Año 2021

Inteligencia Artificial

Trabajo Práctico Nº 2

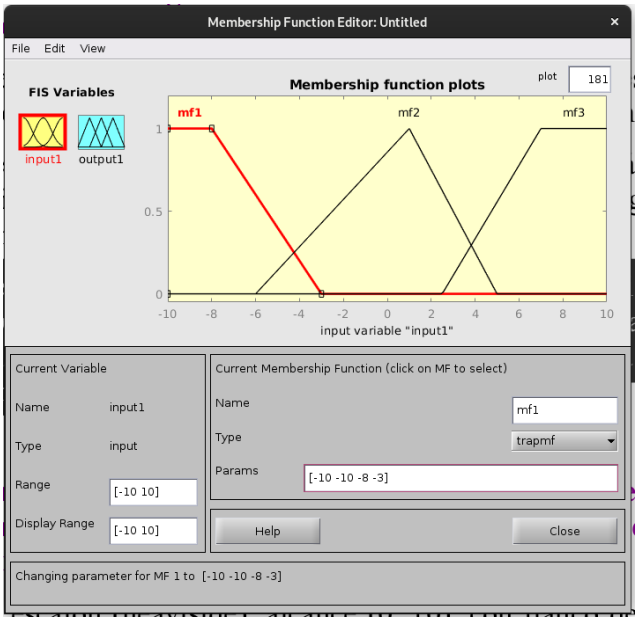
"Lógica Fuzzy"

Grupo: Neural

Alumnos:

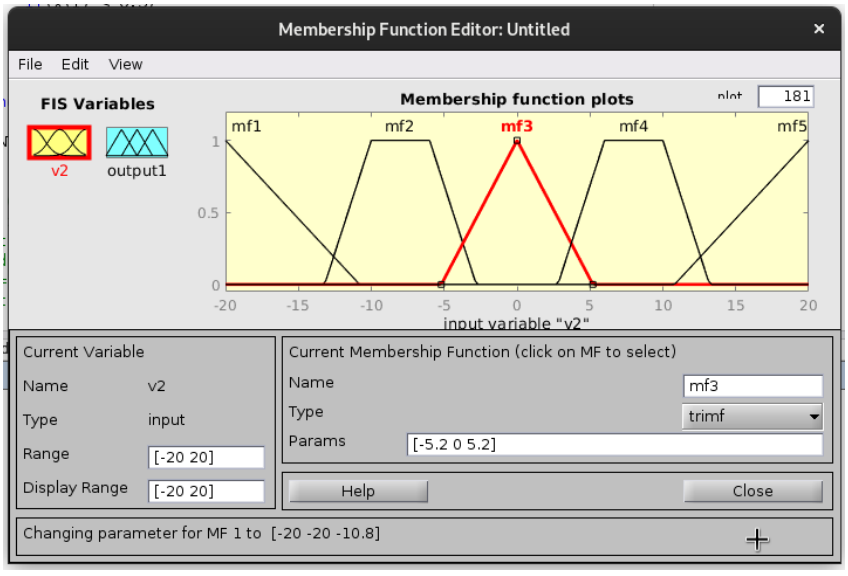
Micaela del valle Castillo	LU:6271	Ing.Inf.
Eduardo Matias Luna	LU: 4843	Ing. Inf.
Eduardo Matias Luna	LU: 681	Lic. Sist.
Judith Graciela Patiño	LU: 4520	Ing. Inf.

1. **Funciones de pertenencia.**
   a. **Utilizando la interfaz gráfica de los sistemas de inferencia, definir por separado las variables que se indican:**
      i. **variable (V1): alcance (-10,+10). Tres particiones lineales hombro-triángulo-hombro con parámetros (-10, -8, -3), (-6, 1, 5), (2.5, 7, 10) respectivamente.**



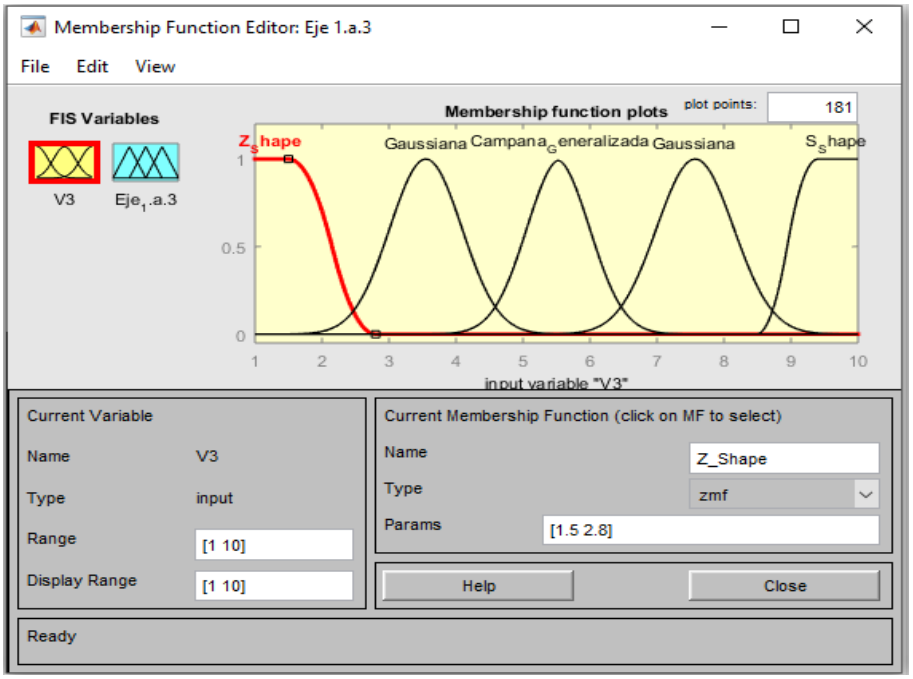
https://github.com/matness-university/U2-fuzzy-logic/blob/main/1_Funciones_de_pertenencia/1_grafica.fis

      ii. **variable (V2): alcance (-20, 20). Cinco particiones tipo triángulo rectángulo izquierdo - trapecio - triángulo - trapecio - triángulo rectángulo derecho, igualmente espaciados con solapamiento del 30%.**



de particiones:
[-20 -20 -10.8]
[-13.2 -10 -6 -2.8]
[-5.2 0 5.2]
[2.8 6 10 13.2]
[10.8 20 60]

iii. **variable (V3): alcance (0, 10). Cinco particiones con funciones continuas y derivables Z shape (zmf()) - gaussiana - campana generalizada - gaussiana - S shape (smf()). Parámetros a definir por el usuario para una distribución simétrica y un solapamiento aproximado al 25%.**

