

Universidad de Huelva

**Departamento de Ingeniería Electrónica, de Sistemas y
Automática**



Síntesis de sistemas de control borroso estables por diseño

**Memoria para optar al grado de doctor
presentada por:**

Antonio Javier Barragán Piña

Fecha de lectura: 7 de julio de 2009

Bajo la dirección del doctor:

José Manuel Andújar Márquez

Huelva, 2010

ISBN: 978-84-92944-72-9

D.L.: H 34-2010



**Universidad
de Huelva**

Departamento de Ingeniería Electrónica, de Sistemas
Informáticos y Automática

Escuela Politécnica Superior

SÍNTESIS DE SISTEMAS DE CONTROL BORROSO ESTABLES POR DISEÑO

Tesis Doctoral de
Antonio Javier Barragán Piña

Dirigida por
José Manuel Andújar Márquez

UNIVERSIDAD DE HUELVA
MAYO 2009

D. José Manuel Andújar Márquez, Catedrático de Escuela Universitaria de la Escuela Politécnica Superior de la Universidad de Huelva,

CERTIFICA:

Que D. Antonio Javier Barragán Piña, Ingeniero en Automática y Electrónica Industrial, ha realizado, bajo mi dirección, y dentro del programa de doctorado **Ingeniería básica y aplicada** el trabajo de investigación correspondiente a su Tesis Doctoral titulado:

Síntesis de Sistemas de Control Borroso Estables por Diseño

Revisado el presente trabajo, estimo que puede ser presentado al Tribunal que ha de juzgarlo.

Y para que conste a efectos de lo establecido en el artículo octavo del Real Decreto 778/1998 y por la Normativa Reguladora del título de Doctor de la Universidad de Huelva, autorizo la presentación de este trabajo en la Universidad de Huelva.

Huelva, a 8 de mayo de 2009



Director:

Dr. José Manuel Andújar Márquez



Dortorando:

Antonio Javier Barragán Piña

A mi esposa Yolanda por el tiempo que hemos perdido de estar juntos, a mis padres, sin cuyo esfuerzo me habría sido imposible llegar hasta aquí, y a mis hermanos por su cariño y apoyo constante.



Agradecimientos

El autor quiere manifestar su más profundo agradecimiento al Dr. José Manuel Andújar Márquez por su dedicación constante a la dirección de esta Tesis, y sin cuyo esfuerzo habría sido imposible su culminación.

En el mismo sentido, se desea reconocer la inestimable ayuda prestada por el Dr. Manuel Emilio Gegúndez Árias, quien nunca ha escatimado dedicar su tiempo para asistir a este doctorando.

El autor también desea agradecer a todos sus compañeros y amigos el apoyo y el ánimo constante que ha recibido.

Índice general

1. Planteamiento general de la Tesis	1
1.1. Resumen de la Tesis	2
1.2. Innovaciones que aporta la Tesis	5
1.3. Rendimiento científico de la Tesis	8
2. Introducción a la lógica borrosa y sus aplicaciones	13
2.1. Introducción	13
2.2. Breve historia de la lógica borrosa	15
2.3. Fundamentos de los sistemas borrosos	19
2.3.1. Conjuntos borrosos y funciones de pertenencia	19
2.3.2. Variables lingüísticas	46
2.3.3. Regla borrosa	48
2.3.4. Sistema de inferencia borroso	49
2.3.5. Tipos de sistemas borrosos	54
2.4. Conclusiones y comentarios bibliográficos	60
3. Modelado borroso de sistemas dinámicos	61
3.1. Introducción	61
3.1.1. Modelo de estado de un sistema dinámico	63
3.2. Modelo de estado de un sistema de control borroso	65
3.2.1. Sistema borroso de tipo TSK	66
3.2.2. Sistema borroso de tipo Mamdani	78
3.3. Identificación de modelos borrosos	79
3.3.1. Identificación de la estructura	80
3.3.2. Estimación de parámetros	83
3.3.3. Validación	90
3.4. Diseño del software	91
3.5. Conclusiones y comentarios bibliográficos	97

4. Análisis de sistemas dinámicos mediante modelos borrosos	99
4.1. Introducción	100
4.2. Linealización de un modelo borroso de estado	101
4.3. Matriz jacobiana de un sistema de control borroso	110
4.4. Estados de equilibrio de un sistema de control borroso	122
4.4.1. Métodos numéricos para la minimización de funciones	123
4.5. Diseño del software	126
4.6. Conclusiones y comentarios bibliográficos	128
5. Diseño de controladores borrosos	131
5.1. Introducción	131
5.2. Teoría de estabilidad de Lyapunov	133
5.2.1. Método directo de Lyapunov	135
5.2.2. Método de Krasovskii	137
5.3. Control heurístico de sistemas dinámicos	138
5.4. Síntesis de controladores borrosos estables por diseño	140
5.4.1. Identificación de la planta	140
5.4.2. Estudio dinámico de la planta	141
5.4.3. Diseño del controlador borroso	141
5.4.4. Algoritmo de diseño	147
5.5. Conclusiones y comentarios bibliográficos	152
6. Aplicaciones	155
6.1. Control heurístico	155
6.1.1. Control borroso basado en heurística de una grúa porta contenedores	156
6.2. Síntesis de controladores borrosos estables por diseño	168
6.2.1. Ejemplo ilustrativo	169
6.2.2. Péndulo	189
6.2.3. Balancín	202
6.3. Conclusiones	231
7. Conclusiones y desarrollos futuros	233
7.1. Conclusiones	234
7.2. Desarrollos futuros	236
A. Leyenda de índices y variables	239
B. Derivadas de las principales funciones de pertenencia	243
C. Descomposición de Cholesky	249

D. Definición del software	251
D.1. Clase Membership	251
D.2. Clase Rule	255
D.3. Clase System	256
D.4. Clase Jacobian	258
D.5. Utilidades	259
D.6. Otros archivos	261
Índice de figuras	269
Índice de tablas	273
Índice alfabético	275
Bibliografía	281

Planteamiento general de la Tesis

EL objetivo fundamental de esta Tesis es establecer una metodología de diseño de controladores borrosos lo más general posible, de forma que se garantice la estabilidad del sistema de control en lazo cerrado. De igual forma, se desea contribuir a la formalización de los sistemas borrosos con herramientas que permitan el análisis de estos sistemas según la teoría de control no lineal aceptada por la comunidad científica.

Este texto está concebido como parte de un trabajo perteneciente al Grupo de Investigación de Control y Robótica (TEP192) en el que se ha desarrollado, por lo que lejos de pretender ser un trabajo aislado, persigue mantener la sinergia dentro de dicho grupo de investigación. En este equipo, y bajo la dirección del Dr. José Manuel Andújar Márquez, se están realizando varias tesis doctorales de forma simultánea cuyas líneas generales de trabajo son, entre otras, la lógica borrosa, el control no lineal y la aplicación práctica de estas herramientas a sistemas reales.

En esta Tesis se estudian fundamentalmente los sistemas borrosos de tipo Takagi-Sugeno-Kang (TSK), ya que, como se verá a lo largo del texto, éstos permiten la representación de cualquier sistema no lineal mediante un número reducido de reglas. Aunque también se estudian otros sistemas borrosos, tan sólo se hace con carácter introductorio, ya que todo el trabajo de la Tesis se basa en los sistemas borrosos de tipo TSK. El grupo de investigación en el que se ha desarrollado esta Tesis tiene una amplia experiencia en el estudio de los sistemas borrosos, y cuenta con bastantes publicaciones en revistas internacionales de primer nivel y en congresos tanto nacionales como internacionales.

Este primer capítulo está estructurado en tres secciones. En la primera sección se presentará un resumen de la Tesis, donde se recorren uno a uno los capítulos que la conforman para dar una idea general de su contenido. En la segunda sección se vuelven a repasar cada uno de los capítulos del texto, pero enfatizando en esta ocasión las aportaciones novedosas que cada uno de ellos aporta al estado del arte. Finalmente, en la última sección se muestra el rendimiento científico de la Tesis, el cual supone una validación del trabajo realizado por parte de la comunidad científica tanto nacional como internacional, así como una muestra clara del grado de innovación de la misma.

1.1. Resumen de la Tesis

Esta memoria está organizada en siete capítulos y cuatro apéndices, cuyo contenido se describe brevemente a continuación:

En el **Capítulo 1**, *Planteamiento general de la Tesis*, como su propio nombre indica, se realiza una descripción de la estructura de la Tesis y se comentan brevemente cada uno de los capítulos que la componen. En este capítulo también se recogen las innovaciones principales de la Tesis y se efectúa un análisis del rendimiento científico que este trabajo ha producido hasta la fecha.

El **Capítulo 2**, *Introducción a la lógica borrosa y sus aplicaciones*, realiza una breve introducción a la lógica borrosa y sus aplicaciones a través un recorrido histórico, marcando los hitos principales y realizando especial hincapié en las aplicaciones de control. Posteriormente se estudian los fundamentos de los sistemas borrosos. Este estudio se realiza con la intención de que el texto sea autocontenido y no se requiera acudir a otros textos para comprender los mecanismos que se utilizarán en los capítulos siguientes, no obstante, se proporcionan abundantes referencias que permiten ampliar dichos contenidos. Esta sección comienza, como no puede ser de otra forma, con la definición de conjunto borroso y las funciones de pertenencia como mecanismo de definición de los mismos, así como algunas propiedades de éstos. Posteriormente se presentan las operaciones básicas con conjuntos borrosos: unión, intersección, negación, . . . , y se define el concepto de variable lingüística. A partir de los conocimientos anteriores ya es posible dar el paso a la construcción de reglas borrosas y al estudio de los distintos tipos de sistemas borrosos existentes junto con sus métodos de inferencia. De esta forma se da por concluido el estudio de los fundamentos de la lógica borrosa.

El **Capítulo 3**, *Modelado borroso de sistemas dinámicos*, afronta la necesidad de obtener un modelo de la planta para poder así estudiar el sistema de manera formal, siempre desde el punto de vista de la lógica borrosa.

Con objeto de poder aplicar las técnicas aceptadas por la comunidad científica para el estudio de sistemas no lineales, el modelado borroso se estudia desde la perspectiva de los modelos de estado. Así se define el modelo borroso de estado como una sucesión de reglas que describen la variación del vector de estado en función del propio vector de estado y del vector de control. Aunque en este capítulo se presentan tanto los sistemas borrosos de tipo Mamdani como los de tipo TSK, el estudio se centra en los sistemas de tipo TSK, ya que en ellos están basados el resto de capítulos de esta Tesis. Una vez definido el modelo borroso de estado de la planta, se obtiene su modelo matemático equivalente y se propone una extensión del vector de estado para simplificar la representación de la ecuación obtenida.

A lo largo del texto no sólo se utiliza el modelo borroso de la planta, si no que será necesario la utilización de un controlador borroso y del sistema formado por la planta y el controlador, es decir, del sistema borroso en lazo cerrado o realimentado. Por lo tanto, se procede al estudio tanto del modelo borroso del controlador como del modelo del sistema en lazo cerrado, obteniéndose en ambos casos el modelo matemático equivalente del sistema objeto de estudio. En ambos modelos matemáticos se utiliza la extensión del vector de estado propuesta para simplificar la representación de las ecuaciones obtenidas.

Para completar este apartado del capítulo 3 se proponen unos algoritmos que permiten el cálculo de los modelos matemáticos en cuestión.

Llegados a este punto del capítulo, se ha visto cómo se puede representar tanto una planta como un controlador desde el punto de vista de la lógica borrosa, y se han obtenido los modelos matemáticos equivalentes a dichas representaciones; sin embargo, aún no se ha visto cómo obtener el modelo borroso de un sistema. En la sección 3.3 se estudia precisamente esto, la identificación de un sistema a partir de datos de entrada/salida mediante la aplicación de modelos borrosos. Esta sección está dividida en tres subsecciones donde se estudian cada una de las fases del proceso de modelado, a saber: identificación de la estructura, identificación de los parámetros y validación del modelo. En la fase de identificación de la estructura se estudian las técnicas de *clustering*, particionamiento uniforme y particionamiento experto; en la fase de identificación de los parámetros se presenta la metodología neuroborrosa y se propone una nueva metodología híbrida bioinspirada y neuroborrosa; finalmente, se estudian distintas técnicas de validación.

Antes de concluir el capítulo 3 se presenta, mediante clases UML, el software diseñado para almacenar modelos borrosos generales de tipo TSK, así como una serie de funciones adicionales que facilitan el tratamiento de dichos sistemas.

El **Capítulo 4**, *Análisis de sistemas dinámicos mediante modelos borrosos*, está dedicado a la extracción de información de un sistema real a partir de un modelo borroso de éste. Tras una breve introducción, se resuelve en primer lugar la linealización de un modelo borroso de estado completamente general, sin limitación en el número de reglas de éste, ni en el tamaño de los vectores de estado y control, ni en el tipo de función de pertenencia utilizada para cada uno de los conjuntos borrosos del modelo, ya que incluso se permite la mezcla de distintas funciones de pertenencia en un mismo modelo. Hay que tener en cuenta que esta linealización se hace de forma completa, es decir, sin eliminar ni simplificar ningún elemento del sistema borroso.

La linealización de un modelo borroso puede ser utilizada para extraer información importante del sistema real al que representa, como por ejemplo la estabilidad local de algún estado de equilibrio. Lo mismo sucede con el sistema en lazo cerrado. Poseer el modelo linealizado de un sistema de control borroso en lazo cerrado permite estudiar dicho sistema y obtener cierta información de su comportamiento. Así pues, el siguiente paso a resolver en este capítulo es la obtención de la matriz jacobiana de un sistema de control borroso completamente general (entiéndase general en los mismos términos anteriores). Realimentando los modelos matemáticos equivalentes de la planta y del controlador obtenidos en el capítulo anterior, y procediendo a calcular su derivada con respecto a cada una de las variables de estado, se obtiene la matriz jacobiana buscada.

La obtención de los estados de equilibrio de un sistema proporciona una valiosa información sobre el comportamiento de éste, especialmente si se trata de un sistema no lineal. En la sección 4.4 se aborda la obtención de los estados de equilibrio de un sistema borroso mediante una metodología basada en métodos numéricos clásicos, tanto para el modelo borroso de la planta como para el del sistema de control borroso en lazo cerrado. En base a las ecuaciones utilizadas por los métodos numéricos de minimización, se propone la

utilización de la matriz jacobiana calculada en este mismo capítulo tanto para acelerar la convergencia de los algoritmos numéricos como para mejorar su precisión.

Por último, el capítulo 4 concluye con la propuesta de unos algoritmos para la realización del cálculo del sistema borroso linealizado y de la matriz jacobiana del sistema borroso en lazo cerrado.

El **Capítulo 5**, *Diseño de controladores borrosos*, está enfocado, como su propio nombre indica, al diseño de controladores borrosos. En primer lugar se realiza una breve revisión histórica, centrada en el ámbito del control, del proceso de formalización de la lógica borrosa realizado por la comunidad científica.

La teoría de estabilidad de Lyapunov es sin duda alguna la herramienta más utilizada para el estudio y diseño de sistemas de control no lineales, y posiblemente también una de las más flexibles. Por todo ello, el principio de este capítulo se dedica a su estudio como herramienta necesaria en los desarrollos posteriores. También se estudia el método de Krasovskii como una forma concreta de aplicación de la teoría de Lyapunov al estudio y diseño de sistema de control no lineales.

El diseño de controladores borrosos puede realizarse fundamentalmente desde dos enfoques: 1) heurístico, a partir del conocimiento de un operario experto, y 2) formal, mediante un estudio de estabilidad del sistema de control. En este capítulo se estudian ambos enfoques. El enfoque heurístico permite diseñar una solución mediante reglas empíricas basadas en el conocimiento de un experto, mientras que el enfoque basado en la teoría de control no lineal permite garantizar formalmente la estabilidad del sistema de control resultante.

Uno de los objetivos fundamentales de esta Tesis es establecer una metodología lo más general posible que permita la síntesis de controladores borrosos estables por diseño, es decir, que la estabilidad quede garantizada durante el propio proceso de diseño del controlador. En este capítulo se propone una metodología de diseño general basada en tres pasos: identificación de la planta y representación de la misma en forma de modelo borroso de estado, estudio de ésta a partir del modelo borroso obtenido, y diseño de un controlador borroso que garantice la estabilidad asintótica del sistema en lazo cerrado. Llegados a este punto ya se han estudiado los dos primeros pasos de esta metodología. Para posibilitar la realización del tercer punto se propone un nuevo teorema de estabilidad basado en la teoría de Lyapunov, e inspirado en el teorema de Krasovskii, a partir del cual se pueden diseñar controladores borrosos estables por diseño. Finalmente, el capítulo concluye con una propuesta de algoritmo de diseño para la síntesis de controladores borrosos estables en base al nuevo teorema propuesto.

En el **Capítulo 6**, *Aplicaciones*, se aborda el análisis y diseño de sistemas de control borroso a partir de las técnicas presentadas en los capítulos anteriores. En primer lugar se diseña un controlador borroso para un sistema complejo, una grúa porta contenedores, a partir del conocimiento de un experto. A continuación se muestra la metodología de diseño propuesta en el capítulo 5 mediante tres ejemplos, para lo cual es necesario realizar: 1) identificación de la planta mediante un modelo borroso de estado según lo visto en el capítulo 3; 2) análisis de la planta mediante la extracción de sus estados de equilibrio y estudio de la estabilidad local de los mismos a partir de su modelo borroso, según lo visto en el capítulo 4; y 3) diseño de un controlador borroso que estabilice la planta, a partir

del algoritmo de diseño propuesto en el capítulo 5. En todos los casos, una vez diseñado el controlador se realiza una simulación para comprobar el correcto funcionamiento del mismo, junto con un análisis de la estabilidad local del estado de equilibrio del sistema de lazo cerrado resultante. Para ello se emplean nuevamente los algoritmos propuestos en el capítulo 4.

El **Capítulo 7**, *Conclusiones y desarrollos futuros*, incide en las conclusiones que se derivan de la Tesis. A continuación, se esbozan los temas que, sustentados en desarrollos y propuestas de la Tesis, dejan el camino abierto para ser resueltos en trabajos futuros.

El **Apéndice A**, *Leyenda de índices y variables*, recoge una tabla con todas las variables utilizadas en la Tesis, incluyendo una breve explicación del significado de cada una de ellas y del contexto en el que se utilizan.

En el **Apéndice B**, *Derivadas de las principales funciones de pertenencia*, se calculan las derivadas de todas las funciones de pertenencia presentadas en el capítulo 1. Estas derivadas serán necesarias en tiempo de ejecución para calcular las aproximaciones lineales de los modelos borrosos de lazo abierto y de lazo cerrado.

En el **Apéndice C**, *Descomposición de Cholesky*, se explica la descomposición de Cholesky, la cual permite la transformar una matriz simétrica definida positiva en el producto de una matriz triangular y su traspuesta, donde se cumple que todos los elementos de la diagonal de dicha matriz triangular son positivos. La descomposición de Cholesky también puede ser utilizada en sentido inverso, es decir, partiendo de una matriz triangular con valores positivos en los elementos de su de la diagonal, se obtiene una matriz simétrica definida positiva. Esta descomposición es parte del algoritmo de diseño de controladores propuesto en el capítulo 5.

En el **Apéndice D**, *Definición del software*, se incluyen las cabeceras (archivos .h) del software diseñado en los capítulos anteriores. Estos ficheros, junto con los diagramas UML realizados, resultan imprescindibles para poder utilizar el software diseñado en el desarrollo de nuevas funciones basadas en modelos borrosos.

1.2. Innovaciones que aporta la Tesis

En esta sección se ordenan por capítulos los aspectos innovadores que pretende incorporar este trabajo al campo de los sistemas de control borroso.

El **Capítulo 2**, *Introducción a la lógica borrosa y sus aplicaciones*, está concebido como una revisión histórica del uso de la lógica borrosa en sistemas de control. A lo largo de este capítulo se ve como las reticencias iniciales de parte de la comunidad científica sobre la lógica borrosa acabaron por desaparecer. Esto se ha debido fundamentalmente al proceso de formalización sufrido y a los excelentes resultados prácticos que ésta ha obtenido tanto en el ambiente académico e investigador como en el industrial. En este capítulo también se realiza una introducción a la lógica borrosa con objeto de conseguir que el texto sea autocontenido y se dispongan en él de todos los fundamentos sobre lógica borrosa que puedan ser necesarios en los capítulos posteriores.

El **Capítulo 3**, *Modelado borroso de sistemas dinámicos*, está centrado en el proceso de modelado de sistemas dinámicos mediante modelos borrosos. En este capítulo se presenta

un modelo borroso en forma de ecuaciones de estado de un sistema de control multivariable no lineal completamente general. Para ello no se aplica ningún tipo de restricciones ni al tamaño del vector de estado, ni al del vector de control, ni al tipo de función de pertenencia utilizada en la definición de los conjuntos borrosos. Para aumentar la generalidad de la propuesta, se considera que el controlador es completamente independiente del proceso de identificación del modelo borroso de la planta, es decir, que la partición del universo de discurso de las variables de estado de la planta y del controlador borroso no tienen porqué coincidir. De la misma forma, los modelos borrosos de la planta y del controlador no tienen porqué tener el mismo número de reglas.

Con objeto de poder simplificar¹ la representación de las ecuaciones anteriores, se propone la extensión del vector de estado en una componente. Con el nuevo vector de estado extendido la representación de las ecuaciones se muestra más compacta, lo que facilita los cálculos futuros y propicia una programación más rápida y sencilla de las mismas.

Respecto a la identificación del modelo borroso de la planta, en este capítulo se estudia la metodología neuroborrosa, y se propone una nueva técnica, denominada híbrida, basada en la hibridación de la citada metodología neuroborrosa y la heurística bioinspirada. Esta nueva técnica está aún en una fase de desarrollo muy primaria, aunque ha demostrado ser bastante prometedora.

Finalmente, se concluye con la propuesta de estructura software que permite almacenar todos los datos de un sistema de control borroso completamente general como el que se estudia a lo largo del capítulo, así como de las herramientas necesarias para su tratamiento.

En el **Capítulo 4**, *Análisis de sistemas dinámicos mediante modelos borrosos*, se calcula el desarrollo de Taylor de orden uno del modelo matemático equivalente obtenido en el capítulo anterior, de esta forma se obtiene la linealización de un modelo borroso de estado completamente general. Esta linealización se realiza sin limitación en el número de reglas de modelo borroso. Tampoco se limita el tamaño de los vectores de estado y control, ni el tipo de función de pertenencia utilizada para cada uno de los conjuntos borrosos del modelo; ya que incluso es posible mezclar distintas funciones de pertenencia en una misma regla. Es importante aclarar que el modelo linealizado se calcula derivando el modelo borroso de estado completo, sin hacer ningún tipo de simplificación sobre el mismo.

El siguiente aspecto desarrollado en este capítulo es la linealización de un sistema de control borroso completamente general, es decir, la obtención de la matriz jacobiana del sistema resultante al realimentar el modelo borroso de la planta con el del controlador.

Tanto la linealización del modelo borroso de la planta como la del sistema de control borroso pueden utilizarse, como se ve en el capítulo 6, para analizar los sistemas no lineales originales, por ejemplo, para estudiar la estabilidad local de sus estados de equilibrio.

Otro aspecto importante a la hora de analizar un sistema no lineal son sus estados de equilibrio. En el capítulo 4 se propone un algoritmo numérico basado en algoritmos clásicos de minimización para la obtención de los estados de equilibrio tanto de un modelo borroso como de un sistema de control borroso. En ambos casos se propone utilizar los modelos

¹Nótese en la expresión (3.20) que en ningún caso se está haciendo una simplificación de las ecuaciones, si no que se modifica la representación de la misma con objeto de que éstas se muestren más compactas.

linealizados calculados con anterioridad para así mejorar la velocidad de convergencia y la precisión de los algoritmos de minimización.

Finalmente, el capítulo concluye con la propuesta de unos algoritmos que permiten realizar las linealizaciones anteriores. Estos algoritmos han sido implementados en C/C++ junto con los diagramas de clases propuestos en este capítulo y en el capítulo anterior. El trabajo de programación realizado supera las 4000 líneas de código y permite ser utilizado tanto en C++ como en MATLAB®, para ello se han programado unas funciones adicionales, descritas en la figura 3.20, que hacen de interfaz entre el software desarrollado y MATLAB®.

En el **Capítulo 5**, *Diseño de controladores borrosos*, se estudia el diseño de controladores borrosos desde dos enfoques: diseño a partir de reglas lingüísticas obtenidas del conocimiento de un experto, y diseño basado en la teoría de control no lineal. En este capítulo se propone una metodología completamente general para la síntesis de sistemas de control borroso estables por diseño, esto es, donde la estabilidad del sistema en lazo cerrado quede garantizada durante el propio diseño del controlador. Esta metodología consta de 3 pasos: 1) obtención de un modelo borroso de la planta a partir de datos de entrada/salida de la misma; 2) estudio de la dinámica de la planta a través de su modelo borroso mediante el cálculo de sus estados de equilibrio y el estudio de la estabilidad local de éstos; y 3) diseño de un controlador borroso mediante la aplicación del teorema de estabilidad propuesto. Los dos primeros puntos se resuelven en capítulos anteriores, mientras que para resolver el tercero de ellos se enuncia un nuevo teorema basado en la teoría de Lyapunov, que inspirado en el teorema de Krasovskii, permite garantizar la estabilidad asintótica del sistema de control. Con objeto de llevar a la práctica esta metodología de diseño, se presenta un algoritmo que genera un controlador borroso inicial a partir del modelo borroso de la planta y, posteriormente, ajusta los parámetros adaptables de los consecuentes hasta conseguir la estabilidad asintótica del sistema en lazo cerrado.

Como mecanismo de ajuste del controlador borroso se escoge una metodología de búsqueda mixta global-local. Este ajuste consta de una fase de búsqueda global mediante algoritmos genéticos, seguida de una fase de búsqueda local mediante algoritmos numéricos. Con esta combinación se consigue complementar la capacidad de búsqueda de los algoritmos genéticos con la velocidad y precisión de los algoritmos numéricos.

El **Capítulo 6**, *Aplicaciones*, aborda el análisis y diseño de sistemas de control borroso en base a las técnicas presentadas en los capítulos anteriores. Así, en este capítulo se realiza el control de un sistema multivariable y altamente no lineal como es una grúa porta contenedores, a partir del conocimiento extraído de un experto. Posteriormente se resuelven tres casos de diseño mediante la metodología propuesta en el capítulo 5. Estos casos son: un ejemplo ilustrativo múltiple entrada múltiple salida (MIMO), cuyo objetivo es facilitar la comprensión de la metodología de diseño propuesta; un péndulo, como sistema no lineal con múltiples estados de equilibrio, que permite validar la fase de análisis y diseño de la metodología propuesta; y, finalmente, un sistema balancín, que representa un ejemplo de uso común en la literatura especializada en control no lineal.

En todos los casos, una vez diseñado el controlador borroso se comprueba su correcto funcionamiento mediante simulación. Para ello se simula el sistema realimentado com-

puesto por el modelo real de la planta y el controlador borroso diseñado. De esta forma se garantiza que el controlador no sólo estabiliza el modelo borroso de la planta, si no también la planta real. Mediante la ejecución de una nueva etapa de análisis de los estados de equilibrio, esta vez para los sistemas realimentados, se comprueba que éstos posean tan sólo un equilibrio dentro del universo de discurso de las variables de estado y que éste sea asintóticamente estable.

Mediante las aplicaciones resueltas en este capítulo se comprueba de forma práctica la metodología de diseño propuesta, así como los mecanismos de análisis de sistemas borrosos desarrollados. Concretamente, analizando los estados de equilibrio obtenidos para cada una de las plantas propuestas mediante su modelo borroso, y comparándolos con los estados de equilibrio reales, se puede comprobar que el resultado es adecuado. Además, si se comparan también los autovalores de los sistemas borrosos linealizados y los de los sistemas reales, también linealizados, se puede comprobar que sus valores son bastante cercanos. Esto es, es posible obtener los estados de equilibrio de un sistema real, y estudiar la estabilidad local de éstos, a partir de un modelo borroso obtenido tan sólo mediante un conjunto de datos de entrada/salida, y sin información previa de la planta.

En conclusión, esta Tesis aporta una metodología completa de modelado, análisis y síntesis de sistemas de control borrosos, novedosa en muchos aspectos, y cuyo desarrollo se estudia paso a paso a lo largo de toda la memoria. Además, en cada capítulo se proporciona tanto la estructura del software como los algoritmos necesarios para su implementación práctica. Un aspecto importante de la metodología desarrollada es que, lejos de ser una metodología cerrada y conclusa, permite su ampliación mediante la elaboración de nuevos métodos de análisis y/o diseño extraídos de la teoría de control no lineal. Por ejemplo, cabría la posibilidad añadir una metodología para analizar la dinámica de sistemas parametrizados y con múltiples equilibrios mediante un modelo borroso del mismo. De esta forma sería posible estudiar la aparición de bifurcaciones y ciclos límite en plantas cuya dinámica sea a priori desconocida.

1.3. Rendimiento científico de la Tesis

Esta sección está dedicada a enmarcar el entorno científico en el que se ha desarrollado la Tesis y a destacar los resultados de la investigación realizada; para ello se detallarán las publicaciones con índice de impacto así como otras de interés. También, por supuesto, dentro del entorno científico de la Tesis, se hará hincapié en los hitos relevantes en su andadura. En definitiva, el objetivo principal de esta sección es mostrar que la memoria que se presenta constituye la recopilación ordenada e hilvanada de una producción científica realizada durante 5 años.

Beca Predoctoral Campus de Excelencia 2005

Beca obtenida en convocatoria competitiva evaluada por la ANEP/ANECA en función del currículum del candidato y el Proyecto de Tesis presentado.

Producción científica por capítulos

Capítulo 3 *Modelado borroso de sistemas dinámicos*

La Tesis comenzó su andadura propiamente dicha, con un gran esfuerzo en el estudio del proceso de modelado de sistemas dinámicos mediante modelos borrosos. Como primer resultado importante, se desarrolló una metodología completa que permite afrontar el modelado borroso, en forma de ecuaciones de estado, de un sistema de control multivariable no lineal. Este paso era y fue muy importante a lo largo de todo el proceso de desarrollo de la Tesis, ya que permitió representar el modelo borroso en el mismo lenguaje utilizado por la comunidad científica que trabajaba en los sistemas de control no lineal, desde un punto de vista, podríamos decir, clásico (no borroso). A partir de aquí se abría la puerta para poder emplear una formalización, ya aceptada por la comunidad científica fuera del ámbito de la lógica borrosa, al modelado y control dentro de ella. La formalización matemática del sistema de control borroso se abordó de forma totalmente general, sin restricciones ni en el vector de estado de la planta ni en el de control. Respecto de las técnicas de identificación de los parámetros del modelo de la planta, se empleó la ya clásica neuroborrosa basada en gradiente descendente junto a otra más novedosa, basada en la hibridación de la técnica anterior con la heurística inspirada en las colonias de hormigas. Esta hibridación de técnicas para la identificación paramétrica de un modelo borroso fue una novedad, ya que no existían publicaciones al respecto. Todo este trabajo quedó recogido en la publicación:

Andújar, J. M., Barragán, A. J., Córdoba, J. M., Fernández, I. Diseño de sistemas de control borroso: Modelado de la planta. *Revista Iberoamericana de Automática e Informática Industrial (RIAI)*, 3(1):75–81, 2006.

Índices de calidad: Primera revista de Ingeniería de Control en español incluida en el *Science Citation Index Expanded* (SciSearch) y *Journal Citation Reports / Science Edition*. Está indexada también en el ISI y en SCOPUS. Está pendiente de índice de impacto en 2008.

Capítulo 4 *Análisis de sistemas dinámicos mediante modelos borrosos*

Una vez resuelto el modelado borroso de estado, se abordó, como paso siguiente, el desarrollo de Taylor de orden uno del modelo borroso, lo cual permitió obtener la linealización de un modelo borroso de estado completamente general. Esta linealización se generalizó hasta el punto de no limitar el número de reglas del modelo borroso, ni el tamaño de los vectores de estado y control, ni el tipo de función de pertenencia utilizada para cada uno de los conjuntos borrosos del modelo, ya que incluso es posible mezclar distintas funciones de pertenencia en una misma regla. Cumplido este paso, se realimentó el modelo del controlador en el de la planta, y se procedió a calcular su derivada con respecto a cada una de las variables de estado, lo cual permitió obtener la matriz jacobiana de un sistema de control borroso en lazo cerrado. El desarrollo teórico se compendió en un algoritmo que luego fue programado en MATLAB® junto a una serie de aplicaciones para facilitar su uso. El software diseñado proporciona funciones

complementarias para facilitar tanto la lectura como la escritura de sistemas borrosos, así como una interfaz que permite el uso de todas las funciones desarrolladas desde el entorno de MATLAB®, lo cual ha posibilitado completar y extender las posibilidades de su *Fuzzy Logic Toolbox*. Tanto es así, que el software está alojado en la propia página de Mathworks: <http://www.mathworks.com/matlabcentral/fileexchange>, bajo el nombre *Fuzzy Logic Toolbox Add-On*. Todo el trabajo descrito fue publicado en:

Andújar, J. M., Barragán, A. J. A methodology to design stable nonlinear fuzzy control systems. *Fuzzy Sets and Systems*, 154(2):157–181, 2005.

Índices de calidad: Revista incluida en el primer tercio del JCR en la categoría *computer science, theory & methods*. Artículo más descargado en 2005. Esta revista está considerada la número 2 del mundo en el campo de la lógica borrosa. Su último índice de impacto disponible (2007) es 1,373.

Una extensión más didáctica y autocontenida del trabajo anterior fue publicada en forma de capítulo de libro:

Andújar, J. M., Barragán, A. J., Gegúndez, M. E., Maestre, M. *Una metodología para el diseño de sistemas de control borroso estables (Nuevas tendencias en control inteligente)*. CEA-IFAC, ISBN: 84-689-8640-2, 31–37, 2006.

Capítulo 5 *Diseño de controladores borrosos*

Capítulo 6 *Aplicaciones*

Estos dos capítulos han de ir necesariamente juntos respecto a su producción científica, ya que los desarrollos del capítulo 5 se aplican en el 6, lo cual implica que ambos capítulos participen en las mismas publicaciones.

Una vez terminado todo el trabajo referente a modelado, representación y obtención de la matriz jacobiana del sistema en lazo cerrado, se estaba en condiciones, ya que se habían creado las herramientas necesarias, de abordar el paso final: establecer una metodología general que permitiera la síntesis de controladores borrosos estables por diseño, es decir, que la estabilidad quedara garantizada durante el propio proceso de diseño del controlador. Sin duda la formalización del concepto de estabilidad debía pasar por la aplicación, de alguna manera, del teorema de estabilidad asintótica de Lyapunov, pero descartando de entrada la búsqueda heurística de funciones de Lyapunov, ya que es una misión, la mayoría de las veces, imposible. Por ello se avanzó en la línea del teorema de Krasovskii generalizado, el cual obtiene la función de Lyapunov como una consecuencia de su cumplimiento. A pesar de que la formalización teórica del procedimiento quedó resuelta, no ocurrió lo mismo con la práctica, ya que aparecieron serios problemas de aplicabilidad. A partir de aquí, se enunció y demostró un nuevo teorema que, basado en la teoría de Lyapunov e inspirado en el teorema de Krasovskii, permite garantizar la estabilidad asintótica del sistema de control borroso en lazo cerrado. Con objeto de llevar a la práctica la aplicación del teorema se desarrolló un algoritmo que genera un controlador borroso inicial a partir del modelo borroso de la planta y, posteriormente, ajusta los parámetros adaptables de los consecuentes hasta conseguir la estabilidad asintótica del sistema en lazo cerrado. Este trabajo, culminación de la Tesis, va a salir publicado en breves fechas:

Andújar, J. M., Barragán, A. J., Gegúndez, M. E. A general and formal methodology for designing stable nonlinear fuzzy control systems. *IEEE Transactions on Fuzzy Systems*, 2009. Artículo aceptado y en prensa.

Índices de calidad: Revista incluida en el primer tercio del JCR en la categorías *computer science, artificial intelligence y engineering, electrical & electronic*. Esta revista está considerada la número 1 del mundo en el campo de la lógica borrosa. Su último índice de impacto disponible (2007) es 2,137.

Por último, se quiso completar la Tesis con un enfoque de diseño de sistemas de control borroso al modo clásico, esto es, desde el punto de vista de la heurística mediante la incorporación del conocimiento de un experto. Este enfoque tiene la gran ventaja de no necesitar de modelo de la planta, aunque por el contrario no permite un análisis riguroso y menos aún formal de la estabilidad del sistema de control resultante. Esta es la metodología de diseño de controladores que probablemente más éxitos prácticos le ha dado a la lógica borrosa, aunque también ha sido la más criticada por su falta de rigurosidad y sistematización, aunque funciona. Fruto de este trabajo ha sido la publicación:

Andújar, J. M., Barragán, A. J., Gegúndez, M. E., Maestre, M. Control borroso multivariable basado en heurística. un caso práctico: Grúa porta contenedores. *Revista Iberoamericana de Automática e Informática Industrial (RIAI)*, 4(2):81–89, 2007.

Índices de calidad: Primera revista de Ingeniería de Control en español incluida en el *Science Citation Index Expanded* (SciSearch) y *Journal Citation Reports / Science Edition*. Está indexada también en el ISI y en SCOPUS. Está pendiente de índice de impacto en 2008.

Introducción a la lógica borrosa y sus aplicaciones

EN este capítulo se inicia el estudio de los sistemas borrosos y sus conceptos fundamentales. El capítulo está estructurado en tres secciones. La primera de ellas realiza una revisión de la breve pero exitosa historia de la lógica borrosa. Partiendo de la relativamente reciente creación de la teoría de conjuntos borrosos a manos de Lofti A. Zadeh, se recorren los hitos más destacados que han culminado en el desarrollo y la aceptación de la que goza hoy en día esta teoría. A continuación se presentan algunas aplicaciones, tanto a nivel industrial como pertenecientes al mercado de consumo, en las que la lógica borrosa ha tenido un gran éxito. Esta sección finaliza con una revisión de la investigación sobre lógica borrosa que se realiza en España, un país puntero en este aspecto, presentando las líneas de investigación de algunos de los grupos de investigación españoles más importantes. En la segunda sección se describen los fundamentos de la lógica borrosa. Comenzando por la propia definición de conjunto borroso, se desarrollan las definiciones y operaciones principales que dan forma a esta teoría, así como los distintos tipos de sistemas borrosos existentes. La tercera y última sección recoge las principales conclusiones de este capítulo, así como una revisión de las referencias bibliográficas incluidas en el mismo.

2.1. Introducción

En 1965, el profesor de teoría de sistemas de la Universidad de California (Berkeley), Lotfi A. Zadeh publicó su famoso artículo *Fuzzy Sets* [1]. En este trabajo presentó a la comunidad científica internacional la idea de *conjunto borroso*¹, y por extensión, de lo que se conocería como *lógica borrosa*. Desde entonces, el término *lógica borrosa*, o *fuzzy logic*, ha sido empleado desde dos puntos de vista diferentes: por un lado, la forma estricta del

¹El término *fuzzy* se suele traducir indistintamente como borroso o difuso, aunque en ingeniería suele predominar la utilización del primero de ellos. No obstante, como se verá a lo largo del texto, la utilización de un término u otro es más una cuestión de escuelas que de clasificación por aplicaciones.

término hace referencia a un sistema lógico que generaliza la lógica digital clásica para admitir el razonamiento impreciso o bajo incertidumbre; por el otro, en un sentido más amplio y muy comúnmente utilizado, la lógica borrosa engloba todas las teorías y tecnologías que emplean conjuntos borrosos. En cualquier caso, la lógica borrosa ha conseguido en menos de medio siglo revolucionar la tecnología e impulsar la inteligencia artificial mediante la conjunción de los fundamentos matemáticos, la lógica y el razonamiento.

La lógica borrosa es una lógica multivaluada que permite valores intermedios para poder definir evaluaciones convencionales. De esta forma, no se determina si un elemento pertenece o no a un conjunto, sino la calidad de la pertenencia evaluada en un rango completo desde la no pertenencia hasta la pertenencia con la máxima calidad posible. Esta característica permite el análisis del razonamiento aproximado, mucho menos estricto que la lógica bivaluada tradicional.

Tomando como base la imprecisión del razonamiento humano, pero bajo un planteamiento matemático, la lógica borrosa es capaz de generar la respuesta a una situación basándose en el conocimiento adquirido sobre ésta, que podrá ser inexacto e incompleto. Por ejemplo, mediante la lógica clásica se puede definir un conjunto de personas altas, pero esto obliga a establecer una frontera clara que separe este subconjunto del resto de elementos, como una altura de 1,80 m. Este razonamiento clásico dejaría a las personas de 1,79 m de estatura fuera del conjunto de personas altas. Es evidente que este razonamiento tan estricto de la lógica clásica choca frontalmente con el razonamiento humano, mucho más vago e impreciso.

La lógica borrosa permite establecer límites suaves (o borrosos) para los conjuntos, haciendo que no todo tenga que ser blanco o negro, sino que es posible la existencia de niveles de grises. Siguiendo con el ejemplo pero desde el punto de vista borroso, se podría decir que una persona que mida 1,90 m está claro que es alta, o la calidad de su pertenencia al conjunto de personas altas es 1. Una que mida 1,50 m está claro que no lo es, es decir, tiene un grado de pertenencia 0 a dicho conjunto. Con las alturas intermedias se podría decir que la calidad de su pertenencia al conjunto variaría entre 0 y 1, siendo mayor el grado de pertenencia cuanto más alta sea la persona a clasificar, por ejemplo 0,25 para las personas de 1,60 m y 0,75 para las de 1,80 m. Así, una persona de 1,79 m no estaría excluida del conjunto de personas altas, aunque la calidad de su pertenencia a dicho conjunto sería menor que la de otra persona de mayor estatura.

Desde sus inicios la lógica borrosa ha tenido que enfrentarse con constantes ataques por parte de sus detractores, y su nacimiento no fue todo lo prometedor que hoy cabría esperar viendo el éxito que ha alcanzado. Pese al rechazo que gran parte de la comunidad científica expresaba inicialmente a esta teoría, en muchos casos tan solo por su nombre o por hacer énfasis en el concepto de imprecisión, se fueron creando grupos de investigación sobre esta temática tanto en Europa como en Estados Unidos. La posterior incorporación de los países asiáticos al desarrollo borroso fue fundamental para dar el último impulso a la teoría de Zadeh. La capacidad de las empresas japonesas para aplicar la lógica borrosa a sus productos, unida a la estrecha colaboración de estas empresas con las Universidades japonesas, fomentaron la transferencia de los conocimientos sobre lógica borrosa al campo de la ingeniería y el control. Como el propio Zadeh dijo en una entrevista concedida a La

Vanguardia en el año 2004 [2], ... en Japón antes de discutir teóricamente mi lógica se pusieron a hacerla funcionar en la práctica, y se impuso. Así, desde los años 80 Japón es el líder indiscutible del mercado de las aplicaciones borrosas y foco de innumerables avances de esta teoría.

2.2. Breve historia de la lógica borrosa

Como ya se ha mencionado anteriormente, Lotfi A. Zadeh es considerado el padre de la lógica borrosa, aceptándose como fecha de su nacimiento 1965, el año de la publicación de su artículo *Fuzzy Sets* [1] en la revista *Information and Control*. Este artículo fue el resultado de una idea, mencionada por el profesor Zadeh en 1962 [3], sobre la necesidad de crear un nuevo tipo de matemáticas, unas matemáticas que permitieran expresar el conocimiento humano en términos distintos a las distribuciones de probabilidad. Desde los inicios de la lógica borrosa no sólo se han producido una gran cantidad de trabajos teóricos y aplicados, sino que han visto la luz multitud de productos comerciales basados en dicha teoría, originando lo que ha dado en llamarse *tecnología fuzzy*.

Raramente sucede en la historia de la ciencia y la técnica que el padre de una teoría matemática pueda presenciar la eclosión de la misma y, sobre todo, su transferencia tecnológica a la ingeniería. Pero el profesor Zadeh no se sólo fue el padre de la lógica borrosa, sino que ha sido también uno de sus grandes impulsores. Desde que en 1965 publicara *Fuzzy Sets* y se centrara en su nueva teoría, Lofti Zadeh ha creado multitud de artículos que la han ampliado y formalizado [4, 5, 6, 7, 8, 9, 10, 11, 12]. Hoy día Lofti Zadeh sigue participando activamente en el desarrollo de la lógica borrosa [13], y no es extraño que prologue libros sobre este tema [14, 15].

Durante la primera década de vida de la lógica borrosa los nacientes grupos de investigación fueron incorporando al campo borroso las estructuras matemáticas típicas: relaciones, funciones, grafos, grupos, autómatas, gramáticas, lenguajes, algoritmos y programas. Así, R. E. Bellman y Zadeh trabajaron en sistemas de toma de decisión en entornos borrosos [16]. G. Lakoff estudió el problema desde un punto de vista lingüístico [17]. J. A. Goguen investigó sobre metodologías para la caracterización de estructuras matemáticas borrosas [18], R. E. Smith y M. Sugeno trabajaron en medidas borrosas [19, 20], J. C. Dunn y J. C. Bezdeck en clustering borroso [21, 22], C. V. Negoita en la recuperación de información borrosa [23, 24] y H. J. Zimmermann estudió la optimización borrosa [25]. Desde España también hubo algunas aportaciones, como el estudio sobre la borrosificación algebraica y los sistemas topológicos realizados por los investigadores I. Sols y J. Meseguer [26, 27].

En 1974, S. Assilian y E. H. Mamdani, del Queen Mary College de Londres, establecieron el primer hito en cuanto a aplicación industrial del control borroso. Mediante un sistema basado en reglas lingüísticas consiguieron controlar un generador de vapor que no habían sido capaces de mantener bajo control mediante técnicas convencionales [28]. Esta realización práctica fue uno de los frutos de la Tesis Doctoral de S. Assilian [29]. Posteriormente, en 1976, las compañías Blue Circle Cement y SIRA desarrollaron en Dinamarca un controlador para un horno de cemento, el cual incorporaba el conocimiento

de operadores experimentados para su funcionamiento [30]. De esta forma, la lógica borrosa comenzó poco a poco a adentrarse en los ámbitos industriales. En 1977, Mamdani y King publican *The application of fuzzy control systems to industrial processes* [31], en el que describen cómo emplear la lógica borrosa como método de control para sistemas industriales complejos.

A partir de los años 80 son las empresas japonesas las que comienzan a destacar en la utilización de la lógica borrosa. Así, Fuji Electric desarrolló una planta de tratamiento de aguas que empleaba un sistema de control borroso, lo que les permitió adquirir los conocimientos necesarios para así desarrollar el primer controlador borroso de propósito general FRUITAX (*Fuzzy Rule Information processing Tool of Advanced control*), lanzado al mercado en 1985, y que estaba constituido básicamente por un procesador de 16 bits junto con un sistema de inferencia borrosa basado en FORTRAN. Al año siguiente se produjo otro de los hitos más importantes en la historia de la lógica borrosa, nació el primer chip VLSI para la realización de inferencias borrosas desarrollado por M. Togai y H. Watanabe [32]. Los chips VLSI incrementaron en gran medida las posibilidades de elección de sistemas basados en reglas para aplicaciones de tiempo real.

En 1987, Seiji Yasunobu y sus colaboradores de la empresa Hitachi pusieron a punto el sistema de control del metro de la ciudad de Senday [33]. El enorme éxito que tuvieron estos proyectos provocó un Japón un auténtico boom de las aplicaciones basadas en la lógica borrosa, favoreciendo que en 1988 el gobierno japonés pusiese en marcha un estudio minucioso sobre el establecimiento de proyectos de lógica borrosa entre las Universidades y la industria. Fruto de este estudio se implantaron dos programas de investigación a nivel nacional sobre lógica borrosa, uno de ellos liderado por el MITI (*Ministry of International Trade and Industry*), y el otro por la STA (*Science and Technology Agency*). A través del programa dependiente del MITI se creó el consorcio denominado LIFE (*Laboratory for International Fuzzy Engineering research*), el cual agrupaba a más de 50 compañías y poseía un presupuesto de 5.000 millones de yenes (unos 50 millones de dólares) para seis años [34].

El éxito de la lógica borrosa no sólo se ha demostrado a niveles industriales [35, 36], sino también a nivel del mercado de consumo. En 1987, Matsushita Electric Industrial Co. (conocida fuera de Japón como Panasonic) fue la primera empresa en emplear la lógica borrosa a un producto de consumo, concretamente en un sistema de ducha que controlaba la temperatura del agua. Esta misma compañía lanzó en 1990 una lavadora automática basada en lógica borrosa. A partir de entonces, una gran cantidad de compañías japonesas se apresuraron a lanzar productos que utilizaban de una u otra forma la lógica borrosa: hornos, cocinas, refrigeradores, etc.

Actualmente son muchos los aparatos de consumo que incluyen algún tipo de controlador borroso. Por ejemplo, tanto las cámaras fotográficas como las de vídeo suelen emplear un controlador borroso para realizar el enfoque automático o el ajuste del tiempo de exposición [37, 38, 39]. En las cocinas de las casas también está presente la lógica borrosa, ya que muchos electrodomésticos la emplean en sus sistemas de control para ahorrar energía o agua [40, 41, 42], mejorando considerablemente su eficiencia.

La industria del automóvil es otro de los sectores donde se ha aplicado la lógica borrosa con mucho éxito [43, 44, 45]. Se han aplicado controladores borrosos en sistemas de suspensión activa [46, 47, 48], en sistemas de frenado [49, 50, 51, 52, 53, 54] y en el control electrónico de motores [55]. Con la acuciante necesidad de eliminar la dependencia del petróleo, casi todos los fabricantes tienen abiertas líneas de investigación sobre vehículos híbridos. En este campo tan tremendamente complejo también está triunfando la lógica borrosa como un excelente método de control, y son muy numerosos los avances que se están consiguiendo gracias al control borroso [56, 57, 58, 59, 60, 61].

En España la investigación sobre la lógica borrosa la impulsó el Profesor Enric Trillas, catedrático de ciencias de la computación e inteligencia artificial en la Universidad Politécnica de Madrid, quien ha sido galardonado con los tres premios más prestigiosos que se conceden en el ámbito de la lógica borrosa: el *Pioneer Award* de la *European Society For Fuzzy Logic and Technologies* (EUSFLAT), el *Fellow* de la *International Fuzzy System Association* (IFSA) y *Fuzzy Systems Pioneer Award* otorgado por la *IEEE Computational Intelligence Society*. Como autor y co-autor de más de doscientos artículos [62, 63, 64, 65, 66, 67, 68, 69, 70] y varios libros [71, 72, 73, 74], el profesor Trillas ha contribuido notablemente a convertir a España en uno de los núcleos de referencia en Europa en la actividad basada en la lógica borrosa. En España existen muchos grupos de investigación que trabajan con lógica borrosa. En algunos casos la lógica borrosa es el centro y parte fundamental de su investigación, teniendo como objetivo seguir ampliando y mejorando dicha teoría; mientras que en otros casos, la lógica borrosa es empleada como herramienta para la resolución de problemas complejos.

Por citar sólo algunos ejemplos, el grupo de investigación *Soft computing y sistemas de información inteligentes* (<http://sci2s.ugr.es>) de la Universidad de Granada posee múltiples líneas de investigación sobre lógica borrosa y sus aplicaciones [75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86]. Entre estas líneas de investigación destacan sus trabajos sobre sistemas genético-borrosos, toma de decisiones mediante lingüística y *soft computing*. El grupo de investigación *ARAI, Razonamiento Aproximado e Inteligencia Artificial* (<http://arai.ugr.es>), también perteneciente a la Universidad de Granada, desarrolla, entre otros, sistemas de aprendizaje, de conocimiento, de soporte a la toma de decisiones y de reconocimiento de patrones basados en lógica borrosa [87, 88, 89, 90, 91, 92]. El grupo de investigación de *Inteligencia computacional* (<http://ic.ugr.es>) es otro grupo de investigación, con sede en la Universidad de Granada, que tiene como principal objetivo la investigación para el desarrollo de sistemas computacionales que estén en la vanguardia de la inteligencia artificial. Para ello, sus investigadores emplean y desarrollan, entre otras, técnicas basadas en lógica borrosa [66, 93, 94, 95, 96, 97, 98, 99]. Los grupos de investigación *Ingeniería Automática y Robótica* [100, 101, 102] y *Robótica, Visión y Control* (<http://grvc.us.es>) [103, 104, 105, 106, 107, 108, 109, 110, 111] de la Universidad de Sevilla, investigan sobre lógica borrosa y, fundamentalmente, sobre sus aplicaciones al campo de la robótica móvil. En Madrid, el grupo *Ingeniería de Sistemas, Control, Automática y Robótica (ISCAR)* de la Universidad Complutense desarrolla numerosos trabajos y proyectos de investigación relacionados con temas de control, optimización, robótica y simulación [112, 113, 114, 115, 116, 117, 118, 119], y el grupo de investigación de *Control*

Inteligente (<http://www.disam.upm.es/control>) de la Universidad Politécnica, centra su investigación en robótica móvil y control de procesos [120, 121, 122, 123, 124, 125], para lo que hacen uso de la lógica borrosa como herramienta. En la Universidad de Málaga, el grupo de *Ingeniería de Sistemas y Automática* (<http://www.isa.uma.es>), también emplea la lógica borrosa en aplicaciones de control [100, 109, 126, 127, 128] y robótica [111, 129, 130]. El grupo de investigación de *Inteligencia Artificial e Ingeniería del Conocimiento (AIKE)* (<http://perseo.inf.um.es/~aike>) de la Universidad de Murcia, desarrolla actividades relacionadas con la aplicación de técnicas de ingeniería y gestión del conocimiento a dominios diversos como la medicina, la agricultura, la evaluación de recursos medioambientales o el diagnóstico y mantenimiento de equipos de plantas industriales, empleando la lógica borrosa como herramienta en muchas de sus investigaciones [131, 132, 133, 134, 135, 136, 137, 138, 139]. La Universidad de Valencia da cobijo a varios grupos de investigación, como el *Grupo de Control Predictivo y Optimización Heurística (CPOH)* (<http://ctl-predictivo.upv.es>), cuya investigación se centra en temas relacionados con el control predictivo basado en modelos y la optimización aplicada al control de procesos [140, 141, 142], y el grupo de *Control de Sistemas Complejos (GCSC)* (<http://www.isa.upv.es/investigacion/sistemas-complejos>), que centra sus estudios en el modelado y control de sistemas dinámicos que presentan incertidumbre, no linealidades y distintos modos de comportamiento [143, 144, 145, 146]. El *Instituto de Automática e Informática Industrial* (<http://www.ai2.upv.es>) posee varias líneas de investigación sobre control de procesos [147, 148, 149, 150] en las que aportan características inteligentes y predictivas a la consecución de soluciones tecnológicas empleando, entre otras herramientas, la lógica borrosa. La división de *Automatización y Control de Procesos* de la fundación CARTIF de Valladolid (<http://www.cartif.es>), utiliza la lógica borrosa como herramienta para la consecución de sus objetivos de investigación aplicada, desarrollo tecnológico e innovación de procesos, productos y sistemas [151, 152, 153, 154, 155]. En la Universidad Politécnica de Cataluña, el grupo de *Robótica Inteligente y Sistemas* (<http://grins.upc.es/>) trabaja en el estudio, evaluación y desarrollo de sistemas robóticos [156, 157, 158], y el *Instituto de Investigación en Inteligencia Artificial* (<http://www.iiia.csic.es>), perteneciente al *Consejo Superior de Investigaciones Científicas* (<http://www.csic.es>), investiga sobre los fundamentos matemáticos de la lógica borrosa, el razonamiento aproximado y la elaboración de modelos conceptuales eficientes [159, 160, 161, 162, 163, 164, 165, 166, 167]. Por último, entre las líneas de investigación del grupo *Control y Robótica* de la Universidad de Huelva, se encuentran la modelización y simulación de procesos industriales, el aprendizaje automático y el control inteligente, empleando como una de las herramientas básicas de trabajo la lógica borrosa [168, 169, 170, 171, 172, 173, 174].

Pese a los éxitos del control borroso y pese a estar presente en muchos de los productos tecnológicos que nos rodean, resulta curioso como en España, a pesar de ser un país de referencia en el campo de la investigación, existe un gran desconocimiento sobre la lógica borrosa fuera de este ámbito. Quizás sea debido a la connotación negativa que ha tenido el término borroso por desconocido, o a la inercia mantenida desde los conflictos iniciales de la teoría tanto en Europa como en Estados Unidos. El caso es que hasta hace relativamente

poco tiempo las empresas no han empezado a publicitar la utilización de la lógica borrosa en sus productos, aunque, eso sí, utilizando la expresión inglesa *fuzzy logic*.

2.3. Fundamentos de los sistemas borrosos

Se habla de sistema borroso cuando al menos uno de sus componentes emplea lógica borrosa. Los sistemas que se tratarán en este trabajo serán en todo caso sistemas de control, por lo que la lógica borrosa podrá estar presente en la planta, en el controlador o en ambos elementos. La planta es el sistema físico que se desea estudiar, controlar o monitorizar, de forma que la lógica borrosa no estará implícitamente presente en ella; sin embargo, la lógica borrosa puede utilizarse para obtener un modelo de la planta (modelo borroso), el cual, como se verá posteriormente, permitirá conocer el comportamiento de la misma tanto de forma matemática como lingüística.

El controlador es el elemento que, mediante la aplicación de distintas señales de control, intenta provocar que la planta tenga el comportamiento deseado por el diseñador, según distintos criterios de desempeño, calidad y/o robustez. El algoritmo de control implementado por éste puede estar realizado mediante lógica borrosa, denominándose en tal caso controlador borroso. En muchas ocasiones se utilizan controladores borrosos cuando el modelo de la planta es desconocido, o demasiado complejo como para ser obtenido, por lo que se suele hablar de sistema de control borroso siempre que se emplea un controlador basado en lógica borrosa en el sistema.

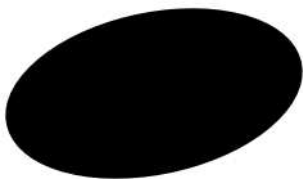
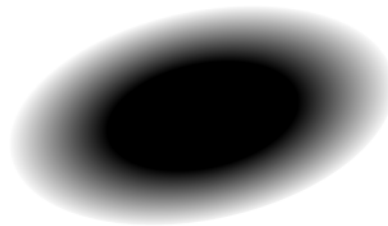
A continuación se estudiarán las definiciones y operaciones principales de la lógica borrosa, así como las fases y conceptos asociados a la generación de conocimiento basado en un conjunto de reglas. El objetivo fundamental que persigue esta sección y en general este capítulo, es describir los conceptos que permitan comprender y diseñar sistemas que, en alguna fase de su implementación, usen términos relativos a la interpretación del conocimiento basado en la teoría de la lógica borrosa.

2.3.1. Conjuntos borrosos y funciones de pertenencia

Los conjuntos borrosos son una extensión de los conjuntos ordinarios que permiten tratar la imprecisión de forma matemática.

Un conjunto ordinario contiene la totalidad de los entes que poseen una propiedad común. Es decir, dada una propiedad de definición de un conjunto, todos los elementos que cumplan dicha propiedad pertenecerán al conjunto, y el resto no. Cuando la propiedad es clara, la pertenencia o no de los elementos a un conjunto queda perfectamente definida, sin embargo, si la definición de un conjunto se realiza de forma ambigua o imprecisa, será imposible saber con certeza si ciertos elementos pertenecen o no al conjunto. Por ejemplo, dado un conjunto de números, su pertenencia al conjunto de *números mayores de 100* está perfectamente definida, pero la pertenencia de éstos al conjunto de *números mucho mayores de 100* ya no estaría tan clara, al ser su definición imprecisa.

La diferencia entre un conjunto borroso y uno ordinario es que los límites de los borrosos no están definidos de forma estricta, sino mediante una función denominada *función de*

**Figura 2.1:** *Conjunto ordinario.***Figura 2.2:** *Conjunto borroso.*

pertenencia. En la lógica clásica, un elemento pertenecerá o no a un determinado conjunto (grado de pertenencia 1 ó 0 respectivamente), mientras que en la lógica borrosa, un elemento puede pertenecer a un conjunto con un grado de pertenencia entre 0 y 1. Al igual que en la lógica clásica, los valores 1 y 0 representan respectivamente la pertenencia absoluta o no del elemento al conjunto, pero en el caso borroso se puede pertenecer a un conjunto con un grado intermedio.

En las figuras 2.1 y 2.2 se muestra gráficamente las diferencias entre un conjunto ordinario o nítido y un conjunto borroso. Si se considera el negro como pertenencia (valor 1) y el blanco como no pertenencia (valor 0), se puede observar que un conjunto ordinario tiene sus límites perfectamente definidos, mientras que un conjunto borroso tiene una región, los distintos niveles de gris, donde la calidad de la pertenencia puede tomar valores intermedios a 0 y 1.

Un concepto fundamental en la teoría de conjuntos borroso es el *grado de pertenencia*. El grado de pertenencia representa la calidad, expresada con un número entre 0 y 1, con la que un elemento pertenece a un determinado conjunto; es decir, la compatibilidad del elemento con el concepto representado por el conjunto.

La función que permite obtener el grado de pertenencia de un elemento a un conjunto borroso se denomina *función de pertenencia*, y se suele denotar mediante el símbolo griego μ . La función de pertenencia del conjunto borroso A se denota como μ_A , y el grado de pertenencia del elemento x al conjunto borroso A , como $\mu_A(x)$. El dominio de definición de la función de pertenencia del conjunto A se conoce como *universo de discurso* de dicho conjunto [175], y recoge todos los posibles valores de la variable x que pueden ser evaluados mediante la función de pertenencia $\mu_A(x)$. De esta forma, un subconjunto borroso A de un conjunto X , puede definirse mediante una aplicación μ_A que relaciona cada elemento $x \in X$ con su grado de pertenencia al conjunto A , $\mu_A(x) \in [0, 1]$. El grado de certeza de la afirmación x pertenece a A depende de la definición de la aplicación $\mu_A(x)$, de forma que mientras más se aproxime el grado de pertenencia a uno, mayor veracidad tendrá dicha afirmación.

La noción humana de pertenencia no juega un papel tan fundamental en el caso de los conjuntos borrosos como en el de los conjuntos ordinarios, salvo el sentido trivial de que

$\mu_A(x)$ sea positivo. Si se desea preservar este concepto de pertenencia clásico, se pueden definir dos niveles α y β , siendo $0 < \alpha < 1$, $0 < \beta < 1$ y $\alpha > \beta$. De esta forma se puede redefinir el concepto clásico de pertenencia por las sentencias:

x pertenece a A si $\mu_A(x) \geq \alpha$
 x no pertenece a A si $\mu_A(x) \leq \beta$
 x tiene un estatus indeterminado con respecto a A si $\beta < \mu_A(x) < \alpha$.

Esta forma de operar lleva al concepto de lógica tri-valuada de Kleene [176], con tres valores de verdad: $T(\mu_A(x) \geq \alpha)$, $F(\mu_A(x) \leq \beta)$ y $U(\beta < \mu_A(x) < \alpha)$.

No debe confundirse el concepto de probabilidad de pertenencia a un conjunto ordinario con el grado de pertenencia a un conjunto borroso. Aunque la función de pertenencia de un conjunto borroso puede tener cierto parecido con una función de probabilidad cuando X es un conjunto numerable (o una función de densidad de probabilidad si X es continuo), hay diferencias sustanciales entre ambos conceptos. Así, no tiene porqué cumplirse que la suma de los grados de pertenencia a distintos conjuntos borrosos sea la unidad, ni tampoco que el grado de pertenencia indique la ocurrencia de un determinado suceso en repetidas experiencias. Por ejemplo, si se indica que el grado de pertenencia del contenido de una botella al conjunto veneno mortal es 0,01, se está indicando que el líquido no es prácticamente venenoso, aunque posea algunas trazas de veneno, por lo que podría no haber riesgo para su ingestión. Sin embargo, si se afirma que la probabilidad de que el líquido sea veneno mortal es de 0,01, el líquido podría ser completamente veneno mortal, aunque con una probabilidad baja, y por lo tanto letal.

Definiciones básicas

Definición 2.1 (*Conjunto borroso*). Sea X un conjunto no vacío, un conjunto borroso A está caracterizado por:

$$\mu_A(x) : X \rightarrow [0, 1], \quad (2.1)$$

donde $\mu_A(x)$ representa el grado de pertenencia del elemento x sobre el conjunto borroso A para cada $x \in X$. A está completamente determinado por el conjunto de pares:

$$A = \{(x, \mu_A(x)), x \in X\}. \quad (2.2)$$

Como se observa en la expresión (2.2), un conjunto borroso queda caracterizado mediante su función de pertenencia. Ésta puede expresarse de forma discreta, enumerando el grado de pertenencia para cada elemento del conjunto, o mediante una definición matemática. En la mayoría de los casos se emplean funciones de pertenencia definidas matemáticamente, ya que estas funciones permiten realizar estudios posteriores con mayor facilidad y son válidas para cualquier elemento dentro de su universo de discurso. Las funciones de pertenencia discretas en principio son sólo válidas para un conjunto de elementos, los empleados para definir la función; sin embargo, si se emplea algún algoritmo de identificación con los pares de datos $(x, \mu_A(x))$, o se define una función a trozos, se podrá obtener una expresión matemática que la represente.

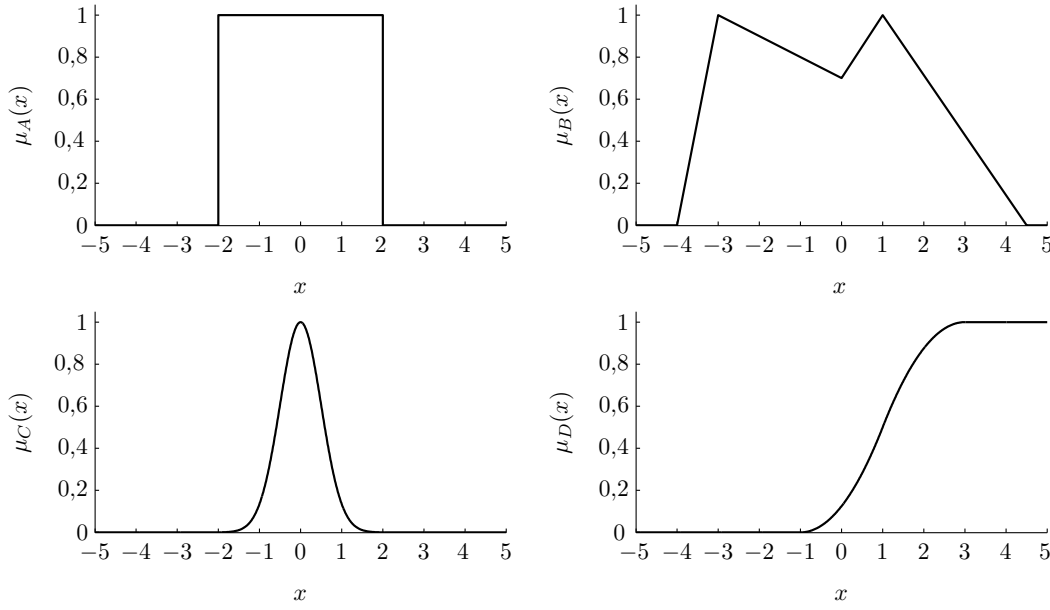


Figura 2.3: Ejemplos de funciones de pertenencia.

En la figura 2.3 se pueden observar distintos ejemplos de funciones de pertenencia, todas ellas con un universo de discurso $X \in [-5, 5]$. La primera, $\mu_A(x)$, es la función de pertenencia de un conjunto nítido, ya que sólo genera grados de pertenencia 0 ó 1, mientras que el resto son funciones de pertenencia borrosas: $\mu_B(x)$ es una función de pertenencia no convexa definida a trozos, $\mu_C(x)$ una gaussiana y $\mu_D(x)$ una función de pertenencia S .

Definición 2.2 (*Conjunto borroso normalizado*). Sea A un conjunto borroso definido en X , se dice que A es un conjunto borroso normalizado si y sólo si $0 \leq \mu_A(x) \leq 1$.

Dado que un grado de pertenencia mayor de uno no tiene mucho sentido lógico, normalmente siempre se emplean conjuntos borrosos normalizados, aunque no se mencione explícitamente.

Definición 2.3 (*Conjunto borroso nulo o vacío*). Sea A un conjunto borroso definido en X , se dice que A es un conjunto borroso nulo o vacío, $A = \emptyset$, si y sólo si $\mu_A(x) = 0$, $\forall x \in X$. Es decir, A es un conjunto borroso nulo si no existe ningún elemento cuyo grado de pertenencia al conjunto sea distinto de cero.

Definición 2.4 (*Conjunto borroso normal*). Sea A un conjunto borroso definido en X , se dice que A es un conjunto borroso normal, o que A es normal, si y sólo si existe al menos un $x \in X$ tal que $\mu_A(x) = 1$. En caso contrario, se dice que A es un conjunto borroso no-normal.

Definición 2.5 (*Conjunto borroso convexo*). Sea A un conjunto borroso definido en X , se dice que A es un conjunto borroso convexo si y sólo si para dos puntos cualesquiera x_1 y x_2 , y cualquier $\lambda \in [0, 1]$, se verifica la ecuación² (2.3). Nótese que esta definición no

²Esta definición de convexidad fue sugerida a Zadeh por su colega E. Berlekamp.

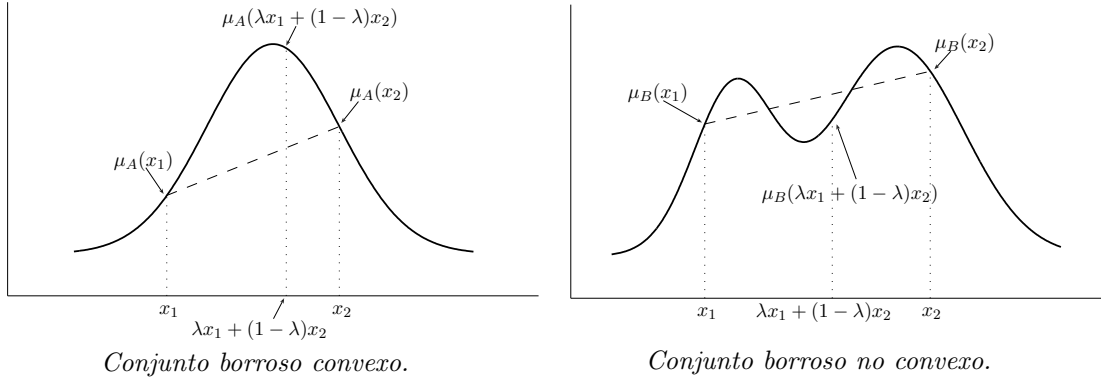


Figura 2.4: *Convexidad en conjuntos borrosos.*

implica que $\mu_A(x)$ deba ser una función convexa.

$$\mu_A(\lambda x_1 + (1 - \lambda)x_2) \geq \min(\mu_A(x_1), \mu_A(x_2)) \quad (2.3)$$

Esta definición indica que, dados dos puntos del universo de discurso, la función de pertenencia de un conjunto borroso convexo evaluada entre esos puntos tomará valores mayores o iguales que la función de pertenencia evaluada en cada uno de dichos puntos. Este concepto se ilustra gráficamente en la figura 2.4.

Definición 2.6 (*Soporte de un conjunto borroso*). Sea A un conjunto borroso definido en X , el soporte de A , denotado por $sop(A)$ o por $sup(A)$, es el subconjunto de X cuyos elementos tienen un grado de pertenencia distinto de cero en A .

$$sop(A) = \{x \in X | \mu_A(x) > 0\} \quad (2.4)$$

Cuando se utilizan funciones de pertenencia asintóticas, como la función gaussiana que se verá posteriormente, esta definición no resulta práctica, ya que el soporte teórico de dichas funciones es infinito. Para limitar el soporte de funciones asintóticas a un conjunto finito de elementos, se suele modificar la definición anterior por la dada en la ecuación (2.5).

$$sop(A) = \{x \in X | \mu_A(x) > \gamma, \gamma > 0\} \quad (2.5)$$

Si se elije γ de valor positivo cercano a cero, se eliminan del soporte de la función aquellos elementos cuya calidad de pertenencia al conjunto es muy baja, consiguiendo a la vez un soporte finito y representativo del conjunto.

Definición 2.7 (*Núcleo de un conjunto borroso*). Sea A un conjunto borroso definido en X , el núcleo de A , denotado por $nuc(A)$ o por $core(A)$, es el subconjunto de X cuyos elementos tienen grado de pertenencia uno en A .

$$nuc(A) = \{x \in X | \mu_A(x) = 1\} \quad (2.6)$$

Si A es un conjunto borroso convexo, su núcleo también es convexo [1].

Definición 2.8 (*Frontera de un conjunto borroso*). Sea A un conjunto borroso definido en X , la frontera de A es el subconjunto de X cuyos elementos tienen grados de pertenencia en A comprendidos entre cero y uno.

Definición 2.9 (*Peso o altura de un conjunto borroso*). Sea A un conjunto borroso definido en X , el peso o altura de A es el máximo valor de la función de pertenencia μ_A . Si A es un conjunto borroso normal (y normalizado), su peso será 1.

Definición 2.10 (*Número borroso*). Un número borroso A es un conjunto borroso definido en $\mathbb{R}$, normal y convexo, cuya función de pertenencia es continua y su soporte limitado. Si el soporte no es limitado, pero se cumple la condición de límite dada por la ecuación (2.7), también se puede decir que A es un número borroso o, más exactamente, cuasi-borroso.

$$\lim_{|x| \rightarrow \infty} A(x) = 0 \quad (2.7)$$

Desde el punto de vista práctico los conceptos de número borroso y número cuasi-borroso son idénticos, ya que ambos representan un valor real desde el punto de vista de la lógica borrosa.

Los números borrosos son conjuntos borrosos con un claro significado cuantitativo, en la medida en que clasifican un concepto alrededor de un número o intervalo de números. Estos números tienen una gran importancia en el control borroso, ya que permiten representar valores numéricos mediante las imprecisión propia de la lógica borrosa.

Tipos básicos de funciones de pertenencia

A continuación se definirán las funciones de pertenencia más comúnmente utilizadas, indicando en cada caso sus características y propiedades.

Definición 2.11 (*Función de pertenencia triangular*). Una función de pertenencia triangular, $Tri_{c\alpha\beta}[c, \alpha, \beta](x)$, se define según la expresión (2.8), siendo c su centro, $\alpha \geq 0$ su ancho izquierdo, y $\beta \geq 0$ su ancho derecho.

$$\mu_{Tri_{c\alpha\beta}}[c, \alpha, \beta](x) = \begin{cases} \frac{1 - (c - x)}{\alpha} & \text{si } (c - \alpha) \leq x < c \\ \frac{1 - (x - c)}{\beta} & \text{si } c \leq x \leq (c + \beta) \\ 0 & \text{en otro caso} \end{cases} \quad (2.8)$$

Otra forma más común de definir la función de pertenencia triangular, es mediante la identificación de los tres puntos del triángulo, $Tri[a, b, c](x)$, con $a \leq b \leq c$. En este caso la función de pertenencia vendrá dada por la expresión (2.9).

$$\mu_{Tri}[a, b, c](x) = \begin{cases} \frac{x - a}{b - a} & \text{si } a < x \leq b \\ \frac{c - x}{c - b} & \text{si } b < x < c \\ 0 & \text{en otro caso} \end{cases} \quad (2.9)$$

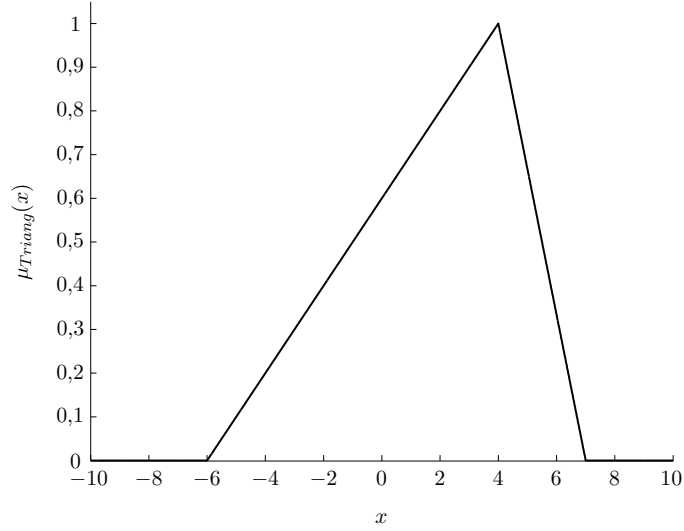


Figura 2.5: *Función de pertenencia triangular.*

Las funciones de pertenencia triangulares son continuas y definen un conjunto borroso normal, convexo y con soporte finito, por lo que pueden emplearse para representar números borrosos.

Si la función de pertenencia se define mediante (2.8), el soporte vendrá dado por $(c - \alpha, c + \beta)$ y el núcleo coincidirá con c ; mientras que si se define mediante (2.9), el soporte será (a, c) y el núcleo coincidirá con b .

En la figura 2.5 se representa la función de pertenencia $Tri_{c\alpha\beta}[4, 10, 3](x)$, que también puede declararse como $Tri[-6, 4, 7](x)$, y cuyo universo de discurso es $X \in [-10, 10]$. El soporte de esta función coincide con el intervalo $(-6, 7)$, y su núcleo con el valor 4.

Definición 2.12 (*Función de pertenencia trapezoidal*). Una función de pertenencia trapezoidal, $Trap_{ab\alpha\beta}[a, b, \alpha, \beta](x)$, se define según la expresión (2.10), siendo $[a, b]$, con $a \leq b$, su intervalo de confianza o núcleo, $\alpha \geq 0$ su ancho izquierdo, y $\beta \geq 0$ su ancho derecho.

$$\mu_{Trap_{ab\alpha\beta}}[a, b, \alpha, \beta](x) = \begin{cases} \frac{1 - (a - x)}{\alpha} & \text{si } (a - \alpha) \leq x < a \\ 1 & \text{si } a \leq x \leq b \\ \frac{1 - (x - b)}{\beta} & \text{si } b < x \leq (b + \beta) \\ 0 & \text{en otro caso} \end{cases} \quad (2.10)$$

Al igual que en el caso anterior, la función de pertenencia trapezoidal también puede definirse mediante los 4 puntos característicos del trapecio, $Trap[a, b, c, d](x)$, siendo $a \leq b \leq c \leq d$. En este caso la función de pertenencia trapezoidal vendrá dada por la

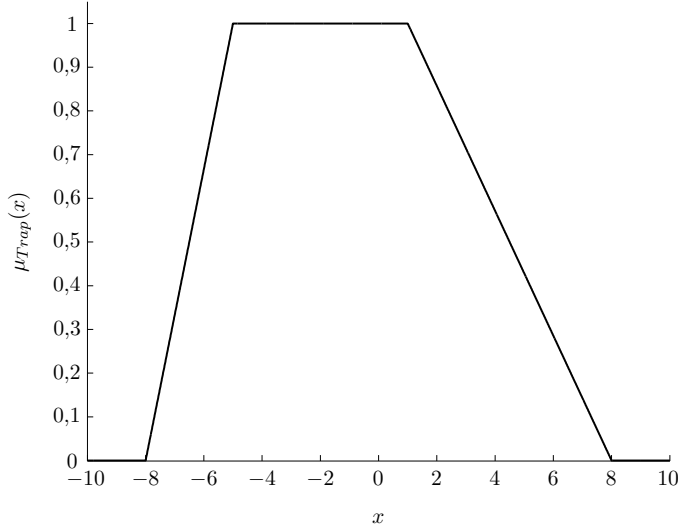


Figura 2.6: Función de pertenencia trapezoidal.

expresión (2.11).

$$\mu_{Trap}[a, b, c, d](x) = \begin{cases} \frac{x-a}{b-a} & \text{si } a < x < b \\ 1 & \text{si } b \leq x \leq c \\ \frac{d-x}{d-c} & \text{si } c < x < d \\ 0 & \text{en otro caso} \end{cases} \quad (2.11)$$

La función de pertenencia trapezoidal es continua y define un conjunto borroso normal, convexo y con soporte finito, por lo que se puede emplear para representar un número borroso.

Si la función de pertenencia se define mediante (2.10), el soporte vendrá dado por $(a - \alpha, b + \beta)$ y el núcleo coincidirá con el intervalo $[a, b]$; mientras que si se define mediante (2.11), su soporte será (a, d) y su núcleo $[b, c]$.

En la figura 2.6 se representa un ejemplo de función de pertenencia trapezoidal con universo de discurso $X \in [-10, 10]$. En este caso, dicha función puede definirse como $Trap_{ab\alpha\beta}[-5, 1, 3, 7](x)$, o como $Trap[-8, -5, 1, 8](x)$. El soporte de dicha función coincide con el intervalo $(-8, 8)$, y su núcleo con $[-5, 1]$.

Definición 2.13 (*Función de pertenencia gaussiana*). Una función de pertenencia gaussiana, $Gauss[c, \beta](x)$, se define según la expresión (2.12), siendo c su centro y $\beta > 0$ el ancho de la función de pertenencia.

$$\mu_{Gauss}[c, \beta](x) = e^{-\left(\frac{x-c}{\beta}\right)^2} \quad (2.12)$$

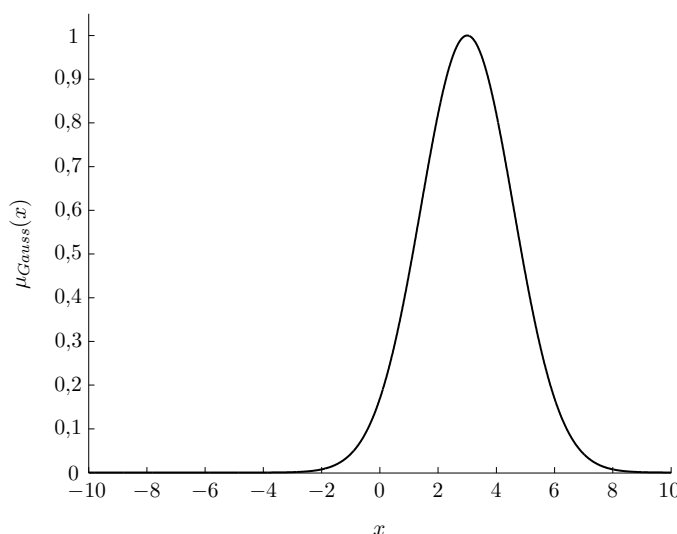


Figura 2.7: *Función de pertenencia gaussiana.*

La función de pertenencia gaussiana es continua, simétrica respecto a c , y define un conjunto borroso normal y convexo. El núcleo de la función de pertenencia coincide con c y su soporte es infinito. Como la función de pertenencia tiende asintóticamente a cero por sus dos extremos, es decir, cumple la condición de límite $\lim_{|x| \rightarrow \infty} \mu_{Gauss}(x) = 0$, se puede emplear para representar un número cuasi-borroso.

El soporte de la función gaussiana es infinito, ya que dicha función tiende a cero de forma asintótica por ambos extremos; sin embargo, a veces es necesario disponer de un valor finito para el soporte. Se puede suponer que el soporte es finito si se desprecian valores de pertenencia menores de cierta cantidad. Un criterio práctico razonable es suponer el soporte de la función gaussiana como todos aquellos valores de x para los que $\mu_A(x)$ sea mayor del 10 % del máximo grado de pertenencia posible para $\mu_A(x)$ (1 si A es un conjunto normalizado y normal). Otra posibilidad es suponer el soporte como una función del ancho de la campana (β), o aproximar su valor mediante la comparación de la función gaussiana con otras funciones de pertenencia de soporte finito, como la función de pertenencia triangular.

En la figura 2.7 se muestra un ejemplo de función de pertenencia gaussiana, cuyo centro es $c = 3$ y su ancho $\beta = 1,5$. Un posible valor finito para el soporte podría ser $(c - 2\beta, c + 2\beta)$, lo que en este ejemplo concreto sería $(0, 6)$.

Definición 2.14 (*Función de pertenencia gaussiana doble*). Una función de pertenencia gaussiana doble, $Gauss2[c_1, c_2, \beta_1, \beta_2](x)$, se define de forma asimétrica mediante el producto de dos funciones gaussianas según el conjunto de ecuaciones (2.13). La parte izquierda de la función se define con el conjunto de parámetros $[c_1, \beta_1]$, con $\beta_1 > 0$, y la parte derecha de la misma con los parámetros $[c_2, \beta_2]$, con $\beta_2 > 0$. Si $c_1 \leq c_2$, el núcleo

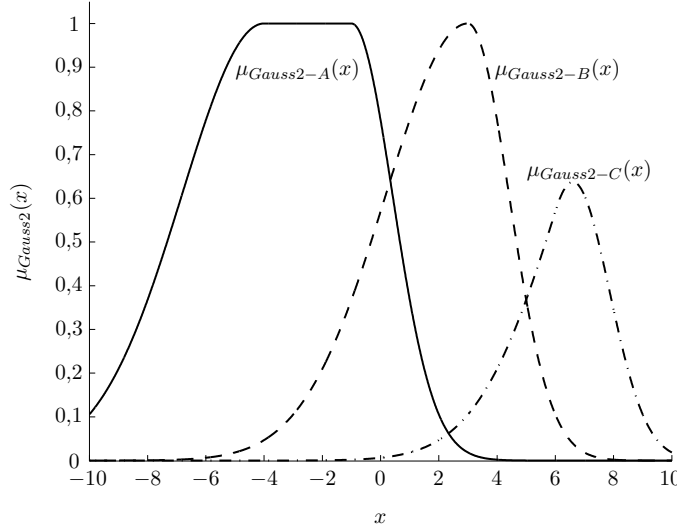


Figura 2.8: Funciones de pertenencia gaussiana doble.

coincidirá con el intervalo $[c_1, c_2]$ y el conjunto borroso A será normal, mientras que si $c_1 > c_2$, el núcleo será nulo y el conjunto borroso A no será normal.

$$\mu_{Gauss2}[c_1, c_2, \beta_1, \beta_2](x) = \mu_1(x)\mu_2(x), \quad (2.13a)$$

siendo

$$\mu_1(x) = \begin{cases} \mu_{Gauss}[c_1, \beta_1](x) & \text{Si } x < c_1 \\ 1 & \text{Si } x \geq c_1 \end{cases} \quad (2.13b)$$

y

$$\mu_2(x) = \begin{cases} \mu_{Gauss}[c_2, \beta_2](x) & \text{Si } x > c_2 \\ 1 & \text{Si } x \leq c_2 \end{cases} \quad (2.13c)$$

El soporte de la función de pertenencia gaussiana doble es infinito, pero el mismo razonamiento dado para la función de pertenencia gaussiana se puede utilizar para obtener un valor finito del soporte de esta función.

La función gaussiana doble es continua y convexa. Aunque su soporte es infinito, al igual que la función gaussiana, cumple con la condición de límite, por lo que siempre que sea normal, es decir, siempre que $c_1 \leq c_2$, podrá utilizarse para representar un número cuasi-borroso.

En la figura 2.8 se muestran varios ejemplos de funciones de pertenencia gaussiana dobles. Si $c_1 < c_2$, la forma de esta función de pertenencia será similar a $\mu_{Gauss2-A}(x)$. Si $c_1 = c_2$ se obtendrá una forma gaussiana pero con brazos asimétricos cuyo núcleo coincide con los centros, como $\mu_{Gauss2-B}(x)$. Si por el contrario $c_1 > c_2$, la función será no-normal y su núcleo será nulo, como $\mu_{Gauss2-C}(x)$.

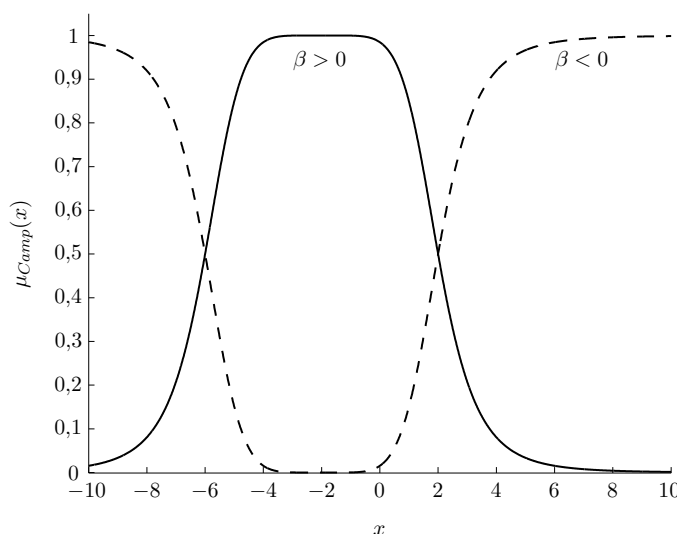


Figura 2.9: Funciones de pertenencia campana.

Definición 2.15 (*Función de pertenencia campana*). Una función de pertenencia campana, $Campana[\alpha, \beta, c](x)$, se define según la expresión (2.14), siendo c el centro de la función de pertenencia, y $\alpha > 0$ y $\beta \neq 0$ dos parámetros de ajuste. El parámetro α tiene relación con el ancho de la función por su parte baja, o pies, mientras que el parámetro β , normalmente positivo, tiene relación con el ancho de la función de pertenencia por la zona alta, u hombros. Si $\beta < 0$, la función de pertenencia toma la forma de una campana invertida y puede ser interesante para predicados³ de tipo negación (*Si x no es ...*).

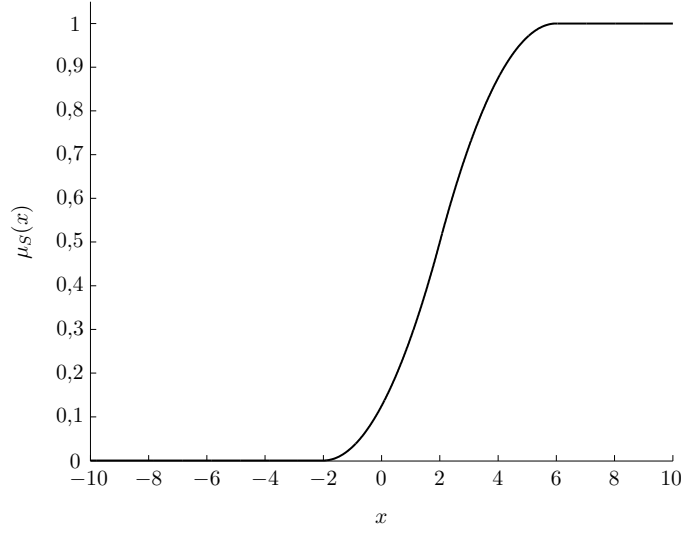
$$\mu_{Campana}[\alpha, \beta, c](x) = \frac{1}{1 + \left| \frac{x-c}{\alpha} \right|^{2\beta}} \quad (2.14)$$

La función de pertenencia campana es continua, simétrica con respecto a c , y define un conjunto borroso normal y convexo. Aunque su soporte es infinito, cumple la condición de límite siempre que $\beta > 0$, por lo que en dicho caso podrá utilizarse para representar números cuasi-borrosos. Para obtener un valor finito del soporte puede aplicarse un razonamiento similar al utilizado con la función de pertenencia gaussiana.

En la figura 2.9 se pueden ver dos funciones de pertenencia campana, una con $\beta = 3$ y la otra con $\beta = -3$. En ambos casos se ha tomado $\alpha = 4$ y $c = -2$.

Definición 2.16 (*Función de pertenencia S*). La función de pertenencia S , $S[a, b](x)$, siendo $a \leq b$, permite establecer una transición suave desde un valor de pertenencia nulo

³La interpretación lingüística de las variables borrosas, así como los predicados que se pueden realizar con éstas, se verán con detenimiento en las secciones 2.3.2 y 2.3.3.

**Figura 2.10:** *Función de pertenencia S.*

a un valor de pertenencia uno. Se define según la expresión (2.15).

$$\mu_{S[a,b]}(x) = \begin{cases} 0 & \text{si } x \leq a \\ 2 \left(\frac{x-a}{b-a} \right)^2 & \text{si } a < x \leq \frac{a+b}{2} \\ 1 - 2 \left(\frac{b-x}{b-a} \right)^2 & \text{si } \frac{a+b}{2} < x < b \\ 1 & \text{si } x \geq b, \end{cases} \quad (2.15)$$

La función de pertenencia S es continua y define un conjunto borroso normal y convexo. Su soporte viene dado por el intervalo (a, ∞) , por lo que es infinito. Al no cumplir la condición de límite no puede emplearse para representar números borrosos, sin embargo, esta función de pertenencia es muy utilizada para representar las etiquetas lingüísticas⁴ del tipo *mayor de*.

En la figura 2.10 se muestra un ejemplo de función S , con $a = -2$ y $b = 6$.

Definición 2.17 (*Función de pertenencia Z*). La función de pertenencia Z , $Z[a,b](x)$, siendo $a \leq b$, permite establecer una transición suave desde un valor de pertenencia uno

⁴Las etiquetas lingüísticas se verán con detenimiento en la sección 2.3.2.

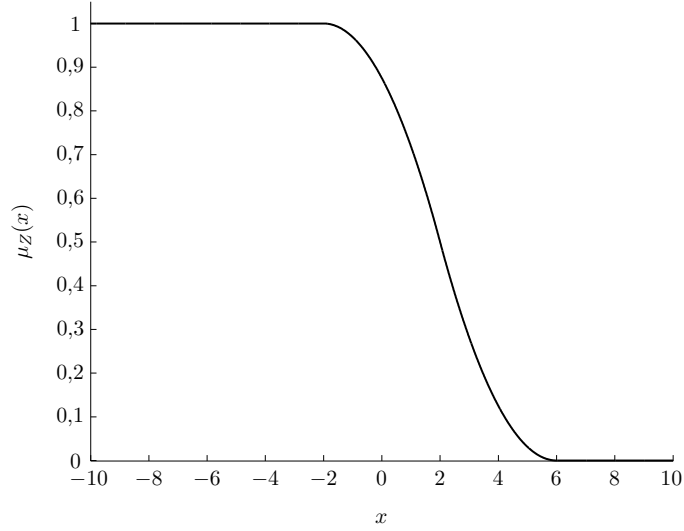


Figura 2.11: *Función de pertenencia Z.*

a un valor de pertenencia nulo. Se define según la expresión (2.16).

$$\mu_Z[a, b](x) = \begin{cases} 1 & \text{si } x \leq a \\ 1 - 2 \left(\frac{x - a}{a - b} \right)^2 & \text{si } a < x < \frac{a+b}{2} \\ 2 \left(\frac{b - x}{a - b} \right)^2 & \text{si } \frac{a+b}{2} \leq x < b \\ 0 & \text{si } x \geq b, \end{cases} \quad (2.16)$$

La función de pertenencia Z es continua y define un conjunto borroso normal y convexo. Su soporte es $(-\infty, b)$, por lo que es infinito. Al no cumplir la condición de límite esta función no puede emplearse para representar números borrosos, sin embargo, esta función de pertenencia es muy utilizada para representar las etiquetas lingüísticas⁵ del tipo *menor de*.

En la figura 2.11 se muestra un ejemplo de función Z , con $a = -2$ y $b = 6$.

Definición 2.18 (*Función de pertenencia Π*). La función de pertenencia Π (Pi), que se puede denotar por $Pi[a, b, c, d](x)$ o por $\Pi[a, b, c, d](x)$, siendo $a \leq b \leq c \leq d$, es una función de pertenencia asimétrica compuesta por el producto de la función $\mu_S[a, b]$ y la función $\mu_Z[c, d]$. Esta función permite establecer una transición suave desde un grado de pertenencia cero a un grado de pertenencia uno mediante la forma S , y una transición descendiente a pertenencia nula mediante la función Z . Otra forma muy común de definir la función Π es utilizando únicamente dos parámetros, resultando en este caso una función simétrica.

⁵Las etiquetas lingüísticas se verán con detenimiento en la sección 2.3.2.

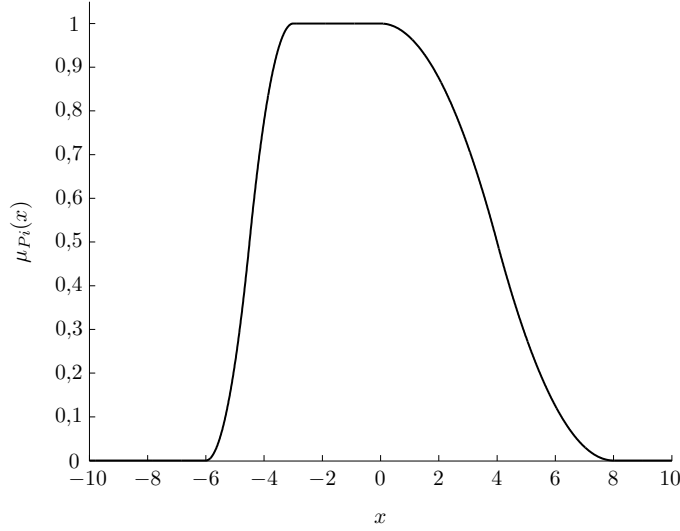


Figura 2.12: Función de pertenencia II.

La función de pertenencia II es continua, define un conjunto borroso normal y convexo. Su soporte es finito, siendo $\text{sup}(A) = (a, d)$, por lo que la función II puede emplearse para representar números borrosos. El núcleo de esta función viene dado por $[b, c]$.

En la figura 2.12 se muestra un ejemplo de función II, con $a = -6$, $b = -3$, $c = 0$ y $d = 8$.

Definición 2.19 (*Función de pertenencia sigmoideal*). Una función de pertenencia sigmoideal, $\text{Sigm}[\alpha, c](x)$, se define según la expresión (2.17), siendo c el punto para el que $\mu_{\text{Sigm}}(c) = 0,5$ y α un parámetro relacionado con la velocidad de transición de la función de pertenencia.

$$\mu_{\text{Sigm}}[\alpha, c](x) = \frac{1}{1 + e^{\alpha(c-x)}} \quad (2.17)$$

Las funciones de pertenencia sigmoideales son continuas y definen un conjunto borroso normal y convexo. Su soporte es infinito y no cumplen la condición de límite, por lo que no pueden emplearse para representar números borrosos.

En la figura 2.13 se representan dos ejemplos de funciones de pertenencia sigmoideales, dadas por los parámetros $\text{Sigm}[-2, -4](x)$ y $\text{Sigm}[2, 1](x)$. Como puede observarse, un valor positivo de α hace que la función tenga una forma parecida a la función S , mientras que un valor negativo provoca que la función sea similar a una función de pertenencia Z .

Existen otras funciones de pertenencia basadas en operaciones con funciones sigmoideales. Las más destacadas son la función diferencia de sigmoideales, obtenida mediante la ecuación (2.18), y la función producto de sigmoideales, que se puede calcular a partir de la ecuación (2.19). En ambos casos, ajustando adecuadamente los parámetros c y α de cada una de las sigmoideales, se pueden obtener funciones tanto normales como no normales, convexas y no convexas, y con soporte finito o funciones de tipo negación de soporte

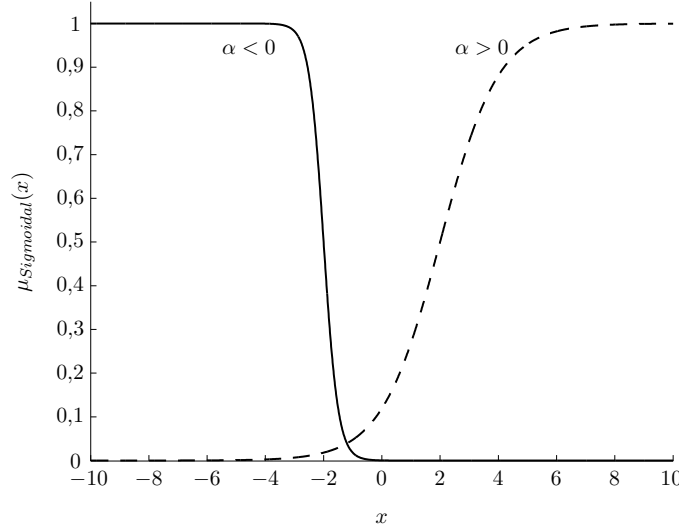


Figura 2.13: Funciones de pertenencia sigmoideas.

infinito. En las figuras 2.14 y 2.15 se pueden ver algunos ejemplos de estas funciones de pertenencia compuestas.

$$\mu_{DSigm}[\alpha_1, c_1, \alpha_2, c_2](x) = \mu_{Sigm}[\alpha_1, c_1](x) - \mu_{Sigm}[\alpha_2, c_2](x) \quad (2.18)$$

$$\mu_{PSigm}[\alpha_1, c_1, \alpha_2, c_2](x) = \mu_{Sigm}[\alpha_1, c_1](x) \cdot \mu_{Sigm}[\alpha_2, c_2](x) \quad (2.19)$$

Definición 2.20 (*Función de pertenencia singleton o punto borroso*). Una función de pertenencia *singleton*, o punto borroso, definida sobre x_0 es aquella función de pertenencia cuyo soporte y núcleo es $\text{sup}(A) = \text{nuc}(A) = x_0$. También se puede utilizar la notación $A = \overline{x_0}$. La función de pertenencia *singleton* es denominada también conjunto borroso escalar.

En la figura 2.16 se muestra a modo de ejemplo la función de pertenencia *singleton* $A = \overline{0,2}$.

Operaciones sobre conjuntos borrosos

Muchas de las operaciones presentes en la lógica clásica de conjuntos se pueden generalizar a la lógica borrosa, como por ejemplo las tres operaciones básicas: la unión, la intersección y el complemento; sin embargo, hay operaciones y propiedades de la lógica clásica que no se cumplen al trasladarlas al ámbito borroso, como la ley del medio excluido y el principio de no contradicción. A continuación se estudiarán algunas de las operaciones básicas que pueden realizarse sobre conjuntos borrosos, así como algunas de sus propiedades.

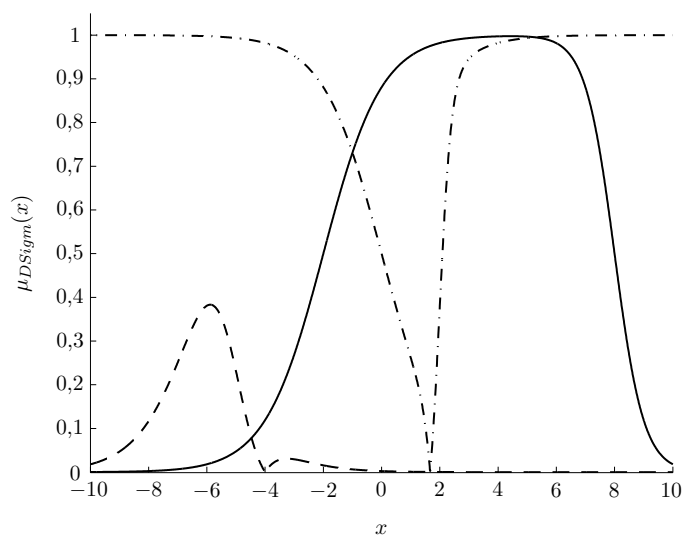


Figura 2.14: *Funciones de pertenencia diferencia de sigmoidales.*

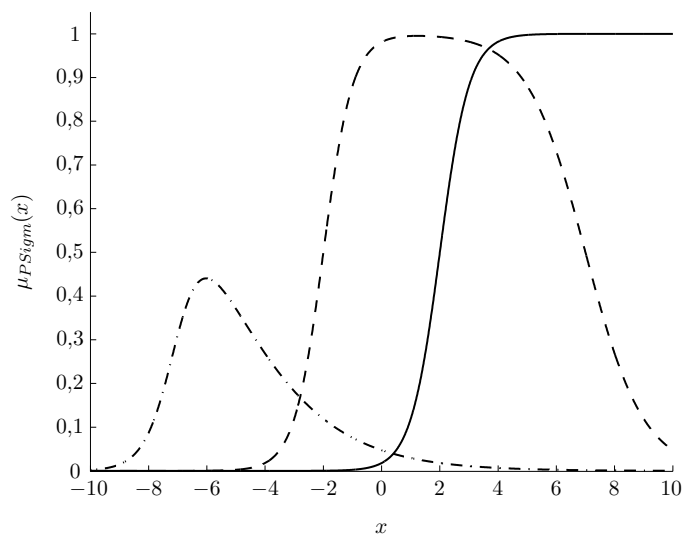


Figura 2.15: *Funciones de pertenencia producto de sigmoidales.*

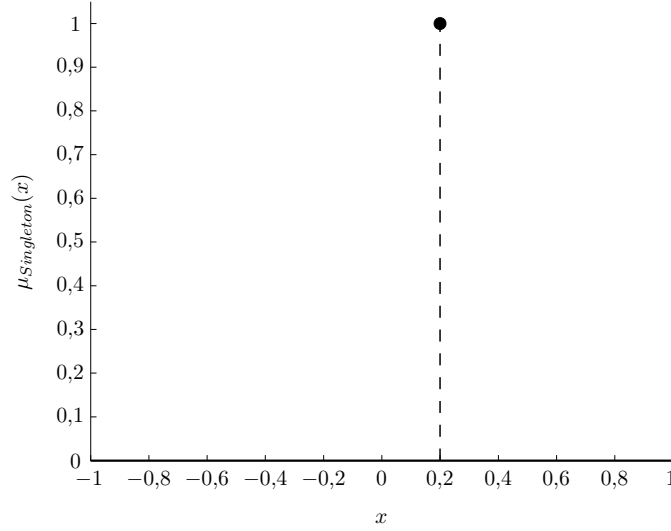


Figura 2.16: *Función de pertenencia Singleton.*

Definición 2.21 (Igualdad). Se dice que dos conjuntos borrosos A y B , con funciones de pertenencia dadas por $\mu_A(x)$ y $\mu_B(x)$, respectivamente, son iguales si y sólo si $\mu_A(x) = \mu_B(x)$ para todo $x \in X$.

$$A = B \Leftrightarrow \mu_A(x) = \mu_B(x), \forall x \in X \quad (2.20)$$

Definición 2.22 (Inclusión). Dados dos conjuntos borrosos A y B , con funciones de pertenencia dadas por $\mu_A(x)$ y $\mu_B(x)$, respectivamente, se dice que A está contenido en B , o de forma equivalente, que A es un subconjunto de B o que A es menor que B , si y sólo si $\mu_A(x) \leq \mu_B(x)$ para todo $x \in X$.

$$\left. \begin{array}{l} A \subset B \\ A \leq B \end{array} \right\} \Leftrightarrow \mu_A(x) \leq \mu_B(x), \forall x \in X \quad (2.21)$$

Definición 2.23 (Intersección). La operación intersección de dos conjuntos borrosos A y B , cuyas funciones de pertenencia son $\mu_A(x)$ y $\mu_B(x)$, respectivamente, se define como un conjunto borroso $C = A \cap B$ cuya función de pertenencia se relaciona con las de A y B mediante la ecuación (2.22). En la figura 2.17 se muestra un ejemplo de la operación intersección entre dos conjuntos borrosos.

$$\mu_C(x) = \mu_{A \cap B}(x) = \min \{ \mu_A(x), \mu_B(x) \}, \forall x \in X \quad (2.22)$$

La ecuación (2.22) también se puede escribir de forma abreviada como:

$$\mu_C = \mu_A \wedge \mu_B. \quad (2.23)$$

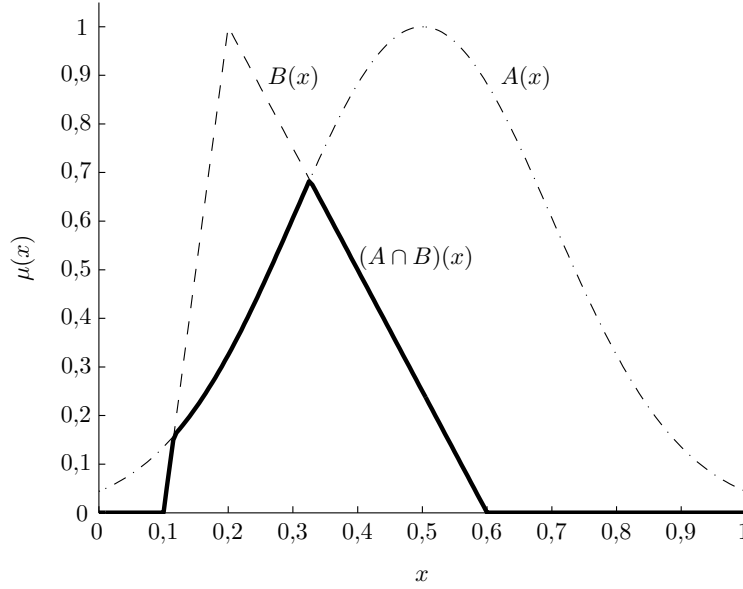


Figura 2.17: *Intersección entre conjuntos borroso.*

Se puede demostrar [1] que si los conjuntos borrosos A y B son convexos, el conjunto intersección C también lo es.

