Trabajo Fin de Grado

Grado en Ingeniería Electrónica, Robótica y Mecatrónica.

Aplicación de técnicas de inteligencia artificial (AI) al diseño de un algoritmo de control adaptativo de un convertidor de potencia DC/DC bidireccional resonante de alta frecuencia.

Autor: Pedro Rodríguez Rodas

Tutor: Eduardo Galván Díez

Juan Manuel Carrasco Solís

Dpto. de Ingeniería electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

ANDALUCÍA TECH
Campus de Excelencia Internacional

Trabajo Fin de Gado
Grado en Ingeniería Electrónica, Robótica y Mecatrónica.

Aplicación de técnicas de inteligencia artificial (AI) al diseño de un algoritmo de control adaptativo de un convertidor de potencia DC/DC bidireccional resonante de alta frecuencia.

Autor:
Pedro Rodríguez Rodas

Tutor:
Eduardo Galván Díez
Juan Manuel Carrasco Solís
Catedráticos de Universidad

Dpto. de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Trabajo Fin de Grado: Aplicación de técnicas de inteligencia artificial (AI) al diseño de un algoritmo de control adaptativo de un convertidor de potencia DC/DC bidireccional resonante de alta frecuencia.

Autor: Pedro Rodríguez Rodas

Tutores: Eduardo Galván Díez

Juan Manuel Carrasco Solís

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2020

El Secretario del Tribunal

Quien se cree poseedor de la verdad, no avanza

A mi familia

A todo aquel que aportó en este largo y duro camino

Agradecimientos

Aquellas personas que me conocen de cerca saben el esfuerzo que me ha supuesto este trabajo, que, aunque llega algo tarde, siempre se ha dicho que lo lento gana en experiencia.

Este camino al que denominamos “grado” ha sido duro y a veces daba la sensación de inacabable. Ha estado lleno de éxitos y fracasos, y justamente a partir de éxitos y fracasos crecen las personas. “A base de palos” como decimos, se hacen los ingenieros, ¡y palos no han faltado!

Quiero agradecer en primer lugar a mis padres todo lo que han aportado para que pueda estar aquí, acabando ya. Sin ellos no solo hubiese sido aún más duro, si no que posiblemente no estaría escribiendo esto.

Agradecer también al resto de mi familia y mis amigos por el apoyo que me han dado en estos años, ojalá algún día yo pueda tener la infinita paciencia que han tenido conmigo.

No olvidar a aquellas personas que me han acompañado y que de una manera u otra me han regalado experiencias inolvidables, ayuda y conocimientos. ¡La lista de agradecimientos sería enorme, por ello mencionar a cada uno de ellos y olvidarme de alguno no sería justo!

Gracias, se acaba un camino, empieza otro.

Pedro Rodríguez Rodas

Alumno en la Escuela Técnica Superior de Ingeniería de Sevilla

Sevilla-Montijo, 2020

Resumen

Según la origen de la electricidad avanza a ser totalmente renovable, día a día, la electrónica de potencia gana importancia en este campo, pues cualquier instalación que se precie (eólica, fotovoltaica…) necesita de dispositivos de potencia para un funcionamiento óptimo. Por ello, se investigan soluciones más precisas y con un rendimiento mayor, ahí aparecen convertidores como los DAB (Dual Active Bridge), que, mediante el uso de un PWM de alta frecuencia, mejoran el rendimiento y reducen el tamaño respecto a los dispositivos y técnicas de modulación clásicas, además de abaratar el coste. Por otro lado, estos dispositivos necesitan de sistemas de control lo suficientemente adecuados y preparados para soportar las situaciones que puedan implicar cambios en las condiciones del control. Estas dos premisas nos llevan a tener en cuenta el uso de redes neuronales y sistemas fuzzy logic (lógica difusa), dando lugar a las estructuras "Neurofuzzy" o "Neural-fuzzy". Estructuras que hasta ahora demuestran gran flexibilidad y sencillez en su diseño e implementación frente a otros controladores hasta ahora usados.

A continuación, en el trabajo aquí expuesto, se intentan aunar estos conceptos, dando lugar a una solución adecuada.

Abstract

Depending on the origin of the electricity it becomes fully renewable, day by day, power electronics gain importance in this field, since any installation worth its salt (wind, photovoltaic ...) needs power devices for optimal operation. For this reason, more precise solutions with higher performance are being investigated, there appear converters such as DAB (Dual Active Bridge), which, through the use of a high frequency PWM, improve performance and reduce the size compared to devices and classical modulation techniques, in addition to lowering the cost. On the other hand, these devices need control systems that are sufficiently adequate and prepared to withstand the situations that may involve changes in the control conditions. These two premises lead us to take into account the use of neural networks and fuzzy logic systems, giving rise to the "Neurofuzzy" or "Neural-fuzzy" structures. Structures that until now show great flexibility and simplicity in their design and implementation compared to other controllers used up to now.

Next, in the work presented here, an attempt is made to combine these concepts, leading to an adequate solution.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1. PROPÓSITO DEL PROYECTO

No hay nada como un reto para sacar lo mejor de un hombre.

- Sean Connery, cineasta escocés-

EN el capítulo que nos ocupa, se tratará de poner al lector en contexto para afrontar la comprensión del proyecto completo, así como de resumir el documento que aquí se inicia.

1.1. Motivaciones para realizar el Proyecto

Debido al aumento en la demanda de sistemas que puedan gestionar alta potencia y cuya eficiencia sea cada vez mayor, sobretodo teniendo en cuenta las soluciones orientadas a las energias renovables, mas concretamente el autoconsumo fotovoltaico, aparece la necesidad de disponer de convertidores de potencia que puedan dar respuesta a dicha demanda.

Por otro lado dichas topologias necesitan estrategias de control adecuadas que puedan conseguir una mayor eficiencia, pero que además puedan reducir los costes de la misma (Menor filtrado, tamaño…)

1.2. Propuesta de proyecto

A partir de la motivación, se genera una propuesta y por tanto fijaremos unos objetivos.