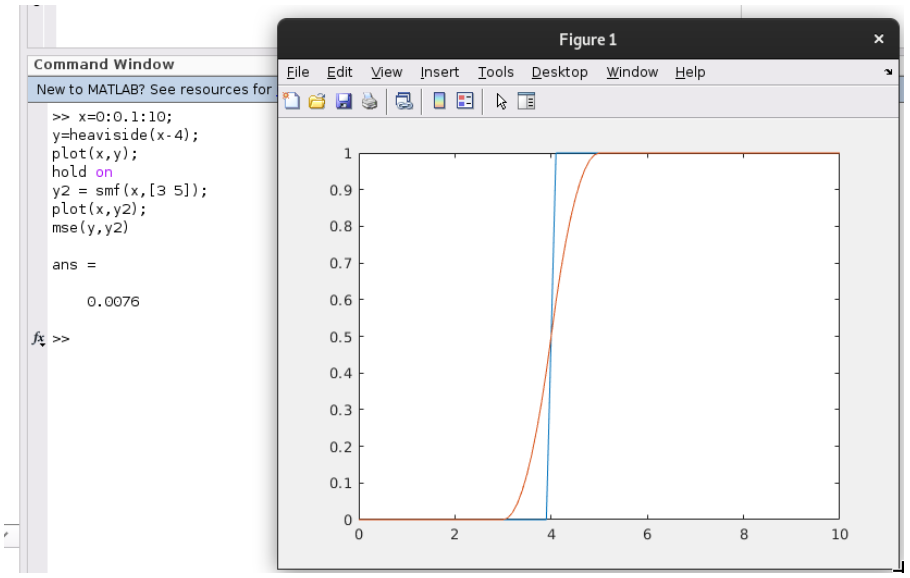


https://github.com/matness-university/U2-fuzzy-logic/blob/main/1_Funciones_de_pertenencia/2_grafica_v2.fis

b. **Un sistema de lógica fuzzy debe reemplazar dos funciones lineales por tramos, con funciones continuas y derivables, de manera que el error cuadrático medio sea menor a 10^-2 . Calcular los parámetros en cada caso:**
   i. **función escalón (heaviside): alcance (0, 10), con flanco decreciente en 4. Reemplazar con función hombro derecho (z-shape) o función sigmoide según convenga.**

```
x=0:0.1:10;
y=heaviside(x-4);
plot(x,y);
```

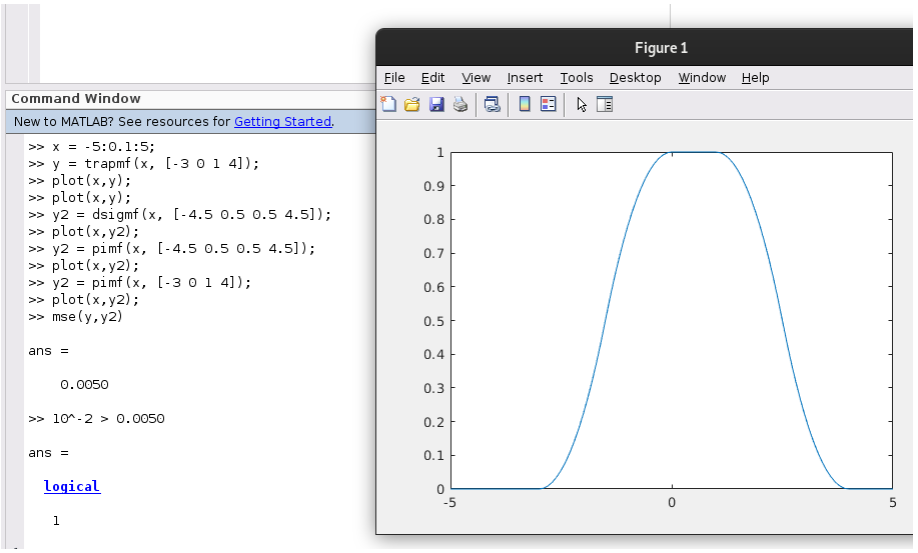
```
hold on
y2 = smf(x,[3 5]);
plot(x,y2);
mse(y,y2)
```



https://github.com/matness-university/U2-fuzzy-logic/blob/main/1_Funciones_de_pertenencia/b_1.m

ii. función trapecio (trapmf): alcance (-5, 5), con parámetros (-3, 0, 1, 4). Reemplazar con función pi o diferencia de sigmoides según convenga.

```
x = -5:0.1:5;
y = trapmf(x, [-3 0 1 4]);
plot(x,y);
y2 = pimf(x, [-3 0 1 4]);
plot(x,y2);
mse(y,y2)
```

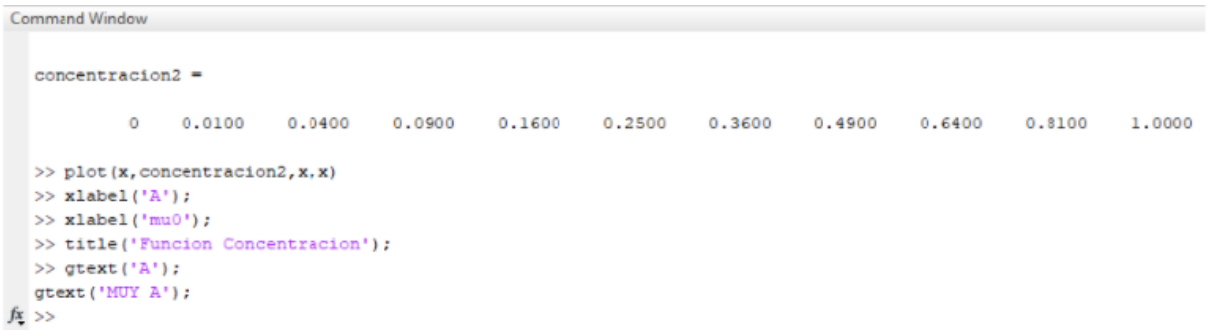


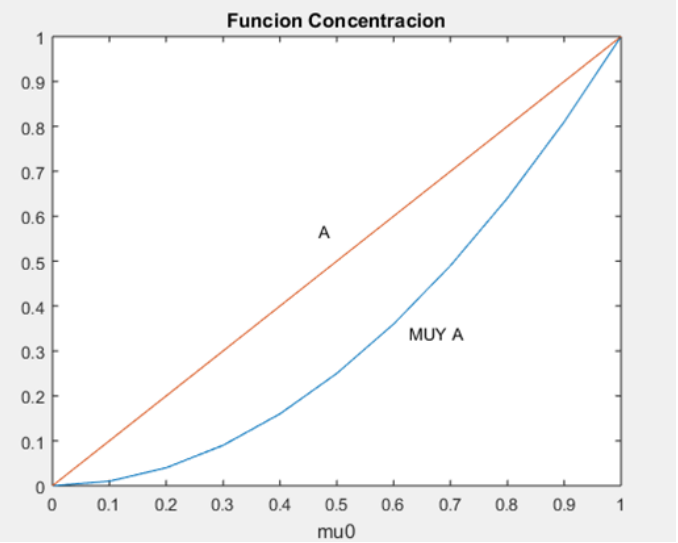
https://github.com/matness-university/U2-fuzzy-logic/blob/main/1_Funciones_de_pertenencia/b_2.m

