UNIVERSITY OF GRANADA

Department of Computer Architecture and Computer Technology

PhD Thesis Dissertation:

New Methodologies for the Design of Evolving Fuzzy Systems for Online Intelligent Control

by

Ana Belén Cara Carmona

Advisors:

Ignacio Rojas Ruiz

Héctor Pomares Cintas

Miguel Damas Hermoso

Granada, February 2012

Editor: Editorial de la Universidad de Granada
Autor: Ana Belén Cara Carmona
D.L.: GR 2183-2012
ISBN: 978-84-9028-120-8

UNIVERSITY OF GRANADA

New Methodologies for the Design of Evolving Fuzzy Systems for Online Intelligent Control

(Nuevas metodologías para el diseño de sistemas difusos auto-organizativos para control inteligente en tiempo real)

Dissertation presented by:
Ana Belén Cara Carmona

To apply for the:

European PhD degree in Computer Science

Signed. Ana Belén Cara Carmona

D. Ignacio Rojas Ruiz, D. Héctor Pomares Cintas y D. Miguel Damas Hermoso, Catedrático y Profesores Titulares de Universidad respectivamente del Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada

CERTIFICAN

Que la memoria titulada ”New Methodologies for the Design of Evolving Fuzzy Systems for Online Intelligent Control” ha sido realizada por D^{a}. Ana Belén Cara Carmona, bajo nuestra dirección en el Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada para optar al grado de Doctor Europeo en Ingeniería Informática.

Granada, a 17 de Febrero de 2012

Fdo. Ignacio Rojas Ruiz, Héctor Pomares Cintas y Miguel Damas Hermoso
Directores de la Tesis

Contents

List of Figures

List of Tables

List of Abbreviations

ANFIS	Adaptive Neuro-Fuzzy Inference System
ANOVA	ANalysis Of VAriances
DB	Data Base
DOF	Degree Of Freedom
EA	Evolutionary Algorithm
EFS	Evolving Fuzzy System
EIS	Evolving Intelligent System
FLC	Fuzzy Logic Controller
FLS	Fuzzy Logic System
FOU	Footprint of Uncertainty
GFS	Genetic Fuzzy System
GL	Global Learning
GMP	Generalized Modus Ponens
IR	Index of Responsibility
KB	Knowledge Base
KM	Karnik-Mendel
KNN	K-Nearest Neighbors

LL	Local Learning
MF	Membership Function
MISO	Multiple-Input Single-Output
MOEA	Multi-Objective Evolutionary Algorithm
MOOP	Multi-Objective Optimization Problem
MSE	Mean Square Error
NRMSE	Normalized Root-Mean-Square Error
NS	Non-Singleton
NSGA-II	Nondominated Sorting Genetic Algorithm II
OSEFC	Online Self-Evolving Fuzzy Controller
OSENF-TaSe	Online Self-Evolving Neuro-Fuzzy controller based on the TaSe-NF model
RB	Rule Base
RBF	Radial Basis Function
RBFNN	Radial Basis Function Neural Network
SISO	Single-Input Single-Output
SPEA2	Strength Pareto Evolutionary Algorithm 2
SSE	Sum of Square Errors
SVD	Singular Value Decomposition
TaSe-NF	Taylor Series Neuro-Fuzzy model
TSK	Takagi-Sugeno-Kang (fuzzy system)
UF	Uncertainty Factor

A mi Juanlu, por enseñarme que hay cosas que no se pueden controlar.
Y por hacer que me alegre de ello.

Abstract

The work presented in this dissertation is a contribution to the areas of automatic design of fuzzy systems and intelligent control. Specifically, this dissertation focuses in the automatic design of fuzzy controllers with less prior knowledge about the plants. Two new methodologies for online self-evolving fuzzy controllers are proposed. These methods are able to design fuzzy controllers from scratch, in an online manner, based on the analysis of the input/output data obtained from the normal operation of the plant. They do not use the information regarding the differential equations governing the system, or make any assumptions about them. The evolution of the controller is achieved through a life-long learning process that combines the adaptation of the rule consequents and the addition of new membership functions and rules. Due to their adaptive nature, these controllers are robust against unexpected changes in the plants and perform well in noisy environments. Moreover, the first methodology is able to automatically select the relevant control inputs, whilst the second tackles the curse of dimensionality and improves the controller's intrepretability. Simulation and experimental results illustrate the capabilities of the proposed methods.

In addition, we study the increasingly popular type-2 fuzzy logic systems (FLSs), which are credited to outperform traditional (type-1) FLSs in the presence of uncertainties. We propose a multi-objective evolutionary algorithm for learning the structure and parameters of type-1 and type-2 fuzzy systems. This method is applied to obtain a common framework for the comparison of type-1 and type-2 FLSs in uncertain environments. Our final aim is to investigate whether the better performance of type-2 FLSs is solely based on the use of extra parameters (as it is often criticized), or whether it is due to the use of an essentially different mechanism for uncertainty handling.

Resumen

El trabajo presentado en esta tesis doctoral representa una contribución a los ámbitos del aprendizaje automático de sistemas difusos y del control inteligente. En concreto, la tesis se centra en el diseño automático de controladores difusos sin conocimiento previo sobre las plantas a controlar. Se proponen dos nuevas metodologías para la auto-organización de controladores en tiempo real que son capaces de diseñar controladores difusos desde cero, basándose en la información de entrada/salida obtenida del propio funcionamiento de la planta. No se utiliza información referente a las ecuaciones diferenciales que rigen el funcionamiento de las plantas, ni se hacen hipótesis sobre las mismas. La evolución del controlador se consigue gracias a un proceso de aprendizaje continuado que combina la adaptación de los consecuentes de las reglas con la adición de nuevas funciones de pertenencia y reglas. Gracias a su naturaleza adaptativa, los métodos propuestos son robustos frente a cambios inesperados en la operación de las plantas y presentan un buen funcionamiento en entornos ruidosos. Es más, el primer método es capaz de seleccionar las entradas de control que son relevantes para el proceso, mientras que el segundo afronta el problema de la maldición de la dimensionalidad a la vez que mejora la interpretabilidad de las reglas. Los resultados experimentales muestran las capacidades de los métodos propuestos.

Adicionalmente, se estudian los cada vez más populares sistemas difusos de tipo 2, a los que se atribuye una mejor respuesta en presencia de incertidumbre que a los sistemas difusos tradicionales (o de tipo 1). En este trabajo se propone un algoritmo evolutivo multi-objetivo para el aprendizaje de la estructura y los parámetros de sistemas difusos de tipo 1 y 2. Este método se utiliza para proporcionar un entorno adecuado para la comparación, en presencia de incertidumbre, de ambos tipos de sistema. El objetivo final de esta comparación es investigar la cuestión de si, tal y como se argumenta habitualmente, el mejor rendimiento de los sistemas difusos de tipo 2 se debe únicamente al uso de un mayor número de parámetros, o si, por el contrario, se debe al uso de un mecanismo diferente para manejar la incertidumbre.

Introducción

Esta introducción es una versión en español del capítulo 1 y ha sido incluida para cumplir los requisitos necesarios para optar a la mención de *Doctorado Europeo*. El resto de esta memoria, a excepción de las conclusiones, se presenta en inglés.

Antecedentes

El término “control convencional (o tradicional)” se utiliza en la actualidad para referirse a las teorías y métodos desarrollados durante las últimas décadas del siglo XX para controlar sistemas dinámicos[1]. Generalmente, estas técnicas se basan en el uso de ecuaciones diferenciales y ecuaciones en diferencias [Antsaklis, 1994]. Sin embargo, en ciertos casos este marco matemático puede no ser lo suficientemente general, pues existen problemas de control que no se pueden describir adecuadamente en términos de ecuaciones diferenciales [Gupta, 2000].

En 1993, el Grupo de Trabajo en Control Inteligente de la Sociedad de Sistemas de Control del IEEE (Task Force on Intelligent Control of the IEEE Control Systems Society, en inglés) definió el término “control inteligente” como la disciplina que aborda el desarrollo de métodos de control que pretenden emular ciertas características de la inteligencia humana [Antsaklis, 1994; Antsaklis and Passino, 1993]. Dichas características incluyen adaptación, aprendizaje y planificación en presencia de incertidumbre, entre otras. Por tanto, se puede considerar que el control inteligente trata de ampliar y mejorar las metodologías de control

[1]Estas metodologías constituyen la tradicionalmente llamada “teoría de control moderna”, que se desarrolló principalmente entre la década de los 60 y la década de los 80.

convencionales para resolver nuevos problemas que presentan un mayor nivel de exigencia.

Para controlar satisfactoriamente sistemas complejos a lo largo del tiempo, con frecuencia es necesario hacer frente a diferentes tipos de incertidumbre que pueden no ser manejados adecuadamente por los métodos de control robustos y adaptativos tradicionales [Chen and Narendra, 2003; Linkens and Nyongesa, 1996]. Para ello, el diagnóstico de fallos, la capacidad de reconfiguración, la adaptación y el aprendizaje son elementos fundamentales.

En cierto sentido, los problemas de control abordados en el área del control inteligente pueden ser vistos como versiones ampliadas, más ambiciosas y generales, de los problemas planteados en el control tradicional [Antsaklis, 1994; Gupta, 2000]. Es más, estos problemas presentan requisitos adicionales que los métodos tradicionales no pueden satisfacer. Como consecuencia, el control inteligente se convierte en un ámbito interdisciplinar en el que se combinan métodos y teorías de áreas diversas, como son el control, las ciencias computacionales y la investigación de operaciones.

Merece la pena aclarar que, en general, las metodologías desarrolladas en estos campos no pueden aplicarse directamente para resolver problemas de control [Antsaklis, 1994]. Esto no significa que no sean métodos válidos, sino que fueron concebidos para cubrir necesidades diferentes. Por ejemplo, el problema del diseño de un controlador puede ser visto como un problema de aproximación funcional. Sin embargo, los métodos comúnmente utilizados para este tipo de problema no se pueden aplicar directamente en el ámbito del control, pues los requisitos y condiciones de operación son diferentes. Por tanto, se plantea la necesidad de mejorar las técnicas existentes y de derivar nuevas metodologías en combinación con ellas. Más aún, incluso puede ser necesario revisar y redefinir conceptos básicos como el de estabilidad, para adaptarlos a las peculiaridades de los nuevos problemas [Antsaklis, 1994].

Por otra parte, en ocasiones también se utiliza el término "control autónomo" para referirse a la disciplina que estamos considerando [Pachter and Chandler, 1998]. Esta denominación realza el hecho de que los controladores inteligentes pretenden, en general, alcanzar altos grados de operación autónoma. Es obvio que la inteligencia es fundamental tanto para responder de manera adecuada a los cambios en el entorno como para alcanzar un alto grado de autonomía. Por tanto, se puede decir que autonomía e inteligencia son nociones complementarias en el ámbito del control.

En resumen, un sistema de control inteligente debe contar con una alta capacidad de adaptación y autonomía frente a cambios imprevistos [Antsaklis, 1994]. Para alcanzar estos objetivos, el aprendizaje es un factor fundamental. Inicialmente, los métodos de aprendizaje para control se plantearon como un medio para mejorar el bajo rendimiento causado por las deficiencias del modelado de sistemas no lineales, ya que permitían incorporar la experiencia obtenida a través de la interacción directa con la propia planta [Gupta, 2000]. Sin embargo, en la actualidad su uso se ha extendido a un rango más amplio, que incluye el manejo de incertidumbres ambientales, de problemas de desgaste en los actuadores y mecanismos, etc. [Angelov, 2004; Doctor et al., 2005a; Schaal and Atkeson, 2010].

En las últimas dos décadas ha aparecido una gran variedad de paradigmas y herramientas para control inteligente, siendo las redes neuronales, los sistemas difusos y los algoritmos genéticos algunos de los ejemplos más populares [Gupta, 2000; Zilouchian and Jamshidi, 2000]. Sin embargo, hay que recalcar que el control inteligente no se limita al uso de estas técnicas. Es más, de acuerdo con las definiciones anteriores, no todos los controladores difusos o neuronales pueden ser considerados "inteligentes" [Antsaklis, 1994]. Para que así sea, deben contar con las capacidades de autonomía y adaptación ya mencionadas, lo que requiere que sean extendidos mediante métodos de aprendizaje y auto-organización.

Con el objetivo de hacer frente a la aparición de nuevas condiciones de operación, a dinámicas cambiantes o a otras influencias ambientales, en la última década ha surgido un nuevo campo de investigación: los denominados sistemas inteligentes auto-organizativos[1] (*evolving intelligent systems*, o simplemente *evolving systems*, en inglés), de los cuales los sistemas difusos auto-organizativos (o

[1]A pesar de que los primeros estudios relativos a los denominados (en inglés) *evolving systems* datan del año 2000, no existe aún una traducción al español ampliamente aceptada en el ámbito científico. En esta tesis se ha optado por utilizar el término *sistema auto-organizativo*, que es una de las traducciones más comunes. Asimismo, no se debe confundir *evolving* con *evolutivo* (*evolutionary*, en inglés), pues ambos términos se refieren a paradigmas de aprendizaje diferentes.

evolving fuzzy systems, en inglés) son un caso particular [Angelov et al., 2010; Lughofer, 2011]. Los sistemas inteligentes auto-organizativos combinan la capacidad de interpolación y la flexibilidad de los sistemas difusos y neuro-difusos con la capacidad adaptativa de las técnicas de aprendizaje dinámico. En el caso de los sistemas difusos auto-organizativos se cuenta con la ventaja adicional de su interpretabilidad lingüística. La característica más importante de estos sistemas es que son capaces de adaptarse, extenderse y evolucionar automáticamente, bajo demanda. Así, un sistema auto-organizativo presenta la capacidad de expandir o contraer su estructura, así como de adaptar sus parámetros, de forma incremental, dinámica y, si es necesario, en tiempo real.

Los sistemas auto-organizativos son capaces de soportar escenarios en los que los datos son dinámicos y pueden cambiar su naturaleza y características a lo largo del tiempo y el espacio. Es más, estos sistemas son capaces de continuar adaptando su conocimiento de manera permanente para integrar nuevas conductas y circunstancias ambientales, en un proceso de aprendizaje continuado que está siempre presente. Por tanto, representan una contribución importante al ámbito de la inteligencia computacional y artificial [Angelov and Kasabov, 2005; Lughofer, 2011].

A partir de esta presentación se puede deducir que las propiedades de los sistemas auto-organizativos los convierten en herramientas adecuadas para el desarrollo de controladores inteligentes. Es más, la capacidad de desarrollar su propia estructura y parámetros puede ser vista como la habilidad de determinar una política de control con un objetivo determinado, lo cual representa un intento por parte del controlador de organizar su "conocimiento" sobre su propia conducta interna. Esta capacidad puede ser considerada como un atributo de "inteligencia", por lo que se puede concluir que un controlador difuso auto-organizativo capaz de auto-diseñarse es un sistema de control inteligente.

Aportaciones de la tesis

Las aportaciones de esta tesis pretenden contribuir al avance en el ámbito de los sistemas difusos auto-organizativos para control inteligente. El principal objetivo es investigar y desarrollar nuevas metodologías capaces de producir controladores eficaces cuando no existe conocimiento previo sobre los sistemas a controlar. Adicionalmente, otra característica deseable en estos controladores es la capacidad de manejar cambios en las condiciones de operación, ruido y otros tipos de incertidumbre que pueden aparecer durante su operación.

En la primera parte de esta tesis se proponen dos nuevas metodologías para la auto-organización de controladores difusos en tiempo real, cuando no existe información previa sobre la planta. Dichos métodos son capaces de construir controladores difusos desde cero, basándose en la información de entrada/salida obtenida del propio funcionamiento de la planta. Los principios subyacentes son los mismos en ambos casos: por una parte, se hace uso de la propiedad de aproximación universal de los sistemas difusos para construir el controlador incrementalmente, a través de la adición de nuevas funciones de pertenencia y reglas. Por otra parte, para proporcionar mayor robustez al método de aprendizaje, la modificación de la estructura del controlador se basa en el análisis de la superficie de error a lo largo de todo el rango de operación del sistema. Algunas de las propiedades comunes a ambas metodologías son las siguientes:

- *Los métodos propuestos no requieren un modelo de la planta ni sus ecuaciones diferenciales*. Es más, al contrario que la mayoría de los métodos propuestos en la literatura [Gao and Er, 2003; Park et al., 2005; Phan and Gale, 2008], tampoco realizan hipótesis sobre la forma o propiedades de dichas ecuaciones (por ejemplo, sus límites). El único requisito es que la monotonía de la planta con respecto a la señal de control tenga signo constante. En otras palabras, tan sólo se precisa cierto conocimiento cualitativo sobre la planta.

- *No se requiere conocimiento previo sobre la política de control*. Tal y como se ha mencionado, el controlador puede empezar a funcionar con una única regla (inicializada a cero) y se auto-desarrollará en tiempo real, mientras controla la planta.

- *Los controladores desarrollados son robustos*, en el sentido de que pueden manejar cambios imprevistos en la dinámica de la planta. Su naturaleza adaptativa les dota de la capacidad de aprender el nuevo comportamiento de la planta para así continuar operando satisfactoriamente.

- *El método auto-organizativo es un proceso continuado de aprendizaje*. Esto significa que el controlador continúa aprendiendo y evolucionando durante toda su operación. En concreto, se aplican dos tipos de aprendizaje: por una parte, la información referente al error en la salida de la planta se utiliza para adaptar los consecuentes de las reglas de forma dinámica. Por otra parte, se analiza el error en la aproximación de la función inversa de la planta para desarrollar incrementalmente la estructura del controlador.

La primera metodología propuesta es OSEFC (por las siglas de *Online Self-Evolving Fuzzy Controller*, en inglés) [Cara et al., 2011a]. Para modificar la estructura del controlador, OSEFC analiza la superficie de error en todo el rango de operación del sistema. Seguidamente, se añade una nueva función de pertenencia a la variable que es mayormente responsable de dicho error. En este método se utiliza un conjunto completo de reglas; por ello, la adición de una función de pertenencia conlleva la creación de todas las reglas que la usan como antecedente. Esta metodología cuenta con la capacidad adicional de poder seleccionar automáticamente las variables de entrada que son más relevantes para el proceso de control, de entre un conjunto de candidatas. De este modo, se reduce la cantidad de información previa necesaria. OSEFC proporciona una buena política de control, a pesar de no utilizar ningún conocimiento previo sobre la planta. Sin embargo, su configuración "en rejilla" provoca el crecimiento exponencial del número de reglas (fenómeno conocido como la maldición de la dimensionalidad [Bellman, 1961]), lo que a su vez influye negativamente en la complejidad e interpretabilidad del controlador.

Con el objetivo de abordar estos inconvenientes, se propone una segunda metodología, que denominamos OSENF-TaSe (por las siglas de *Online Self-Evolving Neuro-Fuzzy controller based on the TaSe-NF model*, en inglés) [Cara et al., 2012]. En OSENF-TaSe se sustituye la distribución en rejilla de las reglas por una distribución "en clustering". De este modo, las reglas se pueden ubicar libremente en las zonas con mayores alinealidades, lo que ayuda a reducir el número de reglas necesarias. Por otra parte, se utiliza el modelo TaSe-NF [Herrera et al., 2011b], recientemente propuesto en la literatura, que ofrece un mecanismo de inferencia difusa modificado que ayuda a mejorar la interpretabilidad de las reglas. Esto es posible gracias a que se garantiza que cada consecuente representa exactamente la salida de la función inversa de la planta en el centro de la regla correspondiente. Por último, en OSENF-TaSe la evolución de la estructura del controlador se lleva a cabo por medio de la división sucesiva de aquellas reglas que presentan errores de control altos en sus zonas de influencia. Es importante resaltar que, mientras que en OSEFC cada cambio estructural implica la creación de varias reglas, en OSENF-TaSe sólo se añade una regla cada vez.

Finalmente, la tercera contribución de esta tesis se orienta hacia el estudio de nuevas estructuras para el diseño de controladores difusos auto-organizativos. En este sentido, se investigan los sistemas difusos de tipo 2[1], cuya popularidad ha aumentado notablemente en los últimos años al presentarse como métodos eficaces para el manejo de incertidumbre. Como paso previo a la adopción de este nuevo modelo difuso, se aborda el mayor criticismo dirigido a este tipo de sistemas, que establece que su mejor rendimiento (con respecto a los sistemas difusos tradicionales o de tipo 1) se debe únicamente al uso de un mayor número de parámetros [Cara et al., 2011b]. En concreto, se intenta determinar si existen diferencias significativas en el rendimiento de ambos tipos de sistemas cuando presentan el mismo número de parámetros y, en tal caso, cuándo es un tipo de sistema difuso preferible al otro.

Con el objetivo de proporcionar un marco común para la comparación de sistemas difusos de tipo 1 (con entradas numéricas y difusas) y de tipo 2, se propone un algoritmo evolutivo multi-objetivo para la optimización de la base de reglas y los parámetros de sistemas difusos de ambos tipos. Este método incluye un módulo para el cálculo óptimo de los consecuentes, tanto para sistemas de tipo 1 como para sistemas de tipo 2. El uso de este módulo ayuda a reducir el tamaño del espacio de búsqueda, a la vez que mejora la calidad de las soluciones proporcionadas por el algoritmo evolutivo. Por último, se presenta un estudio comparativo de los sistemas difusos de tipo 1 y 2 optimizados, para distintos niveles de incertidumbre en el entorno. En dicho estudio se han utilizado tests estadísticos no paramétricos para determinar si las diferencias de rendimiento entre los sistemas difusos de tipo 1 y los de tipo 2 son significativas o no.

[1]En concreto, se estudian los sistemas difusos de tipo 2 intervalo-valorados (*interval type-2 fuzzy systems*, en inglés). Por simplicidad, nos referimos a ellos simplemente como "sistemas difusos de tipo 2".

Estructura de la tesis

El trabajo presentado en esta tesis está organizado en capítulos, cuyo contenido se resume en la siguiente lista. Para mayor claridad, los títulos de los capítulos se presentan en español:

Capítulo 1. Introducción: Es la versión en inglés de esta introducción. En ella se presenta el concepto de control inteligente y se resumen las principales aportaciones de esta tesis doctoral.

Capítulo 2. Fundamentos: En este capítulo se presentan los fundamentos de lógica difusa de tipo 1 y 2 aplicada al campo de control. Estos conceptos representan la base teórica para la correcta comprensión del resto de esta memoria.

Capítulo 3. Control difuso auto-organizativo. Conceptos y estado del arte: Este capítulo describe el área del control difuso auto-organizativo. En primer lugar, se proporciona una breve descripción de la evolución histórica de los sistemas de control. Seguidamente, se define el concepto de sistema difuso auto-organizativo y se presentan sus principales propiedades. Por último, se realiza una revisión del estado del arte en el área del control difuso, prestando especial atención a los sistemas auto-organizativos.

Capítulo 4. OSEFC: Controlador difuso auto-organizativo en tiempo real: En este capítulo se presenta una nueva metodología para la auto-organización de un controlador difuso en tiempo real, empezando desde cero, cuando no hay conocimiento previo sobre la planta. Este método, que llamamos OSEFC (*Online Self-Evolving Fuzzy Controller*, en inglés) [Cara et al., 2011a], es capaz de determinar una topología adecuada para el controlador difuso, basándose en los datos obtenidos durante el funcionamiento normal del sistema. El aprendizaje se divide en dos etapas: la adaptación de los consecuentes de las reglas y la adición de nuevas funciones de pertenencia y reglas. El capítulo incluye resultados experimentales y simulaciones que ilustran las principales características de la metodología propuesta.

Capítulo 5. OSENF-TaSe: Controlador neuro-difuso auto-organizativo basado en el modelo TaSe-NF: Este capítulo presenta la segunda metodología propuesta para el aprendizaje automático de controladores difusos en tiempo real. Este método, que denominamos OSENF-TaSe (*Online Self-Evolving Neuro-Fuzzy controller based on the TaSe-NF model*, en inglés) [Cara et al., 2012], amplía las capacidades de OSEFC reduciendo la complejidad (número de reglas) del controlador a la vez que mejora la interpretabilidad de las reglas gracias al uso del modelo TaSe-NF. En este capítulo se presentan simulaciones y comparaciones de OSENF-TaSe con OSEFC y con otros métodos de control clásicos.

Capítulo 6. Optimización multi-objetivo de sistemas difusos de tipo 1 y 2 con propósito comparativo: En este capítulo se pretende abordar una de las críticas que más frecuentemente se dirigen a los sistemas difusos de tipo 2 y que establece que la mejor respuesta obtenida por estos sistemas (con respecto a los de tipo 1) se debe exclusivamente al uso de un mayor número de parámetros [Cara et al., 2011b]. Para ello, se presenta un algoritmo evolutivo multi-objetivo para la optimización de sistemas difusos de tipo 1 y 2. Su aplicación proporciona un marco común para la comparación, en el que los sistemas de tipo 1 (con entradas numéricas y con entradas difusas) y los de tipo 2 presentan configuraciones optimizadas con un número similar de parámetros, obtenidas en las mismas condiciones. En este contexto, se utilizan tests estadísticos para determinar si las diferencias de rendimiento son significativas o no.

Capítulo 7. Conclusiones: Este capítulo, que se presenta tanto en inglés como en español, resume las principales aportaciones y conclusiones extraídas del trabajo presentado en esta tesis doctoral. El capítulo incluye una lista de las publicaciones derivadas de la presente investigación.

Es importante resaltar que en esta tesis se consideran tanto sistemas difusos "tradicionales" (de tipo 1), como sistemas difusos de tipo 2. Los primeros son utilizados en los capítulos 4, 5 y 6, mientras que los segundos se aplican sólo en el capítulo 6. Para simplificar la notación, en los capítulos 4 y 5 simplemente usamos el término "lógica difusa". En cambio, en el capítulo 6 siempre se aclara a qué tipo de sistema difuso nos referimos en cada momento. Igualmente, hay que aclarar que en esta tesis no se estudian los sistemas difusos de tipo 2 en su versión general, sino que, como ya se ha mencionado, sólo se trata la versión intervalo-valorada.

Por último, para facilitar la lectura de esta tesis, se han adoptado las siguientes convenciones:

- Cada capítulo está dividido en secciones, subsecciones y sub-subsecciones. Cuando se proporcionan referencias a otras partes del texto, dichas referencias incluyen el número de capítulo seguido del número de sección, subsección, etc. Por ejemplo: *ver sección 2.4.4; ir al capítulo 4.*

- Las figuras y tablas se numeran por capítulos. Por ejemplo: *Fig. 2.1*.
- Las expresiones matemáticas se numeran también dentro de cada capítulo y se referencian mediante el número correspondiente entre paréntesis. Por ejemplo: *la planta dada por* (4.55).
- Las referencias bibliográficas se indican del siguiente modo: cuando el trabajo tiene uno o dos autores, se utiliza el apellido de los autores seguido del año de publicación, como por ejemplo en [Angelov, 2004; Wagner and Hagras, 2007]. Para los trabajos con más de tres autores, se utiliza el nombre del autor principal seguido de la abreviatura *et al.* y del año de publicación, como en el siguiente ejemplo: [Rojas et al., 2000].

Chapter 1

Introduction

1.1 Antecedents

The term "conventional (or traditional) control" is nowadays used to refer to the theories and methods that were developed in the last decades of the 20th century to control dynamical systems[1]. These techniques are generally based on the use of differential and difference equations [Antsaklis, 1994]. However, this mathematical framework may not be general enough in certain cases, as there are control problems that cannot be adequately described in terms of differential/difference equations [Gupta, 2000].

In 1993, the Task Force on Intelligent Control of the IEEE Control Systems Society defined the term "intelligent control" as the discipline that tackles the development of control methods that attempt to emulate important characteristics of human intelligence [Antsaklis, 1994; Antsaklis and Passino, 1993]. These characteristics include adaptation, learning and planning under uncertainties, among others. Hence, intelligent control attempts to build upon and enhance the conventional control methodologies to solve new challenging control problems.

In order to achieve its control goals for complex systems over a period of time, an intelligent controller frequently has to cope with significant uncertainty that fixed feedback robust controllers or adaptive controllers may not be able

[1]These methodologies constitute the traditionally known as "modern control theory", mainly developed between the 1960s and the 1980s.

to deal with [Chen and Narendra, 2003; Linkens and Nyongesa, 1996]. In this scenario, fault diagnosis and control reconfiguration, adaptation and learning are important considerations.

In some senses, the control problem tackled in intelligent control can be regarded as an enhanced, more ambitious and general version of the problem tackled in traditional control [Antsaklis, 1994; Gupta, 2000]. It presents increased control demands that cannot be met by the methods used in traditional control. As a consequence, the area of intelligent control is interdisciplinary and it attempts to combine and extend theories and methods from areas such as control, computer science and operations research.

It is worth noting that these theories and methodologies cannot, in general, be applied directly to solve control problems [Antsaklis, 1994]. The reason for this is not that these methods are not valid, but rather that they were conceived to address different needs. For instance, the problem of designing a controller can be cast as a problem of functional approximation of the plant's inverse function. However, techniques for function approximation cannot be applied directly, as the requirements and conditions in a control problem are different. Hence, they need to be enhanced and new methodologies need to be derived in combination with traditional control techniques. Additionally, traditional control concepts such as stability may have to be revisited and redefined [Antsaklis, 1994].

Sometimes, the term "autonomous (intelligent) control" is also used to refer to this discipline [Pachter and Chandler, 1998]. This designation emphasizes the fact that an intelligent controller usually aims to attain higher degrees of autonomy in its operation. Obviously, intelligence is required in order for the systems to provide the desired functioning under changing environments. Moreover, intelligence is also necessary to achieve a high degree of autonomous behavior. Thus, it is observed that the terms "autonomous control" and "intelligent control" are complementary.

In summary, an intelligent control system must be highly adaptable to unanticipated changes [Antsaklis, 1994]. Moreover, it must present a high degree of autonomy in dealing with these changes. To achieve these objectives, learning is fundamental. Learning control techniques were initially developed as a means of improving the performance of poorly modeled nonlinear systems by exploiting the experience gained through the online interaction with the actual plant [Gupta, 2000]. Nowadays, their use has been extended to a much wider use, e.g., accounting for uncertainties in the environment, overcoming problems due to wear or tear in the actuators, etc. [Angelov, 2004; Doctor et al., 2005a; Schaal and Atkeson, 2010].

Many paradigms and tools for intelligent control have appeared in the last two decades. Neural networks, genetic algorithms and fuzzy logic systems (FLSs) are some popular examples of these techniques [Gupta, 2000; Zilouchian and Jamshidi, 2000]. It is important to note that intelligent control does not restrict to the use of these methodologies. Moreover, according to the previous definitions, not all fuzzy or neural controllers would be considered intelligent [Antsaklis, 1994]. On the contrary, in order to provide them with the required autonomy and with the capability to cope with unexpected changes, they have to be extended with learning and self-organization methods.

In order to account for changing system dynamics, new operating conditions and environmental influences, in the last decade a new research topic has emerged, the so-called Evolving Intelligent Systems (EISs), of which Evolving Fuzzy Systems (EFSs) are a particular case [Angelov et al., 2010; Lughofer, 2011]. EISs combine the interpolation abilities of (neuro-) fuzzy systems and their flexibility with the adaptive feature of online learning techniques. In the case of evolving fuzzy systems, there is the additional advantage of their linguistic interpretability. Their main property is that they are automatically adapted, extended and evolved dynamically on-the-fly. Thus, an evolving system presents the ability to expand or shrink its structure, as well as to adapt its parameters, in an incremental way, online and, if necessary, in real time.

Evolving systems are able to support any modeling scenarios for data streams and dynamic data that change their nature and characteristics over time and space. Moreover, it is worth noting that they are permanently updating their knowledge by integrating new system behaviors and environmental influences in a life-long learning process that is always present. Therefore, they can be seen as a valuable contribution within the field of computational and artificial intelligence [Angelov and Kasabov, 2005; Lughofer, 2011].

From the previous discussion, it is obvious that all these properties make of an EIS a suited tool for the development of intelligent controllers. Moreover, the capability of evolving their own structure and parameters can be seen as the ability of designing a control law for a specific objective. This activity can in turn be regarded as the controller's attempt to organize its "knowledge" of its own dynamical behavior, which can be seen as an attribute of "intelligence" [Antsaklis, 1994]. Thus, it is possible to conclude that an evolving fuzzy controller capable of self-designing is an intelligent control system.

1.2 Contributions of the dissertation

The contributions presented in this dissertation aim to advance in the area of evolving fuzzy systems for intelligent control. The main objective is to investigate and develop new methodologies that are able to provide controllers using less prior knowledge about the systems to be controlled. Moreover, it is desired that these controllers present good capabilities for handling changing operating conditions, noise, and other uncertainties that may appear during their operation.

In the first part of this thesis, we propose two new methodologies for the online self-evolution of a fuzzy controller when there is no prior knowledge about the plant. Both methodologies are able to design fuzzy controllers from scratch, in an online manner, based on the analysis of the input/output data obtained from the normal operation of the plant. The underlying principles are the same in both cases: On the one hand, the property of universal approximation of fuzzy systems is exploited to incrementally build the controller by adding new membership functions (MFs) and rules. On the other hand, in order to provide higher robustness, the modification of the controller's structure is based on the analysis of the error surface in the entire operating region. Some of the features that are shared by these two methodologies are the following:

- *The methods proposed do not require a model of the plant or its differential equations*. Unlike most online methods proposed in the literature [Gao and Er, 2003; Park et al., 2005; Phan and Gale, 2008], these methods do not make assumptions about the differential equations (e.g., their bounds). The only requirement is that the monotonicity of the plant with respect to the control signal has a constant sign. Thus, only qualitative knowledge about the plant is used.

- *No previous knowledge about the control policy is required.* As mentioned before, the controller can start operating with only one rule initialized to zero and will self-develop while working online.
- *The controllers are robust*, in the sense that they can cope with unexpected isolated changes in the dynamics of the plant. Their adaptive nature provides them with the ability to learn the new behavior of the plant in order to continue delivering a satisfactory performance.
- *The evolving method is a life-long learning process.* As such, the controller continues learning and evolving during its entire operation. Two different types of learning are applied: On the one hand, the information about the error at the plant's output is used to adapt online the rule consequents. On the other hand, the error of the approximation of the plant's inverse function is analyzed to incrementally evolve the structure of the controller.

The first methodology presented is OSEFC (Online Self-Evolving Fuzzy Controller) [Cara et al., 2011a]. In this methodology, a complete set of rules is used for the fuzzy controller. When the controller's topology has to be modified, the error surface in the entire operating regions is analyzed and one membership function is added to the input variable that is mostly responsible for the error. The method is also able to select the input variables that are relevant for the control, from a set of candidate inputs. This last feature also helps to reduce the initial knowledge required. OSEFC provides controllers with good performance despite the fact of not using any prior knowledge about the plant. However, due to its grid-like structure, the number of rules grows exponentially (i.e., it suffers the well-known curse of dimensionality [Bellman, 1961]). This situation has a negative effect for both the complexity and the interpretability of the controller.

In order to tackle these problems, the second methodology (which we call OSENF-TaSe[1]) is proposed [Cara et al., 2012]. OSENF-TaSe uses a scatter parti-

[1]OSENF-TaSe stands for Online Self-Evolving Neuro-Fuzzy controller based on the TaSe-NF model

tioning of the input space. This distribution reduces the number of rules needed by directly locating them in the regions with higher nonlinearities. In addition, the use of the modified inference process offered by the recently proposed TaSe-NF model [Herrera et al., 2011b] enhances the interpretability of the rules. This is achieved by guaranteeing that each rule consequent represents the output of the plant's inverse function at the center of the corresponding rule. In OSENF-TaSe, the controller's structure evolves through the successive splitting of the rules that present high control errors in their areas of influence. It is important to note that while in OSEFC the addition of one new membership function implies the generation of several rules, in OSENF-TaSe only one rule is added at each step.

Finally, the third contribution of this dissertation is oriented towards the study of new structures for the design of evolving fuzzy controllers. In this sense, we investigate the increasingly popular interval type-2 fuzzy systems, which have emerged in the last years as an effective mechanism for uncertainty handling. As a previous step for the adoption if this new model, we address one of the main criticisms directed to type-2 fuzzy systems, which states that the only reason why they outperform their type-1 counterparts is their use of extra parameters [Cara et al., 2011b]. Specifically, we aim to determine if there are significant differences in the performance of type-1 and interval type-2 FLSs with the same number of parameters and, in that case, when is a method preferred to the other.

With the goal of providing a common framework for the comparison of these systems, we propose a multi-objective evolutionary algorithm for the optimization of the rule base and parameters of type-1 (both singleton and non-singleton) and singleton interval type-2 fuzzy systems. This methodology includes a module for computing the optimal consequents for a given rule base, for both type-1 and singleton interval type-2 FLSs. This module helps to reduce the size of the search space and to improve the accuracy of the solutions found by the multi-objective evolutionary algorithm. Finally, a comparative study, based on the optimized fuzzy systems under different levels of noise, is carried out. Non-parametric statistical tests are used to determine whether the differences between the performance of type-1 and interval type-2 FLSs are significant or not.

1.3 Structure of the dissertation

The work presented in this dissertation is organized in chapters. The following list summarizes the contents of each chapter:

Chapter 1. Introduction: It presents the field of intelligent control and summarizes the contributions and structure of this dissertation. This chapter is also presented in Spanish.

Chapter 2. Preliminaries: In this chapter, we provide the main concepts and definitions of type-1 and type-2 fuzzy logic applied to the field of control. These concepts represent the theoretical basis for the proper understanding of the remaining chapters.

Chapter 3. Evolving fuzzy control. Concepts and state of the art: This chapter introduces the field of evolving fuzzy control. First, a brief historical description of the evolution of control systems is presented. Next, the definition and main properties of evolving fuzzy systems are provided. The chapter ends with a review of the state of the art in the area of fuzzy control.

Chapter 4. OSEFC: Online Self-Evolving Fuzzy Controller: This chapter presents the first methodology proposed for the online self-evolution of a fuzzy controller, starting from scratch, when there is no prior information about the plant. This method, which we call OSEFC [Cara et al., 2011a], is able to determine an adequate topology for the fuzzy controller based on the data obtained during the system's normal operation. It comprises two phases: Online adaptation of the rule consequents, and online addition of new membership functions and rules. The chapter also includes simulation and experimental results to illustrate the main properties of the proposed methodology.

Chapter 5. OSENF-TaSe: Online Self-Evolving Neuro-Fuzzy Controller based on the TaSe-NF model: In this chapter the second methodology for the online self-design of a fuzzy controller (called OSENF-TaSe) is proposed [Cara et al., 2012]. OSENF-TaSe extends OSEFC's capabilities by reducing the controller's complexity (i.e., the number of rules) while increasing the interpretability of the rules thanks to the use of the TaSe-NF model. The chapter includes simulation results and comparisons with OSEFC and with classic controllers.

Chapter 6. Multi-objective optimization of type-1 and type-2 fuzzy systems with comparative purposes: This chapter aims to address one of the most common criticisms of type-2 fuzzy systems, which states that they outperform their type-1 counterparts based solely on the use of extra parameters [Cara et al., 2011b]. The chapter presents a multi-objective evolutionary algorithm for the optimization of type-1 and interval type-2 fuzzy systems. This optimization procedure provides a common framework for the comparison, where (singleton and non-singleton) type-1 and interval type-2 FLSs present optimized configurations with a similar number of parameters, obtained under the same conditions. In this context, statistical tests are used to determine if the differences in their performance are significant.

Chapter 7. Conclusions of the dissertation and list of publications: This chapter summarizes the main conclusions extracted from the work presented in this dissertation. The publications derived from this research are also enumerated. This chapter is also presented in Spanish.

It has to be noted that two types of fuzzy logic systems are considered in this dissertation, namely type-1 and interval type-2 FLSs. The former are used in chapters 4, 5 and 6, whilst the latter are only considered in chapter 6. For simplicity, in the chapters that only apply type-1 fuzzy logic, we omit the adjective "type-1" and simply use terms as "fuzzy controller", "fuzzy logic", and so on. In chapter 6, however, the particles "type-1" and "type-2" are always used to differentiate them. Additionally, it has to be clarified that this dissertation considers singleton interval type-2 FLSs only. For brevity, we sometimes refer to them simply as "type-2 FLSs"; however, the reader has to keep in mind that general type-2 FLSs are not treated in this work.

Finally, in order to make easier the reading of this thesis, the following conventions are applied:

- Each chapter is divided into sections that can include subsections and subsubsections. When a reference to any part of the text is given, this reference includes the number of chapter followed by the number of section, subsection and so on. For instance, *see section 2.4.4; go to chapter 4*.

- Figures and tables are numbered by chapter. For example, *Fig. 2.1*.

- Mathematical expressions are also numbered by chapter and referenced with their number enclosed in parentheses, e.g., *the plant given by* (4.55).

- Bibliographic references are indicated as follows: When the work has one or two authors, the last name of the authors followed by the year of publication is used; for instance, [Angelov, 2004; Wagner and Hagras, 2007]. When there are three or more authors, we use the last name of the main author followed by the abbreviation *et al.* and the year of publication, e.g., [Rojas et al., 2000].

Chapter 2

Preliminaries

In this chapter, we review the main concepts of the theory of fuzzy logic required for the proper comprehension of the work presented in this dissertation. Both type-1 fuzzy logic (or simply "fuzzy logic") and type-2 fuzzy logic are considered. Section 2.1 presents the concepts of (type-1) fuzzy sets, operations on fuzzy sets, fuzzy inference and universal approximation property of fuzzy logic. In section 2.2, the structure of a fuzzy controller and its components are described. The fundamentals of type-2 fuzzy logic and the structure of a type-2 fuzzy logic system are described in sections 2.3 and 2.4, respectively.

2.1 Fuzzy logic

The complexity of most real-world processes (e.g., physical, chemical or other processes from nature) makes it almost impossible to obtain a precise description of their dynamics in terms of a set of differential equations. Moreover, such precise representation may be even unnecessary in many cases to achieve an effective control of the process. On the other hand, human beings can control such types of processes satisfactorily without using complex mathematical models or without requiring a deep comprehension of their internal dynamics. Driving vehicles, controlling the level of liquid in reservoirs or the concentration of chemical compounds in tanks are examples of such processes that are successfully controlled by humans.

To overcome the need of a precise representation of knowledge, in 1965 Zadeh introduced fuzzy sets [Zadeh, 1965] that allowed for the use of approximate forms of reasoning in which the notion of "true" or "false" is a matter of degree. Ten years later, Zadeh extended these fuzzy sets with the so-called type-2 fuzzy sets [Zadeh, 1975], with the purpose of modeling the uncertainties that can appear in the design of ordinary (type-1) fuzzy sets [Lughofer, 2011; Mendel, 2001].

Over the last decades, fuzzy logic has been successfully applied to different areas, such as regression, system modeling, control, pattern classification, etc. Among them, the field of control can be regarded as one of the most relevant [Hagras, 2007; Herrera et al., 2011b; Mendel, 2001]. The main reasons for the wide acceptance of fuzzy systems are the following:

- They avoid the need of a precise mathematical model of the process.
- They present a high generalization capability.
- They avoid the use of black-box systems, that are harder to understand, providing instead a set of comprehensible rules that are easier to be interpreted by humans. In the case of control problems, this means that the linguistic control strategy used by a human operator is translated into an automatic control strategy [Lee, 1990; Lughofer, 2011].

This section focuses on the concepts and fundamentals of type-1 fuzzy theory (which for simplicity we call just "fuzzy logic") that are of interest for the reader of this dissertation. Most of the contents presented are taken from [Pomares, 2000]. The structure and operation of a fuzzy controller is presented later in section 2.2. The concepts and operations for type-2 fuzzy logic, as well as the structure of type-2 fuzzy logic systems, can be found in sections 2.3 and 2.4, respectively.

2.1.1 Fuzzy sets. Concepts and definitions

Classic sets, as they are named to differentiate them from fuzzy sets, are characterized by a clearly defined boundary or frontier. Every element in the universe of discourse whether completely belongs or not to the set. Thus, the membership function (MF) for the element x to the set A is given by

$$\mu_A(x) = \begin{cases} 1 & if x \in A \\ 0 & \text{if } x \notin A \end{cases} \tag{2.1}$$

Mathematically speaking, the set A and its membership function $\mu_A(x)$ are equivalent, in the sense that knowing the former is the same as knowing the latter, and vice versa. However, human knowledge cannot be always so easily categorized. For instance, if Z is defined as the set of all the cars whose *maximum speed is high*, the limit of the set is not clear. One of the main contributions of Zadeh was the definition of the concept of *membership degree* for fuzzy sets, which allows for the use of different levels of belonging of a given element to a set.

Therefore, a fuzzy set Z is defined by the pair $(x, \mu_z(x))$, where x is an element and $\mu_z(x)$ is the membership degree of x to the fuzzy set Z, given by a real value in the interval $[0, 1]$.

For a discrete set, the definition of the elements belonging to the set is

$$Z = \sum_{i=1}^{n} x_i/\mu_z(x_i) = x_1/\mu_z(x_1) + ... + x_n/\mu_z(x_n) \tag{2.2}$$

where the symbol "+" has been used to represent the union of all the elements in the set. These elements have been defined as $x/\mu_z(x)$, where the symbol "/" represents an ordered pair.

In the continuous case, the set is defined as

$$Z = \int_X x/\mu_z(x) \tag{2.3}$$

Fuzzy logic is characterized by the use of linguistic terms instead of numeric values to represent the state of the variables, e.g., *high temperature* or *low speed*. Thus, a linguistic variable is a variable whose values are words instead of numbers. The use of words instead of precise numeric values is motivated by the fact that linguistic descriptions contain more information and are more comprehensible for humans, even if they are more imprecise [Mendel et al., 2010; Zadeh, 1996]. In this section, we define the main concepts used in the field of fuzzy logic, some of which will be used in the rest of this dissertation.

Definition 1. Fuzzy set and membership function: Let X be a collection of objects x. Then the fuzzy set A in the universe of discourse X is defined as a set of ordered pairs of the form

$$A = \{(x, \mu_A(x)) \quad | \quad x \in X\} \tag{2.4}$$

where $\mu_A(x)$ is the membership function of the element x to the fuzzy set A, which takes real values in the interval $[0, 1]$. If the membership function is restricted to take its values in $\{0, 1\}$, then the fuzzy set is equivalent to a classic set.

Definition 2. Inclusion: Given two fuzzy sets A and B in the universe of discourse X, A is said to be included in B (equivalently, A is a subset of B) if and only if for every element $x \in X$ it is satisfied that $\mu_A(x) \leq \mu_B(x)$. Formally,

$$A \subseteq B \Leftrightarrow \mu_A(x) \leq \mu_B(x); \;\; x \in X \tag{2.5}$$

Additionally, A and B are equal if and only if $\mu_A(x) = \mu_B(x)$, $\forall x \in X$.

Definition 3. Convexity: A fuzzy set A is said to be convex if and only if the following property is satisfied:

$$x, y \in X, \; \lambda \in [0, 1] : \; \mu_A(\lambda x + (1 - \lambda)y) \geq \min(\mu_A(x), \mu_A(y)) \tag{2.6}$$

Definition 4. Support: The support of a fuzzy set A is defined as the set of all the elements whose membership degree to A is non-zero. Formally,

$$S(A) = \{x \in X \quad | \quad \mu_A(x) > 0\} \tag{2.7}$$

Definition 5. Singleton: A singleton is a fuzzy set whose support is a single point.

Definition 6. Amplitude: The amplitude of a convex fuzzy set A with support $S(A)$ is defined as

$$Amp(A) = \sup(S(A)) - \inf(S(A)) \tag{2.8}$$

In control applications it is common to work with convex fuzzy sets. In this case, the notions of support and amplitude are generally used indistinctly. It is important to note that the support is an interval; if it is closed, the supremum and infimum operations can be replaced by the maximum and minimum, respectively.

Definition 7. Core: The core of a fuzzy set A is defined as

$$Core(A) = \{x \in X \quad | \quad \mu_A(x) = 1\} \tag{2.9}$$

If the core is formed by a single element x_p (e.g., as in fuzzy sets defined by triangular MFs), then x_p is called the apex of the fuzzy set.

Definition 8. Height: The height of a fuzzy set A is the maximum membership degree of the elements of its support. Formally,

$$Height(A) = \sup\{\mu_A(x) \quad | \quad x \in X\} \tag{2.10}$$

Definition 9. Normal fuzzy set: A fuzzy set is said to be normal if its height is equal to 1 or, equivalently, if its core is not the empty set.

Definition 10. α-Cut: Given a fuzzy set A, its α-cut is the subset of A that contains all the elements $x \in X$ such that their membership degree is greater than or equal to α. Formally,

$$A_\alpha = \{x \in X \quad | \quad \mu_A(x) \geq \alpha\} \tag{2.11}$$

2.1.2 Operations on fuzzy sets

Intersection, union and complement are the basic operations defined in classical set theory. They can be extended to fuzzy sets, although there is not a unique definition for these operations. The definitions given by Zadeh [Zadeh, 1973] are the following:

$$\begin{aligned}
&\forall x \in X : \mu_{A \wedge B}(x) = \min(\mu_A(x), \mu_B(x)) \\
&\forall x \in X : \mu_{A \vee B}(x) = \max(\mu_A(x), \mu_B(x)) \\
&\forall x \in X : \mu_{A'}(x) = 1 - \mu_A(x)
\end{aligned} \tag{2.12}$$

These definitions have the property that, when the membership values are restricted to $\{0, 1\}$, the operations with fuzzy sets are equivalent to the corresponding operations with classic sets. The Minimum and Maximum operators were widely used in the design of the first fuzzy systems, that followed the Mamdani design. Nonetheless, afterwards many efforts were directed to study the properties of such operators on the fuzzy inference process [Cárdenas et al., 1993; Kovalerchuk and Taliansky, 1992] and to prove if some options can be better than others in certain applications [Gupta and Qi, 1991; Kruse et al., 1994]. In the following subsections we define the properties that have to be satisfied by the operators used to implement the operations on fuzzy sets.

2.1.2.1 Intersection. T-norms

Definition 11. T-norm: A continuous function $t : [0,1] \times [0,1] \to [0,1]$ is a t-norm if it satisfies the following properties:

$$\begin{array}{ll} \text{Commutativity:} & t(a,b) = t(b,a) \\ \text{Associativity:} & t(t(a,b),c) = t(a,t(b,c)) \\ \text{Monotonicity:} & \text{if } a \leq c \text{ and } b \leq d \Rightarrow t(a,b) \leq t(c,d) \\ \text{Identity element:} & t(a,1) = a \end{array}$$

where a, b, c, d are real values in the interval $[0,1]$.

From the monotonicity property and the identity element it also follows that

$$a \in [0,1] : \quad t(a,0) = 0$$

Some examples of t-norms are the following [Michels et al., 2006]:

$$\text{Minimum:} \quad t(a,b) = \min(a,b) \tag{2.13a}$$

$$\text{Algebraic product:} \quad t(a,b) = a \cdot b \tag{2.13b}$$

$$\text{Lukasiewicz t-norm:} \quad t(a,b) = \max(0, a+b-1) \tag{2.13c}$$

$$\text{Drastic product:} \quad t(a,b) = \begin{cases} a & \text{if } b = 1 \\ b & \text{if } a = 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.13d}$$

In spite of the wide variety of existing t-norms, in most practical applications, the minimum, the algebraic product and the Lukasiewicz t-norm are chosen [Michels et al., 2006]. Additionally, it can be proven that the largest t-norm is the

Minimum operator, i.e., for any set of inputs any other t-norm produces a value that is smaller than or equal to the minimum of the inputs. More information on t-norms can be found in [Klement et al., 2000].

2.1.2.2 Union. T-conorms

Another relevant operation on fuzzy sets is the union. The properties that must be satisfied by the union operator can be gathered under the following definition.

Definition 12. T-conorm: A continuous function $t^* : [0,1] \times [0,1] \rightarrow [0,1]$ is a t-conorm if it satisfies the following properties:

Commutativity:	$t^*(a,b) = t^*(b,a)$
Associativity:	$t^*(t^*(a,b),c) = t^*(a,t^*(b,c))$
Monotonicity:	$\text{if } a \le c \text{ and } b \le d \Rightarrow t^*(a,b) \le t^*(c,d)$
Identity element:	$t^*(a,0) = a$

where a, b, c, d are real values in the interval $[0,1]$.

Again, from the monotonicity property and the identity element it also follows that

$$a \in [0,1] : \quad t^*(a,1) = 1$$

T-norms and t-conorms are dual concepts [Alsina et al., 1983; Michels et al., 2006] in the sense that each t-norm t induces a t-conorm t^* by

$$t^*(a,b) = 1 - t(1-a, 1-b) \tag{2.14}$$

and, vice versa, from a t-conorm t^* it is possible to obtain the corresponding t-norm t by applying

$$t(a,b) = 1 - t^*(1-a, 1-b) \tag{2.15}$$

Some of the most widely used t-conorms in the literature are:

Maximum: $$t^*(a,b) = \max(a,b) \tag{2.16a}$$

Algebraic sum: $$t^*(a,b) = a + b - a \cdot b \tag{2.16b}$$

Lukasiewicz t-conorm: $$t^*(a,b) = \min(1, a+b) \tag{2.16c}$$

Drastic sum: $$t^*(a,b) = \begin{cases} a & \text{if } b = 0 \\ b & \text{if } a = 0 \\ 1 & \text{otherwise} \end{cases} \tag{2.16d}$$

It is easy to observe that these operators are the dual to the t-norms defined in the previous subsection, as they are obtained from (2.13) by applying (2.14). Thus, it can be proven that the Maximum operator is the smallest t-conorm.

2.1.2.3 Complement. Negation operators

The complement of a fuzzy set can be defined similarly to the definition used in classical set theory, although it has to be noted that the operation is now applied to the values of the corresponding membership function. Thus, the intuitive interpretation of the fuzzy complement is that it measures the degree to which an element $x \in X$ does not belong to a given fuzzy set.

Definition 13. Fuzzy complement: A continuous function $c : [0,1] \to [0,1]$ is a fuzzy complement operator if it satisfies the following properties:

Boundary condition: $c(0) = 1$ and $c(1) = 0$

Negative monotonicity: if $a < b \Rightarrow c(a) > c(b)$

Involution: $c(c(a)) = a$

where a, b are real values in the interval $[0,1]$.

Some examples of complement functions are the following:

Ordinary complement: $$c(a) = 1 - a \tag{2.17a}$$

Yager complement: $$c_w(a) = (1 - a^w)^{1/w}, \text{ with } w \in (0, \infty) \tag{2.17b}$$

Sugeno complement: $$c_\lambda(a) = \frac{1-a}{1+\lambda a}, \text{ with } \lambda \in (-1, \infty) \tag{2.17c}$$

In classical set theory the intersection of a set with its complement is equal to the empty set, whilst the union of a set and its complement yields to the whole universe of discourse. In the context of fuzzy sets, these laws are weakened as $t(a, c(a)) \leq 0.5$ and $t^*(a, c(a)) \geq 0.5$ [Michels et al., 2006]. Another important property of the complement operation is that it has at least one point of equilibrium, i.e., a point a for which $c(a) = a$. This means that the point of equilibrium of a fuzzy set is a value such that the membership degree to the set and to its complement is the same.

2.1.3 Approximate reasoning

Fuzzy propositions are the primitive elements of human knowledge representation in fuzzy logic. A fuzzy proposition is a statement of the form *the temperature is high*, where *temperature* is a linguistic variable and *high* is a linguistic label defined for that variable. Fuzzy propositions can be combined by means of linguistic connectives in many different ways, derived from the following fundamental operations [Lee, 1990]:

- *Conjunction* (denoted $p \wedge q$): It forms a new proposition based on the truthfulness of both propositions.
- *Disjunction* (denoted $p \vee q$): It forms a new proposition based on the veracity of one of the propositions.
- *Implication* (denoted $p \rightarrow q$): Used to form IF-THEN fuzzy rules.
- *Negation* (denoted $\neg p$): It forms a new proposition of the form "it is false that...".

Thus, a fuzzy rule is a conditional statement of the form

$$\text{IF } proposition_A \text{ THEN } proposition_B$$

where $proposition_A$ and $proposition_B$ can be simple or complex fuzzy propositions. The IF-part is the rule antecedent, whilst the THEN-part is the consequent.

For instance, in the rule

IF *temperature* is *high* THEN *pressure* is *very high*

high and *very high* are linguistic labels for the linguistic variables *temperature* and *pressure*, respectively. The proposition "*temperature* is *high*" is the antecedent and "*pressure* is *very high*" is the consequent.

A fuzzy rule of the form "IF X is P THEN Y is B" can be abbreviated as $P \rightarrow B$. This notation essentially describes a relation between the two variables X and Y. Thus, an IF-THEN rule can be also seen as a fuzzy binary relation R on $X \times Y$, where each element $(x, y) \in X \times Y$ is associated to a membership degree $\mu_R(x, y)$. Formally,

$$R = P \rightarrow B = \int_{X\times Y} \mu_P(x) \rightarrow \mu_P(y)/(x, y) \tag{2.18}$$

where $\rightarrow$ represents the implication function chosen, e.g., Kleene-Dienes, Lukasiewicz, Zadeh, Gödel, Mamdani, etc. Therefore, a fuzzy relation represents a degree of presence or absence of association, interaction, or interconnectedness between the elements of two or more fuzzy sets [Mendel, 1995].

To obtain a fuzzy consequence from a set of antecedents, a fuzzy inference process is performed [Cao and Kandel, 1992]. This process operates over the fuzzy sets associated to the propositions and produces a fuzzy or numeric consequence (the latter is more common in control and function approximation problems). In engineering applications, the Generalized Modus Ponens (GMP) is usually the inference mechanism selected. Its operation can be represented as:

premise 1:	X is P'
premise 2:	IF X is P THEN Y is B
consequence:	Y is B'

where P, P', B and B' are linguistic labels. Opposed to traditional crisp logic, it is not required that the premise set is exactly equal to the antecedent set, i.e., a rule can be fired to different levels, depending on how similar the antecedent and the premise sets are. To obtain the fuzzy set B', the operation $B' = P' \circ R$ is performed, where $\circ$ represents the fuzzy composition and R is the fuzzy relation generated by the rule in the second premise. Two basic operations are used to define the fuzzy composition, namely projection and cylindrical extension [Dubois and Prade, 1980].

Definition 14. Projection: Let R be a fuzzy relation on $U = \prod_{i=1}^{n} U_i = U_1 \times U_2 \times \cdots \times U_n$, where $(i_1, \ldots, i_k)$ is a subsequence of $(1, \ldots, n)$ with $k < n$, and $(j_1, \ldots, j_l)$ is its complementary subsequence. Consider $V = \prod_{m=1}^{k} U_{i_m}$. The projection of R on V is defined as

$$proy(R) \;\; on \; V = \int_V \sup_{x_{j_1} \ldots x_{j_l}} \mu_R(x_1, \ldots, x_n)/(x_{i_1}, \ldots, x_{i_k}) \tag{2.19}$$

In the binary case, if we define R on $X \times Y$, we have:

$$proy(R) \; on \; Y = \int_Y \sup_x \mu_R(x, y)/y \tag{2.20}$$

Definition 15. Cylindrical extension: Let S be a fuzzy relation on $V = \prod_{m=1}^{k} U_{i_m}$ (see previous definition). The cylindrical extension of S in U is defined by

$$ce(S) = \int_U \mu_S(x_{i_1}, \ldots, x_{i_k})/(x_1, \ldots, x_n) \tag{2.21}$$

In the binary case, if we define F as a fuzzy set over Y, we have:

$$ce(F) = \int_{X \times Y} \mu_F(y)/(x, y) \tag{2.22}$$

Definition 16. Fuzzy composition: Let A be a fuzzy set defined on X and let R be a fuzzy relation on $X \times Y$. The composition of A and R (denoted $A \circ R$) is a fuzzy set B on Y given by

$$B = A \circ R = proy(ce(A) \cap R) \; on \; Y \tag{2.23}$$

If the minimum is used for the intersection and the maximum for the projection, the above expression is equivalent to the inference rule proposed by Zadeh, i.e.,

$$\mu_b(y) = \max_x \min(\mu_A(x), \mu_R(x, y)) \tag{2.24}$$

Equivalently, if the product is used for the intersection,

$$\mu_b(y) = \max_x(\mu_A(x) \cdot \mu_R(x, y)) \tag{2.25}$$

If conjunctive premises are used in the antecedent part of the rule, then the GMP inference process is as follows:

premise 1:	X is P' AND Y is Q'
premise 2:	IF X is P AND Y is Q THEN Z is B
consequence:	Z is B'

In this case, the second premise is a rule of the form $P \times Q \rightarrow B$, which can be expressed as a ternary fuzzy relation R. The membership level of the elements in the Cartesian product $X \times Y \times Z$ is given by $\mu_R(x, y, z) = \mu_{(P \times Q) \rightarrow B}(x, y, z)$. Finally, the consequence set B' can be computed using the product $\bullet$ (for the implication, intersection and conjunction) and the maximum $\vee$ (for the projection) as follows:

$$\mu_{B'}(z) = \underbrace{\overbrace{\bigvee_{x,y}}^{\text{projection}} \left\{ \left[\mu_{P'}(x) \overbrace{\bullet}^{\text{conjunction}} \mu_{Q'}(y) \right] \overbrace{\bullet}^{\text{intersection}} \left[\underbrace{\left(\mu_P(x) \overbrace{\bullet}^{\text{conjunction}} \mu_Q(y) \right) \overbrace{\bullet}^{\text{implication}} \mu_B(z)}_{\mu_R(x,y,z)} \right] \right\}}_{\text{composition}} \tag{2.26}$$

which leads to

$$\begin{aligned}
\mu_{B'}(z) &= \bigvee_{x,y} \{ [\mu_{P'}(x) \bullet \mu_{Q'}(y)] \bullet [(\mu_P(x) \bullet \mu_Q(y)) \bullet \mu_B(z)] \} \\
&= \bigvee_{x,y} \{\mu_{P'}(x) \bullet \mu_{Q'}(y) \bullet \mu_P(x) \bullet \mu_Q(y)\} \bullet \mu_B(z) \\
&= \left(\bigvee_x \{\mu_{P'}(x) \bullet \mu_P(x)\} \right) \bullet \left(\bigvee_y \{\mu_{Q'}(y) \bullet \mu_Q(y)\} \right) \bullet \mu_B(z) \\
&= (\alpha_1 \bullet \alpha_2) \bullet \mu_B(z) = \alpha^{rule} \bullet \mu_B(z)
\end{aligned} \tag{2.27}$$

where α_1 is the similarity level between the value P in the antecedent and the measured value P' in the premise, α_2 is the similarity level between the value Q in the antecedent and the measured value Q' in the premise, and α^{rule} is the total firing of the rule.

Finally, an aggregation operator is required to combine the output of multiple rules into a final output value. Generally, the maximum or the sum-of-sets is used for Mamdani-type fuzzy systems, whilst the weighted sum or the weighted average are preferred for Takagi-Sugeno-Kang (TSK) systems (see section 2.2). This aggregation produces an output fuzzy set, which is inferred from a given fuzzy input by a set of fuzzy rules. It is important to note that there are other ways to develop the inference process. A detailed analysis can be found in [Driankov et al., 1993; Jager, 1995].

2.1.4 Fuzzy logic systems as universal approximators

One of the most important issues that arises in the design of fuzzy systems is the precision that can be achieved when approximating an arbitrary function by a set of fuzzy rules and membership functions. For neural networks, it was proven that a perceptron with three layers can approximate any real continuous function in a compact domain to any desired accuracy, if an adequate number of neurons is used [Hornick et al., 1989]. The same result was proven in [Wang and Mendel, 1992] for Mamdani fuzzy systems with numeric consequents. Using the Stone-Weierstrass theorem, Buckley [1992] proved that a set of fuzzy controllers with Gaussian membership functions, product t-norm and center-of-maximum defuzzification, can approximate any real continuous function in a compact domain.

Kosko [Kosko, 1992, 1994] developed a similar analysis, although based in the concept of fuzzy patches. In [Buckley, 1993], it was proven that Sugeno fuzzy systems with some specific properties were also able to approximate any type of continuous function. Nonetheless, in [Castro, 1995; Castro and Delgado, 1996] it was shown that there exist other types of fuzzy systems that do not use the structures and primitives defined in the previous works but are also universal approximators. Thus, in [Castro, 1995], it was proven that fuzzy systems that use

- triangular, Gaussian or trapezoidal membership functions,
- t-norm for the conjunction,
- implication functions that satisfy only some fundamental properties, and
- the usual defuzzification methods in the literature (mean of maximum, center of areas, etc.), with the only condition that the defuzzification point belongs to the support of the output set,

are all universal approximators. Additionally, the hybridization of neural paradigms and fuzzy logic also leads to universal approximators [Buckley and Hayashi, 1994]. However, despite of these results, there is not a systematic methodology for the design of fuzzy systems (selection of structure, operators and parameters) that guarantees that the desired accuracy level will be met.

2.2 Structure of a fuzzy controller

The main components of a fuzzy controller (or fuzzy system, in general) are depicted in Fig. 2.1. They can be summarized as:

- *Fuzzifier*: This block transforms the crisp inputs into fuzzy sets, so that they can be used in the fuzzy inference process. This process is known as fuzzification.
- *Knowledge base (KB)*: It contains the definition of the linguistic values for all the input and output linguistic variables and the rules that form the system.
- *Fuzzy inference engine*: This block performs the inference process according to the activation of the rules. It can be considered that this block has the capability of simulating the human process of decision making.
- *Defuzzifier*: In this block, the output of the rules are combined into a single output fuzzy set which is then transformed into a final numeric output value. These processes are called aggregation and defuzzification, respectively.

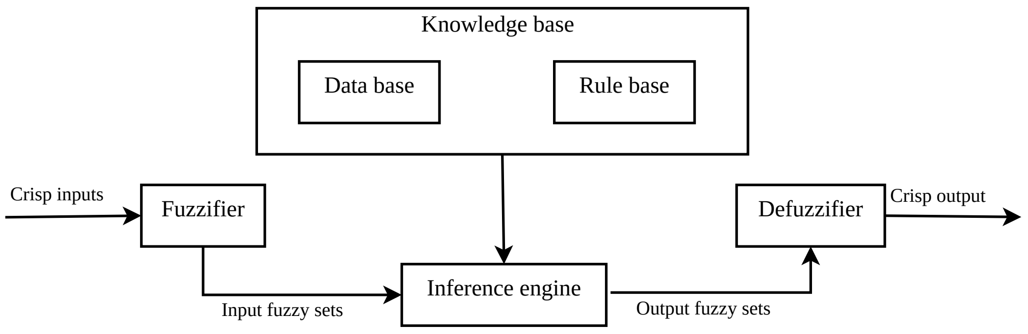


FIGURE 2.1: Structure of a fuzzy controller

In the following subsections we describe in some detail the operation of each block.

2.2.1 Fuzzification

The first step that has to be performed in a fuzzy system is the conversion of the numeric input values into fuzzy sets that can be used as the premises of the rule antecedents in the fuzzy inference process. Formally, the fuzzifier maps a crisp input vector with n inputs $\vec{x} = [x_1, ..., x_n]^T \in X_1 \times ... \times X_n$ into n fuzzy sets A_x. There are two types of fuzzification, namely singleton and non-singleton [Mendel, 1995].

When singleton fuzzification is used, the numeric input value is considered to be precisely the desired input value. Thus, for an input x', a singleton fuzzy set A_x with support x' is defined as (see Fig. 2.2(a))

$$A_x = \begin{cases} 1 & \text{if } x = x' \\ 0 & \text{if } x \neq x' \end{cases} \tag{2.28}$$

On the contrary, a non-singleton fuzzifier produces a fuzzy set A_x for which $\mu_{A_x}(x') = 1$ and $\mu_{A_x}(x)$ decreases from unity as x moves away from x'. Thus, in non-singleton fuzzification, the input value x' is mapped into a fuzzy number, i.e., a fuzzy membership function (e.g., Gaussian or triangular) is associated with it. Fig. 2.2(b) depicts an example of non-singleton fuzzification with a triangular membership function.

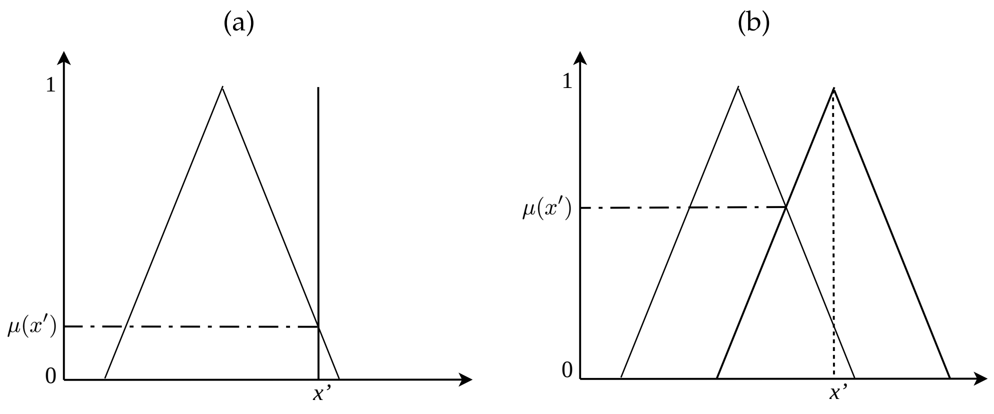


FIGURE 2.2: Types of fuzzification and antecedent matching. (a) Singleton fuzzification. (b) Non-singleton fuzzification

Singleton fuzzification is usually preferred in control applications, because it is simpler and faster to compute. However, it may not always be adequate, especially when the data are corrupted by noise. In this case, non-singleton fuzzification is, at least in theory, a better alternative.

It is a general convention to name a fuzzy system after the type of fuzzification used. Hence, we can define the following types of fuzzy systems:

Definition 17. Singleton (type-1) fuzzy system: A fuzzy system that uses type-1 fuzzy sets (and membership functions) and singleton fuzzification is called a singleton (type-1) fuzzy system. Since this is the most widely used, when there is no possible ambiguity, it is usually referred to simply as "fuzzy system".

Definition 18. Non-singleton (type-1) fuzzy system: A fuzzy system that uses type-1 fuzzy sets (and membership functions) and non-singleton fuzzification is called a non-singleton (type-1) fuzzy system.

2.2.2 Knowledge base

The knowledge base of a fuzzy system stores all the information needed to define the rules and membership functions in the system. It is formed by two components, namely the data base (DB) and the rule base (RB).

The data base stores the definition of the linguistic variables and their linguistic labels. It defines all the membership functions, for both input and output variables. The selection of the granularity or density of membership functions is very important in the modeling process.

The rule base comprises the fuzzy rules defined in the system. These rules define a partition of the input space that can have a grid or scatter structure, depending on whether a complete set of rules (with a rule for each possible combination of the linguistic labels) is used or not.

For a multiple-input single-output (MISO) fuzzy system with n inputs $x_1 \in X_1, ..., x_n \in X_n$ and one output $y \in Y$, the i-th rule is given by

$$\Re^i : \text{IF } x_1 \text{ is } F_1^i \text{ AND } x_2 \text{ is } F_2^i \text{ AND... } x_n \text{ is } F_n^i \text{ THEN } y \text{ is } G^i \tag{2.29}$$

where $F_1^i, ..., F_n^i$ are the input fuzzy sets and G^i is the output fuzzy set or numeric value used to define the consequent of the rule.

2.2.3 Fuzzy inference engine

The fuzzy engine is the real core of the fuzzy controller. It combines the rules in the rule base to provide a mapping from the input fuzzy sets to the output fuzzy sets that will be later defuzzified. The inference process is divided into two steps: First, a matching operation is used to determine the activation of the rules according to the similarity between input fuzzy sets and the rule antecedents. This operation basically consists in finding the intersection between the input fuzzy set and the antecedent's membership function, as shown in Fig. 2.2. Thus, it is usually included in the fuzzification block. In the second step, the actual inference is carried out using the procedure described in section 2.1.3.

Two main types of fuzzy controllers are found in the literature, depending on the type of consequents used in the fuzzy rules: Mamdani [Mamdani, 1974] and Takagi-Sugeno-Kang [Sugeno, 1985a,b; Takagi and Sugeno, 1985]. In the former, fuzzy sets or precise crisp values are used to define the rule consequents, whilst in the latter a numeric function of the input values is used. Thus, the rules of a Mamdani fuzzy system are of the form (2.29), whereas for TSK systems the rules are defined as

$$\text{IF } X \text{ is } P \text{ AND } Y \text{ is } Q \text{ THEN } Z \text{ is } F(x, y) \tag{2.30}$$

where P and Q are fuzzy sets and $F(x, y)$ is a polynomial of the input values, generally linear, i.e., $F(x, y) = p_0 + p_1x + p_2y$. A TSK system is said to be of order $\mathcal{O}$ when it uses polynomials of order $\mathcal{O}$ in its rule consequents, with $\mathcal{O}$ being an integer. Sugeno originally proposed first-order TSK systems [Sugeno and Kang, 1988; Takagi and Sugeno, 1985], although systems of order 0 [Pomares et al., 2004] and 2 [Herrera et al., 2005] are also widely used. It is important to note that in the case of zero-order consequents, TSK and Mamdani systems with crisp consequents are equivalent.

