UCLV
Universidad Central
"Marta Abreu" de Las Villas

FIE
Facultad de
Ingeniería Eléctrica

Departamento de Automática y Sistemas Computacionales

TRABAJO DE DIPLOMA

Título Diseño de software para el modelado de sistemas discretos mediante Mapas Cognitivos Difusos

Autor Alejandro Lázaro Alfonso Yero

Tutores Dr. Iván Santana Ching, Msc. Richar Sosa López,
Ing. Alexandra Madruga Peláez

Santa Clara, junio 2018

PENSAMIENTO

"No debemos afirmar lo que no podemos probar. La intuición es un auxilio, muchas veces poderoso, pero no es una vía científica e indudable para llegar al conocimiento."

José Martí

DEDICATORIA

A mis padres, mi hermana y mi sobrino preferido.

A todos los que me han enseñado algo, en especial al chino.

Al flako.

AGRADECIMIENTOS

En primer lugar, a mis padres por educarme de la forma en que lo hicieron. No se me ocurre una mejor. También a ellos por jugárselo todo por mí, espero nunca defraudarlos.

A mi hermana por ser una segunda madre y porque nadie en el mundo me ha apoyado tanto y ha confiado tanto en mí.

A mis abuelos, los presentes y el ausente. Los cuatro están siempre a mi lado y en mi pensamiento.

A mis amigos, los verdaderos. Castellón, Mario, Eduardo, Onay, Flako, Guille, Abelito, Campbell, Maestro. Nada hubiera sido lo mismo sin ustedes.

A Lucy. Te debo más de lo que te imaginas, a pesar del poco tiempo que hemos compartido. A tu amiga, y ya mía también, Patricia.

A los buenos maestros que me formaron y me brindaron su sabiduría. Elena, Felina, Mora, Pino, Yeiniel. En especial a ti Chino, donde sea que estés, tu inteligencia es mi paradigma.

A mis tutores por el tiempo dedicado y los buenos consejos. Richar, Ching, Alexandra, gracias.

En general a todas las personas que me han ayudado alguna vez y a las que han hecho todo lo contrario, a esas también les debo lo que soy hoy.

RESUMEN

El primer paso para monitorizar y controlar un proceso es conocerlo. Los Mapas Cognitivos Difusos tienen grandes potencialidades en la obtención de modelos precisos e interpretables. Este proyecto consiste en el diseño e implementación de una biblioteca de *software* como interfaz de modelado, basada en la teoría de Mapas Cognitivos Difusos. La herramienta desarrollada es capaz de predecir los comportamientos del sistema mediante un proceso de inferencia. Además, puede modificar los modelos mediante aprendizaje automático usando datos históricos del objeto de estudio. Como parte de la validación de los resultados, el programa fue probado simulando un circuito oscilante RLC de tercer orden. Esto permitió comparar el comportamiento del modelo con el del sistema real, previamente conocido. Los experimentos arrojaron resultados satisfactorios en cuanto a las funcionalidades de inferencia y aprendizaje. Finalmente, se propone una aplicación práctica del *software* en el estudio del desarrollo de las plantaciones en una casa de cultivo.

ABSTRACT

The first step to monitor and control a process is to know it. Fuzzy Cognitive Maps have great potential obtaining accurate and interpretable models. This project consists of a design and implementation of a software library as a modeling interface, based on the Fuzzy Cognitive Maps theory. The developed tool is able to predict the behavior of the system through an inference process. In addition, it can modify the models by machine learning using historical data of the system. The program was tested simulating a third order RLC oscillating circuit. This allowed to compare the behavior of the model with that of the real system, previously known. The experiments yielded satisfactory results in terms of the inference and learning functionalities. Finally, a practical application of the software is proposed in the study of crop development in a greenhouse.

TABLA DE CONTENIDOS

INTRODUCCIÓN

Desde su nacimiento en el pasado siglo, el campo de la Inteligencia Artificial (AI[1]) se ha ganado su lugar en la vanguardia del proceso de desarrollo de la ciencia y la tecnología. Los requerimientos computacionales, cada día mejor cubiertos, y la capacidad de representar los sistemas más complejos, tanto por su alto número de variables como por el tipo de relaciones entre estas, han convertido la AI en una herramienta altamente recomendable y eficaz.

Muchas son, y cada día más frecuentes, las investigaciones que se encuentran, tanto en el ámbito puramente científico como en el tecnológico y productivo, sobre los distintos métodos de AI. Dentro de estos, juegan un papel protagónico los modelos conexionistas, por ser los más fieles a la estructura y funcionamiento conocidos del cerebro.

Las Redes Neuronales Artificiales (ANN[2]) constituyen, tal y como su nombre lo indica, una representación de la estructura y funcionamiento de la red cerebral de los animales. Estas han sido usadas con éxito en problemas de clasificación de patrones, predicción de comportamientos y aproximación de sistemas.

Los Mapas Cognitivos Difusos (FCM[3]) constituyen una de las técnicas de AI más extendida y explorada en los últimos años. La necesidad de realizar modelos causales cercanos a la realidad, unido a los requerimientos de precisión e interpretabilidad han llevado a un incremento en los esfuerzos por desarrollar este tipo de representación matemática.

Por tanto, los FCM no se encargan de imitar al cerebro, sino que persiguen el objetivo de representar directamente a los sistemas dinámicos, teniendo en cuenta las variables que los componen y la relación de causalidad que existe entre estas a lo largo del tiempo. Por su

[1] AI: del inglés *Artificial Intelligence.*
[2] ANN: del inglés *Artificial Neural Network.*
[3] FCM: del inglés *Fuzzy Cognitive Maps.*

estructura de grafo direccionado, es que son clasificados también como modelos conexionistas. También se pueden encontrar en los ámbitos teórico y práctico varias aplicaciones de este tipo de modelo.

Sin embargo, la teoría actual de FCM es rígida en cuanto a una serie de condiciones y consideraciones que son tomadas a la hora de su desarrollo matemático, las cuales, al ser implementadas en la práctica, ofrecen un conjunto de dificultades. Por otro lado, no se cuenta en la actualidad con bibliotecas de *software* que encapsulen la teoría de FCM de modo que pueda ser aplicada con facilidad en el desarrollo de aplicaciones prácticas. Incluso las principales plataformas para el trabajo con minería de datos y Aprendizaje Automático (ML[4]) como WEKA (Holmes et al., 1994) y Orange (Demšar et al., 2013), no poseen módulos para el trabajo con esta modalidad de AI.

Además, los campos de aplicación que han utilizado hasta ahora FCM como alternativas de modelado son muy complejos. Sistemas sociopolíticos, medioambientales, económicos, etc. En este tipo de objetos de estudio, la validación final de los resultados se realiza mediante el criterio de personas calificadas que no pueden hacer más que decir si la respuesta del modelo es coherente o no, sin poder brindar una certeza total, debido a las indeterminaciones de naturaleza estocástica a las que están sujetos estos sistemas. Es razonable entonces, comprobar primero una herramienta de este tipo en ambientes deterministas y exactamente predecibles, antes de enfrentarla a otros más complicados y desconocidos.

Producto del proyecto de colaboración entre la Empresa de Automatización Integral (CEDAI) y el Grupo de Internet de las Cosas, Automatización e Inteligencia Artificial se ha está trabando en el desarrollo de estrategias de modelado de sistemas complejos. La presente investigación es una contribución a este fin y para su desarrollo han sido planteados los siguientes objetivos:

Objetivo General

Diseñar *software* para el modelado de sistemas discretos mediante Mapas Cognitivos Difusos.

Objetivos Específicos

[4] ML: del inglés *Machine Learning.*

- Demostrar la similitud entre Mapas Cognitivos Difusos y sistemas dinámicos en tiempo discreto.
- Diseñar biblioteca de *software* para el trabajo con Mapas Cognitivos Difusos.
- Valorar la biblioteca de *software* mediante su aplicación en dos casos prácticos.

Organización del informe

El informe está estructurado en tres capítulos que se relacionan a continuación:

Capítulo I. Inteligencia Artificial y Mapas Cognitivos Difusos: En este se aborda todo lo necesario en cuanto al basamento teórico y matemático de las técnicas de Inteligencia Artificial (específicamente los Mapas Cognitivos Difusos), así como los métodos de aprendizaje automático. Se explica mediante ejemplos el campo de aplicación de estas herramientas. Además, se justifica la selección de la plataforma de desarrollo utilizada.

Capítulo II. Diseño de software para Mapas Cognitivos Difusos: Aquí se exponen los elementos fundamentales de la implementación del *software* tales como diagramas UML de clases y de casos de usos. Se ofrece una metodología de utilización para la herramienta creada.

Capítulo III. Resultados obtenidos y propuesta de aplicación: En este último capítulo se exponen los resultados obtenidos en los experimentos de prueba del *software*, mediante datos y gráficas. Se hace una propuesta de aplicación real de la biblioteca ajustada a la metodología mostrada en el capítulo anterior.

Al final del informe se muestran las principales conclusiones a las que se arribó y las recomendaciones para futuras investigaciones que sean continuación de esta.

CAPÍTULO 1. INTELIGENCIA ARTIFICIAL Y MAPAS COGNITIVOS DIFUSOS

El presente proyecto se propone el diseño y desarrollo de una biblioteca de *software* para la implementación de una técnica de AI, por lo que se hace necesario ofrecer primeramente un trasfondo breve, pero lo más detallado posible sobre el tema. En este primer capítulo se exponen las ventajas del uso de la AI, y específicamente de los FCM, mediante la exposición de su basamento teórico y la muestra de aplicaciones prácticas en las que han sido introducidos. Se explica, además, la plataforma de desarrollo utilizada y las principales razones de su selección.

Aunque el campo de aplicación de la AI es extremadamente amplio y abarcador, este apartado se limita, con toda intención, a su uso como herramienta de modelado. Conocer las dinámicas que rigen el comportamiento de cualquier sistema complejo es siempre el primer paso para tomar cualquier acción de control sobre el mismo.

1.1 Inteligencia Artificial como herramienta de modelado

El cerebro humano es, por mucho, el sistema más complejo dentro del universo conocido (Redolar Ripoll, 2014). Su capacidad de procesamiento y eficiencia energética son impresionantes. Con el acelerado desarrollo de la tecnología a partir del siglo pasado, surgió la tendencia a crear programas que imitaran estas capacidades. Así surge el campo de la AI.

A menudo se presentan problemas y situaciones que no pueden ser modelados o resueltos por métodos convencionales de programación secuencial. Algunos ejemplos son el reconocimiento de patrones, diagnóstico de enfermedades, predicción de valores basados en datos estadísticos y simulación de sistemas complejos (Díez, 2001). La indeterminación, la

complejidad de las relaciones inter-variables, la no linealidad, son, entre otras, características que sugieren el uso de AI a la hora de modelar o resolver una determinada situación.

Abundan las investigaciones en las que se han usado técnicas de AI para representar sistemas complejos. En (Lek et al., 1996), se realiza una comparación entre los métodos de regresión lineal y las ANN a la hora de modelar sistemas con relaciones no lineales entre las variables. Para esto se basan en el tratamiento de un set de datos del proceso ecológico relativo a la densidad de los sitios de desove de la trucha marrón. La investigación demuestra la superioridad de las redes neuronales en sistemas con estas características complejas.

En el artículo (Chojaczyk et al., 2015) se presenta una encuesta sobre el desarrollo y el uso de los modelos de redes neuronales en el análisis de fiabilidad estructural. La encuesta identifica los diferentes tipos de redes, los métodos de evaluación de la fiabilidad estructural que se utilizan habitualmente, las técnicas propuestas para la mejora del conjunto de formación de los modelos neuronales y también algunas aplicaciones de las aproximaciones de estos para el diseño estructural y los problemas de optimización.

Por otro lado, en (Gutiérrez et al., 2016) se presenta un sistema experto programado en PROLOG capaz de diagnosticar enfermedades en equinos con síntomas de cólicos. Para ello se utilizó una base de datos con información histórica sobre el tema. El sistema usa una serie de predicados basados en reglas que, dado un cuadro sintomático, es capaz de inferir un diagnóstico de la enfermedad de estos animales.

La adopción de métodos probabilísticos de decisión y de aprendizaje estadístico, han llevado a un alto grado de integración entre IA, ML, estadística, teoría de control, neurociencia y otros campos (Russell et al., 2015). El establecimiento de marcos teóricos compartidos, la disponibilidad de datos y la creciente potencia de procesamiento, han producido éxitos notables en diversas tareas tales como: reconocimiento de voz, clasificación de imágenes, vehículos autónomos, traducción automática, locomoción mediante patas y sistemas de preguntas y respuestas.

Uno de los campos de aplicación de la AI de mayor importancia ha sido la medicina. Entre los primeros éxitos de los sistemas expertos se encuentra el sistema MYCIN, un programa interactivo que utiliza reglas de decisión adquiridas para asesorar a los médicos sobre la selección de terapias en enfermedades infecciosas. El mismo se compone de tres módulos

interrelacionados: un sistema de consultas, uno explicativo y otro de adquisición de nuevas reglas (Shortliffe, 1974). En (Zhang, 2016) se revisa el progreso reciente de la AI en el cuidado de la salud y la medicina. En particular, se enfocan en ML y tecnologías de Aprendizaje Profundo (DL[5]) aplicadas a la investigación y los servicios de atención médica.

1.2 Modelos conexionistas de Inteligencia Artificial

Dentro de las principales técnicas de AI que se han desarrollado hasta la actualidad, se encuentran los modelos conexionistas (Granados, 2017), los cuales imitan de un modo bastante básico la estructura biológica del cerebro animal. En este contexto, las ANN poseen el papel protagónico, derivándose en una serie de variantes que se distinguen por su estructura y funcionamiento. Dentro de los distintos tipos de redes existentes se encuentran los FCM, que, aunque son una representación directa de los sistemas, poseen una estructura neuro-sináptica, similar a la del cerebro.

El cerebro humano se compone, básicamente, de un número colosal de neuronas interconectadas entre sí (Braun et al., 2015). Una neurona constituye la unidad fundamental de procesamiento, en la cual la información, que llega en forma de señales electro-químicas desde las neuronas precedentes, es transmitida hacia las siguientes en la cadena sináptica, luego de pasar por un proceso de excitación. En esta estructura son las conexiones sinápticas las que almacenan el conocimiento. Gracias a su plasticidad, esta red interconectada puede variar tanto la estructura como la fuerza con que interactúan entre si las distintas neuronas, permitiendo variar la respuesta ante determinada situación (Ballesteros et al., 2015).

Las variantes informáticas cuentan con un número mucho menor de entidades de procesamiento, aunque se debe destacar que ganan un poco en cuanto a la velocidad de propagación de señales. Las principales ventajas de este tipo de estructuras radican en su plasticidad y capacidad de generalización, así como la tolerancia a fallos provocados por daños aislados en las unidades de procesamiento fundamentales, esto gracias a la descentralización de la información(Babbar et al., 2018).

[5] DL: del inglés *Deep Learning.*

1.3 Redes Neuronales Artificiales

Para mayor entendimiento se muestra a continuación la estructura de la neurona biológica y su análoga representación computacional (Izaurieta and Saavedra, 2000).

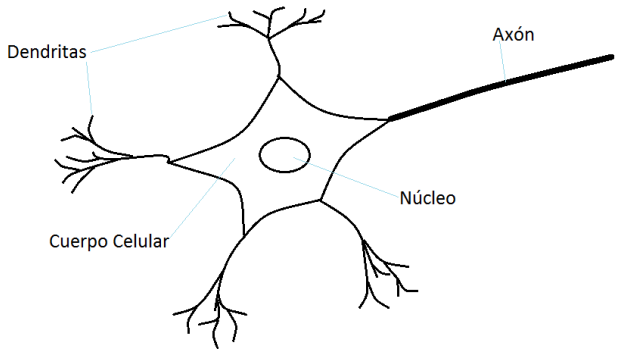

Figura 1.1. Neurona biológica.

En la Figura 1.1 se pueden observar las dendritas, que se encargan de recibir las señales de excitación de las neuronas precedentes, cuya fuerza de llegada depende del poder de las conexiones existentes. Dichas señales se van acumulando y, si logran alcanzar cierto valor umbral, la neurona se excita y propaga esta señal de excitación hacia sus vecinas mediante el axón, de lo contrario la señal propagada es de inhibición (Da Silva et al., 2017).

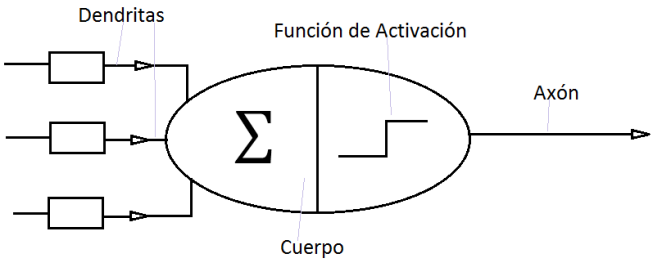

Figura 1.2. Perceptrón Simple.

Por otra parte, el modelo computacional análogo, mostrado en la Figura 1.2, también conocido como Perceptrón Simple (SLP[6]), tiene un comportamiento similar (Kapoor et al., 2016). Cada señal procedente de las neuronas precedentes es multiplicada por el peso

[6] SLP: del inglés *Single Layer Perceptron.*

sináptico correspondiente a cada conexión. Luego, en el cuerpo de la neurona, estos valores se suman y el resultado se evalúa en una función de activación que excita o inhibe la neurona, propagando esta señal como salida. Esto se puede observar más claramente en la ecuación (1.1):

$$S_i = f\left(\sum_j w_{ji} * S_j\right) \tag{1.1}$$

Donde S_i y S_j son las salidas de la i-ésima y j-ésima neuronas de la red, w_{ji} representa la fuerza o peso de la conexión sináptica entre la j-ésima y la i-ésima neuronas y $f(\mathrm{x})$ la función de activación.

Dentro de las funciones de activación usadas, la más frecuente es la función sigmoidea (1.2):

$$f_{(x)} = \frac{1}{1 + e^{-x}} \tag{1.2}$$

Dicha función posee dos asíntotas horizontales en 0 y 1, como se muestra en la Figura 1.3, lo cual la hace ideal para representar la acotación de los valores de salida que no deben exceder ciertos límites.

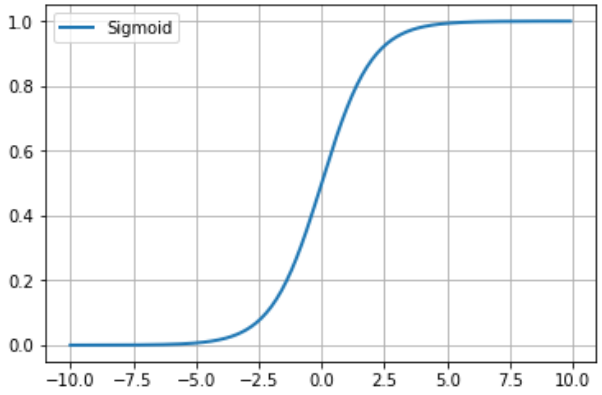

Figura 1.3. Función sigmoidea.

Otra ventaja de esta función es que su derivada (1.3) puede ser expresada en función de sí misma (1.4):

$$\dot{f}_{(x)} = \frac{e^{-x}}{(1+e^{-x})^2} \tag{1.3}$$

$$\dot{f}_{(x)} = f_{(x)}\left(1 - f_{(x)}\right) \tag{1.4}$$

Esto puede resultar conveniente en algunos algoritmos de aprendizaje que utilizan el gradiente de una función de error, basándose en las derivadas parciales respecto a los pesos sinápticos.

Sin embargo, una red neuronal generalmente no se compone de una sola neurona, sino de un conjunto de estas. Son varias las composiciones topológicas posibles, dentro de las cuales, el Perceptrón Multicapa (MLP[7]) es una de las más extendidas (Figura 1.4).

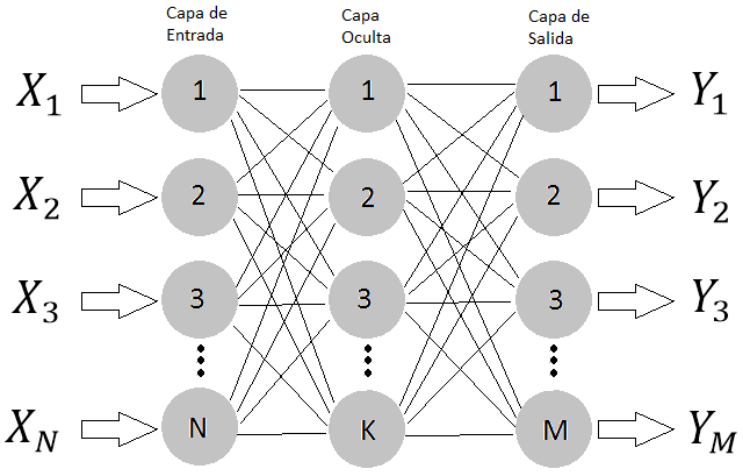

Figura 1.4. Perceptrón Multicapa.

Se trata de una estructura dividida en capas, de tal modo que las neuronas pertenecientes a una de estas solo reciben las señales de las pertenecientes a la anterior y solo envían sus salidas a las que se encuentran en la siguiente (Gardner and Dorling, 1998).

Visto así, se pueden distinguir tres clases de capas. Una capa de entrada, en la cual cada neurona recibe una de las señales provenientes del exterior. Una capa de salida, cuyas

[7] MLP: del inglés *Multi Layer Perceptron.*

neuronas ofrecen la respuesta total de la red a un estímulo dado. Varias capas intermedias u ocultas que intervienen entre las dos anteriores.

En el Anexo I ofrece la demostración de la necesidad de las capas ocultas en la medida que se complejiza el comportamineto a modelar. Para ello se usan como objeto de estudio las funciones lógicas AND y XOR, la primera con un espacio de salida linealmente separable, a diferencia de la segunda.

1.4 Mapas Cognitivos Difusos

Los FCM, introducidos por Bart Kosko en los años 80 (Kosko, 1986), están basados en la idea de Robert Axelrod sobre Mapas Cognitivos (Axelrod, 1976). Representan una herramienta para el modelado y predicción de sistemas complejos, muy poderosa por ser altamente multi-variable e interpretable.

La idea principal de los FCM consiste en modular el objeto de estudio, separándolo en unidades elementales, de modo que se pueda describir la dinámica de interacción interna entre dichas unidades. A cada elemento se le asigna un valor de activación, representando el valor de la variable en cuestión. Si se divide el paso del tiempo en instantes separados por un valor constante, se puede calcular el valor de activación de cada uno de estos conceptos en un instante de tiempo como una dependencia lineal de estos valores de activación en el instante anterior. Esto significa que, conocidas las condiciones iniciales del sistema, y los valores de las entradas, se puede predecir su comportamiento a lo largo del tiempo.

Un mapa cognitivo puede representarse básicamente con cuatro elementos (Nápoles, 2013):

- Un vector $\boldsymbol{C} = \{C_1\ C_2\ C_3\ ...\ C_N\}$ que representa el conjunto de conceptos del sistema.
- Una matriz cuadrada **W** de tamaño NxN, siendo N el orden del sistema, que almacena los valores de causalidad de un concepto sobre otro, donde cada elemento w_{ij} representa la influencia que tiene la presencia del concepto C_i en el sistema sobre el hecho de que se presente el concepto C_j. Cada valor de esta matriz se encuentra dentro del rango [-1; 1].
- Un vector $\boldsymbol{A} = \{A_1\ A_2\ A_3\ ...\ A_N\}$ que contiene los valores de activación de los conceptos. Los valores A_i de cada concepto pertenecen al conjunto [0, 1].

- Una función f, que mantiene acotada la activación de cada concepto dentro del rango [0, 1].

Una vez establecidos estos cuatro elementos, el mapa es capaz de simular en el tiempo el sistema dado. Para esto se inicializa el vector $\boldsymbol{A}$ con los valores iniciales de activación de los conceptos. Luego este vector se actualiza iterativamente según la ecuación (1.5):

$$A_j^{t+1} = f\left(\sum_{i=1}^{N} w_{ij} A_i^t\right), \qquad i \neq j \tag{1.5}$$

De acuerdo a su convergencia un FCM puede clasificarse en (Nápoles Ruiz et al., 2017):

- Estable: si $\exists t_k, k \in \mathbb{N}$: $A_i^{t+1} = A_i^t \, , \forall t > t_k$
- Cíclico: si $\exists t_k, \exists P, \ k \in \mathbb{N}$: $A_i^{t+P} = A_i^t \, , \forall t > t_k$
- Caótico: si no se cumple ninguna de las condiciones anteriores

1.4.1 Representación de sistemas complejos mediante Mapas Cognitivos Difusos

Si se quiere describir un sistema dinámico determinista, cuyas ecuaciones físicas son conocidas a totalidad, existe toda una teoría sólida de representación de los mismos mediante ecuaciones diferenciales. Las ecuaciones diferenciales elementales de un sistema establecen la dependencia temporal de los valores de las variables. De estas ecuaciones se puede derivar un modelo en espacio de estados, ya sea en tiempo continuo o discreto.

Sin embargo, cuando el objeto de estudio posee determinadas complejidades, sobre todo de tipo estocásticas, no se suele contar con estas ecuaciones. Los FCM pueden solucionar esta situación ya que no necesitan tal exactitud en el conocimiento del objeto para describir su comportamiento, y ahí radica su principal fortaleza (Osoba and Kosko, 2017). Supóngase la siguiente situación de ejemplo que permite aclarar la estructura de trabajo de los FCM:

Se quiere analizar el comportamiento de los resultados de un grupo de estudiantes universitarios y se manejan los siguientes conceptos:

- Resultados de los exámenes: **RE**
- Exigencia de los profesores: **EP**
- Dedicación al estudio: **DE**
- Motivación de los profesores: **MP**

- Calidad de las clases impartidas: **CC**

Una posible configuración del mapa cognitivo sería la mostrada en la Figura 1.5:

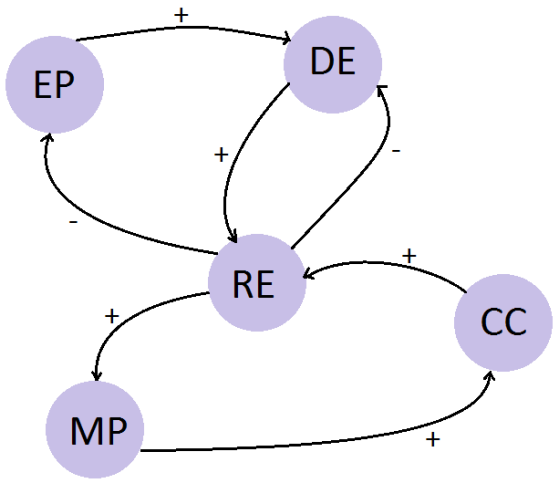

Figura 1.5. Mapa Cognitivo.