2. **Definir y escribir los códigos de Matlab que realicen las siguientes funciones. Graficar.**
    a. **Función concentración**

```
function f=funcion_concentracion2(x,mu0)
clc
V2=size(mu0);
if isequal(size(x),size(mu0))
  for i=0;V2
      f=mu0.^2;
  end
else
    disp('Las variables no tienen la misma dimensión')
end
```

https://github.com/gracejujuy/LogicaFuzzy/blob/99480e0ef47dcd3b2e0b757fe6873bf2dfd1e7b7/Funcion_concentracion2.m

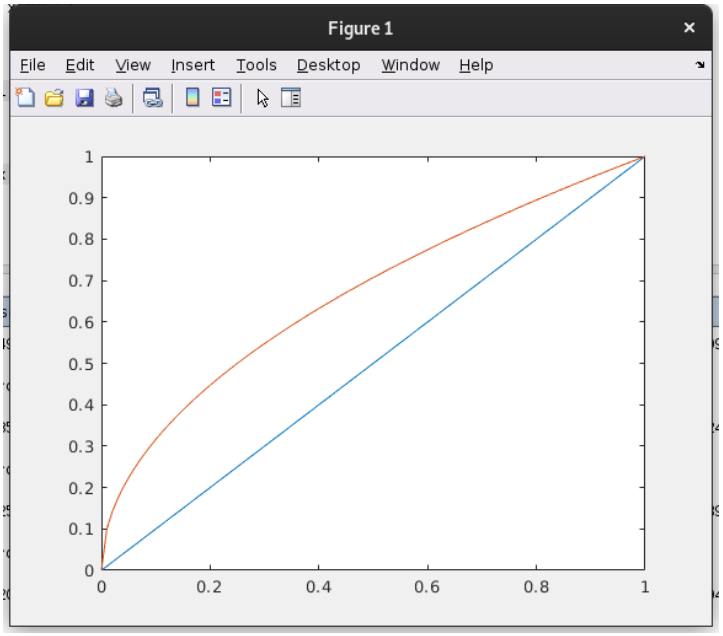




b. Función dilación

```
function DILAC = dilac(x,y)

   if(size(x) ~= size(y))
       error('Los vectores deben tener las mismas dimensiones')
   end

   DILAC = y.^0.5;

end
```

```
x = 0:0.01:1;
y = x;
plot(x,y);
hold on;
y2 = dilac(x,y);
plot(x,y2);
```



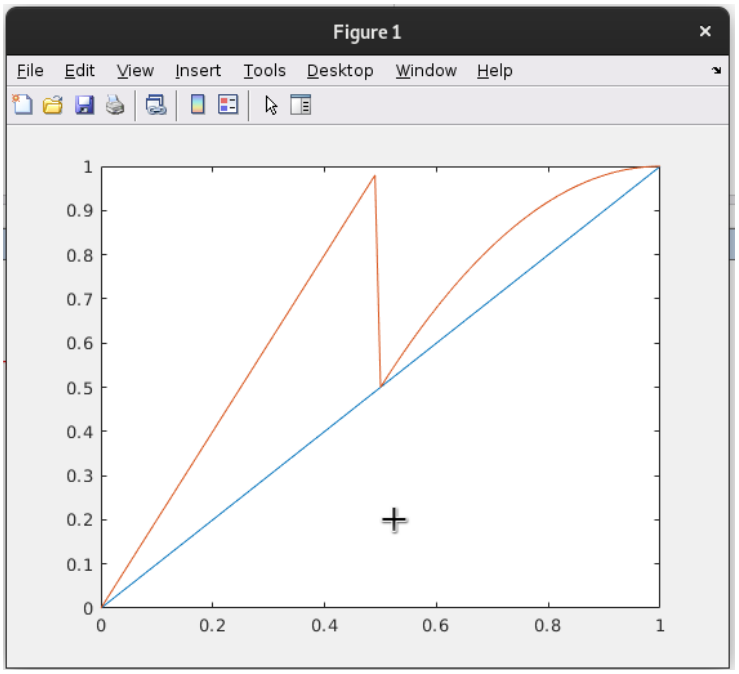
https://github.com/matness-university/U2-fuzzy-logic/blob/main/2_Funciones/dilac.m

c. Función intensificación

```
function INTENS = intens(x,y)

   if(size(x) ~= size(y))
       error('Los vectores deben tener las mismas dimensiones');
   end

   values = zeros(1,length(y));

   for i=1:length(x)
       if(y(i) < 0.5)
           values(i) = 2*y(i);
       else
           values(i) = 1-2*(1-y(i)).^2;
       end
   end

   INTENS = values;
end
```

```
x = 0:0.01:1;
y = x;
plot(x,y);
hold on;
y2 = intens(x,y);
```

```
plot(x,y2);
```



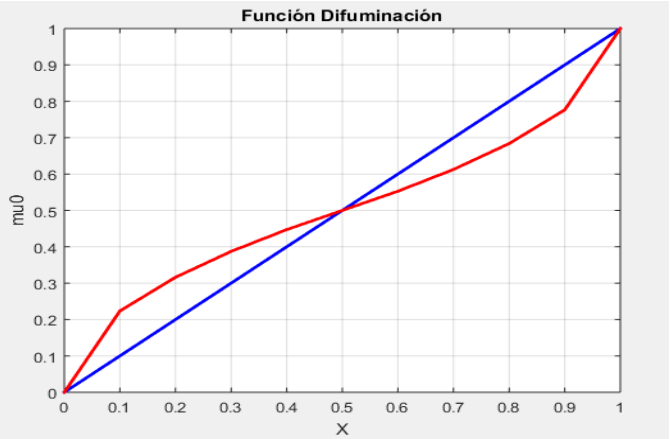
https://github.com/matness-university/U2-fuzzy-logic/blob/main/2_Funciones/intens.m

d. Función difuminación

```
function f=funcion_difuminacion(x,mu0)
clc
[i,j]=size(x);
V2=mu0;
if isequal(size(x),size(mu0))
    for k=1:j
        if V2(i,k)>=0 && V2(i,k)<=0.5
            V2(i,k)=(V2(i,k)/2)^(1/2);
        else
            if V2(i,k)>=0.5 && V2(i,k)<=1
                V2(i,k)=(1-(((1-V2(i,k))/2)^(1/2)));
            end
        end
    end
    f=V2;
else
    disp('Las variables no tienen la misma dimension');
end
end
```

https://github.com/gracejujuy/LogicaFuzzy/blob/99480e0ef47dcd3b2e0b757fe6873bf2dfd1e7b7/funcion_difuminacion.m

```
Command Window

difuminacion =

         0    0.2236    0.3162    0.3873    0.4472    0.5000    0.5528    0.6127    0.6838    0.7764    1.0000

>> plot(x,x,'b','linewidth',2)
>> xlabel('X')
>> ylabel('')
>> ylabel('Y')
>> ylabel('mu0')
>> grid on
>> title('Función Difuminación')
>> hold on
>> plot(x,difuminacion,'r','linewidth',2)
>> hold off
fx >>
```