In most cases, first or higher order TSK systems require less rules than Mamdani system. However, TSK rules are also more difficult to interpret. Moreover, when the rules are extracted from expert knowledge, it may be difficult to translate this knowledge into a polynomial representation.

2.2.4 Defuzzification

Finally, the defuzzifier combines the output fuzzy sets obtained in the inference process and transforms the result into a crisp numeric output value. This process is highly important; for instance, in control applications the actuators only accept numeric signals. Many works devoted to defuzzification strategies can be found in the literature [Filev and Yager, 1993; Hellendoorn and Thomas, 1993; Zhao and Govind, 1991].

For TSK fuzzy systems, the most commonly used defuzzification strategies are the weighted average and the weighted sum. For Mamdani systems, a wide variety of methods exist, including the center-of-area, the center-of-gravity, the center-of-sums, several height methods, etc. A detailed description and a comparative between them can be found in [Pomares, 2000].

2.3 Type-2 fuzzy logic

Quite often, the knowledge used to construct the rules in a fuzzy logic system is uncertain [Liang and Mendel, 2000]. Among the many sources of uncertainty, the following can be considered as the most relevant [Hagras, 2004]:

- Uncertainties in the *inputs* to the FLS, which translate into uncertainties in the inputs' or antecedents' membership functions (in the non-singleton and singleton case respectively) as the sensors' measurements are affected by high noise levels from various sources. In addition, the input sensors can be affected by the conditions of observation, i.e., their characteristics can be changed by the environmental conditions such as wind, sunshine, humidity, rain, etc.

- Uncertainties in the control *outputs*, which translate into uncertainties in the consequents' membership functions of the FLS. Such uncertainties can result from the change of the actuators' characteristics, which can be due to wear, tear, environmental changes, etc.

- *Linguistic* uncertainties, since the meaning of words that are used in the antecedents' and consequents' linguistic labels can be uncertain, i.e., words mean different things to different people [Mendel, 2001]. In addition, experts do not always agree and they often provide different consequents for the same antecedents. A survey of experts will usually lead to a histogram of possibilities for the consequent of a rule; this histogram represents the uncertainty about the consequent of a rule [Mendel, 2001].

- Uncertainties associated with the use of noisy *training data* that could be used to learn, tune or optimize the FLS.

All these uncertainties translate into uncertainties about fuzzy set membership functions [Mendel and John, 2002]. In recent years, it has been argued that classic fuzzy sets (also called type-1 fuzzy sets), as described in section 2.1.1, have limited capabilities to directly handle data uncertainties, in the sense that they may not be able to directly model and minimize the effect of these uncertainties [Mendel, 2007b].

Type-2 fuzzy sets were introduced by Zadeh in 1975 [Zadeh, 1975] as an extension of the concept of ordinary (type-1) fuzzy set, with the purpose of modeling the aforementioned uncertainty [Lughofer, 2011]. The main characteristic of type-2 fuzzy sets is that they have grades of membership that are themselves fuzzy [Liang and Mendel, 2000].

The structure of a type-2 fuzzy logic system (or controller) is similar to its type-1 counterpart (see section 2.2), although it includes an additional element, the type-reducer. Type-2 FLSs are credited to be a good alternative when the circumstances are too uncertain to determine exact membership grades, e.g., when training data is corrupted by noise [Liang and Mendel, 2000]. In the following subsections we describe in detail type-2 fuzzy sets and systems. The structure of a type-2 fuzzy system is presented in section 2.4

2.3.1 Type-2 fuzzy sets. Concepts and definitions

A type-2 fuzzy set is characterized by a fuzzy membership function, i.e., the membership level for each element of this set is a fuzzy set in $[0, 1]$. It is important to note the difference with a type-1 fuzzy set where the membership level is a crisp number in $[0, 1]$ [Hagras, 2004]. The MFs in type-2 fuzzy sets are three-dimensional and include a footprint of uncertainty. These two elements provide additional degrees of freedom that make it possible to directly model and handle uncertainties [Mendel and John, 2002; Mendel, 2001].

As it may be inferred from the previous description, new terminology appeared with the introduction of type-2 fuzzy sets. In this subsection we present the main concepts needed in order to use type-2 fuzzy logic as a model of linguistic uncertainty. In general, we use a tilde over the symbols to distinguish type-2 concepts from their type-1 counterparts, e.g., $\tilde{A}$ denotes a type-2 fuzzy set. Most of the contents of this subsection are taken from [Hagras, 2004; Mendel and John, 2002; Mendel, 2001].

Definition 19. Primary variable: In the context of type-2 fuzzy logic, a primary variable $x \in X$ is the main variable of interest, e.g., *temperature*, *speed*. Note that this definition is equivalent to the notion of linguistic variable considered in traditional (type-1) fuzzy logic.

Definition 20. Primary membership: The primary membership of $x \in X$ is an interval $J_x \subseteq [0, 1]$ of membership values defined for each value of the primary variable x.

Definition 21. Secondary variable: Each element of the primary membership $u \in J_x \subseteq [0, 1]$ is called secondary variable.

Definition 22. Type-2 fuzzy set and type-2 membership function: A type-2 fuzzy set $\tilde{A}$ in the universe of discourse X is defined as a set of ordered pairs of the form

$$\tilde{A} = \{((x,u), \mu_{\tilde{A}}(x,u)) \quad | \quad x \in X,\ u \in J_x \subseteq [0,1]\} \tag{2.31}$$

where $\mu_{\tilde{A}}(x,u)$ is the type-2 membership function that characterizes the type-2 fuzzy set, which satisfies $0 \leq \mu_{\tilde{A}}(x,u) \leq 1$. Note that when uncertainties disappear, the type-2 MF is reduced to a type-1 membership function, in which case the secondary variable u is equal to $\mu_A(x)$ and $0 \leq \mu_A(x) \leq 1$, i.e., the third dimension disappears.

An alternative definition for a type-2 fuzzy set is given by

$$\tilde{A} = \int_{x \in X} \int_{u \in U} \mu_{\tilde{A}}(x,u)/(x,u) \tag{2.32}$$

where $J_x \subseteq [0,1]$ and the integrals denote the logical union over all admissible values of x and u.

The type-2 membership function $\mu_{\tilde{A}}(x,u)$ is also called a secondary grade, and some authors use $f_x(u)$ to denote it.

Definition 23. Footprint of uncertainty: The bounded region that represents the uncertainty in the primary memberships of a type-2 fuzzy set $\tilde{A}$ is called footprint of uncertainty (FOU). Formally, the FOU is the union of all the primary memberships, i.e.,

$$\mathrm{FOU}(\tilde{A}) = \bigcup_{x \in X} J_x \tag{2.33}$$

It has been shown that, regardless of the choice of primary membership function (triangular, trapezoidal, Gaussian, etc.), the FOU is about the same [Mendel and Wu, 2002]. Moreover, according to [Mendel and Wu, 2002], the FOU of a type-2 fuzzy set also handles the rich variety of choices that can be made for a type-1 MF, in the sense that using type-2 MFs instead of type-1 MFs diminishes the importance of the issue of which type of MF to choose.

Definition 24. Secondary membership function (vertical slice, secondary set): At each specific value x' of the primary variable x, the 2-D plane whose axes are u and $\mu_{\tilde{A}}(x',u)$ is called a vertical slice of $\mu_{\tilde{A}}(x,u)$ or secondary membership function. Formally, it is defined as

$$\mu_{\tilde{A}}(x = x', u) \equiv \mu_{\tilde{A}}(x') = \int_{u \in J_{x'}} f_{x'}(u)/(u) \tag{2.34}$$

where $J_{x'} \subseteq [0,1]$ and $0 \le f_{x'} \le 1$. Since this formulation is valid $\forall x' \in X$, the prime notation in $\mu_{\tilde{A}}(x')$ is generally dropped and $\mu_{\tilde{A}}(x)$ is referred to as a secondary membership function. Note that $\mu_{\tilde{A}}(x)$ is a type-1 fuzzy set, also referred to as as a secondary set. Generally, the name used to describe the entire type-2 MF is associated with the name of the secondary membership function.

Definition 25. Domain of a secondary membership function: The domain of a secondary membership function is the primary membership of x, i.e., $J_x \subseteq [0,1]$.

Definition 26. Embedded type-2 set: For discrete universes of discourse X and U, an embedded type-2 set $\tilde{A}_e$ has N elements, one each from $J_{x_1}, J_{x_2}, ..., J_{x_N}$, i.e., $u_1, u_2, ..., u_N$, with the associated secondary grades $f_{x_1}(u_1), f_{x_2}(u_2), ..., f_{x_N}(u_N)$. Formally,

$$\tilde{A}_e = \sum_{i=1}^{N} \left[f_{x_i}(u_i)/u_i \right]/x_i \quad u_i \in J_{x_i} \subseteq U = [0,1] \tag{2.35}$$

Note that set $\tilde{A}_e$ is embedded in $\tilde{A}$.

Definition 27. Embedded type-1 set: Given discrete universes of discourse X and U, an embedded type-1 set A_e contains one element from $J_{x_1}, J_{x_2}, ..., J_{x_N}$, namely $u_1, u_2, ..., u_N$. Formally,

$$A_e = \sum_{i=1}^{N} u_i/x_i \quad u_i \in J_{x_i} \subseteq U = [0,1] \tag{2.36}$$

The set A_e is the union of all the primary memberships of set $\tilde{A}_e$ in (2.35).

Definition 28. Type-2 singleton: A type-2 fuzzy set $\tilde{A}$ is a singleton if it only has a single point of nonzero membership, i.e., $\mu_{\tilde{A}}(x) = 1/1$ for $x = x'$ and $\mu_{\tilde{A}}(x) = 1/0$ for all other $x \neq x'$.

Definition 29. Interval type-2 fuzzy set: A type-2 fuzzy set $\tilde{A}$ is an interval type-2 fuzzy set if all its secondary grades are equal to 1. This condition is formally expressed as

$$\forall x \in X : \; f_x(u) = 1, \; \forall u \in J_x \subseteq [0,1] \tag{2.37}$$

Note that, in this case, all the secondary membership functions are interval (type-1) sets that reflect a uniform uncertainty at the primary memberships of x. Since all the secondary grades in an interval type-1 set are unity, this interval

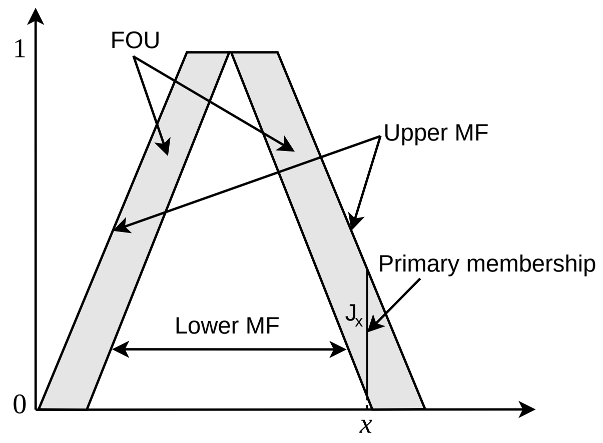


FIGURE 2.3: Example of a triangular interval type-2 fuzzy set with uncertain center. The lower MF is triangular and the upper MF is trapezoidal. The upper vertexes of the trapezoid represent the uncertain center of a triangular type-1 membership function. The grey area depicts the FOU

set can be represented by its left and right end-points as $[l, r]$. An example of an interval type-2 fuzzy set is depicted in Fig. 2.3. The figure also shows the FOU (grey area) and the primary membership J_x.

Definition 30. Lower and upper membership functions: The lower and upper bounds of the footprint of uncertainty of an interval type-2 fuzzy set are two type-1 MFs, called lower and upper membership functions, respectively. The lower MF (denoted $\underline{\mu}_{\tilde{A}}(x)$) is a subset that has the minimum membership values of the FOU. Analogously, the upper MF (denoted $\overline{\mu}_{\tilde{A}}(x)$) is a subset that has the maximum membership values of the footprint of uncertainty. Note that, for a given input x', the lower and upper membership values correspond to the left and right end-points of the interval set mentioned in Definition 29. In Fig. 2.3, the lower and upper MFs are indicated with arrows.

In the literature about interval type-2 fuzzy systems, it is common to use the term FOU to refer to the difference between the upper and the lower MFs. Hence, if there is no risk of ambiguity, we will use this term indistinctly with this double meaning, i.e., to refer to the union of all the primary memberships, as described in Definition 23, and to indicate the difference between the upper and lower MFs.

General type-2 FLSs are computationally expensive because they require a process of type-reduction that is very intensive [Liang and Mendel, 2000]. This process is simpler when interval type-2 fuzzy sets are used, which makes interval type-2 FLSs more suitable for control problems [Hagras, 2004]. For this reason, in this dissertation we consider only interval type-2 fuzzy sets and systems.

2.3.2 Operations on interval type-2 fuzzy sets

The set-theoretic operations of type-2 sets and the properties of the membership values of such sets were initially studied in [Mizumoto and Tanaka, 1976]. The algebraic structure of type-2 sets was later analyzed in [Mizumuoto and Tanaka, 1981] and [Nieminen, 1997]. Finally, the general formula for the extended sup-star composition of type-2 relations presented in [Karnik et al., 1999] is the base of the type-2 FLS theory to handle uncertainties in the FLS parameters [Liang and Mendel, 2000; Mendel, 2001].

For type-2 fuzzy sets in general, and interval type-2 fuzzy sets in particular, two new operators, named *meet* and *join*, are defined to account for the intersection and union [Hagras, 2004]. These operators are described in detail for general type-2 fuzzy sets in [Karnik and Mendel, 2001a]. Here, we present the definition of the meet and join operators for the case of interval type-2 fuzzy sets [Liang and Mendel, 2000].

For the following definitions, we consider two type-2 fuzzy sets $\tilde{A}$ and $\tilde{B}$ in the universe of discourse X. Their membership values are respectively given by $\mu_{\tilde{A}}(x) = \int_u f_x(u)/u$ and $\mu_{\tilde{B}}(x) = \int_w g_x(w)/w$, where $u, w \in J_x$ are the primary memberships and $f_x(u), g_x(w) \in [0,1]$ are the secondary grades of $x \in X$. In the case of interval sets, the membership values are simplified to $\mu_{\tilde{A}}(x) = \int_u 1/u$ and $\mu_{\tilde{B}}(x) = \int_w 1/w$, and the sets are represented by the left and right end-points of the intervals $[l_A, r_A] \subseteq [0,1]$ and $[l_B, r_B] \subseteq [0,1]$, respectively.

2.3.2.1 Intersection. Meet operator

Definition 31. Intersection of type-2 fuzzy sets: Using Zadeh's Extension Principle [Zadeh, 1975], the intersection of two type-2 fuzzy sets $\tilde{A}$ and $\tilde{B}$ is defined as [Karnik and Mendel, 2001a]:

$$\tilde{A} \cap \tilde{B} \Leftrightarrow \mu_{\tilde{A}\cap\tilde{B}}(x) = \mu_{\tilde{A}}(x) \sqcap \mu_{\tilde{B}}(x) = \int_u \int_w \left(f_x(u) \star g_x(w)\right) / (u \star w) \tag{2.38}$$

where $\sqcap$ is the meet operator, $\star$ represents a t-norm, and the integrals indicate logical union.

Definition 32. Meet operator: Under minimum or product t-norm, the meet operator for two interval sets is defined by

$$Q = \tilde{A} \sqcap \tilde{B} = \int_{v \in Q} 1/q = \int_{v \in [l_A \star l_B, r_A \star r_B]} 1/(u \star w) \tag{2.39}$$

Equivalently, the meet of n interval type-1 sets $A_1, ..., A_n$ represented by $[l_1, r_1], ...,$ $[l_n, r_n] \subseteq [0, 1]$ is an interval set whose end-points are given by $[(l_1 \star l_2 \star ... \star l_n), (r_1 \star r_2 \star ... \star r_n)]$.

2.3.2.2 Union. Join operator

Definition 33. Union of type-2 fuzzy sets: Using Zadeh's Extension Principle, the union of two type-2 fuzzy sets $\tilde{A}$ and $\tilde{B}$ is defined as [Karnik and Mendel, 2001a]:

$$\tilde{A} \cup \tilde{B} \Leftrightarrow \mu_{\tilde{A} \cup \tilde{B}}(x) = \mu_{\tilde{A}}(x) \sqcup \mu_{\tilde{B}}(x) = \int_u \int_w (f_x(u) \star g_x(w)) /(u \vee w) \tag{2.40}$$

where $\sqcup$ is the join operator, $\star$ represents a t-norm, $\vee$ is the maximum t-conorm, and the integrals indicate logical union.

Definition 34. Join operator: The join operator for two interval sets is defined by

$$Q = \tilde{A} \sqcup \tilde{B} = \int_{v \in Q} 1/q = \int_{v \in [l_A \vee l_B, r_A \vee r_B]} 1/(u \vee w) \tag{2.41}$$

Equivalently, the join of n interval type-1 sets $A_1, ..., A_n$ represented by $[l_1, r_1], ...,$ $[l_n, r_n] \subseteq [0, 1]$ is an interval set represented by $[(l_1 \vee l_2 \vee ... \vee l_n), (r_1 \vee r_2 \vee ... \vee r_n)]$.

2.3.2.3 Complement. Negation operator

Definition 35. Complement of a type-2 fuzzy set: Using Zadeh's Extension Principle, the complement of a type-2 fuzzy set $\tilde{A}$ is defined as [Karnik and Mendel, 2001a]:

$$c(\tilde{A}) \Leftrightarrow \mu_{c(\tilde{A})}(x) = \neg \mu_{\tilde{A}}(x) = \int_u f_x(u)/(1 - u) \tag{2.42}$$

where $\neg$ is the negation operator and the integral indicates logical union.

2.3.3 Type-2 fuzzy logic systems as universal approximators

In section 2.1.4, the universal approximation property of type-1 fuzzy controllers was discussed. In an analogous way, the issue of whether type-2 fuzzy systems are universal approximators or not is highly important. Again, we remember that the universal approximation property refers to the capability of fuzzy logic systems to approximate any real multivariate continuous function to any degree of accuracy on a compact domain.

The development of an approximation theory for type-2 fuzzy systems is still in its infancy [Ying, 2009]. Up to present, the only works reported in the literature that cover this topic are those of H. Ying [Ying, 2008, 2009; You and Ying, 2010]. These three works apply a two-step constructive proof for general type-2 Mamdani fuzzy systems [Ying, 2008], interval type-2 TSK fuzzy systems with linear consequents [Ying, 2009], and interval type-2 boolean fuzzy systems [You and Ying, 2010]. First, they prove that type-2 fuzzy systems can uniformly approximate any polynomial to an arbitrary accuracy, and then, use the Weierstrass approximation theorem to generalize the first step to any multivariate function.

2.4 Structure of a type-2 fuzzy logic system

The structure of a type-2 fuzzy logic system (or controller) is depicted in Fig. 2.4. It is similar to the structure of a type-1 fuzzy logic controller (FLC) shown in Fig. 2.1, with the difference that an additional block is included before the defuzzification, i.e., the type-reducer, which generates a type-1 fuzzy set. A type-2 FLS is also characterized by a set of IF-THEN rules, but the antecedents and/or consequents are type-2 fuzzy sets instead of type-1 sets. The use of type-2 FLSs is recommended when the circumstances are too uncertain to determine exact membership values [Hagras, 2004].

The operation of a type-2 FLS is as follows: First, the crisp input values are fuzzified into type-2 fuzzy sets and are fed to the inference engine. In the inference process, the rules in the rule base are activated by the input type-2 sets to produce output type-2 sets. These output type-2 sets are processed by the

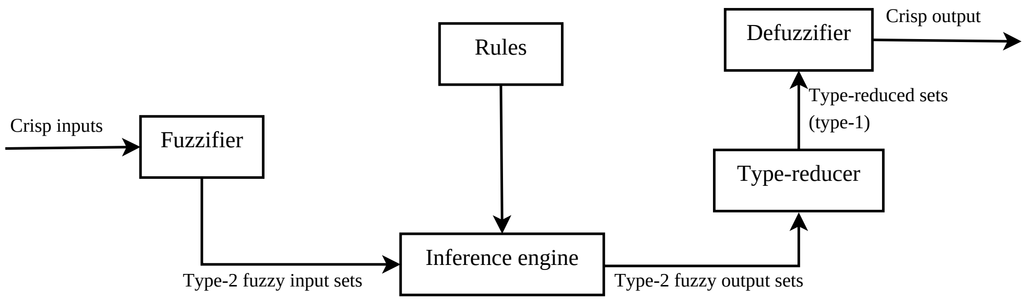


FIGURE 2.4: Structure of a type-2 fuzzy logic system

type-reducer, which combines them and performs a centroid calculation that results in type-1 fuzzy sets, called type-reduced sets. Finally, the type-1 sets are defuzzified to produce crisp outputs. In this dissertation, we consider FLSs with multiple inputs and a single output, thus, a single crisp output value is produced for each input vector. In the following subsections, we describe the operation of the five blocks just mentioned [Hagras, 2004; Mendel, 2001].

2.4.1 Fuzzification

The fuzzifier maps a crisp input vector with n inputs $\vec{x} = [x_1, x_2, ..., x_n]^T \in X_1 \times X_2 \times ... \times X_n \equiv \mathbf{X}$ into n input type-2 fuzzy sets $\tilde{A}_x$. The same types of fuzzification used for type-1 fuzzy systems (namely singleton and non-singleton) can be used for type-2 fuzzy systems. Nonetheless, for type-2 systems, singleton fuzzification is usually preferred, as it is fast to compute [Hagras, 2004] and conceptually easier.

For interval type-2 fuzzy sets, the result of the singleton fuzzification process is a type-1 interval defined by its left and right end-points $[l_x, r_x]$. Fig. 2.5 depicts an example of the fuzzification process. It is easy to observe the similarity with the fuzzification process used for type-1 fuzzy sets (see Fig. 2.2).

As for type-1, a type-2 FLS is named after the type of fuzzification used. Hence, for interval type-2 FLSs we define:

Definition 36. Singleton interval type-2 fuzzy system: A fuzzy system that uses interval type-2 fuzzy sets and singleton fuzzification is called a singleton interval type-2 fuzzy system.

Definition 37. Non-singleton interval type-2 fuzzy system: A fuzzy system that uses interval type-2 fuzzy sets and non-singleton fuzzification is called a

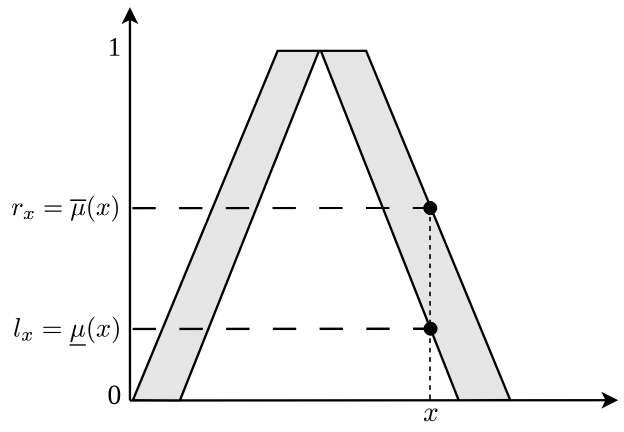


FIGURE 2.5: Singleton fuzzification with an interval type-2 fuzzy set

non-singleton interval type-2 fuzzy system.

It is important to highlight that in this dissertation, we only consider singleton interval type-2 fuzzy systems.

2.4.2 Rule base

According to [Mendel, 2001], the rules in a type-2 FLS remain the same as in type-1 FLSs (see section 2.2.2), with the difference that the antecedents and/or the consequents are represented by type-2 fuzzy sets. Thus, for a MISO system with n inputs $x_1 \in X_1, ..., x_n \in X_n$ and one output $y \in Y$, the i-th rule is given by

$$\mathfrak{R}^i : \text{IF } x_1 \text{ is } \tilde{F}_1^i \text{ AND } x_2 \text{ is } \tilde{F}_2^i \text{ AND... } x_n \text{ is } \tilde{F}_n^i \text{ THEN } y \text{ is } \tilde{G}^i \tag{2.43}$$

where $\tilde{F}_1^i, ..., \tilde{F}_n^i$ are the input type-2 fuzzy sets and $\tilde{G}^i$ is the output type-2 fuzzy set used to define the consequent of the rule.

2.4.3 Fuzzy inference engine

The inference engine combines the rules in the rule base to provide a mapping from input type-2 sets to output type-2 sets. The antecedents in the rules are connected through the meet operation, the membership values in the input set are combined with those in the output sets using the extended sup-star composition [Karnik et al., 1999], and the rules are combined through the join operation [Hagras, 2004].

Each fuzzy rule of the form (2.43) can also be expressed as [Mendel, 2001]

$$\mathfrak{R}^i : \tilde{F}_1^i \times \tilde{F}_2^i \times ... \times \tilde{F}_n^i \to \tilde{G}^i \equiv \tilde{A}^i \to \tilde{G}^i \tag{2.44}$$

where $\tilde{A}^i$ represents the rule's antecedent set, $i = 1, ..., N_r$, and N_r is the number of rules in the rule base. This rule is described by the membership function $\mu_{\mathfrak{R}^i}(\vec{x}, y) = \mu_{\mathfrak{R}^i}(x_1, ..., x_n, y)$. According to [Mendel, 2001], $\mu_{\mathfrak{R}^i}(\vec{x}, y) = \mu_{\tilde{A}^i \to \tilde{G}^i}(\vec{x}, y)$ is given by

$$\mu_{\mathfrak{R}^i}(\vec{x}, y) = \mu_{\tilde{F}_1^i}(x_1) \sqcap ... \sqcap \mu_{\tilde{F}_n^i}(x_n) \sqcap \mu_{\tilde{G}^i}(y) = \left[\prod_{a=1}^{n} \mu_{\tilde{F}_a^i}(x_a) \right] \sqcap \mu_{\tilde{G}^i}(y) \tag{2.45}$$

When singleton fuzzification is used, the input type-2 set $\tilde{A}_x$ contains a single element $\vec{x}'$ and each $\mu_{\tilde{x}_a}(x_a)$ is non zero only at one point $x_a = x'_a$ [Hagras, 2004]. Additionally, in this dissertation we consider the meet operation under product t-norm [Liang and Mendel, 2000]. Therefore, the result of the input and antecedent operations is an interval type-1 set, called the firing set, of the form

$$F^i(\vec{x}') \equiv \prod_{a=1}^{n} \mu_{\tilde{F}_a^i}(x'_a) = \left[\underline{f}^i(\vec{x}'), \overline{f}^i(\vec{x}') \right] \equiv \left[\underline{f}^i, \overline{f}^i \right] \tag{2.46}$$

where $\underline{f}^i(\vec{x}')$ and $\overline{f}^i(\vec{x}')$ represent the lower and upper firing of the rule and can be written as

$$\underline{f}^i(\vec{x}') = \underline{\mu}^i_{\tilde{F}_1}(x'_1) \cdot ... \cdot \underline{\mu}^i_{\tilde{F}_n}(x'_n) \tag{2.47}$$

$$\overline{f}^i(\vec{x}') = \overline{\mu}^i_{\tilde{F}_1}(x'_1) \cdot ... \cdot \overline{\mu}^i_{\tilde{F}_n}(x'_n) \tag{2.48}$$

2.4.4 Type reduction

Before defuzzifying the output fuzzy set to produce a crisp output value, type-2 FLSs require that the output type-2 fuzzy set obtained in the inference process is transformed into a type-1 set. This process is known as type reduction and was proposed in [Karnik et al., 1999]. The resulting type-1 set is known as the type-reduced set.

The process of type reduction is computationally intensive, specially in the case of general type-2 sets [Liang and Mendel, 2000]. Big effort has been put in the research of more efficient type-reduction methods [Liu et al., 2012; Mendel, 2005, 2007a; Wu and Mendel, 2009; Wu and Nie, 2011], thus leading to multiple type-reduction algorithms. Some of the most widely used methods are the following [Karnik and Mendel, 2001b; Mendel, 2001]:

- *Height and modified height type reduction*. This method replaces each output type-2 set by a type-2 singleton, which is situated at the point of maximum primary membership in the output type-2 set. In the modified version, the same idea is applied, although a multiplying scaling factor is applied to the rules' firing levels. They are both simple methods but present problems when only one rule is fired.

- *Centroid type reduction*. This method combines all the fired rule-output type-2 fuzzy sets through the join operator and then calculates the centroid on the resulting set. It is a computationally expensive method, because it requires a high number of centroid calculations.

- *Center-of-sets type reduction*. In this method, each consequent type-2 set is replaced by its centroid, which is a type-1 interval set. Then, the weighted average of all the centroids is computed. It has a reasonable computational cost that lies between the height and the centroid type reduction methods.

- *Center-of-sums type reduction*. This method combines the output type-2 sets by adding them and then finds the centroid of the resulting type-2 set. Its computational cost is similar to that of the centroid type reduction.

In this dissertation, we consider the Karnik-Mendel (KM) algorithm for the center-of-sets type reduction, for which we provide a description below. We have selected this method because it avoids the problems of the simple height methods at a reasonable computational cost [Hagras, 2004]. The description of the remaining methods can be found in [Mendel, 2001].

The type-reduced set produced by the center-of-sets method is an interval type-1 set that can be expressed as [Mendel, 2001]

$$Y_{cos}(\vec{x}) = [y_l, y_r] = \int_{y^1 \in [y_l^1, y_r^1]} \cdots \int_{y^{N_r} \in [y_l^{N_r}, y_r^{N_r}]} \int_{f^1 \in [\underline{f}^1, \overline{f}^1]} \cdots \int_{f^{N_r} \in [\underline{f}^{N_r}, \overline{f}^{N_r}]} 1 / \frac{\sum_{i=1}^{N_r} f^i y^i}{\sum_{i=1}^{N_r} f^i} \tag{2.49}$$

where N_r is the number of rules. The value y^i $(i = 1, ..., N_r)$ represents the centroid of the type-2 interval consequent set $\tilde{G}^i$ of the i-th rule. It is a type-1 interval set determined by its leftmost and rightmost values, i.e., $y^i = [y_l^i, y_r^i]$. Similarly, f^i denotes the firing level of the i-th rule, which is also a type-1 interval of the form $[\underline{f}^i, \overline{f}^i]$.

The calculation of the type-reduced sets is divided into two steps: First, the centroid of each consequent type-2 set has to be obtained. Then, the type-reduced set is actually computed. The following subsections describe these two steps, following the notation used in [Hagras, 2004].

2.4.4.1 Computing the centroids of the rule consequents

The centroid of the i-th output fuzzy set is a type-1 interval set given by

$$y^i = [y_l^i, y_r^i] = \int_{\theta_1 \in J_{y_1}} \cdots \int_{\theta_z \in J_{y_z}} 1 / \frac{\sum_{z=1}^{Z} y_z \theta_z}{\sum_{z=1}^{Z} \theta_z} \tag{2.50}$$

To obtain the limits of this interval set, the iterative procedure developed in [Mendel, 2001] is used. In this procedure, the values of y are minimized and maximized with respect to $(\theta_1, ..., \theta_z)$ to obtain y_l^i and y_r^i, respectively; i.e., the following function has to be optimized:

$$y(\theta_1, ..., \theta_z) = \frac{\sum_{z=1}^{Z} y_z \theta_z}{\sum_{z=1}^{Z} \theta_z} \tag{2.51}$$

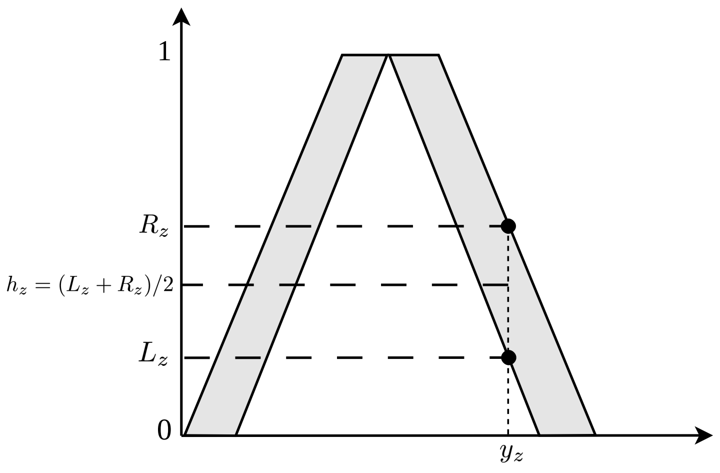


FIGURE 2.6: Parameters needed for each y_z in the procedure for computing the centroid of the rule consequents in an interval type-2 FLS

Thus, the following steps are followed to obtain the right end y_r^i of the interval:

1. Discretize the output fuzzy set into Z points $y_1, ..., y_Z$. Without loss of generality, we assume that the values y_z are arranged in ascending order, i.e., $y_1 \leq y_2 \leq ... \leq y_z$. Additionally, define the following elements:

$$J_{y_z} = [L_z, R_z] \tag{2.52a}$$

$$h_z = (L_z + R_z)/2 \tag{2.52b}$$

$$\Delta_z = (R_z - L_z)/2 \tag{2.52c}$$

   Fig. 2.6 shows how to find the values L_z and R_z for each y_z.

2. Initialize θ_z by setting $\theta_z = h_z$ for $z = 1, ..., Z$, and compute $y' = y(\theta_1, ..., \theta_z)$ using equation (2.51).

3. Find e $(1 \leq e \leq Z - 1)$ such that $y_e \leq y' \leq y_{e+1}$.

4. Update the values θ_z according to

$$\theta_z = \begin{cases} h_z - \Delta_z & \text{if } z \leq e \\ h_z + \Delta_z & \text{if } z > e \end{cases} \tag{2.53}$$

   and compute $y'' = y(\theta_1, ..., \theta_z)$ using equation (2.51) with the new values of θ_z.

5. If $y'' = y'$, then y'' is the maximum value of $y(\theta_1, ..., \theta_z)$ and this is the value of y_r^i. Otherwise, go to step 6.

6. Set $y' = y''$ and go to step 3.

The procedure for computing the lower end y_l^i of the centroid interval is the same, although replacing (2.53) by

$$\theta_z = \begin{cases} h_z + \Delta_z & \text{if } z \leq e \\ h_z - \Delta_z & \text{if } z > e \end{cases} \tag{2.54}$$

It has been proven that this procedure converges in at most Z iterations to find each of the limits of the centroid interval [Mendel, 2001].

2.4.4.2 Computing the type-reduced sets

To obtain the type-reduced set $Y_{cos}(\vec{x})$ defined in (2.49), its left and right end-points have to be computed. In the center-of-sets method [Mendel, 2001], the firing strength f^i of each rule is attached to the centroid of the corresponding consequent y^i. For the sake of clarity, we use f_l^i to denote the lower firing level associated to the lower limit y_l^i of the centroid, and f_r^i to denote the upper firing level associated to the upper limit y_r^i of the centroid. From (2.49), it is seen that

$$y_l = \frac{\sum_{i=1}^{N_r} f_l^i y_l^i}{\sum_{i=1}^{N_r} f_l^i} \tag{2.55}$$

$$y_r = \frac{\sum_{i=1}^{N_r} f_r^i y_r^i}{\sum_{i=1}^{N_r} f_r^i} \tag{2.56}$$

Thus, the following iterative procedure is used to compute y_r [Mendel, 2001]. Without loss of generality, we assume that the pre-computed centroids y_r^i are arranged in ascending order, i.e., $y_r^1 \leq y_r^2 \leq ... \leq y_r^{N_r}$:

1. Initialization step:

(a) For every rule $i = 1, ...N_r$, initialize f_r^i as $f_r^i = (\underline{f}^i + \overline{f}^i)/2$, where $\underline{f}^i$ and $\overline{f}^i$ have been previously computed according to (2.47) and (2.48), respectively.

(b) Compute $y_r' = y_r$ as given in (2.56).

2. Find R ($1 \leq R \leq N_r - 1$) such that $y_r^R \leq y_r' \leq y_r^{R+1}$.

3. Update the firing levels according to

$$f_r^i = \begin{cases} \underline{f}^i & \text{if } i \leq R \\ \overline{f}^i & \text{if } i > R \end{cases} \tag{2.57}$$

and compute $y_r'' = y_r$ using equation (2.56).

4. If $y_r'' \neq y_r'$, then set y_r' equal to y_r'' and go to step 2. Otherwise, stop and return $y_r \equiv y_r''$.

The value R computed in this method is very important, as it defines whether the lower or upper firing is used for each rule [Hagras, 2004]. Thus, for $i \leq R$, $f_r^i = \underline{f}^i$, whereas for $i > R$, $f_r^i = \overline{f}^i$. Hence, the right end y_r of the type-reduced interval, given in (2.56), can be rewritten as

$$y_r = \frac{\sum_{a=1}^{R} \underline{f}^a y_r^a + \sum_{b=R+1}^{N_r} \overline{f}^b y_r^b}{\sum_{a=1}^{R} \underline{f}^a + \sum_{b=R+1}^{N_r} \overline{f}^b} \tag{2.58}$$

The procedure for computing the left end y_l is similar to the one given for y_r, with the following changes:

- Use (2.55) to compute y_l where (2.56) is used to obtain y_r (steps 1b and 3).
- In step 2, find L ($1 \leq L \leq Z - 1$) such that $y_l^L \leq y_l' \leq y_l^{L+1}$.
- In step 3, replace (2.57) by the following equation to update f_l^i:

$$f_l^i = \begin{cases} \overline{f}^i & \text{if } i \leq L \\ \underline{f}^i & \text{if } i > L \end{cases} \tag{2.59}$$

Analogously to y_r, the left end y_l of the type-reduced interval, given in (2.55), can be rewritten as

$$y_l = \frac{\sum_{u=1}^{L} \overline{f}^u y_l^u + \sum_{v=L+1}^{N_r} \underline{f}^v y_l^v}{\sum_{u=1}^{L} \overline{f}^u + \sum_{v=L+1}^{N_r} \underline{f}^v} \tag{2.60}$$

This iterative method is guaranteed to converge in no more than N_r iterations to find y_r and no more than N_r iterations to find y_l, where N_r is the number of rules in the rule base [Liang and Mendel, 2000; Mendel and Wu, 2002].

2.4.5 Defuzzification

Finally, the type-reduced set $Y_{cos}(\vec{x})$ is defuzzified to produce a crisp output value. As proposed in [Liang and Mendel, 2000] and [Mendel, 2001], the average of the left and right end-points of this interval is used as the type-2 FLS final output, i.e.,

$$y = \frac{y_l + y_r}{2} \tag{2.61}$$

Chapter 3

Evolving fuzzy control. Concepts and state of the art

Nowadays control applications can be found in many different fields, such as robotics, industrial applications, ambient intelligence or health care, among others. The underlying theories and techniques have evolved greatly since their appearance in the early 20th century, leading to a nowadays mature control theory. However, as the complexity of the systems increase, so does the need of new methodologies capable of coping with uncertainties, modeling problems and complexity. Thanks to the improvement of computational systems, soft-computing techniques can be applied to control systems to improve their performance and to allow them to tackle more challenging problems.

In this chapter, we present a brief review of the evolution of control systems, from their origins to state-of-the-art intelligent controllers that apply computational intelligence techniques to self-design. We pay special attention to fuzzy controllers, which are the focus of this dissertation. Section 3.1 provides a brief insight into the historical evolution of control systems. Evolving fuzzy systems are throughly used in this dissertation as the paradigm for designing intelligent controllers. Their properties, requirements and goals are described in section 3.2. Finally, in section 3.3, we study the state of the art on the use of fuzzy systems for control applications.

3.1 Introduction. Historical origins of intelligent control

Automatic control has played a fundamental role in the advance of science and engineering [Ogata, 2001]. The term "automatic control" is used to describe a wide variety of methods that aim to minimize the need of human intervention in control systems in order to maximize their efficiency. Despite the huge diversity of processes that can be controlled (generally named *plants*), their basic underlying properties are similar and can be approached with a common mathematical control theory.

Probably, the first significant work in the field of automatic control was James Watt's centrifugal governor for the speed control of steam engines, in the 18th century. However, a theory capable of explaining the underlying principles did not appear until 1868, when James Maxwell [Maxwell, 1868] introduced the analysis of systems through their linear differential equations.

Classical control theory was mainly developed during the first half of the 20th century [Ogata, 2001]. In 1922, Minorsky applied automatic controllers to steering ships and showed how stability could be determined from the differential equations describing the system [Minorsky, 1922]. In 1932, Nyquist developed a relatively simple procedure for determining the stability of close-loop systems [Nyquist, 1932]. In 1934, Hazen discussed the design of relay servomechanisms capable of closely following a changing input [Hazen, 1934].

During the decade of the 1940s, frequency-response methods (especially those based on the Bode diagram [Bode, 1945]) allowed engineers to design linear closed-loop control systems that satisfied performance requirements. In the early 1950s, Evans' root-locus method [Evans, 1950] was fully developed. These two methods are the core of classical control theory. The systems designed using them were in general acceptable, but not optimal in any meaningful sense [Ogata, 2001]. For this reason, in the late 1950s, the attention in the control field shifted to the design of "optimal" systems (in some specific sense).

The complexity of modern plants has increased over the years. As a consequence, the description of a control system now requires a large number of equations. Classical control theory deals with single-input-single-output systems and is unable to properly handle these complex, nonlinear systems [Ogata, 2001]. In the 1960s, the availability of digital computers made possible the development of the so-called modern control theory, which is based on time-domain analysis and synthesis using state variables [Gupta, 2000]. This new theory was able to handle the increasing complexity of modern plants and to meet the requirements on accuracy, weight and cost [Ogata, 2001]. Until the late 1980s, this control theory was widely used in multiple fields and lead to the research on optimal and adaptive control techniques.

However, the methods developed still relied heavily on rigorous mathematical concepts [Gupta, 2000]. The lack of a well-established theory for the analysis of nonlinear systems, together with the appearance of new more challenging requirements for control systems (uncertainties, operation on changing environments, speed, accuracy, etc.), caused the necessity of new advanced methods. In response to this need, in the last decade of the 20th century, soft-computing techniques (such as fuzzy logic, neural networks and evolutionary computation) were applied to control problems with good results [Andersen et al., 1997; Chen and Khalil, 1995; Lee, 1990; Ordonez et al., 1997]. This synergy lead to the appearance of the so-called "intelligent control", which is treated in this dissertation. In the last decade, intelligent control techniques have continued evolving, increasing their capability to cope with the requirements mentioned above [Bleris et al., 2007; Hagras, 2007; Rojas et al., 2006].

3.2 Evolving fuzzy systems

Fuzzy logic systems (both type-1 and type-2) are a widely used paradigm to represent uncertain knowledge. They provide a reliable trade-off between precise and linguistic modeling and offer a tool for solving complex problems in the real world [Lughofer, 2011]. However, the main obstacle in the design of fuzzy rule-based systems is the proper generation of their structure (rule base, membership functions, linguistic labels) and parameters [Angelov and Buswell, 2002].

In the last decade, a new topic has emerged in the field of data-driven design of fuzzy systems: the so-called evolving fuzzy systems, which aim to account for changing system dynamics and behaviors, as well as new operating conditions and environmental influences [Lughofer and Angelov, 2011]. The main property of an evolving fuzzy system is that it develops both its structure and its parameters on demand, based on new incoming information [Angelov et al., 2010]. In this section, we present the main concepts related to this new paradigm, which represents the main core of this dissertation.

3.2.1 From first principle to evolving models

Mathematical models can be divided into four classes [Lughofer, 2011], namely (i) first principle models, (ii) knowledge-based models, (iii) data-driven models, and (iv) hybridization of the former types. First principle models are those which are analytically deduced from well-established theories about physical, chemical, biological processes and other laws. They are often also called analytical models. This is the case of exact formulas that describe non-changing dependencies, differential equations that describe dynamic processes, etcetera [Luo, 2009]. They are usually deduced once and kept fixed afterwards. Their main advantage is that they provide accurate representations of these processes; however, they present the drawback that they are difficult or even impossible to obtain in some cases. Moreover, real-life applications often do not comply with the rigorous assumptions under which the basic conclusions of this theory have been made [Angelov and Buswell, 2002].

Knowledge-based models are extracted from the long-term experience of experts, operators or users of the underlying system. This expert knowledge is usually represented as linguistic rules, leading to the so-called knowledge-based systems or expert systems [Castillo, 1991]. Fuzzy logic systems are a particular case of this type of model. When the rules are properly defined, this type of model may even cover extreme cases that cannot be captured by analytical models [Lughofer, 2011]. However, obtaining these rules from experts may be very time consuming or difficult [Chen et al., 2008] and may be affected by subjective moods or ideas.

Data-driven models appeared to overcome the difficulties mentioned above. This type of model includes any form of mathematical model that is fully designed, extracted or learned from data [Lughofer, 2011]. Opposed to knowledge-based models, they avoid the impact of subjectivities by using objective observations (data) provided by the underlying system. This information may come from several different sources, such as measurement sensors, structured data (e.g., databases) or data streams. The objective nature of the data often makes a data-driven model more trustworthy than a knowledge-based one. Nonetheless, the generalization capability of these models highly depends on the quality of the training data used in the learning process [Bouchachia and Mittermeir, 2006].

Evolving models (or evolving systems) are a particular case of data-driven models. Their main property is that they are automatically adapted, extended and evolved dynamically on-the-fly based on new incoming data samples [Lughofer, 2011]. Thus, any evolving system presents the ability to expand or shrink its structure, as well as to to adapt its parameters, in an incremental way, online and, if necessary, in real time [Angelov et al., 2010]. They are able to support any modeling scenarios for data streams and dynamic data that change their nature and characteristics over time and space.

In this context, it is important to differentiate between adaptive and evolving models (see Table 3.1). The former usually refers to the update of some model parameters, whilst the latter also takes the extension of the model into account, i.e., by generating new structural components on demand. It is also worth noting that evolving models are permanently updating their knowledge by integrating new system behaviors and environmental influences in a life-long learning process that is never really terminated [Lughofer and Angelov, 2011].

Finally, it is to be clarified the distinction between the evolutionary and the evolving approach. In the former, structure and parameters are learned based on genetic operators in an iterative process that uses all the data in each step. Moreover, a complete population of individuals is evolved in order to finally obtain a solution. On the contrary, the evolving approach refers to the system's ability to incorporate new data on-the-fly and to gradually adapt (evolve) its structure and parameters to accommodate such data [Angelov et al., 2010; Lughofer, 2011]. Moreover, in the evolving paradigm, a single solution is handled, instead of a population. Table 3.1 summarizes the differences among the adaptive, the evolving and the evolutionary paradigms.

TABLE 3.1: Comparison of adaptive, evolutionary and evolving systems

	Adaptive	Evolutionary	Evolving
Parameters	Dynamic	Dynamic	Dynamic
Structure	Predefined	Predefined or dynamic	Dynamic
Type of learning	Adaptation Online	Genetic operators Offline	Incremental Online
Data used for learning	As they arrive	All the data	As they arrive
Solutions handled	One	Several (population)	One

3.2.2 Incremental learning in evolving systems

In order to obtain systems that are capable of gradually learning and evolving their structure, techniques of incremental learning are usually applied. Incremental learning can be defined as a sample-wise or block-wise training of a model or system, either from scratch or starting with an already available one [Lughofer, 2011]. Incremental learning techniques can be classified into three groups, depending on the use and storage of the data [Maloof and Michalski, 2004]:

- Full instance memory methods store all the data points received. This option presents several drawbacks, mainly the cost of relearning the models from all the data seen so far and the need to store all the previous data [Cara et al., 2011a; Lughofer, 2011], which may cause the memory to explode.

- No instance memory methods (also known as single-pass incremental learning) represent the opposite case, in which the data are processed immediately and discarded afterwards [Bouchachia and Mittermeir, 2006]. This alternative has the advantage of requiring low memory and computational effort. Nonetheless, it also may lead to robustness problems, e.g., they are strongly affected by the order in which the data points are received [Dagher et al., 1999].

- Partial instance memory methods are intermediate solutions that store only part of the data points received. It has been stated that learners with partial instance memory react more quickly to drifting concepts than learners that do not store any past information [Maloof and Michalski, 2004]. Nonetheless, the way in which the data to be stored are selected is important, as it may lead to forgetting previously learned situations [Lughofer, 2011]. A way to tackle this issue is to store a limited amount of data points that provide a uniform representation of the data space considered [Cara et al., 2010c].

Online learning is also an important concept in the context of evolving systems. This type of learning can be defined as the training process that takes place during online operation modes (e.g., during the normal operation of the system), which implies that the system is permanently trained and expanded with new data collected from the online process.

Finally, incremental learning methods in general, and evolving systems in particular, have to tackle different trade-offs, known as the stability-plasticity dilemma and the accuracy-interpretability trade-off. These issues are described in the following subsections.

3.2.2.1 Stability-plasticity dilemma

Evolving systems are confronted with the stability-plasticity dilemma. This dilemma is concerned with learning new knowledge (plasticity) without forgetting the previously learned one (stability) [Bouchachia, 2010; Grossberg, 1988]. On the one hand, a system is totally stable when it is able to keep all the acquired knowledge without any catastrophic forgetting [Bouchachia and Mittermeir, 2006]. On the other hand, a system is completely plastic if it is able to continually incorporate new knowledge, even if that implies discarding the previously acquired information.

In addition, it can be considered that stability is achieved when the system's components (i.e., rules, in the case of fuzzy systems) are not changed or are changed very slightly [Lughofer, 2011]. In this sense, stability is related to parameter convergence, as small changes in the structure usually do not disturb this convergence. On the contrary, plasticity is related to significant movements of structural components, which are required to accommodate the new information.

Obviously, plasticity is fundamental in the context of evolving systems, as it is the key to the adaptation to new situations as they appear. If learning stops too soon, later events cannot be incorporated. However, if new data always cause important modifications in the system, earlier learning may be forgotten. Since these two desirable qualities are in conflict, an evolving methodology should try to find a compromise between stability (in order to guarantee convergence) and plasticity (in order to guarantee that new information is included in the system) [Lughofer, 2011]. This problem has been thoroughly studied by many researchers [Bouchachia, 2010; French, 1999; Grossberg, 1988].

3.2.2.2 Accuracy-interpretability trade-off

Another important issue in the design of fuzzy systems (whether they are evolving or not) is the trade-off between accuracy and interpretability. The former refers to the system's capability of performing its task with high precision, whilst the latter refers to its ability to express this behavior in an understandable way from the human point of view [Cordón, 2001]. It is clear that accuracy and interpretability are to a considerable extent opposing properties when training fuzzy systems [Herrera et al., 2011b]. As a consequence, many researchers have studied this problem recently [Casillas, 2003; Casillas et al., 2005, 2003; Herrera et al., 2005; Johansen and Babuska, 2003].

In general, the two types of fuzzy system introduced in section 2.2.4 affect this trade-off in a different way [Herrera et al., 2011b]. On the one hand, Mamdani-type systems focus on interpretability, as they use linguistic labels for both the antecedents and the consequents. On the other hand, TSK fuzzy systems are considered to be more accurate due to the use of linear functions in the consequents. This provides a better approximation capability although it also makes these systems less interpretable.

Traditionally, learning mechanisms for fuzzy systems (adaptive, evolutionary, evolving, etc.) have concentrated on the improvement of the accuracy, whilst interpretability has been left as a secondary aspect [Cordón, 2011; Gacto et al., 2009]. Nonetheless, it must be remembered that one of the motivations for the use of fuzzy systems is their higher interpretability with respect to other type of systems (neural networks, genetic programming, etc.) [Lughofer, 2011]. Therefore, it is important that evolving fuzzy systems use techniques that allow for the balancing of these two aspects.

3.2.3 Essential requirements and goals for evolving fuzzy systems

In this section, we conclude our introduction to EFSs by pointing out the main demands that can be made to this type of intelligent systems. From a general point of view, there are two main requirements that should be fulfilled by an evolving system [Lughofer, 2011]:

- The performance of the evolving system should be as close as possible to the performance achieved by its hypothetical batch version. Note that a batch learning algorithm can use all the data for its learning and may even perform several iterations over the data to refine its parameters. Thus, it will usually achieve a stable solution with high accuracy. For this reason, it would be desirable to achieve a similar performance with an online learning method that only "sees" part of the data.
- The evolving system should achieve the so-called *convergence to optimality* with respect to a certain optimization criterion. This requirement means that at least some parameters should converge to the optimal solution achieved by the hypothetical batch method mentioned before. This requirement is closely related to the stability-plasticity dilemma.

In order to achieve these two requirements, an evolving fuzzy system must modify its structure and parameters. More specifically, the following elements should be adapted/evolved:

- The system's linear and nonlinear parameters must be adapted and refined as demanded by the incoming data. The rule consequents in TSK systems are linear parameters, whereas the fuzzy sets in the antecedents are nonlinear. Usually, different methods are required for the adaptation of linear consequents and the update of nonlinear parameters.

- In order to allow the fuzzy system to cope with the appearance of new operation states, its core components should be evolved. This is usually achieved by adding new rules and antecedent fuzzy sets.

- In the case of problems with high-dimensional feature spaces, it would be desirable that the evolving system was capable of selecting those features that are relevant for its operation.

Finally, there are some additional properties that can be desirable in an EFS, even if they are not strictly necessary. Thus, the following enhanced goals can be defined for evolving fuzzy systems:

- The guarantee of process-safety can be a necessity in critical applications in which some states should never be reached. Thus, an evolving system could include mechanisms to guarantee proper operation and to prevent system crashes. Nonetheless, in order to achieve this property, some previous knowledge about such undesired states should be known beforehand, i.e., an expert should provide this information.

- The interpretability of the system should be preserved, as this is one of the key reasons for the use of fuzzy systems.

- Finally, evolving systems could present increased performance an usability with respect to other types of systems. This requirement is not strictly necessary, as the fact that evolving systems can adapt to the environment by themselves is a strong advantage by itself. Nonetheless, it represents an additional nice-to-have feature.

3.3 State of the art

Over the past several years, fuzzy systems have achieved a considerable success in the field of control [Chang, 2010; Gao and Er, 2005; Jiang and Han, 2008; Pomares et al., 2004; Rojas et al., 2006; Wagner and Hagras, 2007; Wang et al., 2008]. The main reasons for this are that fuzzy controllers do not require an accurate model of the plant to be controlled and that they allow for the application of human expert knowledge [Chang, 2010]. Moreover, in [Castro, 1995] it was proven that fuzzy controllers are universal approximators, in the sense that they are able to approximate any continuous function in a compact set to any desired accuracy.

However, there is no systematic approach for the design of fuzzy controllers [Rajapakse et al., 2002]. As a consequence, obtaining high quality controllers is never an easy task [Wagner and Hagras, 2007]. The way a fuzzy controller is designed depends on the available knowledge about the dynamics of the system to be controlled. In Fig. 3.1, we suggest a possible way to classify fuzzy controllers according to the previous information and the type of design used to obtain them. In this section, we follow this taxonomy to review the state of the art on fuzzy control.

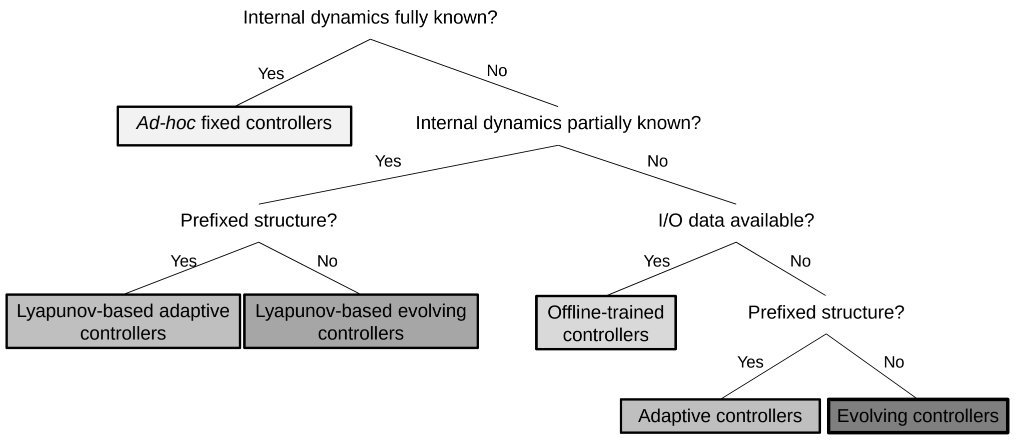

Internal dynamics fully known?
Yes
No
Ad-hoc fixed controllers
Internal dynamics partially known?
Yes
No
Prefixed structure?
Yes
No
Lyapunov-based adaptive controllers
Lyapunov-based evolving controllers
I/O data available?
Yes
No
Offline-trained controllers
Prefixed structure?
Yes
No
Adaptive controllers
Evolving controllers

FIGURE 3.1: A possible classification of fuzzy controllers according to their design technique

If the internal dynamics of the plant (i.e., its differential equations) and their properties are well known, this information can be studied and used to design stable controllers [Galluzzo and Cosenza, 2009; Lalouni et al., 2009]. This way, we obtain *ad-hoc* fixed controllers specifically designed for the plant to be controlled. Their main advantage is that their stability can be studied and proved. Moreover, the design can be optimized to obtain an optimal controller under a certain optimality criterion (i.e., minimum number of rules, adequate topology, etc.). However, this approach presents some problems: Firstly, the design process can be very time-consuming, or even impossible for complex systems.

Secondly, the controllers need to be redefined for every different plant, even if they are similar. Finally, they cannot cope with changes in the dynamics of the plant if the designer was not aware of the possibility of their occurrence and considered them during the design.

On the other hand, for real-world complex plants, it is a common situation that only reduced knowledge (e.g., some properties of the differential equations, such as their upper bounds), or even no knowledge at all, is available. In such cases, other alternative methodologies have to be used, such as data-driven techniques that are able to automatically obtain the controller parameters.

The literature shows a wide variety of methods for the development of fuzzy controllers when enough knowledge about the plant is available to make assumptions about its equations, such as the existence of certain bounds. In [Liu and Zheng, 2009; Wang et al., 2008], the authors assume that the approximation error is bounded and use this information to obtain Lyapunov-based update laws for the parameters of the controller. In [Bellomo et al., 2008], the filtered prediction error is combined with the tracking error to provide a modified composite adaptation law for the rule consequents. In all these cases, the structure of the fuzzy system is fixed and only the parameters are updated online. Hence, these methods could be classified as Lyapunov-based adaptive controllers. Their main advantage is that the stability of the closed-loop system is guaranteed. However, the rule base has to be defined beforehand, i.e., the membership functions and the rules have to be identified. In order to provide a good definition of the rule base, some knowledge about the plant is needed. If this is not the case, the resulting structure may be too complex or too simple to properly respond to the nonlinearities of the plant.

There are also methods that perform an online modification of the controller's structure [Gao and Er, 2003; Park et al., 2005; Phan and Gale, 2008]. These methods also base their update laws on the Lyapunov theory, thereby obtaining stable controllers, although Phan and Gale [2008] were the only ones to prove the stability over the structural changes. Although, these Lyapunov-based evolving methods need less prior knowledge than their adaptive counterparts, they still share the disadvantage of requiring that certain assumptions about the plant are satisfied.

In [Park et al., 2005], the controller's structure is modified whenever an input value falls outside the range defined by the current MFs. This method uses a complete set of rules; hence, when the aforementioned criterion is satisfied, a new MF and all the related rules are added to the knowledge base. The advantage of this method is that it only places membership functions in those regions of the input space that are actually reached by the controller. However, the rules are evenly distributed in this explored space, which may lead to a high concentration of MFs in regions with low nonlinearities or not enough functions in regions where the nonlinearities are high.

Gao and Er [2003] proposed a neuro-fuzzy controller that adapts online both the parameters and the structure. This method adds new rules based on two criteria, namely the ϵ-completeness and the system error, and deletes rules based on the concept of error reduction ratio. It provides a scattered distribution of the rules, thereby placing more of them in the areas with higher nonlinearities. However, the computational cost of this technique is high, as large matrix computations are performed in every step and all the previous input/output data have to be stored.

In [Phan and Gale, 2008], the same criteria, as given in [Gao and Er, 2003], are used for the addition of new membership functions. The new MF is added to the input that achieved the maximum activation degree in the rule with the maximum firing strength. The idea behind this selection criterion is that a large system error indicates that membership functions in that input are not sufficient to represent the nonlinearity in the region. However, it causes high sensibility to noise, as the method will be sensitive to the presence of outliers. This method does not require storing any past input/output (I/O) data. This reduces the memory requirements, although it also causes that all the decisions made by the algorithm only take into account the current situation, i.e., the algorithm may be short-sighted.

The worst possible case for the design of a controller occurs when it is not possible to make any assumptions about the equations of the plant, due to the lack of knowledge. Obviously, in this case, techniques for the automatic design of the controller are needed the most. When there are I/O data available, this task can be performed offline with methods based on pre-training. In [Chen et al., 2009; Lin and Xu, 2006; Mingzhi et al., 2009; Mucientes and Casillas, 2007], evolutionary algorithms are used for parameter learning in controllers with a predefined topology. In [Li and Lee, 2003] and [Juang, 2008], the controller's topology is also learned, providing more flexible solutions, as they do not depend on the ability of the designer to choose the rules. However, all these approaches show some drawbacks: On the one hand, since the controller is learned offline, it works as a fixed controller and it is not able to cope with changes in the dynamics of the plant. On the other hand, the I/O data used for the pre-training may be difficult, or even impossible, to obtain.

When no I/O data are available, online techniques have to be applied. Multiple methods exist for the online adaptation of parameters such as the rule consequents or the membership functions [Andersen et al., 1997; Pomares et al., 2004; Rojas et al., 2006; Zarandi, 2008]. In [Zarandi, 2008], reinforcement learning is used for tuning the parameters of a fuzzy logic controller. In this approach two fuzzy systems are used: an Actor and a Critic. The former uses the input state to determine the next action; the latter combines the information about the action suggested by the Actor with a reinforcement signal to produce an internal reinforcement that is later used to update both fuzzy systems. In [Rojas et al., 2006], three types of parameters are adapted: output scale factors, rule consequents and position of the membership functions. In [Andersen et al., 1997], the error at the controller output is used in a gradient-descent technique to tune the parameters of the controller. Finally, in [Pomares et al., 2004], the error at the plant output is used for a coarse tuning of the rule consequents, whilst the error at the controller output is used for a fine tuning of the membership functions and the consequents. These adaptive techniques share the inconvenient of requiring a prior definition of the structure of the fuzzy controller, and the related problems that have been already commented.

Finally, the case of the online learning of the structure of the fuzzy controller when no prior knowledge about the plant is available has been seldom studied [Angelov, 2004]. The most relevant work found in the literature regarding this subject is authored by P. Angelov [2004]. In this work, the controller's structure is modified by using the information given by every new sample of training

TABLE 3.2: Summary of the properties of the different methodologies for the design of fuzzy controllers[a]

	Prev. knowledge	Type of design	Param. adaptation	Structure learning	Stability proven	Copes with changes
Ad-hoc controller	Full	Offline	✗	✗	✓	✗
Lyapunov-based adaptive	Partial	Online	✓	✗	✓	✻
Lyapunov-based evolving	Partial	Online	✓	✓	✻	✓
Offline-training	I/O data	Offline	✓	✻	✗	✗
Adaptive	Reduced	Online	✓	✗	✗	✻
Evolving	Minimum	Online	✓	✓	✗	✓

[a] Legend: ✓: Yes/property fulfilled; ✻: In some cases/property partially fulfilled; ✗: No/property not present

data. As a consequence, different sequences of training data may produce different structures for the controller. In some cases, as when working in noisy environments, this type of method may not be sufficiently robust.

Other related works can be found in the literature [de Barros and Dexter, 2007; Lin et al., 1995], although they do not completely fulfill the requirements of truly evolving fuzzy controllers. For instance, one of the first contributions in this area was made in [Lin et al., 1995]. In this work, the authors proposed an online method for parameter and structure learning based on training data. However, the fuzzy controller was trained offline and was not actually used to control the plant until the learning had been completed. Hence, the proposed method could be considered as an offline learning method, in the sense that learning does not happen while controlling the plant. A more recent example is found in [de Barros and Dexter, 2007], which proposes an evolving model-based controller. Although the evolving model does not need previous information about the plant, the system uses a fixed fuzzy model to avoid problems during the initial moments, until the evolving model is fully trained. This fixed fuzzy model is trained offline with I/O data from an approximate linear model of the plant. The design of this approximate model, even if it is simple, requires some knowledge about the plant that may not be available.

The properties of the design methods just mentioned are summarized in Table 3.2. From the previous discussion, it becomes clear that obtaining methods able to work with the less-possible information, while the control process is being performed, is a very challenging issue for current research. Nevertheless, relaxing the need of assumptions and the conditions imposed to the plant's dynamics is not costless. It is important to note that these methods cannot rely on mathematical models of the plants, as they are unknown. As a consequence, providing a formal proof for the stability of these controllers becomes a difficult (or even impossible) task. Thus, in many cases, the validity of these methods can be proven only by experimental results [Angelov, 2004].

Chapter 4

OSEFC: Online Self-Evolving Fuzzy Controller

The online evolution and learning of fuzzy systems is highly important when dealing with changing environments. This capability is especially relevant in the field of control, due to the special characteristics of control problems. In this field, techniques capable of developing controllers with a minimum amount of prior knowledge about the plants are desired.

In this chapter a novel fuzzy controller, which is able to self-design from scratch while working online, is proposed. The controller does not use the information regarding the differential equations that govern the plant's behavior or any of their bounds. The methodology presented is able to determine an adequate topology for the fuzzy controller based on the data obtained during the system's normal operation. Therefore, the controller can start operating with an empty set of fuzzy rules and does not need any offline training. The proposed methodology comprises two phases: adaptation of the consequents for every selected topology and online addition of new membership functions. Some of the main advantages of this method are its robustness under changes in the plant's dynamics, its good performance in noisy situations and its ability to perform variable selection among a group of candidate variables. Unlike other online methods, the modification of the topology is based on the analysis of the whole operating region of the plant, thereby providing higher robustness.

4.1 Motivation and goals

As commented in previous sections, fuzzy systems have been successfully applied to many control problems over the past several years. The main reasons for this are the following [Gao and Er, 2005; Jiang and Han, 2008; Wang et al., 2008; Ying, 2000]:

- They need no accurate mathematical models of the system under control.
- They allow for the application of human expert knowledge about the operation of such systems.
- They are universal approximators, in the sense that they are able to approximate any continuous function in a compact set to any desired accuracy [Castro, 1995; Wang and Mendel, 1992].

Unfortunately, there is not a systematic approach for the design of fuzzy controllers [Rajapakse et al., 2002]. Thus, obtaining high quality controllers is never an easy task. The worst-case scenario occurs when the internal dynamics of the plant are not known and it is not possible to make any assumptions about the equations that govern it, due to the lack of knowledge. In this case, techniques for the automatic design of controllers are needed the most. If, in addition, there is no I/O data available, offline learning techniques, such as evolutionary algorithms, cannot be applied and online learning becomes necessary.

As mentioned before, multiple methods exist for the online adaptation of parameters (e.g., adaptation of the rule consequents or the membership functions) [Andersen et al., 1997; Pomares et al., 2004; Rojas et al., 2006; Zarandi, 2008], but they require a prior definition of the structure of the fuzzy controller. On the contrary, the case of the online adaptation of the controller's structure when there is no prior knowledge about the plant has been seldom studied [Angelov, 2004]. In this work, the controller's structure is modified by using the information given by every new sample of training data. As a consequence, the order in which the data is received has a big influence on the structure obtained for the controller [Dagher et al., 1999]. In some cases, as when working in noisy environments, this type of method may not be sufficiently robust. Thus, it is observed that the problem of designing controllers capable of learning online how to control an unknown system is a very challenging one. In the particular case of fuzzy controllers, it requires both parameter learning and structure evolution, i.e., evolving fuzzy systems.

In this chapter, we aim to advance in this direction by presenting a novel Online Self-Evolving Fuzzy Controller (OSEFC) which is able to adapt both its parameters (rule consequents) and its structure (membership functions) while controlling the plant. This method is based on the concept of universal approximation capability of fuzzy systems, which assures that it is possible to reach a desired accuracy level by properly adding enough MFs and rules. The main features of OSEFC are the following:

- No model for the plant or its differential equations are needed.
- No previous knowledge about the control policy is needed. The fuzzy controller can start working with an only rule initialized to zero and will self-develop as it performs the control.
- The controller is robust, in the sense that it is able to cope with unexpected isolated changes in the behavior of the plant. As it does not use specific information about the dynamics of the plant, its operation does not rely on it. Furthermore, its adaptive nature makes the system able to learn how to control the new behavior of the plant.
- No precise knowledge regarding the control inputs is needed, as the algorithm is able to select those that are relevant for the plant state, from a set of candidate inputs.

The aim of this new methodology is to provide a means of obtaining a capable controller when more precise methods cannot be applied due the lack of information. However, this lack of information also causes some limitations: Firstly, it is not possible to provide a formal demonstration of the stability of the closed-loop system. Secondly, the proposed controller cannot be applied when there are states that must not be reached by the plant: Since we do not have information about the plant, there is no way to know which states of the plant should not be entered. In this case, some pre-training should be performed to guarantee that the control policy avoids these states when the controller initially starts operating. Finally, its application is limited to plants for which the controller's sampling period can be set in a way such that the plant's output depends directly on the control signal applied in the previous time step.

The rest of the chapter is organized as follows. In section 4.2, the problem to be solved is introduced together with the structure of the fuzzy controller employed. In section 4.3, the proposed methodology is presented, separating its two phases, i.e., rule consequents adaptation and topology modification by adding new membership functions to the inputs of the controller. Simulation and experimental results illustrating the main features of the algorithm are presented in section 4.4. Finally, conclusions are drawn in section 4.5.

4.2 Problem formulation

Let us consider a single-input single-output (SISO) plant, whose dynamics are given by a set of differential equations of the form

$$\begin{aligned} x^{(n)} &= F(\bar{x}, u) = F(x, \dot{x}, ..., x^{(n-1)}, u) \\ y &= H(\bar{x}) \end{aligned} \tag{4.1}$$

where $\bar{x} \in \mathbb{R}^n$ represents the plant's state, $y \in \mathbb{R}$ is the plant's output, $u \in \mathbb{R}$ is the control signal and F and H are unknown continuous functions.

Using a short enough sampling time T_p, the dynamics of the plant (4.1) can be expressed in terms of its difference equations [Andersen et al., 1997]

$$y(t+1) = f(\bar{x}(t), u(t)) = f(y(t), y(t-1), ..., y(t-p), u(t), u(t-1), ..., u(t-q)) \tag{4.2}$$

where $u(t)$ is the control signal exerted by the controller at time t, and p and q are natural constants that determine the complexity of the plant, by indicating how previous states and control actions affect its current output.

Let us assume that the controller's sampling time T_c (which defines the time instants at which control actions are exerted) can be selected in a way such that the plant's output at time $k+1$ depends on the control signal applied at time k, i.e., the effect of the control action $u(k)$ is reflected at the plant's output at the next time step, $y(k+1)$. Note that this does not mean that previous control actions do not affect the plant's output as well, but rather that $u(k)$ actually has an effect that is measurable at the next time step. Additionally, take notice that the sampling time T_p used to simulate the plant and the sampling time T_c used to exert control actions are not necessarily the same. In fact, if T_p is very small, T_c will generally be chosen larger in order to avoid possible instabilities caused by dead-beat behavior. Unless otherwise stated, in the following we will use the concept "time instant" to refer to the discrete time instants given by the controller's sampling time.

In order for the proposed methodology to be valid to control the plant (4.1), we impose the condition that there must always exist a control policy capable of translating the plant's output to the desired value (within the operation range) [Pomares et al., 2002b]. This means that there must never exist a state in which the plant's output does not depend on the control input. In addition, it is also required that the monotonicity of the plant with respect to the control input has a constant sign. Without loss of generality, we assume that this monotonicity is positive. To clarify this condition, let us suppose that, at a certain time instant k, the plant's state is $\hat{x}(k)$ and a control signal $u(k)$ is applied, producing the output $y(k+1)$ at the next time step. If at the same state $\hat{x}(k)$ the control signal $u'(k) > u(k)$ had been applied, then the output obtained would have been $y'(k+1) > y(k)$.

