UNIVERSIDAD DE GRANADA

NUEVA METODOLOGÍA PARA EL DISEÑO AUTOMÁTICO DE SISTEMAS DIFUSOS

TESIS DOCTORAL

Héctor Pomares Cintas

1999

Departamento de Arquitectura y Tecnología de Computadores

D. Alberto Prieto Espinosa, Catedrático de Universidad, y **D. Ignacio Rojas Ruiz**, Profesor Titular de Universidad del Departamento de Arquitectura y Tecnología de Computadores

CERTIFICAN

Que la memoria titulada: "*Nueva Metodología para el Diseño Automático de Sistemas Difusos*" ha sido realizada por **D. Héctor Pomares Cintas** bajo nuestra dirección en el Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada para optar al grado de Doctor en Ingeniería Electrónica.

Granada, a 26 de Octubre de 1999

Fdo. Alberto Prieto Espinosa
Director de la Tesis

Fdo. Ignacio Rojas Ruiz
Director de la Tesis

UNIVERSIDAD DE GRANADA

NUEVA METODOLOGÍA PARA EL DISEÑO AUTOMÁTICO DE SISTEMAS DIFUSOS

Héctor Pomares Cintas

TESIS DOCTORAL

DIRECTORES:

Alberto Prieto Espinosa
Ignacio Rojas Ruiz

1999

Departamento de Arquitectura y Tecnología de Computadores

ÍNDICE

CAPÍTULO 1: INTRODUCCIÓN

CAPÍTULO 2: FUNDAMENTOS

CAPÍTULO 3: DISEÑO AUTOMÁTICO DE SISTEMAS DIFUSOS A PARTIR DE DATOS DE E/S

CAPÍTULO 4: EJEMPLOS, EVALUACIÓN Y COMPARACIONES DEL ALGORITMO DE APROXIMACIÓN FUNCIONAL

CAPÍTULO 5: DISEÑO AUTOMÁTICO DE CONTROLADORES DIFUSOS EN TIEMPO REAL

CAPÍTULO 1

INTRODUCCIÓN

La creciente complejidad de los sistemas dinámicos junto con las igualmente crecientes prestaciones que se exigen a los sistemas de control obliga a la utilización de controladores cada vez más modernos y sofisticados. Uno de los principales objetivos es obtener controladores “inteligentes” capaces de adaptarse y seguir funcionando ante cambios imprevistos en la planta a controlar y posiblemente capaces de “auto-construirse” automáticamente a partir del análisis en tiempo real de la dinámica del sistema. Esencialmente, se trata de diseñar controladores capaces de exprimir todo el conocimiento existente sobre un sistema dinámico para conseguir un control óptimo según las especificaciones exigidas.

En este capítulo se presentará una breve introducción al tema abordado en esta tesis incluyendo los objetivos que se propone resolver. En la sección final, se presenta una descripción resumida de los temas tratados en cada capítulo de la presente memoria.

1.1 Revisión histórica: Control Automático

El control automático ha jugado un papel vital en el avance de la ciencia y de la ingeniería. El nombre “control automático” se usa para describir una gran variedad de sistemas cuyo propósito es el de minimizar la necesidad de intervención humana en la operación de un sistema particular de control y maximizar su eficiencia. El sistema o planta a controlar puede ser de cualquier tipo: sistemas de piloto automático de aeronaves, sistemas de guía de proyectiles, robots, procesos químicos, control de temperatura, presión, humedad, etc. A pesar de la gran variedad en los procesos a controlar, sus características básicas, desde el punto de vista de la teoría de control, son generalmente similares, de modo que se suelen abordar todos ellos bajo la misma perspectiva matemática.

Uno de los primeros trabajos más significativos en control automático fue el regulador centrífugo de James Watt para el control de la velocidad de una máquina de vapor, en el siglo XVIII. Aunque este tipo de reguladores empezaron a usarse con cierto éxito en algunos campos de control, en ocasiones surgían fallos e incluso el sistema se volvía inestable. Las condiciones bajo las cuales esto sucedía apenas si se lograba llegar a entender hasta que, en 1868, James Maxwel [MAX-68] fue capaz de encontrar un método para analizar sistemas a través de la resolución de las ecuaciones diferenciales lineales que los caracterizaban. Otros avances relevantes en las primeras etapas del desarrollo de la teoría clásica de control se deben a Minorsky, Hazen y Nyquist, entre muchos otros. En 1922 Minorsky trabajó en controladores automáticos de dirección en barcos y mostró cómo se podía determinar la estabilidad a partir de las ecuaciones diferenciales que describen el sistema. En 1932, Nyquist desarrolló un método relativamente simple para determinar la estabilidad de los sistemas de lazo cerrado basándose en la respuesta en lazo abierto con excitación sinusoidal en régimen estacionario. En 1934, Hazen, quien introdujo el término servomecanismos para los sistemas de control de posición, desarrolló el diseño de servomecanismos “repetidores” capaces de seguir con exactitud una entrada cambiante.

Durante la década de los años cuarenta, los métodos de respuesta en frecuencia posibilitaron a los ingenieros el diseño de sistemas lineales de control de lazo cerrado que satisfacían los comportamientos requeridos. De finales de los cuarenta a principios de los cincuenta, Evans desarrolló por completo el método del lugar de las raíces.

Los métodos de respuesta en frecuencia y del lugar de las raíces, que son el corazón de la teoría de control clásica, conducen a sistemas que son estables y que satisfacen un conjunto de especificaciones de funcionamiento más o menos arbitrario. Tales sistemas son, en general, aceptables pero no óptimos en ningún sentido significativo. Desde finales de la década de los cincuenta, el énfasis en problemas de diseño de control hizo que no sólo bastara diseñar uno de los muchos sistemas posibles que funcionaran más o menos adecuadamente, sino que se tendiese al diseño de sistemas óptimos en algún sentido determinado.

Conforme ha ido transcurriendo el tiempo, los procesos a controlar se han hecho cada vez más complejos. Actualmente, la descripción de un sistema moderno de control requiere una gran cantidad de ecuaciones; y la teoría de control clásica, que trata de sistemas lineales con una entrada y una salida, se vuelve absolutamente impotente ante sistemas no lineales de múltiples entradas y salidas. Hacia 1960, gracias a la disponibilidad de computadores digitales, se hizo posible el análisis de sistemas complejos en el dominio del tiempo. Desde entonces, se ha desarrollado la teoría de control moderna [OGA-93], basada en el análisis y síntesis en el dominio del tiempo, utilizando variables de estado, con lo que se posibilita afrontar la complejidad creciente de las plantas modernas y las estrictas especificaciones de exactitud, peso y costo en aplicaciones militares, espaciales e industriales. Posiblemente, la primera aplicación de control digital fue la de orientación espacial a finales de los años cincuenta. La mayoría de los primeros controladores digitales eran simplemente emuladores de sus equivalentes analógicos aunque proporcionaban una mayor flexibilidad. Esta flexibilidad abrió nuevas posibilidades al campo de la ingeniería de control tales como el control óptimo y el control adaptativo.

Actualmente, en el reino del análisis teórico de sistemas de control, los sistemas lineales siguen siendo los predominantes. La razón principal de ello estriba en que los sistemas no lineales son mucho más difíciles de analizar y no existe una teoría de control de sistemas no lineales que se acerque lo más mínimo a las elegantes teorías sobre sistemas lineales. Esencialmente, no existe un método sistemático de resolver analíticamente sistemas de ecuaciones diferenciales no lineales. Desafortunadamente, las alinealidades se presenta en la práctica totalidad de los sistemas reales y la hipótesis de linealidad dista mucho, en la mayoría de los casos, de ser rigurosa. Es por todo esto que en los años setenta y ochenta aparecieran nuevos paradigmas que intentaran abordar el problema del control de sistemas no lineales. Dos de tales paradigmas fueron las redes neuronales artificiales y los sistemas basados en lógica difusa.

Desde el punto de vista funcional, el sistema a controlar se puede definir como una "caja negra" cuyas salidas dependerán de alguna manera de sus entradas, entre las que se encuentran las señales de control. De la misma manera, el controlador se puede considerar como otra "caja negra" cuyas entradas serán las variables que definen el estado de la planta y cuyas salidas serán las señales de control que deberán ser capaces de llevar el estado de la planta al estado deseado en el menor espacio de tiempo posible, según sean las especificaciones exigidas al problema. Desde este punto de vista, el problema del control se interpreta como un problema de aproximación funcional. Tanto las redes neuronales como los sistemas difusos son, bajo ciertas hipótesis, aproximadores universales; esta característica dota a ambos paradigmas de la posibilidad de ser utilizados para resolver el problema del control de sistemas no lineales.

1.2 Descripción de la labor realizada

El trabajo de tesis doctoral presentado en esta memoria es una contribución al diseño automático de sistemas difusos aplicados al campo del control.

Según lo dicho en la sección 1.1, la construcción de un controlador difuso se puede tratar como un problema de aproximación funcional. A partir de una serie de datos de entrada/salida que pertenecen a la función que queremos aproximar, sería deseable

poder encontrar una arquitectura, y unos valores de los parámetros libres que definen dicha arquitectura, que sean capaces de aproximar la función generada por el sistema difuso a la función deseada, cumpliendo las especificaciones del problema. Parte del trabajo realizado en esta tesis intenta abordar y resolver este problema. Se presentará un algoritmo que, sin ningún conocimiento previo y a través de datos de E/S de la función a aproximar, irá construyendo sistemas difusos optimizando sus parámetros libres y analizando de qué forma se debe alterar la topología o estructura del sistema para poder mejorar el grado de aproximación, optimizándose, de esta forma, el compromiso existente entre precisión y complejidad del sistema.

La segunda parte de este trabajo aborda el mismo problema de síntesis automática de la topología de un controlador difuso y el cálculo de sus parámetros óptimos, pero con la dificultad de tener que realizar dicho proceso en tiempo real, es decir, de forma simultánea al propio proceso de control de la planta. Este complejo problema apenas ha sido desarrollado en la actualidad y es, a ciencia cierta, uno de los principales temas de investigación futuros.

A continuación se hace una breve descripción de cada uno de los capítulos de los que consta la presente memoria:

- **Capítulo 1:** Se realiza una breve revisión histórica del tema del control automático y se describe el trabajo que se va a presentar.

- **Capítulo 2:** Se presentan los fundamentos de la lógica difusa aplicada al campo de control que se consideran necesarios para facilitar la comprensión del resto de este trabajo de tesis y se introduce la notación utilizada.

- **Capítulo 3:** Se propone una nueva metodología para la construcción de sistemas difusos a partir de vectores de entrada/salida. Se intenta optimizar tanto la topología del sistema difuso como el valor de las magnitudes de los parámetros que lo definen.

- **Capítulo 4:** Proporciona ejemplos de aplicación del algoritmo presentado en el capítulo 3 que ponen de manifiesto las características principales del mismo. Además se realiza una comparación con otros trabajos de la bibliografía enfocados a la aproximación funcional.

- **Capítulo 5:** Presenta una nueva metodología para la construcción de controladores difusos a partir de la propia experiencia de éstos durante su funcionamiento en tiempo real (auto-aprendizaje) usando un conocimiento previo reducido de la planta a controlar. Al igual que en capítulo 3, tanto la topología del controlador como el valor de las magnitudes de los parámetros que lo definen intentan ser optimizados.

- **Capítulo 6:** Proporciona ejemplos de aplicación del algoritmo presentado en el capítulo 5 que ponen de manifiesto sus características principales. Además se realiza una comparación con otros algoritmos existentes en la bibliografía destinados al control en tiempo real.

- **Capítulo 7:** Recoge las conclusiones y principales aportaciones del presente trabajo, así como las líneas a seguir en investigaciones futuras.

CAPÍTULO 2

FUNDAMENTOS

El cerebro biológico tiene la capacidad de procesar información imprecisa o cualitativa. Frases como “esta persona no es demasiado mayor”, “esta casa es medianamente grande” son analizadas perfectamente en nuestro cerebro a pesar de utilizar información imprecisa o vaga. Sin embargo, este conocimiento expresado en términos lingüísticos no lo puede procesar un computador digital convencional. La lógica difusa o borrosa es una herramienta eficaz para trabajar satisfactoriamente con variables lingüísticas.

En este capítulo se realiza una revisión sobre los conceptos más importantes en el desarrollo de la lógica difusa aplicada al campo del control. En las primeras tres secciones se introducen los conjuntos difusos, las operaciones entre éstos, las reglas difusas y la forma de razonar a partir de ellas. En la sección 2.5 se describe la estructura de un controlador difuso. Una de las propiedades matemáticas más importantes: “la lógica difusa como aproximador universal” se detalla en la sección 2.6. Finalmente, se presentan las principales ventajas de utilizar controladores difusos y se describe la estructura del controlador difuso y la notación que se utilizará a lo largo de este trabajo.

2.1 Introducción

La hipótesis de que todos los sistemas se pueden reducir a un conjunto exacto de ecuaciones diferenciales ha sido rebatida por la experiencia, que demuestra que los procesos reales complejos nunca se pueden modelar, medir ni controlar de forma exacta. A pesar de ello, tales procesos (como puede ser la conducción de un coche) son controlados satisfactoriamente por el hombre sin la necesidad de recurrir a modelos matemáticos complejos o a una comprensión profunda de los procesos físicos que entraña el problema. La dificultad estriba en cómo conseguir representar y computar de una manera sistemática y lógica procesos que están descritos de forma imprecisa.

Lotfi A. Zadeh, profesor en la Universidad de Berkeley, se dio cuenta de que en la mayoría de los procesos complejos era imposible obtener una descripción precisa de la dinámica de los mismos y que, además, ésta era innecesaria para su control efectivo. Para poder superar la necesidad de una representación precisa del conocimiento, Zadeh introdujo los conjuntos difusos [ZAD-65] que permitían tratar con formas aproximadas de razonamiento en las cuales el concepto de verdadero o falso es una cuestión de grado. En palabras de Dubois y Prade [DUB-91]: "*la principal motivación de la teoría de conjuntos difusos es el deseo de construir una base formal, cuantitativa, que sea capaz de asimilar la imprecisión del conocimiento humano conforme éste es expresado usando el lenguaje natural*". Es necesario puntualizar que la imprecisión que se modela con la lógica difusa no hace referencia a errores en las medidas, ni a variables aleatorias ni a procesos estocásticos. La imprecisión está en la misma esencia del lenguaje y del pensamiento humanos [TER-92].

En los últimos años, la lógica difusa ha sido un área de trabajo muy fructífera para el desarrollo de diferentes disciplinas (como pueden ser los problemas de interpolación, modelado e identificación de sistemas, reconocimiento y clasificación de patrones, etc.), aunque el área donde esta teoría ha proporcionado mayores resultados ha sido en el campo del control [YAG-85, YAS-85, CHR-93, SAK-93, KWO-95]. Esto es debido a que, esencialmente, los controladores difusos presentan la propiedad de que son capaces de convertir la estrategia lingüística de control de un experto humano en una estrategia

automática de control [LEE-90]. De esta forma, el control basado en lógica difusa puede considerarse como un paso adelante hacia el acercamiento entre el control convencional (matemático y preciso) y la forma de tomar decisiones propia del ser humano [GUP-80]. La metodología utilizada en sistemas de control difuso resulta de especial interés para su aplicación a sistemas complejos donde la teoría clásica no puede ser aplicada, o en sistemas en los cuales las fuentes de información son interpretadas de forma cualitativa, inexacta o con cierta incertidumbre.

2.2 Conjuntos difusos

Un conjunto clásico, como se denominan en este contexto para diferenciarlos de los conjuntos difusos, tiene una barrera o contorno claramente definida. Un elemento pertenece o no a dicho conjunto, sin existir grados intermedios de pertenencia. Por ejemplo se puede definir:

$$A = \{x / x < 4\}, \text{ con } x \in \ /$$

La función de pertenencia al conjunto *A* (también conocida como la función característica, la función de discriminación o la función de indicación) de un elemento cualquiera *x* puede ser expresada:

$$\mu_A(x) = \begin{cases} 1 & si\ x \in A \\ 0 & si\ x \notin A \end{cases} \qquad (2\text{-}1)$$

El conjunto *A* es matemáticamente equivalente a su función de pertenencia, en el sentido en el que conocer $\mu_A(x)$ es lo mismo que conocer el conjunto *A*. Sin embargo, el conocimiento humano sobre la pertenencia o no de un determinado elemento a un conjunto, no es siempre tan claro y fácil de catalogar como en la teoría clásica. Por ejemplo, si *Z* representa el conjunto de coches que poseen *una velocidad máxima alta*, el contorno que define dicho conjunto no está claramente determinado. Una de las aportaciones principales de Zadeh fue la introducción de grados de pertenencia en el intervalo real para conjuntos difusos. Esto admite la posibilidad de asignar diversos grados de certeza sobre la pertenencia o no de un elemento a un conjunto.

Expresado en notación matemática, un conjunto difuso *Z* está definido por pares de elementos en la forma $(x, \mu_z(x))$, donde *x* es el elemento propiamente dicho y $\mu_z(x)$ es el grado de pertenencia del elemento al conjunto *Z*. Ahora, los valores que puede tomar la

función $\mu_z(x)$ no están restringidos a {0,1}, sino que generalmente están normalizados en el intervalo [0,1]. Para un conjunto discreto, la definición de los elementos de dicho conjunto es:

$$Z = \sum_{i=1}^{n} x_i / \mu_z(x_i) = x_1 / \mu_z(x_1) + \cdots + x_n / \mu_z(x_n) \quad (2\text{-}2)$$

donde se ha utilizado el símbolo de suma '+' para definir la unión de cada uno de los elementos del conjunto. Los elementos se han definido como $x / \mu_z(x)$, donde el símbolo '/' significa que forma un par ordenado. Cuando se trata de un conjunto continuo, la definición sería:

$$Z = \int_X x / \mu_z(x) \quad (2\text{-}3)$$

Definición 1: **Conjunto difuso y Función de pertenencia**. Si X es una colección de objetos denotados genéricamente por x, un conjunto difuso A en el universo de trabajo X se define por un conjunto de pares ordenados en la forma:

$$A = \{(x, \mu_A(x)) / x \in X\}$$

Donde $\mu_A(x)$ es la función de pertenencia de un elemento x al conjunto A. La característica más importante de esta función es que puede tomar cualquier valor dentro del intervalo normalizado [0,1]. Como se puede comprobar, la definición de un conjunto difuso no es más que la extensión de la definición clásica de conjunto, permitiendo que la función de pertenencia tome valores diferentes de 0 y 1. Si la función de pertenencia de un conjunto difuso A está restringida a tomar valores en {0,1}, entonces A se reduce a un conjunto clásico.

2.2.1 Definiciones y Conceptos

En la exposición del conocimiento humano es muy frecuente encontrar términos lingüísticos para expresar el estado de una variable. Por ejemplo, si la variable es la temperatura, son habituales expresiones como *temperatura alta*, o *temperatura muy baja*, etc. Una variable lingüística es aquella cuyos valores no son numéricos, como las variables comunes, sino que son palabras o sentencias. La motivación para el uso de palabras o sentencias en lugar de utilizar números es que una caracterización lingüística en general contiene una mayor cantidad de información y es más comprensible para una persona aun siendo menos específica que una numérica.

Una variable lingüística se caracteriza por una quíntupla *(x, T(x), X, G, M)* en la cual *x* es el nombre de la variable (refiriéndonos al ejemplo anterior, *x* sería la temperatura), *T(x)* son los valores lingüísticos que dicha variable puede tomar (Baja, Media, Alta), cada uno de los cuales viene caracterizado por una función de pertenencia definida en el universo de trabajo *X*, mientras que *G* es la regla sintáctica para generar los nombres de los valores de *x*, y *M* es la regla para asociar a cada uno de los nombres su significado.

A continuación se van introducir las definiciones más usuales sobre conjuntos difusos, algunas de las cuales serán utilizadas a lo largo de los siguientes capítulos de esta memoria.

Definición 2: Sean *A* y *B* dos conjuntos difusos definidos en el mismo universo *X*. El conjunto *A* está contenido en el conjunto *B*, o equivalentemente, el conjunto *A* es un subconjunto de *B*, si y sólo si para todo elemento *x* de *A*, $\mu_A(x) \leq \mu_B(x)$. En forma matemática:

$$A \subseteq B \Leftrightarrow \mu_A(x) \leq \mu_B(x) \quad \forall x \in X$$

Dos conjuntos difusos *A* y *B* son iguales si y solo si $\mu_A(x) = \mu_B(x) \quad \forall x \in X$.

Definición 3: Un conjunto difuso *A* es convexo si y sólo si satisface la siguiente propiedad:

$$\forall x,y \in X , \forall \lambda \in [0,1] : \mu_A (\lambda x + (1-\lambda)y) \geq Min (\mu_A(x), \mu_A(y))$$

Definición 4: El soporte de un conjunto difuso *A* se define como:

$$S (A) = \{ x \in X / \mu_A(x) > 0 \}$$

Definición 5: La amplitud de un conjunto difuso *A* convexo, con soporte *S(A)* se define como:

$$Amplitud(A) = Sup(S(A)) - Inf(S(A))$$

Cuando se trabajan con conjuntos difusos convexos, como es el caso común de las funciones de pertenencia definidas para las variables de un sistema de control, las definiciones de Soporte y Amplitud se utilizan de forma indistinta. El Soporte es un

intervalo, y si es cerrado, las operaciones supremo e ínfimo se pueden sustituir por máximo y mínimo.

Definición 6: El núcleo de un conjunto difuso *A* se obtiene como:

$$Núcleo(A) = \{x \in X / \mu_A(x) = 1\}$$

Si tan sólo hay un punto donde se verifica dicha condición (como sucede con funciones de pertenencia triangulares o gaussianas), el valor x_p donde $\mu_A(x_p) = 1$ se denomina pico del conjunto *A*.

Definición 7: La altura de un conjunto difuso *A* se define como el valor de pertenencia máximo de los elementos del soporte de dicho conjunto o equivalentemente:

$$Altura(A) = Sup\{\mu_A(x) / x \in X\}$$

Definición 8: Un conjunto difuso está normalizado, si su altura 1, o equivalentemente, si su núcleo no es el conjunto vacío.

Definición 9: El concepto de *α-nivel* de un conjunto difuso *A*, representa el subconjunto de *A* en el cual los elementos $x \in X$, tienen un valor de pertenencia $\mu_A(x) \geq \alpha$.

$$\alpha\text{-nivel}(A) = \{x \in X / \mu_A(x) \geq \alpha\}$$

Si en lugar de utilizar el símbolo $\geq$ se utiliza >, se denomina *α-nivel fuerte*.

2.3 Operaciones entre conjuntos difusos

Al igual que se definen operaciones entre conjuntos clásicos, las mismas operaciones se pueden definir entre conjuntos difusos, donde ahora estas operaciones se realizan a través de las funciones de pertenencia. Las operaciones fundamentales entre conjuntos clásicos son la intersección, la unión y el complemento. La extensión de dichas operaciones no se define de forma unívoca en el contexto de la lógica difusa, a diferencia de lo que sucede en lógica clásica. Zadeh [ZAD-73] propuso las siguientes definiciones:

$$\forall x \in X : \mu_{A \cap B}(x) = Min(\mu_A(x), \mu_B(x))$$

$$\forall x \in X : \mu_{A \cup B}(x) = Max(\mu_A(x), \mu_B(x))$$

$$\forall x \in X : \mu_{A'}(x) = 1 - \mu_A(x)$$

Si se restringen los valores de $\mu_A(x)$ y $\mu_B(x)$ al conjunto {0,1}, estas operaciones son las mismas que las operaciones AND, OR, y Complemento de los conjuntos clásicos. Sin embargo, existe una gran cantidad de posibilidades para la implementación de la intersección y unión de conjuntos difusos como veremos a continuación.

2.3.1 Operadores de intersección: T-normas

Para la realización de los primeros controladores difusos inspirados en el diseño propuesto por Mamdani, los operadores que se utilizaban para la intersección y unión eran los operadores Mínimo y Máximo. Sin embargo, ha existido un gran esfuerzo investigador en el estudio de las propiedades de dichos operadores sobre la inferencia difusa [ALS-83, MIZ-89, KOV-92, CAR-93] demostrándose que, para ciertas aplicaciones, unos operadores trabajan mejor que otros [GUP-91, KRU-94].

Definición 10: Una función continua t: [0,1]x[0,1] → [0,1] es una T-norma si satisface los siguientes criterios:

$t(a,b) = t(b,a)$	(propiedad conmutativa)
$t(t(a,b),c) = t(a,t(b,c))$	(propiedad asociativa)
$si\ a \le c\ y\ b \le d \Rightarrow t(a,b) \le t(c,d)$	(función creciente)
$t(a,1) = a$	(elemento neutro)

Siendo *a,b,c,d* números reales en el intervalo [0,1]. De las propiedades elemento neutro y función creciente se deduce que $\forall a \in [0,1]: t(a,0) = 0$. Se puede demostrar que el operador mínimo es la T-norma mayor, es decir, para cualquier conjunto de entradas, cualquier otra T-norma produce un valor menor o igual al mínimo de las entradas. Algunas de las T-normas más utilizadas en la bibliografía son:

$t(a,b) = Min(a,b)$ $\qquad$ $t(a,b) = a \cdot b$

$t(a,b) = Max(0,a+b-1)$ $\qquad$ $t(a,b) = \begin{cases} a & si\ b = 1 \\ b & si\ a = 1 \\ 0 & otro\ caso \end{cases}$

2.3.2 Operadores de unión: T-conormas

Otra función de gran importancia en la composición de conjuntos difusos, es la operación de unión. Aun existiendo artículos destinados al estudio de sus propiedades matemáticas e influencia empírica en el proceso de inferencia difusa, la investigación sobre este tipo de operador es menos intensa que en el caso de las T-normas.

Definición 11: Una función continua t*: [0,1]*x*[0,1] $\rightarrow$ [0,1] es una T-conorma si satisface los siguientes criterios:

$t^*(a,b) = t^*(b,a)$	(propiedad conmutativa)
$t^*(t^*(a,b),c) = t^*(a,t^*(b,c))$	(propiedad asociativa)
$si\ a \leq c\ y\ b \leq d \Rightarrow t^*(a,b) \leq t^*(c,d)$	(función creciente)
$t^*(a,0) = a$	(elemento neutro)

Se puede comprobar que las características de elemento neutro y función creciente conllevan a que $\forall a \in [0,1] : t^*(a,1) = 1$. Alsina [ALS-83] define a una t-conorma como la función *S* de dos variables que hace que la función *T* definida como: $T(a,b) = 1 - S(1-a,1-b)$ sea una t-norma. Cuando la t-norma y t-conorma satisfacen la anterior relación, se dicen que son parejas de operadores conjugados. Ejemplos de ellos pueden ser el mínimo y máximo, el producto y el operador Goguen de unión ($a+b-a\cdot b$), etc. Se demuestra que el operador máximo es la T-conorma menor. Algunas de las T-conormas más utilizadas en la bibliografía son:

$t^*(a,b) = Max(a,b)$ $\qquad$ $t^*(a,b) = a+b-a\cdot b$

$t^*(a,b) = Min(1,a+b)$ $\qquad$ $t^*(a,b) = \begin{cases} a & si\ b = 0 \\ b & si\ a = 0 \\ 1 & otro\ caso \end{cases}$

2.3.3 Operadores de negación

El complemento de un conjunto difuso es una operación similar a la realizada en teoría clásica, aunque hay que hacer notar que la operación se realiza sobre los valores de las funciones de pertenencia.

Definición 12: Una función continua c: [0,1] → [0,1] es una función complemento difuso, si satisface las siguientes propiedades:

$c(0) = 1$

$si\ a < b\ \Psi c(a) > c\ (b)$ (función decreciente)

$c(c(a)) = a$ (propiedad de involución)

Ejemplos de estos tipos de funciones son el complemento ordinario: $c(a) = 1\text{-}a$, el de Yager: $c_w(a) = (1 - a^w)^{1/w}$ con $w \in (0,\infty)$, o el de Sugeno: $c_\lambda(a) = (1\text{-}a) / (1 + \lambda a)$ con $\lambda \in (-1,\infty)$.

Una propiedad importante de estas funciones es que se puede demostrar que toda operación complemento al menos tiene un punto de equilibrio. Se define un punto de equilibrio como un valor *'a'* tal que $c(a) = a$, lo que equivale a afirmar que el punto de equilibrio de un conjunto difuso *A*, es un valor tal que el grado de pertenencia al conjunto o a su complementario es el mismo.

2.4 Razonamiento Aproximado

Los elementos primitivos de representación del conocimiento humano, en el razonamiento aproximado, lo constituyen las proposiciones difusas. Una proposición difusa es una expresión del tipo: *"la temperatura es elevada"*, donde *temperatura* es una variable lingüística y *elevada* es uno de los atributos pertenecientes al conjunto *T(temperatura)*. Estas proposiciones primitivas pueden combinarse mediante conectivas lingüísticas de muy diversas formas, todas ellas derivadas de las cuatro siguientes operaciones fundamentales [LEE-90]:

- **Conjunción** (denotado por p ϖ q): donde se forma una nueva proposición basada en la veracidad de ambas proposiciones.

- **Disyunción** (denotado por p ω q): donde se forma una nueva proposición basada en la veracidad de una de las dos proposiciones

- **Implicación** (denotado por p $\rightarrow$ q): que usualmente toma la forma de reglas difusas de tipo *SI-ENTONCES*.

- **Negación** (denotado por ~p): junto con las proposiciones generadas utilizando conjunción, disyunción o implicación, una nueva proposición puede ser obtenida mediante el uso del prefijo "Es falso que ..." a una proposición existente.

Una regla difusa es, por tanto, una sentencia condicional expresada simbólicamente como

SI proposición_A ENTONCES proposición_B

donde tanto *proposición_A* como *proposición_B* pueden ser proposiciones difusas simples o compuestas. La parte *SI* se denomina antecedente de la regla mientras que la parte *ENTONCES* es el consecuente. Por ejemplo:

SI la temperatura es alta ENTONCES la presión es muy grande

donde *alta* y *muy grande* son etiquetas lingüísticas correspondientes a las variables lingüísticas *temperatura* y *presión* respectivamente. Una regla difusa relaciona el conjunto difuso asociado a la proposición antecedente con el conjunto difuso asociado a la proposición consecuente. Debido a ello, se observa una gran semejanza en la utilización de reglas difusas y el acceso a celdas de memoria asociativas, por lo que a veces a un conjunto de reglas difusas se le denomina “memoria difusa asociativa” FAM (Fuzzy Associative Memory, [KOS-92a, SUD-94]).

Una regla difusa del tipo *“SI X es P ENTONCES Y es B”* se suele abreviar en la forma $P \rightarrow B$. En esencia, esta expresión describe la relación entre dos variables X e Y lo que sugiere que una regla *SI-ENTONCES* puede ser definida como una relación binaria difusa R en el producto cartesiano $X \times Y$. Hay que hacer notar que el producto cartesiano difuso $X \times Y$ es una extensión del producto cartesiano clásico, puesto que ahora cada elemento $(x,y) \in X \times Y$ tiene asociado un grado de pertenencia $\mu_R(x,y)$ siendo:

$$R = P \rightarrow B = \int_{XxY} \mu_P(x) \rightarrow \mu_B(y)/(x,y)$$

y '$\rightarrow$' la función de implicación escogida, que puede ser la de Kleene-Dienes, Lukasiewicz, Zadeh, Gödel, Mamdani, etc. Por tanto, una relación difusa representa el grado de presencia o ausencia de asociación, interacción o interconexión entre elementos de dos o más conjuntos difusos [MEN-95].

Para la obtención de una consecuencia difusa, a partir de unos antecedentes dados, se utiliza lo que se denomina proceso de inferencia difuso [CAO-92, PIS-92, RED-92]. Este proceso consiste fundamentalmente en realizar cálculos entre los conjuntos difusos asociados a las proposiciones que componen las reglas para obtener una conclusión que puede ser difusa o numérica dependiendo de la aplicación (en teoría de control difuso la conclusión debe ser por lo general numérica). Existen dos mecanismos fundamentales de inferencia: el Modus Ponens Generalizado (GPM) y el Modus Tollens Generalizado (GTM). El esquema básico de ambos sistemas de inferencia es:

premisa 1: *X es P'*
premisa 2: *Si X es P ENTONCES Y es B*
---
conclusión: *Y es B'* (GPM)

premisa 1: *Y es B'*
premisa 2: *Si X es P ENTONCES Y es B*
---
conclusión: *X es P'* (GTM)

Donde *P*, *P'*, *B* y *B'* son etiquetas lingüísticas. Se puede observar que en lógica clásica una regla es disparada si y solo si la premisa es exactamente igual al antecedente de la regla. En lógica difusa, por el contrario, una regla es activada siempre y cuando exista una similitud entre la premisa y el antecedente de la regla, permitiendo por tanto distintos grados de activación de la regla. En teoría de control, el método de inferencia generalmente utilizado es el GPM. Para la obtención del conjunto difuso B' hay que realizar la operación $B' = P' \circ R$, donde el operador B se refiere a la composición de relaciones difusas, y *R* es la relación difusa generada por la regla que forma la premisa 2. Para poder definir la composición entre *P'* y *R* debemos antes introducir las dos operaciones fundamentales entre relaciones difusas: proyección y extensión cilíndrica.

Definición 13: Sea *R* una relación difusa sobre $U = \times_{i=1}^{n} U_i = U_1 \times U_2 \times \cdots \times U_n$ y sea $(i_1,...,i_k)$ un subconjunto de índices de $(1,...,n)$ con $k<n$, y $(j_1,...,i_l)$ su secuencia complementaria. Sea $V = \times_{m=1}^{k} U_{i_m}$. La proyección de *R* sobre *V* se define como:

$$proy(R)\ sobre\ V = \int_V \sup_{(x_{j_1},...,x_{j_l})} \mu_R(x_1,...,x_n)/(x_{i_1},...,x_{i_k})$$

En el caso binario, si definimos *R* sobre *X x Y* tendríamos:

$$proy(R)\ sobre\ \mathrm{Y} = \int_Y \sup_x \mu_R(x,y)/y$$

Definición 14: Sea *S* una relación difusa sobre $V = \times_{m=1}^{k} U_{i_m}$ (ver definición anterior). La extensión cilíndrica de *S* en *U* se define como:

$$ce(S) = \int_U \mu_S(x_{i_1},...,x_{i_k})/(x_1,...,x_n)$$

En el caso binario, si definimos *F* como un conjunto difuso sobre *Y* tendríamos:

$$ce(F) = \int_{X\times Y} \mu_F(y)/(x,y)$$

Ahora ya estamos en condiciones de definir el operador 'B':

Definición 15: Sea *A* un conjunto difuso definido sobre *X* y *R* una relación difusa definida sobre *X x Y*. Se define la composición de *A* y *R* (*AB R*) como un conjunto difuso *B* definido sobre *Y*, dado por la expresión:

$$B = A \circ R = proy(ce(A) \cap R)\ sobre\ Y$$

Si el operador de intersección se realiza con el *Min* y la proyección con el *Max* lo que tendremos es la regla de inferencia inicialmente propuesta por Zadeh:

$$\mu_B(y) = \max_x \min(\mu_A(x), \mu_R(x,y))$$

Igualmente, si la intersección se realiza con el producto tendríamos:

$$\mu_B(y) = \max_x \mu_A(x) \cdot \mu_R(x,y)$$

En el caso de utilizar reglas con premisas en el antecedente formadas por la conjunción de dos conjuntos difusos el esquema de razonamiento aproximado utilizado (para el GPM) es:

premisa 1: *X es P' AND Y es Q'*

premisa 2: *Si X es P AND Y es Q ENTONCES Z es B*

---

conclusión: *Z es B'*

La premisa 2 es una regla difusa de estructura $P \times Q \rightarrow B$. Esta regla puede ser transformada en una relación ternaria difusa *R*, en la cual los elementos pertenecientes al conjunto Cartesiano $X \times Y \times Z$ tienen un grado de pertenencia definido por $\mu_R(x,y,z) = \mu_{(P \times Q) \rightarrow B}(x,y,z)$. El conjunto difuso *B'* resultante de la inferencia puede ser calculado utilizando los operadores producto '$\bullet$' (para la función de implicación, para la intersección en la composición y para la conjunción de las premisas) y máximo '$\vee$' (para la proyección) en la forma:

$$\mu_{B'}(z) = \underbrace{\overbrace{\bigvee_{x,y}}^{\text{proyección}} \left\{ \left[\mu_{P'}(x) \overbrace{\bullet}^{\text{conjunción}} \mu_{Q'}(y) \right] \overbrace{\bullet}^{\text{intersección}} \left[\underbrace{\left(\mu_P(x) \overbrace{\bullet}^{\text{conjunción}} \mu_Q(y) \right)}_{\mu_R(x,y,z)} \overbrace{\bullet}^{\text{implicación}} \mu_B(z) \right] \right\}}_{\text{composición}}$$

Desarrollando la expresión anterior tenemos:

$$\begin{aligned} \mu_{B'}(z) &= \bigvee_{x,y} \left\{ \left[\mu_{P'}(x) \bullet \mu_{Q'}(y) \right] \bullet \left[\left(\mu_P(x) \bullet \mu_Q(y) \right) \bullet \mu_B(z) \right] \right\} \\ &= \bigvee_{x,y} \left\{ \mu_{P'}(x) \bullet \mu_{Q'}(y) \bullet \mu_P(x) \bullet \mu_Q(y) \right\} \bullet \mu_B(z) \\ &= \left(\bigvee_x \left\{ \mu_{P'}(x) \bullet \mu_P(x) \right\} \right) \bullet \left(\bigvee_y \left\{ \mu_{Q'}(y) \bullet \mu_Q(y) \right\} \right) \bullet \mu_B(z) \\ &= (\alpha_1 \bullet \alpha_2) \bullet \mu_B(z) = \alpha_{Regla} \bullet \mu_B(z) \end{aligned} \qquad (2\text{-}4)$$

donde se ha simbolizado con α_1 el grado de semejanza entre el valor *P* del antecedente de la regla y el valor *P'* medido, α_2 el grado de semejanza entre el valor *Q* del antecedente de la regla y el valor *Q'* medido y α_{Regla} la actividad total de la regla. Finalmente, si en lugar de tener una regla tenemos muchas, debemos recurrir a un último operador, llamado operador de agregación (generalmente el Max o la Suma de conjuntos difusos) de tal forma que podamos, a partir de una entrada difusa dada, obtener el conjunto difuso inferido a través de un conjunto de reglas difusas.

Existen otras muchas formas de realizar este proceso mediante la utilización de las funciones de implicación difusas. Un análisis más detallado de todo el proceso se puede encontrar en [DRI-93, JAG-95].

2.5 Estructura de un controlador difuso

Durante los últimos años han surgido diferentes modelos para el desarrollo de controladores difusos. Los componentes principales de un controlador difuso se ilustran en la figura 2.1, y vienen dados por:

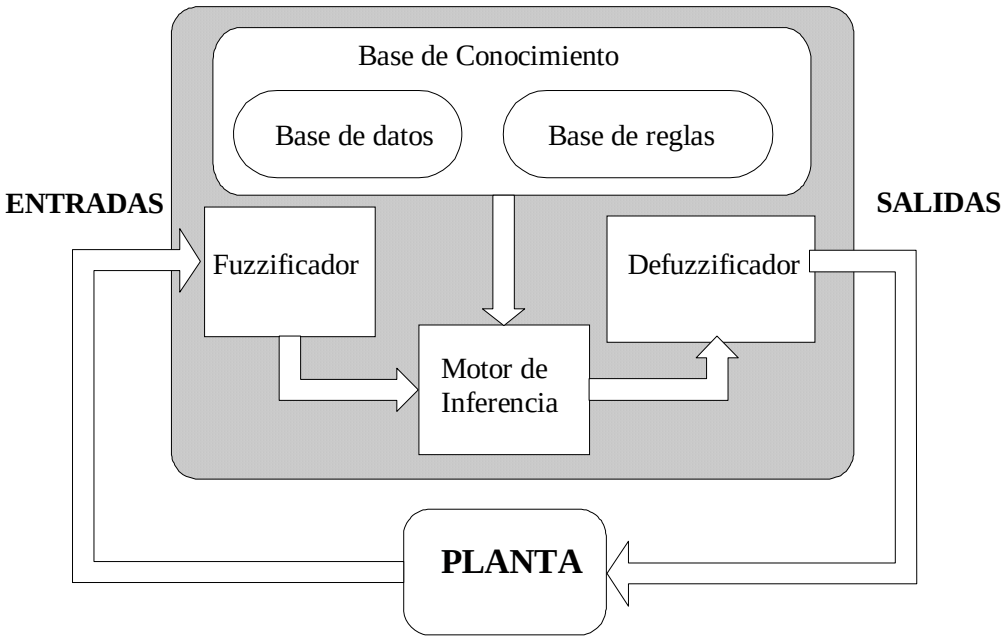


Fig. 2.1 Estructura de un controlador difuso

- Un bloque de conversión de entradas reales en un conjunto difuso, para su posterior utilización utilizando la teoría de conjuntos difusos. Es lo que se denomina proceso de "*fuzzificación*".

- Una base de conocimiento, consistente en la definición de los valores lingüísticos de cada una de las variables de entrada/salida del sistema y las reglas que lo conforman.

- Un motor de inferencia que actúa según las reglas y funciones de pertenencia descritas por el diseñador del controlador difuso. Es realmente el núcleo del

controlador difuso, y tiene la capacidad de simular el proceso de toma de decisiones de un operador humano.

- Una fase de conversión del conjunto difuso obtenido como conclusión en un valor numérico. Es lo que se denomina proceso de *"defuzzificación"*.

A continuación se describen con cierto detalle cada una de las funciones realizadas por los bloques más importantes de un controlador difuso.

2.5.1 Codificación de las entradas: "fuzzificación"

El primer paso para operar con un controlador difuso es transformar la información proveniente generalmente de sensores en términos de valores léxicos usados en las precondiciones de las reglas difusas. La fase de "fuzzificación" requiere dos tareas fundamentales:

- Medición de las variables de entrada al sistema.
- Realización de un ajuste de escala, de forma que se transforme el rango de entradas medido por los sensores en valores apropiados para la definición de los dominios de

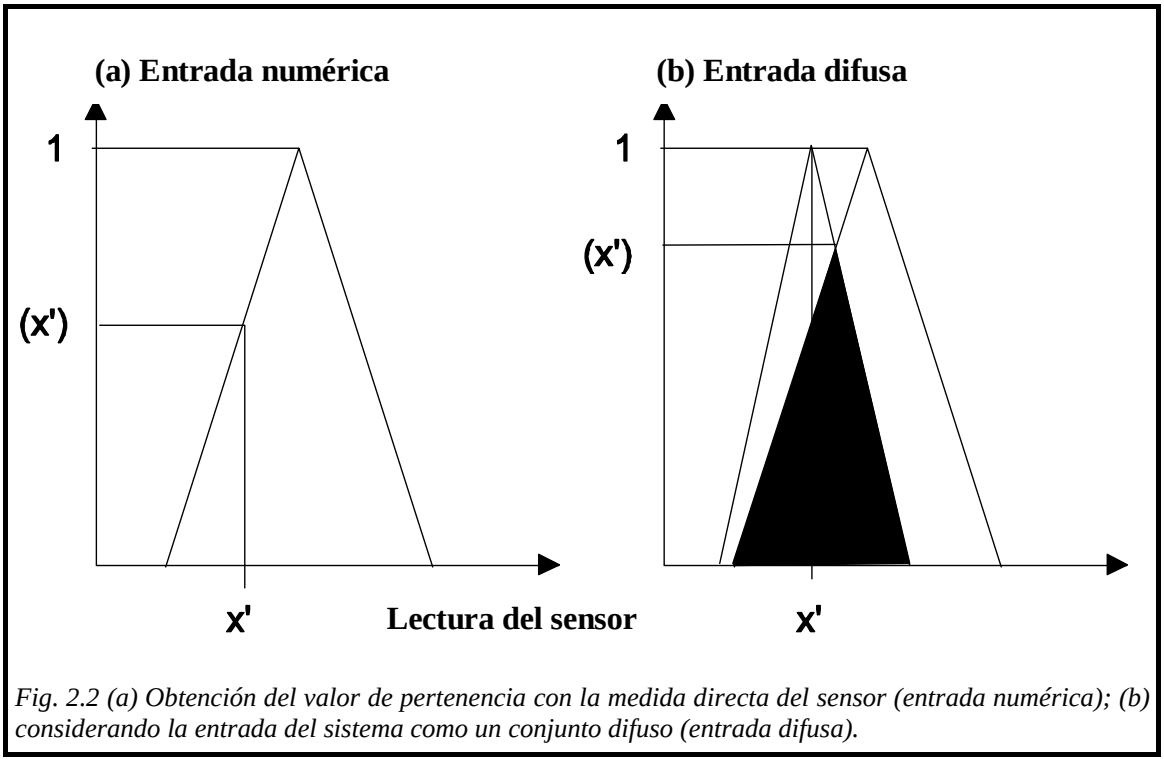


Fig. 2.2 (a) Obtención del valor de pertenencia con la medida directa del sensor (entrada numérica); (b) considerando la entrada del sistema como un conjunto difuso (entrada difusa).

las variables lingüísticas.

Si la salida del sensor es un valor numérico, el proceso de “fuzzificación” requiere comparar el valor numérico con el valor de las diferentes funciones de pertenencia de la variable correspondiente (caso (a) en la figura 2.2). Si la lectura del sensor contiene ruido o está deteriorada, una primera aproximación es modelar esta incertidumbre mediante una función triangular centrada en la lectura obtenida, y la base del triángulo proporcional a la desviación estándar de la lectura. En este caso el objetivo del proceso de “fuzzificación” es encontrar la intersección (usando el Min como T-norma) entre los diferentes valores lingüísticos de la variable y la función triangular formada por una determinada medida (caso (b) en la figura 2.2). El método de “fuzzificación” más utilizado es el primero de los mencionados donde, aunque la medida del sensor pueda tener un cierto error, dicha lectura es tomada directamente para determinar la salida del controlador. La razón principal para no implementar el segundo método de “fuzzificación” es la complejidad computacional que requiere.

2.5.2 Base de conocimiento de un controlador difuso

La base de conocimiento de un controlador difuso comprende la información necesaria para definir unívocamente las reglas y funciones de pertenencia de las variables del sistema, de forma que se logren los objetivos de control deseados. Se compone de:

La base de datos: encargada de la selección y descripción de los valores lingüísticos que definen las funciones de pertenencia de cada una de las variables de entrada y salida. La selección de la granularidad o densidad de las funciones de pertenencia es muy importante para la dinámica de la planta bajo control.

La base de reglas: que representa la política de control por medio de reglas difusas. Una vez establecidas las particiones de cada una de las variables de entrada, se tiene que relacionar las diferentes situaciones en que queda dividido el subespacio de entrada de un proceso con las acciones de control más adecuadas a cada una de esas situaciones. Se trata por tanto de determinar el valor de los consecuentes de las reglas que forman el conjunto de reglas del sistema.

2.5.3 Motor de inferencia

Es realmente el núcleo de operación de un controlador difuso. Su misión es calcular la salida difusa a partir de las entradas del sistema y conforme a las reglas y funciones de pertenencia definidas. Dependiendo del diseño de los consecuentes de las reglas difusas, se pueden distinguir en la bibliografía tres modos fundamentales de realizar dicho proceso:

- Modelo de Mamdani.
- Modelo de Takagi-Sugeno-Kang.
- Modelo de Tsukamoto.

2.5.3.1 Controlador difuso con modelo de Mamdani

Como se ha comentado con anterioridad, uno de los pioneros en la aplicación de la lógica difusa al área de control fue E. Mamdani [MAM-74]. En el controlador de Mamdani, los consecuentes de las reglas hacen referencia a cambios en la acción de control. Dichos cambios pueden ser expresados mediante términos lingüísticos o términos numéricos fijos. Suele estar diseñado para dos variables de entrada. La primera variable es la diferencia entre la señal proveniente de los sensores y la consigna deseada, por tanto será una variable de error. La segunda es la derivada del anterior error respecto del tiempo. Su funcionamiento es, por tanto, equivalente a un controlador PD clásico pero con ganancia no constante. Las reglas utilizadas son del tipo anteriormente presentadas:

SI X es P AND X' es Q ENTONCES Z es B.

2.5.3.2 Controlador difuso con modelo de Takagi-Sugeno-Kang

En 1985, Sugeno [SUG-85a, SUG-85b, TAK-85] realizó un estudio sobre las posibilidades de incluir en el consecuente nuevas formas de actuación. El nuevo modelo de reglas que propuso es de la forma:

Si X es P AND Y es Q ENTONCES Z es F(x,y)

donde *P* y *Q* continúan siendo conjuntos difusos, mientras que la función *F(x,y)* es una función numérica de las entradas del sistema. Usualmente, *F(x,y)* es una función lineal

de las entradas $F(x,y)= p_0 + p_1x + p_2y$, pero en general puede ser cualquier tipo de función polinómica. Cuando el grado del polinomio es uno, el sistema se denomina "Modelo de primer orden de Sugeno", el cual fue originalmente propuesto en [TAK-85, SUG-88]. Cuando $F(x,y)$ es una constante, el sistema obtenido es equivalente al modelo de Mamdani utilizando consecuentes no difusos. En la mayoría de los casos, el número de reglas necesarias para la definición de una función difusa es inferior utilizando el modelo de Sugeno que usando el modelo de Mamdani. Sin embargo, la interpretación de dichas reglas no resulta sencilla debido a que la conclusión es una función de las entradas y no una constante o un valor lingüístico como en el caso del controlador de Mamdani. Además, el conocimiento de un operador humano es difícil de plasmar en términos de reglas con este tipo de estructura.

2.5.3.3 Controlador difuso con modelo de Tsukamoto

En el modelo de Tsukamoto, los consecuentes de cada una de las reglas son funciones monótonas crecientes o decrecientes que dependen del nivel de activación de la regla. La estructura de las reglas es, en este modelo, de la forma:

$$R_i : Si\ X\ es\ P\ AND\ Y\ es\ Q\ ENTONCES\ Z\ es\ F(\alpha_i)$$

Debido a que la salida de cada una de las reglas individuales es también función del α-corte de la premisa, el conjunto de reglas resultante es más difícil de interpretar por un operador humano que en el modelo de Mamdani.

2.5.4 Decodificación de la salida: "defuzzificación"

La fase de "defuzzificación" es necesaria para convertir la salida difusa de un controlador en una representación numérica. Básicamente, este bloque funcional realiza una aplicación desde el espacio de acciones de control difusas (construidas en el dominio de definición de la variable de salida del controlador) sobre un espacio numérico (no difuso) de acciones de control sobre la planta que se necesita gobernar. La razón por la que es necesario efectuar esta operación es porque, en aplicaciones industriales, la salida de un controlador debe ser de forma numérica debido a que los actuadores mecánicos, eléctricos, etc. sólo aceptan señales de esta naturaleza. Existen

cada vez más trabajos enfocados a la investigación de las propiedades de esta operación [ZHA-91, FIL-93, HEL-93, MAB-93, YAG-93, GEN-94].

La estrategia principal de la fase de “defuzzificación” es obtener el elemento numérico que mejor represente el conjunto difuso de salida. Desgraciadamente, no existe hasta la fecha una forma sistemática para la elección del método de “defuzzificación” más adecuado para cada aplicación. En la bibliografía, los métodos mayoritariamente utilizados son:

- Centro de Área (CDA)
- Centro de Sumas (CDS)
- Centro de Picos (CDP)
- Centro del Área Mayor (CDAM)
- Primer Máximo (PM)
- Media de Máximos (MDM)
- Método de calidad ξ

Centro de Área: Es uno de los métodos más utilizados, tanto en aplicaciones reales de control como en realizaciones electrónicas. La salida más representativa del conjunto difuso es determinada como el centro de área de dicha distribución. Para el caso de una distribución continua:

$$Z_{CDA} = \frac{\int_Z z\ \mu_{Z'}(z)\, dz}{\int_Z \mu_{Z'}(z)\, dz} \tag{2-5}$$

Donde Z' es el conjunto difuso final, resultado de combinar los diferentes valores lingüísticos de la variable de salida que son activados. Para una distribución discreta:

$$Z_{CDA} = \frac{\sum_{j=1}^{J} z_j\ \mu_{Z'}(z_j)}{\sum_{j=1}^{J} \mu_{Z'}(z_j)} \tag{2-6}$$

donde *J* es el número de niveles en que es cuantizado el dominio de salida.

Centro de Sumas: Es un método similar al CDA, pero en este caso se trata de evitar el complejo cálculo del conjunto de salida Z'. Para la obtención del elemento numérico se considera el área de cada uno de los valores lingüísticos de salida individualmente. En el caso frecuente de que exista solapamiento entre funciones de pertenencia adyacentes en el dominio de salida, dichas regiones serán reflejadas más de una vez. Para el caso continuo, la expresión matemática es:

$$Z_{CDS} = \frac{\int_Z z \sum_{i=1}^{n^o reglas} \mu_{Z_i'}(z)\, dz}{\int_Z \sum_{i=1}^{n^o reglas} \mu_{Z_i'}(z)\, dz} \tag{2-7}$$

Centro de Picos: Es un método frecuentemente utilizado debido a su reducida complejidad computacional comparada con los dos métodos anteriores ya que es independiente del soporte del conjunto difuso inferido por cada regla y de su forma. Este método toma en consideración sólo el punto en que las funciones de pertenencia individuales de la variable de salida tienen valor igual a la unidad (en el caso de estar normalizadas y ser éste único). Para el caso común de utilizar representaciones triangulares, trapezoidales o gaussianas, el valor considerado es el centro de la función. Si llamamos α_i al grado con el que el valor de entrada activa el antecedente de la regla *i* y c_i es el valor de pico del conjunto difuso de salida que se encuentra en el consecuente de la regla *i* (ver Definición 6), la salida defuzzificada mediante este método sería:

$$Z_{CDP} = \frac{\sum_{i=1}^{n^o reglas} \alpha_i c_i}{\sum_{i=1}^{n^o reglas} \alpha_i} \tag{2-8}$$

Centro del Área mayor: Cuando el conjunto difuso de salida de un controlador es un conjunto no convexo (esto puede suceder cuando por ejemplo dos reglas con consecuentes diferentes son activadas dando lugar a dos conjuntos convexos distintos), la salida numérica es obtenida como el centro de área del subconjunto convexo que tenga mayor área.

Primer máximo: Este método necesita la construcción inicial del conjunto de salida difuso Z'. Una vez calculado, se representa dicho conjunto con el valor numérico más pequeño del dominio de salida que tenga el grado de pertenencia mayor.

Si en lugar de utilizar el valor numérico más pequeño se utiliza el mayor, se obtiene el denominado "Método de "defuzzificación" del último máximo". Si se utiliza el valor medio del primer y último máximos, el método se denomina "Media de máximos".

Método de calidad ξ (EQM): Es posible que no todas las reglas que forman la base de conocimiento del controlador tengan la misma importancia o presenten el mismo grado de incertidumbre. En este método, esto viene reflejado por la amplitud del valor lingüístico que define el consecuente de la regla. Cuando el soporte de dicha función es muy grande, el grado de certeza o importancia de dicha regla debe ser menor. Si por el contrario el soporte es muy pequeño, ello significa que la desviación de la salida es muy pequeña en torno al valor central del consecuente. Sin embargo, utilizando métodos de "defuzzificación" como el Centro de Área, Centro de Sumas o Centro del Área Mayor, esto no se tiene en cuenta. Utilizando el algoritmo de Centro de Picos, este hecho tampoco se tiene en cuenta, ya que este método no se ve afectado ni por la forma ni por la amplitud del soporte de las funciones de pertenencia. H. Hellendoorn [HEL-93] propone introducir esta cualidad en la computación del valor numérico, dividiendo el grado de activación de cada una de las reglas por la amplitud del soporte de las funciones de pertenencia de los consecuentes:

$$Z_{\xi\text{-}Quality} = \frac{\sum_{i=1}^{n^o\,reglas} \frac{\alpha_i}{d_i^{\xi}} c_i}{\sum_{i=1}^{n^o\,reglas} \frac{\alpha_i}{d_i^{\xi}}} \qquad (2\text{-}9)$$

Cuanto mayor sea el parámetro ξ mayor será la importancia que se dé a la amplitud de las funciones de pertenencia. Por ejemplo, para ξ igual a cero, realmente se está computando el Centro de Picos.

2.5.4.1 Propiedades de un método de "defuzzificación" y comparación entre las diferentes alternativas

En este apartado se pretende discutir de forma comparativa las ventajas y desventajas del uso de cada uno de los métodos anteriormente expuestos. Para ello se definirán un conjunto de propiedades que en principio sería deseable que un método de "defuzzificación" pueda satisfacer. Dichos criterios son:

Continuidad: Un método es continuo cuando un pequeño cambio en las entradas del sistema repercute en un cambio de magnitud también pequeño en la variable de salida.

Unicidad: Un método de "defuzzificación" presenta esta propiedad cuando el algoritmo para la determinación del valor numérico se obtiene claramente sin ambigüedad. El único método que no tiene esta característica es el Centro del Área máxima.

Plausibilidad: La finalidad de la fase de "defuzzificación" es obtener un elemento numérico a partir de un conjunto difuso Z'. Un método de "defuzzificación" es plausible si dicho elemento tiene un valor cercano a la mitad del soporte del conjunto difuso y grado de pertenencia elevado.

Complejidad Computacional: Los métodos de "defuzzificación" que requieran el cálculo del conjunto difuso de salida Z' son costosos en tiempo y requieren un alto grado de computación. Por el contrario, métodos como CDP y EQM, al calcular la salida del defuzzificador directamente con la información de las conclusiones individuales de las reglas son computacionalmente menos complejos. Esta propiedad es de gran interés para el desarrollo de aplicaciones que trabajen en tiempo real.

Áreas Solapadas: Los métodos que necesitan construir el conjunto difuso de salida pueden o no tener en cuenta las zonas en las que funciones de pertenencia adyacentes o reglas con mismo consecuente se solapan. Es una propiedad deseable que la salida numérica esté afectada no sólo por la regla con grado de activación más elevado sino también por la cantidad de reglas que indican una determinada acción de control.

Utilizando la comparación realizada por [HEL-93, DRI-93], los métodos que mejores características presentan son el CDS y el EQM (ver tabla 2.1).

Propiedades	**CDA**	**CDS**	**CDP**	**CDAM**	**PM**	**MDM**	**EQM**
Continuidad	++	++	++	0	--	--	++
Unicidad	++	++	++	--	++	++	++
Plausibilidad	0	+	+	++	0	0	+
Complejidad Computacional	--	0	++	--	+	+	++
Áreas Solapadas	--	++	0	--	--	--	++

Tabla 2.1 Comparación entre los diferentes métodos de "defuzzificación"

2.6 La lógica difusa como aproximador universal

Una de las preguntas que más puede interesar en la concepción de un controlador difuso es la precisión con la que una función arbitraria puede ser aproximada mediante un conjunto de reglas y funciones de pertenencia. En el campo de las redes neuronales artificiales, se ha demostrado que un perceptrón de tres niveles puede aproximar tan bien como se requiera cualquier tipo de función real continua en un dominio compacto, con un número adecuado de neuronas en sus niveles [HOR-89]. El mismo resultado ha sido probado para un controlador difuso utilizando el diseño utilizado por Mamdani, pero con consecuentes numéricos en lugar de con variables difusas [WAN-92a]. Utilizando el teorema de Stone-Weierstrass, [BUC-92] demostró que un conjunto de controladores difusos pueden aproximar cualquier tipo de función real continua en un dominio compacto, donde los elementos básicos del controlador son valores lingüísticos representados mediante gaussianas, se utiliza el operador producto para la inferencia e implicación, y como defuzzificador el centro de picos. Un conjunto o tipo de controladores difusos 'Ψ' es universal si y sólo si dado un proceso cualquiera *P*, existe un controlador $\Phi \in \Psi$ de forma que la diferencia entre la salida del controlador difuso y la salida del proceso es menor que una constante ε, tan pequeña como se requiera.

Kosko [KOS-92a, KOS-92b, KOS-94] realizó un análisis muy similar, aunque empleando el concepto de regiones difusas (*fuzzy patches*). Buckley [BUC-93]

demostró que utilizando el modelo propuesto por Sugeno, también podía construir controladores difusos con capacidad de aproximar cualquier tipo de función continua. Las primitivas son:

- Los consecuentes de las reglas son funciones polinómicas.

- El proceso de "defuzzificación" es $Z^* = 3\lambda_i$, donde λ_i es el grado de semejanza entre las entradas reales del sistema y el antecedente de la regla R_i, en lugar de utilizar la normalización $\lambda_i^* = \lambda_i / 3\lambda_i$.

No obstante, como queda reflejado por J.L.Castro [CAS-95, CAS-96], existen diversos controladores difusos que no utilizan las primitivas ni la estructura de la que parten los anteriores resultados y son aproximadores universales. La principal razón es que en muchos diseños se requieren otros tipos de funciones de pertenencia como, por ejemplo, triangulares o trapezoidales. Con respecto al modelo de inferencia empleado, existe una gran diversidad de operadores de implicación al igual que metodologías de "defuzzificación". En [CAS-95] se demuestra que los controladores difusos formados por:

- funciones de pertenencia triangulares, trapezoidales, gaussianas,
- funciones de conjunción utilizando cualquier tipo de operador T-norma,
- funciones de implicación que solo deben satisfacer unas propiedades elementales.
- funciones de "defuzzificación" usuales en la bibliografía (Media de Máximos, Centro de Área, Primer Máximo, etc.) que sólo deben satisfacer que el punto de "defuzzificación" pertenezca al soporte,

son todos ellos aproximadores universales. También la fusión de los paradigmas neuronales y lógica difusa proporciona una herramienta eficaz para aproximar cualquier tipo de función [BUC-94]. Pese a estas demostraciones, no existe aún una metodología definida para la construcción de un sistema difuso (elección del número de funciones de pertenencia para cada una de las reglas, forma y localización de cada uno de los valores lingüísticos, número de reglas y significado de cada una de ellas, operadores de

fuzzificación, implicación, defuzzificador, T-normas, T-conormas, etc.) que nos asegure que el grado de aproximación quede siempre por debajo de una determinada cota dada.

2.7 ¿Por qué utilizar un controlador difuso?

El campo del control es, con enorme diferencia, el campo donde la lógica difusa ha tenido mayor éxito. El actual interés en la teoría difusa es, en gran parte, debido a la gran cantidad de aplicaciones y procesos en que los controladores difusos han sabido operar con efectividad. Algunas de las razones para que esto haya sido así son [WAN-94]:

- Uso efectivo de toda la información disponible: En cualquier trabajo de ingeniería se debe intentar utilizar toda la información que esté disponible. A falta de un modelo matemático del proceso a controlar (modelo que generalmente no se tiene), la mayor parte de la información proviene de: (1) los sensores, que nos dan medidas numéricas de las variables externas del proceso y (2) descripciones generalmente lingüísticas de expertos familiarizados con el proceso. Contrariamente a otros tipos de controladores, los controladores difusos sí son capaces de procesar e incorporar este tipo de información lingüística.
- Control de procesos no lineales: En la sección anterior se ha mencionado que los controladores difusos son aproximadores universales. Por tanto, siempre que se escoja un número de parámetros suficiente y se les dé un valor adecuado, podremos encontrar un controlador difuso apropiado para el control de cualquier sistema no lineal.
- El control difuso es fácil de comprender: Debido a que un controlador difuso emula la estrategia de control de un ser humano, su funcionamiento es fácil de entender por personal no especialista en control. Además, el uso de reglas del tipo SI-ENTONCES facilita que éstas puedan ser localmente manipuladas según la región del espacio donde actúen limitando, de esta forma, sus efectos en regiones vecinas. Esto permite construir algoritmos de una forma más sencilla y puede reducir el tiempo de desarrollo.
- Los controladores difusos son fáciles de implementar: Gracias a la existencia de chips comerciales que pueden ser programados y reprogramados externamente y a que los sistemas difusos admiten un alto nivel de paralelismo, la implementación de

un controlador difuso de altas prestaciones se puede realizar de una forma sencilla y rápida.

2.8 Estructura del controlador difuso utilizado en la presente memoria

En la presente memoria, los conjuntos difusos que particionan el espacio de entrada serán principalmente funciones triangulares aunque también se utilizarán puntualmente funciones gaussianas o pseudo-gaussianas. En el apéndice A se presentan en detalle las configuraciones de funciones de pertenencia empleadas.

En la sección 2.5.3 se presentaron los tres modelos de reglas que más se utilizan en el campo del control difuso. Aquí utilizaremos reglas de la forma (tipo Mamdani):

$$SI\ x_1\ es\ X_1^j\ AND\ ...\ AND\ x_N\ es\ X_N^j\ ENTONCES\ \ Z_j\ es\ R_j \tag{2-10}$$

siendo N el número de variables de entrada y R_j un número real, es decir, que:

$$\mu_{Z_j}(z) = \begin{cases} 1 & z = R_j \\ 0 & otro\ caso \end{cases} \tag{2-11}$$

por lo que, funcionalmente, pueden considerarse también del tipo Takagi-Sugeno-Kang de orden cero, es decir, que el orden del polinomio en el consecuente es de orden cero (una constante).

Cuando el consecuente de una regla es un conjunto difuso, los métodos de inferencia se complican lo que conlleva a que se necesita más tiempo para obtener la salida y a que los algoritmos que traten con este tipo de controladores sean más complejos. Como ya se comentó anteriormente, el tipo de reglas TSK presentan el inconveniente de que la interpretación de dichas reglas no resulta sencilla debido a que la conclusión es una función de las entradas y no una constante o un valor lingüístico como en el caso del controlador de Mamdani. De esa forma, el emplear este tipo de reglas “mezcla” entre el tipo Mamdani y el TSK reúne las ventajas de ambos ya que son reglas que se dejan tratar fácilmente desde el punto de vista matemático conservando, a su vez, su interpretabilidad.

En cuanto al uso de los distintos operadores que intervienen en el proceso de inferencia, se utilizará el operador producto '$\bullet$' para la función de implicación, para la intersección en la composición y para la conjunción de las premisas y el operador máximo '$\vee$' para la proyección. En definitiva, se usarán los mismos operadores con los que se obtuvo la expresión (2-4). De esta forma, utilizando la suma de conjuntos difusos como operador de agregación tendremos que el conjunto difuso inferido en la salida será de la forma:

$$\mu_{Z'}(z) = \underbrace{\sum_{j=1}^{nº\,reglas}}_{agregación} \alpha_j \underbrace{\bullet}_{implicación} \mu_{Z_j}(z) \tag{2-12}$$

siendo (ver apartado 2.4):

- α_j: grado con el que el vector de entrada activa la regla *j*.
- $\mu_{Z_j}(z)$: grado de pertenencia del conjunto difuso que aparece en el consecuente de la regla *j* en el punto *z*.

Ahora debemos aplicar que:

1. $\mu_{Z_j}(z)$ tiene la forma de la ecuación (2-11).
2. La salida del controlador debe ser un valor real por lo que hay que aplicar un método de "defuzzificación". El método que se utilizará aquí será el centro de área.
3. Las entradas al controlador no son difusas sino numéricas.

Analizando las ecuaciones (2-11) y (2-12) es fácil observar que el conjunto difuso inferido en la salida *Z'* sólo puede presentar grados de pertenencia no nulos en aquellos puntos dados por las conclusiones de las reglas. Matemáticamente:

$$Z' = \sum_{j=1}^{nº\,reglas} \alpha_j / R_j \tag{2-13}$$

Una vez que tenemos el conjunto difuso de salida, aplicamos el centro de área para poder obtener un valor numérico en la salida. Utilizando la expresión (2-5) tenemos:

$$Z_{CDA} \equiv \frac{\int_Z z\,\mu_{Z'}(z)\,dz}{\int_Z \mu_{Z'}(z)\,dz} = \frac{\sum_{j=1}^{nº\,reglas} R_j \cdot \alpha_j}{\sum_{j=1}^{nº\,reglas} \alpha_j} \tag{2-14}$$

Finalmente, debemos encontrar el valor de α_j en función de los conjuntos difusos definidos en las premisas de las reglas teniendo en cuenta que las entradas son valores numéricos exactos (no difusos). A partir de (2-4):

$$\alpha_j \equiv \alpha_j(\vec{x}') = \underbrace{\prod_{i=1}^{N}}_{conjunción} \alpha_{i,j}(x'_i) \tag{2-15}$$

siendo $\alpha_{i,j}$ el grado de semejanza entre el conjunto difuso X_i^j que aparece en la premisa de la regla *j* y el conjunto difuso $X_i^{'}$ en la entrada de la variable *j* (que en este caso es el valor numérico $x_i^{'}$), y $\vec{x}'$ el vector de entrada no difuso. De nuevo, recurriendo a la ecuación (2-4):

$$\alpha_{i,j}(x'_i) = \underbrace{\bigvee_{x_i}}_{proyección} \left\{ \mu_{X_i^{'}}(x_i) \overbrace{\bullet}^{intersección} \mu_{X_i^j}(x_i) \right\} \tag{2-16}$$

y, teniendo en cuenta que:

$$\mu_{X_i^{'}}(x_i) = \begin{cases} 1 & x_i = x_i^{'} \\ 0 & otro\ caso \end{cases} \tag{2-17}$$

tenemos finalmente que:

$$\alpha_{i,j}(x'_i) = \mu_{X_i^j}(x_i^{'}) \tag{2-18}$$

En resumen, a partir de (2-14), (2-15) y (2-18), la salida numérica de nuestro controlador para un vector de entrada no difuso $\vec{x}$ genérico vendrá dada por:

$$\tilde{F}(\vec{x}) \equiv Z_{CDA} = \frac{\sum_{j=1}^{nº\,reglas} R_j \cdot \prod_{i=1}^{N} \mu_{X_i^j}(x_i)}{\sum_{j=1}^{nº\,reglas} \prod_{i=1}^{N} \mu_{X_i^j}(x_i)} \tag{2-19}$$

Hay autores que denominan a esta forma de inferencia “razonamiento difuso simplificado” mientras que otros lo denominan “media ponderada”.

Una posible implementación de este tipo de controladores es la que se presenta en la figura 2.3. La capa 1 es la encargada de recoger las entradas del sistema. La capa 2 calcula el grado de pertenencia de cada componente de la entrada a cada una de las funciones de pertenencia definidas para dicha componente. Cada neurona de la capa 3 (habrá tantas como reglas) simplemente realiza el producto de sus entradas de modo que

su salida representará el grado de activación de la regla que tiene asignada. Finalmente, la neurona de salida calcula tanto el numerador como el denominador de (2-19) para así obtener el valor de salida del controlador.

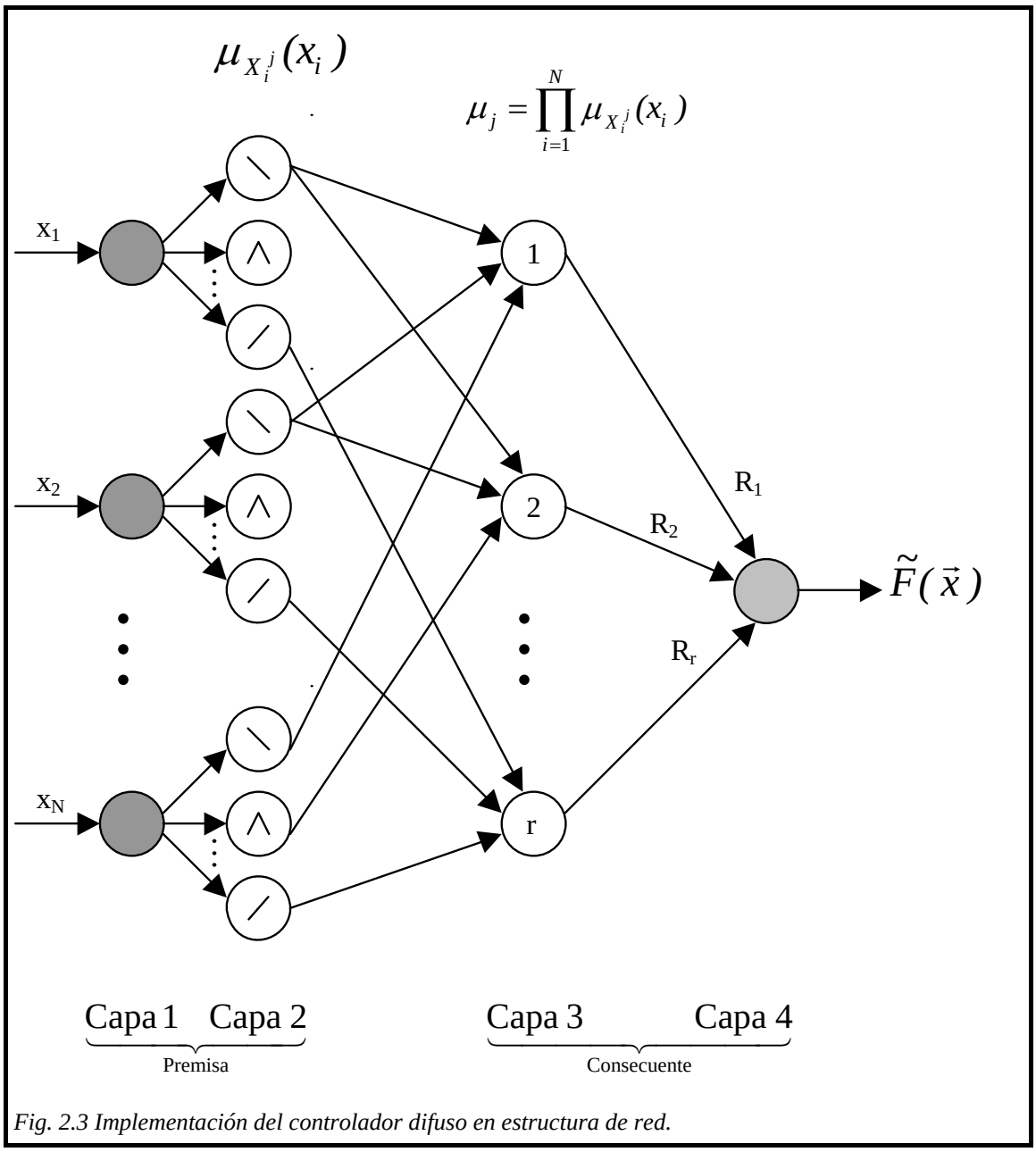


Fig. 2.3 Implementación del controlador difuso en estructura de red.

Como es fácil de apreciar a simple vista, la arquitectura de la figura 2.3 presenta gran analogía con una red neuronal por lo que a esta forma de implementación se le suele denominar red neuro-difusa. La ventaja principal de utilizar este tipo de diseño es que el proceso de inferencia se puede realizar en paralelo, reduciendo considerablemente el tiempo necesario para realizar dicha inferencia. Existen otras maneras de realizar la

implementación hardware de los controladores que se tratarán aquí, pero ésta es una cuestión que queda fuera del ámbito de esta tesis.

2.9 Notación empleada en esta memoria

A lo largo de este trabajo se empleará la siguiente notación:

- Número de variables de entrada (dimensión del espacio de entrada) ≡ N.
- Vector de entrada $\vec{x} = (x_1, x_2, ..., x_N)$.
- Número de funciones de pertenencia de la variable $i \equiv n_i$.
- Funciones de pertenencia de la variable $i \equiv \{X_i^1, X_i^2, ..., X_i^{n_i}\}$.
- Grado de pertenencia de la componente i del vector $\vec{x}$ a la función de pertenencia j de la variable $i \equiv \mu_{X_i^j}(x_i)$, con $j \in [1,n_i]$.

- Centro o pico de la función de pertenencia número j de la variable $v \equiv c_v^j$. El resto de los parámetros que definen las funciones de pertenencia dependerán del tipo de función escogido. En el apéndice A se recoge la notación empleada para dichos parámetros.

- Se considerarán conjuntos completos de reglas [DRI-93], por lo que el consecuente de cada una de ellas se podrá notar por $R_{i_1 i_2 ... i_N}$ de tal modo que las reglas vendrán expresadas de la forma:

$$SI\ x_1\ es\ X_1^{i_1}\ AND\ x_2\ es\ X_2^{i_2}\ AND\ ...\ AND\ x_N\ es\ X_N^{i_N}\ ENTONCES\ \ z=R_{i_1 i_2 ... i_N} \qquad (2\text{-}20)$$

- Teniendo en cuenta lo anterior, la función defuzzificada se podrá reescribir como[1]:

$$\tilde{F}(\vec{x}^k) = \frac{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(R_{i_1 i_2 ... i_N}\cdot\prod_{m=1}^{N}\mu_{X_m^{i_m}}(x_m^k)\right)}{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(\prod_{m=1}^{N}\mu_{X_m^{i_m}}(x_m^k)\right)} \qquad (2\text{-}21)$$

[1] Si alguna regla está marcada como no usada se considerará su grado de activación como cero, de tal forma que no intervendrá para el cómputo de la salida, ni en el numerador ni en el denominador de (2-21).

- Error Cuadrático Medio Normalizado *(ECMN)*: definido por

$$ECMN \equiv \sqrt{\frac{\overline{e^2}}{\sigma_z^2}} \tag{2-22}$$

donde σ_z^2 es la varianza de los datos de salida, y $\overline{e^2}$ el error cuadrático medio entre la salida obtenida y la deseada. El *ECMN* describe el grado de aproximación obtenido siendo su valor independiente de los factores de escala y del número de datos (ver Apéndice B). Además tiene la propiedad de que si aproximamos los datos por un valor constante e igual al valor medio de los datos, el *ECMN* valdrá la unidad.

CAPÍTULO 3

DISEÑO AUTOMÁTICO DE SISTEMAS DIFUSOS A PARTIR DE DATOS DE E/S

En el primer capítulo se hizo notar que, desde el punto de vista funcional, tanto el proceso a controlar como el propio controlador se pueden considerar "cajas negras" cuyas salidas dependen de alguna manera de las entradas. Con cierta frecuencia, en aplicaciones reales se pueden recopilar muestras de cómo se realiza dicha asociación entre entradas y salidas. Sería, por lo tanto, muy conveniente y útil disponer de una herramienta que fuese capaz de encontrar un sistema difuso lo más sencillo posible (según las especificaciones del problema) capaz de aproximar una cierta función a partir de muestras de la misma. El campo de la aproximación funcional tiene numerosas aplicaciones como pueden ser el diseño de controladores, el modelado de procesos, construcción de predictores, etc. Los sistemas difusos (ver capítulo 2) son aproximadores universales, lo que fundamenta su utilización en todos esos campos.

En este capítulo se propone una nueva metodología para el diseño automático de sistemas difusos a partir de la información proporcionada por un conjunto de vectores de E/S.

3.1 Introducción

El problema de estimar una función desconocida f a partir de muestras del tipo $\{(\vec{x}^k;z^k);\ k=1,2,...,K\}$, con $z^k = f(\vec{x}^k)\ 0\ /$ y $\vec{x}^k\ 0\ /^N$ (es decir, aproximación de funciones a partir de un conjunto finito de vectores de E/S), ha sido y seguirá siendo una cuestión fundamental en una gran variedad de disciplinas científicas e industriales. El objetivo principal es aprender una relación funcional desconocida entre vectores de entrada y salida usando un conjunto de ejemplos de entrenamiento. Una vez que dicha relación ha sido identificada, ésta puede ser usada para la obtención de nuevas salidas, dadas ciertas nuevas entradas. Las entradas y salidas pueden ser, en general, variables continuas o categóricas. Aquí utilizaremos variables de salida continuas, considerando, por tanto, problemas de regresión o aproximación de funciones, a diferencia de los problemas de clasificación donde la variable de salida es categórica [CHE-96].

Generalmente, hay dos formas de resolver el problema de la aproximación de funciones a partir de un conjunto de datos:

(1) Construyendo un modelo matemático de la función que ha de ser aprendida.
(2) A partir de un modelo libre.

Una limitación del primer método es que generalmente no existen modelos matemáticos precisos para sistemas alineales complejos o, de existir, su derivación requiere gran dificultad. Por lo tanto, la teoría tradicional basada en ecuaciones diferenciales sólo se puede aplicar a problemas lineales o simples.

En las últimas décadas, sistemas de modelo libre como las redes neuronales artificiales o los sistemas difusos han sido propuestos para resolver este problema. Los sistemas difusos proporcionan una alternativa atractiva a las “cajas negras” características de los sistemas neuronales, ya que su funcionamiento puede ser fácilmente entendido por un operador humano. En efecto, la popularidad y practicidad de los sistemas difusos derivan de su habilidad para expresar relaciones que son complejas o no suficientemente entendidas, en términos de reglas lingüísticas.

Históricamente, las bases de reglas de los sistemas difusos se han construido a partir del conocimiento de expertos humanos mientras que los pesos de las redes neuronales se han aprendido a través de datos [DRI-93]. De esta forma [BAR-98], hasta mediados de la década de los ochenta, la mayor parte de la investigación dedicada a controladores difusos se dirigía hacia la síntesis de reglas SI-ENTONCES que emularan las acciones de un operador. En esta temprana fase, la determinación de la estructura del controlador y la localización de las funciones de pertenencia de los espacios de entrada y salida se realizaban básicamente a través de ensayo y error [MAM-75, TON-79, TAK-83]. En [SUG-85a] se recogen las principales aplicaciones industriales de dicha época. Sin embargo, consultar el conocimiento de un experto puede ser tremendamente difícil y costoso. Adicionalmente, trasladar la experiencia de un operador directamente en reglas difusas es un proceso que se puede ver afectado por consideraciones subjetivas tanto del operador como del propio diseñador. En cualquier caso, el procedimiento no era el adecuado para obtener estructuras ni parámetros óptimos y era necesario encontrar algún procedimiento para sistematizar el proceso.

El primer intento relevante de sintetizar un algoritmo capaz de abordar el problema de la obtención de un sistema difuso y ajuste de sus parámetros a partir de datos de E/S fue el de Takagi y Sugeno, en 1985 [TAK-85]. En este excelente trabajo, los autores introducen el método de inferencia que lleva sus nombres, haciendo que el consecuente de cada regla fuera, en lugar de un conjunto difuso, una función lineal de las variables de entrada. Durante la evolución del algoritmo, los consecuentes de las reglas son optimizados mediante mínimos cuadrados mientras que los parámetros de las funciones de pertenencia son modificados mediante el método “complex”. Finalmente, utilizan un método heurístico-combinatorial para obtener las variables de entrada e ir dividiendo sucesivamente el dominio de las variables seleccionadas.

Sugeno y Kang [SUG-88a] implementaron un algoritmo parecido al anterior en el que intentan resolver el problema del ruido en los datos estableciendo un nuevo criterio de evaluación del sistema. Para ello, dividen el conjunto de datos de E/S en dos grupos e intentan minimizar un índice UC “unbiasedness criterion” [IVA-78] cuya finalidad es

evitar el sobre-ajuste de los datos. Un problema subyacente es cómo elegir ambos conjuntos de datos para que dicho método sea fiable.

A pesar de estas importantes contribuciones, la mayoría de los trabajos realizados seguían basados en estructuras preseleccionadas y ajustes de las funciones de pertenencia mediante ensayo y error. Sin embargo, en la década de los noventa empezaron a surgir numerosos trabajos sobre aproximación funcional, estando la mayoría principalmente dirigidos al ajuste de los parámetros que definen el sistema difuso a partir de datos de E/S, considerando fija su topología.

Así, en 1991, Wang y Mendel [WAN-91, WAN-92b] propusieron un nuevo método que combinaba información numérica y lingüística para la generación de una base de reglas. En su propuesta, el diseñador debía previamente dividir los dominios de entrada y salida usando una topología fija de funciones de pertenencia. Cada dato disponible generaba una regla, aquélla que más se activaba de entre todas las posibles, siendo dicha activación el grado de confianza asociado a la regla. Finalmente, comparando con las reglas proporcionadas por el operador (con su grado de confianza asociado) se iban seleccionando las reglas más fiables. Naturalmente, este algoritmo presenta inconvenientes: Por un lado, es muy sensible a la partición inicial y fija de funciones de pertenencia. Por otro, la existencia de ruido puede provocar casualmente una gran activación de una regla sin que ésta fuera la correcta. Adicionalmente, la configuración obtenida no responde a la minimización de ningún criterio como puede ser el error cuadrático medio. Métodos similares se pueden encontrar en [TAN-93] y [SUD-94].

En otro notable trabajo [WAN-92a], Li-Xin Wang, basándose en otros trabajos sobre funciones de base radial [CHE-91a] obtiene de forma sucesiva las reglas más importantes para aproximar una serie de datos, utilizando un algoritmo de mínimos cuadrados ortogonales (OLS). Para ello, el consecuente de las reglas ya no es un conjunto difuso sino un escalar.

Un avance significativo en la optimización de las funciones de pertenencia que particionan el dominio de entrada surgió con la aplicación de algoritmos basados en el

descenso en gradiente y la aparición de sistemas neuro-difusos [LIN-91, NOM-92, JAN-92, HOR-92, JAN-93]. El principal inconveniente de los métodos basados en el gradiente es que, si no se escoge bien el punto inicial, pueden conducir a mínimos locales que no sean suficientemente buenos. Además, debido a la no limitación del solapamiento entre las funciones de pertenencia durante el proceso, la interpretación de las reglas obtenidas suele deteriorarse.

En 1993, Sugeno y Yasukawa [SUG-93] presentan un nuevo algoritmo parecido al que se propuso en [SUG-88a], pero con varios matices nuevos importantes. Por un lado, el consecuente de las reglas volvía a ser un conjunto difuso, por lo que recuperaba la interpretación lingüística de las reglas y, por otro, se separaba el proceso de identificación de la estructura del sistema del de ajuste de parámetros. Para ello, se utilizaba un proceso de agrupamiento ("clustering") del conjunto de datos de salida. Dichos agrupamientos, utilizando la función inversa a la que queremos identificar, generaban otra serie de agrupamientos en el dominio de entrada generándose así una serie de asociaciones entre grupos de datos de entrada y de salida; y de dichas asociaciones: las reglas. Numerosos han sido los trabajos que han utilizado posteriormente el método del agrupamiento para generar la estructura del sistema [HIG-94, ABE-95, LAN-96, BAN-98]. Sin embargo, los métodos basados en agrupamientos son más adecuados para resolver problemas de clasificación que de aproximación de funciones, debido a que estos métodos no tienen en cuenta la capacidad interpolativa del sistema y se basan, principalmente, en conceptos de cercanía entre grupos de datos.

Finalmente, otros autores han optado por métodos más exóticos para realizar la aproximación de funciones. W-Y. Wang et al [WAN-95, WAN-97] utilizan B-Splines como funciones de pertenencia para realizar la partición de los dominios de entrada, consiguiendo buenas aproximaciones a costa de necesitar una gran cantidad de parámetros. Otros han intentado hacerlo en tiempo real, es decir, ajustando los datos conforme éstos se van obteniendo sin tener que usar nunca dos veces el mismo dato [SUG-91, NIE-96, JUA-98]. La aparición de los algoritmos genéticos también ha hecho mella en la síntesis de sistemas difusos a partir de datos numéricos [PAR-94, KIM-96].

En este capítulo se propone una nueva y completa metodología para el diseño automático de sistemas difusos a partir de la información proporcionada por un conjunto de vectores de E/S. Desde cero y a través del análisis de los datos, el algoritmo aquí propuesto irá construyendo y optimizando sistemas difusos completos. Es decir, se abordará tanto el problema de la identificación de la estructura del sistema como el ajuste de sus parámetros. Más concretamente habrá que:

- determinar qué variables son las más significativas.
- determinar cuántas funciones de pertenencia se deben utilizar para particionar el dominio de cada variable de entrada relevante y, de ahí, obtener el número máximo de reglas que se pueden utilizar.
- localizar y ajustar los parámetros que definen dichas funciones de pertenencia.
- ajustar los consecuentes de las reglas teniendo en cuenta que no todas pueden ser activadas por los datos de que dispongamos.

Finalmente, se obtendrán una serie de sistemas difusos optimizados, cada vez más precisos pero, a su vez, más complejos. El elegir cuál de los sistemas escoger es otro problema que se abordará en este capítulo. Ejemplos de aplicación y comparaciones con otras metodologías utilizadas en la bibliografía se mostrarán en el capítulo siguiente.

La organización de este capítulo será la siguiente:

En la sección 3.2 se planteará el problema a resolver en este capítulo de forma concreta. Posteriormente, se abordará la forma de encontrar los consecuentes óptimos para una configuración fija de funciones de pertenencia. En la sección 3.4 optimizaremos conjuntamente la distribución de las funciones de pertenencia y los consecuentes de las reglas. El problema de decidir qué variables utilizar y cuántas funciones de pertenencia elegir para cada variable escogida se aborda en la sección 3.5. En el siguiente apartado se comentará la cuestión del compromiso existente entre complejidad y exactitud en el campo de la aproximación funcional. Finalmente, se recogen algunas mejoras del algoritmo y las conclusiones de este capítulo.

3.2 Planteamiento del problema

En este capítulo nos proponemos aproximar una función cualquiera *F* de *N* entradas y una salida[1] a partir de un conjunto *D* de muestras del tipo $\{(\vec{x}^k;z^k);\ k=1,2,\ldots,K\}$, con $z^k = F(\vec{x}^k)$ y $\vec{x}^k$. Para ello, utilizaremos sistemas difusos del tipo presentado en las secciones 2.8 y 2.9 del capítulo anterior, es decir, utilizaremos un conjunto completo de reglas difusas SI-ENTONCES $R^*_{i_1 i_2 \ldots i_N}$ con la forma:

$$SI\ x_1\ es\ X_1^{i_1}\ AND\ x_2\ es\ X_2^{i_2}\ AND\ \ldots\ AND\ x_N\ es\ X_N^{i_N}\ \ ENTONCES\ \ z=R_{i_1 i_2 \ldots i_N} \tag{3-1}$$

siendo la salida del sistema (ver 2.9 para comprender la notación empleada):

$$\tilde{F}(\vec{x}^k;R,C) = \frac{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(R_{i_1 i_2 \ldots i_N}\cdot\prod_{m=1}^{N}\mu_{X_m^{i_m}}(x_m^k)\right)}{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(\prod_{m=1}^{N}\mu_{X_m^{i_m}}(x_m^k)\right)} \tag{3-2}$$

donde se ha recalcado de forma explícita la dependencia de la salida del sistema difuso no sólo con el vector de entrada sino también con la matriz de reglas *R* y con el conjunto de parámetros que describen las funciones de pertenencia *C*.

El objetivo de este capítulo será encontrar una configuración *C* y un conjunto de reglas *R* que, minimizando el error global:

$$J(R,C) = \sum_{k\in D}\left(F(\vec{x}^k) - \tilde{F}(\vec{x}^k;R,C)\right)^2 \tag{3-3}$$

mejor se adapte al compromiso entre complejidad y exactitud que se especifique. Para ello se debe determinar, según el grado de aproximación, qué variables tienen mayor importancia, qué topología debe tener el sistema difuso, dónde ubicar las funciones de pertenencia y cuáles deben ser las reglas. Al final del algoritmo se obtendrán una serie de sistemas difusos con diferentes topologías y grados de aproximación por lo que se deberá decidir, según las especificaciones del problema que se pretenda resolver, cuál, de entre todas ellas, será la escogida como solución.

---

[1] Aproximar funciones de *N* entradas y *M* salidas puede considerarse equivalente a aproximar *M* funciones de *N* entradas y una salida.

Como ya se dijo en el capítulo 2, el índice que se utilizará para expresar el grado de aproximación será el error cuadrático medio normalizado *ECMN* (ver ecuación 2-22) ya que éste es independiente de factores de escala y del número de datos.

El algoritmo se irá presentado de forma ascendente, es decir, primero se comenzará con el nivel más básico (la optimización de los consecuentes de las reglas) hasta llegar al más complejo (obtención de una nueva topología, seleccionando las variables más importantes).

3.3 Optimización de las reglas para una distribución fija de funciones de pertenencia

El primer caso que vamos a estudiar de la expresión (3-3) es aquél en el que tanto la distribución como el número de funciones de pertenencia son fijos y, por lo tanto, $\tilde{F}=\tilde{F}(\vec{x},R)$. A partir de (3-2), es obvio que $\tilde{F}$ no sólo es continua y derivable como función de los consecuentes de las reglas ($R_{i_1 i_2 \ldots i_N}$), sino que esta dependencia es lineal. Esta propiedad nos va a permitir encontrar, como veremos a continuación, los consecuentes óptimos de las reglas de forma exacta[2]. En primer lugar analizaremos el caso unidimensional por sencillez y así abordar con más facilidad el caso general en la sección 3.3.2.

3.3.1 Análisis para funciones con una sola variable de entrada

En el caso de aproximar funciones que dependan de una sola variable de entrada, (3-2) quedaría de la siguiente manera:

$$\tilde{\mathrm{F}}(x_1; R_1, R_2, \ldots, R_{n_1}) = \frac{\sum_{i_1=1}^{n_1} R_{i_1} \cdot \mu_{X_1^{i_1}}(x)}{\sum_{i_1=1}^{n_1} \mu_{X_1^{i_1}}(x)} \qquad (3\text{-}4)$$

donde se ha usado intencionadamente el subíndice i_1 con el fin de mantener la compatibilidad con las expresiones que se obtendrán en el caso general.

[2] Otras formas diferentes de obtener estos consecuentes se pueden encontrar en [TAK-85], [SUG-88a] y [WAN-92a].

En la expresión anterior quizás se vea con más claridad la dependencia lineal de nuestra función "defuzzificada" con cada uno de los consecuentes de las reglas (que aquí son nuestras únicas variables a optimizar). Si derivamos $\widetilde{F}$ con respecto a uno cualquiera de los consecuentes, por ejemplo R_{j_1}, obtendremos:

$$\frac{\partial \widetilde{F}(x;R)}{\partial R_{j_1}} = \frac{\mu_{X_1^{j_1}}(x)}{\sum_{i_1=1}^{n_1} \mu_{X_1^{i_1}}(x)} \tag{3-5}$$

expresión que es independiente de *R* (como cabía de esperar, por la linealidad).

