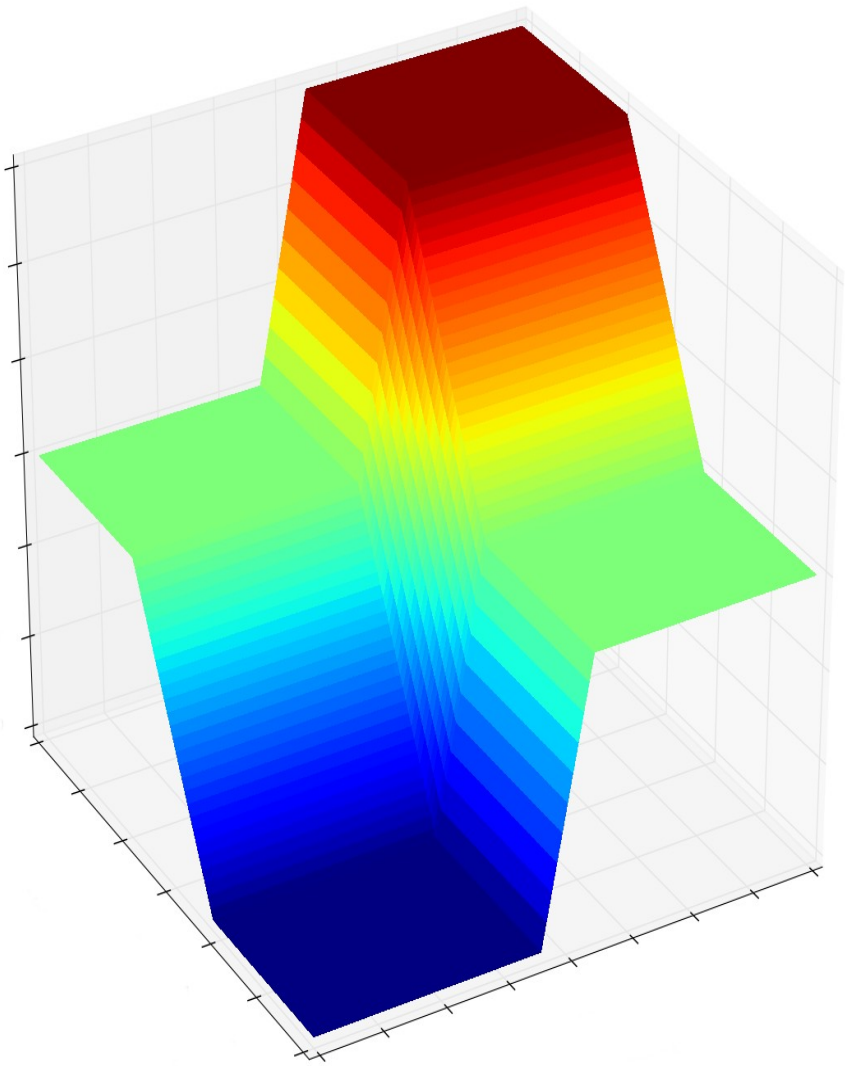
DECISIÓN DIFUSA, UN NUEVO ENFOQUE

CONTROL DIFUSO POR VARIABLE LINGÜÍSTICA

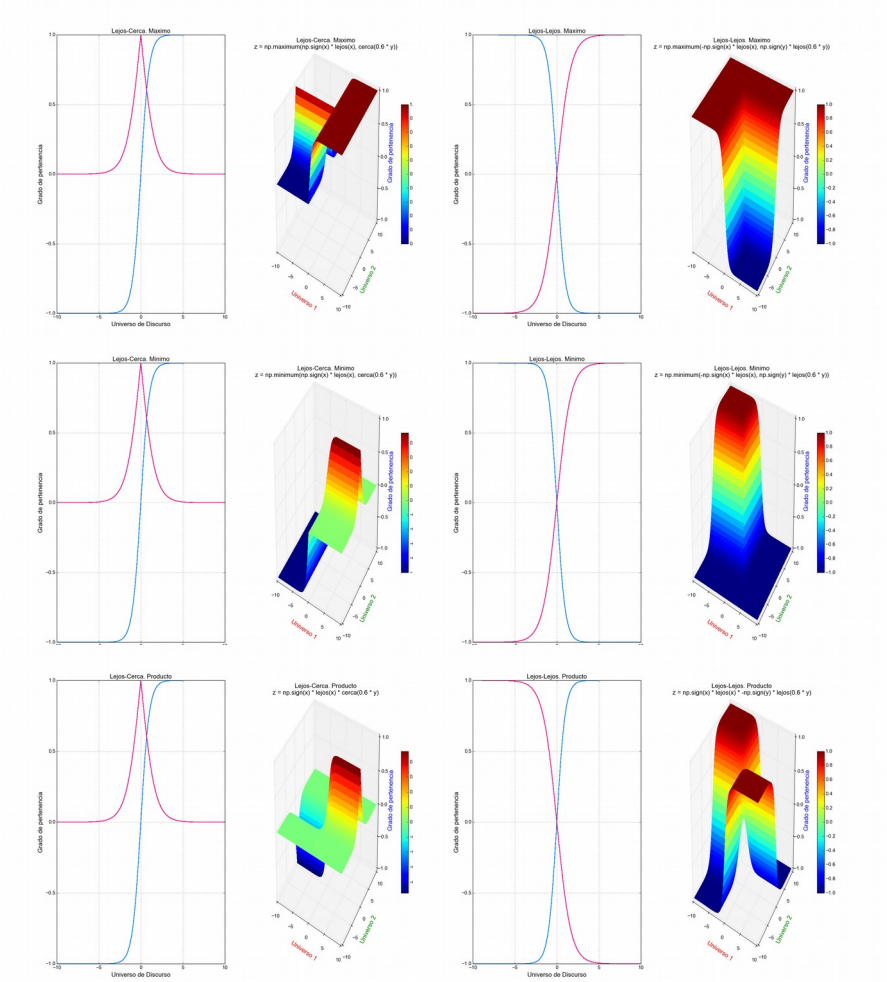
Autor: José María Molina Sánchez.



DECISIÓN DIFUSA, UN NUEVO ENFOQUE

CONTROL DIFUSO POR VARIABLE LINGÜÍSTICA

Autor: José María Molina Sánchez.



Título: DECISIÓN DIFUSA, UN NUEVO ENFOQUE.
Subtítulo: CONTROL DIFUSO POR VARIABLE LINGÜÍSTICA.
Autor: José María Molina Sánchez.
Formato: A5 = 148 x 210 mm.
Páginas: 165.

ÍNDICE.

PARTE 1.

PREFACIO.

Con este libro pretendo demostrar que el actual sistema de control basado en la lógica difusa se puede mejorar, creando un sistema deductivo y control más eficiente energética y computacionalmente. Este nuevo paradigma en los sistemas de control facilita su implementación en todos los sectores dónde se necesita uno de ellos.

Mi formación técnica es la de Técnico Especialista en Electrónica Industrial. Los conocimientos que he ido adquiriendo para este tema han sido de forma totalmente autodidacta.

El libro se divide en dos partes. En la primera desarrollo la decisión borrosa, y en la segunda hago un cambio que simplifica la primera parte.

Doy por supuesto que el lector conoce perfectamente la lógica difusa, y aunque al inicio he puesto una introducción resumida, no es suficiente para entenderla. También doy por sabido que son los sistemas de control. Si no es así probablemente no entienda su contenido.

Conocí por vez primera el concepto de lógica difusa allá por 1.992, en una lectura de la versión en castellano de Scientific American, Investigación y Ciencia. Aunque me fascinó la idea no profundicé en ella, y tuvieron que pasar dos decenios hasta que surgiese de nuevo un renovado interés.

Así que a partir del 2.011 con un renovado interés y con la experiencia adquirida durante años de trabajo en el sector industrial me propuse la tarea de simplificar la lógica borrosa aplicada a los sistemas de control sin que resultara demasiado difícil, esto me llevó unos años de investigación y los escasos conocimientos de lógica difusa que yo tenía los fuí completando autodidactamente. Cuando ya tenía los conocimientos consolidados de lógica difusa advertí que aunque los conceptos son sencillos la forma de aplicarlos era bastante complicada, con numerosos pasos que podrían confundir al lector y ocasionar errores en su implementación, dificultando su implantación masiva para controles basados en muchas entradas con pocas salidas y gran cantidad de

conjuntos difusos por entrada, lo que hace que **el sistema de reglas difusas crezca exponencialmente**. El resultado es este libro en el que he conseguido lo propuesto.

Las matemáticas usadas para comprender esta nueva lógica difusa son las que se dan en secundaria. Así que una persona que tenga esta formación puede llegar a entender sus principios matemáticos.

La simplificación que tenía en mente era que mediante una función sencilla poder deducir su grado de pertenencia, a la vez que esa función fuese un concepto lingüístico. Para ello tuve que desarrollar un nuevo enfoque que expondré en este libro. Hubo muchos errores, búsquedas de concepto y tras un trabajo difícil al final encontré las dos funciones que definían los conceptos buscados y que tenían al mismo tiempo una formulación matemática simple.

De camino en el desarrollo de los conceptos que voy a exponer a continuación me encontré que los pesos de las redes neuronales artificiales (RNA) tienen sentido y son interpretables, y quizás pudiendo deducir de ellos las reglas lógicas que componen el aprendizaje de la RNA sobre lo aprendido por ésta; en el capítulo 23 lo detallo, aunque deberás leerte todos los capítulos anteriores para entender lo que propongo.

Los ejemplos utilizados están sacados de la bibliografía que pueden encontrarse en los sistemas de control que pueblan internet, y para la simulación de ellos he usado XCOS de SCILAB y OMEdit de OpenModelica. En referencias se pueden encontrar algunos de éstos.

La toolbox de fuzzy logic que implementan diferentes programas de desarrollo matemático, ya sean propietarios o libres, se hace innecesaria con esta decisión borrosa. Pudiéndose emplear las funciones no lineales que ya poseen por defecto para implementarla.

Esta lógica difusa en vez de ser algebraica es analítica, y es en el análisis matemático donde tiene su potencial desarrollo.

“La suerte favorece sólo a la mente preparada.”
Isaac Asimov.

1. ANTECEDENTES.

En 1.965 Lotfi Asker Zadeh introduce la teoría de conjuntos borrosos, como un mecanismo para representar la vaguedad e imprecisión de los conceptos empleados en el lenguaje natural.

A mediados de los 70 llega la ampliación del concepto de conjunto al de lógica, apareciendo las lógicas borrosas y las aplicaciones a sistemas de control. Hoy en día son muchas las aplicaciones tanto industriales como domésticas que hacen uso de este paradigma.

2. CRONOLOGÍA DEL DESARROLLO DE LA TECNOLOGÍA DE LA LÓGICA DIFUSA.

1.965 Prof. Lofti Asker Zadeh, Facultad de Ingeniería Eléctrica en U.C. Berkeley, establece el origen de la Teoría de Conjuntos Difusos.

1.970 Primera Aplicación de Lógica Difusa en Ingeniería de Control (Europa).

1.975 Introducción de la Lógica Difusa en Japón.

1.980 Verificación Empírica de la Lógica Difusa en Europa.

1.985 Extensión de la Aplicación de Lógica Difusa en Japón.

1.990 Extensión de la Aplicación de Lógica Difusa en Europa.

1.995 Extensión de la Aplicación de Lógica Difusa en EEUU.

2.000 La Lógica Difusa se ha convertido en una técnica estándar para Control Multivariable.

3. INTRODUCCIÓN A LOS CONJUNTOS DIFUSOS.

Los conjuntos borrosos los definió Zadeh como una prolongación de los conjuntos clásicos en los que se pueden introducir conceptos humanos vagos e imprecisos. Expresiones del tipo "ese hombre es alto", "hoy hace calor" o "voy a tardar un rato" son habituales en nuestro lenguaje. Sin embargo no es fácil precisar que entendemos por "alto", "calor" o "un rato". Difícilmente nos pondremos de acuerdo en concretar a partir de qué altura puede considerarse alta a una persona, o a partir de que temperatura se dice que hace calor, o cuánto tiempo supone esperar un rato. Sin embargo los seres humanos no encontramos dificultades en razonar con estos conceptos imprecisos.

En la teoría clásica de conjuntos, formulada por Georg Cantor a finales del siglo XIX, un cierto elemento puede pertenecer o no a un determinado conjunto, es decir, la relación de pertenencia puede tomar únicamente los valores verdadero o falso, en la lógica de Boole 1 y 0 respectivamente. La modificación propuesta por Zadeh consiste en introducir un grado de pertenencia, esto es, expresar la pertenencia de un elemento a un conjunto como un número real en el intervalo [0, 1]. Un grado de pertenencia 0 indica que un elemento no pertenece a un determinado conjunto, mientras que un grado de pertenencia 1 indica que un elemento pertenece totalmente al conjunto.

Al igual que en la lógica boolena podemos establecer una tabla de verdad para los valores difusos.

Valor de Verdad	Categoría
0,0	falso
0,1	casi falso
0,2	bastante falso
0,3	algo falso
0,4	más falso que verdadero
0,5	tan verdadero como falso
0,6	más verdadero que falso

0,7	algo verdadero
0,8	bastante verdadero
0,9	casi verdadero
1,0	verdadero

La lógica borrosa se fundamenta en tres conceptos: los conjuntos difusos, las variables lingüísticas y reglas difusas SI- ENTONCES- .

Definición: Un conjunto difuso A se define como una función de pertenencia que enlaza o empareja los elementos de un dominio o universo de discurso X con elementos del intervalo [0, 1]:

$$A : X \rightarrow [0,\ 1]$$

Cuanto más cerca esté A(x) del valor 1, mayor será la pertenencia del objeto x al conjunto A. Los valores de pertenencia varían entre 0 (no pertenece en absoluto) y 1 (pertenencia total), ver tabla de verdad.

Función de pertenencia: Un conjunto difuso puede representarse también gráficamente como una función. En el eje de abscisas se representa el universo de discurso X, y en el de ordenadas los grados de pertenencia en el intervalo [0, 1].

Variables lingüísticas: Son variables que se representan mediante palabras, y el valor que toman se hace mediante una función de pertenencia.

Reglas difusas SI- ENTONCES- : Son condiciones en las que según se cumpla un determinado conjunto de entrada, habrá de cumplirse otro en la salida.

A partir de los fundamentos tenemos los elementos básicos de un sistema de lógica difusa, y son: borrosificación, inferencia, composición y desborrosificación.

1. Borrosificación: proceso por el que se aplican las funciones de pertenencia definidas en las variables de entrada sobre los valores reales de los atributos, para determinar el grado de verdad de las premisas de cada regla. La determinación de los grados de pertenencia suele realizarse mediante métodos empíricos.

2. Inferencia: a partir del valor de verdad calculado para las premisas de cada regla se calcula el de la conclusión de la misma. Este resultado es un subconjunto borroso aplicable a cada variable de salida de cada regla. Se usa la regla SI- ENTONCES-.

3. Composición: se combinan los subconjuntos borrosos obtenidos para las variables de salida en un único conjunto borroso.

4. Desborrosificación: A veces es útil examinar los conjuntos borrosos resultantes del proceso de composición, aunque otras veces se necesita convertir el valor borroso en un número, para lo que se aplica un proceso de desborrosificación. Dos de las técnicas más usadas de desborrosificación son la del CENTROIDE y el valor MÁXIMO, aunque existen otras tećnicas.

También se pueden hacer operaciones con los conjuntos difusos, extendiendo las propiedades de los conjuntos clásicos.

Operaciones de conjuntos: A(x), B(x) son conjuntos difusos en el universo X y $\mu_A(x)$, $\mu_B(x)$ sus respectivas funciones de pertenencia. Las tres operaciones que se definen son: intersección, unión y complemento, que se corresponden con los conectivos lógicos AND, OR y NOT respectivamente.

Intersección (AND):

$$(A \cap B)(x) = A(x) \wedge B(x) = \min\{A(x), B(x)\} = \min\{\mu_A(x), \mu_B(x)\}$$

Unión (OR):

$$(A \cup B)(x) = A(x) \vee B(x) = \max\{A(x), B(x)\} = \max\{\mu_A(x), \mu_B(x)\}$$

Complemento (NOT):

$$\overline{A}(x) = 1 - A(x) = 1 - \mu_A(x)$$

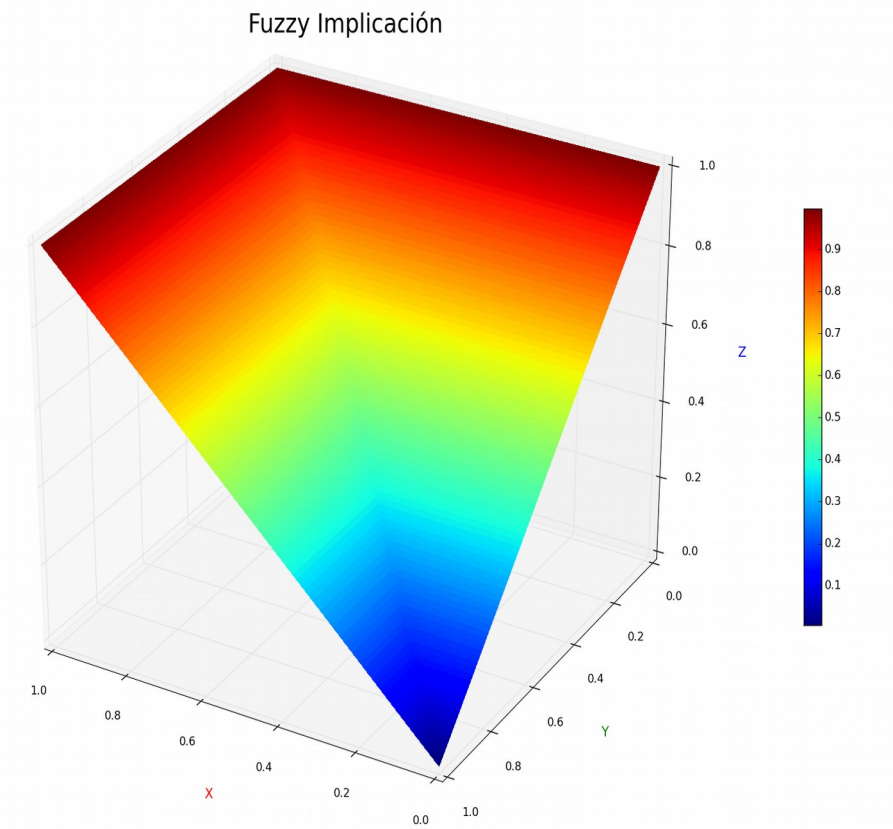
De éstas tres operaciones se obtienen otras tres más que son parte de la lógica estándar: implicación, equivalencia y XOR que se estudian a continuación.

Implicación:

$$(A \Rightarrow B)(x) = (\overline{A} \cup B)(x) = \overline{A}(x) \vee B(x)$$
$$= \max\{\overline{A}(x),\ B(x)\} = \max\{1 - \mu_A(x),\ \mu_B(x)\}$$

En la gráfica es:

$$Z = \max\{1 - X,\ Y\}$$

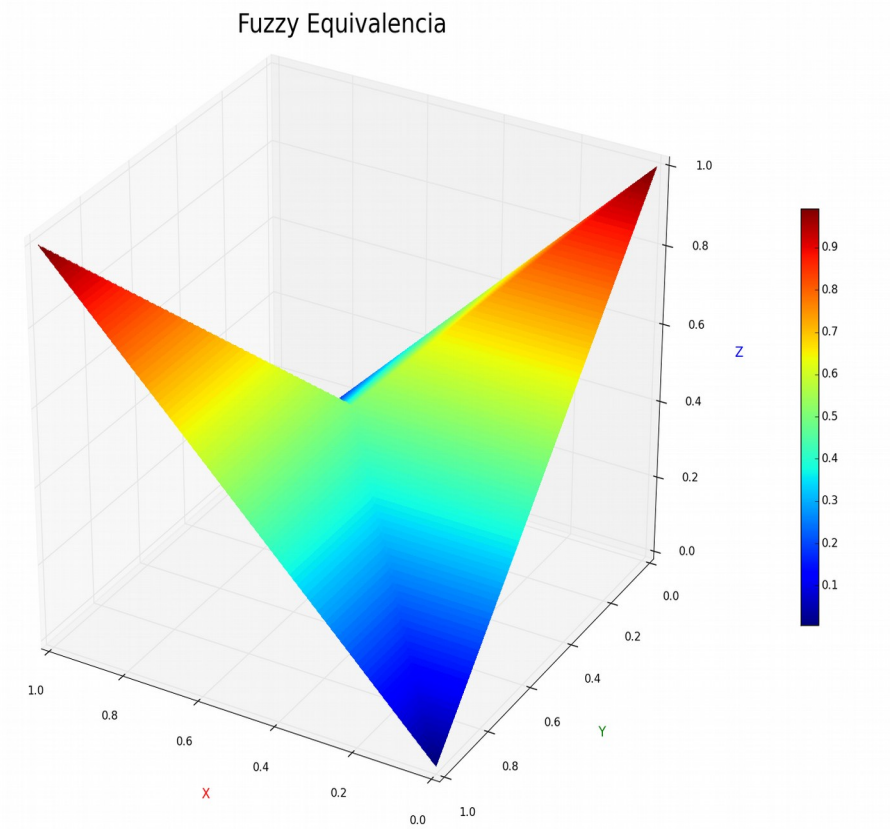


Equivalencia:

$$(A \Leftrightarrow B)(x) = ((A \cap B) \cup (\overline{A} \cap \overline{B}))(x) = (A(x) \wedge B(x)) \vee (\overline{A}(x) \wedge \overline{B}(x)) =$$
$$\max\{\min\{A(x),\ B(x)\},\ \min\{\overline{A}(x),\ \overline{B}(x)\}\} =$$
$$\max\{\min\{\mu_A(x),\ \mu_B(x)\},\ \min\{1 - \mu_A(x),\ 1 - \mu_B(x)\}\}$$

En la gráfica es:

$$Z = \max\{\min\{X,\ Y\},\ \min\{1 - X,\ 1 - Y\}\}$$

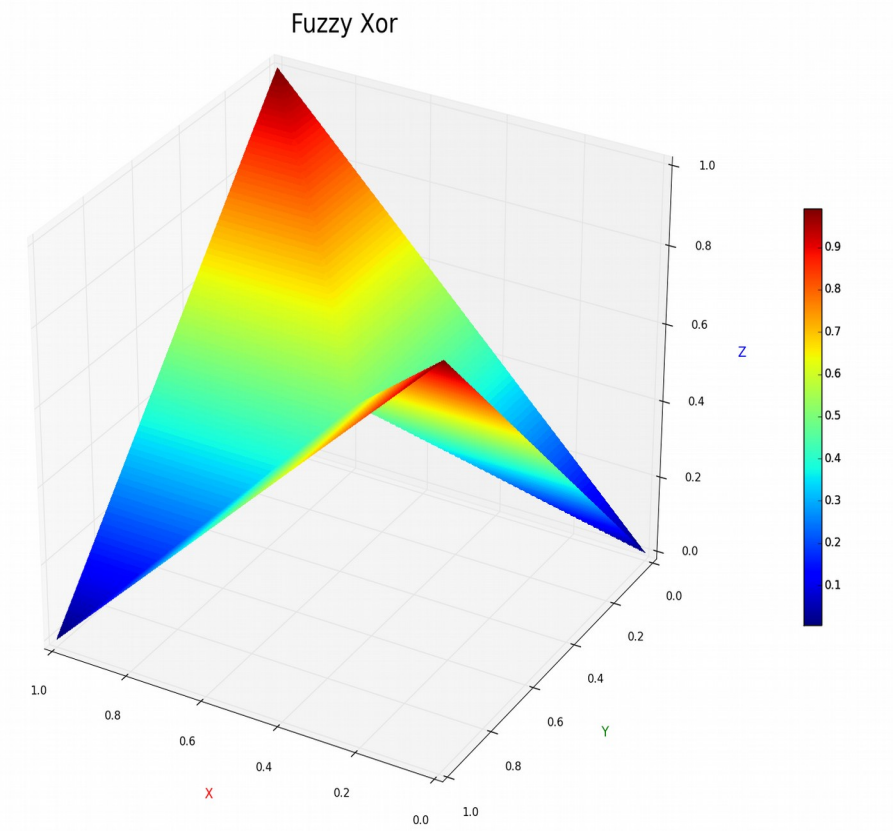


XOR:

$$(A \text{ XOR } B)(x) = A(x) \text{ XOR } B(x) = (A(x) \wedge \overline{B}(x)) \vee (\overline{A}(x) \wedge B(x)) =$$
$$\max\{\ \min\{A(x),\ 1 - B(x)\},\ \min\{1 - A(x),\ B(x)\}\ \} =$$
$$\max\{\ \min\{\mu_A(x),\ 1 - \mu_B(x)\},\ \min\{1 - \mu_A(x),\ \mu_B(x)\}\ \}$$

En la gráfica es:

$$Z = \max\{\min\{X,\ 1 - Y\},\ \min\{1 - X,\ Y\}\}$$



Podemos apreciar que $(A \Leftrightarrow B)(x) = (A \ \overline{XOR} \ B)(x)$.

Te reto a que hagas un sistema combinacional difuso con equivalencia y XOR cómo entradas, con una única salida y con las funciones difusas AND y OR; puedes inspirarte en el apéndice J para completar el reto.

En el apéndice H se encuentran los programas en Python que grafican las operaciones de implicación, equivalencia y XOR.

4. CONJUNTO DIFUSO DE VARIABLE LINGÜÍSTICA.

Para la decisión borrosa que he concebido, he prescindido de las variables lingüísticas al uso y sus correspondientes conjuntos, aunando en una misma función la variable lingüística, los modificadores y la función de pertenencia. El único concepto que permanece inalterable es el grado de pertenencia, de ahí el nuevo enfoque que da título a este libro.

Atendiendo solamente a la definición de conjunto difuso como una función de pertenencia que enlaza o empareja los elementos de un dominio o universo de discurso X con elementos del intervalo [0, 1]:

$$A : X \rightarrow [0,\ 1]$$

he desarrollado un método basado en **dos variables lingüísticas** que utilizan cada una **funciones hiperbólicas** para definir el **grado de pertenencia** del universo de discurso.

A este método lo he llamado **ALBHI**, ALgoritmo Borroso HIperbólico.

5. FUNCIÓN DE PERTENENCIA.

Habitualmente usamos muchos conceptos vagos e imprecisos en nuestro lenguaje, "voy a tardar un poco", "tengo mucho trabajo", "hace mucho frío", "tengo poco dinero", "consume mucha energía", los conceptos imprecisos ‘poco’, ‘mucho’ son difíciles de tratar para una máquina basada en la lógica Booleana, sin embargo definiendo una nueva función de pertenencia con el nuevo enfoque se podrían resumir en dos conceptos que englobarían a los demás, los dos conceptos los llamo **lejos** y **cerca**, y en ellos se basa la **decisión borrosa**.

Estos **adverbios** serán la únicas **variables lingüísticas** para esta nueva forma de lógica borrosa.

Se define lejos cómo: a gran distancia, en lugar o tiempo distante o remoto. Cerca es antónimo de lejos quedando definida por la negación lógica o su complemento en la lógica difusa.

Veamos un ejemplo sencillo, tomando como origen España y en particular la Región de Murcia que es donde vivo habitualmente, el predicado “todos los chinos viven lejos de mi” tiene un grado de pertenencia de 1, puesto que si cogemos cualquier habitante situado en cualquier lugar de China podemos aplicar la misma proposición. Sin embargo si nos atenemos a los habitantes de la región murciana el predicado “todos los murcianos viven lejos de mi” ya no es cierta, ya que para mis vecinos el grado de pertenencia será cero o cercano a él, y los que viven en las zonas de la regiones limítrofes si será cierto, depende también en qué lugar de Murcia me encuentre, ¿se puede expresar de una manera sencilla, cuantificable y rigurosa lo expuesto por estos dos ejemplos?.

ALBHI se basa por tanto en el uso de estas **variables lingüísticas** determinadas ambas por una **función hiperbólica** y su complementaria respectivamente; cada función tiene valores en el intervalo real [0, 1]; de ellas se deriva su correspondiente **función de pertenencia**.

La variable lingüística principal es lejos y su función hiperbólica y de pertenencia se determinan:

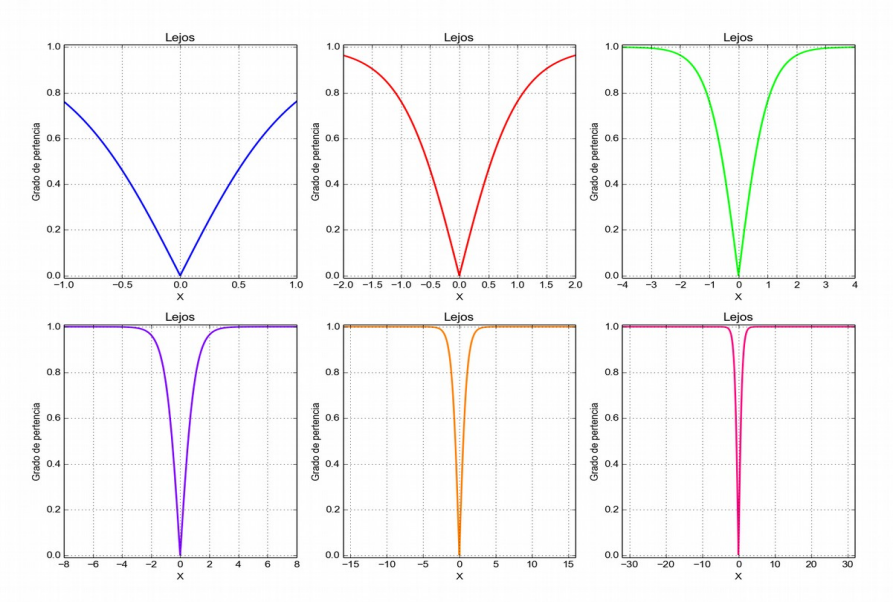
$$\text{lejos}(x) = |\tanh(x)| \qquad \mu_L(x) = \text{lejos}(x)$$

por tanto siendo el antónimo de lejos el término cerca su función hiperbólica se determina por la función complementaria:

$$\text{cerca}(x) = 1 - |\tanh(x)| \qquad \mu_C(x) = \text{cerca}(x)$$

El valor de cada función indica cuán lejos o cerca está x de cero, indicado por el grado de pertenencia de cada valor de x.

A continuación se representa la función lejos(x) correspondientes a los dominios ±1, ±2, ±4, ±8, ±16 y ±32.

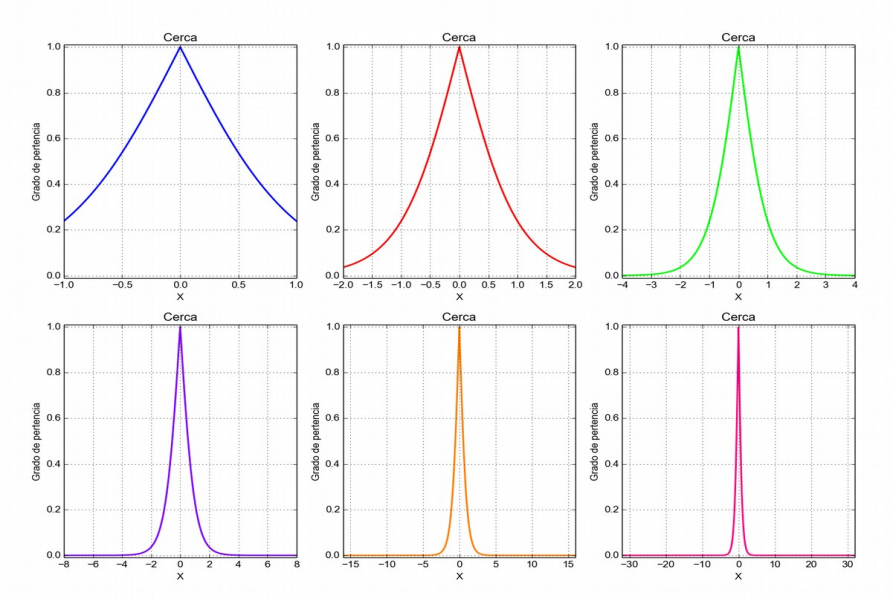


Para lejos(x), $\mu_L(x)$ indica el grado de pertenencia del conjunto de los números reales a 0, para $\mu_L(x) = 0$ es nada lejos, y para $\mu_L(x) = 1$ es muy lejos.

Vemos que ±0,1 está casi nada lejos de 0, que ±0,5 está un poco lejos de 0, que ±1 está algo lejos de 0, que ±2 está lejos de 0, pero a partir de ±5 todos los números están lejísimos de 0, véase la siguiente tabla de verdad.

x	0,0	±0,1	±0,5	±1,0	±1,5	±2,0	±3,0	±4,0	±5,0
$\mu_L(x)$	0,000	0,100	0,462	0,762	0,905	0,964	0,995	0,999	1,000

A continuación se representa la función cerca(x) correspondientes a los dominios ±1, ±2, ±4, ±8, ±16 y ±32.



Para cerca(x), $\mu_C(x)$ indica el grado de pertenencia del conjunto de los números reales a 0, para $\mu_C(x) = 0$ es muy cerca, y para $\mu_C(x) = 1$ es nada cerca.

Vemos que ±0,1 está muy cerca de 0, que ±0,5 está un poco cerca de 0, que ±1 está poco cerca de 0, que ±2 está muy poco cerca de 0, pero a partir de ±5 todos los números están nada cerca de 0.

x	0,0	±0,1	±0,5	±1,0	±1,5	±2,0	±3,0	±4,0	±5,0
$\mu_C(x)$	1,000	0,900	0,538	0,238	0,095	0,036	0,005	0,001	0,000

En el ejemplo “todos los chinos viven lejos de mi” si en el eje X representamos la distancia en Km, será $\mu_L(x) = 1$ para cualquier distancia;

valor de x; desde China. Sin embargo el predicado “todos los murcianos viven lejos de mi” expresado en las mismas unidades será $\mu_L(x) \approx 0$ para todos mis vecinos ($x \leq 0{,}050$ Km), con la tabla de verdad para $\mu_L(x)$ unos párrafos más atrás se puede comprender perfectamente el ejemplo.

Los modificadores lingüísticos de la lógica de Zadeh se corresponderían con el grado de pertenencia de esta decisión borrosa.

Los cuatro elementos básicos de la lógica borrosa actual: la borrosificación, inferencia, composición y desborrosificación se hace innecesaria.

Para una implementación mejor en el ámbito industrial es aconsejable que el eje de abscisas represente el error en vez de cualquier valor del universo de discurso, ello lo veremos en los ejemplos.

El uso de cada variable lingüística dependerá del sistema a controlar, utilizaremos la función lejos(x) cuando la variable controlada sea cero para una variable de entrada cero, y emplearemos la función cerca(x) cuando la variable controlada sea el máximo para una variable de entrada cero.

Estos conceptos serán muy importantes en el cálculo de controladores, pudiendo programar con muy pocos recursos y en poco tiempo, un sistema de control muy robusto que se adecúe al sistema a controlar.

Hay infinidad de funciones lejos(x) que pueden utilizarse como la que he definido siempre y cuando cumplan dos condiciones, que en x = 0 sea lejos(x) = 0 y que la función tienda asintóticamente a uno (o sea igual a 1) cuando x tienda a más o menos infinito, por ejemplo las funciones $1-1/(1+x^2)$, $1-\mathrm{sech}(x)$, $1-e^{-x^2}$, $1-e^{-|x|}$, $\tanh^2(x)$, $|2/(1+e^{-x})-1|$ cumplen con esta regla. Para la función cerca(x) habrá que tomar la función complementaria.

$$\lim_{x \to \pm 0} \text{lejos}(x) = 0 \qquad \lim_{x \to \pm \infty} \text{lejos}(x) = 1$$

$$\lim_{x \to \pm 0} \text{cerca}(x) = 1 \qquad \lim_{x \to \pm \infty} \text{cerca}(x) = 0$$

En el apéndice F relaciono la función de pertenencia lejos con otras dos que se utilizan mucho en machine learning, las funciones σ y tanh. Las funciones lejos(x) antes expuestas pueden usarse también cómo función de activación en las redes neuronales artificiales.

En la lógica difusa la función de pertenencia se altera mediante los modificadores lingüísticos, haciendo que el sistema de control tenga un comportamiento distinto. Para que en **ALBHI** ocurra lo mismo hay que introducir un nuevo concepto, el de **punto máximo**.

6. PUNTO MÁXIMO.

Este término nos dice en qué valor de x la función de pertenencia comienza a valer 1 y define cómo cambia la salida de la función, y por tanto su significado, **potenciando o debilitando la intensidad de la variable lingüística**. Por ello el punto máximo va a multiplicar siempre a cada valor del universo de discurso. Para un mejor manejo de las funciones vamos a utilizar la la abreviatura pmL en la función lejos y pmC en la función cerca. Ambos pueden ser positivos o negativos.

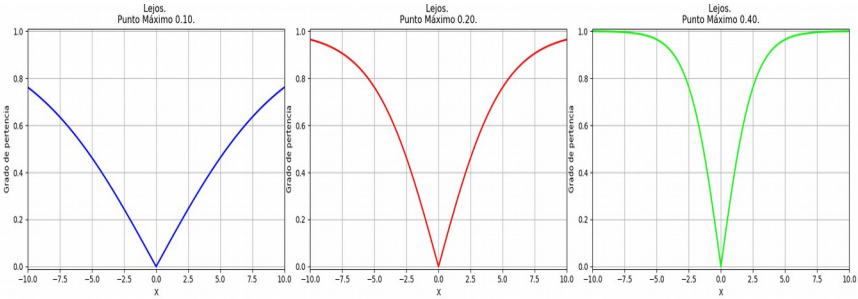
Para la variable lingüística lejos su punto máximo es:

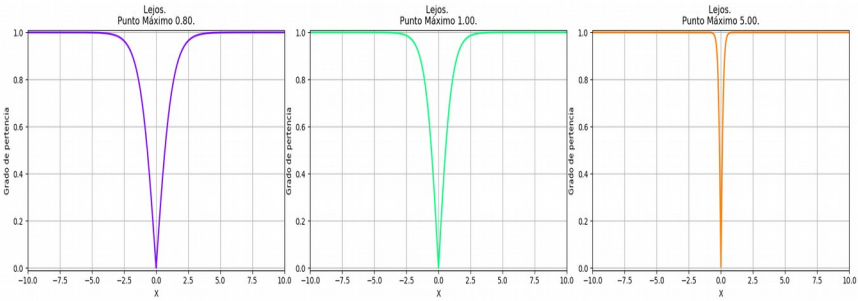
$$\mu_L(x) = \text{lejos}(\text{pmL} \cdot x)$$

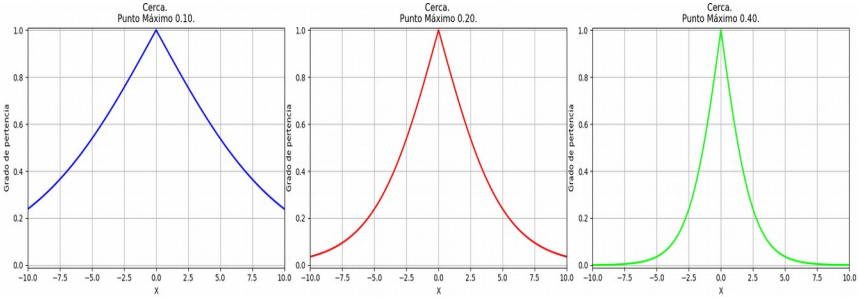
Para la variable lingüística cerca su punto máximo es:

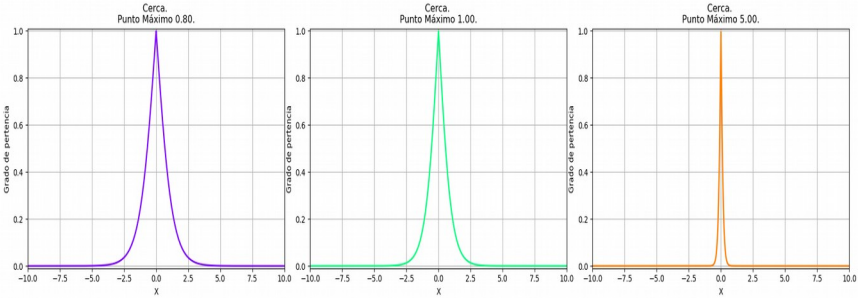
$$\mu_C(x) = \text{cerca}(\text{pmC} \cdot x)$$

En la página siguiente se representan los puntos máximos con los valores 0,1; 0,2; 0,4; 0,8; 1,0 y 5,0 para las funciones de pertenencia $\mu_L(x)$ y $\mu_C(x)$ en el dominio de $x \in [-10, 10]$.