En el ejemplo se observa claramente la principal característica que distingue este tipo de modelo dentro de las ANN. Aquí cada neurona representa un concepto concreto y las sinapsis no son más que las relaciones de causalidad entre los mismos. Un signo positivo significa que al cambiar de valor de activación del concepto "causa" en una dirección determinada, esto provoca un cambio en el valor de activación del concepto "efecto" en el mismo sentido. Lo contrario ocurre cuando la relación de causalidad tiene un signo negativo.

En el ejemplo se puede pensar, con toda razón, que existen otros muchos factores influyentes y que no se tomaron en cuenta. Mientras más elementos se utilicen más exacto será el modelo, y tanto más complicado también. Es por eso que la correcta elección de los conceptos que compondrán el mapa es muy importante para su utilidad.

El correcto comportamiento del mapa depende, en primera instancia, de su estructura y de los valores de la matriz de pesos, los cuales a su vez son el resultado de dos factores. El primero lo constituye la opinión de los expertos, que brindan cierta cantidad de información básica del modelo. Y el segundo, la capacidad de adquirir experiencia y modificar su propio comportamiento a partir de esta. Según los datos estadísticos existentes y los que se van obteniendo en el momento, el mapa tiene que ser capaz de auto-ajustarse para asemejarse

cada vez más a la realidad. La confluencia de estos dos factores le otorga interpretabilidad al modelo, el cual puede ser considerado como una caja gris (Ahvenlampi et al., 2004), a diferencia de los MLP que constituyen modelos de caja negra.

1.5 Algoritmos de aprendizaje para Inteligencia Artificial

Una de las principales características de la inteligencia animal, sobre todo la humana, es la capacidad de mejorar el comportamiento a través de la experiencia. Es lógico entonces que uno de los aspectos fundamentales que distinguen a la AI es la posibilidad de aprender. El aprendizaje de un agente basado en AI no es más que el proceso de reestructuración interna que permite, según el análisis de los estados y resultados anteriores conocidos, "mejorar" los estados y resultados futuros. Un aspecto fundamental en este proceso es la evaluación de que es "mejor" y que es "peor", lo cual va a depender de los objetivos perseguidos.

Debido a que en los modelos conexionistas el conocimiento se almacena en los pesos de las conexiones sinápticas, el proceso de aprendizaje consiste precisamente en el ajuste de esos pesos hasta que la red se comporte del modo deseado (Palacios, 2012).

Existen varias maneras de realizar esta tarea, y se pueden clasificar en dos grupos:

- Aprendizaje no supervisado
- Aprendizaje supervisado

Los métodos no supervisados se basan en la idea de que el agente adquiere el conocimiento constantemente durante su interacción con el medio ambiente. La vía más usada ha sido la regla de Hebb (Benítez et al., 2014), la cual tiene una base netamente biológica. En el año 1949, el neurofisiólogo Donald Hebb descubrió que, si dos neuronas se encontraban excitadas simultáneamente, la fuerza de la conexión sináptica entre estas aumentaba, y en caso contrario, la misma disminuía. De este modo la red aprende paulatinamente de la experiencia cotidiana.

Por otro lado, los métodos supervisados utilizan, básicamente, un set de datos con la información del comportamiento deseado (Tang et al., 2016). De este modo, la red ajusta sus pesos sinápticos en un proceso iterativo, con el fin de acercarse lo más posible al comportamiento que produce esos valores. Este proceso se desarrolla en una fase de entrenamiento previa a la utilización práctica del programa. La gran plasticidad de estos

modelos y el conocimiento distribuido que estos presentan, permite que los mismos sean capaces de identificar y ofrecer salidas suficientemente correctas ante la entrada de valores que no se encontraban dentro del conjunto de entrenamiento. Normalmente se utiliza un porciento de los datos en la etapa de aprendizaje y el resto se destina a la validación del resultado final, comprobando así la capacidad de generalización del conocimiento adquirido.

1.5.1 Método del gradiente descendente como algoritmo de aprendizaje

El método del gradiente descendente (Flores Villarreal, 2005) es uno de los más sencillos y, por tanto, utilizados en aprendizaje de modelos de AI. Se basa en encontrar un juego de parámetros del modelo que optimicen una función objetivo.

En el caso de estudio de esta investigación, los parámetros que componen el modelo son los pesos sinápticos entre conceptos y la función objetivo es el error cuadrático medio entre la salida del modelo y los datos experimentales. Tómese el vector $\boldsymbol{w}$ (1.6) como el contenedor de estos parámetros:

$$\boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_P \end{bmatrix} \tag{1.6}$$

Siendo **P** el número total de parámetros.

Ahora supóngase que se tiene una función $f(\boldsymbol{w})$ de la cual se quiere obtener el valor de $\boldsymbol{w}$ que minimiza su valor. Primero se definirá el gradiente de $f(\boldsymbol{w})$ como el vector de las derivadas parciales de f respecta a cada uno de los parámetros (1.7):

$$\nabla f(\boldsymbol{w}) = \begin{bmatrix} \dfrac{\partial f}{\partial w_1} \\ \dfrac{\partial f}{\partial w_2} \\ \vdots \\ \dfrac{\partial f}{\partial w_P} \end{bmatrix} \tag{1.7}$$

Explicados estos conceptos, se puede definir el algoritmo del gradiente descendente por pasos de la siguiente forma:

- Paso1: Se elige un vector $\boldsymbol{w}_0$ (este valor inicial puede responder a algún criterio en específico o puede ser elegido al azar). Se define $\boldsymbol{k}$ como el número de la iteración del proceso y se hace $\boldsymbol{k} = 0$.
- Paso 2: Se calcula el gradiente para el vector. Si $\nabla f(\boldsymbol{w}_k) \approx 0$ se detiene el proceso y $\boldsymbol{w}_k$ es el vector buscado. De lo contrario, se resta el valor del vector gradiente, multiplicado por un factor de aprendizaje, del vector $\boldsymbol{w}_k$ obteniendo un nuevo valor para este según (1.8):

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha \nabla f(\boldsymbol{w}_k) \tag{1.8}$$

- Paso 3: Se hace $\boldsymbol{k} = \boldsymbol{k} + 1$ y se vuelve al paso 2. Así, iterativamente hasta que se alcance un número máximo de iteraciones, previamente especificado.

Supóngase el caso más sencillo posible donde $\boldsymbol{w}$ posee un solo componente (Figura 1.6) para observar gráficamente como ocurre el proceso.

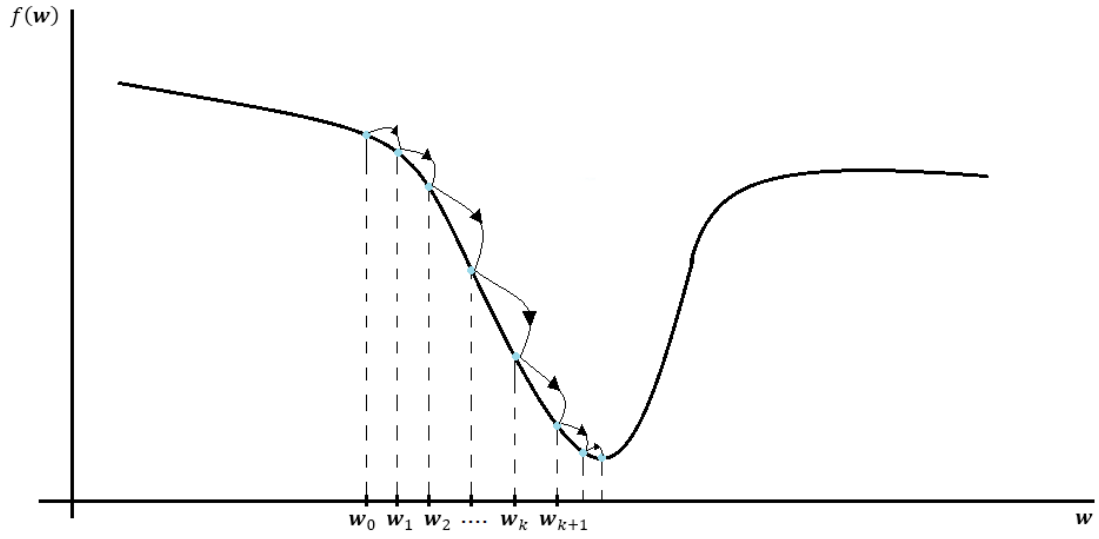

Figura 1.6. Método del gradiente descendente.

Este método solo hace uso del gradiente de la función objetivo, por lo que es considerado un método de primer orden. Existen métodos de orden superior que utilizan, además, la matriz Hessiana (Anexo II).

1.6 Aplicaciones de Redes Neuronales Artificiales y Mapas Cognitivos Difusos

Las ANN son la herramienta de inteligencia artificial que más intenta semejarse a la estructura biológica del cerebro, es por esto que suele verse como el modelo conexionista de AI por excelencia.

Su utilización dentro del mundo científico-técnico es muy extendida, pudiéndose citar numerosos ejemplos. En (Pchelintseva et al., 2017) estudian el problema del reconocimiento de objetos observados, según el patrón generado por los datos EEG[8]. Los datos EEG se registran en el momento de mostrar el cubo Necker (Anexo III) al sujeto y caracterizan el estado apropiado de la actividad cerebral utilizando una imagen biestable de este objeto. Para resolver el problema del reconocimiento, se utilizan ANN, específicamente un MLP como clasificador.

Dentro del campo de las energías renovables, las ANN han demostrado tener un gran potencial. Por ejemplo, en la predicción de la velocidad del viento y, por tanto, de la energía eólica (Yadav and Sahu, 2017), que anteriormente se realizaban mediante análisis estadísticos clásicos. Aquí las ANN han demostrado su versatilidad y superioridad frente a otros métodos más antiguos.

Cuando se modela un sistema físico utilizando ANN solo se tienen en cuenta de forma directa las variables de entrada y de salida del mismo, de modo que la información concerniente a las otras variables, así como las dinámicas e interacciones entre estas, quedan codificadas en la red. De este modo la ANN se convierte en una caja negra, lo cual trae en consecuencia que, una vez sintetizada, no sea posible extraer información útil acerca de las características reales del sistema físico (Trujillano, 2004).

[8] EEG: Electroencefalograma.

En una situación como esta puede ocurrir el hecho de que existan factores intrínsecos en los datos usados en el proceso de aprendizaje y que conduzcan a resultados aparentemente erróneos. En estos casos se hace complicado identificar estos factores.

Una alternativa a este problema es construir una red en la cual cada neurona y cada conexión tengan un significado concreto en el sistema físico que se pretende modelar. Los FCM son este tipo de red, representados por un grafo dirigido bidireccional en el cual cada uno de los nodos, conocidos como conceptos, representa una variable del sistema que se pretende modelar. Por otra parte, las aristas que unen los nodos representan las relaciones de causalidad que existe entre estos, o sea, el grado de influencia que tiene el valor de uno de ellos sobre el otro (Nápoles, 2014).

Son numerosas las investigaciones relacionadas con la teoría de los FCM y los algoritmos de aprendizaje usados en estos. En (Parsopoulos, 2003) se introduce un algoritmo de optimización por enjambre de partículas para la detección de una matriz causal apropiada, con la cual el FCM alcanza un estado estable deseado. En (Papageorgiou, 2004) se introduce un nuevo algoritmo Hebbiano de aprendizaje, el AHL[9], con el objetivo de utilizar el conocimiento de los expertos, en materia de dependencias causales entre conceptos, en el proceso de inferencia de la matriz de pesos. Por otro lado (Stach, 2005) propone un método para el aprendizaje de FCM, basado en algoritmos genéticos, para el ajuste de la matriz de causalidad, apoyándose en la entrada de datos históricos del sistema y sin la intervención humana.

Muchos tipos de problemas ecológicos o ambientales se beneficiarían de modelos basados en el conocimiento de las personas. Para crear modelos ecológicos con conocimiento tanto de los expertos como de las personas locales, en (Özesmi and Özesmi, 2004) se propone un enfoque basado en FCM. En dicha investigación se examinan las percepciones de diferentes partes interesadas en un conflicto ambiental, con el objetivo de facilitar el desarrollo de planes de gestión ambiental participativa y determinar los deseos de reasentamiento de personas desplazadas por un proyecto de presa a gran escala.

En (Hobbs et al., 2002) se presentan los pasos involucrados en la construcción de un FCM de un ecosistema y la interpretación de salida del FCM usando estadísticas multivariables.

[9] AHL: del inglés *Active Hebbian Learning*.

Para ilustrar estas ideas, se crea un modelo de ecosistema complejo, de más de 160 variables, construido para la cuenca hidrográfica del Lago Erie.

Los FCM han sido usados en la agricultura en algunos casos específicos, aunque no es un aspecto muy extendido aún.

En (Papageorgiou, 2010) se investiga el rendimiento y sus variaciones en el complejo proceso de gestión del cultivo del algodón, donde influyen aspectos como el suelo, el propio cultivo y determinados factores climáticos, todo esto usando los FCM para modelar y representar el conocimiento de los expertos.

Luego, en (Papageorgiou, 2011) se utilizan los FCM para predecir el rendimiento del cultivo del algodón. En esta nueva investigación se usa la teoría de los mapas ya no solo para modelar las condiciones del sistema en cuestión, sino, además, para inferir el posible desarrollo del mismo. Aquí se demuestran las potencialidades de esta herramienta a la hora de interpretar datos recopilados en ambientes con dinámicas altamente complejas.

Los FCM obtienen su apellido del modo en que se tratan los datos. La Teoría de Conjuntos Difusos ha constituido a lo largo de la historia un fuerte aliado de la AI, dado que se asemeja en gran medida al modo de tratar, por parte de los seres humanos, los valores numéricos de las variables con las que este interactúa cotidianamente. En la vida diaria, el ser humano tiende a clasificar los valores de las variables cuantitativas en grupos determinados por su rango de valores. De este modo adquieren estas propiedades cierto carácter cualitativo y descriptivo.

1.7 Selección de la plataforma de desarrollo

Un aspecto muy importante a la hora de desarrollar una herramienta de *software*, como lo es la biblioteca implementada durante esta investigación, es la selección de la plataforma de desarrollo. De una correcta elección dependen tanto la rapidez de desarrollo como la eficiencia, portabilidad y reutilización del resultado final.

En este caso particular se utilizó Python como lenguaje de programación, el cual es multiplataforma y multiparadigma (Tulchak and Marchuk, 2016). El mismo cuenta con *frameworks* de gran calibre, los cuales auxilian desde el desarrollo web, hasta el desarrollo de juegos o algoritmos científicos de cálculos avanzados. Es *software* libre y de código

abierto, teniendo en cuenta la licencia PSFL[10], bajo la cual se distribuye, muy parecida a la de GPL[11], pero encontrando la excepción de que se pueden distribuir los binarios del lenguaje sin tener que anexar las fuentes. La versión utilizada fue la 3.6.5 por ser la más actualizada. Esto persigue el objetivo de tener un código lo más versátil posible.

Python se encuentra en multitud de aplicaciones y servicios que usamos habitualmente. Ostenta una gran lista de usuarios de gran importancia como Google, YouTube y Facebook, los cuales utilizan este lenguaje de programación. Poco a poco Python va ganando territorio y, entre los entendidos, se ha convertido en uno de los más solicitados.

La distribución Anaconda es la suite más completa para la ciencia de manejo de datos con Python. Fue elegida como gestor de paquetes para el desarrollo de esta investigación por varias razones. Es una suite de código abierto que abarca una serie de aplicaciones, bibliotecas y conceptos diseñados para el desarrollo de la Ciencia de datos con Python (Doig, 2016). Ofrece una distribución de Python que funciona como gestor de entornos y posee más de 720 paquetes de código abierto**.** Cuenta con una documentación detallada y con una gran comunidad de desarrollo. Integra diversos IDEs de desarrollo tales como Jupyter, JupyterLab, RStudio y Spyder, este último utilizado en este proyecto.

Dentro de las principales ventajas del IDE Spyder que condujeron a su elección están las siguientes: es libre y de código abierto, está integrado dentro de la distribución de Anaconda y posee una terminal de IPython dentro de la ventana principal, permitiendo la depuración y comprobación de los códigos generados. Al manejar Python, el cual es un lenguaje interpretado, los *scripts* pueden ser divididos en células para ser ejecutados por lotes (tal y como lo hace el editor de Matlab), esta característica facilita la depuración y prueba en tiempo de desarrollo. Además, posee las características fundamentales de cualquier buen editor de código tales como auto-completamiento de código, coloración de sintaxis y análisis del código en tiempo real.

[10] PSFL: del inglés *Python Software Foundation License.*
[11] GPL: del inglés *General Public License.*

1.8 Consideraciones del capítulo

A lo largo de este capítulo se ha expuesto la importancia de la AI como técnica de modelado y de interpretación y representación de sistemas. Hasta este punto se puede afirmar que estas técnicas, específicamente los FCM, cuentan con un amplio interés dentro de la comunidad científica, lo que ha permitido su constante evolución, y un amplio campo de aplicación en los más diversos y complejos ámbitos de desarrollo científico y práctico.

Expuestos los principales aspectos teóricos, están creadas las condiciones para cumplir con los objetivos propuestos en este proyecto, pues se cuenta con todos los conocimientos necesarios para comenzar a desarrollar la herramienta de *software* que permitirá utilizar la teoría de los FCM en aplicaciones prácticas de modelado.

CAPÍTULO 2. DISEÑO DE SOFTWARE PARA MAPAS COGNITIVOS DIFUSOS

Para cumplir con el objetivo principal de este proyecto el paso siguiente es la organización del trabajo. Este capítulo se encarga de exponer los aspectos que jugaron papeles importantes en el diseño y la implementación del *software*. A los materiales y métodos utilizados, los esquemas de estructura y funcionamiento y la guía para su correcta utilización están dedicados los siguientes epígrafes.

2.1 Modelado de sistemas dinámicos en tiempo discreto

El modelo de un sistema puede desarrollarse en el espacio de tiempo continuo o discreto. En el caso de los modelos de tiempo discreto los valores de las variables solo cambian en ciertos instantes, separados por intervalos durante el cual permanecen constantes. Existen dos maneras fundamentales de analizar los sistemas de tiempo discretos, estas son el método de la transformada *z* y el de modelo de espacio de estados (Ogata, 2002). En lo concerniente a este trabajo es importante la comprensión del segundo.

El comportamiento de un sistema determinista está descrito por una serie de ecuaciones diferenciales. Estas representan el modo en que cada variable de estado varía en el tiempo en función de los valores que poseen las demás variables de estado y las variables de entrada (externas al sistema). El número de ecuaciones necesarias para que el modelo esté totalmente determinado, coincide con el número de variables de estado que depende del orden del sistema. Supóngase uno general de orden **N** con **M** variables de entrada y llámese **X** al vector de estado (2.1), el cual contiene las variables de estado, y **u** al vector de variables de entrada (2.2):

$$\boldsymbol{X} = (x_1, x_2, x_3, \dots, x_N) \tag{2.1}$$

$$\boldsymbol{u} = (u_1, u_2, u_3, \dots, u_M) \tag{2.2}$$

La forma en que varían en el tiempo cada una de estas variables, está descrita por sus primeras derivadas temporales. Por ello, el modelo está determinado por el vector que contiene las derivadas de cada una de ellas (2.3):

$$\dot{\boldsymbol{X}} = (\dot{x}_1, \dot{x}_2, \dot{x}_{N3}, \dots, \dot{x}_N) \tag{2.3}$$

Según lo dicho anteriormente, el modelo que determina el comportamiento temporal del sistema queda descrito por (2.4), que es una ecuación matricial (2.5):

$$\dot{\boldsymbol{X}} = A\boldsymbol{X} + B\boldsymbol{u} \tag{2.4}$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \vdots \\ \dot{x}_N \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & A_{13} & \dots & A_{1N} \\ A_{21} & A_{22} & A_{23} & \dots & A_{2N} \\ A_{31} & A_{32} & A_{33} & \dots & A_{3N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{N1} & A_{N2} & A_{N3} & \dots & A_{NN} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{bmatrix} + \begin{bmatrix} B_{11} & B_{12} & B_{13} & \dots & B_{1M} \\ B_{21} & B_{22} & B_{23} & \dots & B_{2M} \\ B_{31} & B_{32} & B_{33} & \dots & B_{3M} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ B_{N1} & B_{N2} & B_{N3} & \dots & B_{NM} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_M \end{bmatrix} \tag{2.5}$$

A partir de aquí se puede obtener el modelo en tiempo discreto. Acomodando la ecuación (2.4) y pre-multiplicándola por e^{-AT} , donde T es el tiempo de muestreo se obtiene (2.6):

$$e^{-At}\left[\dot{\boldsymbol{X}}_{(t)} - A\boldsymbol{X}_{(t)}\right] = e^{-At}B\boldsymbol{u}_{(t)} \tag{2.6}$$

Por otra parte, si se calcula la derivada de $e^{-At}\boldsymbol{X}_{(t)}$ (2.7) y (2.8):

$$\frac{d}{dt}\left[e^{-At}\boldsymbol{X}_{(t)}\right] = -Ae^{-At}\boldsymbol{X}_{(t)} + e^{-At}\dot{\boldsymbol{X}}_{(t)} \tag{2.7}$$

$$\frac{d}{dt}\left[e^{-At}\boldsymbol{X}_{(t)}\right] = e^{-At}\left[\dot{\boldsymbol{X}}_{(t)} - A\boldsymbol{X}_{(t)}\right] \tag{2.8}$$

Y luego se combinan (2.6) y (2.8) se obtiene (2.9):

$$\frac{d}{dt}\left[e^{-At}\boldsymbol{X}_{(t)}\right] = e^{-At}B\boldsymbol{u}_{(t)} \tag{2.9}$$

Integrando ambos miembros resulta (2.10):

$$e^{-At}\boldsymbol{X}_{(t)} = \boldsymbol{X}_{(0)} + \int_0^t e^{-A\tau}B\boldsymbol{u}_{(\tau)}d\tau \tag{2.10}$$

Premultiplicando por e^{At} se obtiene la expresión para $\boldsymbol{X}_{(t)}$ (2.11):

$$\boldsymbol{X}_{(t)} = e^{At}\left[\boldsymbol{X}_{(0)} + \int_0^t e^{-A\tau}B\boldsymbol{u}_{(\tau)}d\tau\right] \tag{2.11}$$

Con el objetivo de discretizar el modelo se sustituye $t = kT$ *y* $t = (k+1)T$ en (2.11), obteniendo (2.12) y (2.13):

$$\boldsymbol{X}_{(kT)} = e^{AkT}\left[\boldsymbol{X}_{(0)} + \int_0^{kT} e^{-A\tau}B\boldsymbol{u}_{(\tau)}d\tau\right] \tag{2.12}$$

$$\boldsymbol{X}_{((k+1)T)} = e^{A(k+1)T}\left[\boldsymbol{X}_{(0)} + \int_0^{(k+1)T} e^{-A\tau}B\boldsymbol{u}_{(\tau)}d\tau\right] \tag{2.13}$$

Multiplicando (2.12) por e^{AT} (2.14):

$$e^{AT}\boldsymbol{X}_{(kT)} = e^{A(k+1)T}\left[\boldsymbol{X}_{(0)} + \int_0^{kT} e^{-A\tau}B\boldsymbol{u}_{(\tau)}d\tau\right] \tag{2.14}$$

Y restando de (2.13) se obtiene (2.15):

$$\boldsymbol{X}_{((k+1)T)} - e^{AT}\boldsymbol{X}_{(kT)} = e^{A(k+1)T}\int_{kT}^{(k+1)T} e^{-A\tau}B\boldsymbol{u}_{(\tau)}d\tau \tag{2.15}$$

De donde se puede despejar $\boldsymbol{X}_{((k+1)T)}$ (2.16):

$$\boldsymbol{X}_{((k+1)T)} = e^{AT}\boldsymbol{X}_{(kT)} + e^{A(k+1)T}\int_{kT}^{(k+1)T} e^{-A\tau}B\boldsymbol{u}_{(\tau)}d\tau \tag{2.16}$$

Para un sistema de tiempo discreto el valor de la entrada permanece constante durante todo un período de muestreo (2.17):

$$\boldsymbol{u}_{(t)} = \boldsymbol{u}_{(kT)} \;\; \forall\, kT \leq t < (k+1)T \tag{2.17}$$

Teniendo esto en cuenta en (2.16) se obtiene (2.18):

$$\boldsymbol{X}_{((k+1)T)} = e^{AT}\boldsymbol{X}_{(kT)} + e^{A(k+1)T}\int_{kT}^{(k+1)T} e^{-A\tau}B\boldsymbol{u}_{(kT)}d\tau$$

$$\boldsymbol{X}_{((k+1)T)} = e^{AT}\left[\boldsymbol{X}_{(kT)} + e^{AkT}\int_{kT}^{(k+1)T} e^{-A\tau}B\boldsymbol{u}_{(kT)}d\tau\right]$$

$$\boldsymbol{X}_{((k+1)T)} = e^{AT}\left[\boldsymbol{X}_{(kT)} + \int_{kT}^{(k+1)T} e^{-A(\tau-kT)}B\boldsymbol{u}_{(kT)}d\tau\right] \tag{2.18}$$

Realizando un cambio el cambio de variables descrito en (2.19) y (2.20):

$$t = \tau - kT \tag{2.19}$$

$$dt = d\tau \tag{2.20}$$

Se obtiene (2.21):

$$\boldsymbol{X}_{((k+1)T)} = e^{AT}\left[\boldsymbol{X}_{(kT)} + \int_{0}^{T} e^{-At}B\boldsymbol{u}_{(kT)}dt\right]$$

$$\boldsymbol{X}_{((k+1)T)} = e^{AT}\boldsymbol{X}_{(kT)} + e^{AT}\int_{0}^{T} e^{-At}B\boldsymbol{u}_{(kT)}dt$$

$$\boldsymbol{X}_{((k+1)T)} = e^{AT}\boldsymbol{X}_{(kT)} + \int_{0}^{T} e^{A(T-t)}B\boldsymbol{u}_{(kT)}dt \tag{2.21}$$

Como el vector de entrada **u** permanece constante durante el tiempo **T** este puede salir de la integral junto con la matriz **B,** obteniéndose (2.22):

$$\boldsymbol{X}_{((k+1)T)} = e^{AT}\boldsymbol{X}_{(kT)} + \left[\int_0^T e^{A(T-t)}dt\right]B\boldsymbol{u}_{(kT)} \tag{2.22}$$

Realizando otro cambio de variables, (2.23) y (2.24):

$$\lambda = T - t \tag{2.23}$$

$$d\lambda = -dt \tag{2.24}$$

Se obtiene (2.25):

$$\boldsymbol{X}_{((k+1)T)} = e^{AT}\boldsymbol{X}_{(kT)} + \left[-\int_T^0 e^{A\lambda}d\lambda\right]B\boldsymbol{u}_{(kT)}$$

$$\boldsymbol{X}_{((k+1)T)} = e^{AT}\boldsymbol{X}_{(kT)} + \left[\int_0^T e^{A\lambda}d\lambda\right]B\boldsymbol{u}_{(kT)} \tag{2.25}$$

De esta manera se tiene el modelo en tiempo discreto del sistema de la forma descrita en (2.26), análogo al modelo (2.4), pero en el espacio de tiempo discreto:

$$\boldsymbol{X}_{(k+1)} = G\boldsymbol{X}_{(k)} + H\boldsymbol{u}_{(k)} \tag{2.26}$$

Aquí las dinámicas están contenidas en las matrices G (2.27) y H (2.28):

$$G = e^{AT} \tag{2.27}$$

$$H = \left[\int_0^T e^{A\lambda}d\lambda\right]B \tag{2.28}$$

Para obtener la matriz G se desarrolla en serie de Taylor según (2.29):

$$e^{AT} = \sum_{n=0}^{\infty}\frac{1}{n!}A^nT^n \tag{2.29}$$

Teniendo en cuenta que el tiempo de muestreo es suficientemente pequeño se puede aproximar a los tres primeros términos como se muestra en (2.30):

$$G = I + AT + \frac{1}{2}A^2T^2 \tag{2.30}$$

Por otro lado, la matriz H se desarrolla hasta llegar a (2.31):

$$H = \left[\int_0^T e^{A\lambda} d\lambda\right] B$$

$$H = \left[A^{-1} e^{A\lambda}\Big|_0^T\right] B$$

$$H = [A^{-1}(e^{AT} - e^0)]B \tag{2.31}$$

Usando (2.27) para sustituir e^{AT} se obtiene (2.32):

$$H = [A^{-1}(G - I)]B \tag{2.32}$$

Usando (2.30) para sustituir G se llega a la expresión final de H (2.33):

$$H = \left[A^{-1}\left(AT + \frac{1}{2}A^2T^2\right)\right] B$$

$$H = \left[A^{-1}AT + A^{-1}\frac{1}{2}A^2T^2\right] B$$

$$H = \left[IT + \frac{1}{2}AT^2\right] B \tag{2.33}$$

2.2 Arquitectura del software

En Python, el encapsulamiento del código se basa en los conceptos de módulo y paquete (Bahit, 2012). Un módulo es un archivo con extensión "*.py*", el cual puede contener definiciones de clases. Un paquete es una carpeta que agrupa una serie de módulos con una relación determinada. El *software* creado es un paquete para Python que engloba la teoría de los FCM en cuatro clases fundamentales. A continuación, se muestra el diagrama de clases que contiene los atributos y métodos de cada una de ellas (Figura 2.1).