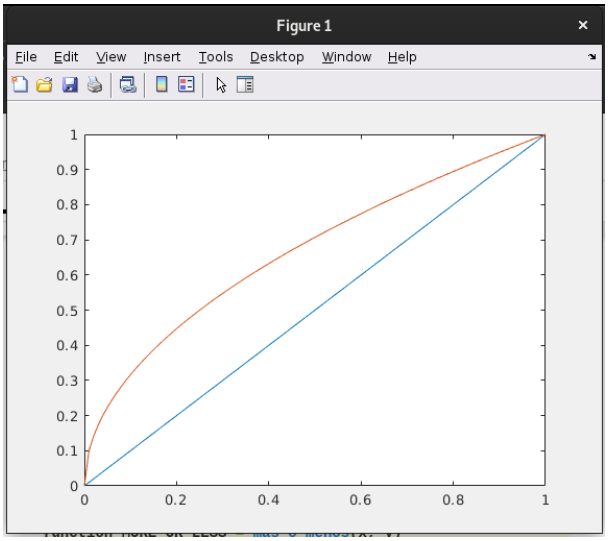


Utilizando las funciones anteriores como base, escribir por lo menos tres funciones representativas de adverbios fuzzy, darles valores en el intervalo [0 ; 1], fuzzyficar y graficar junto a la función de base.

MAS O MENOS(A) = DIL(A)

```
function MORE_OR_LESS = mas_o_menos(x, y)

   MORE_OR_LESS = dilac(x,y);

end
```

```
x = 0:0.01:1;
y = x;
y2 = mas_o_menos(x,y);
plot(x,y);
hold on;
chart = plot(x,y2);
```

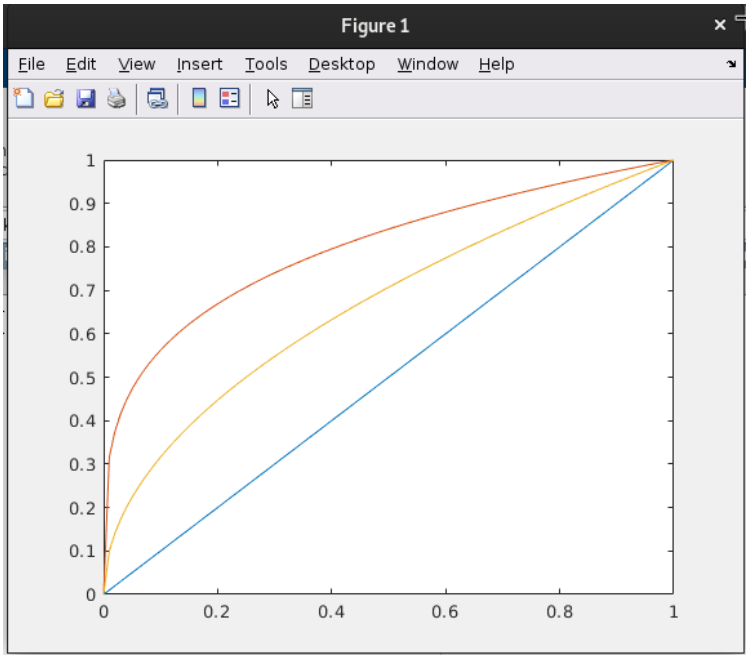


https://github.com/matness-university/U2-fuzzy-logic/blob/main/2_Funciones/mas_o_menos.m

APROXIMADAMENTE(A) = DIL(DIL(A))

```
function APROXIMADAMENTE = aproximadamente(x,y)
   APROXIMADAMENTE = dilac(x, dilac(x,y));
end
```

```
x = 0:0.01:1;
y = x;
y2 = aproximadamente(x,y);
plot(x,y);
hold on;
chart = plot(x,y2);
y3 = dilac(x,y);
plot(x,y3);
```

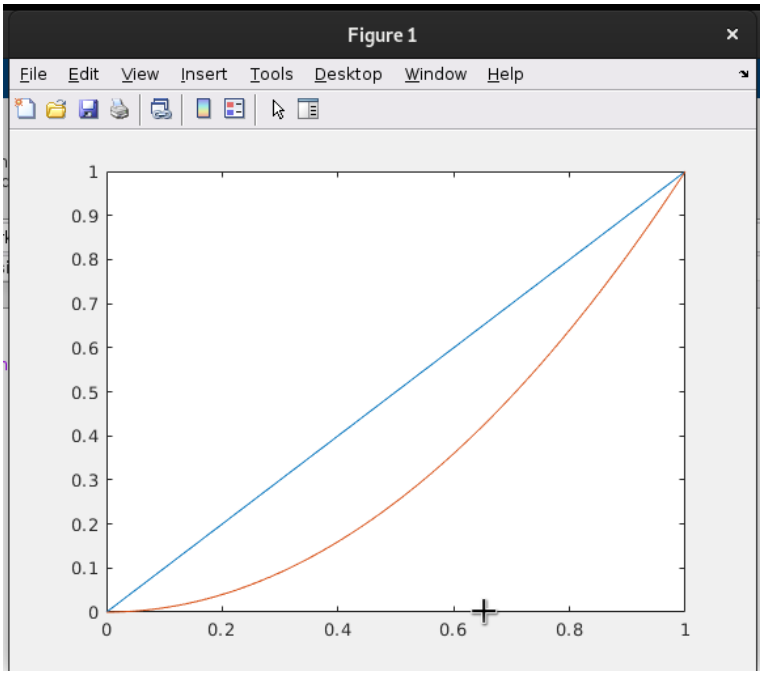


https://github.com/matness-university/U2-fuzzy-logic/blob/main/2_Funciones/aproximadamente.m

