|  |  |  |  |
| --- | --- | --- | --- |
|  | **(-5,1)** | **(0,0)** | **(1,5)** |
| **R1** | 0.7109 | 0.2500 | 0.0005 |
| **R2** | 0.2891 | 0.2500 | 0.0469 |
| **R3** | 0.0000 | 0.2500 | 0.0105 |
| **R4** | 0.0000 | 0.2500 | 0.9421 |

Table 1 - Rule weights for given inputs

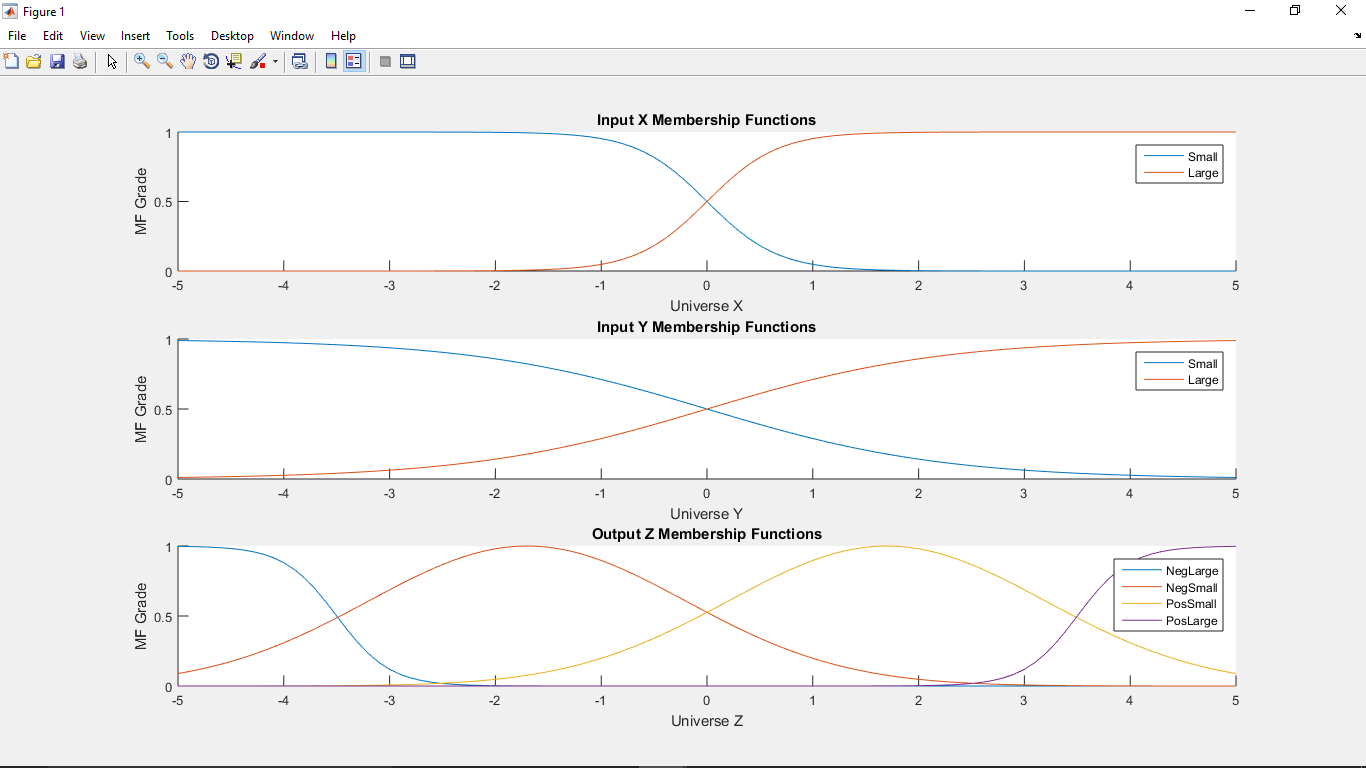


Figure 1 – X, Y, Z Membership Functions

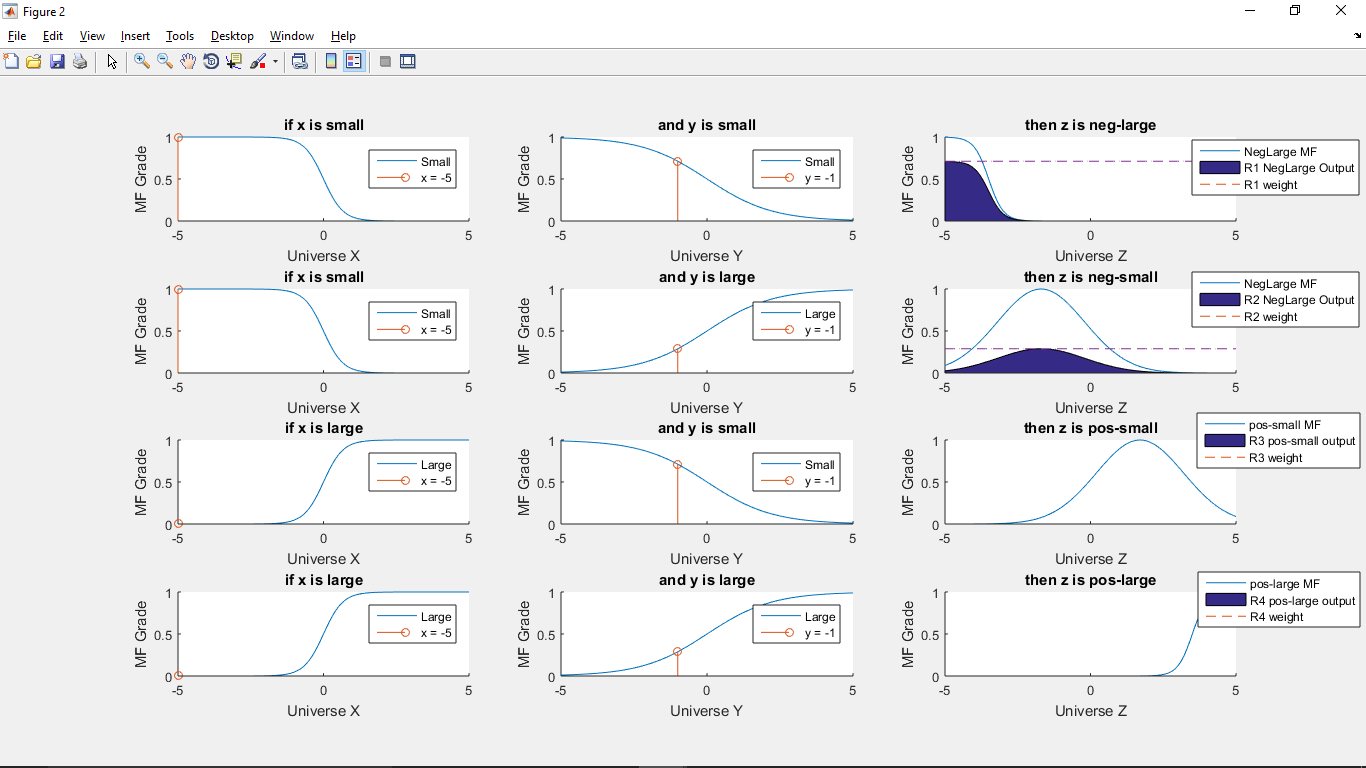


Figure 2 – input (-5,-1)

