1}{2}\sum_{p=1}^{8}(d_p - z_p)^{\boldsymbol{T}}(d_p - z_p)$$

There are 8 training patterns in the question. For every 1 cycle, there are 8 updates to the weights matrices.

Figure 2 and Figure 3 show the cycle and pattern error curves over 500 cycles (4000 steps) of training. The training set is recycled.

Weights after 4000 steps (500 cycles):

$$W_f = W(4001) = \begin{bmatrix} 4.7982 & 6.3878 & -4.8148 & 1.6556 \end{bmatrix}$$

$$\bar{W}_f = \bar{W}(4001) = \begin{bmatrix} -0.5195 & 4.9442 & 0.9252 \\ 5.6967 & -5.2245 & -4.3788 \\ 4.1371 & 4.5040 & -2.2576 \end{bmatrix}$$

Considering two unseen test vectors, x_{t1} and x_{t2} below, we can examine the classification ability of the trained network.

$$x_{t1} = \begin{bmatrix} 0.6263 \\ -0.9803 \end{bmatrix} \qquad x_{t2} = \begin{bmatrix} 0.0700 \\ 0.0500 \end{bmatrix}$$

The desired output for each test vector are as below:

$$d_{t1} = sgn(0.6263 * -0.9803) = -1 \qquad d_{t2} = sgn(0.0700 * 0.0500) = 1$$

Classification with $W(4001)$ and $\overline{W}(4001)$:

$$d^{4001}{}_{t1} = \ -0.4898 = \textbf{CORRECT}$$

$$d^{4001}{}_{t2} = \ -0.5712 = \textbf{INCORRECT}$$

The results show that the trained network with final weight matrices above, correctly classify x_{t1} and fail to correctly classify x_{t2}. Figure 1 shows the training and test set on the Cartesian plane as well as the output regions and boundaries. The proximity of x_{t2} to the origin makes this a difficult test case because it is close to 4 output decision boundaries. While x_{t1} is clearly in Region -1 and therefore easier to classify. In order for this network to perform better, we may need to increase the number of hidden neuron nodes and use a cross-validation training method on a larger data set which is uniformly spaced in the x_1 and x_2 input space.

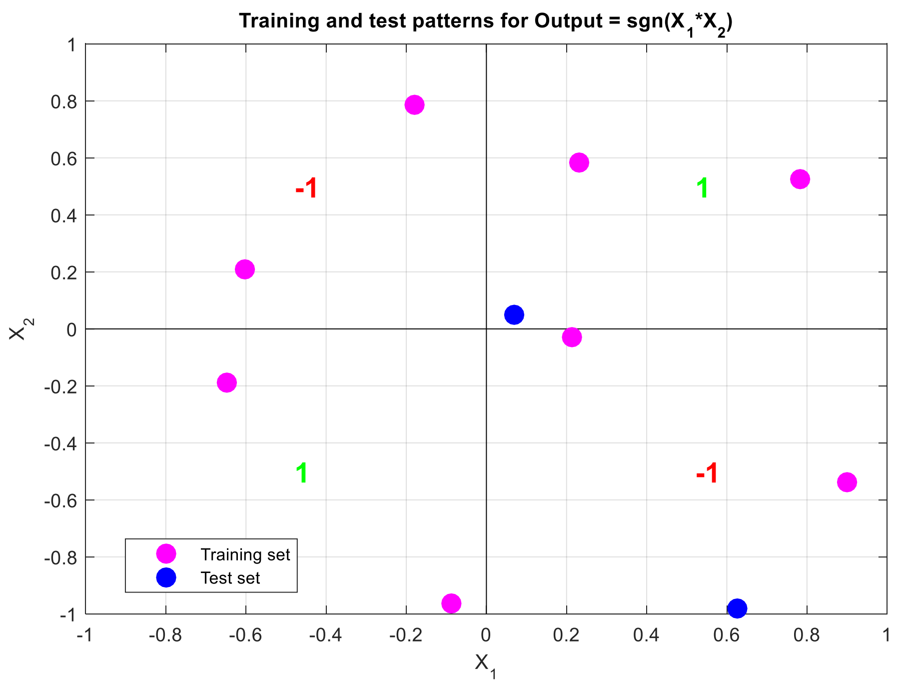

Figure 1: Qu1.2 Training and test set for the XOR Network learning

Figure 2 and Figure 3 how the cycle and pattern error curves over 500 cycles (4000 steps) of training. The cycle error is reducing over each cycle as the weights vector is converging to a minimum in the weights space. The use of augmented input on the input and hidden layer helps with classification because it shifts the centre of each neuron and thus improves the ability of network to generalise (universal function approximation theorem).

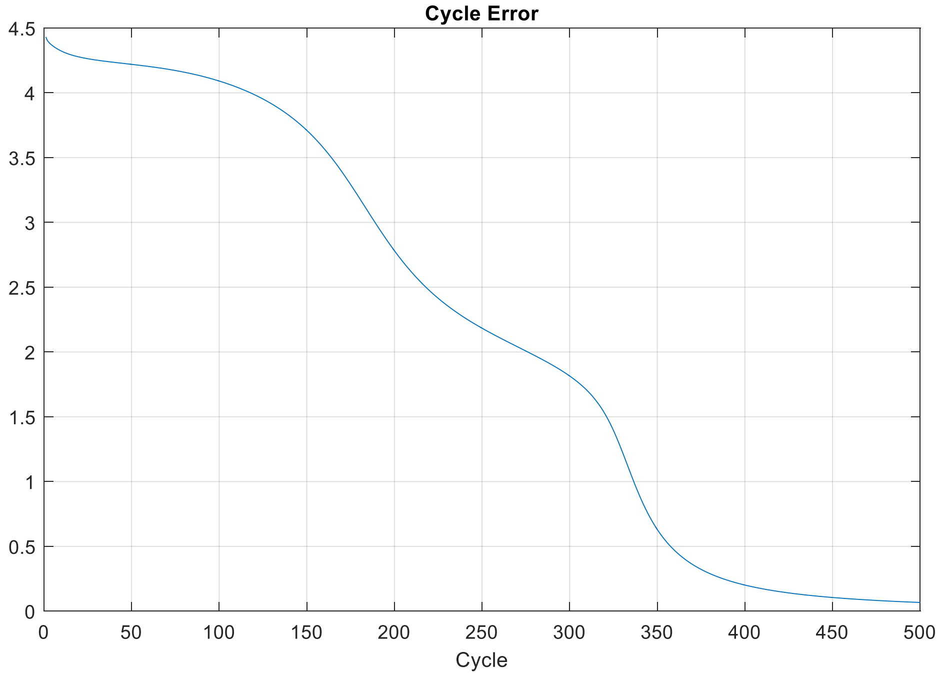

Figure 2: Qu1.2 Cycle error curve for 500 cycles

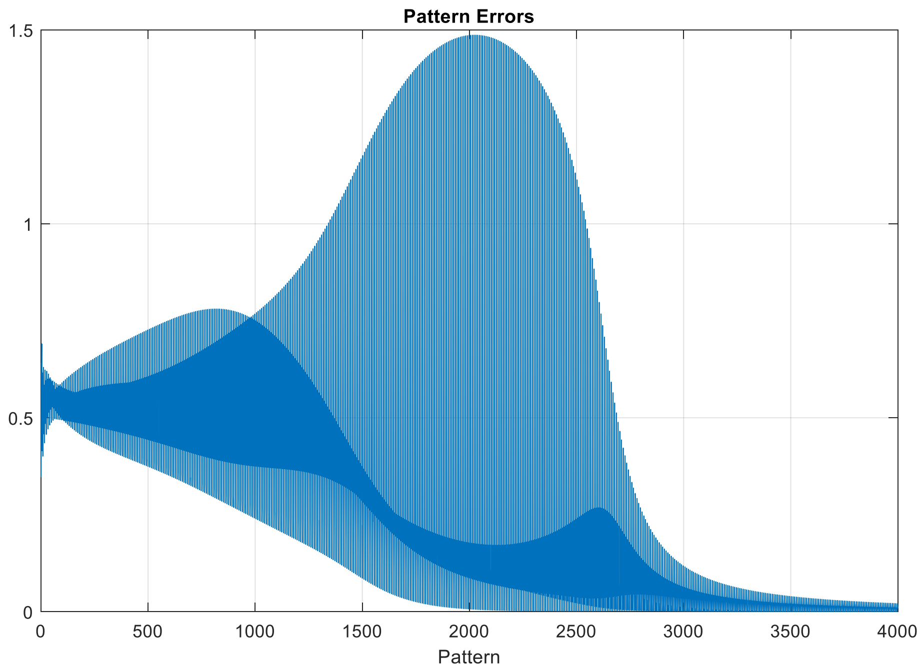

Figure 3: Qu1.2 Pattern error curve for 500 cycles

Qu 2 Truck-Backer Upper Control

To represent the control angle input to the controller, a set of linguistic variables is chosen to represent 5 degrees of position, 5 degrees of truck angle, and 5 degrees of control angle. Membership functions are constructed to represent the input and output values' grades of membership as shown in Figure 4. The rule set in the form of "Fuzzy Associative Memories" below.