1.2.1. Propuesta para el convertidor

Partiendo de la necesidad de utilizar una topología de potencia que tenga cabida en la situación actual de auge del autoconsumo fotovoltaico, queremos realizar un Dual Active Bridge (DAB por sus siglas) con la idea de que pueda gestionar la potencia entre un bus de corriente continua (Podría tener como fuente la salida de una planta fotovoltaica pequeña) y un grupo de baterías.

En el segundo capítulo de este proyecto se explicará su funcionamiento y toda la base teórica de dicha topología.

1.2.2. Propuesta para la estrategia de control

Dicho convertidor, como veremos más adelante tiene como parámetro principal el desfase aplicado entre las ondas cuadradas que reciben cada uno de los puentes. Por tanto, tendríamos dos PWMs y sus complementarios, cuyo ancho y frecuencia son fijos, pero que entre puentes están desfasados de forma que, dependiendo de la magnitud de dicho desfase y su signo, la transferencia de potencia es mayor o menor y/o se realiza de un lado a otro (Boost/Buck)

1.3. Objetivos

Con estos antecedentes que hemos descrito, hemos fijado unos objetivos, cuyos resultados y conclusiones se analizarán a lo largo del documento.

El objetivo inicialmente es una vez analizada la topología de convertidor propuesta y la/s estructura/as de control elegidas, simular cada uno de los sistemas en MATLAB SIMULINK, para luego integrarlos, de forma detallada:

1) Simulación de la topología DAB estudiada.
2) Control mediante PI clásico de dicha topología.
3) Aprendizaje mediante una estructura Neurofuzzy de la función inversa asociada al convertidor.
4) Control del DAB aplicando el aprendizaje realizado.
5) Implementar en un DSP dicho control.

De forma opcional ya que para la simulación nos hemos basado en un convertidor disponible en laboratorio, en un trabajo más extenso en definitiva podrían hacerse pruebas con el DSP para ejecutar el control propuesto en laboratorio con el objetivo de arrancar el montaje del que se dispone.

1.4. Estructura del documento

En las paginas que vendrán a continuación se expondrá toda la teoria asociada tanto al DAB así como las redes neuronales, la logica difuse. Se analizará la implementación de las simulaciones realizadas y sus resultados, asi como el alcance del Proyecto y las posteriors ampliaciones. Acabaremos con las conclusiones.

2. TOPOLOGÍA DUAL ACTIVE BRIDGE (DAB)

El mejor comienzo es el peor final...

- ¿? -

EN el presente capitulo vamos a analizar la topología implementada en el proyecto desde su base teórica hasta aquello que nos concierne para nuestro trabajo.

2.1. Introducción a los DAB

Para nuestro proyecto se propone utilizar lo que se conoce más técnicamente como un Dual Active Bridge, que no es nada mas que un convertidor dc/dc bidireccional con un puente completo de transistores a cada lado de un transformador de alta frecuencia.

La principal ventaja de este tipo de montaje es la posibilidad de hacer "Soft-Switching", es decir, la conmutación se hace bajo condiciones favorables tal que se alcance lo que se denomina ZVZCS(Zero Voltage Zero Current Swtiching). Esto se traduce en pérdidas por conmutación prácticamente cero, menor ruido electromagnetico y por tanto pérmite una alta frecuencia de conmutación.

Esto ocurre ya que antes de que se aplique tension en la Puerta del transistor, el voltaje de conmutación del mismo se lleva a 0. El concepto es idoneo para esquemas que utilicen MOSFETs y una frecuencia de conmutación bastante alta (>20kHz).

Los DAB en los últimos años han aumentado su protagonismo en las aplicaciones de alta potencia, pues aunan distintas ventajas que son:

- Peso reducido
- Poco ruido sin comprometer la eficiencia
- Alta densidad de energia
- Menor coste vs Potencia de diseño respect a otros montajes

Las aplicaciones más frecuentes actualmente se centran en las fuentes renovables de energia, concretamente en las gestion y distribución de la electricidad. Nuestra idea con este proyecto se orienta justamente como posible aplicación a tener una instalación fotovoltaica conectada a nuestro DAB, teniendo la posibilidad de gestionar la carga de baterias que puedan almacenar la energia sobrante, existen multitud de posibilidades.

2.2. Análisis matemático de la topologia utilizada

El convertidor que analizaremos en este capitulo se corresponde con un Dual Active Bridge monofásico dc/dc tipo "*soft-switched*", idoneo para aplicaciones donde la alta densidad de potencia es importante.

La modulación utilizada para el disparo de los transistores se realizamediante un PWM[1]. <

La topologia que queremos implementar en nuestras simulaciones se trata de un DAB como el que podemos observar en la figura 2-1[2].

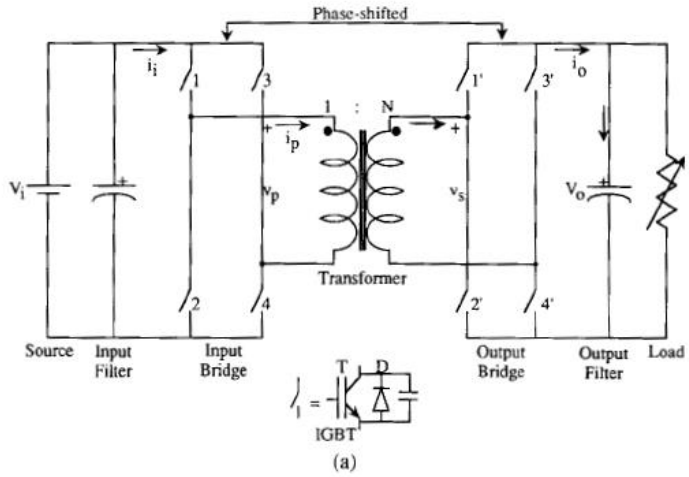


Figura 2-1. Convertidor DAB dc/dc monofásico. Fuente: [1]

Cuyo circuito simplicado asumiendo que

1) La frecuencia es suficientemente alta como para asumir que la transición entre conmutaciones es rapida.
2) Se sustituye el transformador por una inductancia de "fuga" o Leackage, que resulta de la imperfección que existe en el modelo electromagnetico del transformador.
3) El desfase entre puentes es independiente de la bobina antes mencionada y de la carga.
4) La intensidad de la bobina es una función de la variable $\theta = \omega t$ con ω siendo la frecuencia de conmutación en radianes.

