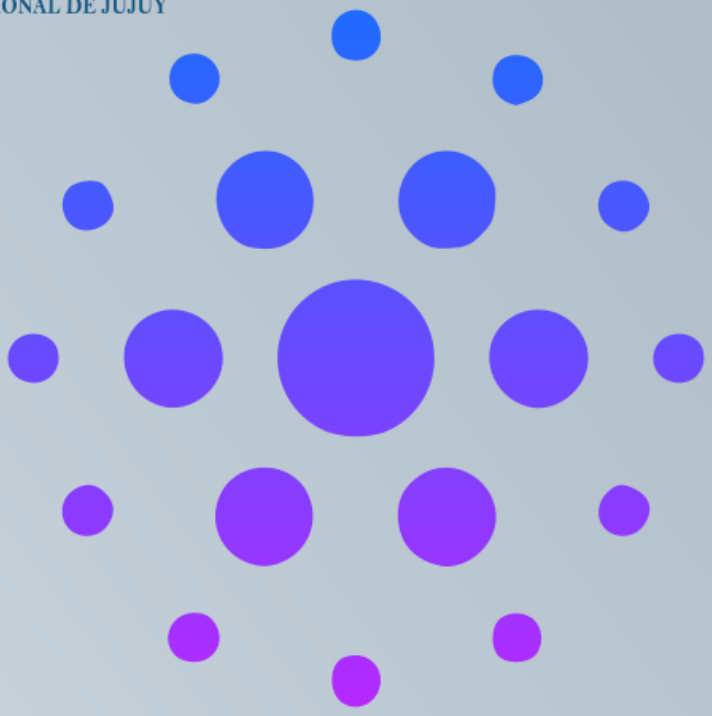
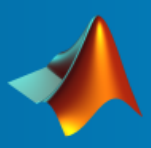
FACULTAD DE
INGENIERIA
UNIVERSIDAD NACIONAL DE JUJUY



INTELIGENCIA ARTIFICIAL

LÓGICA FUZZY

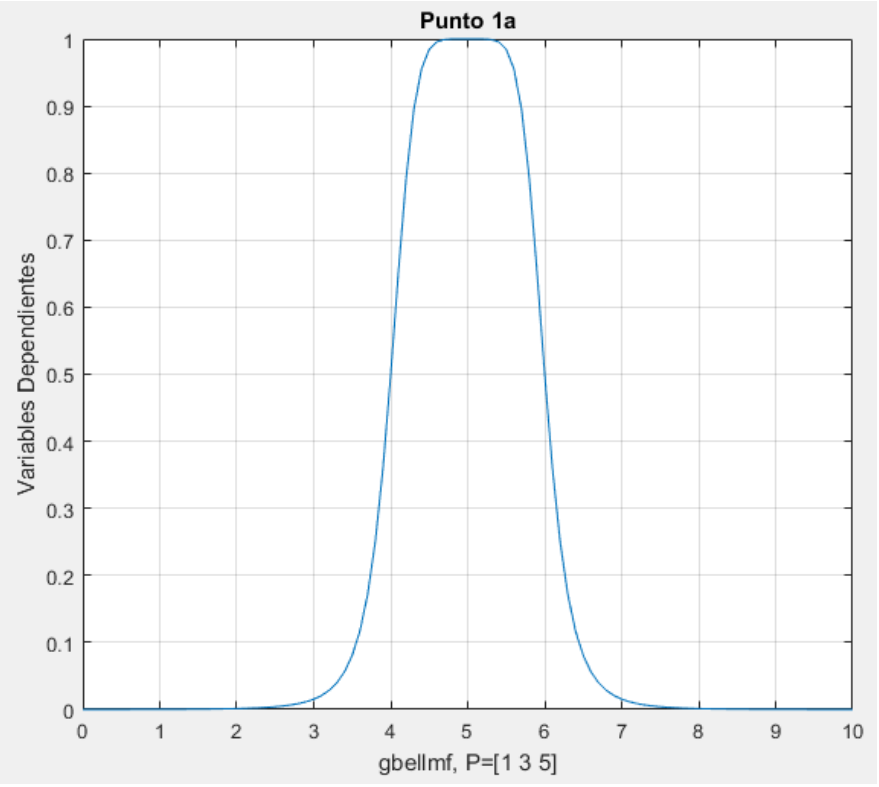
Heredia, Leoonardo Antonio
Madrid, Juan Sebastian
Patino. Judith Graciela



1. **Investigación de funciones.** Para las funciones de Matlab dadas a continuación, buscarlas, interpretarlas, correr una prueba simple y escribir un resumen de su actividad.

   a) **evalmf()**
   Su formato es **y = evalmf(x, mfParams, mfType)**; donde **evalmf** es una función que puede calcular cualquier función de membresía, donde "**x**" es el dominio variable, "**mfType**" es una función de membresía proporcionada por la toolbox y "**mfParams**" es el parámetro correspondiente de esta función de membresía.

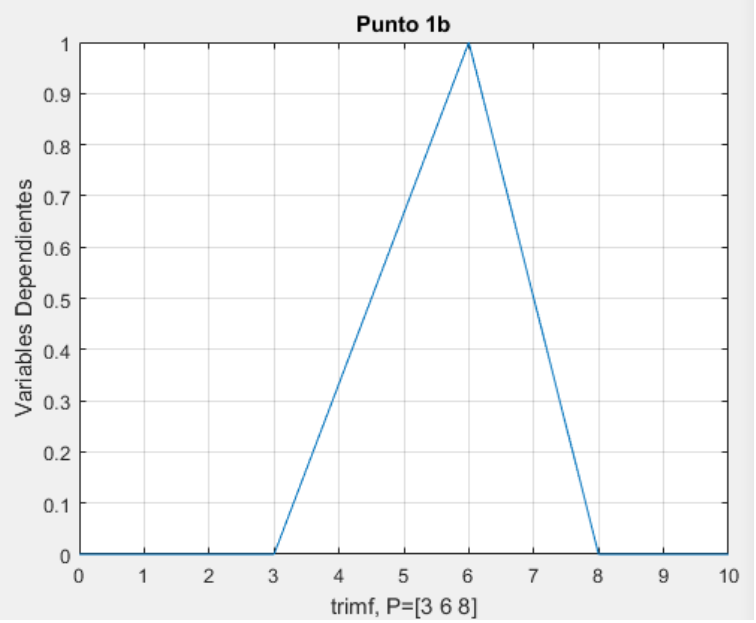
```
clc, clear;
x = 0:0.1:10;
parametros = [1 3 5];
tipo = 'gbellmf';       %Funcion de pertenencia a la curva de campana
%generalizada, devuelve una matriz que es una campana generalizada.
y=evalmf(x,parametros,tipo);
plot(x,y)
xlabel('gbellmf, P=[1 3 5]')
ylabel('Variables Dependientes')

grid on
```



   b) **trimf()**
   Su formato es **y = trimf(x, mfParams)**; donde **trimf** es una función que permite construir funciones triangulares más rápido. En sus parámetros recibe la variable "x" que es un vector que determina el dominio de definición y "mfParams" que son los parámetros que definen la forma

curva. Si se busca dibujar un triángulo abierto lo que se hace es repetir los 2 primeros valores si es hacia la derecha o sino los 2 últimos valores si se busca dibujar hacia la izquierda.

```
1 -    clc, clear;
2 -    x = 0:0.1:10;
3
4 -    parametros = [3 6 8];
5 -    y = trimf(x,parametros);
6 -    plot(x,y)
7
8 -    xlabel('trimf, P=[3 6 8]')
9 -    ylabel('Variables Dependientes')
10 -   title('Punto 1b')
11 -   grid on
12
```