Para encontrar los valores que minimizan *J*, derivamos con respecto a R_{j_1} en la ecuación (3-3) e igualamos a cero.

$$\frac{\partial \mathrm{J(R)}}{\partial \mathrm{R_{j_1}}} = -2 \cdot \sum_{k \in D} \left[\left(F(x^k) - \widetilde{F}(x^k; R, C) \right) \cdot \left(\frac{\partial \widetilde{\mathrm{F}}(\mathrm{x^k; R})}{\partial \mathrm{R_{j_1}}} \right) \right] = 0 \tag{3-6}$$

Substituyendo las ecuaciones anteriores en esta última, obtenemos el siguiente sistema de ecuaciones:

$$\sum_{k \in D} \left[\left(F(\vec{x}^k) - \frac{\sum_{i_1=1}^{n_1} \left(R_{i_1} \cdot \mu_{X_1^{i_1}}(x^k) \right)}{\sum_{i_1=1}^{n_1} \mu_{X_1^{i_1}}(x^k)} \right) \cdot \frac{\mu_{X_1^{j_1}}(x^k)}{\sum_{i_1=1}^{n_1} \mu_{X_1^{i_1}}(x^k)} \right] = 0 \tag{3-7}$$

Ordenando términos e intercambiando el orden de las sumatorias tenemos:

$$\sum_{i_1=1}^{n_1} R_{i_1} \cdot \sum_{k \in D} \left(\frac{\mu_{X_1^{i_1}}(x^k)}{\sum_{i_1=1}^{n_1} \mu_{X_1^{i_1}}(x^k)} \cdot \frac{\mu_{X_1^{j_1}}(x^k)}{\sum_{i_1=1}^{n_1} \mu_{X_1^{i_1}}(x^k)} \right) = \sum_{k \in D} \left(F(\vec{x}^k) \cdot \frac{\mu_{X_1^{j_1}}(x^k)}{\sum_{i_1=1}^{n_1} \mu_{X_1^{i_1}}(x^k)} \right) \tag{3-8}$$

Definiendo

$$S_{i_1, j_1} \equiv \sum_{k \in D} \left(\frac{\mu_{X_1^{i_1}}(x^k)}{\sum_{i_1=1}^{n_1} \mu_{X_1^{i_1}}(x^k)} \cdot \frac{\mu_{X_1^{j_1}}(x^k)}{\sum_{i_1=1}^{n_1} \mu_{X_1^{i_1}}(x^k)} \right) \tag{3-9}$$

$$S_{F,j_1} \equiv \sum_{k \in D} \left(F(\vec{x}^k) \cdot \frac{\mu_{X_1^{j_1}}(x^k)}{\sum_{i_1=1}^{n_1} \mu_{X_1^{i_1}}(x^k)} \right) \tag{3-10}$$

podemos expresar la ecuación (3-8) de la siguiente forma:

$$\sum_{i_1=1}^{n_1} R_{i_1} \cdot S_{i_1,j_1} = S_{F,j_1} \tag{3-11}$$

siendo j_1 cualquier índice entre 1 y n_1. Haciendo este barrido completo, podemos expresar el conjunto completo de ecuaciones en forma matricial:

$$\begin{pmatrix} S_{1,1} & S_{1,2} & \dots & S_{1,n_1} \\ S_{2,1} & S_{2,2} & \dots & S_{2,n_1} \\ S_{3,1} & S_{3,2} & \dots & S_{3,n_1} \\ \vdots & \vdots & \vdots & \vdots \\ S_{n_1,1} & S_{n_1,2} & \dots & S_{n_1,n_1} \end{pmatrix} \bullet \begin{pmatrix} R_1 \\ R_2 \\ R_3 \\ \vdots \\ R_{n_1} \end{pmatrix} = \begin{pmatrix} S_{F,1} \\ S_{F,2} \\ S_{F,3} \\ \vdots \\ S_{F,n_1} \end{pmatrix} \tag{3-12}$$

siendo la matriz simétrica, como se puede comprobar sin más que observar la expresión (3-9). Así pues, el problema de hallar los mejores consecuentes para un conjunto de reglas fijo se puede resolver de forma exacta para funciones de una sola entrada. Dichas soluciones se obtienen sin más que resolver el sistema de ecuaciones anterior, lo que implica invertir una matriz $n_1 \times n_1$. En el siguiente sub-apartado abordaremos el caso general donde comprobaremos que la matriz obtenida sigue siendo bidimensional y simétrica.

3.3.2 Análisis general: funciones con N entradas

En este apartado se realizará un tratamiento paralelo al de la sección anterior pero para el caso general de funciones N-dimensionales. Ahora, la función “defuzzificada” viene expresada por la expresión general (3-2). Derivando con respecto a uno de los consecuentes de las reglas, por ejemplo $R_{j_1 j_2 \dots j_N}$, obtenemos:

$$\frac{\partial \widetilde{\mathrm{F}}(\vec{\mathrm{x}};\mathrm{R})}{\partial \mathrm{R}_{\mathrm{j_1 j_2 \cdots j_N}}} = \frac{\prod_{m=1}^{N} \mu_{X_m^{j_m}}(x_m)}{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(\prod_{m=1}^{N}\mu_{X_m^{i_m}}(x_m)\right)} \tag{3-13}$$

Al igual que antes, los consecuentes óptimos que buscamos deben anular la primera derivada de *J*, que viene expresada por:

$$\frac{\partial \mathrm{J}(\mathrm{R})}{\partial R_{j_1 j_2 \cdots j_N}} = -2 \cdot \sum_{k \in D}\left[\left(F(\vec{x}^{\,k}) - \widetilde{F}(\vec{x}^{\,k};R,C)\right)\cdot\left(\frac{\partial \widetilde{\mathrm{F}}(\vec{\mathrm{x}}^{\,\mathrm{k}};\mathrm{R})}{\partial R_{j_1 j_2 \cdots j_N}}\right)\right] = 0 \tag{3-14}$$

Substituyendo en esta última expresión, obtenemos:

$$\begin{aligned}&\sum_{k \in D} \frac{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(R_{i_0 i_1 \ldots i_{N-1}} \cdot \prod_{m=1}^{N}\mu_{X_m^{i_m}}(x_m^k)\right)}{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(\prod_{m=1}^{N}\mu_{X_m^{i_m}}(x_m^k)\right)} \cdot \frac{\prod_{m=1}^{N}\mu_{X_m^{j_m}}(x_m^k)}{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(\prod_{m=1}^{N}\mu_{X_m^{i_m}}(x_m^k)\right)} = \\ &\sum_{k \in D} F(\vec{x}^{\,k}) \cdot \frac{\prod_{m=1}^{N}\mu_{X_m^{j_m}}(x_m^k)}{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(\prod_{m=1}^{N}\mu_{X_m^{i_m}}(x_m^k)\right)}\end{aligned} \tag{3-15}$$

Intercambiando el orden de los sumatorios e introduciendo la siguiente notación:

$$S_{i_1 i_2 \ldots i_N, j_1 j_2 \ldots j_N} \equiv \sum_{k \in D} \frac{\prod_{m=1}^{N}\mu_{X_m^{i_m}}(x_m^k)}{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(\prod_{m=1}^{N}\mu_{X_m^{i_m}}(x_m^k)\right)} \cdot \frac{\prod_{m=1}^{N}\mu_{X_m^{j_m}}(x_m^k)}{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(\prod_{m=1}^{N}\mu_{X_m^{i_m}}(x_m^k)\right)} \tag{3-16}$$

$$S_{F, j_1 j_2 \ldots j_N} \equiv \sum_{k \in D} F(\vec{x}^{\,k}) \cdot \frac{\prod_{m=1}^{N}\mu_{X_m^{j_m}}(x_m^k)}{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(\prod_{m=1}^{N}\mu_{X_m^{i_m}}(x_m^k)\right)} \tag{3-17}$$

obtenemos el siguiente sistema de ecuaciones lineales:

$$\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(R_{i_1 i_2 \ldots i_N} \cdot S_{i_1 i_2 \ldots i_N, j_1 j_2 \ldots j_N}\right) = S_{F, j_1 j_2 \ldots j_N} \tag{3-18}$$

donde j_1 va desde 1 hasta n_1, j_2 desde 1 hasta n_2, etc. A modo de ejemplo, en el caso de funciones de tres entradas, con tres funciones de pertenencia para cubrir el rango de la

primera variable ($n_1=3$) y dos para la segunda y tercera entradas ($n_2=n_3=2$), tendríamos el siguiente sistema de 12 ecuaciones con 12 incógnitas

$$\begin{pmatrix} S_{111,111} & S_{111,112} & S_{111,121} & S_{111,122} & S_{111,211} & S_{111,212} & S_{111,221} & S_{111,222} & S_{111,311} & S_{111,312} & S_{111,321} & S_{111,322} \\ S_{112,111} & S_{112,112} & S_{112,121} & S_{112,122} & S_{112,211} & S_{112,212} & S_{112,221} & S_{112,222} & S_{112,311} & S_{112,312} & S_{112,321} & S_{112,322} \\ S_{121,111} & S_{121,112} & S_{121,121} & S_{121,122} & S_{121,211} & S_{121,212} & S_{121,221} & S_{121,222} & S_{121,311} & S_{121,312} & S_{121,321} & S_{121,322} \\ S_{122,111} & S_{122,112} & S_{122,121} & S_{122,122} & S_{122,211} & S_{122,212} & S_{122,221} & S_{122,222} & S_{122,311} & S_{122,312} & S_{122,321} & S_{122,322} \\ S_{211,111} & S_{211,112} & S_{211,121} & S_{211,122} & S_{211,211} & S_{211,212} & S_{211,221} & S_{211,222} & S_{211,311} & S_{211,312} & S_{211,321} & S_{211,322} \\ S_{212,111} & S_{212,112} & S_{212,121} & S_{212,122} & S_{212,211} & S_{212,212} & S_{212,221} & S_{212,222} & S_{212,311} & S_{212,312} & S_{212,321} & S_{212,322} \\ S_{221,111} & S_{221,112} & S_{221,121} & S_{221,122} & S_{221,211} & S_{221,212} & S_{221,221} & S_{221,222} & S_{221,311} & S_{221,312} & S_{221,321} & S_{221,322} \\ S_{222,111} & S_{222,112} & S_{222,121} & S_{222,122} & S_{222,211} & S_{222,212} & S_{222,221} & S_{222,222} & S_{222,311} & S_{222,312} & S_{222,321} & S_{222,322} \\ S_{311,111} & S_{311,112} & S_{311,121} & S_{311,122} & S_{311,211} & S_{311,212} & S_{311,221} & S_{311,222} & S_{311,311} & S_{311,312} & S_{311,321} & S_{311,322} \\ S_{312,111} & S_{312,112} & S_{312,121} & S_{312,122} & S_{312,211} & S_{312,212} & S_{312,221} & S_{312,222} & S_{312,311} & S_{312,312} & S_{312,321} & S_{312,322} \\ S_{321,111} & S_{321,112} & S_{321,121} & S_{321,122} & S_{321,211} & S_{321,212} & S_{321,221} & S_{321,222} & S_{321,311} & S_{321,312} & S_{321,321} & S_{321,322} \\ S_{322,111} & S_{322,112} & S_{322,121} & S_{322,122} & S_{322,211} & S_{322,212} & S_{322,221} & S_{322,222} & S_{322,311} & S_{322,312} & S_{322,321} & S_{322,322} \end{pmatrix} \bullet \begin{pmatrix} R_{111} \\ R_{112} \\ R_{121} \\ R_{122} \\ R_{211} \\ R_{212} \\ R_{221} \\ R_{222} \\ R_{311} \\ R_{312} \\ R_{321} \\ R_{322} \end{pmatrix} = \begin{pmatrix} S_{F,111} \\ S_{F,112} \\ S_{F,121} \\ S_{F,122} \\ S_{F,211} \\ S_{F,212} \\ S_{F,221} \\ S_{F,222} \\ S_{F,311} \\ S_{F,312} \\ S_{F,321} \\ S_{F,322} \end{pmatrix}$$

teniendo las incógnitas el significado que se les dio en la ecuación (3-1). En general, tendremos un sistema de $n_1 \cdot n_2 \cdot \ldots \cdot n_N$ ecuaciones lineales, que coincidirá con el número de incógnitas (el número de consecuentes de las reglas) por lo que, si el determinante de la matriz resultante no es nulo, la solución es única. Al igual que ocurría en el caso de funciones con una sola entrada, la matriz a resolver es bidimensional y simétrica (ver ecuación (3-16)). Ahora su tamaño es mucho mayor $(n_1 \cdot n_2 \cdot \ldots \cdot n_N) \times (n_1 \cdot n_2 \cdot \ldots \cdot n_N)$ pero en los ordenadores actuales invertir matrices de cientos de filas y columnas se puede realizar en décimas de segundo. Además, como todas las matrices que se obtienen al aplicar el método de mínimos cuadrados, (3-16) es una matriz de tipo covarianza, por lo que se puede resolver de forma muy rápida utilizando el algoritmo de Cholesky [PRE-93].

El resultado aquí obtenido es independiente de la forma y distribución de las funciones de pertenencia que se escojan (funciones triangulares, gaussianas, trapezoidales, etc.) así como de los rangos de definición de cada variable, de modo que la aplicabilidad de estas ecuaciones es enorme. Hay muchas ocasiones [SUD-94] en las que se prefiere usar una equidistribución de funciones de pertenencia a costa de incrementar su número. Para estos casos, el problema ya queda resuelto de forma exacta e inmediata sin más que construir la matriz apropiada e invertirla. Otras veces, se prefiere usar una distribución más específica para cada problema con el fin de sólo tener que usar unas pocas reglas.

Para este último caso se debe realizar una segunda etapa de optimización, la de las funciones de pertenencia en la entrada, que es lo que se abordará en la sección siguiente.

Por último, se ha de comentar que en el caso de funciones de 2 ó más entradas, es frecuente el no disponer de un barrido completo de los datos por todo el rango posible de entrada [SUD-94]. Esto se debe tanto a la enorme cantidad de datos que se necesitarían como a la imposibilidad, a veces, de obtenerlos físicamente. En este caso, el problema presenta cierto grado de indeterminación que se suele traducir en que la matriz resultante presente un determinante nulo, con lo que la solución no sería única y habría que hacer un análisis adicional. Esto se abordará en la sección 3.8.1.

3.4 Optimización conjunta de las premisas y los consecuentes de las reglas

En el apartado anterior se han obtenido los consecuentes óptimos para una configuración predeterminada de funciones de pertenencia. Sin embargo, como veremos en el capítulo siguiente, la distribución de las funciones pertenencia (forma y localización) tiene una gran influencia en la eficiencia de la aproximación [JAN-93], [LOT-94], [BOS-97], [JAN-97], por lo que resulta deseable optimizar dicha distribución de forma conjunta con las reglas. Para poder hacer esto, es necesario definir algún tipo de funciones de pertenencia. Aunque la metodología que se presentará en esta sección puede adaptarse a cualquiera de los tipos de configuración de funciones de pertenencia definidos en el Apéndice A (ver sección 3.8.2), en lo que sigue se utilizará una partición triangular (TP) [RUS-69], [LEE-90], [SUG-91], es decir, particionaremos el espacio de entrada usando funciones de pertenencia triangulares con solapamiento igual a dos (ver Apéndice A). Las ventajas de usar este tipo de topología son numerosas. Por un lado, sólo es necesario almacenar la localización de los centros de cada una de las funciones de pertenencia, ya que los extremos se calculan según los centros de las funciones vecinas. Por otro lado, un dato siempre activará una de las funciones de pertenencia con grado mayor o igual que 0.5, lo cual es muy deseable en campos como el del control. Además, las reglas así obtenidas son fáciles de entender por un operador humano, ya que un dato jamás activará más de dos funciones de pertenencia. Otra ventaja adicional

que presentan es la simplificación de la expresión de la función "defuzzificada". Utilizando la propiedad de que:

$$\sum_{i_m=1}^{n_m} \mu_{X_m^{i_m}}(x_m^k) = 1 \;\; ; \;\; \forall \vec{x}^k \in D \tag{3-19}$$

el denominador de (3-2) es siempre igual a la unidad (ver Apéndice B), por lo que la salida "defuzzificada" quedaría:

$$\tilde{\mathrm{F}}(\vec{x};\mathrm{R},\mathrm{C}) = \sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(R_{i_1 i_2 \ldots i_N} \cdot \prod_{m=1}^{N} \mu_{X_m^{i_m}}(x_m) \right) \tag{3-20}$$

viniendo los grados de pertenencia dados por la expresión A-2. A pesar de su simplicidad, el problema a resolver es no lineal y eso conlleva que pueden existir numerosos mínimos locales donde podemos quedar atrapados, sin que tengan por qué ser buenas soluciones. En la actualidad no existe ningún método matemático que sea capaz de encontrar el mínimo global de una función cualquiera no analítica y no lineal en un tiempo finito.

3.4.1 Análisis previo

Con la distribución seleccionada y el uso del método directo presentado en la sección anterior, el número de parámetros que quedan por optimizar se ha visto sensiblemente reducido. Por ejemplo, si consideramos un sistema difuso con tres variables de entrada y cinco funciones de pertenencia definidas en cada una de ellas, el número total de parámetros a optimizar sería 134 (5^3 consecuentes y 3+3+3 centros[3] de las funciones de pertenencia). Sin embargo, gracias al método presentado en la sección anterior, sólo los 9 centros han de ser optimizados ahora, ya que, una vez fijos éstos, los 125 consecuentes se pueden obtener de forma óptima mediante una simple resolución de un sistema de ecuaciones lineales. Debido a esto, *en esta sección se hablará solamente de optimización de los centros de las funciones de pertenencia viniendo implícito que cada vez que se tenga una nueva configuración de las mismas se obtendrán automáticamente los consecuentes óptimos, utilizando las expresiones de la sección anterior*.

[3] Siempre se harán coincidir los centros de las funciones de pertenencia que se encuentren en los extremos de cada variable con los límites inferior y superior de dicha variable, por lo que dichos parámetros no serán optimizados.

Podríamos intentar abordar el problema inicialmente de forma rudimentaria: si ya tenemos optimizados la mayoría de los parámetros y sólo nos quedan unos pocos, que además están acotados por los rangos de entrada, podríamos intentar hacer un barrido exhaustivo para intentar buscar ese mínimo global. La idea no es del todo descabellada debido a los pocos parámetros que quedan por optimizar y a que cuando se implementa un controlador físicamente, por lo general no se utiliza una gran número de bits para codificar los diversos parámetros. Podríamos dividir el rango de entrada de cada variable en *M* puntos y empezar a barrer todas las combinaciones posibles. Para cada combinación, tendríamos que plantear el sistema (3-18) y resolverlo para obtener las mejores reglas, y después evaluar la función de salida para calcular el *ECMN* alcanzado. Esto, realmente, sería imposible de realizar si tuviéramos que barrer también las reglas. Hagamos una estimación del número de pruebas que tendríamos que hacer: Sea *M* el número de puntos que vamos a barrer de una variable y *p* el número de funciones de pertenencia que debemos colocar, sin contar con los extremos, que son fijos (es decir, en realidad la variable tendría *p+2* funciones de pertenencia). Para una sola variable, tendríamos que mover *p* puntos de forma que sean todos distintos y que no haya otra configuración con los mismos puntos (en el orden que sea). Para calcular el número total de ensayos que se deberían hacer para una sola variable debemos calcular:

$$\sum_{n_1=1}^{M-(p-1)} \sum_{n_2=n_1+1}^{M-(p-2)} \cdots \sum_{n_p=n_{p-1}+1}^{M} 1 \tag{3-21}$$

es decir, el primer centro se puede mover cogiendo todos los puntos desde el 1 hasta el *M-(p-1)* (como deben estar ordenados, si cogiera el valor de alguno de los últimos *p-1* puntos ya no tendríamos puntos suficientes para el resto de centros). El segundo centro se puede mover desde el punto siguiente al que esté el primer centro hasta el punto *M-(p-2)*, y así sucesivamente. De esta forma, aplicando las fórmulas de inducción y buscando el término general, tenemos que el número de pruebas que hay que hacer para barrer una variable con *p+2* funciones de pertenencia con *M* puntos sería:

$$\frac{\prod_{m=1}^{p}(M+1-m)}{p!} = \frac{M\cdot(M-1)(M-2)\cdots(M+1-p)}{p!} \tag{3-22}$$

En general, si tuviéramos que mover p_i funciones de pertenencia de la variable *i* sobre

M_i puntos posibles, el número total de pruebas sería:

$$\prod_{i=1}^{N}\left[\frac{\prod_{m=1}^{p_i}(M_i+1-m)}{p_i!}\right] \tag{3-23}$$

Para hacer una estimación de ese número de pruebas, consideremos 2 variables, 7 funciones de pertenencia por cada variable y un conjunto de 100 puntos posibles por cada variable. En este caso, el número de configuraciones posibles sería (recordemos que los extremos son fijos y sólo deberíamos mover 5 centros por cada variable):

$$\left(\frac{\prod_{m=1}^{5}(100+1-m)}{5!}\right)^2 \approx 10^{15} \tag{3-24}$$

que sería algo inabordable. De modo que aunque hayamos reducido en gran manera el número de parámetros a optimizar, deberemos utilizar un método matemático para optimizar el resto de parámetros[4] y eso es lo que se abordará en la siguiente subsección.

3.4.2 Determinación de un "buen" punto de partida para la optimización de las premisas de las reglas

Así pues, ahora nos disponemos a resolver el problema general de encontrar la matriz de reglas *R* y la matriz de centros *C* que minimicen el error cuadrático *J* dado por la ecuación (3-3). Como se comentó antes brevemente, ahora tenemos el problema de que $\tilde{F} = \tilde{F}(\vec{x}, R, C)$ no depende linealmente de la posición de los centros como ocurría con los consecuentes de las reglas por lo que *J*, como función de las reglas y los centros, puede presentar un número muy grande de extremos relativos (de los cuales nosotros desearíamos encontrar el mínimo global) y el problema es mucho más complicado. Como consecuencia de ello, usar algún método basado en el gradiente directamente nos conduciría al primer mínimo relativo cercano a la configuración inicial sin tener en cuenta que éste sea el mínimo absoluto o no. Por tanto, es necesario insertar una primera

[4] y, según lo dicho anteriormente, estaremos obligados a prescindir del adjetivo "óptimo" cuando nos refiramos a las configuraciones encontradas.

etapa en nuestro algoritmo que sea capaz de acercarnos a ese mínimo global de forma eficiente. Como no existe ningún método matemático capaz de encontrar un mínimo global para una función cualquiera no analítica y no lineal, deberemos contentarnos con encontrar una metodología para acercarnos a una zona donde los mínimos que existan a su alrededor vayan a ser, de forma razonable, "buenos" mínimos. El cometido de esta parte del algoritmo no es, por tanto, encontrar extremos relativos sino proporcionar una configuración de centros que se encuentre en dicha zona, de tal modo que sirva como punto de partida para algún método basado en el gradiente.

La idea que se propone para ello es que debemos buscar los mínimos partiendo de aquella configuración de centros que equidistribuya los errores cuadráticos en cada una de las zonas delimitadas por las funciones de pertenencia. Si en una variable tenemos, por ejemplo, 5 funciones de pertenencia, entonces tendremos delimitadas 4 regiones e intentaremos encontrar aquella distribución que homogeneice los errores cuadráticos en estas regiones. Como cada vez que se mueven los centros de una variable la función obtenida cambia, tendremos en cada iteración que buscar de nuevo las reglas óptimas y calcular el error cuadrático en cada región. De esta forma, al final del proceso obtendremos la posición de los centros de cada una de las funciones de pertenencia de cada variable que haga que esa distribución de errores cuadráticos sea lo más homogénea posible. La principal justificación para utilizar este criterio es que no se puede considerar una buena aproximación aquélla que sea muy precisa en unas zonas del espacio y muy grosera en otras. Una "buena" aproximación será aquélla que aproxime igualmente bien cualquiera de las zonas del espacio donde la función esté definida.

Para que los parámetros que se utilicen en esta parte del algoritmo sean generales y no dependan de factores de escala, en lugar de hallar la contribución al error cuadrático total de cada tramo, hallaremos la contribución de éste al *ECMN* al cuadrado. Es decir, dividiremos el error cuadrático de cada tramo por la varianza de los valores de salida y por el número de datos total[5].

[5] factores que son constantes y cuyo papel es sólo el de normalizar los errores cuadráticos.

De esta forma, podemos asociar al centro c_{v}^{j} una "pendiente" p_{v}^{j} que será la diferencia entre la contribución al error del tramo que le precede y la del tramo que le sigue, teniendo siempre en mente que los extremos se considerarán fijos:

$$p_{v}^{j}=\frac{1}{K\cdot\sigma_{z}^{2}}\left(\sum_{x_{v}^{k}\in\left[c_{v}^{j-1},c_{v}^{j}\right[}\sum_{k\in D}e^{2}(\vec{x}^{k})-\sum_{x_{v}^{k}\in\left[c_{v}^{j},c_{v}^{j+1}\right[}\sum_{k\in D}e^{2}(\vec{x}^{k})\right) \tag{3-25}$$

Así, un valor positivo de esa pendiente indica que la contribución del tramo de la izquierda es mayor que el de la derecha, por lo que el centro debe moverse a la izquierda para contrarrestar este efecto. Como no debemos permitir que el orden de los centros varíe, una posible forma de realizar este movimiento sería:

$$\Delta c_{\mathrm{v}}^{j}=\begin{cases}\dfrac{c_{\mathrm{v}}^{j-1}-c_{\mathrm{v}}^{j}}{b}\dfrac{p_{\mathrm{v}}^{j}}{p_{\mathrm{v}}^{j}+\dfrac{1}{T_{\mathrm{v}}^{j}}} & \text{si } p_{\mathrm{v}}^{j}\geq 0\\ \dfrac{c_{\mathrm{v}}^{j+1}-c_{\mathrm{v}}^{j}}{b}\dfrac{\left|p_{\mathrm{v}}^{j}\right|}{\left|p_{\mathrm{v}}^{j}\right|+\dfrac{1}{T_{\mathrm{v}}^{j}}} & \text{si } p_{\mathrm{v}}^{j}<0\end{cases} \tag{3-26}$$

donde se han introducido dos nuevos parámetros: El "radio de acción" *b* que nos delimita la distancia máxima que se puede recorrer (un valor típico es *b=2* indicando que, como mucho, un centro se puede mover hasta el punto medio entre éste y el contiguo) y la "temperatura" T_{v}^{j} asociada al centro c_{v}^{j} que nos indica cuánto vamos a mover el centro dentro del rango de acción posible. Así, para temperaturas muy altas los centros se van a mover, en general, grandes distancias mientras que para temperaturas bajas, estos movimientos serán muy pequeños, siempre dependiendo del valor de las pendientes.

Se comenzará con temperaturas altas para todos los centros (100-1000 típicamente[6]) aunque esto no es muy crítico ya que en la evolución del algoritmo se irán ajustando adaptativamente las temperaturas de los mismos. De esta forma, independientemente de los valores iniciales de las pendientes, los centros se irán moviendo inicialmente

[6] En el capítulo siguiente se justificará esta elección.

grandes distancias aumentándose las temperaturas de los que se muevan en un cierto número de iteraciones siempre en la misma dirección y reduciéndose cada vez que cambien de sentido, mejorándose así la velocidad de convergencia del proceso, que finalizará cuando todas las temperaturas o todas las pendientes sean lo suficientemente bajas como para despreciar el movimiento de los centros. En ese momento, la superficie del error cuadrático estará lo más homogéneamente distribuida que sea posible.

3.4.3 Optimización propiamente dicha de las premisas de las reglas

Para encontrar el mínimo local más cercano a la configuración que nos proporciona la etapa anterior necesitamos calcular cómo varía el error cuadrático *J* de la expresión (3-3) con la posición de cada uno de los centros de las funciones de pertenencia. Para ello, debemos computar:

$$\frac{\partial \mathrm{J(C)}}{\partial c_{\mathrm{v}}^{j}} = -2 \cdot \sum_{k \in D} \left[\left(F(\vec{x}^{\,k}) - \widetilde{F}(\vec{x}^{\,k}; R, C) \right) \cdot \left(\frac{\partial \widetilde{\mathrm{F}}(\vec{\mathrm{x}}^{\,\mathrm{k}}; \mathrm{R}, \mathrm{C})}{\partial c_{\mathrm{v}}^{j}} \right) \right] \tag{3-27}$$

Haciendo uso de la ecuación (3-20), válida sólo para la configuración aquí definida[7] (partición triangular), tenemos:

$$\frac{\partial \widetilde{\mathrm{F}}(\vec{\mathrm{x}}^{\,\mathrm{k}}; \mathrm{C})}{\partial c_{\mathrm{v}}^{j}} = \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_N=1}^{n_N} \left(R_{i_1 i_2 \ldots i_N} \cdot \frac{\partial \prod_{m=1}^{N} \mu_{X_m^{i_m}}(x_m^k)}{\partial c_{\mathrm{v}}^{j}} \right) \tag{3-28}$$

y, como sólo las funciones de pertenencia de la variable *v* pueden depender del centro c_{v}^{j} tendremos:

$$\frac{\partial \prod_{m=1}^{N} \mu_{X_m^{i_m}}(x_m^k)}{\partial c_{\mathrm{v}}^{j}} = \frac{\partial \mu_{X_{\mathrm{v}}^{i_{\mathrm{v}}}}(x_{\mathrm{v}}^k)}{\partial c_{\mathrm{v}}^{j}} \cdot \prod_{\substack{m=1 \\ m \neq \mathrm{v}}}^{N} \mu_{X_m^{i_m}}(x_m^k) \tag{3-29}$$

Finalmente, a partir de la expresión A-2:

[7] En la sección 3.8.2 se adaptará el método a otros tipos de funciones de pertenencia.

$$\frac{\partial \mu_{X_v^{i_v}}(x_v^k)}{\partial c_v^j} = \begin{cases} \dfrac{x_v^k - c_v^{j-1}}{\left(c_v^j - c_v^{j-1}\right)^2} U(x_v^k; c_v^{j-1}, c_v^j) & \text{si } i_v = j-1 \\ \dfrac{-\left(x_v^k - c_v^{j-1}\right)}{\left(c_v^j - c_v^{j-1}\right)^2} U(x_v^k; c_v^{j-1}, c_v^j) + \dfrac{c_v^{j+1} - x_v^k}{\left(c_v^{j+1} - c_v^j\right)^2} U(x_v^k; c_v^j, c_v^{j+1}) & \text{si } i_v = j \\ \dfrac{-\left(c_v^{j+1} - x_v^k\right)}{\left(c_v^{j+1} - c_v^j\right)^2} U(x_v^k; c_v^j, c_v^{j+1}) & \text{si } i_v = j+1 \end{cases} \quad (3\text{-}30)$$

cumpliéndose que, por lo general, la derivada no está definida en los puntos donde los datos coincidan con la posición de los centros. En estos puntos habría que usar como valor de la derivada el valor de una de las dos derivadas laterales o la semisuma de éstas. En cualquier caso, debido al carácter discreto de nuestros datos, esto no es un gran problema ya que los centros se pueden mover en un continuo (la precisión de los números en punto flotante de la computadora) y es improbable que uno de los datos coincida exactamente con un centro. En cualquier caso, la derivada total es una suma para todos los datos y el posible efecto de esta discontinuidad en la derivada queda minimizado.

Una vez que tenemos las expresiones necesarias para hallar la derivada del error cuadrático medio con respecto a cada uno de los centros, ya estamos en condiciones de abordar la búsqueda del mínimo relativo más cercano a la configuración que equidistribuye, en la medida de lo posible, los errores cuadráticos. Como ya se ha comentado anteriormente, es de esperar que ese primer mínimo proporcione una solución comparable a la que proporcionaría el “desconocido” mínimo global.

3.4.3.1 Descenso en gradiente

Uno de los métodos más utilizados para encontrar extremos locales es el descenso en gradiente. Para ello, lo único que tenemos que hacer es calcular el gradiente del error cuadrático medio con respecto al vector compuesto por todos los centros de las funciones de pertenencia que no estén en los extremos y moverlos justo en la dirección opuesta según la expresión:

$$\Delta C = -\eta \cdot \nabla_C \mathrm{J} \quad (3\text{-}31)$$

El factor de aprendizaje η se irá adaptando dinámicamente según la evolución del proceso de búsqueda de tal forma que, si en una cierta iteración el error aumenta o el orden de las funciones de pertenencia cambia, volveremos al estado anterior y reduciremos dicho factor. Igualmente, si esto no se produce, su valor se incrementará suavemente (10% de su valor). El valor de η vendrá dado por:

$$\eta = \frac{1}{máx(1, r_y)} \frac{\eta_0}{1 + \sum_{k \in D} \left\| \nabla_\Theta \hat{F}(\hat{x}(k); \Theta) \right\|^2} \tag{3-32}$$

siendo r_y el rango de los datos de salida. De esta forma, con un valor inicial de $\eta_0 = 1.0$ (ver sección 5.5.1.1) nos aseguramos que ya en el primer paso se mejore el error. En cualquier caso, debido a la adaptación dinámica de η, el valor inicial de η_0 no es esencial y, tras un cierto número de iteraciones, encontraremos el mínimo local.

Conforme vamos moviendo los centros debemos ir, como siempre, calculando las reglas óptimas para éstos, utilizando el método de la sección 3.3. Como es habitual, cuando nos encontremos ya muy cerca del mínimo local, el valor del gradiente es muy pequeño y el proceso se suele ralentizar. Gracias a la adaptación del factor de aprendizaje, esta ralentización no ocurre.

El método del descenso en gradiente funciona bien en general pero no nos garantiza que los centros no vayan a cambiar su ordenación. Es por ello que una de las condiciones para disminuir el factor de aprendizaje es que dicha ordenación se pierda. A continuación vamos a presentar un método alternativo, aparentemente más adecuado, muy parecido al descrito en la sección 3.4.2 pero substituyendo el papel de la pendiente p_v^j por la derivada del error cuadrático con respecto al centro c_v^j. De esta forma, vamos moviendo cada centro de forma independiente en la dirección en la que se reduzca el error cuadrático, sin el peligro de que el orden de los centros se altere:

$$\Delta c_v^j = -\frac{\eta_v^j}{K \cdot \sigma_z^2} \frac{\frac{\partial J(R,C)}{\partial c_v^j}}{\left| \frac{\partial J(R,C)}{\partial c_v^j} \right| + \frac{1}{T_v^j}} \quad \text{con} \quad \eta_v^j = \begin{cases} \frac{c_v^j - c_v^{j-1}}{b} & \text{si } \frac{\partial J(R,C)}{\partial c_v^j} \geq 0 \\ \frac{c_v^{j+1} - c_v^j}{b} & \text{si } \frac{\partial J(R,C)}{\partial c_v^j} < 0 \end{cases} \tag{3-33}$$

De forma gráfica, los movimientos de los centros están controlados por una función de saturación como la que se representa en la figura 3.1. En la zona lineal se emplea el descenso en gradiente usual, sin limitaciones. Para movimientos grandes, las regiones de saturación se encargan de que un centro no se desplace más allá de una fracción *b* de la distancia que lo separa de su vecino. Las temperaturas T_v^j se adaptan progresivamente para acelerar la búsqueda, de la misma forma que se hacía en la sección anterior. Como ahora pretendemos ajustar los centros para hallar el mínimo local, el parámetro *b* deberá ser algo mayor y las temperaturas iniciales menores para no dar un salto excesivamente grande que nos saque del mínimo que queremos encontrar. Por ejemplo, se puede usar un radio de acción entre 10 y 20 y unas temperaturas iniciales entre 0.1 y 1.

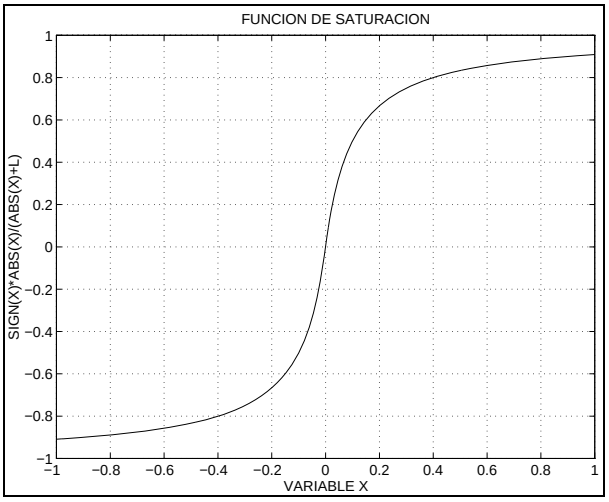


Fig. 3.1 Función de saturación para el control de los movimientos de las funciones de pertenencia.

Utilizando este método, el centro de cada función de pertenencia se va moviendo de forma independiente en la dirección en la que se reduce el error cuadrático, sin la preocupación de que se pueda producir una alteración en el orden de las funciones de pertenencia, conservándose así la interpretación lingüística del sistema obtenido. Cuando todas las parciales sean nulas obtendremos un mínimo local.

En este punto del algoritmo, hemos encontrado finalmente una configuración pseudo-óptima de funciones de pertenencia y consecuentes de las reglas para aproximar nuestros datos de E/S, partiendo de un número fijo de funciones de pertenencia (y, por

tanto, de reglas). En el siguiente apartado se analizará el problema de cuántas funciones de pertenencia se deben asignar a cada variable y, por extensión, qué variables son necesarias para un cierto grado de aproximación dado.

3.5 Búsqueda de una nueva topología

En este punto del algoritmo, acabamos de optimizar la posición de las funciones de pertenencia en la entrada y los consecuentes de las reglas que éstas definen obteniéndose, de esta forma, un *ECMN* dado. El problema fundamental que ahora debemos abordar es: si tuviéramos que colocar una nueva función de pertenencia, en qué variable de entrada colocarla y dónde.

Hay autores [MIT-93], [NOZ-93], [HIG-94] que sitúan funciones de pertenencia en todas las variables de entrada y las colocan de tal forma que su intersección se encuentre en la zona de mayor error. Hay dos graves desventajas que este método entraña:

- Aumentando el número de funciones de pertenencia en todas las variables provoca un aumento excesivo del número de reglas debido a su dependencia exponencial con dicho número de funciones de pertenencia.

- Situar las funciones para minimizar justamente el punto con mayor error es un método críticamente sensible al ruido. Como se indica en [BER-97], ese método le da más importancia a los datos más ruidosos (outliers) que a la verdadera forma de la función.

Como consecuencia de esto, se pone de manifiesto que se debe aumentar la complejidad del sistema de la forma más precisa posible para que no se dispare el número de parámetros a optimizar ni el número de reglas y que se debe realizar un análisis más detallado sobre toda la superficie del error y no sólo fijarse en los puntos donde éste sea mayor.

Otros autores [TAK-85], [SUG-88a], [SUG-93], [BAR-98] sin embargo, utilizan una especie de árbol combinatorial de tal forma que lo que hacen esencialmente es analizar

todas las posibilidades de forma exhaustiva y sistemática, aplicando el algoritmo completo a cada una de éstas, decantándose finalmente por la que mejores resultados produzca. Lo ideal sería, como es obvio, no tener que recurrir a este método.

3.5.1 Determinación de la(s) variable(s) donde se debe añadir una nueva función de pertenencia

La forma de acometer el problema de determinar en qué variable o variables de entrada deberíamos aumentar el número de funciones de pertenencia se debe realizar a partir de un análisis de la superficie de error a la que se ha llegado en la etapa anterior. Posiblemente, la mejor manera de presentar la metodología que aquí se propone es a través de un ejemplo.

Consideremos la siguiente función [ROV-96]:

$$y_5(x_1, x_2) = \frac{1}{2}\left[1 + sin(2\pi x_1)\cos(2\pi x_2)\right] \tag{3-34}$$

definida en el intervalo [0,1]x[0,1] (ver figura 3.2a). En la figura 3.2b se representa la superficie de salida obtenida para una configuración fija de 3 funciones de pertenencia por cada variable. Si analizamos visualmente con cierto detalle la gráfica de la función original, resulta evidente que sería preferible pasar a la configuración 4x3 antes que a la 3x4, ya que teniendo 4 funciones de pertenencia en la variable x_1 y 3 en la x_2 seremos capaces de localizar, aunque sea de forma aproximada, los picos de la función y de esa forma, el perfil principal de la misma. Veamos cómo podemos llegar a esa conclusión a través del análisis de la superficie del error[8]. En la figura 3.3 se representa la superficie resultante de restar la función original de la aproximada para la configuración 3x3. Los centros optimizados se encuentran en el punto 0.17 para x_1 y en 0.5 para x_2.

[8] Evidentemente, difícilmente podremos llegar a ninguna conclusión a partir del análisis individual de la función original (sin saber la aproximada) o de la aproximada (sin conocer la original).

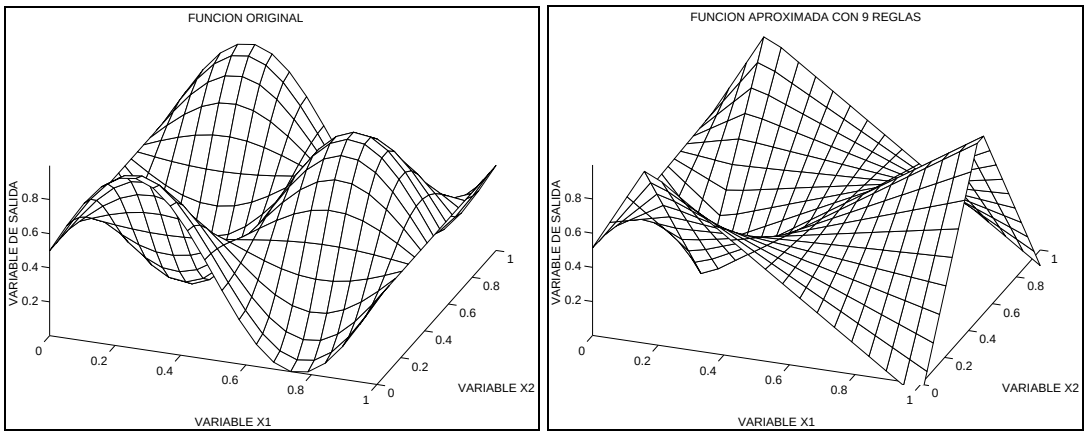


Fig. 3.2 a) Función y5(x1,x2) original. b) Función aproximada con 9 reglas (3x3).

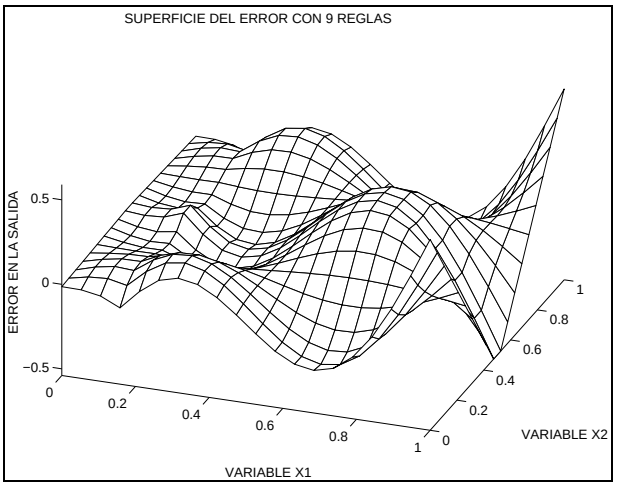


Fig. 3.3 Superficie del error para la configuración 3x3.

Analicemos inicialmente la variable x_2 (centros en 0, 0.5 y 1). Para ello, vamos a dividir la variable x_1 en intervalos infinitesimales de anchura dx_1 y consideraremos individualmente cada uno de los intervalos de la forma $[x_1,x_1+dx_1]$ dejando la variable x_2 libre. Para analizar si la variable x_2 necesita una nueva función de pertenencia veamos primero qué grado de responsabilidad tienen las funciones de pertenencia que haya definidas en esta variable respecto al error existente en la actualidad. Si nos fijamos, por ejemplo, en el último tramo en el que se ha dividido x_1 *[1-dx$_1$,1]*, comprobamos que el error en ese tramo es considerable. Debido a que la variable x_2 consta actualmente de 3 funciones de pertenencia, podemos subdividir el tramo anterior en dos subtramos, uno que va en $x_2 \in [0,0.5]$ y otro que va en [0.5,1]).

Si no existiera más función que la existente en el tramo $[1-dx_1,1]x[0,1]$, lo que tendríamos realmente sería una función de una dimensión que debe ser aproximada mediante 3 funciones de pertenencia (las de x_2) situadas en 0, 0.5 y 1.0. En el caso de utilizar una partición triangular, sabemos que en una dimensión las funciones de salida son funciones rectas a tramos y justamente sería fácil de imaginar cómo se puede aproximar la función resultante de forma prácticamente exacta con esas 3 funciones de pertenencia. Es decir, a pesar del gran error existente, por lo que al intervalo $[1-dx_1,1]x[0,1]$ se refiere, no se debería añadir una nueva función de pertenencia en la variable x_2 ya que con la configuración actual se podría aproximar la función del error en dicho intervalo de forma casi exacta y no tendría sentido añadir una nueva función de pertenencia en x_2 por ese motivo. Además, de tener que añadirla habría que hacerlo justamente en el sitio donde ya hay una función de pertenencia. Como conclusión, el hecho por el que en dicho intervalo exista un gran error no se debe al número insuficiente de funciones de pertenencia en la variable x_2 sino en el resto de variables (en este caso, por exclusión, la variable x_1).

Esta misma operación se debe hacer con cada uno de los intervalos infinitesimales en los que se ha dividido x_1, computándose, en cada caso, el error obtenido de aproximar solamente ese intervalo con esas 3 funciones y sumándolo para obtener un índice final cuyo valor dará una idea de la necesidad de añadir o no una nueva función en la variable x_2. Esencialmente, este índice no es más que el error total de aproximación que se conseguiría si fijáramos esas 3 funciones de pertenencia en la variable x_2, y permitiéramos tener infinitas en la variable x_1, es decir, cuál sería el menor error que podríamos obtener si no aumentáramos el número de funciones de pertenencia de esa variable[9].

Finalmente, realizando los mismos pasos pero fijando ahora la variable x_2, obtenemos el índice equivalente para la variable x_1. Comparando ambos índices

[9] partiendo de la hipótesis de que apenas variará la disposión actual de funciones de pertenencia si no se inserta ninguna nueva.

podemos determinar en qué variable conviene más modificar la configuración de funciones de pertenencia (en aquélla cuyo error asociado sea mayor).

Pasemos ahora a considerar cómo hacer esto matemáticamente. Inicialmente plantearemos la forma general en el caso continuo.

3.5.1.1 Caso continuo

En este caso, debemos computar la suma de errores al cuadrado para cada uno de los tramos en que hemos dividido infinitesimalmente cada variable. Si estamos analizando la variable *v*, el tramo $\left(x_1,...,x_{v-1},\left[c_v^1,c_v^{n_v}\right],x_{v+1}...,x_N\right)$ vendrá dado por:

$$\left[x_1,x_1+dx_1\right]\times\cdots\times\left[x_{v-1},x_{v-1}+dx_{v-1}\right]\times\left[c_v^1,c_v^{n_v}\right]\times\left[x_{v+1},x_{v+1}+dx_{v+1}\right]\times\cdots\times\left[x_N,x_N+dx_N\right] \quad (3\text{-}35)$$

e intentamos aproximar los puntos de la función error que caigan en ese tramo como si fuera una función de una dimensión utilizando la configuración fija de funciones de pertenencia de la variable *v*. Por ser dicha configuración fija, esto lo podremos hacer de forma directa utilizando el procedimiento presentado en la sección 3.3. De esta forma obtendremos una función unidimensional

$$F^v_{x_1,...,x_{v-1},x_{v+1}...,x_N}(x_v) \quad (3\text{-}36)$$

que aproximaría de la mejor forma posible el tramo correspondiente de la hiper-superficie del error, basándose solamente en las funciones de pertenencia de la variable *v* sin influencia de las definidas en el resto de variables. Es decir, que si esa función aproxima perfectamente el tramo correspondiente de la superficie del error utilizando las funciones ya existentes en la variable *v* querrá decir que las funciones de pertenencia de dicha variable no son las responsables del error en dicho tramo (ya que si las demás variables tuvieran funciones suficientes se podría conseguir un error cero). Igualmente, si dicho valor es alto querrá decir que hay una gran responsabilidad de la variable *v* en el error de dicho tramo.

Computando los errores para cada uno de los posibles tramos de la forma (3-35) tendremos una medida global de la suma de los errores al cuadrado mínima que se podría obtener utilizando la configuración actual de funciones de pertenencia de la variable *v*:

$$J_v = \int_{X_1} dx_1 \cdots \int_{X_{v-1}} dx_{v-1} \int_{X_{v+1}} dx_{v+1} \cdots \int_{X_N} dx_N \left[\int_{X_v} dx_v \left(e(\vec{x}) - F^v_{x_1,\ldots,x_{v-1},x_{v+1}\ldots,x_N}(x_v) \right)^2 \right] \qquad (3\text{-}37)$$

donde $e(\vec{x})$ es la superficie del error, que es la función que queremos aproximar ahora.

Finalmente, se computa este valor para cada una de las variables para poder comparar entre ellas. Evidentemente aquella variable con mayor índice será la candidata a llevarse la nueva función de pertenencia.

3.5.1.2 Caso discreto

Como es obvio, el método presentado en el subapartado anterior es impracticable y sólo tiene como fin expresar de forma teórica el caso ideal. Como siempre vamos a partir de un conjunto discreto de vectores de E/S, deberemos plantear una simplificación de (3-37) para este caso. Para ello, los intervalos de anchura infinitesimal dx_i pasarían a tener una anchura discreta Δx_i, por lo que ahora los tramos donde se realiza cada aproximación son de la forma:

$$[x_1, x_1 + \Delta x_1] \times \cdots \times [x_{v-1}, x_{v-1} + \Delta x_{v-1}] \times \left[c_v^1, c_v^{n_v}\right] \times [x_{v+1}, x_{v+1} + \Delta x_{v+1}] \times \cdots \times [x_N, x_N + \Delta x_N] \qquad (3\text{-}38)$$

y ya no se pueden considerar tramos unidimensionales sino N-dimensionales. Cada uno de esos tramos se aproximará con una configuración de dos funciones de pertenencia en todas las variables menos en la variable *v* donde se usarán las funciones de pertenencia que tenga definidas. Cuanto menor sea la anchura del tramo mayor será la similitud con el caso continuo y más se parecerá el tramo a una función unidimensional. La suma de los errores cuadráticos asociados a cada variable tiene ahora la misma forma que en (3-37) pero intercambiando el signo integral por sumas y los diferenciales por diferencias:

$$J_v = \sum_{\Delta x_1} \cdots \sum_{\Delta x_{v-1}} \sum_{\Delta x_{v+1}} \cdots \sum_{\Delta x_N} \left(\sum_{\substack{\vec{x}^k / x_i^k \in \Delta x_i \\ i \neq v}} \left[e(\vec{x}) - F^v_{\Delta \vec{x}}(\vec{x}) \right]^2 \right) \qquad (3\text{-}39)$$

En la expresión anterior se ha recalcado que ahora la función *F* aproxima tramos completos N-dimensionales y que tiene una forma distinta para cada combinación de tramos de todas las variables salvo la que se esté analizando (*v*).

Ahora bien, esa función $F^{v}_{\Delta\vec{x}}(\vec{x})$ con la que aproximamos la superficie del error cometida se podría desglosar aún más como la unión de funciones aproximadas $F^{v}_{\Delta\vec{x},i}(\vec{x})$ para cada intervalo de la forma:

$$[x_1, x_1+\Delta x_1]\times\cdots\times[x_{v-1}, x_{v-1}+\Delta x_{v-1}]\times[c_v^i, c_v^{i+1}]\times[x_{v+1}, x_{v+1}+dx_{v+1}]\times\cdots\times[x_N, x_N+\Delta x_N] \quad (3\text{-}40)$$

de modo que sería como aproximar la superficie del error a partir de la aproximación individual en cada uno de los tramos (3-40), dedicando a cada uno de esos tramos dos funciones de pertenencia por cada variable (ahora también para la variable *v*). Y eso es prácticamente[10] lo mismo que considerar la función de la superficie del error como función a aproximar y definir una configuración fija de funciones de pertenencia compuesta por un alto número de ellas en todas las variables menos *v* y poner exactamente la misma configuración que tenemos de la etapa anterior en dicha variable. Y justamente, con la metodología presentada en la sección 3.3 podemos abordar este problema de forma eficiente en un solo paso sin más que resolver el sistema de ecuaciones lineales (3-18).

Generalmente, la amplitud de los intervalos en el resto de variables se escoge según el número de datos con los que contemos. Así, en el caso de funciones de dos variables muestreadas con 400 datos no sería lógico definir más de 10 funciones de pertenencia por variable ya que en ese caso tendríamos una media de 4 datos por cada sub-tramo (en el caso de tener también 10 funciones en la otra variable) lo cual sería más que suficiente como para considerar el equivalente a un número "muy grande" de funciones de pertenencia[11]. En general, en la metodología presentada, elegiremos el número de funciones de pertenencia con las que particionar el resto de variables de la forma:

[10] La diferencia estriba en que una misma regla sólo se ocupa de un sub-tramo en el primer caso, mientras que en el segundo debe también velar por los subtramos vecinos. Si la partición del resto de variables es suficientemente grande, este efecto es despreciable.

[11] Es muy probable que existan sub-tramos donde no existan datos de E/S por lo que pueden existir indeterminaciones en el cálculo de las reglas. Esto realmente no es problema como se verá en la sección 3.8.1.

$$N_{\Delta} = \frac{\sqrt[N]{K}}{2} \tag{3-41}$$

donde, recordemos, *K* es el número de datos y *N* el número de variables.

De esta forma, cada una de las matrices cuadradas (3-16) tendrán:

$$N_{\Delta}^{N-1} \cdot n_v = \frac{n_v}{2^{N-1}} K^{\frac{N-1}{N}} \tag{3-42}$$

filas y columnas. Para funciones de muchas dimensiones, es posible que haya que reducir el tamaño de la matriz por ser éste demasiado grande pero en cuanto a tiempo de computación para el cálculo de cada elemento de la matriz, el término más significativo (en esta parte y en todo el algoritmo) es el número de datos. Sin embargo hay que matizar que este cálculo para determinar dónde insertar nuevas funciones de pertenencia se realiza unas pocas veces durante el algoritmo, por lo que la contribución de esta etapa al tiempo total de ejecución es despreciable y queda fuera de toda duda la importancia de una buena elección en esta fase del proceso.

Finalmente, una vez obtenido el error J_v asociado a cada variable *v*, para acelerar el proceso podemos determinar qué variables son las que se seleccionarán especificando un tanto por ciento respecto al valor mayor dentro del cuál otras variables también serán seleccionadas, es decir, si llamamos:

$$J_{max} = \max_{v}(J_v) \tag{3-43}$$

el conjunto de variables a las que se les incrementará el número de funciones de pertenencia vendrá dado por:

$$\{x_m / J_m \geq \rho J_{max}, m = 1...N\} \tag{3-44}$$

siendo ρ un número entre 0 y 1. Generalmente utilizaremos ρ = 0.2, indicando que incluiremos también a aquéllas variables cuyo error asociado no difiera más del 20% del valor del error máximo.

La ventaja de este método es que su validez también abarca el caso en que en alguna de las variables de entrada sólo haya definida una función de pertenencia, es decir, que no esté siendo considerada por el sistema difuso actual. Si, tras el proceso, resulta que se le

debe asignar una nueva función de pertenencia a dicha variable, dicha variable entrará en juego en el proceso de inferencia difusa. Hay que tener en cuenta que conforme aumenta la complejidad del sistema, también aumentará su grado de aproximación y es posible, llegados a un cierto nivel de precisión, que una cierta variable comience a considerarse necesaria.

En [SUG-93] se aborda el problema de la eliminación de variables que no repercutan significativamente en la salida. Con el método aquí presentado, dichas variables con poca influencia sobre la variable de salida simplemente no incrementarán el número de sus funciones de pertenencia por lo que nos evitamos tener que recurrir a este proceso y, por lo tanto, a tener que considerar otras configuraciones más complejas cuando éstas no son necesarias.

3.5.2 Localización de la nueva función de pertenencia

En esta sección intentaremos resolver el problema de dónde colocar el nuevo centro. Supongamos que debemos añadir una nueva función de pertenencia a la variable *i*. Intuitivamente, si hay una zona donde el error sea muy grande para un punto de dicha variable, debemos poner una nueva función de pertenencia en dicha región para de esta forma intentar compensar el error pero, como ya se ha comentado anteriormente, esto presenta el problema de que la posible existencia de ruido en los datos pueda provocarnos el situar el nuevo centro en una zona poco adecuada. Por lo tanto, para la colocación del nuevo centro, tendremos en cuenta la distribución completa del error, de manera que tenderemos a situar la nueva función en el centro de gravedad de la distribución de los errores cuadráticos medios a lo largo de la variable.

De esta forma, descompondremos el rango de dicha variable en pequeños tramos diferenciales y calcularemos el error cuadrático medio en cada tramo[12], obteniendo la siguiente estimación de la localización del nuevo centro:

[12] error cuadrático medio referido a la función original, aunque es el mismo que el obtenido si lo hiciéramos respecto a la superficie del error por estar las funciones de pertenencia en la misma posición.

$$c_i^* = \frac{\int_{X_i} x_i \, \overline{e^2(x_i)}\, dx_i}{\int_{X_i} \overline{e^2(x_i)}\, dx_i} \tag{3-45}$$

siendo

$$\overline{e^2(x_i)} = \frac{\int ... \int e^2(x_1, x_2, .., x_i, .., x_n) dx_1 dx_2, ..., dx_{i-1} dx_{i+1}, ..., dx_n}{\int ... \int dx_1 dx_2, ..., dx_{i-1} dx_{i+1}, ..., dx_n} \tag{3-46}$$

Al igual que se hizo en la sección anterior, traducimos estas expresiones a lenguaje discreto. Para ello, dividimos el espacio de la variable *i* en pequeños tramos de anchura Δx_i. Cuanto más pequeños sean éstos mayor precisión obtendremos.

$$c_i^* = \frac{\sum_{\Delta x_i} x_i^* \, \overline{e^2(\Delta x_i)}}{\sum_{\Delta x_i} \overline{e^2(\Delta x_i)}} = \sum_{\Delta x_i} x_i^* \, \overline{e_N^2(\Delta x_i)} \tag{3-47}$$

donde x_i^* es el valor central de cada uno de los intervalos que se va barriendo y $\overline{e_N^2(\Delta x_i)}$ es el error cuadrático medio normalizado de todos los datos cuya componente *i* cae en el rango determinado por Δx_i (que representa un índice en la sumatoria). La expresión del error cuadrático medio sería:

$$\overline{e^2(\Delta x_i)} = \frac{\sum_{\substack{x_j ; j \neq i \\ x_i \in \Delta x_i}} e^2(x_1, x_2, .., x_i, .., x_n)}{\sum_{\substack{x_j ; j \neq i \\ x_i \in \Delta x_i}} 1} \tag{3-48}$$

donde el denominador es simplemente el número de datos cuya componente en la variable *i* cae dentro de Δx_i[13]. Sin embargo, aplicar esta fórmula no es suficiente para tener una buena estimación de dónde colocar el nuevo centro ya que debemos comprobar que realmente el valor dado por (3-47) sea suficientemente representativo. En otras palabras, es posible que la nueva función de pertenencia caiga en una zona donde realmente no hay error debido a que en la distribución de errores cuadráticos medios haya valores grandes a ambos lados de dicha zona pero no justo en ella (es el

[13] por estética se ha prescindido del superíndice *k* indicatorio de que estamos usando los datos de E/S de que disponemos.

caso de distribuciones simétricas, por ejemplo). En tales casos, lo que haremos será equidistribuir todos los centros de dicha variable para que el algoritmo se encargue de irlos colocando en sus lugares correctos. Lo que vamos a hacer pues, es calcular un índice que represente la dispersión de la distribución de los errores cuadráticos medios y dejar que sea un valor umbral el que realice la decisión. Para ello, vamos a calcular dicha dispersión como una medida de la distancia media al valor estimado según la distribución de los errores cuadráticos medios normalizados:

$$\bar{d} = \sum_{\Delta x_i} \overline{e_N^2(\Delta x_i)} \cdot \left| x_i^* - c_i^* \right| \tag{3-49}$$

Los valores límite corresponderían a 0 (cuando todo el error estuviera concentrado en el punto decidido) y al rango de la variable dividido por 2 (cuando todo el error estuviera repartido de forma igual en los bordes de la distribución y, por tanto, el punto decidido fuera el central). Así pues, cuando dicha distancia media sea inferior a un tanto por ciento del rango de entrada de la variable (por ejemplo 5%), el algoritmo colocará la nueva función en el punto calculado. Cuando sobrepase el umbral, las funciones de pertenencia se redistribuirán de forma homogénea por todo el rango de la variable.

3.6 Evaluación de las configuraciones obtenidas por el algoritmo

A lo largo del algoritmo se van obteniendo una serie de sistemas difusos con distintas complejidades donde, para una complejidad dada, se ha usado como criterio minimizador la suma de los errores al cuadrado entre los datos reales y los aproximados. Una vez que se haya alcanzado el número máximo de funciones de pertenencia preestablecido inicialmente o se haya conseguido un error de aproximación por debajo de un cierto límite, el algoritmo llega a su fin y debemos, en este punto, evaluar todas las configuraciones obtenidas en su conjunto para poder quedarnos con la que más convenga según las especificaciones del problema. Si intentamos que los datos sean aproximados de la forma más precisa posible, evidentemente elegiremos aquella configuración que haya conseguido un ECMN menor, y ésta será, por lo general, la que mayor número de parámetros utilice. Si, por el contrario, queremos que el sistema sea lo más simple posible, tenderemos a buscar en configuraciones sencillas con un ECMN posiblemente alto.

En general, el mejor modelo será aquél que consiga el menor error utilizando la menor complejidad estructural (el menor número de parámetros). Por tanto, debe existir un compromiso entre complejidad y precisión a la hora de decantarse por una configuración u otra.

Una primera solución a este problema se puede encontrar usando la "longitud de descripción mínima" [RIS-78]. La longitud de descripción de un modelo se define como la suma de la cantidad de datos necesarios para describirlo y la cantidad necesaria para describir el error entre el modelo y la realidad. Como es obvio, si las muestras utilizadas presentan ruido, el intentar aproximarlas de forma exacta no es la mejor solución ya que, a partir de una cierta complejidad del sistema, dejaríamos de aproximar el modelo subyacente a estos datos sino el propio ruido de éstos. Esta teoría, que usa criterios basados en medidas de la entropía, ha sido aplicado a varios problemas como el ajuste de modelos, la segmentación de imágenes y algoritmos de agrupamiento. Sin embargo, presenta el inconveniente de que, en la mayoría de los casos, se desconoce el error existente entre los datos disponibles y los reales.

El problema de seleccionar de entre diferentes sistemas difusos el que presente los mejores resultados teniendo en consideración el objetivo conjunto de la precisión y la simplicidad también surge en problemas de optimización de sistemas usando algoritmos genéticos [JAG-99]. Un gran número de parámetros en el modelo puede provocar un sobre-ajuste en los datos de entrenamiento con el consecuente deterioro de las propiedades de generalización. Por ejemplo, en [ISH-96] la función de ajuste se obtiene mediante una combinación lineal del error cometido por el sistema y el número de parámetros que lo define.

$$Grado\ de\ ajuste = W_E \cdot Error + W_C \cdot Complejidad \tag{3-50}$$

En dicho método, los pesos asociados al error del sistema y a su complejidad son fijos, por lo que su elección tiene un efecto significativo en la solución final obtenida por el algoritmo genético. Debido a que la importancia de cada objetivo depende principalmente en la preferencia del usuario final de la aplicación, no es nada fácil asignar tales constantes. En el algoritmo propuesto en este capítulo, esto no entraña un

gran problema ya que se van obteniendo las mejores configuraciones según el grado de complejidad y pueden ser presentadas todas ellas al usuario final para que éste tome la decisión. Incluso puede haber distintos usuarios finales que quieran resolver el mismo problema pero con especificaciones distintas. Con el método propuesto en [ISH-96], sin embargo, la elección se debe hacer antes de la ejecución algoritmo, teniéndose éste que ejecutar de nuevo en el caso de un cambio de especificaciones.

Otro método existente para la evaluación del sistema es el que se propuso en [SUG-93]. Para ello, se divide el conjunto de datos de E/S en dos grupos (A y B) y se intenta minimizar un índice de regularidad *RC ("regularity criterion")* cuya finalidad es evitar el sobre-ajuste de los datos. Éste índice viene definido de la forma:

$$RC = \frac{1}{k_A}\sum_{i=1}^{k_A}(y_i^A - y_i^{AB})^2 + \frac{1}{k_B}\sum_{i=1}^{k_B}(y_i^B - y_i^{BA})^2 \qquad (3\text{-}51)$$

donde k_A y k_B es el número de datos en los conjuntos A y B, respectivamente, y_i^A e y_i^B son las salidas deseadas de los conjuntos A y B, mientras que y_i^{AB} e y_i^{BA} son las salidas del modelo para el grupo A una vez entrenado con el conjunto B (y viceversa).

De esta forma, en [SUG-93] se va incrementando de forma combinatorial la complejidad del sistema difuso y evaluando el *RC*. En el momento en que este índice sea mayor para una estructura más compleja que otra, el algoritmo se detiene y elige la configuración anterior como la óptima. Este método presenta dos inconvenientes. El primero de ellos es la elección de ambos conjuntos de datos (cuáles van a un grupo y cuáles a otro). El segundo es mucho más sutil. Como el problema a resolver no es lineal, nunca tenemos la certeza de encontrar el mínimo global para un configuración dada. Se puede dar el caso (y de hecho se da) de que, debido a ciertas simetrías en la función a aproximar, el grado de aproximación vaya dando saltos según si se escoge un número par o impar de funciones de pertenencia. Con el método usado por Sugeno el algoritmo termina repentinamente sin considerar la posibilidad de que la siguiente configuración a aquélla donde se cumple la condición de parada pueda tener un índice *RC* mejor.

Debido a que el determinar qué configuración escoger de entre las que se ha obtenido por el algoritmo es un claro ejemplo de decisión del usuario final humano y a que precisamente la lógica difusa trata del modelado de tales situaciones, en este trabajo se ha optado por traducir las especificaciones del problema dadas por el usuario en forma de tabla de decisión difusa. La salida de este sistema difuso de evaluación (definido por el usuario) se denominará *índice de complejidad/exactitud (ICE)* y sus reglas serán de la forma:

SI ECMN es Pequeño AND Número_ de_ Parámetros es Grande ENTONCES ICE es Mediano (3-52)

De esta forma, la mejor configuración será la que presente un *ICE* mayor. Las reglas (3-52) pueden expresar cualquier relación, no necesariamente lineal, entre los dos objetivos que se pretenden minimizar conjuntamente. Es posible, por ejemplo, desechar cualquier configuración que tenga un *ECMN* por debajo de un nivel mínimo y, una vez superado éste, establecer una relación más o menos lineal. Un ejemplo sencillo de cómo podría ser el conjunto de reglas para la determinación del *ICE* se presenta en la Tabla 3.1.

Z = zero, VS = Very Small, S = Small, M = Medium, B = Big, VB = Very Big.

#Parms / ***ECMN***	**Z**	**S**	**M**	**B**
Z	VB	VB	B	M
S	VB	B	M	M
M	M	S	VS	Z
B	VS	Z	Z	Z

Tabla 3.1 Posible conjunto de reglas para determinar el índice ICE.

3.7 Resumen y funcionamiento global del algoritmo

En la figura 3.4 se muestra un organigrama del algoritmo presentado en este capítulo mostrando cada una de las etapas de que consta.

El punto de partida será un sistema difuso vacío, es decir, un sistema donde cada variable de entrada tiene asignada una sola función de pertenencia, lo que es equivalente a no seleccionarla. De esta manera, el conjunto inicial constará sólo de una regla y la salida será una constante. Posteriormente, se aplicará la expresión (3-18) para obtener el valor óptimo de dicha regla, que en este caso será simplemente el valor medio de los datos de salida.

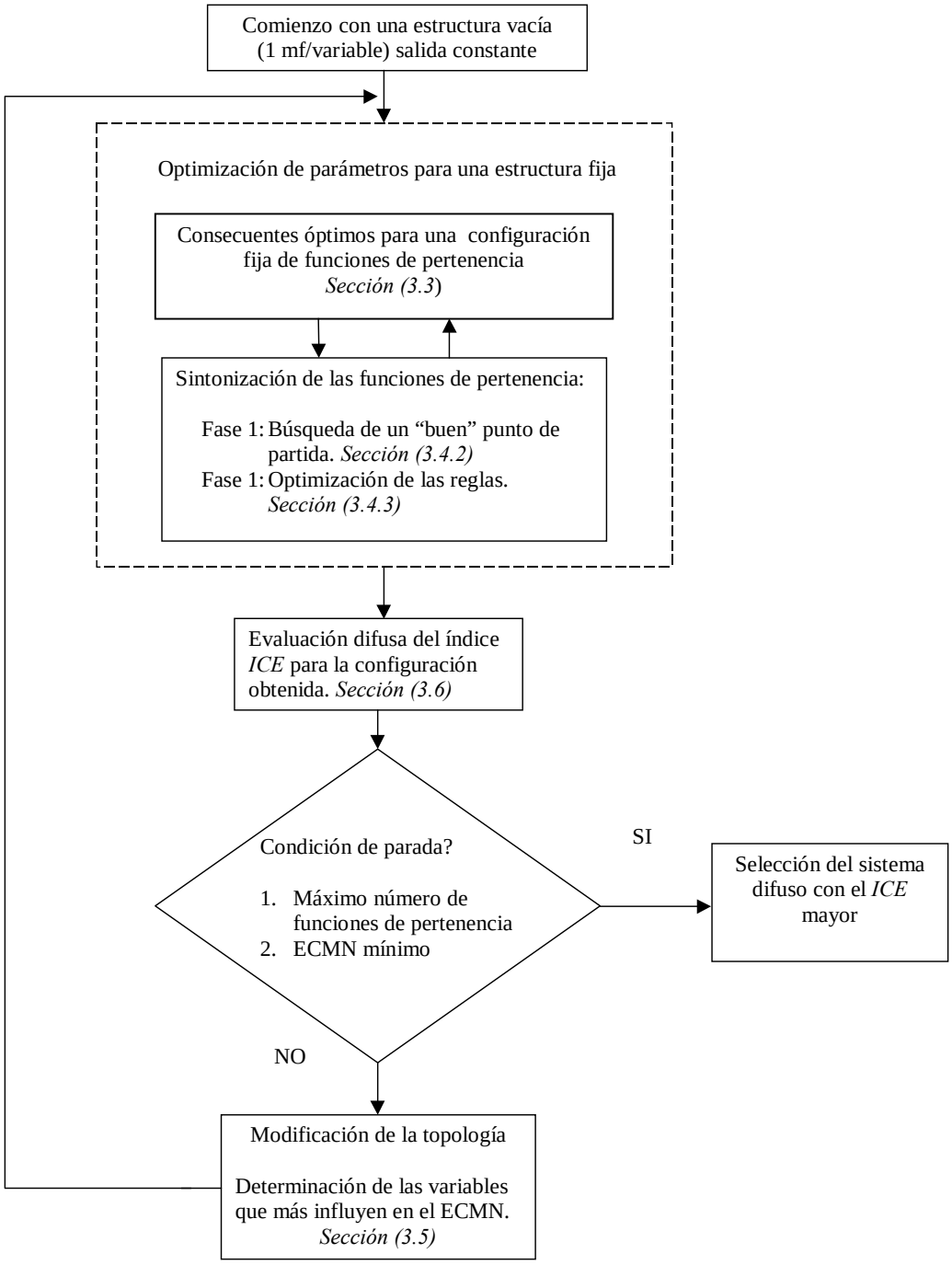


Fig. 3.4 Organigrama del algoritmo propuesto.

Como no hay ninguna función de pertenencia que optimizar, en este caso, se evaluará la configuración obtenida calculando su índice *ICE* (aunque este proceso también se puede llevar a cabo al final del algoritmo) y se almacenará dicha configuración. Posteriormente, se ejecutará la etapa de modificación de la topología (sección 3.5) donde se determinará en qué variables insertar una nueva función de pertenencia, con lo

que ya algunas de éstas comenzarán a contar realmente en el sistema y volveremos a la fase inicial, partiendo de la nueva configuración.

En general, cada configuración inicial deberá ser sometida a las siguientes etapas:

1. Determinación de los consecuentes óptimos de las reglas para una configuración fija de funciones de pertenencia utilizando la expresión (3-18) (sección 3.3).

2. Determinación conjunta de los parámetros que definen las funciones de pertenencia en la entrada y de los consecuentes de las reglas generadas (volviendo a la etapa 1). Dicho proceso de optimización constará dos fases:

   - En la primera (sección 3.4.2, ecuaciones (3-25) y (3-26)) se intentará buscar un "buen" punto de partida donde se deben encontrar los mejores mínimos locales.

   - En la segunda fase (sección 3.4.3, ecuaciones (3-31) ó (3-33)) se realiza un descenso en gradiente para hallar el primer mínimo local que se encuentre cerca del punto de partida.

3. Evaluación del compromiso entre complejidad y exactitud para cada una de las configuraciones obtenidas por el algoritmo (sección 3.6).

4. Análisis de la superficie del error cometido para determinar en qué variables se ha de aumentar el número de funciones de pertenencia que particionan sus dominios de modo que se mejore el *ECMN* de forma óptima (sección 3.5). Se comprueba, asimismo, si se ha alcanzado el número máximo de funciones de pertenencia permitido o si el *ECMN* está por debajo de un valor mínimo preestablecido. Si no es así, se regresa a la fase 1, siendo la nueva configuración inicial la configuración recién obtenida en esta etapa.

3.8 Mejoras del algoritmo

En este apartado se intentarán resolver una serie de cuestiones adicionales surgidas de la aplicación del algoritmo presentado en este capítulo y destinadas a mejorar y generalizar su funcionamiento.

3.8.1 Funciones incompletamente especificadas

En la mayoría de los casos reales, los vectores de E/S de los que partimos como función a aproximar no barren completamente todo el rango posible de entrada. En estos casos pueden existir reglas que jamás se activen y nuestro algoritmo debe estar preparado para ello. Si intentáramos resolver directamente la ecuación (3-18) para estos casos, obtendríamos un determinante nulo y el sistema, por lo tanto, sería indeterminado. La forma de elaborar nuestra teoría nos permite abordar este problema de forma sencilla: basta ver qué elemento de la diagonal de la matriz (3-16) es cero o, está por debajo de un cierto umbral. Para justificar esto, basta darse cuenta de que cada elemento de la diagonal, que viene dado por:

$$S_{i_1 i_2 \ldots i_N, i_1 i_2 \ldots i_N} \equiv \sum_{k \in D} \left[\frac{\prod_{m=1}^{N} \mu_{X_m^{i_m}}(x_m^k)}{\sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_N=1}^{n_N} \left(\prod_{m=1}^{N} \mu_{X_m^{i_m}}(x_m^k) \right)} \right]^2 \tag{3-53}$$

no es más que, esencialmente, la suma de los grados de pertenencia al cuadrado de todos los datos de entrada a las funciones de pertenencia que son los antecedentes de la regla dada por

$$SI\ x_1\ es\ X_1^{i_1}\ AND\ x_2\ es\ X_2^{i_2}\ AND\ \ldots\ AND\ x_N\ es\ X_N^{i_N}\ ENTONCES\ \ z=R_{i_1 i_2 \ldots i_N} \tag{3-54}$$

Si este valor es cero es porque ningún dato activa dicha regla por lo que ésta quedará marcada como ‘no usada’ y no entrará a formar parte en el sistema de ecuaciones a resolver, de modo que no provocará que el determinante se anule por su culpa. De esta manera, aunque tengamos sistemas con un gran número de entradas, si el número de datos no es lo suficientemente grande como para activar todas las reglas, al anularse aquellas que no han sido activadas la matriz obtenida resulta en realidad mucho menor de lo que inicialmente podría ser, permitiendo, de esta manera, abordar el problema del crecimiento exponencial del número de reglas para este tipo de casos.

3.8.2 Adaptación del algoritmo para otras configuraciones de funciones de pertenencia

A lo largo de este capítulo se ha utilizado siempre una partición triangular como forma de particionamiento del espacio de entrada. Sin embargo, como ya se comentó en la sección 3.4, el algoritmo aquí presentado puede ser adaptado a otros tipos de funciones de pertenencia.

Cuando se usa una partición triangular (TP), la función de salida es continua en todos los puntos y derivable también en todos los puntos salvo en aquéllos que caen en los picos de las funciones de pertenencia triangulares. Por lo general, esta no derivabilidad en ciertos puntos no conlleva ningún problema en aplicaciones como el modelado o el control discreto. Hay, sin embargo, situaciones como en control continuo, donde conviene que la función de salida no sea sólo continua sino que también presente "suavidad" en su forma. Esta "suavidad" viene relacionada unívocamente con la existencia de la función derivada a lo largo de todos los puntos de la función.

Cuando se quiere que la superficie de salida sea suave en todos los puntos, lo que se suele hacer en control difuso es substituir las funciones triangulares por gaussianas. Una primera idea que florece para emplear funciones gaussianas en el algoritmo presentado es tratar de utilizar una especie de "partición gaussiana" de la misma forma que tenemos una partición triangular, intentando definir los valores de las desviaciones a partir de la posición de los valores centrales, tal y como ocurre en una TP. Para llevar a cabo esta tarea, es necesario que las gaussianas presenten una caída distinta por un lado que por el otro, por lo que dejarán de ser gaussianas sino pseudo-gaussianas asimétricas (ver Apéndice A, sección A.2.2). Como la derivada en el punto central sigue siendo cero, la función de salida seguirá siendo derivable en todos los puntos. En el caso de una TP, los laterales se hacían coincidir con los centros de las funciones vecinas. En una partición pseudo-gaussiana habrá que fijar una constante L que determine el grado de pertenencia de una función en el centro vecino[14]. De esta forma, tendremos una configuración completamente equivalente a la TP y todo el algoritmo se podrá aplicar a dicha nueva

[14] Otra posibilidad sería definir el grado de pertenencia en el punto de cruce (por ejemplo 0.5)

configuración de forma directa, salvo que ahora el denominador de (3-2) no tendrá por qué valer la unidad y habrá que recurrir a las expresiones generales para calcular el valor de salida y su derivada. En ese último caso, en lugar de la (3-28) tendremos:

$$\frac{\partial \widetilde{F}(\vec{x}^{\,k};R,C)}{\partial c_v^j} = \frac{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(R_{i_1 i_2 \ldots i_N}\cdot\frac{\partial\prod_{m=1}^{N}\mu_{X_m^{i_m}}(x_m^k)}{\partial c_v^j}\right)}{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(\prod_{m=1}^{N}\mu_{X_m^{i_m}}(x_m)\right)} - \widetilde{F}(\vec{x}^{\,k};R,C)\frac{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\frac{\partial\prod_{m=1}^{N}\mu_{X_m^{i_m}}(x_m^k)}{\partial c_v^j}}{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(\prod_{m=1}^{N}\mu_{X_m^{i_m}}(x_m)\right)} \qquad (3\text{-}55)$$

donde los grados de pertenencia y las parciales vendrán dadas, para una partición pseudo-gaussiana (PGP), por las ecuaciones A-7 y A-8, respectivamente.

Hay otras ocasiones, sin embargo, que debido a la dependencia exponencial del número de reglas con el número de funciones de pertenencia, se prefiere complicar éstas últimas para así poder optimizar más parámetros para un mismo número de reglas. En tales casos se puede optar por funciones gaussianas propiamente dichas, donde, además del valor central también se puede optimizar su desviación, o por funciones triangulares libres, es decir, aquéllas en las que los dos laterales de cada función también pueden optimizarse. Al igual que se hizo con la analogía entre TP y partición pseudo-gaussiana, también podemos definir una configuración de pseudo-gaussianas libres, donde habrá tres parámetros por cada función de pertenencia (su centro y las dos desviaciones laterales). Todas estas configuraciones están definidas en el Apéndice A.

Para adaptar el algoritmo a las distintas configuraciones de funciones de pertenencia, vamos a repasar las distintas etapas del algoritmo haciendo hincapié en los cambios en la metodología que dichas configuraciones provocan:

1. Cálculo de los consecuentes óptimos: En este apartado, las ecuaciones (3-18) siguen siendo universalmente válidas ya que su demostración se obtuvo para una configuración de funciones de pertenencia genérica.

2. Búsqueda de un "buen" punto inicial: En el caso de una partición triangular, el proceso de búsqueda era muy fácil de realizar ya que sólo teníamos como parámetros libres los propios centros de las funciones de pertenencia. En el caso de funciones de pertenencia más complicadas disponemos de más grados de libertad por lo que podemos imponer más condiciones al punto inicial. La condición que impondremos aquí es que a ese punto inicial se debe llegar de tal forma que el conjunto de reglas difusas sea fácilmente interpretable. Como se verá en el apartado siguiente, una de las ventajas del control difuso es la facilidad con la que se puede interpretar su funcionamiento y dicha interpretabilidad se ve seriamente amenazada cuando se utilizan, entre otras cosas, funciones de pertenencia con varios parámetros (ver sección 3.8.3). Así pues, durante la duración de esta etapa lo que se hará (en el caso de usar funciones de pertenencia con más de un parámetro) será fijar el resto de parámetros para asegurar la interpretabilidad del estado final y mover los centros de la misma manera como se hacía con una TP. En cada iteración, se volverán a calcular los demás parámetros para seguir verificando las condiciones de interpretabilidad. Por ejemplo, en el caso de una configuración de funciones de pertenencia triangulares libres, una de las posibilidades para asegurar la interpretación es fijar los extremos a los valores de los centros vecinos, asegurándose de esa forma que un dato jamás active más de dos funciones de pertenencia[15]. En el caso de gaussianas habrá que establecer un valor medio de las desviaciones derecha e izquierda que se obtendrían si fueran pseudo-gaussianas libres. Un análisis más detallado de este proceso se realiza en la sección 4.3 del capítulo 4. En el apartado siguiente desarrollaremos el concepto de interpretabilidad de las reglas, mostrando cómo aplicarlo a las distintas configuraciones definidas en el Apéndice A.

3. Descenso en gradiente: El proceso de descenso en gradiente no entraña ninguna dificultad sin más que considerar que ahora el denominador de (3-2) ya no tiene por qué ser la unidad por lo que habría que utilizar la expresión general (3-55) para el

[15] desde ese punto de vista, una TP no sería más que un caso particular de una configuración de triangulares libres donde hemos fijado los extremos para asegurar la interpretabilidad del sistema resultante.

cálculo de las derivadas parciales. En cuanto al proceso de descenso en sí, habría que calcular la derivada no sólo respecto a cada centro sino también respecto a cada uno de los parámetros libres. El trato que se hizo en la sección 3.4.3 puede ser aplicado de igual manera al resto de parámetros si bien, la expresión (3-33) debería expresarse ahora de forma más genérica como:

$$\eta_{\theta_v^j} = \begin{cases} \dfrac{\theta_v^j - \theta_{v,min}^j}{b} & \text{si } \dfrac{\partial J(R,C)}{\partial \theta_v^j} \geq 0 \\ \dfrac{\theta_{v,max}^j - \theta_v^j}{b} & \text{si } \dfrac{\partial J(R,C)}{\partial \theta_v^j} < 0 \end{cases} \tag{3-56}$$

donde $\theta_{v,min}^j$ y $\theta_{v,max}^j$ son los valores límite impuestos al parámetro θ_v^j debido a la condición extra de la interpretabilidad del resultado final. Dichos valores límite dependerán de la situación de las funciones vecinas, como veremos en el apartado siguiente. Al usar este método, la preservación de la interpretación queda asegurada durante el descenso. Si se utilizara la expresión (3-31), la limitación no se haría de forma gradual (soft) sino que habría que hacerla de forma abrupta (hard).

4. Adición de funciones de pertenencia: En cuanto a esta etapa se refiere, aunque la sección 3.5 se escribió teniendo una partición triangular en mente, realmente no hay que hacer ningún cambio en el caso de otras configuraciones. Sólo hay que tener en cuenta que, cuando se analice una variable, hay que mantener fijas sus funciones de pertenencia (y no sólo los centros) tal y como hayan resultado después de la etapa del descenso.

5. Funciones incompletamente especificadas: En el caso de utilizar funciones gaussianas, al tener éstas un grado de pertenencia distinto de cero para todo punto, nunca ocurrirá que una regla no se active, por lo que teóricamente nunca habrá problemas en la indeterminación de la matriz (3-16). Sin embargo, tampoco sería deseable que una regla apenas activada esté ocupando memoria que puede ser vital en casos multidimensionales por lo que es lógico introducir un valor umbral que exprese el grado mínimo activación de una regla para que ésta se tenga en cuenta. Un límite razonable sería considerar el caso en el un sólo dato active completamente la regla por lo que podríamos elegir como valor umbral la unidad.

En el capítulo 4 se mostrarán ejemplos de aproximación con cada una de las configuraciones definidas en el Apéndice A.

3.8.3 Preservación de la interpretabilidad de las reglas

El algoritmo aquí presentado puede utilizarse para encontrar la estructura y parámetros de un sistema difuso a partir de datos de E/S. Como ya se comentó, puede ocurrir que los datos existentes no barran todos los estados posibles por lo que habrá reglas indeterminadas. Tanto en modelado de sistemas como en control, es siempre deseable que las reglas finales puedan ser fácilmente interpretables por un operador humano de forma que éste pueda dar unos valores apropiados a las reglas correspondientes a tales estados para los que no existe ninguna otra información. Sin embargo, cuando aplicamos el algoritmo de descenso en gradiente, éste siempre mueve los parámetros a optimizar en la dirección en la que se disminuye el error sin tener en cuenta la forma final de las funciones de pertenencia.

Supongamos que tenemos definidas cinco funciones de pertenencia correspondientes a la variable lingüística *temperatura*. Dichas funciones podrían ser etiquetadas como: 'mucho frío', 'frío', 'templado', 'calor' y 'mucho calor'. En las figuras 3.5a y 3.5b se representa una posible configuración final de estas funciones de pertenencia habiendo usado una partición triangular y funciones triangulares libres respectivamente.

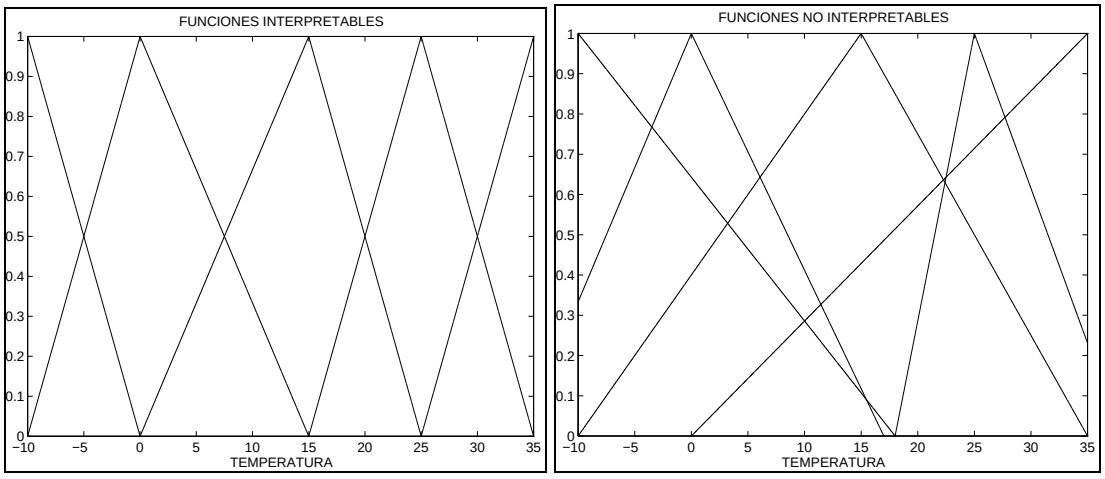


Fig. 3.5 Funciones de pertenencia a) interpretables. b) no interpretables.

Es claro que al usar una partición triangular, las reglas finales pueden ser interpretadas de una manera más sencilla ya que el grado de solapamiento de las funciones es siempre dos. Cuando nos dicen que hace mucho calor con grado 0.4 y calor con grado 0.6 podemos imaginarnos la temperatura real que tenemos. Sin embargo, al usar la configuración de la figura 3.5b, es posible que una temperatura active funciones de pertenencia tan diferentes como 'mucho frío' y 'calor', perdiéndose así la interpretabilidad de tales funciones de pertenencia aunque el sistema final consiga un grado de aproximación superior. Adicionalmente, el número de reglas activadas por un solo dato puede ser muy grande, lo cual puede ser una desventaja a la hora de realizar una implementación hardware del posible controlador.

Pero la interpretabilidad de sistemas difusos no siempre ha sido un factor a tener en cuenta. En [TAK-85], [SUG-88a], [JAN-92] y muchos otros trabajos, los consecuentes de las reglas son del tipo TSK (sección 2.5.3.2) consiguiendo así introducir más parámetros en cada regla y reduciendo, por ello, el número necesario de éstas para un grado de aproximación dado. Sin embargo, esta ventaja se consigue a expensas de perder la interpretabilidad de las reglas. Otra forma de aumentar el número de parámetros en cada regla sin tener que recurrir a consecuentes polinomiales es la que se utiliza en [WAN-92c], donde cada regla lleva asociadas sus propias funciones de pertenencia. Esto permite poder interpretar cada regla de forma individual pero no así el sistema de forma general, ya que una función de pertenencia que puede simbolizar el concepto "pequeño" para una regla tiene una forma distinta para otra.

Una de las principales ventajas de la lógica difusa frente a las redes neuronales y una de las razones por las que se creó, es su habilidad para utilizar el conocimiento intuitivo expresado de forma lingüística y, de forma inversa, poder obtener conocimiento lingüístico a partir de un sistema difuso. Al perder la interpretabilidad del conjunto de reglas, los sistemas difusos se convierten esencialmente en cajas negras del mismo modo que se considera a las redes neuronales. Esta característica puede provocar que el sistema final se considere no adecuado para muchas aplicaciones industriales en las que la fiabilidad y un fácil mantenimiento son consideradas propiedades fundamentales.

El sistema difuso utilizado en este trabajo mantiene las mismas funciones de pertenencia para todas las reglas y los consecuentes son valores escalares, por lo que el único problema que debemos abordar en este apartado es la posible pérdida de la interpretabilidad debida al proceso de descenso en gradiente. En [SON-93] se utiliza una "medida de la similaridad" entre funciones de pertenencia de modo que una función queda eliminada si se hace mucho más estrecha que la función inicial de la que parte y, por otra parte, cuando dos funciones son muy similares se colapsan para formar una sola. De este modo, el descenso en gradiente se puede realizar sin limitaciones. En [LOT-96] se considera que cada parámetro a optimizar está acotado entre unos valores límite de modo que se adapta el descenso en gradiente para que no se sobrepasen dichos límites. Sin embargo, en este trabajo no se aborda el problema de calcular esos límites según unas ciertas especificaciones.

Una solución válida para nuestro algoritmo en estos casos sería utilizar una partición triangular o una partición pseudo-gaussiana cuando se quiera preservar la interpretabilidad de las reglas, pero presentan la desventaja de que, al utilizar un menor número de parámetros por función de pertenencia, para poder llegar a una misma especificación necesitarían un mayor número de funciones de pertenencia por variable y, por tanto, un número mucho mayor de reglas.

Por tanto, debemos llegar a un compromiso entre la precisión de nuestro controlador y la interpretabilidad de las reglas finales. Como esto forma parte de las especificaciones de nuestro controlador, el usuario debe introducir a priori una serie de valores que afectan al solapamiento máximo permitido entre funciones de pertenencia. A continuación intentaremos definir una serie de condiciones para asegurar la interpretabilidad del sistema final, extrayendo de cada una de ellas los valores máximo y mínimo para cada uno de los parámetros definidos en el Apéndice A:

- **Las funciones de pertenencia no deben poder acercarse indefinidamente**: Para evitar este colapso que provoque la identificación de dos funciones de pertenencia como una sola se define un parámetro d_{min} que indique la distancia mínima permitida de acercamiento entre dos centros vecinos. Más concretamente, d_{min}

especifica el tanto por uno, respecto al rango de la variable, de la distancia mínima que pueden separar dos centros vecinos. Un valor típico puede ser $d_{min} = 0.05$ *(5%)* de tal forma que si el rango de la variable es [0,1], dos centros nunca se permitirá que se encuentren más cercanos que una distancia de 0.05. Matemáticamente, para todas las configuraciones tendríamos (r_v es el rango de la variable v, es decir, la anchura de su dominio de definición):

$$\left(c_v^j\right)_{max} = c_v^{j+1} - d_{min} \cdot r_v \qquad \left(c_v^j\right)_{min} = c_v^{j-1} + d_{min} \cdot r_v \tag{3-57}$$

- **Ningún dato debe activar (con grado apreciable) más de dos funciones de pertenencia**: El cerebro humano puede entender que un determinado valor de la temperatura sea caliente en un cierto grado apreciable y templado en otro grado pero no que se active también apreciablemente una tercera función de pertenencia como puede ser frío. Para ello se definirá la "apertura de solapamiento" a_s que especifica el grado de pertenencia máximo que podemos permitir que un dato active una tercera función de pertenencia. Para ello, debemos calcular el grado de solapamiento de la función de pertenencia $\mu_{X_v^j}$ con las funciones $\mu_{X_v^{j-2}}$ y $\mu_{X_v^{j+2}}$ e imponer que dicho valor sea como máximo a_s.

  Tras sencillos cálculos, en el caso de funciones triangulares libres tendremos:

$$\left(c_v^{j,-}\right)_{min} = c_v^{j-2,+} - \frac{a_s}{1-a_s}\left(c_v^j - c_v^{j-2}\right) \quad \left(c_v^{j,+}\right)_{max} = c_v^{j+2,-} + \frac{a_s}{1-a_s}\left(c_v^{j+2} - c_v^j\right) \tag{3-58}$$

  y en el caso de funciones pseudo-gaussianas libres[16]:

$$\left(\sigma_v^{j,-}\right)_{max} = \left(\frac{c_v^j - c_v^{j-2}}{\sqrt{ln\left(1/a_s\right)}} - \sqrt{\sigma_v^{j-2,+}}\right)^2 \quad \left(\sigma_v^{j,+}\right)_{max} = \left(\frac{c_v^{j+2} - c_v^j}{\sqrt{ln\left(1/a_s\right)}} - \sqrt{\sigma_v^{j+2,-}}\right)^2 \tag{3-59}$$

- **No debe existir ningún dato, dentro del rango posible de entrada, que no se active**: Debemos asegurar un grado mínimo de activación para cada dato de entrada. Se definirá la activación mínima a_m de modo que siempre existirá una función de

[16] El caso de funciones gaussianas estándar será tratado como un caso particular de funciones pseudo-gaussianas libres al final de esta sección.

pertenencia que active un dato con grado ≥ a_m. Para ello, debemos calcular el punto de corte de una función de pertenencia con sus vecinas (más allá de este punto, es la función vecina la que más se activaría). Por este concepto obtenemos las siguientes limitaciones:

Funciones triangulares libres:

$$\left(c_v^{j,-}\right)_{max} = c_v^{j-1,+} - \frac{a_m}{1-a_m}\left(c_v^j - c_v^{j-1}\right) \qquad \left(c_v^{j,+}\right)_{min} = c_v^{j+1,-} + \frac{a_m}{1-a_m}\left(c_v^{j+1} - c_v^j\right) \tag{3-60}$$

Funciones pseudo-gaussianas libres:

$$\left(\sigma_v^{j,-}\right)_{min} = \left(\frac{c_v^j - c_v^{j-1}}{\sqrt{ln\left(1/a_m\right)}} - \sqrt{\sigma_v^{j-1,+}}\right)^2 \qquad \left(\sigma_v^{j,+}\right)_{min} = \left(\frac{c_v^{j+1} - c_v^j}{\sqrt{ln\left(1/a_m\right)}} - \sqrt{\sigma_v^{j+1,-}}\right)^2 \tag{3-61}$$

- **Una función de pertenencia no debe comprimirse indefinidamente**: Los valores de sus laterales no deben poder acercarse indefinidamente al centro hasta el punto de que la función de pertenencia no llegue a tener soporte. Para ello, dada una función de pertenencia, en los puntos equidistantes de su centro una distancia *d_min* debe activarse dicha función con grado ≥ a_m. El punto anterior garantiza que al menos hay una que lo hace. En este punto aseguramos que dicha función debe ser justamente ésta:

$$\mu_{X_v^j}\left(c_v^j \pm d_{min} \cdot r_v\right) \geq a_m \tag{3-62}$$

Funciones triangulares libres:

$$\left(c_v^{j,-}\right)_{max} = c_v^j - \frac{d_{min} \cdot r_v}{1-a_m} \qquad \left(c_v^{j,+}\right)_{min} = c_v^j + \frac{d_{min} \cdot r_v}{1-a_m} \tag{3-63}$$

Funciones pseudo-gaussianas libres:

$$\left(\sigma_v^{j,-}\right)_{min} = \left(\sigma_v^{j,+}\right)_{min} = \frac{\left(d_{min} \cdot r_v\right)^2}{ln\left(1/a_m\right)} \tag{3-64}$$

En la tabla 3.2 se presenta un resumen de los valores máximo y mínimo de cada uno de los parámetros. En el caso de utilizar gaussianas estándar, se utilizará, como es lógico,

como valor mínimo (máximo) de la desviación el máximo (mínimo) de los valores que se hubieran obtenido si fueran pseudo-gaussianas libres.