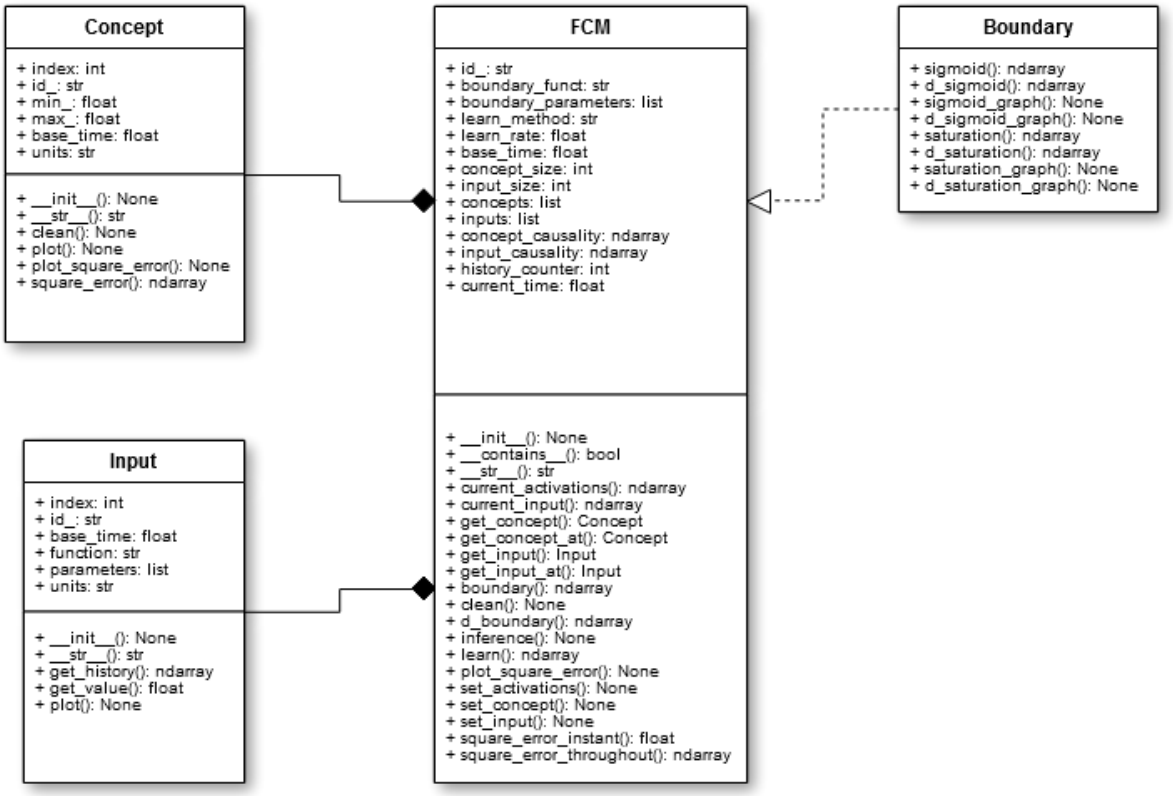

Figura 2.1. Diagrama UML de clases del *software* fcm.

Una instancia de la clase ***fcm*** representa un mapa cognitivo. El mismo posee como atributos una lista de conceptos, instancias de la clase ***Concept***, y una lista de entradas, instancias de la clase ***Input***. La clase ***Boundary*** solo posee métodos estáticos, y por lo tanto no es instanciada en ningún momento. Esta solo se encarga de proporcionar las funciones de acotación y sus derivadas al mapa, durante los procesos de inferencia y de aprendizaje.

El atributo ***index***, de las clases ***Concept*** e ***Input***, es un número natural de orden dentro del mapa contenedor, comenzando por cero. Un mapa con cinco conceptos, por ejemplo, indexa a los mismos en todo momento con la serie 0, 1, 2, 3, 4 en orden de inserción en el mapa.

El atributo ***id_***, al igual que el anterior, constituye un identificador único dentro de cada mapa para las instancias de las clases ***Concept*** e ***Input***. Las instancias de la clase ***fcm*** poseen también un atributo de este tipo. Su objetivo fundamental es dar un nombre explicativo, mediante una cadena de texto, a cada uno de estos elementos dentro del ámbito en que se desarrollan.

El atributo ***base_time***, en las clases ***Concept***, ***Input*** y ***fcm***, representa el período de muestreo del modelo. El mismo es el intervalo de tiempo, medido en segundos, que pasa entre una iteración y otra durante el proceso de inferencia. Los conceptos y entradas pertenecientes a un mapa deben poseer el mismo valor para este atributo que el propio mapa.

Los atributos ***function*** y ***parameters*** de la clase ***Input***, forman en conjunto una función de entrada predeterminada. Por ejemplo, si se necesita que el mapa tenga una entrada sinusoidal de amplitud 10, frecuencia 60Hz y desfasaje inicial 2 rad entonces el atributo ***function*** será una cadena de texto predefinida representando a la función seno y el atributo ***parameters*** será una lista de Python de valor [10, 60, 2]. Esto se aplica también para los atributos ***boundary_func*** y ***boundary_parameters***, los cuales definen la función de acotación usada por el mapa en su proceso de inferencia.

El atributo ***learn_method*** de la clase ***fcm*** establece el algoritmo específico que utiliza el proceso de aprendizaje del mapa. Hasta la versión actual del paquete existe un solo algoritmo programado, el del gradiente descendente.

Los atributos ***concept_size*** e ***input_size*** almacenan en todo momento la cantidad de conceptos y de entradas que posee el mapa respectivamente. Mientras, los atributos ***concepts*** e ***inputs*** son listas que almacenan las instancias de estos conceptos y entradas.

Los atributos ***concept_causality*** e ***input_causality*** son las matrices de causalidad entre los niveles de activación de los conceptos y de las entradas. Estos atributos son las matrices G y H de la ecuación (2.26).

Los atributos ***history_counter*** y ***current_time*** representan el momento actual del mapa en el proceso de inferencia. El primero es la cantidad de iteraciones realizadas y el segundo, el tiempo transcurrido, medido en segundos. El cociente entre el segundo y el primero nos es más que el período de muestreo, o sea, el atributo ***base_time***. Entre estos atributos existe una estrecha relación, tal y como se muestra en la ecuación (2.34)

$$\boldsymbol{base\_time} = \frac{\boldsymbol{current\_time}}{\boldsymbol{history\_counter}} \tag{2.34}$$

2.3 Funcionalidad del software

La principal función del paquete ***fcm*** es crear una instancia de la clase **FCM** para modelar el sistema objeto de estudio, luego de lo cual, mediante los métodos ***set_concept*** y ***set_input***, se introducen los conceptos y las entradas que componen dicho modelo, los cuales son instancias de las clases ***Concept*** e ***Input*** respectivamente.

El paquete proporciona una buena cantidad de métodos de manipulación y representación de la información del modelo. Sin embargo, hay dos funcionalidades que por su importancia pueden considerarse como principales.

Primeramente, el método ***inference***, proporciona la posibilidad de realizar el proceso de inferencia del mapa. Una vez inicializado el mapa, este método calcula y asigna los valores de activación de los distintos conceptos según la ecuación (1.5), guardando en el historial de cada uno dichos valores. Mediante esta función se puede realizar la predicción del comportamiento del sistema en el tiempo, conocidas las condiciones iniciales del mismo, así como sus dinámicas internas, representadas por las matrices causales. En la Figura 2.2 se muestra el flujo del proceso de inferencia del mapa.

Figura 2.2 Diagrama del proceso de inferencia.

Este método hace uso de otras funciones secundarias tales como: ***current_activation*** y ***current_input***, que devuelven los valores de activación de los conceptos y de las entradas del sistema en el momento actual. Estos vectores son multiplicados por las matrices de causalidad. El resultado es acotado por el método ***boundary***, que llama a los métodos estáticos de la clase ***Boundary***. Finalmente, se llama al método ***set_activation*** para actualizar las activaciones de los conceptos, guardando los valores anteriores en el historial.

La otra función principal es el método ***learn***, el cual se encarga del proceso de aprendizaje del mapa. Cuando no se conoce la dinámica del sistema objeto de estudio, y, en su lugar, lo que se posee son datos históricos del comportamiento del mismo, el mapa es capaz de ajustar su estructura interna mediante esta funcionalidad de modo que pueda semejarse lo más posible a la realidad. En la Figura 2.3 se muestra el flujo del proceso de aprendizaje del mapa.

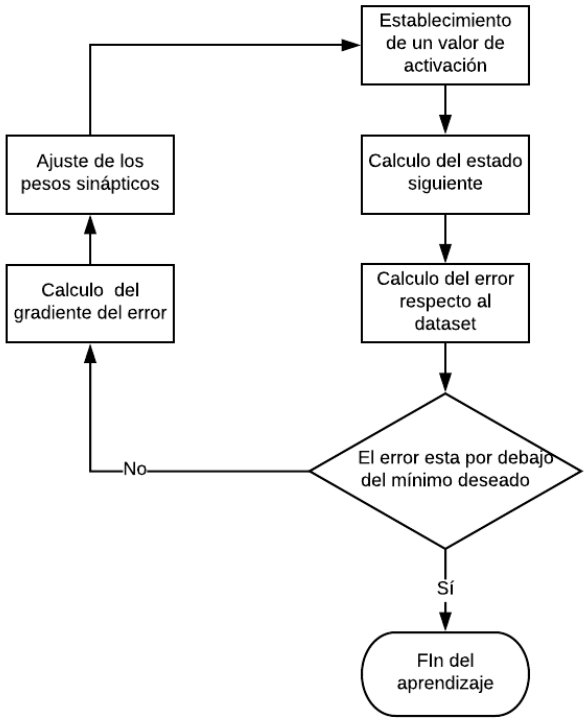

Figura 2.3. Diagrama de flujo del proceso de aprendizaje.

Este método representa un algoritmo iterativo que en cada paso ajusta las conexiones sinápticas del mapa con el objetivo de disminuir cada vez más una función de error. Dicha

función de error se puede calcular y graficar mediante las funciones ***square_error_instant***, ***square_error_throughout*** *y* ***plot_square_error***.

El único algoritmo de aprendizaje programado hasta el momento es el del gradiente descendente, el cual se basa en calcular las derivadas parciales de la función de error respecto a cada uno de los pesos sinápticos del mapa. Para ello, el método ***learn*** hace uso de la función ***d_boundary***, la cual calcula estos diferenciales.

2.4 Metodología para la utilización del software

A la hora de utilizar el paquete ***fcm*** para modelar un sistema en específico se propone seguir la metodología que se expone a continuación (Tabla 2.1). Mediante estos pasos se espera obtener un mejor aprovechamiento y resultado en ese objetivo. Pudiera ser que, en dependencia de las individualidades de cada situación, esta secuencia pudiera sufrir algunas ligeras alteraciones.

Tabla 2.1. Metodología de utilización del paquete ***fcm***.

No.	Acción	Recursos involucrados
1	Determinación de variables	• Expertos
2	Confección del modelo inicial	• Expertos • Clase ***FCM***
3	Evaluación del modelo inicial	• Clase ***FCM*** (cálculo del error)
4	Recopilación de datos históricos	• Sistemas de adquisición de datos
5	Ajuste del modelo mediante ML	• Clase ***FCM*** (aprendizaje)
6	Evaluación del modelo final	• Clase ***FCM*** (cálculo del error)

En el primer paso, el experto determina cuales son las variables del sistema que se deben tener en cuenta como conceptos. Debe establecer para estos conceptos las unidades de

medida, los rangos de valores admisibles, los rangos considerados adecuados. Dentro de esta lista de variables, el experto debe clasificarlas en conceptos internos del sistema y entradas del mismo, además de decir cuáles son las variables objetivo, de cuyos valores depende el rendimiento y el éxito del proceso.

Para la confección del modelo inicial, el experto aporta sus conocimientos sobre cuáles deben ser las relaciones causales entre los distintos conceptos. Dado que esta herramienta está pensada básicamente para sistemas altamente indeterminados y complejos, lo normal es que el experto no pueda aportar un valor específico para estos pesos sinápticos. Por tanto, se pretende que al menos sea capaz decidir sobre una de las cuatro posibles opciones siguientes en cuanto al peso sináptico entre el concepto **A** y **B** (dirigido de **A** hacia **B**):

- Positivo (un cambio en el valor de **A** influye en un cambio en el valor de **B** en el mismo sentido)
- Negativo (un cambio en el valor de **A** influye en un cambio en el valor de **B** en sentido contrario)
- Cero (un cambio en el valor de **A** no influye en el valor de **B**)
- Desconocido (el experto desconoce cuál es el tipo de relación directa entre **A** y **B**)

La evaluación de este modelo se basa en dos acciones. Primero se establecen las condiciones iniciales y se realiza el proceso de inferencia. Luego, los resultados de este proceso se comparan con los datos históricos mediante el cálculo y el graficado del error.

Luego de realizado este trabajo, se pasa a la próxima fase, durante la cual, teniendo en cuenta las variables definidas en la primera etapa, se procede a recopilar valores históricos de las mismas. Aquí es muy importante la correcta elección del período de muestreo, que no es más que el espacio de tiempo que transcurre entre una medición y otra. Dicho período depende, sobre todo, de la velocidad con que se manifiestan las dinámicas internas del sistema y la rapidez con que cambian los valores de los conceptos. Durante esta etapa es importante la forma en la que se almacenan los valores, siendo recomendable utilizar bases de datos en algún sistema empotrado, de la cual puedan ser extraídos los mismos mediante alguna interfaz que los traduzca al formato con que trabaja el paquete ***fcm***.

En la etapa de ajuste del modelo se utiliza algún método de aprendizaje para acercar la respuesta el máximo posible a los datos recopilados. En este proceso se debe utilizar el 70%

de estos valores que fueron previamente almacenados en la etapa anterior. Aquí es posible que se tenga que variar el factor de aprendizaje del mapa hasta obtener los mejores resultados posibles.

Por último, se procede a la validación del resultado final, para lo cual se compara la respuesta del modelo ajustado con el 30% restante de los datos medidos. Calculando y graficando el error que el mapa presenta, el experto puede aprobar o no la veracidad del mismo. En la Figura 2.4 se muestra como debe ubicarse el *software* dentro de un proceso general.

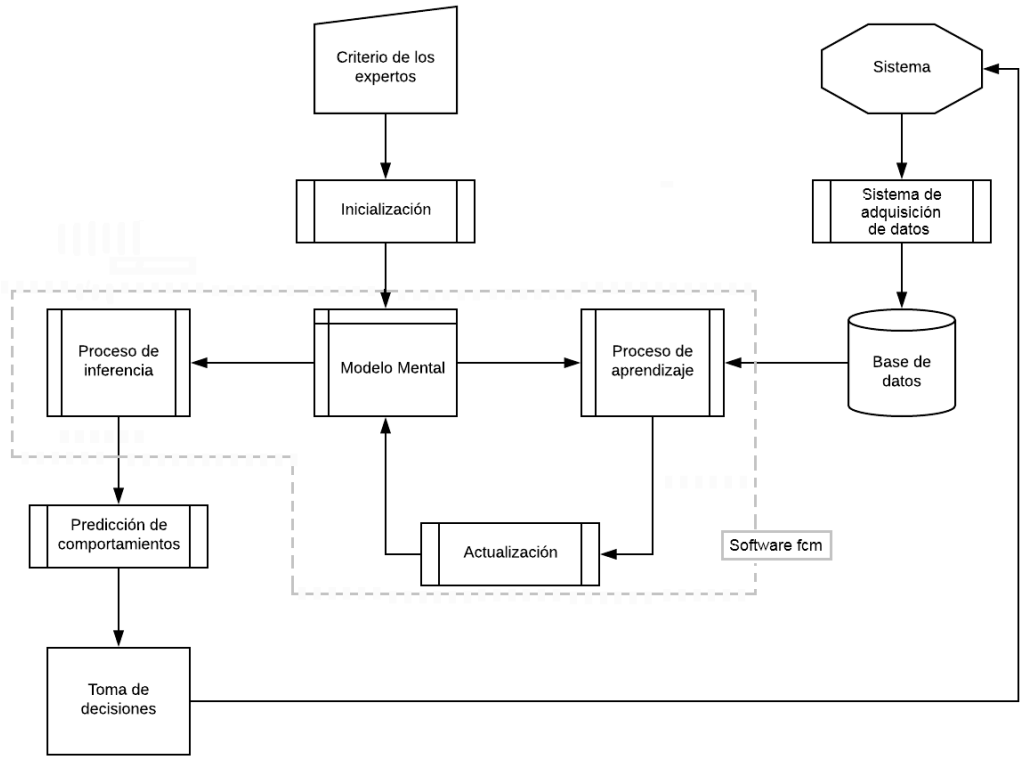

Figura 2.4. Posible utilización del *software* dentro de un proceso real.

2.5 Consideraciones del capítulo

La utilización de plataformas libres y de código abierto en el desarrollo de la biblioteca para Mapas Cognitivos Difusos, convierten a la misma en un proyecto portable y fácilmente actualizable. Lo expuesto durante este capítulo sirve, entre otras cosas, como una fuente de documentación del *software*. Se considera importante que los futuros usuarios del mismo

cuenten con este tipo de información, jugando un papel fundamental la metodología de uso aquí expuesta.

El proceso de desarrollo de la biblioteca *fcm*, aunque cuenta con una primera versión estable y funcional, posee una continuidad de futuras actualizaciones y mejoras. Con esto se pretende acercarse cada vez más a una herramienta que implemente la teoría de FCM lo más eficiente y abarcadora posible.

CAPÍTULO 3. RESULTADOS OBTENIDOS Y PROPUESTA DE APLICACIÓN

Una vez desarrollada la primera versión estable y funcional del paquete *fcm*, herramienta de *software* para el trabajo con FCM, se hizo necesario realizar una serie de pruebas para comprobar su validez. Aunque el principal objetivo de las técnicas de AI, tal y como se ha explicado anteriormente, es el modelado de sistemas altamente no lineales y con dinámicas desconocidas, sería prácticamente imposible probar a cabalidad el programa resultante. Debido a esto se elige un sistema totalmente determinista, un circuito oscilante, cuyas dinámicas y relaciones son conocidas a priori con una gran exactitud, permitiendo una comparación exacta que evalúe el resultado.

Una vez demostrada la validez de la herramienta en este tipo de sistemas, tanto en el proceso de inferencia y simulación como en el de aprendizaje, se hace una propuesta de modelo para uno complejo y no determinista, el comportamiento de las plantaciones de una casa de cultivo. A causa del largo tiempo necesitado para la recopilación de datos de un proceso tan lento, como lo es la evolución de una cosecha, solo se cumple en esta investigación con la primera fase metodológica del procedimiento propuesto (Tabla 2.1).

3.1 Circuito RLC modelado como Mapa Cognitivo Difuso

A continuación, se establecen los modelos de tiempo discreto de un sistema determinista y totalmente conocido, un circuito RLC de tercer orden con una entrada de voltaje. Esta prueba persigue principalmente dos objetivos, demostrar la similitud entre los mapas cognitivos y los modelos de tiempo discreto, y probar el paquete *fcm* en un sistema del cual se conoce su comportamiento exacto y que, por tanto, permite realizar comparaciones.

3.1.1 Obtención del modelo en espacio de estados

En un circuito eléctrico existen componentes almacenadores de energía, los cuales determinan el orden del mismo. Estos componentes son básicamente los capacitores, los cuales almacenan energía en forma de campo eléctrico y los inductores los cuales lo hacen en forma de campo magnético. Supóngase la configuración de la Figura 3.1:

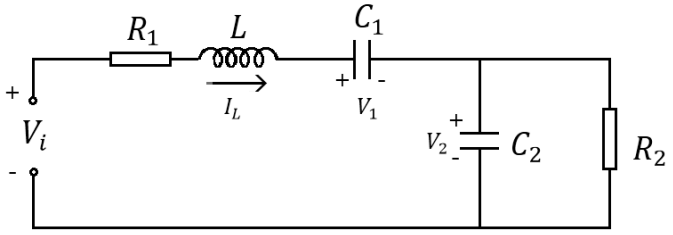

Figura 3.1. Circuito RLC de tercer orden.

Aplicando las leyes de Kirchhoff se obtiene el modelo del sistema en espacio de estados. Para ello se buscan las expresiones de la primera derivada temporal de la corriente en el inductor (3.1), y de los voltajes en los capacitores, (3.2) y (3.3):

$$V_i - I_L R_1 - L\dot{I}_L - V_1 - V_2 = 0$$

$$\dot{I}_L = -\frac{1}{L}V_1 - \frac{1}{L}V_2 - \frac{R_1}{L}I_L + \frac{1}{L}V_i \tag{3.1}$$

$$\dot{V}_1 = \frac{1}{C_1}I_L \tag{3.2}$$

$$\dot{V}_2 = \frac{1}{C_2}(I_L - I_R)$$

Pero:

$$I_R = \frac{V_2}{R_2}$$

Luego:

$$\dot{V}_2 = \frac{1}{C_2}\left(I_L - \frac{V_2}{R_2}\right)$$

$$\dot{V}_2 = -\frac{1}{R_2 C_2} V_2 + \frac{1}{C_2} I_L \tag{3.3}$$

Combinando estas tres ecuaciones se obtiene el modelo en forma matricial (3.4):

$$\begin{bmatrix} \dot{V}_1 \\ \dot{V}_2 \\ \dot{I}_L \end{bmatrix} = \begin{bmatrix} 0 & 0 & \frac{1}{C_1} \\ 0 & -\frac{1}{R_2 C_2} & \frac{1}{C_2} \\ -\frac{1}{L} & -\frac{1}{L} & -\frac{R_1}{L} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ I_L \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{L} \end{bmatrix} [V_i] \tag{3.4}$$

3.1.2 Discretización del modelo

En la ecuación (3.4) se distinguen perfectamente las matrices A y B (2.4) por lo que ya se puede encontrar el modelo en tiempo discreto si se aplican las ecuaciones (2.30) y (2.33). El mismo está descrito por (3.5).

$$\begin{bmatrix} V_1^{k+1} \\ V_2^{k+1} \\ I_L^{k+1} \end{bmatrix} = G \begin{bmatrix} V_1^k \\ V_2^k \\ I_L^k \end{bmatrix} + H\left[V_i^k\right] \tag{3.5}$$

Donde las matrices G (3.6) y H (3.7) determinan la dinámica del modelo.

$$G = \begin{bmatrix} 1 - \frac{T^2}{2LC_1} & -\frac{T^2}{2LC_1} & \frac{T}{C_1} - \frac{R_1 T^2}{2LC_1} \\ -\frac{T^2}{2LC_2} & \frac{T^2}{2R_2^2 C_2^2} - \frac{T^2}{2LC_2} - \frac{T}{L} & -\frac{T^2}{2R_2 C_2^2} - \frac{T^2 R_1}{2LC_2} + \frac{T}{C_2} \\ \frac{R_1 T^2}{2L^2} - \frac{T}{L} & \frac{R_1 T^2}{2L^2} + \frac{T^2}{2LR_2 C_2} - \frac{T}{L} & 1 - \frac{R_1 T}{L} + \frac{R_1^2 T^2}{2L^2} - \frac{T^2}{2LC_2} - \frac{T^2}{2LC_1} \end{bmatrix} \tag{3.6}$$

$$H = \begin{bmatrix} \dfrac{T^2}{2LC_1} \\ \dfrac{T^2}{2LC_2} \\ \dfrac{T - \dfrac{R_1 T^2}{2L}}{L} \end{bmatrix} \tag{3.7}$$

En consecuencia, con la analogía establecida en este proyecto entre FCM y modelos dinámicos en tiempo discreto se puede representar la información anterior mediante un mapa cognitivo tal y como se muestra en la Figura 3.2.