En caso de que la operación de intersección entre A y B resultase un conjunto vacío, es decir, si $A \cap B = \emptyset$, se dice que los conjuntos borrosos A y B son disjuntos.

La operación de intersección definida a través de la función mínimo se suele denominar intersección clásica, ya que fue la definida originalmente por el profesor Zadeh. Posteriormente han aparecido otros operadores de intersección lógica que se pueden englobar en las denominadas T -normas. Estos operadores se verán con mayor detenimiento en la definición 2.26.

Definición 2.24 (Unión). De forma análoga a la operación intersección, la unión de dos conjuntos borrosos A y B , cuyas funciones de pertenencia son $\mu_A(x)$ y $\mu_B(x)$, respectivamente, puede definirse como un conjunto borroso $C = A \cup B$ cuya función de pertenencia se relaciona con las de A y B mediante la expresión (2.24). La función máximo se emplea para definir el operador de unión clásico, sin embargo, la operación de unión lógica puede generalizarse a través de las S -normas, tal como se verá en la definición 2.27. Una muestra de la operación de unión clásica puede verse en la figura 2.18.

$$\mu_C(x) = \mu_{A \cup B}(x) = \max \{ \mu_A(x), \mu_B(x) \}, \forall x \in X \quad (2.24)$$

La ecuación (2.24) también se puede escribir de forma abreviada como:

$$\mu_C = \mu_A \vee \mu_B. \quad (2.25)$$

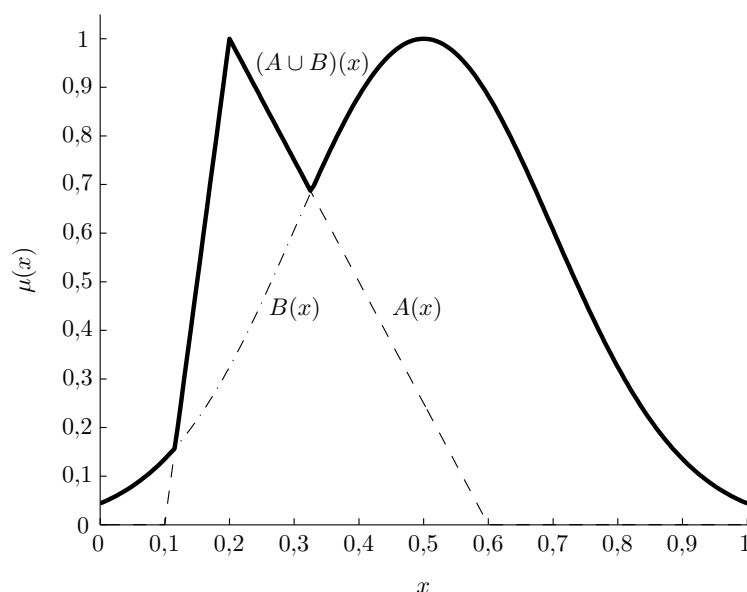


Figura 2.18: Unión de conjuntos borrosos.

Definición 2.25 (*Negación o complemento a uno*). El complemento a uno o negación de un conjunto borroso A , cuya función de pertenencia es $\mu_A(x)$, se representa⁶ por $\overline{A}$, y se define según la expresión (2.26). En la figura 2.19 se muestra un ejemplo de la operación complemento sobre un conjunto borroso definido a través de una función de pertenencia triangular.

$$\mu_{\overline{A}}(x) = 1 - \mu_A(x) \quad (2.26)$$

La operación de complemento a uno es el operador clásico de negación y el más frecuentemente utilizado; no obstante, en la definición 2.28 se presentan otros operadores existentes en la literatura.

Una vez que se han presentado las operaciones lógicas principales, es importante destacar que tanto los conjuntos borrosos como las operaciones de unión, intersección y complemento presentadas anteriormente cumplen las propiedades básicas de sus equivalentes clásicos:

Idempotencia:

$$\begin{aligned} A \cup A &= A \\ A \cap A &= A \end{aligned} \quad (2.27)$$

Identidad:

$$\begin{aligned} A \cup \emptyset &= A \quad \text{y, si } A \subset X, \quad A \cap X = A \\ A \cap \emptyset &= \emptyset \quad \text{y, si } A \subset X, \quad A \cup X = X \end{aligned} \quad (2.28)$$

⁶La operación complemento también se puede encontrar representada de las formas $\neg A$, A^C o A' .

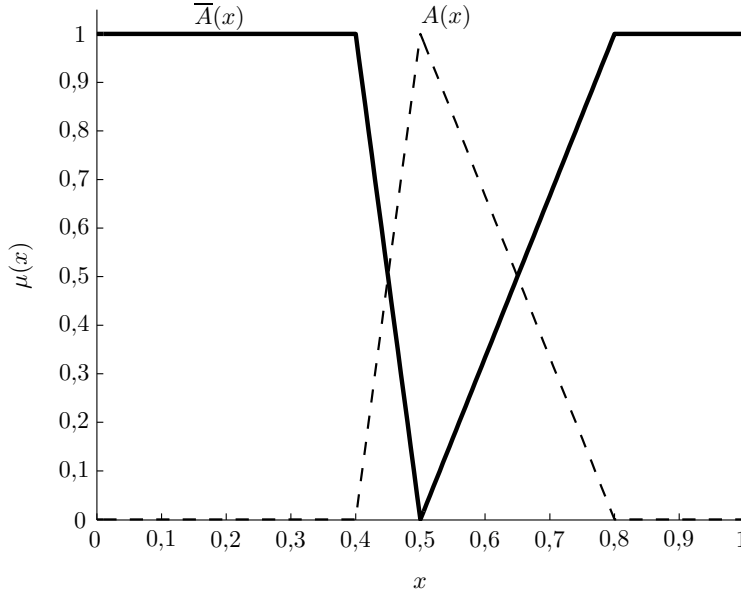


Figura 2.19: Operación negación o complemento a uno.

Transitividad:

$$\text{Si } A \subset B \subset C, \text{ entonces } A \subset C \quad (2.29)$$

Involución:

$$\overline{\overline{A}} = A \quad (2.30)$$

Propiedad conmutativa:

$$\begin{aligned} A \cup B &= B \cup A \\ A \cap B &= B \cap A \end{aligned} \quad (2.31)$$

Propiedad asociativa:

$$\begin{aligned} A \cup (B \cap C) &= (A \cup B) \cap C \\ A \cap (B \cup C) &= (A \cap B) \cup C \end{aligned} \quad (2.32)$$

Con las operaciones y propiedades básicas estudiadas hasta ahora es posible extender a los conjuntos borrosos muchas de las identidades válidas para los conjuntos clásicos. Por ejemplo, se puede comprobar que en la teoría de conjuntos borrosos se cumplen las leyes de De Morgan [1] dadas por la ecuación (2.33), si se consideran los operadores clásicos de unión, intersección y negación propuestos inicialmente por Lofti A. Zadeh (operador máximo, mínimo, y complemento a uno respectivamente). Igualmente, bajo las mismas suposiciones, también se cumplen las leyes distributivas dadas por la ecuación (2.34), es decir, se puede afirmar que los conjuntos borrosos forman un retículo distributivo con

0 y 1 [177].

$$\begin{aligned}\overline{(A \cup B)} &= \overline{A} \cap \overline{B} \\ \overline{(A \cap B)} &= \overline{A} \cup \overline{B}\end{aligned}\tag{2.33}$$

$$\begin{aligned}C \cap (A \cup B) &= (C \cap A) \cup (C \cap B) \\ C \cup (A \cap B) &= (C \cup A) \cap (C \cup B)\end{aligned}\tag{2.34}$$

Tal como se comentó al principio de este epígrafe, no todas las propiedades de los conjuntos clásicos se tienen porqué cumplir en los conjuntos borrosos. Por ejemplo, sea A un subconjunto borroso del conjunto universal⁷ X , $A \subset X$, la ley del medio excluido de conjuntos clásicos asegura que $A \vee \overline{A} = X$, pero si A es un conjunto borroso no normal, $\max\{\mu_A(x), 1 - \mu_A(x)\}$ no será 1 para todo $x \in X$, por lo que $A \vee \overline{A} \neq X$. Lo mismo sucede con el principio de no contradicción. En la lógica clásica se cumple que $A \wedge \overline{A} = \emptyset$, sin embargo en la lógica borrosa $\min\{\mu_A(x), 1 - \mu_A(x)\}$ no tiene porqué ser cero para todo $x \in X$, por lo que no tiene porqué satisfacerse el principio de no contradicción. Si en vez de emplearse el máximo y el mínimo como operadores de unión e intersección, respectivamente, se escogen apropiadamente otras funciones, estas dos propiedades pueden cumplirse en los conjuntos borrosos, sin embargo, lo que se ha querido constatar es que no tienen porqué ser ciertas.

Las funciones lógicas presentadas anteriormente fueron propuestas por el profesor Zadeh en el desarrollo inicial de la teoría de conjuntos borrosos [1], sin embargo, tal como se ha comentado anteriormente, estas funciones pueden ser sustituidas por otras siempre y cuando cumplan determinadas condiciones. A estas funciones que generalizan las operaciones de intersección y unión, derivadas de los conceptos presentados por Menger [178] y Schwizer y Sklar [179], se las conoce como normas y conormas triangulares respectivamente [180, 181, 182].

Además de la función mínimo, existen otros operadores que implementan dicha función, conocidos como normas triangulares o T -normas.

Definición 2.26 (*Normas triangulares o T -normas*). Una norma triangular, o T -norma, es una aplicación $T : [0, 1] \times [0, 1] \rightarrow [0, 1]$ que cumple las propiedades conmutativa, asociativa, de monotonía e identidad unitaria (o condición de frontera). Las T -normas representan un modelo genérico para la intersección lógica (operador lógico Y o AND). Las propiedades conmutativa y asociativa ya han sido introducidas en el texto, ver ecuaciones (2.31) y (2.32). Dados dos pares de valores $x \leq y$, $w \leq z$, la propiedad de monotonía exige que se cumpla que $T(x, w) \leq T(y, z)$. Por último, la condición de frontera que debe cumplir una función para que pueda considerarse como norma triangular (T -norma) es que sea de identidad unitaria, es decir, que $T(x, 1) = x$, para todo $x \in [0, 1]$.

En la tabla 2.1 se muestran algunas de las principales T -normas, y en la figura 2.20 se presentan a modo de ejemplo el resultado de aplicar las T -normas producto y de Lukasiewicz a dos conjuntos borrosos.

⁷Un conjunto universal es aquel que contiene todos los elementos, es decir, todos los elementos tienen un grado de pertenencia a dicho conjunto igual a 1.

Nombre	$T(x, y)$
Mínimo (estándar)	$\min \{x, y\}$
Producto	xy
Producto drástico	$\begin{cases} \min \{x, y\} & \text{si } \max \{x, y\} = 1 \\ 0 & \text{en otro caso} \end{cases}$
Producto acotado (Lukasiewicz)	$\max \{0, (1 + \lambda)(x + y - 1) - \lambda xy\}, \lambda \geq -1$
Hamacher	$\frac{xy}{\lambda + (1 - \lambda)(x + y - xy)}, \lambda \geq 0$
Yager	$1 - \min \left\{ 1, \sqrt[\lambda]{(1 - x)^\lambda + (1 - y)^\lambda} \right\}, \lambda \geq 0$
Dubois-Prade	$\frac{xy}{\max \{x, y, \alpha\}}, \alpha \in (0, 1)$
Frank	$\log_\lambda \left(1 + \frac{(\lambda^x - 1)(\lambda^y - 1)}{\lambda - 1} \right), \lambda > 0, \lambda \neq 1$
Producto de Einstein	$\frac{xy}{1 + (1 - x) + (1 - y)}$

Tabla 2.1: *T-normas principales.*

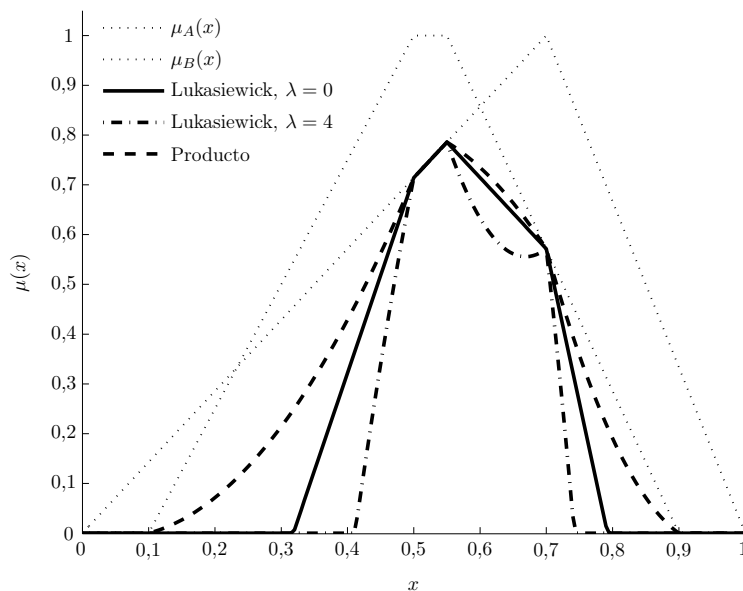


Figura 2.20: *T-normas producto y producto acotado.*

Nombre	$S(x, y)$
Máximo (estándar)	$\max \{x, y\}$
Suma-producto	$x + y - xy$
Suma drástica	$\begin{cases} \max \{x, y\} & \text{si } \min \{x, y\} = 0 \\ 1 & \text{en otro caso} \end{cases}$
Suma acotada (Lukasiewicz)	$\min \{1, (x + y + \lambda xy)\}, \lambda \geq 0$
Hamacher	$\frac{x + y - (2 - \lambda)xy}{1 - (1 - \lambda)xy}, \lambda \geq 0$
Yager	$\min \left\{ 1, \sqrt[\lambda]{x^\lambda + y^\lambda} \right\}, \lambda \geq 0$
Dubois-Prade	$1 - \frac{(1-x)(1-y)}{\max \{(1-x), (1-y), \alpha\}}, \alpha \in [0, 1]$
Frank	$\log_\lambda \left(1 + \frac{(\lambda^{(1-x)} - 1)(\lambda^{(1-y)} - 1)}{\lambda - 1} \right), \lambda > 0, \lambda \neq 1$

Tabla 2.2: S -normas principales.

Nótese que para que una función pueda considerarse T -norma debe cumplir la propiedad asociativa. Esta exigencia permite extender las T -normas a más de dos argumentos. Además, puesto que también deben cumplir la propiedad conmutativa, el orden de aplicación de las T -normas no afecta al resultado final.

Como sucede con la intersección, la unión borrosa también puede definirse mediante otras funciones distintas al máximo. Estas funciones son conocidas como T -conormas o S -normas.

Definición 2.27 (T -conorma o S -norma). Una conorma triangular, también llamada T -conorma o S -norma, es una aplicación $S : [0, 1]^2 \rightarrow [0, 1]$ que cumple las propiedades conmutativa, asociativa, de monotonía e identidad cero (o condición de frontera). Las S -normas representan un modelo genérico para la unión lógica (operador lógico O u OR). En la tabla 2.2 se muestran algunas de las principales S -normas, y en la figura 2.21 se muestra el resultado de aplicar las S -normas *suma-producto* y de *Lukasiewicz* a dos conjuntos borrosos.

Al igual que sucede con las T -normas, las S -normas o T -conormas pueden extenderse a más de dos argumentos empleando su asociatividad, sin importar su orden de aplicación (conmutatividad). La condición de frontera que debe cumplir una función para que pueda considerarse como conorma triangular (S -norma) es que sea de identidad cero, es decir, que $S(x, 0) = x, \forall x \in [0, 1]$.

Para cada T -norma existe una S -norma dual o conjugada y viceversa. Así para una T -norma, $T(x, y)$, su S -norma conjugada sería: $S(x, y) = 1 - T((1 - x), (1 - y))$.

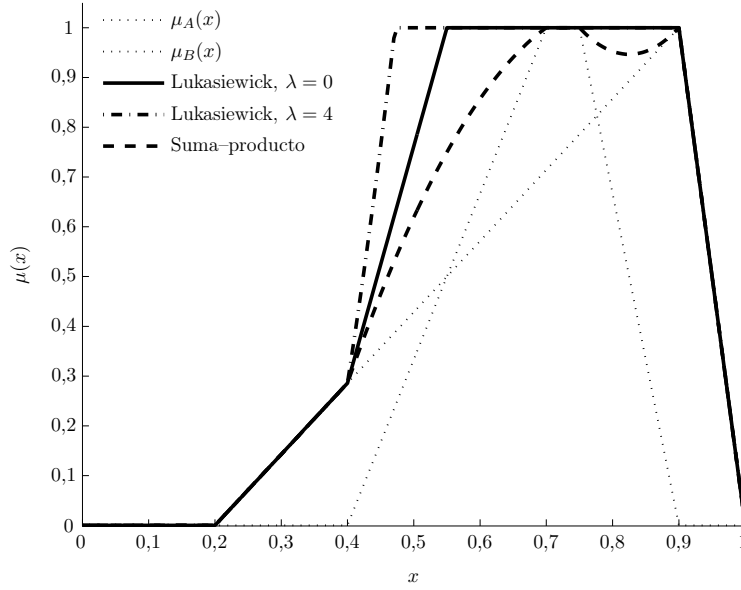


Figura 2.21: S -normas suma-producto y suma acotada.

Aunque las T -normas y S -normas no pueden ordenarse, sí que puede identificarse claramente la mayor y menor de ellas. Así, la mayor de las T -normas es la función mínimo y la menor el producto drástico. En el caso de las S -normas, la mayor es la suma drástica, y la menor la función máximo. El resto de T -normas y S -normas están comprendidas entre éstas.

Con respecto a las propiedades que cumplen las T -normas y S -normas, en general se puede decir que estas operaciones no satisfacen las propiedades de contradicción ni exclusión del medio, a excepción de la familia de *Lukasiewicz* con $\lambda = 0$. Las propiedades de idempotencia y la propiedad distributiva sólo se cumple en el caso de los operadores estándar, es decir, si se emplean el máximo y el mínimo como operadores de unión e intersección, respectivamente.

Como sucede con las operaciones de unión e intersección, el operador de negación también se puede definir de forma distinta a su definición original.

Definición 2.28 (Operador de negación). Una función de negación es una aplicación $N : [0, 1] \rightarrow [0, 1]$ continua, monótona no creciente, involutiva, es decir, $N(N(x)) = x$, para todo $x \in X$, y que cumple las condiciones de frontera $N(0) = 1$ y $N(1) = 0$.

En la tabla 2.3 se presentan algunos operadores de negación y en la figura 2.22 se pueden ver algunos ejemplos. Nótese que si $\lambda = 0$ el operador de Sugeno coincide con la operación de negación estándar e igual sucede si $\omega = 1$ con el operador de Yager.

Junto con las operaciones clásicas se pueden realizar otro tipo de operaciones sobre los conjuntos borroso. A continuación se definen algunas de esas otras operaciones.

Definición 2.29 (α -corte). Sea A un conjunto borroso, cuya función de pertenencia es $\mu_A(x)$, el α -corte de A es un conjunto borroso que se denota por $[A]^\alpha$ y se define según

Nombre	$N(x)$
Estándar	$1 - x$
Umbral	$\begin{cases} 1 & \text{si } x \leq \lambda \\ 0 & \text{si } x > \lambda \end{cases} \quad 0 < \lambda < 1$
Sugeno	$\frac{1 - x}{1 + \lambda x} \quad \lambda > -1$
Yager	$\sqrt[\omega]{1 - x^\omega} \quad \omega > 0$

Tabla 2.3: Operadores de Negación.

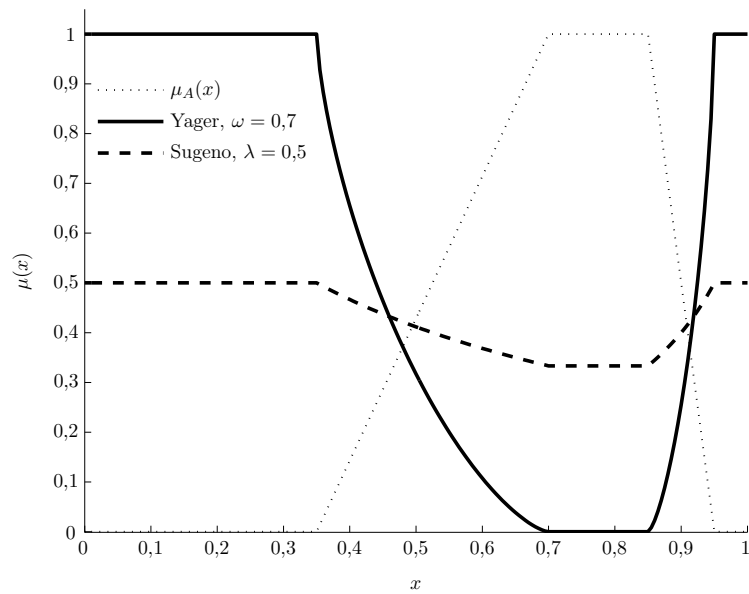


Figura 2.22: Operadores de negación de Yager y Sugeno.

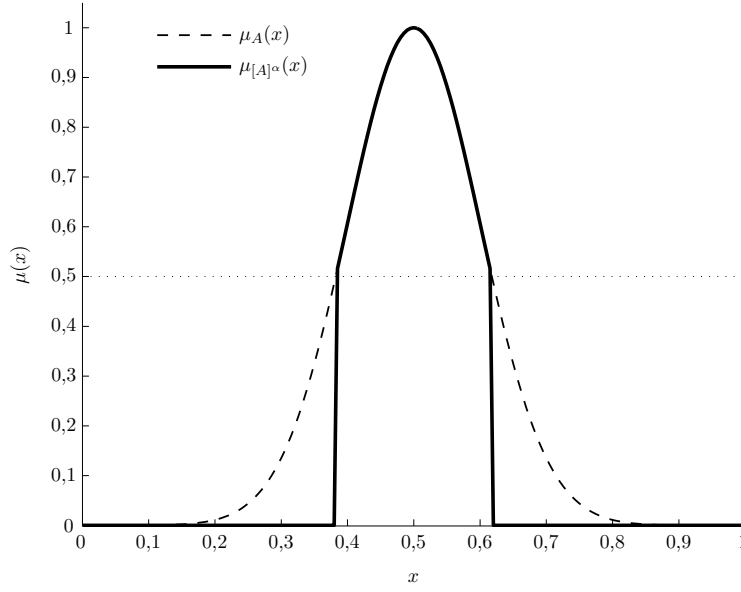


Figura 2.23: α -corte de un conjunto borroso para $\alpha = 0,5$.

la ecuación (2.35). En la figura 2.23 se muestra un ejemplo de conjunto borroso definido a partir de la operación α -corte.

$$[A]^\alpha = \{x \in X | \mu_A(x) > \alpha\}, \alpha > 0 \quad (2.35)$$

Definición 2.30 (Concentración y dilatación). Sea A un conjunto borroso definido a través de la función de pertenencia $\mu_A(x)$, entonces A^k , con $k \in \mathbb{R}^+$, denota la operación de concentración si $k > 1$ o de dilatación si $k \in [0, 1)$. Normalmente se utiliza $k = 2$ para la operación de concentración y $k = 0,5$ para la dilatación. La función de pertenencia del conjunto borroso concentrado o dilatado se obtiene mediante la ecuación (2.36).

$$\mu_{A^k} = \mu_A(x)^k \quad (2.36)$$

Definición 2.31 (Producto algebraico). El producto algebraico de dos conjuntos borrosos A y B , cuyas funciones de pertenencia son $\mu_A(x)$ y $\mu_B(x)$, respectivamente, se denota por AB y se define mediante la ecuación (2.37). El conjunto borroso AB cumple $AB \subset A \cap B$.

$$\mu_{(AB)}(x) = \mu_A(x)\mu_B(x) \quad (2.37)$$

Definición 2.32 (Suma algebraica). La suma algebraica de dos conjuntos borrosos A y B , cuyas funciones de pertenencia son $\mu_A(x)$ y $\mu_B(x)$, respectivamente, se denota por $A + B$ y se define mediante la ecuación (2.38). Si A y B son conjuntos borrosos normalizados, el conjunto borroso obtenido mediante la suma algebraica de A y B sólo será normalizado si $\mu_A(x) + \mu_B(x) \leq 1$ para todo $x \in X$, es decir, a diferencia del producto algebraico,

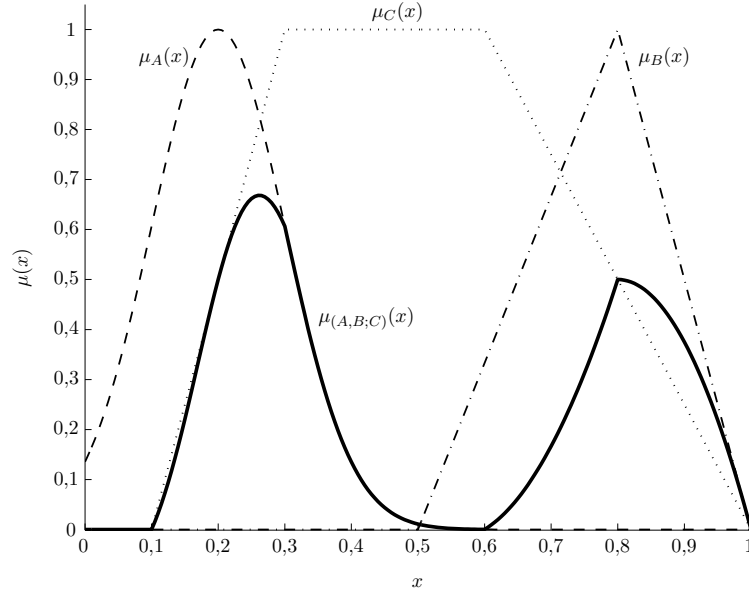


Figura 2.24: *Combinación convexa de dos conjuntos borrosos.*

la suma algebraica de dos conjuntos borrosos sólo tendrá sentido si $\mu_{A+B} \leq 1$ para todo $x \in X$.

$$\mu_{(A+B)}(x) = \mu_A(x) + \mu_B(x) \quad (2.38)$$

Definición 2.33 (Suma). La suma de dos conjuntos borrosos A y B , cuyas funciones de pertenencia son $\mu_A(x)$ y $\mu_B(x)$, respectivamente, se denota $A \oplus B$ y se define mediante la ecuación (2.39). Si A y B son conjuntos borrosos normalizados, el conjunto borroso $A \oplus B$ será normalizado en todo caso, por lo que la operación suma sí tiene sentido en todo el universo de discurso de los conjuntos A y B .

$$\begin{aligned} (A \oplus B) &= A + B - AB \\ \mu_{(A \oplus B)}(x) &= \mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x), \forall x \in X \end{aligned} \quad (2.39)$$

Nótese que el producto algebraico es una T -norma, por lo que es equivalente a la operación de intersección borrosa, sin embargo la suma algebraica no es una T -conorma, por lo que no sería la operación dual al producto algebraico. En lógica borrosa, la operación dual del producto algebraico es la suma, ya que ésta sí es equivalente a la operación unión (es una S -norma).

Definición 2.34 (Combinación convexa). La combinación convexa de dos conjuntos borrosos A y B , con funciones de pertenencia $\mu_A(x)$ y $\mu_B(x)$, respectivamente, a través de un tercer conjunto borroso C , con función de pertenencia $\mu_C(x)$, se denota por $(A, B; C)$, y se define mediante la ecuación (2.40). Un ejemplo de combinación convexa entre dos conjuntos borrosos puede verse en la figura 2.24.

$$\begin{aligned}
(A, B; C) &= CA + \overline{C}B \\
\mu_{(A, B; C)}(x) &= \mu_C(x)\mu_A(x) + (1 - \mu_C(x))\mu_B(x), \forall x \in X
\end{aligned}
\tag{2.40}$$

La combinación convexa de A y B mediante un conjunto borroso C es un conjunto borroso intermedio entre la unión y la intersección de los conjuntos A y B ; es decir, cumple:

$$A \cap B \subset (A, B; C) \subset A \cup B, \forall C \tag{2.41}$$

2.3.2. Variables lingüísticas

Una variable lingüística es una variable cuyos valores son palabras o sentencias en un idioma natural o artificial [183]. El concepto de variable lingüística fue introducido por Zadeh [5] y desarrollado posteriormente por él mismo en varios trabajos [10, 11, 12, 184]. Las variables lingüísticas permiten el acercamiento de la lógica borrosa al lenguaje natural, facilitando la utilización lógica del lenguaje ordinario y eludiendo así las deficiencias del lenguaje preciso en el área del control. Esta idea subyace en los primeros trabajos de Zadeh, y se formalizó finalmente en el enunciado de su conocido Principio de Incompatibilidad [6]:

A medida que la complejidad de un sistema aumenta, disminuye nuestra capacidad para hacer afirmaciones precisas, incluso significativas, sobre su comportamiento, hasta que se alcanza un umbral más allá del cual precisión y relevancia son características casi mutuamente excluyentes.

Tal como se puede ver en la figura 2.25, cada término lingüístico se representa de forma cuantitativa mediante una función de pertenencia o distribución de posibilidad. En este caso, la variable lingüística *edad* se ha representado mediante las etiquetas lingüísticas *niño*, *joven*, *adulto* y *anciano*. Los términos *niño* y *anciano* se han creado utilizando funciones de pertenencia trapezoidales, la etiqueta *joven* mediante una función gaussiana, y *adulto* empleando una gaussiana doble.

Además de los términos lingüísticos primarios, se pueden definir etiquetas derivadas mediante la utilización de modificadores lingüísticos. Siguiendo con el ejemplo anterior, se podría utilizar algún operador de negación de la tabla 2.3 para crear la etiqueta *no adulto*. También se puede utilizar la operación de concentración para crear las etiquetas *más adulto*, *muy adulto* o *bastante adulto*, o el operador de dilatación para definir los términos lingüísticos *menos adulto*, *más o menos adulto* o *algo adulto*. Nótese en los ejemplos de la figura 2.26 que los modificadores lingüísticos basados en concentraciones o dilataciones⁸ del conjunto borroso, no implican un desplazamiento de éste, sino una modificación de su forma. Esto quiere decir que el significado de la variable lingüística *más adulto* del ejemplo no se refiere a una persona de mayor edad, sino a una persona de la que se quiere asegurar su buena pertenencia a dicho conjunto.

Para un mismo universo de discurso, cuanto mayor sea el número de términos lingüísticos empleados para representar una variable, mayor precisión se podrá obtener del sistema

⁸Estos operadores se introdujeron en la definición 2.30.

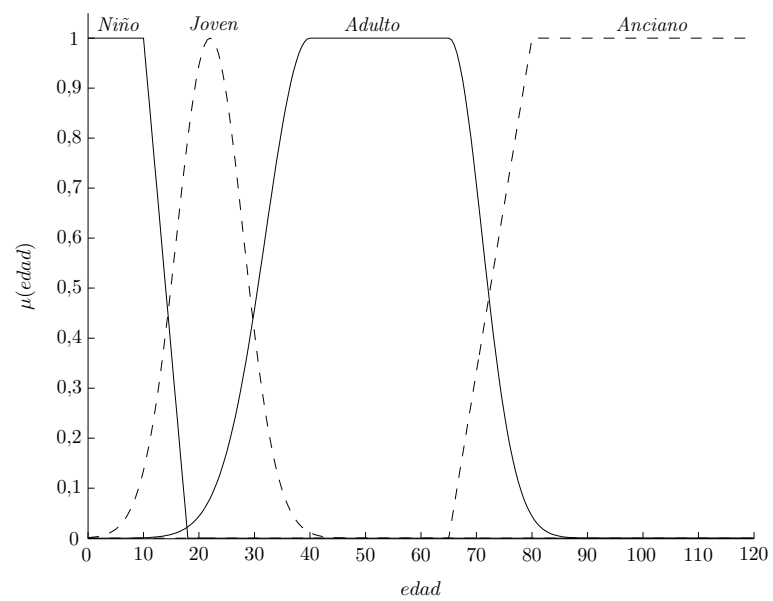


Figura 2.25: Variable lingüística edad.

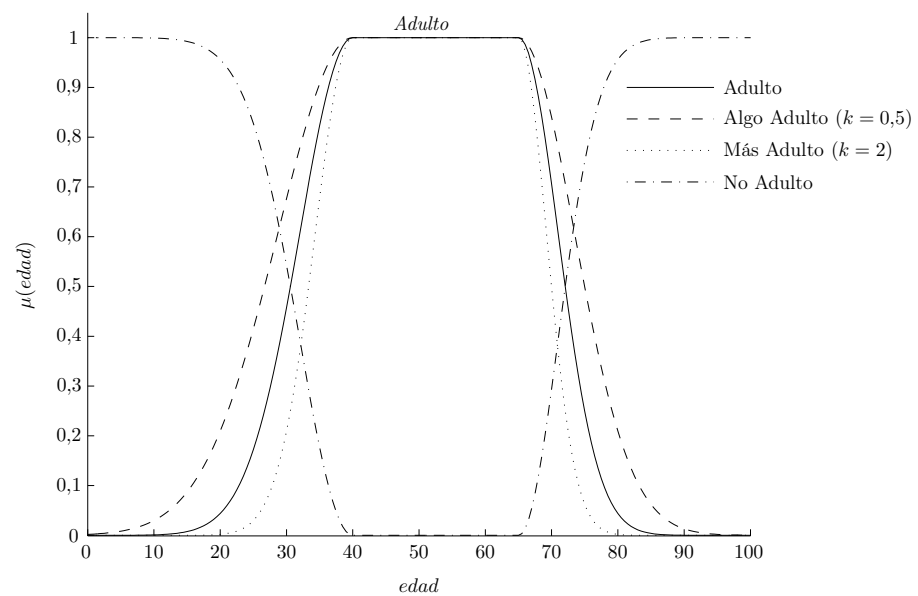


Figura 2.26: Concentración y dilatación de una variable lingüística.

borroso. Evidentemente, al tener un número mayor de etiquetas lingüísticas cada una de ellas representará un subespacio menor, por lo que se podrá caracterizar el sistema más finamente; sin embargo, también tendrá mayor complejidad. Según Miller [185], el número de valores capaces de discriminar un ser humano varía entre cinco y nueve. Otro detalle a tener en cuenta si se desea hacer un tratamiento lingüístico del sistema borroso, es que normalmente se debe escoger un número impar de etiquetas. De esta forma existirá una etiqueta central, *adulto* en la figura 2.26, que hará las veces de neutral, y el resto se distribuirá a sus lados.

Cuando la dimensionalidad del espacio entrada-salida es grande, el número de reglas necesario para producir un recubrimiento suficientemente fino de todo el universo de discurso se incrementa exponencialmente. Esto es lo que se denomina *problema de la dimensionalidad de los sistemas borrosos*. Una forma de reducir este problema consiste en emplear procedimientos automáticos para la selección de reglas, de forma que se tenga más precisión en las zonas que lo requieran sin aumentar excesivamente el número de éstas [186]. En cualquier caso, tanto para la elección del número como para la distribución de las etiquetas lingüísticas se requiere de un análisis detallado del problema.

2.3.3. Regla borrosa

Una regla borrosa es una sentencia condicional que tiene la estructura Si-Entonces (*If-Then*), siendo el conjunto de reglas de un sistema borroso lo que se conoce como *base de reglas* o *base de conocimiento* del sistema. En su forma más simple, una regla borrosa viene dada por la expresión (2.42), aunque esta expresión puede hacerse más compleja mediante la aplicación de operadores de negación, unión, intersección, modificadores lingüísticos, etc. Por ejemplo, una construcción más compleja de una regla borrosa podría ser la mostrada en la ecuación (2.43).

$$\text{Si } x \text{ es } A \text{ entonces } y \text{ es } B \quad (2.42)$$

$$\text{Si la } \textit{temperatura} \text{ es } \textit{alta} \text{ y el } \textit{volumen} \text{ es } \textit{pequeño} \text{ entonces la } \textit{presión} \text{ es } \textit{alta} \quad (2.43)$$

La premisa de la regla, es decir, la condición, se conoce como *antecedente*, mientras que la consecuencia se conoce como *consecuente*. En el ejemplo anterior, “Si la *temperatura* es *alta* y el *volumen* es *pequeño*” se corresponde con el antecedente de la regla, en este caso formado por la unión mediante el operador *y* de dos premisas. Las variables lingüísticas de entrada son *temperatura* y *volumen*, etiquetadas lingüísticamente en esta regla como *alta* y *pequeño*, respectivamente. “La *presión* es *alta*” es el consecuente de la regla, donde *presión* se corresponde con la variable lingüística de salida y *alta* con la etiqueta lingüística que se le asignará si se cumple el antecedente. El consecuente de una regla puede ser un conjunto borroso, un punto borroso, o una función dependiente de las entradas.

El operador que relaciona la premisa con la consecuencia se conoce como *operador de implicación*, y se representa lingüísticamente con el adverbio *entonces* (*then*). Existen muchas maneras de definir el concepto de implicación [187, 188, 189]. Basándose en cada una de ellas se pueden generar distintas funciones en base a *T*-normas y *S*-normas. En la tabla 2.4 se presentan varios operadores de implicación, siendo los más importantes y utilizados los operadores de Mamdani y Larsen.

Nombre	$\mu_{\text{Implicación}}(x, y)$
Mamdani (mínimo)	$\min \{\mu_A(x), \mu_B(y)\}$
Larsen (producto)	$\mu_A(x) \cdot \mu_B(y)$
Reichenbach	$1 - \mu_A(x) + \mu_A(x) \cdot \mu_B(y)$
Lukasiewicz	$\min \{1, 1 - \mu_A(x) + \mu_B(y)\}$
Willmott	$\max \{1 - \mu_A(x), \min \{\mu_A(x), \mu_B(y)\}\}$
Kleene–Dienes	$\max \{1 - \mu_A(x), \mu_B(y)\}$

Tabla 2.4: Operadores de implicación.

Al contrario de lo que sucede en la lógica clásica, la lógica borrosa no requiere que las reglas se cumplan de forma exacta, sino que pueden cumplirse en cierto grado. Esta propiedad de la lógica borrosa es lo que caracteriza al razonamiento aproximado. Esto implica que se puede inferir una conclusión aunque el hecho no verifique la regla plenamente. Evidentemente, si el grado de cumplimiento del antecedente no es completo, el grado de acción del consecuente tampoco lo será. De esta forma, el consecuente final se parecerá más a la conclusión formal de la regla cuanto mayor sea el grado de cumplimiento de ésta.

2.3.4. Sistema de inferencia borroso

Los sistemas de inferencia borrosos son algoritmos basados en la lógica borrosa y el razonamiento aproximado. En función de la aplicación a la que se destine el sistema de inferencia éste puede denominarse de distintas formas: *modelo borroso*, *sistema experto borroso*, *controlador borroso*, *memoria asociativa borrosa*, etc. Los sistemas de inferencia borrosos se componen básicamente de dos elementos, el *motor de inferencia* y la *base de conocimiento*, sin embargo, según la aplicación para que vaya a emplearse el sistema de inferencia, puede ser necesario añadir elementos adicionales, como el *borrosificador* y el *desborrosificador* que permiten conectar al sistema de inferencia borroso con el mundo real [190]. A continuación se estudiarán cada uno de los elementos de un sistema de inferencia borrosa, cuya configuración puede verse en la figura 2.27.

Borrosificador

En muchos casos, especialmente en sistemas de control, las entradas que deben aplicarse a un sistema borroso son valores numéricos concretos, no borrosos, procedentes generalmente de sensores, por lo que es necesario establecer un interfaz con el motor de inferencia borrosa que elabore conjuntos borrosos a partir de entradas no borrosas. Este interfaz se

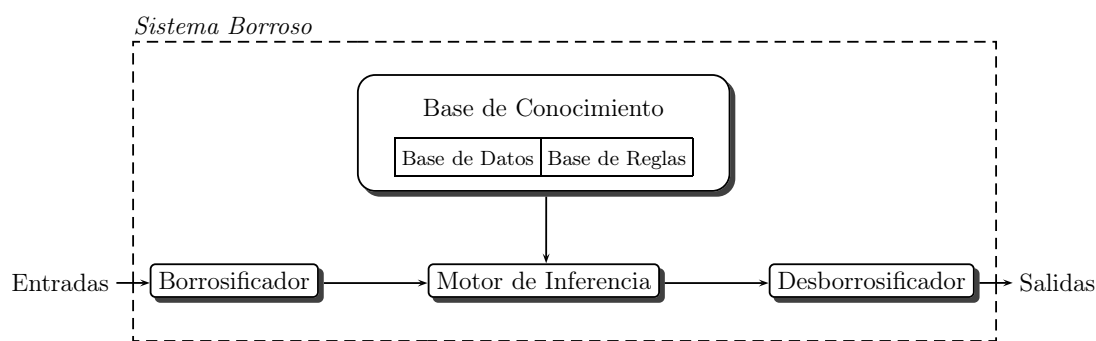


Figura 2.27: Sistema de inferencia borroso.

conoce como borrosificador (*fuzzifier*). Evidentemente, no es necesario emplear un borrosificador si la entrada al sistema es ya un conjunto borroso.

El sistema de borrosificación más simple consiste en sustituir el valor numérico por un conjunto borroso de tipo *singleton*, o punto borroso. Esta borrosificación simplifica de forma notable los cálculos necesarios para la aplicación posterior de la norma de composición. Por ejemplo, un valor de entrada no borroso $x = 11$ puede ser representado por el conjunto borroso $A = \overline{11}$, cuya función de pertenencia se correspondería con la ecuación (2.44).

$$\mu_A(x) = \begin{cases} 1 & \text{si } x = 11 \\ 0 & \text{en otro caso} \end{cases} \quad (2.44)$$

Otro posible borrosificador sería un algoritmo que asignara a cada entrada una representación borrosa mediante términos lingüísticos definidos en su correspondiente universo de discurso. Evidentemente, este esquema es más complejo y aporta poco al sistema, por lo que no es frecuente su utilización. Por último, existe la posibilidad de emplear la capacidad de la lógica borrosa para manejar la incertidumbre como método de borrosificación. En este caso, se sustituye la variable numérica de entrada por un conjunto borroso que represente mediante una distribución de posibilidad (función de pertenencia) los posibles valores que ésta podría tomar realmente. Este borrosificador resulta especialmente útil si hay que tratar con un continuo de medidas dentro de un rango, o si se desea incorporar la incertidumbre propia de toda medida al sistema de control, por ejemplo para considerar el ruido o los errores del sistema de instrumentación; sin embargo, supone una mayor complejidad del sistema de inferencia que debe ser considerada.

Inferencia borrosa

El motor de inferencia borrosa es el mecanismo que permite obtener la salida de un sistema borroso en función de sus reglas y las entradas que le sean aplicadas. Esta salida puede ser tanto un conjunto borroso como un valor numérico, según se requiera por la aplicación. Para realizar la inferencia se deben seguir los pasos siguientes:

1. En primer lugar es necesario calcular el grado de cumplimiento de cada antecedente en función de la entrada del sistema. Si se utiliza un borrosificador tipo *singleton*,

este valor se obtendrá directamente de la intersección de la entrada con la función de pertenencia del antecedente; mientras que si la entrada está representada por una función de pertenencia no *singleton*, será necesario aplicar un operador de intersección, una T -norma, sobre dichos conjuntos.

2. Una vez calculados todos los antecedentes de cada regla, se componen según la definición lingüística de la misma, es decir, aplicando los operadores de intersección para condiciones Y , y los de unión para condiciones O . Esta medida proporciona un valor que representa la calidad con la que las reglas son satisfechas por la condición de entrada del sistema borroso, esto es, su grado de activación.
3. Mediante el operador de implicación, *Entonces*, se obtiene el consecuente resultante de cada regla. En aplicaciones relacionadas con el control de procesos, los operadores de implicación más utilizados son las T -normas mínimo y producto. Al emplear el operador mínimo, la función de pertenencia del consecuente es recortada a la altura definida por el grado de veracidad de la regla; mientras que si se emplea el operador producto, la salida es escalada según dicho grado de veracidad.
4. Por último, la regla composicional u operador de agregación, combina los resultados de todas las reglas, esto es, el resultado de la base de reglas, en un único conjunto borroso. El máximo y la suma acotada suelen ser las S -normas más utilizadas para obtener el conjunto borroso de salida.
5. Si la aplicación requiere que la salida del sistema borroso sea numérica, se necesitará aplicar un desborrosificador a la misma.

El motor de inferencia puede verse matemáticamente como un esquema de interpolación no lineal, ya que permite la fusión de la información de múltiples reglas en una conclusión única. Esta conclusión se obtiene a partir de la combinación de los consecuentes de las reglas del sistema, ponderadas según una función del grado de cumplimiento de cada una de ellas. Desde el punto de vista lógico, la inferencia borrosa puede verse como una generalización del esquema clásico de razonamiento *modus ponens*⁹.

En la figura 2.28 se muestra un ejemplo de inferencia tanto para una entrada borrosa pura (Entrada 1), como para un punto borroso (Entrada 2). El grado de activación de la primera regla se obtiene mediante el máximo valor de la intersección clásica entre el conjunto borroso de entrada y el del antecedente. Este valor es propagado hacia el consecuente de la regla, obteniéndose el conjunto borroso de salida de ésta mediante el operador de implicación producto. Para la segunda regla, el grado de activación se obtiene evaluando la función de pertenencia del Antecedente 2 en el punto dado por la Entrada 2. Este grado de activación es propagado hacia su consecuente, obteniéndose mediante el operador de implicación mínimo el conjunto borroso de salida de la segunda regla. El conjunto borroso final resultado de las dos reglas, obtenido con el operador de conjunción máximo, puede observarse en la figura 2.29.

⁹El *modus ponens* es la regla de inferencia básica en el cálculo proposicional. Dado un postulado lógico de la forma “Si A entonces B ”, si se demuestra cierta la premisa antecedente, es decir, si la sentencia A es cierta, se afirmará inmediatamente la premisa consecuente, es decir, B también lo será.

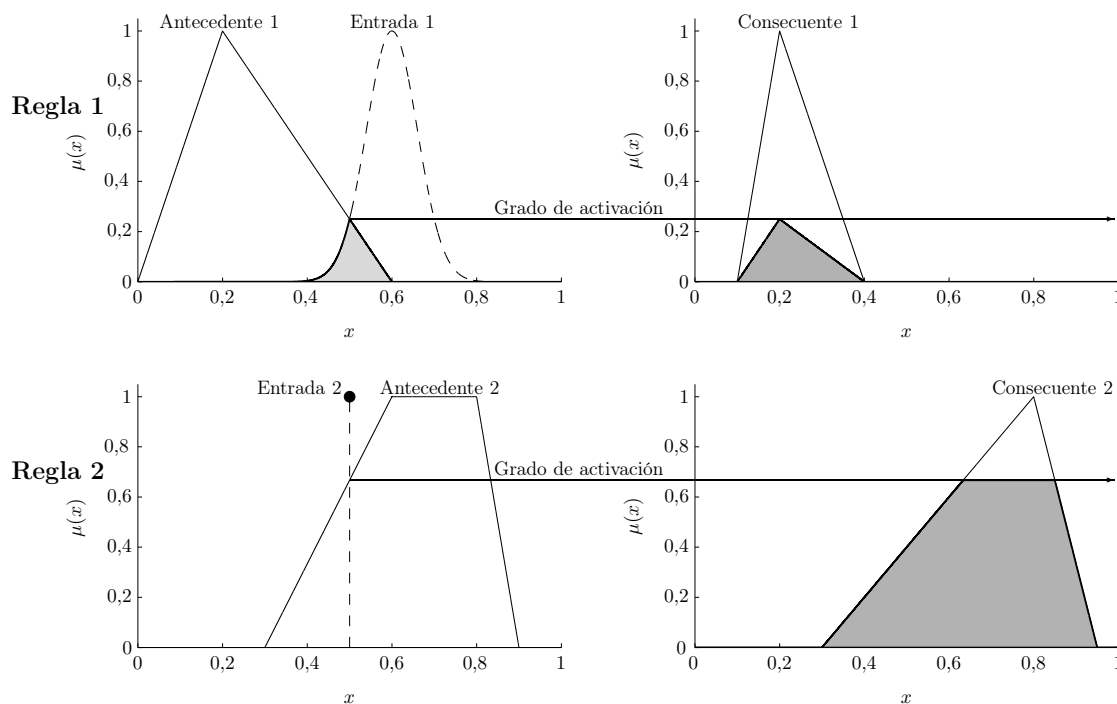


Figura 2.28: Ejemplo de inferencia.

Base de conocimiento

La base de conocimiento (*knowledge base*) almacena las relaciones existentes entre las entradas y salidas del sistema. Basándose en este conocimiento, el proceso de inferencia obtendrá las salidas asociadas a las entradas del mismo. Este elemento puede subdividirse en dos partes: la base de datos y la base de reglas. La base de datos (*data base*) contiene la definición de las variables lingüísticas empleadas en las reglas, es decir, las funciones de pertenencia que definen cada una de las etiquetas lingüísticas. La base de reglas (*rule base*) contiene la colección de reglas lingüísticas del sistema. Ésta puede representarse como una lista de reglas, que es la forma más habitual, o mediante una tabla o matriz de decisión, que proporciona un formato más compacto si existen pocas variables. Existen varios métodos para la obtención de las reglas de un sistema borroso, entre los que cabe destacar:

- Basados en el conocimiento de un experto, o heurísticos. Generalmente se obtiene su conocimiento mediante unos cuestionarios cuidadosamente organizados que permitan la generación de reglas del tipo Si-Entonces.
- Basados en las acciones de control de un operador, en función de los datos de entrada-salida observados durante un período suficientemente amplio de operación.
- Basados en aprendizaje mediante algoritmos automáticos.

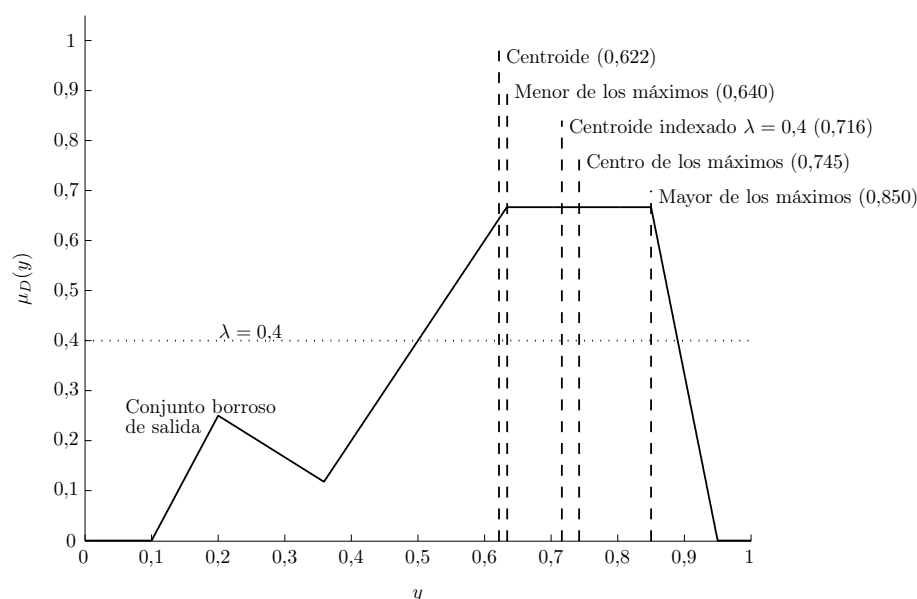


Figura 2.29: Ejemplo de desborrosificación.

Desborrosificador

Cuando la salida del sistema de inferencia es un conjunto borroso pero la aplicación requiere que sea un valor numérico concreto, es necesario emplear un desborrosificador (*defuzzifier*). Este elemento convierte dicho conjunto en un valor numérico del universo de discurso de salida que será representativo de la conclusión obtenida.

Existen muchos métodos de desborrosificación [190, 191] y aún sigue siendo frecuente la aparición de nuevos métodos en la literatura [192, 193, 194, 195, 196]. A continuación se presentarán los desborrosificadores más comunes, cuya representación gráfica, respecto del conjunto borroso final de salida a desborrosificar, puede verse en la figura 2.29.

1. *Método del centro de gravedad o del centroide.* La salida numérica de este desborrosificador se obtiene calculando el centro de gravedad del conjunto borroso de salida. Siendo $D(y)$ el conjunto borroso de salida, su valor desborrosificado mediante este método y_o , se obtiene a partir de la ecuación (2.45). En la ecuación (2.46) se muestra la versión para valores discretos. Aunque es más costoso computacionalmente que otros métodos de desborrosificación, este método es sin duda el más empleado, ya que proporciona variaciones suaves y continuas de los valores de salida.

$$y_o = \frac{\int y \mu_D(y) dy}{\int \mu_D(y) dy} \quad (2.45)$$

$$y_o = \frac{\sum y \mu_D(y)}{\sum \mu_D(y)} \quad (2.46)$$

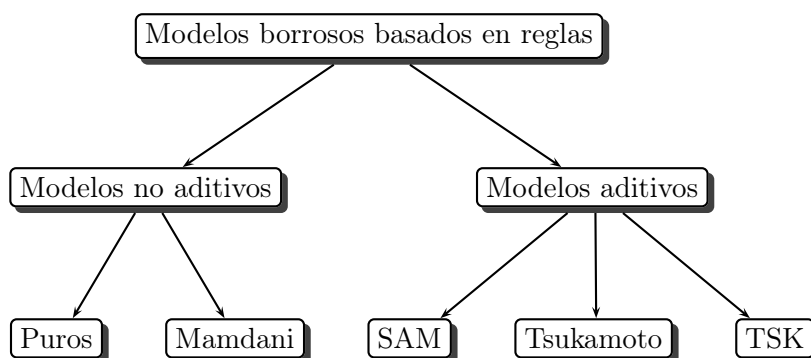


Figura 2.30: Tipos de modelos borrosos

2. *Método de la máxima pertenencia.* Siendo $D(y)$ el conjunto borroso de salida, el valor numérico y_o obtenido mediante el desborrosificador del máximo se corresponde con el valor de y para el que $\mu_D(y)$ alcanza su máximo valor. Puesto que pueden existir múltiples valores para los que $\mu_D(y)$ tenga un mismo valor máximo, se hace necesario escoger algún método para elegir el valor de y_o . Estos métodos son (ver figura 2.29): el *menor de los máximos*, el *centro de los máximos* y el *mayor de los máximos*. El desborrosificador de la máxima pertenencia se caracteriza por ser de muy fácil implementación, sin embargo, genera cambios bruscos en la salida que en muchas aplicaciones no son admisibles.
3. *Método del centroide indexado.* Este método calcula el centro de gravedad de la parte correspondiente al conjunto borroso inferido cuyo grado de pertenencia sea mayor de un determinado valor λ .

2.3.5. Tipos de sistemas borrosos

En función de la forma de las reglas y del tipo de entradas y salidas, se distinguen cuatro tipos de sistemas borrosos [14, 197], relacionados esquemáticamente como se muestra en la figura 2.30: los sistemas puros, los sistema de tipo Mamdani, los de Sugeno o TSK y los modelos aditivos de Kosko o SAM, junto con algunas variantes como los modelos de Tsukamoto.

Sistema borroso puro

Los sistemas borrosos puros son sistemas cuyas entradas y salidas son conjuntos borrosos. Al no necesitar realizar ninguna transformación sobre las entradas o salidas, están compuestos tan sólo por la base de conocimiento y el motor de inferencia.

Modelo Mamdani

En 1974 E. H. Mamdani estableció las bases para la aplicación de la lógica borrosa al control de sistemas [198]. Posteriormente, junto con S. Assilian, consiguieron controlar una

máquina de vapor mediante un conjunto de reglas lingüísticas obtenidas de la experiencia de los operadores de la máquina [28]. De esta forma demostraron la potencia práctica de la teoría borrosa de Zadeh con la creación del primer sistema borroso aplicado a un problema de control. Desde entonces, el esquema de control borroso que utilizaron [199] se conoce como modelo de control de Mamdani–Assilian, o más frecuentemente, como modelo de Mamdani. El esquema de control borroso de Mamdani consta, según se muestra en la figura 2.27, de un motor de inferencia, una base de conocimiento y unos interfaces tanto de borrosificación como de desborrosificación para procesar las entradas y salidas del sistema.

Los sistemas borrosos de tipo Mamdani presentan una serie de ventajas que propician su utilización en el ámbito del control:

- Los modelos de tipo Mamdani se comportan como aproximadores universales [200].
- Pueden utilizarse en aplicaciones reales, ya que tratan con facilidad entradas y salidas reales.
- Proporcionan un marco natural para la inclusión del conocimiento de expertos en forma de reglas lingüísticas.
- Existe gran libertad a la hora de escoger el método de inferencia borrosa, así como los interfaces de borrosificación y desborrosificación.

Por el contrario, también poseen una serie de limitaciones [197]:

- Falta de flexibilidad debido a la rigidez con que se particionan los espacios de entrada y salida.
- No existe una distinción clara entre el conocimiento experto y la definición de las variables lingüísticas incluidas en las reglas borrosas.
- Cuando las variables de entrada al sistema dependen unas de otras, es muy complicado obtener una partición borrosa adecuada de los espacios de entrada.
- El tamaño de la base de conocimiento depende directamente del número de variables y términos lingüísticos que existan en el sistema.
- La interpretabilidad del sistema está ligada a la utilización de un número no excesivo de reglas, lo cual generalmente va en contra de la precisión del sistema borroso implementado.

La base de reglas de un sistema Mamdani se configura mediante un conjunto de reglas de la forma:

$$\text{Regla } r : \text{ Si } x_1 \text{ es } A_1^r \text{ y } x_2 \text{ es } A_2^r \text{ y } \dots \text{ y } x_n \text{ es } A_n^r \text{ entonces } y \text{ es } B^r, \quad (2.47)$$

donde $x_1, x_2, \dots, x_n$ son las entradas del sistema, $A_1^r, A_2^r, \dots, A_n^r$ son los conjuntos borrosos del antecedente de la regla r , $r = 1..R$, definidos en los universos de discurso de sus entradas asociadas, y es la salida del sistema, y B^r el conjunto borroso consecuente de la regla r ,

definido en el universo de discurso de la salida y . En este tipo de sistemas, tanto las entradas x_i , $i = 1..n$, como la salida y son valores numéricos.

El grado de activación de la regla, ω^r , se obtiene mediante la aplicación del operador de intersección, que normalmente será alguna de las T -normas presentadas: $\omega^r = \alpha_1^r \wedge \alpha_2^r \wedge \dots \wedge \alpha_n^r$, donde cada α_i^r representa el grado de emparejamiento entre la entrada x_i y el conjunto borroso antecedente A_i^r . El conjunto borroso de salida de la regla que genera el motor de inferencia, $\mu_{B'^r}$, viene dado por el operador de implicación: $\mu_{B'^r} = \omega^r \wedge \mu_{B^r}$, siendo μ_{B^r} la función de pertenencia que define al conjunto borroso consecuente B^r . La salida borrosa del motor de inferencia, μ_B , se calcula agregando mediante una regla composicional, normalmente una S -norma, los efectos de todas las reglas del sistema: $\mu_B = \bigcup_{r=1}^R (\mu_{B'^r})$. Finalmente, el desborrosificador calcula la salida numérica del sistema, y . Nótese que en los sistemas de tipo Mamdani tanto los antecedentes como los consecuentes están formados por conjuntos borrosos, mientras que las entradas y salidas son valores numéricos.

Se ha demostrado que los sistemas borrosos de tipo Mamdani son aproximadores universales [200, 201, 202, 203, 204]. Uniendo la interpretabilidad de sus reglas a la capacidad aproximadora universal de estos modelos, se tiene que los modelos borrosos de Mamdani son una herramienta tremendamente poderosa para el análisis y control de sistemas complejos. No obstante, una buena aproximación con un modelo borroso de tipo Mamdani puede requerir de un número elevado de reglas, por lo que la interpretabilidad del sistema queda mermada conforme aumenta su complejidad.

Modelo aditivo estándar, SAM

El modelo aditivo estándar, SAM por sus siglas en inglés (*Standard Additive Model*), es un caso particular de los modelos borrosos aditivos. Fue propuesto por Bart Kosko [205, 206] y su esquema de bloques puede verse en la figura 2.31. La principal ventaja de los modelos aditivos es su eficiencia, ya que las reglas pueden precomponerse en un formato menos costoso computacionalmente.

El esquema de inferencia de un sistema SAM es similar al de los modelos TSK que se verán posteriormente, ya que ambos generan su salida en base a la suma ponderada de las consecuencias de cada una de las reglas de la base de conocimiento. Sin embargo, la estructuras de las reglas de un sistema SAM es idéntica a las de los modelos de Mamdani, por lo que podrían considerarse como una variación de éstos. No obstante, existen cuatro diferencias fundamentales entre los esquemas de inferencia de los sistemas SAM y Mamdani [14]:

1. En los modelos SAM la entrada (x) se considera siempre como un punto borroso.
2. Para la inferencia se utiliza el operador producto, frente al operador mínimo que suele emplearse en los modelos de Mamdani.
3. Frente al operador máximo que emplean los modelos de Mamdani, en los modelos SAM la combinación final (D) de las conclusiones parciales de cada regla borrosa (D'^r), se realiza mediante una suma ponderada según los pesos ω . Esta forma de

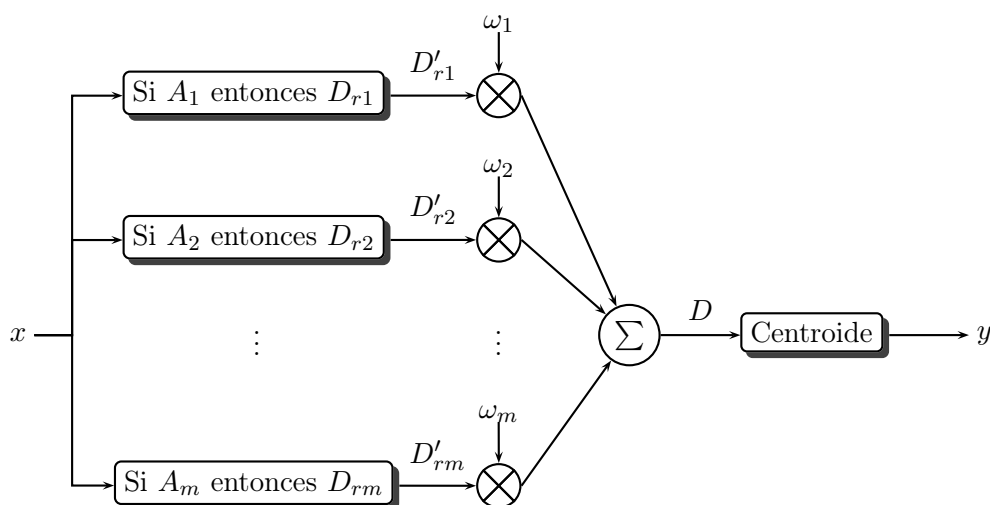


Figura 2.31: Diagrama de bloques de un modelo aditivo estándar (SAM).

operar sitúa a los modelos SAM más cerca estructuralmente de los modelos TSK que de los Mamdani.

4. La salida (y) de un modelo aditivo estándar se calcula mediante la adición de las salidas parciales ponderadas, mientras que en los de Mamdani se debe especificar un desborrosificador.

Modelo de Tsukamoto

Este modelo, propuesto por Tsukamoto en 1979 [207], se caracteriza por representar el consecuente de cada regla borrosa mediante un conjunto borroso con una función de pertenencia monótona. La salida inferida por cada regla es un valor numérico inducido por el grado de activación de la regla. La salida global se toma como la media ponderada de todas las salidas parciales. Este tipo de modelos permiten eliminar el consumo de tiempo correspondiente a la desborrosificación, sin embargo, no son modelos muy utilizados por no ser tan transparentes como los de Mamdani o Sugeno. El mecanismo de inferencia no sigue estrictamente la regla composicional de inferencia y la salida es siempre numérica, incluso cuando las entradas son borrosas.

Modelo Takagi–Sugeno–Kang (TSK)

Los investigadores japoneses Takagi y Sugeno [208], con la posterior incorporación de Kang [209], propusieron un tipo de sistema borroso que en lugar de emplear reglas completamente lingüísticas, utiliza como consecuente una función de las variables de entrada del sistema. Este tipo de sistemas se conoce como modelos Takagi–Sugeno–Kang (TSK), modelos Takagi–Sugeno (TS) o de forma más simplificada, modelos Sugeno. Una regla

típica de un sistema TSK tiene la forma:

$$\text{Regla } r : \text{ Si } x_1 \text{ es } A_1^r \text{ y } x_2 \text{ es } A_2^r \text{ y } \dots \text{ y } x_n \text{ es } A_n^r \text{ entonces } y^r = g^r(\mathbf{x}), \quad (2.48)$$

donde x_i , $i = 1..n$, son las entradas del sistema e y^r es la salida inferida por la regla r . Los elementos A_i^r son los conjuntos borrosos del antecedente de la regla r , definidos en los universos de discurso de sus entradas asociadas. La consecuencia inferida a partir de la regla se calcula con la función $g^r(\mathbf{x})$, dependiente de las entradas del sistema.

La función consecuente, $g^r(\mathbf{x})$, puede escogerse libremente, aunque los consecuentes más utilizados en este tipo de sistemas son los polinomios de grado 0 y de grado 1. La utilización de polinomios de órdenes mayores o de funciones no polinómicas añaden complejidad al sistema y dificultan su interpretabilidad, por lo que no es muy común su utilización. Si la función consecuente es un polinomio de orden 0, se denomina *consecuente afín*, y las reglas tienen la forma dada por la ecuación (2.49); mientras que si se emplea un *consecuente lineal*, o de orden 1, las reglas tienen la forma dada en (2.50). Los elementos $b_0^r, b_1^r, \dots, b_n^r$ son parámetros reales, siendo b_i^r los términos que multiplican a la variable i -ésima del sistema, y b_0^r el término afín de la regla r . El consecuente TSK crea un hiperplano n -dimensional, donde los elementos b_i^r caracterizan el hiperplano, y el término afín se comporta como un *offset* o desplazamiento de la salida, y^r . Por ejemplo, en un sistema de orden 1, $n = 1$, el consecuente TSK tiene la forma $y^r = b_0^r + b_1^r x_1$, donde $b_1^r x_1$ crea una recta de pendiente b_1^r , y el término afín b_0^r es el valor de la salida y^r en $x_1 = 0$.

$$\text{Si } x_1 \text{ es } A_1^r \text{ y } x_2 \text{ es } A_2^r \text{ y } \dots \text{ y } x_n \text{ es } A_n^r \text{ entonces } y^r = b_0^r \quad (2.49)$$

$$\text{Si } x_1 \text{ es } A_1^r \text{ y } x_2 \text{ es } A_2^r \text{ y } \dots \text{ y } x_n \text{ es } A_n^r \text{ entonces } y^r = b_0^r + b_1^r x_1 + \dots + b_n^r x_n \quad (2.50)$$

La salida de un sistema TSK se calcula mediante la suma ponderada o el promedio ponderado del efecto individual de todas sus reglas. La suma ponderada es la suma de los valores obtenidos por la función consecuente, ponderadas por el grado de activación de cada una de las reglas. El promedio ponderado consiste en dividir la suma ponderada por el sumatorio de las fuerzas de activación de las reglas. Estos métodos de agregación permiten, de forma matemáticamente sencilla, que las reglas que se cumplen en mayor grado afecten más a la salida del sistema que las que se cumplen con una menor calidad. Si la base de reglas está formada por R reglas y ω^r es el grado de cumplimiento de la regla r , la salida aplicando la suma ponderada o el promedio ponderado, se obtiene mediante la aplicación de las ecuaciones (2.51) ó (2.52), respectivamente.

$$y = \sum_{r=1}^R \omega^r y^r \quad (2.51)$$

$$y = \frac{\sum_{r=1}^R \omega^r y^r}{\sum_{r=1}^R \omega^r} \quad (2.52)$$

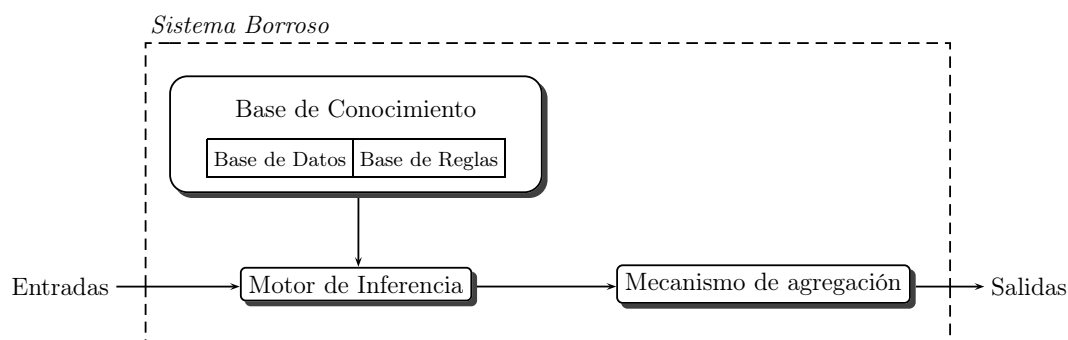


Figura 2.32: Sistema borroso de Takagi-Sugeno-Kang (TSK).

El grado de cumplimiento de cada regla, ω^r , se obtendrá mediante la aplicación de un operador de conjunción que se modela mediante una T -norma; normalmente la función mínimo o producto: $\omega^r = T \{A_1^r(x_1), A_2^r(x_2), \dots, A_n^r(x_n)\}$.

La estructura de un sistema borroso TSK, mostrada en la figura 2.32, es ligeramente distinta de la estructura típica vista en los apartados anteriores. Dispone como todo sistema borroso de un motor de inferencia y una base de conocimiento, pero como las entradas al sistema son siempre numéricas, y su efecto sobre las salidas se calcula directamente mediante la aplicación de las ecuaciones anteriores, no es necesaria la utilización de un borrosificador.

La salida de cada una de las reglas es un valor numérico, de forma que tampoco se requiere de desborrosificador para generar la salida del sistema. No obstante, es imprescindible algún mecanismo para agregar todas las conclusiones parciales de las reglas en una única salida. Para ello se utilizan la suma ponderada o el promedio ponderado. Aunque es muy común nombrar a esta operación como desborrosificador, sobre todo al promedio ponderado que se le suele denominar desborrosificador centro promedio, esto no es estrictamente correcto, ya que su entrada no es un conjunto borroso.

Los sistemas borrosos TSK son una poderosa herramienta para el modelado de sistemas complejos y altamente no lineales, ya que son aproximadores universales tanto de la función [210, 211, 212, 213] como de su derivada [214, 215, 216, 217]. Además, si se utiliza como consecuente un polinomio de orden 1 completo, es decir, con el término afín distinto de cero, un sistema TSK también puede aproximar de forma universal la 2ª derivada de una función [218].

La utilización en los sistemas TSK de un consecuente polinómico permite aproximar funciones con gran precisión y con un número mucho menor de reglas que los sistemas de tipo Mamdani. No obstante, este aumento de precisión se consigue a costa de empeorar la interpretabilidad de las reglas.

Como los consecuentes funcionales empleados en los sistemas borrosos de tipo TSK son difíciles de definir por un experto, para la obtención de este tipo de modelos borrosos se suele recurrir a sistemas automáticos de aprendizaje, como los algoritmos basados en redes

neuronales o los algoritmos bioinspirados (algoritmos basados en colonias de hormigas, algoritmos genéticos, etc.).

2.4. Conclusiones y comentarios bibliográficos

A lo largo de este capítulo se ha visto como la lógica borrosa se ha ganado el reconocimiento del que goza en la actualidad en base a los excelentes resultados que ha obtenido, tanto en el ambiente académico e investigador como en el industrial. Con los investigadores japoneses al frente casi desde sus inicios, y la posterior incorporación de investigadores de todo el Mundo, la lógica borrosa se ha convertido en una tecnología de vanguardia, útil y eficaz en multitud de aplicaciones de muy diversos ámbitos. El desarrollo inicial de la lógica borrosa, liderado fundamentalmente por su creador, Lofti A. Zadeh, puede consultarse en las referencias [1] a [16], mientras que el proceso de formalización y adaptación de esta teoría al ámbito industrial y de consumo puede documentarse en las referencias [17] a [36], y [37] a [61], respectivamente. En las referencias [62] a [174] se pueden consultar algunos de los trabajos realizados por investigadores españoles sobre lógica borrosa.

En este capítulo también se ha visto como, tomando como base la imprecisión del razonamiento humano pero bajo un planteamiento matemático, la lógica borrosa es capaz de generar la respuesta a una situación basándose en el conocimiento adquirido de ésta, aunque éste sea inexacto o incompleto. Los fundamentos que permiten esta inferencia se recogen en los trabajos [175] a [197].

Como se ha visto, la lógica borrosa se basa en el concepto de conjunto borroso. En este capítulo se han estudiado dichos conjuntos, tanto la forma de definirlos a partir de su función de pertenencia, como algunas de las operaciones que se pueden realizar con ellos. Posteriormente se ha estudiado el concepto de variable lingüística, introducido por Zadeh en [5], el cual permite transformar el lenguaje natural en conocimiento cuantitativo utilizable desde el punto de vista matemático, pero sin las restricciones que impone el formalismo preciso de la lógica clásica.

Por último, se han presentado los principales tipos de sistemas borrosos existentes, así como los métodos de inferencia de cada uno de ellos. La descripción del modelo borroso de tipo Mamdani puede consultarse en las referencias [198] a [204], en las referencias [205] y [206] las de los modelos de tipo SAM y en [207] las de los modelos de tipo Tsukamoto. Se ha prestado especial atención a los sistemas Takagi–Sugeno–Kang (TSK), aproximadores universales de cualquier función y su derivada, tal como se demuestra en las referencias [208] a [218], ya que estos serán la herramienta básica del resto de capítulos de este texto.

De entre las referencias incluidas en este capítulo, cabe destacar por su especial relevancia el primer trabajo del profesor Zadeh, [1], que dio inicio a la lógica borrosa, así como [5], donde el propio Zadeh introdujo el concepto de variable lingüística. También es necesario destacar el trabajo [28] de Assilian y Mamdani por constituir un punto de inflexión en la utilización de la lógica borrosa a nivel industrial, y los trabajos de [208] y [209] de Takagi, Sugeno y Kang porque en base a ellos se crearon los modelos borrosos de tipo TSK.

Modelado borroso de sistemas dinámicos

OBTENER un modelo del sistema que se pretende estudiar suele ser uno de los primeros pasos que ha de afrontar el ingeniero de control. Este capítulo está estructurado en cinco secciones que pretenden cubrir los conocimientos necesarios para la obtención de un modelo borroso para un sistema dinámico. En la primera sección se realiza una introducción al concepto de modelo de estado y se presentan algunas definiciones importantes. La segunda sección se centra en el modelado borroso de estado de un sistema de control completamente general. Para ello se estudia el modelo borroso de estado de la planta, el del controlador y el del sistema de control borroso en lazo cerrado, todo ello sin aplicar ningún tipo de restricción ni al tamaño del vector de estado ni al del vector de control. En cada uno de los casos se presenta un algoritmo que permite realizar el cálculo de dichos sistemas. La tercera sección está dedicada a las tres fases de la identificación de un modelo borroso, a saber: la identificación de la estructura, la estimación de sus parámetros, y la validación del modelo obtenido. Dentro de la fase de estimación, de entre las muchas metodologías existentes, el texto se centra en dos técnicas híbridas que han demostrado su potencia y versatilidad: la metodología neuroborrosa y la metodología bioinspirada híbrida (algoritmos bioinspirados + gradiente descendente). En la cuarta sección se presenta el software desarrollado para facilitar el almacenamiento de todos los datos de un modelo borroso, y en la quinta y última sección se recogen las principales conclusiones del capítulo, así como una revisión de las referencias bibliográficas incluidas en el mismo.

3.1. Introducción

Un modelo es una representación conceptual o física de un proceso o sistema, con el fin de analizar su naturaleza, desarrollar o comprobar hipótesis o supuestos y permitir una mejor comprensión del fenómeno real al cual el modelo representa. Es casi imposible obtener un modelo perfecto 100 % veraz de un sistema, pero tampoco tiene porqué ser necesario. La complejidad de los sistemas reales se simplifica en los modelos que los representan como forma de facilitar su obtención o la comprensión de su funcionamiento. El esfuerzo para la

obtención de un modelo debe ser coherente con el uso que se le pretenda dar y la aplicación a la que se vaya a dedicar, es decir, el modelo final debe ser aquel que suponga el mínimo esfuerzo posible de modelado y que cumpla los requerimientos de error de la aplicación.

Los modelos se pueden clasificar como [219]: mentales, verbales, físicos y matemáticos:

- Los *modelos mentales* son intuitivos y permiten predecir el comportamiento de sistemas cotidianos en base a la experiencia acumulada. Su carácter es difícil de formalizar por su propia naturaleza.
- Los *modelos verbales* son cualitativos, y representan mediante palabras las relaciones entre las entradas y salidas de un sistema. Los modelos borrosos lingüísticos pueden considerarse modelos de tipo verbal, aunque realmente contienen un trasfondo matemático que los puede situar indistintamente en esta categoría o en la de modelos matemáticos.
- Los *modelos físicos* son representaciones a escala de sistemas reales que se desarrollan para investigar el comportamiento de éstos en entornos controlados, como un laboratorio, o para reducir los riesgos o costes que implicaría realizar dichas pruebas sobre el sistema real. El modelado físico de todos los fenómenos involucrados en el sistema suele ser muy complejo, y el coste de este tipo de modelos es, en general, elevado.
- Los *modelos matemáticos* son aquellos donde las relaciones entre las distintas variables del sistema se expresan mediante relaciones matemáticas. Estos modelos pueden llegar a ser tremendamente precisos, pero como contraprestación, son poco interpretables por la complejidad matemática que pueden alcanzar. En ingeniería, y particularmente en el campo de la ingeniería de control, los modelos empleados son prácticamente siempre de tipo matemático, no sólo por la exactitud que presentan, sino por la posibilidad de emplear las matemáticas como herramienta de análisis sobre el sistema a través de su modelo.

Los modelos borrosos, tal como se ha comentado anteriormente, son modelos matemáticos que pueden representarse verbalmente mediante la utilización de reglas lingüísticas. Esta propiedad confiere a los modelos borrosos una ventaja sobre otro tipo de modelos, ya que conjuga la precisión matemática con la interpretabilidad. Aunque precisión e interpretabilidad son criterios mutuamente excluyentes, mediante la utilización de la lógica borrosa se pueden obtener modelos que conjuguen de forma equilibrada simplicidad, interpretabilidad y exactitud. Los modelos de tipo Mamdani son excelentes en los casos en los que se desea priorizar la interpretabilidad sobre la exactitud del modelo, mientras que si es la precisión la característica más deseada, los modelos borrosos de tipo TSK serían la elección adecuada. En este capítulo se estudiarán ambos casos, aunque se prestará especial atención a los modelos de tipo TSK, ya que en los capítulos posteriores el estudio se centrará fundamentalmente en este tipo de sistemas borrosos.

3.1.1. Modelo de estado de un sistema dinámico

El primer paso para abordar el análisis riguroso de un sistema dinámico consiste en deducir su modelo. El modelo de estado de un sistema representa, mediante una serie de ecuaciones, las relaciones entre las variables internas del sistema, o estados, y las entradas y salidas del mismo. El hecho de emplear variables internas propicia que el modelo de estado de un sistema también sea denominado descripción interna del mismo.

Independientemente del número de variables de estado del sistema y de la complejidad de su dinámica, los modelos de estado tienen la gran ventaja de ser perfectamente válidos tanto para sistemas lineales como para sistemas no lineales, con una o múltiples entradas y salidas, y tanto si se estudian en tiempo continuo como en instantes muestreados [220, 221, 222, 223, 224].

Definición 3.1 (*Estado*). El estado de un sistema dinámico es la menor colección de variables, denominadas *variables de estado*, cuyo valor en un determinado instante resume el pasado dinámico del sistema, y es suficiente, junto con las entradas aplicadas al mismo, para predecir su evolución futura.

Definición 3.2 (*Variable de estado*). Una variable de estado es cada una de las componentes $x_i(t)$, $i = 1..n$, que conforman el vector de estado $\mathbf{x}(t)$. Como se ve en la ecuación (3.1), este vector se representa en forma de columna, siendo cada variable de estado un elemento de dicho vector y n el orden del sistema, o lo que es lo mismo, el tamaño del vector de estado.

$$\mathbf{x}(t) = \begin{pmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{pmatrix} \quad (3.1)$$

Las variables de estado de un sistema pueden ser magnitudes físicas dependientes de su dinámica, como posición, velocidad, tensión, fuerza, ..., o por el contrario, pueden ser magnitudes matemáticas abstractas sin sentido físico asociadas a las ecuaciones del sistema. La selección de las variables de estado no es única, por lo que el modelo de estado de un sistema tampoco lo es. Desde el punto de vista matemático, el vector de estado forma un espacio vectorial de orden n . Cualquier conjunto de n vectores linealmente independientes puede representar el mismo espacio, por lo que mediante un cambio de base pueden obtenerse infinitos modelos de estado de un mismo sistema. Esta diversidad no es un problema, sino que representa una gran ventaja de las representaciones internas, ya que proporcionan una gran flexibilidad a los modelos de estado, por ejemplo, para facilitar demostraciones matemáticas sobre los mismos.

Definición 3.3 (*Modelo de estado*). El modelo de estado de un sistema dinámico representa la variación de las variables de estado del sistema frente al estado actual y las señales de control aplicadas al mismo. Para el caso genérico de un sistema dinámico no lineal multivariable, su modelo de estado viene dado por las ecuaciones (3.2) ó (3.3), según que el

sistema sea de tiempo continuo o discreto, respectivamente. Estas ecuaciones también se conocen como *ecuaciones de estado*.

$$\begin{aligned} \dot{x}_1(t) &= f_1(x_1(t), x_2(t), \dots, x_n(t), u_1(t), u_2(t), \dots, u_m(t)) \\ \dot{x}_2(t) &= f_2(x_1(t), x_2(t), \dots, x_n(t), u_1(t), u_2(t), \dots, u_m(t)) \\ &\vdots \\ \dot{x}_n(t) &= f_n(x_1(t), x_2(t), \dots, x_n(t), u_1(t), u_2(t), \dots, u_m(t)) \end{aligned} \quad (3.2)$$

$$\begin{aligned} x_1(k+1) &= \bar{f}_1(x_1(k), x_2(k), \dots, x_n(k), u_1(k), u_2(k), \dots, u_m(k)) \\ x_2(k+1) &= \bar{f}_2(x_1(k), x_2(k), \dots, x_n(k), u_1(k), u_2(k), \dots, u_m(k)) \\ &\vdots \\ x_n(k+1) &= \bar{f}_n(x_1(k), x_2(k), \dots, x_n(k), u_1(k), u_2(k), \dots, u_m(k)) \end{aligned} \quad (3.3)$$

Las funciones $f_1, f_2, \dots, f_n$ y $\bar{f}_1, \bar{f}_2, \dots, \bar{f}_n$ de estas expresiones son funciones genéricas, por lo que pueden ser no lineales, dependientes de las variables de estado $x_i, i = 1..n$, y las entradas del sistema, $u_j, j = 1..m$.

La ecuación (3.2) puede representarse de forma compacta:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \quad (3.4)$$

siendo $\mathbf{x}(t)$ el vector de estado, $\mathbf{u}(t)$ el vector de entrada o control, y $\dot{\mathbf{x}}(t)$ la variación temporal del vector de estado representada por su primera derivada con respecto al tiempo.

De igual forma, la ecuación de estado de un sistema discreto, (3.3), puede representarse de la forma compacta siguiente:

$$\mathbf{x}(k+1) = \bar{\mathbf{f}}(\mathbf{x}(k), \mathbf{u}(k)), \quad (3.5)$$

donde $\mathbf{x}(k)$ es el vector de estado y $\mathbf{u}(k)$ el vector de entrada o control, ambos en tiempo discreto. $\mathbf{x}(k+1)$ es el valor que tomará el vector de estado en el siguiente instante de muestreo.

Si además de las variables de estado interesa modelar las salidas del sistema, entonces éstas se representan mediante una función de las entradas y las propias variables de estado. El vector de salidas del sistema se representa mediante las ecuaciones (3.6) y (3.7), siendo $\mathbf{y}(t)$ e $\mathbf{y}(k)$ las salidas para un sistema de tiempo continuo o discreto, respectivamente.

$$\mathbf{y}(t) = \mathbf{h}(\mathbf{x}(t), \mathbf{u}(t)) \quad (3.6)$$

$$\mathbf{y}(k) = \bar{\mathbf{h}}(\mathbf{x}(k), \mathbf{u}(k)) \quad (3.7)$$

Nótese que las ecuaciones de salida no contienen dinámica alguna, sólo una transformación entre el espacio de las variables de estado y el de las salidas del sistema, ya que toda

la dinámica queda reflejada en la ecuación de estado. Por esta razón, la ecuación de salida se obviará en los cálculos y desarrollos posteriores.

Definición 3.4 (*Estado de equilibrio*). Los estados de equilibrio de un sistema dinámico, también denominados *puntos singulares*, son aquellos estados del sistema en los que la variación del vector de estado es nula, es decir, los estados en los que el sistema permanece inalterable mientras el sistema no sea perturbado. Un estado de equilibrio $\mathbf{x}_{eq}$ de un sistema dinámico debe satisfacer las expresiones (3.8) ó (3.9), según sea un sistemas de tiempo continuo o de tiempo discreto, respectivamente. $\mathbf{u}_{eq}$ es el vector de control constante para el estado de equilibrio $\mathbf{x}_{eq}$.

$$\dot{\mathbf{x}}(t) = \mathbf{0} = \mathbf{f}(\mathbf{x}_{eq}, \mathbf{u}_{eq}) \quad (3.8)$$

$$\mathbf{x}(k+1) = \mathbf{x}(k) = \bar{\mathbf{f}}(\mathbf{x}_{eq}, \mathbf{u}_{eq}) \quad (3.9)$$

Si el sistema objeto de estudio fuese un sistema de control realimentado, las señales de control se podrían obtener a partir de las variables de estado, de forma que las expresiones anteriores se reducirían a

$$\dot{\mathbf{x}}(t) = \mathbf{0} = \mathbf{f}(\mathbf{x}_{eq}) \quad (3.10)$$

y

$$\mathbf{x}(k+1) = \mathbf{x}(k) = \bar{\mathbf{f}}(\mathbf{x}_{eq}), \quad (3.11)$$

respectivamente.

Nota. Con objeto de facilitar la comprensión y sencillez de la notación, en lo sucesivo la dependencia explícita del tiempo de las variables de estado y control se empleará sólo en los casos en los que se desee destacar, omitiéndose en el resto de expresiones.

3.2. Modelo de estado de un sistema de control borroso

Un sistema de control (ver figura 3.1) está formado por la planta, o sistema que se desea controlar, el controlador y los elementos auxiliares que permiten la correcta interconexión de dicho conjunto, como los sensores, actuadores, acondicionadores de señal, etc. El funcionamiento dinámico de estos elementos puede modelarse mediante sistemas borrosos. El modelo borroso de la planta deberá reflejar mediante reglas la dinámica del sistema real, manteniendo un compromiso entre la precisión del modelo y su complejidad en función de los requisitos de la aplicación. El controlador borroso deberá inferir, en función del estado del sistema, las acciones correctoras que deben aplicarse sobre la planta para que ésta se comporte según unos requisitos pre-establecidos. Si se obtienen los modelos matemáticos equivalentes tanto de la planta como del controlador borroso, será posible calcular un equivalente matemático del sistema de control en lazo cerrado. La obtención del modelo matemático de lazo cerrado permite el estudio de la estabilidad y desempeño del sistema en función de distintos criterios de calidad.

Una vez que se ha introducido de forma breve el modelado un sistema dinámico mediante su ecuación de estado, se procederá a representar dichas ecuaciones a través de modelos

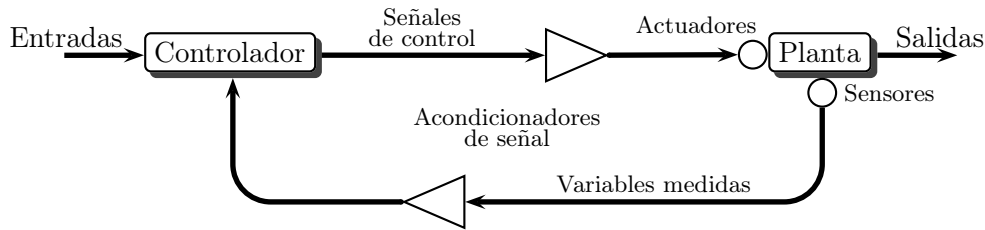


Figura 3.1: Sistema de control.

borrosos. La forma de obtener la estructura del modelo y el valor de los parámetros se verá posteriormente, concretamente en la sección 3.3.

Tal como se vio en el capítulo 2, un modelo borroso representa las relaciones entre sus entradas y sus salidas con reglas de tipo Si-Entonces (*If-Then*), por lo tanto, la forma de representar la dinámica de un sistema empleando la lógica borrosa pasa por establecer reglas que representen las ecuaciones de estado y salida del mismo. Es importante recordar que tanto los modelos borrosos de tipo Mamdani como los de tipo TSK son aproximadores universales, en el caso de los TSK incluso de la segunda derivada, por lo que los modelos borrosos pueden representar de forma fiable dinámicas complejas.

A continuación se estudiará la representación de sistemas de control mediante modelos borrosos de tipo Mamdani y TSK, tanto para sistemas de tiempo continuo como de tiempo discreto. Se prestará especial atención a los sistemas borrosos TSK de tiempo continuo, por ser éstos sistemas parte fundamental del resto de capítulos del presente texto.

3.2.1. Sistema borroso de tipo TSK

Modelo borroso de la planta

Un sistema dinámico de tiempo continuo completamente general puede ser representado por la ecuación (3.2), o de forma más compacta por la ecuación (3.4). Estas ecuaciones representan un modelo sin limitaciones en el tamaño del vector de estados ni en el vector de control, e igualmente sin restricciones en el tipo de funciones que regula la dinámica del sistema, por lo que éstas pueden ser no lineales.

Sea n el orden del sistema y m el número de entradas de éste, un modelo borroso equivalente múltiple entrada y múltiple salida –MIMO (*Multiple Input Multiple Output*)– de un proceso de tiempo continuo, puede ser representado mediante el siguiente conjunto de reglas [208, 225, 226, 227] (base de reglas denominada R_p para enfatizar que se trata del proceso o planta):

$$\begin{aligned}
Rp^{(l,i)} : \text{ Si } & x_1 \text{ es } A_{1i}^l \text{ y } x_2 \text{ es } A_{2i}^l \text{ y } \dots \text{ y } x_n \text{ es } A_{ni}^l \\
& \text{ y } u_1 \text{ es } B_{1i}^l \text{ y } u_2 \text{ es } B_{2i}^l \text{ y } \dots \text{ y } u_m \text{ es } B_{mi}^l \\
\text{Entonces } & \dot{x}_i^l = \mathcal{F}_i^l(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}_i^l),
\end{aligned} \tag{3.12}$$

donde $l = 1..M_i$ es el índice de la regla y M_i el número de reglas que modelan la i -ésima ecuación diferencial del proceso, $\dot{x}_i$. Nótese que empleando esta representación, la dinámica de cada una de las variables de estado se puede modelar mediante un número distinto de reglas, lo que facilita la reducción del número de reglas totales necesarias para modelar correctamente un sistema complejo. Este hecho también facilita el proceso de modelado, ya que se puede abordar de forma independiente para cada $\dot{x}_i$ sin tener en cuenta el número de reglas que deben emplearse.

Los elementos A_{ki}^l son los conjuntos borrosos de los antecedentes definidos en el universo de discurso de sus respectivas variables de estado, x_k , $k = 1..n$, y los elementos B_{ji}^l son los conjuntos borrosos definidos en el universo de discurso de la j -ésima entrada del sistema, u_j , $j = 1..m$.

La función $\mathcal{F}_i^l(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}_i^l)$ representa el consecuente de la l -ésima regla para la ecuación de estado i -ésima, donde $\boldsymbol{\theta}_i^l$ es el vector de parámetros adaptables del consecuente. Empleando un consecuente lineal con término afín, éste tendrá la forma dada en (3.13), y $\boldsymbol{\theta}_i^l$ será $(a_{0i}^l, a_{1i}^l, \dots, a_{ni}^l, b_{1i}^l, b_{2i}^l, \dots, b_{mi}^l)$.

$$\mathcal{F}_i^l(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}_i^l) = a_{0i}^l + a_{1i}^l x_1 + \dots + a_{ni}^l x_n + b_{1i}^l u_1 + b_{2i}^l u_2 + \dots + b_{mi}^l u_m \tag{3.13}$$

Si se utiliza el promedio ponderado como método de agregación, la salida del modelo borroso generada por el conjunto de todas las reglas $Rp^{(l,i)}$, que modelan la dinámica de la coordenada x_i del vector de estado, puede calcularse a través de la expresión (3.14) [228, 229], donde $w_i^l(\mathbf{x}, \mathbf{u})$ representa el grado de activación de las reglas de la planta.

$$\dot{x}_i = \frac{\sum_{l=1}^{M_i} w_i^l(\mathbf{x}, \mathbf{u}) \mathcal{F}_i^l(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}_i^l)}{\sum_{l=1}^{M_i} w_i^l(\mathbf{x}, \mathbf{u})} \tag{3.14}$$

Si las funciones de pertenencia del universo de discurso de las variables de estado están definidas por $\mu_{ki}^l(x_k, \boldsymbol{\sigma}_{ki}^l)$, las funciones de pertenencia del universo de discurso de las entradas de control por $\mu_{ji}^l(u_j, \boldsymbol{\alpha}_{ji}^l)$, y se emplea el operador de inferencia producto, el grado de cumplimiento de las reglas de la planta, $w_i^l(\mathbf{x}, \mathbf{u})$, se calcula mediante la ecuación (3.15). Los vectores $\boldsymbol{\sigma}_i^l$ y $\boldsymbol{\alpha}_i^l$ representan los conjuntos de parámetros adaptables de los antecedentes de las reglas de la planta en los universos de discurso de las variables de estado y las señales de control, respectivamente. Nótese que estas funciones de pertenencia definen los conjuntos borrosos A_{ki}^l y B_{ji}^l del antecedente de las reglas definidas en (3.12).

$$w_i^l(\mathbf{x}, \mathbf{u}) = \prod_{k=1}^n \mu_{ki}^l(x_k, \boldsymbol{\sigma}_{ki}^l) \prod_{j=1}^m \mu_{ji}^l(u_j, \boldsymbol{\alpha}_{ji}^l) \tag{3.15}$$

En la ecuación (3.15) se puede observar que tanto las funciones de pertenencia relativas a las variables de estado, como las relativas a las variables de control, poseen tres índices. Estos índices permiten que dichas funciones de pertenencia puedan ser distintas para cada una de las variables de estado, índice k , o de las entradas, índice j , de cada una de las reglas, índice l , que definen cada ecuación de estado, índice i . Por lo tanto, la distribución de las funciones de pertenencia de esta metodología [168,170] permite la máxima flexibilidad, no limitando en modo alguno la utilización de un tipo u otro de función de pertenencia, ni obligando a que éstas deban ser iguales o deban cumplir ningún tipo de requerimiento en cuanto a su distribución en sus respectivos universos de discurso.

Puesto que el consecuente de las reglas está dado por la ecuación (3.13), la expresión (3.14) puede escribirse como:

$$\begin{aligned} \dot{x}_i = & \frac{\sum_{l=1}^{M_i} w_i^l(\mathbf{x}, \mathbf{u}) a_{0i}^l}{\sum_{l=1}^{M_i} w_i^l(\mathbf{x}, \mathbf{u})} + \frac{\sum_{l=1}^{M_i} w_i^l(\mathbf{x}, \mathbf{u}) a_{1i}^l}{\sum_{l=1}^{M_i} w_i^l(\mathbf{x}, \mathbf{u})} + \dots + \frac{\sum_{l=1}^{M_i} w_i^l(\mathbf{x}, \mathbf{u}) a_{ni}^l}{\sum_{l=1}^{M_i} w_i^l(\mathbf{x}, \mathbf{u})} + \\ & + \frac{\sum_{l=1}^{M_i} w_i^l(\mathbf{x}, \mathbf{u}) b_{1i}^l}{\sum_{l=1}^{M_i} w_i^l(\mathbf{x}, \mathbf{u})} + \dots + \frac{\sum_{l=1}^{M_i} w_i^l(\mathbf{x}, \mathbf{u}) b_{mi}^l}{\sum_{l=1}^{M_i} w_i^l(\mathbf{x}, \mathbf{u})}, \end{aligned} \quad (3.16)$$

o de forma más simplificada:

$$\begin{aligned} \dot{x}_i = & a_{0i}(\mathbf{x}, \mathbf{u}) + a_{1i}(\mathbf{x}, \mathbf{u})x_1 + \dots + a_{ni}(\mathbf{x}, \mathbf{u})x_n + \\ & + b_{1i}(\mathbf{x}, \mathbf{u})u_1 + \dots + b_{mi}(\mathbf{x}, \mathbf{u})u_m, \end{aligned} \quad (3.17)$$

donde los términos $a_{ki}(\mathbf{x}, \mathbf{u})$ y $b_{ji}(\mathbf{x}, \mathbf{u})$ son coeficientes variables [230] definidos por:

$$a_{ki}(\mathbf{x}, \mathbf{u}) = \frac{\sum_{l=1}^{M_i} w_i^l(\mathbf{x}, \mathbf{u}) a_{ki}^l}{\sum_{l=1}^{M_i} w_i^l(\mathbf{x}, \mathbf{u})}, \quad b_{ji}(\mathbf{x}, \mathbf{u}) = \frac{\sum_{l=1}^{M_i} w_i^l(\mathbf{x}, \mathbf{u}) b_{ji}^l}{\sum_{l=1}^{M_i} w_i^l(\mathbf{x}, \mathbf{u})}, \quad \begin{array}{ll} k = 0, 1, \dots, n \\ i = 1, \dots, n \\ j = 1, \dots, m \end{array} \quad (3.18)$$

Puesto que no hay riesgos de confundir los coeficientes variables anteriores con los parámetros de los consecuentes de las reglas, ya que los coeficientes variables no dependen del índice l , en las expresiones posteriores se omitirá, con objeto de simplificar la notación, la dependencia de estos coeficientes de los vectores de estado ($\mathbf{x}$) y control ($\mathbf{u}$).

La ecuación (3.17) puede agruparse pues en la forma:

$$\dot{x}_i = a_{0i} + \sum_{k=1}^n a_{ki}x_k + \sum_{j=1}^m b_{ji}u_j. \quad (3.19)$$

```

1: Calcular el vector de estado extendido  $\tilde{\mathbf{x}}$  según (3.20)
2: Para cada ecuación de estado ( $i = 1..n$ ) hacer
3:   Inicializar la salida  $i$ -ésima ( $\dot{x}_i = 0$ )
4:   Suma  $w_i = 0$ 
5:   Para cada regla ( $l = 1..M_i$ ) hacer
6:     Calcular  $w_i^l$  según (3.15) y hacer Suma  $w_i = \text{Suma } w_i + w_i^l$ 
7:   Fin Para
8:   Para cada variable de estado extendida ( $k = 0, 1, \dots, n$ ) hacer
9:     Calcular el numerador de  $a_{ki}$  según (3.18)
10:     $\dot{x}_i = \dot{x}_i + a_{ki}\tilde{x}_k$ 
11:   Fin Para
12:   Para cada variable de control ( $j = 1..m$ ) hacer
13:     Calcular el numerador de  $b_{ji}$  según (3.18)
14:     $\dot{x}_i = \dot{x}_i + b_{ji}u_j$ 
15:   Fin Para
16:    $\dot{x}_i = \dot{x}_i / \text{Suma } w_i$  {Aquí se añade el denominador de  $a_{ki}$  y  $b_{ji}$ }
17: Fin Para
18: Retornar  $\dot{\mathbf{x}}$ 

```

Figura 3.2: Algoritmo para el cálculo del modelo borroso de la planta.

Si el vector de estado se extiende en una coordenada [168] mediante $\tilde{x}_0 = 1$, el vector de estado extendido, $\tilde{\mathbf{x}}$ adopta la forma:

$$\tilde{\mathbf{x}} = (\tilde{x}_0, \tilde{x}_1, \dots, \tilde{x}_n)^T = (1, x_1, \dots, x_n)^T. \quad (3.20)$$

Finalmente, aplicando (3.20) sobre la ecuación (3.19), se obtiene la expresión final, más compacta, que representa la ecuación de estado de una planta completamente general de tiempo continuo, modelada mediante lógica borrosa:

$$\dot{x}_i = \sum_{k=0}^n a_{ki}\tilde{x}_k + \sum_{j=1}^m b_{ji}u_j. \quad (3.21)$$

La ecuación (3.21) representa el equivalente matemático de un modelo borroso de estado en tiempo continuo completamente general, sin limitaciones en el tamaño del vector de estado, ni en el del vector de control, ni en el número de reglas. Cada una de las ecuaciones de estado se caracteriza por un número independiente de reglas, y cada una de sus funciones de pertenencia pueden definirse de forma completamente independiente, incluso permitiéndose la mezcla de distintos tipos de funciones de pertenencia en una misma regla. En la figura 3.2 se muestra un algoritmo que, haciendo uso de la ecuación (3.21), calcula la salida del modelo borroso dado por el conjunto de reglas (3.12), en un estado $\mathbf{x}$ y bajo una señal de control $\mathbf{u}$.

La ecuación (3.21) también puede escribirse matricialmente:

$$\dot{\mathbf{x}}(t) = \mathbf{A}_0(\mathbf{x}) + \mathbf{A}(\mathbf{x})\mathbf{x}(t) + \mathbf{B}(\mathbf{x})\mathbf{u}(t), \quad (3.22)$$

siendo $\mathbf{A}_0(\mathbf{x})$ la matriz de coeficientes variables relacionada con los términos afines, y $\mathbf{A}(\mathbf{x})$ y $\mathbf{B}(\mathbf{x})$ matrices de coeficientes variables relacionadas con el resto de parámetros de los consecuentes:

$$\mathbf{A}_0(\mathbf{x}) = \begin{pmatrix} a_{01}(\mathbf{x}) \\ a_{02}(\mathbf{x}) \\ \vdots \\ a_{0n}(\mathbf{x}) \end{pmatrix} \quad (3.23a) \quad \mathbf{A}(\mathbf{x}) = \begin{pmatrix} a_{01}(\mathbf{x}) & a_{11}(\mathbf{x}) & \dots & a_{n1}(\mathbf{x}) \\ a_{02}(\mathbf{x}) & a_{12}(\mathbf{x}) & \dots & a_{n2}(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ a_{0n}(\mathbf{x}) & a_{1n}(\mathbf{x}) & \dots & a_{nn}(\mathbf{x}) \end{pmatrix} \quad (3.23b)$$

$$\mathbf{B}(\mathbf{x}) = \begin{pmatrix} b_{11}(\mathbf{x}) & b_{21}(\mathbf{x}) & \dots & b_{m1}(\mathbf{x}) \\ b_{12}(\mathbf{x}) & b_{22}(\mathbf{x}) & \dots & b_{m2}(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ b_{1n}(\mathbf{x}) & b_{2n}(\mathbf{x}) & \dots & b_{mn}(\mathbf{x}) \end{pmatrix}. \quad (3.24)$$

Si se utiliza la extensión del vector de estado dada en (3.20), y considerando su derivada, $\dot{\tilde{\mathbf{x}}}(t) = (0, \dot{x}_1, \dots, \dot{x}_n)^T$, la ecuación (3.21) puede expresarse de forma matricial compacta:

$$\dot{\tilde{\mathbf{x}}}(t) = \tilde{\mathbf{A}}(\mathbf{x})\tilde{\mathbf{x}}(t) + \tilde{\mathbf{B}}(\mathbf{x})\mathbf{u}(t), \quad (3.25)$$

siendo $\tilde{\mathbf{A}}(\mathbf{x})$ y $\tilde{\mathbf{B}}(\mathbf{x})$ matrices de coeficientes variables:

$$\tilde{\mathbf{A}}(\mathbf{x}) = \left(\begin{array}{c|c} \mathbf{0} & \mathbf{0} \\ \hline \mathbf{A}_0(\mathbf{x}) & \mathbf{A}(\mathbf{x}) \end{array} \right) = \begin{pmatrix} 0 & 0 & \dots & 0 \\ a_{01}(\mathbf{x}) & a_{11}(\mathbf{x}) & \dots & a_{n1}(\mathbf{x}) \\ a_{02}(\mathbf{x}) & a_{12}(\mathbf{x}) & \dots & a_{n2}(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ a_{0n}(\mathbf{x}) & a_{1n}(\mathbf{x}) & \dots & a_{nn}(\mathbf{x}) \end{pmatrix} \quad (3.26)$$

$$\tilde{\mathbf{B}}(\mathbf{x}) = \left(\begin{array}{c} \mathbf{0} \\ \hline \mathbf{B}(\mathbf{x}) \end{array} \right) = \begin{pmatrix} 0 & 0 & \dots & 0 \\ b_{11}(\mathbf{x}) & b_{21}(\mathbf{x}) & \dots & b_{m1}(\mathbf{x}) \\ b_{12}(\mathbf{x}) & b_{22}(\mathbf{x}) & \dots & b_{m2}(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ b_{1n}(\mathbf{x}) & b_{2n}(\mathbf{x}) & \dots & b_{mn}(\mathbf{x}) \end{pmatrix}. \quad (3.27)$$

Para un sistema muestreado, el conjunto de reglas que representa la planta vendrá dado por la ecuación (3.28).

$$\begin{aligned}
Rp^{(l,i)} : \text{ Si } & x_1(d) \text{ es } A_{1i}^l \text{ y } x_2(d) \text{ es } A_{2i}^l \text{ y } \dots \text{ y } x_n(d) \text{ es } A_{ni}^l \\
& \text{ y } u_1(d) \text{ es } B_{1i}^l \text{ y } u_2(d) \text{ es } B_{2i}^l \text{ y } \dots \text{ y } u_m(d) \text{ es } B_{mi}^l \\
\text{Entonces } & x_i^l(d+1) = \overline{\mathcal{F}}_i^l(\mathbf{x}(d), \mathbf{u}(d), \boldsymbol{\theta}_i^l),
\end{aligned} \tag{3.28}$$

donde d representa el instante de muestreo normalizado¹, es decir $t = dT_s$ siendo T_s el periodo de muestreo del sistema.

Siguiendo el procedimiento empleado anteriormente para una planta modelada en tiempo continuo, la deducción del equivalente matemático de una planta modelada en tiempo discreto mediante el conjunto de reglas (3.28) es inmediata, siendo su expresión final la mostrada en la ecuación (3.29). Nótese que las expresiones auxiliares necesarias para evaluar la ecuación (3.29) son idénticas para el caso continuo y discreto, salvo evidentemente la dependencia temporal de las variables de estado y control.

$$x_i(d+1) = \sum_{k=0}^n a_{ki} \tilde{x}_k(d) + \sum_{j=1}^m b_{ji} u_j(d) \tag{3.29}$$

Al igual que en el caso continuo, esta expresión puede representarse de forma matricial, ver ecuación (3.30), o de la forma matricial compacta dada por la ecuación (3.31) si se emplea la extensión del vector de estado.

$$\mathbf{x}(d+1) = \mathbf{A}_0(\mathbf{x}) + \mathbf{A}(\mathbf{x})\mathbf{x}(d) + \mathbf{B}(\mathbf{x})\mathbf{u}(d) \tag{3.30}$$

$$\tilde{\mathbf{x}}(d+1) = \tilde{\mathbf{A}}(\mathbf{x})\tilde{\mathbf{x}}(d) + \tilde{\mathbf{B}}(\mathbf{x})\mathbf{u}(d) \tag{3.31}$$

Modelo borroso del controlador

Un controlador multivariable de tiempo continuo completamente general, y que por lo tanto puede ser no lineal, sin limitaciones ni en el tamaño del vector de control ni en el vector de estado, puede definirse mediante el sistema de ecuaciones (3.32), o de forma compacta por (3.33).

$$\begin{aligned}
u_1(t) &= g_1(x_1(t), x_2(t), \dots, x_n(t)) \\
u_2(t) &= g_2(x_1(t), x_2(t), \dots, x_n(t)) \\
&\vdots \\
u_m(t) &= g_m(x_1(t), x_2(t), \dots, x_n(t))
\end{aligned} \tag{3.32}$$

$$\mathbf{u}(t) = \mathbf{g}(\mathbf{x}(t)), \tag{3.33}$$

¹Se ha optado por denominar d al instante de tiempo normalizado, en vez de k como suele ser habitual, para evitar posibles confusiones de éste con el índice k que se utiliza para recorrer el vector de estado.

siendo n el orden del sistema y m el número de entradas de control. Un modelo borroso equivalente del controlador dado por las ecuaciones anteriores, puede ser representado mediante el siguiente conjunto de reglas [231, 232] (base de reglas denominada R_c para enfatizar el hecho de que se trata del controlador):

$$\begin{aligned} Rc^{(r,j)} : \text{ Si } & x_1 \text{ es } C_{1j}^r \text{ y } x_2 \text{ es } C_{2j}^r \text{ y } \dots \text{ y } x_n \text{ es } C_{nj}^r \\ \text{Entonces } & u_j^r = \mathcal{G}_j^r(\mathbf{x}, \boldsymbol{\vartheta}_j^r), \end{aligned} \quad (3.34)$$

donde $r = 1..N_j$ es el índice de las reglas del controlador y N_j el número de reglas que modelan la j -ésima señal de control. C_{kj}^r , con $k = 1..n$, representa los n conjuntos borrosos del antecedente de la regla r de la señal de control u_j^r en el universo de discurso de las variables de estado.