MUY(A) = CON(A)

```
function CON = con(x,y)
   if(size(x) ~= size(y))
               error('Los vectores deben tener las mismas
dimensiones');
   end

   CON = y.^2
end
```

```
x = 0:0.01:1;
y = x;
y2 = con(x,y);
plot(x,y);
hold on;
chart = plot(x,y2);
```



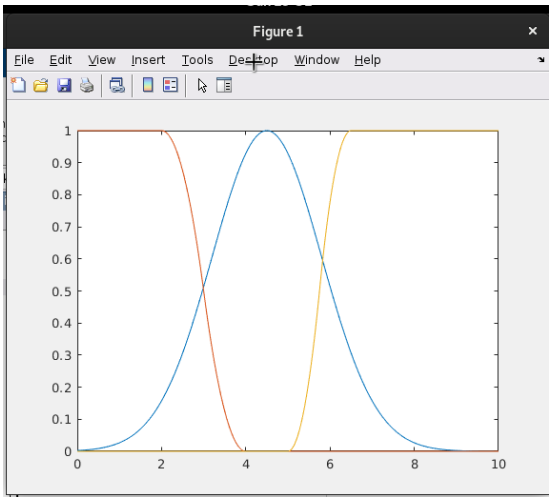
https://github.com/matness-university/U2-fuzzy-logic/blob/main/2_Funciones/con.m

3. **Definir según un criterio coherente, las variables lingüísticas movimiento (particiones "lento", "moderado", "rápido"); y temperamento (particiones "calmado", "insensible" "furioso"). Seleccionar las funciones de pertenencia para cada partición, en un rango [0 ; 10]. Luego, mostrar cómo se componen las funciones de pertenencia cuando se aplican los operadores lógicos que se indican a continuación:**
   a. **Rápido y furioso.**
   b. **Muy lento y algo calmado.**
   c. **No rápido XOR insensible (en este caso conviene descomponer la función XOR en los operadores básicos.**

   mov = movimiento
   U = [0;10]
   T(mov) = [lento, moderado, rápido]
   M(lento) = movimiento entre 0 y 4
   M(moderado) = movimiento entre 0 y 9
   M(rápido) = movimiento entre 5 y 10

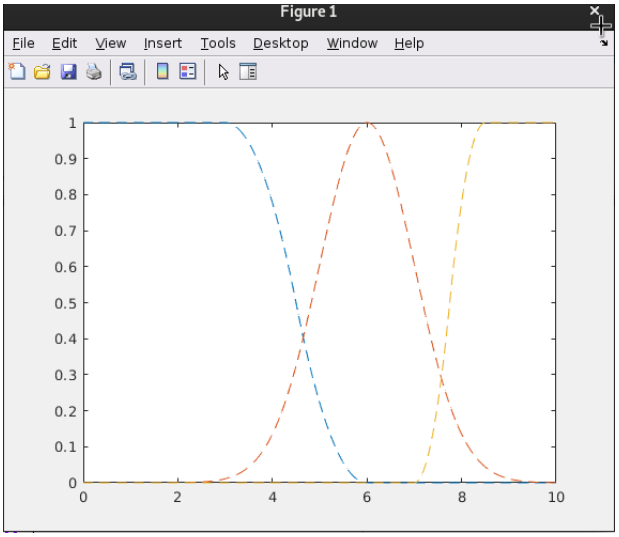
```
x=0:0.01:10;

lento=zmf(x,[2 4]);
moderado = gaussmf(x, [1.3 4.5]);
rapido=smf(x,[5 6.5]);

plot(x,moderado);
```

```
hold on;
plot(x,lento);
plot(x,rapido);
```



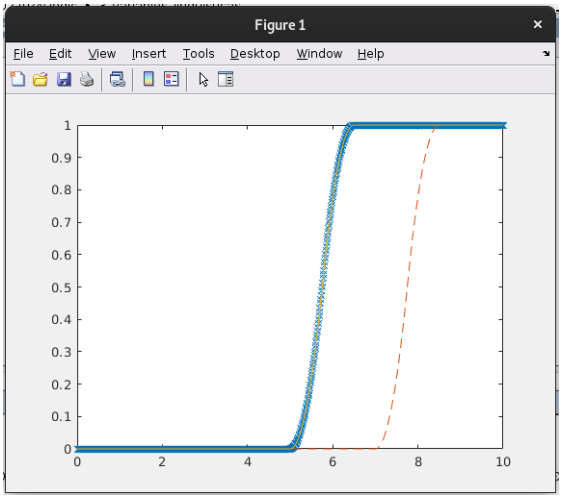
temp = temperamento
U = [0;10]
T(temp) = [calmado, insensible, furioso]
M(calmado) = movimiento entre 0 y 6
M(insensible) = movimiento entre 4 y 8
M(furioso) = movimiento entre 7 y 10

```
x=0:0.01:10;

calmado=zmf(x,[3 6]);
insensible = gaussmf(x, [1 6]);
furioso=smf(x,[7 8.5]);
plot(x,calmado, 'LineStyle', '--');
hold on;
plot(x,insensible, 'LineStyle', '--');
plot(x,furioso, 'LineStyle', '--');
```



Rápido y furioso.

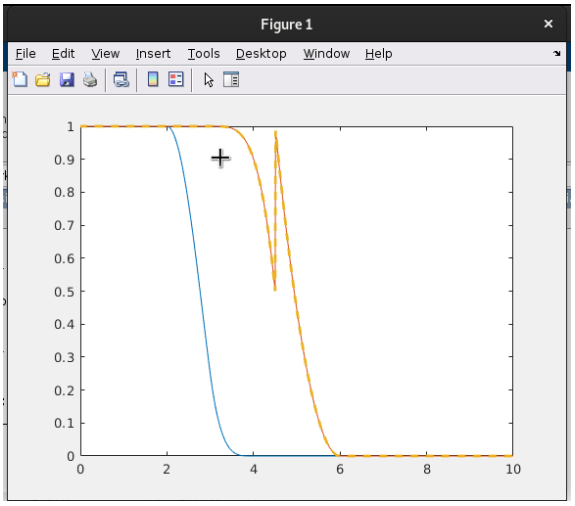
```
rapido_furioso = max(rapido,furioso);
plot(x,rapido_furioso, 'Marker', 'x');
plot(x,furioso, 'LineStyle', '--');
plot(x,rapido);
```



Muy lento y algo calmado.