Lejos.
Punto Máximo 0.10.
Lejos.
Punto Máximo 0.20.
Lejos.
Punto Máximo 0.40.
Grado de pertencia
X


Lejos.
Punto Máximo 0.80.
Lejos.
Punto Máximo 1.00.
Lejos.
Punto Máximo 5.00.
Grado de pertencia
X


Cerca.
Punto Máximo 0.10.
Cerca.
Punto Máximo 0.20.
Cerca.
Punto Máximo 0.40.
Grado de pertencia
X


Cerca.
Punto Máximo 0.80.
Cerca.
Punto Máximo 1.00.
Cerca.
Punto Máximo 5.00.
Grado de pertencia
X

El punto máximo modifica el comportamiento de la función de pertenencia. Para un punto máximo pequeño en la función $\mu_L(x)$ los valores de x que antes se consideraban muy lejos se convierten en algo, poco o nada lejos, y para un valor grande lo que antes era casi nada lejos se puede volver muy lejos.

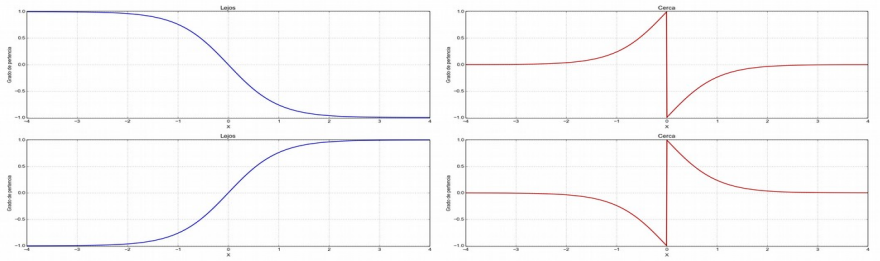
En la función $\mu_C(x)$ un punto máximo pequeño conlleva que los valores de x que antes se consideraban nada cerca se convierten algo o muy cerca, y para un valor grande lo que antes era muy cerca se puede volver nada cerca.

Los grados de pertenencia $\mu_L(x)$ y $\mu_C(x)$ están definidos entre 0 y 1. Pero en cualquier sistema a controlar deberán estar entre -1 y 1 debiendo multiplicarlos por ±signo(x), quedando las funciones de pertenencia definidas de la siguiente manera:

$$\mu_L(x) = \pm\text{signo}(\text{pmL} \cdot x) \cdot \text{lejos}(\text{pmL} \cdot x)$$

$$\mu_C(x) = \pm\text{signo}(\text{pmC} \cdot x) \cdot \text{cerca}(\text{pmC} \cdot x)$$

Las gráficas de los signos se representan a continuación.



Cómo se observa en la gráfica la función $\mu_L(x)$ es la tanh(x). La función $\mu_C(x)$ sin embargo no tiene sentido, puesto que en cero tiene dos valores, así que no se puede utilizar en esta forma, pero sí en su forma original

$\text{cerca}(x) = 1 - |\tanh(x)|$ con su respectiva función de pertenencia $\mu_C(x) = \text{cerca}(\text{pmC} \cdot x)$.

Si bien la función lejos(x) se puede poner cómo la ±tanh(x), no es posible hacer lo mismo con la función cerca(x), por lo que se aconseja seguir utilizando ±signo(x) para una mejor estética.

Si el universo de discurso es el error y el punto máximo es 1, podemos considerar el dominio de éste en [-4, 4] como máximo.

Para el punto máximo con valores mayores que uno la función de pertenencia tiende a estrecharse, esto hará que el sistema de control tenga un funcionamiento rápido y abrupto. Cuando el punto máximo tiene valores menores que uno la función de pertenencia tiende a ensancharse, esto hará que el sistema de control tenga un funcionamiento lento y suave.

Al considerar valores menores que cero en la función de pertenencia estamos teniendo en cuenta un valor lógico negativo, siendo ilógico por el uso de la lógica booleana y borrosa. Ésta inverosimilitud del signo en los valores lógicos de salida; y al tratarse de un sistema de regulación y control; la podemos estimar entre otras cosas cómo los sentidos de giro de un motor, y su valor la cantidad de potencia, par, aceleración angular, velocidad angular, ángulo u otra magnitud a controlar del mismo. Así que si el motor a controlar no es de CC, el sistema de control tendrá dos salidas por motor, una para controlar un parámetro y otra para el sentido de giro. Otra consideración posible es cuando valoramos el error, que puede ser positivo o negativo realizando el sistema de control la acción correspondiente, en los ejemplos se verá claramente a que me refiero.

La función de pertenencia tiende asintóticamente a 1 cuando x vale ±∞, pero necesitamos en las simulaciones que a un determinado valor de x sea 1; cosa que en los controles reales no hace falta por ser discretos o de 8 bits; así que para un valor x determinado la función de pertenencia sea μ(x) = 1,000 hay que redefinir un poco dicha función.

Para la variable lingüística lejos(x) es:

$$\text{lejos}(x) = \min\left(\frac{|\tanh(x)|}{0{,}999},\ 1\right)$$

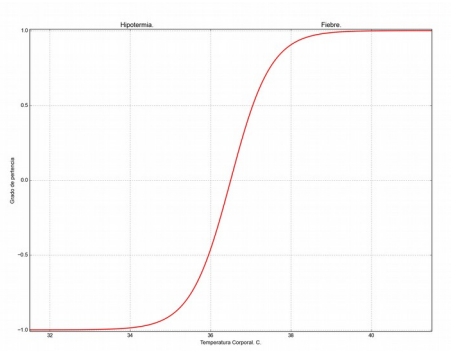
$$\mu_L(x) = \pm\text{signo}(\text{pmL} \cdot x) \cdot \text{lejos}(\text{pmL} \cdot x)$$

y para la variable lingüística cerca(x) es:

$$\text{cerca}(x) = \max\left(1 - \frac{|\tanh(x)|}{0{,}999},\ 0\right)$$

$$\mu_C(x) = \pm\text{signo}(\text{pmC} \cdot xx) \cdot \text{cerca}(\text{pmC} \cdot x)$$

De esta forma para valores superiores de |tanh(x)| a 0,999 en la función de pertenencia μ(x) valdrá 1 ó 0 según la variable lingüística correspondiente y para valores inferiores variará entre 0 y 1.



Tomemos cómo ejemplo la temperatura corporal, con el nuevo enfoque toma otra forma de entenderlo. Si definimos el conjunto como Hipotermia-Fiebre, en 36,5 ºC el grado de pertenencia es 0, para temperaturas mayores de 41,5 ºC es 1, y para temperaturas menores de 31,5 ºC es -1. La función a utilizar será lejos(x) y el pmL = 0,760. Para la hipotermia usamos el grado de pertenencia menor que cero, que es lo contrario de la fiebre. Según el grado de pertenencia usaremos para temperaturas mayores de 36,5 ºC términos tales cómo “escasa fiebre”, “ligera fiebre”, “temperatura alta”, “bastante fiebre” y finalmente “mucha fiebre”. Para temperaturas menores en vez de fiebre nos referiremos a hipotermia con los mismos modificadores lingüísticos.

En la elección de cada función de pertenencia hay implícito el significado del dominio, en el ejemplo de la temperatura corporal sería Hipotermia-Fiebre, y el punto máximo define un contexto. Por tanto el significado del dominio es al mismo tiempo una regla lógica dada por nosotros mismos.

Si el sistema a controlar dispone de una entrada y una salida la función de control es sencilla, pero para dos o más entradas hay que recurrir a **reglas de inferencia**, que en nada son cómo las de la lógica difusa, ya que en ningún momento se usan reglas de la forma SI- ENTONCES- siendo el razonamiento de forma totalmente diferente mediante funciones, las deducciones serán tanto analíticas cómo lógicas.

El cálculo de los puntos máximos se explica en el apéndice A.

7. REGLAS DE INFERENCIA.

Podemos distinguir dos formas de inferir los consecuentes con **ALBHI**; la primera es multiplicando el grado de pertenencia por el valor máximo de la variable a controlar, y la segunda es la antiimagen del grado de pertenencia del conjunto de salida del grado de pertenencia del conjunto de entrada.

8. MULTIPLICAR POR EL GRADO DE PERTENENCIA.

Para inferir los consecuentes con **ALBHI** hay que tener en cuenta que las variables lingüísticas son funciones analíticas y no lógicas. Por tanto habrá que usar operaciones aritméticas para hallar las conclusiones, pudiendo utilizar funciones definidas a trozos si es menester.

La ventaja de la lógica difusa sobre el PID es que permite el control de un sistema multivariable, y con **ALBHI** se torna más sencillo pues cada entrada se asocia a una única función de pertenencia.

Para dos entradas y una salida establecemos las reglas de inferencia de las soluciones en los sistemas de control con cuatro[1] operaciones: el producto, el valor máximo, el valor mínimo y XOR difuso de las variables de entrada; incluyendo el signo de la función y sí hay o no valor opuesto de la función signo(x). La función XOR difuso se define en la página 14.

Sean X_1 y X_2 los universos de entrada de un sistema y $\mu(x_1)$, $\mu(x_2)$ sus respectivas funciones de pertenencia. Las 9 superficies de control o matrices de inferencia resultantes serán las siguientes:

Para el producto de las variables de entrada tenemos:

1. $\mu(x_1, x_2) = \mu(x_1) \cdot \mu(x_2)$
2. $\mu(x_1, x_2) = \text{signo}(x_1) \cdot \mu(x_1) \cdot \mu(x_2)$
3. $\mu(x_1, x_2) = -\text{signo}(x_1) \cdot \mu(x_1) \cdot \mu(x_2)$
4. $\mu(x_1, x_2) = \mu(x_1) \cdot \text{signo}(x_2) \cdot \mu(x_2)$
5. $\mu(x_1, x_2) = \mu(x_1) \cdot -\text{signo}(x_2) \cdot \mu(x_2)$
6. $\mu(x_1, x_2) = \text{signo}(x_1) \cdot \mu(x_1) \cdot \text{signo}(x_2) \cdot \mu(x_2)$
7. $\mu(x_1, x_2) = \text{signo}(x_1) \cdot \mu(x_1) \cdot -\text{signo}(x_2) \cdot \mu(x_2)$
8. $\mu(x_1, x_2) = -\text{signo}(x_1) \cdot \mu(x_1) \cdot \text{signo}(x_2) \cdot \mu(x_2)$
9. $\mu(x_1, x_2) = -\text{signo}(x_1) \cdot \mu(x_1) \cdot -\text{signo}(x_2) \cdot \mu(x_2)$

Para el máximo de las variables de entrada tenemos:

1. $\mu(x_1, x_2) = \max(\mu(x_1), \mu(x_2))$
2. $\mu(x_1, x_2) = \max(\text{signo}(x_1) \cdot \mu(x_1), \mu(x_2))$
3. $\mu(x_1, x_2) = \max(-\text{signo}(x_1) \cdot \mu(x_1), \mu(x_2))$
4. $\mu(x_1, x_2) = \max(\mu(x_1), \text{signo}(x_2) \cdot \mu(x_2))$

1 Se pueden implementar más operaciones para obtener el consecuente, cómo la implicación, la equivalencia u otra definida por nosotros.

5. $\mu(x_1, x_2) = \max(\mu(x_1), -\text{signo}(x_2) \cdot \mu(x_2))$
6. $\mu(x_1, x_2) = \max(\text{signo}(x_1) \cdot \mu(x_1), \text{signo}(x_2) \cdot \mu(x_2))$
7. $\mu(x_1, x_2) = \max(\text{signo}(x_1) \cdot \mu(x_1), -\text{signo}(x_2) \cdot \mu(x_2))$
8. $\mu(x_1, x_2) = \max(-\text{signo}(x_1) \cdot \mu(x_1), \text{signo}(x_2) \cdot \mu(x_2))$
9. $\mu(x_1, x_2) = \max(-\text{signo}(x_1) \cdot \mu(x_1), -\text{signo}(x_2) \cdot \mu(x_2))$

Para el mínimo de las variables de entrada tenemos:

1. $\mu(x_1, x_2) = \min(\mu(x_1), \mu(x_2))$
2. $\mu(x_1, x_2) = \min(\text{signo}(x_1) \cdot \mu(x_1), \mu(x_2))$
3. $\mu(x_1, x_2) = \min(-\text{signo}(x_1) \cdot \mu(x_1), \mu(x_2))$
4. $\mu(x_1, x_2) = \min(\mu(x_1), \text{signo}(x_2) \cdot \mu(x_2))$
5. $\mu(x_1, x_2) = \min(\mu(x_1), -\text{signo}(x_2) \cdot \mu(x_2))$
6. $\mu(x_1, x_2) = \min(\text{signo}(x_1) \cdot \mu(x_1), \text{signo}(x_2) \cdot \mu(x_2))$
7. $\mu(x_1, x_2) = \min(\text{signo}(x_1) \cdot \mu(x_1), -\text{signo}(x_2) \cdot \mu(x_2))$
8. $\mu(x_1, x_2) = \min(-\text{signo}(x_1) \cdot \mu(x_1), \text{signo}(x_2) \cdot \mu(x_2))$
9. $\mu(x_1, x_2) = \min(-\text{signo}(x_1) \cdot \mu(x_1), -\text{signo}(x_2) \cdot \mu(x_2))$

Para XOR difuso de las variables de entrada tenemos:

1. $\mu(x_1, x_2) = \text{XOR}\ (\mu(x_1), \mu(x_2))$
2. $\mu(x_1, x_2) = \text{XOR}\ (\text{signo}(x_1) \cdot \mu(x_1), \mu(x_2))$
3. $\mu(x_1, x_2) = \text{XOR}\ (-\text{signo}(x_1) \cdot \mu(x_1), \mu(x_2))$
4. $\mu(x_1, x_2) = \text{XOR}\ (\mu(x_1), \text{signo}(x_2) \cdot \mu(x_2))$
5. $\mu(x_1, x_2) = \text{XOR}\ (\mu(x_1), -\text{signo}(x_2) \cdot \mu(x_2))$
6. $\mu(x_1, x_2) = \text{XOR}\ (\text{signo}(x_1) \cdot \mu(x_1), \text{signo}(x_2) \cdot \mu(x_2))$
7. $\mu(x_1, x_2) = \text{XOR}\ (\text{signo}(x_1) \cdot \mu(x_1), -\text{signo}(x_2) \cdot \mu(x_2))$
8. $\mu(x_1, x_2) = \text{XOR}\ (-\text{signo}(x_1) \cdot \mu(x_1), \text{signo}(x_2) \cdot \mu(x_2))$
9. $\mu(x_1, x_2) = \text{XOR}\ (-\text{signo}(x_1) \cdot \mu(x_1), -\text{signo}(x_2) \cdot \mu(x_2))$

Para una determinada salida su valor instantáneo dependerá del mayor valor que puede tomar ésta y la superficie de control resultante:

$$\text{salida} = \text{salida}_{\text{maxima}} \cdot \mu(x_1, x_2)$$

Para no complicar las reglas de inferencia he omitido los puntos máximos.

Podemos utilizar directamente $\mu(x_1, x_2)$ si el sistema a controlar tiene como salida PWM, en el apéndice N se explica cómo implementarlo.

La operación valor máximo es equivalente a la operación lógica OR y la operación valor mínimo es equivalente a la operación lógica AND, la operación producto no tiene equivalente booleano. También se puede definir la operación suma, pero cómo su valor máximo será 2 habrá que dividirlo entre ese valor o truncarlo a uno con la función límite.

En el producto de las variables de entrada tenemos 2 superficies de control equivalentes, las número 6 y 9 dan el mismo resultado final, y también las 7 y 8; así que podemos prescindir de las reglas 8 y 9, por tanto hay 34 soluciones posibles con μ_L-μ_L para las entradas x_1 y x_2, otras tantas para μ_C-μ_C y lo mismo[2] para la combinación μ_L-μ_C. En total hay; para dos universos de discurso de entrada y un único universo de discurso de salida; 108 patrones diferentes de programación lingüística, siempre y cuando esté sólo en función de las 4 operaciones analizadas; el producto, el valor máximo, el valor mínimo y XOR difuso. En el apéndice L se muestra la forma generalizada para cualquier operación con dos entradas y una salida.

Se puede ampliar a más entradas y salidas e incrementar las operaciones cuanto se quiera, tales cómo la implicación difusa, la media aritmética del mínimo y el máximo de las variables de entrada, la equivalencia borrosa u otra relación difusa definida por el usuario; ver el apéndice J.

De todas las soluciones posibles son pocas las que efectivamente serán útiles para un sistema de control, **si cada universo de discurso es el error** entre la variable consignada y la variable de entrada **la función a**

2 Lo mismo da que $\mu(x_1)$ y $\mu(x_2)$ sean de la forma μ_L-μ_C que μ_C-μ_L.

utilizar será casi siempre lejos(x) para cada dominio; si además la salida tiene signo habrá que realizar un control teniendo en cuenta el mismo.

error = variable consignada - variable de entrada

De esta forma pasamos de tener 108 patrones diferentes de programación lingüística a sólo 34 soluciones posibles con μ_L-μ_L para las entradas x_1 y x_2. Aunque los otros patrones habrá que seguir teniéndolos en cuenta para otras posibles combinaciones de funciones de pertenencia.

Cuando el error sea positivo habrá que realizar; casi siempre; un control en el mismo sentido al signo del error, puesto que la variable consignada será mayor que la variable de entrada.

La función cerca(x) se podrá utilizar en aquellos sistemas en los que hayan fuerzas externas al mismo que lo desvíen del valor máximo, cómo por ejemplo un sistema de carga colgante; ya que este tipo de control requiere para una oscilación cero $\mu_C(0) = 1$ y para una oscilación grande $\mu_C(x_{max}) = 0$; el cambio de carril involuntario de un vehículo, etc.

Las reglas de inferencia se asemejan a un sistema combinacional de puertas lógicas, con la salvedad de que en **ALBHI** tenemos entradas y salidas difusas.

En el apéndice B se representan algunas de las reglas de inferencia en forma de superficies de control, y en el apéndice C los programas correspondientes en Python.

9. ANTIIMAGEN DEL GRADO DE PERTENENCIA.

A diferencia del anterior sistema de inferencia éste se parece un poco más al de la lógica difusa ya que en primer lugar se obtiene el grado de

pertenencia del conjunto de las entradas y aplicando ese grado al conjunto de las salidas obtener el consecuente.

Primero se enlaza o empareja los elementos de un dominio o universo de discurso X con elementos del intervalo [0, 1]:

$$A : X \rightarrow [0,\ 1]$$

y después se enlaza o empareja los elementos del intervalo [0, 1] resultante del universo de discurso X con los elementos de un dominio o universo de discurso Y:

$$B : [0,\ 1] \rightarrow Y$$

Por tanto establecemos los conceptos antilejos y anticerca:

$$\text{antilejos}(x) = \frac{\text{arctanh}(\mu(x))}{\text{pmL}\ y}$$

$$\text{anticerca}(x) = \frac{\text{arctanh}(1 - \mu(x))}{\text{pmC}\ y}$$

donde 'y' es la salida del universo de discurso, μ(x) es la función de pertenencia de la entrada, y pueden ser $\mu_L(x)$ o $\mu_C(x)$.

Por consiguiente hay dos posibles combinaciones en las variables lingüísticas de la salida.

Salida lejos. Posiblemente será la más utilizada porque establecemos el universo de discurso un grado cero con valor cero y un grado 1 con valor máximo de salida, pudiendo ser la entrada cerca o lejos, además de multiplicar por 0,999 sólo para las simulaciones por lo expuesto en párrafos anteriores, por tanto la ecuación queda:

$$y = \min(\text{antilejos}(x),\ \text{ymax})$$

Salida cerca. Ocurre lo opuesto al anterior por tanto la ecuación queda:

$$y = \max(\text{anticerca}(x),\ 0)$$

Para varias entradas y salidas establecemos las mismas reglas de inferencia que en el capítulo 8, siendo x_1 y x_2 los universos de entrada de un sistema y $\mu(x_1, x_2)$ su grado de pertenencia. Los conceptos antilejos y anticerca para dos variables de entrada teniendo en cuenta el grado de pertenencia $\mu(x_1, x_2)$ son:

$$\text{antilejos}(x_1,\ x_2) = \frac{\text{arctanh}(\mu(x_1,\ x_2))}{\text{pmL}\ y}$$

$$\text{anticerca}(x_1,\ x_2) = \frac{\text{arctanh}(1 - \mu(x_1,\ x_2))}{\text{pmC}\ y}$$

y sus respectivas salidas:

$$y = \min(\text{antilejos}(x_{1,}\ x_2),\ \text{ymax})$$

$$y = \max(\text{anticerca}(x_{1,}\ x_2),\ 0)$$

Para multiplicar por 0,999 hay que hacerlo en cada grado de pertenencia individualmente.

No he utilizado esta forma en los ejemplos ya que multiplicar por el grado de pertenencia es más sencillo y más intuitivo, y no recomiendo utilizarlo para no complicar los modelos de control.

Con esto desaparece el problema de definir los conjuntos borrosos de cada variable medida y controlada, simplificando la tarea controlar un sistema.

La función de pertenencia μ_L tiene parentesco con la función σ y tanh de las redes neuronales; ver apéndice F; en dónde se utiliza cómo función de

activación en un sistema de ecuaciones lineales. Sin embargo cómo acabamos de ver en las reglas de inferencia no hace falta constreñirse a éste, puesto que ampliar a otras operaciones matemáticas tenemos más versatilidad sin que necesitemos aumentar las capas para obtener el mismo resultado.

ALBHI supone un avance extraordinario en la teoría de control, ya que con un algoritmo sencillo y potente podemos controlar cualquier variable tanto física como lógica, superando así al PID y a la lógica difusa. Pudiéndose aplicar a la resolución de cualquier sistema de control multivariable.

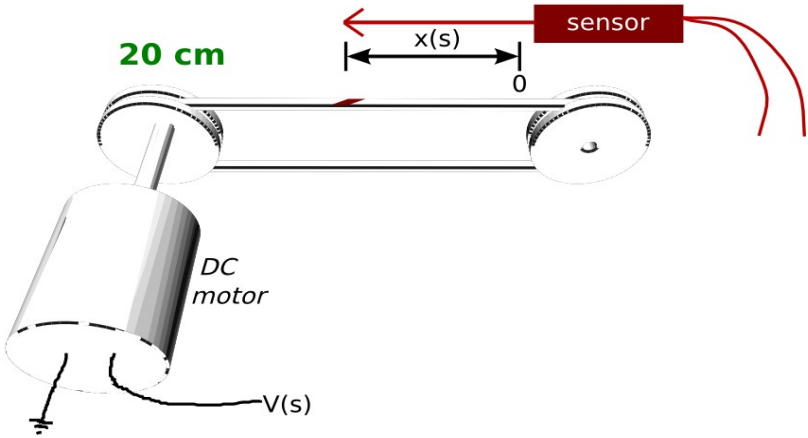
A continuación vienen cuatro ejemplos diferentes de los sistemas de control: control de posición de un cartucho de tinta en una impresora, calefacción de una habitación mediante una estufa eléctrica de 1.500 W, control de un motor DC y simulación de un carro con péndulo invertido.

10. CONTROL DE POSICIÓN DE UN CARTUCHO DE TINTA EN UNA IMPRESORA.

El problema a solucionar es la posición de un cartucho de tinta en una impresora.

La distancia se mide en metros y la máxima distancia a recorrer será 0,20 m ó 20 cm.

En la imagen se expone el mecanismo.



Y la ecuación de planta del sistema es:

$$\frac{x(s)}{V(s)} = \frac{s + 1}{s^2 + 3 \times 10^{-5} s + 2 \times 10^{-10}} .$$

error = variable consignada - distancia a recorrer

la variable consignada es la posición donde se situará el carro, que será cero cuando error = 0 y máxima en ±20 cm, por tanto la variable lingüística

es lejos, y su función de pertenencia $\mu_L(\text{error}) = \text{lejos}(\text{pmL} \cdot \text{error})$ para $\text{lejos}(x) = \min(\frac{|\tanh(x)|}{0{,}999}, 1)$.

La posición consignada se ha de alcanzar lo más rápido posible y la velocidad deberá disminuir al acercarse a la posición requerida, por ejemplo a la mitad cuando falte un centímetro; $\mu_L(0{,}01) = 0{,}500$.

Por el apéndice D el valor de la función de pertenencia es 1 en 0,07 m ó 7 cm.

Usaremos $\text{pmL} = \frac{0{,}549}{x}$ para x = 0,01; siendo pmL = 54,9.

$$\mu_L(\text{error}) = \text{lejos}(54{,}9 \cdot \text{error})$$

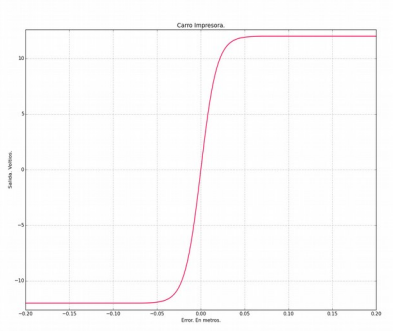
La regla de inferencia a utilizar será la del signo por la variable lingüística.

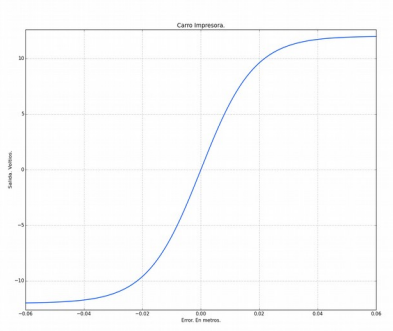
$$\mu(\text{error}) = \text{signo}(\text{error}) \cdot \mu_L(\text{error})$$

La variable a controlar es el voltaje del motor, siendo éste de 12 V máximo.

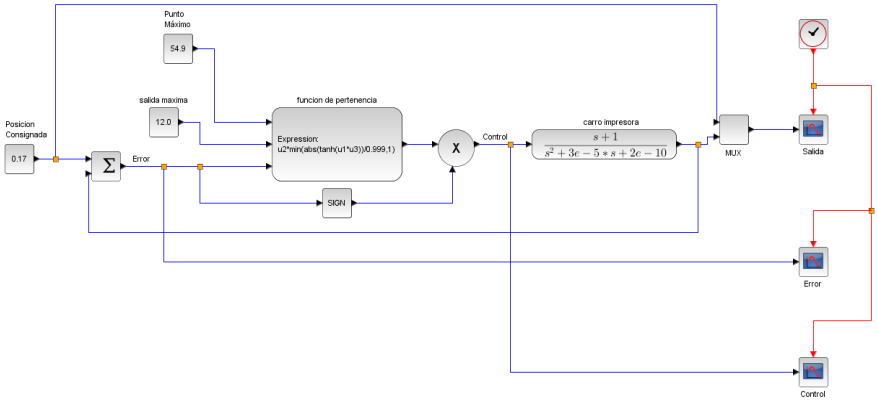
$$\text{voltaje} = 12 \cdot \mu(\text{error})$$

La representación de la variable de control se da a continuación.



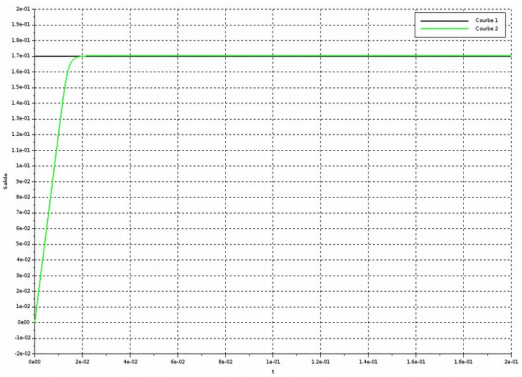


Vamos a simular el sistema con Xcos de Scilab 5.5.0.



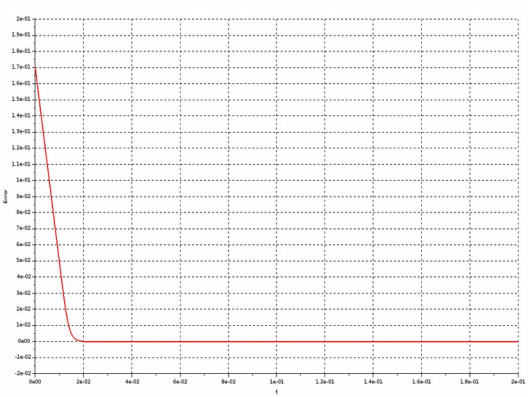
Para una posición consignada de 0,17 m tenemos las siguientes gráficas de la salida.

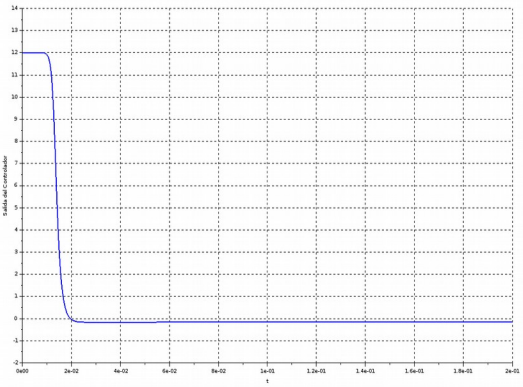
En negro el valor consignado y en verde la posición del carro.



Se observa claramente que hasta que no está cerca del objetivo el desplazamiento es constante, y aminora la velocidad cuando queda poco para llegar a la posición consignada.

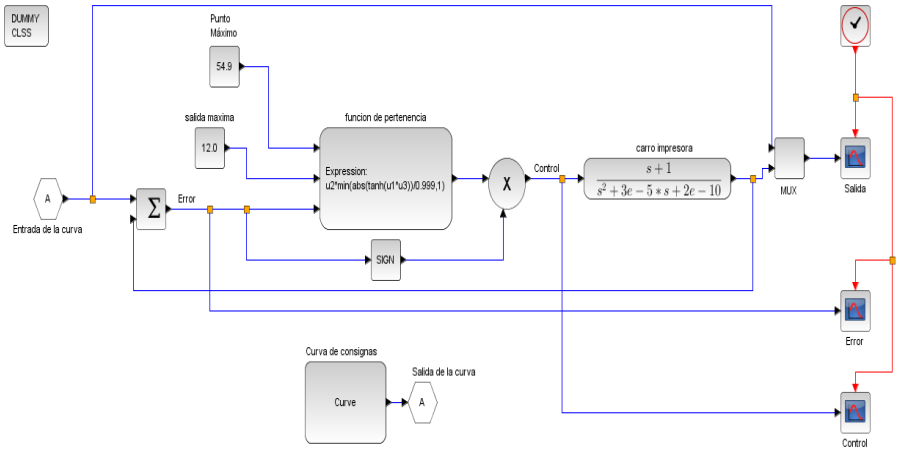
Vemos que el sistema de control tarda 20 ms en alcanzar la posición deseada, por tanto es un sistema eficiente, fácil de implementar y bajo costo en programación.



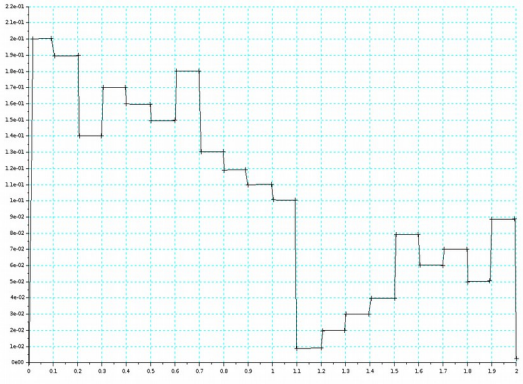


La salida del controlador es suave y continua, que es lo buscado en cualquier sistema de control.

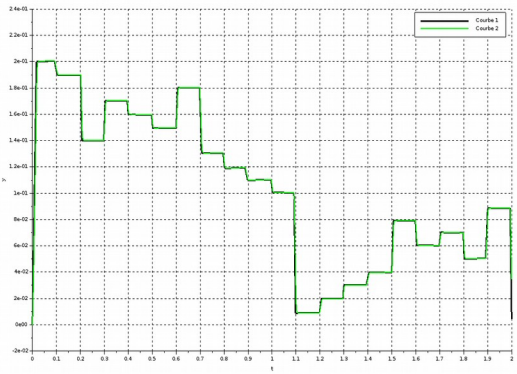
Vamos a poner **ALBHI** a prueba, en la siguiente imagen se muestra la simulación para distintos niveles de entrada.



La entrada es una curva de diferentes consignas tal y cómo se muestra en la figura de la derecha, éstas están comprendidas entre 0 m y 0,20 m para la entrada y cada una dura 100 ms, durante 2 segundos.

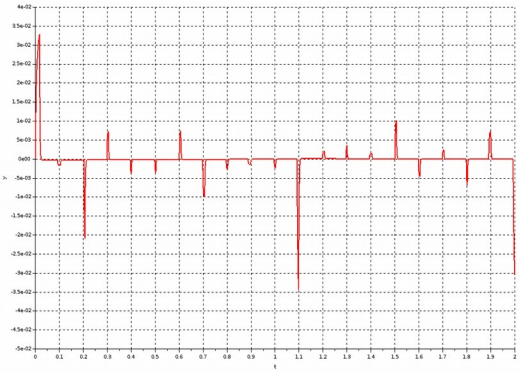


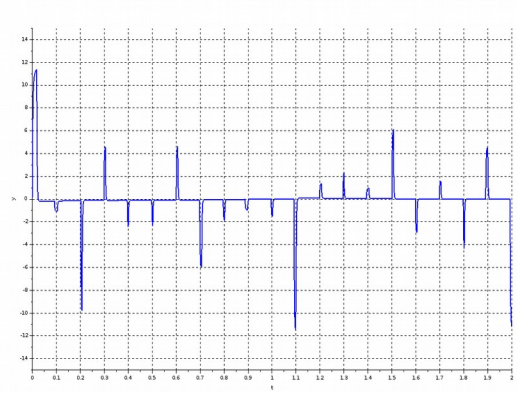
Las gráficas de salida son las siguientes.



En negro los valores consignados y en verde la posición del carro. Se ve claramente lo rápido que alcanza el cartucho la posición referenciada.

Se aprecia que una vez alcanzada la posición el error permanece en cero.





La salida del controlador es impecable, cumple todas las expectativas de cualquier sistema de control.

11. CALEFACCIÓN DE UNA HABITACIÓN MEDIANTE UNA ESTUFA ELÉCTRICA DE 1.500 W.

Hay que solucionar el problema de la calefacción para una habitación de un apartamento mediante una estufa eléctrica de 1.500 w.

La ecuación de planta de la habitación es $\frac{T(s)}{Q(s)} = \frac{0{,}194}{6490 \cdot s + 1}$. La temperatura de consigna será de 21 ºC con 10 ºC al comienzo.

error = temperatura consignada – temperatura medida

error = 21 – temperatura medida

La variable consignada es la temperatura, que será cero cuando error = 0 y máxima mientras no se alcance, por tanto la variable lingüística es lejos, y su función de pertenencia $\mu_L(\text{error}) = \text{lejos}(\text{pmL} \cdot \text{error})$ para $\text{lejos}(x) = \min(\frac{|\tanh(x)|}{0{,}999}, 1)$.

Para una diferencia de 0,5º C el sistema ha de ir disminuyendo su potencia, así que el grado de pertenencia es $\mu_L(0{,}5) = 0{,}999$.

Usaremos $\text{pmL} = \frac{3{,}8}{x}$ para x = 0,5; siendo pmL = 7,6.

$$\mu_L(\text{error}) = \text{lejos}(7{,}6 \cdot \text{error})$$

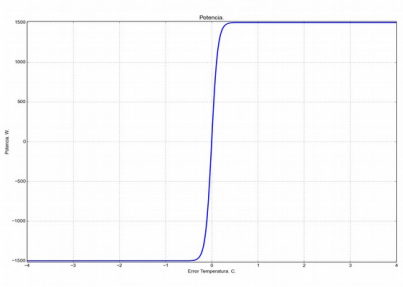
La regla de inferencia a utilizar será la del signo por la variable lingüística.

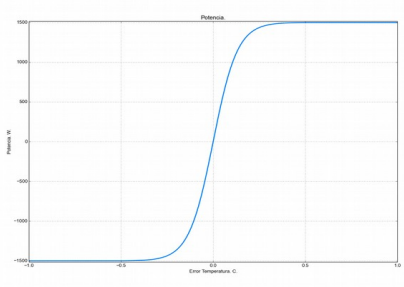
$$\mu(\text{error}) = \text{signo}(\text{error}) \cdot \mu_L(\text{error})$$

La variable a controlar es la potencia de la estufa, siendo ésta de 1.500 w máximo.

$$potencia = 1500 \cdot \mu(error)$$

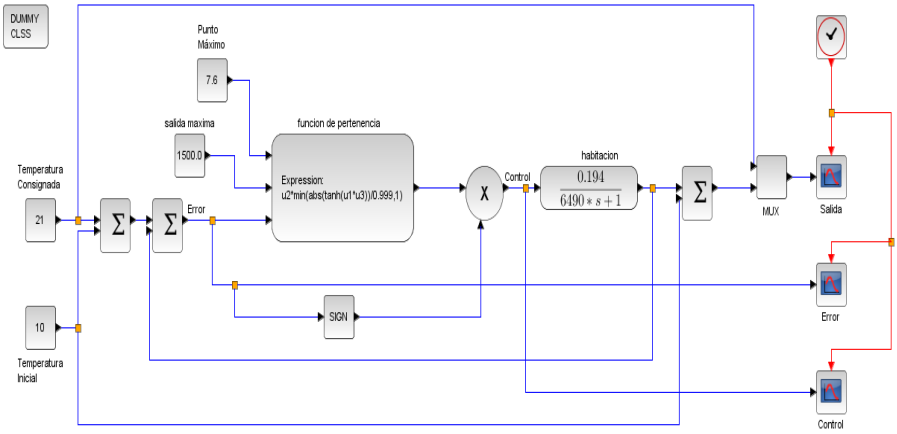
La representación de la variable de control se da a continuación.



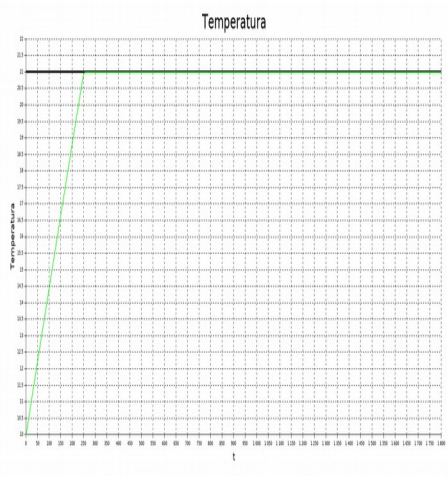


Aunque en las gráficas está representado un error negativo en la realidad no va a ser así, puesto que estamos tratando de un sistema calefacción y no de aire acondicionado; por tanto debemos tener en cuenta el cuadrante positivo.