Es:

[1] Del inglés, modulación por ancho de pulsos.
[2] Nos hemos basado en la referencia a bibliografía [1]

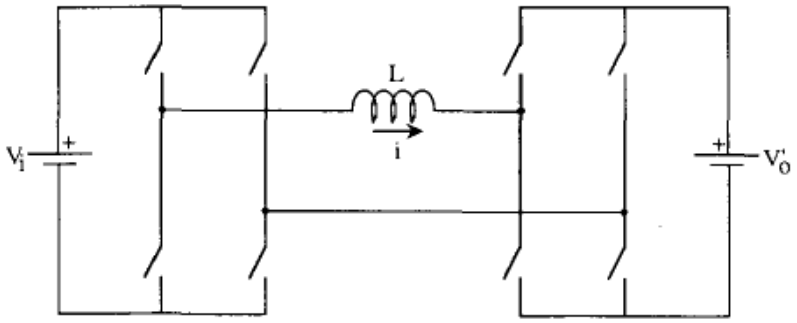


Figura 2-2. Circuito equivalente referido al primario [1].

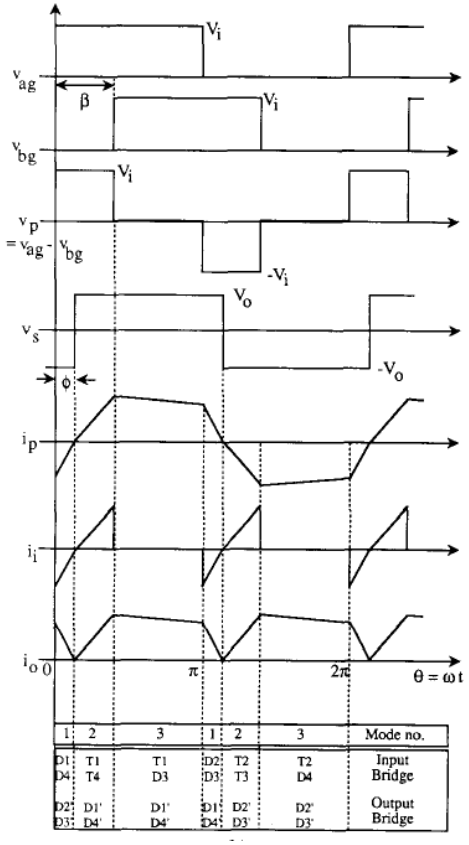


Figura 2-3. Formas de onda del convertidor en funcionamiento [1].

Tomando como referencia la Figura 2-3:

$$i(\theta) = \left[\frac{V_i + V_o'}{\omega L}\right]\theta + i(0) \qquad (2.1)$$

$$i(\theta)=\left[\frac{V_i-V_o^{'}}{\omega L}\right](\theta-\phi)+i(\phi) \qquad (2.2)$$

Las expresiones 2.1 y 2.2, determinan la intensidad en el modo 1 y 2 de funcionamiento respectivamente.

Sabemos que el modo 1 termina cuando $\theta=\phi$ y que al final de cada ciclo debe cumplirse la condición limite $i(\pi)=-i(0)$. Aplicando en ambas expresiones de la intensidad y despejando $i(0)$ e $i(\phi)$ tenemos que:

$$i(0)=\frac{(V_o^{'}-V_i)\pi-2V_o^{'}\phi}{2\omega L} \qquad (2.3)$$

$$i(\phi)=\frac{(V_o^{'}-V_i)\pi+2V_i\phi}{2\omega L} \qquad (2.4)$$

$$\begin{aligned} i(0)&\le 0\\ i(\phi)&\ge 0 \end{aligned} \qquad (2.5)$$

$$d=\frac{V_o^{'}}{V_i} \qquad (2.6)$$

Ahora teniendo en cuenta las condiciones de "Soft-Switching" mostradas en (2.5), es decir, lo que limita las región de operación para el convertidor, nos quedan dos expresiones dependientes de ϕ que despejando y sustituyendo las tensiones de entrada y salida aplicando (2.6) (que es la relación de voltaje del primario al secundario) nos da lo que hemos denominado desfase del puente de entrada y desfase del puente de salida, (Expresión 2.7) :

$$\begin{aligned} \phi_{out}&=\frac{(1-d)\pi}{2}\\ \phi_{in}&=\frac{(d-1)\pi}{2d} \end{aligned} \qquad (2.7)$$

Para obtener la Potencia que transfiere el convertidor de un lado a otro, calculamos la Intensidad media por la bobina (2.8) y multiplicamos por Vi:

$$i_{AV}=\frac{1}{\pi}\left(\int_0^{\phi}\left(\left[\frac{V_i+V_o^{'}}{\omega L}\right]\theta+i(0)\right)+\int_{\phi}^{\theta}\left(\left[\frac{V_i-V_o^{'}}{\omega L}\right](\theta-\phi)+i(\phi)\right)\right) \qquad (2.8)\rightarrow$$

$$\rightarrow i_{AV} = \frac{V_i}{\omega L} d\phi \left[1 - \frac{\phi}{\pi} \right] \qquad (2.9)$$

La potencia a la salida, que depende de d, v_i y ϕ como parámetros variables es:

$$P_o = \frac{V_i^2}{\omega L} d\phi \left[1 - \frac{\phi}{\pi} \right] \qquad (2.10)$$

Si dividimos la expresión anterior de la potencia por $\frac{V_i^2}{\omega L}$, obtenemos la potencia normalizada (2.11):

$$P_{oN} = d\phi \left[1 - \frac{\phi}{\pi} \right] \qquad (2.11)$$

Expresión que sólo depende del desfase entre puentes y la relación de transformación.