Table 1: FAM (Fuzzy associative memories)

Position (x)

Truck angle (ϕ)

	NM	**NS**	**ZE**	**PS**	**PM**
NM	ZE	NS	NM	NM	NM
NS	PS	ZE	NS	NM	NM
ZE	PM	PS	ZE	NS	NM
PS	PM	PM	PS	ZE	NS
PM	PM	PM	PM	PS	ZE

The desired state of the truck is $(x_f, \phi_f) = (10m, 90°)$. The initial state of the truck is $[\phi(1), x(1), y(1)] = [35°, 15m, 15m]$. This is shown in the initial fuzzy mapping below. The membership functions are centred on the desired state.

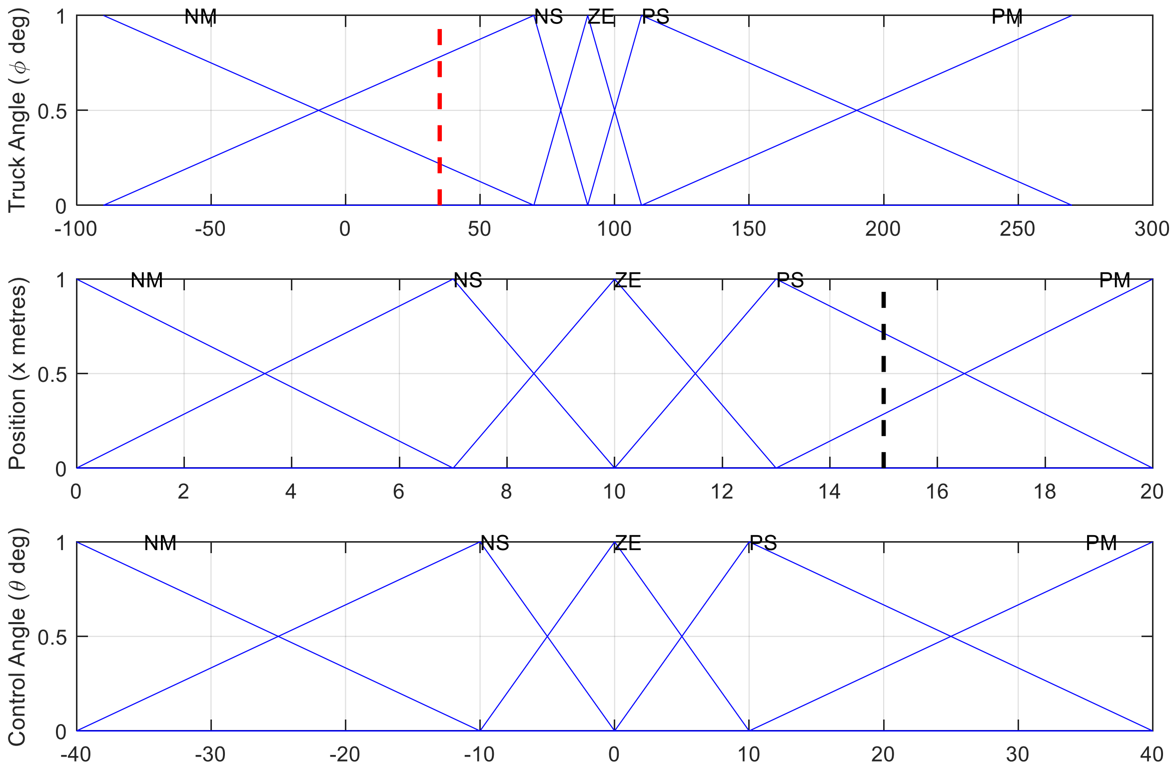

Figure 4: Qu2 Membership functions

In fuzzy notation;

$$Position\_x_{15} = \frac{0.7143}{PS} + \frac{0.2857}{PM}$$

$$TruckAngle_{35} = \frac{0.2188}{NM} + \frac{0.7813}{NS}$$

Using the FAM above given in question and inference for each associated method;

Table 2: State 1 UCOA Rule Decomposition

STATE 1	$TruckAngle$	$Position\_x$	U(COA [Max-min])
Rule 1	$\frac{0.2188}{NM}$	$\frac{0.7143}{PS}$	$\frac{0.2188}{NM}$
Rule 2	$\frac{0.2188}{NM}$	$\frac{0.2857}{PM}$	$\frac{0.2188}{NM}$
Rule 3	$\frac{0.7813}{NS}$	$\frac{0.7143}{PS}$	$\frac{0.7143}{NM}$
Rule 4	$\frac{0.7813}{NS}$	$\frac{0.2857}{PM}$	$\frac{0.2857}{NM}$

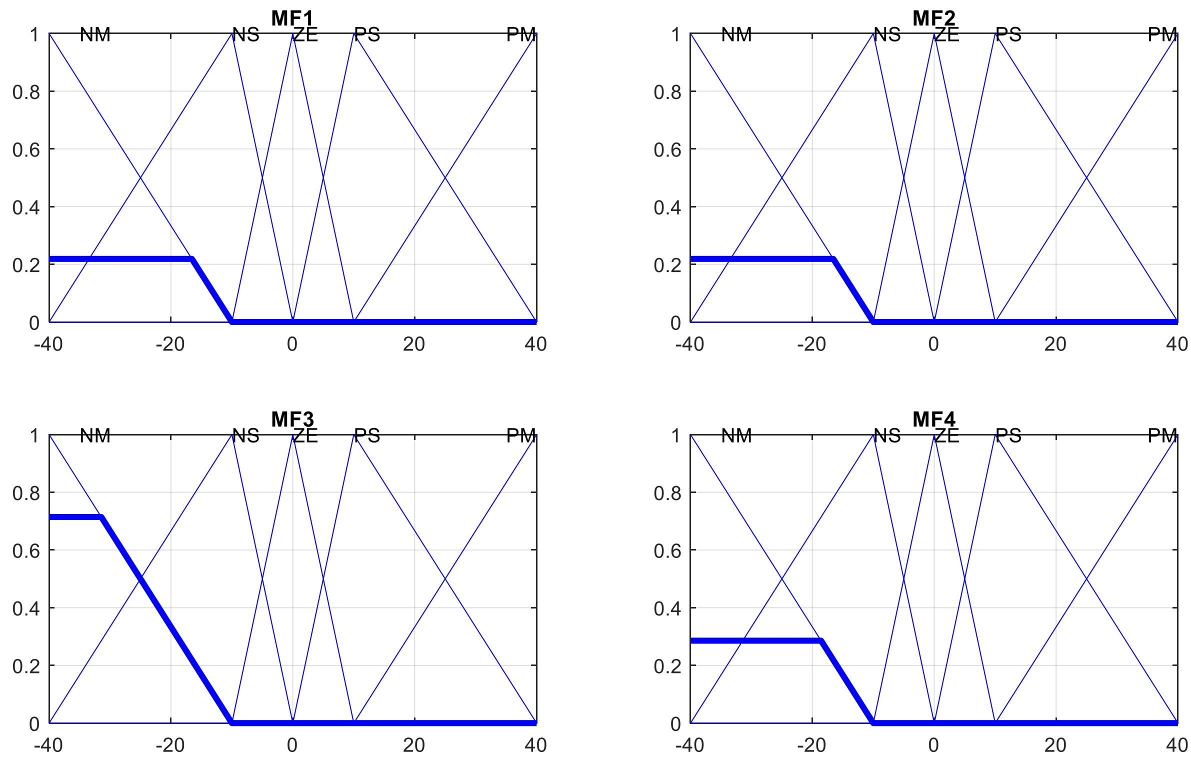

Figure 5: Qu2.1 UCOA Membership Rules for initial position and truck angle

According to the COA (Centre of area) method, the output u^{*} can be found using Max-min inference as:

Defuzzification Piecewise Functions

Function 1 between -40 and -31.4286 is u(U) = 0.714286

Area 1 = 6.12245

Moment 1 = -218.659

Function 2 between -31.4286 and -10 is u(U) = -0.0333333*u + -0.333333

Area 2 = 7.65306

Moment 2 = -185.86

COA Method (max-min) - Defuzzified Output for initialXPosError = 15 m, initialTruckAngleError = 35 deg