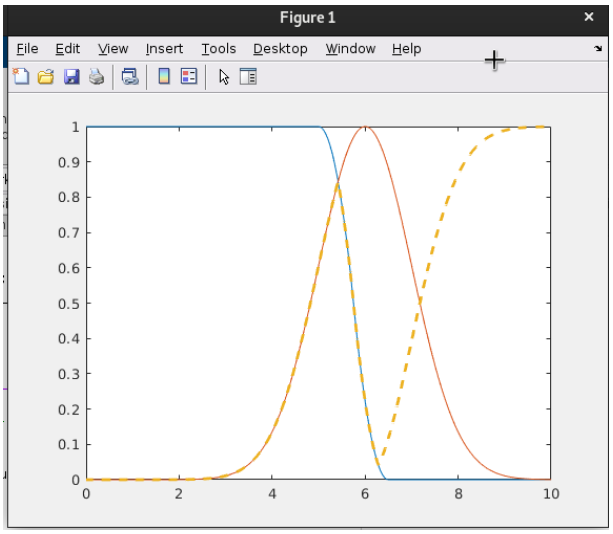
```
muy_lento = lento.^2;
algo_calmado = intens(x,calmado);
muy_lento_algo_calmado = max(muy_lento,algo_calmado)

plot(x,muy_lento);
hold on;
plot(x,algo_calmado);
```

```
plot(x,muy_lento_algo_calmado,      'LineStyle',      '--',
'LineWidth', 2);
```



No rápido XOR insensible (en este caso conviene descomponer la función XOR en los operadores básicos.

```
no_rapido = 1 - rapido;
no_rapido_xor_insensible                                    =
xor_function(no_rapido,insensible);
plot(x,no_rapido);
hold on;
plot(x,insensible);
plot(x,no_rapido_xor_insensible,      'LineStyle',      '--',
'LineWidth', 2);
```



https://github.com/matness-university/U2-fuzzy-logic/blob/main/3_Variables_linguisticas/variables.m

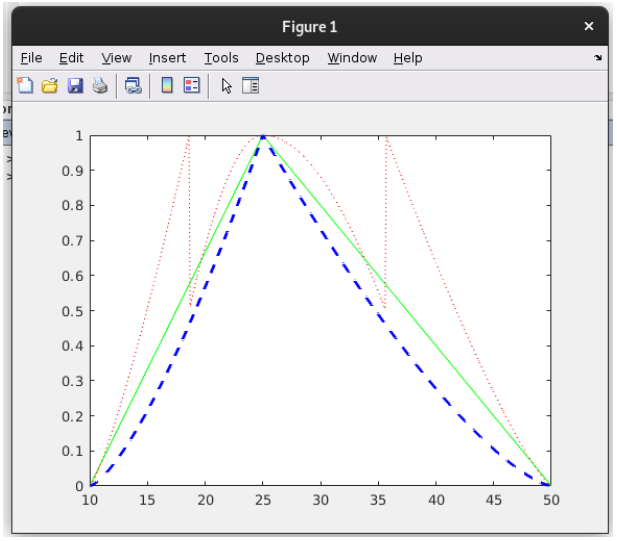
https://github.com/matness-university/U2-fuzzy-logic/blob/main/3_Variables_linguisticas/xor_function.m

Comentar cómo resultan las funciones de pertenencia compuestas (comprimidas, expandidas, sesgadas, deformadas, etc).

En la primera gráfica (Rápido y Furioso) no se presentan grandes rasgos distintivos pues una de las funciones trata valores máximos a la otra función en todo el alcance de la variable, Algo similar ocurre en la segunda gráfica. En cambio en la última gráfica se construye una gráfica compuesta por ambas gráficas base y se obtiene una deformación que a simple vista cuesta comprender se trata de una xor.

4. **Dibujar las modificaciones que sufren las funciones de pertenencia dadas, al aplicar sucesivamente los adverbios fuzzy que se indican:**
   a.

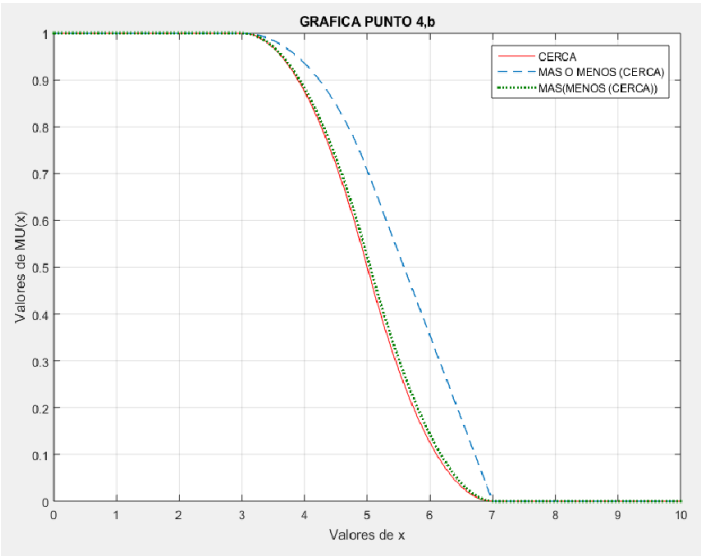
```
x = 10:0.1:50;
y = base(x);

MAS = y.^1.25;
INT = intens(x,MAS);
CON = y.^2;
CERCANO = y.^1.4;
LIGERAMENTE_CERCANO = max(INT,CON);

plot(x,y, 'Color', 'g');
hold on;
plot(x,CERCANO, 'Color', 'b','LineStyle', '--', 'LineWidth',
2);
plot(x,LIGERAMENTE_CERCANO, 'Color', 'r', 'LineStyle', ':');
```



https://github.com/matness-university/U2-fuzzy-logic/blob/main/4_Adverbios_fuzzy/ligeramente_cercano.m

b.

```
>> %FUNCION MAS O MENOS CERCA%
>> x=[0:0.1:10];
>> MU=zmf(x,[3,7]);
>> ylim([-0.05 1.05]);
>> xlabel('Valores de x')
>> ylabel('Valores de MU(x)')
>> plot(x,MU,'-r')
>> MU1=MU.^(1/2);    %MAS O MENOS (CERCA)%
>> hold on
>> plot(x,MU1,'--','linewidth',1);
>> MU2=((MU.^(3/4)).^(5/4));       %MAS(MENOS(CERCA))%
>> hold on
>> plot(x,MU2,'-b','linewidth',1);
>> legend('CERCA','MAS O MENOS (CERCA)','MAS(MENOS (CERCA))')
>> title('GRAFICA PUNTO 4,b')
>> grid;
>>
>> xlabel('Valores de x')
>> ylabel('Valores de MU(x)')
>>
```



Calcular u(x) de las figuras, utilizando por lo menos 100 puntos. Comentar si las heurísticas aplicadas producen modificaciones coherentes, analizando algunos puntos de referencia (por lo menos dos).