De esta forma, cuando realicemos la etapa de descenso en gradiente, cada parámetro a optimizar contará, en cada instante, con unos valores mínimo y máximo permitidos que son los que se utilizarán en la expresión (3-56). En el caso de usar un descenso en gradiente tradicional (ecuación (3-31)) el valor de salida del descenso en cada parámetro vendrá limitado de forma abrupta por la función:

$$sat(x;a,b)=\begin{cases} a & x<a \\ x & x\geq a \text{ y } x\leq b \\ b & x>b \end{cases} \tag{3-65}$$

	TP PGP	Triangulares libres	Pseudo-gaussianas libres	Gaussianas
c_v^j	$\left(c_v^j\right)_{max}=c_v^{j+1}-d_{min}\cdot r_v$ $\left(c_v^j\right)_{min}=c_v^{j-1}+d_{min}\cdot r_v$			
$\theta_v^{j,-}$		$\left(c_v^{j,-}\right)_{min}=c_v^{j-2,+}-\frac{a_s}{1-a_s}\left(c_v^j-c_v^{j-2}\right)$ $\left(c_v^{j,-}\right)_{max}=min\begin{pmatrix} c_v^{j-1,+}-\frac{a_m}{1-a_m}\left(c_v^j-c_v^{j-1}\right) \\ c_v^j-\frac{d_{min}\cdot r_v}{1-a_m}\end{pmatrix}$	$\left(\sigma_v^{j,-}\right)_{max}=\left(\frac{c_v^j-c_v^{j-2}}{\sqrt{ln(1/a_s)}}-\sqrt{\sigma_v^{j-2,+}}\right)^2$ $\left(\sigma_v^{j,-}\right)_{min}=max\begin{pmatrix}\left(\frac{c_v^j-c_v^{j-1}}{\sqrt{ln(1/a_m)}}-\sqrt{\sigma_v^{j-1,+}}\right)^2 \\ \frac{(d_{min}\cdot r_v)^2}{ln(1/a_m)}\end{pmatrix}$	$\sigma_{v,min}^i=max(\sigma_{v,min}^{i,-},\sigma_{v,min}^{i,+})$
$\theta_v^{j,+}$		$\left(c_v^{j,+}\right)_{max}=c_v^{j+2,-}+\frac{a_s}{1-a_s}\left(c_v^{j+2}-c_v^j\right)$ $\left(c_v^{j,+}\right)_{min}=max\begin{pmatrix} c_v^{j+1,-}+\frac{a_m}{1-a_m}\left(c_v^{j+1}-c_v^j\right) \\ c_v^j+\frac{d_{min}\cdot r_v}{1-a_m}\end{pmatrix}$	$\left(\sigma_v^{j,+}\right)_{max}=\left(\frac{c_v^{j+2}-c_v^j}{\sqrt{ln(1/a_s)}}-\sqrt{\sigma_v^{j+2,-}}\right)^2$ $\left(\sigma_v^{j,+}\right)_{min}=max\begin{pmatrix}\left(\frac{c_v^{j+1}-c_v^j}{\sqrt{ln(1/a_m)}}-\sqrt{\sigma_v^{j+1,-}}\right)^2 \\ \frac{(d_{min}\cdot r_v)^2}{ln(1/a_m)}\end{pmatrix}$	$\sigma_{v,max}^i=min(\sigma_{v,max}^{i,-},\sigma_{v,max}^{i,+})$

Tabla 3.2 Condiciones impuestas a cada uno de los parámetros de las distintas funciones de pertenencia para mantener la interpretabilidad de las reglas finales.

3.9 Conclusión

En este capítulo se ha propuesto una nueva y completa metodología para el diseño automático de sistemas difusos a partir de la información proporcionada por un conjunto de vectores de E/S. Desde cero y a través del análisis de los datos, el algoritmo

va construyendo y optimizando sistemas difusos completos. Es decir, se ha abordado tanto el problema de la identificación de la estructura del sistema como el ajuste de sus parámetros.

Inicialmente, se calculan los consecuentes óptimos de las reglas para una configuración fija de funciones de pertenencia. Posteriormente, se realiza una determinación conjunta de los parámetros que definen las funciones de pertenencia en la entrada y de los consecuentes de las reglas generadas. Dicho proceso de optimización se ha desglosado en dos fases: en la primera fase se busca un "buen" punto de partida donde se deben encontrar los mejores mínimos locales. En la segunda realizamos un descenso en gradiente para hallar el primer mínimo local que se encuentre cerca del punto de partida.

Una vez optimizada la configuración actual, se analiza la superficie del error cometido para determinar en qué variables se ha de aumentar el número de funciones de pertenencia que particionan sus dominios, de modo que mejoremos el *ECMN* de manera óptima. De esta forma, el algoritmo es capaz de hallar cuántas funciones de pertenencia se deben utilizar para cada variable de entrada y, como caso particular, qué entradas son realmente necesarias para la consecución de un cierto grado de aproximación.

Finalmente, de entre todas las configuraciones obtenidas, se evalúa el compromiso entre complejidad y exactitud para cada una de ellas seleccionando aquélla con mayor índice *ICE*, viniendo éste definido en forma de reglas difusas que proporciona el usuario final de la aplicación.

En la última parte del capítulo se han presentado una serie de mejoras del algoritmo. En concreto, se propone cómo analizar conjuntos de datos de E/S que no exploren completamente el espacio de entrada, cómo adaptar el algoritmo a otras configuraciones de funciones de pertenencia y la forma de conseguir que el resultado final mantenga su interpretabilidad.

En el capítulo siguiente se mostrarán ejemplos de aplicación y comparaciones con otras metodologías utilizadas en la bibliografía.

CAPÍTULO 4

EJEMPLOS, EVALUACIÓN Y COMPARACIONES DEL ALGORITMO DE APROXIMACIÓN FUNCIONAL

En este capítulo se presentan ejemplos que demuestran la validez del procedimiento de síntesis automática de sistemas difusos a partir de datos de E/S, propuesto en el capítulo anterior.

En la sección 4.2 se utilizarán diversos ejemplos con objeto de poner de relieve las principales características del algoritmo. En los dos últimos subapartados de dicha sección, se evaluará la robustez del algoritmo frente al número de muestras utilizadas y su calidad (nivel de ruido). En la sección 4.3 se mostrará la utilización del algoritmo con distintas configuraciones de funciones de pertenencia y la forma de preservar la interpretabilidad de los sistemas obtenidos. Finalmente, en la sección 4.4 se compararán los resultados obtenidos con los proporcionados mediante otros algoritmos también destinados a la aproximación funcional, y se presentarán las conclusiones en el apartado 4.5.

4.1 Introducción

En el capítulo anterior se ha presentado una nueva metodología para el diseño automático de sistemas difusos a partir de datos de Entrada/Salida. A modo de resumen, dicha metodología constaba de una serie de pasos:

1. Determinación de los consecuentes óptimos de las reglas para una configuración fija de funciones de pertenencia (sección 3.3).

2. Determinación conjunta de los parámetros que definen las funciones de pertenencia en la entrada y de los consecuentes de las reglas generadas (volviendo a la etapa 1). Dicho proceso de optimización constaba de dos fases:

   - En la primera fase (sección 3.4.2) se intentaba buscar un "buen" punto de partida donde se deben encontrar los mejores mínimos locales.
   - En la segunda fase (sección 3.4.3) se realizaba un descenso en gradiente para hallar el primer mínimo local que se encuentre cerca del punto de partida.

3. Análisis de la superficie del error cometido para determinar en qué variables se ha de aumentar el número de funciones de pertenencia que particionan sus dominios, de modo que se mejorara el *ECMN* de forma óptima (sección 3.5). Una vez hecho esto, se verificaba si se había alcanzado el número máximo de funciones de pertenencia permitido o si el *ECMN* estaba por debajo de un valor mínimo preestablecido. Si no es así, se vuelve a la etapa 1 considerando como nueva configuración inicial la configuración recién obtenida en esta etapa.

4. Evaluación del compromiso entre complejidad y exactitud para cada una de las configuraciones obtenidas por el algoritmo, seleccionando aquélla con mayor índice *ICE* (sección 3.6).

Finalmente, se introducían ciertas mejoras en el algoritmo:

- Aplicación del algoritmo a datos de E/S que no exploren completamente todo el dominio posible de entrada.
- Utilización de otros tipos de funciones de pertenencia.
- Preservación de la interpretabilidad del sistema final.

A lo largo de este capítulo se estudiarán cada una de estas etapas, presentando ejemplos que pongan de relieve las características principales del algoritmo. Además, se analizará la robustez del mismo frente al número de datos y la presencia de ruido en los mismos. Finalmente se compararán los resultados con los obtenidos por otras metodologías existentes en la bibliografía destinadas a la aproximación funcional.

La mayoría de los ejemplos utilizados en este capítulo son funciones analíticas recogidas en [CHE-96] y [ROV-96], que presentan propiedades diversas y que son frecuentemente utilizadas para testear algoritmos de aproximación funcional. La ventaja de utilizar este tipo de funciones es que podemos conocer el valor de la función en cualquier punto, lo que facilita la evaluación de la aproximación obtenida.

La forma general de obtener los datos de E/S será la siguiente:

Los datos de entrenamiento se obtendrán dividiendo el espacio de entrada en retículas de mayor o menor tamaño, según la dimensionalidad de la función a aproximar, y se seleccionará un dato aleatoriamente dentro de cada una de dichas retículas. Así, para funciones de una dimensión dividiremos el espacio de entrada en 50 zonas equidistantes y seleccionaremos un dato al azar de cada una de ellas. En dos dimensiones, se dividirá cada componente en 20 zonas equidistantes que delimitarán un total de 400 retículas. De igual forma obtendremos 1000 puntos para funciones de tres dimensiones, 4096 para las de cuatro, etc.

Los datos de test, para aquellas secciones donde se utilizan, se generarán utilizando 100 puntos uniformemente distribuidos para funciones de una dimensión, 1000 para los de dos y 10000 para el resto de dimensiones. En los casos donde utilicemos funciones no analíticas, se cogerá una porción de los datos para entrenamiento y otro para los de test.

4.2 Principales características del algoritmo

En este apartado se analizarán las principales características del algoritmo propuesto en el capítulo anterior utilizando una serie de ejemplos sencillos que pongan de relieve dichas características. Todos los ejemplos de esta sección utilizarán como configuración para las funciones de pertenencia una partición triangular (ver Apéndice A). En la sección 4.3 se mostrarán ejemplos para otros tipos de configuraciones.

4.2.1 Consecuentes de las reglas

Al utilizar una partición triangular, se puede demostrar fácilmente que la función defuzzificada $\tilde{F}$ en una dimensión resulta ser una función continua y lineal a tramos, es decir, tiene la forma '$a+bx$' entre cada dos centros. En dos dimensiones, $\tilde{F}$ tiene la forma *'$a_0+b_1x_1+b_2x_2+c_{12}x_1x_2$'* entre cada una de las zonas delimitadas por las funciones de pertenencia, en tres dimensiones: *'$a_0+b_1x_1+b_2x_2+b_3x_3+c_{12}x_1x_2+c_{13}x_1x_3+c_{23}x_2x_3+d_{123}x_1x_2x_3$'*, etc. siempre valiendo $R_{j_1 j_2 \cdots j_N}$ en el punto $\vec{x} = (c_1^{j_1}, c_2^{j_2}, \ldots, c_N^{j_N})$, recordando que $c_v^{j_v}$ es el centro de la función de pertenencia j_v de la variable v. Conforme aumenta el número de entradas también aumenta la complejidad del tipo de función con el que se va aproximando cada región del espacio de entrada. No es difícil notar que la función resultante es continua pero no derivable, por lo general, en los puntos donde se encuentran los centros.

De esta forma, utilizando una partición triangular podremos aproximar de forma exacta cualquier recta, plano, hiperplano, superficies del tipo *'$a+bx+cy+dxy$'*, etc. utilizando tan sólo dos funciones de pertenencia por cada variable de entrada. Como ejemplo, para testear el método descrito en la sección 3.3, supongamos la función [ROV-96]:

$$y_1(x_1, x_2) = \frac{x_1 + x_2}{2} \tag{4-1}$$

definida en el intervalo [0,1]×[0,1]. En este caso, por ser un plano debería bastar con escoger 2 funciones de pertenencia por cada variable. La expresión 3-18 tendría en este caso la forma:

$$\begin{pmatrix} 46.81 & 21.61 & 21.61 & 9.97 \\ 21.61 & 46.81 & 9.97 & 21.61 \\ 21.61 & 9.97 & 46.81 & 21.61 \\ 9.97 & 21.61 & 21.61 & 46.81 \end{pmatrix} \bullet \begin{pmatrix} R_{11} \\ R_{12} \\ R_{21} \\ R_{22} \end{pmatrix} = \begin{pmatrix} 31.58 \\ 50.00 \\ 50.00 \\ 68.42 \end{pmatrix}$$

cuya solución es: $(R_{11},R_{12},R_{21},R_{22})=(0.0,0.5,0.5,1.0)$, es decir, las 4 reglas de este sistema serían:

$$\begin{array}{|l|}\hline IF\ x_1 \text{ is } X_1^1\ AND\ x_2 \text{ is } X_2^1\ THEN\ z = 0.0 \\ IF\ x_1 \text{ is } X_1^1\ AND\ x_2 \text{ is } X_2^2\ THEN\ z = 0.5 \\ IF\ x_1 \text{ is } X_1^2\ AND\ x_2 \text{ is } X_2^1\ THEN\ z = 0.5 \\ IF\ x_1 \text{ is } X_1^2\ AND\ x_2 \text{ is } X_2^2\ THEN\ z = 1.0 \\ \hline \end{array}$$

y su error cuadrático medio, como ya adelantábamos, es cero. En la figura 4.1 se representa la gráfica de la función. Aunque éste es uno de los casos más sencillos, nos sirve para empezar a testear el comportamiento de nuestro aproximador.

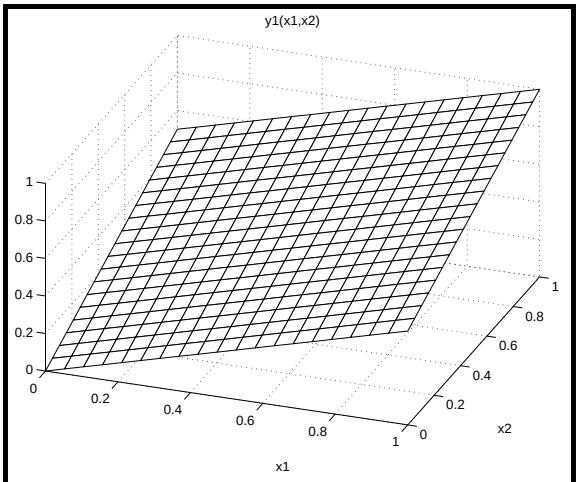


Fig. 4.1 Función $y_1(x_1,x_2)$.

En general, para cualquier tipo de rectas, planos e hiperplanos, con la distribución aquí presentada se puede obtener una aproximación exacta con tan sólo usar dos funciones de pertenencia por variable. Pero, evidentemente, lo realmente interesante es ver cómo se aproximan funciones mucho más complejas que éstas. Esto es lo que se estudiará en los dos subapartados siguientes. En el primero veremos cómo aproxima funciones de una sola entrada con el fin de ayudarnos a comprender mejor la forma de actuar de nuestro aproximador para la distribución de funciones de pertenencia definida. En el apartado 4.2.1.2 utilizaremos funciones con mayor número de entradas.

4.2.1.1 Funciones lineales a tramos

Consideremos la función $f(x)=x^2$ definida en el rango [0,10] (fig. 4.2). Intentaremos aproximarla usando 2, 3, 4, 5, 6 y 10 funciones de pertenencia. La forma de operar es la misma que ya se indicó. Primero generamos un fichero con 50 muestras (en este caso) de la función deseada, vamos construyendo la matriz correspondiente y resolvemos el

sistema de ecuaciones. En las figuras 4.3a a la 4.3f se muestran las distintas funciones aproximadas utilizando siempre un configuración equidistribuida de funciones de pertenencia. En la tabla que acompaña a la figura 4.2 se presentan los errores cuadráticos medios normalizados obtenidos:

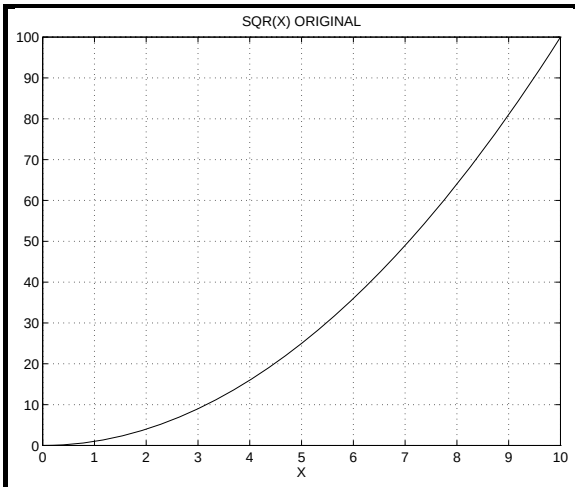


Fig. 4.2 Función x^2 original.

Nº de funciones de pertenencia	ECMN
2	0.255
3	0.0635
4	0.0282
5	0.0158
6	0.0101
10	0.0031

Como cabía esperar, conforme aumenta el número de funciones de pertenencia mejora substancialmente el grado de aproximación, pero éste no es el único factor que influye. Para ilustrar esto consideremos la función (fig. 4.4a)

$$f_2(x)=e^{-5x}\,sin(2\pi x) \qquad (4\text{-}2)$$

definida en el intervalo [0,1]. En la figura 4.4b aproximamos la función usando 4 funciones de pertenencia equidistribuidas (figura 4.5a). Utilizando la metodología presentada en la sección 3.3 del capítulo anterior obtenemos las reglas óptimas para este caso, consiguiéndose un *ECMN* = 0.511 (bastante mala aproximación). Sin embargo, si colocáramos las 4 funciones de pertenencia centradas en los puntos (0.0, 0.113, 0.504, 1.0) (figura 4.5b) que son los que nos proporciona el algoritmo completo (véanse siguientes secciones) obtendríamos un grado de aproximación sustancialmente mejor (*ECMN* = 0.0998) (ver fig. 4.4c). Este resultado es muy importante. La localización de los centros influye de forma crítica en el resultado final, a pesar de que las reglas son las óptimas para ambos casos. Además, como es obvio, esta localización es tanto más importante cuanto menor sea el número de funciones de pertenencia que se utilicen. Para poder obtener un grado de aproximación similar utilizando funciones equidistribuidas, tendríamos que necesitar, en este caso, 9 funciones de pertenencia, es decir, más del doble de reglas (figura 4.4d) con las que se obtiene un *ECMN* = 0.0918.

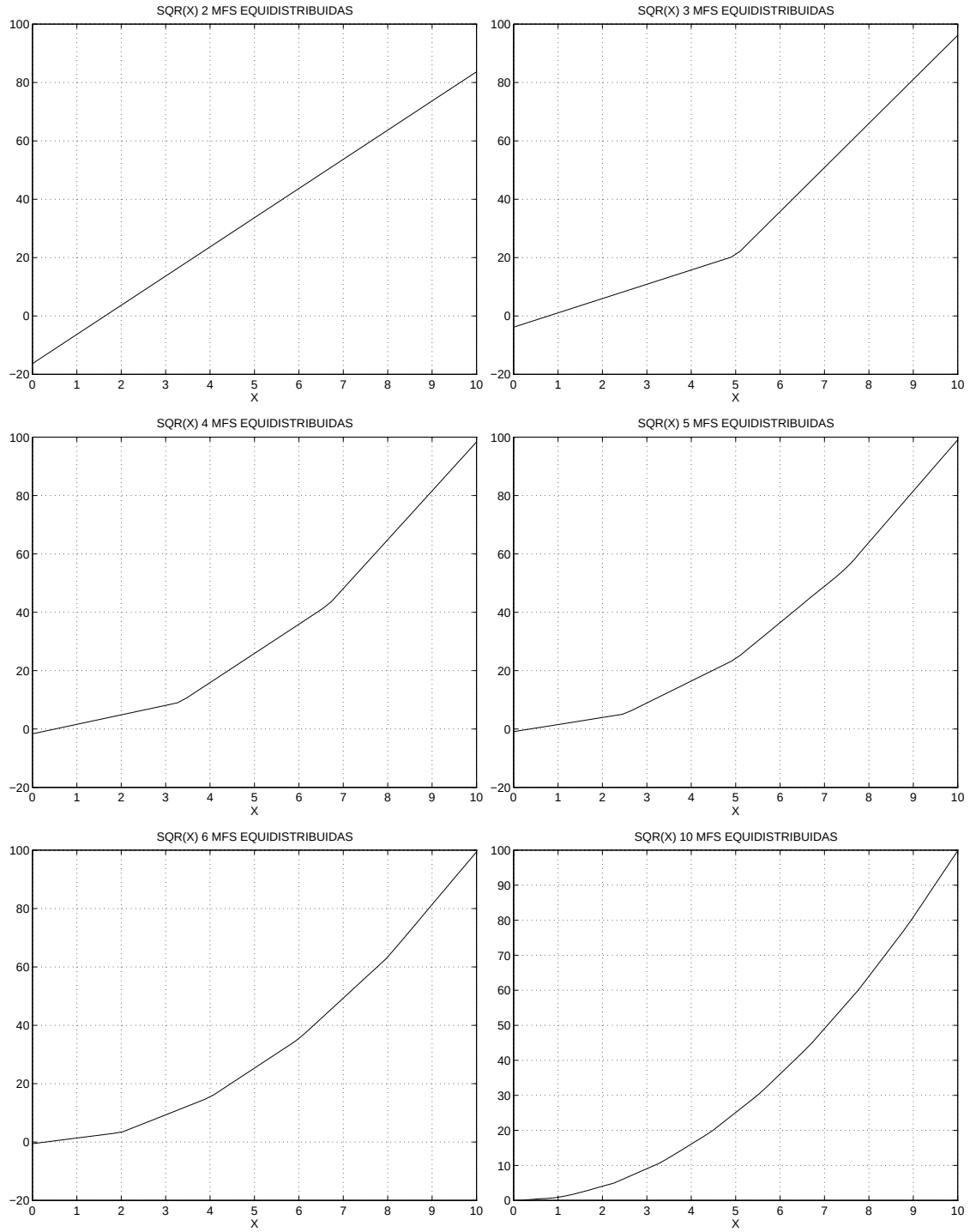


Fig. 4.3 Aproximación utilizando funciones de pertenencia equidistribuidas. a) 2 reglas b) 3 reglas c) 4 reglas d) 5 reglas e) 6 reglas f) 10 reglas.

El utilizar funciones sencillas unidimensionales facilita la comprensión de la forma en la que funciona el algoritmo y hace que todo sea muy intuitivo. Si aumentamos el número de centros en la zona donde la función es más alineal y encima donde se recogen los

valores mayores, vamos a obtener una mayor exactitud en esas zonas que son las más difíciles de aproximar y las que más van a contribuir al error cuadrático medio total.

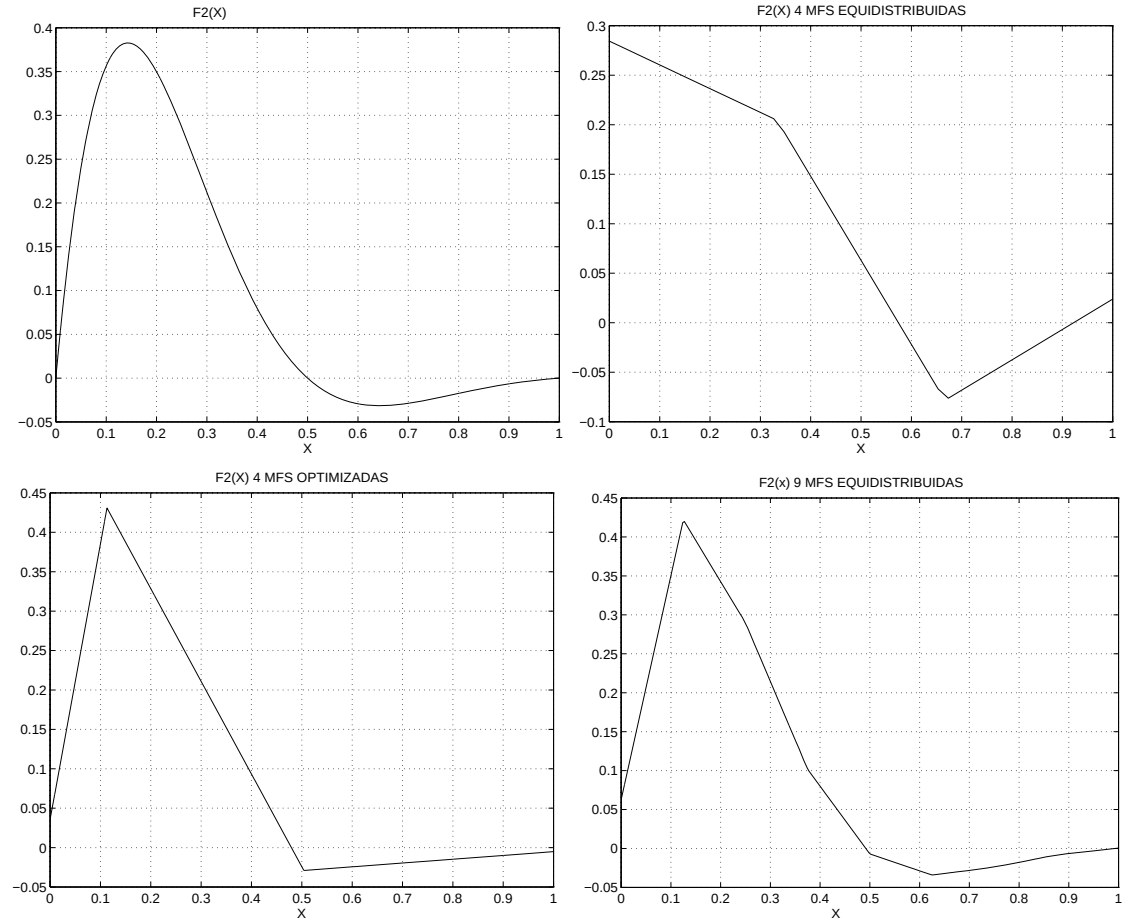


Fig. 4.4 a) Función $f_2(x)$ original. b) Función $f_2(x)$ aproximada mediante 4 funciones de pertenencia equidistribuidas. c) 4 MFs optimizadas. d) 9 MFs equidistribuidas.

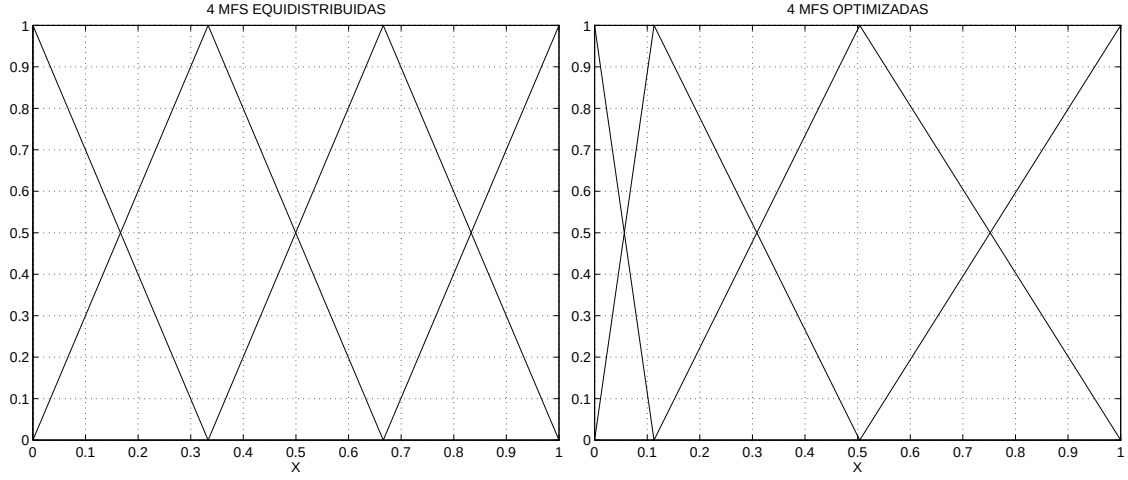


Fig. 4.5 Funciones de pertenencia usadas a) en la figura 4.4b. b) en la figura 4.4c.

Así, podemos concluir que:

1) Dependiendo de dónde estén nuestros centros tendremos, independientemente de que siempre obtengamos las reglas óptimas para éstos, resultados muy diversos.
2) Para pocas funciones de pertenencia, la localización de los centros es un factor crítico.

4.2.1.2 Algunas funciones de dos o más variables

Finalmente, en esta sección vamos a ver varios ejemplos de aproximación de funciones más complejas y con mayor número de entradas, utilizando siempre una configuración equidistribuida y fija de funciones de pertenencia. Las funciones de dos entradas permiten que se represente la superficie de salida y sea más intuitivo el estudiar la forma en que el aproximador realiza su trabajo. Las funciones de tres o más entradas se representarán mostrando distintas proyecciones sobre cada plano.

Consideremos, ahora, las siguientes funciones [ROV-96], [CHE-96]:

$$y_3(x_1,x_2) = \frac{1}{2} e^{-20\cdot\left[\left(x_1-\frac{1}{2}\right)^2+\left(x_2-\frac{1}{2}\right)^2\right]} + e^{-10\left(x_1^2+x_2^2\right)} \quad x_1,x_2 \in [0,1] \tag{4-3}$$

$$y_5(x_1,x_2) = \frac{1}{2}\left[1+sin\left(2\pi\, x_1\right)cos\left(2\pi\, x_2\right)\right] \qquad x_1,x_2 \in [0,1] \tag{4-4}$$

$$f_8(x_1,x_2) = sin\left(2\pi\sqrt{x_1^2+x_2^2}\right) \qquad x_1,x_2 \in [-1,1] \tag{4-5}$$

En las gráficas 4.6-4.8 se muestran los resultados. Estas funciones son mucho más complejas que la función y_1 anterior y se necesita un número mayor de funciones de pertenencia para conseguir buenas aproximaciones. De nuevo, es claro que la elección de los centros (que aquí se suponen distribuidos uniformemente) es un factor tanto más importante cuanto menor es el número de reglas que manejemos. En el caso bidimensional, la optimización de las funciones de pertenencia empieza a ser un factor mucho más crítico debido a la dependencia exponencial del número de reglas con el número de funciones de pertenencia. Para conseguir un mismo grado de aproximación utilizando funciones equidistribuidas se necesitará un mayor número de funciones de por cada variable lo que provoca un aumento muchísimo mayor en el número de reglas.

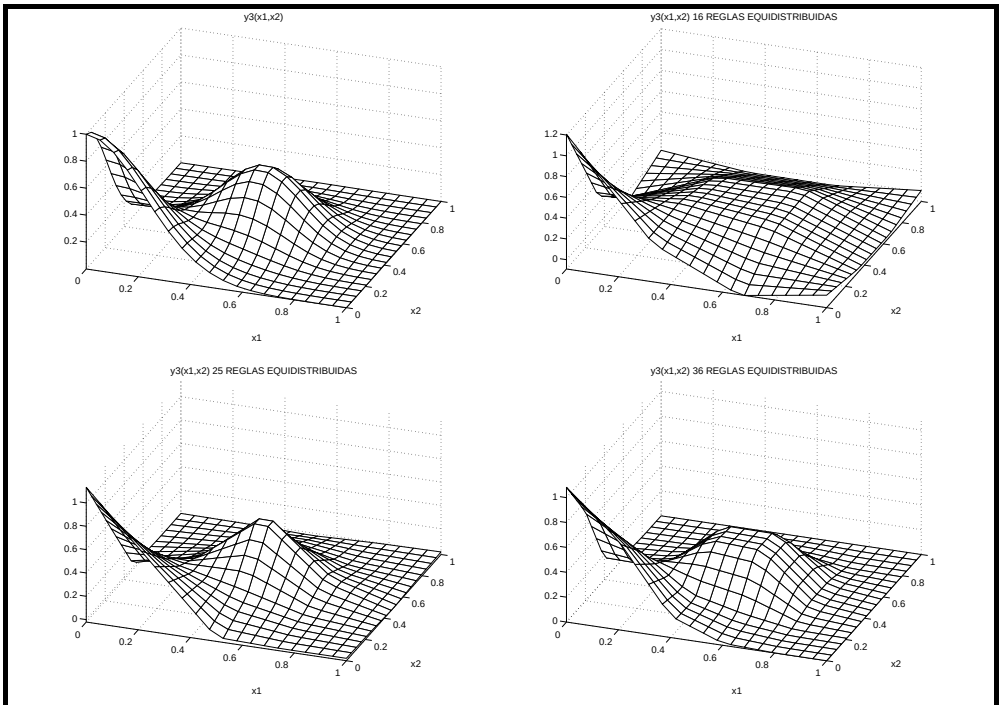


Fig. 4.6 a) Función $y_3(x_1,x_2)$ original. b) Aproximada mediante 4 funciones de pertenencia equidistribuidas en cada variable. ECMN=0.281. c) 5 funciones de pertenencia equidistribuidas. ECMN=0.103. d) 6 funciones de pertenencia equidistribuidas. ECMN=0.0947.

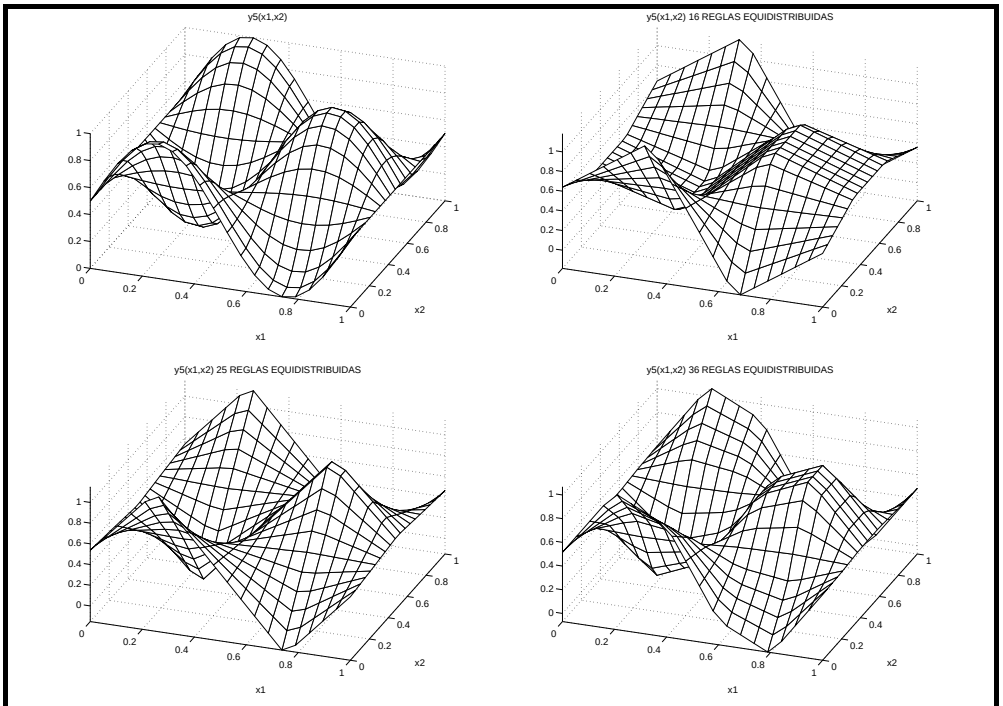


Fig. 4.7 a) Función $y_5(x_1,x_2)$ original. b) Aproximada mediante 4 funciones de pertenencia equidistribuidas. ECMN=0.327. c) 5 funciones de pertenencia equidistribuidas. ECMN=0.169. d) 6 funciones de pertenencia equidistribuidas. ECMN=0.100.

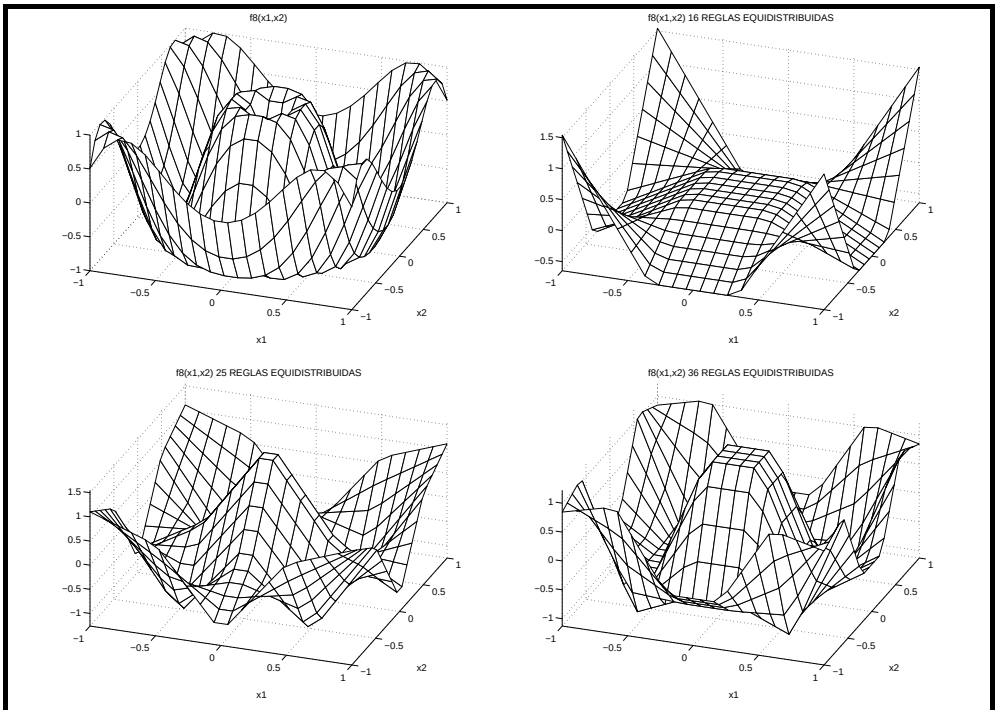


Fig. 4.8 a) Función $f_8(x_1,x_2)$ original. b) Aproximada mediante 4 funciones de pertenencia equidistribuidas. ECMN=0.803. c) 5 funciones de pertenencia equidistribuidas. ECMN=0.512. d) 6 funciones de pertenencia equidistribuidas. ECMN=0.321.

Finalmente, vamos a poner un ejemplo de función con tres entradas. La función a aproximar será [CHE-96]:

$$f_{10}(x_1,x_2,x_3) = 2x_1x_2e^{-4x_3}\cos(2\pi\, x_1x_2x_3) \qquad (4\text{-}6)$$

donde todas las entradas varían en el rango [0,1]. En las figuras 4.9a a la 4.9d se representa la proyección de dicha función fijando la tercera variable al valor x_3=0.31. Usando 5 funciones de pertenencia por cada variable se obtiene un *ECMN* = 0.0639, un valor más que aceptable pero el número de reglas ya empieza a ser considerable (125 reglas). En posteriores secciones veremos cómo se pueden obtener mejores grados de aproximación para la mayoría de estas funciones utilizando un número mucho menor de reglas.

A lo largo de esta sección se ha comprobado el funcionamiento de la metodología presentada en la sección 3.3 del capítulo anterior para la obtención de los consecuentes óptimos, pero asimismo, se ha puesto de manifiesto que, en la mayoría de las ocasiones, conviene optimizar también la localización de las funciones de pertenencia para poder

conseguir una mejor aproximación para un mismo número de reglas o, a la inversa, una drástica reducción del número de reglas para un mismo grado de aproximación.

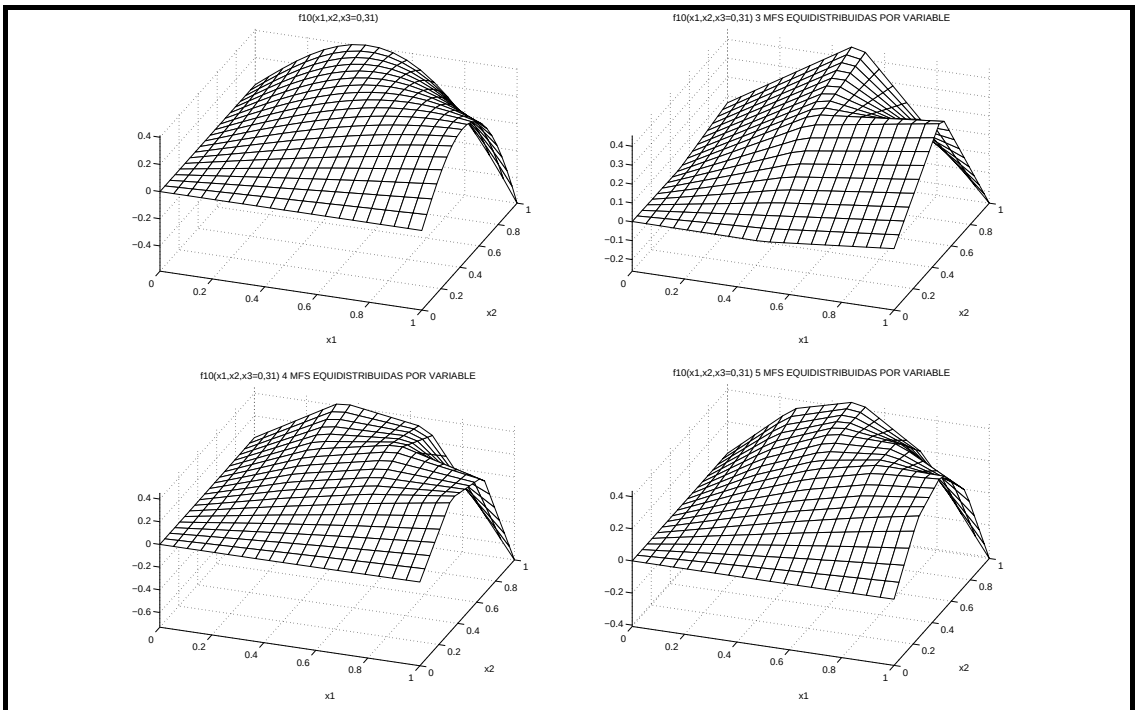


Fig. 4.9 Función $f_{10}(x_1,x_2,x_3)$ proyectada para x_3=0.31. a) Función original. b) Aproximada mediante 3 funciones de pertenencia equidistribuidas en cada variable. ECMN=0.223. c) 4 funciones de pertenencia equidistribuidas. ECMN=0.126. d) 5 funciones de pertenencia equidistribuidas. ECMN=0.0680.

4.2.2 Optimización de las funciones de pertenencia

Como ha quedado patente en la sección anterior, a veces no es suficiente con optimizar los consecuentes de las reglas sino que también conviene hacer esto mismo con las premisas, es decir, con la posición de las funciones de pertenencia que particionan el dominio de entrada.

En la sección 3.4 del capítulo anterior se presentó una metodología para hacer esto, consistente en dos etapas.

- En la primera (sección 3.4.2) se intentaba buscar un “buen” punto de partida donde se deben encontrar los mejores mínimos locales.
- En la segunda (sección 3.4.3) se realizaba un descenso en gradiente para hallar el primer mínimo local que se encuentre cerca del punto de partida.

En este apartado analizaremos la importancia de cada una de estas etapas.

4.2.2.1 Importancia de la etapa inicial

Para poner de manifiesto la importancia de una etapa inicial en nuestro algoritmo, consideremos la función (fig. 4.10):

$$f(x)=e^{-3x}\,sin(10\pi x) \tag{4-7}$$

definida en el intervalo [0,1]. Vamos a escoger 100 posibles configuraciones iniciales aleatorias para cada número de funciones de pertenencia entre 3 y 12 y realizaremos un descenso en gradiente para cada una de ellas para encontrar el primer mínimo relativo que se encuentre a partir de cada una de dichas configuraciones iniciales. En la figura 4.11a se representan gráficamente los resultados. Es obvio que, incluso para funciones unidimensionales, existen un gran número de mínimos locales y que éstos son cada vez más numerosos conforme aumenta el número de funciones de pertenencia a optimizar. En la tabla 4.1 se presentan los valores medios obtenidos junto con la desviación típica de cada distribución. Asimismo se muestran los valores mínimos, los obtenidos por el algoritmo completo y los obtenidos partiendo de una configuración equidistribuida de funciones de

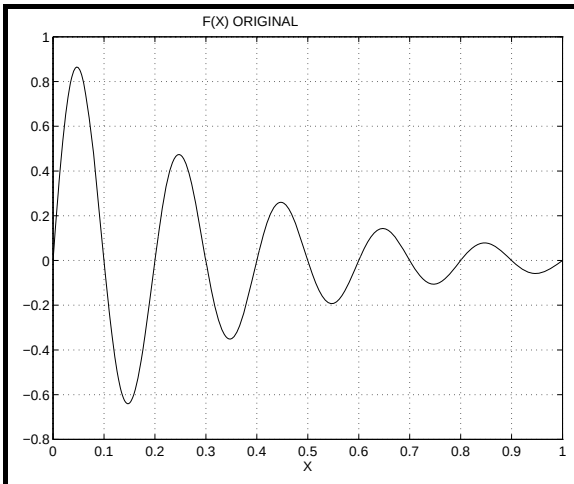


Fig. 4.10 Función f(x) original.

Nº MFs	ECMN (medio)	Dispersión	ECMN (mínimo)	ECMN (equidistr.)	ECMN (algoritmo)
3	0.869	0.048	0.845	0.945	0.845
4	0.855	0.064	0.728	0.940	0.730
5	0.822	0.110	0.539	0.832	0.542
6	0.796	0.108	0.523	0.818	0.412
7	0.723	0.140	0.303	0.696	0.304
8	0.701	0.163	0.297	0.615	0.237
9	0.617	0.205	0.200	0.541	0.182
10	0.534	0.204	0.154	0.507	0.153
11	0.581	0.222	0.121	0.507	0.126
12	0.455	0.215	0.110	0.114	0.114

Tabla 4.1 Estudio de la evolución del ECMN según el punto inicial donde se realice el proceso de descenso en gradiente.

pertenencia. Estos datos se representan gráficamente en la figura 4.11b donde se ha marcado con un '*' los mínimos alcanzados por el algoritmo completo, con '×' los obtenidos a partir de una configuración inicial equidistribuida y con una línea continua los mejores valores de cada una de las 100 muestras para cada configuración.

Varias son las conclusiones que se pueden destacar de estos resultados:

1. El utilizar como configuración inicial funciones de pertenencia equidistribuidas en todo el rango de entrada no es una buena solución. Salvo en el caso de 12 funciones de pertenencia que, por casualidad, resulta ser un buen punto de partida, en el resto de los casos es una solución prácticamente equivalente a una inicialización aleatoria.

2. Ni utilizando un número razonablemente grande de configuraciones aleatorias iniciales se puede asegurar que una de ellas sea un punto de partida para encontrar el mínimo global. En varios casos, la aproximación encontrada por el algoritmo completo (con etapa inicial), resulta ser superior que la mejor de las búsquedas aleatorias.

3. Usando la etapa inicial propuesta en este algoritmo siempre conseguimos una aproximación parecida a la mejor de las aproximaciones aleatorias, e incluso a veces mejor. Sin embargo, hay configuraciones para las que la mejor aproximación

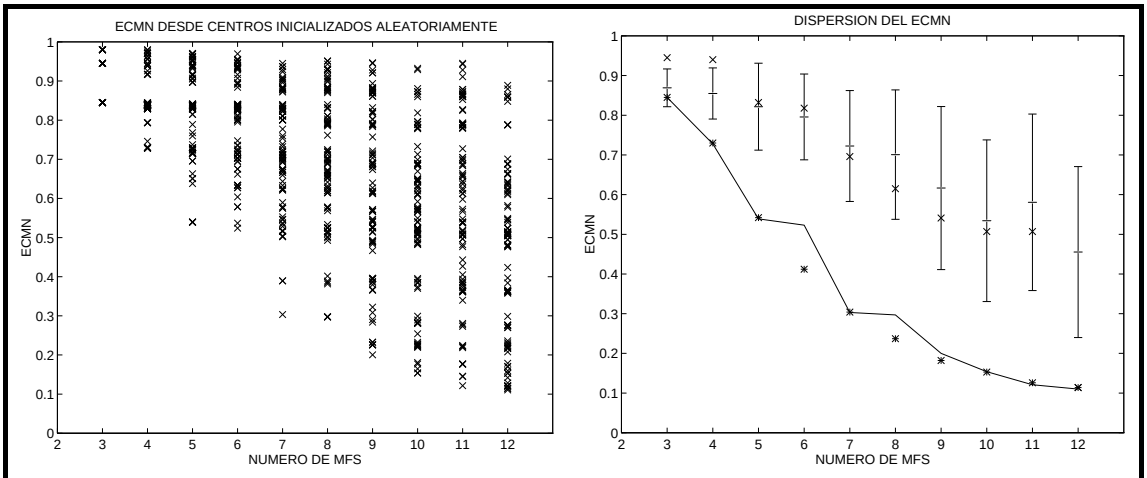
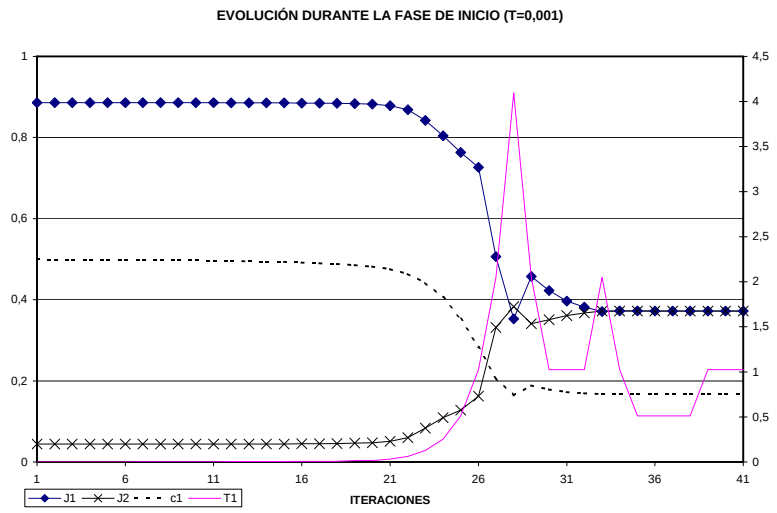


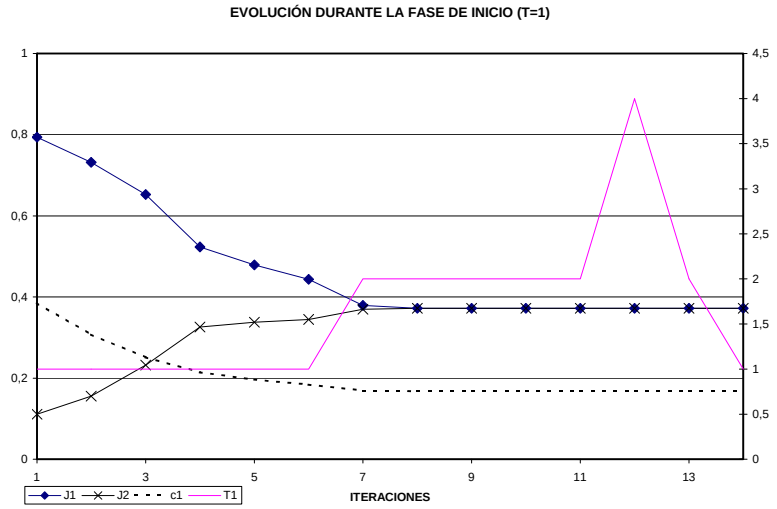
Fig. 4.11 a) ECMN obtenido al realizar un descenso en gradiente desde 100 configuraciones distintas aleatorias para cada número de funciones de pertenencia. b) Comparación de los ECMN generados por el método anterior con los obtenidos a partir de una distribución equidistante y con el algoritmo propuesto.

aleatoria es un poco superior a la obtenida por el algoritmo. Como ya se dijo en el capítulo anterior, el algoritmo no puede asegurar el encontrar el mínimo global pero siempre encontrará un "buen" mínimo.

Finalmente, veamos cómo funcionan las expresiones 3-25 y 3-26 que son la que se utiliza para encontrar una configuración inicial "adecuada" en el algoritmo (ver capítulo anterior). Recordemos que la intención de tales expresiones es encontrar aquella configuración que iguale los errores cuadráticos en todas las zonas que delimitan los centros de las funciones de pertenencia. Para ello, cada centro tenía asociada una pendiente dada por la diferencia entre la contribución al error del tramo que le precede y la del tramo que le sigue. Igualmente se le asociaba un parámetro T que se iba adaptando dinámicamente según la evolución de su centro.

En la figura 4.12 se representa el caso más sencillo posible de una función unidimensional (ecuación (4-7)) aproximada con 3 funciones de pertenencia, utilizando distintos valores iniciales del parámetro T. Al estar los extremos fijos a los valores mínimo y máximo del dominio de la variable, sólo un centro ha de ser relocalizado. En todos los casos, el parámetro T se va actualizando dinámicamente para acelerar la velocidad de convergencia y consiguen igualar los errores cuadráticos situando el centro en la misma posición. La única diferencia existente está en el número de iteraciones necesario para llegar a la situación de equilibrio. Para valores intermedios y altos del valor inicial de T, éste número es bastante bajo (14 iteraciones para $T_{inicial}$ = 1.0 y 20 para $T_{inicial}$ = 500) mientras que se necesitan 41 iteraciones para $T_{inicial}$ = 0.001. Esto es debido a que dicho valor es excesivamente pequeño y se necesitan bastantes iteraciones para que alcance los valores adecuados. En el algoritmo, para dividir el valor de T por 2 sólo hace falta que las pendientes cambien de signo entre dos iteraciones consecutivas mientras que para que se produzca el efecto contrario se necesitan que las pendientes conserven su signo durante un cierto número de iteraciones que también varía dinámicamente según el número de cambios de signo pero cuyo valor mínimo es 4. En la figura 4.13 se representa la misma evolución, con 7 funciones de pertenencia, para valores de T medios y grandes. El número de iteraciones es de nuevo muy similar para ambos procesos. Como conclusión:





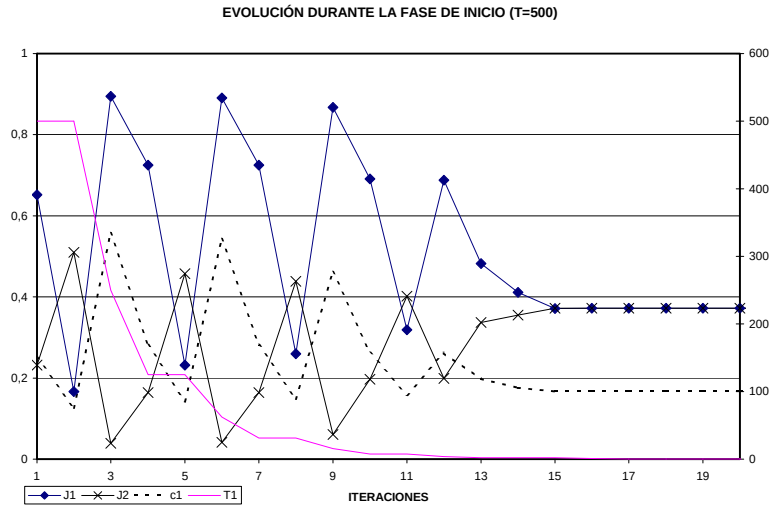


Fig. 4.12 Evolución de la fase de inicio para 3 funciones de pertenencia (los valores de T se representan en el eje secundario). a) $T_{inicial}$ = 0.001 b) $T_{inicial}$ = 1 c) $T_{inicial}$ = 500.

1. La configuración a la que se llega durante la etapa inicial al descenso en gradiente siempre es la misma independientemente del valor inicial de los parámetros *T* asociados a cada centro.

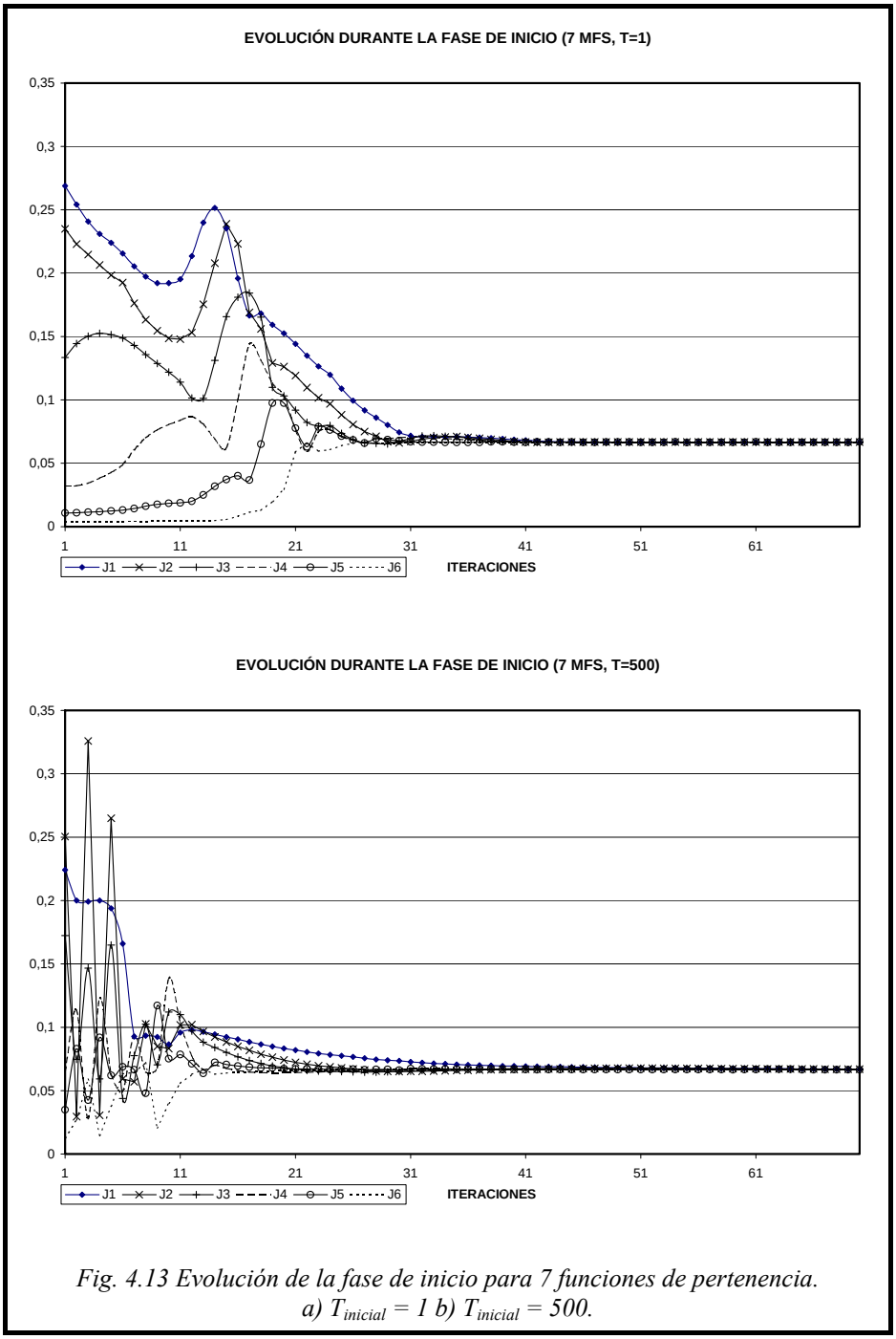


Fig. 4.13 Evolución de la fase de inicio para 7 funciones de pertenencia. a) $T_{inicial} = 1$ b) $T_{inicial} = 500$.

2. El algoritmo es más rápido partiendo de temperaturas elevadas que de temperaturas bajas por lo que, de aquí en adelante, siempre se usará una temperatura inicial alta (*T=500*) para cada centro. Además, debido a la evolución dinámica de las temperaturas, el radio de acción *'b'* de la expresión 3-25 no va a influir en la rapidez

de convergencia y su único propósito es la de asegurar que dos centros no intercambien sus posiciones.

4.2.2.2 Descenso en gradiente

Una vez que la fase previa anterior nos haya proporcionado un punto inicial de búsqueda, el siguiente paso será utilizar un método de minimización para encontrar el mínimo relativo que se encuentre próximo a dicho punto inicial. En la sección 3.4 del capítulo anterior se presentaron dos formas de realizar este proceso. La primera (ecuación 3-31) es un descenso en gradiente convencional con la particularidad de que el factor de aprendizaje se adapta dinámicamente dependiendo de la evolución del error y de si se produce un cambio en la ordenación de los centros. La segunda (ecuación 3-33) se basaba en el método utilizado en la etapa inicial, e intentaba minimizar la derivada del error de cada parámetro individualmente.

Como es sabido, éstos no son los únicos métodos utilizados comúnmente para el proceso de minimización de funciones. Existen métodos teóricamente más rápidos también basados en el gradiente como el “gradiente conjugado”, donde la dirección de búsqueda se obtiene a partir de información del gradiente en el punto actual y de la dirección previa de tal forma que los gradientes sean ortogonales entre ellos (si es posible) y las direcciones conjugadas. La forma de obtener una nueva dirección conjugada se suele realizar principalmente con el método de Fletcher-Reeves [PRE-93]. Una vez obtenida ésta, se debe buscar, en la dirección que indica, el punto donde la función obtiene el mínimo y coger ese punto como el nuevo punto de partida para la siguiente iteración. Existen otros métodos más complejos, basados en el cálculo directo o indirecto del hessiano de la función en cada punto, como los de Newton-Raphson [PRE-93]. Otros, como el de Levenberg-Marquardt [PRE-93], se basan en un gradiente conjugado en las primeras iteraciones (donde realmente no es necesaria mucha precisión y, por tanto, el cálculo del hessiano no es tan importante) conmutando posteriormente, cerca del mínimo, a un método basado en el hessiano.

Cualquiera que sea el método elegido, es evidente que el número de iteraciones necesario para encontrar el mínimo depende fuertemente con el número de parámetros que hay que optimizar. Debido a las características del tipo de funciones (ecuación 3-2)

que usamos para aproximar los datos de E/S, éstas siempre dependen linealmente con los consecuentes de las reglas por lo que, para una configuración fija de funciones de pertenencia, siempre podemos calcular los consecuentes óptimos usando la expresión 3-18. Como se dijo en el apartado 3.4 del capítulo anterior, cada vez que realizamos una iteración del descenso en gradiente, calculamos los consecuentes de las reglas de tal forma que el número de parámetros reales a optimizar disminuye considerablemente. Muchos autores no utilizan esta propiedad y realizan el descenso en gradiente tanto de parámetros como de reglas conjuntamente. Para ver cómo influye esto en el algoritmo aquí propuesto consideremos las funciones ya usadas anteriormente:

$$f(x) = e^{-3x}\, sin(10\pi x) \quad x \in [0,1] \tag{4-8}$$

e

$$y_5(x_1, x_2) = \frac{1}{2}\left[1 + sin(2\pi\, x_1) cos(2\pi\, x_2)\right] \qquad x_1, x_2 \in [0,1] \tag{4-9}$$

y realizaremos dos descensos en gradiente, el primero optimizando conjuntamente centros y reglas y el segundo optimizando solamente centros y calculando los consecuentes óptimos en cada iteración mediante la expresión 3-18. En las tablas 4.2 y 4.3 se presentan los resultados obtenidos, donde se indica entre paréntesis el número de iteraciones necesarias durante el proceso de descenso.

Nº MFs	ECMN centros + consecuentes	ECMN centros (consecuentes óptimos)
3	0.850 (10)	0.845 (8)
4	0.765 (155)	0.730 (53)
5	0.539 (3661)	0.542 (63)
6	0.457 (496)	0.412 (73)
7	0.383 (213)	0.304 (88)
8	0.237 (3554)	0.237 (156)
9	0.182 (2004)	0.182 (162)
10	0.153 (2009)	0.153 (273)
11	0.123 (1948)	0.126 (220)
12	0.114 (2641)	0.114 (338)

Tabla 4.2 Comparación del ECMN obtenido y el número de iteraciones necesario para la función $f(x) = e^{-3x}\, sin(10\pi x)$ optimizando conjuntamente los centros de las MFs y las reglas ó solamente los centros.

Config.	ECMN centros + consecuentes	ECMN centros (consecuentes óptimos)
3x3	0.575 (280)	0.576 (46)
4x4	0.158 (532)	0.158 (80)
5x5	0.123 (3476)	0.122 (158)
6x6	0.056 (1790)	0.056 (101)
7x7	0.039 (8879)	0.039 (145)
8x8	0.028 (10903)	0.028 (121)
9x9	0.022 (21886)	0.022 (193)

Tabla 4.3 Comparación del ECMN obtenido y el número de iteraciones necesario (entre paréntesis) para la función $y_5(x_1,x_2)$ optimizando conjuntamente los centros de las MFs y las reglas ó solamente los centros (obteniendo los consecuentes óptimos en cada iteración).

A la vista de dichas tablas se ve, como cabía esperar, que, aunque el mínimo obtenido es en ambos casos prácticamente el mismo, el optimizar conjuntamente los centros y las

reglas requiere muchísimas más iteraciones que si sólo se optimizaran los centros. En el caso de la configuración 9x9 de la función bidimensional, en el segundo caso tendríamos que optimizar sólo 14 parámetros (7 centros por cada variable ya que los extremos están fijos) mientras que en el primero, el número de parámetros sería 95 (ya que tenemos 81 consecuentes de las reglas). El número de iteraciones necesario es unas 100 veces mayor.

Por otro lado, cabría la posibilidad de pensar que aunque el primer método requiera más iteraciones, cada una de éstas se realiza más rápido ya que no hay que resolver la expresión 3-18 para calcular los consecuentes. En efecto esto es así pero, en el caso de la función unidimensional el tiempo utilizado para optimizar todas las configuraciones conjuntamente es de poco más de 9 minutos en el primer caso y de 52 segundos en el segundo. La diferencia se hace incluso más patente para la función bidimensional (unas 8 horas para el primer caso y poco más de 15 minutos para el segundo). En la figura 4.14 se representan las primeras 500 iteraciones de la evolución del *ECMN* para el caso de 8 funciones de pertenencia al aproximar la función unidimensional. La diferencia es obvia. En el caso de funciones de mayor dimensionalidad, el optimizar conjuntamente los consecuentes y los centros se haría prácticamente inabordable.

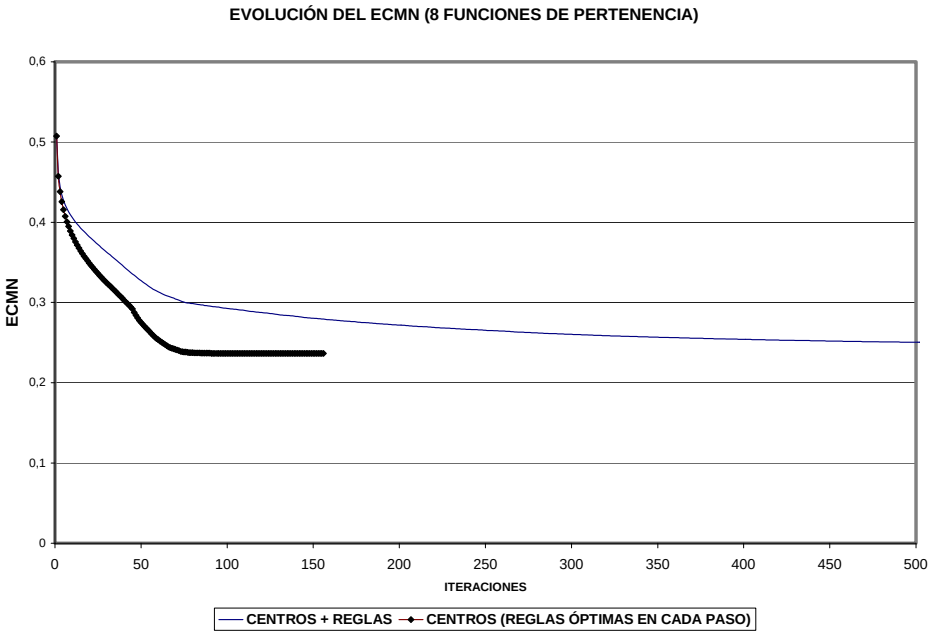


Fig. 4.14 Comparación de la evolución del ECMN para aproximar $f(x)=e^{-3x}\sin(10\pi x)$ con 8 funciones de pertenencia optimizando los centros y los consecuentes conjuntamente ó sólo los centros (calculando los consecuentes óptimos en cada iteración)

Análisis del proceso de minimización

Finalmente, compararemos, para las dos funciones anteriores, los dos métodos de búsqueda del mínimo local presentados en el capítulo anterior para distintos valores iniciales de los respectivos parámetros, junto con el método del gradiente conjugado. En las tablas 4.4 y 4.5 se presentan los valores obtenidos. Entre paréntesis figura el número de iteraciones necesario en cada caso. En el caso del gradiente conjugado se ha añadido también el número de iteraciones total consumido durante el proceso de búsqueda en línea del mínimo, para cada una de las direcciones obtenidas por el algoritmo. De las tablas se puede deducir:

Nº MFs	**η_0= 0.0001**	**η_0= 0.01**	**η_0= 1.0**	**η_0= 100**	**η_0= 10000**	**GC**	**T= 0.001**	**T= 1.0**	**T= 500**
3	0.845 (134)	0.845 (76)	0.845 (8)	0.845 (15)	0.845 (12)	0.845 (7) (142)	0.845 (47)	0.845 (28)	0.845 (39)
4	0.730 (158)	0.730 (110)	0.730 (53)	0.730 (50)	0.730 (34)	0.730 (13) (201)	0.730 (103)	0.730 (69)	0.730 (22)
5	0.542 (157)	0.542 (111)	0.542 (63)	0.730 (50)	0.651 (12)	0.707 (3) (40)	0.714 (107)	0.714 (76)	0.696 (93)
6	0.412 (180)	0.412 (127)	0.412 (73)	0.561 (33)	0.412 (65)	0.406 (40) (612)	0.522 (98)	0.522 (79)	0.522 (153)
7	0.304 (187)	0.304 (142)	0.304 (88)	0.304 (60)	0.309 (60)	0.303 (37) (718)	0.393 (108)	0.393 (71)	0.389 (256)
8	0.237 (263)	0.237 (210)	0.237 (156)	0.237 (124)	0.237 (129)	0.237 (71) (1197)	0.237 (122)	0.237 (103)	0.237 (209)
9	0.182 (273)	0.182 (217)	0.182 (162)	0.182 (127)	0.183 (130)	0.182 (137) (2388)	0.182 (92)	0.182 (93)	0.181 (95)
10	0.153 (362)	0.153 (327)	0.153 (273)	0.154 (251)	0.153 (236)	0.153 (81) (1532)	0.154 (103)	0.155 (76)	0.153 (90)
11	0.126 (333)	0.126 (262)	0.126 (220)	0.121 (165)	0.120 (179)	0.121 (78) (1420)	0.148 (153)	0.148 (65)	0.148 (366)
12	0.114 (399)	0.114 (392)	0.114 (338)	0.113 (270)	0.114 (271)	0.114 (158) (2556)	0.114 (204)	0.114 (108)	0.113 (149)

Tabla 4.4 ECMN y número de iteraciones para los distintos métodos de minimización utilizados y distintos parámetros iniciales. Función $f(x) = e^{-3x} \sin(10\pi x)$.

- Utilizar valores inferiores a $\eta_0 = 1$ no es una buena solución ya que siempre alcanzan el mismo mínimo pero con mayor número de iteraciones. Recordemos que

η_0 = 1 es un valor teórico (ver apartado 5.5.1.1 del capítulo 5) para el cual, en la siguiente iteración, tenemos asegurado que vamos a mejorar el error cometido ya que nos vamos a desplazar por el mínimo relativo más cercano.

Config	**η_0= 0.01**	**η_0= 1.0**	**η_0= 100**	**η_0= 10000**	**GC**	**T= 0.001**	**T= 1.0**	**T= 500**
3x3	0.576 (110)	0.576 (46)	0.575 (29)	0.575 (36)	0.575 (8) (204)	0.576 (109)	0.576 (65)	0.575 (134)
4x4	0.158 (138)	0.158 (80)	0.158 (22)	0.158 (30)	0.158 (25) (463)	0.158 (128)	0.158 (109)	0.158 (156)
5x5	0.122 (204)	0.122 (158)	0.122 (122)	0.122 (127)	0.122 (42) (996)	0.131 (358)	0.130 (415)	0.131 (369)
6x6	0.056 (153)	0.056 (101)	0.056 (42)	0.056 (44)	0.056 (22) (516)	0.058 (237)	0.060 (134)	0.064 (21)
7x7	0.039 (196)	0.039 (145)	0.039 (92)	0.039 (87)	0.039 (54) (1278)	0.045 (277)	0.045 (250)	0.045 (148)
8x8	0.028 (179)	0.028 (121)	0.028 (66)	0.028 (72)	0.028 (24) (804)	0.033 (239)	0.033 (300)	0.032 (432)
9x9	0.022 (238)	0.022 (193)	0.022 (169)	0.022 (158)	0.022 (93) (2709)	0.029 (830)	0.029 (774)	0.030 (23)

Tabla 4.5 ECMN y número de iteraciones para los distintos métodos de minimización utilizados y distintos parámetros iniciales. Función $y_5(x_1,x_2)$.

- El usar valores mayores de η_0 nos asegura un número de iteraciones menor para hallar el mínimo pero no siempre se trata del que queremos buscar (el más cercano). Puede suceder, como en el caso de 11 funciones de pertenencia para la función unidimensional, que el ECMN sea ligeramente mejor pero, como el caso de 5 y 7 funciones, también puede suceder que se empeore. En cualquier caso, la diferencia no es grande, salvo excepciones, debido a que adaptamos dinámicamente el factor de aprendizaje. Como ya se dijo, si el error empeora o se cambia el orden de los centros se vuelve a la configuración original y se disminuye dicho factor. De esta forma, realmente es engañoso el usar factores iniciales muy altos ya que inmediatamente disminuyen hasta los valores realmente efectivos. En la figura 4.15 se observa esto para el caso de una configuración 5x5 para la función y_5. Prácticamente nada más comenzar, los factores decrecen rápidamente hasta valores en torno a la unidad. Hay que hacer notar que cada vez que se va en dirección

opuesta al gradiente se cuenta como iteración aquella en la que finalmente el factor de aprendizaje es el adecuado para disminuir el error cuadrático sin cambiar el orden de los centros. Sopesando los pros y los contras, un valor de η_0 inicial adecuado es 1.0 ya que nos asegura un salto dentro de la región inicial y el número de iteraciones es razonablemente bajo.

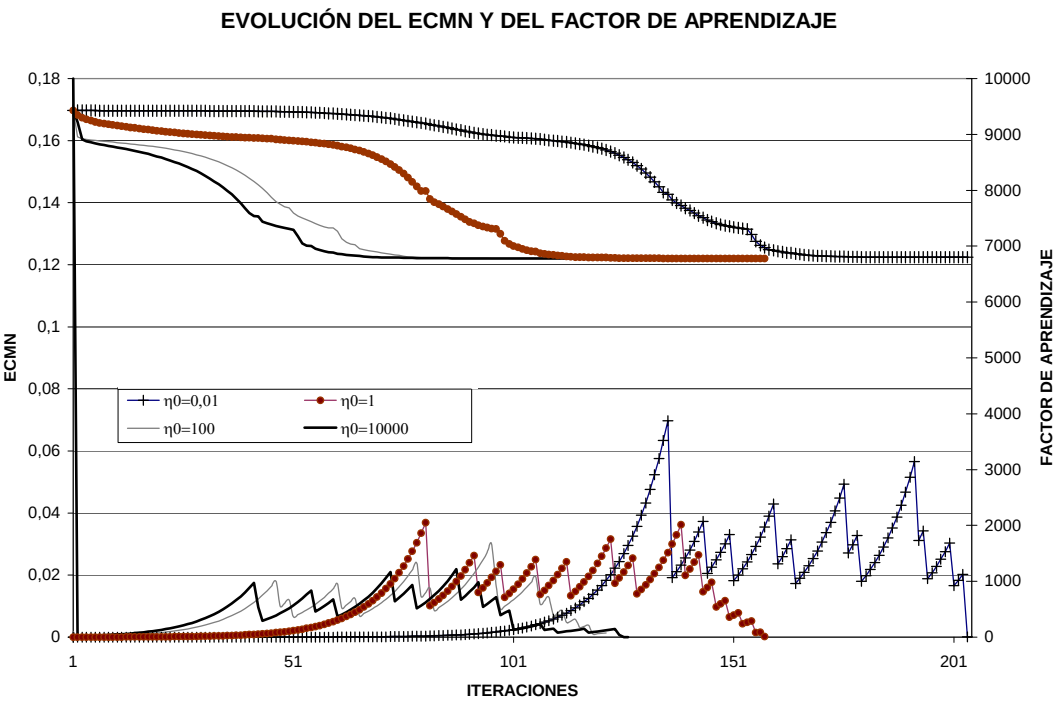


Fig. 4.15 Evolución del ECMN y del factor de aprendizaje para la configuración 5x5 de la función y_5 para distintos valores de η_0.

- Utilizar un descenso en gradiente minimizando individualmente las parciales de cada parámetro nos proporciona valores finales generalmente peores que los obtenidos utilizando el gradiente globalmente como vector, independientemente del valor inicial de las temperaturas asociadas a cada parámetro.

- El gradiente conjugado obtiene generalmente unos mínimos relativos equivalentes al gradiente normal utilizando menos evaluaciones del gradiente. Sin embargo, presenta dos inconvenientes:

  1. Como ya se ha comentado, para cada evaluación del gradiente, el método de Fletcher-Reeves proporciona una dirección conjugada a las anteriores por la que

desplazarse. Dentro de dicha dirección, se debe encontrar el punto en la línea que define donde se minimice la función. Esto requiere un proceso de búsqueda y, en cada uno de ellos, se debe evaluar la función para hallar el *ECMN*. Dicho proceso requiere una serie de iteraciones, que son las marcadas en el segundo paréntesis de la columna correspondiente. Así pues, puede parecer engañoso el bajo número de iteraciones principales (cálculo del gradiente). Por ello, para poder comparar realmente ambos métodos debemos calcular el tiempo total que dura el proceso. Para el caso de la función unidimensional, el gradiente conjugado necesita 1 minuto y 25 segundos para realizar la simulación completa (recordemos que para $\eta_0 = 1$ se tardaban 52 segundos). En el caso de la función bidimensional la diferencia comienza a ser notable (45 minutos frente a 15).

2. El segundo inconveniente es aplicable al resto de algoritmos de minimización como los basados en el cálculo del hessiano. El problema que queremos resolver es, en realidad, un problema de minimización con restricciones en los parámetros. Cada vez que uno de estos métodos propone un punto al que movernos no siempre se puede realizar dicho movimiento (es lo que sucede en el caso de 5 funciones de pertenencia para la función unidimensional). En el caso del gradiente conjugado, esto provoca que los gradientes dejen de ser realmente ortogonales por lo que la siguiente dirección ya no es realmente la que debiera y se puede dar el caso incluso de movernos en dirección positiva del gradiente. Adicionalmente, los consecuentes de las reglas se calculan mediante otro proceso no controlado por el algoritmo, por lo que puede influir en su rendimiento y, como ha quedado patente, siempre será más rápido un algoritmo que obtenga los consecuentes mediante la expresión 3-18 que el acarrear con todos los parámetros conjuntamente.

Como conclusión, el método que mejor se ajusta a las características del problema que queremos resolver y lo resuelve de la forma más fiable y rápida es el descenso en gradiente con el factor de aprendizaje ajustado dinámicamente y es el que usaremos en adelante, siendo $\eta_0 = 1.0$.

4.2.3 Adición de nuevas funciones de pertenencia

Una vez optimizados las funciones de pertenencia y los consecuentes de las reglas para una cierta topología, en la sección 3.5 del capítulo anterior se abordaba el problema fundamental de encontrar una nueva topología que mejore el grado de aproximación a los datos de E/S de que disponemos. Para ello, lo que esencialmente se proponía era analizar individualmente el grado de importancia de cada variable fijando las funciones de pertenencia optimizadas obtenidas en el paso anterior y añadiendo un número grande de funciones en el resto de variables. Cada uno de estos subsistemas es aproximado de forma rápida y directa utilizando la expresión 3-18 obteniéndose así un valor J_v que indica el mejor error cuadrático que se podría obtener con la configuración actual de funciones de pertenencia en la variable *v*. La variable con el mayor de dichos valores será la primera candidata a aumentar su complejidad.

Como primer ejemplo para ver como funciona la metodología propuesta, consideremos una función artificial difusa generada por las funciones de pertenencia triangulares y las reglas de la tabla 4.6 y cuya representación gráfica se muestra en la figura 4.16.

	c_1^1 = **0.0**	c_1^2 = **0.2**	c_1^3 = **0.6**	c_1^4 = **1.0**
c_2^1 = **0.0**	0.0	10.0	5.0	20.0
c_2^2 = **0.5**	0.5	14.0	5.5	19.0
c_2^3 = **0.8**	1.5	9.0	7.3	17.0
c_2^4 = **1.5**	1.6	12.0	6.0	18.0
c_2^5 = **2.0**	1.3	12.0	7.0	22.0

Tabla 4.6 Funciones de pertenencia y consecuentes de las reglas con las que se ha generado la función Fuzz1(x_1,x_2).

En la tabla 4.7 se presentan los resultados obtenidos por el algoritmo al analizar 400 muestras recogidas de la función anterior, comenzando con una configuración de dos funciones de pertenencia por variable. En las columnas intermedias se muestra el *ECMN* obtenido por la configuración actual y los valores J_1 y J_2 (normalizados). En la última columna se muestra el *ECMN* que se obtendría si se hubiera escogido la

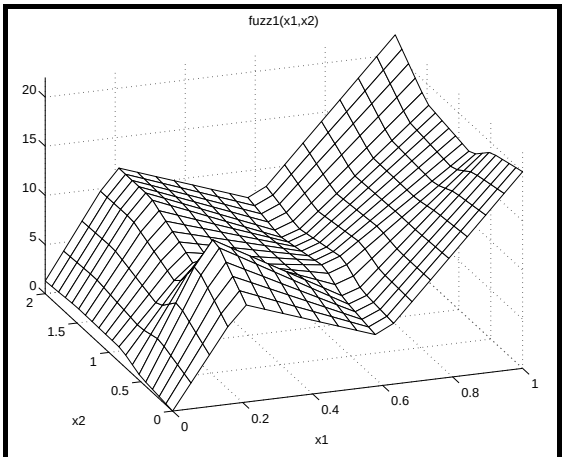


Fig. 4.16 Función Fuzz1(x_1,x_2) original.

configuración contraria a la que indica el algoritmo. Varias son las conclusiones que se pueden obtener de esta tabla:

Config.	**ECMN**	**J_1**	**J_2**	**ECMN caso contrario**
2x2	0.643	0.631	0.164	
3x2	0.509	0.489	0.164	2x3: 0.636
4x2	0.164	$1.84 \cdot 10^{-5}$	0.164	3x3: 0.497
4x3	0.114	$4.03 \cdot 10^{-4}$	0.114	5x2: 0.164
4x4	0.0696	$1.48 \cdot 10^{-3}$	0.0695	5x3: 0.113
4x5	$7.82 \cdot 10^{-4}$	$3.47 \cdot 10^{-5}$	$7.82 \cdot 10^{-4}$	5x4: 0.066

Tabla 4.7 Evolución del ECMN para las distintas configuraciones obtenidas por el algoritmo al aproximar la función Fuzz1(x_1,x_2).

- El algoritmo es capaz de obtener la configuración original con la que se obtuvo la función inicial. Comenzando con tan sólo dos funciones de pertenencia por variable, el método ha sido capaz de encontrar la topología de 4 funciones de pertenencia en la primera variable y 5 en la segunda, obteniéndose un *ECMN* prácticamente cero. En la tabla 4.8 se presenta dicha configuración final obtenida por el algoritmo, donde se aprecia la prácticamente completa similitud con la función original.

	c_1^1 = **0.000**	c_1^2 = **0.200**	c_1^3 = **0.600**	c_1^4 = **1.000**
c_2^1 = **0.000**	0.000	10.000	5.000	20.000
c_2^2 = **0.500**	0.500	14.000	5.500	19.001
c_2^3 = **0.800**	1.501	9.003	7.298	16.996
c_2^4 = **1.505**	1.600	12.013	5.998	18.017
c_2^5 = **2.000**	1.299	11.995	7.004	22.007

Tabla 4.8 Funciones de pertenencia y consecuentes de las reglas para la función Fuzz1(x_1,x_2) aproximada mediante una configuración 4x5.

- La forma de llegar a la configuración final exacta es óptima. Hay muchas maneras de, partiendo de una configuración 2x2, llegar a la configuración 4x5. De entre todas ellas, el algoritmo ha escogido el camino siempre adecuado ya que, como se comprueba en la última columna de la tabla 4.7, siempre el *ECMN* alcanzado por la configuración aconsejada por el algoritmo es menor que el que se obtendría si se realizara la decisión contraria. Analizando la gráfica de la función a aproximar es obvio que, aunque al final se necesiten 5 funciones de pertenencia en la variable 2, en las primeras configuraciones es mejor añadir funciones de pertenencia en la

primera variable ya que ésta es realmente la más importante, pudiéndose considerar, para un cierto grado de aproximación dado, la variable 2 como una variable de ruido. En la figura 4.17a se presenta la función aproximada para una configuración 3x2 donde se aprecia que el primer uso que se hacen de las funciones es para intentar aproximar la parte donde se producen los mayores errores, que es la correspondiente a valores altos de la variable x_1. En la figura 4.17b se muestra la superficie del error para esta configuración, comprobándose que ahora dicha zona ya está bien aproximada y no interferirá para la posterior adición de funciones en dicha variable. Una vez llegadas a las 4 funciones de pertenencia en la variable x_1, el efecto de la segunda variable deja de ser despreciable y por ello entra en consideración por el algoritmo.

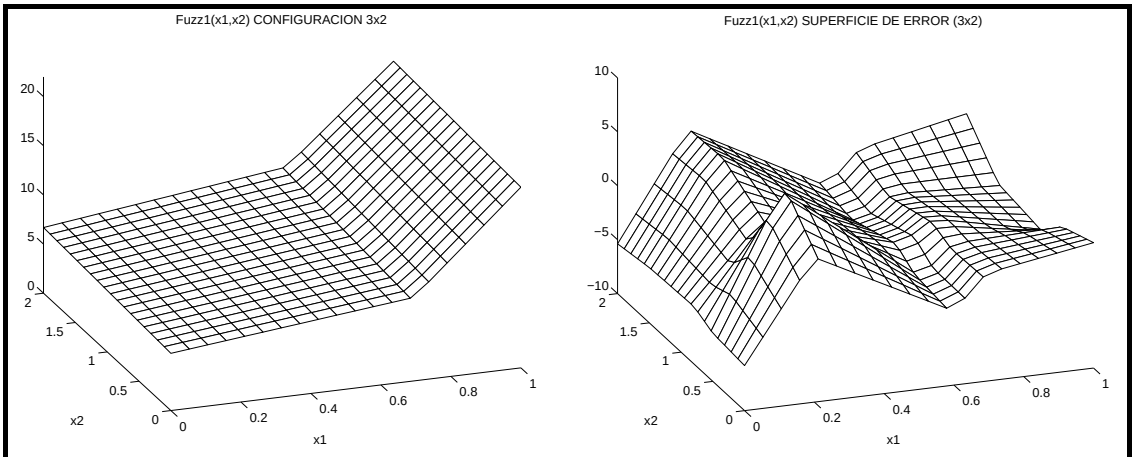


Fig. 4.17 a) Función Fuzz1(x_1,x_2) aproximada mediante una configuración 3x2. b) Superficie del error obtenida.

Al final de la sección 3.5, se abordaba también el problema de situar los nuevos centros de las funciones de pertenencia. Para ello, con la misma distribución con la que se obtuvo el J_v correspondiente, se descompone la variable v en pequeñas subregiones (10 según la ecuación 3-41) y se analiza la distribución del error en cada una de estas subregiones. En la figura 4.18a se representa dicha distribución normalizada para la variable 1, que es la elegida por el algoritmo en el caso inicial de una configuración de 2x2. El valor medio de dicha distribución (obtenido con la ecuación 3-47) es 0.49 y la medida de la dispersión de la misma (ecuación 3-49) es 0.090. El valor límite (10 % del valor máximo) es 0.05. De esta forma, como ya era aparente de la distribución de la figura 4.18a, el valor medio obtenido no es significativo y los centros se equidistribuyen pasando ahora a estar en 0.0, 0.5 y 1.0. En la figura 4.18b se representa la configuración

correspondiente a esta misma variable tras haber analizado la configuración 3x2. Los centros optimizados para este caso están en 0.0, 0.69 y 1.0. En este caso, la distribución está menos diseminada obteniéndose una dispersión de 0.035, inferior al valor límite de 0.05 por lo que el valor medio obtenido de 0.204 es significativo y se sitúa en ese punto el nuevo centro (además es prácticamente el valor óptimo en el que debe situarse).

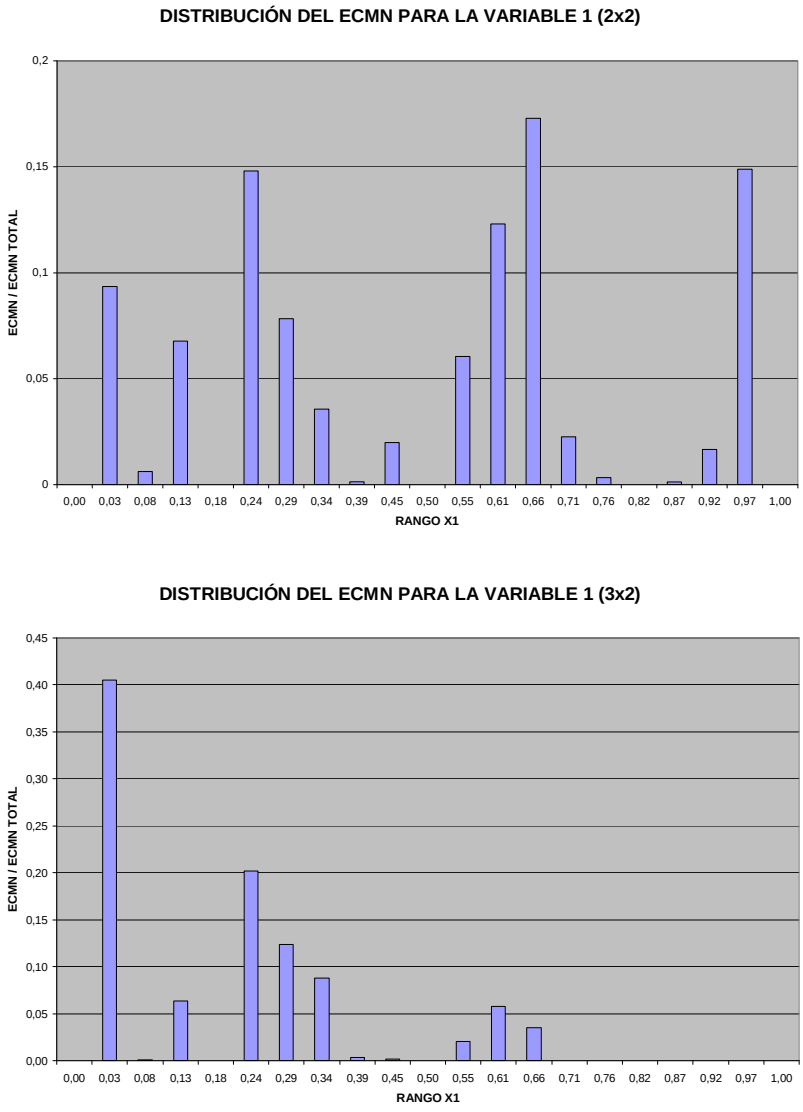


Fig. 4.18 Distribución del ECMN para la variable x_1 al aproximar la función Fuzz1(x_1,x_2) a) mediante una configuración 2x2. b) mediante una configuración 3x2.

Finalmente, en lo que queda de sección y en la siguiente vamos a presentar el funcionamiento de esta etapa con funciones más complicadas. Consideremos de nuevo la función:

$$y_5(x_1, x_2) = \frac{1}{2}\left[1 + sin(2\pi x_1)\cos(2\pi x_2)\right] \qquad x_1, x_2 \in [0,1] \tag{4-10}$$

cuya representación gráfica se mostró en la figura 4.7. En la tabla 4.9 se presenta la evolución del algoritmo donde sólo se ha elegido la variable con mayor error cuadrático medio asociado (es decir ρ = 0.0 en la expresión 3-44). En este caso, se aprecia que el algoritmo escoge la variable más significativa de forma correcta en la mayoría de los casos pero que, sin embargo, cuando los valores de J_v son muy próximos entre sí, puede ocurrir que la decisión no sea del todo la correcta (es lo que sucede, por ejemplo, al elegir la configuración 7x6 sobre la 6x7, entre otras). Esto es debido a que la forma de obtener los valores de los errores cuadráticos medios es realmente una aproximación del caso continuo ideal por lo que habría que tener en cuenta esta falta de precisión al realizar la decisión. Es por ello por lo que se introdujo el parámetro ρ de la expresión 3-44 y, en adelante, utilizaremos un valor de 0.2 indicando que permitiremos también que aquellas variables con un error cuadrático medio asociado dentro del 20 % del valor máximo aumenten su número de funciones de pertenencia.

Config.	ECMN	J_1	J_2	ECMN caso contrario
2x2	0.999	0.673	0.990	
2x3	0.688	0.673	0.128	3x2: 0.998
3x3	0.576	0.558	0.127	2x4: 0.688
4x3	0.168	0.108	0.127	3x4: 0.573
4x4	0.158	0.107	0.116	5x3: 0.157
4x5	0.141	0.104	0.091	5x4: 0.148
5x5	0.122	0.085	0.091	4x6: 0.121
5x6	0.094	0.086	0.034	6x5: 0.099
6x6	0.056	0.039	0.033	5x7: 0.086
7x6	0.052	0.035	0.033	6x7: 0.049
8x6	0.038	0.019	0.032	7x7: 0.039
8x7	0.032	0.022	0.021	9x6: 0.037
9x7	0.029	0.017	0.021	8x8: 0.029
9x8	0.027	0.016	0.018	10x7: 0.027
9x9	0.022	0.015	0.015	10x8: 0.022

Tabla 4.9 Evolución del ECMN para las distintas configuraciones obtenidas por el algoritmo al aproximar la función $y_5(x_1,x_2)$ ($\rho = 0.0$).

4.2.4 Selección automática de las variables más influyentes

Como ya se indicó en el capítulo anterior, la metodología propuesta para la determinación de en qué variable añadir una nueva función de pertenencia funciona para cualquier número inicial de funciones de pertenencia. Esto nos permite asignar

inicialmente una función de pertenencia por variable de entrada de modo que, las variables que permanezcan con dicho número, esencialmente son variables que no intervienen en la aproximación actual. De esta forma, la metodología presentada se puede utilizar para descartar variables en la entrada, con la consiguiente reducción en el número de parámetros necesarios para un cierto grado de aproximación dado.

