SEP **TECNOLÓGICO NACIONAL DE MÉXICO**

INSTITUTO TECNOLÓGICO DE TIJUANA
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

GENERACIÓN AUTOMÁTICA DE CONTROLADORES DIFUSOS PARA UNA CLASE DE SISTEMAS

TRABAJO DE TESIS PRESENTADO POR
LIC. AMAURY SÁNCHEZ NEGRÍN

PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS DE LA INGENIERÍA

BAJO LA DIRECCIÓN DE
DR. NOHÉ RAMÓN CÁZAREZ CASTRO
Y CO-DIRECCION DE
M.C. ARMANDO MARTÍNEZ GRACILIANO

Abril 2021 **TIJUANA, B.C., MÉXICO**

"2020, Año de Leona Vicario, Benemérita Madre de la Patria"

Tijuana, Baja California, 18/diciembre/2020
ASUNTO: Autorización de impresión de Trabajo de Tesis

DRA. YAZMIN MALDONADO ROBLES
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
PRESENTE

En lo referente al trabajo de tesis, "**Generación Automática de Controladores Difusos para una Clase de Sistemas**", presentado por el **Lic. Amaury Sánchez Negrín**, alumno(a) del programa de Maestría en Ciencias de la Ingeniería, con número de control **G18211242**; informamos a usted que después de una minuciosa revisión e intercambio de opiniones, los miembros del comité manifiestan ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias, por lo que se autoriza al interesado para que proceda de inmediato a la impresión del mismo.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
Por una Juventud Integrada al Desarrollo de México ®

DR. NOHÉ RAMÓN CAZAREZ CASTRO
PRESIDENTE

M.C. ARMANDO MARTÍNEZ GRACILIANO
SECRETARIO

DR. SELENE LILETTE CÁRDENAS MACIEL
VOCAL

DR. JORGE ANTONIO LÓPEZ RENTERÍA
SUPLENTE

C.p. Alumno interesado.
Dr. José Ricardo Cárdenas Valdez – Coordinador Académico de la Maestría en Ciencias de la Ingeniería.

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLOGICO NACIONAL DE MEXICO

Instituto Tecnológico de Tijuana

Tijuana, Baja California, 12/enero/2021
OFICIO No. 001/DEPI/2021
Asunto: Autorización de Impresión de Tesis

MARIBEL GUERRERO LUIS
JEFA DEL DEPARTAMENTO DE SERVICIOS ESCOLARES
PRESENTE

En lo referente al trabajo de tesis, "Generación automática de controladores difusos para una clase de sistemas". Presentado por C. Amaury Sánchez Negrín, alumno de la Maestría en Ciencias de la Ingeniería con número de control G18211242; informo a usted que a solicitud del comité de tutorial, tengo a bien Autorizar la impresión de Tesis, atendiendo las disposiciones de los Lineamientos para la Operación de Estudios de Posgrado del Tecnológico Nacional de México.

Sin más por el momento le envío un cordial saludo.

ATENTAMENTE
Excelencia en Educación Tecnológica
Por una juventud integrada al desarrollo de México

EDUCACIÓN

INSTITUTO TECNOLÓGICO DE TIJUANA

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

YAZMIN MALDONADO ROBLES
JEFA DE DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

ccp. Archivo

TECNOLÓGICO
NACIONAL DE MÉXICO

Instituto Tecnológico de Tijuana
Posgrado en Ciencias de la Ingeniería

CARTA DE CESION DE DERECHOS

En la ciudad de Tijuana, Baja California, el día **18** del mes de **enero** del año **2021**, el que suscribe **Amaury Sánchez Negrín**, con número de control **G18211242**, alumno de **Maestría** del programa de Posgrado en Ciencias de la Ingeniería, manifiesta que es autor intelectual del presente trabajo de Tesis bajo la dirección del **Dr. Nohé Ramón Cázarez Castro**, cede los derechos del trabajo titulado '**Generación Automática de Controladores Difusos para una Clase de Sistemas**' al Tecnológico Nacional de México para su difusión, con fines académicos y de investigación en la comunidad estudiantil y científica del país.

Los usuarios de la información no deben reproducir el contenido textual, gráficas, código, formulas o datos del trabajo sin permiso expreso del autor o director del trabajo. Este debe ser obtenido escribiendo a cualquiera de las siguientes direcciones de correo electrónico amaury.sanchez18@tectijuana.edu.mx y nohe@tectijuana.edu.mx o bien, dirigirse a las instalaciones del Instituto Tecnológico de Tijuana en Calzada del Tecnológico S/N Esq. Av. Castillo de Chapultepec y calle Cuauhtemotzin, Fracc. Tomás Aquino C.P. 22414, Tijuana, Baja California, conmutador 664-6078400.

Si se otorga el permiso, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo como lo indique el autor intelectual o el director del trabajo de Tesis.

ATENTAMENTE

AMAURY SÁNCHEZ NEGRÍN
ALUMNO DEL POSGRADO EN CIENCIAS DE LA INGENIERÍA

GENERACIÓN AUTOMÁTICA DE CONTROLADORES DIFUSOS PARA UNA CLASE DE SISTEMAS.

Resumen

En la generación de sistemas de inferencia difusa, la tarea principal es la creación y ajuste de funciones de pertenencia y reglas difusas. Esta tesis aborda el problema del control a través de métodos basados en sistemas difusos que utilizan funciones triangulares. En esta tesis se describen aspectos básicos del control difuso así como el diseño de controladores donde se aplica un método de variación a los parámetros de las funciones de pertenencia de un sistema difuso, obteniendo como resultado una familia de sistemas difusos, mismos que fueron probados por simulación como controladores en lazo cerrado utilizando modelos representativos de una familia de sistemas, a pesar de la existencia de una dinámica uniforme. Además, el software desarrollado se presenta de manera que implementa el ajuste de los parámetros de las funciones de pertenencia para obtener un conjunto de sistemas difusos de forma automática.

Keywords: sistemas difusos sectoriales, sistemas adaptativos, control difuso.

AUTOMATIC GENERATION OF FUZZY CONTROLLERS FOR A CLASS OF SYSTEMS

Abstract

In the generation of fuzzy inference systems, the primary task is the creation and adjustment of membership functions and fuzzy rules. This thesis addresses the problem of regulatory control through methods based on fuzzy systems using triangular functions. This thesis describes basic aspects of fuzzy control as well as the design of controllers where a method of variation is applied to the parameters of membership functions of a fuzzy system, obtaining as a result a family of fuzzy systems, those who were tested as controllers by simulation of the closed loop using representative models of a family of systems, despite the existence of a uniform dynamics. Furthermore, the developed software is presented such that implements the adjustment of membership functions paremeters to obtain a set of fuzzy systems automatically.

Keywords: sectorial fuzzy systems, adaptive systems, fuzzy control.

Dedicatoria

A mi familia por su apoyo sincero.

A mis profesores y en especial a mi asesor por darme la oportunidad de realizar esta investigación.

Agradecimientos

Mi agradecimiento a:

Mis amigos, siempre conté con su apoyo y ayuda.

A mis profesores del posgrado y mi comité de tesis, especialmente a mi asesor.

Al Tecnológico Nacional de México.

Al financiamiento de CONACYT.

Al Instituto Tecnológico de Tijuana por permitirme ser parte de su posgrado.

Índice general

Lista de abreviaturas

PID	Proporcional Integral Derivativo
TS	Takagi-Sugeno
M	Mamdani
CLD	Control Lógico Difuso
LD	Lógica Difusa
SD	Sistema Difuso
CD	Conjunto Difuso
IoT	Internet of Things
IA	Inteligencia Artificial
MF	Función de pertenencia

Capítulo 1

INTRODUCCIÓN

En ingeniería de control se debe trabajar de forma permanente en la obtención de nuevos métodos de regulación que mejoren los que hasta el momento se han estado utilizando. Esta vertiginosa demanda de aplicaciones en control de procesos se debe en gran parte a la creciente variedad de dispositivos de instrumentación y control que se ponen a disposición en el mercado. Ante el aumento de posibilidades tecnológicas para la implementación de sistemas de regulación, no es posible la consideración de un mejor tipo de control, lo cual motiva a analizar las ventajas y los inconvenientes de cada uno de ellos, en función de las especificaciones y prestaciones que requiere la aplicación en el proceso.

El control basado en modelo se ha venido empleando tanto en la industria, como en la comunidad investigadora, esto se debe a que desarrolla la forma más general de formular el problema de control en el dominio del tiempo, e integra el control óptimo, control de procesos con tiempos muertos, procesos multivariable, etc. El modelado matemático [4, 18] de sistemas reales se ha ido desarrollando como una ciencia de gran importancia, ya que estos modelos se vienen utilizando para predicciones de comportamientos, simulaciones, análisis o diseño de controladores basados en modelos. Sin embargo, para sistemas no lineales se hace más difícil la construcción del modelo debido a que las leyes que rigen su comportamiento conllevan un profundo análisis y se dificulta concluir con la actividad si no se dispone del conocimiento previo que es útil para comprender los principios de la física y de las matemáticas que se deben aplicar, de ahí la lógica difusa (LD) [28, 42, 48] ha venido ganando terreno para modelación de procesos complejos. Las técnicas de control basado en modelos requieren del entendimiento de la dinámica del proceso y su ambiente operacional, lo cual dificulta la obtención de un modelo matemático preciso, debido a diversos factores que actúan como agentes que provocan perturbaciones u oscilaciones. En función de sobreponerse a esos problemas, las técnicas de control libre de modelo han sido aplicadas directamente a procesos complejos y en ambientes no bien definidos.

En la década de los sesenta fueron introduciéndose técnicas [19] de control lógico difuso a problemas de ingeniería, y ha tenido un crecimiento vertiginoso a través de los años [36]. Al mismo tiempo, esas técnicas permitieron introducir los conceptos necesarios para su aplicación en áreas industriales, debido a que se desempeñaban de forma eficiente en aplicaciones prácticas. Se desarrollaron nuevas teorías relacionadas a sistemas difusos que garantizaban los objetivos de control [14, 25], de este desarrollo se destaca el uso del sistema difuso del tipo Takagi-Sugeno (TS) [42] el cual consiste en reglas *IF-THEN* con antecedentes difusos y consecuentes aritméticos, estas reglas no son más que las acciones de control que realizaría un operador ante situaciones presentadas durante el proceso.

Cabe destacar que las técnicas de control difuso son una herramienta que permite diseñar controladores para sistemas complejos y con restricciones, ya que se logra obtener un mejor desempeño que con controladores tradicionales, pues están capacitados para tomar decisiones correctas con base a información lingüística imprecisa. Estos controladores reúnen los conocimientos de expertos en el área, los cuales son almacenados y representados como un conjunto de reglas formuladas con sentencias condicionales. Las ventajas mencionadas tienen un precio asociado, que es: el coste computacional, referido al tiempo de procesamiento y análisis de las reglas; y esfuerzos adicionales para hacer la sintonización del controlador.

Las distintas técnicas tradicionales de control para resolver problemas de regulación requieren conocimiento en técnicas y métodos matemáticos para la manipulación de ecuaciones diferenciales que corresponden a los modelos matemáticos de los sistemas dinámicos a controlar. Por otra parte las técnicas de inteligencia computacional también se ha utilizado como estrategias de solución para estos problemas de control, típicamente se han usado metodologías heurísticas para el diseño de las reglas de un sistema de inferencia difusa.

Incluso cuando se han presentado investigaciones y estudios con resultados concretos enfocados desde diferentes estrategias de control, se hace imperativo ampliar las investigaciones utilizando técnicas de control y de la inteligencia computacional tales como Redes Neuronales Artificiales (RNA), Control Lógico Difuso(CLD) o variantes que incluyen la unión de dos o más de ellos [27, 34, 44], debido a las características de aprendizaje y adaptación, así como robustez ante perturbaciones y frente a los efectos no lineales. Cuando no se dispone del modelo del sistema o su dinámica tiene grandes incertidumbres se hace complejo realizar el análisis y aplicar metodologías para el diseño de controladores basados en modelos. Construir los modelos en espacios de estados de una planta puede ser complicado y a veces se hace poco práctico, debido a las características del modelo físico de la planta, que se hace inevitable modelar toda su dinámica.

Se puede verificar, que a pesar de que los sistemas de control libres de modelos son una herramienta relativamente nueva en los tópicos de automatización, existe una amplia lista de aplicaciones exitosas en los diversos campos [2]. La literatura especializada presenta numerosos trabajos que hacen referencia al tema [7] y ha sido amplio el espectro o rango de técnicas utilizadas, que pueden ir desde redes neuronales hasta algoritmos genéticos. Ha sido significativo el desarrollo del control libre de modelos para sistemas no lineales, no solo teóricamente sino también en aplicaciones prácticas.

La metodología y teorías del control adaptativo asumen que ya se conoce la estructura de la planta a controlar, pero no los parámetros y que existe poca o ninguna variación en el tiempo. Para sistemas lineales se han desarrollado controles adaptativos y han sido numerosas las aportaciones acerca de este tema, no obstante, la no linealidad es una característica común de los sistemas reales, tales como sistemas de tráfico, procesos industriales, sistemas eléctricos [23, 45].

1.1. Estado del arte

La teoría de control convencional se basa en la utilización de modelos matemáticos (analíticos) explícitos del proceso que será controlado y las especificaciones del comportamiento deseado en lazo cerrado para diseñar el controlador. Este enfoque se puede dificultar si el modelo del proceso es:

- Difícil de obtener.
- Parcialmente desconocido.
- No lineal.

Para llevar a cabo el desarrollo de este proyecto se revisó la literatura relacionada con el tema para la recopilación de la información. De esta forma, se identificaron varias propuestas relacionadas con la generación de CLD. En [19], se propone una técnica, la cual se basa en un modelo no físico que utiliza un diferenciador numérico en línea para estimar las derivadas de la señal de salida, esta ley de control fue aplicada a un convertidor DC en tiempo real. En [14], es presentado un modelo de aprendizaje libre basado en el concepto de pseudo-gradiente y en el procedimiento de optimización de un sistema general discreto. Asimismo, se realiza la propuesta de un control adaptativo de aprendizaje libre de modelo basado en el concepto de pseudo gradiente con compensación. En [30, 31] se propone un método de diseño sistemático para CLD de tipo Proporcional Derivativo (PD) y Proporcional Integral (PI) [29]. En [40], es propuesto un esquema de control de rechazo

de perturbaciones usando algoritmos PID y retroalimentación no lineal. El enfoque consiste en una ecuación diferencial utilizada como generador de trayectoria transitoria, una tolerancia al ruido un diferenciador de seguimiento y un observador de estado extendido para obtener la estimación total de perturbación y rechazo.

De manera similar, en [22] se propone un esquema de control adaptativo libre de modelo basado en una linearización parcial, donde sus características son: proponer un esquema que utilice solo los datos de entrada/salida del sistema a controlar, de esta forma no es utilizado modelo matemático o información estructural de la planta, lo que implica que se puede diseñar un controlador independientemente y a la misma vez obtener un controlador genérico del proceso. El mecanismo de control no necesita proceso de entrenamiento ni señales de excitación externas, las cuales son requeridas en otros métodos de control adaptativo para sistemas no lineales.

Una de las aplicaciones de la lógica difusa son las relacionadas con controladores difusos directos, donde las acciones de control se calculan directamente por el controlador difuso, una clase de estos controladores difusos, llamados controladores difusos sectoriales, tienen propiedades sectoriales útiles de sus asignaciones de entra-salida, un ejemplo de estos puede verse en [8]. Similarmente, en [16] analizan el comportamiento de sistemas mecánicos. En particular, se trabaja con sistemas mecánicos subactuados, los cuales se caracterizan por contar en su estructura con más de 2 grados de libertad.

El problema de regulación para un péndulo invertido, problema clásico debido a sus aplicaciones en robótica, ha sido de gran interes, y para su solución se han venido utilizando varias técnicas de control, desde técnicas clásicas como las propuestas por [33] hasta llegar a la reportada en [15]. Además, han sido utilizadas también técnicas inteligentes como por ejemplo [11], donde se diseña controladores difusos Tipo-1 y Tipo-2 para sistemas mecánicos. De similar forma, en [10] se propone un algoritmo genético difuso tipo-2 para resolver el problema de regulación para un mecanismo con retardo (*backlash*).

Otras aplicaciones de inteligencia computacional en el control son presentadas en [39], donde es propuesta la utilización de Sistemas Difusos acompañados de otras técnicas inteligentes para dotar a robots de un control capaz de realizar trabajos con eficiencia y productividad.

Una propuesta metodológica para el diseño de controladores difusos a través de la síntesis difusa de Lyapunov puede verse en [12], está sustentada por la teoría de estabilidad de Lyapunov, de manera que es posible garantizar estabilidad de mecanismos, desde la etapa de diseño del controlador difuso. Por otra parte, en [46] se utiliza un controlador difuso sectorial, para el seguimiento de trayectoria en robots manipuladores, demostrando a través de la teoría de estabilidad de Lyapunov, que el sistema en lazo cerrado es

globalmente asintóticamente estable.

El creciente éxito de la lógica difusa ha motivado el desarrollo de numerosas herramientas dedicadas al diseño de este tipo de sistemas, su implementación varia en dirección al objetivo de la aplicación. Han sido propuestas varias técnicas para la síntesis automática de sistemas difusos mediante hardware específico basado en técnicas de diseño analógicas y digitales, así como, varias implementaciones de software utilizando lenguajes de alto nivel [20, 21, 26]. De igual forma, han surgido diferentes alternativas que aprovechan las herramientas incluidas en el entorno Matlab/Simulink para facilitar el diseño e implementación de controladores difusos [3]. En [43], se presenta el desarrollo de una herramienta que brinda la posibilidad de interactuar con CLD tipo Mamdani permitiendo su implementación como sistema embebido, cuyo objetivo principal es la generación automática del código en lenguaje C para el CLD deseado por el usuario.