Total Area = 13.7755

Total Moment = -404.519

$\theta(1) = -29.3651\, deg$

$[\phi(2), x(2), y(2)] = [49.1928°, 15.7139\text{m}, 15.4999\text{m}]$

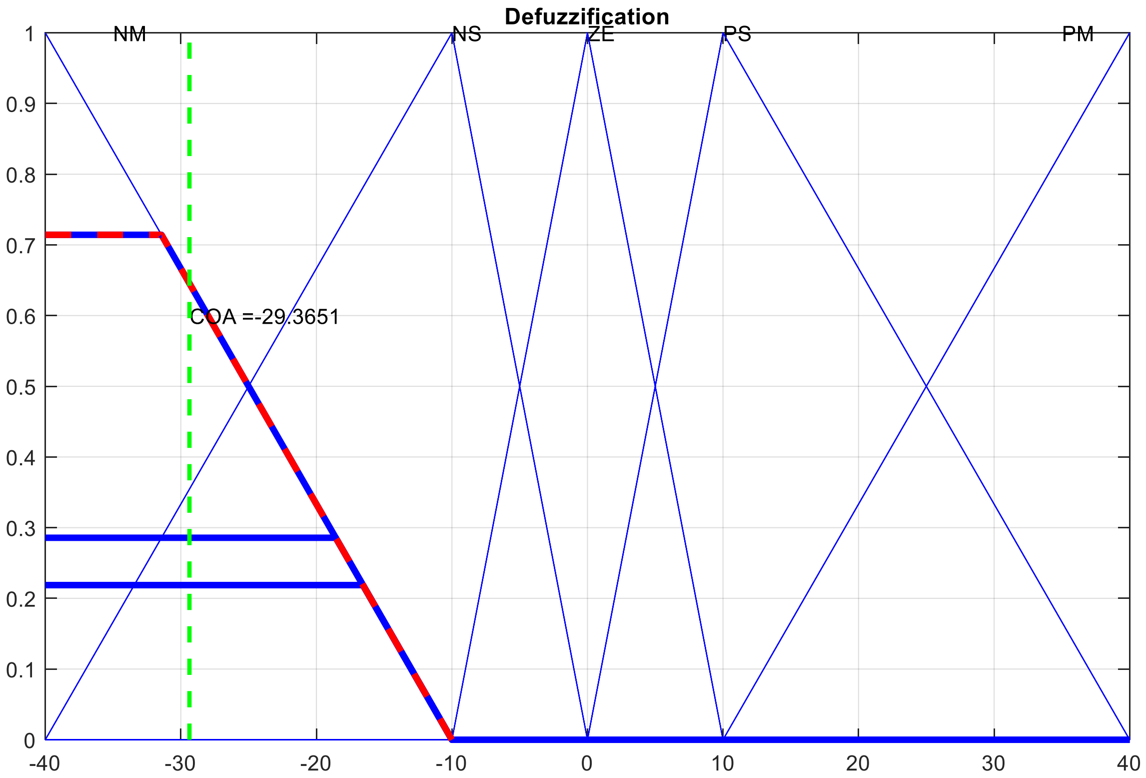

Figure 6: Qu2.1 Defuzzification of controller output using the COA Strategy

Qu 2.2 UMOM Control Strategy

Using the same initial conditions as above, we can compute the MOM (mean of maximum) output truck control angle for state 1 and following states.

Table 3: State 1 UMOM Rule Decomposition

STATE 1	$TruckAngle$	$Position\_x$	U(MOM [prod])
Rule 1	$\frac{0.2188}{NM}$	$\frac{0.7143}{PS}$	$\frac{0.15625}{NM}$

Rule 2	$\frac{0.2188}{NM}$	$\frac{0.2857}{PM}$	$\frac{0.0625}{NM}$
Rule 3	$\frac{0.7813}{NS}$	$\frac{0.7143}{PS}$	$\frac{0.558036}{NM}$
Rule 4	$\frac{0.7813}{NS}$	$\frac{0.2857}{PM}$	$\frac{0.223214}{NM}$

MOM Method (prod) - Defuzzified Output for xPosError = 15 m, truckAngleError = 35 deg

$\theta(1) = -40 \text{ deg}$

$[\phi(2), x(2), y(2)] = [53.7472°, 15.6275\text{m}, 15.4394\text{m}]$

Then performing the same calculation for state 2:

Table 4: State 2 UMOM Rule Decomposition

STATE 2	U(MOM [prod])
Rule 1	$\frac{0.063451}{NM}$
Rule 2	$\frac{0.0381288}{NM}$
Rule 3	$\frac{0.561191}{NM}$
Rule 4	$\frac{0.337229}{NM}$

MOM Method (prod) - Defuzzified Output for xPosError = 15.6275 m, truckAngleError = 53.7472 deg

$\theta(2) = -40 \text{ deg}$

$[\phi(3), x(3), y(3)] = [72.4945°, 16.0805\text{m}, 16.0571\text{m}]$

This is then performed for 100 sampling points ahead, computing the control angle output and new state vector. Figure 7 shows the position of the truck while backing up in Cartesian plane. The truck reaches its desired x-position of 10m within 45m of its initial y-position.

Figure 8 and Figure 9 show the controller response over sampling time eventually reaching the desired set points. The controller output angle shows the initial truck control angle saturating at -40 deg then stabilising at 0 deg. The error output plots show more overshoot (both acceptable at < 10% for step change between sample 1 and 13) in x-position than truck angle and settling within 40 sample points.

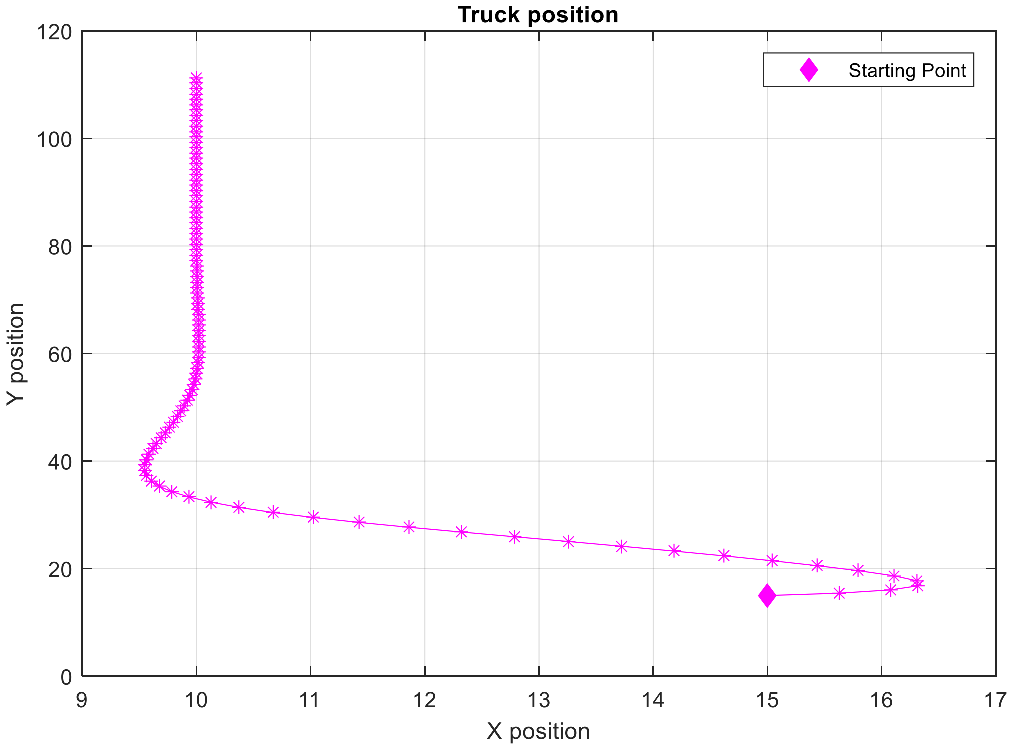

Figure 7: Qu2.2 Plot of corresponding vertical truck position y(k) against the horizontal truck position x(k) for 100 sampling points

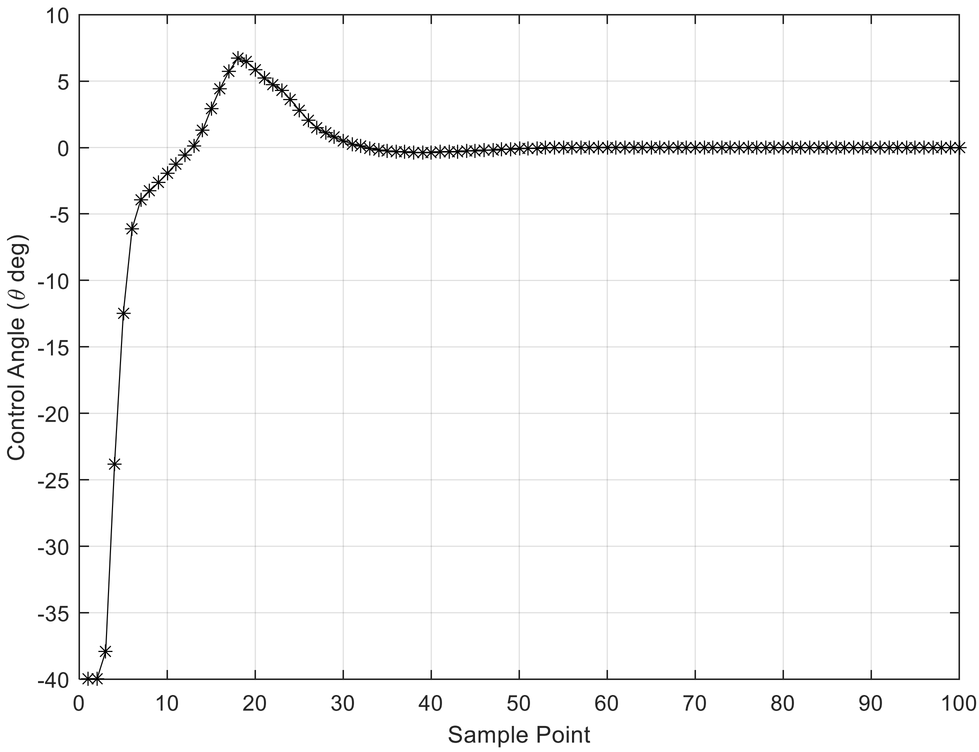

Figure 8: Qu2.2 Plot of the output fuzzy logic control angle θ for 100 sampling points