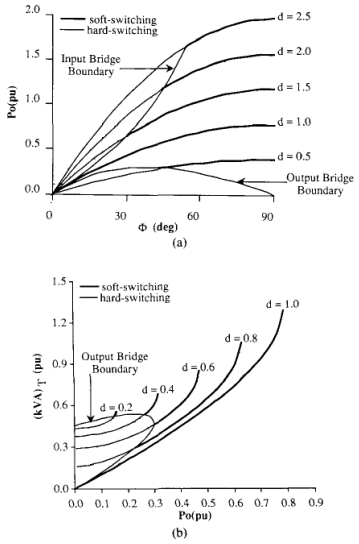


Figura 2-4. Potencia de salida VS desfase variando parámetro d [1].

La figura 2-4 muestra la potencia de salida (p.u) [3] en función del desfase medido en grados, imponiendo las condiciones límites de cada puente antes mencionadas, podemos ver el rango que se considera "Soft-Switching" para este convertidor

La figura 2-4 también muestra el KVA rating en función de la potencia de salida (p.u) con d como parámetro. Imponiendo la condición de límite para el puente de salida, una vez más quedan definidos nuestro rango de "Soft-Swiching".

$$\eta_T = \frac{P_o}{(kVA)_T} \qquad (2.12)$$

[3] Por unidad, normalizada

Adentrándonos en el rendimiento del convertidor respecto a la potencia por unidad que puede dar a la salida y el "KVA rating[4]" del transformador, que viene dado por (2.13)

$$\left(kVA\right)_T = V_1\left[1+d\right]I_{RMS} \qquad (2.13)$$

El convertidor respecto a montajes que utilizan "Hard-Switching" arroja un rendimiento, definido por (2.12) pobre, pues en los primeros este se aproxima a la unidad. En el documento [1], y a partir de esta última figura, se justifica esto último, "el KVA mínimo para transferir la máxima potencia, 0.475 (p.u) y 0.302 (p.u) respectivamente", para el mismo KVA, "la máxima potencia se puede incrementar a 0.422 (p.u) con d=1". La conclusión es que queda definida la máxima relación η_T como 0.89.

El interés en este aspecto para el montaje analizado radica que la mayor frecuencia de conmutación usada aumenta la densidad de potencia, se reduce el peso y por ello la relación tamaña/peso es mucho mayor. Si además extendemos nuestro convertidor a una topología trifásica (que no analizaremos aquí, pero si aparece completamente detallada en la fuente que antes mencionamos), entonces compensa el montaje en los términos que hemos tratado, de ahí su auge y gran utilidad hoy en día.

[4] Mide la capacidad de carga de una maquina eléctrica, en este caso el transformados del montaje propuesto

3 ESTRUCTURAS "NEUROFUZZY"

La innovación distingue a los líderes de los seguidores.

- Steve Jobs, cofundador de Apple -

El capitulo pretende introducir la segunda disciplina que vemos en el proyecto, es decir, las redes neuronales artificiales (ANN)[5] en combinación con la lógica difusa, dando lugar a las estructuras "Neural-Fuzzy" o "Neurofuzzy". Así como explicar la arquitectura de aprendizaje utilizada en el convertidor.

3.1. Introducción a los aproximadores de funciones

Tipicamente los aproximadores de funciones, se basaban en modelos matematicos que representaban el del cual queria obtenerse una predicción de comportamiento. El inconveniente de todo esto, es que no siempre se cuenta con una definición matemática del Sistema, no existe o este no puede aportarnos datos precisos debido al possible ruido de los datos de entrada. Por ello se recurre a aproximadores basados en inteligencia artificial (IA).

3.1.1. Aproximadores polinómicos

Mediante un polinomio de grado n, es posible ajustar el comportamiento de un sistema a partir de una entrada, lo que se denomina también interpolador, los coeficientes de dicho polinomio determinan la salida, se supone como más básico un sistema SISO[6] no lineal. Cuando los datos de entrada tienen ruido, el resultado es oscilatorio, unido a que no puede asumir nuevos datos distintos a los de entrenamiento, acaba por no ser satisfactorio.

Un sistema SISO no lineal queda descrito por el siguiente polinomio de orden M:

$$y = f(x) = a_o + a_1 x + a_2 x^k + ... a_M x^k = \sum_{k=0}^{M} a_k x^k \qquad (3.1)$$

Siendo x la entrada e y la salida. Definidos N datos de entrada, el objetivo del aproximador es ajustarse a dichos puntos mediante un polinomio minimizando el error de la función:

El error queda modelado por la expresión (3.2), de tal forma que valores de salida correspondientes a $x_1, x_2, x_3, ..., x_N$ son $y_{d1}, y_{d2}, y_{d3}, ..., y_N$.

[5] Del inglés, Artificial Neural Network
[6] Del inglés, Single Input Single Output

$$E = 0.5\sum_{n=1}^{N}(y_n - y_{dn})^2 \qquad (3.2)$$

Lo que nos lleva a pensar que para minimizar el error se requiere una técnica de entrenamiento para alcanzar el objetivo.

De las anteriores expresiones, podemos deducir, que el mínimo error puede hallarse con la solución de un conjunto de (M+1) ecuaciones lineales, cada una conteniendo los coeficientes, estás ecuaciones se obtienen mediante la derivada del error respecto a cada coeficiente.

Según el orden del polinomio aumenta, el ajuste es mucho mejor, el inconveniente en algunos sistemas radica en que según se aproxima la función por un polinomio de mayor grado, puede aparecer oscilación debido al sobreajuste.

3.1.2. Teorema de Kolgomorov y aproximadores neuronales

Como hemos podido comprobar, un aproximador polinómico, tiene sus inconvenientes y limitaciones, por ello ahora vamos a ver una alternativa basada en funciones multivariable aproximadas por una suma de funciones no lineales.