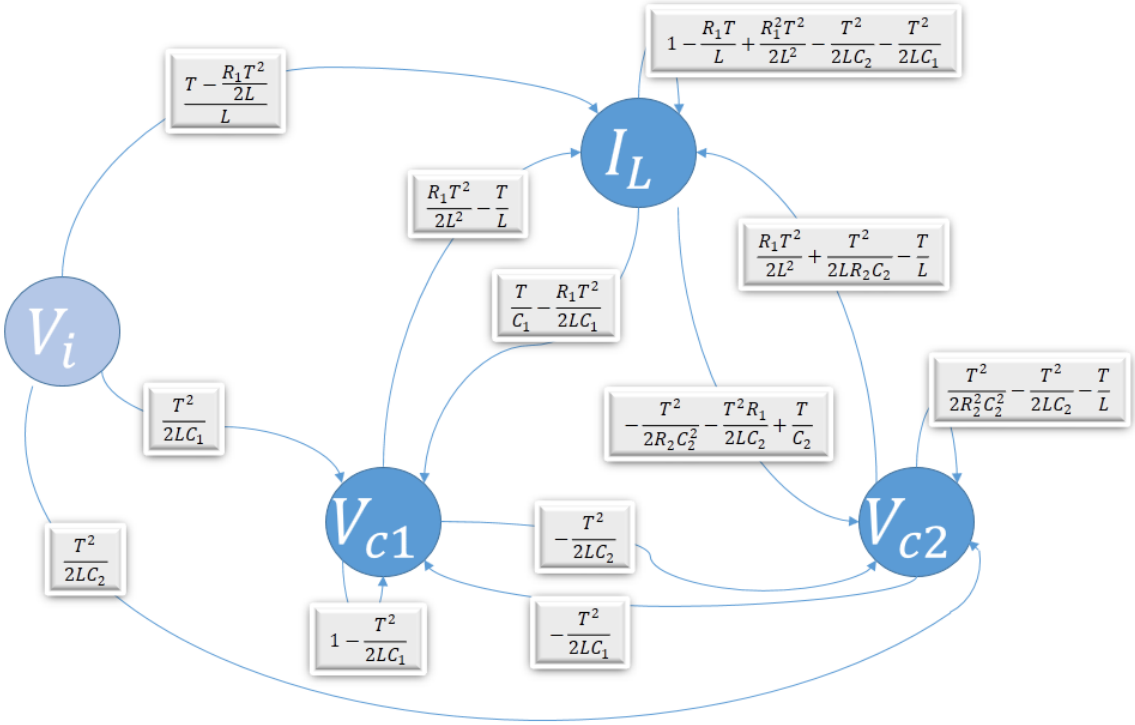

Figura 3.2. Representación en forma de FCM del circuito RLC.

3.1.3 Diseño y resultados del experimento

La realización del experimento se realizó en dos etapas. En cada una de ellas se puso a prueba una de las dos funcionalidades del *software*, los procesos de inferencia y aprendizaje. Para la primera se realizaron los pasos siguientes:

1. Simulación del circuito en el *software* Multisim.

2. Obtención de las gráficas del comportamiento de las variables de estado en la simulación.
3. Representación del circuito en un FCM mediante la herramienta de *software.*
4. Obtención de las gráficas del comportamiento de los conceptos en la herramienta de *software*.
5. Comparación de gráficas.

En la Figura 3.3 se muestra la simulación en el *software* Multisim del circuito RLC objeto del experimento.

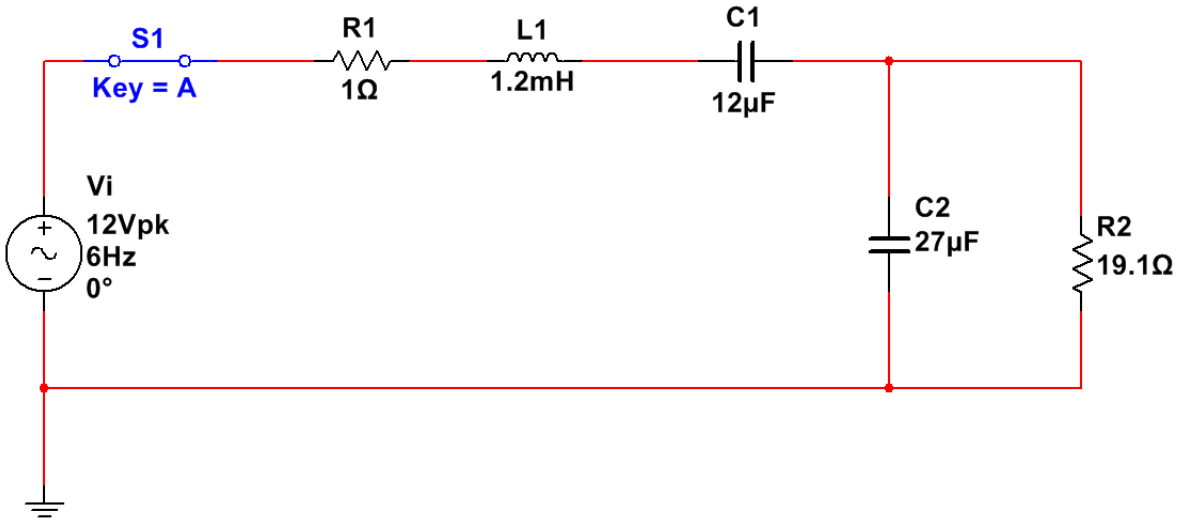

Figura 3.3. Circuito RLC de tercer orden con una entrada de voltaje.

A continuación, se muestran las gráficas obtenidas tanto en esta simulación como en el modelado mediante el FCM para que se pueda apreciar la similitud de ambos resultados. La Figura 3.4 muestra la evolución del voltaje en el capacitor 1 en la simulación del circuito, mientras que la Figura 3.5 muestra lo mismo pero según el proceso de inferencia realizado por la biblioteca fcm.

Además, se brinda la misma comparación para el voltaje en el capacitor 2 (Figura 3.6 y Figura 3.7) y para la corriente en el inductor (Figura 3.10 y Figura 3.11).

Puesto que una de las principales ventajas de las modificaciones hechas a la teoría de los FCM es la correcta simulación de los estados transitorios, los mismos se muestran para el voltaje en el capacitor 2 (Figura 3.8 y Figura 3.9) y para la corriente en el inductor (Figura

3.12 y Figura 3.13). Fueron escogidos estos conceptos por poseer estados transitorios bastante característicos (a diferencia del voltaje en el capacitor 1).

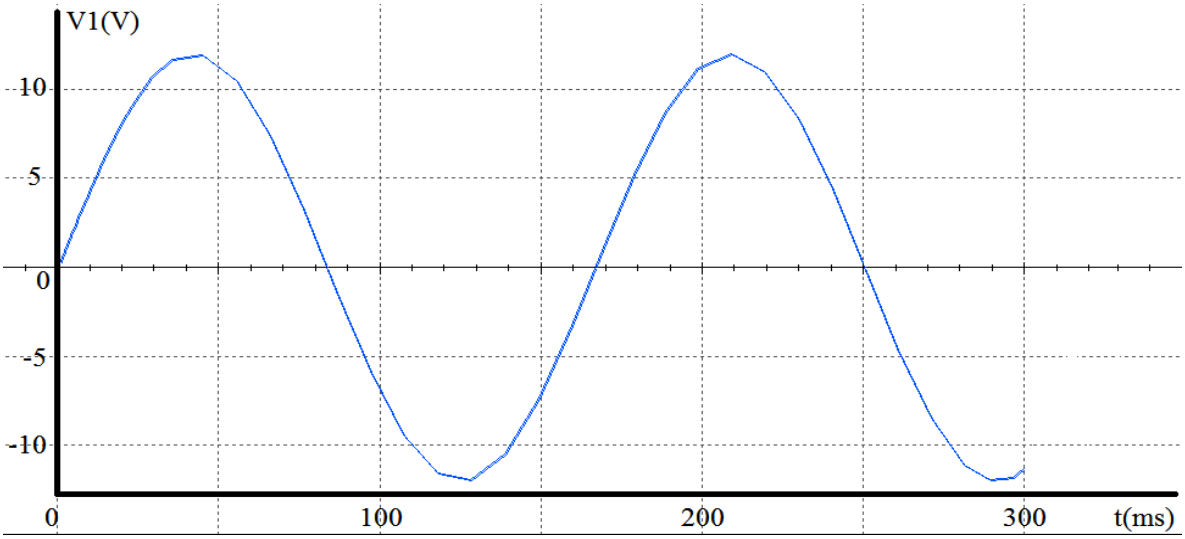

Figura 3.4. Voltaje en el capacitor 1. Simulación en Multisim.

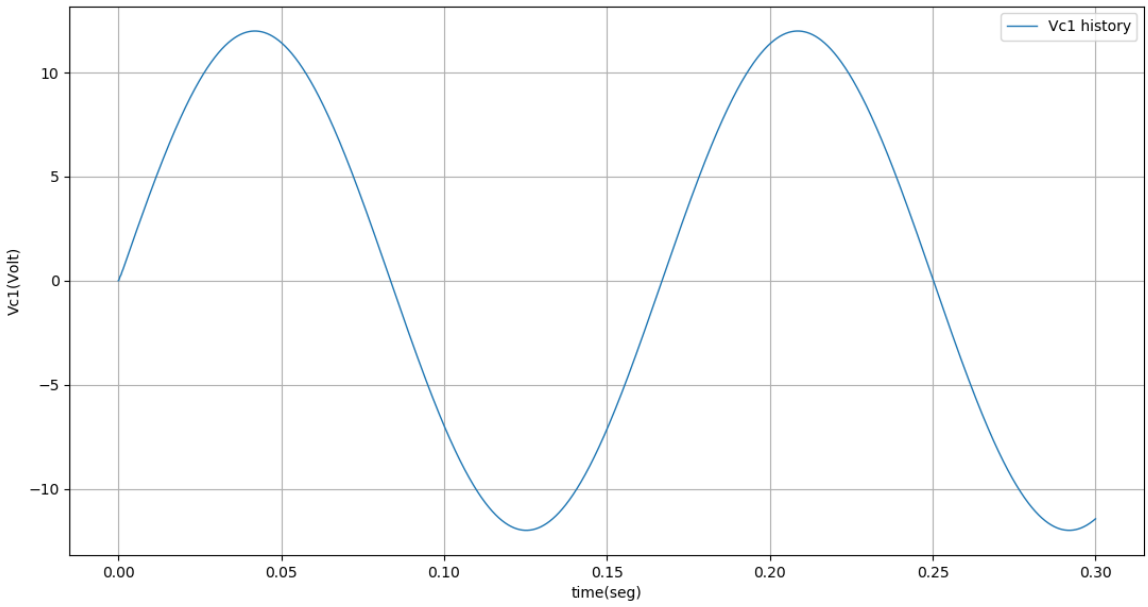

Figura 3.5. Voltaje en el capacitor 1. Biblioteca fcm.

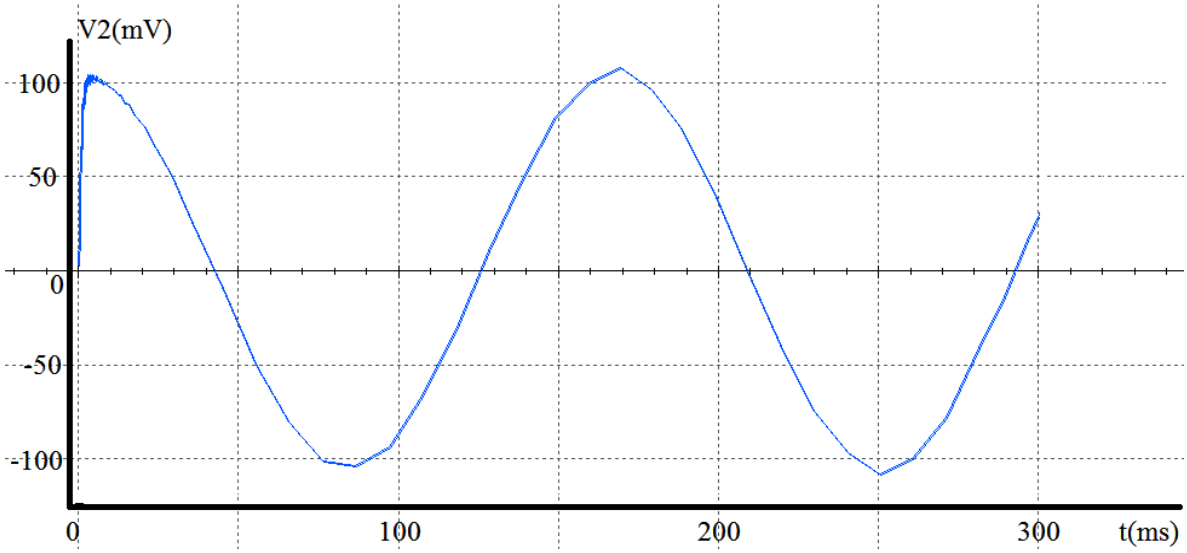

Figura 3.6. Voltaje en el capacitor 2. Simulación en Multisim.

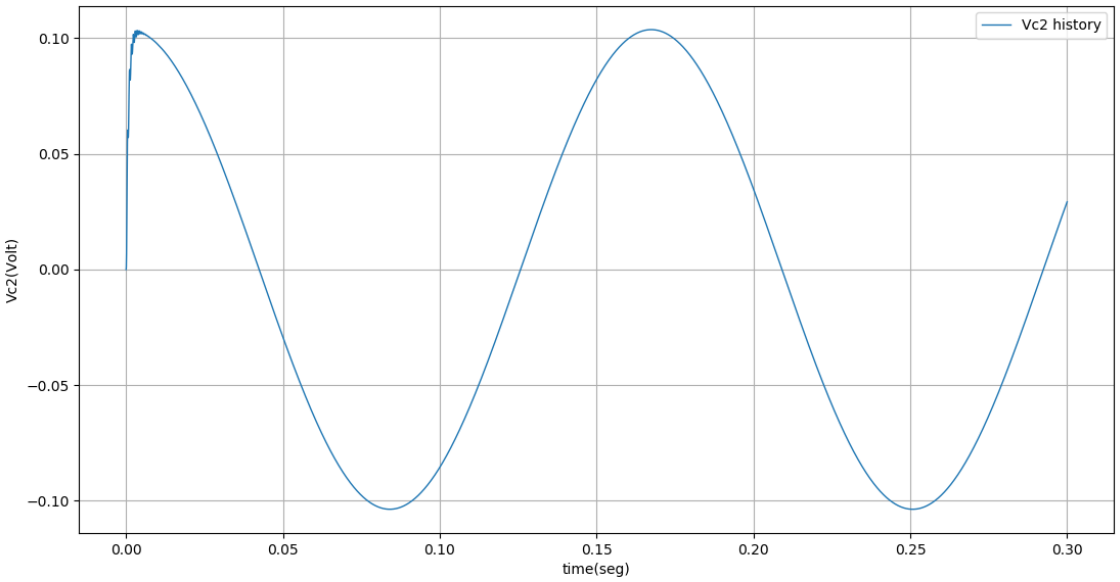

Figura 3.7. Voltaje en el capacitor 2. Biblioteca fcm.

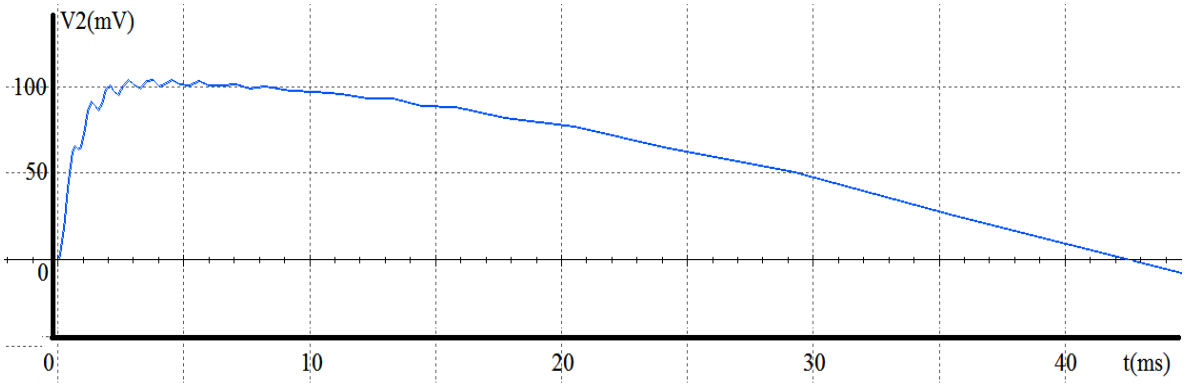

Figura 3.8. Voltaje en el capacitor 2, estado transitorio. Simulación en Multisim.

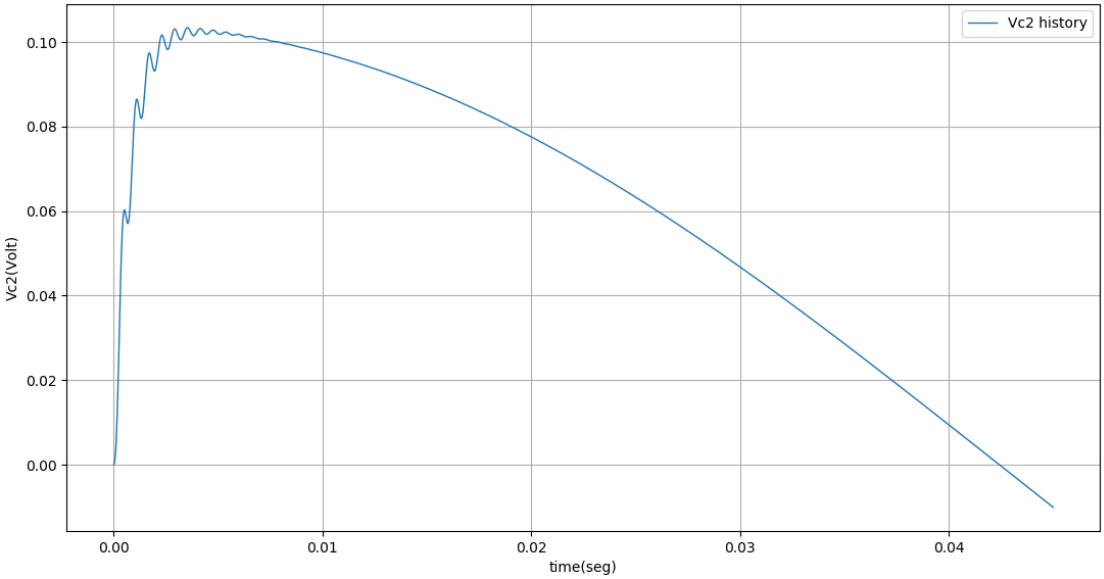

Figura 3.9. Voltaje en el capacitor 2, estado transitorio. Biblioteca fcm.

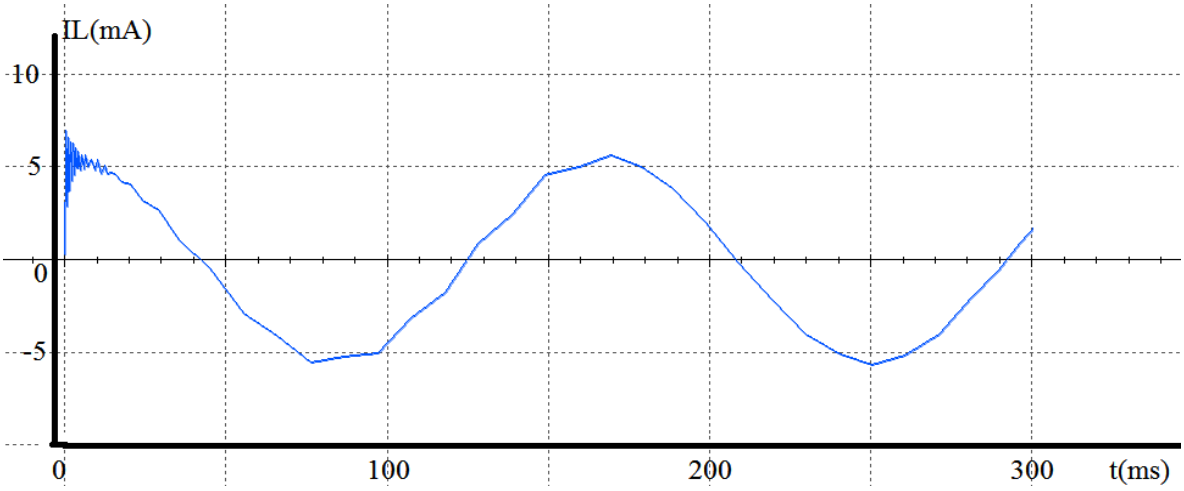

Figura 3.10. Corriente en el inductor. Simulación en Multisim.

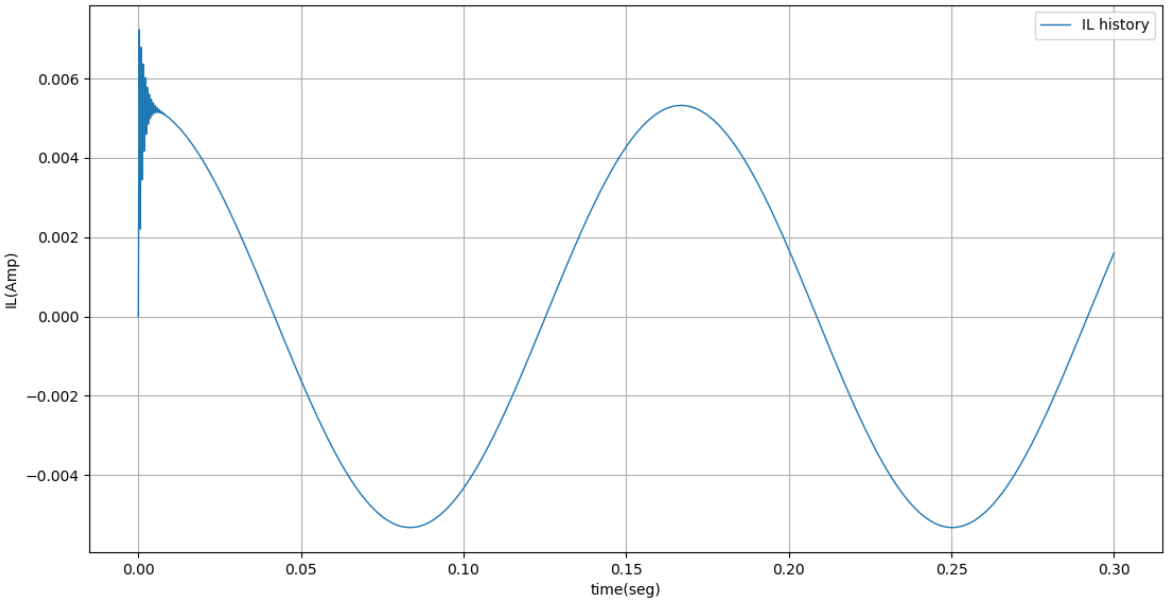

Figura 3.11. Corriente en el inductor. Biblioteca fcm.

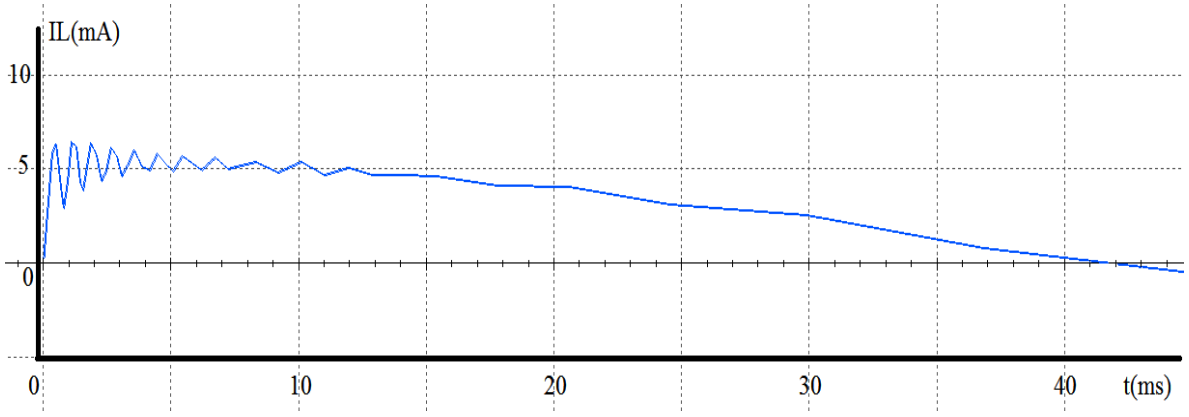

Figura 3.12. Corriente en el inductor, estado transitorio. Simulación en Multisim.