The aim of the controller is to guarantee that the plant's output tracks a given reference signal $r(k)$. In the absence of actuator bounds, we can assume that there exists an optimal control policy G capable of achieving the control objective, i.e.,

$$u(k) = G(r(k), \vec{x}(k)) \tag{4.3}$$

where $\vec{x}(k)$ is a finite set of variables that determine the control policy.

According to (4.3), the control problem may be cast as a problem of functional approximation of the plant's inverse function [Pomares et al., 2004]. To attain this approximation, we employ a zero-order Takagi-Sugeno-Kang fuzzy system with a complete set of rules [Lee, 1990] defined as

$$\text{IF } x_1 \text{ is } X_1^{i_1} \text{ AND } x_2 \text{ is } X_2^{i_2} \text{ AND... } x_N \text{ is } X_N^{i_N} \text{ THEN } u = R_{i_1 i_2 \ldots i_N}$$

where $X_v^{i_v} \in \{X_v^1, X_v^2, ..., X_v^{n_v}\}$ are the membership functions of input X_v, n_v is the number of membership functions for that input variable, and $R_{i_1 i_2 ... i_N}$ is a scalar value representing the rule consequent.

Homogeneously distributed triangular membership functions [Lee, 1990; Rojas et al., 2000] are used in the methodology presented in this chapter. With this configuration, only the center of each function needs to be stored, as the slopes of the triangles are computed according to the centers of the neighboring functions. Moreover, as an additional benefit, this configuration assures that for any input value within the input range, exactly two membership functions present a non-zero activation degree. This property greatly reduces the complexity of computations while, at the same time, guarantees that all the possible operating regions are covered by the firing of at least one fuzzy rule.

Lastly, the product is chosen as the t-norm for the inference method and the weighted average is chosen as the defuzzification strategy. Hence, the fuzzy controller's output is given by:

$$u(k) = \hat{G}(\vec{x}(k); \Theta) = \frac{\sum_{i=1}^{N_r} R_i \cdot \mu_i(\vec{x}(k))}{\sum_{i=1}^{N_r} \mu_i(\vec{x}(k))} = \frac{\sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_N=1}^{n_N} R_{i_1 i_2 ... i_N} \cdot \prod_{m=1}^{N} \mu_{X_m^{i_m}}(x_m(k))}{\sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_N=1}^{n_N} \prod_{m=1}^{N} \mu_{X_m^{i_m}}(x_m(k))} \tag{4.4}$$

where N_r is the total number of fuzzy rules, μ_i is the activation degree of the i-th rule, and $\mu_{X_m^{i_m}}$ is the activation degree of the i_m-th membership function of input X_m. In the above expression, we have explicitly highlighted the output dependency not only on the input vector, but also on the parameters of the fuzzy system, Θ. These parameters are the set of rule consequents ($R_{i_1 i_2 ... i_N}$), the number of membership functions for each input variable (n_v) and the centers of the membership functions (θ_v^j), where v denotes the input variable and j is a natural value between 1 and n_v.

Note that the challenge tackled in this chapter, i.e., the online evolution and control starting from no knowledge about the plant, is more complex than typical function approximation problems, because the approximation of (4.3) must be obtained while controlling the plant in a proper way.

4.3 Architecture of the Online Self-Evolving Fuzzy Controller

As commented before, the universal approximation property of fuzzy systems implies that it is possible to achieve any desired approximation accuracy by properly adding more membership functions and rules. This idea can be used to obtain controllers capable of learning the control policy from the beginning of their operation, with a fast convergence. The lack of knowledge about the plant makes automatic design necessary for these controllers, which also need to have adaptive parameters in order to cope with noise, disturbances or changes in the plant's dynamics. With this objective in mind, the method proposed in this chapter produces a fuzzy controller capable of a proper operation despite the fact that no prior knowledge about the plant is available.

Fig. 4.1 presents the proposed methodology integrated in a closed-loop control system. The adaptation mechanism of OSEFC is shown on the top. Internally, it consists of two blocks, namely parameter learning and self-evolution of the topology, and a memory to store I/O data collected online during the operation of the plant. All these elements are explained later in this chapter. Below, the fuzzy logic controller that is modified by the adaptation mechanism (dotted line) provides the control signal that is applied to the plant. OSEFC is the combination of the adaptation mechanism and the fuzzy logic controller. Finally, the plant produces the output y that is fed back to the controller. The Input Preprocessing block is optional. It simply represents the possibility of using different sets of inputs for the fuzzy controller.

The general flowchart of the adaptation mechanism is depicted in Fig. 4.2. No initial control parameters are available, so it starts working with an empty controller (i.e., only one membership function per input; hence, only one fuzzy rule [Pomares et al., 2002a]). The control process is carried out in two phases:

- *Parameter learning*: In this phase, the consequents of the existing rules are adapted based on the plant's output error. The method applies a reward/penalty policy that takes into account the sign of the dependence of the plant's output with respect to the control signal. This process is described in section 4.3.1.

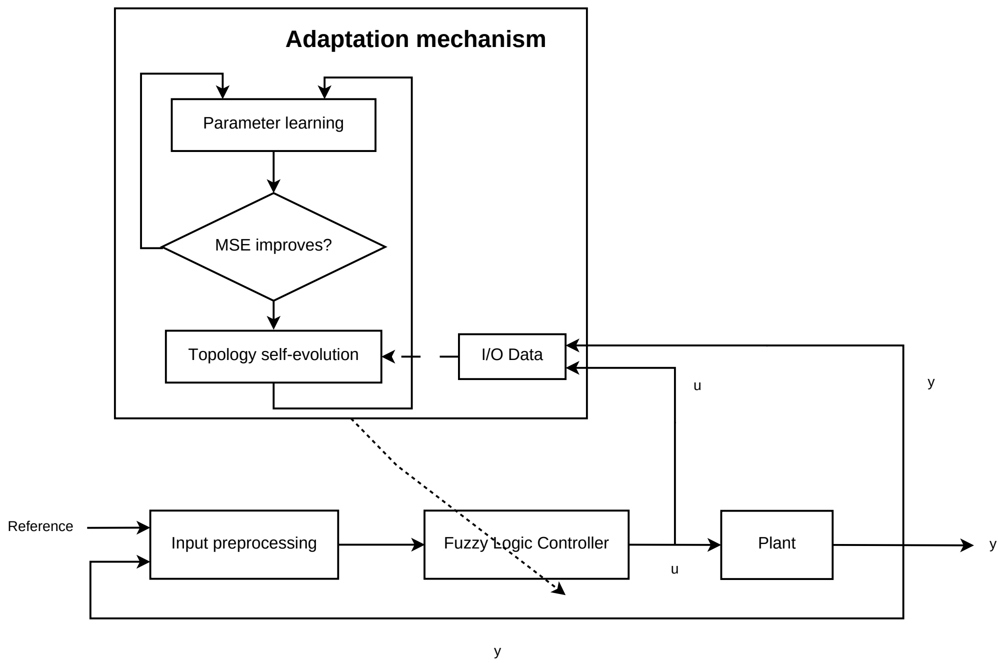


FIGURE 4.1: Representation of the closed-loop control system with OSEFC

- *Topology self-evolution*: Once the modification of the fuzzy rules no longer improves the quality of the control, the algorithm switches to this phase, in which a new membership function is added to one of the controller inputs and, hence, new fuzzy rules are created. Then, the consequents of all the rules need to be adapted again, so the algorithm switches back to the parameter learning phase. Section 4.3.2 presents a detailed explanation of the process followed to modify the topology of the controller.

It is important to note that the adaptation mechanism works online, while the control is being performed. Therefore, its application never stops.

4.3.1 Phase 1: Online parameter learning

To improve the control quality, a fuzzy controller needs to "learn" which actions lead to the stabilization of the plant. In the context of parameter adaptation, such learning is translated into the modification of the consequents of the fuzzy rules so that the new control actions may provide better results. However, the

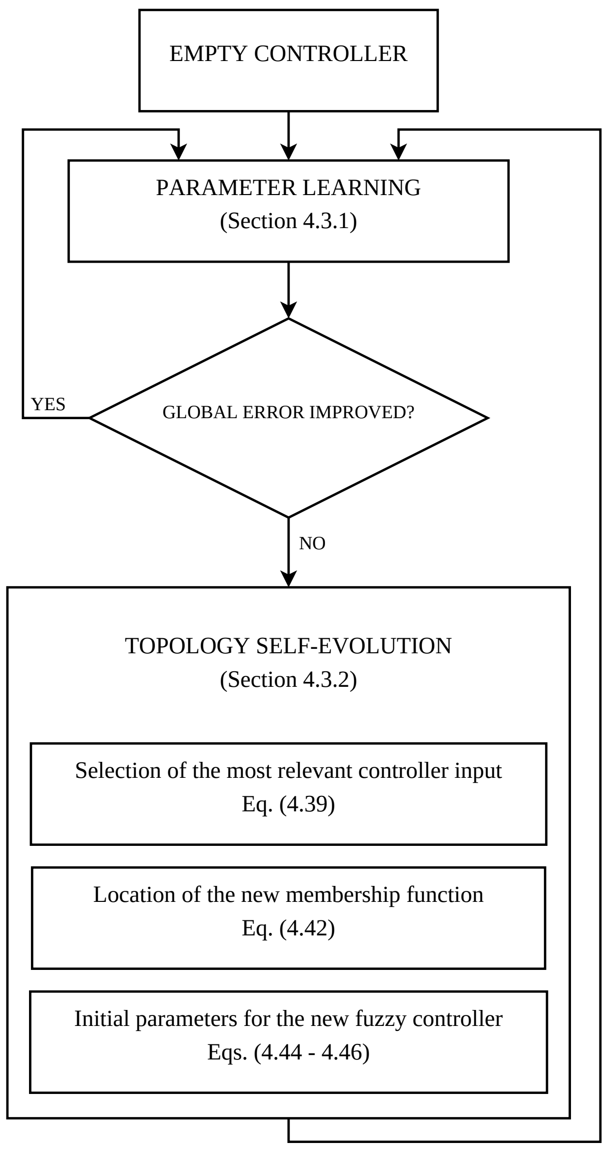


FIGURE 4.2: General flowchart of OSEFC

lack of knowledge about the plant's internal functioning makes us unaware of how to modify the controller's parameters.

In order to apply a gradient-descent algorithm, we would need to compute the partial derivative of the plant's output with respect to the control input ($\partial y/\partial u$), which leads us to two different problems: On the one hand, this derivative is unknown, since our hypothesis here is that the differential equations governing the plant are unknown. On the other hand, if sampling periods are long, it may be not possible to approximate the aforementioned derivative by $\Delta y/\Delta u$.

Nevertheless, in Section 4.2 the existence of a control policy capable of translating the plant's output to the desired reference value (within the operation range) was stated. Additionally, this condition implies that the monotonicity of the plant with respect to the control input has a constant sign. The next subsection describes an online local learning method for the rule consequents based on the sign of the monotonicity of the plant and the error at the plant's output.

4.3.1.1 Online local learning of the rule consequents

From a "local" point of view, the goal of the controller is to bring the plant's output from its current value to the desired reference value as soon as possible, i.e., ideally $y(k+1) = r(k)$, where k and $k+1$ represent consecutive control steps. At each time step, it is possible to obtain the error at the plant's output as $e_y(k) = r(k-1) - y(k)$. The combination of this error with the sign of the monotonicity of the plant with respect to the control input provides valuable information about the direction in which the rule consequents have to be moved in order to achieve the local control objective [Rojas et al., 2006].

Let us assume that the monotonicity is positive. This means that, given a state $x(k)$ of the plant, if we apply the control signal $u'(k) > u(k)$, then the output obtained for $u'(k)$ is larger than the one obtained for $u(k)$. Then, there are three possible cases regarding the error at the plant's output:

- If the error $e_y(k)$ is negative, i.e., $y(k) > r(k-1)$, it means that the control signal applied at time k was too large. To correct this behavior, the consequents of the fuzzy rules must be decreased.

- When $e_y(k) > 0$, i.e., $y(k) < r(k-1)$, the control signal was too small and the consequents must be increased.
- Finally, if $e_y(k) = 0$, the plant's output has reached the reference value and no correction is needed in the rule consequents.

In the case of negative monotonicity, the same reasoning still holds, but the changes in the consequents go in the opposite direction.

Thus, our consequent adaptation process is based on the evaluation of the current state of the plant and the proposal of a correction (either a reward or a penalty) for the rules responsible of reaching such state. Since not all the rules contributed to reaching the current state, it does not seem reasonable to apply the same penalty to all of them. A better option is modifying only those rules which were triggered to obtain the control input $u(k)$. In the same way, such modification must be proportional to their degree of activation.

Therefore, the following expression is used for the modification of the consequent of the i-th rule of the fuzzy system:

$$\Delta R_i(k) = C \cdot \mu_i(k-1) \cdot e_y(k) = C \cdot \mu_i(k-1) \cdot (r(k-1) - y(k)) \tag{4.5}$$

where $\mu_i(k-1)$ is the activation degree of the i-th rule at instant $k-1$, $r(k-1)$ is the set point at that time, and $y(k)$ is the current plant's output. C is a normalization constant that is further described in section 4.3.1.2. It is important to note the use of the reference value at instant $k-1$ in (4.5), instead of its current value $r(k)$. The reason for this is that the rules activated at instant $k-1$ aimed to reach the former and not the latter.

Finally, it must be highlighted that the whole consequent adaptation process is performed online, while the controller is operating on the real plant; therefore, control is applied from the first moment, with increasing accuracy as the adaptation evolves.

In the following subsections, some additional aspects about the parameter learning methodology are clarified.

4.3.1.2 Value of the normalization constant C

In this subsection, we analyze the effect of the normalization constant C introduced in (4.5) and establish the conditions it must fulfill [Pomares, 2000]. To do so, we consider the plant given by the difference equation (4.2) and a controller of the form (4.4). Thus, the output of the controller at time k is given by

$$u^1(k) = \hat{G}(\vec{x}(k); \Theta^1(k)) \tag{4.6}$$

where $\Theta^1(k)$ represents the parameters of the controller at time k. The superscript 1 is used in this and the following equations to refer to the original configuration of the controller. Accordingly, the plant's output obtained as a result of the application of this control signal is

$$y^1(k+1) = f(\tilde{x}(k), u^1(k)) \tag{4.7}$$

and the error at the plant's output is

$$e_y^1(k+1) = r(k) - y^1(k+1) \tag{4.8}$$

In order to correct this error, the rule consequents are modified as indicated in (4.5). Since the objective of the correction is to reduce the error at the plant's output, it is obvious that C must have the same sign as the monotonicity of the plant with respect to the control signal. Without loss of generality, we assume that this monotonicity is positive.

Let us now study the effect of this change by applying the new rules under the same conditions initially considered. In this case, the control signal would be

$$u^2(k) = \hat{G}(\vec{x}(k); \Theta^2(k)) \tag{4.9}$$

where $\Theta^2(k)$ represents the controller's modified parameters. The associated plant's output and error would then be given by

$$y^2(k+1) = f(\tilde{x}(k), u^2(k)) \tag{4.10}$$

and

$$e_y^2(k+1) = r(k) - y^2(k+1), \tag{4.11}$$

respectively. As in the initial case, the superscript 2 is used to refer to the configuration of the controller after applying the change (4.5) to the rules.

The maximum possible change in a consequent takes place when only one rule was triggered, i.e., its activation degree was equal to 1. Hence, depending on the sign of the error (4.8), there are two cases to be considered:

(i) If $e_y^1(k+1) > 0$, then the modified control signal is larger than the initial one and the following relationship holds:

$$\begin{aligned} u^1(k) \leq u^2(k) &\leq u^1(k) + C \cdot e_y^1(k+1) \\ 0 \leq u^2(k) - u^1(k) &\leq C \cdot e_y^1(k+1) \end{aligned} \tag{4.12}$$

(ii) If $e_y^1(k+1) < 0$, then the modified control signal is smaller and the relationship is

$$\begin{aligned} u^1(k) + C \cdot e_y^1(k+1) \leq u^2(k) &\leq u^1(k) \\ C \cdot e_y^1(k+1) \leq u^2(k) - u^1(k) &\leq 0 \end{aligned} \tag{4.13}$$

Additionally, if the modification of the consequents is adequate, the error in the modified case must be smaller than the initial error. Formally,

$$\left|e_y^2(k+1)\right| \leq \left|e_y^1(k+1)\right| \tag{4.14}$$

with $\left|e_y^2(k+1)\right| = \left|e_y^1(k+1)\right|$ if and only if they are both zero, i.e., the plant's output has reached the desired reference value.

On the other hand, assuming that $f(\tilde{x}(k), u(k))$ is continuous and differentiable with respect to the control signal, and since $[u^1(k), u^2(k)]$ is a closed and bounded interval, the mean value theorem establishes that there exists a value $\tilde{u}$ such that

$$y^2(k+1) = y^1(k+1) + \frac{\partial f}{\partial u}(\tilde{x}, \tilde{u})\bigg|_{\tilde{x}(k),\tilde{u}} \left(u^2(k) - u^1(k)\right) \tag{4.15}$$

As a consequence, we also have that

$$e_y^2(k+1) = e_y^1(k+1) - \frac{\partial f}{\partial u}(\tilde{x}, \tilde{u})\bigg|_{\tilde{x}(k),\tilde{u}} \left(u^2(k) - u^1(k)\right) \tag{4.16}$$

and since we assume that the monotonicity of the plant w.r.t. the control signal is positive, the term $\frac{\partial f}{\partial u}\Big|_{\tilde{x}(k),\tilde{u}}$ is a real, positive, non-zero value.

Finally, by taking absolute values in (4.16), we obtain

$$\left|e_y^2(k+1)\right| = \left|e_y^1(k+1) - \left(\frac{\partial f}{\partial u}(\tilde{x}, \tilde{u})\bigg|_{\tilde{x}(k),\tilde{u}}\right)\left(u^2(k) - u^1(k)\right)\right| \tag{4.17}$$

in which the partial derivative has a constant sign. It can be observed that the limit values of (4.17) correspond with the limit values of $u^2(k) - u^1(k)$. Therefore, we have again two cases to consider:

(i) If $u^2(k) - u^1(k) = 0$, it means that the consequents have not suffered any modification. This can only happen if the error in the initial case was zero, i.e., $e_y^2(k+1) = e_y^1(k+1) = 0$.

(ii) If $u^2(k) - u^1(k) = C \cdot e_y^1(k+1)$, equation (4.17) can be rewritten as

$$\begin{aligned}\left|e_y^2(k+1)\right| &= \left|e_y^1(k+1) - \left(\frac{\partial f}{\partial u}(\tilde{x}, \tilde{u})\bigg|_{\tilde{x}(k),\tilde{u}}\right) \cdot C \cdot e_y^1(k+1)\right| \\ &= \left|e_y^1(k+1)\right| \cdot \left|1 - \left(\frac{\partial f}{\partial u}(\tilde{x}, \tilde{u})\bigg|_{\tilde{x}(k),\tilde{u}}\right) \cdot C\right|\end{aligned} \tag{4.18}$$

and the condition (4.14) is only satisfied if

$$\left|1 - \left(\frac{\partial f}{\partial u}(\tilde{x}, \tilde{u})\bigg|_{\tilde{x}(k),\tilde{u}}\right) \cdot C\right| \leq 1 \Leftrightarrow -1 \leq 1 - \left(\frac{\partial f}{\partial u}(\tilde{x}, \tilde{u})\bigg|_{\tilde{x}(k),\tilde{u}}\right) \cdot C \leq 1 \tag{4.19}$$

Finally, this leads to the following conditions for the value of C:

$$\frac{\partial f}{\partial u}(\tilde{x}, \tilde{u})\bigg|_{\tilde{x}(k),\tilde{u}} \cdot C \geq 0 \tag{4.20a}$$

$$\frac{\partial f}{\partial u}(\tilde{x}, \tilde{u})\bigg|_{\tilde{x}(k),\tilde{u}} \cdot C \leq 2 \tag{4.20b}$$

The first condition confirms our previous idea about the sign of C being the same as the sign of the monotonicity. The second condition is more restrictive and establishes a bound for the value of C. Thus, in order to guarantee the convergence of the learning method for any state of the plant, the value of C must verify

$$C \leq \frac{2}{\left.\frac{\partial f}{\partial u}(\tilde{x}, \tilde{u})\right|_{max}} \tag{4.21}$$

where in the denominator we use the maximum value of the partial derivative, to guarantee that condition (4.20b) is satisfied for each state of the plant.

In the case of negative monotonicity of the plant with respect to the control signal, a similar approach can be used. Hence, the previous condition can be generalized as

$$|C| \leq \frac{2}{\left.\frac{\partial f}{\partial u}(\tilde{x}, \tilde{u})\right|_{max}} \tag{4.22}$$

However, our assumption is that the differential equations governing the plant are unknown. Therefore, it is not possible to compute the partial derivative in (4.22) and an approximation must be used. If we approximate the partial derivative by $\Delta y/\Delta u$ and take the mean value of (4.22), then the absolute value of C can be set offline as

$$|C| = \frac{\Delta u}{\Delta r} \tag{4.23}$$

where Δu is the range of the controller's actuator and Δr is the range in which the reference signal varies. Note that we do not use Δy directly because the range of the output values of the plant may be unknown. On the contrary, the ranges of the actuator and the reference signal are known beforehand by the user of the control system.

It is also important to highlight that the specific value of C is not critical, as long as it satisfies the condition (4.22). When it does, the value of C influences the convergence speed, but not the convergence itself.

4.3.1.3 Effect of the presence of actuator bounds

A fact that must be taken into account is that most real life controllers have limitations on their operation, which highly affects the control process. For instance, let us consider an actuator that is only able to operate within the range

$[u_{min}, u_{max}]$. If at a given moment the optimal control input to reach the reference value is $u(k) > u_{max}$, the input that will be finally applied to the plant will be u_{max}. In this case, it will not be possible to reach the desired set point at the next time step. However, the rules should not be penalized, as they are already providing the best possible answer.

To solve this inconvenience, no penalty is applied to the rules when the error is due to the actuator's limitations. Thus, the updating law for the consequents is redefined as:

$$\Delta R_i(k) = \begin{cases} 0 & \text{if } u(k-1) = u_{min} \ \& \ \Delta R_i(k) < 0 \\ 0 & \text{if } u(k-1) = u_{max} \ \& \ \Delta R_i(k) > 0 \\ \Delta R_i(k) & \text{otherwise} \end{cases} \tag{4.24}$$

4.3.1.4 Comments on the stability of the parameter learning process

As mentioned before, one of the limitations derived from the reduction of assumptions about the mathematical model of the plant is that a rigorous mathematical formulation for the proof of stability is almost impossible. Nevertheless, we provide here a qualitative approach to this important issue.

First, the most simple case of a controller with only one rule is tackled. In this case, the controller's output is given by

$$u(k) = \mu(k) \cdot R(k) = R(k), \tag{4.25}$$

as the normalized activation of the only rule is always $\mu(k) = 1$.

Let us now assume that the optimum value of the rule consequent is $R^*$and that, at time k, the consequent value is $R(k)$ and the modification to be applied is

$$\Delta R(k) = C \cdot \mu(k-1) \cdot e_y(k) = C \cdot e_y(k) \tag{4.26}$$

Then, the consequent at the next time instant will be

$$R(k+1) = R(k) + C \cdot e_y(k) = R(k) + C \cdot (r(k-1) - y(k)) \tag{4.27}$$

If we define $E_R(k) = R^* - R(k)$, we can consider the quadratic Lyapunov candidate function given by

$$V(E_R(k)) = E_R(k)^2 = (R^* - R(k))^2 \tag{4.28}$$

It is easy to verify that this function fulfills the requirements of a Lyapunov function, namely $V(0) = 0$, $V(E_R(k)) \geq 0 \, \forall E_R(k)$, and $V(E_R(k)) \to \infty$ as $E_R(k) \to \infty$.

The difference in the Lyapunov candidate function over time can be defined as

$$\begin{aligned}\Delta V(E_R(k+1)) &= V(E_R(k+1)) - V(E_R(k)) \\ &= (R^* - R(k+1))^2 - (R^* - R(k))^2 \\ &= (R^* - R(k+1) + R^* - R(k)) - (R^* - R(k+1) - R^* + R(k)) \\ &= (2 \cdot R^* - R(k+1) - R(k)) \cdot (R(k) - R(k+1))\end{aligned} \tag{4.29}$$

In order to obtain $\Delta V < 0$, the two elements in the product (4.29) must have a different sign. Without loss of generality, let us assume that the monotonicity of the plant with respect to the control input is positive and that $e_y(k) > 0$. In this case, $R(k+1) > R(k)$ and $R(k) - R(k+1) < 0$, which leads to

$$\begin{aligned}&2 \cdot R^* - R(k+1) - R(k) > 0 \\ &2 \cdot R^* - (R(k) + C \cdot e_y(k)) - R(k) > 0 \\ &2 \cdot R^* - 2 \cdot R(k) - C \cdot e_y(k) > 0 \\ &2(R^* - R(k)) - C \cdot e_y(k) > 0\end{aligned} \tag{4.30}$$

Replacing $e_y(k) = r(k) - y(k+1)$, this condition can be rewritten as

$$C < 2 \cdot \frac{R^* - R(k)}{r(k) - y(k+1)}$$

Thus, it can be concluded that the local stability of the learning mechanism depends only on the value of the parameter C. It is easy to observe that the value proposed for C in (4.23) also fulfills this condition. Moreover, it can be verified that the same condition for the value of C is obtained for the cases of negative monotonicity and negative error at the plant's output.

Now, let us consider a controller with N_r rules, which aims to approximate the optimal control policy (4.3). Let $\vec{R}(k) = [R_1(k), R_2(k), ..., R_{N_r}(k)]^T$ be the vector of current values for the rule consequents, and $\vec{R}^* = [R_1^*, R_2^*, ..., R_{N_r}^*]^T$ be the vector of optimal consequents that provide the best possible approximation for the controller's current structure. At each time step k, the rule consequents $R_i(k)$ are modified according to (4.5).

Again, if we define the vector of parameter errors at time k as $\vec{E}_R(k) = \vec{R}^* - \vec{R}(k)$, we can consider the quadratic Lyapunov candidate function

$$V(\vec{E}_R(k)) = \vec{E}_R{}^T \vec{E}_R = (\vec{R}^* - \vec{R}(k))^T(\vec{R}^* - \vec{R}(k)) = \sum_{i=1}^{N_r} (R_i^* - R_i(k))^2 \tag{4.31}$$

which also fulfills the conditions of a Lyapunov function mentioned in the previous case.

In this case, the variation on the Lyapunov candidate function is given by

$$\begin{aligned}
\Delta V(\vec{E}_R(k+1)) &= V(\vec{E}_R(k+1)) - V(\vec{E}_R(k)) \\
&= \sum_{i=1}^{N_r} (R_i^* - R_i(k+1))^2 - \sum_{i=1}^{N_r} (R_i^* - R_i(k))^2 \\
&= \sum_{i=1}^{N_r} \left[(R_i^* - R_i(k+1))^2 - (R_i^* - R_i(k))^2\right] \\
&= \sum_{i=1}^{N_r} \left[(R_i^* - R_i(k+1) + R_i^* - R_i(k))(R_i^* - R_i(k+1) - R_i^* + R_i(k))\right] \\
&= \sum_{i=1}^{N_r} \left[(2 \cdot R_i^* - R_i(k+1) - R_i(k))(R_i(k) - R_i(k+1))\right] \qquad (4.32)
\end{aligned}$$

If the sign of $(2 \cdot R_i^* - R_i(k+1) - R_i(k))$ is different from the sign of $(R_i(k) - R_i(k+1))$ for every rule, then all the elements in the sum (4.32) are negative and so is ΔV. Without loss of generality, let us assume that $C > 0$ and $e_y(k) > 0$. In this case, $R_i(k+1) > R_i(k)$ and $R_i(k) - R_i(k+1) < 0$, meaning that the following inequality must be satisfied for all the rule consequents:

$$\begin{aligned}
&2 \cdot R_i^* - R_i(k+1) - R_i(k) > 0 \\
&2 \cdot R_i^* - (R_i(k) + C \cdot \mu_i(k-1) \cdot e_y(k)) - R_i(k) > 0 \\
&2 \cdot R_i^* - 2 \cdot R_i(k) - C \cdot \mu_i(k-1) \cdot e_y(k) > 0 \\
&2(R_i^* - R_i(k)) - C \cdot \mu_i(k-1) \cdot e_y(k) > 0 \qquad (4.33)
\end{aligned}$$

Again, replacing $e_y(k)$ by its definition, the previous inequality can be rewritten as:

$$C < 2 \cdot \frac{R_i^* - R_i(k)}{\mu_i(k-1) \cdot (r(k) - y(k+1))} \qquad \forall\, i = 1, ..., N_r \qquad (4.34)$$

From this derivation it can be concluded that the local stability of the learning mechanism depends again on the value of the parameter C. Finally, for the value of C given in (4.23), we have that

$$C < 2 \cdot \frac{R_i^* - R_i(k)}{(r(k) - y(k+1))} < 2 \cdot \frac{R_i^* - R_i(k)}{\mu_i(k-1) \cdot (r(k) - y(k+1))}$$

since $\mu_i(k-1) \leq 1\, \forall i = 1...N_r$. Hence, the condition (4.34) is satisfied and the consequent learning policy leads to a locally stable controller. It is easy to verify that the same condition for the value of C is obtained for the cases of negative monotonicity and negative error at the plant's output.

4.3.2 Phase 2: Topology self-evolution

In the previous section, we presented a method for the adaptation of the rule consequents of a fuzzy controller that has a predefined configuration of MFs. Therefore, it is necessary to fix the structure of the fuzzy controller beforehand (i.e., control inputs, number of membership functions for each input, parameters of the MFs, etc.). This may not be a trivial task, as it requires certain knowledge about the system to be controlled [Phan and Gale, 2008; Rojas et al., 2006].

In order to operate without any previous knowledge, an intelligent controller should be able to adapt not only its parameters, but its full structure as well, i.e., an evolving methodology is required. This way, it will be possible to start the operation of the control system with a very simple controller, e.g., just one rule automatically initialized. Then, as the operation goes on, more rules can be added to the controller, as needed to meet the accuracy demands of the closed-loop system.

As commented in section 4.2, the problem of designing a fuzzy controller may be cast as a problem of functional approximation of the inverse function of the plant [Pomares et al., 2004]. Nevertheless, there is the additional difficulty that the only information available to tackle this problem is the data obtained from the normal operation of the closed-loop system. There are few methods reported in the literature that aim to evolve the topology of a fuzzy controller while operating on the plant [Angelov, 2004; Park et al., 2005; Phan and Gale, 2008]. Moreover, it is common that these methods base the structure modification on local information, i.e., only the data obtained from the system at the current time step. This has a negative impact on the robustness of the method [Dagher et al., 1999]:

- The structure of the resulting fuzzy system greatly depends on the sequence of data, i.e., the same data received in a different order may lead to a fuzzy system with a completely different structure.
- The methodologies may be short-sighted, i.e., the data obtained at each time step is processed and forgotten at that moment. Therefore, future decisions may not take past information into account.
- The presence of outliers may be more difficult to handle.

A way to address these issues is to base the modification of the controller's topology on a set of data instead of using just a single point. In the ideal case, a set of data that provides a uniform representation of the entire range of operation of the plant is available. Nonetheless, in practice it is very unlikely that the normal operation of the plant will span the entire operating range. For this reason, the learning of the controller's structure has to be based on a uniform representation of those regions that have been reached by the controller during its operation.

As a consequence, the information obtained may not be sufficient to perform a complete functional approximation of the plant's inverse function. Nonetheless, it may be enough to obtain a good approximation in the regions of interest, i.e., those effectively reached by the controller's normal operation. Hence, it is possible to obtain a controller that will properly control the already-visited regions, even if the approximation is somehow erratic in other areas. In the event that the control operation ever goes into those regions, the evolving mechanism will be able to learn them in order to incorporate them to the control policy.

The method proposed in this dissertation exploits the fact that the very operation of the system provides input/output data about the true inverse function of the plant to be controlled [Pomares et al., 2004]. Let us assume that, at time instant k, the control signal $u(k)$ is applied to the plant and the output value $y(k+1)$ is obtained as a result. Regardless of whether $y(k+1)$ is the desired output or not, if in the future the plant is again in the same state and the desired reference value is $r(k') = y(k+1)$, the control signal that has to be applied to the plant is $u(k)$, since $u(k)$ is capable of taking the plant output from $y(k)$ to $y(k+1)$. Hence, the pairs $(\vec{z}(k), u(k))$, where $\vec{z}(k)$ is obtained from $\vec{x}(k)$ by replacing $r(k)$ with $y(k+1)$, represent I/O data concerning the true inverse function of the plant under control. Note that this reasoning is valid as long as the relationship between the states, the control signal and the plant's output remains over time. If these I/O relationships are not reliable (e.g., in the presence of unbounded internal dynamics) or change rapidly in time, the data previously gathered are no longer valid and cannot be used to approximate the inverse function of the plant.

With the aim of using this information towards the self-evolution of the fuzzy controller, we store the I/O data provided by the plant in a memory M. Then, we use these data to decide which input variable mostly needs a new membership function. In practice, storing all the I/O data produced during the system operation will cause the memory to explode. To avoid this, we define the memory M as a fixed grid in the input space. Every time a new datum is obtained, it is stored in the corresponding hypercube defined by this grid. No weights are assigned to the data in the memory. The reason for this is as follows: In real control applications, it is common that the reference signal stays constant for a certain time. If weights were assigned, the weight assigned to the corresponding cell would grow to the point of making irrelevant the information stored in all the other cells. As a consequence, the approximation would be based only on local information. By not assigning weights, all the data stored in the memory M are equally important, and therefore, the information in M provides a uniform representation of the input space of the plant's inverse function.

The modification of the fuzzy structure involves the following steps (see Fig. 4.2), which are detailed in the next subsections:

- First, it is necessary to decide which input to the fuzzy controller will receive the new membership function. Adding MFs to all the inputs is not feasible, as the number of rules exponentially depends on the number of membership functions [Pomares et al., 2002a].

- Once the most appropriate input variable has been selected, the location of the new function must be chosen. This is done taking into account the entire error distribution, in order to reduce the method's sensibility to noise.

- Finally, the introduction of a new membership function implies the creation of new fuzzy rules. Therefore, the parameters of the new rules must be initialized in such a way that the controller's performance is minimally affected.

4.3.2.1 Selection of the most relevant controller input

One of the main questions arising when modifying the topology of a fuzzy system is the selection of the input which is going to receive the new membership function. Up to the present, only a few authors have tackled this problem offline [Li and Lee, 2003; Wang and Frayman, 2002] or online [Park et al., 2005; Phan and Gale, 2008]. Among them, some authors [Wang and Frayman, 2002] add MFs to all the inputs, locating them so that their intersection will be in the area of highest error. This alternative presents several drawbacks, which are even more dramatic when working online: On the one hand, the number of rules increases too much, due to its exponential dependency on the number of membership functions. On the other hand, minimizing the point of greatest error leads to methods which are too sensitive to noise [Phan and Gale, 2008]. Consequently, in the proposed method, we only add a membership function to one input variable at a time. This input is previously selected based on the study of the complete error surface reached by the current configuration, instead of considering only the point of maximum error.

To introduce the proposed methodology, a simple example will be used. Let us consider the plant given by [Andersen et al., 1997]:

$$y(k+1) = 0.8\sin(2(y(k)) + 1.2u(k) \tag{4.35}$$

Fig. 4.3(a) shows its inverse function, whilst Fig. 4.3(b) depicts the output given by a fuzzy configuration with two membership functions in the input $r(k)$ and four in the input $y(k)$. Visually analyzing the original function, it is possible to determine that it is preferable to go to a 2x5 configuration instead of choosing a 3x4 one, since two memberships functions are enough to approximate the linear outline of the function in dimension $r(k)$.

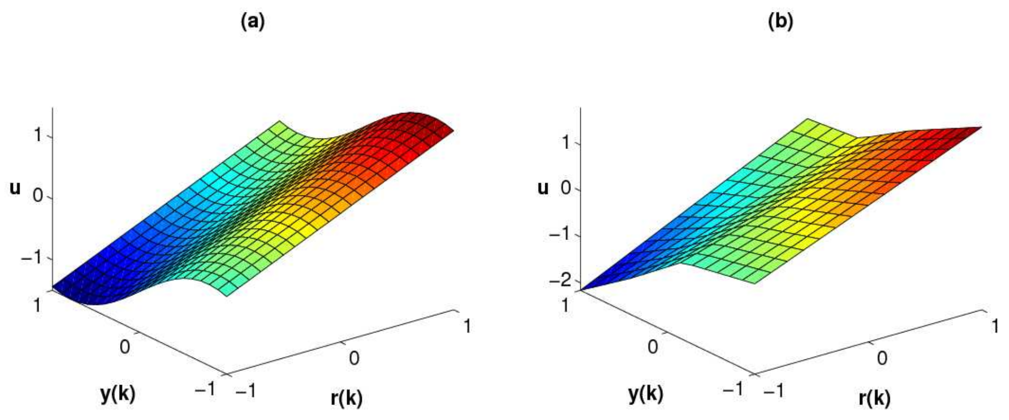


FIGURE 4.3: Example of the selection of the most relevant controller input. (a) Plant's true inverse function. (b) Controller approximation with 8 rules

This same conclusion may be drawn from the analysis of the error of the current approximation to the inverse function by the fuzzy system. To this purpose, let us analyze variable $r(k)$ to determine its degree of responsibility on the approximation error. First, the other variable's range will be divided into infinitesimal intervals with width dy, and each of these intervals will be considered individually. Hence, the situation in each of these small intervals is the same as if we tried to approximate a one-dimensional function. Since a triangular partition is being used, the fuzzy output in one dimension is a linear piecewise function. Therefore, it is obvious that the current configuration of MFs for input

$r(k)$ is enough for an excellent approximation and that adding a new membership function to this input will not improve it. If we perform the same analysis on every interval and add the approximation error existing in each one, we obtain an index that gives us an idea of the need to add a new MF to the input under consideration. For instance, in the example mentioned before (Fig. 4.3), the value of this index is 0.226 for the first input and 16.616 for the second, which confirms the intuitive selection previously made.

After introducing the main idea, let us present the mathematical approach. The first task to be tackled is the computation of the sum of square errors (SSE) for the approximation in each one of the infinitesimal intervals mentioned. For the analysis of variable x_v, these intervals are given by:

$$[x_1, x_1 + dx_1] \times \ldots \times [x_{v_{min}}, x_{v_{max}}] \times \ldots \times [x_N, x_N + dx_N] \tag{4.36}$$

where $[x_{v_{min}}, x_{v_{max}}]$ denotes the full range of the input variable x_v.

As stated before, our goal is to approximate the previously collected points (in the memory M) of the plant's inverse function within this interval. This is the same as approximating a one-dimensional function with a fixed configuration of membership functions for variable x_v. Let us assume that the solution of the approximation problem is the function F^v, which gives the best possible approximation for the plant's inverse function within the considered interval with respect to the variable x_v. The square error for this approximation is given by:

$$e_a^v(\vec{x}) = (u(\vec{x}) - F^v(\vec{x}))^2 \tag{4.37}$$

where $u(\vec{x})$ represents the plant's true inverse function points obtained during normal operation (i.e., the real controller's output) and $F^v(\vec{x})$ is the result of evaluating the obtained function F^v at those points.

Assuming that all the other variables have enough membership functions, if the current configuration for variable x_v were enough to obtain a perfect approximation, then the error $e_a^v(\vec{x})$ given in (4.37) would be zero. This means that variable x_v is not responsible for the approximation error in this interval. On the other hand, a large value of $e_a^v(\vec{x})$ denotes a high responsibility of x_v in the approximation error. Computing the error $e_a^v(\vec{x})$ for each interval of the form (4.36), we obtain a global measure of the SSE that could be reached using the current configuration of membership functions for input x_v. Let us call this measure Index of Responsibility of the input x_v, IR_v:

$$IR_v = \int_{X_1} dx_1 \ldots \int_{X_{v-1}} dx_{v-1} \int_{X_{v+1}} dx_{v+1} \ldots \int_{X_N} dx_N \left[\int_{X_v} dx_v (u(\vec{x}) - F^v(\vec{x}))^2 \right] \quad (4.38)$$

Finally, if we compute this index for every input in the fuzzy system, we are able to compare them and choose the most suitable input, which is the one with the largest value of IR$_v$.

Although the plants considered are continuous in time, the controller exerts its control actions in discrete instants. Therefore, the previous approach has to be reformulated in discrete terms. In this sense, the definition of infinitesimal intervals is replaced by setting a large number of membership functions in every input but the one under analysis, whereas the integrals in expression (4.38) are replaced by sums. This leads us to the following algorithm:

1. For each input x_v:
   (a) Define an auxiliary fuzzy system with the same inputs as the current fuzzy controller:
      i. Assign a large number of membership functions (N_Δ) to each of the input variables, except for x_v, and distribute them homogeneously, forming a triangular partition of the input space. A possible value for N_Δ is given in section 4.3.4.
      ii. The input x_v receives the same MFs in the same location as in the current controller.

   (b) The resulting fuzzy system produces an approximation $F^v_{N_\Delta}$ [Pomares et al., 2000] of the data stored in the memory M.

   (c) Compute the Index of Responsibility IR$_v$ for variable x_v as

   $$IR_v = \sum_{\vec{z}_i \in M} (u(\vec{z}_i) - F^v_{N_\Delta}(\vec{z}_i))^2 \quad (4.39)$$

   where $\vec{z}_i$ is the i-th input vector stored in the memory M, and $u(\vec{z}_i)$ is the output stored for that input.

2. Let $V_{max} = \arg\max_{v}(IR_v)$ be the index associated to the input variable with the largest approximation error in the case of not receiving an additional membership function. This is the selected variable.

This way we have selected the variable which is most responsible of the current approximation error and, hence, the one which needs to receive one more membership function. Now, the next step is finding a proper location for this new MF.

4.3.2.2 Location of the new membership function

Once the input variable has been selected, it is necessary to determine the location of the new membership function. This decision is never easy, as all the possible options have some drawbacks. Intuitively, it might be thought that the best location for the new function is at the point where the error is largest; however, it was already pointed out that this would give the algorithm a high sensitiveness to noise. Another option is to place the new MF at the center of gravity of the mean square error (MSE) distribution through the variable's full range. This is a good option when the data are equally distributed, as it considers the entire error distribution and not only its maximum point. As mentioned before, the use of a grid memory provides a uniform representation of the input space of the plant's inverse function. Therefore, this option is suited for our case.

With this aim, we divide again the range of the input variable in infinitesimal intervals and compute the mean square error in each of them. The center of the new membership function is located at the position given by:

$$c_i^* = \frac{\int_{X_i} x_i \cdot \overline{e^2(x_i)} dx_i}{\int_{X_i} \overline{e^2(x_i)} dx_i} \tag{4.40}$$

where

$$\overline{e^2(x_i)} = \frac{\int \ldots \int e^2(x_1 \ldots x_i \ldots x_n) dx_1 \ldots dx_{i-1} dx_{i+1} \ldots dx_n}{\int \ldots \int dx_1 \ldots dx_{i-1} dx_{i+1} \ldots dx_n} \tag{4.41}$$

Once again, this expression needs to be formulated in discrete terms. To do so, infinitesimal intervals become intervals with a reduced width Δx_i, and equations (4.40) and (4.41) are rewritten as:

$$c_i^* = \frac{\sum\limits_{\Delta x_i} x_i^* \cdot \overline{e^2(\Delta x_i)}}{\sum\limits_{\Delta x_i} \overline{e^2(\Delta x_i)}} = \sum_{\Delta x_i} x_i^* \cdot \overline{e_N^2(\Delta x_i)} \tag{4.42}$$

where x_i^* is the central value of the interval Δx_i and $\overline{e_N^2(\Delta x_i)}$ is the normalized mean square error of all those data points whose component x_i is within the range determined by Δx_i. The expression used to compute the mean square error is:

$$\overline{e^2(\Delta x_i)} = \frac{\sum\limits_{\substack{x_j;j\neq i\\ x_i\in\Delta x_i}} e^2(x_1, x_2, ..., x_i, ..., x_n)}{\sum\limits_{\substack{x_j;j\neq i\\ x_i\in\Delta x_i}} 1} \tag{4.43}$$

where the denominator simply represents the number of data whose component x_i is within the range Δx_i.

4.3.2.3 Initial parameters for the new fuzzy controller

As a consequence of the addition of new membership functions, new fuzzy rules have been created. Although it would be possible to initialize their consequents to any random value and letting them be adapted by the parameter learning algorithm (section 4.3.1), this would cause a sudden decrease in the control quality right after the topology modification. To avoid this situation, it is convenient to initialize the new rules to values that guarantee minimum quality degradation.

Since we are using a complete set of fuzzy rules, every time a new membership function is added to an input x_v, $\prod\limits_{\substack{i=1\\ i\neq v}}^{N} n_i$ new rules have to be created. These new rules have the form

IF x_1 is $X_1^{i_1}$ AND...AND x_v is $X_v^{j_v}$ AND...x_N is $X_N^{i_N}$ THEN $u = R_{i_1 i_2 \ldots i_v = j_v \ldots i_N}$

where j_v is the position of the new membership function within the list of ordered centers (see Fig. 4.4).

To explain the changes applied to the parameters of the fuzzy system, we will name the set of old parameters (before the new MF was added) as $\tilde{\Theta}$ and the new set of parameters as Θ. Firstly, the changes applied to the centers of the membership functions are as follows (see Fig. 4.4(a)). Note that the MFs belonging to the rest of variables are not affected by any changes:

$$\theta_v^j = \tilde{\theta}_v^j \qquad if\; j < j_v \tag{4.44a}$$

$$\theta_v^j = \tilde{\theta}_v^{j-1} \qquad if\; j > j_v \tag{4.44b}$$

$$\theta_i^j = \tilde{\theta}_i^j \qquad if\; i \neq v \tag{4.44c}$$

Secondly, to prevent a severe damage on the controller's performance, the rule consequents have to be readjusted so that the outline of the global function represented by the fuzzy system is kept the same as before, i.e., $\hat{G}(\vec{x};\Theta) = \hat{G}(\vec{x};\tilde{\Theta})\;\forall\vec{x}$. The new rules will be activated only when the input value for variable x_v is within the range $[\theta_v^{j_v-1}, \theta_v^{j_v+1}]$, which is the partition created by the new membership function. Therefore, in the rest of cases, the same old consequents are used (see Fig. 4.4(b)):

$$R_{i_1,\dots,i_v,\dots,i_N} = \tilde{R}_{i_1,\dots,i_v,\dots,i_N} \qquad if\; i_v < j_v \tag{4.45a}$$

$$R_{i_1,\dots,i_v,\dots,i_N} = \tilde{R}_{i_1,\dots,i_{v-1},\dots,i_N} \qquad if\; i_v > j_v \tag{4.45b}$$

To obtain the initial consequents of the new rules, we impose the following condition: at the point of maximum activation of the rule, the consequent equals the output produced by the system under its previous configuration for the same input. As the maximum activation degree is reached when all the inputs are located at the centers of the antecedent MFs, we have that:

$$R_{i_1,\dots,i_v=j_v,\dots,i_N} = \hat{G}(\vec{c};\tilde{\Theta}) \tag{4.46}$$

where $\vec{c} = (\theta_1^{i_1},\dots,\theta_v^{i_v=j_v},\dots,\theta_N^{i_N})$, with $i_1 = 1,\dots,n_1$, ..., $i_N = 1,\dots,n_N$.

Finally, the only change to be made to the number of membership functions is adding one to the variable to which the new function has been added, while

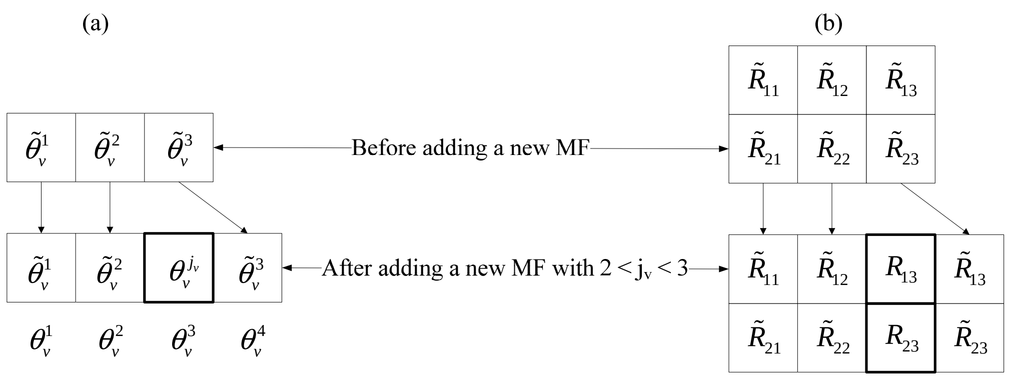

(a)
(b)
$\tilde{\theta}_v^1$ $\tilde{\theta}_v^2$ $\tilde{\theta}_v^3$
Before adding a new MF
$\tilde{R}_{11}$ $\tilde{R}_{12}$ $\tilde{R}_{13}$
$\tilde{R}_{21}$ $\tilde{R}_{22}$ $\tilde{R}_{23}$
$\tilde{\theta}_v^1$ $\tilde{\theta}_v^2$ $\theta_v^{j_v}$ $\tilde{\theta}_v^3$
After adding a new MF with 2 < j_v < 3
$\tilde{R}_{11}$ $\tilde{R}_{12}$ R_{13} $\tilde{R}_{13}$
$\tilde{R}_{21}$ $\tilde{R}_{22}$ R_{23} $\tilde{R}_{23}$
θ_v^1 θ_v^2 θ_v^3 θ_v^4

FIGURE 4.4: Topology change: modifications made in the fuzzy controller. (a) Addition of a new membership function. (b) New rules created after adding the new membership function

the rest remain unchanged:

$$n_v = \tilde{n}_v + 1 \tag{4.47a}$$

$$n_i = \tilde{n}_i \quad \forall i \neq v \tag{4.47b}$$

4.3.2.4 Input variable selection

Let us suppose that only one membership function is assigned to the input x_v, such that its activation is equal to 1 for any input value. This means that the value of x_v actually has no effect on the activation degree of the rules, as 1 is the identity element for the product operator. Therefore, we can state that assigning only one membership function to an input variable is the same as not considering that variable in the fuzzy inference process.

One of the advantages of the proposed method is that it is also valid in the case just mentioned, which makes possible its use as a mechanism for the selection of input variables when it is hard to identify *a priori* the relevant variables. Hence, if the algorithm decides to assign one more membership function to an input variable, it will mean that such a variable is relevant and has to be taken into account. Furthermore, with this approach, there exists the chance of having variables that are irrelevant at the beginning but which become important when the accuracy increases.

This feature makes it possible to begin with very simple structures (even empty ones, i.e., only one membership for each input variable), as the algorithm will determine by itself which the best topology is. Thus, we avoid considering more complex configurations and the need of applying techniques for rule pruning, as it occurs in other methods [Phan and Gale, 2008], [Gao and Er, 2003].

4.3.2.5 Comments on the stability of the topology self-evolving process

In section 4.3.1.4, the stability for each given topology was commented. Now, to establish the stability of OSEFC, the stability across the structural changes has to be considered, i.e., when a new membership function is added to the controller.

Given a specific topology, the control law can be parametrized as $u = \vec{R}^{*}\mu(k)$, where $\vec{R}^{*}$ denotes the optimal parameters (consequents) of the controller for this topology. In this case, the expression (4.5) given for the consequents' adaptation is a gradient update law, already proven to be stable in the sense that the update law will make the vector of consequents $\vec{R}$ converge to $\vec{R}^{*}$, and the tracking error will converge to a ball around the origin [Nounou and Passino, 2004; Spooner et al., 1997].

Concerning the structural changes, we use the assumption made in (4.3) (see section 4.2) about the existence of a control law that achieves the perfect tracking. This control law is approximated by the self-organizing controller OSEFC. In this case, if at a given moment, the controller's topology is formed by n_1 membership functions, we have that

$$u = \vec{R}^{*}(n_1)\mu(n_1) + \epsilon(n_1) \tag{4.48}$$

where $\vec{R}^{*}(n_1)$ are the optimal consequents for this topology and $\epsilon(n_1)$ is the approximation error. As already commented, this controller will converge to a ball around the origin with radius $b(1)$.

Consider now that a new membership function θ_v^j is added. The universal approximation property of fuzzy controllers implies that the approximation of the resulting controller will be better at least in the area corresponding to the location of the new membership function (i.e., in the range $[\theta_v^{j-1}, \theta_v^{j+1}]$), even if the consequents are not adapted. Once the consequents of the new controller are adapted, the best control policy that can be achieved with this structure is

$$u = \vec{R}^{*}(n_2)\mu(n_2) + \epsilon(n_2) \tag{4.49}$$

where $\vec{R}^{*}(n_2)$ are again the optimal consequents for this structure and $\epsilon(n_2)$ is the approximation error, which is smaller than $\epsilon(n_1)$. This means that the radius of the ball where the tracking error will converge will be $b(2) < b(1)$. Hence, since the radii decrease, the tracking error will eventually converge to zero, as the operation time goes to infinity.

Equivalently, if we consider the Lyapunov function $V(k) = b(k)$, for which $\Delta V = V(k+1) - V(k) < 0$, then the asymptotic stability is proven. Therefore, although the mathematical formulation is not possible, this discussion shows that the proposed methodology provides a reasonably stable controller.

4.3.3 Alternative methodology for parameter learning

The parameter adaptation method presented in section 4.3.1 follows a local learning approach. It aims to reduce the tracking error at the present time instant, thereby taking care of the short-term response of the controller. This type of adaptation is usually sufficient to satisfy the accuracy demands of the control system. Nonetheless, there may be more demanding applications that require a finer tuning of the controller's parameters. In this section, we present a learning approach based on global information [Cara et al., 2010c] that works in combination with the local learning described in section 4.3.1. In order to differentiate both types of learning, in the reminder of this section we will refer to the local learning method as LL-module (Local Learning module) and to the global learning as GL-module (Global Learning module).

The combined learning procedure is depicted in Fig. 4.5 and can be summarized as follows: First, only the LL-module works during a period T'. During this time, the control policy may be rough, but the local learning allows for the gathering of truly useful data from the plant. After this period, and while the plant is locally controlled, the GL-module starts tuning the consequents to achieve a globally finer control policy. It is important to note that the changes proposed

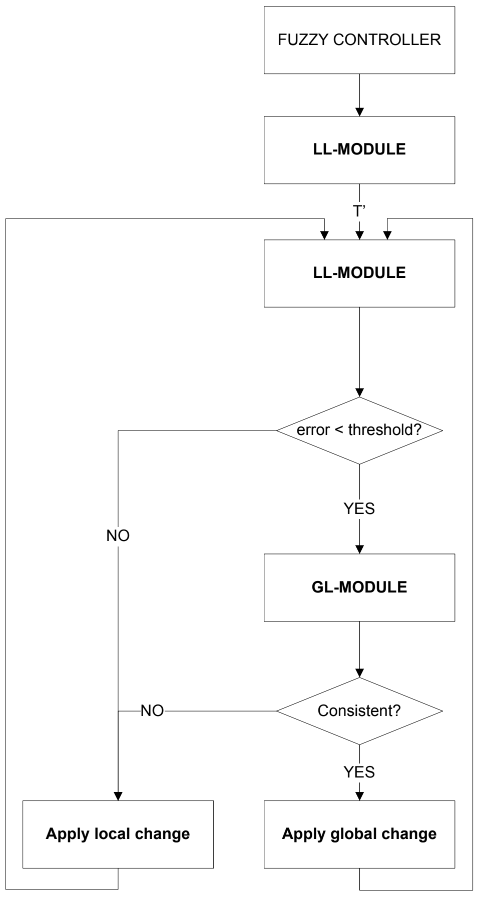


FIGURE 4.5: Flowchart of the online parameter learning method combining local (LL-module) and global (GL-module) learning

by the GL-module must always be consistent with those suggested by the LL-module. The reason for this is that the main objective at every moment is to keep the plant under control, i.e., the short-term response cannot be degraded by the global changes. Hence, the LL-module acts as a supervisor of the GL-module.

The GL-module aims to assure the global performance of the control policy, as a means of providing a better response to the changes in the reference signal. In order to improve the global performance, the adaptation process cannot simply concentrate on the rules that are being fired at the present moment, but rather needs to consider the remaining rules as well. To do so, historical data about the behavior of the plant is needed.

Since our hypothesis here is that the dynamics of the plant are unknown, we cannot compute the partial derivative $\partial y/\partial u$. Therefore, it is not possible to apply any gradient-based techniques to minimize the error at the plant's output. However, such methodologies can be applied if the error at the controller's output is used instead [Pomares et al., 2004]. As mentioned before, the fuzzy controller is an approximator of the plant's true inverse function; therefore, the error at the controller's output is equivalent to the error of this approximation. Thus, minimizing the error made by the controller indirectly minimizes the tracking error made at the plant's output. Thanks to this idea, the need for the model of the plant can be avoided.

For the process of global learning, the memory M described at the beginning of section 4.3.2 is used again, as it stores the I/O data that represent the true inverse function of the plant. Hence, it is possible to compute the error at the controller's output at time k for the m-th datum as

$$e_u(m) = u_m - \tilde{u}_m \tag{4.50}$$

where u_m is the control signal stored in the memory M and $\tilde{u}_m$ is the output produced by the current controller for the input vector $\vec{z}_m$.

At each control step k, the objective of the GL-module is to minimize the mean square error for all the data in M:

$$J(k) = \sum_{m=1}^{K} e_u^2(m) = \sum_{m=1}^{K} (u_m - \tilde{u}_m)^2 = \sum_{m=1}^{K} \left(u_m - \hat{G}(\vec{z}_m;\Phi_k)\right)^2 \tag{4.51}$$

where K is the number of elements stored in the memory M.

The new consequents proposed for the fuzzy controller are those that minimize (4.51), that is,

$$\Gamma^g(k) = \arg\min_R(J(k)) = \arg\min_R\left(\sum_{m=1}^{K}\left(u_m - \hat{G}(\vec{z}_m;\Phi_k)\right)^2\right) \tag{4.52}$$

where $\Gamma^g(k) = \left\{R_1^g(k), R_2^g(k), ..., R_{N_r}^g(k)\right\}$ is the set of new values for all the consequents.

Before replacing the old consequents in the fuzzy controller by the new ones, it must be kept in mind that we have minimized the error at the controller's output and not the error at the plant's output. Thus, it is necessary to verify if the new consequents proposed by the GL-module actually improve the tracking of the reference signal. In this sense, we use the LL-module as a supervisor. As stated before, the monotonicity of the plant w.r.t. the control signal allows us to determine the right direction in which the consequents have to be moved. Therefore, if the modification proposed for a consequent by the GL-module goes in the same direction as that indicated by the LL-module, the change is accepted. Otherwise, the local modification is applied.

Finally, it is important to note that the global learning cannot be used without the existence of the local module as the latter is the one which is initially capable of starting to control the plant and obtaining truly useful I/O data.

4.3.4 Additional comments

In this section, the description of the proposed method is completed by pointing out some additional issues:

- A criterion to switch from the parameter learning phase to the topology self-evolving stage is required. As stated before, we consider that a structural change is needed when the consequent adaptation is unable to continue improving the quality of the control. The mean square error of the reference tracking for n_s samples, computed according to (4.53), is used as the quality measure:

  $$\mathrm{MSE} = \frac{1}{n_s}\sum_{k=1}^{n_s}(r(k) - y(k+1))^2 \tag{4.53}$$

  The value n_s used to compute this MSE can be estimated according to the variability of the reference signal, which is usually known beforehand. In general, it is desirable that n_s is big enough as to provide a general view of the evolution of the control in different operating regions. Nonetheless, it should not be too large, as it would make the learning process slower. Although there is not a general guideline for selecting this value, the following suggestions can be applied as a rule of thumb:

  - When stepwise reference signals are used, n_s should be chosen large enough as to cover several steps.
  - If the reference signal changes randomly and rapidly, smaller values of n_s can be used, as it can be expected that more operating regions will be covered in less time.
  - When the references follow a pattern, the size of this pattern can be selected as n_s, as it covers the entire operation of the control.

- It is important to note that a decreasing of the improvement rate of the MSE does not always mean that the current topology is insufficient to provide a proper control. When the plant is operating in steady state (i.e., the reference has already been reached), the MSE stops improving but there is no need of modifying the topology, since the plant is already under control. To avoid changes in this situation, a threshold for the minimum control error is defined. Whenever the MSE is below that value, there is no need to add more membership functions.

- When the improvement of the MSE is below a given threshold but its value is larger than the desired minimum, the data in the memory M is frozen and the topology self-evolving process is run in parallel with the normal operation of the control system, which is not interrupted at any point.

- Obtaining a very accurate control policy may require a large amount of membership functions. In some practical applications, this may not be desirable, due to the limitations they may present, e.g., hardware limitations. To avoid this situation, an upper bound for the number of generated membership functions can be established.

- It is also important to highlight that the number of input dimensions chosen for the fuzzy controller has a direct impact on the data gathering process. Firstly, since the memory M uses a grid structure, the number of data points needed to represent the plant's inverse function grows exponentially with the number of input dimensions. As these data are gathered online, this implies that it will take a longer time to collect enough information to properly represent the inverse function of the plant. Secondly, increasing the number of input dimensions also affects the interpretability of the rules; i.e., since all the input variables are used to define the antecedents, the rules obtained will be complex and more difficult to interpret for the user.

- The quality of the data stored in the memory M affects the approximation of the inverse function of the plant obtained. Thus, enough representative data points are needed. To address this issue, we can use a threshold on the minimum number of data in M required to modify the controller's topology, e.g., do not modify the structure until there is data in a certain percentage of the hypercubes. Although there is no guideline to select this threshold, its selection is not very hard, as the user of a control system usually knows how variable the reference signal is. Therefore, if there is not much variation, a smaller value for the threshold will be preferred, whereas larger values will be chosen in the case of more changing reference signals.

- In the algorithm for the selection of the input variable that must receive a new MF (see section 4.3.2.1), we defined an auxiliary fuzzy system with N_Δ membership functions in all the variables but the one under analysis. The value of N_Δ depends on the amount of I/O data available. On the one hand, if we set too many membership functions, there will be no data in some of the intervals defined by them and the estimation will not be reliable. On the other hand, if there are too few MFs, they will be too wide and will be far from the ideal continuous case. For instance, if we have a controller with two input variables and 400 I/O data points stored, it will not be reasonable to assign more than 10 membership functions per variable. Note that, in that case, there would be an average of four data samples per interval, which is enough to be considered equivalent to "a large number of membership functions". Thus, since the grid memory M provides a uniform representation of the input space of the plant's inverse function, a possible estimation of the number of MFs is:

$$N_\Delta = \frac{\sqrt[N]{K}}{2} \tag{4.54}$$

where K is the number of I/O data stored in the memory M and N is the number of input variables.

Note that this fuzzy system is only used for the selection of the input variable that is receiving a new function. Its aim is producing an approximation of the inverse function of the plant that is "perfect" in all the dimensions but one. The modification of the topology is performed in parallel with the normal operation of the controller. Therefore, the fuzzy system with N_Δ membership functions neither operates over the plant nor performs any type of control. Only when the new structure is completely defined (i.e., the new MF is located and the new rules are initialized), the fuzzy controller is replaced by the new one.

- In the parameter learning method with global information (section 4.3.3), we defined a period of time T' for the gathering of useful I/O data to be used by the GL-module. Since the goal during this time is to obtain enough representative data about the plant's inverse function, its value should depend on the variability of the reference values. Again, it is not possible to provide a general value for this parameter that works well in every possible case. Nonetheless, the guidelines provided for the estimation of n_s can be also applied in this case.

4.4 Experimentation and results

With the aim of illustrating the main features of the proposed OSEFC, a set of experimental results is gathered in this section. These tests can be classified into three groups:

- *Description of the basic features of OSEFC*: In this group, we analyze the basic properties of the proposed methodology by applying OSEFC to a benchmark of synthetic plants [Cara et al., 2011a]. This type of plants is useful to provide an insight of the operation of the methodology without the "interferences" caused by the complexity of real-world systems. The plants used for this type of experiment are described by their difference equations.

- *Simulation of real-world systems*: In this category, we include software simulations of systems from the real world [Cara et al., 2010b, 2011a]. These experiments are obviously more complex than the ones in the first set and aim to provide an idea of the operation of the controller with real plants. To simulate these plants, the fourth-order Runge-Kutta method has been used. It is important to highlight the difference between the sampling time T_p used for the simulation and the sampling time used to exert control actions T_c (see section 4.2).

- *Experiments with real systems*: Finally, some experiments with a real system have been carried out. Obviously, these are the most challenging tests for the control methodology, as the environment cannot be controlled and there may exist multiple external factors influencing the behavior of the plant [Cara et al., 2010a,c].

It has to be highlighted once more that OSEFC does not use any information about the internal dynamics of the plants, other than the sign of their monotonicity with respect to the control signal. For this reason, the plants used for the experimentation are described only from a qualitative point of view. Nonetheless, the reader can find the information regarding the differential equations and parameters of the plants in Appendix A.

4.4.1 Description of the basic features of OSEFC

With the goal of clarifying the operation of the proposed methodology, in this section we present a series of simulation results for synthetic plants. For the sake of clarity, each plant is described by its difference equation, although this equation is not used for the control process.

4.4.1.1 Parameter learning for a given topology

Firstly, the features of the consequent adaptation phase (see section 4.3.1) are discussed. To do so, only this phase has been applied to a fuzzy controller with a predefined fixed topology. First, the performance of the local learning procedure discussed in section 4.3.1 is analyzed. Then, the addition of the global learning module (section 4.3.3) is considered. Finally, the response of OSEFC to an unexpected change in the dynamics of the plant is also shown.

Consider the plant defined by the following equation [Pomares et al., 2002b]:

$$y(k+1) = -0.075\sin(y(k)) + \frac{u(k) + u^3(k)}{4} \tag{4.55}$$

Despite of its apparent simplicity, this plant shows a nonlinear behavior with respect to both the control signal and the variable to be controlled. Moreover, it is to be noted that a control signal $u(k) = 0$ does not guarantee a stationary plant output.

To control this plant, a fuzzy controller with two input variables (namely the reference signal $r(k)$ and the plant output $y(k)$) is used. The reference is a step-wise function that changes randomly in the range [-1,1]. The actuator's range of operation is [-1.5 1.5]. Since the monotonicity of the plant with respect to u is positive, and according to (4.23), the constant C is set to 1.5. Finally, the sampling frequency is set to 100 samples per second.

For the test with local learning only, we have selected a topology with a medium number of rules. Specifically, an evenly distributed triangular partition with four MFs for the input $r(k)$ and two for $y(k)$ is used; therefore, eight rules form the fuzzy controller. We have selected this topology because it is one of the configurations obtained by the self-evolving methodology, as we show later in Table 4.1. Initially, the value of all the consequents is set to zero. Thus, the situation is equivalent to having an empty controller.

Fig. 4.6 depicts the evolution of the eight rule consequents. During the first iterations there are sharp changes in their values, although after a few seconds, a clear convergence is shown. Fig. 4.7 shows the reference tracking obtained during the initial 30 seconds. It is possible to note the effect of the lack of knowledge about the plant, which causes a poor performance at the beginning; however, the parameter learning process soon produces a significant enhancement. Some overshoot can be observed at the end of the simulation. This is due to the limitations of the topology selected for the controller.

In order to depict the quality improvement that may derive from the use of global information in the learning process, we have developed two simulations

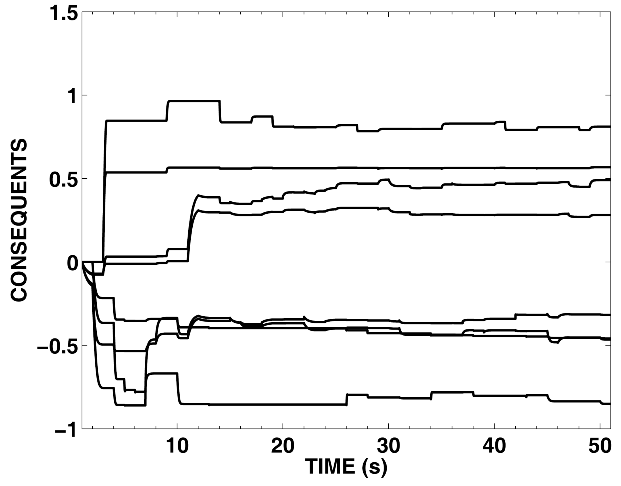


FIGURE 4.6: Online local learning of the consequents during the first 50 seconds controlling plant (4.55) with a 4x2 topology

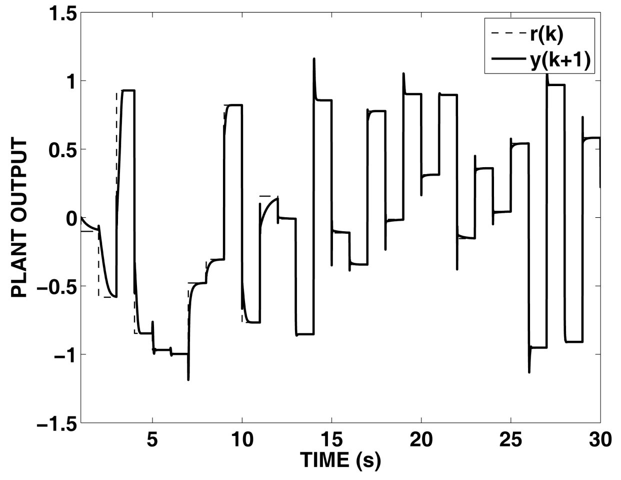


FIGURE 4.7: Reference tracking during the first 30 seconds of controlling plant (4.55) with local consequent learning only

in the same conditions. The first one only applies the local learning, whilst the second one applies both the local and the global learning policies, as described in section 4.3.3. For this example, five evenly distributed MFs are assigned to each input. Therefore, 25 fuzzy rules initialized to zero are used. This topology has been selected to provide enough MFs to both inputs, assuming that the plant is unknown.

In this case, the reference signal follows a pattern of one-second-long ran-

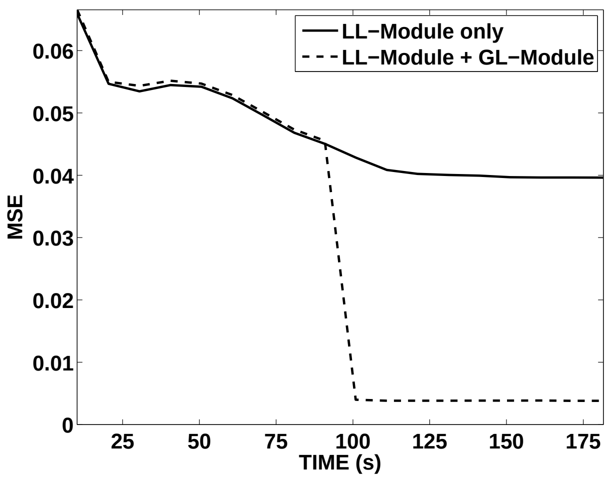


FIGURE 4.8: Local learning performance versus local and global learning performance for the control of plant (4.55)

dom steps that is repeated every 10 seconds, i.e., the pattern is formed by 10 steps. The period of initial data gathering for the GL-module has been set to $T' = 100$ seconds, i.e., 10 presentations of the reference pattern. This value has been selected to provide enough time for the LL-module to obtain a satisfactory tuning of the parameters.

Fig. 4.8 compares the evolution of the MSE for the cases of applying only local learning (solid line) and both local and global learning (dotted line). The MSE is computed for every presentation of the reference pattern, thus, the number of samples used is $n_s = 1000$. It can be observed that, at the beginning, both learning methods show the same evolution. Obviously, the reason for this is that only local learning is being applied in both cases, since the method with global learning is operating in the initial data gathering period. However, after 100 seconds, the effect of the global learning is evident: although the minimum error achieved by the local learning at the end of the simulation is 0.039, the addition of the GL-module allows the controller to reach a value of 0.0038, i.e., the performance is ten times better thanks to the global learning. This experiment has been repeated with different reference patterns, obtaining similar results every time.