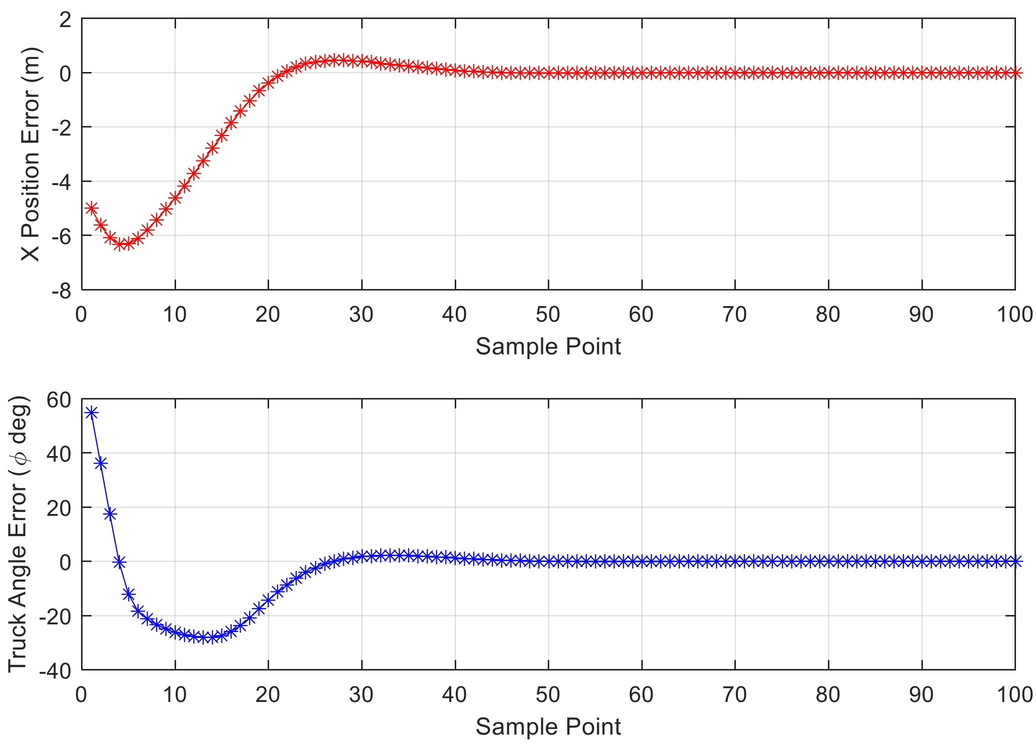

Figure 9: Qu2.2 Plot of the error between controlled variables (x and ϕ) for 100 sampling points

Qu 2.2 UMOM Control Strategy – Dominant Rule

Self-organising procedures can be used to modify the generic fuzzy associative memories linking each input fuzzy set to the output controller fuzzy set in order to improve performance. The following procedure is used:

- Calculate the fire strength α_i for each control rule
- Perform fuzzy reasoning u_i for each control rule
- Find the dominant rule which contributes most to the control action
- Modify the dominant rule to a new control output for desired performance

This process was conducted on the truck backer upper problem using UMOM strategy and modifying the first dominant rule after the first state evaluation to a softer output rule (one rule weaker – NM modified to NS). This was to enforce a slower controller response based on the trucks initial position. Examining Table 3: State 1 UMOM Rule Decomposition, the dominant rule is Rule 3, $\frac{0.558036}{NM}$. Therefore, the corresponding controller output rule was reduced to NS. The resulting modified FAM table with the modified dominant rule highlighted is below:

Table 5: Modified FAM (Fuzzy associative memories)

Position (x)

Truck angle (ϕ)

	NM	**NS**	**ZE**	**PS**	**PM**
NM	ZE	NS	NM	NM	NM
NS	PS	ZE	NS	**NS**	NM
ZE	PM	PS	ZE	NS	NM
PS	PM	PM	PS	ZE	NS
PM	PM	PM	PM	PS	ZE

This modified FAM table is used throughout the rest of the control process. Using the same initial conditions as above, we can compute the MOM (mean of maximum) output truck control angle for state 1 and following states.

Table 6: State 1 UMOM Rule Decomposition – Modified Rule

STATE 1	U(MOM [prod])
Rule 1	$\frac{0.15625}{NM}$
Rule 2	$\frac{0.0625}{NM}$
Rule 3	$\frac{0.558036}{NS}$
Rule 4	$\frac{0.223214}{NM}$

MOM Method (prod) - Defuzzified Output for xPosError = 15 m, truckAngleError = 35 deg

$\theta^*(1) = \; -23.2589\text{deg}$

$[\phi(2), x(2), y(2)] = [46.3875°, 15.7526\text{m}, 15.527\text{m}]$

This is then performed for 100 sampling points ahead, computing the control angle output and new state vector using the modified FAM. Figure 10 shows the softer control action due to the modified rule. This causes the truck to back up with larger overshoot of the y-position set point.

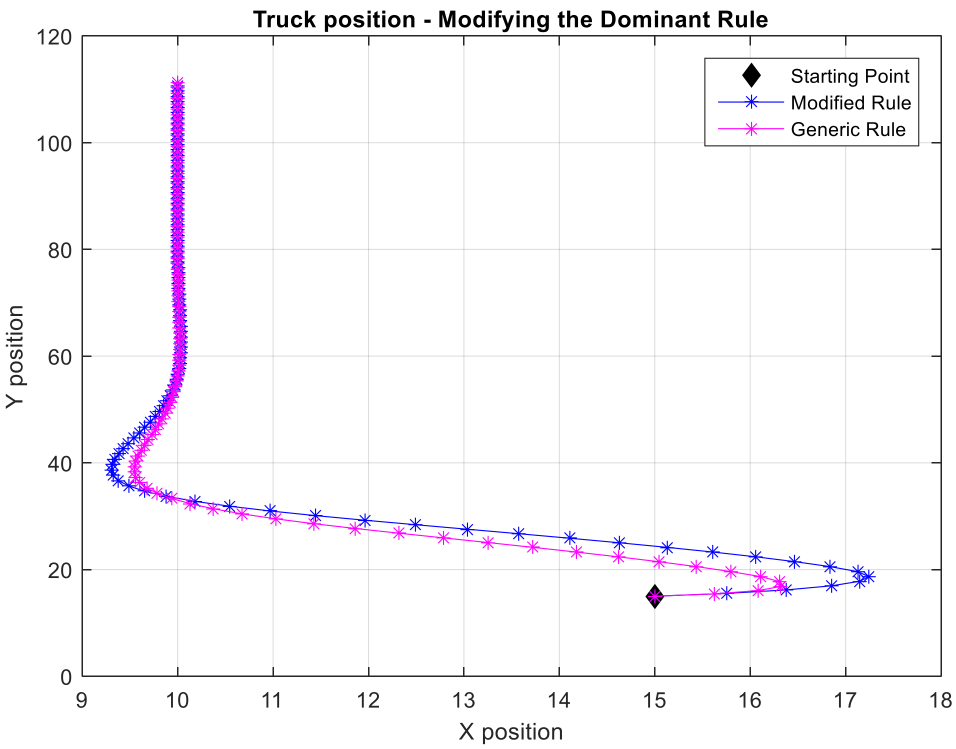

Figure 10: Qu2.2 Plot of vertical truck position y(k) against the horizontal truck position x(k) for 100 sampling points showing the effect of the modified rule