Vamos a simular el sistema con Xcos de Scilab 5.5.0.



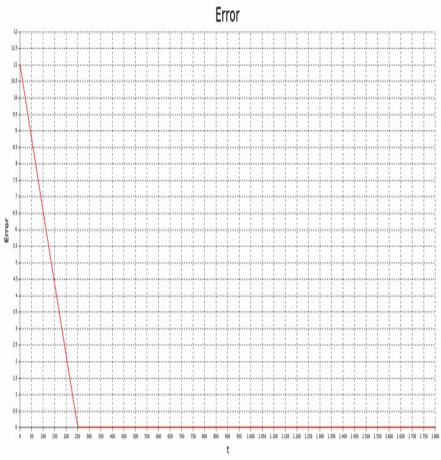
Para una temperatura consignada de 21ºC y temperatura inicial de 10ºC tenemos las siguientes gráficas de la salida.

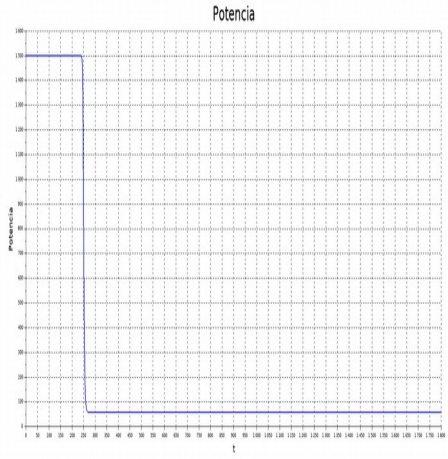


En negro el valor consignado y en verde la temperatura de la estancia.

Se observa claramente que hasta que no está cerca del objetivo el aumento de la temperatura es constante, y aminora el incremento cuando queda poco para llegar a la temperatura consignada.

Vemos que el sistema de control tarda 250 s; 4 m 10 s; en alcanzar la temperatura deseada, por tanto es un sistema eficiente, fácil de implementar y bajo costo en programación.





La salida del controlador es suave y continua, que es lo buscado en cualquier sistema de control.

Se observa que mientras no se alcanza la temperatura deseada la estufa está funcionando a la máxima potencia, para después reducirla y mantenerse constante para igualar las pérdidas.

12. CONTROL DEL MOTOR DC BN12-28.

Vamos a controlar la velocidad del motor de corriente continua BN12-28. La velocidad angular máxima es de 998 rad/s, así que ésta será la única variable a tener en cuenta.

$$\text{error} = 998 - \text{velocidad angular medida}$$

La variable consignada es la velocidad angular, que será cero cuando error = 0 y máxima mientras no se alcance, por tanto la variable lingüística es lejos, y su función de pertenencia $\mu_L(\text{error}) = \text{lejos}(\text{pmL} \cdot \text{error})$ para $\text{lejos}(x) = \min(\frac{|\tanh(x)|}{0{,}999}, 1)$.

Para una diferencia de 0,38 rad/s el sistema ha de ir disminuyendo su voltaje, así que el grado de pertenencia es $\mu_L(0{,}38) = 0{,}999$.

Usaremos $\text{pmL} = \frac{3{,}8}{x}$ para x = 0,38; siendo pmL = 10.

$$\mu_L(\text{error}) = \text{lejos}(10 \cdot \text{error})$$

La regla de inferencia a utilizar será la del signo por la variable lingüística.

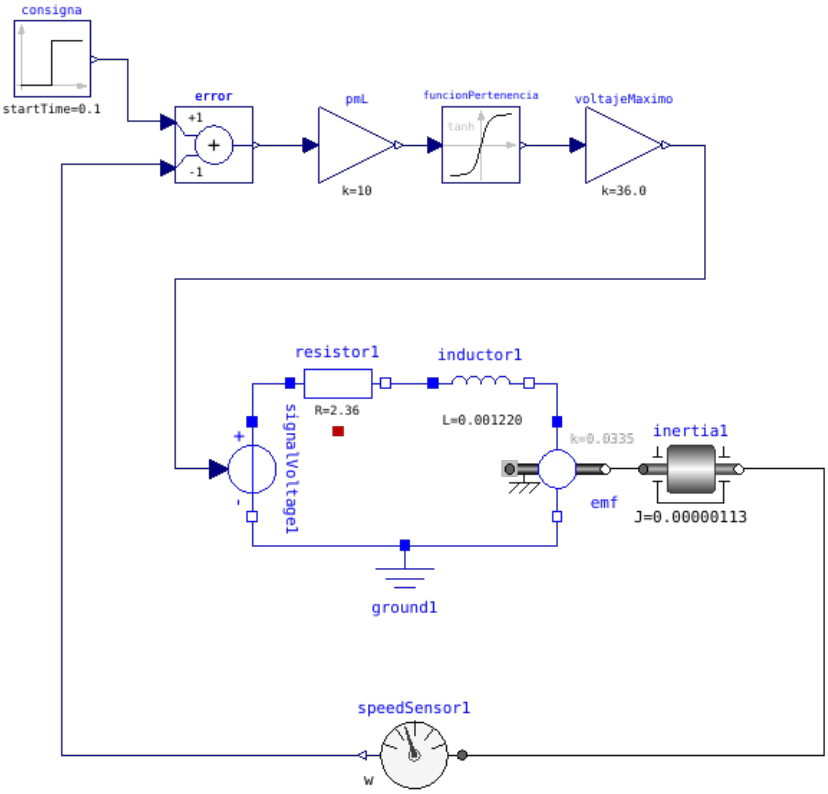
$$\mu(\text{error}) = \text{signo}(\text{error}) \cdot \mu_L(\text{error})$$

La variable a controlar es el voltaje del motor, siendo éste de 36 V máximo.

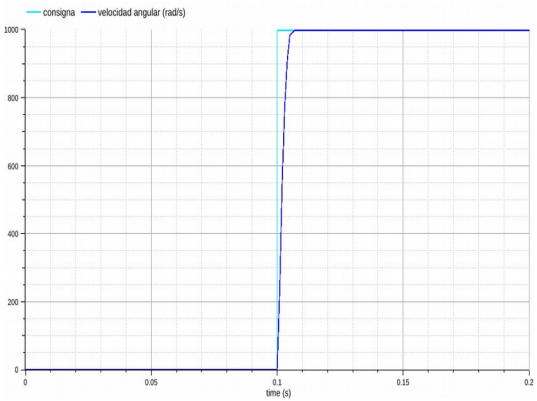
$$\text{señal} = 36 \cdot \mu(\text{error})$$

Las características principales del motor son; 2,36 Ω de impedancia; 1,22 mH de inductancia, un momento de inercia de 11,3 $g \cdot cm^2$ y una f.e.m de 0,0335 Nm/A.

La simulación se llevará a cabo con Open Modelica y el modelaje se hará con OMEdit.

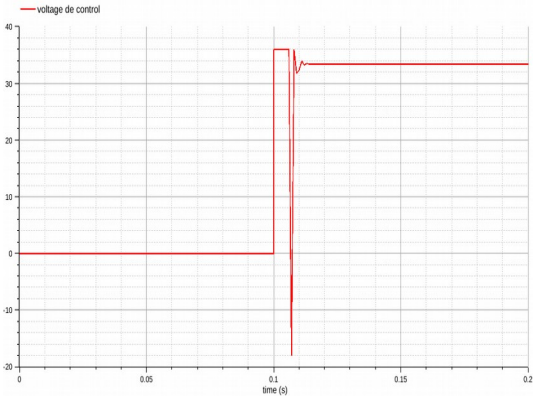


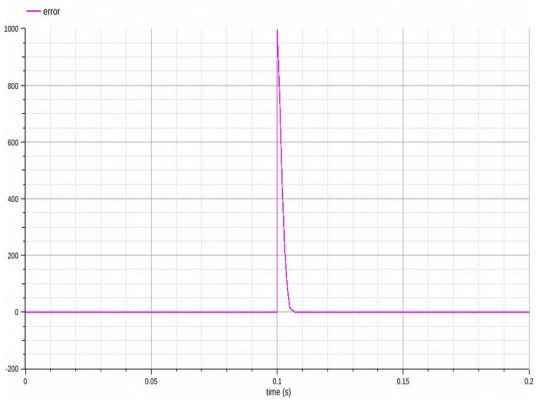
Para el valor consignado de 998 rad/s tenemos tres gráficas de datos, la salida, el error y el control.



El escalón se produce 0,1 segundos desde el inicio, y en menos de 0,01 segundos llega al valor consignado.

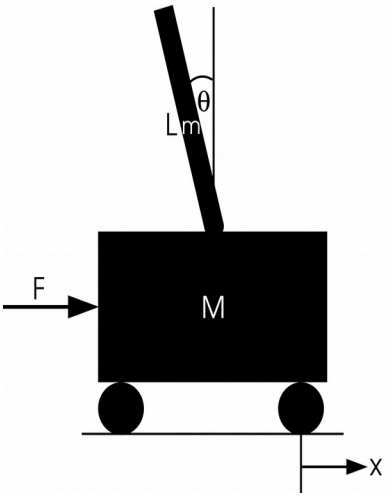
El control es efectivo y preciso, en él podemos apreciar que tipo de control hace y cuánto tiempo tarda en llegar al valor consignado.





En el error podemos apreciar con mejor detalle la breve duración en alcanzar el objetivo.

13. SIMULACIÓN DE UN CARRO CON PÉNDULO INVERTIDO.



M; Masa del carro.
m; Masa del péndulo.
b; Fricción del carro.
L; Longitud al centro de masa del péndulo.
I; Inercia del péndulo.
F; Fuerza aplicada al carro.
x; Coordenadas de posición del carro.
θ; Ángulo del péndulo respecto de la vertical.

Hay que mantener un péndulo en posición vertical con el eje de giro en la parte inferior y solidario a un carro. En otras palabras, la idea es encontrar la fuerza que ha de aplicarse al carro para que el péndulo no se caiga, incluso si se le perturba con un empujón.

Las variables de entrada a tener en cuenta son el ángulo respecto de la vertical y la velocidad lineal del carrito.

El $error_1$ va a ser el ángulo respecto de la vertical, θ, pero como los simuladores miden el ángulo desde cero el error es,

$$error_1 = \theta = \pi/2 - \varphi$$

donde φ es el ángulo respecto de la horizontal.

Para un $error_1 = 0$ no actúa el sistema de control y para un error mayor la acción de control ha de ser mayor, así que la función de pertenencia será

$$\mu_L(error_1) = lejos(pmL \cdot error_1) \text{ para } lejos(x) = \min(\frac{|\tanh(x)|}{0{,}999}, 1).$$

Para un ángulo $\varphi = 0{,}24$ rad el grado de pertenencia es $\mu_L(0{,}24) = 0{,}999$.

Usaremos $pmL = \frac{3{,}8}{x}$ para $x = 0{,}24$; siendo $pmL = 16$.

$$\mu_L(error_1) = lejos(16 \cdot error_1)$$

El $error_2$ va a ser la velocidad lineal,

$$error_2 = 0 - v$$

Para un $error_2 = 0$ no actúa el sistema de control y para un error mayor la acción de control ha de ser menor, así que la función de pertenencia será

$$\mu_L(error_2) = lejos(pmL \cdot error_2) \text{ para } lejos(x) = \min(\frac{|\tanh(x)|}{0{,}999}, 1).$$

Para una velocidad de −7,17 m/s el grado de pertenencia es $\mu_L(-7{,}17) = 0{,}999$.

Usaremos $pmL = \frac{3{,}8}{x}$ para $x = -7{,}17$; siendo $pmL = -0{,}53$.

$$\mu_L(error_2) = lejos(-0{,}53 \cdot error_2)$$

La regla de inferencia a utilizar será la 6 del promedio del máximo y el mínimo.

$$\mu_{MAX}(error_1, error_2) = \max(signo(error_1) \cdot \mu_L(error_1), signo(error_2) \cdot \mu_L(error_2))$$

$$\mu_{MIN}(error_1, error_2) = \min(signo(error_1) \cdot \mu_L(error_1), signo(error_2) \cdot \mu_L(error_2))$$

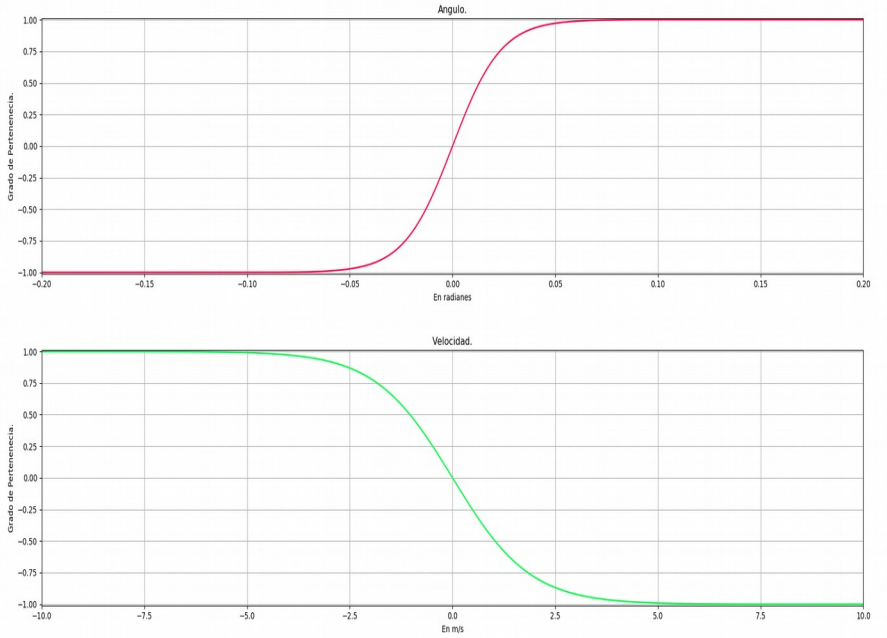
$$\mu(error_1, error_2) = \frac{\mu_{MAX}(error_1, error_2) + \mu_{MIN}(error_1, error_2)}{2}$$

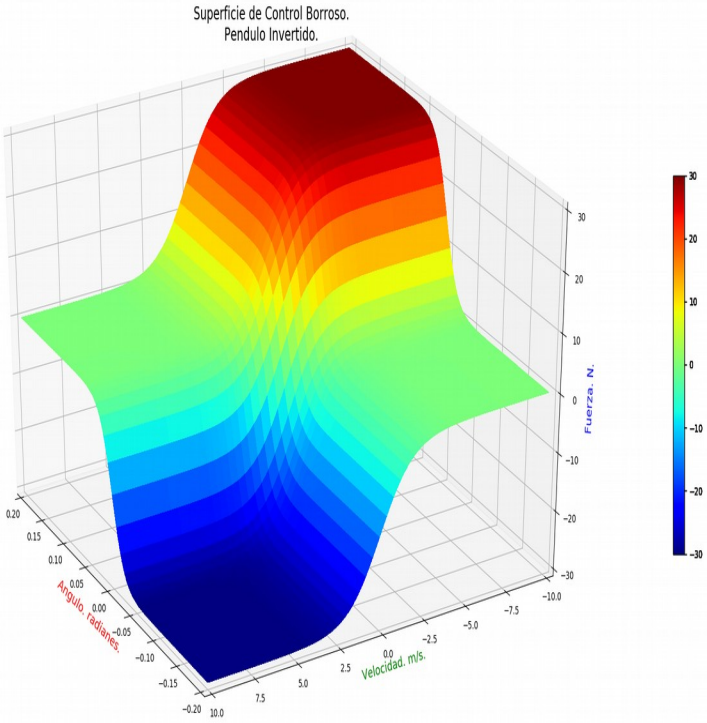
La variable a controlar es la fuerza del carro, siendo ésta de 30 N máximo.

$$fuerza = 30 \cdot \mu(error_1, error_2)$$

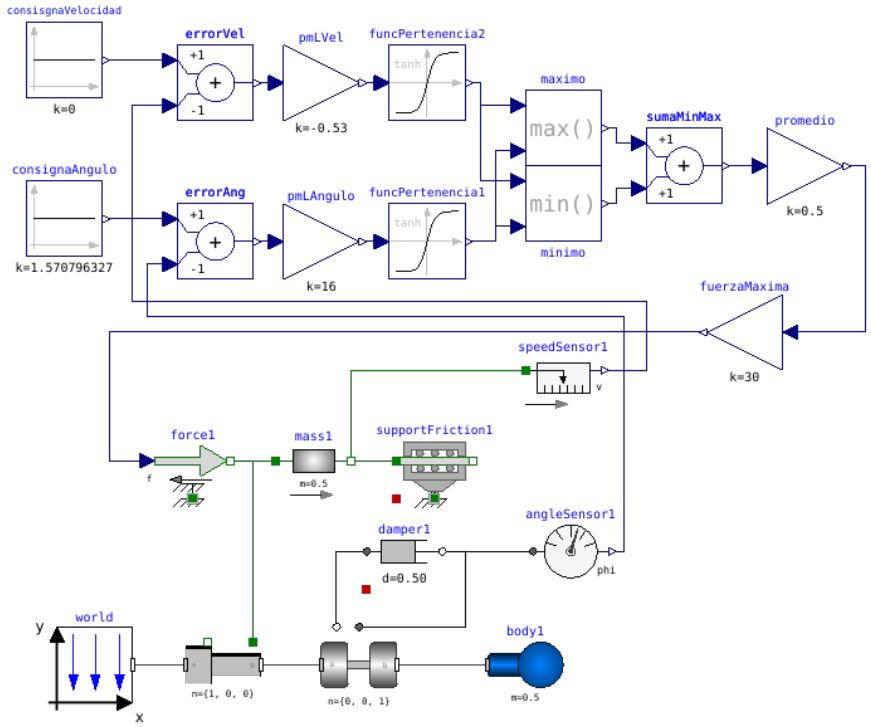
Los datos de entrada son: 0,500 Kg para la masa del carro y el péndulo; 0,100 m para la longitud del carro; 0,300 m para la distancia al centro de masas del péndulo; 0,100 N para la fricción del carro; 0,006 Kg·m² para la inercia del péndulo.

La representación de las variables de control se dan a continuación.



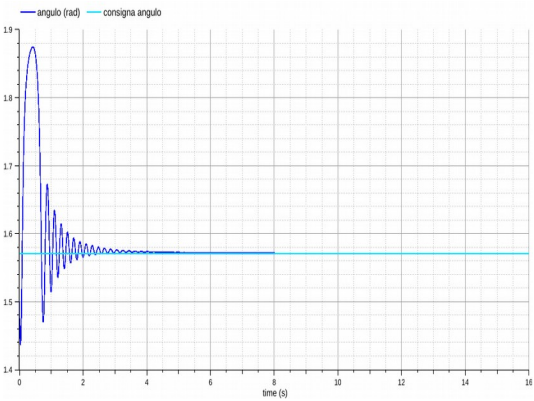

Superficie de Control Borroso.
Pendulo Invertido.
Fuerza. N.
Angulo. radianes.
Velocidad. m/s.

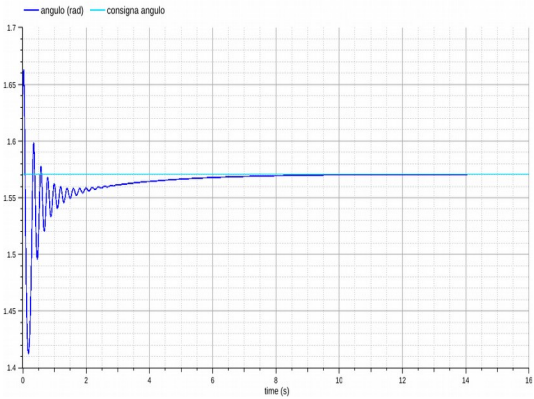
La simulación se llevará a cabo con Open Modelica y el modelaje se hará con OMEdit.



Para un ángulo consignado de π/2 rad y ángulos iniciales de 1,5 rad y 1,6416 rad (-1,5 rad) con velocidades angulares de -4 rad/s y 2 rad/s respectivamente tenemos las siguientes gráficas de la salida del ángulo, la fuerza, la velocidad y el error.

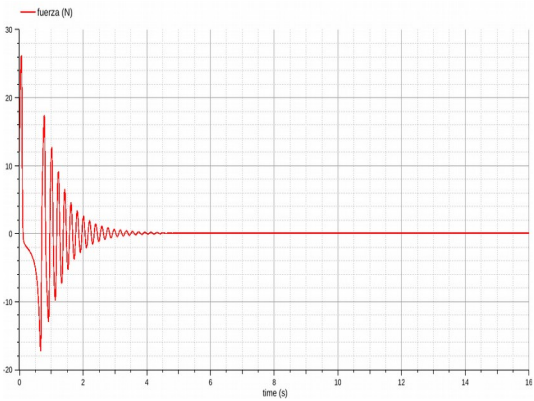
Para el ángulo inicial de 1,5 rad y velocidad angular de -4 rad/s se observa que en aproximadamente 8 segundos alcanza la posición vertical de π/2 radianes.

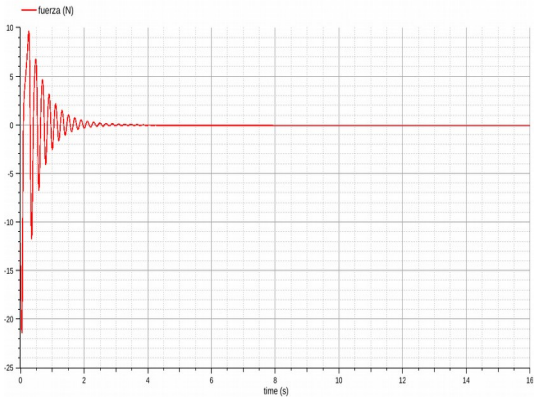




Para el ángulo inicial de 1,6416 rad y velocidad angular de 2 rad/s se observa que en aproximadamente 14 segundos alcanza la posición vertical de π/2 radianes.

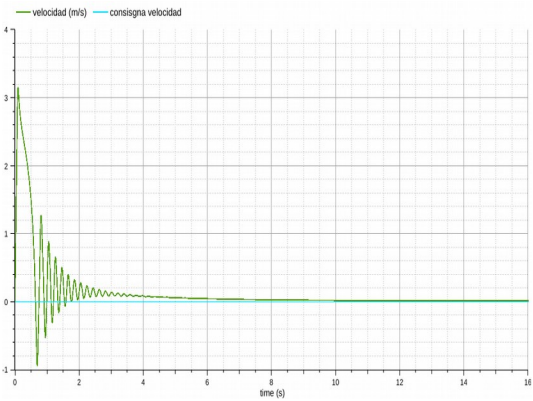
Para el ángulo inicial de 1,5 rad y velocidad angular de -4 rad/s se observa que la fuerza dura apenas dura 5 s.

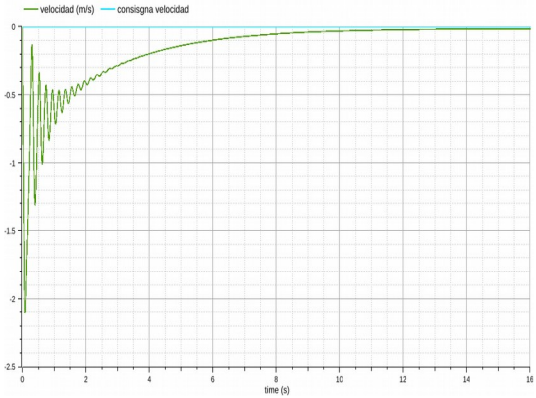




Para el ángulo inicial de 1,6416 rad y velocidad angular de 2 rad/s se observa que la fuerza dura apenas dura 5 s.

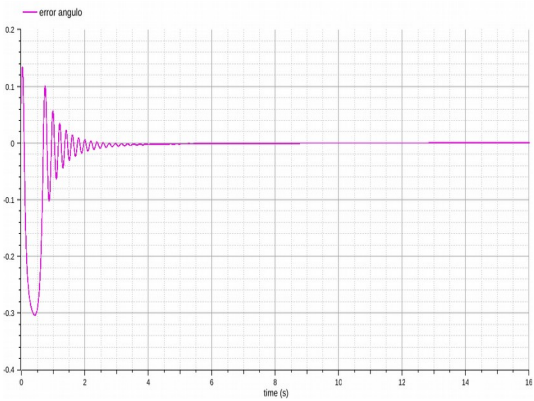
Para el ángulo inicial de 1,5 rad y velocidad angular de -4 rad/s se observa que la velocidad va disminuyendo hasta ser cero en 16 segundos.

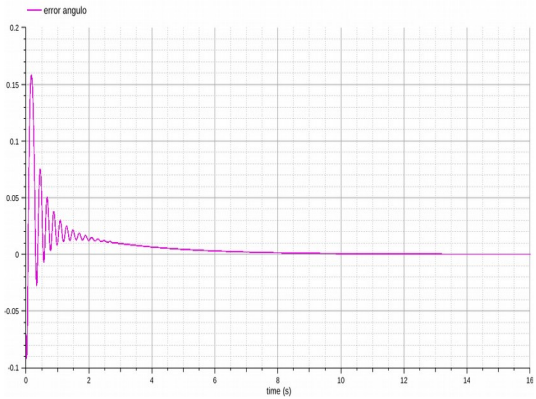




Para el ángulo inicial de 1,6416 rad y velocidad angular de 2 rad/s se observa que la velocidad va disminuyendo hasta ser cero en 16 segundos.

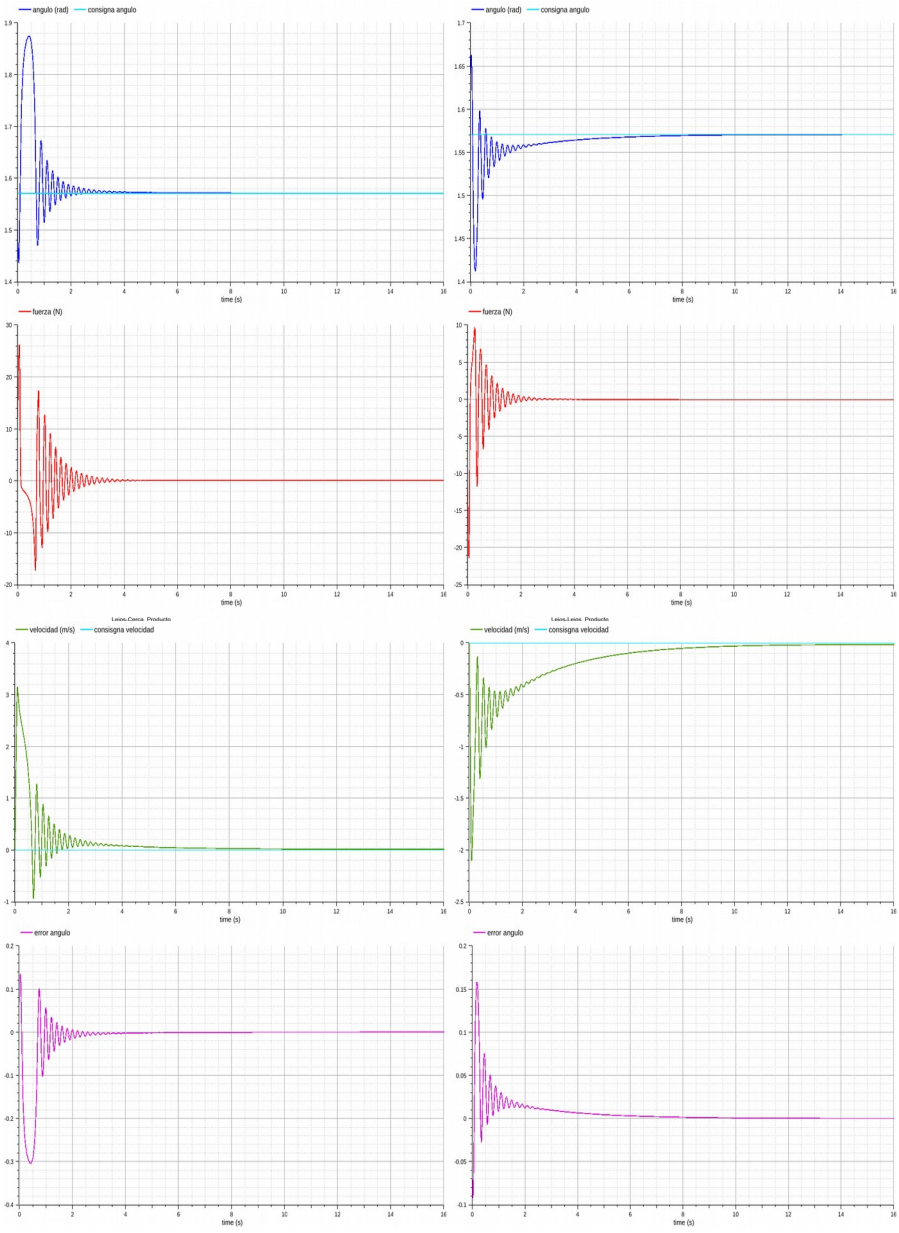
Para el ángulo inicial de 1,5 rad y velocidad angular de -4 rad/s se observa que el error en radianes se va haciendo cada menor hasta ser cero en 8 segundos.





Para el ángulo inicial de 1,6416 rad y velocidad angular de 2 rad/s se observa que el error en radianes se va haciendo cada menor hasta ser cero en 14 segundos.

Las siguientes gráficas se muestran el ángulo, fuerza, velocidad y error para los ángulos iniciales de 1,5 rad y 1,6416 rad.



Aplicarlo a un sistema de transporte individual basado en el péndulo invertido resulta ser una tarea trivial.

Hasta ahora la lógica difusa requería mayor cantidad de cálculos y pasos para realizar éste control difuso, en la que hay que definir cada conjunto borroso, hacer la borrosificación, después la inferencia mediante reglas SI-ENTONCES-, seguidamente la composición y finalmente la desborrosificación.

Sin embargo en **ALBHI** sólo hay que definir el error, ver la función de pertenencia necesaria, calcular el punto máximo y aplicar la regla; o conjunto de ellas; de inferencia requerida para la salida.

Con este último ejemplo termina la primera parte. Cómo ha podido deducir el lector con el sistema de control **ALBHI** se puede controlar cualquier tipo de máquina.

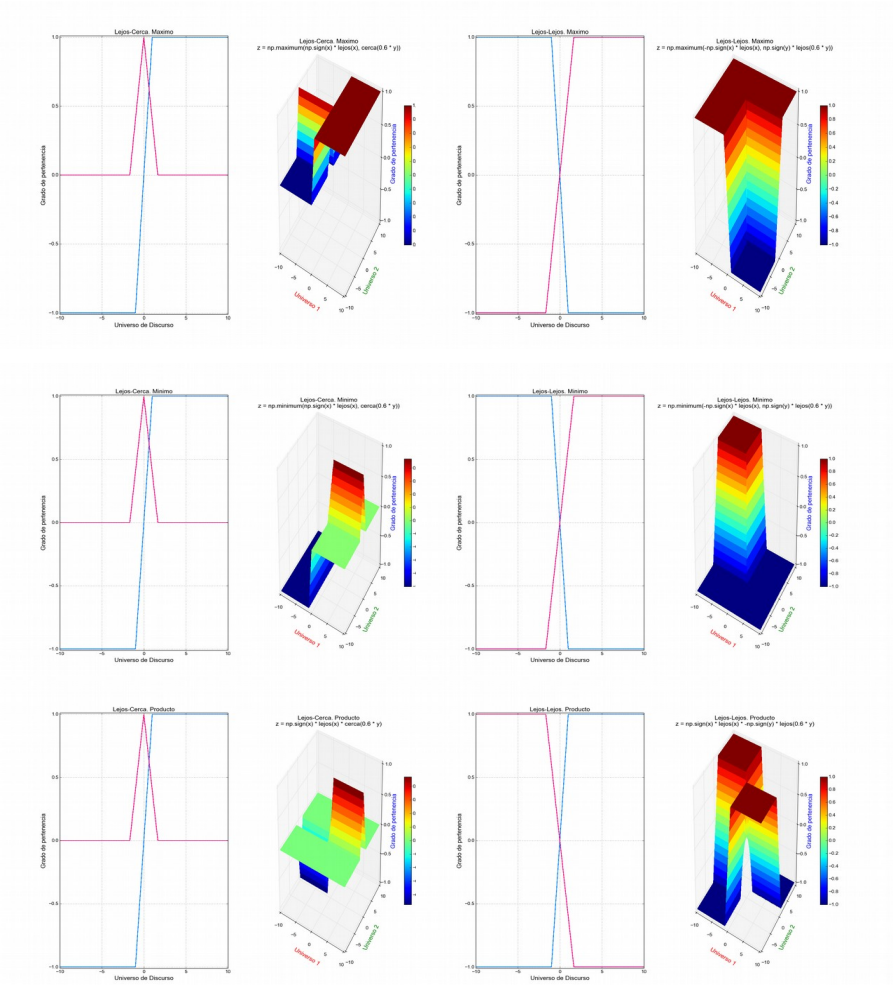
ALBHI supone un avance extraordinario en la teoría de control, ya que con un algoritmo sencillo podemos controlar cualquier variable tanto física como lógica. Superando al PID y a la lógica difusa.

PARTE 2.

DECISIÓN DIFUSA 2, UN NUEVO ENFOQUE

CONTROL DIFUSO POR VARIABLE LINGÜÍSTICA 2

Autor: José María Molina Sánchez.



“La fortuna es de las personas audaces.”

Virgilio.

14. MEJORANDO ALBHI.

Durante las simulaciones con OpenModelica y XCOS advertí que la función límite tiene un cierto parecido a la tanh, así que decidí sustituir la función para ver si el resultado que obtenía era similar.

En las siguientes páginas demuestro que este simple cambio mejora la decisión borrosa en el sentido de que no hay que definir ninguna función de pertenencia en el ámbito real aunque sí en las simulaciones, pues **la aritmética de saturación es la propia función límite y por tanto la función de pertenencia**.

Los ejemplos que hay en este apartado son los mismos que los anteriores, para que el lector pueda comparar los resultados.

15. CONJUNTO DIFUSO DE VARIABLE LINGÜÍSTICA.

Atendiendo solamente a la definición de conjunto difuso como una función de pertenencia que enlaza o empareja los elementos de un dominio o universo de discurso X con elementos del intervalo [0, 1]:

$$A : X \rightarrow [0,\ 1]$$

he ideado una forma más sencilla de hacer lo mismo que **ALBHI** con mucho menos coste de cálculo. A este método lo he llamado **ALBLI,** ALgoritmo Borroso Limite.

16. FUNCIÓN DE PERTENENCIA.

La idea es similar a **ALBHI**, a partir de cierto número el valor de la imagen o grado de pertenencia es 1, y si es menor aumenta de forma proporcional hasta ese valor. Para ello utilizo as funciones máximo y mínimo, que tiene validez solamente en las simulaciones.

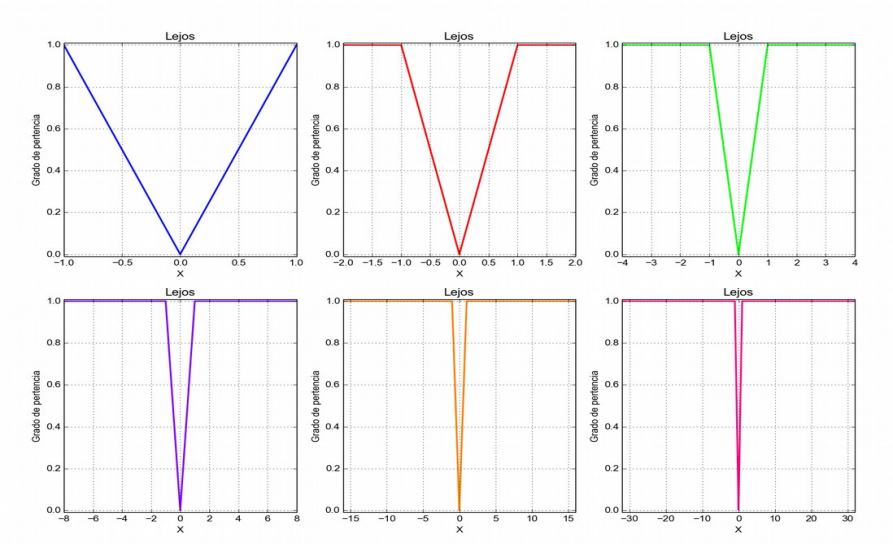
De esta forma la variable lingüística lejos(x) y su correspondiente función de pertenencia se definen:

$$\text{lejos}(x) = \text{minimo}(|x|,\ 1) \qquad \mu_L(x) = \text{lejos}(x)$$

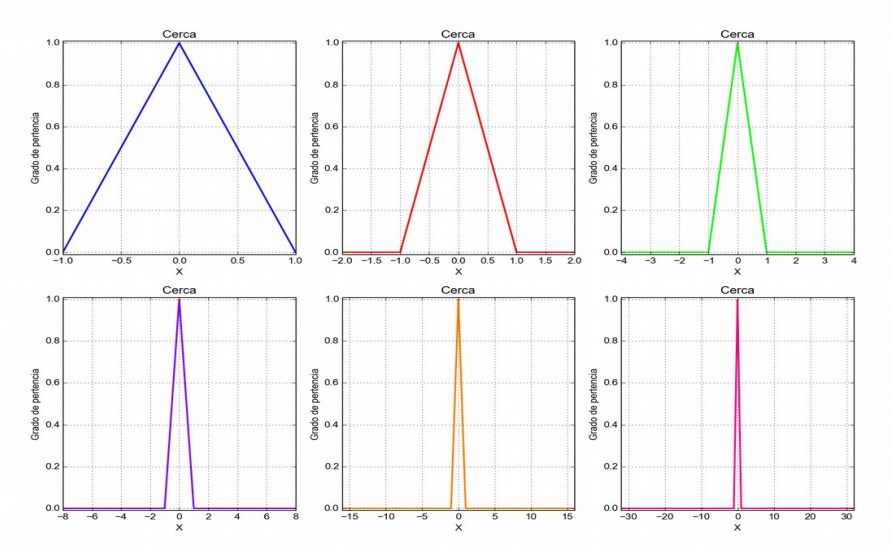
y para la variable lingüística cerca(x) se definen:

$$\text{cerca}(x) = \text{maximo}(1 - |x|,\ 0) \qquad \mu_C(x) = \text{cerca}(x)$$

A continuación se representa la función lejos(x) correspondientes a los dominios ±1, ±2, ±4, ±8, ±16 y ±32.



A continuación se representa la función cerca(x) correspondientes a los dominios ±1, ±2, ±4, ±8, ±16 y ±32.