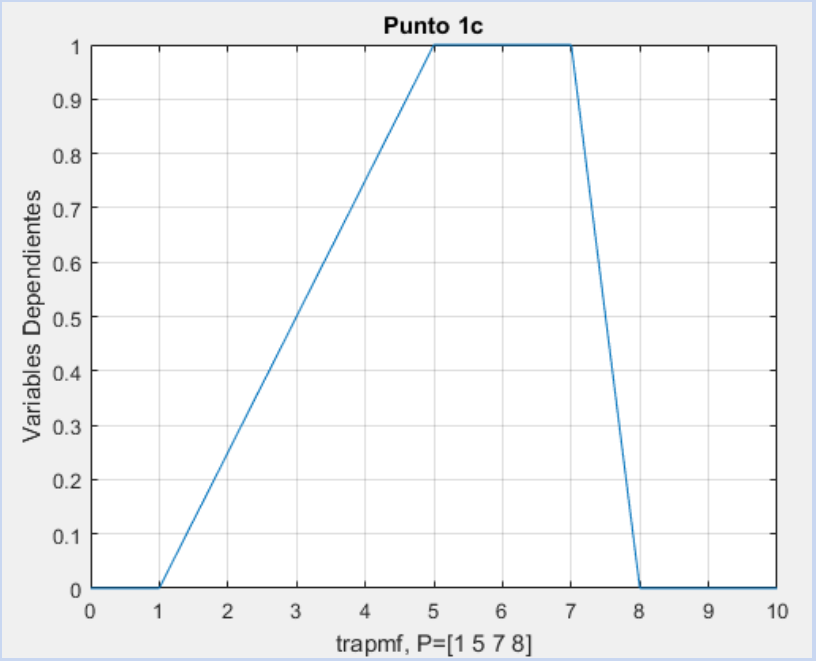


c) **trapmf()**

Su formato es **y=trapmf(x,[a b c d])**, es una función que dibuja un trapezoide tiene un núcleo más amplio que está dado por un intervalo cuando en la función triangular es solo un punto. Dicho intervalo marca los valores en ascenso entre **a b**, el núcleo entre **b c** y valores en descenso entre **c d**.

Si se desea dibujar una función abierta por izquierda se deben repetir los últimos valores del intervalo y si se desea dibujar abierta hacia la derecha se deben repetir los primeros 2 valores del intervalo.

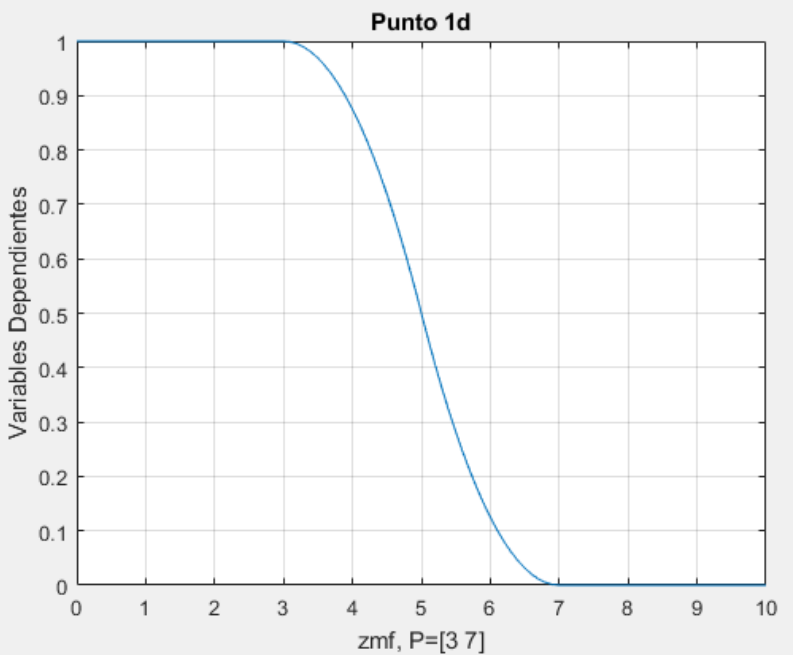
```
clc, clear;
x = 0:0.1:10;

parametros = [1 5 7 8];
y = trapmf(x,parametros);
plot(x,y)

xlabel('trapmf, P=[1 5 7 8]')
ylabel('Variables Dependientes')
title('Punto 1c')
grid on

```



d)**zmf()**

Su forma es **y = zmf(x,[a b])** que devuelve una matriz que tiene forma de Z donde '**x**' es la variable independiente y el intervalo '**[a b]**' son parámetros para determinar la forma de la curva. El parámetro '**a**' define el hombro de la función de pertenencia y '**b**' define su pie.

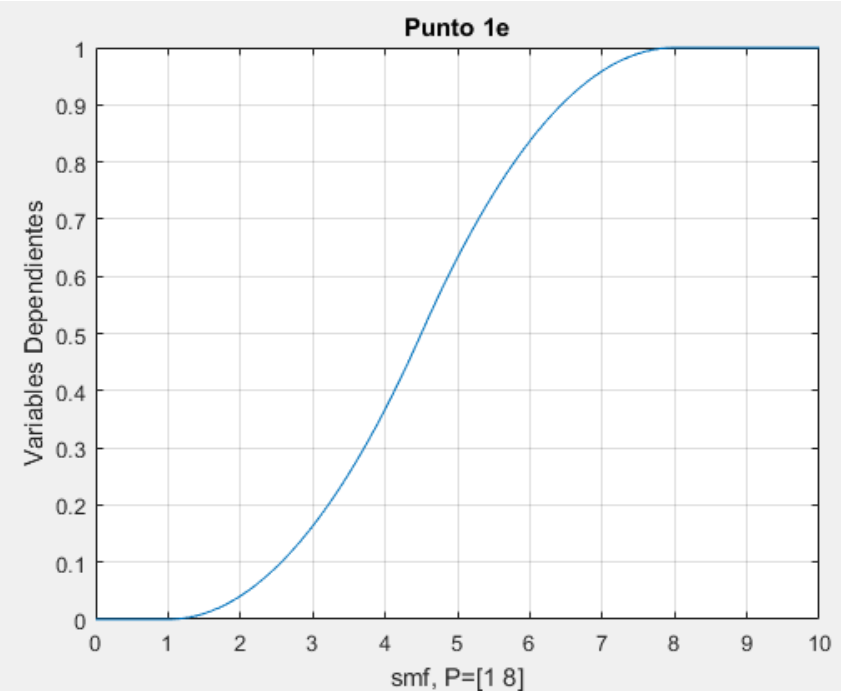
```
clc, clear;
x = 0:0.1:10;

parametros = [3 7];
y = zmf(x,parametros);
plot(x,y)

xlabel('zmf, P=[3 7]')
ylabel('Variables Dependientes')
title('Punto 1d')
grid on

```



e) **smf()**

Su forma **y = smf(x,[a b])** que devuelve una matriz que tiene forma de S, donde los valores de '**x**' son de entrada para los cuales se calculan valores de pertenencia expresados como escalares o vectores. El intervalo '**a b**' son parámetros de la función de pertenencia, donde '**a**' define el pie y '**b**' define su hombro.

```
clc, clear;
x = 0:0.1:10;

parametros = [1 8];
y = smf(x,parametros);
plot(x,y)

xlabel('smf, P=[1 8]')
ylabel('Variables Dependientes')
title('Punto 1e')
grid on

```



f) **mse()**

Su forma **y = mse(M,V,P)**, es una función de rendimiento del error cuadrático medio. Mide el rendimiento de la red según la media de los errores cuadráticos. Los valores de '**M**' es la matriz de errores, los valores de '**X**' son el vector de todos los valores de peso y sesgo y '**P**' son los parámetros de rendimiento.

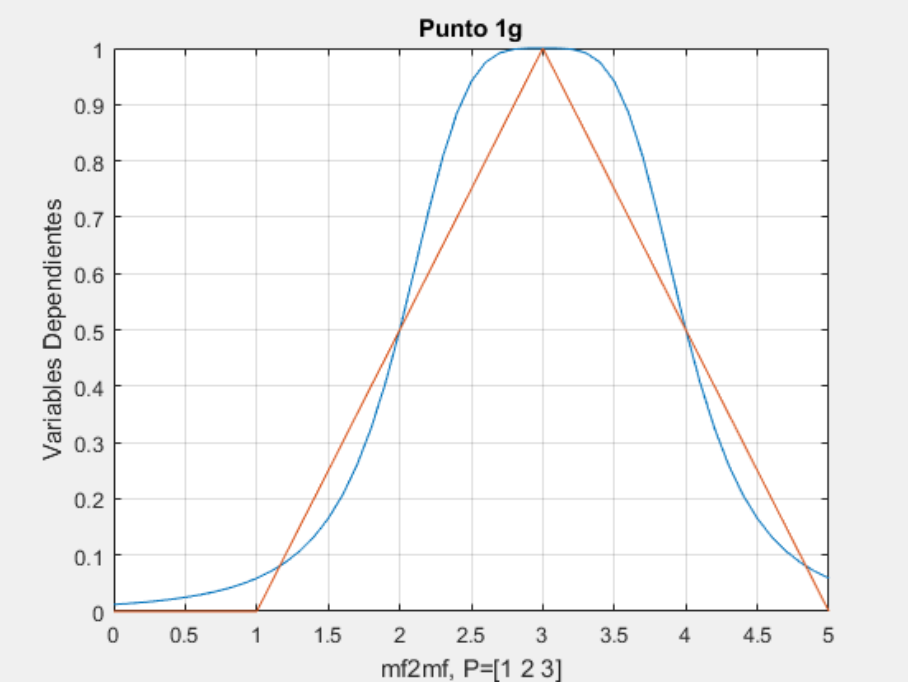
```
%Dado dos matrices que muestran los valores reales y los valores
%pronosticados. Se utuliza la funcion 'mse' para calcular el error
%cuadratico medio entre las dos matrices.

clc, clear;
actual = [34 37 44 47 48 48 46 43 32 27 26 24];
predicho = [37 40 46 44 46 50 45 44 34 30 22 23];

%Calcular MSE entre los valores actuales y los predichos.

respuesta = mse(actual,predicho)