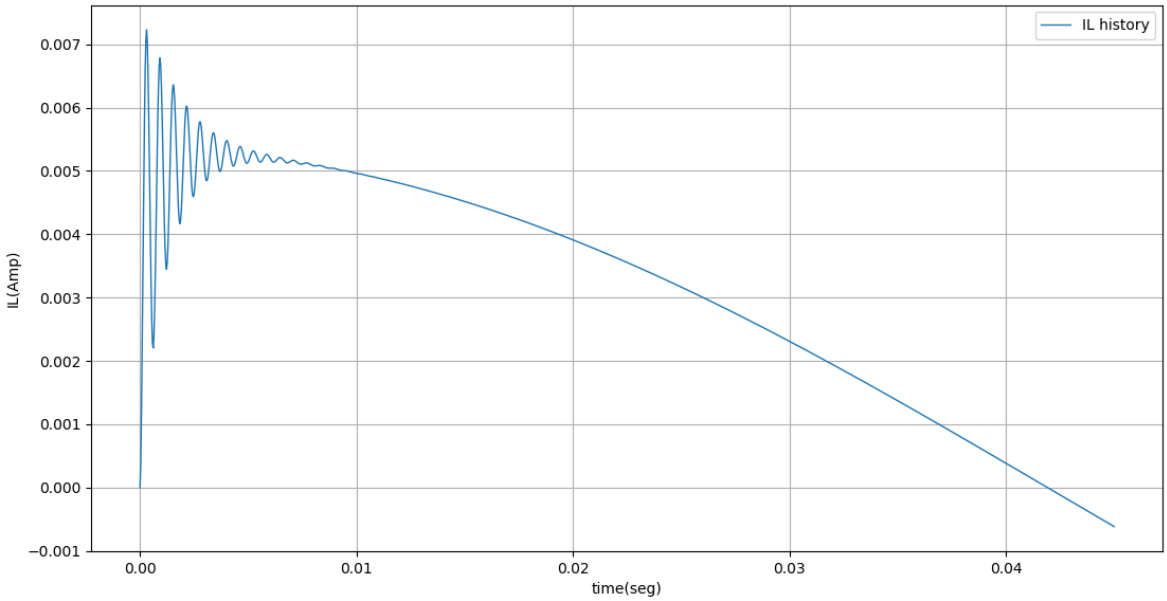

Figura 3.13. Corriente en el inductor, estado transitorio. Biblioteca fcm.

En una segunda etapa se comprobó la funcionalidad del proceso de aprendizaje de la herramienta de *software*. Para esto se utilizó la configuración de la Figura 3.14, similar a la anterior, pero sin el voltaje sinusoidal de entrada. En este caso la fuente de corriente directa solo se usa para establecer las condiciones iniciales del sistema.

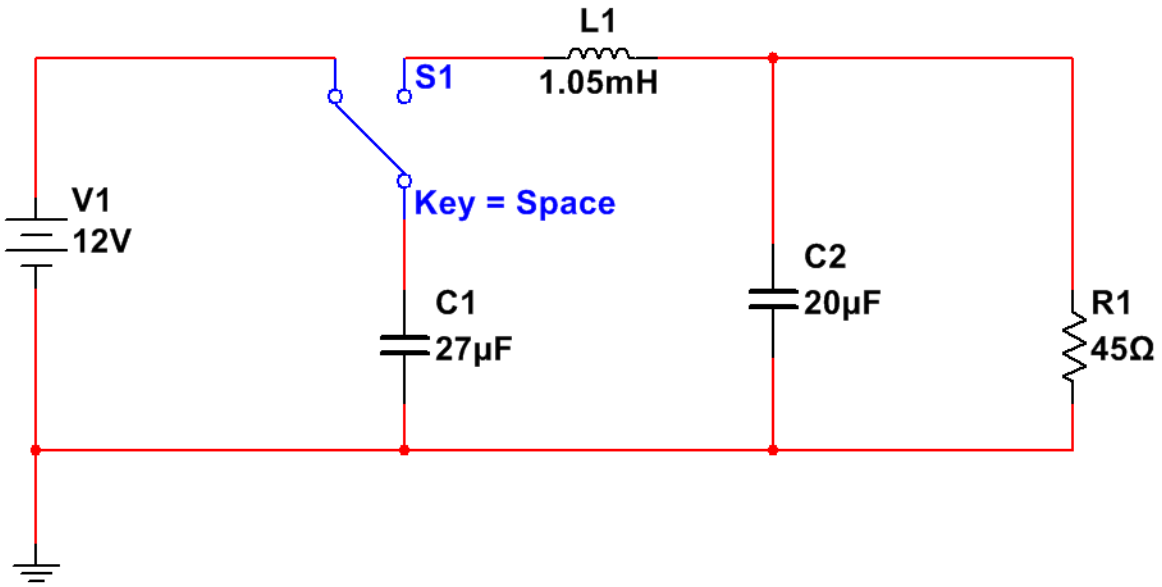

Figura 3.14. Circuito RLC de tercer orden sin entradas.

Este segundo experimento se realizó siguiendo los pasos siguientes:

1. Montaje del circuito mediante la herramienta de *software.*
2. Generación de un set de datos usando el proceso de inferencia (probado en la fase anterior del experimento).
3. Creación de un modelo en forma de FCM con parámetros arbitrarios.
4. Cálculo y graficado del error del resultado de este modelo respecto al set de datos.
5. Realización del proceso de aprendizaje del modelo utilizando el set de datos.
6. Repetición del paso No. 4 y comparación.

Como parte de la generación del set de datos se le impusieron diez juegos de condiciones iniciales al circuito obteniéndose diez juegos de datos distintos. La razón de esto es utilizar siete de estos en el proceso de aprendizaje y los tres restantes en la validación del resultado. De este modo se puede comprobar la capacidad de generalización del modelo al enfrentarlo a situaciones no vistas durante el aprendizaje.

A estos datos se le introdujo un nivel de ruido para hacerlos lo más semejantes posible a los que se hubieran medido en un sistema real. A continuación, se muestra el comportamiento del modelo antes del proceso de aprendizaje en comparación con los datos experimentales

(Figura 3.15, Figura 3.16 y Figura 3.17). La línea azul corresponde a los datos experimentales mientras que la roja, al comportamiento del modelo.

Además, se muestra la curva correspondiente al error cuadrático medio relativo a este comportamiento (Figura 3.18). Las gráficas mostradas corresponden a uno de los sets de datos utilizados en el proceso de validación. En el Anexo IV se ofrecen los resultados relativos a los tres sets de datos usados.

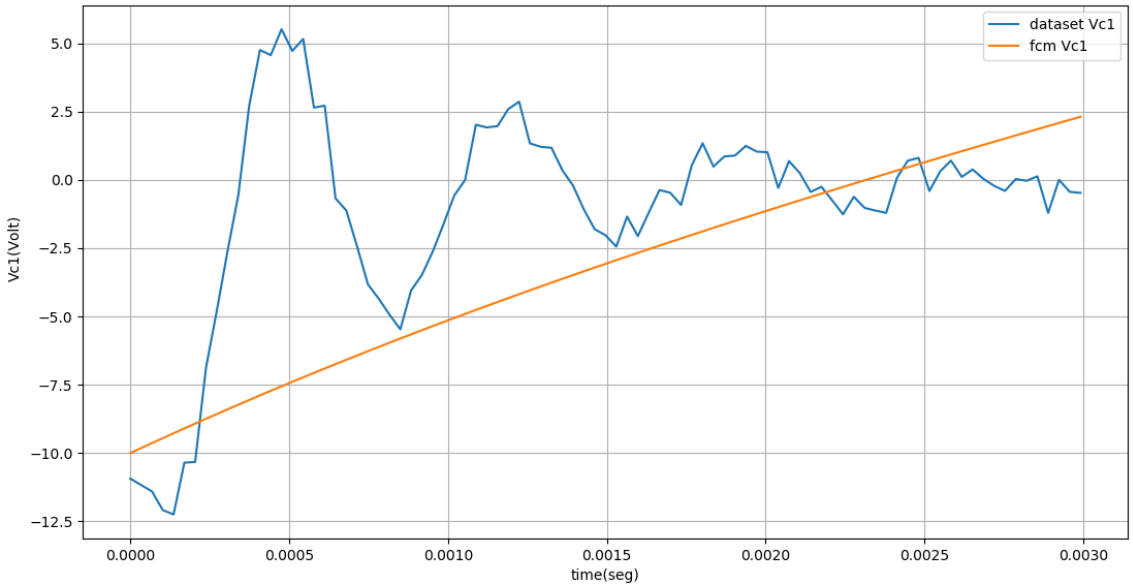

Figura 3.15. Voltaje en el capacitor 1 antes del aprendizaje.

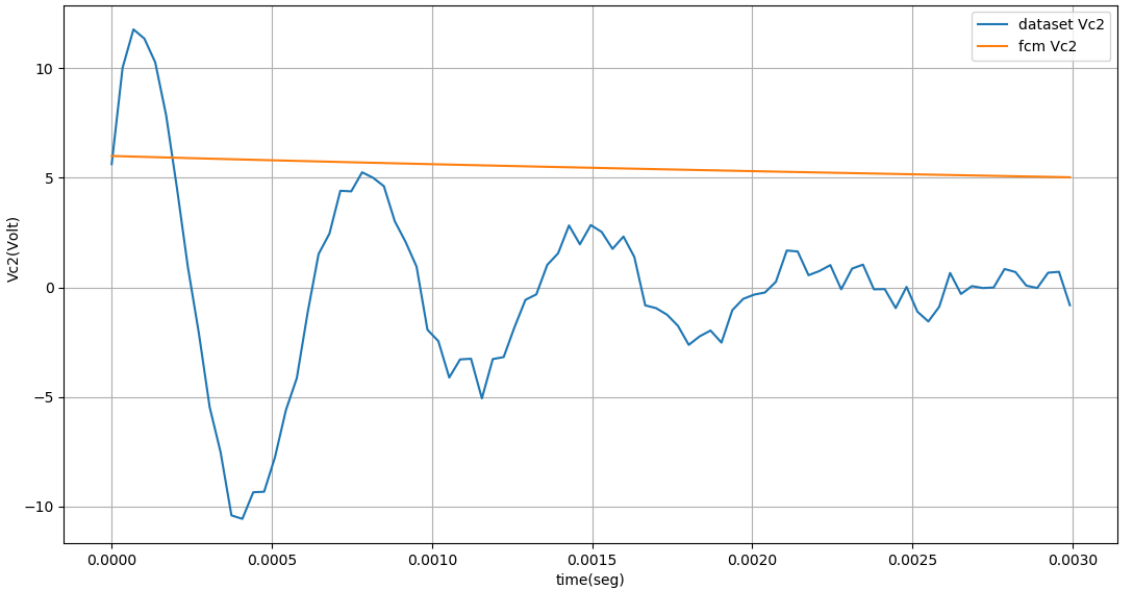

Figura 3.16. Voltaje en el capacitor 2 antes del aprendizaje.

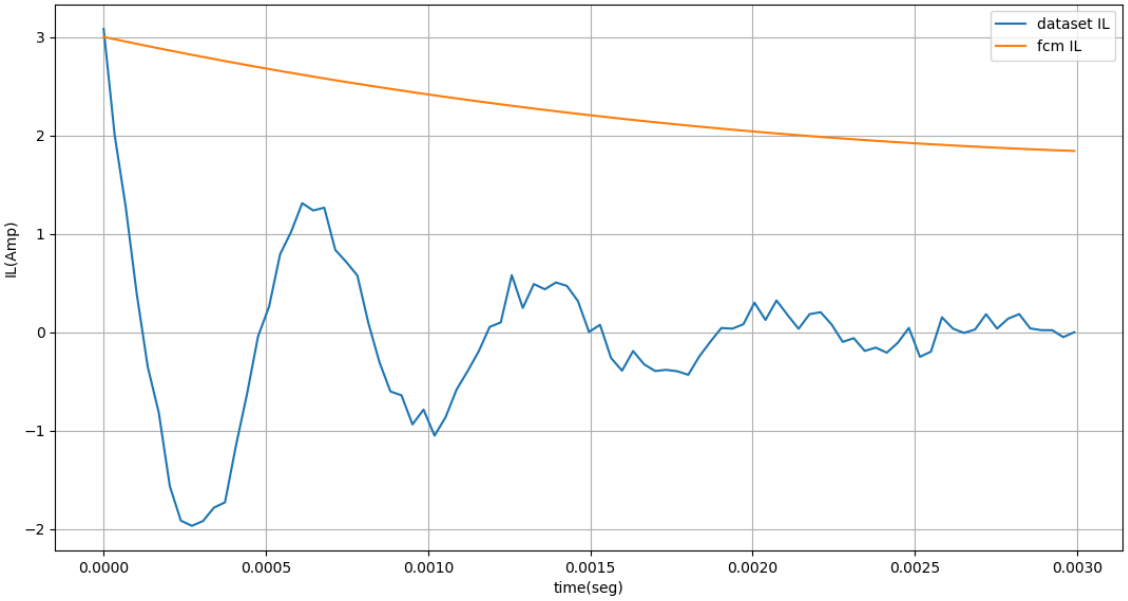

Figura 3.17. Corriente en el inductor antes del aprendizaje.

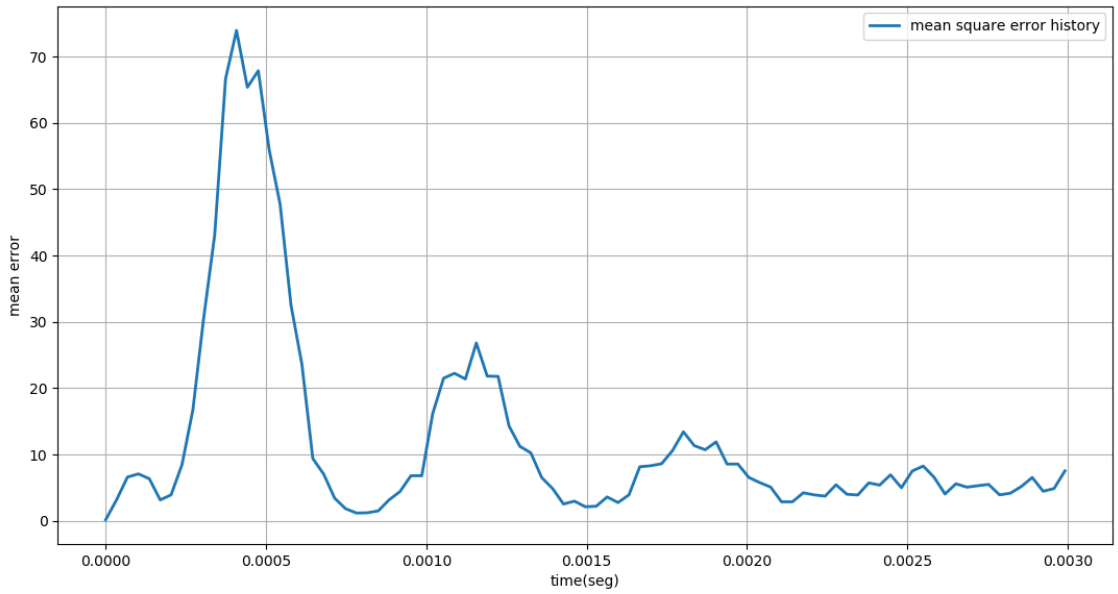

Figura 3.18. Error cuadrático medio antes del aprendizaje.

Luego de realizar el proceso del aprendizaje para ajustar los pesos sinápticos del modelo se obtienen nuevamente las gráficas de comportamiento (Figura 3.19, Figura 3.20 y Figura 3.21) y de error cuadrático medio (Figura 3.22). Comparando estas gráficas con las anteriores se puede concluir que el proceso de aprendizaje fue exitoso puesto que el modelo se ajusta al comportamiento real del sistema.

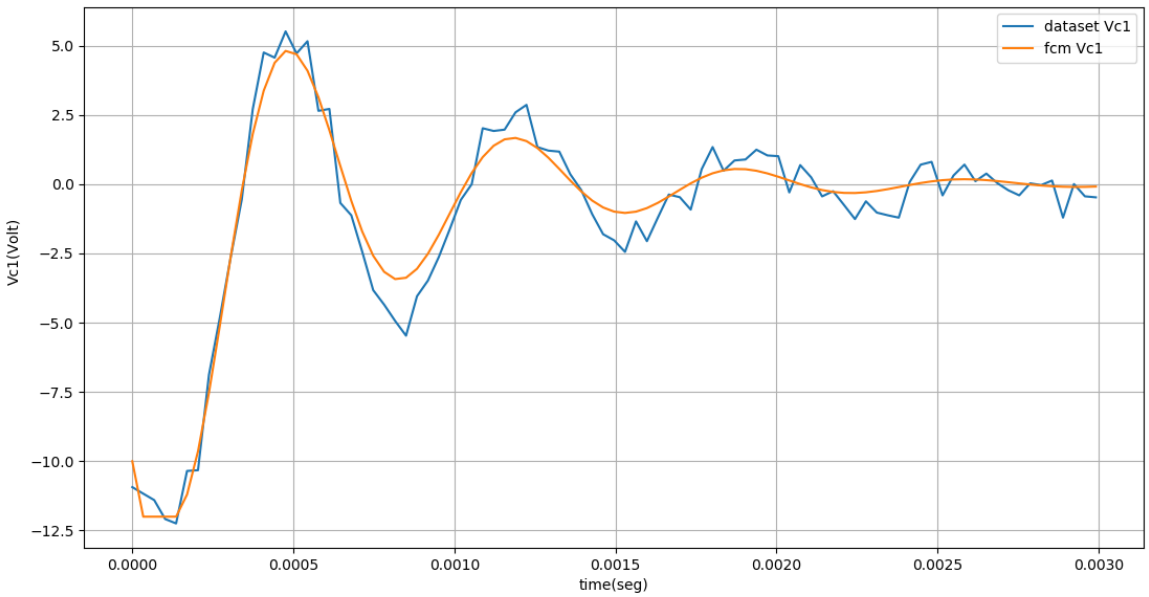

Figura 3.19. Voltaje en el capacitor 1 después del aprendizaje.

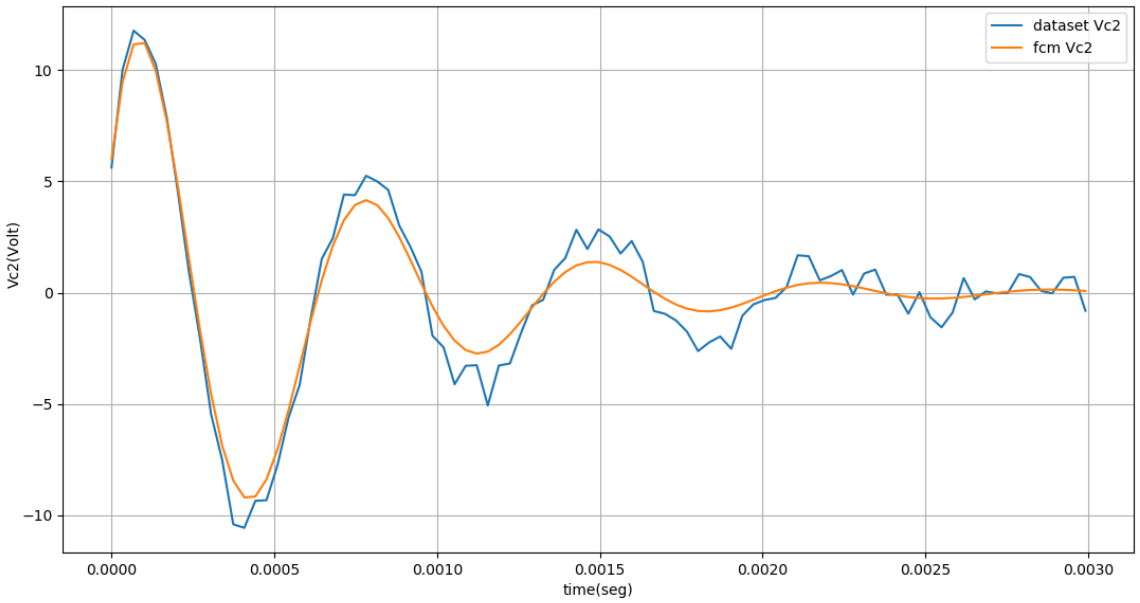

Figura 3.20. Voltaje en el capacitor 2 después del aprendizaje.

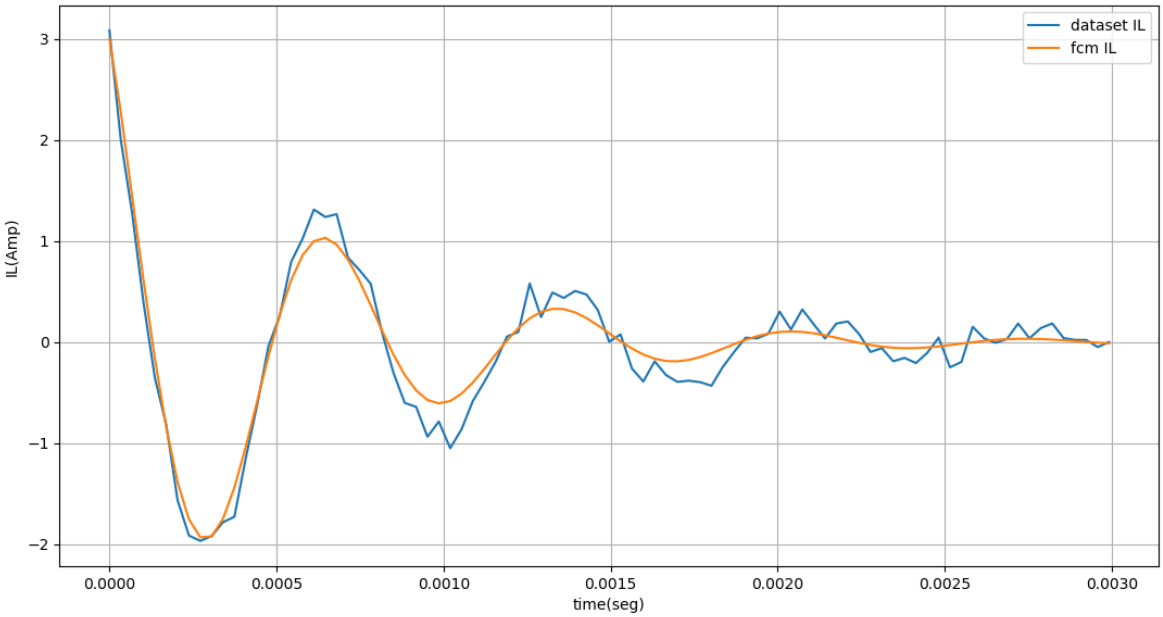

Figura 3.21. Corriente en el inductor después del aprendizaje.

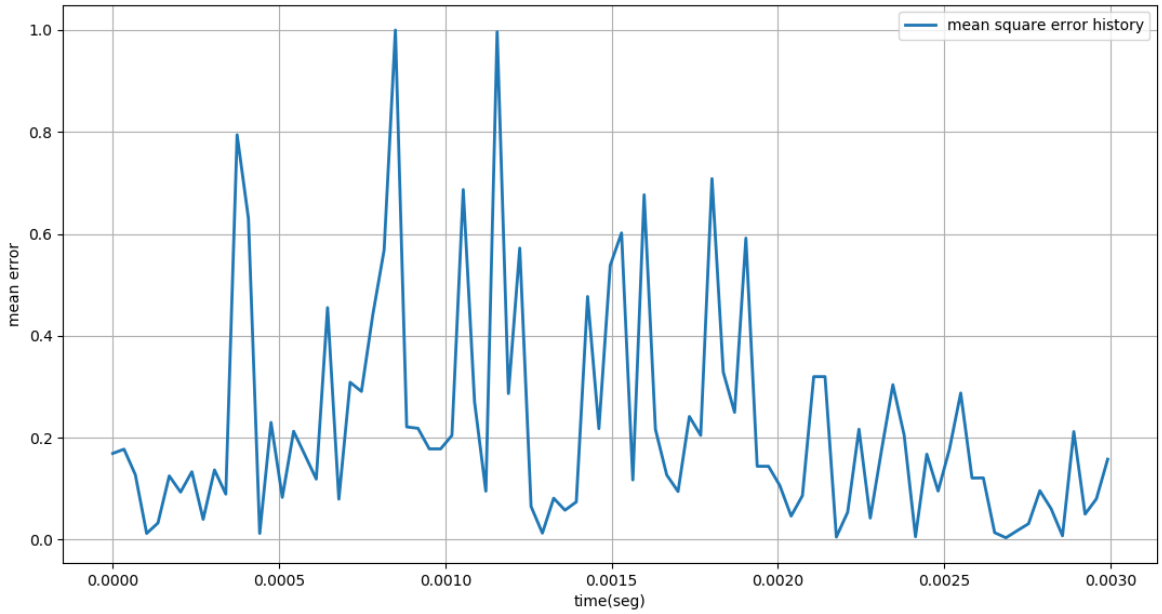

Figura 3.22. Error cuadrático medio después del aprendizaje.

El proceso de aprendizaje basado en el algoritmo del gradiente descendente es iterativo y su objetivo es disminuir el error respecto a los datos de referencia en cada iteración. Ahora se muestra el comportamiento del error cuadrático medio durante las iteraciones de este proceso (Figura 3.23). Esta gráfica se obtuvo buscando el error máximo cada diez pasos iterativos de entrenamiento de la red.