Another interesting fact is the response of OSEFC to unexpected changes in the dynamics of the plant. To show this behavior, we have carried out the same

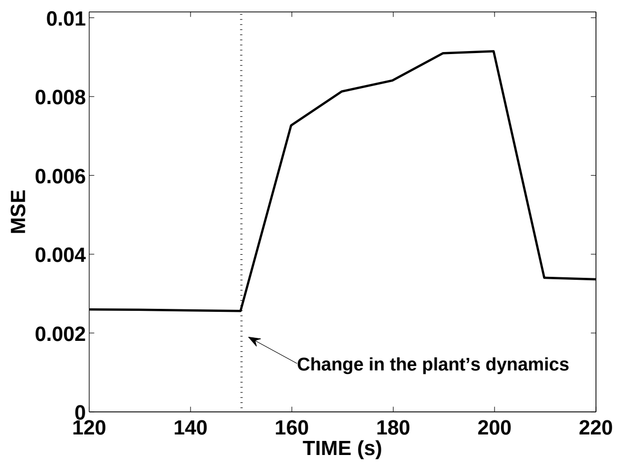


FIGURE 4.9: Response of OSEFC's parameter learning procedure to an unexpected change in the dynamics of the plant (4.55), that decreases the response of the actuator by 20%

simulation as described above. However, after 150 seconds the definition of the plant's equation is changed as follows:

$$y(k+1) = -0.075\sin(y_k) + 0.2u(k) + 0.25u^3(k) \tag{4.56}$$

The only difference with the previous definition of the plant is the multiplying factor applied to $u(k)$, which is reduced by 20%. This situation represents a decrease in the actuator's response, which could be due, for instance, to an incipient fault. Fig. 4.9 depicts the evolution of the MSE before and after the change in the dynamics of the plant. It is observed that the change in the plant causes a sudden increase in the mean square error. However, after some time, and due to the joint action of the LL-module and the GL-module, the error decreases again.

4.4.1.2 Global operation of OSEFC

As stated before, the adaptation of the rule consequents is not enough to obtain a high quality control, unless an adequate topology for the fuzzy controller is known beforehand. In this section, the full adaptive self-evolving method is applied again to plant (4.55), to show its capability to obtain a suitable topology that provides a good approximation to the plant's inverse function.

TABLE 4.1: Topology evolution for the control of plant (4.55)

Configuration	IR_1		IR_2		MSE	
	Example	Mean $\pm\sigma$	Example	Mean $\pm\sigma$	Example	Mean $\pm\sigma$
1x1	26.42	25.09 ± 1.20	1.94	1.83 ± 0.10	2.0769	1.666 ± 0.363
2x1	17.99	22.85 ± 4.21	4.87	5.37 ± 0.45	0.5347	0.418 ± 0.101
3x1	21.19	25.29 ± 3.55	5.43	5.70 ± 0.24	0.5051	0.418 ± 0.080
4x1	1.91	1.98 ± 0.06	5.70	5.87 ± 0.16	0.0533	0.048 ± 0.025
4x2	1.97	1.98 ± 0.03	5.35	5.38 ± 0.03	0.0289	0.032 ± 0.020
4x3	2.00	2.03 ± 0.03	5.45	5.41 ± 0.03	0.0110	0.018 ± 0.015
4x4	1.98	2.02 ± 0.05	0.21	0.35 ± 0.14	0.0078	0.008 ± 0.005
5x4	5.11	5.14 ± 0.04	0.22	0.36 ± 0.14	0.0073	0.006 ± 0.002
6x4	0.60	0.59 ± 0.01	0.23	0.33 ± 0.13	0.0031	0.004 ± 0.003
7x4	1.14	1.14 ± 0.01	0.22	0.36 ± 0.14	0.0021	0.003 ± 0.002

The parameters defining the system (e.g., actuator's range, constant C, etc.) are the same as defined at the beginning of section 4.4.1.1. The input variables are also the same (namely the reference signal and the plant output), although, in this case, the controller starts with an empty configuration, i.e., only one membership function for each variable and the only rule initialized to zero. The memory M is defined as a 30x30 grid which stores I/O data samples from the true inverse function, obtained online while the control system is operating.

A randomly changing stepwise reference signal varying in the range [-1,1] is used. This reference signal does not follow a pattern, which means that the sequence of steps is not repeated. This is important, since the effect of any specific reference value on the evolution of the controller is very limited, specially as the time goes to infinity. Nonetheless, the simulation has been repeated five times, always obtaining the same topologies for the controller.

Table 4.1 depicts the controller's evolution through the entire process. For each topology change, it shows the value of the IR for both input variables and the MSE obtained after adapting the consequents of the new rules, i.e., right before the next topology change. The MSE is computed every 10 seconds (i.e., n_s = 1000 samples), which corresponds to 40 steps in the reference signal. This time allows the plant to reach multiple operating regions, thereby providing

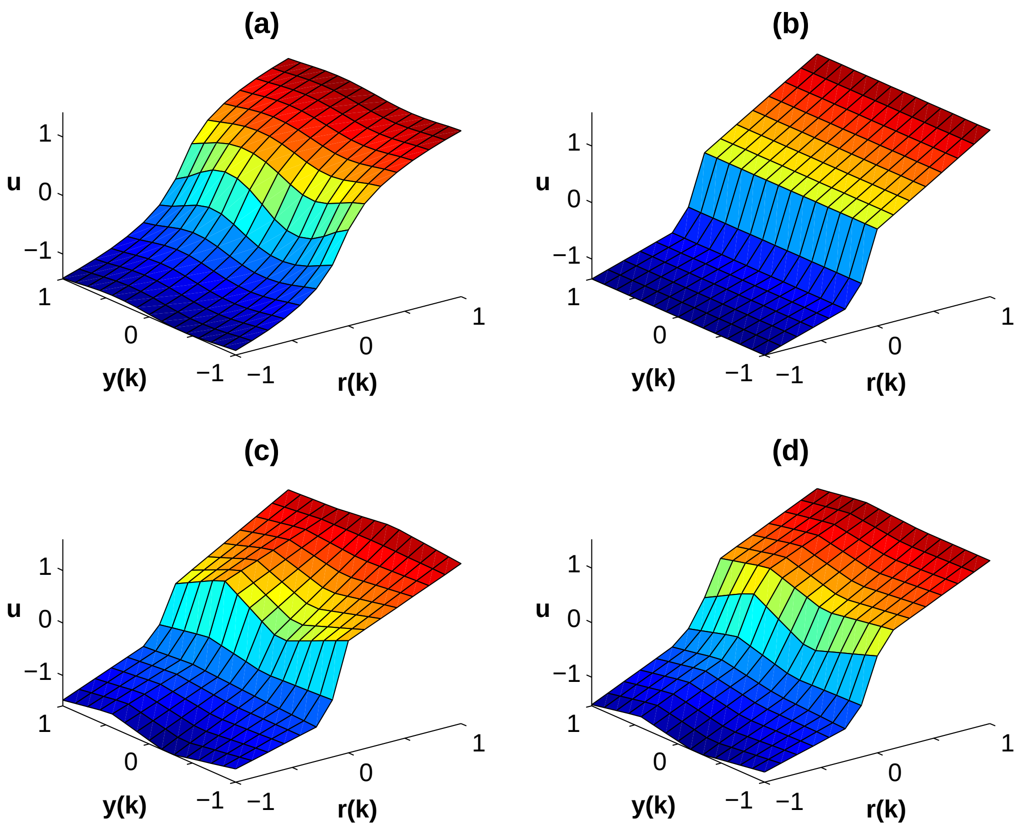


FIGURE 4.10: Evolution of the approximation to the plant's true inverse function with different topologies. (a) Plant's true inverse function. (b) Controller function with 4x1 membership functions. (c) Controller function with 4x4 MFs. (d) Controller function with 6x4 MFs

representative information about its response for each computation of the MSE. For each measure, the table presents the specific value obtained in the example shown and the mean and standard deviation of the five executions.

At first, the input $r(k)$ is clearly more important than $y(k)$, which can be easily confirmed by looking at the true inverse function depicted in Fig. 4.10(a). Therefore, the algorithm always assigns new MFs to this input. However, when the 4x1 configuration is reached, a finer control policy is needed and the second variable comes into play. Fig. 4.10(b)-(d) show the progressive improvement in the approximation to the plant's inverse function made by the fuzzy controller. Comparing configuration 4x1 (Fig. 4.10(b)) to configuration 6x4 (Fig. 4.10(d)), it can be noted that the MFs added to the input $y(k)$ provide a good approximation

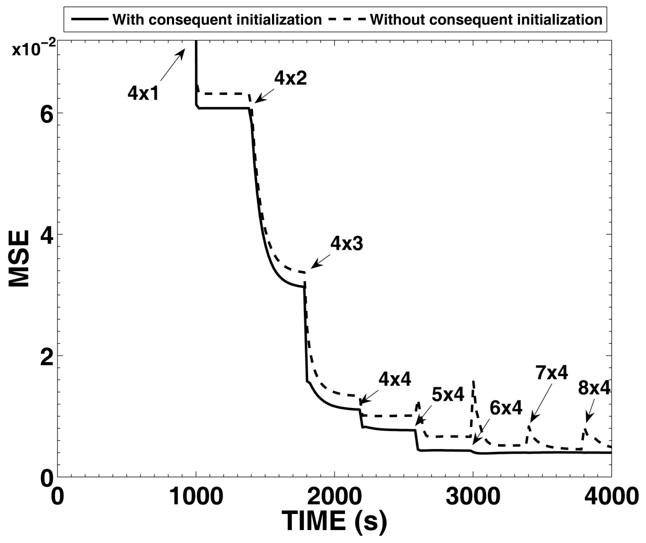


FIGURE 4.11: Evolution of the MSE with and without initialization of the new rules

to the sine wave in this component of the inverse function. In cases like this, when the topology provides a fine approximation of the plant's inverse function and the consequents start converging, the adaptation methodology can be stopped and the application of the resulting fixed controller is possible. In this case, the controller achieves the so-called convergence of the topology, which is related to the stability-plasticity dilemma mentioned in section 3.2.2.1.

Fig. 4.11 shows the evolution of the mean square error during the simulation. The solid line corresponds to the MSE when the consequents of the new rules are initialized according to (4.46). The first thing to observe is that, every time a topology change is performed, the MSE has stopped decreasing, which means that the adaptation of the consequents cannot longer improve the control. In every case, the addition of a new MF helps reducing the MSE. In contrast, the dotted line depicts the evolution of the error when the consequents of the new rules are initialized to random values. It can be observed that the initialization proposed in this work has two main effects: First, the MSE decreases faster. Second, the lack of initialization produces sudden increases in the error right after the changes in the topology, because the new rules do not use the knowledge already acquired by the controller. This undesirable effect is avoided by the proposed initialization.

It is important to note that, in the case of random initialization, the rules would eventually achieve proper consequents; therefore, the error would reach the same values as when using initialization. In order to make the comparison of both cases possible, the topology changes have been forced to happen at fixed times. Otherwise, the simulation without initialization would have taken a longer time but, in the end, the results would have been the same as when using initialization.

4.4.1.3 Automatic input variable selection

In section 4.3.2.4, we commented OSEFC's capability of selecting the most relevant input variables from a set of candidates. Here, we present an example of this interesting feature. Consider the following plant [Chen and Khalil, 1995]:

$$y(k+1) = \frac{1.5y(k-1)y(k)}{1+y^2(k-1)+y^2(k)} + 0.35\sin(y(k-1)+y(k)) + 1.2u(k) \tag{4.57}$$

Initially, four input variables (namely $r(k)$, $y(k)$, $y(k-1)$, and $y(k-2)$) are considered. These variables are actually "candidates" to be a part of the control process, as all of them initially have only one membership function assigned, and therefore, are not participating in the fuzzy inference process. Within this context, "selecting" variables is equivalent to adding membership functions to them. At the end, those variables having a unique membership function will be ignored in the inference process, i.e., they are considered to be irrelevant for the control.

The values of the control signal are limited to the range $[-1, 1]$ and the monotonicity of the plant w.r.t. the control signal is positive, as shown below in equation (4.58). Randomly varying stepwise reference signals in the range $[-1, 1]$ are used for this simulation. Hence, according to (4.23), the value of the constant C is set to 1. In order to reduce the influence of the specific reference values on the results, the reference steps do not form a repetitive pattern. Nonetheless, the experiment has been repeated five times, obtaining analogous results in all of them.

Table 4.2 depicts the evolution of the controller's topology. For every selected configuration, it shows the IR associated to each input variable and the MSE

TABLE 4.2: Topology evolution for the control of plant (4.57)

Configuration	IR_1	IR_2	IR_3	IR_4	$MSE \cdot 10^3$
1x1x1x1	0.59	0.03	0.02	0.0002	454.61
2x1x1x1	0.02	205.80	210.46	0.02	131.49
2x1x2x1	0.03	294.75	6.09	0.03	84.56
2x2x2x1	0.04	10.51	10.54	0.041	3.75
2x2x3x1	0.04	10.54	6.68	2.08	2.36
2x3x3x1	0.04	6.76	6.79	0.04	1.05
2x3x4x1	0.04	6.94	0.62	0.04	0.72
2x4x4x1	0.04	0.61	0.59	0.04	0.52

achieved by the chosen topology at the moment of the next change. The MSE is computed using $n_s = 2000$ samples, which corresponds to the presentation of 20 reference steps.

The first decision made by OSEFC is to consider the reference signal as a relevant variable for the control. Although $r(k)$ does not appear in the definition of the plant, it is easy to see its importance in the control process. As already stated, the evolving fuzzy controller's aims to approximate the inverse function of the plant. The following expression is obtained when the difference equation (4.57) is solved for the control signal $u(k)$:

$$\begin{aligned} u(k) &= 0.83y(k+1) - \frac{1.25y(k-1)y(k)}{1+y^2(k-1)+y^2(k)} - 0.29\sin(y(k-1)+y(k)) \\ &\equiv 0.83r(k) - \frac{1.25y(k-1)y(k)}{1+y^2(k-1)+y^2(k)} - 0.29\sin(y(k-1)+y(k)) \end{aligned} \tag{4.58}$$

where the reference $r(k)$ has been used to replace $y(k+1)$.

This equation confirms that the control signal depends linearly on the reference signal, which is adequately acknowledged by OSEFC by assigning two MFs to this input. On the contrary, no membership functions are ever added to the fourth variable. Equation (4.58) confirms that $u(k)$ does not depend on $y(k-2)$; hence, the decision of not taking it into account for the control is correct. Finally, the same number of MFs is assigned to $y(k)$ and $y(k+1)$, which is also consistent with the fact that $u(k)$ depends in the same way on both of them. Further, it can be observed that these two inputs present similar values of the IR when they have the same number of MFs assigned, e.g., observe the configurations 2x2x2x1 and 2x3x3x1. This means that they present a similar degree of responsibility on the approximation error, and therefore, any of them could receive a new MF. On the contrary, when one of them presents one more MF, the index of responsibility clearly states the need of assigning a new MF to the other one, thereby indicating that both inputs are equally important for the control process.

4.4.1.4 Tolerance to noise

The learning capabilities of OSEFC provide the control system with a higher tolerance to noise. In this section, this feature is studied through the use of a plant that presents noisy readings for the plant output $y(k)$. Specifically, equally distributed additive noise is added to $y(k)$ to three different levels, namely 5%, 10% and 20% of the plant's operating range.

The plant used for this example is defined in (4.35) and repeated here for clarity:

$$y(k+1) = 0.8\sin(2y(k)) + 1.2u(k)$$

The simulation is started with a two-input controller (i.e., $r(k)$ and $y(k)$) and two membership functions for each input. Initially, all the rule consequents are set to zero; therefore, the controller is empty. Random steps in the range $[-1, 1]$ are used as the reference signal. The actuator's range of operation is $[-1, 1]$. It is easy to observe that the monotonicity of this plant w.r.t. the control signal is positive. Hence, C is set to 1. Again, the experiment has been repeated five times, obtaining similar results in all of them.

Table 4.3 compares the simulation results obtained for the noise-free case with the results of the three aforementioned noisy cases. The table shows the different topology configurations obtained by OSEFC, the MSE achieved after optimizing the consequents, and the standard deviation of the measurement of the MSE in all the executions. The MSE is computed every $n_s = 2000$ samples, i.e., 20 reference steps.

TABLE 4.3: Topology evolution in a noisy environment

No noise		Noise 5%			Noise 10%			Noise 20%		
Cfg.	MSE·10^3	Cfg.	MSE·10^3	σ	Cfg.	MSE·10^3	σ	Cfg.	MSE·10^3	σ
2x2	7.91	2x2	7.70	1.89	2x2	10.29	2.52	2x2	25.54	2.81
2x3	1.26	2x3	1.96	0.32	2x3	6.07	0.76	2x3	17.39	1.52
2x4	0.86	2x4	1.79	0.31	2x4	4.84	1.15	2x4	16.39	1.02
2x5	0.36	2x5	1.75	0.62	2x5	4.30	0.88	3x4	14.97	1.33
2x6	0.43	2x6	1.39	0.29	2x6	4.35	0.24	4x4	14.43	0.50
2x7	0.29	2x7	1.10	0.38	3x6	4.12	0.45	4x5	15.07	0.52

When the noise is 5%, the MSE is higher than in the noise-free case, but the evolution of the topology is unaffected. This behavior is observed in the five executions. Thus, it can be said that the learning process can effectively handle this level of noise. For a 10% of noise, in four of the executions only the last topology change is affected, whilst in the remaining execution, the same topology as in the noise-free case is obtained. This means that OSEFC can also cope with this level of noise, but the noise's effect is noticeable when a high accuracy is required. Finally, it can be observed that for a noise level of 20%, errors in the selections start to happen sooner. The table shows the evolution of the configuration obtained in four out of the five runs of the experiment. In the fifth execution, the controller's topology changed from 3x4 MFs to 3x5 (instead of 4x4, as shown in the table) and then, to 4x5. Note that most executions provide the same results, which can be seen as a sign of the robustness of the proposed methodology.

4.4.2 Simulation of real-world systems

In this section we analyze the operation of the algorithm when applied to real-world systems. Three different cases have been studied: a tank of liquid, an industrial mechanical suspension system and the control of the elevation angle of a helicopter. These experiments aim to illustrate the operation of the proposed methodology for plants that are more complex than the ones presented in the previous section. Moreover, we also present examples of the response of OSEFC in the presence of changes in the dynamics of the plants and compare this response with that provided by a non-adaptive fuzzy controller.

For these experiments, we have used randomly varying stepwise reference signals. Each simulation has been repeated five times, with different sequences of reference values, and similar results have been obtained in all of them. For the sake of brevity, in the following subsections we include only a representative example of each case considered.

It is important to remember that OSEFC does not use any specific information about the differential equations that govern the plants under control. The only information it requires to work is the sign of the monotonicity of the plant with respect to the control signal and the range of operation of the actuators and the plant's output. In order to keep this idea clear in this dissertation, we do not include here any specifics about the internal dynamics of the plants. Only a qualitative description about the systems is presented. Nonetheless, the complete information about the plants, including the mathematical models used for simulation, is summarized in Appendix A for the interested reader.

4.4.2.1 Control of a tank of liquid

As a first example, we consider the plant commonly known as *tank of liquid* or *water tank* [Cara et al., 2010b], depicted in Fig. 4.12. This plant represents a tank with a valve that controls the introduction of liquid, and a vent in its lower side that lets the liquid out. The amount of liquid introduced in the tank depends on the voltage applied to the entrance valve, whilst the liquid that flows out depends on the level reached by the liquid inside. The control objective is to compensate the loss of liquid by adjusting the power of liquid entrance, so that the final level of liquid in the tank reaches a determined height H, measured in meters. This system presents the peculiarity of limiting the actuator's operation to positive values. This means that if the liquid level has to be decreased, the only possible action is to stop the entrance of liquid in the tank (i.e., the input voltage is zero) and wait for the liquid to flow out.

For this example we have used a fuzzy controller with two inputs, namely the reference signal $r(k)$ and the plant's output $y(k)$. Initially, the controller is

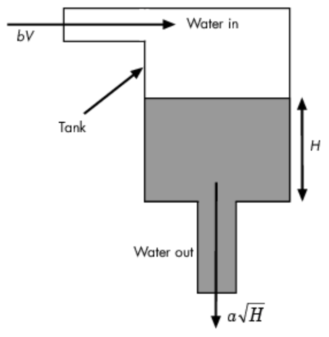


FIGURE 4.12: Schematic representation of a tank of liquid

empty, which means that every input variable has only one membership function and the only rule is initialized to zero. The control objective is to track a reference signal composed by several random steps within the range $[0.5, 4]$ m, forming a 1000-iteration long pattern. The actuator operates within the range $[0, 5]$ V. It is easy to see that the monotonicity of the plant w.r.t. the control signal is positive, since applying larger voltages to the entrance valve implies introducing more liquid in the tank. Hence, according to (4.23), the value of the normalization constant is $C = 5/3.5 = 1.43$. Finally, the controller's sampling time is $T_c = 0.1$ s.

Table 4.4 depicts the evolution of the self-evolving process, with each row representing a topology change. For each one, we show the new topology obtained, the values of the IR used to decide the next change, and the mean square error reached for the given topology after the adaptation of the new consequents, i.e., right before the next topology change. The number of samples used for computing the values of the MSE is $n_s = 1000$, which corresponds to a presentation of the reference pattern. The first two changes set the topology to 2x2 MFs, which means that both variables are relevant for the control process. After that, the importance of both inputs is kept similar, as we reach an even topology, i.e., 4x4 MFs. However, in the end $r(k)$ becomes more important, as it is observed from the fact that it receives six MFs, compared to the four assigned

TABLE 4.4: Topology evolution for the control of a tank of liquid

Configuration	IR_1	IR_2	MSE (m^2)
1x1	253.729	237.603	0.711
2x1	0.452	208.457	0.248
2x2	$7.104 \cdot 10^{-4}$	2.584	0.052
2x3	$1.989 \cdot 10^{-4}$	$4.525 \cdot 10^{-4}$	0.041
2x4	$1.219 \cdot 10^{-4}$	$8.874 \cdot 10^{-5}$	0.034
3x4	$9.639 \cdot 10^{-5}$	$6.152 \cdot 10^{-5}$	0.029
4x4	$4.624 \cdot 10^{-5}$	$3.536 \cdot 10^{-5}$	0.026
5x4	$5.497 \cdot 10^{-5}$	$2.961 \cdot 10^{-5}$	0.024
6x4			0.023

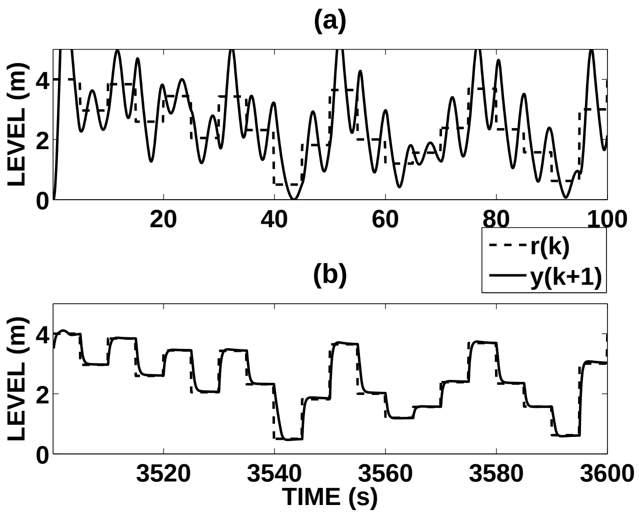


FIGURE 4.13: Reference tracking for the tank of liquid. (a) At the beginning. (b) After one hour

to $y(k)$. This means that, in order to achieve a better accuracy, the control policy has to be finer for the values of $r(k)$.

The tracking of the reference signal at different moments of the experiment is depicted in Fig. 4.13. At the beginning of the simulation the controller is empty, and therefore, incapable of properly controlling the plant (Fig. 4.13(a)). However, the addition of new membership functions and rules leads to a good

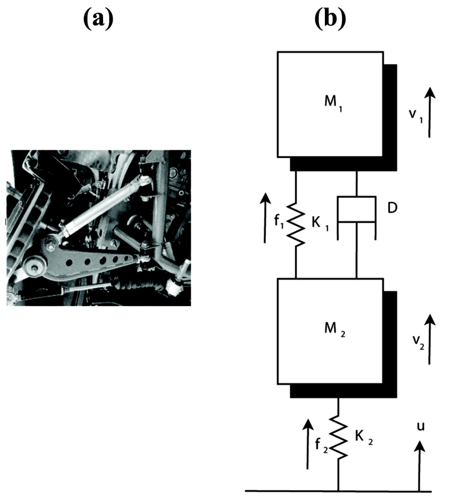


FIGURE 4.14: Suspension system used for experimentation. (a) Suspension system. (b) Simplified mathematical model

performance before one hour elapses (Fig. 4.13(b)). In the lower plot, it is also possible to observe the effect of the actuator's limitations, i.e., the decrease of the liquid level is always slower than its increase (e.g., at time 3540 s). As already mentioned, the cause for this is that the only possible action to decrease the level of liquid is closing the valve completely. Nonetheless, it is also worth noting that the level of liquid does not go beyond the desired value at any point, which means that the controller has adequately learned to re-open the valve.

4.4.2.2 Control of a mechanical suspension system

In this section, we consider the control of an industrial mechanical suspension system, as depicted in Fig. 4.14 [Ogata, 2001]. This model is formed by a sprung mass M_1 that represents a body, and an unsprung mass M_2 representing the mechanical components whose duty is to move the body. Between these two elements there is a transfer system, which is characterized by a spring K_1 and a damper D. Finally, a spring K_2 serves as a model of the compressibility of the mechanical components. The values of all the parameters of the system are provided in Appendix A.2. In this problem, the objective is to control the vertical velocity of the body v_1, measured in meters per second.

To control this plant, a fuzzy controller with two inputs has been used, namely

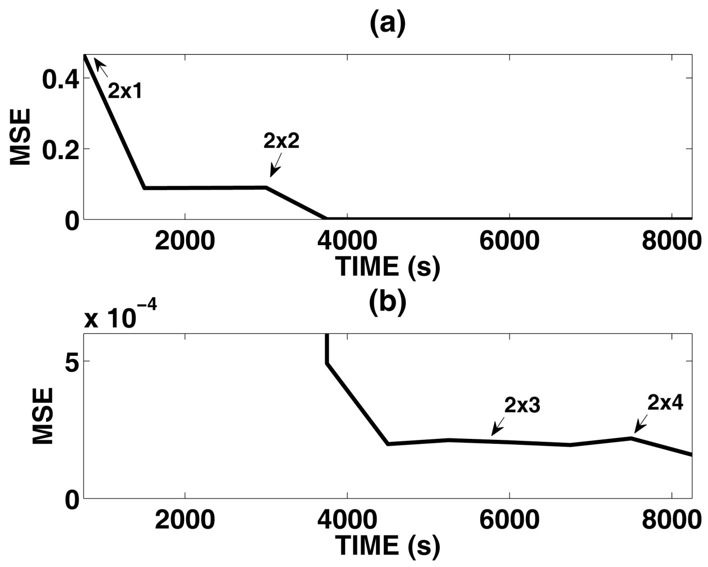


FIGURE 4.15: MSE evolution for the control of the suspension system. (a) At the beginning. (b) At the end

the reference signal $r(k)$ and the system's output $y(k)$. Both inputs take values in the range $[-1, 1]$ m/s. The actuator's range is scaled and limited to the values $[-1, 1]$, which represent the minimum and maximum voltage allowed by the physical system. The sign of the monotonicity of the plant w.r.t. to the control signal has been experimentally identified by applying different control signals to the plant, with a common initial state. It has been observed that larger control signals produce larger outputs, i.e., the monotonicity is positive. Thus, the constant for the consequents' adaptation is set to $C = 1$, as given by (4.23). Initially, the controller is empty, i.e., only one membership function is assigned to each input, and therefore, only one zero-initialized fuzzy rule is defined.

The reference trajectory is given by a randomly changing stepwise signal in the range $[-1, 1]$ m/s. To simulate the plant, the fourth-order Runge-Kutta method has been used, with sampling time $T_p = 0.05$ seconds. The controller's sampling time is also set to $T_c = 0.05$ s. The total simulation time was 9000 seconds (150 minutes).

Fig. 4.15 depicts the evolution of the MSE through the entire simulation. The moments in which the topology has changed are indicated with arrows. We have split the figure into two parts to make the visualization easier. Fig. 4.15(a)

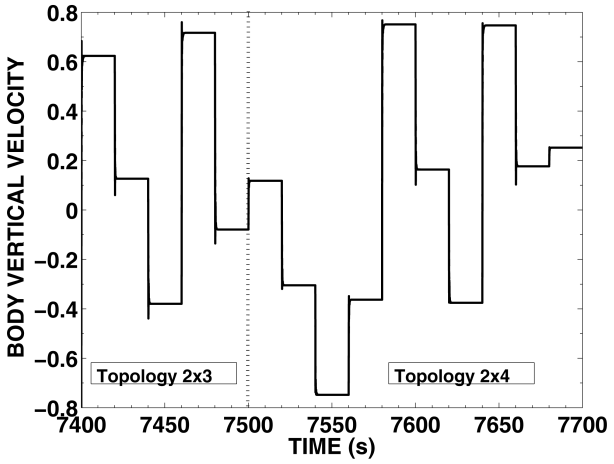


FIGURE 4.16: Reference tracking for the suspension system

shows the evolution at the beginning, when the error is large, whilst Fig. 4.15(b) depicts the evolution at the end, when the MSE takes small values. OSEFC assigns the first membership function to the input $r(k)$, thereby indicating that this variable is relevant for the control process. However, after that, the second input becomes clearly more important, as it is observed from the fact that $y(k)$ receives four membership functions, compared with the two assigned to $r(k)$.

In addition, this figure also shows how the tracking error decreases due to the modification of the topology. This fact is also depicted in Fig. 4.16, which represents the reference tracking during 300 seconds. During the first 100 seconds the topology was 2x3. In this time span the control is quite satisfactory, although some overshoot is shown, specially when large changes occur in the reference signal. At time 7500 s, the topology changes to 2x4, i.e., a new membership function is added to the second input variable, $y(k)$. This change helps reducing this overshoot, although it is not completely removed.

4.4.2.3 Control of a 1-DOF helicopter

The last application considered in this section is a 1-degree of freedom (DOF) helicopter setup (see Fig. 4.17 and Appendix A.3). This setup consists of a beam attached to a fixed pole. The beam can freely rotate in the vertical plane while the horizontal position is fixed. At the end of the beam there is a DC motor

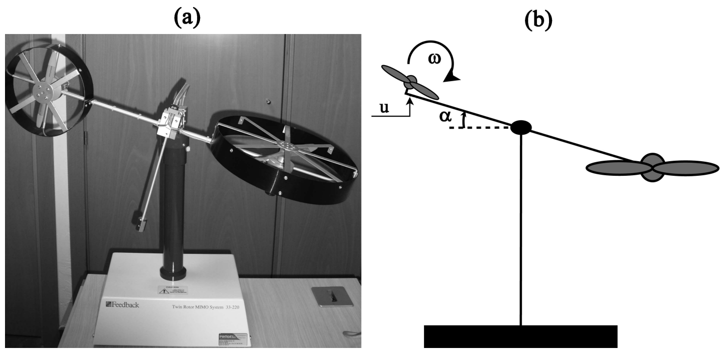


FIGURE 4.17: Helicopter setup used for experimentation. (a) Helicopter setup. (b) Schematic model of the helicopter setup

with a propeller attached. The control input u is the voltage applied to the motor to control the elevation angle of the beam α. It takes values in the range $[-1, 1]$, where -1 represents the maximal voltage (in absolute value) that makes the propeller and the beam rotate in the negative sense of α, and +1 is the maximal voltage that causes rotation of the beam in the positive sense of α. Since the horizontal position of the beam is fixed, a constant voltage is applied to the front rotor. There are two measured outputs, namely the angular velocity ω of the propeller and the angle α of the beam. The former takes values in the range [-1,1], representing the maximal negative and positive angular velocity measured in radians per seconds, respectively. The latter is measured in radians. The control objective is to make the angle α follow an specific reference trajectory, formed by a series of random steps in the range [-1,1] rad.

For this experiment, the sampling time is set as $T_p = T_c = 0.001$ s. A three-input fuzzy controller has been used; these inputs are the plant's output $y(k)$, the error at the plant's output $e(k)$ and the derivative of the error $\dot{e}(k)$. They take values in the ranges $[-1, 1]$, $[-0.006, 0.006]$ and $[-0.003, 0.003]$, respectively. Initially, only one membership function is assigned to each input variable, and the only rule is initialized to zero. From the description of the plant, it can be deduced that the monotonicity of the plant w.r.t. the control signal is positive. Thus, the value of C is set to 1.

Table 4.5 shows the evolution of the self-structuring process, with each row representing a topology change. For each of them, we show the new topology obtained after the change, the values of the indexes of responsibility used

TABLE 4.5: Topology evolution for the control of a 1-DOF helicopter

Configuration	IR_1	IR_2	IR_3	MSE (rad^2)
1x1x1	127.47	160.92	$6.20 \cdot 10^{-8}$	0.23
1x2x1	280.41	$2.95 \cdot 10^{-10}$	$4.3 \cdot 10^{-8}$	0.043
2x2x1	$3.16 \cdot 10^{-7}$	$2.89 \cdot 10^{-9}$	$3.89 \cdot 10^{-9}$	0.034
3x2x1	$1.42 \cdot 10^{-6}$	$2.72 \cdot 10^{-9}$	$2.96 \cdot 10^{-9}$	0.022
4x2x1	$2.85 \cdot 10^{-7}$	$2.70 \cdot 10^{-9}$	$2.65 \cdot 10^{-9}$	0.018
5x2x1				0.016

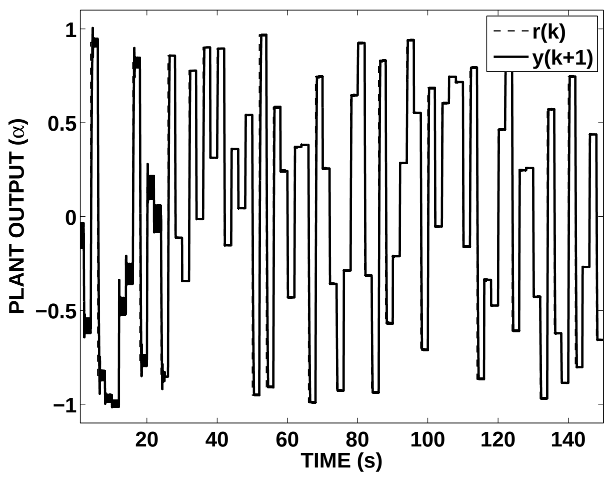


FIGURE 4.18: Reference tracking for the 1-DOF helicopter during the first 150 seconds

to decide the next change, and the MSE reached for the given topology after the adaptation of the new consequents. The MSE is computed every 25 s. The first new membership function is added to the plant's error $e(k)$. The reason for this decision is that the control objective is to make the error equal to zero, and therefore, its value is important for the control process. After that, the plant's output clearly becomes the most important variable for the control process. It is interesting to note that the derivative of the error does not receive any MF. This means that the controller considers that it is not relevant for the control. In other words, this example shows the capability of the proposed methodology of choosing only the important control variables among a group of candidates.

Finally, Fig. 4.18 depicts the tracking of the reference signal during the first 150 seconds of the experiment. Initially, the lack of knowledge and the simple topology selected makes the controller unable to stop the oscillations at the plant's output. However, after developing the structure and adapting the rule consequents, the control achieves a satisfactory performance.

4.4.2.4 Robustness against changes in the plant

The proposed method does not use any specific information or makes any hypotheses about the system it controls, except for the sign of the monotonicity with respect to the control signal u. Thus, an important feature of OSEFC is its robustness against unexpected changes in the plant under control.

In order to visualize this property, the suspension system (section 4.4.2.2) and the 1-DOF helicopter (section 4.4.2.3) are used again, with the same initial configurations and parameters as previously commented. In both cases, the initial values of the plant's parameters are the ones shown in Tables A.2 and A.3, respectively. For the suspension system, the mass of the moving body M_1 is multiplied by 5 after 175 minutes, while in the case of the helicopter setup the gain K_1 is multiplied by 2 after 150 s. The effect of these changes is similar in both cases: a clear deterioration in the performance of the control takes place right after the change. Nevertheless, the proposed methodology starts counteracting this deterioration from the beginning and soon the control is as good as it was before the change in the plant's dynamics, as depicted in Fig. 4.19 for the case of the helicopter.

The response of the proposed OSEFC has been compared with the response of a classic non-adaptive fuzzy controller [Lalouni et al., 2009], for the case of a change in the dynamics of the suspension system. Fig. 4.20 shows the tracking of the reference signal in both cases and Fig. 4.21 depicts the tracking errors. It is observed that both controllers perform equally well before the change. However, the response of the controllers is different when the change in the plant's dynamics occurs. On the one hand, OSEFC is able to learn the new behavior and produces a satisfactory tracking after some time. On the other hand, the non-adaptive controller, which has been designed for the previous plant configuration, is not capable of properly controlling the plant under the new conditions.

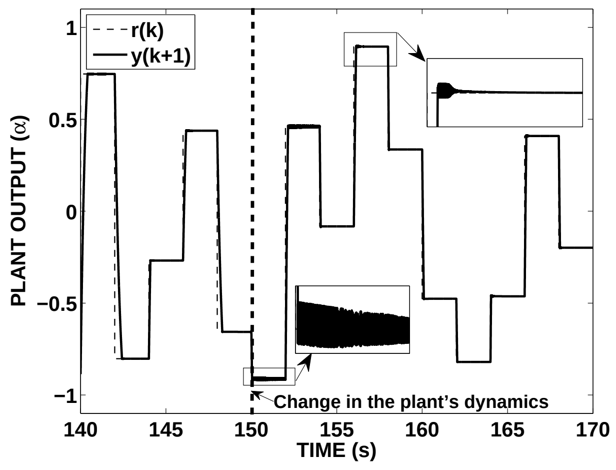


FIGURE 4.19: Reference tracking for the helicopter setup after a change in the plant's dynamics. The vertical dotted line indicates the moment of the change in the plant's dynamics

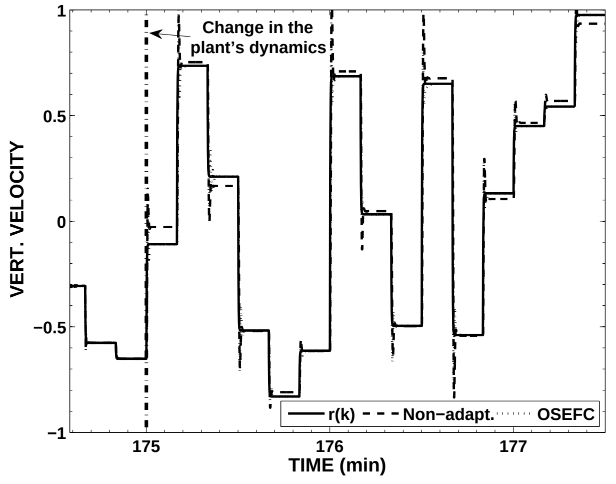


FIGURE 4.20: Reference tracking for the suspension system after a change in the plant's dynamics. Comparison with a non-adaptive fuzzy controller

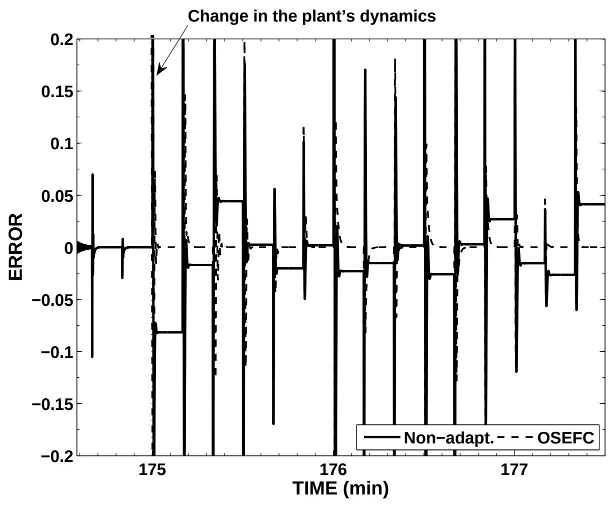


FIGURE 4.21: Comparison of the tracking errors after a change in the plant's dynamics for a non-adaptive fuzzy controller and the proposed OSEFC

4.4.3 Experiments with a real system: Control of a nonlinear servo system

In this section, the proposed OSEFC is applied to the real nonlinear servo system depicted in Fig. 4.22. This experimental setup consists of a weight attached to a disk that is actuated by a DC motor and rotates in a vertical plane. The presence of the extra weight introduces a gravity term that causes the system's nonlinear behavior. Thus, the control objective is to compensate this nonlinearity in order to make the position of the disk (measured as an angle θ) track a desired reference trajectory r. The complete description of the plant is included in Appendix A.4. It is important to note that the results reported in this section were obtained in real-time experiments developed at the Delft Center for Systems and Control (Delft University of Technology, The Netherlands), under the supervision of Prof. R. Babuška.

For this experiment, the parameter learning method based on global information (see section 4.3.3) has been used for learning the rule consequents. Three inputs are considered for the fuzzy controller, namely the reference value $r(k)$, the current output angle $\theta(k)$, and the angular velocity $\dot{\theta}(k)$. The inputs take values in the ranges $[-\pi, \pi]$ rad, $[-\pi, \pi]$ rad, and $[-5, 5]$ rad/s, respectively. Initially, two MFs are assigned to each input, thus 8 fuzzy rules form the initial controller.

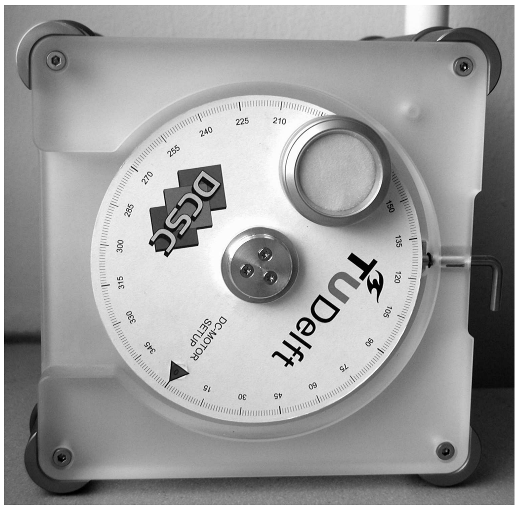


FIGURE 4.22: Nonlinear servo system

However, since all the consequents are initialized to zero, the situation is equivalent to having an empty controller.

The actuator's range is limited to $[-10, 10]$ V. The speed and sense of the rotation of the disk depend on the voltage applied to the DC motor. In the initial experiments, it was observed that larger control signals produced larger values of the position angle θ. Thus, the monotonicity of the plant with respect to the control signal is positive and the value of the constant C for the adaptation of the consequents can be set to $C = 10/\pi$, as given by (4.23). The sampling period is chosen as $T_c = 0.05$ s. The reference signal is given by a randomly changing function in the interval $[-\pi, \pi]$ rad, forming a 1000-sample long pattern (i.e., 50-second long pattern). The total time for the experiment is 45 minutes.

The operation of the control system at different moments of the experiment is depicted in Fig. 4.23, 4.24 and 4.25. On the top part of the figures, the reference tracking is shown, with the solid line representing the reference signal and the dotted line corresponding to the plant's output. In addition, the control signal applied is depicted in the lower part of the plots. At the beginning of the experiment (Fig. 4.23), the lack of knowledge about the plant causes oscillations in the plant's output. However, after only 20 seconds, the system starts following the trajectory marked by the reference signal, although the control is still not satisfactory. In Fig. 4.24(a), the effect of the global learning can be observed: The

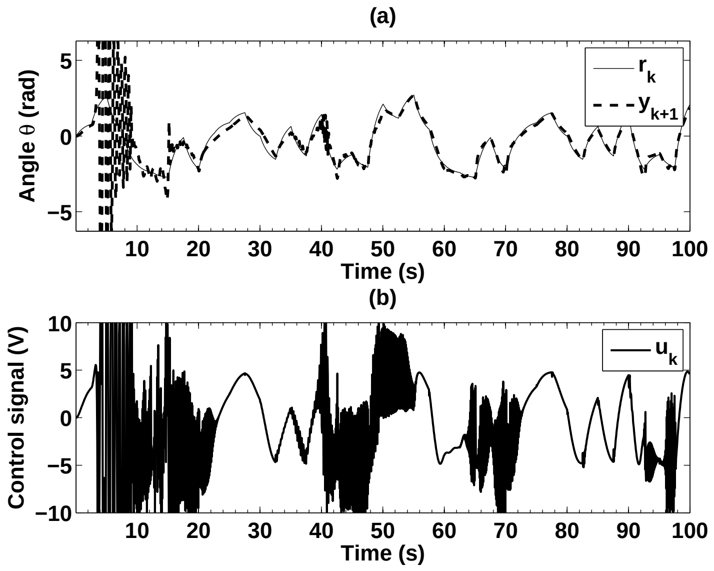


FIGURE 4.23: Reference tracking for the nonlinear servo system during the first 100 seconds. (a) Reference tracking. (b) Control signal

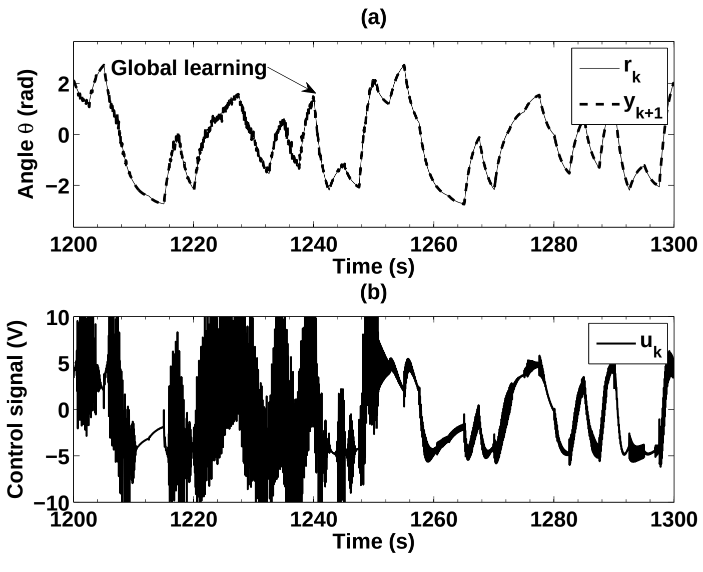


FIGURE 4.24: Effect of the global learning on the reference tracking for the nonlinear servo system. (a) Reference tracking. (b) Control signal

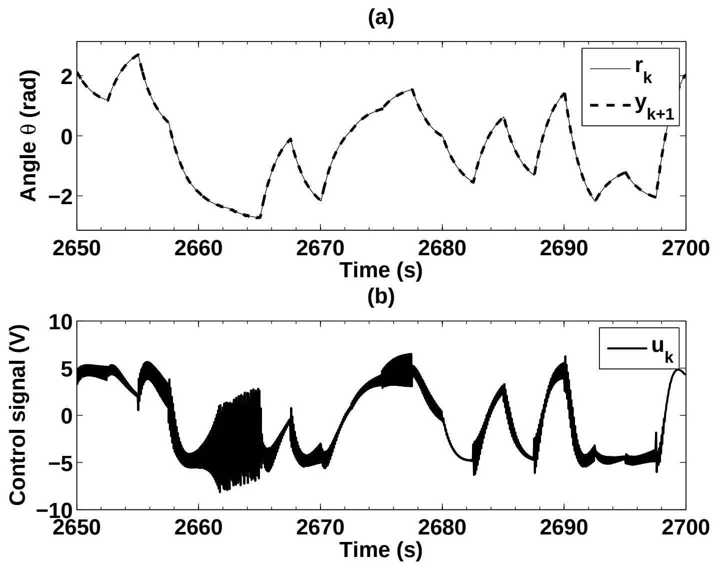


FIGURE 4.25: Reference tracking for the nonlinear servo system at the end of the experiment. (a) Reference tracking. (b) Control signal

TABLE 4.6: Topology evolution for the control of the nonlinear servo system

Time (s)	Configuration	IR_1	IR_2	IR_3	MSE (rad^2)
0	2x2x2	7.761	18.531	4.538	2.897
748.3	2x3x2	13.038	29.533	7.512	0.153
1469.5	2x4x2	12.899	12.609	7.318	0.191
2173.7	3x4x2	12.033	12.470	7.135	0.020
2274.5	3x5x2	11.875	6.714	6.997	0.019
2376.4	4x5x2	11.703	6.652	6.978	0.015
2685.6	5x5x2				0.014

GL-module starts operating at time 1238 s, helping to eliminate the small oscillations that were present before. Finally, Fig. 4.25 depicts the control at the end of the experiment. It can be observed that the tracking at this point is virtually perfect.

The evolution of the controller's structure is summarized in Table 4.6. Each row represents a topology change, showing the time of change, the new distribution of MFs, the values of the IR used to select the variable to receive the new MF,

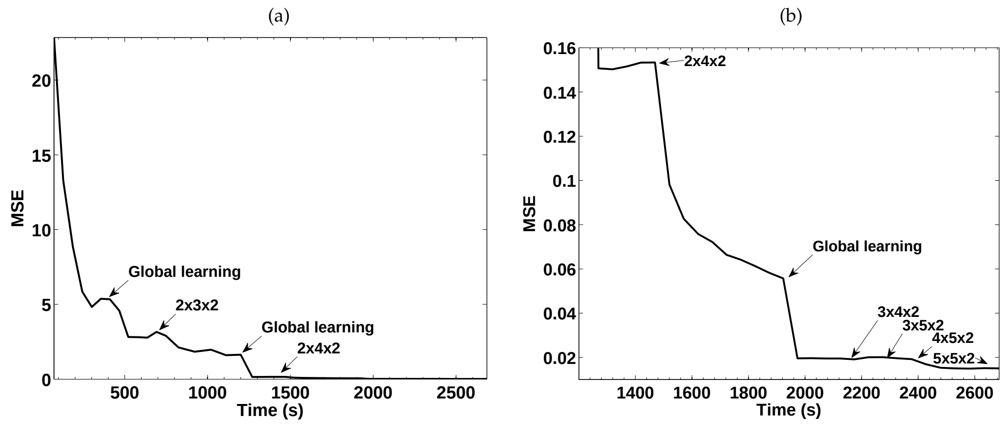


FIGURE 4.26: Evolution of the MSE during the control of the nonlinear servo system. (a) During the entire experiment. (b) Details of the evolution at the end of the experiment

and the MSE reached with that topology after tuning the new consequents. The MSE is computed after each presentation of the reference pattern, i.e., $n_s = 1000$ samples. In the first two changes, it is clear that the second input (i.e., the angle θ) is the most important, as the difference between its IR and the others is high. However, after optimizing the consequents for the topology 2x4x2, IR_1 and IR_2 have very similar values, the former being slightly higher. This means that, at this point, the error due to the plant's output has been corrected by the two extra MFs and the lack of precision on the first variable (i.e., the reference value) is more relevant. At the end of the execution, both inputs are equally important for the control, as they both have the same number of MFs. The initial configuration of the input $\dot{\theta}$ does not change, which indicates that a linear approximation is sufficient in that dimension.

To illustrate the positive effect of the global learning in this problem, we have depicted the evolution of the MSE in Fig. 4.26. Fig. 4.26(a) shows the evolution of the MSE for the first topology changes, whilst Fig. 4.26(b) depicts the details of this evolution once the error reaches small values. The arrows point out the moments in which new MFs are added, as well as the moments in which the global learning starts operating for some topologies. Firstly, it is observed that the error presents a clear decreasing tendency, which is faster at the beginning but continues until the end of the experiment, i.e., all the topology changes help

TABLE 4.7: Final rules for $\dot{\theta} = X_3^1$ in the control of the nonlinear servo system

Input θ	Input r				
	X_1^1	X_1^2	X_1^3	X_1^4	X_1^5
X_2^1	4.77	52.20	53.69	69.73	104.57
X_2^2	-30.85	23.50	24.95	43.33	83.24
X_2^3	-48.44	8.39	10.38	29.47	70.91
X_2^4	-70.15	-10.05	-8.46	11.47	54.80
X_2^5	-90.84	-45.52	-44.18	-28.95	4.11

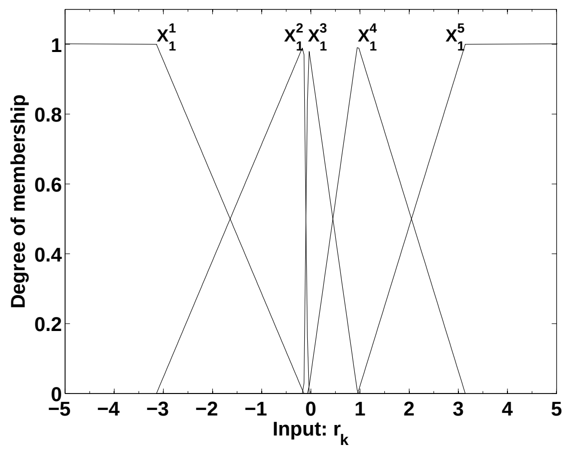
TABLE 4.8: Final rules for $\dot{\theta} = X_3^2$ in the control of the nonlinear servo system

Input θ	Input r				
	X_1^1	X_1^2	X_1^3	X_1^4	X_1^5
X_2^1	-4.77	43.37	44.98	61.26	96.63
X_2^2	-39.81	15.15	16.60	35.06	75.15
X_2^3	-56.55	0.25	2.21	21.20	62.43
X_2^4	-77.44	-18.33	-16.70	2.98	45.78
X_2^5	-98.57	-53.87	-52.57	-37.54	-4.89

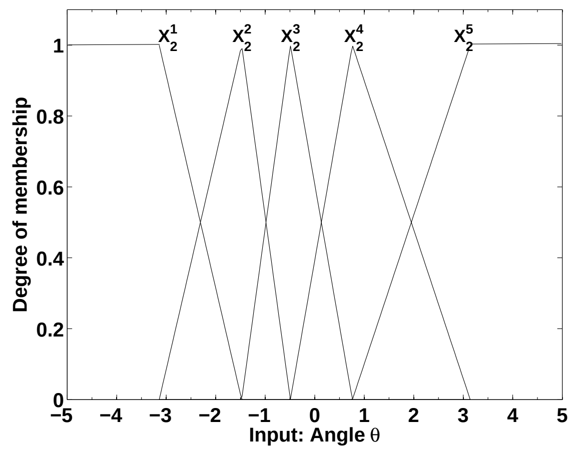
to improve the performance of the controller. Secondly, the figure shows that the global learning procedure improves the MSE for each topology, thereby helping to reduce the number of topology changes required.

Finally, we show the structure of the final fuzzy controller. Fig. 4.27 depicts the final location of the MFs for the three input variables. In the case of the angular velocity $\dot{\theta}$, this location is the same as the initial configuration, since no MFs have been added to this input. At the end of the experiment, the controller is formed by 50 rules, whose consequents are gathered in two tables: Table 4.7 presents the consequents of the 25 rules for MF X_3^1, whilst Table 4.8 shows the consequents of the remaining 25 rules, for MF X_3^2. In these tables, the rules are identified by the MFs that form their antecedent part, and each cell shows the value of the corresponding consequent.

(a)



(b)



(c)

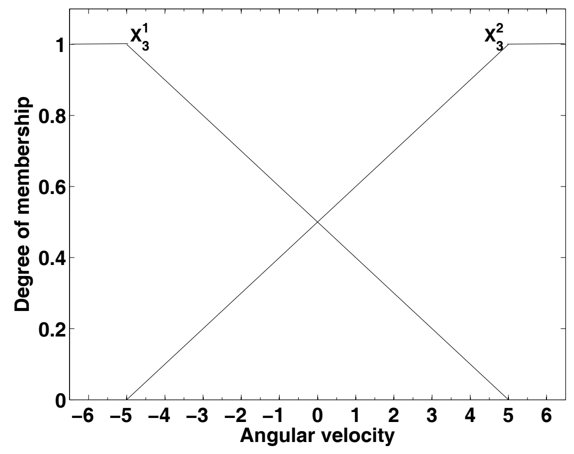


FIGURE 4.27: Final location of the MFs assigned to the inputs in the control of the nonlinear servo system. (a) Input $r(k)$. (b) Input $\theta(k)$. (c) Input $\dot{\theta}(k)$

4.5 Conclusions

In this chapter, a novel online self-evolving fuzzy controller, named OSEFC, has been presented. The aim of this methodology is to provide a means of obtaining fuzzy controllers when no prior information about the plant is available. Hence, OSEFC is able to self-design from scratch while working online. It does not use the information regarding the differential equations that govern the plant or any of their bounds. The only prior information required is the sign of the monotonicity of the plant with respect to the control signal.

The learning mechanism of OSEFC is based on the property of universal approximation of fuzzy systems and is divided into two phases: online parameter learning for the optimization of the rule consequents, and topology self-evolution based on the addition of antecedent membership functions and rules. In both stages, the information used for the learning is obtained directly from the normal operation of the plant.

Two complementary methods have been proposed for the adaptation of the rule consequents. The first one is an online local adaptation process based on the analysis of the current error at the plant's output. It applies a reward/penalty policy to move the consequents in the direction that reduces the error at the plant's output. In the second method, the global I/O information provided by the plant's operation is used to obtain a finer tuning of the rule consequents. In this method, a gradient-based technique is used to minimize the error in the controller's output as a means to indirectly minimize the error at the plant's output. Additionally, the online local adaptation method is used as a supervisor to guarantee that all the changes applied to the consequents are consistent with the local control objective, i.e., to guarantee that the changes help to keep the plant under control.

The topology self-evolution method proposed allows OSEFC to start operating with a simple structure and then add more membership functions and rules as they are needed. In order to provide more robustness to the method, this phase is based on the analysis of the error in the entire operating range of the plant. This analysis is used to identify the input variables that are mostly responsible for the error, and therefore, need more membership functions. Moreover, it has been shown that this method can be used as an input selector mechanism when there is no precise knowledge about which inputs are relevant for the control.

The main features of the proposed OSEFC can be summarized as follows:

- OSEFC does not require a model for the plant or its differential equations. Only the sign of the monotonicity of the plant with respect to the control signal is needed.
- No previous knowledge about the control policy is needed. OSEFC can start working with an only rule initialized to zero and will self-develop as it performs the control.
- OSEFC is robust in the sense that it is able to cope with unexpected isolated changes in the behavior of the plant.
- OSEFC performs well in noisy situations, due to its adaptive nature.
- There is no need to precisely define the set of input variables for the controller, as OSEFC is able to select those that are relevant from a set of candidate inputs.

Simulations and real experiments have been presented to illustrate all these features. Although the lack of information about the plant makes it difficult to provide a formal demonstration of the stability, a qualitative discussion about the stability of our approach has also been provided.

Finally, the proposed methodology also presents some limitations that are a consequence of the lack of information about the plant. Firstly, since the controller self-designs from scratch while working online, the full methodology proposed here cannot be applied to plants with critic states that must never be reached. In this case, the controller can be pre-trained to obtain a control policy that is sufficiently reliable when it starts being applied to the plant. Secondly, its application is limited to plants in which the controller's sampling period T_c can be set in a way such that the plant's output depends directly on the control signal applied in the previous time step. Nevertheless, our proposal sets an initial framework in which to develop new controllers to achieve the ultimate goal of online intelligent control with minimum previous knowledge about the plant.

Chapter 5

OSENF-TaSe: Online Self-Evolving Neuro-Fuzzy Controller based on the TaSe-NF model

In the previous chapter, we presented a new methodology for the online self-evolution of a fuzzy controller that aims to provide accurate controllers when there is no prior information about the plant. That methodology mainly focuses on the accuracy, in the sense that the only objective is to reduce the control error. However, other properties, such as the reduction of the complexity and/or the improvement of the interpretability, are also desirable.

This chapter presents a new online self-evolving neuro-fuzzy controller based on the Taylor Series Neuro-Fuzzy (TaSe-NF) model. Under the assumption of no prior knowledge about the differential equations that define the plant to be controlled, this methodology is capable of incrementally evolving the structure of the controller and adapting its parameters online, while controlling the plant. The new methodology uses a scatter distribution of the fuzzy rules, thus reducing the number of rules in the fuzzy controller. Furthermore, the use of the TaSe-NF model to represent the antecedents of the rules enhances the interpretability of the obtained rules.

5.1 Motivation and goals

There are two characteristics highly important when designing a fuzzy controller: On the one hand, the interpretability of the rules is important in order to allow for the verification and completion of the rule base by human experts [Guillaume, 2001]. On the other hand, the need of implementing fuzzy controllers in embedded systems makes it necessary to reduce the number of rules defined [Doctor et al., 2005b; Hagras et al., 2007].

Traditionally, the design of a fuzzy controller has relied on the expertise of human operators to establish the linguistic terms and fuzzy rules [Chang, 2010]. However, the process of extracting this knowledge from human experts may not be possible in some cases or may be very time-consuming [Chen et al., 2008]. As a consequence, much effort has been dedicated to the development of methods for the automatic generation of fuzzy rules from input/output data [Chen et al., 2009; Navale and Nelson, 2010; Pomares et al., 2000; Rojas et al., 2000]. However, training data is not always available and online techniques become necessary [Angelov, 2004; Cara et al., 2010c; Chang, 2010; Chen et al., 2008; Gao and Er, 2003].

Some methods [Chang, 2010; Chen et al., 2008; Gao and Er, 2003; Hsu, 2007] base their adaptation laws in the Lyapunov theory, so the stability of the controllers can be proven. Nevertheless, this type of approach requires some knowledge about the internal dynamics of the plant (e.g., the bounds of the differential equations). On the other hand, there are also methods that intend to reduce the information required to develop the controller and therefore, do not apply the Lyapunov approach [Angelov, 2004; Cara et al., 2011a, 2010c; de Barros and Dexter, 2007]. Although these approaches cannot provide a mathematical proof of the stability, they provide good control performance in situations in which there is no prior information about the plant to be controlled.

In [Cara et al., 2011a, 2010c; Park et al., 2005; Phan and Gale, 2008; Tewari and Macdonald, 2010], grid partitions have been used to divide the input space. Although this option is conceptually simpler, it presents the drawback of the well-known curse of dimensionality, i.e., the number of rules grows exponentially with the number of input variables and the number of membership functions per input variable [Herrera et al., 2007]. Scatter-partitioning fuzzy systems do not suffer from this problem and are therefore good solutions for dealing with control problems. In [Angelov, 2004], a non-grid clustering technique is applied. In this approach, every new incoming data point is analyzed to decide if a new rule has to be added to the controller. However, basing this decision in one data point at a time provides less robustness than using a set of points.

Under certain conditions, fuzzy systems, and especially scatter-partitioning ones, are also called neuro-fuzzy systems [Haykin, 1998; Jang and Sun, 1993], as they aim to combine the interpretability of fuzzy systems with the accuracy of neural networks. Within this scope, each rule can be seen as a sub-model that performs the approximation in a region of the input space, and the union of all the sub-models globally approximates the underlying model. However, traditional methods for automatic learning of neuro-fuzzy systems from I/O data barely deal with simultaneous optimization of the local and global models [Leng et al., 2006; Mollov et al., 2004]. This can lead to a deficient interpretability of the extracted set of rules, which can cause a lack of usability of the models by the experts [Johansen and Babuska, 2003]. Some works have proposed solutions to this problem by using a special type of MF and a particular structure for the rule consequents in grid-based fuzzy systems [Bikdash, 1999; Guillaume, 2001; Herrera et al., 2005; Sala and Ario, 2009; Tewari and Macdonald, 2010]. They have been recently adapted to scatter-partitioning fuzzy systems (Taylor Series Neuro-Fuzzy, or TaSe-NF model) [Herrera et al., 2011b] thanks to a modification of the fuzzy inference process.

In this chapter, we present a new Online Self-Evolving Neuro-Fuzzy controller based on the TaSe-NF model, which we call OSENF-TaSe. This new methodology can be seen as an extension of the method presented in chapter 4 (see Table 5.1), in the sense that it adds two new capabilities [Herrera et al., 2011b]:

- It preserves the interpretability of the local models by using the TaSe-NF architecture.

- It avoids the exponential growth of the number of fuzzy rules by using a scatter partitioning of the input space.

TABLE 5.1: Comparison of the properties of OSEFC and OSENF-TaSe

	OSEFC	OSENF-TaSe
Type of MFs	Triangular	Gaussian
Partitioning of the input space	Grid	Scatter
Inference model	Zero-order TSK	Zero-order TaSe-NF
Previous knowledge	– Sign of the monotonicity – Set of candidate input variables	– Sign of the monotonicity – Set of input variables
Parameter learning	– Online with local information (consequents) – Optional use of global information (consequents)	– Online with local information (consequents) – Global information (consequents and antecedents)
Structure evolution based on	Addition of MFs to the input that is most responsible for the error	Splitting of the rules with high control errors in their areas of influence
Number of rules added each time	All the combinations of the new MF with the MFs in all the other inputs	One
Advantages	– Automatic input selection – Conceptually simpler	– Reduction of the number of rules – Enhancement of the interpretability
Drawbacks	Curse of dimensionality	More nonlinear parameters

In addition, OSENF-TaSe preserves the fundamental properties of OSEFC, namely:

- It assumes no prior knowledge about the plant. Therefore, it starts operating with only one rule and learns both the structure and the parameters of the neuro-fuzzy controller while it performs the control.
- The learning method of the OSENF-TaSe is also based on the online local learning of the rule consequents and a structure self-evolution algorithm that adds rules to the system. As in the case of OSEFC, the former aims to reduce the current error at the plant's output, whilst the latter distributes the rules based on the analysis of the entire error surface.

The main properties of OSEFC and OSENF-TaSe are summarized and compared in Table 5.1.

The reminder of this chapter is organized as follows. Section 5.2 states the problem that is tackled and presents the structure of the neuro-fuzzy controller. The TaSe-NF model is described in some detail in section 5.3. The architecture of the proposed OSENF-TaSe controller is then presented in section 5.4. Section 5.5 presents simulation results that illustrate the capabilities of OSENF-TaSe. Finally, conclusions are drawn in section 5.6.

5.2 Problem formulation

The problem tackled in this chapter is very similar in nature to the one presented in chapter 4. For the sake of simplicity, in this section we just summarize its main aspects and assumptions and the elements for which the notation used is different. We refer the reader to section 4.2 for the complete details.

We consider again the single-input single-output plant whose dynamics are given by the set of differential equations (4.1), which can be expressed in terms of its difference equations (4.2) when the sampling time T_p is short enough. As in section 4.2, we assume that the controller's sampling time T_c can be selected in a way such that the plant's output at time $k + 1$ depends on the control signal applied at time k, i.e., the effect of the control action u_k is reflected at the plant's output at the next time step y_{k+1}. Again, it is to be highlighted that this does not mean that previous control actions do not affect the plant's output as well, but rather that u_k actually has an effect that is measurable at the next time step.

In order for the proposed methodology to be valid to control the plant (4.1), we continue imposing the condition of the existence of a control policy capable of translating the plant's output to the desired value (within the operation range) [Pomares et al., 2002b]. This means that there must never exist a state in which the plant's output does not depend on the control input. In addition, it is also required that the monotonicity of the plant with respect to the control signal has a constant sign.

The control objective is to make the plant's output track a given reference signal r_k. In the absence of actuator bounds, we can assume that there exists an optimal control policy G capable of achieving the control objective, i.e.,

$$u_k = G(r_k, \vec{x}_k) \tag{5.1}$$

where $\vec{x}_k$ is a finite set of variables that determine the control policy. Again, our objective is to approximate such control policy.

In this case, we choose to use a zero-order TSK fuzzy system with scatter partitioning of the input space instead of a complete set of rules. Thus, the rule base is formed by N_r IF-THEN rules of the form

$$\mathfrak{R}^i : \text{IF } x_1 \text{ is } \mu_1^i \text{ AND } x_2 \text{ is } \mu_2^i \text{ AND... } x_n \text{ is } \mu_n^i \text{ THEN } u = Q^i$$

where $i = 1...N_r$, $\vec{x} = [x_1, ..., x_n]^T$ is the input vector to the fuzzy system, μ_j^i represents the i-th fuzzy set defined for the input variable x_j, which is characterized by the membership function $\mu_j^i(x_j)$, and Q^i is a scalar value that represents the rule consequent. In this chapter, we assume that a set $\vec{x}$ of input variables that are enough to approximate the control policy is known beforehand (e.g., based on qualitative knowledge about the plant).

If we choose the product as the t-norm for the fuzzy inference process, and the weighted average as the defuzzification method, the output of the fuzzy controller can be expressed as

$$\hat{u}_k = \hat{G}(\vec{x}_k; \Phi_k) = \frac{\sum_{i=1}^{N_r} Q^i \cdot \mu^i(\vec{x}_k)}{\sum_{i=1}^{N_r} \mu^i(\vec{x}_k)} = \frac{\sum_{i=1}^{N_r} Q^i \cdot \prod_{j=1}^{n} \mu^i_j(x_j)}{\sum_{i=1}^{N_r} \prod_{j=1}^{n} \mu^i_j(x_j)} \tag{5.2}$$

where Φ_k is the set of parameters defining the fuzzy controller at time k and $\mu^i(\vec{x}_k) = \prod_{j=1}^{n} \mu^i_j(x_j)$ is the firing strength of the i-th rule.

In general, scatter partitioning fuzzy systems represent the antecedent fuzzy sets μ^i_j by Gaussian membership functions of the form

$$\mu^i_j(x_j) = \exp\left(-\frac{(x_j - c^i_j)^2}{2{\sigma^i_j}^2}\right) \tag{5.3}$$

where c^i_j is the center of the membership function and σ^i_j is its radius. We have selected to use the same radius σ for all the MFs, as it simplifies the optimization of the neuro-fuzzy system.

As in the case of OSEFC, it is important to remember that approximating the inverse function of a plant while keeping it under control is a challenging task with special properties that may not be present in traditional functional approximation problems, e.g., the I/O data arrives while the controller is working and is unlikely to span the entire operation range. Moreover, the approximation has to be obtained while guaranteeing that the plant is under control.

5.3 The TaSe-NF model

In this section, the TaSe-NF model [Herrera et al., 2011b] is described. TaSe-NF is a scatter-partitioning modified neuro-fuzzy model whose main characteristic is its ability to perform a global model approximation whilst preserving the interpretation of the local sub-models. It is a TSK modified model equivalent to radial basis function networks that offers the possibility to extract a set of locally interpretable fuzzy rules. These characteristics are possible thanks to a special partitioning of the input space by means of a modified calculation of the final normalized activation of the rules, which controls their overlap. The rule consequents of the model take the form of a truncated Taylor series, which means that they describe the behavior of the model output around the rule centers. In section 5.3.1, the equivalence between TSK fuzzy systems and radial basis function networks is quickly revised. Then, the details about the TaSe-NF model are presented, paying special attention to the partitioning of the input space (section 5.3.2.1) and its effect on the interpretability of the fuzzy rules (section 5.3.2.2).

5.3.1 Equivalence between RBFNNs and TSK fuzzy systems

The equivalence between TSK fuzzy systems and Radial Basis Function Neural Networks (RBFNNs) is well known [Jang and Sun, 1993]. This equivalence essentially occurs when the output of the network is normalized and Gaussian functions - or Radial Basis Functions (RBFs) in general - of the form (5.3) are used as membership functions [Herrera et al., 2011a]. Therefore, in the RBFNN approach, equation (5.2) represents the output function of the RBF neural network, where $\mu^i(\vec{x}_k) = \prod_{j=1}^{n} \mu^i_j(x_j)$ are the activations of the hidden neurons and Q^i are their output weights.

It is possible to use a common radius σ for all the MFs (also called *clusters* or *nodes* in the context of RBFNNs) or a different σ^i for each node i. In the first case, the optimization of the neuro-fuzzy model is easier, since the number of non-linear parameters is smaller. On the other hand, TSK models usually use different radius for each fuzzy set in each variable and node. Thus, the equivalent RBFNN, which could be called *multiple-radii RBF neural network*, has a higher training computational cost.

In general, the rule consequents Q^i have polynomial form, i.e.,

$$Q^i \equiv Y^i(\vec{x}) = a^i + (\vec{b}^i)^T \vec{x} + ..., \tag{5.4}$$

where a^i and $\vec{b}^i$ are the polynomial coefficients of the i-th rule consequent. A TSK model is said to be of order $\mathcal{O}$ when it uses polynomials of order $\mathcal{O}$ in its rule consequents, with $\mathcal{O}$ being an integer, generally 0 [Pomares et al., 2004], 1 [Jang, 1993] or 2 [Herrera et al., 2005].

The TaSe-NF model used in this chapter is a modified neuro-fuzzy model whose starting point is a zero-order ($\mathcal{O} = 0$) TSK fuzzy system with Gaussian membership functions, which is equivalent to a RBF neural network model [Herrera et al., 2011b].