En el apartado a de este ejercicio, se encuentran las funciones cercano y ligeramente cercano. La primera denota un gráfico acorde a lo esperado, pues a medida que nos acercamos al valor crítico la pertenencia crece abruptamente gracias a la curva pronunciada de esta propiedad. Lo mismo sucede con la curva correspondiente a ligeramente cercano, en el cual se espera que ésta alcance valores máximos cuando se encuentra próxima, lo cual se sucede en las mitades de los costados de las gráficas.

En el segundo apartado (b) se obtiene, como es de esperarse, que las curvas correspondientes a “Mas o menos” y “Mas(Menos)” son casi idénticas. Sobre todo la curva “Mas(Menos Cerca)” que es muy aproximada a la función “Cerca” original.

5. **Función para cálculo de adverbios múltiples. Escribir una función de Matlab que aplique sucesivamente hasta tres adverbios fuzzy a una secuencia de pertenencia base mu0, con las siguientes condiciones:**
   a. **La función debe recibir como entrada una secuencia [x0 , mu0].**
   b. **Debe incorporar adicionalmente una, dos o tres etiquetas que indiquen el tipo de adverbio a aplicar. En la función se debe reconocer la cantidad de adverbios a aplicar (comando nargin). Las etiquetas pueden ser por ejemplo ‘no’, ‘muy’, ‘algo’, ‘aproximadamente’, etc.**
   c. **Incorporar controles para asegurar que x0 y mu0 sean de la misma longitud y que los valores de mu0 estén en el intervalo [0;1].**
   d. **Incorporar un mínimo de cinco adverbios reconocibles por la función (se pueden utilizar los propuestos en clase). Puede utilizar internamente las funciones definidas en el punto anterior como funciones auxiliares (deben estar en la misma carpeta).**
   e. **Como argumentos de salida, la función debe generar la secuencia de base x0 (la misma que ingresa), la secuencia de pertenencia mu0 (la misma que ingresa) y la secuencia de pertenencia modificada -muadv- por la aplicación sucesiva de los adverbios indicados en el argumento de la función. Por ejemplo:**

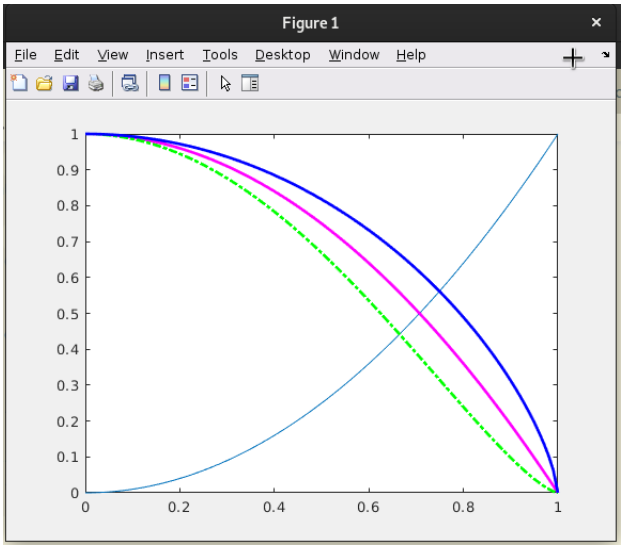
      [x0,mu0,muadv] = adv_multiples(x0,mu0,’no’,’muy’,’aproximadamente’)

   f. **La función debe generar además una gráfica que muestre el conjunto de base y los conjuntos modificados con diferentes colores, sobre el mismo sistema de ejes.**

```
function [x0, mu0, muadv] = multiples(x0, mu0, varargin)

   ADVERBS = ['no', 'muy', 'con', 'cerca', 'dilac', 'aproximadamente'];

   if(max(mu0) > 1)
```

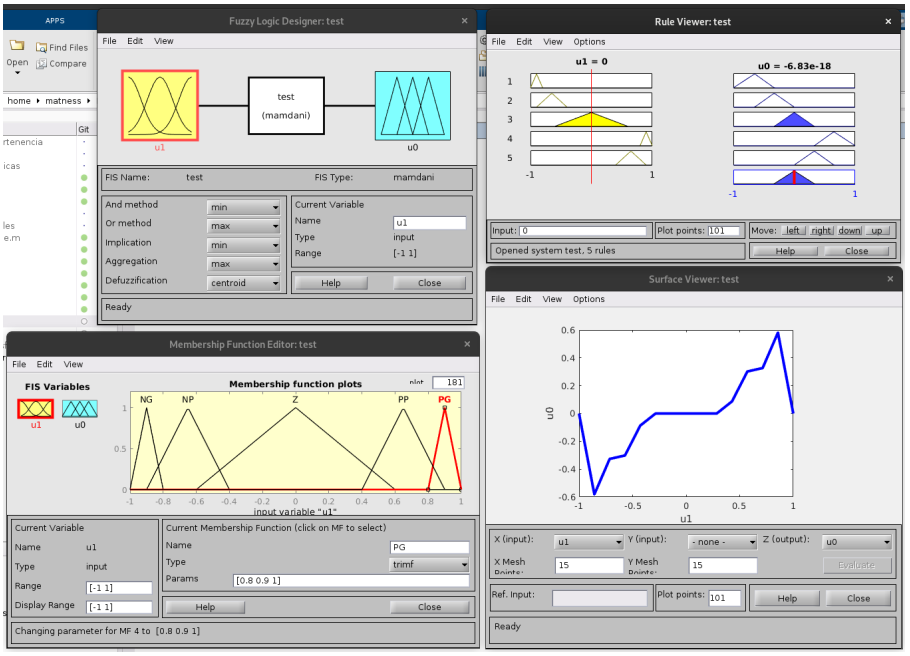
```
        error('mu0 no puede contener valores mayores a 1');
    end

    if(length(x0) ~= length(mu0))
        error('Los vectores deben tener las mismas dimensiones');
    end

    if(length(varargin) > 3)
        error('Sólo se pueden recibir hasta 3 adverbios');
    end

    for i=1:length(varargin)
        if(~all(ismember(varargin{i},ADVERBS)==1))
                error('Sólo se pueden usar uno de estos adverbios no, muy, algo, cerca,
aproximadamente')
        end
    end

    AUX = mu0;

    plot(x0,mu0);
    hold on;

    for i=1:length(varargin)
        switch varargin{i}
            case 'no'
                AUX = no(AUX);
                plot(x0,AUX, 'Color', 'm','LineStyle', '-', 'LineWidth', 2);
            case 'muy'
                AUX = muy(AUX);
                plot(x0,AUX, 'Color', 'c','LineStyle', '--', 'LineWidth', 2);
            case 'con'
                AUX = con(AUX);
                plot(x0,AUX, 'Color', 'r','LineStyle', ':', 'LineWidth', 2);
            case 'cerca'
                AUX = cerca(AUX);
                plot(x0,AUX, 'Color', 'g','LineStyle', '-.', 'LineWidth', 2);
            case 'dilac'
                AUX = dilac(AUX);
                plot(x0,AUX, 'Color', 'b','LineStyle', '-', 'LineWidth', 2);
            case 'aproximadamente'
                AUX = dilac(AUX);
                plot(x0,AUX, 'Color', 'k','LineStyle', '--', 'LineWidth', 2);
        end
    end

    muadv = AUX;

end
```

```
x = 0:0.01:1;
y = x.^2
[x0,mu0,muadv] = multiples(x, y, 'no', 'cerca', 'dilac')
```



https://github.com/matness-university/U2-fuzzy-logic/blob/main/5_Adverbios_multiples/multiples.m