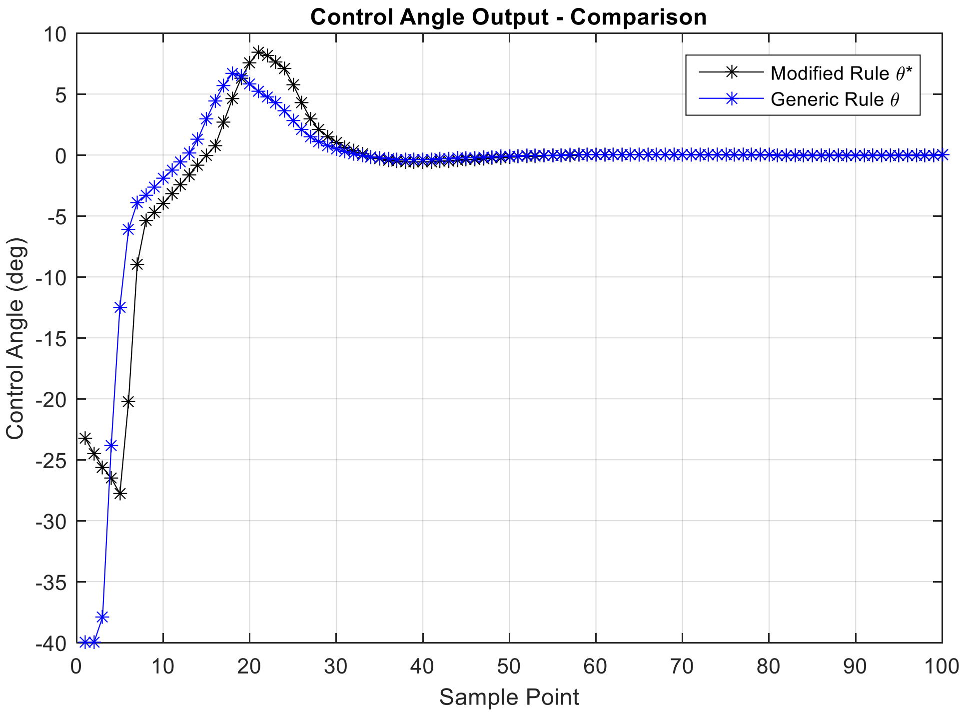

Figure 11: Qu2.2 Plot of the output fuzzy logic control angle θ and θ for 100 sampling points showing the effect of the modified rule*

Figure 11 and Figure 12 show the controller response over sampling time, comparing the net effect of the modified rule. The controller output angle shows the initial truck control angle beginning at ~-24 deg then stabilising while the generic rule saturates at -40 deg. This weakening of the dominant rule has a long term effect on the fuzzy controller as the output lags behind causing greater overshoot in each of the state variables when compared with the generic rule simulation results.

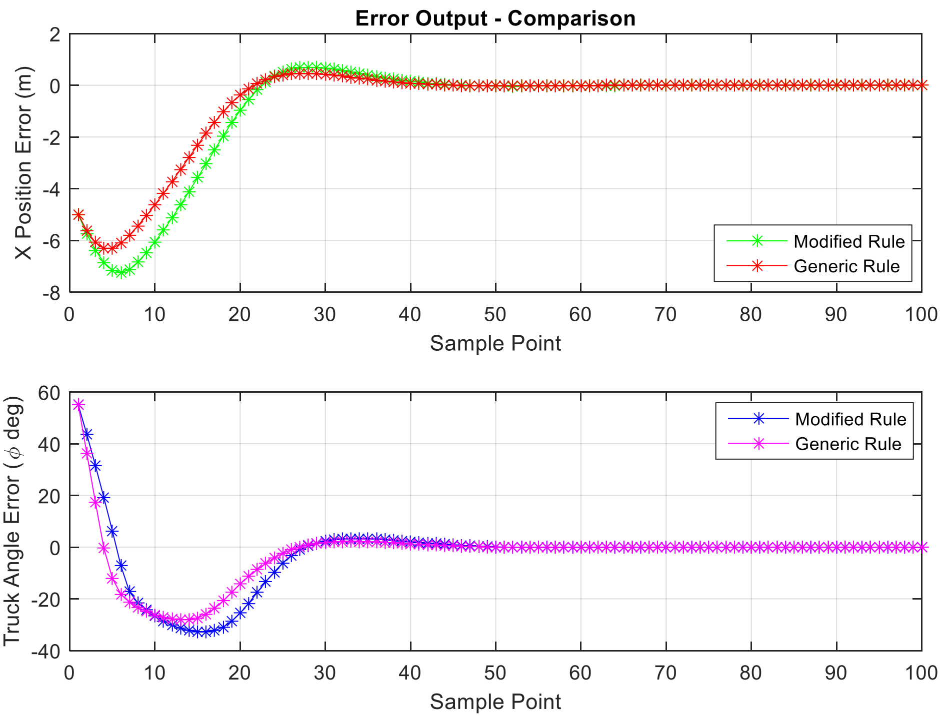

Figure 12: Qu2.2 Plot of the error between controlled variables (x and ϕ) for 100 sampling points showing the effect of the modified rule