Definimos una función y de n variables, y cuyo aproximador utiliza la suma de funciones no lineales

$g_1, g_2, \ldots, g_n$, siendo cada una de estas una función no lineal de una sola variable:

$$y = f\left(x_1 + x_2 + x_3, ..., x_n\right) = g_1 + g_2 + g_2 + ... \qquad (3.3)$$

Para sustentar este concepto, tenemos el Teorema de Kolgomorov, que establecer que “Una función no lineal real y continua $y = f(x)$ de n variables $x = \left(x_1 + x_2 + x_3, ..., x_n\right)$, puede ser aproximada por una suma de funciones continuas de una sola variable”:

$$y(x) = f\left(x_1 + x_2 + x_3, ..., x_n\right) = \sum_{i=1}^{2n+1} g_i = g_1 + g_2 + g_3, ..., g_n \qquad (3.4)$$

Siendo g, una función real y continua no lineal, que depende solo de una sola variable z_i:

$$g_i = g_i(z_i) \qquad (3.5)$$

Entonces,

$$z_i = \sum_{j=1}^{n} h_{ji}(x_j) = h_{1i}(x_1) + h_{2i}(x_2) + ...h_{ni}(x_n) \qquad (3.6)$$

De forma compacta, el Teorema de Kolgomorov, describe matemáticamente un aproximador de funciones no

lineales como:

$$y(x_1 + x_2 + x_3, ..., x_n) = \sum_{i=1}^{2n+1} g_i \left(\sum_{j=1}^{n} h_{ji}(x_j) \right) \qquad (3.7)$$

Por tanto, con 2n+1 funciones es posible aproximar de forma suficiente un sistema. Existe una forma de Kolgomorov, modificada de forma que h_{ij} es reemplazado por $\lambda_i h_j$ con λ_i una constante y h_j una función estrictamente monótona creciente.

Estos términos, no se muestran en el teorema, es decir, no hay una técnica para hallarlos. Pero dicho teorema se ha utilizado como base para comprobar las capacidades de las redes neuronales artificiales como aproximadores de funciones. Las redes neuronales de este tipo con métodos de realimentación del error, más conocidos como "Backpropagation", se consideran buenas aproximadoras de una función. A continuación, un esquema básico de entrenamiento de una ANN:

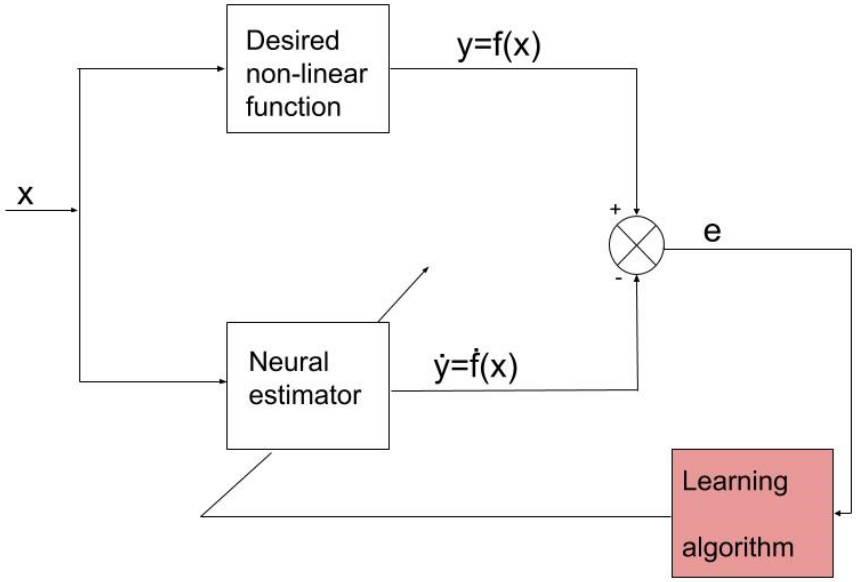


Figura 3-1. Esquema básico entrenamiento ANN [2].

El error se obtiene a partir de la diferencia entre la salida de la función no lineal deseada y la salida del "estimador". Mediante un ajuste de los pesos entre nodos, es decir, coeficiente que pondera las distintas relaciones entre nodos, de forma que una suma normalizada de los mismos a la salida aproxima la función que queremos interpolar.

Mas adelante explicaremos en detalle las redes neuronales artificiales.

3.1.3. Aproximadores de funciones difusos

Ahora hablaremos de la lógica difusa para introducir el segundo concepto que utilizamos en nuestro aprendizaje, ya que como hemos explicado previamente, se trata de una estructura que aun por un lado las redes neuronales artificiales y por otro la lógica difusa.

La lógica difusa, es un sistema que permite expresar los conocimientos de forma sencilla y cercana al lenguaje de comunicación de las personas, de forma que se presenta como una lógica multivaluada que permite representar matemáticamente la incertidumbre, proporcionando herramientas formales para su tratamiento [3].

Las reglas que permiten esta representación matemática pueden ser del tipo “IF-THEN”, entonces una función no lineal se puede aproximar por un numero finito de reglas lingüísticas, conjunto denominado “base”. Cada regla divide el espacio de entrada en un rango determinado. Una aproximador de tipo difuso, por tanto, cubre todo el rango de una función con N reglas lingüísticas.

Un razonamiento difuso sigue una estructura:

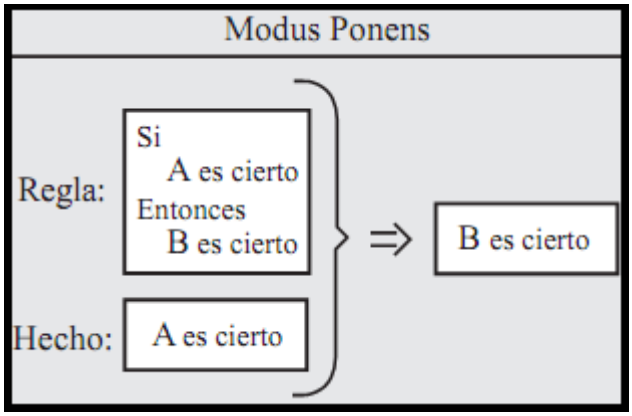


Figura 3-2. Razonamiento difuso [4].