17. PUNTO MÁXIMO.

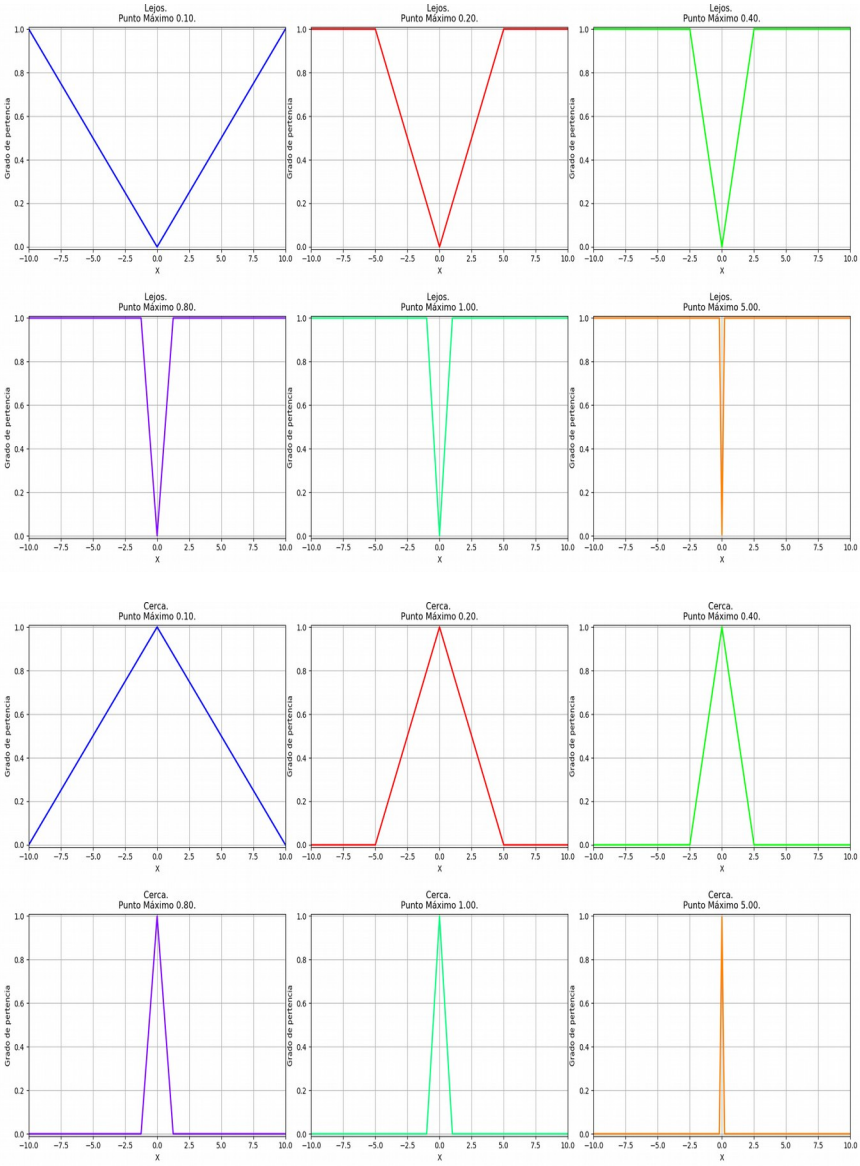
En **ALBLI** también se define el punto máximo de la misma manera que en **ALBHI**, pero aquí además indica la pendiente de la recta que queda entre 1 y -1, quedando finalmente para la variable lingüística lejos:

$$\mu_L(x) = \text{lejos}(\text{pmL} \cdot x)$$

y para la variable lingüística cerca:

$$\mu_C(x) = \text{cerca}(\text{pmC} \cdot x)$$

A continuación se representan los puntos máximos con los valores 0,1; 0,2; 0,4; 0,8; 1,0 y 5,0 para las funciones de pertenencia $\mu_L(x)$ y $\mu_C(x)$ en el dominio de $x \in [-10, 10]$.



Los grados de pertenencia $\mu_L(x)$ y $\mu_C(x)$ están definidos entre 0 y 1. Pero en cualquier sistema a controlar deberán estar entre -1 y 1 debiendo multiplicarlos por ±signo(x), quedando las funciones de pertenencia definidas de la siguiente manera:

$$\mu_L(x) = \pm\text{signo}(\text{pmL} \cdot x) \cdot \text{lejos}(\text{pmL} \cdot x)$$

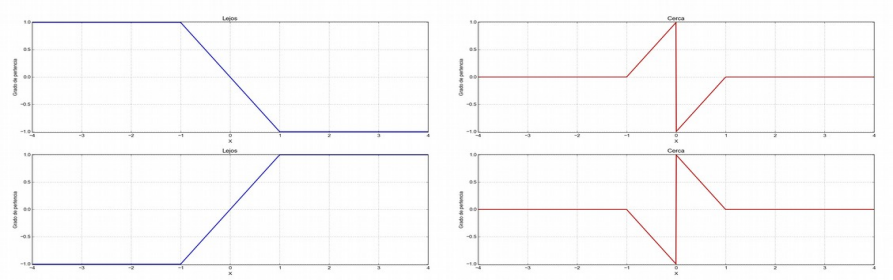
$$\mu_C(x) = \pm\text{signo}(\text{pmC} \cdot x) \cdot \text{cerca}(\text{pmC} \cdot x)$$

También podemos redefinir las variables lingüísticas como:

$$\text{lejos}(x) = \text{maximo}(\text{minimo}(x, 1), -1)$$

$$\text{cerca}(x) = \text{signo}(x) \cdot \text{maximo}(1 - |x|, 0)$$

La gráficas de los signos se representan a continuación.

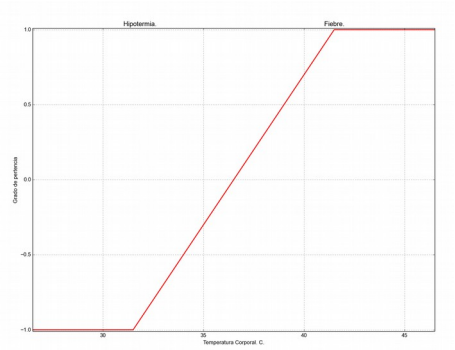


La función $\mu_L(x)$ es similar a la sigmoide pudiéndose considerar como la linealización de dicha función, que coincide con la función límite, y es más sencilla de manejar, ya que el grado de pertenencia es más fácil de obtener que en el caso de las funciones hiperbólicas. La función $\mu_C(x)$ sin embargo no tiene sentido, puesto que en cero tiene dos valores, así que no se puede utilizar en esta forma, pero sí en su forma original $\text{cerca}(x) = \text{maximo}(1 - |x|, 0)$ con su respectiva función de pertenencia $\mu_C(x) = \text{cerca}(\text{pmC} \cdot x)$.

Esta forma redefine la función complemento, si para valores del grado de pertenencia comprendidos entre 0 y 1 es 1 – μ(x), cuando los valores del grado de pertenencia están entre -1 y 1 el complemento es 0 – μ(x), ó -μ(x). Ello se aprecia claramente en la gráfica anterior.

Si el universo de discurso es el error y el punto máximo > 1, podemos considerar el dominio de éste en [-1, 1] como máximo.

Para el punto máximo con valores mayores que uno la función de pertenencia tiende a estrecharse, esto hará que el sistema de control tenga un funcionamiento rápido y abrupto. Cuando el punto máximo tiene valores menores que uno la función de pertenencia tiende a ensancharse, esto hará que el sistema de control tenga un funcionamiento lento y suave.



En el ejemplo de la temperatura corporal de **ALBHI** definimos el conjunto como Hipotermia-Fiebre, si aplicamos esto a **ALBLI** torna más sencillo. En 36,5 ºC el grado de pertenencia es 0, para temperaturas mayores de 41,5 ºC es 1, y para temperaturas menores de 31,5 ºC es -1. La función a utilizar será lejos(x) y el pmL = 0,200. Para la hipotermia usamos el grado de pertenencia menor que cero, que es lo contrario de la fiebre. Según el grado de pertenencia usaremos para temperaturas mayores de 36,5 ºC términos tales cómo “escasa fiebre”, “ligera fiebre”, “temperatura alta”, “bastante fiebre” y finalmente “mucha fiebre”. Para temperaturas menores en vez de fiebre nos referiremos a hipotermia con los mismos modificadores lingüísticos.

Aunque hemos usado la función límite en la definición de **ALBLI**, realmente no es necesario, puesto que para un registro de unos determinados bits, cuando llegue al máximo de éstos no podrá incrementarse más. Con **ALBLI la aritmética de saturación** deja de ser

un problema para convertirse en parte de **la solución**. Las variables lingüísticas y las funciones de pertenencia quedan así:

$$\text{lejos}(x) = x$$

$$\mu_L(x) = \text{lejos}(\pm pmL \cdot x)$$

$$\text{cerca}(x) = \text{signo}(n - |x|) \cdot (n - |x|)$$

$$\mu_C(x) = \text{cerca}(\pm pmC \cdot x)$$

Por lo dicho en párrafos más arriba μ_c no tiene sentido.

Y la salida será:

$$\text{salida} = \frac{\mu(x) \cdot \text{salida}_{maxima}}{n}$$

Donde salida_{maxima} es el valor máximo a controlar, y n es el valor máximo del registro; para m bits $n = 2^{m-1} - 1$ puesto que el BMS es el bit de signo. Si consideramos 8 bits con signo n = 127, ya que $\mu(x) \in [-128, 127]$ si el negativo se hace complemento a dos.

Para cerca(x) el valor absoluto se define poniendo el BMS a cero permanentemente, y la función signo(x) sólo con el BMS.

En caso de que el sistema produzca un error cuando se rebase el registro (overflow), habrá que definir una excepción al inicio del programa para que lo ignore.

Si utilizamos el estándar IEEE 754, basta una precisión de 2 bytes para controlar cualquier parámetro de cualquier sistema de control, ya que tiene 1 bit para signo, 5 bits para exponente y 10 bits para la fracción, con 15 bits para el Bias del exponente, de esta manera cubriríamos un rango de ±262.016 con pasos de $\pm 2{,}98 \cdot 10^{-8}$. También se puede modificar a nuestro

gusto con una FPGA, asignando distintos bits al exponente y a la fracción para que el control se ajuste a nuestras necesidades, si queremos que el control sea mas fino en un registro de 16 bits pondremos 1 bit para signo, 6 bits para exponente y 9 bits para la fracción, con 31 bits para el Bias del exponente, cubriendo un rango de $\pm 1{,}72 \cdot 10^{10}$ para controlar cualquier variable física o lógica, siendo el paso de $\pm 4{,}55 \cdot 10^{-13}$. Si queremos un rango y precisión menores podríamos asignar 1 bit al signo, 4 bits al exponente y 11 a la fracción, con 7 bits para el Bias del exponente, de esta manera tendríamos un rango de ±1.023,75 con una definición de $\pm 3{,}81 \cdot 10^{-6}$. El rango difuso está en ±1, por tanto debemos dividir el intervalo entre el máximo valor, 262.016 para el estándar IEEE 754 de 2 bytes.

Si el control se va a hacer desde un ordenador de 64 bits el rango y la precisión serán mucho mayores, y cómo acabamos de ver con 16 bits ya es más que suficiente, se pude escoger trabajar con menos bits poniendo en el programa el tipo de número con el que queremos trabajar, incluso actuar a nivel de bits; ya sea con 8 o 16; sin el estándar IEEE 754.

Los microcontroladores trabajan con 8 o 16 bits y tienen entradas digitales y analógicas, podemos concluir por tanto que para cualquier sistema de control los medios tecnológicos actuales son más que suficientes.

En la electrónica analógica tenemos otro ejemplo similar al de la aritmética de saturación, y lo tenemos en los amplificadores operacionales con las regiones de saturación positiva y negativa en los extremos y la región lineal en el centro, cuya curva es idéntica al la función límite. El punto máximo sería la ganancia del A.O. ya sea en amplificador inversor cómo no inversor, siendo el modo comparador con la entrada inversora a tierra el equivalente a la función signo. Para varias entradas utilizaríamos el sumador inversor seguido de otro inversor de ganancia unidad si queremos que la salida no sea inversa de la de entrada, donde la relación ente R_f y cada R_i sería la ponderación de cada entrada que es el equivalente al punto máximo.

Para ahorrarnos el segundo inversor en el sumador inversor el error habría que cambiarlo a: error = variable de entrada - variable consignada.

El cálculo de los puntos máximos se explica en el apéndice A.

18. REGLAS DE INFERENCIA.

A priori también hay dos formas de inferir los consecuentes con **ALBLI**; la primera es multiplicando el grado de pertenencia por el valor máximo de la variable a controlar y la segunda es la antiimagen del grado de pertenencia del conjunto de salida del grado de pertenencia del conjunto de entrada.

Sin embargo al analizar detalladamente ambas formas se llega a la conclusión de que son lo mismo.

Demostración:

Sea 'y' la salida, por tanto hay un 'ymax', así que $y = ymax \cdot \mu(x)$.

En la antimagen $pm\ y = \frac{1}{ymax} \Rightarrow ymax = \frac{1}{pm\ y}$ y $\mu(y) = \mu(x)$ por tanto $y = \frac{\mu(y)}{pm\ y} = \frac{\mu(x)}{pm\ y}$. Así que $y = ymax \cdot \mu(x)$.

Si queremos usar la antiimagen tendremos que utilizar conjuntamente **ALBHI** y **ALBLI**. Ya sea uno u otro el primero, aunque no es necesario por ser **ALBLI** más sencillo.

Al igual que en **ALBHI** ponemos dos entradas y una salida y establecemos las reglas de inferencia de las soluciones en los sistemas de control con cuatro[3] operaciones: el producto, el valor máximo, el valor mínimo y XOR difuso de las variables de entrada; incluyendo el signo de la función y sí hay o no valor opuesto del signo.

---

3 Se pueden implementar más operaciones para obtener el consecuente, cómo la implicación, la equivalencia u otra definida por nosotros.

Para X_1 y X_2 universos de entrada de un sistema y $\mu(x_1)$, $\mu(x_2)$ sus respectivas funciones de pertenencia. Las superficies de control o matrices de inferencia resultantes serán las mismas que en **ALBHI**, ver páginas 27 y 28, pero con la función límite en vez de la tangente hiperbólica (pág 115).

Con las operaciones difusas básicas $\min(\mu(x_1), \mu(x_2), \mu(x_3), ...)$, $\max(\mu(x_1), \mu(x_2), \mu(x_3), ...)$ y $1-\mu(x)$ que se corresponden con los operadores básicos AND, OR y NOT de la lógica booleana respectivamente se pueden formar sistemas combinacionales; ver apéndice J; ya sean síncronos o no, capaces de resolver cualquier problema que nos imaginemos con las variables ligüísticas lejos(x) y cerca(x). Además se pueden añadir a este sistema operaciones matemáticas tales cómo el producto, promedio, suma truncada, potencia, etc.

Cómo acabamos de ver en las reglas de inferencia no hace falta constreñirse a un sistema de ecuaciones lineales cómo en las redes neuronales artificiales, puesto que ampliar a otras operaciones matemáticas y/o lógicas tenemos más versatilidad sin que necesitemos aumentar las capas para obtener el mismo resultado.

Podemos utilizar microcontroladores y Arduino con **ALBLI** para resolver cualquier control, pero si utilizamos FPGA entonces las soluciones no tienen límites. En el capítulo 25 desarrollo esta idea.

ALBLI supone un avance más extraordinario que **ALBHI** en la teoría de control, ya que con un algoritmo todavía más sencillo y potente que era considerado un problema inherente al la computación, podemos controlar cualquier variable tanto física como lógica, superando así al PID y a la lógica difusa.

En el apéndice B se representan algunas de las reglas de inferencia en forma de superficies de control, y en el apéndice C los programas correspondientes en Python.

A continuación vienen cuatro ejemplos diferentes de los sistemas de control: control de posición de un cartucho de tinta en una impresora,

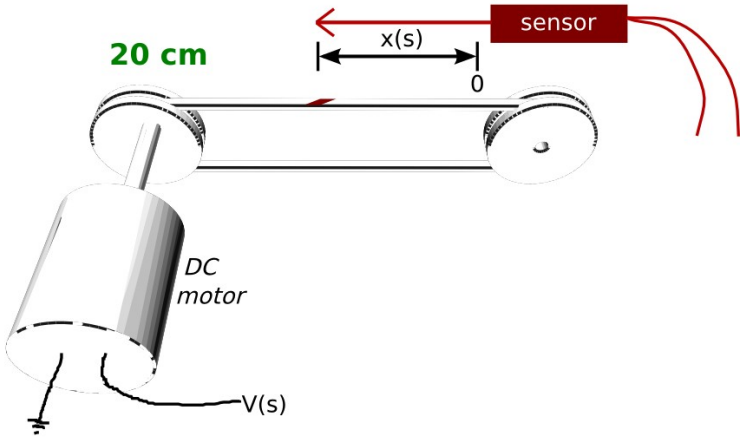
calefacción de una habitación mediante una estufa eléctrica de 1.500 W, control de un motor DC y simulación de un carro con péndulo invertido.

19. CONTROL DE POSICIÓN DE UN CARTUCHO DE TINTA EN UNA IMPRESORA.

El problema a solucionar es la posición de un cartucho de tinta en una impresora.

La distancia se mide en metros y la máxima distancia a recorrer será 0,20 m ó 20 cm.

En la imagen se expone el mecanismo.



Y la ecuación de planta del sistema es

$$\frac{x(s)}{V(s)} = \frac{s + 1}{s^2 + 3 \times 10^{-5} s + 2 \times 10^{-10}} .$$

error = variable consignada - distancia a recorrer

la variable consignada es la posición donde se situará el carro, que será cero cuando error = 0 y máxima en ±20 cm, por tanto la variable lingüística es lejos, y su función de pertenencia $\mu_L(\text{error}) = \text{lejos}(\text{pmL} \cdot \text{error})$ para $\text{lejos}(x) = \text{maximo}(\text{minimo}(x, 1), -1)$.

La posición consignada se ha de alcanzar lo más rápido posible, por tanto la velocidad deberá disminuir al acercarse a la posición requerida, por ejemplo a la mitad cuando falte un centímetro; $\mu_L(0{,}01) = 0{,}500$.

Usaremos $\text{pmL} = \frac{0{,}5}{x}$ para x = 0,01; siendo pmL = 50.

$$\mu_L(\text{error}) = \text{lejos}(50 \cdot \text{error})$$

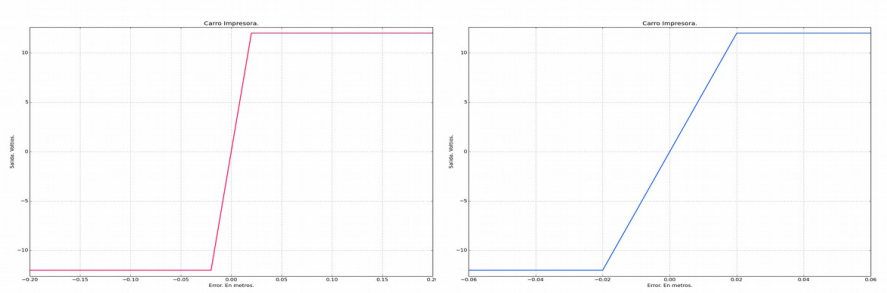
La regla de inferencia a utilizar será la del signo por la variable lingüística.

$$\mu(\text{error}) = \text{signo}(\text{error}) \cdot \mu_L(\text{error})$$

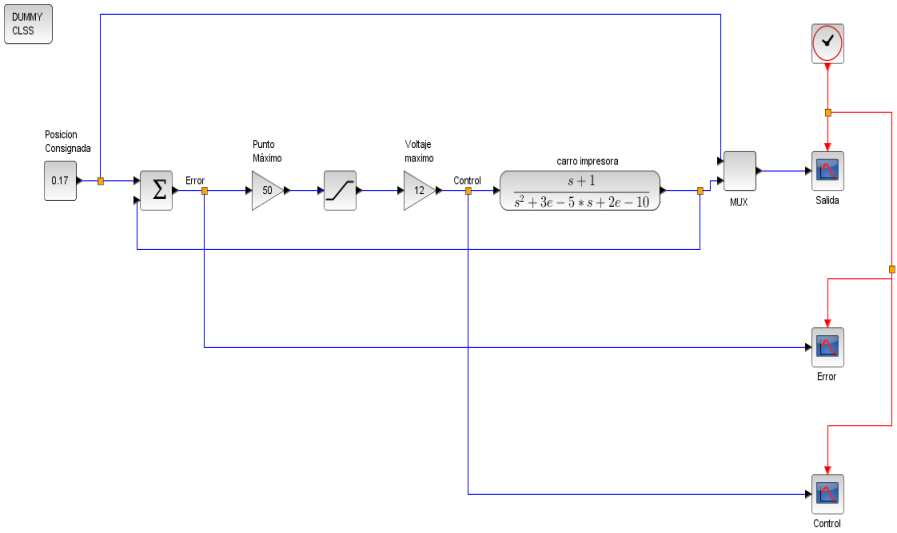
La variable a controlar es el voltaje del motor, siendo éste de 12 V máximo.

$$\text{voltaje} = 12 \cdot \mu(\text{error})$$

La representación de la variable de control se da a continuación.

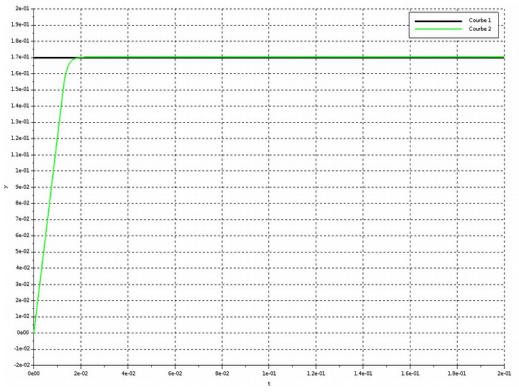


Vamos a simular el sistema con Xcos de Scilab 5.5.0.



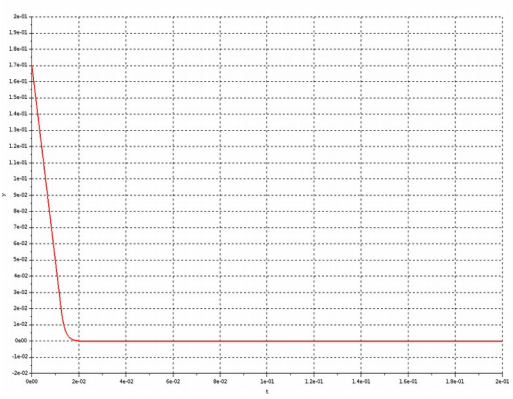
Para una posición consignada de 0,17 m tenemos las siguientes gráficas de la salida.

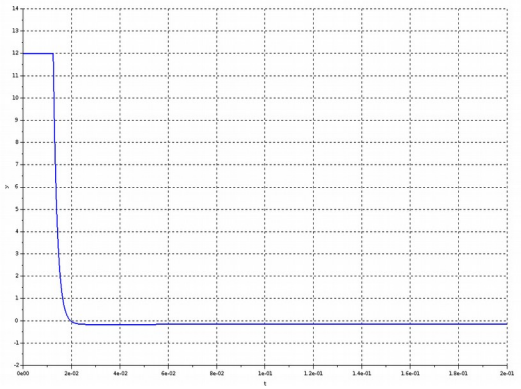
En negro el valor consignado y en verde la posición del carro.



Se observa claramente que hasta que no está cerca del objetivo el desplazamiento es constante, y aminora la velocidad cuando queda poco para llegar a la posición consignada.

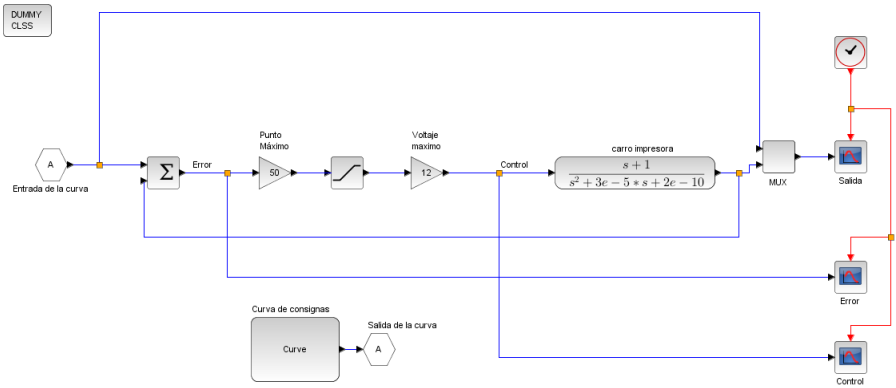
Vemos que el sistema de control tarda 20 ms en alcanzar la posición deseada, por tanto es un sistema eficiente, fácil de implementar y bajo costo en programación.





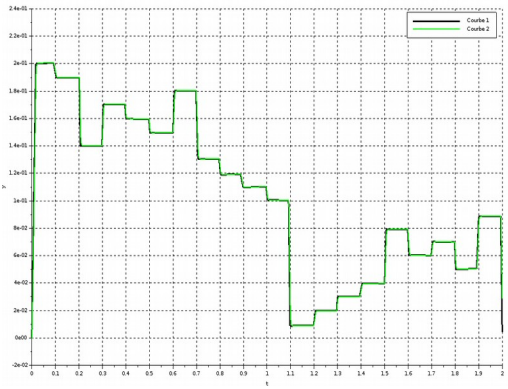
La salida del controlador es suave y continua, que es lo buscado en cualquier sistema de control.

Vamos a poner **ALBLI** a prueba, en la siguiente imagen se muestra la simulación para distintos niveles de entrada.



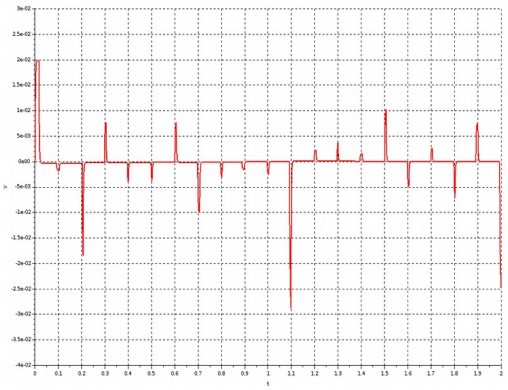
La entrada es una curva de diferentes consignas tal y cómo se muestra en la figura de la derecha, éstas están comprendidas entre 0 m y 0,20 m para la entrada y cada una dura 100 ms, durante 2 segundos.

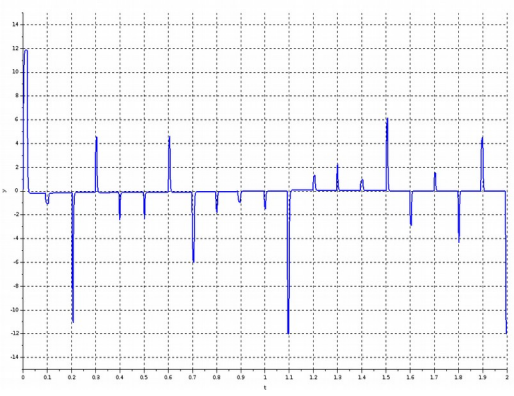
Las gráficas de salida son las siguientes.



En negro los valores consignados y en verde la posición del carro. Se ve claramente lo rápido que alcanza el cartucho la posición referenciada.

Se aprecia que una vez alcanzada la posición el error permanece en cero.





La salida del controlador es impecable, cumple todas las expectativas de cualquier sistema de control.

20. CALEFACCIÓN DE UNA HABITACIÓN MEDIANTE UNA ESTUFA ELÉCTRICA DE 1.500 W.

El problema a solucionar es calefactar la habitación de un apartamento mediante una estufa eléctrica de 1.500 w.

La ecuación de planta de la habitación es $\frac{T(s)}{Q(s)} = \frac{0{,}194}{6490 \cdot s + 1}$. La temperatura de consigna será de 21 ºC con 10 ºC al inicio.

error = temperatura consignada – temperatura medida

error = 21 – temperatura medida

La variable consignada es la temperatura, que será cero cuando error = 0 y máxima mientras no se alcance, por tanto la variable lingüística es lejos, y su función de pertenencia $\mu_L(\text{error}) = \text{lejos}(\text{pmL} \cdot \text{error})$ para $\text{lejos}(x) = \text{maximo}(\text{minimo}(x, 1), -1)$.

Para una diferencia de 0,5º C el sistema ha de ir disminuyendo su potencia, así que el grado de pertenencia es $\mu_L(0{,}5) = 1$.

Usaremos $\text{pmL} = \frac{1}{x}$ para x = 2; siendo pmL = 2.

$$\mu_L(\text{error}) = \text{lejos}(2 \cdot \text{error})$$

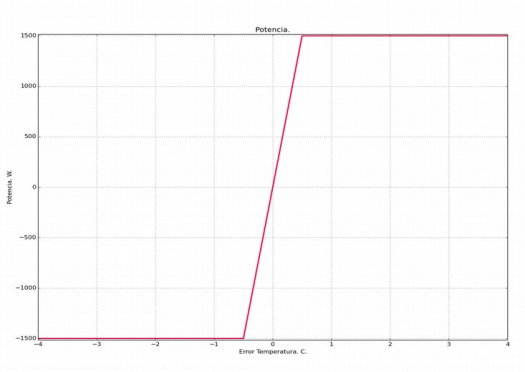
La regla de inferencia a utilizar será la del signo por la variable lingüística.

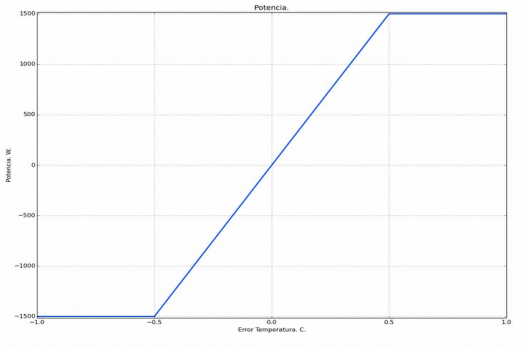
$$\mu(\text{error}) = \text{signo}(\text{error}) \cdot \mu_L(\text{error})$$

La variable a controlar es la potencia de la estufa, siendo ésta de 1.500 w máximo.

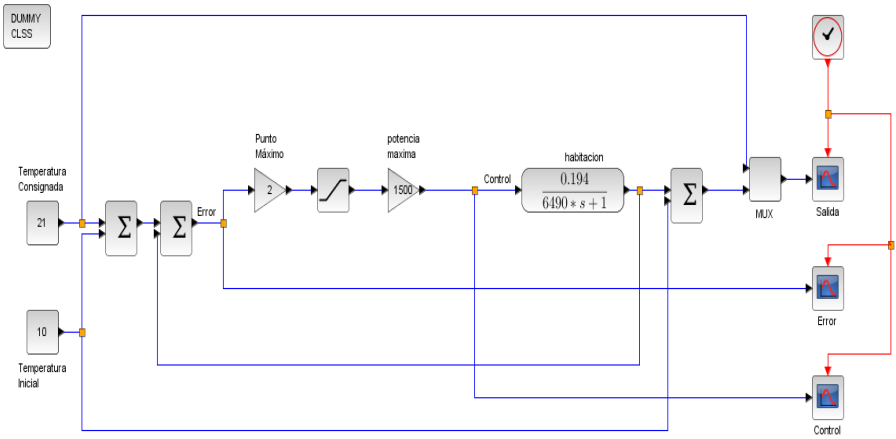
$$potencia = 1500 \cdot \mu(error)$$

Aunque en las gráficas está representado un error negativo en la realidad no va a ser así, puesto que estamos tratando de un sistema calefacción y no de aire acondicionado; por tanto debemos tener en cuenta el cuadrante positivo.

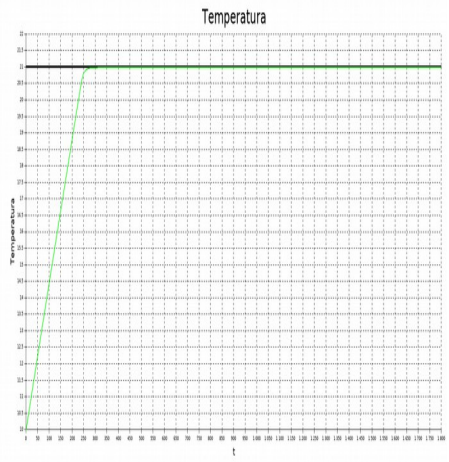




Vamos a simular el sistema con Xcos de Scilab 5.5.0.



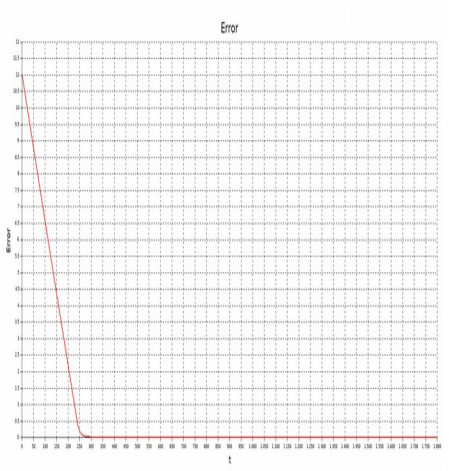
Para una temperatura consignada de 21ºC y temperatura inicial de 10ºC tenemos las siguientes gráficas de la salida.

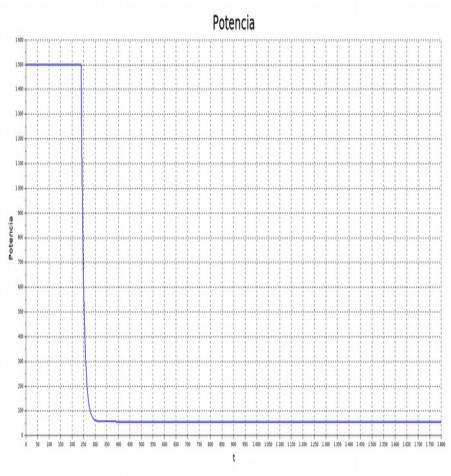


En negro el valor consignado y en verde la temperatura de la estancia.

Se observa claramente que hasta que no está cerca del objetivo el aumento de la temperatura es constante, y aminora el incremento cuando queda poco para llegar a la temperatura consignada.

Vemos que el sistema de control tarda 300 s; 5 min; en alcanzar la temperatura deseada, por tanto es un sistema eficiente, fácil de implementar y bajo costo en programación.





La salida del controlador es suave y continua, que es lo buscado en cualquier sistema de control.

Se observa que mientras no se alcanza la temperatura deseada la estufa está funcionando a la máxima potencia, para después reducirla y mantenerse constante para igualar las pérdidas.

21. CONTROL DEL MOTOR DC BN12-28.

Vamos a controlar la velocidad del motor de corriente continua BN12-28. La velocidad angular máxima es de 998 rad/s, así que ésta será la única variable a tener en cuenta.

error = 998 – velocidad angular medida

La variable consignada es la velocidad angular, que será cero cuando error = 0 y máxima mientras no se alcance, por tanto la variable lingüística es lejos, y su función de pertenencia $\mu_L(\text{error}) = \text{lejos}(\text{pmL} \cdot \text{error})$ para $\text{lejos}(x) = \text{maximo}(\text{minimo}(x, 1), -1)$.

Para una diferencia de 0,10 rad/s el sistema ha de ir disminuyendo su voltaje, así que el grado de pertenencia es $\mu_L(0{,}10) = 0{,}999$.

Usaremos $\text{pmL} = \frac{1}{x}$ para x = 0,10; siendo pmL = 10.

$$\mu_L(\text{error}) = \text{lejos}(10 \cdot \text{error})$$

La regla de inferencia a utilizar será la del signo por la variable lingüística.

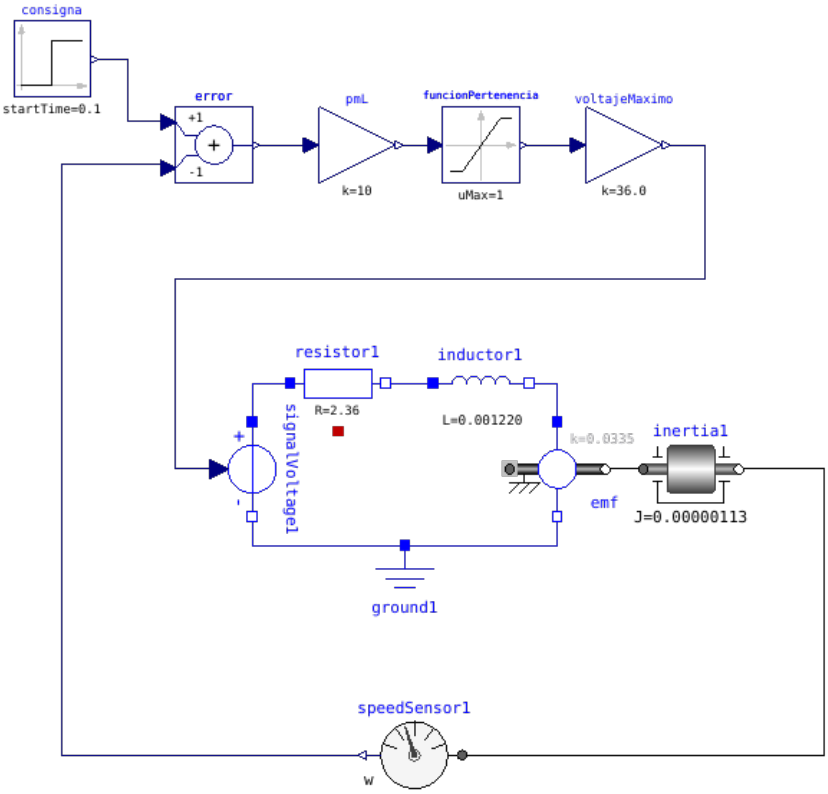
$$\mu(\text{error}) = \text{signo}(\text{error}) \cdot \mu_L(\text{error})$$

La variable a controlar es el voltaje de entrada, siendo éste de 36 V máximo.

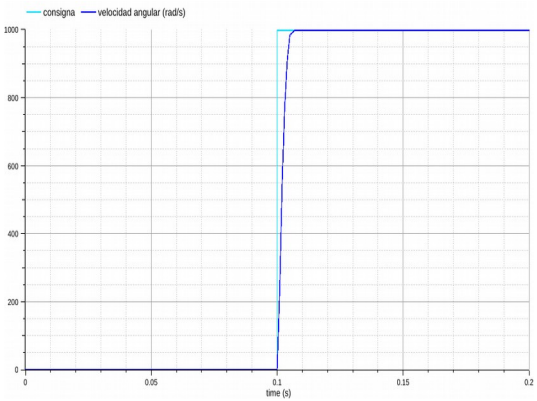
$$\text{señal} = 36 \cdot \mu(\text{error})$$

Las características principales del motor son; 2,36 Ω de impedancia; 1,22 mH de inductancia, un momento de inercia de 11,3 $g \cdot cm^2$ y una f.e.m de 0,0335 Nm/A.

La simulación se llevará a cabo con Open Modelica y el modelaje se hará con OMEdit.

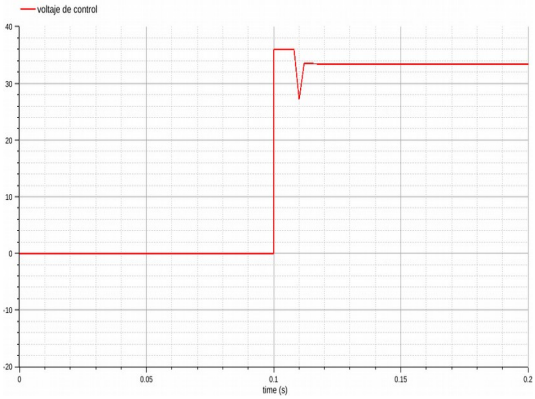


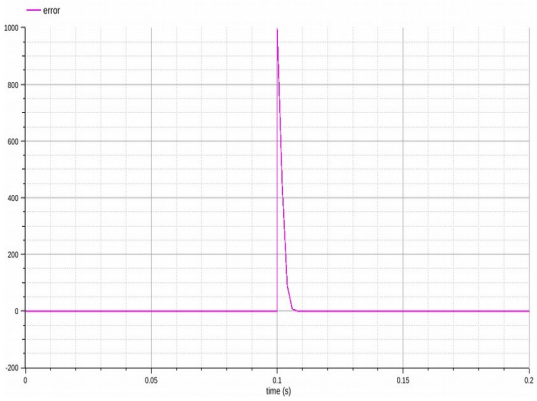
Para el valor consignado de 998 rad/s tenemos tres gráficas de datos, la salida, el error y el control.