Source code

```
clear all;
close all;
clc;

%% Neural Networks Assignment 2
% Joel Cappelli
% 12137384

%% Qu1
x1 = [0.7826 0.5242];
x2 = [0.9003 -0.5377];
x3 = [-0.0871 -0.9630];
x4 = [-0.1795 0.7873];
x5 = [0.2309 0.5839];
x6 = [0.2137 -0.0280];
x7 = [-0.6475 -0.1886];
x8 = [-0.6026 0.2076];

setX = [x1;x2;x3;x4;x5;x6;x7;x8];

X = setX;
augInput = -1;
X = [setX augInput*ones(size(setX,1),1)];

d1 = 1;
d2 = -1;
d3 = 1;
d4 = -1;
d5 = 1;
d6 = -1;
d7 = 1;
d8 = -1;

D = [d1;d2;d3;d4;d5;d6;d7;d8];

n = 0.2;
m = 0;
smse = 1e-6;
plotFig = 0;

L = [3 4 1];
hiddenBias = -1;

nLayers = length(L); % we'll use the number of layers often
initWeights = cell(nLayers-1,1); % a weight matrix between each layer

initWeights{2} = [ 0.3443 0.6762 -0.9607 0.3626 ];

initWeights{1} = [ -0.2410 0.4189 -0.6207;...
                   0.6636 -0.1422 -0.6131;...
                   0.0056 -0.3908 0.3644
                 ];

% activaFn = @bipolarLog;
% activaDerivFn = @bipolarLogDeriv;
fnHandles = {@bipolarLog, @bipolarLogDeriv};

maxNumEpochs = 1;
Network =
backpropNN(L,n,m,smse,X(1,:),D(1,:),fnHandles,maxNumEpochs,plotFig,hiddenBias,initW
eights);

% disp('Initial weights:');
% disp(wtsArray(:,1));
```

```
fprintf('Qu1.1\n');

fprintf('Weights after 1 step (first training pattern):\n');
fprintf('Wbar(2) = \n');
disp(Network.weights{1});
fprintf('W(2) = \n');
disp(Network.weights{2});

maxNumEpochs = 500;
Network =
backpropNN(L,n,m,smse,X,D,fnHandles,maxNumEpochs,plotFig,hiddenBias,initWeights);

fprintf('Qu1.2\n');

fprintf('Weights after 4000 steps (500 cycles):\n');
fprintf('Wbar(4001) = \n');
disp(Network.weights{1});
fprintf('W(4001) = \n');
disp(Network.weights{2});

figure;
plot(Network.cycleErrors);
title('Cycle Error');
grid on;
xlabel('Cycle');

figure;
plot(Network.patternErrors);
title('Pattern Errors');
grid on;
xlabel('Pattern');

%write up feedforward function
xtest1 = [0.6263 -0.9803];
xtest2 = [0.0700 0.0500];

settestX = [xtest1;xtest2];

testX = settestX;
augInput = -1;
testX = [settestX augInput*ones(size(settestX,1),1)];

testD = [sign(prod(xtest1));sign(prod(xtest2))];
output = feedfwdMLP(testX,L,Network.weights,fnHandles,hiddenBias);

fprintf('Classification after 4000 steps (500 cycles):\n');

fprintf('xTV1 = \n');
disp(xtest1');
fprintf('xTV2 = \n');
disp(xtest2');

fprintf('outputTrueTV1 = \n');
disp(testD(1)');
fprintf('classifyTV1 = \n');
disp(output(1)');

fprintf('outputTrueTV2 = \n');
disp(testD(2)');
fprintf('classifyTV2 = \n');
disp(output(2)');

% maxNumEpochs = 2000;
% Network =
backpropNN(L,n,m,smse,X,D,fnHandles,maxNumEpochs,plotFig,hiddenBias,initWeights);
%
% fprintf('Weights after 16000 steps (2000 cycles):\n');
% fprintf('Wbar(16001) = \n');
```

```
% disp(Network.weights{1});
% fprintf('W(16001) = \n');
% disp(Network.weights{2});
%
% figure;
% plot(Network.cycleErrors);
% title('Cycle Error');
% grid on;
% xlabel('Cycle');
%
% figure;
% plot(Network.patternErrors);
% title('Pattern Errors');
% grid on;
% xlabel('Pattern');
%
% output = feedfwdMLP(testX,L,Network.weights,fnHandles,hiddenBias);
%
% fprintf('Classification after 16000 steps (2000 cycles):\n');
%
% fprintf('xTV1 = \n');
% disp(xtest1');
% fprintf('xTV2 = \n');
% disp(xtest2');
%
% fprintf('outputTrueTV1 = \n');
% disp(testD(1)');
% fprintf('classifyTV1 = \n');
% disp(output(1)');
%
% fprintf('outputTrueTV2 = \n');
% disp(testD(2)');
% fprintf('classifyTV2 = \n');
% disp(output(2)');

%% Qu 2
fprintf('\nQu2 Fuzzy logic Controller\n\n');
X = 1;
Y = 2;
PHI = 3;
xDesired = 10;
phiDesired = 90;

initialState = zeros(1,3);
initialState(X) = 15;
initialState(Y) = 15;
initialState(PHI) = 35;

minXPosErrorRange = 0;
maxXPosErrorRange = 20;
maxXPosError = 1;

minTruckAngleErrorRange = -90;
maxTruckAngleErrorRange = 270;
maxTruckAngleError = 1;

minControlAngleRange = -40;
maxControlAngleRange = 40;
maxOutput = 1;

initialXPosError =
initialState(X);%checkInputRange(minXPosErrorRange,maxXPosErrorRange,xDesired -
initialState(X));
initialTruckAngleError =
initialState(PHI);%checkInputRange(minTruckAngleErrorRange,maxTruckAngleErrorRange,
phiDesired - initialState(PHI));

setXPosError = {'NM','NS','ZE','PS','PM'};
```

```
truckAngleErrorTextXpos = [-60 70 90 110 240];%(minErrorRange - [-4 -0.8 0 0.8
4])./(minErrorRange-maxErrorRange);
setTruckAngleError = {'NM','NS','ZE','PS','PM'};
errorTextXpos = [1 7 10 13 19];%(minDeltaErrorRange - [-0.6 0
0.6])./(minDeltaErrorRange-maxDeltaErrorRange);
degXPosError = size(setXPosError,2);
degTruckAngleError = size(setTruckAngleError,2);

if(degXPosError > degTruckAngleError)
    setOutput = setXPosError;
else
    setOutput = setTruckAngleError;
end
contOutputTextXpos = [-35 -10 0 10 35];%(minOutputRange -[-8 -4 0 4
8])./(minOutputRange-maxOutputRange);
degOutput = size(setOutput,2);

FAM_TruckAngleError_XPosError = {'ZE','NS','NM','NM','NM';...
                                'PS','ZE','NS','NM','NM';...
                                'PM','PS','ZE','NS','NM';...
                                'PM','PM','PS','ZE','NS';...
                                'PM','PM','PM','PS','ZE'};

cellfind = @(string)(@(cell_contents)(strcmp(string,cell_contents)));
logical_col = cellfun(cellfind('NS'),setXPosError);
logical_row = cellfun(cellfind('PS'),setTruckAngleError);
findOutput =
cellfun(cellfind(FAM_TruckAngleError_XPosError(logical_row,logical_col)),setOutput)
;
test = setOutput(findOutput);

MAX = 1;
LINEAR = 2;
MIN = 3;

TruckAngleErrorFuzzySet1_Func = 'NM';
TruckAngleErrorFuzzySet1_output = [1 0 0];
TruckAngleErrorFuzzySet1_inputBreakPts = [minTruckAngleErrorRange 70
maxTruckAngleErrorRange];

TruckAngleErrorFuzzySet2_Func = 'NS';
TruckAngleErrorFuzzySet2_output = [0 1 0 0];
TruckAngleErrorFuzzySet2_inputBreakPts = [minTruckAngleErrorRange 70 90
maxTruckAngleErrorRange];

TruckAngleErrorFuzzySet3_Func = 'ZE';
TruckAngleErrorFuzzySet3_output = [0 0 1 0 0];
TruckAngleErrorFuzzySet3_inputBreakPts = [minTruckAngleErrorRange 70 90 110
maxTruckAngleErrorRange];

TruckAngleErrorFuzzySet4_Func = 'PS';
TruckAngleErrorFuzzySet4_output = [0 0 1 0];
TruckAngleErrorFuzzySet4_inputBreakPts = [minTruckAngleErrorRange 90 110
maxTruckAngleErrorRange];

TruckAngleErrorFuzzySet5_Func = 'PM';
TruckAngleErrorFuzzySet5_output = [0 0 1];
TruckAngleErrorFuzzySet5_inputBreakPts = [minTruckAngleErrorRange 110
maxTruckAngleErrorRange];

TruckAngleErrorFuzzySet_Funcs =
{TruckAngleErrorFuzzySet1_Func;TruckAngleErrorFuzzySet2_Func;TruckAngleErrorFuzzySe
t3_Func;TruckAngleErrorFuzzySet4_Func;TruckAngleErrorFuzzySet5_Func};
TruckAngleErrorFuzzySet_output =
{TruckAngleErrorFuzzySet1_output;TruckAngleErrorFuzzySet2_output;TruckAngleErrorFuz
zySet3_output;TruckAngleErrorFuzzySet4_output;TruckAngleErrorFuzzySet5_output};
TruckAngleErrorFuzzySet_inputBreakPts =
{TruckAngleErrorFuzzySet1_inputBreakPts;TruckAngleErrorFuzzySet2_inputBreakPts;Truc
```

```
kAngleErrorFuzzySet3_inputBreakPts;TruckAngleErrorFuzzySet4_inputBreakPts;TruckAngl
eErrorFuzzySet5_inputBreakPts};

numTruckAngleErrorFuzzySets = size(TruckAngleErrorFuzzySet_Funcs,1);

xPosErrorFuzzySet1_Func = 'NM';
xPosErrorFuzzySet1_output = [1 0 0];
xPosErrorFuzzySet1_inputBreakPts = [minXPosErrorRange 7 maxXPosErrorRange];

xPosErrorFuzzySet2_Func = 'NS';
xPosErrorFuzzySet2_output = [0 1 0 0];
xPosErrorFuzzySet2_inputBreakPts = [minXPosErrorRange 7 10 maxXPosErrorRange];

xPosErrorFuzzySet3_Func = 'ZE';
xPosErrorFuzzySet3_output = [0 0 1 0 0];
xPosErrorFuzzySet3_inputBreakPts = [minXPosErrorRange 7 10 13 maxXPosErrorRange];

xPosErrorFuzzySet4_Func = 'PS';
xPosErrorFuzzySet4_output = [0 0 1 0];
xPosErrorFuzzySet4_inputBreakPts = [minXPosErrorRange 10 13 maxXPosErrorRange];

xPosErrorFuzzySet5_Func = 'PM';