5.3.2 Description of the TaSe-NF model for control problems

According to the formulation given in equations (5.2) and (5.3) (see section 5.2), the output of a TSK fuzzy controller is a continuous and differentiable function with respect to the parameters defining its rules. Once these parameters are optimized, the TSK controller is expected to globally approximate the behavior of the true inverse function of the plant being controlled. However, for both grid-based partitioning and scatter partitioning, the system's global output at a given point will normally depend on the contribution of several rules. If rule overlapping is not controlled, this multiple dependence can occur at the same rule centers. This can cause that the optimization of the local output of each rule (i.e., its consequent) is affected by the nearby rules, inducing it to not properly reflect the specific behavior of the model in its area of influence. The main implication is that, from the point of view of the user, the meaning of each independent rule may not be clearly related to the output of the controller.

The TaSe-NF model was introduced in [Herrera et al., 2011b] to solve this problem. One of its main characteristics is that it reduces the problem of the curse of dimensionality by using a scatter partition of the input space, whilst maintaining the interpretability of the sub-models. It performs a global optimization of the system whilst obtaining the optimal sub-models without the need to perform a trade-off between one objective and the other. These characteristics provide a significant performance advantage with respect to the TaSe model [Herrera et al., 2005] and other approaches that use grid-based partitioning [Bikdash, 1999; Zhou and Gan, 2004].

The local modeling in TaSe-NF is achieved through the use of the Taylor series expansion of a function around a point, which describes the behavior of that function in the vicinity of such point [Klein, 1998]. To achieve the objectives mentioned above, TaSe-NF imposes two requirements in the structure of the neuro-fuzzy model [Herrera, 2007]:

- The degree of overlap of all the rules is forced to vanish at each rule center. This way, every point of the n-dimensional space $\vec{c}^{\,i}$, identified by the center of a rule i, is only affected by its respective rule in the global output function.

- The consequents should have the form of a Taylor Series of a function around a point, namely the respective rule center. Nevertheless, in online control problems, the system's structure should be kept as simple as possible. For this reason, constant consequents are used in OSENF-TaSe and this requirement does not represent any modification.

In the following subsections, we describe in detail the partitioning of the input space and its effect on the interpretability of the resulting fuzzy rules. The examples shown have been taken from [Herrera et al., 2011b] and [Herrera, 2007].

5.3.2.1 Partitioning of the input space

Under scatter partitioning, the degree of overlap amongst the local models cannot be directly controlled through the use of a specific type of membership function. The scattered distribution of the membership functions along the n-dimensional input space makes it necessary to perform a deeper modification of the structure of the fuzzy system. In the TaSe-NF model, the final rule activation is modified to satisfy the property of overlapping mentioned above. To do so, this activation depends on both the activation of the rest of the rules and on the relative position of its center $\vec{c}^{\,i}$ with respect to the rest of the rule centers. For the sake of simplicity, this modification is first presented for the simple one-dimensional case and then extended to the general n-dimensional case.

Let us consider as an example a one-dimensional input space with domain $[0, 1]$. We define two membership functions (and thus, two rules) centered at $c^1 = 0.2$ and $c^2 = 0.8$, with $\sigma = 0.3$, as depicted in Fig. 5.1(a) [Herrera et al., 2011b]. A slight overlap between the rules can be observed in this case. In order to comply with the condition of overlap, the domain of the first MF $\mu^1(x)$ is limited by the function $1 - \mu^2(x)$. This implies that when the firing of the second rule is maximum (i.e., $\mu^2(x) = 1$), the firing of the first rule is zero, i.e., $\mu^1(x) = 0$. Formally, the activation of the first rule is modulated by

$$1 - \mu'^{2/1}(x); \;\; \mu'^{2/1}(x) = \begin{cases} \mu^2(x) & \text{if } x < c^2 \\ 1 & \text{if } x \geq c^2 \end{cases} \tag{5.5}$$

and, similarly, the activation of the second rule $\mu^2(x)$ is modulated by

$$1 - \mu'^{1/2}(x); \;\; \mu'^{1/2}(x) = \begin{cases} \mu^1(x) & \text{if } x > c^1 \\ 1 & \text{if } x \leq c^1 \end{cases} \tag{5.6}$$

Therefore, the final firing of the rules (see Fig. 5.1(b)) is given by

$$\mu^{1*}(x) = \mu^1(x)\left(1 - \mu'^{2/1}(x)\right) \tag{5.7}$$

$$\mu^{2*}(x) = \mu^2(x)\left(1 - \mu'^{1/2}(x)\right) \tag{5.8}$$

which after normalization leads to (see Fig. 5.1(c))

$$\hat{\mu}^{1*}(x) = \frac{\mu^{1*}(x)}{\mu^{1*}(x) + \mu^{2*}(x)} \tag{5.9}$$

$$\hat{\mu}^{2*}(x) = \frac{\mu^{2*}(x)}{\mu^{1*}(x) + \mu^{2*}(x)} \tag{5.10}$$

Generalizing to the n-dimensional case, the activation of the i-th rule is given by [Herrera et al., 2011b]

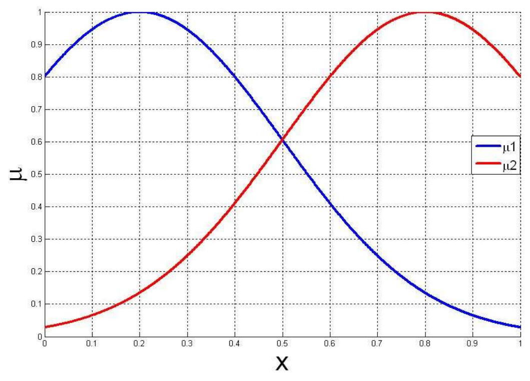
$$\mu^{i*}(\vec{x}) = \mu^i(\vec{x}) \prod_{\substack{j=1 \\ j \neq i}}^{N_r} (1 - \mu'^{j/i}(\vec{x})) \tag{5.11}$$

where

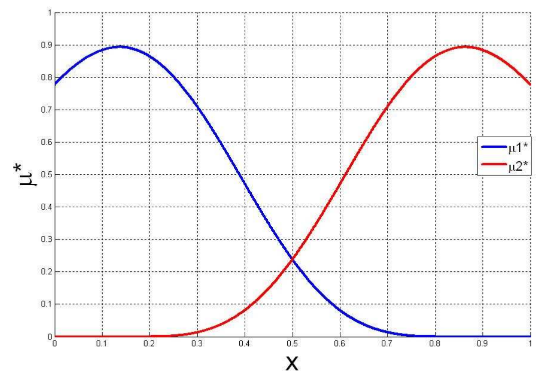
$$\mu'^{j/i}(\vec{x}) = \prod_{k=1}^{n} \begin{cases} 1 & \text{if } (x_k > c_k^j) \text{ and } (c_k^i < c_k^j) \\ 1 & \text{if } (x_k < c_k^j) \text{ and } (c_k^i > c_k^j) \\ \mu_k^j(x_k) & \text{otherwise} \end{cases} \tag{5.12}$$

Finally, with this new computation of the activation degrees of the rules, the

(a)



(b)



(c)

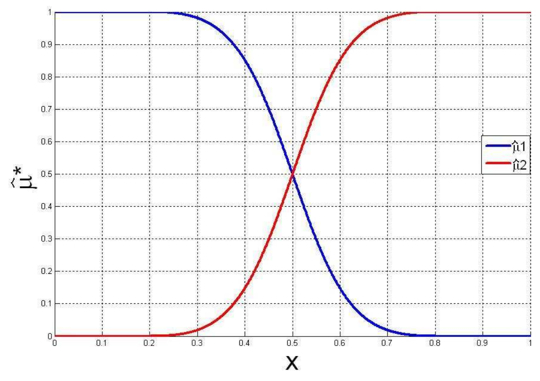


FIGURE 5.1: Control of the degree of overlap between local models in the TaSe-NF model for a one-dimensional example [Herrera et al., 2011b]. (a) Original μ^1 and μ^2 MFs. (b) Final rule activations μ^{1*} and μ^{2*}, computed according to (5.7) and (5.8). (c) Normalized final rule activations $\hat{\mu}^{1*}$ and $\hat{\mu}^{2*}$, computed according to (5.9) and (5.10)

output of the fuzzy controller can be defined as follows

$$u_k = \hat{G}_{TaSe-NF}(\vec{x}_k;\Phi_k) = \frac{\sum_{i=1}^{N_r} Q^i \cdot \mu^{i*}(\vec{x}_k)}{\sum_{i=1}^{N_r} \mu^{i*}(\vec{x}_k)} = \frac{\sum_{i=1}^{N_r}\left(\mu^i(\vec{x}_k)\prod_{\substack{j=1\\ j\neq i}}^{N_r}(1-\mu'^{j/i}(\vec{x}_k))\right)Q^i}{\sum_{i=1}^{N_r}\left(\mu^i(\vec{x}_k)\prod_{\substack{j=1\\ j\neq i}}^{N_r}(1-\mu'^{j/i}(\vec{x}_k))\right)} \tag{5.13}$$

5.3.2.2 Interpretability of the fuzzy rules

The use of (5.11), (5.12) and (5.13) preserves the properties of interpolation, continuity and differentiability of fuzzy systems, while guaranteeing that the global output of the system at each rule center is precisely the value of its corresponding consequent [Herrera, 2007]. To clarify this property, let us consider the one-dimensional example presented in the previous subsection. From the definitions given in equations (5.5)-(5.10), it is straightforward that the following properties are satisfied, due to the continuity of the Gaussian MFs and the stepwise functions presented:

$$\begin{array}{l} \hat{\mu}^{2*}(c^1) = 0;\ \hat{\mu}^{1*}(c^1) = 1 \Rightarrow \hat{G}_{TaSe-NF}(c^1) = Q^1 \\ \hat{\mu}^{1*}(c^2) = 0;\ \hat{\mu}^{2*}(c^2) = 1 \Rightarrow \hat{G}_{TaSe-NF}(c^2) = Q^2 \end{array} \tag{5.14}$$

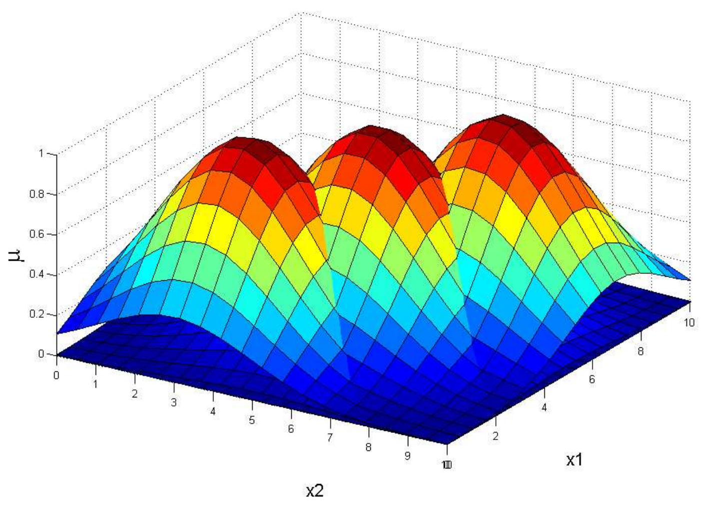
In the general n-dimensional case, the previous properties also hold and can be rewritten as

$$\hat{\mu}^{j*}(\vec{c}^i) = 0,\ \forall j \neq i;\ \hat{\mu}^{i*}(\vec{c}^i) = 1 \Rightarrow \hat{G}_{TaSe-NF}(\vec{c}^i) = Q^i \tag{5.15}$$

To illustrate the general case, Fig. 5.2 depicts a bidimensional example [Herrera et al., 2011b] with domain $[0, 10]$ and three Gaussian rules centered at $(3, 3)$, $(5, 5)$ and $(7, 7)$, respectively, with $\sigma = 2$ (see Fig. 5.2(a)). In Fig. 5.2(b), it is observed that thanks to the modified firing (5.11), the only rule with a non-zero activation at the position of each rule center is the respective rule.

Therefore, it can be seen that this model performs an intuitive partitioning of the input space whereby no rule has an effect on the other rule centers and the firing level is limited by the location of the remaining rules. This property

(a)



(b)

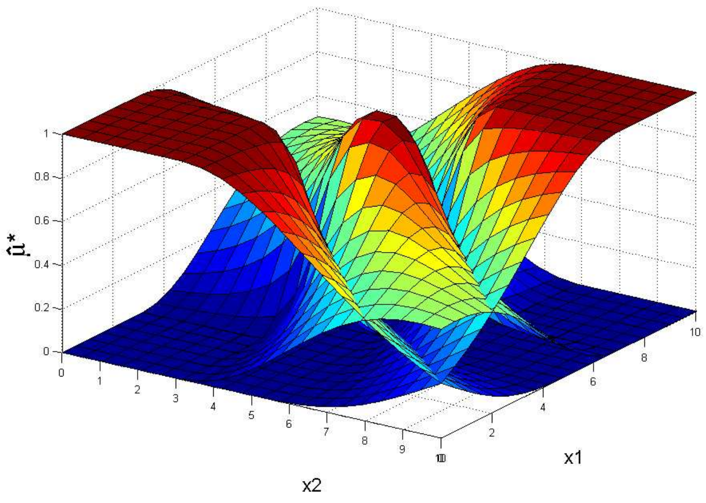


FIGURE 5.2: Modified rule activations for a bidimensional example. (a) Original rule activations with three Gaussian MFs. (a) Modified rule activations with three Gaussian MFs

guarantees that the model's global output at each rule center is precisely its corresponding consequent. Thus, the rules are more comprehensible from the human point of view, as they are directly related to the controller's output.

5.4 Architecture of the Online Self-Evolving Neuro-Fuzzy Controller based on the TaSe-NF model

The structure of the proposed OSENF-TaSe is depicted in Fig. 5.3. The learning mechanism (on the left) consists of two main parts: On the one hand, an online local learning method is applied to the rule consequents with the aim of reducing the current error at the plant's output. On the other hand, a structure self-evolution algorithm is applied to modify the structure of the controller by adding rules and globally updating their parameters, based on the analysis of the error surface at the controller's output. The switching from local learning to structure self-evolution occurs when the consequent adaptation by itself no longer improves the quality of the control. Note that these two levels of learning intend to provide a balance between exploration and exploitation in our approach.

On the right part of the figure, the TaSe-NF controller described in Section 5.3 is modified by the learning mechanism (dotted line). This TaSe-NF controller provides the control signal u that is applied to the plant. The OSENF-TaSe controller is the combination of the learning mechanism and the TaSe-NF controller.

It must be noted that most real applications may present limitations regarding the maximum number of rules that can be handled by the system (i.e., real-time requirements). In our approach, we consider two cases in which the addition of rules is stopped:

- Firstly, a threshold for the tracking error is defined. Whenever the error at the plant's output is below this value, the system is considered to be under control and no rules need to be added.
- However, obtaining a very accurate approximation of the plant's inverse function may require a large number of rules. Although the use of a scatter partition of the input space helps to tackle this problem, an upper bound

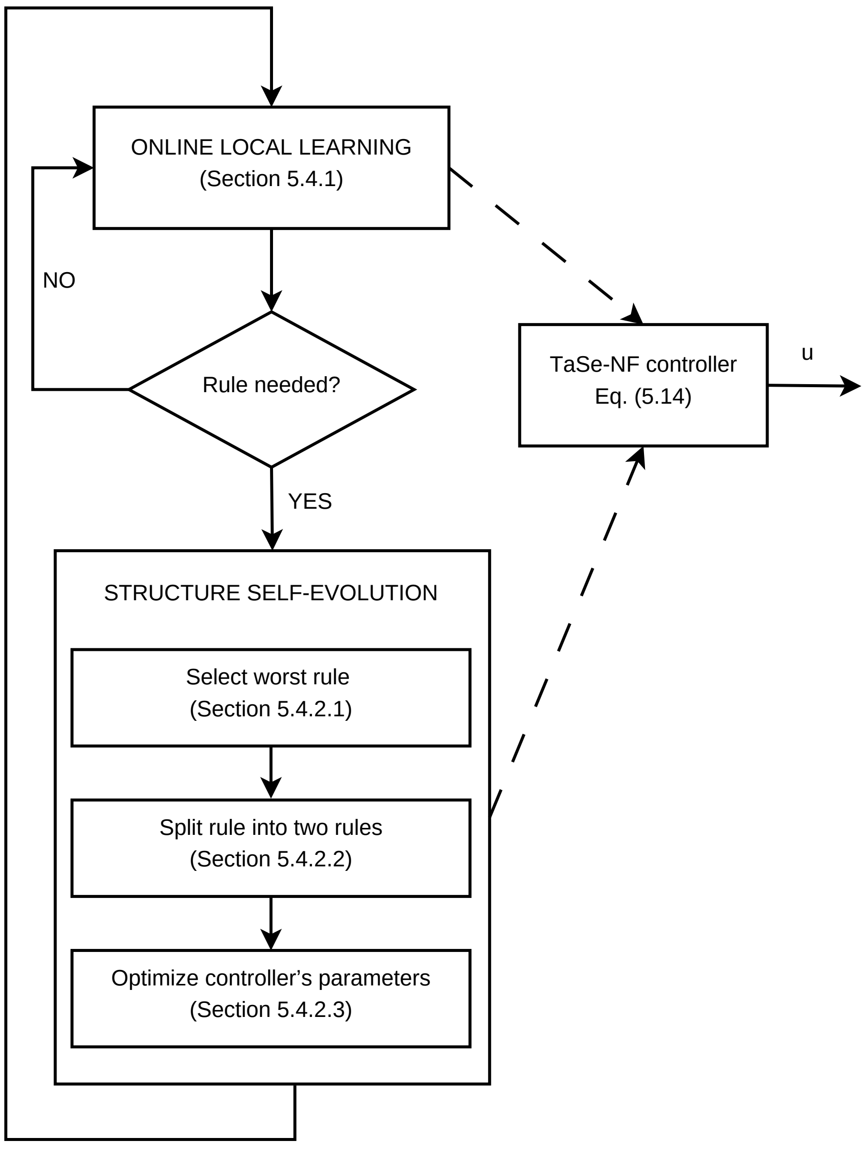


FIGURE 5.3: Structure of the OSENF-TaSe controller

for the maximum number of rules can also be defined. Note that, once this upper bound is reached, the existing rules continue being adapted. This way, the controller continues learning how to respond to the different control needs that may appear in different moments.

In the following sections, the elements that compose the learning mechanism are described in detail. Section 5.4.1 is devoted to the online local learning of the rule consequents, whilst the structure self-evolution algorithm is presented in section 5.4.2.

5.4.1 Phase 1: Online local learning of the rule consequents

The method for the online learning of the rule consequents presented in section 4.3.1 is independent on the type of MFs or the distribution of the rule base used in the fuzzy system [Pomares, 2000]. Therefore, it can be applied to learn the consequents in a FLS with scatter partitioning of the input space and, in particular, in a system based on the TaSe-NF model. Additionally, this learning method is both simple (intuitive) and computationally cheap, which makes it very suitable for online control. Thus, in OSENF-TaSe we use the same consequent learning method as in OSEFC. Since the procedure is completely described in section 4.3.1, we just present here a summary of the main idea lying behind it.

At every time step, the local objective of a controller is to keep the plant's output as close to the reference value as possible [Cara et al., 2011a]. With this objective in mind, the learning method combines the information about the sign of the monotonicity of the plant with respect to the control signal and the error at the plant's output to propose corrections to the rule consequents [Pomares et al., 2002b].

Hence, the adaptive method analyzes the current error at the plant's output and applies a penalty or a reward to the rules that are responsible for the error. Since not all the rules contribute in the same degree to the plant's output value, the modification applied to each rule consequent is proportional to its contribution, leading to the update law (4.5). For the sake of clarity, we reproduce again the modification to be applied to the consequent of the *i*-th fuzzy rule at instant *k*:

$$\Delta Q_k^i = C \cdot \mu_{k-1}^i(\vec{x}) \cdot e_k(\vec{x}) = C \cdot \mu_{k-1}^i(\vec{x}) \cdot (r_{k-1} - y_k) \tag{5.16}$$

where $\mu_{k-1}^i(\vec{x})$ is the activation degree of the *i*-th rule at the previous time step, when the rule was fired to obtain r_{k-1}, and C is a normalization constant. According to the discussion in section 4.3.1.2, the value of C is set offline as $|C| = \Delta u / \Delta r$, where Δu is the range of variation of the control signal and Δr is the range of possible reference values. It is worth reminding the reader that these two values are known beforehand, since the user of a control system has to know the operating ranges.

5.4.2 Phase 2: Structure self-evolution method

Although there are methods that evolve only the parameters of the controller while keeping a prefixed structure (i.e., number of rules) [Hsu and Lin, 2005; Pomares et al., 2004; Tewari and Macdonald, 2010], in order to operate without any previous knowledge, a mechanism to automatically develop the structure of the controller is also needed. This allows for the controller to start working with a very simple configuration, i.e., only one fuzzy rule automatically initialized. Then, as the control operation goes on, more rules are added as needed to meet the accuracy demands of the closed-loop system. OSENF-TaSe adds only one rule at a time, thus providing controllers with fewer rules than other approaches [Cara et al., 2011a, 2010c; Park et al., 2005]. It is important to note that by starting with only one rule and adding rules one by one as they are required, the necessity of applying rule pruning techniques is avoided, as more complex configurations are not considered until needed.

The only information available to tackle the problem of the online evolution of the controller's structure is the data obtained from the normal operation of the closed-loop system. Unlike other methods proposed to solve this problem [Angelov, 2004; Chen et al., 2008; Phan and Gale, 2008], OSENF-TaSe considers the entire operating region when modifying the controller's structure. Thus, the modifications are based on a set of historic data and not on a single point, which reduces the negative effect of outliers and provides higher robustness.

The principle behind the structure self-evolution method that we present in this chapter is the same as in the case of OSEFC (see section 4.3.2), that is, the fact that the very operation of the system provides input/output data about the true inverse function of the plant to be controlled [Pomares et al., 2004]. The interpretation of this idea is as follows: if the control signal u_k applied to the plant at time k produces the output value y_{k+1}, we know that if in the future the plant is again in the same state and the desired reference value is $r_{k'} = y_{k+1}$, the control signal that has to be applied to the plant is u_k (since u_k takes the plant output from y_k to y_{k+1}). Again, it has to be highlighted that this reasoning is valid as long as the relationship between the states, the control signal and the plant's output remains over time. If these I/O relationships are not reliable (e.g., in the presence of unbounded internal dynamics) or change rapidly in time, the data previously gathered are no longer valid and cannot be used to approximate the inverse function of the plant.

Hence, the pairs $(\vec{z}_k, u_k)$, where $\vec{z}_k$ is obtained from $\vec{x}$ by replacing r_k with y_{k+1}, represent I/O data concerning the true inverse function of the plant under control. To make use of this information, we use again the memory M described in section 4.3.2, which provides a uniform representation of the input space of the plant's inverse function in the regions actually reached by the operation.

In OSENF-TaSe, the refinement of the controller's structure is achieved by further splitting the rules that present an unsatisfactory performance. The idea is that the nonlinearities are higher in those areas with a larger approximation error, and therefore, more rules are needed in those regions [Phan and Gale, 2008]. Thus, the modification of the fuzzy structure involves three steps (see Fig. 5.3):

- First, a rule to be split has to be selected. This selection is based on the quality of the approximation obtained by each rule in its area of influence, which provides an idea of the strength of the nonlinearity in that area [Gao and Er, 2003].
- Once a rule is selected, it is split into two new ones. The objective is to locate more rules in those areas where the nonlinearity is high [Phan and Gale, 2008].

- Finally, the parameters of the resulting TaSe-NF controller are optimized. As discussed in section 4.3.2.3, when the structure of a controller is modified online, the new rules created can have a negative effect on the control performance until their parameters are tuned by the learning algorithm [Cara et al., 2011a]. To avoid this situation, we use the information stored in M to obtain optimized values for all the parameters [Pomares et al., 2002b].

The following subsections describe in detail each of these steps.

5.4.2.1 Selection of the rule to be split

To decide which area of the input space requires a new rule, the error in the approximation of the plant's inverse function is analyzed. In chapter 4, we discussed that adding rules directly to the points of maximum error leads to methods that are too sensitive to noise. A way to tackle this problem is to consider the error over the entire operating region, to detect problematic areas (instead of individual points).

The scatter partitioning of the input space used in OSENF-TaSe provides a set of clusters that are directly associated with the fuzzy rules. To detect the region in the input space that requires higher precision, we analyze the error related to each of these clusters and select the one with the largest error for further splitting. The algorithm for this process is as follows:

1. For each cluster C^i determined by the antecedent part of the i-th rule $\Re^i$, $[\mu_1^i, ..., \mu_n^i]^T$:

   (a) Select the points belonging to the cluster C^i. These are the points for which the rule with the maximum activation degree is $\Re^i$. Formally,

   $$X_{C^i} = \left\{ \vec{z} \mid \mu^i(\vec{z}) > \mu^j(\vec{z}) \quad \forall\, j = 1, ..., N_r \text{ and } j \neq i \right\} \tag{5.17}$$

   where $\vec{z}$ represents the input data stored in the memory M and $\mu^i(\vec{z})$ is the activation degree of the i-th rule for that input, as given in (5.11).

(b) Compute the approximation error associated to the cluster C^i as

$$E(C^i) = \sum_{\vec{z} \in X_{C^i}} (u(\vec{z}) - \hat{u}(\vec{z}))^2 \tag{5.18}$$

where $\hat{u}(\vec{z})$ is the output produced by the current TaSe-NF controller for the input vector $\vec{z}$, and $u(\vec{z})$ is the output stored in the memory M for that input vector.

2. Select the cluster C^* with the largest error, i.e.,

$$C^* = \arg\max_i (E(C^i)) \tag{5.19}$$

With this procedure, the rule with the worst performance has been selected. In order to increase the accuracy of the approximation in its area of influence, this rule is split into two new ones.

5.4.2.2 Generation of the new rules

Once the cluster with the worst performance has been selected, it is replaced by two new clusters C^{1*} and C^{2*}, thereby adding one more fuzzy rule to the system. The centers of these two new clusters, namely $[\mu_1^{1*}, ..., \mu_n^{1*}]^T$ and $[\mu_1^{2*}, ..., \mu_n^{2*}]^T$, are obtained by applying the K-means algorithm [MacQueen, 1967] to the points in X_{C^*}. Note that the K-means algorithm[1] is only used to obtain the initial location of the centers of the new clusters, which will be later optimized.

The resulting TaSe-NF controller is initialized as follows:

- The rule $\mathfrak{R}^*$ associated to the cluster C^* is removed and two new rules $\mathfrak{R}^{1*}$ and $\mathfrak{R}^{2*}$ are added to the rule base. These new rules are defined as

$$\mathfrak{R}^{1*} : \text{IF } x_1 \text{ is } \mu_1^{1*} \text{ AND ... AND } x_n \text{ is } \mu_n^{1*} \text{ THEN } u = Q^{1*} \tag{5.20a}$$

$$\mathfrak{R}^{2*} : \text{IF } x_1 \text{ is } \mu_1^{2*} \text{ AND ... AND } x_n \text{ is } \mu_n^{2*} \text{ THEN } u = Q^{2*} \tag{5.20b}$$

[1]In this work, we have used the K-means implementation provided in the Matlab®Statistics Toolbox, version 6.1 (R2007b) [MATLAB, 2007b], with the default values in all the parameters, i.e., 20 iterations, euclidean distance, and random selection of the centroids (among the points in X_{C^*}).

- The membership functions that formed the antecedent part of rule $\mathfrak{R}^*$, i.e., $[\mu_1^*, ..., \mu_n^*]^T$, are replaced by the membership functions that define the two new rules, i.e., $[\mu_1^{1*}, ..., \mu_n^{1*}]^T$ and $[\mu_1^{2*}, ..., \mu_n^{2*}]^T$.

- The radius of all the membership functions is set to the mean of the minimum distance amongst the centers of the clusters. Note that, although it may be considered that using the same radius for all the MFs reduces the flexibility of the model, the normalization of the system's output made in (5.13) decreases the influence of this parameter.

- The consequents of the two new rules, namely Q^{1*} and Q^{2*}, take the values of the controller's output under its previous configuration at the points of maximum activation of the new rules, i.e., when the inputs are equal to the centers of the new MFs. Hence, according to (5.13), the new consequents are given by

$$Q^{1*} = \hat{G}_{TaSe-NF}(x^{1*}; \Phi_{k-1}) \tag{5.21a}$$

$$Q^{2*} = \hat{G}_{TaSe-NF}(x^{2*}; \Phi_{k-1}) \tag{5.21b}$$

  where $x^{j*} = [c_1^{j*}, ..., c_n^{j*}]$ represents the centers of the MFs in the antecedent part of the new rules and Φ_{k-1} represents the parameters of the controller before adding the new rule.

5.4.2.3 Optimization of the parameters of the new controller

Once an initial configuration of the parameters of the TaSe-NF controller is obtained, we use a local search procedure to find an optimal configuration of the parameters defining every rule (i.e., centers and radius for the antecedent MFs and values of the consequents) [Pomares et al., 2002b]. Note that using the same radius for all the MFs reduces the risk of falling on a local (non-global) minimum during this search, as the number of nonlinear parameters that have to be optimized is smaller. This is especially important due to the fact that the amount of I/O data available is reduced, as it is obtained from the normal operation of the closed-loop system.

Multiple methods can be found in the literature for this search process, e.g., gradient descent, Newton-Raphson, Levenberg-Marquardt, etc. Herein we have used the Levenberg-Marquardt (LM) algorithm [More, 1978] due to its efficiency and robustness [1].

It is important to highlight that the process of adding a new rule is performed in parallel with the normal control operation of the system, which does not stop at any point. This is especially important in connection with the way in which the structural changes take place in the proposed method. Obviously, in every learning/evolving method, there is always a time lapse between the moment in which it is decided to modify the controller's structure and the moment in which the change has been made effective and the new controller can start operating. In OSENF-TaSe, the decision of modifying the structure is based on the general evolution of the controller's performance over a period of time and not on the analysis of the properties of every single point. This means that the structural changes do not have to take place at any specific moment, i.e., the modification of the structure does not aim to accommodate the properties of any specific point to the control policy. As a consequence, the learning time is less critical.

For each change, all the information stored in the grid memory M until the current time instant is used. Since the controller's operation does not stop during the structure evolution phase, if the process is not finished before the next time step, new data points will be gathered in the meantime and stored for their use in future structural changes. Therefore, they will not be missed. Moreover, it is likely that these new points will belong to an already explored region of the input space, so their only effect will be reinforcing the information already stored in M. In the case that the new points belong to a new operating region, they will not be relevant enough as to generate a new rule in their area until enough points are received.

Nevertheless, in an online environment it is highly desirable to learn the new structure as fast as possible with respect to the controller's operating pace. For this reason, we have selected the LM algorithm with a small number of

[1]Specifically, we have used the implementation of the LM algorithm provided in the Matlab®Optimization Toolbox, version 3.1.2 (R2007b) [MATLAB, 2007a], with the default configuration for the parameters of the algorithm, i.e., $\lambda = 0.01$, all the weights set to 1, and 20 iterations.

iterations, instead of other more-intensive search techniques, such as evolutionary algorithms. This way, in the worst case, only a few new data may arrive during the structure modification process and they will have a reduced global relevance.

5.4.3 Comments on the stability of OSENF-TaSe

As for OSEFC, in OSENF-Tase we do not assume any knowledge about the mathematical model of the plant to be controlled. As a consequence, a rigorous mathematical formulation for the proof of stability is almost impossible. However, this issue can be approached with a discussion similar to the one presented in chapter 4 for OSEFC.

In the first place, the same method is applied for learning the rule consequents. As mentioned before, this method is independent on the type or distribution of the rules [Pomares, 2000]. Thus, the analysis presented in section 4.3.1.4 is still valid here.

Regarding the stability across the structural changes, the same idea can also be applied here. Thus, let us represent the approximation of the control policy achieved by the controller with N_r rules as $u(N_r) = \hat{Q}(N_r)\mu(N_r) + \epsilon(N_r)$, where $\hat{Q}(N_r)$ represents the vector of optimal consequents for this rule base and $\epsilon(N_r)$ is the approximation error. As previously commented, this controller is locally stable, in the sense that the tracking error will converge to a ball around the origin with radius d_1 [Nounou and Passino, 2004; Spooner et al., 1997].

Consider now that the rule with the worst performance in the rule base (i.e., $\Re^*$) is replaced by two new rules (namely $\Re^{1*}$ and $\Re^{2*}$) that are placed in a location that minimizes the approximation error for the I/O data available. According to the universal approximation property of fuzzy controllers, the approximation of the resulting controller will be better at least in the area of influence of the new rules. Once the consequents are adapted again, the best control policy that will be achieved by this new configuration is $u(N_r + 1) = \hat{Q}(N_r + 1)\mu(N_r + 1) + \epsilon(N_r + 1)$, where the approximation error $\epsilon(N_r + 1)$ is smaller than $\epsilon(N_r)$. This means that the tracking error will converge to a ball with radius $d_2 < d_1$. Hence, since the radii decrease, the tracking error will eventually converge to zero, as the operation time goes to infinity.

On the other hand, it has to be kept in mind that the proposed methodology is based on the I/O data provided by the normal operation of the plant. Thus, it is necessary that the data gathered truly represent the I/O relationships present in the inverse function of the plant. In those situations in which this happens, the above discussion suggests that the proposed method provides a reasonably stable controller, despite the impossibility of carrying out a rigorous mathematical analysis.

5.5 Experimentation and results

In this section, we present simulation results to provide a better insight of the operation of the OSENF-TaSe controller. In section 5.5.1, the OSENF-TaSe controller is applied to a set of synthetic plants with the objective of illustrating its main properties. In section 5.5.2, OSENF-TaSe is applied to a real-world system. The proposed method is also compared with a classic controller and its robustness against unexpected changes in the plant is studied.

5.5.1 Description of the basic features of OSENF-TaSe

In order to illustrate the main properties of the proposed methodology in a clear and understandable way, we present some examples based on the control of a set of synthetic plants. First, we show the benefits of the new method by analyzing the reduction of the number of rules and the improvement of the controller's interpretability in comparison to grid-partitioning and traditional scatter-partitioning methods. Then, we present two additional examples that present special characteristics: Firstly, we analyze a case in which the controller's sampling time T_c has to be chosen larger than the plant's sampling time T_p, in order for our methodology to obtain a valid controller. Finally, we conclude this subsection by applying the proposed approach to a plant with internal zero dynamics.

5.5.1.1 Reduction of the number of rules

The use of a scatter partitioning of the input space helps to reduce the number of rules in the fuzzy controller, thus tackling the problem of the curse of dimensionality [Herrera et al., 2011b]. In this section we aim to illustrate this feature by comparing OSENF-TaSe to OSEFC, which employs a grid partitioning of the input space (see chapter 4). To do so, we consider again the plant (4.55) [Cara et al., 2010c; Pomares et al., 2000], whose dynamics are given by the difference equation

$$y(k+1) = -0.075\sin(y(k)) + \frac{u_k + u_k^3}{4} \tag{5.22}$$

The OSENF-TaSe controller applied has the same structural elements as used in section 4.4.1.2 for OSEFC. Hence, both controllers have two input variables, namely the reference signal $r(k)$ and the plant's output $y(k)$, which take values in the range $[-1, 1]$. The control objective is to follow a step-like reference signal in the range $[-1, 1]$. Initially, there is only one rule in each controller, with its consequent set to zero. The range for the control signal u is $[-1.5, 1.5]$ and the parameter C is set to 1.5.

The true inverse function of the plant (5.22) is depicted in Fig. 5.4. Fig. 5.5 represents the approximation of the inverse function obtained by OSENF-TaSe with four rules. Fig. 5.6 shows the approximation achieved by the grid-based method OSEFC with 4 and 20 rules (Fig. 5.6(a) and Fig. 5.6(b), respectively); the approximations obtained by other configurations of MFs can also be found in Fig. 4.10. It is observed that OSENF-TaSe is able to obtain a very precise approximation with only four rules, whilst the grid method requires 5 times more rules to achieve a similar accuracy.

Table 5.2 shows the evolution of the mean square error for the different structures of both controllers. For each structure considered by the online methodologies, we show the number of parameters to be optimized and the MSE reached by the controller at the moment of the next structural change. In this table, it is also observed that OSENF-TaSe provides a better performance with a smaller number of rules than OSEFC. Although the number of parameters for a given number of rules is slightly higher for OSENF-TaSe, the results show that the tracking error obtained by this method is also smaller than the error obtained by OSEFC with a similar number of parameters. For instance, with the topology 4x2 in OSEFC (i.e., 8 rules and 14 parameters) the MSE is 0.029, whilst the

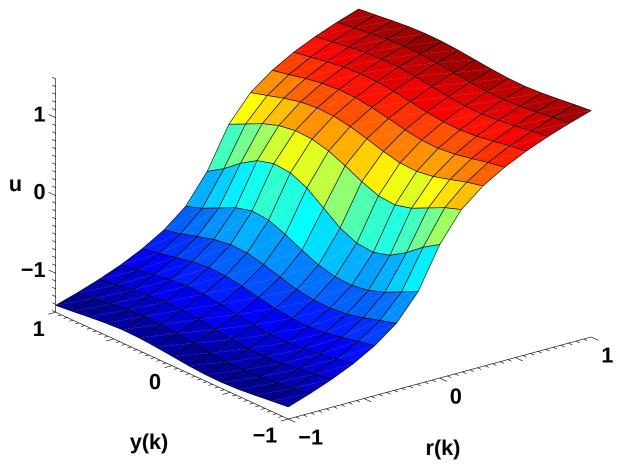


FIGURE 5.4: True inverse function of plant (5.22)

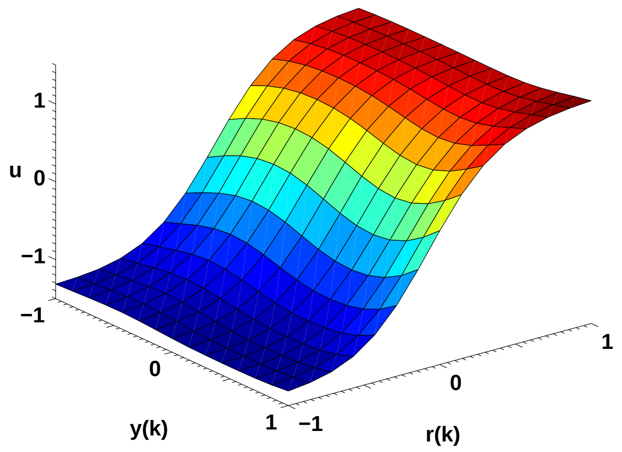


FIGURE 5.5: Approximation of the inverse function of plant (5.22) achieved by OSENF-TaSe with 4 rules

MSE obtained by OSENF-TaSe with 4 rules (i.e., 13 parameters) is 0.002. Moreover, OSEFC requires 28 rules (i.e., 39 parameters) to provide the same MSE as OSENF-TaSe.

5.5.1.2 Improvement of the interpretability

In order to illustrate the improvement of the interpretability of the fuzzy rules provided by OSENF-TaSe, in this subsection we analyze the controller obtained in the previous example and compare it with a traditional scatter-partitioning

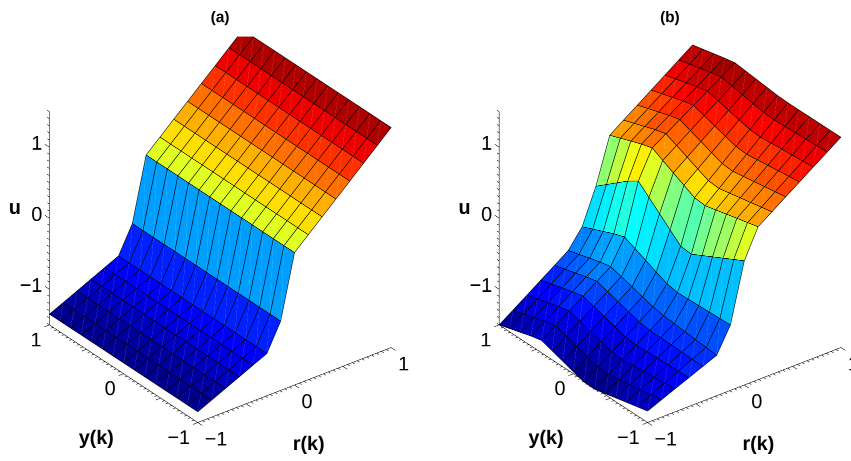


FIGURE 5.6: Approximation of the inverse function of plant (5.22) achieved by OSEFC. (a) With a structure of 4x1 MFs (4 rules). (b) With a structure of 5x4 MFs (20 rules)

TABLE 5.2: MSE with different number of rules for OSENF-TaSe and OSEFC

Rules	OSENF-TaSe		OSEFC		
	Parameters	MSE	Structure	Parameters	MSE
1	4	0.154	1x1	3	2.077
2	7	0.009	2x1	5	0.535
3	10	0.008	3x1	7	0.505
4	13	0.002	4x1	9	0.053
8			4x2	14	0.029
12			4x3	19	0.011
16			4x4	24	0.008
20			5x4	29	0.007
24			6x4	34	0.003
28			7x4	39	0.002

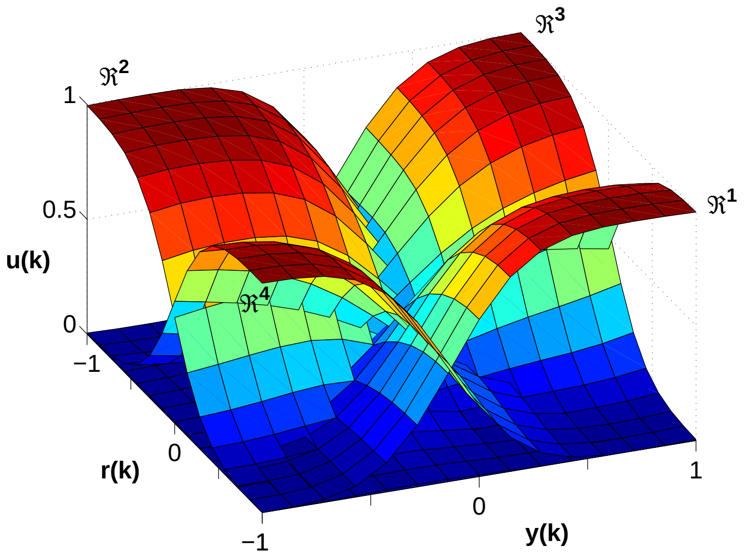


FIGURE 5.7: Rule activations for the four rules obtained by the OSENF-TaSe controller for plant (5.22). The rules are defined in (5.23)

TSK controller. The four rules obtained by OSENF-TaSe are given by

$$\mathfrak{R}^1 : \text{IF} \quad r(k) \text{ is} \quad \widehat{0.93} \quad \text{AND} \quad y(k) \text{ is} \quad \widehat{0.70} \quad \text{THEN} \quad u = 1.19 \tag{5.23a}$$

$$\mathfrak{R}^2 : \text{IF} \quad r(k) \text{ is} \quad \widehat{-0.93} \quad \text{AND} \quad y(k) \text{ is} \quad \widehat{-0.70} \quad \text{THEN} \quad u = -1.28 \tag{5.23b}$$

$$\mathfrak{R}^3 : \text{IF} \quad r(k) \text{ is} \quad \widehat{-0.92} \quad \text{AND} \quad y(k) \text{ is} \quad \widehat{0.98} \quad \text{THEN} \quad u = -1.25 \tag{5.23c}$$

$$\mathfrak{R}^4 : \text{IF} \quad r(k) \text{ is} \quad \widehat{0.93} \quad \text{AND} \quad y(k) \text{ is} \quad \widehat{-0.95} \quad \text{THEN} \quad u = 1.22 \tag{5.23d}$$

where $\widehat{c}$ represents the fuzzy set centered at point c, e.g., for the first premise in the antecedent part of rule $\mathfrak{R}^1$, the center of the fuzzy set $\widehat{0.93}$ is $c = 0.93$.

As commented in section 5.3.2.2, the partitioning of the input space used in the TaSe-NF model forces the overlapping of the rules to vanish at each rule center. This effect can be observed in Fig. 5.7, which depicts the rule activations for the four rules obtained by OSENF-TaSe.

The main consequence of this overlapping property is that it is guaranteed that each rule consequent represents the output of the plant's inverse function at the rule center, thereby providing a direct and clear interpretation to the rule. This property is illustrated in Fig. 5.8. This figure depicts the approximation of the plant's inverse function (blue surface) together with the activation of the

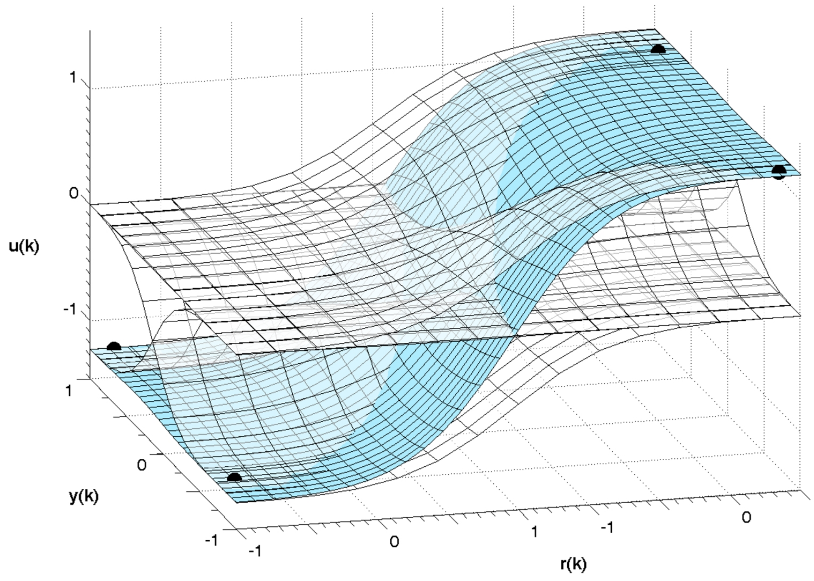


FIGURE 5.8: Approximation of the inverse function of the plant (5.22) achieved by the OSENF-TaSe controller (blue surface), together with the activation of the rules (transparent surfaces) and the rule consequents (black dots) located at the position of the corresponding centers

rules, which has been weighted by the corresponding rule consequent (transparent surfaces). The rule consequents are also depicted at the position of their respective centers (black dots). It can be observed that the output of the rules at their centers corresponds to the respective values of the plant's inverse function in those points. This figure also shows that the rule overlapping at the rule centers is zero. Thanks to this behavior, the rules obtained by OSENF-TaSe are easier to interpret than in other approaches.

To further illustrate this capability, we have applied our self-evolving methodology to a traditional scatter-partitioning TSK controller. Fig. 5.9 represents the approximated inverse function of the plant (blue surface), the weighted activation of the rules (transparent surfaces) and the rule consequents (black dots) obtained. In this case, there is a slight overlapping between the activations at the rule centers, causing the controller's output at these points to be different from the output of the plant's inverse function. Fig. 5.9 shows that the centers of the rules located in the lower values of $r(k)$ are below the real behavior of the controller at these points. The same happens to the centers of the other two rules, which are located above the approximated surface. As a consequence, the

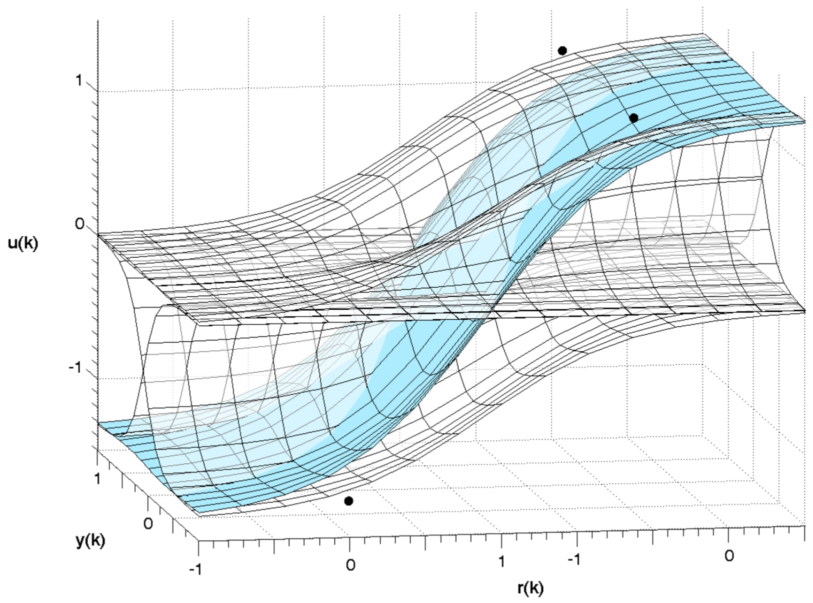


FIGURE 5.9: Approximation of the inverse function of the plant (5.22) achieved by a traditional TSK scatter partitioning controller (blue surface), together with the activation of the rules (transparent surfaces) and the rule consequents (black dots) located at the position of the corresponding centers

interpretation of the rules is not straightforward from the controller's actions.

5.5.1.3 Further examples

In this subsection, we present two additional examples of the application of OSENF-TaSe to control plants with special requirements. The first example considers a plant for which the controller's sampling time T_c is required to be larger than the plant's sampling time T_p. In the second example, a plant with zero dynamics is used.

The importance of selecting an appropriate sampling time for the controller has been commented in sections 4.2 and 5.2. To illustrate this issue, let us consider the simple plant given by

$$y_{k+1} = y_k - 2 \cdot u_{k-1} + u_k \tag{5.24}$$

It is easy to see that applying the proposed methodology to this plant with $T_c = T_p$ does not allow for the learning of the control policy, as the optimal

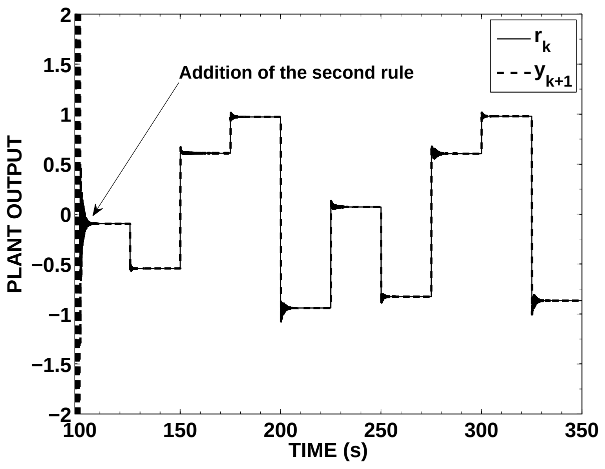


FIGURE 5.10: Reference tracking for plant (5.24) with $T_p = 0.01$ s and $T_c = 5 \cdot T_p = 0.05$ s

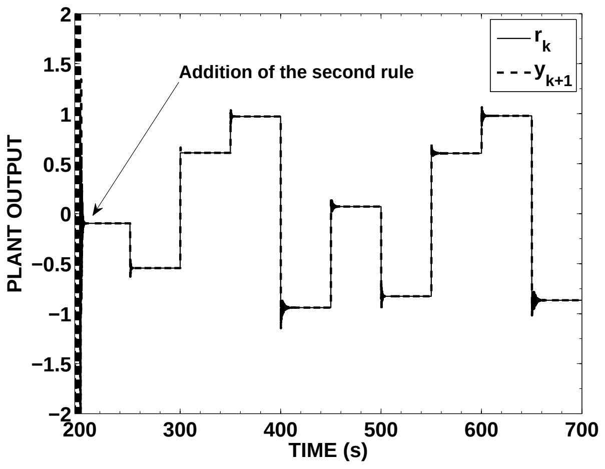


FIGURE 5.11: Reference tracking for plant (5.24) with $T_p = 0.01$ s and $T_c = 10 \cdot T_p = 0.10$ s

control policy would be unbounded in this case. Nonetheless, the problem can be tackled by using a larger sampling time for the controller. For instance, we consider two cases, namely, $T_c = 5 \cdot T_p$ and $T_c = 10 \cdot T_p$. Fig. 5.10 and Fig. 5.11 show the response obtained in these two cases by an OSENF-TaSe controller with two inputs, namely the error e_k and the control signal in the previous step u_{k-1}. In both figures it is possible to see how the controller stabilizes the plant's output after adding a second rule to the system.

Finally, we consider the control of plants with zero dynamics. As commented in section 5.4.2, the addition of new rules is based on the relationship between the plant's states, its output and the control signals. Plants with internal zero dynamics have states that are not observable from the plant's output. In this work, we assume a lack of knowledge about the internal dynamics of the plant and thus base our control methodology on the information measured during the normal operation of the system. This means that there is no way of being aware of the existence of these unbounded internal dynamics. For this reason, it is not possible to make a general statement about the validity of the approach for this type of systems, as it will depend on the effect of the unbounded internal states on the plant's output.

Nonetheless, we present here an example in which the control of a plant with zero dynamics is successfully tackled. Consider the plant

$$y_{k+1} = 1.6 \cdot y_k - 0.63 \cdot y_{k-1} + u_k - 2 \cdot u_{k-1} \tag{5.25}$$

Fig. 5.12 depicts the reference tracking achieved by a controller with three inputs (the error e_k, the variation of the plant's output $\dot{y}_k$, and the control signal in the previous step u_{k-1}) and two rules automatically created. The rules are uniformly distributed along the diagonal of the input space and the consequents are initialized to zero. Hence, the initial situation is equivalent to having an empty controller. It can be observed that OSENF-TaSe is able to keep this plant under control, despite of the presence of small overshoots.

5.5.2 Control of a real-world system

In this section, OSENF-TaSe is applied to the control of a real-world system, specifically, a mechanical suspension system [Ogata, 2001]. Firstly, we analyze the evolution of the control performance as new rules are added to the controller. Then, we study the method's robustness against changes in the dynamics of the plant and compare it with a classic controller that can be regarded as a "theoretical optimum" controller.

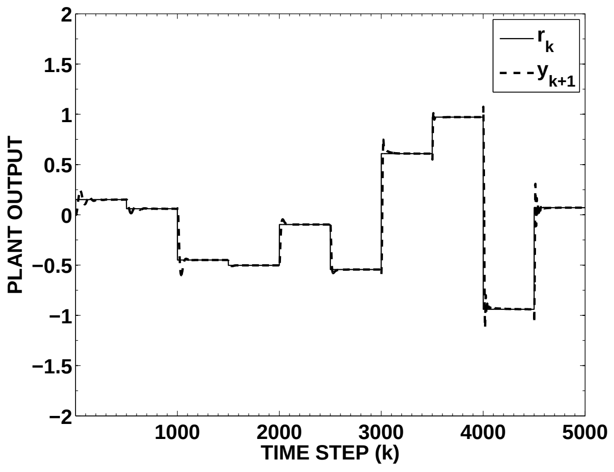


FIGURE 5.12: Reference tracking for plant (5.25)

5.5.2.1 Control of a mechanical suspension system

In this section, the OSENF-TaSe controller is applied to the industrial mechanical suspension system described in Appendix A.2 and depicted in Fig. 4.14. As for OSEFC, OSENF-TaSe does not make any use of the differential equations that govern the plant.

To control this system, an OSENF-TaSe controller with three inputs has been used. These inputs are the plant's output y_k, the tracking error e_k and the first derivative of the error $\dot{e}_k$. The input variables take values in the ranges $[-1, 1]$, $[-2, 2]$ and $[-0.6, 0.6]$, respectively. The actuator's range is limited to the values $[-1, 1]$, which represent the minimum and maximum voltage allowed by the physical system. The controller's sampling time is set to $T_c = 0.05\,\mathrm{s}$. The constant C for the consequents adaptation is then set to 1. Initially there is only one fuzzy rule with its consequent initialized to zero.

The control objective is to make the output variable follow a desired trajectory r_k, given by a randomly changing stepwise signal in the range $[-1, 1]$. To simulate the plant, the fourth-order Runge-Kutta method has been used. The total simulation time was 5000 seconds (83.33 minutes).

Fig. 5.13 depicts the evolution of the MSE through the entire simulation. The moments at which the topology has changed are indicated with vertical dotted lines. The MSE at the end of the simulation is zoomed in the right part of the

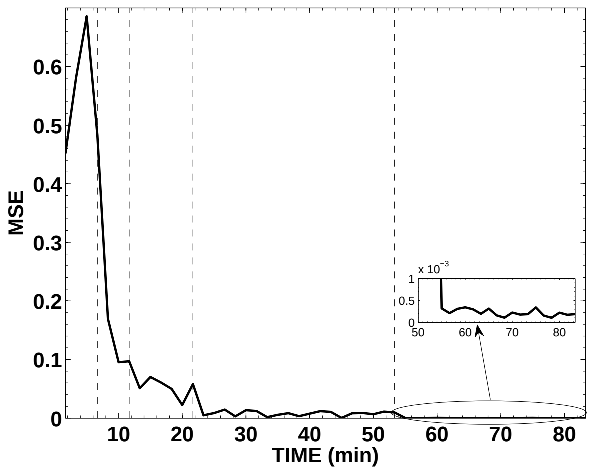


FIGURE 5.13: Evolution of the MSE when OSENF-TaSe is used to control the suspension system. The vertical dotted lines indicate the addition of a new rule

graphic. Four rules are added to the system during the simulation, thus obtaining a controller with five rules. It is observed that the addition of every new rule improves the performance of the controller, thereby reducing the MSE.

The reference tracking with the 5-rule controller is depicted in Fig. 5.14. We can observe that OSENF-TaSe provides a good control quality, despite the small number of rules used. It is worth noting that, since there are three inputs to the controller, a grid-based methodology would create $5^3 = 125$ rules in the same number of structural modifications.

5.5.2.2 Robustness against changes in the plant. Comparison with a classic controller

In order to further illustrate the capabilities of the proposed OSENF-TaSe, in this section we analyze its response when an unexpected change in the plant's dynamics occurs. Moreover, a comparison with a "theoretical optimum" controller is also presented. For this comparison we have used a classic controller based on full knowledge about the plant, designed by feedback linearization [Isidori, 1995].

In Fig. 5.15, the reference tracking for OSENF-TaSe (dashed line) and for the classic controller (dotted line) are depicted. In this figure it is observed that, even

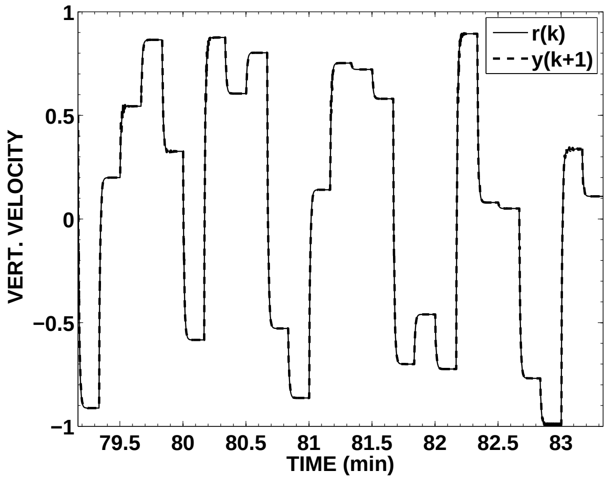


FIGURE 5.14: Reference tracking for the suspension system with a 5-rule OSENF-TaSe controller

though both controllers have a similar performance and provide a good control policy, the response of the evolving controller is slightly better, as shown in the zoomed area of the figure. In Fig. 5.16, the evolution of the MSE is depicted for both controllers. This figure confirms the previous observation, as it shows how the addition of the fifth rule to the OSENF-TaSe reduces the MSE below the values obtained by the classic controller. The importance of this result is highlighted by the fact that full knowledge about the equations describing the plant has been used to design the classic controller, whilst the evolving controller has not used any of this information.

It is important to note that classic methods, such as feedback linearization, require complete knowledge of the differential equations governing the plant. However, in many cases, these are unknown or very difficult to obtain. Moreover, the equations are a model of the plant, rather than its real and precise representation; hence, they are subject to deviations from the real behavior of the system due to modeling errors or simplifications. If the controller is later applied to the real plant, these deviations may cause an undesired response in the control process. In the case of OSENF-TaSe, the design of the controller is not based on a model of the plant, but on the real I/O data obtained from the very operation of the plant. This makes the method more robust and capable of handling modifications in the internal behavior of the plant.

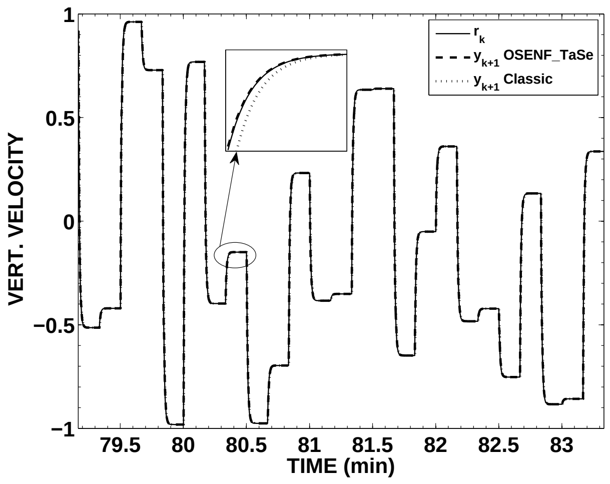


FIGURE 5.15: Comparison of the reference tracking for the suspension system obtained by OSENF-TaSe and by a classic controller

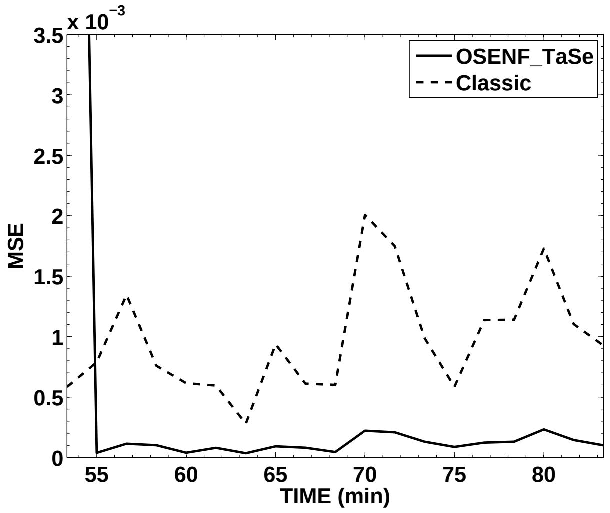


FIGURE 5.16: MSE evolution for the control of the suspension system. Comparison between OSENF-TaSe and a classic controller

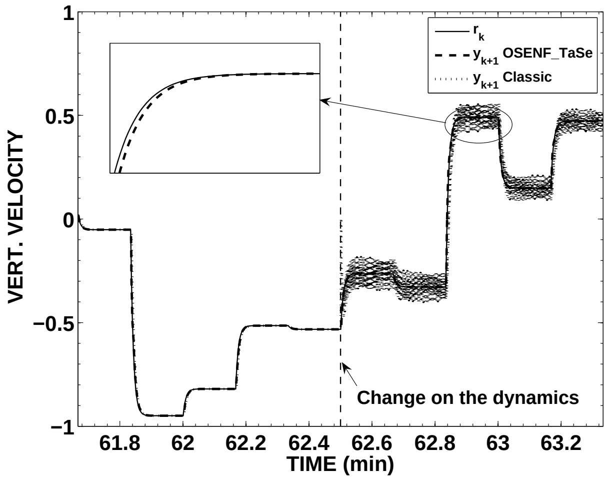


FIGURE 5.17: Reference tracking for the suspension system when there is a change in the plant's dynamics (i.e., M_1 is increased by 65%). Comparison between OSENF-TaSe and a classic controller

To illustrate this point, we have developed an experiment in which both the classic and the OSENF-TaSe controllers are applied to a plant that suffers an unexpected change in its internal dynamics. Specifically, the parameter M_1 in the suspension system is increased to different degrees during the execution. Fig. 5.17 depicts the response of the controllers when M_1 is increased by a 65% of its initial value. As it is observed, the classic controller is unable to cope with this change, causing large oscillations at the plant's output. However, as depicted in the zoomed area, the response of the OSENF-TaSe is almost unaffected by the change. Finally, Fig. 5.18 shows the response of the OSENF-TaSe when M_1 is increased by 500%. For such a large increase, the effect of the change is observable at the plant's output, which oscillates for some time. However, the self-evolving capabilities of OSENF-TaSe allow for the learning of the new behavior, thereby providing a good tracking response after some time. In this case, the response of the classic controller is not shown because the excessive growth of the oscillations at the plant's output caused computing errors.

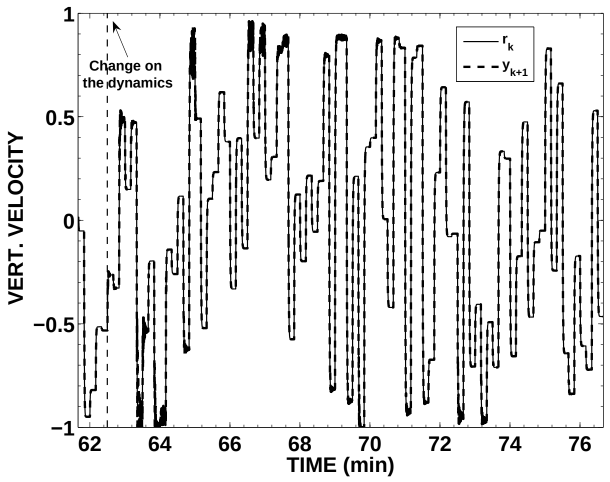


FIGURE 5.18: Reference tracking for the suspension system when there is a large change in the plant's dynamics (i.e., M_1 is increased by 500%)

5.6 Conclusions

In this chapter, a new online self-evolving neuro-fuzzy controller, called OSENF-TaSe, has been presented. This methodology can be seen as an extension of OSEFC (see chapter 4) that improves its interpretability and tackles the problem of the curse of dimensionality through to the use of the TaSe-NF model and its modified inference process.

As in the case of OSEFC, this new methodology assumes no prior knowledge about the plant to be controlled and develops the fuzzy structure online, while controlling the plant. Hence, the controller starts with a very simple structure consisting of only one rule automatically initialized and uses the I/O information obtained directly from the plant to add new rules and adapt the rule consequents online.

The learning mechanism of OSENF-TaSe is divided into two steps: On the one hand, the online local learning method takes care of the control on the short term by updating the rule consequents. Its objective is to guarantee a proper control at the present moment. On the other hand, the structure self-evolution process uses the information about the entire error surface to add new rules to the fuzzy controller. The refinement of the controller's structure is achieved by further splitting those rules that present a bad performance. In addition, a local search procedure is used to obtain an optimal configuration for the controller's parameters (i.e., centers and radius of the antecedent MFs and rule consequents).

OSENF-TaSe is based on the TaSe-NF model, which provides a scatter partitioning of the input space with locally interpretable models. As a consequence, two of the main characteristics of the OSENF-TaSe controller are the enhancement of the interpretability and the reduction of the number of rules with respect to the methods that use a grid partitioning of the input space. Hence, the features of OSENF-TaSe can be summarized as follows:

- OSENF-TaSe assumes no prior knowledge about the plant. Therefore, it starts operating with only one rule and learns both the structure and the parameters of the neuro-fuzzy controller while it performs the control.
- The learning method of OSENF-TaSe is based on the online local learning of the rule consequents and a structure self-evolution algorithm that adds rules to the system. The former aims to reduce the current error at the plant's output, whilst the latter distributes the rules based on the analysis of the entire error surface. This combination produces a good balance between stability and plasticity and between exploration and exploitation.
- OSENF-TaSe is robust under unexpected changes in the behavior of the plant.
- It preserves the interpretability of the local models by using the TaSe-NF architecture.
- It avoids the exponential growth of the number of fuzzy rules by using a scatter partitioning of the input space.

The three former properties are to some extent inherited from its predecessor OSEFC, whilst the two latter are derived from the use of the TaSe-NF model.

Simulation results have been used to illustrate the properties of the new OSENF-TaSe controller. On the one hand, it has been compared to OSEFC, which is based on a grid-like distribution of the fuzzy rules. As a result of this comparison, it has been observed that OSENF-TaSe is capable of obtaining better performance with a smaller number of rules than its grid-based counterpart.

Moreover, the comparison of OSENF-TaSe with a traditional scatter-partitioning fuzzy controller has shown the improvement on the interpretability of the rules.

On the other hand, OSENF-TaSe has been applied for controlling a real-world system. Its response has been compared with the response of a classic controller based on full knowledge about the plant. It has been shown that under normal conditions, OSENF-TaSe is capable of learning a control policy that is as good as the classic one, despite the fact that no prior knowledge about the plant is used. Moreover, the results show that OSENF-TaSe is able to cope with an unexpected change in the plant's dynamics to a further degree than its classic counterpart.

Chapter 6

Multi-objective optimization of type-1 and type-2 fuzzy systems with comparative purposes

Over the past years, type-2 fuzzy systems have become increasingly popular as a tool for uncertainty handling in a wide variety of real-world problems (e.g., control, modeling, ambient intelligence, etc.). However, their popularity is not without controversy. On the one hand, one of the main criticisms of singleton type-2 FLSs is that they outperform (the usually singleton-) type-1 FLSs because they use extra parameters in their fuzzy sets, thus making improved performance an obvious result. On the other hand, the counter-argument is that these performance differences are due to an essentially different way of handling the uncertainty between the (non-) singleton type-1 and the type-2 FLSs.

In this chapter, we aim to directly address this criticism by providing a common framework for comparing type-1 (both singleton and non-singleton) and singleton interval type-2 FLSs. With this objective in mind, we present a multi-objective evolutionary algorithm that optimizes the complexity and the accuracy of the three types of fuzzy systems considered. We compare the optimized type-1 and type-2 FLSs in the context of function approximation and use statistical tests to determine whether the differences in their performance are statistically significant or not. Finally, we analyze the response of type-1 and singleton interval type-2 FLCs for a sample control application.

6.1 Motivation and goals

6.1.1 Handling uncertainties: comparing type-1 and type-2 fuzzy systems

Type-2 FLSs were proposed to deal with uncertainties [Mendel, 2001]. As described in section 2.3, there are many sources of uncertainty facing the FLS in dynamic real-world unstructured environments and real-world applications. The most relevant ones are related to the inputs and outputs of the FLS, to their linguistic interpretation and to the presence of noise in the training data [Hagras, 2004].

Non-singleton (NS) type-1 FLSs (see section 2.2.1) were initially considered as the alternative to be used in the presence of uncertainties [Mendel, 1995]. However, considering the sources of uncertainty and mapping them to the different types of FLSs, it can be realized that the non-singleton type-1 FLS can only handle the first point relating to the uncertainties associated with the input measurements. The non-singleton type-1 FLS (as the singleton type-1 FLS) cannot handle the uncertainties associated with the output variables nor the linguistic uncertainties associated with the linguistic labels. An interval type-2 FLS however, through the employed interval type-2 fuzzy sets which include a FOU, can address and aim to handle all the different sources of uncertainty mentioned in section 2.3 [Cara et al., 2011b; Hagras, 2004].

Further, it is important to note that the non-singleton type-1 FLS handling of the input uncertainties "finishes" at the fuzzification stage, where the fuzzification returns a crisp membership value (like the singleton type-1 FLS) relating to the intersection of the type-1 fuzzy variable that represents the input measurement and the antecedent type-1 fuzzy sets. Thus, the uncertainty in the NS type-l FLS is not propagated further to affect the inference engine and defuzzification stage.

On the contrary, in the singleton type-2 FLS, the uncertainty propagates through the various blocks of the type-2 FLS to provide a decision that takes into consideration the various uncertainties in the inputs and outputs. Hence, it can be seen that the non-singleton type-1 FLS and the singleton type-2 FLS are conceptually different in the way they handle the uncertainties. Thus, giving the singleton type-1 FLS more degrees of freedom through the non-singleton fuzzifier will not make the NS type-1 FLS equivalent to the singleton type-2 FLS, as they handle uncertainties in a fundamentally different way.

It has been shown that singleton interval type-2 FLSs can handle the uncertainties and outperform their type-1 counterparts in applications with high uncertainty levels (e.g., mobile robot control [Figueroa et al., 2005; Hagras, 2004; Linda and Manic, 2011], video streaming control [Jammeh et al., 2009], noisy function approximation [Juang et al., 2010; Lin et al., 2009], etc.). However, one of the main criticisms of singleton interval type-2 FLSs is that the fact that they outperform singleton type-1 FLSs is solely based on their use of extra degrees of freedom (extra parameters) and that type-1 FLSs with a larger number of parameters may provide the same performance as interval type-2 FLSs.

In addition, most works on type-2 FLSs only compare their results with singleton type-1 FLSs, but fail to consider non-singleton type-1 systems [Juang et al., 2010; Lin et al., 2009; Sepúlveda et al., 2007]. As part of this chapter, we aim to directly address and investigate this criticism. To do so, we will perform a comparative study between optimized singleton type-1, non-singleton type-1 and singleton interval type-2 FLSs under the presence of noise in the training data.