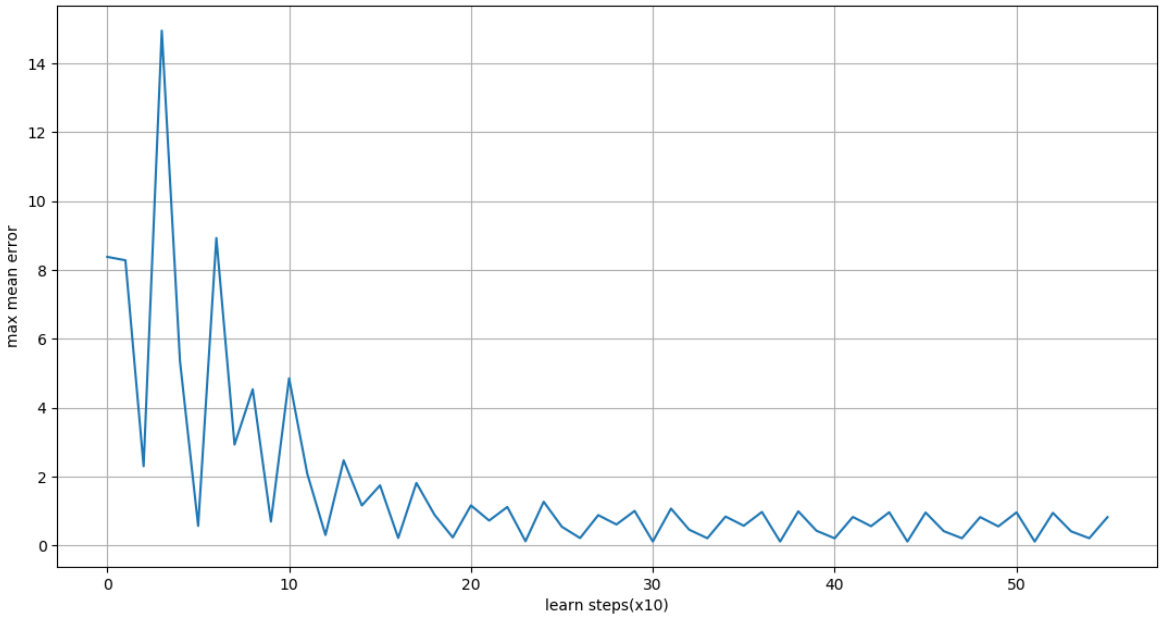

Figura 3.23. Evolución del error cuadrático medio durante el aprendizaje.

3.2 Propuesta de modelo de fertirrigación para casas de cultivo

Para la realización del modelo de la casa de cultivo se tienen en cuenta las principales variables que lo componen. Cada una de estas variables representa un concepto del mapa y se puede clasificar de varias maneras. En este aspecto juega un papel fundamental el criterio de los distintos expertos. Ellos son los que tienen la capacidad de aportar una idea inicial de cuáles son las variables que se deben tener en cuenta a la hora de modelar el sistema.

3.2.1 Definición de variables

Las variables fundamentales para el modelo se pueden clasificar en cuatro grupos tal y como se muestra en la Tabla 3.1:

Tabla 3.1 Variables del sistema según su tipo.

Condiciones del suelo	**Condiciones climatológicas**	**Riego y fertilización**	**Características fenotípicas de la planta**
Humedad del suelo	Temperatura media diurna	Cantidad de agua	Tamaño de la raíz
Conductividad eléctrica de la disolución	Temperatura media nocturna	Concentración de sulfato de magnesio	Tamaño de la planta
pH de la disolución	Humedad relativa media diurna	Concentración de sulfato de potasio	Diámetro de la base del tallo
	Humedad relativa media nocturna	Concentración de nitrato de magnesio	Número de flores
	Diferencia de temperatura entre el día y la noche	Concentración de nitrato de potasio	Número de racimos
	Diferencia de humedad relativa entre el día y la noche	Concentración de nitrato de calcio	Número de frutos
	Iluminación media diurna	Concentración de quelato de calcio	Coloración de los frutos
	Tiempo de iluminación diario	Concentración de fosfato monopotásico	
		Concentración de plantar micro	
		Concentración de ácido fosfórico	
		Concentración de ácido nítrico	

Otro modo de organizar estos conceptos es atendiendo a como se manifiestan en el modelo, según lo cual pueden ser variables de entrada o variables internas del sistema. Es necesario destacar que las entradas se separan en manipulables (en este caso las relacionadas al riego) y no manipulables (como las climatológicas). Por otra parte, se debe aclarar también que dentro de las variables internas del sistema existen algunas que definen los objetivos y metas del proceso, funcionando de algún modo como variables de salida (en este caso los indicadores fenotípicos de las plantas).

Se pretende que en futuras investigaciones se pueda contar con datos históricos de estas plantaciones, suficientes para completar el modelo basado en el *software* aquí desarrollado. Para esto se recomienda continuar con los siguientes pasos propuestos en la metodología de utilización (Tabla 2.1).

3.3 Análisis económico y medioambiental

En la actualidad, en un mundo sobrepoblado y contaminado como el nuestro, el ahorro de recursos y el cuidado del medio ambiente juegan un papel cada vez más protagónico en las estrategias a largo plazo. Cualquier empresa productora se ve entonces obligada a garantizar una sostenibilidad económica y medioambiental. Las técnicas de AI, como la desarrollada en este proyecto, pueden contribuir en gran medida a la optimización de los procesos productivos con el fin de utilizar un mínimo de materias primas y recursos naturales.

Si se analiza la propuesta de aplicación del *software* hecha en este capítulo se puede entender más claramente esta idea. En una casa de cultivo se obtienen, en general, mejores resultados que campos abiertos. Para ello se invierten recursos en el desarrollo de estructuras para la protección, el riego y la fertilización.

Por una parte, se debe tener en cuenta que estos recursos generan gastos adicionales. Esto demanda un resultado realmente superior en la producción que la haga económicamente sustentable. La AI como herramienta de modelado y monitorización cumple la función de organizar y dosificar estos recursos para que sean aprovechados al máximo. El agua, los fertilizantes y los pesticidas son ejemplos que pueden llegar a escasear o a tener altos precios en el mercado.

Por otro lado, estos mismos recursos tienen un impacto medioambiental tanto en los lugares de donde son obtenidos como en los propios terrenos donde son usados. Es por esto que utilizar solo las cantidades necesarias y suficientes no solo ofrece un beneficio económico, sino que contribuye al cuidado del entorno.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

Mediante la realización de este proyecto se llegó a las siguientes conclusiones:

1. El modelado de sistemas dinámicos discretos se puede realizar a través de FCM debido a la similitud matemática que existe entre ellos.
2. El uso de Anaconda como distribución de Python, así como los paquetes numpy y matplotlib, asegura el diseño del paquete de FCM con un tiempo de desarrollo más corto.
3. El experimento realizado en el circuito RLC demuestra que el paquete FCM permite el modelado de sistemas discretos mediante aprendizaje automático y la predicción del comportamiento de sistemas mediante el proceso de inferencia.

Recomendaciones

Este trabajo está enfocado a la creación de una biblioteca de *software* y se pretende que la misma tenga un continuo desarrollo. Para esto se recomiendan las siguientes acciones futuras:

1. Implementar otros métodos de ajuste más complejos y eficaces, como alternativa al del gradiente descendente, así como estrategias para evitar mínimos locales.
2. Realizar una interfaz gráfica para los usuarios sin conocimientos de programación, que necesiten el uso de la herramienta. Dicha interfaz facilitaría la visualización y el análisis de los datos, algo tan importante en las ciencias de la información.

REFERENCIAS BIBLIOGRÁFICAS

Ahvenlampi, T., Rantanen, R., Tervaskanto, M., Kortela, U., 2004. Enhancing Controllability of Kappa Number Using Gray-Box Modeling. IFAC Proc. Vol. 37, 85–90.

Axelrod, R., 1976. Structure of decision: The cognitive maps of political elites. Princeton university press.

Babbar, P., Yadav, K., Singhal, A., Sharma, V., 2018. Connectionist Model in Artificial Intelligence. Int. J. Appl. Eng. Res. 13, 5154–5159.

Bahit, E., 2012. Curso Python para principiantes. eugeniabahit. com.

Ballesteros, S., Kraft, E., Santana, S., Tziraki, C., 2015. Maintaining older brain functionality: a targeted review. Neurosci. Biobehav. Rev. 55, 453–477.

Benítez, R., Escudero, G., Kanaan, S., Rodó, D.M., 2014. Inteligencia artificial avanzada. Editorial UOC.

Braun, U., Muldoon, S.F., Bassett, D.S., 2015. On human brain networks in health and disease. eLS.

Chojaczyk, A.A., Teixeira, A.P., Neves, L.C., Cardoso, J.B., Guedes Soares, C., 2015. Review and application of Artificial Neural Networks models in reliability analysis of steel structures. Struct. Saf. 52, 78–89. https://doi.org/10.1016/j.strusafe.2014.09.002

Da Silva, I.N., Spatti, D.H., Flauzino, R.A., Liboni, L.H.B., dos Reis Alves, S.F., 2017. Artificial Neural Networks. Springer.

Demšar, J., Curk, T., Erjavec, A., Gorup, Č., Hočevar, T., Milutinovič, M., Možina, M., Polajnar, M., Toplak, M., Starič, A., 2013. Orange: data mining toolbox in Python. J. Mach. Learn. Res. 14, 2349–2353.

Díez, R.P., 2001. Introducción a la inteligencia artificial: sistemas expertos, redes neuronales artificiales y computación evolutiva. Universidad de Oviedo.

Doig, C., 2016. Embracing Open Data Science in your Organization.

Flores Villarreal, H.J., 2005. Operación eficiente de sistemas de transporte de gas natural mediante el método de gradiente reducido generalizado. Universidad Autónoma de Nuevo León.

Gardner, M.W., Dorling, S.R., 1998. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. Atmos. Environ. 32, 2627–2636.

Granados, L.F.M., 2017. Línea de inteligencia artificial y procesos de razonamiento. TED Tecné Episteme Didaxis.

Gutiérrez, J.F., Frías, T.E.R., Ledesma, M.Á.G., Pérez, N.V.R., Montalvo, A.F., 2016. SISTEMA BASADO EN INTELIGENCIA ARTIFICIAL PARA EL DIAGNÓSTICO SOBRE TIPOS DE CÓLICOS EN EQUINOS DESARROLLADO EN PROLOG. Pist. Educ. 38.

Hobbs, B.F., Ludsin, S.A., Knight, R.L., Ryan, P.A., Biberhofer, J., Ciborowski, J.J., 2002. Fuzzy cognitive mapping as a tool to define management objectives for complex ecosystems. Ecol. Appl. 12, 1548–1565.

Holmes, G., Donkin, A., Witten, I.H., 1994. Weka: A machine learning workbench, in: Intelligent Information Systems, 1994. Proceedings of the 1994 Second Australian and New Zealand Conference On. IEEE, pp. 357–361.

Izaurieta, F., Saavedra, C., 2000. Redes neuronales artificiales. Dep. Física Univ. Concepc. Chile.

Kapoor, A., Wiebe, N., Svore, K., 2016. Quantum perceptron models, in: Advances in Neural Information Processing Systems. pp. 3999–4007.

Kosko, B., 1986. Fuzzy cognitive maps. Int. J. Man-Mach. Stud. 24, 65–75.

Lek, S., Delacoste, M., Baran, P., Dimopoulos, I., Lauga, J., Aulagnier, S., 1996. Application of neural networks to modelling nonlinear relationships in ecology. Ecol. Model. 90, 39–52.

Nápoles, G., 2013. Learning stability features on sigmoid fuzzy cognitive maps through a swarm intelligence approach, in: Iberoamerican Congress on Pattern Recognition. Springer, pp. 270–277.

Nápoles, R., 2014. Algoritmo para mejorar la convergencia en Mapas Cognitivos Difusos Sigmoidales. Universidad Central "Marta Abreu" de Las Villas. Facultad de Matemática, Física y Computación. Departamento Ciencias de la Computación.

Nápoles Ruiz, G., Espinosa, M.L., Grau, I., Vanhoof, K., Bello, R., 2017. Fuzzy Cognitive Maps Based Models for Pattern Classification: Advances and Challenges.

Ogata, K., 2002. Modern control engineering. Prentice hall India.

Osoba, O.A., Kosko, B., 2017. Fuzzy cognitive maps of public support for insurgency and terrorism. J. Def. Model. Simul. 14, 17–32.

Özesmi, U., Özesmi, S.L., 2004. Ecological models based on people's knowledge: a multi-step fuzzy cognitive mapping approach. Ecol. Model. 176, 43–64.

Palacios, A., 2012. Implementación de un módulo para el entrenamiento y evaluación de redes neuronales mediante GPUs.

Papageorgiou, E.I., 2011. Fuzzy cognitive map based approach for predicting yield in cotton crop production as a basis for decision support system in precision agriculture application. Appl. Soft Comput. 11, 3643–3657.

Papageorgiou, E.I., 2010. Soft computing technique of fuzzy cognitive maps to connect yield defining parameters with yield in cotton crop production in central Greece as a basis for a decision support system for precision agriculture application, in: Fuzzy Cognitive Maps. Springer, pp. 325–362.

Papageorgiou, E.I., 2004. Active Hebbian learning algorithm to train fuzzy cognitive maps. Int. J. Approx. Reason. 37, 219–249.

Parsopoulos, K.E., 2003. A first study of fuzzy cognitive maps learning using particle swarm optimization, in: Evolutionary Computation, 2003. CEC'03. The 2003 Congress On. IEEE, pp. 1440–1447.

Pchelintseva, S.V., Runnova, A.E., Musatov, V.Y., Hramov, A.E., 2017. Recognition and classification of oscillatory patterns of electric brain activity using artificial neural network approach, in: Dynamics and Fluctuations in Biomedical Photonics XIV. International Society for Optics and Photonics, p. 1006317.

Redolar Ripoll, D., 2014. Neurociencia cognitiva. Editor. Panam. Madr. 5.

Russell, S., Dietterich, T., Horvitz, E., Selman, B., Rossi, F., Hassabis, D., Legg, S., Suleyman, M., George, D., Phoenix, S., 2015. Letter to the editor: Research priorities for robust and beneficial artificial intelligence: An open letter. AI Mag. 36, 3–4.

Shortliffe, E.H., 1974. A rule-based computer program for advising physicians regarding antimicrobial therapy selection, in: Proceedings of the 1974 Annual ACM Conference-Volume 2. ACM, pp. 739–739.

Stach, W., 2005. Genetic learning of fuzzy cognitive maps. Fuzzy Sets Syst. 153, 371–401.

Tang, J., Deng, C., Huang, G.-B., 2016. Extreme learning machine for multilayer perceptron. IEEE Trans. Neural Netw. Learn. Syst. 27, 809–821.

Trujillano, J., 2004. Aproximación metodológica al uso de redes neuronales artificiales para la predicción de resultados en medicina. Med. Clin. (Barc.) 122, 59–67.

Tulchak, L.V., Marchuk, A.O., 2016. History of python. ВНТУ.

Yadav, A., Sahu, K., 2017. WIND FORECASTING USING ARTIFICIAL NEURAL NETWORKS: A SURVEY AND TAXONOMY. Int. J. Res. Sci. Eng. 3.

Zhang, B.-T., 2016. Artificial intelligence and medicine. 춘· 추계 학술대회 KASL 2016, 65–65.

ANEXOS

Anexo I Uso de SLP y MLP para modelar las funciones lógicas AND y XOR

Se quiere modelar mediante una ANN las funciones lógicas AND y XOR (OR exclusivo). A continuación, se muestran las tablas de verdad de las funciones a modelar:

Tabla A.1. Función AND.

X_1	X_2	Y
0	0	0
0	1	0
1	0	0
1	1	1

Tabla A.2 Función XOR.

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

Para estos ejemplos se usará como función de activación la función paso. La capa de entrada posee dos neuronas, cada una de las cuales recibe los valores de X_1 y X_2, mientras que la capa de salida solo tiene una neurona que ofrece la salida Y. Primero se utilizará una red sin capas ocultas, la cual es la más sencilla posible (Figura A.1).

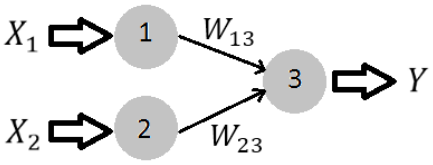

Figura A.1. ANN sin capas ocultas.

La salida de esta red está representada por la ecuación (A.1), donde u es el umbral de inhibición de las neuronas:

$$Y = \begin{cases} 0, & W_{13}X_1 + W_{23}X_2 < u \\ 1, & W_{13}X_1 + W_{23}X_2 \geq u \end{cases} \quad \text{(A.1)}$$

El espacio de entrada es el plano X_1-X_2 y la frontera entre los puntos de este plano con salida $Y = 0$ y $Y = 1$ es la recta $W_{13}X_1 + W_{23}X_2 = u$. Estas cuestiones se aprecian claramente en la Figura A.2:

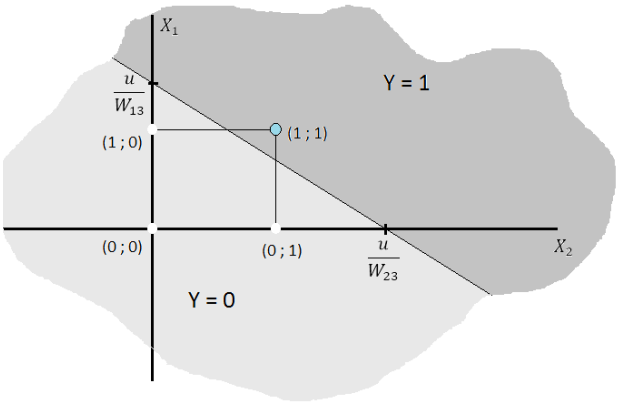

Figura A.2. Separabilidad lineal de la función AND.

Las salidas deseadas para las cuatro combinaciones del espacio de entrada se pueden separar fácilmente por esta recta.

Por otra parte, la función XOR no es tan sencilla. De las cuatro combinaciones, separar las que producen 0 como salida de las que producen 1, con solo una recta es imposible, tal y como se muestra en la Figura A.3

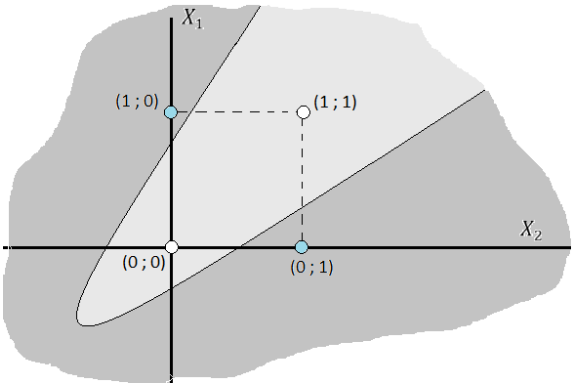

Figura A.3. Curva hipotética que puede separar la función XOR.

Una posible solución sería utilizar dos rectas como se muestra en la Figura A.4. Esto se logra con una configuración que contiene una capa oculta, cuya salida está representada por estas rectas. Esta configuración se muestra en la Figura A.5.

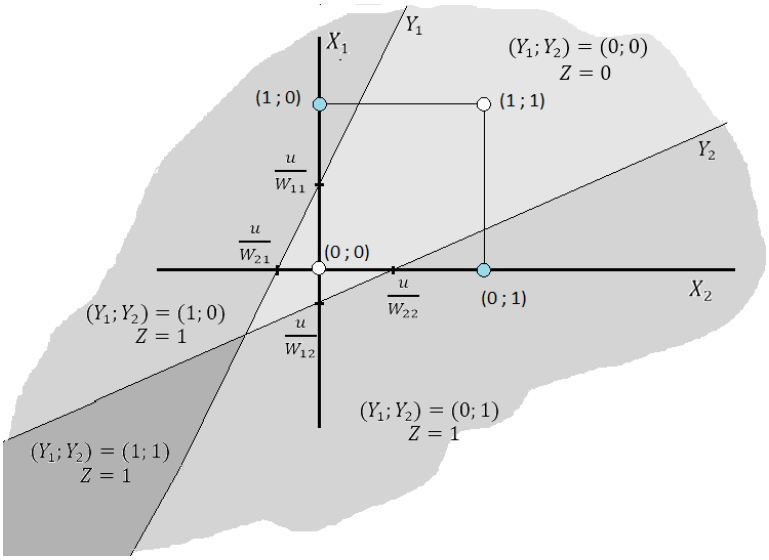

Figura A.4. Separabilidad de segundo orden de la función XOR.

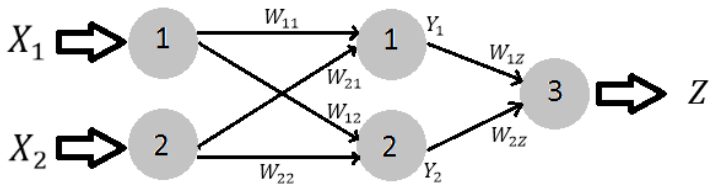

Figura A.5. Estructura de ANN de segundo orden.

De este modo las salidas intermedias Y_1 y Y_2 hacen a su vez de entradas de la única neurona en la capa más externa. Como se aprecia en la Figura A.4 la salida Z no es más que el resultado de aplicar la función lógica OR a Y_1 y Y_2. Al igual que la función AND, la función OR posee un espacio de salida en el plano X_1-X_2 que es perfectamente separable por una línea recta (el lector puede comprobarlo por sí mismo), que en este caso estaría dada por la ecuación (A.2):

$$Z = \begin{cases} 0, & W_{1z}Y_1 + W_{2z}Y_2 < u \\ 1, & W_{1z}Y_1 + W_{2z}Y_2 \geq u \end{cases} \tag{A.2}$$

Puede concluirse entonces que una ANN puede modelar cualquier función lógica por compleja que esta sea. Es válido destacar que en caso de tener más de dos entradas el plano que se ha usado hasta ahora se convierte en un híper-plano de N dimensiones (siendo N el número de entradas) y las posibles salidas se separan por otro híper-plano de N-1 dimensiones. Existen métodos matemáticos capaces de determinar topologías adecuadas para modelar determinada función, pero son temas que se salen del marco de esta investigación.

Anexo II Matriz Hessiana

Dada una función definida $f: \mathbb{R}^n \rightarrow \mathbb{R}$ si existen todas las segundas derivadas parciales de $f(\boldsymbol{x})$ se define su Matriz Hessiana como (A.3):

$$H_f(\boldsymbol{x})_{i,j} = \frac{\partial^2 f(\boldsymbol{x})}{\partial x_i \partial x_j} \tag{A.3}$$

Anexo III Cubo de Necker

El cubo Necker es una ilusión óptica representada por una figura imposible de realizar físicamente. En la Figura A.6, se muestra una imagen del mismo.

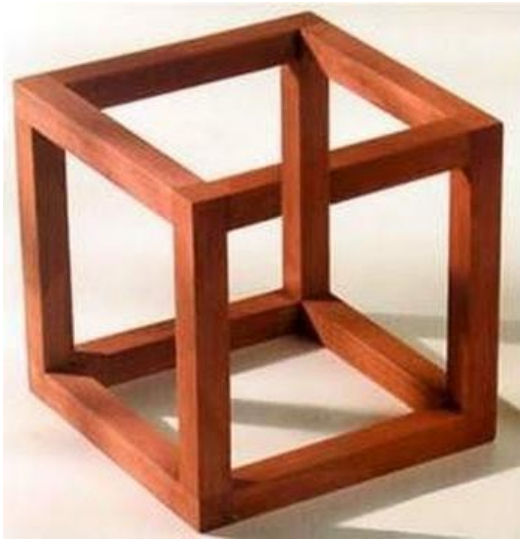

Figura A.6. Cubo de Necker.

Anexo IV Resultados del experimento con el circuito RLC

A continuación, se muestran las gráficas resultantes del modelo antes y después del proceso de aprendizaje en los tres sets de datos

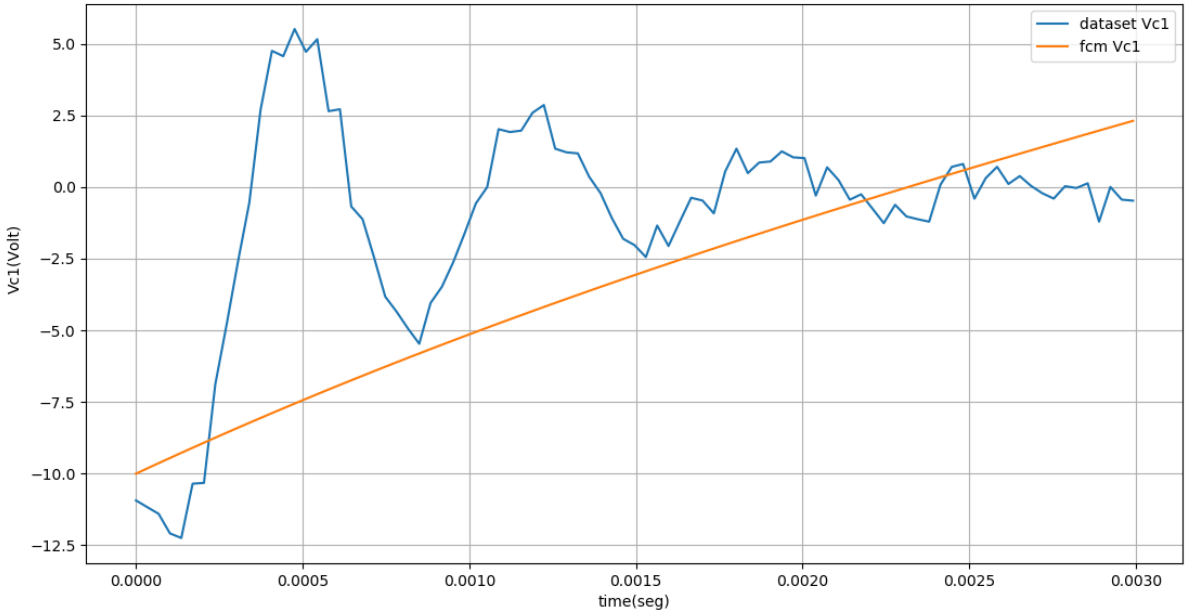

Figura A.7. Voltaje en el capacitor 1 antes del aprendizaje (*dataset* 8).