La primera parte de la regla, denominada premisa, se conoce como antecedente, “SI A es cierto”, la segunda parte, que sería la conclusión, se denomina consecuente, es decir, “*Entonces* B cierto”.

A y B representan conjuntos difusos, típicamente las reglas toman la forma “Si x pertenece a A, entonces y pertenece a B”, x e y son valores dentro de los conjuntos difusos A y B. Las reglas lingüísticas nos permiten relacionar entradas con salidas.

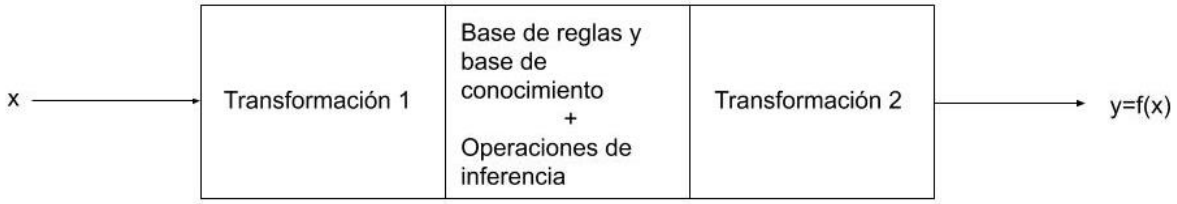


Figura 3-3. Aproximador de funciones difuso básico. Fuente: [2]

En la figura anterior podemos ver un esquema que representa un aproximador defunciones difuso, a partir de unos datos de entrada, mediante una transformación, pasan a ser valores lingüísticos (Proceso conocido como Fuzzyfication), en este momento aparecen las funciones de pertenencia, que asigna un valor del conjunto difuso concreto. En el segundo bloque a partir de una base de conocimiento (reglas IF-THEN) toma una decisión (inferencia) sobre los valores de entrada recibidos del bloque 1. Por último, se transforma de nuevo en un valor numérico de nuevo similar al de la entrada.

Los sistemas más conocidos de lógica difusa son Sugeno, Tsukamoto y Mamdani. En un apartado posterior, trataremos mejor los sistemas de lógica difusa.

3.1.4. Aproximadores de funciones Neural-fuzzy)[7]

Como mencionamos al principio del capítulo, una estructura "Neurofuzzy", une el concepto de lógica difusa y redes neuronales artificiales, esto es así, debido a que podemos coger la parte útil de cada uno, dando lugar a un sistema hibrido de aprendizaje, que al igual que sus componentes es un aproximador de funciones también.

Las principales diferencias con un aproximador de funciones convencional (visto al inicio del capítulo) que dan lugar al hibrido es que el mapeado no lineal que se hacía, se obtiene usando funciones de pertenencias difusas, las cuales se basan en ANNs.

Un Neurofuzzy, tiene la gran ventaja de que puede "aprender" cualquier función no lineal, este tipo de estructuras nos permiten crear controladores de distintos sistemas de forma que se adapten a situaciones imprevistas en las cuales una función matemática no nos serviría, y mucho menos en tiempo real.

3.2. Redes neuronales artificiales. Fundamentos.

Ahora explicaremos mas detalladamente aspectos basicos de las redes neuronales, desde la neurona biologica hasta las posibilidades de implementacion hardware.

3.2.1. Modelos de neurona. ANNs

Para introducir el subcapítulo, es necesario empezar por el modelo de neurona más básico conocido, para poder llegar a una red neuronal artificial multicapa como las utilizadas hoy en día, baza principal de las IA.

3.2.1.1. Modelo de neurona en el ámbito de la biología

biológicamente, nuestro cerebro es una gran red neuronal que contiene aproximadamente 100 billones de neuronas con 1000 interconexiones cada una, dando lugar a 1 cuatrillón de interconexiones. Esto se traduce en un enorme sistema que gestiona tantos nuestros sentidos como nuestros conocimientos.

Una neurona biológica contiene 4 partes diferenciadas: (Ver figura 3-5)

- Las dendritas, que se corresponderían con las entradas de información.
- El cuerpo de la "célula", conocido como soma, el cual recolecta la información de cada conexión combinándola y transformándola.
- La salida o axón, que transmite la información a otras neuronas
- Por ultimo los contactos, que es la unión del axón con las dendritas de otras neuronas. También denominados sinapsis.

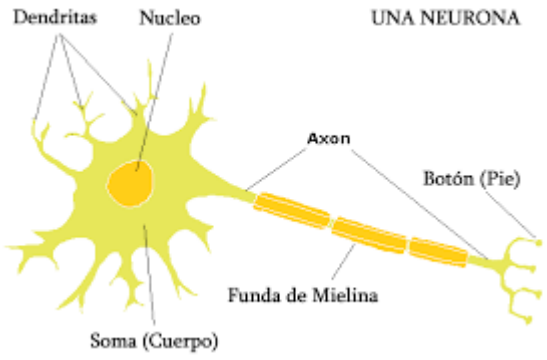


Figura 3-3. Esquema de una neurona biológica [5].

[7] Durante el presente proyecto, se ha optado por la abreviatura "Neurofuzzy" para nombrar el mismo concepto

La transformación a la que aludíamos antes se trata de una suma ponderada de pesos, a la que ya nos hemos referido como un mecanismo básico de las redes neuronales, si este valor es mayor a un umbral determinado, se produce la activación, dando lugar a una salida similar a un tren de pulsos, denominados impulsos nerviosos.