6.1.2 Multi-objective optimization of type-2 fuzzy systems

As previously commented, the design of a FLS involves two steps: (i) selection of the structure of the system, i.e., the rules in the rule base, and (ii) determination of the right parameters (or database) , i.e., antecedents, consequents, linguistic variables, etc. Thus, manually designing and tuning a FLS is a difficult task, particularly as the number of MF parameters increases [Wagner and Hagras, 2007].

The automatic identification of the fuzzy system's parameters and/or structure from data samples can be seen as an optimization or search process [Alcalá et al., 2009]. Due to the high number of parameters to be defined in a FLS (especially in the case of type-2 FLSs), the size of the search space is rather big, thereby requiring powerful search methods capable of managing large dimensional spaces. At present, evolutionary algorithms (EAs) [Goldberg, 1989] are

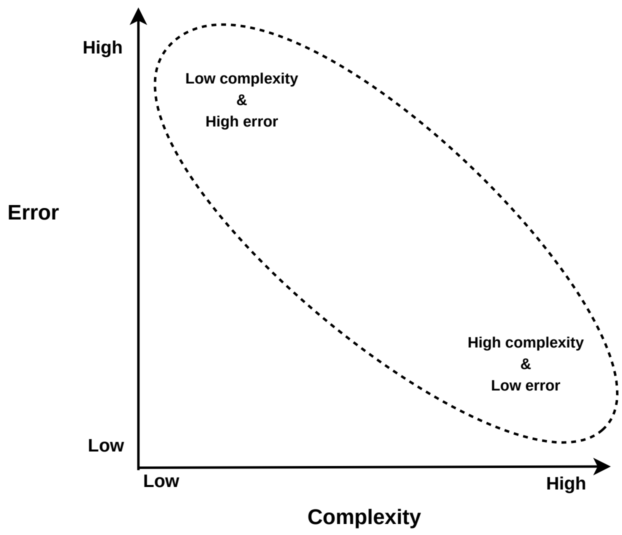


FIGURE 6.1: Trade-off between the complexity (interpretability) and the error (accuracy) of a rule base [Ishibuchi, 2007]

deemed to be powerful and effective global search techniques for this type of problem [Herrera, 2008]. As a result, their use for the learning of fuzzy systems (usually called Genetic Fuzzy Systems, GFS) has been a popular research topic in the past several years [Cordón et al., 2004; Cordón, 2011; Herrera, 2008].

There are always two main requirements in the design of a FLS: accuracy and interpretability. The former refers to the fuzzy system's capability of performing its task (e.g., modeling, control, classification) with the highest precision, whereas the latter refers to its ability to express this behavior in an understandable way from the human point of view [Cordón, 2011]. Although both objectives are equally important and should be satisfied in a similar degree, this is generally not possible because they are in conflict, i.e., a system with a low number of rules has higher interpretability but lower accuracy, and vice versa (see Fig. 6.1) [Alcalá et al., 2009; González et al., 2006; Ishibuchi, 2007].

Traditionally, the research in the field of GFSs has concentrated its efforts in the improvement of the accuracy, whilst interpretability has been left as a secondary aspect, often forgotten altogether [Cordón, 2011; Gacto et al., 2009]. Thus, multiple GFSs that use the accuracy as the single fitness measure have been proposed for both type-1 [Cordón, 2001; Cordón et al., 2004; Cordón, 2011; Hagras et al., 2004; Herrera, 2008] and type-2 FLSs and FLCs [Castillo et al., 2011;

Celikyilmaz and Turksen, 2008; Martinez et al., 2009; Wagner and Hagras, 2007].

Nonetheless, in the last years, a new tendency towards the balancing of interpretability and accuracy has appeared [Alcalá et al., 2006; Casillas et al., 2003; Cordón, 2011; Guillaume, 2001; Ishibuchi, 2007]. Although initially there were attempts at tackling this problem by using single-objective optimization that combined both aspects [Ishibuchi et al., 1995], the current trend is mainly oriented towards the use of multi-objective evolutionary algorithms (MOEAs) [Alcalá et al., 2009; Antonelli et al., 2010; Gacto et al., 2009; González et al., 2007; Guenounou et al., 2009; Zhang et al., 2011].

MOEAs [Coello et al., 2007; Deb, 2001] are evolutionary tools to tackle problems where two or more conflicting objectives have to be optimized at the same time, i.e., problems in which the improvement of one objective causes the worsening of the others. In this type of problem, known as multi-objective optimization problem (MOOP), the optimal solutions are usually sub-optimal for each objective in particular, but "acceptable" when all the objectives are taken into account [González et al., 2006]. For this reason, the goal of a MOOP is not to find an optimal solution, but a set of solutions that provide different compromises between the objectives. The population-based nature of EAs makes them a very suitable tool to solve this type of problem, as it allows them to provide a set of non-dominated (Pareto-optimal) solutions in a single run [Cordón, 2011].

The use of MOEAs for the optimization of fuzzy systems has widely spread in the last years and has become a hot topic in the GFS area [Cordón, 2011; Ishibuchi, 2007]. The first efforts in this field were directed to classification problems [Ishibuchi et al., 1997, 2001; Ishibuchi and Nojima, 2007; Pulkkinen and Koivisto, 2008]. Later, some attention has been put on function approximation problems as well. A MOEA with expert evolutionary operators is proposed in [González et al., 2006] to learn both the rule base and the parameters of the FLS. This work presents a set of genetic operators that aim to accelerate the convergence of the algorithm by avoiding the generation of offspring that are worse than their parents. In [Guenounou et al., 2009], a MOEA is combined with a backpropagation learning method in a two-step procedure that first learns the parameters of the FLS by backpropagation and then uses the MOEA to fine-tune the parameters and optimize the number of rules. A classic tuning of the MF parameters and a rule selection method are combined in [Gacto et al., 2009] with the objective of focusing the search on the most accurate part of the Pareto front. Finally, in [Alcalá et al., 2009], a method that learns both the parameters and the rule base is presented. In this approach, the search space is reduced by considering only displacements of the linguistic labels (i.e., the MF parameters) from an initial point. This way, only one parameter is needed instead of three (for triangular MFs), but the flexibility of the model is also reduced.

On the other hand, the automatic design of type-2 FLSs is nowadays an open research topic [Juang et al., 2010]. Traditionally, most learning methods for type-2 FLSs are based on their hybridization with neural networks and the application of gradient-descent methods [Jeng et al., 2009; Juang and Tsao, 2008; Mendel, 2004; Uncu and Turksen, 2007; Wang et al., 2004]. Different learning techniques can be found in [Biglarbegian et al., 2011; Juang et al., 2010]. In [Juang et al., 2010], a structure and parameter learning method was proposed for noisy regression problems. Online clustering was used for rule generation and a two-phase linear support vector regression method was used for parameter optimization. Finally, in [Biglarbegian et al., 2011], a method for the design of robust type-2 FLSs is presented. This method starts with a type-1 FLS obtained by means of some well known method (e.g., ANFIS [Jang, 1993]), then uses a genetic algorithm to obtain the FOUs, and finally checks the robustness through a constraint optimization problem.

As mentioned before, EAs have been used for learning type-2 FLSs [Castillo et al., 2011; Celikyilmaz and Turksen, 2008; Martinez et al., 2009; Park et al., 2009; Wagner and Hagras, 2007]. However, to the best of our knowledge, MOEAs have not been yet applied to the optimization of the structure and parameters of type-2 FLSs.

In this chapter, we present a MOEA for the optimization of singleton type-1, non-singleton type-1, and singleton interval type-2 fuzzy systems for function approximation problems. With this approach we follow two main goals: (i) the development of a common methodology for the learning of the three types of FLSs from data samples, and (ii) the creation of a common framework for the comparison of type-1 and type-2 FLSs that allows us to address the aforementioned criticism. It is important to highlight that we only consider singleton interval type-2 FLSs and do not address the more complex cases of general type-2 FLSs or non-singleton interval type-2 FLSs. For the sake of simplicity, in the reminder of this chapter we may refer to the singleton interval type-2 FLS indistinctly as "type-2 FLS" or "interval type-2 FLS".

The rest of the chapter is organized as follows. Section 6.2 introduces multi-objective evolutionary algorithms as a tool to tackle optimization problems with conflicting objectives. In section 6.3, the proposed multi-objective evolutionary algorithm for the optimization of type-1 and type-2 fuzzy systems is described in detail, including a method for computing the optimal consequents for a given rule base (see section 6.3.2.5). Experimental results are gathered in section 6.4, which includes an analysis of the optimized fuzzy systems obtained by the MOEA for function approximation (section 6.4.1) and a comparison of type-1 and interval type-2 fuzzy controllers for a sample application (section 6.4.2). Finally, the conclusions of the chapter are summarized in section 6.5.

6.2 Multi-objective evolutionary algorithms

6.2.1 Evolutionary algorithms

The evolutionary algorithm paradigm is based on the use of probabilistic search algorithms inspired by certain aspects in the Darwinian theory of evolution [Spears et al., 1993]. The basic idea is to maintain a population of chromosomes or individuals (representing candidate solutions to the specific problem being solved) that evolves over time through a process of competition and controlled variation [Goldberg, 1989].

The general flowchart of an EA is depicted in Fig. 6.2. Its components are summarized as [Cordón et al., 2004; Espejo et al., 2010]:

- A group (*population*) of candidate or partial solutions (*chromosomes* or *individuals*).

- A generational inheritance method whose objective is to advance towards better chromosomes. *Genetic operators* are applied to the chromosomes

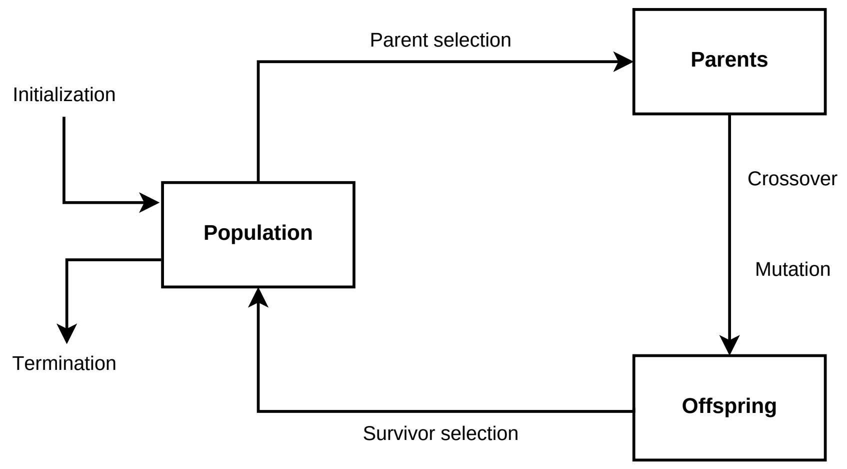


FIGURE 6.2: Flowchart of a general evolutionary algorithm (adapted from [Eiben and Smith, 2008])

of a population to give birth to a new population of chromosomes (the next *generation*). The main genetic operators are *crossover* and *mutation*. Crossover swaps a part of the genetic material of two individuals, whilst mutation randomly changes a small portion of the genetic material of one chromosome.

- A fitness-biased selection method. A *fitness function* is used in order to measure the quality of an individual. The better the fitness of an individual, the higher its probability of being selected to take part in the breeding of the next generation of chromosomes, thereby increasing the probability that its genetic material will survive throughout the evolutionary process.

EAs can be applied to the learning of FLSs at different complexity levels, ranging from parameter optimization to learning the rule base [Cordón et al., 2004]. In the latter case, the two most widely applied approaches in the literature are Pittsburgh [Smith, 1980] and Michigan [Holland and Reitman, 1977]. In the former, each chromosome represents an entire rule base, thus maintaining a population of candidate rule bases. On the contrary, in the Michigan approach each chromosome represents a single rule and the entire population forms the rule set.

6.2.2 Multi-objective evolutionary optimization. The Non-dominated Sorting Genetic Algorithm II

As explained before, a MOOP is an optimization problem in which several conflicting goals have to be optimized at the same time. Since the goals are in conflict, the improvement of one of them causes the worsening of the others [González et al., 2006]. In this type of problem, there is not a single solution, but a set of incomparable (non-dominated) solutions that represent different trade-offs between the objectives.

To clarify the concept of *dominance*, let us consider a MOOP with K conflicting objectives $f_1, f_2, ..., f_K$ that have to be simultaneously minimized (analogously maximized). A solution x_1 is said to dominate another solution x_2 (denoted by $x_1 \prec x_2$) if it is at least as good as x_2 in all the objectives and strictly better in at least one of them. Formally,

$$x_1 \prec x_2 \Leftrightarrow \forall i = 1, 2, ..., K : f_i(x_1) \leq f_i(x_2) \wedge \exists j = 1, 2, ..., K : f_j(x_1) < f_j(x_2).$$

A solution is *non-dominated* or *Pareto optimal* if there exists no other feasible solution that dominates it. The set of all the Pareto optimal solutions is known as the Pareto front [Zitzler et al., 2008].

There are two main reasons that make MOEAs well suited to solve multi-objective optimization problems. Firstly, the fact that they naturally handle a set of solutions (the population) makes them able to find Pareto-optimal solutions in a single run [Cordón, 2011]. Secondly, they are less susceptible to the shape or continuity of the Pareto front, e.g., they can deal with discontinuous and concave Pareto fronts [Gacto et al., 2009].

Several MOEAs have been proposed in the literature over the years [Fonseca and Fleming, 1993; Knowles and Corne, 2000; Srinivas and Deb, 1994], although it is generally agreed that the most representative ones are Strength Pareto Evolutionary Algorithm 2 (SPEA2) [Zitzler et al., 2002] and Nondominated Sorting Genetic Algorithm II (NSGA-II) [Deb et al., 2002] .

In this work, we use NSGA-II. Briefly, there are two main features that make it a high performance MOEA [Gacto et al., 2009]: fitness evaluation based on Pareto ranking and a crowding measure, and an elitist generation update.

The evaluation of the current population is carried out as follows: First, all the non-dominated solutions in the population are assigned a rank equal to 1 and are "removed" from the population. Then, from the remaining population, the non-dominated solutions are ranked as 2 and also removed. This procedure is iterated until all the solutions have been ranked. Solutions with a smaller rank are considered to be better. Hence, the Pareto front is formed by solutions with rank equal to 1.

To differentiate among the solutions with the same rank, a crowding measure is considered as an additional criterion. This crowding measure calculates the distance in the objective space between adjacent solutions with the same rank. Less crowded solutions with larger values of the crowding measure are considered better than more crowded solutions with smaller values of the crowding measure [Gacto et al., 2009].

Finally, to create the next population, an elitist policy is applied. First, the current and the offspring populations are merged together. Each solution in this combined population is evaluated using both the Pareto rank and the crowding measure. Then, the next generation is formed by the best valued chromosomes of the merged population. Elitism is implemented in NSGA-II in this manner [Gacto et al., 2009].

6.3 Multi-objective evolutionary algorithm for the optimization of the parameters and structure of type-1 and type-2 fuzzy systems

In this section, we present a MOEA for the optimization of the structure (rules) and parameters (antecedents, consequents) of type-1 and singleton interval type-2 fuzzy systems. With this approach we follow two main goals: Firstly, we aim to develop a common methodology for the learning of the three types of FLSs (namely singleton type-1, non-singleton type-1 and singleton interval type-2 FLSs) from data samples. Secondly, we intend to investigate the criticism on the performance of type-2 FLSs by comparing them to non-singleton type-1 FLSs. The use of the MOEA supplies a common framework for this analysis, thereby providing fuzzy systems with a similar number of parameters that are optimized under the same conditions.

The proposed MOEA includes a module for the computing of optimal consequents for a given rule base. This approach was initially proposed for type-1 FLSs [Pomares et al., 2000] and is extended to singleton interval type-2 FLSs with scalar consequents in this dissertation. The use of optimal consequents reduces the number of parameters to be optimized by the MOEA, thereby reducing the size of the search space.

6.3.1 Description of the fuzzy systems considered

The properties of the fuzzy systems used in this chapter are described in this subsection. As it is well known, there are multiple choices that can be taken regarding the structure of a FLS, such as the type of MFs, the organization of the rule base, the type of consequents (Mamdani or TSK), etc. In our case, these choices follow two different objectives: (i) Reducing the size of the search space for the MOEA, and (ii) ensuring that the number of parameters in the NS type-1 and in the singleton interval type-2 FLSs is the same. Thus, the following elements are chosen for the design of the fuzzy systems:

- A scatter partitioning of the input space is used for the rule base. As already mentioned, this type of partition helps to tackle the curse of dimensionality by reducing the number of rules. This way, the rules can be located in those areas of the input space that really contribute to minimize the output error [González et al., 2006].

- Zero-order TSK fuzzy systems are used. Thus, the consequents are represented by crisp values in the three cases. Although it may be thought that this reduces the flexibility of the type-2 FLS, it actually favors meeting the two objectives commented before. On the one hand, the type-2 FLS and the NS type-1 FLSs have the same number of parameters, since the NS system only incorporates mechanisms for handling the uncertainty at the system's input. On the other hand, the consequents can be optimally computed for each rule base [Pomares et al., 2000], thus making unnecessary to use the EA to optimize them.

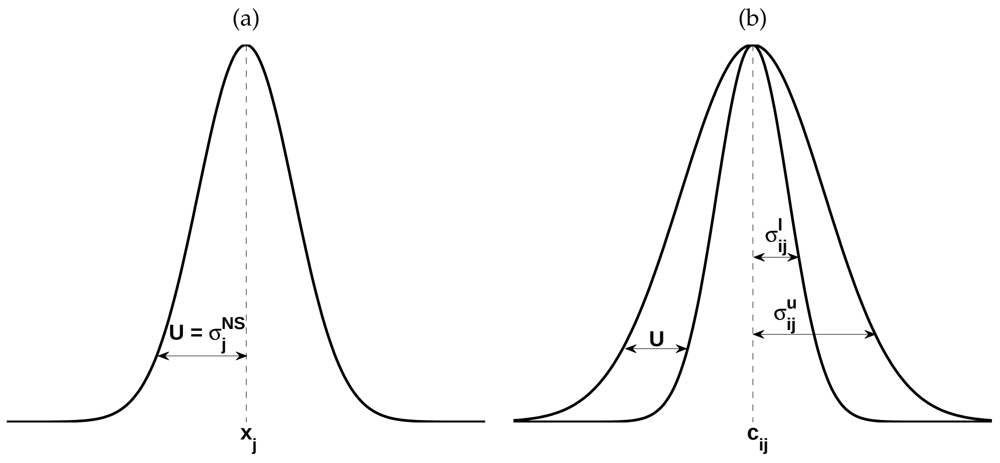


FIGURE 6.3: Gaussian functions used to handle the uncertainty in the input data. (a) Gaussian input function used in the non-singleton type-1 FLS. (b) Interval type-2 MF used in the type-2 FLS

- The inputs to the non-singleton FLS are fuzzy values instead of crisp numbers (see section 2.2.1). Gaussian MFs centered in the crisp value of the input are used to this purpose. These MFs are therefore defined by their standard deviation σ_j^{NS}, as depicted in Fig 6.3(a).

- Gaussian membership functions are also used for the rule antecedents. In the case of type-1 FLSs (both singleton and non-singleton), the MFs are defined by their center c_{ij} and their standard deviation σ_{ij}, where $i = 1, ..., N_r$ and $j = 1, ..., N$ represent the rule and the input variable, respectively. In the case of interval type-2 FLSs, the MFs have certain center and uncertain standard deviation. Thus, they are described by three parameters, namely $(c_{ij}, \sigma_{ij}^l, \sigma_{ij}^u)$, where σ_{ij}^l and σ_{ij}^u represent the lower and upper standard deviation, respectively (see Fig. 6.3(b)).

- Finally, the product is used as a t-norm and the weighted average is selected as the defuzzification method. In the case of the type-2 FLS, the center-of-sets method (see section 2.4.4) is used for the type-reduction process.

Since the number of parameters is an important issue in this study, we summarize now the parameters that define each of the FLSs considered. For the three systems, each rule is defined by N antecedents (given by the corresponding Gaussian MF) and one consequent. Thus, the number of parameters required to represent a system with N_r rules in each case is obtained as follows:

- The simplest case is the singleton type-1 FLS. Each antecedent MF is defined by two parameters, namely the center and standard deviation of the Gaussian function. Hence, the total number of parameters is $N_r(2N + 1)$. Since the consequents are optimally computed, only $2N_rN$ parameters are optimized by the MOEA.

- The rules in the NS type-1 FLS are defined by the same parameters as in the singleton case. Nonetheless, the standard deviation of the input MFs has to be defined. If a different σ_j^{NS} is used for each input variable, then the total number of parameters defining the FLS is $N+N_r(2N+1)$, of which $(2N_r+1)N$ are optimized by the MOEA. A way to reduce this number of parameters is using the same standard deviation for all the inputs, which leads to $1 + N_r(2N + 1)$ parameters (i.e., $1 + 2NN_r$ for the optimization procedure).

- Finally, the number of parameters for the singleton interval type-2 FLS depends on the definition of the FOUs. If a different FOU is used for each antecedent MF, then the number of parameters needed to define the system is $N_r(3N + 1)$, of which $3NN_r$ form the search space for the MOEA. Obviously, this represents a larger number of parameters than in the non-singleton type-1 FLS. Nonetheless, the FOU can be expressed as $\text{FOU}_{ij} = \sigma_{ij}^u - \sigma_{ij}^l$ (see Fig. 6.3(b)). Hence, if the same FOU is used for all the MFs, we can simply use all the σ_{ij}^l and a single value U to obtain the corresponding upper standard deviations as $\sigma_{ij}^u = \sigma_{ij}^l + U$. In this case, the number of parameters is reduced to $1 + N_r(2N + 1)$, of which $1 + 2NN_r$ are optimized by the MOEA. There is also the option of using the same FOU for all the MFs in the same input dimension, but different FOUs in each dimension. In this case, we can use the lower deviations σ_{ij}^l and N additional values $U_j\,(j = 1, ..., N)$ to obtain the upper deviations as $\sigma_{ij}^u = \sigma_{ij}^l + U_j$. This way, the system is defined by $N + N_r(2N + 1)$ parameters (i.e., $(2N_r + 1)N$ for the MOEA).

TABLE 6.1: Number of parameters in scatter-partitioning FLSs with Gaussian antecedent MFs and zero-order consequents

Type of FLS		Total	Optimized by the MOEA
Singleton type-1		$N_r(2N+1)$	$2NN_r$
Non-singleton type-1	Different U	$N+N_r(2N+1)$	$2NN_r+N$
	Same U	$1+N_r(2N+1)$	$2NN_r+1$
Singleton interval type-2	Different U	$N+N_r(2N+1)$	$2NN_r+N$
	Same U	$1+N_r(2N+1)$	$2NN_r+1$

Thus, it is observed that it is possible to configure the NS type-1 FLS and the singleton interval type-2 FLS in a way such that they have the same number of parameters. Note that the standard deviation σ_j^{NS} and the FOU are the mechanisms used by the the non-singleton type-1 and the type-2 FLSs to handle the uncertainty in the input data. In order to use a uniform notation for the description of the MOEA, we define the term *Uncertainty Factor* (UF) , which we denote by U. This parameter is used to represent the level of uncertainty considered in the system. In the case of the non-singleton type-1 FLS, U_j represents the standard deviation σ_j^{NS} of the input MF (see Fig. 6.3(a)), whereas for the type-2 FLS, U_j represents the FOU, i.e, the difference between the upper and the lower standard deviations, $U_j = \sigma_{ij}^u - \sigma_{ij}^l$, as explained before (see Fig. 6.3(b)). Table 6.1 summarizes the number of parameters in each type of system, indicating the total number of parameters and how many of them are actually optimized by the MOEA. It can be observed that with these configurations, the NS type-1 and the singleton interval type-2 FLSs only have 1 or N parameters more than the singleton type-1 FLS, depending on whether the same or different uncertainty factors are used in all the input dimensions.

6.3.2 Proposed multi-objective evolutionary algorithm

The following subsections describe in detail the elements that form the proposed multi-objective evolutionary approach. These components are the representation of chromosomes or solution encoding (section 6.3.2.1), the generation

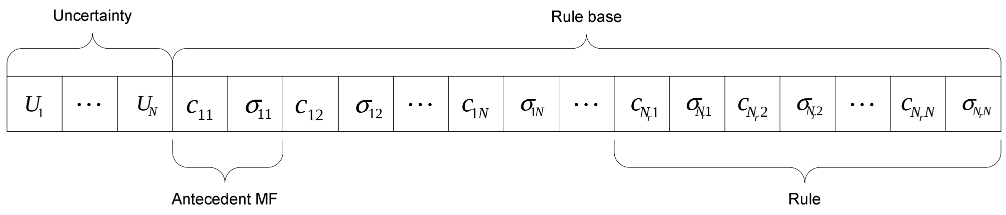

Uncertainty
Rule base
U_1 $\cdots$ U_N c_{11} σ_{11} c_{12} σ_{12} $\cdots$ c_{1N} σ_{1N} $\cdots$ c_{N_r1} σ_{N_r1} c_{N_r2} σ_{N_r2} $\cdots$ c_{N_rN} σ_{N_rN}
Antecedent MF
Rule

FIGURE 6.4: Chromosome representation

of the initial population (section 6.3.2.2), the evaluation of the fitness function, including the mentioned trade-off between accuracy and interpretability (section 6.3.2.3), and the genetic operators for crossover and mutation (section 6.3.2.4). Finally, the method for the optimal learning of the rule consequents is described in section 6.3.2.5.

6.3.2.1 Solution encoding

The chromosome representation used in the proposed MOEA is depicted in Fig. 6.4. The chromosome is composed of two parts: the uncertainty in the system and the rule base. The uncertainty is represented through 0, 1 or N uncertainty factors $U_1, ..., U_N$, corresponding to the cases of singleton type-1 FLS, the same UF in all the input dimensions, and a different UF in each input dimension, respectively. The rule base is represented as a sequence of rules, each of them described by the center and standard deviation of the antecedent MFs. Since the consequents are computed outside the genetic search process (see section 6.3.2.5), they are not included in the chromosome.

One of the objectives of the MOEA is to optimize the number of rules (complexity) of the FLS. Thus, the chromosome has a variable length, which has to be taken into account when applying the genetic operators. As in other works [Alcalá et al., 2009; González et al., 2006], we establish bounds for the maximum and minimum number of rules allowed in the systems, i.e., N_r^{max} and N_r^{min}, respectively. Note that in real applications, very complex solutions with a large number of rules are not desired [Alcalá et al., 2009]. Similarly, too-simple solutions are not expected to produce satisfactory results. Additionally, the presence of these two limits helps to reduce the size of the search space [González et al., 2006].

6.3.2.2 Initial population

It is common in the literature to find that the initial populations are randomly generated [Alcalá et al., 2009; Guenounou et al., 2009]. Nonetheless, it has been stated that MOEAs can obtain better solutions if some expert knowledge about the problem to be solved is included [Baeck et al., 1997]. In our case, we apply this idea by using heuristics to obtain an initial population formed by relatively good solutions, which can save many generations of evolutionary search [González et al., 2006]. Hence, the initial population is generated as follows:

- A chromosome with each possible number of rules in the range $[N_r^{min}, N_r^{max}]$ is generated, thus obtaining $N_r^{max} - N_r^{min} + 1$ chromosomes. The centers of the antecedent MFs are located using the K-means method [MacQueen, 1967] with the training data, whereas the standard deviations are selected by applying the K-nearest neighbors (KNN) algorithm [Cover and Hart, 1967] with $k = 2$ to the centers previously obtained [González et al., 2006].

- If necessary, the rest of the population is randomly generated, to introduce diversity.

- For all the chromosomes, the uncertainty factors are randomly generated in the range $[U_{min}, U_{max}]$. To establish these bounds, qualitative knowledge about the expected uncertainty can be used, e.g., the expected level of noise. It is worth noting that this range is only used for the generation of the initial values, as the actual uncertainty factors obtained by the MOEA can actually be outside this range. Thus, the specific range selected is not critical.

6.3.2.3 Fitness evaluation

The selection procedure in the MOEAs is based on the evaluation of the fitness function. As mentioned before, there are two conflicting objectives that we aim to optimize when learning FLSs: interpretability and accuracy. The former is measured as the number of rules in the rule base, N_r, whilst the latter is represented in terms of the normalized root-mean-square error (NRMSE) of the approximation, which is given by

$$\text{NRMSE} = \sqrt{\frac{\overline{e^2}}{\sigma_y^2}} = \sqrt{\frac{1}{N_s \cdot \sigma_y^2} \sum_{i=1}^{N_s} (F(x_i) - y_i)^2} \tag{6.1}$$

where $\overline{e^2}$ is the mean square error, σ_y^2 is the variance of the output data, N_s is the number of data samples, $F(x_i)$ is the output of the FLS for the i-th input vector, and y_i is the desired output for that input. The NRMSE is used instead of the MSE because of its independence on the scale of the data, which makes it more suitable for comparing results from problems with different ranges of values. Nonetheless, computationally speaking, it is more efficient to simply compute the sum of square errors $\text{SSE} = \sum_{i=1}^{N_s}(F(x_i) - y_i)^2$, as it avoids computing the square root. Thus, internally, we have used this SSE.

Hence, the following two-objective problem is considered:

$$\text{Minimize NRMSE}(c_i) \text{ and } N_r(c_i)$$

where c_i represents the i-th chromosome in the population.

6.3.2.4 Genetic operators

In evolutionary computation, it is important to find a proper trade-off between the exploration of the search space and the exploitation of the promising areas already found [Herrera et al., 2003]. Our selection of the two genetic operators, crossover and mutation, aims to find a good balance of this two aspects.

Crossover operator Traditionally, crossover operators combine genetic material of two parents to produce two offspring. However, in the NSGA-II implementation used in this dissertation only one offspring is generated from two parents. We use a hybrid crossover operator that combines the classical one-point operator [Goldberg, 1989] with the BLX-α operator [Herrera et al., 2003] with $\alpha = 0.5$.

We consider that an even number of parents has been selected using a suitable selection function (e.g., tournament). Then, the crossover process is performed as follows:

1. Randomly select an even amount of parents for single-point crossover, with probability P_{sp}.
2. Apply the single-point operator at rule level to each pair of parents C_1, C_2 in the selected group (see Fig. 6.5):
   (a) Select a cutting point $c \in [N_r^{min}, n]$, where $n = \min(N_r(C_1), N_r(C_2))$.
   (b) The exchange of genetic material between parents is done at rule level, i.e., complete rules are exchanged. Thus, the cutting gene is computed as $g = N_U + 2Nc$, where $N_U \in \{0, 1, N\}$ is the number of UFs used in the FLS.
   (c) The offspring takes the first g genes from parent C_1 (including the values of the uncertainty factors) and the remaining ones from parent C_2.
3. Apply the BLX-α operator to the remaining parents (note that there is an even number of parents left after the application of the single-point operator). For each pair of parents $C_1 = (x_1, ..., x_{g_1})$ and $C_2 = (y_1, ..., y_{g_2})$ with $x_i, y_i \in [a_i, b_i]$, an offspring $Z = (z_1, ..., z_G)$ with $G = \max(g_1, g_2)$ is generated as follows:
   (a) For $i = 1, ..., \min(g_1, g_2)$, the i-th gene z_i is a random value uniformly generated in the range $[l_i, u_i]$, where

$$\begin{aligned} l_i &= \max(a_i, c_{min} - I) \\ u_i &= \min(b_i, c_{max} + I) \end{aligned} \tag{6.2}$$

   with

$$\begin{aligned} c_{min} &= \min(x_i, y_i) \\ c_{max} &= \max(x_i, y_i) \\ I &= (c_{max} - c_{min}) \cdot \alpha \end{aligned}$$

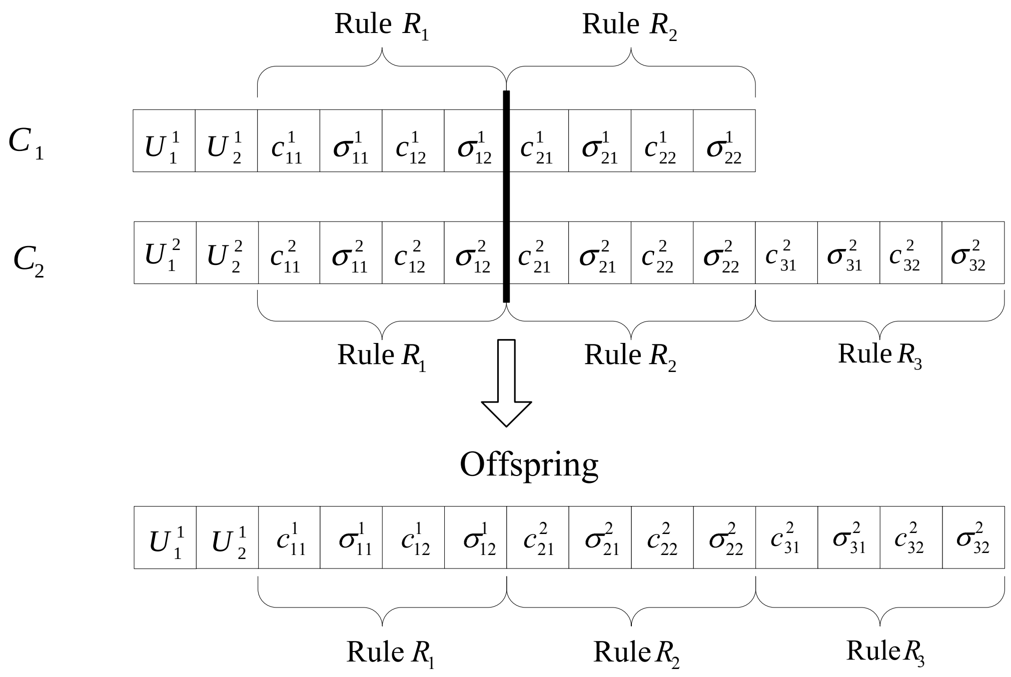

Rule R_1
Rule R_2
C_1
U_1^1 U_2^1 c_{11}^1 σ_{11}^1 c_{12}^1 σ_{12}^1 c_{21}^1 σ_{21}^1 c_{22}^1 σ_{22}^1
C_2
U_1^2 U_2^2 c_{11}^2 σ_{11}^2 c_{12}^2 σ_{12}^2 c_{21}^2 σ_{21}^2 c_{22}^2 σ_{22}^2 c_{31}^2 σ_{31}^2 c_{32}^2 σ_{32}^2
Rule R_1
Rule R_2
Rule R_3
Offspring
U_1^1 U_2^1 c_{11}^1 σ_{11}^1 c_{12}^1 σ_{12}^1 c_{21}^2 σ_{21}^2 c_{22}^2 σ_{22}^2 c_{31}^2 σ_{31}^2 c_{32}^2 σ_{32}^2
Rule R_1
Rule R_2
Rule R_3

FIGURE 6.5: Single-point crossover at rule level with cutting point $c = 1$ and cutting gene $g = 6$

(b) For $i = \{\min(g_1, g_2) + 1, ..., G\}$, the genes z_i take their values from the parent with the largest number of rules.

This operator has been included because it produces a balanced relationship between exploitation and exploration. The reason for this is that the probability of the offspring's gene lying inside the interval defined by the values of the parents' genes (i.e., exploitation) is the same as the probability of it lying outside this range (i.e., exploration) [Herrera et al., 2003]. The value $\alpha = 0.5$ has been selected because it is the most standard value in the literature [Eshelman and Schaffer, 1993; Herrera et al., 2002]. Fig. 6.6 shows how the BLX-α operator works at different moments of the evolutionary process. The grey areas represent the range in which the offspring's gene lies. It can be observed how this range shrinks as the EA evolves.

Mutation operator This operator creates new chromosomes by modifying the existing individuals. Its main objective is to improve the exploration of the

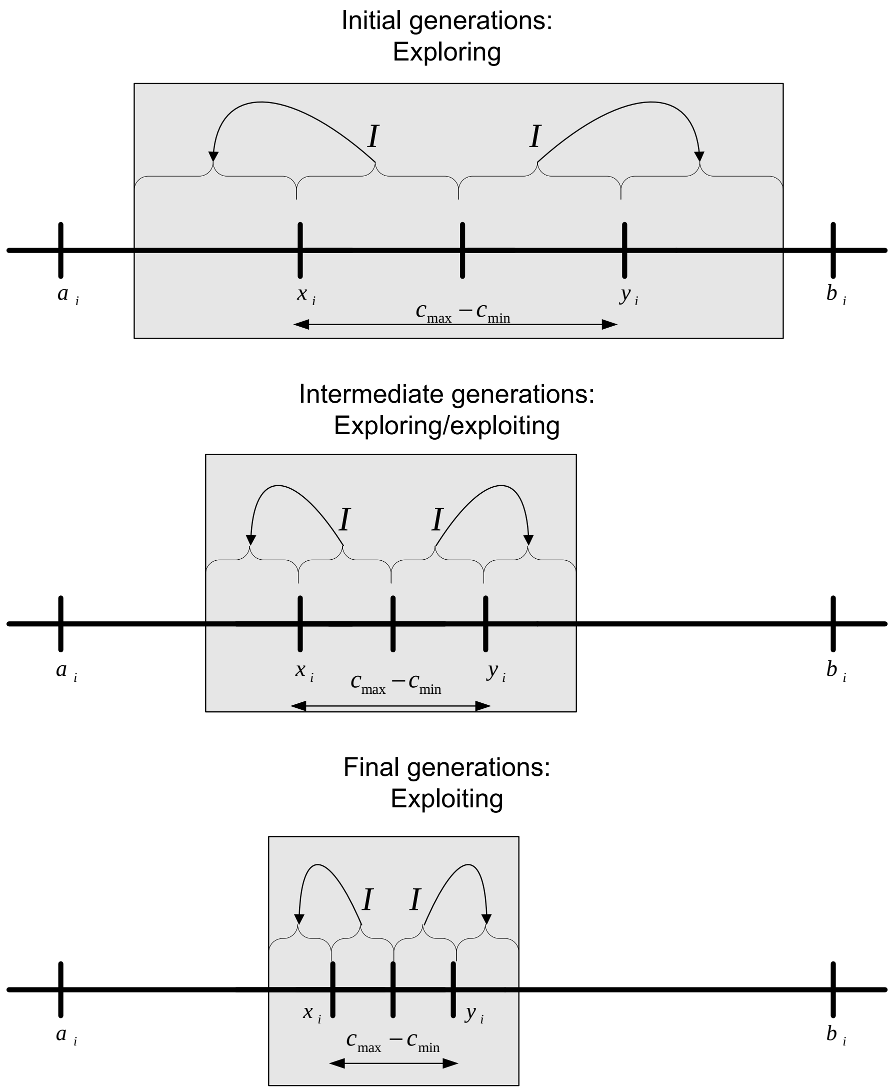


FIGURE 6.6: Evolution of the BLX-α ($\alpha = 0.5$) crossover operator during the evolutionary process (adapted from [Alcalá et al., 2009])

search space by introducing new genetic material. In our approach, we apply two mutation operators:

1. The first mutation operator modifies the number of rules in the chromosome by adding or subtracting r rules [Alcalá et al., 2009], with a probability P_{mod}. The value r is randomly generated in $[1, R_{max}]$, with the constraint that the number of rules in the resulting rule base must lie in the range $[N_r^{min}, N_r^{max}]$.

2. The second mutation operator simply applies random changes to some genes, with probability P_m [Guenounou et al., 2009]. Note that any chromosome can be modified, even if it has been modified by the first mutation operator.

The crossover operator combines existing solutions that have survived the selection procedure through the generations, thus helping to improve the exploitation of the search space. Regarding the mutation operators, the first one is useful to generate solutions with different levels of complexity; its use in combination with the second mutation operator favors the exploration of new areas of the search space.

6.3.2.5 Optimal consequents

For type-1 FLSs, given a rule base R and a set of K data samples $(\vec{x}_i, y_i)$, $i = 1, ..., K$, it is possible to compute the optimal consequents that approximate the unknown function represented by the data samples [Pomares et al., 2000], i.e., that minimize the sum of square errors

$$J(R, Q) = \sum_{i=1}^{K} (y_i - F(\vec{x}_i; R, Q))^2 \tag{6.3}$$

where Q represents the set of rule consequents, y_i is the desired output for the input vector $\vec{x}_i$ and $F(\vec{x}_i; R, Q)$ is the output of the fuzzy system formed by the rule base R and the consequents Q for that input vector.

In [Pomares, 2000], it is shown that the function F is continuous and differentiable with respect to the rule consequents and that this dependency is linear.

This property allows us to find the optimum values for the consequents by solving the overdetermined system of linear equations:

$$A \cdot Q = Y \equiv \begin{pmatrix} \alpha^1(\vec{x}_1) & \cdots & \alpha^{N_r}(\vec{x}_1) \\ \alpha^1(\vec{x}_2) & \cdots & \alpha^{N_r}(\vec{x}_2) \\ \vdots & \ddots & \vdots \\ \alpha^1(\vec{x}_K) & \cdots & \alpha^{N_r}(\vec{x}_K) \end{pmatrix} \cdot \begin{pmatrix} Q_1 \\ Q_2 \\ \vdots \\ Q_{N_r} \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_K \end{pmatrix} \tag{6.4}$$

where $Q_1, ..., Q_{N_r}$ are the rule consequents and $\alpha^j(\vec{x}_i)$ ($j = 1, ..., N_r$) is the normalized firing of the j-th rule for the input vector $\vec{x}_i$, given by

$$\alpha^j(\vec{x}_i) = \frac{\mu^j(\vec{x}_i)}{\sum_{k=1}^{N_r} \mu^k(\vec{x}_i)} = \frac{\prod_{a=1}^{N} \mu_a^j(x_a)}{\sum_{k=1}^{N_r} \prod_{a=1}^{N} \mu_a^j(x_a)} \tag{6.5}$$

If the determinant of the matrix $X = \left(A^T A\right)$ is not null, the solution of the system (6.4) is unique. This is a covariance-type matrix and the system can be easily solved by using some well known methods, such as the Cholesky algorithm [Pomares et al., 2000] or singular value decomposition (SVD) [Yen and Wang, 1996]. It is important to highlight that this reasoning is independent of the form and distribution of the membership functions, so it is valid for both grid-like and scatter partitioning of the input space.

In this section, we adapt this procedure to the case of interval type-2 FLSs with zero-order consequents. In order to do so, let us consider a fuzzy system with N_r rules in which the antecedents are given by interval type-2 MFs and the consequents are singleton values. According to (2.61), the output of such system is

$$y = \frac{y_l + y_r}{2} \tag{6.6}$$

where y_l and y_r represent the left and right ends of the output type-reduced set (2.49) [Hagras, 2004]. For the sake of clarity, the expressions used to compute these end points are reproduced again here. Thus, the left end of the interval is given by

$$y_l = \frac{\sum_{u=1}^{L} \overline{f}^u y_l^u + \sum_{v=L+1}^{N_r} \underline{f}^v y_l^v}{\sum_{u=1}^{L} \overline{f}^u + \sum_{v=L+1}^{N_r} \underline{f}^v} \tag{6.7}$$

and the right end is given by

$$y_r = \frac{\sum_{a=1}^{R} \underline{f}^a y_r^a + \sum_{b=R+1}^{N_r} \overline{f}^b y_r^b}{\sum_{a=1}^{R} \underline{f}^a + \sum_{b=R+1}^{N_r} \overline{f}^b} \tag{6.8}$$

where $\overline{f}^u$ and $\underline{f}^u$ (analogously, $\overline{f}^v$ and $\underline{f}^v$; $\overline{f}^a$ and $\underline{f}^a$; $\overline{f}^b$ and $\underline{f}^b$) are the upper and lower firing of the u-th rule (analogously the v-th, a-th, and b-th rules), respectively. The values L and R are computed as explained in section 2.4.4.2. The values y_l^u and y_r^u represent the left and right end of the centroid of the consequent of the u-th rule (see section 2.4.4.1). Nonetheless, since crisp values are used for the consequents, it holds that $y_l^u = y_r^u$. To simplify the notation, in the following derivation we simply use y^u to refer to the rule consequents.

We now define the normalized firing of the rules as

$$\underline{\alpha}_l^i = \frac{\underline{f}^i}{\sum_{u=1}^{L} \overline{f}^u + \sum_{v=L+1}^{N_r} \underline{f}^v} \tag{6.9}$$

$$\overline{\alpha}_l^i = \frac{\overline{f}^i}{\sum_{u=1}^{L} \overline{f}^u + \sum_{v=L+1}^{N_r} \underline{f}^v} \tag{6.10}$$

$$\underline{\alpha}_r^i = \frac{\underline{f}^i}{\sum_{a=1}^{R} \underline{f}^a + \sum_{b=R+1}^{N_r} \overline{f}^b} \tag{6.11}$$

$$\overline{\alpha}_r^i = \frac{\overline{f}^i}{\sum_{a=1}^{R} \underline{f}^a + \sum_{b=R+1}^{N_r} \overline{f}^b} \tag{6.12}$$

Note that the normalized lower (respectively upper) firing of the rules is different depending on whether y_l or y_r is being computed, since the summations in the denominators of (6.7) and (6.8) are different. Hence, we use the subscripts l and r to clarify this aspect.

Substituting the normalized firings in (6.7) and (6.8), the limits of the output type-reduced set (2.49) can be rewritten as

$$y_l = \sum_{u=1}^{L} \overline{\alpha}_l^u y^u + \sum_{v=L+1}^{N_r} \underline{\alpha}_l^v y^v \tag{6.13}$$

and

$$y_r = \sum_{a=1}^{R} \underline{\alpha}_r^a y^a + \sum_{b=R+1}^{N_r} \overline{\alpha}_r^b y^b \tag{6.14}$$

Thus, the output of our type-2 FLS can be expressed as

$$\begin{aligned} y &= \frac{1}{2}\left(\sum_{u=1}^{L} \overline{\alpha}_l^u y^u + \sum_{v=L+1}^{N_r} \underline{\alpha}_l^v y^v + \sum_{a=1}^{R} \underline{\alpha}_r^a y^a + \sum_{b=R+1}^{N_r} \overline{\alpha}_r^b y^b \right) \\ &= \frac{1}{2}\left[\sum_{i=1}^{q} y^i \left(\overline{\alpha}_l^i + \underline{\alpha}_r^i \right) + \sum_{j=q+1}^{p} y^j \Psi^j(L,R) + \sum_{k=p+1}^{N_r} y^k \left(\underline{\alpha}_l^k + \overline{\alpha}_r^k \right) \right] \end{aligned} \tag{6.15}$$

where $q = \min(L, R)$, $p = \max(L, R)$, and $\Psi^j(L, R)$ is given by

$$\Psi^j(L,R) = \begin{cases} \left(\underline{\alpha}_l^j + \underline{\alpha}_r^j \right) & \text{if } L < R \\ 0 & \text{if } L = R \\ \left(\overline{\alpha}_l^j + \overline{\alpha}_r^j \right) & \text{if } L > R \end{cases} \tag{6.16}$$

Finally, from (6.15) we obtain the following matrix of coefficients for the system of linear equations:

$$A_{T2} = \frac{1}{2} \begin{pmatrix} \left(\overline{\alpha}_l^1(\vec{x}_1) + \underline{\alpha}_r^1(\vec{x}_1)\right) & \cdots & \Psi^{q+1}(L,R)(\vec{x}_1) & \cdots & \left(\underline{\alpha}_l^{p+1}(\vec{x}_1) + \overline{\alpha}_r^{p+1}(\vec{x}_1)\right) & \cdots & \left(\underline{\alpha}_l^{N_r}(\vec{x}_1) + \overline{\alpha}_r^{N_r}(\vec{x}_1)\right) \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \left(\overline{\alpha}_l^1(\vec{x}_K) + \underline{\alpha}_r^1(\vec{x}_K)\right) & \cdots & \Psi^{q+1}(L,R)(\vec{x}_K) & \cdots & \left(\underline{\alpha}_l^{p+1}(\vec{x}_K) + \overline{\alpha}_r^{p+1}(\vec{x}_K)\right) & \cdots & \left(\underline{\alpha}_l^{N_r}(\vec{x}_K) + \overline{\alpha}_r^{N_r}(\vec{x}_K)\right) \end{pmatrix} \tag{6.17}$$

Therefore, it is possible to obtain the optimal consequents for a given rule base in an interval type-2 FLS with singleton consequents by applying the same method as in the case of type-1 FLSs, only with a different coefficient matrix. This is an important result, as it avoids the need to use the evolutionary approach to optimize the rule consequents, thus reducing the size of the search space.

6.4 Experimentation and results

As previously commented, one of the main criticisms of singleton interval type-2 FLSs is that they could outperform singleton type-1 FLSs as they use extra parameters/degrees of freedom in the FOU and that, consequently, a type-1 FLS with the same number of parameters may provide similar performance to an interval type-2 FLS. Additionally, it can be argued that the potential of type-1 FLSs has not been sufficiently exploited, as most applications of type-1 FLSs are based on their most simple form – the singleton type-1 FLS.

As part of this chapter, we aim to address these criticisms by working at two different levels:

- Firstly, we apply the MOEA described in section 6.3 to optimize the rule base and parameters of singleton type-1, non-singleton type-1 and singleton interval type-2 FLSs for function approximation problems. With these experiments we aim to provide an objective comparison between the three types of FLS considered. The use of the MOEA provides a framework in which the fuzzy systems not only have the same number of parameters, but also present configurations that have been obtained through the same optimization process, under the same conditions. Statistical tests are used to support the conclusions drawn from the comparative analysis.

- Secondly, we present a sample application consisting in the control of a nonlinear servo system [Cara et al., 2010c]. In this example, we compare an automatically tuned singleton type-1 FLC with a non-singleton type-1 FLC and a singleton interval type-2 FLC that are obtained from the former by applying the same uncertainty factors to the input MFs. The three controllers present the same structure (rule base), the same consequents, and a similar number of parameters (the same, in the case of the NS type-1 and the singleton interval type-2 FLCs). Hence, it is possible to carry out a direct comparison of the performance of the three types of FLCs (for this specific sample application) under different levels of input noise.

6.4.1 Optimized type-1 and type-2 fuzzy systems for function approximation

In this section, we aim to provide an objective comparison between type-1 and interval type-2 fuzzy systems. With this objective in mind, we have applied the MOEA described in section 6.3 to singleton type-1, non-singleton type-1 and singleton interval type-2 FLSs with different levels of noise in the training data. The parameters of the MOEA, the training data, and all the remaining assumptions are the same in the three cases. This way, the FLSs are obtained under equivalent conditions and can be expected to take advantage of their respective capabilities, thus providing good performances in all the cases. First, we describe the experimental setup used, and then, we present the results obtained.

6.4.1.1 Experimental setup

Our benchmark consists of ten function approximation problems taken from [Cherkassky et al., 1996] and [Rovatti and Guerrieri, 1996], that are commonly used in the literature [Echanobe et al., 2007; Florido et al., 2011; Guillén et al., 2009; Wang et al., 2010]. Table 6.2 shows the definition of the functions considered. For each problem, a training dataset formed by 400 data points is used. This dataset is obtained by defining a 20×20 grid in the input space and taking a random point in each zone delimited by that grid [Pomares et al., 2002a]. In order to facilitate the comparison of the results, all the data points have been normalized in the range $[0, 1]$ for the computing process. Note that due to the use of the NRMSE as the accuracy measure, this normalization does not affect the evaluation this objective.

Since type-2 FLSs are claimed to outperform their type-1 counterparts when uncertainties are present, we add Gaussian noise with 0-mean and σ-deviation

TABLE 6.2: Functions considered for the comparison of type-1 and type-2 FLSs

Function	Input range
$f_1(x_1, x_2) = \sin(x_1 x_2)$	$x_1, x_2 \in [-2, 2]$
$f_2(x_1, x_2) = \exp(x_1 \sin(\pi x_2))$	$x_1, x_2 \in [-1, 1]$
$f_4(x_1, x_2) = \dfrac{1 + \sin(2x_1 + 3x_2)}{3.5 + \sin(x_1 - x_2)}$	$x_1, x_2 \in [-2, 2]$
$f_5(x_1, x_2) = 42.659(2 + x_1)/20 + \mathrm{Re}(z^5)$ with $z = x_1 + ix_2 - 0.5(1 + i)$	$x_1, x_2 \in [-0.5, 0.5]$
$f_6(x_1, x_2) = 1.3356[1.5\,(1 - x_1) + \exp(2x_1 - 1)\ \sin(3\pi(x_1 - 0.6)^2) + \exp(3(x_2 - 0.5))\sin(4\pi(x_2 - 0.9)^2)]$	$x_1, x_2 \in [0, 1]$
$f_7(x_1, x_2) = 1.9[1.35{+}\exp(x_1)\sin(13(x_1{-}0.6)^2)\exp(-x_2)\sin(7x_2)]$	$x_1, x_2 \in [0, 1]$
$f_{10}(x_1, x_2, x_3, x_4) = \exp(2x_1 \sin(\pi x_4)) + \sin(x_2 x_3)$	$x_1, ..., x_4 \in [-0.5, 0.5]$
$y_2(x_1, x_2) = 0.5 + 64\dfrac{(x_1 - 0.5)(x_2 - 0.5)(x_1 + 0.2)}{1 + (4x_1 - 2)^2 + (4x_2 - 2)^2}$	$x_1, x_2 \in [0, 1]$
$y_3(x_1, x_2) = 0.5\exp\{-20[(x_1 - 0.5)^2 + (x_2 - 0.5)^2]\} + \exp[-10(x_1^2 + x_2^2)]$	$x_1, x_2 \in [0, 1]$
$y_5(x_1, x_2) = 0.5\,[1 + \sin(2\pi x_1) cos(2\pi x_2)]$	$x_1, x_2 \in [0, 1]$

to the input vectors at nine different levels, i.e., $\sigma = \{0, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.40, 0.50\}$. For testing, we consider a test dataset with 2025 noise-free samples forming a uniformly distributed 45×45 grid in the input space. In order to deal with the stochastic nature of EAs, the MOEA has been run ten times for each instance of the problems and the average results are used for the upcoming study.

In this work, we have used the implementation of the multi-objective evolutionary algorithm NSGA-II provided by MATLAB's Global Optimization Tool-

TABLE 6.3: Parameters of the proposed MOEA for the optimization of type-1 and interval type-2 FLSs

Parameter	Description	Value	Source
$[N_r^{min}, N_r^{max}]$	Minimum and maximum number of rules allowed in the FLS	$[3, 30]$	[Alcalá et al., 2009]
$[U_{min}, U_{max}]$	Initial range of values for the uncertainty factor. Final values can be outside this range	$[0, 0.2]$	[Cara et al., 2011b]
P_{sp}	Probability of a parent being chosen for single-point crossover	0.5	[Alcalá et al., 2009]
P_{mod}	Probability of a parent being modified by the mutation operator that modifies the number of rules	0.5	[Alcalá et al., 2009]
R_{max}	Maximum number of rules that can be added/removed by the mutation operator	5	[Alcalá et al., 2009]
P_m	Probability of a gene being changed by the second mutation operator	0.01	[Guenounou et al., 2009]

box® [MATLAB, 2010a]. It has to be noted that the proposed MOEA just intends to be a tool for obtaining comparable type-1 and type-2 FLSs, since this comparison is our final goal. For this reason, we have tried to reduce the influence of the learning method on the results by applying a commonly used optimization technique with a general configuration of parameters.

In this sense, the parameters that are specific for the proposed MOEA take values that have been already used in the literature with satisfactory results. These values are summarized in Table 6.3, which includes the source for the value of each parameter. The only exception is the range of initial values for the uncertainty factors $[U_{min}, U_{max}]$. To the best of our knowledge, the concept of "uncertainty factor" has not been explictly used in the literature for the definition of interval type-2 MFs. Hence, since there is not a guideline for selecting an appropriate value for this parameter, we have simply selected reasonable values. Note that $U_{min} = 0$ corresponds to the definition of a singleton type-1 FLS, whilst using $U_{max} = 0.20$ indicates that there is a 20% of uncertainty in the definition of the MF, which seems to be adequate. Moreover, it has been observed that higher values of U cause very large spreads in the MFs, and therefore, the FLSs may behave like random systems [Cara et al., 2011b].

TABLE 6.4: Main parameters for NSGA-II

Parameter	Description	Value
Population size	Number of chromosomes in the population	100
Crossover fraction	Fraction of the population that is created by the crossover operator	0.8
Mutation fraction	Fraction of the population that is created by the mutation operator	0.2
Maximum number of generations	Maximum number of iterations to be performed by the algorithm	400
Number of elite individuals	Number of individuals guaranteed to survive to the next generation	2
Selection function	Function that selects parents for the crossover and mutation operators	Tournament
Stall limit	The algorithm stops if there is no improvement for the specified number of iterations	50

For the remaining parameters (i.e., those related to MATLAB's implementation of NSGA-II), we have applied the default values, which are summarized in Table 6.4. The only non-default parameters are the population size and the number of generations. For the first one, we have selected the value used in the work where NSGA-II was initially proposed [Deb et al., 2002] for problems of similar complexity, which is still widely used in the literature. For the number of generations, we have selected a high value that allows the MOEA to achieve an appropriate convergence in all the cases considered [Gacto et al., 2009], i.e., the evolution stops because the improvement in the Pareto front over a period of time is not significant.

Nonetheless, it is worth noting that selecting a more specific configuration of parameters could increase the quality of the solutions obtained by the MOEA. Hence, this aspect will be addressed in future research.

6.4.1.2 Comparing the quality of the Pareto fronts

In order to investigate if the differences between type-1 and interval type-2 FLSs are significant, we compare the Pareto fronts obtained by the MOEA for the

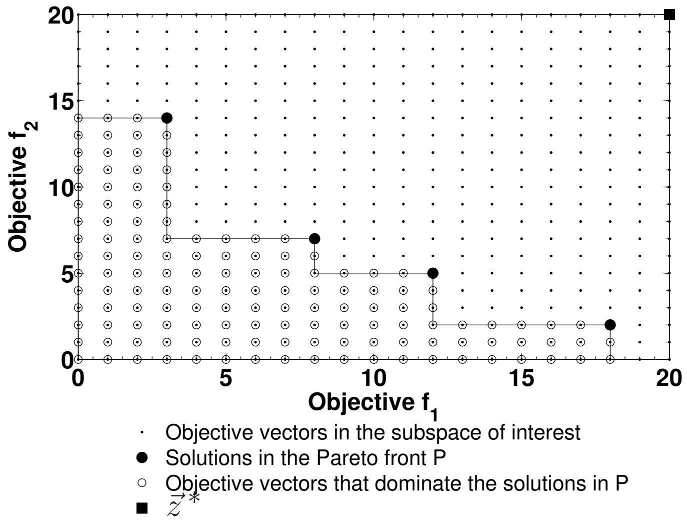


FIGURE 6.7: Example of the hypervolume indicator. $A(P)$ is determined by the objective vectors that dominate the solutions in the Pareto front P (depicted by circles), i.e., it represents the area of the surface delimited by the solid line

three types of FLS considered, at different levels of noise. To do so, we apply a comparison method based on a quality indicator [Zitzler et al., 2008]. The idea is to map each Pareto front into a real number (known as the *quality indicator*) and then perform a statistical analysis of the resulting values. Specifically, we use the hypervolume measure [Zitzler and Thiele, 1998] as the quality indicator for our study. The reason is that the hypervolume is the only strictly monotonic unary indicator known [Zitzler et al., 2007], i.e., for any Pareto front compared to another, if the former dominates the latter, then its quality indicator is also better.

Traditionally, the hypervolume indicator measures the volume of the portion of the objective space that is dominated by a specific Pareto front P [Zitzler et al., 2008]; hence, higher values of the indicator are considered better. However, since we are addressing a minimization problem, we redefine this quality indicator as the volume of the portion of the objective space that *dominates* the given Pareto front (see Fig. 6.7). Thus, our objective is to minimize this measure.

This quality indicator can be formally defined as

$$A(P) = \int_{\vec{0}}^{\vec{z}^*} \alpha_P(\vec{z})\, d\vec{z} \tag{6.18}$$

where $\vec{0} = (0, 0)$ and $\vec{z}^*$ is the objective vector representing the upper bound for the portion of the objective space within which the hypervolume is calculated, as depicted in Fig. 6.7. The function α_P is the attainment function [da Fonseca et al., 2001] for the Pareto front P, which is defined as

$$\alpha_P(\vec{z}) = \begin{cases} 0 & \text{if } \exists\, \vec{x} \in P : \vec{x} \prec \vec{z} \\ 1 & \text{otherwise} \end{cases} \tag{6.19}$$

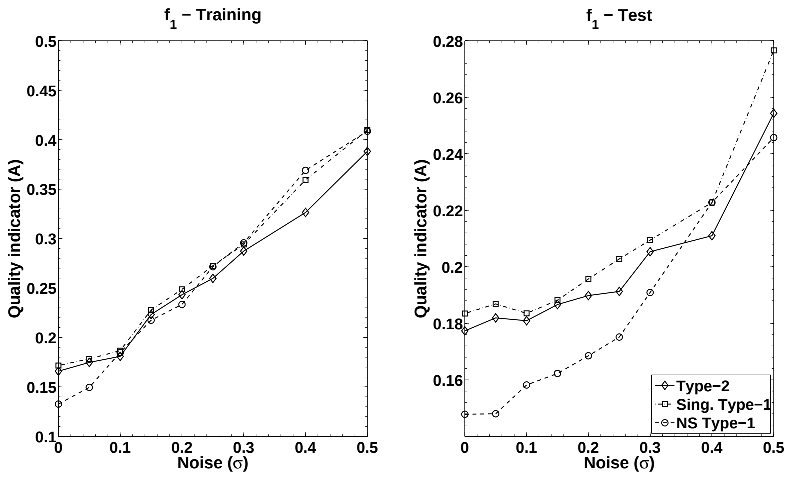
Note that α_P returns 1 for those objective vectors that dominate the solutions in the Pareto front P, and returns 0 for those that are dominated by some solution in P. For our two-objective optimization problem, the hypervolume $A(P)$ can be seen as the area of the surface that lies under the Pareto front P. For simplicity, the values of the hypervolume are normalized, so they represent the proportion of objective vectors (within the subspace considered) that dominate the solutions in the Pareto front P. Thus, for the example shown in Fig. 6.7, the value of the indicator is equal to $A(P) = 138/441 = 0.313$.

The hypervolume indicator is sensitive to the choice of the upper bound $\vec{z}^*$, i.e., it is not scaling invariant [Zitzler et al., 2008]. To overcome this issue, we choose a common point $\vec{z}^*$ for computing the hypervolume of all the Pareto fronts obtained. Specifically, we use the maximum point in all the Pareto fronts obtained, among all the problem instances considered. This way, the quality measure is computed with respect to a common subspace and the Pareto fronts obtained for the different problems can be compared.

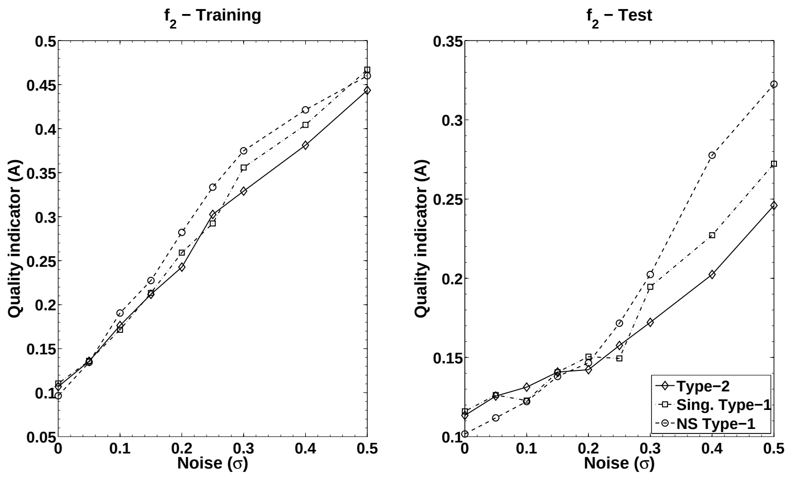
Fig. 6.8 depicts the evolution of the quality indicator $A(P)$ as the noise level increases. For each problem, type of FLS and noise level, the data shown is the average of the area obtained in the ten runs of the MOEA. The numeric data (mean areas and their standard deviation) are presented in Appendix B, in Tables B.1 and B.2 for training and test, respectively.

Obviously, the area increases as the noise level does, indicating a worsening of the quality of the Pareto fronts. However, it is observed that the effect of the noise on the quality indicators is not the same for the three FLSs. Although for lower noise levels the three systems present similar quality measures, as the noise raises, the type-2 FLS is observed to generally have a slower increase of

(a)



(b)



(c)

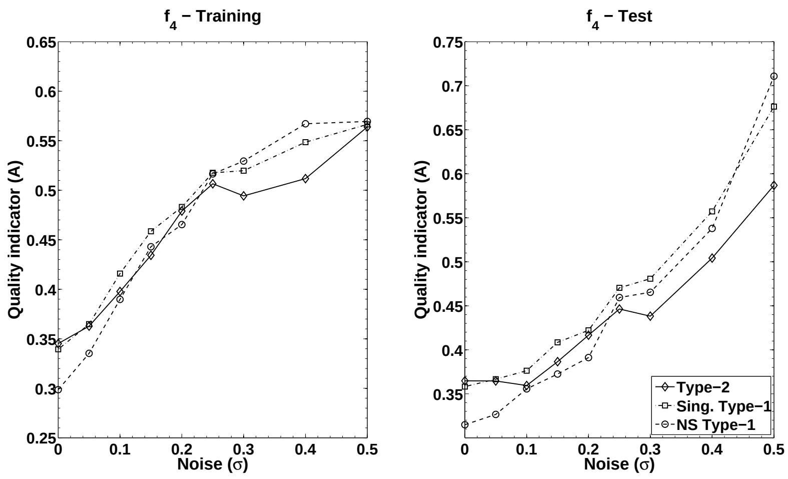
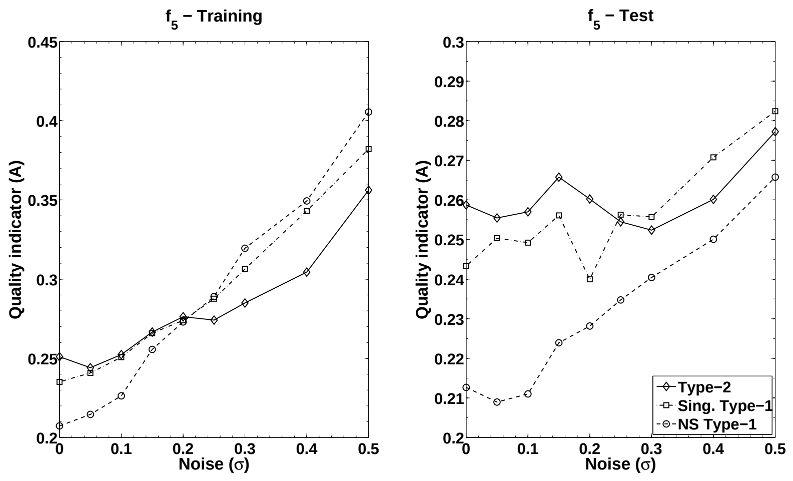
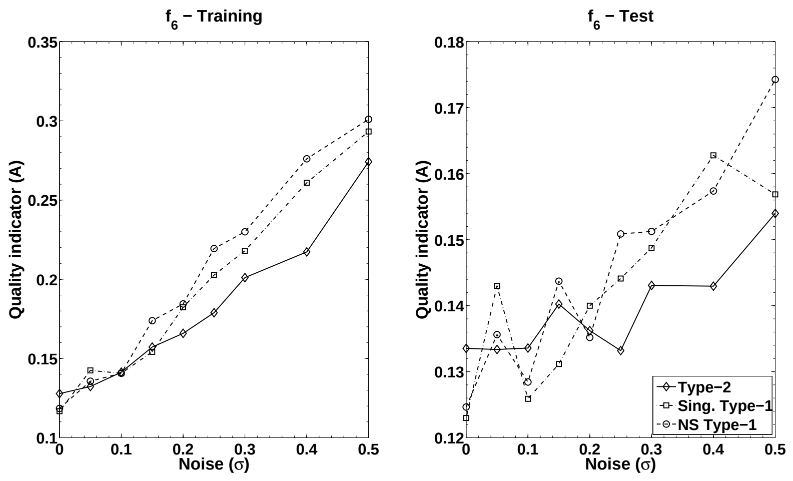


FIGURE 6.8: Evolution of the areas under the Pareto front as the noise level increases for training and test. (a) f_1, (b) f_2, (c) f_4

(d)



(e)



(f)

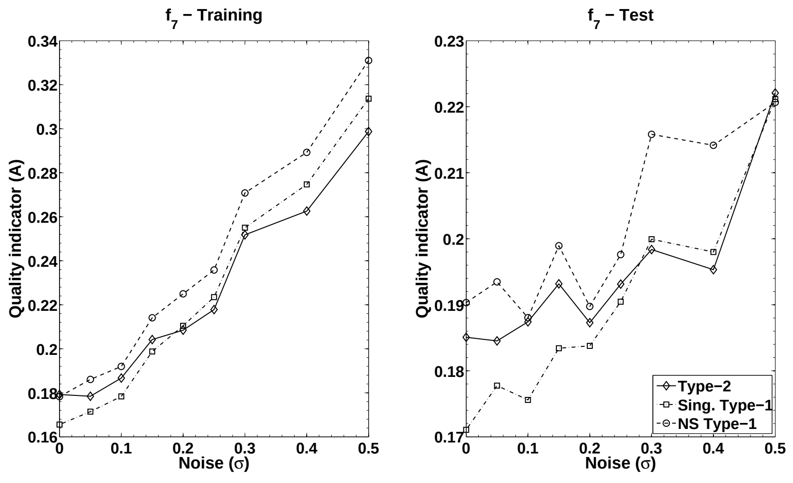
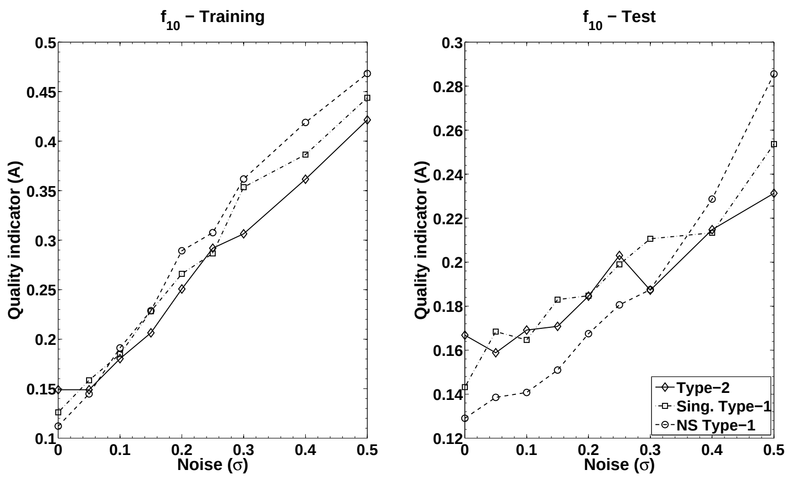
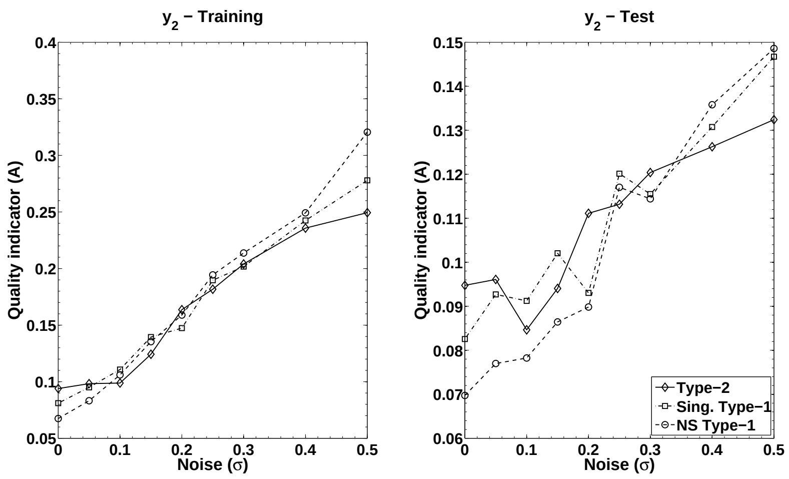


FIGURE 6.8: Evolution of the areas under the Pareto front as the noise level increases for training and test - continued. (d) f_5, (e) f_6, (f) f_7

(g)



(h)



(i)

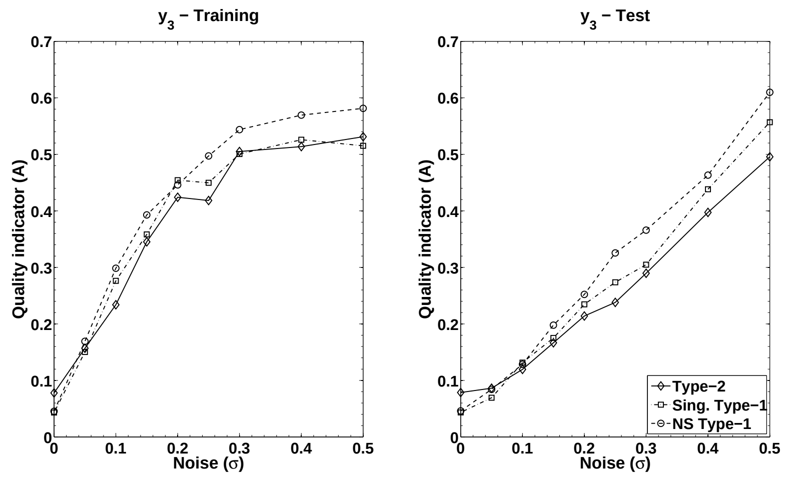


FIGURE 6.8: Evolution of the areas under the Pareto front as the noise level increases for training and test - continued. (g) f_{10}, (h) y_2, (i) y_3

(j)

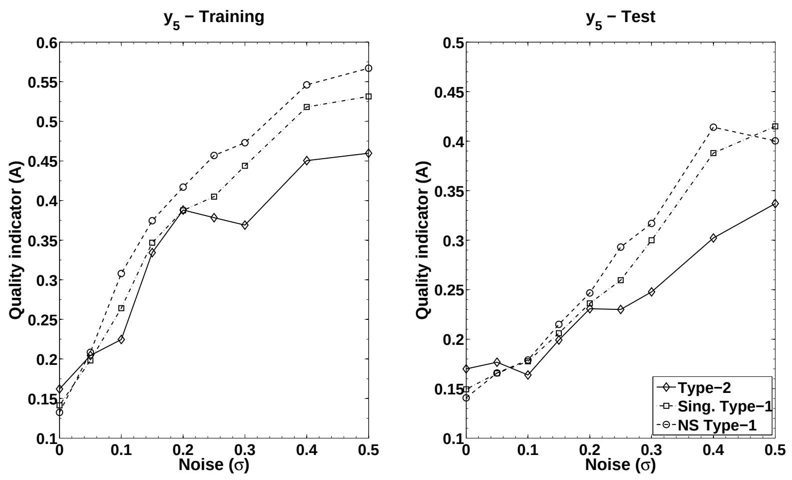


FIGURE 6.8: Evolution of the areas under the Pareto front as the noise level increases for training and test - continued. (j) y_5

the indicator. This suggests that the type-2 FLS handles the uncertainties (i.e., the noise) better than the other two FLSs.