```

```
respuesta =

    5.9167

fx >>
```

g) **mf2mf()**

Su forma **y = mf2mf(entradaParametros,entradaTipo,salidaTipo)**, esta función convierte los parámetros de un tipo de función de pertenencia en los de otra función de pertenencia. Esta traducción da como resultado la pérdida de información, de modo que si los parámetros de salida se vuelven a traducir al tipo de función original, esta función no tiene el mismo aspecto que la función original.

```
1  -    clc, clear;
2
3  -    x=0:0.1:5;
4  -    mfp1 = [1 2 3];
5  -    mfp2 = mf2mf(mfp1,'gbellmf','trimf');
6  -    plot(x,gbellmf(x,mfp1),x,trimf(x,mfp2))
7
8  -    xlabel('mf2mf, P=[1 2 3]')
9  -    ylabel('Variables Dependientes')
10 -    title('Punto 1g')
11 -    grid on
```

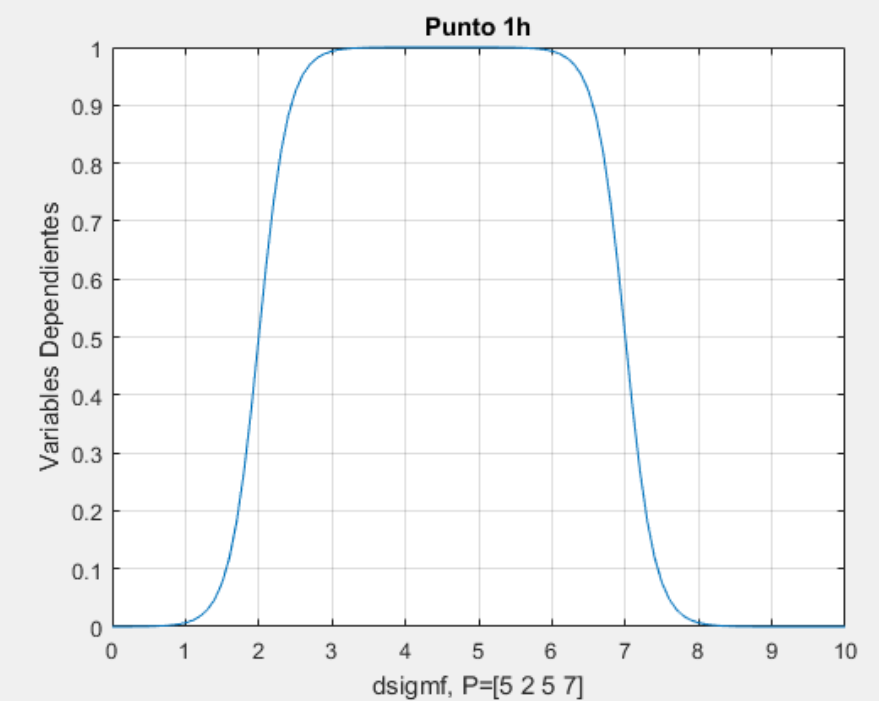


h) **dsigmf()**

Su forma **y = dsigmf(x,params)** función que calcula la diferencia entre 2 funciones de pertenencia sigmoidal. Donde los valores de '**x**' es un escalar o un vector para el cual se calculan los valores de pertenencia

que pueden ser un escalar o un vector. El '**params**' es un vector de parámetros de la función que puede estar dado por los valores **[a1 c1 a2 c2]**, donde **a1 c1** son los parámetros de la primera función sigmoidal y **a2 c2** son los parámetros de la segunda función sigmoidal

```
1 -    clc, clear;
2
3 -    x=0:0.1:10;
4 -    params = [5 2 5 7];
5 -    y = dsigmf(x,params);
6
7 -    plot(x,y)
8
9 -    xlabel('dsigmf, P=[5 2 5 7]')
10 -   ylabel('Variables Dependientes')
11 -   title('Punto 1h')
12 -   grid on
```



i) **rmvar()**

Su forma **outfis = rmvar(infis,varType,varIndex)**, esta función elimina la variable de entrada o de salida con el índice especificado del sistema difuso pasado por parámetro. Donde '**infis**' es el sistema difuso especificado como objeto FIS, '**varType**' es un tipo de variable especificado como input u output, '**varIndex**' es un número entero que trabaja como índice y '**outfis**' es el sistema difuso devuelto como objeto FIS.

j) **rmmf()**

Su forma **outfis = rmmf(infis,varType,varIndex,'mf',mfIndex)** elimina la funcion de pertenencia especificada de una variable de entrada o salida dada del sistema difuso infis. El sistema difuso está representado por el objeto "infis", el "varType" es un tipo de variable especificado como input o output, la "varIndex" es un número positivo que se utiliza como índice, "mfIndex" es un número positivo que se utiliza como índice de la función de pertenencia y "outfis" es el sistema difuso actualizado que devuelve un objeto.

k) **plotmf()**

Una de sus formas **plotmf(fis,variableType,variableIndex)**, esta función dibuja funciones de membresía para variables de entrada o salida en el sistema de inferencia difusa FIS.

2. Sustitución de funciones

a) Conceptos teóricos:

a1) Indicar ventajas y desventajas de utilizar funciones de pertenencia

lineales o no lineales.

Si consideramos que las funciones lineales (triangular, hombro, trapezoidal,etc) no son continuas debido a algún punto de corte, también podemos observar que esto las hace discontinuas.

En cambio las funciones no lineales (gaussiana, sigmoidal, pi, etc) son continuas y derivables en todos sus puntos.

Destacando estas características de cada función podemos mencionar como ventaja, que las funciones no lineales ofrecen un mejor resultado al resolver problemas en el diseño de controladores ya que los mismos tienen una mejor adaptación a los puntos de análisis en cuestión. En resumen, los controladores de tipo no lineal son más eficientes y efectivos.

a2) Investigar y mencionar por lo menos 2 funciones de pertenencia no convencionales.

Las funciones no convencionales

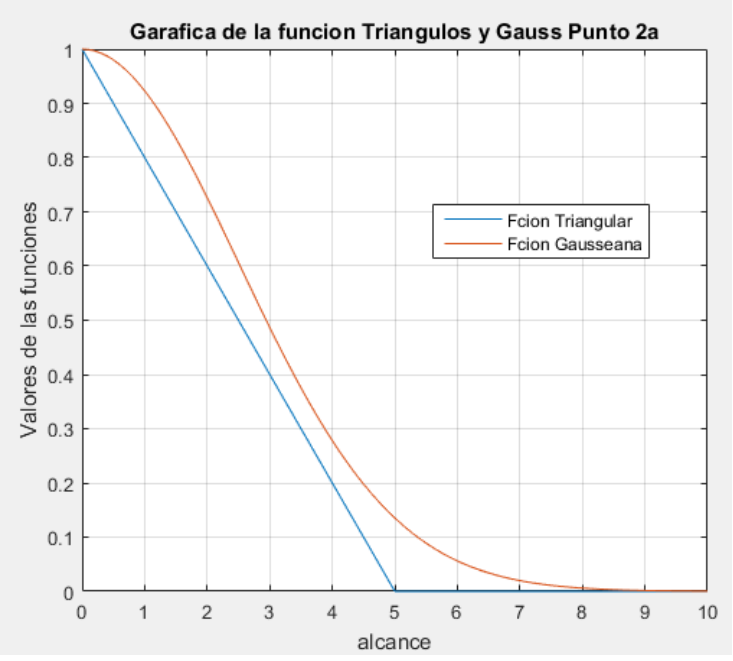
b) Un sistema de lógica fuzzy debe reemplazar dos funciones lineales por tramos, con funciones continuas y derivables, de

manera que el error cuadrático medio sea menor a 10-2. Calcular los parámetros en cada caso, para las sustituciones que se indican:

1º función triángulo rectángulo izquierdo (trimf): alcance (0, 10), con parámetros (0,0,5). Reemplazar con "media" función gaussiana o sigmoide.

```
%Funcion lineal TRIANGULO - Funcion continua Gaussiana
%Alcance
clc,clear;
x=[0:0.1:10];
%funcion triangulo con las tres particiones lineales
y=trimf(x,[0 0 5]);
%dibujamos el triangulo abierto por la izquierda
plot(x,y)
%la funcion, nos permite reutilizar el eje de coordenadas
hold on;
%calculamos los valores para las funciones continuas y derivables de Gauss
y1=gaussmf(x,[2.5 0]); %centro en cero y amplitud 2.5
%Dibujamos la funcion continua
plot(x,y1)
grid on
title('Garafica de la funcion Triangulos y Gauss Punto 2a')
xlabel('alcance')
ylabel('Valores de las funciones')
%Calculamos el error cuadratico medio, con dos funciones
err = immse(y,y1) %tambien calcula el error cuadratico medio
error= mse(y,y1)
%ambas dan el mismo error cuadratico medio menor a 10^-2
```

```
err =

    0.0062


error =

    0.0062
```

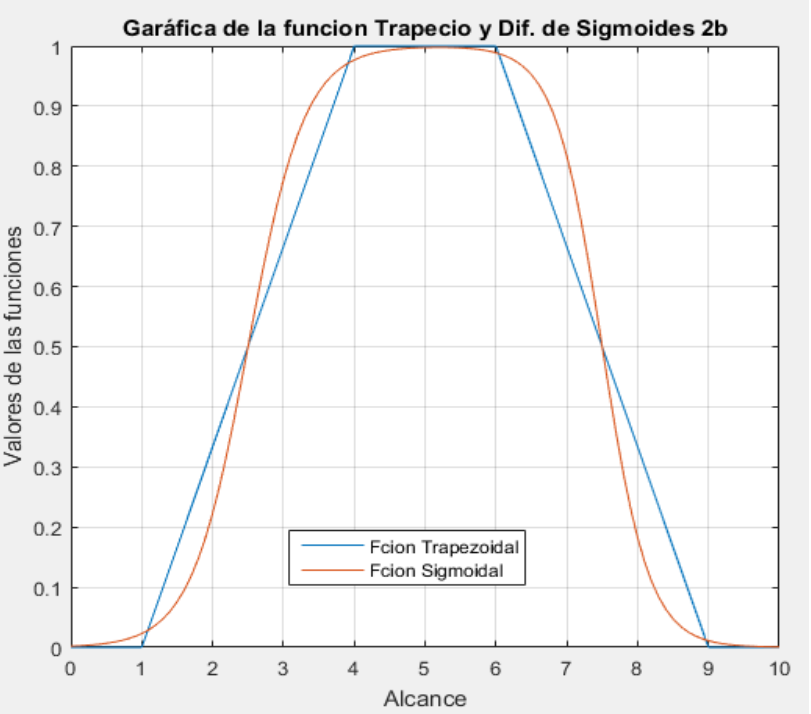


https://github.com/openjuy/ia2022/blob/main/tp02/02/punto2a.m

2º función trapecio: alcance (0, 10), trapmf() con parámetros (1,4,6,9). Reemplazar con función PI o diferencia de funciones sigmoides (dsigmf()), según convenga.

```
%Funcion lineal Trapecio - funcion lineal PI
clc, clear;
%Alcance
x=[0:0.1:10];
%funcion triangulo con las tres particiones lineales
y=trapmf(x,[1 4 6 9])
%dibujamos el triangulo abierto por la izquierda
plot(x,y)
grid
%la funcion, nos permite reutilizar el eje de coordenadas
hold on;
%calculamos los valores para la funcion continuas diferencia
%de funciones sigmoides (dsigmf())
yl=dsigmf(x,[2.5 2.5 3 7.5])
%Dibujamos la funcion continua
plot(x,yl)
grid on
title('Garáfica de la funcion Trapecio y Dif. de Sigmoides 2b')
xlabel('Alcance')
ylabel('Valores de las funciones')
%Calculamos el error cuadratico medio, con dos funciones
error= mse(y,yl)
%el error cuadratico es menor que 10^-2
```

```
error =

    0.0057

fx >>
```



https://github.com/openjuy/ia2022/blob/main/tp02/02/punto2b.m

Orientación 1: En Matlab el error cuadrático medio se calcula con la función mse(). ECM = Error Cuadrático Medio (en español); MSE = Mean Square Error (en inglés).

Orientación 2: Investigar la posible utilización de la aplicación gráfica cftool.

3. Definición de funciones de pertenencia.

a) Conceptos teóricos:

a1) Indicar ventajas y desventajas de definir una variable lingüística con pocas o muchas particiones.

Las variables lingüísticas son variables cuyos valores se pueden dar con palabras, de esta forma podemos modelar un problema e identificar las variables en las premisas.

La ventaja de poder definir varias particiones en una vble. lingüística es aplicar una mejora al modelo que nos permita disminuir un índice de error en la salida y aumentar el rendimiento.

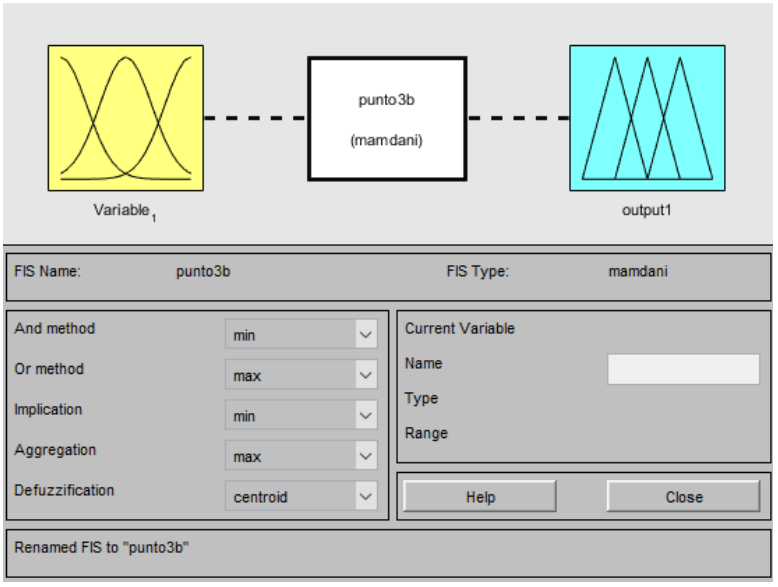
La desventaja es la complejidad con la que se debe diseñar el modelo para trabajar con estas variables.

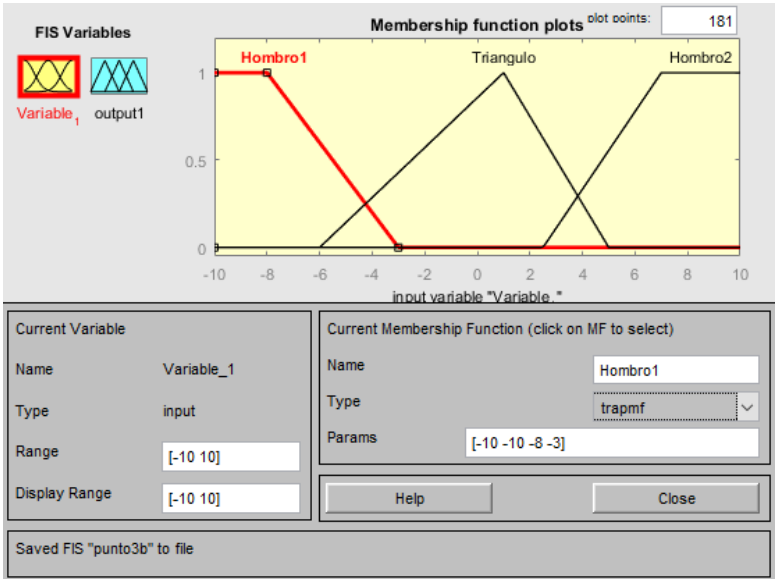
a2) ¿Qué características mínimas debe contener una función de pertenencia?