El escalón se produce 0,1 segundos desde el inicio, y en menos de 0,01 segundos llega al valor consignado.

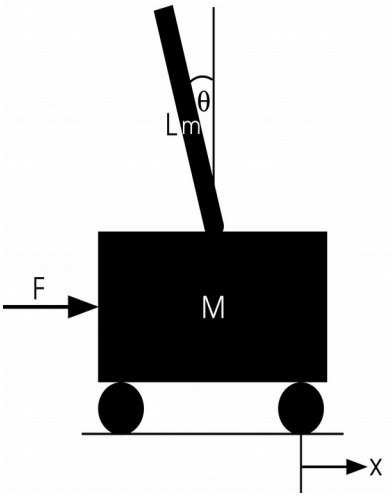
El control es efectivo y preciso, en él podemos apreciar el tipo de control que hace y cuánto tiempo tarda en llegar al valor consignado.





En el error podemos apreciar con mejor detalle la breve duración en alcanzar el objetivo.

22. SIMULACIÓN DE UN CARRO CON PÉNDULO INVERTIDO.



M; Masa del carro.
m; Masa del péndulo.
b; Fricción del carro.
L; Longitud al centro de masa del péndulo.
I; Inercia del péndulo.
F; Fuerza aplicada al carro.
x; Coordenadas de posición del carro.
θ; Ángulo del péndulo respecto de la vertical.

Hay que mantener un péndulo en posición vertical con el eje de giro en la parte inferior y solidario a un carro. En otras palabras, la idea es encontrar la fuerza que ha de aplicarse al carro para que el péndulo no se caiga, incluso si se le perturba con un empujón.

Las variables de entrada a tener en cuenta son el ángulo respecto de la vertical y la velocidad del carrito.

El $error_1$ va a ser el ángulo respecto de la vertical, θ, pero como los simuladores miden el ángulo desde cero el error es,

$$error_1 = \theta = \pi/2 - \varphi$$

donde φ es el ángulo respecto de la horizontal.

Para un $error_1 = 0$ no actúa el sistema de control y para un error mayor la acción de control ha de ser mayor, así que la función de pertenencia será
$\mu_L(error_1) = lejos(pmL \cdot error_1)$ para
$lejos(x) = maximo(minimo(x, 1), -1)$.

Para un ángulo $\varphi = 0{,}17$ rad el grado de pertenencia es $\mu_L(0{,}17) = 1$.

Usaremos $pmL = \frac{1}{x}$ para x = 0,1; siendo coeficiente de pmL = 10.

$$\mu_L(error_1) = lejos(10 \cdot error_1)$$

El $error_2$ va a ser la velocidad,

$$error_2 = 0 - v$$

Para un $error_2 = 0$ no actúa el sistema de control y para un error mayor la acción de control ha de ser mayor, así que la función de pertenencia será
$\mu_L(error_2) = lejos(pmL \cdot error_2)$ para
$lejos(x) = maximo(minimo(x, 1), -1)$.

Para una velocidad angular de -2,86 m/s el grado de pertenencia es $\mu_L(-2{,}86) = 1$.

Usaremos $pmL = \frac{1}{x}$ para x = -2,86; siendo pmL = -0,35.

$$\mu_L(error_2) = lejos(-0{,}35 \cdot error_2)$$

La regla de inferencia a utilizar será la 6 de la media del máximo y el mínimo.

$$\mu_{MAX}(error_1, error_2) = \max(signo(error_1) \cdot \mu_L(error_1), signo(error_2) \cdot \mu_L(error_2))$$

$$\mu_{MIN}(error_1, error_2) = \min(signo(error_1) \cdot \mu_L(error_1), signo(error_2) \cdot \mu_L(error_2))$$

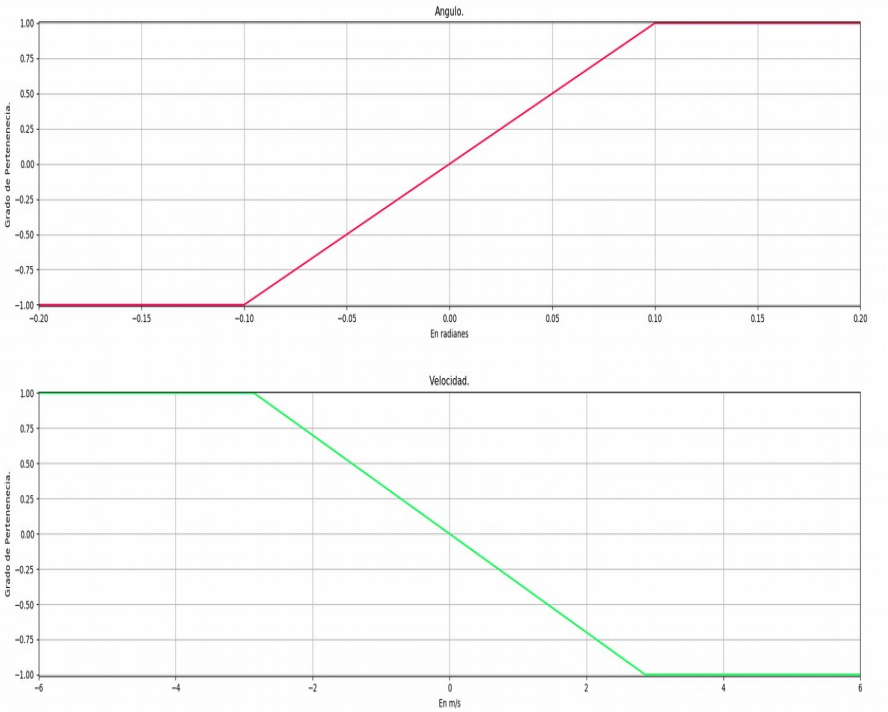
$$\mu(error_1, error_2) = \frac{\mu_{MAX}(error_1, error_2) + \mu_{MIN}(error_1, error_2)}{2}$$

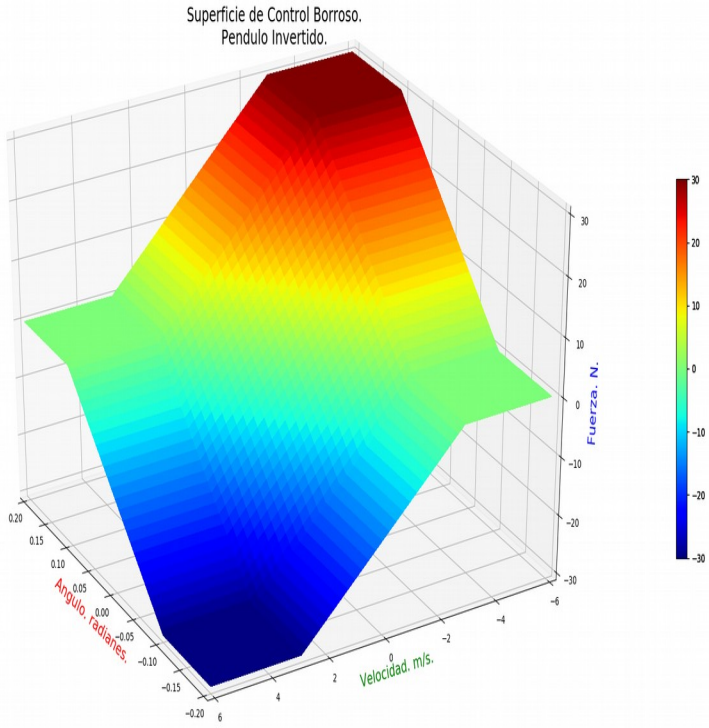
La variable a controlar es la fuerza del carro, siendo ésta de 30 N máximo.

$$fuerza = 30 \cdot \mu(error_1, error_2)$$

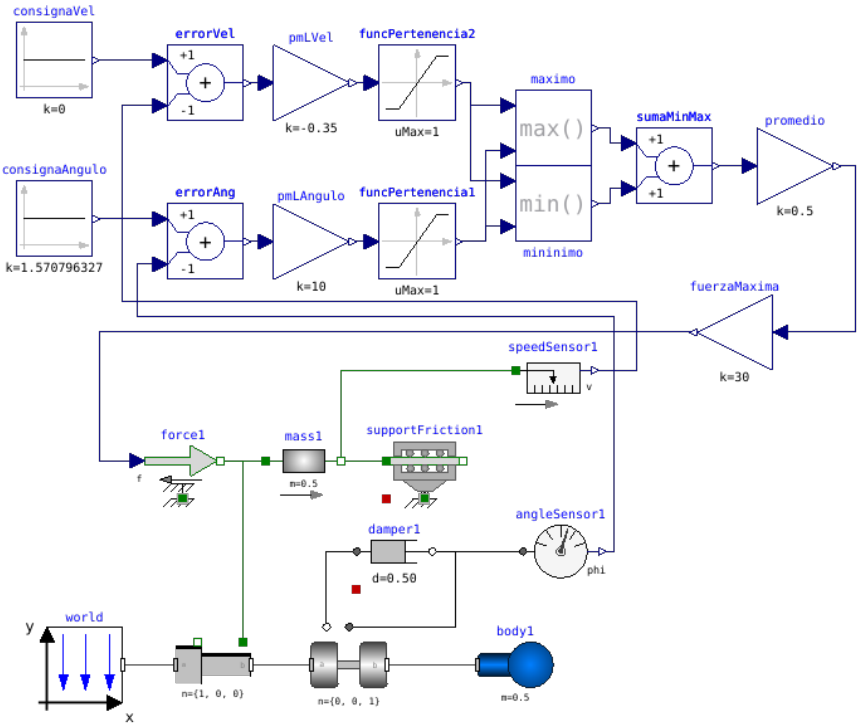
Los datos de entrada son: 0,500 Kg para la masa del carro y el péndulo; 0,100 m para la longitud del carro; 0,300 m para la distancia al centro de masas del péndulo; 0,100 N para la fricción del carro; 0,006 Kg·m² para la inercia del péndulo.

La representación de las variables de control se dan a continuación.

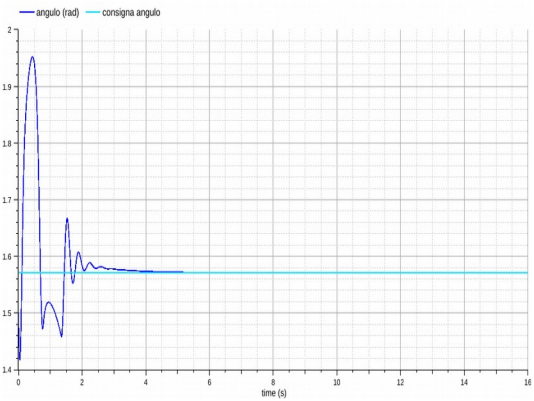



Superficie de Control Borroso.
Pendulo Invertido.
Fuerza. N.
Angulo. radianes.
Velocidad. m/s.

La simulación se llevará a cabo con Open Modelica y el modelaje se hará con OMEdit.

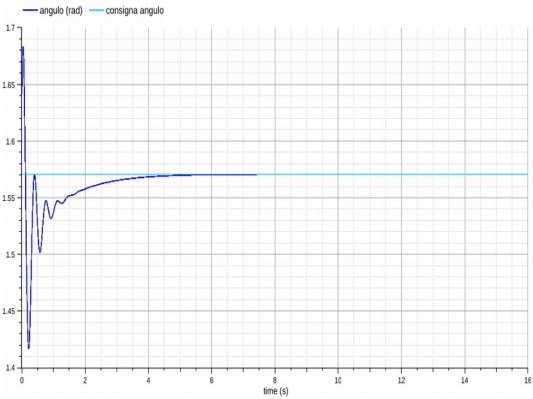


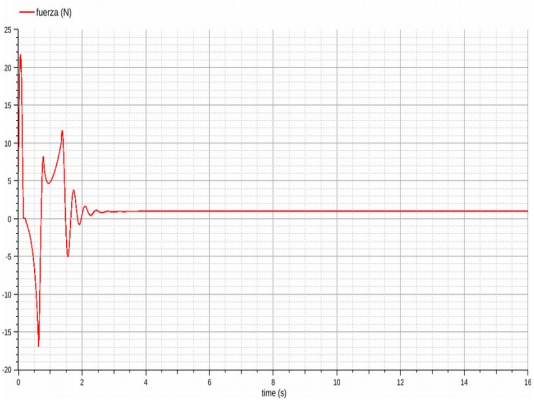
Para un ángulo consignado de π/2 rad y ángulos iniciales de 1,5 rad y 1,6416 rad (-1,5 rad) con velocidades angulares de -4 rad/s y 2 rad/s respectivamente tenemos las siguientes gráficas de la salida del ángulo, la fuerza, la velocidad y el error.



Para el ángulo inicial de 1,5 rad y velocidad angular de -4 rad/s se observa que en aproximadamente 5 segundos alcanza la posición vertical de π/2 radianes.

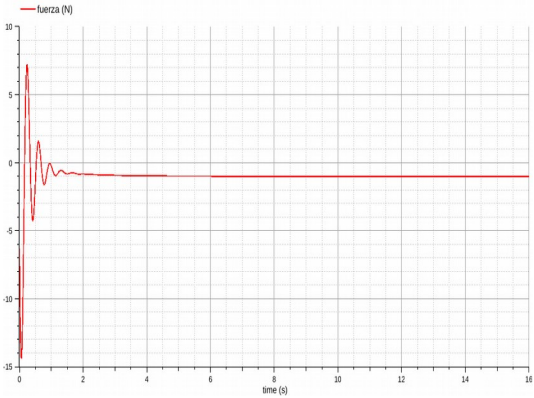
Para el ángulo inicial de 1,6416 rad y velocidad angular de 2 rad/s se observa que en aproximadamente 8 segundos alcanza la posición vertical de π/2 radianes.

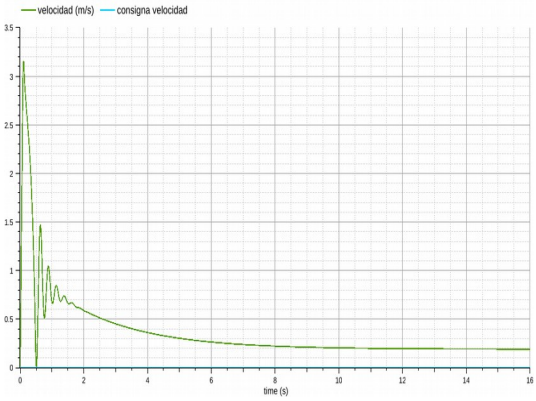




Para el ángulo inicial de 1,5 rad y velocidad angular de -4 rad/s se observa que la fuerza dura apenas dura 4 s.

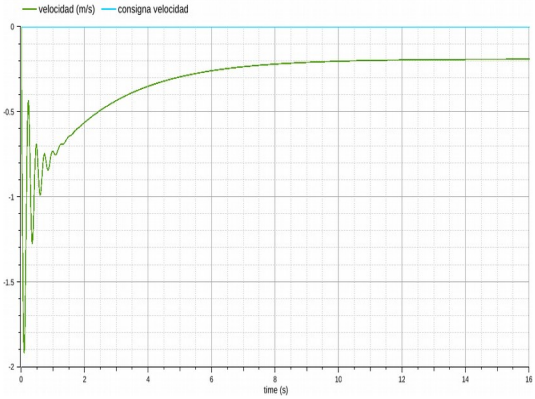
Para el ángulo inicial de 1,6416 rad y velocidad angular de 2 rad/s se observa que y en aproxi-madamente 6 s apenas hace fuerza.

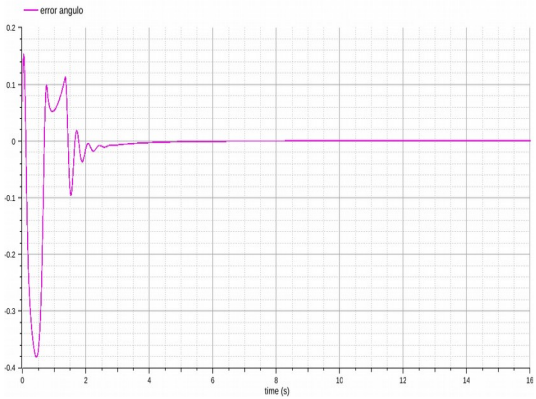




Para el ángulo inicial de 1,5 rad y velocidad angular de -4 rad/s se observa que la velocidad disminuye.

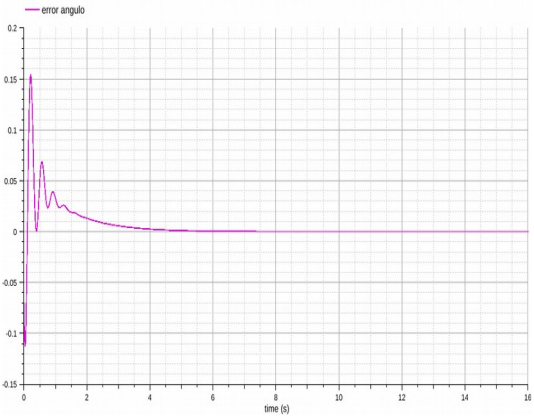
Para el ángulo inicial de 1,6416 rad y velocidad angular de 2 rad/s se observa que la velocidad disminuye.



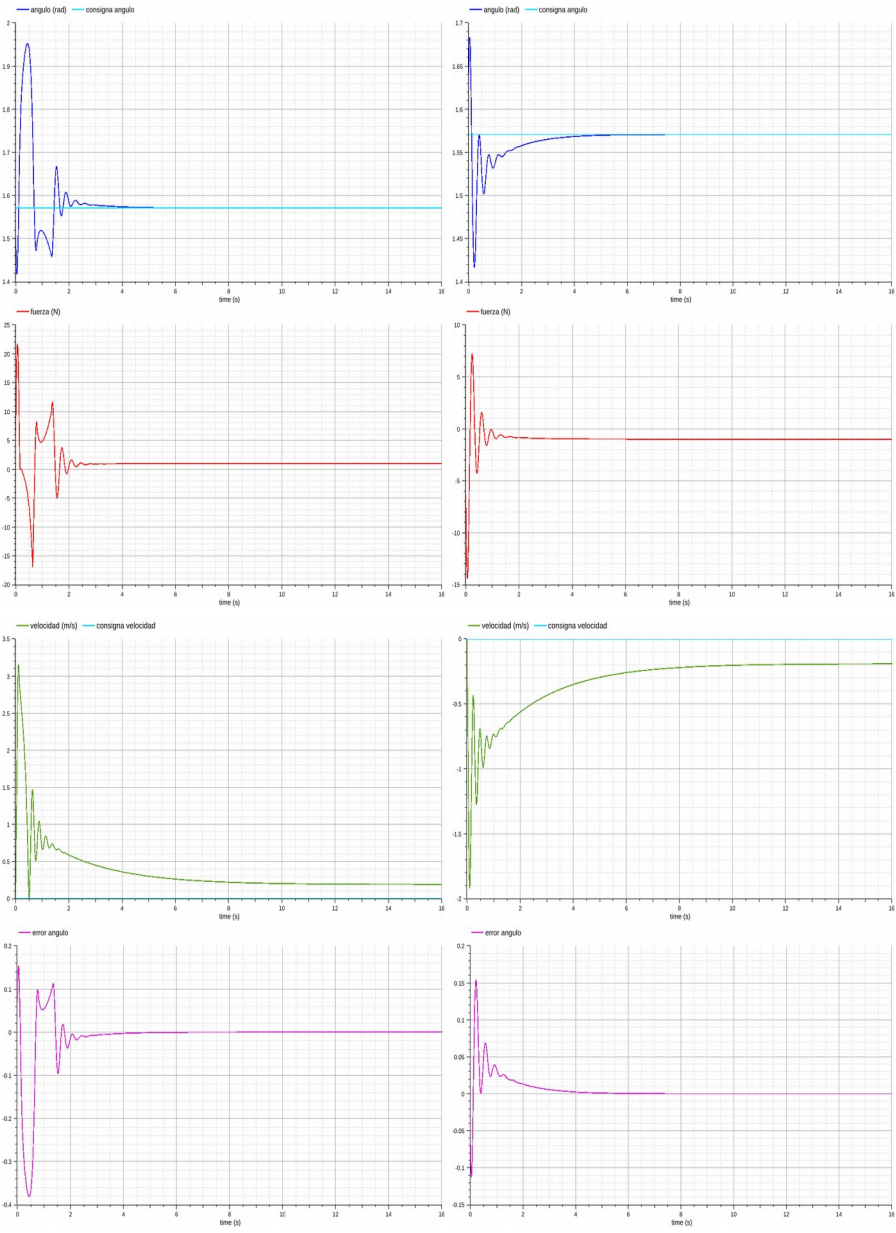


Para el ángulo inicial de 1,5 rad y velocidad angular de -4 rad/s se observa que el error del ángulo se hace cero en unos 8 segundos.

Para el ángulo inicial de 1,6416 rad y velocidad angular de 2 rad/s se observa que en 8 segundos el error es cero.



Las siguientes gráficas se muestran el ángulo, fuerza y error para los ángulos iniciales de 1,5 rad y 1,6414 rad.



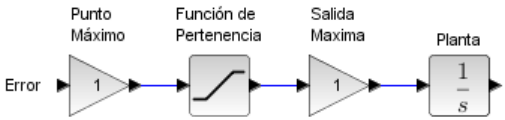
Cómo se puede ver los resultados son prácticamente los mismos que en **ALBHI**.

Aplicarlo a un sistema de transporte individual basado en el péndulo invertido resulta ser una tarea más fácil que en **ALBHI**.

Cómo ha podido deducir el lector con el sistema de control **ALBLI** se puede controlar cualquier tipo de máquina a muy bajo coste computacional. Ello lo demuestro en el capítulo 24.

23. SINOPSIS Y SIGNIFICACIÓN.

El nuevo enfoque de la lógica difusa se puede sintetizar en el gráfico siguiente:



El error es multiplicado por el valor del punto máximo y se le aplica la función de pertenencia que está comprendida ente -1 y 1 en valores de 'Y' y finalmente se multiplica por el valor máximo de salida y de ahí a la planta. Esto hace innecesario al PID y a los conjuntos difusos, obteniéndose mejores resultados con pocos cálculos.

Cuando empecé este proyecto hace años no conocía las redes neuronales artificiales (RNA) en profundidad, y seguramente el lector avezado habrá podido comprobar cierta similitud con ellas respecto de la función de pertenencia y su semejanza con la función de activación. Aprendiendo sobre ellas desde hace pocos meses me he dado cuenta de algo que los expertos en IA andan buscando y que he hallado impensadamente al desarrollar este nuevo enfoque de la lógica borrosa, ¿cuál es el significado de los pesos de las RNA a la hora de deducir de ellas unas reglas lógicas?, la solución es bien sencilla a tenor de lo que acabo de desarrollar y la tenemos todos delante de nuestros ojos pasando inadvertida cuán gorila que pasa delante de nuestras narices mientras prestamos atención a otro asunto, y es que **la función de activación es también una regla lógica** al igual que la función de pertenencia (en el capítulo 5 defino otras funciones de pertenencia), por tanto el valor de los pesos tiene sentido y es interpretable dando un contexto y una importancia a dichas reglas según su valor y signo, por ello a la hora de definir las reglas del sistema en términos lingüísticos y/o lógicos no sólo debemos tener en cuenta los pesos, sino además las funciones de activación y la relación con ellos. Si tomamos cada uno por separado no podremos concretar específicamente dichas reglas como está ocurriendo actualmente. Además nos puede ayudar a determinar la cantidad de neuronas y capas de nuestros modelos.

24. ESTUDIO COMPARATIVO: LÓGICA DIFUSA Y ALBLI.

En la lógica difusa un conjunto borroso triangular se define con 2 líneas de código Python. Se definen 5 conjuntos[4] difusos triangulares o etiquetas para cada entrada y salida.

Si diseñamos un sistema de control difuso al uso con una salida y 4 entradas con 5 etiquetas cada una tenemos que hay $5 \times 4 = 20$ líneas de código para las entradas, más otras 5 para la salida, más $5^4 = 625$ líneas para las bases de reglas SI- ENTONCES-; éstas se calculan con el producto cartesiano de las entradas; más una línea para la salida desborrificada, lo que nos da **653 líneas de código**. Esta forma de control tiene 3 líneas de código fijas; dos para la definición de la función triangular y una para la salida desborrificada; y habrá que sumar las líneas pertenecientes a las salidas, entradas y conjuntos.

$$\text{líneas de código} = 3 + (\text{conjuntos} \times \text{entradas}) + (\text{conjuntos} \times \text{salidas}) + \text{conjuntos}^{\text{entradas}}$$

En **ALBLI** definimos la función lejos(x) y cerca(x) con dos líneas cada una de código Python, lo que hace 4 líneas de código. Para cada entrada se define una de estas funciones, por tanto es una línea cada una.

Si diseñamos un sistema de control difuso **ALBLI** con una salida y 4 entradas tenemos que hay 4 líneas de código para las entradas y bases de reglas, más las 4 de las variables lingüísticas, más la salida con la regla de inferencia que escojamos, lo que nos da **9 líneas de código**. Esta forma de control tiene 4 líneas de código fijas y habrá que sumar las líneas pertenecientes a las entradas y salidas.

$$\text{líneas de código} = 4 + \text{entradas} + \text{salidas}$$

4 Aunque se puede definir más o menos conjuntos difusos por entrada o salida, el óptimo según los expertos en la materia es 5. Siendo éstos NG, NP, C, PP y PG.

A continuación una tabla comparativa en el que se muestran las líneas de código necesarias para un sistema borroso y **ALBLI** dependiente del número de entradas, salidas y conjuntos difusos, y la proporción de código que hay entre ellas.

Salidas	Conjuntos
1	5

Entradas	Líneas Fuzzy	Líneas ALBLI	Líneas Reglas	Proporción (1)	Proporción (2)
2	43	7	25	6,1	12,5
3	148	8	125	18,5	41,7
4	653	9	625	72,6	156,3
5	3.158	10	3.125	315,8	625,0

(1) Es la proporción de líneas totales entre el sistema fuzzy y albli.

(2) Es la proporción de líneas bases de reglas entre el sistema fuzzy y albli.

Se observa inmediatamente el potencial de **ALBLI**, ya que en la lógica disfusa el número de líneas de la base de reglas crece exponencialmente, mientras que en **ALBLI** lo hace linealmente. Por tanto con **ALBLI** la lógica borrosa se torna más fácil de manejar, es más rápida y consume menos energía, pues con menos memoria y velocidad se puede hacer lo mismo. Ello hace que se pueda utilizar tecnología algo obsoleta para casi todo, pudiendo reutilizar hardware de unos 10 años o menos con **ALBLI** teniendo las mismas prestaciones que la tecnología actual con el sistema de lógica difusa SI- ENTONCES- cuando se tienen muchas entradas.

Expresando lo anterior en términos de rendimiento computacional, para la base de reglas de la lógica borrosa el orden de complejidad es $O(c^n)$, mientras que el de **ALBLI** es O(n), donde n denota cuántas entradas hay y c es el número conjuntos difusos o etiquetas de cada una. Queda así demostrada la ineficiencia de la lógica difusa basada en la regla SI-ENTONCES- frente a **ALBLI**.

En el mercado hay un coprocesador difuso para la industria automotriz y el procesamiento de imágenes en tiempo real que a 20 MHz procesa hasta 10 millones de reglas por segundo y admite la definición de un máximo de 80 reglas y 4 entradas para una salida, por tanto en 8 µs resuelve las 80 reglas. Sin embargo si se utilizase **ALBLI** y una entrada por regla, con el mismo consumo de energía, velocidad de procesamiento y memoria se diseñaría otro coprocesador difuso que hiciese todavía más; cómo el control de un coche autónomo, un cohete espacial reutilizable, producción de alimentos, robots exploradores autónomos, robots agricultores, robots mineros, reconocimiento de patrones, producción de energía limpia y renovable, etc. Puesto que la proporción de líneas en la base de reglas es $\frac{5^{80}}{80}$ veces[5] superior, algo impracticable hasta para un supercomputador.

Veámoslo desde el hardware libre y tomemos Arduino Due; con 12 entradas analógicas; las entradas irán conectadas a sensores digitales de posición (lineal o angular). Internamente a cada una se le hace su primera y segunda derivada con lo que hay un total de 36 entradas, por tanto la proporción de líneas en la base de reglas entre el sistema fuzzy de Zadeh y **ALBLI** es de $\frac{5^{36}}{36}$ veces superior, que es un número difícil de manejar para un sistema de base de reglas SI- ENTONCES- hasta en un superordenador. Los microcontroladores que hay actualmente en el mercado son mas que suficientes para realizar cualquier cosa que nos podamos imaginar a un precio baratísimo. Si los sensores son de aceleración habrá que hacer a cada entrada una primera integral para calcular la velocidad y una segunda para calcular la posición.

En tan sólo 36 gramos de peso, 512 KB de almacenamiento, y 84 MHz de velocidad de reloj, Arduino Due y cualquier microcontrolador o hardware similar a Arduino; tiene el potencial de un supercomputador en los sistemas de control utilizando solamente **ALBLI**.

5 El numerador hace referencia a la base de reglas de un sistema SI- ENTONCES- con 5 conjuntos difusos por entrada y 80 entradas en total, mientras que el denominador es la base de reglas ALBLI.

La desventaja que tiene la lógica borrosa es la dificultad de encontrar conjuntos difusos y reglas fiables sin la participación de un experto humano, pero con **ALBLI** casi no necesitamos reglas; puesto que la función de pertenencia es una regla en sí misma; cómo he demostrado en los ejemplos, ya que basta con aplicar **ALBLI** al error para los sistemas de control.

El uso de algoritmos genéticos para encontrar el mejor candidato en el control de máquinas con gran cantidad de entradas y salidas aceleraría la implementación y el desarrollo de aplicaciones de **ALBLI**. Se podrían criar programas en granjas de software que fuesen capaces de hacer cualquier cálculo, o diseñar todo tipo de sistemas de control, o reconocimiento de patrones, etc.

El potencial que tiene **ALBLI** para manejar una gran cantidad de variables de entrada hace que sea el sistema idóneo para el control de máquinas complejas autónomas que están por devenir, e incluso simular un ser vivo.

ALBLI supone un avance extraordinario en la teoría de control, ya que con un algoritmo sencillo podemos controlar cualquier variable tanto física como lógica, de una forma eficiente. Superando así al PID y a la lógica difusa basada en la regla SI- ENTONCES-, e incluso a **ALBHI**.

Vamos a hacer un ejercicio de imaginación para ver que nos depararía el futuro cuando introducimos **ALBLI** en la ciencia y la tecnología.

25. FUTURO.

Actualmente el rápido desarrollo que tienen las FPGA, entre ellas las libres, permiten hacer modelos con costes de desarrollo y adquisición mucho menores, y configurarlas de manera que hagan uso de la aritmética de saturación para recrear **ALBLI** en circuitos combinacionales difusos ya sean síncronos o no; ver el apéndice J; con las funciones de alto nivel (como sumadores y multiplicadores) embebidas en la propia matriz de interconexiones, pudiendo hacer sistemas operativos difusos embebidos. Esto hace que los controladores de automatización programables (PAC) sean más fáciles de desarrollar pudiendo hacer sistemas de control baratos y comercializarlos de forma independiente, abriendo un nuevo mercado para todo tipo de automatizaciones de bajo coste.

También hay otros hardwares libres que están teniendo éxito en los sistemas de control y en los que podemos definir las variables lingüísticas lejos(x) y cerca(x) como son Arduino y similares, o pudiendo programar **ALBLI** directamente en microcontroladores; haciendo innecesario los procesadores difusos actuales; y si queremos más recursos computacionales podemos optar por el mini ordenador Raspberry Pi. Con Raspberry Pi se puede experimentar la programación de kerneles difusos libres como un FUZZY LINUX; y también con microkerneles; y crear sistemas operativos borrosos, desarrollando superordenadores que cabrían en la palma de la mano, o hacer un sistema de control muy potente y avanzado.

El futuro de la programación está en el diseño de software de decisiones que será el centro de la Inteligencia Artificial, ya sea con C++ o Python, o bien con un lenguaje propio. La idea que transmite **ALBLI** será el núcleo sobre el que se base este nuevo tipo de software. Todo sistema operativo tiene las funciones de gestionar los recursos del hardware y proveer servicios a los programas de aplicación de software, por tanto se basa en última instancia en tomar una decisión para realizar la acción de control correspondiente a su grado de pertenencia.

La robótica de las cosas y máquinas autónomas empiezan a ser una realidad en algunos casos cómo el robot que barre toda la casa o las impresoras 3D, y en pocos años los coches, drones, cohetes, exoesqueletos, etc, empezaremos a verlos como algo normal. **ALBLI** será el mejor aliado a la hora de crear sistemas de control para ellos, que como he demostrado en capítulos anteriores serán robustos, baratos, rápidos y con escaso coste computacional y energético. Los smartphones, tablets, smartv y demás hardware seguramente aumentarán su velocidad y eficiencia, pudiendo darles nuevos y mejores usos. Haciendo que el internet de las cosas se torne muy fácil de implementar. Y si además se utiliza FPGA también actualizaríamos el hardware.

ALBLI puede contar pronto con un aliado inesperado y que parece ser el futuro de la computación, el transistor ferroeléctrico, que combinaría nanocables de silicio con un polímero ferroeléctrico, un material que cambia la polaridad de los campos electromagnéticos cuando éstos se aplican y que la mantiene en ausencia de ella, es decir aplicar el punto máximo para la función de pertenencia. También se está trabajando en otro tipo de dispositivo tan novedoso como el transistor ferroeléctrico, el memristor. El memristor puede variar la resistencia y mantenerla incluso si le falta la alimentación o cambiarla al polarizarla negativamente, lo que puede ser muy bueno cómo memoria aplicando **ALBLI** constituiría al mismo tiempo un procesador difuso, ya que la resistencia varía continuamente entre un máximo y un mínimo; otro mecanismo similar que varíe su valor según la entrada puede ser usado cómo procesador borroso basado en **ALBLI** uniendo memoria y procesamiento en un sólo instrumento. Otro dispositivo que podría competir con los anteriores en el desarrollo del hardware futuro y que se puede aplicar a **ALBLI** es el Light-Effect Transistor (LET), el LET no utiliza campos eléctricos para modular la corriente que atraviesa el transistor, sino luz. Consiste en una fibra a escala nanométrica cuyo material se vuelve conductor cuando recibe luz y aislante cuando no la recibe. Controlando dicha luz tendríamos otro procesador difuso.

Las superficies de inferencia que he expuesto en los capítulos anteriores han sido estáticas, pero nada impide que puedan ser dinámicas si se acoplan a otras superficies de inferencia. Se podría generar una jerarquía

de superficies de inferencia a modo de árbol en la que haya unas fijas y otras dependientes de éstas o dinámicas, haciendo un sistema de control flexible y adaptativo capaz de lidiar con cualquier situación, en la que los puntos máximos de las superficies dinámicas sean las salidas de las superficies estáticas multiplicadas por sus respectivas salidas máximas o estén en un determinado rango de valores que no tengan que partir de cero, constituyendo poderosos algoritmos. Sería similar a las RNA, con la salvedad de que ya habría una parte fija y otra para aprender y además no sólo con las dos operaciones de suma y producto. Dicha jerarquía de superficies de inferencia originaría un sistema operativo muy potente, ya que tendría muy pocas líneas de programación en comparación con el sistema actual, además de ser muy estable y prácticamente inmune a errores. Programar las tres leyes de la robótica sería relativamente sencillo. El coche autónomo sería mas fácil de implementar con dicho sistema. Como símil podemos considerar que todos los seres vivos con un cerebro parecen comportarse de manera similar, ya que nacen con un sesgo inductivo (superficies estáticas) y otras que se han de aprender (superficies dinámicas).

Las superficies de inferencia representan la distribución espacial de valores lógicos difusos, por tanto pueden ser vistas como campos. Estos campos los podemos definir como **campos lógico-difusos**; con propiedades diferentes a los de la física; pudiendo crear zonas mediante funciones definidas a trozos en las que no hay ningún valor, es decir, crear agujeros lógicos. O quizá si se deducen de algún sistema axiomático comprender mejor dicho sistema.

En la decisión borrosa las funciones de control son el comportamiento del sistema a controlar, por lo que si deseamos otro tipo de actuación deberemos introducir otras funciones diferentes a las estudiadas en este libro y realizar sus propias reglas de inferencia. Esto le da a la lógica difusa un potencial infinito e ilimitado, haciendo que la inteligencia artificial sea más eficiente y su desarrollo más rápido.

En todos los ejemplos he mostrado superficies de inferencia, pero nada indica que no se puedan hacer en más dimensiones, cada superficie indica

el comportamiento del sistema, y donde no la hay son situaciones que no existen o no podemos concebir, esto mismo lo podemos extrapolar a volúmenes o hipervolúmenes en cuyo espacio se cumplen las reglas del sistema.

Podemos imaginar que la combinación de varios átomos va a alterar los orbitales en la nueva molécula, reforzando unos, eliminando otros o produciendo nuevos, dichos orbitales pueden constituir una superficie de inferencia, pudiendo ser la que queramos, con lo que la química adquiere un significado nuevo. Así mismo toda la bioquímica cobra otro sentido, ya que no sólo el ADN se puede considerar cómo un sistema computacional, sino las proteínas, ácidos grasos, sacáridos, etc, que constituyen un ser vivo puede ser visto cómo minisistemas difusos que forman parte de otros sistemas difusos, extendiéndose a todos los fenómenos biológicos. Y también la física puede ser vista como computación en la que cada elemento de la tabla periódica es un supercomputador cuántico difuso, por ello los orbitales atómicos son el mejor candidato a superficie de inferencia ya que el futuro de la informática es la computación cuántica, y con **ALBLI** será computación cuántica difusa.

APÉNDICE A.

Para calcular los puntos máximos en **ALBHI** hay que recurrir a la inversa de la tangente hiperbólica.

Para la función de pertenencia $\mu_L(x_0)$ se calcula de la siguiente forma:

$$\mu_L(x_0) = \text{lejos}(\text{pmL} \cdot x_0)$$

$$\mu_L(x_0) = \left|\tanh(\text{pmL} \cdot x_0)\right|$$

$$\text{pmL} \cdot x_0 = \text{atanh}(\mu_L)$$

$$\mathbf{pmL = \frac{atanh(\mu_L)}{x_0}}$$

Para $\mu_L(x_0)$ = 0,99 el numerador es 2,647.

$$\text{pmL} = \frac{2{,}647}{x_0}$$

Para $\mu_L(x_0)$ = 0,999 el numerador es 3,800.

$$\text{pmL} = \frac{3{,}800}{x_0}$$

Para $\mu_L(x_0)$ = 0,9999 el numerador es 4,952.

$$pmL = \frac{4{,}952}{x_0}$$

Para $\mu_L(x_0) = 0{,}5$ el numerador es 0,549.

$$pmL = \frac{0{,}549}{x_0}$$

Basta $\mu_L(x_0) = 0{,}999$ para que la función se adapte óptimamente al problema, por tanto el numerador es 3,800.

$$\mathbf{pmL = \frac{3{,}800}{x_0}}$$

Para la función de pertenencia $\mu_C(x_0)$ se calcula de la siguiente forma:

$$\mu_C(x_0) = \text{cerca}(pmC \cdot x_0)$$

$$\mu_C(x_0) = 1 - |\tanh(pmC \cdot x_0)|$$

$$pmC \cdot x_0 = \text{atanh}(1 - \mu_C)$$

$$\mathbf{pmC = \frac{atanh(1 - \mu_C)}{x_0}}$$

Para $\mu_C(x_0) = 0{,}01$ el numerador es 2,647.

$$pmC = \frac{2{,}647}{x_0}$$

Para $\mu_C(x_0) = 0{,}001$ el numerador es 3,800.

$$pmC = \frac{3{,}800}{x_0}$$

Para $\mu_C(x_0) = 0{,}0001$ el numerador es 4,952.

$$pmC = \frac{4{,}952}{x_0}$$

Para $\mu_C(x_0) = 0{,}5$ el numerador es 0,549.

$$pmC = \frac{0{,}549}{x_0}$$

Basta $\mu_C(x_0) = 0{,}001$ para que la función se adapte óptimamente al problema, por tanto el numerador es 3,800.

$$\mathbf{pmC = \frac{3{,}800}{x_0}}$$

Ambas funciones de pertenencia son complementarias:

$$\mu_L(x) = 1 - \mu_C(x)$$

El cálculo de los puntos máximos en **ALBLI** es más sencillo que en **ALBHI**. Cómo es $|x|$ con que nos constriñamos a valores x > 0 es suficiente.