La función $\mathcal{G}_j^r(\mathbf{x}, \boldsymbol{\vartheta}_j^r)$ es el consecuente de la regla r para la j -ésima señal de control, siendo $\boldsymbol{\vartheta}_j^r$ el vector de parámetros adaptables del consecuente. Si se utiliza un consecuente TSK con término afín, éste tendrá la forma dada en (3.35), y $\boldsymbol{\vartheta}_j^r$ será $(c_{0j}^r, c_{1j}^r, \dots, c_{nj}^r)$.

$$\mathcal{G}_j^r(\mathbf{x}, \boldsymbol{\vartheta}_j^r) = c_{0j}^r + c_{1j}^r x_1 + \dots + c_{nj}^r x_n \quad (3.35)$$

Utilizando el promedio ponderado como método de agregación, la salida del modelo borroso generada por el conjunto de todas las reglas $Rc^{(r,j)}$ puede calcularse empleando las expresiones (3.36) y (3.37) [228, 229], donde $\omega_j^r(\mathbf{x})$ representa el grado de activación de las reglas del controlador y $\mu_{kj}^r(x_k, \boldsymbol{\Phi}_{kj}^r)$ son las funciones de pertenencia de los antecedentes de las reglas, cuyos parámetros ajustables se definen por $\boldsymbol{\Phi}_{kj}^r$.

$$u_j = \frac{\sum_{r=1}^{N_j} \omega_j^r(\mathbf{x}) \mathcal{G}_j^r(\mathbf{x}, \boldsymbol{\vartheta}_j^r)}{\sum_{r=1}^{N_j} \omega_j^r(\mathbf{x})} \quad (3.36)$$

$$\omega_j^r = \prod_{k=1}^n \mu_{kj}^r(x_k, \boldsymbol{\Phi}_{kj}^r) \quad (3.37)$$

Procediendo con el cálculo de forma similar a como se realizó en (3.16) y (3.17), se pueden obtener las expresiones (3.38) y (3.39), donde en este caso los coeficientes variables² $c_{kj}(\mathbf{x})$ se calculan mediante la ecuación (3.40).

$$u_j = \frac{\sum_{r=1}^{N_j} \omega_j^r(\mathbf{x}) c_{0j}^r}{\sum_{r=1}^{N_j} \omega_j^r(\mathbf{x})} + \frac{\sum_{r=1}^{N_j} \omega_j^r(\mathbf{x}) c_{1j}^r}{\sum_{r=1}^{N_j} \omega_j^r(\mathbf{x})} + \dots + \frac{\sum_{r=1}^{N_j} \omega_j^r(\mathbf{x}) c_{nj}^r}{\sum_{r=1}^{N_j} \omega_j^r(\mathbf{x})} \quad (3.38)$$

$$u_j = c_{0j}(\mathbf{x}) + c_{1j}(\mathbf{x}) x_1 + \dots + c_{nj}(\mathbf{x}) x_n \quad (3.39)$$

²En las expresiones posteriores se omitirá la dependencia de los coeficientes variables del controlador del vector de estado con objeto de simplificar la notación.

```

1: Calcular el vector de estado extendido  $\tilde{\mathbf{x}}$  según (3.20)
2: Para cada señal de control ( $j = 1..m$ ) hacer
3:   Inicializar la salida  $j$ -ésima ( $u_j = 0$ )
4:   Suma  $\omega_j = 0$ 
5:   Para cada regla ( $r = 1..N_j$ ) hacer
6:     Calcular  $\omega_j^r$  según (3.37) y hacer Suma  $\omega_j = \text{Suma } \omega_j + \omega_j^r$ 
7:   Fin Para
8:   Para cada variable de estado extendida ( $k = 0, 1, \dots, n$ ) hacer
9:     Calcular el numerador de  $c_{kj}$  según (3.40)
10:     $u_j = u_j + c_{kj}\tilde{x}_k$ 
11:   Fin Para
12:    $u_j = u_j / \text{Suma } \omega_j$  {Aquí se añade el denominador de  $c_{kj}$ }
13: Fin Para
14: Retornar  $\mathbf{u}$ 

```

Figura 3.3: Algoritmo para el cálculo del controlador borroso.

$$c_{kj}(\mathbf{x}) = \frac{\sum_{r=1}^{N_j} \omega_j^r(\mathbf{x}) c_{kj}^r}{\sum_{r=1}^{N_j} \omega_j^r(\mathbf{x})} \quad (3.40)$$

La ecuación (3.39) puede agruparse, dando lugar a (3.41), que representa el equivalente matemático del controlador borroso.

$$u_j = c_{0j} + \sum_{k=1}^n c_{kj} x_k \quad (3.41)$$

En este caso, el vector de estados también puede extenderse en una coordenada [168, 233], $\tilde{\mathbf{x}} = (\tilde{x}_0, \tilde{x}_1, \dots, \tilde{x}_n)^T = (1, x_1, \dots, x_n)^T$, con lo que el controlador borroso puede representarse de forma más compacta por la ecuación (3.42).

$$u_j = \sum_{k=0}^n c_{kj} \tilde{x}_k \quad (3.42)$$

La expresión (3.42) representa el modelo matemático equivalente de un controlador borroso multivariable completamente general, sin limitaciones en el tamaño del vector de estado, ni en el del vector de control, ni en el número de reglas. Cada salida del controlador se caracteriza por un número independiente de reglas, y cada una de sus funciones de pertenencia pueden definirse de forma completamente independiente, incluso pudiendo ser de distinto tipo en una misma regla. En la figura 3.3 se muestra un algoritmo que calcula la salida $\mathbf{u}$ del controlador borroso dado por el conjunto de reglas (3.34) en un estado $\mathbf{x}$.

La ecuación (3.42) puede ser escrita de forma matricial como:

$$\mathbf{u}(t) = \mathbf{C}_0(\mathbf{x}) + \mathbf{C}(\mathbf{x})\mathbf{x}(t), \quad (3.43)$$

donde $\mathbf{C}_0(\mathbf{x})$ es una matriz de coeficientes variables relacionada con los términos afines, y $\mathbf{C}(\mathbf{x})$ una matriz de coeficientes variables relacionadas con el resto de parámetros de los consecuentes:

$$\mathbf{C}_0(\mathbf{x}) = \begin{pmatrix} c_{01}(\mathbf{x}) \\ c_{02}(\mathbf{x}) \\ \vdots \\ c_{0m}(\mathbf{x}) \end{pmatrix} \quad (3.44a) \quad \mathbf{C}(\mathbf{x}) = \begin{pmatrix} c_{11}(\mathbf{x}) & \dots & c_{n1}(\mathbf{x}) \\ c_{12}(\mathbf{x}) & \dots & c_{n2}(\mathbf{x}) \\ \vdots & \ddots & \vdots \\ c_{1m}(\mathbf{x}) & \dots & c_{nm}(\mathbf{x}) \end{pmatrix}. \quad (3.44b)$$

Si se utiliza la extensión del vector de estado dada en (3.20), la ecuación (3.42) puede expresarse de forma matricial compacta:

$$\mathbf{u}(t) = \tilde{\mathbf{C}}(\mathbf{x})\tilde{\mathbf{x}}(t), \quad (3.45)$$

siendo $\tilde{\mathbf{C}}(\mathbf{x})$ una matriz de coeficientes variables dada por:

$$\tilde{\mathbf{C}}(\mathbf{x}) = \left(\mathbf{C}_0(\mathbf{x}) \mid \mathbf{C}(\mathbf{x}) \right) = \left(\begin{array}{c|ccc} c_{01}(\mathbf{x}) & c_{11}(\mathbf{x}) & \dots & c_{n1}(\mathbf{x}) \\ c_{02}(\mathbf{x}) & c_{12}(\mathbf{x}) & \dots & c_{n2}(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ c_{0m}(\mathbf{x}) & c_{1m}(\mathbf{x}) & \dots & c_{nm}(\mathbf{x}) \end{array} \right). \quad (3.46)$$

Para el caso de un sistema muestreado, un controlador multivariable de tiempo discreto completamente general, y que por lo tanto puede ser no lineal, sin limitaciones ni en el tamaño del vector de control ni en el del vector de estado, puede definirse mediante el sistema de ecuaciones (3.47), o de forma compacta por (3.48).

$$\begin{aligned} u_1(d) &= \bar{g}_1(x_1(d), x_2(d), \dots, x_n(d)) \\ u_2(d) &= \bar{g}_2(x_1(d), x_2(d), \dots, x_n(d)) \\ &\vdots \end{aligned} \quad (3.47)$$

$$u_m(d) = \bar{g}_m(x_1(d), x_2(d), \dots, x_n(d))$$

$$\mathbf{u}(d) = \bar{\mathbf{g}}(\mathbf{x}(d)) \quad (3.48)$$

Un modelo borroso equivalente del controlador dado por las ecuaciones anteriores, puede ser representado mediante el conjunto de reglas:

$$\begin{aligned} Rc^{(r,j)} : \text{ Si } & x_1(d) \text{ es } C_{1j}^r \text{ y } x_2(d) \text{ es } C_{2j}^r \text{ y } \dots \text{ y } x_n(d) \text{ es } C_{nj}^r \\ \text{Entonces } & u_j^r(d) = \bar{\mathcal{G}}_j^r(\mathbf{x}(d), \boldsymbol{\vartheta}_j^r) \end{aligned} \quad (3.49)$$

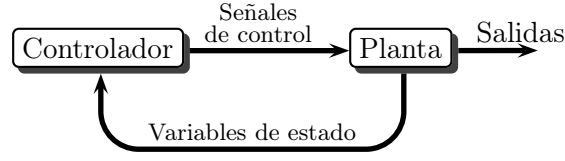


Figura 3.4: Sistema de control en lazo cerrado.

Operando a partir del conjunto de reglas anterior de forma idéntica a como se ha realizado para el caso continuo, se obtiene la expresión (3.50), que representa el modelo matemático equivalente del controlador borroso de tiempo discreto definido en (3.49).

$$u_j(d) = \sum_{k=0}^n c_{kj} \tilde{x}_k(d) \quad (3.50)$$

Al igual que en el caso de la planta, la ecuación anterior puede expresarse de forma matricial, según la ecuación (3.51), o de forma matricial compacta, tal como refleja la ecuación (3.52).

$$\mathbf{u}(d) = \mathbf{C}_0(\mathbf{x}) + \mathbf{C}(\mathbf{x})\mathbf{x}(d) \quad (3.51)$$

$$\mathbf{u}(d) = \tilde{\mathbf{C}}(\mathbf{x})\tilde{\mathbf{x}}(d) \quad (3.52)$$

Modelo borroso de lazo cerrado

Dado un sistema de control en lazo cerrado como el representado en la figura 3.4, si tanto la planta como el controlador se modelan mediante sistemas borrosos de tipo TSK, se puede obtener un modelo matemático equivalente de dicho sistema.

Para el caso de tiempo continuo, (3.21) es el modelo matemático equivalente de la planta, y (3.42) el del controlador. Sustituyendo u_j de la segunda ecuación en la primera, se obtiene:

$$\dot{x}_i = \sum_{k=0}^n a_{ki} \tilde{x}_k + \sum_{j=1}^m \left(b_{ji} \sum_{k=0}^n c_{kj} \tilde{x}_k \right). \quad (3.53)$$

Operando con los sumatorios:

$$\dot{x}_i = \sum_{k=0}^n a_{ki} \tilde{x}_k + \sum_{k=0}^n \sum_{j=1}^m (b_{ji} c_{kj} \tilde{x}_k), \quad (3.54)$$

que finalmente puede simplificarse en:

$$\dot{x}_i = \sum_{k=0}^n \left(a_{ki} + \sum_{j=1}^m b_{ji} c_{kj} \right) \tilde{x}_k. \quad (3.55)$$

La ecuación (3.55) representa el modelo matemático equivalente de un sistema de control en lazo cerrado multivariable y no lineal, basado en la teoría de la lógica borrosa. Es importante recalcar que el modelo equivalente desarrollado no presenta restricciones ni en el tamaño del vector de estados, ni en el del vector de control. Además, el diseño de la estructura del controlador borroso se considera completamente independiente del proceso de modelado de la planta, de forma que la partición del universo de discurso de los estados de la planta y del controlador no tienen porqué coincidir. Igualmente, la dinámica de la planta y la del controlador no tienen porqué tener la misma complejidad, por lo que el número de reglas de cada uno de estos sistemas borrosos se considera independiente.

En la figura 3.5 se muestra un algoritmo que calcula la salida de un sistema de control borroso en un estado $\mathbf{x}$, mediante la utilización de la ecuación (3.55).

La ecuación (3.55) también puede escribirse matricialmente:

$$\dot{\mathbf{x}}(t) = \mathbf{A}_0(\mathbf{x}) + \mathbf{B}(\mathbf{x})\mathbf{C}_0(\mathbf{x}) + (\mathbf{A}(\mathbf{x}) + \mathbf{B}(\mathbf{x})\mathbf{C}(\mathbf{x})) \mathbf{x}(t), \quad (3.56)$$

siendo $\mathbf{A}_0(\mathbf{x})$ y $\mathbf{C}_0(\mathbf{x})$ matrices de coeficientes variables relacionadas con los términos afines, y $\mathbf{A}(\mathbf{x})$, $\mathbf{B}(\mathbf{x})$ y $\mathbf{C}(\mathbf{x})$, matrices de coeficientes variables relacionadas con el resto de parámetros de los consecuentes. Estas matrices han sido definidas previamente en las ecuaciones (3.23), (3.24) y (3.44).

Si se utiliza la extensión del vector de estado dada en (3.20), y considerando su derivada, $\dot{\tilde{\mathbf{x}}}(t) = (0, \dot{x}_1, \dots, \dot{x}_n)^T$, la ecuación (3.55) puede expresarse de forma matricial compacta:

$$\dot{\tilde{\mathbf{x}}}(t) = \left(\tilde{\mathbf{A}}(\mathbf{x}) + \tilde{\mathbf{B}}(\mathbf{x})\tilde{\mathbf{C}}(\mathbf{x}) \right) \tilde{\mathbf{x}}(t), \quad (3.57)$$

siendo $\tilde{\mathbf{A}}(\mathbf{x})$, $\tilde{\mathbf{B}}(\mathbf{x})$ y $\tilde{\mathbf{C}}(\mathbf{x})$ matrices de coeficientes variables definidas por las ecuaciones (3.26), (3.27) y (3.46), respectivamente.

Para el caso de tiempo discreto, la ecuación (3.29) representa el modelo matemático equivalente de la planta, y (3.50) el del controlador. Operando como en el caso continuo se obtiene la expresión (3.58), que representa el modelo matemático equivalente de lazo cerrado de dicho sistema.

$$x_i(d+1) = \sum_{k=0}^n \left(a_{ki} + \sum_{j=1}^m b_{ji} c_{kj} \right) \tilde{x}_k(d) \quad (3.58)$$

Evidentemente, esta expresión también puede expresarse de forma matricial:

$$\mathbf{x}(d+1) = \mathbf{A}_0(\mathbf{x}) + \mathbf{B}(\mathbf{x})\mathbf{C}_0(\mathbf{x}) + (\mathbf{A}(\mathbf{x}) + \mathbf{B}(\mathbf{x})\mathbf{C}(\mathbf{x})) \mathbf{x}(d), \quad (3.59)$$

y de forma matricial compacta utilizando la extensión del vector de estado:

$$\tilde{\mathbf{x}}(d+1) = \left(\tilde{\mathbf{A}}(\mathbf{x}) + \tilde{\mathbf{B}}(\mathbf{x})\tilde{\mathbf{C}}(\mathbf{x}) \right) \tilde{\mathbf{x}}(d). \quad (3.60)$$

```

1: Si las dimensiones de la planta y el controlador no son coherentes entonces
2:   Retornar error
3: Fin Si
4: Calcular el vector de estado extendido  $\tilde{\mathbf{x}}$  según (3.20)
5: Para cada señal de control ( $j = 1..m$ ) hacer
6:   Suma  $\omega_j = 0$ 
7:   Para cada regla ( $r = 1..N_j$ ) hacer
8:     Calcular  $\omega_j^r$  según (3.37) y hacer Suma  $\omega_j = \text{Suma } \omega_j + \omega_j^r$ 
9:   Fin Para
10:  Para cada variable de estado extendida ( $k = 0, 1, \dots, n$ ) hacer
11:    Calcular el numerador de  $c_{kj}$  según (3.40)
12:  Fin Para
13:   $c_{kj} = c_{kj} / \text{Suma } \omega_j$ 
14: Fin Para
15: Para cada ecuación de estado ( $i = 1..n$ ) hacer
16:   Suma  $w_i = 0$ 
17:   Para cada regla ( $l = 1..M_i$ ) hacer
18:     Calcular  $w_i^l$  según (3.15) y hacer Suma  $w_i = \text{Suma } w_i + w_i^l$ 
19:   Fin Para
20: Fin Para
21: Para cada ecuación de estado ( $i = 1..n$ ) hacer
22:   Inicializar la salida  $i$ -ésima ( $\dot{x}_i = 0$ )
23:   Para cada variable de control ( $j = 1..m$ ) hacer
24:     Calcular el numerador de  $b_{ji}$  según (3.18)
25:      $\dot{x}_i = \dot{x}_i + b_{ji}u_j$ 
26:   Fin Para
27:   Para cada variable de estado extendida ( $k = 0, 1, \dots, n$ ) hacer
28:     Calcular el numerador de  $a_{ki}$  según (3.18)
29:      $\dot{x}_i = \dot{x}_i + a_{ki}\tilde{x}_k$ 
30:      $\dot{x}_{aux} = 0$ 
31:     Para cada variable de control ( $j = 1..m$ ) hacer
32:        $\dot{x}_{aux} = \dot{x}_{aux} + b_{ji}c_{kj}$  {Sólo se utilizan el numerador de  $b_{ji}$ }
33:     Fin Para
34:      $\dot{x}_i = \dot{x}_i + (a_{ki} + \dot{x}_{aux})\tilde{x}_k$  {Sólo se utiliza el numerador de  $a_{ki}$ }
35:   Fin Para
36:    $\dot{x}_i = \dot{x}_i / \text{Suma } w_i$  {Aquí se añade el denominador de  $a_{ki}$  y  $b_{ji}$ }
37: Fin Para
38: Retornar  $\dot{\mathbf{x}}$ 

```

Figura 3.5: Algoritmo para el cálculo del sistema de control borroso en lazo cerrado.

3.2.2. Sistema borroso de tipo Mamdani

Los modelos borrosos de Mamdani destacan frente a los TSK por su mayor interpretabilidad, de ahí que estos modelos se empleen frecuentemente para extraer reglas descriptivas del funcionamiento de sistemas [234, 235, 236], o como predictores lingüísticamente comprensibles [237, 238]. Desde el punto de vista del control, los modelos de Mamdani permiten la creación de controladores basados en el conocimiento de un experto, también denominados controladores heurísticos³. Estos controladores propiciaron el primer gran auge de la lógica borrosa gracias a su capacidad de controlar sistemas complejos en base a predicados lingüísticos obtenidos de un experto, y sin requerir el modelado matemático de la planta [198, 239].

Un sistema de tiempo continuo de orden n y m entradas, puede representarse mediante un modelo de Mamdani de la forma:

$$\begin{aligned} Rp^{(l,i)} : \text{ Si } & x_1 \text{ es } A_{1i}^l \text{ y } x_2 \text{ es } A_{2i}^l \text{ y } \dots \text{ y } x_n \text{ es } A_{ni}^l \\ & \text{ y } u_1 \text{ es } B_{1i}^l \text{ y } u_2 \text{ es } B_{2i}^l \text{ y } \dots \text{ y } u_m \text{ es } B_{mi}^l \\ \text{Entonces } & \dot{x}_i^l = F_i^l, \end{aligned} \quad (3.61)$$

donde $l = 1..M_i$ es el índice de la regla y M_i el número de reglas que modelan la i -ésima ecuación diferencial del proceso, $\dot{x}_i$. Los elementos A_{ki}^l son los conjuntos borrosos de los antecedentes definidos en el universo de discurso de sus respectivas variables de estado, x_k , $k = 1..n$, y los elementos B_{ji}^l son los conjuntos borrosos definidos en el universo de discurso de la j -ésima entrada del sistema, u_j , $j = 1..m$. F_i^l es el conjunto borroso del consecuente de la regla l para la ecuación de estado i -ésima, definido en el universo de discurso de $\dot{x}_i$.

Un controlador borroso genérico de tipo Mamdani de tiempo continuo, vendrá dado por el conjunto de reglas:

$$\begin{aligned} Rc^{(r,j)} : \text{ Si } & x_1 \text{ es } C_{1j}^r \text{ y } x_2 \text{ es } C_{2j}^r \text{ y } \dots \text{ y } x_n \text{ es } C_{nj}^r \\ \text{Entonces } & u_j^r = G_j^r, \end{aligned} \quad (3.62)$$

donde $r = 1..N_j$ es el índice de las reglas del controlador y N_j el número de reglas que modelan la j -ésima señal de control. C_{kj}^r , con $k = 1..n$, representa los n conjuntos borrosos del antecedente de la regla r de la señal de control j en el universo de discurso de las variables de estado, y G_j^r el conjunto borroso del consecuente de dicha regla, definido en el universo de discurso de u_j^r .

Para el caso de sistemas de tiempo discreto, los modelos borrosos de la planta y del controlador pueden representarse por el conjunto de reglas de las ecuaciones (3.63) y (3.64), respectivamente.

³Este tipo de controladores se estudiarán con más detenimiento en el capítulo 5.

$$\begin{aligned}
Rp^{(l,i)} : \text{ Si } & x_1(d) \text{ es } A_{1i}^l \text{ y } x_2(d) \text{ es } A_{2i}^l \text{ y } \dots \text{ y } x_n(d) \text{ es } A_{ni}^l \\
& \text{ y } u_1(d) \text{ es } B_{1i}^l \text{ y } u_2(d) \text{ es } B_{2i}^l \text{ y } \dots \text{ y } u_m(d) \text{ es } B_{mi}^l \\
\text{Entonces } & x_i^l(d+1) = \overline{F}_i^l
\end{aligned} \tag{3.63}$$

$$\begin{aligned}
Rc^{(r,j)} : \text{ Si } & x_1(d) \text{ es } C_{1j}^r \text{ y } x_2(d) \text{ es } C_{2j}^r \text{ y } \dots \text{ y } x_n(d) \text{ es } C_{nj}^r \\
\text{Entonces } & u_j^r(d) = \overline{G}_j^r
\end{aligned} \tag{3.64}$$

Al igual que sucede con los modelos TSK estudiados anteriormente, para los modelos de Mamdani también es posible obtener un modelo matemático equivalente. Sin embargo, en este caso no hay un sistema de inferencia claramente predominante sobre el resto en la literatura, como sí que sucede con el operador de inferencia producto y la agregación por promedio ponderado en los sistemas TSK. Por lo tanto, es necesario obtener el modelo matemático equivalente en función de la combinación de los operadores que se desee emplear. El estudio matemático de los modelos de Mamdani no es un objetivo de este texto, por lo que se obviará dicho cálculo.

3.3. Identificación de modelos borrosos

Muchos procesos reales contienen complejas relaciones no lineales difíciles de modelar por técnicas clásicas. Puesto que los sistemas borrosos pueden comportarse como aproximadores universales [240], éstos se han convertido en una herramienta muy potente para el modelado de sistemas complejos. En concreto, los modelos de Takagi y Sugeno [208] son capaces de describir sistemas altamente no lineales utilizando un número de reglas reducido.

El problema de la obtención del modelo borroso de una planta consiste, fundamentalmente, en el ajuste de los parámetros característicos del mismo, de modo que represente, de la forma más fiel posible, la dinámica de ésta. Como se muestra en la figura 3.6, el proceso de modelado de un sistema consta de 3 fases [14]: identificación de la estructura del sistema, estimación de los parámetros y validación del modelo obtenido. La fase de identificación consiste en localizar las entradas importantes del sistema de entre todas las candidatas, establecer las particiones del universo de discurso de cada una de éstas y determinar las reglas que relacionarán las entradas y salidas del modelo. Durante la fase de estimación se ajustan los parámetros del modelo borroso para minimizar el error de modelado. En función del algoritmo empleado, pueden ajustarse los parámetros de los consecuentes de las reglas, o los parámetros de los consecuentes más los de los antecedentes (funciones de pertenencia). Por último, la fase de validación se utiliza para comprobar el correcto funcionamiento del modelo obtenido. Si el modelo resultante no cumple las expectativas deseadas deberá volverse a la fase inicial para así obtener un nuevo modelo.

En la bibliografía se han propuesto diferentes metodologías para el ajuste de los parámetros de un modelo borroso, aunque las más ampliamente utilizadas han sido las basadas en

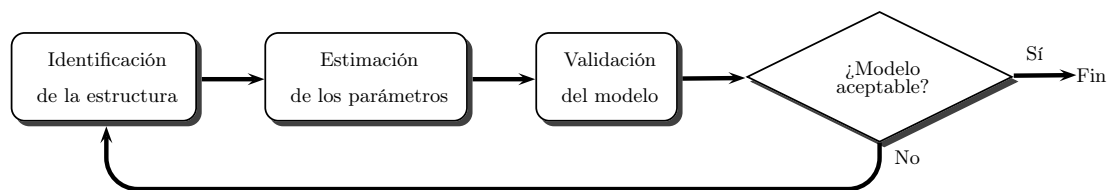


Figura 3.6: *Proceso de identificación.*

gradiente descendente [241, 242] y, más recientemente, en algoritmos evolutivos [243, 244]. Independientemente de la metodología empleada, la velocidad de convergencia y la calidad de la estimación final, dependen fuertemente del valor inicial de los parámetros del modelo, para lo cual se han propuesto en la literatura varias técnicas, como la prueba y error [245], *clustering* [246] o la simple asignación aleatoria [14].

3.3.1. Identificación de la estructura

La selección del conjunto adecuado de variables de entrada es el primer paso antes de identificar la estructura del modelo borroso de un sistema dinámico, y uno de los mayores problemas a la hora de construir dicho modelo, ya que son muchos los factores que pueden influir en la salida de un sistema [247, 248, 249, 250]. Sólo deben escogerse como entradas del modelo aquellas variables cuya relevancia sea significativa con respecto al resto. La inclusión de variables de poca importancia complicará el proceso de modelado sin aportar un beneficio claro al resultado final. La selección de las variables de entrada de un modelo puede resolverse en base al conocimiento que se tenga sobre el sistema a modelar, aunque en la mayoría de los casos dicha información suele ser insuficiente. Existen métodos estadísticos de selección que, junto con el conocimiento que se posea sobre el sistema, pueden utilizarse para facilitar dicha labor [251].

Una vez escogidas las variables de entrada que se emplearán en el modelo del sistema, el siguiente paso es identificar la estructura del mismo. Para ello se deben establecer las divisiones del universo de discurso de las variables de entrada, y escoger las funciones de pertenencia que se utilizarán para la realización de dichas divisiones. Este particionado dará lugar a las reglas que definirán el modelo borroso. Existen tres métodos de particionar el universo de discurso:

1. *Particionado experto.* El particionado experto consiste en dividir el universo de discurso en base al conocimiento que se posea sobre el sistema a modelar. Si el sistema no es muy complejo y el conocimiento sobre éste es suficientemente preciso, es posible dividir el universo de discurso de las variables de entrada sin necesidad de emplear ningún algoritmo específico.
2. *Particionado uniforme.* El particionado uniforme divide el universo de discurso de las variables de entrada en partes iguales mediante funciones de pertenencia idénticas y simétricas. Normalmente las funciones de pertenencia se distribuyen de forma que no exista solape con más de una función a la vez, lo que facilita la interpretabilidad

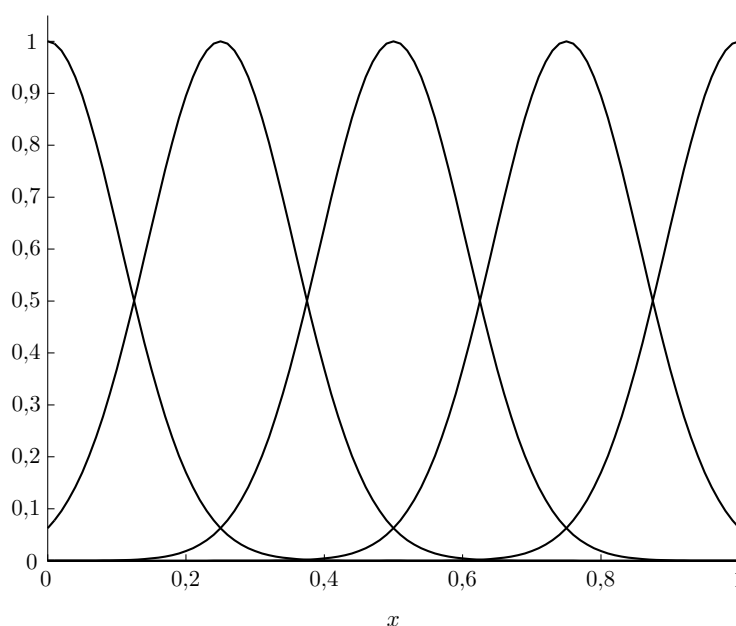


Figura 3.7: Ejemplo de particionamiento uniforme.

del modelo. En la figura 3.7 se muestra un ejemplo de particionado uniforme de un universo de discurso normalizado mediante 5 funciones de pertenencia gaussianas.

3. *Particionado por clustering.* Los algoritmos de *clustering* facilitan el agrupamiento de los datos de entrada en torno a valores de salidas que cumplan cierto patrón de comportamiento. Si el grado de pertenencia de los puntos a los *clusters* se asigna de forma borrosa y el patrón de agrupamiento se asigna en función del tipo de consecuente deseado, los algoritmos de *clustering* permiten la creación de modelos borrosos donde cada uno de los *clusters* obtenidos se corresponde con una regla del modelo [209, 252, 253]. Los algoritmos de *clustering* más empleados por sus buenos resultados en el modelado borroso [254] son el algoritmo *c-means* [255, 256], y el algoritmo de Gustafson–Kessel [257]. En la figura 3.8 se muestra un ejemplo de particionado por *clustering* en un universo de discurso normalizado mediante funciones de pertenencia gaussianas. Como puede observarse, las funciones de pertenencia no están igualmente espaciadas, y existe solape entre todas ellas, lo que dificulta la interpretabilidad del modelo borroso; sin embargo, al ajustarse los antecedentes de forma más precisa a los datos de entrada, el modelo resultante será normalmente más preciso que con los métodos de particionamiento anteriores. En la literatura existen algoritmos de *clustering* que permiten incorporar el conocimiento experto como método para mejorar la interpretabilidad y el resultado obtenido por el proceso de agrupamiento [258, 259].
4. *Particionado en línea.* Las técnicas de particionado en línea son especialmente útiles cuando se desea modificar un modelo en tiempo real, o cuando existen restricciones

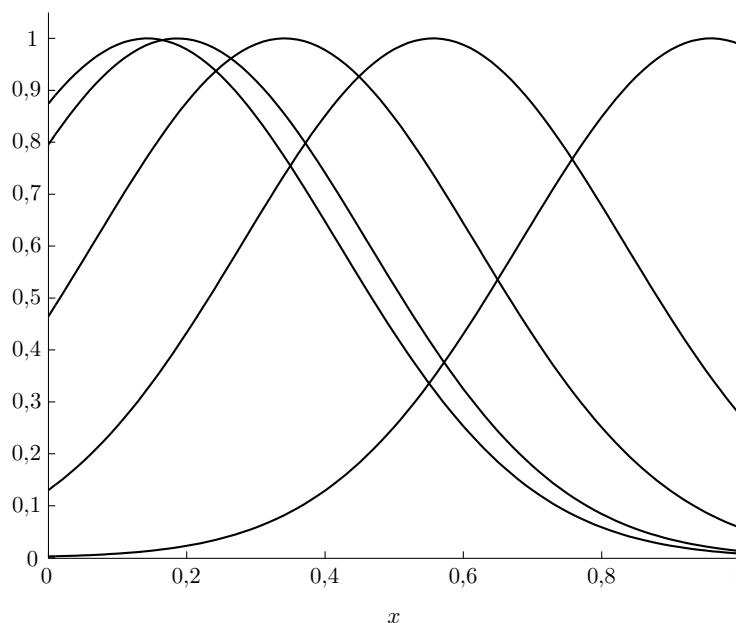


Figura 3.8: Ejemplo de particionamiento por clustering.

temporales o de costo computacional. Los métodos de *clustering* vistos en el punto anterior obtienen buenos resultados, pero requieren de un tiempo de procesado y un costo computacional alto, por lo que no son válidos para aplicaciones de tiempo real. Los algoritmos de particionado en línea suelen emplear algoritmos de pasada simple, es decir, algoritmos donde cada dato añadido es analizado para actualizar el modelo y luego es descartado. De esta forma se evita el procesado recurrente de todos los datos y, por lo tanto, se reduce el número de operaciones necesarias para la actualización del modelo y se agiliza el proceso de agrupamiento, permitiendo su utilización en línea. Como ejemplos de estos algoritmos destacan el algoritmo de agrupamiento evolutivo [260], el método de agrupamiento aglomerativo [261] y el método de agrupamiento evolutivo-aglomerativo [262].

Cualquiera de los algoritmos anteriores puede ser utilizado para generar la estructura de un modelo borroso, así como un modelo borroso inicial que después puede ser mejorado en la fase de estimación; sin embargo, es importante notar que en muchas ocasiones se pueden generar un número excesivo de reglas. Aunque teóricamente un mayor número de reglas mejora la capacidad de aproximación del modelo borroso [263], en la práctica, un ajuste fino sobre los datos empleados para el modelado no asegura siempre un buen ajuste sobre los datos futuros, especialmente si las muestras están afectadas por ruido o si el número de parámetros adaptables del modelo es elevado con respecto al número de datos. Un modelo borroso con un excesivo número de reglas posee mayor riesgo de entrar en sobre-entrenamiento (*overfitting*), es decir, el modelo se vuelve muy específico a la información presentada en la muestra de entrenamiento, aprendiendo incluso el ruido presente en los

datos, pero es incapaz de obtener una salida aceptable para nuevos datos. Para extraer las reglas importantes de un modelo borroso y descartar aquellas menos significativas, se pueden emplear algoritmos de simplificación o reducción de modelos [14, 264, 265, 266, 267].

3.3.2. Estimación de parámetros

El objetivo de la fase de estimación o ajuste de parámetros es encontrar los valores óptimos de los parámetros ajustables del modelo borroso, para que éste se aproxime de la forma más exacta posible al sistema real que se pretende modelar. También es posible incorporar otros objetivos a la fase de estimación, como la interpretabilidad [234] del modelo resultante, aunque el error de estimación es sin duda el parámetro de ajuste de mayor importancia.

En un modelo borroso se distinguen claramente dos tipos de parámetros ajustables: los parámetros de los antecedentes y los de los consecuentes. Los parámetros de los antecedentes son los valores que determinan las funciones de pertenencia del modelo. Estos valores se ajustan durante la fase de identificación de la estructura, normalmente mediante *clustering*, aunque pueden refinarse en la fase de estimación. Los parámetros de los consecuentes de las reglas borrosas se ajustan durante la fase de estimación, de forma que se reduzca el error de estimación del modelo. Existen multitud de algoritmos de estimación de parámetros para modelos borrosos [225, 241, 242, 246, 256, 268, 269]. No obstante, por su potencia y posibilidades, el texto se centra en técnicas híbridas. A saber: lógica borrosa + redes neuronales (metodología neuroborrosa [270, 271]) y algoritmos bioinspirados + gradiente descendente (metodología bioinspirada híbrida [169, 233, 272]).

Metodología neuroborrosa

Las redes neuronales artificiales y la lógica borrosa son dos tecnologías complementarias, ya que la conjunción de la capacidad de aprendizaje de las redes neuronales con la habilidad de procesamiento de predicados vagos de la lógica borrosa, proporciona un marco formal cercano a la forma de razonamiento del ser humano.

La hibridación redes neuronales artificiales + lógica borrosa proporciona un sistema inteligente híbrido denominado sistema neuroborroso (*Neuro-Fuzzy System, NFS*). La cualidad principal de los sistemas neuroborrosos es que son aproximadores universales, como las redes neuronales, pero con la habilidad adicional de ofrecer reglas Si-Entonces interpretables que permiten dar significado físico a los modelos.

Como se puede observar en la figura 3.9, un sistema borroso, cuyo modelo matemático equivalente viene dado por la expresión (3.14), puede representarse como una red neuronal de flujo directo de 3 capas hacia adelante [6, 273], esto es, un perceptrón de 3 capas [228]. La primera capa representa la capa de entrada, y en ella se sitúan todas las entradas del sistema, $x_1, \dots, x_n$. Los datos de cada entrada, la k -ésima por ejemplo, son ponderados entre la capa de entrada y la capa intermedia, o capa oculta, a través de las funciones de pertenencia correspondientes a cada regla l , μ_{ki}^l . La capa oculta se conecta con la capa de salida mediante el consecuente correspondiente a cada regla $l = 1..M_i$, y finalmente la capa de salida calcula la salida i -ésima del modelo neuroborroso. Nótese como cada

nodo de capa está conectado a cada nodo de la capa siguiente, tal como sucede en un perceptrón de tres capas. Naturalmente, el comportamiento de los nodos y las sinapsis en un sistema neuroborroso no son iguales que en el perceptrón, sin embargo, pueden emplearse los mismos algoritmos de adaptación.

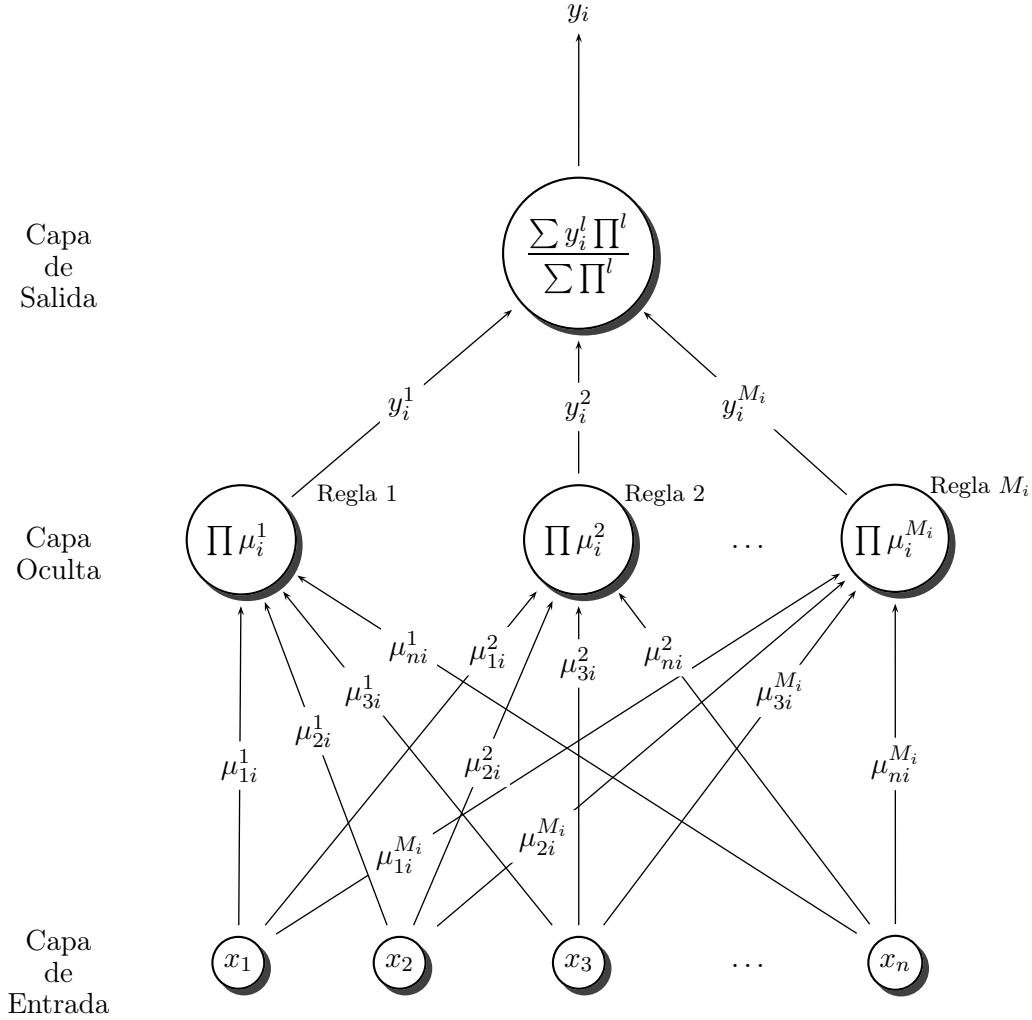


Figura 3.9: Red neuronal equivalente a un modelo borroso.

Para mostrar con más detalle la representación neuronal de un sistema borroso, en la figura 3.10 se muestra la red neuronal equivalente al modelo borroso de una planta visto en la sección 3.2.1, cuyo modelo matemático viene dado por la ecuación (3.21).

La conexión red neuronal-lógica borrosa permite aprovechar la habilidad de las redes neuronales artificiales para construir modelos borrosos. Así, entre otros, es de aplicación el conocido algoritmo de *back propagation* (BP), uno de los más importantes algoritmos de aprendizaje de las redes neuronales, a la identificación de modelos borrosos con la estructura de las figuras 3.9 ó 3.10. El algoritmo BP es un algoritmo de aprendizaje

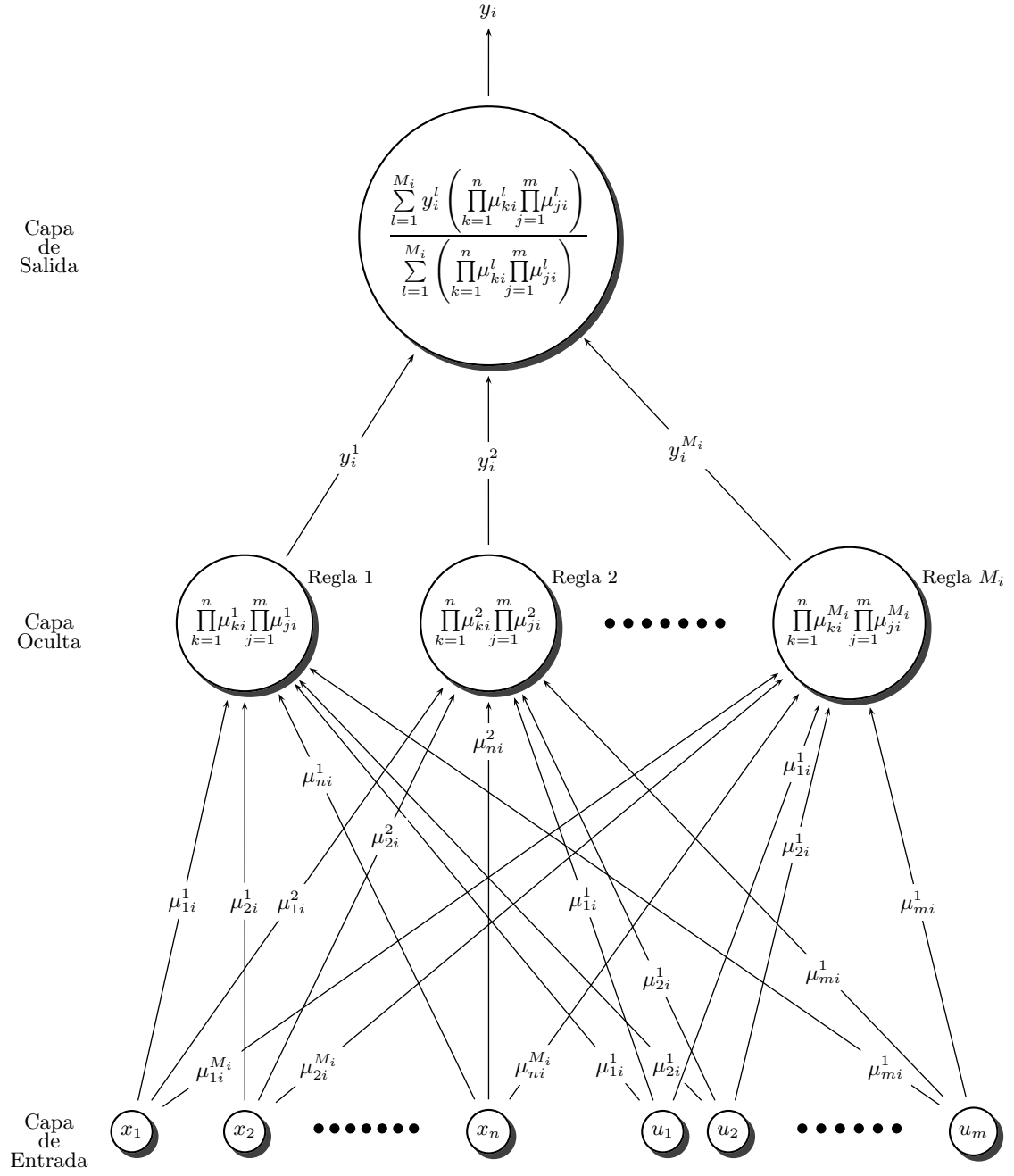


Figura 3.10: Red neuronal equivalente al modelo borroso de la planta.

supervisado que consiste en minimizar el error de estimación (comúnmente cuadrático) por medio de gradiente descendiente, por lo que la parte esencial del algoritmo es el cálculo de las derivadas parciales de dicho error con respecto a los parámetros del modelo.

En una red neuronal [14], el algoritmo BP [268] realiza dos pasadas a través de las diferentes capas de la red, una hacia delante y otra hacia atrás. En la primera pasada (hacia adelante), se aplica un vector de entrada a los nodos de entrada de la red, propagándose su efecto capa a capa a través de la misma. Durante esta fase los pesos sinápticos permanecen fijos. Después de la última capa se obtiene, como consecuencia del vector de entrada introducido, un vector respuesta. Durante la segunda pasada (hacia atrás), todos los pesos sinápticos se ajustan en función de una regla de corrección del error (una generalización de la regla delta del algoritmo de aprendizaje de Widrow y Hoff [274]). La señal de error resultante, entendiendo por señal de error la diferencia entre la respuesta de salida de la red y la deseada, se propaga entonces hacia atrás, en sentido contrario a las conexiones sinápticas. Este método de ajuste de pesos se repite hasta que ya no se producen cambios en los mismos, lo cual se conoce como convergencia del algoritmo de aprendizaje. En este punto, el proceso de aprendizaje ha finalizado y la respuesta real de la red es lo más parecida posible a la deseada.

En el caso de un modelo borroso, la aplicación del algoritmo anterior funciona en la pasada hacia delante de forma análoga a la red neuronal, tal como se muestra en el algoritmo de la figura 3.11, sólo que en vez de permanecer fijos los pesos sinápticos, se mantienen sin cambio los parámetros del antecedente y el consecuente de las reglas. En la pasada hacia atrás, la señal de error resultante se propaga para ajustar los parámetros del modelo según la regla de corrección del error. Si se define $J(k)$ como la función de coste, error cuadrático medio, en la k -ésima iteración del algoritmo, su expresión vendrá dada por:

$$J(k) = \frac{1}{2} (y(k) - \hat{y}(k))^2; \quad (3.65)$$

donde $y(k)$ es la salida real de la k -ésima iteración e $\hat{y}(k)$ la salida estimada.

A partir de la función de ajuste que implementa el método de gradiente descendente (3.66) [275] se pueden obtener las expresiones de adaptación paramétrica tanto de los antecedentes como de los consecuentes de las reglas [169]:

$$\theta(k+1) = \theta(k) - \eta \left. \frac{\partial J(k)}{\partial \theta(k)} \right|_{\theta=\theta(k)}, \quad (3.66)$$

donde $\theta(k)$ es el parámetro a adaptar y η es un factor de escala del término derivativo, denominado factor de aprendizaje del algoritmo, y que afecta al tiempo de convergencia del mismo. Por ejemplo, para una función de pertenencia gaussiana, el ajuste del centro, $c(k)$, y ancho, $\beta(k)$, de la función dentro de la regla l , vendrá dado por las expresiones (3.67) y (3.68), respectivamente; mientras que el ajuste de los parámetros de un consecuente lineal vendrá dado por la ecuación (3.69), siendo $e(k) = y(k) - \hat{y}(k)$ el error de modelado de la k -ésima iteración.

```

1: Mientras  $k < \text{época final}$  hacer
2:   Para cada punto de entrenamiento hacer
3:     Calcular  $\hat{y}(k) \rightarrow$  Pasada hacia adelante
4:     Calcular  $e(k) = y(k) - \hat{y}(k)$ 
5:     Actualizar antecedentes y consecuentes  $\rightarrow$  Pasada hacia atrás
6:   Fin Para
7:   Si  $e(k) < \text{error deseado}$  entonces
8:     Finalizar el algoritmo
9:   Fin Si
10:  Calcular  $\eta(k+1)$  según la ecuación (3.70)
11:  Retornar parámetros ajustables del modelo borroso
12: Fin Mientras

```

Figura 3.11: Algoritmo de adaptación neuroborrosa.

$$c_i^l(k+1) = c_i^l(k) - 2\eta e(k) \frac{\prod_{i=1}^n \mu_i^l(x_i(k), c_i^l(k), \beta_i^l(k))}{\sum_{l=1}^M \left(\prod_{i=1}^n \mu_i^l(x_i(k), c_i^l(k), \beta_i^l(k)) \right)} \frac{(x_i(k) - c_i^l(k))}{(\beta_i^l(k))^2} \quad (3.67)$$

$$\beta_i^l(k+1) = \beta_i^l(k) - 2\eta e(k) \frac{\prod_{i=1}^n \mu_i^l(x_i(k), c_i^l(k), \beta_i^l(k))}{\sum_{l=1}^M \left(\prod_{i=1}^n \mu_i^l(x_i(k), c_i^l(k), \beta_i^l(k)) \right)} \frac{(x_i(k) - c_i^l(k))^2}{(\beta_i^l(k))^3} \quad (3.68)$$

$$\theta_i^l(k+1) = \theta_i^l(k) - 2\eta e(k) \frac{\prod_{i=1}^n \mu_i^l(x_i(k), c_i^l(k), \beta_i^l(k))}{\sum_{l=1}^M \left(\prod_{i=1}^n \mu_i^l(x_i(k), c_i^l(k), \beta_i^l(k)) \right)} \quad (3.69)$$

Para mejorar la eficiencia del proceso de identificación, el factor de aprendizaje puede hacerse adaptativo [276], siendo:

$$\eta(k+1) = \begin{cases} \Delta\eta \times \eta(k) & \text{si } J(k) \leq J(k-1) \\ \nabla\eta \times \eta(k) & \text{si } J(k) \geq r \times J(k-1) \\ \eta(k) & \text{en otro caso,} \end{cases} \quad (3.70)$$

donde r , $\Delta\eta$ y $\nabla\eta$ son valores ajustables definidos como margen de tolerancia y momentos, respectivamente.

En [271] se pueden consultar las funciones de ajuste para el método del gradiente descendente de los tipos de funciones de pertenencia más comunes, así como varios casos prácticos de modelado mediante esta técnica.

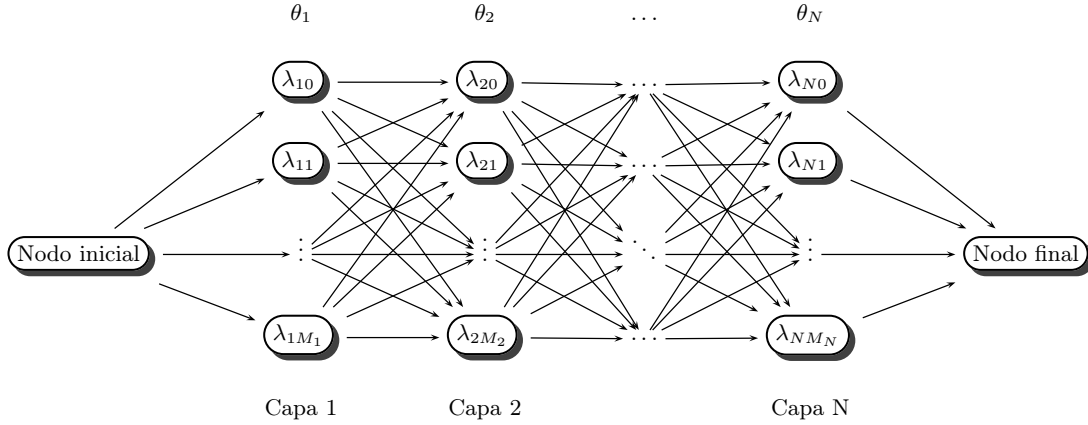


Figura 3.12: Grafo para el ajuste de modelos TSK mediante OCH.

Metodología bioinspirada híbrida

Recientemente, los algoritmos bioinspirados se han mostrado como uno de los más prometedores campos de investigación en el desarrollo de metaheurísticas. Dentro de estas metaheurísticas se encuentran, entre otros, los algoritmos de optimización basados en colonias de hormigas (OCH). La metaheurística OCH [277, 278] se basa en simular la forma en que las hormigas llevan la comida hasta el hormiguero recorriendo un camino de longitud mínima. En el proceso de búsqueda de comida, las hormigas van depositando una sustancia denominada feromona, y en base a esta sustancia, dirigen su búsqueda. Cuando una hormiga llega a una intersección, decide el camino a seguir de un modo probabilístico en función de la feromona depositada en cada ramificación. Este movimiento continuado provoca que los caminos más cortos sean más frecuentados, por lo que reciben mayor cantidad de feromona, mientras que los más largos serán progresivamente abandonados y la feromona asociada acabará por evaporarse. De este modo, se acaba formando un camino de longitud mínima entre el hormiguero y la comida. Dependiendo del problema, esta longitud vendrá definida por una función de coste a minimizar.

Para poder aplicar este tipo de algoritmos, es necesario que el problema se pueda representar como un camino en un grafo. Este grafo se representará como $G = (C, L)$, donde C es un conjunto de nodos y L es un conjunto de arcos que conectan los nodos de C .

Para poder aplicar los algoritmos OCH al modelado borroso TSK se ha diseñado un grafo G donde el conjunto de nodos C se ha organizado por capas, ver figura 3.12. Cada capa c_i es asociada con un parámetro del modelo borroso, θ_i , siendo $i = 1..N$ un índice que recorre todos los parámetros del modelo. Cada capa se compone por un conjunto de nodos λ_{ij} , con $j = 1..M_i$, donde M_i es el número de valores posibles del parámetro i . Cada nodo de una capa se conecta con todos los nodos de la capa siguiente mediante arcos, de esta forma, el ajuste de los parámetros del modelo TSK puede ser interpretado como el recorrido de las hormigas por el grafo G guiadas por la feromona. Cada nodo recorrido por una hormiga indica la elección de un valor para cada uno de los parámetros del modelo borroso, por lo que cada camino completo es un modelo candidato.

-
- 1: **Para** cada hormiga **hacer**
 - 2: Inicialización de la hormiga
 - 3: Situación de la hormiga en el nodo inicial
 - 4: **Mientras** nodo actual $\neq$ nodo final **hacer**
 - 5: Calcular las probabilidades de transición mediante la ecuación (3.71)
 - 6: Ejecutar la transición
 - 7: **Fin Mientras**
 - 8: **Fin Para**
 - 9: **Retornar** nodos recorridos del grafo
-

Figura 3.13: Algoritmo de activación de las hormigas.

Las hormigas se sitúan en el nodo inicial, y van recorriendo el grafo escogiendo entre los distintos caminos de forma probabilística según se describe en el algoritmo de la figura 3.13. La probabilidad de que una hormiga elija el j -ésimo nodo destino partiendo del i -ésimo nodo, viene determinada por la cantidad de feromona depositada en el arco que une dichos nodos según la ecuación:

$$p_{ij}(k) = \frac{\tau_{ij}(k)}{\sum_{n=1}^{M_j} \tau_{in}(k)}, \quad (3.71)$$

donde $\tau_{ij}(k)$, con $i = 1..N$ y $j = 1..M_i$, es la cantidad de feromona del arco $i - j$ en la k -ésima iteración. El denominador de (3.71) es un factor normalizador que hace que la suma de las probabilidades de todos los arcos disponibles sea la unidad.

Una vez que se han obtenido las soluciones aportadas por las hormigas que han recorrido el grafo, se procede a evaluarlas. Para ello se aplica la función de coste (3.65) empleada en el algoritmo de gradiente descendente. A continuación de deben dotar de feromona los arcos del grafo en función de la calidad de las soluciones obtenidas. Puesto que interesa asignar mayor cantidad de feromona a los arcos que han aportado mejores soluciones, el incremento de feromona depende de la inversa de la función de coste y de un factor de evaporación de la feromona existente por las iteraciones previas del algoritmo, $0 < \rho < 1$, según se muestra en la expresión (3.72).

$$\tau_{ij}(k+1) = (1 - \rho)\tau_{ij}(k) + 1/J(k) \quad (3.72)$$

Nótese que en el grafo no pueden ser representados todos los valores posibles de los parámetros ajustables, ya que el grafo tendría dimensión infinita. Por lo tanto, es necesario discretizar el espacio de búsqueda asignando a cada parámetro un vector de valores posibles. Esta discretización puede realizarse de forma homogénea estableciendo un valor mínimo, un valor máximo, y la precisión deseada para cada uno de los parámetros del modelo. Puesto que el rango de valores y la precisión de cada uno de los parámetros puede ser distinto al del resto, el número de nodos totales de cada capa M_i es independiente del resto de capas del grafo.

El procedimiento de modelización híbrido [169, 272], ver figura 3.14, se basa, en primer lugar, en la aplicación del algoritmo OCH de forma que se generen soluciones iniciales

-
- 1: **Mientras** No se cumpla la condición de parada **hacer**
 - 2: Generación y activación de las hormigas
 - 3: Calcular $J(k)$ para cada hormiga, ecuación (3.65)
 - 4: Actualizar la feromona de los arcos, ecuación (3.72)
 - 5: **Fin Mientras**
 - 6: Aplicar gradiente descendente a la mejor hormiga
 - 7: **Retornar** parámetros ajustables del modelo borroso
-

Figura 3.14: *OCH-neuroborroso.*

prometedoras. La ventaja que ofrece la utilización de colonias de hormigas, es que es posible mantener varias soluciones candidatas activas en diferentes puntos del espacio de búsqueda. Esta simultaneidad permite explorar varios caminos hacia el mínimo global de la función de coste. Una vez concluido el algoritmo OCH, se aplica una búsqueda local basada en gradiente descendente para explotar las mejores soluciones obtenidas, generando así los valores finales de los parámetros ajustables del modelo borroso. La búsqueda local mediante gradiente descendente permite la optimización de los parámetros de forma continua, por lo que se corrigen los problemas a la hora de obtener una solución óptima generados por la discretización del espacio de búsqueda del algoritmo OCH.

3.3.3. Validación

Una vez que se ha determinado la estructura del modelo borroso, y se han estimado los parámetros, es necesario estudiar el comportamiento del modelo para conocer la calidad del mismo. Este proceso se conoce como validación [279].

Uno de los procedimientos más utilizados para validar un modelo es el denominado *validación cruzada* [280]. El método de validación cruzada consiste en dividir los datos de entrada-salida en dos subconjuntos. Esta división suele hacerse en una razón que puede variar entre el 70/30 % y el 80/20 %. El mayor de los subconjunto de datos se utiliza para generar el modelo borroso mediante los procedimientos descritos anteriormente, mientras que los datos restantes se emplean para calcular un índice de calidad del modelo. Aunque existen distintos métodos de obtener un índice de calidad para la fase de validación [281], como el criterio de Sugeno [209,282], el análisis de residuos [283], o la interpretabilidad del modelo [264,266,284]; el más comúnmente empleado es el error cuadrático medio (MSE). El error cuadrático medio se define mediante la ecuación (3.73), donde $y(k)$ e $\hat{y}(k)$ son respectivamente la salida real y estimada del sistema, y N es el número total de muestras utilizadas para calcular el error.

$$MSE = \frac{1}{N} \sum_{k=1}^N (y(k) - \hat{y}(k))^2 \quad (3.73)$$

Puesto que los datos empleados para la validación no han sido utilizados durante la fase de estimación, si se produjese un sobreentrenamiento del modelo sería posible detectarlo, ya que el error promedio del modelo en la fase de validación sería anormalmente alto con respecto al error obtenido durante la fase de estimación.

Si se han obtenido varios modelos de un mismo sistema y se desea escoger uno de ellos, la validación puede ser una buena herramienta de selección. En este caso es aconsejable dividir el conjunto de datos de entrada-salida en tres subconjuntos, por ejemplo mediante una razón comprendida entre el 70/15/15 % y el 80/10/10 %. Los dos primeros subconjuntos de datos se emplean para la identificación y validación de los modelos. A continuación se debe escoger el modelo que presente un menor error promedio en la fase de validación, aunque pueden emplearse otros criterios como la interpretabilidad o sencillez de los modelos obtenidos. Por último, el tercer subconjunto de datos se emplea para estimar el error promedio del modelo elegido. En este caso no se deben emplear los datos de la fase de validación para el cálculo de dicho error, ya que si el modelo se ha escogido en base a estos datos, la estimación del error estará sesgada; es decir, la estimación del error con la muestra de validación es optimista.

3.4. Diseño del software

Con objeto de facilitar la programación de los distintos algoritmos de este texto, se ha desarrollado un software capaz de almacenar todos los datos de un modelo borroso TSK completamente general. Este software se ha diseñado siguiendo una metodología de orientación a objetos y se ha programado en C++, por lo que la estructuración y jerarquización obtenidas permite que éste pueda ser ampliado y/o modificado de forma sencilla. De esta forma, se facilita un esquema de programación sencillo de emplear en el desarrollo de algoritmos complejos, se obtiene una buena velocidad de ejecución, y se garantiza la compatibilidad con cualquier sistema operativo.

También se han programado una serie de interfaces entre las distintas clases y utilidades y MATLAB®, permitiendo la compilación de todo el software desarrollado sobre dicho entorno mediante archivos MEX [285, 286]. De esta forma, el software desarrollado se convierte en un complemento idóneo de la *Fuzzy Logic Toolbox* de MATLAB® [287] para el análisis y la síntesis de sistemas de control borroso.

Todos los ficheros de cabecera que definen estas clases y sus funciones auxiliares pueden consultarse en el apéndice D.

El esquema de relación entre clases, presentado en formato UML (*Unified Modeling Language*) [288], se muestra en la figura 3.15. Como se puede observar, se ha definido una clase **Sistema**, que representa un modelo borroso completo. Esta clase está formada por **Reglas**, que a su vez contienen funciones de **Pertenencia**.

La clase **Pertenencia**, ver figura 3.16, contiene los métodos y atributos comunes a todas las funciones de pertenencia, como el tipo (**tipo_fp**), el número de parámetros (**n**) y sus valores (**param**). Las funciones de evaluación de la función de pertenencia (**eval**) y su derivada (**evalder**), se han declarado como funciones virtuales, de forma que cada tipo de función redefina estos métodos según sus características, pero manteniendo un interfaz común. Lo mismo sucede con el método **test**, que proporciona una herramienta para comprobar, en tiempo de ejecución, que la función de pertenencia se ha definido correctamente.

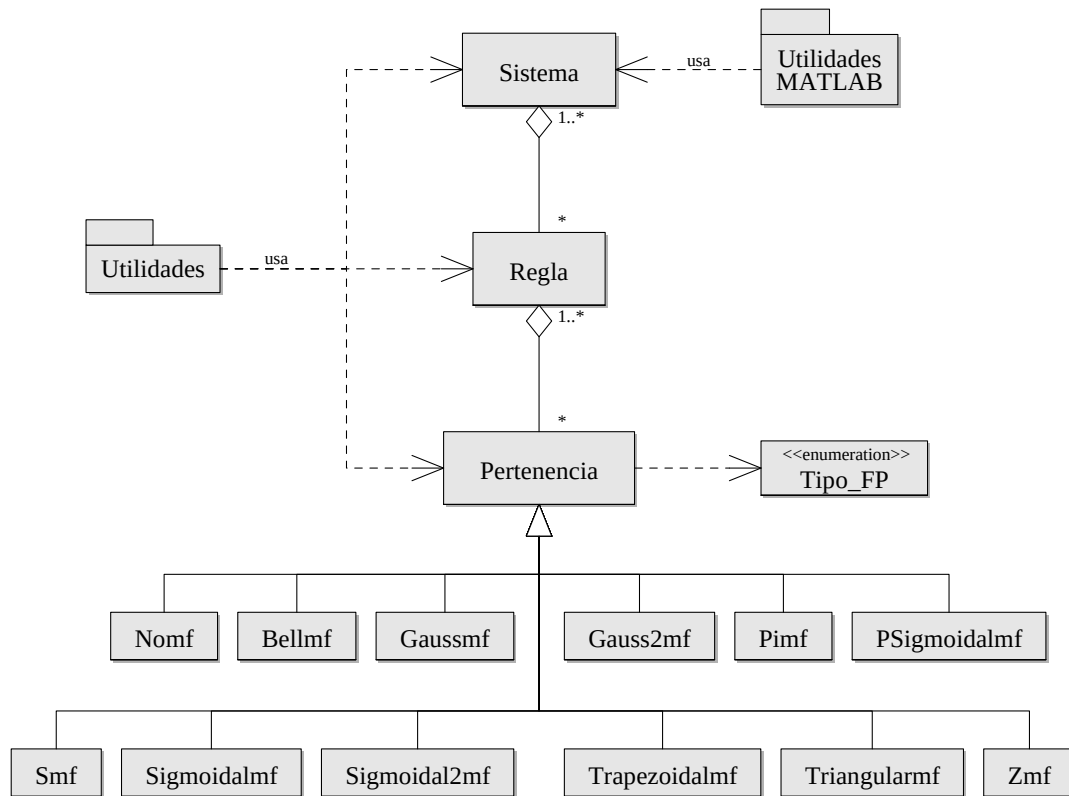


Figura 3.15: Diagrama UML de la estructura de clases.

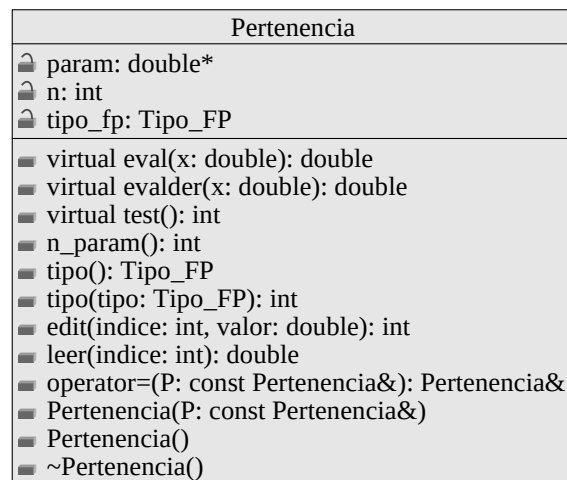


Figura 3.16: Clase Pertenencia.

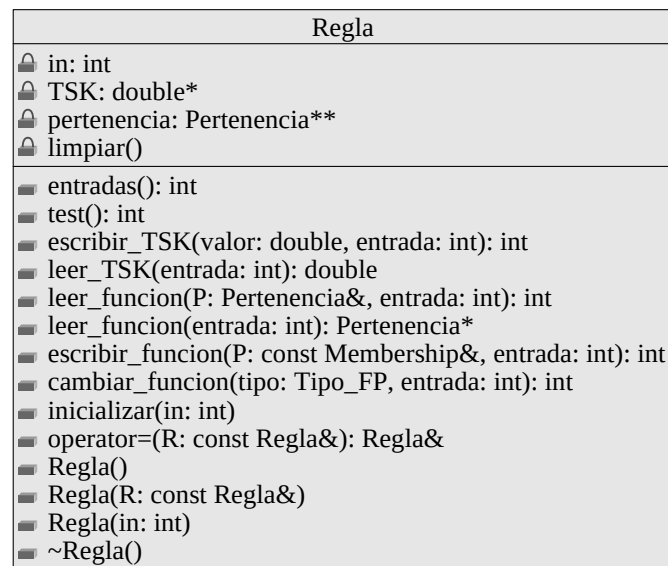


Figura 3.17: Clase Regla.

De la clase **Pertenencia** se derivan todos los tipos de funciones de pertenencia soportados por el software. Actualmente se permite la utilización de las funciones de pertenencia más comunes: triangular, trapezoidal, gaussiana simple y doble, campana, S , Z , Π , sigmoideal, y las funciones de pertenencia diferencia y producto de sigmoidales; aunque la ampliación a otras funciones de pertenencia se puede realizar de forma sencilla, mediante la derivación de la clase **Pertenencia** y la actualización de la enumeración **Tipo_FP**. La clase **Nomf** es una función de pertenencia que no requiere parámetros y que se utiliza para facilitar la inicialización de las reglas.

La clase **Regla**, cuya estructura se muestra en la figura 3.17, proporciona el objeto de almacenamiento de las reglas del modelo borroso. En cada objeto de tipo **Regla** se almacena un vector con los antecedentes (**pertenencia**) y el consecuente de tipo TSK de la regla (**TSK**). Además, esta clase proporciona los métodos necesarios para facilitar la programación y modificación de reglas borrosas dentro del software.

En la figura 3.18 se muestra la clase **Sistema**. El cometido de esta clase es permitir el almacenamiento en un solo objeto de todo un modelo borroso, por lo tanto, la clase **Sistema** contiene todos los atributos necesarios para almacenar las características del modelo borroso: número de entradas (**in**) y salidas (**out**), número de reglas que se emplean para inferir cada una de las salidas (**N**), la definición de dichas reglas (**R**), y los límites del universo de discurso de las variables de entrada y salida del modelo borroso. Al igual que el resto de clases, la clase **Sistema** también provee los métodos necesarios para facilitar el tratamiento de modelos borrosos, como la lectura y escritura de datos del modelo, su inicialización o la copia de un modelo borroso completo.

Junto con las distintas clases presentadas, también se ha desarrollado un paquete de utilidades genérico, y otro específico para MATLAB®. El paquete genérico se muestra en

Sistema
in: int out: int N: int* R: Regla** min_in: double* min_out: double* max_in: double* max_out: double*
reglas(salida: int): int entradas(): int salidas(): int in_min(entrada: int): double in_max(entrada: int): double out_min(salida: int): double out_max(salida: int): double in_min(limite: double*): in_max(limite: double*): out_min(limite: double*): out_max(limite: double*): test(): int escribir_Regla(R: const Regla&, r: int, salida: int): int leer_Regla(R: Regla&, r: int, salida: int): int leer_Regla(r: int, salida: int): Regla* inicializar(in: int, out: int, N: int*) operator=(S: const Sistema&): Sistema& Sistema() Sistema(S: const Sistema&) Sistema(in: int, out: int, N: int*) ~Sistema()

Figura 3.18: Clase Sistema.

la figura 3.19, y contiene una serie de funciones destinadas a facilitar la programación de algoritmos que empleen sistemas borrosos:

- **obtenerAntecedentes** y **obtenerConsecuentes** permiten la obtención de los parámetros adaptables de un modelo borroso.
- La función **activacion** calcula el grado de activación de las reglas de un modelo en un punto dado.
- La función **subSistema** obtiene desde un nuevo modelo borroso de entrada, otro modelo compuesto por algunas de las reglas del modelo inicial.
- La función **imprimirSistema** devuelve una representación lingüística de las reglas de un modelo borroso.
- **TXT2Sistema** y **Sistema2TXT** facilitan la lectura y almacenamiento de modelos borrosos en ficheros de tipo texto siguiendo una estructura muy similar a las reglas lingüísticas, lo que permite su creación y edición fácilmente.

Utilidades
TXT2Sistema (fichero: char*, S: Sistema&): int Sistema2TXT (S: Sistema&, fichero: char*): int Evaluar (x: double*, Salida: double*, S: Sistema&): int Evaluar (x: double*, Salida: double*, S: Sistema&, C: Sistema&): int linealizar (S: Sistema&, x: double*, u: double*, A: double**, B: double**, F: double*): int imprimirSistema (fichero: char*, S: Sistema&, entradas: char**, salidas: char**, decimales: int):int numeroDatosAntecedentes (S: Sistema&): int numeroDatosAntecedentes (S: Sistema&, out: int, rule: int): int numeroDatosConsecuentes (S: Sistema&): int numeroDatosConsecuentes (S: Sistema&, out: int, rule: int): int obtenerAntecedentes (S: Sistema&, parametros: double*, num_datos: int): int obtenerAntecedentes (S: Sistema&, parametros: double*, num_datos: int, out: int, rule: int): int obtenerConsecuentes (S: Sistema&, parametros: double*, num_datos: int): int obtenerConsecuentes (S: Sistema&, parametros: double*, num_datos: int, out: int, rule: int): int almacenarAntecedentes (S: Sistema&, parametros: double*, num_datos: int): int almacenarAntecedentes (S: Sistema&, parametros: double*, num_datos: int, out: int, rule: int): int almacenarConsecuentes (S: Sistema&, parametros: double*, num_datos: int): int almacenarConsecuentes (S: Sistema&, parametros: double*, num_datos: int, out: int, rule: int): int activacion (S: Sistema&, salida: int, regla: int, Punto: double*, W: double*): int activacion (S: Sistema&, salida: int, regla: int, Punto: double*, W: double*, dW: double*): int subSistema (S: Sistema&, ParteS: Sistema&, nreglas: int, salidas: int*, reglas: int*): int

Figura 3.19: *Paquete Utilidades.*

- La función **Evaluar** realiza el cálculo de los modelos borrosos de la planta, del controlador y del sistema de lazo cerrado que se muestran en las figuras 3.2, 3.3 y 3.5, respectivamente.
- La función **linealizar** se verá en el capítulo 4.

El paquete de utilidades de MATLAB® mostrado en la figura 3.20, incluye la función **FIS2Sistema** para leer un objeto de tipo **Sistema**, desde una variable **FIS** (*Fuzzy Inference System*) de MATLAB®, y la versión opuesta de esta función (**Sistema2FIS**). Además se incluye la función **readModel**, que lee un objeto **Sistema** desde un fichero de texto, un fichero **FIS**, o una variable **FIS** indistintamente.

Con idea de facilitar lo más posible la utilización de los algoritmos diseñados, se han programado las funciones **MEX** que se listan a continuación⁴. Su ejecución se realiza desde

⁴Se han programado otras funciones que se presentarán en el capítulo siguiente.

Utilidades MATLAB
FIS2Sistema (FIS: mxArray*, S: Sistema&): int Sistema2FIS (S: Sistema&, FIS: mxArray*): int readModel (modelo: mxArray*, S: Sistema&):int

Figura 3.20: *Paquete Utilidades MATLAB.*

el mismo entorno de MATLAB® de forma completamente transparente al usuario, pero con la velocidad de procesamiento de un programa compilado en C++.

- **activation:** Calcula el grado de activación de las reglas de un sistema borroso.
- **antec2mat:** Obtiene un vector con todos los parámetros de los antecedentes del modelo.
- **conseq2mat:** Obtiene un vector con todos los parámetros de los consecuentes del modelo.
- **fis2txt:** Escribe un fichero TXT con el modelo borroso dado desde una variable FIS.
- **fuz2mat:** Obtiene un vector con todos los parámetros del modelo (antecedentes y consecuentes).
- **fuzcomb:** Une varios modelos borrosos en uno sólo siempre que tengan el mismo número de entradas.
- **fuzeval:** Calcula la salida de un modelo borroso mediante los algoritmo de las figuras 3.2, 3.3 o 3.5 según el caso.
- **fuzprint:** Crea un fichero de texto con la representación lingüística de un modelo borroso.
- **mat2antec:** Escribe los antecedentes de un modelo borroso con los datos de un vector dado.
- **mat2conseq:** Escribe los consecuentes de un modelo borroso con los datos de un vector dado.
- **mat2fuz:** Escribe todos los parámetros de un modelo borroso (antecedentes y consecuentes) con los datos de un vector dado.
- **subsistema:** Crea un modelo borroso a partir de una serie de reglas de otro modelo.
- **txt2fis:** Lee un modelo borroso desde un fichero de texto.

3.5. Conclusiones y comentarios bibliográficos

Este capítulo se ha centrado en el modelado borroso de los sistemas dinámicos de control. A lo largo del mismo se ha presentado una metodología que permite afrontar, en base a datos de entrada/salida, el modelado borroso en forma de ecuaciones de estado de un sistema de control multivariable no lineal. La formalización matemática del sistema de control borroso se ha abordado de forma totalmente general, es decir, sin aplicar ningún tipo de restricciones ni al tamaño del vector de estado, ni al del vector de control, ni al tipo de función de pertenencia utilizada en la definición de los conjuntos borrosos. Se ha considerado que el controlador es completamente independiente del proceso de identificación del modelo borroso de la planta, de forma que la partición del universo de discurso de las variables de estado de la planta y del controlador borroso no tienen porqué coincidir; por la misma razón, los modelos borrosos de la planta y del controlador no tienen porqué tener el mismo número de reglas. Teniendo en cuenta estas consideraciones se ha obtenido un modelo matemático equivalente tanto del modelo borroso de la planta como del controlador. También se ha obtenido un modelo matemático equivalente para el sistema de control borroso de lazo cerrado, lo que constituye el paso previo para la síntesis de sistemas de control con estabilidad garantizada.

La introducción al modelado mediante variables de estado se puede consultar en las referencias [219] a [224]. En las referencias [225] a [233] se profundiza en los sistemas borrosos de tipo TSK y en [234] a [239] en los sistemas borrosos de tipo Mamdani. Respecto al proceso de identificación de la planta, las referencias [209], [240] y de [245] a [262] reúnen una amplia información sobre la identificación de la estructura del modelo borroso. De entre las técnicas de adaptación paramétrica existentes, en este capítulo se han estudiado la ya clásica metodología neuroborrosa basada en gradiente descendente y se ha propuesto una técnica más novedosa, denominada híbrida, basada en la hibridación de la técnica anterior con la heurística bioinspirada. La metodología neuroborrosa puede consultarse en las referencias [169], [268] y [270] a [276], y la metodología híbrida en [169], [243], [244], [272], [277] y [278].

Las referencias [279] a [284] muestran como realizar el proceso de validación de un modelo y cómo escoger el mejor de entre varios en el caso de tener distintas alternativas.

Finalmente, el final del capítulo se ha dedicado al diseño del software necesario para almacenar todos los datos de los modelos borrosos estudiados anteriormente, así como de proveer las herramientas que faciliten posteriormente la programación de los distintos algoritmos que puedan hacer uso de éstos. Las referencias [285] a [287] son pertenecientes al entorno de cálculo y programación MATLAB®, mientras que [288] es un libro especializado en el lenguaje de modelado de software UML utilizado para la descripción del software.

De entre las referencias incluidas en este capítulo cabe destacar [271], donde se presenta la metodología de modelado neuroborrosa mediante el caso real de una factoría de celulosa, así como [169] y [272], donde se propone la metodología bioinspirada híbrida.

Análisis de sistemas dinámicos mediante modelos borrosos

ANALIZAR dinámicamente un sistema permite conocer su funcionamiento y la respuesta que el mismo presentará a distintos estímulos de entrada. Si nos referimos a un sistema de control, el análisis de su estabilidad resulta fundamental, ya que sin estabilidad ni siquiera puede hablarse de sistema de control. Tal como se ha visto en el capítulo anterior, a partir de una planta, aunque esta sea completamente desconocida, es posible obtener un modelo borroso si se posee un conjunto suficiente de datos de entrada/-salida. Una vez se posee un modelo borroso, mediante el análisis de éste se puede extraer información de la planta y así facilitar su conocimiento. En este capítulo se presentan algunas herramientas de análisis que van a permitir extraer información de la planta a partir de su modelo borroso. El capítulo está estructurado en cinco secciones. En la primera sección, partiendo del modelo matemático equivalente obtenido en el capítulo anterior, se obtiene la linealización de un modelo borroso de estado completamente general, y que por lo tanto puede ser no lineal y multivariable. En la segunda sección se obtiene la matriz jacobiana de un sistema de control borroso, nuevamente considerando que tanto el modelo borroso de la planta como el del controlador son completamente generales y que, por lo tanto, pueden ser no lineales y multivariables. La tercera sección del capítulo está dedicada a la extracción de los estados de equilibrio de un sistema de control borroso. En esta sección se presenta un algoritmo numérico que permite obtener los estados de equilibrio tanto de un modelo borroso de estado como de un sistema de control borroso en lazo cerrado. También se propone la inclusión de la matriz jacobiana calculada con anterioridad en dichos algoritmos para así mejorar su velocidad de convergencia y precisión. La cuarta sección presenta formalmente el algoritmo de linealización de un modelo borroso y el algoritmo para el cálculo de la matriz jacobiana de un sistema de control borroso. Finalmente, la quinta sección recoge las principales conclusiones del capítulo, así como una revisión de las referencias bibliográficas incluidas en el mismo.

4.1. Introducción

La herencia tradicional de la ingeniería de control ha sido lineal, de hecho, cuando es posible asumir, como ocurre en muchas aplicaciones, que los problemas son suficientemente locales, es permisible obtener su solución usando métodos lineales. Sin embargo, cuando los problemas son de naturaleza global, el uso de métodos no lineales se convierte en una necesidad.

La razón fundamental que exige el uso de modelos no lineales es que la dinámica de los modelos lineales no es lo suficientemente rica como para describir una serie de fenómenos que se dan de forma asidua en la vida real [221]. El comportamiento dinámico de los sistemas lineales, independientemente de su orden, está gobernado esencialmente por los autovalores de su matriz de estado. Sin embargo, los sistemas no lineales tienen mucho más margen de maniobra. Los sistemas de orden 2 pueden originar oscilaciones autoexcitadas, conocidas como *ciclos límite*, pero cuando la dimensión del sistema no lineal se aumenta de 2 a 3, puede aparecer un nuevo fenómeno dinámico realmente excitante: las trayectorias presentan un comportamiento aperiódico y son críticamente sensibles a las condiciones iniciales, aparece el *caos* [289]. Otros fenómenos dinámicos exclusivos de los sistemas no lineales son los múltiples estados de equilibrio y las bifurcaciones [223].

El punto de partida idóneo en el análisis de un sistema no lineal es su representación mediante un modelo matemático, generalmente un modelo de estado. Cuando el sistema es conocido y su dinámica no es excesivamente compleja, es posible obtener un modelo de estado del sistema; sin embargo, en multitud de ocasiones el sistema objeto de estudio puede ser demasiado complejo, ya sea por su dinámica o por su dimensión, o simplemente porque se desconocen las ecuaciones que gobiernan su funcionamiento. En estos casos, la lógica borrosa se presenta como una poderosa herramienta de modelado, ya que, tal como se vio en el capítulo 3, permite la obtención de un modelo borroso del sistema a partir de una colección de datos de entrada/salida de éste.

Pese a que la lógica borrosa es conocida por su capacidad para tratar la imprecisión, no hay que confundir este hecho con la exactitud y rigurosidad de la misma. Como se ha demostrado en el capítulo anterior, un modelo borroso es un modelo matemático que, además, se comporta como un aproximador universal. Por lo tanto, un modelo de estado representado mediante un modelo borroso, es susceptible de ser estudiado utilizando las herramientas matemáticas existentes para los sistemas no lineales.

El conocimiento de los estados de equilibrio que posee un sistema, así como la estabilidad de dichos estados, son datos que pueden resultar muy interesantes a la hora de analizar o diseñar un sistema de control. Si se parte de un sistema completamente desconocido, esta información puede aclarar en muchos aspectos el funcionamiento del mismo, así como facilitar el diseño de un sistema de control adecuado. En este capítulo se presentan las herramientas matemáticas desarrolladas para extraer dicha información de un modelo borroso [231, 290, 291], así como el software creado para tal fin [168].

4.2. Linealización de un modelo borroso de estado

La linealización es una de las técnicas más empleadas en el diseño de sistemas de control no lineales. Aunque es una técnica no muy recomendable en muchos casos, ya que se desprecian los efectos de las no linealidades de los sistemas controlados, sí que puede ser válida para controlar sistemas no muy complejos, o cuya dinámica sea conocida, en regiones donde su comportamiento sea aproximadamente lineal.

Además de como método de control, la linealización puede emplearse para obtener información de un sistema no lineal. En sabido que, con algunas salvedades, el comportamiento de un sistema no lineal en torno a un estado de equilibrio es muy similar al del sistema linealizado en torno a dicho estado [222, 223, 224]; por lo tanto, la obtención del sistema lineal equivalente de un modelo borroso no lineal, puede ser una herramienta muy útil para obtener información del sistema modelado.

Sea el modelo de estado genérico de un sistema no lineal dado por:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \quad (4.1)$$

el desarrollo en series de Taylor de orden 1 de dicho sistema en torno a un punto $(\mathbf{x}_0, \mathbf{u}_0)$, determina que éste se puede aproximar por la expresión (4.2), donde las matrices del sistema lineal se calculan mediante las expresiones (4.3), (4.4) y (4.5).

$$\dot{\mathbf{x}}(t) \approx \mathbf{A}_0 + \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \quad (4.2)$$

$$\mathbf{A}_0 = \mathbf{f}(\mathbf{x}_0, \mathbf{u}_0) \quad (4.3)$$

$$\mathbf{A} = \left(\begin{array}{cccc} \frac{\partial f_1(\mathbf{x}(t), \mathbf{u}(t))}{\partial x_1} & \frac{\partial f_1(\mathbf{x}(t), \mathbf{u}(t))}{\partial x_2} & \cdots & \frac{\partial f_1(\mathbf{x}(t), \mathbf{u}(t))}{\partial x_n} \\ \frac{\partial f_2(\mathbf{x}(t), \mathbf{u}(t))}{\partial x_1} & \frac{\partial f_2(\mathbf{x}(t), \mathbf{u}(t))}{\partial x_2} & \cdots & \frac{\partial f_2(\mathbf{x}(t), \mathbf{u}(t))}{\partial x_n} \\ \cdots & \cdots & \ddots & \cdots \\ \frac{\partial f_n(\mathbf{x}(t), \mathbf{u}(t))}{\partial x_1} & \frac{\partial f_n(\mathbf{x}(t), \mathbf{u}(t))}{\partial x_2} & \cdots & \frac{\partial f_n(\mathbf{x}(t), \mathbf{u}(t))}{\partial x_n} \end{array} \right) \bigg|_{(\mathbf{x}_0, \mathbf{u}_0)} \quad (4.4)$$

$$\mathbf{B} = \left(\begin{array}{cccc} \frac{\partial f_1(\mathbf{x}(t), \mathbf{u}(t))}{\partial u_1} & \frac{\partial f_1(\mathbf{x}(t), \mathbf{u}(t))}{\partial u_2} & \cdots & \frac{\partial f_1(\mathbf{x}(t), \mathbf{u}(t))}{\partial u_m} \\ \frac{\partial f_2(\mathbf{x}(t), \mathbf{u}(t))}{\partial u_1} & \frac{\partial f_2(\mathbf{x}(t), \mathbf{u}(t))}{\partial u_2} & \cdots & \frac{\partial f_2(\mathbf{x}(t), \mathbf{u}(t))}{\partial u_m} \\ \cdots & \cdots & \ddots & \cdots \\ \frac{\partial f_n(\mathbf{x}(t), \mathbf{u}(t))}{\partial u_1} & \frac{\partial f_n(\mathbf{x}(t), \mathbf{u}(t))}{\partial u_2} & \cdots & \frac{\partial f_n(\mathbf{x}(t), \mathbf{u}(t))}{\partial u_m} \end{array} \right) \bigg|_{(\mathbf{x}_0, \mathbf{u}_0)} \quad (4.5)$$

Si el punto $(\mathbf{x}_0, \mathbf{u}_0)$ es un estado de equilibrio del sistema, todos los elementos de la matriz $\mathbf{A}_0$ serán nulos, ya que por definición, un estado de equilibrio hace nula la ecuación de estado.

Si el sistema (4.1) se representa mediante un modelo borroso, tal como se vio en el capítulo 3, su equivalente matemático viene dado por la expresión (3.21), reescrita a continuación por claridad:

$$\dot{x}_i = f_i(\mathbf{x}, \mathbf{u}) = \sum_{k=0}^n a_{ki} \tilde{x}_k + \sum_{j=1}^m b_{ji} u_j, \quad (3.21)$$

donde i representa la i -ésima ecuación de estado del proceso; x_k , con $k = 0..n$, es la k -ésima componente del vector de estado extendido, y u_j , con $j = 1..m$, es la j -ésima señal de control. Los términos a_{ki} y b_{ji} son coeficientes variables definidos por:

$$a_{ki}(\mathbf{x}, \mathbf{u}) = \frac{\sum_{l=1}^{M_i} w_i^l(\mathbf{x}, \mathbf{u}) a_{ki}^l}{\sum_{l=1}^{M_i} w_i^l(\mathbf{x}, \mathbf{u})}, \quad b_{ji}(\mathbf{x}, \mathbf{u}) = \frac{\sum_{l=1}^{M_i} w_i^l(\mathbf{x}, \mathbf{u}) b_{ji}^l}{\sum_{l=1}^{M_i} w_i^l(\mathbf{x}, \mathbf{u})}, \quad \begin{array}{ll} k = 0, 1, \dots, n \\ i = 1, \dots, n \\ j = 1, \dots, m, \end{array} \quad (3.18)$$

donde $w_i^l(\mathbf{x}, \mathbf{u})$ es el grado de activación de las reglas del modelo borroso de la planta.

$$w_i^l(\mathbf{x}, \mathbf{u}) = \prod_{k=1}^n \mu_{ki}^l(x_k, \sigma_{ki}^l) \prod_{j=1}^m \mu_{ji}^l(u_j, \alpha_{ji}^l) \quad (3.15)$$

Si se linealiza la expresión (3.21) en torno al punto $(\mathbf{x}_0, \mathbf{u}_0)$, se puede representar el modelo matemático equivalente de un sistema borroso linealizado como:

$$\dot{x}_i \approx f_i(\mathbf{x}_0, \mathbf{u}_0) + \sum_{q=1}^n \left. \frac{\partial f_i}{\partial x_q} \right|_{(\mathbf{x}_0, \mathbf{u}_0)} x_q + \sum_{v=1}^m \left. \frac{\partial f_i}{\partial u_v} \right|_{(\mathbf{x}_0, \mathbf{u}_0)} u_v, \quad (4.6)$$

donde x_q y u_v son las componentes de los vectores $\mathbf{x}_0$ y $\mathbf{u}_0$, respectivamente. Operando con la expresión anterior:

$$\begin{aligned} \dot{x}_i \approx f_i(\mathbf{x}_0, \mathbf{u}_0) &+ \sum_{q=1}^n \left[\left. \frac{\partial}{\partial x_q} \left(\sum_{k=0}^n a_{ki} \tilde{x}_k + \sum_{j=1}^m b_{ji} u_j \right) \right|_{(\mathbf{x}_0, \mathbf{u}_0)} x_q \right] + \\ &+ \sum_{v=1}^m \left[\left. \frac{\partial}{\partial u_v} \left(\sum_{k=0}^n a_{ki} \tilde{x}_k + \sum_{j=1}^m b_{ji} u_j \right) \right|_{(\mathbf{x}_0, \mathbf{u}_0)} u_v \right], \end{aligned} \quad (4.7)$$

que desarrollando se puede expresar como:

$$\begin{aligned}
\dot{x}_i \approx f_i(\mathbf{x}_0, \mathbf{u}_0) &+ \sum_{q=1}^n \left[\frac{\partial \left(\sum_{k=0}^n a_{ki} \tilde{x}_k \right)}{\partial x_q} + \frac{\partial \left(\sum_{j=1}^m b_{ji} u_j \right)}{\partial x_q} \right] \bigg|_{(\mathbf{x}_0, \mathbf{u}_0)} x_q + \\
&+ \sum_{v=1}^m \left[\frac{\partial \left(\sum_{k=0}^n a_{ki} \tilde{x}_k \right)}{\partial u_v} + \frac{\partial \left(\sum_{j=1}^m b_{ji} u_j \right)}{\partial u_v} \right] \bigg|_{(\mathbf{x}_0, \mathbf{u}_0)} u_v.
\end{aligned} \tag{4.8}$$

A continuación se resolverán independientemente cada una de las derivadas parciales anteriores. En primer lugar,

$$\frac{\partial \left(\sum_{k=0}^n a_{ki} \tilde{x}_k \right)}{\partial x_q} = \sum_{k=0}^n \frac{\partial (a_{ki} \tilde{x}_k)}{\partial x_q} = \sum_{k=0}^n \frac{\partial a_{ki}}{\partial x_q} \tilde{x}_k + \sum_{k=0}^n a_{ki} \frac{\partial \tilde{x}_k}{\partial x_q}. \tag{4.9}$$

Puesto que $\frac{\partial \tilde{x}_k}{\partial x_q} = 1$ si $k = q$ y 0 en caso contrario:

$$\sum_{k=0}^n a_{ki} \frac{\partial \tilde{x}_k}{\partial x_q} = a_{qi}. \tag{4.10}$$

Teniendo en cuenta el valor de a_{ki} dado por la primera ecuación de (3.18):

$$\frac{\partial a_{ki}}{\partial x_q} = \frac{\partial}{\partial x_q} \left(\frac{\sum_{l=1}^{M_i} w_i^l a_{ki}^l}{\sum_{l=1}^{M_i} w_i^l} \right) = \frac{\frac{\partial \left(\sum_{l=1}^{M_i} w_i^l a_{ki}^l \right)}{\partial x_q} \sum_{l=1}^{M_i} (w_i^l) - \frac{\partial \left(\sum_{l=1}^{M_i} w_i^l \right)}{\partial x_q} \sum_{l=1}^{M_i} (w_i^l a_{ki}^l)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2}. \tag{4.11}$$

El grado de activación de las reglas de la planta, w_i^l , es un valor dependiente del vector de estado y del vector de control, mientras que los elementos a_{ki}^l son los coeficientes constantes del consecuente de las reglas de la planta. De esta forma, se puede resolver:

$$\frac{\partial \left(\sum_{l=1}^{M_i} w_i^l a_{ki}^l \right)}{\partial x_q} = \sum_{l=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} a_{ki}^l \right), \tag{4.12}$$

y

$$\frac{\partial \left(\sum_{l=1}^{M_i} w_i^l \right)}{\partial x_q} = \sum_{l=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} \right), \quad (4.13)$$

con lo que se puede expresar:

$$\begin{aligned} \frac{\partial a_{ki}}{\partial x_q} &= \frac{\sum_{l=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} a_{ki}^l \right) \sum_{l=1}^{M_i} (w_i^l) - \sum_{l=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} \right) \sum_{l=1}^{M_i} (w_i^l a_{ki}^l)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2} = \\ &= \frac{\sum_{l=1}^{M_i} \sum_{p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} a_{ki}^l w_i^p \right) - \sum_{l=1}^{M_i} \sum_{p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} a_{ki}^p w_i^p \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2}, \end{aligned} \quad (4.14)$$

que simplificando, se obtiene:

$$\frac{\partial a_{ki}}{\partial x_q} = \frac{\sum_{l=1}^{M_i} \sum_{p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} w_i^p (a_{ki}^l - a_{ki}^p) \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2}. \quad (4.15)$$

Como los dos sumatorios de la expresión anterior tienen los mismos límites, se puede simplificar su notación por:

$$\frac{\partial a_{ki}}{\partial x_q} = \frac{\sum_{l,p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} w_i^p (a_{ki}^l - a_{ki}^p) \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2}. \quad (4.16)$$

Por lo tanto, la primera derivada de (4.8) se obtiene como:

$$\frac{\partial \left(\sum_{k=0}^n a_{ki} \tilde{x}_k \right)}{\partial x_q} = a_{qi} + \sum_{k=0}^n \left(\frac{\sum_{l,p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} w_i^p (a_{ki}^l - a_{ki}^p) \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2} \right) \tilde{x}_k \quad (4.17)$$

Nótese que en la expresión anterior falta por calcular la derivada del grado de activación de las reglas del modelo borroso de la planta con respecto a cada una de las variables de estado. Este punto se resolverá una vez se hayan calculado todas las derivadas de la ecuación (4.8).

La segunda de las derivadas a calcular es:

$$\frac{\partial \left(\sum_{j=1}^m b_{ji} u_j \right)}{\partial x_q} = \sum_{j=1}^m \frac{\partial (b_{ji} u_j)}{\partial x_q} = \sum_{j=1}^m \frac{\partial b_{ji}}{\partial x_q} u_j + \sum_{j=1}^m b_{ji} \frac{\partial u_j}{\partial x_q}. \quad (4.18)$$

En un sistema en lazo abierto las señales de control no son dependientes de las variables de estado, por lo que $\partial u_j / \partial x_q = 0$. De esta forma:

$$\frac{\partial \left(\sum_{j=1}^m b_{ji} u_j \right)}{\partial x_q} = \sum_{j=1}^m \frac{\partial b_{ji}}{\partial x_q} u_j. \quad (4.19)$$

Teniendo en cuenta el valor de b_{ji} dado por la segunda ecuación de (3.18):

$$\frac{\partial b_{ji}}{\partial x_q} = \frac{\partial}{\partial x_q} \left(\frac{\sum_{l=1}^{M_i} w_i^l b_{ji}^l}{\sum_{l=1}^{M_i} w_i^l} \right) = \frac{\frac{\partial \left(\sum_{l=1}^{M_i} w_i^l b_{ji}^l \right)}{\partial x_q} \sum_{l=1}^{M_i} (w_i^l) - \frac{\partial \left(\sum_{l=1}^{M_i} w_i^l \right)}{\partial x_q} \sum_{l=1}^{M_i} (w_i^l b_{ji}^l)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2}. \quad (4.20)$$

Operando de forma similar a como se hizo con la derivada anterior, se puede obtener fácilmente que:

$$\frac{\partial b_{ji}}{\partial x_q} = \frac{\sum_{l,p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} w_i^p (b_{ji}^l - b_{ji}^p) \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2}. \quad (4.21)$$

Por lo tanto, la segunda derivada de (4.8) se puede calcular como:

$$\frac{\partial \left(\sum_{j=1}^m b_{ji} u_j \right)}{\partial x_q} = \sum_{j=1}^m \left(\frac{\sum_{l,p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} w_i^p (b_{ji}^l - b_{ji}^p) \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2} \right) u_j. \quad (4.22)$$

La tercera derivada a calcular es:

$$\frac{\partial \left(\sum_{k=0}^n a_{ki} \tilde{x}_k \right)}{\partial u_v} = \sum_{k=0}^n \frac{\partial (a_{ki} \tilde{x}_k)}{\partial u_v} = \sum_{k=0}^n \frac{\partial a_{ki}}{\partial u_v} \tilde{x}_k + \sum_{k=0}^n a_{ki} \frac{\partial \tilde{x}_k}{\partial u_v}. \quad (4.23)$$

Puesto que no existe dependencia entre las señales de control y las variables de estado, $\partial \tilde{x}_k / \partial u_v = 0$, es decir,

$$\frac{\partial \left(\sum_{k=0}^n a_{ki} \tilde{x}_k \right)}{\partial u_v} = \sum_{k=0}^n \frac{\partial a_{ki}}{\partial u_v} \tilde{x}_k. \quad (4.24)$$

Sustituyendo a_{ki} por su valor:

$$\frac{\partial a_{ki}}{\partial u_v} = \frac{\partial}{\partial u_v} \left(\frac{\sum_{l=1}^{M_i} w_i^l a_{ki}^l}{\sum_{l=1}^{M_i} w_i^l} \right) = \frac{\frac{\partial \left(\sum_{l=1}^{M_i} w_i^l a_{ki}^l \right)}{\partial u_v} \sum_{l=1}^{M_i} (w_i^l) - \frac{\partial \left(\sum_{l=1}^{M_i} w_i^l \right)}{\partial u_v} \sum_{l=1}^{M_i} (w_i^l a_{ki}^l)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2}. \quad (4.25)$$

Puesto que el grado de activación de las reglas de la planta es dependiente del vector de control, y los parámetros a_{ki}^l son constantes, se tiene que:

$$\frac{\partial \left(\sum_{l=1}^{M_i} w_i^l a_{ki}^l \right)}{\partial u_v} = \sum_{l=1}^{M_i} \left(\frac{\partial w_i^l}{\partial u_v} a_{ki}^l \right), \quad (4.26)$$

y

$$\frac{\partial \left(\sum_{l=1}^{M_i} w_i^l \right)}{\partial u_v} = \sum_{l=1}^{M_i} \left(\frac{\partial w_i^l}{\partial u_v} \right), \quad (4.27)$$

de forma que

$$\begin{aligned} \frac{\partial a_{ki}}{\partial u_v} &= \frac{\sum_{l=1}^{M_i} \left(\frac{\partial w_i^l}{\partial u_v} a_{ki}^l \right) \sum_{l=1}^{M_i} (w_i^l) - \sum_{l=1}^{M_i} \left(\frac{\partial w_i^l}{\partial u_v} \right) \sum_{l=1}^{M_i} (w_i^l a_{ki}^l)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2} = \\ &= \frac{\sum_{l=1}^{M_i} \sum_{p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial u_v} a_{ki}^l w_i^p \right) - \sum_{l=1}^{M_i} \sum_{p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial u_v} a_{ki}^p w_i^p \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2}. \end{aligned} \quad (4.28)$$

que simplificando, se obtiene:

$$\frac{\partial a_{ki}}{\partial u_v} = \frac{\sum_{l,p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial u_v} w_i^p (a_{ki}^l - a_{ki}^p) \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2}. \quad (4.29)$$

Por lo tanto, la tercera de las derivadas de la ecuación (4.8) se obtiene como:

$$\frac{\partial \left(\sum_{k=0}^n a_{ki} \tilde{x}_k \right)}{\partial u_v} = \sum_{k=0}^n \left(\frac{\sum_{l,p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial u_v} w_i^p (a_{ki}^l - a_{ki}^p) \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2} \right) \tilde{x}_k. \quad (4.30)$$

En la expresión anterior falta por calcular la derivada del grado de activación de las reglas de la planta con respecto a las señales de control. Este punto se resolverá una vez se hayan calculado todas las derivadas de la ecuación (4.8).

Por último, la cuarta derivada de la ecuación (4.8) se obtiene mediante la expresión:

$$\frac{\partial \left(\sum_{j=1}^m b_{ji} u_j \right)}{\partial u_v} = \sum_{j=1}^m \frac{\partial (b_{ji} u_j)}{\partial u_v} = \sum_{j=1}^m \frac{\partial b_{ji}}{\partial u_v} u_j + \sum_{j=1}^m b_{ji} \frac{\partial u_j}{\partial u_v}. \quad (4.31)$$

Puesto que $\frac{\partial u_j}{\partial u_v} = 1$ si $j = v$ y 0 en caso contrario:

$$\sum_{j=1}^m b_{ji} \frac{\partial u_j}{\partial u_v} = b_{vi}. \quad (4.32)$$

Teniendo en cuenta el valor de b_{ji} dado por la segunda ecuación de (3.18):

$$\frac{\partial b_{ji}}{\partial u_v} = \frac{\partial}{\partial u_v} \left(\frac{\sum_{l=1}^{M_i} w_i^l b_{ji}^l}{\sum_{l=1}^{M_i} w_i^l} \right) = \frac{\frac{\partial \left(\sum_{l=1}^{M_i} w_i^l b_{ji}^l \right)}{\partial u_v} \sum_{l=1}^{M_i} (w_i^l) - \frac{\partial \left(\sum_{l=1}^{M_i} w_i^l \right)}{\partial u_v} \sum_{l=1}^{M_i} (w_i^l b_{ji}^l)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2}. \quad (4.33)$$

Operando como con las derivadas anteriores:

$$\frac{\partial b_{ji}}{\partial u_v} = \frac{\sum_{l,p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial u_v} w_i^p (b_{ji}^l - b_{ji}^p) \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2}, \quad (4.34)$$

por lo que la última de las derivadas de la ecuación (4.8) se puede expresar como:

$$\frac{\partial \left(\sum_{j=1}^m b_{ji} u_j \right)}{\partial u_v} = b_{vi} + \sum_{j=1}^m \left(\frac{\sum_{l,p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial u_v} w_i^p (b_{ji}^l - b_{ji}^p) \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2} \right) u_j. \quad (4.35)$$

Sustituyendo las expresiones (4.17), (4.22), (4.30) y (4.35) en (4.8), el modelo matemático equivalente de un sistema borroso linealizado en torno a un punto $(\mathbf{x}_0, \mathbf{u}_0)$ viene dado por:

$$\begin{aligned} \dot{x}_i &\approx f_i(\mathbf{x}_0, \mathbf{u}_0) + \\ &+ \sum_{q=1}^n \left[\begin{aligned} &a_{qi} + \sum_{k=0}^n \left(\frac{\sum_{l,p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} w_i^p (a_{ki}^l - a_{ki}^p) \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2} \right) \tilde{x}_k \\ &+ \sum_{j=1}^m \left(\frac{\sum_{l,p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} w_i^p (b_{ji}^l - b_{ji}^p) \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2} \right) u_j \end{aligned} \right] \Bigg|_{(\mathbf{x}_0, \mathbf{u}_0)} x_q + \\ &+ \sum_{v=1}^m \left[\begin{aligned} &\sum_{k=0}^n \left(\frac{\sum_{l,p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial u_v} w_i^p (a_{ki}^l - a_{ki}^p) \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2} \right) \tilde{x}_k \\ &b_{vi} + \sum_{j=1}^m \left(\frac{\sum_{l,p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial u_v} w_i^p (b_{ji}^l - b_{ji}^p) \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2} \right) u_j \end{aligned} \right] \Bigg|_{(\mathbf{x}_0, \mathbf{u}_0)} u_v. \end{aligned} \quad (4.36)$$

Para facilitar la representación de esta ecuación de una forma más compacta puede escribirse como:

$$\begin{aligned} \dot{x}_i &\approx f_i(\mathbf{x}_0, \mathbf{u}_0) + \\ &+ \sum_{q=1}^n \left(a_{qi} + \sum_{k=0}^n \Gamma_{x_q}^{\mathcal{A}_{ki}} \tilde{x}_k + \sum_{j=1}^m \Gamma_{x_q}^{\mathcal{B}_{ji}} u_j \right) \Bigg|_{(\mathbf{x}_0, \mathbf{u}_0)} x_q + \\ &+ \sum_{v=1}^m \left(b_{vi} + \sum_{k=0}^n \Gamma_{u_v}^{\mathcal{A}_{ki}} \tilde{x}_k + \sum_{j=1}^m \Gamma_{u_v}^{\mathcal{B}_{ji}} u_j \right) \Bigg|_{(\mathbf{x}_0, \mathbf{u}_0)} u_v. \end{aligned} \quad (4.37)$$

donde

$$\Gamma_{x_q}^{\mathcal{A}_{ki}} = \frac{\sum_{l,p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} w_i^p (a_{ki}^l - a_{ki}^p) \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2}, \quad (4.38)$$

$$\Gamma_{x_q}^{\mathcal{B}_{ji}} = \frac{\sum_{l,p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} w_i^p (b_{ji}^l - b_{ji}^p) \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2}, \quad (4.39)$$

$$\Gamma_{u_v}^{\mathcal{A}_{ki}} = \frac{\sum_{l,p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial u_v} w_i^p (a_{ki}^l - a_{ki}^p) \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2}, \quad (4.40)$$

y

$$\Gamma_{u_v}^{\mathcal{B}_{ji}} = \frac{\sum_{l,p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial u_v} w_i^p (b_{ji}^l - b_{ji}^p) \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2}. \quad (4.41)$$

Derivada del grado de activación de las reglas de la planta

El grado de activación de las reglas de la planta es dependiente del vector de estado y del vector de control, según la expresión (3.15); de forma que:

$$\begin{aligned} \frac{\partial w_i^l(\mathbf{x}, \mathbf{u})}{\partial x_q} &= \frac{\partial \left(\prod_{k=1}^n \mu_{ki}^l(x_k, \boldsymbol{\sigma}_{ki}^l) \right)}{\partial x_q} \prod_{j=1}^m \mu_{ji}^l(u_j, \boldsymbol{\alpha}_{ji}^l) + \\ &+ \prod_{k=1}^n \mu_{ki}^l(x_k, \boldsymbol{\sigma}_{ki}^l) \frac{\partial \left(\prod_{j=1}^m \mu_{ji}^l(u_j, \boldsymbol{\alpha}_{ji}^l) \right)}{\partial x_q} \end{aligned} \quad (4.42)$$

$$\begin{aligned} \frac{\partial w_i^l(\mathbf{x}, \mathbf{u})}{\partial u_v} &= \frac{\partial \left(\prod_{k=1}^n \mu_{ki}^l(x_k, \boldsymbol{\sigma}_{ki}^l) \right)}{\partial u_v} \prod_{j=1}^m \mu_{ji}^l(u_j, \boldsymbol{\alpha}_{ji}^l) + \\ &+ \prod_{k=1}^n \mu_{ki}^l(x_k, \boldsymbol{\sigma}_{ki}^l) \frac{\partial \left(\prod_{j=1}^m \mu_{ji}^l(u_j, \boldsymbol{\alpha}_{ji}^l) \right)}{\partial u_v} \end{aligned} \quad (4.43)$$

En un sistema en lazo abierto las señales de control son independientes del vector de estado, por lo tanto:

$$\frac{\partial \left(\prod_{j=1}^m \mu_{ji}^l(u_j, \boldsymbol{\alpha}_{ji}^l) \right)}{\partial x_q} = 0 \quad (4.44)$$

y

$$\frac{\partial \left(\prod_{k=1}^n \mu_{ki}^l(x_k, \sigma_{ki}^l) \right)}{\partial u_v} = 0. \quad (4.45)$$

De esta forma,

$$\frac{\partial w_i^l(\mathbf{x}, \mathbf{u})}{\partial x_q} = \frac{\partial \left(\prod_{k=1}^n \mu_{ki}^l(x_k, \sigma_{ki}^l) \right)}{\partial x_q} \prod_{j=1}^m \mu_{ji}^l(u_j, \alpha_{ji}^l) \quad (4.46)$$

y

$$\frac{\partial w_i^l(\mathbf{x}, \mathbf{u})}{\partial u_v} = \prod_{k=1}^n \mu_{ki}^l(x_k, \sigma_{ki}^l) \frac{\partial \left(\prod_{j=1}^m \mu_{ji}^l(u_j, \alpha_{ji}^l) \right)}{\partial u_v}. \quad (4.47)$$

Puesto que $\frac{\partial x_k}{\partial x_q} = 1$ si $k = q$ y 0 en caso contrario:

$$\frac{\partial w_i^l(\mathbf{x}, \mathbf{u})}{\partial x_q} = \frac{\partial \mu_{qi}^l(x_q, \sigma_{qi}^l)}{\partial x_q} \prod_{k=1, k \neq q}^n \mu_{ki}^l(x_k, \sigma_{ki}^l) \prod_{j=1}^m \mu_{ji}^l(u_j, \alpha_{ji}^l). \quad (4.48)$$

De la misma forma, puesto que $\frac{\partial u_j}{\partial u_v} = 1$ si $j = v$ y 0 en caso contrario:

$$\frac{\partial w_i^l(\mathbf{x}, \mathbf{u})}{\partial u_v} = \frac{\partial \mu_{vi}^l(u_v, \alpha_{vi}^l)}{\partial u_v} \prod_{k=1}^n \mu_{ki}^l(x_k, \sigma_{ki}^l) \prod_{j=1, j \neq v}^m \mu_{ji}^l(u_j, \alpha_{ji}^l). \quad (4.49)$$

Nota. Las derivadas de las funciones de pertenencia se pueden obtener directamente de la definición de la propia función. En el apéndice B se presentan las derivadas de las principales funciones de pertenencia.