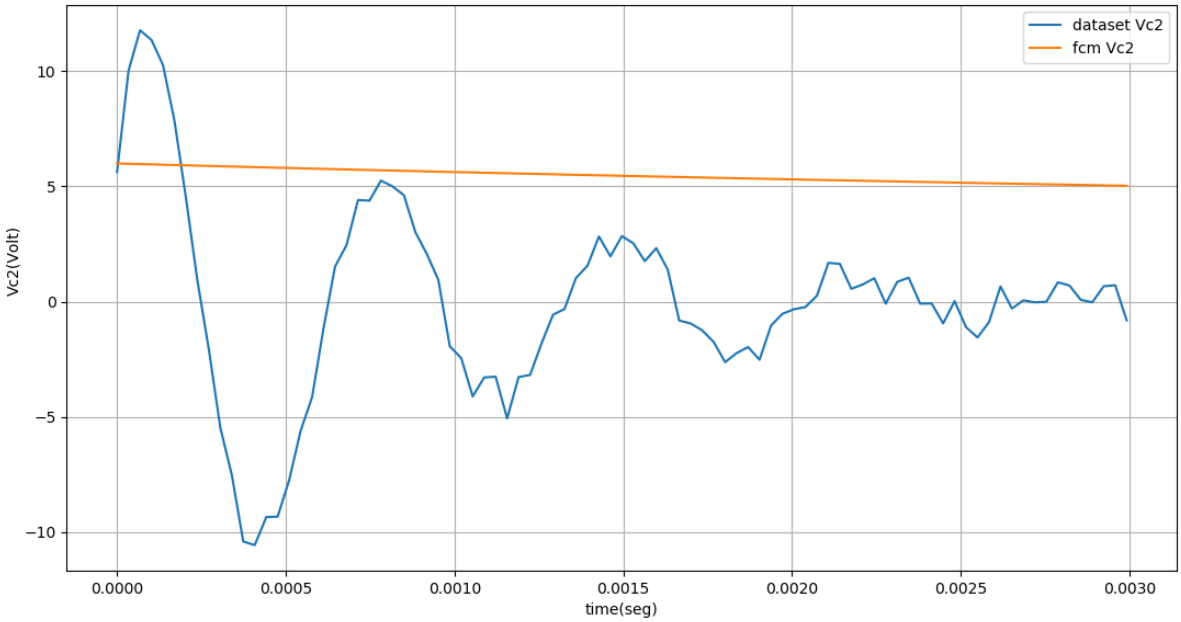

Figura A.8. Voltaje en el capacitor 2 antes del aprendizaje (*dataset* 8).

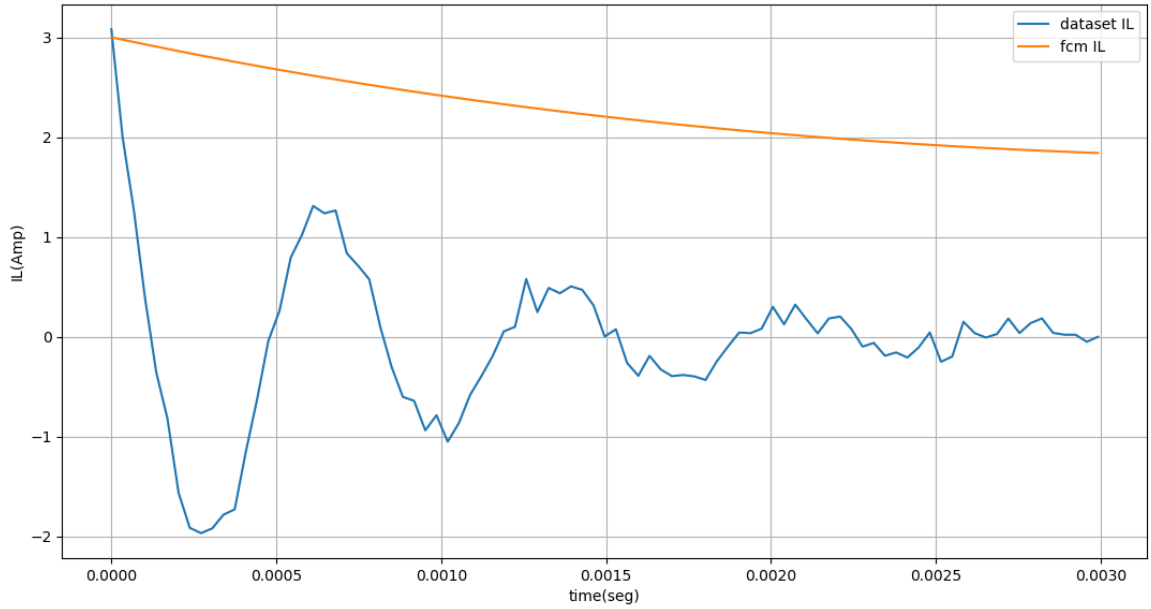

Figura A.9. Corriente en el inductor antes del aprendizaje (*dataset* 8).

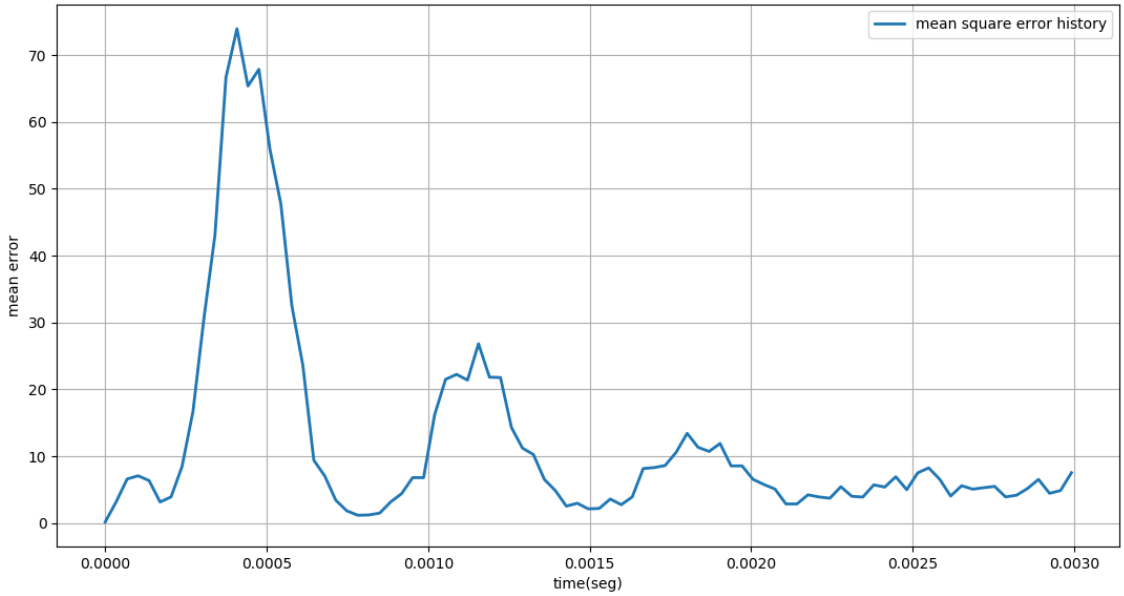

Figura A.10. Error cuadrático medio antes del aprendizaje (*dataset* 8).

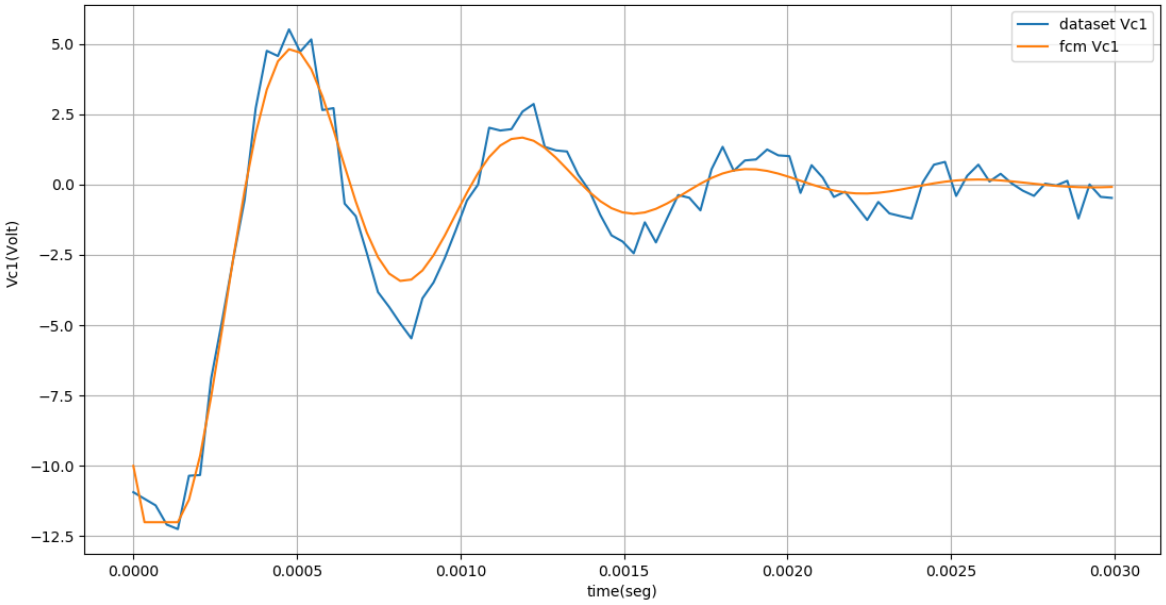

Figura A.11. Voltaje en el capacitor 1 después del aprendizaje (*dataset* 8).

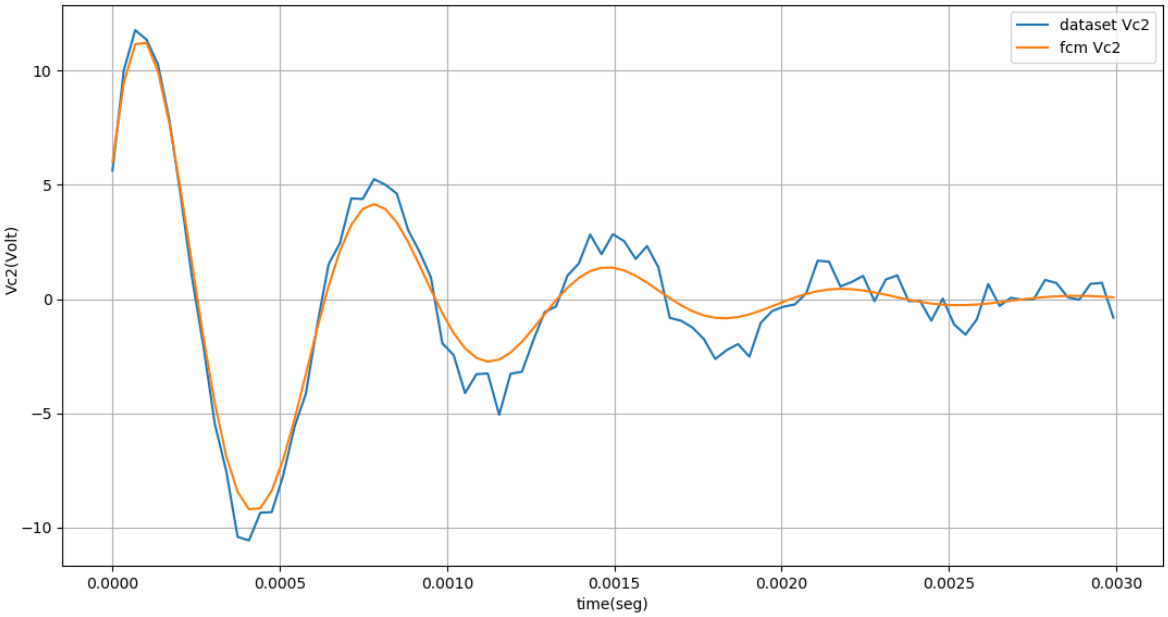

Figura A.12. Voltaje en el capacitor 2 después del aprendizaje (*dataset* 8).

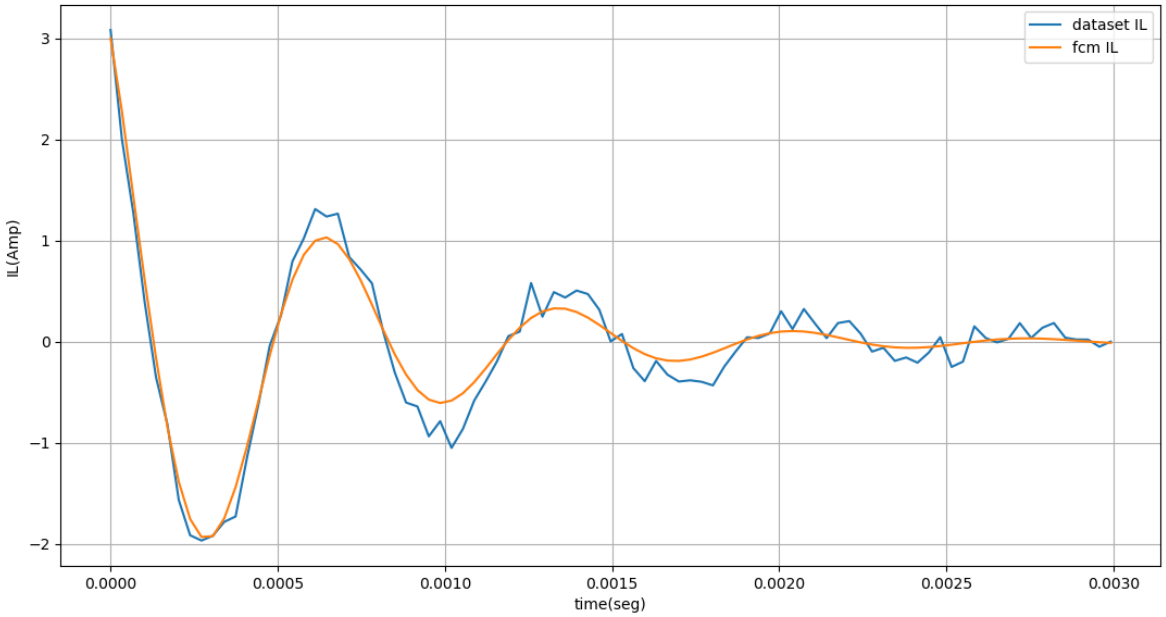

Figura A.13. Corriente en el inductor después del aprendizaje (*dataset* 8).

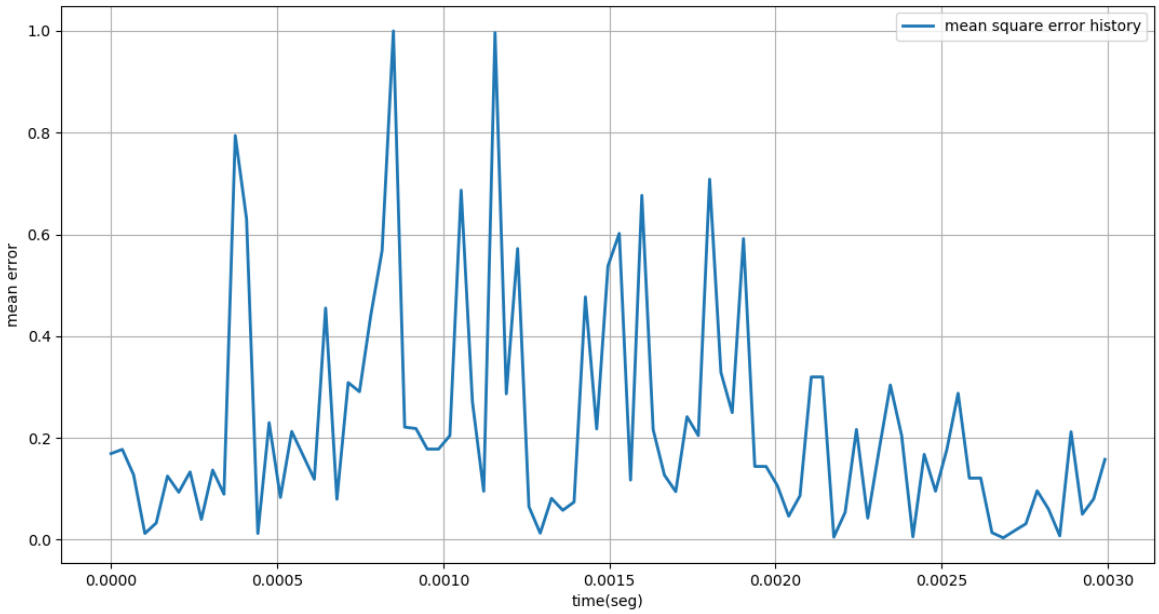

Figura A.14. Error cuadrático medio después del aprendizaje (*dataset* 8).

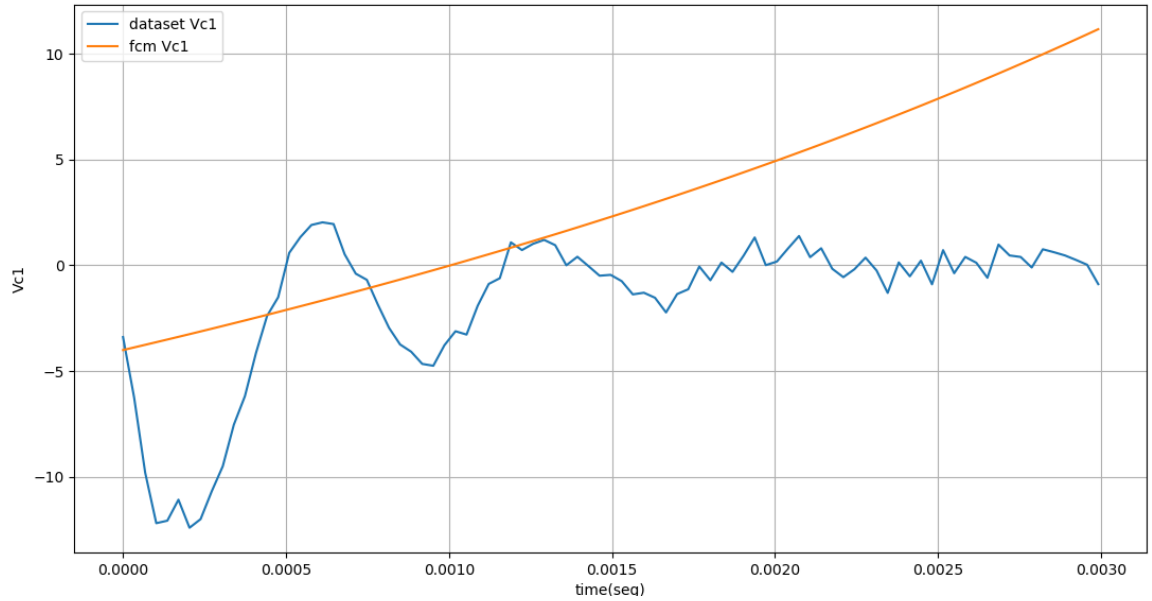

Figura A.15. Voltaje en el capacitor 1 antes del aprendizaje (*dataset* 9).

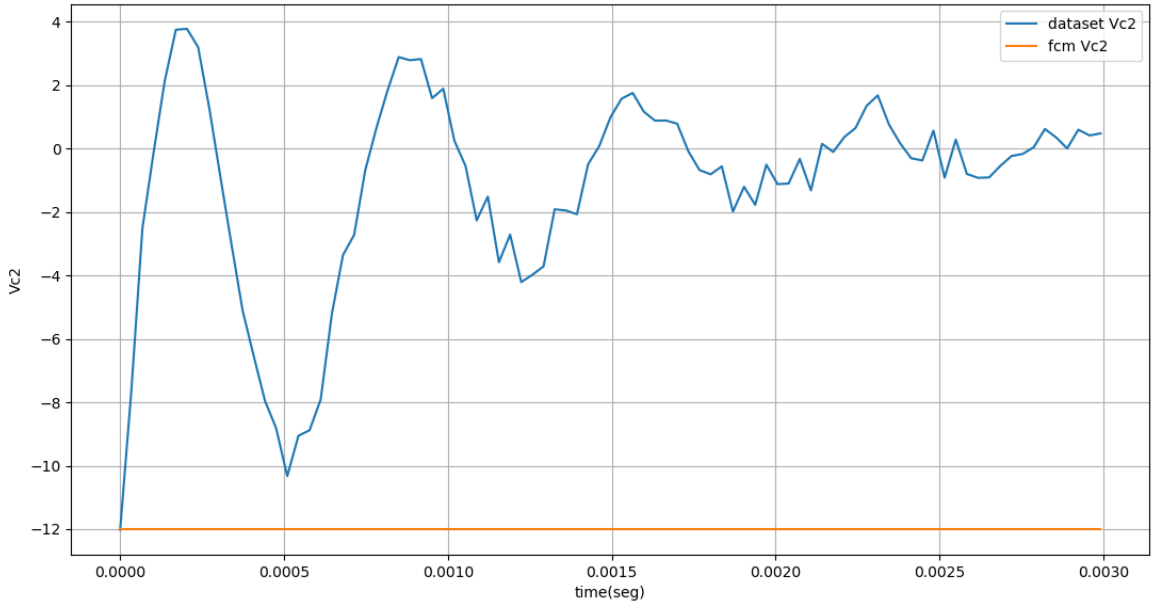

Figura A.16. Voltaje en el capacitor 2 antes del aprendizaje (*dataset* 9).

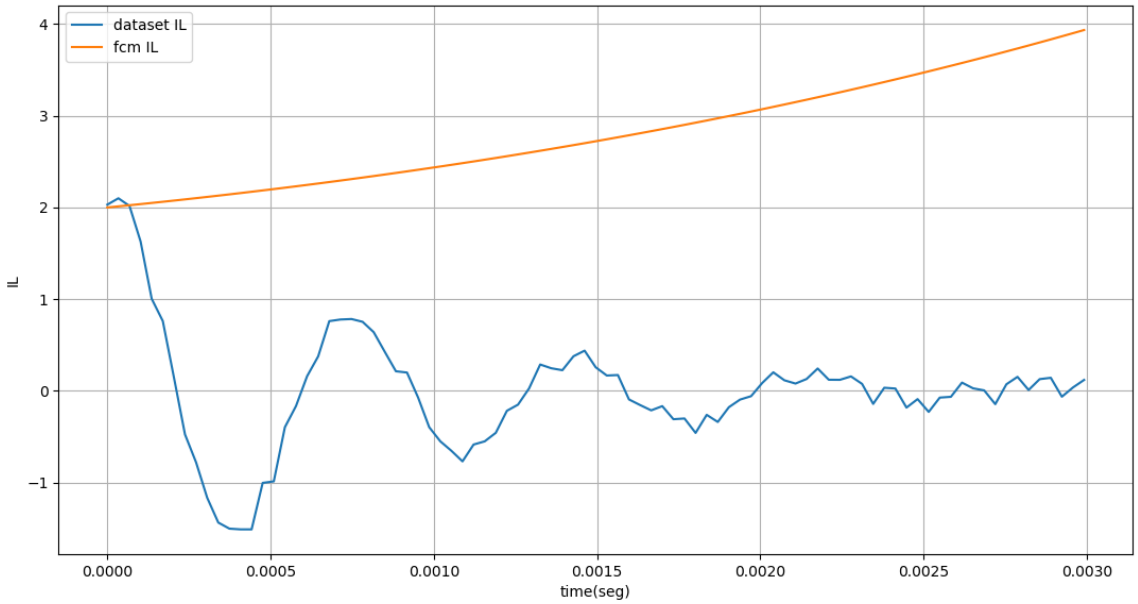

Figura A.17. Corriente en el inductor antes del aprendizaje (*dataset* 9).

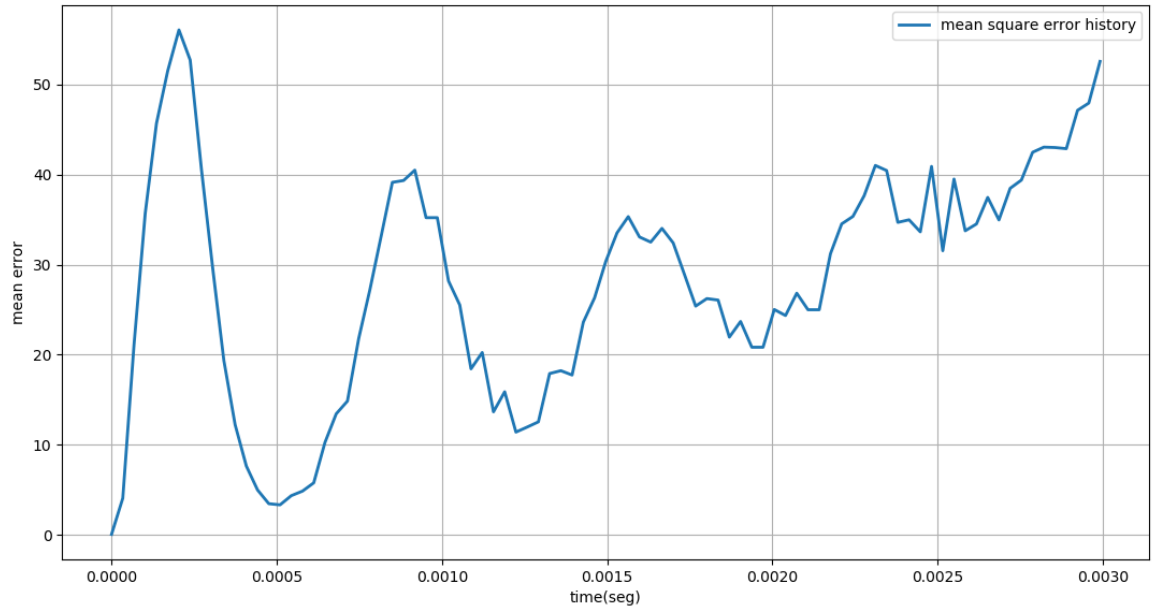

Figura A.18. Error cuadrático medio antes del aprendizaje (*dataset* 9).

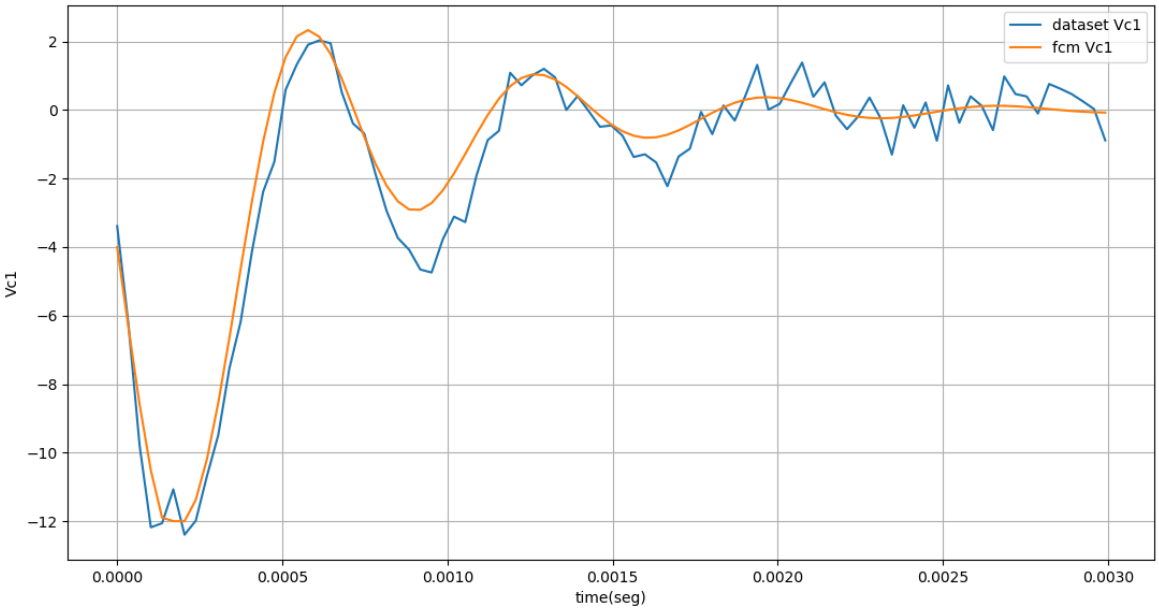

Figura A.19. Voltaje en el capacitor 1 después del aprendizaje (*dataset* 9).