Para la variable lingüística lejos el punto máximo se calcula:

$$\mu_L(x_0) = \text{lejos}(\text{pmL} \cdot x_0)$$

$$1 = \text{lejos}(\text{pmL} \cdot x_0)$$

$$1 = |\text{pmL} \cdot x_0|$$

$$\forall x>0$$

$$1 = \text{pmL} \cdot x_0$$

$$\mathbf{pmL = \frac{1}{x_0}}$$

Para la variable lingüística cerca el punto máximo se calcula:

$$\mu_C(x_0) = \text{cerca}(\text{pmC} \cdot x_0)$$

$$0 = \text{cerca}(\text{pmC} \cdot x_0)$$

$$0 = 1 - |\text{pmC} \cdot x_0|$$

$$\forall x>0$$

$$0 = 1 - pmC \cdot x_0$$

$$pmC \cdot x_0 = 1$$

$$\mathbf{pmC = \frac{1}{x_0}}$$

Los puntos máximos representan la pendiente de la recta entre 0 y x_0.

Ambas funciones de pertenencia son complementarias:

$$\mu_L(x) = 1 - \mu_C(x)$$

Para que **ALBLI** tenga un comportamiento similar a **ALBHI** los puntos máximos una vez calculados hay que dividirlos entre 1,1552. La recta de **ALBLI** corta a la de **ALBHI** en μ(x) = 0,6; y es el resultado de resolver la igualdad siguiente: $k \cdot 0,6 = \operatorname{atanh}(0,6)$.

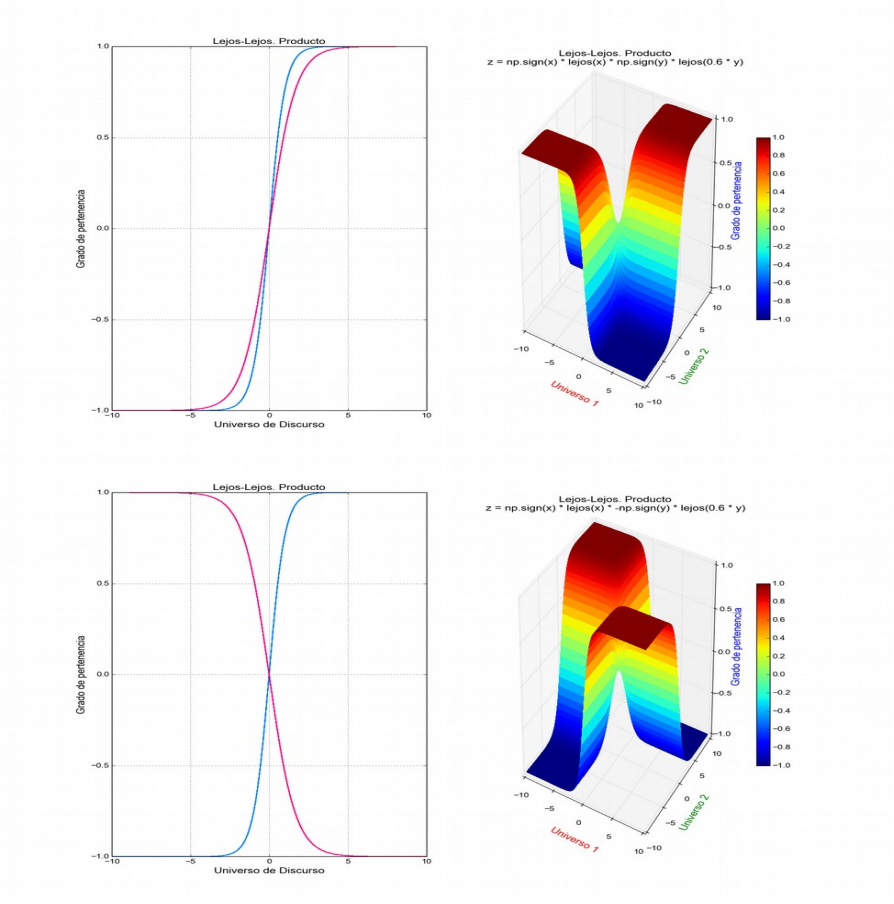
$$k = \frac{\operatorname{atanh}(0,6)}{0,6}.$$

APÉNDICE B.

Se van a representar 14 superficies de control, de **ALBHI**, μ_L-μ_L correspondientes a su regla de inferencia, usando los signos de los universos de discurso en ambas variables de entrada.

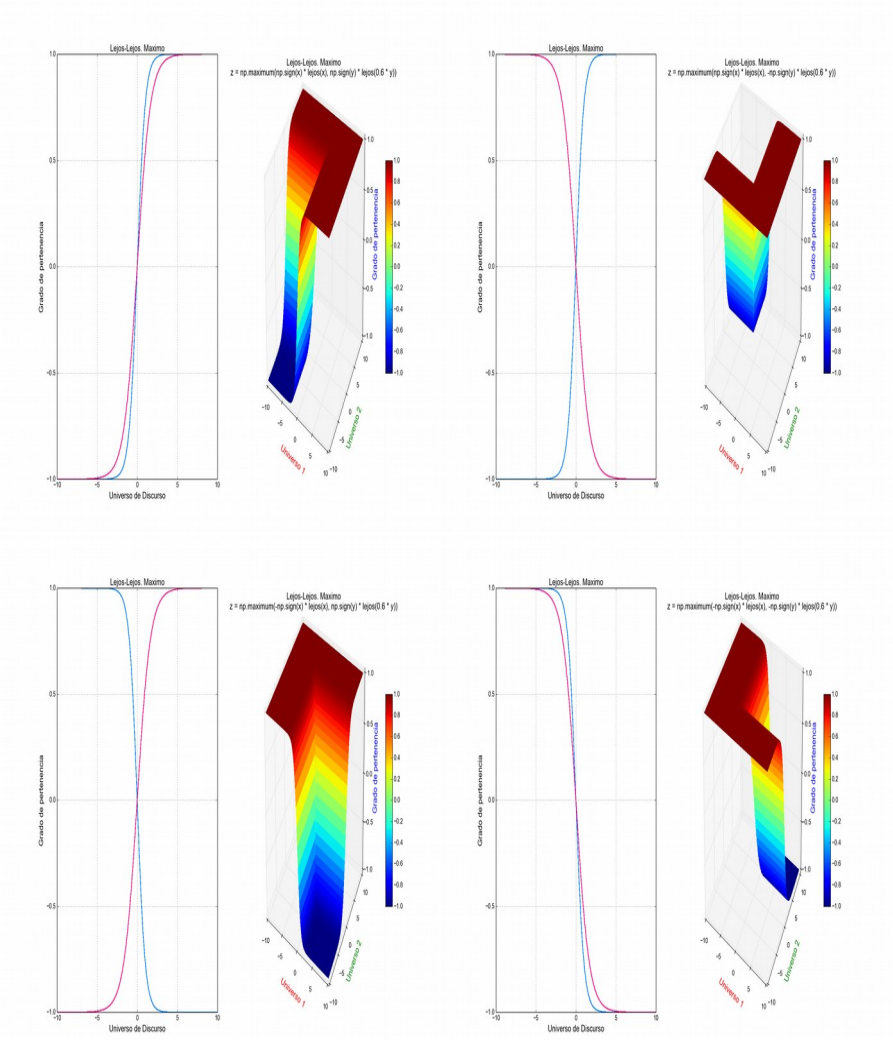
Para el producto de las variables de entrada tenemos:

$$\text{salida}(x_1, x_2) = \text{signo}(x_1) \cdot \mu_L(x_1) \cdot \text{signo}(x_2) \cdot \mu_L(x_2)$$
$$\text{salida}(x_1, x_2) = \text{signo}(x_1) \cdot \mu_L(x_1) \cdot -\text{signo}(x_2) \cdot \mu_L(x_2)$$



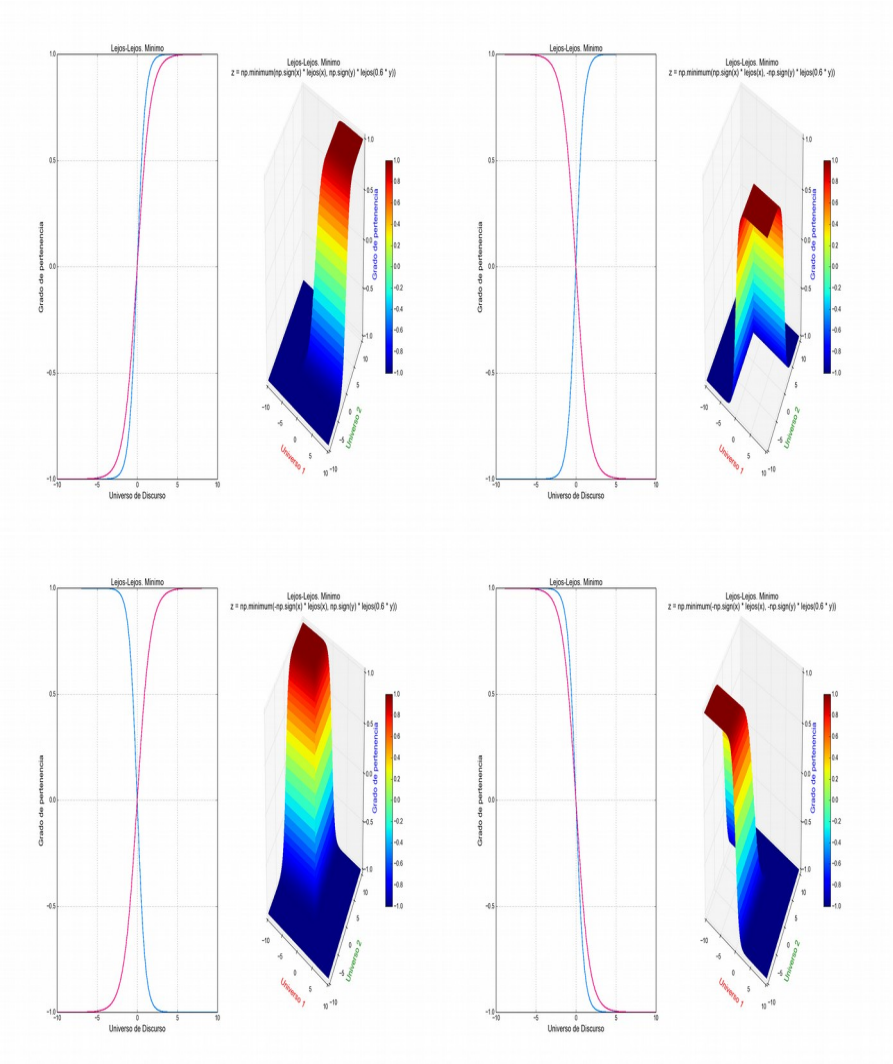
Para el máximo de las variables de entrada tenemos:

$$\text{salida}(x_1,\ x_2) = \max(\text{signo}(x_1) \cdot \mu_L(x_1),\ \text{signo}(x_2) \cdot \mu_L(x_2))$$
$$\text{salida}(x_1,\ x_2) = \max(\text{signo}(x_1) \cdot \mu_L(x_1),\ -\text{signo}(x_2) \cdot \mu_L(x_2))$$
$$\text{salida}(x_1,\ x_2) = \max(-\text{signo}(x_1) \cdot \mu_L(x_1),\ \text{signo}(x_2) \cdot \mu_L(x_2))$$
$$\text{salida}(x_1,\ x_2) = \max(-\text{signo}(x_1) \cdot \mu_L(x_1),\ -\text{signo}(x_2) \cdot \mu_L(x_2))$$



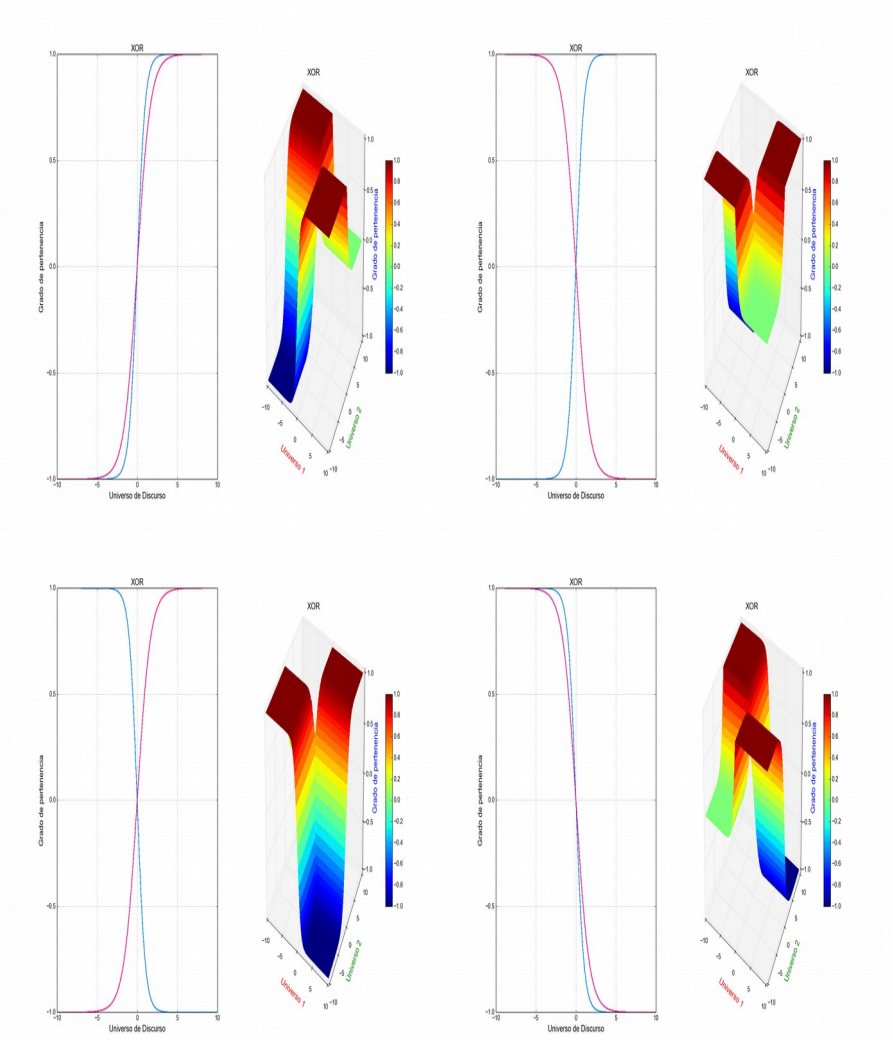
Para el mínimo de las variables de entrada tenemos:

$$\text{salida}(x_1, x_2) = \min(\text{signo}(x_1) \cdot \mu_L(x_1),\ \text{signo}(x_2) \cdot \mu_L(x_2))$$
$$\text{salida}(x_1, x_2) = \min(\text{signo}(x_1) \cdot \mu_L(x_1),\ -\text{signo}(x_2) \cdot \mu_L(x_2))$$
$$\text{salida}(x_1, x_2) = \min(-\text{signo}(x_1) \cdot \mu_L(x_1),\ \text{signo}(x_2) \cdot \mu_L(x_2))$$
$$\text{salida}(x_1, x_2) = \min(-\text{signo}(x_1) \cdot \mu_L(x_1),\ -\text{signo}(x_2) \cdot \mu_L(x_2))$$



Para XOR difuso de las variables de entrada tenemos:

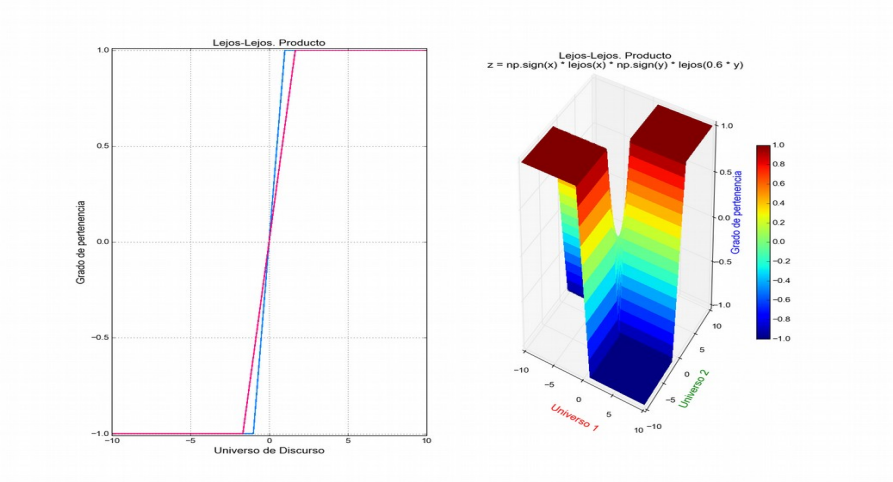
$$\text{salida}(x_1, x_2) = \text{XOR}\ \{\text{signo}(x_1) \cdot \mu(x_1),\ \text{signo}(x_2) \cdot \mu(x_2)\}$$
$$\text{salida}(x_1, x_2) = \text{XOR}\ \{-\text{signo}(x_1) \cdot \mu(x_1),\ \text{signo}(x_2) \cdot \mu(x_2)\}$$
$$\text{salida}(x_1, x_2) = \text{XOR}\ \{\text{signo}(x_1) \cdot \mu(x_1),\ -\text{signo}(x_2) \cdot \mu(x_2)\}$$
$$\text{salida}(x_1, x_2) = \text{XOR}\ \{-\text{signo}(x_1) \cdot \mu(x_1),\ -\text{signo}(x_2) \cdot \mu(x_2)\}$$

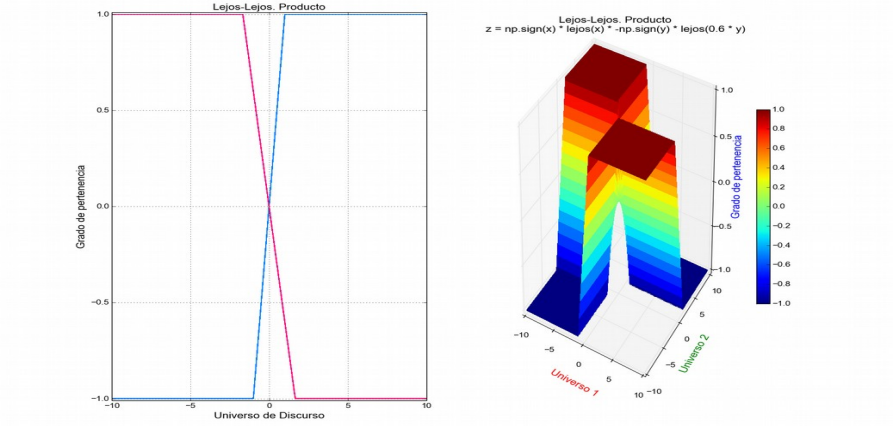


Se van a representar 14 superficies de control, de **ALBLI**, μ_L-μ_L correspondientes a su regla de inferencia, usando los signos de los universos de discurso en ambas variables de entrada.

Para el producto de las variables de entrada tenemos:

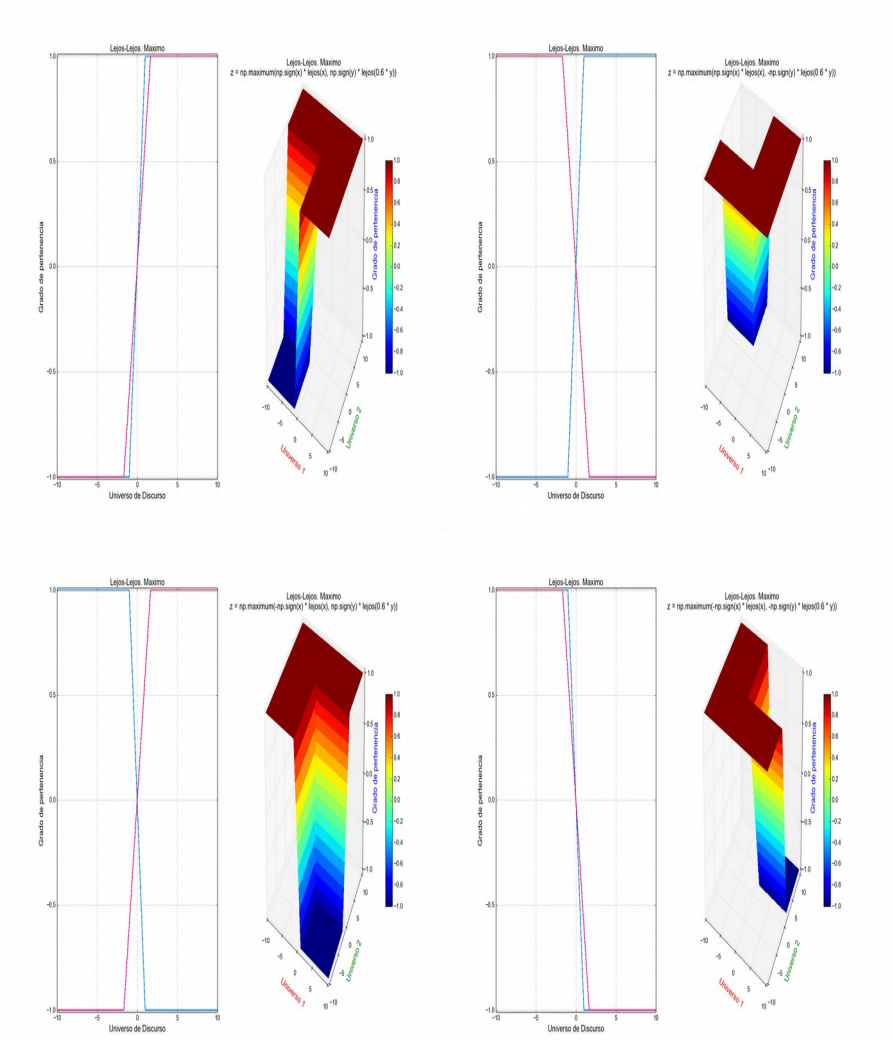
$$\text{salida}(x_1, x_2) = \text{signo}(x_1) \cdot \mu_L(x_1) \cdot \text{signo}(x_2) \cdot \mu_L(x_2)$$
$$\text{salida}(x_1, x_2) = \text{signo}(x_1) \cdot \mu_L(x_1) \cdot -\text{signo}(x_2) \cdot \mu_L(x_2)$$





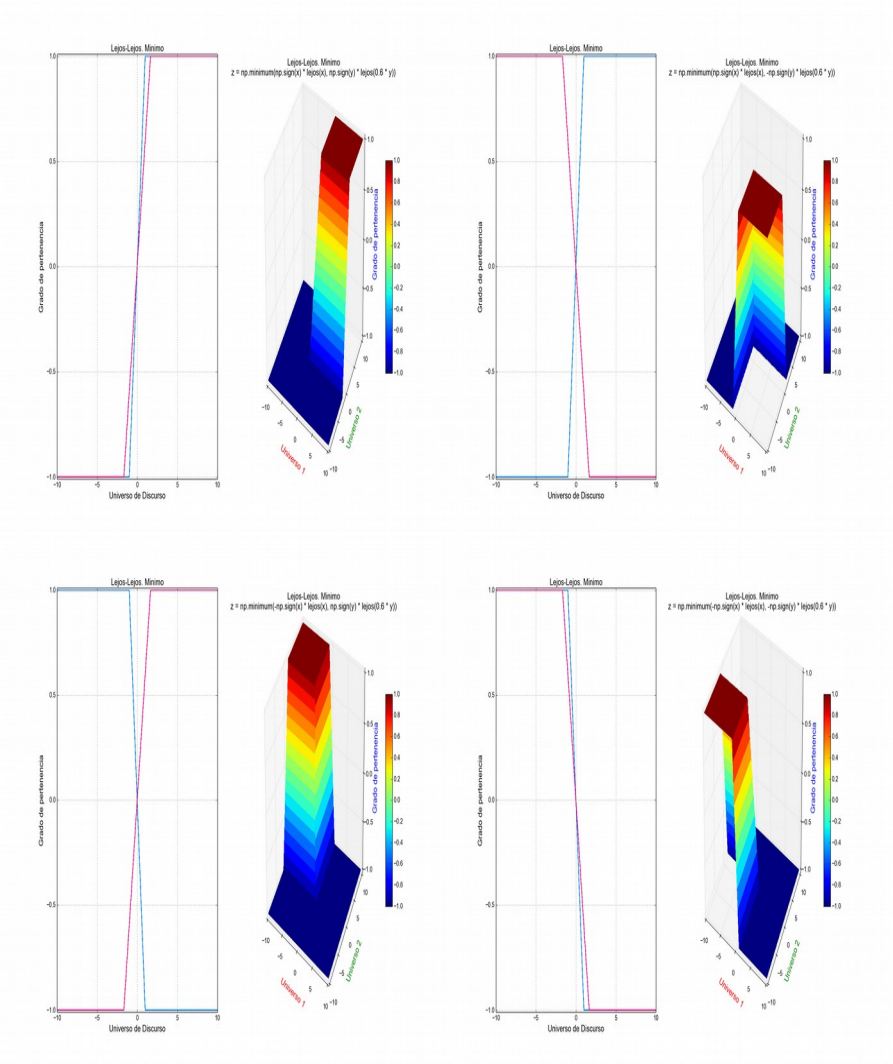
Para el máximo de las variables de entrada tenemos:

$$\text{salida}(x_1, x_2) = \max(\text{signo}(x_1) \cdot \mu_L(x_1), \text{signo}(x_2) \cdot \mu_L(x_2))$$
$$\text{salida}(x_1, x_2) = \max(\text{signo}(x_1) \cdot \mu_L(x_1), -\text{signo}(x_2) \cdot \mu_L(x_2))$$
$$\text{salida}(x_1, x_2) = \max(-\text{signo}(x_1) \cdot \mu_L(x_1), \text{signo}(x_2) \cdot \mu_L(x_2))$$
$$\text{salida}(x_1, x_2) = \max(-\text{signo}(x_1) \cdot \mu_L(x_1), -\text{signo}(x_2) \cdot \mu_L(x_2))$$



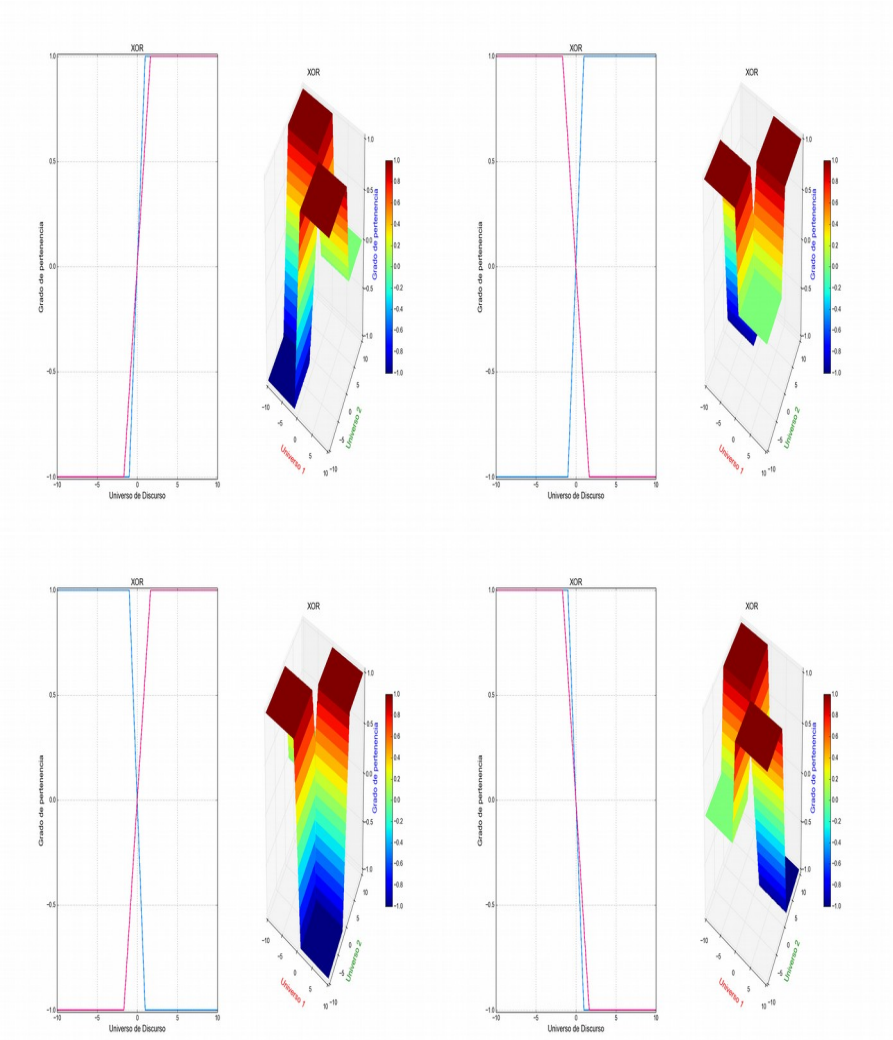
Para el mínimo de las variables de entrada tenemos:

$$\text{salida}(x_1, x_2) = \min(\text{signo}(x_1) \cdot \mu_L(x_1),\ \text{signo}(x_2) \cdot \mu_L(x_2))$$
$$\text{salida}(x_1, x_2) = \min(\text{signo}(x_1) \cdot \mu_L(x_1),\ -\text{signo}(x_2) \cdot \mu_L(x_2))$$
$$\text{salida}(x_1, x_2) = \min(-\text{signo}(x_1) \cdot \mu_L(x_1),\ \text{signo}(x_2) \cdot \mu_L(x_2))$$
$$\text{salida}(x_1, x_2) = \min(-\text{signo}(x_1) \cdot \mu_L(x_1),\ -\text{signo}(x_2) \cdot \mu_L(x_2))$$



Para XOR difuso de las variables de entrada tenemos:

$$\text{salida}(x_1, x_2) = \text{XOR}\ \{\text{signo}(x_1) \cdot \mu(x_1),\ \text{signo}(x_2) \cdot \mu(x_2)\}$$
$$\text{salida}(x_1, x_2) = \text{XOR}\ \{-\text{signo}(x_1) \cdot \mu(x_1),\ \text{signo}(x_2) \cdot \mu(x_2)\}$$
$$\text{salida}(x_1, x_2) = \text{XOR}\ \{\text{signo}(x_1) \cdot \mu(x_1),\ -\text{signo}(x_2) \cdot \mu(x_2)\}$$
$$\text{salida}(x_1, x_2) = \text{XOR}\ \{-\text{signo}(x_1) \cdot \mu(x_1),\ -\text{signo}(x_2) \cdot \mu(x_2)\}$$



APÉNDICE C.

Se presentan 14 programas en Python que representan las superficies de control del apéndice B.

Para el producto de las variables de entrada tenemos:

$$\text{salida}(x_1, x_2) = \text{signo}(x_1) \cdot \mu_L(x_1) \cdot \text{signo}(x_2) \cdot \mu_L(x_2)$$

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#ALBHI. PRODUCTO.
#z = np.sign(x) * lejos(x) * np.sign(y) * lejos(0.6 * y)
#JOSE MARIA MOLINA SANCHEZ


import numpy as np #importo Numpy
import matplotlib.pyplot as plt #importo MatPlolib
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm


# funcion borrosa lejos de
def lejos(x):
 return abs(np.tanh(x))


trozos = 1e+3
x = np.linspace(-10, 10, trozos)
y = np.linspace(-10, 10, trozos)
x , y = np.meshgrid(x, y)
z = np.sign(x) * lejos(x) * np.sign(y) * lejos(0.6 * y)


plt.ion()
fig = plt.figure()
ax = fig.add_subplot(121)
plt.plot(y, np.sign(y) * lejos(y), c=(0, 0.5, 1))
plt.plot(y, np.sign(y) * lejos(0.6 * y), c=(1, 0, 0.5))
plt.xlim(-10, 10)
plt.ylim(-1.01, 1.01)
plt.xlabel("Universo de Discurso", fontsize = 15)
plt.ylabel("Grado de pertenencia", fontsize = 15)
plt.title('Lejos-Lejos. Producto', fontsize = 15)
plt.grid(True)

ax = fig.add_subplot(122, projection='3d')
p = ax.plot_surface(x, y, z, cmap = cm.jet, linewidth = 0, antialiased = False)

ax.set_xlim(-10, 10)
ax.set_ylim(-10, 10)
ax.set_zlim(-1, 1)

ax.set_xlabel('Universo 1', fontsize = 15, color = 'red')
ax.set_ylabel('Universo 2', fontsize = 15, color = 'green')
ax.set_zlabel('Grado de pertenencia', fontsize = 15, color = 'blue')
ax.set_title('Lejos-Lejos. Producto\nz = np.sign(x) * lejos(x) * np.sign(y) * lejos(0.6 * y)\n',
fontsize = 15)

#ax.view_init(10, 300)
fig.colorbar(p, shrink = 0.5)

plt.show()
```

$$\text{salida}(x_{1,}\ x_2) = \text{signo}(x_1) \cdot \mu_L(x_1) \cdot -\text{signo}(x_2) \cdot \mu_L(x_2)$$

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#ALBHI. PRODUCTO.
#z = np.sign(x) * lejos(x) * -np.sign(y) * lejos(0.6 * y)
#JOSE MARIA MOLINA SANCHEZ


import numpy as np #importo Numpy
import matplotlib.pyplot as plt #importo MatPlolib
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm


# funcion borrosa lejos de
def lejos(x):
 return abs(np.tanh(x))


trozos = 1e+3
x = np.linspace(-10, 10, trozos)
y = np.linspace(-10, 10, trozos)
x , y = np.meshgrid(x, y)
z = np.sign(x) * lejos(x) * -np.sign(y) * lejos(0.6 * y)


plt.ion()
fig = plt.figure()
ax = fig.add_subplot(121)
plt.plot(y, np.sign(y) * lejos(y), c=(0, 0.5, 1))
plt.plot(y, np.sign(y) * lejos(0.6 * y), c=(1, 0, 0.5))
plt.xlim(-10, 10)
plt.ylim(-1.01, 1.01)
plt.xlabel("Universo de Discurso", fontsize = 15)
plt.ylabel("Grado de pertenencia", fontsize = 15)
plt.title('Lejos-Lejos. Producto', fontsize = 15)
plt.grid(True)

ax = fig.add_subplot(122, projection='3d')
p = ax.plot_surface(x, y, z, cmap = cm.jet, linewidth = 0, antialiased =
False)

ax.set_xlim(-10, 10)
ax.set_ylim(-10, 10)
ax.set_zlim(-1, 1)

ax.set_xlabel('Universo 1', fontsize = 15, color = 'red')
ax.set_ylabel('Universo 2', fontsize = 15, color = 'green')
ax.set_zlabel('Grado de pertenencia', fontsize = 15, color = 'blue')
ax.set_title('Lejos-Lejos. Producto\nz = np.sign(x) * lejos(x) * np.sign(y) *
lejos(0.6 * y)\n', fontsize = 15)

#ax.view_init(10, 300)
fig.colorbar(p, shrink = 0.5)

plt.show()
```

Para el máximo de las variables de entrada tenemos:

$$\text{salida}(x_{1,}\ x_2) = \max(\text{signo}(x_1) \cdot \mu_L(x_1),\ \text{signo}(x_2) \cdot \mu_L(x_2))$$

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#ALBHI. MAXIMO.
#z = np.maximum(np.sign(x) * lejos(x), np.sign(y) * lejos(0.6 * y))
#JOSE MARIA MOLINA SANCHEZ


import numpy as np #importo Numpy
import matplotlib.pyplot as plt #importo MatPlolib
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm


# funcion borrosa lejos de
def lejos(x):
 return abs(np.tanh(x))


trozos = 1e+3
x = np.linspace(-10, 10, trozos)
y = np.linspace(-10, 10, trozos)
x , y = np.meshgrid(x, y)
z = np.maximum(np.sign(x) * lejos(x), np.sign(y) * lejos(0.6 * y))


plt.ion()
fig = plt.figure()
ax = fig.add_subplot(121)
plt.plot(y, np.sign(y) * lejos(y), c=(0, 0.5, 1))
plt.plot(y, np.sign(y) * lejos(0.6 * y), c=(1, 0, 0.5))
plt.xlabel("Universo de Discurso", fontsize = 15)
plt.ylabel("Grado de pertenencia", fontsize = 15)
plt.title('Lejos-Lejos. Maximo', fontsize = 15)
plt.grid(True)

ax = fig.add_subplot(122, projection='3d')
p = ax.plot_surface(x, y, z, cmap = cm.jet, linewidth = 0 antialiased =
False)

ax.set_xlim(-10, 10)
ax.set_ylim(-10, 10)
ax.set_zlim(-1, 1)

ax.set_xlabel('Universo 1', fontsize = 15, color = 'red')
ax.set_ylabel('Universo 2', fontsize = 15, color = 'green')
ax.set_zlabel('Grado de pertenencia', fontsize = 15, color = 'blue')
ax.set_title('Lejos-Lejos. Maximo\nz = np.maximum(np.sign(x) * lejos(x),
np.sign(y) * lejos(0.6 * y))\n', fontsize = 15)

#ax.view_init(10, 225)
fig.colorbar(p, shrink = 0.5)

plt.show()
```

$$\text{salida}(x_1, x_2) = \max(\text{signo}(x_1) \cdot \mu_L(x_1), -\text{signo}(x_2) \cdot \mu_L(x_2))$$

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#ALBHI. MAXIMO.
#z = np.maximum(np.sign(x) * lejos(x), -np.sign(y) * lejos(0.6 * y))
#JOSE MARIA MOLINA SANCHEZ


import numpy as np #importo Numpy
import matplotlib.pyplot as plt #importo MatPlolib
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm


# funcion borrosa lejos de
def lejos(x):
 return abs(np.tanh(x))


trozos = 1e+3
x = np.linspace(-10, 10, trozos)
y = np.linspace(-10, 10, trozos)
x , y = np.meshgrid(x, y)
z = np.maximum(np.sign(x) * lejos(x), -np.sign(y) * lejos(0.6 * y))


plt.ion()
fig = plt.figure()
ax = fig.add_subplot(121)
plt.plot(y, np.sign(y) * lejos(y), c=(0, 0.5, 1))
plt.plot(y, -np.sign(y) * lejos(0.6 * y), c=(1, 0, 0.5))
plt.xlabel("Universo de Discurso", fontsize = 15)
plt.ylabel("Grado de pertenencia", fontsize = 15)
plt.title('Lejos-Lejos. Maximo', fontsize = 15)
plt.grid(True)

ax = fig.add_subplot(122, projection='3d')
p = ax.plot_surface(x, y, z, cmap = cm.jet, linewidth = 0 antialiased =
False)

ax.set_xlim(-10, 10)
ax.set_ylim(-10, 10)
ax.set_zlim(-1, 1)

ax.set_xlabel('Universo 1', fontsize = 15, color = 'red')
ax.set_ylabel('Universo 2', fontsize = 15, color = 'green')
ax.set_zlabel('Grado de pertenencia', fontsize = 15, color = 'blue')
ax.set_title('Lejos-Lejos. Maximo\nz = np.maximum(np.sign(x) *
lejos(x), -np.sign(y) * lejos(0.6 * y))\n', fontsize = 15)

#ax.view_init(10, 300)
fig.colorbar(p, shrink = 0.5)

plt.show()
```

$$\text{salida}(x_{1,}\ x_2) = \max(-\text{signo}(x_1) \cdot \mu_L(x_1),\ \text{signo}(x_2) \cdot \mu_L(x_2))$$