Para poner de manifiesto esta característica, consideremos la función [DIC-96]:

$$dick(x_1,x_2)=3x_1(x_1-1.0)(x_1-1.9)(x_1+0.7)(x_1+1.8)$$
$$x_1 \in [-2.1,2.1] \quad x_2 \in [0,1] \tag{4-11}$$

donde la variable x_2 es realmente muda (la función es, en realidad, una función unidimensional). La representación gráfica se muestra en la figura 4.19a. Aplicaremos el algoritmo comenzando con la configuración más simple posible, una función de pertenencia para cada variable, es decir, una sola regla que será igual al valor medio de la función obteniéndose, consecuentemente, un *ECMN* igual a la unidad. Para este caso (ver tabla 4.10) tenemos que J_1 = 1.0 y J_2 = $2.39 \cdot 10^{-7}$ indicando de forma obvia que si fijamos la variable x_1 tal como está y ponemos muchas funciones en la variable x_2 vamos a dejar el *ECMN* exactamente como estaba, mientras que si hacemos lo contrario el *ECMN* sería prácticamente nulo. Como consecuencia, el algoritmo coloca la nueva función de pertenencia en la variable 1. En la tabla 4.10 se aprecia que esto sucede en todas las configuraciones por las que pasa el algoritmo por lo que se comprueba que ha detectado perfectamente y de forma automática que la variable x_2 no influye en la mejora del grado de aproximación. En la figura 4.19b se representa la función aproximada utilizando 7 funciones de pertenencia. Hay que reseñar que otros algoritmos que colocan funciones de pertenencia en cada variable hubieran necesitado 49 reglas para poder llegar al mismo grado de aproximación al que la metodología presentada llega utilizando tan solo 7 reglas.

Config.	ECMN	J_1	J_2
1x1	1.000	1.000	2.39E-07
2x1	0.973	0.973	2.39E-07
3x1	0.519	0.519	2.87E-07
4x1	0.258	0.258	1.07E-07
5x1	0.237	0.237	1.78E-07
6x1	0.113	0.113	1.91E-07
7x1	0.043	0.043	5.13E-07
8x1	0.043	0.043	2.55E-07
9x1	0.026	0.026	1.43E-07
10x1	0.017	0.017	3.96E-07
11x1	0.018	0.018	2.56E-07
12x1	0.012	0.012	2.51E-07
13x1	0.015	0.015	3.23E-07
14x1	7.90E-04	7.90E-04	2.33E-07
15x1	1.17E-05	1.17E-05	2.42E-07

Tabla 4.10 Evolución del ECMN para la función $dick(x_1,x_2)$ siendo x_2 una variable muda.

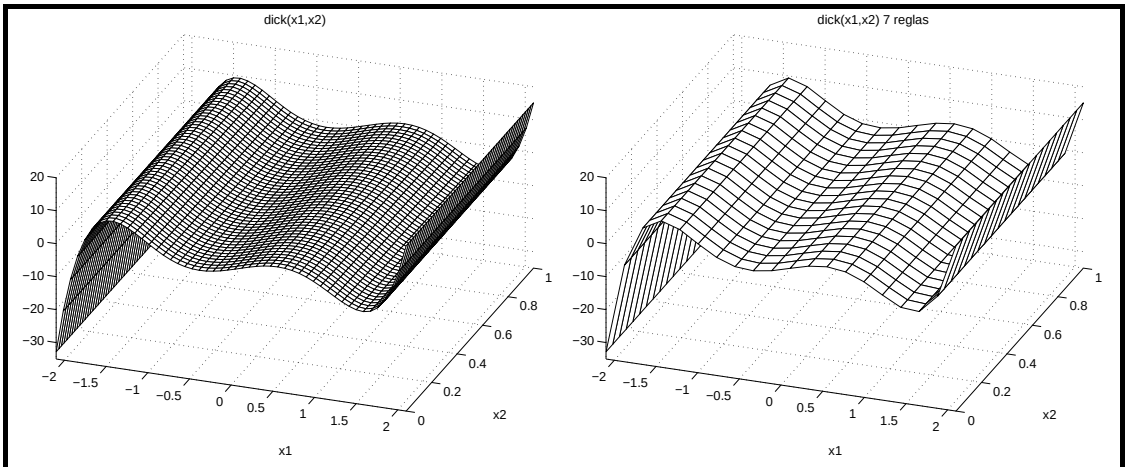


Fig. 4.19 Función dick(x_1,x_2) a) original.b) Aproximada mediante 7 reglas.

Consideremos ahora una función más compleja, adaptada de [FRI-91]:

$$f_9(x_1,x_2,x_3,x_4) = 10sin(\pi x_1 x_2) + 5x_3 + 0x_4 \qquad (4\text{-}12)$$

$$x_1, x_2, x_3, x_4 \in [-1,1]$$

En este caso tenemos una función de cuatro variables de entrada, con una variable muda y otra cuya dependencia es lineal. A la vista de la tabla 4.11 donde se presenta la evolución del algoritmo se comprueba que:

Config.	ECMN	J_1	J_2	J_3	J_4
1x1x1x1	1.000	0.884	0.884	0.446	0.013
2x2x1x1	0.720	0.223	0.225	0.361	0.013
2x2x2x1	0.553	0.390	0.393	0.013	0.013
3x3x2x1	0.469	0.331	0.334	0.016	0.016
4x4x2x1	0.087	0.062	0.062	0.017	0.018
5x5x2x1	0.067	0.049	0.048	0.021	0.022
6x6x2x1	0.033	0.027	0.027	0.021	0.022

Tabla 4.11 Evolución del ECMN para la función $f_9(x_1,x_2,x_3,x_4)$ siendo x_4 una variable muda.

- La variable muda es detectada perfectamente y es descartada por el algoritmo.
- El algoritmo, una vez que la variable x_3 tiene asignadas dos funciones de pertenencia, deja de asociarle un número mayor ya que con una partición triangular bastan dos funciones para describir una dependencia lineal.

De esta forma, para conseguir, por ejemplo, un *ECMN* inferior a 0.1 sólo serían necesarias 32 reglas mientras que un algoritmo que inserte funciones en todas las

variables de entrada hubiera necesitado 256. La diferencia sería incluso más dramática para grados de aproximación mayores.

Finalmente, consideremos la función (ver figura 4.20):

$$f_6(x_1,x_2) = 1.3356 \cdot \left\{1.5(1-x_1) + e^{(2x_1-1)} sin\left(3\pi(x_1 - 0.6)^2\right) + e^{3x_2-0.5} sin\left(4\pi(x_2 - 0.9)^2\right)\right\} \quad (4\text{-}13)$$

$$x_1,x_2 \in [0,1]$$

En este caso ambas variables influyen en la salida, si bien la variable x_1 es la más significativa. En la tabla 4.12 se presenta la evolución del algoritmo para esta función. En dicha tabla se comprueba que sólo a partir de un *ECMN* inferior a 0.5 la segunda variable comienza a hacerse significativa y ser necesaria su utilización. En la última columna se presenta nuevamente el *ECMN* que se obtendría si se escogiera la decisión contraria al algoritmo, verificándose el buen funcionamiento de éste.

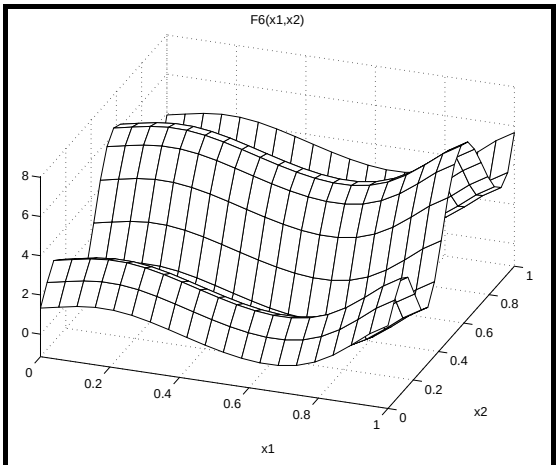


Fig. 4.20 Función $f_6(x_1,x_2)$ original.

Config.	ECMN	J_1	J_2	ECMN caso contrario
1x1	1.000	0.380	0.919	
1x2	0.939	0.380	0.850	2x1: 1.000
1x3	0.783	0.379	0.677	2x2: 0.939
1x4	0.544	0.380	0.375	2x3: 0.783
2x5	0.461	0.376	0.245	
3x5	0.278	0.114	0.245	2x6: 0.394
3x6	0.146	0.113	0.090	4x5: 0.257
4x6	0.104	0.052	0.089	3x7: 0.135
4x7	0.088	0.052	0.068	5x6: 0.098
4x8	0.076	0.052	0.067	5x7: 0.076
4x9	0.062	0.052	0.032	5x8: 0.075
5x9	0.041	0.027	0.031	4x10: 0.062

Tabla 4.12 Evolución del ECMN para la función $f_6(x_1,x_2)$.

4.2.5 Selección de la configuración final

Como se ha comprobado, a lo largo del algoritmo vamos obteniendo una serie de sistemas difusos con distintas complejidades y distintos grados de aproximación. Como se comentó en la sección 3.6 del capítulo anterior, debemos tener en cuenta un cierto

compromiso entre el grado de aproximación y la complejidad del sistema difuso y es realmente el usuario final el que debe decidir según las especificaciones del problema. Para intentar aliviar parcialmente este problema, se propuso utilizar un sistema difuso auxiliar para poder dar una estimación objetiva de la bondad de cada configuración, sopesando la complejidad y la exactitud de la aproximación. Al índice resultante de dicho sistema se le denominó *índice de complejidad/exactitud ICE*.

A modo de ejemplo y, a falta de otra información particular a cada problema, una posibilidad de construir dicho sistema es utilizando una dependencia lineal entre el número de reglas utilizado y el *ECMN* imponiendo como condición, por ejemplo, que el *ECMN* sea menor que un cierto límite dado (aquí escogeremos *ECMN* < 0.5). En la figura 4.21 se representa la función que utilizaremos a lo largo de este capítulo para funciones unidimensionales. Para funciones de mayor dimensionalidad la función será la misma pero poniendo otro límite superior para el número de reglas (debido a la dependencia lineal, realmente se trata exactamente de la misma función pero normalizada de distinta forma).

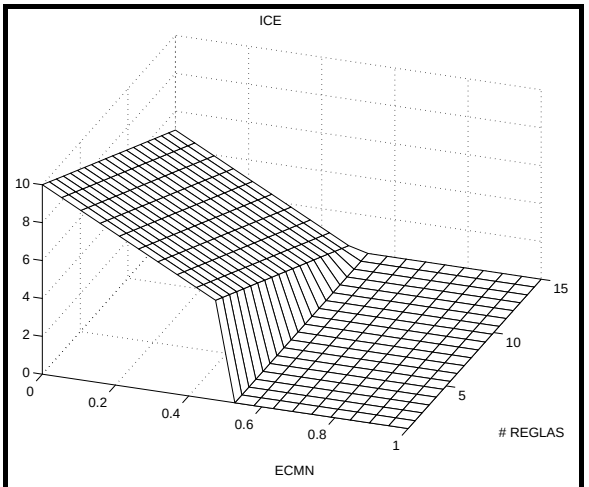


Fig. 4.21 Función que se usará para determinar el índice ICE.

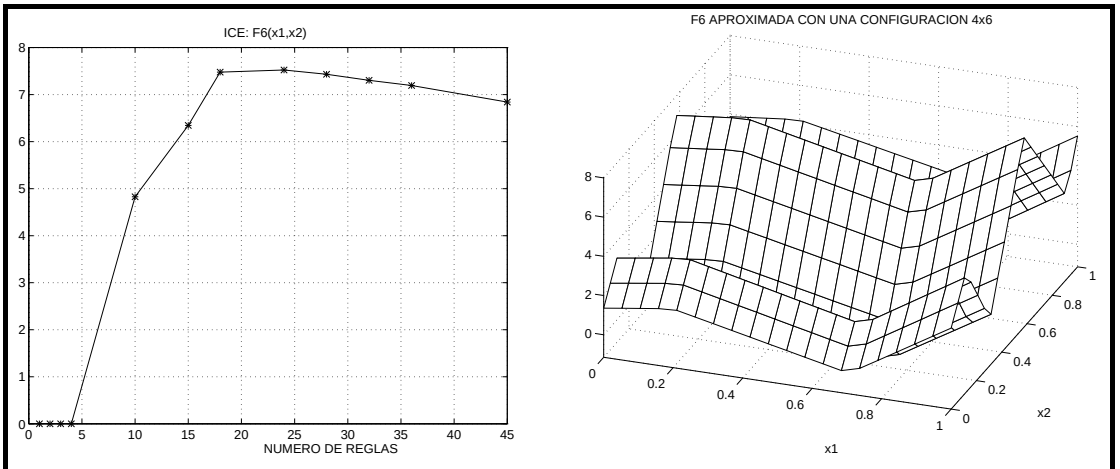


Fig. 4.22 a) Evolución del ICE para la función $f_6(x_1,x_2)$. b) Función $f_6(x_1,x_2)$ aproximada mediante 24 reglas (4x6).

Utilizando este criterio, el índice *ICE* para la función $f_6(x_1,x_2)$ de la ecuación (4-13) (figura 4.20, tabla 4.12) tendría la forma de la figura 4.22a. De esta manera, dicha función alcanzaría un máximo para la configuración 4x6 y ésta sería la que aconsejaría el algoritmo. En la figura 4.22b se representa la función aproximada para dicha configuración.

Finalmente, aunque más adelante veremos cómo se comporta el algoritmo ante datos ruidosos, con el fin de comparar con el criterio *RC* propuesto en [SUG-93] para evitar el sobreajuste en los datos, consideremos de nuevo la función (fig. 4.4)

$$f_2(x)=e^{-5x}\,sin(2\pi x) \qquad x\in[0,1] \tag{4-14}$$

y construyamos dos conjuntos (*A* y *B*) de 20 datos cada uno, a los que añadiremos un gran ruido aditivo del 20 %. En las figuras 4.23a y 4.23b se representan ambos conjuntos.

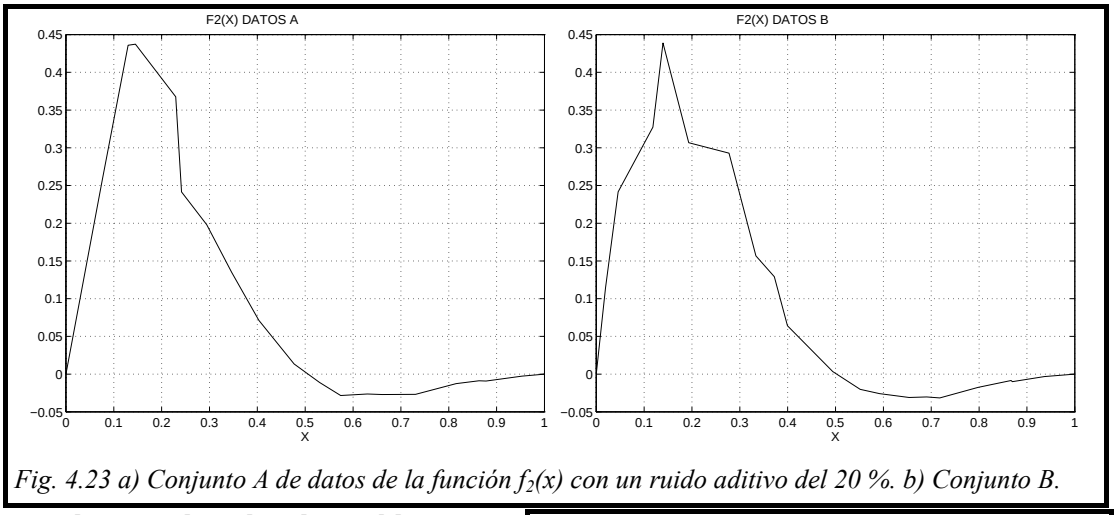


Fig. 4.23 a) Conjunto A de datos de la función $f_2(x)$ con un ruido aditivo del 20 %. b) Conjunto B.

Con la ayuda de la tabla 4.13 obtenemos el valor *RC* normalizado para cada configuración (ver ecuación 3-51). A la vista de los distintos valores de este índice, la configuración elegida sería la de 4 funciones de pertenencia (ver figura 4.24) donde se comprueba que efectivamente dicho método es capaz de eliminar el ruido.

Nº MFs	$ECMN_{AB}$	$ECMN_{BA}$	RC (normalizado)
3	0.646	0.591	0.619
4	0.231	0.160	0.196
5	0.513	0.164	0.339
6	0.271	0.200	0.236
7	0.516	0.262	0.389
8	0.279	0.235	0.257
9	0.427	0.436	0.432
10	0.305	0.394	0.350
11	0.308	0.628	0.468

Tabla 4.13 Cálculo del RC normalizado para la función $f_2(x)$. $ECMN_{AB}$ es el ECMN obtenido utilizando los datos del conjunto A como datos de entrenamiento y los de B como test. Equivalentemente para $ECMN_{BA}$.

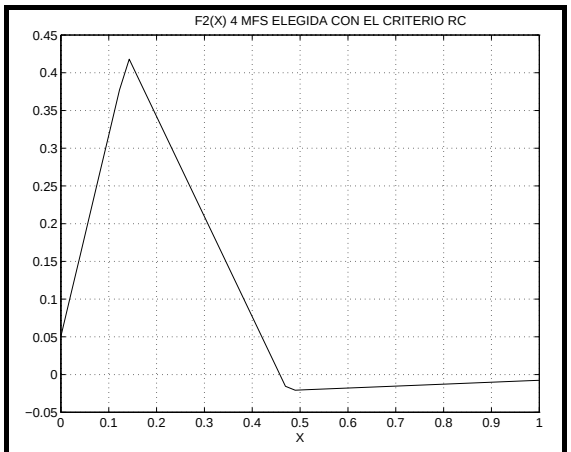


Fig. 4.24 Función $f_2(x)$ aproximada mediante 4 funciones de pertenencia (seleccionada mediante el criterio RC).

Una de las desventajas ya citadas de este método es que no existe ningún criterio para seleccionar los conjuntos *A* y *B*. Adicionalmente, este criterio se utiliza para el caso de tener un número muy reducido de datos ruidosos y realmente sería deseable que se aprovechasen todos los datos disponibles para ser aproximados como un único conjunto. Esto último sí se puede hacer con la metodología aquí utilizada. En la tabla 4.14 se presentan los resultados obtenidos considerando el conjunto completo de datos (figura 4.25). Con el fin de poder contrastar los resultados, en la última columna de la tabla anterior se han evaluado las configuraciones obtenidas utilizando datos exentos de ruido. En las figuras 4.26a y 4.26b se presenta la evolución del *ICE* para ambos casos comprobándose que el algoritmo también es capaz de eliminar el ruido y llegar a las mismas conclusiones que con el criterio *RC*, con la diferencia de que hemos utilizado todos los datos disponibles sin necesidad de repartirlos en dos conjuntos. Gracias a esta característica, el índice final de aproximación para 4 funciones de pertenencia es superior al que se obtendría con el criterio *RC*. En la figura 4.27 se representa la función seleccionada mediante el *ICE*, comprobándose que el ruido también queda eliminado. En la sección 0 volveremos a analizar la forma en la que el *ICE* es capaz de eliminar el sobreajuste de los datos mediante otros ejemplos.

Nº MFs	ECMN (A+B)	ECMN (test)
3	0.590	0.559
4	0.166	0.135
5	0.166	0.135
6	0.149	0.096
7	0.157	0.160
8	0.144	0.168
9	0.133	0.212
10	0.139	0.144
11	0.112	0.125

Tabla 4.14 ECMN (entrenamiento y test) para las distintas configuraciones utilizadas para aproximar conjuntamente los datos A + B de la función $f_2(x)$.

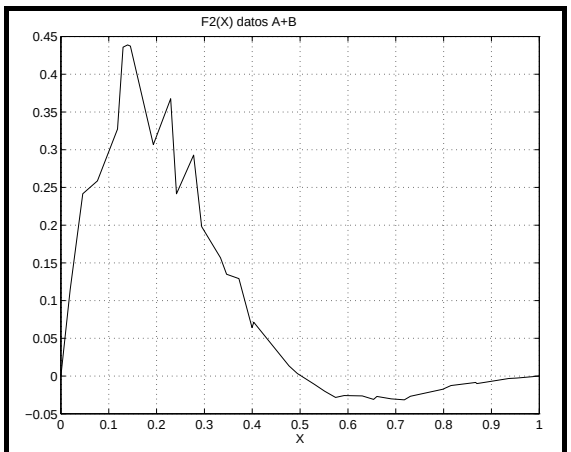


Fig. 4.25 Conjunto A+B de datos de la función $f_2(x)$ con un ruido aditivo del 20 %.

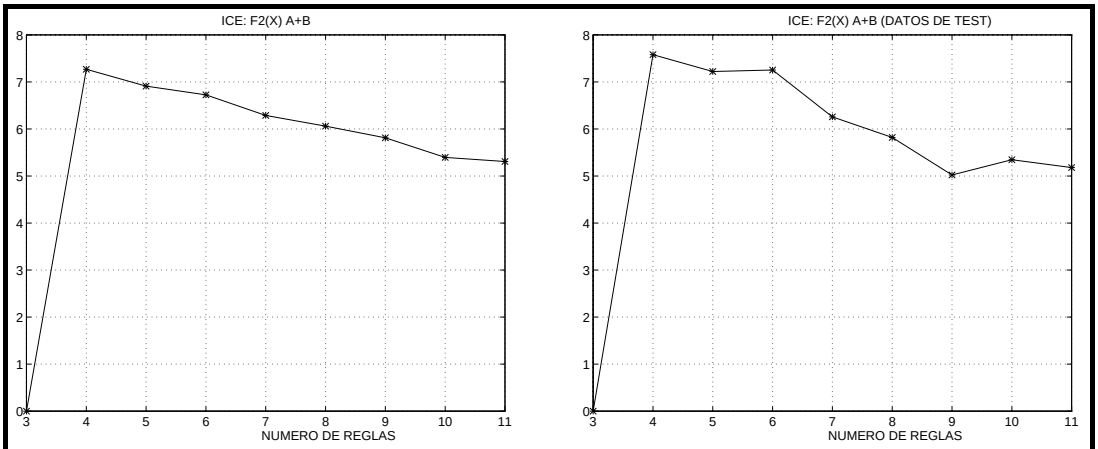


Fig. 4.26 Evolución del ICE para la función $f_2(x)$ a) usando los datos de entrenamiento (A + B) b) usando datos de test carentes de ruido.

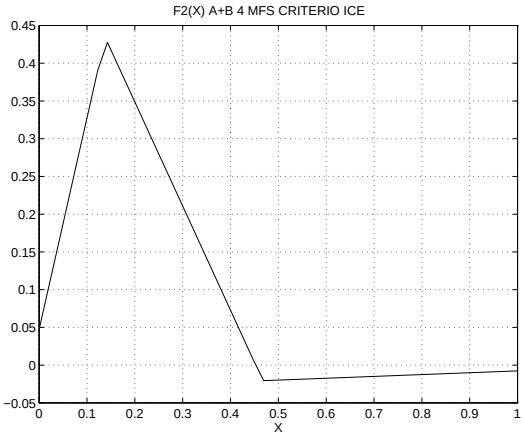


Fig. 4.27 Función $f_2(x)$ aproximada mediante 4 funciones de pertenencia (seleccionada mediante el criterio ICE).

4.2.6 Funciones incompletamente especificadas

Por último, queda comprobar cómo trabaja el algoritmo con funciones incompletas. Consideremos nuevamente la función [ROV-96]

$$y_{5i}(x_1, x_2) = \frac{1}{2}\left[1 + sin(2\pi x_1)cos(2\pi x_2)\right] \tag{4-15}$$

definida en el intervalo [0,1]x[0,1] y quitemos todas las muestras que estén en el rango [0.5,1]x[0.25,1], de tal forma que la función a aproximar sería la representada en la figura 4.28a. Es evidente que el algoritmo va a usar funciones de pertenencia en todo el rango de entrada por lo que va a haber reglas que no se activen. Como ya se dijo en la sección 3.8.1 del capítulo anterior, esto provoca que ciertos elementos de la diagonal de

la matriz dada por la ecuación 3-16 sean nulos (o casi nulos) por lo que el algoritmo eliminará el índice de esas reglas que no se activen, de modo que la matriz tenga un determinante no nulo y se pueda resolver el sistema de ecuaciones de forma exacta, sin indeterminaciones.

En la tabla 4.15 se presenta la evolución del algoritmo para este caso, donde ahora el número de reglas ya no tiene por qué coincidir con el número máximo de ellas. En la tabla 4.16 se muestra el sistema difuso optimizado para la configuración 6x6 comprobándose que existen 6 reglas no activadas por los datos, por lo que quedan marcadas como "no usadas". En la figura 4.28b se representa la superficie de salida para este último caso.

Config.	**Nº de reglas**	**ECMN**
3x3	9	0.395
4x4	15	0.142
5x5	24	0.096
6x6	30	0.052
7x7	41	0.035
8x8	40	0.032

Tabla 4.15 Evolución del ECMN al aproximar la función $y_{5i}(x_1,x_2)$.

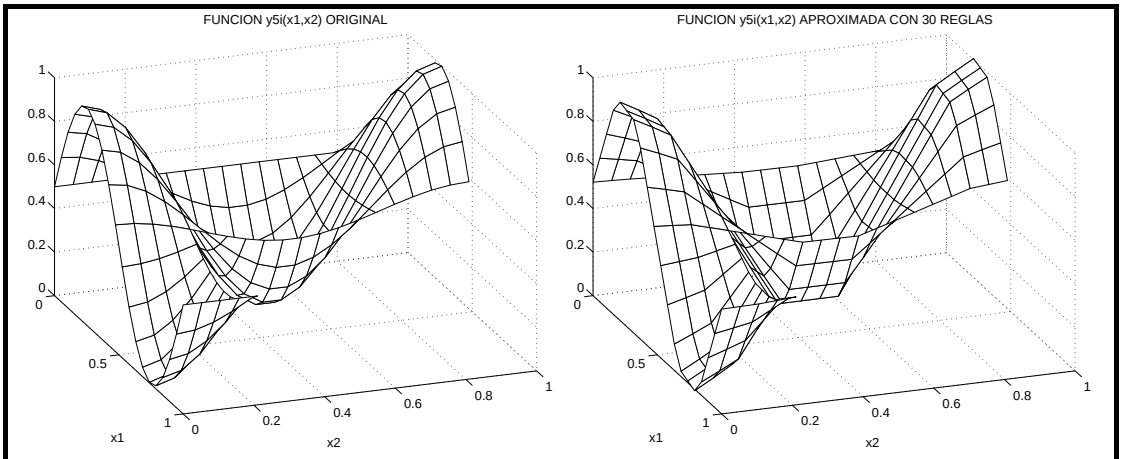


Fig. 4.28 a) Función $y_{5i}(x_1,x_2)$ original. b) Aproximada mediante una configuración de 6x6 (30 reglas usadas realmente)

	c_1^1 = **0.0**	c_1^2 = **0.185**	c_1^3 = **0.335**	c_1^4 = **0.663**	c_1^5 = **0.813**	c_1^6 = **1.0**
c_2^1 = **0.000**	0.521	1.007	0.975	0.030	-0.015	0.481
c_2^2 = **0.104**	0.515	0.915	0.888	0.120	0.080	0.484
c_2^3 = **0.420**	0.483	0.022	0.058	0.911	0.965	0.518
c_2^4 = **0.580**	0.481	0.026	0.059	0.905	No se usa	No se usa
c_2^5 = **0.896**	0.516	0.915	0.883	0.150	No se usa	No se usa
c_2^6 = **1.000**	0.518	1.011	0.971	0.066	No se usa	No se usa

Tabla 4.16 Sistema difuso optimizado para aproximar la función $y_{5i}(x_1,x_2)$ mediante 30 reglas.

En definitiva, el algoritmo, a pesar de considerar conjuntos completos de reglas, es capaz de anular aquéllas que no se activen lo que nos permite, entre otras cosas, tratar con el problema de la predicción de series temporales. La predicción de series temporales es un problema práctico importante. Aplicaciones de este problema se pueden encontrar en áreas como la predicción económica y empresarial, predicción del clima, procesamiento de señales y control, entre muchas otras. Sea x[k] (k = 1, 2, 3...) una serie temporal. El problema de la predicción de series temporales se puede formular de la siguiente manera: Dados x[k-(n-1))], x[k-(n-2))], ..., x[k-)], x[k] debemos determinar x[k+P], donde n,) y P son enteros positivos fijos.

La predicción de series temporales basadas en la ecuación diferencial de Mackey-Glass se suele utilizar como patrón de test en la mayoría de los algoritmos que intentan resolver este tipo de problemas. La serie temporal caótica de Mackey-Glass se genera, esencialmente, a partir de la siguiente ecuación en diferencias:

$$x[k+1] = (1-b)\cdot x[k] + \frac{a\cdot x[k-\tau]}{1+x^{10}[k-\tau]} \qquad (4\text{-}16)$$

Cuando $\vartheta \geq 17$, la ecuación anterior muestra comportamiento caótico. Siguiendo otros estudios previos [WHI-96], generaremos nuestra serie temporal usando a = 0.2, b = 0.1 y $\vartheta = 17$, y escogeremos las entradas x_i, x_{i-6}, x_{i-12} y x_{i-18} para predecir x_{i+6}.

La serie se genera a partir de un sistema no lineal determinista y es suficientemente complicada como para aparentar ser una serie caótica. Sin embargo, realmente no lo es. En la figura 4.29 se muestran los 1000 puntos que usaremos en nuestro estudio. Como suele ser usual en estas situaciones, los primeros 500 datos serán utilizados como datos de entrenamiento mientras que los 500 restantes se usarán para validar el modelo generado.

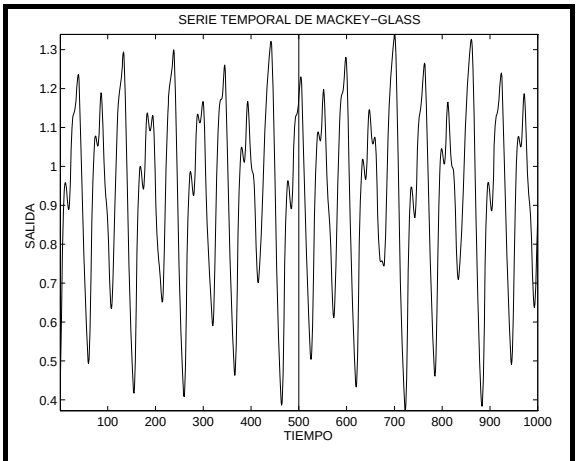


Fig. 4.29 Serie temporal de Mackey-Glass.

En la tabla 4.17 se muestra la evolución del algoritmo para este caso. En la penúltima columna se refleja el *ECMN* para los 500 datos de chequeo comprobándose que, para configuraciones sencillas, la capacidad de generalización es bastante satisfactoria, mientras que para la última configuración presentada el error es mayor en los datos de test que en el caso de la configuración anterior, produciéndose el fenómeno del sobreajuste en los datos de aprendizaje, perdiéndose, por tanto, la capacidad de generalización. En las figuras 4.30a y 4.30b se representa la serie original junto con la aproximada para el caso de una configuración realmente sencilla 2x2x2x3, en la que el grado de aproximación es ya bastante significativo. Si se requiriese un mayor nivel de aproximación, podríamos usar la configuración 4x4x5x5 donde realmente se usan 173 de las 400 reglas posibles (figuras 4.30c y 4.30d) comprobándose las altas prestaciones del algoritmo. En la sección 4.4 compararemos los resultados obtenidos con otros existentes en la bibliografía.

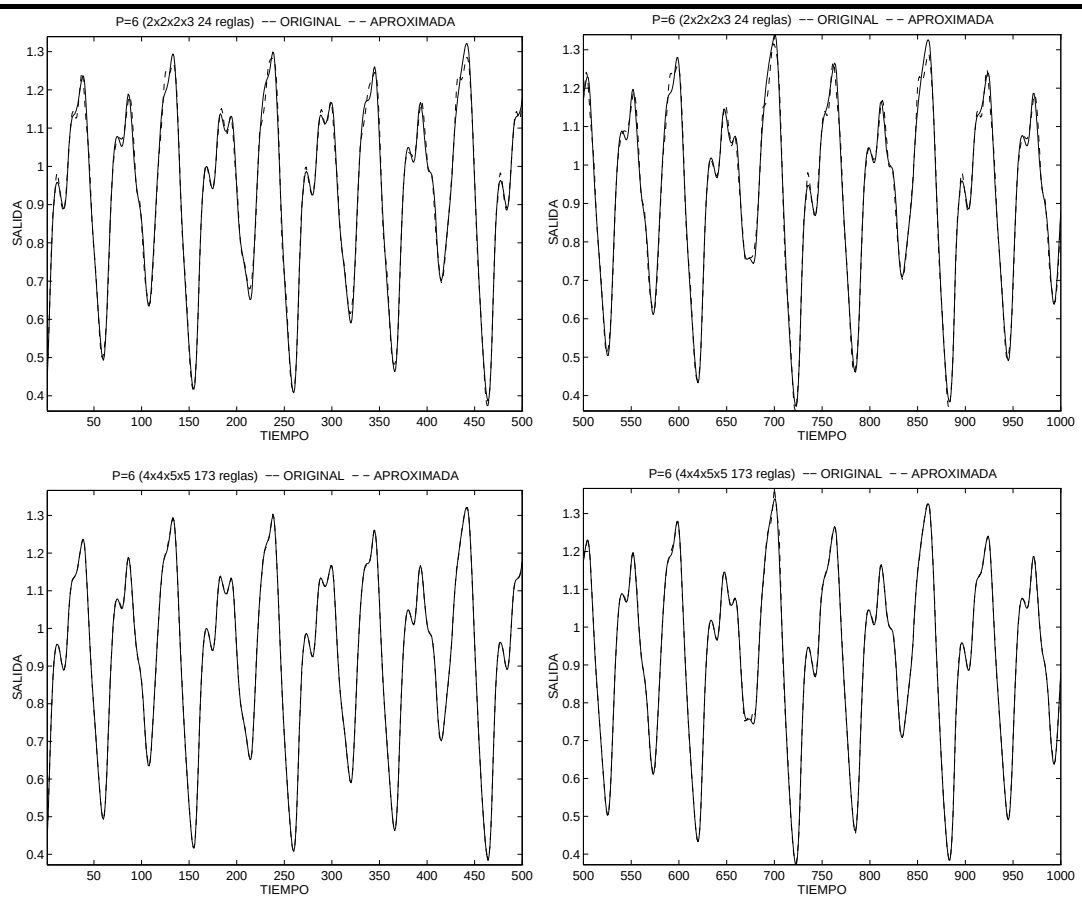


Fig. 4.30 a) Aproximación de los datos de entrenamiento de la serie de Mackey-Glass mediante una configuración 2x2x2x3 (24 reglas) b) chequeo para 2x2x2x3. c) Aproximación de los datos de entrenamiento de la serie de Mackey-Glass mediante una configuración 4x4x5x5 (173 reglas) d) chequeo para 4x4x5x5.

Config.	Nº de reglas	ECMN (entren.)	ECMN (chequeo)	ECMN (todos)
1x1x1x1	1	1.000	1.000	1.000
1x1x2x1	2	0.880	0.873	0.876
1x1x2x2	4	0.428	0.428	0.428
2x2x2x3	24	0.074	0.079	0.077
2x3x3x3	48	0.036	0.046	0.042
3x4x4x4	109	0.022	0.030	0.026
4x4x5x5	173	0.011	0.026	0.021
5x5x5x6	242	0.008	0.044	0.032

Tabla 4.17 Evolución del ECMN al aproximar la serie temporal de Mackey-Glass. Los primeros 500 datos se consideran de entrenamiento y los 500 siguientes de chequeo.

4.2.7 Efecto del ruido

Para muchas tareas es de gran interés saber cómo manejar una cierta cantidad de ruido en los datos que queremos aproximar. Cuando los datos se recogen de un sistema real, los sensores suelen añadir pequeñas perturbaciones que quedan incluidas en los propios datos. Para simular este efecto, añadiremos distintas cantidades de ruido a nuestras muestras y comprobaremos cómo, debido a que el algoritmo propuesto comienza con configuraciones sencillas, a que minimiza el error de forma global y a que la etapa de adición de funciones de pertenencia considera la superficie del error total y no aquella con el mayor error de aproximación, éste es capaz de tratar con datos ruidosos de forma eficaz.

En la sección 4.2.5 ya se realizó un breve estudio de cómo el índice ICE puede ayudar a seleccionar una configuración que elimine el ruido de los datos de Entrada/Salida. En esta sección analizaremos el efecto del ruido desde otra perspectiva. Consideremos para ello la función [CHE-96] (ver figura 4.31a):

$$F_1(x_1, x_2) = \sin(x_1 x_2) \qquad x_1, x_2 \in [-2, 2] \tag{4-17}$$

cuyo rango de salida es el intervalo [-1,1]. Hemos añadido a los datos de entrenamiento un error aditivo blanco del 1%, es decir, sumamos a cada muestra un valor aleatorio entre -0.01 y 0.01. Posteriormente hacemos esto mismo para un ruido 10 veces mayor (10%). En las figuras 4.31b y 4.31c se representan ambos conjuntos de datos de entrenamiento.

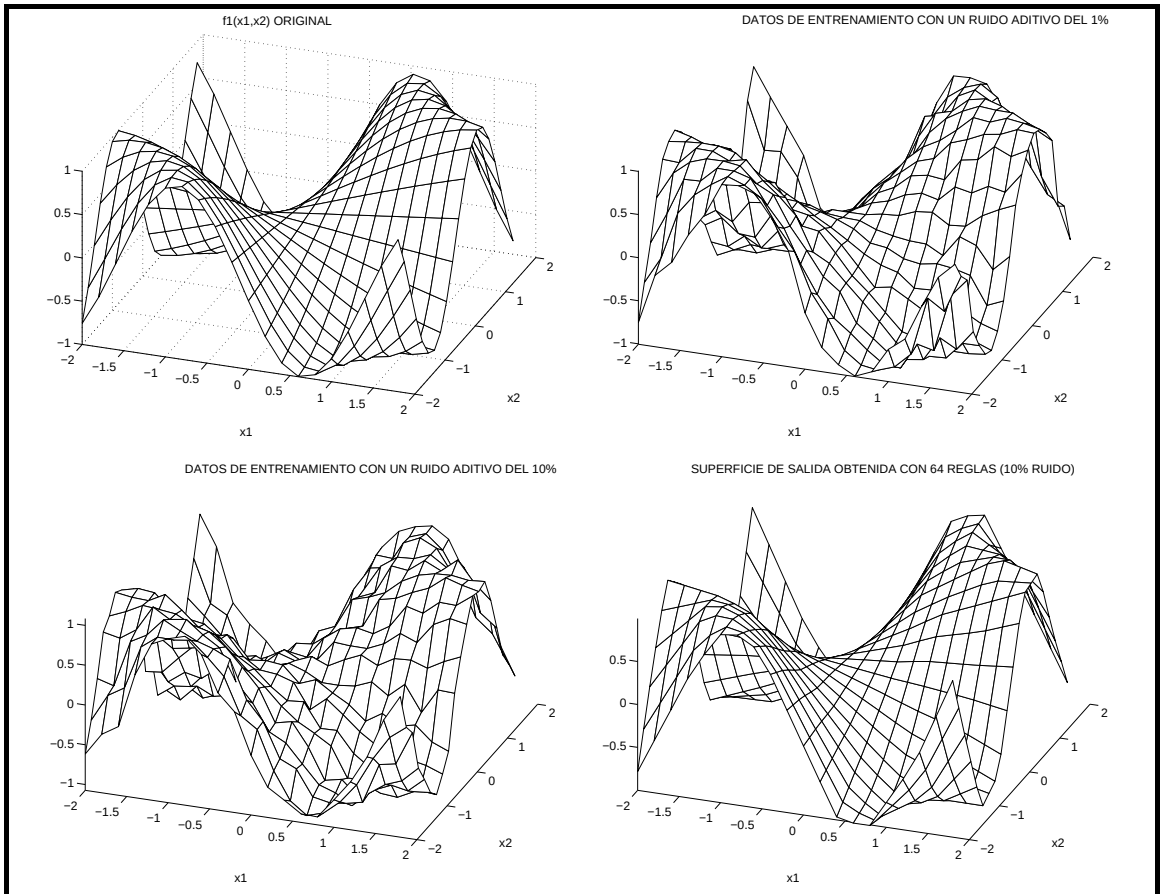


Fig. 4.31 a) Función $f_1(x_1,x_2)$ original. b) Datos de entrenamiento de la función $f_1(x_1,x_2)$ con ruido aditivo del 1%. c) Datos de entrenamiento para ruido aditivo del 10%. d) Superficie de salida obtenida por el algoritmo con 64 reglas después de analizar datos con ruido aditivo del 10%. ECMN = 0.0442.

Los resultados de la aplicación del algoritmo a estos dos casos, se presentan en la tabla 4.18, donde se ha utilizado un conjunto de 400 datos equidistribuidos exentos de ruido para verificar cada una de las configuraciones obtenidas. A la vista de los resultados, se comprueba que un pequeño ruido apenas afecta a las prestaciones del sistema ya que los resultados son prácticamente iguales a los producidos al aproximar datos aleatorios carentes de ruido. Más importante aún, para grandes niveles de ruido y configuraciones complejas, el algoritmo consigue mejores grados de aproximación con los datos de test que con los ruidosos datos de entrenamiento verificándose cómo, en cierta medida, éste ha sido capaz de filtrar el ruido existente. La figura 4.31d muestra la superficie de salida obtenida para una configuración 8x8 entrenada con ruido del 10 % en la que, aunque el ECMN es algo mayor que para el caso de haber sido entrenado sin ruido, la forma original de la función queda completamente recuperada. Como es lógico pensar, a mayor complejidad del sistema utilizado, mayor será la importancia del ruido ya que

mayor precisión se le exigirá al sistema. Sin embargo, al existir ruido en todos los datos de E/S, el algoritmo nunca aproximará los datos individualmente (a no ser que haya un número de reglas equivalente al de datos).

Funciones de pertenencia		**ECMN (sin ruido)**		**ECMN (ruido 1%)**		**ECMN (ruido 10%)**	
x_1	x_2	entren.	test	entren.	test	entren.	test
2	2	0.755	0.698	0.758	0.700	0.759	0.699
3	3	0.634	0.590	0.646	0.590	0.645	0.591
4	4	0.121	0.117	0.119	0.117	0.146	0.118
5	5	0.0967	0.0942	0.0922	0.0941	0.128	0.0982
6	6	0.0469	0.0505	0.0507	0.0518	0.0989	0.0583
7	7	0.0351	0.0404	0.0438	0.0459	0.0925	0.0544
8	8	0.0267	0.0305	0.0287	0.0302	0.0859	0.0442
9	9	0.0256	0.0295	0.0252	0.0285	0.0821	0.0438

Tabla 4.18 Evolución de ECMN ante diferentes niveles de ruido en los datos de entrenamiento.

Conclusiones análogas se pueden obtener utilizando otro tipo de funciones. Como segundo ejemplo usaremos la función [CHE-96] (ver figura 4.32):

$$f_7(x_1, x_2) = 1.9 \cdot \left[1.35 + \exp(x_1)\sin(13(x_1 - 0.6)^2)\exp(-x_2)\sin(7x_2)\right] \tag{4-18}$$

$$x_1, x_2 \in [0,1]$$

de la que hemos extraído 400 datos con un ruido del 10 % como datos de entrenamiento (ver figura 4.33a). En la figura 4.33b se representa la salida de la función aproximada mediante una configuración 6x8 (48 reglas). Como sucedía en el caso anterior, el ECMN es menor para los datos de test (0.099) que para los de entrenamiento (0.154) y, nuevamente, la forma principal de la función queda recuperada.

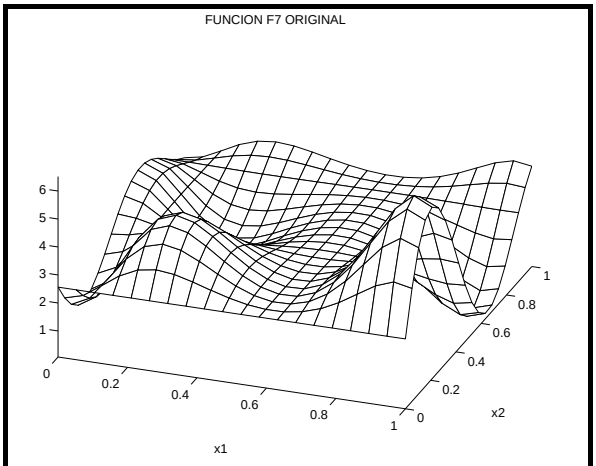


Fig. 4.32 Función $f_7(x_1,x_2)$ original

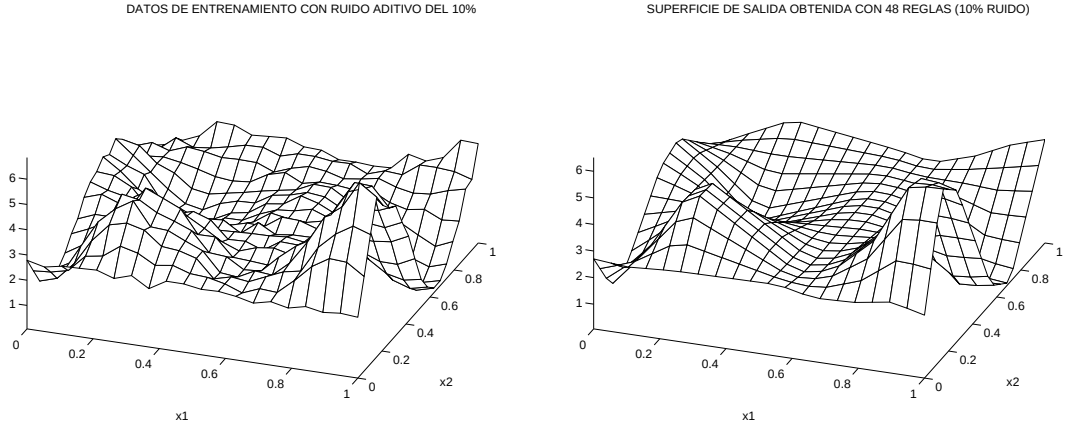


Fig. 4.33 *(a) Datos de entrenamiento de la función $f_7(x_1,x_2)$ con ruido aditivo del 10%. (b) Superficie de salida obtenida por el algoritmo con 48 reglas después de analizar datos con ruido aditivo del 10%. ECMN (entrenamiento) = 0.154, ECMN (test) = 0.099.*

4.2.8 Efecto del número de datos de E/S

Una cuestión importante cuando hablamos de aproximación de funciones a partir de un conjunto de datos de E/S no es sólo la calidad de dichos datos (ya abordado en la anterior sección) sino también su cantidad. Como es obvio, cuanto mayor sea el número de datos que utilicemos, mejor estará representada la función original que queremos aproximar. La cantidad de datos necesarios dependerá, lógicamente, de la propia función. Si ésta contiene cambios bruscos, deberá ser muestreada con mayor precisión. Por otro lado, hay que tener cuidado con el problema del sobreajuste, es decir, no aproximar excesivamente los datos de entrenamiento de manera que el sistema final pierda la capacidad de generalización. En este capítulo se ha elegido siempre un número grande de datos de E/S, dependiendo éste de la dimensionalidad de la función a aproximar de manera que la función siempre esté adecuadamente representada.

Para realizar una estimación de la influencia del número de datos de entrenamiento en el resultado final, consideremos distintas cantidades de muestras de la función definida por la ecuación (4-17) que ya fue representada en la figura 4.31. En la tabla 4.19 se presentan los resultados obtenidos, donde se han utilizado 1000 datos equidistribuidos como datos de validación.

De la tabla, podemos observar que para configuraciones sencillas el número de datos de entrenamiento no es muy importante. Sin embargo, cuando la complejidad del sistema

aproximador aumenta, el mencionado problema del sobreajuste aparece en aquellos casos de bajo número de datos y la diferencia entre los índices de error entre los datos de entrenamiento y los de validación aumenta rápidamente. Esto da lugar a una falsa impresión de que utilizando pocos datos se consiguen mejores índices de error ya que los datos de test prueban que realmente la aproximación es mucho peor.

Config.		**ECMN (49 datos)**		**ECMN (100 datos)**		**ECMN (225 datos)**		**ECMN (400 datos)**	
x_1	x_2	entren.	test	entren.	test	entren.	test	entren.	test
2	2	0.904	0.798	0.825	0.735	0.791	0.710	0.755	0.698
3	3	0.743	0.639	0.702	0.612	0.669	0.598	0.634	0.590
4	4	0.088	0.141	0.105	0.125	0.124	0.125	0.121	0.117
5	5	0.049	0.130	0.0735	0.102	0.095	0.099	0.097	0.094
6	6	0.018	0.137	0.0435	0.073	0.050	0.056	0.047	0.051
7	7	2.4E-7	0.095	0.0301	0.062	0.044	0.046	0.035	0.040
8	8			0.0064	0.054	0.019	0.035	0.027	0.031
9	9			0.0013	0.049	0.019	0.036	0.026	0.029

Tabla 4.19 Evolución del ECMN ante distintas cantidades de datos de entrenamiento.

Como es obvio, cualquier algoritmo trata de aproximar lo mejor que puede los datos de entrenamiento, que son los únicos a los que tiene acceso, por lo que si se quieren verificar algoritmos de aproximación de funciones, los datos han de ser suficientemente significativos y representativos de la función que intentan describir. Por este motivo se ha usado en este trabajo, siempre que ha sido posible, un número de muestras relativamente elevado.

4.3 Utilización de otros tipos de funciones de pertenencia

En la sección 3.8.2 del capítulo anterior se comentó la posibilidad de adaptar el algoritmo propuesto a otros tipos de funciones de pertenencia distintos de una partición triangular (TP). En este apartado vamos a mostrar algunos ejemplos de aplicación de la metodología utilizada a los 5 tipos de funciones de pertenencia que se definen en el Apéndice A. Para ilustrar las analogías y diferencias entre los distintos tipos de particionamiento del espacio de entrada, utilizaremos una función común propuesta en [CHE-96]:

$$f_4(x_1, x_2) = \frac{1+\sin(2x_1+3x_2)}{3.5+\sin(x_1-x_2)} \qquad x_1,x_2 \in [-2,2] \tag{4-19}$$

cuya representación gráfica se muestra en la figura 4.34 en la que se observa que es una función bastante difícil de aproximar pese a la simplicidad de su expresión analítica.

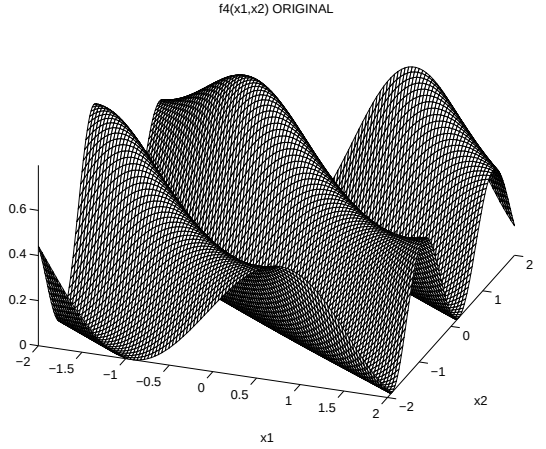


Fig. 4.34 Función $f_4(x_1,x_2)$ original.

Como siempre, seleccionaremos 400 muestras pseudo-aleatorias de la función aproximar. A lo largo de este apartado, sin embargo, representaremos las funciones mediante 10000 puntos de tal forma que se pueda apreciar mejor la función defuzzificada y observar más fácilmente las diferencias entre los distintos tipos de funciones de pertenencia.

En la tabla 4.20 se presentan los resultados obtenidos al aproximar dicha función mediante una partición triangular. En dicha tabla se muestra también el número total de parámetros a optimizar para cada configuración y los errores cuadráticos medios normalizados para los 10000 datos de test. Una primera conclusión a la vista de los resultados es, como ya habíamos anticipado, que la función es realmente compleja para ser aproximada ya que se necesitan más de 94 parámetros para poder alcanzar un *ECMN* por debajo de 0.1. Se extraerán otro tipo de valoraciones cuando se presenten los resultados para otros tipos de funciones de pertenencia más adelante. En la figura 4.35a se presenta la evolución del *Índice de Complejidad-Exactitud* (ICE) para este caso donde se recomienda la configuración de 75 parámetros (7x9) como representativa, siendo su valor en torno a seis[1]. A modo de ejemplo, en las figuras 4.35b y 4.35c se representan las funciones de

Config.	Número de Parámetros	ECMN (entrenamiento)	ECMN (test)
1x1	1	1.000	1.000
2x2	4	0.972	0.969
3x3	11	0.926	0.916
4x4	20	0.744	0.736
4x5	25	0.602	0.594
4x6	30	0.489	0.487
5x7	43	0.281	0.296
6x8	58	0.189	0.196
7x9	75	0.121	0.140
8x10	94	0.084	0.105
9x11	115	0.066	0.085

Tabla 4.20 Evolución del ECMN al aproximar la función $f_4(x_1,x_2)$ mediante una partición triangular.

[1] A lo largo de este apartado utilizaremos exactamente la misma función para obtener el ICE de modo que se puedan comparar globalmente configuraciones de distintas funciones de pertenencia.

pertenencia obtenidas para la configuración 5x7 (como veremos más adelante, esta misma configuración se repite para el resto de funciones de pertenencia, así que será la elegida para poder comparar unos tipos de funciones con otros). En las figuras 4.35d, e y f se representan distintas funciones defuzzificadas para las configuraciones 5x7, 7x9 y 9x11 donde se aprecia la evolución del grado de aproximación observándose claramente los puntos no derivables propios de este tipo de funciones triangulares.

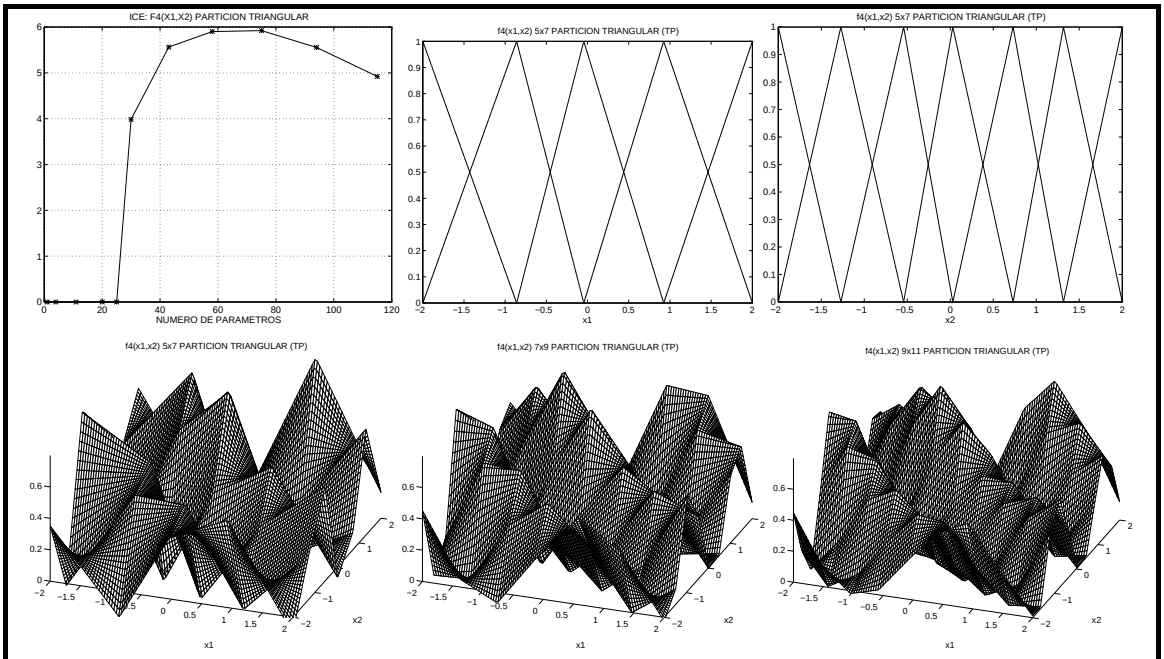


Fig. 4.35 a) Evolución del ICE para la función $f_4(x_1,x_2)$ utilizando una partición triangular. b) y c) Funciones de pertenencia al aproximar la función $f_4(x_1,x_2)$ mediante una configuración 5x7. d) Configuración 5x7. e) Configuración 7x9. f) Configuración 9x11.

Intentemos aproximar ahora la misma función pero utilizando funciones triangulares libres (ver Apéndice A). En este caso, el cambio más importante en el algoritmo consiste en utilizar una partición triangular para la etapa previa al descenso en gradiente. Una vez que se obtiene la configuración inicial, el proceso de minimización optimiza tanto los centros de las funciones de pertenencia como sus laterales. En la tabla 4.21 se presentan los resultados

Config.	Número de Parámetros	ECMN (entrenamiento)	ECMN (test)
1x1	1	1.000	1.000
2x2	8	0.972	0.969
3x3	19	0.905	0.896
3x4	25	0.750	0.725
4x5	39	0.416	0.401
4x6	46	0.262	0.256
5x6	55	0.129	0.132
5x7	63	0.086	0.089
6x7	73	0.066	0.072
7x8	93	0.042	0.057
8x9	115	0.034	0.053

Tabla 4.21 Evolución del ECMN al aproximar la función $f_4(x_1,x_2)$ mediante funciones triangulares libres.

obtenidos. En ésta se observa, como cabía esperar, que necesitamos ahora un número menor de reglas para obtener grados de aproximación similares. Además, en el caso particular de esta función (no es generalizable) para un mismo número de parámetros, las funciones triangulares libres consiguen un grado de aproximación mucho mejor que la partición triangular. En la figura 4.36a se representa el ICE para este caso, donde la configuración recomendada, la de 63 parámetros (5x7), obtiene un índice próximo a 6.75 que es bastante mejor que el obtenido en el caso anterior. Las funciones de pertenencia optimizadas para dicha configuración se presentan en las figuras 4.36b y 4.36c donde se puede apreciar la diferencia con las equivalentes para el caso anterior (dicha diferencia se traduce en un cambio enormemente significativo en el grado de aproximación obtenido). Finalmente, en las figuras 4.36d, 4.36e y 4.36f se representan las aproximaciones obtenidas para las configuraciones 4x6, 5x7 y 7x8 donde se aprecia cómo el hecho de poder optimizar también los laterales de las funciones triangulares consigue suavizar mucho más las funciones defuzzificadas.

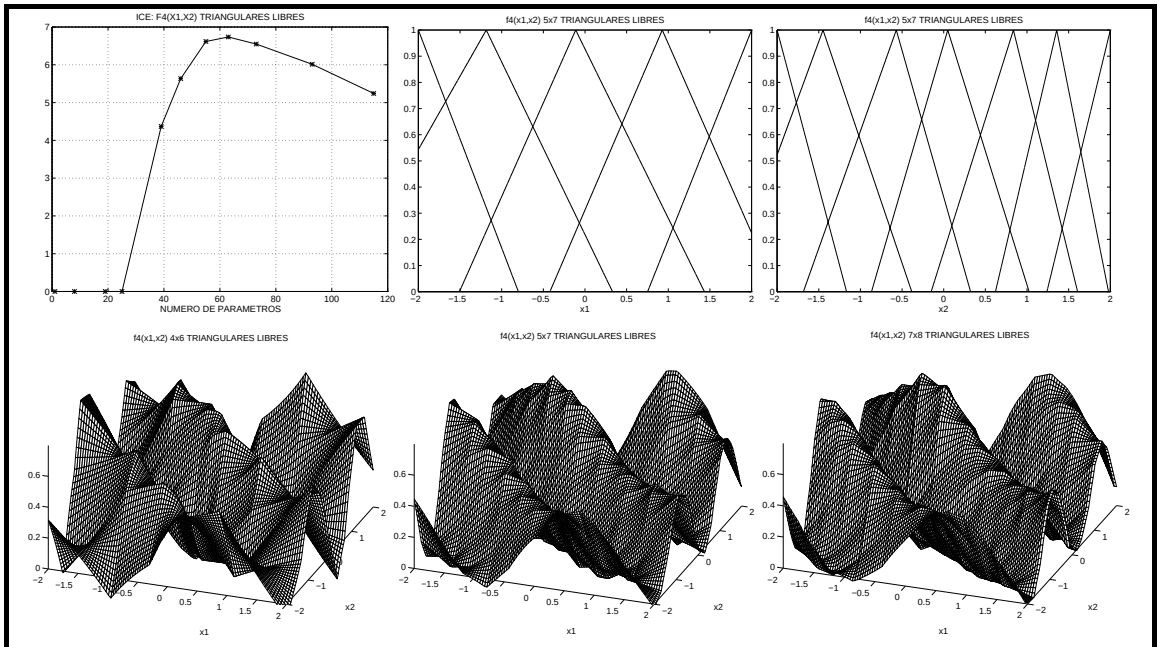


Fig. 4.36 a) Evolución del ICE para la función $f_4(x_1,x_2)$ utilizando funciones triangulares libres. b) y c) Funciones de pertenencia al aproximar la función $f_4(x_1,x_2)$ mediante una configuración 5x7. d) Configuración 4x6. e) Configuración 5x7. f) Configuración 7x8.

Como ya se comentó en el capítulo anterior, cuando se usa una partición triangular (TP), la función de salida es continua en todos los puntos y derivable también en todos los puntos salvo en aquéllos que caen en los picos de las funciones de pertenencia triangulares. Por lo general, esta no derivabilidad en ciertos puntos no conlleva ningún

problema en aplicaciones como el modelado o el control discreto. Hay, como ya se dijo, situaciones como en control continuo, donde conviene que la función de salida no sea sólo continua sino que también presente "suavidad" en su forma. Esta "suavidad" viene relacionada unívocamente con la existencia de la función derivada a lo largo de todos los puntos de la función.

Cuando se quiere que la superficie de salida sea suave en todos los puntos lo que se suele hacer en control difuso es substituir las funciones triangulares por gaussianas. Como ya se comentó, la primera idea que florece para utilizar funciones gaussianas en el algoritmo es intentar definir una partición pseudo-gaussiana de la misma forma que tenemos una partición triangular, intentando fijar los extremos para así poder definirla completamente con la posición de los centros, tal y como ocurre en una partición triangular. Para hacer esto, es necesario que las gaussianas presenten una caída distinta por un lado que por otro, por lo que dejarán de ser gaussianas sino pseudo-gaussianas asimétricas (ver Apéndice A, sección A.2.2). Como la derivada en el punto central sigue siendo cero, la función de salida seguirá siendo derivable. En el caso de una TP, los laterales se hacían coincidir con los centros de las funciones vecinas. En una partición pseudo-gaussiana habrá que fijar una constante L que determine el grado de pertenencia de una función en el centro vecino.

En la figura 4.37 se representan distintas funciones de pertenencia pseudo-gaussianas con distintos valores de L, en un intervalo $[c_v^i, c_v^{i+1}]$ delimitado por dos centros de la variable v. También se representa la función de pertenencia de una partición triangular. Nuestro propósito será encontrar el valor de L que compense el área total existente entre ambos tipos de funciones de pertenencia. El valor obtenido será el elegido para nuestras funciones pseudo-gaussianas.

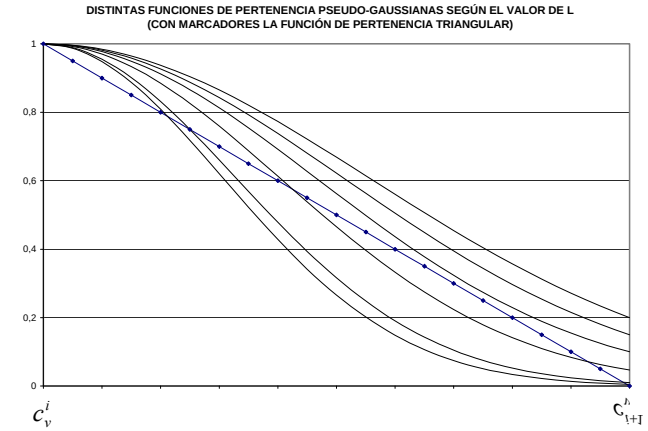


Fig. 4.37 Comparación entre una función triangular y funciones pseudo-gaussianas con distintos valores de L.

Para ello, definamos primero el valor de los grados de pertenencia en dicho intervalo para los dos tipos de funciones:

$$\tilde{\mu}_{X_v^i}(x) = \frac{c_v^{i+1} - x}{c_v^{i+1} - c_v^i} \qquad \textit{Función triangular} \tag{4-20}$$

$$\mu_{X_v^i}(x) = e^{-\frac{\left(x - c_v^i\right)^2}{\sigma_v^{i,+}}} \qquad \textit{Función pseudo-gaussiana} \tag{4-21}$$

El valor de *L* venía definido como (ver Apéndice A):

$$\mu_{X_v^i}(x = c_v^{i+1}) = e^{-\frac{\left(c_v^{i+1} - c_v^i\right)^2}{\sigma_v^{i,+}}} = L \tag{4-22}$$

de tal modo que:

$$\sigma_v^{i,+} = \frac{\left(c_v^{i+1} - c_v^i\right)^2}{ln\left(1/L\right)} \tag{4-23}$$

La función que queremos anular será, por tanto:

$$I(L) = \int_{c_v^i}^{c_v^{i+1}} \left\{\tilde{\mu}_{X_v^i}(x) - \mu_{X_v^i}(x)\right\} dx = 0 \tag{4-24}$$

o, equivalentemente:

$$I(L) = \int_{c_v^i}^{c_v^{i+1}} \left\{ \frac{c_v^{i+1} - x}{c_v^{i+1} - c_v^i} - e^{-\frac{\left(x - c_v^i\right)^2}{\sigma_v^{i,+}}} \right\} dx = \int_{c_v^i}^{c_v^{i+1}} \left\{ \frac{c_v^{i+1} - x}{c_v^{i+1} - c_v^i} - e^{-\left(\frac{x - c_v^i}{c_v^{i+1} - c_v^i}\right)^2 ln\,1/L} \right\} dx = 0 \tag{4-25}$$

donde hemos utilizado la equivalencia entre $\sigma_v^{i,+}$ y *L* dada por la ecuación (4-23).

Realizando los siguientes cambios:

$$\begin{gathered} t = \frac{x - c_v^i}{c_v^{i+1} - c_v^i} \qquad dt = \frac{dx}{c_v^{i+1} - c_v^i} \qquad \rightarrow \qquad x = c_v^i + \left(c_v^{i+1} - c_v^i\right) t \\ \int_{c_v^i}^{c_v^{i+1}} dx \quad \rightarrow \quad \left(c_v^{i+1} - c_v^i\right) \int_0^1 dt \end{gathered} \tag{4-26}$$

obtenemos:

$$I(L) = \left(c_v^{i+1} - c_v^i\right) \int_0^1 \left(1 - t - e^{-t^2 ln\,1/L}\right) dt = 0 \tag{4-27}$$

de tal forma que el valor que anula *I(L)* es independiente del valor de los centros de las funciones de pertenencia y su valor es:

$$L = 0.047 \tag{4-28}$$

que será el que utilicemos para nuestras funciones pseudo-gaussianas. En el caso de funciones pseudo-gaussianas libres dicho valor se utilizará para la etapa previa al descenso en gradiente (ver secciones 3.4.2 y 3.8.2). En la figura 4.38 se representa la función pseudo-gaussiana resultante y se compara con la función triangular a la que intenta asemejarse.

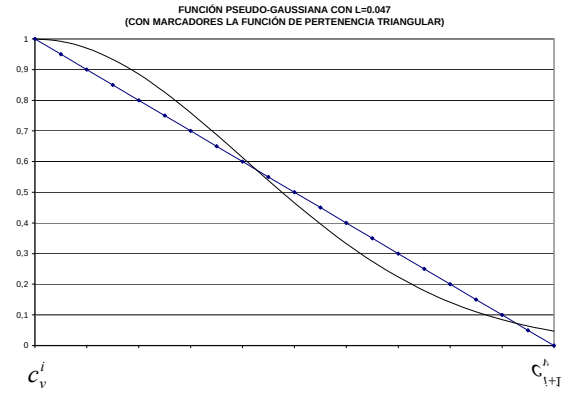


Fig. 4.38 Función pseudo-gaussiana con L=0.047.

Una vez fijado el valor de *L*, aplicaremos el algoritmo con este tipo de funciones de pertenencia a la misma función que en los casos anteriores. En la tabla 4.22 se presentan los resultados.

Para las configuraciones más sencillas, los ECMNs obtenidos son bastante parecidos a los de una partición triangular. Sin embargo, a partir de la configuración 5x7, la partición pseudo-gaussiana comienza a ser bastante peor que la triangular. Este hecho, al igual que pasaba al comparar con las triangulares libres, no es generalizable. Para otro tipo de funciones, la partición pseudo-gaussiana puede resultar igual o mejor que la triangular. En la figura 4.39a se representa el ICE para este caso. Como ya presumíamos, el valor recomendado (el de la configuración 6x8) presenta un índice bastante inferior al obtenido para la partición triangular. En las figuras 4.39b y 4.39c se muestran las funciones de pertenencia correspondientes a la configuración 5x7 donde se observa que la posición de los centros de las funciones de

Config.	Número de Parámetros	ECMN (entrenamiento)	ECMN (test)
1x1	1	1.000	1.000
2x2	4	0.975	0.971
3x3	11	0.925	0.917
4x4	20	0.717	0.708
4x5	25	0.606	0.595
4x6	30	0.493	0.492
5x7	43	0.325	0.331
6x8	58	0.228	0.241
7x9	75	0.168	0.189
8x9	85	0.151	0.171
8x10	94	0.129	0.152
9x11	115	0.107	0.128

Tabla 4.22 Evolución del ECMN al aproximar la función $f_4(x_1,x_2)$ mediante una partición pseudo-gaussiana.

pertenencia es prácticamente idéntico al caso de una TP. Sin embargo, la diferencia intrínseca entre funciones gaussianas y triangulares desequilibran la balanza del lado de éstas últimas para este caso concreto. Finalmente, en las figuras 4.39d, e y f se representan las aproximaciones para las configuraciones de 6x8, 7x9 y 9x11 donde se aprecia que, si bien la función de salida es derivable en todos los puntos, el hecho de no poder optimizar los laterales de las funciones pseudo-gaussianas provoca la existencia de pequeños montículos en la superficie de salida.

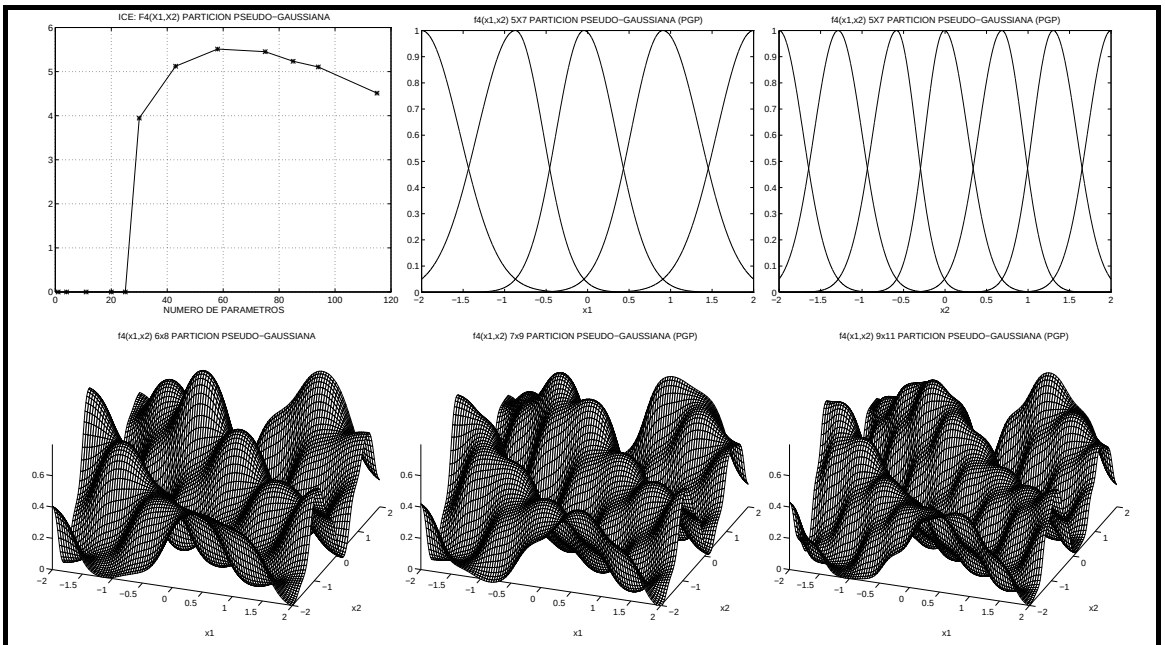


Fig. 4.39 a) Evolución del ICE para la función $f_4(x_1,x_2)$ utilizando una partición pseudo-gaussiana. b) y c) Funciones de pertenencia al aproximar la función $f_4(x_1,x_2)$ mediante una configuración 5x7. d) Configuración 6x8. e) Configuración 7x9. f) Configuración 9x11.

El caso de las funciones gaussianas es un poco más complejo que los anteriores, ya que se trata de funciones simétricas y no podemos tener caídas distintas según la rama de la gaussiana de la que se trate. En este caso debemos hacer un tratamiento adicional. Para ello, consideraremos inicialmente la función de pertenencia como si fuera pseudo-gaussiana y posteriormente calcularemos el valor de la desviación de la función gaussiana que compense la diferencia entre ambas funciones.

Consideremos la función centrada en el punto c_v^i de la variable *v*. Si la función fuera pseudo-gaussiana, utilizando el valor de *L* dado por (4-28) y el valor de los centros vecinos podemos calcular el valor de las desviaciones izquierdo y derecho obteniendo:

$$\tilde{\mu}_{X_v^i}(x) = e^{-\frac{(x-c_v^i)^2}{\sigma_v^{i,-}}} U(x;-\infty,c_v^i) + e^{-\frac{(x-c_v^i)^2}{\sigma_v^{i,+}}} U(x;c_v^i,\infty) \tag{4-29}$$

pero la función finalmente ha de ser de tipo gaussiano:

$$\mu_{X_v^i}(x) = e^{-\frac{(x-c_v^i)^2}{\sigma_v^i}} \tag{4-30}$$

De tal forma que elegiremos el valor de σ_v^i de tal forma que anule:

$$I(\sigma_v^i) = \int_{-\infty}^{\infty} \left\{ \tilde{\mu}_{X_v^i}(x) - \mu_{X_v^i}(x) \right\} dx = 0 \tag{4-31}$$

Substituyendo (4-29) y (4-30) en la expresión anterior tenemos:

$$I(\sigma_v^i) = \int_{-\infty}^{\infty} \left\{ e^{-\frac{(x-c_v^i)^2}{\sigma_v^{i,-}}} U(x;-\infty,c_v^i) + e^{-\frac{(x-c_v^i)^2}{\sigma_v^{i,+}}} U(x;c_v^i,\infty) - e^{-\frac{(x-c_v^i)^2}{\sigma_v^i}} \right\} dx = 0 \tag{4-32}$$

y, teniendo en cuenta la definición de la función U (ver A-1):

$$\int_{-\infty}^{\infty} e^{-\frac{(x-c_v^i)^2}{\sigma_v^i}} dx = \int_{-\infty}^{c_v^i} e^{-\frac{(x-c_v^i)^2}{\sigma_v^{i,-}}} dx + \int_{c_v^i}^{\infty} e^{-\frac{(x-c_v^i)^2}{\sigma_v^{i,+}}} dx \tag{4-33}$$

Haciendo ahora el cambio $x\text{-}c_v^i = t$:

$$\int_{-\infty}^{\infty} e^{-\frac{x^2}{\sigma_v^i}} dx = \int_{-\infty}^{0} e^{-\frac{x^2}{\sigma_v^{i,-}}} dx + \int_{0}^{\infty} e^{-\frac{x^2}{\sigma_v^{i,+}}} dx \tag{4-34}$$

y teniendo en cuenta que:

$$\int_0^{\infty} e^{-\frac{x^2}{s}} dx = \frac{\sqrt{\pi}\sqrt{s}}{2} \tag{4-35}$$

y la paridad del integrando, obtenemos fácilmente la siguiente igualdad:

$$\sqrt{\pi}\sqrt{\sigma_v^i} = \frac{\sqrt{\pi}\sqrt{\sigma_v^{i,-}}}{2} + \frac{\sqrt{\pi}\sqrt{\sigma_v^{i,+}}}{2} \tag{4-36}$$

de donde la solución que buscamos es:

$$\sigma_v^i = \left(\frac{\sqrt{\sigma_v^{i,-}} + \sqrt{\sigma_v^{i,+}}}{2} \right)^2 \tag{4-37}$$

En la figura 4.40 se presenta un ejemplo de cómo queda la función gaussiana resultante.

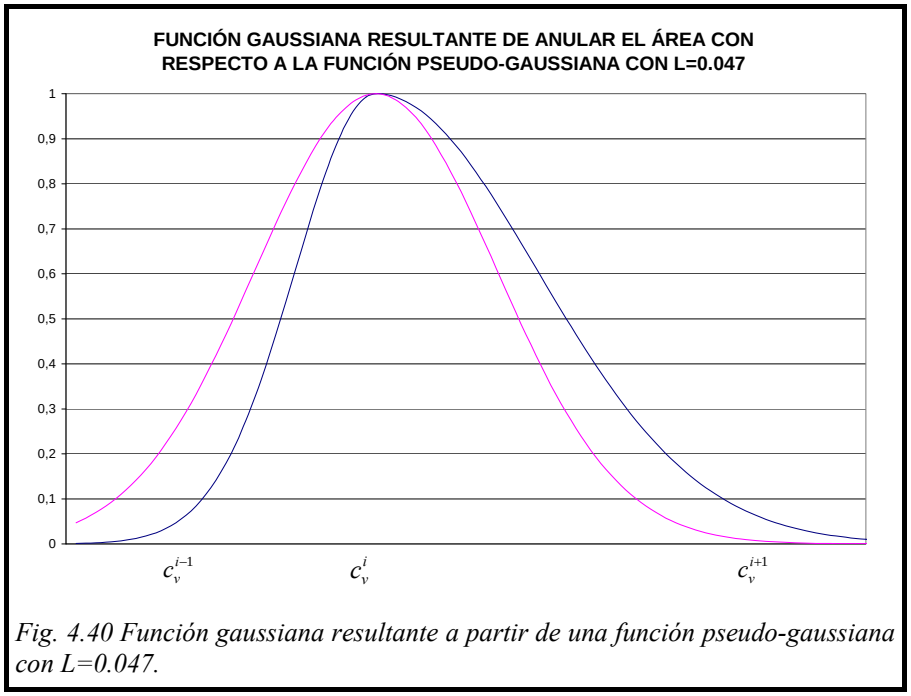


Fig. 4.40 Función gaussiana resultante a partir de una función pseudo-gaussiana con L=0.047.

Apliquemos ahora el algoritmo para el caso de aproximar la función $f_4(x_1,x_2)$ mediante funciones gaussianas. La evolución del *ECMN* para este caso se presenta en la tabla 4.23.

Config.	Número de Parámetros	ECMN (entrenamiento)	ECMN (test)
1x1	1	1.000	1.000
2x2	8	0.970	0.968
3x3	17	0.875	0.865
3x4	22	0.673	0.673
4x5	34	0.443	0.418
5x6	48	0.139	0.134
5x7	55	0.091	0.089
6x7	64	0.063	0.061
7x8	82	0.026	0.028
8x8	92	0.023	0.026
9x8	102	0.019	0.022
9x9	113	0.013	0.015

Tabla 4.23 Evolución del ECMN al aproximar la función $f_4(x_1,x_2)$ mediante funciones gaussianas.

Comparando dicha tabla con la obtenida para funciones triangulares libres podemos observar que los grados de aproximación son bastante similares si nos atenemos al número total de reglas pero no así respecto al número de parámetros. Sin embargo, para las configuraciones más complejas, las funciones gaussianas superan apreciablemente a las funciones triangulares. Otro aspecto significativo es que la diferencia entre el error

cuadrático medio normalizado para los datos de entrenamiento y los de test es mucho menor para el caso de las funciones gaussianas lo cual quiere decir que su capacidad de interpolación es mucho mejor que la de las funciones triangulares para esta función concreta. En la figura 4.41a se representa el ICE para este caso donde se puede comprobar la superioridad de las funciones gaussianas con respecto a las funciones anteriores. La configuración que obtiene mayor índice es la de 55 parámetros (configuración 5x7) con un valor de 7. En las figuras 4.41b y 4.41c se representan las funciones de pertenencia obtenidas para esta configuración pudiendo observarse que son bastante diferentes que las equivalentes para funciones triangulares libres, siendo, sin embargo, el *ECMN* para los datos de test idéntico en ambos casos. En las figuras 4.41d, 4.41e y 4.41f se representan las funciones defuzzificadas para las configuraciones 5x6, 5x7 y 7x8 donde, en esta última, prácticamente no se puede apreciar diferencia visual alguna entre la función original y la aproximada.

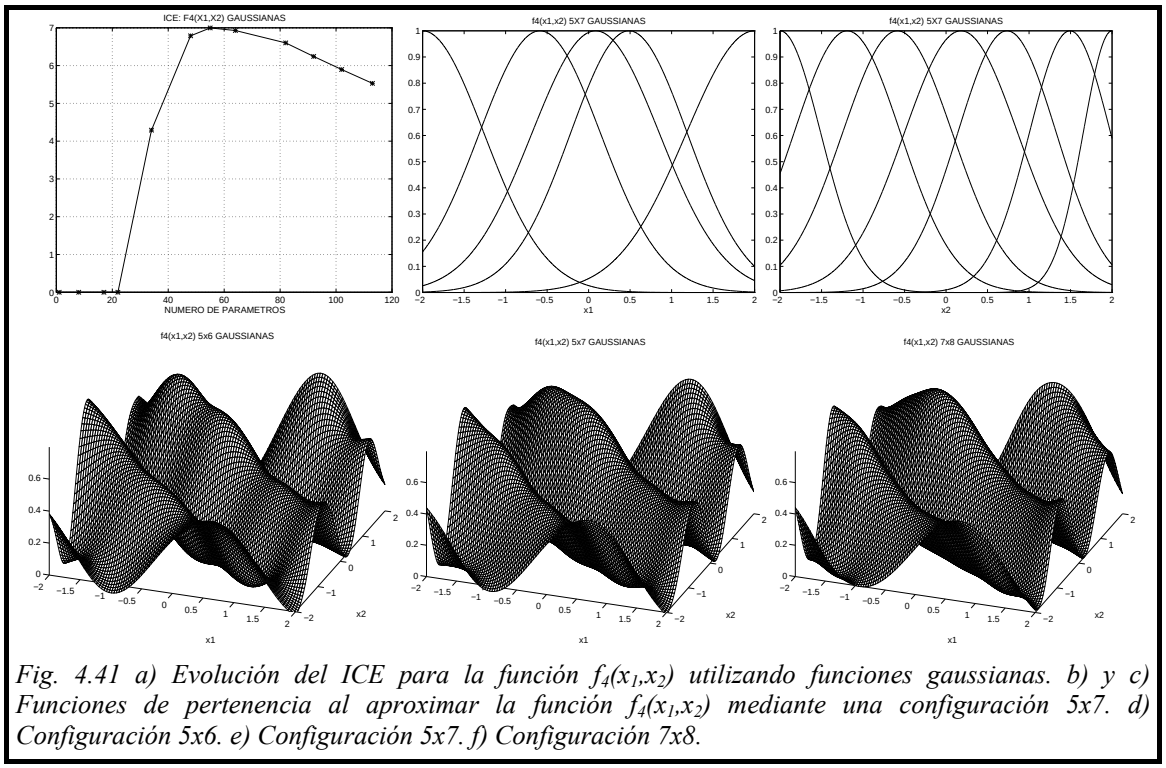


Fig. 4.41 a) Evolución del ICE para la función $f_4(x_1,x_2)$ utilizando funciones gaussianas. b) y c) Funciones de pertenencia al aproximar la función $f_4(x_1,x_2)$ mediante una configuración 5x7. d) Configuración 5x6. e) Configuración 5x7. f) Configuración 7x8.

El caso de las funciones pseudo-gaussianas libres se aborda de la misma forma que las triangulares libres pero partiendo de una partición pseudo-gaussiana en la etapa previa. En la tabla 4.24 se presentan los resultados obtenidos para este caso. Se puede apreciar que prácticamente la mayoría de las configuraciones coinciden en el grado de aproximación con las obtenidas mediante funciones gaussianas. Esto es debido a que,

por el tipo de función que se trata aproximar, las funciones óptimas resultan tener la misma desviación a cada lado. Esto se aprecia también al ver las funciones triangulares libres anteriores donde prácticamente todas las funciones son simétricas.

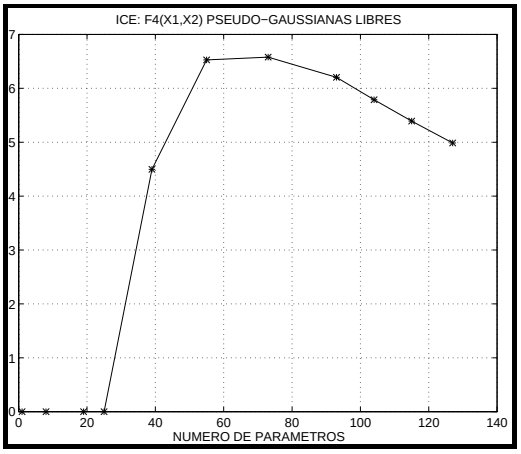


Fig. 4.42 Evolución del ICE para la función $f_4(x_1,x_2)$ utilizando funciones pseudo-gaussianas libres.

Config.	Número de Parámetros	ECMN (entrenamiento)	ECMN (test)
1x1	1	1.000	1.000
2x2	8	0.970	0.968
3x3	19	0.875	0.865
3x4	25	0.673	0.673
4x5	39	0.403	0.381
5x6	55	0.138	0.133
6x7	73	0.063	0.061
7x8	93	0.023	0.025
8x8	104	0.022	0.025
9x8	115	0.019	0.022
9x9	127	0.013	0.015

Tabla 4.24 Evolución del ECMN al aproximar la función $f_4(x_1,x_2)$ mediante funciones pseudo-gaussianas libres.

Esto quiere decir que, para este ejemplo, esencialmente sobra un parámetro por cada función de pertenencia lo que provoca que el índice ICE (ver figura 4.42) sea peor que para el caso de funciones gaussianas. Para poder observar el verdadero potencial de las funciones pseudo-gaussianas libres utilizaremos la función [CHE-96]:

$$f_5(x_1, x_2) = 42.659 \cdot \left\{0.1 + x_1\left[0.05 + (x_1^4 - 10x_1^2x_2^2 + 5x_2^4)\right]\right\} \qquad (4\text{-}38)$$

donde cada variable está definida en el intervalo [-0.5,0.5] (ver figura 4.43).

Ésta es una función mucho más sencilla que la anterior por lo que se obtienen mejores grados de aproximación con menor número de parámetros. En las figuras 4.44a y 4.44b se representan las funciones gaussianas optimizadas al aproximar la función anterior mediante una configuración de 4x5. La función defuzzificada es la que ilustra la figura 4.45. El *ECMN* en este caso para los datos

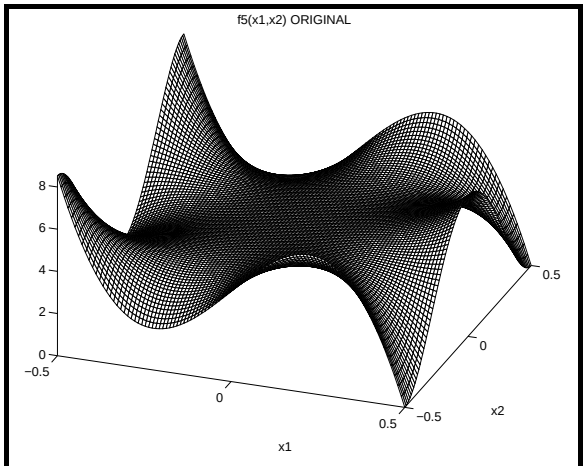


Fig. 4.43 Función $f_5(x_1,x_2)$ original.

de test es de 0.152. A continuación intentaremos aproximar la misma función para esa misma configuración utilizando funciones pseudo-gaussianas libres.

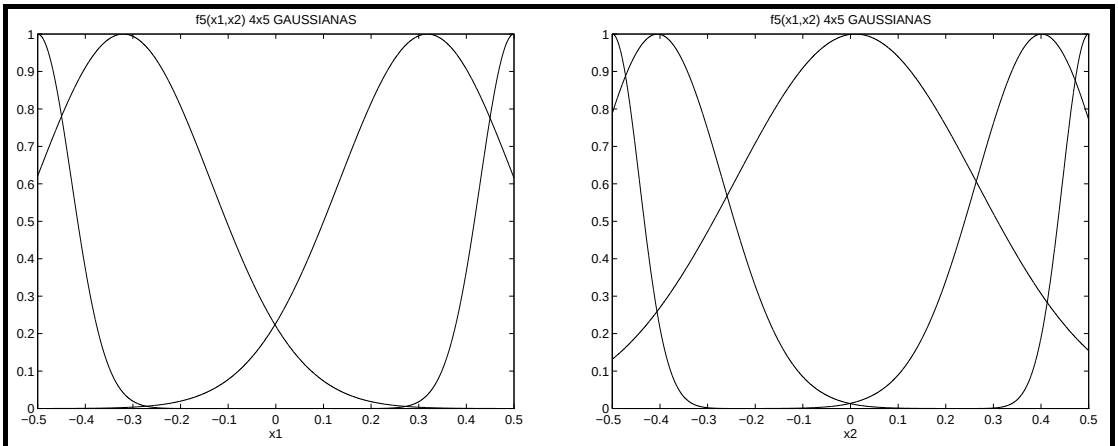


Fig. 4.44 Funciones de pertenencia al aproximar la función $f_5(x_1,x_2)$ mediante una configuración 4x5 usando funciones gaussianas. a) Variable x_1 b) Variable x_2.

Las funciones de pertenencia optimizadas para este caso son las de las figuras 4.47a y 4.47b donde se puede apreciar claramente la diferencia con las funciones gaussianas. El *ECMN* para los datos de test es en este caso de 0.049, es decir, un valor significativamente mejor que el obtenido en el caso anterior. En la figura 4.46 se representa la función defuzzificada pudiéndose observar el gran grado de aproximación conseguido con tan sólo 20 reglas.

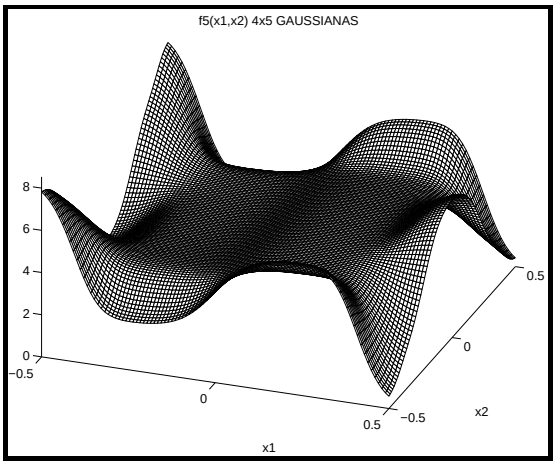


Fig. 4.45 Función $f_5(x_1,x_2)$ aproximada mediante una configuración 4x5 de funciones gaussianas.

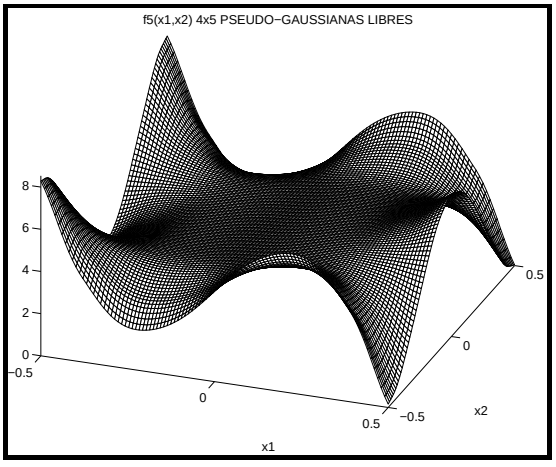


Fig. 4.46 Función $f_5(x_1,x_2)$ aproximada mediante una configuración 4x5 de funciones pseudo-gaussianas libres.

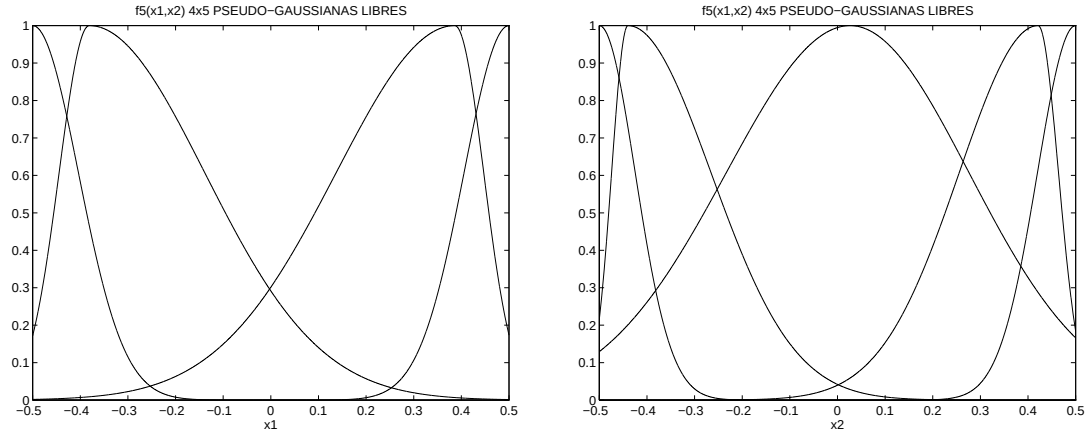


Fig. 4.47 Funciones de pertenencia al aproximar la función $f_5(x_1,x_2)$ mediante una configuración 4x5 usando funciones pseudo-gaussianas libres. a) Variable x_1 b) Variable x_2.

4.3.1 Interpretabilidad de las reglas

Una de las principales ventajas de utilizar una partición triangular para aproximar nuestras funciones es que la interpretabilidad de las funciones triangulares optimizadas es muy sencilla ya que su grado de solapamiento es siempre el mismo y cada una tiene bien delimitada su zona de actuación. Al introducir nuevos tipos de particionamiento del espacio de entrada y un mayor número de parámetros, la etapa de descenso en gradiente siempre mueve los parámetros a optimizar en la dirección en la que se disminuye el error, sin tener en cuenta la forma final de las funciones de pertenencia. Es por ello que algunos autores no recomienden el uso de esta técnica de minimización para optimizar sistemas difusos. En la sección 3.8.3 del capítulo anterior se presentó un método para adaptar el algoritmo presentado para los casos en los que la interpretabilidad final de las funciones optimizadas sea una especificación a tener en cuenta en el problema a resolver. Para ello se introdujo una serie de parámetros que el usuario final debe especificar y cuyo fin era:

1.- Evitar que dos centros se aproximaran excesivamente (d_{min}).

2.- Evitar que un dato activara de forma apreciable más de dos funciones de pertenencia simultáneamente (a_s).

3.- Evitar que existiera un dato dentro del rango permitido que no activase ninguna regla (a_m).

4.- Evitar que una función de pertenencia se colapsase.