4.3. Matriz jacobiana de un sistema de control borroso

Al igual que sucede con el modelo de la planta, es interesante disponer del modelo linealizado de un sistema de control en lazo cerrado. Esta información puede ser útil para el diseño de controladores mediante técnicas de linealización, aunque como se comentó anteriormente no suele ser una técnica muy recomendable.

En un sistema en lazo cerrado, las señales de control pueden ser calculadas por el controlador a partir de las variables de estado del sistema. De esta forma, el vector de control es dependiente del vector de estado, y la representación lineal del sistema viene determinada por su matriz jacobiana.

El modelo de estado de un sistema de control no lineal, multivariable y autónomo completamente general, puede representarse mediante el siguiente sistema de ecuaciones:

$$\begin{aligned}
\dot{x}_1(t) &= \chi_1(x_1(t), x_2(t), \dots, x_n(t)) \\
\dot{x}_2(t) &= \chi_2(x_1(t), x_2(t), \dots, x_n(t)) \\
&\vdots \\
\dot{x}_n(t) &= \chi_n(x_1(t), x_2(t), \dots, x_n(t)).
\end{aligned} \tag{4.50}$$

El desarrollo en series de Taylor del sistema (4.50) en torno a un estado $\mathbf{x}^0 = (x_1^0, x_2^0, \dots, x_n^0)^T$, es:

$$\frac{d(\mathbf{x} - \mathbf{x}^0)}{dt} = \begin{pmatrix} (x_1 - x_1^0) \frac{\partial \chi_1(\mathbf{x})}{\partial x_1} \Big|_{\mathbf{x}^0} + \dots + (x_n - x_n^0) \frac{\partial \chi_1(\mathbf{x})}{\partial x_n} \Big|_{\mathbf{x}^0} + R_1(\mathbf{x}^0) \\ (x_1 - x_1^0) \frac{\partial \chi_2(\mathbf{x})}{\partial x_1} \Big|_{\mathbf{x}^0} + \dots + (x_n - x_n^0) \frac{\partial \chi_2(\mathbf{x})}{\partial x_n} \Big|_{\mathbf{x}^0} + R_2(\mathbf{x}^0) \\ \dots + \dots + \dots \\ (x_1 - x_1^0) \frac{\partial \chi_n(\mathbf{x})}{\partial x_1} \Big|_{\mathbf{x}^0} + \dots + (x_n - x_n^0) \frac{\partial \chi_n(\mathbf{x})}{\partial x_n} \Big|_{\mathbf{x}^0} + R_n(\mathbf{x}^0) \end{pmatrix}. \tag{4.51}$$

Dado un estado de equilibrio $\mathbf{x}^e$ del sistema (4.50), si los términos no lineales $R_i(\mathbf{x}^e)$, $i = 1..n$, de (4.51) satisfacen la condición (4.52), siendo K_i una constante lo suficientemente pequeña, entonces la matriz jacobiana (4.53) del sistema (4.50) determina las propiedades de estabilidad del estado de equilibrio $\mathbf{x}^e$.

$$|R_i(\mathbf{x}^e)| \leq K_i ((x_1 - x_1^e)^2 + \dots + (x_n - x_n^e)^2) \tag{4.52}$$

Como se puede observar, para el calculo de la matriz jacobiana, (4.53), es necesario obtener las derivadas parciales de cada una de las i -ésimas ecuaciones de estado con respecto a cada variable de estado.

$$\mathbf{J}(\mathbf{x}^e) = \begin{pmatrix} \frac{\partial \chi_1(\mathbf{x}^e)}{\partial x_1} & \frac{\partial \chi_1(\mathbf{x}^e)}{\partial x_2} & \dots & \frac{\partial \chi_1(\mathbf{x}^e)}{\partial x_n} \\ \frac{\partial \chi_2(\mathbf{x}^e)}{\partial x_1} & \frac{\partial \chi_2(\mathbf{x}^e)}{\partial x_2} & \dots & \frac{\partial \chi_2(\mathbf{x}^e)}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \chi_n(\mathbf{x}^e)}{\partial x_1} & \frac{\partial \chi_n(\mathbf{x}^e)}{\partial x_2} & \dots & \frac{\partial \chi_n(\mathbf{x}^e)}{\partial x_n} \end{pmatrix} \tag{4.53}$$

La estabilidad de un estado de equilibrio viene determinada por la estabilidad del sistema linealizado en dicho estado [224], es decir, por el valor de los autovalores de la matriz jacobiana calculada en el estado de equilibrio objeto de estudio. Sin embargo, mucho más interesante que conocer la estabilidad de un sistema, puede ser diseñar un controlador que la garantice. Para este fin, en la teoría de sistemas no lineales existen muchos métodos que se basan en la matriz jacobiana del sistema, como el segundo método de Lyapunov [220],

el método de Krasovskii [224], o los métodos basados en índices de estabilidad y robustez de sistemas [100, 292]. Por lo tanto, la obtención de la matriz jacobiana de un sistema de control borroso genérico abre las puertas a la aplicación de estas técnicas, ampliamente conocidas y aceptadas por la comunidad científica, a los sistemas borrosos [233, 293, 294].

Si el sistema de control en lazo cerrado (4.50) representa un sistema de control borroso, donde tanto la planta como el controlador son modelos borrosos de tipo TSK con motor de inferencia producto y agregación por promedio ponderado, la ecuación (4.50) puede sustituirse por el modelo matemático equivalente de dicho sistema, obtenido en la sección 3.2.1. A continuación se reescribe dicha expresión por comodidad:

$$\dot{x}_i = \sum_{k=0}^n \left(a_{ki} + \sum_{j=1}^m b_{ji} c_{kj} \right) \tilde{x}_k. \quad (3.55)$$

Según (4.53), la obtención de la matriz jacobiana del sistema borroso requerirá calcular las derivadas parciales para las $i = 1..n$ ecuaciones de estado de (3.55), con respecto a cada una de las $q = 1..n$ variables de estado del sistema:

$$J_{iq}(\mathbf{x}) = \frac{\partial \chi_i(\mathbf{x})}{\partial x_q} = \frac{\partial}{\partial x_q} \left(\sum_{k=0}^n \left(a_{ki} + \sum_{j=1}^m b_{ji} c_{kj} \right) \tilde{x}_k \right). \quad (4.54)$$

Separando los sumatorios en la ecuación anterior:

$$J_{iq}(\mathbf{x}) = \frac{\partial \left(\sum_{k=0}^n a_{ki} \tilde{x}_k \right)}{\partial x_q} + \frac{\partial \left(\sum_{k=0}^n \sum_{j=1}^m b_{ji} c_{kj} \tilde{x}_k \right)}{\partial x_q}, \quad (4.55)$$

se puede obtener la derivada de cada uno de los sumandos por separado.

Derivada del primer sumando

La derivada del primer sumando de (4.55) se obtiene de la forma:

$$\frac{\partial \left(\sum_{k=0}^n a_{ki} \tilde{x}_k \right)}{\partial x_q} = \sum_{k=0}^n \frac{\partial (a_{ki} \tilde{x}_k)}{\partial x_q} = \sum_{k=0}^n \frac{\partial a_{ki}}{\partial x_q} \tilde{x}_k + \sum_{k=0}^n a_{ki} \frac{\partial \tilde{x}_k}{\partial x_q}. \quad (4.56)$$

Puesto que $\frac{\partial \tilde{x}_k}{\partial x_q} = 1$ si $k = q$ y 0 en caso contrario:

$$\sum_{k=0}^n a_{ki} \frac{\partial \tilde{x}_k}{\partial x_q} = a_{qi}. \quad (4.57)$$

Respecto del primer sumando de (4.56), teniendo en cuenta el valor de a_{ki} dado por la primera ecuación de (3.18):

$$\frac{\partial a_{ki}}{\partial x_q} = \frac{\partial}{\partial x_q} \left(\frac{\sum_{l=1}^{M_i} w_i^l a_{ki}^l}{\sum_{l=1}^{M_i} w_i^l} \right) = \frac{\frac{\partial \left(\sum_{l=1}^{M_i} w_i^l a_{ki}^l \right)}{\partial x_q} \sum_{l=1}^{M_i} (w_i^l) - \frac{\partial \left(\sum_{l=1}^{M_i} w_i^l \right)}{\partial x_q} \sum_{l=1}^{M_i} (w_i^l a_{ki}^l)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2}. \quad (4.58)$$

El grado de activación de las reglas de la planta, w_i^l , es un valor dependiente del vector de estado, mientras que los elementos a_{ki}^l son los coeficientes constantes del consecuente de las reglas de la planta. De esta forma, se tiene que:

$$\frac{\partial \left(\sum_{l=1}^{M_i} w_i^l a_{ki}^l \right)}{\partial x_q} = \sum_{l=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} a_{ki}^l \right), \quad (4.59)$$

y

$$\frac{\partial \left(\sum_{l=1}^{M_i} w_i^l \right)}{\partial x_q} = \sum_{l=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} \right). \quad (4.60)$$

Reemplazando las expresiones anteriores en la ecuación (4.58),

$$\begin{aligned} \frac{\partial a_{ki}}{\partial x_q} &= \frac{\sum_{l=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} a_{ki}^l \right) \sum_{l=1}^{M_i} (w_i^l) - \sum_{l=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} \right) \sum_{l=1}^{M_i} (w_i^l a_{ki}^l)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2} = \\ &= \frac{\sum_{l=1}^{M_i} \sum_{p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} a_{ki}^l w_i^p \right) - \sum_{l=1}^{M_i} \sum_{p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} a_{ki}^p w_i^p \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2}, \end{aligned} \quad (4.61)$$

y simplificando, se obtiene:

$$\frac{\partial a_{ki}}{\partial x_q} = \frac{\sum_{l=1}^{M_i} \sum_{p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} w_i^p (a_{ki}^l - a_{ki}^p) \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2}. \quad (4.62)$$

Como los dos sumatorios de la expresión anterior tienen los mismos límites, se puede simplificar su notación por la siguiente:

$$\frac{\partial a_{ki}}{\partial x_q} = \frac{\sum_{l,p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} w_i^p (a_{ki}^l - a_{ki}^p) \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2}. \quad (4.63)$$

Una vez realizados los cálculos intermedios pertinentes es posible reescribir la ecuación (4.56), obteniéndose la expresión definitiva de la derivada del primer sumando de la jacobiana del sistema de control borroso:

$$\frac{\partial \left(\sum_{k=0}^n a_{ki} \tilde{x}_k \right)}{\partial x_q} = \sum_{k=0}^n \left(\frac{\sum_{l,p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} w_i^p (a_{ki}^l - a_{ki}^p) \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2} \tilde{x}_k \right) + a_{qi}. \quad (4.64)$$

Nótese que para evaluar la expresión (4.64) es necesario calcular previamente la derivada del grado de activación de las reglas de la planta con respecto a cada una de las variables de estado, $\partial w_i^l / \partial x_q$. Este punto se resuelve posteriormente (ver página 117).

Derivada del segundo sumando

La derivada del segundo sumando de la ecuación (4.55) puede reescribirse como:

$$\frac{\partial \left(\sum_{k=0}^n \sum_{j=1}^m b_{ji} c_{kj} \tilde{x}_k \right)}{\partial x_q} = \sum_{k=0}^n \frac{\partial \left(\sum_{j=1}^m b_{ji} c_{kj} \tilde{x}_k \right)}{\partial x_q}, \quad (4.65)$$

que a su vez puede descomponerse en:

$$\sum_{k=0}^n \frac{\partial \left(\sum_{j=1}^m b_{ji} c_{kj} \tilde{x}_k \right)}{\partial x_q} = \sum_{k=0}^n \frac{\partial \left(\sum_{j=1}^m b_{ji} c_{kj} \right)}{\partial x_q} \tilde{x}_k + \sum_{k=0}^n \left(\frac{\partial \tilde{x}_k}{\partial x_q} \sum_{j=1}^m b_{ji} c_{kj} \right), \quad (4.66)$$

pero $\frac{\partial \tilde{x}_k}{\partial x_q} = 1$ si $k = q$ y 0 en caso contrario, por tanto:

$$\sum_{k=0}^n \left(\frac{\partial \tilde{x}_k}{\partial x_q} \sum_{j=1}^m b_{ji} c_{kj} \right) = \sum_{j=1}^m b_{ji} c_{qj}. \quad (4.67)$$

La derivada del primer sumando de (4.66) puede ser desarrollada como:

$$\sum_{k=0}^n \frac{\partial \left(\sum_{j=1}^m b_{ji} c_{kj} \right)}{\partial x_q} \tilde{x}_k = \sum_{k=0}^n \sum_{j=1}^m \left(\frac{\partial b_{ji}}{\partial x_q} c_{kj} + \frac{\partial c_{kj}}{\partial x_q} b_{ji} \right) \tilde{x}_k, \quad (4.68)$$

donde el término b_{ji} es un coeficiente variable que se calcula a partir de la segunda ecuación de (3.18) y c_{kj} es el coeficiente variable correspondiente al controlador borroso que se calcula mediante (3.40).

Dada la gran similitud de las expresiones para el cálculo de los coeficientes variables a_{ki} y b_{ji} , es fácil deducir que:

$$\frac{\partial b_{ji}}{\partial x_q} = \frac{\sum_{l,p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} w_i^p (b_{ji}^l - b_{ji}^p) \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2}. \quad (4.69)$$

Por otro lado, operando de forma similar se puede obtener la derivada de c_{kj} respecto de las variables de estado:

$$\frac{\partial c_{kj}}{\partial x_q} = \frac{\partial}{\partial x_q} \left(\frac{\sum_{r=1}^{N_j} \omega_j^r c_{kj}^r}{\sum_{r=1}^{N_j} \omega_j^r} \right) = \frac{\frac{\partial \left(\sum_{r=1}^{N_j} \omega_j^r c_{kj}^r \right)}{\partial x_q} \sum_{r=1}^{N_j} (\omega_j^r) - \frac{\partial \left(\sum_{r=1}^{N_j} \omega_j^r \right)}{\partial x_q} \sum_{r=1}^{N_j} (\omega_j^r c_{kj}^r)}{\left(\sum_{r=1}^{N_j} \omega_j^r \right)^2}. \quad (4.70)$$

El grado de activación de las reglas del controlador, ω_j^r , es un valor dependiente del vector de estado, mientras que los elementos c_{kj}^r son los coeficientes constantes del consecuente de las reglas de éste. Por lo tanto:

$$\frac{\partial \left(\sum_{r=1}^{N_j} \omega_j^r c_{kj}^r \right)}{\partial x_q} = \sum_{r=1}^{N_j} \left(\frac{\partial \omega_j^r}{\partial x_q} c_{kj}^r \right), \quad (4.71)$$

y

$$\frac{\partial \left(\sum_{r=1}^{N_j} \omega_j^r \right)}{\partial x_q} = \sum_{r=1}^{N_j} \left(\frac{\partial \omega_j^r}{\partial x_q} \right). \quad (4.72)$$

Reemplazando las expresiones anteriores en la ecuación (4.70),

$$\begin{aligned}
\frac{\partial c_{kj}}{\partial x_q} &= \frac{\sum_{r=1}^{N_j} \left(\frac{\partial \omega_j^r}{\partial x_q} c_{kj}^r \right) \sum_{r=1}^{N_j} (\omega_j^r) - \sum_{r=1}^{N_j} \left(\frac{\partial \omega_j^r}{\partial x_q} \right) \sum_{r=1}^{N_j} (\omega_j^r c_{kj}^r)}{\left(\sum_{r=1}^{N_j} \omega_j^r \right)^2} = \\
&= \frac{\sum_{r=1}^{N_j} \sum_{s=1}^{N_j} \left(\frac{\partial \omega_j^r}{\partial x_q} c_{kj}^r \omega_j^s \right) - \sum_{r=1}^{M_i} \sum_{s=1}^{M_i} \left(\frac{\partial \omega_j^r}{\partial x_q} c_{kj}^s \omega_j^s \right)}{\left(\sum_{r=1}^{N_j} \omega_j^r \right)^2},
\end{aligned} \tag{4.73}$$

que simplificando y agrupando los sumatorios, puede escribirse como:

$$\frac{\partial c_{kj}}{\partial x_q} = \frac{\sum_{r,s=1}^{N_j} \left(\frac{\partial \omega_j^r}{\partial x_q} \omega_j^s (c_{kj}^r - c_{kj}^s) \right)}{\left(\sum_{r=1}^{N_j} \omega_j^r \right)^2}. \tag{4.74}$$

Sustituyendo (4.69) y (4.74) en (4.68):

$$\sum_{k=0}^n \frac{\partial \left(\sum_{j=1}^m b_{ji} c_{kj} \right)}{\partial x_q} \tilde{x}_k = \sum_{k=0}^n \sum_{j=1}^m \left(\frac{\sum_{l,p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} w_i^p (b_{ji}^l - b_{ji}^p) \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2} c_{kj} + \right. \\
\left. + \frac{\sum_{r,s=1}^{N_j} \left(\frac{\partial \omega_j^r}{\partial x_q} \omega_j^s (c_{kj}^r - c_{kj}^s) \right)}{\left(\sum_{r=1}^{N_j} \omega_j^r \right)^2} b_{ji} \right) \tilde{x}_k. \tag{4.75}$$

Una vez realizados todos los cálculos intermedios, puede sustituirse (4.67) y (4.75) en la expresión (4.66), con lo que el segundo sumando de la jacobiana del sistema de control borroso puede expresarse por:

$$\begin{aligned}
\frac{\partial \left(\sum_{k=0}^n \sum_{j=1}^m b_{ji} c_{kj} \tilde{x}_k \right)}{\partial x_q} &= \sum_{j=1}^m b_{ji} c_{qj} + \\
&+ \sum_{k=0}^n \sum_{j=1}^m \left(\frac{\sum_{l,p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} w_i^p (b_{ji}^l - b_{ji}^p) \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2} c_{kj} + \frac{\sum_{r,s=1}^{N_j} \left(\frac{\partial \omega_j^r}{\partial x_q} \omega_j^s (c_{kj}^r - c_{kj}^s) \right)}{\left(\sum_{r=1}^{N_j} \omega_j^r \right)^2} b_{ji} \right) \tilde{x}_k.
\end{aligned} \quad (4.76)$$

Nótese que para evaluar la expresión (4.76) es necesario calcular previamente tanto la derivada del grado de activación de las reglas de la planta, como la derivada del grado de activación de las reglas del controlador, con respecto a cada una de las variables de estado. Estas derivadas, $\partial w_i^l / \partial x_q$ y $\partial \omega_j^r / \partial x_q$ respectivamente, se resuelven a continuación.

Derivada del grado de activación de las reglas de la planta

El grado de activación de las reglas de la planta es dependiente del vector de estado y del vector de control, según la expresión (3.15), que se repite a continuación por comodidad. Si se tiene un sistema de control en lazo cerrado, las señales de control son generadas a partir del valor del vector de estado, en el que el vector $\mathbf{u}$ es dependiente del vector $\mathbf{x}$.

$$w_i^l(\mathbf{x}, \mathbf{u}) = \prod_{k=1}^n \mu_{ki}^l(x_k, \sigma_{ki}^l) \prod_{j=1}^m \mu_{ji}^l(u_j, \alpha_{ji}^l) \quad (3.15)$$

Se puede representar el grado de activación de las reglas de la planta, $w_i^l(\mathbf{x}, \mathbf{u})$, como el producto de una componente dependiente del vector de estado, $w_i^l(\mathbf{x})$, y otra componente dependiente del vector de control, $w_i^l(\mathbf{u})$:

$$w_i^l(\mathbf{x}, \mathbf{u}) = \underbrace{\prod_{k=1}^n \mu_{ki}^l(x_k, \sigma_{ki}^l)}_{w_i^l(\mathbf{x})} \underbrace{\prod_{j=1}^m \mu_{ji}^l(u_j, \alpha_{ji}^l)}_{w_i^l(\mathbf{u})}, \quad (4.77)$$

con lo que su derivada respecto de las variables de estado se pueden calcular como:

$$\frac{\partial w_i^l}{\partial x_q} = \frac{\partial}{\partial x_q} \left(w_i^l(\mathbf{x}) w_i^l(\mathbf{u}) \right) = w_i^l(\mathbf{u}) \frac{\partial w_i^l(\mathbf{x})}{\partial x_q} + w_i^l(\mathbf{x}) \frac{\partial w_i^l(\mathbf{u})}{\partial x_q}. \quad (4.78)$$

Puesto que

$$\frac{\partial w_i^l(\mathbf{x})}{\partial x_q} = \frac{\partial}{\partial x_q} \left(\prod_{k=1}^n \mu_{ki}^l(x_k, \sigma_{ki}^l) \right), \quad (4.79)$$

se obtiene,

$$\begin{aligned} \frac{\partial w_i^l(\mathbf{x})}{\partial x_q} = & \frac{\partial \mu_{1i}^l(x_1, \boldsymbol{\sigma}_{1i}^l)}{\partial x_q} \prod_{k=1, k \neq 1}^n \mu_{ki}^l(x_k, \boldsymbol{\sigma}_{ki}^l) + \frac{\partial \mu_{2i}^l(x_2, \boldsymbol{\sigma}_{2i}^l)}{\partial x_q} \prod_{k=1, k \neq 2}^n \mu_{ki}^l(x_k, \boldsymbol{\sigma}_{ki}^l) + \\ & + \dots + \frac{\partial \mu_{ni}^l(x_n, \boldsymbol{\sigma}_{ni}^l)}{\partial x_q} \prod_{k=1, k \neq n}^n \mu_{ki}^l(x_k, \boldsymbol{\sigma}_{ki}^l), \end{aligned} \quad (4.80)$$

que se puede representar de forma más compacta por:

$$\frac{\partial w_i^l(\mathbf{x})}{\partial x_q} = \sum_{p=1}^n \left(\frac{\partial \mu_{pi}^l(x_p, \boldsymbol{\sigma}_{pi}^l)}{\partial x_q} \prod_{k=1, k \neq p}^n \mu_{ki}^l(x_k, \boldsymbol{\sigma}_{ki}^l) \right). \quad (4.81)$$

Teniendo en cuenta que

$$\frac{\partial f(x_i)}{\partial x_j} = 0, \quad \forall i \neq j, \quad (4.82)$$

la expresión (4.81) puede reducirse a:

$$\frac{\partial w_i^l(\mathbf{x})}{\partial x_q} = \frac{\partial \mu_{qi}^l(x_q, \boldsymbol{\sigma}_{qi}^l)}{\partial x_q} \prod_{k=1, k \neq q}^n \mu_{ki}^l(x_k, \boldsymbol{\sigma}_{ki}^l). \quad (4.83)$$

Por otro lado, la parte dependiente del vector de control es:

$$\frac{\partial w_i^l(\mathbf{u})}{\partial x_q} = \frac{\partial}{\partial x_q} \left(\prod_{j=1}^m \mu_{ji}^l(u_j, \boldsymbol{\alpha}_{ji}^l) \right), \quad (4.84)$$

de cuyo desarrollo resulta:

$$\begin{aligned} \frac{\partial w_i^l(\mathbf{u})}{\partial x_q} = & \frac{\partial \mu_{1i}^l(u_1, \boldsymbol{\alpha}_{1i}^l)}{\partial x_q} \prod_{j=1, j \neq 1}^m \mu_{ji}^l(u_j, \boldsymbol{\alpha}_{ji}^l) + \frac{\partial \mu_{2i}^l(u_2, \boldsymbol{\alpha}_{2i}^l)}{\partial x_q} \prod_{j=1, j \neq 2}^m \mu_{ji}^l(u_j, \boldsymbol{\alpha}_{ji}^l) + \\ & + \dots + \frac{\partial \mu_{mi}^l(u_m, \boldsymbol{\alpha}_{mi}^l)}{\partial x_q} \prod_{j=1, j \neq m}^m \mu_{ji}^l(u_j, \boldsymbol{\alpha}_{ji}^l), \end{aligned} \quad (4.85)$$

o de forma más compacta:

$$\frac{\partial w_i^l(\mathbf{u})}{\partial x_q} = \sum_{v=1}^m \left(\frac{\partial \mu_{vi}^l(u_v, \boldsymbol{\alpha}_{vi}^l)}{\partial x_q} \prod_{j=1, j \neq v}^m \mu_{ji}^l(u_j, \boldsymbol{\alpha}_{ji}^l) \right). \quad (4.86)$$

En un sistema de control borroso en lazo cerrado, las señales de control son generadas por un controlador borroso y, por lo tanto, se pueden calcular a partir del modelo matemático equivalente de un controlador TSK dado por la ecuación (3.42).

$$u_j = \sum_{k=0}^n c_{kj} \tilde{x}_k, \quad (3.42)$$

Nótese que para calcular $\frac{\partial \mu_{vi}^l(u_v, \alpha_{vi}^l)}{\partial x_q}$ en (4.86) será necesario obtener $\frac{\partial u_v}{\partial x_q}$, ya que aplicando la regla de la cadena:

$$\frac{\partial \mu_{vi}^l(u_v, \alpha_{vi}^l)}{\partial x_q} = \frac{\partial \mu_{vi}^l(u_v, \alpha_{vi}^l)}{\partial u_v} \frac{\partial u_v}{\partial x_q}, \quad (4.87)$$

donde $\frac{\partial \mu_{vi}^l(u_v, \alpha_{vi}^l)}{\partial u_v}$ se puede calcular directamente de la definición de la propia función de pertenencia¹. Para resolver $\partial u_v / \partial x_q$ se parte de la ecuación (3.42), de forma que

$$\frac{\partial u_v}{\partial x_q} = \frac{\partial \left(\sum_{k=0}^n c_{kv} \tilde{x}_k \right)}{\partial x_q} = \sum_{k=0}^n \frac{\partial c_{kv}}{\partial x_q} \tilde{x}_k + \sum_{k=0}^n c_{kv} \frac{\partial \tilde{x}_k}{\partial x_q}, \quad (4.88)$$

es decir,

$$\frac{\partial u_v}{\partial x_q} = \sum_{k=0}^n \frac{\partial c_{kv}}{\partial x_q} \tilde{x}_k + c_{qv}. \quad (4.89)$$

Sustituyendo (4.74) en (4.89), se obtiene:

$$\frac{\partial u_v}{\partial x_q} = c_{qv} + \sum_{k=0}^n \left(\frac{\sum_{r,s=1}^{N_v} \left(\frac{\partial \omega_v^r}{\partial x_q} \omega_v^s (c_{kv}^r - c_{kv}^s) \right)}{\left(\sum_{r=1}^{N_v} \omega_v^r \right)^2} \right) \tilde{x}_k. \quad (4.90)$$

Finalmente, sustituyendo las ecuaciones (4.83) y (4.86) en (4.78), se obtiene la expresión que permite calcular la derivada del grado de activación de las reglas de la planta en un sistema de control borroso:

$$\begin{aligned} \frac{\partial w_i^l(\mathbf{x}, \mathbf{u})}{\partial x_q} &= \prod_{j=1}^m \mu_{ji}^l(u_j, \alpha_{ji}^l) \frac{\partial \mu_{qi}^l(x_q, \sigma_{qi}^l)}{\partial x_q} \prod_{k=1, k \neq q}^n \mu_{ki}^l(x_k, \sigma_{ki}^l) + \\ &+ \prod_{k=1}^n \mu_{ki}^l(x_k, \sigma_{ki}^l) \sum_{v=1}^m \left(\frac{\partial \mu_{vi}^l(u_v, \alpha_{vi}^l)}{\partial x_q} \prod_{j=1, j \neq v}^m \mu_{ji}^l(u_j, \alpha_{ji}^l) \right), \end{aligned} \quad (4.91)$$

donde $\frac{\partial \mu_{vi}^l(u_v, \alpha_{vi}^l)}{\partial x_q}$ se calcula aplicando (4.90) sobre (4.87).

¹En el apéndice B se presentan las derivadas de las principales funciones de pertenencia.

Derivada del grado de activación de las reglas del controlador

El grado de activación de las reglas del controlador viene dado por:

$$\omega_j^r = \prod_{k=1}^n \mu_{kj}^r(x_k, \Phi_{kj}^r), \quad (3.37)$$

de modo que:

$$\frac{\partial \omega_j^r}{\partial x_q} = \frac{\partial}{\partial x_q} \left(\prod_{k=1}^n \mu_{kj}^r(x_k, \Phi_{kj}^r) \right). \quad (4.92)$$

Operando:

$$\begin{aligned} \frac{\partial \omega_j^r}{\partial x_q} = & \frac{\partial \mu_{1j}^r(x_1, \Phi_{1j}^r)}{\partial x_q} \prod_{k=1, k \neq 1}^n \mu_{kj}^r(x_k, \Phi_{kj}^r) + \frac{\partial \mu_{2j}^r(x_2, \Phi_{2j}^r)}{\partial x_q} \prod_{k=1, k \neq 2}^n \mu_{kj}^r(x_k, \Phi_{kj}^r) + \\ & + \dots + \frac{\partial \mu_{nj}^r(x_n, \Phi_{nj}^r)}{\partial x_q} \prod_{k=1, k \neq n}^n \mu_{kj}^r(x_k, \Phi_{kj}^r). \end{aligned} \quad (4.93)$$

Finalmente, si se tiene en cuenta que

$$\frac{\partial f(x_i)}{\partial x_j} = 0, \quad \forall i \neq j, \quad (4.82)$$

la expresión (4.93) puede simplificarse a:

$$\frac{\partial \omega_j^r}{\partial x_q} = \frac{\partial \mu_{qj}^r(x_q, \Phi_{qj}^r)}{\partial x_q} \prod_{k=1, k \neq q}^n \mu_{kj}^r(x_k, \Phi_{kj}^r), \quad (4.94)$$

que representa la expresión final para el cálculo de la derivada del grado de activación de las reglas del controlador con respecto a la q -ésima variable de estado. Las derivadas de las funciones de pertenencia se pueden obtener directamente de la definición de la propia función (en el apéndice B se presentan las derivadas de las principales funciones de pertenencia).

Expresión final de la matriz jacobiana de un sistema de control borroso

Cada uno de los elementos de la matriz jacobiana del sistema, $J_{iq}(\mathbf{x})$, para $i, q = 1..n$, viene determinado por la derivada de la expresión (3.55) con respecto a cada una de las variables de estado x_q . Los dos sumandos que conforman el cálculo de los elementos de la matriz jacobiana, según la ecuación (4.55), han sido calculados, y sus resultados se muestran en las expresiones (4.64) y (4.76), respectivamente. Sustituyendo dichos resultados en la ecuación (4.55) se obtiene:

$$\begin{aligned}
J_{iq}(\mathbf{x}) = \frac{\partial \chi_i(\mathbf{x})}{\partial x_q} = a_{qi} + \sum_{k=0}^n \left(\frac{\sum_{l,p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} w_i^p (a_{ki}^l - a_{ki}^p) \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2} \tilde{x}_k \right) + \sum_{j=1}^m b_{ji} c_{qj} + \\
+ \sum_{k=0}^n \sum_{j=1}^m \left(\frac{\sum_{l,p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} w_i^p (b_{ji}^l - b_{ji}^p) \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2} c_{kj} + \frac{\sum_{r,s=1}^{N_j} \left(\frac{\partial \omega_j^r}{\partial x_q} \omega_j^s (c_{kj}^r - c_{kj}^s) \right)}{\left(\sum_{r=1}^{N_j} \omega_j^r \right)^2} b_{ji} \right) \tilde{x}_k.
\end{aligned} \tag{4.95}$$

Reagrupando los sumatorios y reordenando la expresión, los elementos de la matriz jacobiana del sistema de control borroso en lazo cerrado dado en (4.53) se pueden calcular mediante la ecuación (4.96).

$$\begin{aligned}
J_{iq} = a_{qi} + \sum_{j=1}^m b_{ji} c_{qj} + \\
+ \sum_{k=0}^n \left(\frac{\sum_{l,p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} w_i^p (a_{ki}^l - a_{ki}^p) \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2} + \right. \\
\left. + \sum_{j=1}^m \left(\frac{\sum_{l,p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} w_i^p (b_{ji}^l - b_{ji}^p) \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2} c_{kj} + \frac{\sum_{r,s=1}^{N_j} \left(\frac{\partial \omega_j^r}{\partial x_q} \omega_j^s (c_{kj}^r - c_{kj}^s) \right)}{\left(\sum_{r=1}^{N_j} \omega_j^r \right)^2} b_{ji} \right) \right) \tilde{x}_k
\end{aligned} \tag{4.96}$$

Para facilitar la representación de esta ecuación de una forma más compacta puede escribirse como:

$$J_{iq} = a_{qi} + \sum_{j=1}^m b_{ji} c_{qj} + \sum_{k=0}^n \left(\Gamma_q^{\mathcal{A}_{ki}} + \sum_{j=1}^m \left(c_{kj} \Gamma_q^{\mathcal{B}_{ji}} + b_{ji} \Gamma_q^{\mathcal{C}_{kj}} \right) \right) \tilde{x}_k, \tag{4.97}$$

siendo

$$\Gamma_q^{\mathcal{A}_{ki}} = \frac{\sum_{l,p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} w_i^p (a_{ki}^l - a_{ki}^p) \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2}, \tag{4.98}$$

$$\Gamma_q^{\mathcal{B}_{ji}} = \frac{\sum_{l,p=1}^{M_i} \left(\frac{\partial w_i^l}{\partial x_q} w_i^p (b_{ji}^l - b_{ji}^p) \right)}{\left(\sum_{l=1}^{M_i} w_i^l \right)^2} \quad (4.99)$$

y

$$\Gamma_q^{C_{kj}} = \frac{\sum_{r,s=1}^{N_j} \left(\frac{\partial \omega_j^r}{\partial x_q} \omega_j^s (c_{kj}^r - c_{kj}^s) \right)}{\left(\sum_{r=1}^{N_j} \omega_j^r \right)^2}. \quad (4.100)$$

Con la intención de no complicar las ecuaciones anteriores no se han sustituido las expresiones de las derivadas de los grados de activación de las reglas de la planta ni del controlador, dadas por las ecuaciones (4.91) y (4.94), respectivamente.

4.4. Estados de equilibrio de un sistema de control borroso

La búsqueda de los estados de equilibrio de un sistema de control es uno de los primeros problemas que se deben resolver para proceder con el estudio de su dinámica. Como análisis previo al diseño del sistema de control, la obtención de los estados de equilibrio proporciona una valiosa información sobre el comportamiento de éste, especialmente si se trata de un sistema no lineal, ya que estos puntos son candidatos ideales para el estudio de estos sistemas mediante técnicas de linealización. Una vez diseñado el controlador, la obtención de los estados de equilibrio del sistema realimentado puede ser útil para comprobar que el diseño se ha realizado adecuadamente, así como para poder aplicar análisis de estabilidad a dichos estados.

Un sistema dinámico de tiempo continuo puede ser representado de forma general por la expresión (4.101). Tal como se vio en la definición 3.4, los estados de equilibrio de dicho sistema son todos aquellos puntos, $\mathbf{x} = \mathbf{x}_{eq}$, que satisfacen la expresión $\dot{\mathbf{x}}(t) = \mathbf{0}$, siendo $\mathbf{u} = \mathbf{u}_{eq}$ un vector de control constante.

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}, \mathbf{u}) \quad (4.101)$$

El modelo matemático equivalente de un modelo borroso de estado viene dado por la ecuación (3.21), por lo que la obtención de sus estados de equilibrio requiere resolver el sistema de ecuaciones (4.102), obtenido de igualar a cero dicha expresión.

$$\begin{aligned}
\sum_{k=0}^n a_{k1} \tilde{x}_k + \sum_{j=1}^m b_{j1} u_j &= 0 \\
\sum_{k=0}^n a_{k2} \tilde{x}_k + \sum_{j=1}^m b_{j2} u_j &= 0 \\
&\vdots \\
\sum_{k=0}^n a_{kn} \tilde{x}_k + \sum_{j=1}^m b_{jn} u_j &= 0
\end{aligned} \tag{4.102}$$

Si se realimenta el sistema con un controlador borroso, el modelo matemático equivalente del sistema de lazo cerrado está dado por la ecuación (3.55), por lo que para obtener sus estados de equilibrio se deberá resolver el sistema de ecuaciones (4.103).

$$\begin{aligned}
\sum_{k=0}^n \left(a_{k1} + \sum_{j=1}^m b_{j1} c_{kj} \right) \tilde{x}_k &= 0 \\
\sum_{k=0}^n \left(a_{k2} + \sum_{j=1}^m b_{j2} c_{kj} \right) \tilde{x}_k &= 0 \\
&\vdots \\
\sum_{k=0}^n \left(a_{kn} + \sum_{j=1}^m b_{jn} c_{kj} \right) \tilde{x}_k &= 0
\end{aligned} \tag{4.103}$$

Dada la inherente capacidad de los sistemas borrosos para representar dinámicas no lineales, se hace necesaria la utilización de métodos capaces de resolver el sistema de ecuaciones no lineales que se plantea: (4.102) ó (4.103), según se pretenda resolver los estados de equilibrio de la planta o del sistema en lazo cerrado. A diferencia de los sistemas lineales, que pueden tener uno o infinitos estados de equilibrio, un sistema no lineal puede tener cualquier número de éstos. Este hecho, junto con la propia complejidad de las ecuaciones que definen los sistemas no lineales, provoca que el cálculo de los estados de equilibrio de los sistemas borrosos puede convertirse en un problema de envergadura.

Para el cálculo de los estados de equilibrio se van a utilizar métodos numéricos clásicos derivados del método de Newton [295, 296], como son el método *Line-Search* [297] y el de Región de Confianza [298, 299].

4.4.1. Métodos numéricos para la minimización de funciones

Los métodos de Newton proporcionan una importante clase de algoritmos de optimización local. En su forma básica, la siguiente iteración se obtiene en base al vector gradiente, $\mathbf{G}(x_1 \dots x_n)$, y la matriz hessiana, $\mathbf{H}(x_1, \dots, x_n)$. La siguiente iteración, $k+1$, se obtiene desde la minimización de $r_{\mathbf{x}_k}(\mathbf{d}_k)$, siendo

$$r_{\mathbf{x}_k}(\mathbf{d}_k) = r(\mathbf{x}_k) + \mathbf{G}(\mathbf{x}_k)^T \mathbf{d}_k + \mathbf{d}_k^T \mathbf{H}(\mathbf{x}_k) \mathbf{d}_k / 2 \tag{4.104}$$

un modelo cuadrático de la función objetivo en torno al valor de $\mathbf{x}$ en la k -ésima iteración, de forma que $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}_k$. Cuando la matriz hessiana es definida positiva, el modelo

cuadrático tiene una única solución que puede obtenerse mediante la resolución del sistema simétrico lineal

$$\mathbf{H}(\mathbf{x}_k)\mathbf{d}_k = -\mathbf{G}(\mathbf{x}_k). \quad (4.105)$$

La convergencia del método de Newton está garantizada siempre que el punto inicial esté lo suficientemente cerca de algún mínimo local, $\mathbf{x}^*$, en el cual la matriz hessiana sea definida positiva. Además, la razón de convergencia es cuadrática, esto es

$$\|\mathbf{x}_{k+1} - \mathbf{x}^*\| \leq \beta \|\mathbf{x}_k - \mathbf{x}^*\|^2 \quad (4.106)$$

para una constante β positiva.

Si el punto de partida no está lo suficientemente cerca de un mínimo local, entonces la convergencia puede conseguirse mediante los métodos *Line-Search* y de Región de Confianza. El método *Line-Search* modifica la dirección de búsqueda, mientras que el método de Región de Confianza utiliza el modelo cuadrático inicial de la función original, pero restringiendo la búsqueda de la siguiente iteración a un entorno del punto actual donde el modelo cuadrático es lo suficientemente fiable.

En lo que sigue, se utilizarán dos métodos de mínimos cuadrados para aproximar el estado de equilibrio: el método de Gauss-Newton y el método de Levenberg-Marquardt [300, 301, 302], donde la función a minimizar, $r(\mathbf{x})$, se obtiene según la expresión cuadrática siguiente:

$$r(\mathbf{x}) = \frac{1}{2} \|\mathbf{f}(\mathbf{x})\|_2^2 = \frac{1}{2} \sum_{i=1}^n (f_i(\mathbf{x}))^2 \quad (4.107)$$

siendo $f_i(\mathbf{x})$, con $i = 1..n$, cada una de las ecuaciones de estado representadas por (3.21) o (3.55), según el problema a resolver.

Si se denota $\mathbf{J}(\mathbf{x})$ a la matriz jacobiana de $\mathbf{f}(\mathbf{x})$, $\mathbf{G}(\mathbf{x})$ al gradiente de $r(\mathbf{x})$, $\mathbf{H}(\mathbf{x})$ a la matriz hessiana de $r(\mathbf{x})$, y $H_i(\mathbf{x})$ a la matriz hessiana de $f_i(\mathbf{x})$, se tiene que

$$\mathbf{G}(\mathbf{x}) = \mathbf{J}(\mathbf{x})^T \mathbf{f}(\mathbf{x}) \quad (4.108)$$

y

$$\mathbf{H}(\mathbf{x}) = \mathbf{J}(\mathbf{x})^T \mathbf{J}(\mathbf{x}) + \mathbf{Q}(\mathbf{x}), \quad (4.109)$$

donde

$$\mathbf{Q}(\mathbf{x}) = \sum_{i=1}^n f_i(\mathbf{x}) H_i(\mathbf{x}) \quad (4.110)$$

tiende a cero cuando $\|\mathbf{f}(\mathbf{x})\|$ tiende a cero si $\mathbf{x}_k$ se aproxima a la solución.

En el método de Gauss-Newton, la dirección de búsqueda de la nueva iteración, $\mathbf{d}_k$, es la solución del problema de mínimos cuadrados siguiente:

$$\min_{\mathbf{x} \in \mathbb{R}} \|\mathbf{J}(\mathbf{x}_k) \mathbf{d}_k + \mathbf{f}(\mathbf{x}_k)\|^2. \quad (4.111)$$

Esta dirección puede utilizarse a su vez en una estrategia de minimización para garantizar que la función $r(\mathbf{x})$ decrece en cada iteración.

El método de Gauss-Newton puede no ser efectivo si al acercarse a la solución $\|\mathbf{f}(\mathbf{x}_k)\|$ no es un valor pequeño y el término de segundo orden $\mathbf{Q}(\mathbf{x})$ es significativo. El método de Levenberg-Marquardt salva este problema utilizando una dirección de búsqueda que se obtiene como solución de la expresión siguiente:

$$(\mathbf{J}(\mathbf{x}_k)^T \mathbf{J}(\mathbf{x}_k) + \lambda_k \mathbf{I}) \mathbf{d}_k = -\mathbf{J}(\mathbf{x}_k)^T \mathbf{f}(\mathbf{x}_k). \quad (4.112)$$

En esta expresión, λ_k controla la dirección y magnitud de $\mathbf{d}_k$, de forma que si $\lambda_k = 0$, $\mathbf{d}_k$ es la misma que en el método de Gauss-Newton. Conforme λ_k tiende a infinito, $\mathbf{d}_k$ tiende a la dirección de máximo descenso.

Los métodos numéricos presentados, como casi la totalidad de los algoritmos numéricos de minimización de funciones, emplean la matriz jacobiana de la función a minimizar para mejorar la convergencia del algoritmo y la precisión de su resultados [296, 303, 304], de forma que si ésta no está disponible se aproxima por diferencias finitas de la forma:

$$\mathbf{J}(\mathbf{x}) \approx \begin{pmatrix} \frac{f_1(x_1 + \varepsilon, x_2, \dots, x_n) - f_1(\mathbf{x})}{\varepsilon} & \dots & \frac{f_1(x_1, x_2, \dots, x_n + \varepsilon) - f_1(\mathbf{x})}{\varepsilon} \\ \frac{f_2(x_1 + \varepsilon, x_2, \dots, x_n) - f_2(\mathbf{x})}{\varepsilon} & \dots & \frac{f_2(x_1, x_2, \dots, x_n + \varepsilon) - f_2(\mathbf{x})}{\varepsilon} \\ \vdots & \ddots & \vdots \\ \frac{f_n(x_1 + \varepsilon, x_2, \dots, x_n) - f_n(\mathbf{x})}{\varepsilon} & \dots & \frac{f_n(x_1, x_2, \dots, x_n + \varepsilon) - f_n(\mathbf{x})}{\varepsilon} \end{pmatrix} \quad (4.113)$$

donde $\varepsilon > 0$ es una constante pequeña. Además, la matriz jacobiana es utilizada a su vez para el cálculo aproximado de la matriz hessiana, por lo que los errores en las aproximaciones iniciales de la primera se incrementan ineludiblemente en los cálculos de la segunda. Existen dos metodologías para aproximar la matriz hessiana. En la primera de ellas se utiliza la diferencia de dos evaluaciones del vector gradiente en puntos cercanos para obtener por diferencias los valores aproximados de dicha matriz. La segunda metodología, conocida como métodos Quasi-Newton [305], consiste en ir calculando la matriz hessiana de la iteración actual en función del valor del gradiente y de la matriz hessiana de la iteración anterior.

Los métodos descritos anteriormente para la obtención del valor de $\mathbf{x}$ que minimice la función $r(\mathbf{x})$ (4.107), se pueden resumir en el algoritmo de la figura 4.1.

En la sección 4.2 se ha resuelto la matriz jacobiana del modelo borroso de una planta genérica, y en la sección 4.3 la del modelo borroso de un sistema en lazo cerrado. Puesto que es aconsejable emplear el cálculo explícito de la matriz jacobiana siempre que sea posible, dichos cálculos se han incorporado a los algoritmos numéricos para la obtención de los estados de equilibrio de sistemas borrosos, mejorando tanto la velocidad de convergencia como la precisión de los estados de equilibrio obtenidos [291].

```

1: Inicio 1ª iteración,  $k = 1$ 
2: Calcular  $r(\mathbf{x}_1)$  mediante (4.107)
3: Mientras  $r(\mathbf{x}_k) < toletancia$  y  $k < k_{limite}$  hacer
4:   Si está disponible la matriz jacobiana entonces
5:     Calcular  $\mathbf{J}(\mathbf{x})$ 
6:   en otro caso
7:     Calcular la aproximación de  $\mathbf{J}(\mathbf{x})$  mediante (4.113)
8:   Fin Si
9:   Obtener la dirección de descenso:
10:  Si Gauss–Newton entonces
11:    calcular  $\mathbf{d}_k$  mediante (4.111)
12:  en otro caso Si Levenberg–Marquardt entonces
13:    calcular  $\mathbf{d}_k$  mediante (4.112)
14:  Fin Si
15:   $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}_k$ 
16:  Calcular  $r(\mathbf{x}_{k+1})$  mediante (4.107)
17:   $k = k + 1$ 
18: Fin Mientras
19: Retornar  $\mathbf{x}$ 

```

Figura 4.1: Algoritmo numérico de minimización.

4.5. Diseño del software

En la figura 4.2 se muestra el algoritmo que realiza el cálculo del modelo borroso linealizado mediante las expresiones obtenidas en la sección 4.2. Este algoritmo se ha programado haciendo uso del sistema de clases descrito en 3.4, y se incluye en el paquete de utilidades descrito en esa misma sección, ver 3.19. Para facilitar su utilización desde MATLAB®, se ha creado la función `fuzlinear`, la cual obtiene los datos directamente del entorno de MATLAB®, realiza los cálculos empleando el software desarrollado, y retorna los resultados obtenidos.

Antes de proceder con el diseño del algoritmo de cálculo de la matriz jacobiana, se ha diseñado una nueva clase que permitirá el almacenamiento y facilitará el acceso a los elementos de dicha matriz. Esta clase se muestra en la figura 4.3 y se relaciona con las clases anteriores tal como se muestra en la figura 4.4.

El algoritmo que facilita el cálculo de la matriz jacobiana de un sistema de control borroso mediante la resolución de la ecuación (4.97), se muestra en la figura 4.5. Desde MATLAB®, la función `fuzjac` es un *MEX-file* que hace las veces de interfaz entre MATLAB® y la clase `Jacobiana`, de forma que realiza el cálculo de la matriz jacobiana tomando directamente como planta y controlador del sistema dos modelos FIS.

```

1: Calcular el vector de estado extendido  $\tilde{\mathbf{x}}$  según (3.20)
2: Para cada ecuación de estado ( $i = 1..n$ ) hacer
3:   Suma  $w_i = 0$ 
4:   Para cada regla ( $l = 1..M_i$ ) hacer
5:     Calcular  $w_i^l$  según (3.15) y hacer Suma  $w_i = \text{Suma } w_i + w_i^l$ 
6:     Para cada variable de estado ( $q = 1..n$ ) hacer
7:       Calcular  $\partial w_i^l(\mathbf{x}, \mathbf{u})/\partial x_q$  mediante la ecuación (4.48)
8:     Fin Para
9:     Para cada señal de control ( $v = 1..m$ ) hacer
10:      Calcular  $\partial w_i^l(\mathbf{x}, \mathbf{u})/\partial u_v$  mediante la ecuación (4.49)
11:    Fin Para
12:  Fin Para
13: Fin Para
14: Para cada ecuación de estado ( $i = 1..n$ ) hacer
15:   Para cada señal de control ( $v = 1..m$ ) hacer
16:     Calcular  $b_{vi}$  mediante la ecuación (3.18) {Sustituir el índice  $j$  de la ecuación por  $v$ }
17:      $B_{vi} = b_{vi}$  {Se inicializa el elemento  $vi$  de la matriz  $\mathbf{B}$ }
18:     Para cada variable de estado extendida ( $k = 0, 1, \dots, n$ ) hacer
19:       Calcular  $\Gamma_{u_v}^{\mathcal{A}_{ki}} \tilde{x}_k$  utilizando la ecuación (4.40)
20:        $B_{vi} = B_{vi} + \Gamma_{u_v}^{\mathcal{A}_{ki}} \tilde{x}_k$ 
21:     Fin Para
22:     Para cada señal de control ( $j = 1..m$ ) hacer
23:       Calcular  $\Gamma_{u_v}^{\mathcal{B}_{ji}} u_j$  utilizando la ecuación (4.41)
24:        $B_{vi} = B_{vi} + \Gamma_{u_v}^{\mathcal{B}_{ji}} u_j$ 
25:     Fin Para
26:   Fin Para
27:   Para cada variable de estado ( $q = 1..n$ ) hacer
28:     Calcular  $a_{qi}$  mediante la ecuación (3.18) {Sustituir el índice  $k$  de la ecuación por  $q$ }
29:      $A_{qi} = a_{qi}$  {Se inicializa el elemento  $qi$  de la matriz  $\mathbf{A}$ }
30:     Para cada variable de estado extendida ( $k = 0, 1, \dots, n$ ) hacer
31:       Calcular  $\Gamma_{x_q}^{\mathcal{A}_{ki}} \tilde{x}_k$  utilizando la ecuación (4.38)
32:        $A_{qi} = A_{qi} + \Gamma_{x_q}^{\mathcal{A}_{ki}} \tilde{x}_k$ 
33:     Fin Para
34:     Para cada señal de control ( $j = 1..m$ ) hacer
35:       Calcular  $\Gamma_{x_q}^{\mathcal{B}_{ji}} u_j$  utilizando la ecuación (4.39)
36:        $A_{qi} = A_{qi} + \Gamma_{x_q}^{\mathcal{B}_{ji}} u_j$ 
37:     Fin Para
38:   Fin Para
39: Fin Para
40: Obtener  $\mathbf{A}_0$  mediante el algoritmo de la figura 3.2
41: Retornar  $\mathbf{A}_0, \mathbf{A}$  y  $\mathbf{B}$ 

```

Figura 4.2: Algoritmo de linealización de un modelo borroso.

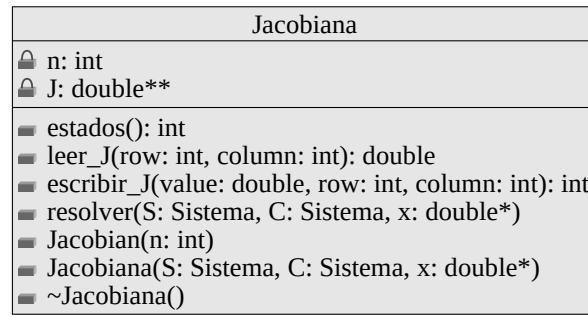


Figura 4.3: Clase *Jacobiana*.

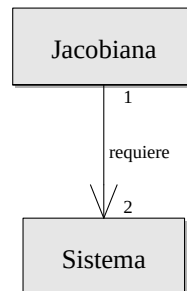


Figura 4.4: Relación entre las clases *Jacobiana* y *Sistema*.

4.6. Conclusiones y comentarios bibliográficos

La utilización de modelos no lineales para representar la dinámica de los sistemas dinámicos permite realizar enfoques más realistas y diseños más óptimos que mediante la utilización de sistemas lineales. En las referencias [220] a [224] se estudian modelos dinámicos no lineales así como distintas técnicas de análisis y diseño existentes para este tipo de sistemas.

La linealización de un modelo borroso de estado es una herramienta que puede resultar muy útil para la obtención de información de un sistema real a partir de su modelo borroso, como por ejemplo para conocer la estabilidad local de un estado de equilibrio. En este capítulo se ha calculado el modelo linealizado de un modelo de estado borroso completamente general, donde no se ha impuesto ningún tipo de limitación sobre el orden de vector de estado ni sobre el orden el vector de control, donde cada una de las variables de salida del modelo borroso puede ser inferida por cualquier número de reglas, y donde los conjuntos borrosos de los antecedentes puede ser descritos por cualquier función de pertenencia, incluso mezclando distintos tipos de funciones de pertenencia en una misma regla.

Al igual que sucede con el modelo borroso de la planta, disponer de la linealización del sistema de control borroso en lazo cerrado, es decir, de su matriz jacobiana, puede resultar de interés para el análisis de dicho sistema. No obstante, la mayor utilidad del cálculo de

```

1: Ejecutar en línea el algoritmo de la figura 3.3 {Proporciona  $\tilde{\mathbf{x}}, \omega_j^r$ , Suma  $\omega_j, c_{kj}$  y  $u_j$ }
2: Para cada ecuación de estado ( $i = 1..n$ ) hacer
3:   Suma  $w_i = 0$ 
4:   Para cada regla ( $l = 1..M_i$ ) hacer
5:     Calcular  $w_i^l$  según (3.15) y hacer Suma  $w_i = \text{Suma } w_i + w_i^l$ 
6:   Fin Para
7: Fin Para
8: Para cada variable de control ( $j = 1..m$ ) hacer
9:   Para cada variable de estado ( $q = 1..n$ ) hacer
10:    Calcular  $\partial\omega_j^r/\partial x_q$  mediante la ecuación (4.94)
11:     $\partial u_j/\partial x_q = c_{qj}$ 
12:    Para cada variable de estado extendida ( $k = 0, 1, \dots, n$ ) hacer
13:      Calcular  $\Gamma_q^{C_{kj}}$  mediante la ecuación (4.100)
14:       $\partial u_j/\partial x_q = \partial u_j/\partial x_q + \Gamma_q^{C_{kj}} \tilde{x}_k$  según (4.90)
15:    Fin Para
16:  Fin Para
17: Fin Para
18: Para cada ecuación de estado ( $i = 1..n$ ) hacer
19:   Para cada variable de control ( $j = 1..m$ ) hacer
20:    Calcular  $b_{ji}$  mediante (3.18)
21:   Fin Para
22:   Para cada variable de estado ( $q = 1..n$ ) hacer
23:    Calcular  $\sum_{j=1}^m b_{ji}c_{qj}$  e inicializar  $J_{iq} = \sum_{j=1}^m b_{ji}c_{qj}$ , según (4.97)
24:    Para cada regla ( $l = 1..M_i$ ) hacer
25:      Calcular el numerador de  $a_{qi}$  según (3.18)
26:      Calcular  $\partial w_i^l(\mathbf{x}, \mathbf{u})/\partial x_q$  mediante (4.91). Debe usarse la ecuación (4.90)
27:    Fin Para
28:    Calcular  $a_{qi} = a_{qi}/\text{Suma } w_i$  y actualizar  $J_{iq} = J_{iq} + a_{qi}$ , según (4.97)
29:    Para cada variable de estado extendida ( $k = 0, 1, \dots, n$ ) hacer
30:       $J_{aux} = 0$ 
31:      Calcular  $\Gamma_q^{A_{ki}}$  mediante (4.98)
32:      Para cada variable de control ( $j = 1..m$ ) hacer
33:        Calcular  $\Gamma_q^{B_{ji}}$  y  $\Gamma_q^{C_{kj}}$  mediante (4.99) y (4.100), respectivamente
34:         $J_{aux} = J_{aux} + c_{kj}\Gamma_q^{B_{ji}} + b_{ji}\Gamma_q^{C_{kj}}$ , según (4.97)
35:      Fin Para
36:       $J_{aux} = J_{aux} + \Gamma_q^{A_{ki}}$ 
37:    Fin Para
38:     $J_{iq} = J_{iq} + J_{aux}\tilde{x}_k$ 
39:  Fin Para
40: Fin Para
41: Retornar J

```

Figura 4.5: Algoritmo para el cálculo de la matriz Jacobiana de un modelo borroso.

la matriz jacobiana de un sistema de control en lazo cerrado radica en poder aplicarla al diseño de dicho sistema de control, por ejemplo mediante la aplicación del teorema de Krasovskii. En este capítulo se ha calculado la matriz jacobiana de un sistema de control borroso siguiendo la misma filosofía de generalidad empleada anteriormente para calcular el modelo linealizado de un modelo borroso, considerándose además que la planta y el controlador son a todos los efectos dos sistemas borrosos completamente independientes y que, por lo tanto, ni el número de reglas ni la partición del universo de discurso de las variables de estado para la planta y el controlador tienen porqué coincidir.

En las referencias [100], [233], y de [290] a [294] se muestra la utilidad de aplicar técnicas de análisis y diseño de sistemas no lineales, como la linealización, la teoría de estabilidad de Lyapunov, el método de Krasovskii, etc., a los sistemas de control borrosos.

Los estados de equilibrio de un sistema proporcionan mucha información sobre el comportamiento del mismo, y su búsqueda y estudio suele ser uno de los primeros problemas que se deben resolver para estudiar su dinámica. En este capítulo se ha presentado un algoritmo numérico que permite obtener los estados de equilibrio de un sistema borroso, independientemente de si se desea estudiar sólo la planta o el sistema de lazo cerrado. Se ha propuesto hacer uso de las linealizaciones realizadas con anterioridad para acelerar la velocidad de convergencia de estos algoritmos y la precisión de los resultados obtenidos. En las referencias [291] y de [295] a [305] se pueden consultar los métodos numéricos empleados en esta sección.

Finalmente, en la última sección del capítulo se han presentado los algoritmos que permiten calcular a partir de un sistema de control borroso, tanto el sistema linealizado de la planta, como la matriz jacobiana del sistema de lazo cerrado.

De entre las referencias incluidas en este capítulo merece la pena destacar [220] y [224] como lecturas básicas para el estudio de los sistemas no lineales y las metodologías de estudio y aplicación de dichos sistemas. También es importante destacar [168], donde se propone el cálculo de la matriz jacobiana de un sistema de control borroso y una estructura software para su realización; así como [291], donde se realiza el estudio de los estados de equilibrio de varios modelos borrosos y se demuestra el beneficio de incluir el cálculo explícito de la matriz jacobiana en dicho estudio.

Diseño de controladores borrosos

EL diseño de controladores borrosos puede realizarse fundamentalmente desde dos enfoques: a partir del conocimiento de un operario experto, y mediante un estudio formal de estabilidad del sistema de control. En este capítulo se estudiarán ambos enfoques, para ello el capítulo se estructura en cuatro secciones. En la primera de ellas se estudia la teoría de estabilidad de Lyapunov, ya que esta teoría es la base del análisis de estabilidad y el diseño en sistemas no lineales. La segunda sección está dedicada al control heurístico basado en el conocimiento de un experto. Este tipo de control tiene la ventaja de no requerir un modelo de la planta, ya que el diseño se realiza a partir del conocimiento que el operario posee sobre su manejo. En la tercera sección se propone una metodología de diseño completamente formal y general para la síntesis de sistemas de control borroso estables por diseño. Finalmente, la última sección recoge las principales conclusiones de este capítulo y una revisión de las referencias bibliográficas incluidas en el mismo.

5.1. Introducción

El problema del control de una planta está íntimamente ligado al concepto de estabilidad. Hasta tal punto están ligados dichos conceptos, que no tiene sentido hablar de sistema de control si éste no es estable. Cuando se desea diseñar un sistema de control para una planta no lineal es deseable, y a veces imprescindible, no realizar aproximaciones lineales de la misma, ya que la pérdida de información que esto supone puede provocar que el sistema resultante no sea estable o lo sea muy localmente.

Los sistemas no lineales presentan una gran riqueza en cuanto a sus respuestas dinámicas. Esta riqueza dificulta enormemente la consecución de una teoría que permita el diseño de sistemas de control de forma sistemática, tal como sucede con los sistemas lineales. No obstante, siempre y cuando el modelo matemático de la planta sea conocido, existen multitud de técnicas de control que pueden ser aplicadas para la síntesis de controladores no lineales, como la linealización por realimentación, la planificación de ganancias, el control adaptativo o el control deslizante, entre otras. Dada la capacidad inherente de los siste-

mas borrosos para modelar dinámicas no lineales, estas técnicas pueden emplearse para la síntesis de controladores borrosos.

La carencia de herramientas formales para el análisis dinámico de los sistemas de control borroso, junto con la utilización frecuente de soluciones empíricas o construidas *ad-hoc*, provocaron que, durante muchos años, gran parte de la comunidad científica del ámbito de la automática, no terminara de aceptar a la lógica borrosa como una técnica formal de diseño de sistemas de control [306]. Como muestra se transcriben las palabras de Rudolph E. Kalman, uno de los padres de la ingeniería de control moderna, a Lofti A. Zadeh en 1972:

... I would like to comment briefly on Prof. Zadeh's presentation. His proposals could be severely, ferociously, even brutally criticized from a technical point of view. This would be out of place here. But a blunt question remains: Is Zadeh presenting important ideas or is he indulging in wishful thinking? The most serious objection of fuzzification of system analysis is that lack of methods of system analysis is not the principal scientific problem in the systems field. That problem is one of developing basic concepts and deep insight into the nature of systems, perhaps trying to find something akin to the laws of Newton. In my opinion, Zadeh's suggestions have no chance to contribute to the solution of this basic problem...

La crítica de Kalman a Zadeh es realmente dura, catalogándolo casi de visionario. Nótese además cómo hace especial énfasis en la falta de una sistematización en el análisis. De forma similar, el brillante y admirado colega de Zadeh, Prof. William Kaham, afirmó en 1975 lo siguiente:

... Fuzzy theory is wrong, wrong, and pernicious. I can not think of any problem that could not be solved better by ordinary logic. What Zadeh is saying is the same sort of things: Technology got us into this mess and now it can't get us out. Well, technology did not get us into this mess. Greed and weakness and ambivalence got us into this mess. What we need is more logical thinking, not less. The danger of fuzzy theory is that it will encourage the sort of imprecise thinking that has brought us so much trouble...

Esta crítica es aún más dura que la anterior, y se cataloga a Zadeh poco menos que de causante de la ruptura en el orden del pensamiento científico.

Aún en los años 90 podían encontrarse críticas, aunque ya ciertamente escasas, como la que Jon Konieki declaró en *AI Expert* en 1991:

... Fuzzy logic is based on fuzzy thinking. It fails to distinguish between the issues specifically addressed by the traditional methods of logic, definition, and statistical decision-making...

En la actualidad la lógica borrosa ha vencido todos estos prejuicios gracias a los excelentes resultados prácticos que ha cosechado, pero también gracias al esfuerzo de formalización

realizado por la comunidad científica. En este sentido han trabajado muchos investigadores, fundamentalmente a partir de la década de los 90, proporcionando a la lógica borrosa una base formal centrada en la teoría de control no lineal [121,307]. De esta forma pudieron empezar a aplicarse a los sistemas de control borroso, los métodos clásicos de análisis de la teoría de control no lineal: el criterio de hiperestabilidad de Popov [308,309,310,311], el criterio del círculo [175,312], el método directo de Lyapunov [227,313,314,315,316,317], las técnicas de análisis en el espacio de fase [175,318], el método de la función descriptiva [319], los métodos basados en índices de estabilidad y robustez de sistemas [100,319], el criterio de conicidad [319,320], los métodos basados en la teoría de estabilidad de entrada/salida [312,319,321] y los métodos heurísticos [322,323].

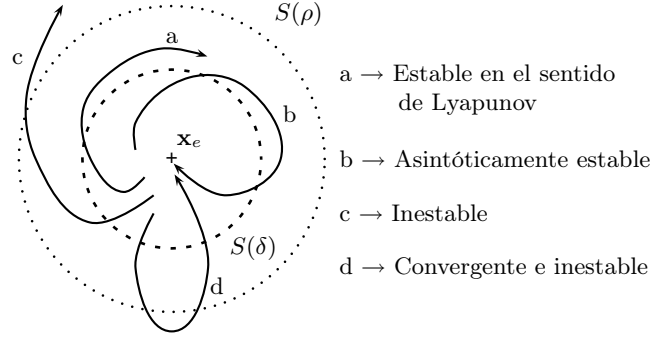
Más recientemente se han empleado soluciones basadas en LMIs (*Linear Matrix Inequalities*) como metodología para garantizar la estabilidad asintótica del sistema de control borroso en lazo cerrado [324,325,326,327,328,329,330,331]. En estos métodos, la evolución de cada una de las variables de estado requiere del mismo número de reglas, ya que las reglas se suelen expresar de forma matricial; los antecedentes están intencionadamente bien distribuidos, normalmente cumpliendo los requisitos de la *Standard Fuzzy Partition* (SFP) [332]; y se suele considerar nulo o despreciable el término afín de los consecuentes. Mediante todas estas limitaciones, los autores que trabajan con LMIs minimizan el esfuerzo de diseño y la complejidad del sistema no lineal, reduciéndolo a un conjunto de sistemas lineales más fácilmente abordables. En general estos métodos obtienen buenos resultados prácticos, y permiten un diseño relativamente sencillo del controlador borroso, sin embargo, su ámbito de aplicación queda restringido por la necesidad de conocer previamente el modelo matemático de la planta, ya que de lo contrario sería complicado poder cumplir todas las restricciones impuestas.

La mayoría de los trabajos publicados sobre análisis de estabilidad de sistemas de control borroso, parten del conocimiento previo del modelo matemático de la planta [121]. Cuando el modelo matemático de una planta no lineal es conocido, existen muchas técnicas de control que permiten garantizar la estabilidad del mismo, con resultados probados y avalados por la comunidad científica. Por lo tanto, en principio no parece muy justificable la utilización de un controlador borroso en estos casos, aunque se puede respaldar su elección bajo la robustez que suelen presentar los controladores borrosos, o por la necesidad de obtener un controlador interpretable lingüísticamente. Sin embargo, si el modelo de la planta es desconocido, o es demasiado complejo como para obtenerlo, o está sometido a incertidumbres, la lógica borrosa se presenta como una herramienta poderosa, posiblemente una de las mejores, para afrontar el diseño del controlador no lineal.

5.2. Teoría de estabilidad de Lyapunov

La teoría de estabilidad de Lyapunov permite tanto el análisis de los sistemas de control no lineales, como la síntesis de controladores estables.

Teorema 5.1 (*Estabilidad en el sentido de Lyapunov*). Se dice que un estado de equilibrio $\mathbf{x}_e$ es estable en el sentido de Lyapunov, si dado un número real $\rho > 0$, existe otro número

**Figura 5.1:** Estabilidad.

real $\delta > 0$, de forma que para todo estado inicial $\mathbf{x}(t_0)$ para el que se cumpla

$$\|\mathbf{x}(t_0) - \mathbf{x}_e\| < \delta, \quad (5.1)$$

también se cumple

$$\|\mathbf{x}(t) - \mathbf{x}_e\| < \rho, \forall t > t_0, \quad (5.2)$$

donde $\|\mathbf{x}(t) - \mathbf{x}_e\|$ es la norma euclídea¹, que se define por

$$\|\mathbf{x}(t) - \mathbf{x}_e\| = \sqrt{(x_1(t) - x_{1e})^2 + (x_2(t) - x_{2e})^2 + \dots + (x_n(t) - x_{ne})^2}. \quad (5.3)$$

Como se puede ver de forma gráfica en la figura 5.1, el teorema anterior implica que se considera estable en el sentido de Lyapunov a todo estado de equilibrio cuyas trayectorias que se originen en sus cercanías, dentro de la región $S(\delta)$, no abandonan un entorno de dicho estado, región $S(\rho)$, cuando el tiempo aumenta de forma indefinida.

La estabilidad tal como se define en el sentido de Lyapunov no suele ser una condición suficiente para los sistemas de control, por lo que se utiliza el concepto de estabilidad asintótica.

Teorema 5.2 (Estabilidad asintótica). Un estado de equilibrio $\mathbf{x}_e$ se dice que es asintóticamente estable, si es estable en el sentido de Lyapunov y toda trayectoria que comience dentro de la región $S(\delta)$, converge hacia $\mathbf{x}_e$ sin abandonar $S(\rho)$ en ningún momento. De forma más precisa, se dice que $\mathbf{x}_e$ es asintóticamente estable, si dados dos números reales $\rho > \eta > 0$ tan pequeños como se desee, existe un número real $\delta > 0$ y un instante de tiempo t_f , de forma que se cumple (5.1), (5.2) y además

$$\|\mathbf{x}(t) - \mathbf{x}_e\| < \eta, \forall t > t_f. \quad (5.4)$$

Teorema 5.3 (Estabilidad asintótica global). Un estado de equilibrio se dice que es asintóticamente estable de forma global, si es estable y toda solución converge hacia $\mathbf{x}_e$ al incrementar el tiempo indefinidamente, es decir, si la región $S(\delta)$ es todo el espacio de estado. Obviamente, para que haya estabilidad asintótica global es condición necesaria que haya un solo estado de equilibrio en todo el espacio de estado.

¹La norma euclídea suele ser la más utilizada, aunque puede emplearse cualquier otra variante, como la norma uno o la norma infinito.

La estabilidad asintótica global es una cualidad deseable en todo sistema de control, sin embargo, no siempre puede conseguirse o demostrarse dicha cualidad. En estos casos, el problema se transforma en la determinación de la mayor región de estabilidad asintótica, lo cual suele ser un problema complejo. Desde el punto de vista práctico, basta con determinar una región de estabilidad asintótica suficientemente grande como para que las perturbaciones sufridas por el sistema no la excedan.

Teorema 5.4 (Inestabilidad). Se dice que un estado de equilibrio $\mathbf{x}_e$ es inestable, si dado un número real $\rho > 0$ y cualquier otro número real $\delta > 0$, existe un estado $\mathbf{x}(t_0)$ en la vecindad de $\mathbf{x}_e$, es decir, que satisface (5.1), para el que las trayectorias que parten de $\mathbf{x}(t_0)$ cumplen

$$\|\mathbf{x}(t) - \mathbf{x}_e\| > \rho, \quad (5.5)$$

en algún instante $t > t_0$, es decir, superan el límite dado por ρ . Nótese que las trayectorias no tienen porqué fugar a infinito, sino que basta con que se alejen de $\mathbf{x}_e$ más allá de la región $S(\rho)$, por ejemplo para acabar en otro estado de equilibrio o en un ciclo límite.

5.2.1. Método directo de Lyapunov

De la teoría clásica de la mecánica se sabe que un sistema es estable si su energía total, una función definida positiva, es decreciente de forma continua hasta alcanzar un estado de equilibrio. En un sistema no lineal no tiene porqué existir una manera simple de definir una función de energía, por ello, el método directo de Lyapunov introduce una función de energía ficticia, denominada *función de Lyapunov*.

Sea una región S cerrada del espacio de estado, sin huecos, con un estado de equilibrio $\mathbf{x}_e$ no en el mismo borde; una función escalar $V(\mathbf{x}(t))$ es una función de Lyapunov del sistema en S , si:

1. Es definida positiva en toda la región, excepto en $\mathbf{x}_e$, donde es nula:

$$\begin{aligned} V(\mathbf{x}(t)) &= 0 \quad \text{para} \quad \mathbf{x}(t) = \mathbf{x}_e \\ V(\mathbf{x}(t)) &> 0 \quad \text{para} \quad \mathbf{x}(t) \neq \mathbf{x}_e \end{aligned} \quad (5.6)$$

2. Su derivada es decreciente en toda la región:

$$\frac{dV(\mathbf{x}(t))}{dt} \leq 0, \forall \mathbf{x} \in S \quad (5.7)$$

3. Es continuamente diferenciable en $\mathbf{x}(t)$:

$$\exists \frac{\partial V(\mathbf{x}(t))}{\partial x_i}, \forall x_i \quad (5.8)$$

En la figura 5.2 se muestra una posible función de Lyapunov para un sistema de 2º orden, donde el estado de equilibrio se sitúa en el origen de coordenadas.

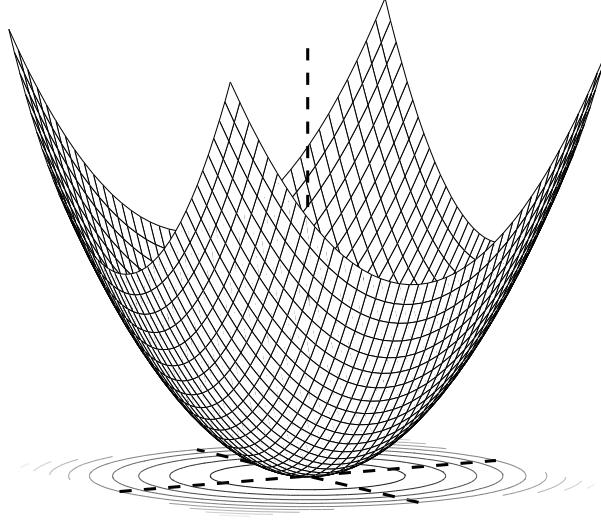


Figura 5.2: *Función de Lyapunov.*

El método directo de Lyapunov establece que, si se halla una función de Lyapunov alrededor de un estado de equilibrio, el sistema es estable en el sentido de Lyapunov en las cercanías de dicho estado. Nótese como el método directo establece una condición suficiente, pero no necesaria, por lo que no encontrar una función de Lyapunov no implica que el sistema no sea estable.

Si además se exige que $V(\mathbf{x})$ sea estrictamente decreciente, esto es, que

$$\begin{aligned} \frac{dV(\mathbf{x}(t))}{dt} &= 0 \quad \text{para} \quad \mathbf{x}(t) = \mathbf{x}_e \\ \frac{dV(\mathbf{x}(t))}{dt} &< 0 \quad \text{para} \quad \mathbf{x}(t) \neq \mathbf{x}_e, \end{aligned} \quad (5.9)$$

entonces, el sistema es asintóticamente estable.

El método directo de Lyapunov también se puede aplicar para estudiar la estabilidad o estabilidad asintótica global, es decir, cuando la región S es todo el espacio de estado. En este caso es necesario exigir a la función de Lyapunov que cumpla:

$$\lim_{\|\mathbf{x}(t)\| \rightarrow \infty} V(\mathbf{x}(t)) = \infty, \quad (5.10)$$

ya que de esta forma se asegura que las curvas o superficies de nivel correspondientes a un mismo valor de $V(\mathbf{x})$ sean cerradas. Si estas curvas o superficies de nivel no fuesen cerradas, podrían producirse trayectorias que se escapasen del estado de equilibrio mientras disminuye el valor de la función de Lyapunov.

La teoría de Lyapunov es la más flexible a la hora de abordar el diseño de un sistema de control no lineal [333], sin embargo, esta flexibilidad también supone un problema práctico; ya que no suele ser fácil establecer una función de Lyapunov candidata que propicie un diseño estable en una región suficientemente amplia. Además, como se ha visto

anteriormente, el no cumplimiento del teorema de Lyapunov para una función candidata no implica que no pueda cumplirse para otra función distinta, de forma que la elección de una u otra función de Lyapunov forma parte del propio proceso de diseño.

5.2.2. Método de Krasovskii

Es fácil intuir la enorme problemática que supone el encontrar funciones de Lyapunov para sistemas no lineales generales. El método de Krasovskii sugiere una forma simple de función de Lyapunov candidata (5.13) para sistemas no lineales autónomos de la forma (5.11). La idea básica del método es simplemente comprobar si esta elección satisface los requisitos de una función de Lyapunov.

Teorema 5.5 (*de Krasovskii*). Sea un sistema autónomo no lineal

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)), \quad (5.11)$$

con un estado de equilibrio aislado situado, sin pérdida de generalidad², en el origen del espacio de estado, y sea $\mathbf{J}(\mathbf{x}(t))$ la matriz jacobiana del sistema; si la matriz

$$\mathbf{F}(\mathbf{x}(t)) = \mathbf{J}(\mathbf{x}(t))^T + \mathbf{J}(\mathbf{x}(t)) \quad (5.12)$$

es definida negativa en una región Ω alrededor del origen, se cumple que el estado de equilibrio en el origen es asintóticamente estable, y que

$$V(\mathbf{x}(t)) = \mathbf{f}(\mathbf{x}(t))^T \mathbf{f}(\mathbf{x}(t)) \quad (5.13)$$

es una función de Lyapunov para el sistema. Además, si Ω es todo el espacio de estado y se cumple (5.10), se puede afirmar que el estado de equilibrio es asintóticamente estable de forma global.

El teorema de Krasovskii proporciona condiciones suficientes de estabilidad asintótica para sistemas no lineales, y condiciones necesarias y suficientes para sistemas lineales, y proporciona una función de Lyapunov válida para el sistema. Esto significa que un estado de equilibrio de un sistema no lineal puede ser estable aún sin satisfacer las condiciones de este teorema, sin embargo, si se diseña un controlador de forma que cumpla el teorema de Krasovskii, la estabilidad asintótica en un entorno del estado de equilibrio estará garantizada.

Mientras que el uso del teorema de Krasovskii es muy directo, su aplicabilidad práctica está limitada, ya que muchos sistemas no cumplen con la definición de negatividad de $\mathbf{F}(\mathbf{x}(t))$. Para salvar en parte este inconveniente, este teorema puede enunciarse de la forma más general siguiente:

Teorema 5.6 (*de Krasovskii generalizado*). Es condición suficiente para la estabilidad asintótica del sistema (5.11) en una región Ω alrededor del origen, que la matriz

$$\mathbf{F}(\mathbf{x}(t)) = \mathbf{J}(\mathbf{x}(t))^T \mathbf{P} + \mathbf{P} \mathbf{J}(\mathbf{x}(t)) + \mathbf{Q} \quad (5.14)$$

²En el caso de que el estado de equilibrio no se situase en el origen de coordenadas, siempre será posible establecer un cambio de base del sistema de coordenadas que lo sitúe en dicha posición.

sea semidefinida negativa; siendo $\mathbf{P}$ y $\mathbf{Q}$ dos matrices simétricas definidas positivas. En este caso, la función de Lyapunov resultante es:

$$V(\mathbf{x}(t)) = \mathbf{f}(\mathbf{x}(t))^T \mathbf{P} \mathbf{f}(\mathbf{x}(t)). \quad (5.15)$$

5.3. Control heurístico de sistemas dinámicos

La heurística es la manera de buscar la solución de un problema mediante métodos no rigurosos, como por tanteo, mediante reglas empíricas, etc. Por ejemplo, el control manual que realiza un operario experimentado sobre un sistema es un tipo de control heurístico, donde el operario decide las acciones de control en función del estado del sistema y de su experiencia y conocimientos sobre el mismo.

Tal como se vio en el capítulo 2, la lógica borrosa permite inferir conclusiones a partir de una serie de condiciones expresadas de forma lingüística mediante reglas Si-Entonces. Esta capacidad de la lógica borrosa puede ser empleada para incorporar el conocimiento de un experto a un sistema de control automático, obteniéndose un controlador borroso basado en heurística cuyo comportamiento, una vez realizados los ajustes pertinentes, puede ser bastante similar al del operario real. Además, no es necesario disponer de un modelo de la planta a controlar, ya que en muchos casos basta con el conocimiento del operario y el *modelo mental* que éste posee sobre la planta para desarrollar un controlador. Precisamente estos aspectos de la lógica borrosa fueron los que resultaron más atractivos en sus inicios, y los que provocaron el gran auge que tuvo esta tecnología durante los años 70. Por ejemplo, Mandani y Assilian [28] describieron en 1975 un experimento sobre la síntesis lingüística de un controlador para una máquina de vapor, en la que la lógica borrosa es usada para convertir las reglas de control heurístico obtenidas de operadores humanos en una estrategia de control automático. Más recientemente, aún se siguen proponiendo mecanismos de control para sistemas complejos basados en heurística, como por ejemplo, un sistema de navegación para robots móviles [263], un sistema de balanceado de cargas para servidores distribuidos [334], o el control de una grúa porta contenedores [171].

El no disponer de modelo de la planta y, por tanto, de modelo del sistema en lazo cerrado, ciertamente puede traer problemas asociados, como por ejemplo la imposibilidad de poder realizar un análisis formal de propiedades del sistema tales como estabilidad y robustez. Sin embargo, las múltiples herramientas existentes hoy en día para simular sistemas complejos, permiten que vía simulaciones, y de una forma un tanto empírica si se quiere, puedan ser analizadas las propiedades anteriores en las regiones de operación de interés.

Al contrario de lo que suele suceder en el traslado de una técnica al campo práctico, en el caso del control borroso los éxitos técnicos y comerciales han ido por delante del desarrollo de la teoría. Esto puede ser atribuido a dos factores [335]:

1. Los mecanismos de la lógica borrosa proporcionan una herramienta útil y exitosa para implementar reglas lingüísticas. En concreto, dadas las reglas lingüísticas, la realización del controlador borroso exige unos tiempos y costes de desarrollo modestos.

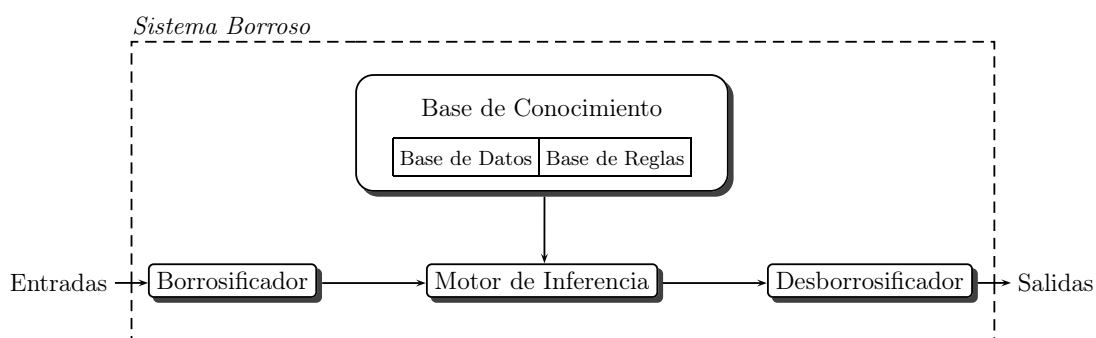


Figura 5.3: Sistema de inferencia borroso.

2. El hecho de que el controlador esté realizado mediante reglas lingüísticas permite entender fácilmente su comportamiento. En concreto, si el comportamiento del sistema en lazo cerrado no es satisfactorio, suele ser posible determinar la regla responsable y en qué dirección ha de ser modificada.

El enfoque de diseño de controladores borrosos heurísticos se ha basado en dos fuentes de información: conocimiento de la dinámica de la planta y conocimiento de su manejo por un operario experto. El buen conocimiento de la planta permite identificar las variables que intervienen en el problema y su comportamiento, así como contribuir a derivar las reglas del controlador. Por ejemplo: *Si la variable a controlar tiene el valor deseado y su derivada es cero, entonces mantener la señal de control actual*. Respecto del enfoque de extraer conocimiento del operario experto, se trata de implementar ese conocimiento en las reglas del controlador borroso. Claramente, este es el mecanismo usado en el diseño clásico de *sistemas expertos*.

En la figura 5.3 se muestra el esquema de control borroso presentado en el capítulo 2, donde se puede apreciar que el proceso de control borroso consta de tres pasos: borrosificación, inferencia borrosa y desborrosificación.

En primer lugar, la etapa de borrosificación traslada las variables medidas al dominio lingüístico mediante las funciones de pertenencia. A continuación, la etapa de inferencia borrosa evalúa, mediante el conjunto de reglas del controlador, el comportamiento del sistema y las acciones de control a realizar. El resultado son, de nuevo, variables lingüísticas. Puesto que al final del proceso de inferencia se ha de actuar sobre la planta, la etapa de desborrosificación traslada los resultados lingüísticos de la fase de inferencia a los valores reales que finalmente se aplican a la planta.

El diseño de controladores borrosos basado en heurística, que también podría denominarse tradicional, tiene la desventaja de carecer, por regla general, de una metodología formal de diseño y análisis. Sin embargo, es posible establecer una metodología de diseño, más bien empírica, que podría resumirse en los siguientes pasos: a partir del conocimiento de la planta se definen los conjuntos borrosos, y por tanto las variables lingüísticas que originan las entradas del controlador borroso, las señales de control y la/s salida/s. Una vez creadas las variables lingüísticas de entrada, salida y control, y definidos sus universos de discurso, se deducen las pautas que permiten definir la estrategia de control, empleando

para el estudio la forma de operar de un experto sobre la planta. Estas pautas de operación deben representarse en forma de reglas Si-Entonces para poder ser almacenadas en la base de reglas del controlador borroso.

Una vez diseñado un controlador inicial, será necesario probar su comportamiento sobre el sistema para refinar cada uno de sus componentes. Para ello se pueden utilizar dos estrategias distintas: una estrategia práctica, que implica la implementación del controlador sobre el sistema y su prueba real; y una estrategia basada en la simulación, para lo cual será necesario obtener un modelo de la planta. Obviamente, teniendo en cuenta las sucesivas fases de refinamiento que requeriría un problema de control complejo, unido a los problemas que puede conllevar la implementación de un controlador inadecuado sobre un sistema real, parece procedente realizar siempre que sea posible un primer ajuste mediante simulación y, posteriormente, realizar un ajuste fino en campo.

5.4. Síntesis de controladores borrosos estables por diseño

Uno de los objetivos fundamentales de este texto es establecer una metodología lo más general posible que permita la síntesis de controladores borrosos estables por diseño, es decir, que la estabilidad quede garantizada durante el propio proceso de diseño del controlador.

Con objeto de hacer la metodología lo más general posible, se supondrá como hipótesis de partida que la dinámica de la planta es totalmente desconocida, o lo suficientemente compleja como para no poderse obtener un modelo matemático de la misma. Evidentemente, en un caso práctico sería posible incorporar los datos que se conozcan del sistema para facilitar tanto el modelado de la planta como el diseño del controlador borroso.

Para diseñar un controlador para una planta desconocida se propone una metodología de diseño consistente en los pasos siguientes [170, 233, 293]:

1. Identificación de la planta y representación en forma de modelo borroso de estado.
2. Estudio de la planta y obtención de información dinámica de la misma a partir de su modelo borroso.
3. Diseño de un controlador borroso que garantice la estabilidad asintótica en lazo cerrado.

5.4.1. Identificación de la planta

El primer paso antes de controlar un sistema es conocer su dinámica. Puesto que se parte de la premisa de que la única información disponible sobre la planta es un histórico de datos de entrada/salida, se debe obtener un modelo de la misma en base a dichos datos.

Los sistemas no lineales presentan dinámicas complejas que pueden ser difíciles de modelar con técnicas matemáticas usuales, sin embargo, tal como se vio en el capítulo 2, los modelos borrosos de tipo Takagi-Sugeno (TS o TSK) pueden emplearse para representar dinámicas complejas, ya que son aproximadores universales tanto de la función [210, 211, 212, 213] como de su derivada [214, 215, 216, 217].

Si se considera una planta multivariable completamente general, sin limitaciones en el tamaño del vector de estado ni en el del vector de control, ésta puede representarse por un conjunto de reglas borrosas como las mostradas en la expresión (3.12). Esta representación facilita un modelo en forma de variables de estado del sistema, donde cada regla es completamente independiente del resto, y donde no se limita ni el tipo ni la distribución de sus funciones de pertenencia.

La identificación de la estructura del modelo borroso se puede realizar mediante las técnicas que se han presentado en la sección 3.3.1, y la estimación de sus parámetros adaptables mediante los algoritmos mostrados en 3.3.2. Una vez que el modelo borroso ha sido validado adecuadamente, ver sección 3.3.3, se dispone de un modelo que puede representarse de forma lingüística y que refleja la dinámica del sistema con la precisión requerida. El modelo borroso obtenido puede emplearse tanto para estudiar la dinámica del sistema, como para diseñar un controlador que garantice su estabilidad asintótica.

La implementación y tratamiento de modelos borroso puede realizarse mediante el software desarrollado en las secciones 3.4 y 4.5, junto con MATLAB® y la *Fuzzy Logic Toolbox* [287].

5.4.2. Estudio dinámico de la planta

El conocimiento de los estados de equilibrio que posee un sistema, así como la estabilidad de dichos estados, son datos que pueden resultar muy interesantes a la hora de analizar o diseñar un sistema de control. Si se parte de un sistema completamente desconocido, esta información puede aclarar en muchos aspectos el funcionamiento del mismo, así como facilitar el diseño de un sistema de control adecuado.

En el capítulo 4 se presentaron las herramientas matemáticas que se han desarrollado para extraer información de un modelo borroso, así como el software creado para tal fin: linealización de un modelo borroso de estado, en la sección 4.2; matriz jacobiana de un sistema de control borroso, en la sección 4.3; y una metodología para la obtención de los estados de equilibrio del sistema, en la sección 4.4.

Empleando estas herramientas es posible, por ejemplo, identificar de forma aproximada los estados de equilibrio de un sistema a partir de su modelo borroso, y analizar su estabilidad local mediante la linealización del sistema en dichos puntos.

5.4.3. Diseño del controlador borroso

Como se vio en 3.2.1, un controlador multivariable completamente general, sin limitaciones ni en el tamaño del vector de control ni en el vector de estado, puede definirse mediante el conjunto de reglas (3.34). El proceso de modelado de la planta y el de diseño del controlador se consideran completamente independientes, de forma que cada uno de los modelos borrosos podrá tener un número distinto de reglas, distintas funciones de pertenencia, etc. Además, al igual que sucede con la planta, cada regla se considera completamente independiente del resto, y no se limita ni el tipo ni la distribución de sus funciones de pertenencia.

Como se ha mencionado en este capítulo, la teoría de Lyapunov puede ser utilizada para analizar la estabilidad de los sistemas borrosos. De hecho, en la literatura se ha abordado tanto problemas de estabilización como de seguimiento con controladores de tipo TSK utilizando funciones de Lyapunov [229, 232, 271, 336, 337]; sin embargo, quizás el problema principal que aparece es encontrar la función de Lyapunov necesaria.

El teorema de Krasovskii generalizado proporciona una función de Lyapunov para el sistema, e impone unas condiciones suficientes de estabilidad asintótica más flexibles que en su versión simplificada. Este teorema puede ser aplicado como herramienta de síntesis de controladores borrosos estables por diseño, es decir, que la estabilidad quede garantizada durante el propio proceso de diseño del controlador [168, 233, 293, 294]. Aunque la idea puede parecer muy intuitiva, su puesta en práctica resulta compleja por varias razones: 1) se han de obtener las matrices $\mathbf{P}$ y $\mathbf{Q}$ necesarias para su comprobación; 2) se debe establecer un algoritmo de ajuste de los parámetros adaptables del controlador borroso, de forma que se satisfaga la condición sobre $\mathbf{F}(\mathbf{x}(t))$ que asegure la estabilidad asintótica del sistema; y 3) hay que tener en cuenta que, puesto que el teorema de Krasovskii debe satisfacerse en un entorno del estado de equilibrio, no basta con evaluarlo sólo en dicho punto, sino que debe asegurarse su cumplimiento en una región suficientemente amplia alrededor de dicho estado. Además, la segunda razón dada más arriba, requiere la obtención de la matriz jacobiana del sistema de control borroso en lazo cerrado. No obstante lo anterior, estas dificultades están resueltas de forma completamente general, incluso permitiendo la mezcla de distintos tipos de funciones de pertenencia en un mismo modelo [168, 231, 290]. El procedimiento se ha desarrollado completamente en el apartado 4.3.

En problemas relativamente complejos, el espacio de soluciones que permiten el cumplimiento del teorema de Krasovskii en una región suficientemente amplia alrededor del estado de equilibrio, es demasiado restrictivo. Por lo tanto, la utilización del teorema de Krasovskii, incluso en su versión generalizada, como metodología general de diseño de controladores borrosos sólo es viable en casos relativamente simples [168, 233, 293].

Con objeto de utilizar la teoría de Lyapunov para la resolución de problemas de mayor complejidad que los que pueden abordarse mediante el teorema de Krasovskii, se propone el teorema siguiente:

Teorema 5.7. Considérese el sistema autónomo general definido por³:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}), \quad (5.16)$$

donde el estado de equilibrio de interés se sitúa, sin pérdida de generalidad⁴, en el origen de coordenadas, y sean $s \in \mathbb{N}$ regiones Θ_q , con $q=1..s$, definidas en el espacio de estado mediante s funciones continuamente diferenciables $\varphi_q(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}^1$, tal que $\varphi_q(\mathbf{x}) \in \mathcal{C}^1$. Sea $\mathbf{J}(\mathbf{x})$ la matriz jacobiana del sistema (5.16). Una condición suficiente para que el origen sea asintóticamente estable es que existan s matrices simétricas definidas positivas,

³La dependencia de las variables de estado con respecto al tiempo se han omitido en todo el desarrollo del teorema para simplificar la notación y mejorar su claridad.

⁴En el caso de que el estado de equilibrio no se situase en el origen de coordenadas, siempre será posible establecer un cambio de base del sistema de coordenadas que lo sitúe en dicha posición.

$\mathbf{P}_1, \dots, \mathbf{P}_s$, tal que $\forall \mathbf{x} \neq \mathbf{0}$, la matriz

$$\mathbf{F}(\mathbf{x}) = \sum_{q=1}^s \left(\varphi_q(\mathbf{x}) (\mathbf{J}(\mathbf{x})^T \mathbf{P}_q + \mathbf{P}_q \mathbf{J}(\mathbf{x})) + \frac{d\varphi_q(\mathbf{x})}{d\mathbf{x}} \mathbf{f}(\mathbf{x}) \mathbf{P}_q \right) \quad (5.17)$$

sea definida negativa en un entorno $B_V(\mathbf{0}) \subseteq \bigcup_{q=1}^s \Theta_q$ del origen de coordenadas. En este caso, la función

$$V(\mathbf{x}) = \mathbf{f}(\mathbf{x})^T \left(\sum_{q=1}^s \varphi_q(\mathbf{x}) \mathbf{P}_q \right) \mathbf{f}(\mathbf{x}) \quad (5.18)$$

es una función de Lyapunov del sistema (5.16). Además, si la región $B_V(\mathbf{0})$ es el espacio de estado completo y

$$\lim_{\|\mathbf{x}\| \rightarrow \infty} V(\mathbf{x}) = \infty, \quad (5.19)$$

entonces el sistema es asintóticamente estable de forma global.

Demostración. En primer lugar se debe garantizar que $\forall \mathbf{x} \neq \mathbf{0}$, $V(\mathbf{x}) > 0$, ver (5.6), pero esto queda demostrado por el hecho de que $\varphi_q(\mathbf{x}) \geq 0$ para todo $\mathbf{x} \in B_V(\mathbf{0})$, y que $\mathbf{P}_q$ son matrices definidas positivas para todo $q = 1..s$. Por otro lado, se puede afirmar también que la continuidad de las derivadas parciales de $V(\mathbf{x})$, ecuación (5.7), queda garantizada por la propia continuidad de las derivadas de $\varphi_q(\mathbf{x})$, ya que $\varphi_q(\mathbf{x}) \in \mathcal{C}^1$.

Para demostrar que $\dot{V}(\mathbf{x})$ es definida negativa, ecuación (5.7), se reescribe $V(\mathbf{x})$ como

$$V(\mathbf{x}) = \sum_{q=1}^s (\varphi_q(\mathbf{x}) \mathbf{f}(\mathbf{x})^T \mathbf{P}_q \mathbf{f}(\mathbf{x})), \quad (5.20)$$

y derivando se obtiene que:

$$\dot{V}(\mathbf{x}) = \frac{dV(\mathbf{x})}{dt} = \sum_{q=1}^s \left(\varphi_q(\mathbf{x}) \frac{d}{dt} (\mathbf{f}(\mathbf{x})^T \mathbf{P}_q \mathbf{f}(\mathbf{x})) + \frac{d\varphi_q(\mathbf{x})}{dt} (\mathbf{f}(\mathbf{x})^T \mathbf{P}_q \mathbf{f}(\mathbf{x})) \right). \quad (5.21)$$

Puesto que,

$$\frac{d(\mathbf{f}(\mathbf{x})^T \mathbf{P}_q \mathbf{f}(\mathbf{x}))}{dt} = \frac{d\mathbf{f}(\mathbf{x})^T}{dt} \mathbf{P}_q \mathbf{f}(\mathbf{x}) + \mathbf{f}(\mathbf{x})^T \frac{d\mathbf{P}_q}{dt} \mathbf{f}(\mathbf{x}) + \mathbf{f}(\mathbf{x})^T \mathbf{P}_q \frac{d\mathbf{f}(\mathbf{x})}{dt}, \quad (5.22)$$

y

$$\frac{d\mathbf{f}(\mathbf{x})}{dt} = \frac{d\mathbf{f}(\mathbf{x})}{d\mathbf{x}} \dot{\mathbf{x}} = \mathbf{J}(\mathbf{x}) \mathbf{f}(\mathbf{x}), \quad (5.23)$$

entonces

$$\frac{d(\mathbf{f}(\mathbf{x})^T \mathbf{P}_q \mathbf{f}(\mathbf{x}))}{dt} = (\mathbf{J}(\mathbf{x}) \mathbf{f}(\mathbf{x}))^T \mathbf{P}_q \mathbf{f}(\mathbf{x}) + \mathbf{f}(\mathbf{x})^T \frac{d\mathbf{P}_q}{dt} \mathbf{f}(\mathbf{x}) + \mathbf{f}(\mathbf{x})^T \mathbf{P}_q \mathbf{J}(\mathbf{x}) \mathbf{f}(\mathbf{x}). \quad (5.24)$$

Teniendo en cuenta que la matrices $\mathbf{P}_q$ son independientes del tiempo y del vector de estado, esto es:

$$\frac{d\mathbf{P}_q}{dt} = 0, \quad (5.25)$$

y que la traspuesta de un producto de matrices es igual al producto de las traspuestas en orden permutado, es decir:

$$(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T, \quad (5.26)$$

la ecuación (5.24) puede reescribirse como:

$$\begin{aligned} \frac{d(\mathbf{f}(\mathbf{x})^T \mathbf{P}_q \mathbf{f}(\mathbf{x}))}{dt} &= \mathbf{f}(\mathbf{x})^T \mathbf{J}(\mathbf{x})^T \mathbf{P}_q \mathbf{f}(\mathbf{x}) + \mathbf{f}(\mathbf{x})^T \cancel{\frac{d\mathbf{P}_q}{dt}} \mathbf{f}(\mathbf{x}) + \mathbf{f}(\mathbf{x})^T \mathbf{P}_q \mathbf{J}(\mathbf{x}) \mathbf{f}(\mathbf{x}) \\ &= \mathbf{f}(\mathbf{x})^T (\mathbf{J}(\mathbf{x})^T \mathbf{P}_q + \mathbf{P}_q \mathbf{J}(\mathbf{x})) \mathbf{f}(\mathbf{x}). \end{aligned} \quad (5.27)$$

Por otro lado,

$$\frac{d\varphi_q(\mathbf{x})}{dt} = \frac{d\varphi_q(\mathbf{x})}{d\mathbf{x}} \dot{\mathbf{x}} = \frac{d\varphi_q(\mathbf{x})}{d\mathbf{x}} \mathbf{f}(\mathbf{x}). \quad (5.28)$$

Sustituyendo las expresiones (5.27) y (5.28) en (5.21):

$$\dot{V}(\mathbf{x}) = \sum_{q=1}^s \left(\varphi_q(\mathbf{x}) \mathbf{f}(\mathbf{x})^T (\mathbf{J}(\mathbf{x})^T \mathbf{P}_q + \mathbf{P}_q \mathbf{J}(\mathbf{x})) \mathbf{f}(\mathbf{x}) + \left(\frac{d\varphi_q(\mathbf{x})}{d\mathbf{x}} \mathbf{f}(\mathbf{x}) \right) (\mathbf{f}(\mathbf{x})^T \mathbf{P}_q \mathbf{f}(\mathbf{x})) \right) \quad (5.29)$$

Como las funciones $\varphi_q(\mathbf{x})$ son funciones escalares, pueden permutar su posición con $\mathbf{f}(\mathbf{x})^T$:

$$\dot{V}(\mathbf{x}) = \sum_{q=1}^s \left(\mathbf{f}(\mathbf{x})^T \varphi_q(\mathbf{x}) (\mathbf{J}(\mathbf{x})^T \mathbf{P}_q + \mathbf{P}_q \mathbf{J}(\mathbf{x})) \mathbf{f}(\mathbf{x}) + \left(\frac{d\varphi_q(\mathbf{x})}{d\mathbf{x}} \mathbf{f}(\mathbf{x}) \right) (\mathbf{f}(\mathbf{x})^T \mathbf{P}_q \mathbf{f}(\mathbf{x})) \right) \quad (5.30)$$

Lo mismo sucede con el producto $\frac{d\varphi_q(\mathbf{x})}{d\mathbf{x}} \mathbf{f}(\mathbf{x})$, ya que

$$\frac{d\varphi_q(\mathbf{x})}{d\mathbf{x}} = \left(\frac{\partial \varphi_q(\mathbf{x})}{\partial x_1}, \frac{\partial \varphi_q(\mathbf{x})}{\partial x_2}, \dots, \frac{\partial \varphi_q(\mathbf{x})}{\partial x_n} \right)_{(1 \times n)} \quad (5.31)$$

y

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_n \end{pmatrix}_{(n \times 1)}, \quad (5.32)$$

de forma que $\dot{V}(\mathbf{x})$ puede escribirse como

$$\dot{V}(\mathbf{x}) = \sum_{q=1}^s \left(\mathbf{f}(\mathbf{x})^T \varphi_q(\mathbf{x}) (\mathbf{J}(\mathbf{x})^T \mathbf{P}_q + \mathbf{P}_q \mathbf{J}(\mathbf{x})) \mathbf{f}(\mathbf{x}) + \mathbf{f}(\mathbf{x})^T \frac{d\varphi_q(\mathbf{x})}{d\mathbf{x}} \mathbf{f}(\mathbf{x}) \mathbf{P}_q \mathbf{f}(\mathbf{x}) \right). \quad (5.33)$$

Finalmente, sacando $\mathbf{f}(\mathbf{x})^T$ como factor común a la izquierda, y $\mathbf{f}(\mathbf{x})$ a la derecha y extrayendo dichas funciones del sumatorio, ya que no dependen del índice q , se obtiene la expresión final de $\dot{V}(\mathbf{x})$:

$$\dot{V}(\mathbf{x}) = \mathbf{f}(\mathbf{x})^T \sum_{q=1}^s \left(\varphi_q(\mathbf{x}) (\mathbf{J}(\mathbf{x})^T \mathbf{P}_q + \mathbf{P}_q \mathbf{J}(\mathbf{x})) + \frac{d\varphi_q(\mathbf{x})}{d\mathbf{x}} \mathbf{f}(\mathbf{x}) \mathbf{P}_q \right) \mathbf{f}(\mathbf{x}). \quad (5.34)$$

Dada $\mathbf{F}(\mathbf{x})$ según se define en la ecuación (5.17), la ecuación anterior puede ser escrita como:

$$\dot{V}(\mathbf{x}) = \mathbf{f}(\mathbf{x})^T \mathbf{F}(\mathbf{x}) \mathbf{f}(\mathbf{x}). \quad (5.35)$$

Por lo tanto, queda demostrado que si $\mathbf{F}(\mathbf{x})$ es definida negativa, $\dot{V}(\mathbf{x})$ también lo es, y $V(\mathbf{x})$ es una función de Lyapunov del sistema (5.16). $\square$

Nota. Obviamente, si la dinámica de la planta sólo se conoce a través de su modelo borroso obtenido mediante datos de entrada/salida, en ningún caso será posible garantizar la estabilidad asintótica global del sistema, ya que se desconoce el comportamiento de éste fuera del universo de discurso de sus variables. Así pues, la estabilidad asintótica sólo podrá garantizarse para aquellas trayectorias que cumplan el teorema 5.7 y permanezcan íntegramente dentro del universo de discurso. Esto es:

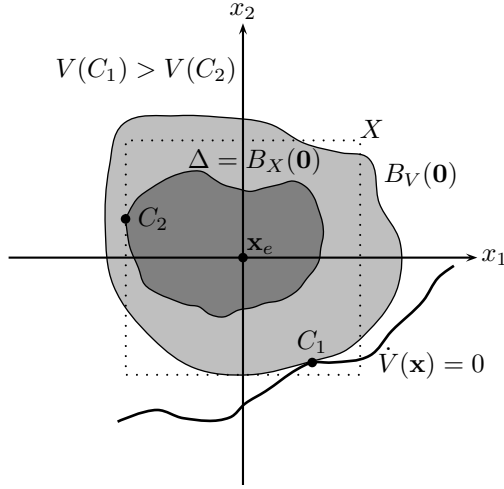
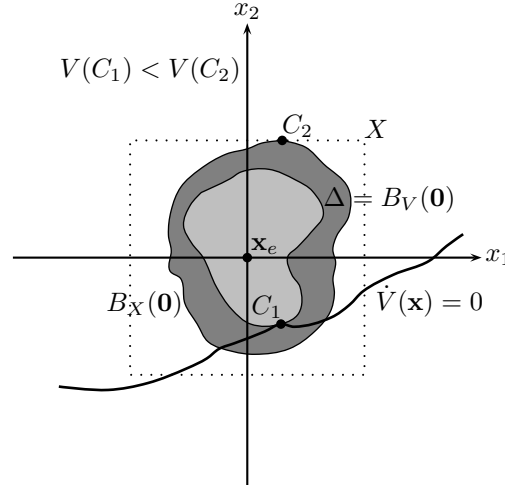
Sea X el universo de discurso de las variables de estado y ∂X su borde, y sean $C_1, C_2 \in \mathbb{R}^+$, de forma que $C_1 = \min\{V(\mathbf{x}) \mid \dot{V}(\mathbf{x}) = 0\}$ y $C_2 = \min\{V(\mathbf{x}) \mid \mathbf{x} \in \partial X\}$. La estimación más conservadora sobre la región de atracción Δ que contiene el estado de equilibrio situado en el origen, es aquella región contenida en el universo de discurso de las variables de estado, $\Delta \subseteq X$, en la que $V(\mathbf{x})$ se comporta como una función de Lyapunov y que satisface la ecuación (5.36).

$$\Delta \subseteq \{\mathbf{x} \in X \mid V(\mathbf{x}) < \min\{C_1, C_2\}\} \quad (5.36)$$

Nótese que el signo menor estricto de la ecuación (5.36) implica que la región de atracción no incluye su propio borde. Esto debe ser necesariamente así, ya que una trayectoria que se iniciase en el borde podría no converger hacia $\mathbf{x}_e$, por ejemplo, saliéndose del universo de discurso X .