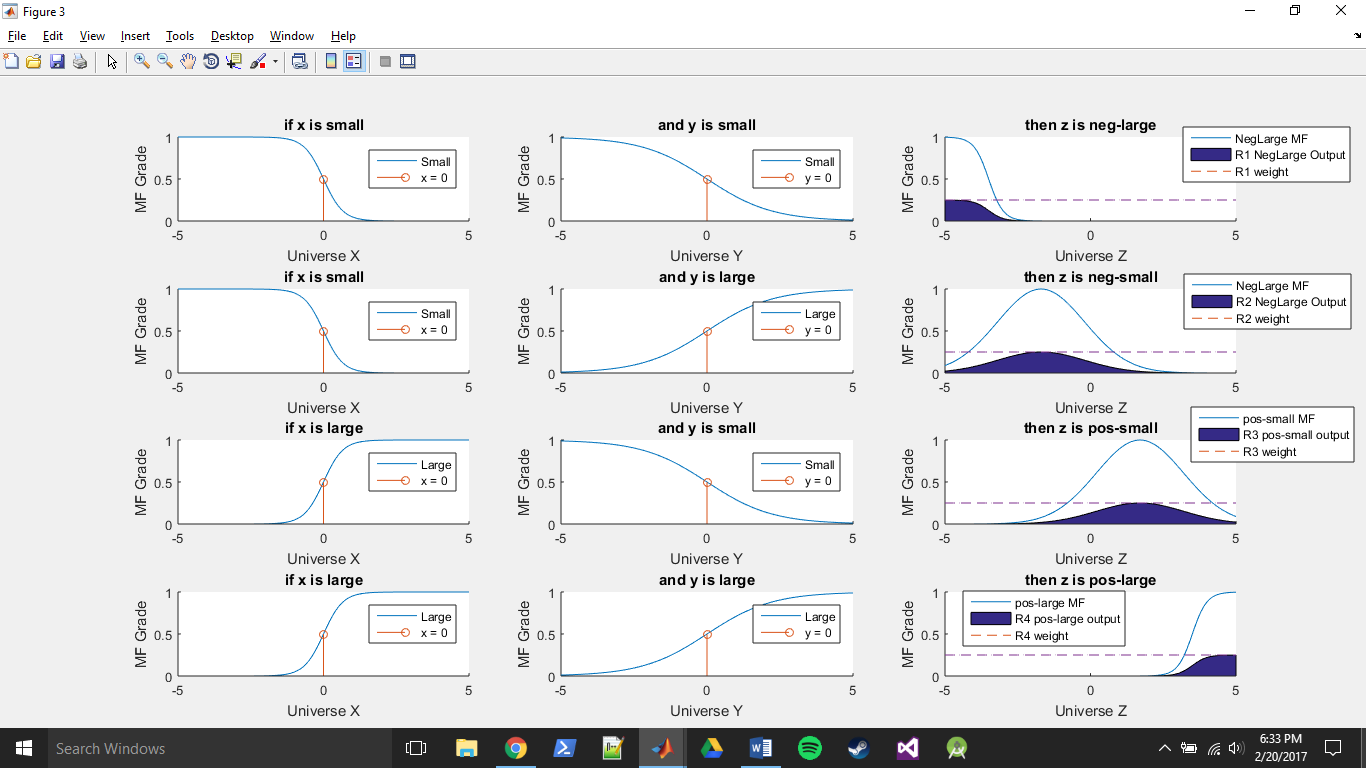


Figure 3 – input (0,0)