De entre todas estas condiciones, realmente la segunda es la que afecta de forma directa a la interpretabilidad de las reglas finales ya que el resto son condiciones necesarias generales para el buen funcionamiento del algoritmo y que ya han sido aplicadas por defecto a todos los ejemplos mostrados en este capítulo. Para mostrar un ejemplo de aplicación de estas restricciones, consideremos nuevamente la función:

$$f_2(x)=e^{-5x}\,sin(2\pi x) \qquad x\in[0,1] \tag{4-39}$$

cuya gráfica ya se mostró en la figura 4.4. Intentaremos aproximar dicha función mediante pseudo-gaussianas libres imponiendo la condición de que la apertura de solapamiento sea $a_s=0.3$, es decir, que si un dato activa más de dos funciones de pertenencia, como mucho su grado de activación a otras funciones sea menor o igual a 0.3. En la figura 4.48a se representa la función defuzzificada al utilizar 5 funciones de pertenencia (*ECMN*=0.017) y en la figura 4.48b la posición final de las funciones pseudo-gaussianas libres utilizadas. En dicha figura se comprueba que el grado de solapamiento entre la función etiquetada como '*Z*' y la etiquetada como '*M*' se ha quedado estancado en el valor 0.3 que es el impuesto por a_s. Si se hubiera permitido un mayor solapamiento el *ECMN* obtenido sería de 0.016 (un poco mejor) pero a expensas de que un dato cercano a 0.1 activara con grado mayor de 0.3 simultáneamente tres funciones de pertenencia. Por ejemplo, si dichas funciones pertenecieran a la variable "Valor Absoluto del Error", un cierto error podría considerarse simultáneamente "cero", "pequeño" y "mediano".

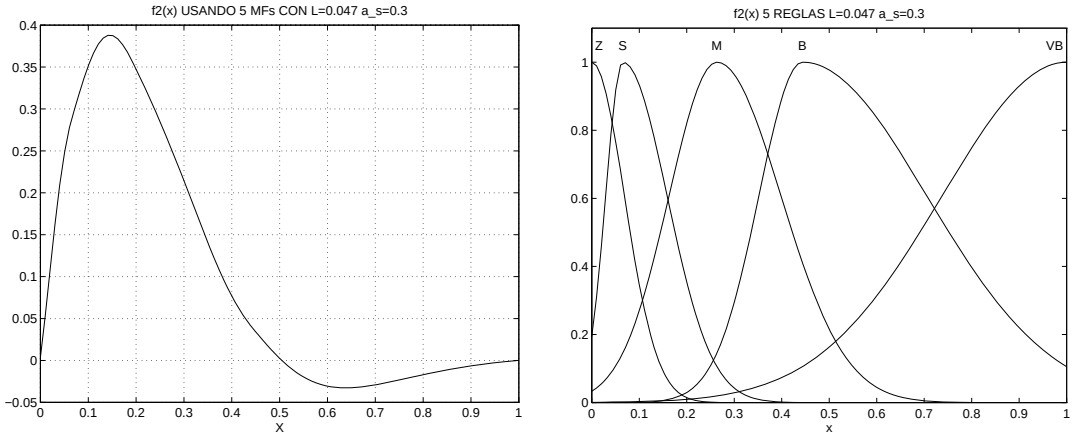


Fig. 4.48 a) Función $f_2(x)$ aproximada mediante una configuración de 5 funciones de pertenencia pseudo-gaussianas libres y $a_s=0.3$. b) Funciones de pertenencia obtenidas.

Consideremos ahora el caso de que intentásemos aproximar la misma función pero sin la limitación del solapamiento entre funciones (dado por a_s) y con un valor alto del parámetro L para la etapa previa al descenso en gradiente (por ejemplo L=0.6). En este caso, el error cuadrático medio normalizado es, nuevamente 0.016 (como en el caso de L=0.047 sin limitaciones), por lo que la función defuzzificada es muy similar a la presentada en la figura 4.48a pero con la diferencia de que ahora las funciones de pertenencia son comparablemente mucho más difíciles de interpretar (ver figura 4.49).

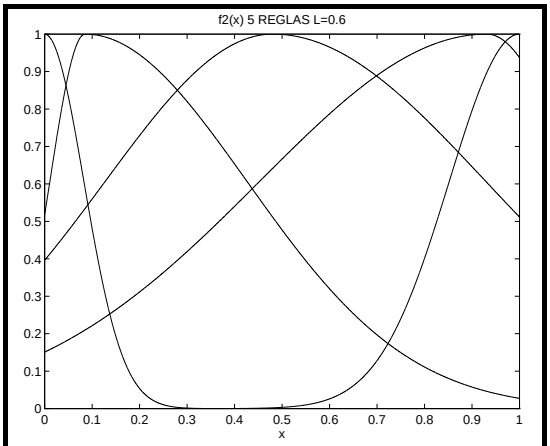


Fig. 4.49 Funciones de pertenencia optimizadas al aproximar $f_2(x)$ con L=0.6 y sin restricciones de solapamiento.

Esto justifica el utilizar un valor de L que asemeje a las funciones triangulares y a las pseudo-gaussianas de tal forma que las funciones de pertenencia tras la etapa previa sean fácilmente interpretables con la esperanza de que durante el descenso en gradiente el mínimo local más cercano sea relativamente parecido a la configuración inicial. Gracias a esto, las funciones de pertenencia de la figura 4.48b se pueden utilizar fácilmente para explicar un problema de control, obteniéndose, para este caso, reglas del tipo:

SI Valor Absoluto del Error ES Pequeño ENTONCES Salida ES Grande
SI Valor Absoluto del Error ES Mediano ENTONCES Salida ES Mediana
SI Valor Absoluto del Error ES Muy Grande ENTONCES Salida ES Cero

4.4 Comparación con otros métodos propuestos en la bibliografía

Finalmente, en esta sección se realizará una comparación entre los resultados obtenidos por el algoritmo aquí presentado y los de otras metodologías propuestas en la bibliografía. Para ello, distinguiremos entre algoritmos dedicados a la síntesis de sistemas difusos y los que utilizan otros paradigmas como las redes neuronales, algoritmos genéticos, sistemas neuro-difusos, etc.

4.4.1 Comparación con otros algoritmos para la síntesis directa de sistemas difusos

Uno de los primeros trabajos que generaban reglas difusas a partir de muestras fue publicado por L.X.Wang [WAN-92b] donde, esencialmente, cada dato generaba una regla y posteriormente se seleccionaban aquellas que más se activaban. Posteriormente, T.Sudkamp y R.J.Hamell [SUD-94] mejoraban el algoritmo anterior reduciendo la posibilidad de que el consecuente de una regla se viera afectado por datos ruidosos. Asimismo, desarrollaron una teoría para completar la base de reglas basada en el concepto de que la similaridad en la entrada debe producir similaridad en la salida. Para

Función	**Propuestos en la bibliografía**				
	Algoritmo		**Nº de Reglas**	**Índice Evaluador**	
$F(x)=x^3$ $x\in[-1,1]$	T.Sudkamp & R.J.Hammell [SUD-94]			Error Medio	Error Máximo
		Region growing	15	0.044	0.193
			25	0.021	0.083
		Weighted average	15	0.044	0.193
			25	0.021	0.083
	L-X.Wang & J.M. Mendel [WAN-92b]		15	0.029	0.079
			25	0.018	0.088
	Mejora de L-X.Wang [SUD-94]		15	0.079	0.071
			25	0.088	0.050
$F(x)=sin(2\pi x)$ $x\in[-1,1]$	T.Sudkamp & R.J.Hammell [SUD-94]			Error Medio	Error Máximo
		Region growing	15	0.131	0.408
			25	0.052	0.167
		Weighted average	15	0.180	1.051
			25	0.082	0.759
	L-X.Wang & J.M. Mendel [WAN-92b]		15	0.060	0.159
			25	0.026	0.092
	Mejora de L-X.Wang [SUD-94]		15	0.071	0.179
			25	0.031	0.083
$F(x_1,x_2)=x_1^2+x_2^2-1$ $x_1,x_2\in[-1,1]$	T.Sudkamp & R.J.Hammell [SUD-94]			Error Medio	Error Máximo
		Region growing	15x15	0.038	0.286
			25x25	0.035	0.250
		Weighted average	15x15	0.060	1.286
			25x25	0.148	1.333
	L-X.Wang & J.M. Mendel [WAN-92b]		5x5	0.107	0.5
			15x15	0.034	0.143
	Mejora de L-X.Wang [SUD-94]		5x5	0.154	0.4
			15x15	0.03	0.143

Tabla 4.25a Comparación del algoritmo propuesto. Resultados de [SUD-94, WAN-92b].

ello, presentaron dos métodos relativamente equivalentes: el método de "crecimiento por regiones" (Region Growing) y la "media ponderada" (Weighted Average).

En la tabla 4.25a se presentan los resultados obtenidos por los anteriores autores para tres funciones distintas y en la 4.25b los proporcionados por el algoritmo presentado usando una partición triangular. Aunque los índices proporcionados por los autores son, en este caso, el error medio (la media de los valores absolutos de los errores) y el error máximo, en la tabla anterior se muestran también otros valores para comparaciones futuras. En negrita se destacan los índices que se deben comparar con la tabla 4.25a. A la vista de las tablas queda patente que el algoritmo propuesto es capaz, en todos los casos, de obtener configuraciones mejores con un menor número de funciones de pertenencia y, consecuentemente, un menor número de parámetros. Como muestra, para el caso de la función bidimensional, el mejor resultado del error medio es de 0.03 para el caso de 225 reglas (Wang también utilizaba una partición triangular) mientras que con tan sólo 25 el algoritmo propuesto ya puede rebajar dicho valor (0.0225).

Función	**Algoritmo propuesto**					
	Nº de reglas	**Índice evaluador**				
		ECMN	**ECM**	**Error Medio**	**Error Mínimo**	**Error Máximo**
$F(x)=x^3$ $x\in[-1,1]$	4	0.080	9.7E-4	**0.026**	-0.075	**0.072**
	6	0.032	1.6E-4	**0.011**	-0.032	**0.028**
	8	0.017	4.5E-5	**0.006**	-0.016	**0.015**
$F(x)=sin(2\pi x)$ $x\in[-1,1]$	6	0.112	0.006	**0.068**	-0.178	**0.217**
	8	0.0823	0.003	**0.0473**	-0.162	**0.149**
	10	0.0237	0.0002	**0.0262**	-0.072	**0.078**
$F(x_1,x_2)=$ $x_1^2+x_2^2-1$ $x_1,x_2\in[-1,1]$	5x5	0.0617	0.0007	**0.0225**	-0.080	**0.043**
	6x6	0.0394	0.0003	**0.0144**	-0.053	**0.028**
	7x7	0.0273	0.00015	**0.0099**	-0.035	**0.019**

Tabla 4.25b Comparación del algoritmo propuesto con [SUD-94, WAN-92b].

En un trabajo más reciente, J.A.Dickerson y B.Kosko [DIC-96] utilizaban un sistema híbrido neuro-difuso para demostrar el comportamiento de su algoritmo basado en

reglas elipsoidales y así reducir su número. En sus resultados utilizaron diferentes valores para los pesos de las reglas y diferentes métodos de aprendizaje. En el mismo año, J.H.Nie y T.H.Lee [NIE-96] presentan también tres algoritmos distintos de aproximación funcional (a los que llaman P1, P2 y P3) junto con dos métodos de razonamiento difuso (K1 y K2). Para demostrar su funcionamiento, aproximan una función unidimensional mediante 30 reglas difusas y utilizan como índice evaluador el error cuadrático medio y el error máximo. En las tablas 4.26a y 4.26b se presentan los resultados obtenidos por estos autores y los que proporciona el algoritmo propuesto usando una partición triangular, pudiéndose concluir nuevamente que tanto el grado de aproximación como la complejidad del sistema empleado son notablemente mejores.

Función	**Propuestos en la bibliografía**					
	Algoritmo				**Nº de Reglas**	**ECM**
$F(x) = 3x(x-1)(x-1.9)\cdot(x+0.7)(x+1.8)$ $x\in[-2.1,2.1]$	J.A. Dickerson & B.Kosko [DIC-96]	Efecto de los diferentes pesos de las reglas	$W_k=1$		6	94.65
			$W_k=1/V_k$			28.25
			$W_k=1/V_k^2$			10.53
		Aprendizaje no supervisado				7.927
		Aprendizaje supervisado				3.069
$F(x)=3exp(-x^2)\cdot sin(\pi x)$ $x\in[-3,3]$	J.H.Nie & T.H.Lee [NIE-96]	Método			30	**ECM**	**Error Máx.**
		P1/γ1				0.0114	0.354
		P1/γ2				0.0078	0.353
		P2/γ1				0.0071	0.278
		P2/γ2				0.0068	0.278
		P3/γ1				0.0081	0.388
		P3/γ2				0.0080	0.388

Tabla 4.26a Comparación del algoritmo propuesto. Resultados de [DIC-96, NIE-96].

Otro método reseñable de la bibliografía es el que proponen R.Rovatti y R.Guerrieri [ROV-96] en donde tanto las funciones de pertenencia en la entrada como en la salida están fijos lo que les permite realizar un proceso de minimización simbólica obteniendo los resultados que se recogen en la tabla 4.27a, donde escogieron como índice evaluador la raíz cuadrada del error cuadrático medio (RECM).

Función	Algoritmo propuesto					
	Nº de Reglas	Índice Evaluador				
		ECMN	ECM	Error Medio	Error Mín.	Error Máx.
$F(x) = 3x(x-1)(x-1.9)\cdot(x+0.7)(x+1.8)$ $x \in [-2.1,2.1]$	5	0.329	**5.01**	1.85	-4.01	4.92
	6	0.17	**1.35**	0.74	-3.36	4.92
	7	0.10	**0.46**	0.49	-2.66	1.72
$F(x) = 3exp(-x^2)\cdot sin(\pi x)$ $x \in [-3,3]$	6	0.111	**0.0113**	0.090	-0.296	**0.296**
	7	0.099	**0.0092**	0.081	-0.178	**0.296**
	10	0.051	**0.0024**	0.033	-0.162	**0.162**

Tabla 4.26b Comparación del algoritmo propuesto con [DIC-96, NIE-96].

En los trabajos anteriores, el número de reglas viene siempre fijado antes de ejecutar el algoritmo. Sin embargo, I.Rojas [ROJ-00] elaboró un algoritmo de aproximación funcional que no sólo optimizaba las reglas sino también el número de ellas. Para ello, los consecuentes de las reglas se determinaban ponderando la salida de los datos con el grado de activación de cada una de ellas y las funciones de pertenencia triangulares se optimizaban minimizando un índice que determinaba la controversia entre las reglas. Dicho índice se usaba posteriormente para incrementar la complejidad del sistema aproximador.

En la tabla 4.27a se recogen los resultados obtenidos por estos autores para varias funciones y en la 4.27b los proporcionados por el algoritmo. Nuevamente, tanto la complejidad del sistema como el grado de aproximación son claramente superiores a los obtenidos por los métodos anteriores, si bien, en el primer caso hay que hacer constar que los consecuentes de las reglas son difusos y no discretos.

Función	Propuesto en la bibliografía		
	Algoritmo	**Nº de Reglas**	**RECM**
$F(x_1, x_2) = \frac{x_1 + x_2}{2},\ x_1, x_2 \in [0,1]$	R. Rovatti & R. Guerrieri [ROV-96]	5x5	0.070
$F(x_1, x_2) = \frac{1}{2} + 16\frac{(x_1 - 1/2)(x_2 - 1/2)(x_1 + 1/8)}{1 + (4x_1 - 2)^2 + (4x_2 - 2)^2}$ $x_1, x_2 \in [0,1]$		5x5	0.041
$F(x_1, x_2) = \frac{1}{2}[1 + sin(2\pi x_1)cos(2\pi x_2)]$ $x_1, x_2 \in [0,1]$		5x5	0.130
$F(x_1, x_2) = sin(x_1 x_2)$ $x_1, x_2 \in [-2,2]$	Rojas [ROJ-00]	36	0.157
		49	0.119
		64	0.089
$F(x_1, x_2) =$ $42.659 \cdot \{0.1 + x_1[0.05 + (x_1^4 - 10x_1^2x_2^2 + 5x_2^4)]\}$ $x_1, x_2 \in [-0.5, 0.5]$		42	0.341
		56	0.264
		90	0.158

Tabla 4.27a Comparación del algoritmo propuesto. Resultados de [ROV-96, ROJ-00].

Función	Algoritmo propuesto					
	Nº de Reglas	**Índice evaluador**				
		ECMN	**RECM**	**Error Medio**	**Error Mín.**	**Error Máx.**
$F(x_1, x_2) = \frac{x_1 + x_2}{2},\ x_1, x_2 \in [0,1]$	2x2	0	**0**	0	0	0
$F(x_1, x_2) = \frac{1}{2} + 16\frac{(x_1 - 1/2)(x_2 - 1/2)(x_1 + 1/8)}{1 + (4x_1 - 2)^2 + (4x_2 - 2)^2}$ $x_1, x_2 \in [0,1]$	5x5	0.050	**0.009**	0.008	-0.029	0.0345
	6x6	0.024	**0.005**	0.004	-0.012	0.0128
	8x8	0.011	**0.002**	0.002	-0.006	0.0060
$F(x_1, x_2) = \frac{1}{2}[1 + sin(2\pi x_1)cos(2\pi x_2)]$ $x_1, x_2 \in [0,1]$	4x4	0.158	**0.080**	0.063	-0.250	0.217
	5x5	0.122	**0.030**	0.023	-0.107	0.153
	6x6	0.056	**0.014**	0.011	-0.055	0.052
$F(x_1, x_2) = sin(x_1 x_2)$ $x_1, x_2 \in [-2,2]$	4x4	0.121	**0.077**	0.056	-0.276	0.239
	5x5	0.097	**0.062**	0.042	-0.266	0.265
	6x6	0.047	**0.030**	0.021	-0.138	0.095
$F(x_1, x_2) =$ $42.659 \cdot \{0.1 + x_1[0.05 + (x_1^4 - 10x_1^2x_2^2 + 5x_2^4)]\}$ $x_1, x_2 \in [-0.5, 0.5]$	4x4	0.270	**0.298**	0.211	-1.197	1.217
	4x5	0.194	**0.214**	0.151	-0.684	0.706
	6x6	0.090	**0.099**	0.075	-0.387	0.297

Tabla 4.27b Comparación del algoritmo propuesto con [ROV-96, ROJ-00].

Finalmente aplicaremos el algoritmo a un caso real con cierta complejidad: la predicción de la concentración de CO_2 de un horno de gas teniendo en cuenta la concentración actual y el flujo de gas hacia el horno. Este es un problema muy conocido, introducido inicialmente por Box y Jenkins [BOX-76] que proporcionaron 296 datos de E/S que son usualmente utilizados para testear algoritmos de modelización de sistemas. El índice de evaluación que se suele utilizar en este caso es:

$$J = \frac{1}{246}\sum_{k=51}^{296}\left(y(k)-\hat{y}(k)\right)^2 \tag{4-40}$$

donde $y(k)$ es el tanto por ciento de gas en el horno en el instante k e $\hat{y}(k)$ es la salida del modelo.

Algoritmos propuestos en la bibliografía		Nº de Reglas	J
Box & Jenkins Horno de Gas	Sugeno & Tanaka [SUG-91]	2	0.359
	Tong [TON-78]	19	0.469
	Pedrycz [PED-84]	25	0.320
	Xu [XU-87]	81	0.328

Tabla 4.28a Comparación del algoritmo propuesto. Resultados para el horno de Box & Jenkins.

En la tabla 4.28a se presentan los resultados obtenidos para este ejemplo por diferentes autores, todos ellos utilizando un sistema difuso para modelar el sistema. Los valores proporcionados por el algoritmo se recogen en la tabla 4.28b, donde las primeras 4 filas corresponden a una partición triangular y las últimas 5 a funciones pseudo-gaussianas libres. Hay que hacer notar, en este caso, que aunque Sugeno y Tanaka obtengan un valor de J aceptable usando tan solo dos reglas, en el algoritmo que utilizan realmente cada una de esas reglas tiene definidas sus propias funciones de pertenencia y que los consecuentes

Algoritmo Propuesto	Nº de Reglas	ECMN	J	Error Medio	Error Mínimo	Error Máximo
Box & Jenkins Horno de Gas	2x2	0.196	**0.363**	0.442	-1.13	2.18
	3x3	0.182	**0.312**	0.400	-1.11	2.05
	4x4	0.179	**0.304**	0.389	-0.993	2.06
	5x5	0.177	**0.297**	0.385	-1.02	2.07
	2x2	0.190	**0.342**	0.431	-1.19	2.10
	3x3	0.182	**0.314**	0.403	-1.09	2.12
	4x4	0.177	**0.296**	0.393	-0.996	2.03
	5x5	0.174	**0.286**	0.387	-1.10	2.03
	6x6	0.167	**0.265**	0.365	-0.950	1.79

Tabla 4.28b Comparación del algoritmo propuesto para los datos del horno de gas de Box & Jenkins. Las últimas 5 filas corresponden a usar funciones pseudo-gaussianas libres (las primeras 4 a una partición triangular).

son del tipo Takagi-Sugeno-Kang. De esta forma, cada regla tiene definidos un máximo de 7 parámetros (2 para cada función de pertenencia lineal y 3 para el consecuente), pudiendo, por la tanto, usar un total de 14 parámetros. En nuestro caso, para la configuración 3x3 los parámetros a optimizar son 9 consecuentes de las reglas más 2 centros de funciones de pertenencia (ya que los extremos son fijos), es decir, un total de 11 parámetros. Otro factor a tener en cuenta cuando se construyen sistemas difusos, como se comentó anteriormente, es la interpretabilidad de las reglas obtenidas. Los sistemas con consecuentes del tipo TSK son bastante más complejos de interpretar para un operador humano aunque tienen la característica de poder obtener un modelo bastante fiel utilizando un número muy reducido de reglas difusas. A modo de ejemplo, en la figura 4.50 se muestran los valores originales junto con los predichos por la configuración 3x3 antes mencionada.

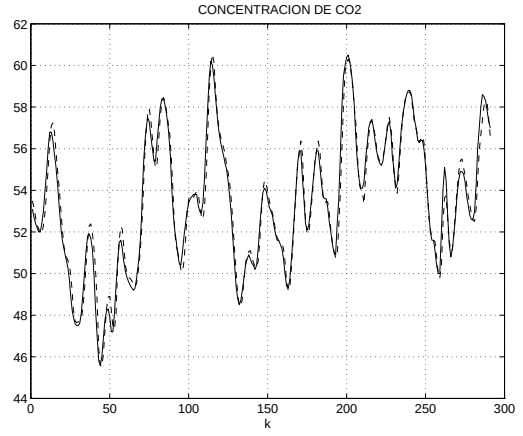


Fig. 4.50 Predicción de la concentración de CO_2 en un horno de gas utilizando una configuración de 3x3 (TP).

Como conclusión, el algoritmo propuesto obtiene unos resultados notablemente superiores a otros algoritmos existentes en la bibliografía para la obtención de sistemas difusos. No sólo se consiguen mejores grados de aproximación para una función dada sino que la complejidad del sistema difuso resultante es menor. Asimismo, la mayoría de algoritmos dan una única solución a un problema dado mientras que la presente metodología proporciona diferentes sistemas con distintos grados de aproximación y complejidad, de modo que el usuario final tiene la libertad de decidirse por uno u otro según las especificaciones del problema a resolver.

4.4.2 Comparación con otros paradigmas

A continuación, evaluaremos el rendimiento del algoritmo propuesto con los resultados proporcionados por otros paradigmas que también son usualmente utilizados para resolver el problema de la aproximación funcional.

En 1981 J.H.Friedman y W.Stuetzle [FRI-81] introdujeron un método adaptativo llamado "Projection Pursuit" (PP) en el que se aproximaba la función deseada utilizando una suma de funciones que eran combinación lineal de las entradas. Posteriormente, en 1991 el propio J.H.Friedman [FRI-91] presentó un método llamado "Multivariate Adaptive Regression Splines" (MARS) en el que adaptativamente se dividía el dominio de entrada en regiones inconexas y modelaba cada región con un valor constante. Un método muy parecido al anterior es el "Constrained Topological Mapping" introducido por V.Cherkassky y H.Lari-Najafi [CHE-91b] donde se usaban mapas auto-organizativos para determinar el particionamiento inicial.

Función	**Propuesto en la bibliografía ECMN**			
	MLP	**PP**	**CTM**	**MARS**
$F(x_1,x_2) = 42.659 \cdot \left(0.1 + x_1\left[0.05 + (x_1^4 - 10x_1^2x_2^2 + 5x_2^4)\right]\right)$ $x_1,x_2 \in [-0.5,0.5]$	0.308	0.504	0.131	0.190
$F(x_1,x_2) = 1.3356 \cdot \left\{\begin{array}{l} 1.5(1-x_1) + e^{(2x_1-1)}sin\left(3\pi(x_1-0.6)^2\right)+ \\ e^{3(x_2-0.5)}sin\left(4\pi(x_2-0.9)^2\right) \end{array}\right\}$ $x_1,x_2 \in [0,1]$	0.096	0.128	0.170	0.063
$F(x_1,x_2) = 1.9 \cdot \left[\begin{array}{l} 1.35 + \\ exp(x_1)sin(13(x_1-0.6)^2)exp(-x_2)sin(7x_2) \end{array}\right]$ $x_1,x_2 \in [0,1]$	0.227	0.206	0.197	0.179

Tabla 4.29a Comparación del algoritmo propuesto con otros paradigmas.

En la tabla 4.29a se presentan los resultados obtenidos por estos paradigmas para tres funciones bidimensionales junto con los resultados proporcionados por los autores para un perceptrón multicapa (MLP) con 15 neuronas en la capa oculta. Los resultados obtenidos por el algoritmo propuesto para estas funciones se reflejan en la tabla 4.29b donde se muestra que siempre existen configuraciones relativamente simples que consiguen un grado de aproximación mejor que todos los métodos anteriores.

Función	Algoritmo propuesto		
	Nº de Reglas	Índice evaluador	
		ECMN	Error Medio
$F(x_1,x_2)=42.659\cdot\left(0.1+x_1\left[0.05+(x_1^4-10x_1^2x_2^2+5x_2^4)\right]\right)$ $x_1,x_2\in[-0.5,0.5]$	4x5	**0.194**	0.151
	6x6	**0.090**	0.075
	8x8	**0.044**	0.037
$F(x_1,x_2)=1.3356\cdot\left\{\begin{matrix}1.5(1-x_1)+e^{(2x_1-1)}sin\left(3\pi(x_1-0.6)^2\right)+\\ e^{3(x_2-0.5)}sin\left(4\pi(x_2-0.9)^2\right)\end{matrix}\right\}$ $x_1,x_2\in[0,1]$	3x5	**0.278**	0.225
	4x6	**0.104**	0.096
	5x9	**0.041**	0.041
$F(x_1,x_2)=1.9\cdot\left[\begin{matrix}1.35+\\ exp(x_1)sin(13(x_1-0.6)^2)exp(-x_2)sin(7x_2)\end{matrix}\right]$ $x_1,x_2\in[0,1]$	4x4	**0.161**	0.147
	5x5	**0.109**	0.082
	7x6	**0.059**	0.044

Tabla 4.29b Comparación del algoritmo propuesto con otros paradigmas.

Uno de los mejores trabajos que se han publicado sobre la comparación de diversos paradigmas para el problema de aproximación funcional fue el realizado por V.Cherkassky, D.Gehring y F.Mulier [CHE-96]. En dicho trabajo se comparan también, aparte de los paradigmas anteriormente descritos, métodos bastante optimizados con redes neuronales artificiales con resultados realmente notables. Para hacernos una idea

Función	[CHE-96] ECMN	
	Aleatorio	Espiral
$F(x_1,x_2)=sin(x_1x_2)$ $x_1,x_2\in[-2,2]$	0.033	0.017
$F(x_1,x_2)=exp(x_1sin(\pi x_2))$ $x_1,x_2\in[-1,1]$	0.016	0.015
$F(x_1,x_2)=\frac{1+sin(2x_1+3x_2)}{3.5+sin(x_1-x_2)}$ $x_1,x_2\in[-2,2]$	0.120	0.052
$F(x_1,x_2)=2.565+1.9\cdot\left[exp(x_1)sin(13(x_1-0.6)^2)exp(-x_2)sin(7x_2)\right]$ $x_1,x_2\in[0,1]$	0.061	0.034
$F(x_1,x_2)=sin\left(2\pi\sqrt{x_1^2+x_2^2}\right)$ $x_1,x_2\in[-1,1]$	0.106	0.049

Tabla 4.30a Comparación del algoritmo propuesto. Resultados recogidos en [CHE-96].

del número de parámetros que utilizaron estos autores en sus simulaciones, los métodos basados en redes neuronales presentaban 40 neuronas en la capa oculta.

En el caso de funciones bidimensionales, que son las que principalmente se utilizan, el número total de parámetros sería de 161 (40*2 +40 pesos entre las neuronas ocultas y la de salida, más (40+1) términos aditivos o "bias"). En la tabla 4.30a se presentan los mejores resultados obtenidos por todos los métodos que utilizaron. Los autores proporcionaron dos tipos de resultados según el tipo de datos de entrenamiento. En el primero de ellos, se escogen 400 datos aleatorios de entrenamiento mientras que, en el segundo, dichos 400 datos se distribuyen en espiral por todo el dominio de entrada. Finalmente, utilizaron 961 datos equidistribuidos (31x31) como patrón de test. En la tabla 4.30b se presentan los resultados proporcionados por el algoritmo propuesto para las mismas funciones. En esta tabla se indica también el tipo de funciones de pertenencia utilizados en cada caso así como el número total de parámetros optimizados. Nuevamente, el algoritmo consigue, en todos los casos, un mejor grado de aproximación con una complejidad inferior aunque, a diferencia de los métodos anteriores presentados en esta sección, las diferencias ya no son tan espectaculares aunque sí bastante significativas.

Función	**Algoritmo propuesto**			
	Config.	**Tipo**	**Nº Parámetros**	**ECMN**
$F(x_1,x_2)=sin(x_1x_2)$ $x_1,x_2\in[-2,2]$	12x12	TP	164	0.017
	7x7	G	73	0.007
	8x8	PGL	104	0.003
$F(x_1,x_2)=exp(x_1sin(\pi x_2))$ $x_1,x_2\in[-1,1]$	9x14	TP	145	0.015
	5x8	PGL	71	0.009
	6x8	G	72	0.008
$F(x_1,x_2)=\frac{1+sin(2x_1+3x_2)}{3.5+sin(x_1-x_2)}$ $x_1,x_2\in[-2,2]$	6x7	G	64	0.061
	7x8	G	82	0.028
	9x9	G	113	0.015
$F(x_1,x_2)=2.565+1.9\cdot$ $\left[exp(x_1)sin(13(x_1-0.6)^2)exp(-x_2)sin(7x_2)\right]$ $x_1,x_2\in[0,1]$	9x9	TP	95	0.035
	8x7	TL	93	0.029
	9x8	TL	115	0.017
$F(x_1,x_2)=sin\left(2\pi\sqrt{x_1^2+x_2^2}\right)$ $x_1,x_2\in[-1,1]$	7x7	PGL	83	0.045
	9x9	TL	127	0.040
	8x8	G	92	0.031

Tabla 4.30b Comparación del algoritmo propuesto con [CHE-96].

Finalmente, volvemos a considerar el problema de la serie temporal de Mackey-Glass que ya se presentó en la sección 4.2.6. Recordemos que la serie se generaba a partir de la ecuación (4-16) y que presentaba un comportamiento aparentemente caótico. Como ya se hizo en dicha sección, los primeros 500 datos serán utilizados como datos de entrenamiento mientras que los 500 restantes se usarán para validar el modelo generado. En la tabla 4.31a se presentan los resultados obtenidos por distintas metodologías para resolver el problema de predecir x_{i+6} a partir de x_i, x_{i-6}, x_{i-12} y x_{i-18} (es decir un problema de 4 entradas y una salida) usando a = 0.2, b = 0.1 y ϑ = 17. Estos resultados se pueden obtener de [CRO-90], [LEE-94], [JAN-97] y [KIM-97]. De entre todos ellos, el proporcionado por el sistema neuro-difuso del ANFIS de Jang destaca especialmente, obteniendo un RECM = 0.007. En la tabla 4.31b se muestran los resultados obtenidos por el algoritmo propuesto utilizando una partición triangular y funciones pseudo-gaussianas libres (las últimas 2 filas) donde se comprueba que, con la configuración de 3x4x4x4 (TP), se consigue igualar el resultado de Jang mientras que tanto la configuración 4x4x5x5 (TP) como la 3x3x3x3 (PGL) consiguen resultados superiores (RECM = 0.0063 y 0.0058 respectivamente).

Método		**Error de Predicción (RECM)**
Modelo Auto-Regresivo		0.19
Correlación en Cascada NN		0.06
Retropropagación NN		0.02
Polinomios de sexto orden		0.04
Método de predicción lineal		0.55
D.Kim & C.Kim (Algoritmo Genético + Sistema Difuso)	5 MFs por variable	0.0492
	7 MFs por variable	0.0423
	9 MFs por variable	0.0379
ANFIS (NN + Lógica Difusa)		0.007
L-X.Wang	T-Norma: Producto	0.0907
	T-Norma: Mínimo	0.0904

Tabla 4.31a Comparación del algoritmo propuesto. Resultados para la serie de Mackey-Glass.

Configuración	**Entrenamiento**		**Chequeo**		**Todos**	
TP	*ECMN*	*RECM*	*ECMN*	*RECM*	*ECMN*	*RECM*
2x3x3x3	0.036	0.0084	0.046	**0.0109**	0.042	0.0097
3x4x4x4	0.022	0.0051	0.030	**0.0071**	0.026	0.0062
4x4x5x5	0.011	0.0026	0.026	**0.0063**	0.021	0.0048
PGL						
2x2x2x2	0.085	0.0197	0.086	**0.0203**	0.085	0.0200
3x3x3x3	0.017	0.0040	0.025	**0.0058**	0.021	0.0050

Tabla 4.31b Comparación del algoritmo propuesto para la serie de Mackey-Glass.

4.5 Conclusión

En este capítulo se han mostrado, individual y colectivamente, las principales características del algoritmo de aproximación de funciones a partir de datos de E/S presentado en el capítulo anterior.

En la sección 4.2 se ha justificado cada una de las etapas de que consta el algoritmo, haciendo especial hincapié en la necesidad de una etapa previa al descenso en gradiente y en el proceso de obtención de las variables más significativas con el fin de mejorar el grado de aproximación obtenido, teniendo siempre en consideración la complejidad estructural del sistema difuso. También se han mostrado ejemplos del comportamiento del algoritmo ante datos ruidosos y de cómo el índice de complejidad-exactitud nos puede ayudar de forma objetiva a elegir entre unas configuraciones y otras aunque su funcionalidad es meramente informativa.

En la sección 4.3 se ha aplicado el algoritmo a otros tipos de funciones de pertenencia (triangulares libres, partición pseudo-gaussiana, gaussianas y funciones pseudo-gaussianas libres). Posteriormente, se ha mostrado cómo adaptar el algoritmo para garantizar la interpretabilidad final de las funciones optimizadas.

Finalmente, en la sección 4.4, se han comparado las prestaciones del algoritmo tanto a nivel de grado de aproximación como de complejidad con respecto a otros algoritmos de extracción de sistemas difusos a partir de datos, y otros paradigmas también comúnmente utilizados para resolver el problema de aproximación funcional. En dichas comparaciones, el algoritmo propuesto es siempre capaz de obtener configuraciones con mayor grado de aproximación y menor complejidad estructural que las proporcionadas en la bibliografía. Los principales motivos de que esto sea así radica, principalmente en que:

- Los consecuentes de las reglas son siempre los óptimos.

- Se introduce una etapa previa al descenso en gradiente capaz de situar el punto inicial cerca de un “buen” mínimo local.

- Se determina la influencia de las distintas variables, pudiéndose así reducir el número total de reglas de forma significativa ya que éste depende de forma exponencial con el número de funciones que particionan cada dominio de entrada.

- El algoritmo no proporciona una única solución a un problema dado sino que obtiene una serie de configuraciones con complejidades y grados de aproximación crecientes. El Índice de Complejidad/Exactitud (ICE) proporciona, para una configuración dada, una medida objetiva del equilibrio entre el grado de aproximación obtenido y la complejidad del sistema difuso, pero su finalidad es meramente informativa a no ser que sea definido de forma exacta por el usuario final. La mayoría de algoritmos de aproximación sólo proporcionan una única configuración sin dar la posibilidad de escoger otras más sencillas o más complejas con otros grados de aproximación. El algoritmo empieza por las configuraciones más sencillas y va añadiendo complejidad al sistema allí donde es más necesario consiguiendo, implícitamente, avanzar en la dirección en la que se pueda disminuir el Índice de Complejidad-Exactitud (ICE).

CAPÍTULO 5

DISEÑO AUTOMÁTICO DE CONTROLADORES DIFUSOS EN TIEMPO REAL

Desde que, en 1975, E. H. Mamdani y S. Assilian [MAM-75] implementaron por primera vez un controlador difuso, los controladores difusos han demostrado ser herramientas considerablemente eficaces para controlar aquellos procesos dinámicos donde obtener un modelo matemático es una tarea ardua, si no imposible. De entre los distintos tipos de algoritmos de control, los controladores difusos adaptativos presentan además la característica de poder ajustar los parámetros que los definen en tiempo real sin la ayuda externa de un operador.

En este capítulo se presenta un nuevo método de adaptación y auto-aprendizaje para controladores difusos operando en tiempo real. Tras la introducción y el planteamiento del problema, en las secciones 5.3 a la 5.7 presentaremos la arquitectura de control utilizada y las principales características del algoritmo de control. En la sección 5.8 se introducirán una serie de mejoras en el algoritmo al considerar las limitaciones del rango del actuador, el error máximo permitido, etc. Finalmente las conclusiones se mostrarán en la sección 5.9.

5.1 Introducción

La implementación de controladores capaces de adaptarse y auto-aprender en tiempo real es una de las cuestiones fundamentales en el área de control [ANT-94]. Mientras que el control difuso no adaptativo ha demostrado su valía en algunas aplicaciones, en aquellas situaciones en las que pueden existir incertidumbres o en las que el sistema a controlar pueda presentar variaciones con el tiempo, el control adaptativo juega un papel fundamental ya que es, en principio, capaz de adecuarse a cambios imprevistos significativos. En [ORD-97] se realiza un análisis comparativo de diversas técnicas de control tanto adaptativo como no adaptativo donde se pone de manifiesto que los métodos adaptativos son mucho más robustos y estables cuando trabajan sobre la implementación real (no simulada) de la planta.

Históricamente, los controladores difusos se han diseñado a partir de la extracción del conocimiento de expertos directamente en forma de reglas difusas. Sin embargo, hay situaciones en las que este conocimiento no está disponible. Además, las reglas difusas generadas de esta forma suelen ser imprecisas e incompletas.

Otro método común para llevar a cabo el diseño de controladores es a partir de un conjunto de datos de E/S que pueden pertenecer a acciones de control reales o a un modelo de la planta a controlar. En el primer caso se requiere que el sistema sea controlado de antemano para poder obtener dichos datos. En el segundo, necesitamos conocer generalmente las ecuaciones diferenciales de la planta o tener un modelo aproximado de la misma. En cualquier caso, cuando se tiene un modelo de la planta en forma de datos de E/S existe otro inconveniente: a partir de esos datos no sabemos qué regiones de operación son más importantes que otras e incluso puede haber datos que cubran estados en los que la planta nunca se encontrará cuando ésta se controle en tiempo real, desperdiciándose así algunos de los parámetros utilizados por el controlador.

Es por todo esto por lo que el método más fiable y óptimo de diseñar un controlador consiste en adaptar sus parámetros en tiempo real según el comportamiento actual de la

planta sin la necesidad de intervención externa, es decir, utilizar controladores adaptativos y/o auto-organizativos.

Dos son las políticas principales de control comúnmente utilizadas en el campo de control adaptativo:

- Control adaptativo **directo**: en el que los parámetros del controlador son ajustados utilizando directamente la información aportada por las variables de entrada y salida de la planta.

- Control adaptativo **indirecto**: en el que se realiza una estimación en tiempo real del modelo de la planta y, basándose en este modelo, los parámetros del controlador son optimizados.

Cuando un controlador difuso está trabajando en tiempo real, hay tres alternativas principales para su adaptación: factores de escala (tanto en la entrada como en la salida), funciones de pertenencia y reglas difusas.

El ajuste de los factores de escala proporciona una mejor sintonización de los dominios de entrada y/o de salida de la función difusa ya que modifican los soportes de cada uno de los valores lingüísticos en igual proporción para cada una de las funciones de una variable. La alteración del factor de escala de la salida proporciona una modificación en el rango del actuador a la planta. El ajuste de los factores de escala ha sido ampliamente estudiado en la bibliografía tanto para ajustar las variables de entrada como la de salida aunque en la mayoría de las veces no se realiza en tiempo real. Como muestra más significativa, en [MAE-92] se evaluaba, una vez concluido el proceso de control, variables importantes como el sobredisparo, el tiempo de subida y el error en el estado estacionario para adaptar los factores de escala según un conjunto de reglas difusas definidas por el autor. Otro método notable que sí realiza el ajuste de los factores en tiempo real es el que se presenta en [CHO-94] donde los factores son regulados de tal forma que la respuesta del sistema ante un escalón en el espacio de las fases tenga forma helicoidal.

La modificación de las funciones de pertenencia suele ir casi siempre acompañada de la modificación de los consecuentes de las reglas difusas por lo que ambos métodos serán tratados conjuntamente. Igualmente, existen numerosos trabajos en la bibliografía donde se adaptan conjuntamente los factores de escala junto con las reglas [MAE-92, ROJ-99].

El primer controlador adaptativo difuso fue propuesto por Procyk y Mamdani [PRO-79] en 1979. En este trabajo se proponía un algoritmo de aprendizaje capaz de generar y modificar las reglas difusas según unos valores de premio o castigo para cada acción de control basándose en el estado actual del sistema. Desde entonces, el método propuesto por estos autores ha sido utilizado con ciertas modificaciones por una gran cantidad de autores [SUG-88b, SHA-88, MAE-92, SIN-98, ROJ-99].

Un notable trabajo sobre controladores difusos tanto directos como indirectos fue el que realizó Li-Xin Wang [WAN-94] para un determinado tipo de sistemas. Basándose esencialmente en el conocimiento de ciertas acotaciones sobre las ecuaciones diferenciales del sistema y en una trayectoria deseada, Wang extrae una política de control en tiempo continuo capaz de seguir la trayectoria deseada de forma razonable y demostrando además la estabilidad del sistema completo de lazo cerrado. Un trabajo relativamente parecido al control indirecto realizado por Wang pero para control discreto fue presentado en [CHE-95] donde el lazo cerrado se utiliza para linealizar el sistema (feedback linearization) y en el que los términos desconocidos de la ecuación diferencial se aproximan en tiempo real mediante descenso en gradiente (de la misma forma que hacía Wang pero para el caso discreto).

Otro método bastante utilizado y que no es ni directo ni indirecto es el control adaptativo basado en un modelo de referencia (MRAC) [LAY-92, FON-93]. La principal ventaja de este procedimiento es que no requiere un conocimiento explícito del modelo de la planta sino que se basa en un modelo de referencia de la misma, es decir, cómo queremos que ésta se comporte.

En 1997, H.C. Andersen et al. [AND-97] propusieron un método de sintonización directa de los parámetros del controlador basado en el error en la salida del controlador en lugar de en la salida de la planta. Para ello, necesitaban la existencia previa de un sistema de control que fuera capaz de controlar la planta de forma relativamente satisfactoria y posteriormente utilizar valores reales de la función inversa de la planta en tiempo real para realizar un ajuste fino de los parámetros.

Finalmente, existen también otros algoritmos de adaptación para arquitecturas híbridas como los controladores neuro-difusos [BER-92, JAN-95, LIN-96] y genético-difusos [HOM-95, LIN-97].

En definitiva, existen numerosas metodologías dirigidas al control adaptativo pero la mayoría de ellas necesitan un pre-entrenamiento y/o el conocimiento de las ecuaciones diferenciales de la planta. Además, ningún método propuesto en la bibliografía es capaz de modificar la estructura del controlador en tiempo real.

En este capítulo se propone una nueva metodología para el diseño automático de controladores difusos en tiempo real. Dicho método es capaz de optimizar, durante el propio proceso de control, tanto las funciones de pertenencia como los consecuentes de las reglas difusas sin la necesidad de un modelo de la planta (control directo). Además, la topología del controlador podrá ser modificada a través de un análisis global de la política de control y se podrán seleccionar las variables que más influyan en el sistema, todo ello en tiempo real y sin apenas la necesidad de un conocimiento previo cuantitativo de la planta a controlar.

La organización de este capítulo será la siguiente:

En la sección 5.2 se planteará el problema a resolver de forma concreta. Posteriormente, se presentará la arquitectura del controlador difuso con la que pretendemos abordar dicho problema, resaltando los principales bloques funcionales de los que consta. El ajuste de los parámetros del controlador principal se realiza por medio de dos etapas correlativas cuyo uso combinado aumenta la velocidad de aprendizaje y explota toda la

información que se puede obtener durante el proceso de control. La primera, es una etapa de adaptación de los consecuentes de las reglas y de las posiciones centrales de las funciones de pertenencia basada en el error de la salida de la planta (sección 5.4). La segunda etapa, más exacta, que se encargará del ajuste fino de todos los parámetros del controlador utilizando el error en la salida del controlador se detallará en la sección 5.5. En la sección siguiente (5.6) se describe el módulo encargado de la modificación de la topología del controlador para mejorar el rendimiento del mismo, incluyendo la posibilidad de añadir variables que todavía no se estaban utilizando. Finalmente, se presenta un resumen y un organigrama del algoritmo de control en la sección 5.7. En la segunda parte del capítulo se incluyen algunas mejoras introducidas en la metodología presentada para tener en cuenta cuestiones como la limitación del rango del actuador (sección 5.8.1), el error máximo permitido especificado por el usuario (sección 5.8.2) y la adecuación del algoritmo a otros tipos de funciones de pertenencia (sección 5.8.3). Por último, en la sección 5.9 se presentan las conclusiones del capítulo.

5.1.1 Algunas notas sobre la estabilidad del proceso de control

Siempre que se habla del control de un proceso hay que tener en cuenta el problema de la estabilidad del sistema total en lazo cerrado (controlador + planta). Cuando el sistema a controlar es lineal, la teoría clásica proporciona una gran cantidad de métodos para el diseño de sistemas de control estables (lugar de las raíces, criterio de estabilidad de Nyquist, diagramas de Bode, etc.). Para procesos no lineales, la demostración de que el sistema final en lazo cerrado vaya a ser estable es mucho más compleja y hay que echar mano del teorema principal de estabilidad de Lyapunov [LYA-92] que se basa en el conocimiento de las ecuaciones diferenciales que rigen el sistema y de la búsqueda de una función escalar del estado del sistema con primeras derivadas parciales continuas con la propiedad de que sea definida positiva y su derivada definida negativa.

Existen métodos en la bibliografía que utilizan el teorema de Lyapunov para demostrar la estabilidad del proceso de control [WAN-94, CHE-95b] pero siempre basados en el conocimiento de la forma de las ecuaciones diferenciales y en acotaciones de sus términos.

En el método propuesto en este capítulo se suponen desconocidas las ecuaciones diferenciales del proceso a controlar y sólo se requiere un conocimiento mínimo en términos de la monotonía existente entre la salida de la planta y la señal de control y el retraso de la planta. Es por ello, que el algoritmo que se presenta tiene un rango de aplicación mucho mayor que los métodos anteriormente comentados pero asimismo, aunque se demuestra la estabilidad local de cada una de las partes del algoritmo, no se puede realizar ningún análisis global de la estabilidad del sistema final al no conocerse las ecuaciones que rigen el sistema a controlar.

5.2 Planteamiento del problema

Nuestro objetivo será lograr el control en tiempo real de un sistema cuyas ecuaciones desconocemos y la optimización de dicho control.

En nuestro planteamiento, trataremos de optimizar las reglas del controlador, el número de funciones de pertenencia en cada una de las variables de entrada y los parámetros que las definen con el propósito de poder llevar el estado de la planta al valor deseado en el menor tiempo posible.

El sistema o planta a controlar se expresa generalmente por sus ecuaciones diferenciales o, equivalentemente, por sus ecuaciones en diferencias siempre que éstas sean obtenidas de aquéllas usando un periodo de muestreo suficientemente pequeño. Matemáticamente tendremos:

$$x^{(n)} = f(\vec{x},u) = f(x,\dot{x},...,x^{(n-1)},u) \qquad y = h(\vec{x}) \tag{5-1}$$

donde $\vec{x}$ es un vector que representa el estado de la planta, y es la salida de la misma, u es la señal de control y f y h son funciones continuas. Equivalentemente, si discretizamos las ecuaciones diferenciales tenemos:

$$y(k) = f'(y(k-1),...,y(k-p'),u(k-d),...,u(k-d-q')) \tag{5-2}$$

que siempre se puede expresar como (ver Apéndice D)

$$y(k+d) = f(y(k),...,y(k-p),u(k),...,u(k-q)) \tag{5-3}$$

donde d es el retraso de la planta y f es una función continua.

Las restricciones mínimas y evidentes que se suelen imponer a estos sistemas es que sean controlables, es decir, que siempre exista una política de control capaz de llevar la salida hacia el valor deseado (dentro del rango de operación). Esto se traduce en que no debe existir ningún estado en el que la variable de salida no dependa de la entrada de control. Como consecuencia de ello, por ser las plantas continuas respecto a todas sus variables, la salida variará de forma monótona con la señal de control *u*. Si esto no ocurriera, por continuidad, habría siempre al menos un estado en el que la salida no dependiera de la entrada de control. De esta manera, suponemos que existe una función *F* tal que la señal de control dada por:

$$u(k) = F\left(\vec{x}(k)\right) \tag{5-4}$$

con $\vec{x}(k) = (r(k), y(k), \ldots, y(k-p), u(k-1), \ldots, u(k-q))$, siendo $r(k)$ la salida deseada en el instante k, que hace que $y(k+d) = r(k)$.

En el algoritmo propuesto en este capítulo no necesitamos información sobre las ecuaciones que rigen la planta aunque sí es necesario conocer el signo de su monotonía y el retardo que presenta, es decir, cuánto tiempo tiene que pasar para que la entrada de control actual $u(k)$ influya en la salida.

Al igual que en capítulos anteriores, utilizaremos el tipo de controladores difusos definidos en la sección 2.8 del capítulo 2. De esta forma, la salida de nuestro controlador difuso viene dada por la ecuación 2-21, que aquí recogemos de nuevo:

$$u(k) = \hat{F}\left(\vec{x}(k)\right) = \frac{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(R_{i_1 i_2 \ldots i_N} \cdot \prod_{m=1}^{N} \mu_{X_m^{i_m}}\left(x_m(k)\right)\right)}{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(\prod_{m=1}^{N} \mu_{X_m^{i_m}}\left(x_m(k)\right)\right)} \tag{5-5}$$

cuyo valor dependerá del conjunto de reglas y funciones de pertenencia que se definan.

Así pues, nuestro objetivo se puede resumir en diseñar un algoritmo de control capaz de auto-diseñar un controlador que aprenda en tiempo real a controlar un determinado sistema y optimice sus parámetros de tal modo que, para un estado actual de la planta dado, encuentre la señal de control $u(k)$ capaz de hacer que, tras el retardo de la planta, la salida $y(k+d)$ se aproxime al valor deseado $r(k)$ dentro de los límites que el usuario

especifique. Dependiendo de dichos límites se necesitarán más o menos funciones de pertenencia en el controlador y, consecuentemente, mayor o menor número de parámetros. En lo que sigue, supondremos una partición triangular como modelo de funciones de pertenencia. En la sección 5.8.3 se mostrará cómo adaptar el algoritmo propuesto a otros tipos de funciones definidos en el Apéndice A.

5.3 Arquitectura de control

En la figura 5.1 se muestra el diagrama de bloques de la arquitectura de control que se utilizará en este capítulo. El bloque marcado en gris es el controlador principal donde se han resaltado los sub-bloques que contienen los parámetros que definen las funciones de pertenencia y los consecuentes de las reglas. El resto de los bloques son sistemas o controladores auxiliares encargados de la adaptación de dichos parámetros.

- El controlador auxiliar 1 (ca_1) es el encargado de la adaptación de los consecuentes de las reglas utilizando como información el error en la salida de la planta. Dicho sistema auxiliar será el encargado principal de la adaptación en la primera etapa del

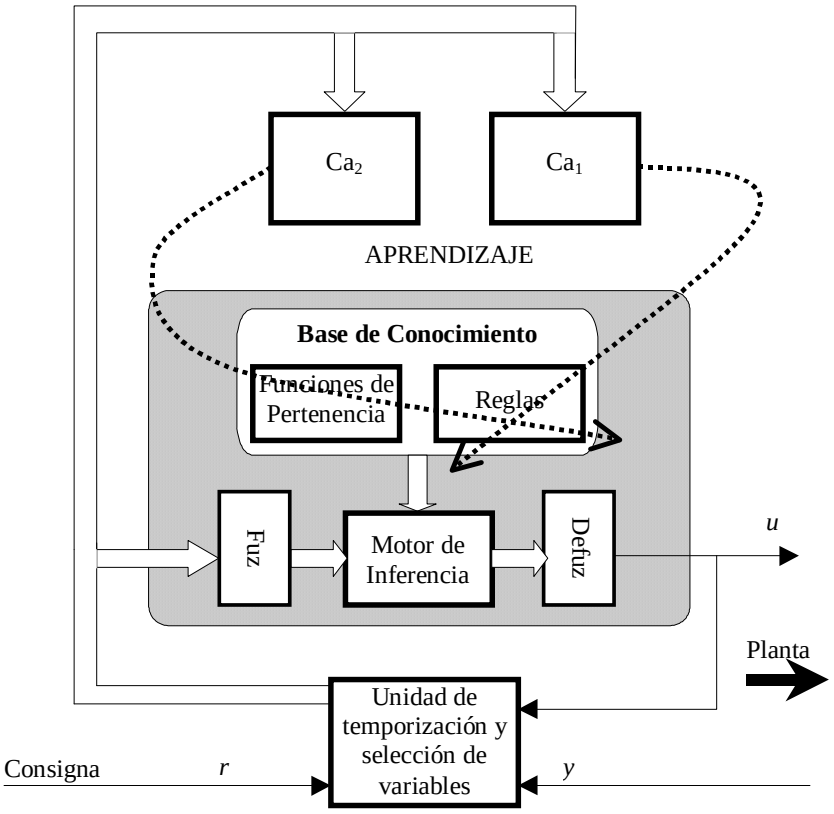


Fig. 5.1 Diagrama de bloques de la arquitectura de control propuesta.

algoritmo mientras que en la etapa de sintonización fina actuará como supervisor.

- El controlador auxiliar 2 (ca_2) tomará el control del proceso de adaptación en la segunda etapa del algoritmo (la encargada del ajuste fino). Dicho sistema se basa en el error en la salida del controlador y podrá sintonizar tanto los consecuentes de las reglas como las funciones de pertenencia del controlador principal.

- Finalmente, la unidad de temporización y selección de variables se encargará de conmutar entre las diversas etapas del algoritmo y será la encargada de suministrar las variables de entrada tanto al controlador principal como a los dos sistemas auxiliares. También será el encargado de almacenar datos reales de la función inversa de la planta para determinar, en su momento, cómo modificar la topología del controlador principal con la finalidad de mejorar su rendimiento.

En la figura 5.2 se muestra un organigrama del algoritmo de control propuesto en este capítulo. El algoritmo comenzará con una estructura inicial del controlador principal muy sencilla (incluso vacía) y se encargará de modificar tanto la topología como los parámetros del controlador siempre en tiempo real. El periodo global de control T' indicará cada cuanto tiempo la unidad de temporización comprobará si se ha llegado a la condición necesaria para conmutar entre las diversas etapas del algoritmo. Dicho periodo dependerá principalmente del periodo de muestreo y de la variación de las consignas. A diferencia de muchos otros métodos de control propuestos en los que sólo se tiene en cuenta el comportamiento ante un escalón simple, en este trabajo la consigna puede variar incluso aleatoriamente, indicando T' una estimación de cada cuanto tiempo se puede evaluar el proceso de control completo. En el capítulo siguiente se comprobará que la influencia de este parámetro no es significativamente crítica en la obtención del diseño final del controlador principal siempre y cuando dicho parámetro sea lo suficientemente alto.

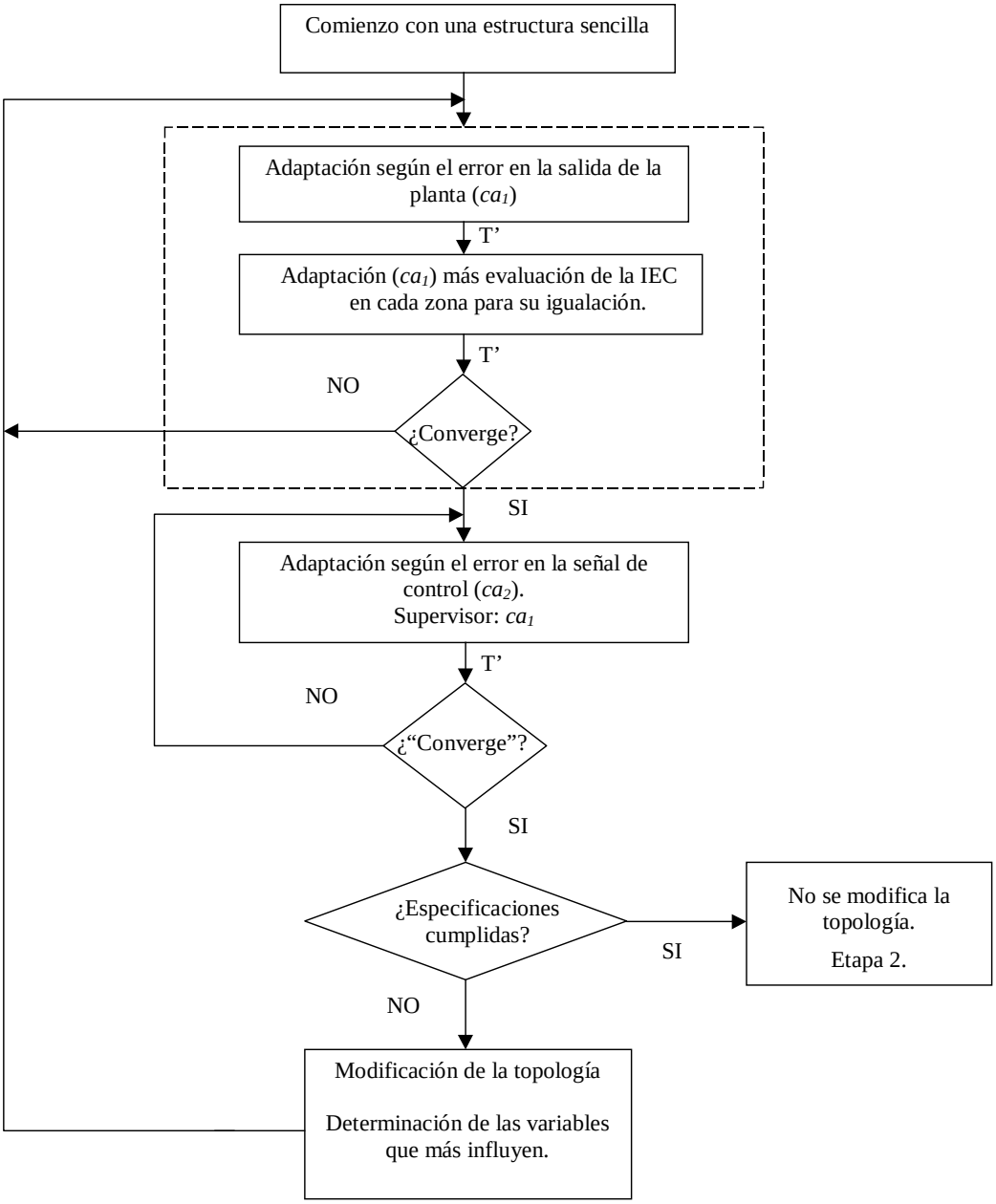


Fig. 5.2 Organigrama del algoritmo propuesto.

5.4 Primera etapa: Adaptación mediante el error en la salida de la planta

En este apartado se presentará la parte del algoritmo encargada de la adaptación de los parámetros del controlador a partir de la información obtenida por el error actual en la salida de la planta. La metodología presentada en esta sección podría ser utilizada

directamente como un algoritmo completo para la determinación tanto de los consecuentes de las reglas como de los valores centrales de las funciones de pertenencia del controlador principal en tiempo real. Sin embargo, en este capítulo se utilizará como una etapa previa de obtención de parámetros iniciales para posteriormente ser ajustados de forma fina por una segunda etapa descrita en la sección siguiente.

5.4.1 Adaptación de los consecuentes de las reglas

El problema principal en los algoritmos de control en tiempo real estriba en que, al desconocer el funcionamiento interno del sistema a controlar, no sabemos cómo modificar los parámetros del controlador a partir del error de salida. Si quisiéramos usar un algoritmo de descenso en gradiente necesitaríamos computar la derivada parcial de la salida de la planta con respecto a la señal de control para un estado dado ($\partial y/\partial u$), derivada que desconocemos. Además, en el caso de periodos de muestreos relativamente altos no se puede aproximar dicha derivada por $\Delta y/\Delta u$ por lo que el problema se vuelve complejo.

Sin embargo, como se comentó en la sección 5.2, sabemos que para que la planta sea controlable dicha derivada parcial debe tener un signo definido constante. De esta forma podemos hacer uso de la información sobre la monotonía de la planta para poder obtener una dirección adecuada en la que mover los consecuentes de nuestras reglas utilizando un procedimiento cualitativamente similar al ya propuesto inicialmente por Procyk y Mamdani en 1979. Así, en una planta con retardo menor que el tiempo de muestreo (es decir, en la que la salida en el instante *k+1* es consecuencia directa de la entrada de control en el instante anterior), si la señal de control *u(k)* nos ha proporcionado una salida *y(k+1) > r(k)*, sabemos que se debería haber usado una señal menor, en el caso de que la salida crezca con la entrada de control (alternativamente, deberíamos haber usado *u(k)* mayor si la monotonía tuviera el signo contrario).

De esta forma, la monotonía de la planta nos da la información valiosa de en qué dirección adaptar los consecuentes de nuestras reglas. Para modificar dichos consecuentes, como es lógico, sólo se deben tener en cuenta los de aquellas reglas que

hayan intervenido para la obtención de *u(k)* como salida del controlador. Debido al carácter local de las reglas difusas, esta adaptación implica auto-aprendizaje.

De ello se deduce de forma natural la necesidad de un controlador auxiliar, que denotaremos por ca_1, que se encargue de evaluar el estado actual de la planta y cuya salida sea la corrección de las reglas responsables de dicho estado a modo de premio/castigo. Matemáticamente, si llamamos F_{ca_1} a la salida de dicho controlador auxiliar, la variación de la regla *i*-ésima en el instante $k+d$ vendrá dada por:

$$\Delta R_i(k+d) = \alpha_i(k) \cdot F_{ca_1}\left(e(k+d)\right) \tag{5-6}$$

donde $\alpha_i(k)$ es el grado de activación de la regla en el instante *k*, y $e_y(k+d)$ es el error cometido en la salida en el instante $k+d$ y viene dado por:

$$e(k+d) = r(k) - y(k+d) \tag{5-7}$$

ya que *r(k)* es el valor deseado en el instante *k*. Se debe hacer notar que sería incorrecto considerar *r(k+d)* en la expresión anterior ya que las reglas que se activaron en el instante *k* lo hicieron para conseguir el valor deseado *r(k)* y no *r(k+d)*.

Un factor muy importante a tener en cuenta es que la modificación impuesta por el sistema auxiliar ca_1 debe ser proporcional al grado de activación de cada una de las reglas responsables. De esta forma, cuando el estado de la planta cambie a una región vecina, aquellas reglas antes activadas en mayor grado ya no lo estarán tanto, evitando de esta forma la pérdida de información obtenida en el instante anterior. Por ello, se ha utilizado en la expresión (5-6) el grado de activación $\alpha_i(k)$ de la regla *i*-ésima en el instante *k*.

Debido a que hay que esperar hasta el instante $k+d$ para poder evaluar la señal de control *u(k)*, para poder implementar este método de adaptación es necesario definir una cola de tamaño igual al retardo de la planta donde almacenaremos los grados de activación de las reglas.

El controlador auxiliar ca_1 puede ser implementado de diversas formas, siendo una posibilidad el hacerlo mediante un sistema difuso. A continuación se hará un análisis de

las características que debe reunir tal sistema auxiliar para proporcionar un cambio que provoque una disminución en el error de la salida de la planta y, de esta forma, su convergencia a cero.

5.4.1.1 Requisitos exigidos al sistema auxiliar ca_1

Supongamos un sistema a controlar regido, de forma genérica, por la ecuación en diferencias dada por la ecuación (5-3) y un controlador difuso de la forma descrita en (5-5). En el instante *k*, la salida de dicho controlador será (ver (5-4)):

$$u_k(1) = \hat{F}(\vec{x}_k, \Theta_k(1)) \tag{5-8}$$

donde Θ_k es el conjunto de parámetros del controlador principal en el instante *k*. De esta forma, la salida de la planta *d* instantes de tiempo después será:

$$y_{k+d}(1) = f(\vec{x}_k, u_k(1)) \tag{5-9}$$

obteniéndose un error en la salida de la planta en el instante $k+d$:

$$e_{k+d}(1) = r_k - y_{k+d}(1) \tag{5-10}$$

Dicho error es el que evalúa el controlador auxiliar ca_1 para modificar las reglas responsables de la obtención de la señal de control u_k. Es lógico pensar que la corrección dada por dicho sistema auxiliar sea proporcional al error actual de la planta. De este modo, inicialmente supondremos que:

$$F_{ca_1}(e(k+d)) = C \cdot e(k+d) \tag{5-11}$$

siendo *C* un valor que puede depender de otros factores pero cuyos límites debemos determinar con el fin de asegurar que el proceso de adaptación nos dirija a una disminución del error en la salida. Lo que sí sabemos es que si la monotonía de la planta es creciente, el valor de *C* debe ser positivo ya que la corrección debería tener el mismo signo que el error en este caso. Inversamente, *C* debe ser negativo en el caso de monotonía decreciente. En adelante, supondremos, sin pérdida de generalidad, que la monotonía es creciente dejando para el final la extracción de conclusiones para el caso contrario.

Tras la adaptación aconsejada por el sistema auxiliar ca_1, las reglas del controlador principal sufren, por lo tanto, el siguiente cambio:

$$\Delta R_i(k+d) = \alpha_i(k)\cdot C\cdot e(k+d) \tag{5-12}$$

Para evaluar si dicho cambio ha mejorado el comportamiento del controlador principal, volvamos a las mismas condiciones que había en el instante *k* pero ahora utilizando los nuevos valores de las reglas. En ese caso tendríamos que la nueva señal de control sería:

$$u_k(2) = \hat{F}(\vec{x}_k, \Theta_k(2)) \tag{5-13}$$

donde $\Theta_k(2)$ simboliza los nuevos parámetros del controlador principal (los adaptados por el sistema auxiliar ca_1). Esta nueva señal de control produciría una nueva salida en la planta *d* instantes de tiempo después:

$$y_{k+d}(2) = f(\vec{x}_k, u_k(2)) \tag{5-14}$$

siendo su error asociado

$$e_{k+d}(2) = r_k - y_{k+d}(2) \tag{5-15}$$

Pues bien, el problema que queremos ahora resolver es entre qué valores puede oscilar el valor de *C* para asegurarnos la convergencia del proceso de adaptación, es decir, para que:

$$|e_{k+d}(2)| \le |e_{k+d}(1)| \tag{5-16}$$

verificándose la igualdad si y sólo si $e_{k+d}(2) = e_{k+d}(1) = 0$.

Suponiendo que $f(\vec{x}_k, u_k)$ es continua y derivable con respecto a u_k y teniendo en cuenta que el intervalo $[u_k(1), u_k(2)]$ es cerrado y acotado, por el teorema del valor medio sabemos que existe un valor $\tilde{u}$ en dicho intervalo tal que:

$$y_{k+d}(2) = y_{k+d}(1) + \frac{\partial f}{\partial u}(\vec{x}, u)\Big|_{\vec{x}_k, \tilde{u}} \cdot (u_k(2) - u_k(1)) \tag{5-17}$$

de modo que

$$e_{k+d}(2) = e_{k+d}(1) - \frac{\partial f}{\partial u}(\vec{x}, u)\Big|_{\vec{x}_k, \tilde{u}} \cdot (u_k(2) - u_k(1)) \tag{5-18}$$

siendo $\frac{\partial f}{\partial u}(\vec{x}, u)\Big|_{\vec{x}_k, \tilde{u}}$ un número real positivo (distinto de cero) ya que hemos supuesto que la monotonía de la planta era creciente.

Por otro lado, tendremos que:

$$\begin{aligned} &\text{Si } e_{k+d}(1) > 0 \quad \rightarrow \quad u_k(1) \le u_k(2) \le u_k(1) + C\cdot e_{k+d}(1) \\ &\text{Si } e_{k+d}(1) < 0 \quad \rightarrow \quad u_k(1) + C\cdot e_{k+d}(1) \le u_k(2) \le u_k(1) \end{aligned} \tag{5-19}$$

ya que el máximo cambio en la salida del controlador tras el proceso de adaptación se producirá cuando una regla se active con grado máximo con lo que en la salida se verá reflejada la corrección completa del sistema auxiliar ca_1. De esta forma

$$\begin{aligned} &\text{Si } e_{k+d}(1) > 0 \quad \rightarrow \quad 0 \le u_k(2) - u_k(1) \le C\cdot e_{k+d}(1) \\ &\text{Si } e_{k+d}(1) < 0 \quad \rightarrow \quad C\cdot e_{k+d}(1) \le u_k(2) - u_k(1) \le 0 \end{aligned} \tag{5-20}$$

Finalmente, extrayendo valores absolutos de (5-18)

$$\left|e_{k+d}(2)\right| = \left|e_{k+d}(1) - \left(\frac{\partial f}{\partial u}(\vec{x},u)\Big|_{\vec{x}_k,\tilde{u}}\right)\cdot\left(u_k(2) - u_k(1)\right)\right| \tag{5-21}$$

Por ser el signo de la parcial existente en la expresión anterior constante, los valores extremos que se obtendrán corresponden a los valores límite de $u_k(2) - u_k(1)$. En el caso en el que $u_k(2) = u_k(1)$ tendremos que $\left|e_{k+d}(2)\right| = 0$, es decir, un error cero en la salida. Pero esto sólo se dará si $e_{k+d}(1) = 0$, ya que, en caso contrario, el controlador auxiliar propondría un cambio distinto de cero y la salida ya no podría ser igual si se repitieran las mismas condiciones.

En el otro caso límite ($u_k(2) - u_k(1) = C\cdot e_{k+d}(1)$) tendríamos:

$$\left|e_{k+d}(2)\right| = \left|e_{k+d}(1) - \left(\frac{\partial f}{\partial u}(\vec{x},u)\Big|_{\vec{x}_k,\tilde{u}}\right)\cdot C\cdot e_{k+d}(1)\right| = \left|e_{k+d}(1)\right|\cdot\left|1 - \left(\frac{\partial f}{\partial u}(\vec{x},u)\Big|_{\vec{x}_k,\tilde{u}}\right)\cdot C\right| \tag{5-22}$$

y se cumpliría la condición (5-16) si y sólo si

$$\left|1 - \left(\frac{\partial f}{\partial u}(\vec{x},u)\Big|_{\vec{x}_k,\tilde{u}}\right)\cdot C\right| \le 1 \quad \Leftrightarrow \quad -1 \le 1 - \frac{\partial f}{\partial u}(\vec{x},u)\Big|_{\vec{x}_k,\tilde{u}}\cdot C \le 1 \tag{5-23}$$

es decir

$$\frac{\partial f}{\partial u}(\vec{x},u)\Big|_{\vec{x}_k,\tilde{u}}\cdot C \ge 0 \tag{5-24}$$

$$\left.\frac{\partial f}{\partial u}(\vec{x},u)\right|_{\vec{x}_k,\tilde{u}} \cdot C \leq 2$$

La primera condición nos dice que el parámetro *C* debe tener el mismo signo que la monotonía de la planta. En el caso de monotonía creciente, *C* debe ser positivo, extremo al que ya habíamos llegado anteriormente de forma razonada. La segunda condición es la realmente restrictiva y nos dice que si el sistema auxiliar ca_1 viene definido por la expresión (5-11) y las reglas se modifican según (5-12), la condición (5-16), que nos garantiza la bondad del cambio realizado, se cumplirá siempre que

$$C \leq \frac{2}{\left.\frac{\partial f}{\partial u}(\vec{x},u)\right|_{\vec{x}_k,\tilde{u}}} \tag{5-25}$$

Como queremos que esto suceda para cualquier estado de la planta deberemos imponer que

$$C \leq \frac{2}{\left.\frac{\partial f}{\partial u}(\vec{x},u)\right|_{máx}} \tag{5-26}$$

El caso de monotonía descendente tiene un tratamiento prácticamente idéntico al utilizado con la única diferencia ahora de que tanto *C* como el valor la parcial son negativos. En el caso general tendremos que la condición que debe reunir *C* para la convergencia del proceso es que

$$|C| \leq \frac{2}{\left|\frac{\partial f}{\partial u}(\vec{x},u)\right|_{máx}} \tag{5-27}$$

Evidentemente los casos límite obtenidos pertenecen a las situaciones en las que $e_{k+d}(2) = e_{k+d}(1)$ y $e_{k+d}(2) = -e_{k+d}(1)$. Aunque el sistema puede ser fuertemente alineal, es lógico optar por un valor medio:

$$|C| = \frac{1}{\left|\frac{\partial f}{\partial u}(\vec{x},u)\right|_{máx}} \tag{5-28}$$

Hallar dicho valor, sin embargo, no es fácil pero si se conoce el proceso físico que se está tratando se podría hacer una estimación de su valor calculando cuál es el cambio máximo que se podrá obtener en la salida *d* instantes de tiempo después ante un cambio

pequeño en la señal de control. Recordemos que siempre que se realiza un proceso de control se debe tener una estimación del rango de variación de la variable a controlar y del rango del actuador que se vaya a utilizar. Utilizando estos valores siempre se podrá obtener una estimación por exceso del valor de dicha parcial.

5.4.1.2 Aceleración del proceso de convergencia

Finalmente, una vez que se conocen los límites de actuación del sistema auxiliar ca_1, podemos intentar acelerar el proceso de convergencia para el caso de consignas constantes (como sucede con las funciones escalón u otras de variación lenta). Tras el análisis anterior sabemos que si representamos la salida de ca_1 de la forma

$$F_{ca_1}\left(e(k+d)\right) = \frac{C'}{\left|\frac{\partial f}{\partial u}(\vec{x},u)\right|_{máx}} \cdot e(k+d)$$

tenemos la convergencia asegurada siempre que $C' \in [0,2]$. Por otro lado, la planta puede ser fuertemente alineal y la derivada máxima ser grande comparada con las derivadas en otros estados de la planta en los que, evidentemente, los cambios aconsejados por el controlador auxiliar podrían resultar pequeños y ralentizar el proceso de convergencia. Para estos casos todavía podemos utilizar el rango permitido de C' y usar la información de la derivada del error en nuestro provecho. En estos casos, los cambios que se llevarán a cabo en los consecuentes de las reglas vendrán dados por:

$$\Delta R_i(k+d) = \alpha_i(k) \cdot \frac{F_{ca_1}\left(e(k+d),\dot{e}(k+d)\right)}{\left|\frac{\partial f}{\partial u}(\vec{x},u)\right|_{máx}} \cdot e(k+d) \qquad (5\text{-}29)$$

donde ahora el controlador auxiliar ca_1 será un sistema difuso con una salida entre 0 y 2. Su definición es muy sencilla. Por ejemplo, si el error actual es grande y positivo y el error en el instante anterior es parecido al error actual, entonces el valor debe tender al valor máximo (2) ya que nos estamos acercando muy lentamente. En el caso en que sean de parecidos valores absolutos pero de signo contrario la salida debe disminuirse ya que en esa zona el sistema es muy sensible. Todo esto sería válido, como es lógico, sólo para consignas fijas. En cualquier otro caso la salida será siempre 1.

En definitiva, el controlador auxiliar ca_1 es esencialmente un evaluador de la salida de la planta que proporciona un valor corrector a los consecuentes de las reglas. Además,

se puede construir a partir de información reducida de la planta (esencialmente la monotonía y el retraso de la misma) y de una estimación de la variación máxima de la salida del sistema con respecto a la señal de control. En el capítulo siguiente se mostrarán ejemplos de cómo definir este controlador y de su funcionamiento.

5.4.2 Movimiento de las funciones de pertenencia

Durante la primera etapa del algoritmo, el controlador auxiliar ca_1 recién presentado se encarga de la adaptación de los consecuentes de las reglas pero con la metodología introducida no se pueden modificar también las funciones de pertenencia. En esta sección se abordará una forma de realizar esto último utilizando como información el error en la salida de la planta. Recordemos que al no tener información de cómo varía ésta con respecto a la señal de control en cada región de operación, no podremos utilizar ningún algoritmo basado en el gradiente para acometer esta tarea.

En analogía con la etapa previa introducida en el capítulo 3, la idea básica que usaremos ahora es que *no debe existir ninguna región de operación peor controlada que otras*. Al actuar el controlador en tiempo real, algunas regiones de operación serán más importantes que otras por lo que, aunque el error sea pequeño en ellas, debido a su mayor uso el error acumulado podría llegar a ser alto y, por tanto, indeseable. De esta manera, la información que usaremos para mover los centros de las funciones de pertenencia será la del criterio del error cuadrático durante un periodo global de control completo T'. Aunque utilicemos una configuración de funciones de pertenencia que no sea una partición triangular, en esta etapa fijaremos los valores extremos de cada una de ellas tal y como se hizo en la etapa previa presentada en la sección 3.4.2 del capítulo 3.

La idea que se propone para ello, por tanto, es encontrar una configuración de funciones de pertenencia que equidistribuya un determinado criterio de evaluación (como la integral del error cuadrático, IEC) en cada una de las zonas delimitadas por las funciones de pertenencia (regiones de operación) teniendo en cuenta el comportamiento real de la planta, es decir, sin realizar ninguna media en cada zona. Las zonas activadas más veces sufrirán mayores contribuciones que deberán ser compensadas por los errores mayores de las regiones menos utilizadas. Si existen estados en los que la planta nunca

se encuentre, la contribución de su zona asociada será cero por lo que el algoritmo tenderá a desplazar las funciones de pertenencia a otras zonas más necesitadas e importantes.

De esta forma, podemos asociar al centro c_v^j una "pendiente" p_v^j que será la diferencia entre la contribución al criterio de la integral del error cuadrático del tramo que le precede y la del tramo que le sigue[1]:

$$p_v^j = \frac{1}{r_y}\left(\int_t^{t+T'} e^2(\vec{x}^k)/x_v^k \in \left[c_v^{j-1}, c_v^j\right[- \int_t^{t+T'} e^2(\vec{x}^k)/x_v^k \in \left[c_v^j, c_v^{j+1}\right[\right) \tag{5-30}$$

Para que los parámetros que se utilicen en esta parte del algoritmo sean generales y no dependan de factores de escala, hemos normalizado los valores de las pendientes por el rango de la variable de salida r_y valor que conocemos ya que sabemos el rango de variación de las consignas que vamos a introducir en el sistema.

Así, un valor positivo de esta pendiente indica que la contribución del tramo de la izquierda es mayor que el de la derecha por lo que el centro debe moverse a la izquierda para contrarrestar este efecto. Al igual que hicimos en el capítulo 3, como no debemos permitir que el orden de los centros varíe, una posible forma de realizar este movimiento sería:

$$\Delta c_v^j = \begin{cases} \dfrac{c_v^{j-1} - c_v^j}{b} \dfrac{p_v^j}{p_v^j + \dfrac{1}{T_v^j}} & \text{si } p_v^j \geq 0 \\ \dfrac{c_v^{j+1} - c_v^j}{b} \dfrac{\left|p_v^j\right|}{\left|p_v^j\right| + \dfrac{1}{T_v^j}} & \text{si } p_v^j < 0 \end{cases} \tag{5-31}$$

donde se ha utilizado de nuevo el "radio de acción" b y la temperatura T_v^j asociada al centro c_v^j (ver sección 3.4.2).

Existen, sin embargo, dos diferencias principales con lo expuesto en el capítulo 3:

[1] En la metodología presentada, los centros de las funciones de los extremos están fijados a los valores mínimo y máximo de cada variable, por lo que esta expresión sólo se aplica para los centros interiores.

- Ahora no contamos con un método de obtención directa de las reglas sino que este proceso se realiza en tiempo real a través del sistema auxiliar ca_1. Por ello, los movimientos de los centros no han de ser demasiado grandes por lo que el radio de acción deberá ser más alto (por ejemplo $b = 5$ indicando que, como mucho, nos desplazaremos hasta una quinta parte de la distancia que nos separa del centro vecino).

- Tampoco tenemos ahora un conjunto fijo de datos que nos garanticen la convergencia del proceso. Si bien el periodo global de control T' debe ser lo suficientemente alto como para poder evaluar globalmente la política de control completa, esto no es suficiente como para asegurar una convergencia. De esta forma, en el algoritmo se irán adaptando las temperaturas de forma que cuando haya un cambio de pendiente éstas decrezcan pero sin permitir el proceso inverso. Así, siempre llegará un momento en el que, o bien los errores serán todos lo suficientemente parecidos o las temperaturas serán lo suficientemente bajas como para que los centros apenas se muevan y el proceso finalice. Una condición típica de convergencia es que ningún centro se mueva más allá de un cierto tanto por ciento del rango de su variable. Dicha condición no tiene por qué ser muy restrictiva, ya que esta etapa se utilizará como fase previa al proceso de sintonización fina que se abordará en la sección 5.5.

5.4.3 Temporización de la primera etapa

En el organigrama de la figura 5.2 se puede observar la temporización de esta primera etapa. Como ya se comentó anteriormente, hay que tener en cuenta que cada vez que se realiza un cambio en las funciones de pertenencia, los consecuentes de las reglas no se pueden obtener en un solo paso sino que deben ir ajustándose a los nuevos valores de forma progresiva. De esta forma, no es lógico contabilizar en la integral del error cuadrático los errores pertenecientes a los instantes inmediatamente posteriores al cambio de las funciones ya que los consecuentes todavía no se han ajustado a dicho cambio. Por ello, en el algoritmo se utiliza la evaluación del error cuadrático de forma alternada dejando siempre un periodo T' completo para la adaptación de las reglas a los nuevos valores de las funciones de pertenencia.

5.5 Segunda etapa: Adaptación mediante el error en la señal de control

En el apartado anterior se ha puesto de manifiesto la dificultad de usar un algoritmo de descenso en gradiente a partir del error en la salida de la planta debido a la imposibilidad de computar $\partial y/\partial u$ por ser desconocido el funcionamiento interno de la misma. En esta sección utilizaremos un método de descenso en gradiente basado en el error en la señal de control en lugar del producido en la salida de la planta. De esta manera, los valores de los parámetros obtenidos en la etapa anterior serán utilizados como valores iniciales ya capaces de controlar la planta para posteriormente ser ajustados de forma fina en esta segunda fase del algoritmo. Debido a que dicha metodología no se fundamenta de forma directa en la salida de la planta, se justificará la necesidad de un módulo supervisor que asegure que los cambios introducidos en esta fase repercutan directamente en el acercamiento de la variable a controlar al valor de consigna.

5.5.1 Adaptación mediante el error en la señal de control

Cuando nuestro controlador nos proporciona una señal de control en el instante *k*, *u(k)*, y evaluamos la salida *d* periodos de muestreo después, *y(k+d)*, el error cometido en la salida de la planta no es la única información que podemos obtener. Así, si medimos *y(k+d)* como salida, sabemos que, en las mismas condiciones en las que obtuvimos *u(k)*, si el valor deseado *r(k)* fuera *y(k+d)*, la salida óptima debería ser precisamente *u(k)*. Por tanto, tenemos un valor exacto de la verdadera función inversa de la planta. Esta es una información valiosa que no debe ser despreciada.

Para aprovechar dicha información, debemos evaluar cuál es la salida $\hat{u}$ que nos proporcionaría el controlador en el caso de que las condiciones fueran idénticas al instante *k* salvo que el valor deseado *r(k)* fuera el verdadero valor de la salida *y(k+d)* recién medido y compararla con la señal que ya sabemos que sería la óptima *u(k)*. Utilizando la notación anterior tendríamos:

$$u(k) = \hat{F}\left(\vec{x}(k);\Theta(k)\right) = F\left(\hat{x}(k)\right) \tag{5-32}$$

$$\hat{u}(k) = \hat{F}\left(\hat{x}(k); \Theta(k+d)\right) \tag{5-33}$$

donde $\Theta(k)$ simboliza el valor en el instante k de los parámetros que definen nuestro controlador (reglas + funciones de pertenencia) y

$$\hat{x}(k) = (y(k+d), y(k), \ldots, y(k-p), u(k-1), \ldots, u(k-q)) \tag{5-34}$$

es decir, la entrada con la que debemos evaluar nuestro controlador y cuya salida debería ser $u(k)$ ya que hemos reemplazado $r(k)$ por el valor que realmente ha salido en el instante $k+d$ manteniendo el resto de entradas como fueron en el instante k.

Debe notarse que para calcular $\hat{u}(k)$ debemos esperar al instante $k+d$, por lo que los parámetros del controlador difuso que debemos adaptar son los que haya en el instante $k+d$. Simplemente se trata de evaluar nuestro controlador en el instante $k+d$ para ver de qué forma es capaz de aproximar un punto que sabemos que pertenece a la función inversa real F y que teóricamente está situado cerca del estado actual de la planta.

Para ello, debemos introducir un segundo controlador auxiliar ca_2 equivalente al primero pero cuya entrada no será el error en la salida de la planta sino el producido en la señal de control

$$e_u(k+d) = u(k) - \hat{u}(k) \tag{5-35}$$

Dicho error deberá ser computado en el instante $k+d$ ya que hasta ese instante no sabremos cuál ha sido la salida de la planta $y(k+d)$.

Obviamente, para usar este método es necesario disponer de una segunda cola donde almacenaremos el nuevo vector de E/S perteneciente a la verdadera función inversa de la planta. Hay que notar que podría convenirnos el obligar a que justamente nuestro controlador diera la salida exacta para el punto recién descubierto de la función inversa. Esto, sin embargo, no es deseable ya que puede provocar una concentración de los parámetros para aproximar muy bien esos puntos en detrimento de otras regiones de operación.

Gracias a este tipo de información, podemos disponer de un método basado en el gradiente que vaya sintonizando los parámetros que definen a las funciones de

pertenencia en la dirección en la que se vaya disminuyendo el error en la salida del controlador. En cada instante *k* tendremos:

$$J(k) = \frac{1}{2} e_u^2(k) \tag{5-36}$$

$$\theta_i(k+1) = \theta_i(k) - \eta \frac{\partial J(k)}{\partial \theta_i} \tag{5-37}$$

donde η es el factor de aprendizaje. Conviene destacar que el algoritmo de descenso en gradiente en tiempo real no tiene las mismas propiedades que el que se ejecuta usando un conjunto fijo y finito de vectores de entrada/salida. En un descenso en gradiente en tiempo real, los parámetros se van desplazando en la dirección adecuada para disminuir **el error actual**, mientras que el usado en el capítulo 3 se desplazan en la dirección en la que minimizan el error global. Esta diferencia es lo que hace que un método converja de forma exacta y el otro no.

Derivando con respecto a θ_i en la ecuación (5-36) y usando (5-32), (5-33) y (5-35) obtenemos:

$$\frac{\partial J(k)}{\partial \theta_i} = -e_u(k) \frac{\partial \hat{F}\left(\hat{x}(k-d);\Theta(k)\right)}{\partial \theta_i} \tag{5-38}$$

de modo que la ecuación (5-37) quedaría:

$$\theta_i(k+1) = \theta_i(k) + \eta \cdot e_u(k) \frac{\partial \hat{F}\left(\hat{x}(k-d);\Theta(k)\right)}{\partial \theta_i} \tag{5-39}$$

Finalmente, debemos encontrar la expresión de la derivada parcial del último término de la ecuación anterior. Para ello, recordemos de la ecuación (5-5) que:

$$\hat{\mathrm{F}}\left(\hat{x}(k-d);\Theta(k)\right) = \frac{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(R_{i_1 i_2 \ldots i_N}(k)\cdot\prod_{m=1}^{N}\mu_{X_m^{i_m}(k)}\left(\hat{x}_m(k-d)\right)\right)}{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(\prod_{m=1}^{N}\mu_{X_m^{i_m}(k)}\left(\hat{x}_m(k-d)\right)\right)} \tag{5-40}$$

donde se ha incluido el índice *k* en cada regla y en cada función de pertenencia para recordar que estamos considerando la configuración de reglas y de funciones de pertenencia existente en el instante *k*, $R_{i_1,\ldots,i_N}(k)$, $X_m^{i_m}(k)$.

De este modo tendremos que:

- Si $\theta_i = R_{j_1,j_2,\ldots,j_N}$:

$$\frac{\partial \hat{F}\left(\hat{x}(k\text{-}d);\Theta(k)\right)}{\partial \theta_i} = \frac{\prod_{m=1}^{N} \mu_{X_m^{j_m}(k)}\left(\hat{x}_m(k-d)\right)}{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(\prod_{m=1}^{N} \mu_{X_m^{i_m}(k)}\left(\hat{x}_m(k-d)\right)\right)} \tag{5-41}$$

- Si θ_i es un parámetro de las funciones de pertenencia el cálculo es más complejo:

$$\begin{aligned}\frac{\partial \hat{F}\left(\hat{x}(k\text{-}d);\Theta(k)\right)}{\partial \theta_i} = &\frac{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(R_{i_1 i_2 \ldots i_N}(k)\cdot\frac{\partial\prod_{m=1}^{N}\mu_{X_m^{i_m}(k)}\left(\hat{x}_m(k-d)\right)}{\partial\theta_i}\right)}{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(\prod_{m=1}^{N}\mu_{X_m^{i_m}(k)}\left(\hat{x}_m(k-d)\right)\right)} - \\ &- \hat{F}\left(\hat{x}(k\text{-}d);\Theta(k)\right)\frac{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\frac{\partial\prod_{m=1}^{N}\mu_{X_m^{i_m}(k)}\left(\hat{x}_m(k-d)\right)}{\partial\theta_i}}{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(\prod_{m=1}^{N}\mu_{X_m^{i_m}(k)}\left(\hat{x}_m(k-d)\right)\right)}\end{aligned} \tag{5-42}$$

Finalmente, si el parámetro θ_i respecto al que se deriva es, por ejemplo, el centro de la *j*-ésima función de pertenencia de la variable *v* (c_v^j) tendremos que sólo las funciones de pertenencia de la variable *v* podrán depender de dicho parámetro, por lo que:

$$\frac{\partial\prod_{m=1}^{N}\mu_{X_m^{i_m}(k)}\left(\hat{x}_m(k-d)\right)}{\partial c_v^j} = \frac{\partial\mu_{X_v^{i_v}(k)}\left(\hat{x}_v(k-d)\right)}{\partial c_v^j}\cdot\prod_{\substack{m=1\\ m\neq v}}^{N}\mu_{X_m^{i_m}(k)}\left(\hat{x}_m(k-d)\right) \tag{5-43}$$

En el Apéndice A se encuentran los valores de estas últimas derivadas para las configuraciones que se usarán en este trabajo.

5.5.1.1 Factor de aprendizaje

Hasta ahora no hemos dicho nada sobre el factor de aprendizaje η que se usa en el descenso en gradiente en tiempo real (ecuación (5-37)). Dicho factor no ha de ser demasiado pequeño, lo que ralentizaría la velocidad de aprendizaje, ni demasiado grande para que no nos saltemos el valor al que se debería llegar.

Para analizar el valor que debería tener el factor de aprendizaje, llamaremos estado *a* al que corresponde antes de computar la ecuación (5-37) y estado *b* al posterior. El error que queremos minimizar en el instante *a* es $e_u(a) = \hat{u} - \hat{u}(a)$, siendo

$$\hat{u}(a) = \hat{F}\left(\hat{x}(a); \Theta(a)\right)$$

donde, para mantener la notación utilizada en este capítulo, se ha llamado[2]:

$$\hat{u} = \hat{F}\left(\vec{x}(k-d); \Theta(k-d)\right)$$

$$\hat{x}(a) \equiv \hat{x}(k-d)$$

$$\Theta(a) \equiv \Theta(k)$$

Cuando computemos un paso en la etapa de descenso en gradiente tendremos:

$$\theta_i(b) = \theta_i(a) - \eta \left.\frac{\partial J(k)}{\partial \theta_i}\right|_a$$

donde el índice *i* barre todos los parámetros de las funciones de pertenencia y las reglas que hay que optimizar. Conviene expresar la ecuación anterior en forma vectorial de la forma:

$$\vec{\Theta}(b) = \vec{\Theta}(a) - \eta \cdot \nabla_{\Theta} J(a) \qquad (5\text{-}44)$$

Lo que buscamos es que esa nueva configuración de parámetros disminuya el error producido para el caso en el que se repita el mismo estado *a*. Es decir, si llamamos:

$$\hat{u}(b) \equiv \hat{F}\left(\hat{x}(a); \Theta(b)\right)$$

$$e_u(b) = \hat{u} - \hat{u}(b)$$

el problema a tratar se expresaría de forma matemática como: *"Buscar un factor de aprendizaje η adecuado de tal forma que $|e_u(b)| \leq |e_u(a)|$, dándose la igualdad sólo en el caso de que ambos errores sean cero"*.

En primer lugar, conviene expresar el gradiente de *J* de forma vectorial

---

[2] Conviene recordar que û no es más que la salida que dio el controlador en el instante *k-d*, para obtener el valor deseado *r(k-d)* y que, evidentemente, es el valor ideal si dicha consigna fuera la salida que realmente se obtuvo posteriormente en el instante k.

$$\nabla_{\Theta} J(a) = -e_u(a) \cdot \nabla_{\Theta} \hat{u}(a) \tag{5-45}$$

de modo que la ecuación (5-44) ahora se reescribiría:

$$\vec{\Theta}(b) = \vec{\Theta}(a) + \eta \cdot e_u(a) \cdot \nabla_{\Theta} \hat{u}(a) \tag{5-46}$$

Supongamos que el factor de aprendizaje es lo suficientemente pequeño como para que se cumpla:

$$\left\| \vec{\Theta}(b) - \vec{\Theta}(a) \right\| << 1 \tag{5-47}$$

De esta forma, podremos realizar la siguiente aproximación lineal:

$$\hat{u}(b) \approx \hat{u}(a) + \left(\vec{\Theta}(b) - \vec{\Theta}(a) \right)^t \bullet \nabla_{\Theta} \hat{u}(a)$$

Restando la salida deseada $\hat{u}$ en cada término y multiplicando por (-1):

$$e_u(b) \approx e_u(a) - \left(\vec{\Theta}(b) - \vec{\Theta}(a) \right)^t \bullet \nabla_{\Theta} \hat{u}(a)$$

Reemplazando la expresión (5-46) en la anterior:

$$e_u(b) \approx e_u(a) - \left(\eta \cdot e_u(a) \cdot \nabla_{\Theta} \hat{u}(a) \right)^t \bullet \nabla_{\Theta} \hat{u}(a)$$

$$e_u(b) \approx e_u(a) \cdot \left(1 - \eta \left\| \nabla_{\Theta} \hat{u}(a) \right\|^2 \right) \tag{5-48}$$

por lo que si eligiéramos

$$\eta = \frac{1}{\left\| \nabla_{\Theta} \hat{u}(a) \right\|^2} \tag{5-49}$$

teóricamente obtendríamos $e_u(b) = 0$ en primera aproximación. Sin embargo, para valores pequeños de $\left\| \nabla_{\Theta} \hat{u}(a) \right\|^2$, el factor η sería grande y ya no se cumpliría la ecuación (5-47) que es la hipótesis inicial de donde hemos partido para obtener la expresión (5-49). Para evitar esto, escogeremos un factor de aprendizaje de la forma:

$$\eta = \frac{\eta_0}{1 + \left\| \nabla_{\Theta} \hat{u}(a) \right\|^2} \tag{5-50}$$

siendo η_0 el valor máximo permitido para η que debe ser ≤ 1. Substituyendo este nuevo valor en la ecuación (5-48) tendríamos:

$$e_u(b) \approx e_u(a) \cdot \left(1 - \eta_0 \frac{\left\| \nabla_{\Theta} \hat{u}(a) \right\|^2}{1 + \left\| \nabla_{\Theta} \hat{u}(a) \right\|^2} \right) \tag{5-51}$$

asegurando que $|e_u(b)| \leq |e_u(a)|$. Finalmente, comprobemos que nuestro factor de aprendizaje escogido cumple la condición (5-47) demostrándose así su validez.

$$\left\|\vec{\Theta}(b)-\vec{\Theta}(a)\right\| = \left\|\eta \cdot e_u(a) \cdot \nabla_\Theta \hat{u}(a)\right\| = \eta_0 \cdot \left|e_u(a)\right| \cdot \frac{\left\|\nabla_\Theta \hat{u}(a)\right\|}{1+\left\|\nabla_\Theta \hat{u}(a)\right\|^2} \tag{5-52}$$

La función $f(x) = |x| / (1 + x^2)$ presenta un máximo en $x = 1$ y el valor de dicho máximo vale ½, por lo que

$$\left\|\vec{\Theta}(b)-\vec{\Theta}(a)\right\| \leq \frac{1}{2}\eta_0 \cdot \left|e_u(a)\right| \tag{5-53}$$

Si escogemos $\eta_0 = máx(1{,}1/r_{actuador})$, siendo $r_{actuador}$ el rango del actuador, tendremos que $\frac{1}{2}\eta_0 \cdot \left|e_u(a)\right| << 1$, asegurándose, por tanto, la condición (5-47).