Si el teorema 5.7 se cumple en todo el universo de discurso de las variables de estado, C_1 quedará fuera de X , por lo que la región de atracción vendrá determinada necesariamente por la curva de nivel $V(C_2)$.

En las figuras 5.4 y 5.5 se muestran gráficamente las dos situaciones posibles. En el primer caso mostrado en la figura 5.4, la región de cumplimiento del teorema 5.7, $B_V(\mathbf{0})$, no está estrictamente contenida en el universo de discurso X , es decir, $V(C_1) > V(C_2)$, por lo que la región de atracción Δ está determinada por todos los puntos del interior de la curva de nivel $V(C_2)$. Nótese que las trayectorias originadas en el exterior de dicha curva pueden salirse del universo de discurso de $\mathbf{x}$ y, por lo tanto, tener un comportamiento impredecible. En el segundo caso, mostrado en la figura 5.5, el teorema 5.7 se cumple en una región $B_X(\mathbf{0})$ estrictamente contenida en el universo de discurso X , es decir,

**Figura 5.4:** *Región de atracción, caso 1.***Figura 5.5:** *Región de atracción, caso 2.*

$V(C_1) < V(C_2)$, por lo que la región de atracción Δ está formada por todos los puntos que cumplen el teorema, o lo que es lo mismo, por todos los puntos del interior de la curva de nivel $V(C_1)$. Nótese que en el exterior y en el borde de dicha curva hay puntos donde $V(\mathbf{x})$ no es una función de Lyapunov válida, ya que $\dot{V}(\mathbf{x}) \geq 0$.

El teorema 5.7 demuestra que si la matriz $\mathbf{F}(\mathbf{x})$ dada por la expresión (5.17) es definida negativa en un entorno del origen de coordenadas, éste es asintóticamente estable. Por lo tanto, el diseño del controlador borroso consiste en determinar un conjunto de parámetros adaptables del mismo, tanto antecedentes como consecuentes, y un conjunto de regiones Θ_q , de forma que $\mathbf{F}(\mathbf{x})$ sea definida negativa en un entorno suficientemente amplio alrededor del origen. La solución de la condición anterior no es un problema trivial, ya que en un modelo borroso intervienen gran cantidad de parámetros: el tipo de cada función de pertenencia y los parámetros que la definen, el tipo de consecuente y el valor de sus parámetros, el número de reglas, etc. Además, no se ha establecido ninguna limitación en las dimensiones del sistema, con lo que el vector de parámetros ajustables podría tener un tamaño considerable. A todo esto es necesario añadir que durante el propio proceso de diseño hay que establecer las matrices $\mathbf{P}_q$, así como las regiones Θ_q a partir de sus funciones de pertenencia $\varphi_q(\mathbf{x})$, pero a priori no se sabe cuántas regiones se deben definir ni se dispone de información para su determinación.

Para obtener la matriz $\mathbf{F}(\mathbf{x})$ definida en el teorema 5.7 es necesario calcular previamente el modelo borroso de estado en lazo cerrado, $\mathbf{f}(\mathbf{x})$, la matriz jacobiana del sistema, $\mathbf{J}(\mathbf{x})$, y la derivada de la función de definición de las regiones con respecto a las variables de estado $d\varphi_q(\mathbf{x})/d\mathbf{x}$. El modelo borroso en lazo cerrado se puede calcular mediante la aplicación del algoritmo 3.5, y la matriz jacobiana a través del algoritmo 4.5, mientras que el cálculo de $d\varphi_q(\mathbf{x})/d\mathbf{x}$ dependerá de la definición que se utilice para $\varphi_q(\mathbf{x})$.

La pertenencia de cada punto del espacio de estado a cada una de las regiones Θ_q puede hacerse de forma borrosa, esto es, cada punto puede pertenecer a una o varias regiones con

una grado de pertenencia comprendido entre 0 y 1, con lo cual, la partición del espacio de estado será más flexible que si se hace de forma clásica (no borrosa). Para ello se define la función de pertenencia a la región Θ_q , como

$$\varphi_q(\mathbf{x}) = \prod_{i=1}^n \mu_{iq}(x_i, \Psi_{iq}), \quad (5.37)$$

donde $\mu_{iq}(x_i, \Psi_{iq}) \in [0, 1]$ es una función de pertenencia de parámetros adaptables Ψ_{iq} , con $i = 1..n$ y $q = 1..s$, siendo n el orden del sistema y s el número de regiones definidas. La función $\varphi_q(\mathbf{x})$ es derivable respecto a $\mathbf{x}$ si $\mu_{iq}(x_i)$ lo es $\forall i, q$, y cumple $\varphi_q(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}^1$ tal como exige el teorema.

Observando la similitud de la expresión (5.37) con el grado de activación de las reglas del controlador borroso, (3.37), es fácil deducir la expresión (5.38) a partir de (4.94). De esta forma ya se tienen todos los elementos necesarios para evaluar (5.17).

$$\frac{\partial \varphi_q(\mathbf{x})}{\partial x_i} = \frac{\partial \mu_{iq}(x_i)}{\partial x_i} \prod_{k=1, k \neq i}^n \mu_{kq}(x_k) \quad (5.38)$$

Al aplicar la teoría de Lyapunov a través del teorema 5.7 se garantiza la generalidad de la metodología, ya que no se realizan linealizaciones ni simplificaciones de ningún tipo sobre el sistema, y la condición que se impone es lo suficientemente flexible como para que pueda ser satisfecha por sistemas complejos [170].

Una vez concluido el proceso de diseño del controlador, las mismas herramientas utilizadas para el estudio dinámico de la planta pueden emplearse para comprobar el funcionamiento del sistema de control borroso de lazo cerrado.

5.4.4. Algoritmo de diseño

En este apartado se presenta un algoritmo para el diseño del controlador borroso, de forma que, haciendo uso de la metodología presentada anteriormente, se garantice la estabilidad del sistema de control en lazo cerrado.

Con objeto de facilitar el diseño del controlador borroso, se divide el proceso de síntesis en tres fases generales: una *fase de inicialización*, una *fase de diseño* y una *fase final* de ajuste del estado de equilibrio [170]. En la *fase de inicialización* se definen las regiones Θ_q mediante sus correspondientes funciones de pertenencia $\varphi_q(\mathbf{x})$, y se inicializan tanto las matrices $\mathbf{P}_q$ como el controlador borroso: número de reglas, antecedentes y consecuentes. Durante la *fase de diseño* se ajustan los parámetros del controlador, así como las matrices $\mathbf{P}_q$ hasta que se cumple el teorema 5.7, lo que garantiza la estabilidad asintótica del sistema de control borroso en lazo cerrado. Por último, en la *fase final* se ajusta adecuadamente el estado de equilibrio del sistema en lazo cerrado.

Fase de inicialización

Para el caso más general, el modelo matemático de la planta será desconocido, y toda la información disponible sobre la dinámica del sistema residirá en el modelo borroso que

se realice de la planta. Por lo tanto, parece razonable inicializar el controlador borroso utilizando dicho conocimiento. Para ello, las reglas correspondientes a cada una de las m salidas del controlador se formulan empleando toda la base de conocimientos de la planta, es decir, cada una de las m salidas del controlador se calcularán mediante la inferencia de todas las reglas de la planta para cada una de las i -ésimas ecuaciones de estado. De esta forma, la j -ésima salida del controlador será inferida por $N_j = M_1 + M_2 + \dots + M_i$ reglas. En principio, este procedimiento puede parecer inadecuado porque provoca que el controlador tenga un gran número de reglas, $m(M_1 + M_2 + \dots + M_i)$, sin embargo, este hecho no representa un problema muy serio, ya que existen bastantes algoritmos en la literatura que pueden ser utilizados en el caso de que sea necesario reducir el número de reglas del controlador [264, 266, 267, 338, 339].

Una vez que se han definido los antecedentes del controlador borroso, éstos permanecerán constantes a lo largo de todo el proceso de diseño, de esta forma únicamente se ajustarán los consecuentes de las reglas.

Otro punto que debe resolverse durante la fase de inicialización es la definición de las regiones Θ_q involucradas en el teorema 5.7. El grado de pertenencia a una regla borrosa representa una región del espacio de estado, la región de cumplimiento de dicha regla, por lo que las reglas que se han definido para el controlador borroso pueden utilizarse para establecer las regiones Θ_q del teorema. De esta forma, el espacio de estado queda dividido en tantas regiones como reglas posea el controlador, es decir:

$$s = M_1 + M_2 + \dots + M_i. \quad (5.39)$$

Igualmente, para definir las funciones de pertenencia $\varphi_q(\mathbf{x})$ se puede emplear el grado de activación de las reglas del controlador borroso, $\omega_j^r(\mathbf{x})$, cuyo cálculo se realiza a partir de la ecuación (3.37), repetida a continuación por claridad:

$$\omega_j^r = \prod_{k=1}^n \mu_{kj}^r(x_k, \Phi_{kj}^r) \quad (3.37)$$

Nótese que con la inicialización propuesta anteriormente para el controlador borroso, sus antecedentes cumplen $\omega_1^r(\mathbf{x}) = \omega_2^r(\mathbf{x}) = \dots = \omega_m^r(\mathbf{x})$. Esto es, para cualquier valor de j , por ejemplo $j = 1$, $\varphi_q(\mathbf{x})$ puede obtenerse como

$$\varphi_q(\mathbf{x}) = \omega_j^q(\mathbf{x}), \quad (5.40)$$

o lo que es equivalente,

$$\Psi_{iq} = \Phi_{ij}^q, \quad (5.41)$$

para cualquier valor de j entre 1 y m .

El último paso que es necesario realizar durante la primera fase del algoritmo de diseño es la inicialización de las matrices $\mathbf{P}_q$. Puesto que se requiere que estas matrices sean definidas positivas, un posible valor para su inicialización es la matriz identidad, esto es:

$$\mathbf{P}_q = \mathbf{I}, \forall q = 1..s. \quad (5.42)$$

La fase de inicialización del algoritmo de diseño se puede sintetizar en el algoritmo mostrado en la figura 5.6.

```

1: Inicialización del controlador:
2: Para  $j = 1..m$  hacer
3:    $r = 1$ 
4:   Para  $i = 1..n$  hacer
5:     Para  $l = 1..M_i$  hacer
6:       Para  $k = 1..n$  hacer
7:          $C_{kj}^r = A_{ki}^l$  {Inicialización de los antecedentes  $\Rightarrow \omega_j^r(\mathbf{x})$ }
8:          $c_{kj} = 0$  {Inicialización de los consecuentes}
9:       Fin Para
10:       $c_{0j} = 0$  {Inicialización de los términos afines de los consecuentes}
11:       $r = r + 1$ 
12:    Fin Para
13:  Fin Para
14: Fin Para
15: Inicialización de las matrices  $P_q$  y definición de las regiones  $\Theta_q$ :
16: Para  $q = 1..s$  hacer
17:    $\mathbf{P}_q = \mathbf{I}$ 
18:    $j = 1$ 
19:   Para  $i = 1..n$  hacer
20:      $\Psi_{iq} = \Phi_{ij}^q$  {Definición de las regiones  $\Theta_q$ , resultando  $\varphi_q(\mathbf{x}) = \omega_1^q(\mathbf{x})$ }
21:   Fin Para
22: Fin Para

```

Figura 5.6: *Algoritmo de inicialización.*

Fase de diseño

Durante la fase de diseño deben obtenerse unos valores para los parámetros de los consecuentes del controlador borroso y para las matrices $\mathbf{P}_q$ que impliquen el cumplimiento del teorema 5.7.

En todo momento es imprescindible garantizar que las matrices $\mathbf{P}_q$ sean definidas positivas. Para ello puede emplearse la descomposición de Cholesky, descrita más detalladamente en el apéndice C, la cual establece que

$$\mathbf{P}_q = \mathbf{T}_q^T \mathbf{T}_q, \quad (5.43)$$

es una matriz simétrica definida positiva, si $\mathbf{T}_q$ es una matriz triangular superior donde todos los elementos de su diagonal son positivos, es decir, si $\mathbf{T}_q$ tiene la forma dada en la ecuación (5.44).

$$\mathbf{T}_q = \begin{pmatrix} t_{11}^q & t_{12}^q & \cdots & t_{1n}^q \\ 0 & t_{22}^q & \cdots & t_{2n}^q \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & t_{nn}^q \end{pmatrix}, \quad t_{ii}^q > 0, \quad i = 1..n. \quad (5.44)$$

La utilización de la descomposición de Cholesky permite garantizar que $\mathbf{P}_q$ es definida positiva de forma sencilla. Además, si se utiliza la matriz $\mathbf{T}_q$ de la descomposición de Cholesky como matriz ajustable durante el desarrollo del algoritmo, el número de parámetros ajustables se reduce desde los n^2 parámetros que conforman todos los elementos de $\mathbf{P}_q$, a los $n(n+1)/2$ parámetros que es necesario establecer para construir la matriz $\mathbf{T}_q$, siendo n el orden del sistema.

El teorema 5.7 debe satisfacerse de forma continua en una región $B_V(\mathbf{0})$ alrededor del estado de equilibrio, es decir, en los infinitos puntos contenidos en la región $B_V(\mathbf{0})$. Sin embargo, desde el punto de vista práctico es necesario discretizar dicho espacio y evaluar el cumplimiento del teorema en un número finito de puntos. Si el hecho de que $\mathbf{F}(\mathbf{x})$ sea definida negativa se evalúa de forma discreta, el cumplimiento de la condición $\mathbf{F}(\mathbf{x}) < 0$ no es suficiente para garantizar que $V(\mathbf{x})$ sea una función de Lyapunov válida para el sistema, y por lo tanto, que el origen sea asintóticamente estable. No obstante, se demuestra [340] que si se cumple $\mathbf{F}(\mathbf{x}) < 0$ en un conjunto finito suficientemente denso de puntos y $\mathbf{f}(\mathbf{x})$, $\varphi_q(\mathbf{x})$ y $d\varphi_q(\mathbf{x})/d\mathbf{x}$ son funciones acotadas y localmente funciones de Lipschitz⁵, entonces $V(\mathbf{x})$ es una función de Lyapunov válida para todo $\mathbf{x} \in B_V(\mathbf{0})$.

Para el diseño del controlador borroso se ha utilizado un algoritmo iterativo que impone el cumplimiento del teorema 5.7 en una región lo más amplia posible alrededor del estado de equilibrio. Si como algoritmo de diseño se utiliza un algoritmo de minimización, se requiere disponer de una función a minimizar que represente el cumplimiento de dicho

⁵Una función $f : M \rightarrow N$ se dice que es de Lipschitz en una región R , si existe una constante $L > 0$, llamada constante de Lipschitz, tal que para cualesquiera que sean x e y en R , se cumple que $\|f(x) - f(y)\| \leq L(\|x - y\|)$.

```

1: Objetivo = 0
2: Para todo los Puntos hacer
3:   Calcular  $\mathbf{f}(\mathbf{x})$ , mediante el algoritmo 3.5
4:   Calcular  $\mathbf{J}(\mathbf{x})$ , mediante el algoritmo 4.5
5:   Calcular  $\varphi_q(\mathbf{x})$ , ecuación (5.40)
6:   Calcular  $d\varphi_q(\mathbf{x})/d\mathbf{x}$ , ecuación (5.38)
7:   Calcular  $\mathbf{F}(\mathbf{x})$ , ecuación (5.17)
8:   Si  $\text{máximo\_autovalor}(\mathbf{F}(\mathbf{x})) > -\varepsilon$  entonces
9:     Objetivo = Objetivo +  $\text{máximo\_autovalor}(\mathbf{F}(\mathbf{x})) + \varepsilon$ 
10:    { $\varepsilon > 0$  es valor pequeño que garantiza que  $\mathbf{F}(\mathbf{x})$  sea estrictamente definida negativa.}
11:   Fin Si
12: Fin Para

```

Figura 5.7: *Función objetivo.*

```

1: Mientras (Objetivo > 0) hacer
2:   Algoritmo global para la minimización de la función objetivo de la figura 5.7
3:   Si (Objetivo > 0) entonces
4:     Algoritmo local para la minimización de la función objetivo de la figura 5.7
5:   Fin Si
6: Fin Mientras

```

Figura 5.8: *Algoritmo híbrido de minimización.*

teorema. En la figura 5.7 se muestra un algoritmo para el cálculo de la función objetivo, que hace uso del máximo autovalor de la matriz $\mathbf{F}(\mathbf{x})$, siempre que éste sea positivo, para medir el cumplimiento del teorema 5.7. El mínimo valor de la función objetivo propuesta, $\text{Objetivo} = 0$, se produce cuando el mayor de los autovalores de $\mathbf{F}(\mathbf{x})$ es menor que $-\varepsilon$, siendo ε un número positivo pequeño. Por lo tanto, cuando la función objetivo es nula, $\mathbf{F}(\mathbf{x})$ es definida negativa y el teorema queda satisfecho en todos los puntos evaluados, resultando un sistema asintóticamente estable.

Dada la complejidad del problema y el gran número de parámetros que deben ser ajustados, además de la alta no linealidad que puede llegar a presentar la función a minimizar, un algoritmo numérico difícilmente convergería hacia una solución válida. Por lo tanto, la utilización de un algoritmo de minimización global y alta capacidad de exploración, como por ejemplo un algoritmo genético, puede resultar muy interesante. No obstante, los algoritmos globales de minimización genéticos suelen presentar una velocidad de convergencia bastante lenta, incluso cuando parten de un punto cercano a una solución, en contraposición con los algoritmos numéricos, cuya velocidad de convergencia puede ser muy alta si parten de una posición inicial adecuada. Teniendo en cuenta ésto, se ha empleado el algoritmo híbrido global-local de minimización mostrado en la figura 5.8.

Los algoritmos de minimización global y local empleados en el algoritmo híbrido de la figura 5.8 no necesitan de un diseño específico, por lo que pueden utilizarse cualquiera de los múltiples algoritmos existentes en la literatura de eficiencia ya contrastada. Más concretamente, en la implementación desarrollada en este trabajo se ha empleado un algoritmo

genético básico [341] para la minimización global, y el algoritmo numérico de minimización de Nelder–Mead [342], capaz de obtener buenos resultados incluso en problemas altamente no lineales, para la minimización local.

Fase final

Al no aplicarse de forma continua la condición de estabilidad establecida en el teorema 5.7, el estado de equilibrio obtenido puede estar desplazado del origen. Para corregir esta situación, una vez concluida la fase de ajuste se ejecuta una nueva iteración del algoritmo híbrido de minimización global–local, pero para minimizar la función F_{eq} dada por la expresión (5.45). El valor de *Objetivo* viene determinado por el algoritmo 5.7, mientras que β representa una constante positiva cuyo valor ha de ser bastante alto para garantizar que la estabilidad del sistema conseguida en la fase de ajuste no se vea afectada por este ajuste del estado de equilibrio.

$$F_{eq} = \|\mathbf{f}(\mathbf{0})\|_1 + \beta \cdot \text{Objetivo} \quad (5.45)$$

Un valor apropiado para el factor β es aquel que penaliza suficientemente cualquier valor distinto de cero de *Objetivo* en la expresión (5.45). Por lo tanto, el ajuste del controlador en esta última fase se hace sólo y exclusivamente minimizando la distancia⁶ del estado de equilibrio del sistema con respecto al origen de coordenadas del mismo, es decir, minimizando $\|\mathbf{f}(\mathbf{0})\|_1$. El valor adecuado de β puede variar en función del sistema que se diseñe y del número de puntos utilizados, aunque un valor de $\beta = 10^6$ suele ser suficiente en la mayoría de los casos.

5.5. Conclusiones y comentarios bibliográficos

Aunque en los inicios de la lógica borrosa se criticó bastante su escasa formalización, como puede comprobarse en [306], a lo largo de este capítulo se ha visto que desde entonces se ha avanzado mucho en este aspecto. En las referencias [100], [121], [175], [227], de [307] a [332], [336] y [337] pueden estudiarse múltiples técnicas de análisis y síntesis de controladores borrosos, basadas en su mayor parte en la teoría de control no lineal.

La capacidad innata de la lógica borrosa para transferir el conocimiento de un experto a un controlador borroso que opere de forma autónoma fue una de las características más atrayentes en sus inicios. En la segunda sección del capítulo se ha visto cómo diseñar un controlador borroso mediante síntesis lingüística en base al conocimiento de un experto. En las referencias [28], [171], [263] y [334] se puede estudiar más a fondo esta técnica y consultar algunos ejemplos prácticos.

En este capítulo se ha propuesto una metodología formal y completamente general que permite la síntesis de sistemas de controladores borrosos estables por diseño, es decir, donde la estabilidad del sistema en lazo cerrado queda garantizada por el propio proceso

⁶Como medida de distancia se propone emplear la norma 1, $\|\cdot\|_1$, ya que su cálculo es más eficiente que el de otras normas. En cualquier caso, si se desea, es posible utilizar cualquier otra medida de distancia, como la norma euclídea, $\|\cdot\|_2$, o la norma infinito, $\|\cdot\|_\infty$.

de diseño del controlador borroso. Esta metodología se basa en 3 pasos: 1) identificación de la planta a partir de datos de entrada/salida; 2) extracción de información de la dinámica de la planta mediante la obtención de sus estados de equilibrio, el estudio de la estabilidad local de éstos mediante la linealización del modelo borroso de la planta; y 3), diseño de un controlador borroso que garantice la estabilidad de lazo cerrado del sistema. Los dos primeros puntos de esta metodología han sido fruto del estudio realizado en los capítulos anteriores, mientras que el diseño del controlador se basa en el teorema de estabilidad 5.7, propuesto y demostrado en este mismo capítulo y en el artículo [170]. La metodología propuesta en este capítulo puede consultarse en las referencias [168], [170], [233] y [293]. Finalmente se propone un algoritmo de diseño para, a partir del teorema propuesto, diseñar un controlador que asegure la estabilidad del sistema de control borroso en lazo cerrado. El algoritmo propuesto también permite el diseño en base al teorema de Krasovskii, ya que es una generalización de éste. Puede ampliarse información sobre la metodología de diseño basada en el teorema de Krasovskii en las referencias [168], [233], [293] y [294].

En las referencias [210] a [217] se demuestra que los modelos borrosos de tipo TSK son aproximadores universales de cualquier función y de su derivada, lo que permite que la metodología propuesta pueda considerarse de aplicación a cualquier planta.

Como base tanto para el algoritmo de diseño del controlador, como para la mejora del algoritmo de extracción de los estados de equilibrio de la planta, se utiliza la matriz jacobiana del sistema de control borroso. El cálculo de la matriz jacobiana se estudió en el capítulo 4 y puede consultarse también en las referencias [168], [231] y [290].

En las referencias [264] a [267], [338] y [339] pueden ser consultados distintos algoritmos de simplificación de modelos borrosos, lo cual puede ser útil si se desea reducir el número de reglas tanto de la planta como del controlador borroso.

Por último, como algoritmo de optimización global y local se ha propuesto la utilización de un algoritmo genético y un algoritmo numérico, respectivamente. En las referencias [341] puede estudiarse un algoritmo genético de optimización, y en [342] el algoritmo de minimización de Nelder–Mead.

Aplicaciones

ESTE capítulo aborda el análisis y diseño de sistemas de control borroso a partir de las técnicas presentadas en los capítulos anteriores. El capítulo se divide en tres secciones. La primera de ellas se dedica al control heurístico. En esta sección se diseña un controlador borroso para una grúa porta contenedores a partir del conocimiento de un experto, y se comprueba mediante simulación el correcto funcionamiento del mismo. La segunda sección está dedicada a la síntesis de controladores borrosos estables por diseño. En esa sección se realizan tres ejemplos que siguen la metodología de diseño presentada en el capítulo 5, esto es, partiendo de un conjunto de datos de entrada/salida se obtiene un modelo borroso de la planta, la cual se considera desconocida en todo momento. A partir del modelo borroso obtenido se calculan los estados de equilibrio de la planta, y se estudia la estabilidad local de los mismos. Posteriormente se diseña un controlador que garantice la estabilidad del sistema en lazo cerrado. Para comprobar el correcto funcionamiento del controlador borroso diseñado, se realiza una simulación del sistema y se obtienen los estados de equilibrio del sistema de control borroso en lazo cerrado, comprobando de nuevo su estabilidad local. Los tres ejemplos realizados en esta sección son: un ejemplo ilustrativo de múltiple entrada y múltiple salida (MIMO), que pretende facilitar la comprensión de la metodología de diseño propuesta; un péndulo, que representa un sistema no lineal con múltiples estados de equilibrio, cuyo objetivo es demostrar la validez de la fase de análisis y diseño; y un balancín, que representa un sistema no lineal de uso común en la literatura relacionada con sistemas no lineales. Finalmente, en la tercera y última sección se recogen las principales conclusiones de este capítulo.

6.1. Control heurístico

En esta sección se aborda el diseño de un controlador borroso basado en heurística para una grúa porta contenedores [171]. Para ello se emplea la metodología presentada en la sección 5.3, donde las reglas heurísticas son inferidas a partir del conocimiento de la dinámica de la planta y de su manejo por un operario experto.



Figura 6.1: *Grúa porta contenedores.*

6.1.1. Control borroso basado en heurística de una grúa porta contenedores

Las grúas porta contenedores, como la que se muestra en la figura 6.1, son usadas en muchos puertos para cargar y descargar contenedores hacia y desde barcos de transporte especiales. Los contenedores quedan fijados mediante cables flexibles montados en la cabeza de la grúa, lo que provoca que cuando éste se levanta y la cabeza de la grúa comienza a moverse, el contenedor empieza a balancearse. Este balanceo, si bien puede no ser un problema durante el transporte del contenedor, sí lo es para poder dejar caer el contenedor en el barco. Los gruistas expertos compensan el balanceo con avances y retrocesos a partir del control continuo de velocidad del motor de la cabeza de la grúa, hasta posicionar el contenedor en el sitio exacto.

La solución para la automatización de este problema de control no es trivial, ya que el comportamiento de la planta es sensiblemente no lineal y las perturbaciones tales como rachas de viento son imprevisibles. A pesar de esto, un operador experimentado es capaz de controlar la grúa basándose en su experiencia, sin necesidad de utilizar complejas ecuaciones diferenciales. Por lo tanto, este problema parece apropiado para ser abordado desde el punto de vista de la lógica borrosa mediante la incorporación del conocimiento de un experto, el gruista [171]. Para la realización del controlador borroso se ha optado por un modelo de tipo Mamdani, ver las secciones 2.3.5 y 3.2.2, y un enfoque de diseño basado en heurística, ver sección 5.3, inferida a partir de dos fuentes de información: conocimiento de la dinámica de la planta y conocimiento de su manejo por un operario experto.

Respecto del conocimiento de la planta, ésta puede verse de forma esquemática en la figura 6.2. Las variables que intervienen en el problema son las siguientes: J_1 , momento de inercia del motor del carro; b_1 , radio equivalente del tambor del motor del carro; J_2 ,

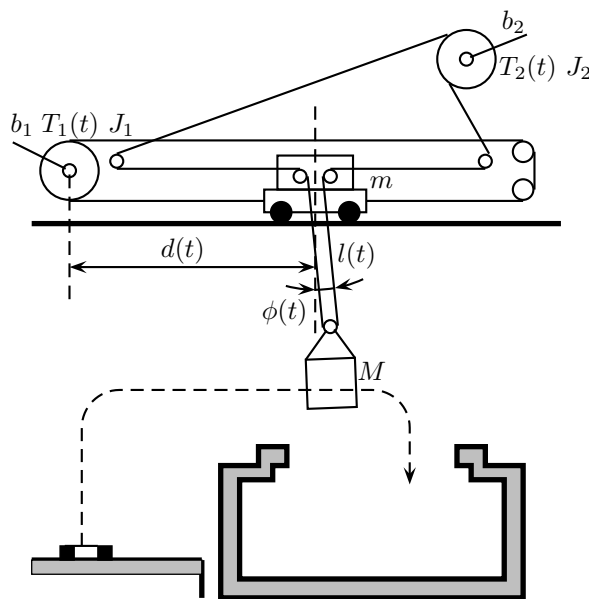


Figura 6.2: Representación esquemática de una grúa porta contenedores.

momento de inercia del motor de la grúa; b_2 , radio equivalente del tambor del motor de la grúa; m , masa del carro; M , masa de la carga del contenedor; y g , aceleración de la gravedad ($9,81\text{ms}^{-2}$). Las variables de salida del sistema son: $d(t)$, posición del carro; $l(t)$, longitud del cable; y $\phi(t)$, ángulo de balanceo de la carga. Junto a ellas, y respecto del problema del control, pueden ser consideradas sus velocidades respectivas, esto es, $\dot{d}(t)$, $\dot{l}(t)$ y $\dot{\phi}(t)$. Las entradas son: $T_1(t)$, el par generado por el motor del carro; y $T_2(t)$, el par generado por el motor de la grúa.

Diseño del controlador borroso

A partir del conocimiento de la planta se han definido los siguientes conjuntos borrosos tanto para la posición, d , como para la longitud, l : *origen*, *media*, *cerca*, *objetivo* y *corrección*, donde el valor *corrección* indica que la grúa se ha pasado del objetivo. Ha de tenerse en cuenta que aunque los conjuntos borrosos que definen d y l tienen los mismos términos lingüísticos, varían un poco sus valores, ver figuras 6.3 y 6.4.

Para la variable *ángulo*, ϕ , se han definido los siguientes conjuntos borrosos: *positivo_grande*, *positivo_pequeño*, *cero*, *negativo_pequeño* y *negativo_grande*, ver figura 6.5.

Finalmente, para las velocidades de las variables posición y longitud, $\dot{d}$ y $\dot{l}$ respectivamente, se han tomado los conjuntos: *neg_media*, *neg_pequeña*, *cero*, *pos_pequeña*, *pos_media* y *pos_grande*, ver figura 6.6.

Para las variables de control, se han elegido también funciones de pertenencia gaussianas para los siguientes conjuntos borrosos: *neg_alta*, *neg_media*, *cero*, *pos_media* y *pos_alta*, ver figuras 6.7 y 6.8 para T_1 y T_2 , respectivamente. Nótese que los conjuntos borrosos que

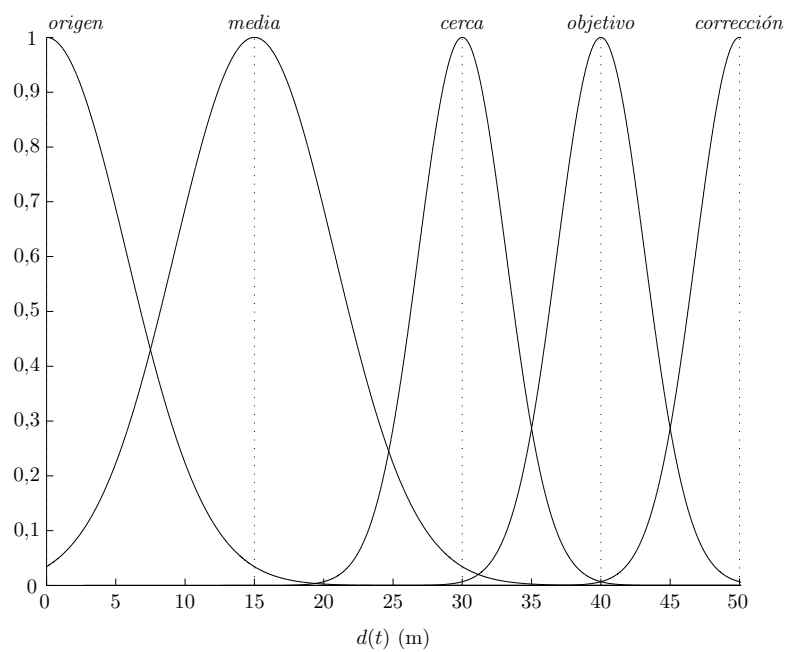


Figura 6.3: Funciones de pertenencia para la variable posición, d .

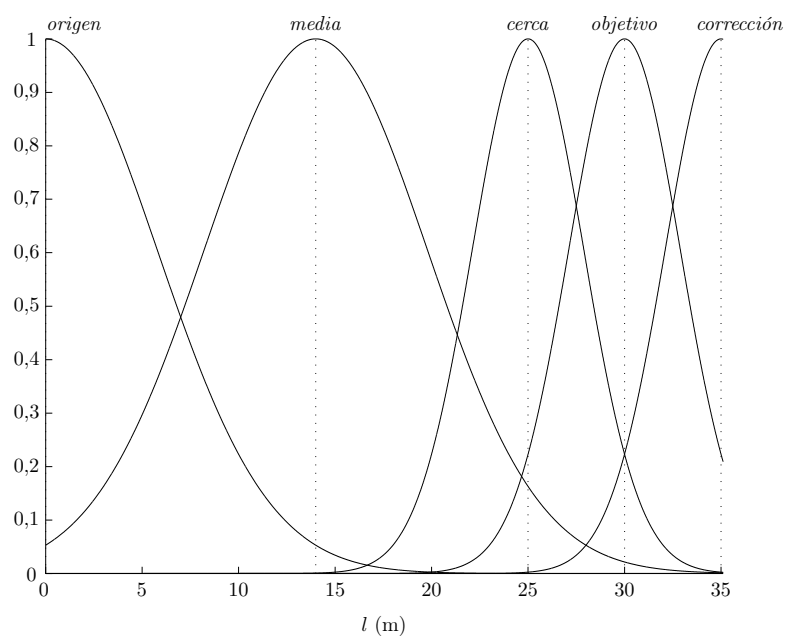


Figura 6.4: Funciones de pertenencia para la variable longitud del cable, l .

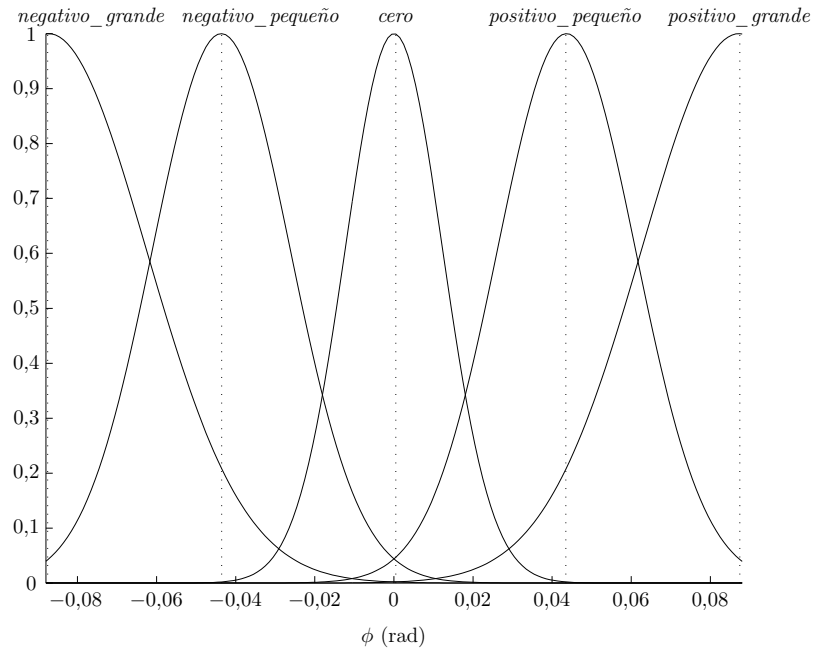


Figura 6.5: Funciones de pertenencia para la variable ángulo, ϕ .

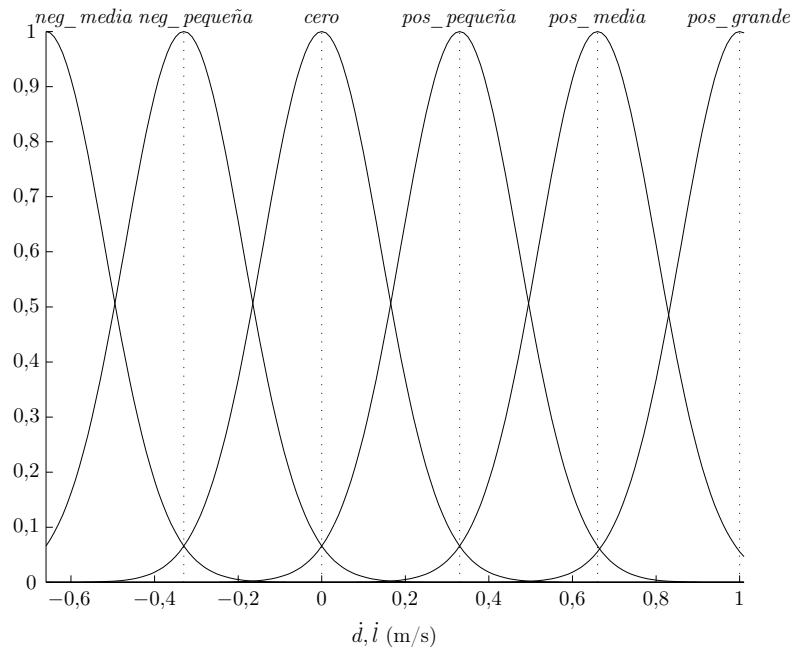


Figura 6.6: Funciones de pertenencia para las variables $\dot{d}$ y $\dot{l}$.

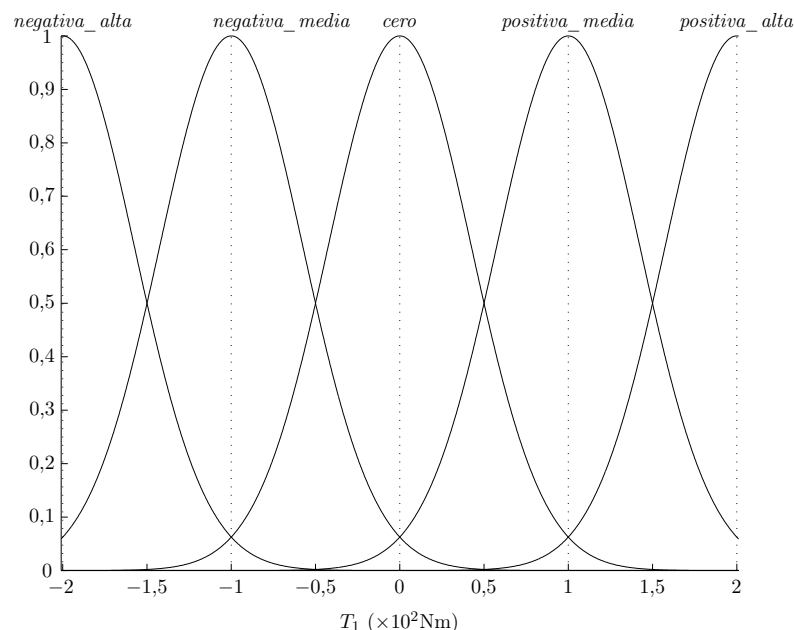


Figura 6.7: Funciones de pertenencia para T_1 .

definen los mismos términos lingüísticos de T_1 y T_2 tienen valores distintos, aunque se hayan utilizado los mismos términos lingüísticos para ambas variables.

Del conocimiento de la planta y de la observación de la operación del experto (gruista), se pueden deducir una serie de pautas que permiten definir una estrategia de control, por ejemplo, entre otras:

- Arrancar a media potencia.
- Durante el recorrido y hasta próximo a la posición final, la cabeza de la grúa puede ir por delante del contenedor.
- Aceleraciones bruscas enlentecen el proceso porque para equilibrar el contenedor hay que realizar varios cambios del sentido de la marcha continuos.
- Cuando la cabeza de la grúa esté cerca del objetivo hay que enlentecer la marcha de modo que al final el contenedor llegue casi sin oscilaciones.
- Cuando la cabeza de la grúa está en el objetivo y el cable no tiene balanceo, se para el motor del carro y se activa el de la polea de descenso.

A partir de, entre otras, estas consideraciones, y tras diferentes pasos de refinamiento, se define la siguiente base de reglas que generan las dos señales de control, T_1 y T_2 , que gobiernan, respectivamente, el motor de tracción del carro de la grúa sobre la horizontal, y el motor de enrollado/desenrollado del cable:

Para T_1 :

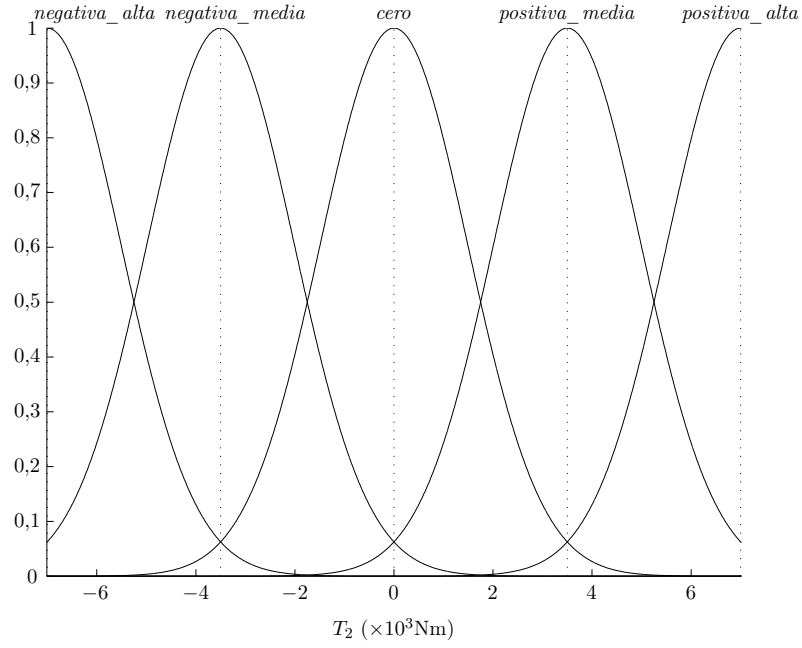


Figura 6.8: Funciones de pertenencia para T_2 .

- R1: SI l es *origen* y d es *origen* y $\dot{d}$ no es *pos_pequeña* ENTONCES T_1 es *pos_media*.
- R2: SI l es *origen* y d es *media* y $\dot{d}$ es *pos_grande* ENTONCES T_1 es *neg_media*.
- R3: SI l es *origen* y d es *media* y $\dot{d}$ es *pos_pequeña* ENTONCES T_1 es *cero*.
- R4: SI l es *origen* y d es *media* y $\dot{d}$ es *pos_pequeña* ENTONCES T_1 es *pos_media*.
- R5: SI l es *origen* y d es *cerca* y $\dot{d}$ es *pos_pequeña* ENTONCES T_1 es *neg_media*.
- R6: SI l es *origen* y d es *cerca* y $\dot{d}$ es *pos_pequeña* ENTONCES T_1 es *cero*.
- R7: SI l es *origen* y d es *cerca* y $\dot{d}$ es *cero* ENTONCES T_1 es *pos_media*.
- R8: SI l es *origen* y d es *cerca* y $\dot{d}$ es *neg_pequeña* ENTONCES T_1 es *pos_media*.
- R9: SI l es *origen* y d es *cerca* y $\dot{d}$ es *neg_media* ENTONCES T_1 es *pos_alta*.
- R10: SI d es *objetivo* y $\dot{d}$ es *pos_pequeña* ENTONCES T_1 es *pos_alta*.
- R11: SI d es *objetivo* y $\dot{d}$ es *pos_pequeña* ENTONCES T_1 es *neg_media*.
- R12: SI d es *objetivo* y $\dot{d}$ es *cero* ENTONCES T_1 es *cero*.
- R13: SI d es *objetivo* y $\dot{d}$ es *neg_pequeña* ENTONCES T_1 es *pos_media*.
- R14: SI d es *objetivo* y $\dot{d}$ es *neg_media* ENTONCES T_1 es *pos_alta*.
- R15: SI d es *corrección* y $\dot{d}$ es *pos_pequeña* ENTONCES T_1 es *neg_alta*.
- R16: SI d es *corrección* y $\dot{d}$ es *pos_pequeña* ENTONCES T_1 es *neg_alta*.

- R17: SI d es *corrección* y $\dot{d}$ es *cero* ENTONCES T_1 es *neg_media*.
 R18: SI d es *corrección* y $\dot{d}$ es *neg_pequeña* ENTONCES T_1 es *cero*.
 R19: SI d es *corrección* y $\dot{d}$ es *neg_media* ENTONCES T_1 es *pos_media*.
 R20: SI ϕ es *negativo_grande* ENTONCES T_1 es *neg_alta*.
 R21: SI ϕ es *positivo_grande* ENTONCES T_1 es *pos_alta*.

Para T_2 :

- R1: SI $\dot{l}$ es *pos_grande* ENTONCES T_2 es *neg_media*.
 R2: SI d es *origen* y l es *origen* ENTONCES T_2 es *cero*.
 R3: SI d es *origen* y l es *media* ENTONCES T_2 es *neg_media*.
 R4: SI d es *origen* y l es *cerca* ENTONCES T_2 es *neg_media*.
 R5: SI d es *origen* y l es *objetivo* ENTONCES T_2 es *neg_media*.
 R6: SI d es *origen* y l es *corrección* ENTONCES T_2 es *neg_media*.
 R7: SI d es *origen* y $\dot{l}$ es *neg_media* ENTONCES T_2 es *pos_media*.
 R8: SI d es *objetivo* y l es *origen* y x_5 es *cero* ENTONCES T_2 es *pos_media*.
 R9: SI d es *objetivo* y l es *media* y $\dot{l}$ es *pos_media* ENTONCES T_2 es *cero*.
 R10: SI d es *objetivo* y l es *media* y $\dot{l}$ es *pos_pequeña* ENTONCES T_2 es *pos_media*.
 R11: SI d es *objetivo* y l es *cerca* y $\dot{l}$ es *pos_media* ENTONCES T_2 es *neg_media*.
 R12: SI d es *objetivo* y l es *cerca* y $\dot{l}$ es *pos_pequeña* ENTONCES T_2 es *cero*.
 R13: SI d es *objetivo* y l es *cerca* y $\dot{l}$ es *cero* ENTONCES T_2 es *pos_media*.
 R14: SI d es *objetivo* y l es *cerca* y $\dot{l}$ es *neg_pequeña* ENTONCES T_2 es *pos_media*.
 R15: SI d es *objetivo* y l es *cerca* y $\dot{l}$ es *neg_media* ENTONCES T_2 es *pos_media*.
 R16: SI d es *objetivo* y l es *objetivo* y $\dot{l}$ es *pos_media* ENTONCES T_2 es *neg_media*.
 R17: SI d es *objetivo* y l es *objetivo* y $\dot{l}$ es *pos_pequeña* ENTONCES T_2 es *neg_media*.
 R18: SI d es *objetivo* y l es *objetivo* y $\dot{l}$ es *cero* ENTONCES T_2 es *cero*.
 R19: SI d es *objetivo* y l es *objetivo* y $\dot{l}$ es *neg_pequeña* ENTONCES T_2 es *pos_media*.
 R20: SI d es *objetivo* y l es *objetivo* y $\dot{l}$ es *neg_media* ENTONCES T_2 es *pos_media*.
 R21: SI d es *objetivo* y l es *corrección* y $\dot{l}$ es *pos_media* ENTONCES T_2 es *neg_media*.
 R22: SI d es *objetivo* y l es *corrección* y $\dot{l}$ es *pos_pequeña* ENTONCES T_2 es *neg_media*.
 R23: SI d es *objetivo* y l es *corrección* y $\dot{l}$ es *cero* ENTONCES T_2 es *neg_media*.
 R24: SI d es *objetivo* y l es *corrección* y $\dot{l}$ es *neg_pequeña* ENTONCES T_2 es *cero*.
 R25: SI d es *objetivo* y l es *corrección* y $\dot{l}$ es *neg_media* ENTONCES T_2 es *pos_media*.

Puesto que al final del proceso de inferencia se ha de actuar tanto sobre el motor de la grúa como sobre el del carro, cada componente de la señal de control debe ser desborrosificada para adquirir un valor concreto. Aplicando el método de inferencia producto y el desborrosificador centro-promedio, o centroide, cada una de las señales de control viene determinada por:

$$T_j(d, l, \phi, \dot{d}, \dot{l}) = \frac{\sum_{i=1}^N w_i T_j^k}{\sum_{i=1}^N w_i} \quad (6.1)$$

donde w_i es el grado de cumplimiento de la regla i -ésima, N el número de reglas, $j = 1, 2$ el número de la componente de la señal de control y $k = 1..5$ indica el k -ésimo conjunto borroso en cada universo de discurso de cada T_j . Es decir, T_j^k es el par para el que se produce el valor máximo del conjunto borroso de salida en la regla correspondiente.

Simulación

Una vez diseñado el controlador borroso, su prueba y el análisis formal del comportamiento del sistema en lazo cerrado se puede realizar mediante simulación, para lo cual es necesario un modelo matemático de la planta.

El modelo matemático de una grúa porta contenedores está dado por [343]:

$$\begin{aligned} \ddot{d} &= u_1 - \delta_1 \phi u_2 + \delta_1 g \phi \\ \ddot{l} &= -\delta_2 \phi u_1 + u_2 \\ \ddot{\phi} &= -\frac{1}{l}(u_1 - \delta_1 \phi u_2 + (1 + \delta_1)g\phi + 2\dot{l}\dot{\phi}), \end{aligned} \quad (6.2)$$

donde las variables son las mostradas en la figura 6.2 y los parámetros δ_1 , δ_2 y las señales de control u_1 y u_2 están definidos como:

$$\begin{aligned} \delta_1 &= \frac{Mb_1^2}{J_1 + mb_1^2}, & \delta_2 &= \frac{Mb_2^2}{J_2 + Mb_2^2}, \\ u_1 &= \frac{b_1 T_1}{J_1 + mb_1^2}, & u_2 &= \frac{b_2(T_2 + Mb_2 g)}{J_2 + Mb_2^2}. \end{aligned} \quad (6.3)$$

Para un sistema práctico, los valores de los parámetros pueden estar dados por: $J_1 = 3,75 \text{ Kg m}^2$, $b_1 = 1,44 \times 10^{-2} \text{ m}$, $J_2 = 78,5 \text{ Kg m}^2$, y $b_2 = 1,22 \times 10^{-2} \text{ m}$.

Si se denomina $x_1 = d$, $x_2 = \dot{x}_1 = \dot{d}$, $x_3 = l$, $\dot{x}_3 = \dot{l} = x_4$, $x_5 = \phi$, $x_6 = \dot{x}_5 = \dot{\phi}$, el modelo (6.2) puede ser escrito como el modelo de estado (6.4).

$$\begin{aligned}
\dot{x}_1(t) &= x_2(t) \\
\dot{x}_2(t) &= u_1(t) - \delta_1 u_2(t) x_5(t) + \delta_1 g x_5(t) \\
\dot{x}_3(t) &= x_4(t) \\
\dot{x}_4(t) &= -\delta_2 u_1(t) x_5(t) + u_2(t) \\
\dot{x}_5(t) &= x_6(t) \\
\dot{x}_6(t) &= -\frac{1}{x_3(t)} (u_1(t) - \delta_1 u_2(t) x_5(t) + (1 + \delta_1) g x_5(t) + 2x_4(t) x_6(t))
\end{aligned} \tag{6.4}$$

El vector de estado tiene por componentes: x_1 , x_2 , x_3 , x_4 , x_5 y x_6 que representan, respectivamente, el desplazamiento horizontal, la velocidad de desplazamiento horizontal, el desplazamiento vertical, la velocidad de desplazamiento vertical, el ángulo de balanceo y su velocidad angular.

Equilibrios del sistema

Igualando a $\mathbf{0}$ la ecuación (6.4), se tiene que los estados de equilibrio del sistema son todos aquellos puntos de la forma $(d_{eq}, 0, l_{eq}, 0, 0, 0)$. Esto es, si el brazo de la grúa está a una distancia d_{eq} y el cable está enrollado a una longitud l_{eq} y no se aplica energía alguna ni hay perturbaciones externas (viento por ejemplo), el sistema permanecerá en esa posición indefinidamente.

La matriz jacobiana del sistema evaluada en los puntos de equilibrio es:

$$\mathbf{J}(d_{eq}, 0, l_{eq}, 0, 0, 0) = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \delta_1(g - u_2) & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\delta_2 u_1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{1}{l_{eq}^2} u_1 & 0 & -\frac{1}{l_{eq}} [-\delta_1 u_2 + (1 + \delta_1)g] & 0 \end{pmatrix} \tag{6.5}$$

donde, tal como muestra (6.3), se considera que $\mathbf{u}$ no depende de las coordenadas del vector de estado. Esta matriz tiene un autovalor cero doble, lo que da una idea de la complejidad del sistema.

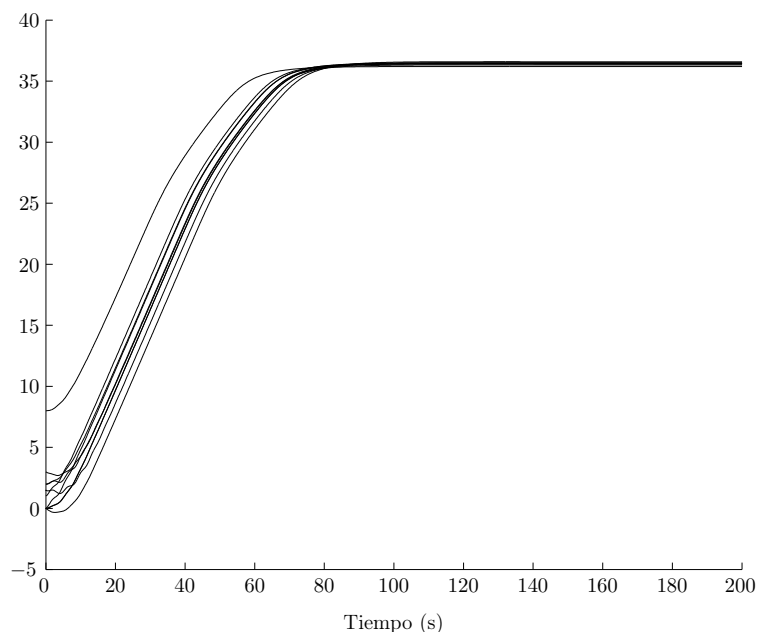


Figura 6.9: Evolución de la variable de estado $x_1(t)$.

Comportamiento del sistema en lazo cerrado

Introduciendo la expresión del controlador borroso diseñado (6.1) en el sistema (6.4), se obtiene la expresión del sistema en lazo cerrado. Esta expresión permite realizar diferentes simulaciones para evaluar el comportamiento del controlador.

En las figuras 6.9 a 6.14 se muestran los resultados de la simulación para las condiciones de funcionamiento siguientes: distancia a recorrer por el carro 37 m y longitud del cable hasta depositar el contenedor en el barco 30 m. Nótese que sin haber tenido en cuenta, por no ser algo trivial ni intuitiva para el experto, la influencia de $\dot{\phi}$ en la elaboración de las bases de reglas de T_1 y T_2 , los resultados de las simulaciones muestran un comportamiento satisfactorio del sistema de control desarrollado, el cual, a partir de diferentes condiciones iniciales, es capaz de realizar el trabajo del gruista en un tiempo razonable. Nótese cómo a partir de diferentes condiciones iniciales el contenedor se deposita en el barco en aproximadamente 2 minutos¹

Como se ha visto, a partir del conocimiento de la planta y de su manejo por un operador experto, se puede diseñar la base de reglas de un controlador borroso que gobierne dicha planta de forma adecuada, y sin necesidad de realizar un modelo de la misma. En este caso, un sistema no lineal muy complejo e inestable que requiere para su manejo óptimo de gruistas expertos, como es la grúa porta-contenedores, puede controlarse a partir de 5

¹Un paso siguiente no considerado en este trabajo sería tener en cuenta la influencia de perturbaciones externas, fundamentalmente el viento, lo cual obligaría a modificar las bases de reglas de T_1 y T_2 . Evidentemente, para este caso el modelo que se ha utilizado para realizar las simulaciones (6.2) ya no sería válido.

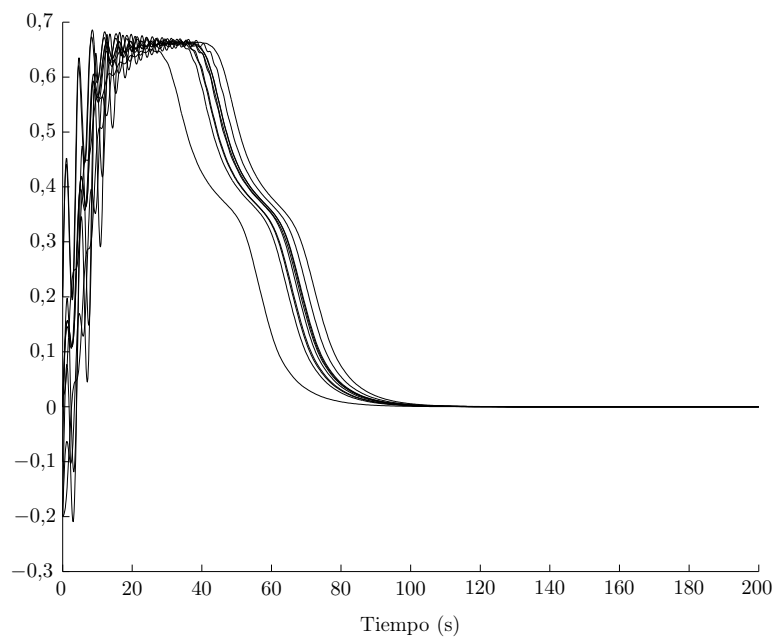


Figura 6.10: Evolución de la variable de estado $x_2(t)$.

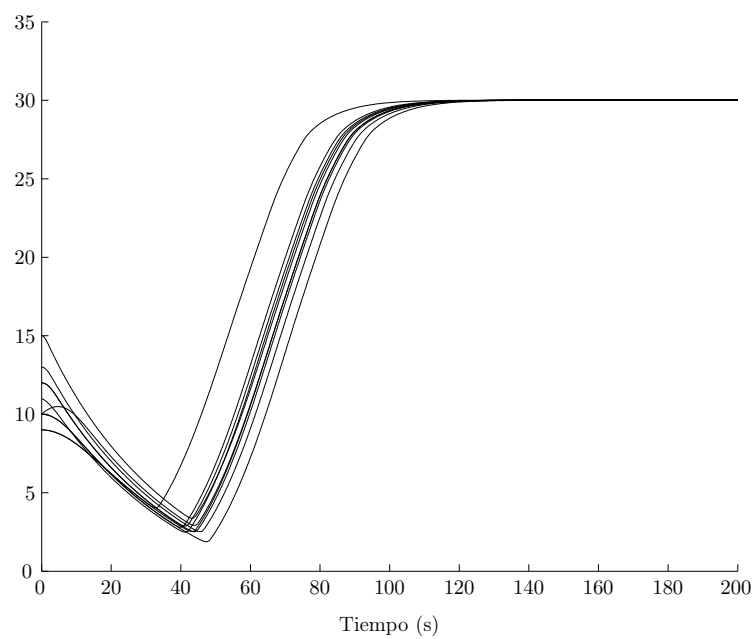


Figura 6.11: Evolución de la variable de estado $x_3(t)$.

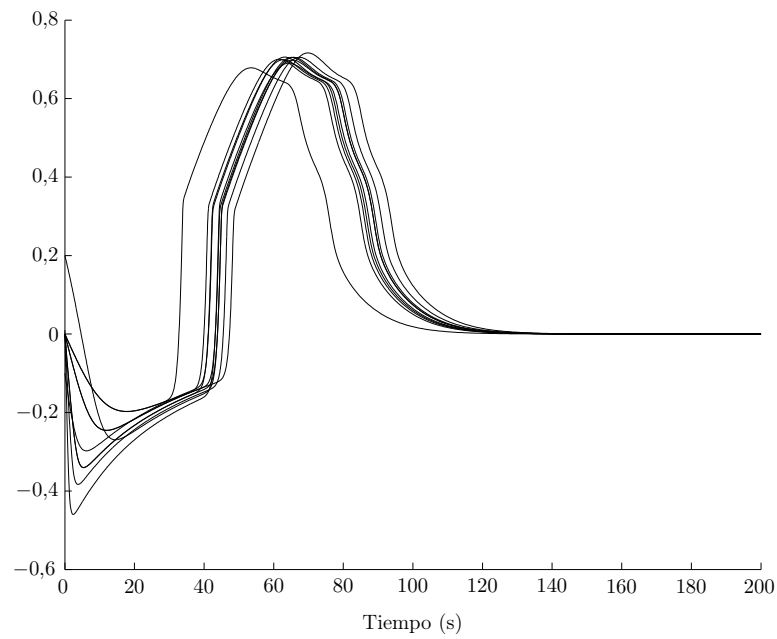


Figura 6.12: *Evolución de la variable de estado $x_4(t)$.*

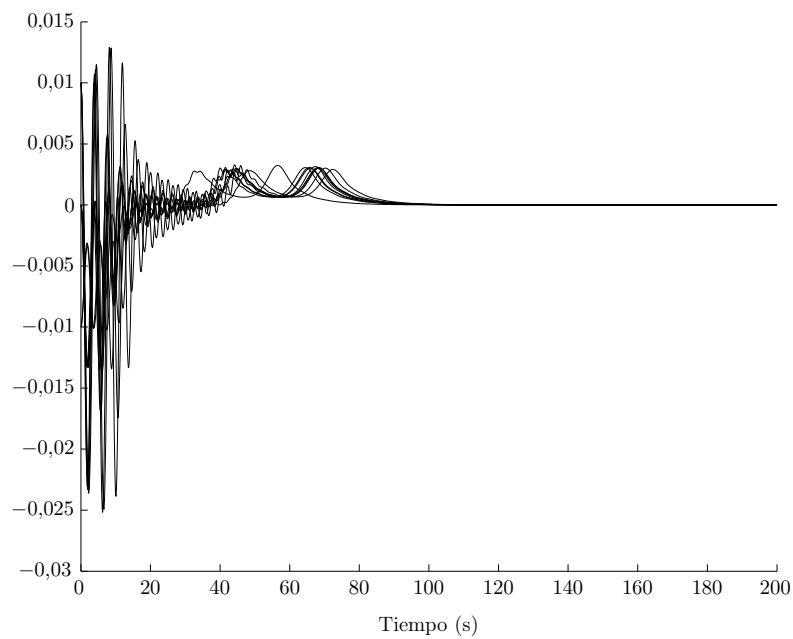


Figura 6.13: *Evolución de la variable de estado $x_5(t)$.*

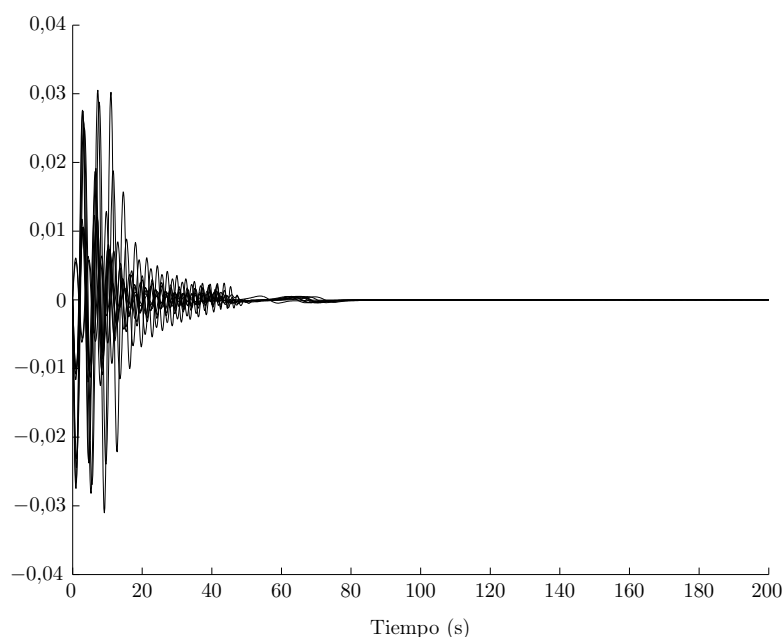


Figura 6.14: *Evolución de la variable de estado $x_6(t)$.*

variables medidas (distancia del carro horizontal, velocidad del carro, cantidad de cable colgante, velocidad de enrollado o de desenrollado del cable y ángulo del cable colgante sobre la horizontal), mediante dos señales de control dadas por la ecuación (6.1), que actúan sobre el motor de tracción del carro de la grúa y sobre el motor de enrollado/desenrollado del cable.

6.2. Síntesis de controladores borrosos estables por diseño

En esta sección se desarrollan varios ejemplos de diseño de controladores borrosos según la metodología presentada en el apartado 5.4. Para todos los diseños mostrados se ha mantenido la hipótesis de partida planteada en dicha sección, es decir, que la dinámica de la planta es completamente desconocida y el único conocimiento que se posee es un histórico de datos de entrada/salida.

En todos los casos planteados el modelo matemático (exacto) de la planta realmente sí es conocido, aunque tan sólo será utilizado para la obtención de una base de datos de entrada/salida. Esto es, en una planta real el proceso consistiría en aplicar señales a su entrada y medir los valores de las variables de interés como respuesta a estas entradas. Mediante estos datos se procede a la identificación de un modelo borroso a partir del cual se realiza todo el proceso de diseño.

Siguiendo con el esquema de este texto, en las siguientes aplicaciones se estudian paso a paso cada una de las etapas del proceso de diseño presentado en el apartado 5.4. Se comienza por el modelado de la planta a partir de datos de entrada/salida (capítulo 3), a continuación se estudian los estados de equilibrio de la planta a partir de su modelo

borroso (capítulo 4) y se diseña un controlador borroso que garantice la estabilidad del sistema de control en lazo cerrado (capítulo 5). Finalmente, se puede realizar un nuevo análisis de los estados de equilibrio del sistema, ahora sobre el sistema de control borroso en lazo cerrado (capítulo 4). De esta forma es posible comprobar la existencia de un único estado de equilibrio tras la aplicación del controlador borroso, así como su estabilidad.

Nota. En los siguientes ejercicios todas las simulaciones se han realizado realimentando la señal de control generada por el controlador borroso con la planta real, es decir, con el modelo matemático exacto de ésta. De esta forma se garantiza el correcto funcionamiento del controlador borroso con el sistema real, en vez de con el modelo borroso que se ha utilizado para su diseño.

6.2.1. Ejemplo ilustrativo

Sea el siguiente sistema no lineal representado mediante su modelo de estado:

$$\begin{aligned}\dot{x}_1(t) &= x_1(t) + 2x_2(t) + u_1(t) \\ \dot{x}_2(t) &= x_1(t) - 2x_2^3(t) + u_2(t),\end{aligned}\tag{6.6}$$

donde se define el vector de estado como:

$$\mathbf{x}(t) = \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix},\tag{6.7}$$

y el vector de control como:

$$\mathbf{u}(t) = \begin{pmatrix} u_1(t) \\ u_2(t) \end{pmatrix};\tag{6.8}$$

y sea el universo de discurso de las variables de estado y control:

$$X \subseteq \{\mathbf{x} \mid \mathbf{x} \in [-10, 10]\}\tag{6.9}$$

y

$$U \subseteq \{\mathbf{u} \mid \mathbf{u} \in [-50, 50]\},\tag{6.10}$$

respectivamente.

El sistema (6.6) es un sistema MIMO (*Multiple-Input-Multiple-Output*) no lineal relativamente sencillo, cuyo objetivo es mostrar la metodología presentada en la sección 5.4. La sencillez del sistema permite que sea modelado mediante pocas reglas, y de esta forma se facilita la comprensión de cada uno de los pasos propuestos en la metodología de diseño desarrollada.

Puesto que se supone que la dinámica de la planta es desconocida, el primer paso es obtener un modelo borroso del sistema a partir de datos de entrada/salida del mismo.

Antes de iniciar la fase de identificación se han obtenido 1250 pares de datos de entrada/salida para cada una de las salidas del sistema (6.6), de forma que 1000 datos (el

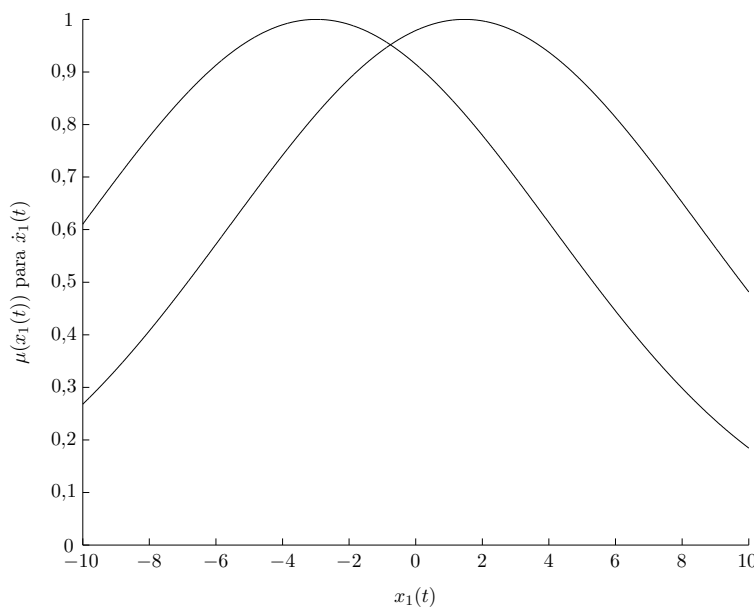


Figura 6.15: Funciones de pertenencia para la entrada $x_1(t)$ en la inferencia de $\hat{x}_1(t)$.

80 %) se emplean en el proceso de modelado y 250 (el 20 % restante) en el de validación. Nótese que el modelo (6.6) se ha utilizado sólo y exclusivamente para generar el conjunto de datos de entrada/salida a partir del cual se inicia la metodología de diseño, de modo que a todos los efectos, el modelo de la planta se considera completamente desconocido.

Identificación de la planta

Como se vio en la sección 3.3, el proceso de modelado borroso se desarrolla en tres fases: identificación de la estructura, identificación de los parámetros del modelo borroso y validación del modelo obtenido.

Utilizando el algoritmo de *clustering* propuesto por S. Chiu [246], se han obtenido 2 reglas para la inferencia de $\hat{x}_1(t)$, donde las funciones de pertenencia de los antecedentes se muestran en la figuras 6.15 a 6.18. Empleando el mismo algoritmo de *clustering* para $\hat{x}_2(t)$ se han obtenido otras 2 reglas, donde las funciones de pertenencia de los antecedentes se muestran en la figuras 6.19 a 6.22.

Al ajustar los parámetros del modelo borroso mediante el algoritmo neuroborroso propuesto en [271], se obtiene el siguiente conjunto de reglas:

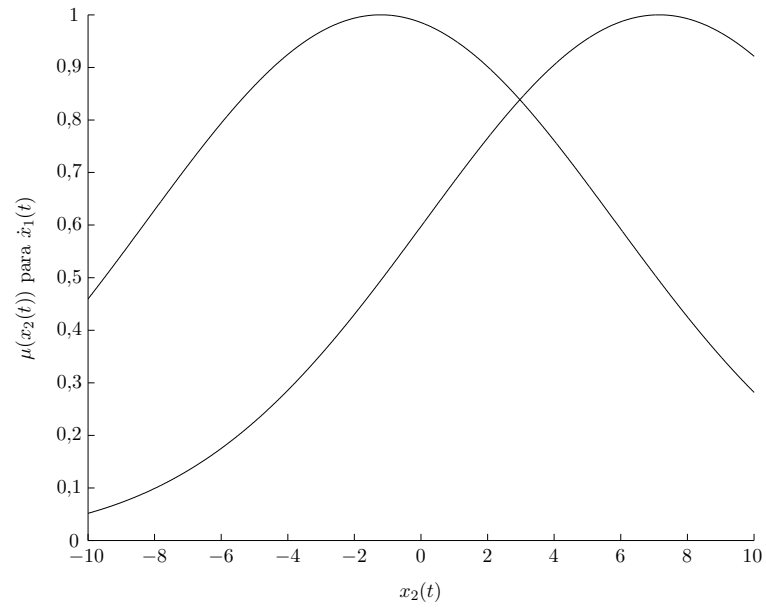


Figura 6.16: Funciones de pertenencia para la entrada $x_2(t)$ en la inferencia de $\dot{x}_1(t)$.

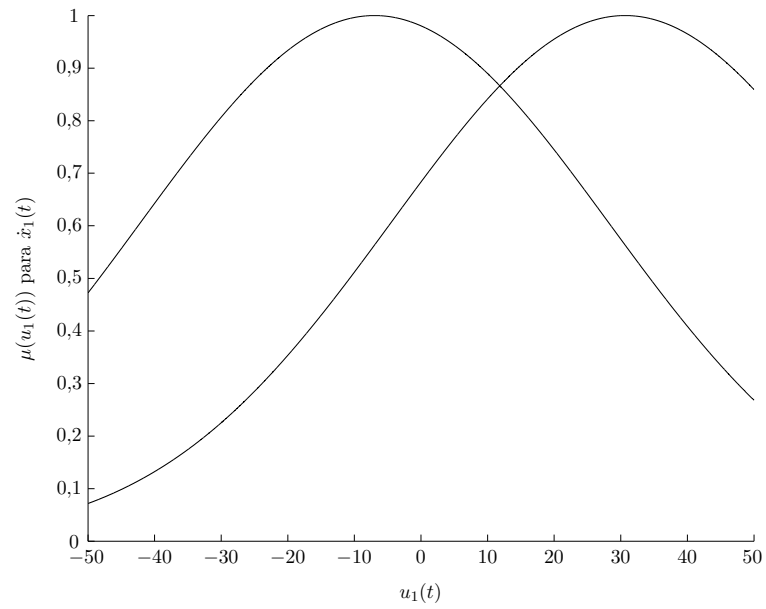


Figura 6.17: Funciones de pertenencia para la entrada $u_1(t)$ en la inferencia de $\dot{x}_1(t)$.

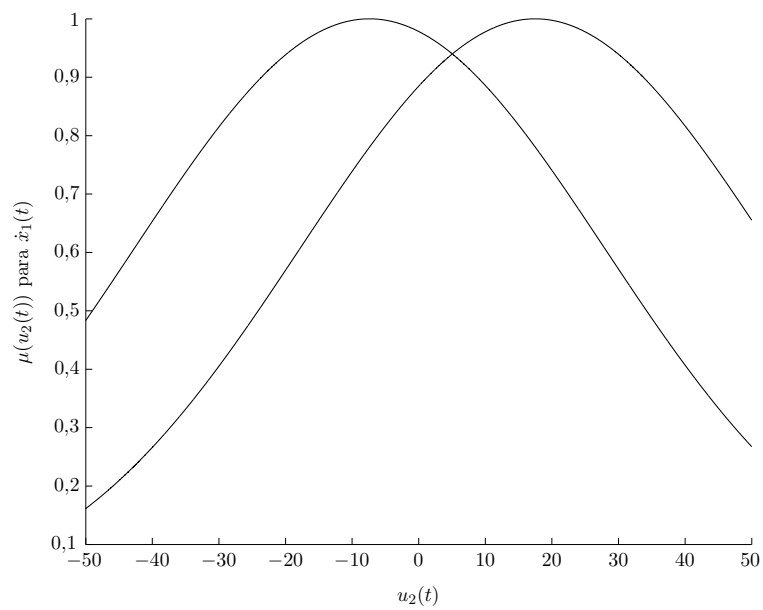


Figura 6.18: Funciones de pertenencia para la entrada $u_2(t)$ en la inferencia de $\dot{x}_1(t)$.

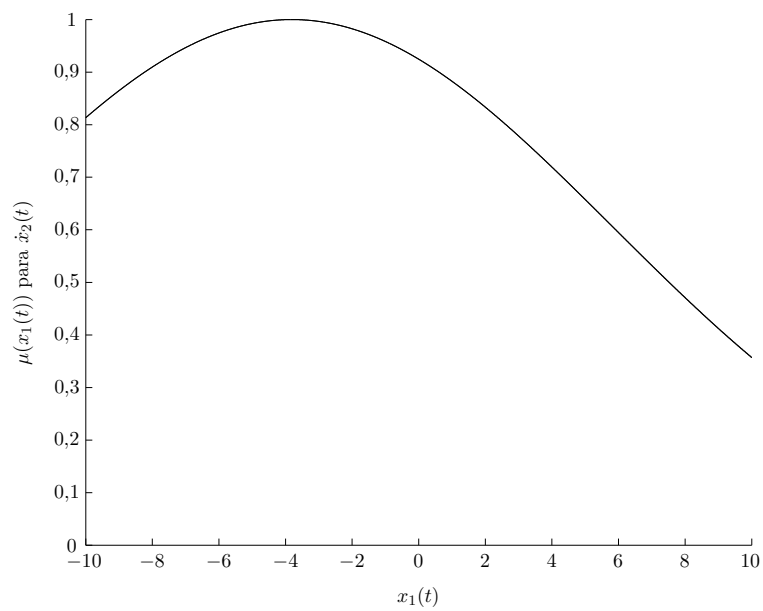


Figura 6.19: Funciones de pertenencia para la entrada $x_1(t)$ en la inferencia de $\dot{x}_2(t)$.

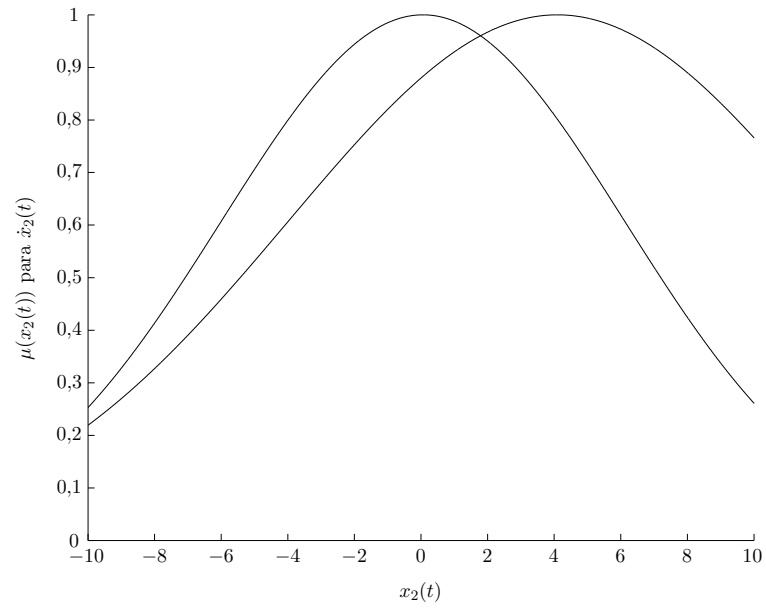


Figura 6.20: Funciones de pertenencia para la entrada $x_2(t)$ en la inferencia de $\dot{x}_2(t)$.

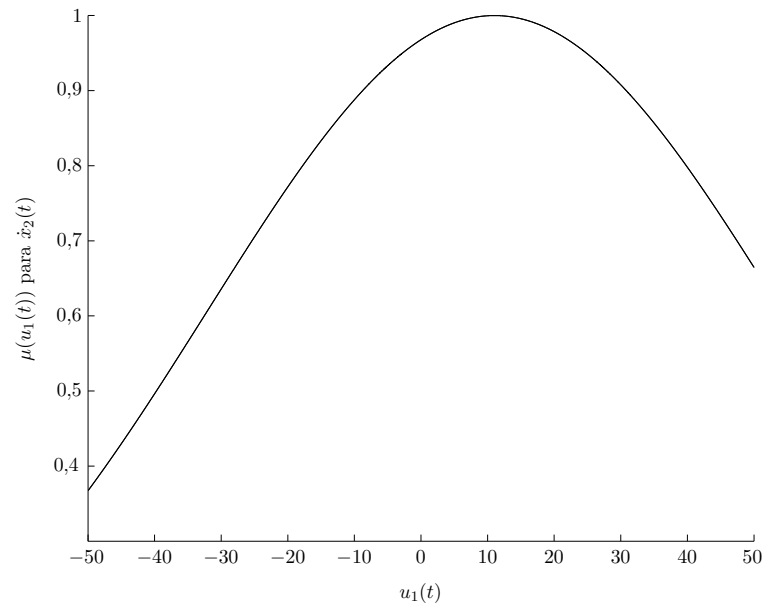


Figura 6.21: Funciones de pertenencia para la entrada $u_1(t)$ en la inferencia de $\dot{x}_2(t)$.

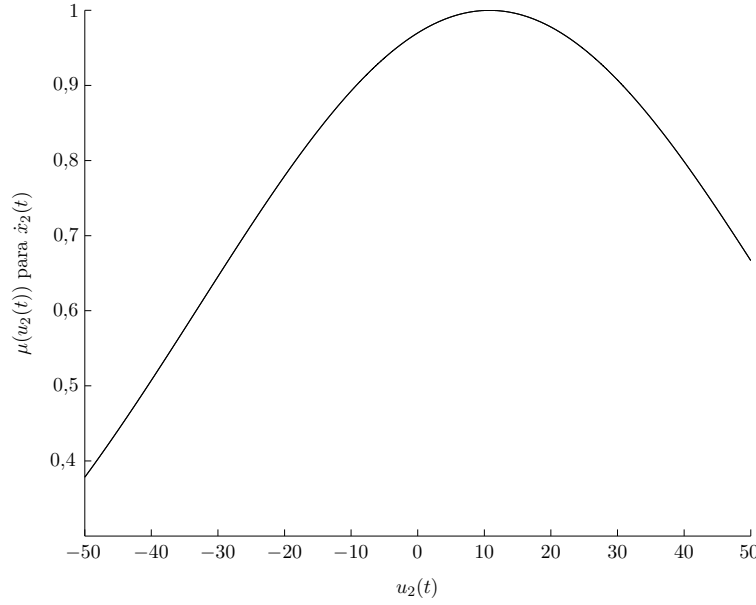


Figura 6.22: Funciones de pertenencia para la entrada $u_2(t)$ en la inferencia de $\dot{x}_2(t)$.

SI x_1 es Gaussmf(1,464;9,987) y x_2 es Gaussmf(-1,211;9,964) y u_1 es Gaussmf(-6,976;49,67) y u_2 es Gaussmf(-7,39;49,98)

ENTONCES $\dot{x}_1 = -1,551 \times 10^{-8} + 1x_1 + 2x_2 + 1u_1 + 2,196 \times 10^{-11}u_2$

SI x_1 es Gaussmf(-2,982;9,986) y x_2 es Gaussmf(7,149;9,96) y u_1 es Gaussmf(30,65;49,67) y u_2 es Gaussmf(17,5;49,98)

ENTONCES $\dot{x}_1 = 3,164 \times 10^{-8} + 1x_1 + 2x_2 + 1u_1 - 2,148 \times 10^{-10}u_2$

SI x_1 es Gaussmf(-3,811;13,62) y x_2 es Gaussmf(0,06026;8,578) y u_1 es Gaussmf(11,03;60,97) y u_2 es Gaussmf(10,77;61,63)

ENTONCES $\dot{x}_2 = -6498 + 1,993x_1 + 419,8x_2 - 0,1538u_1 + 0,9489u_2$

SI x_1 es Gaussmf(-3,812;13,61) y x_2 es Gaussmf(4,095;11,44) y u_1 es Gaussmf(11,04;60,97) y u_2 es Gaussmf(10,78;61,63)

ENTONCES $\dot{x}_2 = 7385 + 0,549x_1 - 926x_2 + 0,1001u_1 + 1,027u_2$

Donde Gaussmf representa una función de pertenencia gaussiana, ver ecuación (2.12), cuyos parámetros son el centro y el ancho, respectivamente.

El error cuadrático medio obtenido al finalizar el proceso de modelado ha sido de $3,211 \times 10^{-17}$ para $\dot{x}_1$, y de 8,624 para $\dot{x}_2$.

Antes de dar por bueno el modelo borroso obtenido, éste debe ser validado con los datos no utilizados durante el ajuste de sus parámetros. Empleando pues los 250 datos reservados para la validación se ha obtenido un error cuadrático medio de $7,319 \times 10^{-17}$ para $\dot{x}_1(t)$, y de 9,909 para $\dot{x}_2(t)$. Estos resultados, como puede comprobarse gráficamente en las figuras 6.23 a 6.26, permiten comprobar el buen comportamiento del modelo borroso obtenido y pasar a la siguiente fase de diseño.

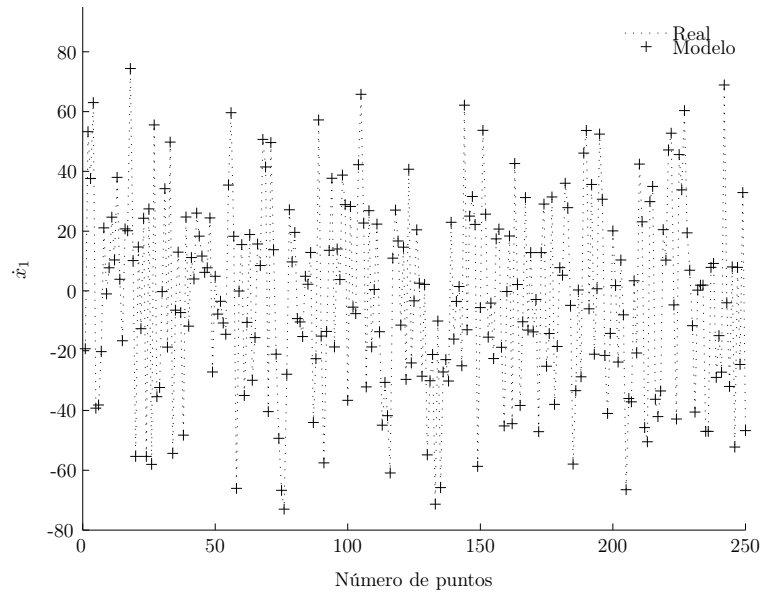


Figura 6.23: Validación del modelo borroso para x_1 .

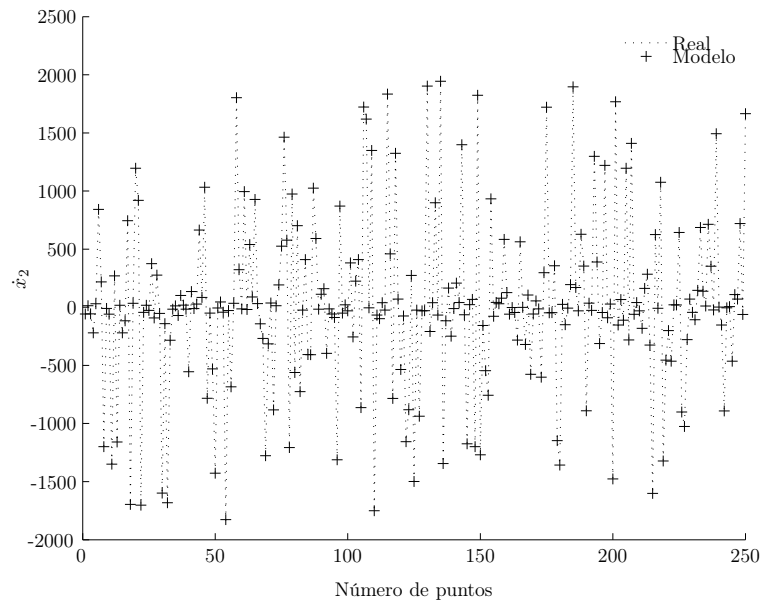


Figura 6.24: Validación del modelo borroso para x_2 .

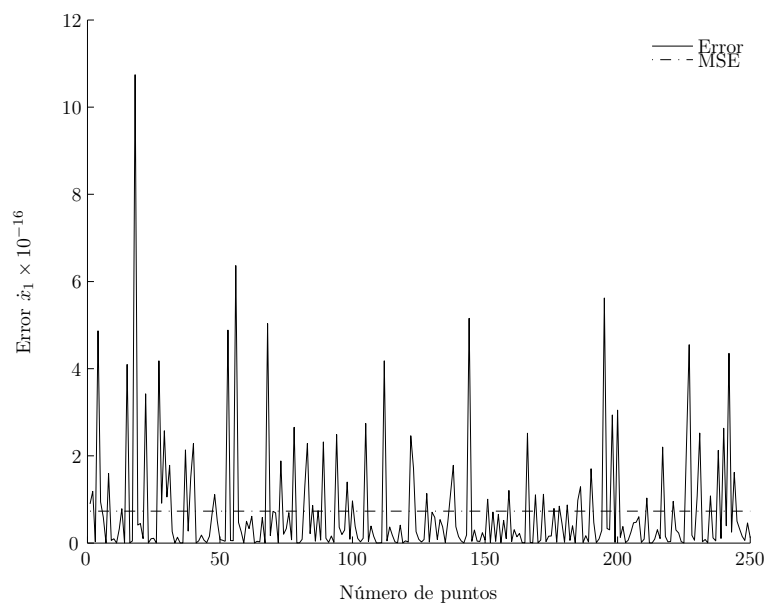


Figura 6.25: Error de modelado durante la validación de $\dot{x}_1$.

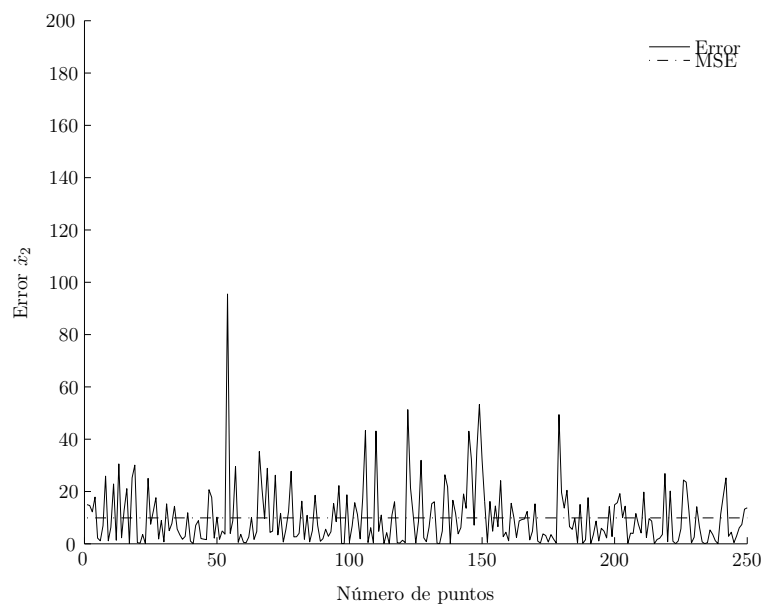


Figura 6.26: Error de modelado durante la validación de $\dot{x}_2$.

Estudio de la planta a partir de su modelo borroso

Una vez se ha obtenido un modelo borroso de la planta, éste puede ser utilizado para estudiar sus propiedades y comportamiento dinámico. Como se vio en la sección 4.4, el modelo borroso de un sistema puede ser empleado para obtener sus estados de equilibrio. Evidentemente, la precisión del modelo borroso afectará de forma determinante a la precisión de los estados de equilibrio obtenidos, pero más allá de los datos cuantitativos que se verán afectados por errores inherentes al proceso de modelado, cualitativamente se puede descubrir de forma aproximada la existencia de estados de equilibrio en un sistema inicialmente desconocido, y posteriormente estudiar su estabilidad local, ver la sección 4.2.

Se ha aplicado cinco veces el algoritmo de la figura 4.1 al modelo borroso de la planta, partiendo en cada una de las ejecuciones de un punto de inicio $\mathbf{x}_0$ aleatorio perteneciente al universo de discurso de $\mathbf{x}$. Los resultados obtenidos en cada una de esas ejecuciones, tanto para el método de Gauss–Newton como para el método de Levenberg–Marquardt, aplicando el cálculo explícito de la matriz jacobiana y sin aplicarlo, se muestran en las tablas 6.3 a 6.7. En todos los casos se ha supuesto $\mathbf{u}(t) = \mathbf{0}$.

	Gauss–Newton	G–N con jacobiana	Levenberg–Marquardt	L–M con jacobiana
Estado de equilibrio	$\begin{pmatrix} 1,26062 \\ -0,63031 \end{pmatrix}$	$\begin{pmatrix} 1,26062 \\ -0,63031 \end{pmatrix}$	$\begin{pmatrix} 1,26062 \\ -0,63031 \end{pmatrix}$	$\begin{pmatrix} 1,26062 \\ -0,63031 \end{pmatrix}$
Nº Iteraciones	22	22	22	22
Evaluaciones de la función objetivo	69	46	69	46
Distancia del punto obtenido al $\mathbf{0}$	4,42124e-16	1,8866e-16	4,42124e-16	1,8866e-16
Tiempo empleado (s)	0,665378	0,143811	0,086432	0,068031

Tabla 6.3: Resultados de la búsqueda de estados de equilibrio. Ejecución 1.

	Gauss–Newton	G–N con jacobiana	Levenberg–Marquardt	L–M con jacobiana
Estado de equilibrio	$\begin{pmatrix} 1,26062 \\ -0,63031 \end{pmatrix}$	$\begin{pmatrix} 1,26062 \\ -0,63031 \end{pmatrix}$	$\begin{pmatrix} 1,26062 \\ -0,63031 \end{pmatrix}$	$\begin{pmatrix} 1,26062 \\ -0,63031 \end{pmatrix}$
Nº Iteraciones	7	8	7	8
Evaluaciones de la función objetivo	24	18	24	18
Distancia del punto obtenido al $\mathbf{0}$	1,42917e-13	2,65363e-24	1,42917e-13	2,65363e-24
Tiempo empleado (s)	0,617109	0,121069	0,039524	0,045003

Tabla 6.4: Resultados de la búsqueda de estados de equilibrio. Ejecución 2.

	Gauss-Newton	G-N con jacobiana	Levenberg-Marquardt	L-M con jacobiana
Estado de equilibrio	$\begin{pmatrix} 1,26062 \\ -0,63031 \end{pmatrix}$	$\begin{pmatrix} 1,26062 \\ -0,63031 \end{pmatrix}$	$\begin{pmatrix} 1,26062 \\ -0,63031 \end{pmatrix}$	$\begin{pmatrix} 1,26062 \\ -0,63031 \end{pmatrix}$
Nº Iteraciones	10	10	10	10
Evaluaciones de la función objetivo	33	22	33	22
Distancia del punto obtenido al 0	9,12017e-15	5,37422e-15	9,12017e-15	5,37422e-15
Tiempo empleado (s)	0,619826	0,122609	0,053173	0,050231

Tabla 6.5: Resultados de la búsqueda de estados de equilibrio. Ejecución 3.

	Gauss-Newton	G-N con jacobiana	Levenberg-Marquardt	L-M con jacobiana
Estado de equilibrio	$\begin{pmatrix} 1,26062 \\ -0,63031 \end{pmatrix}$	$\begin{pmatrix} 1,26062 \\ -0,63031 \end{pmatrix}$	$\begin{pmatrix} 1,26062 \\ -0,63031 \end{pmatrix}$	$\begin{pmatrix} 1,26062 \\ -0,63031 \end{pmatrix}$
Nº Iteraciones	6	6	6	6
Evaluaciones de la función objetivo	21	14	21	14
Distancia del punto obtenido al 0	7,31016e-15	1,33097e-14	7,31016e-15	1,33097e-14
Tiempo empleado (s)	0,635315	0,114088	0,03717	0,041436

Tabla 6.6: Resultados de la búsqueda de estados de equilibrio. Ejecución 4.

	Gauss-Newton	G-N con jacobiana	Levenberg-Marquardt	L-M con jacobiana
Estado de equilibrio	$\begin{pmatrix} 1,26062 \\ -0,63031 \end{pmatrix}$	$\begin{pmatrix} 1,26062 \\ -0,63031 \end{pmatrix}$	$\begin{pmatrix} 1,26062 \\ -0,63031 \end{pmatrix}$	$\begin{pmatrix} 1,26062 \\ -0,63031 \end{pmatrix}$
Nº Iteraciones	12	12	12	12
Evaluaciones de la función objetivo	39	26	39	26
Distancia del punto obtenido al 0	1,39385e-21	7,65821e-23	1,39385e-21	7,65821e-23
Tiempo empleado (s)	0,639078	0,130633	0,060114	0,055742

Tabla 6.7: Resultados de la búsqueda de estados de equilibrio. Ejecución 5.

Como se puede comprobar, en todas las ejecuciones se ha obtenido prácticamente el mismo estado de equilibrio situado en el punto $(1,26062, -0,63031)^T$ (las diferencias entre el resultado de las distintas ejecuciones se sitúan a partir del noveno decimal). Como es lógico, este punto no se corresponde exactamente con el equilibrio real que presenta el sistema al considerar $\mathbf{u} = \mathbf{0}$, el cual está situado en $\mathbf{x} = (0, 0)^T$; ya que los errores propios del modelo se reflejan en cualquier cálculo que se realice sobre éste. Sin embargo, se ha obtenido un único estado de equilibrio como en el sistema original, y, teniendo en cuenta que $\dot{x}_1$ puede sobrepasar un valor de 60 y $\dot{x}_2$ los 2000, y que el error de validación de $\dot{x}_2$ se sitúa en torno a un valor de 9, éste puede considerarse bastante cercano al estado de equilibrio real.

Observando las tablas 6.3 a 6.7 puede comprobarse que el algoritmo de Levenberg–Marquardt es más rápido que el de Gauss–Newton, a pesar de requerir el mismo número de iteraciones para converger. Un resultado más interesante se observa al comparar los datos referentes a la aplicación del cálculo explícito de la matriz jacobiana a cada uno de los algoritmos. Al aplicarlo al método de Gauss–Newton, éste presenta mayor rapidez en la convergencia, obteniendo también, excepto en un caso, una mayor precisión del resultado. Con respecto al algoritmo de Levenberg–Marquardt, la aplicación del cálculo explícito de la matriz jacobiana implica normalmente una mejora de la precisión, y, en bastantes casos, una mejora de la velocidad de convergencia.

Al linealizar el modelo borroso mediante la aplicación del algoritmo 4.2 en torno al estado de equilibrio obtenido previamente, $\mathbf{x}^T = (1,26062, -0,63031)^T$, con $\mathbf{u} = (0, 0)^T$, se obtiene el sistema lineal:

$$\dot{\mathbf{x}}(t) \simeq \begin{pmatrix} 1 & 2 \\ 0,98464 & -2,41350 \end{pmatrix} \mathbf{x}(t) + \begin{pmatrix} 1 & 0 \\ -0,01313 & 0,99485 \end{pmatrix} \mathbf{u}(t), \quad (6.11)$$

cuyos autovalores son $\lambda_1 = -2,91$ y $\lambda_2 = 1,5$. Por lo tanto, se puede afirmar que el sistema es inestable en lazo abierto, y su estado de equilibrio se comporta como un punto de silla.

La linealización del sistema real (6.6) en torno su estado de equilibrio, $\mathbf{x}^T = (0, 0)^T$, con $\mathbf{u} = (0, 0)^T$, viene determinada por (6.12), siendo sus autovalores $\lambda_1 = -1$ y $\lambda_2 = 2$. Comparando² estos resultados con los obtenidos mediante el modelo borroso, se puede comprobar que el análisis previo realizado únicamente mediante la utilización del modelo borroso era correcto, ya que efectivamente el sistema es inestable en lazo abierto y su estado de equilibrio se comporta como un punto de silla.

$$\dot{\mathbf{x}}(t) \simeq \begin{pmatrix} 1 & 2 \\ 1 & 0 \end{pmatrix} \mathbf{x}(t) + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \mathbf{u}(t) \quad (6.12)$$

²Nótese que (6.11) es consecuencia de un modelo borroso de estado obtenido en base a datos de entrada/salida, de la obtención de los estados de equilibrio del mismo y, por último, de la linealización del modelo borroso de estado en torno al único estado de equilibrio obtenido.

Diseño de un controlador borroso estable

Como se vio en la sección 5.2.2, el teorema de Krasovskii, tanto en su versión simple como generalizada, impone unas condiciones suficientes de estabilidad asintótica que pueden ser empleadas como herramienta de síntesis de controladores borrosos; además, proporciona una función de Lyapunov para el sistema diseñado (ver ecuaciones (5.13) y (5.15)).

Mediante la utilización del algoritmo presentado en la sección 5.4.4 puede realizarse el diseño de un controlador borroso en base al teorema de Krasovskii. Para que las expresiones (5.17) y (5.12) sean completamente equivalentes, basta suponer una única región Θ_q igual al espacio de estado completo, es decir, $s = 1$ y $q = 1$, donde la función $\varphi_q(\mathbf{x}) = 1 \forall \mathbf{x} \in X$ y la matriz $\mathbf{P}_1$ es igual a la matriz identidad. Aplicando dicho algoritmo sobre el modelo borroso de la planta en cuestión, se obtiene el siguiente controlador borroso:

SI x_1 es Gaussmf(1,464;9,987) y x_2 es Gaussmf(-1,211;9,964)
 Entonces $u_1 = 0,1053-3,045x_1-2,627x_2$
 SI x_1 es Gaussmf(-2,982;9,986) y x_2 es Gaussmf(7,149;9,96)
 Entonces $u_1 = 0,06726-0,9329x_1-1,521x_2$
 SI x_1 es Gaussmf(-3,811;13,62) y x_2 es Gaussmf(0,06026;8,578)
 Entonces $u_1 = -1,941-1,462x_1-1,693x_2$
 SI x_1 es Gaussmf(-3,812;13,61) y x_2 es Gaussmf(4,095;11,44)
 Entonces $u_1 = 2,037-2,479x_1-2,575x_2$
 SI x_1 es Gaussmf(1,464;9,987) y x_2 es Gaussmf(-1,211;9,964)
 Entonces $u_2 = 2,595-0,7221x_1-0,6478x_2$
 SI x_1 es Gaussmf(-2,982;9,986) y x_2 es Gaussmf(7,149;9,96)
 Entonces $u_2 = -2,121-1,189x_1-0,5057x_2$
 SI x_1 es Gaussmf(-3,811;13,62) y x_2 es Gaussmf(0,06026;8,578)
 Entonces $u_2 = 3,31-0,3085x_1-1,957x_2$
 SI x_1 es Gaussmf(-3,812;13,61) y x_2 es Gaussmf(4,095;11,44)
 Entonces $u_2 = 0,9739-2,09x_1-2,571x_2$

Analizando el controlador borroso generado se puede comprobar el funcionamiento del algoritmo de inicialización de la figura 5.6. Como puede observarse en la página 174, el modelo borroso de la planta está formado por 4 reglas, de forma que la fase de inicialización crea 4 reglas para cada una de las señales de control; es decir, 4 reglas para u_1 y otras 4 para u_2 . Como se puede observar, los antecedentes de las reglas del controlador para cada una de sus salidas son idénticos entre sí, y a su vez idénticos a los antecedentes del modelo borroso de la planta a controlar. Los parámetros adaptables de los consecuentes se inician todos a cero.

Una vez creado el controlador inicial, el algoritmo de ajuste (figura 5.8) modifica los parámetros adaptables de los consecuentes del controlador borroso para que quede satisfecha la condición de estabilidad; es decir, que la matriz $\mathbf{F}(\mathbf{x}) = \mathbf{J}(\mathbf{x})^T + \mathbf{J}(\mathbf{x})$ sea definida negativa en un entorno del origen. El resultado pasa a la fase final del algoritmo de diseño para ajustar el estado de equilibrio, y así se obtiene el controlador borroso de arriba.

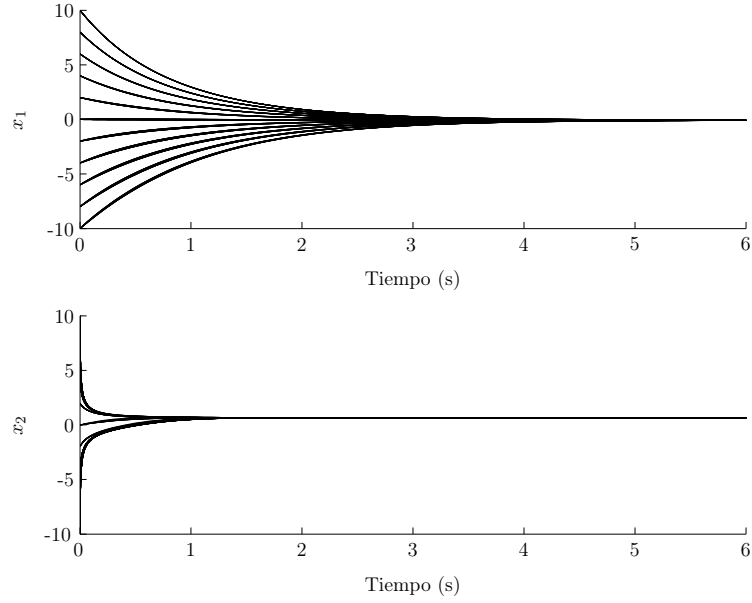


Figura 6.27: *Simulación para distintas condiciones iniciales.*

En las figuras 6.27 y 6.28 se puede comprobar la evolución del sistema de control diseñado partiendo de distintas condiciones iniciales. En dichas figuras efectivamente puede observarse la estabilidad del sistema.

Un vez diseñado el controlador borroso es posible estudiar los estados de equilibrio resultantes de la realimentación de la planta con éste. Para ello se ha aplicado cinco veces el algoritmo numérico de minimización de la figura 4.1, partiendo en cada una de las ejecuciones de un punto de inicio $\mathbf{x}_0$ aleatorio perteneciente al universo de discurso de las variables de estado. Como puede observarse las tablas 6.8 a 6.12 que recogen los resultados obtenidos, prácticamente sólo existen un único estado de equilibrio que se corresponde con el origen de coordenadas. Nótese cómo la inclusión del cálculo explícito de la matriz jacobiana mejora en muchos casos tanto la velocidad de convergencia como la precisión del resultado.

Calculando la matriz jacobiana del sistema de control borroso en lazo cerrado en el origen, se obtiene:

$$J(\mathbf{0}) = \begin{pmatrix} -1,09579 & -0,13038 \\ 0,10028 & -0,87830 \end{pmatrix}, \quad (6.13)$$

cuyos autovalores son $\lambda_1 = -0,98704 + 0,03535i$ y $\lambda_2 = -0,98704 - 0,03535i$. Por lo tanto, se demuestra que el estado de equilibrio es asintóticamente estable.