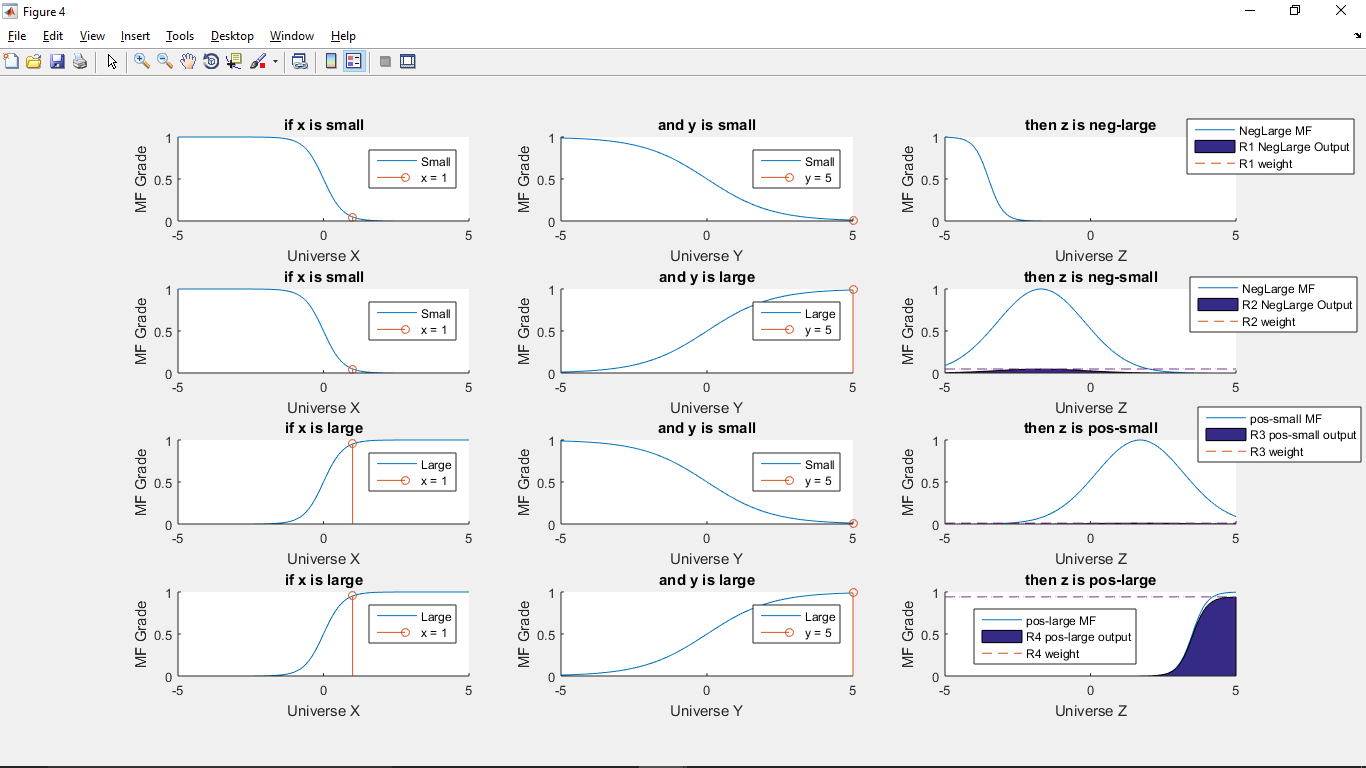


Figure 4 – input (1,5)

% Matlab code implementing Product Inference Engine (PIE)  
x = [-5:.1:5];  
y = x;

% X input MF's

xsmall = 1./(1+exp(-(-3)\*(x-0)));

xlarge = 1./(1+exp(-(3)\*(x-0)));

subplot(3,1,1);

hold on

plot(x,xsmall);

plot(x,xlarge);

xlabel('Universe X');

ylabel('MF Grade');

title('Input X Membership Functions');

legend('Small', 'Large')

% Y input MF's

ysmall = 1./(1+exp(-(-0.9)\*(y-0)));

ylarge = 1./(1+exp(-(0.9)\*(y-0)));

subplot(3,1,2);

hold on

plot(y,ysmall);

plot(y,ylarge);

xlabel('Universe Y');

ylabel('MF Grade');

title('Input Y Membership Functions');

legend('Small', 'Large')

% Z output MF's

zneglarge = 1./(1+exp(-(-4)\*(x-(-3.5))));

znegsmall = exp(-0.5\*(((x-(-1.7))/1.5).^2));

zpossmall = exp(-0.5\*(((x-(1.7))/1.5).^2));

zposlarge = 1./(1+exp(-(4)\*(x-(3.5))));

subplot(3,1,3);

hold on

plot(x,zneglarge);

plot(x,znegsmall);

plot(x,zpossmall);

plot(x,zposlarge);

xlabel('Universe Z');

ylabel('MF Grade');

title('Output Z Membership Functions');

legend('NegLarge', 'NegSmall', 'PosSmall', 'PosLarge')

%% Product Inference Engine (PIE)

% Crisp inputs are used as fuzzy singleton facts

xfacts = [-5 0 1];

yfacts = [-1 0 5];

% Evaluate X and Y membership functions at these fact values

% this output known as "firing strength" ?

xfacts\_xsmall\_MFgrade = 1./(1+exp(-(-3)\*(xfacts-0)));

xfacts\_xlarge\_MFgrade = 1./(1+exp(-(3)\*(xfacts-0)));

yfacts\_ysmall\_MFgrade = 1./(1+exp(-(-0.9)\*(yfacts-0)));

yfacts\_ylarge\_MFgrade = 1./(1+exp(-(0.9)\*(yfacts-0)));

% Rule 1, if x is small and y is small then z is neglarge

wR1 = xfacts\_xsmall\_MFgrade .\* yfacts\_ysmall\_MFgrade; % "rule weight" from inputs (antecedents)

R1\_zneglarge = wR1' \* zneglarge;

% Rule 2, if x is small and y is large then z is negsmall

wR2 = xfacts\_xsmall\_MFgrade .\* yfacts\_ylarge\_MFgrade;

R2\_znegsmall = wR2' \* znegsmall;

% Rule 3, if x is large and y is small then z is possmall

wR3 = xfacts\_xlarge\_MFgrade .\* yfacts\_ysmall\_MFgrade;

R3\_zpossmall = wR3' \* zpossmall;

% Rule 4, if x is large and y is large then z is poslarge

wR4 = xfacts\_xlarge\_MFgrade .\* yfacts\_ylarge\_MFgrade;

R4\_zposlarge = wR4' \* zposlarge;

% Plotting code left out for brevity