1.2. Planteamiento del problema

El problema que se aborda en este trabajo es el de generar y ajustar controladores difusos por métodos computacionales. La problemática obedece a resolver problemas de control no lineal, lo cual obliga a acotar el problema como sigue:

Dada una estructura de sistema difuso definida en términos de variables de entrada–salida, funciones de membresía y reglas, obtener de forma automatizada una familia de CLD mediante la manipulación de los parámetros de las funciones de membresía.

1.3. Objetivos

Objetivo general:

Diseñar una estrategia de variación de los parámetros de las funciones de membresía e implementarla en un artefacto de software, para que a partir de un CLD dado, construir una familia de CLD para exhibir el desempeño en lazo cerrado con una clase de sistemas, preservando la estabilidad estructural.

Por tanto, los objetivos específicos del presente proyecto investigativo son:

Objetivos específicos:

- Estudiar y analizar las características de controladores difusos sectoriales y aplicarlo a sistemas dinámicos.
- Desarrollo de un artefacto de software, para la generación de sistemas difusos compatibles con Matlab.

- Verificar el desempeño de los controladores diseñados mediante la simulación de los sistemas.
- Validar de forma experimental los modelos y los controladores obtenidos.
- Analizar los resultados obtenidos.

1.4. Métodos y herramientas utilizadas

La métodos empleados se enumeran a continuación:

- El **método hipotético-deductivo**, al elaborar la hipótesis a partir del marco teórico con respecto al CLD y las técnicas basadas en la inteligencia computacional.
- El **método sistemático**, al relacionar varias problemáticas y proponer una solución basada en lógica difusa al problema de control. En este caso se propone la obtención de una familia de sistemas a través de un software.
- El **método de modelado**, para analizar complejidad visualizar alternativas de diseño y arquitecturas antes de empezar a desarrollar, representar las relaciones entre variables y dinámica del comportamiento del software.
- La **simulación** para comparar respuesta en lazo cerrado de la familia de CLD con cada uno de los sistemas caso de estudio, realizando de forma fácil las comparaciones de los parámetros que definen la respuesta de un controlador con respecto a los demás.

1.5. Contribuciones

Las contribuciones principales se enumeran a continuación:

- Propuesta de un método de generación automática de una familia de CLD mediante la variación de las funciones de membresía a partir de un controlador generado por un experto.
- Artefacto de software que provee una familia de controladores difusos que resuelven el problema de control, que sirve como herramienta de asistencia al usuario que le permite seleccionar el controlador según determinados criterios utilizados al observar la respuesta de control.

1.6. Estructura de la tesis

La tesis contiene esta introducción, tres capítulos, conclusiones, recomendaciones y referencias bibliográficas. A continuación, se resumen los aspectos tratados en los tres capítulos de contenido:

Capítulo 2: Se presenta un análisis del estado del arte relacionado con la temática del los sistemas difusos, lógica difusa y controladores difusos, se presentan las principales definiciones sobre el tema y se presenta una síntesis del estado del arte.

Capítulo 3 Se explica la alternativa propuesta, haciendo uso de la descripción del software desarrollado de forma modular y la obtención de la familia de CLDs mediante el algoritmo implementado.

Capítulo 4 Este capítulo hace refiere a experimentos realizados en el entorno Matlab/Simulink^{TM} a los sistemas que se identificaron previamente con la familia CLDs obtenidos, así como la gráfica de control que establecen estos a los sistemas.

Capítulo 2

SISTEMAS DIFUSOS

2.1. Introducción

Este capítulo aborda las principales definiciones y teoremas necesarios para el desarrollo de la metodología que se presenta en el capítulo 3. Se comienza planteando las definiciones de conjuntos difusos. Una breve descripción acerca de los sistemas difusos, específicamente el de tipo Mamdani que se utiliza en el capítulo 3. El capítulo termina con el epígrafe acerca de una selección de trabajos estudiados y referenciados donde se dan a conocer diferentes enfoques para solucionar problemas de control.

2.2. Lógica y razonamiento

La lógica es la base para el razonamiento, en ella se incluye el estudio de métodos y principios del razonamiento; el razonamiento no es más que la obtención de nuevas proposiciones a partir de las existentes. La lógica clásica maneja las proposiciones con valores estrictos de verdadero o falso, es decir es una lógica bivalente. La lógica difusa es una extensión al conjunto de teorías de la lógica matemática en la cual los valores de verdad están dados en términos de variables lingüísticas.

A partir de las primeras aplicaciones de la lógica difusa a problemas de ingeniería, se han definido varias vertientes de aplicacion tales como [35]:

1. Representación del conocimiento aproximado.

2. Aproximación universal de funciones.

3. Auto-organización.

Los sistemas basados en lógica difusa son buenos para la representación de conocimiento parcial e incompleto, imitando el razonamiento humano mediante reglas, con aplicaciones en control. En la segunda categoría se establece que estos sistemas combinados con otros algoritmos son capaces de estimar funciones, aplicando este conocimiento directamente al modelado de sistemas no lineales, utilizando técnicas de agrupamiento. En la tercera categoría nos encontramos métodos de aprendizaje para mejorar el rendimiento de estos sistemas, el aprendizaje no es más que un método de optimización con capacidad de memoria asociativa [35].

En la actualidad los sistemas lógico-difusos(también llamados sistemas difusos) por su versatilidad y considerando los avances que nos presentan son utilizados en combinación con técnicas avanzadas tales como control basado en modelos, control adaptativo, control por modos deslizantes, control predictivo, entre otras.

2.2.1. Lógica difusa

Los problemas de ingeniería de control han generado intereses hacia técnicas no convencionales que permitan resolver problemas que se presentan en el mundo real, es decir, no linealidad, varianza en el tiempo, etc. Los sistemas basados en inteligencia artificial se incluyeron en el desarrollo de controladores para dotarlos de un comportamiento como un operador humano. Los sistemas basados en lógica difusa considerados por varios investigadores como software del cerebro porque procesan la información como el cerebro humano han facilitado algorítmicamente la representación del conocimiento.

La Lógica Difusa nace cuando el Dr. Zadeh publica un artículo por el año 1965 titulado **Conjuntos Difusos** (CD) [48]. En este se describe como trabajar matemáticamente con expresiones imprecisas tal como lo hace el ser humano.

Según Zadeh, no debería considerarse la teoría de lógica difusa como una simple teoría, sino que se debería considerar el proceso de fuzzificación (en inglés fuzzification) como una metodología para generalizar cualquier teoría desde su versión ordinaria (discreta) a una nueva versión continua difusa. Así puede hablarse de *cálculo difuso, ecuaciones diferenciales difusas, autómatas difusos, sistemas dinámicos difusos*, etc.

Después de varios años de investigación, la lógica difusa ha demostrado su potencial aplicación en la ingeniería. Esta metodología es ideal para el modelado y el control de sistemas no lineales debido a sus características de ambigüedad, (en contraste con los sí/no o verdadero/falso de la lógica tradicional) que permiten considerar grados en las características consideradas en los problemas de ingeniería. Incluso, cuando no se disponga de modelos matemáticos rigurosos.

Los sistemas que utilizan lógica difusa han sido llamados sistemas basados en el conocimiento, pues estos se basan en conjuntos de reglas que utilizaría un operador que controla un proceso, con toda la incertidumbre que es generada por el lenguaje natural; por esta razón se les puede denominar controladores lingüísticos. Se ha constatado que su utilización puede llegar a ser ventajosa en sistemas de los cuales no se tenga dominio de conocimiento, cuando el modelo de la planta no sea fácil de obtener o cuando una o más variables a controlar son continuas.

A diferencia del álgebra de Boole clásica, en la cual la propiedad de un ente de pertenecer a un conjunto específico sólo puede tomar dos valores (falso, verdadero) a los que se les asigna por convenio los valores extremos 0 y 1, los conjuntos difusos son aquellos en los que se permite el grado de pertenencia parcial de los elementos que los forman, así como la descripción de conceptos en los cuales los límites entre poseer una propiedad y no poseerla no son claros.

La lógica difusa es considerada un sistema dirigido a proporcionar un modelo para los modos de razonamiento humano, los cuales son más aproximados que exactos (la mayoría del razonamiento humano, en particular el sentido común, es así.). En la teoría de los conjuntos difusos todo es una cuestión de grados. Está dirigida a tratar con fenómenos complejos que no pueden ser analizados mediante métodos clásicos basados en la lógica clásica.

Para un conjunto clásico, cualquier elemento del universo, o bien pertenece al conjunto o no pertenece. Para el caso de conjuntos difusos, un elemento del universo puede pertenecer a uno o más conjuntos con distintos grados de pertenencia.

Tomando como base la imprecisión del razonamiento humano, pero bajo un planteamiento matemático, la lógica difusa es capaz de generar la respuesta a una situación basándose en el conocimiento adquirido sobre ésta, que podrá ser inexacto e incompleto. Por lo tanto, se hace evidente que el razonamiento estricto de la lógica clásica contradice el razonamiento humano, mucho más vago e impreciso.

Las técnicas basadas en inferencia difusa han sido aplicadas al mundo industrializado (proceso y automatización) brindando un buen desempeño.

La teoría de los conjuntos difusos se puede enfocar en tratar fenómenos complejos que no pueden ser analizados mediante métodos clásicos basados en la lógica bivalente o la teoría de probabilidades. En la teoría clásica de los conjuntos un elemento del universo puede pertenecer o no a determinado conjunto. En el caso de conjuntos difusos, ese elemento del universo puede pertenecer a uno o más conjuntos con distintos grados de pertenencia.

A continuación se plantearán definiciones importantes que son de utilidad a lo largo

de la presente investigación:

Definición 2.1 (Universo de discurso [5, 32])**.** Determina el conjunto de valores que pueden tomar los elementos que poseen la propiedad definida por la variable lingüística.

Definición 2.2 (Etiquetas[5, 32])**.** Son las diferentes clasificaciones que se efectúan sobre la variable lingüística. Cada etiqueta tendrá un conjunto difuso asociado.

Definición 2.3 (Función de pertenencia o membresía $\mu(x)$ [5, 32])**.** Es una relación que asocia cada elemento en un conjunto con su grado de pertenencia al mismo (un número real en el intervalo $[0, 1]$). Puede expresarse como un grupo de valores discretos o como una función continua.

Definición 2.4 (Conjunto difuso [5, 32])**.** Es una relación que asocia cada elemento a un grado de pertenencia en un conjunto (un número real en el intervalo $[0, 1]$). Sea X_c un conjunto no vacío. El valor de x pertenece al conjunto X_c de la forma $\mu(x) \in [0, 1]$. Si $\mu(x) = 1$ indica que x está totalmente en X_c. Si se cumple que $\mu(x) = 0$ entonces $x \notin X_c$. El otro caso es $0 < \mu(x) < 1$ que indica que x está parcialmente en el conjunto X_c.

Definición 2.5 (Conjunto difuso nulo o vacío [5, 32])**.** Conjunto difuso nulo o vacío si no existe ningún elemento cuyo grado de pertenencia al conjunto sea distinto de cero.

Definición 2.6 (Núcleo de un conjunto difuso [5, 32])**.** Conjunto difuso a los valores que tengan grado de pertenencia igual a uno.

La operación $\alpha - corte$ de un conjunto difuso A denotada por $[A]^\alpha$ se define como:

Definición 2.7. Sea $\mu : \mathbb{R} \to [0, 1]$, la función de membresía de un conjunto difuso sobre $\mathbb{R}$, el $\alpha - corte$ o $\alpha - nivel$ es definido como el conjunto $[A]^\alpha = \{x \in \mathbb{R} : \mu_A(x) \geq \alpha\}$, para cada $0 < \alpha \leq 1$.

Otra de las propiedades importantes de los conjuntos difusos es el soporte y se define como:

Definición 2.8. Se define como soporte del conjunto difuso A y se denota como (A) al conjunto $(A) = \{x \in \mathbb{R} : \mu_A(x) > 0\}$ [48].

Una de las propiedades que se usará adelante es la altura de un conjunto difuso y se define como:

Definición 2.9. Sea $\mu_A : \mathbb{R} \to [0, 1]$, la función de membresía de un conjunto difuso A sobre $\mathbb{R}$, se define como altura y se denota por $h(A) = \sup_{x \in A} \mu_A(x)$. Donde sup es el supremo de un conjunto.

La operación defusificación sobre un conjunto difuso es quien permite convertir la información difusa en un valor preciso. Uno de los métodos de defusificación es la media de los máximos (MOM), la cual se define como:

Definición 2.10. [6] MOM se define como:

$$v = \frac{\sum_{x_i \in M}(x_i)}{|M|}, \tag{2.1}$$

donde: $M = h(A)$ y $|M|$ se refiere a la cardinalidad de M.

Funciones de pertenencia.

La esencia de los sistemas difusos usados para el control son las reglas lingüísticas apropiadas, basados en usar un cierto procedimiento de toma de decisión teniendo en cuenta la experiencia del usuario y las bases de datos del control humano. Las reglas difusas son establecidas basándose en la experiencia humana. Las funciones de pertenencias fundamentales son las siguientes:

- Funciones lineales (Triangular, trapezoidal y singleton)
- Funciones no lineales (Gaussiana, Sigmoidales.)

Definición 2.11 (Función de pertenencia triangular [5, 32])**.** Ver Figura 2.1. Una función de pertenencia triangular, se define según la expresión (2.2), siendo $\mathbf{m}$ su centro, $\mathbf{b} \geq 0$ su límite izquierdo, $\mathbf{a} \geq 0$ su límite derecho, con una relacion de orden ($\mathbf{a} < \mathbf{m} < \mathbf{b}$):

$$\mu(x) = \begin{cases} \dfrac{x - \mathbf{a}}{\mathbf{m} - \mathbf{a}} & \text{si} \quad \mathbf{a} < x \leq \mathbf{m} \\ \dfrac{\mathbf{b} - x}{\mathbf{b} - \mathbf{m}} & \text{si} \quad \mathbf{m} < y \leq \mathbf{b} \\ 0 & \text{en otro caso} \end{cases} \tag{2.2}$$

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

Definición 2.12 (Función de pertenencia singleton [5, 32])**.** Una función de pertenencia *singleton* es aquella función de pertenencia, soporte y núcleo que cumple $sup(x) = nuc(x) = x$. La función de pertenencia *singleton* es denominada también conjunto difuso escalar.

$$\mu(x) = 1. \tag{2.3}$$

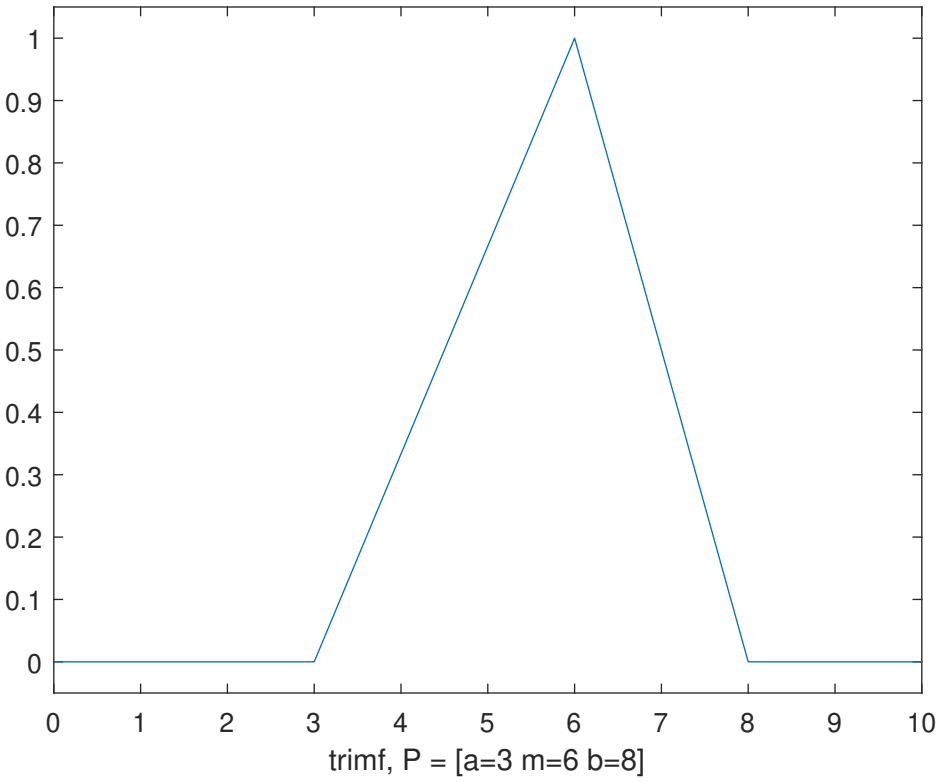

Figura 2.1: Función de pertenencia tipo triangular con parámetros a=3, m=6 y b=8.