En este ejemplo se ha mostrado cómo funciona el algoritmo de diseño presentado en el capítulo 5, y cómo puede emplearse junto con el teorema de Krasovskii para la síntesis de un controlador borroso que asegure la estabilidad del sistema de control en lazo cerrado. Sin embargo, no siempre es posible utilizar el teorema de Krasovskii para diseñar un con-

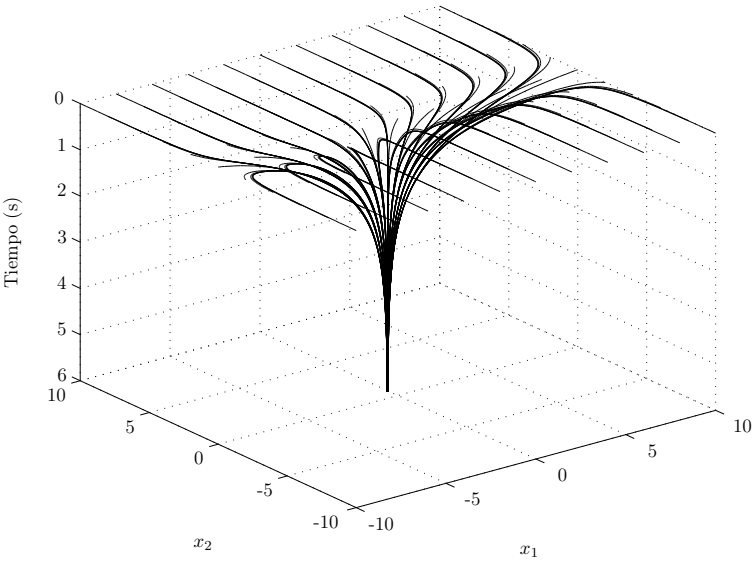


Figura 6.28: Retrato de fase temporal.

	Gauss-Newton	G-N con jacobiana	Levenberg-Marquardt	L-M con jacobiana
Estado de equilibrio	$\begin{pmatrix} 9,15647e-11 \\ -7,83691e-10 \end{pmatrix}$	$\begin{pmatrix} 7,70351e-12 \\ -7,19906e-11 \end{pmatrix}$	$\begin{pmatrix} 9,15647e-11 \\ -7,83691e-10 \end{pmatrix}$	$\begin{pmatrix} 7,70351e-12 \\ -7,19906e-11 \end{pmatrix}$
Nº Iteraciones	5	5	5	5
Evaluaciones de la función objetivo	18	12	18	12
Distancia del punto obtenido al 0	4,85501e-19	3,97491e-21	4,85501e-19	3,97491e-21
Tiempo empleado (s)	0,624002	0,103941	0,032167	0,037212

Tabla 6.8: Resultados de la búsqueda de estados de equilibrio para el sistema en lazo cerrado. Ejecución 1.

	Gauss-Newton	G-N con jacobiana	Levenberg-Marquardt	L-M con jacobiana
Estado de equilibrio	$\begin{pmatrix} -9,84543e-12 \\ 8,23248e-11 \end{pmatrix}$	$\begin{pmatrix} 2,7535e-13 \\ -2,81427e-12 \end{pmatrix}$	$\begin{pmatrix} -9,84543e-12 \\ 8,23248e-11 \end{pmatrix}$	$\begin{pmatrix} 2,7535e-13 \\ -2,81427e-12 \end{pmatrix}$
Nº Iteraciones	12	12	12	12
Evaluaciones de la función objetivo	39	26	39	26
Distancia del punto obtenido al 0	5,33778e-21	6,57783e-24	5,33778e-21	6,57783e-24
Tiempo empleado (s)	0,685474	0,121571	0,066475	0,067101

Tabla 6.9: Resultados de la búsqueda de estados de equilibrio para el sistema en lazo cerrado. Ejecución 2.

	Gauss-Newton	G-N con jacobiana	Levenberg-Marquardt	L-M con jacobiana
Estado de equilibrio	$\begin{pmatrix} -8,81312e-12 \\ 7,36569e-11 \end{pmatrix}$	$\begin{pmatrix} 1,64023e-13 \\ -1,79828e-12 \end{pmatrix}$	$\begin{pmatrix} -8,81312e-12 \\ 7,36569e-11 \end{pmatrix}$	$\begin{pmatrix} 1,64023e-13 \\ -1,79828e-12 \end{pmatrix}$
Nº Iteraciones	12	12	12	12
Evaluaciones de la función objetivo	39	26	39	26
Distancia del punto obtenido al 0	4,44495e-21	1,24341e-24	4,44495e-21	1,24341e-24
Tiempo empleado (s)	0,704302	0,124267	0,061997	0,068536

Tabla 6.10: Resultados de la búsqueda de estados de equilibrio para el sistema en lazo cerrado. Ejecución 3.

	Gauss-Newton	G-N con jacobiana	Levenberg-Marquardt	L-M con jacobiana
Estado de equilibrio	$\begin{pmatrix} 3,7486e-11 \\ -3,33726e-10 \end{pmatrix}$	$\begin{pmatrix} 1,12331e-11 \\ -1,14324e-10 \end{pmatrix}$	$\begin{pmatrix} 3,7486e-11 \\ -3,33726e-10 \end{pmatrix}$	$\begin{pmatrix} 1,12331e-11 \\ -1,14324e-10 \end{pmatrix}$
Nº Iteraciones	6	6	6	6
Evaluaciones de la función objetivo	21	14	21	14
Distancia del punto obtenido al 0	8,75808e-20	1,01724e-20	8,75808e-20	1,01724e-20
Tiempo empleado (s)	0,646222	0,102585	0,040036	0,044547

Tabla 6.11: Resultados de la búsqueda de estados de equilibrio para el sistema en lazo cerrado. Ejecución 4.

	Gauss-Newton	G-N con jacobiana	Levenberg-Marquardt	L-M con jacobiana
Estado de equilibrio	$\begin{pmatrix} 8,08046e-09 \\ -7,87225e-08 \end{pmatrix}$	$\begin{pmatrix} 6,16959e-09 \\ -6,26487e-08 \end{pmatrix}$	$\begin{pmatrix} 8,08046e-09 \\ -7,87225e-08 \end{pmatrix}$	$\begin{pmatrix} 6,16959e-09 \\ -6,26487e-08 \end{pmatrix}$
Nº Iteraciones	12	12	12	12
Evaluaciones de la función objetivo	39	26	39	26
Distancia del punto obtenido al 0	4,89517e-15	3,0981e-15	4,89517e-15	3,0981e-15
Tiempo empleado (s)	0,704675	0,119984	0,061284	0,061249

Tabla 6.12: Resultados de la búsqueda de estados de equilibrio para el sistema en lazo cerrado. Ejecución 5.

trolador borroso adecuado. En sistemas complejos el teorema de Krasovskii puede resultar demasiado restrictivo como para asegurar su cumplimiento en una región suficientemente amplia en torno al origen [170]. Por ello, en este tipo de sistemas resulta más adecuada la utilización del teorema 5.7, cuya condición es más sencilla de ajustar debido a la división del espacio de estado en regiones.

A continuación se abordará el diseño de un controlador borroso para la misma planta anterior, pero aplicando el algoritmo del apartado 5.4.4 en base al teorema 5.7.

En primer lugar es necesario inicializar el controlador borroso, aunque, al tratarse de la misma planta, la inicialización del controlador es idéntica al caso anterior. El siguiente paso es definir las regiones en las que se dividirá el espacio de estado e inicializar las matrices $\mathbf{P}_q$. Tal como se muestra en la ecuación (5.39), se crearán tantas regiones como reglas totales tenga la planta, en este caso $s = 4$. Una vez conocido el número de regiones, s , se procede a inicializar las matrices $\mathbf{P}_q$ como matrices identidad, con $q = 1..s$.

Aunque se ha establecido el número de regiones, aún no se han definido como tales. Para ello se utiliza una división borrosa del espacio de estado a partir de las funciones de pertenencia $\varphi_q(\mathbf{x})$. Como se puede ver en el algoritmo de la figura 5.6, estas funciones de pertenencia se definen como las funciones de pertenencia de una de las salidas del controlador³.

Una vez realizado el proceso de inicialización, se procede con las fases de diseño y de ajuste del estado de equilibrio, esta vez aplicando el teorema 5.7. De esta forma se obtiene el siguiente controlador borroso:

³Hay que recordar que todas las salidas se inicializan con reglas idénticas y, por lo tanto, son idénticas funciones de pertenencia. Ver fase de inicialización del apartado 5.4.4.

SI x_1 es Gaussmf(1,464;9,987) y x_2 es Gaussmf(-1,211;9,964)
 ENTONCES $u_1 = 0,09714 + 0,5753x_1 - 0,2195x_2$
 SI x_1 es Gaussmf(-3,811;13,62) y x_2 es Gaussmf(0,06026;8,578)
 ENTONCES $u_1 = -0,02684 - 2,864x_1 + 1,148x_2$
 SI x_1 es Gaussmf(-2,982;9,986) y x_2 es Gaussmf(7,149;9,96)
 ENTONCES $u_1 = 0,1881 + 0,7072x_1 + 0,6508x_2$
 SI x_1 es Gaussmf(-3,812;13,61) y x_2 es Gaussmf(4,095;11,44)
 ENTONCES $u_1 = -0,2111 - 34,79x_1 + 0,3739x_2$
 SI x_1 es Gaussmf(1,464;9,987) y x_2 es Gaussmf(-1,211;9,964)
 ENTONCES $u_2 = -3,25 + 0,293x_1 - 0,01715x_2$
 SI x_1 es Gaussmf(-3,811;13,62) y x_2 es Gaussmf(0,06026;8,578)
 ENTONCES $u_2 = 7,839 + 0,2617x_1 - 2,995x_2$
 SI x_1 es Gaussmf(-2,982;9,986) y x_2 es Gaussmf(7,149;9,96)
 ENTONCES $u_2 = 0,2527 + 13,86x_1 + 0,08707x_2$
 SI x_1 es Gaussmf(-3,812;13,61) y x_2 es Gaussmf(4,095;11,44)
 ENTONCES $u_2 = 1,16 - 18,54x_1 + 0,6936x_2$

Las matrices $\mathbf{P}_q$, $q = 1..4$, obtenidas durante la fase de diseño son:

$$\mathbf{P}_1 = \begin{pmatrix} 2,186 & 0,398 \\ 0,398 & 0,126 \end{pmatrix}$$

$$\mathbf{P}_2 = \begin{pmatrix} 2,176 & 1,885 \\ 1,885 & 2,581 \end{pmatrix}$$

$$\mathbf{P}_3 = \begin{pmatrix} 6,834 & 9,743 \\ 9,743 & 30,865 \end{pmatrix}$$

$$\mathbf{P}_4 = \begin{pmatrix} 39,31 & -16,585 \\ -16,585 & 11,082 \end{pmatrix}.$$

En las figuras 6.29 y 6.30 se muestran, respectivamente, las salidas obtenidas para distintas condiciones iniciales y el retrato de fase temporal del sistema. Nótese como todas las trayectorias convergen al origen del espacio de estado.

Ha de tenerse en cuenta que las simulaciones presentadas en las figuras 6.29 y 6.30 han sido realizadas utilizando aplicando el controlador borroso al sistema real dado por (6.6), con lo que se demuestra que el controlador borroso diseñado estabiliza la planta real, y no sólo el modelo borroso de la planta que fue utilizado como base para su diseño.

Una vez diseñado un controlador borroso pueden estudiarse los estados de equilibrio del sistema en lazo cerrado. Este estudio puede ser útil, por ejemplo, para comprobar la existencia de un único estado equilibrio en todo el universo de discurso de las variables de estado, y que éste sea el origen.

En el apartado 4.4 se muestra el procedimiento para realizar el estudio de los estados de equilibrio del sistema en lazo cerrado. Tal como se vio en este mismo ejemplo con el estudio de los estados de equilibrio de la planta, la inclusión de la matriz jacobiana puede

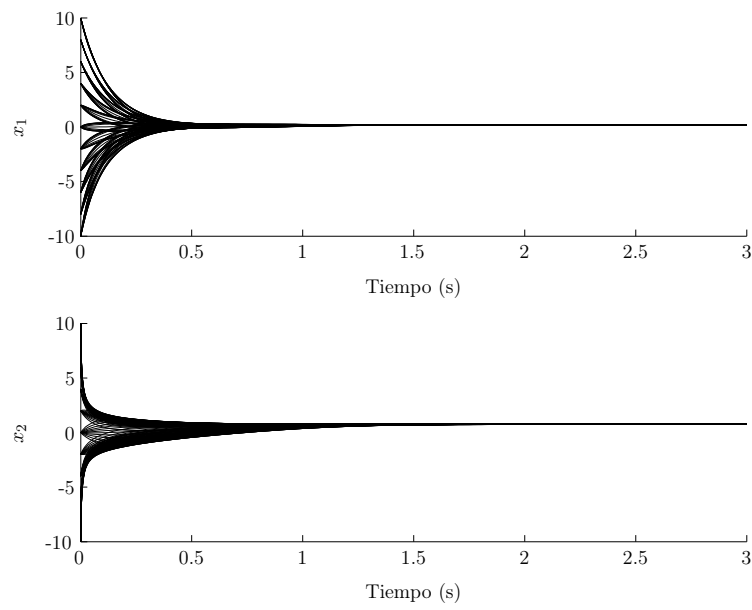


Figura 6.29: Simulación para distintas condiciones iniciales.

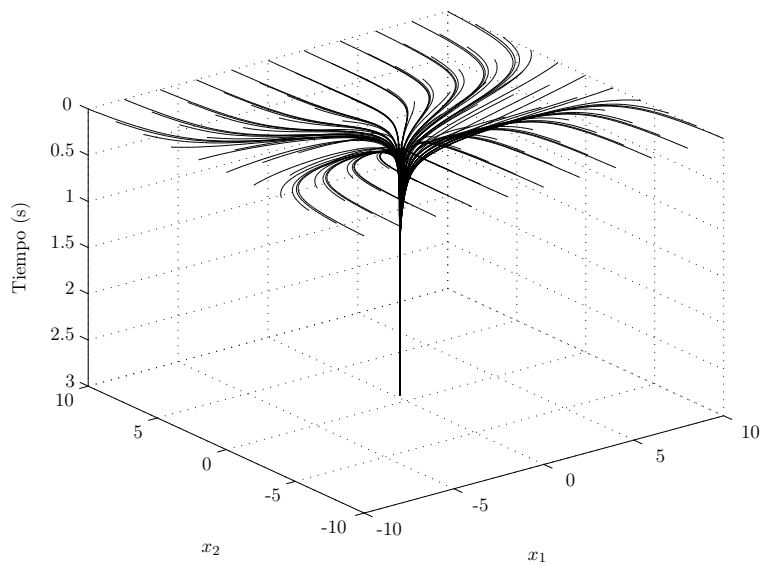


Figura 6.30: Retrato de fase temporal.

acelerar la velocidad de convergencia del algoritmo numérico empleado para el estudio, así como mejorar la precisión del resultado obtenido.

Se ha aplicado cinco veces el algoritmo numérico de minimización de la figura 4.1, partiendo en cada una de las ejecuciones de un punto de inicio $\mathbf{x}_0$ aleatorio perteneciente al universo de discurso de las variables de estado. Los resultados obtenidos se muestran en las tablas 6.13 a 6.17. Como se puede observar en dichas tablas, el estado de equilibrio obtenido se corresponde prácticamente con el origen de coordenadas. Nótese también cómo la inclusión del cálculo explícito de la matriz jacobiana suele mejorar tanto la velocidad de convergencia como la precisión del resultado obtenido en la mayoría de los casos.

	Gauss–Newton	G–N con jacobiana	Levenberg–Marquardt	L–M con jacobiana
Estado de equilibrio	$\begin{pmatrix} -3,34318e-05 \\ -0,000105908 \end{pmatrix}$	$\begin{pmatrix} -3,34332e-05 \\ -0,000105912 \end{pmatrix}$	$\begin{pmatrix} -3,34318e-05 \\ -0,000105908 \end{pmatrix}$	$\begin{pmatrix} -3,34332e-05 \\ -0,000105912 \end{pmatrix}$
Nº Iteraciones	13	13	13	13
Evaluaciones de la función objetivo	42	28	42	28
Distancia del punto obtenido al $\mathbf{0}$	3,31651e-19	9,05161e-19	3,31651e-19	9,05161e-19
Tiempo empleado (s)	1,43302	0,191314	0,059753	0,062342

Tabla 6.13: Resultados de la búsqueda de estados de equilibrio para el sistema en lazo cerrado. Ejecución 1.

	Gauss–Newton	G–N con jacobiana	Levenberg–Marquardt	L–M con jacobiana
Estado de equilibrio	$\begin{pmatrix} -3,42709e-05 \\ -0,000108738 \end{pmatrix}$	$\begin{pmatrix} -3,44296e-05 \\ -0,000109271 \end{pmatrix}$	$\begin{pmatrix} -3,42709e-05 \\ -0,000108738 \end{pmatrix}$	$\begin{pmatrix} -3,44296e-05 \\ -0,000109271 \end{pmatrix}$
Nº Iteraciones	9	9	9	9
Evaluaciones de la función objetivo	30	20	30	20
Distancia del punto obtenido al $\mathbf{0}$	8,00997e-13	1,13204e-12	8,00997e-13	1,13204e-12
Tiempo empleado (s)	0,625972	0,117236	0,048302	0,042835

Tabla 6.14: Resultados de la búsqueda de estados de equilibrio para el sistema en lazo cerrado. Ejecución 2.

La matriz jacobiana obtenida para el sistema de control borroso en lazo cerrado calculada en el origen es:

$$J(\mathbf{0}) = \begin{pmatrix} -8,23421 & 2,46510 \\ -1,05931 & -0,00137 \end{pmatrix}, \quad (6.14)$$

	Gauss–Newton	G–N con jacobiana	Levenberg–Marquardt	L–M con jacobiana
Estado de equilibrio	$\begin{pmatrix} -3,34347e-05 \\ -0,000105918 \end{pmatrix}$	$\begin{pmatrix} -3,3433e-05 \\ -0,000105912 \end{pmatrix}$	$\begin{pmatrix} -3,34347e-05 \\ -0,000105918 \end{pmatrix}$	$\begin{pmatrix} -3,3433e-05 \\ -0,000105912 \end{pmatrix}$
Nº Iteraciones	10	10	10	10
Evaluaciones de la función objetivo	33	22	33	22
Distancia del punto obtenido al 0	6,56634e-18	6,03109e-19	6,56634e-18	6,03109e-19
Tiempo empleado (s)	0,652951	0,127586	0,051015	0,055226

Tabla 6.15: Resultados de la búsqueda de estados de equilibrio para el sistema en lazo cerrado. Ejecución 3.

	Gauss–Newton	G–N con jacobiana	Levenberg–Marquardt	L–M con jacobiana
Estado de equilibrio	$\begin{pmatrix} -3,3435e-05 \\ -0,000105918 \end{pmatrix}$	$\begin{pmatrix} -3,34355e-05 \\ -0,00010592 \end{pmatrix}$	$\begin{pmatrix} -3,3435e-05 \\ -0,000105918 \end{pmatrix}$	$\begin{pmatrix} -3,34355e-05 \\ -0,00010592 \end{pmatrix}$
Nº Iteraciones	13	13	13	13
Evaluaciones de la función objetivo	42	28	42	28
Distancia del punto obtenido al 0	8,13052e-18	1,16902e-17	8,13052e-18	1,16902e-17
Tiempo empleado (s)	0,652871	0,138385	0,060831	0,063552

Tabla 6.16: Resultados de la búsqueda de estados de equilibrio para el sistema en lazo cerrado. Ejecución 4.

	Gauss–Newton	G–N con jacobiana	Levenberg–Marquardt	L–M con jacobiana
Estado de equilibrio	$\begin{pmatrix} -3,34298e-05 \\ -0,000105901 \end{pmatrix}$	$\begin{pmatrix} -3,34329e-05 \\ -0,000105912 \end{pmatrix}$	$\begin{pmatrix} -3,34298e-05 \\ -0,000105901 \end{pmatrix}$	$\begin{pmatrix} -3,34329e-05 \\ -0,000105912 \end{pmatrix}$
Nº Iteraciones	9	9	9	9
Evaluaciones de la función objetivo	30	20	30	20
Distancia del punto obtenido al 0	7,42411e-18	4,23078e-19	7,42411e-18	4,23078e-19
Tiempo empleado (s)	0,63158	0,130122	0,04691	0,051984

Tabla 6.17: Resultados de la búsqueda de estados de equilibrio para el sistema en lazo cerrado. Ejecución 5.

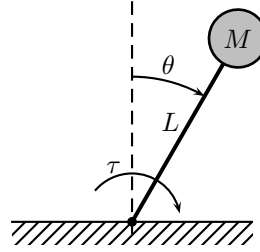


Figura 6.31: Péndulo.

cuyos autovalores son $\lambda_1 = -7,90377$ y $\lambda_2 = -0,33177$. Por lo tanto, se demuestra que el estado de equilibrio es asintóticamente estable.

En este ejemplo se ha mostrado paso a paso la metodología de diseño presentada en el capítulo 5: modelado de la planta, estudio de ésta a partir de su modelo borroso, y finalmente, diseño de un controlador borroso que estabilice el sistema de control en lazo cerrado. En la fase de diseño se ha mostrado la aplicación tanto del teorema de Krasovskii, como del teorema 5.7 como metodologías de diseño. Como se ha comprobado mediante simulación, en ambos casos el sistema resultante es asintóticamente estable.

6.2.2. Péndulo

La dinámica del péndulo de la figura 6.31 viene determinada por la ecuación (6.15), donde M es la masa del péndulo, L la longitud de la varilla, g la aceleración de la gravedad, b el coeficiente de fricción viscosa, θ el ángulo que forma el péndulo con respecto a la vertical, y τ el par aplicado sobre éste; siendo $M = 1$ Kg, $L = 0,5$ m, $g = 9,8$ m/s² y $b = 0,5$ N/m/s.

$$ML^2\ddot{\theta}(t) + b\dot{\theta}(t) - MgL\sin(\theta(t)) = \tau(t) \quad (6.15)$$

Si se considera el vector de estado:

$$\mathbf{x}(t) = \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} = \begin{pmatrix} \theta(t) \\ \dot{\theta}(t) \end{pmatrix}, \quad (6.16)$$

y la señal de control $u(t) = \tau(t)$, el sistema (6.15) puede representarse por el modelo de estado:

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= \frac{g}{L}\sin(x_1(t)) - \frac{b}{ML^2}x_2(t) + \frac{1}{ML^2}u(t). \end{aligned} \quad (6.17)$$

Puesto que la dinámica de la planta se considera desconocida, el primer paso ha de ser obtener un modelo válido de la misma.

Identificación de la planta

La elección de las variables de estado es necesariamente conocida, por lo que $\dot{x}_1(t)$ puede modelarse sin necesidad de emplear ningún tipo de algoritmo de aprendizaje. Por ejemplo, utilizando funciones de pertenencia triangulares, esta relación puede representarse por la siguiente regla borrosa:

SI x_1 es Trimf($-\infty;0;\infty$) y x_2 es Trimf($-\infty;0;\infty$) y τ es Trimf($-\infty;0;\infty$)

ENTONCES $\dot{x}_1 = x_2$,

donde Trimf representa la función de pertenencia triangular dada por la ecuación (2.9), cuyos parámetros se corresponden con los tres puntos característicos del triángulo. Nótese que la función de pertenencia Trimf($-\infty;0;\infty$) siempre toma valor 1, por lo que la regla anterior modela $\dot{x}_1(t)$ de forma exacta.

Tomando como universo de discurso de las variables de estado:

$$X \subseteq \{\mathbf{x} \mid x_1 \in [-\pi, \pi], x_2 \in [-10, 10]\},$$

y como universo de discurso de la señal de control:

$$U \subseteq \{u \mid u \in [-10^3, 10^3]\},$$

se han obtenido 1250 conjuntos de datos de entrada/salida para la segunda salidas del sistema (6.17), $\dot{x}_2(t)$, de forma que 1000 datos (el 80 %) se emplearán en el proceso de modelado y 250 (el 20 % restante) en el de validación. Nótese de nuevo que el modelo real de la planta sólo se ha utilizado para generar el conjunto de datos de entrada/salida, considerándose a partir de entonces desconocido.

La identificación de la estructura del modelo borroso, es decir, la identificación del número de reglas y de los antecedentes y consecuentes iniciales de éstas, se ha realizado mediante la aplicación del algoritmo de *clustering* de S. Chiu [246], obteniéndose 3 reglas para la inferencia de $\dot{x}_2(t)$. Las funciones de pertenencia de los antecedentes de estas reglas se muestran en la figuras 6.32 a 6.34.

Una vez ajustados los parámetros del modelo borroso mediante el algoritmo neuroborroso propuesto en [271], el modelo borroso de estado del sistema (6.15) viene representado por el siguiente conjunto de reglas:

SI x_1 es Trimf($-\infty;0;\infty$) y x_2 es Trimf($-\infty;0;\infty$) y τ es Trimf($-\infty;0;\infty$)

ENTONCES $\dot{x}_1 = x_2$

SI x_1 es Gaussmf(1,013;1,509) y x_2 es Gaussmf(1,123;10,06) y τ es Gaussmf(106,6;924,4)

ENTONCES $\dot{x}_2 = 0,9452 - 2,6 * x_1 - 1,5 * x_2 + 4,015 * \tau$

SI x_1 es Gaussmf(1,463;0,9176) y x_2 es Gaussmf(-1,33;9,816) y τ es Gaussmf(-782,7;924,4)

ENTONCES $\dot{x}_2 = 65,57 - 2,805 * x_1 - 1,403 * x_2 + 4,038 * \tau$

SI x_1 es Gaussmf(-0,876;0,8156) y x_2 es Gaussmf(8,382;12,32) y τ es Gaussmf(944,3;924,4)

ENTONCES $\dot{x}_2 = -31,73 - 0,4918 * x_1 - 1,379 * x_2 + 4,013 * \tau$

La función Gaussmf representa la función de pertenencia gaussiana dada por la ecuación (2.12), cuyos parámetros son el centro y el ancho, respectivamente.

El error cuadrático medio obtenido al finalizar el proceso de modelado de $\dot{x}_2(t)$ ha sido de 1,613. Al validar el modelo mediante los 250 datos restantes, se ha obtenido un error

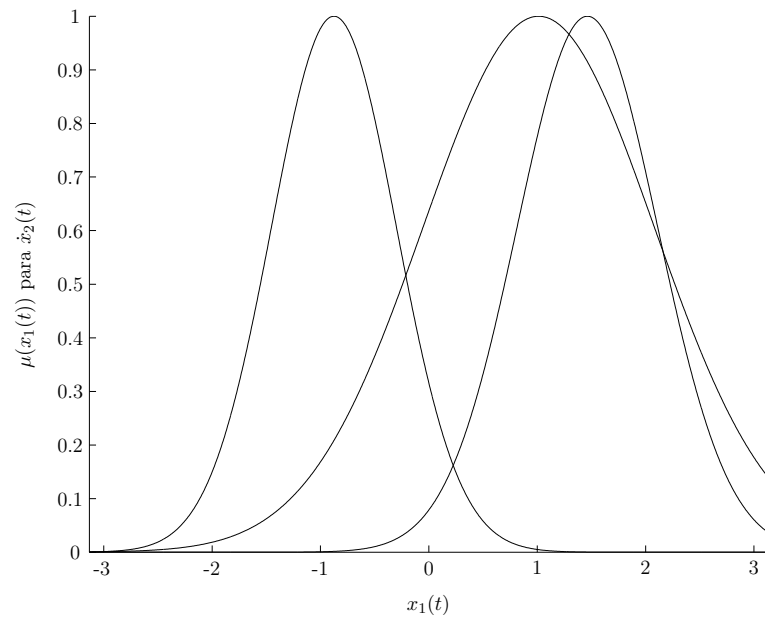


Figura 6.32: Funciones de pertenencia para la entrada $x_1(t)$ en la inferencia de $\dot{x}_2(t)$.

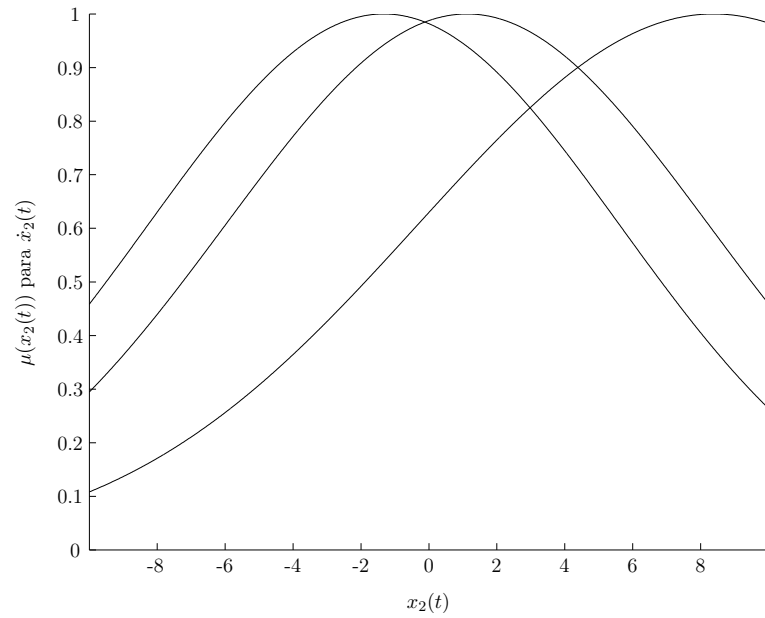


Figura 6.33: Funciones de pertenencia para la entrada $x_2(t)$ en la inferencia de $\dot{x}_2(t)$.

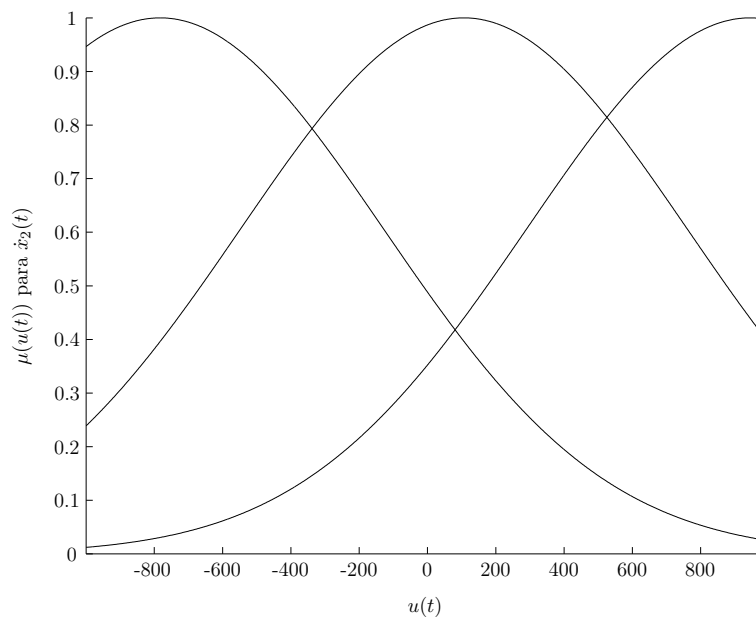


Figura 6.34: Funciones de pertenencia para la entrada $u(t)$ en la inferencia de $\dot{x}_2(t)$.

cuadrático medio de 2,729, por lo que dicho modelo puede considerarse válido dentro del universo de discurso en el que ha sido realizado. La representación gráfica de los errores presentados durante la validación del modelo de $\dot{x}_2(t)$ pueden verse en las figuras 6.35 y 6.36.

Estudio de la planta a partir de su modelo borroso

Una vez ha sido validado el modelo borroso de la planta, éste puede ser estudiado mediante el algoritmo presentado en la sección 4.4. Para ello se ha aplicado seis veces el algoritmo de la figura 4.1 al modelo borroso de la planta, partiendo en cada una de las ejecuciones de un punto de inicio $\mathbf{x}_0$ aleatorio perteneciente al universo de discurso de las variables de estado. Los resultados obtenidos, tanto para el método de Gauss–Newton como para el método de Levenberg–Marquardt, aplicando el cálculo explícito de la matriz jacobiana y sin aplicarlo, se muestran en las tablas 6.18 a 6.23. En todos los casos se ha supuesto $\mathbf{u}(t) = \mathbf{0}$.

Analizando los resultados de las tablas 6.18 a 6.23 puede comprobarse que se han obtenido tres estados de equilibrio, correspondientes aproximadamente con los puntos $[0, 0]^T$, $[\pi, 0]^T$ y $[-\pi, 0]^T$. Estos puntos son efectivamente los estados de equilibrio que presenta la planta real, pero han sido obtenidos tan sólo a partir de su modelo borroso, sin conocimiento previo de ésta.

Con respecto a la respuesta de los distintos algoritmos aplicados para el cálculo de los estados de equilibrio, en este caso no se puede concluir que la utilización del cálculo explícito de la matriz jacobiana haya supuesto una mejora en la precisión de los resultados,

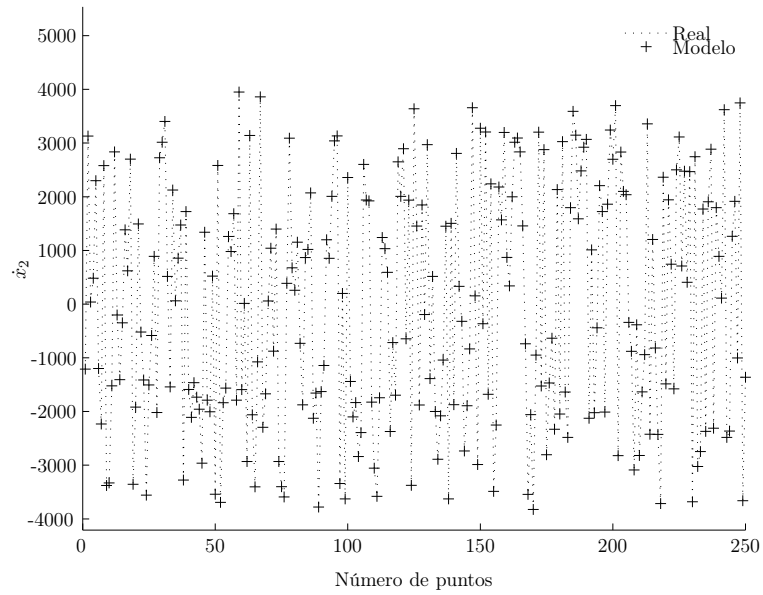


Figura 6.35: Validación del modelo borroso para $\dot{x}_2$.

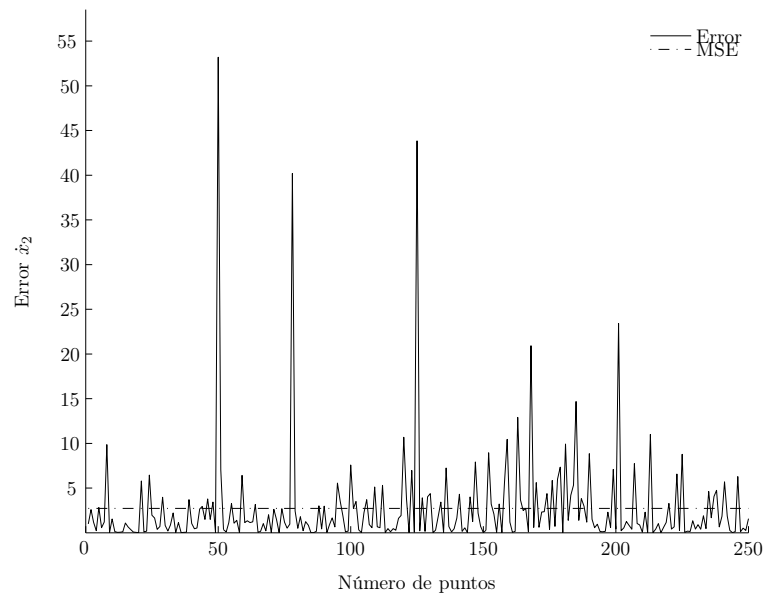


Figura 6.36: Error de modelado durante la validación de $\dot{x}_2$.

	Gauss–Newton	G–N con jacobiana	Levenberg–Marquardt	L–M con jacobiana
Estado de equilibrio	$\begin{pmatrix} -0,0655727 \\ 0 \end{pmatrix}$	$\begin{pmatrix} -0,0655727 \\ 0 \end{pmatrix}$	$\begin{pmatrix} -0,0655727 \\ 0 \end{pmatrix}$	$\begin{pmatrix} -0,0655727 \\ 0 \end{pmatrix}$
Nº Iteraciones	5	5	5	5
Evaluaciones de la función objetivo	18	12	18	12
Distancia del punto obtenido al 0	4,15428e-26	5,69144e-26	4,15428e-26	5,69144e-26
Tiempo empleado (s)	1,30448	0,140312	0,039413	0,039227

Tabla 6.18: Resultados de la búsqueda de estados de equilibrio. Ejecución 1.

	Gauss–Newton	G–N con jacobiana	Levenberg–Marquardt	L–M con jacobiana
Estado de equilibrio	$\begin{pmatrix} 3,13936 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 3,13936 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 3,13936 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 3,13936 \\ 0 \end{pmatrix}$
Nº Iteraciones	6	6	6	6
Evaluaciones de la función objetivo	21	14	21	14
Distancia del punto obtenido al 0	8,66094e-30	1,06096e-28	8,66094e-30	1,06096e-28
Tiempo empleado (s)	0,670333	0,110325	0,044947	0,045406

Tabla 6.19: Resultados de la búsqueda de estados de equilibrio. Ejecución 2.

	Gauss–Newton	G–N con jacobiana	Levenberg–Marquardt	L–M con jacobiana
Estado de equilibrio	$\begin{pmatrix} -3,01697 \\ 0 \end{pmatrix}$	$\begin{pmatrix} -3,01697 \\ 0 \end{pmatrix}$	$\begin{pmatrix} -3,01697 \\ 0 \end{pmatrix}$	$\begin{pmatrix} -3,01697 \\ 0 \end{pmatrix}$
Nº Iteraciones	9	9	9	9
Evaluaciones de la función objetivo	28	20	28	20
Distancia del punto obtenido al 0	5,42175e-16	5,41697e-16	5,42175e-16	5,41697e-16
Tiempo empleado (s)	0,660155	0,112981	0,060938	0,063303

Tabla 6.20: Resultados de la búsqueda de estados de equilibrio. Ejecución 3.

	Gauss–Newton	G–N con jacobiana	Levenberg–Marquardt	L–M con jacobiana
Estado de equilibrio	$\begin{pmatrix} 3,13936 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 3,13936 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 3,13936 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 3,13936 \\ 0 \end{pmatrix}$
Nº Iteraciones	4	4	4	4
Evaluaciones de la función objetivo	15	10	15	10
Distancia del punto obtenido al 0	1,10738e-16	1,10306e-16	1,10738e-16	1,10306e-16
Tiempo empleado (s)	0,65582	0,103466	0,039713	0,047287

Tabla 6.21: Resultados de la búsqueda de estados de equilibrio. Ejecución 4.

	Gauss–Newton	G–N con jacobiana	Levenberg–Marquardt	L–M con jacobiana
Estado de equilibrio	$\begin{pmatrix} -0,0655727 \\ 0 \end{pmatrix}$	$\begin{pmatrix} -0,0655727 \\ 0 \end{pmatrix}$	$\begin{pmatrix} -0,0655727 \\ 0 \end{pmatrix}$	$\begin{pmatrix} -0,0655727 \\ 0 \end{pmatrix}$
Nº Iteraciones	9	9	9	9
Evaluaciones de la función objetivo	28	20	28	20
Distancia del punto obtenido al 0	8,9211e-32	3,10851e-31	8,9211e-32	3,10851e-31
Tiempo empleado (s)	0,65535	0,122932	0,059896	0,059501

Tabla 6.22: Resultados de la búsqueda de estados de equilibrio. Ejecución 5.

	Gauss–Newton	G–N con jacobiana	Levenberg–Marquardt	L–M con jacobiana
Estado de equilibrio	$\begin{pmatrix} 3,13936 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 3,13936 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 3,13936 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 3,13936 \\ 0 \end{pmatrix}$
Nº Iteraciones	5	5	5	5
Evaluaciones de la función objetivo	18	12	18	12
Distancia del punto obtenido al 0	1,75384e-28	1,06096e-28	1,75384e-28	1,06096e-28
Tiempo empleado (s)	0,641308	0,099564	0,05429	0,040814

Tabla 6.23: Resultados de la búsqueda de estados de equilibrio. Ejecución 6.

pero sí en la velocidad de cálculo al aplicar el método de Gauss-Newton, no así en al aplicar Levenberg-Marquardt.

Al linealizar el modelo borroso mediante la aplicación del algoritmo 4.2 en torno al estado de equilibrio $\mathbf{x}^T = (-0,63031, 0)^T$, con $\tau = 0$, se obtiene el sistema lineal:

$$\dot{\mathbf{x}}(t) \simeq \begin{pmatrix} 0 & 1 \\ 17,931 & -1,953 \end{pmatrix} \mathbf{x}(t) + \begin{pmatrix} 0 \\ 4,003 \end{pmatrix} \tau(t), \quad (6.18)$$

cuyos autovalores son $\lambda_1 = 3,369$ y $\lambda_2 = -5,322$. Por lo tanto, se puede afirmar que el sistema es inestable en lazo abierto, y el estado de equilibrio se comporta como un punto de silla.

La linealización del sistema real en torno al estado de equilibrio, $\mathbf{x}^T = (0, 0)^T$, con $\tau = 0$, viene determinada por (6.19), siendo sus autovalores $\lambda_1 = 3,534$ y $\lambda_2 = -5,534$. Comparando estos resultados con los obtenidos mediante el modelo borroso, se puede comprobar la semejanza tanto del modelo lineal como de los autovalores obtenidos; además, el análisis previo realizado únicamente mediante la utilización del modelo borroso es perfectamente válido, ya que efectivamente el sistema es inestable en lazo abierto.

$$\dot{\mathbf{x}}(t) \simeq \begin{pmatrix} 0 & 1 \\ 19,558 & -2 \end{pmatrix} \mathbf{x}(t) + \begin{pmatrix} 0 \\ 4 \end{pmatrix} \tau(t) \quad (6.19)$$

Si se procede ahora a linealizar el modelo borroso en torno al estado de equilibrio $\mathbf{x}^T = (3,13936, 0)^T$, con $\tau = 0$, se obtiene el sistema lineal:

$$\dot{\mathbf{x}}(t) \simeq \begin{pmatrix} 0 & 1 \\ -16,161 & -1,808 \end{pmatrix} \mathbf{x}(t) + \begin{pmatrix} 0 \\ 4,004 \end{pmatrix} \tau(t), \quad (6.20)$$

cuyos autovalores son $\lambda_{1,2} = -0,904 \pm 3,917i$, lo que implica que el estado de equilibrio es asintóticamente estable. En realidad no se puede confiar en exceso en los autovalores obtenidos, ya que su proximidad a cero hace que cualquier error en el modelo pueda modificar el signo de la parte real y, por tanto, su estabilidad. Si se compara el sistema anterior, linealizado a partir del modelo borroso, con el sistema linealizado real representado en (6.21), se puede comprobar que la diferencia existente en la determinación del modelo linealizado es relativamente pequeña; sin embargo, podría ser suficiente para provocar un cambio de signo en la parte real de los autovalores. En este caso no es así, ya que los autovalores del sistema real linealizado en el punto $\mathbf{x}^T = (\pi, 0)^T$, con $\tau = 0$, son $\lambda_{1,2} = -1 \pm 4,313i$, muy similares a los obtenido a partir del modelo borroso y también implican la estabilidad del estado de equilibrio.

$$\dot{\mathbf{x}}(t) \simeq \begin{pmatrix} 0 & 1 \\ -19,6 & -2 \end{pmatrix} \mathbf{x}(t) + \begin{pmatrix} 0 \\ 4 \end{pmatrix} \tau(t) \quad (6.21)$$

Finalmente, si se procede a linealizar el modelo borroso en torno al estado de equilibrio $\mathbf{x}^T = (3,13936, 0)^T$, con $\tau = 0$, se obtiene el sistema lineal:

$$\dot{\mathbf{x}}(t) \simeq \begin{pmatrix} 0 & 1 \\ -21,847 & -2,073 \end{pmatrix} \mathbf{x}(t) + \begin{pmatrix} 0 \\ 4,001 \end{pmatrix} \tau(t), \quad (6.22)$$

cuyos autovalores son $\lambda_{1,2} = -1,036 \pm 4,558i$. Al igual que sucede en el caso anterior, es arriesgado afirmar que el estado de equilibrio es asintóticamente estable, ya que no se puede confiar en los autovalores obtenidos debido a su proximidad a cero. Si se linealiza el modelo real del sistema en el estado de equilibrio $\mathbf{x}^T = (-\pi, 0)^T$, con $\tau = 0$, se obtiene el sistema lineal:

$$\dot{\mathbf{x}}(t) \simeq \begin{pmatrix} 0 & 1 \\ -19,448 & -2 \end{pmatrix} \mathbf{x}(t) + \begin{pmatrix} 0 \\ 4 \end{pmatrix} \tau(t), \quad (6.23)$$

cuyos autovalores son $\lambda_{1,2} = -1 \pm 4,295i$. Nuevamente, tanto el modelo linealizado como los autovalores son similares a los obtenidos a partir del modelo borroso, lo que viene a reafirmar que dicho modelo es bastante preciso.

Diseño de un controlador borroso estable

Utilizando el algoritmo presentado en la sección 5.4.4 se procede a diseñar un controlador borroso que garantice la estabilidad del sistema (6.15) en torno al estado de equilibrio inestable situado en el origen.

Se crean tantas regiones como reglas totales tenga la planta, en este caso $s = 4$, y se inicializan las matrices $\mathbf{P}_q$, con $q = 1..4$, como la identidad. Tal como se establece en el algoritmo 5.6, las regiones $\varphi_q(\mathbf{x})$ se definen a partir de las funciones de pertenencia del controlador. Una vez concluidas las fases de diseño y de ajuste del estado de equilibrio, se obtiene el siguiente controlador borroso:

```

SI  $x_1$  es Trimf( $-\infty;0;\infty$ ) y  $x_2$  es Trimf( $-\infty;0;\infty$ )
ENTONCES  $\tau = 5,353-7,417*x_1-1,85*x_2$ 
SI  $x_1$  es Gaussmf(1,013;1,509) y  $x_2$  es Gaussmf(1,123;10,06)
ENTONCES  $\tau = -8,169-0,4213*x_1-0,5807*x_2$ 
SI  $x_1$  es Gaussmf(1,463;0,9176) y  $x_2$  es Gaussmf(-1,33;9,816)
ENTONCES  $\tau = -20,31-3,055*x_1-2,173*x_2$ 
SI  $x_1$  es Gaussmf(-0,876;0,8156) y  $x_2$  es Gaussmf(8,382;12,32)
ENTONCES  $\tau = 4,055-0,9655*x_1+0,06538*x_2$ 

```

Las matrices $\mathbf{P}_q$, $q = 1..4$, obtenidas durante la fase de diseño son:

$$\mathbf{P}_1 = \begin{pmatrix} 0,467 & 0,009 \\ 0,009 & 0,011 \end{pmatrix}$$

$$\mathbf{P}_2 = \begin{pmatrix} 0,011 & 0 \\ 0 & 0,0109 \end{pmatrix}$$

$$\mathbf{P}_3 = \begin{pmatrix} 0,024 & 0 \\ 0 & 0,012 \end{pmatrix}$$

$$\mathbf{P}_4 = \begin{pmatrix} 0,01 & 0 \\ 0,001 & 0,0352 \end{pmatrix}$$

Se puede comprobar que el algoritmo de inicilización (ver figura 5.6) ha generado 4 reglas para el controlador borroso procedentes del modelo borroso de la planta mostrado en la página 190. Todos los parámetros adaptables de los consecuentes se inician a cero, y posteriormente se ajustan a través del algoritmo de ajuste (ver figura 5.8) hasta que quede satisfecha la condición de estabilidad del teorema 5.7.

En las figuras 6.37 y 6.38 se puede observar la evolución del sistema diseñado partiendo de distintas condiciones iniciales. En dichas figuras puede comprobarse que efectivamente, todas las trayectorias que no abandonan el universo de discurso de las variables de estado tienden al estado de equilibrio situado en el origen. La estabilidad de aquellas trayectorias que abandonan el universo de discurso de las variables de estado, mostradas en trazo punteado, no puede garantizarse, ya que el modelo borroso de la planta no tiene porqué ser válido fuera del rango de valores para el que fue obtenido. Por lo tanto, al no ser posible garantizar el cumplimiento del teorema 5.7 para dichas trayectorias, no puede asegurarse formalmente su estabilidad.

Ha de tenerse en cuenta que las simulaciones presentadas en las figuras 6.37 y 6.38 han sido realizadas utilizando aplicando el controlador borroso al sistema real dado por (6.17), con lo que se demuestra que el controlador borroso diseñado estabiliza la planta real, y no sólo el modelo borroso de la planta que fue utilizado como base para su diseño.

En la figura 6.39 puede verse con más claridad las trayectorias que abandonan el universo de discurso de las variables de estado, pese a iniciarse dentro de éste. La evolución real del sistema es completamente imprevisible una vez está fuera del universo de discurso para el que es válido el modelo, ya que la simulación sólo muestra el comportamiento extrapolado de dicho modelo fuera de su rango de validez.

Aunque la estabilidad del estado de equilibrio situado en el origen se observa intuitivamente en la figura 6.39, ésta puede estudiarse a través de los autovalores de la matriz jacobiana del modelo borroso de lazo cerrado⁴.

Aplicando el algoritmo de la figura 4.5 al sistema de control borroso en lazo cerrado en el origen, se obtiene la siguiente matriz:

$$J(\mathbf{0}) = \begin{pmatrix} 0 & 1 \\ -25,04 & -6,850 \end{pmatrix} \quad (6.24)$$

cuyos autovalores son $-3,425 \pm 3,648i$. Estos autovalores se corresponden con un foco estable, como efectivamente puede comprobarse en la figura 6.39.

⁴Nótese que esta demostración tan sólo es válida para las cercanías del estado de equilibrio. La estabilidad queda garantizada por la aplicación del teorema 5.7 durante la fase de diseño del controlador borroso.

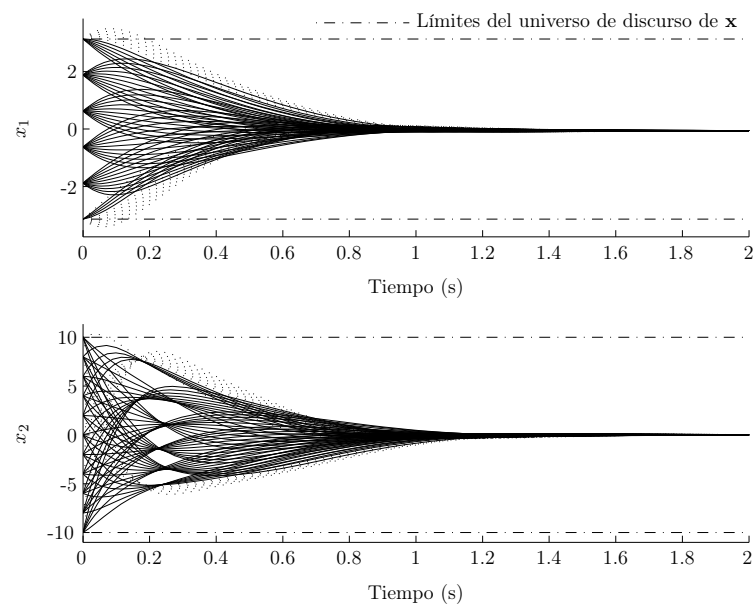


Figura 6.37: Simulación para distintas condiciones iniciales.

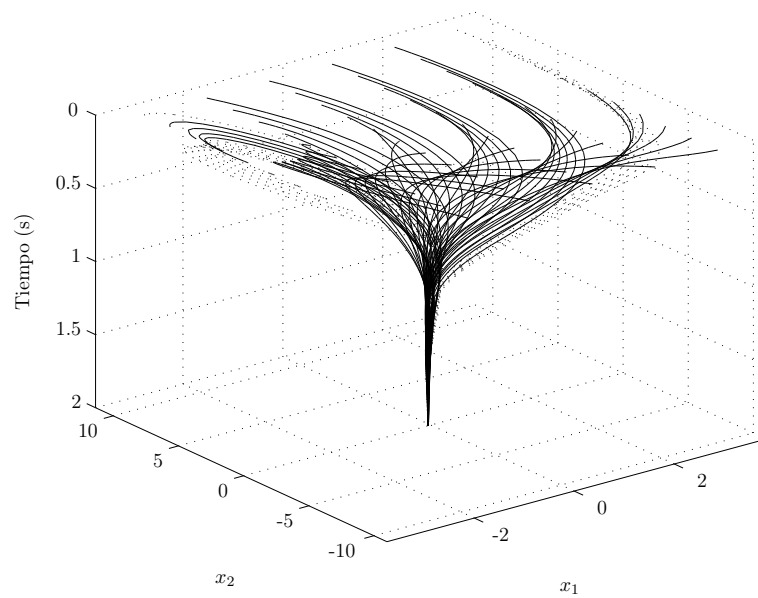


Figura 6.38: Retrato de fase temporal.

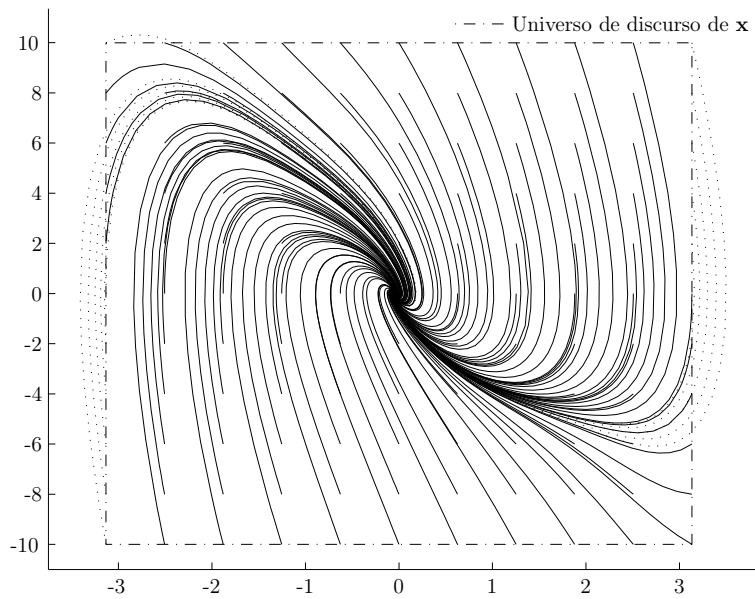


Figura 6.39: Retrato de fase.

Por último, es posible estudiar los estados de equilibrio del sistema en lazo cerrado como método de garantizar que sólo exista uno dentro del universo de discurso de las variables de estado. Para ello se ha aplicado cinco veces el algoritmo numérico de minimización de la figura 4.1, partiendo en cada una de las ejecuciones de un punto de inicio $\mathbf{x}_0$ aleatorio perteneciente al universo de discurso de las variables de estado. En los resultados obtenidos, mostrados en las tablas 6.24 a 6.28, se puede comprobar que efectivamente sólo existe un estado de equilibrio prácticamente situado sobre el origen de coordenadas.

	Gauss–Newton	G–N con jacobiana	Levenberg–Marquardt	L–M con jacobiana
Estado de equilibrio	$\begin{pmatrix} -2,49024e-11 \\ 0 \end{pmatrix}$	$\begin{pmatrix} -2,49027e-11 \\ 0 \end{pmatrix}$	$\begin{pmatrix} -2,49024e-11 \\ 0 \end{pmatrix}$	$\begin{pmatrix} -2,49027e-11 \\ 0 \end{pmatrix}$
Nº Iteraciones	6	6	6	6
Evaluaciones de la función objetivo	21	14	21	14
Distancia del punto obtenido al $\mathbf{0}$	3.8844e-28	1.63687e-28	3.8844e-28	1.63687e-28
Tiempo empleado (s)	0.889011	0.134602	0.041155	0.050629

Tabla 6.24: Resultados de la búsqueda de estados de equilibrio para el sistema en lazo cerrado. Ejecución 1.

	Gauss-Newton	G-N con jacobiana	Levenberg-Marquardt	L-M con jacobiana
Estado de equilibrio	$\begin{pmatrix} -2,46137e-11 \\ 0 \end{pmatrix}$	$\begin{pmatrix} -2,46244e-11 \\ 0 \end{pmatrix}$	$\begin{pmatrix} -2,46137e-11 \\ 0 \end{pmatrix}$	$\begin{pmatrix} -2,46244e-11 \\ 0 \end{pmatrix}$
Nº Iteraciones	13	13	13	13
Evaluaciones de la función objetivo	36	28	36	28
Distancia del punto obtenido al 0	5.25857e-23	4.8746e-23	5.25857e-23	4.8746e-23
Tiempo empleado (s)	0.653583	0.124124	0.061951	0.081563

Tabla 6.25: Resultados de la búsqueda de estados de equilibrio para el sistema en lazo cerrado. Ejecución 2.

	Gauss-Newton	G-N con jacobiana	Levenberg-Marquardt	L-M con jacobiana
Estado de equilibrio	$\begin{pmatrix} -2,48627e-11 \\ 0 \end{pmatrix}$	$\begin{pmatrix} -2,48643e-11 \\ 0 \end{pmatrix}$	$\begin{pmatrix} -2,48627e-11 \\ 0 \end{pmatrix}$	$\begin{pmatrix} -2,48643e-11 \\ 0 \end{pmatrix}$
Nº Iteraciones	15	15	15	15
Evaluaciones de la función objetivo	40	32	40	32
Distancia del punto obtenido al 0	1.02971e-24	9.54464e-25	1.02971e-24	9.54464e-25
Tiempo empleado (s)	0.758528	0.143927	0.07344	0.079727

Tabla 6.26: Resultados de la búsqueda de estados de equilibrio para el sistema en lazo cerrado. Ejecución 3.

	Gauss-Newton	G-N con jacobiana	Levenberg-Marquardt	L-M con jacobiana
Estado de equilibrio	$\begin{pmatrix} 1,13555e-09 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1,13544e-09 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1,13555e-09 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1,13544e-09 \\ 0 \end{pmatrix}$
Nº Iteraciones	6	6	6	6
Evaluaciones de la función objetivo	21	14	21	14
Distancia del punto obtenido al 0	8.44552e-16	8.44392e-16	8.44552e-16	8.44392e-16
Tiempo empleado (s)	0.592863	0.097751	0.035944	0.042486

Tabla 6.27: Resultados de la búsqueda de estados de equilibrio para el sistema en lazo cerrado. Ejecución 4.

	Gauss–Newton	G–N con jacobiana	Levenberg–Marquardt	L–M con jacobiana
Estado de equilibrio	$\begin{pmatrix} 6,96902e-10 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 6,96546e-10 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 6,96902e-10 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 6,96546e-10 \\ 0 \end{pmatrix}$
Nº Iteraciones	10	10	10	10
Evaluaciones de la función objetivo	31	22	31	22
Distancia del punto obtenido al 0	3.26748e-16	3.26426e-16	3.26748e-16	3.26426e-16
Tiempo empleado (s)	0.715253	0.123721	0.052973	0.095405

Tabla 6.28: Resultados de la búsqueda de estados de equilibrio para el sistema en lazo cerrado. Ejecución 5.

A partir de datos de entrada/salida se ha conseguido realizar el estudio de una planta inicialmente desconocida, obteniendo sus tres estados de equilibrio. Posteriormente se ha diseñado un controlador que estabilice el sistema de control en lazo cerrado. Como se ha comprobado, el sistema resultante es asintóticamente estable posee un único estado de equilibrio en el universo de discurso de las variables de estado.

6.2.3. Balancín

En la figura 6.40 se muestra un sistema de balancín. Si se supone que la masa de la bola y el rozamiento de ésta con el balancín son despreciables, la dinámica del sistema viene representada por la ecuación (6.25), siendo $\theta(t)$ el ángulo que forma el balancín con la horizontal y $\dot{\theta}(t)$ su velocidad, $d(t)$ la distancia existente entre la bola y el centro de giro del balancín y $\dot{d}(t)$ su velocidad, $J = 1 \text{ Kg}\cdot\text{m}^2$ el momento de inercia del balancín, $b = 1 \text{ Kg}\cdot\text{m}^2/\text{s}$ el coeficiente de fricción viscosa, y $g = 9,8 \text{ m/s}^2$ la aceleración de la gravedad.

$$\begin{aligned}\ddot{d}(t) &= -g \sin(\theta(t)) \\ J\ddot{\theta}(t) &= \tau(t) - b\dot{\theta}(t)\end{aligned}\tag{6.25}$$

Si se considera el vector de estado

$$\mathbf{x}(t) = \begin{pmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{pmatrix} = \begin{pmatrix} d(t) \\ \dot{d}(t) \\ \theta(t) \\ \dot{\theta}(t) \end{pmatrix},\tag{6.26}$$

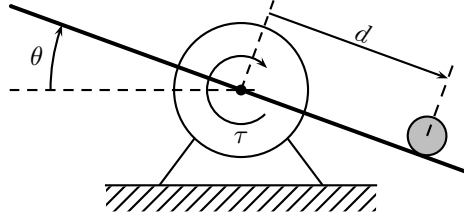


Figura 6.40: *Balancín.*

y la señal de control $u(t) = \tau(t)$, el sistema (6.25) puede representarse por el modelo de estado:

$$\begin{aligned}\dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= -g \sin(x_1(t)) \\ \dot{x}_3(t) &= x_4(t) \\ \dot{x}_4(t) &= -\frac{b}{J}x_1(t) + \frac{u(t)}{J}.\end{aligned}\tag{6.27}$$

Al igual que en los casos anteriores, la dinámica de la planta se considera desconocida, por lo que antes de poder realizar ningún tipo de análisis y/o diseño será necesario obtener un modelo de la misma.

Identificación de la planta

Tomando como universo de discurso de las variables de estado:

$$X \subseteq \{\mathbf{x} \mid x_1 \in [-10, 10], x_2 \in [-10, 10], x_3 \in [-80^\circ, 80^\circ], x_4 \in [-10, 10]\},$$

y como universo de discurso de la señal de control:

$$U \subseteq \{u \mid u \in [-10^3, 10^3]\},$$

se han obtenido 1250 conjuntos de datos de entrada/salida para cada una de las salidas del sistema (6.27), de forma que 1000 datos (el 80 %) se emplearán en el proceso de modelado y 250 (el 20 % restante) en el de validación.

Aplicando el algoritmo de *clustering* de S. Chiu [246] se ha obtenido la estructura del modelo borroso para cada una de las cuatro variables de salida del modelo de estado de la planta, obteniéndose 3 reglas para la inferencia de $\dot{x}_1(t)$, 7 reglas para la de $\dot{x}_2(t)$, 3 reglas para la de $\dot{x}_3(t)$ y, finalmente, 2 reglas para la de $\dot{x}_4(t)$, por lo que el modelo borroso de estado del sistema (6.27) consta en total de 15 reglas.

Las funciones de pertenencia de los antecedentes de las 3 reglas que modelan $\dot{x}_1(t)$ se muestran en las figuras 6.41 a 6.45, las correspondientes a las 7 reglas de $\dot{x}_2(t)$ en las figuras 6.46 a 6.50, las de los antecedentes de las 3 reglas de $\dot{x}_3(t)$ en la figuras 6.51 a 6.55,

y, finalmente, las funciones de pertenencia de los antecedentes de las 2 reglas que modelan $\dot{x}_4(t)$ se muestran en las figuras 6.56 a 6.60.

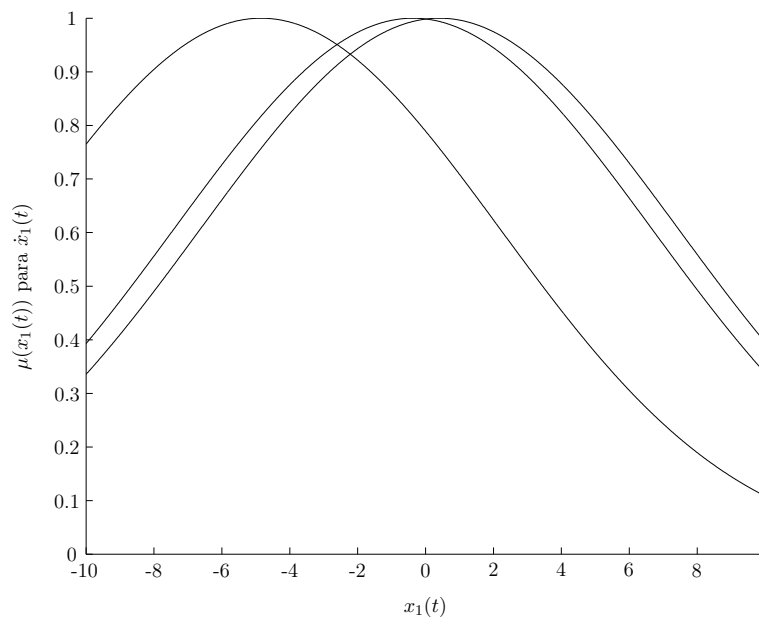


Figura 6.41: Funciones de pertenencia para la entrada $x_1(t)$ en la inferencia de $\dot{x}_1(t)$.

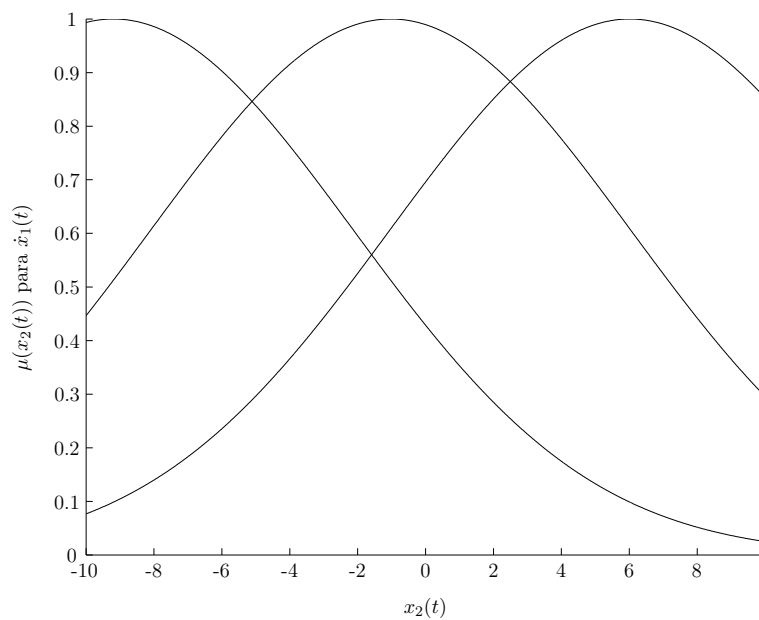


Figura 6.42: Funciones de pertenencia para la entrada $x_2(t)$ en la inferencia de $\dot{x}_1(t)$.

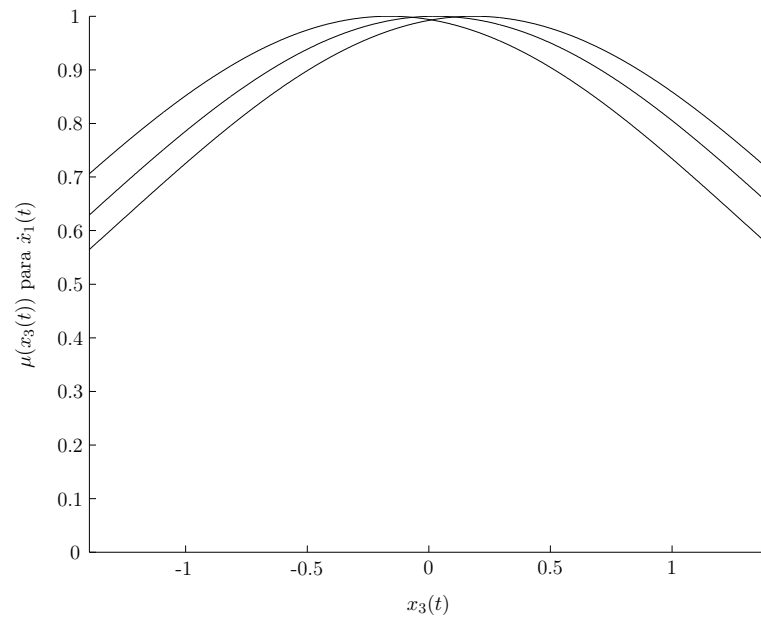


Figura 6.43: Funciones de pertenencia para la entrada $x_3(t)$ en la inferencia de $\dot{x}_1(t)$.

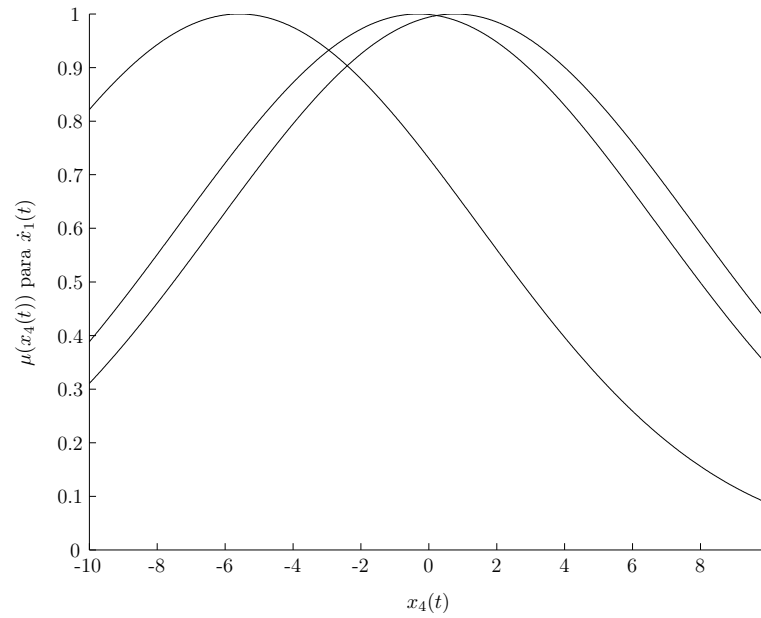


Figura 6.44: Funciones de pertenencia para la entrada $x_4(t)$ en la inferencia de $\dot{x}_1(t)$.

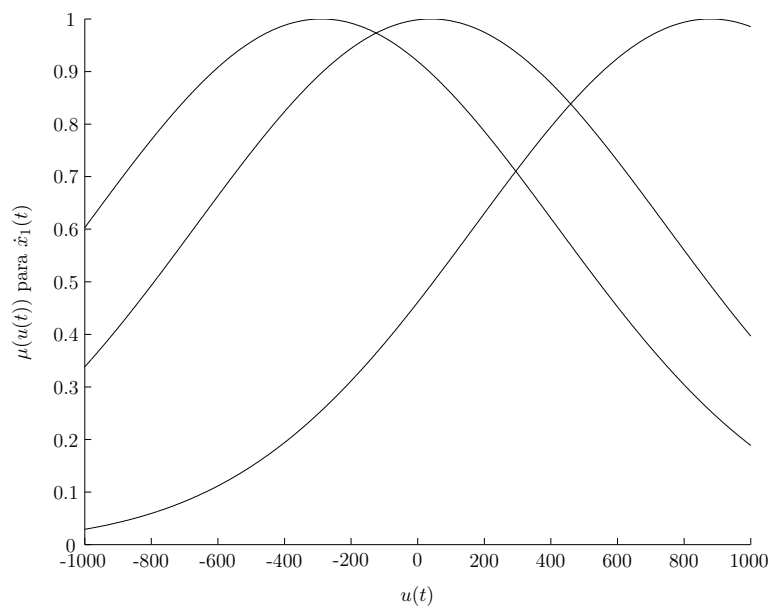


Figura 6.45: Funciones de pertenencia para la entrada $u(t)$ en la inferencia de $\dot{x}_1(t)$.

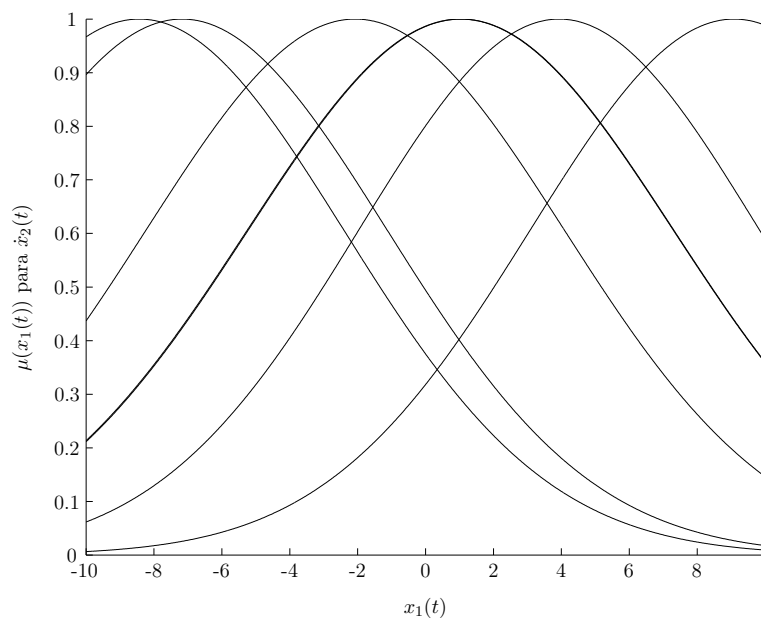


Figura 6.46: Funciones de pertenencia para la entrada $x_1(t)$ en la inferencia de $\dot{x}_2(t)$.

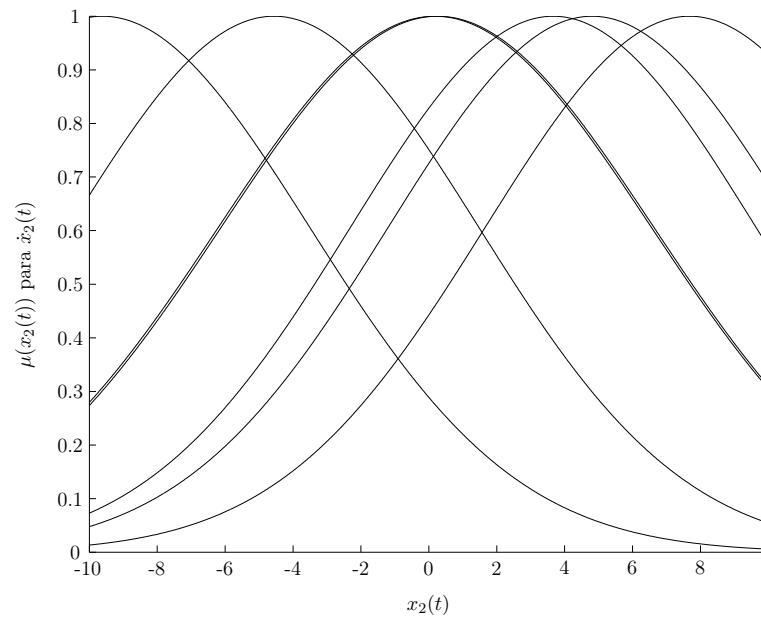


Figura 6.47: Funciones de pertenencia para la entrada $x_2(t)$ en la inferencia de $\dot{x}_2(t)$.

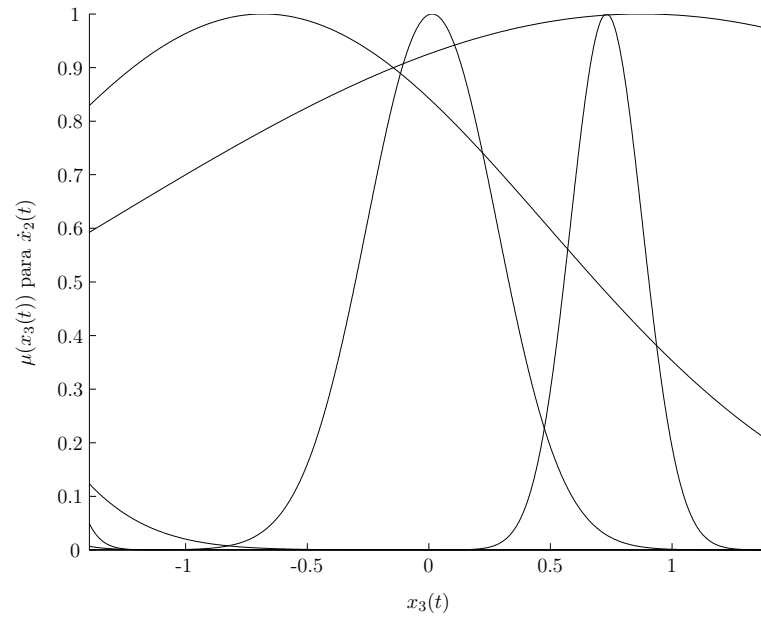


Figura 6.48: Funciones de pertenencia para la entrada $x_3(t)$ en la inferencia de $\dot{x}_2(t)$.

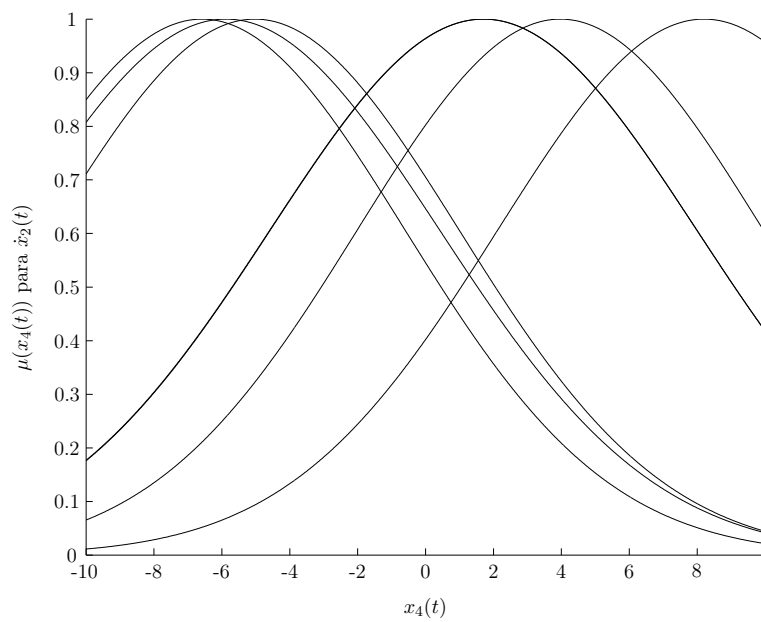


Figura 6.49: Funciones de pertenencia para la entrada $x_4(t)$ en la inferencia de $\dot{x}_2(t)$.

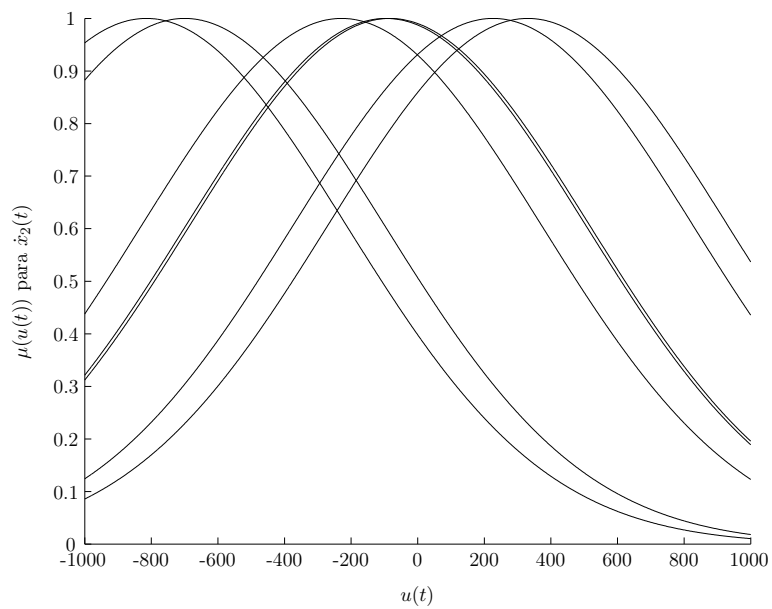


Figura 6.50: Funciones de pertenencia para la entrada $u(t)$ en la inferencia de $\dot{x}_2(t)$.

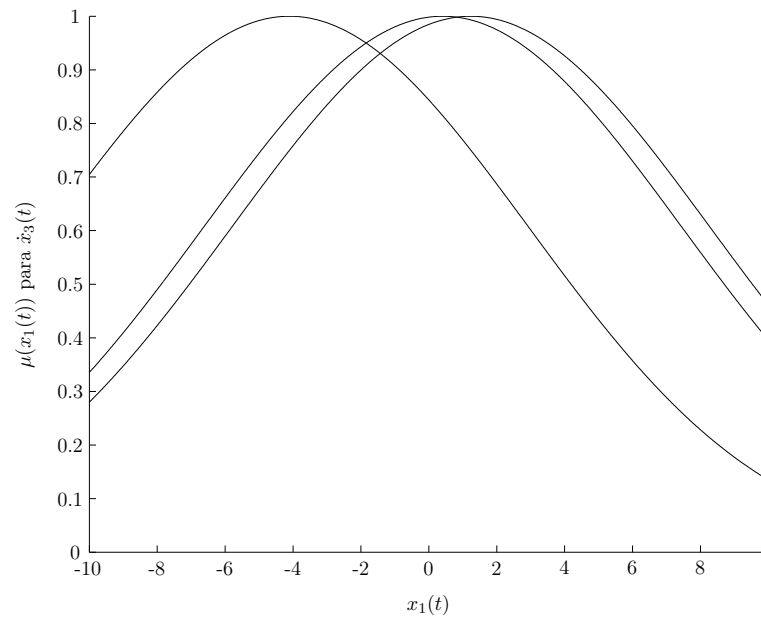


Figura 6.51: Funciones de pertenencia para la entrada $x_1(t)$ en la inferencia de $\dot{x}_3(t)$.

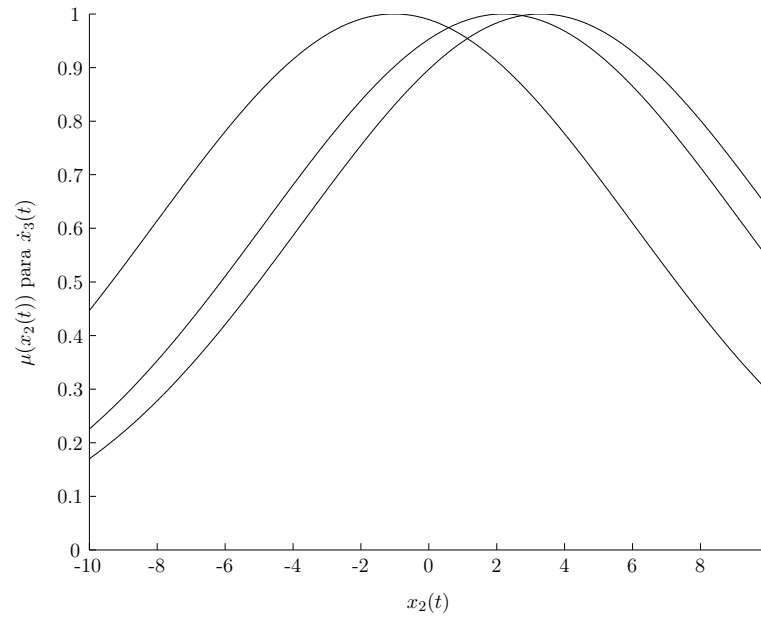


Figura 6.52: Funciones de pertenencia para la entrada $x_2(t)$ en la inferencia de $\dot{x}_3(t)$.

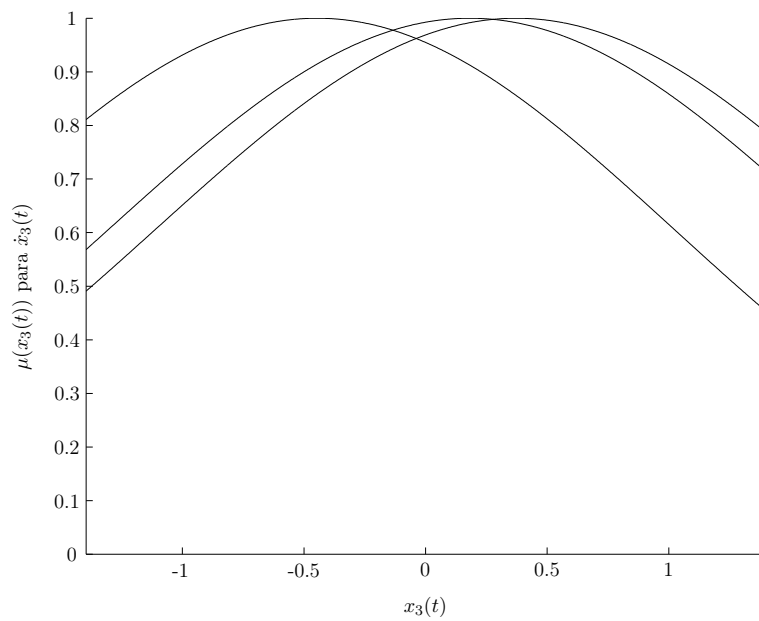


Figura 6.53: *Funciones de pertenencia para la entrada $x_3(t)$ en la inferencia de $\dot{x}_3(t)$.*

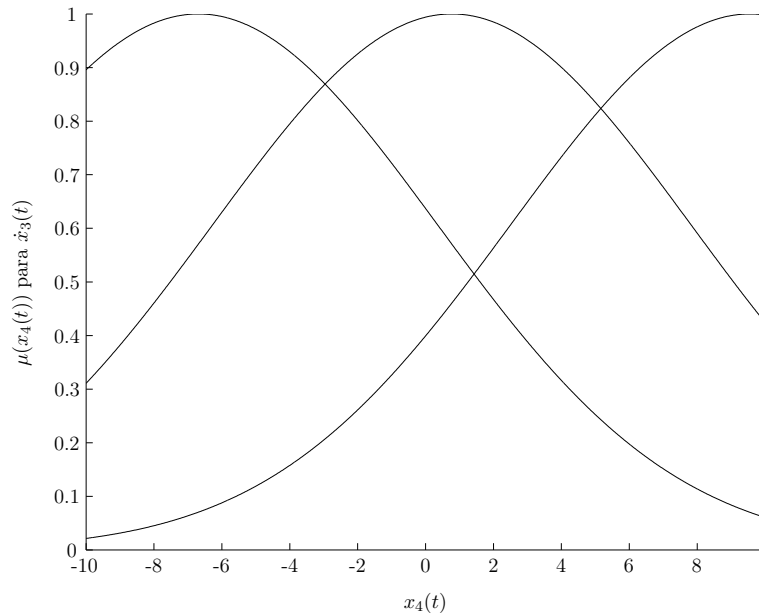


Figura 6.54: *Funciones de pertenencia para la entrada $x_4(t)$ en la inferencia de $\dot{x}_3(t)$.*

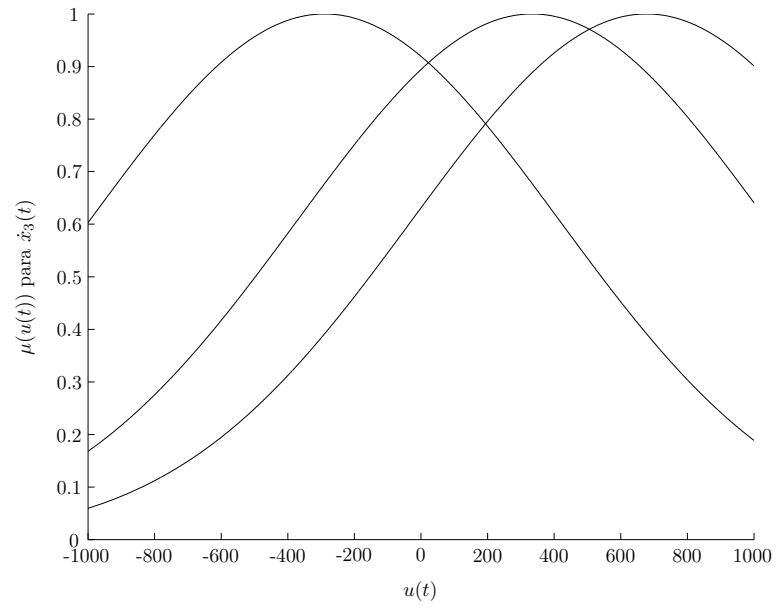


Figura 6.55: Funciones de pertenencia para la entrada $u(t)$ en la inferencia de $\dot{x}_3(t)$.

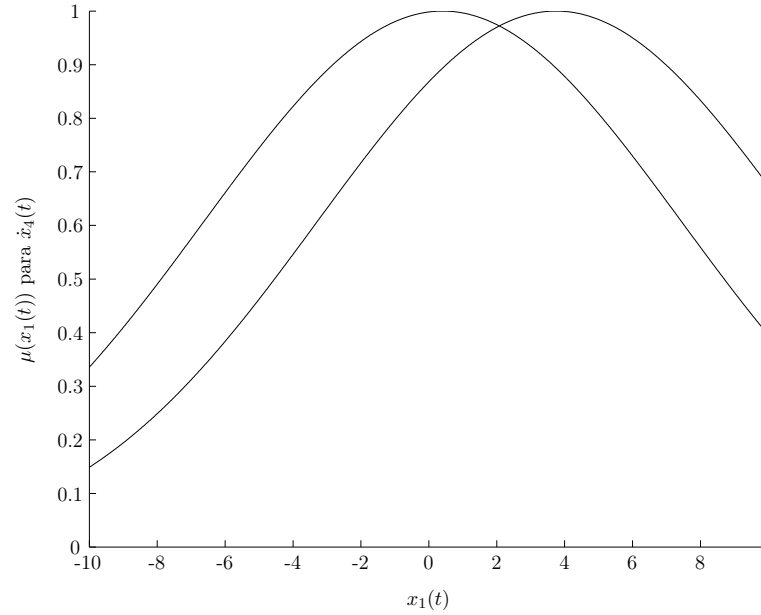


Figura 6.56: Funciones de pertenencia para la entrada $x_1(t)$ en la inferencia de $\dot{x}_4(t)$.

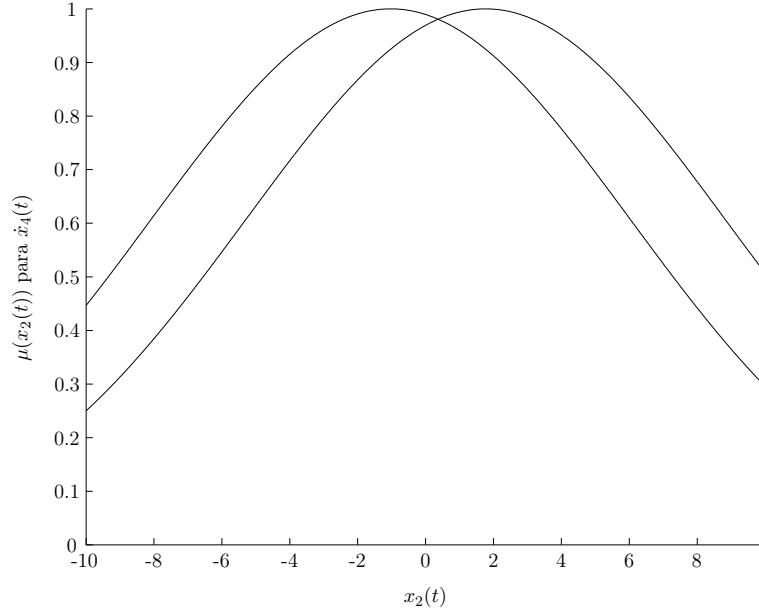


Figura 6.57: Funciones de pertenencia para la entrada $x_2(t)$ en la inferencia de $\dot{x}_4(t)$.

Al ajustar los parámetros del modelo borroso mediante el algoritmo neuroborroso propuesto en [271], se obtiene el siguiente conjunto de reglas, donde la función Gaussmf representa la función de pertenencia gaussiana dada por la ecuación (2.12), cuyos parámetros son el centro y el ancho, respectivamente:

SI x_1 es Gaussmf(0,4073;9,959) y x_2 es Gaussmf(-1,028;9,991) y x_3 es Gaussmf(0,1844;2,091) y x_4 es Gaussmf(0,7741;9,963) y τ es Gaussmf(-289,5;998,5)

ENTONCES $\dot{x}_1 = 5,55 \times 10^{-6} - 5,427 + 1x_2 - 1,812 \times 10^{-6}x_3 - 2,737 \times 10^{-7}x_4$

SI x_1 es Gaussmf(-0,3721;9,959) y x_2 es Gaussmf(6,016;9,991) y x_3 es Gaussmf(0,02722;2,091) y x_4 es Gaussmf(-5,582;9,963) y τ es Gaussmf(39,83;998,5)

ENTONCES $\dot{x}_1 = -1,165 \times 10^{-5} - 7,638 + 1x_2 + 3,315 \times 10^{-6}x_3 - 1,145 \times 10^{-6}x_4 + 1,638 \times 10^{-8}\tau$

SI x_1 es Gaussmf(-4,845;9,959) y x_2 es Gaussmf(-9,196;9,991) y x_3 es Gaussmf(-0,1623;2,091) y x_4 es Gaussmf(-0,3103;9,963) y τ es Gaussmf(878,6;998,5)

ENTONCES $\dot{x}_1 = 1,25 \times 10^{-6} + 1,934 \times 10^{-7}x_1 + 1x_2 + 7,341 \times 10^{-7}x_3 + 2,985 \times 10^{-7}x_4$

SI x_1 es Gaussmf(1,043;8,858) y x_2 es Gaussmf(0,255;9,007) y x_3 es Gaussmf(-0,6822;1,649) y x_4 es Gaussmf(1,709;8,882) y τ es Gaussmf(-84,04;848,8)

ENTONCES $\dot{x}_2 = 26,35 - 0,01899x_1 - 0,05199x_2 - 10,05x_3 - 0,01124x_4 - 0,0009459\tau$

SI x_1 es Gaussmf(1,022;8,867) y x_2 es Gaussmf(0,1771;9,013) y x_3 es Gaussmf(0,8741;3,138) y x_4 es Gaussmf(1,7;8,888) y τ es Gaussmf(-95,53;848,7)

ENTONCES $\dot{x}_2 = -23,77 + 0,001001x_1 + 0,001826x_2 + 6,809x_3 + 0,0001997x_4$

SI x_1 es Gaussmf(-2,073;8,709) y x_2 es Gaussmf(3,663;8,441) y x_3 es Gaussmf(0,01327;0,3791) y x_4 es Gaussmf(-5,881;8,901) y τ es Gaussmf(226,2;848,7)

ENTONCES $\dot{x}_2 = -0,42 - 0,01322x_1 + 0,01463x_2 - 9,655x_3 - 0,03124x_4 + 0,0001677\tau$

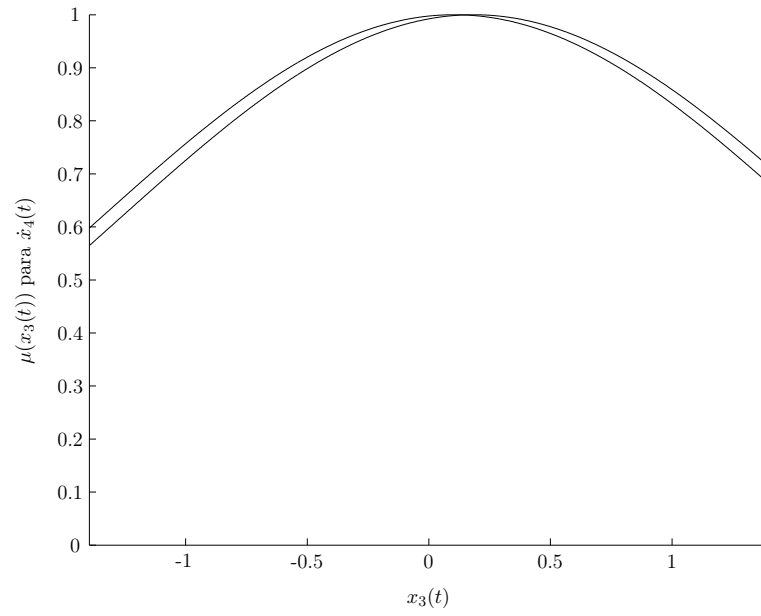


Figura 6.58: Funciones de pertenencia para la entrada $x_3(t)$ en la inferencia de $\dot{x}_4(t)$.

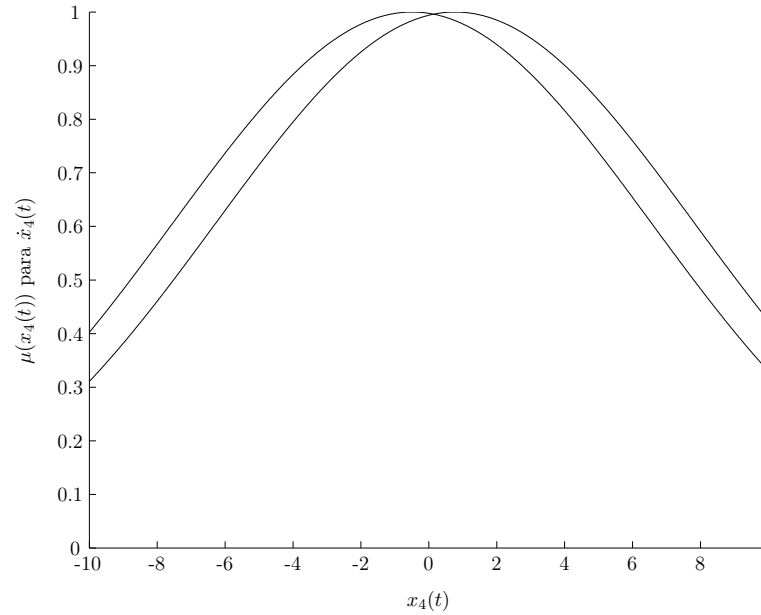


Figura 6.59: Funciones de pertenencia para la entrada $x_4(t)$ en la inferencia de $\dot{x}_4(t)$.

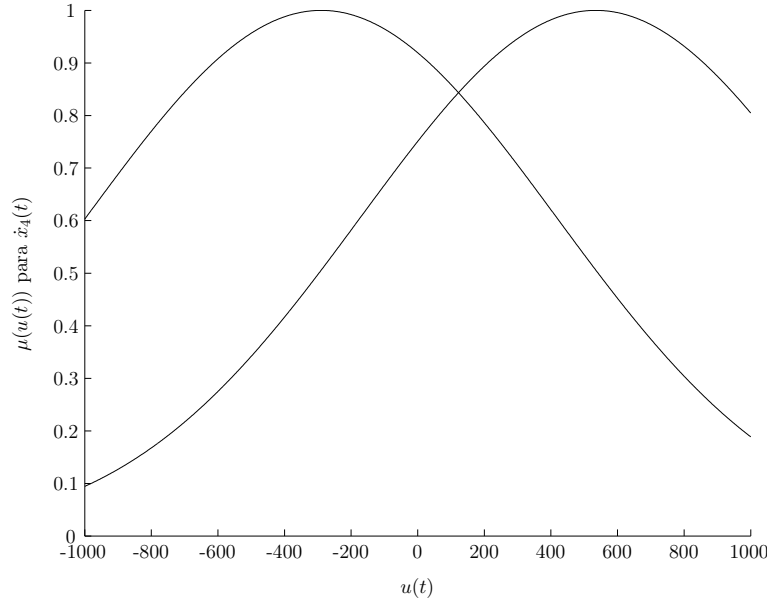


Figura 6.60: Funciones de pertenencia para la entrada $u(t)$ en la inferencia de $\dot{x}_4(t)$.

SI x_1 es Gaussmf(-7,172;8,54) y x_2 es Gaussmf(-4,562;8,528) y x_3 es Gaussmf(-2,232;0,3734) y x_4 es Gaussmf(-5,029;8,509) y τ es Gaussmf(-699,9;848,7)

ENTONCES $\dot{x}_2 = 14,87 + 0,0424x_1 + 0,01048x_2 + 2,694x_3 + 0,024x_4 + 7,933 \times 10^{-6}\tau$

SI x_1 es Gaussmf(-8,432;8,516) y x_2 es Gaussmf(4,808;8,488) y x_3 es Gaussmf(0,7307;0,2098) y x_4 es Gaussmf(3,968;8,457) y τ es Gaussmf(-814,6;848,7)

ENTONCES $\dot{x}_2 = -1,625 - 0,002903x_1 + 0,005995x_2 - 6,621x_3 - 0,006514x_4 + 8,155 \times 10^{-5}\tau$

SI x_1 es Gaussmf(9,087;8,49) y x_2 es Gaussmf(7,677;8,504) y x_3 es Gaussmf(-1,72;0,1867) y x_4 es Gaussmf(-6,585;8,456) y τ es Gaussmf(330,5;848,7)

ENTONCES $\dot{x}_2 = 9,978 + 0,03442x_1 + 0,006406x_2 + 0,4106x_3 - 0,01601x_4 + 1,73 \times 10^{-5}\tau$

SI x_1 es Gaussmf(3,933;8,345) y x_2 es Gaussmf(-9,609;8,622) y x_3 es Gaussmf(-2,479;0,7481) y x_4 es Gaussmf(8,201;8,593) y τ es Gaussmf(-228,6;848,7)

ENTONCES $\dot{x}_2 = 12,31 - 0,0139x_1 + 0,02918x_2 + 1,299x_3 - 0,03136x_4 - 0,0001634\tau$

SI x_1 es Gaussmf(0,4097;9,958) y x_2 es Gaussmf(-1,027;9,991) y x_3 es Gaussmf(0,1811;2,098) y x_4 es Gaussmf(0,7732;9,964) y τ es Gaussmf(-289,5;998,5)

ENTONCES $\dot{x}_3 = -1,415 \times 10^{-6} - 1,448 \times 10^{-7}x_1 + 1,347 \times 10^{-6}x_3 + 1x_4$

SI x_1 es Gaussmf(1,238;9,958) y x_2 es Gaussmf(3,299;9,99) y x_3 es Gaussmf(-0,4462;2,075) y x_4 es Gaussmf(-6,692;9,965) y τ es Gaussmf(334;998,5)

ENTONCES $\dot{x}_3 = 4,591 \times 10^{-6} - 4,168 \times 10^{-7}x_2 - 2,117 \times 10^{-6}x_3 + 1x_4 - 8,434 \times 10^{-9}\tau$

SI x_1 es Gaussmf(-4,104;9,961) y x_2 es Gaussmf(2,195;9,992) y x_3 es Gaussmf(0,3734;2,097) y x_4 es Gaussmf(9,556;9,973) y τ es Gaussmf(677,5;998,5)

ENTONCES $\dot{x}_3 = 1,174 \times 10^{-7}x_2 - 2,867 \times 10^{-7}x_3 + 1x_4 + 2,686 \times 10^{-9}\tau$

SI x_1 es Gaussmf(0,4073;9,959) y x_2 es Gaussmf(-1,028;9,991) y x_3 es Gaussmf(0,1844;2,091) y x_4

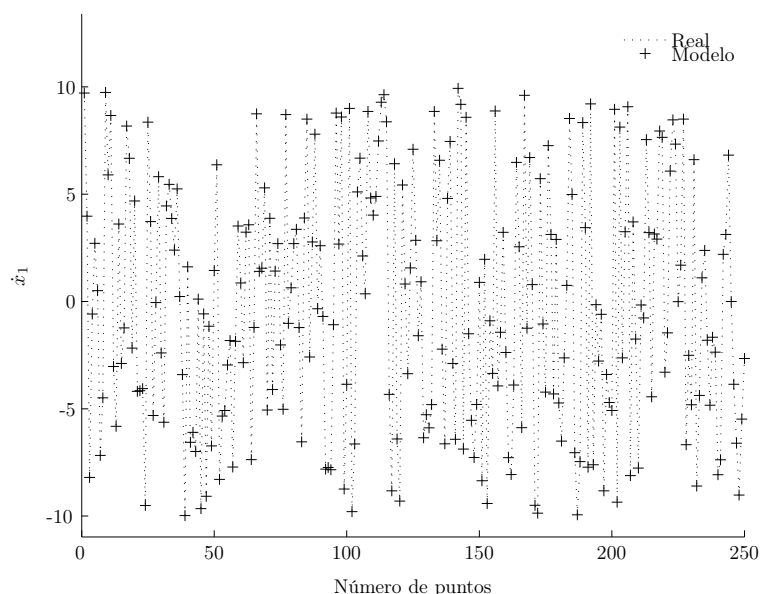


Figura 6.61: Validación del modelo borroso para $\dot{x}_1$.

es $\text{Gaussmf}(0,7741;9,963)$ y τ es $\text{Gaussmf}(-289,5;998,5)$

ENTONCES $\dot{x}_4 = -0,002216-1x_1-2,969 \times 10^{-5}x_2+0,0001712x_3-1,9 \times 10^{-5}x_4+1\tau$

SI x_1 es $\text{Gaussmf}(3,747;9,959)$ y x_2 es $\text{Gaussmf}(1,766;9,991)$ y x_3 es $\text{Gaussmf}(0,1036;2,091)$ y x_4 es $\text{Gaussmf}(-0,4926;9,963)$ y τ es $\text{Gaussmf}(534,6;998,5)$

ENTONCES $\dot{x}_4 = 0,00336-1x_1-0,0001285x_2-0,0002992x_3+0,0001179x_4+1\tau$

Al finalizar el modelado de la planta, el error cuadrático medio resultante para $\dot{x}_1(t)$, $\dot{x}_2(t)$, $\dot{x}_3(t)$ y $\dot{x}_4(t)$ fue de $5,03354 \times 10^{-12}$, $0,000102$, $6,739 \times 10^{-12}$ y $3,225 \times 10^{-6}$, respectivamente. Al validar el modelo se ha obtenido un error cuadrático medio de $6,985 \times 10^{-12}$ para $\dot{x}_1(t)$, $0,001171$ para $\dot{x}_2(t)$, $2,123 \times 10^{-12}$ para $\dot{x}_3(t)$ y $1,041 \times 10^{-7}$ para $\dot{x}_4(t)$. Observando estos resultados y la representación gráfica de los errores de validación del modelo de las figuras 6.61 y 6.68, puede concluirse que el modelo borroso es válido dentro del universo de discurso en el que ha sido realizado.

Estudio de la planta a partir de su modelo borroso

Inicialmente la planta (6.25) era completamente desconocida, sin embargo, tal como se vio en la sección 4.4, una vez que se posee un modelo borroso validado es posible extraer información muy útil de la planta, como sus estados de equilibrio y la estabilidad de éstos. Para ello se ha aplicado cinco veces el algoritmo de la figura 4.1 al modelo borroso de la planta, partiendo en cada una de las ejecuciones de un punto de inicio $\mathbf{x}_0$ aleatorio perteneciente al universo de discurso de las variables de estado. Los resultados obtenidos, tanto para el método de Gauss–Newton como para el método de Levenberg–Marquardt,

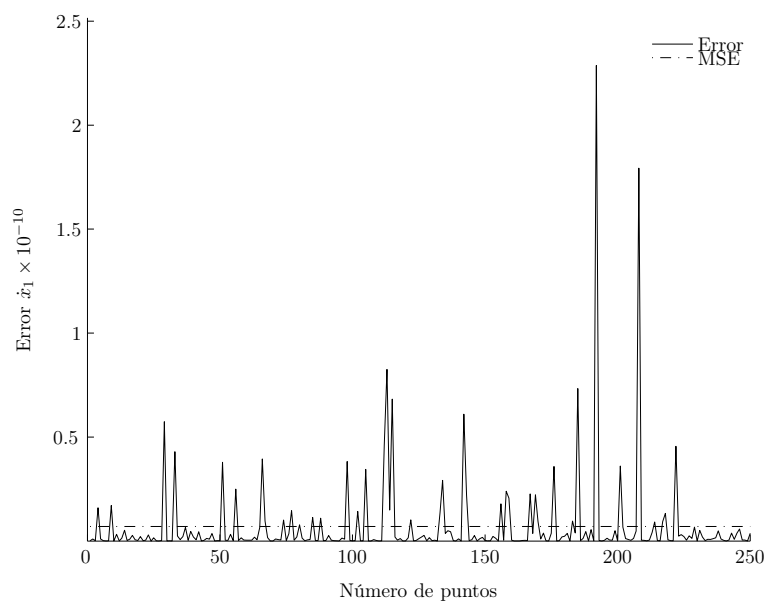


Figura 6.62: Error de modelado durante la validación de $\dot{x}_1$.