En resumen, en nuestra etapa de descenso en gradiente utilizaremos como factor de aprendizaje el siguiente:

$$\eta(a) = \frac{máx\left(1,\ 1/r_{actuador}\right)}{1+\left\|\nabla_\Theta \hat{u}(a)\right\|^2} \tag{5-54}$$

que deshaciendo los cambios se convierte en:

$$\eta(k) = \frac{máx\left(1,\ 1/r_{actuador}\right)}{1+\left\|\nabla_\Theta \hat{F}\left(\hat{x}(k-d);\Theta(k)\right)\right\|^2} \tag{5-55}$$

5.5.2 Sistema supervisor

Según el análisis de la sección anterior, sabemos que al modificar los parámetros del controlador principal mediante la expresión (5-39) utilizando el factor de aprendizaje definido en (5-55), tenemos asegurado que $|e_u(b)| \leq |e_u(a)|$. Sin embargo, lo que realmente interesa en el proceso de control es que el error en la variable de salida sea el que disminuya, es decir, que $|e(b)| \leq |e(a)|$.

Suponiendo la existencia de un conjunto de parámetros óptimo para la región de operación en la que nos encontremos es fácil demostrar que, en primera aproximación,

tal propiedad se cumple. Sin embargo, si los valores de los parámetros se encuentran relativamente distanciados de los valores óptimos, la condición de validez de una aproximación de la función óptima mediante su primera derivada ya no se verificaría y la convergencia del error de la salida de la planta no se podría asegurar.

Este razonamiento justifica la necesidad de un sistema supervisor en la segunda etapa del algoritmo que asegure que los cambios introducidos en esta fase repercutan directamente en el acercamiento de la variable a controlar al valor de consigna.

Dicho sistema supervisor debe evaluar los cambios producidos por el sistema auxiliar ca_2 basados en el error en la señal de control con el fin de comprobar que dichos cambios provoquen una disminución del error en la salida y actuar en el caso de que esto no sea así. La respuesta de cómo diseñar tal sistema la tenemos en la primera etapa. Precisamente el controlador auxiliar ca_1 tiene como finalidad primera la de reducir el error en la salida del proceso y, aunque su información no sea del todo exacta, sí que sabemos (por el desarrollo realizado en la sección 5.4.1.1) los límites que deben tener los cambios impuestos al controlador principal. De esta forma, dicho sistema evaluará los nuevos parámetros comprobando si verifican las condiciones impuestas por las ecuaciones (5-19) o (5-20), invirtiendo los cambios y proponiendo un cambio de los consecuentes de las reglas (que son los únicos parámetros que puede modificar) en caso negativo. En la sección 6.2.2.2 del capítulo siguiente se analizará el funcionamiento de este sistema supervisor con detalle.

En resumen:

- Si queremos controlar el sistema utilizando la información del error en la salida de la planta, sabemos cuál es el valor deseado pero no sabemos calcular las derivadas parciales, por lo que es necesario utilizar un método no basado en el gradiente (primera etapa).
- Durante el proceso de control en tiempo real podemos obtener, en cada iteración, un punto de la función inversa de la planta y ahora sí sabemos calcular las derivadas parciales. Aunque no sabemos cuál es el valor deseado para un $\vec{x}(k)$ dado sí lo sabemos para otro punto $\hat{x}(k)$ (segunda etapa).

- El controlador auxiliar ca_1 se encargará de ir buscando $e_y(k) = 0$, durante la primera etapa del algoritmo proporcionando, de paso, nuevos valores de la función inversa de la planta. Durante la segunda fase, su labor será la de supervisar que los cambios introducidos en esta fase repercutan directamente en el acercamiento de la variable a controlar al valor de consigna.

Finalmente, aunque este proceso no tiene por qué converger de forma exacta, sí que llegará un momento en el que los valores medios de los parámetros durante un periodo global de control completo se estanquen (esta es una de las condiciones que se imponen en la definición de T'). La condición de parada de la segunda etapa se cumplirá cuando no exista ningún parámetro cuya diferencia entre dos valores medios consecutivos sea mayor que un cierto tanto por ciento del rango de actuación local del parámetro.

5.6 Cambio del número de funciones de pertenencia en las variables de entrada

La modificación de la topología del controlador difuso en tiempo real es una cuestión que no ha sido nunca abordada en la bibliografía. Hasta la fecha, la práctica totalidad de los controladores adaptativos difusos propuestos en la bibliografía presentan siempre un número fijo de funciones de pertenencia. La razón de que esto sea así es comprensible ya que tampoco existen muchos trabajos que intenten acometer esta tarea para el caso teóricamente más simple de la modificación de la estructura a partir de una serie de datos de E/S en el campo de la aproximación funcional.

Para abordar este problema es necesario poseer información de todas las regiones de operación por las que ha circulado la planta ya que se trata de un análisis a nivel global de la política de control. Para ello, debido a las características de funcionamiento en tiempo real, dicha información debe ser recopilada conforme el proceso de control se esté llevando a cabo.

Una primera idea para realizar esto sería computar las variaciones medias de cada parámetro durante la etapa de sintonización fina a fin de tener una idea de las zonas con mayor necesidad de adición de nuevos parámetros. Sin embargo, dichas variaciones

medias tampoco son convergentes en sí debido al funcionamiento en tiempo real del algoritmo y en esta sección se ha preferido utilizar un método más exacto aunque con una mayor complejidad computacional.

Dicho método está basado en la teoría expuesta en el capítulo 3 y explota el hecho de que somos capaces de obtener datos de E/S de la verdadera función inversa de la planta en tiempo real de la manera que se ha comentado en las secciones anteriores. De esta forma, lo que haremos será construir una memoria adicional de tipo cola de tamaño *M* donde almacenaremos los últimos *M* datos de E/S pertenecientes a la función inversa de la planta para, posteriormente, aplicar el método de la sección 3.5 del capítulo 3. Debido a que este proceso es computacionalmente costoso, cuando se den las condiciones de "convergencia" de la etapa segunda, se proseguirá con dicha etapa congelando los datos existentes en ese momento en la memoria *M* y se ejecutará en paralelo el método presentado en dicha sección. De esta forma, tendremos resuelto automáticamente tanto el problema de selección de las variables que se deben utilizar como la localización de la nueva función de pertenencia. Sin embargo, ahora hay que tener en cuenta un nuevo problema que no existía antes: el tamaño de la memoria y el valor inicial de las nuevas reglas.

- El tamaño de *M* es una cuestión fundamental ahora, ya que necesitamos un conjunto suficientemente representativo de la función inversa de la planta. Para resolver esto, nos apoyaremos en la definición del periodo global de control. Según esta definición, T' es un periodo suficientemente grande como para poder evaluar la política de control durante su duración. De esta forma, los valores recopilados durante un periodo T' serán, por tanto, suficientemente significativos. Es por ello, por lo que seleccionaremos un tamaño proporcional a T', generalmente $M = T'/T$ donde T es el periodo real de muestreo, es decir, cada cuántos segundos vamos a controlar el sistema.

- En el capítulo 3, cada vez que nos encontrábamos con una nueva configuración o con valores nuevos de las funciones de pertenencia, no teníamos nada más que aplicar la expresión 3-18 para determinar los consecuentes óptimos. En este caso,

aplicar dicha expresión nos conducirá a valores de las reglas que globalmente aproximarán los datos que poseamos en la memoria *M*. Pero, debido al hecho de que el proceso se realiza en tiempo real, hay que tener en cuenta que los parámetros existentes en el momento de realizar el cambio estarán especializados en la región de operación actual de la planta, por lo que no nos interesa realizar dicho cambio global, sino mantener dicho nivel de especialización en dicha zona de forma que no se note externamente el cambio. Esta es la cuestión que se abordará en la siguiente sub-sección.

5.6.1 Valores iniciales de las nuevas reglas

Cuando añadimos una nueva función de pertenencia en la variable *v* se generan automáticamente $\prod_{\substack{i=1 \\ i\neq v}}^{N} n_i$ reglas nuevas, que son las que tienen la forma:

$$IF\ x_1 \text{ is } X_1^{i_1}\ AND \ldots AND\ x_v \text{ is } X_v^{j_v}\ AND \ldots AND\ x_N \text{ is } X_N^{i_N}\ THEN\ z = R_{i_1 i_2 \ldots i_v = j_v \ldots i_N} \quad (5\text{-}56)$$

siendo j_v la posición que ocupa la nueva función de pertenencia. Consecuentemente, el centro que antes ocupaba dicha posición pasa a llamarse $c_v^{j_v+1}$ y así sucesivamente con el resto de centros que haya a la derecha del nuevo $c_v^{j_v}$. Igual pasa con el resto de parámetros.

En este apartado denotaremos por $\tilde{\Theta}$ al conjunto antiguo de parámetros y por Θ al nuevo. De esta forma:

$$n_v = \tilde{n}_v + 1$$

$$n_i = \tilde{n}_i \quad \forall i \neq v$$

$$\theta_v^j = \tilde{\theta}_v^j \quad si\ j < j_v$$

$$\theta_v^j = \tilde{\theta}_v^{j-1} \quad si\ j > j_v$$

$$\theta_i^j = \tilde{\theta}_i^j \quad \forall i \neq v$$

Lo ideal sería introducir los valores nuevos de tal forma que la función global sea la misma que en el caso anterior (con la diferencia de que ahora en el proceso de adaptación tenemos más parámetros libres), es decir, que:

$$\hat{F}(\vec{x};\Theta) = \hat{F}(\vec{x};\tilde{\Theta}) \quad \forall \vec{x} \tag{5-57}$$

Al introducir la nueva función de pertenencia de la misma forma que se hacía en el capítulo 3, pero dejando ahora el resto de funciones intacto, tenemos que el grado de pertenencia de dicha función en los centros vecinos es despreciable[3] (y con más motivo en los puntos más allá de dichos centros vecinos) por lo que en el proceso de defuzzificación de todos los puntos cuya componente *v* caiga fuera del rango $[c_v^{j_v-1}, c_v^{j_v+1}]$ no participará ninguna de las reglas nuevas. De esta forma, en todos esos puntos se cumplirá la ecuación (5-57) y no se perderá ninguna información si escogemos las reglas nuevas de la forma:

$$\begin{aligned} R_{i_1,..,i_v,...,i_N} &= \tilde{R}_{i_1,..,i_v,...,i_N} \quad si\ i_v < j_v \\ R_{i_1,..,i_v,...,i_N} &= \tilde{R}_{i_1,..,i_v-1,...,i_N} \quad si\ i_v > j_v \end{aligned} \tag{5-58}$$

Para puntos cuya componente *v* caiga en el rango anterior lo único que podemos hacer es imponer la condición:

$$\hat{F}(\vec{c};\Theta) = \hat{F}(\vec{c};\tilde{\Theta}) \tag{5-59}$$

donde $\vec{c} = (c_1^{i_1},...,c_v^{i_v=j_v},...,c_N^{i_N})$, con $i_1=1...n_1$, ..., $i_N=1...n_N$. Es decir, justo en los puntos donde cada una de las nuevas reglas $R_{i_1,..,i_v=j_v,...,i_N}$ se activan con grado máximo. Para aclarar el proceso, veamos cómo se haría esto para el caso de *N=1* (funciones de una sola dimensión).

Supongamos que estamos usando funciones triangulares libres y que antes de añadir una nueva función de pertenencia la configuración es la que se representa en la figura 5.3a. Supongamos que usando el método del apartado anterior, se añade una nueva función de pertenencia en el punto $\bar{c} = 0.6$, viniendo dados los valores iniciales de los laterales

[3] nulo en el caso de funciones triangulares y *L* en el caso de gaussianas.

izquierdo y derecho por la posición de los centros vecinos (como si fuera una partición triangular) (ver figura 5.3b). En este caso:

$$n_1 = \tilde{n}_1 + 1 = 5$$

$$\left\{c_1^1, c_1^{1-}, c_1^{1+}\right\} = \left\{\tilde{c}_1^1, \tilde{c}_1^{1-}, \tilde{c}_1^{1+}\right\}$$

$$\left\{c_1^2, c_1^{2-}, c_1^{2+}\right\} = \left\{\tilde{c}_1^2, \tilde{c}_1^{2-}, \tilde{c}_1^{2+}\right\}$$

$$\left\{c_1^3, c_1^{3-}, c_1^{3+}\right\} = \left\{\bar{c}, \tilde{c}_1^2, \tilde{c}_1^3\right\}$$

$$\left\{c_1^4, c_1^{4-}, c_1^{4+}\right\} = \left\{\tilde{c}_1^3, \tilde{c}_1^{3-}, \tilde{c}_1^{3+}\right\}$$

$$\left\{c_1^5, c_1^{5-}, c_1^{5+}\right\} = \left\{\tilde{c}_1^4, \tilde{c}_1^{4-}, \tilde{c}_1^{4+}\right\}$$

$$\left\{R_1, R_2, R_4, R_5\right\} = \left\{\tilde{R}_1, \tilde{R}_2, \tilde{R}_3, \tilde{R}_4\right\}$$

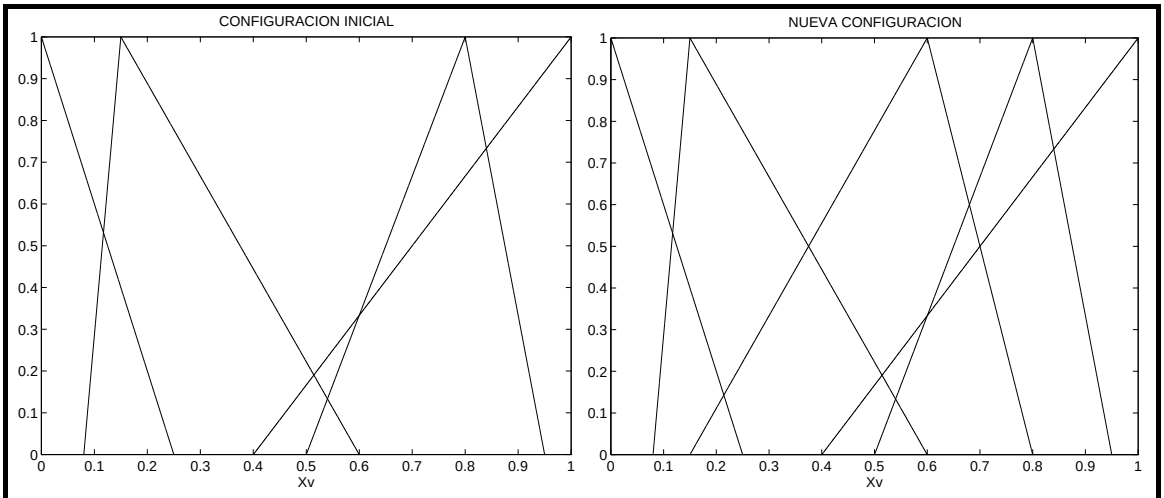


Fig. 5.3 Situación de las funciones de pertenencia.
(a) Antes de la etapa de inserción. (b) Tras dicha etapa.

y tendríamos que calcular el valor de la nueva regla R_3 para que se cumpliera que:

$$\hat{F}(\bar{c};\Theta) = \hat{F}(\bar{c};\tilde{\Theta}) \tag{5-60}$$

siendo

$$\hat{F}(\bar{c};\Theta) = \frac{R_1 \cdot \mu_{X_1^1}(\bar{c}) + R_2 \cdot \mu_{X_1^2}(\bar{c}) + R_3 \cdot \mu_{X_1^3}(\bar{c}) + R_4 \cdot \mu_{X_1^4}(\bar{c}) + R_5 \cdot \mu_{X_1^5}(\bar{c})}{\mu_{X_1^1}(\bar{c}) + \mu_{X_1^2}(\bar{c}) + \mu_{X_1^3}(\bar{c}) + \mu_{X_1^4}(\bar{c}) + \mu_{X_1^5}(\bar{c})} \tag{5-61}$$

$$\hat{F}(\bar{c};\tilde{\Theta}) = \frac{\tilde{R}_1 \cdot \mu_{\tilde{X}_1^1}(\bar{c}) + \tilde{R}_2 \cdot \mu_{\tilde{X}_1^2}(\bar{c}) + \tilde{R}_3 \cdot \mu_{\tilde{X}_1^3}(\bar{c}) + \tilde{R}_4 \cdot \mu_{\tilde{X}_1^4}(\bar{c})}{\mu_{\tilde{X}_1^1}(\bar{c}) + \mu_{\tilde{X}_1^2}(\bar{c}) + \mu_{\tilde{X}_1^3}(\bar{c}) + \mu_{\tilde{X}_1^4}(\bar{c})} \tag{5-62}$$

Reemplazando los valores antiguos por los nuevos en esta última ecuación obtenemos:

$$\hat{F}(\bar{c};\tilde{\Theta}) = \frac{R_1 \cdot \mu_{X_1^1}(\bar{c}) + R_2 \cdot \mu_{X_1^2}(\bar{c}) + R_4 \cdot \mu_{X_1^4}(\bar{c}) + R_5 \cdot \mu_{X_1^5}(\bar{c})}{\mu_{X_1^1}(\bar{c}) + \mu_{X_1^2}(\bar{c}) + \mu_{X_1^4}(\bar{c}) + \mu_{X_1^5}(\bar{c})} \tag{5-63}$$

Igualando (5-61) y (5-63), considerando que $\mu_{X_1^3}(\bar{c}) = 1$ y despejando obtenemos el valor de la regla que nos falta (R_3):

$$R_3 = \left\{ \frac{1 + \mu_{X_1^1}(\bar{c}) + \mu_{X_1^2}(\bar{c}) + \mu_{X_1^4}(\bar{c}) + \mu_{X_1^5}(\bar{c})}{\mu_{X_1^1}(\bar{c}) + \mu_{X_1^2}(\bar{c}) + \mu_{X_1^4}(\bar{c}) + \mu_{X_1^5}(\bar{c})} \left(R_1\mu_{X_1^1}(\bar{c}) + R_2\mu_{X_1^2}(\bar{c}) + R_4\mu_{X_1^4}(\bar{c}) + R_5\mu_{X_1^5}(\bar{c}) \right) - R_1\mu_{X_1^1}(\bar{c}) + R_2\mu_{X_1^2}(\bar{c}) + R_4\mu_{X_1^4}(\bar{c}) + R_5\mu_{X_1^5}(\bar{c}) \right.$$

de donde

$$R_3 = \frac{R_1 \cdot \mu_{X_1^1}(\bar{c}) + R_2 \cdot \mu_{X_1^2}(\bar{c}) + R_4 \cdot \mu_{X_1^4}(\bar{c}) + R_5 \cdot \mu_{X_1^5}(\bar{c})}{\mu_{X_1^1}(\bar{c}) + \mu_{X_1^2}(\bar{c}) + \mu_{X_1^4}(\bar{c}) + \mu_{X_1^5}(\bar{c})} = \hat{F}(\bar{c};\tilde{\Theta}) \tag{5-64}$$

es decir, el valor que hay que darle a la nueva regla coincide con el valor de la función antigua evaluada justo en el centro de la nueva función de pertenencia.

En el Apéndice C se presenta el desarrollo teórico para funciones de *N* dimensiones y se justifica que lo que hemos hecho para una dimensión también se puede hacer en el caso general. De esta forma, las nuevas reglas vendrán dadas por:

$$\mathrm{R}_{i_1,\ldots,i_v=j_v,\ldots,i_N} = \hat{F}(\vec{c};\tilde{\Theta}) \tag{5-65}$$

donde $\vec{c} = (c_1^{i_1},\ldots,c_v^{i_v=j_v},\ldots,c_N^{i_N})$, con $i_1 = 1,\ldots,n_1$, ..., $i_N = 1,\ldots,n_N$.

En el caso de haber equidistribuido todas las funciones de pertenencia de la variable *v*, se usará la misma expresión anterior pero moviendo también la variable i_v desde 1 hasta n_v barriéndose, por tanto, todas las reglas. Finalmente, hay que hacer notar que en el caso de que tras este proceso se conmute a la etapa 1 (y no a la 2), al utilizarse en dicha etapa sólo funciones de pertenencia definidas a través del valor de sus centros (TP ó PGP) se debe aplicar igualmente la expresión anterior a todas las reglas, una vez transformada la configuración actual en una TP ó una PGP.

5.7 Resumen y funcionamiento global del algoritmo

En la figura 5.2 se mostró un organigrama del algoritmo presentado en este capítulo donde se indican cada uno de los pasos analizados en las secciones anteriores.

El punto de partida será un controlador difuso principal relativamente simple, es decir, un sistema donde la mayoría de las variables de entrada tienen asignadas una sola función de pertenencia, lo que es equivalente a no seleccionarlas. El conjunto inicial de reglas será vacío (valores fijos o aleatorios) excepto en los casos donde el conocimiento del sistema permita dar un primer valor cuantitativo a las mismas.

Inmediatamente el controlador comenzará a operar en tiempo real. En la primera etapa de control (sección 5.4), el sistema auxiliar ca_1 se encargará de adaptar los consecuentes de las reglas evaluando el estado actual de la planta y el grado de responsabilidad de cada una de las reglas sobre dicho estado. Asimismo, de forma alternada en dicha etapa, se irán ajustando los valores centrales de las funciones de pertenencia con el fin de equidistribuir la contribución de cada zona a la integral del error cuadrático durante un periodo global de control T'.

Al cumplirse el criterio de convergencia de esta primera etapa, se habrán encontrado unos valores adecuados tanto de los consecuentes de las reglas como de los centros funciones de pertenencia capaces de controlar de forma razonable el sistema, según el número de parámetros con los que contemos.

Durante la segunda etapa del algoritmo (sección 5.5) explotamos el hecho que la planta ya está siendo controlada y que podemos disponer datos de E/S de la verdadera función inversa de la planta en tiempo real para poder realizar un proceso de minimización basado en el gradiente. De esta forma, el sistema auxiliar ca_2, al reducir el error en la salida del controlador, reduce indirectamente el error en la salida de la planta. Como teóricamente puede ocurrir que esto no sea así, se introduce además un sistema supervisor (que es el propio ca_1) que nos garantice que las modificaciones que se realicen vayan siempre dirigidas en la dirección que nos interesa: la que reduce el error en la salida de la planta.

Finalmente, durante esta segunda etapa, en lugar de desechar los datos de E/S que se utilizan en cada iteración, éstos se van recopilando en una memoria de tipo cola que será

utilizada al final del proceso para determinar un cambio adecuado en la topología del sistema con el fin de mejorar la política de control (sección 5.6).

Al funcionar el sistema en tiempo real, el algoritmo realmente no tiene final. En el caso de que los errores siempre sean menores que un cierto error máximo especificado, el algoritmo dejará de añadir funciones de pertenencia y permanecerá constantemente en la etapa 2.

Si comparamos el algoritmo de este capítulo y el del capítulo 3 podemos observar grandes analogías. Realmente es lógico que esto sea así ya que ambos esencialmente pretenden construir un sistema difuso, aunque uno de ellos utilizando un conjunto fijo de datos de E/S y otro trabajando directamente en tiempo real. El sistema auxiliar ca_1 hace el trabajo equivalente al de la expresión 3-18, es decir, buscar los consecuentes de las reglas. La etapa de igualación del criterio del error cuadrático es equivalente a la etapa previa de la sección 3.4.2, es decir, sólo modifica los centros de las funciones de pertenencia para encontrar un buen punto de partida para la etapa de descenso en gradiente. Por último, las etapas de minimización y selección de variables son también prácticamente equivalentes, con algunas diferencias ya indicadas anteriormente. A pesar de ello, la tarea que se intenta resolver en este capítulo es, evidentemente, mucho más compleja debido a su operación en tiempo real y, sobre todo, al hecho de que se debe realizar todo el proceso mientras se realiza el control de un sistema indeterminado inicialmente.

5.8 Mejoras generales del algoritmo

En este apartado se intentarán resolver una serie de cuestiones adicionales surgidas de la aplicación del algoritmo presentado en este capítulo y destinadas a mejorar y generalizar su funcionamiento.

5.8.1 Rango del actuador

En muchos casos prácticos, el rango del actuador no es todo lo holgado que se desearía para un cierto periodo de muestreo por lo que aparece un problema adicional:

Sea $u(k) = F(\vec{x}(k))$ la señal de control óptima en el instante *k*. Dicho control es el que precisamente consigue que *y(k+d) = r(k)*. Supongamos que nuestro actuador sólo admite un rango entre –5 y 5 voltios. Si en un cierto instante *u(k) = 7V*, sería imposible, aun en el caso óptimo, llevar la salida de la planta al valor deseado, *d* iteraciones más tarde. En ese caso, la salida de control óptima de nuestro controlador no sería $u(k) = F(\vec{x}(k))$ sino:

$$u(k) = sat(F(\vec{x}(k)); u_{min}, u_{max}) \tag{5-66}$$

Donde la función de saturación *sat(x;a,b)* está definida por:

$$sat(x;a,b) = \begin{cases} a & x < a \\ x & x \geq a \text{ y } x \leq b \\ b & x > b \end{cases} \tag{5-67}$$

Todo esto hay que tenerlo en cuenta en nuestro algoritmo de adaptación y auto-aprendizaje. Veamos cómo influye este hecho en cada una de las partes del mismo.

5.8.1.1 Efecto sobre la primera etapa

Recordemos que el controlador auxiliar ca_1 nos proporcionaba un cambio en los consecuentes de las reglas según el error actual en la salida de la planta partiendo del conocimiento de la monotonía (que supondremos siempre positiva sin pérdida de generalidad) y el retraso de la misma.

Supongamos que en el instante *k* la salida del controlador fue $\hat{F}(\vec{x}(k);\Theta(k)) > u_{max}$, la consigna era *r(k)* y que *d* instantes después tenemos la salida *y(k+d)*. La entrada real que recibe la planta será, debido al rango del actuador, $u(k) = u_{max}$. El controlador auxiliar ca_1 entonces tendrá como entrada *e(k+d) = r(k) - y(k+d)*. Supongamos que el controlador auxiliar para ese caso dé como salida una corrección de las reglas positivo, es decir, que los consecuentes deberían haber tenido valores mayores para que *u(k)* fuera mayor. Evidentemente, por mucho que se hiciera esta corrección, si se vuelve al mismo estado volveríamos a tener la misma salida u_{max} y el controlador auxiliar volvería a decir que se aumenten los consecuentes. Esta política de adaptación sería incorrecta y debemos, por tanto, modificar la estructura de ca_1 para tener en cuenta estos casos.

Para solucionar este problema, debemos evitar que se penalicen reglas por culpa del rango del actuador. En el caso anterior, la salida u_{max} sería la óptima ya que realmente, con la limitación del actuador, esa es la salida que daría la función inversa de la planta para este caso (ecuación (5-66)).

Así pues, la expresión (5-29) quedaría

$$\Delta R_i(k+d) = \alpha_i(k) \cdot \frac{F_{ca_1}\left(e(k+d),\dot{e}(k+d);u(k)\right)}{\left|\frac{\partial f}{\partial u}(\vec{x},u)\right|_{máx}} \cdot e(k+d) \qquad (5\text{-}68)$$

donde ahora el controlador auxiliar también tiene como entrada la señal de control responsable del estado actual de la planta. Realmente, dicha entrada sólo la usará en el caso de que $u(k) = u_{max}$ y la salida sea positiva, o bien, que $u(k) = u_{min}$ y la salida sea negativa. En cualquier otro caso, la salida dependerá exclusivamente del error en la salida de la planta. Igualmente, en la etapa de igualación de la integral del error cuadrático, en el caso de que el error actual se deba exclusivamente al rango del actuador, dicho error será considerado como cero y, evidentemente, no contribuirá como tal en su zona de acción.

Un detalle importante que hay que reseñar es que los consecuentes de las reglas no vienen limitados por el actuador sino que el filtro impuesto por el actuador se utiliza una vez obtenida la salida del controlador. Este detalle, aparentemente sin importancia, nos proporciona una gran ventaja como veremos a continuación.

Supongamos un ejemplo sencillo de una planta cuya función inversa óptima venga dada por la ecuación:

$$u(k) = F\left(\vec{x}(k)\right) = 10 \cdot e(k)$$

siendo $e(k) = r(k) - y(k)$, y que el retardo de la planta sea d=1. De esta forma, aplicando tal señal de control tendríamos que $y(k+1) = r(k)$. Supongamos que la región de operación donde vamos a actuar es el intervalo [-1,1] y que el rango del actuador es de –5 a 5V. En este caso la señal de control óptima sería la que se representa en tramos rectos en la figura 5.4. Para conseguir dicha función usando una partición triangular e

imponiendo la restricción de que las reglas no puedan tener valores por encima del rango del actuador necesitaríamos al menos 4 funciones de pertenencia en la variable del error, situadas en los puntos –1, -0.5, 0.5, 1 siendo los valores de las reglas en dichos puntos –5, -5, 5 y 5.

Sin embargo, quitando tal restricción podríamos obtener la misma función usando sólo dos funciones de pertenencia situadas en los puntos –1 y 1 con valores de las reglas –10 y 10, y dejando que sea el limitador del actuador el que actúe una vez obtenida la salida de control. De esta forma, nos ahorraríamos siempre dos funciones de pertenencia. Si esto lo pensamos para funciones de más variables de entrada el ahorro en funciones de pertenencia y en número de reglas es muy significativo ya que éste último depende exponencialmente del primero.

Por tanto, el algoritmo permitirá que las reglas evolucionen según lo que indique el controlador ca_1 (que ya tendrá en cuenta el rango del actuador) sin restringir su valor y dejar que sea el propio actuador el que limite la salida del controlador directamente. Dicha salida será, en cada instante *k*:

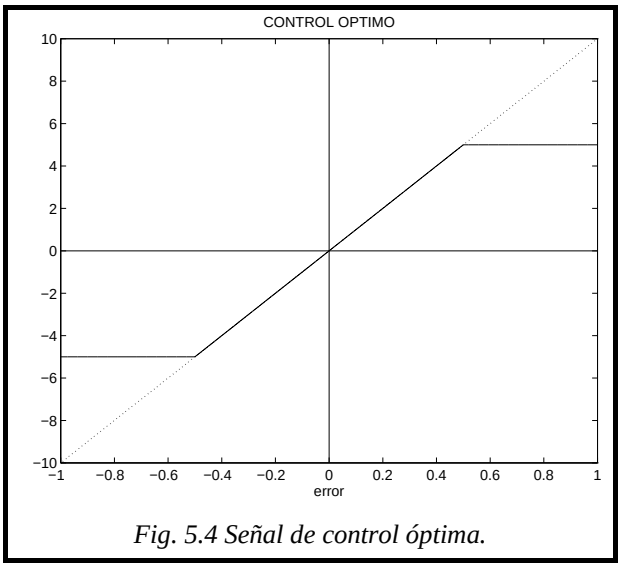


Fig. 5.4 Señal de control óptima.

$$u(k) = sat\left(\hat{F}\left(\vec{x}(k);\Theta(k)\right);u_{min},u_{max}\right) \tag{5-69}$$

5.8.1.2 Efecto sobre el controlador auxiliar ca_2

El controlador auxiliar ca_2 se basa en un punto de E/S de la verdadera función inversa de la planta. En cada instante *k*, el controlador proporciona la salida dada por la ecuación (5-69) y *d* instantes después se obtiene *y(k+d)*. Para el mismo estado que había en el instante *k*, ahora cambiamos *r(k)* por *y(k+d)* y evaluamos la salida que daría el controlador, obteniendo la señal de control $\hat{u}(k) = \hat{F}\left(\hat{x}(k);\Theta(k+d)\right)$ (que no se aplica nunca sobre la planta). De esta forma, aunque *u(k)* no fuera la salida deseada para el *r(k)* para el que se generó, sí sabemos que lo es para el caso de que *r(k)* fuera *y(k+d)* por lo que teníamos un error en la salida del controlador $e_u(k+d) = u(k) - \hat{u}(k)$.

Debido precisamente a la corrección que hemos hecho en la sección anterior, la salida del controlador que usamos para computar el error en la señal de control es justamente la dada por la ecuación (5-69), por lo que esta salida ya está limitada al rango del actuador. Si esa salida, que está dentro del rango, ha sido capaz de conseguir una salida de la planta *y(k+d)*, entonces quiere decir que si *û(k)* no es igual a *u(k)* puede y debe ser corregida sin restricciones ya que son valores reales ya limitados de la función inversa de la planta dada por la ecuación (5-66).

Por tanto, el efecto del rango del actuador no afecta a ca_2 siempre y cuando se utilice como señal deseada la señal de control real que sale por el actuador.

5.8.2 Especificación del error máximo permitido

En un problema de control real, generalmente no se pretende que el error en la salida sea exactamente cero sino que, según el problema de que se trate, se puede aceptar un cierto intervalo de error dentro del cual se considera que el sistema está controlado.

En el algoritmo expuesto en este capítulo, todos los elementos buscan siempre un error cero y se van añadiendo funciones de pertenencia hasta que se consiga un control perfecto. Sin embargo, hay aplicaciones para las cuales un cierto error en la salida, por ejemplo del 1% del rango de la misma, se considera más que suficiente y no se debe, por lo tanto, aumentar la complejidad del sistema para conseguir sobre-especificaciones que no se nos han exigido.

En la evolución del algoritmo, el controlador auxiliar ca_1 va a ir modificando las reglas hasta conseguir un error cero en la salida de la planta, e igualmente, el controlador ca_2 modificará tanto las reglas como las funciones de pertenencia hasta conseguir un error cero en la salida del controlador. Esto provoca que los parámetros se vayan especializando cada vez más en el estado actual del sistema en detrimento del resto de estados. Este fenómeno, que aparece en todos los algoritmos que trabajan en tiempo real, se conoce por el término “sobreajuste” (overfitting en inglés). Al especificar un intervalo de error permisible en la salida, los parámetros se van ajustando hasta

conseguir una salida dentro de dicho intervalo pero sin llegar más allá, lo cual disminuye el sobreajuste.

Por estos motivos, puede resultar útil incluir en el algoritmo el intervalo de error en la salida especificado por el usuario. Para ello, denotaremos por ε el error máximo permitido en la salida de la planta. El verdadero error que se comete ahora viene dado por:

$$e(k) = D\big(r(k-d) - y(k); \varepsilon\big) \tag{5-70}$$

donde $D(x;\varepsilon)$ es una función continua que crea una "zona muerta" en torno del punto $x = 0$ de anchura ε y viene expresada matemáticamente por:

$$D(x;\varepsilon) = \begin{cases} x+\varepsilon & si\ x < -\varepsilon \\ 0 & si\ |x| \le \varepsilon \\ x-\varepsilon & si\ x > \varepsilon \end{cases} \tag{5-71}$$

Ahora, si $e(k) = 0$, es decir, si $|r(k-d) - y(k)| \le \varepsilon$, el control se considera perfecto por lo que no se deberían modificar los parámetros que han originado dicho estado. Veamos cómo afecta esta característica al conjunto de elementos que intervienen en el algoritmo.

5.8.2.1 Efecto sobre la primera etapa

Para el controlador auxiliar ca_1, el efecto de la especificación de un error máximo permitido en la salida de la planta se puede incluir fácilmente sin más que utilizar como entrada el error de la planta dada por la ecuación (5-70). Con esta simple modificación, se puede utilizar un controlador auxiliar estándar independiente de dicha especificación y válida para todas las aplicaciones. Supongamos que F_{ca_1}, en función del error real de la planta tiene la forma de la figura 5.5a. Al reemplazar dicho error por el error dado por la ecuación (5-70) la salida equivalente tendría la forma de la figura 5.5b, donde se ha escogido un error máximo de ±0.05.

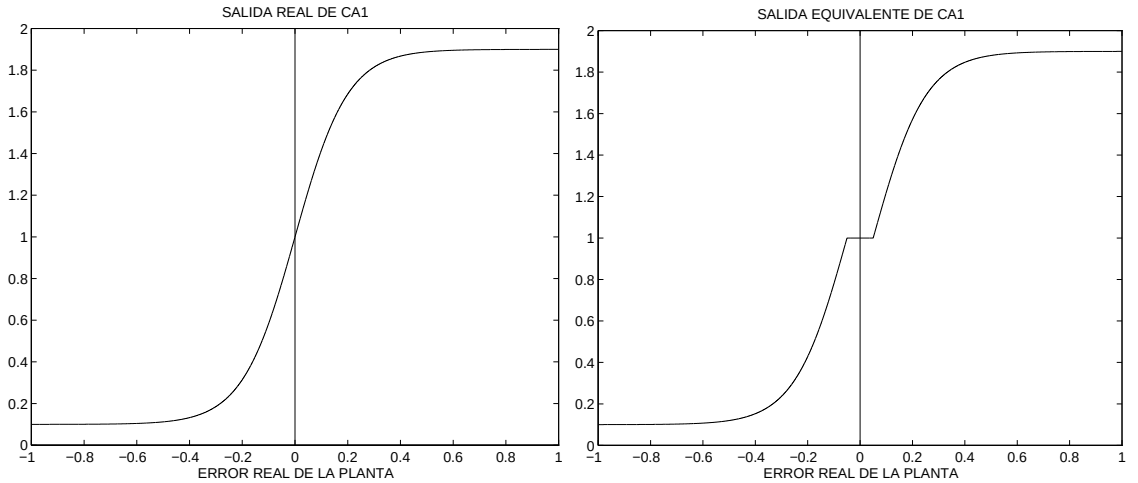


Fig. 5.5 (a) Salida real del controlador auxiliar ca_1. (b) Salida equivalente tras aplicar la zona muerta.

De esta forma, el utilizar la ecuación (5-70) como entrada para el controlador ca_1 origina que éste incluya la zona muerta sin tener que cambiar la forma de la función al utilizar otra especificación. Por supuesto, al considerar toda esa zona como error cero, la salida del controlador auxiliar debe ser cero, indicando que no debe producirse ninguna modificación en las reglas responsables en el estado actual de la planta.

Igualmente, para la etapa de igualación de la integral del error cuadrático, se utilizará como valor del error el dado por la expresión (5-70), por lo que errores dentro del límite permitido no contribuirán en su zona correspondiente.

5.8.2.2 Efecto sobre el controlador auxiliar ca_2

En este caso, el problema es un poco más complicado. La entrada del controlador auxiliar ca_2 es el error en la salida del controlador $e_u(k+d) = u(k) - \hat{u}(k)$ siendo

$$u(k) = F\left(\hat{x}(k)\right)$$

$$\hat{u}(k) = \hat{F}\left(\hat{x}(k);\Theta(k+d)\right)$$

$$\hat{x}(k) = (y(k+d),y(k),...,y(k-p),u(k-1),...,u(k-q))$$

Para ca_2, el valor deseado es $u(k)$, que consigue una salida deseada $y(k+d)$. Para analizar el problema habría que ver cuál sería la salida de la planta $\hat{y}(k+d)$ si se hubiera utilizado $\hat{u}(k)$ en lugar de $u(k)$ y comprobar si está dentro de los límites exigidos por la especificación del error máximo permitido en la salida.

La función de la planta (desconocida) tiene la forma (ecuación (5-3)):

$$y(k+d) = f(y(k),...,y(k-p),u(k),...,u(k-q))$$

Suponiendo que $\hat{u}(k)$ y $u(k)$ están relativamente próximos, lo cual será cierto ya que el algoritmo evoluciona para que esto sea así, podemos aproximar la salida producida por $\hat{u}(k)$ a partir de la producida por $u(k)$ linealizando la función de la planta en torno a la región de operación de la misma en el instante k:

$$\hat{y}(k+d) \approx y(k+d) + \left(\hat{u}(k) - u(k)\right) \cdot \left.\frac{\partial f}{\partial u}\right|_{y(k),y(k-1),...,u(k-1),...}$$

de donde

$$\hat{e}_y(k+d) \equiv y(k+d) - \hat{y}(k+d) = e_u(k+d) \cdot \left.\frac{\partial f}{\partial u}\right|_{y(k),y(k-1),...,u(k-1),...}$$

La condición para que $|\hat{e}_y(k+d)| < \varepsilon$ se traduce en que:

$$\left|e_u(k+d)\right| \leq \frac{\varepsilon}{\left|\partial f / \partial u\right|} \tag{5-72}$$

El problema es que no conocemos la función de la planta, por lo que no podemos computar directamente el valor de la derivada parcial de la ecuación (5-72). Para poder tener una estimación de dicha derivada utilizaremos el teorema de la función inversa que dice que:

> Si $y = f(x)$ es una función definida sobre un intervalo real, derivable y con derivada no nula en ningún punto de su dominio, la función inversa es diferenciable y cumple que:
>
> $$\left.\frac{\partial f^{-1}(y)}{\partial y}\right|_{y=y_0} = \frac{1}{\left.\frac{\partial f(x)}{\partial x}\right|_{x=x_0}}$$
>
> siendo $y_0 = f(x_0)$.

En nuestro caso, a la función inversa de la planta respecto a $u(k)$ la hemos denotado como (ver ecuación (5-4)):

$$u(k) = F\left(\vec{x}(k)\right)$$

siendo $\vec{x}(k)$ *=(r(k),y(k),...,y(k-p),u(k-1),...,u(k-q))* y *r(k) = y(k+d)* en este caso, por lo que $\vec{x}(k)$ es en realidad $\hat{x}(k)$. Por lo tanto, el teorema de la derivada de la función inversa quedaría:

$$\left.\frac{\partial F(\vec{x})}{\partial r}\right|_{\vec{x}=\hat{x}(k)} = \frac{1}{\left.\frac{\partial f}{\partial u}\right|_{y(k),y(k-1),...}} \tag{5-73}$$

derivada que existe ya que la planta es controlable y su derivada con respecto a *u(k)* nunca puede anularse por lo dicho en el apartado 5.2.

Finalmente, aunque tampoco conocemos la derivada parcial de la función inversa de la planta con respecto a la salida deseada, sí que tenemos una aproximación de dicha función inversa que es justamente la función que desempeña nuestro controlador. Como sí que conocemos la expresión matemática de la salida de nuestro controlador podemos evaluar dicha derivada parcial en cada instante *k*:

$$\left.\frac{\partial F(\vec{x})}{\partial r}\right|_{\vec{x}=\hat{x}(k)} \approx \left.\frac{\partial \hat{F}\left(\vec{x};\Theta(k+d)\right)}{\partial r}\right|_{\vec{x}=\hat{x}(k)} \tag{5-74}$$

De esta forma, la ecuación (5-72) quedaría de la siguiente manera:

$$\left|e_u(k+d)\right| \le \delta(k+d) \tag{5-75}$$

siendo

$$\delta(k+d) \equiv \varepsilon \cdot \left| \left.\frac{\partial \hat{F}\left(\vec{x};\Theta(k+d)\right)}{\partial r}\right|_{\vec{x}=\hat{x}(k)} \right| \tag{5-76}$$

Al igual que en la sección anterior, debido a la especificación de un error máximo permitido en la salida del controlador, el error verdadero en la salida del controlador tendría una zona muerta del tamaño dado por el parámetro δ de la ecuación anterior. Matemáticamente:

$$e_u(k+d) = D\left(u(k) - \hat{u}(k); \delta(k+d)\right) \tag{5-77}$$

Por lo que, al aplicar el descenso en gradiente, tendríamos:

$$\theta_i(k+1) = \theta_i(k) + \eta \cdot e_u(k) \frac{\partial e_u(k)}{\partial \theta_i}$$

donde ahora $e_u(k)$ viene dada por la ecuación (5-77) para contabilizar el efecto de la especificación del error máximo permitido en la salida de la planta a controlar. De esta forma, cuando $|u(k) - \hat{u}(k)|$ sea menor que $\delta(k)$, $e_u(k)$ será cero y no habrá actualización de parámetros.

Es importante reseñar un comentario sobre este proceso. Ahora $e_u(k)$ depende de $\delta(k)$ a través de la expresión (5-77) por lo que al computar la derivada también deberíamos tener en cuenta la derivada de $\delta(k)$ con respecto al parámetro θ_i. Si $e_u(k)$ es cero no será necesario computar la derivada ya que la corrección será siempre cero. En caso contrario, lo que tenemos que hacer es modificar los parámetros del controlador en la dirección en la que se disminuya el error entre $u(k)$ y $\hat{u}(k)$ y no en la dirección en la que aumente $\delta(k)$ para conseguir disminuir $e_u(k)$ a través de dicho parámetro. De esta forma, para el cómputo de la derivada consideraremos $\delta(k)$ como constante de modo que no interfiera en la dirección de cambio de los parámetros. Así, la ecuación (5-39) sigue siendo válida pero teniendo en cuenta que $e_u(k)$ viene dada por la ecuación (5-77).

5.8.3 Utilización de otros tipos de funciones de pertenencia

A lo largo de este capítulo, se ha tenido casi siempre en mente una partición triangular como forma de particionamiento del espacio de entrada. Sin embargo, al igual que se hizo en el capítulo 3, el algoritmo aquí presentado puede ser adaptado a otros tipos de funciones de pertenencia. En las secciones 3.8.2, 4.3 y en el Apéndice A, ya se presentó una serie de configuraciones de funciones de pertenencia distintas de la partición triangular y cómo modificar el algoritmo del capítulo 3 para poder ser utilizadas. En este apartado, por tanto, se abordará directamente el problema de cómo se puede adaptar el algoritmo presentado en este capítulo a las distintas configuraciones de funciones de pertenencia definidas en el Apéndice A, presuponiendo conocidas tales funciones.

1. Controlador auxiliar ca_1: el sistema auxiliar ca_1 modifica sólo los consecuentes de las reglas según el error en la salida de la planta. Como la política que se usa es la de la modificación de cada regla según su grado de responsabilidad, cuando se utilizan

otros tipos de funciones de pertenencia, los grados de activación pueden ser distintos pero el modo de razonamiento es exactamente el mismo, por lo que no es necesaria ninguna modificación en este sistema auxiliar.

2. Igualación de la integral del error cuadrático: En el caso de una partición triangular, el proceso de búsqueda modificaba las funciones de pertenencia completas ya que sólo teníamos como parámetros libres los propios centros de las funciones y era fácil determinar en qué dirección mover cada uno de ellos para intentar homogeneizar la distribución de errores cuadráticos. En el caso de funciones de pertenencia más complicadas disponemos de más grados de libertad por lo que debemos imponer ciertas restricciones al resto de parámetros de la misma forma que hacíamos en la etapa previa en el algoritmo del capítulo 3. Así pues, a lo largo de esta etapa, las funciones triangulares libres se considerarán como una partición triangular y las funciones gaussianas operarán como una partición pseudo-gaussiana fijando el valor del parámetro L (ver sección 4.3).

3. Descenso en gradiente: El proceso de descenso en gradiente no entraña ninguna dificultad ya que ahora sí conocemos el valor de la parcial del error con respecto a cualquiera de los parámetros del controlador principal. Es en esta etapa donde realmente utilizamos todos los parámetros ajustables sin restricción.

4. Adición de funciones de pertenencia: Al utilizar la misma teoría que en el capítulo 3, todo lo expuesto en dicho capítulo para otras funciones de pertenencia sigue siendo válido aquí. La única diferencia es el cálculo de los nuevos consecuentes de las reglas. Por ello, el Apéndice C ya se escribió teniendo en cuenta otros tipos de funciones de pertenencia.

En el capítulo 6 se mostrarán ejemplos de controladores definidos por cada una de las configuraciones que aparecen en el Apéndice A.

5.9 Conclusión

En este capítulo se ha propuesto una nueva y completa metodología para el diseño automático de controladores difusos directos en tiempo real. Prácticamente desde una estructura vacía y a través del análisis de la evolución del sistema de control mediante dos sistemas difusos auxiliares, el algoritmo va construyendo y optimizando el controlador difuso principal. Es decir, se ha abordado tanto el problema de la identificación de la estructura del sistema como el ajuste de sus parámetros en tiempo real.

La metodología consta esencialmente de tres etapas: En la primera, se adaptan los consecuentes de las reglas utilizando información directa del error en la salida de la planta. Mientras, de forma alternada, se modifican los centros de las funciones de pertenencia intentando igualar la contribución de cada zona a la integral del error cuadrático durante un periodo global de control T'. Dicho periodo debe ser lo suficientemente alto como para considerar que se puede evaluar la política de control de forma global durante dicho periodo.

Posteriormente se realiza un ajuste fino tanto de los parámetros que definen las funciones de pertenencia en la entrada como de los consecuentes de las reglas generadas, a través de un método basado en el gradiente. Debido a que dicho método se basa en el error en la salida del controlador y no de la planta, se ha justificado la necesidad de un sistema supervisor que asegure que el proceso evolucione de forma que siempre se aproxime el valor de salida de la planta al valor de consigna.

Una vez optimizada la configuración existente, se realiza un análisis global de la política de control utilizando una memoria de tipo cola que contendrá los últimos M datos recopilados de la verdadera función inversa de la planta. Dicho análisis nos permitirá identificar, de la misma forma que se hizo en el capítulo 3, las variables donde se ha de aumentar el número de funciones de pertenencia que particionan sus dominios de modo que se mejore la política de control. De esta forma, el algoritmo es capaz de hallar cuántas funciones de pertenencia se deben utilizar para cada variable de entrada y, como caso particular, cuáles de las entradas proporcionadas son realmente necesarias

para el cumplimiento de las especificaciones del problema (principalmente el error máximo permitido en la salida).

En la última parte del capítulo se presenta una serie de mejoras del algoritmo para tener en cuenta un rango limitado del actuador, para utilizar la especificación del error máximo permitido con objeto de evitar el sobre-ajuste y para adaptar el algoritmo propuesto a otros tipos de funciones de pertenencia distintos de una partición triangular.

En el capítulo siguiente se muestran ejemplos de aplicación y comparaciones con otras metodologías utilizadas en la bibliografía.

CAPÍTULO 6

EJEMPLOS, EVALUACIÓN Y COMPARACIONES DEL ALGORITMO PARA CONTROL EN TIEMPO REAL

En este capítulo se presentarán ejemplos que demuestran la validez del procedimiento para la adaptación y auto-aprendizaje de controladores difusos operando en tiempo real, presentado en el capítulo anterior.

Tras un breve resumen del método presentado (sección 6.1) se abordará en la sección 6.1.1 una serie de cuestiones preliminares acerca de las condiciones generales que se utilizarán en las simulaciones presentadas a lo largo del capítulo. En la sección 6.2 se utilizarán ejemplos, con distintas complejidades, con objeto de poner de relieve las principales características del algoritmo (primera etapa, sistema auxiliar ca_2, sistema supervisor y modificación de la topología del controlador). En la sección 6.3 se analizarán otros factores como la especificación de un error máximo permitido, el comportamiento frente a plantas variables con el tiempo, y el uso de otros tipos de funciones de pertenencia. Finalmente, en la sección 6.4 se comparan los resultados obtenidos con los proporcionados mediante otros algoritmos también destinados al campo de control en tiempo real, y se presentan las conclusiones en el apartado 6.5.

6.1 Introducción

En el capítulo anterior se ha presentado una nueva metodología para la adaptación y auto-aprendizaje de controladores difusos operando completamente en tiempo real sin ningún entrenamiento previo. A modo de resumen, dicha metodología constaba de una serie de pasos:

El punto de partida es un controlador difuso principal relativamente simple, es decir, un sistema donde la mayoría de las variables de entrada tienen asignadas una sola función de pertenencia, lo que es equivalente a no seleccionarlas. En este capítulo supondremos que no existe conocimiento cualitativo previo de los valores iniciales de las reglas por lo que el conjunto inicial de éstas será vacío (valores fijos a 0). El único conocimiento previo requerido es el signo de la monotonía de la planta, el retraso que presenta su salida con respecto a la señal de control y una estimación de qué entradas pueden ser las que realmente influyen en el proceso.

Etapa 1:
El controlador comenzará inmediatamente a operar en tiempo real. En la primera etapa de control (sección 5.4 del capítulo anterior) el sistema auxiliar ca_1 se encargará de adaptar los consecuentes de las reglas evaluando el estado actual de la planta y el grado de responsabilidad de cada una de las reglas sobre dicho estado. Asimismo, de forma alternada en dicha etapa, se irá ajustando los valores centrales de las funciones de pertenencia con el fin de equidistribuir la contribución de cada zona a la integral del error cuadrático (IEC) durante un periodo global de control T'.

Al cumplirse el criterio de convergencia de esta primera etapa (en este caso que los parámetros involucrados no se muevan más allá de un determinado tanto por ciento de su rango local de actuación), se habrán encontrado unos valores iniciales tanto de los consecuentes de las reglas como de los centros de las funciones de pertenencia, capaces de controlar de forma razonable el sistema acorde con el número de parámetros con los que cuenta el controlador principal.

Etapa 2:

Durante la segunda etapa del algoritmo (sección 5.5) explotamos el hecho de que podemos disponer datos de E/S de la verdadera función inversa de la planta en tiempo real para poder realizar un proceso de minimización basado en el gradiente. De esta forma, el sistema auxiliar ca_2, al reducir el error en la salida del controlador, reduce indirectamente el error en la salida de la planta. Como teóricamente puede ocurrir que esto no sea así, se introduce además un sistema supervisor (que es el propio ca_1) que nos garantice que las modificaciones que se realicen vayan siempre dirigidas en la dirección que realmente interesa: la que reduce el error en la salida de la planta.

Adicionalmente, durante esta segunda etapa, en lugar de desechar los datos de E/S que se utilizan en cada iteración, éstos se van recopilando en una memoria de tipo cola que será utilizada al final del proceso para determinar un cambio adecuado en la topología del sistema con el fin de mejorar la política de control (sección 5.6).

Finalmente, en el capítulo anterior, se introducían ciertas mejoras en el algoritmo. En concreto, cómo tener en cuenta un rango limitado real del actuador, cómo utilizar la especificación del error máximo permitido para evitar el sobre-ajuste y cómo adaptar el algoritmo propuesto a otros tipos de funciones de pertenencia.

Al funcionar el sistema en tiempo real, el algoritmo realmente no tiene final. En el caso de que los errores siempre sean menores que un cierto error máximo especificado, el algoritmo dejará de añadir funciones de pertenencia y permanecerá constantemente en la etapa 2.

A lo largo de este capítulo se analizarán cada una de estas etapas presentando ejemplos que pongan de relieve las características principales del algoritmo y sus mejoras. Finalmente se compararán los resultados con los obtenidos por otras metodologías existentes en la bibliografía destinadas al control en tiempo real.

6.1.1 Condiciones generales de las simulaciones

Con el fin de no tener que repetir en cada una de las simulaciones que aparecen a lo largo de este capítulo el valor de ciertos parámetros del algoritmo que no son los que se

analizan expresamente, en esta sección se indican una serie de valores y condiciones generales que se emplearán por defecto en todas las simulaciones salvo que se indique explícitamente lo contrario en el apartado correspondiente:

Tipos de funciones de pertenencia: A lo largo de todo el capítulo salvo en la sección 6.3.3 se utilizará siempre una partición triangular (TP) como modelo de funciones de pertenencia.

Valores iniciales de las reglas: En todas las simulaciones de este capítulo se comenzará con un conjunto de reglas con un valor inicial de 0.0, es decir, no se supondrá conocimiento alguno previo sobre dichos valores.

Consignas: Como consignas a seguir por la salida de la planta, en las simulaciones mostradas se ha procurado utilizar una gran variedad de patrones tanto en lo que se refiere a la forma del patrón como al rango de definición. A lo largo de este capítulo se han empleado patrones de señales aleatorias, funciones escalón, patrones combinados de escalones, senos y dientes de sierra, funciones senoidales con frecuencias aleatorias, etc. utilizando rangos diversos.

Parámetros del algoritmo: Como se comentó en el capítulo anterior, la constante C que se utiliza en el sistema auxiliar ca_1 necesita teóricamente ser fijada previamente antes de comenzar el algoritmo a través de una estimación de cual puede ser la variación máxima de la salida de la planta con respecto a la señal de control. Sin embargo, como la intención de este capítulo es mostrar el funcionamiento del algoritmo con el mínimo conocimiento previo posible sobre la planta, se utilizará en todos los ejemplos (salvo en las secciones donde se analiza este parámetro) un valor fijo de $C=0.1$, es decir, una pendiente máxima de *10*, que es más que suficiente para todos los ejemplos que se muestran. Recordemos que el papel del sistema auxiliar ca_1 es ajustar el valor de los consecuentes de las reglas a unos valores iniciales adecuados y que no se requiere que dicha evolución sea lo más óptima posible. Es en la segunda etapa donde se realiza un ajuste fino sin la necesidad de parámetros arbitrarios.

La temperatura inicial para los centros durante la etapa de igualación del criterio de la integral del error cuadrático por zonas será de 100 aunque este valor inicial no es crítico ya que se va adaptando también conforme evoluciona dicha etapa. El radio de acción *b* se fijará al valor 5 indicando que, como mucho, nos desplazaremos hasta una quinta parte de la distancia que nos separa del centro vecino aunque realmente la función primordial de este parámetro es simplemente la de asegurar que los centros no varíen de orden.

Finalmente, el último parámetro que aparece el algoritmo es el periodo global de control T'. En el caso de consignas pertenecientes a la repetición de un patrón dado, se escogerá como T' el periodo de tal patrón y en el caso de consignas aleatorias se utilizará por defecto un periodo global de control de 100 iteraciones. Como se verá en la sección 6.2.1.2 este factor no es especialmente importante (siempre que sea lo suficientemente alto) en la etapa de igualación del criterio de la integral del error. Además, cuando se utiliza este parámetro para conmutar entre etapas, realmente lo que se utiliza es la evolución de la modificación de los parámetros y ésta es independiente del valor de T' aunque se evalúe en los instantes indicados por dicho valor.

Criterios de convergencia: Como se especificó anteriormente, la condición de convergencia tanto de la etapa 1 como de la etapa 2 será que el valor medio de cada uno de los parámetros que se ajustan no exceda en ningún caso del 1% sobre el rango local medio asignado a cada parámetro. Es decir, en el caso de los centros de las funciones de pertenencia de una variable, el límite será el 1% del rango de la variable en cuestión dividido por el número de funciones de pertenencia de dicha variable. En el caso de funciones gaussianas, el valor de la desviación se traducirá en valores equivalentes respecto al rango de la variable. En el caso de las reglas se utilizará el 1% del rango real del actuador (que se actualiza cada T' iteraciones) dividido por el número total de reglas.

Error máximo permitido: No se tendrá en cuenta, es decir, se considerará 0, salvo en las secciones 6.3.1 y 6.3.3 donde se analizará la forma de utilizar este parámetro a favor de las prestaciones del algoritmo.

Rango del actuador: En todos los ejemplos mostrados, excepto en algunos de la sección 6.2.1.1 donde se estudia esta característica, se ha considerado un rango del actuador ilimitado, es decir, el sistema se adaptará libremente en cuanto al rango del actuador se refiere. También conviene indicar que en aquellas expresiones donde se utiliza el rango del actuador (r_{act}) se refieren al rango real del actuador durante el proceso real de control, que es una variable que se va actualizando cada T' iteraciones conforme evoluciona el proceso.

Ecuaciones de la planta: Las plantas que se simularán en este capítulo pueden venir expresadas a través de sus ecuaciones diferenciales o de sus ecuaciones en diferencias. En el primer caso, conviene distinguir entre el periodo de muestreo utilizado para simular las ecuaciones diferenciales en el computador y el periodo real que se utiliza para obtener las señales de control. En todas las simulaciones que se han realizado sobre plantas expresadas a través de sus ecuaciones diferenciales se ha utilizado un periodo para su simulación de 0.001 segundos utilizando el método de Runge-Kutta de 4º orden. Los periodos de muestreo del controlador (al que llamamos T), sin embargo, oscilarán a lo largo de este capítulo entre 0.1s (en el caso del péndulo invertido) y 5s (en el caso del tanque de líquido).

6.2 Principales características del algoritmo

En esta sección se utilizarán ejemplos, con distintas complejidades, cuyo fin es poner de relieve las principales características del algoritmo: Adaptación mediante el error en la salida de la planta, adaptación mediante el error en la salida del controlador y la modificación de la topología del controlador.

6.2.1 Adaptación mediante el error en la salida de la planta

La primera etapa del algoritmo se basa en la adaptación tanto de los consecuentes de las reglas difusas como de los valores centrales de las funciones de pertenencia basándose en el error en la salida de la planta. Recordemos que en dicha etapa la adaptación de las reglas se llevaba a cabo a través del sistema auxiliar ca_1 mientras que la modificación de

los centros se realizaba de forma alternada intentado igualar la integral del error cuadrático entre cada una de las zonas definidas por las funciones de pertenencia.

6.2.1.1 Sistema auxiliar ca_1

El sistema auxiliar ca_1 es el encargado de modificar los consecuentes de las reglas del controlador utilizando la información de la monotonía de la planta y el retraso de la misma. Dicha modificación esencialmente es de la forma:

$$\Delta R_i(k) = \alpha_i(k\text{-}d)\cdot C\cdot e(k) \tag{6-1}$$

donde $\alpha_i(k\text{-}d)$ es el grado de activación de la regla i-ésima al inferir la señal de control $u(k\text{-}d)$, k es la iteración actual, d el retraso de la planta y $e(k)$ es el error actual en la salida de la planta. C es un parámetro que teóricamente viene dado por:

$$|C| = \frac{1}{\left|\frac{\partial f}{\partial u}(\vec{x},u)\right|_{máx}} \tag{6-2}$$

donde f es la función de la planta. Lo primero que haremos será analizar la importancia de dicho parámetro en la evolución del sistema. Consideremos para ello una planta muy sencilla dada por la siguiente ecuación diferencial:

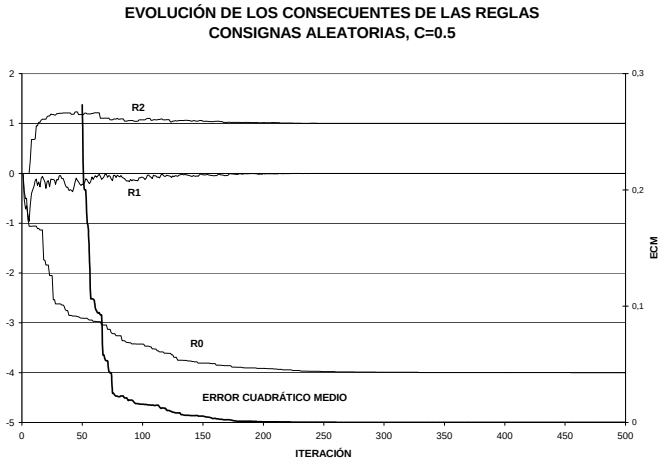
$$\Pi_2 \equiv \frac{dy(t)}{dt} = \phi(u(t)) \tag{6-3}$$

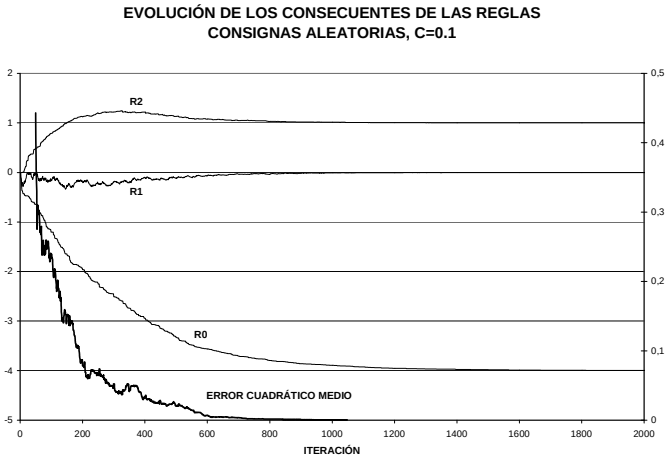
siendo

$$\phi(x) = \begin{cases} 2x & \text{si } x > 0 \\ x/2 & \text{si } x < 0 \end{cases} \tag{6-4}$$

Lo primero que observamos es que la planta presenta monotonía ascendente ya que la salida siempre crecerá al crecer la señal de control. Además, la planta no presenta ningún tipo de retraso por lo que podremos escoger $d=1$ para cualquier periodo de muestreo. Consideremos un controlador con una sola variable de entrada, que será el error en la salida de la planta $e(k) = r(k) - y(k)$, siendo $r(k)$ el valor de consigna, que se hará variar de forma aleatoria en el rango [-1,1]. El valor teóricamente adecuado del parámetro C en este caso sería $C=0.5$ ya que el valor máximo de la parcial de $y(t)$ con respecto a $u(t)$ es 2. Consideremos un periodo de muestreo de 0.1 segundos (aunque es irrelevante para este tipo de plantas) y un controlador inicial con 3 funciones de pertenencia en la variable del error situadas en los puntos -2, 0 y 2. Las reglas, como

siempre, se inicializarán al valor 0. En la figura 6.1a se muestra la evolución de los consecuentes de las reglas para este caso así como la del error cuadrático medio medido en ventanas de 50 iteraciones.





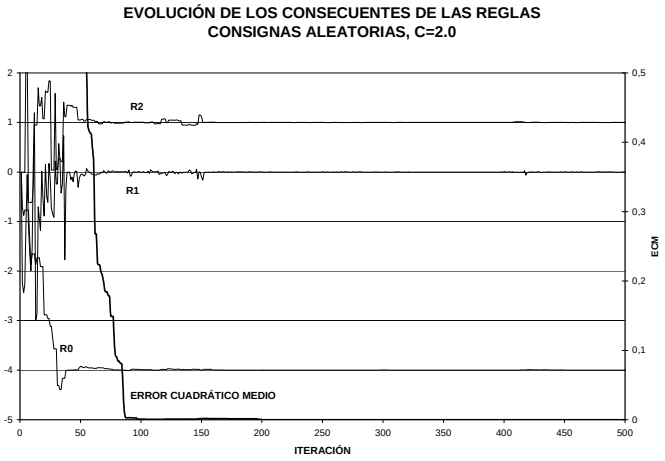


Fig. 6.1 Evolución del error cuadrático medio y de los consecuentes de las reglas para a) C=0.5, b) C=0.1, c) C=2.0.

En dicha figura se observa cómo las reglas convergen rápidamente a sus valores óptimos (que en el caso de esta planta tan sencilla existen para las funciones de pertenencia utilizadas y vienen dados por $R_0=-4$, $R_1=0$, $R_2=1$) y el error cuadrático medio decae hasta el valor cero en menos de 500 iteraciones. En las figuras 6.1b y 6.1c se muestra la misma evolución para el caso de $C=0.1$ y $C=2$. En el primer caso la evolución sigue siendo correcta aunque bastante más lenta mientras que en el segundo los valores fluctúan demasiado violentamente (salvo para el caso de R_0 para el que el valor de $C=2$ es realmente adecuado ya que se trata de la parte donde $u(k)$ es negativa) e incluso, aunque aparentemente parezca que el sistema ha llegado a la convergencia, esto no es así ya que todavía en la iteración 400 se observan movimientos en las reglas. Para valores mucho más grandes de C el sistema deja de converger.

Una nota importante a reseñar en este momento es que sólo es en las primeras iteraciones del algoritmo donde el factor C desempeña un papel predominante ya que se trata de un periodo en el que las reglas no están aprendidas y deben realizar grandes desplazamientos. Una vez llegados a ese punto, los movimientos de las reglas serán mucho menores para el resto de la evolución del algoritmo por lo que un C bastante menor del teórico sigue siendo tan bueno como éste. Sin embargo, escoger un C demasiado grande sí que trae complicaciones durante todo el proceso ya que las reglas van a ir adaptándose con movimientos excesivos deteriorando las prestaciones del proceso de control e incluso imposibilitando la convergencia, como ya se demostró teóricamente en el capítulo anterior. Asimismo, hay que tener cuidado con la interpretación de los resultados obtenidos en tal desarrollo teórico. El valor calculado para C es el que asegura que los nuevos valores de las reglas van a mejorar el error en la salida del controlador en el caso de que se dieran las mismas condiciones en las que se hizo la inferencia difusa. Durante un proceso real de control, el estado del sistema va dando "saltos" entre las distintas regiones de operación por lo que las modificaciones que se realizan sólo deben llevarse a cabo en aquellas reglas responsables de la región de operación que se evalúa. Es por ello que en la expresión (6-1) se pondera por el nivel de activación de las reglas.

Para poner de manifiesto este hecho, consideremos el ejemplo anterior, con el mejor C posible ($C=0.5$) pero realizando la adaptación sin ponderar por los grados de activación de las reglas:

$$\Delta R_i(k+d) = C \cdot e(k+d) \; si \; \alpha_i(k)>0 \qquad (6\text{-}5)$$

Teóricamente, esta forma de adaptación (que usan numerosos autores como se indicó en el capítulo anterior) también se adecua a las condiciones de convergencia establecidas en el capítulo anterior. Sin embargo, aunque cada modificación es teóricamente correcta de forma individual, con esta política de adaptación estamos modificando apreciablemente también reglas que apenas influyen en la obtención de un $u(k)$ dado, por lo que al conmutar el estado de la planta entre diversas regiones de operación las reglas responsables ahora tienen otros valores debido a errores pasados en los que escasamente influyeron. Este fenómeno se puede observar en la figura 6.2 donde se aprecian las grandes oscilaciones de las reglas sin que se llegue a producir nunca la convergencia.

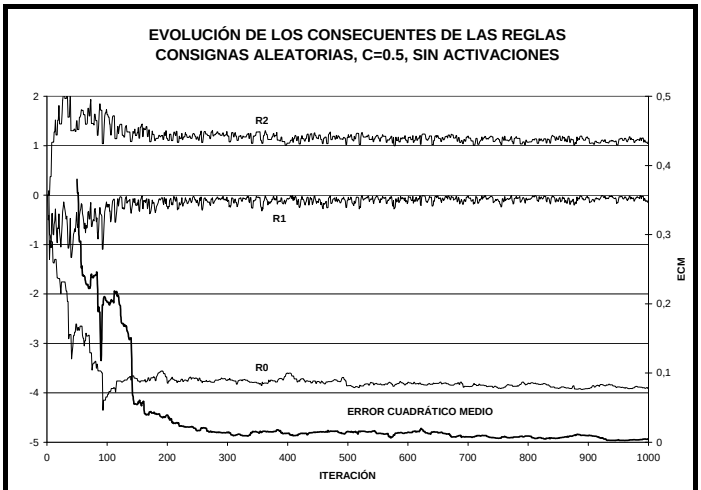


Fig. 6.2 Evolución sin ponderar el cambio de las reglas por el grado de activación de las mismas.

Rango del actuador: En la sección 5.8.1 del capítulo anterior se determinaron las modificaciones que había que incluir en el funcionamiento del sistema supervisor ca_1 para poder realizar la adaptación de los consecuentes de las reglas cuando el rango del actuador está limitado. Esencialmente, lo único que había que analizar es si la variable de control que se utilizó en el instante k-d estaba justo en el límite del rango y si la modificación impuesta por el sistema adaptativo iba en la dirección de desplazar el consecuente de la regla más allá del rango. En ese caso, como es lógico, las reglas no se modificaban. En la figura 6.3 se representa la misma planta que en el ejemplo anterior con idénticas condiciones salvo que ahora el rango del actuador está limitado en el intervalo [-1,1]. En el segundo eje de ordenadas se representa el error cuadrático medio

cada 50 iteraciones tanto teniendo en cuenta el error completo real como el que se calcula eliminando aquellos errores que se deben exclusivamente a la limitación del rango del actuador (de lo cual, el controlador no es responsable). En la figura se aprecia como el error del que sí es responsable el controlador decrece con el número de iteraciones y se va haciendo cero aunque el error cuadrático medio total se mantiene en valores altos con muchas fluctuaciones debido al carácter aleatorio de las consignas. En este caso la convergencia es más lenta debido a que ahora son muchas menos las iteraciones que realmente se utilizan para adaptar las reglas. En el caso de no utilizar la modificación mencionada en el algoritmo de adaptación las reglas irían creciendo indefinidamente y el sistema se volvería completamente inestable.

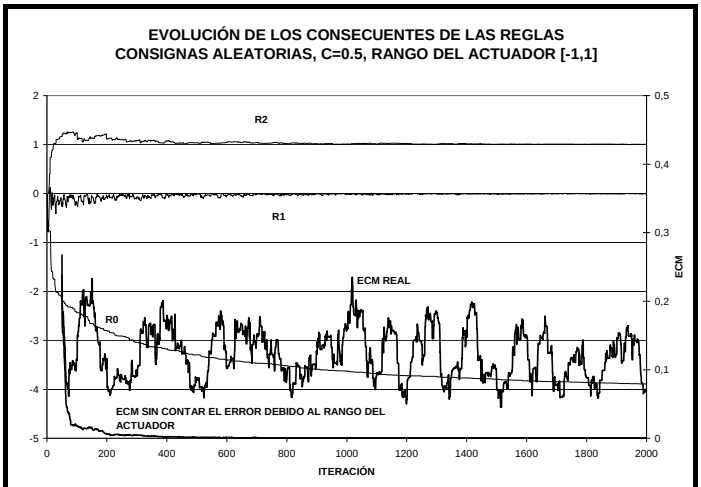


Fig. 6.3 Evolución limitando el rango del actuador al intervalo [-1,1].

Un ejemplo real donde esta característica se aprecia con mayor claridad es el caso del control del nivel en un tanque de líquido. Un tanque con nivel de líquido es un sistema no lineal entre la altura del nivel de líquido y el caudal efectivo que entra en el sistema. Dicho caudal efectivo será la diferencia entre el caudal de entrada (controlado por una válvula) y el de salida (que en este caso es un orificio de superficie efectiva A_s). Si llamamos C_1 a la capacidad hidráulica del tanque (que es igual a la superficie efectiva del tanque) y tenemos en cuenta que la velocidad de salida del líquido por el orificio (situado en el punto inferior del tanque) viene dada por $v_s(t) = \sqrt{2g\ y(t)}$ siendo *y(t)* el nivel de líquido en el tanque, tendremos que:

$$\Pi_{\text{tanque}} \equiv C_1 \frac{dy(t)}{dt} + C_2 \sqrt{y(t)} = Q_{entrada} \tag{6-6}$$

con $C_2 = \sqrt{2g \cdot A_s}$. Consideremos, a modo de ejemplo, un tanque de $1m^2$ de sección eficaz con un orificio abierto en su fondo de $0.2m^2$ y hagamos que el nivel siga un patrón de 1000 iteraciones compuesto por diversas funciones escalón entre 0.5 y 4m. El

periodo de muestreo será en este caso de 5s. El valor mínimo posible del actuador viene físicamente limitado por el valor 0 (cuando la válvula está completamente cerrada).

En la figura 6.4a se muestra la evolución del error cuadrático medio (multiplicado por 1000) exclusivamente debido al controlador después de la presentación de cada patrón para el caso de utilizar 2 variables de entrada (el error y el nivel de líquido), con 5 funciones de pertenencia en cada una de ellas. Como puede comprobarse, el error se hace prácticamente cero (hay que recordar que el ECM está multiplicado por 1000 para utilizar valores en los ejes más tratables) después de 30 presentaciones del patrón. En la figura 6.4b se representa la evolución durante un patrón completo en dicho instante. Se comprueba que la limitación del actuador no influye en el proceso de adaptación de las reglas, que se han ido ajustando para anular tanto los tiempos de subida, como el sobredisparo y el error en estado estacionario, mientras que durante las bajadas el controlador no puede hacer otra cosa que dejar la válvula cerrada y esperar al momento adecuado para actuar.

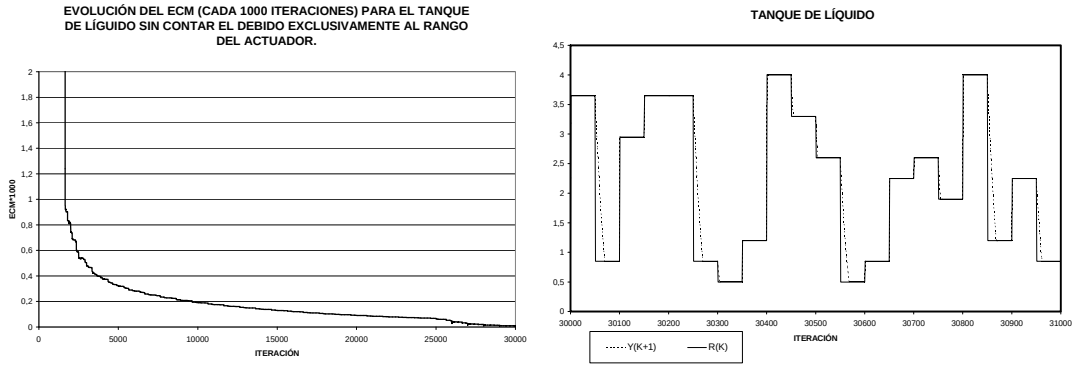


Fig. 6.4 a) Evolución del ECM en el tanque de líquido sin contar los errores debido exclusivamente al rango del actuador. b) Evolución del nivel del tanque durante un patrón completo.

Aceleración del proceso de convergencia: En la sección 5.4.1.2 del capítulo anterior se propuso una pequeña modificación en la política de adaptación del sistema auxiliar ca_1 para aquellos casos en los que la consigna variara suavemente o fuera constante. Dicha modificación consistía en aprovechar el rango teórico de validez del factor *C* para mejorar la velocidad de convergencia de las reglas. De esta forma, se proponía para estos casos una política de adaptación de la forma:

$$\Delta R_i(k+d) = \alpha_i(k) \cdot C \cdot F_{ca_1}\left(e(k+d), \dot{e}(k+d)\right) \cdot e(k+d) \qquad (6\text{-}7)$$

donde la función F_{ca_1} está expresada en forma de reglas difusas y su valor de salida debe estar comprendido en el intervalo [0,2]. Como ya se comentó, el papel de *C* tiene realmente su importancia en las primeras iteraciones (cuando las reglas no están aprendidas) por lo que el efecto de esta modificación sólo se podrá observar en dichos instantes iniciales. Consideremos para ello que las reglas difusas vienen dadas por la tabla 6.1 y utilicemos de nuevo la planta de la ecuación (6-3) en las mismas condiciones que en el primer ejemplo de esta sección salvo que las consignas serán ahora funciones escalón diversas repetidas cada 1000 iteraciones. En las figuras 6.5a y 6.5b se presenta la evolución del error cuadrático medio (cada 1000 iteraciones consecutivas) y de las reglas para los casos en los que no se utiliza la tabla 6.1 (equivalentemente F_{ca_1}=1) y en los que sí. De las figuras se aprecia como en el segundo caso las reglas convergen más rápidamente y el ECM decrece antes aunque la diferencia tampoco es excesivamente significativa (realmente insignificante si nos atenemos a la evolución completa del algoritmo).

e(k-1) / e(k)	N	Z	P
N	PB	PB	Z
Z	PM	PM	PM
B	Z	PB	PB

Tabla 6.1 Conjunto simplificado de reglas para el sistema auxiliar ca_1. PB = 2, PM = 1, Z = 0.

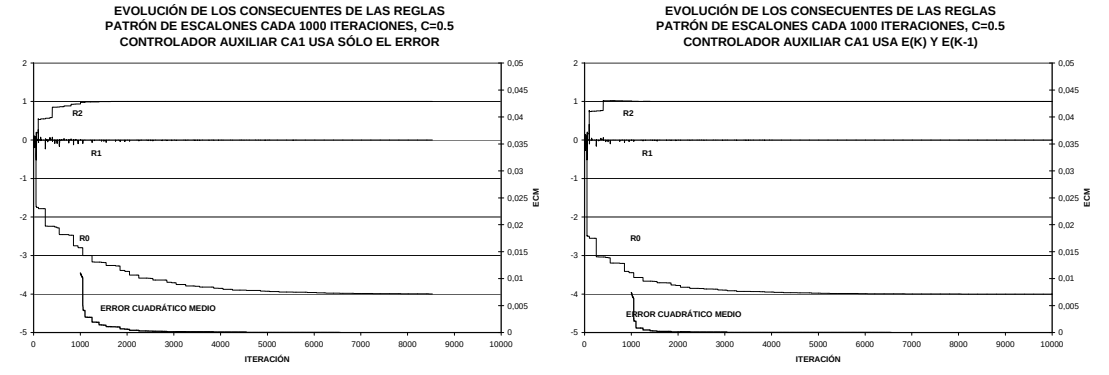


Fig. 6.5 Evolución del ECM y de las reglas para seguir un patrón de funciones escalón diversas. a) Sin aceleración del proceso b) Acelerando el proceso de convergencia usando e(k) y e(k-1).

Plantas con retrasos mayores que 1: Consideremos ahora la planta dada por la siguiente ecuación en diferencias:

$$\Pi_{retraso} \equiv y(k+10) = y(k) + \phi(u(k)) \tag{6-8}$$

El tratamiento para este tipo de plantas es completamente equivalente al resto de plantas con retrasos unitarios ya que el sistema siempre evaluará el error cometido con respecto

a la consigna proporcionada *d* instantes de tiempo anteriores (en lugar de en el instante anterior). Utilizaremos el error actual como entrada y 3 funciones de pertenencia situadas en -2, 0 y 2 (todo exactamente igual que en el primer ejemplo). La función de control óptima es, en este caso, la misma que en el primer ejemplo, es decir, la dada por:

$$u(k) = \phi^{-1}(e(k)) \tag{6-9}$$

con la única diferencia de que ahora el parámetro *C* debe ser 10 veces menor, es decir, C=0.05. Recordemos que *C* dependía de la pendiente máxima de cambio entre la señal de salida, *y(k+10)* en este ejemplo, con respecto a la entrada de control *u(k)*. Si en el caso de *d=1* dicha pendiente máxima era 2, ahora la variable de salida 10 iteraciones después puede haber cambiado 10 veces más por lo que la pendiente será 10 veces mayor. En las figuras 6.6a y 6.6b se representa la evolución del error cuadrático medio y de las reglas para el caso de consignas aleatorias y para el de un patrón de escalones de 1000 iteraciones, en el rango [-1,1]. La escala para el ECM se ha hecho menor que en el primer ejemplo para observar que efectivamente el sistema converge sin ningún problema a los valores óptimos, si bien dicha evolución es más lenta que en el caso de *d=1* ya que al principio, cuando las reglas no están aprendidas, los valores erróneos tardan más en ser corregidos.

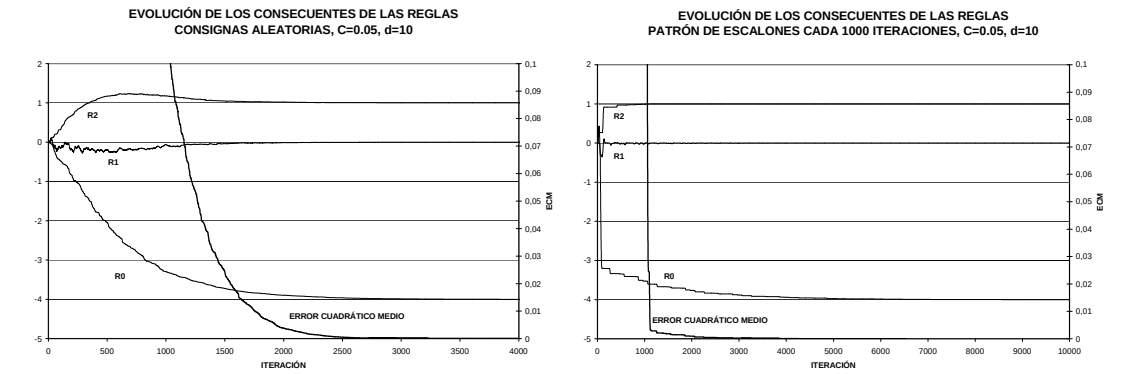


Fig. 6.6 Evolución del ECM y de las reglas para una planta con d=10 a) siguiendo consignas aleatorias b) siguiendo un patrón de 1000 iteraciones de funciones escalón diversas.

6.2.1.2 Importancia de la igualación de la integral del error cuadrático (IEC)

Tal y como se dijo en el capítulo anterior, a veces no es suficiente con adaptar solamente los consecuentes de las reglas sino que la localización de las funciones de pertenencia puede jugar un factor crítico en las prestaciones del proceso de control.

Modificando las funciones de pertenencia, el sistema puede ser capaz de conseguir una política de control mucho más adecuada sin la necesidad de aumentar el número de reglas utilizadas. En la sección 5.4.2 del capítulo anterior se presentó una metodología, capaz de co-actuar junto con el sistema auxiliar ca_1, que modificaba la localización de las funciones de pertenencia basándose también en el error en la salida de la planta. El objetivo de dicha metodología era intentar igualar la contribución a la integral del error cuadrático en cada una de las zonas definidas por las funciones de pertenencia, en analogía al apartado equivalente del algoritmo del capítulo 3. La filosofía es la misma que la utilizada entonces: debemos buscar un buen punto inicial para la etapa siguiente de ajuste fino y para ello no debemos permitir que haya regiones de operación donde el controlador actúe peor que en otras. Por supuesto, las regiones de operación más transitadas sufrirán mayores contribuciones a la integral del error cuadrático que deberán ser compensadas por los errores mayores de las regiones menos utilizadas. Si existen estados en los que la planta nunca se encuentre, la contribución de su zona asociada será cero por lo que el algoritmo tenderá a desplazar las funciones de pertenencia a otras zonas más necesitadas e importantes.

Para ello, el centro de cada función de pertenencia *j* de la variable *v* tenía asociada una pendiente dada por la diferencia del la contribución a la integral del error cuadrático medio durante un periodo global de control de sus zonas vecinas:

$$p_v^j = \frac{1}{r_y}\left(\int_t^{t+T'} e^2(\vec{x}^k)/x_v^k \in \left[c_v^{j-1}, c_v^j\right[- \int_t^{t+T'} e^2(\vec{x}^k)/x_v^k \in \left[c_v^j, c_v^{j+1}\right[\right) \tag{6-10}$$

La forma de actuación se hacía de forma alternada entre periodos globales de control de modo que se le diera tiempo al sistema adaptativo a ajustar los nuevos consecuentes de las reglas durante un periodo T' sin que influyan en el cómputo de la IEC.

Para comprobar el funcionamiento conjunto de la primera etapa completa consideremos el sistema definido por la siguiente ecuación diferencial:

$$\Pi_4 \equiv \frac{dy(t)}{dt} = \phi_2(u(t)) \tag{6-11}$$

siendo

$$\phi_2(x)=\begin{cases} 2x & \text{si x} > -1/2 \\ x/2-3/4 & \text{si x} < \text{-}1/2 \end{cases} \tag{6-12}$$

Esta planta es formalmente idéntica a la de la ecuación (6-3) con la diferencia de que ahora el valor del centro óptimo está desplazado al valor c_1=*-1*. Consideremos un controlador con una entrada (el error) y 3 funciones de pertenencia. De esta forma, la función inversa de la planta (el control perfecto) vendría dado por los centros situados en -2, -1 y 1 y el valor de las reglas sería -2.5, -0.5 y 1. En las figuras 6.7a y b se representa la evolución del ECM, de los consecuentes de las reglas y la del único centro móvil para el caso de T'=50 y T'=500 respectivamente, al intentar seguir consignas aleatorias en el rango [-1,1]. Como se aprecia en ambas figuras, tanto los consecuentes como el centro móvil acaban convergiendo a los valores óptimos, si bien se puede apreciar que en el primer caso lo hace mucho más rápido que en el segundo (como es lógico). Sin embargo también hay que tener en cuenta que si T' es pequeño, al principio, cuando las reglas todavía no están aprendidas, se pueden hacer movimientos en la dirección contraria a la que debiera (aunque esto se corrige de forma también rápida). Evidentemente, este es un caso muy simple y, en plantas más complejas, al igual que sucedía en el capítulo 3, el hecho de igualar la contribución a la IEC no implica que se vaya a encontrar el control perfecto. Esto se ve más claro en el siguiente ejemplo.

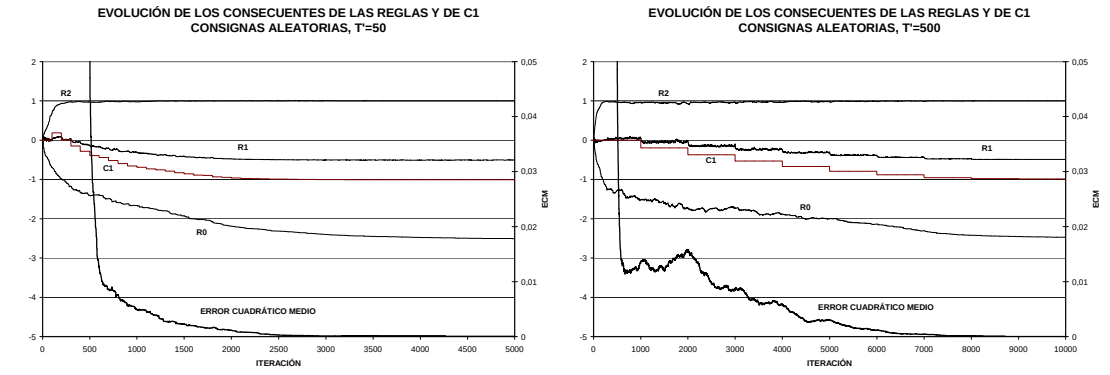


Fig. 6.7 Evolución del ECM, de las reglas y del único centro móvil a) T'=50 b) T'=500.

Consideremos la planta dada por la siguiente ecuación en diferencias:

$$\Pi_6 \equiv y(k+1) = -0.3\,sin(y(k)) + u(k) + u^3(k) \tag{6-13}$$

Esta planta es bastante más compleja que la anterior ya que una señal de control u(k)=0 no implica que la salida se quede estacionaria (incluso todo lo contrario, la salida puede

variar de signo). Además, la salida ya no depende linealmente con la señal de control. A pesar de eso es fácil de comprobar que la planta sigue manteniendo una monotonía fija (ascendente).

En este caso, consideraremos un sistema de control con dos variables de entrada (la consigna y la propia variable de salida) usando 3 funciones de pertenencia por cada una de ellas. La localización inicial de dichas funciones está en -1, -0.5 y 1 (para ambas variables). Se le pide al controlador que siga una consigna de tipo senoidal de periodo 50 iteraciones que se hace coincidir con T'. En la figura 6.8 se presenta la evolución del ECM para el caso de sólo utilizar el sistema auxiliar ca_1, para el caso de utilizar la etapa 1 completa y en el caso de usar la etapa 2 a continuación, en el instante en el que se produce la condición de convergencia de la etapa 1 (que en este caso ocurre en la iteración 12000). En dicha figura se pueden apreciar varias características importantes:

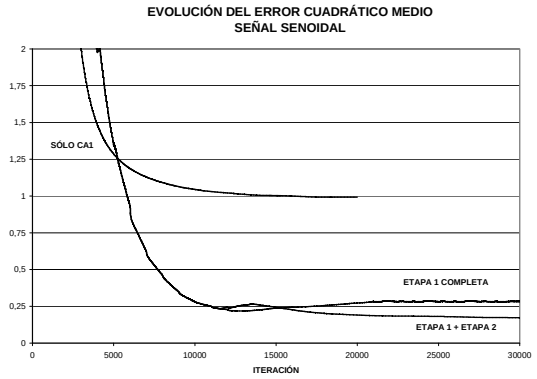


Fig. 6.8 Comparación de la evolución del ECM actuando sólo ca_1, etapa 1 completa, etapa 1 + etapa 2.

- El sistema auxiliar ca_1, cuando opera aisladamente, consigue adaptar los consecuentes de forma más rápida que cuando tiene que trabajar conjuntamente con el sistema igualador de la IEC. Esto es lógico ya que cada vez que se modifican los centros de las funciones de pertenencia los valores antiguos de las reglas dejan de ser los más adecuados.

- A pesar de lo anterior, la localización de los centros de las funciones de pertenencia juegan un papel primordial en las prestaciones del sistema de control (más importante cuanto menor es el número de éstos). Se comprueba que al modificar dichos centros, el valor final del ECM tras la etapa 1 es, en este caso, hasta 4 veces mejor que si actuara el sistema auxiliar 1 solamente.

- El estado límite en el que la IEC está igualada no representa de por sí el control óptimo ya que se aprecia como el estado estacionario final de la etapa 1 es algo mayor que otros estados por los que ha pasado anteriormente.

- Sin embargo, el incluir la igualación de dicho criterio sí da pie a un buen punto inicial para la segunda etapa de ajuste fino que es capaz de continuar reduciendo el ECM, aunque de una forma más lenta. Un análisis más detallado de la etapa 2 se mostrará en el apartado siguiente.

En la figura 6.9a se puede observar la evolución de la variable a controlar en las primeras 500 iteraciones de control. En este caso, al utilizar un valor de C=0.1 (varias veces por debajo del teórico) el sistema inicialmente evoluciona de forma más lenta (eso no impide que incluso existan oscilaciones importantes como se aprecia en la figura) pero las reglas acaban por alcanzar valores adecuados ya en la iteración 500. En las figuras 6.9b, c y d se muestra la evolución para los tres casos mencionados anteriores donde se puede apreciar de forma gráfica lo ya indicado anteriormente.