3.2.1.2. Modelo de neurona artificial

Las ANNs están basadas en el modelo del cerebro humano, contienen muchas neuronas, pero artificiales, cuya unión es un peso, están dispuestas en distintas capas y formando estructuras paralelas. Frente a un estimador convencional, las redes neuronales artificiales, a partir de un set de datos, pueden aprender, y por tanto ahora se denominan estimadores de funciones.

Aunque las ANNs pueden implementarse por software o hardware, actualmente están en auge las redes artificiales software.

El modelo de neurona artificial, también denominado perceptrón, al igual que el biológico, tiene 4 partes

(Ver figura 3-4):

- N entradas.
- Sumador ponderado según los pesos de las entradas.
- Una función de activación, típicamente no lineal.
- M salidas. La suma de las entradas por los pesos junto a la función de activación sería similar al soma de las neuronas biológicas, el núcleo de nuestro perceptrón.

  Mientras que las entradas multiplicadas por su peso serian equivalentes a la sinapsis.

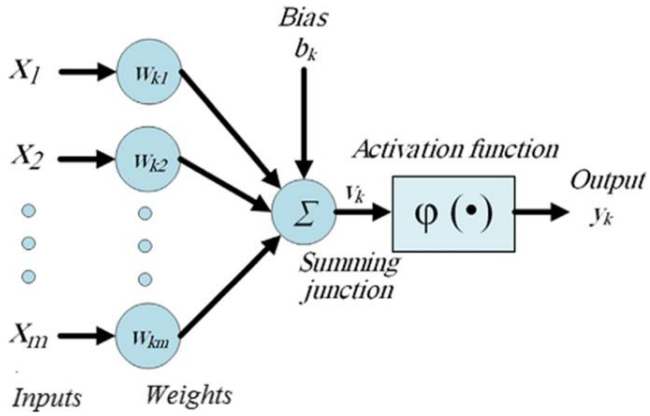


Figura 3-4. Esquema de un perceptrón estático básico.

El modelo de neurona básico se denomina estático debido a que la función de activación, no le sigue ningún elemento dinámico, es decir, un elemento de retraso (delay).

Los pesos, w_{ij} que representan constantes numéricas, se desplazan durante la etapa de entrenamiento, hasta que el error entre la salida y la deseada, se reduce a cero.

Las entradas, x_i que pueden ser externas o internas, las internas, sería el equivalente a la conexión con otras neuronas e introduce un desplazamiento de la respuesta, debido a la suma b_k (figura 3-4). Por tanto, tendríamos n+1 entradas.

La razón de usar el añadido externo (bias) e el sumatorio en algunos modelos, radica en el modelado de sistemas

no lineales, ya que este elemento modifica la activación, de forma que no solo se depende de la función de activación que coloquemos.

La función de activación suele ser no lineal, asemejando a la neurona biológica (transformación somática), de forma que nuestro ajuste durante el aprendizaje pueda adaptarse al sistema.

La función de activación matemática para el caso lineal queda expuesta en la expresión (3.8), conocido como modelo de "Widrow-Hoff":

$$S_i = f_i\left(\sum_{j=1}^{n}\left(w_{ij}x_j(t)+b_i\right)\right) \qquad (3.8)$$

Sin profundizar en las funciones de activación, se pueden clasificar en dos tipos;

- Derivables y no derivables:

  Dentro de esta clasificación tenemos además dos tipos de función de activación, "Soft-limiting" y "Hard-limiting", las primeras devuelven números dentro de un rango de valores reales, y se utilizan cuando es necesario aprender los pesos de la red neuronal. Por ejemplo, la función sigmoide, la función gaussiana o de base radial.

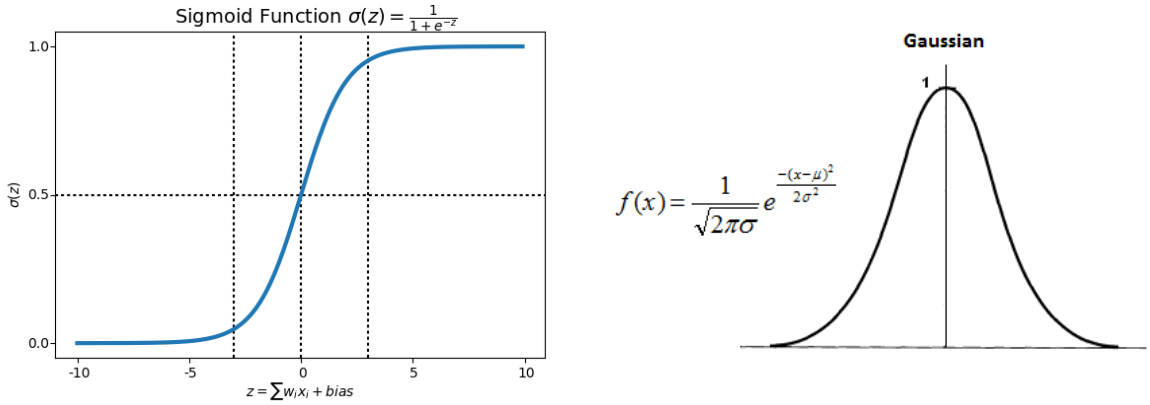


Figura 3-5. Función gaussiana (Derecha) y sigmoide (Izquierda) [8] [6].

Las segundas, se utilizan en algoritmos discretos, y la salida corresponde a un valor binario.

La expresión de la función sigmoide es (3.9), siendo g, la ganancia de activación, este parámetro controla la pendiente.

$$f(z) = \frac{1}{1+e^{(-gz)}} \qquad (3.9)$$

Cuya primera derivada seria (3.10)

$$f'(z) = \frac{ge^{(-gz)}}{\left(1+e^{(-gz)}\right)^2} \qquad (3.10)$$

La función sigmoide, tiene gran parecido a la neurona biológica si aplicamos el limites $g \to \infty$, la activación es

[8] Notar que si se tuviese en cuenta el termino bias, ambas funciones quedarían desplazadas en el eje de coordenadas respecto a la original.

equivalente.