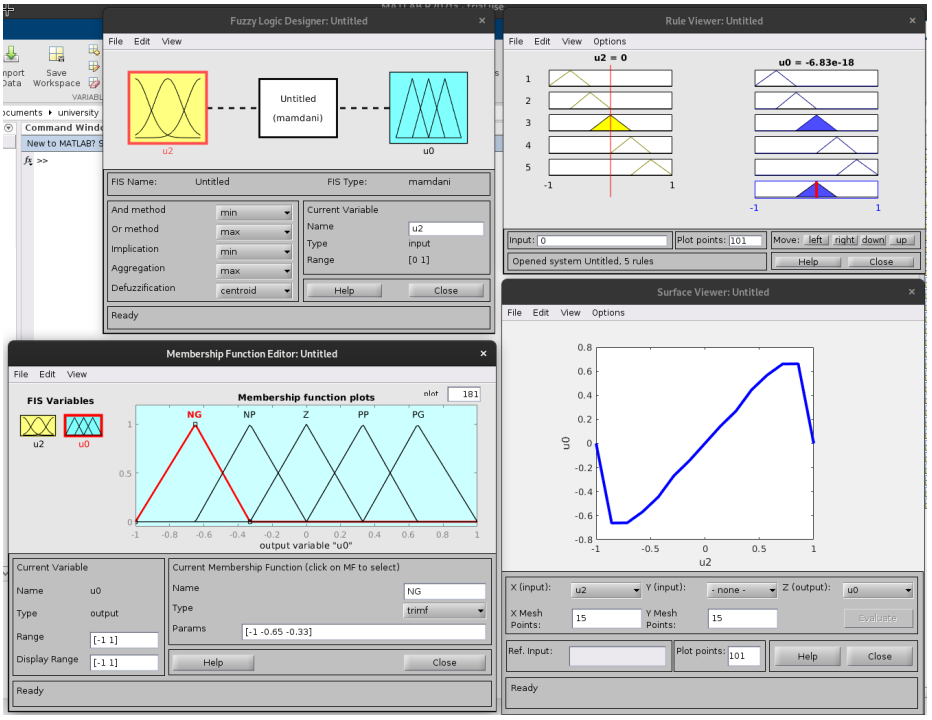
6. **Función de transferencia fuzzy. Para cada uno de los conjuntos fuzzy de entrada (1 , 2 , 3), y considerando que el conjunto fuzzy de la derecha ( ) es de salida, y con la asistencia de la GUI fuzzy, mostrar cómo resultan las funciones de transferencia. Considerar la implicación de Mamdani, defuzzyficación por centroide y el trazado de la curva de inferencia. Se debe generar una base de reglas que relacionen entrada y salida.**
    a. **u1**



En este caso encontramos pendientes muy pronunciadas en los extremos de la gráfica, esto indica que ante pequeños cambios en nuestras entradas, las salidas cambian abruptamente.

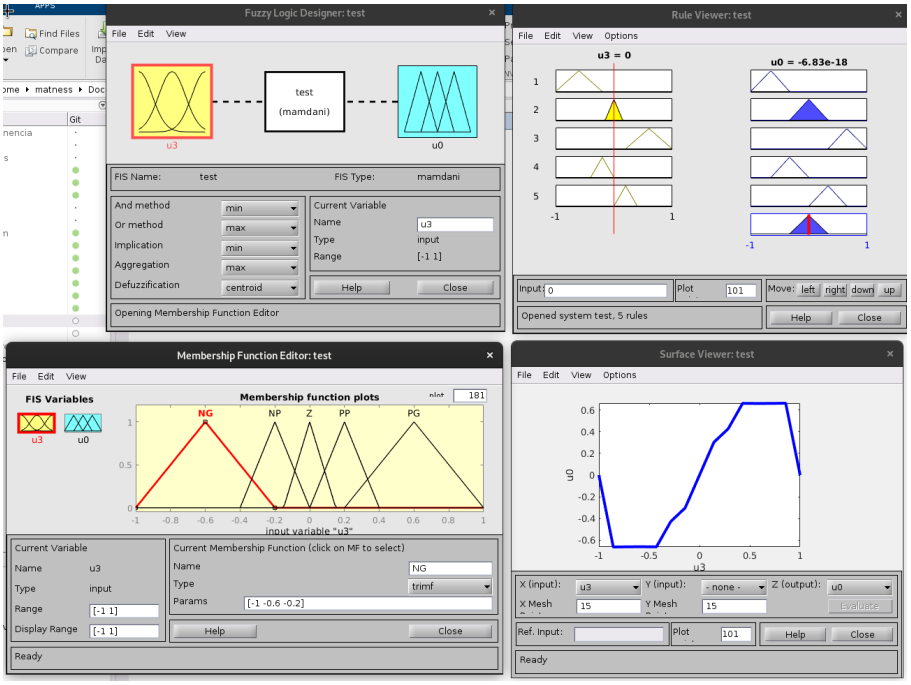
La gráfica también presenta una meseta en la cual no se producirán cambios para valores de entrada entre -0.4;0.4 aproximadamente.
Es necesario un rediseño pues en ningún punto de la gráfica se alcanzan los extremos superior e inferior.

b. u2



En este caso también encontramos pendientes muy pronunciadas en los extremos de la gráfica, esto indica que ante pequeños cambios en nuestras entradas, las salidas cambian abruptamente.
La gráfica también presenta dos mesetas en la cual no se producirán cambios para distintos valores de entrada.
También es necesario un rediseño pues en ningún punto de la gráfica se alcanzan los extremos superior e inferior.

c. u3



Esta gráfica presenta 3 pendientes muy pronunciadas en los extremos de la gráfica, esto indica que ante pequeños cambios en nuestras entradas, las salidas cambian abruptamente.
La gráfica también presenta dos mesetas en la cual no se producirán cambios para valores de entrada.
Es necesario también un rediseño pues en ningún punto de la gráfica se alcanzan los extremos superior e inferior.

https://github.com/matness-university/U2-fuzzy-logic/tree/main/6_Funcion_de_transferencia