In order to verify this observation, and to determine at which noise level significant differences appear, we perform statistical tests based on the quality indicators. Since the conditions for the application of parametric tests (e.g., the repeated-measures ANOVA [Box et al., 1978]) are not satisfied, we apply the equivalent non-parametric test [Luengo et al., 2009]. Thus, for each noise level we employ the Friedman test [Friedman, 1937] to determine whether the differences are statistically significant or not. Since the number of systems to be compared is small, we apply a variation of the Friedman test, the Friedman Aligned Ranks test [García et al., 2010].

The null hypothesis to be rejected states the equality of medians between the populations, i.e., its rejection implies the existence of differences between the behavior of the three systems. Table 6.5 summarizes the ranks obtained by each type of FLS in the Friedman Aligned Ranks test. It also shows the values of the Friedman's statistic in each case and the associated p-value. For a level of significance $\alpha = 0.05$, the value of the statistic in all the cases is higher than the corresponding critical value for a χ^2 distribution with 2 degrees of freedom[1].

[1]The critical value for a χ^2 distribution with 2 degrees of freedom is $\chi^2 = 5.99$ [Sheskin, 2007].

TABLE 6.5: Ranks obtained by each FLS in the Friedman Aligned Ranks test (α = 0.05) for the average quality indicators of the Pareto fronts

	Noise level (σ)								
	0	0.05	0.10	0.15	0.20	0.25	0.30	0.40	0.50
Singleton type-1	15.1	18.6	19.4	18.2	17.4	17.4	19.1	18.1	18.2
Non-singleton type-1	**7.6**	**8.4**	**9.8**	**12.4**	**12.0**	17.3	18.2	21.4	20.6
Interval type-2	23.8	19.5	17.3	15.9	17.1	**11.8**	**9.2**	**7.0**	**7.7**
Friedman's statistic F_{AR}	7.15	7.24	7.41	7.75	7.73	7.72	7.44	7.14	7.26
p-value	0.0280	0.0268	0.0246	0.0207	0.0210	0.0210	0.0243	0.0282	0.0265

Therefore, there are significant differences among the three types of FLS, with a confidence level of 95% [Sheskin, 2007].

The Friedman Aligned Ranks test only provides information about the existence of statistically significant differences about the samples compared. In order to detect the specific differences, a post-hoc test is required [Derrac et al., 2011]. The post-hoc procedure compares a control algorithm (in our case, the one with the best ranking for each noise level) with the remaining algorithms. In this study, we use the Li test [Li, 2008] because of its simplicity and power [García et al., 2010].

The results of the post-hoc test are presented in Table 6.6. For each noise level, the value of the statistic z and the corresponding unadjusted and adjusted p-values are shown. The FLSs to be compared to the control method are ordered in ascending order of their p-value. The last column indicates if the null hypothesis is rejected, i.e, whether the difference between the control method and the corresponding FLS is statistically significant or not. Several conclusions can be drawn from this table:

- When the noise level is very low (i.e., noise-free case and σ = 0.05) the non-singleton type-1 FLS shows the best performance. Nonetheless, it has to be noted that for the noise-free case, the difference with the singleton type-1 is not significant. On the contrary, the interval type-2 FLS presents the worst performance in this case. The reason for this behavior is that the level of uncertainty in the data is very low, and therefore, the mech-

TABLE 6.6: Li post-hoc test to detect differences between the different types of FLS, using the system with the best ranking as the control method, with a level of significance $\alpha = 0.05$

Noise level (σ)	Control method	FLS	$z = (R_0 - R_i)/SE$	p_u	p_{Li}	Rejected?
0	NS type-1	Type-2	4.1148	$3.9 \cdot 10^{-5}$	$4.1 \cdot 10^{-5}$	Yes
		Sing. type-1	1.9050	0.0568	0.0568	No
0.05	NS type-1	Type-2	2.8194	0.0048	0.0048	Yes
		Sing. type-1	2.5908	0.0096	0.0096	Yes
0.10	NS type-1	Sing. type-1	2.4384	0.0148	0.0154	Yes
		Type-2	1.9050	0.0568	0.0568	No
0.15	NS type-1	Sing. type-1	1.4732	0.1407	0.1835	No
		Type-2	0.8890	0.3740	0.3740	No
0.20	NS type-1	Sing. type-1	1.3716	0.1702	0.1745	No
		Type-2	1.2954	0.1952	0.1952	No
0.25	Type-2	Sing. type-1	1.4224	0.1549	0.1561	No
		NS type-1	1.3970	0.1624	0.1624	No
0.30	Type-2	Sing. type-1	2.5146	0.0119	0.0120	Yes
		NS type-1	2.2860	0.0222	0.0222	Yes
0.40	Type-2	NS type-1	3.6576	0.0003	0.0003	Yes
		Sing. type-1	2.8194	0.0048	0.0048	Yes
0.50	Type-2	NS type-1	3.2766	0.0010	0.0011	Yes
		Sing. type-1	2.6670	0.0077	0.0077	Yes

anisms provided by the interval type-2 FLS to handle the uncertainty are not needed.

- However, this difference decreases as the noise level increases. Hence, when $\sigma = 0.10$, the difference between the non-singleton type-1 FLS and the singleton type-1 FLS is still significant (the null hypothesis is rejected with $p = 0.0154$), but the difference between the non-singleton type-1 and the singleton interval type-2 is not ($p = 0.0568$).

- For medium levels of noise (i.e., $\sigma = \{0.15, 0.20, 0.25\}$) the differences between the three systems are not significant, i.e., there is not a system that is clearly better than the others.

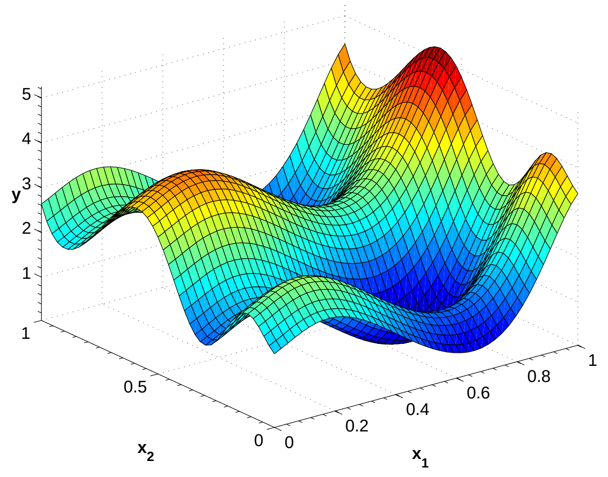


FIGURE 6.9: Function f_6

- Finally, when the noise level reaches high values, namely $\sigma = 0.30$ and above, the singleton interval type-2 FLS clearly outperforms the other two.

6.4.1.3 Example of the approximation obtained

In the previous subsection, the quality of the Pareto fronts obtained by the MOEA has been studied, leading to the conclusion that NS type-1 FLSs are preferred for low noise levels, but the type-2 FLS outperforms its type-1 counterparts when the noise level is higher. In this subsection, we present a representative example of the approximations obtained by the different types of FLSs.

Thus, we analyze the approximations obtained for function f_6 (depicted in Fig. 6.9) when the training data presents low and high levels of noise. Specifically, we consider the cases of $\sigma = 0.05$ (see Fig. 6.10(a)) and $\sigma = 0.50$ (see Fig. 6.10(b)), respectively. Since it is not possible to show the approximations obtained by all the solutions in all the Pareto fronts, we have used the following criteria to select the example shown: First, for each FLS, we have selected the execution that produces the Pareto front with the median value of the quality indicator; then, we have chosen the most accurate solution in that Pareto front.

The approximations obtained for the test data by the three FLSs are shown in Fig. 6.11 and 6.12, and the corresponding fitness is presented in Table 6.7. It is observed that when the noise level is low, the non-singleton type-1 system

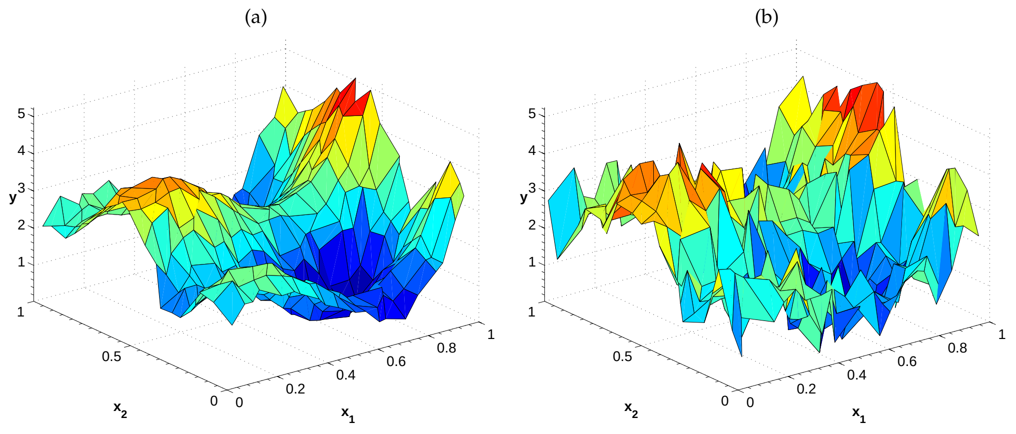


FIGURE 6.10: Training data for function f_6. (a) Training data with noise level $\sigma = 0.05$. (b) Training data with noise level $\sigma = 0.50$

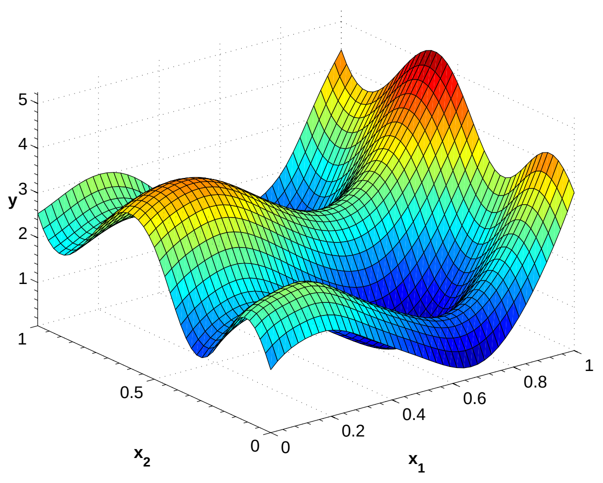
TABLE 6.7: Fitness of the most accurate approximations of function f_6 with different levels of noise

Type of FLS	Noise level $\sigma = 0.05$		Noise level $\sigma = 0.50$	
	Number of rules	NRMSE	Number of rules	NRMSE
Singleton type-1	29	0.0589	17	0.1447
Non-singleton type-1	30	0.0391	18	0.1539
Singleton interval type-2	28	0.0567	16	0.1374

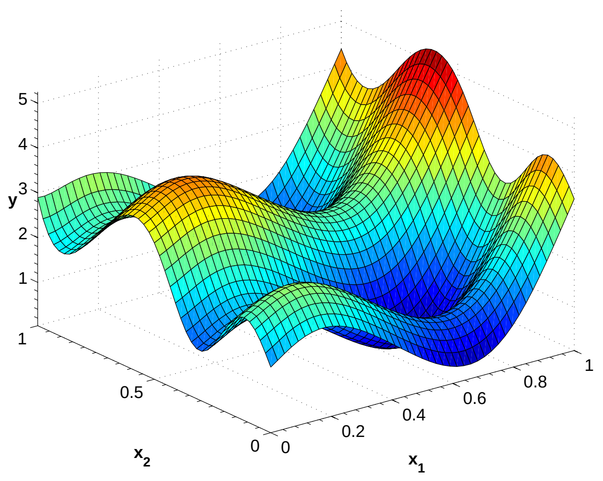
produces the most accurate approximation. However, the interval type-2 FLS provides a better approximation when the noise level is high. This result was expected in the view of the statistical results obtained in the previous section.

Finally, in Fig. 6.13, we present the Pareto fronts obtained for both cases. This figure shows that the evolution of the Pareto fronts coincides with the behavior previously observed: when the noise level is low, the non-singleton type-1 FLS clearly outperforms the others, whilst the singleton interval type-2 FLS behaves better in the presence of high noise. Additionally, it is remarkable that the Pareto fronts include solutions with a higher number of rules when the noise level is low. The reason for this is that, as the level of noise increases, the training process is more difficult and some overfitting may appear for longer solutions.

(a)



(b)



(c)

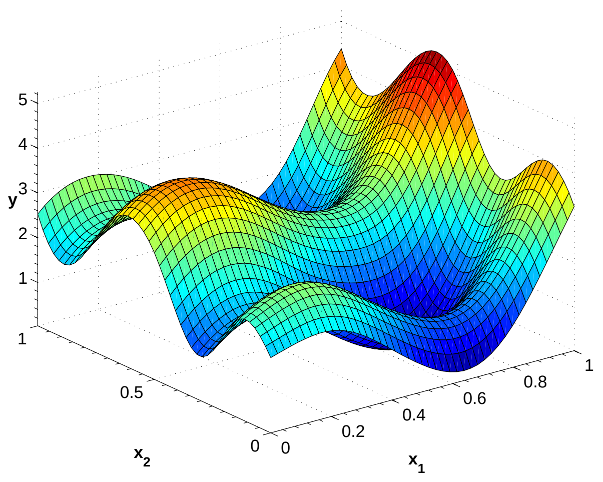
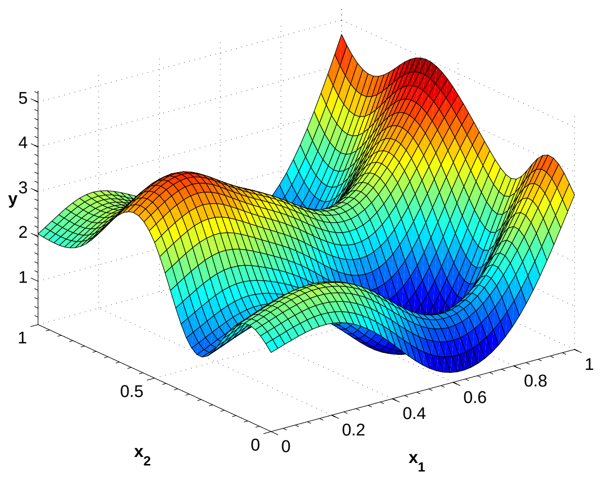
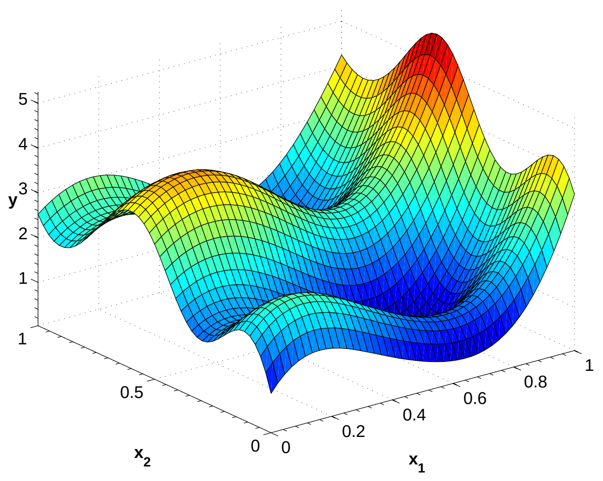


FIGURE 6.11: Approximation of function f_6 when the noise level in the training data is $\sigma = 0.05$. (a) Singleton type-1 FLS. (b) Non-singleton type-1 FLS. (c) Singleton interval type-2 FLS

(a)



(b)



(c)

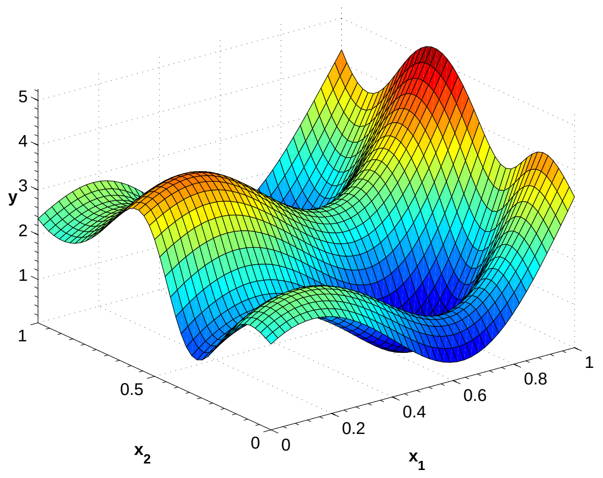


FIGURE 6.12: Approximation of function f_6 when the noise level in the training data is $\sigma = 0.50$. (a) Singleton type-1 FLS. (b) Non-singleton type-1 FLS. (c) Singleton interval type-2 FLS

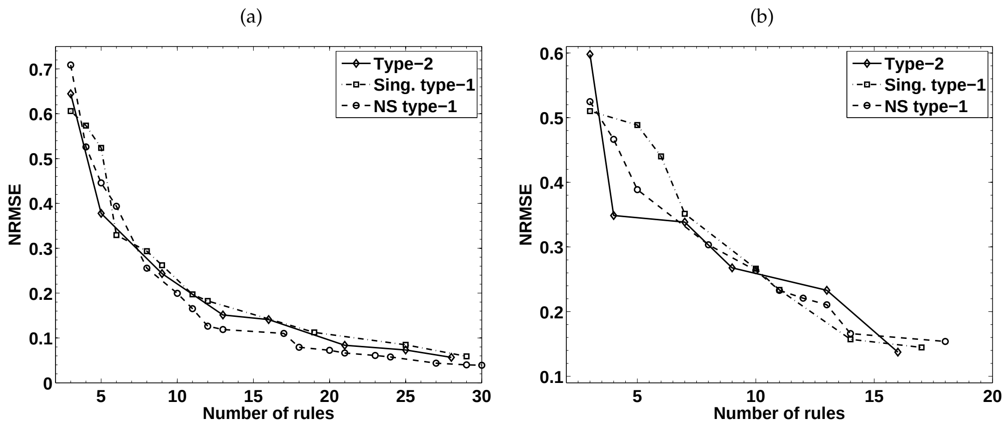


FIGURE 6.13: Pareto fronts obtained for the approximation of function f_6 with different levels of noise (test) (a) When the noise in the training data is $\sigma = 0.05$. (a) When the noise in the training data is $\sigma = 0.50$.

6.4.1.4 Effect of the use of optimal consequents

In section 6.3.2.5, we introduced a method for computing the optimal consequents for a given rule base. The use of this method within the context of the multi-objective evolutionary algorithm serves two purposes: On the one hand, it reduces the size of the search space, thereby reducing in N_r the number of parameters that have to be optimized for a system with N_r rules. On the other hand, it enhances the optimization process by improving the quality of each solution in the population, thus leading to more accurate Pareto fronts.

In order to visualize this property, we present here an example of this effect. Specifically, we have applied the MOEA to optimize a singleton interval type-2 FLS for the approximation of function f_1. Starting from the same initial population, we have performed the evolutionary search twice, i.e., using the optimal consequents as described in this chapter, and including the consequents in the chromosome and using the MOEA to optimize them. In both cases, the same seed has been used for the generator of pseudo-random values. The test has been performed five times, obtaining similar results in all of them.

Fig. 6.14 shows the fitness obtained for the portion of the initial population resulting from the application of the K-means and the KNN algorithms to select the parameters of the antecedents (see section 6.3.2.2). The stars represent the fitness of the chromosomes that use optimal consequents, whilst the circles

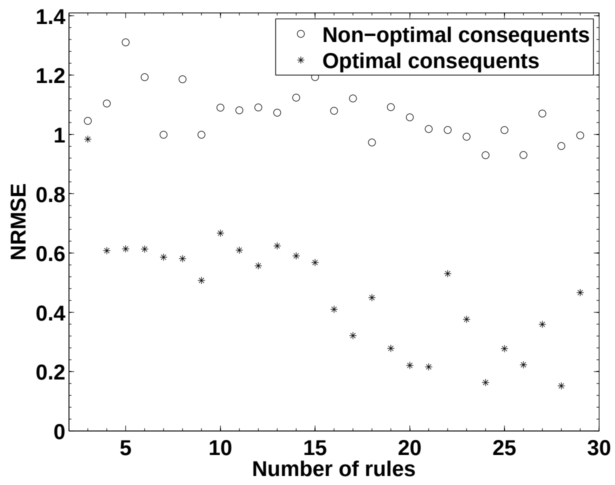


FIGURE 6.14: Fitness of a portion of the initial population of the MOEA when optimal consequents are used for the rules (stars), and when they are included in the evolutionary optimization process (circles)

correspond to the solutions that optimize the consequents as part of the evolutionary approach. In both cases, the antecedents of the rules are the same.

The enhancement in the quality of the rules when the optimal consequents are used is obvious, even for this population in which the antecedents have not been refined yet. This improvement helps the search process to obtain final Pareto fronts with better accuracy for the same number of rules. Fig 6.15 depicts the final Pareto fronts obtained in the two cases considered. It can be observed that the solutions obtained when using the optimal consequents are more accurate.

6.4.2 Analysis of a sample application: control of a nonlinear servo system

We conclude this section of experimental results by presenting an example of the application of the three types of FLS considered for a control problem. Specifically, we a singleton type-1 FLC, a non-singleton type-1 FLC, and a singleton interval type-2 FLC to control the nonlinear servo system described in Appendix A.4 [Cara et al., 2010a]. The three controllers have the same structure and use the same rule base, since the non-singleton type-1 FLC and the interval

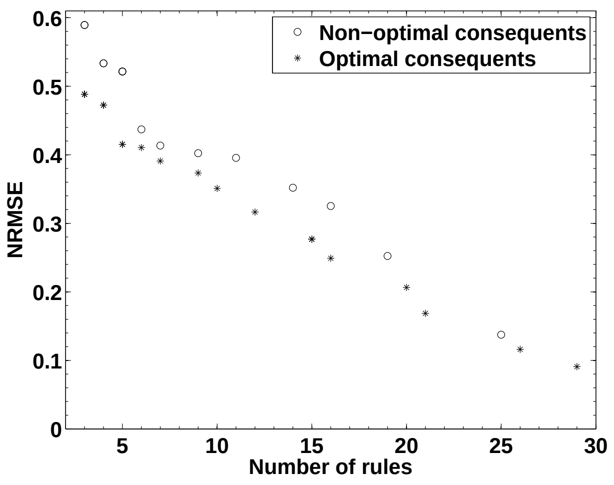


FIGURE 6.15: Pareto front obtained by the MOEA when optimal consequents are used for the rules (stars), and when they are included in the evolutionary optimization process (circles)

type-2 FLC have been obtained from the singleton type-1 controller. The output MFs are also the same.

The control objective in these experiments is to track a step-like reference signal r, for a time span of 100 seconds. The sampling time used to exert control actions is $T_c = 0.05\,s$. To measure the performance of the controllers, we have computed the mean square error of the tracking over the whole simulation time and the standard deviation of the square errors.

The three controllers use three input variables, namely the reference value r, the current output angle of the plant θ, and the angular velocity $\dot{\theta}$. Uniformly distributed triangular membership functions are used for both the inputs and the outputs, the minimum is used as the t-norm and the center-of-sets is selected as the defuzzification method. More specifically, the three controllers are designed as follows:

- The singleton type-1 FLC was automatically obtained and tuned by applying the online self-evolving method described in chapter 4 [Cara et al., 2011a]. Once the control performance was satisfactory, the online learning mechanism was switched off and the resulting classical type-1 controller was used for the upcoming experiments. As a result of the learning, three

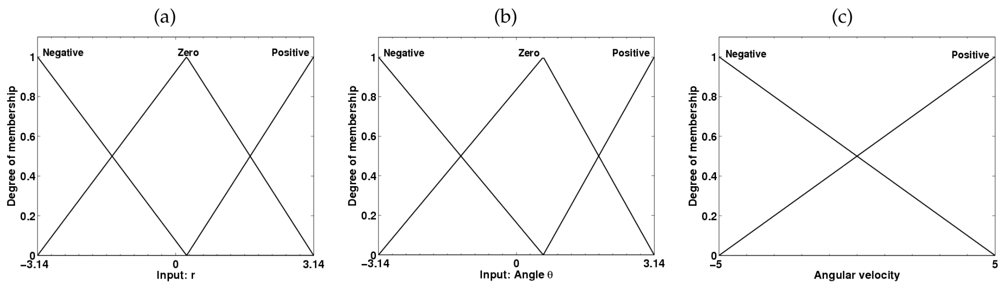


FIGURE 6.16: MFs for the singleton type-1 controller. (a) Input r, (b) input θ, and (c) input $\dot{\theta}$

membership functions were assigned to the inputs r and θ, and two MFs were assigned to the input $\dot{\theta}$. These MFs are depicted in Fig. 6.16.

- The non-singleton type-1 FLC uses the same MFs for the inputs and outputs and the same rules as the singleton type-1 FLC. The only difference is that the inputs to the non-singleton controller are fuzzy values instead of crisp numbers. The fuzzy input variables to the non-singleton type-1 FLC are centered on the crisp input value and spread according to the manually specified uncertainty factor U, as shown in Fig. 6.17(a).

- The type-2 FLC parameters are also obtained from the singleton type-1 FLC. The FOUs of the input sets use the type-1 fuzzy sets as the lower MFs and blurry them with the uncertainty factor U. Thus, the upper MFs are obtained by adding U to the left and right ends of the type-1 lower MFs. Fig. 6.17(b) depicts the way the type-2 MFs are obtained.

- Finally, the output MFs are the same in all the controllers. Triangular type-1 membership functions have been used for the consequents of the rules. The centers of these MFs are the values obtained for the consequents by the evolving method and the slopes are set accordingly to the centers of the adjacent MFs.

The experiments are classified into three groups, according to the level of noise applied to the inputs of the controller: First, we analyze the cases of low or no noise; then, we study the case for medium noise levels; finally, we comment on the cases with high levels of noise. For these three groups of tests, we have

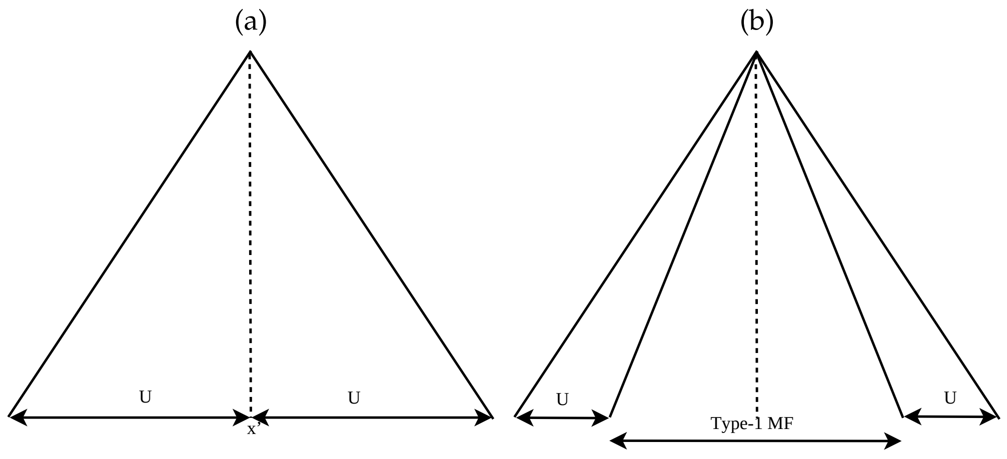


FIGURE 6.17: Construction of non-singleton type-1 and type-2 MFs from singleton triangular type-1 MFs. (a) Triangular input MF for the input value x' in the NS type-1 FLC, with uncertainty factor U. (b) Triangular type-2 MF with uncertainty factor U

analyzed different values for the uncertainty factor U, i.e., different spreads for the MFs. The values of noise applied to the inputs are $N \in \{0, 5, 10, 15, 20\}\,\%$ of the range of the input variables. The values of the uncertainty factor used in this study are $U \in \{5, 10, 15, 20, 30\}\,\%$. Larger values for the uncertainties were also tried; however, they cause very large spreads in the MFs and, therefore, the controllers behaved like random systems. In the following subsections, we present the results of the comparative study.

6.4.2.1 Ideal case: no noise or a small level of noise

In this first group of experiments, we consider the ideal case (i.e., there is no noise in the inputs) and the case in which the noise is $N = 5\%$ of the range of the input variables. Fig. 6.18, 6.19 and 6.20 show the tracking performance of the singleton type-1 FLC, the non-singleton type-1 FLC and the singleton interval type-2 FLC, respectively, when there is no noise present in the controller's inputs. The thin solid line depicts the desired trajectory r, while the dotted lines represent the output of the plant. For the non-singleton type-1 and the singleton interval type-2 FLCs, three lines are shown, corresponding to different uncertainty factors (namely 10, 20 and 30% of the input range). The first thing that

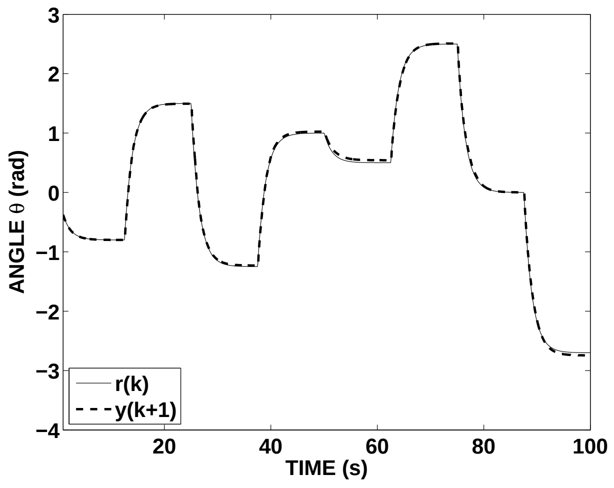


FIGURE 6.18: Reference tracking with the singleton type-1 FLC (no noise)

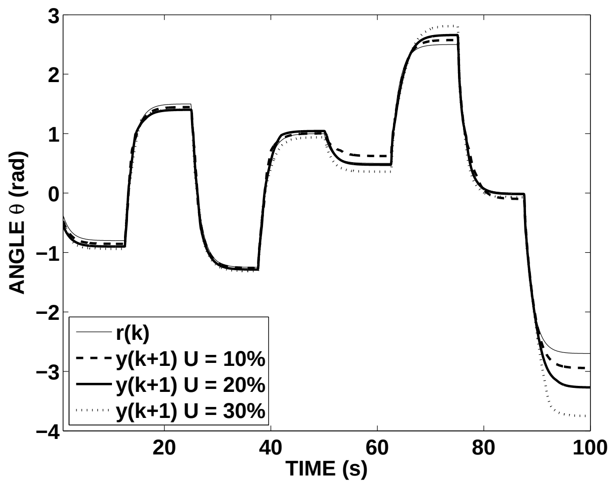


FIGURE 6.19: Reference tracking with the non-singleton type-1 FLC (no noise)

we observe is that in the ideal case with no noise, the best option is to use a singleton type-1 FLC. Furthermore, Fig., 6.19 and 6.20 show that larger values of the uncertainty factor (i.e., larger spreads in the membership functions) cause a worsening in the performance when – as in this case – there is no noise actually present in the environment/application.

Table 6.8 shows the MSE and the standard deviations obtained when the noise in the controller′s inputs is $N = 5\%$. Again, the results show that the best control performance is achieved by the singleton type-1 FLC.

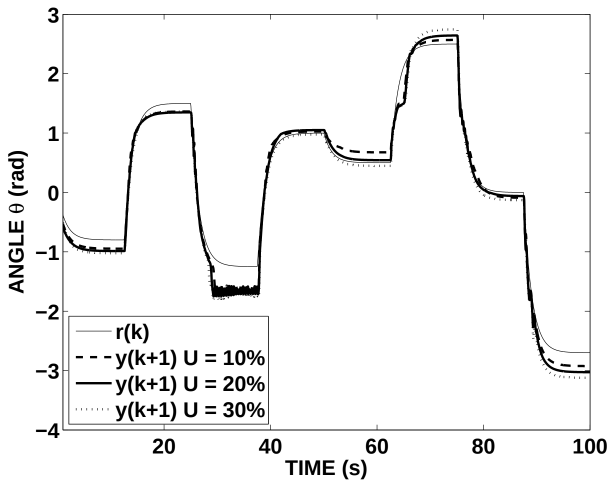


FIGURE 6.20: Reference tracking with the singleton interval type-2 FLC (no noise)

TABLE 6.8: Performance of the three types of controllers with 5% of noise

U	Singleton type-1		Non-singleton type-1		Interval type-2	
	MSE	Std. Dev	MSE	Std. Dev	MSE	Std. Dev
0%	0.0074	0.0136	0.0074	0.0136	0.0074	0.0136
5%	-	-	0.0112	0.0166	0.0397	0.0513
10%	-	-	0.0164	0.0271	0.0459	0.0582
15%	-	-	0.0235	0.0492	0.0520	0.0684
20%	-	-	0.0398	0.0947	0.0600	0.0809
30%	-	-	0.1182	0.3048	0.0839	0.1118

These results were expected, as the singleton type-1 controller has been specially designed to control this plant and, therefore, it is more precise than those that assume that there is some type of uncertainty when in practice there is no such uncertainty. Of course, it is worth noting that this ideal, uncertainty–free state is a direct result of employing a simulated environment and is generally not achievable in real world applications.

TABLE 6.9: Performance of the three types of controllers with 10% of noise

U	Singleton type-1		Non-singleton type-1		Interval type-2	
	MSE	Std. Dev	MSE	Std. Dev	MSE	Std. Dev
0%	216.440	237.491	216.440	237.491	216.440	237.491
5%	-	-	58.659	154.676	0.0655	0.0931
10%	-	-	0.0682	0.2239	0.0717	0.1010
15%	-	-	0.0547	0.1258	0.0731	0.0938
20%	-	-	0.0666	0.1415	0.0788	0.1015
30%	-	-	0.1462	0.3471	0.1185	0.1746

6.4.2.2 Experiments with medium levels of noise

In this second category of experiments, we have included those in which the noise level is N = 10% and N = 15%. Note that these values of noise cause three different situations with respect to the expected uncertainty factors: First, the noise may be smaller than the expected uncertainty; second, the noise may be close or equal to the uncertainty factor; and, finally, the noise may be larger than expected when the controllers were designed.

Table 6.9 gathers the MSE and the standard deviation obtained by the three controllers when there is a 10% of noise in the inputs. First, it is clear that the singleton type-1 FLC is not capable of coping with this level of noise. As mentioned before, this is due to the precise definition of the membership functions during the design. On the other hand, we observe that the non-singleton type-1 FLC is also unable to control the system when the noise is higher than the given uncertainty factor (i.e., N = 10% and U = 5%). However, the singleton interval type-2 FLC is able to handle this noise and produces a good response in the tracking. Nevertheless, when the noise level is close to the expected uncertainty factor, the MSE is very similar in both cases, although the standard deviation of the non-singleton type-1 FLC is slightly higher. Only when the uncertainty factor is much larger than the noise (i.e., N = 10% and U = 30%) the type-2 FLC obtains a smaller error and deviation.

Fig. 6.21 depicts the MSE and the standard deviation of the non-singleton

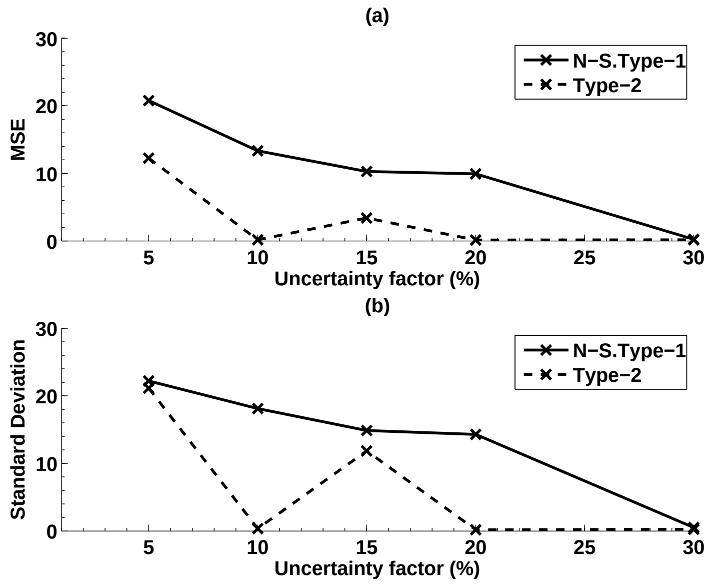


FIGURE 6.21: Control performance for the non-singleton type-1 FLC and the type-2 FLC when there is a 15% of noise. (a) MSE. (b) Standard deviation

type-1 FLC and the singleton interval type-2 FLC when the noise level is 15% for the different values of uncertainty factors considered during the design. In this case, the type-2 FLC clearly outperforms the non-singleton type-1 FLC. Note that neither one of the controllers is capable of handling the noise when the uncertainty factor is much smaller (i.e., U = 5%), although the type-2 controller starts providing good control performance with a smaller uncertainty factor (i.e., U = 20% in contrast with the U = 30% required by the non-singleton type-1 FLC).

6.4.2.3 Experiments with high levels of noise

Finally, we present the results when higher levels of noise are present in the controllers' inputs. Fig. 6.22 depicts the MSE and the standard deviation of the non-singleton type-1 FLC and the type-2 FLC when the noise level is 20%, for the different uncertainty factors considered during the design. First, it is observed that neither controller is capable of properly controlling the plant when the noise is above the uncertainty factor, although the errors provided by the type-2 FLC are smaller. Furthermore, the performance of the non-singleton type-1 controller

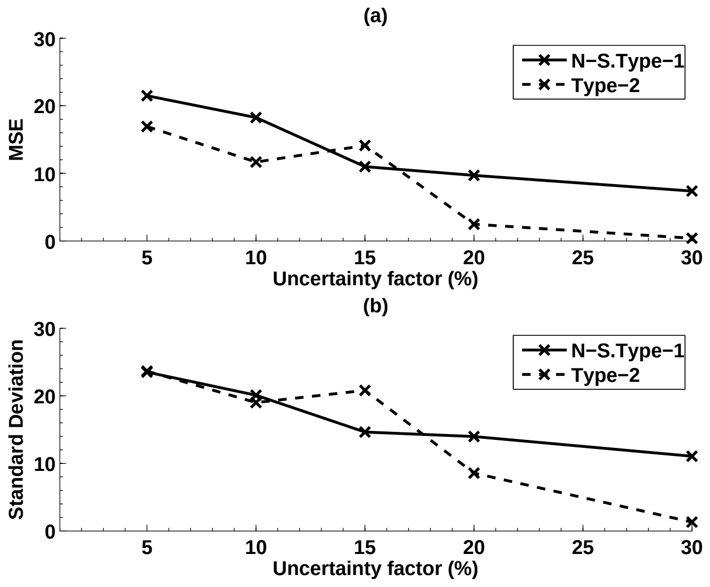


FIGURE 6.22: Control performance for the non-singleton type-1 FLC and the type-2 FLC when there is a 20% of noise. (a) MSE. (b) Standard deviation

is poor in all the cases, showing that it is not capable of handling high levels of noise, even if they were expected during the design process. On the contrary, the type-2 FLC continues producing a satisfactory control when the noise is below the uncertainty factor, i.e., for $N = 20\%$ and $U = 30\%$.

During the experiments, it has been observed that the values of uncertainty factors that produce better results are $U = 20\%$ and $U = 30\%$.

6.5 Conclusions

In this chapter, we have presented a comparative analysis between singleton type-1 FLSs, non-singleton type-1 FLSs and singleton interval type-2 FLSs. As we have detailed in this dissertation, the type-2 FLS is theoretically capable of handling uncertainties from various sources whilst, at least in concept, the non-singleton type-1 FLS can only handle the input uncertainties. In addition, the NS type-1 FLS handles the uncertainties only in the fuzzification stage, whereas the type-2 FLS propagates the uncertainty through the different inference stages. Hence, it should be expected that the singleton interval type-2 FLS should be better able to handle the faced uncertainties.

To provide an adequate comparative framework, we have presented a multi-objective evolutionary algorithm for the optimization of type-1 and singleton interval type-2 FLSs from data samples. The proposed MOEA concurrently optimizes the complexity and the accuracy of the fuzzy systems, thus providing a set of Pareto-optimal configurations. In addition, the proposed optimization approach includes a module for computing the optimal consequents for a given rule base for both type-1 and singleton interval type-2 fuzzy systems. This module helps to reduce the size of the search space, as the consequents do not need to be optimized by the MOEA. Moreover, it has been shown that the solutions obtained when using optimal consequents are more accurate (for the same system complexity) than the ones obtained when the consequents are optimized by the MOEA.

The three types of FLSs have been compared for function approximation and for a sample control application. In the first case, we have used the proposed MOEA to optimize the three types of FLSs for several function approximation problems under different levels of noise in the input data. We have used the hypervolume indicator as a quality measure to compare the Pareto fronts obtained by the MOEA for each type of fuzzy system and each noise level. Then, we have applied non-parametric statistical tests to analyze the results and to try to answer two different questions: (i) Whether the differences in the performance of type-1 and type-2 FLSs are significant or not, and (ii) at which noise level type-2 fuzzy systems should be preferred to type-1 FLSs, and vice versa.

In the second case, we have presented a framework where the different types of controllers have the same rule base, the same output fuzzy sets, and where the non-singleton type-1 FLC and the singleton interval type-2 FLC share the same uncertainty factors. For this experiment, we have used a singleton type-1 FLC automatically optimized (using the approach presented in chapter 4) for the control of a nonlinear servo system. Then, we have designed a non-singleton type-1 and a singleton interval type-2 FLCs that are structurally equivalent to their singleton type-1 counterpart. This has provided us with a framework in which all the controllers are as similar to each other as possible. We have carried out comparative experiments with different levels of noise in the controller's inputs.

Different levels of uncertainty factors have also been defined in the controllers during their design.

The results show that for low noise levels, the best performance is obtained by the non-singleton type-1 FLS. Nonetheless, the noise-free case requires a deeper consideration: On the one hand, in the experiments related to function approximation, the NS type-1 FLS showed the best performance, but the difference with the singleton type-1 FLS was not statistically significant. On the other hand, for the control sample application, the classical singleton type-1 FLC provided the best response in the absence of noise. This was expected, since this controller had been specifically tuned for this control problem. It is worth noting that the singleton type-1 FLS has the simplest structure and therefore, it should be used when possible. However, in real-world applications it is almost impossible to avoid uncertainties (e.g., noise in the sensors inputs) and more flexible methods capable of handling them are needed.

In this sense, it has been observed that both the non-singleton type-1 and the singleton interval type-2 FLSs can handle noise in the system's inputs to a certain degree. However, it has been shown that the type-2 FLS is able to accept higher levels of noise than its non-singleton type-1 counterpart. As such, the results show that while the non-singleton type-1 FLS provides advantages over the singleton type-1 FLS for medium levels of noise, the singleton type-2 FLS is better able to handle higher uncertainty levels.

Finally, by performing a side-by-side comparison of non-singleton type-1 FLSs and singleton interval type-2 FLSs, we have been able to clarify the difference in the actual handling of the uncertainty, i.e., it is only in the type-2 FLS that the uncertainty flows from the inputs to the outputs, through the complete inference system. It is this capability (rather than the increased number of parameters) that generally allows type-2 FLSs to perform better uncertainty handling than comparable type-1 FLSs.

Chapter 7

Conclusions of the dissertation and list of publications

The work presented in this dissertation represents a contribution to the areas of evolving fuzzy systems and intelligent control. The main contributions and conclusions extracted from this work are summarized in this chapter.

7.1 Conclusions and contributions

In this thesis, we have proposed two new methodologies for the online self-evolution of fuzzy controllers when there is no prior knowledge about the plant to be controlled. They provide a means of obtaining fuzzy controllers when the lack of knowledge makes it difficult (or impossible) to apply other methods. These two methodologies share some common properties and features, which can be summarized as follows:

- *No model of the plant or its differential equations are required*. Moreover, the methods proposed do not make any assumptions about the differential equations and only require qualitative knowledge about the plant (specifically, the sign of the monotonicity of the plant with respect to the control signal).

- *The fuzzy controller is designed from scratch, while controlling the plant online*. To do so, only the input/output information obtained online from the system's normal operation is used.

- *The structure of the fuzzy controller is incrementally evolved by exploiting the property of universal approximation of fuzzy systems*. Thus, increasing degrees of accuracy are achieved as the operation goes on. Moreover, the information applied to develop the controller covers the entire operating region of the system, thereby improving the robustness of the learning method.

- *The controllers developed are robust*, in the sense that they can cope with unexpected isolated changes in the plant's dynamics. Their adaptive nature provides them with the ability to learn the new behavior of the plant in order to continue delivering a satisfactory performance.

- *The evolving methodology is a life-long learning process*. As such, the controller continues learning and evolving during its entire operation. Two different types of learning are applied: On the one hand, in the parameter learning phase, the information about the error at the plant's output is used to adapt online the rule consequents. On the other hand, the error of the approximation of the plant's inverse function is analyzed to incrementally evolve the structure of the controller.

The first methodology proposed is OSEFC (see chapter 4). In this method, a complete set of rules is applied for the controller. Thus, adding a new membership function implies creating all the possible rules that use this new MF as an antecedent. Two complementary methods for parameter learning have been proposed for OSEFC. The first method is an online local adaptation procedure based on the reduction of the current error at the plant's output (i.e., based on local information). It applies a reward/penalty policy to move the consequents in the direction that reduces the error. In the second method, the global input/output data provided by the plant's operation is used to fine-tune the rule consequents. Specifically, a gradient-based optimization technique is used to minimize the error at the controller's output as a means of indirectly minimizing the error at the plant's output. In this case, the local learning method acts as a supervisor to guarantee that the changes applied to the consequents do not compromise the local control of the plant.

As mentioned before, the evolution of the structure in OSEFC is based on the improvement of the approximation of the plant's inverse function. The approximation error in the entire operating region is analyzed to identify the variable that is mostly responsible for this error. Then, a new membership function is added to this input variable, along with all the related rules. Thanks to this process, OSEFC is able to detect which inputs are important for the control process, thereby acting as an automatic input selector.

Simulation and experimental results have been used to illustrate the capabilities of OSEFC. The results with plants of increasing complexity have shown that OSEFC can effectively learn the control policy, even when starting from an empty controller. It has also been shown that it can deal with noise in the input readings and with changes in the dynamics of the plant. Moreover, it has been observed that OSEFC can outperform a non-adaptive fuzzy controller in the latter situation. However, the use of a complete set of rules causes the exponential growth of the number of rules, i.e., OSEFC suffers the curse of dimensionality, a well-known problem that is inherent to the use of grid-like distributions of fuzzy rules. As a consequence, the complexity of the system increases and damages the interpretability of the controller.

In order to tackle this problem, we have proposed a second methodology, named OSENF-TaSe (see chapter 5). In OSENF-TaSe, a scatter partitioning of the input space is used. This allows for an uneven distribution of the fuzzy rules, which can be directly placed in areas with high nonlinearities. In addition, this methodology is based on the TaSe-NF model, which preserves the interpretability of the local sub-models (rules). This is achieved by the use of a modified inference process that guarantees that the rule consequents equal the system's global output at the respective rule centers. Hence, each rule consequent has a direct meaning that is related to the controller's output.

As in OSEFC, the online local learning method takes care of the short term response of the controller by updating the rule consequents. Then, the structure of the controller is incrementally refined by splitting the rules that show an unsatisfactory performance in their areas of influence. Unlike OSEFC, OSENF-TaSe adds just one rule at a time. This helps to keep the growth of the number of rules under control. Finally, a gradient-descent method is used to further optimize all the parameters of the fuzzy controller (i.e., the center and radius of the antecedents and the values of the consequents) when a new rule is added.

The simulation results presented in chapter 5 have shown that OSENF-TaSe effectively reduces the complexity of the fuzzy controller, compared to OSEFC. Similarly, it has been shown how the interpretability of the rule base is enhanced with respect to traditional scatter-partitioning fuzzy controllers. OSENF-TaSe has also been compared with a classic controller designed with full knowledge about the plant. In this case, the results have shown that the control performance achieved by OSENF-TaSe is comparable to the performance of the classic controller. Moreover, the proposed methodology outperforms the classic controller when facing changes in the dynamics of the plant.

These results demonstrate that OSEFC and OSENF-TaSe are suitable alternatives for the automatic design of fuzzy controllers when there is no prior knowledge about the plant. Nonetheless, as a consequence of the lack of prior information about the plant, these approaches suffer some limitations. Firstly, since the controllers self-design from scratch while working online, the full methodologies proposed cannot be applied to plants with critic states that must never be reached. In this case, the controllers can be pre-trained in order to obtain a control policy that is sufficiently reliable when it starts being applied to the plant. Secondly, their application is limited to plants for which the controller's sampling period can be set in a way such that the plant's output depends on the control signal applied in the previous time step. Finally, the lack of a mathematical model makes it almost impossible to provide a formal demonstration of the stability of these controllers (although this issue has been approached from a qualitative perspective in this dissertation). Nevertheless, our proposals set an initial framework in which to develop new controllers to achieve the ultimate goal of online intelligent control with minimum previous knowledge about the plant.

Finally, a comparative analysis of type-1 and type-2 fuzzy logic systems has been presented in chapter 6. This study aimed to answer two questions that often arise in the context of type-2 fuzzy systems: (i) Are the differences in the performance of type-1 and type-2 FLSs significant? And (ii) for which level of uncertainty is one method preferred to the other?

With this goal in mind, we have proposed a multi-objective evolutionary algorithm for the optimization of the rule base and the parameters of type-1 and interval type-2 fuzzy systems. This MOEA provides a set of Pareto-optimal solutions with respect to the conflicting objectives of complexity and accuracy. The proposed method includes a module for computing the optimal consequents for a given rule base, based on a set of input/output vectors. This module improves the performance of the solutions obtained by the MOEA and simplifies the optimization process by reducing the size of the search space. The use of the MOEA provides a set of FLSs whose configurations have been optimized under the same conditions, thereby providing a common framework for the comparison of type-1 and interval type-2 fuzzy systems.

Specifically, we have compared singleton type-1, non-singleton type-1 and singleton interval type-2 FLSs in two different contexts:

- Firstly, we have applied the proposed MOEA to optimize the three types of FLSs for several function approximation problems under different levels of noise in the input data. Using the hypervolume measure as a quality indicator, we have compared the Pareto fronts obtained and have carried out a non-parametric statistical test to analyze the results.

- Secondly, we have analyzed a sample application consisting in the control of a nonlinear servo system. In this case, we have presented a framework where the different types of controllers considered have the same rule base, the same output fuzzy sets, and where the non-singleton type-1 FLC and the singleton interval type-2 FLC share the same uncertainty factors. First, we have applied OSEFC to automatically obtain a singleton type-1 FLC for the application considered. Then, we have designed a non-singleton type-1 FLC and a singleton interval type-2 FLC that are structurally equivalent to their singleton type-1 counterpart. This has provided us with a framework in which all the controllers are as similar to each other as possible. We have carried out comparative experiments with different levels of noise in the controller's inputs. Different levels of uncertainty factors have also been defined in the controllers during their design.

The results have shown that the traditional singleton type-1 FLS provides a satisfactory performance in the absence of noise. This is the simplest type of system and therefore, it should be applied when possible. However, it is almost impossible to avoid uncertainties in real-world applications and more flexible methods capable of handling them are needed. Our study has shown that the non-singleton type-1 FLS and the singleton interval type-2 FLS can adequately handle noise in the system's inputs to a certain degree. Specifically, it has been observed that the former presents a better performance for lower and medium levels of noise, whilst the latter is preferred to handle larger amounts of uncertainty. These results are important, as they address one of the main criticisms received by type-2 FLSs. They have allowed us to clarify that the reason for the better performance of interval type-2 FLSs under large amounts of uncertainty is not their use of extra parameters, but rather their different way of handling such uncertainty.

For future work, we aim to further extend the capabilities of the proposed evolving methodologies, according to the issues mentioned before. Some open questions that can be addressed are: (i) Increase of the controller's autonomy, e.g., by automatically determining the sign of the monotonicity of the plant with respect to the control signal, from the plant's response to the control actions. (ii) Design of input selection mechanisms for OSENF-TaSe. (iii) Incorporation of new soft-computing techniques to the different steps of the evolving procedure, e.g., new methods for parameter learning, advanced clustering techniques for the location of the rule centers, etc. In addition, we aim to continue the study of the properties of type-2 FLSs, with the objective of developing evolving techniques for their automatic design.

7.2 List of publications

This section gathers a list of the publications in which the results presented in this dissertation have appeared. These publications have been classified into three groups: The first group is formed by articles that have been published in scientific journals. Next, we include the works presented in international conferences. Finally, we present a series of works that are not directly related to the topic of this dissertation but that have been developed in parallel with it and have contributed to my training in Computer Architecture and Computer Technology.

Journal publications

1. Cara, A. B., Herrera, L. J., Pomares, H., and Rojas, I. (2012).New online self-evolving neuro-fuzzy controller based on the TaSe-NF model. *Information Sciences.* Accepted with minor revision.

    - Impact factor (JCR 2010): 2.836.
    - Ranking per subject category:
        - Computer Science, Information Systems: 10/128 (Q1)

2. Cara, A. B., Pomares, H., and Rojas, I. (2011). A new methodology for the online adaptation of fuzzy self-structuring controllers. *IEEE Transactions on Fuzzy Systems.*, 19(3):449–464.

    - Impact factor (JCR 2010): 2.695.
    - Ranking per subject category:
        - Computer Science, Artificial Intelligence: 14/108 (Q1)
        - Engineering, Electrical and Electronic: 17/247 (Q1)
    - Times cited: 1.

3. Cara, A. B., Pomares, H., Rojas, I., Lendek, Zs., and Babuška, R. (2010). Online self-evolving fuzzy controller with global learning capabilities. *Evolving systems*, 1(4):225–239.

    - Times cited: 2.

International conferences

1. Cara, A. B., Rojas, I., Pomares, H., Wagner, C., and Hagras, H. (2011). On comparing non-singleton type-1 and singleton type-2 fuzzy controllers for a nonlinear servo system. In *2011 IEEE Symposium on Advances in Type-2 Fuzzy Logic Systems (T2FUZZ)*, pages 126–133.

2. Herrera, L. J., Ghinea, R. I., Cara, A. B., Paravina, R. D., Pérez, M. M. (2011). On Color Threshold Calculations In Oral Tissues And Dental Materials. *Histology And Histopathology*, vol 26, num. SUPL 1, pages. 306–306.

3. Cara, A. B., Lendek, Zs., Babuška, R., Pomares, H., and Rojas, I. (2010). Online self-organizing adaptive fuzzy controller: application to a nonlinear servo system. In *Proc. 2010 IEEE World Congress on Computational Intelligence*, pages 2491–2498.

4. Cara, A. B., Pomares, H., and Rojas, I. (2010). An algorithm for online self-organization of fuzzy controllers. *Lecture Notes in Artificial Intelligence*, volume 6097, pages 212–221. Springer Berlin Heidelberg.

Other publications

1. Pomares, H., Rojas, I., Guillén, A., González, J., Valenzuela, O., Florido,J. P., Urquiza, J. M, Cara, A. B., Lopez-Mansilla, L., Egea-Serrano, S. (2011). Desarrollo de un entorno integrado para un computador didáctico elemental para la asignatura de Fundamentos de Informática del nuevo grado en Ingeniería de Tecnologías de Telecomunicaciones. In *Enseñanza y aprendizaje de Ingeniería de Computadores*, vol. 1, pages 43–49.

2. Pomares, H., Rojas, I., Guillén, A., Herrera, L.J., Rubio, G., Florido, J.P., Urquiza, J.M, Cara, A.B., Lopez-Mansilla, L., Egea-Serrano, S. (2009). Implementation of a didactic interpreter for CODE-2. In *V International Conference on Multimedia and Information and Communication Technologies in Education Research, Lisbon, Portugal. Reflections and Innovations in Integrating ICT in Education*, Vol.2. pages 960-964.

3. Pomares, H., Garcia-Garcia, C., Rojas, I., Damas, M., González, J., Florido, J.P., Urquiza, J.M, Cara, A.B., Lopez-Mansilla, L., Egea-Serrano, S. (2009). Teaching Digital Systems Design with a New Didactic Environment through the Internet. In *V Internacional Conference on Multimedia and Information and Communication Technologies in Education Research, Lisbon, Portugal. Reflections and Innovations in Integrating ICT in Education*, Vol.2. pages 1021-1026.

4. Cara, A.B., Moreno, C., Cañas, A. (2008). Virtual whiteboard and chat for a learning management system. In *Proceedings of the 2008 IADIS International Conference on E-Learning*, Vol. 1, pages 263-270.

5. Cara, A.B., Moreno, C., Cañas, A. (2008). Pizarra virtual y chat para una plataforma de teleformación. In *Actas de las XIV Jornadas de Enseñanza Universitaria de la Informática*, pages 619-626.

Conclusiones de la tesis y lista de publicaciones

Este capítulo contiene la versión en español de las conclusiones presentadas en el capítulo 7 y ha sido incluido para cumplir con los requisitos necesarios para optar a la mención de *Doctorado europeo*.

Conclusiones y aportaciones

El trabajo presentado en esta tesis doctoral es una contribución a las áreas de sistemas difusos auto-organizativos y control inteligente. En esta sección se resumen las principales aportaciones y las conclusiones extraídas del trabajo de investigación desarrollado.

En esta tesis se han propuesto dos nuevas metodologías para el diseño de controladores difusos auto-organizativos en tiempo real, cuando no existe conocimiento previo sobre la planta a controlar. Estas metodologías proporcionan mecanismos para la obtención de controladores difusos cuando la falta de información no permite aplicar otros métodos. Las principales características presentes en ambos métodos se pueden resumir del siguiente modo:

- *No es necesario disponer de un modelo de la planta ni de sus ecuaciones diferenciales*. Es más, los métodos propuestos no realizan hipótesis sobre las propiedades de dichas ecuaciones y tan sólo utilizan información cualitativa sobre la planta (concretamente, el signo de la monotonía de la planta con respecto a la señal de control).

- *El controlador difuso se diseña desde cero, a la vez que se controla la planta.* Para ello, tan sólo se utiliza la información de entrada/salida obtenida en tiempo real de la propia operación de la planta.

- *La estructura del controlador se obtiene de forma incremental, explotando la propiedad de aproximación universal de los sistemas difusos.* De este modo, se proporcionan políticas de control cuya precisión aumenta conforme avanza el tiempo de operación. La información utilizada en este proceso cubre todo el rango de operación de la planta, lo que proporciona mayor robustez al proceso auto-organizativo.

- *Los controladores desarrollados son robustos*, en el sentido de que pueden responder a cambios inesperados en la dinámica de la planta. Su naturaleza adaptativa les dota de la capacidad de aprender el nuevo comportamiento de la planta para así continuar operando satisfactoriamente.

- *El método auto-organizativo es un proceso continuado de aprendizaje.* Por tanto, el controlador continúa aprendiendo y evolucionando durante toda su operación. El aprendizaje actúa en dos niveles: por una parte, en la fase de ajuste de parámetros se utiliza la información referente al error en la salida de la planta para adaptar los consecuentes de las reglas. Por otra parte, se analiza el error en la aproximación de la función inversa de la planta para desarrollar incrementalmente la estructura del controlador.

La primera metodología propuesta es OSEFC (véase el capítulo 4). En este método se hace uso de un conjunto completo de reglas, por lo que la adición de una nueva función de pertenencia implica la creación de todas las reglas que la usan como antecedente. Para OSEFC se han propuesto dos mecanismos de ajuste de parámetros, basados en aprendizaje local y global, respectivamente. El primer método es un proceso de adaptación basado en la reducción del error actual en la salida de la planta (por tanto, en información local). Este método aplica una política de recompensa/penalización para desplazar los consecuentes en la dirección que ayuda a reducir dicho error. El segundo mecanismo propuesto hace uso de la información global de entrada/salida proporcionada por la propia planta para conseguir un ajuste fino de los consecuentes de las reglas.

Concretamente, se aplica una técnica de optimización basada en gradiente para minimizar el error presente en la salida del controlador y así, de forma indirecta, reducir el error en la salida de la planta. En este caso, el método de aprendizaje local se utiliza a modo de supervisor, con el objetivo de garantizar que los cambios aplicados no comprometen la calidad del control desde un punto de vista local.

Como se ha comentado, la evolución de la estructura de OSEFC se basa en la mejora sucesiva de la aproximación de la función inversa de planta realizada por el controlador. En este sentido, se analiza el error de aproximación a lo largo de todo el rango de operación del sistema para localizar la variable de entrada con mayor responsabilidad en dicho error. Dicha variable recibe una nueva función de pertenencia y, como resultado, todas las reglas asociadas a ella se añaden a la base de reglas del controlador. Este proceso permite a OSEFC detectar cuáles son las variables más relevantes para el proceso de control, dotándole de la capacidad de actuar como un selector de entradas automático.

Las principales propiedades de esta metodología han sido mostradas a través de múltiples experimentos, incluyendo el control de un sistema real. Los resultados obtenidos con plantas de complejidad creciente han demostrado que OSEFC es capaz de aprender la política de control de forma efectiva, a pesar de comenzar el control con topologías vacías. Igualmente, se ha mostrado que es capaz de manejar entradas ruidosas y afrontar cambios inesperados en la dinámica de las plantas, mejorando la respuesta de un controlador difuso tradicional (no adaptativo) en este último caso. Sin embargo, el uso de un conjunto completo de reglas hace que OSEFC padezca el problema conocido como la maldición de la dimensionalidad, inherente a la distribución de las reglas en rejilla. Como consecuencia, la complejidad del sistema aumenta y se degrada la interpretabilidad del controlador.

Con el objetivo de afrontar este problema, se ha propuesto una segunda metodología, denominada OSENF-TaSe (véase el capítulo 5). En esta nueva metodología, se ha optado por una distribución “en clustering” de la base de reglas, que permite ubicar las reglas de forma no uniforme a lo largo del espacio de entrada. De este modo, es posible colocar las reglas directamente en las zonas donde las alinealidades son más fuertes. Por otra parte, esta metodología se basa en el modelo TaSe-NF, cuya principal característica es la preservación de la interpretabilidad de las reglas (o sub-modelos locales). Esto es posible gracias al uso de un mecanismo de inferencia difusa modificado que garantiza que la salida global del controlador en el punto asociado al centro de cada regla coincide exactamente con el consecuente correspondiente. De este modo, cada consecuente tiene un significado que está directamente relacionado con la salida del controlador.

Al igual que en OSEFC, se utiliza el mecanismo local de aprendizaje para adaptar los consecuentes y garantizar la respuesta a corto plazo del controlador. Por otra parte, la estructura del controlador se refina de modo incremental por medio de la subdivisión de aquellas reglas que presentan un rendimiento insatisfactorio en sus áreas de influencia. Una diferencia importante entre OSEFC y OSENF-TaSe es que éste último tan sólo añade una regla en cada cambio estructural, lo que ayuda a mantener el crecimiento del número de reglas bajo control. Por último, OSENF-TaSe incluye un método de descenso en gradiente para obtener una mejor estimación de los parámetros del controlador (concretamente, de los centros y radios de los antecedentes y de los consecuentes).

Las simulaciones presentadas en el capítulo 5 han confirmado que OSENF-TaSe reduce la complejidad del controlador difuso con respecto a OSEFC. Del mismo modo, se ha mostrado la mejora en la interpretabilidad de las reglas en comparación con un controlador difuso basado en clustering tradicional. Por último, se ha comparado la respuesta de OSENF-TaSe con la de un método de control clásico que se ha diseñado haciendo uso de las ecuaciones diferenciales de la planta. Los resultados han mostrado que OSENF-TaSe puede controlar la planta con una calidad equivalente a la del controlador clásico y que, de hecho, lo supera cuando se produce un cambio inesperado en la dinámica interna de la planta.

Todos estos resultados demuestran que OSEFC y OSENF-TaSe son alternativas válidas para el diseño automático de controladores difusos cuando no existe información previa sobre la planta. No obstante, esta falta de información provoca algunas limitaciones. En primer lugar, puesto que los controladores se auto-diseñan en tiempo real a la vez que controlan la planta, los métodos propuestos no se pueden aplicar directamente en plantas con estados críticos que no deben alcanzarse nunca. Sin embargo, en estos casos es posible aplicar métodos de pre-entrenamiento para conseguir un controlador inicial lo suficientemente estable con el que iniciar el proceso auto-organizativo. En segundo lugar, la aplicación de estos métodos requiere que el período de muestreo del controlador se pueda definir de forma que la salida de la planta dependa directamente de la señal de control aplicada en el instante anterior. Por último, la falta de un modelo matemático hace que sea prácticamente imposible proporcionar una demostración formal de la estabilidad de estos controladores, si bien este aspecto ha sido abordado desde una perspectiva cualitativa en este trabajo. A pesar de lo anterior, las metodologías propuestas establecen un marco de trabajo inicial para el desarrollo automático de controladores con el que alcanzar el objetivo final de proporcionar un control inteligente sin ningún tipo de conocimiento previo.

Por último, la parte final de la tesis se ha dedicado al estudio comparativo de sistemas difusos de tipo 1 y 2 (véase el capítulo 6). El objetivo de dicho estudio ha sido intentar responder a dos de las cuestiones que con mayor frecuencia se plantean en el ámbito de los sistemas difusos de tipo 2: en primer lugar, ¿existen diferencias significativas en el rendimiento de los distintos paradigmas difusos? Y, en segundo lugar, ¿a partir de qué nivel de incertidumbre es un método preferible al otro?

Con este objetivo en mente, se ha presentado un algoritmo evolutivo multiobjetivo para la optimización de la base de reglas y los parámetros de sistemas difusos de tipo 1 y 2 (en concreto, para sistemas difusos de tipo 2 intervalo-valorados). Este algoritmo evolutivo proporciona un conjunto de sistemas Pareto-óptimos con respecto a los objetivos contrapuestos de complejidad y precisión. El método de optimización propuesto incluye un módulo para el cálculo de los consecuentes óptimos para un conjunto de reglas dado, a partir de datos de entrada/salida. El uso de los consecuentes óptimos mejora la precisión de las soluciones obtenidas por el algoritmo evolutivo, a la vez que reduce el espacio de búsqueda. La aplicación de este método evolutivo permite obtener un marco adecuado para llevar a cabo la comparación mencionada, pues proporciona un conjunto de sistemas difusos que han sido optimizados en las mismas condiciones.

En concreto, se han comparado sistemas difusos de tipo 1 (con entrada numérica y con entrada difusa) con sistemas de tipo 2 intervalo-valorados, en dos contextos diferentes:

- En primer lugar, el método evolutivo propuesto se ha aplicado para optimizar los tres tipos de sistema difuso en varios problemas de aproximación funcional, bajo distintos niveles de ruido en los datos de entrada. Los frentes Pareto obtenidos se han comparado utilizando un indicador de calidad basado en el hiper-volumen de los frentes. Finalmente, se ha empleado un test estadístico no paramétrico para determinar si las diferencias detectadas son significativas o no.

- En segundo lugar, se ha estudiado una aplicación de ejemplo consistente en el control de un servomotor no lineal. En este caso, se ha presentado un entorno en el que los tres tipos de controladores presentan la misma base de reglas y los mismos consecuentes. El controlador de tipo 1 con entrada numérica se ha obtenido de forma automática, aplicando el método OSEFC propuesto en esta tesis, mientras que el controlador de tipo 1 con entrada difusa y el controlador de tipo 2 intervalo-valorado se han obtenido a partir de éste, aplicando los mismos factores de incertidumbre en ambos casos. De este modo, se ha conseguido que los tres controladores sean lo más similares posible. En este contexto, se han realizado experimentos con distintos niveles de ruido en las entradas del controlador y con distintos factores de incertidumbre en el diseño de los controladores.

Los resultados obtenidos han mostrado que, en ausencia de ruido, el sistema difuso de tipo 1 con entrada numérica proporciona un rendimiento satisfactorio. Este tipo de sistema es el más simple y, por tanto, debería ser utilizado siempre que sea posible. Sin embargo, hay que tener en cuenta que en las aplicaciones del mundo real es prácticamente imposible evitar la incertidumbre y que, por tanto, se necesitan métodos más flexibles que sean capaces de manejarla. Por otra parte, nuestro análisis ha mostrado que tanto los sistemas de tipo 1 con entradas difusas como los de tipo 2 son capaces de manejar adecuadamente el ruido hasta un cierto nivel. Concretamente, se ha observado que los primeros responden mejor ante niveles de ruido bajos y medios, mientras que los segundos aceptan niveles de ruido mayores. Estos resultados son relevantes, pues abordan una de las críticas que con mayor frecuencia se dirigen a los sistemas difusos de tipo 2. Así, nos han permitido aclarar que su mejor rendimiento ante altos niveles de incertidumbre no se debe al uso de más parámetros, sino a su mecanismo para manejar dicha incertidumbre.

Como trabajo futuro, se pretende extender las capacidades de las metodologías auto-organizativas propuestas, de acuerdo con los aspectos mencionados anteriormente. Algunas de las cuestiones abiertas que se pueden abordar son: (i) Aumento de la autonomía del controlador; por ejemplo, mediante la detección automática del signo de la monotonía de la planta con respecto a la señal de control, a partir de la respuesta de la planta. (ii) Diseño de mecanismos de selección de entradas para OSENF-TaSe. (iii) Incorporación de nuevas técnicas de *soft-computing* en los distintos pasos del proceso auto-organizativo; por ejemplo, nuevos métodos para el aprendizaje de parámetros, técnicas de agrupamiento (*clustering*) para la localización de los centros de las funciones de pertenencia y reglas, etc. Igualmente, se pretende continuar el estudio de las propiedades de los sistemas difusos de tipo 2, con el objetivo de desarrollar técnicas auto-organizativas para su diseño automático.