xPosErrorFuzzySet5_output = [0 0 1];
xPosErrorFuzzySet5_inputBreakPts = [minXPosErrorRange 13 maxXPosErrorRange];

xPosErrorFuzzySet_Funcs =
{xPosErrorFuzzySet1_Func;xPosErrorFuzzySet2_Func;xPosErrorFuzzySet3_Func;xPosErrorF
uzzySet4_Func;xPosErrorFuzzySet5_Func};
xPosErrorFuzzySet_output =
{xPosErrorFuzzySet1_output;xPosErrorFuzzySet2_output;xPosErrorFuzzySet3_output;xPos
ErrorFuzzySet4_output;xPosErrorFuzzySet5_output};
xPosErrorFuzzySet_inputBreakPts =
{xPosErrorFuzzySet1_inputBreakPts;xPosErrorFuzzySet2_inputBreakPts;xPosErrorFuzzySe
t3_inputBreakPts;xPosErrorFuzzySet4_inputBreakPts;xPosErrorFuzzySet5_inputBreakPts}
;

numxPosErrorFuzzySets = size(xPosErrorFuzzySet_Funcs,1);

contOutputFuzzySet1_Func = 'NM';
contOutputFuzzySet1_output = [1 0 0];
contOutputFuzzySet1_inputBreakPts = [minControlAngleRange -10
maxControlAngleRange];

contOutputFuzzySet2_Func = 'NS';
contOutputFuzzySet2_output = [0 1 0 0];
contOutputFuzzySet2_inputBreakPts = [minControlAngleRange -10 0
maxControlAngleRange];

contOutputFuzzySet3_Func = 'ZE';
contOutputFuzzySet3_output = [0 0 1 0 0];
contOutputFuzzySet3_inputBreakPts = [minControlAngleRange -10 0 10
maxControlAngleRange];

contOutputFuzzySet4_Func = 'PS';
contOutputFuzzySet4_output = [0 0 1 0];
contOutputFuzzySet4_inputBreakPts = [minControlAngleRange 0 10
maxControlAngleRange];

contOutputFuzzySet5_Func = 'PM';
contOutputFuzzySet5_output = [0 0 1];
contOutputFuzzySet5_inputBreakPts = [minControlAngleRange 10 maxControlAngleRange];

contOutputFuzzySet_Funcs =
{contOutputFuzzySet1_Func;contOutputFuzzySet2_Func;contOutputFuzzySet3_Func;contOut
putFuzzySet4_Func;contOutputFuzzySet5_Func};
contOutputFuzzySet_output =
{contOutputFuzzySet1_output;contOutputFuzzySet2_output;contOutputFuzzySet3_output;c
ontOutputFuzzySet4_output;contOutputFuzzySet5_output};
```

```
contOutputFuzzySet_inputBreakPts =
{contOutputFuzzySet1_inputBreakPts;contOutputFuzzySet2_inputBreakPts;contOutputFuzz
ySet3_inputBreakPts;contOutputFuzzySet4_inputBreakPts;contOutputFuzzySet5_inputBrea
kPts};

numContOutputFuzzySets = size(contOutputFuzzySet_Funcs,1);

plotFig = 1;
if(plotFig)

plotMembershipFunctions(contOutputFuzzySet_output,contOutputFuzzySet_inputBreakPts,
setOutput,contOutputTextXpos,...
                                    xPosErrorFuzzySet_output,
xPosErrorFuzzySet_inputBreakPts,setXPosError,errorTextXpos,...
                                    TruckAngleErrorFuzzySet_output,
TruckAngleErrorFuzzySet_inputBreakPts,setTruckAngleError,truckAngleErrorTextXpos,..
.
                                    initialXPosError,initialTruckAngleError);
end

truckAngleErrorFuzzyified =
fuzzifiedMemFunc(TruckAngleErrorFuzzySet_Funcs,TruckAngleErrorFuzzySet_output,Truck
AngleErrorFuzzySet_inputBreakPts,initialTruckAngleError,maxTruckAngleError);
xPosErrorFuzzyified =
fuzzifiedMemFunc(xPosErrorFuzzySet_Funcs,xPosErrorFuzzySet_output,xPosErrorFuzzySet
_inputBreakPts,initialXPosError,maxXPosError);

fprintf('Qu2.1 COA Strategy\n\n');
UCOAoutputFiringRules =
determineRules(truckAngleErrorFuzzyified,xPosErrorFuzzyified,'COA-
min',setTruckAngleError,setXPosError,setOutput,FAM_TruckAngleError_XPosError,0);

disp('Fuzzification UCOA - min');
for i = 1:size(UCOAoutputFiringRules,2)
    fprintf('Rule %d: %g/%s \n',i,
UCOAoutputFiringRules{1,i},UCOAoutputFiringRules{2,i});
end

UCOA_vals = cell2mat(UCOAoutputFiringRules(1,:));

UCOA_Rule1_inputBreakPts = [minControlAngleRange
linearPosx(linearEqu([minControlAngleRange 1],[-10 0]),UCOA_vals(1)) -10
maxControlAngleRange];
UCOA_Rule1_output = [UCOA_vals(1) UCOA_vals(1) 0 0];

figure;
for i = 1:numContOutputFuzzySets
str = setOutput{i};
subplot(2,2,1);plot(contOutputFuzzySet_inputBreakPts{i},contOutputFuzzySet_output{i
},'b');
subplot(2,2,1);text(contOutputTextXpos(i),1,str);
hold on;
end
subplot(2,2,1);plot(UCOA_Rule1_inputBreakPts,UCOA_Rule1_output,'b','linewidth',3);
grid on;
title('MF1');

UCOA_Rule2_inputBreakPts = [minControlAngleRange
linearPosx(linearEqu([minControlAngleRange 1],[-10 0]),UCOA_vals(2)) -10
maxControlAngleRange];
UCOA_Rule2_output = [UCOA_vals(2) UCOA_vals(2) 0 0];

for i = 1:numContOutputFuzzySets
str = setOutput{i};
subplot(2,2,2);plot(contOutputFuzzySet_inputBreakPts{i},contOutputFuzzySet_output{i
},'b');
subplot(2,2,2);text(contOutputTextXpos(i),1,str);
hold on;
```

```
end
subplot(2,2,2);plot(UCOA_Rule2_inputBreakPts,UCOA_Rule2_output,'b','linewidth',3);
grid on;
title('MF2');

UCOA_Rule3_inputBreakPts = [minControlAngleRange
linearPosx(linearEqu([minControlAngleRange 1],[-10 0]),UCOA_vals(3)) -10
maxControlAngleRange];
UCOA_Rule3_output = [UCOA_vals(3) UCOA_vals(3) 0 0];

for i = 1:numContOutputFuzzySets
str = setOutput{i};
subplot(2,2,3);plot(contOutputFuzzySet_inputBreakPts{i},contOutputFuzzySet_output{i
},'b');
subplot(2,2,3);text(contOutputTextXpos(i),1,str);
hold on;
end
subplot(2,2,3);plot(UCOA_Rule3_inputBreakPts,UCOA_Rule3_output,'b','linewidth',3);
grid on;
title('MF3');

UCOA_Rule4_inputBreakPts = [minControlAngleRange
linearPosx(linearEqu([minControlAngleRange 1],[-10 0]),UCOA_vals(4)) -10
maxControlAngleRange];
UCOA_Rule4_output = [UCOA_vals(4) UCOA_vals(4) 0 0];

for i = 1:numContOutputFuzzySets
str = setOutput{i};
subplot(2,2,4);plot(contOutputFuzzySet_inputBreakPts{i},contOutputFuzzySet_output{i
},'b');
subplot(2,2,4);text(contOutputTextXpos(i),1,str);
hold on;
end
subplot(2,2,4);plot(UCOA_Rule4_inputBreakPts,UCOA_Rule4_output,'b','linewidth',3);
grid on;
title('MF4');

figure;
for i = 1:numContOutputFuzzySets
str = setOutput{i};
plot(contOutputFuzzySet_inputBreakPts{i},contOutputFuzzySet_output{i},'b');
text(contOutputTextXpos(i),1,str);
hold on;
end
plot(UCOA_Rule1_inputBreakPts,UCOA_Rule1_output,'b','linewidth',3);

for i = 1:numContOutputFuzzySets
plot(contOutputFuzzySet_inputBreakPts{i},contOutputFuzzySet_output{i},'b');
hold on;
end
plot(UCOA_Rule2_inputBreakPts,UCOA_Rule2_output,'b','linewidth',3);
grid on;

for i = 1:numContOutputFuzzySets
plot(contOutputFuzzySet_inputBreakPts{i},contOutputFuzzySet_output{i},'b');
hold on;
end
plot(UCOA_Rule3_inputBreakPts,UCOA_Rule3_output,'b','linewidth',3);

for i = 1:numContOutputFuzzySets
plot(contOutputFuzzySet_inputBreakPts{i},contOutputFuzzySet_output{i},'b');
hold on;
end
plot(UCOA_Rule4_inputBreakPts,UCOA_Rule4_output,'b','linewidth',3);
grid on;

defuzz_output = [UCOA_vals(3) UCOA_vals(3) 0];
```

```
defuzz_breakPts = [minControlAngleRange linearPosx(linearEqu([minControlAngleRange
1],[-10 0]),UCOA_vals(3)) -10];

plot(defuzz_breakPts,defuzz_output,'--r','linewidth',3);
grid on;
title('Defuzzification');

% y = mx + b
% [m b]
func1 = [0 UCOA_vals(3)];linearEqu([minControlAngleRange 1],[-10 0]);
func2 = linearEqu([minControlAngleRange 1],[-10 0]);
% func3 = linearEqu([-10 1],[0 0]);
% func4 = [0 UCOA_vals(1)];
% func5 = linearEqu([0 1],[10 0]);

defuzz_Funcs = [func1 func2];
areas = zeros(1,size(defuzz_breakPts,2)-1);
moments = zeros(1,size(defuzz_breakPts,2)-1);

fprintf('\n');
disp('Defuzzification Piecewise Functions');
for i = 1:(size(defuzz_breakPts,2)-1)
    if(defuzz_Funcs(2*i-1) ~= 0)
        fprintf('Function %d between  %g and %g is u(U) = %g*u + %g \n',i,
defuzz_breakPts(i),defuzz_breakPts(i+1),defuzz_Funcs(2*i-1),defuzz_Funcs(2*i));
    else
        fprintf('Function %d between  %g and %g is u(U) = %g \n',i,
defuzz_breakPts(i),defuzz_breakPts(i+1),defuzz_Funcs(2*i));
    end
    areas(i) = integral(@(x)(defuzz_Funcs(2*i-1)*x +
defuzz_Funcs(2*i)),defuzz_breakPts(i),defuzz_breakPts(i+1));
    moments(i) = integral(@(x)(defuzz_Funcs(2*i-1)*(x.*x) +
defuzz_Funcs(2*i)*x),defuzz_breakPts(i),defuzz_breakPts(i+1));
    fprintf('Area %d = %g\n',i,areas(i));