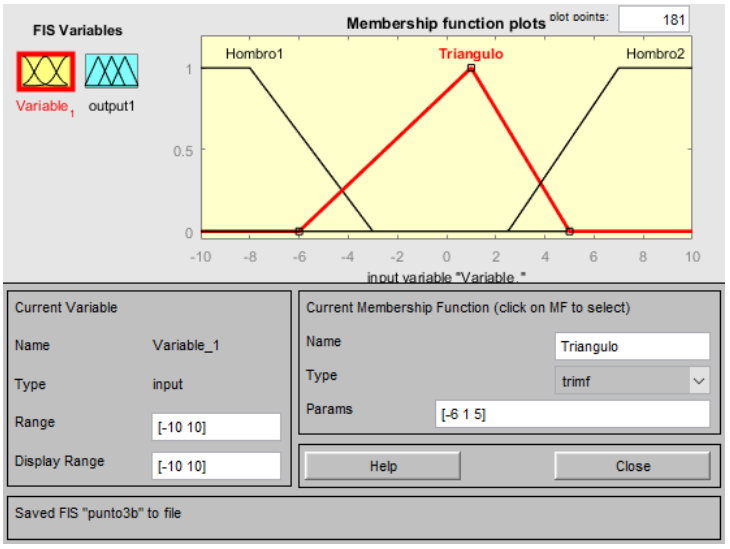
Las características mínimas que debe tener una función de membresía son: núcleo, límite y soporte.

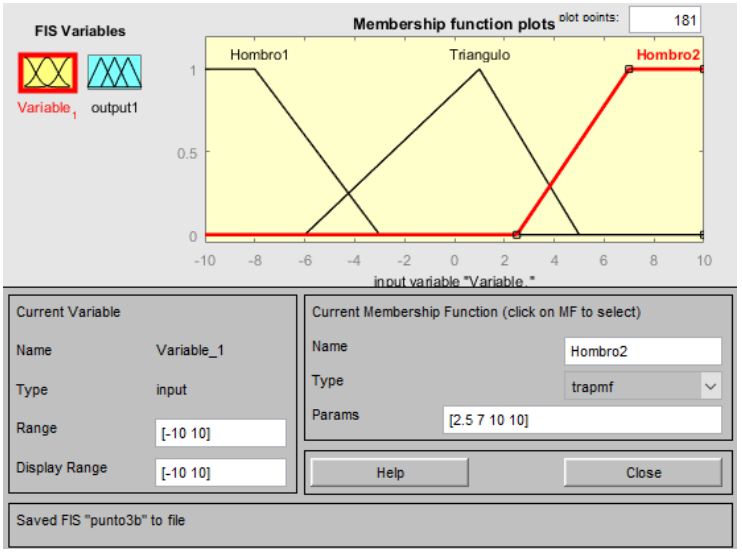
b) En forma manual o utilizando la interfaz gráfica de los sistemas de inferencia (GUI Fuzzy), definir por separado las variables que se indican. Adjuntar una gráfica:

RESOLVER EN CLASE 1º variable (V1): alcance (-10,+10). Tres particiones lineales hombro- triángulo-hombro con parámetros (-10, -8, -3), (-6, 1, 5), (2.5, 7, 10) respectivamente.



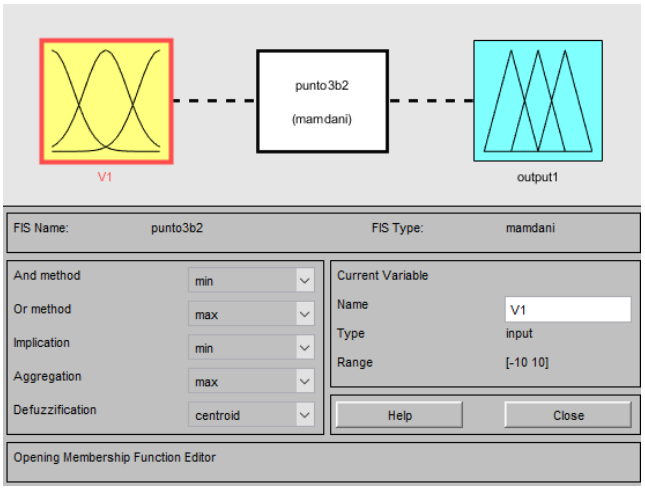


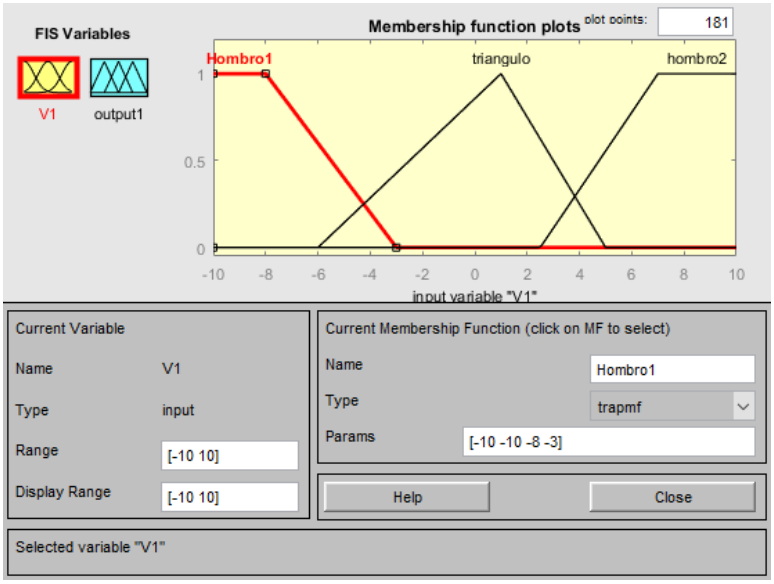


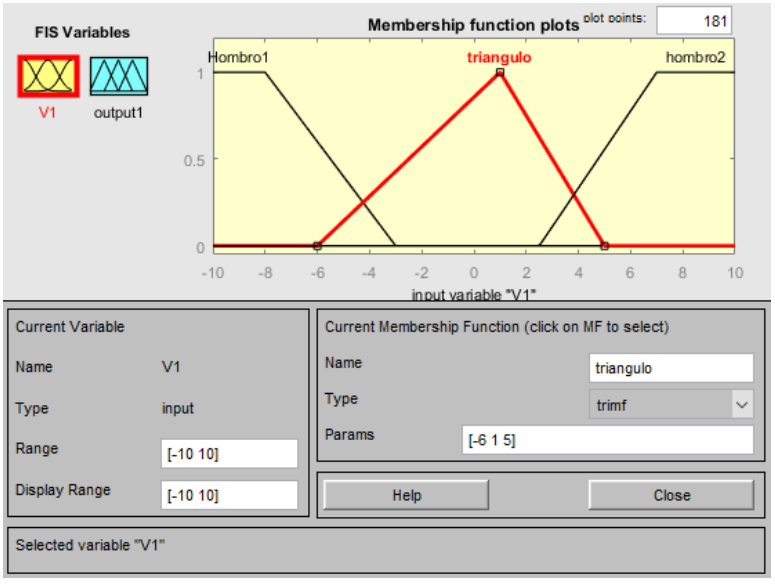


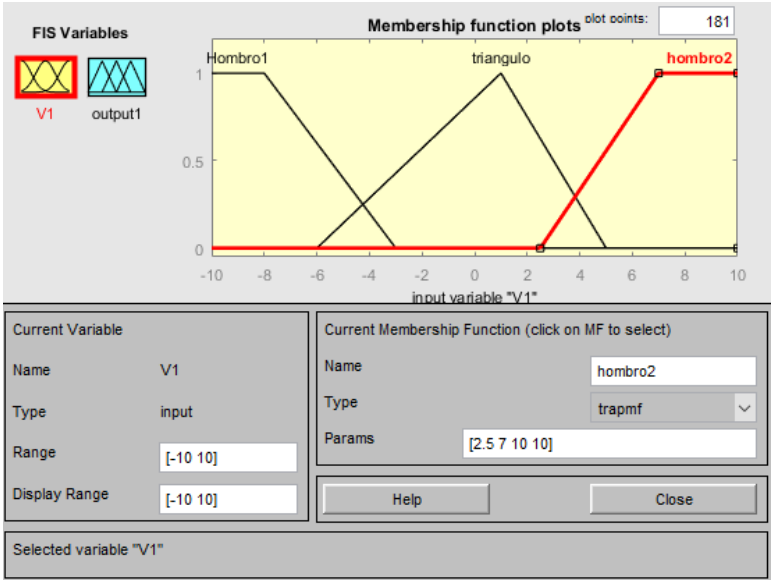
https://github.com/openjuy/ia2022/blob/main/tp02/03/punto3b1.fis