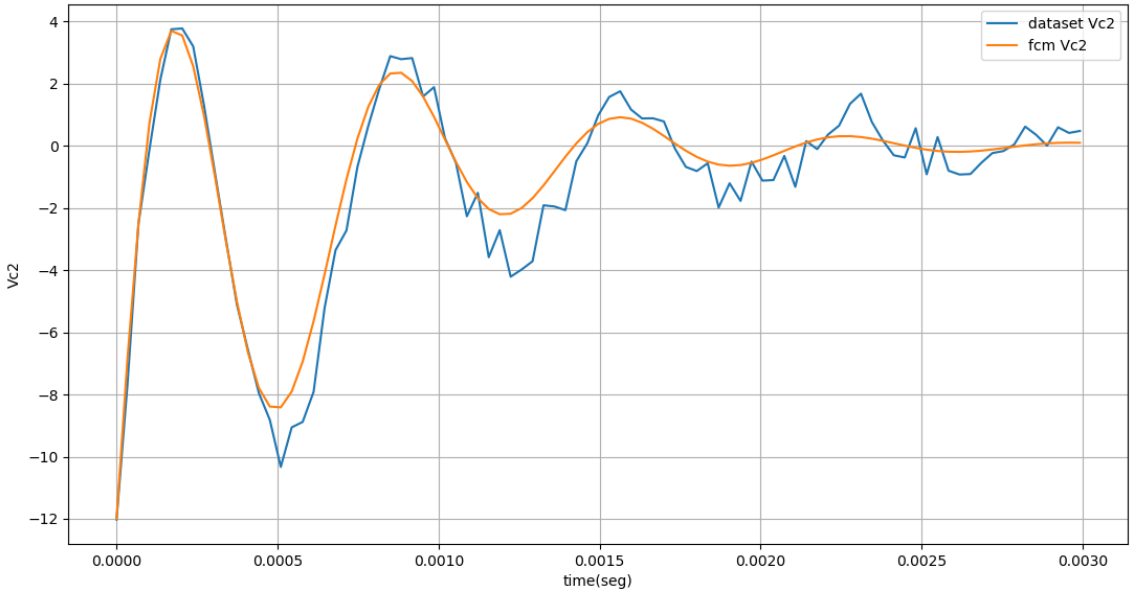

Figura A.20. Voltaje en el capacitor 2 después del aprendizaje (*dataset* 9).

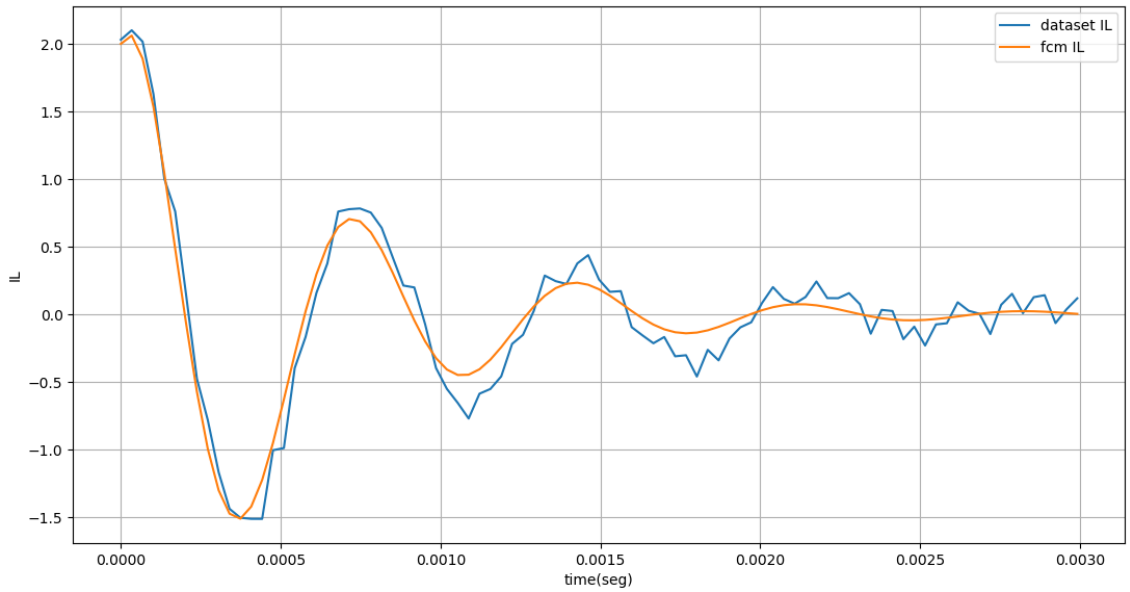

Figura A.21. Corriente en el inductor después del aprendizaje (*dataset* 9).

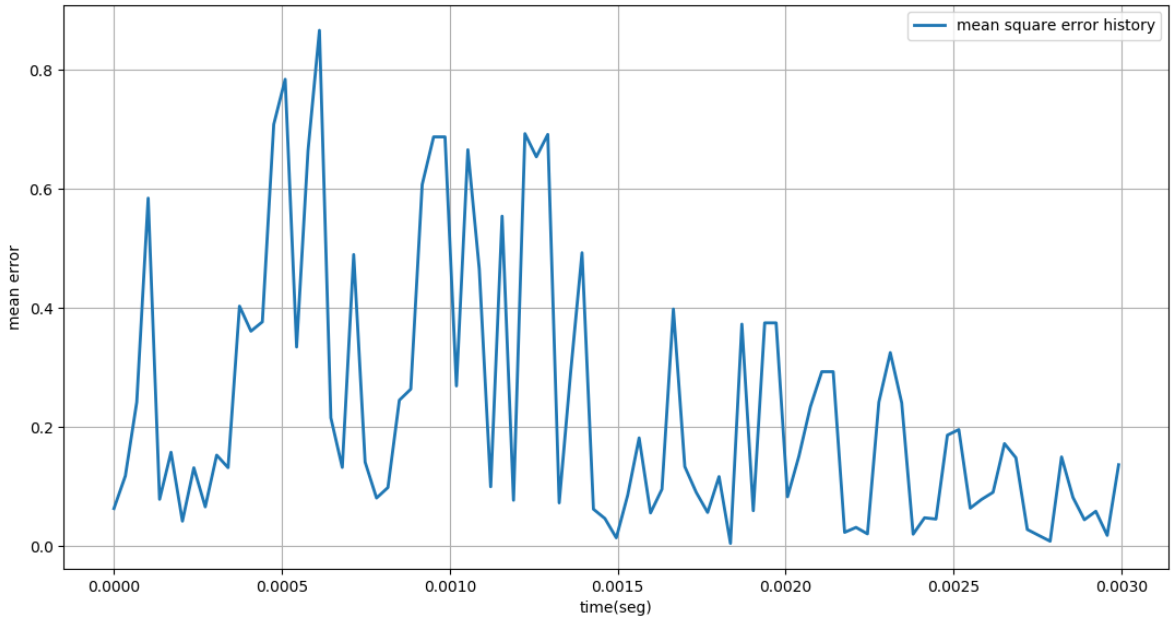

Figura A.22. Error cuadrático medio después del aprendizaje (*dataset* 9).

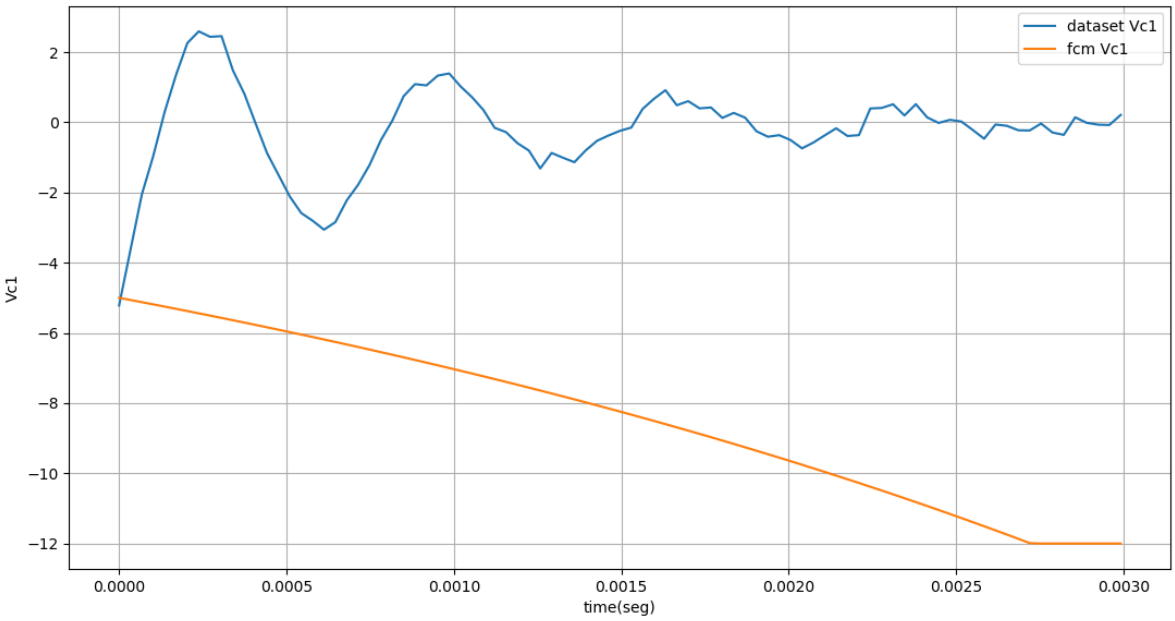

Figura A.23. Voltaje en el capacitor 1 antes del aprendizaje (*dataset* 10).

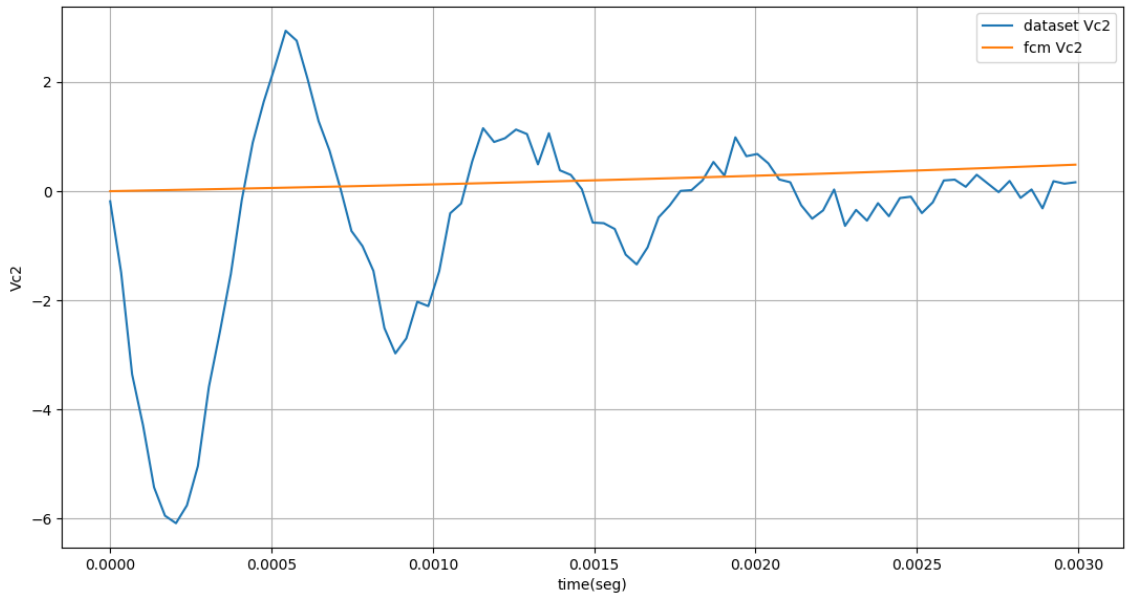

Figura A.24. Voltaje en el capacitor 2 antes del aprendizaje (*dataset* 10).

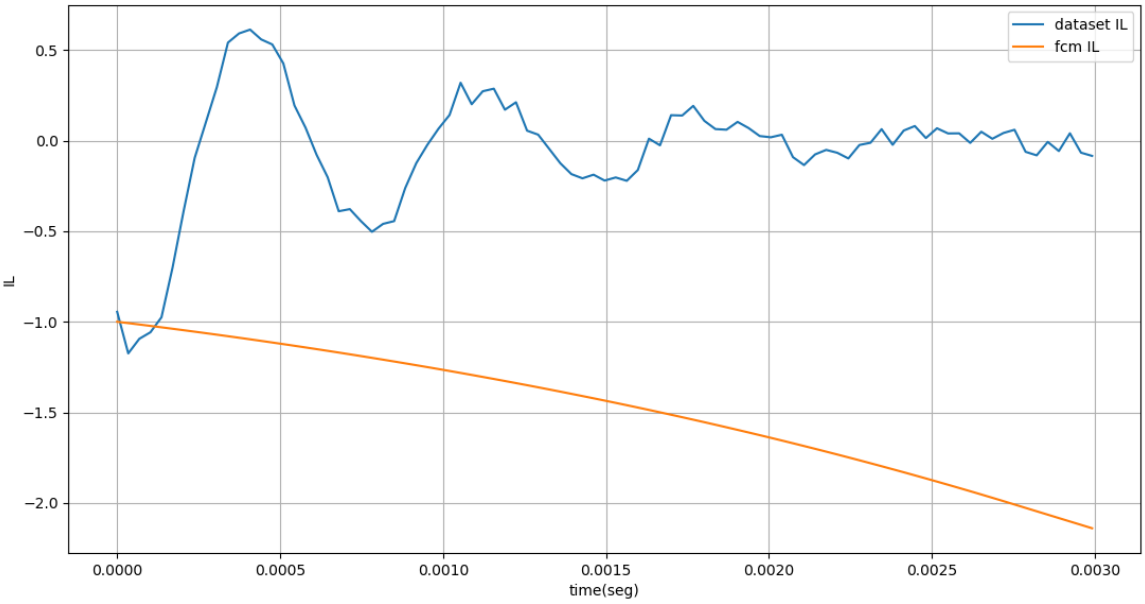

Figura A.25. Corriente en el inductor antes del aprendizaje (*dataset* 10).

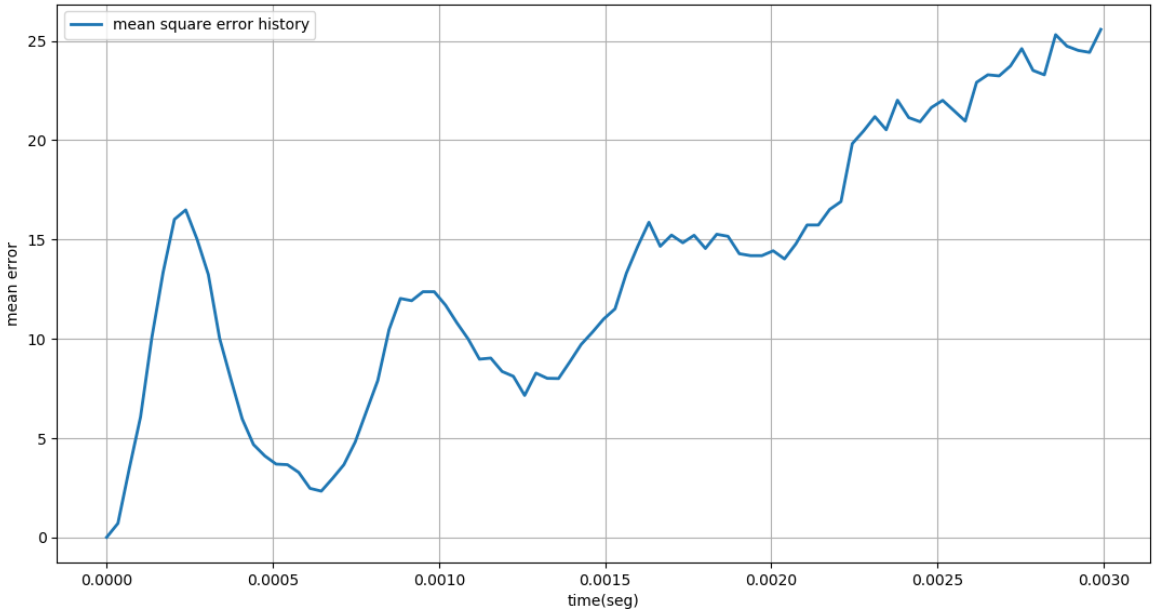

Figura A.26. Error cuadrático medio antes del aprendizaje (*dataset* 10).

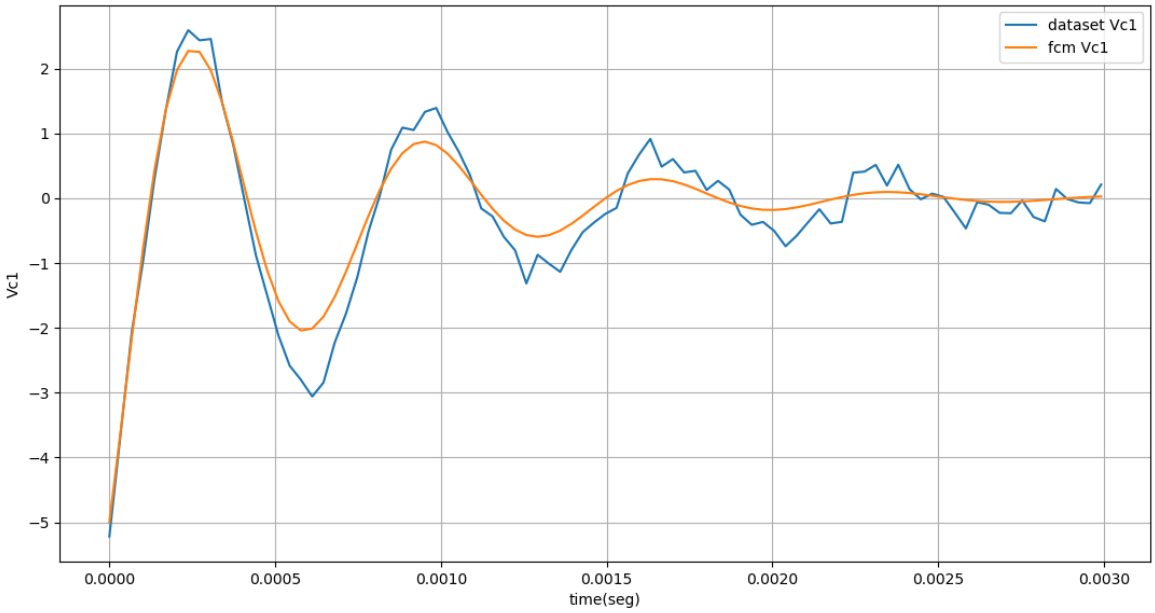

Figura A.27. Voltaje en el capacitor 1 después del aprendizaje (*dataset* 10).

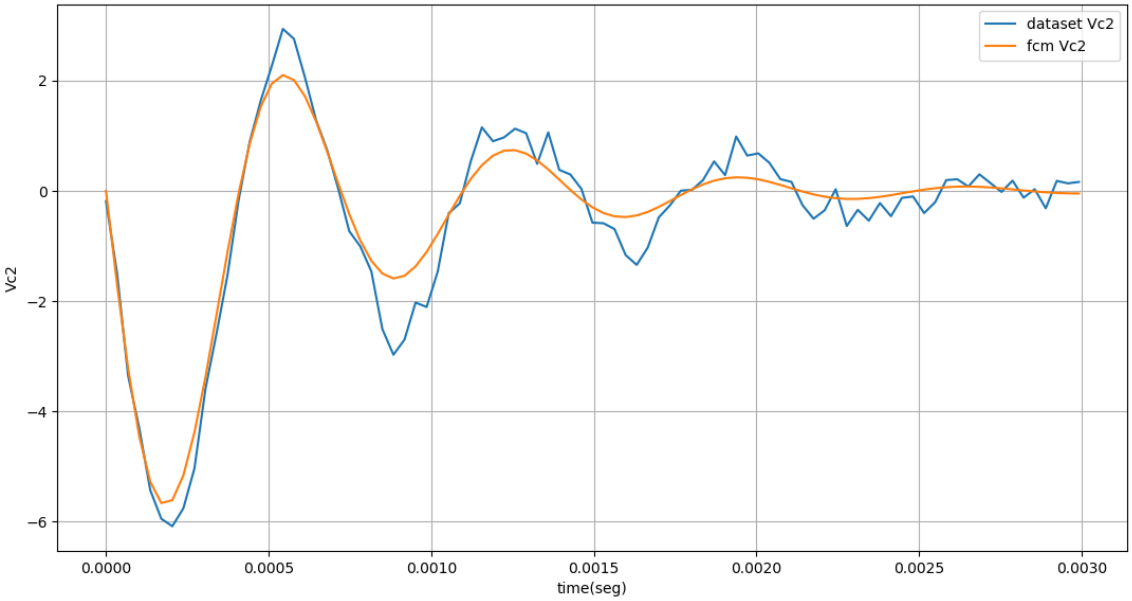

Figura A.28. Voltaje en el capacitor 2 después del aprendizaje (*dataset* 10).

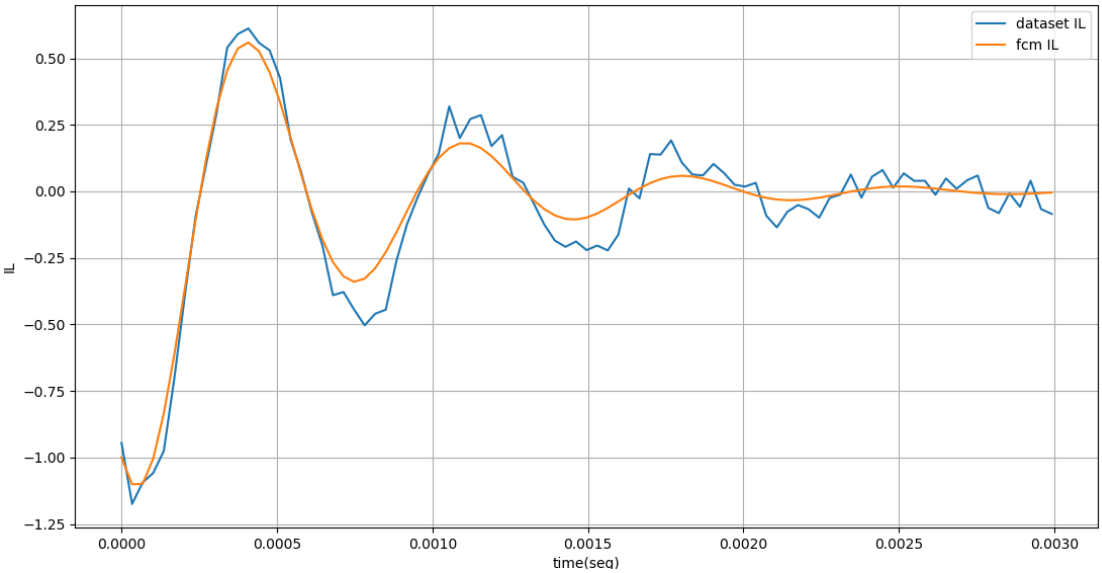

Figura A.29. Corriente en el inductor después del aprendizaje (*dataset* 10).

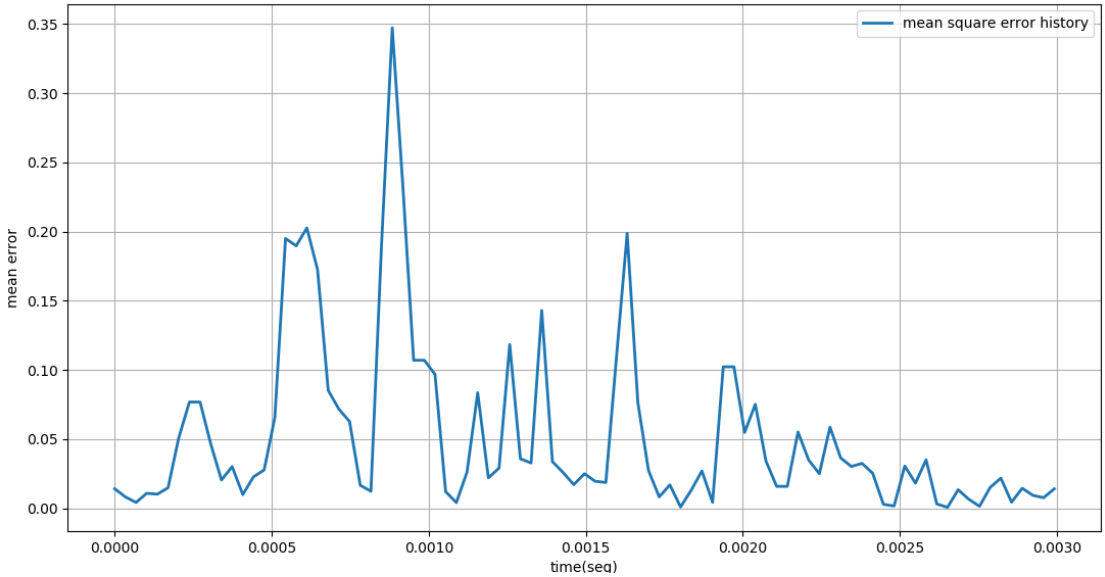

Figura A.30. Error cuadrático medio después del aprendizaje. (*dataset* 10).