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#ALBHI. MAXIMO.
#z = np.maximum(-np.sign(x) * lejos(x), np.sign(y) * lejos(0.6 * y))
#JOSE MARIA MOLINA SANCHEZ


import numpy as np #importo Numpy
import matplotlib.pyplot as plt #importo MatPlolib
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm


# funcion borrosa lejos de
def lejos(x):
 return abs(np.tanh(x))


trozos = 1e+3
x = np.linspace(-10, 10, trozos)
y = np.linspace(-10, 10, trozos)
x , y = np.meshgrid(x, y)
z = np.maximum(-np.sign(x) * lejos(x), np.sign(y) * lejos(0.6 * y))


plt.ion()
fig = plt.figure()
ax = fig.add_subplot(121)
plt.plot(y, -np.sign(y) * lejos(y), c=(0, 0.5, 1))
plt.plot(y, np.sign(y) * lejos(0.6 * y), c=(1, 0, 0.5))
plt.xlabel("Universo de Discurso", fontsize = 15)
plt.ylabel("Grado de pertenencia", fontsize = 15)
plt.title('Lejos-Lejos. Maximo', fontsize = 15)
plt.grid(True)

ax = fig.add_subplot(122, projection='3d')
p = ax.plot_surface(x, y, z, cmap = cm.jet, linewidth = 0 antialiased =
False)

ax.set_xlim(-10, 10)
ax.set_ylim(-10, 10)
ax.set_zlim(-1, 1)

ax.set_xlabel('Universo 1', fontsize = 15, color = 'red')
ax.set_ylabel('Universo 2', fontsize = 15, color = 'green')
ax.set_zlabel('Grado de pertenencia', fontsize = 15, color = 'blue')
ax.set_title('Lejos-Lejos. Maximo\nz = np.maximum(-np.sign(x) *
lejos(x), np.sign(y) * lejos(0.6 * y))\n', fontsize = 15)

#ax.view_init(10, 300)
fig.colorbar(p, shrink = 0.5)

plt.show()
```

$$\text{salida}(x_{1,}\ x_2) = \max(-\text{signo}(x_1) \cdot \mu_L(x_1),\ -\text{signo}(x_2) \cdot \mu_L(x_2))$$

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#ALBHI. MAXIMO.
#z = np.maximum(-np.sign(x) * lejos(x), -np.sign(y) * lejos(0.6 * y))
#JOSE MARIA MOLINA SANCHEZ


import numpy as np #importo Numpy
import matplotlib.pyplot as plt #importo MatPlolib
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm


# funcion borrosa lejos de
def lejos(x):
 return abs(np.tanh(x))


trozos = 1e+3
x = np.linspace(-10, 10, trozos)
y = np.linspace(-10, 10, trozos)
x , y = np.meshgrid(x, y)
z = np.maximum(-np.sign(x) * lejos(x), -np.sign(y) * lejos(0.6 * y))


plt.ion()
fig = plt.figure()
ax = fig.add_subplot(121)
plt.plot(y, -np.sign(y) * lejos(y), c=(0, 0.5, 1))
plt.plot(y, -np.sign(y) * lejos(0.6 * y), c=(1, 0, 0.5))
plt.xlabel("Universo de Discurso", fontsize = 15)
plt.ylabel("Grado de pertenencia", fontsize = 15)
plt.title('Lejos-Lejos. Maximo', fontsize = 15)
plt.grid(True)

ax = fig.add_subplot(122, projection='3d')
p = ax.plot_surface(x, y, z, cmap = cm.jet, linewidth = 0 antialiased =
False)

ax.set_xlim(-10, 10)
ax.set_ylim(-10, 10)
ax.set_zlim(-1, 1)

ax.set_xlabel('Universo 1', fontsize = 15, color = 'red')
ax.set_ylabel('Universo 2', fontsize = 15, color = 'green')
ax.set_zlabel('Grado de pertenencia', fontsize = 15, color = 'blue')
ax.set_title('Lejos-Lejos. Maximo\nz = np.maximum(-np.sign(x) *
lejos(x), -np.sign(y) * lejos(0.6 * y))\n', fontsize = 15)

#ax.view_init(10, 300)
fig.colorbar(p, shrink = 0.5)

plt.show()
```

Para el mínimo de las variables de entrada tenemos:

$$\text{salida}(x_1, x_2) = \min(\text{signo}(x_1) \cdot \mu_L(x_1),\ \text{signo}(x_2) \cdot \mu_L(x_2))$$

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#ALBHI. MINIMO.
#z = np.minimum(np.sign(x) * lejos(x), np.sign(y) * lejos(0.6 * y))
#JOSE MARIA MOLINA SANCHEZ


import numpy as np #importo Numpy
import matplotlib.pyplot as plt #importo MatPlolib
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm


# funcion borrosa lejos de
def lejos(x):
 return abs(np.tanh(x))


trozos = 1e+3
x = np.linspace(-10, 10, trozos)
y = np.linspace(-10, 10, trozos)
x , y = np.meshgrid(x, y)
z = np.minimum(np.sign(x) * lejos(x), np.sign(y) * lejos(0.6 * y))


plt.ion()
fig = plt.figure()
ax = fig.add_subplot(121)
plt.plot(y, np.sign(y) * lejos(y), c=(0, 0.5, 1))
plt.plot(y, np.sign(y) * lejos(0.6 * y), c=(1, 0, 0.5))
plt.xlabel("Universo de Discurso", fontsize = 15)
plt.ylabel("Grado de pertenencia", fontsize = 15)
plt.title('Lejos-Lejos. Minimo', fontsize = 15)
plt.grid(True)

ax = fig.add_subplot(122, projection='3d')
p = ax.plot_surface(x, y, z, cmap = cm.jet, linewidth = 0 antialiased =
False)

ax.set_xlim(-10, 10)
ax.set_ylim(-10, 10)
ax.set_zlim(-1, 1)

ax.set_xlabel('Universo 1', fontsize = 15, color = 'red')
ax.set_ylabel('Universo 2', fontsize = 15, color = 'green')
ax.set_zlabel('Grado de pertenencia', fontsize = 15, color = 'blue')
ax.set_title('Lejos-Lejos. Minimo\nz = np.minimum(np.sign(x) * lejos(x),
np.sign(y) * lejos(0.6 * y))\n', fontsize = 15)

#ax.view_init(10, 300)
fig.colorbar(p, shrink = 0.5)

plt.show()
```

$$\text{salida}(x_1, x_2) = \min(\text{signo}(x_1) \cdot \mu_L(x_1), -\text{signo}(x_2) \cdot \mu_L(x_2))$$

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#ALBHI. MINIMO.
#z = np.minimum(np.sign(x) * lejos(x), -np.sign(y) * lejos(0.6 * y))
#JOSE MARIA MOLINA SANCHEZ


import numpy as np #importo Numpy
import matplotlib.pyplot as plt #importo MatPlolib
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm


# funcion borrosa lejos de
def lejos(x):
 return abs(np.tanh(x))


trozos = 1e+3
x = np.linspace(-10, 10, trozos)
y = np.linspace(-10, 10, trozos)
x , y = np.meshgrid(x, y)
z = np.minimum(np.sign(x) * lejos(x), -np.sign(y) * lejos(0.6 * y))


plt.ion()
fig = plt.figure()
ax = fig.add_subplot(121)
plt.plot(y, np.sign(y) * lejos(y), c=(0, 0.5, 1))
plt.plot(y, -np.sign(y) * lejos(0.6 * y), c=(1, 0, 0.5))
plt.xlabel("Universo de Discurso", fontsize = 15)
plt.ylabel("Grado de pertenencia", fontsize = 15)
plt.title('Lejos-Lejos. Minimo', fontsize = 15)
plt.grid(True)

ax = fig.add_subplot(122, projection='3d')
p = ax.plot_surface(x, y, z, cmap = cm.jet, linewidth = 0 antialiased =
False)

ax.set_xlim(-10, 10)
ax.set_ylim(-10, 10)
ax.set_zlim(-1, 1)

ax.set_xlabel('Universo 1', fontsize = 15, color = 'red')
ax.set_ylabel('Universo 2', fontsize = 15, color = 'green')
ax.set_zlabel('Grado de pertenencia', fontsize = 15, color = 'blue')
ax.set_title('Lejos-Lejos. Minimo\nz = np.minimum(np.sign(x) *
lejos(x), -np.sign(y) * lejos(0.6 * y))\n', fontsize = 15)

#ax.view_init(10, 300)
fig.colorbar(p, shrink = 0.5)

plt.show()
```

$$\text{salida}(x_{1,}\ x_2) = \min(-\text{signo}(x_1) \cdot \mu_L(x_1),\ \text{signo}(x_2) \cdot \mu_L(x_2))$$

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#ALBHI. MINIMO.
#z = np.minimum(-np.sign(x) * lejos(x), np.sign(y) * lejos(0.6 * y))
#JOSE MARIA MOLINA SANCHEZ


import numpy as np #importo Numpy
import matplotlib.pyplot as plt #importo MatPlolib
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm


# funcion borrosa lejos de
def lejos(x):
 return abs(np.tanh(x))


trozos = 1e+3
x = np.linspace(-10, 10, trozos)
y = np.linspace(-10, 10, trozos)
x , y = np.meshgrid(x, y)
z = np.minimum(-np.sign(x) * lejos(x), np.sign(y) * lejos(0.6 * y))


plt.ion()
fig = plt.figure()
ax = fig.add_subplot(121)
plt.plot(y, -np.sign(y) * lejos(y), c=(0, 0.5, 1))
plt.plot(y, np.sign(y) * lejos(0.6 * y), c=(1, 0, 0.5))
plt.xlabel("Universo de Discurso", fontsize = 15)
plt.ylabel("Grado de pertenencia", fontsize = 15)
plt.title('Lejos-Lejos. Minimo', fontsize = 15)
plt.grid(True)

ax = fig.add_subplot(122, projection='3d')
p = ax.plot_surface(x, y, z, cmap = cm.jet, linewidth = 0 antialiased =
False)

ax.set_xlim(-10, 10)
ax.set_ylim(-10, 10)
ax.set_zlim(-1, 1)

ax.set_xlabel('Universo 1', fontsize = 15, color = 'red')
ax.set_ylabel('Universo 2', fontsize = 15, color = 'green')
ax.set_zlabel('Grado de pertenencia', fontsize = 15, color = 'blue')
ax.set_title('Lejos-Lejos. Minimo\nz = np.minimum(-np.sign(x) *
lejos(x), np.sign(y) * lejos(0.6 * y))\n', fontsize = 15)

#ax.view_init(10, 300)
fig.colorbar(p, shrink = 0.5)

plt.show()
```

$$\text{salida}(x_1, x_2) = \min(-\text{signo}(x_1) \cdot \mu_L(x_1), -\text{signo}(x_2) \cdot \mu_L(x_2))$$

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#ALBHI. MINIMO.
#z = np.minimum(-np.sign(x) * lejos(x), -np.sign(y) * lejos(0.6 * y))
#JOSE MARIA MOLINA SANCHEZ


import numpy as np #importo Numpy
import matplotlib.pyplot as plt #importo MatPlolib
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm


# funcion borrosa lejos de
def lejos(x):
 return abs(np.tanh(x))


trozos = 1e+3
x = np.linspace(-10, 10, trozos)
y = np.linspace(-10, 10, trozos)
x , y = np.meshgrid(x, y)
z = np.minimum(-np.sign(x) * lejos(x), -np.sign(y) * lejos(0.6 * y))


plt.ion()
fig = plt.figure()
ax = fig.add_subplot(121)
plt.plot(y, -np.sign(y) * lejos(y), c=(0, 0.5, 1))
plt.plot(y, -np.sign(y) * lejos(0.6 * y), c=(1, 0, 0.5))
plt.xlabel("Universo de Discurso", fontsize = 15)
plt.ylabel("Grado de pertenencia", fontsize = 15)
plt.title('Lejos-Lejos. Minimo', fontsize = 15)
plt.grid(True)

ax = fig.add_subplot(122, projection='3d')
p = ax.plot_surface(x, y, z, cmap = cm.jet, linewidth = 0 antialiased =
False)

ax.set_xlim(-10, 10)
ax.set_ylim(-10, 10)
ax.set_zlim(-1, 1)

ax.set_xlabel('Universo 1', fontsize = 15, color = 'red')
ax.set_ylabel('Universo 2', fontsize = 15, color = 'green')
ax.set_zlabel('Grado de pertenencia', fontsize = 15, color = 'blue')
ax.set_title('Lejos-Lejos. Minimo\nz = np.minimum(-np.sign(x) *
lejos(x), -np.sign(y) * lejos(0.6 * y))\n', fontsize = 15)

#ax.view_init(10, 300)
fig.colorbar(p, shrink = 0.5)

plt.show()
```

Para XOR difuso de las variables de entrada tenemos:

$$\text{salida}(x_1, x_2) = \text{XOR}\left\{\text{signo}(x_1) \cdot \mu(x_1), \text{signo}(x_2) \cdot \mu(x_2)\right\}$$

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#ALBHI. XOR.
#z = np.maximum(np.minimum(np.sign(x) * lejos(x), 1 - np.sign(y) * lejos(0.6
* y)), np.minimum(1 - np.sign(x) * lejos(x), np.sign(y) * lejos(0.6 * y)))
#JOSE MARIA MOLINA SANCHEZ


import numpy as np #importo Numpy
import matplotlib.pyplot as plt #importo MatPlolib
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm


# funcion borrosa lejos de
def lejos(x):
 return abs(np.tanh(x))


trozos = 1e+3
x = np.linspace(-10, 10, trozos)
y = np.linspace(-10, 10, trozos)
x , y = np.meshgrid(x, y)
z = np.maximum(np.minimum(np.sign(x) * lejos(x), 1 - np.sign(y) * lejos(0.6 *
y)), np.minimum(1 - np.sign(x) * lejos(x), np.sign(y) * lejos(0.6 * y)))


plt.ion()
fig = plt.figure()
ax = fig.add_subplot(121)
plt.plot(y, np.sign(y) * lejos(y), c = (0, 0.5, 1))
plt.plot(y, np.sign(y) * lejos(0.6 * y), c = (1, 0, 0,5))
plt.xlabel("Universo de Discurso", fontsize = 15)
plt.ylabel("Grado de pertenencia", fontsize = 15)
plt.title('XOR', fontsize = 15)
plt.grid(True)

ax = fig.add_subplot(122, projection='3d')
p = ax.plot_surface(x, y, z, cmap = cm.jet, linewidth = 0 antialiased = False)

ax.set_xlim(-10, 10)
ax.set_ylim(-10, 10)
ax.set_zlim(-1, 1)

ax.set_xlabel('Universo 1', fontsize = 15, color = 'red')
ax.set_ylabel('Universo 2', fontsize = 15, color = 'green')
ax.set_zlabel('Grado de pertenencia', fontsize = 15, color = 'blue')
ax.set_title('XOR\n', fontsize = 15)

#ax.view_init(10, 300)
fig.colorbar(p, shrink = 0.5)

plt.show()
```

$$\text{salida}(x_{1,}\ x_2) = \text{XOR}\ \{-\text{signo}(x_1) \cdot \mu(x_1),\ \text{signo}(x_2) \cdot \mu(x_2)\}$$

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#ALBHI. XOR.
#z = np.maximum(np.minimum(np.sign(x) * lejos(x), 1 - -np.sign(y) *
lejos(0.6 * y)), np.minimum(1 - np.sign(x) * lejos(x), -np.sign(y) * lejos(0.6 *
y)))
#JOSE MARIA MOLINA SANCHEZ


import numpy as np #importo Numpy
import matplotlib.pyplot as plt #importo MatPlolib
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm


# funcion borrosa lejos de
def lejos(x):
 return abs(np.tanh(x))


trozos = 1e+3
x = np.linspace(-10, 10, trozos)
y = np.linspace(-10, 10, trozos)
x , y = np.meshgrid(x, y)
z = np.maximum(np.minimum(np.sign(x) * lejos(x), 1 - -np.sign(y) *
lejos(0.6 * y)), np.minimum(1 - np.sign(x) * lejos(x), -np.sign(y) * lejos(0.6 *
y)))


plt.ion()
fig = plt.figure()
ax = fig.add_subplot(121)
plt.plot(y, np.sign(y) * lejos(y), c=(0, 0.5, 1))
plt.plot(y, -np.sign(y) * lejos(0.6 * y), c=(1, 0, 0.5))
plt.xlabel("Universo de Discurso", fontsize = 15)
plt.ylabel("Grado de pertenencia", fontsize = 15)
plt.title('XOR', fontsize = 15)
plt.grid(True)

ax = fig.add_subplot(122, projection='3d')
p = ax.plot_surface(x, y, z, cmap = cm.jet, linewidth = 0 antialiased =
False)

ax.set_xlim(-10, 10)
ax.set_ylim(-10, 10)
ax.set_zlim(-1, 1)

ax.set_xlabel('Universo 1', fontsize = 15, color = 'red')
ax.set_ylabel('Universo 2', fontsize = 15, color = 'green')
ax.set_zlabel('Grado de pertenencia', fontsize = 15, color = 'blue')
ax.set_title('XOR\n', fontsize = 15)

#ax.view_init(10, 300)
fig.colorbar(p, shrink = 0.5)

plt.show()
```

$$\text{salida}(x_1, x_2) = \text{XOR}\left\{\text{signo}(x_1) \cdot \mu(x_1), -\text{signo}(x_2) \cdot \mu(x_2)\right\}$$

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#ALBHI. XOR.
#z = np.maximum(np.minimum(-np.sign(x) * lejos(x), 1 - np.sign(y) *
lejos(0.6 * y)), np.minimum(1 - -np.sign(x) * lejos(x), np.sign(y) * lejos(0.6 *
y)))
#JOSE MARIA MOLINA SANCHEZ


import numpy as np #importo Numpy
import matplotlib.pyplot as plt #importo MatPlolib
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm


# funcion borrosa lejos de
def lejos(x):
 return abs(np.tanh(x))


trozos = 1e+3
x = np.linspace(-10, 10, trozos)
y = np.linspace(-10, 10, trozos)
x , y = np.meshgrid(x, y)
z = np.maximum(np.minimum(-np.sign(x) * lejos(x), 1 - np.sign(y) *
lejos(0.6 * y)), np.minimum(1 - -np.sign(x) * lejos(x), np.sign(y) * lejos(0.6 *
y)))


plt.ion()
fig = plt.figure()
ax = fig.add_subplot(121)
plt.plot(y, -np.sign(y) * lejos(y), c=(0, 0.5, 1))
plt.plot(y, np.sign(y) * lejos(0.6 * y), c=(1, 0, 0.5))
plt.xlabel("Universo de Discurso", fontsize = 15)
plt.ylabel("Grado de pertenencia", fontsize = 15)
plt.title('XOR', fontsize = 15)
plt.grid(True)

ax = fig.add_subplot(122, projection='3d')
p = ax.plot_surface(x, y, z, cmap = cm.jet, linewidth = 0 antialiased =
False)

ax.set_xlim(-10, 10)
ax.set_ylim(-10, 10)
ax.set_zlim(-1, 1)

ax.set_xlabel('Universo 1', fontsize = 15, color = 'red')
ax.set_ylabel('Universo 2', fontsize = 15, color = 'green')
ax.set_zlabel('Grado de pertenencia', fontsize = 15, color = 'blue')
ax.set_title('XOR\n', fontsize = 15)

#ax.view_init(10, 300)
fig.colorbar(p, shrink = 0.5)

plt.show()
```

$$\text{salida}(x_1, x_2) = \text{XOR}\left\{-\text{signo}(x_1) \cdot \mu(x_1),\ -\text{signo}(x_2) \cdot \mu(x_2)\right\}$$

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#ALBHI. XOR.
#z = np.maximum(np.minimum(-np.sign(x) * lejos(x), 1 - -np.sign(y) *
lejos(0.6 * y)), np.minimum(1 - -np.sign(x) * lejos(x), -np.sign(y) * lejos(0.6
* y)))
#JOSE MARIA MOLINA SANCHEZ


import numpy as np #importo Numpy
import matplotlib.pyplot as plt #importo MatPlolib
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm


# funcion borrosa lejos de
def lejos(x):
 return abs(np.tanh(x))


trozos = 1e+3
x = np.linspace(-10, 10, trozos)
y = np.linspace(-10, 10, trozos)
x , y = np.meshgrid(x, y)
z = np.maximum(np.minimum(-np.sign(x) * lejos(x), 1 - -np.sign(y) *
lejos(0.6 * y)), np.minimum(1 - -np.sign(x) * lejos(x), -np.sign(y) * lejos(0.6
* y)))


plt.ion()
fig = plt.figure()
ax = fig.add_subplot(121)
plt.plot(y, -np.sign(y) * lejos(y), c=(0, 0.5, 1))
plt.plot(y, -np.sign(y) * lejos(0.6 * y), c=(1, 0, 0.5))
plt.xlabel("Universo de Discurso", fontsize = 15)
plt.ylabel("Grado de pertenencia", fontsize = 15)
plt.title('XOR', fontsize = 15)
plt.grid(True)

ax = fig.add_subplot(122, projection='3d')
p = ax.plot_surface(x, y, z, cmap = cm.jet, linewidth = 0 antialiased =
False)

ax.set_xlim(-10, 10)
ax.set_ylim(-10, 10)
ax.set_zlim(-1, 1)

ax.set_xlabel('Universo 1', fontsize = 15, color = 'red')
ax.set_ylabel('Universo 2', fontsize = 15, color = 'green')
ax.set_zlabel('Grado de pertenencia', fontsize = 15, color = 'blue')
ax.set_title('XOR\n', fontsize = 15)

#ax.view_init(10, 300)
fig.colorbar(p, shrink = 0.5)

plt.show()
```

Se presentan 14 programas en Python que representan las superficies de control, de **ALBLI**, del apéndice B.

Para el producto de las variables de entrada tenemos:

$$\text{salida}(x_1, x_2) = \text{signo}(x_1) \cdot \mu_L(x_1) \cdot \text{signo}(x_2) \cdot \mu_L(x_2)$$

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#ALBLI. PRODUCTO.
#z = np.sign(x) * lejos(x) * np.sign(y) * lejos(0.6 * y)
#JOSE MARIA MOLINA SANCHEZ


import numpy as np #importo Numpy
import matplotlib.pyplot as plt #importo MatPlolib
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm


# funcion borrosa lejos de
def lejos(x):
 return np.minimum(np.abs(x), 1)


trozos = 1e+3
x = np.linspace(-10, 10, trozos)
y = np.linspace(-10, 10, trozos)
x , y = np.meshgrid(x, y)
z = np.sign(x) * lejos(x) * np.sign(y) * lejos(0.6 * y)


plt.ion()
fig = plt.figure()
ax = fig.add_subplot(121)
plt.plot(y, np.sign(y) * lejos(y), c=(0, 0.5, 1))
plt.plot(y, np.sign(y) * lejos(0.6 * y), c=(1, 0, 0.5))
plt.xlim(-10, 10)
plt.ylim(-1.01, 1.01)
plt.xlabel("Universo de Discurso", fontsize = 15)
plt.ylabel("Grado de pertenencia", fontsize = 15)
plt.title('Lejos-Lejos. Producto', fontsize = 15)
plt.grid(True)

ax = fig.add_subplot(122, projection='3d')
p = ax.plot_surface(x, y, z, cmap = cm.jet, linewidth = 0, antialiased = False)

ax.set_xlim(-10, 10)
ax.set_ylim(-10, 10)
ax.set_zlim(-1, 1)

ax.set_xlabel('Universo 1', fontsize = 15, color = 'red')
ax.set_ylabel('Universo 2', fontsize = 15, color = 'green')
ax.set_zlabel('Grado de pertenencia', fontsize = 15, color = 'blue')
ax.set_title('Lejos-Lejos. Producto\nz = np.sign(x) * lejos(x) * np.sign(y) * lejos(0.6 *
y)\n', fontsize = 15)

#ax.view_init(10, 300)
fig.colorbar(p, shrink = 0.5)

plt.show()
```

$$\text{salida}(x_1, x_2) = \text{signo}(x_1) \cdot \mu_L(x_1) \cdot -\text{signo}(x_2) \cdot \mu_L(x_2)$$

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#ALBLI. PRODUCTO.
#z = np.sign(x) * lejos(x) * -np.sign(y) * lejos(0.6 * y)
#JOSE MARIA MOLINA SANCHEZ


import numpy as np #importo Numpy
import matplotlib.pyplot as plt #importo MatPlolib
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm


# funcion borrosa lejos de
def lejos(x):
 return np.minimum(np.abs(x), 1)


trozos = 1e+3
x = np.linspace(-10, 10, trozos)
y = np.linspace(-10, 10, trozos)
x , y = np.meshgrid(x, y)
z = np.sign(x) * lejos(x) * -np.sign(y) * lejos(0.6 * y)


plt.ion()
fig = plt.figure()
ax = fig.add_subplot(121)
plt.plot(y, np.sign(y) * lejos(y), c=(0, 0.5, 1))
plt.plot(y, -np.sign(y) * lejos(0.6 * y), c=(1, 0, 0.5))
plt.xlim(-10, 10)
plt.ylim(-1.01, 1.01)
plt.xlabel("Universo de Discurso", fontsize = 15)
plt.ylabel("Grado de pertenencia", fontsize = 15)
plt.title('Lejos-Lejos. Producto', fontsize = 15)
plt.grid(True)

ax = fig.add_subplot(122, projection='3d')
p = ax.plot_surface(x, y, z, cmap = cm.jet, linewidth = 0, antialiased =
False)

ax.set_xlim(-10, 10)
ax.set_ylim(-10, 10)
ax.set_zlim(-1, 1)

ax.set_xlabel('Universo 1', fontsize = 15, color = 'red')
ax.set_ylabel('Universo 2', fontsize = 15, color = 'green')
ax.set_zlabel('Grado de pertenencia', fontsize = 15, color = 'blue')
ax.set_title('Lejos-Lejos. Producto\nz = np.sign(x) * lejos(x) * -np.sign(y) *
lejos(0.6 * y)\n', fontsize = 15)

#ax.view_init(10, 300)
fig.colorbar(p, shrink = 0.5)

plt.show()
```

Para el máximo de las variables de entrada tenemos:

$$\text{salida}(x_{1,}\ x_2) = \max(\text{signo}(x_1) \cdot \mu_L(x_1),\ \text{signo}(x_2) \cdot \mu_L(x_2))$$

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#ALBLI. MAXIMO.
#z = np.maximum(np.sign(x) * lejos(x), -np.sign(y) * lejos(0.6 * y))
#JOSE MARIA MOLINA SANCHEZ


import numpy as np #importo Numpy
import matplotlib.pyplot as plt #importo MatPlolib
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm


# funcion borrosa lejos de
def lejos(x):
 return np.minimum(np.abs(x), 1)


trozos = 1e+3
x = np.linspace(-10, 10, trozos)
y = np.linspace(-10, 10, trozos)
x , y = np.meshgrid(x, y)
z = np.maximum(np.sign(x) * lejos(x), -np.sign(y) * lejos(0.6 * y))


plt.ion()
fig = plt.figure()
ax = fig.add_subplot(121)
plt.plot(y, np.sign(y) * lejos(y), c=(0, 0.5, 1))
plt.plot(y, -np.sign(y) * lejos(0.6 * y), c=(1, 0, 0.5))
plt.xlim(-10, 10)
plt.ylim(-1.01, 1.01)
plt.xlabel("Universo de Discurso", fontsize = 15)
plt.ylabel("Grado de pertenencia", fontsize = 15)
plt.title('Lejos-Lejos. Maximo', fontsize = 15)
plt.grid(True)

ax = fig.add_subplot(122, projection='3d')
p = ax.plot_surface(x, y, z, cmap = cm.jet, linewidth = 0, antialiased = False)

ax.set_xlim(-10, 10)
ax.set_ylim(-10, 10)
ax.set_zlim(-1, 1)

ax.set_xlabel('Universo 1', fontsize = 15, color = 'red')
ax.set_ylabel('Universo 2', fontsize = 15, color = 'green')
ax.set_zlabel('Grado de pertenencia', fontsize = 15, color = 'blue')
ax.set_title('Lejos-Lejos.  Maximo\nz  =  np.maximum(np.sign(x)  *  lejos(x),  -
np.sign(y) * lejos(0.6 * y))\n', fontsize = 15)

#ax.view_init(10, 300)
fig.colorbar(p, shrink = 0.5)

plt.show()
```

$$\text{salida}(x_1, x_2) = \max(-\text{signo}(x_1) \cdot \mu_L(x_1), \text{signo}(x_2) \cdot \mu_L(x_2))$$

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#ALBLI. MAXIMO.
#z = np.maximum(-np.sign(x) * lejos(x), np.sign(y) * lejos(0.6 * y))
#JOSE MARIA MOLINA SANCHEZ


import numpy as np #importo Numpy
import matplotlib.pyplot as plt #importo MatPlolib
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm


# funcion borrosa lejos de
def lejos(x):
 return np.minimum(np.abs(x), 1)


trozos = 1e+3
x = np.linspace(-10, 10, trozos)
y = np.linspace(-10, 10, trozos)
x , y = np.meshgrid(x, y)
z = np.maximum(-np.sign(x) * lejos(x), np.sign(y) * lejos(0.6 * y))


plt.ion()
fig = plt.figure()
ax = fig.add_subplot(121)
plt.plot(y, -np.sign(y) * lejos(y), c=(0, 0.5, 1))
plt.plot(y, np.sign(y) * lejos(0.6 * y), c=(1, 0, 0.5))
plt.xlim(-10, 10)
plt.ylim(-1.01, 1.01)
plt.xlabel("Universo de Discurso", fontsize = 15)
plt.ylabel("Grado de pertenencia", fontsize = 15)
plt.title('Lejos-Lejos. Maximo', fontsize = 15)
plt.grid(True)

ax = fig.add_subplot(122, projection='3d')
p = ax.plot_surface(x, y, z, cmap = cm.jet, linewidth = 0, antialiased =
False)

ax.set_xlim(-10, 10)
ax.set_ylim(-10, 10)
ax.set_zlim(-1, 1)

ax.set_xlabel('Universo 1', fontsize = 15, color = 'red')
ax.set_ylabel('Universo 2', fontsize = 15, color = 'green')
ax.set_zlabel('Grado de pertenencia', fontsize = 15, color = 'blue')
ax.set_title('Lejos-Lejos. Maximo\nz = np.maximum(-np.sign(x) * lejos(x),
np.sign(y) * lejos(0.6 * y))\n', fontsize = 15)

#ax.view_init(10, 300)
fig.colorbar(p, shrink = 0.5)

plt.show()
```

$$\text{salida}(x_1, x_2) = \max(-\text{signo}(x_1) \cdot \mu_L(x_1), -\text{signo}(x_2) \cdot \mu_L(x_2))$$

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#ALBLI. MAXIMO.
#z = np.maximum(-np.sign(x) * lejos(x), -np.sign(y) * lejos(0.6 * y))
#JOSE MARIA MOLINA SANCHEZ


import numpy as np #importo Numpy
import matplotlib.pyplot as plt #importo MatPlolib
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm


# funcion borrosa lejos de
def lejos(x):
 return np.minimum(np.abs(x), 1)


trozos = 1e+3
x = np.linspace(-10, 10, trozos)
y = np.linspace(-10, 10, trozos)
x , y = np.meshgrid(x, y)
z = np.maximum(-np.sign(x) * lejos(x), -np.sign(y) * lejos(0.6 * y))


plt.ion()
fig = plt.figure()
ax = fig.add_subplot(121)
plt.plot(y, -np.sign(y) * lejos(y), c=(0, 0.5, 1))
plt.plot(y, -np.sign(y) * lejos(0.6 * y), c=(1, 0, 0.5))
plt.xlim(-10, 10)
plt.ylim(-1.01, 1.01)
plt.xlabel("Universo de Discurso", fontsize = 15)
plt.ylabel("Grado de pertenencia", fontsize = 15)
plt.title('Lejos-Lejos. Maximo', fontsize = 15)
plt.grid(True)

ax = fig.add_subplot(122, projection='3d')
p = ax.plot_surface(x, y, z, cmap = cm.jet, linewidth = 0, antialiased =
False)

ax.set_xlim(-10, 10)
ax.set_ylim(-10, 10)
ax.set_zlim(-1, 1)

ax.set_xlabel('Universo 1', fontsize = 15, color = 'red')
ax.set_ylabel('Universo 2', fontsize = 15, color = 'green')
ax.set_zlabel('Grado de pertenencia', fontsize = 15, color = 'blue')
ax.set_title('Lejos-Lejos. Maximo\nz = np.maximum(-np.sign(x) * lejos(x), -
np.sign(y) * lejos(0.6 * y))\n', fontsize = 15)

#ax.view_init(10, 300)
fig.colorbar(p, shrink = 0.5)

plt.show()
```

Para el mínimo de las variables de entrada tenemos:

$$\text{salida}(x_1, x_2) = \min(\text{signo}(x_1) \cdot \mu_L(x_1), \text{signo}(x_2) \cdot \mu_L(x_2))$$

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#ALBLI. MINIMO.
#z = np.minimum(np.sign(x) * lejos(x), -np.sign(y) * lejos(0.6 * y))
#JOSE MARIA MOLINA SANCHEZ


import numpy as np #importo Numpy
import matplotlib.pyplot as plt #importo MatPlolib
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm


# funcion borrosa lejos de
def lejos(x):
 return np.minimum(np.abs(x), 1)


trozos = 1e+3
x = np.linspace(-10, 10, trozos)
y = np.linspace(-10, 10, trozos)
x , y = np.meshgrid(x, y)
z = np.minimum(np.sign(x) * lejos(x), -np.sign(y) * lejos(0.6 * y))


plt.ion()
fig = plt.figure()
ax = fig.add_subplot(121)
plt.plot(y, np.sign(y) * lejos(y), c=(0, 0.5, 1))
plt.plot(y, -np.sign(y) * lejos(0.6 * y), c=(1, 0, 0.5))
plt.xlim(-10, 10)
plt.ylim(-1.01, 1.01)
plt.xlabel("Universo de Discurso", fontsize = 15)
plt.ylabel("Grado de pertenencia", fontsize = 15)
plt.title('Lejos-Lejos. Minimo', fontsize = 15)
plt.grid(True)

ax = fig.add_subplot(122, projection='3d')
p = ax.plot_surface(x, y, z, cmap = cm.jet, linewidth = 0, antialiased = False)

ax.set_xlim(-10, 10)
ax.set_ylim(-10, 10)
ax.set_zlim(-1, 1)

ax.set_xlabel('Universo 1', fontsize = 15, color = 'red')
ax.set_ylabel('Universo 2', fontsize = 15, color = 'green')
ax.set_zlabel('Grado de pertenencia', fontsize = 15, color = 'blue')
ax.set_title('Lejos-Lejos.  Minimo\nz  =  np.minimum(np.sign(x)  *  lejos(x),  -
np.sign(y) * lejos(0.6 * y))\n', fontsize = 15)

#ax.view_init(10, 300)
fig.colorbar(p, shrink = 0.5)

plt.show()
```

$$\text{salida}(x_{1,}\ x_2) = \min(-\text{signo}(x_1) \cdot \mu_L(x_1),\ \text{signo}(x_2) \cdot \mu_L(x_2))$$

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#ALBLI. MINIMO.
#z = np.minimum(-np.sign(x) * lejos(x), np.sign(y) * lejos(0.6 * y))
#JOSE MARIA MOLINA SANCHEZ


import numpy as np #importo Numpy
import matplotlib.pyplot as plt #importo MatPlolib
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm


# funcion borrosa lejos de
def lejos(x):
 return np.minimum(np.abs(x), 1)


trozos = 1e+3
x = np.linspace(-10, 10, trozos)
y = np.linspace(-10, 10, trozos)
x , y = np.meshgrid(x, y)
z = np.minimum(-np.sign(x) * lejos(x), np.sign(y) * lejos(0.6 * y))


plt.ion()
fig = plt.figure()
ax = fig.add_subplot(121)
plt.plot(y, -np.sign(y) * lejos(y), c=(0, 0.5, 1))
plt.plot(y, np.sign(y) * lejos(0.6 * y), c=(1, 0, 0.5))
plt.xlim(-10, 10)
plt.ylim(-1.01, 1.01)
plt.xlabel("Universo de Discurso", fontsize = 15)
plt.ylabel("Grado de pertenencia", fontsize = 15)
plt.title('Lejos-Lejos. Minimo', fontsize = 15)
plt.grid(True)

ax = fig.add_subplot(122, projection='3d')
p = ax.plot_surface(x, y, z, cmap = cm.jet, linewidth = 0, antialiased =
False)

ax.set_xlim(-10, 10)
ax.set_ylim(-10, 10)
ax.set_zlim(-1, 1)

ax.set_xlabel('Universo 1', fontsize = 15, color = 'red')
ax.set_ylabel('Universo 2', fontsize = 15, color = 'green')
ax.set_zlabel('Grado de pertenencia', fontsize = 15, color = 'blue')
ax.set_title('Lejos-Lejos.  Minimo\nz  =  np.minimum(-np.sign(x)  *  lejos(x),
np.sign(y) * lejos(0.6 * y))\n', fontsize = 15)

#ax.view_init(10, 300)
fig.colorbar(p, shrink = 0.5)

plt.show()
```

$$\text{salida}(x_1, x_2) = \min(-\text{signo}(x_1) \cdot \mu_L(x_1), -\text{signo}(x_2) \cdot \mu_L(x_2))$$

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#ALBLI. MINIMO.
#z = np.minimum(-np.sign(x) * lejos(x), -np.sign(y) * lejos(0.6 * y))
#JOSE MARIA MOLINA SANCHEZ


import numpy as np #importo Numpy
import matplotlib.pyplot as plt #importo MatPlolib
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm


# funcion borrosa lejos de
def lejos(x):
 return np.minimum(np.abs(x), 1)


trozos = 1e+3
x = np.linspace(-10, 10, trozos)
y = np.linspace(-10, 10, trozos)
x , y = np.meshgrid(x, y)
z = np.minimum(-np.sign(x) * lejos(x), -np.sign(y) * lejos(0.6 * y))


plt.ion()
fig = plt.figure()
ax = fig.add_subplot(121)
plt.plot(y, -np.sign(y) * lejos(y), c=(0, 0.5, 1))
plt.plot(y, -np.sign(y) * lejos(0.6 * y), c=(1, 0, 0.5))
plt.xlim(-10, 10)
plt.ylim(-1.01, 1.01)
plt.xlabel("Universo de Discurso", fontsize = 15)
plt.ylabel("Grado de pertenencia", fontsize = 15)
plt.title('Lejos-Lejos. Minimo', fontsize = 15)
plt.grid(True)

ax = fig.add_subplot(122, projection='3d')
p = ax.plot_surface(x, y, z, cmap = cm.jet, linewidth = 0, antialiased =
False)

ax.set_xlim(-10, 10)
ax.set_ylim(-10, 10)
ax.set_zlim(-1, 1)

ax.set_xlabel('Universo 1', fontsize = 15, color = 'red')
ax.set_ylabel('Universo 2', fontsize = 15, color = 'green')
ax.set_zlabel('Grado de pertenencia', fontsize = 15, color = 'blue')
ax.set_title('Lejos-Lejos. Minimo\nz = np.minimum(-np.sign(x) * lejos(x), -
np.sign(y) * lejos(0.6 * y))\n', fontsize = 15)

#ax.view_init(10, 300)
fig.colorbar(p, shrink = 0.5)

plt.show()
```