    fprintf('Moment %d = %g\n',i,moments(i));
end

fprintf('\nCOA Method (max-min) - Defuzzified Output for initialXPosError = %g m,
initialTruckAngleError = %g deg\n',initialXPosError,initialTruckAngleError);
fprintf('Total Area = %g\n',sum(areas));
fprintf('Total Moment = %g\n',sum(moments));
outputAngle = sum(moments)/sum(areas);
fprintf('theta(%d) = %g deg\n',1,outputAngle);
state2Vec_COA =
truckBackerUpperDynamics(initialState(PHI),outputAngle,initialState(X),initialState
(Y));
fprintf('[phi(%d), x(%d), y(%d)] = [%g, %g,
%g]\n\n',2,2,2,state2Vec_COA(1,PHI),state2Vec_COA(1,X),state2Vec_COA(1,Y));

text(outputAngle,0.6,strcat('COA = ',num2str(outputAngle)));
plot([outputAngle outputAngle],[0 1],'--g','linewidth',2);

fprintf('\nQu2.2 MOM Strategy\n\n');

numSamplingPoints = 100;
% initialState = zeros(1,3);
% initialState(X) = 15;
% initialState(Y) = 15;
% initialState(PHI) = 35;
initialOutputAngle = 0;
controlAngleTheta = zeros(numSamplingPoints,1);
errorVec = zeros(numSamplingPoints,2);
stateVec = zeros(numSamplingPoints,3);
stateVec(1,:) = initialState;
findDominant = 0;

for i = 1:(numSamplingPoints-1)
```

```
    XPosError = checkInputRange(minXPosErrorRange,maxXPosErrorRange,stateVec(i,X));
    TruckAngleError =
checkInputRange(minTruckAngleErrorRange,maxTruckAngleErrorRange,stateVec(i,PHI));
    errorVec(i,:) = [xDesired - XPosError,phiDesired - TruckAngleError];

    truckAngleErrorFuzzyified =
fuzzifiedMemFunc(TruckAngleErrorFuzzySet_Funcs,TruckAngleErrorFuzzySet_output,Truck
AngleErrorFuzzySet_inputBreakPts,TruckAngleError,maxTruckAngleError);
    xPosErrorFuzzyified =
fuzzifiedMemFunc(xPosErrorFuzzySet_Funcs,xPosErrorFuzzySet_output,xPosErrorFuzzySet
_inputBreakPts,XPosError,maxXPosError);

    [UMOMoutputFiringRules,FAM_TruckAngleError_XPosError] =
determineRules(truckAngleErrorFuzzyified,xPosErrorFuzzyified,'MOM-
prod',setTruckAngleError,setXPosError,setOutput,FAM_TruckAngleError_XPosError,findD
ominant);
    UMOM_crispVals = zeros(1,size(UMOMoutputFiringRules,2));
    for j = 1:size(UMOM_crispVals,2)
        logical_row =
cellfun(cellfind(UMOMoutputFiringRules{2,j}),contOutputFuzzySet_Funcs);
        output = contOutputFuzzySet_output(logical_row);
        inputBreakPts = contOutputFuzzySet_inputBreakPts(logical_row);
        inputBreakPtsArray = inputBreakPts{:};
        UMOM_crispVals(j) = mean(inputBreakPtsArray(find(output{:} ==1)));
    end

    controlAngleTheta(i) =
dot(cell2mat(UMOMoutputFiringRules(1,:)),UMOM_crispVals)/sum(cell2mat(UMOMoutputFir
ingRules(1,:)));
    stateVec(i+1,:) =
truckBackerUpperDynamics(stateVec(i,PHI),controlAngleTheta(i),stateVec(i,X),stateVe
c(i,Y));

    if(i==1 || i ==2)
        disp(strcat('Fuzzification UMOM -prod: State: ',num2str(i)));
        for j = 1:size(UMOMoutputFiringRules,2)
            fprintf('Rule %d: %g/%s \n',j,
UMOMoutputFiringRules{1,j},UMOMoutputFiringRules{2,j});
        end

        fprintf('MOM Method (prod) - Defuzzified Output for xPosError = %g m,
truckAngleError = %g deg\n',XPosError,TruckAngleError);
        fprintf('theta(%d) = %g deg\n',i,controlAngleTheta(i));
        fprintf('[phi(%d), x(%d), y(%d)] = [%g, %g,
%g]\n\n',i+1,i+1,i+1,stateVec(i+1,PHI),stateVec(i+1,X),stateVec(i+1,Y));
    end
end

figure;
h1 = plot(stateVec(1,X),stateVec(1,Y),'md','MarkerFaceColor','m','MarkerSize',8);
hold on;
plot(stateVec(1:end,X),stateVec(1:end,Y),'m*-');
grid on;
title('Truck position');
xlabel('X position');
ylabel('Y position');
legend([h1],'Starting Point');
grid on;

figure;
plot(controlAngleTheta,'k*-');
xlabel('Sample Point');
ylabel('Control Angle (\theta deg)');
grid on;

figure;
subplot(2,1,1);plot(errorVec(:,X),'r*-');
grid on;
```

```
ylabel('X Position Error (m)');
xlabel('Sample Point');
subplot(2,1,2);plot(errorVec(:,2),'b*-');
ylabel('Truck Angle Error (\phi deg)');
xlabel('Sample Point');
grid on;

fprintf('\nQu2.2 MOM Strategy - Dominant Rule\n\n');

controlAngleThetaModRule = zeros(numSamplingPoints,1);
errorVecModRule = zeros(numSamplingPoints,2);
stateVecModRule = zeros(numSamplingPoints,3);
stateVecModRule(1,:) = initialState;
findDominant = 1;

for i = 1:(numSamplingPoints-1)

    XPosError =
checkInputRange(minXPosErrorRange,maxXPosErrorRange,stateVecModRule(i,X));
    TruckAngleError =
checkInputRange(minTruckAngleErrorRange,maxTruckAngleErrorRange,stateVecModRule(i,P
HI));
    errorVecModRule(i,:) = [xDesired - XPosError,phiDesired - TruckAngleError];

    truckAngleErrorFuzzyified =
fuzzifiedMemFunc(TruckAngleErrorFuzzySet_Funcs,TruckAngleErrorFuzzySet_output,Truck
AngleErrorFuzzySet_inputBreakPts,TruckAngleError,maxTruckAngleError);
    xPosErrorFuzzyified =
fuzzifiedMemFunc(xPosErrorFuzzySet_Funcs,xPosErrorFuzzySet_output,xPosErrorFuzzySet
_inputBreakPts,XPosError,maxXPosError);

    [UMOMoutputFiringRules,FAM_TruckAngleError_XPosError] =
determineRules(truckAngleErrorFuzzyified,xPosErrorFuzzyified,'MOM-
prod',setTruckAngleError,setXPosError,setOutput,FAM_TruckAngleError_XPosError,findD
ominant);
    UMOM_crispVals = zeros(1,size(UMOMoutputFiringRules,2));
    for j = 1:size(UMOM_crispVals,2)
        logical_row =
cellfun(cellfind(UMOMoutputFiringRules{2,j}),contOutputFuzzySet_Funcs);
        output = contOutputFuzzySet_output(logical_row);
        inputBreakPts = contOutputFuzzySet_inputBreakPts(logical_row);
        inputBreakPtsArray = inputBreakPts{:};
        UMOM_crispVals(j) = mean(inputBreakPtsArray(find(output{:} ==1)));
    end

    controlAngleThetaModRule(i) =
dot(cell2mat(UMOMoutputFiringRules(1,:)),UMOM_crispVals)/sum(cell2mat(UMOMoutputFir
ingRules(1,:)));
    stateVecModRule(i+1,:) =
truckBackerUpperDynamics(stateVecModRule(i,PHI),controlAngleThetaModRule(i),stateVe
cModRule(i,X),stateVecModRule(i,Y));

    if(i==1 || i ==2)
        disp(strcat('Fuzzification UMOM -prod: State: ',num2str(i)));
        for j = 1:size(UMOMoutputFiringRules,2)
            fprintf('Rule %d: %g/%s \n',j,
UMOMoutputFiringRules{1,j},UMOMoutputFiringRules{2,j});
        end

        fprintf('MOM Method (prod) - Defuzzified Output for xPosError = %g m,
truckAngleError = %g deg\n',XPosError,TruckAngleError);
        fprintf('theta*(%d) = %g deg\n',i,controlAngleThetaModRule(i));
        fprintf('[phi(%d), x(%d), y(%d)] = [%g, %g,
%g]\n\n',i+1,i+1,i+1,stateVecModRule(i+1,PHI),stateVecModRule(i+1,X),stateVecModRul
e(i+1,Y));
    end

    if(i == 1)
```

```
        fprintf('The modified FAM table is: \n');
        disp(FAM_TruckAngleError_XPosError);
        findDominant = 0;
    end
end

figure;
h1 =
plot(stateVecModRule(1,X),stateVecModRule(1,Y),'kd','MarkerFaceColor','k','MarkerSi
ze',8);
hold on;
h2=plot(stateVecModRule(1:end,X),stateVecModRule(1:end,Y),'b*-');
hold on;
h3=plot(stateVec(1:end,X),stateVec(1:end,Y),'m*-');
grid on;
title('Truck position - Modifying the Dominant Rule');
xlabel('X position');
ylabel('Y position');
legend([h1 h2 h3],'Starting Point','Modified Rule','Generic Rule');
grid on;

figure;
p1=plot(controlAngleThetaModRule,'k*-');
hold on;
p2=plot(controlAngleTheta,'b*-');
title('Control Angle Output - Comparison');
legend([p1 p2],'Modified Rule \theta*','Generic Rule \theta');
xlabel('Sample Point');
ylabel('Control Angle (deg)');
grid on;

figure;
subplot(2,1,1);p1=plot(errorVecModRule(:,X),'g*-');
hold on;
subplot(2,1,1);p2=plot(errorVec(:,X),'r*-');
grid on;
legend([p1 p2],'Modified Rule','Generic Rule');
title('Error Output - Comparison');
ylabel('X Position Error (m)');
xlabel('Sample Point');
subplot(2,1,2);p1=plot(errorVecModRule(:,2),'b*-');
hold on;
subplot(2,1,2);p2=plot(errorVec(:,2),'m*-');
legend([p1 p2],'Modified Rule','Generic Rule');
ylabel('Truck Angle Error (\phi deg)');
xlabel('Sample Point');
grid on;
```