Definición 2.13 (Función de pertenencia trapezoidal [5, 32])**.** Ver Figura. 2.2. Una función de pertenencia trapezoidal, se define según la expresión (2.4), siendo $[\mathbf{b}, \mathbf{c}]$, con $\mathbf{b} < \mathbf{c}$.

$$\mu(x) = \begin{cases} \dfrac{x - \mathbf{a}}{\mathbf{b} - \mathbf{a}} & \text{si} \quad \mathbf{a} < x \leq \mathbf{b} \\ 1 & \text{si} \quad \mathbf{b} \leq x \leq \mathbf{c} \\ \dfrac{\mathbf{d} - x}{\mathbf{d} - \mathbf{c}} & \text{si} \quad \mathbf{c} < x \leq \mathbf{d} \\ 0 & \text{en otro caso} \end{cases} \tag{2.4}$$

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

Definición 2.14 (Función de pertenencia gaussiana [5, 32])**.** Una función de pertenencia gaussiana, se define según la expresión (2.5), siendo m su centro:

$$\mu(x) = e^{-k(x-m)^2}. \tag{2.5}$$

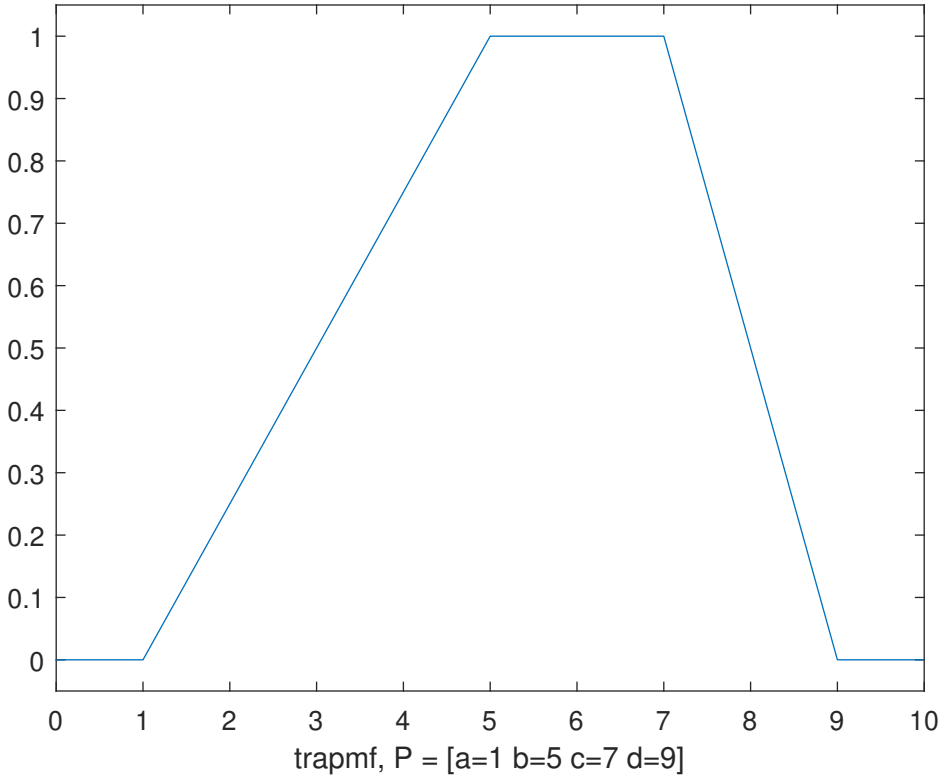

Figura 2.2: Función de pertenencia trapezoidal con parámetros a=1, b=5, c=7, d=9

La función de pertenencia gaussiana es continua, convexa y simétrica respecto al parámetro m, con un valor de $k > 0$ que define la amplitud de la campana de la función.

Es importante aclarar que las funciones lineales son las más usadas dentro de los algoritmos difusos con respecto a otras funciones más complejas. Esto es debido a la simplicidad de dichas funciones, además, las otras demandan un alto costo computacional.

2.2.2. Sistema difuso

Un Sistema Difuso (SD) de manera general Figura 2.3, es una función multivaluada donde a las entradas le corresponden salidas y la función de dicha relación es basada en reglas [13].

Específicamente los SD cuentan con una máquina de inferencia que es la encargada de realizar las operaciones que relacionan las entradas y salidas.

Existen varios modelos de SD, los más conocidos son: Takagi–Sugeno, Tsukamoto y Mamdani; este último se define a continuación y será el usado para este trabajo.

Un SD tipo Mamdani se define como:

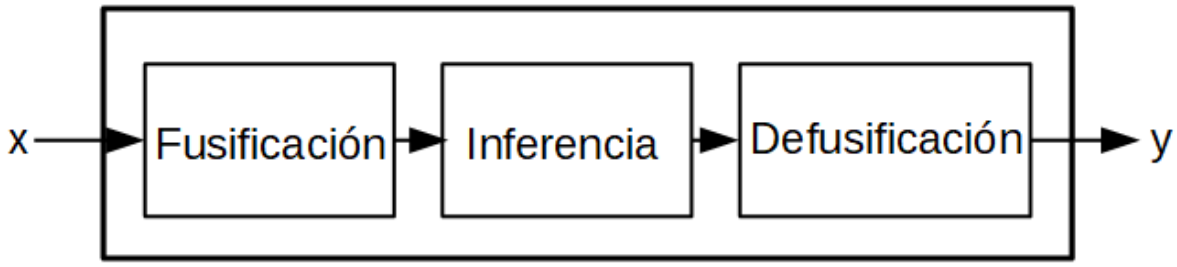

Figura 2.3: Estructura de un SD.

Definición 2.15. Un sistema de inferencia difuso tipo Mamdani [24], es una terna $I = (E, R, S)$ donde:

- E es el conjunto de variables lingüísticas de la entrada.
- R conjunto de reglas que relacionan la entrada con la salida.
- S es el conjunto de variables lingüísticas de la salida.

Las variables lingüísticas están constituidas por etiquetas lingüísticas que permiten asociar un significado lingüístico a intervalos en el dominio de cada variable. Una variable puede tener una o más etiquetas lingüísticas, matemáticamente estas se representan a través de funciones de pertenencia y en consecuencia por conjuntos difusos.

La relación de las variables lingüísticas de entrada y salida se hace a través de reglas **SI** − **ENTONCES**, en este trabajo se asume que el SD tiene múltiples entradas y una salida. Por ello, las reglas tienen la forma:

$$R_i : \textbf{SI}\ x_1 \text{ es } M_{p1}, x_2 \text{ es } M_{p2}, x_n \text{ es } M_{pk}, \textbf{ENTONCES}\ y_m \text{ es } N_{lj}, \tag{2.6}$$

donde $R_i \in R$ representa las reglas, las $x_n \in E$ son las variables lingüísticas de la entrada, M_{pk} son las etiquetas de las variables de entrada, $y_m \in S$ son las variables de salida y N_{lj} sus respectivas etiquetas lingüísticas.

El proceso de inferencia que realiza el SD consiste en tomar el mínimo de los conjuntos difusos formados por las activaciones de las reglas y el máximo al operar los conjuntos que están a la salida de las reglas formando un conjunto difuso que se somete a la etapa de defusificación la cual retorna un solo número real.

Existen varios métodos de defusificación, la mayoría se apoya en el área del conjunto difuso de salida , los más usados son: centro de área, bisectriz de área, máximo de los más grandes, media de los máximos y máximo más pequeño [47]. Para este trabajo se usa el

método MOM definido en (2.1), debido a que este método es de tendencia central y por tanto se evita coincidencias con los extremos siendo esto ventajoso para la aplicación.

2.2.3. Controladores lógico-difusos

La esencia de la lógica difusa se basa en implementar reglas lingüísticas apropiadas en un determinado procedimiento de toma de decisión, a partir de una tabla de la regla construida basada en la experiencia y recolección de datos del control humano. Los sistemas difusos están constituidos por variables de entrada y salida, fusificadores donde a partir de las leyes de inferencia son normalizados los valores de las variables en el rango [0 1], y desfusificadores. De acuerdo al valor difuso intervienen los desfusificadores que convierten ese valor en acción de mando sobre el actuador.

Tomando como base la imprecisión del razonamiento humano, pero bajo un planteamiento matemático, la lógica difusa es capaz de generar la respuesta a una situación basándose en el conocimiento adquirido sobre ésta, que podrá ser inexacto e incompleto. Por lo tanto, se hace evidente que el razonamiento estricto de la lógica clásica contradice el razonamiento humano, que es vago e impreciso.

Las técnicas basadas en la inferencia difusa han sido aplicadas al mundo industrializado (proceso y automatización) brindando un buen desempeño. Los resultados de este uso han demostrado que presentan ventajas en relación con algoritmos de control tradicional, tales como:

- No es necesario construir un modelo matemático detallado.
- Pueden funcionar con un gran número de entradas.
- Pueden ser adaptados fácilmente en sistemas no lineales.
- El conocimiento humano puede ser aplicado fácilmente.

El funcionamiento del Control Lógico Difuso es como un algoritmo que consta de tres pasos fundamentales:

- Fusificación de las variables de entrada: a cada variable de entrada se le asigna su valor de pertenencia correspondiente con cada conjunto difuso del universo de discurso.
- Inferencia basada en reglas: el controlador aplica el mecanismo de inferencia a la información de entrada y proporciona una conclusión difusa que determina los grados de pertenencia a los conjuntos difusos de salida.

- Desfusificación de la variable de salida: mediante métodos matemáticos se obtiene un valor concreto de la variable de salida o valor *crisp* a partir de los grados de pertenencias obtenidos en el proceso de inferencia.

La aplicación de técnicas difusas al control de sistemas dinámicos es ventajosa en términos de simplicidad del diseño y puesta en práctica. La experiencia demuestra que el éxito depende del nivel de conocimiento referente al comportamiento de la planta. Sin el criterio de experto, un ajuste puede comprometer el desempeño del sistema con el control difuso.

Las técnicas de control inteligente se han venido utilizando como estrategia de solución para problemas de control. Disímiles han sido los problemas de control de procesos solucionados mediante la utilización de la lógica difusa, la cual puede ser aplicada a procesos no lineales o variantes en el tiempo.

Existen herramientas de computo Algebraico como tales como el *Fuzzy Logic Toolbox* de Matlab^{TM}, que ofrece un entorno de desarrollo para aplicaciones de software y además tiene programas agrupados en toolboxes que permite hacer verificación numérica de los sistemas en lazo cerrado usando métodos numéricos implementados y permite realizar de forma manual ejecuciones de comandos de forma independiente.

Capítulo 3

SOFTWARE PARA LA GENERACIÓN AUTOMÁTICA DE CLDs

3.1. Introducción

En este capítulo se presenta el artefacto de software desarrollado para la generación de una familia de CLD. Se describe los requisitos del sistema, se define formalmente en la sección 3.4.1 el método para ajustar parámetros de funciones de pertenencia y la forma en que se generan cada CLD nuevo, los cuales representan el elemento primordial del proyecto posteriormente en 3.4.2 se presentan los elementos que conforman el artefacto de software y se describen las funciones de cada uno de ellos desde una perspectiva general de los procesos hasta mostrar el diseño de clases y la implementacion algoritmica del método definido en la sección 3.3.

3.2. Requerimientos de la Arquitectura de Software

Existen aspectos comunes entre las soluciones informáticas. La estandarización de todos estos patrones que se manifiestan de forma evidente, es lo que nos lleva a desarrollar una solución que carezca de compromisos visuales y que agrupe las funcionalidades comunes y que por otra parte tenga la escalabilidad como base en su desempeño. Una aplicación como esta podría extenderse y adaptarse a los requerimientos individuales de la solución que lo use.

3.2.1. Requerimientos Funcionales

Se pretende que la solución sea altamente integrable con otros sistemas que puedan usar la solución. En el caso de qué tecnología usar para el desarrollo, se necesita que el resultado sea estable y aceptado, para no caer en negativas por parte de los futuros usuarios.

Resumen Funcional.

1. Carencia de Compromisos Visuales: no exista dependencia entre la parte que maneja el negocio y la presentación del producto al cliente.

2. Altamente Integrable: exista la posibilidad de manejar librerías de código que faciliten la integración con código de otros lenguajes de programación.

3. Tecnología Estable y Aceptada: uso de una tecnología que esté asentada en el mercado del desarrollo de software y compita entre los más utilizados por los desarrolladores.

4. Gestión de Archivo: tener la posibilidad y estar abierto a la lectura, escritura y salvaguarda de la mayor variedad de extensiones de archivos y bases de datos que se manejan actualmente.

5. Variación de Plataforma: exista la posibilidad de manejar librerías de código que faciliten la integración con plataformas de hardware de distintos fabricantes.

6. Escalabilidad: permita programar por niveles estructurales que faciliten adicionar nuevas funcionalidades de negocio.

3.2.2. Arquitectura Propuesta.

Arquitectura Multicapas

Muchas aplicaciones están compuestas por múltiples componentes, donde cada uno realiza una tarea distinta. Todas las soluciones de software contienen similares componentes considerando el tipo específico de negocio al que este dirigido [38]. Por ejemplo muchos tipos de negocio contienen componentes de acceso a datos, encapsulan reglas de negocio, manipulan interacciones con los usuarios. Identificando los tipos de componentes comunes encontrados en las soluciones nos ayuda a construir un buen diseño para aplicaciones [38]. Una arquitectura multicapa particiona todo el sistema en distintas unidades funcionales cliente, presentación, lógica–de –negocio e integración. Esto asegura una división clara de

responsabilidades y hace que el sistema sea más extensible. Los sistemas con tres o más capas se han probado como más escalables y flexibles que un sistema cliente-servidor, en el que no existe la capa central de lógica de negocios [38]. Escalabilidad y flexibilidad. Se necesita una arquitectura escalable que permita introducir nuevos módulos.

Programación orientada a objetos

Como lenguaje de programación orientado a objetos, las aplicaciones Java™ se basan en el concepto de objetos. Uno de los beneficios de los lenguajes de programación orientados a objetos es que el código es modular y se puede reutilizar. Además, la programación orientada a objetos proporciona a los desarrolladores una guía clara sobre cómo los objetos pueden interactuar entre sí, mejorando así la productividad de los desarrolladores[17, 37, 41].

Los desarrolladores han podido separar de forma óptima las tareas que no debían estar mezcladas internamente en la lógica específica de cada aplicación, dentro de la implementación de código orientado a objetos. Se trata de tareas que normalmente se aplican a gran cantidad de componentes de una forma horizontal, como conexiones a bases de datos, utilización transparente de transacciones o comprobaciones de seguridad [1, 38].

Componentes de interfaz de usuario: Muchas soluciones necesitan proporcionar la vía al usuario de interactuar con las aplicaciones. Las interfaces de usuario están implementadas usando formas de Windows o páginas Web, controles y otras muchas tecnologías para brindar y mostrar datos al usuario y adquirir, validar datos provenientes de los mismos [38].

Componentes de Negocio

Considerando que el proceso de negocio puede consistir en un simple paso o en un total engranado flujo de trabajo, las aplicaciones probablemente requerirán de componentes que sean capaces de implementar reglas de negocio y realizar tareas del negocio [38].

Componente lógico de acceso a datos

Muchas aplicaciones necesitan acceder a los datos durante el proceso de negocio, se hace sensible abstraer el acceso a datos de los componentes lógicos de acceso a datos. Esto logra centralizar las funcionalidades de acceso y logra facilidad de configuración y mantenimiento [38].

3.2.3. Construcción del Software.

El proceso de implantación del sistema comienza con la definición de la metodología que podría involucrar la solución a lo largo del proyecto. Se trata de una implantación progresiva y por fases, que abarca los pasos siguientes:

- Análisis
- Diseño
- Construcción
- Implementación
- Objetivo

La finalidad es maximizar la productividad y eficiencia de la implementación y adecuar la solución a las exigencias del cliente.

3.3. JAVA™ como variante de desarrollo.

Una respuesta rápida sería decir que todos los lenguajes indicados tienen ventajas y defectos, y en definitiva la elección será siempre subjetiva. El mejor lenguaje será aquel que mejor encaje en las preferencias de cada usuario, y sirva mejor al fin que persigue. No obstante hay algunas cuestiones que pueden servir para decidirse por Java™. Seguro que hay muchas más, pero te vamos a describir un conjunto que destaca, y no se trata de características que no se encuentren en otros lenguajes de programación u otras plataformas de desarrollo, pero la suma de todas es una de las cosas que más se valoran para desarrollar aplicaciones Java™

Se trata de un lenguaje de programación y una plataforma informática de desarrollo que permite el diseño y desarrollo de diversas aplicaciones y que, además, hace que funcionen correctamente en todos los dispositivos desde ordenadores hasta smartphones.

Java™ es una plataforma de desarrollo de propósito general. Existen muchos motivos, técnicos y no técnicos, para que sea una de las más populares entre los programadores de todo el mundo. Pero existen razones para desarrollar un programa con Java™.

Seguro que hay muchas más, pero te vamos a describir un conjunto que destaca, y no se trata de características que no se encuentren en otros lenguajes de programación u otras plataformas de desarrollo, pero la suma de todas es una de las cosas que más se valoran para desarrollar aplicaciones Java™. [17, 41].

Es multiplataforma

El atributo escribir una vez, ejecutar en cualquier lugar de Java™ apoyó la adopción meteórica de esta plataforma en las primeras dos décadas del siglo XXI. Las empresas descubrieron que no necesitan invertir en hardware específico para implementar aplicaciones Java™ y, por lo tanto, trasladan el banco de tecnología de software a Java™ para adaptarse a Infraestructuras de TI heterogéneas. Mientras tanto, los desarrolladores apreciaron Java™ porque podían implementar Aplicaciones Java™ en cualquier dispositivo, siempre que tenga instalado Java Runtime Environment (JRE) [17].

Actualmente existen muchas otras plataformas que ofrecen esta posibilidad, pero el hecho de que Java siga funcionando en cualquier servidor y sistema operativo sigue siendo uno de sus mayores atractivos para cualquier programador. Existe una implementación de la máquina virtual de Java (JVM) para casi cualquier sistema.