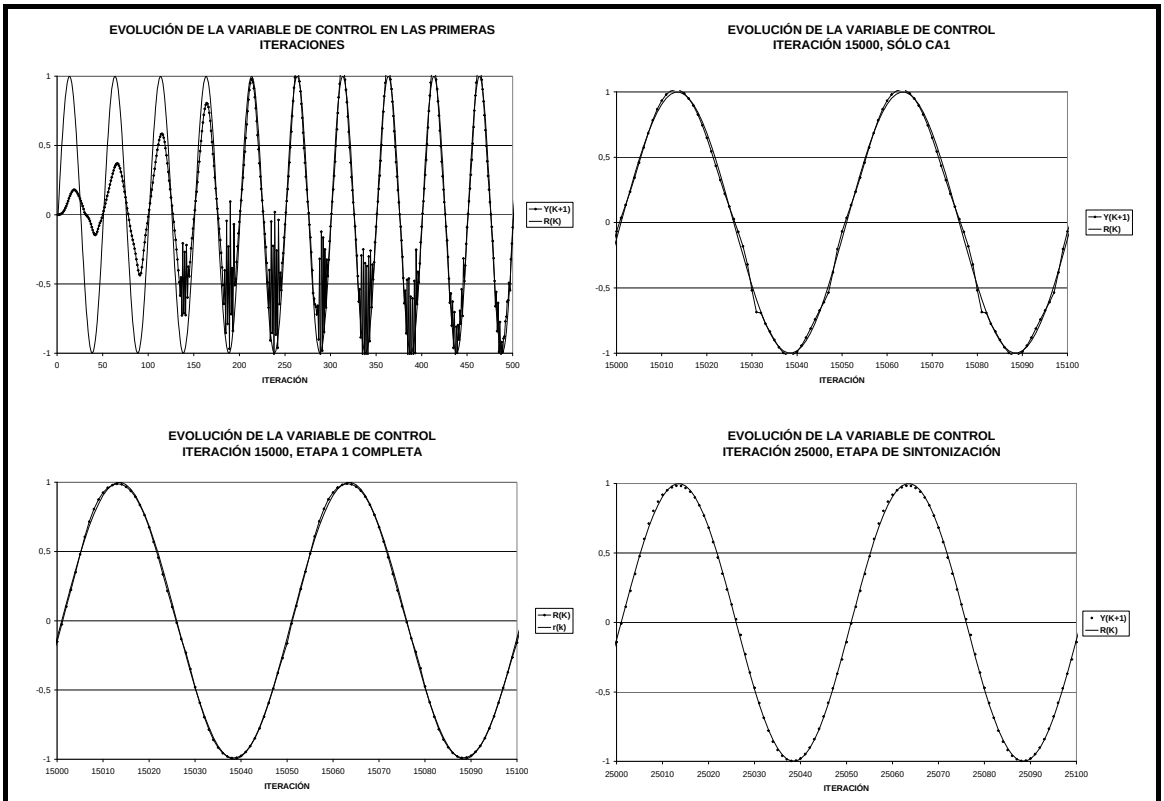


Fig. 6.9 Evolución de la variable a controlar a) primeras iteraciones b) sólo ca_1 c) tras la primera etapa d) durante la segunda etapa.

6.2.2 Adaptación mediante el error en la salida del controlador

La segunda etapa del algoritmo se basa en un ajuste fino tanto de los consecuentes de las reglas difusas como de los parámetros que definen las funciones de pertenencia, utilizando la información aportada por el error en la salida del controlador a través de un método de descenso en gradiente. Dicho ajuste se realiza por medio del sistema auxiliar ca_2. Como se comentó en el apartado 5.5.1 del capítulo anterior, el funcionamiento de dicho sistema auxiliar ya no depende de ningún tipo de parámetro adicional debido a que se basa en el conocimiento explícito de las derivadas parciales de la salida del controlador respecto a cada uno de sus parámetros. Por otro lado, el método presentaba la desventaja de que no siempre se cumplía que los cambios impuestos por tal sistema sobre el controlador principal fueran en la dirección en la que se reducía el error actual en la salida de la planta, por lo que se hizo necesaria la inserción de un sistema supervisor que nos asegurara que tal condición se cumpliera.

6.2.2.1 Sistema auxiliar ca_2

Ya en el apartado anterior se ha dado una muestra de cómo mejora el rendimiento la etapa de sintonización fina. La principal ventaja de esta etapa es que no requiere ningún modelo de la planta a controlar ni depende de factores externos como en el caso de la constante *C* en el sistema auxiliar ca_1. El sistema auxiliar ca_2 sólo se fija en el error en la salida del controlador y, como conocemos el funcionamiento interno de éste, la sintonización es más exacta, aunque como ya se ha dicho, se requiere que la planta ya esté relativamente bien controlada, extremo al que teóricamente ya se ha llegado durante la etapa inicial.

Recordemos que el sistema auxiliar ca_2 adapta el valor de cada uno de los parámetros que definen al controlador (reglas y funciones de pertenencia) de la forma:

$$\theta_i(k+1) = \theta_i(k) - \eta \frac{\partial J(k)}{\partial \theta_i} \tag{6-14}$$

siendo *J(k)* el error al cuadrado en la salida del controlador. Dicho error viene dado por:

$$e_u(k+d) = u(k) - \hat{u}(k) \tag{6-15}$$

donde *u(k)* es la salida del controlador en el instante *k*, que fue obtenida para alcanzar un valor de consigna *r(k)* pero, aunque no lo consiguiera, sabemos que es la óptima

para, en las mismas condiciones, obtener la salida que realmente se dio *y(k+d)*. *û(k)* es la salida que daría el controlador (que no es la que realmente se introduce en la planta) con los parámetros existentes en el instante *k+d* (que es cuando conocemos *y(k+d)*) para el caso *r(k)=y(k+d)*, y cuyo valor óptimo ya sabemos que es *u(k)*. Matemáticamente:

$$\begin{aligned} u(k) &= \hat{F}\left(\vec{x}(k);\Theta(k)\right) = F\left(\hat{x}(k)\right) \\ \hat{u}(k) &= \hat{F}\left(\hat{x}(k);\Theta(k+d)\right) \\ \hat{x}(k) &= (y(k+d),y(k),\ldots,y(k-p),u(k-1),\ldots,u(k-q)) \end{aligned} \tag{6-16}$$

Para ver un primer ejemplo del funcionamiento de este sistema auxiliar, consideremos la planta definida por la siguiente ecuación en diferencias:

$$\Pi_1 \equiv y(k+1) = 0.8\, sen(2\, y(k)) + 1.2\, u(k) \tag{6-17}$$

que deberá seguir un patrón de consignas aleatorias en el rango [-1,1]. Las variables de entrada son el error en la salida de la planta y la propia salida de la misma, con 4 funciones de pertenencia equidistribuidas en cada variable. En la figura 6.10 se representa el error cuadrático medio (multiplicado por 1000) donde se puede apreciar claramente el salto cuantitativo que produce esta etapa de ajuste fino cuando se ejecuta a partir de la iteración 28000.

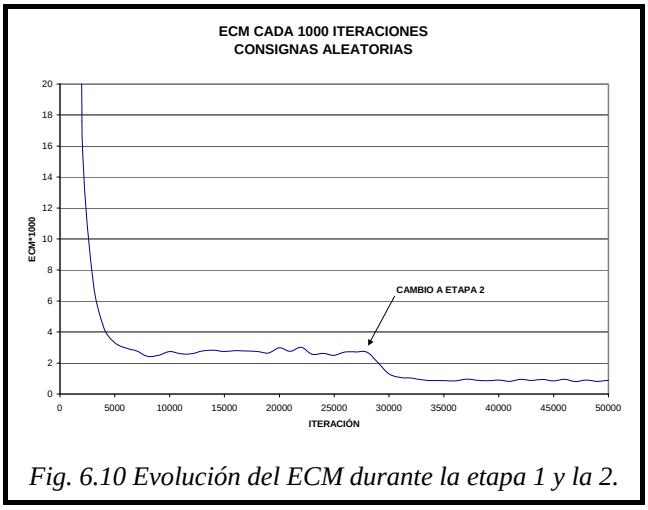


Fig. 6.10 Evolución del ECM durante la etapa 1 y la 2.

Un comportamiento análogo se puede apreciar al considerar el siguiente sistema a controlar:

$$\Pi_5 \equiv y(k+1) = \frac{1.5\, y(k-1)\, y(k)}{1+y^2(k-1)+y^2(k)} + 0.35\, sen\left(y(k-1)+y(k)\right) + 1.2\, u(k) \tag{6-18}$$

donde ahora utilizamos 3 variables de entrada (r(k), y(k), y(k-1)) con dos funciones de pertenencia para la primera variable y 4 para el resto (ya veremos en la sección 6.2.3 que esta distribución no es caprichosa). En la figura 6.11a se observa la evolución del ECM al intentar seguir consignas aleatorias en el rango [-1,1]. En la figura se aprecia que la etapa de ajuste fino consigue mejorar las prestaciones del controlador de forma excelente.

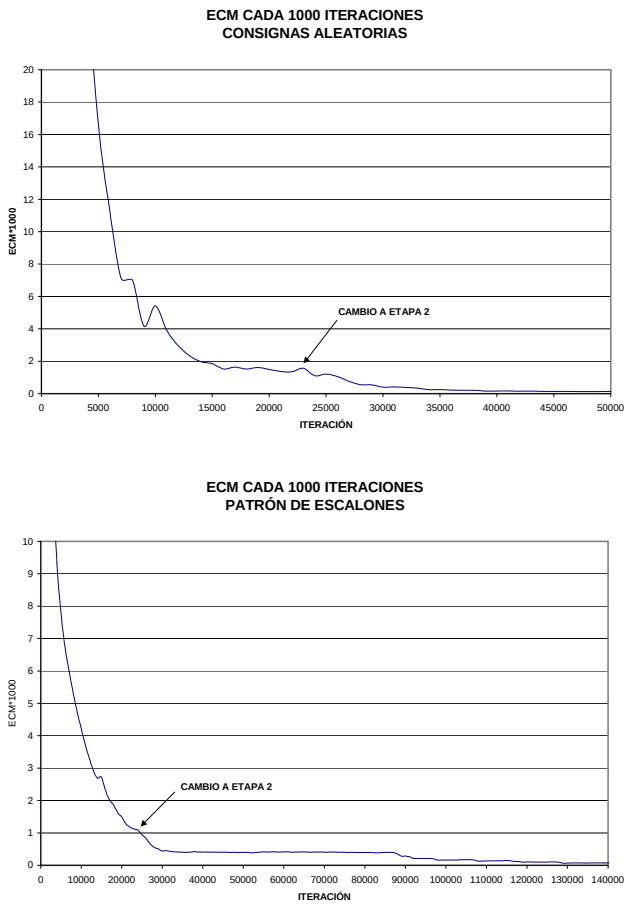


Fig. 6.11 Evolución del ECM a) consignas aleatorias b) patrón de escalones con una condición suave de salida de la etapa 1.

En la figura 6.11b se representa el mismo caso anterior pero ahora intentando seguir un patrón fijo de 1000 iteraciones compuesto por diversas funciones escalón e imponiendo una condición de convergencia más suave a la etapa 1 (10 % del rango local en lugar del 1 %). En la figura se observa que se conmuta a la etapa 2 tras la presentación de 24 patrones y que este cambio ha sido un poco prematuro ya que, aunque el error disminuye muy rápidamente al principio, el sistema auxiliar necesita una gran cantidad de iteraciones para poder encontrar los valores realmente adecuados para la

configuración de la que se trata. En las figuras 6.12a, b y c se representa la evolución real de la variable a controlar durante las primeras iteraciones, justo antes de conmutar a la etapa 2 y durante esta última etapa, respectivamente.

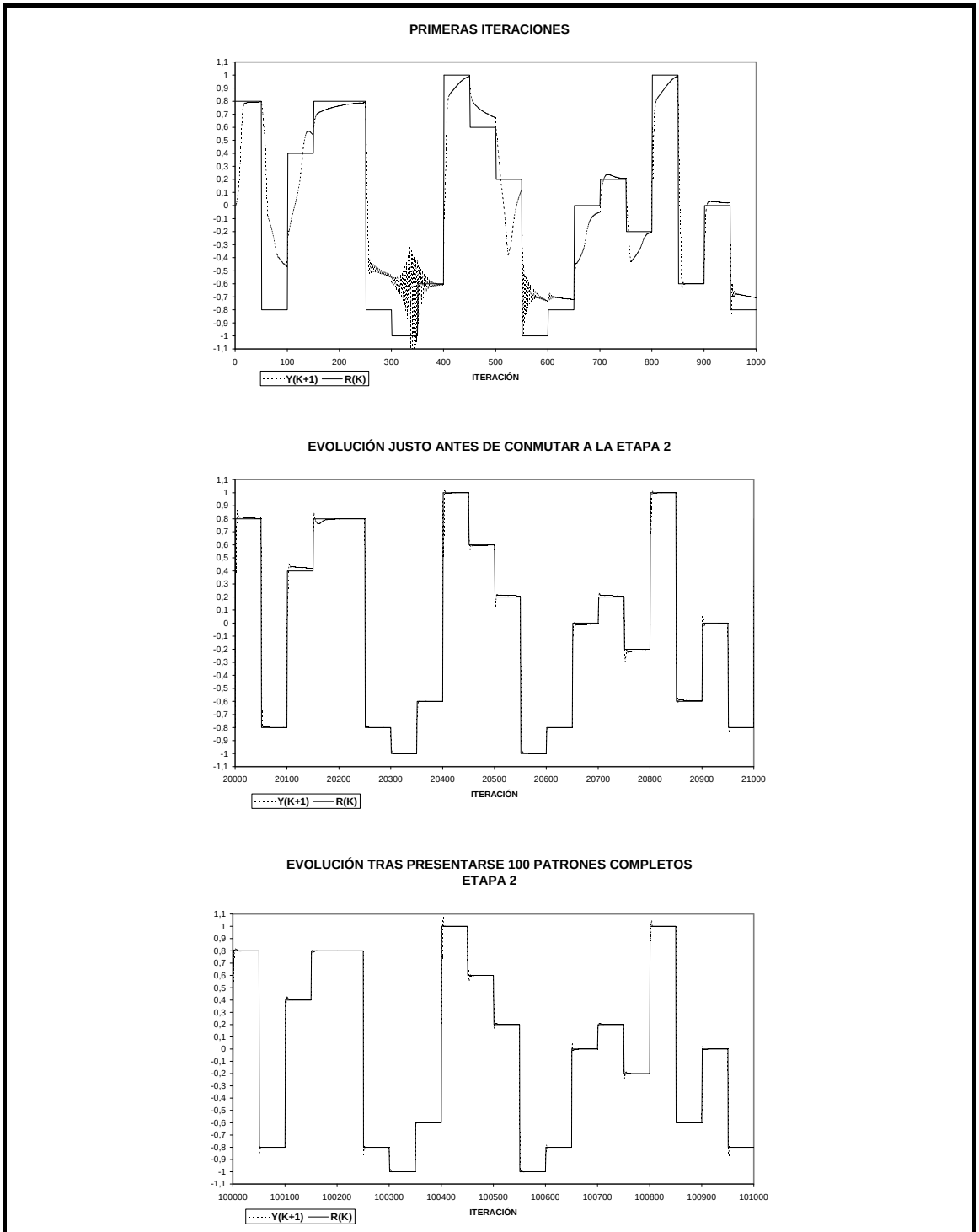


Fig. 6.12 Evolución de la variable a controlar a) primeras iteraciones b) antes de conmutar a la etapa 2 c) durante la etapa 2

Como tercer ejemplo, consideraremos uno de los sistemas que más frecuentemente aparecen en la bibliografía: el sistema del péndulo invertido [CHE-89, BER-92, WAN-94]. Este sistema consiste en un péndulo montado sobre un carro que puede ser propulsado mediante una fuerza exterior (figura 6.13). En este ejemplo, consideraremos el problema de controlar el ángulo descrito por el péndulo, sin preocuparnos de la posición espacial del carro. Como es sabido, este sistema se considera inestable en el sentido de que el péndulo puede caer en cualquier momento salvo que esté situado justamente en la posición de equilibrio ($\theta=0$). Las ecuaciones diferenciales exactas que rigen en el sistema son:

$$\Pi_{\text{péndulo}} \equiv \frac{d^2\theta(t)}{dt^2} = \frac{g\,sin(\theta) + \frac{cos(\theta)}{m_c+m}\left[u(t) - ml\dot{\theta}^2\,sen(\theta)\right]}{\left(\frac{4}{3} - \frac{m\,cos^2(\theta)}{m_c+m}\right)\cdot l} \tag{6-19}$$

donde g es la aceleración de la gravedad (9.8 m/s^2), m es la masa del péndulo (1Kg), m_c la del carro (0.1Kg) y l es la semilongitud del péndulo (0.5m). A diferencia de otros autores, que linealizan la ecuación anterior para trabajar en rangos de ángulos pequeños, en nuestro caso vamos a llevar el péndulo por valores angulares grandes de modo que el comportamiento puede ser fuertemente alineal.

Las condiciones para la simulación serán las siguientes:

1.- Debido a que se parte de un conjunto vacío de reglas, consideraremos que el péndulo presenta unos topes situados en ±45º de modo que éste jamás atravesará dichos límites.

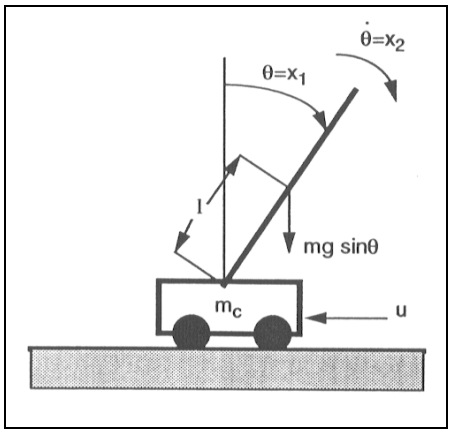


Fig. 6.13 Sistema del péndulo invertido.

2.- El periodo de muestreo para el controlador será de 0.1s.

3.- Utilizaremos como entradas al controlador el error angular, el ángulo actual, el ángulo en el instante anterior y la señal de control en el instante anterior, todas ellas con dos funciones de pertenencia (por tanto 8 reglas).

4.- Se exigirá que el péndulo describa una señal del tipo *r(t) = 0.5 sen(t)*, es decir, ángulos entre -0.5 y 0.5 radianes (±29º).

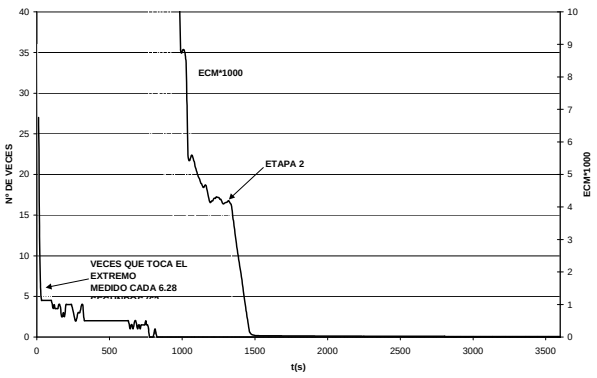


Fig. 6.14 Número de veces que el péndulo alcanza los topes y evolución del ECM para el sistema del péndulo invertido.

Los resultados de la simulación se recogen en la figura 6.14 donde se ha representado en el primer eje de ordenadas el número de veces que toca el péndulo los topes de 45º por cada periodo de la función seno y en el segundo eje el ECM (multiplicado por 1000) para las diferentes etapas del algoritmo. En la figura se puede apreciar como, al principio, cuando las reglas no están aprendidas, el péndulo rebasa los límites permitidos un gran número de veces y que poco a poco las reglas van alcanzando los valores adecuados hasta conseguir un sistema estable. En cuanto al ECM, la inclusión de la segunda etapa es realmente vital ya que se trata de un sistema altamente sensible. En cuanto el sistema conmuta a la etapa de sintonización fina, los parámetros se ajustan enormemente bien consiguiendo un control prácticamente perfecto. En las figuras 6.15a y 6.15b se representa la evolución del ángulo que describe el péndulo y la de la señal de control justo antes de conmutar a la etapa 2 y durante dicha etapa. La diferencia entre ambas gráficas es realmente notoria.

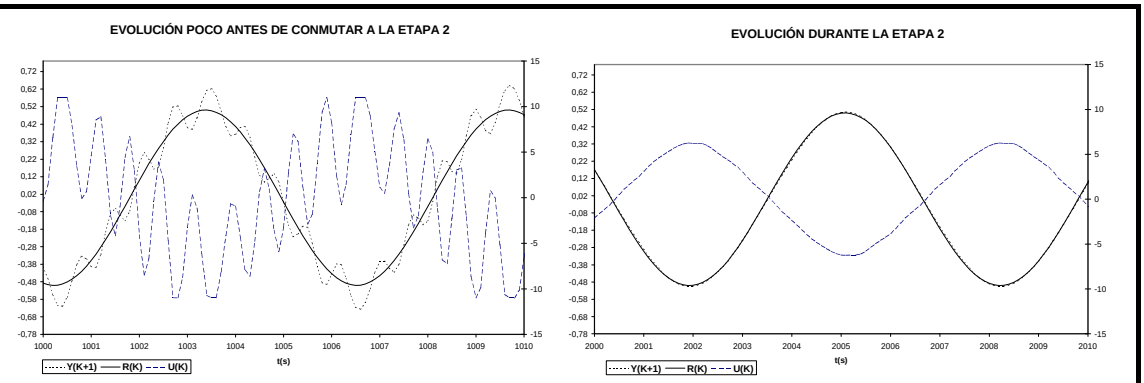


Fig. 6.15 Situación angular del péndulo y variable de control a) antes de conmutar a la etapa 2 b) durante la etapa 2.

A modo de ejemplo, en [WAN-94] se propone un algoritmo también de control en tiempo real y se aplica a este mismo ejemplo aunque con una señal de amplitud mucho menor: $r(t)=\frac{\pi}{30}sen(t)$. En este algoritmo, que actuaba en tiempo continuo, L-X. Wang necesitaba un entrenamiento previo de los parámetros del controlador (que contaba con 25 reglas) además de una serie de acotaciones sobre los elementos de la ecuación diferencial. En la figura 6.16 se refleja el resultado que obtuvo el autor donde se comprueba que aunque el control es estable, no llega al grado de precisión obtenido en la figura 6.15b para amplitudes 5 veces mayores. El problema del control del péndulo invertido se volverá a utilizar para analizar otra serie de factores en la sección 6.3.2.

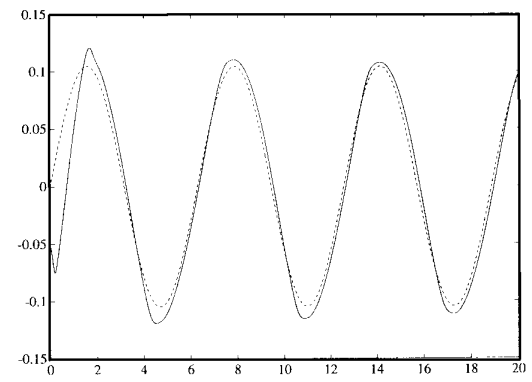


Fig. 6.16 Evolución obtenida por L-X. Wang para el problema del péndulo invertido.

6.2.2.2 Sistema supervisor

En la sección 5.5.2 del capítulo anterior se justificó la necesidad de un segundo sistema que supervisara el funcionamiento del sistema auxiliar ca_2. Para dicho segundo sistema se escogió el propio ca_1 ya que, si bien éste no es capaz de realizar un ajuste tan fino como ca_2, sí era capaz de evaluar si los cambios aconsejados por ca_2 iban en la dirección en la que se redujera el error actual en la salida de la planta.

Para ello, el sistema supervisor evalúa la salida que daría el controlador $u_{k-d}(2)$ tras las modificaciones de ca_2, para d instantes de tiempo anteriores y la compara con la señal de control que se introdujo en la planta en dicho instante $u_{k-d}(1)$. Si la diferencia no está entre los límites impuestos por las ecuaciones:

$$\begin{aligned} &\text{Si } e_k > 0 \quad \rightarrow \quad 0 \le u_{k-d}(2) - u_{k-d}(1) \le C \cdot e_k \\ &\text{Si } e_k < 0 \quad \rightarrow \quad C \cdot e_k \le u_{k-d}(2) - u_{k-d}(1) \le 0 \end{aligned} \tag{6-20}$$

el supervisor actúa, deshace los cambios realizados por ca_2 e impone su propias modificaciones (sólo en los consecuentes de las reglas, naturalmente).

La necesidad del sistema supervisor se pone de manifiesto principalmente cuando el número de reglas es demasiado pequeño como para poder realizar una política de control fina. Como primer ejemplo, consideremos nuevamente la planta:

$$\Pi_6 \equiv y(k+1) = -0.3\,sin(y(k)) + u(k) + u^3(k) \qquad (6\text{-}21)$$

y definiremos dos variables de entrada (el error y la salida de la planta) con 3 funciones de pertenencia en cada una de ellas. Las consignas responderán a un patrón de 100 iteraciones compuesto por una función seno y un escalón doble en el rango [-1.5, 1.5]. Este rango obliga a valores de la señal de control altos por lo que la dependencia polinomial de tercer orden afecta de forma significativa. En la figura 6.17a se representa la evolución del ECM para este problema "con" y "sin" la existencia del sistema supervisor. En el segundo eje de ordenadas se muestra el número de veces que actúa el sistema supervisor, por cada periodo de la consigna, para el primer caso.

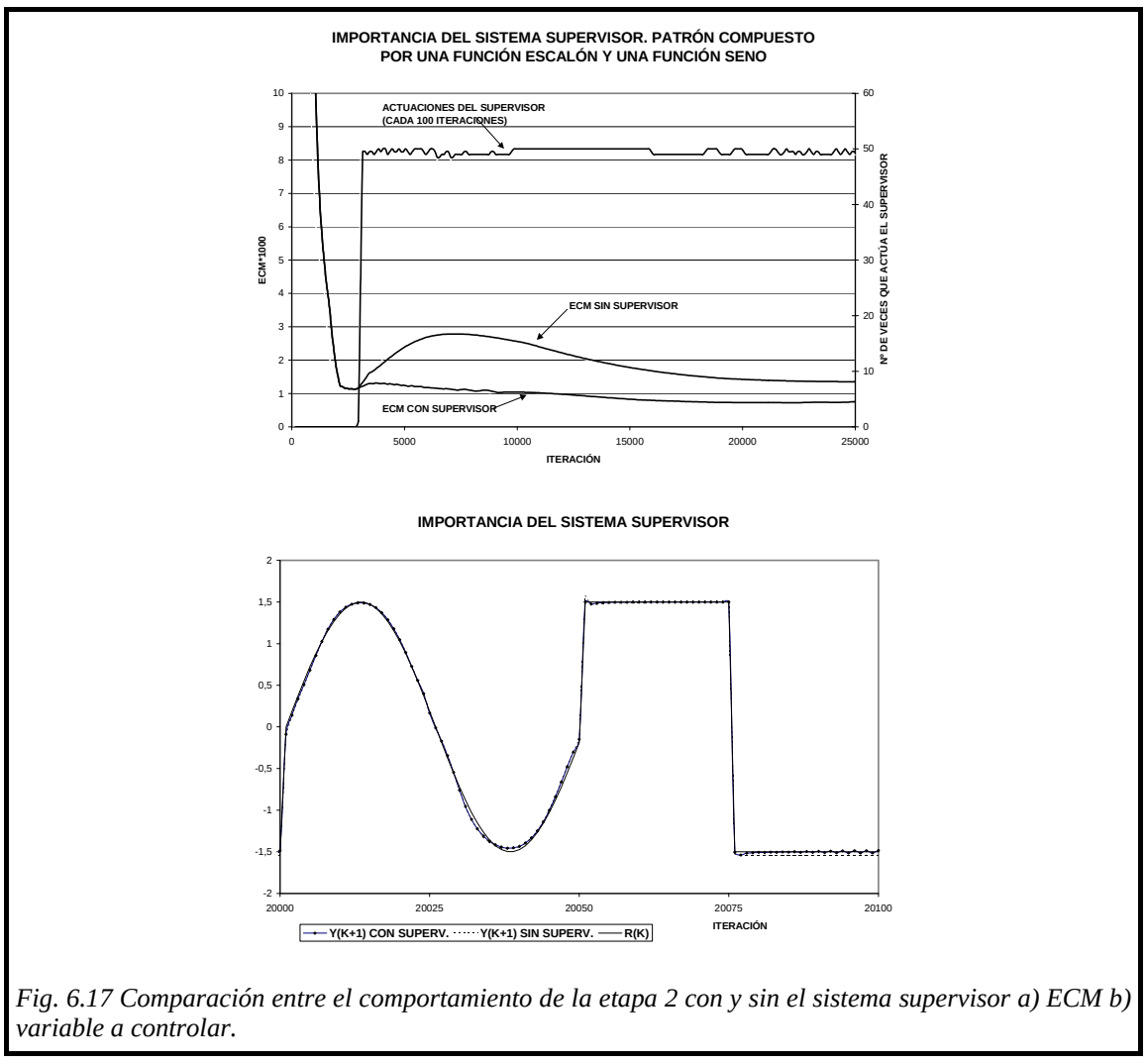


Fig. 6.17 Comparación entre el comportamiento de la etapa 2 con y sin el sistema supervisor a) ECM b) variable a controlar.

A partir de la figura, se observa cómo el sistema supervisor reacciona ante los cambios propuestos por el sistema ca_2 que no vayan dirigidos al objetivo prioritario, que es la disminución el error actual de la planta. Sin la existencia del sistema supervisor, el ECM durante la etapa 2 empieza a crecer de forma muy significativa aunque finalmente se recupera sin llegar a los límites conseguidos durante la primera etapa. En este caso con tan pocas funciones de pertenencia, el supervisor ha tenido que actuar prácticamente en el 50 % de las ocasiones, principalmente durante la función escalón ya que ante este tipo de funciones en los que el valor de consigna no varía, en el sistema auxiliar ca_2 se incrementa el riesgo de quedarse estancado al repetirse siempre el mismo valor óptimo de la función inversa de la planta del que tiene que aprender. En la figura 6.17b se muestra la evolución de la señal a controlar para ambos casos. En las últimas 25 iteraciones de la figura se puede comprobar el fenómeno al que nos referimos.

Un ejemplo mucho más espectacular de este fenómeno se produce si intentáramos ejecutar el algoritmo sin la existencia de la primera etapa, es decir, comenzando directamente con el sistema auxiliar ca_2. Consideremos, por ejemplo, el sistema:

$$\Pi_5 \equiv y(k+1) = \frac{1.5\, y(k-1)\, y(k)}{1+y^2(k-1)+y^2(k)} + 0.35\, sen\big(y(k-1)+y(k)\big) + 1.2\, u(k) \qquad (6\text{-}22)$$

definiendo tres variables de entrada (*r(k), y(k), y(k-1)*) y una configuración de 2x5x5 funciones de pertenencia. Las consignas serán ahora un patrón de 1000 iteraciones de funciones escalón diversas en el rango [-1.5,1.5]. En la figura 6.18 se muestra la evolución de la variable a controlar durante las primeras iteraciones "con" y "sin" sistema supervisor. En el segundo caso, al comenzar las reglas con el valor 0 y cumplirse que *y(k+1) = 0* para *u(k) = 0*, el primer dato del que aprende el sistema auxiliar ca_2 es que si *y(k)=0*, *y(k-1)=0* y *r(k)* fuera *0* entonces *u(k) = 0*. Pero para ese valor el sistema ya obtiene un $e_u(k) = 0$ por lo que ningún parámetro se adapta y el sistema permanece siempre al valor cero independientemente del

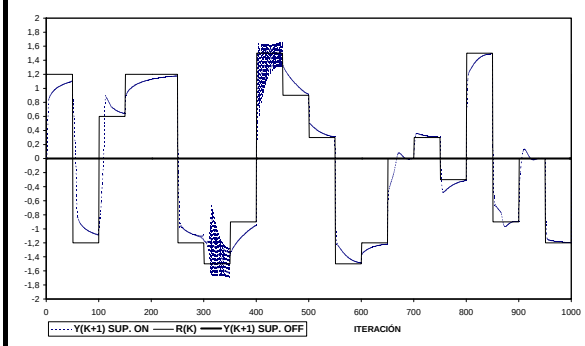


Fig. 6.18 Comparación durante las primeras iteraciones comenzando directamente por la etapa 2 con y sin supervisor.

valor de la consigna. Sin embargo, al utilizar un sistema supervisor, éste comprueba que los cambios se hagan en la dirección correcta por lo que si no es así, revierte los posibles cambios de ca_2 y asume el control.

En la figura 6.19a se observa la evolución del ECM para esta planta, utilizando directamente la etapa 2 desde el principio con la ayuda del sistema supervisor. En el primer eje de ordenadas se representa el número de intervenciones de éste por cada periodo del patrón. Se observa como, a medida que las reglas van asimilando los valores correctos, el sistema supervisor va decrementando su influencia sin que al final se consiga la total independencia del controlador con el supervisor. En la figura 6.19b se muestra la misma evolución anterior junto con la evolución normal del algoritmo (comenzando por la etapa 1). Se observa claramente que la inclusión de la primera etapa es, no sólo conveniente para la aceleración del proceso de convergencia, sino que además el valor final del ECM es bastante menor. Además, en este segundo caso, el supervisor no ha sido necesario en ningún momento. En la figura 6.19c se observa la variable a controlar durante la segunda etapa de la evolución normal del algoritmo.

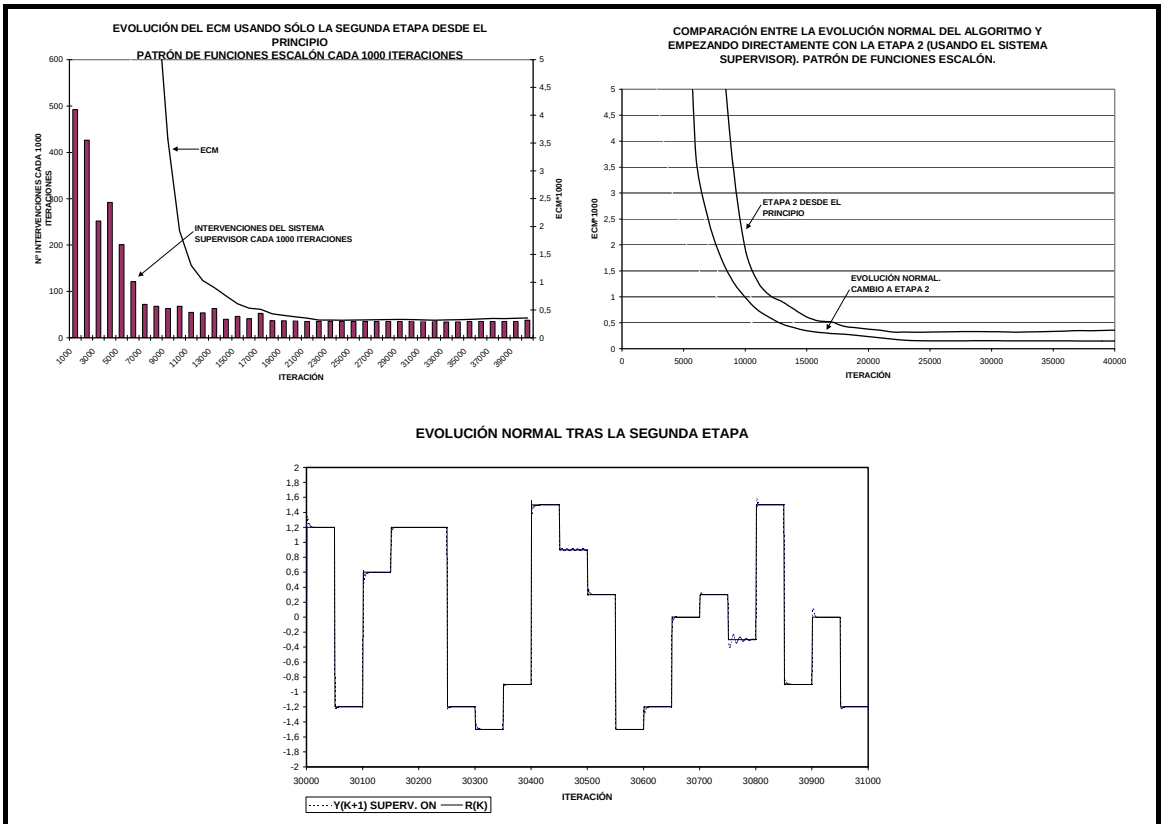


Fig. 6.19 Evolución del algoritmo comenzando directamente por la etapa 2 desde el principio a) ECM e intervenciones del supervisor b) comparación del ECM con el obtenido con la evolución normal del algoritmo c) variable de salida con la evolución normal del algoritmo.

6.2.3 Adición de funciones de pertenencia

En todos los ejemplos anteriores se ha supuesto, sin justificación, una determinada configuración inicial de funciones de pertenencia en cada variable de entrada. Cuando se aborda un problema de control, a partir del conocimiento cualitativo de la planta a controlar siempre se pueden estimar qué variables de entrada son las que pueden influir en la evolución del sistema pero nunca podemos estar seguros del grado de precisión con la que se debe considerar cada una de ellas ni si realmente alguna variable de entrada que creemos necesaria influye realmente de forma significativa en el proceso.

Cuando, durante la etapa 2, los parámetros del controlador apenas varían ya de forma significativa, podemos considerar que las prestaciones del proceso de control apenas van a variar apreciablemente y puede ocurrir que la política de control todavía no sea la esperada. En estos casos, la solución habitual siempre ha sido la de utilizar un mayor número de reglas aumentando el número de funciones de pertenencia en alguna variable y probar a modo de ensayo-error si la nueva topología introducida cumple los requisitos.

En el capítulo anterior, sin embargo, se introdujo una nueva metodología para poder seleccionar en tiempo real qué variable o variables de entrada son las que realmente habría que cuantificar con más precisión para mejorar las prestaciones globales del proceso. Esta es una característica que dota al algoritmo propuesto de una versatilidad enorme ya que no tenemos que ir probando distintas configuraciones de forma casi aleatoria hasta conseguir la política de control correcta, sino que todo este proceso se realiza de forma automática y en tiempo real.

El método utilizado se basa en un análisis de los últimos M datos recopilados por el sistema auxiliar ca_2 pertenecientes a la verdadera función inversa de la planta. El proceso se realiza de forma concurrente tras la verificación de la condición de convergencia de la etapa 2 y es totalmente equivalente al presentado en la sección 3.5 del capítulo 3, con la diferencia de que ahora debemos dar un valor inicial a las nuevas reglas obtenidas, utilizando la teoría expuesta en la sección 5.6.1.

Consideremos como primer ejemplo la planta:

$$\Pi_1 \equiv y(k+1) = 0.8\, sen(2\, y(k)) + 1.2\, u(k) \qquad (6\text{-}23)$$

Vamos a seleccionar como variables de entrada *r(k)* e *y(k)* con un valor inicial de dos funciones de pertenencia para cada una de ellas. Del conocimiento de la planta, sabemos que ambas entradas son necesarias pero desconocemos cual debe ser el número de funciones que particionen cada una de ellas. Como consignas utilizaremos aquí un patrón de 1000 iteraciones compuesto por una mezcla de funciones escalón, senoidales y rampas en el intervalo [-1,1]. El tamaño de la memoria de tipo cola se hará coincidir con la longitud del patrón, es decir, 1000 datos. En la figura 6.20 se muestra la evolución del ECM y las decisiones tomadas por el algoritmo durante 260 presentaciones del patrón completo. La escala del eje de ordenadas se ha seleccionado de forma logarítmica para poder observar mejor dicha evolución. Como se puede comprobar en la figura, el algoritmo siempre escoge la variable segunda como la seleccionada para añadir una nueva función de pertenencia. En la tabla 6.2 se demuestra que las decisiones son siempre correctas ya que siempre el ECM tras la etapa 2 en cada caso es mucho mejor para la configuración seleccionada que para el caso de utilizar la decisión contraria. En dicha tabla también figuran los valores de J_1 y J_2 para cada variable seleccionada (ver sección 3.5). A lo largo de este capítulo utilizaremos como valor umbral de selección el mismo que se utilizó en la teoría presentada para aproximación de funciones, es decir, un 20% del valor máximo. Finalmente, en las figuras 6.21a y 6.21b se puede observar la evolución de la variable a controlar durante las primeras iteraciones del algoritmo (con la configuración inicial de 2x2) y la obtenida tras la etapa 2 de la configuración 2x6 (con tan solo 12 reglas). Como se puede comprobar, el aprendizaje tanto de los parámetros del controlador como de la topología del mismo en tiempo real se realiza de forma muy favorable.

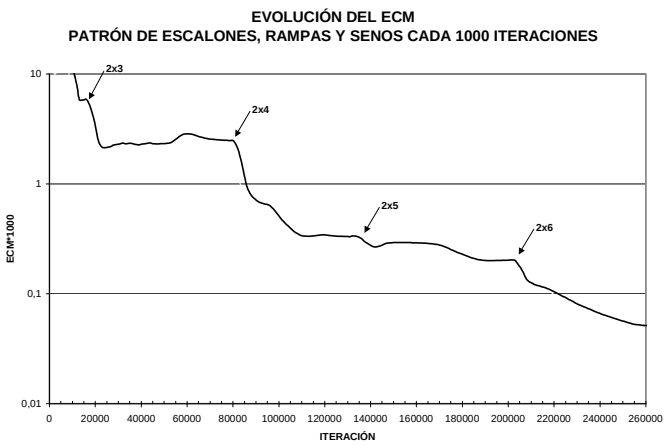


Fig. 6.20 Evolución del ECM para distintas configuraciones encontradas por el algoritmo.

Config.	ECM (x10^3) tras etapa 2	J_1	J_2	ECM (x10^3) tras etapa 2 (Caso contrario)
2x2	5.78	0.0685	0.2	
2x3	2.48	0.0094	0.175	3x2: 3.917
2x4	0.33	0.00554	0.0192	3x3: 2.047
2x5	0.20	0.0092	0.0288	3x4: 0.260
2x6	0.048	0.0076	0.0336	3x5: 0.204

Tabla 6.2 Evolución del algoritmo para las distintas configuraciones por las que transita (Π_1).

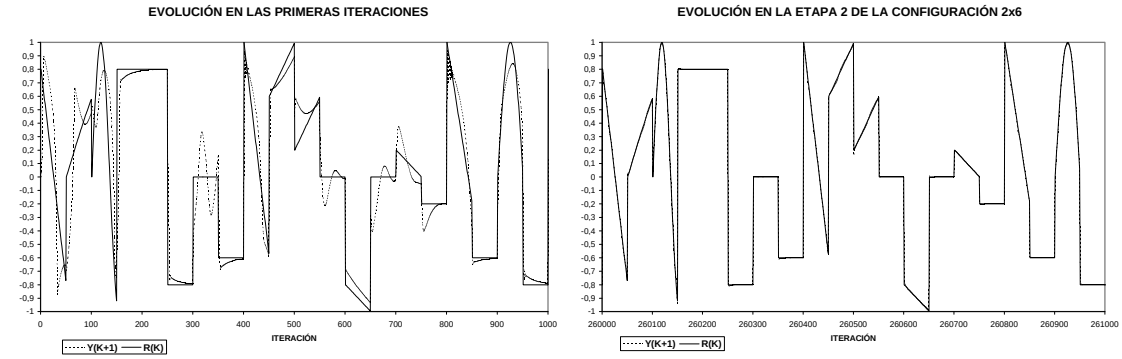


Fig. 6.21 Evolución de la variable a controlar a) durante las primeras iteraciones (2x2) b) tras la etapa 2 de la configuración 2x6.

Como ya se indicó en el capítulo anterior, la metodología de selección de variables de pertenencia es igualmente válida cuando una variable tiene solamente definida un sola función, que es equivalente a no considerarla. Por otro lado, si las consignas evolucionan de forma aleatoria, teóricamente la planta pasará por todas las regiones de operación posibles, por lo que el controlador tendrá que aproximar prácticamente igual de bien cada una de las partes de la función real inversa de la planta y el problema esencialmente se traduce en el de aproximación funcional con la complejidad añadida de que debe hacerse en tiempo real y que la salida de la planta ha de estar controlada en todo momento.

Para ver que esto efectivamente es así, volvemos a considerar la planta dada por las siguientes ecuaciones en diferencias:

$$\Pi_6 \equiv y(k+1) = -0.3\, sin(\, y(k)) + u(k) + u^3(k) \qquad (6\text{-}24)$$

donde vamos a utilizar como variables de entrada nuevamente *r(k)* e *y(k)*, pero asignándoles inicialmente sólo una función de pertenencia a cada una de ellas. De esta

forma, el controlador inicialmente estará compuesto simplemente por una regla que, además, tiene un valor inicial de 0, es decir, un controlador completamente vacío. Haremos variar la consigna de forma aleatoria en el rango [-4,4] y nuevamente utilizaremos una memoria que almacenará los últimos 1000 datos obtenidos. En la figura 6.22 se representa la función de la planta a controlar y en la 6.23a su función inversa respecto a la variable de control, es decir, la función que teóricamente debería aproximar nuestro controlador. Los datos correspondientes a la evolución del algoritmo se reflejan en la tabla 6.3 donde se especifica el error cuadrático medio (multiplicado por 1000) para cada una de las configuraciones por las que ha ido evolucionando el controlador.

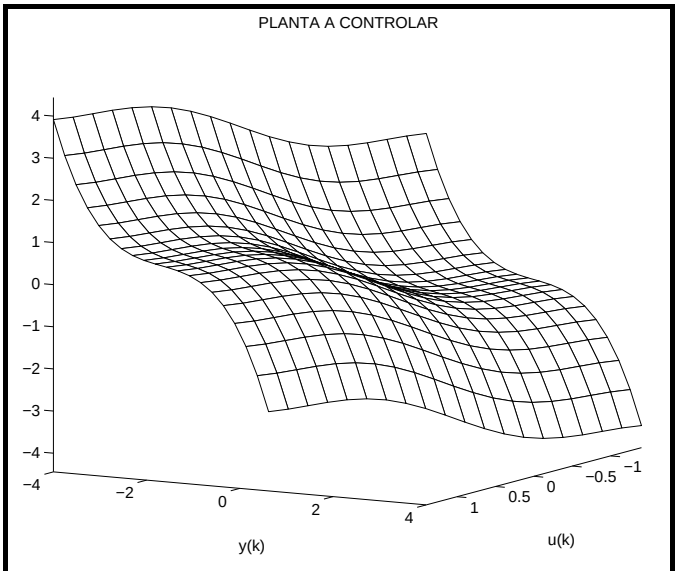


Fig. 6.22 Función de la planta a controlar.

Lo primero que observamos es que, al principio, el proceso de control se da cuenta de que la variable primera *r(k)* es mucho más necesaria que la segunda lo cual es evidente si observamos la figura 6.23a ya que la influencia de la parte senoidal de (6-24) es mucho menor que la de la consigna. En las figuras 6.23b y 6.23c se representan las distintas funciones que implementa el controlador tras llegar a la convergencia para las configuraciones 2x1 y 4x1 respectivamente. A partir de la configuración 4x1, si se quiere conseguir una política de control más fina, se debe recurrir ya a la segunda variable. De esta forma, el controlador evoluciona atravesando la configuración 4x4 (figura 6.23d), 5x4, 6x4 (figura 6.23e), etc. hasta llegar a la configuración 10x7 (figura 6.23f) donde ya no se puede distinguir apenas visualmente entre la función óptima y la aproximada. Hay que tener en cuenta que, si bien en las primeras configuraciones la variable primera es la más importante, cuando se quiere hacer un control fino es finalmente la variable segunda la que necesita más funciones de pertenencia. En el caso de un patrón que no sea aleatorio, existirán regiones de operación más importantes que

otros y el proceso se podría ver como una aproximación de funciones local, es decir, más exacta en aquellas regiones de operación por los que transcurre más a menudo el estado de la planta.

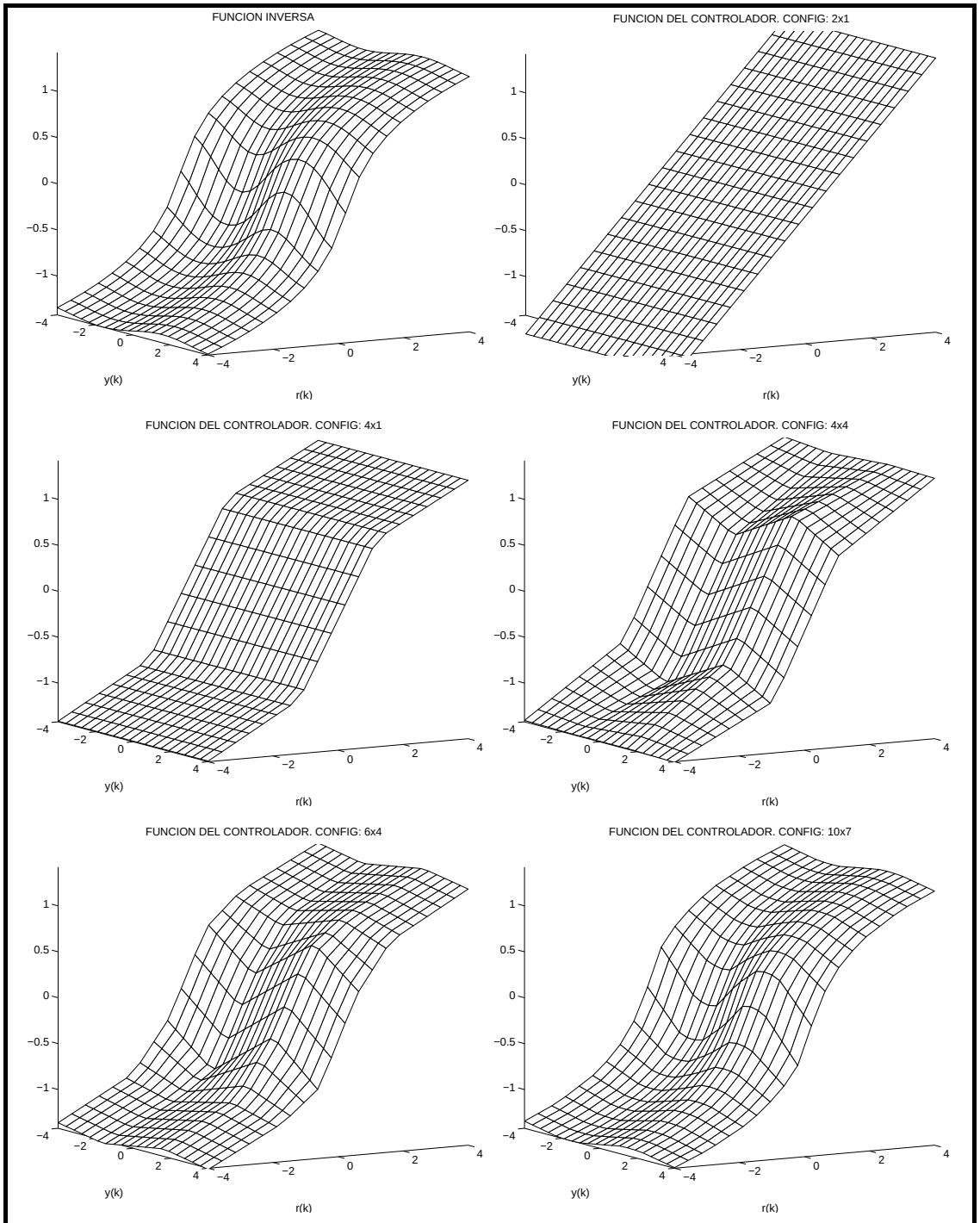


Fig. 6.23 Función implementada por el controlador a) Función inversa óptima b) configuración 2x1 c) configuración 4x1 d) configuración 4x4 e) configuración 6x4 f) configuración 10x7.

Config.	J_1	J_2	ECM*10^3
1x1	0.682	0.118	8040
2x1	0.200	0.113	540
3x1	0.169	0.109	400
4x1	0.0435	0.0869	60.1
4x2	0.0424	0.0857	55.1
4x3	0.0381	0.0714	45.9
4x4	0.0372	0.0141	21.3
5x4	0.0247	0.0126	11.1
6x4	0.0156	0.0101	4.09
7x4	0.0116	0.0117	2.14
8x5	0.0092	0.0108	1.54
9x6	0.0076	0.0070	0.651
10x7	0.0055	0.0060	0.415

Tabla 6.3 Evolución del algoritmo para las distintas configuraciones por las que transita (Π_6).

Finalmente, analicemos el comportamiento de esta etapa del algoritmo frente a la planta:

$$\Pi_5 \equiv y(k+1) = \frac{1.5\, y(k-1)\, y(k)}{1+y^2(k-1)+y^2(k)} + 0.35\, sen\big(y(k-1)+y(k)\big) + 1.2\, u(k) \qquad (6\text{-}25)$$

con consignas aleatorias entre -4 y 4. Para este rango de valores, la planta anterior se muestra tremendamente difícil de controlar de manera fina como se puede apreciar en la gráfica 6.24, en la que se ha proyectado la función de la planta para el punto *u(k)=0*.

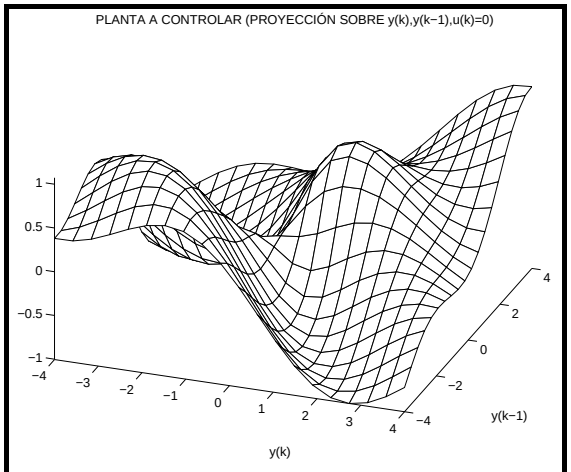


Fig. 6.24 Planta a controlar proyectada sobre u(k) = 0.

Esta vez, como variables de entrada seleccionaremos *r(k)*, *y(k)*, *y(k-1)* e *y(k-2)* todas ellas inicialmente con una sola función de pertenencia. En la tabla 6.4 figuran los distintos estados por lo que evoluciona el controlador así como los valores de J_1 a J_4 y el ECM (multiplicado por 1000) tras cada configuración. La primera decisión que se toma es utilizar la consigna como variable. Hay que tener en cuenta que aunque el controlador principal no utilice la consigna como variable de entrada, los sistemas auxiliares sí la utilizan para el proceso de adaptación por lo que el proceso de control se puede realizar incluso con una sola regla (el sistema se comportaría como un control adaptativo sin aprendizaje). Sin embargo, una vez que hay dos funciones de pertenencia en la primera variable, el

sistema comienza a añadir funciones de pertenencia en las variables intermedias y nunca en la cuarta variable. Esta evolución es realmente la óptima si nos fijamos en (6-25). Si despejamos la señal de control de esta ecuación comprobaremos que *u(k)* depende linealmente con *r(k)*, que no depende en absoluto de *y(k-2)* y que la dependencia con *y(k)* e *y(k-1)* es idéntica.

	r(k)	y(k)	y(k-1)	y(k-2)	
Config.	**J_1**	**J_2**	**J_3**	**J_4**	**ECM*1000 Tras etapa 2**
1x1x1x1	0.485	0.144	0.126	0.0154	6493
2x1x1x1	0.0320	0.201	0.194	0.0324	307.4
2x2x2x1	0.0233	0.105	0.105	0.0216	108.8
2x3x3x1	0.0148	0.0681	0.0687	0.0140	88.11
2x4x4x1	0.00975	0.0228	0.0235	0.00968	12.3
2x5x5x1	0.00816	0.137	0.0138	0.00862	5.17
2x6x6x1	0.00754	0.0956	0.0964	0.00787	1.74

Tabla 6.4 Evolución del algoritmo para las distintas configuraciones por las que transita (Π_5).

6.3 Análisis de otras características adicionales

Una vez mostrados varios ejemplos que ponen de manifiesto el funcionamiento de las partes principales del algoritmo, en esta sección abordaremos otras cuestiones secundarias que no por ello dejan de ser importantes. En primer lugar mostraremos ejemplos de cómo se comporta el algoritmo cuando no se le exige en todo momento un error cero, sino que existe la especificación de un error máximo permitido. Posteriormente veremos cómo se comporta ante plantas que pueden cambiar algunas de sus características con el tiempo y finalmente veremos que una partición triangular no tiene por qué ser la única configuración de funciones de pertenencia utilizada, sino que se pueden utilizar también funciones gaussianas, pseudo-gaussianas, etc.

6.3.1 Especificación del error máximo permitido

En la sección anterior se ha comprobado cómo el algoritmo es capaz de mejorar la política de control primero adaptando los parámetros del controlador y después modificando la topología del mismo de forma que se añadan funciones de pertenencia sólo en aquellas variables que influyan de forma significativa en el proceso de control. Teóricamente, el algoritmo no presenta ninguna condición de parada, lo cual es lógico ya que se pretende controlar la planta en tiempo real de forma indefinida y no se ha impuesto ningún tipo de condición por la que se debiera dejar de añadir más funciones

de pertenencia. Sin embargo, es común en la mayoría de los casos especificar una cota de error por debajo de la cual la planta se considera controlada de forma satisfactoria.

En la sección 5.8.2 se analizó y mostró cómo, con unos cambios sencillos, el algoritmo puede amoldarse de manera que dirija su objetivo principal a la reducción del error máximo producido en la planta. Así, se introdujo una función $D(x;\varepsilon)$ que definía una zona muerta dentro de la cual los parámetros del controlador no eran adaptados. Con la ayuda de estas zonas muertas, los esfuerzos de adaptación se dirigirán en la dirección de corregir los errores superiores al especificado en detrimento de ajustar de forma fina errores por debajo de éste.

La mejor forma de ver cómo tener en cuenta este factor es mediante un ejemplo. Consideremos, nuevamente:

$$\Pi_1 \equiv y(k+1) = 0.8\, sen(2\, y(k)) + 1.2\, u(k) \qquad (6\text{-}26)$$

y compararemos el comportamiento del algoritmo para el caso de un error máximo permitido de 0.1 y el caso en el que éste no venga especificado. Utilizaremos como variables de entrada $e(k)$ e $y(k)$, con una configuración de 2x8 funciones de pertenencia. La consigna será un patrón de 1000 iteraciones compuesto por escalones diversos en el rango [-4,4].

En la figura 6.25 se representa la evolución del error máximo para cada patrón (1000 iteraciones) para los dos casos que se comparan. Como se puede apreciar, cuando se especifica un error máximo permitido, el algoritmo trata de concentrar sus esfuerzos en aquellas zonas con mayores errores (en este caso los sobredisparos) de manera que finalmente consigue que el error máximo

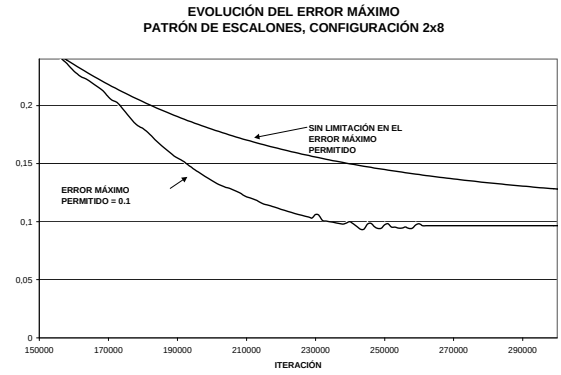
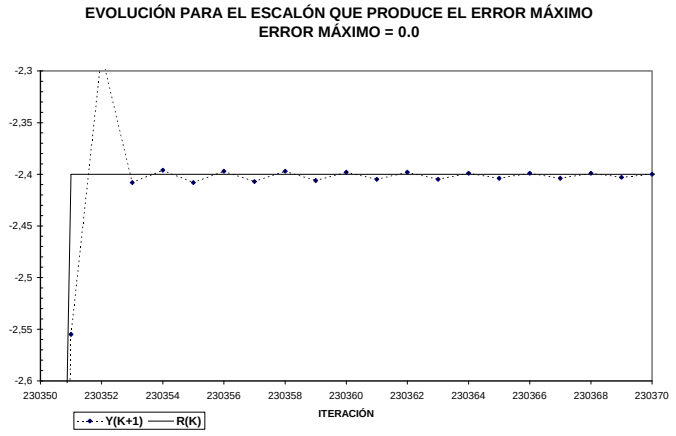


Fig. 6.25 Evolución del error máximo cada 1000 iteraciones con y sin la especificación de una cota de error máxima permitida.

quede por debajo de la cota especificada. En ese instante, los parámetros dejan de adaptarse (aunque el controlador ca_2 siempre está alerta para corregir un error superior) y el controlador principal actúa solo.

Gráficamente, se puede analizar lo que está pasando realmente en la planta observando las figuras 6.26a y b que corresponden al escalón donde se produce el error máximo en todo el patrón. En la figura 6.26a, no existe error máximo especificado por lo que el controlador intenta adaptar siempre los parámetros que, en este caso, no son los suficientes como para conseguir un control perfecto. Como se puede apreciar en la figura, tenemos un error máximo ligeramente superior a 0.15 en este caso. En la figura 6.26b se observa la evolución para el mismo instante anterior pero con la especificación de un error máximo de 0.1. En ella se observa que el controlador ha ido adaptando los parámetros sólo en las zonas necesarias hasta que ha conseguido que el error máximo



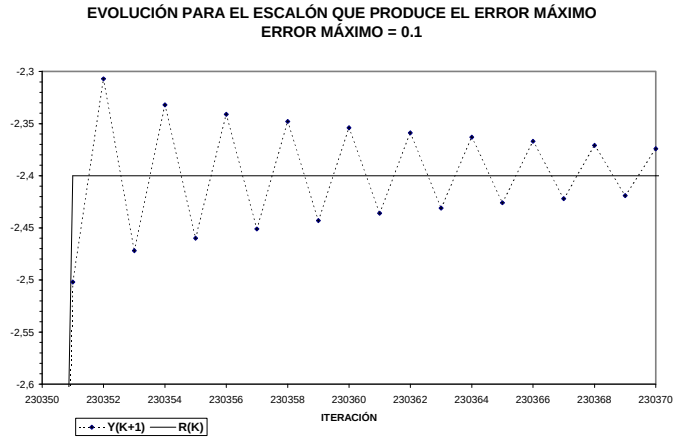


Fig. 6.26 Comportamiento de la variable a controlar en la zona del error máximo a) sin la especificación de una cota de error b) con un error máximo permitido de 0.1.

sea justamente 0.1. Como es obvio, esto lo hace a expensas de empeorar el comportamiento de otras zonas, aunque siempre se controlan por debajo del error especificado.

Finalmente, en el caso de especificar un error máximo, el algoritmo completo ya sí presenta una condición de convergencia final. Éste se produce cuando el error queda siempre por debajo de los límites establecidos. En ese caso, el algoritmo dejará de añadir funciones de pertenencia y la etapa 2 proseguirá indefinidamente.

6.3.2 Control de sistemas variables con el tiempo

Una característica intrínseca al algoritmo es que éste es capaz de controlar igualmente plantas cuyas características varíen con el tiempo. Es el caso de sistemas cuyo funcionamiento varía de alguna forma con la temperatura, humedad, roturas accidentales, etc. Esta característica es intrínseca al algoritmo ya que en ningún momento se ha necesitado un entrenamiento previo de los parámetros del controlador y éste actúa enteramente en tiempo real. Esto no quiere decir, como es obvio, que la planta pueda variar con el tiempo de una forma aleatoria (lo cual la haría incontrolable no por éste, sino por cualquier algoritmo posible) ni que de repente la monotonía de ésta cambie de signo.

Para poner un ejemplo de qué es lo que pasa cuando alguna característica de la planta varía en un cierto momento, consideremos de nuevo el sistema del péndulo invertido ya introducido en la sección 6.2.2.1. Al igual que hacíamos en dicha sección, utilizaremos como variables de entrada $e_\theta(k)$, $\theta(k)$, $\theta(k\text{-}1)$ y $u(k\text{-}1)$ con dos funciones de pertenencia para cada una de ellas y los mismos valores nominales para los distintos parámetros del sistema a controlar (m, m_c, l). El periodo de muestreo es de nuevo 0.1s. Como consignas, utilizaremos funciones senoidales de amplitud 0.5 radianes (29º) y de frecuencias aleatorias entre 0.02 y 50Hz.

En la simulación, dejaremos al proceso evolucionar para los valores nominales del péndulo y el carro e introduciremos los siguientes cambios alrededor de la iteración 30000:

- Disminución de la masa del carro de 1Kg a 0.5Kg (Figura 6.27a)
- Aumento de la masa del carro de 0.5Kg a 1Kg (Figura 6.27b)
- Disminución de la semilongitud del péndulo de 0.5m a 0.25m (Figura 6.27c)
- Aumento de la semilongitud del péndulo de 0.5m a 4m (Figura 6.27d)

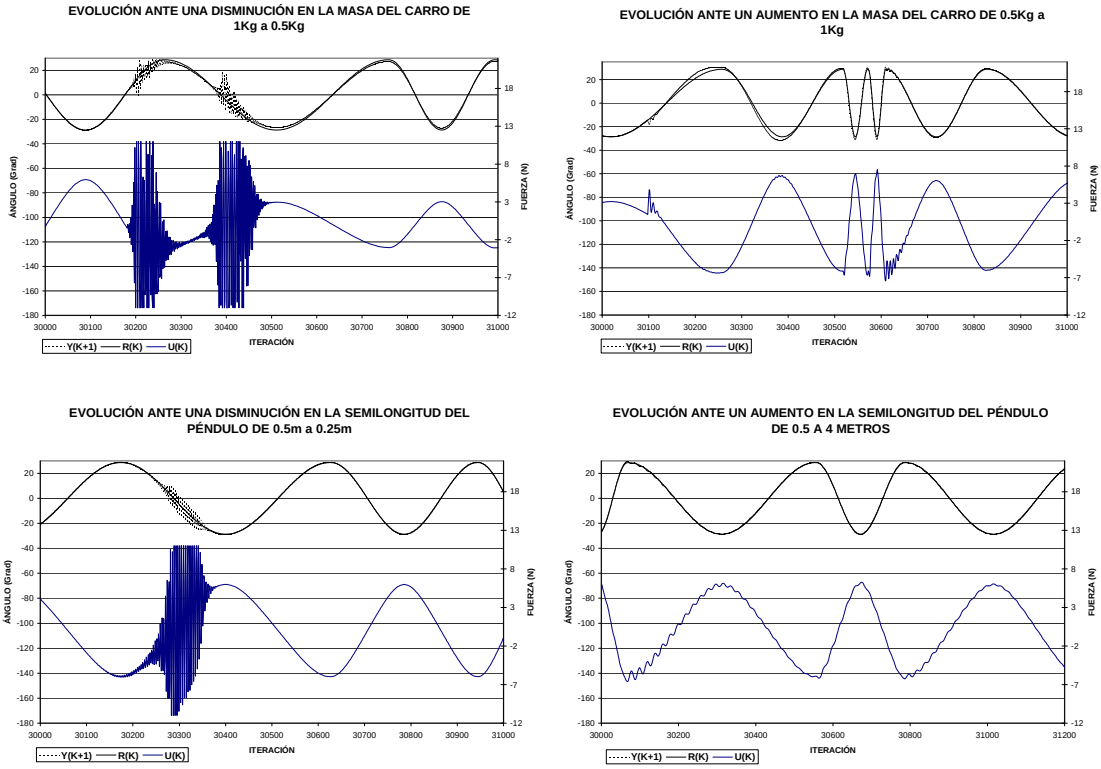


Fig. 6.27 Evolución del ángulo descrito con el péndulo ante cambios en las características del sistema a) masa del carro de 1 a 0.5 Kg. b) masa del carro de 0.5 a 1Kg c) semilongitud del péndulo de 0.5 a 0.25m d) semilongitud del péndulo de 0.5 a 4m.

En todas las figuras mostradas se ha representado la evolución angular del péndulo junto con los valores de consigna (primer eje de ordenadas) y la señal de control (segundo eje de ordenadas). En todos los casos, los cambios introducidos en los valores nominales de las plantas se traducen en pequeñas oscilaciones de la variable a controlar si bien, para el caso tanto de la disminución de la masa del carro como de la longitud del péndulo, los cambios provocan que el sistema se vuelva mucho más sensible y se aprecian grandes oscilaciones en la señal de control que, finalmente, se recuperan y el sistema vuelve a controlar de la misma forma que lo hacía antes, pero esta vez ya con los nuevos valores de los parámetros de la planta.

6.3.3 Utilización de otros tipos de funciones de pertenencia

En la sección 5.8.3 del capítulo anterior se mostró la forma de adaptar el algoritmo a otros tipos de funciones de pertenencia definidos en el Apéndice A. Esencialmente, las únicas modificaciones significativas se producían en la etapa de igualación del criterio de la integral del error cuadrático donde en el caso de utilizar funciones triangulares libres, éstas se consideraban como una partición triangular y el caso de funciones gaussianas y pseudo-gaussianas libres se trataba como si fuera una partición pseudo-gaussiana, todo de la misma forma que se hacía en el capítulo de aproximación de funciones para la etapa previa.

Para mostrar un ejemplo de cómo se comporta el algoritmo usando distintos tipos de funciones de pertenencia, consideremos el sistema regido por la siguiente ecuación en diferencias:

$$\Pi_7 \equiv y(k+2) = \frac{0.8 \cdot sen\left(2 \cdot \phi(y(k))\right)}{1+y^2(k)} + cos\left(y(k)/2\right) \cdot u(k) + u^3(k) \qquad (6\text{-}27)$$

Dicho sistema presenta un retardo de dos iteraciones y la monotonía del mismo es positiva, siempre y cuando *y(k)* se mantenga dentro del rango]-π,π[. La consigna estará compuesta por una función senoidal con un periodo de 25 iteraciones en el rango [-2,2].

En la figura 6.28 se observa cómo debe ser el control perfecto sobre la planta. En ella se ha representado *y(k+2)*, *r(k)* (eje principal de ordenadas) y la señal óptima de control *u(k)* (eje secundario). Como se puede comprobar, para obtener un control óptimo, la señal de control debe describir una función realmente complicada y ésta será la que deban implementar nuestros controladores al final del algoritmo.

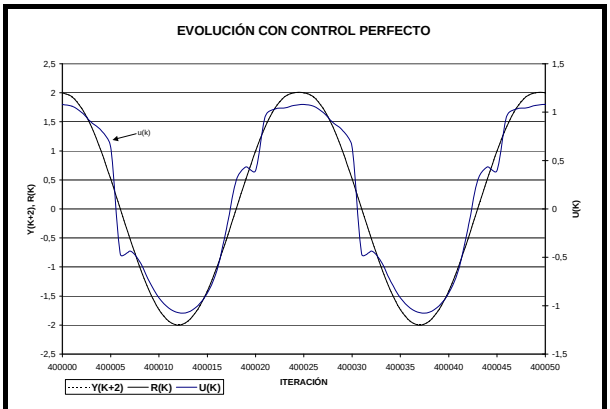


Fig. 6.28 Control perfecto de Π_7.

Las variables de entrada serán el error actual en la planta y la salida de la misma, comenzándose en todos los casos con una sola función de pertenencia en cada una de ellas. El algoritmo se ejecutará hasta que se dé la condición de que el error máximo

quede siempre por debajo de la cota de 0.001. En las figuras 6.29a y b se representa la evolución del ECM y del valor absoluto del error máximo para cada uno de los tipos de funciones recogidos en el Apéndice A. En las tablas 6.5 a 6.9 se presentan los valores numéricos significativos de las figuras anteriores junto con las configuraciones de funciones de pertenencia por las que atraviesa el controlador principal.

Iteración al conmutar de Nº de mfs	ECM·10^3	% medio de actuaciones del supervisor	Error máximo	Configuración
2000	1325.1	96 %	1.99	1x1
8600	380.3	73 %	1.40	2x2
33200	55.3	33 %	0.56	2x3
51400	36.5	23 %	0.46	2x4
76800	3.36	2 %	0.16	2x5
111200	0.72	0	0.061	2x6
151200	0.52	0	0.060	2x7
278000	0.015	0	0.012	2x8
329400	0.00095	0	0.0025	3x9
416800	0.000093	0	0.00086	4x9

Tabla 6.5 Evolución del algoritmo para controlar la planta π_7 usando una partición triangular.

Iteración al conmutar de Nº de mfs	ECM·10^3	% medio de actuaciones del supervisor	Error máximo	Configuración
2000	1325.1	96 %	1.99	1x1
19800	116.1	56 %	0.89	2x2
34800	38.6	30 %	0.50	2x3
67400	4.53	5 %	0.18	2x4
102200	2.68	0	0.15	2x5
147600	0.51	0	0.060	2x6
191200	0.42	0	0.052	2x7
322000	0.012	0	0.011	2x8
369400	0.00091	0	0.0022	3x9
456800	0.000080	0	0.00080	4x9

Tabla 6.6 Evolución del algoritmo para controlar la planta π_7 usando funciones triangulares libres.

Iteración al conmutar de Nº de mfs	ECM·10^3	% medio de actuaciones del supervisor	Error máximo	Configuración
2000	1325.1	96 %	1.99	1x1
8600	57.8	29 %	0.73	2x2
28800	51.2	31 %	0.70	2x3
51800	10.2	17 %	0.33	2x4
69800	1.28	2 %	0.13	2x5
112000	0.40	0	0.052	2x6
155800	0.13	0	0.030	2x7
212800	0.084	0	0.044	2x8
294400	0.000001	0	0.00060	2x9

Tabla 6.7 Evolución del algoritmo para controlar la planta π_7 usando una partición pseudo-gaussiana.

Iteración al conmutar de Nº de mfs	ECM·10^3	% medio de actuaciones del supervisor	Error máximo	Configuración
2000	1325.1	96 %	1.99	1x1
15200	22.8	18 %	0.38	2x2
42800	10.9	17 %	0.24	2x3
69800	4.42	6 %	0.20	2x4
95000	0.56	0	0.097	2x5
148000	0.15	0	0.032	2x6
196000	0.049	0	0.015	2x7
282000	0.0040	0	0.00097	2x8

Tabla 6.8 Evolución del algoritmo para controlar la planta π_7 usando funciones gaussianas.

Iteración al conmutar de Nº de mfs	ECM·10^3	% medio de actuaciones del supervisor	Error máximo	Configuración
2000	1325.1	96 %	1.99	1x1
15200	22.8	18 %	0.38	2x2
42800	9.87	17 %	0.24	2x3
61800	3.23	4 %	0.13	2x4
95000	0.40	0	0.052	2x5
148000	0.13	0	0.031	2x6
236000	0.042	0	0.015	2x7
312000	0.0004	0	0.0008	2x8

Tabla 6.9 Evolución del algoritmo para controlar la planta π_7 usando funciones pseudo-gaussianas libres.

En la figura 6.29a se observa que son las funciones gaussianas las que primero alcanzan la cota máxima de error permitida, seguidas muy de cerca por la partición pseudo-gaussiana. Este hecho se debe a que, aunque las funciones gaussianas tengan un parámetro más que optimizar que en el caso de una partición pseudo-gaussiana, la configuración en la que se alcanza el valor límite es la de 2x8 a diferencia del segundo caso en el que se debe recurrir a la configuración 2x9. En el caso de las funciones

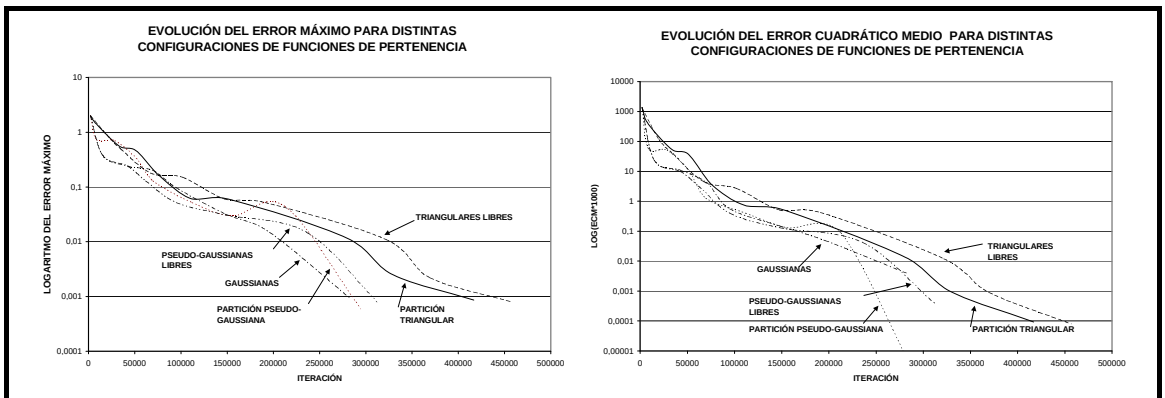


Fig. 6.29 Evolución del algoritmo con distintos tipos de funciones de pertenencia a) error máximo b) ECM.

pseudo-gaussianas libres, también basta con una configuración 2x8 pero el algoritmo requiere más iteraciones para que se verifique la condición de convergencia en cada etapa debido a la existencia de un mayor número de parámetros a optimizar.

En el caso de funciones triangulares libres y de la partición triangular, el algoritmo requiere mayor número de iteraciones ya que, a partir de la configuración 2x8, la primera variable vuelve a entrar en juego y, por tanto, el número de configuraciones por las que atraviesa el controlador principal son mayores. Sin embargo, en el caso de todas las funciones basadas en gaussianas han bastado dos funciones de pertenencia tan solo para la primera variable.

Por supuesto, para otro tipos de sistemas a controlar las evoluciones pueden ser muy diversas y en absoluto se puede concluir que una distribución sea mejor que la otra.

6.4 Comparación con otros métodos propuestos en la bibliografía

Finalmente, en esta sección se realizará una comparación entre los resultados obtenidos por el algoritmo aquí presentado con los de otras metodologías propuestas en la bibliografía. En el campo de control en tiempo real, es realmente difícil realizar comparaciones entre distintas metodologías ya que los autores no suelen dar índices de error en sus ejemplos sino que se suele presentar simplemente la evolución de los algoritmos y generalmente sólo para el caso de una sola función escalón.

Ya en la sección 6.2.2.1 se realizó una breve comparación de los resultados obtenidos para el caso de un péndulo invertido con los obtenidos por L-X. Wang en [WAN-94]. En este apartado recogeremos los resultados obtenidos por H.C.Andersen [AND-98] donde se comparaba el algoritmo de control en tiempo real que él proponía con las principales metodologías existentes.

De entre los algoritmos con los que se realizó la comparación, los métodos indirectos basados en un modelo de la planta “Indirect Output Matching (**IOM**)” [NGU-90, JOR-90, WER-90, TAN-92], el método de linealización en lazo cerrado “Feedback

Linearization (**FL**)" [CHE-95] y el método basado en el error en la salida del controlador "Controller Output Error Method (**COEM**)" [AND-97] resultaron ser muy superiores a otros controladores lineales y a controladores PI, PD y PID. Esos serán los tres métodos con los que compararemos en esta sección.

• El método **IOM** se basa en un modelo de la planta a controlar para adaptar el controlador principal, de modo que necesitan un conjunto adicional de parámetros a optimizar para obtener dicho modelo en tiempo real. Si el emulador tiene la forma:

$$y(k+1) \approx \hat{f}\,[y(k), \ldots, y(k-m+1), u(k), \ldots, u(k-n+1)] \tag{6-28}$$

y dicha aproximación es relativamente buena, entonces el algoritmo ya es capaz de obtener el valor de la derivada parcial de la salida de la planta con respecto a la señal de control para poder realizar un método basado en el gradiente.

• En el caso del método **FL**, se hace la suposición de que el modelo de la planta se puede ajustar al patrón:

$$\begin{aligned} y(k+1) = {} & \hat{f}_1[y(k),\ldots,y(k-m),u(k-1),\ldots,u(k-n)] + \\ & u(k)\cdot\hat{f}_2[y(k),\ldots,y(k-m),u(k-1),\ldots,u(k-n)] \end{aligned} \tag{6-29}$$

de modo que la señal de control se podría calcular de la forma:

$$u(k) = \frac{r(k) - \hat{f}_1[y(k),\ldots,y(k-m),u(k-1),\ldots,u(k-n)]}{\hat{f}_2[y(k),\ldots,y(k-m),u(k-1),\ldots,u(k-n)]} \tag{6-30}$$

En este caso también se requiere un modelo de la planta pero, a diferencia del método anterior, el controlador principal no tiene ningún parámetro que optimizar ya que la salida de control se calcula directamente de (6-30). Sin embargo, este método no necesita la mitad de parámetros que en el IOM ya que son dos las funciones ($\hat{f}_1$ y $\hat{f}_2$) que se necesita aproximar para el modelado del proceso. Este algoritmo funciona muy bien para el caso de plantas que se ajusten al modelo definido en (6-29). En el caso en que esto no sea así, las prestaciones se empobrecen, como veremos más adelante.

• Finalmente, el **COEM** es un método de control directo que no necesita ningún modelo de la planta. Dicho método, en el que está basado el sistema auxiliar ca_2 de este capítulo, se basa en la adaptación de los parámetros del controlador a partir del error en la salida del controlador utilizando datos reales de la función inversa de la planta que, como sabemos, no tienen por qué pertenecer a la región de operación que realmente interesa en el momento actual. Por ello, dicho método requiere que la planta ya esté controlada de antemano para poder funcionar.

Las condiciones de las simulaciones realizadas en [AND-98], y que aquí utilizaremos, para los tres métodos con los que vamos a comparar el algoritmo propuesto son:

1. **Plantas utilizadas para las comparaciones**:

$$Planta\ 1 \equiv y(k+1) = 0.9 \cdot y(k) + \phi(u(k))$$

$$Planta\ 2 \equiv y(k+1) = 0.8 \cdot sen(2 \cdot y(k)) + 1.2 \cdot u(k)$$

$$Planta\ 3 \equiv y(k+1) = \frac{1.5 \cdot y(k-1) \cdot y(k)}{1 + y^2(k-1) + y^2(k)} + 0.35 \cdot sen\left[y(k-1) + y(k)\right] + 1.2 \cdot u(k)$$

$$Planta\ 4 \equiv y(k+1) = -0.3 \cdot sin(y(k)) + u(k) + u^3(k)$$

2. **Número de parámetros**: Los tres métodos utilizan redes neuronales para realizar las distintas tareas. Cada red neuronal está compuesta por una capa oculta de 5 neuronas utilizando la función tangente hiperbólica como función de activación y una neurona de salida lineal. Contando los términos de orden cero (bias), cada red tiene un total de 21 parámetros para el caso bidimensional y 26 en el caso de tres variables de entrada. Se supone que se conocen de antemano las entradas que realmente pueden influir en el proceso de control (*r(k)* e *y(k)* para todas las plantas excepto para la planta 3 donde también se utiliza *y(k-1)*). Los pesos son inicializados aleatoriamente en el rango [-1,1].

3. **Pre-entrenamiento**: Para cada una de las plantas testeadas se obtienen 1000 muestras aleatorias de la misma. Los rangos de las señales de control se encuentran entre -1 y 1 para todas las plantas excepto para la número 3 cuyo rango es [-2.2,2.2]. Estos rangos han sido elegidos empíricamente para excitar la planta suficientemente.

Con dichas muestras, todos los controladores son entrenados *off-line*, es decir, como si de una aproximación funcional se tratase, durante 2000 iteraciones.

4. **Consignas**: Se utilizarán dos tipos de consignas para comparar los resultados. En primer lugar se utilizarán consignas aleatorias uniformemente distribuidas en el rango [-1.5,1.5]. Como segunda comparación se usará un patrón, en dicho rango, compuesto por una función senoidal seguida de un escalón doble con un periodo total de 100 iteraciones.

5. **Resultados**: Los sistemas se evaluarán durante las primeras 20000 iteraciones, registrando el valor absoluto medio del error entre la salida *y(k+1)* y la consigna *r(k)* cada 1000 iteraciones.

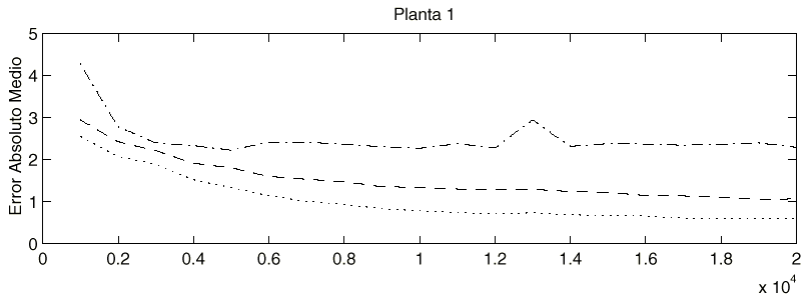
En cuanto a las condiciones impuestas al algoritmo que aquí se presenta, partiremos de un controlador principal con un número de parámetros similar a los utilizados en los casos anteriores. No habrá ningún entrenamiento previo del controlador, por lo que las funciones de pertenencia se equidistribuirán inicialmente y los consecuentes de las reglas serán inicializados al valor 0. El valor del periodo global de control será T'=200 para las señales aleatorias y T'=100 para el patrón de 100 muestras. En las representaciones gráficas se mostrará el error absoluto medio junto con el número medio de veces en el que actúa el sistema supervisor, ambos cada T' muestras. A partir de la iteración 20000 se obligará al algoritmo a cambiar la topología del sistema para ver la evolución del nuevo controlador durante otras 20000 iteraciones.

La convención que usaremos para las representaciones gráficas que siguen será:

- IOM: Líneas de puntos (· · ·)
- FL: Líneas de puntos y rayas (– · –)
- COEM: Líneas de rayas (– – –)

En la figura 6.30a se muestran los resultados obtenidos por los tres algoritmos anteriores para la Planta 1, con consignas aleatorias. En este caso, es el método IOM el

que consigue un mejor control obteniendo un error absoluto medio (EAM) alrededor de 0.6, muy por debajo del obtenido por el FL que, como ya se indicó antes, empobrece mucho su eficiencia ante plantas que no responden al patrón (6-29), como es este caso. En la figura 6.30b se muestran los resultados obtenidos por el algoritmo comenzando por una configuración 4x4 (es decir, optimizando 16 reglas + 2 + 2 centros móviles = 20 parámetros). En dicha figura se comprueba cómo el algoritmo propuesto supera con creces los resultados obtenidos por los métodos anteriores para esta configuración (un EAM casi 10 veces menor) mientras que para la configuración 5x5 (donde ya se aprovecha de los parámetros optimizados de la etapa anterior) consigue rápidamente reducir el error hasta la mitad. Como se aprecia en la figura, el sistema supervisor tiene una influencia escasa en el proceso, actuando siempre por debajo del 8% de las ocasiones.



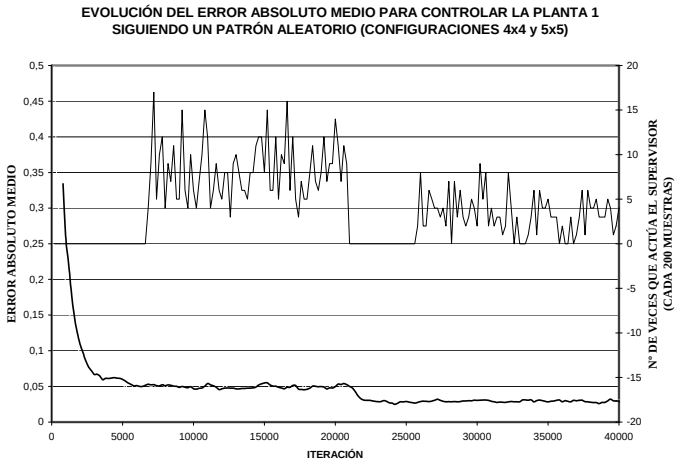


Fig. 6.30 Comparación del algoritmo con otras metodologías: Planta 1, consignas aleatorias a) IOM (···), FL(– · –), COEM (– – –) b) algoritmo propuesto.

Para el caso del patrón de 100 iteraciones, las gráficas comparativas se muestran en las figuras 6.31a y 6.31b. En este caso, los resultados obtenidos por el IOM, el COEM y el algoritmo propuesto son bastante parecidos aunque en este último caso se observa un

resultado mejor. Además, el método propuesto presenta la ventaja de que siempre se puede modificar la topología del algoritmo para mejorar las prestaciones, como se observa cuando el algoritmo conmuta a la configuración 4x5 (25 parámetros).

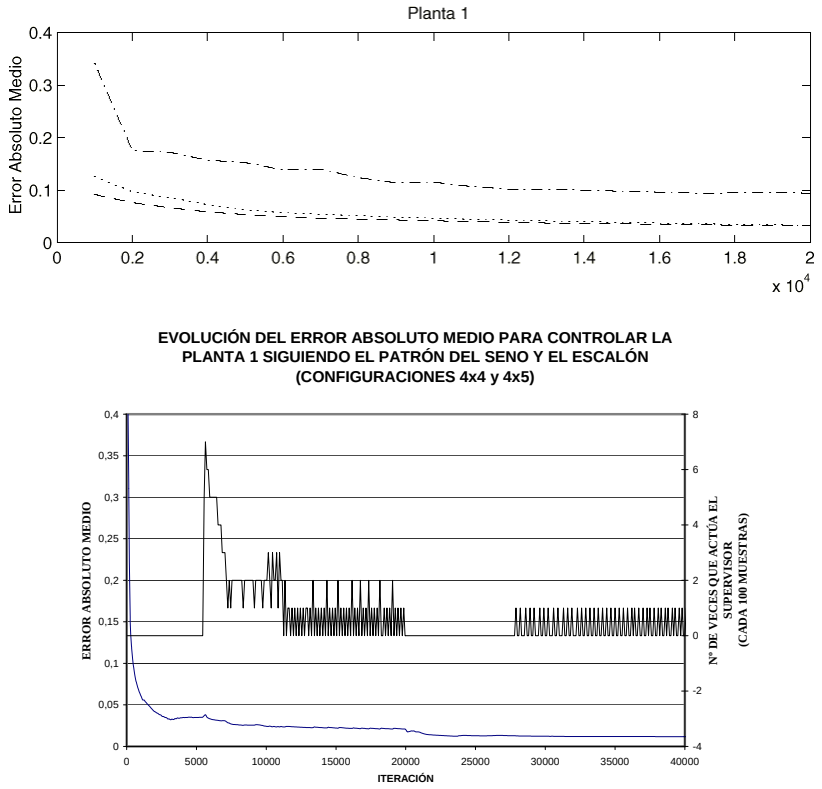


Fig. 6.31 Comparación del algoritmo con otras metodologías: Planta 1, patrón 2 a) IOM (···), FL(– · –), COEM (– – –) b) algoritmo propuesto.

En el caso de la Planta 2, los resultados son muy diferentes a los anteriores. En esta caso, la planta ya sí responde al patrón de sistemas para los que el método FL va dirigido y eso se comprueba fácilmente en las figuras 6.32a y 6.33a. El mejor EAM que se consigue en este caso es aproximadamente 0.3 para consignas aleatorias y 0.03 para el patrón del seno y el escalón doble. Sin embargo, en este caso, el algoritmo propuesto, cuando se inicia desde 1x1, la configuración que encuentra con un número de parámetros similar al usado por las demás metodologías es la de 2x7 (19 parámetros) y 2x8 (22 parámetros) lo cual representa una gran ventaja, como se comprueba en las figuras 6.32b y 6.33b, donde el EAM es muy significativamente superior al obtenido por el FL. En este caso, la intervención del sistema supervisor no ha sido necesaria.

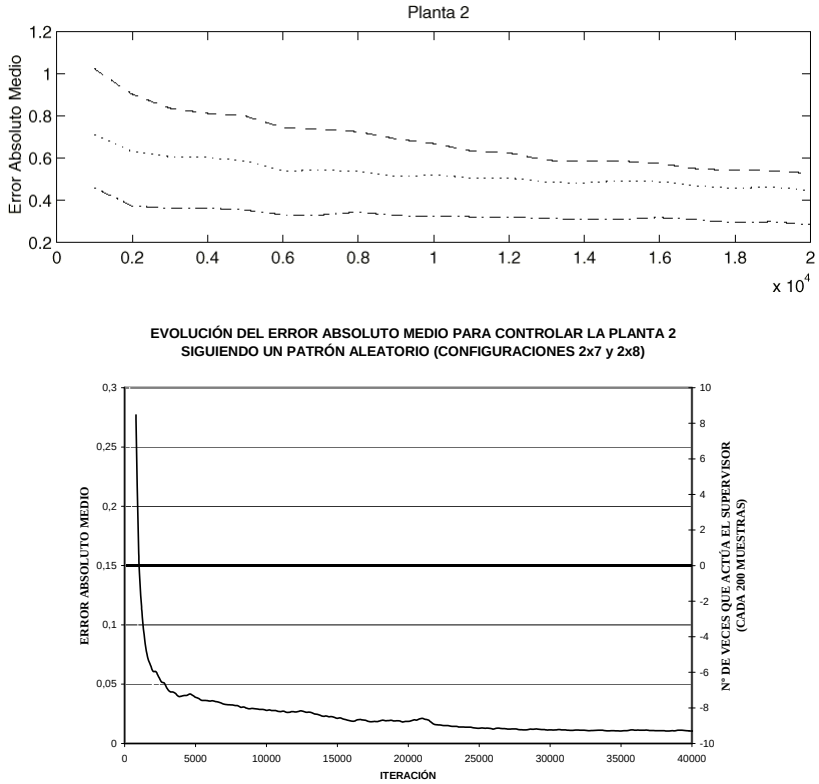


Fig. 6.32 Comparación del algoritmo con otras metodologías: Planta 2, consignas aleatorias a) IOM (···), FL(– · –), COEM (– – –) b) algoritmo propuesto.

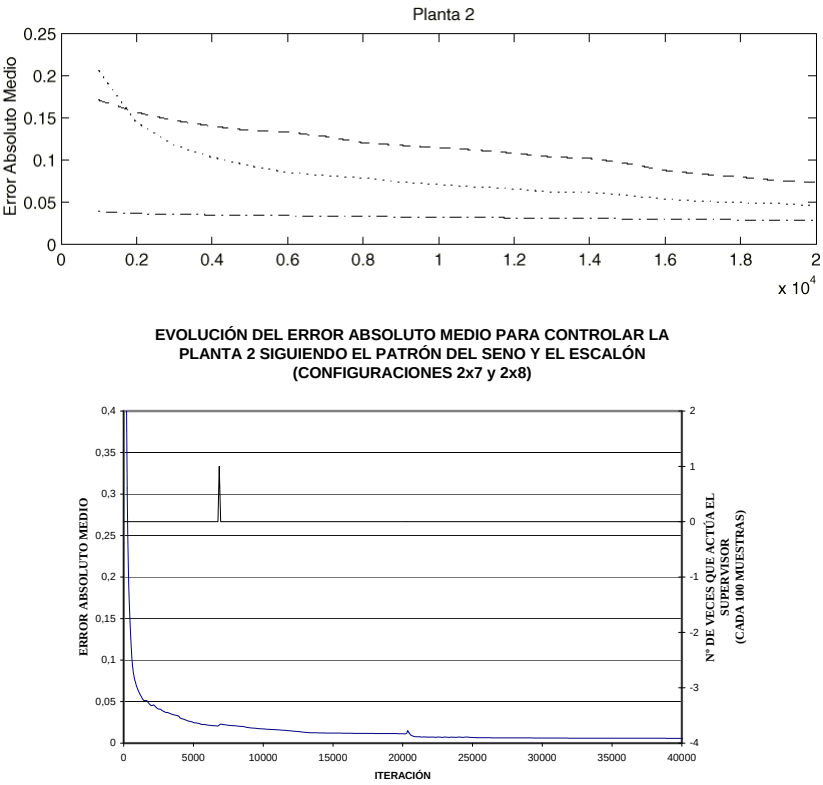


Fig. 6.33 Comparación del algoritmo con otras metodologías: Planta 2, patrón 2 a) IOM (···), FL(– · –), COEM (– – –) b) algoritmo propuesto.

Para la planta nº 3, es de nuevo el FL el que obtiene mejores resultados por la misma razón que en el caso anterior. Los mejores errores absolutos medios corresponden en este caso a 0.3 para consignas aleatorias y 0.016 para el patrón con el seno y el escalón doble (figuras 6.34a y 6.35a). En este caso, el controlador principal utiliza tres variables de entrada por lo que las configuraciones que obtiene ahora el algoritmo para el rango de parámetros que usan los métodos anteriores (26 parámetros) son 2x3x3 y 2x4x4. En el caso de consignas aleatorias (figura 6.34b) el algoritmo propuesto vuelve a obtener resultados claramente superiores mientras que para el segundo patrón (figura 6.35b), la configuración 2x3x3 (20 parámetros) consigue una política de control un poco inferior a la del resto de metodologías. Sin embargo, cuando el algoritmo cambia a la configuración 2x4x4 (36 parámetros) el EAM se consigue reducir considerablemente. También se puede observar en esta gráfica la gran influencia del sistema supervisor para este caso, actuando del orden del 50 % de las veces.