2º variable (V2): alcance (-20, 20). Cinco particiones tipo triángulo rectángulo izquierdo-trapecio-triángulo-trapecio-triángulo rectángulo derecho, igualmente espaciados con solapamiento del 30%.


punto3b2
(mamdani)
V1
output1
FIS Name: punto3b2
FIS Type: mamdani
And method min
Or method max
Implication min
Aggregation max
Defuzzification centroid
Current Variable
Name V1
Type input
Range [-10 10]
Help
Close
Opening Membership Function Editor

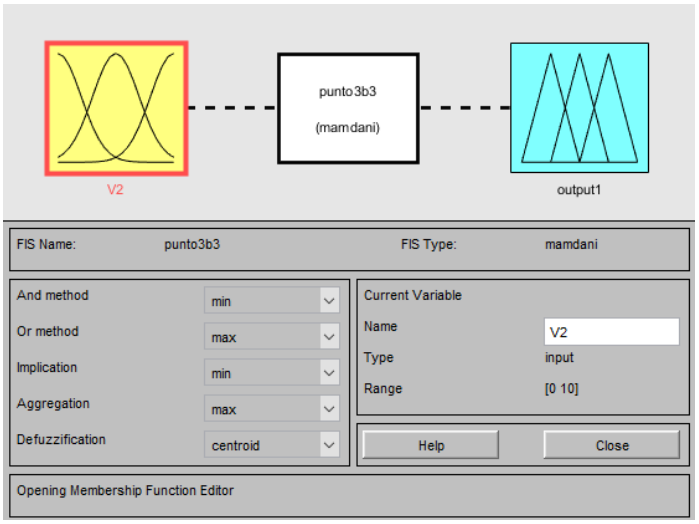

FIS Variables
V1
output1
Membership function plots
plot points: 181
Hombro1
triangulo
hombro2
input variable "V1"
Current Variable
Name V1
Type input
Range [-10 10]
Display Range [-10 10]
Current Membership Function (click on MF to select)
Name Hombro1
Type trapmf
Params [-10 -10 -8 -3]
Help
Close
Selected variable "V1"

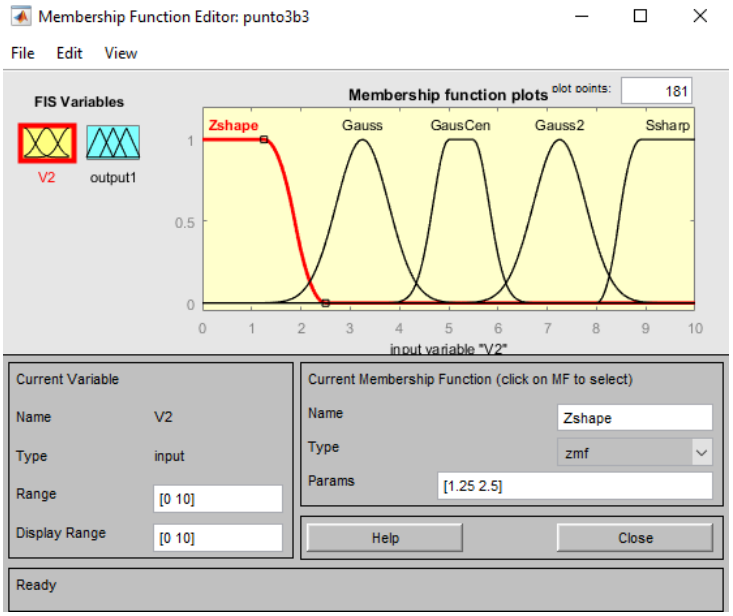

FIS Variables
V1
output1
Membership function plots
plot points: 181
Hombro1
triangulo
hombro2
input variable "V1"
Current Variable
Name V1
Type input
Range [-10 10]
Display Range [-10 10]
Current Membership Function (click on MF to select)
Name triangulo
Type trimf
Params [-6 1 5]
Help
Close
Selected variable "V1"

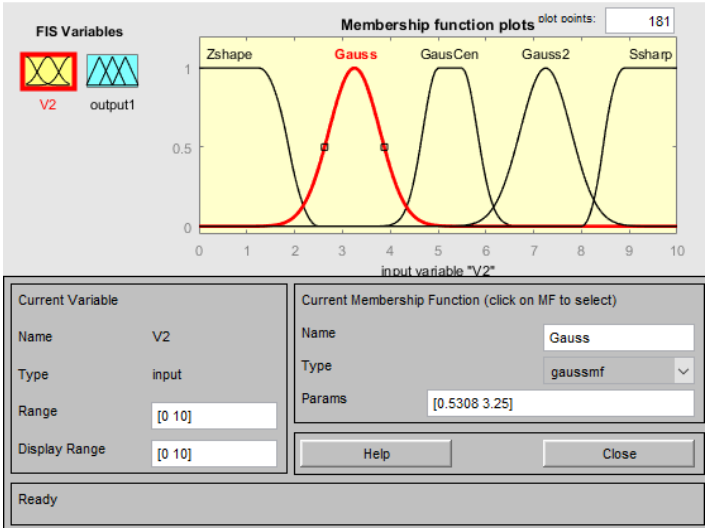


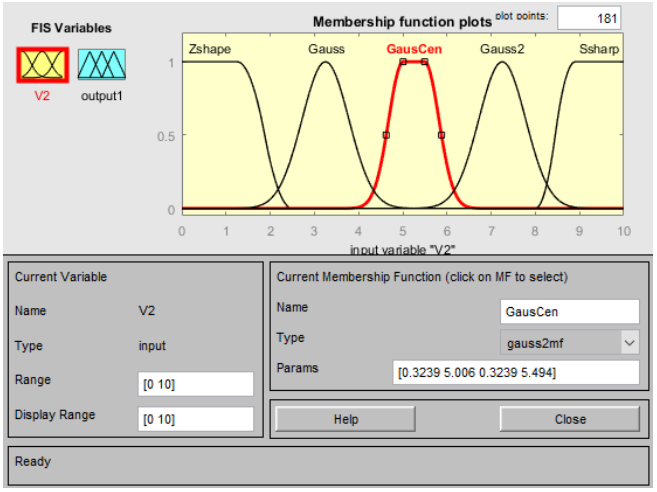
https://github.com/openjuy/ia2022/blob/main/tp02/03/punto3b2.fis

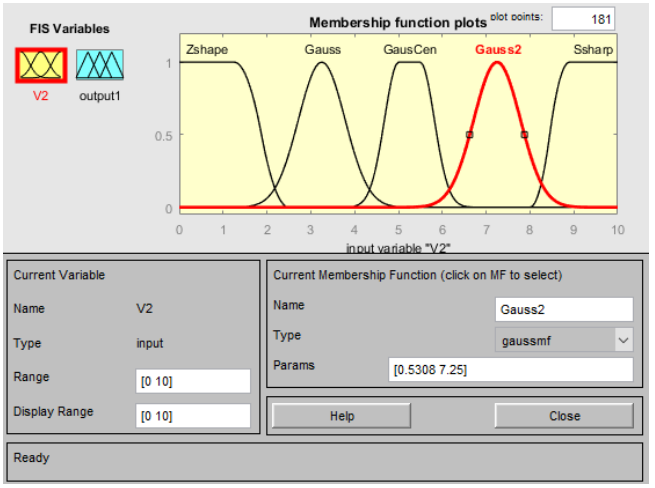
3º variable (V3): alcance (0, 10). Cinco particiones con funciones continuas y derivables Z shape (zmf()) - gaussiana - campana generalizada - gaussiana - S shape (smf()). Parámetros a definir por el usuario para una distribución simétrica y un solapamiento aproximado al 25%.