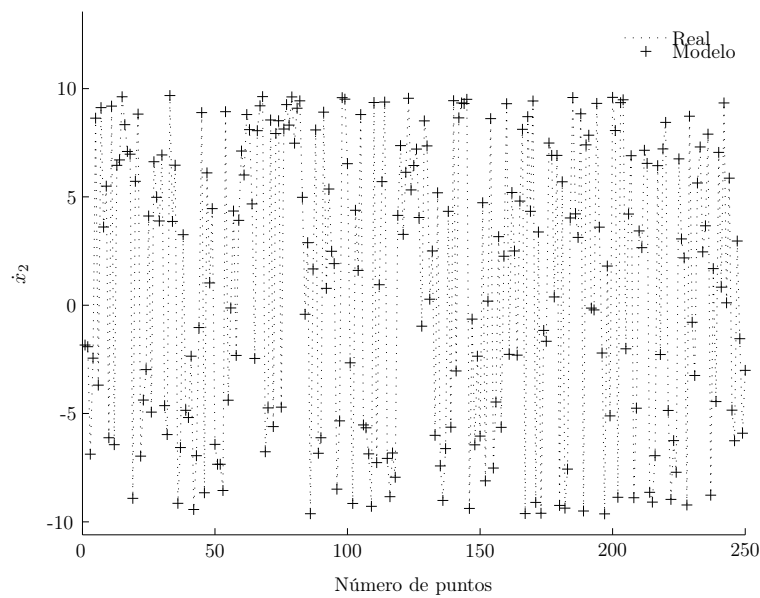


Figura 6.63: Validación del modelo borroso para $\dot{x}_2$.

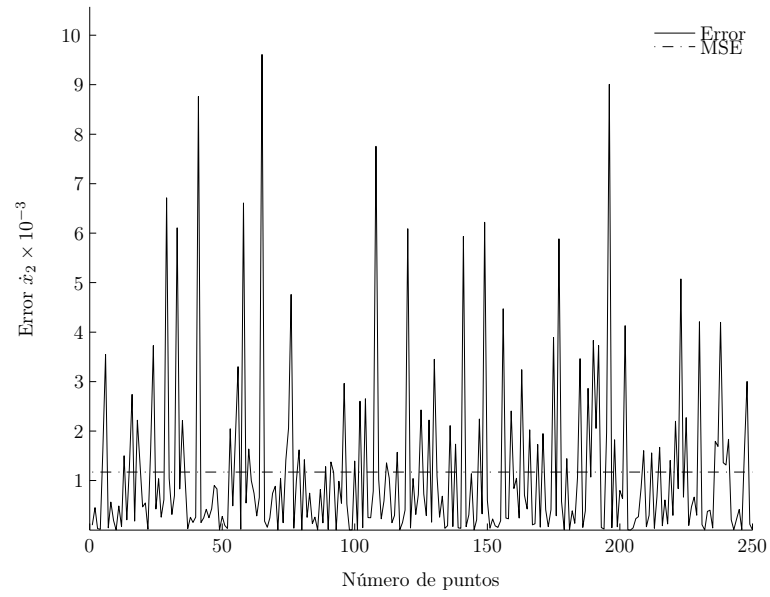


Figura 6.64: Error de modelado durante la validación de $\dot{x}_2$.

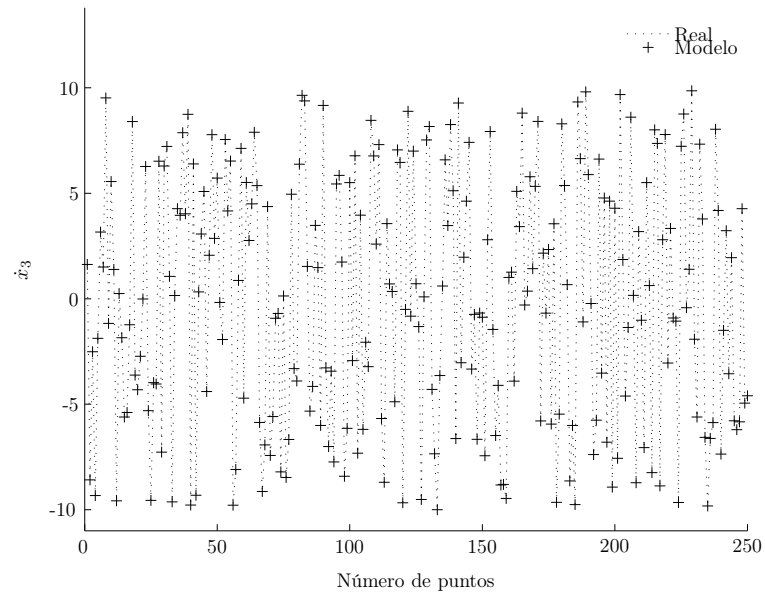


Figura 6.65: Validación del modelo borroso para x_3 .

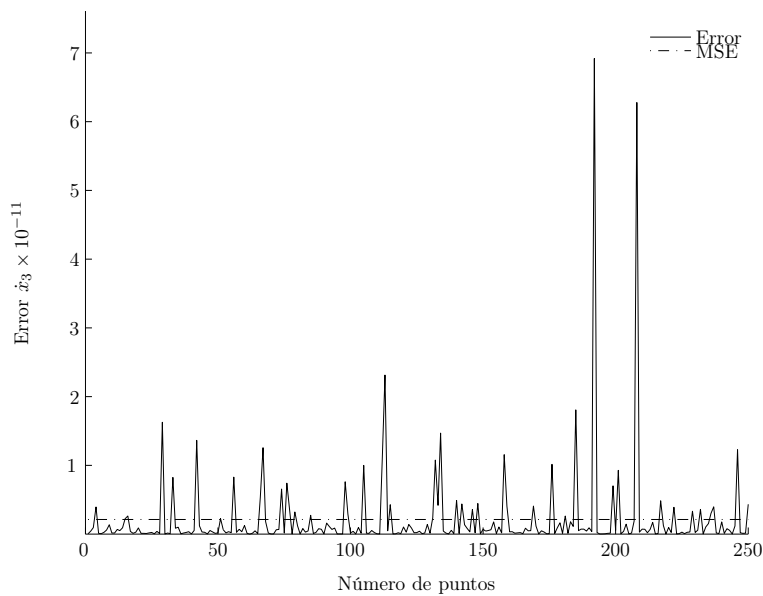


Figura 6.66: Error de modelado durante la validación de $\dot{x}_3$.

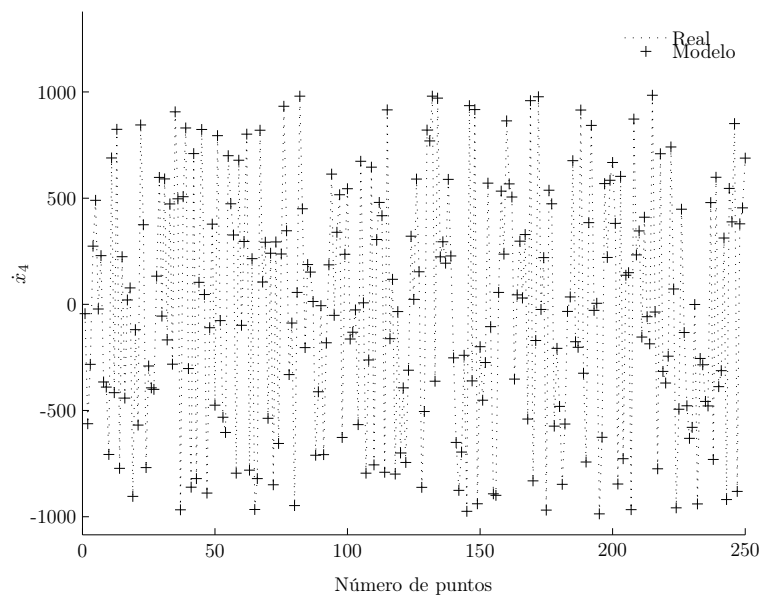


Figura 6.67: Validación del modelo borroso para $\dot{x}_4$.

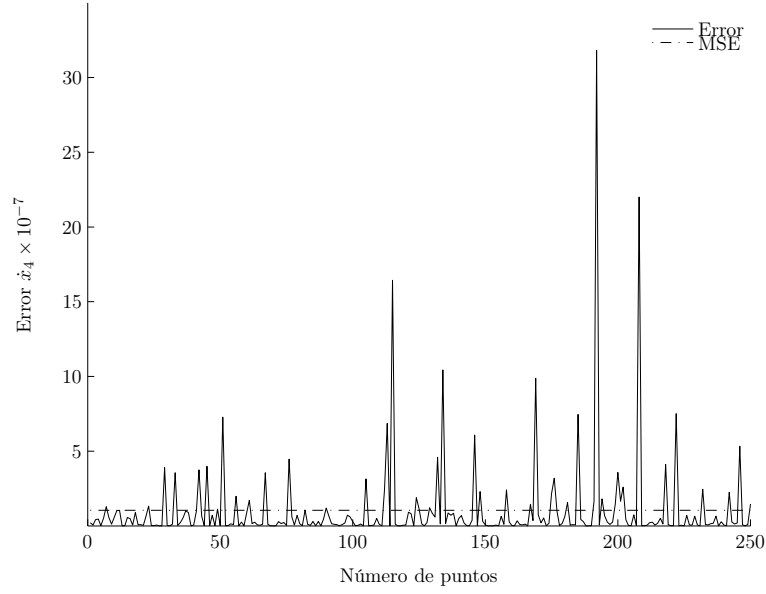


Figura 6.68: Error de modelado durante la validación de $\dot{x}_4$.

aplicando el cálculo explícito de la matriz jacobiana y sin aplicarlo, se muestran en las tablas 6.29 a 6.33. En todos los casos se ha supuesto $\mathbf{u}(t) = \mathbf{0}$.

Al analizando los resultados de las tablas 6.29 a 6.33 se observa que se ha obtenido un único estado de equilibrio correspondiente prácticamente con el origen de coordenadas. Efectivamente el origen es el único estado de equilibrio que posee la planta dentro del universo de discurso, y ha sido obtenido correctamente por el algoritmo de la figura 4.1 partiendo del modelo borroso de la planta, y sin requerir conocimiento previo de ésta salvo un conjunto de datos de entrada/salida.

Atendiendo a las respuestas de los distintos algoritmos empleados, para el caso del método de Gauss-Newton queda patente una mejoría prácticamente generalizada con la incorporación del cálculo explícito de la matriz jacobiana; sin embargo, para el método de Levenberg-Marquardt la mejoría no es tan generalizada ni tan destacable.

Siguiendo con el estudio de la planta a partir de su modelo borroso, es posible calcular el modelo lineal de ésta en torno al estado de equilibrio obtenido anteriormente. Al aplicar el algoritmo 4.2 en el estado de equilibrio $\mathbf{x}^T = (8,334 \times -5, 4,737 \times -7, 0,0012, -5,782 \times -7)^T$, con $\tau = 0$, se obtiene el sistema lineal:

$$\dot{\mathbf{x}}(t) \simeq \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0,004 & -0,0059 & -9,7417 & 0,0101 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & -0,0001 & 0 \end{pmatrix} \mathbf{x}(t) + \begin{pmatrix} 0 \\ -0,0001 \\ 0 \\ 1 \end{pmatrix} \tau(t), \quad (6.28)$$

	Gauss–Newton	G–N con jacobiana	Levenberg–Marquardt	L–M con jacobiana
Estado de equilibrio	$\begin{pmatrix} 8,334e-05 \\ 4,738e-07 \\ 0,0012 \\ -5,782e-07 \end{pmatrix}$	$\begin{pmatrix} 8,334e-05 \\ 4,738e-07 \\ 0,0012 \\ -5,782e-07 \end{pmatrix}$	$\begin{pmatrix} 8,334e-05 \\ 4,738e-07 \\ 0,0012 \\ -5,782e-07 \end{pmatrix}$	$\begin{pmatrix} 8,334e-05 \\ 4,738e-07 \\ 0,0012 \\ -5,782e-07 \end{pmatrix}$
Nº Iteraciones	4	4	4	4
Evaluaciones de la función objetivo	25	10	25	10
Distancia del punto obtenido al 0	2,25336e-15	2,25677e-15	2,25336e-15	2,25677e-15
Tiempo empleado (s)	0,59785	0,124506	0,044559	0,039814

Tabla 6.29: Resultados de la búsqueda de estados de equilibrio. Ejecución 1.

	Gauss–Newton	G–N con jacobiana	Levenberg–Marquardt	L–M con jacobiana
Estado de equilibrio	$\begin{pmatrix} 8,334e-05 \\ 4,738e-07 \\ 0,0012 \\ -5,782e-07 \end{pmatrix}$	$\begin{pmatrix} 8,334e-05 \\ 4,738e-07 \\ 0,0012 \\ -5,782e-07 \end{pmatrix}$	$\begin{pmatrix} 8,334e-05 \\ 4,738e-07 \\ 0,0012 \\ -5,782e-07 \end{pmatrix}$	$\begin{pmatrix} 8,334e-05 \\ 4,738e-07 \\ 0,0012 \\ -5,782e-07 \end{pmatrix}$
Nº Iteraciones	5	5	5	5
Evaluaciones de la función objetivo	30	12	30	12
Distancia del punto obtenido al 0	9,04042e-30	5,76613e-30	9,04042e-30	5,76613e-30
Tiempo empleado (s)	0,57885	0,10104	0,051286	0,041344

Tabla 6.30: Resultados de la búsqueda de estados de equilibrio. Ejecución 2.

	Gauss-Newton	G-N con jacobiana	Levenberg-Marquardt	L-M con jacobiana
Estado de equilibrio	$\begin{pmatrix} 8,334e-05 \\ 4,738e-07 \\ 0,0012 \\ -5,782e-07 \end{pmatrix}$	$\begin{pmatrix} 8,334e-05 \\ 4,738e-07 \\ 0,0012 \\ -5,782e-07 \end{pmatrix}$	$\begin{pmatrix} 8,334e-05 \\ 4,738e-07 \\ 0,0012 \\ -5,782e-07 \end{pmatrix}$	$\begin{pmatrix} 8,334e-05 \\ 4,738e-07 \\ 0,0012 \\ -5,782e-07 \end{pmatrix}$
Nº Iteraciones	5	5	5	5
Evaluaciones de la función objetivo	30	12	30	12
Distancia del punto obtenido al 0	3,31461e-19	3,27946e-19	3,31461e-19	3,27946e-19
Tiempo empleado (s)	0,579256	0,110237	0,052842	0,062197

Tabla 6.31: Resultados de la búsqueda de estados de equilibrio. Ejecución 3.

	Gauss-Newton	G-N con jacobiana	Levenberg-Marquardt	L-M con jacobiana
Estado de equilibrio	$\begin{pmatrix} 8,334e-05 \\ 4,738e-07 \\ 0,0012 \\ -5,782e-07 \end{pmatrix}$	$\begin{pmatrix} 8,334e-05 \\ 4,738e-07 \\ 0,0012 \\ -5,782e-07 \end{pmatrix}$	$\begin{pmatrix} 8,334e-05 \\ 4,738e-07 \\ 0,0012 \\ -5,782e-07 \end{pmatrix}$	$\begin{pmatrix} 8,334e-05 \\ 4,738e-07 \\ 0,0012 \\ -5,782e-07 \end{pmatrix}$
Nº Iteraciones	5	5	5	5
Evaluaciones de la función objetivo	30	12	30	12
Distancia del punto obtenido al 0	1,63751e-28	1,36959e-31	1,63751e-28	1,36959e-31
Tiempo empleado (s)	0,579351	0,111862	0,05261	0,04175

Tabla 6.32: Resultados de la búsqueda de estados de equilibrio. Ejecución 4.

	Gauss-Newton	G-N con jacobiana	Levenberg-Marquardt	L-M con jacobiana
Estado de equilibrio	$\begin{pmatrix} 8,334e-05 \\ 4,738e-07 \\ 0,0012 \\ -5,782e-07 \end{pmatrix}$	$\begin{pmatrix} 8,334e-05 \\ 4,738e-07 \\ 0,0012 \\ -5,782e-07 \end{pmatrix}$	$\begin{pmatrix} 8,334e-05 \\ 4,738e-07 \\ 0,0012 \\ -5,782e-07 \end{pmatrix}$	$\begin{pmatrix} 8,334e-05 \\ 4,738e-07 \\ 0,0012 \\ -5,782e-07 \end{pmatrix}$
Nº Iteraciones	5	5	5	5
Evaluaciones de la función objetivo	30	12	30	12
Distancia del punto obtenido al 0	3,30549e-16	3,30791e-16	3,30549e-16	3,30791e-16
Tiempo empleado (s)	0,598084	0,099627	0,053277	0,059275

Tabla 6.33: Resultados de la búsqueda de estados de equilibrio. Ejecución 5.

cuyos autovalores son $\lambda_1 = -1,7695$ y $\lambda_2 = 1,7650$ y $\lambda_{3,4} = -0,0007 \pm 1,7661i$. Por lo tanto, se puede afirmar que, al haber un autovalor con parte real positiva, el estado de equilibrio del sistema es inestable en lazo abierto.

Si con objeto de contrastar los resultados anteriores, se calcula el sistema linealizado a partir de la ecuación de estado (6.27) de la planta real, se obtiene que:

$$\dot{\mathbf{x}}(t) \simeq \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -9,8 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \end{pmatrix} \mathbf{x}(t) + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tau(t), \quad (6.29)$$

cuyos autovalores son $\lambda_1 = -1,7693$ y $\lambda_2 = 1,7639$ y $\lambda_{3,4} = 0 \pm 1,7693i$. Como se puede observar, tanto el sistema linealizado como los autovalores obtenidos tan sólo a partir del estudio del modelo borroso de la planta, son bastante aproximados a los reales.

Diseño de un controlador borroso estable

El siguiente paso es diseñar un controlador borroso utilizando el algoritmo presentado en la sección 5.4.4, de forma que se garantice la estabilidad del sistema de control de lazo cerrado en torno al estado de equilibrio situado en el origen.

Durante la fase de inicialización se crean tantas regiones como reglas totales tenga la planta, en este caso $s = 3 + 7 + 3 + 2 = 15$, y se inicializan las matrices $\mathbf{P}_q$, con $q = 1, 15$, como la matriz identidad. Las regiones $\varphi_q(\mathbf{x})$ se definen a partir de las funciones de pertenencia del controlador, según se establece en el algoritmo 5.6. Una vez concluidas las fases de diseño y de ajuste del estado de equilibrio, se obtiene el siguiente controlador borroso:

SI x_1 es Gaussmf(0,4073;9,959) y x_2 es Gaussmf(-1,028;9,991) y x_3 es Gaussmf(0,1844;2,091) y x_4 es Gaussmf(0,7741;9,963)

ENTONCES $\tau = -234,8+1,82x_1+4,862x_2-848,3x_3-128,4x_4$

SI x_1 es Gaussmf(-0,3721;9,959) y x_2 es Gaussmf(6,016;9,991) y x_3 es Gaussmf(0,02722;2,091) y x_4 es Gaussmf(-5,582;9,963)

ENTONCES $\tau = -685,6+1,056x_1+55,02x_2-461,9x_3-78,43x_4$

SI x_1 es Gaussmf(-4,845;9,959) y x_2 es Gaussmf(-9,196;9,991) y x_3 es Gaussmf(-0,1623;2,091) y x_4 es Gaussmf(-0,3103;9,963)

ENTONCES $\tau = -402,5+22,52x_1+77,02x_2-999,4x_3-563x_4$

SI x_1 es Gaussmf(1,043;8,858) y x_2 es Gaussmf(0,255;9,007) y x_3 es Gaussmf(-0,6822;1,649) y x_4 es Gaussmf(1,709;8,882)

ENTONCES $\tau = -71,62+4,288x_1+4,271x_2-403,8x_3-84,73x_4$

SI x_1 es Gaussmf(1,022;8,867) y x_2 es Gaussmf(0,1771;9,013) y x_3 es Gaussmf(0,8741;3,138) y x_4 es Gaussmf(1,7;8,888)

ENTONCES $\tau = 607,5+1,606x_1+0,5682x_2-290,8x_3-124,3x_4$

SI x_1 es Gaussmf(-2,073;8,709) y x_2 es Gaussmf(3,663;8,441) y x_3 es Gaussmf(0,01327;0,3791) y x_4 es Gaussmf(-5,881;8,901)

ENTONCES $\tau = -5,442+0,918x_1+1,093x_2-969x_3-32,39x_4$

SI x_1 es Gaussmf(-7,172;8,54) y x_2 es Gaussmf(-4,562;8,528) y x_3 es Gaussmf(-2,232;0,3734) y x_4 es Gaussmf(-5,029;8,509)

ENTONCES $\tau = -455,2+0,8671x_1-26,72x_2-90,27x_3-127,8x_4$

SI x_1 es Gaussmf(-8,432;8,516) y x_2 es Gaussmf(4,808;8,488) y x_3 es Gaussmf(0,7307;0,2098) y x_4 es Gaussmf(3,968;8,457)

ENTONCES $\tau = 984,2+2,523x_1+5,306x_2-674,2x_3-245,4x_4$

SI x_1 es Gaussmf(9,087;8,49) y x_2 es Gaussmf(7,677;8,504) y x_3 es Gaussmf(-1,72;0,1867) y x_4 es Gaussmf(-6,585;8,456)

ENTONCES $\tau = -379,2+0,3281x_1-3,544x_2-11,78x_3-2,804x_4$

SI x_1 es Gaussmf(3,933;8,345) y x_2 es Gaussmf(-9,609;8,622) y x_3 es Gaussmf(-2,479;0,7481) y x_4 es Gaussmf(8,201;8,593)

ENTONCES $\tau = 686,3-1,819x_1+0,5094x_2-53,64x_3-3,184x_4$

SI x_1 es Gaussmf(0,4097;9,958) y x_2 es Gaussmf(-1,027;9,991) y x_3 es Gaussmf(0,1811;2,098) y x_4 es Gaussmf(0,7732;9,964)

ENTONCES $\tau = -642+1,13x_1+42,87x_2-160,2x_3-79,8x_4$

SI x_1 es Gaussmf(1,238;9,958) y x_2 es Gaussmf(3,299;9,99) y x_3 es Gaussmf(-0,4462;2,075) y x_4 es Gaussmf(-6,692;9,965)

ENTONCES $\tau = 295,1+1,224x_1+0,2051x_2-236,5x_3-19,86x_4$

SI x_1 es Gaussmf(-4,104;9,961) y x_2 es Gaussmf(2,195;9,992) y x_3 es Gaussmf(0,3734;2,097) y x_4 es Gaussmf(9,556;9,973)

ENTONCES $\tau = 251,9+1,481x_1+6,343x_2-15,19x_3-53,51x_4$

SI x_1 es Gaussmf(0,4073;9,959) y x_2 es Gaussmf(-1,028;9,991) y x_3 es Gaussmf(0,1844;2,091) y x_4

es Gaussmf(0,7741;9,963)

ENTONCES $\tau = -146,2+1,066x_1+18,05x_2-127,2x_3-77,6x_4$

SI x_1 es Gaussmf(3,747;9,959) y x_2 es Gaussmf(1,766;9,991) y x_3 es Gaussmf(0,1036;2,091) y x_4 es Gaussmf(-0,4926;9,963)

ENTONCES $\tau = 924,2+1,156x_1+4,171x_2-998,5x_3-318,4x_4$

Las matrices $\mathbf{P}_q, q = 1..15$, obtenidas durante la fase de diseño son:

$$\mathbf{P}_1 = \begin{pmatrix} 1,03605 & 0,26700 & -33,59060 & -0,14408 \\ 0,26700 & 139,22268 & -83,97937 & -1,65894 \\ -33,59060 & -83,97937 & 33600,18560 & 154,76559 \\ -0,14408 & -1,65894 & 154,76559 & 13,02369 \end{pmatrix}$$

$$\mathbf{P}_2 = \begin{pmatrix} 1,05110 & 0,72560 & -0,96851 & -0,02303 \\ 0,72560 & 18,96391 & -50,66650 & -0,16380 \\ -0,96851 & -50,66650 & 543,51591 & 5,70102 \\ -0,02303 & -0,16380 & 5,70102 & 1,61686 \end{pmatrix}$$

$$\mathbf{P}_3 = \begin{pmatrix} 1,58212 & 0,47886 & -6,41573 & -0,14843 \\ 0,47886 & 6,70578 & -19,35691 & -0,36642 \\ -6,41573 & -19,35691 & 295,25209 & 6,88047 \\ -0,14843 & -0,36642 & 6,88047 & 4,06310 \end{pmatrix}$$

$$\mathbf{P}_4 = \begin{pmatrix} 1,08457 & 0,21714 & -9,73892 & -0,00011 \\ 0,21714 & 16,04206 & -6,05531 & -0,33527 \\ -9,73892 & -6,05531 & 1227,02882 & 0,57986 \\ -0,00011 & -0,33527 & 0,57986 & 3,18787 \end{pmatrix}$$

$$\mathbf{P}_5 = \begin{pmatrix} 1,05994 & 0,51358 & -2,65288 & -0,02379 \\ 0,51358 & 71,11985 & -211,75601 & -1,75572 \\ -2,65288 & -211,75601 & 2072,49323 & 18,06140 \\ -0,02379 & -1,75572 & 18,06140 & 2,59142 \end{pmatrix}$$

$$\mathbf{P}_6 = \begin{pmatrix} 1,24421 & 0,09989 & 14,74030 & 0,11935 \\ 0,09989 & 470,03554 & -116,22256 & -7,61379 \\ 14,74030 & -116,22256 & 188101,52383 & 1217,19083 \\ 0,11935 & -7,61379 & 1217,19083 & 291,52925 \end{pmatrix}$$

$$\mathbf{P}_7 = \begin{pmatrix} 1,00981 & 0,04209 & 1,22501 & 0,02482 \\ 0,04209 & 25,35674 & 0,13656 & 0,88133 \\ 1,22501 & 0,13656 & 594,88711 & 10,44824 \\ 0,02482 & 0,88133 & 10,44824 & 4,18270 \end{pmatrix}$$

$$\mathbf{P}_8 = \begin{pmatrix} 1,05658 & 0,26597 & -17,40225 & -0,01933 \\ 0,26597 & 47,38054 & -39,99620 & -0,53799 \\ -17,40225 & -39,99620 & 6041,15180 & 8,55216 \\ -0,01933 & -0,53799 & 8,55216 & 12,77277 \end{pmatrix}$$

$$\mathbf{P}_9 = \begin{pmatrix} 1,21353 & 0,80756 & 2,86685 & 0,25734 \\ 0,80756 & 133,76820 & 12,54050 & 4,25095 \\ 2,86685 & 12,54050 & 49,03863 & 4,50226 \\ 0,25734 & 4,25095 & 4,50226 & 3,65691 \end{pmatrix}$$

$$\mathbf{P}_{10} = \begin{pmatrix} 1,29029 & 0,00385 & 4,55386 & 0,24077 \\ 0,00385 & 9,99605 & -0,10397 & 0,17388 \\ 4,55386 & -0,10397 & 85,86731 & 4,36586 \\ 0,24077 & 0,17388 & 4,36586 & 1,55064 \end{pmatrix}$$

$$\mathbf{P}_{11} = \begin{pmatrix} 1,01317 & 0,19151 & -2,74596 & -0,02890 \\ 0,19151 & 113,29462 & -50,73942 & -0,19035 \\ -2,74596 & -50,73942 & 2549,77750 & 21,21616 \\ -0,02890 & -0,19035 & 21,21616 & 1,76804 \end{pmatrix}$$

$$\mathbf{P}_{12} = \begin{pmatrix} 1,20771 & 0,07370 & -0,36812 & 0,00034 \\ 0,07370 & 90,34005 & -66,78123 & -3,46191 \\ -0,36812 & -66,78123 & 377,34342 & 3,95020 \\ 0,00034 & -3,46191 & 3,95020 & 1,76752 \end{pmatrix}$$

$$\mathbf{P}_{13} = \begin{pmatrix} 1,45936 & 0,03273 & -12,56280 & -0,13766 \\ 0,03273 & 3,36161 & -0,79467 & -0,31895 \\ -12,56280 & -0,79467 & 372,91436 & 4,13503 \\ -0,13766 & -0,31895 & 4,13503 & 2,79482 \end{pmatrix}$$

$$\mathbf{P}_{14} = \begin{pmatrix} 1,01904 & 4,23703 & -31,92389 & -1,96656 \\ 4,23703 & 982,68227 & -6701,97343 & -410,26987 \\ -31,92389 & -6701,97343 & 82803,86893 & 3359,93929 \\ -1,96656 & -410,26987 & 3359,93929 & 3459,89161 \end{pmatrix}$$

$$\mathbf{P}_{15} = \begin{pmatrix} 1,08186 & 3,35248 & -9,64731 & -0,06333 \\ 3,35248 & 975,82452 & -785,56530 & -7,07640 \\ -9,64731 & -785,56530 & 243397,41648 & 802,38340 \\ -0,06333 & -7,07640 & 802,38340 & 118,52195 \end{pmatrix}$$

En las figuras 6.69, 6.70, 6.71 y 6.72 se puede observar la evolución del sistema diseñado partiendo de distintas condiciones iniciales. En dichas figuras puede comprobarse que efectivamente, todas las trayectorias que no abandonan el universo de discurso de las variables de estado tienden al estado de equilibrio situado en el origen. La estabilidad de aquellas trayectorias que abandonan el universo de discurso de las variables de estado, mostradas en trazo punteado, no puede garantizarse, ya que el modelo borroso de la planta no tiene porqué ser válido fuera del rango de valores para el que fue obtenido. Por lo tanto, al no ser posible garantizar el cumplimiento del teorema 5.7 para dichas trayectorias, no puede asegurarse formalmente su estabilidad.

Ha de tenerse en cuenta que las simulaciones presentadas en las figuras 6.69 a 6.72 han sido realizadas utilizando aplicando el controlador borroso al sistema real dado por (6.27), con lo que se demuestra que el controlador borroso diseñado estabiliza la planta real, y no sólo el modelo borroso de la planta que fue utilizado como base para su diseño.

Una vez se aplica el controlador borroso diseñado a la planta, el sistema resultante presenta un estado de equilibrio asintóticamente estable en el origen, tal como puede verse en las figuras 6.69 a 6.72. Esto puede demostrarse⁵ a través de los autovalores de la matriz jacobiana del sistema de lazo cerrado.

Aplicando el algoritmo de la figura 4.5 al sistema de control borroso en lazo cerrado en el origen, se obtiene la siguiente matriz jacobiana:

$$\mathbf{J}(\mathbf{0}) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0,0032 & -0,0074 & -9,7115 & 0,0183 \\ 0 & 0 & 0 & 1 \\ 11,4953 & 22,8319 & -477,8912 & -126,1496 \end{pmatrix} \quad (6.30)$$

cuyos autovalores son $\lambda_1 = -122,26$, $\lambda_2 = -3,44$ y $\lambda_{3,4} = -0,23 \pm 0,46i$. Efectivamente, como puede observarse, la parte real de dichos autovalores es negativa, lo que implica que el estado de equilibrio es asintóticamente estable.

⁵Nótese que esta demostración tan sólo es válida para las cercanías del estado de equilibrio. La estabilidad queda garantizada por la aplicación del teorema 5.7 durante la fase de diseño del controlador borroso.

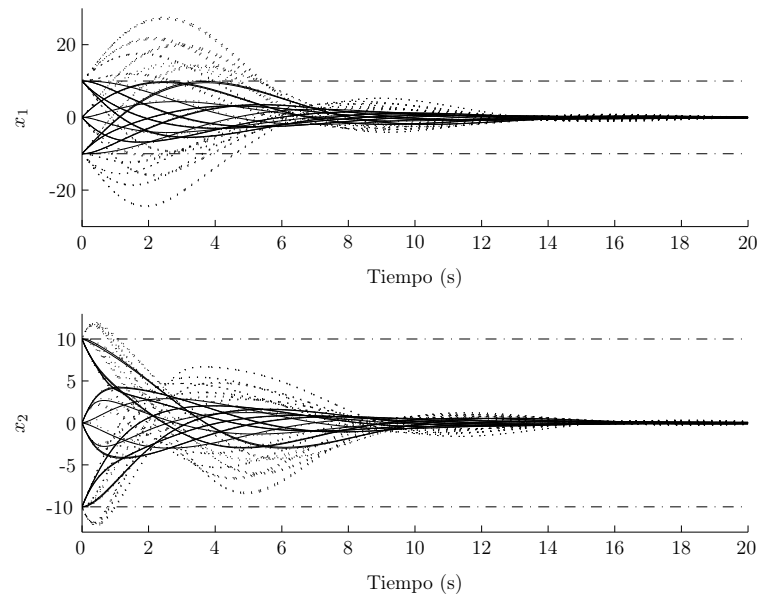


Figura 6.69: Simulación para distintas condiciones iniciales $\dot{x}_1$ y $\dot{x}_2$.

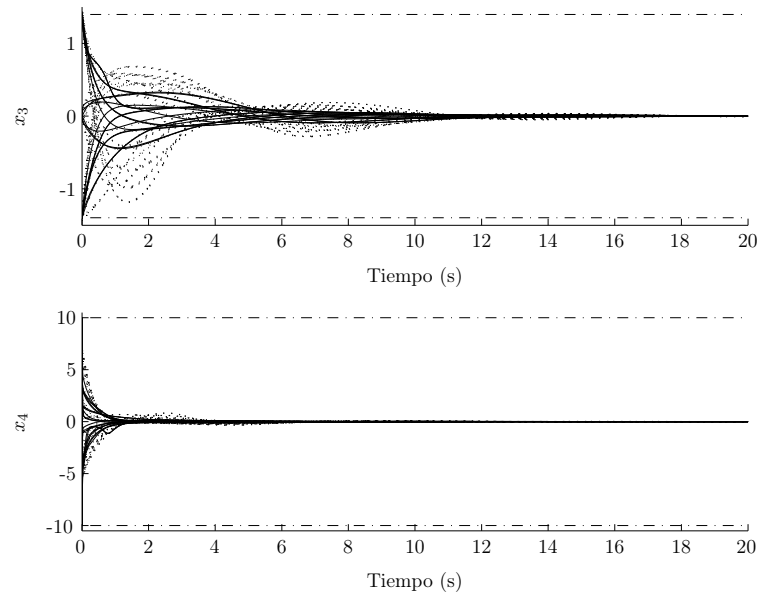


Figura 6.70: Simulación para distintas condiciones iniciales $\dot{x}_3$ y $\dot{x}_4$.

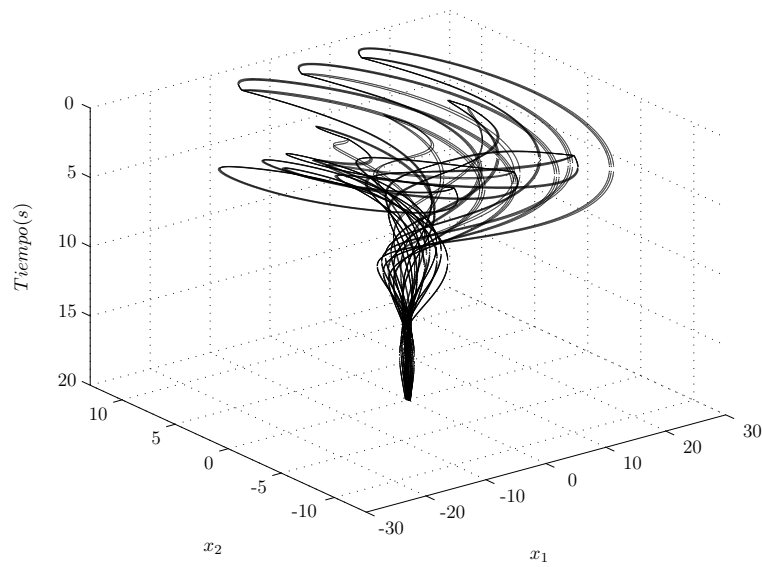


Figura 6.71: Retrato de fase temporal $\dot{x}_1$ y $\dot{x}_2$.

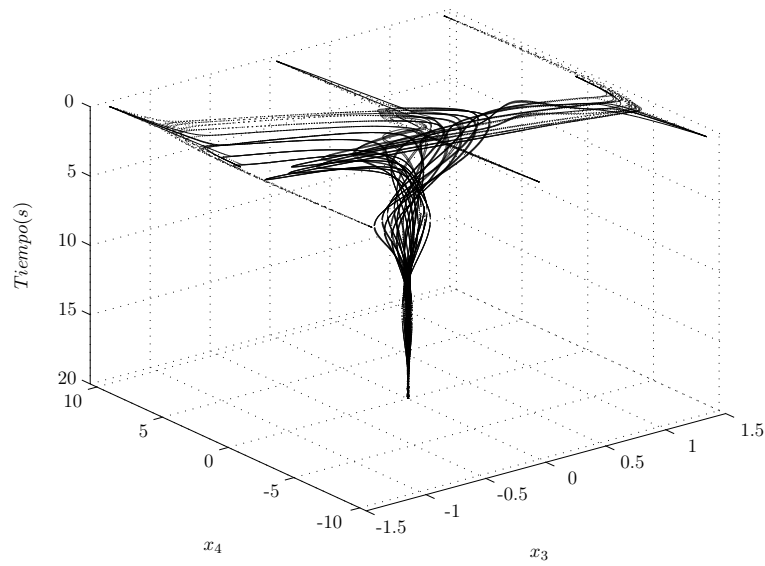


Figura 6.72: Retrato de fase temporal $\dot{x}_3$ y $\dot{x}_4$.

	Gauss-Newton	G-N con jacobiana	Levenberg-Marquardt	L-M con jacobiana
Estado de equilibrio	$\begin{pmatrix} -0,0064 \\ 4,7396e-07 \\ 0,0012 \\ -5,785e-07 \end{pmatrix}$	$\begin{pmatrix} -0,0064 \\ 4,7396e-07 \\ 0,0012 \\ -5,785e-07 \end{pmatrix}$	$\begin{pmatrix} -0,0064 \\ 4,7396e-07 \\ 0,0012 \\ -5,785e-07 \end{pmatrix}$	$\begin{pmatrix} -0,0064 \\ 4,7396e-07 \\ 0,0012 \\ -5,785e-07 \end{pmatrix}$
Nº Iteraciones	25	25	25	25
Evaluaciones de la función objetivo	110	52	110	52
Distancia del punto obtenido al 0	2,76701e-28	9,64293e-28	2,76701e-28	9,64293e-28
Tiempo empleado (s)	0,737866	0,243621	0,199435	0,186437

Tabla 6.34: Resultados de la búsqueda de estados de equilibrio para el sistema en lazo cerrado. Ejecución 1.

Finalmente, para completar el estudio sobre el sistema es posible estudiar los estados de equilibrio del sistema en lazo cerrado. Para ello se ha aplicado cinco veces el algoritmo numérico de minimización de la figura 4.1, partiendo en cada una de las ejecuciones de un punto de inicio $\mathbf{x}_0$ aleatorio perteneciente al universo de discurso de las variables de estado. En los resultados obtenidos, mostrados en las tablas 6.34 a 6.38, se puede comprobar que efectivamente sólo existe un estado de equilibrio prácticamente situado sobre el origen de coordenadas.

	Gauss-Newton	G-N con jacobiana	Levenberg-Marquardt	L-M con jacobiana
Estado de equilibrio	$\begin{pmatrix} -0,0064 \\ 4,7396e-07 \\ 0,0012 \\ -5,785e-07 \end{pmatrix}$	$\begin{pmatrix} -0,0064 \\ 4,7396e-07 \\ 0,0012 \\ -5,785e-07 \end{pmatrix}$	$\begin{pmatrix} -0,0064 \\ 4,7396e-07 \\ 0,0012 \\ -5,785e-07 \end{pmatrix}$	$\begin{pmatrix} -0,0064 \\ 4,7396e-07 \\ 0,0012 \\ -5,785e-07 \end{pmatrix}$
Nº Iteraciones	551	1465	551	1465
Evaluaciones de la función objetivo	2548	2932	2548	2932
Distancia del punto obtenido al 0	2,40302e-18	4,29154e-29	2,40302e-18	4,29154e-29
Tiempo empleado (s)	4,71242	9,84239	4,46588	9,71609

Tabla 6.35: Resultados de la búsqueda de estados de equilibrio para el sistema en lazo cerrado. Ejecución 2.

	Gauss–Newton	G–N con jacobiana	Levenberg–Marquardt	L–M con jacobiana
Estado de equilibrio	$\begin{pmatrix} -0,0064 \\ 4,7396e-07 \\ 0,0012 \\ -5,785e-07 \end{pmatrix}$	$\begin{pmatrix} -0,0064 \\ 4,7396e-07 \\ 0,0012 \\ -5,785e-07 \end{pmatrix}$	$\begin{pmatrix} -0,0064 \\ 4,7396e-07 \\ 0,0012 \\ -5,785e-07 \end{pmatrix}$	$\begin{pmatrix} -0,0064 \\ 4,7396e-07 \\ 0,0012 \\ -5,785e-07 \end{pmatrix}$
Nº Iteraciones	121	121	121	121
Evaluaciones de la función objetivo	590	244	590	244
Distancia del punto obtenido al 0	1,01851e-22	2,03138e-22	1,01851e-22	2,03138e-22
Tiempo empleado (s)	1,58014	0,864876	0,999378	0,806881

Tabla 6.36: Resultados de la búsqueda de estados de equilibrio para el sistema en lazo cerrado. Ejecución 3.

	Gauss–Newton	G–N con jacobiana	Levenberg–Marquardt	L–M con jacobiana
Estado de equilibrio	$\begin{pmatrix} -0,0064 \\ 4,7396e-07 \\ 0,0012 \\ -5,785e-07 \end{pmatrix}$	$\begin{pmatrix} -0,0064 \\ 4,7396e-07 \\ 0,0012 \\ -5,785e-07 \end{pmatrix}$	$\begin{pmatrix} -0,0064 \\ 4,7396e-07 \\ 0,0012 \\ -5,785e-07 \end{pmatrix}$	$\begin{pmatrix} -0,0064 \\ 4,7396e-07 \\ 0,0012 \\ -5,785e-07 \end{pmatrix}$
Nº Iteraciones	8	8	8	8
Evaluaciones de la función objetivo	45	18	45	18
Distancia del punto obtenido al 0	1,39525e-22	1,0284e-24	1,39525e-22	1,0284e-24
Tiempo empleado (s)	0,652616	0,132435	0,082876	0,074827

Tabla 6.37: Resultados de la búsqueda de estados de equilibrio para el sistema en lazo cerrado. Ejecución 4.

	Gauss-Newton	G-N con jacobiana	Levenberg-Marquardt	L-M con jacobiana
Estado de equilibrio	$\begin{pmatrix} -0,0064 \\ 4,7396e-07 \\ 0,0012 \\ -5,785e-07 \end{pmatrix}$	$\begin{pmatrix} -0,0064 \\ 4,7396e-07 \\ 0,0012 \\ -5,785e-07 \end{pmatrix}$	$\begin{pmatrix} -0,0064 \\ 4,7396e-07 \\ 0,0012 \\ -5,785e-07 \end{pmatrix}$	$\begin{pmatrix} -0,0064 \\ 4,7396e-07 \\ 0,0012 \\ -5,785e-07 \end{pmatrix}$
Nº Iteraciones	17	17	17	17
Evaluaciones de la función objetivo	70	36	70	36
Distancia del punto obtenido al 0	2,25028e-23	1,33639e-26	2,25028e-23	1,33639e-26
Tiempo empleado (s)	0,680848	0,193084	0,13714	0,135608

Tabla 6.38: Resultados de la búsqueda de estados de equilibrio para el sistema en lazo cerrado. Ejecución 5.

6.3. Conclusiones

Es preciso no obviar que la lógica borrosa permite reflejar de forma relativamente sencilla el conocimiento de un experto mediante reglas, lo que condujo a los primeros éxitos de la lógica borrosa a nivel industrial. En este capítulo se ha abordado el diseño de un controlador borroso basado en el conocimiento de un operario experto, demostrando mediante simulación el correcto funcionamiento del mismo.

La proliferación actual de sistemas complejos donde un operario no puede realizar el control tan sólo por su conocimiento sobre la planta, y donde se hace difícil el diseño mediante las técnicas clásicas de control no lineal, propicia la necesidad de una metodología de diseño general de síntesis de control no lineales. En este capítulo se han abordado varios ejemplos de diseño mediante la metodología presentada en el capítulo 5, la cual hace uso de todas las técnicas tratadas en los capítulos anteriores, esto es: modelado de la planta a partir de datos de entrada/salida (capítulo 3), estudio de los estados de equilibrio de ésta a través de su modelo borroso (capítulo 4), y diseño de un controlador borroso que garantice la estabilidad del sistema de control en lazo cerrado (capítulo 5). Un vez diseñado el controlador borroso, se ha procedido a estudiar el sistema de control borroso en lazo cerrado para así comprobar su correcto funcionamiento. Hay que destacar que en todos los casos estudiados se han obtenido los estados de equilibrio de la planta tan sólo a partir de su modelo borroso, es decir, considerando la planta completamente desconocida y tan sólo aplicando las técnicas propuestas en el capítulo 4 a su modelo borroso. Así, se ha conseguido conocer los estados de equilibrio de la planta con bastante exactitud, además de la estabilidad local de los mismos. También se ha conseguido, en todos los casos, el diseño de un controlador borroso que estabiliza el sistema de lazo cerrado en una región bastante amplia alrededor del estado de equilibrio deseado. Este paso se ha realizado, como el anterior, partiendo de la suposición de que la planta es desconocida, y sin aportar ningún conocimiento externo al proceso de diseño.

Conclusiones y desarrollos futuros

ESTE último capítulo cierra la memoria de la Tesis extrayendo las principales conclusiones derivadas de los trabajos de investigación presentados. En una segunda sección del capítulo se esgrimen las principales aportaciones y líneas de trabajo futuras que, sustentados en los desarrollos y propuestas presentados, se prevén puedan surgir a partir de la Tesis.

En esta Tesis se ha presentado una metodología de análisis y diseño completamente general para sistemas borrosos multivariantes no lineales, así como el software necesario para su implementación práctica. La metodología desarrollada está basada en tres pasos, los cuales han sido estudiados de forma sistemática a lo largo de todos los capítulos que conforman la Tesis: 1) identificación de la planta a partir de datos de entrada/salida (Capítulo 3); 2) obtención de información de la dinámica de la planta mediante la obtención de sus estados de equilibrio, y el estudio de la estabilidad local de éstos mediante la linealización del modelo borroso de la planta (Capítulo 4); y 3), diseño de un controlador borroso que garantice la estabilidad en lazo cerrado del sistema (Capítulo 5). La novedad y generalidad de la metodología desarrollada está sustentada en las razones siguientes:

1. Hace uso del hecho probado de que los modelos borrosos de tipo TSK son aproximadores universales, tanto de la función [210, 211, 212, 213] como de su derivada [214, 215, 216, 217].
2. Puede ser aplicada a sistemas no lineales multivariantes.
3. No es necesario conocer el modelo matemático de la planta ni poseer conocimientos previos sobre su dinámica, ya que su modelo puede ser obtenido a partir de datos de entrada/salida y su dinámica estudiada a partir de dicho modelo. Evidentemente, si se posee información relativa a la planta podrá utilizarse para facilitar tanto el proceso de modelado como el posterior diseño del controlador.
4. El proceso de modelado de la planta y el de diseño del controlador se consideran completamente independientes, de forma que cada uno de los modelos borrosos podrá tener un número distinto de reglas, distintas funciones de pertenencia, etc.

5. La complejidad tanto del modelo borroso de la planta como del controlador no está limitada, de forma que la metodología no está sujeta a especificidades ni simplificaciones.
6. En lo que respecta a los antecedentes tanto del modelo borroso de la planta como del controlador, cada regla se considera completamente independiente del resto, y no se limita el tipo ni la distribución de sus funciones de pertenencia, lo cual resultará especialmente útil si se realiza particionamiento por *clustering*.
7. Esta metodología permite la utilización de distintos tipos de funciones de pertenencia, incluso mezcladas en un mismo modelo borroso.
8. Durante el proceso de diseño no se realizan linealizaciones ni simplificaciones de ningún tipo.

7.1. Conclusiones

Atendiendo al estudio y la revisión del estado del arte realizados en el **Capítulo 2**, se puede concluir que la lógica borrosa se ha ganado el reconocimiento del que goza en la actualidad en base a los excelentes resultados que ha cosechado, tanto en el ámbito investigador como en el industrial. Gran parte de este reconocimiento se debe al hecho de que la lógica borrosa es capaz de generar la respuesta a una situación basándose en el conocimiento adquirido de ésta, aunque éste sea inexacto o incompleto, tomando como base la imprecisión del razonamiento humano, pero eso sí, bajo un planteamiento matemático. En este capítulo también se han estudiado los distintos tipos de sistemas borrosos, concluyéndose en base al estudio bibliográfico realizado, que los sistemas borrosos de tipo Takagi–Sugeno–Kang, o TSK de forma abreviada, son aproximadores universales de cualquier función y su derivada.

En el **Capítulo 3** se ha presentado una metodología que permite afrontar, en base a datos de entrada/salida, el modelado borroso en forma de ecuaciones de estado de un sistema de control multivariable no lineal. La formalización matemática de dicha metodología se aborda de forma totalmente general, es decir, sin aplicar ningún tipo de restricciones ni al tamaño del vector de estado, ni al del vector de control, ni al tipo de función de pertenencia utilizada en la definición de los conjuntos borrosos. Teniendo en cuenta estas consideraciones, se ha obtenido el modelo borroso de estado de una planta y de un controlador completamente generales, así como del sistema de control borroso en lazo cerrado formado por éstos.

Atendiendo al proceso de identificación de la planta en base a datos de entrada/salida, en este capítulo se han estudiado las distintas fases que componen dicho proceso, y se ha propuesto una nueva metodología, denominada híbrida, basada en la conjunción de la clásica metodología neuroborrosa y la heurística bioinspirada. Como se puede comprobar en [169], esta nueva técnica, aún en una fase inicial de desarrollo, resulta bastante prometedora, por lo que es previsible que sea objeto de un futuro trabajo de refinamiento y mejora.

El **Capítulo 4** propone una metodología de análisis de los sistemas borrosos basada en la teoría de control no lineal, y cuya formalización matemática se ha desarrollado a lo largo del propio capítulo. Partiendo del modelo borroso de estado obtenido en el capítulo anterior se calcula el modelo lineal de un modelo de estado borroso en un punto. Dado que en el modelo de partida no se han definido restricciones de ningún tipo, el cálculo del modelo linealizado también puede considerarse completamente general. De igual forma, partiendo del modelo matemático equivalente del sistema de control borroso en lazo cerrado, manteniendo la misma filosofía de generalidad que en el caso anterior, y considerándose que la planta y el controlador son a todos los efectos sistemas completamente independientes, se ha obtenido su modelo linealizado en un punto, o sea, su matriz jacobiana.

Adicionalmente, en el capítulo 4 también se ha presentado un algoritmo para la obtención de los estados de equilibrio del sistema borroso, independientemente de si se considera únicamente la planta o el sistema en lazo cerrado. Con objeto de mejorar la convergencia y la precisión de los resultados de estos algoritmos, se propone incorporar a los mismos el cálculo explícito de la matriz jacobiana obtenida previamente.

El modelo linealizado de un sistema no lineal permite extraer generalmente mucha información sobre el comportamiento de éste, por lo que la consecución del algoritmo para su cálculo, propuesto en este capítulo, puede ser de gran ayuda en el análisis de sistemas dinámicos. Si además se añade la posibilidad de obtener los estados de equilibrio del sistema no lineal, la información que puede extraerse del mismo, tan sólo a partir de su modelo borroso, es mucho más interesante. Por ejemplo, se pueden hallar los estados de equilibrio del sistema, y estudiar la estabilidad local de los mismos a través del modelo linealizado, y todo ello sin haber requerido previamente el conocimiento explícito de las ecuaciones del sistema no lineal, si no a partir de un modelo borroso obtenido en base a datos de entrada/salida.

El **Capítulo 5** cierra la metodología general de análisis y diseño de sistemas borrosos mediante la propuesta de un teorema de estabilidad para sistemas no lineales. Este teorema permite afrontar la síntesis de controladores borrosos estables por diseño, es decir, donde la estabilidad del sistema en lazo cerrado quede garantizada en el propio proceso de diseño, de forma más flexible que otros teoremas existentes, como el teorema de Krasovskii. En este mismo capítulo también se ha desarrollado un algoritmo de diseño que permite aplicar el teorema propuesto a la síntesis de un controlador borroso para un caso general, es decir, donde inicialmente la planta fuese completamente desconocida y sólo se tuviese acceso a su modelo borroso.

El **Capítulo 6**, ha estado dedicado a resolver una serie de aplicaciones prácticas sobre análisis y el diseño de sistemas de control borroso mediante las técnicas desarrolladas en los capítulos previos. En primer lugar se ha diseñado un controlador borroso basado en el conocimiento de un operario experto para una planta compleja como es una grúa porta contenedores, de esta forma se pretende demostrar de forma práctica las posibilidades que la lógica borrosa brinda de realizar *control con palabras*, y que, como se vio en el capítulo 2, propició la inclusión masiva de la lógica borrosa en los ámbitos industriales.

La siguiente sección del capítulo 6 estuvo dedicada a la metodología de análisis y diseño que constituye el cuerpo de esta Tesis. En esta sección se estudiaron tres sistemas dinámicos

no lineales. Sus dinámicas se suponen totalmente desconocidas para así presentar de la forma más general posible la metodología de diseño propuesta en el capítulo 5:

1. Identificación de la planta y representación en forma de modelo borroso de estado.
2. Estudio de la planta y obtención de información dinámica de la misma a partir de su modelo borroso.
3. Diseño de un controlador borroso que garantice la estabilidad asintótica en lazo cerrado.

Los tres sistemas estudiados han sido: un ejemplo ilustrativo de múltiple entrada y múltiple salida (MIMO), que pretende facilitar la comprensión de la metodología de diseño propuesta; un péndulo, que representa un sistema no lineal con múltiples estados de equilibrio, cuyo objetivo es demostrar la validez de la fase de análisis y diseño; y un balancín, que representa un sistema no lineal de uso común en la literatura relacionada con sistemas no lineales.

Una vez obtenido un modelo borroso de cada uno de los sistemas, se ha procedido a estudiar sus estados de equilibrio y la estabilidad local de los mismos. Hay que destacar que en todos los casos se han obtenido unos estados de equilibrio muy similares a los reales, partiendo únicamente del modelo borroso de la planta, y que el estudio de estabilidad local realizado en todos los casos demostró la validez de la metodología propuesta. Finalmente se ha procedido a diseñar un controlador borroso mediante el algoritmo de diseño propuesto en el capítulo 5, cuya estabilidad se ha demostrado mediante simulación y mediante un nuevo análisis de los estados de equilibrio del sistema, ahora ya realimentado, y de su estabilidad local.

7.2. Desarrollos futuros

Normalmente un trabajo de investigación no suele ser algo cerrado, si no más bien todo lo contrario, una puerta abierta al estudio de nuevas situaciones que se plantean durante el desarrollo de la propia investigación. En una Tesis doctoral este hecho es, si cabe, aún más notorio; ya que la extensión del trabajo realizado tanto en el tiempo como en los contenidos abordados, va generando constantemente nuevas ideas que necesariamente deben ser aparcadas para su estudio posterior.

En esta sección se pretende presentar de forma resumida algunas de las posibles líneas futuras de investigación que pueden manar de este trabajo:

Con respecto al **Capítulo 3**, dados los buenos resultados preliminares que ha mostrado la metodología híbrida propuesta, se hace necesario una profundización en su estudio y un mayor desarrollo de los algoritmos de identificación paramétrica. Uno de los mayores inconvenientes que presenta esta metodología, es la necesidad de discretizar el espacio de soluciones para aplicar la heurística de colonias de hormigas; sin embargo, trabajos publicados recientemente [344, 345, 346, 347] han resuelto este problema, permitiendo la aplicación de esta heurística a la optimización de problemas en espacios continuos. Por lo

tanto, se plantea como una línea de investigación futura la modificación de la metodología híbrida propuesta para adaptarla a los nuevos algoritmos existentes de optimización mediante colonias de hormigas para espacios continuos.

En el **Capítulo 4** se ha planteado la necesidad de continuar con la formalización de los sistemas borrosos mediante la aplicación de los métodos existentes en la teoría de control no lineal al estudio de éstos. Siguiendo con esta filosofía, es posible extender el estudio realizado durante este capítulo a sistemas no lineales complejos, con idea de diseñar algoritmos que permitan, por ejemplo, la detección de bifurcaciones o ciclos límites. Actualmente se está iniciando un trabajo a este respecto. Para ello se está empleando como planta un circuito biestable con diodo túnel [220], cuyos estados de equilibrio se modifican drásticamente con una pequeña variación en los parámetros del circuito, pudiendo pasar de poseer uno a tres estados de equilibrio.

En el **Capítulo 5** se ha propuesto un algoritmo de diseño que permite la obtención de un controlador borroso que estabilice el sistemas de control en lazo cerrado para una planta dada. Este algoritmo se basa en la búsqueda de los parámetros del controlador para el cumplimiento del teorema 5.7 mediante la aplicación recursiva de una fase de búsqueda global y otra local. Para la búsqueda global se ha utilizado un algoritmo genético [341], mientras que para la búsqueda local se ha utilizado el algoritmo numérico de Nelder–Mead [342]. Este procedimiento resulta lento, y la convergencia del algoritmo no puede garantizarse. Por lo tanto, como trabajo futuro surge la posibilidad de optimizar el algoritmo de diseño presentado en esta Tesis. Esta optimización puede realizarse de forma simultánea por dos vertientes:

- Optimización de la programación del propio algoritmo. Esta optimización podría incidir en un aumento considerable de la velocidad de cómputo y, por lo tanto, en una reducción del tiempo de convergencia de éste. Esta optimización puede realizarse sustituyendo todo el proceso programado en MATLAB® por un código C/C++, mucho más rápido que el primero. Además, si la implementación del algoritmo en C/C++ se aprovecha, en base a un estudio previo que sería necesario, para paralelizar ciertas fases del mismo, se puede aumentar aún más la velocidad de convergencia.

Otra posibilidad es la preparación del algoritmo diseñado para su ejecución en sistemas de cómputo basados en GPUs, cuya velocidad en el procesamiento de cálculos paralelos está provocando que la comunidad científica utilice cada vez más esta alternativa a la computación tradicional [348, 349].

- Optimización de la metodología de diseño empleada en el algoritmo. El algoritmo de diseño debe realizar la búsqueda de las soluciones al problema en un espacio de dimensión muy elevada, ya que debe obtener los valores de cada uno de los parámetros adaptables del consecuente de cada una de las reglas, lo que supone $n + 1$ parámetros por cada regla del controlador borroso, donde n el orden del sistema. Además, el mismo algoritmo debe obtener los $n(n + 1)/2$ parámetros necesarios para definir cada una de las matrices P_q , donde $q = 1..s$ y s es el número de regiones en las que se divide el universo de discurso de las variables de estado para la realización del diseño del controlador. Por lo tanto, la incorporación al algoritmo de información

que facilite la estabilización del sistema de control, podría redundar en una enorme reducción del tiempo de cómputo y una mejora en la convergencia del mismo. Esta información podría provenir, por ejemplo, de un análisis basado en linealización. Estos valores seguramente no permitirían el cumplimiento del teorema 5.7, pero sí acotarían el problema al disponer de valores cercanos a los válidos para estabilizar el sistema.

La optimización de la metodología también puede basarse en un estudio puramente teórico de las funciones no lineales que representan el sistema de control borroso en lazo cerrado, ver sección 3.2.1. Este estudio es sin lugar a dudas arduo, dada la propia complejidad y el carácter general de dichas ecuaciones, pero proporcionaría un mecanismo de síntesis de controladores borrosos general y directo.

Además de las posibles líneas de investigación que se han comentado hasta este instante, también resulta factible e interesante la iniciación de una línea más inmediata, el desarrollo de la metodología equivalente para sistemas discretos.

Finalmente cabría comentar que, tal como se afirma en [350], la investigación en el área de control no ha explotado aún suficientemente la idea de la propagación de la incertidumbre a las funciones de pertenencia; lo que se corresponde con los denominados conjuntos borrosos tipo-2. Los sistemas borrosos tipo-2 pueden mejorar notablemente el modelado de los sistemas dinámicos no lineales, empleando para ello un número menor de reglas [351], por lo tanto, su utilización puede ser muy beneficiosa para el objetivo de analizar y diseñar sistemas de control no lineales mediante lógica borrosa. La incorporación de los sistemas borrosos tipo-2 a la metodología presentada en este texto supone todo un reto, y por lo tanto una línea de investigación muy interesante. Para realizarlo sería necesario extender todas las técnicas desarrolladas en esta Tesis a dicho tipo de conjuntos borrosos, y seguramente, diseñar nuevos algoritmos para el análisis y diseño de este tipo de sistemas.

Apéndice A

Leyenda de índices y variables

A continuación se presenta a modo de ayuda una tabla donde se muestran ordenados alfabéticamente los índices y variables más importantes empleados en los desarrollos de este texto.

Término	Descripción
a_{0i}^l, a_{0i}^p	Término constante correspondiente al consecuente afín de la regla l o p del modelo borroso de la planta.
a_{ki}^l, a_{ki}^p	Término constante correspondiente al consecuente TSK asociado a la variable de estado x_k de la regla l o p del modelo borroso de la planta.
$a_{ki}(\mathbf{x}, \mathbf{u})$	Término variable del modelo borroso de la planta, dependientes de las variables de estado y control, asociado a la variable de estado x_k y a la i -ésima ecuación del proceso.
$a_{qi}(\mathbf{x}, \mathbf{u})$	Estos términos son los mismos que los definidos mediante el índice k en vez de q , salvo que se utilizan cuando se hace referencia a la derivada con respecto a la variable de estado x_q .
A_{ki}^l	Conjunto borroso definido en el universo de discurso de la variable de estado x_k , en la regla l de la i -ésima ecuación de estado de la planta.
b_{ji}^l, b_{ji}^p	Término constante correspondiente al consecuente TSK asociado a la señal de control u_j de la regla l o p del modelo borroso de la planta.
$b_{ji}(\mathbf{x}, \mathbf{u})$	Término variable del modelo borroso de la planta, dependientes de las variables de estado y control, asociado a la señal de control u_j y a la i -ésima ecuación del proceso.
B_{ji}^l	Conjunto borroso definido en el universo de discurso de la señal de control u_j , en la regla l de la i -ésima ecuación de estado de la planta.

Término	Descripción
c_{0i}^r, c_{0i}^s	Término constante correspondiente al consecuente afín de la regla r o s del modelo borroso del controlador.
c_{kj}^r, c_{kj}^s	Término constante correspondiente al consecuente TSK asociado a la variable de estado x_k de la regla r o s del modelo borroso del controlador.
$c_{kj}(\mathbf{x})$	Término variable del modelo borroso del controlador, dependientes de las variables de estado y asociado a la variable de estado x_k y la j -ésima señal de control.
$c_{qj}(\mathbf{x})$	Estos términos son los mismos que los definidos mediante el índice k en vez de q , salvo que se utilizan cuando se hace referencia a la derivada con respecto a la variable de estado x_q .
C_{kj}^r	Conjunto borroso definido en el universo de discurso de la variable de estado x_k , en la regla r de la j -ésima ecuación del controlador.
i	Indicador de la i -ésima ecuación representativa del proceso, $i = 1..n$.
j	Índice que recorre las señales de control, $j = 1..m$.
$\mathbf{J}(\mathbf{x})$	Matriz Jacobiana del sistema de control. Tiene dimensión $n \times n$ y cada elemento se referencia como J_{iq} .
k	Índice que recorre las variables de estado del sistema. Si se utiliza el vector de estados extendido, $k = 0, 1..n$, de lo contrario, $k = 1..n$.
l	Índice que recorre las reglas del modelo borroso de la planta, $l = 1..M_i$.
m	Número de señales de control (orden del vector de control).
M_i	Número de reglas que representan la i -ésima ecuación de estado del modelo borroso de la planta.
n	Número de variables de estado (orden del sistema).
N_j	Número de reglas que representan la j -ésima ecuación del controlador.
p	Índice que recorre las reglas del modelo borroso de la planta en bucles donde ya se emplea el índice l , $p = 1..M_i$.
P_q	Matriz simétrica definida positiva para la q -ésima región en la que se divide el universo de discurso de las variables de estado para el diseño del controlador.
q	Coordenada del vector de estado con respecto a la que se deriva para calcular la matriz jacobiana del sistema, $q = 1..n$.

Término	Descripción
	En el contexto del diseño del controlador, el índice $q = 1..s$ recorre las regiones en las que se divide el universo de discurso de las variables de estado.
r	Índice empleado para recorrer las reglas del modelo borroso del controlador., $r = 1..N_j$.
$Rc^{(r,j)}$	Regla r de la ecuación de control j del modelo borroso del controlador.
$Rp^{(l,i)}$	Regla l de la ecuación de estado i del modelo borroso de la planta.
s	Índice empleado para recorrer las reglas del modelo borroso del controlador en bucles donde ya se emplea el índice r , $s = 1..N_j$. En el contexto del diseño del controlador, s representa el número de regiones en las que se divide el universo de discurso de las variables de estado para realizar dicho diseño.
u_j, u_v	j -ésima, o v -ésima señal de control del sistema.
U	Universo de discurso de las señales de control.
v	Índice que recorre las señales de control en bucles donde ya se emplea el índice j , $v = 1..m$.
w_i^l, w_i^p	Grado de activación de la regla l o p de la i -ésima ecuación de estado de la planta.
x_k, x_q	k -ésima, o q -ésima, variable de estado.
$\tilde{x}_k$	k -ésima variable de estado cuando se emplea el vector de estado extendido.
X	Universo de discurso de las variables de estado.
α_{ji}^l	Conjunto de parámetros adaptables del antecedente de la regla l de la i -ésima ecuación de estado del modelo borroso de la planta, asociada a la señal de control u_j .
μ_{ki}^l	Función de pertenencia de la regla l de la i -ésima ecuación de estado de la planta, asociada a la variable de estado x_k .
μ_{ji}^l	Función de pertenencia de la regla l de la j -ésima ecuación de estado de la planta, asociada a la señal de control u_j .
μ_{kj}^r	Función de pertenencia de la regla r de la j -ésima ecuación del controlador, asociada a la variable de estado x_k .

Término	Descripción
$\varphi_q(\mathbf{x})$	Función que delimita la región Θ_q del universo de discurso de las variables de estado para el diseño del controlador.
Φ_{kj}^r	Conjunto de parámetros adaptables del antecedente de la regla r de la j -ésima ecuación del controlador, asociada a la variable de estado x_k .
Ψ_{iq}	Conjunto de parámetros adaptables de la i -ésima función de pertenencia que declara la región Θ_q en la metodología de diseño del controlador de la sección 5.4.
Θ_q	q -ésima región en la que se divide el universo de discurso de las variables de estado para el diseño del controlador.
σ_{ki}^l	Conjunto de parámetros adaptables del antecedente de la regla l de la i -ésima ecuación de estado del modelo borroso de la planta, asociada a la variable de estado x_k .
ω_j^r, ω_j^s	Grado de activación de las reglas del controlador.

Tabla A.1: *Leyenda de índices y variables.*

Derivadas de las principales funciones de pertenencia

Para calcular la derivada del grado de activación de las reglas de la planta de un sistema de control borroso, así como la derivada del grado de activación de las reglas del controlador, es necesario obtener la derivada de las funciones de pertenencia con respecto a su variable independiente; es decir, es necesario calcular $d\mu(x)/dx$ o $d\mu(u)/du$.

A continuación se calculan las derivadas de las funciones de pertenencia más comunes, definidas en el epígrafe *Tipos básicos de funciones de pertenencia* de la sección 2.3.1 (página 24). Aunque las ecuaciones que definen estas funciones de pertenencia ya han sido descritas en dicho epígrafe, se reescribirán para facilitar la lectura de este apéndice.

Nota. Puesto que todas las funciones de pertenencia son continuas y derivables, o derivables a trozos, y como el conjunto de puntos singulares de las funciones derivables a trozos tiene medida nula¹, a efectos prácticos se puede asumir que son continuas y derivables en todo su rango. En los puntos singulares se puede suponer que la derivada es indistintamente la de un punto infinitesimalmente cercano por la derecha, x^+ , o la de un punto infinitesimalmente cercano por la izquierda, x^- , o como se ha optado en este texto, como la media de dichas derivadas.

Derivada de la función de pertenencia triangular

Sea la función de pertenencia triangular definida por:

¹Un subconjunto de $\mathbb{R}^n$ se dice de medida nula si para todo $\varepsilon > 0$ existe una familia finita o infinita numerable de intervalos abiertos cuyo volumen total es menor que ε . Todos los conjuntos numerables son nulos, así como los subconjuntos de $\mathbb{R}^n$ de dimensión inferior a n , como por ejemplo una línea o curva en $\mathbb{R}^2$.

$$\mu_{Tri}[a, b, c](x) = \begin{cases} \frac{x-a}{b-a} & \text{si } a < x \leq b \\ \frac{c-x}{c-b} & \text{si } b < x < c \\ 0 & \text{en otro caso,} \end{cases} \quad (2.9)$$

su derivada respecto a x es:

$$\frac{d\mu_{Tri}[a, b, c](x)}{dx} = \begin{cases} \frac{1}{2(b-a)} & \text{si } x = a \\ \frac{1}{b-a} & \text{si } a < x < b \\ \frac{1}{2} \left(\frac{1}{b-a} + \frac{1}{c-b} \right) & \text{si } x = b \\ \frac{1}{c-b} & \text{si } b < x < c \\ \frac{1}{2(c-b)} & \text{si } x = c \\ 0 & \text{en otro caso.} \end{cases} \quad (B.1)$$

Derivada de la función de pertenencia trapezoidal

La función de pertenencia trapezoidal se define mediante:

$$\mu_{Trap}[a, b, c, d](x) = \begin{cases} \frac{x-a}{b-a} & \text{si } a < x < b \\ 1 & \text{si } b \leq x \leq c \\ \frac{d-x}{d-c} & \text{si } c < x < d \\ 0 & \text{en otro caso,} \end{cases} \quad (2.11)$$

por lo que su derivada respecto a x es:

$$\frac{d\mu_{Trap}[a, b, c, d](x)}{dx} = \begin{cases} \frac{1}{b-a} & \text{si } a < x < b \\ \frac{1}{2(b-a)} & \text{si } x = a \text{ o } x = b \\ \frac{1}{d-c} & \text{si } c < x < d \\ \frac{1}{2(d-c)} & \text{si } x = c \text{ o } x = d \\ 0 & \text{en otro caso.} \end{cases} \quad (\text{B.2})$$

Derivada de la función de pertenencia gaussiana

Sea la función de pertenencia gaussiana definida por:

$$\mu_{Gauss}[c, \beta](x) = e^{-\left(\frac{x-c}{\beta}\right)^2}, \quad (2.12)$$

su derivada respecto a x es:

$$\frac{d\mu_{Gauss}[c, \beta](x)}{dx} = -2 \left(\frac{x-c}{\beta^2} \right) \mu_{Gauss}[c, \beta](x). \quad (\text{B.3})$$

Derivada de la función de pertenencia gaussiana doble

La función de pertenencia gaussiana doble se define mediante la expresión:

$$\mu_{Gauss2}[c_1, c_2, \beta_1, \beta_2](x) = \mu_1(x)\mu_2(x), \quad (2.13a)$$

siendo

$$\mu_1(x) = \begin{cases} \mu_{Gauss}[c_1, \beta_1](x) & \text{Si } x < c_1 \\ 1 & \text{Si } x \geq c_1 \end{cases} \quad (2.13b)$$

y

$$\mu_2(x) = \begin{cases} \mu_{Gauss}[c_2, \beta_2](x) & \text{Si } x > c_2 \\ 1 & \text{Si } x \leq c_2, \end{cases} \quad (2.13c)$$

por lo que su derivada respecto a x es:

$$\frac{d\mu_{Gauss2}[c_1, c_2, \beta_1, \beta_2](x)}{dx} = \begin{cases} \frac{d\mu_{Gauss}[c_1, \beta_1](x)}{dx} & \text{si } x < c_1 \text{ y } x < c_2 \\ \frac{d\mu_{Gauss}[c_2, \beta_2](x)}{dx} & \text{si } x > c_1 \text{ y } x > c_2 \\ \frac{d\mu_{Gauss}[c_1, \beta_1](x)}{dx} \mu_{Gauss}[c_2, \beta_2](x) + \frac{d\mu_{Gauss}[c_2, \beta_2](x)}{dx} \mu_{Gauss}[c_1, \beta_1](x) & \text{en otro caso.} \end{cases} \quad (B.5)$$

Derivada de la función de pertenencia campana

Sea la función de pertenencia campana definida por:

$$\mu_{Campana}[\alpha, \beta, c](x) = \frac{1}{1 + \left| \frac{x-c}{\alpha} \right|^{2\beta}}. \quad (2.14)$$

La expresión anterior puede reescribirse eliminando el valor absoluto como:

$$\mu_{Campana}[\alpha, \beta, c](x) = \begin{cases} \frac{1}{1 + \left(\frac{-(x-c)}{\alpha} \right)^{2\beta}} & \text{si } x < c \\ 1 & \text{si } x = c \\ \frac{1}{1 + \left(\frac{(x-c)}{\alpha} \right)^{2\beta}} & \text{si } x > c \end{cases} \quad (B.6)$$

Calculando la derivada de la expresión anterior respecto a x , se obtiene:

$$\frac{d\mu_{Campana}[\alpha, \beta, c](x)}{dx} = \begin{cases} \frac{2\beta}{\alpha} (\mu_{Campana}[\alpha, \beta, c](x))^2 \left(\frac{-(x-c)}{\alpha} \right)^{2\beta-1} & \text{si } x < c \\ 0 & \text{si } x = c \\ -\frac{2\beta}{\alpha} (\mu_{Campana}[\alpha, \beta, c](x))^2 \left(\frac{(x-c)}{\alpha} \right)^{2\beta-1} & \text{si } x > c, \end{cases} \quad (B.7)$$

que puede expresarse de forma simplificada por:

$$\frac{d\mu_{Campana}[\alpha, \beta, c](x)}{dx} = -\frac{2\beta}{\alpha} (\mu_{Campana}[\alpha, \beta, c](x))^2 \left| \frac{x-c}{\alpha} \right|^{2\beta-1} \text{sign}(x-c), \quad (B.8)$$

donde $\text{sign}(\cdot)$ es la función signo.

Derivada de la función de pertenencia S

Sea la función de pertenencia S definida por:

$$\mu_S[a, b](x) = \begin{cases} 0 & \text{si } x \leq a \\ 2 \left(\frac{x-a}{b-a} \right)^2 & \text{si } a < x \leq \frac{a+b}{2} \\ 1 - 2 \left(\frac{b-x}{b-a} \right)^2 & \text{si } \frac{a+b}{2} < x < b \\ 1 & \text{si } x \geq b, \end{cases} \quad (2.15)$$

su derivada respecto a x es:

$$\frac{d\mu_S[a, b](x)}{dx} = \begin{cases} 0 & \text{si } x \leq a, x \geq b \\ 4 \frac{(x-a)}{(b-a)^2} & \text{si } a < x \leq \frac{a+b}{2} \\ 4 \frac{(b-x)}{(b-a)^2} & \text{si } \frac{a+b}{2} < x < b. \end{cases} \quad (\text{B.9})$$

Derivada de la función de pertenencia Z

Sea la función de pertenencia Z definida por:

$$\mu_Z[a, b](x) = \begin{cases} 1 & \text{si } x \leq a \\ 1 - 2 \left(\frac{x-a}{a-b} \right)^2 & \text{si } a < x < \frac{a+b}{2} \\ 2 \left(\frac{b-x}{a-b} \right)^2 & \text{si } \frac{a+b}{2} \leq x < b \\ 0 & \text{si } x \geq b, \end{cases} \quad (2.16)$$

su derivada respecto a x es:

$$\frac{d\mu_Z[a, b](x)}{dx} = \begin{cases} 0 & \text{si } x \leq a, x \geq b \\ -4 \frac{(x-a)}{(a-b)^2} & \text{si } a < x < \frac{a+b}{2} \\ -4 \frac{(b-x)}{(a-b)^2} & \text{si } \frac{a+b}{2} \leq x < b. \end{cases} \quad (\text{B.10})$$

Derivada de la función de pertenencia Π

Como la función de pertenencia $\Pi[a, b, c, d]$ se define como el producto de las funciones de pertenencia $\mu_S[a, b](x)$ y $\mu_Z[c, d](x)$, su derivada puede calcularse como:

$$\frac{d\mu_{\Pi}[a, b, c, d](x)}{dx} = \frac{d\mu_S[a, b](x)}{dx} \mu_Z[c, d](x) + \frac{d\mu_Z[c, d](x)}{dx} \mu_S[a, b](x), \quad (\text{B.11})$$

donde la derivada respecto de x de las funciones de pertenencia S y Z se pueden obtener a partir de las expresiones (B.9) y (B.10), respectivamente.

Derivada de la función de pertenencia sigmoidal

Sea la función de pertenencia sigmoidal definida por:

$$\mu_{Sigm}[c, \alpha](x) = \frac{1}{1 + e^{\alpha(c-x)}}, \quad (2.17)$$

su derivada respecto a x se calcula mediante la expresión:

$$\frac{d\mu_{Sigm}[c, \alpha](x)}{dx} = \alpha e^{\alpha(c-x)} (\mu_{Sigm}[c, \alpha](x))^2. \quad (\text{B.12})$$

A partir de la derivada de la función de pertenencia sigmoidal es inmediata la deducción de las derivadas de las funciones de pertenencia dadas por el producto o la diferencia de funciones sigmoidales.

Descomposición de Cholesky

Sea $\mathbf{S}$ una matriz cuadrada y simétrica de dimensión $n \times n$, y sea $\mathbf{x}$ un vector columna arbitrario y no nulo de dimensión $n \times 1$, se dice que $\mathbf{S}$ es una matriz definida positiva, y se denota como $\mathbf{S} > 0$, si cumple la condición dada en la ecuación (C.1). Si $\mathbf{S} > 0$, sus autovalores, o valores propios, son todos positivos.

$$\mathbf{x}^T \mathbf{S} \mathbf{x} > 0 \quad (\text{C.1})$$

Si $\mathbf{S}$ es una matriz simétrica definida positiva, entonces siempre existe una matriz triangular superior (o inferior) $\mathbf{T}$ de la misma dimensión, tal que

$$\mathbf{S} = \mathbf{T}^T \mathbf{T}. \quad (\text{C.2})$$

A la descomposición de la matriz $\mathbf{S}$ en el producto $\mathbf{T}^T \mathbf{T}$ se le llama descomposición de Cholesky [352]. Si los elementos de la matriz $\mathbf{S}$ se denotan por s_{ij} , con $s_{ij} = s_{ji}$ para $i, j = 1..n$, y los de la matriz $\mathbf{T}$ como t_{ij} , con $t_{ij} = 0$ para $i > j$, es decir, considerando $\mathbf{T}$ como matriz triangular superior, se demuestra que los elementos de $\mathbf{T}$ se pueden calcular por filas, de izquierda a derecha y de arriba a abajo, mediante las siguientes expresiones [352]:

$$t_{ii} = \sqrt{s_{ii} - \sum_{k=1}^{i-1} t_{ki}^2} \quad (\text{C.3})$$

$$t_{ij} = \begin{cases} \frac{s_{ij} - \sum_{k=1}^{i-1} t_{ki} t_{kj}}{t_{ii}} & \text{si } i < j \\ 0 & \text{si } i \geq j \end{cases} \quad (\text{C.4})$$

Tal como se ha enunciado, la descomposición de Cholesky permite obtener a partir de una matriz simétrica definida positiva, una matriz triangular cuyos elementos de la diagonal son positivos y que satisface la ecuación (C.2). Sin embargo, aplicando el mismo principio de forma inversa, se puede obtener una matriz simétrica definida positiva a

partir de una matriz triangular aplicando la ecuación (C.2), siempre que los elementos de la diagonal de la matriz triangular sean positivos. Es decir, siendo $\mathbf{T}$ una matriz triangular

$$\mathbf{T} = \begin{pmatrix} t_{11} & t_{12} & \dots & t_{1n} \\ 0 & t_{22} & \dots & t_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & t_{nn} \end{pmatrix}, \quad (\text{C.5})$$

tal que $t_{ii} > 0$ para $i = 1..n$, se puede demostrar que la matriz $\mathbf{S}$ calculada mediante la ecuación (C.2) es una matriz simétrica definida positiva.

Definición del software

En este apéndice se incluyen los ficheros de cabecera del software diseñado. El software está implementado en C/C++. Para más información sobre estas clases puede dirigirse a las secciones 3.4 y 4.5.

D.1. Clase Membership

Esta clase permite almacenar una función de pertenencia con todos sus parámetros, y contiene los métodos necesarios para su tratamiento. El diagrama UML de esta clase se muestra en la figura 3.16.

```

1  /* #####
2
3      Copyright 2004–2009 Antonio Javier áBarragn ñPia , antonio.barragan@diesia.uhu.
         es , http://www.uhu.es/antonio.barragané
4      Jos Manuel úAndjar áMrquez , andujar@diesia.uhu.es
5      Manuel Emilio úGegndez Árias , e-mail: gegundez@dmat.uhu.es
6
7      Dpto. de Ing. óElectrnica , de Sistemas áInformticos y áAutomtica
8      E.P.S. La áRbida, University of Huelva (Spain)
9
10     For more information , please contact with authors.
11
12  ##### */
13
14  /*
15     This software is free software: you can redistribute it and/or modify
16     it under the terms of the GNU General Public License as published by
17     the Free Software Foundation, either version 3 of the License , or
18     (at your option) any later version .
19
20     This software is distributed in the hope that it will be useful ,
21     but WITHOUT ANY WARRANTY; without even the implied warranty of
22     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
23     GNU General Public License for more details .
24
25     You should have received a copy of the GNU General Public License
26     along with this program . If not , see <http://www.gnu.org/licenses/> .
27  */

```



```

28
29 #ifndef _MEMBERSHIP_H
30 #define _MEMBERSHIP_H_
31
32 #include <iostream>
33 #include "ext_math.h"
34 #include "messages.h"
35
36 #if defined (MATLAB_MEX_FILE)
37     #include "mex.h"
38 #endif
39
40 #define MAX_TIPO_NAME 32
41
42 // Tipos de Funciones de Membership Implementadas
43
44 /*
45 Si se cambia la enumeracion, deben modificarse los arrays TIPO_NOMBRES y
46 TIPO_N_PARAM para que sean coherentes con esta.
47
48 Si se modifica el 1er o ultimo elemento de la enumeracion, hay que cambiar el
49 codigo de 'membership.cpp' para que las comparaciones que se hacen es este
50 sean correctas con la declaracion.
51 */
52
53 enum TYPE_MF // Tipos posibles de Funciones de Membership
54 {
55     NOMF=0,
56     GAUSSMF,
57     GAUSS2MF,
58     TRAPEZOIDALMF,
59     TRAP2MF,
60     CAMPANAMF,
61     PIMF,
62     PSIGMOIDALMF,
63     SMF,
64     SIGMOIDALMF,
65     SIGMOIDAL2MF,
66     TRIANGULARMF,
67     ZMF
68 };
69
70 static const char *TIPO_NOMBRES[]=
71 { // Nombres de las Funciones de Membership segun TYPE_MF, todos en Mayuscula
72     "NOMF", "GAUSSMF", "GAUSS2MF", "TRAPMF", "TRAP2MF", "GBELLMF", "PIMF", "PSIGMF", "SMF",
73     "SIGMF", "DSIGMF", "TRIMF", "ZMF"
74 };
75
76 static const int TIPO_N_PARAM[]=
77 { // Numero de Parametros de cada tipo segun TYPE_MF
78     0, 2, 4, 4, 8, 3, 4, 4, 2, 2, 4, 3, 2
79 };
80 // Declarations #####
81
82 class Membership // -Clase Padre-
83 {
84 protected:
85     double *param; // Vector de parametros que definen la Funcion de Membership
86     int n; // Numero de parametros de la Funcion de Membership
87     TYPE_MF type_mf;
88 public:

```

```

89     virtual double eval(double x) const; // Evaluar la Funcion de Membership
90     virtual double evalder(double x) const; // Derivada de la F. de Membership
91     virtual int test(void) const; // Modifica los parametros para que sean
        correctos
92     int n_param(void) const; // Lee el numero de parametros que definen la F.
        Membership
93     TYPE_MF tipo(void) const;
94     int tipo(TYPE_MF type_mf);
95     int edit(int indice, double valor); // Editar un parametro. 0 si error
96     double leer(int indice) const; // Leer un parametro
97     Membership &operator=(const Membership &P);
98     Membership(const Membership &P);
99     Membership();
100    ~Membership();
101 };
102
103 Membership * crearFP(TYPE_MF t); // Constructor virtual
104
105 // Nomf #####
106 class Nomf:public Membership // En MATLAB no existe
107 { // Se emplea para inicializar las reglas de forma segura
108     public:
109         double eval(double x) const;
110         double evalder(double x) const;
111         Nomf(void);
112 };
113
114 // Gaussmf #####
115 class Gaussmf:public Membership // En MATLAB es gaussmf
116 {
117     public:
118         double eval(double x) const;
119         double evalder(double x) const;
120         int test(void) const; // Si los parametros de la F.P. son validos retorna 0
121         Gaussmf(double Centro=0.5, double Ancho=0.3); // Ancho_MATLAB=Ancho/sqrt(2);
122 };
123
124 // Gauss2mf #####
125 class Gauss2mf:public Membership // En MATLAB es gauss2mf
126 {
127     public:
128         double eval(double x) const;
129         double evalder(double x) const;
130         int test(void) const; // Modifica los parametros para que sean correctos
131         Gauss2mf(double Centro1=0.45, double Ancho1=0.25, double Centro2=0.55,
            double Ancho2=0.25); // Ancho_MATLAB=Ancho/sqrt(2);
132 };
133
134 // Trapezoidalmf #####
135 class Trapezoidalmf:public Membership // En MATLAB es trapmf
136 {
137     public:
138         double eval(double x) const;
139         double evalder(double x) const;
140         int test(void) const; // Modifica los parametros para que sean correctos
141         Trapezoidalmf(double a=0.05, double b=0.45, double c=0.55, double d=0.95);
142 };
143
144 // Trap2mf #####
145 class Trap2mf:public Membership // En MATLAB no existe pero la he creado
146 {
147     public:

```

```

148     double eval(double x) const;
149     double evalder(double x) const;
150     int test(void) const; // Modifica los parametros para que sean correctos
151     Trap2mf(double a1=0.0, double b1=0.2, double c1=0.3, double d1=0.4, double
        a2=0.6, double b2=0.8, double c2=0.9, double d2=1);
152 };
153
154 // Campanamf #####
155 class Campanamf:public Membership // En MATLAB es gbellmf
156 {
157     public:
158         double eval(double x) const;
159         double evalder(double x) const;
160         int test(void) const; // Modifica los parametros para que sean correctos
161         Campanamf(double a=0.25, double b=2.5, double c=0.5);
162 };
163
164 // Pimf #####
165 class Pimf:public Membership
166 {
167     public:
168         double eval(double x) const;
169         double evalder(double x) const;
170         Pimf(double a=0.05, double b=0.45, double c=0.55, double d=0.95);
171 };
172 // PSigmoidalmf #####
173 class PSigmoidalmf:public Membership // En MATLAB es psigmf
174 {
175     public:
176         double eval(double x) const;
177         double evalder(double x) const;
178         PSigmoidalmf(double a1=11, double c1=0.25, double a2=-11, double c2=0.75);
179 };
180
181 // Smf #####
182 class Smf:public Membership // En MATLAB es smf
183 {
184     public:
185         double eval(double x) const;
186         double evalder(double x) const;
187         Smf(double a=0.05, double b=0.45);
188 };
189
190 // Sigmoidalmf #####
191 class Sigmoidalmf:public Membership // En MATLAB es sigmf
192 {
193     public:
194         double eval(double x) const;
195         double evalder(double x) const;
196         Sigmoidalmf(double a=14, double c=0.25);
197 };
198
199 // Sigmoidal2mf #####
200 class Sigmoidal2mf:public Membership // En MATLAB es dsigmf
201 {
202     public:
203         double eval(double x) const;
204         double evalder(double x) const;
205         Sigmoidal2mf(double a1=11, double c1=0.25, double a2=11, double c2=0.75);
206 };
207
208 // Triangularmf #####

```

```

209 class Triangularmf:public Membership // En MATLAB es trimf
210 {
211     public:
212         double eval(double x) const;
213         double evalder(double x) const;
214         int test(void) const; // Modifica los parametros para que sean correctos
215         Triangularmf(double a=0.0, double b=0.5, double c=1.0);
216 };
217
218 // Zmf #####
219 class Zmf:public Membership // En MATLAB es zmf
220 {
221     public:
222         double eval(double x) const;
223         double evalder(double x) const;
224         Zmf(double a=0.55, double b=0.95);
225 };
226 #endif

```

D.2. Clase Rule

Esta clase permite almacenar una regla borrosa con todos sus parámetros, y contiene los métodos necesarios para su tratamiento. El diagrama UML de esta clase se muestra en la figura 3.17.

```

1  /* #####
2
3      Copyright 2004–2009 Antonio Javier áBarragn ñPia, antonio.barragan@diesia.uhu.
         es, http://www.uhu.es/antonio.barragané
4      Jos Manuel úAndjar áMrquez, andujar@diesia.uhu.es
5      Manuel Emilio úGegndez Árias, e-mail: gegundez@dmate.uhu.es
6
7      Dpto. de Ing. óElectrónica, de Sistemas áInformáticos y áAutomática
8      E.P.S. La áRbida, University of Huelva (Spain)
9
10     For more information, please contact with authors.
11
12  ##### */
13
14  /*
15     This software is free software: you can redistribute it and/or modify
16     it under the terms of the GNU General Public License as published by
17     the Free Software Foundation, either version 3 of the License, or
18     (at your option) any later version.
19
20     This software is distributed in the hope that it will be useful,
21     but WITHOUT ANY WARRANTY; without even the implied warranty of
22     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
23     GNU General Public License for more details.
24
25     You should have received a copy of the GNU General Public License
26     along with this program. If not, see <http://www.gnu.org/licenses/>.
27  */
28
29  #ifndef _REGLA_H_
30  #define _REGLA_H_
31
32  #include <stdio.h>

```

```

33 #include <iostream>
34 #include "membership.h"
35 #include "messages.h"
36
37 #if defined (MATLAB_MEX_FILE)
38     #include "mex.h"
39 #endif
40
41 // Declarations #####
42
43 class Regla
44 {
45     int in; //Numero Entradas de la Regla
46     double *TSK; //Consecuente TSK de la Regla
47     Membership **membership; // Funciones de Membership de la Regla
48     inline void limpiar(void);
49 public:
50     int entradas(void) const;
51     int test(void) const; // Si los parametros de todas las F.P. de la Regla son
        validos retorna 1
52     int escribir_TSK(double valor, int entrada); // Retorna 0 si error
53     double leer_TSK(int entrada) const;
54     int leer_funcion(Membership &P, int entrada) const;
55     Membership *leer_funcion(int entrada);
56     int escribir_funcion(const Membership &P, int entrada);
57     int cambiar_funcion(TYPE_MF tipo, int entrada);
58     void inicializar(int in);
59     Regla &operator=(const Regla &R);
60     Regla();
61     Regla(const Regla &R);
62     Regla(int in);
63     ~Regla();
64 };
65
66 #endif

```

D.3. Clase System

Esta clase permite almacenar un sistema borroso con todos sus parámetros, y contiene los métodos necesarios para su tratamiento. El diagrama UML de esta clase se muestra en la figura 3.18.

```

1  /* #####
2
3      Copyright 2004–2009 Antonio Javier áBarragn ñPia, antonio.barragan@diesia.uhu.
        es, http://www.uhu.es/antonio.barragané
4      Jos Manuel úAndjar áMrquez, andujar@diesia.uhu.es
5      Manuel Emilio úGegndez Árias, e-mail: gegundez@dmate.uhu.es
6
7      Dpto. de Ing. óElectrónica, de Sistemas áInformáticos y áAutomática
8      E.P.S. La árbeda, University of Huelva (Spain)
9
10     For more information, please contact with authors.
11
12 ##### */
13
14 /*
15     This software is free software: you can redistribute it and/or modify

```

```

16     it under the terms of the GNU General Public License as published by
17     the Free Software Foundation, either version 3 of the License, or
18     (at your option) any later version.
19
20     This software is distributed in the hope that it will be useful,
21     but WITHOUT ANY WARRANTY; without even the implied warranty of
22     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
23     GNU General Public License for more details.
24
25     You should have received a copy of the GNU General Public License
26     along with this program. If not, see <http://www.gnu.org/licenses/>.
27 */
28
29 #ifndef _SISTEMA_H_
30 #define _SISTEMA_H_
31
32 #include <stdio.h>
33 #include <iostream>
34 #include "membership.h"
35 #include "rule.h"
36 #include "messages.h"
37
38 #define MIN_DEFAULT_LIMIT -100.0
39 #define MAX_DEFAULT_LIMIT 100.0
40
41 #if defined (MATLAB_MEX_FILE)
42     #include "mex.h"
43 #endif
44
45 // Declarations #####
46
47 class Sistema
48 {
49     int in,out,*N; // Entradas, Salidas y Reglas(por cada salida)
50     double *min_in, *min_out,*max_in, *max_out;
51     Regla **R;
52 public:
53     int reglas(int salida) const;
54     int entradas(void) const;
55     int salidas(void) const;
56     double in_min(int entrada) const;
57     double in_max(int entrada) const;
58     double out_min(int salida) const;
59     double out_max(int salida) const;
60     void in_min(double *limits);
61     void in_max(double *limits);
62     void out_min(double *limits);
63     void out_max(double *limits);
64     int test(void) const; // Si los parametros de todas las F.P. del Sistema son
        validos retorna 1
65     int escribir_Regla(const Regla &R, int r, int salida);
66     int leer_Regla(Regla &R, int r, int salida) const;
67     Regla* leer_Regla(int r, int salida);
68     void inicializar(int in, int out, int *N);
69     Sistema & operator=(const Sistema &S);
70     Sistema();
71     Sistema(const Sistema &S);
72     Sistema(int in, int out, int *N);
73     ~Sistema();
74 };
75
76 #endif

```

D.4. Clase Jacobian

Esta clase permite almacenar y calcular la matriz jacobiana de un sistema de control borroso. Contiene los métodos necesarios para su cálculo y para el tratamiento de la clase. El diagrama UML de esta clase se muestra en la figura 4.3.

```

1  /* #####
2
3      Copyright 2004–2009 Antonio Javier áBarragn ñPia, antonio.barragan@diesia.uhu.
         es, http://www.uhu.es/antonio.barragané
4      Jos Manuel úAndjar áMrquez, andujar@diesia.uhu.es
5      Manuel Emilio úGegndez Árias, e-mail: gegundez@dmate.uhu.es
6
7      Dpto. de Ing. óElectrónica, de Sistemas áInformáticos y áAutomática
8      E.P.S. La áRbida, University of Huelva (Spain)
9
10     For more information, please contact with authors.
11
12  ##### */
13
14  /*
15     This software is free software: you can redistribute it and/or modify
16     it under the terms of the GNU General Public License as published by
17     the Free Software Foundation, either version 3 of the License, or
18     (at your option) any later version.
19
20     This software is distributed in the hope that it will be useful,
21     but WITHOUT ANY WARRANTY; without even the implied warranty of
22     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
23     GNU General Public License for more details.
24
25     You should have received a copy of the GNU General Public License
26     along with this program. If not, see <http://www.gnu.org/licenses/>.
27  */
28
29  #ifndef _JACOBIANO_H_
30  #define _JACOBIANO_H_
31
32  #include <stdio.h>
33  #include <iostream>
34  #include "membership.h"
35  #include "rule.h"
36  #include "system.h"
37  #include "messages.h"
38  #include "ext_math.h"
39  #include "utilities.h"
40
41  // Declarations #####
42
43  class Jacobiano
44  {
45      int n; // orden del Sistema
46      double **J; // Matrix Jacobiana
47  public:
48      int estados() const;
49      double leer_J(int fila, int columna) const;
50      int escribir_J(double valor, int fila, int columna);
51      // Resuelve la Matriz J en el Punto 'x' utilizando el modelo borroso
52      // S para el Sistema y C para el Controlador.
53      int resolver(Sistema S, Sistema C, const double * const x);
54      // Resuelve la Matriz J en el Punto 'x' utilizando el modelo borroso

```

```

55 // S para el Sistema y C para el Controlador mediante aproximacion por
    diferencia finita.
56 int aprox(Sistema S, Sistema C, const double * const x, double h=0.001);
57 // Jacobiano ();
58 Jacobiano(int n);
59 Jacobiano(Sistema S, Sistema C, const double * const x);
60 ~Jacobiano ();
61 };
62
63 #endif

```

D.5. Utilidades

Este archivo define las cabeceras de una serie de funciones de utilidad para el tratamiento de sistemas borrosos. El diagrama UML de esta clase se muestra en la figura 3.19.

```

1 /* #####
2
3 Copyright 2004–2009 Antonio Javier áBarragn ñPia, antonio.barragan@diesia.uhu.
    es, http://www.uhu.es/antonio.barragané
4 Jos Manuel úAndjar áMrquez, andujar@diesia.uhu.es
5 Manuel Emilio úGegndez Árias, e-mail: gegundez@dmate.uhu.es
6
7 Dpto. de Ing. óElectrónica, de Sistemas áInformáticos y áAutomática
8 E.P.S. La áRbida, University of Huelva (Spain)
9
10 For more information, please contact with authors.
11
12 ##### */
13
14 /*
15 This software is free software: you can redistribute it and/or modify
16 it under the terms of the GNU General Public License as published by
17 the Free Software Foundation, either version 3 of the License, or
18 (at your option) any later version.
19
20 This software is distributed in the hope that it will be useful,
21 but WITHOUT ANY WARRANTY; without even the implied warranty of
22 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
23 GNU General Public License for more details.
24
25 You should have received a copy of the GNU General Public License
26 along with this program. If not, see <http://www.gnu.org/licenses/>.
27 */
28
29 #ifndef UTILIDADES_H
30 #define UTILIDADES_H
31
32 #include <stdlib.h>
33 #include <iostream>
34 #include <string>
35 #include "membership.h"
36 #include "rule.h"
37 #include "system.h"
38 #include "messages.h"
39
40 // Declarations #####
41
42 int TXT2Sistema(const char *fichero, Sistema &S);

```



```

43 int Sistema2TXT(Sistema &S, const char *fichero); // Escribe un archivo de Texto
    desde un Sistem
44 int Evaluar(const double * const x, double *Salida, Sistema &S); // Devuelve -1
    si se produce un Desbordamiento y 1 si se realiza el calculo correctamente
45 int Evaluar(const double * const x, double *dX, Sistema &S, Sistema &C); //
    Devuelve 0 si los Modelos no son Compatibles, -1 si se produce un
    Desbordamiento y 1 si se realiza el calculo correctamente.
46 int linealizar(Sistema &S, const double *const x, const double *const u,
    double **A, double **B, double *F=NULL); // Devuelve 0 si se produce un
    error u 1 si el calculo es correcto
47 int linearaprox(Sistema &S, const double *const x, const double *const u,
    double **A, double **B, double h=0.001, double *F=NULL); // Devuelve 0 si
    se produce un error u 1 si el calculo es correcto
48 int imprimirSistema(const char *fichero, const Sistema &S, char *entradas[]=
    NULL, char *salidas[]=NULL, int decimales=100); // Escribe un Sistema en un
    archivo de texto en la forma: "SI x1 es A111 y x2 es ... ENTONCES x1' es ..."
49
50 int numeroDatosAntecedentes(Sistema &S); // Obtiene el numero de datos que posee
    almacenado el modelo borroso S como Antecedentes (Los parametros de las todas
    Funciones de Membership)
51 int numeroDatosAntecedentes(Sistema &S, int out, int rule); // Obtiene el numero
    de datos que posee almacenada la Regla 'rule' para la salida 'out' del modelo
    borroso S como Antecedentes (Los parametros de las todas Funciones de
    Membership)
52 int numeroDatosConsecuentes(Sistema &S); // Obtiene el numero de datos que posee
    almacenado el modelo borroso S en los Consecuentes
53 int numeroDatosConsecuentes(Sistema &S, int out, int rule); // Obtiene el numero
    de datos que posee almacenada la Regla 'rule' para la salida 'out' del modelo
    borroso S en los Consecuentes
54
55 int obtenerAntecedentes(Sistema &S, double * parametros, int num_datos); //
    Obtiene un vector con todos los parametros de los Antecedentes del Sistema.
    Retorna el numero de datos o 0 si error.
56 int obtenerAntecedentes(Sistema &S, double * parametros, int num_datos, int out,
    int rule); // Obtiene un vector con todos los parametros de los Antecedentes
    de la Regla 'rule' para la salida 'out' del Sistema. Retorna el numero de datos
    o 0 si error.
57 int obtenerConsecuentes(Sistema &S, double * parametros, int num_datos); //
    Obtiene un vector con todos los parametros de los Consecuentes del Sistema.
    Retorna el numero de datos o 0 si error.
58 int obtenerConsecuentes(Sistema &S, double * parametros, int num_datos, int out,
    int rule); // Obtiene un vector con todos los parametros de los Consecuentes
    de la Regla 'rule' para la salida 'out' del Sistema. Retorna el numero de datos
    o 0 si error.
59
60 int almacenarAntecedentes(Sistema &S, const double * const parametros, int
    num_datos); // Almacena los parametros dados de los Antecedentes en S (S debe
    estar definido completamente antes de ejecutar la funcion). Retorna 1 si se
    realizo correctamente y 0 si hubo un error.
61 int almacenarAntecedentes(Sistema &S, const double * const parametros, int
    num_datos, int out, int rule); // Almacena los parametros dados de los
    Antecedentes en la Regla 'rule' para la salida 'out' de S (S debe estar
    definido completamente antes de ejecutar la funcion). Retorna 1 si se realizo
    correctamente y 0 si hubo un error.
62 int almacenarConsecuentes(Sistema &S, const double * const parametros, int
    num_datos); // Almacena los parametros dados de los Consecuentes en S (S debe
    estar definido completamente antes de ejecutar la funcion). Retorna 1 si se
    realizo correctamente y 0 si hubo un error.
63 int almacenarConsecuentes(Sistema &S, const double * const parametros, int
    num_datos, int out, int rule); // Almacena los parametros dados de los
    Consecuentes en la Regla 'rule' para la salida 'out' de S (S debe estar

```

```

    definido completamente antes de ejecutar la funcion). Retorna 1 si se realizo
    correctamente y 0 si hubo un error.
64
65 int activacion(Sistema &S, int salida, int regla, double *Punto, double* W); //
    Obtiene los Grados de Activacion de las Funciones de Pertenencia de la Regla
    indicada en un Punto. Retorna 0 si se ejecuto correctamente
66 int activacion(Sistema &S, int salida, int regla, double *Punto, double* W,
    double *dW); // Obtiene los Grados de Activacion de las Funciones de
    Pertenencia de la Regla indicada en un Punto y las derivadas de esta respecto a
    X. Retorna 0 si se ejecuto correctamente
67 int subSistema(Sistema &S, Sistema &ParteS, int nreglas, int *salidas, int *reglas
    ); // Obtiene un subsistema con las reglas especificadas. Retorna 0 si se
    ejecuto correctamente
68
69 #endif

```

D.6. Otros archivos

Esta cabecera permite la inclusión de todas las librerías necesarias para el tratamiento de los sistema borrosos en una sola llamada.

```

1 /* #####
2
3     Copyright 2004–2009 Antonio Javier áBarragn ñPia, antonio.barragan@diesia.uhu.
        es, http://www.uhu.es/antonio.barragané
4     Jos Manuel úAndjar áMrquez, andujar@diesia.uhu.es
5     Manuel Emilio úGegndez Árias, e-mail: gegundez@dmат.uhu.es
6
7     Dpto. de Ing. óElectrñica, de Sistemas áInformticos y áAutomtica
8     E.P.S. La áRbida, University of Huelva (Spain)
9
10    For more information, please contact with authors.
11
12 ##### */
13
14 /*
15     This software is free software: you can redistribute it and/or modify
16     it under the terms of the GNU General Public License as published by
17     the Free Software Foundation, either version 3 of the License, or
18     (at your option) any later version.
19
20     This software is distributed in the hope that it will be useful,
21     but WITHOUT ANY WARRANTY; without even the implied warranty of
22     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
23     GNU General Public License for more details.
24
25     You should have received a copy of the GNU General Public License
26     along with this program. If not, see <http://www.gnu.org/licenses/>.
27 */
28
29 #ifndef _FUZZY_H_
30     #define _FUZZY_H_
31
32     #include "membership.h"
33     #include "rule.h"
34     #include "system.h"
35     #include "ext_math.h"
36     #include "utilities.h"
37     #include "messages.h"

```

38 **#endif**

En el siguiente archivo se definen las funciones creadas para permitir lectura y escritura de modelos borrosos directamente desde MATLAB®. Estas funciones son de gran ayuda para la utilización del resto de funciones en MATLAB® de forma transparente al usuario. El diagrama UML de esta clase se muestra en la figura 3.20.

```

1  /* #####
2
3      Copyright 2004–2009 Antonio Javier áBarragn ñPia, antonio.barragan@diesia.uhu.
         es, http://www.uhu.es/antonio.barragané
4      Jos Manuel úAndjar áMrquez, andujar@diesia.uhu.es
5      Manuel Emilio úGegndez Árias, e-mail: gegundez@dmat.uhu.es
6
7      Dpto. de Ing. óElectrnica, de Sistemas áInformticos y áAutomtica
8      E.P.S. La árvida, University of Huelva (Spain)
9
10     For more information, please contact with authors.
11
12  ##### */
13
14  /*
15     This software is free software: you can redistribute it and/or modify
16     it under the terms of the GNU General Public License as published by
17     the Free Software Foundation, either version 3 of the License, or
18     (at your option) any later version.
19
20     This software is distributed in the hope that it will be useful,
21     but WITHOUT ANY WARRANTY; without even the implied warranty of
22     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
23     GNU General Public License for more details.
24
25     You should have received a copy of the GNU General Public License
26     along with this program. If not, see <http://www.gnu.org/licenses/>.
27  */
28
29  #if defined (MATLAB_MEX_FILE)
30
31  #ifndef _AUX_MATLAB_H_
32  #define _AUX_MATLAB_H_
33
34  #include <stdio.h>
35  #include <iostream>
36
37  #include "mex.h"
38  #include "../actual/system.h"
39  #include "../actual/utilities.h"
40  #include "../actual/messages.h"
41
42  // Declarations #####
43
44  int FIS2System(const mxArray *FIS, Sistema &S); // Read the Fuzzy Model of FIS
         variable
45  int System2FIS(Sistema &S, mxArray *FIS); // Write the Fuzzy Model in FIS variable
46  int readModel(const mxArray *modelo, Sistema &S); // Read a Fuzzy Model (TXT, FIS
         file or FIS variable)
47
48  #endif
49  #endif

```

En esta cabecera de definen las funciones básicas de entrada/salida para las clases anteriores.

```

1  /* #####
2
3      Copyright 2004–2009 Antonio Javier áBarragn ñPia, antonio.barragan@diesia.uhu.
         es, http://www.uhu.es/antonio.barragané
4      Jos Manuel úAndjar áMrquez, andujar@diesia.uhu.es
5      Manuel Emilio úGegndez Árias, e-mail: gegundez@dmate.uhu.es
6
7      Dpto. de Ing. óElectrónica, de Sistemas áInformáticos y áAutomática
8      E.P.S. La áRbida, University of Huelva (Spain)
9
10     For more information, please contact with authors.
11
12  ##### */
13
14  /*
15     This software is free software: you can redistribute it and/or modify
16     it under the terms of the GNU General Public License as published by
17     the Free Software Foundation, either version 3 of the License, or
18     (at your option) any later version.
19
20     This software is distributed in the hope that it will be useful,
21     but WITHOUT ANY WARRANTY; without even the implied warranty of
22     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
23     GNU General Public License for more details.
24
25     You should have received a copy of the GNU General Public License
26     along with this program. If not, see <http://www.gnu.org/licenses/>.
27  */
28
29  #ifndef FUZZY_IO_H
30  #define FUZZY_IO_H
31
32  #ifndef strcmpi
33      #define strcmpi strcasecmp // Realiza el cambio para compilar en Linux
34  #endif
35
36  #include <string.h>
37  #include <stdlib.h>
38  #include <iostream>
39  #include "fuzzy.h"
40  #include "jacobian.h"
41
42  // In-Out Declarations #####
43  #ifndef MATLAB_MEX_FILE
44      std::ostream &operator<<(std::ostream &F, TYPE_MF &T);
45      std::istream &operator>>(std::istream &F, TYPE_MF &T);
46      std::ostream &operator<<(std::ostream &F, Membership &P);
47      std::istream &operator>>(std::istream &F, Membership &P);
48      std::ostream &operator<<(std::ostream &F, Regla &R);
49      std::istream &operator>>(std::istream &F, Regla &R);
50      std::ostream &operator<<(std::ostream &F, Sistema &S);
51      std::istream &operator>>(std::istream &F, Sistema &S);
52      std::ostream &operator<<(std::ostream &F, Jacobiano &J);
53      std::istream &operator>>(std::istream &F, Jacobiano &J);
54  #endif
55
56  #endif