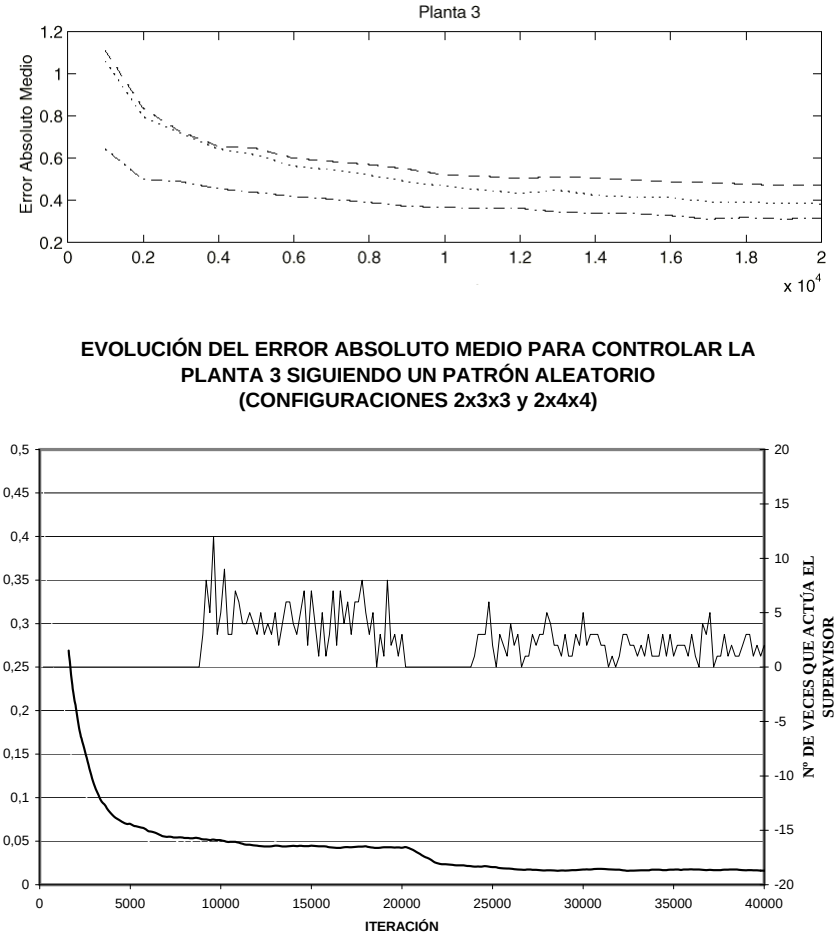


Fig. 6.34 Comparación del algoritmo con otras metodologías: Planta 3, consignas aleatorias a) IOM (···), FL(– · –), COEM (– – –) b) algoritmo propuesto.

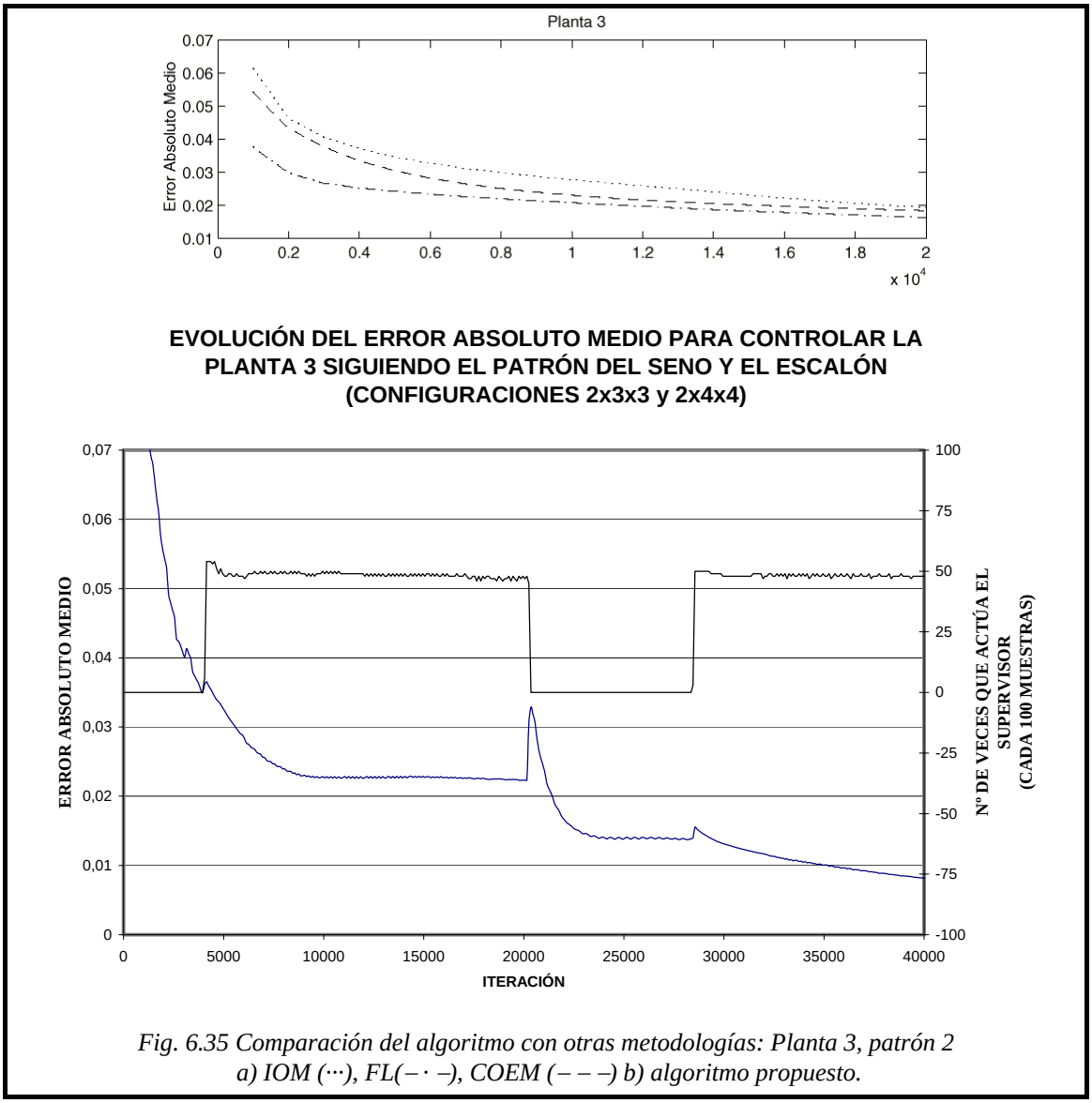


Fig. 6.35 Comparación del algoritmo con otras metodologías: Planta 3, patrón 2 a) IOM (···), FL(– · –), COEM (– – –) b) algoritmo propuesto.

Finalmente, para la Planta nº 4, vuelve a ser el IOM el que consigue los mejores resultados de entre los 3 algoritmos con los que comparamos. En este caso, la dependencia de la salida de la planta dista mucho de depender linealmente con la señal de control por lo que el método FL no obtiene buenos resultados para este caso. Nuevamente, el algoritmo propuesto supera claramente a los métodos anteriores para ambos tipos de consignas (figuras 6.36a-b y figuras 6.37a-b).

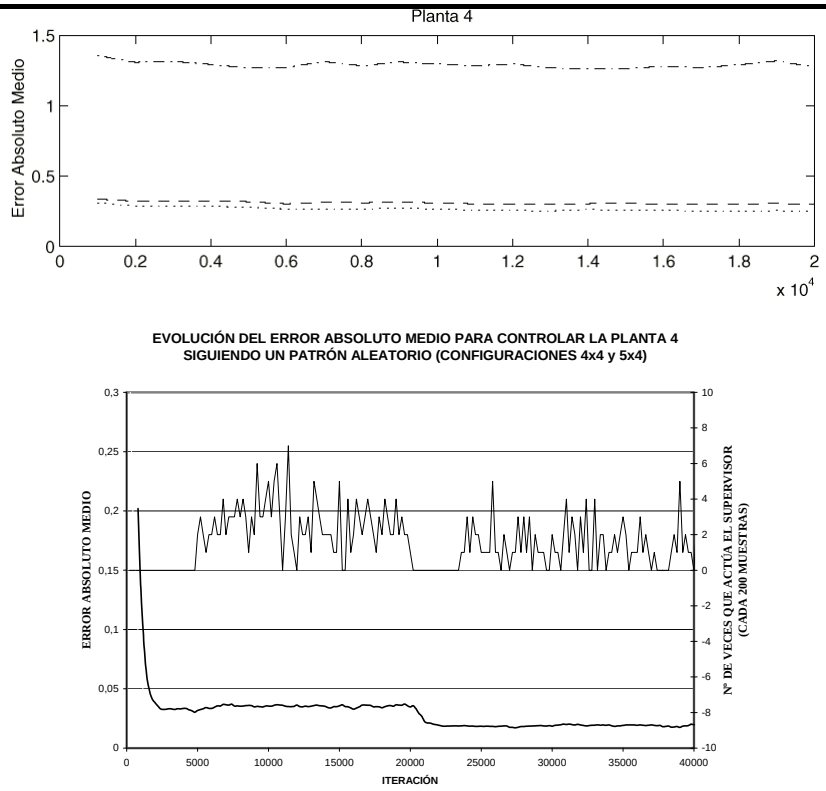


Fig. 6.36 Comparación del algoritmo con otras metodologías: Planta 4, consignas aleatorias a) IOM (···), FL(– · –), COEM (– – –) b) algoritmo propuesto.

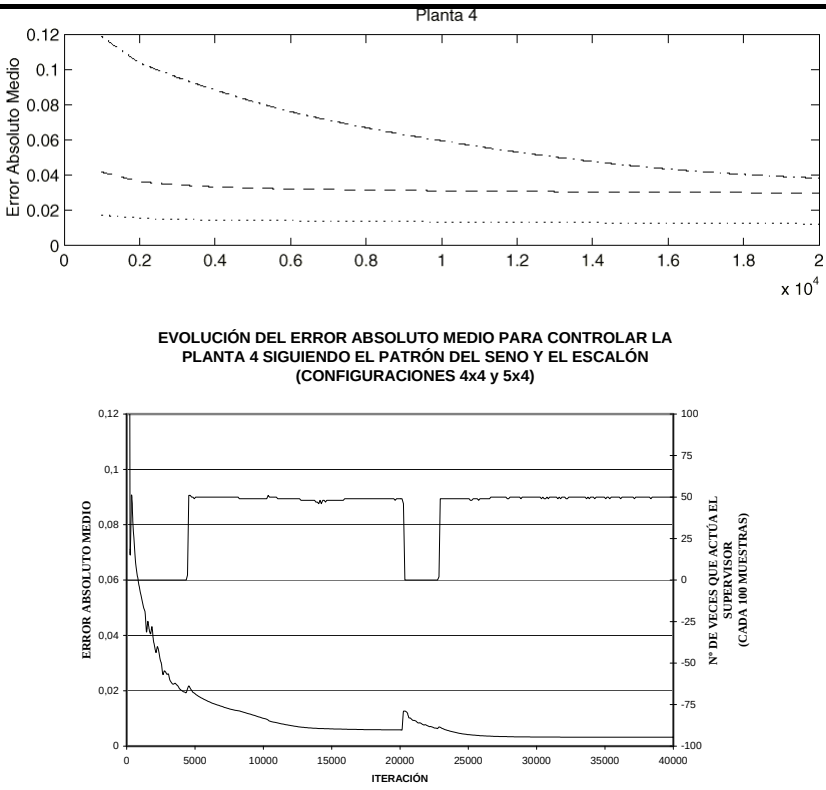


Fig. 6.37 Comparación del algoritmo con otras metodologías: Planta 4, patrón 2 a) IOM (···), FL(– · –), COEM (– – –) b) algoritmo propuesto.

6.5 Conclusión

En este capítulo se han presentado diversos ejemplos cuyo propósito ha sido analizar detalladamente el comportamiento del controlador difuso adaptativo y auto-organizativo que se propuso en el capítulo anterior.

En primer lugar, en la sección 6.2 se han estudiado las principales características del proceso de adaptación y aprendizaje en tiempo real, añadiendo en cada sección un componente nuevo al sistema de control y analizando sus propiedades y ventajas. Más concretamente:

- Se ha evaluado el funcionamiento del sistema auxiliar ca_1 durante la primera etapa del algoritmo, analizando cómo realiza el proceso de adaptación de las reglas difusas a partir del error en la salida de la planta y justificando las distintas características que posee.

- Se ha comprobado la validez y ventajas de la adición de un segundo sistema durante la etapa 1, encargado de buscar una nueva posición de los centros de las funciones de pertenencia de modo que se intente homogeneizar la distribución de la contribución a la integral del error cuadrático (IEC) de las distintas zonas delimitadas por éstas.

- Se ha analizado el comportamiento del sistema auxiliar ca_2 durante la segunda etapa del algoritmo. Este sistema se encarga de sintonizar los parámetros del controlador principal (reglas y funciones de pertenencia) utilizando como información el error en la salida del controlador.

- Igualmente se ha demostrado la necesidad de la existencia de un sistema que supervise el proceso durante la etapa 2, de modo que la señal a controlar evolucione siempre en la dirección en la que disminuya el error en la salida de la planta y se han mostrado ejemplos de su funcionamiento.

- Por último se han presentado ejemplos que ponen de manifiesto cómo el algoritmo es capaz de modificar de forma eficiente la topología del controlador principal para conseguir mejorar la política de control.

Posteriormente, en la sección 6.3 se ha comprobado el funcionamiento del algoritmo cuando:

1. Se especifica una cota superior en el error en la salida de la planta.
2. La planta a controlar presenta cambios imprevistos en su estructura interna en un momento determinado del proceso de control.

Asimismo, se ha mostrado la evolución del algoritmo utilizando diversos tipos de funciones de pertenencia definidos en el Apéndice A.

Finalmente, en la sección 6.4 se ha comparado el rendimiento del algoritmo propuesto con el obtenido por otras metodologías para distintos sistemas a controlar. En dicha sección se ha comprobado que, a pesar de que en el resto de métodos se realiza (y se requiere) un entrenamiento previo para conseguir unos valores iniciales adecuados para los parámetros del controlador, el algoritmo propuesto mejora sensiblemente los resultados obtenidos por éstos para una topología con un número de parámetros equivalente y partiendo de un conocimiento inicial de la planta prácticamente nulo.

CAPÍTULO 7

Conclusiones y principales aportaciones

El trabajo de tesis doctoral presentado en esta memoria es una contribución al diseño automático de sistemas difusos aplicados al campo de la aproximación funcional y al campo de control en tiempo real. Las principales aportaciones y conclusiones obtenidas quedan resumidas a continuación.

En la primera parte de este trabajo, se ha propuesto una nueva y completa metodología para el diseño automático de sistemas difusos a partir de la información proporcionada por un conjunto de vectores de E/S. Sin la existencia de información previa y a través del análisis de dichos vectores, el algoritmo va construyendo y optimizando sistemas difusos completos resolviendo tanto el problema de la identificación de la estructura del sistema como el ajuste de sus parámetros.

La metodología presentada a este efecto consta, esencialmente, de cuatro etapas: en la primera, se calculan los consecuentes óptimos de las reglas para una configuración fija de funciones de pertenencia. Posteriormente se realiza una determinación conjunta de los valores centrales de las funciones de pertenencia en la entrada y de los consecuentes de las reglas generadas. Dicho proceso de optimización se ha desglosado en dos fases: En la primera, se busca un “buen” punto de partida donde se deben encontrar los

mejores mínimos locales. En la segunda empleamos un algoritmo de optimización basado en el gradiente para hallar el primer mínimo local que se encuentre cerca del punto de partida.

Una vez optimizada la configuración actual, se analiza la superficie del error cometido para determinar las variables donde se ha de aumentar el número de funciones de pertenencia que particionan sus dominios, de modo que mejoremos el error cuadrático medio normalizado *(ECMN)* de forma óptima. De esta forma, el algoritmo es capaz de hallar automáticamente el número de funciones de pertenencia que se deben utilizar para cada variable de entrada y, como caso particular, se identifican las entradas que son realmente necesarias para la consecución de un cierto grado de aproximación.

Finalmente, de entre todas las configuraciones obtenidas, se evalúa el compromiso entre complejidad y exactitud para cada una de ellas seleccionando aquélla con mejor índice de complejidad/exactitud *ICE*, viniendo éste definido en forma de reglas difusas que proporciona el usuario final de la aplicación.

Como características adicionales, se han introducido pequeñas modificaciones y mejoras en el algoritmo principal para incrementar su versatilidad. En concreto, se han realizado modificaciones para poder analizar conjuntos de datos de E/S que no exploren completamente el espacio de entrada, para adaptar el algoritmo a otras configuraciones de funciones de pertenencia y conseguir que las reglas finales obtenidas mantengan su interpretabilidad.

En el capítulo 4 se han mostrado diversos ejemplos que justifican y demuestran la validez de cada una de las partes que componen el algoritmo de aproximación funcional y verifican su robustez frente a datos con diferentes niveles de ruido. Asimismo, se han comparado las prestaciones del algoritmo tanto a nivel de grado de aproximación como de complejidad con respecto a otros algoritmos de extracción de sistemas difusos a partir de datos de E/S y a otros paradigmas también comúnmente utilizados para resolver el problema de aproximación funcional. En dichas comparaciones, el algoritmo propuesto siempre se muestra capaz de obtener configuraciones con mayor grado de

aproximación y menor complejidad estructural que las proporcionadas en la bibliografía.

En la segunda parte de esta memoria se ha propuesto una nueva y completa metodología para el diseño automático de controladores difusos directos en tiempo real. Desde una estructura prácticamente vacía y a partir de una información cualitativa reducida sobre el sistema a controlar (concretamente el signo de la monotonía de la planta, su retraso y una estimación de las variables de entrada que pueden influir en su comportamiento), el algoritmo analiza la evolución del proceso y va auto-diseñando el controlador difuso principal mientras éste opera en tiempo real, abordándose, al igual que la metodología anterior, tanto el problema de la identificación de la estructura del controlador como el ajuste de sus parámetros.

La metodología presentada consta de tres etapas: en la primera de ellas, se adaptan los consecuentes de las reglas utilizando información directa del error en la salida de la planta. Durante este proceso y de forma alternada, se modifican los centros de las funciones de pertenencia intentando igualar la contribución de cada zona delimitada por las mismas a la integral del error cuadrático.

Posteriormente se realiza un ajuste fino tanto de los parámetros que definen las funciones de pertenencia en la entrada como de los consecuentes de las reglas generadas a través de un método basado en el gradiente. Debido a que dicho método se basa en el error en la salida del controlador y no de la planta, se ha justificado la necesidad de un sistema supervisor que asegure que el proceso evolucione siempre de forma que se aproxime el valor de salida actual de la planta al valor de consigna.

Tras la segunda etapa, se realiza un análisis global del sistema a controlar utilizando como información los últimos datos recopilados de la verdadera función inversa de la planta. Dicho análisis nos permitirá determinar, de la misma forma que se hizo en el caso de aproximación funcional, las variables en las que se ha de aumentar el número de funciones de pertenencia que particionan sus dominios de modo que se mejore la política de control. Como caso particular, se resuelve el problema de identificar cuáles

de las entradas proporcionadas son realmente necesarias para el cumplimiento de las especificaciones del problema.

Finalmente, se han introducido una serie de mejoras en el algoritmo para tener en cuenta un rango limitado en el actuador, la existencia de una especificación sobre el error máximo permitido y para adaptarlo a otros tipos de funciones de pertenencia distintos de una partición triangular.

En el último capitulo se han presentado diversos ejemplos que analizan detalladamente el comportamiento del controlador difuso adaptativo y auto-organizativo propuesto, justificando de forma razonada y verificando cada una de las características de la metodología propuesta. Asimismo, se han comparado las prestaciones del sistema de control con otras metodologías existentes en la bibliografía. En dichas comparaciones se ha comprobado que, a pesar de que en el resto de métodos se realiza (y se requiere) un entrenamiento previo para conseguir unos valores iniciales adecuados para los parámetros del controlador, el algoritmo propuesto mejora sensiblemente los resultados obtenidos por éstos para una topología con un número de parámetros equivalente y partiendo de un conocimiento inicial muy reducido sobre la planta.

Existe todavía, sin embargo, un problema que es inherente a los sistemas difusos con reglas definidas a lo largo de una rejilla y que se conoce popularmente como “la maldición de la dimensionalidad”. Este problema se debe a que el número total de reglas utilizado crece de forma exponencial con el número de dimensiones del espacio de entrada. Para el caso de sistemas con más de seis entradas y varias funciones de pertenencia en cada una de ellas, el número total de reglas hace cualquier tratamiento matemático inabordable para los computadores actuales. Una posible solución para este tipo de casos consiste en definir los antecedentes de las reglas de forma individual de modo que cada regla tendría asociada un número mayor de parámetros pero evitando así que el número de ellas crezca tan abruptamente. La otra solución sería definir sub-sistemas con un menor número de variables de entrada y conectarlos en cascada. La adaptación de los algoritmos propuestos a ambos tipos de soluciones es una cuestión abierta en la que ya estamos trabajando en la actualidad.

Apéndice A

FUNCIONES DE PERTENENCIA

En este apéndice se presentan las configuraciones de funciones de pertenencia utilizadas a lo largo de esta memoria. Común para todas estas configuraciones es la definición de la función escalón *U(x;a,b)* dada por:

$$U(x;a,b) = \begin{cases} 1 \text{ si a} \le x < b \\ 0 \text{ en otro caso} \end{cases} \quad \text{(A-1)}$$

A.1 Funciones triangulares

En este apartado se presentan las expresiones matemáticas de las funciones de pertenencia triangulares usadas en este trabajo. En el apartado A.1.1 se definirá el concepto de partición triangular, en donde los extremos de las funciones triangulares coinciden con los centros de las funciones vecinas. En el apartado siguiente, A.1.2, se considera el caso más general donde tanto los centros como los extremos son parámetros libres.

A.1.1 Partición triangular (TP)

Es la configuración de funciones de pertenencia principal en este trabajo. Las características de una partición triangular son:

- Presentan un **grado de solapamiento dos**. Generalmente, cuando se utilizan controladores difusos se suele exigir que un dato no active más de dos funciones de pertenencia simultáneamente ya que el cerebro humano no suele actuar de esa forma y el significado de las reglas obtenidas sería complejo. Además, esto permite que las reglas puedan ser fácilmente comprendidas y completadas en aquellas regiones en las que no existen vectores de E/S para determinarlas.

- **Para todo valor del rango que cubren se cumple que** $\sum_{s=1}^{n} \mu_{X^s}(x) = 1$**.** De esta forma, si un dato activa una función de pertenencia con valor $\mu_{X^{s'}}(x)$ debe haber otra función de pertenencia que active con un valor $1 - \mu_{X^{s'}}(x)$. Esta condición, junto con la anterior hace que la distribución quede completamente definida por la posición de los centros de cada una de las funciones triangulares. Los otros dos parámetros (el lateral izquierdo y el derecho) vienen definidos implícitamente por los centros de las dos funciones vecinas. En la siguiente figura se representa la distribución dada por los centros situados en los puntos $\{c_v^1 = 0.0,\ c_v^2 = 0.20,\ c_v^3 = 0.70,\ c_v^4 = 0.90,\ c_v^5 = 1.0\}$ para la variable *v*:

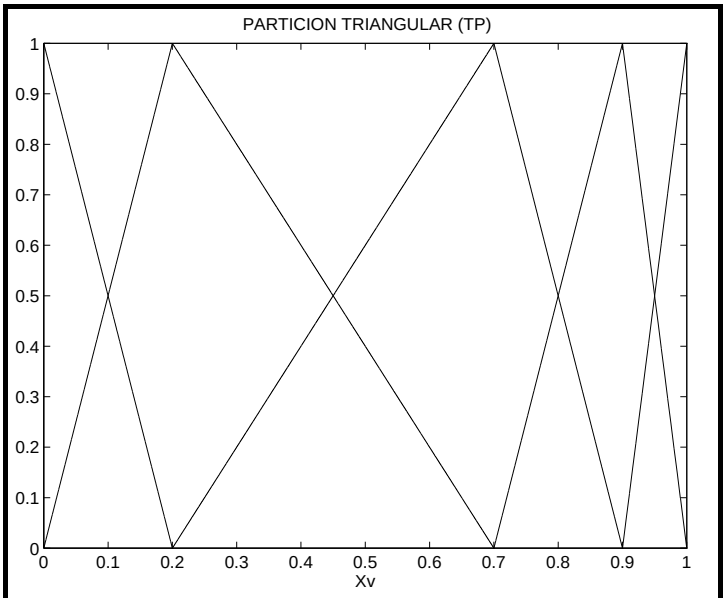


Fig. A.1 Partición triangular (TP).

Además, es muy fácil obtener la función $\mu_{X_v^i}(x)$ para un x dado. Realizando operaciones sencillas es fácil demostrar que:

$$\mu_{X_v^i}(x) = \begin{cases} U(x; -\infty, c_v^i) + \dfrac{c_v^{i+1} - x}{c_v^{i+1} - c_v^i} U(x; c_v^i, c_v^{i+1}) & \text{si i = 1} \\ \dfrac{x - c_v^{i-1}}{c_v^i - c_v^{i-1}} U(x; c_v^{i-1}, c_v^i) + \dfrac{c_v^{i+1} - x}{c_v^{i+1} - c_v^i} U(x; c_v^i, c_v^{i+1}) & \text{i} \neq 1 \text{ e i} \neq \text{n} \\ \dfrac{x - c_v^{i-1}}{c_v^i - c_v^{i-1}} U(x; c_v^{i-1}, c_v^i) + U(x; c_v^i, \infty) & \text{si i = n} \end{cases} \qquad \text{(A-2)}$$

donde se ha considerado que las funciones de los extremos presentan un grado de pertenencia 1 para los puntos que están más allá del rango de definición de la variable.

Las derivadas de las funciones de pertenencia así definidas con respecto a sus parámetros (que son sólo los centros) vendrán dadas por:

$$\frac{\partial \mu_{X_v^i}(x)}{\partial c_v^j} = \begin{cases} \dfrac{-\left(c_v^i - x\right)}{\left(c_v^i - c_v^{i-1}\right)^2} U(x; c_v^{i-1}, c_v^i) & \text{si } j = i-1 \\ \dfrac{-\left(x - c_v^{i-1}\right)}{\left(c_v^i - c_v^{i-1}\right)^2} U(x; c_v^{i-1}, c_v^i) + \dfrac{c_v^{i+1} - x}{\left(c_v^{i+1} - c_v^i\right)^2} U(x; c_v^i, c_v^{i+1}) & \text{si } j = i \\ \dfrac{x - c_v^i}{\left(c_v^{i+1} - c_v^i\right)^2} U(x; c_v^i, c_v^{i+1}) & \text{si } j = i+1 \end{cases}$$

siendo la derivada nula en cualquier otro caso.

A.1.2 Funciones triangulares libres (TL)

En este caso, las funciones de pertenencia son triangulares y no se les obliga a tener un grado de solapamiento igual a 2 sino que los extremos se consideran también variables libres. Denotando por un signo '+' el lateral derecho y por un signo '–' el izquierdo tendremos que para la función de pertenencia X_v^i, su centro vendrá dado por c_v^i y sus laterales izquierdo y derecho por $c_v^{i,-}$ y $c_v^{i,+}$ respectivamente. En la figura A.2 se

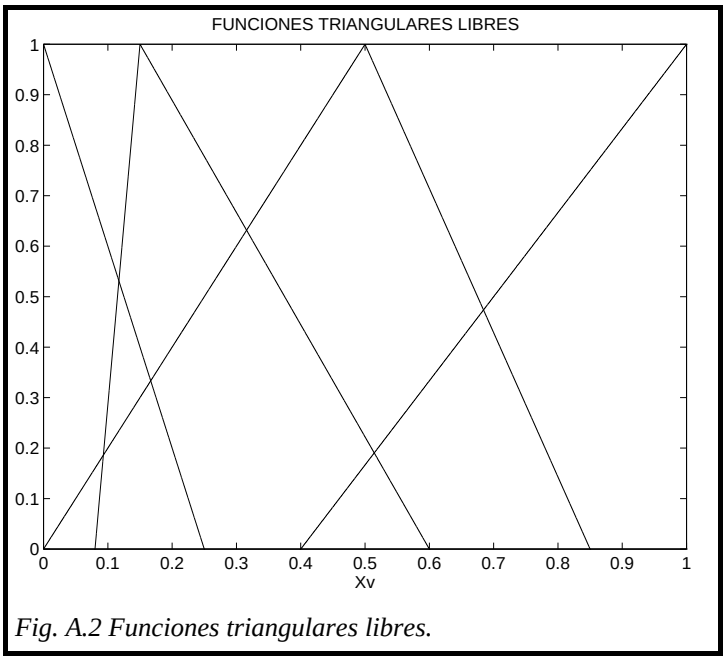


Fig. A.2 Funciones triangulares libres.

representa una configuración con estas características.

En este caso, el grado de pertenencia del valor de entrada x a la función de pertenencia X_v^i vendrá dado por:

$$\mu_{X_v^i}(x) = \frac{x - c_v^{i,-}}{c_v^i - c_v^{i,-}} U(x; c_v^{i,-}, c_v^i) + \frac{c_v^{i,+} - x}{c_v^{i,+} - c_v^i} U(x; c_v^i, c_v^{i,+}) \quad \text{(A-3)}$$

Las derivadas de las funciones de pertenencia así definidas con respecto a sus parámetros serían, en este caso:

$$\frac{\partial \mu_{X_v^j}(x)}{\partial c_v^j} = \frac{-\left(x - c_v^{j,-}\right)}{\left(c_v^j - c_v^{j,-}\right)^2} U(x; c_v^{j,-}, c_v^j) + \frac{c_v^{j,+} - x}{\left(c_v^{j,+} - c_v^j\right)^2} U(x; c_v^j, c_v^{j,+})$$

$$\frac{\partial \mu_{X_v^j}(x)}{\partial c_v^{j,-}} = \frac{-\left(c_v^j - x\right)}{\left(c_v^j - c_v^{j,-}\right)^2} U(x; c_v^{j,-}, c_v^j)$$

$$\frac{\partial \mu_{X_v^j}(x)}{\partial c_v^{j,+}} = \frac{x - c_v^j}{\left(c_v^{j,+} - c_v^j\right)^2} U(x; c_v^j, c_v^{j,+})$$

A.2 Funciones gaussianas

En este apartado se presentan las expresiones matemáticas de las funciones de pertenencia gaussianas usadas en este trabajo. En el apartado A.2.1 se definirá la configuración estándar. En la sección siguiente (A.2.2), una configuración equivalente a la partición triangular, donde las funciones gaussianas son limitadas fijando la intersección entre ellas a un cierto valor *L*. Finalmente, se presentan las gaussianas generalizadas o pseudo-gaussianas asimétricas, donde las funciones de pertenencia están representadas por gaussianas con distinta desviación (σ) a la izquierda y a la derecha de modo que la función de pertenencia deja de ser simétrica.

A.2.1 Funciones de pertenencia gaussianas (G)

Es, junto con la partición triangular, una de las configuraciones más ampliamente utilizadas en la bibliografía. En este caso, cada función de pertenencia tiene la forma de una gaussiana fijada por su centro y su desviación, valiendo la unidad en el punto

central y tendiendo a cero conforme nos alejamos de él, sin llegar nunca a dicho valor. En la siguiente figura vemos un ejemplo de dicha configuración.

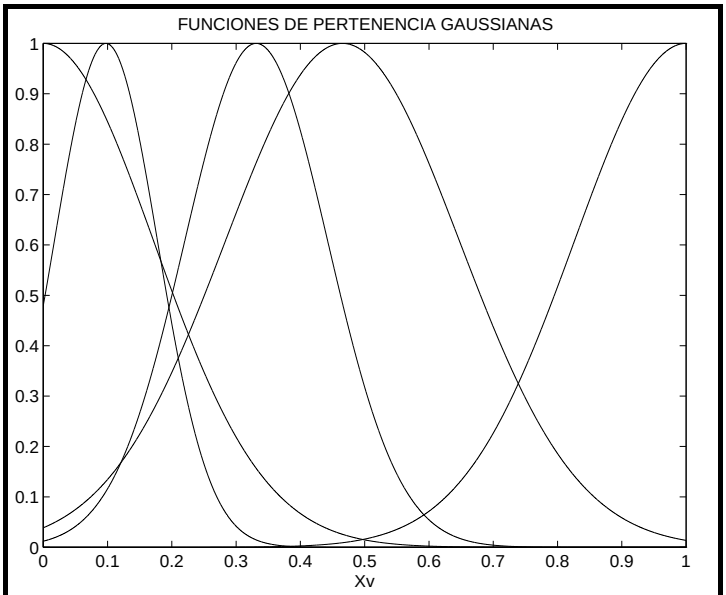


Fig. A.3 Funciones de pertenencia gaussianas.

El grado de pertenencia de la entrada *x* a la función de pertenencia X_v^i vendrá dado por:

$$\mu_{X_v^i}(x) = e^{-\frac{\left(x-c_v^i\right)^2}{\sigma_v^i}} \tag{A-4}$$

Las derivadas de las funciones de pertenencia así definidas con respecto a sus parámetros serían, en este caso:

$$\frac{\partial \mu_{X_v^i}(x)}{\partial c_v^i} = 2\frac{x-c_v^i}{\sigma_v^i}\mu_{X_v^i}(x)$$

$$\frac{\partial \mu_{X_v^i}(x)}{\partial \sigma_v^i} = 2\left(\frac{x-c_v^i}{\sigma_v^i}\right)^2 \mu_{X_v^i}(x)$$

A.2.2 Partición pseudo-gaussiana (PPG)

En este apartado presentaremos un tipo novedoso de configuración de funciones de pertenencia que también depende exclusivamente de la localización de los centros igual que pasaba con la partición triangular del apartado A.1.1. Para poder obtener configuraciones parecidas a las obtenidas con las funciones triangulares vamos a usar pseudo-gaussianas asimétricas, es decir, vamos a usar distintos valores de las desviaciones típicas según la rama que se considere. Denotando por un signo '+' el

valor de la desviación en la zona derecha y por un signo '–' la de la zona izquierda, tendremos que para la función de pertenencia X_v^i su centro vendrá dado por c_v^i y sus desviaciones izquierda y derecha por $\sigma_v^{i,-}$ y $\sigma_v^{i,+}$ respectivamente. Fijando el valor *L* que queremos que tenga la función justo en los centros contiguos tenemos que:

$$\mu_{X_v^i}(x = c_v^{i+1}) = e^{-\frac{\left(c_v^{i+1} - c_v^i\right)^2}{\sigma_v^{i,+}}} = L$$

$$\mu_{X_v^i}(x = c_v^{i-1}) = e^{-\frac{\left(c_v^{i-1} - c_v^i\right)^2}{\sigma_v^{i,-}}} = L$$

Despejando las desviaciones izquierda y derecha en cada una de las expresiones anteriores obtenemos que:

$$\sigma_v^{i,+} = \frac{\left(c_v^{i+1} - c_v^i\right)^2}{\ln\left(1/L\right)} \equiv k\left(c_v^{i+1} - c_v^i\right)^2 \qquad \text{(A-5)}$$

$$\sigma_v^{i,-} = \frac{\left(c_v^i - c_v^{i-1}\right)^2}{\ln\left(1/L\right)} \equiv k\left(c_v^i - c_v^{i-1}\right)^2 \qquad \text{(A-6)}$$

de forma que las desviaciones vendrán unívocamente determinadas por la localización de los centros, tal y como sucedía en el caso de la partición triangular. De las ecuaciones (A-5) y (A-6) además tenemos $\sigma_v^{i,+} = \sigma_v^{i+1,-}$.

Si se quiere poner como referencia no el valor del grado de pertenencia en el centro vecino sino en el punto medio (que puede parecer lo más apropiado) lo que tendremos será:

$$\mu_{X_v^i}\left(x = c_v^i + \frac{c_v^{i+1} - c_v^i}{2}\right) = e^{-\frac{\left(c_v^{i+1} - c_v^i\right)^2}{4\sigma_v^{i,+}}} = M$$

resultando finalmente que $L=M^4$.

En la siguiente figura se representa la distribución definida por los centros situados en los puntos {*0.0, 0.20, 0.70, 0.90, 1.0*}, con *L=0.01*:

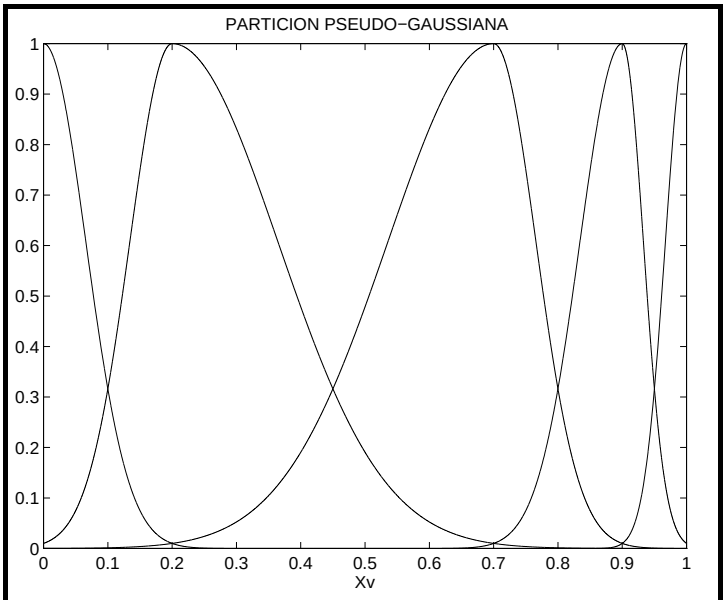


Fig. A.4 Partición pseudo-gaussiana.

El grado de pertenencia de la entrada *x* a la función de pertenencia X_v^i vendrá, en una partición pseudo gaussiana, dada por:

$$\mu_{X_v^i}(x)=\begin{cases} U(x;-\infty,c_v^i)+e^{-\frac{(x-c_v^i)^2}{\sigma_v^{i,+}}}U(x;c_v^i,\infty) & \text{i}=1 \\ e^{-\frac{(x-c_v^i)^2}{\sigma_v^{i,-}}}U(x;-\infty,c_v^i)+e^{-\frac{(x-c_v^i)^2}{\sigma_v^{i,+}}}U(x;c_v^i,\infty) & \text{i}\neq 1 \text{ e i}\neq \text{n} \\ e^{-\frac{(x-c_v^i)^2}{\sigma_v^{i,-}}}U(x;-\infty,c_v^i)+U(x;c_v^i,\infty) & \text{i}=\text{n} \end{cases} \qquad \text{(A-7)}$$

donde, al igual que en el caso de la partición triangular, las funciones de los extremos presentan un grado de pertenencia 1 para los puntos que están más allá del rango de definición de la variable.

Finalmente, a partir de la expresión anterior y de la dependencia de las desviaciones con los centros, tendremos que:

$$\frac{\partial \mu_{X_v^i}(x)}{\partial c_v^j} = \begin{cases} -2k\left(c_v^i - c_v^{i-1}\right)\left(\dfrac{x-c_v^i}{\sigma_v^{i,-}}\right)^2 e^{-\frac{\left(x-c_v^i\right)^2}{\sigma_v^{i,-}}} U(x;-\infty,c_v^i) & j = i-1 \\ 2\dfrac{x-c_v^i}{\sigma_v^{i,-}} e^{-\frac{\left(x-c_v^i\right)^2}{\sigma_v^{i,-}}}\left(1+k\dfrac{x-c_v^i}{\sigma_v^{i,-}}\left(c_v^i - c_v^{i-1}\right)\right)U(x;-\infty,c_v^i)+ & \\ 2\dfrac{x-c_v^i}{\sigma_v^{i,+}} e^{-\frac{\left(x-c_v^i\right)^2}{\sigma_v^{i,+}}}\left(1-k\dfrac{x-c_v^i}{\sigma_v^{i,+}}\left(c_v^{i+1} - c_v^i\right)\right)U(x;c_v^i,\infty) & i = j \\ 2k\left(c_v^{i+1} - c_v^i\right)\left(\dfrac{x-c_v^i}{\sigma_v^{i,+}}\right)^2 e^{-\frac{\left(x-c_v^i\right)^2}{\sigma_v^{i,+}}} U(x;c_v^i,\infty) & j = i+1 \end{cases} \quad \text{(A-8)}$$

A.2.3 Funciones pseudo-gaussianas libres (PGL)

Esta es una configuración equivalente a las funciones triangulares libres. Ahora los parámetros de las funciones pseudo-gaussianas del apartado anterior se dejan libres dotando a las funciones de pertenencia de un número mayor de grados de libertad. Al igual que en el caso anterior, la función de pertenencia X_v^i de la variable *v* vendrá dada por tres parámetros: su centro c_v^i y sus desviaciones izquierda y derecha $\sigma_v^{i,-}$ y $\sigma_v^{i,+}$. En la siguiente figura se representa un ejemplo de dichas funciones de pertenencia:

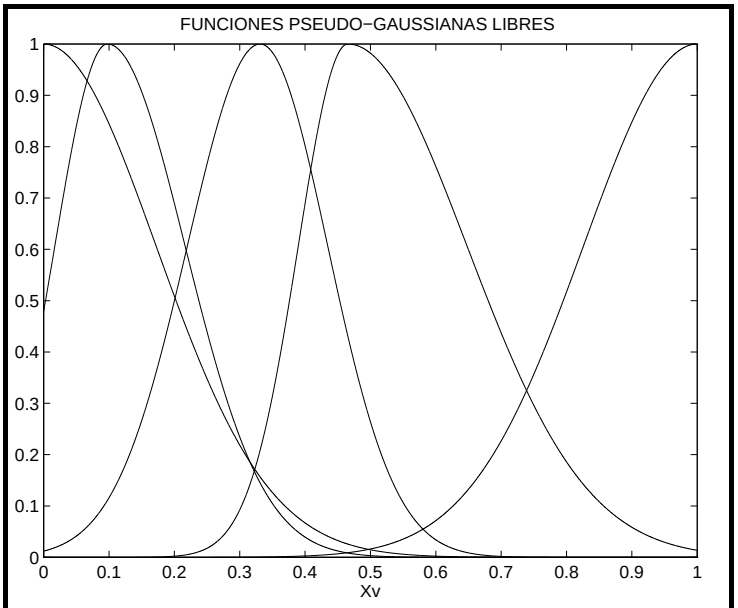


Fig. A.5 Funciones pseudo-gaussianas libres.

El grado de pertenencia de la entrada *x* a la función de pertenencia X_v^i vendrá dado por:

$$\mu_{X_v^i}(x) = e^{-\frac{\left(x-c_v^i\right)^2}{\sigma_v^{i,-}}} U(x;-\infty,c_v^i) + e^{-\frac{\left(x-c_v^i\right)^2}{\sigma_v^{i,+}}} U(x;c_v^i,\infty) \tag{A-9}$$

Las derivadas de las funciones de pertenencia así definidas con respecto a sus parámetros serían:

$$\frac{\partial \mu_{X_v^i}(x)}{\partial c_v^i} = 2\frac{x-c_v^i}{\sigma_v^{i,-}} e^{-\frac{\left(x-c_v^i\right)^2}{\sigma_v^{i,-}}} U(x;-\infty,c_v^i) + 2\frac{x-c_v^i}{\sigma_v^{i,+}} e^{-\frac{\left(x-c_v^i\right)^2}{\sigma_v^{i,+}}} U(x;c_v^i,\infty)$$

$$\frac{\partial \mu_{X_v^i}(x)}{\partial \sigma_v^{i,-}} = 2\left(\frac{x-c_v^i}{\sigma_v^{i,-}}\right)^2 e^{-\frac{\left(x-c_v^i\right)^2}{\sigma_v^{i,-}}} U(x;-\infty,c_v^i)$$

$$\frac{\partial \mu_{X_v^i}(x)}{\partial \sigma_v^{i,+}} = \left(\frac{x-c_v^i}{\sigma_v^{i,+}}\right)^2 e^{-\frac{\left(x-c_v^i\right)^2}{\sigma_v^{i,+}}} U(x;c_v^i,\infty)$$

Apéndice B

DERIVACIÓN DE EXPRESIONES

En este apéndice se derivarán las expresiones que han sido incluidas en la presente memoria y cuya demostración no es trivial.

B.1 Invarianza del ECMN ante cambios de escala

Recordemos que el Error Cuadrático Medio Normalizado *(ECMN)* estaba definido por (ecuación 2-22):

$$ECMN \equiv \sqrt{\frac{\overline{e^2}}{\sigma_z^2}} \tag{B-1}$$

donde σ_z^2 es la varianza de los datos de salida y $\overline{e^2}$ el error cuadrático medio entre la salida obtenida y la verdadera. Sería deseable, a modo comparativo, que un índice que mida el grado de aproximación obtenido sea independiente de factores de escala. Así, la función $z_1(x) = x^2$ y la función $z_2(x) = 10x^2 + 8$ son esencialmente la misma. Sin embargo, índices como el error cuadrático medio darán siempre valores mayores al aproximar z_2 que z_1, para configuraciones que son realmente equivalentes. El error cuadrático medio normalizado no presenta este inconveniente como veremos a continuación proporcionando, por tanto, un índice mucho más representativo que el error cuadrático medio.

Supongamos que tenemos dos funciones $z_{1d}(\vec{x})$ y $z_{2d}(\vec{x})$, siendo $z_{2d} = a \cdot z_{1d} + b$, con a y b constantes y que hemos utilizado una serie de K vectores de Entrada/Salida para aproximar la función z_{1d} obteniendo la función z_1. Pues bien, la función z_2 obtenida a partir de $z_2 = a \cdot z_1 + b$ es, claramente, una aproximación a z_{2d} equivalente a la de z_1 con

respecto a z_{1d}. El ECMN así lo confirma como veremos a continuación. Para ello, debemos calcular la relación que guardan entre sus errores cuadráticos medios y entre las varianzas de las salidas.

$$ECMN_{z_1} \equiv \sqrt{\frac{\overline{e_{z_1}^2}}{\sigma_{z_1}^2}} \qquad ECMN_{z_2} \equiv \sqrt{\frac{\overline{e_{z_2}^2}}{\sigma_{z_2}^2}} \tag{B-2}$$

Errores cuadráticos medios:

$$\begin{aligned} &\overline{e_{z_2}^2} = \frac{1}{K}\sum_{k=1}^{K}\left(e_{z_2}(\vec{x}_k)\right)^2 = \frac{1}{K}\sum_{k=1}^{K}\left(z_{2d}(\vec{x}_k) - z_2(\vec{x}_k)\right)^2 = \\ &\frac{1}{K}\sum_{k=1}^{K}\left(\{a\cdot z_{1d}(\vec{x}_k)+b\} - \{a\cdot z_1(\vec{x}_k)+b\}\right)^2 = \\ &a^2\frac{1}{K}\sum_{k=1}^{K}\left(z_{1d}(\vec{x}_k) - z_1(\vec{x}_k)\right)^2 = a^2\frac{1}{K}\sum_{k=1}^{K}\left(e_{z_2}(\vec{x}_k)\right)^2 = a^2\overline{e_{z_1}^2} \end{aligned} \tag{B-3}$$

Varianzas de las salidas:

$$\begin{aligned} &\sigma_{z_2}^2 = \frac{1}{K}\sum_{k=1}^{K}\left(z_2(\vec{x}_k) - \overline{z_2}\right)^2 = \frac{1}{K}\sum_{k=1}^{K}\left(\{a\cdot z_1(\vec{x}_k)+b\} - \{a\cdot\overline{z_1}+b\}\right)^2 = \\ &= a^2\frac{1}{K}\sum_{k=1}^{K}\left(z_1(\vec{x}_k) - \overline{z_1}\right)^2 = a^2\sigma_{z_1}^2 \end{aligned} \tag{B-4}$$

De donde se obtiene fácilmente que $ECMN_{z_1} = ECMN_{z_2}$. El realizar la raíz cuadrada se debe a que de esta forma el resultado tiene la misma "dimensionalidad" que los valores de salida.

B.2 Demostración de la expresión 3-20

Cuando utilizamos una partición triangular como configuración de funciones de pertenencia (ver Apéndice A) y empleamos un conjunto completo de reglas, la expresión de la función de salida es mucho más sencilla ya que el denominador de 3-2 vale siempre la unidad, es decir:

$$\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(\prod_{m=1}^{N}\mu_{X_m^{i_m}}(x_m)\right) = 1 \tag{B-5}$$

con lo que

$$\tilde{F}(\vec{x}) = \sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(R_{i_1 i_2 \ldots i_N}\cdot\prod_{m=1}^{N}\mu_{X_m^{i_m}}(x_m)\right) \tag{B-6}$$

Para demostrar que esta característica es cierta para una partición triangular, basta notar que cada sumatorio es independiente del resto y aplicar la condición $\sum_{s=1}^{n} \mu_{X^s}(x) = 1$ (ver Apéndice A), que es válida para cada conjunto de funciones en cada variable de forma independiente:

$$\begin{aligned}
&\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(\prod_{m=1}^{N}\mu_{X_m^{i_m}}(x_m)\right) = \sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(\mu_{X_1^{i_1}}(x_1)\cdot\mu_{X_2^{i_2}}(x_2)\cdots\mu_{X_N^{i_N}}(x_N)\right) \\
&= \sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_{N-1}=1}^{n_{N-1}}\left(\mu_{X_1^{i_1}}(x_1)\cdot\mu_{X_2^{i_2}}(x_2)\cdots\mu_{X_{N-1}^{i_{N-1}}}(x_{N-1})\right)\sum_{i_N=1}^{n_N}\left(\mu_{X_N^{i_N}}(x_N)\right) \qquad \text{(B-7)} \\
&= \sum_{i_1=1}^{n_1}\left(\mu_{X_1^{i_1}}(x_1)\right)\cdot\sum_{i_2=1}^{n_2}\left(\mu_{X_2^{i_2}}(x_2)\right)\cdots\sum_{i_N=1}^{n_N}\left(\mu_{X_N^{i_N}}(x_N)\right) = 1\cdot 1\cdots 1 = 1
\end{aligned}$$

Apéndice C

VALORES DE LAS NUEVAS REGLAS

En este apéndice se analiza el caso general del desarrollo expuesto en el apartado 5.6.1 donde se analizaba el problema de los valores iniciales que deben tener las nuevas reglas que se generan como consecuencia de la inserción de una nueva función de pertenencia en alguna de las variables de entrada. Se demostrará que la expresión 5-65 es válida para el caso de usar particiones triangulares o pseudo-gaussianas y se justificará su uso para el resto de los casos.

C.1 Derivación de la expresión 5-65

Recordemos que al añadir una nueva función de pertenencia en la variable v se generaban automáticamente $\prod_{\substack{i=1 \\ i\neq v}}^{N} n_i$ reglas nuevas que tienen la forma:

$$IF\ x_1 \text{ is } X_1^{i_1}\ AND \ldots AND\ x_v \text{ is } X_v^{j_v}\ AND \ldots AND\ x_N \text{ is } X_N^{i_N}\ THEN\ z = R_{i_1 i_2 \ldots i_v = j_v \ldots i_N}$$

siendo j_v la posición que ocupa la nueva función de pertenencia. Consecuentemente, el centro que antes ocupaba dicha posición pasa a llamarse $c_v^{j_v+1}$ y así sucesivamente con el resto de centros que haya a la derecha del nuevo $c_v^{j_v}$. Igual pasa con el resto de parámetros.

Como ya se hizo en el apartado 5.6.1 denotaremos por $\tilde{\Theta}$ al conjunto antiguo de parámetros y por Θ al nuevo. De esta forma:

$$n_v = \tilde{n}_v + 1$$

$$n_i = \tilde{n}_i \quad \forall i \neq v$$

$$\theta_v^j = \tilde{\theta}_v^j \quad si\ j < j_v$$

$$\theta_v^j = \tilde{\theta}_v^{j-1} \quad si\ j > j_v$$

$$\theta_i^j = \tilde{\theta}_i^j \quad \forall i \neq v$$

y se comprobó que seleccionando las reglas de la forma

$$\mathrm{R}_{\mathrm{i}_1,..,\mathrm{i}_v,...,\mathrm{i}_N} = \tilde{\mathrm{R}}_{\mathrm{i}_1,..,\mathrm{i}_v,...,\mathrm{i}_N} \quad si\ i_v < j_v$$

$$\mathrm{R}_{\mathrm{i}_1,..,\mathrm{i}_v,...,\mathrm{i}_N} = \tilde{\mathrm{R}}_{\mathrm{i}_1,..,\mathrm{i}_v\text{-}1,...,\mathrm{i}_N} \quad si\ i_v > j_v$$

se cumplía que:

$$\hat{F}(\vec{x};\Theta) = \hat{F}(\vec{x};\tilde{\Theta}) \tag{C-1}$$

en todos los puntos cuya componente *v* caiga fuera del rango $[c_v^{j_v-1}, c_v^{j_v+1}]$. Dentro de dicho rango, el criterio de selección de reglas se escogía de tal forma que se cumpliera:

$$\hat{F}(\vec{c};\Theta) = \hat{F}(\vec{c};\tilde{\Theta}) \tag{C-2}$$

donde $\vec{c} = (c_1^{i_1},...,c_v^{i_v=j_v},...,c_N^{i_N})$, con i_1=1...n_1, ..., i_N=1...n_N. Es decir, justo en los puntos donde cada una de las nuevas regla $\mathrm{R}_{\mathrm{i}_1,..,\mathrm{i}_v=\mathrm{j}_v,...,\mathrm{i}_N}$ se activan con grado 1.

Vamos ahora a analizar la ecuación (C-2) para el caso general de funciones de N dimensiones. Utilizando la expresión matemática de la función defuzzificada obtenemos, en cada caso:

$$\hat{\mathrm{F}}(\vec{c};\Theta) = \frac{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(R_{i_1 i_2 \ldots i_N}\cdot\mu_{i_1 i_2 \ldots i_N}(\vec{c})\right)}{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\mu_{i_1 i_2 \ldots i_N}(\vec{c})} \tag{C-3}$$

$$\hat{\mathrm{F}}(\vec{c};\tilde{\Theta}) = \frac{\sum_{i_1=1}^{\tilde{n}_1}\sum_{i_2=1}^{\tilde{n}_2}\cdots\sum_{i_N=1}^{\tilde{n}_N}\left(\tilde{R}_{i_1 i_2 \ldots i_N}\cdot\tilde{\mu}_{i_1 i_2 \ldots i_N}(\vec{c})\right)}{\sum_{i_1=1}^{\tilde{n}_1}\sum_{i_2=1}^{\tilde{n}_2}\cdots\sum_{i_N=1}^{\tilde{n}_N}\tilde{\mu}_{i_1 i_2 \ldots i_N}(\vec{c})} \tag{C-4}$$

Reemplazando los valores antiguos por los nuevos en esta última ecuación obtenemos:

$$\hat{\mathrm{F}}\left(\vec{c};\tilde{\Theta}\right)=\frac{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{\substack{i_v=1\\ i_v\neq j_v}}^{n_v}\cdots\sum_{i_N=1}^{n_N}\left(R_{i_1i_2\ldots i_N}\cdot\mu_{i_1i_2\ldots i_N}(\vec{c})\right)}{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{\substack{i_v=1\\ i_v\neq j_v}}^{n_v}\cdots\sum_{i_N=1}^{n_N}\mu_{i_1i_2\ldots i_N}(\vec{c})} \tag{C-5}$$

Por otra parte, la ecuación (C-3) se puede expresar de la siguiente manera:

$$\hat{\mathrm{F}}\left(\vec{c};\Theta\right)=\frac{\sum_{i_1=1}^{n_1}\cdots\sum_{\substack{i_v=1\\ i_v\neq j_v}}^{n_v}\cdots\sum_{i_N=1}^{n_N}\left(R_{i_1i_2\ldots i_N}\mu_{i_1i_2\ldots i_N}(\vec{c})\right)+\sum_{i_1=1}^{n_1}\cdots\sum_{i_{v-1}=1}^{n_{v-1}}\sum_{i_{v+1}=1}^{n_{v+1}}\cdots\sum_{i_N=1}^{n_N}\left(R_{i_1i_2\ldots i_v=j_v\ldots i_N}\mu_{i_1i_2\ldots i_v=j_v\ldots i_N}(\vec{c})\right)}{\sum_{i_1=1}^{n_1}\cdots\sum_{\substack{i_v=1\\ i_v\neq j_v}}^{n_v}\cdots\sum_{i_N=1}^{n_N}\mu_{i_1i_2\ldots i_N}(\vec{c})+\sum_{i_1=1}^{n_1}\cdots\sum_{i_{v-1}=1}^{n_{v-1}}\sum_{i_{v+1}=1}^{n_{v+1}}\cdots\sum_{i_N=1}^{n_N}\mu_{i_1i_2\ldots i_v=j_v\ldots i_N}(\vec{c})}$$

donde simplemente se han sacado los términos de la sumatoria que contienen precisamente los consecuentes de las nuevas reglas. De esta forma, la ecuación (C-2) quedaría de la forma:

$$\frac{a+c}{b+d}=\frac{a}{b}$$

siendo

$$a\equiv\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{\substack{i_v=1\\ i_v\neq j_v}}^{n_v}\cdots\sum_{i_N=1}^{n_N}\left(R_{i_1i_2\ldots i_N}\cdot\mu_{i_1i_2\ldots i_N}(\vec{c})\right)$$

$$b\equiv\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{\substack{i_v=1\\ i_v\neq j_v}}^{n_v}\cdots\sum_{i_N=1}^{n_N}\mu_{i_1i_2\ldots i_N}(\vec{c})$$

$$c\equiv\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_{v-1}=1}^{n_{v-1}}\sum_{i_{v+1}=1}^{n_{v+1}}\cdots\sum_{i_N=1}^{n_N}\left(R_{i_1i_2\ldots i_v=j_v\ldots i_N}\cdot\mu_{i_1i_2\ldots i_v=j_v\ldots i_N}(\vec{c})\right)$$

$$d\equiv\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_{v-1}=1}^{n_{v-1}}\sum_{i_{v+1}=1}^{n_{v+1}}\cdots\sum_{i_N=1}^{n_N}\mu_{i_1i_2\ldots i_v=j_v\ldots i_N}(\vec{c})$$

de donde:

$$\frac{c}{d}=\frac{1}{d}\left(\frac{(b+d)}{b}a-a\right)=\frac{a}{b}$$

es decir:

$$\frac{\sum_{i_1=1}^{n_1}\cdots\sum_{i_{v-1}=1}^{n_{v-1}}\sum_{i_{v+1}=1}^{n_{v+1}}\cdots\sum_{i_N=1}^{n_N}\left(R_{i_1 i_2 \ldots i_v=j_v \ldots i_N}\,\mu_{i_1 i_2 \ldots i_v=j_v \ldots i_N}(\vec{c})\right)}{\sum_{i_1=1}^{n_1}\cdots\sum_{i_{v-1}=1}^{n_{v-1}}\sum_{i_{v+1}=1}^{n_{v+1}}\cdots\sum_{i_N=1}^{n_N}\mu_{i_1 i_2 \ldots i_v=j_v \ldots i_N}(\vec{c})} = \frac{\sum_{i_1=1}^{n_1}\cdots\sum_{\substack{i_v=1\\ i_v\neq j_v}}^{n_v}\cdots\sum_{i_N=1}^{n_N}\left(R_{i_1 i_2 \ldots i_N}\,\mu_{i_1 i_2 \ldots i_N}(\vec{c})\right)}{\sum_{i_1=1}^{n_1}\cdots\sum_{\substack{i_v=1\\ i_v\neq j_v}}^{n_v}\cdots\sum_{i_N=1}^{n_N}\mu_{i_1 i_2 \ldots i_N}(\vec{c})} = \hat{F}(\vec{c};\widetilde{\Theta})$$

lo cual quiere decir que la función defuzzificada para $\vec{x}=\vec{c}$ utilizando sólo las reglas nuevas debe coincidir con el valor que para esos puntos daba la función antigua.

Si expresamos la ecuación anterior de la forma:

$$\sum_{i_1=1}^{n_1}\cdots\sum_{i_{v-1}=1}^{n_{v-1}}\sum_{i_{v+1}=1}^{n_{v+1}}\cdots\sum_{i_N=1}^{n_N}\left(R_{i_1 i_2 \ldots i_v=j_v \ldots i_N}\,\mu_{i_1 i_2 \ldots i_v=j_v \ldots i_N}(\vec{c})\right) = \hat{F}(\vec{c};\widetilde{\Theta})\cdot\sum_{i_1=1}^{n_1}\cdots\sum_{i_{v-1}=1}^{n_{v-1}}\sum_{i_{v+1}=1}^{n_{v+1}}\cdots\sum_{i_N=1}^{n_N}\mu_{i_1 i_2 \ldots i_v=j_v \ldots i_N}(\vec{c})$$

y teniendo en cuenta que es válida para todos los centros de la forma

$\vec{c}=(c_1^{i_1},\ldots,c_v^{i_v=j_v},\ldots,c_N^{i_N})$, con $i_1=1\ldots n_1$, ..., $i_N=1\ldots n_N$

resulta que la expresión anterior nos proporciona un sistema de $\prod_{\substack{i=1\\ i\neq v}}^{N} n_i$ ecuaciones lineales con igual número de incógnitas (las nuevas reglas) de modo que se puede invertir para obtener el valor de las nuevas reglas.

En el caso de que tengamos una partición triangular, el sistema de ecuaciones se simplifica radicalmente ya que en este caso cada función de pertenencia presenta un grado de pertenencia cero en los centros vecinos (y por supuesto también en el resto de los centros que no sean el suyo). En este caso, si consideramos el vector $\vec{c}_{j_1\ldots j_N} = (c_1^{j_1},\ldots, c_v^{j_v},\ldots, c_N^{j_N})$, tendremos que sólo la regla $R_{j_1,\ldots,j_v,\ldots,j_N}$ se activa, es decir:

$$\mu_{i_1,\ldots,j_v,\ldots,i_N}\left(\vec{c}_{j_1\ldots j_N}\right) = \begin{cases} 1 & \text{si } (i_1,\ldots,i_N)=(j_1,\ldots,j_N) \\ 0 & \text{en el resto de los casos} \end{cases} \qquad \text{(C-6)}$$

De esta forma, las nuevas reglas vendrán dadas por:

$$R_{i_1,\ldots,i_v=j_v,\ldots,i_N} = \hat{F}(\vec{c};\widetilde{\Theta}) \qquad \text{(C-7)}$$

donde $\vec{c}=(c_1^{i_1},\ldots,c_v^{i_v=j_v},\ldots,c_N^{i_N})$, con $i_1=1\ldots n_1$, ..., $i_N=1\ldots n_N$.

En el caso de utilizar una partición pseudo-gaussiana, la ecuación (C-6) también se cumple de forma aproximada siempre que se escoja el parámetro L suficientemente pequeño (ver Apéndice A). En el resto de los casos el grado de solapamiento de las funciones de pertenencia puede ser diverso pero siempre se cumplirá que la regla $R_{j_1,..j_v,...j_N}$ se activará con grado 1 mientras que el resto lo harán con menor grado. Como el grado de solapamiento por lo general no suele ser alto (de hecho, para mantener la interpretabilidad de las reglas esto es deseable (ver apartado 3.8.3)), y debido a que las reglas están continuamente siendo adaptadas mediante los controladores auxiliares ca_1 y ca_2, es justificable extrapolar el resultado obtenido para particiones triangulares y pseudo-gaussianas para el caso de otras configuraciones. De este modo, en lugar de resolver el sistema de ecuaciones lineales anterior, es una buena aproximación el utilizar la expresión (C-7) para calcular los valores iniciales de las nuevas reglas.

En el caso de haber equidistribuido todas las funciones de pertenencia de la variable v, se usará la misma expresión anterior pero moviendo también la variable i_v desde 1 hasta n_v barriéndose, por tanto, todas las reglas. En este caso la condición (C-6) se cumple prácticamente de forma exacta ya que el proceso de equidistribución de las funciones de pertenencia de una variable se realiza como si fueran una TP ó una PGP, según sea el caso.

Finalmente, hay que hacer notar que en el caso de que tras este proceso se conmute a la etapa 1 (y no a la 2), al utilizarse en dicha etapa sólo funciones de pertenencia definidas a través del valor de sus centros (TP ó PGP) se debe aplicar igualmente la expresión anterior a todas las reglas, una vez transformada la configuración actual en una TP ó una PGP.

Apéndice D

PLANTAS CON RETARDOS SUPERIORES AL PERIODO DE MUESTREO

D.1 Demostración de la expresión 5-3

Un caso especial de sistemas son aquéllos en los que la salida en el instante *k+1* no depende de la entrada de control en el instante anterior sino que sufren un retardo de *d* muestras. En estos casos, la ecuación en diferencias del sistema vendrá dado por:

$$y(k) = f'(y(k-1),...,y(k-p'),u(k-d),...,u(k-d-q')) \quad \text{(D-1)}$$

donde queda patente la independencia de la salida con las últimas *d* entradas de control. Si hacemos el cambio de índice $k \rightarrow k+d$ tendremos:

$$y(k+d) = f'(y(k+d-1),...,y(k+d-p'),u(k),...,u(k-q')) \quad \text{(D-2)}$$

Si intentamos despejar de aquí la señal de control en el instante actual *k*, haciendo *y(k+d) = r(k)*, tendríamos:

$$u(k) = F'(r(k),y(k+d-1),...,y(k+d-p'),u(k-1),...,u(k-q'))$$

y, aparentemente, no podríamos obtener nunca la entrada de control óptima al existir una ley de control no causal, ya que necesitaríamos conocer las *d-1* salidas siguientes del sistema *y(k+1),…,y(k+d-1)*.

Sin embargo, este razonamiento es engañoso ya que, por el mismo motivo por el que la salida *y(k)* no depende de *u(k-1), u(k-2),…,u(k-d+1)* las salidas anteriores tampoco dependerán de sus respectivas señales de control anteriores. De esta forma, la salida en el instante *k-1*, *y(k-1)*, ya vendrá predeterminada una vez que se haya introducido en el

sistema la entrada *u(k-d-1)* por lo que no es necesario tener que esperar al instante *k-1* para obtener alguna información nueva. Para comprobar esto matemáticamente, de la ecuación (D-2) haciendo $k \rightarrow k-1$ tendremos:

$$y(k+d-1) = f'(y(k+d-2),...,y(k+d-p'-1),u(k-1),...,u(k-q'-1))$$

por lo que dicha ecuación quedaría:

$$y(k+d) = f'(f'(y(k+d-2),...,y(k+d-p'-1),u(k-1),...,u(k-q'-1)), y(k+d-2),..., y(k+d-p'), u(k), ..., u(k-q'))$$

que pasaría a tener la forma:

$$y(k+d) = f''(y(k+d-2),...,y(k+d-p''),u(k),...,u(k-q''))$$

con $p'' = p'+1$ y $q'' = q'+1$. Igualmente, podríamos hacer lo mismo con *y(k+d-2)*, *y(k+d-3)*, etc. hasta llegar a *y(k+1)*. Finalmente, llegaríamos a obtener la expresión:

$$y(k+d) = f(y(k),...,y(k-p),u(k),...,u(k-q)) \qquad \text{(D-3)}$$

con $p = p'+d-1$ y $q = q'+d-1$.

De esta forma, comprobamos que sí es posible también en estos casos establecer una ley de control causal ya que, igualando *y(k+d) = r(k)*, despejando de la expresión (D-3) y llamando *F* a la función inversa tendríamos que:

$$u(k) = F(r(k),y(k),...,y(k-p),u(k-1),...,u(k-q))$$

y ya sí que conocemos todas esas entradas en el instante *k*.

Bibliografía

[ABE-95] S.Abe, M.S.Lan, "Fuzzy rules extraction directly from numerical data for function approximation", IEEE Trans. Syst. Man and Cyber., vol.25, no.1, pp.119-129, January 1995.

[ALS-83] C.Alsina, E.Trillas, L.Valverde, "On some logical connectives for fuzzy sets theory", Journal of mathematical analysis and applications, vol.93, pp.15-26, 1983.

[AND-97] H.C.Andersen, A.Lotfi, A.C.Tsoi, "A new approach to adaptive fuzzy control: The Controller Output Error Method", IEEE Trans. Syst. Man and Cyber.- Part B, vol.27, no.4, pp.686-691, August 1997.

[AND-98] H.C.Andersen, "The Controller Output Error Method", Ph.D. Thesis, 1998.

[ANT-94] P.Antsaklis, "Defining intelligent control", IEEE Control Systems, vol.14, no.3, pp.4-5, 58-66, 1994

[BAR-98] S.Barada, H.Singh, "Generating optimal adaptive fuzzy-neural models of dynamical systems with applications to control", IEEE Trans. Syst. Man and Cyber., Part C, vol.28, no.3, pp.371-391, Aug. 1998.

[BER-92] H.R.Berenji, P.Khedkar, "Learning and tuning fuzzy logic controllers through reinforcements", IEEE Trans. Neural Networks, vol.3, no.5, pp.724-739, 1992.

[BER-97] M.R.Berthold, K-P.Huber, "Constructing fuzzy graphs from examples", International Computer Science Institute, December 1997.

[BOS-97] J.R.Boston, "Effects of membership function parameters on the performance of a fuzzy signal detector", IEEE Trans. Fuzzy Systems, vol.5, no.2, pp.249-255, May 1997.

[BOX-76] G.E.P.Box, G.M.Jenkins, "Time Series Analysis, Forecasting and Control", 2nd ed. San Francisco, CA: Holden-Day, 1976.

[BUC-92] J.J.Buckley, "Universal fuzzy controllers", Automatica, vol.28, no.6, pp.1245-1248, 1992.

[BUC-93] J.J.Buckley, "Sugeno Type Controllers are Universal Controllers", Fuzzy Sets and Systems, vol.53, pp.299-304, 1993.

[BUC-94] J.J.Buckley, Y.Hayashi "Can fuzzy neural networks approximate continuous fuzzy functions?", Fuzzy Sets and Systems, vol.61, pp.43-52, 1994.

[CAO-92] Z.Cao, A.Kandel "Investigations on the applicability of fuzzy inference", Fuzzy Sets and Systems, vol.49, pp.151-169, 1992.

[CAR-93] E.Cárdenas, J.C.Castillo, O.Cordón, F.Herrera, A.Peregrín, "Influence of fuzzy implication functions and defuzzification methods in fuzzy control", BUSEFAL 57, pp.69-79, 1993.

[CAS-95] J.L.Castro, "Fuzzy logic controllers are universal approximators", IEEE Trans. Syst. Man and Cyber., vol.25, no.4, pp.629-635, April 1995.

[CAS-96] J.L.Castro, M.Delgado, "Fuzzy systems with defuzzification are universal approximators", IEEE Trans. Syst. Man and Cyber., vol.26, no.1, pp.149-152, Feb. 1996.

[CHE-89] Y.Y.Chen, T.C.Taso, "A description of the dynamical behavior of fuzzy systems", IEEE Trans. Syst., Man, Cyber., vol.19, no.4, pp.745-755, 1989.

[CHE-91a] S.Chen, C.F.N.Cowan, P.M.Grant, "Orthogonal Least Squares learning algorithm for radial basis function networks", IEEE Trans. Neural Networks, vol.2, no.2, March 1991.

[CHE-91b] V.Cherkassky, H. Lari-Najafi, "Constrained topological mapping for non-parametric regression analysis", Neural Networks, vol.4, no.1, pp.27-40, 1991.

[CHE-95] F-C.Chen, H.K.Khalil, "Adaptive control of a class of nonlinear discrete-time systems using neural networks", IEEE Trans. Automatic Control, vol.40, no.5, pp.791-801, May 1995.

[CHE-96] V.Cherkassky, D.Gehring, F.Mulier, "Comparison of adaptive methods for function estimation from samples", IEEE Trans. Neural Networks, vol.7, no.4, pp.969-984, July 1996.

[CHO-94] C.Chou, H.Lu, "A heuristic self-tuning fuzzy controller", Fuzzy Sets and Systems, vol.61, pp.249-264, 1994.

[CHR-93] P.Christian, "Fuzzy logic control of a hypervelocity interceptor", Proc. 2nd IEEE Conf. on Fuzzy Syst., pp.877-882, 1993.

[CRO-90] R.S.Crowder III, “Predicting the Mackey-Glass time series with cascade-correlation learning”, In D.Touretzky, G.Hinton and T.Sejnowski, editors, Proceedings of the 1990 Connectionist Model Summer School, pp.117-123, Carnegic Mellon University, 1990.

[DIC-96] J.A.Dickerson, B.Kosko, “Fuzzy Function Approximation with Ellipsoidal Rules”, IEEE Trans. Syst. Man and Cyber.- Part B, vol.26, no.4, pp.542-560, 1996.

[DRI-93] D.Driankov, H.Hellendoorn, M.Reinfrank, “An introduction to fuzzy control”, Springer-Verlag, 1993.

[DUB-91] D.Dubois, H.Prade, “Fuzzy sets in approximate reasoning”, Fuzzy Sets and Systems, vol.40, pp.143-202, 1991.

[FIL-93] D.P.Filev, R.R.Yager, “An adaptive approach to defuzzification based on level sets”, Fuzzy Sets and Systems, vol.54, pp.355-360, 1993.

[FON-93] K.F.Fong, A.P.Loh, “MRAC Control of non-linear systems using neural networks with recursive least squares adaptation”, Proceedings of the IEEE International Conference on Neural Networks, pp.529-533, 1993.

[FRI-81] J.H.Friedman, “Projection pursuit regression”, J. Amer. Statist. Assoc., vol.76, pp.817-823, 1981.

[FRI-91] J.H.Friedman, “Multivariate adaptive regression splines (with discussion)”, Ann. Statist., vol.19, pp.1-141, 1991.

[GEN-94] H.Genther, T.A.Runkler, M.Glesner, “Defuzzification based on fuzzy clustering”, IEEE International Conference on Fuzzy Systems, pp.1645-1648, 1994.

[GUP-80] M.M.Gupta, Y.Tsukamoto, “Fuzzy logic controllers: A perspective”, Proc. Joint Automatic Control Conf., San Francisco, Aug. 1980.

[GUP-91] M.M.Gupta, J.Qi, “Theory of T-norms and fuzzy inference methods”, Fuzzy Sets and Systems, vol.40, pp.431-450, 1991.

[HEL-93] H.Hellendoorn, C.Thomas, “The ξ-Quality defuzzification method”, Fifth IFSA World Congress, pp.1159-1162, 1993.

[HIG-94] C.M.Higgins, R.M.Goodman, “Fuzzy rule-based networks for control”, IEEE Trans. Fuzzy Systems, vol.2, no.1, pp.82-88, 1994.

[HOM-95] A.Homaifar, E.McCormick, “Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms”, IEEE Trans. Fuzzy Systems, vol.3, no.2, pp.129-139, May 1995.

[HOR-89] K.Hornick, M.Stinchcombe, H.White, “Multilayer feedforward networks are Universal Approximators”, Neural Networks, vol.2, no.5, pp.359-366, 1989.

[HOR-92] S.Horikawa, T.Furuhashi, Y.Uchikawa, “On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm”, IEEE Trans. Neural Networks, vol.3, no.5, pp.801-806, 1992.

[ISH-96] H.Ishibuchi, T.Murata, H.Tanaka, “Construction of fuzzy classification systems with linguistic if-then rules using genetic algorithms”, in S.K.Pal and P.P.Wang, Eds., Genetic Algorithms for Pattern Recognition (CRC Press, Boca Raton, FL, USA), pp.227-251, 1996.

[IVA-78] A.G.Ivakhnenko, V.N.Vysotskiy, N.A.Ivakhnenko, “Principal versions of the minimum bias criterion for a model and an investigation of their noise immunity”, Soviet Automat. Control 11, pp.27-45, 1978.

[JAG-95] R.Jager, “Fuzzy Logic in Control”, Tesis Doctoral, Amsterdam, 1995.

[JAG-99] I.Jagielska, C.Mathhews, T.Whitfort, “An investigation into the application of neural networks, fuzzy logic, genetic algorithms, and rough sets to automated knowledge acquisition for classification problems”, Neurocomputing, vol.24, pp.37-54, 1999.

[JAN-92] J.S.R.Jang “Self-learning fuzzy controllers based on temporal back propagation”, IEEE Trans. Neural Networks, vol.3, no.5, pp.714-723, 1992.

[JAN-93] J.S.R.Jang, “ANFIS: Adaptive-network-based fuzzy inference systems”, IEEE Trans. Syst. Man and Cyber., vol.23, pp.665-685, 1993.

[JAN-95] J.S.R.Jang, C.T.Sun, “Neuro-fuzzy modeling and control”, IEEE Proceeding, vol.83, no.3, pp.378-406, 1995.

[JAN-97] J.S.R.Jang, C.T.Sun, E.Mizutani, “Neuro-Fuzzy and soft computing”, Prentice Hall, ISBN 0-13-261066-3, 1997.

[JOR-90] M.I.Jordan and D.E.Rumelhart, “Forward models: Supervised learning with a distal teacher”, Cognitive Science, vol.16, pp.313–355, 1990.

[JUA-98] C-F.Juang, C-T.Lin, “An on-line self-constructing neural fuzzy inference network and its applications”, IEEE Trans. Fuzzy Systems, vol.6, no.1, February 1998.

[KIM-96] J.Kim, B.P.Zeigler, “Designing fuzzy logic controllers using a multiresolutional search paradigm”, IEEE Trans. Fuzzy Systems, vol.4, no.3, August 1996.

[KIM-97] D.Kim, C.Kim, “Forecasting time series with genetic fuzzy predictor ensemble”, IEEE Transactions on Fuzzy Systems, vol.5, no.4, pp.523-535, November 1997.

[KOS-92a] B.Kosko, “Neural Networks and Fuzzy Systems”, Englewood Cliffs, NJ, Prentice Hall, 1992.

[KOS-92b] B.Kosko, “Fuzzy function approximation”, IEEE Int. Conf. Fuzzy Systems, pp.209-213, 1992.

[KOS-94] B.Kosko, “Fuzzy systems as universal approximators”, IEEE Trans. Computers, vol.43, no.1, pp.1329-1333, Nov. 1994.

[KOV-92] B.Kovalerchuk, V.Taliansky, “Comparison of empirical and computed values of fuzzy conjunction”, Fuzzy Sets and Systems, vol.46, pp.49-53, 1992.

[KRU-94] R.Kruse, J.Gebhardt, F.Klawonn, “Foundations of fuzzy systems”, John Wiley & Sons, 1994.

[KWO-95] W.A.Kwong, K.M.Passino, E.G.Laukonen, S.Yurkovich, “Expert supervision of fuzzy learning systems for fault tolerant aircraft control”, IEEE Proceeding, vol.83, no.3, pp.466-483, 1995.

[LAN-96] R.Langari, L.Wang, “Complex systems modeling via fuzzy logic”, IEEE Trans. SMC, Part B, vol.26, pp.100-106, Feb. 1996.

[LAY-92] J.R.Layne and K.M.Passino, “Fuzzy Model Reference Learning Control”, Proceedings of IEEE on Control Applications, pp.686–691, 1992.

[LEE-90] C.C.Lee, “Fuzzy logic in control systems: fuzzy logic controller - Part I,II”, IEEE Trans. Syst. Man and Cyber., vol.20, no.2, pp.404-435, 1990.

[LEE-94] S-H.Lee, I.Kim, “Time series analysis using fuzzy learning”, in Proc. Int. Conf. Neural Inform. Processing, Seoul, Korea, pp.1577-1582, vol.6, Oct. 1994.

[LIN-91] C.Lin, C.S.G.Lee, “Neural-Network-Based fuzzy logic control and decision system”, IEEE Trans. Computers, vol.40, no.12, 1320-1336, 1991.

[LIN-96] C-J.Lin, C-T.Lin, “Reinforcement learning for an ART-Based fuzzy adaptive learning control network”, IEEE Trans. Neural Networks, vol.7, no.3, pp.709-731, 1996.

[LIN-97] S-C.Lin, Y-Y.Chen, “Design of self-learning fuzzy sliding mode controllers based on genetic algorithms”, Fuzzy Sets and Systems vol.86, pp.139-153, 1997.

[LOT-94] A.Lotfi, A.C.Tsoi, “Importance of membership functions: a comparative study on different learning methods for fuzzy inference systems”, in Proc. Third IEEE Int. Conf. Fuzzy Systems, Orlando (FL), pp.1791-1796, June 1994.

[LOT-96] A.Lotfi, H.C.Andersen, A.C.Tsoi, “Interpretation Preservation of Adaptive Fuzzy Inference Systems”, Int. J. Approximate Reasoning, vol.15, no.4, pp.379-394, November 1996.

[LYA-92] A.M.Lyapunov, A.T.Fuller (ed.). “The general problem of the stability of motion”, Taylor and Francis Ltd., 1992.

[MAB-93] S.Mabuchi, “A proposal for a defuzzification strategy by the concept of sensitivity analysis”, Fuzzy Sets and Systems, vol.55, pp.1-14, 1993.

[MAE-92] M.Maeda, S.Murakami, “A self-tuning fuzzy controller”, Fuzzy Sets and Systems, vol.51, pp.29-40, 1992.

[MAM-74] E.H.Mamdani, “Application on fuzzy algorithms for the control of a dynamic plant”, Proc. IEEE, vol.121, no.12, pp.1585-1588, 1974.

[MAM-75] E.H.Mamdani, S.Assilian, “An experiment in linguistic synthesis with a fuzzy logic controller”, International Journal Man-Machine Studies, vol.7, no.1, pp.1–12, 1975.

[MAX-68] J.C.Maxwell, “On Governors”, Proceedings of the Royal Society, vol.16, pp.270–283, 1868.

[MEN-95] J.M.Mendel, “Fuzzy logic systems for engineering”, IEEE Proceeding, vol.83, no.3, pp.345-377, 1995.

[MIT-93] K.Mitsubuchi, S.Isaka, Z.Y.Zhao, “A fuzzy rule generation system”, Proc. IFSA World Congress Seoul Korea, pp.11-14, 1993.

[MIZ-89] M.Mizumoto, “Pictorial representations of fuzzy connectives, Part I : cases of t-norms, t-conorms and averaging operators”, Fuzzy Sets and Systems, vol.31, pp.217-242, 1989.

[NGU-90] D.H.Nguyen and B.Widrow, “Neural Networks for Self-Learning Control Systems”, IEEE Control Systems Magazine, vol.10, no.4, pp.18–23, April 1990.

[NIE-96] Junhong H.Nie, T.H.Lee, “Rule-based modeling: fast construction and optimal manipulation”, IEEE Trans SMC, Part A, vol.26, no.6, November 1996.

[NOM-92] H.Nomura, I.Hayashi, N.Wakami, “A learning method of fuzzy inference rules by descent method”, Proc. of IEEE Int. Conf. on Fuzzy Systems, pp.203-210, 1992.

[NOZ-93] K.Nozaki, H.Ishibuchi, H.Tanaka, “Performance evaluation of fuzzy rule based classification methods”, IFSA, pp.167-170, 1993.

[OGA-93] K.Ogata, “Ingeniería de control moderno”, Prentice Hall, 1993.

[ORD-97] R.Ordóñez, J.Zumberge, J.T.Spooner, and K.M.Passino, “Adaptive Fuzzy Control: Experiments and Comparative Analyses”, IEEE Trans. Fuzzy Systems, vol.5, no.2, pp.167–188, May 1997.

[PAR-94] D.Park, A.Kandel, G.Langholz, “Genetic-Based new fuzzy reasoning models with application to fuzzy control”, IEEE Trans. Syst. Man and Cyber., vol.24, no.1, pp.39-47, 1994.

[PED-84] W.Pedrycz, “An identification algorithm in fuzzy relational systems”, Fuzzy Sets and Systems

13, pp.153-167, 1984.

[PIS-92] A.Piskunov, “Fuzzy implication in fuzzy systems control”, Fuzzy Sets and Systems, vol.45, pp.25-35, 1992.

[POM-99] **H.Pomares**, I.Rojas, F.J. Fernández, M.Anguita, E.Ros, A.Prieto, "A New Approach for the Design of Fuzzy Controllers in Real Time", Proceedings of the 8th International Conference on Fuzzy Systems (Fuzz-IEEE'99), pp.522-526, IEEE Service Center, NJ, USA., ISBN-0-7803-5406-0. 22-25 August 1999, Seoul, Korea.

[POM-00] **H.Pomares**, I.Rojas, J.Ortega, M.Anguita, A.Prieto, “A systematic approach to a self-generating fuzzy rule-table for function approximation”, IEEE Transactions on Systems, Man. and Cybernetics, Part B (aceptado SMC 099-08-B807 Nov. 99).

[PRE-93] W.H.Press, B.P.Flannery, S.A.Teukolsky, W.T.Vetterling, “Numerical Recipes in C”, Cambridge University Press, 1993.

[PRO-79] T.Procyk, E.Mamdani, “A linguistic self-organizing process controller”, Automatica, vol.15, no.1, pp.15-30, 1979.

[RED-92] P.V.S.Reddy, M.S.Babu, “Some methods of reasoning for fuzzy conditional propositions”, Fuzzy Sets and Systems, vol.52, 1992.

[RIS-78] J.Rissanen, “Modelling by shortest data description”, Automatica, vol.14, pp.464-471, 1978.

[ROJ-96] I.Rojas, “Desarrollo e implementación electrónica de sistemas difusos”, Tesis Doctoral, Universidad de Granada, 1996.

[ROJ-97] I.Rojas, **H.Pomares**, A.Prieto: “Nueva Metodología para el Desarrollo de Controladores Difusos Adaptativos y con Capacidad de Aprendizaje en Tiempo Real”, Seminario Anual de Automática, Electrónica Industrial e Instrumentación, SAAEI’97, Septiembre 13-15 1997.

[ROJ-98a] I.Rojas, **H.Pomares**, E.Ros, A.Prieto, “Automatic Construction of Fuzzy Rules and Membership Functions from Training Examples”, 6th European Congress on Intelligent Techniques and Soft Computing (EUROFIT’98), Aachen,Germany, September 7-10, 1998.

[ROJ-98b] I.Rojas, **H.Pomares**, E.Ros, A.Prieto, “Adaptive and Self-Learning Fuzzy Controllers in Real Time”, 6th European Congress on Intelligent Techniques and Soft Computing (EUROFIT’98), Aachen,Germany, September 7-10, 1998.

[ROJ-99a] I.Rojas, **H.Pomares**, F.J.Pelayo, M.Anguita, E.Ros, A.Prieto, “New methodology for the development of adaptive and self-learning fuzzy controllers in real time”, International Journal of Approximate Reasoning, vol.21, pp.109-136, 1999.

[ROJ-99b] Rojas, **H.Pomares**, J.L.Bernier, J.Ortega, E.Ros, A.Prieto, "Sequential Learning Algorithm for PG-RBF network using regression weights for Time Series Prediction"Lecture Notes in Computer Science, vol.1606, pp.631-640, ISSN: 0302-9743, Springer-Verlag, 1999. In: Foundations and Tools for Neural Modeling, J.Mira, Juan V.Sánchez-Andres (Eds) (5th International Work-Conference on Artificial and Natural Neural Networks (IWANN'99) Alicante, June 2-4, 1999).

[ROJ-99c] I.Rojas, **H.Pomares**, F.J.Pelayo, M.Anguita, E.Ros, A.Prieto, "New methodology for the development of adaptive and self-learning fuzzy controllers in real time", International Journal of Approximate Reasoning , vol.21, no.2, pp.109-136, 1999.

[ROJ-99d] I.Rojas, **H.Pomares**, F.J. Fernández, J.L. Bernier, F.J.Pelayo, A.Prieto, "A New Methodology to Obtain Fuzzy Systems Autonomously from Training Data", Proceedings of the 8th International Conference on Fuzzy Systems (Fuzz-IEEE'99), pp.527-532, IEEE Service Center, NJ, USA, ISBN-0-7803-5406-0. 22-25 August 1999, Seoul, Korea.

[ROJ-99e] I.Rojas, **H.Pomares**, F.Hoffman, F.J.Pelayo, A.Prieto, "New Design for an On-line Adaptive Fuzzy Controler", Proceedings of the 8th International Conference on Fuzzy Systems (Fuzz-IEEE'99), pp.1287-1292, IEEE Service Center, NJ, USA, ISBN-0-7803-5406-0. 22-25 August 1999, Seoul, Korea.

[ROJ-00] I.Rojas, **H.Pomares**, J.Ortega, A.Prieto, "Self-organized fuzzy system generation from training examples", IEEE Trans. Fuzzy Systems. (Nº 97095R accepted: 4 Oct. 1999).

[ROV-96] R.Rovatti, R.Guerrieri, “Fuzzy sets of rules for system identification”, IEEE Trans. Fuzzy Systems, vol.4, no.2, pp.89-102, 1996.

[RUS-69] E.H.Ruspini, “A new approach to Clustering”, Info Control no.15, pp.22-32, 1969.

[SAK-93] S.Sakaguchi, I.Skai, T.Haga, "Application of fuzzy logic to shift scheduling method for automatic transmission", Proc. 2nd IEEE Conf. on Fuzzy Syst., pp.52-58, 1993.

[SHA-88] S.Shao, "Fuzzy self-organizing controller and its applications for dynamic processes", Fuzzy Sets and Systems, vol.26, pp.151-164, 1988.

[SIN-98] Y.P.Singh, "A modified self-organizing controller for real-time process control applications", Fuzzy Sets and Systems, vol.96, pp.147-159, 1998.

[SON-93] B.G.Song et al., "Adaptive membership function fusion and annihilation in fuzzy if-then rules", Proc. 2nd IEEE Int. Conf. Fuzzy Systems, pp.961-967, San Francisco 1993.

[SUD-94] T.Sudkamp, R.J.Hammell, "Interpolation, Completion and Learning Fuzzy Rules", IEEE Trans. Syst. Man and Cyber., vol.24, no.2, pp.332-342, February 1994.

[SUG-85a] M.Sugeno, "Industrial applications of fuzzy control", Amsterdam, North-Holland, 1985.

[SUG-85b] M.Sugeno, "An introductory survey of fuzzy control", Information Sciences, vol.36, pp.59-83, 1985.

[SUG-88a] M.Sugeno, G.Kang "Structure identification of fuzzy model", Fuzzy Sets and Systems vol.28, pp.15-33, 1988.

[SUG-88b] K.Sugiyama, "Rule-based self-organizing controller", in M.M.Gupta and T.Yamakawa, Eds., Fuzzy Computing (Elsevier, Amsterdam), 1988.

[SUG-91] M.Sugeno, K.Tanaka, "Successive identification of a fuzzy model and its applications to prediction of a complex system", Fuzzy Sets and Syst., vol42, pp.315-334, 1991.

[SUG-93] M.Sugeno, T.Yasukawa, "A fuzzy-logic based approach to qualitative modeling", IEEE Trans. Fuzzy Syst., vol.1, no.1, Feb. 1993.

[TAK-83] T.Takagi, M.Sugeno, "Derivation of fuzzy control rules from human operators actions", Proc. IFAC Symp. Fuzzy Inf., Marseille, France, pp.55-60, 1983.

[TAK-85] T.Takagi, M.Sugeno, "Fuzzy identification of systems and its applications to modelling and control", IEEE Trans. Syst. Man and Cyber., vol.15, pp.116-132, 1985.

[TAN-92] J.Tanomaru, S.Omatu, “Process control by on-line trained neural controllers”, IEEE Transactions on Industrial Electronics, vol.39, no.6, pp.511–521, December 1992.

[TAN-93] H.Tanaka, K.Yokode, H.Ishibuchi, “GMDH by fuzzy if-then rules with certainty factors”, Fifth IFSA World Congress, pp.802-805, 1993.

[TER-92] T.Terano, K.Asai, M.Sugeno, “Fuzzy systems theory and its applications”, Academic Press, 1992.

[TON-78] R.M.Tong, “Synthesis of fuzzy models for industrial processes”, Int. J. General Systems 4, pp.143-162, 1978.

[TON-79] R.M.Tong, “The construction and evaluation of fuzzy models”, Advances in Fuzzy Set Theory and Applications, M.M.Gupta, R.K.Ragade, R.R.Yager, eds. Amsterdam, The Netherlands: North-Holland, pp.559-576, 1979.

[WAN-91] L.X.Wang, J.M.Mendel, “Generating fuzzy rules by learning from examples”, Proc. 6th IEEE Int. Symp. Intell. Contr., pp.263-268, 1991.

[WAN-92a] L.X.Wang, J.M.Mendel, “Fuzzy basis functions, universal approximation, and orthogonal least squares learning”, IEEE Trans. Neural Networks, vol.3, pp.807-813, Sep.1992.

[WAN-92b] L.X.Wang, J.M.Mendel, “Generating fuzzy rules by learning from examples”, IEEE Trans. Syst. Man and Cyber, vol.22, no.6, pp.1414-1427, Nov. 1992.

[WAN-92c] L.X.Wang, J.M.Mendel, “Back-propagation fuzzy system as nonlinear dynamic system identifiers”, Proc. IEEE Int. Conf. Fuzzy Systems, pp.1409-1418, San Diego, 1992.

[WAN-94] L.X.Wang, “Adaptive fuzzy systems and control. Design and stability analysis”, Prentice Hall, 1994.

[WAN-95] C-H.Wang, W-Y.Wang, T-T.Lee, P-S.Tseng, “Fuzzy B-Spline membership function (BMF) and its application in fuzzy-neural control”, IEEE Trans. Syst., Man and Cyber.-Part B, vol.25, no.5, 1995.

[WAN-97] W-Y.Wang, T-T.Lee, C-L.Liu, C-H.Wang, “Function approximation using fuzzy neural networks with robust learning algorithm”, IEEE Trans. Syst., Man and Cyber.-Part B, vol.27, no.4, Aug. 1997.

[WER-90] P.J.Werbos, “Overview of Design and Capabilities”, In Neural Networks for Control, pp.59–65. MIT Press, MA, USA, 1990.

[WHI-96] B.A.Whitehead, T.D.Choate, “Cooperative-Competitive Genetic Evolution of Radial Basis Function Centers and Widths for Time Series Prediction”, IEEE Transaction on Neural Networks, vol.7, no.4, pp.869-880, July 1996.

[XU-87] C.Xu, Y.Lu, “Fuzzy model identification and self learning for dynamic systems”, IEEE Trans. Syst., Man and Cyber., vol.17, pp.683-689, 1987.

[YAG-85] O.Yagishita, O.Itoh, M.Sugeno, “Application of fuzzy reasoning to the water purification process”, Industrial Applications of Fuzzy Control, M.Sugeno Ed. Amsterdam: North Holland, pp.19-40, 1985.

[YAG-93] R.R.Yager, D.Filev, “On the issue of defuzzification and selection based on a fuzzy set”, Fuzzy Sets and Systems, vol.55, pp.255-271, 1993.

[YAS-85] S.Yasunobu, S.Miyamoto, “Automatic train operation by predictive fuzzy control”, Industrial Application of Fuzzy Control, M.Sugeno Ed. Amsterdam: North Holland, pp.1-18, 1985.

[ZAD-65] L.A.Zadeh, “Fuzzy Sets”, Information and Control, vol.8, pp.338–353, 1965.

[ZAD-73] L.A.Zadeh, “Outline of a new approach to the analysis of complex systems and decision processes”, IEEE Trans. Syst., Man and Cyber., vol.3, pp.28-44, 1973.

[ZHA-91] R.Zhao, R.Govind, “Defuzzification of fuzzy intervals”, Fuzzy Sets and Systems, vol.43, pp.45-55, 1991.