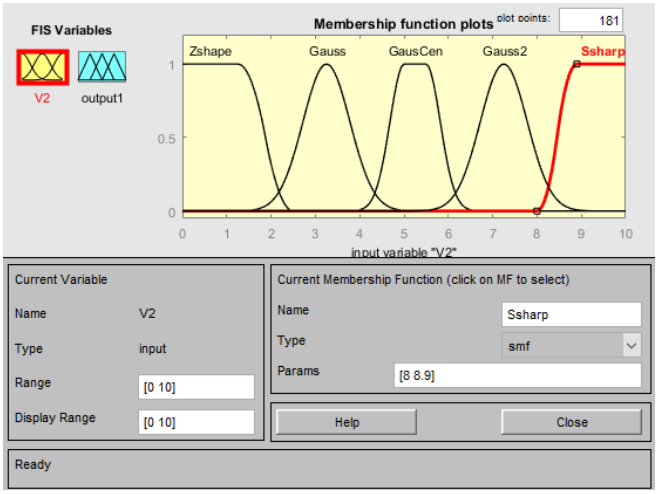

punto3b3
(mamdani)
V2
output1
FIS Name: punto3b3
FIS Type: mamdani
And method min
Or method max
Implication min
Aggregation max
Defuzzification centroid
Current Variable
Name V2
Type input
Range [0 10]
Help
Close
Opening Membership Function Editor


Membership Function Editor: punto3b3
File Edit View
FIS Variables
V2
output1
Membership function plots
plot points: 181
Zshape
Gauss
GausCen
Gauss2
Ssharp
input variable "V2"
Current Variable
Name V2
Type input
Range [0 10]
Display Range [0 10]
Current Membership Function (click on MF to select)
Name Zshape
Type zmf
Params [1.25 2.5]
Help
Close
Ready


FIS Variables
V2
output1
Membership function plots
plot points: 181
Zshape
Gauss
GausCen
Gauss2
Ssharp
input variable "V2"
Current Variable
Name V2
Type input
Range [0 10]
Display Range [0 10]
Current Membership Function (click on MF to select)
Name Gauss
Type gaussmf
Params [0.5308 3.25]
Help
Close
Ready







https://github.com/openjuy/ia2022/blob/main/tp02/03/punto3b3.fis

ver: Utilización de la GUI Fuzzy en PROBLEMAS RESUELTOS; Tutorial FIS. Manejo de la aplicación gráfica FIS en MANUALES Y TUTORIALES.

4. Funciones adverbiales básicas.

a) Conceptos teóricos:

a1) Explicar el concepto de adverbios fuzzy

Son operadores o modificadores que se asocian a las variables lingüísticas definidas en la premisa para expresar en forma más intensa o minimizar el significado de las mismas.

a2) ¿Qué efectos pueden causar a las funciones que se apliquen?

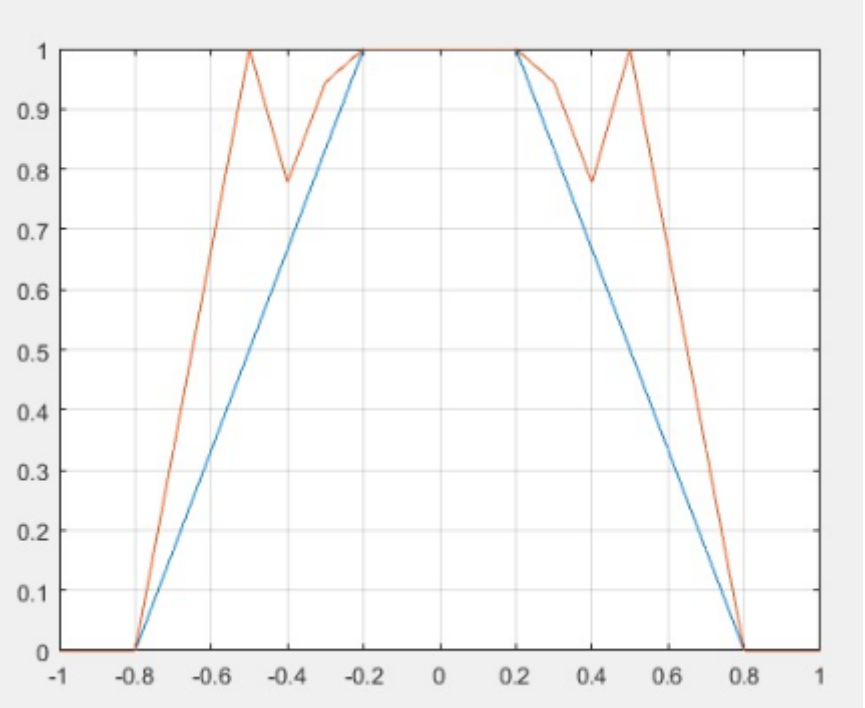
Como ya lo mencionamos la aplicación de estos modificadores pueden aumentar o disminuir el significado de las función a la que se le aplica.

b) Escribir las funciones en Matlab que apliquen las funciones adverbiales básicas que se indican.

RESOLVER EN CLASE

b3) Función intensificación f3 = intens(x,mu0) $f_3 = \begin{cases} 2*\mu_0 & si\ \mu_0 \in [0\ ;\ 0.5] \\ 1-2*(1-\mu_0)^2 & si\ \mu_0 \in [0.5\ ;\ 1] \end{cases}$

```
function [f3]= intens(x,mu0)
%b3) Función intensificación f3 = intens(x,mu0)
% f3= 2*mu0 si mu0 E [0; 0.5]
% f3= 1-2*(1-mu0)^2 si mu0 E (0.5; 1]
clc
[i,j]=size(x);
[k,l]=size(mu0);
aux=mu0;
if ((i==1)&&(k==1))
    if (j==1)
        for m=1:j
            if  aux(1,m)>=0 && aux(1,m)<=0.5
                aux(1,m)=2 * aux(1,m);
            else
                 aux(i,m)=1 - (2 * (1 - aux(i,m))^ 2);
            end
        end
    else
        fprintf('dimensiones distintas')
    end
else
      fprintf('distinto de vector fila')
end
 f3=aux;
end
```



https://github.com/openjuy/ia2022/blob/main/tp02/04/intens.m

Conclusión: La gráfica inicial es un trapecio que va desde -0.8 a +0.8 y tiene como núcleo -0.2 a +0.2. Al aplicarle el operador “intensificador” se obtiene una gráfica que se traza por el borde del trapecio asemejando su formato pero con un desplazamiento que tiene picos en -0.5 y +0.5 aproximadamente.

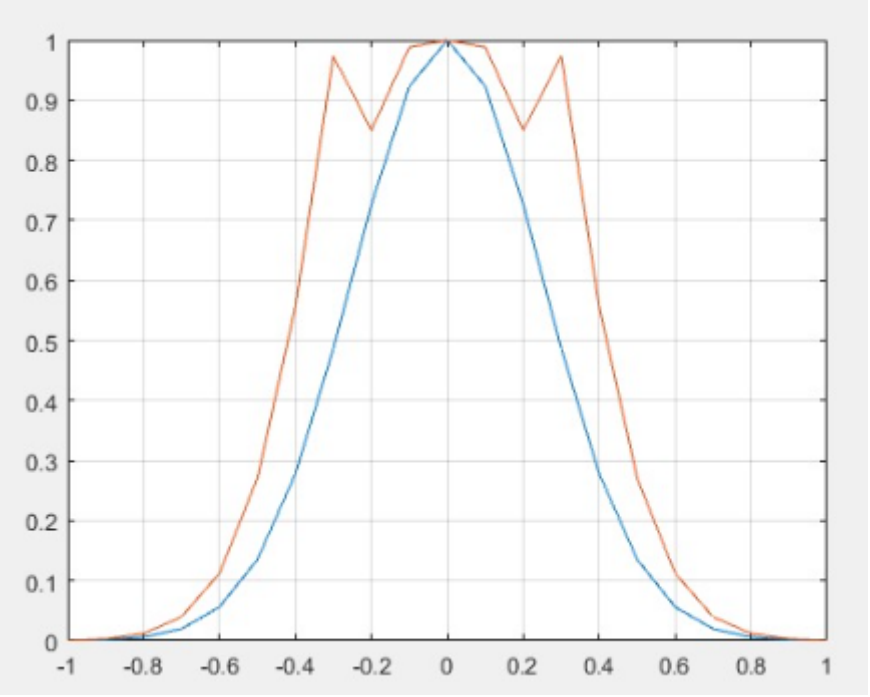
b4) Función difuminación f4 = difum(x,mu0)

$$f_3 = \begin{cases} \left(\frac{\mu_0}{2}\right)^{1/2} & \text{si } \mu_0 \in [0\,;\,0.5] \\ 1-\left(\frac{1-\mu_0}{2}\right)^{1/2} & \text{si } \mu_0 \in [0.5\,;\,1] \end{cases}$$