Madurez

Java™ es conocido por su confiabilidad con respecto a la calidad de las actualizaciones del JDK y el rigor de la revisión y pruebas que ha recibido a lo largo de los años. Ha sido refinado y mejorado por los desarrolladores, arquitectos e ingenieros durante un período de 25 años. La vida útil de Java™ de 25 años marcaron décadas de intensa y rápida digitalización que incluyó la adopción meteórica de Internet, nuevas tecnologías, el nacimiento de la computación en la nube y las iniciativas de transformación digital contemporáneas. Java está asociado con la preparación de nivel empresarial que se deriva de ser probado en batalla por millones de desarrolladores a lo largo de su historia [17, 37].

Desempeño

Los desarrolladores asocian Java™ con su rendimiento y capacidad de escalar. El compilador Just in Time (JIT) que forma parte de la JVM aprovecha una multitud de optimizaciones para mejorar el rendimiento de Java aplicaciones. Como el compilador Just in Time compila una aplicación en tiempo de ejecución, tiene la capacidad de optimizar dinámicamente tanto el código en sí como la intersección del código con el código específico hardware en el que se ejecuta la aplicación [17, 37].

Comunidad

Java™ cuenta con una de las comunidades de desarrolladores más vibrantes del mundo. Son técnicos con opiniones sólidas sobre el futuro de Java™. Mientras tanto, los desarrolladores pueden asistir a grupos de usuarios de Java™ virtuales o en persona para

obtener más información sobre cómo sus pares utilizan Java™. Uno de los atributos que hacen la diferencia de la comunidad Java™ consiste en su atención a cuestiones relacionadas con el progreso de Java™ y la transparencia de las discusiones asociadas sobre su futuro [37].

Biblioteca de clases

La API de Java™ es una lista de todas las clases que forman parte del JDK. Estas clases proporcionan a los desarrolladores funcionalidades preescritas listas para su reutilización en el proceso de desarrollo. Los desarrolladores pueden aprovechar el poder de la biblioteca de clases masiva de Java para acelerar el desarrollo.

Java™ ofrece un manejo automático de la memoria, no compromete los datos de otras aplicaciones o del sistema operativo. Básicamente, trabaja con objetos que hacen referencia a datos dentro de su máquina virtual, nunca a datos que estén fuera, por lo que su uso se hace eficientemente sin que el programador tenga que preocuparse del manejo de objetos que no se utilizan o la memoria [37].

Fácil de aprender

Java™ incluye una enorme cantidad de funcionalidades de base, listas para ser utilizadas. Pero, se encuentran implementados y a tu disposición multitud de código listo para ser usado. Esta es una de las grandes ventajas que tiene el que se trate de una plataforma Open Source, es decir, de código abierto. La comunidad te va a facilitar lo que necesites. Una comunidad que lleva más de veinte años trabajado en la creación de aplicaciones Java™. Además, Java tiene garantía de seguridad, ya que muchas de las bibliotecas creadas las mantienen grandes compañías como Google, Facebook o la fundación Apache. No existen muchas otras plataformas que puedan aportar la misma exuberante variedad de código utilizable [37].

Versatilidad

Mientras que el atributo escribir una vez, ejecutar en cualquier lugar de la JVM fue uno de los principales impulsores de Java™ adopción generalizada, Java™ enfrenta el desafío de seguir siendo relevante dadas las innovaciones recientes en desarrollo de aplicaciones modernas, como un mayor uso de contenedores, microservicios, pruebas automatizadas y metodologías de desarrollo rápido de aplicaciones [37]. Dentro de estas aplicaciones y por su portabilidad tecnológica y seguridad, hace que sea ideal para implementaciónes de software que van desde portátiles hasta centros de datos, consolas para juegos, súper

computadoras, para telefonía móviles, Internet, equipamiento para Internet de las cosas de sus siglas en ingles (IoT), Java^{TM} está en todas partes.

.

3.4. Generación de CLDs

3.4.1. Modificación de funciones de membresía

Se propone una metodología basada en las ideas de [9], que es implementada en software para producir una familia de CLD a través de modificaciones a las funciones de membresía.

Es conocido que las característica de desempeño de controladores difusos tienen que ver con la forma y disposición de las funciones de pertenencia dentro del universo de discurso de la variable lingüíistica, con ello el enfoque de solución al problema de esta tesis se basa en realizar modificaciones en los parámetros de las funciones de pertenencia. Con base en la idea de modificación uniforme de funciones de pertenencia realizada en [9] se describe la manera de hacer modificación de parámetros de funciones de pertenencia tipo triangular.

Sea una variable lingüística con su conjunto difuso A_i sobre $X = [D_{min}, D_{max}]$ y cada una de sus funciones de pertenencia $\mu_{A_{i,j}}(x; a_{i,j}, b_{i,j}, c_{i,j})$ que satisfacen la Definición 2.11. Considérese una cantidad $\epsilon <= min\{c_{i,j} - b_{i,j}, b_{i,j} - a_{i,j}\} > 0$ y fija. $\forall \mu_{A_{i,j}}$ se aplica la modificación de los parámetros $a_{i,j}$ y $c_{i,j}$ usando ϵ para obtener de cada $\mu_{A_{i,j}}$ dos nuevas funciones de membresía tal que $\mu_{A_{i,j}}(x; a_{i,j} - \epsilon, b_{i,j}, c_{i,j} + \epsilon)$ y $\mu_{A_{i,j}}(x; a_{i,j} + \epsilon, b_{i,j}, c_{i,j} - \epsilon)$.

El efecto que geométrico que implica este suma y diferencia en una función de pertenencia triangular es la de ensanchar y estrechar la base del triangulo respectivamente manteniendo fijo su tercer vértice que sería el relativo a la altura respecto a la base; ver Figura 3.1.

Cada nuevo CLD se construye haciendo uso del algoritmo definido en Algoritmo 1. Este proceso de creación de archivos de texto se ejecuta tantas veces como modificaciones a funciones de membresía se lleven a cabo y daría como resultado una cantidad de:

$$\binom{n}{2} = \frac{(n)!}{2!(n-2)!}, \tag{3.1}$$

controladores difusos. Siendo $n = 2 * cantidad$ de funciones de pertenencia (MF).

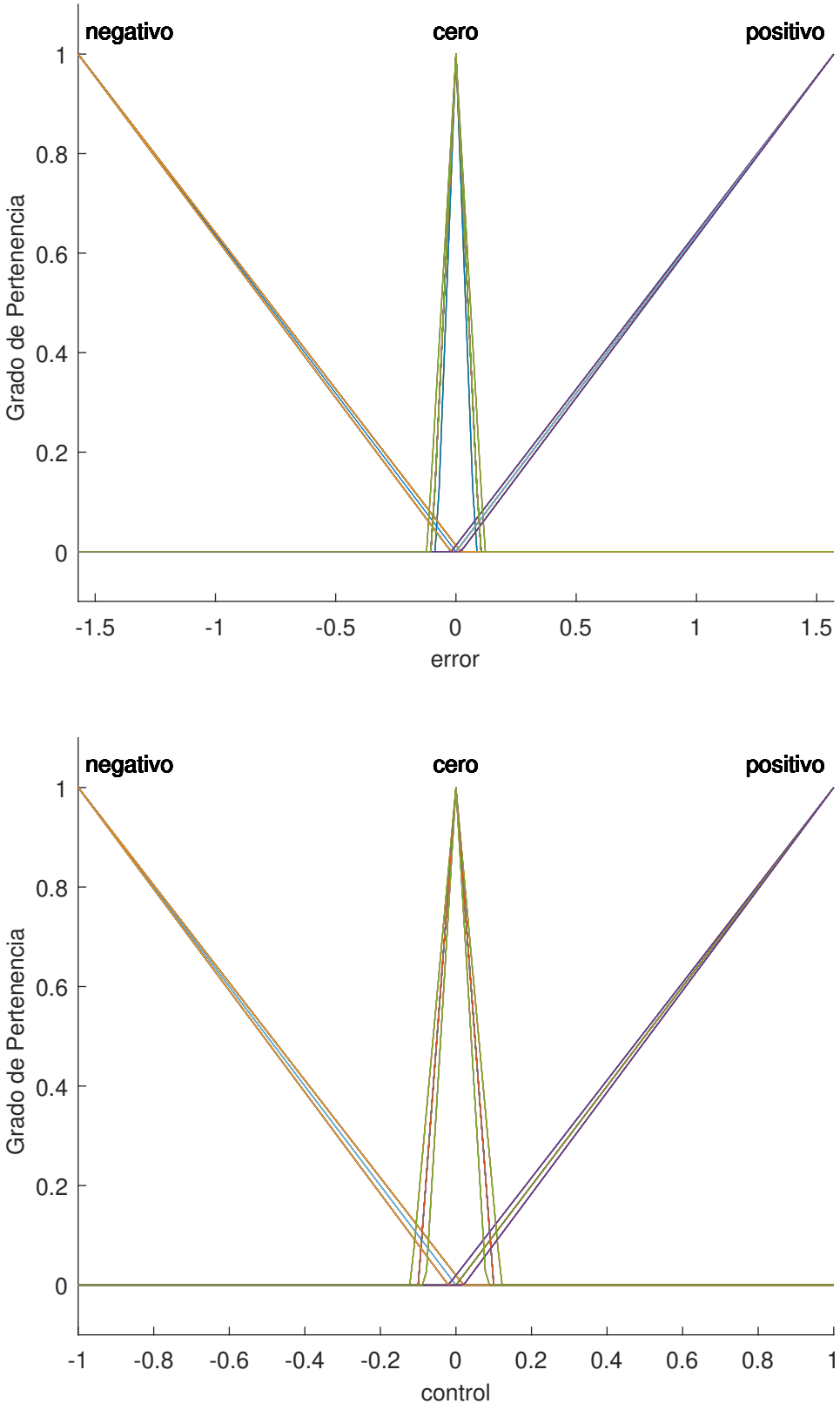

Figura 3.1: Funciones de Membresía obtenidas por el software para la variables linguisticas *error* (arriba) y *control* (abajo).

Algorithm 1: Algoritmo para la Generación de CLD

```
initialization;
for (i : n − 1) do
    for MF_i do
        varParams(double epsilonFis);
        for (j = i + 1 : n − 1) do
            for MFj do
                varParams(double epsilonFis);
            end
        end
    end
end
```

3.4.2. Descripción del Artefacto de Software

La generación automática de CLDs se realiza a través de una herramienta de software que se compone de dos módulos o subsistemas: (i) el primer módulo, el cual está implementado en lenguaje de programación Java™, recibe el CLD diseñado por el usuario y aplica la modificación de parámetros de funciones de membresía, cuyo resultado lo utiliza para crear nuevos CLDs que son representados con una estructura apropiada y posteriormente almacenados en archivos de texto; (ii) El segundo módulo esta implementado en Matlab™, se encarga de ejecutar simulación del sistema en lazo cerrado que usa cada nuevo CLD, cada respuesta en lazo cerrado del sistema controlado se almacena para futuros procesamientos y se utiliza para generar reportes gráficos.

- (Módulo I) Implementado en Java™. El recibe los datos del controlador difuso que genera el experto para la generación del controlador difuso, luego y dada las características descriptivas de dicho controlador, pues se establecen las reglas y se genera un valor *epsilon* (ϵ) el cual tiene que cumplir con los requisitos establecidos en 4.0.1, de esta forma se completa el ciclo de creación del conjunto de controladores difusos(CLDs). El código fuente de las clases que interactuan en el proceso se encuentran en el anexo A.

- (Módulo II) Implementado en Matlab™. Este script mostrado en el anexo B, carga cada CLDs salvaguardado por el Módulo I e implementa la modelación matemática

de las ecuaciones diferenciales del sistema a controlar, funciones que evaluan numéricamente el sistema en lazo cerrado con cada controlador; y funciones que generan los resultados numéricos así como el comportamiento del error, también se obtienen gráficas del comportamiento de la familia de CLDs estudiados para cada uno de los sistemas escogidos.

En la Figura 3.2 se representa de forma general como es el proceso del sistema que se propone para la generación automática de CLD's.

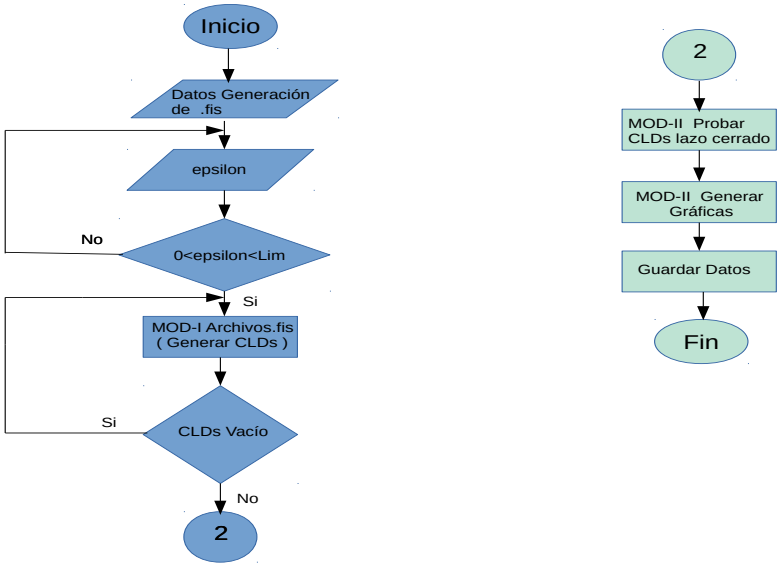

Figura 3.2: Diagrama de Flujo del Proceso de Generación de CLDs.

En la Figura 3.3 se muestra el diagrama de clases con sus atributos y sus métodos, que conforman al primer módulo. La clase *fisCreator* es la encargada de llevar el objeto instanciado de la clase *FISFILE* (que representa cada CLD) a objeto archivo con extensión .fis; en la clase *Controller* se genera la familia de archivos (nuevos CLDs) mediante la función *generate* el cual es el método donde se realiza la variación de parámetros de funciones de membresía que se describe en la subsección 4.0.1, también se implementa el método para la construcción de la base de reglas. La clase *Var* contiene los métodos y atributos que representan a una variable lingüística, a ella de encuentra agregada la clase *Range* y la superclase *MF* contiene métodos y atributos generales, de esta última clase heredan las clases que representan a funciones de pertenencia como es el caso de *TRIMF* que contiene métodos y atributos característicos de una función de pertenencia tipo triangular.

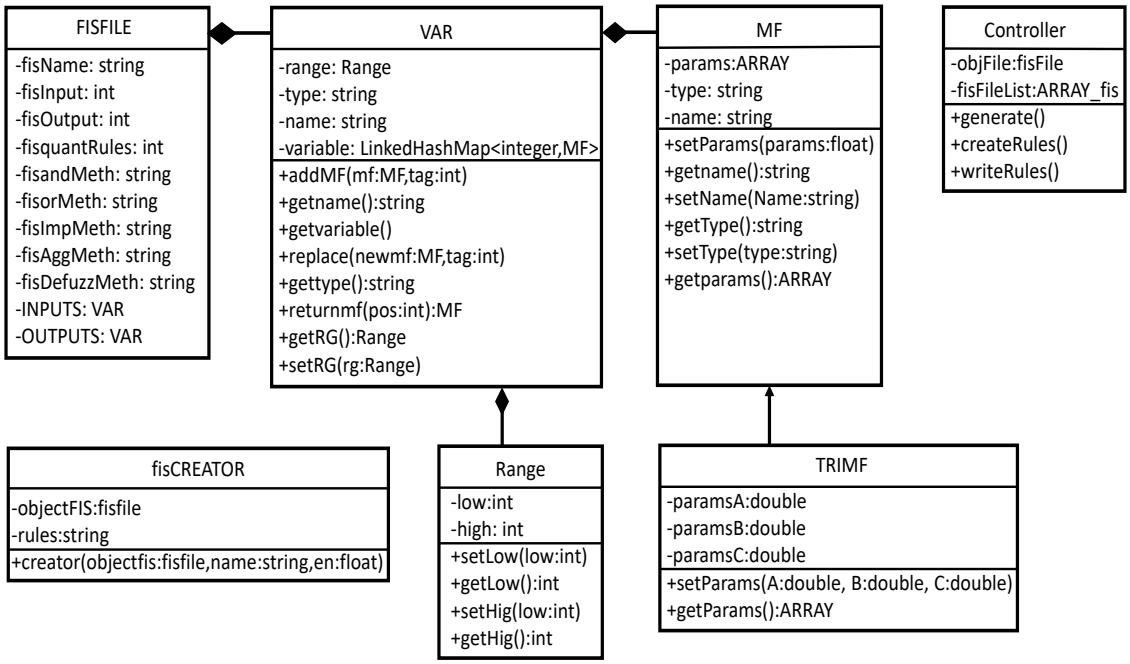

Figura 3.3: Diagrama de Clases del Modulo desarrollado en Java^{TM}.

La interfaz gráfica de usuario (GUI) permite recibir los datos que describen al CLD, los cuales son utilizados para hacer instancias de un objeto de la clase *FISFILE*. La Figura 4.1 corresponde a la GUI que permite introducir el nombre del CLD, variables lingüisticas de entrada y salida, funciones de membresía, los métodos que se aplicarán para la inferencia y la defuzificación; La GUI mostrada en la Figura 4.2 permite capturar cada una de las reglas y un número nombrado *Epsilon* (ϵ) que el usuario decide e indica el grado de variación que se aplica en los parámetros de funciones de membresía cuando se crean la familia de CLDs.