```

El siguiente archivo lista los mensajes que puede generar el software en tiempo de ejecución. Se han separado del código para permitir la fácil traducción del mismo.

```

1  /* #####
2
3      Copyright 2004–2009 Antonio Javier áBarragn ñPia, antonio.barragan@diesia.uhu.
         es, http://www.uhu.es/antonio.barragané
4      Jos Manuel úAndjar áMrquez, andujar@diesia.uhu.es
5      Manuel Emilio úGegndez Árias, e-mail: gegundez@dmate.uhu.es
6
7      Dpto. de Ing. óElectrónica, de Sistemas áInformáticos y áAutomática
8      E.P.S. La árbida, University of Huelva (Spain)
9
10     For more information, please contact with authors.
11
12  ##### */
13
14  /*
15     This software is free software: you can redistribute it and/or modify
16     it under the terms of the GNU General Public License as published by
17     the Free Software Foundation, either version 3 of the License, or
18     (at your option) any later version.
19
20     This software is distributed in the hope that it will be useful,
21     but WITHOUT ANY WARRANTY; without even the implied warranty of
22     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
23     GNU General Public License for more details.
24
25     You should have received a copy of the GNU General Public License
26     along with this program. If not, see <http://www.gnu.org/licenses/>.
27  */
28
29  #ifndef _MESSAGES_H_
30  #define _MESSAGES_H_
31
32  // Para evitar que MATLAB se cierre en Linux con mexErrMsgTxt se redefine
33  #ifndef strcmpi
34      #define ERRORMSG(men) {mexPrintf("??? Error using ==> %s\b\b\b\n%s\n\n",
          __FILE__, men); return;}
35  #else
36      #define ERRORMSG(men) {mexErrMsgTxt(men);}
37  #endif
38
39  // óDefinición de Constantes empleadas en el Programa
40  #define MAX_CHAR_LONG 256 // Longitud Máxima de las Cadenas de Caracteres
41  #define MAX_FILE_NAME 256 // Longitud Máxima de los Nombres de Archivos
42  #define PRECISION 100 // Número de dígitos decimales al escribir en fichero o
        pantalla
43
44  // Error Messages( E_... ) #####
45  // Model
46      const static char *E_Model={"Error reading model."};
47      const static char *E_NoSugeno={"Model 'type' should be 'sugeno'."};
48      const static char *E_NoOutputs={"Error, the model has not any Output."};
49      const static char *E_NoInputs={"Error, the model has not any Input."};
50      const static char *E_NoRules={"Error, some Outputs of the model have not
        any Rule."};
51      const static char *E_No1Conseq={"Only one consequent for rule is allowed
        ."};
52      const static char *E_SameIns={"All models must have the same inputs."};
53      const static char *E_NoCoherent={"The Controller is not coherent with
        the Plant."};

```

```

54     const static char *E_InNoCoherent={"Inputs arguments are not coherent
55         with model inputs."};
56     const static char *E_BadModel={"Undefined or wrong fuzzy model.\nCheck
57         rules, membership functions and theirs parameters"};
58     const static char *E_NoCoherent_BadX={"The Controller is not coherent
59         with the Plant, or the point is not coherent with the closed loop
60         system."};
61     const static char *E_ArgNoValid={"Empty and NaN matrices are not valid
62         inputs arguments."};
63     const static char *E_AccuracyNoNum={"The accuracy should be a number."};
64     const static char *E_Acquire={"Error reading model or incorrect initial
65         model."};
66     const static char *E_Store={"Error storing data. Wrong vector size or
67         any data not valid for membership function, i.e. a zero width in
68         Gaussmf function."};
69 // FIS
70     const static char *E_NoFIS={"Fuzzy Model must be a FIS structure."};
71     const static char *E_CreateFIS={"Error at create FIS."};
72     const static char *E_NoProd={"Implication method must be 'prod'."};
73     const static char *E_FISOut={"Error writing FIS."};
74     const static char *E_RulesExceded={"Rule limit exceeded."};
75     const static char *E_InputExceded={"Input limit exceeded."};
76     const static char *E_OutputExceded={"Output limit exceeded."};
77 // Membership Functions
78     const static char *E_MFType={"Error reading membership function type:"};
79     const static char *E_BadMF={"Membership function was not identified."};
80     const static char *E_MFTypeDef={"Error in membership function type."};
81     const static char *E_ParamMF={"Error reading membership function
82         parameters."};
83     const static char *E_NoValidFP={"Any parameter of membership function is
84         not valid."};
85     const static char *E_MFDef={"Error in membership function definition."};
86 // Consequent
87     const static char *E_Conseq={"It is not posible read consequent type in
88         rule:"};
89     const static char *E_ParamConseq={"Error reading TSK's consequent
90         parameters."};
91     const static char *E_BadConseq={"Not valid cosequent."};
92 // Files
93     const static char *E_FileInput={"Input file error."};
94     const static char *E_FileOutput={"Output file error."};
95 // Others
96     const static char *E_ArrayExceded={"Array limit exceded."};
97     const static char *E_NoSumProm={"Error, 'defuzzMethod' should be 'wtaver
98         '."};
99     const static char *E_NumberParam={"Number of parameters incorrect."};
100    const static char *E_NumberArg={"Error in number of arguments."};
101    const static char *E_NumberArgIn={"Error in inputs arguments."};
102    const static char *E_NumberArgOut={"Error in outputs arguments."};
103    const static char *E_Column={"The state vector must be a column vector."
104    };
105    const static char *E_File={"Rules and Outputs should be a file vector."
106    };
107    const static char *E_PointCoherent={"The point/s should be coherent with
108        fuzzy models."};
109    const static char *E_InNames={"Error reading names of inputs."};
110    const static char *E_OutNames={"Error reading names of outputs."};
111    const static char *E_R_No_Scalar={"The number of rule must be a scalar."
112    };
113    const static char *E_Out_No_Scalar={"The number of output must be a
114        scalar."};

```

```

97     const static char *E_Epoch_No_Scalar={"The number of epoch must be a
          scalar."};
98     const static char *E_Alfa_No_Scalar={"Alfa must be a scalar."};
99     const static char *E_Error_No_Scalar={"The error must be a scalar."};
100    const static char *E_ModelError={"Modeling error."};
101    const static char *E_H_GE_0={"'h' must be >= 0."};
102
103
104 // Mensajes de Usuario ( U_... ) #####
105 // FIS
106     const static char *U_FISName={"Unnamed"};
107     const static char *U_FISInput={"Input"};
108     const static char *U_FISOutput={"Output"};
109     const static char *U_RuleAbbreviation={"R"};
110     const static char *U_InputAbbreviation={"-In"};
111     const static char *U_OutputAbbreviation={"-Out"};
112     const static char *U_If={"IF"};
113     const static char *U_Is={"is"};
114     const static char *U_And={"and"};
115     const static char *U_OR={"or"};
116     const static char *U_Then={"THEN"};
117 // Otros
118     const static char *U_SeeManual={"See also Fuzzy Logic Toolbox Manual."};
119     const static char *U_SeeHelp={"see the help."};
120     const static char *U_CheckModel={"Check model's definition."};
121     const static char *U_MathException={"Warning. Math exception generated."};
122     const static char *U_Overflow={"Happened an overflow."};
123     const static char *U_FewData={"There are few data for modeling the
          system."};
124
125 // Otros Mensajes ( O_... ) #####
126 // Entradas
127     const static char *O_Input={"Input:"};
128     const static char *O_OffIn={"of in"};
129 // Salidas
130     const static char *O_OfOut={"of out"};
131 // Reglas
132     const static char *O_RuleWeigh={"The rule's weigh:"};
133     const static char *O_RuleConnection={"The rule's connection:"};
134     const static char *O_AntecedentSize={"The size of rule's antecedent"};
135     const static char *O_ConsequentSize={"The size of rule's consequent"};
136     const static char *O_Rule={"Rule:"};
137 //Otros
138     const static char *O_DifferentOf={"is different of"};
139     const static char *O_IsNotCorrect={"is not correct."};
140     const static char *O_MF={"Membership Function:"};
141     const static char *O_OfPlant={"of the Plant"};
142     const static char *O_OfController={"of Controller"};
143
144 #endif

```

En el siguiente archivo se definen funciones matemáticas necesarias para el software y que no están definidas en C/C++ de forma estándar.

```

1 /* #####
2
3     Copyright 2004–2009 Antonio Javier áBarragn ñPia, antonio.barragan@diesia.uhu.
         es, http://www.uhu.es/antonio.barragané
4     Jos Manuel úAndjar áMrquez, andujar@diesia.uhu.es
5     Manuel Emilio úGegndez Árias, e-mail: gegundez@dmate.uhu.es
6

```

```

7      Dpto. de Ing. óElectrónica , de Sistemas áInformáticos y áAutomática
8      E.P.S. La áRbida, University of Huelva (Spain)
9
10     For more information , please contact with authors .
11
12 ##### */
13
14 /*
15     This software is free software: you can redistribute it and/or modify
16     it under the terms of the GNU General Public License as published by
17     the Free Software Foundation, either version 3 of the License , or
18     (at your option) any later version .
19
20     This software is distributed in the hope that it will be useful ,
21     but WITHOUT ANY WARRANTY; without even the implied warranty of
22     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
23     GNU General Public License for more details .
24
25     You should have received a copy of the GNU General Public License
26     along with this program. If not , see <http://www.gnu.org/licenses/>.
27 */
28
29 #ifndef _EXT_MATH_H_
30 #define _EXT_MATH_H_
31
32 #include <math.h>
33 #include "messages.h"
34
35 #ifdef MATLAB_MEX_FILE
36     #include <iostream>
37     #include "mex.h"
38 #endif
39
40 # define M_EPS 1e-16
41
42 inline int sign(double x)
43 {
44     if (x>0)
45         return 1;
46     else if (x<0)
47         return -1;
48     else
49         return 0;
50 }
51
52 #endif

```


Índice de figuras

2.1. Conjunto ordinario.	20
2.2. Conjunto borroso.	20
2.3. Ejemplos de funciones de pertenencia.	22
2.4. Convexidad en conjuntos borrosos.	23
2.5. Función de pertenencia triangular.	25
2.6. Función de pertenencia trapezoidal.	26
2.7. Función de pertenencia gaussiana.	27
2.8. Funciones de pertenencia gaussiana doble.	28
2.9. Funciones de pertenencia campana.	29
2.10. Función de pertenencia S	30
2.11. Función de pertenencia Z	31
2.12. Función de pertenencia Π	32
2.13. Funciones de pertenencia sigmoidales.	33
2.14. Funciones de pertenencia diferencia de sigmoidales.	34
2.15. Funciones de pertenencia producto de sigmoidales.	34
2.16. Función de pertenencia Singleton.	35
2.17. Intersección entre conjuntos borroso.	36
2.18. Unión de conjuntos borrosos.	37
2.19. Operación negación o complemento a uno.	38
2.20. T -normas producto y producto acotado.	40
2.21. S -normas suma-producto y suma acotada.	42
2.22. Operadores de negación de Yager y Sugeno.	43
2.23. α -corte de un conjunto borroso para $\alpha = 0,5$	44
2.24. Combinación convexa de dos conjuntos borrosos.	45
2.25. Variable lingüística <i>edad</i>	47
2.26. Concentración y dilatación de una variable lingüística.	47
2.27. Sistema de inferencia borroso.	50
2.28. Ejemplo de inferencia.	52
2.29. Ejemplo de desborrosificación.	53

2.30. Tipos de modelos borrosos	54
2.31. Diagrama de bloques de un modelo aditivo estándar (SAM).	57
2.32. Sistema borroso de Takagi–Sugeno–Kang (TSK).	59
3.1. Sistema de control.	66
3.2. Algoritmo para el cálculo del modelo borroso de la planta.	69
3.3. Algoritmo para el cálculo del controlador borroso.	73
3.4. Sistema de control en lazo cerrado.	75
3.5. Algoritmo para el cálculo del sistema de control borroso en lazo cerrado.	77
3.6. Proceso de identificación.	80
3.7. Ejemplo de particionamiento uniforme.	81
3.8. Ejemplo de particionamiento por <i>clustering</i>	82
3.9. Red neuronal equivalente a un modelo borroso.	84
3.10. Red neuronal equivalente al modelo borroso de la planta.	85
3.11. Algoritmo de adaptación neuroborrosa.	87
3.12. Grafo para el ajuste de modelos TSK mediante OCH.	88
3.13. Algoritmo de activación de las hormigas.	89
3.14. OCH–neuroborroso.	90
3.15. Diagrama UML de la estructura de clases.	92
3.16. Clase Pertenencia	92
3.17. Clase Regla	93
3.18. Clase Sistema	94
3.19. Paquete Utilidades	95
3.20. Paquete Utilidades MATLAB	96
4.1. Algoritmo numérico de minimización.	126
4.2. Algoritmo de linealización de un modelo borroso.	127
4.3. Clase Jacobiana.	128
4.4. Relación entre las clases Jacobiana y Sistema.	128
4.5. Algoritmo para el cálculo de la matriz Jacobiana de un modelo borroso.	129
5.1. Estabilidad.	134
5.2. Función de Lyapunov.	136
5.3. Sistema de inferencia borroso.	139
5.4. Región de atracción, caso 1.	146
5.5. Región de atracción, caso 2.	146
5.6. Algoritmo de inicialización.	149
5.7. Función objetivo.	151
5.8. Algoritmo híbrido de minimización.	151
6.1. Grúa porta contenedores.	156
6.2. Representación esquemática de una grúa porta contenedores.	157
6.3. Funciones de pertenencia para la variable <i>posición</i> , <i>d</i>	158
6.4. Funciones de pertenencia para la variable <i>longitud</i> del cable, <i>l</i>	158

6.5. Funciones de pertenencia para la variable <i>ángulo</i> , ϕ	159
6.6. Funciones de pertenencia para las variables $\dot{d}$ y $\dot{l}$	159
6.7. Funciones de pertenencia para T_1	160
6.8. Funciones de pertenencia para T_2	161
6.9. Evolución de la variable de estado $x_1(t)$	165
6.10. Evolución de la variable de estado $x_2(t)$	166
6.11. Evolución de la variable de estado $x_3(t)$	166
6.12. Evolución de la variable de estado $x_4(t)$	167
6.13. Evolución de la variable de estado $x_5(t)$	167
6.14. Evolución de la variable de estado $x_6(t)$	168
6.15. Funciones de pertenencia para la entrada $x_1(t)$ en la inferencia de $\dot{x}_1(t)$. .	170
6.16. Funciones de pertenencia para la entrada $x_2(t)$ en la inferencia de $\dot{x}_1(t)$. .	171
6.17. Funciones de pertenencia para la entrada $u_1(t)$ en la inferencia de $\dot{x}_1(t)$. .	171
6.18. Funciones de pertenencia para la entrada $u_2(t)$ en la inferencia de $\dot{x}_1(t)$. .	172
6.19. Funciones de pertenencia para la entrada $x_1(t)$ en la inferencia de $\dot{x}_2(t)$. .	172
6.20. Funciones de pertenencia para la entrada $x_2(t)$ en la inferencia de $\dot{x}_2(t)$. .	173
6.21. Funciones de pertenencia para la entrada $u_1(t)$ en la inferencia de $\dot{x}_2(t)$. .	173
6.22. Funciones de pertenencia para la entrada $u_2(t)$ en la inferencia de $\dot{x}_2(t)$. .	174
6.23. Validación del modelo borroso para $\dot{x}_1$	175
6.24. Validación del modelo borroso para $\dot{x}_2$	175
6.25. Error de modelado durante la validación de $\dot{x}_1$	176
6.26. Error de modelado durante la validación de $\dot{x}_2$	176
6.27. Simulación para distintas condiciones iniciales.	181
6.28. Retrato de fase temporal.	182
6.29. Simulación para distintas condiciones iniciales.	186
6.30. Retrato de fase temporal.	186
6.31. Péndulo.	189
6.32. Funciones de pertenencia para la entrada $x_1(t)$ en la inferencia de $\dot{x}_2(t)$. .	191
6.33. Funciones de pertenencia para la entrada $x_2(t)$ en la inferencia de $\dot{x}_2(t)$. .	191
6.34. Funciones de pertenencia para la entrada $u(t)$ en la inferencia de $\dot{x}_2(t)$. .	192
6.35. Validación del modelo borroso para $\dot{x}_2$	193
6.36. Error de modelado durante la validación de $\dot{x}_2$	193
6.37. Simulación para distintas condiciones iniciales.	199
6.38. Retrato de fase temporal.	199
6.39. Retrato de fase.	200
6.40. Balancín.	203
6.41. Funciones de pertenencia para la entrada $x_1(t)$ en la inferencia de $\dot{x}_1(t)$. .	204
6.42. Funciones de pertenencia para la entrada $x_2(t)$ en la inferencia de $\dot{x}_1(t)$. .	204
6.43. Funciones de pertenencia para la entrada $x_3(t)$ en la inferencia de $\dot{x}_1(t)$. .	205
6.44. Funciones de pertenencia para la entrada $x_4(t)$ en la inferencia de $\dot{x}_1(t)$. .	205
6.45. Funciones de pertenencia para la entrada $u(t)$ en la inferencia de $\dot{x}_1(t)$. .	206
6.46. Funciones de pertenencia para la entrada $x_1(t)$ en la inferencia de $\dot{x}_2(t)$. .	206
6.47. Funciones de pertenencia para la entrada $x_2(t)$ en la inferencia de $\dot{x}_2(t)$. .	207

6.48. Funciones de pertenencia para la entrada $x_3(t)$ en la inferencia de $\dot{x}_2(t)$. . .	207
6.49. Funciones de pertenencia para la entrada $x_4(t)$ en la inferencia de $\dot{x}_2(t)$. . .	208
6.50. Funciones de pertenencia para la entrada $u(t)$ en la inferencia de $\dot{x}_2(t)$. . .	208
6.51. Funciones de pertenencia para la entrada $x_1(t)$ en la inferencia de $\dot{x}_3(t)$. . .	209
6.52. Funciones de pertenencia para la entrada $x_2(t)$ en la inferencia de $\dot{x}_3(t)$. . .	209
6.53. Funciones de pertenencia para la entrada $x_3(t)$ en la inferencia de $\dot{x}_3(t)$. . .	210
6.54. Funciones de pertenencia para la entrada $x_4(t)$ en la inferencia de $\dot{x}_3(t)$. . .	210
6.55. Funciones de pertenencia para la entrada $u(t)$ en la inferencia de $\dot{x}_3(t)$. . .	211
6.56. Funciones de pertenencia para la entrada $x_1(t)$ en la inferencia de $\dot{x}_4(t)$. . .	211
6.57. Funciones de pertenencia para la entrada $x_2(t)$ en la inferencia de $\dot{x}_4(t)$. . .	212
6.58. Funciones de pertenencia para la entrada $x_3(t)$ en la inferencia de $\dot{x}_4(t)$. . .	213
6.59. Funciones de pertenencia para la entrada $x_4(t)$ en la inferencia de $\dot{x}_4(t)$. . .	213
6.60. Funciones de pertenencia para la entrada $u(t)$ en la inferencia de $\dot{x}_4(t)$. . .	214
6.61. Validación del modelo borroso para $\dot{x}_1$	215
6.62. Error de modelado durante la validación de $\dot{x}_1$	216
6.63. Validación del modelo borroso para $\dot{x}_2$	216
6.64. Error de modelado durante la validación de $\dot{x}_2$	217
6.65. Validación del modelo borroso para $\dot{x}_3$	217
6.66. Error de modelado durante la validación de $\dot{x}_3$	218
6.67. Validación del modelo borroso para $\dot{x}_4$	218
6.68. Error de modelado durante la validación de $\dot{x}_4$	219
6.69. Simulación para distintas condiciones iniciales $\dot{x}_1$ y $\dot{x}_2$	227
6.70. Simulación para distintas condiciones iniciales $\dot{x}_3$ y $\dot{x}_4$	227
6.71. Retrato de fase temporal $\dot{x}_1$ y $\dot{x}_2$	228
6.72. Retrato de fase temporal $\dot{x}_3$ y $\dot{x}_4$	228

Índice de tablas

2.1. T -normas principales.	40
2.2. S -normas principales.	41
2.3. Operadores de Negación.	43
2.4. Operadores de implicación.	49
6.3. Resultados de la búsqueda de estados de equilibrio. Ejecución 1.	177
6.4. Resultados de la búsqueda de estados de equilibrio. Ejecución 2.	177
6.5. Resultados de la búsqueda de estados de equilibrio. Ejecución 3.	178
6.6. Resultados de la búsqueda de estados de equilibrio. Ejecución 4.	178
6.7. Resultados de la búsqueda de estados de equilibrio. Ejecución 5.	178
6.8. Búsqueda de los estados de equilibrio de lazo cerrado. Ejecución 1.	182
6.9. Búsqueda de los estados de equilibrio de lazo cerrado. Ejecución 2.	183
6.10. Búsqueda de los estados de equilibrio de lazo cerrado. Ejecución 3.	183
6.11. Búsqueda de los estados de equilibrio de lazo cerrado. Ejecución 4.	183
6.12. Búsqueda de los estados de equilibrio de lazo cerrado. Ejecución 5.	184
6.13. Búsqueda de los estados de equilibrio de lazo cerrado. Ejecución 1.	187
6.14. Búsqueda de los estados de equilibrio de lazo cerrado. Ejecución 2.	187
6.15. Búsqueda de los estados de equilibrio de lazo cerrado. Ejecución 3.	188
6.16. Búsqueda de los estados de equilibrio de lazo cerrado. Ejecución 4.	188
6.17. Búsqueda de los estados de equilibrio de lazo cerrado. Ejecución 5.	188
6.18. Resultados de la búsqueda de estados de equilibrio. Ejecución 1.	194
6.19. Resultados de la búsqueda de estados de equilibrio. Ejecución 2.	194
6.20. Resultados de la búsqueda de estados de equilibrio. Ejecución 3.	194
6.21. Resultados de la búsqueda de estados de equilibrio. Ejecución 4.	195
6.22. Resultados de la búsqueda de estados de equilibrio. Ejecución 5.	195
6.23. Resultados de la búsqueda de estados de equilibrio. Ejecución 6.	195
6.24. Búsqueda de los estados de equilibrio de lazo cerrado. Ejecución 1.	200
6.25. Búsqueda de los estados de equilibrio de lazo cerrado. Ejecución 2.	201
6.26. Búsqueda de los estados de equilibrio de lazo cerrado. Ejecución 3.	201

6.27. Búsqueda de los estados de equilibrio de lazo cerrado. Ejecución 4.	201
6.28. Búsqueda de los estados de equilibrio de lazo cerrado. Ejecución 5.	202
6.29. Resultados de la búsqueda de estados de equilibrio. Ejecución 1.	220
6.30. Resultados de la búsqueda de estados de equilibrio. Ejecución 2.	220
6.31. Resultados de la búsqueda de estados de equilibrio. Ejecución 3.	221
6.32. Resultados de la búsqueda de estados de equilibrio. Ejecución 4.	221
6.33. Resultados de la búsqueda de estados de equilibrio. Ejecución 5.	222
6.34. Búsqueda de los estados de equilibrio de lazo cerrado. Ejecución 1.	229
6.35. Búsqueda de los estados de equilibrio de lazo cerrado. Ejecución 2.	229
6.36. Búsqueda de los estados de equilibrio de lazo cerrado. Ejecución 3.	230
6.37. Búsqueda de los estados de equilibrio de lazo cerrado. Ejecución 4.	230
6.38. Búsqueda de los estados de equilibrio de lazo cerrado. Ejecución 5.	231
A.1. Leyenda de índices y variables.	242

Índice alfabético

- Agregación, 51, 67, 72, 112
 α -corte, 42–44
Algoritmo, 3, 5, 69, 73, 76, 77, 79–83, 86–91, 94–96, 123, 125–127, 129, 141, 147–151, 170, 177, 179, 180, 184, 190, 192, 196–198, 212, 215, 219, 222
 genético, 151, 152
 numérico, 123–125, 151, 152, 181, 187, 200, 229
Algoritmo numérico, 4
Antecedente, 48, 49, 51, 55, 58, 67, 72, 78, 79, 83, 86, 87, 93, 146–148, 170, 180, 190, 203, 204, 234, 239–242
Aproximador universal, 55, 56, 59, 66, 79, 83, 100, 140, 233
Autovalor, 100, 111, 164, 179, 196–198, 222, 226, 249

Base de conocimiento, 48, 49, 52, 55, 59
Base de datos, 52, 168
Base de reglas, 52, 55, 58, 140, 160, 165
Bifurcación, 100
Borrosificador, 49–50, 55, 59, 139

Calidad de pertenencia, *véase* grado de pertenencia
Caos, 100
Capa de entrada, 83
Capa de salida, 83
Capa oculta, 83

Cholesky, 150, 249–250
Ciclo límite, 100
Clustering, 3, 83, 170, 190, 203
Clustering borroso, 15
Coeficiente variable, 68, 70, 72, 74, 76, 102, 115, 239, 240
Combinación convexa, 45–46
Complemento, *véase* Negación borrosa
Concentración, 44, 46
Condición de frontera, 39, 41, 42
Condición de límite, 24, 27–29
Conjunto borroso, 2, 3, 13, 19–24, 33–46, 48–51, 55–58, 67, 72, 78, 139, 157, 163, 239–242
 convexo, 22–24, 36
 disjunto, 36
 frontera (de un), 24
 normal, 22, 24
 normalizado, 22, 44, 45
 nulo, 22
Conjunto borroso escalar, *véase* Función de pertenencia singleton
Conjunto clásico, 19–21, 39
Conjunto ordinario, *véase* Conjunto clásico
Conjunto universal, 39
Conorma triangular, *véase* S -norma
Consecuente, 48, 49, 55, 57, 59, 67, 68, 70, 72, 74, 76, 78, 79, 83, 86, 87, 103,

- 113, 115, 133, 146–148, 150, 180, 190, 198, 239–242
- afín, 58, 67, 70, 72, 74, 76, 133, 239, 240
- lineal, 58, 86, 93, 239, 240
- TSK, *véase* Consecuente lineal
- Control borroso, 15–18, 24, 55, 138, 139, 141
- Control de procesos, 51
- Control heurístico, 78, 138–140, 155–168
- Controlador, 3, 19, 65, 71, 72, 74, 76, 110–112, 115, 117, 122, 133, 137, 139–142, 148, 152, 180, 184, 233, 239, 242
- Controlador borroso, 2, 19, 49, 65, 73, 76, 115, 118, 132, 133, 138–140, 142, 147, 148, 150, 155, 156, 163, 165, 168, 169, 180, 181, 184, 185, 189, 197, 198, 222, 226, 236, 243
- Desborrosificador, 49, 53–55, 59, 139, 163
- centroide, 53–54
- centroide indexado, 54
- máximo, 54
- Descripción interna, *véase* Modelo de estado
- Dilatación, 44
- Diseño, 5, 126, 139, 140, 142, 157–163, 168–170, 189, 233
- algoritmo, 147, 148
- controlador, 1, 4, 76, 100, 101, 110, 122, 131, 133, 136, 140–142, 146–148, 165, 168, 180, 185, 189, 202, 233, 234, 236
- fase de, 147, 150–152, 174, 180, 184, 185, 189, 197, 222, 224
- software, 91, 126
- Distribución de posibilidad, 46
- Ecuación de estado, 64–66, 69, 111, 112
- Ecuación de salida, 64, 66
- Entrada de control, *véase* Señal de control
- Entrada/Salida, 3, 97, 100, 133, 140, 145, 168–170, 179, 190, 202, 203, 219, 233
- Error cuadrático, 86
- Error cuadrático medio, 90, 174, 179, 190, 192, 215
- Espacio de estado, 134–137, 142, 143, 146, 148, 180, 184, 185
- Estabilidad, 1, 3, 4, 65, 100, 111, 122, 131, 133, 136, 138, 140–142, 169, 180, 181, 196, 198, 222, 226
- asintótica, 7, 8, 134, 136, 137, 140–142, 147, 150, 151, 180, 189, 196, 197, 202, 226, 236
- asintótica global, 134–137
- en el sentido de Lyapunov, 133–134, 136
- Estado, 63, 65, 69, 73, 76
- Estado de equilibrio, 3, 65, 100–102, 111, 122–125, 133–137, 141, 142, 145, 147, 150, 152, 164, 168, 169, 177, 179–181, 184, 185, 187, 189, 192, 196–198, 200, 202, 215, 219, 222, 226, 229
- Etiqueta lingüística, 30, 31, 46, 48, 52
- Extensión del vector de estado, 2, *véase* Vector de estado extendido
- Factor de aprendizaje, 86
- Función de coste, 86, 88–90
- Función de Lipschitz, 150
- Función de pertenencia, 2, 3, 20–33, 44, 46, 50–52, 56, 57, 67–69, 72, 73, 79, 80, 83, 87, 91, 93, 119, 139, 141, 142, 146–148, 157, 170, 184, 190, 203, 233, 234, 241–243
- campana, 29, 246–247
- derivada de, 243–248
- gaussiana, 26–27, 81, 86, 157, 174, 190, 212, 245
- gaussiana doble, 27–28, 245–246
- Π , 31–32, 248
- S , 29–30, 247
- sigmoidal, 32–33, 248
- sigmoidal (diferencia), 33, 248
- sigmoidal (producto), 33, 248
- singleton, 33, 48, 50, 51, 56
- trapezoidal, 25–26, 244–245
- triangular, 24–25, 190, 243–244

- Z, 30–31, 247
- Fuzzy logic, *véase* lógica borrosa
- Gradiente, 123–125
- Gradiente descendente, 80, 86, 87, 89, 90
- Grado de activación, 51, 57, 58, 67, 72, 102–104, 106, 107, 109, 113–115, 117, 119, 122, 147, 148, 163, 241–243
- Grado de pertenencia, 14, 20–23, 27, 81, 147, 148
- Heurística, 18, 52, 78, 88, 138–140, 156
- Idempotencia, 37, 42
- Identidad, 37
cero, 41
unitaria, 39
- Identificación, *véase* Modelado
- Igualdad borrosa, 33–35
- Inclusión borrosa, 35
- Inestabilidad, 135, 179, 196, 222
- Inferencia borrosa, 2, 16, 49–52, 67, 139, 163, 170, 190, 203
- Inicialización, 93, 147–148, 180, 184, 198, 222
- Intersección borrosa, 2, 35–39, 41, 45, 51, 56, *véase también* T-norma
- Involución, 38
- Krasovskii, 4, 7, 112, 137–138, 142, 180, 181, 184, 189
- Lógica borrosa, 1–4, 13–20, 24, 33, 39, 46, 49, 76, 83, 100, 132, 133, 138, 156, 231
- Lógica clásica, 14, 20, 33, 39
- Line-Search, 123
- Linealización, 3, 101–103, 108, 110, 111, 122, 126, 131, 141, 147, 179, 196, 197, 222
- LMI, 133
- Lyapunov, 4, 7, 111, 133–138, 142, 143, 145, 147, 150, 180, 184, 189, 198, 226
- Margen de tolerancia, 87
- MATLAB®, 7, 9, 10, 91, 93, 95–97, 126, 141, 237, 261
- Matriz hessiana, 123–125
- Matriz jacobiana, 3, 4, 110–112, 114, 116, 120, 121, 124–126, 141, 142, 146, 177, 179, 181, 185, 187, 192, 198, 219, 226, 240
- Método de Newton, 123
- Método numérico, *véase también* Algoritmo numérico
- Mínimos cuadrados, 124
- Modelado, 3, 65, 79–91, 100, 140, 141, 168, 169, 174, 177, 190, 215, 233, 236
borroso, 2, 61, 81, 88, 170
de la planta, 168, 233
mediante colonias de hormigas, 3, 88–90, 236
neuroborroso, 3, 83–87
- Modelo, 62, 83, 138, 141, 163, 179, 192, 196, 197, 203, *véase también* Sistema
de la planta, 19, 110, 140, 170
de lazo cerrado, 65, 110, 138, 139, 163, 165
físico, 62
matemático, 62, 100, 131, 133, 140, 147, 163, 168, 233, 235
mental, 62
verbal, 62
- Modelo borroso, 3, 4, 19, 49, 62, 66, 67, 69, 72, 74, 79, 83, 84, 90, 91, 93, 94, 96, 100–102, 140, 141, 145, 147, 168–170, 174, 179, 189, 190, 192, 196, 197, 203, 212, 215, 219, 222, 233, 234, 236
de la planta, 2, 3, 65–71, 78, 85, 95, 102, 104, 125, 140, 177, 180, 185, 192, 198, 215, 226, 234, 236, 239–242
de lazo cerrado, 2, 3, 75–76, 95, 125, 146, 147, 187, 198, 226
del controlador, 71–75, 78, 95, 234, 240, 241
- Mamdani, 2, 54–57, 59, 62, 66, 78–79, 156

- puro, 54
- SAM, 56–57
- TSK, 1–3, 57–60, 62, 66–76, 79, 88, 91, 112, 140, 233
- Tsukamoto, 57
- Modelo de estado, 2, 63, 69, 100, 140, 141, 163, 169, 189, 190, 203, 236
 - de tiempo continuo, 64–76, 78
 - de tiempo discreto, 64–66, 70–71, 74–76, 78–79
- Modelo matemático equivalente, 2, 3, 79
 - de la planta, 3, 65, 69, 71, 75, 76, 122
 - de lazo cerrado, 3, 65, 76, 112, 123
 - del controlador, 3, 65, 73, 75, 76, 118
 - linealizado, 102, 108
- Modificador lingüístico, 46
- Momento de inercia, 156, 202
- Momentos, 87
- Monotonicidad, 39, 41
- Motor de inferencia, 49–51, 54, 55, 59, 112
- Negación borrosa, 2, 36–37, 42
- Norma, 152
- Norma triangular, *véase* *T*-norma
- Núcleo, 23, 25–27, 33
- Número borroso, 24–26, 32
- Número cuasi-borroso, 24, 27–29
- OCH, *véase* Modelado mediante colonias de hormigas
- Operador de implicación, 48, 51, 56
- Orden, 63, 66, 72, 147, 150, 240
- Origen, 135, 137, 142, 143, 145, 146, 150, 152, 180, 181, 184, 185, 187, 197, 198, 200, 219, 222, 226, 229
- Parámetros adaptables, 67, 72, 94, 141, 142, 146, 147, 150, 180, 198, 241, 242
- Particionado
 - en línea, 81
 - experto, 3, 80
 - por clustering, 81
 - uniforme, 3, 80
- Peso, 24
- Planta, 2–4, 19, 65, 67, 69, 75, 76, 78, 79, 103, 106, 107, 109, 112–114, 117, 119, 122, 123, 131, 133, 138–141, 145, 147, 155–157, 160, 163, 165, 168, 169, 177, 181, 184, 185, 189, 190, 197, 202, 203, 215, 219, 222, 233, 236, 239–243
- Principio de Incompatibilidad, 46
- Probabilidad de pertenencia, 21
- Producto algebraico, 44
- Promedio ponderado, 58, 59, 67, 72, 112
- Propiedad
 - asociativa, 38, 39, 41
 - conmutativa, 38, 39, 41
 - distributiva, 38, 42
 - involutiva, 42
- Punto borroso, *véase* función de pertenencia singleton
- Punto singular, *véase* Estado de equilibrio
- Razonamiento aproximado, 49
- Red neuronal, 60, 83, 84, 86
- Región de atracción, 145
- Región de confianza, 123
- Regla borrosa, 1–3, 15, 48–49, 55, 57, 59, 65–69, 72–74, 76, 78, 79, 82, 83, 86, 91, 93, 94, 103, 113, 115, 119, 133, 138, 139, 141, 147, 148, 163, 169, 170, 180, 184, 190, 197, 203, 204, 212, 222, 233, 234, 239–243
- regla borrosa, 2
- Regla lingüística, *véase* Regla borrosa
- Retrato de fase, 182, 185, 186, 199, 200, 228
- S-norma, 39, 41–42, 45, 48, 51, 56
- Señal de control, 65, 67, 69, 72, 78, 102, 105, 107, 109, 110, 117, 118, 139, 163, 180, 189, 190, 203, 239–242
- Señal de error, 86
- Sistema
 - borroso, 19, 48–51, 55, 59, 62, 94, 123, 125, 132, 233

- de control, 1, 19, 49, 50, 66, 100, 101, 110, 112, 117, 122, 131, 132, 141, 147, 163, 169, 181, 189, 202, 222
- de control borroso, 75, 76, 91, 112, 114, 116, 118–121, 126, 132, 133, 142, 147, 169, 181, 187, 198, 226, 243
- de inferencia, 49–54
- lineal, 100, 131
- neuroborroso, 83
- no lineal, 1–4, 6–8, 63, 64, 66, 71, 74, 76, 79, 100, 101, 110, 123, 131, 133, 136, 169, 233
- Sobre-entrenamiento, 82
- Software, 3, 91–96, 100, 126, 141
- Soporte, 23–33
- Standard Fuzzy Partition, 133
- Suma, 45
- Suma algebraica, 44–45
- Suma ponderada, 58, 59
- T-conorma, *véase* S-norma
- T-norma, 39–42, 45, 48, 51, 56, 59
- Tecnología fuzzy, 15
- Tiempo normalizado, 71
- Transitividad, 38
- UML, 3, 91, 126
- Unión borrosa, 2, 36, 39, 51, *véase también* S-norma
- Universo de discurso, 8, 20–23, 45, 46, 48, 50, 53, 55, 58, 67, 68, 72, 76, 78–80, 93, 139, 145, 169, 181, 185, 187, 190, 192, 198, 200, 202, 203, 215, 219, 226, 229, 239–242
- Validación, 3, 90–91, 141, 170, 174, 179, 190, 192, 203, 215
- Validación cruzada, 90
- Variable de control, 65, 68, 71, 157, 169, 239–242
- Variable de estado, 3, 8, 63–65, 67, 68, 71, 72, 78, 105, 110–112, 114, 115, 117, 120, 133, 141, 145, 169, 181, 185, 187, 190, 192, 198, 200, 202, 203, 215, 226, 229, 239–242
- Variable lingüística, 2, 46–48, 52, 139, 157, 160
- Vector de control, 2, 3, 64–66, 68, 69, 71, 73, 74, 76, 103, 106, 109, 110, 117, 118, 122, 141, 169
- Vector de estado, 2, 3, 63–69, 71–74, 76, 103, 109, 110, 113, 115, 117, 141, 143, 164, 169, 189, 202
- Vector de estado extendido, 69, 71, 73, 76, 77, 102, 127, 240, 241
- Vector de salida, 64

Bibliografía

- [1] Lofti A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.
- [2] Lluís Amigué. Las máquinas tienen que hablar en humano. *Entrevista a Lofti A. Zadeh*, *La Vanguardia*, 23/11/2004, página 68.
- [3] Lofti A. Zadeh. From circuit theory to system theory. In *Proceedings of the Institute of Radio Engineers*, volume 50, pages 856–865, May 1962.
- [4] E.T. Lee and Lotfi A. Zadeh. Note on fuzzy languages. *Information Sciences*, 1(4):421–434, October 1969.
- [5] Lofti A. Zadeh. Similarity relations and fuzzy orderings. *Information Sciences*, 3(2):177–200, April 1971.
- [6] Lofti A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man, and Cybernetics*, 3:28–44, January 1973.
- [7] Lofti A. Zadeh. Fuzzy logic. *IEEE Computer*, 21(4):83–93, April 1988.
- [8] Lofti A. Zadeh. Fuzzy algorithms. *Information and Control*, 12(2):94–102, 1968.
- [9] Lofti A. Zadeh. Fuzzy logic and its application to approximate reasoning. In *IFIP Congress*, pages 591–594, 1974.
- [10] Lofti A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning – I. *Information Sciences*, 8(3):199–249, 1975.
- [11] Lofti A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning – II. *Information Sciences*, 8(4):301–357, 1975.
- [12] Lofti A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning – III. *Information Sciences*, 9(1):43–80, 1975.

- [13] Lofti A. Zadeh. Generalized theory of uncertainty (GTU)—principal concepts and ideas. *Computational Statistics & Data Analysis*, 51(1):15–46, November 2006.
- [14] John Yen and Reza Langari. *Fuzzy Logic: Intelligence, control, and information*. Prentice Hall, 1999.
- [15] Lotfi A. Zadeh. Granular computing and Rough Set Theory. In Springer Berlin / Heidelberg, editor, *Granular Computing and Rough Set Theory*, volume 4585/2007, pages 1–4, 2007.
- [16] R. E. Bellman and Lofti A. Zadeh. Decision-making in a fuzzy environment. *Management Science*, 17(4):B141–B164, December 1970.
- [17] George Lakoff. Hedges: A study in meaning criteria and the logic of fuzzy concepts. *Journal of Philosophical Logic*, 2(4):458–508, October 1973.
- [18] J. A. Goguen. The logic of inexact concepts. *Synthese*, 19(3-4):325–373, April 1969.
- [19] R. E. Smith. *Measure theory on fuzzy sets*. PhD thesis, Dept. of Mathematics, University of Saskatchewan, Saskatoon, Canada, 1970.
- [20] Michio Sugeno. *Theory of fuzzy integrals and its applications*. PhD thesis, Tokyo Institute of Technology, 1974.
- [21] James C. Bezdek and Joseph C. Dunn. Optimal fuzzy partitions: A heuristic for estimating the parameters in a mixture of normal distributions. *IEEE Transactions on Computers*, C-24(8):835–838, August 1975.
- [22] J. C. Dunn. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3:32–57, 1973.
- [23] Constantin Virgil Negoita. *Information retrieval systems*. PhD thesis, Polytechnical Institute, Bucharest, Romania, 1969.
- [24] Constantin Virgil Negoita. Fuzzy systems and artificial intelligence. *Cybernetes*, 3:173–178, 1974.
- [25] H. J. Zimmermann. Description and optimization of fuzzy systems. *International Journal of General Systems*, 2:209–215, 1976.
- [26] José Meseguer and Ignacio Sols. Automata in semimodule categories. In *Proceedings of the First International Symposium on Category Theory Applied to Computation and Control*, pages 193–198, London, UK, 1975. Springer-Verlag.
- [27] Ignacio Sols. *Aportaciones a la teoría de topos, al álgebra universal y a las matemáticas fuzzy*. PhD thesis, Universidad de Zaragoza, Zaragoza, España, 1975.
- [28] Ebrahim H. Mamdani and S. Assilian. An experimental in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7(1):1–13, 1975.

- [29] S. Assilian. *Artificial intelligence in the control of real dynamical systems*. PhD thesis, Queen Mary College, London University, 1974.
- [30] Ebrahim H. Mamdani. Twenty years of fuzzy control: experiences gained and lessons learned. *Second IEEE International Conference on Fuzzy Systems*, 1:339–344, March 1993.
- [31] P. J. King and E. H. Mamdani. The application of fuzzy control systems to industrial processes. *Automatica*, 13(3):235–242, May 1977.
- [32] M. Togai and H. Watanabe. Expert system on a chip: an engine for real-time approximate reasoning. In *Proceedings of the ACM SIGART international symposium on Methodologies for intelligent systems*, pages 147–154, New York, NY, USA, 1986. ACM Press.
- [33] S. Yasunobu and S. Miyamoto. Automatic train operation by predictive fuzzy control. In Sugeno, editor, *Industrial Applications of Fuzzy Control*, pages 1–18, Amsterdam, 1985.
- [34] Kaoru Hirota. Industrial education on fuzzy systems in Japan. In *Proceedings of the Industrial Electronics, Control, and Instrumentation*, volume 1, pages 176–181, Maui, HI, USA, November 1993.
- [35] J. Gebhardt. Application of fuzzy logic to the control of a wind energy converter. In *First European Congress on Fuzzy and Intelligent Technologies*, pages 339–348, Aachen, 1993.
- [36] C. von Altrock, B. Krause, K. Limper, and W. Schäfers. Optimization of a waste incineration plant using fuzzy logic. In *Second European Congress on Fuzzy and Intelligent Technologies*, Aachen, 1994.
- [37] Oscal T. Chen, Yao-Chou Lu, and Hwai-Tsu Chang. Fuzzy reasoning processor for camera image autofocus. In Lance T. Wu, editor, *Visual Communications and Image Processing'95*, volume 2501, pages 347–354. SPIE, 1995.
- [38] T. Haruki and K. Kikuchi. Video camera system using fuzzy logic. *IEEE Transactions on Consumer Electronics*, 38(3):624–634, August 1992.
- [39] Lee Yongman, Jang Seong-Ik, Chung Keewook, Lee Dongyun, Kim Wonchan, and Lee Choong-Woong. Fuzzy-control processor for automatic focusing. *IEEE Transactions on Consumer Electronics*, 40(2):138–144, May 1994.
- [40] P. Hofbauer, H. O. Arend, and D. Pfannstiel. New heating systems controls based on the use of fuzzy logic. In *First European Congress on Fuzzy and Intelligent Technologies*, pages 1036–1042, Aachen, 1993.
- [41] H. Steinmüller and O. Wick. Fuzzy and neurofuzzy applications in european washing machines. In *First European Congress on Fuzzy and Intelligent Technologies, EUFIT'93*, pages 1031–1035, Aachen, 1993.

- [42] José Ferrer, María A. Rodrigo, Aurora Seco, and Josep Manuel Peña-roja. Energy saving in the aeration process by fuzzy logic control. *Water Science and Technology*, 38(3):209–217, 1998.
- [43] C. von Altrock. Fuzzy logic technologies in automotive engineering. In *Embedded systems conference*, volume 2, pages 407–422, 1994.
- [44] C. von Altrock, B. Krause, and H. J. Zimmermann. Advanced fuzzy logic control of a model car in extreme situations. *Fuzzy Sets and Systems*, 48(1):41–52, 1992.
- [45] C. von Altrock, B. Krause, and H. J. Zimmermann. Advanced fuzzy logic control technologies in automotive applications. In *IEEE Conference on Fuzzy Systems*, pages 831–842, 1992.
- [46] A.S. Cherry and R.P. Jones. Fuzzy logic control of an automotive suspension system. In *IEE Proceedings: Control Theory and Applications*, pages 149–160. IEE, Stevenage, United Kingdom, March 1995.
- [47] V. Rao, M.V.C. adn Prahlad. A tunable fuzzy logic controller for vehicle-active suspension systems. *Fuzzy Sets and Systems*, 85(1):11–21, January 1997.
- [48] Y. Taskin, Y. Hacioglu, and N. Yagiz. The use of fuzzy-logic control to improve the ride comfort of vehicles. *Journal of Mechanical Engineering*, 53(4):233–240, April 2007.
- [49] Hyun Mun Kim, Julie Dickerson, and Bart Kosko. Fuzzy throttle and brake control for platoons of smart cars. *Fuzzy Sets and Systems*, 84(3):209–234, December 1996.
- [50] Jeffery R. Layne, Kevin M. Passino, and Stephen Yurkovich. Fuzzy learning control for antiskid braking systems. *IEEE Transactions on Control Systems Technology*, 1(2):122–129, June 1993.
- [51] Y. Lee and S.H. Zak. Designing a genetic neural fuzzy antilock-brake-system controller. *IEEE Transactions on Evolutionary Computation*, 6(2):198–211, April 2002.
- [52] William K. Lennon and Kevin M. Passino. Intelligent control for brake systems. *IEEE Transactions on Control Systems Technology*, 7(2):188–202, March 1999.
- [53] D.P. Madau, F. Yuan, Jr. Davis, L.I., and L.A. Feldkamp. Fuzzy logic anti-lock brake system for a limited range coefficient of friction surface. In *Second IEEE International Conference on Fuzzy Systems*, pages 883–888. IEEE, Piscataway, NJ, United States, 1993.
- [54] Georg F. Mauer. Fuzzy logic controller for an ABS braking system. *IEEE Transactions on Fuzzy Systems*, 3(4):381–388, November 1995.
- [55] G.J. Vachtsevanos, S.S. Farinwata, and D.K. Pirovolou. Fuzzy logic control of an automotive engine. *IEEE Control Systems Magazine*, 13(3):62–68, June 1993.

- [56] B.M. Baumann, G. Washington, B.C. Glenn, and G. Rizzoni. Mechatronic design and control of hybrid electric vehicles. *IEEE/ASME Transactions on Mechatronics*, 5(1):58–72, March 2000.
- [57] S.D. Farrall and R.P. Jones. Energy management in an automotive electric/heat engine hybrid powertrain using fuzzy decision making. In *Proceedings of the 1993 IEEE International Symposium on Intelligent Control*, pages 463–468, Chicago, IL, August 1993.
- [58] Hyeoun-Dong Lee and Seung-Ki Sul. Fuzzy-logic-based torque control strategy for parallel-type hybrid electric vehicle. *IEEE Transactions on Industrial Electronics*, 45(4):625–632, August 1998.
- [59] N.J. Schouten, M.A. Salman, and N.A. Kheir. Fuzzy logic control for parallel hybrid vehicles. *IEEE Transactions on Control Systems Technology*, 10(3):460–468, May 2002.
- [60] N.J. Schouten, M.A. Salman, and N.A. Kheir. Energy management strategies for parallel hybrid vehicles using fuzzy logic. *Control Engineering Practice*, 11(2):171–177, February 2003.
- [61] Jong-Seob Won and Reza Langari. Fuzzy torque distribution control for a parallel hybrid vehicle. *Expert Systems*, 19(1):4–10, February 2002.
- [62] Claudi Alsina and Enric Trillas. On uniformly close fuzzy preorders. *Fuzzy Sets and Systems*, 53(3):343–346, 1993.
- [63] Claudi Alsina and Enric Trillas. On the law $S(S(x, y), T(x, y)) = S(x, y)$ of fuzzy logic. *Fuzzy Optimization and Decision Making*, 6(2):99–107, 2007.
- [64] Claudi Alsina, Enric Trillas, and Llorenç James Valverde. On some logical connectives for fuzzy sets theory. *Journal of Mathematical Analysis and Applications*, 93(1):15–26, January 1983.
- [65] Juan Luis Castro and Enric Trillas. The management of the inconsistency in expert systems. *Fuzzy Sets and Systems*, 58(1):51–57, 1993.
- [66] Juan Luis Castro, Enric Trillas, and José Manuel Zurita. Non-monotonic fuzzy reasoning. *Fuzzy Sets and Systems*, 94(2):217–225, March 1998.
- [67] Claudio Moraga, Enric Trillas, and Sergio Guadarrama. Multiple-valued logic and artificial intelligence fundamentals of fuzzy control revisited. *Artificial Intelligence Review*, 20(3–4):169–197, December 2003.
- [68] Ana Pradera, Enric Trillas, and Tomasa Calvo. A general class of triangular norm-based aggregation operators: Quasi-linear T-S operators. *International Journal of Approximate Reasoning*, 30(1):57–72, May 2002.

- [69] Enric Trillas. Sobre funciones de negación en la teoría de conjuntos difusos. *Stochastica*, 3:47–60, 1979.
- [70] Enric Trillas and Cubillo Alsina. Some remarks on approximate entailment. *International Journal of Approximate Reasoning*, 6(4):525–533, 1992.
- [71] Enric Trillas. *Conjuntos borrosos*. Vicens–Vives, 1980.
- [72] Enric Trillas, Cubillo Alsina, and J. M. Terricabras. *Introducción a la lógica borrosa*. Ariel, Barcelona, 1995.
- [73] Enric Trillas and Susana Cubillo. *Primeras lecciones sobre lógica borrosa*. Facultad de Informática, Servicio de Publicaciones, 1999.
- [74] Enric Trillas, Miguel Delgado, M. Amparo Vila, Juan Luis Castro, J. L. Verdegay, Claudio Moraga, José Cuenca, Julio Gutierrez, and Antonio Ruiz. *Fundamentos e introducción a la ingeniería “Fuzzy”*. Omron Electronics S.A., 1994.
- [75] F. Chiclana, F. Herrera, and E. Herrera-Viedma. Integrating three representation models in fuzzy multipurpose decision making based on fuzzy preference relations. *Fuzzy Sets and Systems*, 97(1):33–48, July 1998.
- [76] O. Cordón, F. Gomide, F. Herrera, F. Hoffmann, and L. Magdalena. Ten years of genetic fuzzy systems: Current framework and new trends. *Fuzzy Sets and Systems*, 141(1):5–31, January 2004.
- [77] Miguel Delgado, Francisco Herrera, Enrique Herrera-Viedma, and Luís Martínez. Combining numerical and linguistic information in group decision making. *Information Sciences*, 107(1-4):177–194, 1998.
- [78] F. Herrera and E. Herrera-Viedma. Aggregation operators for linguistic weighted information. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, 27(5):646–656, September 1997.
- [79] F. Herrera, E. Herrera-Viedma, and F. Chiclana. Multiperson decision-making based on multiplicative preference relations. *European Journal of Operational Research*, 129(2):372–385, March 2001.
- [80] F. Herrera, E. Herrera-Viedma, and L. Martínez. A fusion approach for managing multi-granularity linguistic term sets in decision making. *Fuzzy Sets and Systems*, 114(1):43–58, August 2000.
- [81] F. Herrera, E. Herrera-Viedma, and J.L. Verdegay. Sequential selection process in group decision making with a linguistic assessment approach. *Information sciences*, 85(4):223–239, July 1995.
- [82] F. Herrera, E. Herrera-Viedma, and J.L. Verdegay. Direct approach processes in group decision making using linguistic OWA operators. *Fuzzy Sets and Systems*, 79(2):175–190, April 1996.

- [83] F. Herrera, E. Herrera-Viedma, and J.L. Verdegay. A model of consensus in group decision making under linguistic assessments. *Fuzzy Sets and Systems*, 78(1):73–87, February 1996.
- [84] F. Herrera, E. Herrera-Viedma, and J.L. Verdegay. A rational consensus model in group decision making using linguistic assessments. *Fuzzy Sets and Systems*, 88(1):31–49, May 1997.
- [85] F. Herrera, M. Lozano, and J.L. Verdegay. A learning process for fuzzy control rules using genetic algorithms. *Fuzzy Sets and Systems*, 100(1-3):143–158, November 1998.
- [86] F. Herrera and L. Martínez. A 2-tuple fuzzy linguistic representation model for computing with words. *IEEE Transactions on Fuzzy Systems*, 8(6):746–752, December 2000.
- [87] Miguel Delgado, Nicolás Marín, Daniel Sánchez, and María Amparo Vila. Fuzzy association rules: General model and applications. *IEEE Transactions on Fuzzy Systems*, 11(2):214–225, 2003.
- [88] Miguel Delgado, Daniel Sánchez, María J. Martín-Bautista, and María Amparo Vila. A probabilistic definition of a nonconvex fuzzy cardinality. *Fuzzy Sets and Systems*, 126(2):177–190, 2002.
- [89] Miguel Delgado, Daniel Sánchez, and María Amparo Vila. Fuzzy cardinality based evaluation of quantified sentences. *International Journal of Approximate Reasoning*, 23(1):23–66, 2000.
- [90] Miguel Delgado, José Luis Verdegay, and María Amparo Vila. Linguistic decision-making models. *International Journal of Intelligent Systems*, 7(5):479–492, March 1992.
- [91] Miguel Delgado, José Luis Verdegay, and María Amparo Vila. On aggregation operations of linguistic labels. *International Journal of Intelligent Systems*, 8(3):351–370, March 1993.
- [92] Miguel Delgado, José Luis Verdegay, and María Amparo Vila. Model for linguistic partial information in decision-making problems. *International Journal of Intelligent Systems*, 9(4):365–378, April 1994.
- [93] Pablo Carmona, Juan Luis Castro, and José Manuel Zurita. Strategies to identify fuzzy rules directly from certainty degrees: A comparison and a proposal. *IEEE Transactions on Fuzzy Systems*, 12(5):631–640, October 2004.
- [94] Juan Luis Castro. Fuzzy logics as families of bivaluated logics. *Fuzzy Sets and Systems*, 64(3):321–332, June 1994.
- [95] Juan Luis Castro. Fuzzy logic controllers are universal approximators. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(4):629–635, April 1995.

- [96] Juan Luis Castro, José Jesús Castro-Schez, and J.M. Zurita. Learning maximal structure rules in fuzzy logic for knowledge acquisition in expert systems. *Fuzzy Sets and Systems*, 101(3):331–342, February 1999.
- [97] Juan Luis Castro and Miguel Delgado. Fuzzy systems with defuzzification are universal approximators. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, 26(1):149–152, February 1996.
- [98] Juan Luis Castro, C.J. Mantas, and José M. Benítez. Interpretation of artificial neural networks by means of fuzzy rules. *IEEE Transactions on Neural Networks*, 13(1):101–116, January 2002.
- [99] Juan Luis Castro and J.M. Zurita. An inductive learning algorithm in fuzzy systems. *Fuzzy Sets and Systems*, 89(2):193–203, July 1997.
- [100] Javier Aracil, Anibal Ollero, and Alfonso Garcia-Cerezo. Stability indices for the global analysis of expert control systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):998–1007, September 1989.
- [101] Javier Aracil and Francisco Gordillo. Describing function method for stability analysis of PD and PI fuzzy controllers. *Fuzzy Sets and Systems*, 143(2):233–249, April 2004.
- [102] Francisco Gordillo, Javier Aracil, and Teodoro Alamo. Determining limit cycles in fuzzy control systems. In *IEEE International Conference on Fuzzy Systems*, volume 1, pages 193–198, July 1997.
- [103] Begoña C. Arrue, Anibal Ollero, and Jose Ramiro Martinez De Dios. An intelligent system for false alarm reduction in infrared forest-fire detection. *IEEE Intelligent Systems and Their Applications*, 15(3):64–73, May 2000.
- [104] Federico Cuesta, Francisco Gordillo, Javier Aracil, and Anibal Ollero. Stability analysis of nonlinear multivariable Takagi–Sugeno fuzzy control systems. *IEEE Transactions on Fuzzy Systems*, 7(5):508–520, October 1999.
- [105] Federico Cuesta, Anibal Ollero, Begoña C. Arrue, and Reinhard Braunstingl. Intelligent control of nonholonomic mobile robots with fuzzy perception. *Fuzzy Sets and Systems*, 134(1):47–64, February 2003.
- [106] Federico Cuesta, Enrique Ponce, and Javier Aracil. Local and global bifurcations in simple Takagi–Sugeno fuzzy systems. *IEEE Transactions on Fuzzy Systems*, 9(2):355–368, April 2001.
- [107] Fernando Gómez-Bravo, Federico Cuesta, and Anibal Ollero. Parallel and diagonal parking in nonholonomic autonomous vehicles. *Engineering Applications of Artificial Intelligence*, 14(4):419–434, August 2001.

- [108] Jose Ramiro Martinez-De Dios and Anibal Ollero. A multiresolution-fuzzy method for robust threshold selection in image segmentation. *Intelligent Automation and Soft Computing*, 12(4):419–430, 2006.
- [109] Anibal Ollero, Javier Aracil, and Alfonso García-Cerezo. Robust design of rule-based fuzzy controllers. *Fuzzy Sets and Systems*, 70(2–3):249–273, January 1995.
- [110] Anibal Ollero, Javier Aracil, and Francisco Gordillo. Stability analysis of MIMO fuzzy control systems in the frequency domain. In *IEEE International Conference on Fuzzy Systems*, volume 1, pages 49–54, Anchorage, AK, May 1998.
- [111] Anibal Ollero, Alfonso García-Cerezo, and Jorge Martinez. Fuzzy supervisory path tracking of mobile robots. *Control Engineering Practice*, 2(2):313–319, April 1994.
- [112] Sebastian Dormido, Matilde Santos, A. P. de Madrid, and F. Morilla. Autosintonía de controladores borrosos utilizando técnicas clásicas basadas en reguladores PID. In Universidad de Santiago de Compostela, editor, *Comunicaciones del III FLAT*, pages 217–225, Santiago de Compostela, España, 1993.
- [113] Matilde Santos, Jesús Manuel de la Cruz, and Sebastián Dormido. Self-tuning of fuzzy logic controllers in cascade loops. *Intelligent Systems and Soft Computing for Nuclear Science and Industry*, pages 258–264, 1996.
- [114] Matilde Santos, Sebastián Dormido, and Jesús Manuel de la Cruz. Evaluation of the fuzzy controllers performance. In *4th European Congress on Intelligent Techniques and Soft Computing EUFIT'96*, volume 2, pages 1056–1060, Verlag Mainz, Aachen, 1996.
- [115] Matilde Santos, Sebastián Dormido, and Jesús Manuel de la Cruz. Fuzzy-PID controllers vs. fuzzy-PI controllers. In *5th IEEE International Conference on Fuzzy Systems FUZZ-IEEE'96*, pages 1598–1604, 1996.
- [116] Matilde Santos, Sebastian Dormido, and Jesús Manuel de la Cruz. Influence of the information processing in fuzzy logic controllers. In Proyecto Sur de Ediciones, editor, *6th International Conference Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU'96*, volume 1, pages 81–86, 1996.
- [117] Matilde Santos, Sebastián Dormido, Jesús Manuel de la Cruz, and José Antonio López. Tuning of fuzzy controllers: Applications of the relay method. In *EUROSIM*, volume 3, pages 1131–1136, New Orleans, 1995.
- [118] Matilde Santos, Sebastian Dormido, A. P. de Madrid, and Jesús Manuel de la Cruz. Between fuzzy-PID and PID-conventional controllers: a good choice. *Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS)*, pages 123–127, 1996.

- [119] Matilde Santos, Sebastian Dormido, A. P. de Madrid, F. Morilla, and Jesús Manuel de la Cruz. Tuning fuzzy logic controllers by classical techniques. In *CASR'94: Selected papers from the 4th International Workshop on Computer Aided System Theory*, pages 214–224, London, UK, 1996. Springer-Verlag.
- [120] Basil M. Al-Hadithi, Fernando Matía, and Agustín Jiménez. Slide-mode fuzzy controller for highly non linear systems. In *The 10th IEEE International Conference on Fuzzy Systems, 2001*, pages 1515–1518, 2001.
- [121] Basil M. Al-Hadithi, Fernando Matía, and Agustín Jiménez. Análisis de estabilidad de sistemas borrosos. *Revista Iberoamericana de Automática e Informática Industrial (RIAI)*, 4(2):7–25, April 2007.
- [122] Basil M. Al-Hadithi, Fernando Matía, and Agustín Jiménez. Fuzzy controller for robot manipulators. In *IFSA*, pages 688–697, 2007.
- [123] Fernando Matía, Basil M. Al-Hadithi, and Agustín Jiménez. On the global stability of Takagi–Sugeno general model. *Mathware and Soft Computing*, 6(2–3):293–304, 1999.
- [124] Fernando Matía, Basil M. Al-Hadithi, and Agustín Jiménez. Generalization of stability criterion for Takagi–Sugeno continuous fuzzy model. *Fuzzy Sets and System*, 129(3):295–309, August 2002.
- [125] Fernando Matía, Agustín Jiménez, Basil M. Al-Hadithi, Diego Rodríguez-Losada, and Ramón Galán. The fuzzy Kalman filter: State estimation using possibilistic techniques. *Fuzzy Sets and Systems*, 157(16):2145–2170, August 2006.
- [126] Alfonso García-Cerezo, Anibal Ollero, and Javier Aracil. Dynamic analysis of weighed-output fuzzy control systems. *Annual Review in Automatic Programming*, 19(C):43–48, June 1994.
- [127] María J. López-Baldán, Alfonso García-Cerezo, José M. Cejudo, and A. Romero. Fuzzy modeling of a thermal solar plant. *International Journal of Intelligent Systems*, 17(4):369–379, April 2002.
- [128] José Ruiz-Gómez, María J. López-Baldán, and Alfonso García-Cerezo. Fuzzy modelling of a ternary batch distillation column. *System and Control: Theory and Applications*, pages 320–324, 2000.
- [129] Alfonso Garcia-Cerezo, Anthony Mandow, and María J. Lopez-Baldan. Fuzzy modelling operator navigation behaviors. In *IEEE International Conference on Fuzzy Systems*, volume 3, pages 1339–1345, Piscataway, NJ, United States, July 1997. IEEE.
- [130] María J. Lopez-Baldan, José Ruiz-Gómez, R. Fernandez, and Alfonso García-Cerezo. Input-output fuzzy modelling applied to a mobile robot. *Computational Intelligence and Applications*, pages 283–288, 1999.

- [131] Senén Barro, Roque Marín, Francisco Palacios, and Ramón Ruíz. Fuzzy logic in a patient supervision system. *Artificial Intelligence in Medicine*, 21(1-3):193–199, 2001.
- [132] J.M. Cárdenas, M.A. Marín, and I. Navarrete. Fuzzy temporal constraint logic: a valid resolution principle. *Fuzzy Sets and Systems*, 117(2):231–250, January 2001.
- [133] Paulo Félix, Senén Barro, and Roque Marín. Fuzzy constraint networks for signal pattern recognition. *Artificial Intelligence*, 148(1-2):103–140, August 2003.
- [134] Paulo Félix, Santiago Fraga, Senén Barro, and Roque Marín. Linguistic representation of fuzzy temporal profiles. *International Journal Uncertainly Fuzziness Knowledge-Based Systems*, 7(3):243–256, 1999.
- [135] Paulo Félix, Santiago Fraga, Roque Marín, and Senén Barro. Trend detection based on a fuzzy temporal profile model. *Artificial Intelligence in Engineering*, 13:341–349, October 1999.
- [136] Antonio F. Gómez-Skarmeta, Mercedes Valdés, Fernando Jiménez, and Javier G. Marín-Blázquez. Approximative fuzzy rules approaches for classification with hybrid-GA techniques. *Information Sciences*, 136(1-4):193–214, August 2001.
- [137] F. Jiménez, J. M. Cadenas, J. L. Verdegay, and G. Sánchez. Solving fuzzy optimization problems by evolutionary algorithms. *Information Sciences*, 152(1):303–311, June 2003.
- [138] F. Jiménez, J. M. Cadenas, G. Sánchez, A.F. Gómez-Skarmeta, and J.L. Verdegay. Multi-objective evolutionary computation and fuzzy optimization. *International Journal of Approximate Reasoning*, 43(1):59–75, September 2006.
- [139] Jose Palma, Jose M. Juarez, Manuel Campos, and Roque Marín. Fuzzy theory approach for temporal model-based diagnosis: An application to medical domains. *Artificial Intelligence in Medicine*, 38(2):197–218, October 2006.
- [140] José Vicente Salcedo and M. Ángeles Martínez. BIBO stabilisation of Takagi-Sugeno fuzzy systems under persistent perturbations using fuzzy output-feedback controllers. *IET Control Theory and Applications*, 2(6):513–523, June 2008.
- [141] José Vicente Salcedo and M. Ángeles Martínez. Design of PDC fuzzy controllers under persistent disturbances and application in mechanical systems. *Advances in Engineering Software*, 39(11):937–946, November 2008.
- [142] José Vicente Salcedo, M. Ángeles Martínez, and S. García. Stabilization conditions of fuzzy systems under persistent perturbations and their application in nonlinear systems. *Engineering Applications of Artificial Intelligence*, In-press.
- [143] Jorge Bondia and Jesús Picó. Analysis of linear systems with fuzzy parametric uncertainty. *Fuzzy Sets and Systems*, 135(1):81–121, April 2003.

- [144] Jorge Bondia and Jesús Picó. Application of functional intervals to the response evaluation of linear time-invariant systems with fuzzy input. *Reliable Computing*, 10(5):369–387, October 2004.
- [145] Jorge Bondia, Antonio Sala, Jesús Picó, and Miguel Ángel Sainz. Controller design under fuzzy pole-placement specifications: An interval arithmetic approach. *IEEE Transactions on Fuzzy Systems*, 14(6):822–836, December 2006.
- [146] Antonio Sala and Pedro Albertos. Inference error minimisation: fuzzy modelling of ambiguous functions. *Fuzzy Sets and Systems*, 121(1):95–111, July 2001.
- [147] Pedro Albertos and Antonio Sala. Perspectives of fuzzy control: Lights and shadows. In *2nd International IEEE Conference 'Intelligent Systems'*, volume 1, pages 25–32, 2004.
- [148] Jose Luis Díez, Jose Luis Navarro, and Antonio Sala. A fuzzy clustering algorithm enhancing local model interpretability. *Soft Computing*, 11(10):973–983, August 2007.
- [149] Antonio Sala and Pedro Albertos. Fuzzy systems evaluation: The inference error approach. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, 28(2):268–275, April 1998.
- [150] Antonio Sala, Jose Luis Díez, Jose Luis Navarro, and Pedro Albertos. Fuzzy model usage and readability in identification for control. In *Intelligent Automation and Control Trends, Principles, and Applications - Proceedings of the Sixth Biannual World Automation Congress, WAC*, pages 113–118, Seville, 2004.
- [151] J. Juez, Gregorio I. Sainz, E. J. Moya, and J. R. Perán. Early detection and diagnosis of faults in an AC motor using neurofuzzy techniques: FasArt+fuzzy K nearest neighbors. *Lectures Notes in Computer Science*, 2085:571–578, 2001.
- [152] L. J. de Miguel, M. Mediavilla, and J. R. Peran. Fault diagnosis system based on sensitivity analysis and fuzzy logic. In *26th International Symposium on Multiple-Valued Logic (ISMVL'96)*, page 50, Washington, DC, USA, 1996. IEEE Computer Society.
- [153] Gregorio I. Sainz, M. J. Fuente, and P. Vega. Recurrent neuro-fuzzy modelling of a wastewater treatment plant. *European Journal of Control*, 10(1):83–95, January 2004.
- [154] Gregorio I. Sainz, J. J. Santamaria, and Y. A. Dimitriadis. Performance of a recurrent neuro-fuzzy ART based system for pattern recognition and modeling of dynamical systems: RFasArt. *IFSA World Congress and 20th NAFIPS International Conference, 2001. Joint 9th*, 1(1):534–539, July 2001.
- [155] Gregorio Sainz, Yannis Dimitriadis, J. M. Cano, E. Gómez, and E. Parrado. *ART based model set for pattern recognition: FasArt family*, chapter Neuro-Fuzzy systems for pattern recognition. Scientific World Publ. Co., 2000.

- [156] J. Acosta, A. Nebot, P. Villar, and J. M. Fuertes. Automatic learning of fuzzy partitions in human central nervous system modeling using genetic algorithms. In J. G. Anderson and M. Katzper, editors, *Inter. Conf. on Health Science Simulation*, pages 90–95. SCS International, 2004.
- [157] A. Barcelo, E. Montseny, and P. Sobrevilla. On fuzzy texture spectrum for natural microtextures characterization. In *conference EUSFLAT'05*, Barcelona, Spain, September 2005.
- [158] E. Montseny, P. Sobrevilla, and P. Mares. Edge orientation-based fuzzy hough transform (EOFHT). In *3rd Internacional Conference on Fuzzy Logic and Technology . Prof. Rainer Hampel*, pages 683–688, 2003.
- [159] Teresa Alsinet, C. Chesñevar, Lluís Godo, Sandra Sandri, and G. Simari. Formalizing argumentative reasoning in a possibilistic logic programming setting with fuzzy unification. *International Journal of Approximate Reasoning*, 2008.
- [160] Teresa Alsinet, C. Chesñevar, Lluís Godo, and G. Simari. A logic programming framework for possibilistic argumentation: Formalization and logical properties. *Fuzzy Sets and Systems*, 159(10):1208–1228, May 2008.
- [161] Félix Bou, Francesc Esteva, and Lluís Godo. Modal systems based on many-valued logics. In M. Stepnicka, V. Novák, and U. Bodenhofer, editors, *New dimensions in fuzzy logic and related technologies. EUSFLAT*, volume I, pages 177–182, 2007.
- [162] Didier Dubois, Francesc Esteva, Lluís Godo, and Henri Prade. *Fuzzy-set Based Logics. An History-oriented Presentation of their Main Developments*, volume 8, pages 325–449. Elsevier, 2007.
- [163] Francesc Esteva, Joan Gispert, Lluís Godo, and Carles Noguera. Adding truth-constants to logics of continuous T-norms: axiomatization and completeness results. *Fuzzy Sets and Systems*, 158(6):597–618, March 2007.
- [164] Tommaso Flaminio and Lluís Godo. A logic for reasoning about the probability of fuzzy events. *Fuzzy Sets and Systems*, 158(6):625–638, March 2007.
- [165] Lluís Godo and Ricardo Oscar Rodriguez. *Logical approaches to fuzzy similarity-based reasoning: an overview*. Springer-Verlag, 2008.
- [166] Lluís Godo and Vicenç Torra. On aggregation operators for ordinal qualitative information. *IEEE Transactions on Fuzzy Systems*, 8(2):143–154, Apr 2000.
- [167] Vicenç Torra and Yasuo Narukawa. Fuzzy measures and integrals in evaluation of strategies. *Information Sciences*, 177(21):4686–4695, November 2007.
- [168] José Manuel Andújar and Antonio Javier Barragán. A methodology to design stable nonlinear fuzzy control systems. *Fuzzy Sets and Systems*, 154(2):157–181, September 2005.

- [169] José Manuel Andújar, Antonio Javier Barragán, Juan Manuel Córdoba, and Iñaki Fernández. Diseño de sistemas de control borroso: Modelado de la planta. *Revista Iberoamericana de Automática e Informática Industrial (RIAI)*, 3(1):75–81, January 2006.
- [170] José Manuel Andújar, Antonio Javier Barragán, and Manuel Emilio Gegúndez. A general and formal methodology for designing stable nonlinear fuzzy control systems. *IEEE Transactions on Fuzzy Systems*, En prensa, 2009.
- [171] José Manuel Andújar, Antonio Javier Barragán, Manuel Emilio Gegúndez, and Manuel Maestre. Control borroso multivariable basado en heurística. un caso práctico: Grúa porta contenedores. *Revista Iberoamericana de Automática e Informática Industrial (RIAI)*, 4(2):81–89, 2007.
- [172] Javier Aroba, José Antonio Grande, José Manuel Andújar, María Luisa De La Torre, and Juan Carlos Riquelme. Application of fuzzy logic and data mining techniques as tools for qualitative interpretation of acid mine drainage processes. *Environmental Geology*, 53(1):135–145, September 2007.
- [173] José Antonio Grande, José Manuel Andújar, Javier Aroba, María Luisa De La Torre, and Rafael Beltrán. Precipitation, pH and metal load in AMD river basins: An application of fuzzy clustering algorithms to the process characterization. *Journal of Environmental Monitoring*, 7(4):325–334, April 2005.
- [174] María Luisa De la Torre, José Antonio Grande, Javier Aroba, and José Manuel Andújar. Optimization of fertirrigation efficiency in strawberry crops by application of fuzzy logic techniques. *Journal of Environmental Monitoring*, 7(11):1085–1092, November 2005.
- [175] D. Driankov, H. Hellendoorn, and M. Reinfrank. *An introduction to fuzzy control*. Springer-Verlag, New York, USA, 1993.
- [176] S. C. Kleene. *Introduction to metamathematics*. Von Nostrand, New York, 1952.
- [177] G. Birkhoff. *Lattice Theory*. American Mathematical Society, 2nd edition, 1948.
- [178] K. Menger. Statistical metric spaces. In *National Academy of Sciences*, volume 37, pages 535–537, USA, 1942.
- [179] B. Schweizer and A. Sklar. *Probabilistic metric spaces*. North-Holland, Amsterdam, 1983.
- [180] Erich-Peter Klement, Radko Mesiar, and Endre Pap. Triangular norms. Position paper I: basic analytical and algebraic properties. *Fuzzy Sets and Systems*, 143(1):5–26, 2004.
- [181] Erich-Peter Klement, Radko Mesiar, and Endre Pap. Triangular norms. Position paper II: general constructions and parameterized families. *Fuzzy Sets and Systems*, 145(3):411–438, 2004.

- [182] Erich-Peter Klement, Radko Mesiar, and Endre Pap. Triangular norms. Position paper III: continuous T-norms. *Fuzzy Sets and Systems*, 145(3):439–454, 2004.
- [183] Javier Corrales Holgado. *Controladores en lógica borrosa*. Colección Tecnología e investigación. Servidio de Publicaciones de la Universidad de Cádiz, 1995.
- [184] Lofti A. Zadeh. Semantic inference from fuzzy premises. In *Proceedings of the 6th Int. symposium on Multiple-valued logic*, pages 217–218, Los Alamitos, CA, USA, 1976. IEEE Computer Society Press.
- [185] G. A. Miller. The magical number seven or minus two: some limits on our capacity of processing information. *Psychological Rev.*, 63:81–97, 1956.
- [186] Bart Kosko and Julie A. Dickerson. *Theoretical aspects of fuzzy control*, chapter Function approximation with additive fuzzy systems, pages 313–347. John Wiley & Sons, Inc., New York, NY, USA, 1995.
- [187] Masaharu Mizumoto. Fuzzy controls under various fuzzy reasoning methods. *Information Sciences*, 45(2):129–151, July 1988.
- [188] Masaharu Mizumoto and Hans-Jurgen Zimmermann. Comparison of fuzzy reasoning methods. *Fuzzy Sets and Systems*, 8(3):253–283, September 1982.
- [189] I. B. Turksen and Y. Tian. Combination of rules or their consequences in fuzzy expert systems. *Fuzzy Sets and Systems*, 58(1):3–40, August 1993.
- [190] Jerry M. Mendel. Fuzzy logic systems for engineering: A tutorial. *Proceedings of the IEEE*, 83(3):345–377, March 1995.
- [191] George J. Klir and Bo Yuan. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice-Hall International, New Jersey, November 1995.
- [192] Dimitar P. Filev and Ronald R. Yager. An adaptive approach to defuzzification based on level sets. *Fuzzy Sets and Systems*, 54(3):355–360, March 1993.
- [193] Xinwang Liu. Parameterized defuzzification with maximum entropy weighting function—another view of the weighting function expectation method. *Mathematical and Computer Modelling*, 45(1-2):177–188, January 2007.
- [194] S. Mabuchi. A proposal for a defuzzification strategy by the concept of sensitivity analysis. *Fuzzy Sets and Systems*, 55(1):1–14, April 1993.
- [195] Thomas Vetterlein and Mirko Navara. Defuzzification using Steiner points. *Fuzzy Sets and Systems*, 157(11):1455–1462, June 2006.
- [196] Ronald R. Yager and Dimitar Filev. On the issue of defuzzification and selection based on a fuzzy set. *Fuzzy Sets and Systems*, 55(3):255–271, May 1993.

- [197] O. Gordon. *An Evolutionary Methodology for Designing Fuzzy Rule-Based Systems*. PhD thesis, University of Granada, 1997.
- [198] Ebrahim H. Mamdani. Application of fuzzy algorithms for control of a simple dynamic plant. *Proceedings of the IEEE*, 121(12):1585–1588, December 1974.
- [199] Ebrahim H. Mamdani. Application of fuzzy logic to approximate reasoning using linguistic synthesis. *IEEE Transactions on Computers*, C-26(12):1182–1191, December 1977.
- [200] Yongsheng Ding, Hao Ying, and Shihuang Huang Shao. Necessary conditions on minimal system configuration for general MISO Mamdani fuzzy systems as universal approximators. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, 30(6):857–864, December 2000.
- [201] Puyin Liu. Mamdani fuzzy system: Universal approximator to a class of random processes. *IEEE Transactions on Fuzzy Systems*, 10(6):756–766, December 2002.
- [202] Irina G. Perfilieva and Vladik Ya Kreinovich. A new universal approximation result for fuzzy systems, which reflects cnf-dnf duality. *International Journal of Intelligent Systems*, 17(12):1121–1130, December 2002.
- [203] Hao Ying, Yongsheng Ding, Shaokuan Li, and Shihuang Shao. Typical takagi-sugeno and mamdani fuzzy systems as universal approximators: Necessary conditions and comparison. In *IEEE International Conference on Fuzzy Systems*, volume 1, pages 824–828, 1998.
- [204] Ke Zeng, Wenli Xu, and Naiyao Zhang. Universal approximation of special mamdani fuzzy systems. *Kongzhi yu Juece/Control and Decision*, 15(4):435–438, July 2000.
- [205] Bart Kosko. *Neural networks and fuzzy systems: a dynamical systems approach to machine intelligence*. Prentice–Hall, Inc., Upper Saddle River, NJ, USA, 1991.
- [206] Bart Kosko. *Fuzzy engineering*. Prentice–Hall, Inc., Upper Saddle River, NJ, USA, 1997.
- [207] Y. Tsukamoto. An approach to fuzzy reasoning model. In *Advances in fuzzy Set Theory and Applications*, Amsterdam, North-Holland, 1979.
- [208] T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, 15(1):116–132, 1985.
- [209] M. Sugeno and G. T. Kang. Structure identification of fuzzy model. *Fuzzy Sets and Systems*, 28(1):15–33, October 1988.
- [210] Bart Kosko. Fuzzy systems as universal approximators. *IEEE Transactions on Computers*, 43(11):1329–1333, November 1994.

- [211] Li-Xin Wang. Fuzzy systems are universal approximators. In *IEEE International Conference on Fuzzy Systems*, pages 1163–1170, San Diego, CA, USA, 1992.
- [212] Hao Ying. Sufficient conditions on uniform approximation of multivariate functions by general Takagi–Sugeno fuzzy systems with linear rule consequent. *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*, 28(4):515–520, July 1998.
- [213] Ke Zeng, Nai-Yao Zhang, and Wen-Li Xu. A comparative study on sufficient conditions for Takagi–Sugeno fuzzy systems as universal approximators. *IEEE Transactions on Fuzzy Systems*, 8(6):773–780, December 2000.
- [214] Vladik Kreinovich, Hung T. Hguyen, and Yeung Yam. Fuzzy systems are universal approximators for a smooth function and its derivatives. *International journal of Intelligent Systems*, 15(6):565–574, June 2000.
- [215] Vladik Kreinovich, C. G. Mouzouris, and Hung T. Hguyen. *Fuzzy rule based modeling as a universal approximation tool*. Fuzzy Systems: Modeling and Control. Kluwer Academic, 1998.
- [216] Arianna Mencattini, Marcello Salmeri, and Adelio Salsano. Sufficient conditions to impose derivative constraints on MISO Takagi–Sugeno fuzzy logic systems. *IEEE Transactions on Fuzzy Systems*, 13(4):454–467, August 2005.
- [217] Riccardo Rovatti. Fuzzy piecewise multilinear and piecewise linear systems as universal approximators in Sobolev norms. *IEEE Transactions on Fuzzy Systems*, 6(2):235–249, May 1998.
- [218] C. Fantuzzi and R. Rovatti. On the approximation capabilities of the homogeneous Takagi–Sugeno model. In *Proceedings of the 5th IEEE International Conference on Fuzzy Systems*, volume 2, pages 1067–1072, New Orleans, LA, September 1996.
- [219] Lennart Ljung and Torkel Glad. On global identifiability for arbitrary model parametrizations. *Automatica*, 30(2):265–276, February 1994.
- [220] H. K. Khalil. *Nonlinear systems*. Prentice-Hall, NJ, 2000.
- [221] H. J. Marquez. *Nonlinear control systems. Analysis and design*. John Wiley & Sons, Inc., 2003.
- [222] Henk Nijmeijer and Arjan van der Schaft. *Nonlinear dynamical control systems*. Springer Verlag, Berlin, 1990.
- [223] S. Sastry. *Nonlinear system: analysis, stability, and control*. Springer, New York, 1999.
- [224] Jean-Jacques E. Slotine and Weiping Li. *Applied nonlinear control*. Prentice-Hall, NJ, 1991.

- [225] Robert Babuška. Fuzzy modeling – a control engineering perspective. In *Proceedings of FUZZ-IEEE/IFES'95*, volume 4, pages 1897–1902, Yokohama, Japan, March 1995.
- [226] Robert Babuška and H. B. Verbruggen. A new identification method for linguistic fuzzy models. In *Proceedings of FUZZ-IEEE/IFES'95*, volume 4, pages 905–912, Yokohama, Japan, March 1995.
- [227] Hung T. Nguyen, Michio Sugeno, Richard M. Tong, and Ronald R. Yager. *Theoretical aspects of fuzzy control*. John Wiley Sons, 1995.
- [228] L. X. Wang. *Adaptive fuzzy systems and control*. Prentice Hall, New Jersey, 1994.
- [229] Li-Xin Wang. *A course in fuzzy systems and control*. Prentice Hall, New Yersey, USA, 1997.
- [230] L.K. Wong, F.H.F. Leung, and P.K.S. Tam. Stability design of TS model based fuzzy systems. In *IEEE International Conference on Fuzzy Systems*, volume 1, pages 83–86, Barcelona, Spain, July 1997.
- [231] José Manuel Andújar and Antonio Javier Barragán. Matriz jacobiana de un sistema de control borroso en lazo cerrado. Aplicación a la síntesis de sistemas de control borroso. In CEA-IFAC, editor, *XXIV Jornadas de Automática*, León, España, September 2003.
- [232] José Manuel Andújar, José Manuel Bravo, and Antonio Peregrín. Stability analysis and synthesis of multivariable fuzzy systems using interval arithmetic. *Fuzzy Sets and Systems*, 148(3):337–353, December 2004.
- [233] José Manuel Andújar and Antonio Javier Barragán. Una aproximación a la síntesis de sistemas de control borroso estables por diseño. In ISTANET, editor, *I Simposio de computación y sistemas inteligentes de la red ISTANET*, pages 303–322, Málaga, España, December 2004.
- [234] Oscar Cerdón and Francisco Herrera. A proposal for improving the accuracy of linguistic modeling. *IEEE Transactions on Fuzzy Systems*, 8(3):335–344, 2000.
- [235] Vilém Novák and Irina Perfilieva. On the semantics of perception-based fuzzy logic deduction. *International Journal of Intelligent Systems*, 19(11):1007–1031, November 2004.
- [236] Ester Van Broekhoven and Bernard De Baets. Monotone Mamdani–Assilian models under mean of maxima defuzzification. *Fuzzy Sets and Systems*, In Press.
- [237] S. Akkurt, G. Tayfur, and S. Can. Fuzzy logic model for the prediction of cement compressive strength. *Cement and Concrete Research*, 34(8):1429–1433, 2004.

- [238] H.A. Nefeslioglu, C. Gokceoglu, and H. Sonmez. A Mamdani model to predict the weighted joint density. In *Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science)*, volume 2773 PART 1, pages 1052–1057, July 2003.
- [239] SSL Chang and Lotfi A. Zadeh. On fuzzy mapping and control. *IEEE Computer*, SMC-2(1):30–34, January 1972.
- [240] L. X. Wang. *Adaptive fuzzy systems and control*. Prentice Hall, New Jersey, 2nd edition, 1997.
- [241] Jyh-Shing Roger Jang. ANFIS: adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(3):665–685, May 1993.
- [242] Jyh-Shing Roger Jang and Chuen-Tsai Sun. Neuro-fuzzy modeling and control. *Proceedings of the IEEE*, 83(3):378–406, March 1995.
- [243] Frank Hoffmann. Incremental tuning of fuzzy controllers by means of an evolution strategy. In John R. Koza, Wolfgang Banzhaf, Kumar Chellapilla, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max H. Garzon, David E. Goldberg, Hitoshi Iba, and Rick Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 843–851, University of Wisconsin, Madison, Wisconsin, USA, 1998. Morgan Kaufmann.
- [244] Frank Hoffmann, Daniel Schauten, and Sebastian Hölemann. Incremental evolutionary design of tsf fuzzy controllers. *IEEE Transactions on Fuzzy Systems*, 15(4):563–577, August 2007.
- [245] J.S.R. Jang, C.T. Sun, and E. Mizutani. Neuro-fuzzy and soft computing-A computational approach to learning and machine intelligence. *IEEE Transactions on Automatic Control*, 42(10):1482–1484, October 1997.
- [246] S. Chiu. Fuzzy model identification based on cluster estimation. In *Journal of Intelligent & Fuzzy Systems*, volume 2, pages 267–278, 1994.
- [247] I. Drezga and S. Rahman. Input variable selection for ann-based short-term load forecasting. *IEEE Transactions on Power Systems*, 13(4):1238–1244, November 1998.
- [248] Edward I. George. The variable selection problem. *Journal of the American Statistical Association*, 95(452):1304–1308, December 2000.
- [249] R.R. Hocking. The analysis and selection of variables in linear regression. *Biometrics*, 32(1):1–49, March 1976.
- [250] Hui Zou and Trevor J. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 67(2):301–320, March 2005.

- [251] N.R. Draper and H. Smith. *Applied Regression Analysis*. John Wiley & Sons, 2nd edition, 1981.
- [252] Ruck Thawonmas and Shigeo Abe. Function approximation based on fuzzy rules extracted from partitioned numerical data. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, 29(4):525–534, August 1999.
- [253] Shiqian Wu, Meng Joo Er, and Yang Gao. A fast approach for automatic generation of fuzzy rules by generalized dynamic fuzzy neural networks. *IEEE Transactions on Fuzzy Systems*, 9(4):578–594, August 2001.
- [254] José L. Díez, José L. Navarro, and Antonio Sala. Algoritmos de clustering en la identificación de modelos borrosos. In CEA-IFAC, editor, *XXIII Jornadas de Automática*, La Laguna, Santa Cruz de Tenerife, September 2002.
- [255] James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 1981.
- [256] Euntai Kim, Minkee Park, Seunghwan Ji, and Mignon Park. A new approach to fuzzy modeling. *IEEE Transactions on Fuzzy Systems*, 5(3):328–337, August 1997.
- [257] D.E. Gustafson and W.C. Kessel. Fuzzy clustering with fuzzy covariance matrix. In M.M. Gupta, R.K. Ragade, and R. R Yager, editors, *Advances in Fuzzy Set Theory and Applications*, pages 605–620, North-Holland, Amsterdam, 1979.
- [258] Tzung-Pei Hong and Chai-Ying Lee. Induction of fuzzy rules and membership functions from training examples. *Fuzzy Sets and Systems*, 84(1):33–47, June 1996.
- [259] Li-Xin Wang and Jerry M. Mendel. Generating fuzzy rules by learning from examples. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(6):1414–1427, November 1992.
- [260] Nikola Kirilov Kasabov and Qun Song. Denfis: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction. *IEEE Transactions on Fuzzy Systems*, 10(2):144–154, April 2002.
- [261] Isaac David Guedalia, Mickey London, and Michael Werman. An on-line agglomerative clustering method for nonstationary data. *Neural Computation*, 11(2):521–540, February 1999.
- [262] Boris Martínez, Francisco Herrera, Jesús Fernández, and Erick Marichal. Método de agrupamiento en línea para la identificación e modelos borrosos Takagi-Sugeno. *Revista Iberoamericana de Automática e Informática Industrial (RIAI)*, 5(3):63–69, July 2008.
- [263] John Yen and Nathan Pfluger. A fuzzy logic based extension to payton and rosenblatt’s command fusion method for mobile robot navigation. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(6):971–978, June 1995.

- [264] Robert Babuška, Magne Setnes, Uzay Kaymak, and Hans R. van Nauta Lemke. Rule base simplification with similarity measures. In *Proceedings of the 5th IEEE International Conference on Fuzzy Systems*, volume 3, pages 1642–1647, New Orleans, LA, September 1996.
- [265] P. Dewilde and E. Deprettere. *Singular value decomposition: An introduction*. SVD and signal processing: Algorithms, applications and architectures. Elsevier Science Publishers, 1988.
- [266] Magne Setnes, Robert Babuška, Uzay Kaymak, and Hans R. van Nauta Lemke. Similarity measures in fuzzy rule base simplification. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, 28(3):376–386, June 1998.
- [267] Yeung Yam, Peter Baranyi, and Chi-Tin Yang. Reduction of fuzzy rule base via singular value decomposition. *IEEE Transactions on Fuzzy Systems*, 7(2):120–132, April 1999.
- [268] Shin-ichi Horikawa, Takeshi Furuhashi, and Yoshiki Uchikawa. On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm. *IEEE Transactions on Neural Networks*, 3(5):801–806, September 1992.
- [269] Magne Setnes and H. Roubos. GA-fuzzy modeling and classification: complexity and performance. *IEEE Transactions on Fuzzy Systems*, 8(5):509–522, October 2000.
- [270] José Manuel Andújar, Javier Aroba, María Luisa de la Torre, and José Antonio Grande. Contrast of evolution models for agricultural contaminants in ground waters by means of fuzzy logic and data mining. *Environmental Geology*, 49(3):458–466, January 2006.
- [271] José Manuel Andújar and José Manuel Bravo. Multivariable fuzzy control applied to the physical-chemical treatment facility of a cellulose factory. *Fuzzy Sets and Systems*, 150(3):475–492, March 2005.
- [272] José Manuel Andújar, Juan Manuel Córdoba, and Iñaki Fernández. Una técnica híbrida de modelización neuroborrosa mediante colonia de hormigas. In CEA-IFAC, editor, *XXV Jornadas de Automática*, Ciudad Real, España, September 2004.
- [273] Hiroyoshi Nomura, Isao Hayashi, and Noboru Wakami. A self-tuning method of fuzzy inference rules by descent method. In R. Lowen and M. Roubens, editors, *Fuzzy Logic*, pages 465–475. Kluwer Academic Publishers, Dordrecht, 1993.
- [274] Bernard Widrow and Marcian E. Hoff. Adaptive switching circuits. In *Wescon Conference Record*, pages 709–717, 1989.
- [275] James A. Freeman and David M. Skapura. *Neural networks: algorithms, applications, and programming techniques*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1991.

- [276] Jerzy Mościński and Zbigniew Ogonowski. *Advanced control with MATLAB and SIMULINK*. Ellis Horwood, Upper Saddle River, NJ, USA, 1995.
- [277] Marco Dorigo, Gianni Di Caro, and Luca Maria Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5(2):137–172, 1999.
- [278] Marco Dorigo, Vittorio Maniezzo, and Alberto Coloni. Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, 26(1):29–41, February 1996.
- [279] Lennart Ljung. *System identification: Theory for the user*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1987.
- [280] M. Stone. Cross-validity choice and assessment of statistical predictions. *Journal of the Royal Statistical Society*, 36(B):111–133, 1974.
- [281] S. J. Farlow. *The GMDH algorithm*, chapter Self-organizing methods in modeling: GMDH type algorithms, pages 1–24. Marcel Dekker, 1984.
- [282] Michio Sugeno and Takahiro Yasukawa. Fuzzy-logic-based approach to qualitative modeling. *IEEE Transactions on Fuzzy Systems*, 1(1):7–31, February 1993.
- [283] John Yen and Liang Wang. Application of statistical information criteria for optimal fuzzy model construction. *IEEE Transactions on Fuzzy Systems*, 6(3):362–372, August 1998.
- [284] John Yen and Wayne Gillespie. Integrating global and local evaluations for fuzzy model identification using genetic algorithms. In *Sixth World Congress of Int. Fuzzy Systems Association (IFSA 95)*, San Pablo, Brasil, 1995.
- [285] The MathWorks, Inc. *MATLAB external interfaces*. 2005.
- [286] The MathWorks, Inc. *MATLAB external interfaces reference*. 2005.
- [287] The MathWorks, Inc. *Fuzzy Logic Toolbox 2. User's guide*. 2007.
- [288] Grady Booch, James Rumbaugh, and Ivar Jacobson. *Unified Modeling Language User Guide*. Addison–Wesley, 1998.
- [289] Stephen Wiggins. *Introduction to Applied Nonlinear Dynamical Systems and Chaos*. Texts in Applied Mathematics. Springer, 2nd edition, October 2003.
- [290] José Manuel Andújar and Antonio Javier Barragán. Algoritmo para el cálculo de la matriz jacobiana de un sistema de control borroso en lazo cerrado. In EUSFLAT, editor, *XII Congreso español sobre tecnologías y lógica fuzzy*, pages 101–106, Jaén, España, September 2004.

- [291] José Manuel Andújar, Antonio Javier Barragán, Manuel Emilio Gegúndez, and Francisco Jesús de Toro. Aplicación de la matriz jacobiana de un sistema de control borroso a la obtención de sus estados de equilibrio. In CEA-IFAC, editor, *XXV Jornadas de Automática*, page 10, Ciudad Real, España, September 2004.
- [292] J. Holgado, A. Ollero, and J. Aracil. Statistical study on stability indices of fuzzy control systems. In *Proceedings of the 3rd IEEE Conference on Fuzzy Systems. IEEE World Congress on Computational Intelligence*, volume 2, pages 936–941, Orlando, FL, USA, June 1994.
- [293] José Manuel Andújar and Antonio Javier Barragán. Una metodología de diseño de sistemas de control borroso. In CEA-IFAC, editor, *I Simposio sobre control inteligente*, Huelva, España, June 2005.
- [294] José Manuel Andújar, Antonio Javier Barragán, Manuel Emilio Gegúndez, and Manuel Maestre. *Una metodología para el diseño de sistemas de control borroso estables (Nuevas tendencias en control inteligente)*. CEA-IFAC, 31–37, 2006.
- [295] J.E. Dennis Jr. *Nonlinear least-squares*, pages 269–312. State of the art in numerical analysis. Academic Press, 1977.
- [296] J.E. Dennis Jr. and Robert B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, volume 16 of *Classics in Applied Mathematics*. Soc for Industrial & Applied Math, 1996.
- [297] L. Grippo, F. Lampariello, and S. Lucidi. A nonmonotone line search technique for newton’s method. *SIAM Journal on Numerical Analysis*, 23(4):707–716, 1986.
- [298] J.P. Bulteau and J.P. Vial. Curvilinear path and trust region in unconstrained optimization: a convergence analysis. *Mathematical Programming Study*, 1:82–101, February 1983.
- [299] M.J.D. Powell. On the global convergence of trust region algorithms for unconstrained minimization. *Mathematical Programming*, 29(3):297–303, August 1984.
- [300] K. Levenberg. A method for the solution of certain problems in least squares. In *Quart. Appl. Math.*, volume 2, pages 164–168, 1944.
- [301] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM J. Appl. Math.*, 11:431–441, 1963.
- [302] Jorge J. Moré. The Levenberg-Marquardt algorithm: Implementation and theory. In G.A. Watson, editor, *Numerical Analysis*, pages 105–116. Springer Verlag, Berlin, 1977.
- [303] Andrew R. Conn, Nicholas I.M. Gould, and Philippe L. Toint. *Trust-region methods*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

- [304] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer Verlag, 2nd edition, 2006.
- [305] D.F. Shanno. Conditioning of quasi-Newton methods for function minimization. *Mathematics of Computing*, 24(111):647–656, July 1970.
- [306] Lofti A. Zadeh. The evolution of systems analysis and control: A personal perspective. *IEEE Control Systems Magazine*, 16(3):95–98, July 1996.
- [307] Andrzej Piegat. *Fuzzy modeling and control*. Springer-Verlag Company, Heidelberg-New York, 2001.
- [308] Y. Li and Y. Yonezawa. Stability analysis of fuzzy control systems by the hyperstability theorem. *Japanese journal of fuzzy theory and systems*, 3(2):209–214, 1991.
- [309] Andrzej Piegat. Hyperstability of fuzzy-control systems and degrees of freedom. In *5th European Congress on Intelligent Techniques and Soft Computing, EUFIT'97*, volume 2, pages 1446–1450, 1997.
- [310] Liang Wang and Reza Langari. Fuzzy controller design via hyperstability approach. In *IEEE International Conference on Fuzzy Systems*, volume 1, pages 178–182, 1994.
- [311] Ming Jian Zhao, Ying Yang, and Jing Zhu. Design of fuzzy adaptive controller based on Popov's hyperstability theory. *Zidonghua Xuebao/Acta Automatica Sinica*, 27(3):406–410, May 2001.
- [312] H.-R. Lin and W.-J. Wang. L_2 -stabilization design for fuzzy control systems. *Fuzzy Sets and Systems*, 100(1-3):159–172, November 1998.
- [313] G. Feng, S. G. Cao, and N. W. Rees. Stable adaptive control of fuzzy dynamic systems. *Fuzzy Sets and Systems*, 131(2):217–224, October 2002.
- [314] C. I. Siettos and G. V. Bafas. Semiglobal stabilization of nonlinear systems using fuzzy control and singular perturbation methods. *Fuzzy Sets and Systems*, 129(3):275–294, August 2002.
- [315] Q. Sun, R. Li, and P. Zhang. Stable and optimal adaptive fuzzy control of complex systems using fuzzy dynamic model. *Fuzzy Sets and Systems*, 133(1):1–17, January 2003.
- [316] Kazuo Tanaka and Michio Sugeno. Stability analysis and design of fuzzy control systems. *Fuzzy Sets and Systems*, 45(2):135–156, January 1992.
- [317] S. Tong and H.-H. Li. Observer-based robust fuzzy control of nonlinear systems with parametric uncertainties. *Fuzzy Sets and Systems*, 131(2):165–184, October 2002.
- [318] M. Maeda and S. Murakami. Stability analysis of fuzzy control systems using phase planes. In *Japanese journal of fuzzy theory and systems*, volume 3, pages 149–160, 1991.

- [319] Javier Aracil. *Stability Issues in Fuzzy Control*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2000.
- [320] Javier Aracil, A. García, and Anibal Ollero. Fuzzy control of dynamical systems. Stability analysis based on the conicity criterion. In *4th International Fuzzy Systems Association Congress (IFSA)*, pages 5–8, Brussels, Belgium, July 1991.
- [321] Heidar A. Malki, Huaidong Li, and Guanrong Chen. New design and stability analysis of fuzzy proportional-derivative control systems. *IEEE Transactions on Fuzzy Systems*, 2(4):245–254, November 1994.
- [322] Hua O. Wang, Kazuo Tanaka, and Michael F. Griffin. An approach to fuzzy control of nonlinear systems: stability and design issues. *IEEE Transactions on Fuzzy Systems*, 4(1):14–23, February 1996.
- [323] Hao Ying. Practical design of nonlinear fuzzy controllers with stability analysis for regulating processes with unknown mathematical models. *Automatica*, 30(7):1185–1195, July 1994.
- [324] W. Assawinchaichote and Sing Kiong Nguang. H_∞ fuzzy control design for nonlinear singularly perturbed systems with pole placement constraints: an LMI approach. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, 34(1):579–588, February 2004.
- [325] Thierry Marie Guerra and Laurent Vermeiren. Lmi-based relaxed nonquadratic stabilization conditions for nonlinear systems in the Takagi-Sugeno’s form. *Automatica*, 40(5):823–829, May 2004.
- [326] Sung Kyung Hong and Yoonsu Nam. Stable fuzzy control system design with pole-placement constraint: an LMI approach. *Computers in Industry*, 51(1):1–11, May 2003.
- [327] H. K. Lam, F. M. Leung, and P. K. S. Tam. A linear matrix inequality approach for the control of uncertain fuzzy systems. *IEEE Control Systems Magazine*, 22(4):20–25, August 2002.
- [328] H.K. Lam, F.H.F. Leung, and P.K.S. Tam. An improved stability analysis and design of fuzzy control systems. In *Fuzzy Systems Conference Proceedings, 1999. FUZZ-IEEE '99. 1999 IEEE International*, volume 1, pages 430–433, August 1999.
- [329] K. Tanaka, T. Hori, and H.O. Wang. A fuzzy lyapunov approach to fuzzy control system design. In *American Control Conference, 2001. Proceedings of the 2001*, volume 6, pages 4790–4795, June 2001.
- [330] K. Tanaka, H. Ohtake, and H. O. Wang. A descriptor system approach to fuzzy control system design via fuzzy Lyapunov functions. *IEEE Transactions on Fuzzy Systems*, 15(3):333–341, June 2007.

- [331] Kazuo Tanaka, Takayuki Ikeda, and Hua O. Wang. Fuzzy regulators and fuzzy observers: relaxed stability conditions and LMI-based designs. *IEEE Transactions on Fuzzy Systems*, 6(2):250–265, May 1998.
- [332] Zhi-Hong Xiu and Guang Ren. Stability analysis and systematic design of Takagi-Sugeno fuzzy control systems. *Fuzzy Sets and Systems*, 151(1):119–138, April 2005.
- [333] Jorge Luis Martínez Rodríguez. *Control de procesos industriales*. Málaga, España, 3rd edition, September 2002.
- [334] Jian Cao, Haiyan Zhao, and Minglu L. Li. A fuzzy rule based load balancing model for a distributed service process engine. In *3rd International Conference on Grid and Pervasive Computing Symposia/Workshops, GPC 2008*, pages 9–14, 2008.
- [335] M. Margaliot and G. Langholz. *New approaches to fuzzy modeling and control: design and analysis*. Machine Perception Artificial Intelligence. World Scientific, Singapore, 2000.
- [336] José Manuel Andújar, Antonio Javier Barragán, Manuel Emilio Gegúndez, and Manuel Maestre. Aplicación de la síntesis de Lyapunov borrosa a un caso práctico real: la grúa porta contenedores. In CEA-IFAC, editor, *XXVII Jornadas de automática*, pages 1069–1074, Almería, España, September 2006.
- [337] Kazuo Tanaka. Stability and stabilizability of fuzzy–neural–linear control systems. *IEEE Transactions on Fuzzy Systems*, 3(4):438–447, November 1995.
- [338] Min-You Chen and D. A. Linkens. Rule-base self-generation and simplification for data-driven fuzzy models. *Fuzzy Sets and Systems*, 142(2):243–265, March 2004.
- [339] Tadanari Taniguchi, Kazuo Tanaka, Hiroshi Ohtake, and Hua O. Wang. Model construction, rule reduction, and robust compensation for generalized form of Takagi-Sugeno fuzzy systems. *IEEE Transactions on Fuzzy Systems*, 9(4):525–538, August 2001.
- [340] Tor A. Johansen. Computation of Lyapunov functions for smooth nonlinear systems using convex optimization. *Automatica*, 36(11):1617–1626, November 2000.
- [341] David E. Goldberg. *Genetic algorithms in search, optimization & machine learning*. Addison-Wesley, 1989.
- [342] J. A. Nelder and R. A. Mead. A simplex method for function minimization. *Computer Journal*, 7(4):308–313, 1965.
- [343] Y. Sakawa and Y. Shindo. Optimal control of container cranes. *Automatica*, 18(3):257–266, 1982.
- [344] Krzysztof Socha and Marco Dorigo. Ant colony optimization for continuous domains. *European Journal of Operational Research*, 185(3):1155–1173, August 2008.

- [345] Wei Dong Gao. New continuous ant colony algorithm. In *Proceedings of the World Congress on Intelligent Control and Automation (WCICA)*, pages 1275–1279, June 2008.
- [346] Qin Zhang and Xiong H. Wang. Binary and continuous ant colony algorithms research for solving continuous global optimization problem. In *Proceedings - The 1st International Conference on Intelligent Networks and Intelligent Systems, ICINIS 2008*, pages 1–4, November 2008.
- [347] Yuntao Zhao, Jing Xian Wang, and Xinliang Xie. Continuous ant colony algorithm based on entity and its convergence. In *Proceedings - 2008 2nd International Symposium on Intelligent Information Technology Application, IITA 2008*, volume 1, pages 80–84, December 2008.
- [348] Ka-Ling Fok, Tien-Tsin Wong, and Man-Leung Wong. Evolutionary computing on consumer graphics hardware. *IEEE Intelligent Systems*, 22(2):69–78, April 2007.
- [349] Man Leung Wong and Tien Tsin Wong. *Intelligent and Evolutionary Systems*, volume 187 of *Studies in Computational Intelligence*, chapter Implementation of Parallel Genetic Algorithms on Graphics Processing Units, pages 197–216. SpringerLink, 2009.
- [350] Pedro Albertos and Antonio Sala. Control borroso. Una metodología integradora. *Revista Iberoamericana de Automática e Informática Industrial (RIAI)*, 1(2):22–31, July 2004.
- [351] Ching-Hung Lee and Hung-Yi Pan. Performance enhancement for neural fuzzy systems using asymmetric membership functions. *Fuzzy Sets and Systems*, 160(7):949–971, April 2009.
- [352] William H. Press. *Numerical recipes in C: the art of scientific computing*, chapter 2, pages 96–97. Cambridge University Press, Cambridge, England, 2nd edition, 1998.