x = secuencia de datos de entrada en el eje de abscisas;

mu0 = valores de pertenencia correspondiente a los datos x.

```
function [f4]= difum(x,mu0)
%Función difuminación f4 = difum(x,mu0)
%f3 = (mu0/2)^(1/2) si mu0 E[0; 0.5]
%f3 = 1-((1-mu0)/2)^(1/2 si mu0 e (0.5; 1]
clc
[i,j]=size(x);
[k,l]=size(mu0);
aux=mu0;
if ((i==1)&&(k==1))  %controlar que sea un vector fila
    if (j==1) % que tengan la misma cantidad de columnas
        for m=1:j
            if  aux(1,m)>=0 && aux(1,m)<=0.5
                aux(1,m)= (aux(1,m)/2)^(0.5);
            else
                 aux(i,m)= 1-((1-aux(1,m)/2)^(0.5));
            end
        end
    else
        fprintf('dimensiones distintas')
    end
else
      fprintf('distinto de vector fila')
end
 f4=aux;
end
```



https://github.com/openjuy/ia2022/blob/main/tp02/04/difum.m

Conclusión: la función original es una gaussiana y al aplicar el operador difuminación nos da otra función que se traza por el borde con picos en -0.3 a +0.3, por fuera de la gráfica Gausseana.

c) En todos los casos implementar el control de datos para cada función, como por ejemplo que los argumentos sean vectores fila de igual longitud, que la función de pertenencia de entrada esté en el alcance correcto, que el recorrido del cálculo esté dentro de los límites, etc.

d) Aplicar y graficar superpuestas las funciones definidas en b) sobre un trapecio con alcance [-1,1] y formato [-0.8 , -0.2 , 0.2 , 0.8]. Igualmente hacerlo sobre una función gaussiana con el mismo alcance, centrada en cero y apertura = 0.25.

e) Interpretar y comentar los resultados obtenidos en d).

5. Composición de funciones de pertenencia.

a) Concepto teórico: ¿Los adverbios fuzzy se pueden aplicar repetida y/o simultáneamente sobre el mismo término de base? Explicar.

Una función de membresía no puede ser representada simultáneamente por dos adverbios diferentes. Por ejemplo, si tenemos una partición denominada alto a esta partición solamente puede aplicarse un adverbio al definirla en una regla, no así a la misma división en la misma regla darle otro adverbio diferente.

b) Definir según un criterio coherente, las variables lingüísticas calidad (particiones "baja", "media", "alta"); y precio (particiones "barato", "accesible" "caro"). Seleccionar las funciones de pertenencia para cada partición, en un rango [0 ; 10].

c) Luego, mostrar cómo se componen las funciones de pertenencia cuando se aplican los operadores lógicos que se indican a continuación:

c1) Barato y de baja calidad.

Resuelto en el siguiente enlace:

https://github.com/openjuy/ia2022/blob/main/tp02/05/punto5c.m

RESOLVER EN CLASE

c2) Algo caro o de calidad un poco alta.

Resuelto en el siguiente enlace:

https://github.com/openjuy/ia2022/blob/main/tp02/05/punto5c.m

c3) Precio No accesible XNOR calidad media (en este caso conviene descomponer la función XNOR en los operadores básicos.

Resuelto en el siguiente enlace:

https://github.com/openjuy/ia2022/blob/main/tp02/05/punto5c.m

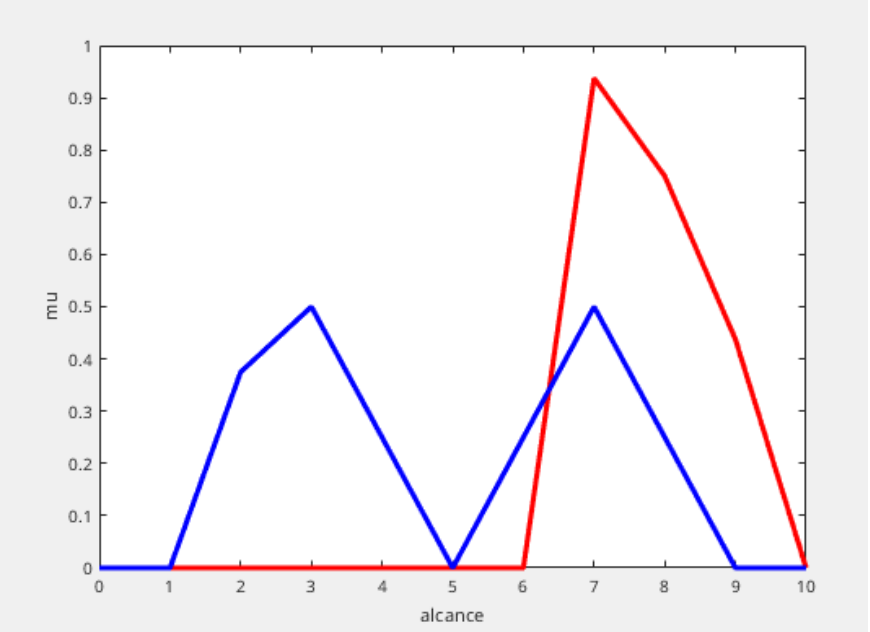
Editor - /home/seba/Documentos/01ia/ia2022/tp02/05/punto5c.m

punto5c.m

```
x=0:10;
%definimos vbles calidad
calidad_baja=trimf(x,[-4 0 4]);
calidad_media=trimf(x,[1 5 9]);
calidad_alta=trimf(x,[6 10 14]);
%definimos vbles precio
precio_barato= trimf(x,[-4 0 4]);
precio_accesible= trapmf(x,[1.4 3 6 8.6]);
precio_caro= trimf(x,[6 10 14]);

% c1) Barato y de baja calidad.
c1= min(precio_barato, calidad_baja)
% plot(x,c1,'r','linewidth',3);
% hold on;

% c2) Algo caro o de calidad un poco alta.
% ALGO(A) = INT(CON(A)) AND NOT(CON(A))
antecedente1=intens(x, concen(x,precio_caro))
antecedente2= 1 - concen(x, precio_caro)
resultado=min(antecedente1, antecedente2)
% plot(x,resultado,'g','linewidth',3);
% hold on;
% UN POCO(A) = INT(DIL(A) AND INT(DIL(NOT(A)
var1=intens(x, dilac(x, calidad_alta))
var2= 1 - concen(x, calidad_alta)
resultado2=min(var1,var2)
resultado3=max(resultado, resultado2)
plot(x,resultado3,'r','linewidth',3);
hold on;

% c3) Precio No accesible XNOR calidad media
resultado4=max(min(precio_accesible, 1 - calidad_media), min(1 - precio_accesible, calidad_media))
plot(x,resultado4,'b','linewidth',3);
xlabel('alcance');
ylabel('mu');
```

https://github.com/openjuy/ia2022/blob/main/tp02/05/codigo.png

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



Las presentes funciones presentan formas sesgadas ya que se encuentran cortas en los lados de la función roja y azul, además se encuentran deformadas, todo esto se debe a que se aplicaron los adverbios. Así como también los operadores fuzzy que son MÁX, MIN y NOT.

6. Adverbios múltiples

a) Concepto teórico: ¿Cómo se interpreta y se procede cuando uno o más adverbios sacan a la función de pertenencia modificada de su recorrido normal? Explicar.

b) Escribir una función de Matlab que aplique sucesivamente hasta tres adverbios fuzzy a una secuencia de pertenencia base mu0, con las siguientes condiciones:

b1) La función debe recibir como entrada una secuencia [x0 , mu0].

b2) 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.

b3) 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].

b4) 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.

4b) anterior como funciones auxiliares (deben estar en la misma carpeta).

b5) Como argumentos de salida, la función debe devolver la secuencia de base x0 (la misma que ingresa), la secuencia de pertenencia mu0 (la misma que ingresa) y generar la secuencia de pertenencia modificada -mumod- por la aplicación sucesiva de los adverbios indicados en el argumento de la función. Por ejemplo: [x0,mu0,mumod] = adverbios_mf(x0,mu0,’no’,’muy’,’aproximadamente’)

b6) 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.

c) Para probar si hay conmutatividad en la aplicación de los adverbios, plantear un ejemplo que aplique tres adverbios sobre una función de base, luego conmutar dos veces el orden de estos adverbios y volver a generar las funciones. Dibujar las tres secuencias resultantes sobre un mismo eje, comparar y comentar.