3.5. Metodología Propuesta

La metodología utilizada para la solución del problema planteado (en el capítulo 1) se resume como sigue::

Teniendo el conocimiento de expertos expresado como un SD(representado de manera lingüística mediante las reglas, operadores y MF de tipo triangulares), se puede obtener una familia de CLDs haciendo modificaciones a los parámetros a y b de dichas funciones, usando el siguiente procedimiento:

1) Introducir datos descriptivos y valores numéricos del sistema lógico difuso inicial, ver Cuadro 4.1 y el valor de epsilon ϵ definido de la forma descrita en la sección

3.4.2 para obtener una familia de CLD.

2) Usar la familia de CLD que se obtienen del módulo I para realizar la simulación de cada controlador en lazo cerrado que se realiza en Módulo II, descrito en la sección 3.4.2.

3) Analizar las gráficas de control obtenidos en Módulo II.

Capítulo 4

OBTENCIÓN DE CLDs y SIMULACIÓN EN LAZO CERRADO

Mendiante las interfaces de usuario que se muestran en la Figura 4.1 y la Figura 4.2, los datos numéricos y descriptivos del controlador inicial diseñado por un experto y representado en el Cuadro 4.1 son introducidos al software con un valor de ($epsilon = 0{,}02$) fijo, mediante el software se realiza la variación de los valores a y b de las funciones de pertenencia de cada variable lingüística, obteniendo de esta forma la familia de controladores que se observan en el Cuadro 4.2 en términos de los parámetros de funciones de membresía.

```
Name ='error'
Range=[-1.57 1.57]
MF_1 ='negativo':'trimf',[-2.827 -1.571 0.0]
MF_2 ='positivo':'trimf',[0.0 1.571 2.827]
MF_3 ='cero':'trimf',[-0.1 0.0 0.1]

Name ='control'
Range=[-1.0 1.0]
MF_1 ='negativo':'trimf',[-1.8 -1.0 0.0]
MF_2 ='positivo':'trimf',[0.0 1.0 1.8]
MF_3 ='cero':'trimf',[-0.1 0.0 0.1]
```

R_1 :Si ξ es negativo Entonces u es negativo
R_2 :Si ξ es cero Entonces u es cero
R_3 :Si ξ es positivo Entonces u es positivo

Cuadro 4.1: Representación Lingüística del CLD inicial.

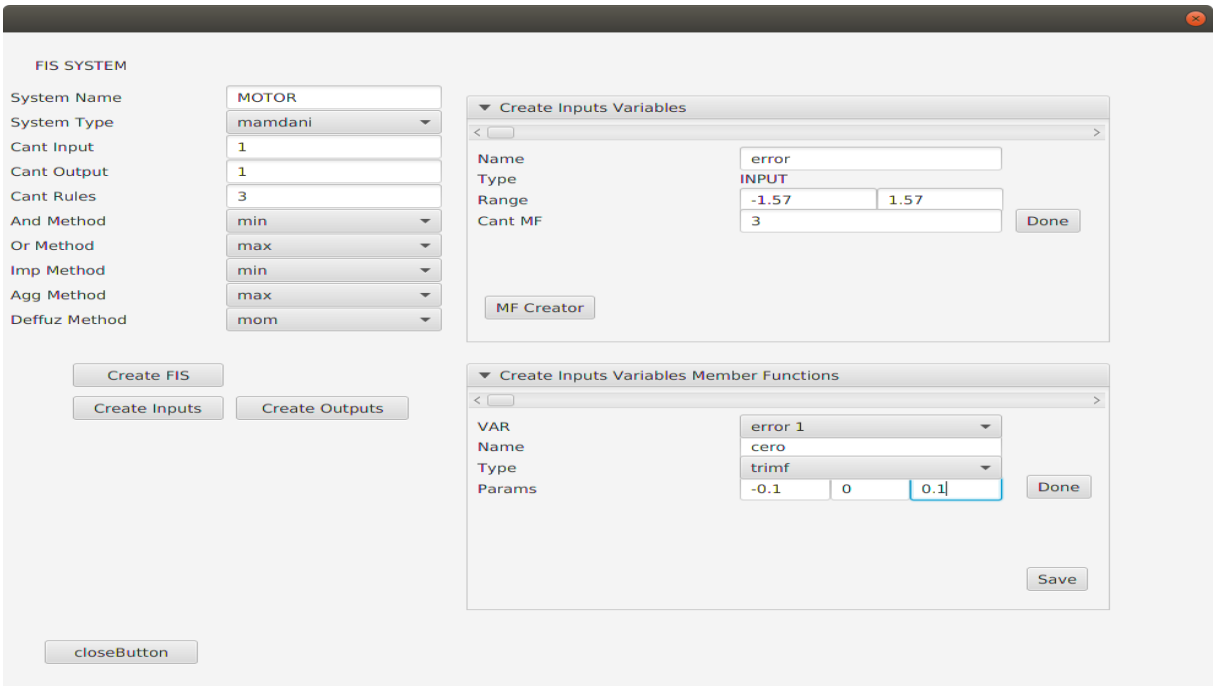

Figura 4.1: GUI del módulo I para introducir variables lingüísticas del CLD inicial.

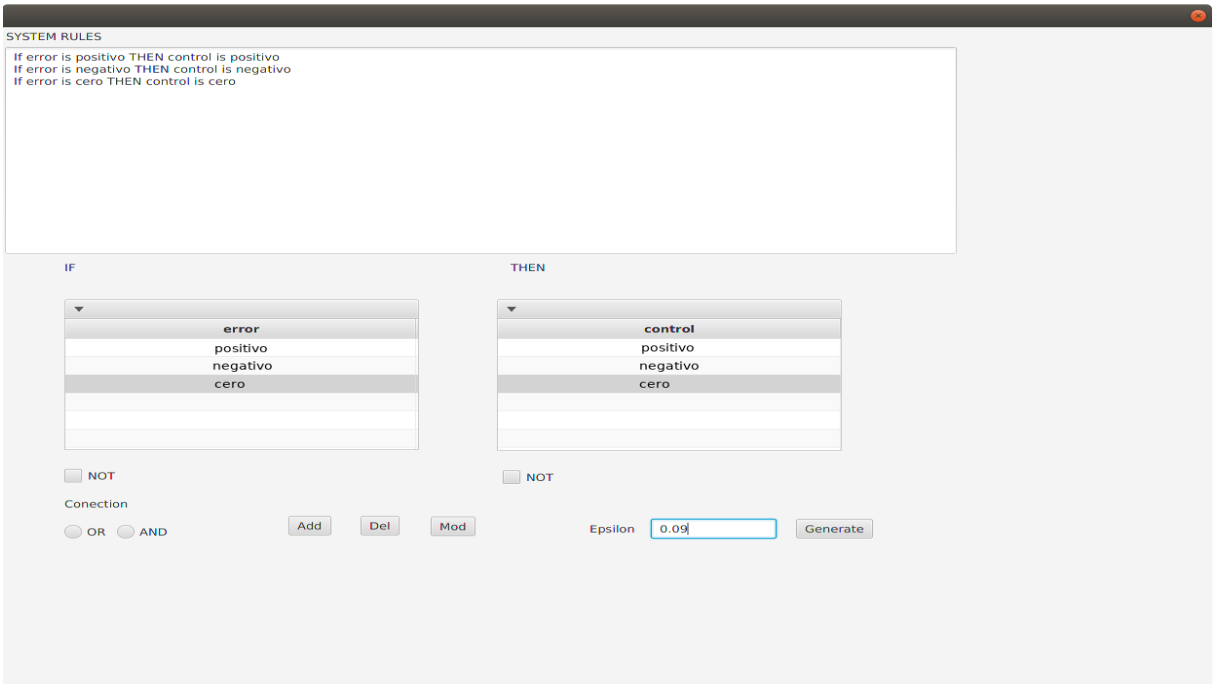

Figura 4.2: GUI del módulo I para introducir la base de reglas del CLD inicial.

4.0.1. Conjunto de CLDs.

En 4.1 se observa la estructura básica del CLD inicial, donde podemos apreciar la descripción de las variables lingüísticas con su nombre definido mediante el campo *Name*; el campo *Range* delimitado por sus valores extremos define el rango de la variable; y sus funciones de membresía MF_i conformadas por la descripción, el tipo de función(en este caso son de tipo triangular) y sus parámetros regidos por la definición 2.11.

Las variaciones que se obtienen en las funciones de pertenencia de la variable error (ξ) y la variable control (u) al aplicar el (Algoritmo 1), implementado en el método *generate* de la clase *controller* se pueden observar en la Tabla 4.2, que muestra los parámetros de funciones de membresía para las variables de entrada (ξ) y salida (u) de 25 CLDs con los que se realizaron las pruebas en lazo cerrado usando el modulo II.

i	Variable	$[D_{min}, D_{max}]$	**negativo**			**cero**			**positivo**		
			$a_{i,1}$	$b_{i,1}$	$c_{i,1}$	$a_{i,2}$	$b_{i,2}$	$c_{i,2}$	$a_{i,3}$	$b_{i,3}$	$c_{i,3}$
1	ξ	[-1.57, 1.57]	-2.827	-1.571	0.0	-0.1	0.0	0.1	0.0	1.571	2.827
	u	[-1.00, 1.00]	-1.800	-1.000	0.0	-0.1	0.0	0.1	0.09	1.000	1.71
2	ξ	[-1.57, 1.57]	-2.807	-1.571	-0.02	-0.1	0.0	0.1	0.0	1.571	2.827
	u	[-1.00, 1.00]	-1.800	-1.000	0.0	-0.1	0.0	0.1	0.0	1.00	1.800
3	ξ	[-1.57, 1.57]	-2.807	-1.571	-0.02	-0.1	0.0	0.1	0.0	1.571	2.827
	u	[-1.00, 1.00]	-1.78	-1.000	-0.02	-0.1	0.0	0.1	0.0	1.000	1.8
4	ξ	[-1.57, 1.57]	-2.807	-1.571	-0.02	-0.1	0.0	0.1	0.0	1.571	2.827
	u	[-1.00, 1.00]	-1.78	-1.000	-0.02	-0.1	0.0	0.1	0.02	1.000	1.78
5	ξ	[-1.57, 1.57]	-2.807	-1.571	-0.02	-0.1	0.0	0.1	0.0	1.571	2.827
	u	[-1.00, 1.00]	-1.78	-1.000	-0.02	-0.08	0.0	0.08	0.02	1.000	1.78
6	ξ	[-1.57, 1.57]	-2.807	-1.571	-0.02	-0.1	0.0	0.1	0.02	1.571	2.807
	u	[-1.00, 1.00]	-1.78	-1.0	-0.02	-0.08	0.0	0.08	0.02	1.00	1.78
7	ξ	[-1.57, 1.57]	-2.807	-1.571	-0.02	-0.1	0.0	0.1	0.02	1.571	2.807
	u	[-1.00, 1.00]	-1.78	-1.00	-0.02	-0.1	0.0	0.1	0.0	1.000	1.8
8	ξ	[-1.57, 1.57]	-2.807	-1.571	-0.02	-0.1	0.0	0.1	0.02	1.571	2.807
	u	[-1.00, 1.00]	-1.78	-1.0	-0.02	-0.1	0.0	0.1	0.02	1.000	1.78
9	ξ	[-1.57, 1.57]	-2.807	-1.571	-0.02	-0.1	0.0	0.1	0.02	1.571	2.807
	u	[-1.00, 1.00]	-1.78	-1.00	-0.02	-0.08	0.0	0.08	0.02	1.000	1.78
10	ξ	[-1.57, 1.57]	-2.807	-1.571	-0.02	-0.08	0.0	0.08	0.02	1.571	2.807
	u	[-1.00, 1.00]	-1.78	-1.00	-0.02	-0.08	0.0	0.08	0.02	1.000	1.78
11	ξ	[-1.57, 1.57]	-2.807	-1.571	-0.02	-0.08	0.0	0.08	0.02	1.571	2.807

Sigue en la página siguiente.

i	Variable	$[D_{min}, D_{max}]$	**negativo**			**cero**			**positivo**		
			$a_{i,1}$	$b_{i,1}$	$c_{i,1}$	$a_{i,2}$	$b_{i,2}$	$c_{i,2}$	$a_{i,3}$	$b_{i,3}$	$c_{i,3}$
	u	[-1.00, 1.00]	-1.78	-1.000	-0.02	-0.1	0.0	0.1	0.0	1.000	1.800
12	ξ	[-1.57, 1.57]	-2.807	-1.571	-0.02	-0.08	0.0	0.08	0.02	1.571	2.807
	u	[-1.00, 1.00]	-1.78	-1.000	-0.02	-0.1	0.0	0.1	0.02	1.00	1.78
13	ξ	[-1.57, 1.57]	-2.807	-1.571	-0.02	-0.08	0.0	0.08	0.02	1.571	2.807
	u	[-1.00, 1.00]	-1.78	-1.000	-0.02	-0.08	0.0	0.08	0.02	1.000	1.78
14	ξ	[-1.57, 1.57]	-2.847	-1.571	0.02	-0.1	0.0	0.1	0.00	1.571	2.827
	u	[-1.00, 1.00]	-1.800	-1.000	0.0	-0.1	0.0	0.1	0.0	1.000	1.800
15	ξ	[-1.57, 1.57]	-2.847	-1.571	0.02	-0.1	0.0	0.1	0.0	1.571	2.827
	u	[-1.00, 1.00]	-1.82	-1.000	0.02	-0.1	0.0	0.1	0.0	1.000	1.8
16	ξ	[-1.57, 1.57]	-2.847	-1.571	0.02	-0.1	0.0	0.1	0.0	1.571	2.827
	u	[-1.00, 1.00]	-1.82	-1.000	0.02	-0.1	0.0	0.1	-0.02	1.000	1.82
17	ξ	[-1.57, 1.57]	-2.847	-1.571	0.02	-0.1	0.0	0.1	0.0	1.571	2.827
	u	[-1.00, 1.00]	-1.82	-1.000	0.02	-0.12	0.0	0.12	-0.02	1.000	1.82
18	ξ	[-1.57, 1.57]	-2.847	-1.571	0.02	-0.1	0.0	0.1	-0.02	1.571	2.847
	u	[-1.00, 1.00]	-1.82	-1.000	0.02	-0.12	0.0	0.12	-0.02	1.000	1.82
19	ξ	[-1.57, 1.57]	-2.847	-1.571	0.02	-0.1	0.0	0.1	-0.02	1.571	2.847
	u	[-1.00, 1.00]	-1.82	-1.00	0.02	-0.1	0.0	0.1	0.0	1.000	1.8
20	ξ	[-1.57, 1.57]	-2.847	-1.571	0.02	-0.1	0.0	0.1	-0.02	1.571	2.847
	u	[-1.00, 1.00]	-1.82	-1.000	0.02	-0.1	0.0	0.1	-0.02	1.000	1.82
21	ξ	[-1.57, 1.57]	-2.847	-1.571	0.02	-0.1	0.0	0.1	-0.02	1.571	2.847
	u	[-1.00, 1.00]	-1.82	-1.000	0.02	-0.12	0.0	0.12	-0.02	1.000	1.82
22	ξ	[-1.57, 1.57]	-2.847	-1.571	0.02	-0.12	0.0	0.12	-0.02	1.571	2.847
	u	[-1.00, 1.00]	-1.82	-1.000	0.02	-0.12	0.0	0.12	-0.02	1.000	1.82
23	ξ	[-1.57, 1.57]	-2.847	-1.571	0.02	-0.12	0.0	0.12	-0.02	1.571	2.847
	u	[-1.00, 1.00]	-1.82	-1.000	0.02	-0.1	0.0	0.1	0.0	1.000	1.800
24	ξ	[-1.57, 1.57]	-2.847	-1.571	0.02	-0.12	0.0	0.12	-0.02	1.571	2.847
	u	[-1.00, 1.00]	-1.82	-1.000	0.02	-0.1	0.0	0.1	-0.02	1.000	1.82
25	ξ	[-1.57, 1.57]	-2.847	-1.571	0.02	-0.12	0.0	0.12	-0.02	1.571	2.847
	u	[-1.00, 1.00]	-1.82	-1.000	0.02	-0.12	0.0	0.12	-0.02	1.000	1.82

Cuadro 4.2: Conjunto de CLDs generados por el software.

4.0.2. Simulacion en lazo cerrado.

Motor de corriente directa.

Considérese el modelo en espacio de estados de un motor de corriente directa (CD) dado como:

$$\begin{pmatrix} \dot{x_1} \\ \dot{x_2} \end{pmatrix} = \begin{pmatrix} -\frac{f_m}{J_m} & \frac{K_a}{J_m} \\ -\frac{K_b}{L_a} & -\frac{R_a}{L_a} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{L_a} \end{pmatrix} u, \tag{4.1}$$

$$y = (0\ 1) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}. \tag{4.2}$$

Donde $[x_1, x_2]^T$ es el vector de estados que representa la posición y la velocidad angular y los párametros $f_m, K_a, J_m K_b, R_a$ y L_a del motor, se muestran en Tabla 4.3.

Cuadro 4.3: Parámetros del motor de corriente directa.