Lista de publicaciones

En esta sección se recoge una lista de las publicaciones en las que han aparecido los distintos resultados presentados en esta tesis. Las publicaciones se han agrupado en tres categorías, correspondientes a revistas científicas, congresos internacionales y, por último, otras publicaciones que no están directamente relacionadas con los temas tratados en esta tesis pero que han contribuido a mi formación en el área de la Arquitectura y Tecnología de Computadores.

Publicaciones en revistas científicas

1. Cara, A. B., Herrera, L. J., Pomares, H., and Rojas, I. (2012).New online self-evolving neuro-fuzzy controller based on the TaSe-NF model. *Information Sciences.* Aceptado con revisión menor.

- Índice de impacto (JCR 2010): 2.836.
- Clasificación según la categoría:
  - Computer Science, Information Systems: 10/128 (Q1)

2. Cara, A. B., Pomares, H., and Rojas, I. (2011). A new methodology for the online adaptation of fuzzy self-structuring controllers. *IEEE Transactions on Fuzzy Systems.*, 19(3):449–464.
   - Índice de impacto (JCR 2010): 2.695.
   - Clasificación según la categoría:
     - Computer Science, Artificial Intelligence: 14/108 (Q1)
     - Engineering, Electrical and Electronic: 17/247 (Q1)
   - Número de veces que ha sido citado: 1.

3. Cara, A. B., Pomares, H., Rojas, I., Lendek, Zs., and Babuška, R. (2010). Online self-evolving fuzzy controller with global learning capabilities. *Evolving systems*, 1(4):225–239.
   - Número de veces que ha sido citado: 2.

Congresos internacionales

1. Cara, A. B., Rojas, I., Pomares, H., Wagner, C., and Hagras, H. (2011). On comparing non-singleton type-1 and singleton type-2 fuzzy controllers for a nonlinear servo system. In *2011 IEEE Symposium on Advances in Type-2 Fuzzy Logic Systems (T2FUZZ)*, pages 126–133.

2. Herrera, L. J., Ghinea, R. I., Cara, A. B., Paravina, R. D., Pérez, M. M. (2011). On Color Threshold Calculations In Oral Tissues And Dental Materials. *Histology And Histopathology*, vol 26, num. SUPL 1, pages. 306–306.

3. Cara, A. B., Lendek, Zs., Babuška, R., Pomares, H., and Rojas, I. (2010). Online self-organizing adaptive fuzzy controller: application to a nonlinear servo system. In *Proc. 2010 IEEE World Congress on Computational Intelligence*, pages 2491–2498.

4. Cara, A. B., Pomares, H., and Rojas, I. (2010). An algorithm for online self-organization of fuzzy controllers. *Lecture Notes in Artificial Intelligence*, volume 6097, pages 212–221. Springer Berlin Heidelberg.

Otras publicaciones

1. Pomares, H., Rojas, I., Guillén, A., González, J., Valenzuela, O., Florido,J. P., Urquiza, J. M, Cara, A. B., Lopez-Mansilla, L., Egea-Serrano, S. (2011). Desarrollo de un entorno integrado para un computador didáctico elemental para la asignatura de Fundamentos de Informática del nuevo grado en Ingeniería de Tecnologías de Telecomunicaciones. In *Enseñanza y aprendizaje de Ingeniería de Computadores*, vol. 1, pages 43–49.

2. Pomares, H., Rojas, I., Guillén, A., Herrera, L.J., Rubio, G., Florido, J.P., Urquiza, J.M, Cara, A.B., Lopez-Mansilla, L., Egea-Serrano, S. (2009). Implementation of a didactic interpreter for CODE-2. In *V International Conference on Multimedia and Information and Communication Technologies in Education Research, Lisbon, Portugal. Reflections and Innovations in Integrating ICT in Education*, Vol.2. pages 960-964.

3. Pomares, H., Garcia-Garcia, C., Rojas, I., Damas, M., González, J., Florido, J.P., Urquiza, J.M, Cara, A.B., Lopez-Mansilla, L., Egea-Serrano, S. (2009). Teaching Digital Systems Design with a New Didactic Environment through the Internet. In *V Internacional Conference on Multimedia and Information and Communication Technologies in Education Research, Lisbon, Portugal. Reflections and Innovations in Integrating ICT in Education*, Vol.2. pages 1021-1026.

4. Cara, A.B., Moreno, C., Cañas, A. (2008). Virtual whiteboard and chat for a learning management system. In *Proceedings of the 2008 IADIS International Conference on E-Learning*, Vol. 1, pages 263-270.

5. Cara, A.B., Moreno, C., Cañas, A. (2008). Pizarra virtual y chat para una plataforma de teleformación. In *Actas de las XIV Jornadas de Enseñanza Universitaria de la Informática*, pages 619-626.

Appendix A

Description of the plants used in this dissertation

The methodologies presented in this dissertation do not use the information about the differential equations of the plants. For this reason, the plants used for experimentation have only been described from a qualitative point of view. However, their mathematical models are gathered in this appendix for the interested reader.

A.1 Tank of liquid

The plant known as *tank of liquid* or *water tank* [Ogata, 2001] (see Fig. 4.12) represents a tank with a valve that controls the introduction of liquid, and a vent in its lower side that lets the liquid out. The amount of liquid that gets in the tank is proportional to the voltage v applied to the entrance valve, whilst the liquid that flows out is proportional to the square root of the level reached by the liquid inside. The change in the amount of liquid in the tank is equal to the difference between the amount of liquid getting in and the amount of liquid getting out. Thus, the nonlinear differential equation that defines the level of liquid H stored in the tank is given by

$$\frac{dVol}{dt} = A\frac{dH}{dt} = bv - a\sqrt{H} \tag{A.1}$$

where Vol is the volume of liquid inside the tank (measured in m^3), A is the area of the transversal section of the tank, b is a constant associated to the liquid

TABLE A.1: Parameters of the tank of liquid

Symbol	Parameter	Value
A	Area of the section of the tank	$10\ \mathrm{m}^2$
a	Constant for the exit rate	$1\ \mathrm{m}^{5/2} \cdot \mathrm{s}^{-1}$
b	Constant for the entrance rate	$2.5\ \mathrm{m}^3 \cdot (\mathrm{Vs})^{-1}$

entrance rate, a is a constant related to the liquid exit rate. The level of liquid is measured in meters.

The control objective for this problem is to adjust the power of the liquid entrance so as to achieve a determined height in the liquid level. The voltage applied to the plant can vary in the range $[0, 5]$ V. Note that negative control signals are not allowed, which means that the only way to decrease the level of liquid in the tank is to stop the liquid entrance (i.e., close the valve) and wait for the liquid to flow out.

Table A.1 shows the values of the parameters used for the experiment presented in section 4.4.2.1.

A.2 Mechanical suspension system

In this section, we describe the plant known as *mechanical suspension system* (see Fig. 4.14), whose model was provided by the Delft Center for Systems and Control (Delft University of Technology, The Netherlands). This model is formed by a sprung mass M_1 that represents a body, and an unsprung mass M_2 representing the mechanical components whose duty is to move the body. Between these two elements there is a transfer system, which is characterized by a spring K_1 and a damper D. Finally, a spring K_2 serves as a model of the compressibility of the mechanical components.

The state of the system is given by $x = [v_1, f_1, v_2, f_2]$, where v_1 is the body vertical velocity, f_1 is the force associated with the transfer system, v_2 is the vertical velocity of the mechanical components and f_2 is the force associated with them. The velocities are measured in meters per second and the forces are measured in

Newtons. In this case, the control objective is to make the output variable follow a desired trajectory r.

The dynamics of the system can be modeled by the following nonlinear differential equations:

$$\begin{aligned}
\dot{v}_1 &= \frac{1}{M_1}(-f_1 - D(v_1 - v_2)) \\
\dot{f}_1 &= \frac{cosh^2(\gamma f_1)}{\gamma}(v_1 - v_2) \\
\dot{v}_2 &= \frac{1}{M_2}(f_1 + D(v_1 - v_2)) - f_2 \\
\dot{f}_2 &= K_2(v_2 - u) \\
y &= v_1
\end{aligned} \tag{A.2}$$

where u is the control signal, y is the output of the system and γ is a constant associated with the transfer system. The values of all the parameters are given in Table A.2.

TABLE A.2: Parameters of the suspension system

Symbol	Parameter	Value
M_1	Mass of the moving body	305 kg
M_2	Mass of the mechanical components	38 kg
γ	Constant for the transfer system	$38 \cdot 10^{-6}$ m $\cdot$ N^{-1}
K_2	Compressibility of mechanical components	$231 \cdot 10^{3}$ N $\cdot$ m^{-1}
D	Damper	6000 N $\cdot$ s $\cdot$ m^{-1}

A.3 1-DOF Helicopter setup

In this section, we describe a simple 1-DOF helicopter setup (see Fig. 4.17), also from the Delft Center for Systems and Control. This setup consists of a beam attached to a fixed pole. The beam can freely rotate in the vertical plane while the horizontal position is fixed. At the end of the beam there is a DC motor with a propeller attached. The control input u is the voltage applied to the motor to control the elevation angle of the beam. It takes values in the range $[-1, 1]$, where -1 represents the maximal voltage (in absolute value) that makes the propeller and the beam rotate in the negative sense of α, and +1 is the maximal voltage that causes rotation of the beam in the positive sense of α. There are two measured outputs, namely the angular velocity ω of the propeller and the angle α of the beam. The former takes values in the range [-1,1], representing the maximal negative and positive angular velocity, respectively. The latter is measured in radians. The control objective is to make the angle α follow a specific reference trajectory in the range [-1,1].

The nonlinear differential equations that govern this plant are the following:

$$\begin{aligned} \tau\dot{\omega} + \omega &= K_1 u \\ \ddot{\alpha} + b\dot{\alpha} + K_2 \sin\alpha &= K_3\omega \end{aligned} \tag{A.3}$$

Table A.3 summarizes the meaning and values of all the parameters involved.

TABLE A.3: Parameters of the helicopter setup

Symbol	Parameter	Value
τ	Time constant of the motor	0.3 s
b	Damping (viscous friction) of the beam's motion	$0.5491\ \mathrm{s}^{-1}$
K_1	Gain control signal-propeller's velocity	$1.3\ \mathrm{V}\cdot\mathrm{s}\cdot\mathrm{rad}^{-1}$
K_2	Constant of influence of the gravity force	$0.8374\ \mathrm{rad}\cdot\mathrm{s}^{-2}$
K_3	Gain propeller's velocity-beam's ang. acceleration	0.75 s

A.4 Nonlinear servo system

In this section, we present the experimental setup known as *nonlinear servo system*, from the Delft Center for Systems and Control (Delft University of Technology, The Netherlands). This system consists of a weight attached to a disk that is actuated by a DC motor and rotates in a vertical plane, as depicted in Fig. 4.22. The presence of the extra weight introduces a gravity term that causes the system's nonlinear behavior. Thus, the control objective is to compensate this nonlinearity in order to make the position of the disk (measured as an angle θ) track a desired reference trajectory r.

The mathematical model of the plant is defined by

$$\begin{aligned} \dot{\mathbf{x}} &= f(\mathbf{x}, u) = A\mathbf{x} + Bu + G \\ &= \begin{bmatrix} 0 & 1 \\ 0 & \frac{-K^2-bR}{RJ} \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ \frac{K}{RJ} \end{bmatrix} u - \begin{bmatrix} 0 \\ \frac{mgl}{J}\sin(\theta) \end{bmatrix} \end{aligned} \tag{A.4}$$

where $\mathbf{x} = \left[\theta, \, \dot{\theta}\right]^T \in \mathbb{R}^2$ is the state of the plant, given by the angle θ (measured in radians) and the angular velocity $\dot{\theta}$ (measured in rad/s); u is the control signal, and $G = \left(-\frac{mgl}{J}\sin(\theta)\right)$ is the gravity term. The control signal is the voltage applied to the plant, whose values range in $[-10, 10]$ V.

Table A.4 gathers the meaning and value of all the parameters involved. It has to be noted that these parameters correspond to the configuration of the real experimental setup.

TABLE A.4: Parameters of the nonlinear servo system

Symbol	Parameter	Value
J	Moment of inertia of the rotor	$1.91 \cdot 10^{-4}$ kg $\cdot$ m^2
b	Damping of the mechanical system	$3 \cdot 10^{-6}$ kg $\cdot$ s^{-1}
K	Electromotive force constant	0.0536 N $\cdot$ m $\cdot$ A^{-1}
R	Electric resistance	9.5 Ω
m	Mass of the weight	0.055 kg
l	Length from the center of the disk to the center of mass	0.042 m
g	Gravity acceleration	9.81 m $\cdot$ s^{-2}

Appendix B

Additional tables of results for the multi-objective evolutionary algorithm

In this appendix, we present the numeric data corresponding to the experimentation presented in section 6.4.1.2 for the multi-objective optimization of type-1 and type-2 fuzzy systems. The following tables depict the mean values and standard deviations of the quality indicators obtained for each function, level of noise, and type of fuzzy system, i.e., the information represented in Fig. 6.8. Table B.1 shows the information obtained during the training process, whilst Table B.2 gathers the data obtained during test.

TABLE B.1: Average quality indicators of the Pareto fronts obtained by the MOEA for the three types of FLSs (training)

Function	FLS	Noise level (σ)								
		0	0.05	0.1	0.15	0.2	0.25	0.3	0.4	0.5
f_1	ST1	0.172 ±0.009	0.178 ±0.009	0.187 ±0.009	0.228 ±0.011	0.249 ±0.008	0.273 ±0.015	0.294 ±0.007	0.359 ±0.011	0.410 ±0.016
	NST1	0.133 ±0.010	0.150 ±0.005	0.185 ±0.005	0.217 ±0.008	0.233 ±0.005	0.272 ±0.006	0.296 ±0.006	0.369 ±0.006	0.409 ±0.003
	T2	0.166 ±0.009	0.175 ±0.010	0.181 ±0.010	0.223 ±0.017	0.243 ±0.011	0.260 ±0.016	0.287 ±0.021	0.326 ±0.044	0.388 ±0.034
f_{10}	ST1	0.126 ±0.022	0.158 ±0.023	0.185 ±0.025	0.228 ±0.036	0.266 ±0.029	0.287 ±0.043	0.353 ±0.032	0.386 ±0.044	0.444 ±0.077
	NST1	0.112 ±0.011	0.145 ±0.009	0.191 ±0.008	0.229 ±0.012	0.289 ±0.006	0.308 ±0.005	0.362 ±0.010	0.419 ±0.004	0.468 ±0.012
	T2	0.149 ±0.016	0.149 ±0.013	0.180 ±0.016	0.206 ±0.030	0.251 ±0.039	0.292 ±0.043	0.306 ±0.048	0.362 ±0.078	0.421 ±0.053
f_2	ST1	0.110 ±0.016	0.136 ±0.012	0.171 ±0.018	0.213 ±0.026	0.259 ±0.020	0.292 ±0.024	0.356 ±0.020	0.404 ±0.024	0.467 ±0.025
	NST1	0.096 ±0.008	0.134 ±0.008	0.190 ±0.008	0.227 ±0.004	0.282 ±0.005	0.333 ±0.006	0.375 ±0.006	0.421 ±0.010	0.460 ±0.009
	T2	0.107 ±0.009	0.136 ±0.010	0.176 ±0.013	0.212 ±0.021	0.243 ±0.021	0.303 ±0.030	0.329 ±0.040	0.381 ±0.023	0.444 ±0.055
f_4	ST1	0.339 ±0.023	0.365 ±0.011	0.416 ±0.009	0.459 ±0.014	0.483 ±0.018	0.518 ±0.017	0.520 ±0.036	0.549 ±0.017	0.567 ±0.011
	NST1	0.299 ±0.019	0.335 ±0.017	0.390 ±0.012	0.443 ±0.013	0.465 ±0.020	0.517 ±0.013	0.529 ±0.005	0.567 ±0.014	0.570 ±0.022
	T2	0.345 ±0.011	0.363 ±0.019	0.398 ±0.029	0.434 ±0.033	0.479 ±0.025	0.507 ±0.029	0.494 ±0.034	0.512 ±0.045	0.564 ±0.020

Continued on next page

Table B.1 Average quality indicators of the Pareto fronts (training)
– continued from previous page

Function	FLS	Noise level (σ)								
		0	0.05	0.1	0.15	0.2	0.25	0.3	0.4	0.5
f_5	ST1	0.235 ±0.013	0.241 ±0.021	0.251 ±0.013	0.266 ±0.017	0.274 ±0.017	0.288 ±0.021	0.306 ±0.016	0.343 ±0.013	0.382 ±0.014
	NST1	0.207 ±0.013	0.215 ±0.015	0.226 ±0.015	0.256 ±0.013	0.273 ±0.004	0.289 ±0.011	0.319 ±0.009	0.349 ±0.008	0.406 ±0.004
	T2	0.251 ±0.022	0.244 ±0.018	0.252 ±0.017	0.267 ±0.009	0.276 ±0.030	0.274 ±0.016	0.285 ±0.021	0.304 ±0.023	0.356 ±0.021
f_6	ST1	0.117 ±0.016	0.142 ±0.012	0.141 ±0.014	0.154 ±0.011	0.182 ±0.009	0.203 ±0.015	0.218 ±0.019	0.261 ±0.015	0.293 ±0.014
	NST1	0.118 ±0.016	0.136 ±0.009	0.141 ±0.011	0.174 ±0.011	0.184 ±0.011	0.219 ±0.006	0.230 ±0.006	0.276 ±0.007	0.301 ±0.007
	T2	0.128 ±0.011	0.132 ±0.011	0.141 ±0.014	0.157 ±0.011	0.166 ±0.010	0.179 ±0.022	0.201 ±0.015	0.217 ±0.027	0.274 ±0.021
f_7	ST1	0.166 ±0.014	0.171 ±0.014	0.178 ±0.018	0.199 ±0.017	0.210 ±0.013	0.223 ±0.014	0.255 ±0.012	0.275 ±0.014	0.314 ±0.013
	NST1	0.178 ±0.011	0.186 ±0.015	0.192 ±0.006	0.214 ±0.011	0.225 ±0.010	0.236 ±0.010	0.271 ±0.010	0.289 ±0.005	0.331 ±0.005
	T2	0.179 ±0.010	0.178 ±0.012	0.187 ±0.008	0.204 ±0.008	0.208 ±0.017	0.218 ±0.017	0.252 ±0.012	0.263 ±0.013	0.299 ±0.015
y_2	ST1	0.081 ±0.007	0.095 ±0.008	0.111 ±0.007	0.139 ±0.008	0.147 ±0.014	0.190 ±0.015	0.202 ±0.015	0.242 ±0.018	0.278 ±0.009
	NST1	0.067 ±0.007	0.083 ±0.005	0.106 ±0.004	0.135 ±0.002	0.159 ±0.005	0.194 ±0.006	0.214 ±0.006	0.249 ±0.004	0.321 ±0.003
	T2	0.094 ±0.011	0.098 ±0.019	0.099 ±0.011	0.124 ±0.024	0.164 ±0.009	0.182 ±0.020	0.204 ±0.017	0.236 ±0.021	0.249 ±0.025

Continued on next page

Table B.1 Average quality indicators of the Pareto fronts (training)
– continued from previous page

Function	FLS	Noise level (σ)								
		0	0.05	0.1	0.15	0.2	0.25	0.3	0.4	0.5
y_3	ST1	0.043 ±0.007	0.150 ±0.010	0.276 ±0.008	0.358 ±0.027	0.454 ±0.016	0.450 ±0.016	0.501 ±0.032	0.526 ±0.021	0.515 ±0.038
	NST1	0.045 ±0.005	0.169 ±0.012	0.298 ±0.006	0.393 ±0.005	0.446 ±0.004	0.497 ±0.014	0.544 ±0.016	0.569 ±0.002	0.581 ±0.002
	T2	0.078 ±0.022	0.157 ±0.018	0.234 ±0.028	0.345 ±0.046	0.424 ±0.061	0.418 ±0.057	0.505 ±0.040	0.514 ±0.047	0.531 ±0.042
y_5	ST1	0.141 ±0.006	0.198 ±0.011	0.264 ±0.009	0.347 ±0.011	0.388 ±0.014	0.405 ±0.018	0.444 ±0.021	0.518 ±0.015	0.531 ±0.032
	NST1	0.133 ±0.004	0.208 ±0.006	0.308 ±0.008	0.375 ±0.002	0.417 ±0.005	0.457 ±0.003	0.473 ±0.007	0.546 ±0.002	0.567 ±0.001
	T2	0.162 ±0.003	0.204 ±0.019	0.225 ±0.021	0.334 ±0.024	0.388 ±0.014	0.378 ±0.040	0.369 ±0.037	0.450 ±0.041	0.460 ±0.048

TABLE B.2: Average quality indicators of the Pareto fronts obtained by the MOEA for the three types of FLSs (validation)

Function	FLS	Noise level (σ)								
		0	0.05	0.1	0.15	0.2	0.25	0.3	0.4	0.5
f_1	ST1	0.183 ±0.009	0.187 ±0.008	0.184 ±0.009	0.188 ±0.010	0.196 ±0.006	0.203 ±0.010	0.209 ±0.005	0.223 ±0.012	0.277 ±0.020
	NST1	0.148 ±0.009	0.148 ±0.004	0.158 ±0.005	0.162 ±0.008	0.169 ±0.006	0.175 ±0.010	0.191 ±0.009	0.223 ±0.009	0.246 ±0.010
	T2	0.177 ±0.010	0.182 ±0.012	0.181 ±0.010	0.187 ±0.013	0.190 ±0.006	0.191 ±0.010	0.205 ±0.017	0.211 ±0.023	0.254 ±0.024
f_{10}	ST1	0.143 ±0.025	0.168 ±0.026	0.165 ±0.023	0.183 ±0.032	0.185 ±0.017	0.199 ±0.032	0.211 ±0.021	0.213 ±0.026	0.254 ±0.045
	NST1	0.129 ±0.013	0.139 ±0.009	0.141 ±0.010	0.151 ±0.010	0.167 ±0.009	0.181 ±0.008	0.188 ±0.009	0.229 ±0.014	0.286 ±0.017
	T2	0.167 ±0.018	0.159 ±0.016	0.169 ±0.016	0.171 ±0.027	0.185 ±0.021	0.203 ±0.028	0.187 ±0.025	0.215 ±0.043	0.231 ±0.031
f_2	ST1	0.116 ±0.018	0.126 ±0.013	0.123 ±0.015	0.141 ±0.018	0.151 ±0.014	0.149 ±0.010	0.195 ±0.019	0.227 ±0.021	0.272 ±0.018
	NST1	0.102 ±0.008	0.112 ±0.009	0.122 ±0.009	0.138 ±0.007	0.147 ±0.007	0.172 ±0.009	0.202 ±0.014	0.278 ±0.016	0.323 ±0.013
	T2	0.113 ±0.009	0.126 ±0.012	0.131 ±0.013	0.141 ±0.017	0.142 ±0.017	0.158 ±0.018	0.172 ±0.019	0.202 ±0.018	0.246 ±0.041
f_4	ST1	0.358 ±0.023	0.367 ±0.010	0.376 ±0.012	0.408 ±0.021	0.422 ±0.023	0.470 ±0.032	0.481 ±0.055	0.557 ±0.028	0.676 ±0.114
	NST1	0.315 ±0.020	0.327 ±0.019	0.356 ±0.024	0.372 ±0.024	0.391 ±0.033	0.459 ±0.009	0.465 ±0.019	0.538 ±0.019	0.711 ±0.065
	T2	0.365 ±0.009	0.365 ±0.019	0.359 ±0.026	0.386 ±0.029	0.417 ±0.020	0.446 ±0.036	0.438 ±0.020	0.504 ±0.050	0.587 ±0.040

Continued on next page

Table B.2 Average quality indicators of the Pareto fronts (validation)
– continued from previous page

Function	FLS	Noise level (σ)								
		0	0.05	0.1	0.15	0.2	0.25	0.3	0.4	0.5
f_5	ST1	0.243 ±0.013	0.250 ±0.023	0.249 ±0.014	0.256 ±0.023	0.240 ±0.019	0.256 ±0.028	0.256 ±0.019	0.271 ±0.017	0.282 ±0.018
	NST1	0.213 ±0.014	0.209 ±0.015	0.211 ±0.016	0.224 ±0.015	0.228 ±0.004	0.235 ±0.014	0.240 ±0.014	0.250 ±0.012	0.266 ±0.011
	T2	0.259 ±0.022	0.255 ±0.019	0.257 ±0.019	0.266 ±0.011	0.260 ±0.026	0.254 ±0.015	0.252 ±0.017	0.260 ±0.013	0.277 ±0.013
f_6	ST1	0.123 ±0.014	0.143 ±0.012	0.126 ±0.015	0.131 ±0.015	0.140 ±0.008	0.144 ±0.016	0.149 ±0.016	0.163 ±0.013	0.157 ±0.013
	NST1	0.125 ±0.015	0.136 ±0.009	0.128 ±0.011	0.144 ±0.011	0.135 ±0.013	0.151 ±0.007	0.151 ±0.008	0.157 ±0.011	0.174 ±0.011
	T2	0.134 ±0.010	0.133 ±0.012	0.134 ±0.011	0.140 ±0.009	0.136 ±0.008	0.133 ±0.015	0.143 ±0.011	0.143 ±0.010	0.154 ±0.014
f_7	ST1	0.171 ±0.014	0.178 ±0.016	0.176 ±0.020	0.183 ±0.018	0.184 ±0.012	0.190 ±0.014	0.200 ±0.011	0.198 ±0.012	0.221 ±0.018
	NST1	0.190 ±0.011	0.193 ±0.017	0.188 ±0.005	0.199 ±0.014	0.190 ±0.012	0.198 ±0.014	0.216 ±0.014	0.214 ±0.008	0.221 ±0.009
	T2	0.185 ±0.011	0.185 ±0.012	0.187 ±0.008	0.193 ±0.008	0.187 ±0.013	0.193 ±0.010	0.198 ±0.007	0.195 ±0.009	0.222 ±0.008
y_2	ST1	0.083 ±0.007	0.093 ±0.008	0.091 ±0.007	0.102 ±0.008	0.093 ±0.009	0.120 ±0.013	0.116 ±0.008	0.131 ±0.012	0.147 ±0.007
	NST1	0.070 ±0.007	0.077 ±0.005	0.078 ±0.005	0.086 ±0.004	0.090 ±0.007	0.117 ±0.007	0.114 ±0.008	0.136 ±0.006	0.149 ±0.006
	T2	0.095 ±0.010	0.096 ±0.019	0.085 ±0.008	0.094 ±0.019	0.111 ±0.009	0.113 ±0.014	0.120 ±0.013	0.126 ±0.011	0.132 ±0.013

Continued on next page

Table B.2 Average quality indicators of the Pareto fronts (validation)
– continued from previous page

Function	FLS	Noise level (σ)								
		0	0.05	0.1	0.15	0.2	0.25	0.3	0.4	0.5
y_3	ST1	0.043 ±0.007	0.069 ±0.010	0.132 ±0.004	0.175 ±0.010	0.235 ±0.007	0.274 ±0.013	0.305 ±0.037	0.438 ±0.023	0.557 ±0.031
	NST1	0.046 ±0.005	0.084 ±0.008	0.129 ±0.010	0.198 ±0.006	0.252 ±0.007	0.326 ±0.004	0.366 ±0.020	0.463 ±0.012	0.610 ±0.011
	T2	0.079 ±0.022	0.086 ±0.014	0.119 ±0.018	0.167 ±0.021	0.214 ±0.038	0.238 ±0.042	0.290 ±0.025	0.397 ±0.052	0.496 ±0.057
y_5	ST1	0.149 ±0.005	0.166 ±0.011	0.178 ±0.007	0.206 ±0.006	0.236 ±0.008	0.260 ±0.010	0.300 ±0.012	0.388 ±0.014	0.415 ±0.043
	NST1	0.141 ±0.004	0.166 ±0.005	0.179 ±0.015	0.215 ±0.004	0.247 ±0.007	0.293 ±0.009	0.317 ±0.003	0.414 ±0.007	0.400 ±0.006
	T2	0.170 ±0.005	0.177 ±0.017	0.164 ±0.011	0.199 ±0.012	0.231 ±0.007	0.230 ±0.019	0.248 ±0.024	0.302 ±0.022	0.337 ±0.034

Appendix C

High-performance computing architecture for the MOEA

Parallel architectures have been employed for the experimentation related to the multi-objective optimization of type-1 and type-2 FLS (see chapte 6), thus providing a high-performance computing environment. As it is well known, MOEAs are generally costly both in computation time and resources (mainly memory space) [Coello et al., 2007; Zhang and Li, 2007]. For this reason we have used the computer cluster BIOATC, available at the Department of Computer Architecture and Computer Technology of the University of Granada. In this appendix, we describe the configuration and main properties of this cluster.

The cluster BIOATC is made up of 19 hosts and provides up to 304 CPUs and 304 GB of memory. Fig. C.1 depicts the configuration of the cluster and the way it has been used to distribute the execution of the experiments. The main node is in charge of distributing the different instances of the experiments among the remaining 18 nodes. The total number of problem instances tested is 270 (all the combinations of the 10 functions, 9 levels of noise and 3 FLSs) and the total number of executions is 2700 (i.e., 10 repetitions for each instance). Thus, the work load has been distributed as uniformly as possible among all the nodes, following a round-robin scheme [Silberschatz et al., 2008].

Each case of the optimization problem was executed in a separate MATLAB instance in the assigned node. Additionally, MATLAB's Global Optimization Toolbox provides the possibility of using a pool of MATLAB workers to add

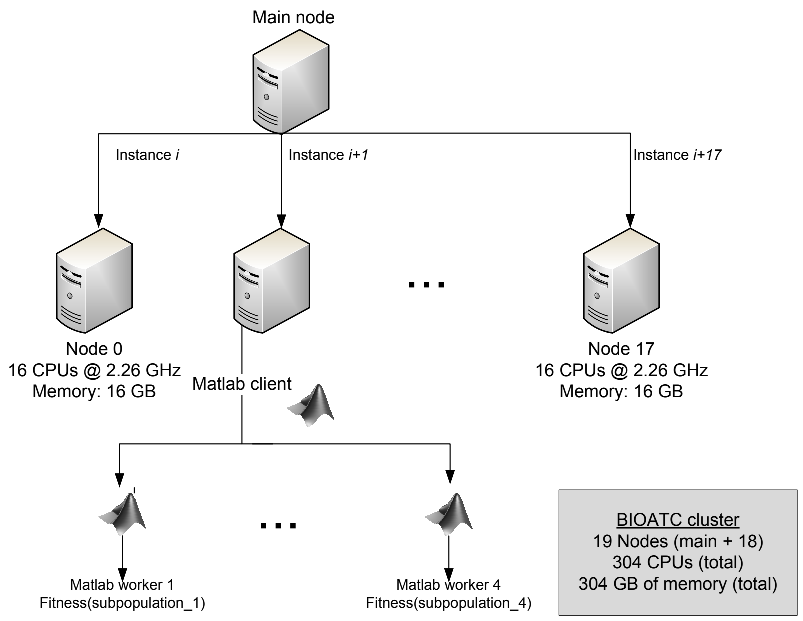


FIGURE C.1: Use of the cluster BIOATC for the execution of the MOEA

another degree of parallelism [MATLAB, 2010b]. More precisely, these workers can be used to evaluate the fitness of the population at each generation, by splitting the population into several subpopulations and evaluating them in parallel. Although in theory it is possible to have as many workers as CPUs are available in a node, the use of BIOATC is restricted to up to four workers per MATLAB instance. Fig. C.1 shows the detail of the workers in node 1, which is the same in all the other nodes.

Therefore, it can be considered that this procedure provides parallelism at two different levels, in the following sense: (i) *Task-parallelism* is provided through the distribution of the problem instances among the nodes in the system, and (ii) *data-parallelism* is obtained through the distribution of subpopulations among the workers for their evaluation. Nonetheless, it has to be kept in mind that, strictly speaking, data-parallelism refers to the distribution of data across different parallel computing resources [Rauber and Rünger, 2010] and, in this case, we are referring to a set of individuals as “data”.

Bibliography

Alcalá, R., Alcalá-Fdez, J., Casillas, J., Cordón, O., and Herrera, F. (2006). Hybrid learning models to get the interpretability-accuracy trade-off in fuzzy modeling. *Soft Computing*, 10(9):717–734.

Alcalá, R., Ducange, P., Herrera, F., Lazzerini, B., and Marcelloni, F. (2009). A multiobjective evolutionary approach to concurrently learn rule and data bases of linguistic Fuzzy-Rule-Based systems. *IEEE Trans. Fuzzy Syst.*, 17(5):1106–1122.

Alsina, C., Trillas, E., and Valverde, L. (1983). On some logical connectives for fuzzy sets theory. *Journal of mathematical analysis and applications*, 93:15–26.

Andersen, H., Lotfi, A., and Tsoi, A. (1997). A new approach to adaptive fuzzy control: the controller output error method. *IEEE Trans. Syst., Man, Cybern. B*, 27(4):686–691.

Angelov, P. (2004). A fuzzy controller with evolving structure. *Information Sciences*, 161(1-2):21–35.

Angelov, P. and Buswell, R. (2002). Identification of evolving fuzzy rule-based models. *IEEE Trans. Fuzzy Syst.*, 10(5):667–677.

Angelov, P., Filev, D. P., and Kasabov, N. K. (2010). *Evolving Intelligent Systems: Methodology and Applications*. Wiley-IEEE.

Angelov, P. and Kasabov, N. (2005). Evolving computational intelligence systems. *1st International Workshop on Genetic Fuzzy Systems*.

Antonelli, M., Ducange, P., Lazzerini, B., and Marcelloni, F. (2010). Learning concurrently data and rule bases of mamdani fuzzy rule-based systems by exploiting a novel interpretability index. *Soft Computing*, 15:1981–1998.

Antsaklis, P. J. (1994). Defining intelligent control – report of the task force on intelligent control. *IEEE Control Syst. Mag.*, pages 4–5, 56–58.

Antsaklis, P. J. and Passino, K. M., editors (1993). *An introduction to intelligent and autonomous control*. Kluwer Academic Publishers, Norwell, MA, USA.

Baeck, T., Fogel, D., and Michalewicz, Z. (1997). *Handbook of Evolutionary Computation*. Taylor & Francis, 1 edition.

Bellman, R. (1961). *Adaptive Control Processes: A Guided Tour*. Princeton University.

Bellomo, D., Naso, D., and Babuska, R. (2008). Adaptive fuzzy control of a non-linear servo-drive: Theory and experimental results. *Engineering Applications of Artificial Intelligence*, 21(6):846–857.

Biglarbegian, M., Melek, W., and Mendel, J. (2011). On the robustness of type-1 and interval type-2 fuzzy logic systems in modeling. *Information Sciences*, 181(7):1325–1347.

Bikdash, M. (1999). A highly interpretable form of sugeno inference systems. *IEEE Trans. Fuzzy Syst.*, 7(6):686–696.

Bleris, L. G., Vouzis, P. D., Garcia, J. G., Arnold, M. G., and Kothare, M. V. (2007). Pathways for optimization-based drug delivery. *Control Engineering Practice*, 15(10):1280–1291.

Bode, H. W. (1945). *Network analysis and feedback amplifier design*. Van Nostrand.

Bouchachia, A. (2010). Fuzzy classification in dynamic environments. *Soft Computing*, 15(5):1009–1022.

Bouchachia, A. and Mittermeir, R. (2006). Towards incremental fuzzy classifiers. *Soft Computing*, 11(2):193–207.

Box, G. E. P., Hunter, W. G., Hunter, J. S., and Hunter, W. G. (1978). *Statistics for Experimenters: An Introduction to Design, Data Analysis, and Model Building*. John Wiley & Sons.

Buckley, J. (1992). Universal fuzzy controllers. *Automatica*, 28(6):1245–1248.

Buckley, J. (1993). Sugeno type controllers are universal controllers. *Automatica*, 53:299–304.

Buckley, J. and Hayashi, Y. (1994). Can fuzzy neural networks approximate continuous fuzzy functions? *Fuzzy Sets and Systems*, 61:43–52.

Cao, Z. and Kandel, A. (1992). Investigations on the applicability of fuzzy inference. *Fuzzy Sets and Systems*, 49:151–169.

Cara, A. B., Herrera, L. J., Pomares, H., and Rojas, I. (2012). New online self-evolving neuro-fuzzy controller based on the TaSe-NF model. *Information Sciences*. Accepted with minor revision.

Cara, A. B., Lendek, Zs., Babuska, R., Pomares, H., and Rojas, I. (2010a). Online self-organizing adaptive fuzzy controller: application to a nonlinear servo system. In *Proc. 2010 IEEE World Congress on Computational Intelligence*, pages 2491–2498.

Cara, A. B., Pomares, H., and Rojas, I. (2010b). An algorithm for online self-organization of fuzzy controllers. In García-Pedrajas, N., Herrera, F., Fyfe, C., Benítez, J. M., and Ali, M., editors, *Trends in Applied Intelligent Systems*, volume 6097, pages 212–221. Springer Berlin Heidelberg, Berlin, Heidelberg.

Cara, A. B., Pomares, H., and Rojas, I. (2011a). A new methodology for the online adaptation of fuzzy self-structuring controllers. *IEEE Trans. Fuzzy Syst.*, 19(3):449–464.

Cara, A. B., Pomares, H., Rojas, I., Lendek, Zs., and Babuska, R. (2010c). Online self-evolving fuzzy controller with global learning capabilities. *Evolving systems*, 1(4):225–239.

Cara, A. B., Rojas, I., Pomares, H., Wagner, C., and Hagras, H. (2011b). On comparing non-singleton type-1 and singleton type-2 fuzzy controllers for a nonlinear servo system. In *2011 IEEE Symposium on Advances in Type-2 Fuzzy Logic Systems (T2FUZZ)*, pages 126–133.

Cárdenas, E., Castillo, J., Cordón, O., Herrera, F., and Peregrín, A. (1993). Influence of fuzzy implication functions and defuzzification methods in fuzzy control. *BUSEFAL*, 57:69–79.

Casillas, J. (2003). *Accuracy Improvements in Linguistic Fuzzy Modeling*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Casillas, J., Cordón, O., del Jesus, M. J., and Herrera, F. (2005). Tuning of fuzzy rule deep structures preserving interpretability and its interaction with fuzzy rule set reduction. *IEEE Trans. Fuzzy Syst.*, 13(1):12–29.

Casillas, J., Cordón, O., Triguero, F. H., and Magdalena, L. (2003). *Interpretability Issues in Fuzzy Modeling (Studies in Fuzziness and Soft Computing)*. Springer, 1 edition.

Castillo, E. (1991). *Expert systems: Uncertainty and learning*. Elsevier Applied Science.

Castillo, O., Melin, P., Alanis, A., Montiel, O., and Sepulveda, R. (2011). Optimization of interval type-2 fuzzy logic controllers using evolutionary algorithms. *Soft Computing*, 15(6):1145–1160.

Castro, J. (1995). Fuzzy logic controllers are universal approximators. *IEEE Trans. Syst., Man, Cybern.*, 25(4):629–635.

Castro, J. and Delgado, M. (1996). Fuzzy systems with defuzzification are universal approximators. *IEEE Trans. Syst. Man and Cyber.*, 26(1):149–152.

Celikyilmaz, A. and Turksen, I. B. (2008). Uncertainty modeling of improved fuzzy functions with evolutionary systems. *IEEE Trans. Syst., Man, Cybern. B*, 38(4):1098–1110.

Chang, M. (2010). An adaptive self-organizing fuzzy sliding mode controller for a 2-DOF rehabilitation robot actuated by pneumatic muscle actuators. *Control Engineering Practice*, 18(1):13–22.

Chen, C. H., Lin, C. J., and Lin, C. T. (2009). Nonlinear system control using adaptive neural fuzzy networks based on a modified differential evolution. *IEEE Trans. Syst., Man, Cybern. C*, 39(4):459–473.

Chen, F. and Khalil, H. (1995). Adaptive control of a class of nonlinear discrete-time systems using neural networks. *IEEE Trans. Autom. Control*, 40(5):791–801.

Chen, L. and Narendra, K. S. (2003). Intelligent control using multiple neural networks. *International Journal of Adaptive Control and Signal Processing*, 17(6):417–430.

Chen, P., Hsu, C., Lee, T., and Wang, C. (2008). Fuzzy-identification-based adaptive backstepping control using a self-organizing fuzzy system. *Soft Computing*, 13(7):635–647.

Cherkassky, V., Gehring, D., and Mulier, F. (1996). Comparison of adaptive methods for function estimation from samples. *IEEE Trans. Neural Netw.*, 7(4):969–984.

Coello, C. A. C., Lamont, G. B., and Veldhuizen, D. A. V. (2007). *Evolutionary algorithms for solving multi-objective problems*. Springer.

Cordón, O. (2001). *Genetic fuzzy systems: evolutionary tuning and learning of fuzzy knowledge bases*. World Scientific.

Cordón, O., Gomide, F., Herrera, F., Hoffmann, F., and Magdalena, L. (2004). Ten years of genetic fuzzy systems: current framework and new trends. *Fuzzy Sets and Systems*, 141(1):5–31.

Cordón, O. (2011). A historical review of evolutionary learning methods for mamdani-type fuzzy rule-based systems: Designing interpretable genetic fuzzy systems. *International Journal of Approximate Reasoning*, 52(6):894–913.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory*, 13(1):21–27.

da Fonseca, V., Fonseca, C., and Hall, A. (2001). Inferential performance assessment of stochastic optimisers and the attainment function. In Zitzler, E., Deb, K., Thiele, L., Coello, C., and Corne, D., editors, *Evolutionary Multi-Criterion Optimization, Proceedings*, volume 1993, pages 213–225. Springer-Verlag Berlin, Berlin.

Dagher, I., Georgiopoulos, M., Heileman, G., and Bebis, G. (1999). An ordering algorithm for pattern presentation in fuzzy ARTMAP that tends to improve generalization performance. *IEEE Trans. Neural Netw.*, 10(4):768–778.

de Barros, J. and Dexter, A. (2007). Evolving fuzzy model-based adaptive control. In *IEEE International Conference on Fuzzy Systems, 2007. FUZZ-IEEE 2007*, pages 1–5.

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. John Wiley and Sons.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.*, 6(2):182–197.

Derrac, J., García, S., Molina, D., and Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18.

Doctor, F., Hagras, H., and Callaghan, V. (2005a). A fuzzy embedded agent-based approach for realizing ambient intelligence in intelligent inhabited environments. *IEEE Trans. Syst., Man, Cybern. A*, 35(1):55– 65.

Doctor, F., Hagras, H., and Callaghan, V. (2005b). A type-2 fuzzy embedded agent to realise ambient intelligence in ubiquitous computing environments. *Information Sciences*, 171(4):309–334.

Driankov, D., Hellendoorn, H., and Reinfrank, M. (1993). *An introduction to fuzzy control*. Springer-Verlag.

Dubois, D. and Prade, H. M. (1980). *Fuzzy sets and systems: theory and applications*. Academic Press.

Echanobe, J., Delcampo, I., and Bosque, G. (2007). An adaptive neuro-fuzzy system for efficient implementations. *Information Sciences*, 178(9):2150–2162.

Eiben, A. E. and Smith, J. (2008). *Introduction to Evolutionary Computing*. Springer.

Eshelman, L. and Schaffer, J. (1993). *Real-coded genetic algorithms and interval-schemata*. Morgan Kaufmann Pub Inc, San Mateo.

Espejo, P. G., Ventura, S., and Herrera, F. (2010). A survey on the application of genetic programming to classification. *IEEE Trans. Syst., Man, Cybern. C*, 40(2):121–144.

Evans, W. R. (1950). Control system synthesis by root locus method. *IEEE Trans. American Institute of Electrical Engineers*, 69(1):66–69.

Figueroa, J., Posada, J., Soriano, J., Melgarejo, M., and Rojas, S. (2005). A type-2 fuzzy controller for tracking mobile objects in the context of robotic soccer games. In *IEEE International Conference on Fuzzy Systems, 2005. FUZZ-IEEE 2007*, pages 359–364.

Filev, D. and Yager, R. (1993). An adaptive approach to defuzzification based on level sets. *Fuzzy Sets and Systems*, 54:355–360.

Florido, J. P., Pomares, H., and Rojas, I. (2011). Generating balanced learning and test sets for function approximation problems. *International Journal of Neural Systems*, 21(03):247–263.

Fonseca, C. M. and Fleming, P. J. (1993). *Genetics algorithms for multiobjective optimization - Formulation, discussion and generalization*. Morgan Kaufmann Pub Inc, San Mateo.

French, R. (1999). Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701.

Gacto, M. J., Alcalá, R., and Herrera, F. (2009). Adaptation and application of multi-objective evolutionary algorithms for rule reduction and parameter tuning of fuzzy rule-based systems. *Soft Computing*, 13(5):419–436.

Galluzzo, M. and Cosenza, B. (2009). Control of the biodegradation of mixed wastes in a continuous bioreactor by a type-2 fuzzy logic controller. *Computers & Chemical Engineering*, 33(9):1475–1483.

Gao, Y. and Er, M. J. (2003). Online adaptive fuzzy neural identification and control of a class of MIMO nonlinear systems. *IEEE Trans. Fuzzy Syst.*, 11(4):462–477.

Gao, Y. and Er, M. J. (2005). An intelligent adaptive control scheme for postsurgical blood pressure regulation. *IEEE Trans. Neural Netw.*, 16(2):475–483.

García, S., Fernández, A., Luengo, J., and Herrera, F. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, 180(10):2044–2064.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition.

González, J., Rojas, I., Pomares, H., Herrera, L. J., Guillén, A., Palomares, J. M., and Rojas, F. (2007). Improving the accuracy while preserving the interpretability of fuzzy function approximators by means of multi-objective evolutionary algorithms. *International Journal of Approximate Reasoning*, 44(1):32–44.

González, J., Rojas, I., Pomares, H., Rojas, F., and Palomares, J. (2006). Multiobjective evolution of fuzzy systems. *Soft Computing*, 10(9):735–748.

Grossberg, S. (1988). Nonlinear neural networks - principles, mechanisms, and architectures. *Neural Networks*, 1(1):17–61.

Guenounou, O., Belmehdi, A., and Dahhou, B. (2009). Multi-objective optimization of TSK fuzzy models. *Expert Systems with Applications*, 36(4):7416–7423.

Guillaume, S. (2001). Designing fuzzy inference systems from data: An interpretability-oriented review. *IEEE Trans. Fuzzy Syst.*, 9(3):426–443.

Guillén, A., Pomares, H., González, J., Rojas, I., Valenzuela, O., and Prieto, B. (2009). Parallel multiobjective memetic RBFNNs design and feature selection for function approximation problems. *Neurocomputing*, 72(16-18):3541–3555.

Gupta, M. and Qi, J. (1991). Theory of t-norms and fuzzy inference methods. *Fuzzy Sets and Systems*, 40:431–450.

Gupta, M. M. (2000). *Soft computing and intelligent systems: theory and applications*. Elsevier.

Hagras, H. (2004). A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots. *IEEE Trans. Fuzzy Syst.*, 12(4):524–539.

Hagras, H. (2007). Type-2 FLCs: a new generation of fuzzy controllers. *IEEE Computational Intelligence Magazine*, 2(1):30–43.

Hagras, H., Callaghan, V., and Colley, M. (2004). Learning and adaptation of an intelligent mobile robot navigator operating in unstructured environment based on a novel online Fuzzy-Genetic system. *Fuzzy Sets and Systems*, 141(1):107–160.

Hagras, H., Doctor, F., Callaghan, V., and Lopez, A. (2007). An incremental adaptive life long learning approach for type-2 fuzzy embedded agents in ambient intelligent environments. *IEEE Trans. Fuzzy Syst.*, 15(1):41–55.

Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 2 edition.

Hazen, H. L. (1934). Theory of servomechanism. *Journal of the Franklin Institute*, 218:543–580.

Hellendoorn, H. and Thomas, C. (1993). The -quality defuzzification method. *Fifth IFSA World Congress*, pages 1159–1162.

Herrera, F. (2008). Genetic fuzzy systems: taxonomy, current research trends and prospects. *Evolutionary Intelligence*, 1:27–46.

Herrera, F., Lozano, M., Pérez, E., Sánchez, A. M., and Villar, P. (2002). Multiple crossover per couple with selection of the two best offspring: An experimental study with the blx-alpha crossover operator for real-coded genetic algorithms. In *Proceedings of the 8th Ibero-American Conference on AI: Advances in Artificial Intelligence*, IBERAMIA 2002, pages 392–401. Springer-Verlag.

Herrera, F., Lozano, M., and Sanchez, A. (2003). A taxonomy for the crossover operator for real-coded genetic algorithms: An experimental study. *International Journal of Intelligent Systems*, 18(3):309–338.

Herrera, L. J. (2007). *Sistemas inteligentes adaptativos para aproximación y predicción utilizando arquitecturas avanzadas (in Spanish)*. PhD thesis, University of Granada.

Herrera, L. J., Pomares, H., Rojas, I., Guillén, A., Gonzalez, J., Awad, M., and Herrera, A. (2007). Multigrid-based fuzzy systems for time series prediction: CATS competition. *Neurocomputing*, 70(13-15):2410–2425.

Herrera, L. J., Pomares, H., Rojas, I., Guillén, A., Rubio, G., and Urquiza, J. (2011a). Global and local modelling in RBF networks. *Neurocomputing*, 74(16):2594–2602.

Herrera, L. J., Pomares, H., Rojas, I., Guillén, A., and Valenzuela, O. (2011b). The TaSe-NF model for function approximation problems: Approaching local and global modelling. *Fuzzy Sets and Systems*, 171(1):1–21.

Herrera, L. J., Pomares, H., Rojas, I., Valenzuela, O., and Prieto, A. (2005). TaSe, a taylor series-based fuzzy system model that combines interpretability and accuracy. *Fuzzy Sets and Systems*, 153:403–427.

Holland, J. H. and Reitman, J. S. (1977). Cognitive systems based on adaptive algorithms. *SIGART Bull.*, (63):49–49.

Hornick, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

Hsu, C. (2007). Self-Organizing adaptive fuzzy neural control for a class of nonlinear systems. *IEEE Trans. Neural Netw.*, 18(4):1232–1241.

Hsu, C. and Lin, C. (2005). Fuzzy-identification-based adaptive controller design via backstepping approach. *Fuzzy Sets and Systems*, 151(1):43–57.

Ishibuchi, H. (2007). Multiobjective genetic fuzzy systems: Review and future research directions. In *2007 IEEE International Conference on Fuzzy Systems*, pages 911–916.

Ishibuchi, H., Murata, T., and Turksen, I. (1997). Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems. *Fuzzy Sets and Systems*, 89(2):135–150.

Ishibuchi, H., Nakashima, T., and Murata, T. (2001). Three-objective genetics-based machine learning for linguistic rule extraction. *Information Sciences*, 136(1-4):109–133.

Ishibuchi, H. and Nojima, Y. (2007). Analysis of interpretability-accuracy trade-off of fuzzy systems by multiobjective fuzzy genetics-based machine learning. *International Journal of Approximate Reasoning*, 44(1):4–31.

Ishibuchi, H., Nozaki, K., Yamamoto, N., and Tanaka, H. (1995). Selecting fuzzy if-then rules for classification problems using genetic algorithms. *IEEE Trans. Fuzzy Syst.*, 3(3):260–270.

Isidori, A. (1995). *Nonlinear Control Systems*. Springer, 3rd edition.

Jager, R. (1995). *Fuzzy Logic in Control*. Tesis Doctoral, Amsterdam.

Jammeh, E. A., Fleury, M., Wagner, C., Hagras, H., and Ghanbari, M. (2009). Interval type-2 fuzzy logic congestion control for video streaming across IP networks. *IEEE Trans. Fuzzy Syst.*, 17(5):1123–1142.

Jang, J. (1993). ANFIS - adaptive-network-based fuzzy inference system. *IEEE Trans. Syst., Man, Cybern.*, 23(3):665–685.

Jang, J. and Sun, C. (1993). Functional equivalence between radial basis function networks and fuzzy inference systems. *IEEE Trans. Neural Netw.*, 4(1):156–159.

Jeng, W. R., Yeh, C., and Lee, S. (2009). General type-2 fuzzy neural network with hybrid learning for function approximation. In *2009 IEEE International conference on fuzzy systems*, pages 1534–1539.

Jiang, X. and Han, Q. (2008). On designing fuzzy controllers for a class of nonlinear networked control systems. *IEEE Trans. Fuzzy Syst.*, 16(4):1050–1060.

Johansen, T. A. and Babuska, R. (2003). Multiobjective identification of takagi-sugeno fuzzy models. *IEEE Trans. Fuzzy Syst.*, 11(6):847–860.

Juang, C. (2008). A symbiotic genetic algorithm with local-and-global mapping search for reinforcement fuzzy control. *Journal of Intelligent and Fuzzy Systems*, 19(2):103–114.

Juang, C., Huang, R., and Cheng, W. (2010). An interval type-2 Fuzzy-Neural network with Support-Vector regression for noisy regression problems. *IEEE Trans. Fuzzy Syst.*, 18(4):686–699.

Juang, C. and Tsao, Y. (2008). A Self-Evolving interval type-2 fuzzy neural network with online structure and parameter learning. *IEEE Trans. Fuzzy Syst.*, 16(6):1411–1424.

Karnik, N. and Mendel, J. (2001a). Operations on type-2 fuzzy sets. *Fuzzy Sets and Systems*, 122(2):327–348.

Karnik, N., Mendel, J., and Liang, Q. (1999). Type-2 fuzzy logic systems. *IEEE Trans. Fuzzy Syst.*, 7(6):643–658.

Karnik, N. N. and Mendel, J. M. (2001b). Centroid of a type-2 fuzzy set. *Information Sciences*, 132(1-4):195–220.

Klein, M. (1998). *Calculus: An Intuitive and Physical Approach.* Dover.

Klement, E. P., Mesiar, R., and Pap, E. (2000). *Triangular norms.* Springer.

Knowles, J. D. and Corne, D. W. (2000). Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172.

Kosko, B. (1992). *Neural Networks and Fuzzy Systems*. Englewood Cliffs, NJ, Prentice Hall.

Kosko, B. (1994). Fuzzy systems as universal approximators. *IEEE Trans. Computers*, 43(1):1329–1333.

Kovalerchuk, B. and Taliansky, V. (1992). Comparison of empirical and computed values of fuzzy conjunction. *Fuzzy Sets and Systems*, 46:49–53.

Kruse, R., Gebhardt, J., and Klawonn, F. (1994). *Foundations of fuzzy systems*. John Wiley & Sons.

Lalouni, S., Rekioua, D., Rekioua, T., and Matagne, E. (2009). Fuzzy logic control of stand-alone photovoltaic system with battery storage. *Journal of Power Sources*, 193(2):899–907.

Lee, C. (1990). Fuzzy logic in control systems: fuzzy logic controller. II. *IEEE Trans. Syst., Man, Cybern.*, 20(2):419–435.

Leng, G., McGinnity, T. M., and Prasad, G. (2006). Design for self-organizing fuzzy neural networks based on genetic algorithms. *IEEE Trans. Fuzzy Syst.*, 14(6):755–766.

Li, C. and Lee, C. (2003). Self-organizing neuro-fuzzy system for control of unknown plants. *IEEE Trans. Fuzzy Syst.*, 11(1):135–150.

Li, J. D. (2008). A two-step rejection procedure for testing multiple hypotheses. *Journal of Statistical Planning and Inference*, 138(6):1521–1527.

Liang, Q. and Mendel, J. (2000). Interval type-2 fuzzy logic systems: Theory and design. *IEEE Trans. Fuzzy Syst.*, 8(5):535–550.

Lin, C. and Xu, Y. (2006). A novel genetic reinforcement learning for nonlinear fuzzy control problems. *Neurocomputing*, 69(16-18):2078–2089.

Lin, C. T., Lin, C. J., and Lee, C. S. G. (1995). Fuzzy adaptive learning control network with on-line neural learning. *Fuzzy Sets and Systems*, 71(1):25–45.

Lin, T., Liu, H., and Kuo, M. (2009). Direct adaptive interval type-2 fuzzy control of multivariable nonlinear systems. *Engineering Applications of Artificial Intelligence*, 22(3):420–430.

Linda, O. and Manic, M. (2011). Uncertainty-Robust design of interval type-2 fuzzy logic controller for delta parallel robot. *IEEE Trans. Ind. Informat.*, 7(4):661–670.

Linkens, D. A. and Nyongesa, H. O. (1996). Learning systems in intelligent control: an appraisal of fuzzy, neural and genetic algorithm control applications. *IEE Proceedings of Control Theory and Applications*, 143(4):367–386.

Liu, X., Mendel, J. M., and Wu, D. (2012). Study on enhanced Karnik–Mendel algorithms: Initialization explanations and computation improvements. *Information Sciences*, 184(1):75–91.

Liu, Y. and Zheng, Y. (2009). Adaptive robust fuzzy control for a class of uncertain chaotic systems. *Nonlinear Dynamics*, 57(3):431–439.

Luengo, J., García, S., and Herrera, F. (2009). A study on the use of statistical tests for experimentation with neural networks: Analysis of parametric test conditions and non-parametric tests. *Expert Systems with Applications*, 36(4):7798–7808.

Lughofer, E. (2011). *Evolving Fuzzy Systems: Methodologies, Advanced Concepts and Applications*. Springer.

Lughofer, E. and Angelov, P. (2011). Handling drifts and shifts in on-line data streams with evolving fuzzy systems. *Applied Soft Computing*, 11(2):2057–2068.

Luo, A. C. J. (2009). *Discontinuous Dynamical Systems on Time-varying Domains*. Springer, 1 edition.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press.

Maloof, M. and Michalski, R. (2004). Incremental learning with partial instance memory. *Artificial Intelligence*, 154(1-2):95–126.

Mamdani, E. (1974). Application on fuzzy algorithms for the control of a dynamic plant. *Proc. IEEE*, 121(12):1585–1588.

Martinez, R., Rodriguez, A., Castillo, O., Melin, P., and Aguilar, L. T. (2009). Optimization of type-2 fuzzy logic controllers for mobile robots using evolutionary methods. In *2009 IEEE International Conference on Systems, Man and Cybernetics (SMC 2009)*, pages 4764–4769.

MATLAB (2007a). *Optimization Toolbox User's Guide (version 3.1.2 R2007b).*

MATLAB (2007b). *Statistics Toolbox User's Guide (version 6.1 R2007b).*

MATLAB (2010a). *Global Optimization Toolbox User's Guide (version 3.1 R2010b).*

MATLAB (2010b). *Parallel Computing Toolbox User's Guide (version 5.0 R2010b).*

Maxwell, J. (1868). On governors. *Proceedings of the Royas Society*, 16:270–283.

Mendel, J. (1995). Fuzzy logic systems for engineering. *IEEE Proceeding*, 83(3):345–377.

Mendel, J. (2004). Computing derivatives in interval type-2 fuzzy logic systems. *IEEE Trans. Fuzzy Syst.*, 12(1):84–98.

Mendel, J. (2005). On a 50% savings in the computation of the centroid of a symmetrical interval type-2 fuzzy set. *Information Sciences*, 172(3-4):417–430.

Mendel, J. and John, R. (2002). Type-2 fuzzy sets made simple. *IEEE Trans. Fuzzy Syst.*, 10(2):117–127.

Mendel, J. and Wu, H. (2002). Uncertainty versus choice in rule-based fuzzy logic systems. In *2002 IEEE International conference on fuzzy systems*, pages 1336—1341.

Mendel, J., Zadeh, L., Trillas, E., Yager, R., Lawry, J., Hagras, H., and Guadarrama, S. (2010). What computing with words means to me [Discussion forum]. *IEEE Computational Intelligence Magazine*, 5(1):20–26.

Mendel, J. M. (2001). *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions*. Prentice Hall.

Mendel, J. M. (2007a). Advances in type-2 fuzzy sets and systems. *Information Sciences*, 177(1):84–110.

Mendel, J. M. (2007b). Type-2 fuzzy sets and systems: An overview [corrected reprint]. *IEEE Computational Intelligence Magazine*, 2(2):20–29.

Michels, K., Klawonn, F., Kruse, R., and Nürnberger, A. (2006). *Fuzzy Control: Fundamentals, Stability and Design of Fuzzy Controllers*. Springer, 1 edition.

Mingzhi, H., Jinquan, W., Yongwen, M., Yan, W., Weijiang, L., and Xiaofei, S. (2009). Control rules of aeration in a submerged biofilm wastewater treatment process using fuzzy neural networks. *Expert Systems with Applications*, 36(7):10428–10437.

Minorsky (1922). Directional stability of automatically steered bodies. *Journal of the American Society for Naval Engineers*, 34(2):280–309.

Mizumoto, M. and Tanaka, K. (1976). Some properties of fuzzy sets of type-2. *Information and Control*, 31(4):312–340.

Mizumuoto, M. and Tanaka, K. (1981). Fuzzy-sets of type-2 under algebraic product and algebraic sum. *Fuzzy Sets and Systems*, 5(3):277–290.

Mollov, S., Babuska, R., Abonyi, J., and Verbruggen, H. B. (2004). Effective optimization for fuzzy model predictive control. *IEEE Trans. Fuzzy Syst.*, 12(5):661–675.

More, J. (1978). The Levenberg-Marquardt algorithm: Implementation and theory. *Lecture Notes in Mathematics*, 630:105–116.

Mucientes, M. and Casillas, J. (2007). Quick design of fuzzy controllers with good interpretability in mobile robotics. *IEEE Trans. Fuzzy Syst.*, 15(4):636–651.

Navale, R. L. and Nelson, R. M. (2010). Use of evolutionary strategies to develop an adaptive fuzzy logic controller for a cooling coil. *Energy and Buildings*, 42(11):2213–2218.

Nieminen, J. (1997). On the algebraic structure of fuzzy sets of type 2. *Kybernetika*, 13(4):(261)–273.

Nounou, H. and Passino, K. (2004). Stable Auto-Tuning of adaptive Fuzzy/Neural controllers for nonlinear Discrete-Time systems. *IEEE Trans. Fuzzy Syst.*, 12(1):70–83.

Nyquist, H. (1932). Regeneration theory. *Bell System Technical Journal*, 11:126–147.

Ogata, K. (2001). *Modern Control Engineering*. Prentice Hall, 4 edition.

Ordonez, R., Zumberge, J., Spooner, J., and Passino, K. (1997). Adaptive fuzzy control: Experiments and comparative analyses. *IEEE Trans. Fuzzy Syst.*, 5(2):167–188.

Pachter, M. and Chandler, P. R. (1998). Challenges of autonomous control. *IEEE Control Systems*, 18(4):92–97.

Park, J., Park, G., Kim, S., and Moon, C. (2005). Direct adaptive self-structuring fuzzy controller for nonaffine nonlinear system. *Fuzzy Sets and Systems*, 153(3):429–445.

Park, K., Oh, S., and Pedrycz, W. (2009). Design of interval type-2 fuzzy neural networks and their optimization using real-coded genetic algorithms. In *2009 IEEE International conference on fuzzy systems*, pages 2013–2018.

Phan, P. A. and Gale, T. J. (2008). Direct adaptive fuzzy control with a self-structuring algorithm. *Fuzzy Sets and Systems*, 159(8):871–899.

Pomares, H. (2000). *Nueva metodología para el diseño automático de sistemas difusos (in Spanish)*. PhD thesis, University of Granada.

Pomares, H., Rojas, I., Gonzalez, J., Damas, M., Pino, B., and Prieto, A. (2004). Online global learning in direct fuzzy controllers. *IEEE Trans. Fuzzy Syst.*, 12(2):218–229.

Pomares, H., Rojas, I., Gonzalez, J., and Prieto, A. (2002a). Structure identification in complete rule-based fuzzy systems. *IEEE Trans. Fuzzy Syst.*, 10(3):349–359.

Pomares, H., Rojas, I., Gonzalez, J., Rojas, F., Damas, M., and Fernandez, F. J. (2002b). A two-stage approach to self-learning direct fuzzy controllers. *International Journal of Approximate Reasoning*, 29(3):267–289.

Pomares, H., Rojas, I., Ortega, J., Gonzalez, J., and Prieto, A. (2000). A systematic approach to a self-generating fuzzy rule-table for function approximation. *IEEE Trans. Syst., Man, Cybern. B*, 30(3):431–447.

Pulkkinen, P. and Koivisto, H. (2008). Fuzzy classifier identification using decision tree and multiobjective evolutionary algorithms. *International Journal of Approximate Reasoning*, 48(2):526–543.

Rajapakse, A., Furuta, K., and Kondo, S. (2002). Evolutionary learning of fuzzy logic controllers and their adaptation through perpetual evolution. *IEEE Trans. Fuzzy Syst.*, 10(3):309–321.

Rauber, T. and Rünger, G. (2010). *Parallel Programming: for Multicore and Cluster Systems*. Springer, 1st edition. edition.

Rojas, I., Pomares, H., Gonzalez, J., Herrera, L., Guillén, A., Rojas, F., and Valenzuela, O. (2006). Adaptive fuzzy controller: Application to the control of the temperature of a dynamic room in real time. *Fuzzy Sets and Systems*, 157(16):2241–2258.

Rojas, I., Pomares, H., Ortega, J., and Prieto, A. (2000). Self-organized fuzzy system generation from training examples. *IEEE Trans. Fuzzy Syst.*, 8(1):23–36.

Rovatti, R. and Guerrieri, R. (1996). Fuzzy sets of rules for system identification. *IEEE Trans. Fuzzy Syst.*, 4(2):89–102.

Sala, A. and Ario, C. (2009). Polynomial fuzzy models for nonlinear control: A taylor series approach. *IEEE Trans. Fuzzy Syst.*, 17(6):1284–1295.

Schaal, S. and Atkeson, C. (2010). Learning control in robotics. *IEEE Robotics & Automation Magazine*, 17(2):20–29.

Sepúlveda, R., Castillo, O., Melin, P., Rodríguez-Díaz, A., and Montiel, O. (2007). Experimental study of intelligent controllers under uncertainty using type-1 and type-2 fuzzy logic. *Information Sciences*, 177(10):2023–2048.

Sheskin, D. J. (2007). *Handbook of Parametric and Nonparametric Statistical Procedures: Fourth Edition*. Chapman and Hall/CRC, 4 edition.

Silberschatz, A., Galvin, P. B., and Gagne, G. (2008). *Operating System Concepts*. Wiley, 8 edition.

Smith, S. F. (1980). *A learning system based on genetic adaptive algorithms*. PhD thesis, University of Pittsburgh, Pittsburgh, PA, USA.

Spears, W. M., De Jong, K. A., Bäck, T., Ba, T., Fogel, D. B., and De Garis, H. (1993). An overview of evolutionary computation. *Lecture Notes in Computer Science*, 667(1):442–459.

Spooner, J., Ordonez, R., and Passino, K. (1997). Direct adaptive fuzzy control for a class of discrete-time systems. In *Proceedings of the 1997 American Control Conference*, volume 3, pages 1814–1818.

Srinivas, N. and Deb, K. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248.

Sugeno, M. (1985a). *Industrial applications of fuzzy control*. Amsterdam, North-Holland.

Sugeno, M. (1985b). An introductory survey of fuzzy control. *Information Sciences*, 36:59–83.

Sugeno, M. and Kang, G. (1988). Structure identification of fuzzy model. *Fuzzy Sets and Systems*, 28:15–33.

Takagi, T. and Sugeno, M. (1985). Fuzzy identification of systems and its applications to modelling and control. *IEEE Trans. Syst. Man and Cyber.*, 15:116–132.

Tewari, A. and Macdonald, M. (2010). Knowledge-based parameter identification of TSK fuzzy models. *Applied Soft Computing*, 10(2):481–489.

Uncu, O. and Turksen, I. B. (2007). Discrete interval type 2 fuzzy system models using uncertainty in learning parameters. *IEEE Trans. Fuzzy Syst.*, 15(1):90–106.

Wagner, C. and Hagras, H. (2007). Evolving type-2 fuzzy logic controllers for autonomous mobile robots. In Melin, P., Castillo, O., Ramírez, E., Kacprzyk, J., and Pedrycz, W., editors, *Analysis and Design of Intelligent Systems using Soft Computing Techniques*, volume 41 of *Advances in Soft Computing*, pages 16–25. Springer Berlin / Heidelberg.

Wang, C., Cheng, C., and Lee, T. (2004). Dynamical optimal training for interval type-2 fuzzy neural network (T2FNN). *IEEE Trans. Syst., Man, Cybern. B*, 34(3):1462–1477.

Wang, L. and Frayman, Y. (2002). A dynamically generated fuzzy neural network and its application to torsional vibration control of tandem cold rolling mill spindles. *Engineering Applications of Artificial Intelligence*, 15(6):541–550.

Wang, L. X. and Mendel, J. M. (1992). Fuzzy basis functions, universal approximation, and orthogonal least-squares learning. *IEEE Trans. Neural Netw.*, 3(5):807–814.

Wang, S., Huang, X., and Yam, Y. (2010). A neural network of smooth hinge functions. *IEEE Trans. Neural Netw.*, 21(9):1381–1395.

Wang, W., Chien, Y., and Li, I. (2008). An on-line robust and adaptive T-S fuzzy-neural controller for more general unknown systems. *International Journal of Fuzzy Systems*, 10(1):33–43.

Wu, D. and Mendel, J. M. (2009). Enhanced Karnik–Mendel algorithms. *IEEE Trans. Fuzzy Syst.*, 17(4):923–934.

Wu, D. and Nie, M. (2011). Comparison and practical implementation of Type-Reduction algorithms for type-2 fuzzy sets and systems. In *2011 IEEE International Conference on Fuzzy Systems*, pages 2131–2138.

Yen, J. and Wang, L. (1996). An SVD-based fuzzy model reduction strategy. In *1996 IEEE International Conference on Fuzzy Systems*, volume 2, pages 835–841.

Ying, H. (2000). *Fuzzy Control and Modeling: Analytical Foundations and Applications*. Wiley-IEEE Press.

Ying, H. (2008). General interval type-2 mamdani fuzzy systems are universal approximators. In *Fuzzy Information Processing Society, 2008. NAFIPS 2008. Annual Meeting of the North American*, pages 1–6. IEEE.

Ying, H. (2009). Interval type-2 Takagi-Sugeno fuzzy systems with linear rule consequent are universal approximators. In *Fuzzy Information Processing Society, 2009. NAFIPS 2009. Annual Meeting of the North American*, pages 1–5. IEEE.

You, F. and Ying, H. (2010). Interval type-2 boolean fuzzy systems are universal approximators. In *Fuzzy Information Processing Society (NAFIPS), 2010 Annual Meeting of the North American*, pages 1–4. IEEE.

Zadeh, L. (1965). Fuzzy sets. *Information and Control*, 8:338–353.

Zadeh, L. (1973). Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Trans. Syst., Man and Cyber.*, 3:28–44.

Zadeh, L. (1975). Concept of a linguistic variable and its application to approximate reasoning, 1. *Information Sciences*, 8(3):199–249.

Zadeh, L. A. (1996). Fuzzy logic = computing with words. *IEEE Trans. Fuzzy Syst.*, 4(2):103–111.

Zarandi, M. (2008). Reinforcement learning for fuzzy control with linguistic states. *Journal of Uncertain Systems*, 2(1):54–66.

Zhang, Q. and Li, H. (2007). MOEA/D: a multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput.*, 11(6):712–731.

Zhang, Y., Wu, X.-b., Xing, Z.-y., and Hu, W. (2011). On generating interpretable and precise fuzzy systems based on pareto multi-objective cooperative co-evolutionary algorithm. *Applied Soft Computing*, 11(1):1284–1294.

Zhao, R. and Govind, R. (1991). Defuzzification of fuzzy intervals. *Fuzzy Sets and Systems*, 43:45–55.

Zhou, S. and Gan, J. (2004). Improving the interpretability of Takagi-Sugeno fuzzy model by using linguistic modifiers and a multiple objective learning scheme. In *Proc. 2004 IEEE International Joint Conference on Neural Networks*, volume 3, pages 2385–2390.

Zilouchian, A. and Jamshidi, M., editors (2000). *Intelligent Control Systems Using Soft Computing Methodologies*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition.

Zitzler, E., Brockhoff, D., and Thiele, L. (2007). The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration. In *Evolutionary Multi-Criterion Optimization*, volume 4403, pages 862–876. Springer-Verlag Berlin, Berlin.

Zitzler, E., Knowles, J., and Thiele, L. (2008). Multiobjective optimization. page 373–404. Springer-Verlag, Berlin, Heidelberg.

Zitzler, E., Laumanns, M., and Thiele, L. (2002). SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems, EUROGEN 2001*, pages 95–100.

Zitzler, E. and Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms - a comparative case study. In *Parallel Problem Solving from Nature*, volume 1498, pages 292–301. Springer-Verlag Berlin.