Para XOR difuso de las variables de entrada tenemos:

$$\text{salida}(x_1, x_2) = \text{XOR}\left\{\text{signo}(x_1) \cdot \mu(x_1), \text{signo}(x_2) \cdot \mu(x_2)\right\}$$

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#ALBLI. XOR.
#z = np.maximum(np.minimum(np.sign(x) * lejos(x), 1 - np.sign(y) * lejos(0.6
* y)), np.minimum(1 - np.sign(x) * lejos(x), np.sign(y) * lejos(0.6 * y)))
#JOSE MARIA MOLINA SANCHEZ


import numpy as np #importo Numpy
import matplotlib.pyplot as plt #importo MatPlolib
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm


# funcion borrosa lejos de
def lejos(x):
 return np.minimum(np.abs(x), 1)


trozos = 1e+3
x = np.linspace(-10, 10, trozos)
y = np.linspace(-10, 10, trozos)
x , y = np.meshgrid(x, y)
z = np.maximum(np.minimum(np.sign(x) * lejos(x), 1 - np.sign(y) * lejos(0.6 *
y)), np.minimum(1 - np.sign(x) * lejos(x), np.sign(y) * lejos(0.6 * y)))


plt.ion()
fig = plt.figure()
ax = fig.add_subplot(121)
plt.plot(y, np.sign(y) * lejos(y), c = (0, 0.5, 1))
plt.plot(y, np.sign(y) * lejos(0.6 * y), c = (1, 0, 0.5))
plt.xlim(-10, 10)
plt.ylim(-1.01, 1.01)
plt.xlabel("Universo de Discurso", fontsize = 15)
plt.ylabel("Grado de pertenencia", fontsize = 15)
plt.title('XOR', fontsize = 15)
plt.grid(True)

ax = fig.add_subplot(122, projection='3d')
p = ax.plot_surface(x, y, z, cmap = cm.jet, linewidth = 0, antialiased = False)

ax.set_xlim(-10, 10)
ax.set_ylim(-10, 10)
ax.set_zlim(-1, 1)

ax.set_xlabel('Universo 1', fontsize = 15, color = 'red')
ax.set_ylabel('Universo 2', fontsize = 15, color = 'green')
ax.set_zlabel('Grado de pertenencia', fontsize = 15, color = 'blue')
ax.set_title('XOR\n', fontsize = 15)

#ax.view_init(10, 300)
fig.colorbar(p, shrink = 0.5)

plt.show()
```

$$\text{salida}(x_1, x_2) = \text{XOR}\ \{-\text{signo}(x_1) \cdot \mu(x_1),\ \text{signo}(x_2) \cdot \mu(x_2)\}$$

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#ALBLI. XOR..
#z = np.maximum(np.minimum(np.sign(x) * lejos(x), 1 - -np.sign(y) * lejos(0.6 * y)), np.minimum(1 - np.sign(x) * lejos(x), -np.sign(y) * lejos(0.6 * y)))
#JOSE MARIA MOLINA SANCHEZ


import numpy as np #importo Numpy
import matplotlib.pyplot as plt #importo MatPlolib
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm


# funcion borrosa lejos de
def lejos(x):
 return np.minimum(np.abs(x), 1)


trozos = 1e+3
x = np.linspace(-10, 10, trozos)
y = np.linspace(-10, 10, trozos)
x , y = np.meshgrid(x, y)
z = np.maximum(np.minimum(np.sign(x) * lejos(x), 1 - -np.sign(y) * lejos(0.6 * y)), np.minimum(1 - np.sign(x) * lejos(x), -np.sign(y) * lejos(0.6 * y)))


plt.ion()
fig = plt.figure()
ax = fig.add_subplot(121)
plt.plot(y, np.sign(y) * lejos(y), c=(0, 0.5, 1))
plt.plot(y, -np.sign(y) * lejos(0.6 * y), c=(1, 0, 0.5))
plt.xlim(-10, 10)
plt.ylim(-1.01, 1.01)
plt.xlabel("Universo de Discurso", fontsize = 15)
plt.ylabel("Grado de pertenencia", fontsize = 15)
plt.title('XOR', fontsize = 15)
plt.grid(True)

ax = fig.add_subplot(122, projection='3d')
p = ax.plot_surface(x, y, z, cmap = cm.jet, linewidth = 0, antialiased = False)

ax.set_xlim(-10, 10)
ax.set_ylim(-10, 10)
ax.set_zlim(-1, 1)

ax.set_xlabel('Universo 1', fontsize = 15, color = 'red')
ax.set_ylabel('Universo 2', fontsize = 15, color = 'green')
ax.set_zlabel('Grado de pertenencia', fontsize = 15, color = 'blue')
ax.set_title('XOR\n', fontsize = 15)

#ax.view_init(10, 300)
fig.colorbar(p, shrink = 0.5)

plt.show()
```

$$\text{salida}(x_1, x_2) = \text{XOR}\left\{\text{signo}(x_1) \cdot \mu(x_1), -\text{signo}(x_2) \cdot \mu(x_2)\right\}$$

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#ALBLI. XOR.
#z = np.maximum(np.minimum(-np.sign(x) * lejos(x), 1 - np.sign(y) * lejos(0.6 * y)), np.minimum(1 - -np.sign(x) * lejos(x), np.sign(y) * lejos(0.6 * y)))
#JOSE MARIA MOLINA SANCHEZ


import numpy as np #importo Numpy
import matplotlib.pyplot as plt #importo MatPlolib
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm


# funcion borrosa lejos de
def lejos(x):
 return np.minimum(np.abs(x), 1)


trozos = 1e+3
x = np.linspace(-10, 10, trozos)
y = np.linspace(-10, 10, trozos)
x , y = np.meshgrid(x, y)
z = np.maximum(np.minimum(-np.sign(x) * lejos(x), 1 - np.sign(y) * lejos(0.6 * y)), np.minimum(1 - -np.sign(x) * lejos(x), np.sign(y) * lejos(0.6 * y)))


plt.ion()
fig = plt.figure()
ax = fig.add_subplot(121)
plt.plot(y, -np.sign(y) * lejos(y), c=(0, 0.5, 1))
plt.plot(y, np.sign(y) * lejos(0.6 * y), c=(1, 0, 0.5))
plt.xlim(-10, 10)
plt.ylim(-1.01, 1.01)
plt.xlabel("Universo de Discurso", fontsize = 15)
plt.ylabel("Grado de pertenencia", fontsize = 15)
plt.title('XOR', fontsize = 15)
plt.grid(True)

ax = fig.add_subplot(122, projection='3d')
p = ax.plot_surface(x, y, z, cmap = cm.jet, linewidth = 0, antialiased = False)

ax.set_xlim(-10, 10)
ax.set_ylim(-10, 10)
ax.set_zlim(-1, 1)

ax.set_xlabel('Universo 1', fontsize = 15, color = 'red')
ax.set_ylabel('Universo 2', fontsize = 15, color = 'green')
ax.set_zlabel('Grado de pertenencia', fontsize = 15, color = 'blue')
ax.set_title('XOR\n', fontsize = 15)

#ax.view_init(10, 300)
fig.colorbar(p, shrink = 0.5)

plt.show()
```

$$\text{salida}(x_1, x_2) = \text{XOR}\left\{-\text{signo}(x_1) \cdot \mu(x_1),\ -\text{signo}(x_2) \cdot \mu(x_2)\right\}$$

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#ALBLI. XOR.
#z = np.maximum(np.minimum(-np.sign(x) * lejos(x), 1 - -np.sign(y) * lejos(0.6 * y)), np.minimum(1 - -np.sign(x) * lejos(x), -np.sign(y) * lejos(0.6 * y)))
#JOSE MARIA MOLINA SANCHEZ


import numpy as np #importo Numpy
import matplotlib.pyplot as plt #importo MatPlolib
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm


# funcion borrosa lejos de
def lejos(x):
 return np.minimum(np.abs(x), 1)


trozos = 1e+3
x = np.linspace(-10, 10, trozos)
y = np.linspace(-10, 10, trozos)
x , y = np.meshgrid(x, y)
z = np.maximum(np.minimum(-np.sign(x) * lejos(x), 1 - -np.sign(y) * lejos(0.6 * y)), np.minimum(1 - -np.sign(x) * lejos(x), -np.sign(y) * lejos(0.6 * y)))


plt.ion()
fig = plt.figure()
ax = fig.add_subplot(121)
plt.plot(y, -np.sign(y) * lejos(y), c=(0, 0.5, 1))
plt.plot(y, -np.sign(y) * lejos(0.6 * y), c=(1, 0, 0.5))
plt.xlim(-10, 10)
plt.ylim(-1.01, 1.01)
plt.xlabel("Universo de Discurso", fontsize = 15)
plt.ylabel("Grado de pertenencia", fontsize = 15)
plt.title('XOR', fontsize = 15)
plt.grid(True)

ax = fig.add_subplot(122, projection='3d')
p = ax.plot_surface(x, y, z, cmap = cm.jet, linewidth = 0, antialiased = False)

ax.set_xlim(-10, 10)
ax.set_ylim(-10, 10)
ax.set_zlim(-1, 1)

ax.set_xlabel('Universo 1', fontsize = 15, color = 'red')
ax.set_ylabel('Universo 2', fontsize = 15, color = 'green')
ax.set_zlabel('Grado de pertenencia', fontsize = 15, color = 'blue')
ax.set_title('XOR\n', fontsize = 15)

#ax.view_init(10, 300)
fig.colorbar(p, shrink = 0.5)

plt.show()
```

APÉNDICE D.

Si, en **ALBHI**, para un determinado x_0 y $\mu_L(x_0) = 0{,}5$ el numerador para calcular el punto máximo es 0,549 ¿qué valor de x es $\mu_L(x) = 0{,}999$ o $\mu_C(x) = 0{,}001$?.

Sea x_1 y $\mu_L(x_1) = 0{,}999$ por la definición de punto máximo tenemos:

$$0{,}999 = \text{lejos}(\text{pmL} \cdot x_1)$$

y con x_0:

$$\text{pmL} = \frac{0{,}549}{x_0}$$

sustituyendo:

$$0{,}999 = \text{lejos}(\frac{0{,}549}{x_0} \cdot x_1)$$

aplicamos la función lejos:

$$3{,}800 = \frac{0{,}549}{x_0} \cdot x_1$$

$$x_1 = \frac{3{,}800}{0{,}549} \cdot x_0$$

$$\mathbf{x_1 = 6{,}918 \cdot x_0}$$

$$\mathbf{x_1 \approx 7 \cdot x_0}$$

A un determinado x_0 es $\mu(x_0) = 0{,}5$; en x_1 será $\mu(x_1) = 1$ o $\mu(7 \cdot x_0) = 1$.

x_1 se puede ver cómo el valor a partir del cual el sistema empieza a reaccionar disminuyendo su salida.

Para otras funciones de pertenencia los cálculos de x_0 y x_1 varían.

APÉNDICE E.

Para otras funciones de pertenencia los puntos máximos son:

$$\mu_L(x) = 1 - \frac{1}{1 + (pmL \cdot x)^2} \qquad pmL = \frac{\sqrt{\left(\frac{1}{1 - \mu_L(x)} - 1\right)}}{x}$$

$$\mu_L(x) = 1 - \frac{1}{\cosh(pmL \cdot x)} \qquad pmL = \frac{\operatorname{acosh}\left(\frac{1}{1 - \mu_L(x)}\right)}{x}$$

$$\mu_L(x) = 1 - e^{-(pmL \cdot x)^2} \qquad pmL = \frac{\sqrt{(-\ln(1 - \mu_L(x)))}}{x}$$

$$\mu_L(x) = \left|\frac{2}{1 + e^{-(pmL \cdot x)}} - 1\right| \qquad pmL = \frac{-\ln\left(\frac{2}{1 + \mu_L(x)} - 1\right)}{x}$$

$$\mu_L(x) = \tanh(pmL \cdot x)^2 \qquad pmL = \frac{\operatorname{atanh}(\sqrt{(\mu_L(x))})}{x}$$

$$\mu_L(x) = 1 - e^{-|pmL \cdot x|} \qquad pmL = \frac{-\ln(1 - \mu_L(x))}{x}$$

Para calcular el punto máximo hay que sustituir 1 - $\mu_L(x)$ por $\mu_C(x)$, y $\mu_L(x)$ por 1 – $\mu_C(x)$.

Los numeradores de los puntos máximos para $\mu_L(x) = 0{,}999$ de las funciones son:

$$\mu_L(x) = 1 - \frac{1}{1 + (pmL \cdot x)^2} \qquad pmL = \frac{31{,}607}{x}$$

$$\mu_L(x) = 1 - \frac{1}{\cosh(pmL \cdot x)} \qquad pmL = \frac{7{,}601}{x}$$

$$\mu_L(x) = 1 - e^{-(pmL \cdot x)^2} \qquad pmL = \frac{2{,}628}{x}$$

$$\mu_L(x) = \left| \frac{2}{1 + e^{-(pmL \cdot x)}} - 1 \right| \qquad pmL = \frac{7,600}{x}$$

$$\mu_L(x) = \tanh(pmL \cdot x)^2 \qquad pmL = \frac{4,147}{x}$$

$$\mu_L(x) = 1 - e^{-|pmL \cdot x|} \qquad pmL = \frac{6,908}{x}$$

Los numeradores de los puntos máximos para $\mu_C(x) = 0,001$ de las funciones son los mismos.

APÉNDICE F.

La función $\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$ es equivalente a $\tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$ si multiplicamos el numerador y el denominador por e^{-x}, que se puede reducir a $\tanh(x) = \frac{2}{1 + e^{-2x}} - 1$.

Demostración:

$$\frac{1 - e^{-2x}}{1 + e^{-2x}} = \frac{(2 - 1) - e^{-2x}}{1 + e^{-2x}} = \frac{2 - (1 + e^{-2x})}{1 + e^{-2x}} = \frac{2}{1 + e^{-2x}} - \frac{1 + e^{-2x}}{1 + e^{-2x}} = \frac{2}{1 + e^{-2x}} - 1$$

Por tanto la función de pertenencia $\mu_L(x) = \frac{2}{1 + e^{-x}} - 1$ es $\tanh\left(\frac{x}{2}\right)$.

El inverso se calcula:

$$\operatorname{atanh}(x) = -\frac{1}{2} \ln\left(\frac{1 - \mu_L(x)}{1 + \mu_L(x)}\right);\ |\mu_L(x)| < 1$$

$$\operatorname{atanh}(x) = -\frac{1}{2} \ln\left(\frac{2}{1 + \mu_L(x)} - 1\right);\ |\mu_L(x)| < 1$$

Hay otra función similar a $\mu_L(x)$ procedente de la IA y es $\sigma(x) = \frac{1}{1 + e^{-x}}$ de la que se puede obtener la tanh(x) de la siguiente manera: $\tanh(x) = 2 \cdot \sigma(2 \cdot x) - 1$. También se puede obtener $\mu_L(x)$ a partir de σ(x) así: $\mu_L(x) = 2 \cdot \sigma(x) - 1$.

El coste computacional de esta forma de calcular tanh(x) y su inversa es menor que la forma ordinaria, puesto que se pasa de calcular cuatro funciones a una solamente con tres constantes.

A continuación el programa que demuestra que es más del doble rápida esta forma de calcular la tanh.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-


"""
Created on Tue Jun 25 17:48:18 2019
@author:JOSE MARIA MOLINA SANCHEZ
Cálculo tanh
"""


import numpy as np #importo Numpy
from time import time


#defino los parmetros de entrada
num = 1000000
x = np.linspace(-100, 100, num)


#bloque de calculo de la tanh de la forma tradicional
inicio_de_tiempo = time()

def tanh1(x):
    return (np.exp(x) - np.exp(-x)) / (np.exp(x) + np.exp(-x))

y1 = tanh1(x)
tiempo_final = time()
tiempo_transcurrido1 = tiempo_final - inicio_de_tiempo
print ("\nTomó {:n} milisegundos el cálculo de la tanh de la forma tradicional, con {:n} de entradas.".format(np.round(1000 * tiempo_transcurrido1, 2), num))


#bloque de calculo de la tanh de la forma reducida
inicio_de_tiempo = time()

def tanh2(x):
    return (2 / (1 + np.exp(-2 * x))) - 1

y2 = tanh2(x)
tiempo_final = time()
tiempo_transcurrido2 = tiempo_final - inicio_de_tiempo
print ("\nTomó {:n} milisegundos el cálculo de la tanh de la forma reducida, con {:n} de entradas.".format(np.round(1000 * tiempo_transcurrido2, 2), num))

ratio = tiempo_transcurrido1 / tiempo_transcurrido2

print ("\nLa forma reducida es {:n} veces más rápida que la forma tradicional.".format(np.round(ratio, 2)))
```

APÉNDICE G.

Generalizamos la función del apéndice F.

Si $n^x = e^x$ y el numerador es “-” y el denominador es “+” entonces $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \tanh(x)$, por tanto:

$$\tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$

y si el numerador es “+” y el denominador es “-” entonces:

$$\operatorname{cotanh}(x) = \frac{2}{1 - e^{-2x}} - 1$$

El coste computacional de esta forma de calcular f(x) es menor que la forma ordinaria, puesto que se pasa de calcular cuatro funciones a una solamente con tres constantes.

APÉNDICE H.

En este apéndice se encuentran los programas en Python de implicación, equivalencia y XOR.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#Fuzzy Implicación
#JOSE MARIA MOLINA SANCHEZ


import numpy as np #importo Numpy
import matplotlib.pyplot as plt #importo MatPlolib
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm


trozos = 1000
x = np.linspace(0, 1, trozos)
y = np.linspace(0, 1, trozos)
x , y = np.meshgrid(x, y)
z = np.maximum(1 - x, y)


plt.ion()
fig = plt.figure('Fuzzy Implicación')
ax = Axes3D(fig)
p = ax.plot_surface(x, y, z, cmap=cm.jet, linewidth=0, antialiased=False)
ax.view_init(30, 300)
cb = fig.colorbar(p, shrink=0.5)
ax.set_xlabel('X', fontsize=14, color='red')
ax.set_ylabel('Y', fontsize=14, color='green')
ax.set_zlabel('Z', fontsize=14, color='blue')
ax.set_title('Fuzzy Implicación\n', fontsize=25)
plt.show()
```

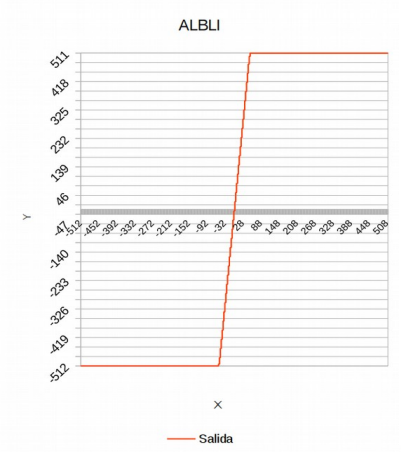
```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#Fuzzy Equivalencia
#JOSE MARIA MOLINA SANCHEZ


import numpy as np #importo Numpy
import matplotlib.pyplot as plt #importo MatPlolib
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm


trozos = 1000
x = np.linspace(0, 1, trozos)
y = np.linspace(0, 1, trozos)
x , y = np.meshgrid(x, y)
z = np.maximum(np.minimum(x, y), np.minimum(1 - x, 1 - y))


plt.ion()
fig = plt.figure('Fuzzy Equivalencia')
ax = Axes3D(fig)
p = ax.plot_surface(x, y, z, cmap=cm.jet, linewidth=0, antialiased=False)
ax.view_init(30, 120)
cb = fig.colorbar(p, shrink=0.5)
ax.set_xlabel('X', fontsize=14, color='red')
ax.set_ylabel('Y', fontsize=14, color='green')
ax.set_zlabel('Z', fontsize=14, color='blue')
ax.set_title('Fuzzy Equivalencia\n', fontsize=25)
plt.show()
```

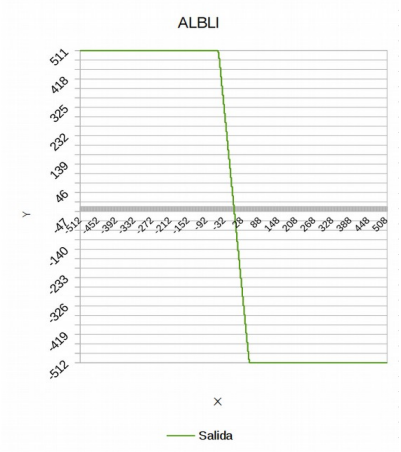
```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#Fuzzy XOR
#JOSE MARIA MOLINA SANCHEZ


import numpy as np #importo Numpy
import matplotlib.pyplot as plt #importo MatPlolib
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm


trozos = 1000
x = np.linspace(0, 1, trozos)
y = np.linspace(0, 1, trozos)
x , y = np.meshgrid(x, y)
z = np.maximum(np.minimum(x, 1 - y), np.minimum(1 - x, y))


plt.ion()
fig = plt.figure('Fuzzy XOR')
ax = Axes3D(fig)
p = ax.plot_surface(x, y, z, cmap=cm.jet, linewidth=0, antialiased=False)
ax.view_init(30, 300)
cb = fig.colorbar(p, shrink=0.5)
ax.set_xlabel('X', fontsize=14, color='red')
ax.set_ylabel('Y', fontsize=14, color='green')
ax.set_zlabel('Z', fontsize=14, color='blue')
ax.set_title('Fuzzy XOR\n', fontsize=25)
plt.show()
```

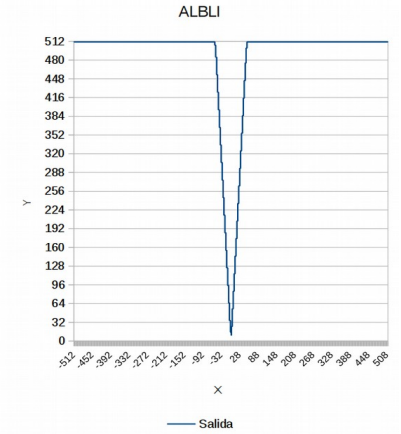
APÉNDICE I.

Con la forma de hacer lógica difusa en **ALBLI** podemos aprovechar el potencial que ofrece la aritmética de saturación; es una versión de aritmética en la que todas las operaciones están limitadas a un rango fijo entre un valor mínimo y uno máximo; tal y como se ha definido las funciones lejos(x) y cerca (x); y tal cómo es la función límite.

En un sistema binario de 10 bits con complemento a dos para definir los números negativos y un pmL = ±1 la función lejos(x) quedaría como muestran las imágenes.







El eje X representa el universo de discurso, y el eje Y el grado de pertenencia.

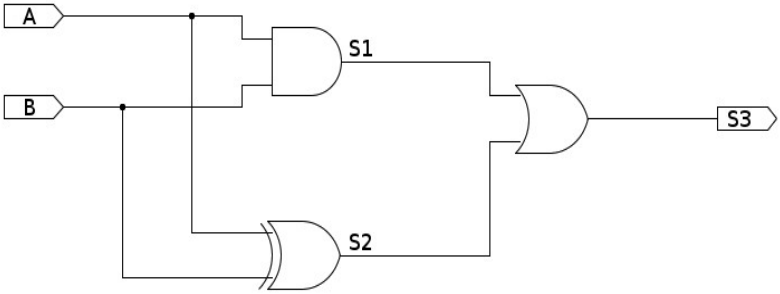
Podemos utilizar así el desbordamiento como un factor positivo a la hora de hacer un cómputo, pues nos está indicando el máximo y el mínimo valor que puede tener el sistema lógico difuso, y sacar provecho para simplificar y acelerar los cálculos. Habría que dividir el resultado entre el valor máximo que puede tomar para que varíe entre -1 y 1, y multiplicarlo por el valor máximo de salida para controlar la máquina.

En el ejemplo la función varía entre 0 y 512, lo podemos tomar cómo $\mu_L(x) = \frac{(\pm pmL \cdot x)}{512}$.

Muchos microcontroladores utilizan 8 bits que son suficientes para realizar un control difuso **ALBLI**, en caso de utilizar microprocesadores de 64 bits se pueden segmentar en 8 partes de 8 bits u ocho entradas/salidas por palabra. El ejemplo del siguiente apéndice cabría en 5 de estos segmentos, es decir ocuparía 40 bits, ello sin contar la definición de cada puerta lógica difusa.

APÉNDICE J.

Vamos realizar un circuito combinacional para dos entradas y una salida.



Las puertas lógicas aquí representadas son difusas. Aunque he utilizado la simbología digital a falta de otra.

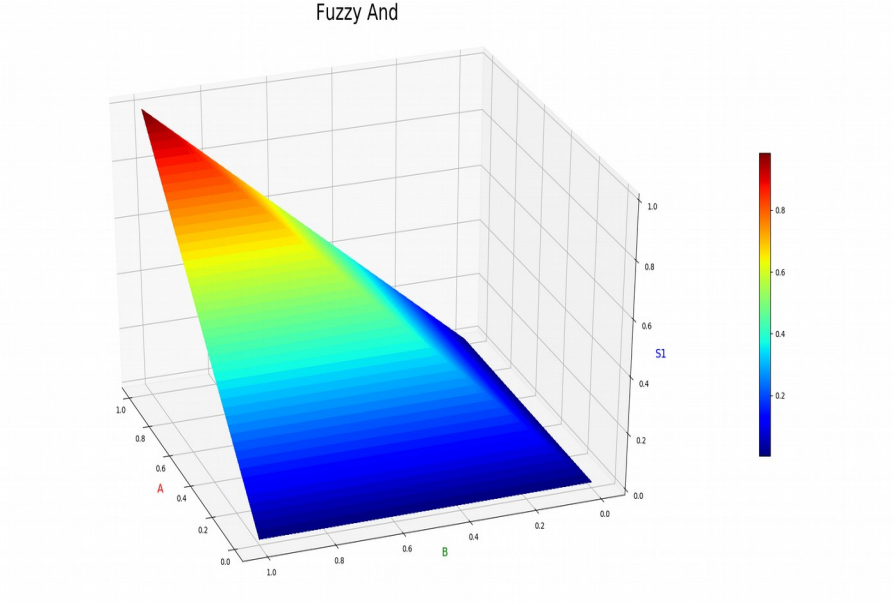
A, B $\in$ [0, 1].

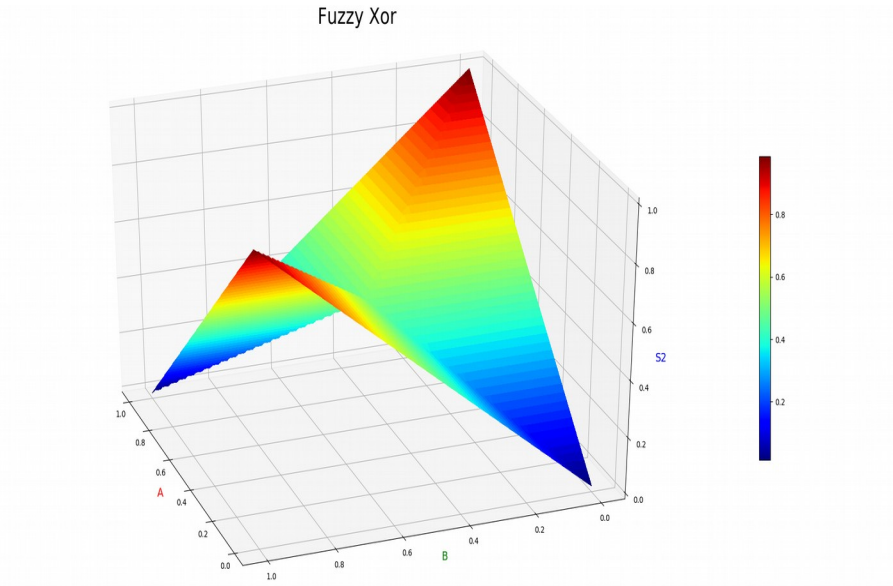
AND es min(A, B) = S1.

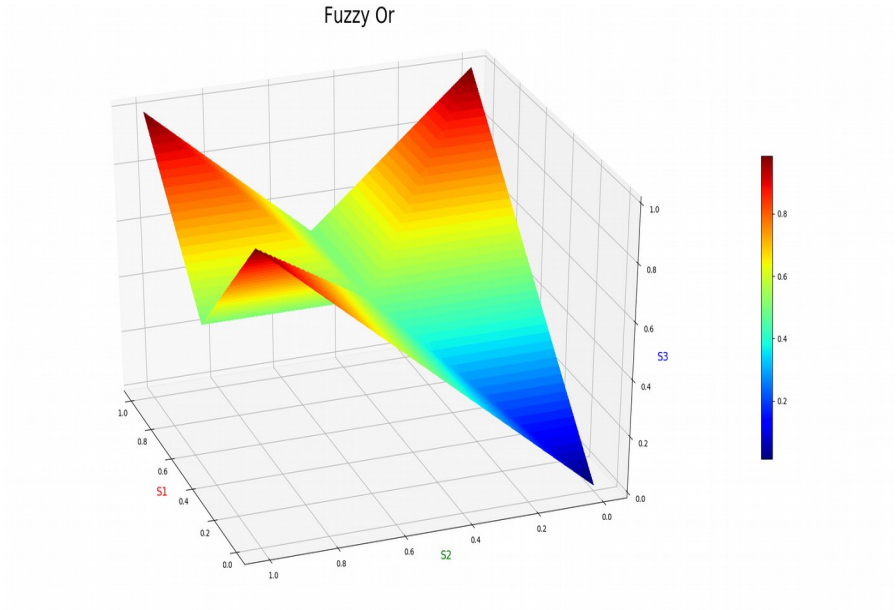
XOR es max(min(A, 1-B), min(1-A, B)) = S2.

OR es max(S1, S2) = S3.

Las gráficas de cada puerta son las siguientes.







La gráfica Fuzzy OR es la salida del sistema combinacional, y representa todas las soluciones.

He omitido las variables lingüísticas lejos(x) y cerca(x) en el circuito combinacional y dejo al lector que sea él mismo quien lo haga, pudiendo publicarlo libremente en internet bajo la misma licencia que este libro.

Huelga decir que podemos introducir además de las operaciones lógicas otros operandos diferentes tal y cómo se ha hecho en las reglas de inferencia, ya sea suma, producto, división, raíces, potencias, etc. Pero se ha de tener en cuenta que la salida ha de estar en el dominio comprendido entre 0 y 1, caso de que no sea así habrá que dividir entre el valor máximo o truncarlo a 1 con la función límite.

APÉNDICE K.

A continuación el programa en Python que representa las superficies de control del circuito combinacional.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#Combi.
#JOSE MARIA MOLINA SANCHEZ


import numpy as np #importo Numpy
import matplotlib.pyplot as plt #importo MatPlolib
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm


trozos = 1000
x = np.linspace(0, 1, trozos)
y = np.linspace(0, 1, trozos)
x , y = np.meshgrid(x, y)
s1 = np.minimum(x, y)
s2 = np.maximum(np.minimum(x, 1 - y), np.minimum(1 - x, y))
s3 = np.maximum(s1, s2)

plt.ion()

fig = plt.figure('S1')
ax = Axes3D(fig)
p = ax.plot_surface(x, y, s1, cmap=cm.jet, linewidth=0, antialiased=False)
ax.view_init(30, 160)
cb = fig.colorbar(p, shrink=0.5)
ax.set_xlabel('A', fontsize=14, color='red')
ax.set_ylabel('B', fontsize=14, color='green')
ax.set_zlabel('S1', fontsize=14, color='blue')
ax.set_title('Fuzzy And\n', fontsize=25)

fig = plt.figure('S2')
ax = Axes3D(fig)
p = ax.plot_surface(x, y, s2, cmap=cm.jet, linewidth=0, antialiased=False)
ax.view_init(30, 160)
cb = fig.colorbar(p, shrink=0.5)
ax.set_xlabel('A', fontsize=14, color='red')
ax.set_ylabel('B', fontsize=14, color='green')
ax.set_zlabel('S2', fontsize=14, color='blue')
ax.set_title('Fuzzy XOR\n', fontsize=25)

fig = plt.figure('S3')
ax = Axes3D(fig)
p = ax.plot_surface(x, y, s3, cmap=cm.jet, linewidth=0, antialiased=False)
ax.view_init(30, 160)
cb = fig.colorbar(p, shrink=0.5)
ax.set_xlabel('S1', fontsize=14, color='red')
ax.set_ylabel('S2', fontsize=14, color='green')
ax.set_zlabel('S3', fontsize=14, color='blue')
ax.set_title('Fuzzy Or\n', fontsize=25)

plt.show()
```

APÉNDICE L.

Podemos definir muchas operaciones para las reglas de inferencia, pero todas tienen la misma forma generalizada.

Sea OP dicha operación, para dos entradas las reglas de inferencia quedan así definidas:

1. $\mu(x_{1,}\ x_2) = \text{OP}\ (\mu(x_1),\ \mu(x_2))$
2. $\mu(x_{1,}\ x_2) = \text{OP}\ (\text{signo}(x_1) \cdot \mu(x_1),\ \mu(x_2))$
3. $\mu(x_{1,}\ x_2) = \text{OP}\ (-\text{signo}(x_1) \cdot \mu(x_1),\ \mu(x_2))$
4. $\mu(x_{1,}\ x_2) = \text{OP}\ (\mu(x_1),\ \text{signo}(x_2) \cdot \mu(x_2))$
5. $\mu(x_{1,}\ x_2) = \text{OP}\ (\mu(x_1),\ -\text{signo}(x_2) \cdot \mu(x_2))$
6. $\mu(x_{1,}\ x_2) = \text{OP}\ (\text{signo}(x_1) \cdot \mu(x_1),\ \text{signo}(x_2) \cdot \mu(x_2))$
7. $\mu(x_{1,}\ x_2) = \text{OP}\ (\text{signo}(x_1) \cdot \mu(x_1),\ -\text{signo}(x_2) \cdot \mu(x_2))$
8. $\mu(x_{1,}\ x_2) = \text{OP}\ (-\text{signo}(x_1) \cdot \mu(x_1),\ \text{signo}(x_2) \cdot \mu(x_2))$
9. $\mu(x_{1,}\ x_2) = \text{OP}\ (-\text{signo}(x_1) \cdot \mu(x_1),\ -\text{signo}(x_2) \cdot \mu(x_2))$

Para saber cuantas reglas de inferencia puede tener una combinación dada de diferentes entradas para una sola salida tenemos que ver cuantas formas tiene cada una de ellas. Puede ir sola, con +signo y con -signo, es decir 3. Por tanto tenemos que recurrir al número de variaciones con repetición de 3 elementos tomados de n en n, donde n son las entradas.

$$\text{reglas de inferencia} = 3^n$$

Como hemos visto para dos entradas tenemos 9 reglas de inferencia, para 3 entradas tendríamos 27, y así sucesivamente.

Por lo visto en los ejemplos las reglas más utilizadas serán la 6, la 7, la 8 y la 9; 4 reglas en total. Así que para saber cuantas reglas de inferencia útiles serán las más usadas vemos que hay dos formas, con +signo y con -signo. Por tanto tenemos que recurrir al número de variaciones con repetición de 2 elementos tomados de n en n.

$$\text{reglas de inferencia utiles} = 2^n$$

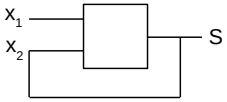
En realidad se trata de una misma matriz de inferencia que para el caso de dos dimensiones gira 90 grados cada vez con los ejes de coordenadas fijos, ver el apéndice B.

APÉNDICE M.

Puesto que se puede definir cualquier operación podemos hacer el siguiente generador de patrones caótico:

$$\mu(x_{1,}\ x_2) = 4 \cdot \mu(x_1) \cdot \mu(x_2) \cdot (1 - \mu(x_2))$$

En el que la salida se realimentaría a x_2, y $x_1 \in [0,1]$ constituye la única entrada.



Además de usarlo cómo patrón caótico independiente, se puede asociar a otros sistemas ponderando la salida, que al estar unidos generará una cierta inestabilidad tal y cómo ocurre en los sistemas biológicos. Es más, podemos cambiar el 4 por un peso que esté entre cero y cuatro. O juntar varios con las salidas ponderadas y unido a un sistema **ALBLI** poder simular patrones existentes cómo el mercado de valores, clima, etc. Pudiendo diseñar una verdadera inteligencia con creatividad.

APÉNDICE N.

Muchos sistemas de control utilizan para controlar la potencia de los motores eléctricos DC la modulación por ancho de pulso; PWM; respecto de una señal de entrada, la formulación matemática es la siguiente:

$$PWM\,(e) = \lfloor f \cdot t \rfloor - \lfloor f \cdot t - e \rfloor$$

donde f es la frecuencia, t es el tiempo, e es la entrada y $\lfloor x \rfloor$ es la función floor(x), $\forall\ f,\ t,\ e \in \mathbb{R};\ \ e \in [0,\ 1]$.

Podemos utilizar la función de pertenencia $\mu(x_1,\ x_2,\ \ldots,\ x_n)$ cómo entrada, ya que varía entre 0 y 1 quedando la función PWM de la forma siguiente:

$$PWM\,(\mu) = \lfloor f \cdot t \rfloor - \lfloor f \cdot t - \mu(x_1,\ x_2,\ ...,\ x_n) \rfloor$$

En el caso de que la función de pertenencia varíe entre -1 y 1 utilizaremos la siguiente función:

$$PWM\,(\mu) = \lfloor f \cdot t \rfloor - \lfloor f \cdot t - |\mu(x_1,\ x_2,\ ...,\ x_n)| \rfloor$$

Y en el sentido giro del motor habrá que tener en cuenta el signo de $\mu(x_1,\ x_2,\ ...,\ x_n)$.

APÉNDICE O.

Los diferentes tipos de transistores que hay en el mercado tienen curvas de respuesta no lineales, éstas o parte de ellas se pueden utilizar cómo funciones de pertenencia ya sean en solitario o definidas cómo funciones a trozos junto con otros componentes.

REFERENCIAS.

http://www.ilustrados.com/publicaciones/EpZEEVZVZVRyHMwXhd.php

http://www.imse.cnm.es/Xfuzzy/Fleb/Fleb.htm

http://www.lcc.uma.es/~ppgg/FSS/

http://www.monografias.com/trabajos6/lalo/lalo.shtml

http://es.wikipedia.org/wiki/L%C3%B3gica_difusa

https://www.youtube.com/playlist?list=PL1b0cquYkGiDyKEP5YF3OA2Zb9Y1biLrF

https://www.youtube.com/playlist?list=PLIyIZGa1sAZoWAeT_tL7zCv3wi1ISrBa0

https://www.youtube.com/playlist?list=PLIyIZGa1sAZqL3nPM_YERDxwcor0bYdoL

https://www.youtube.com/playlist?list=PLYWD-VqrD5BCJDW70Dc4XMEQKg6aTmIak

https://www.youtube.com/watch?v=z6UV1IrNvJU

El objetivo de este libro es demostrar que el actual sistema de control basado en la lógica difusa se puede mejorar, creando un sistema deductivo y control más eficiente energética y computacionalmente.

La idea subyacente es sencilla, y su aplicación lo es todavía más.

La simplificación que tenía en mente era que mediante una función sencilla poder deducir su grado de pertenencia, a la vez que esa función fuese un concepto lingüístico.

Esta lógica difusa en vez de ser algebraica es analítica, y es en el análisis matemático donde tiene su potencial desarrollo.