Constante	Parámetro	Valor
J_m	Momento de inercia del motor	0,01 $kg \cdot m^2$
f_m	Fricción viscosa	0,1 $N \cdot m \cdot s$
K_b	Fuerza electromotriz	0,01 $v \cdot s/rad$
K_a	Torque	0,01 $N \cdot m/Amp$
R_a	Resistencia eléctrica	1 Ohm
L_a	Inductancia eléctrica	0,5 H

Considerando el sistema descrito en (4.2), el objetivo es determinar un comando de control u tal que el sistema sea llevado a $\theta_d = \pi/2$ rad. utilizando un sistema SISO con error como variable de entrada, definido como $\xi = y - \theta_d$ siendo θ_d la posición deseada y u como señal de control a la salida; por lo que el problema de control a resolver es $\lim_{t\to\infty} \|\xi(t)\| = \lim_{t\to\infty} \|y(t) - \theta_d\| = 0$.

Resultados.

Para evaluar la familia de CLDs obtenida, y conocer si responden al control, se hace uso del módulo implementado en Matlab^{TM}, se comprueba el comportamiento del sistema con cada controlador generado y se grafica la respuesta en lazo cerrado de cada uno de ellos.

Los resultados que arrojan las pruebas del software aplicados al sistema de motor de corriente directa se muestra en la Figura. 4.3, donde se puede observar la evolución de las trayectorias que resultan de usar los CLDs para el control de posición, estas trayectorias

permiten identificar que los controladores tienen un desempeño muy similar que varia levemente desde el 0,5 seg. y es notorio que se cumple con el objetivo de control.

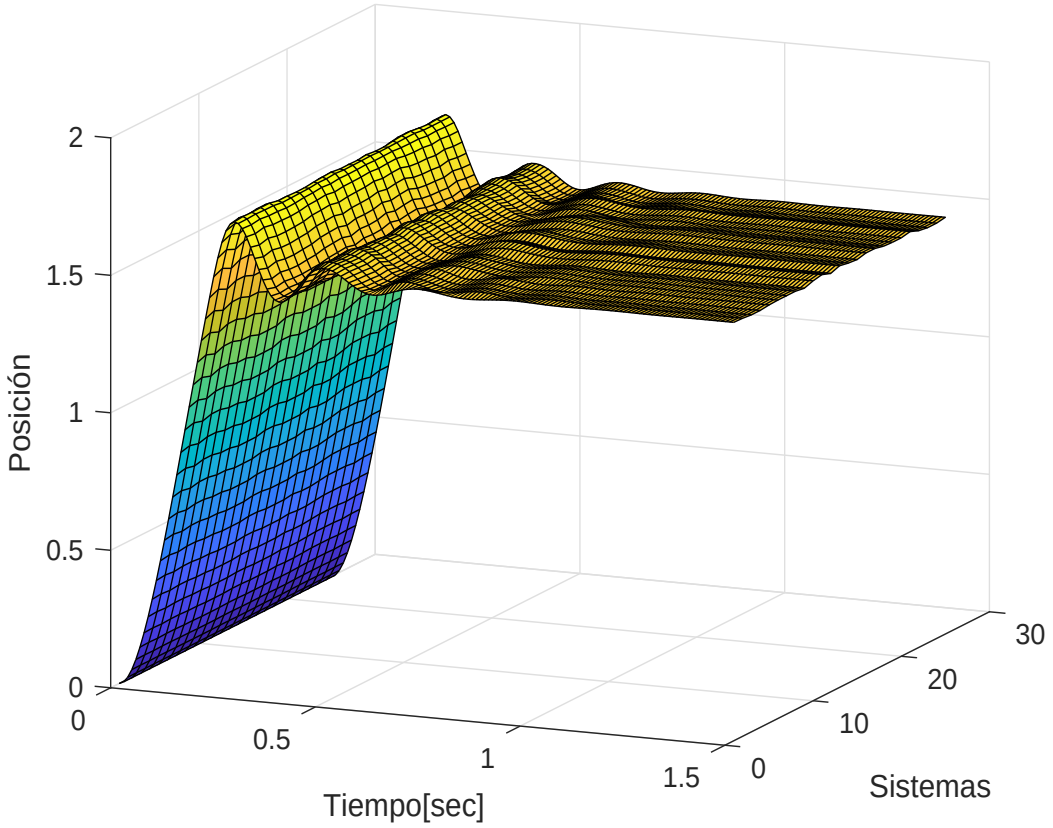

Figura 4.3: Respuesta en el tiempo de la variable *error* usando cada CLD generado.

Sistema pendular simple.

Se considera experimentar y realizar simulaciones sobre un péndulo físico, el cual que se encuentra regido por la siguiente ecuación diferencial de segundo orden

$$ml^2\ddot{\theta} = -mgl\sin(\theta) - f_v\dot{\theta} + Ku(t), \tag{4.3}$$

siendo $\theta(t) \in [-2\pi, 2\pi]$ es la posición angular del péndulo, $\dot{\theta}(t) \in \mathbb{R}$ es la velocidad angular, $u(t) \in \mathbb{R}$ es la entrada de control, $t \in \mathbb{R}^+$ es el tiempo, $m > 0$ es la masa de la carga, $l > 0$ denota la longitud del péndulo, $f_v > 0$ es el coeficiente de fricción viscosa, g es la constante de aceleración de la gravedad, KT es la constante motor-torque. Los parámetros del péndulo se encuentran resumidos en el Cuadro 4.4.

Cuadro 4.4: Especificaciones del péndulo simple.

Descripción	Valor	Unidades
Coeficiente de fricción viscosa (f_v)	2×10^{-2}	N × m × s
Constante de gravedad (g)	9.81	m/s^2
Longitud del péndulo (l_{ps})	0.19304	m
Masa del péndulo (m_{ps})	0.118	kg
Constante de Torque (KT)	0.25	N·m
Entrada de control $u(t)$	$[-10\ 10]$	V

Se considera que $\sin(x) \simeq x$ para ángulos pequeños y se introduce un cambio de coordenadas tal que $x_1(t) = \theta$ y $x_2(t) = \dot{\theta}$. Entonces el sistema (4.3) puede ser representado en espacio de estados tal que:

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{g}{l} & -\frac{f_v}{ml^2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{K}{ml^2} \end{bmatrix} u(t), \tag{4.4}$$

Considerando el sistema descrito en (4.4), el objetivo es determinar un comando de control u tal que el sistema sea llevado a $\theta_d = \pi/4$ rad. utilizando un sistema SISO con error como variable de entrada, definido como $\xi = y - \theta_d$ siendo θ_d la posición deseada y u como señal de control a la salida; por lo que el problema de control a resolver es $\lim_{t\to\infty} \|\xi(t)\| = \lim_{t\to\infty} \|y(t) - \theta_d\| = 0$.

Resultado de simulación del sistema en lazo cerrado.

La evolución en el tiempo de las trayectorias de la posición se muestran la Figura. 4.4 para el conjunto de CLDs que fueron generados por el software. Se puede observar que todos logran realizar un control cercano al objetivo, establecido en la posición $\theta_d = \pi/4$

rad, aunque se aprecia que tiene un sobrepico muy grande, no obstante el desempeño en estado estable de los controladores es muy similar en todos los casos.

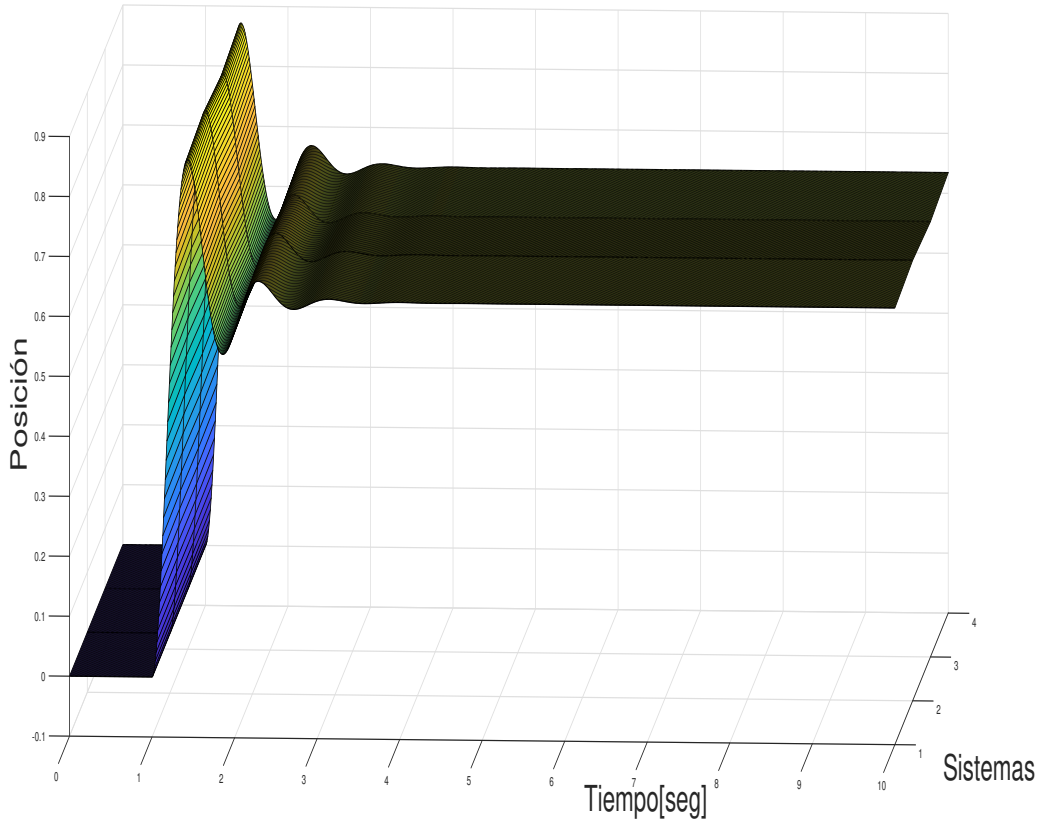

Figura 4.4: Trayectoria de la posición del péndulo simple usando la familia de CLDs.

Sistema de tanque abierto.

Se experimentó con la familia de CLD's en Simulink, para observar el comportamiento de los controles diseñados en lazo cerrado ante una entrada en escalón con el modelo de la función de transferencia para un tanque abierto.

La ecuación diferencial que representa el comportamiento de nivel de un tanque abierto con entrada constante es:

$$A\dot{h} = K_1 a_1 - K_2 a_2 \sqrt{2gh}, \tag{4.5}$$

Donde $h \in [0, 1]$, K_1 y K_2 son las constantes K_v de las válvulas y a_1, a_2 la abertura respectivamente, g es la constante de aceleración de la gravedad.

Cuadro 4.5: Datos del Sistema de Tanque Simple.

Descripción	Valor	Unidades
Área base(A)	140	cm^2
Volumen (V)	14000	cm^3
Constante de gravedad (g)	9.81	m/s^2
Tiempo (t)	60	s
$Kv_1(K_1)$	0,05	m^3/s
$Kv_2(K_2)$	0,015	m^3/s
Abertura 1(a_1)	0,6	-
Abertura 2(a_2)	0,5	-

Aplicando transformada de Laplace en la ecuación la ecuación (4.5) obtenemos la función de transferencia que está definida en (4.6), con los datos numéricos y parámetros que se encuentran resumidos en el Cuadro 4.5.

$$G(s) = \frac{1}{700s + 1/1.98}. \tag{4.6}$$

El objetivo es determinar un comando de control u tal que el sistema sea llevado a un nivel cercano al 80 % de su capacidad total, utilizando un sistema SISO con error como variable de entrada definido como $\xi = y - \theta_d$ siendo θ_d la altura deseada y u como señal de control a la salida.

Resultados de simulación del sistema en lazo cerrado.

El sistema de control implementado pretende mantener el nivel del tanque al 80 %. Las evolución de las trayectorias de nivel se muestran en la Figura. 4.5 para el conjunto de CLDs que fueron generados por el software, se puede observar que todos mantienen el nivel cercanamente a la posición deseada, se puede identificar que el desempeño en estado

estable de los controladores no es igual pero se observa que se mantienen cercanos en su respuesta de control.

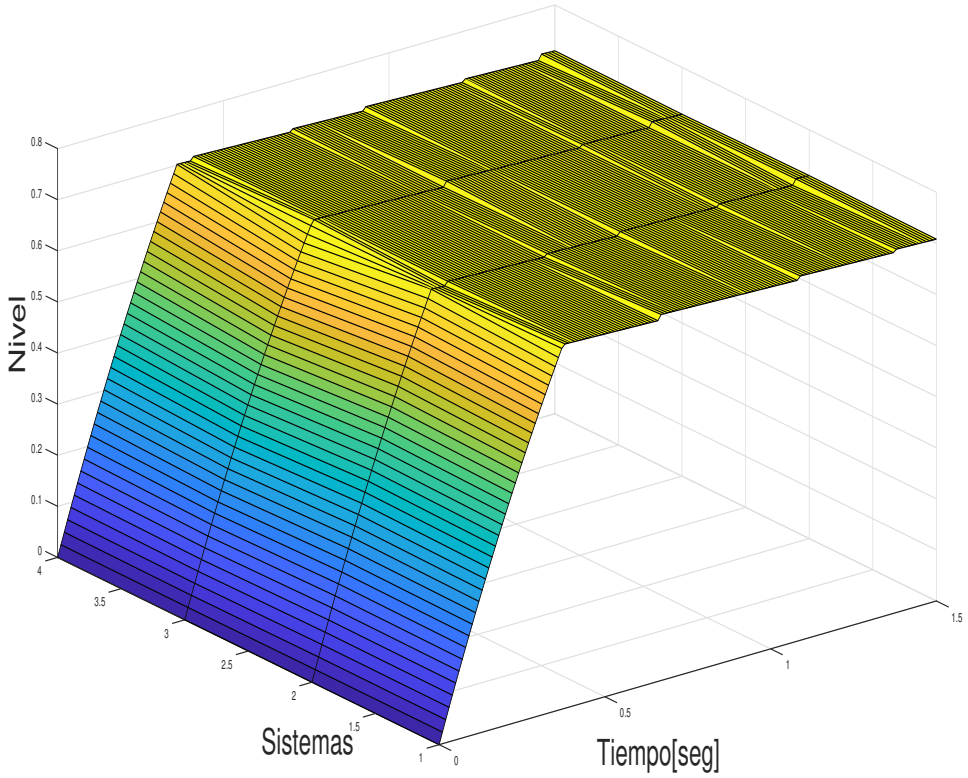

Figura 4.5: Respuesta en el tiempo de la variable *nivel* usando cada CLD generado.

Capítulo 5

CONCLUSIONES

Basados en [9], donde los autores exponen que, al realizar una modificación uniforme de los parámetros de las funciones de pertenencia hasta cierto límite, el sistema en lazo cerrado mantiene muchas de sus propiedades, pero varía en otras como el rendimiento, el presente trabajo de tesis se mostró una metodología para la construcción de un conjunto de CLDs a partir de un modelo diseñado por un experto haciendo una variación de parámetros de forma iterativa y se implementó esta metodología mediante un software. El software es susceptible a convertirse en asistencia durante el ciclo de diseño de controladores.

La metodología fue probada para la construcción de una familia de CLDs y el software implementado fue probado para resolver el problema de control de regulación para tres sistemas que se estudian en ingeniería y son utilizados como parte de procesos industriales. Se diseñó el sistema de inferencia difusa tipo Mamdani que agrupa el conocimiento experto sobre control difuso tipo proporcional y se aplicó la metodología para obtener mediante el software la familia de controladores.

Los resultados obtenidos muestran que el método propuesto de implementar en un software la variación de parámetros para la obtención de una familia de CLDs a partir de un modelo de SD desarrollado por un experto, incorpora una gama de soluciones de control, pudiéndose comprobar mediante las gráficas de respuestas de control en lazo cerrado que pueden ser aplicados a diferentes tipos de problemas obteniendo variedad de soluciones de los cuales el usuario podría seleccionar según determinados criterios que se siguen para medir respuestas de control.

5.1. Trabajo futuro

Se puede probar la metodología propuesta en modelos de control que no se limiten a la solución difusa proporcional, se pudiera incluir difuso PI, PD o PID para el control de los modelos que sean escogidos. Otra arista de investigación corresponde a extender el estudio de otros métodos de variación de parámetros en funciones de pertenencia distinta a la triangular, como por ejemplo aquellos que utilizan funciones de membresía no lineales tales como gaussianas, sigmoidales, entre otras.

Para la construcción de los CLDs pueden ser utilizadas técnicas de IA como pueden ser algoritmos genéticos o redes neuronales. La principal ventaja de estas técnicas consiste en que se pueden obtener de forma automatizada algunas de las características deseadas por el usuario, como puede ser el controlador óptimo o el que menor sobrepico tiene u otras características que describen la respuesta del sistema en lazo cerrado, también se pudiera de alguna manera implícita obtener validación estadística en cuanto al tiempo de respuesta, sobrepico o variación del error de la respuesta del conjunto CLDs.

Bibliografía

[1] Application architecture for .net: Designing application and services. *http://www.microsoft.com/technet/treeview/default.asp?*, pages 10–25, 08-2003.

[2] Hassane Abouaïssa, Michel Fliess, Violina Iordanova, and Cédric Join. Freeway ramp metering control made easy and efficient. *arXiv preprint arXiv:1206.5937*, 2012.

[3] Ismail H Altas and Adel M Sharaf. A generalized direct approach for designing fuzzy logic controllers in matlab/simulink gui environment. *International journal of information technology and intelligent computing*, 1(4):1–27, 2007.

[4] R Babuska and HB Verbruggen. Identification of composite linear models via fuzzy clustering. In *Proceedings of European Control Conference, Rome, Italy*, pages 1207–1212, 1995.

[5] Robert Babuska. Fuzzy and neural control. disc course lecture notes. *Delft University of Technology. Delft, the Netherlands*, 2001.

[6] Ying Bai and Dali Wang. Fundamentals of fuzzy logic control fuzzy sets, fuzzy rules and defuzzifications. In *Advanced Fuzzy Logic Technologies in Industrial Applications*, pages 17–36. Springer, 2006.

[7] M Bilal Kadri. Model free adaptive fuzzy control: beginners approach. *Doctoral, Oxford University*, 2009.

[8] George Calcev. Some remarks on the stability of mamdani fuzzy control systems. *IEEE Transactions on Fuzzy Systems*, 6(3):436–442, 1998.

[9] Oscar Castillo, Luís Aguilar, Nohé Cázarez, and Selene Cárdenas. Systematic design of a stable type-2 fuzzy logic controller. *Applied Soft Computing Journal*, pages 1274–1279, 6 2008. ISSN 1568-4946. doi: 10.1016/j.asoc.2007.02.021.

[10] Nohe R Cazarez-Castro, Luis T Aguilar, and Oscar Castillo. Fuzzy logic control with genetic membership function parameters optimization for the output regulation of a servomechanism with nonlinear backlash. *Expert Systems with Applications*, 37(6): 4368–4378, 2010.

[11] Nohe R Cazarez-Castro, Luis T Aguilar, and Oscar Castillo. Designing type-1 and type-2 fuzzy logic controllers via fuzzy lyapunov synthesis for nonsmooth mechanical systems. *Engineering Applications of Artificial Intelligence*, 25(5):971–979, 2012.

[12] Nohe R Cazarez-Castro, Luis T Aguilar, Selene L Cardenas-Maciel, Carlos A Goribar-Jiménez, and Mauricio Odreman-Vera. Diseño de un controlador difuso mediante la síntesis difusa de lyapunov para la estabilización de un péndulo de rueda inercial. *Revista Iberoamericana de Automática e Informática industrial*, 14(2):133–140, 2017.

[13] Guanrong Chen and Trung Tat Pham. *Introduction to fuzzy systems*. Chapman and Hall/CRC, 2005.

[14] Leandro dos Santos Coelho, Marcelo Wicthoff Pessôa, Rodrigo Rodrigues Sumar, and Antonio Augusto Rodrigues Coelho. Model-free adaptive control design using evolutionary-neural compensator. *Expert Systems with Applications*, 37(1):499–508, 2010.

[15] Luisa F Escobar-Dávila, Oscar D Montoya-Giraldo, and Didier Giraldo-Buitrago. Control global del péndulo de furuta empleando redes neuronales artificiales y realimentación de variables de estado. *TecnoLógicas*, pages 71–94, 2013.

[16] Isabelle Fantoni and Rogelio Lozano. Control of nonlinear mechanical systems. *European Journal of Control*, 7(2-3):328–348, 2001.

[17] Jaider Manuel Fernandez Torres. Sun certified java programmer. *url = "https://repository.uniminuto.edu/"*, 2008.

[18] A Fink, S Töpfer, and R Isermann. Nonlinear model-based control with local linear neuro-fuzzy models. *Archive of Applied Mechanics*, 72(11-12):911–922, 2003.

[19] Michel Fliess and Cédric Join. Intelligent pid controllers. In *2008 16th Mediterranean Conference on Control and Automation*, pages 326–331. IEEE, 2008.

[20] R Gonzalez, A Torralba, and LG Franquelo. Afan, a tool for the automatic synthesis of neural and fuzzy controllers with architecture optimization. In *Proceedings of 1997*

IEEE International Symposium on Circuits and Systems. Circuits and Systems in the Information Age ISCAS'97, volume 1, pages 637–640. IEEE, 1997.

[21] Thomas Hollstein, Saman K Halgamuge, and Manfred Glesner. Computer-aided design of fuzzy systems based on generic vhdl specifications. *IEEE Transactions on Fuzzy Systems*, 4(4):403–417, 1996.

[22] Zhongsheng Hou and Shangtai Jin. Model-free adaptive control for a class of nonlinear discrete-time systems based on the partial form linearization. *IFAC Proceedings Volumes*, 41(2):3509–3514, 2008.

[23] Margarita Jiménez and Iván Sarmiento. Sistema adaptativo de control y optimización del tráfico de un corredor vial semaforizado. aplicación a la ciudad de medellín. *url=https://www.redalyc.org/*, 78(169):71–78, 2011.

[24] Yuney Gorrin Ortega Jorge Antonio Lopez Renteria, Nohe Ramon Cazarez Castro. Construcción del coeficiente en $\mathcal{F}_{\mathcal{R}}$ para la ecuación diferencial difusa x' = kx. *Master Thesis Instituto Tecnológico de Tijuana*, 2019.

[25] James M Kelly. The role of damping in seismic isolation. *Earthquake engineering & structural dynamics*, 28(1):3–20, 1999.

[26] Daijin Kim. An implementation of fuzzy logic controller on the reconfigurable fpga system. *IEEE Transactions on industrial Electronics*, 47(3):703–715, 2000.

[27] Sung-Woo Kim and Ju-Jang Lee. Design of a fuzzy controller with fuzzy sliding surface. *Fuzzy sets and systems*, 71(3):359–367, 1995.

[28] Miranda La Hera and Pedro Xavier. *Contributions to motion planning and orbital stabilization: case studies: Furuta pendulum swing up, inertia wheel oscillations and biped robot walking*. PhD thesis, Umeå university, 2008.

[29] Jihong Lee. On methods for improving performance of pi-type fuzzy logic controllers. *IEEE transactions on fuzzy systems*, 1(4):298–301, 1993.

[30] Han-Xiong Li and HB Gatland. A new methodology for designing a fuzzy logic controller. *IEEE transactions on systems, man, and cybernetics*, 25(3):505–512, 1995.

[31] Han-Xiong Li and HiB Gatland. Conventional fuzzy control and its enhancement. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(5): 791–797, 1996.

[32] John H Lilly. *Fuzzy control and identification*. John Wiley & Sons, 2011.

[33] Fernando Castanos Luna. Levantamiento y estabilización del péndulo invertido. *Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, Departamento de Control Automático url=https://www.academia.edu/*, 2003.

[34] O.; Poulsen N. K.; Hansen L. K. Norgaard, M.; Ravn. *Neural Networks for Modelling and Control of Dynamic Systems*. Springer-Verlag, 2000.

[35] M Olivares and Ricardo Rojas. Modelado y control difuso. *Dpto. Electrónica, Universidad Técnica Federico Santa María*, pages 6–9, 2002.

[36] Javier Ollervides, Víctor Santibáñez, Miguel Llama, and Alejandro Dzul. Aplicación de control borroso a un sistema de suspensión magnética: Comparación experimental. *Revista Iberoamericana de Automática e Informática Industrial RIAI*, 7(3):63–71, 2010.

[37] Oracle. Taking stock of java's contributions to development and digital transformation. *url = "https://www.oracle.com/a/ocom/resources/java-turns-25.pdf"*, 2012.

[38] Microsoft Press, Lin Joyner, and Graeme Malcolm. *Application Architecture for. NET: Designing Applications and Services*. Microsoft Press, 2002.

[39] Pablo J Prieto and Nohe R Cazarez-Castro. Aplicaciones de la inteligencia computacional en el control de robots neumáticos. *Una primera aproximación a la Interfaz de Cómputo Cerebral*, 2016.

[40] Guoyuan Qi, Zengqiang Chen, and Zhuzhi Yuan. Model-free control of affine chaotic systems. *Physics Letters A*, 344(2-4):189–202, 2005.

[41] Agustín Froufe Quintas. *Java 2: manual de usuario y tutorial*. Grupo Editorial RA-MA, 2000.

[42] Vesna M Ranković and Ilija Ž Nikolić. Application of the takagi-sugeno fuzzy controller for solving the robots' inverse kinematics problem. *Facta universitatis-series: Mechanics, Automatic Control and Robotics*, 3(15):1039–1054, 2003.

[43] Sebastián Fernando Puente Reyes, Cesar Andrey Perdomo Charry, Elvis Eduardo Gaona García, et al. Generación automática de sistemas lógicos difusos (sld) tipo mamdani sobre microcontrolador de 8 bits. *Tecnura*, 17:93–108, 2013.

[44] Leonid Reznik. *Fuzzy Controllers*. Newnes, 1997.

[45] Santiago Sánchez-Solano and M Brox. Síntesis automática de sistemas difusos mediante xfuzzy. *uri = "http://hdl.handle.net/10261/84030"*, 2012.

[46] Victor Santibañez, Rafael Kelly, and Miguel A Llama. Global asymptotic stability of a tracking sectorial fuzzy controller for robot manipulators. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 34(1):710–718, 2004.

[47] Guanrong Chen Trung Tat Pham. *Introduction to fuzzy sets, fuzzy logic, and fuzzy control systems*. CRC press, 2000.

[48] Lotfi A Zadeh et al. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.

Apéndice A

Módulo I Código *Java*®

A continuación se muestran las clases que interactuan para obtener la familia de CLDs representados en archivos con extensión .fis.

```
package classPackage;
import FisGen.Main;
import javafx.beans.property.IntegerProperty;
import javafx.beans.property.SimpleIntegerProperty;
import javafx.beans.property.SimpleStringProperty;
import javafx.beans.property.StringProperty;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;

import java.util.LinkedList;
import java.util.Map;
import java.util.HashMap;
import javafx.collections.ObservableMap;
import javafx.collections.MapChangeListener;


public class fisFile {



    private StringProperty   fisName;

    public void setFisName(String value) { fisNameProperty().set(value); }

    public String getFisName() { return fisNameProperty().get(); }

    public StringProperty fisNameProperty() {
        if (fisName == null) fisName = new SimpleStringProperty(this, "fisName");
        return fisName;
    }

    private StringProperty   fisType;

    public void setFisType(String value) { fisNameProperty().set(value); }

    public String getType() { return fisNameProperty().get(); }

    public StringProperty fisTypeProperty() {
        if (fisType == null) fisType = new SimpleStringProperty(this, "fisType");
        return fisType;
    }

    private  IntegerProperty  fisInputs;

    public void setFisInputs(Integer value) { FisInputsProperty().set(value); }
```

```
public Integer getfisInputs() { return FisInputsProperty().get(); }

public IntegerProperty FisInputsProperty() {
    if (fisInputs == null) fisInputs = new SimpleIntegerProperty(this, "fisName");
    return fisInputs;
}

private IntegerProperty  fisOutputs;

public void setfitOutInputs(Integer value) { FisInputsProperty().set(value); }

public Integer getfisOutputs() { return FisInputsProperty().get(); }

public IntegerProperty FisOutProperty() {
    if (fisOutputs == null) fisOutputs = new SimpleIntegerProperty(this, "fisOutputs"
        );
    return fisOutputs;
}

private  IntegerProperty  fisquantRules;

public void setfisquantRules(Integer value) { fisquantRulesProperty().set(value); }

public Integer getfisquantRules() { return fisquantRulesProperty().get(); }

public IntegerProperty fisquantRulesProperty() {
    if (fisquantRules == null) fisquantRules = new SimpleIntegerProperty(this, "
        fisquantRules");
    return fisquantRules;
}

private StringProperty   fisAndMethod;

public void setfisAndMethod(String value) { fisAndMethodProperty().set(value); }

public String getfisAndMethod() { return fisAndMethodProperty().get(); }

public StringProperty fisAndMethodProperty() {
    if (fisAndMethod == null) fisAndMethod = new SimpleStringProperty(this, "
        fisAndMethod");
    return fisAndMethod;
}


private StringProperty   fisOrMethod;

public void setfisOrMethod(String value) { fisOrMethodProperty().set(value); }

public String getfisOrMethod() { return fisOrMethodProperty().get(); }

public StringProperty fisOrMethodProperty() {
    if (fisOrMethod == null) fisOrMethod = new SimpleStringProperty(this, "
        fisOrMethod");
    return fisOrMethod;
}


private StringProperty   fisImpMethod;

public void setfisImpMethod(String value) { fisImpMethodProperty().set(value); }

public String getfisImpMethod() { return fisImpMethodProperty().get(); }

public StringProperty fisImpMethodProperty() {
    if (fisImpMethod == null) fisImpMethod = new SimpleStringProperty(this, "
        fisImpMethod");
    return fisImpMethod;
```

```
    }

    private StringProperty   fisAggMethod;

    public void setfisAggMethod(String value) { fisAggMethodProperty().set(value); }

    public String getfisAggMethod() { return fisAggMethodProperty().get(); }

    public StringProperty fisAggMethodProperty() {
        if (fisAggMethod == null) fisAggMethod = new SimpleStringProperty(this, "
            fisAggMethod");
        return fisAggMethod;
    }


    private StringProperty   fisDefuzzMethod;

    public void setfisDefuzzMethod(String value) { fisDefuzzMethodProperty().set(value);
        }

    public String getfisDefuzzMethod() { return fisDefuzzMethodProperty().get(); }

    public StringProperty fisDefuzzMethodProperty() {
        if (fisDefuzzMethod == null) fisDefuzzMethod = new SimpleStringProperty(this, "
            fisDefuzzMethod");
        return fisDefuzzMethod;
    }



    public HashMap <Double, VAR> INPUT  = new HashMap <Double, VAR> ();
    public HashMap <Double, VAR> OUTPUT = new HashMap <Double, VAR> ();
    public  String reglas;
    public LinkedList<String> reglasFis=new LinkedList<>();

    public fisFile() {
    }

    public String getReglas() {
        return reglas;
    }

    public void setReglas(String reglas) {
        this.reglas = reglas;
    }

    public fisFile(String fisName, String fisType, Integer fisInputs, Integer fisOutputs,
         Integer fisRules, String fisAndMethod, String fisOrMethod, String fisImpMethod,
        String fisAggMethod, String fisDefuzzMethod) {
        // Initial example DATA
        this.fisName= new SimpleStringProperty(fisName);
        this.fisType= new SimpleStringProperty(fisType);

        this.fisInputs       = new SimpleIntegerProperty(fisInputs) ;
        this.fisOutputs      = new SimpleIntegerProperty(fisOutputs);
        this.fisquantRules   = new SimpleIntegerProperty(fisRules);
        this.fisAndMethod    = new SimpleStringProperty(fisAndMethod);
        this.fisOrMethod     = new SimpleStringProperty(fisOrMethod);
        this.fisImpMethod    = new SimpleStringProperty(fisImpMethod);
        this.fisAggMethod    = new SimpleStringProperty(fisAggMethod);
        this.fisDefuzzMethod = new SimpleStringProperty(fisDefuzzMethod);


    }


}
```

```
package classPackage;

import java.util.LinkedHashMap;
import java.util.Map;
import java.util.Map.Entry;

public class VAR {

    private  String name;

    private  LinkedHashMap<Integer, MF> variable;
    private  String type;
    private  range rg;

    public VAR(String name, String type, range rg) {
        this.name = name;
        this.type = type;
        this.rg = rg;
        variable = new LinkedHashMap<>();
    }

    public  void  addMF(MF mf, Integer tag){
        variable.put(tag,mf);
    }


    public String getName() {
        return name;
    }

    public LinkedHashMap <Integer, MF> getVariable() {
        return variable;
    }

    public void replace(MF newMf, Integer tag){
        variable.put(tag,newMf);


    }

    public String getType() {
        return type;
    }

    public MF returnMF(Integer pos){


        return (MF) variable.get(pos);
    }

    public range getRg() {
        return rg;
    }

    public void setRg(range rg) {
        this.rg = rg;
    }

    public Integer cantMF(){
        return this.variable.size();
    }
}
```

```
package classPackage;
```

```
import java.util.DoubleSummaryStatistics;
import java.util.LinkedList;
import java.util.Queue;

public class MF {

 private String name;
 private String type;
 public Queue <Double> params=new LinkedList();

    public MF(String name, String type) {
        this.name = name;
        this.type = type;
    }

    public String  paramsToString(){
        String cadena="";
        while(!this.params.isEmpty()){
              cadena+=this.params.poll()+" ";
        }

        return cadena;
    }

    public void setParams(Double param) {
        this.params.add(param) ;
    }

    public void varParams(Double param, int pos){
        Queue<Double> paramtmp = new LinkedList<>();
        Double value=0.0;
        for(int i=0; i<this.params.size();i++){

            if(i==pos){
                value=this.params.poll()+param;
                paramtmp.add(value);
            }else{
                value=this.params.poll();
                paramtmp.add(value);

            }

        }
       this.params=paramtmp;
    }

    public void setParams(Queue<Double> neap) {
        this.params=neap;
    }
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public Queue <Double> getParams() {
        return params;
    }
```

```
    public Double getParams(int j){
        Queue<Double> paramtmp=this.params;
        Double value = 0.0;
        Double valuert=0.0;
        for(int i=0; i<params.size();i++){
            valuert=params.poll();
            paramtmp.add(valuert);
            if(i==j){
                value=valuert;
            }

        }
         this.params = paramtmp;


        return value;
    }
}
```

```
package classPackage;

public class range{
    private int low;
    private int hig;

    public range(int low, int hig) {
        this.low = low;
        this.hig = hig;
    }

    public void setLow(int low) {
        this.low = low;
    }

    public void setHig(int hig) {
        this.hig = hig;
    }

    public int getLow() {
        return low;
    }

    public int getHig() {
        return hig;
    }
}
```

```
package FisGen;

import classPackage.*;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import sun.font.StandardGlyphVector;

import java.net.URL;
import java.util.*;

public class newRulerController implements Initializable {
    public static String Str="[Rules]"+"\n";
    FisNew FisNew_inRules;
    fisFile fisFile_C;
```

```
LinkedList<fisFile> objectFIS= new LinkedList<fisFile>();
@FXML
private Button btnAdd;
@FXML
private Button btnDel;
@FXML
private Button btnMod;
@FXML
private TextArea textAreaRules;
@FXML
private TextField epsilon;

LinkedList<Integer> objectIn= new LinkedList<Integer>();
LinkedList<Integer> objectOut= new LinkedList<Integer>();



private ObservableList<fisVar> listTable= FXCollections.observableArrayList(


);
private ObservableList<fisVar> listTableOut= FXCollections.observableArrayList(


);



@FXML
private TableView<fisVar> tabInput;
@FXML
public TableColumn<fisVar, String> ColInput;
@FXML
public TableView<fisVar> tabOutput;
@FXML
private TableColumn<fisVar, String> ColOutput;


@FXML
private void btnGenerate(ActionEvent event){


    int i = 0, j = 0, ci = 0, co=0;
    int v[] = new int[n];

    fisFile newFile;
    newFile=fisFile_C;
    objectFIS.add(fisFile_C);
    HashMap <Double, VAR> inEntry   = newFile.INPUT;
    HashMap <Double, VAR> ouOuter   = newFile.OUTPUT;

    Double epsilonFis=Double.valueOf(epsilon.getText());

    for (Map.Entry<Double, VAR> entry : inEntry.entrySet()) {
        Double key = entry.getKey();
        VAR value = entry.getValue();
        Iterator it = value.getVariable().entrySet().iterator();

        while (it.hasNext()) {
            Map.Entry e = (Map.Entry) it.next();
            MF tmpMF = (MF) e.getValue();
            while (ci < tmpMF.params.size()) {
                tmpMF.varParams(epsilonFis, ci);
                objectFIS.add(newFile);
                //System.out.println("Ciclo externo  "+tmpMF.getParams(ci).toString()
                    );
                for (j = ci + 1; j < tmpMF.params.size(); j++) {
                    tmpMF.varParams(epsilonFis, j);
```

```
                    objectFIS.add(newFile);

                }

                ci++;
            }

        for (Map.Entry<Double, VAR> entryout : ouOuter.entrySet()) {

            Double keyou = entryout.getKey();
            VAR valueou = entryout.getValue();
            Iterator itou = valueou.getVariable().entrySet().iterator();

             while (itou.hasNext()){
                    Map.Entry o = (Map.Entry) itou.next();
                    MF tmpMFo = (MF) o.getValue();
                    while (co < tmpMFo.params.size()) {
                    tmpMFo.varParams(epsilonFis, co);
                    objectFIS.add(newFile);
                    //System.out.println("Ciclo externo  "+tmpMF.getParams(ci).
                        toString());
                      for (j = co + 1; j < tmpMFo.params.size(); j++) {
                          tmpMFo.varParams(epsilonFis, j);
                          objectFIS.add(newFile);

                    }

                    co++;
                 }

             }




            }
        }
    }

    for (Map.Entry<Double, VAR> entry : ouOuter.entrySet()) {
        Double key = entry.getKey();
        VAR value = entry.getValue();

        Iterator it = value.getVariable().entrySet().iterator();
        while (it.hasNext()) {
            Map.Entry e = (Map.Entry)it.next();
            MF tmpMF=(MF)e.getValue();
            while(ci < tmpMF.params.size())
            {
                tmpMF.varParams(epsilonFis,ci);
                objectFIS.add(newFile);
                //System.out.println("Ciclo externo  "+tmpMF.getParams(ci).toString()
                    );
                for (j = ci + 1; j < tmpMF.params.size(); j++) {
                    tmpMF.varParams(epsilonFis,j);
                    objectFIS.add(newFile);

                }

                ci++;
            }



        }
        // ...
    }
```

```
        Iterator<fisFile> it= objectFIS.iterator();
        while(it.hasNext()) {
            fisFile nombre= it.next();
            //System.out.println(nombre.getFisName());
            //System.out.println(nombre.FisInputsProperty().getValue());
            fisCreator creador;
            creador=new fisCreator(nombre);
        }


    }







    @FXML
    private void btnmodificar(ActionEvent event){

    }
    @FXML
    private void btneliminar(ActionEvent event){




    }

    @FXML
    private void btnCreateRule(ActionEvent event){

        if (objectIn.size()==1){

            //System.out.println("IN "+ objectIn.pollFirst());
            Str+=objectIn.pollFirst().toString()+" ";
        }
        else {
            int i=0;
            while(i<=objectIn.size()+1) {

                //System.out.println(" " +objectIn.pollFirst().toString());
                Str+=objectIn.pollFirst().toString()+","+" " ;
                i++;
            }}

        if (objectOut.size()==1){
            //System.out.println("Out "+objectOut.pollFirst());
            Str+=objectOut.pollFirst().toString()+" ";
        }
        else{
            int o=0;
            while ( o<=objectOut.size()+1) {


                //System.out.println(" "+ objectOut.pollFirst().toString());
                Str+=objectOut.pollFirst().toString()+","+" ";
                o++;
            }}


        StringBuilder builder=new StringBuilder(Str);
        String sCadenaInvertida=builder.reverse().toString();

       // System.out.println("cadena invertida 1 "+sCadenaInvertida);
```

```
        String sCadenaInvertida1=sCadenaInvertida.substring(2);


        StringBuilder strchange=new StringBuilder(sCadenaInvertida1);
        String changed=strchange.reverse().toString();

        Str=changed;

        Str+=" (1) : 1";


       // fisFile_C.setReglas(Str);

        System.out.println(Str);
    }



    @FXML
    public void btnadicionar(ActionEvent event) {

        String rule="IF entry THEN out";


        TablePosition posin = tabInput.getSelectionModel().getSelectedCells().get(0);
        int rowin = posin.getRow();

        TablePosition posout=tabOutput.getSelectionModel().getSelectedCells().get(0);
        int rowout = posout.getRow();

        objectIn.add(rowin);
        objectOut.add(rowout);

        String selectedin = tabInput.getItems().get(rowin).name.getValue();
        String selectedout = tabOutput.getItems().get(rowout).name.getValue();


        //String precedente=ColInput.getCellData(2);
        //String consecuente=ColOutput.getCellData(2);

        //rule.replace("entry",precedente );
        //rule.replace("out",consecuente);

        textAreaRules.appendText("If "+selectedin+" THEN "+selectedout);

        textAreaRules.appendText(System.lineSeparator());

        writeRules(rowin,rowout);

    }

    private void writeRules(Integer selectedin, Integer selectedout) {
        String regla = " ";

         regla=selectedin+", "+selectedout;
         regla+=" (1) : 1";
         Str+=regla+"\n";
         fisFile_C.reglasFis.add(Str);
         fisFile_C.setReglas(Str);

    }


    @Override
    public void initialize(URL location, ResourceBundle resources) {

        //make sure the property value factory should be exactly same as the e.g getSId
```

```
            from your model class
        ColInput.setCellValueFactory(new PropertyValueFactory<>("name"));
        ColOutput.setCellValueFactory(new PropertyValueFactory<>("name"));

        //add your data to the table here.
        tabInput.setItems(listTable);
        tabOutput.setItems(listTableOut);
    }


    public void Receive(FisNew stagefisNew, fisFile fis) {
        FisNew_inRules=stagefisNew;
        fisFile_C=fis;

        fisVar vartmp;


        HashMap <Double, VAR> entrada  = fisFile_C.INPUT;
        HashMap <Double, VAR> salida   = fisFile_C.OUTPUT;

        for (Map.Entry<Double, VAR> entry : entrada.entrySet()) {
            Double key = entry.getKey();
            VAR value = entry.getValue();

            Iterator it = value.getVariable().entrySet().iterator();
            while (it.hasNext()) {
                Map.Entry e = (Map.Entry)it.next();
                MF tmpMF=(MF)e.getValue();
                vartmp=new fisVar("INPUT", tmpMF.getName());
                listTable.add(vartmp);
                //System.out.println("Este es llave: " + e.getKey() + " Este es valor: "
                    + tmpMF.getName());
            }
            // ...
        }

        for (Map.Entry<Double, VAR> entry : salida.entrySet()) {
            Double key = entry.getKey();
            VAR value = entry.getValue();

            Iterator it = value.getVariable().entrySet().iterator();
            while (it.hasNext()) {
                Map.Entry e = (Map.Entry)it.next();
                MF tmpMF=(MF)e.getValue();
                vartmp=new fisVar("INPUT", tmpMF.getName());
                listTableOut.add(vartmp);
                //System.out.println("Este es llave: " + e.getKey() + " Este es valor: "
                    + tmpMF.getName());
            }
            // ...
        }

    }
}
```

```
package FisGen;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.control.TextArea;
import javafx.scene.layout.AnchorPane;
import javafx.stage.FileChooser;
import javafx.stage.Modality;
import javafx.stage.Stage;
```

```
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.stream.IntStream;
import java.util.List;

public class Controller {

    String fisFile;

    Controller mainStage;

    @FXML
    private Label lblFileName;

    @FXML
    private TextArea txtFIS;

   Stage stage;

    public void initialize(){
        mainStage=this;


    }

    public void closeWindows(ActionEvent event) {
        System.exit(0);


    }

    public void pressOpenMenu(ActionEvent event)throws IOException {

        final FileChooser fileChooser = new FileChooser();
        configureFileChooser(fileChooser);
        File file = fileChooser.showOpenDialog(stage);

        if (file != null) {
            lblFileName.setText("Archivo abierto: " + file.getAbsolutePath());
            lblFileName.setVisible(true);
            //xmlHandler(file);
            xmlHandlerFIS( file);

        }


    }

    private static void configureFileChooser(
            final FileChooser fileChooser) {
        fileChooser.setTitle("View XML Files");
        /*fileChooser.setInitialDirectory(
                new File(System.getProperty("user.home"))
        );*/
        fileChooser.getExtensionFilters().addAll(
                //new FileChooser.ExtensionFilter("All Images", "*.*"),
                //new FileChooser.ExtensionFilter("JPG", "*.jpg"),
                new FileChooser.ExtensionFilter("XML", "*.xml")
        );
```

```
}

///Handler Fis
public void xmlHandlerFIS(File file){

    File fXmlFile;
    /////
    try {
        fXmlFile = file;

        DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
        DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
        Document doc = dBuilder.parse(fXmlFile);
        doc.getDocumentElement().normalize();

        NodeList testBusNodeList = doc.getElementsByTagName("System");

        for (int parameter = 0; parameter < testBusNodeList.getLength(); parameter++)
             {

            Node node = testBusNodeList.item(parameter);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element eElement = (Element) node;

                String name = eElement.getElementsByTagName("Name").item(0).
                    getTextContent();
                String type = eElement.getElementsByTagName("Type").item(0).
                    getTextContent();
                String version = eElement.getElementsByTagName("Version").item(0).
                    getTextContent();
                String numInputs = eElement.getElementsByTagName("NumInputs").item(0)
                    .getTextContent();
                String numOutputs = eElement.getElementsByTagName("NumOutputs").item
                    (0).getTextContent();
                String numRules = eElement.getElementsByTagName("NumRules").item(0).
                    getTextContent();
                String andMth = eElement.getElementsByTagName("AndMethod").item(0).
                    getTextContent();
                String orMth = eElement.getElementsByTagName("OrMethod").item(0).
                    getTextContent();
                String imnMth = eElement.getElementsByTagName("ImpMethod").item(0).
                    getTextContent();
                String aggMth = eElement.getElementsByTagName("AggMethod").item(0).
                    getTextContent();
                String defMth = eElement.getElementsByTagName("DefuzzMethod").item(0)
                    .getTextContent();

                fisFile=new String();
                fisFile="[System]"+ "\n" +
                        "Name="+name + "\n" +
                        "Type ="+type + "\n" +
                        "Version="+version + "\n" +
                        "NumInputs="+numInputs + "\n" +
                        "NumOutputs="+numOutputs + "\n" +
                        "NumRules="+numRules + "\n" +
                        "AndMethod="+andMth + "\n" +
                        "OrMethod="+orMth + "\n" +
                        "ImpMethod="+imnMth + "\n" +
                        "AggMethod="+aggMth + "\n" +
                        "DefuzzMethod="+defMth + "\n" + "\n";


                NodeList inputs = eElement.getElementsByTagName("InputField");


                IntStream.range(0, inputs.getLength()).forEach(bName -> {
```

```
                    fisFile=fisFile+inputs.item(bName).getChildNodes().item(1).
                        getTextContent()+"\n"+
                            inputs.item(bName).getChildNodes().item(3).getNodeName()+
                                "="+inputs.item(bName).getChildNodes().item(3).
                                getTextContent()+"\n"+
                            inputs.item(bName).getChildNodes().item(5).getNodeName()+
                                "="+inputs.item(bName).getChildNodes().item(5).
                                getTextContent()+"\n"+
                            inputs.item(bName).getChildNodes().item(7).getNodeName()+
                                "="+inputs.item(bName).getChildNodes().item(7).
                                getTextContent()+"\n"+
                            inputs.item(bName).getChildNodes().item(9).getNodeName()+
                                "="+inputs.item(bName).getChildNodes().item(9).
                                getTextContent()+"\n"+
                            inputs.item(bName).getChildNodes().item(11).getNodeName()
                                +"="+inputs.item(bName).getChildNodes().item(11).
                                getTextContent()+"\n"+
                            inputs.item(bName).getChildNodes().item(13).getNodeName()
                                 +"="+  inputs.item(bName).getChildNodes().item(13).
                                getTextContent()+ "\n"+
                            "\n";
                });

                NodeList outputs = eElement.getElementsByTagName("OutputField");


                IntStream.range(0, outputs.getLength()).forEach(bName -> {

                    fisFile=fisFile+outputs.item(bName).getChildNodes().item(1).
                        getTextContent()+"\n"+
                            outputs.item(bName).getChildNodes().item(3).getNodeName()
                                +"="+outputs.item(bName).getChildNodes().item(3).
                                getTextContent()+"\n"+
                            outputs.item(bName).getChildNodes().item(5).getNodeName()
                                +"="+outputs.item(bName).getChildNodes().item(5).
                                getTextContent()+"\n"+
                            outputs.item(bName).getChildNodes().item(7).getNodeName()
                                +"="+outputs.item(bName).getChildNodes().item(7).
                                getTextContent()+"\n"+
                            outputs.item(bName).getChildNodes().item(9).getNodeName()
                                +"="+outputs.item(bName).getChildNodes().item(9).
                                getTextContent()+"\n"+
                            outputs.item(bName).getChildNodes().item(11).getNodeName
                                ()+"="+outputs.item(bName).getChildNodes().item(11).
                                getTextContent()+"\n"+
                            outputs.item(bName).getChildNodes().item(13).getNodeName
                                () +"="+outputs.item(bName).getChildNodes().item(13).
                                getTextContent()+ "\n"+
                            "\n";

                });

                fisFile=fisFile+"[Rules]"+"\n";
                NodeList rules = eElement.getElementsByTagName("ruleval");

                IntStream.range(0, rules.getLength()).forEach(bName -> {
                    fisFile=fisFile+rules.item(bName).getTextContent()+"\n";
                });

            }
        }
    } catch (Exception e) {
        System.out.println("!!!!!!!!  Exception while reading xml file :" + e.
            getMessage());
    }

    txtFIS.setText(fisFile);
    txtFIS.setVisible(Boolean.TRUE);
```

```
    }
    ///End Handler Fis

    public void pressNewMenu(ActionEvent event)throws IOException {

        Stage loadWin=new Stage();
        FXMLLoader loader= new FXMLLoader();
        AnchorPane rooPane=(AnchorPane)loader.load(getClass().getResource("FisNew.fxml").
            openStream());
        FisNew loadDataView=(FisNew) loader.getController();
        Scene scene=new Scene(rooPane);
        loadWin.setScene(scene);
        loadWin.alwaysOnTopProperty();
        loadWin.initModality(Modality.APPLICATION_MODAL);
        loadWin.show();
    }

    public void pressViewMenu(ActionEvent event)throws IOException {

        Stage loadWin=new Stage();
        FXMLLoader loader= new FXMLLoader();
        AnchorPane rooPane=(AnchorPane)loader.load(getClass().getResource("FisVIEW.fxml")
            .openStream());
        FisViewController loadDataView=(FisViewController) loader.getController();
        Scene scene=new Scene(rooPane);
        loadWin.setScene(scene);
        loadWin.alwaysOnTopProperty();
        loadWin.initModality(Modality.APPLICATION_MODAL);
        loadWin.show();
    }


}
```

Apéndice B

Módulo II *Script* de *MatLab*®

A continuación se presenta el *scripts* de *MatLab*® desarrollado para realizar el control en lazo cerrado de los modelos seleccionados con los CLDs generados.

```
clc
path = pwd;
ext  = '.fis';
ar =ls(path);
cf=0;
x = [];
data=[];
mp=[];
figure();
hold on,

axis([0 5 0 2]);
for i=1:size(ar,1)

     cn=ar(i,:);
     [~,~,ex]=fileparts(cn);
     if (and(~isfolder(fullfile(path,cn)),or(strcmpi(strtrim(ex),ext),isempty(ext))))
         cf=cf+1;
         file=strtrim(cn);
         difuso = readfis(file);
         sim('controlador2018a.slx');

         x = [x xxx];


      end
end
 for j=1:25
     data=[data x(j).Data];
     mp=[mp x(j).Data];

 end
hold off
%plot(x)
[X,Y] = meshgrid(0:0.01:1.5,1:13);
surf(X,Y,data')
xlabel('Tiempo[sec]','FontSize',15)
ylabel('Sistemas','FontSize',15)
zlabel('Posicin','FontSize',15)
```