El interés de la función sigmoide está en dos aspectos importantes, su derivada es fácil de obtener, lo que facilita su uso en algoritmos de Backpropagation, y además puede implementar mediante un amplificador no lineal.

- De tipo pulso/escalón:

  El valor de salida solo es significativo si la entrada está cercana al 0, por ejemplo, función escalón y función signo.

3.2.1.3. Redes neuronales artificiales

Se dividen en redes de una sola capa y en redes multicapa.

Las redes neuronales de una sola capa incorporan por lo menos una sola neurona como las que hemos definidos previamente y n entradas, el número de salidas m, queda definido por el número de neuronas. La expresión algebraica de la salida es (3.11).

Siendo w_1 y $F1$, la matriz de pesos correspondiente a cada entrada x[9], de dimensión m filas y n columnas.

y la matriz de activación diagonal de m elementos respectivamente.

$$y1 = F1(w1x + B1) \qquad (3.11)$$

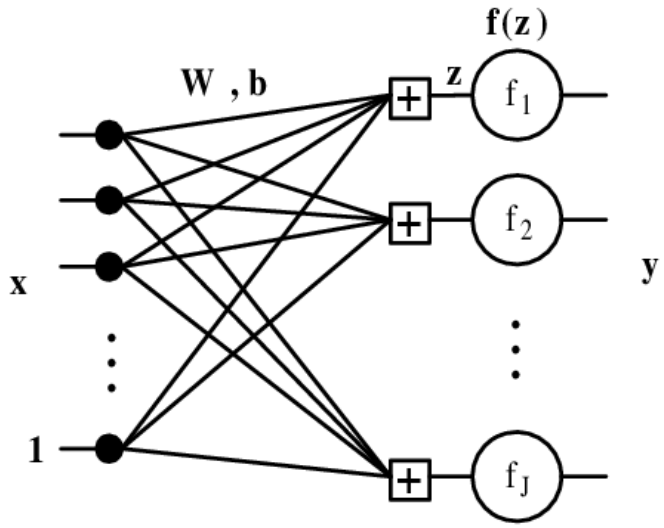


Figura 3-6. Red neuronal de una sola capa [7].

Las redes neuronales multicapa, implementan un esquema mucho más complejo al introducir muchas capas

[9] El vector de entrada define lo que se conoce como espacio de entrada

paralelas de distinto número de nodos, estas capas denominadas ocultas, realizan cada una procesados de datos similares al de la red de una sola capa, entre sus nodos (los de las capas ocultas) no hay interconexiones, pero todos los nodos de la capa anterior y la siguiente están conectados, esta conexión sí que puede ser variable.

Según aumentan el número de capas, la complejidad crece a un ritmo alto, normalmente con dos capas es suficiente para aprender con precisión una mayoría de sistemas no lineales. Y a diferencia de una red de una sola capa, no hay una diferencia de coste computacional significativa respecto a una de 2-3 capas.

A continuación, tenemos un ejemplo de red neuronal de 2 capas ocultas y una capa de salida (Figura 3-7). Para obtener la expresión de salida de esta red explicaremos el procedimiento detallado.

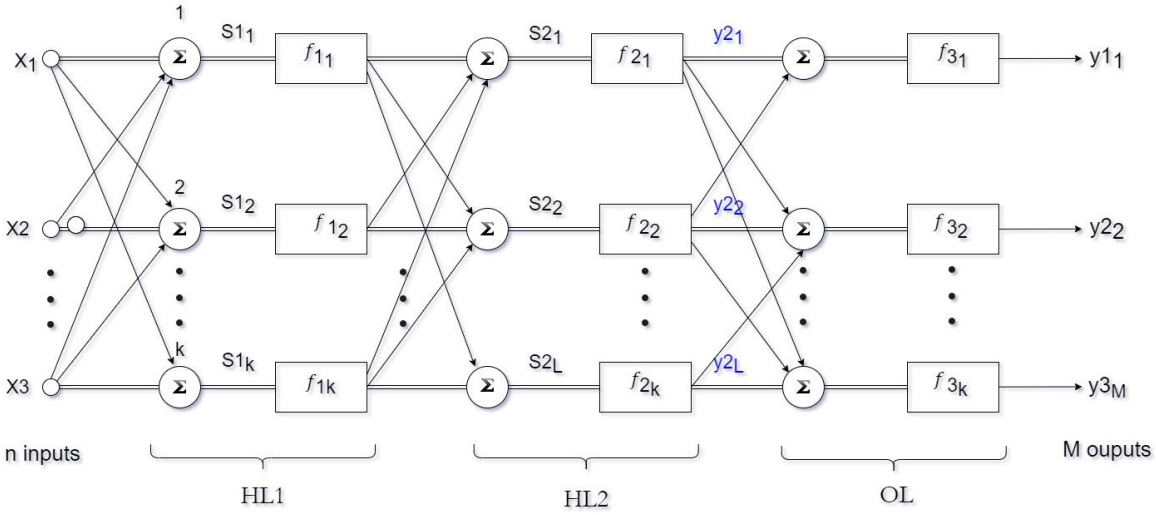


Figura 3-7. Red neuronal artificial de 3 capas "Feedfoward" [2].

Podemos recursivamente establecer la salida de una capa como la salida de la anterior hallada con (3.12), tal que así:

$$\begin{aligned} y1 &= F1(w1x + B1) = F1(S1) \\ y2 &= F2(w2y1 + B2) = F2(S2) \\ y3 &= F3(w3y2 + B3) = F3(S3) \end{aligned} \qquad (3.12)$$

Siendo x e y las entradas y salidas respectivamente, de dimensiones N, y las salidas dependiendo del número de nodos de cada capa K, L y M (3.13):

$$\begin{aligned} x &= [x_1, x_2, ..., x_n]^T \\ y1 &= [y1_1, y1_2, ..., y1_k]^T \\ y2 &= [y2_1, y2_2, ..., y2_L]^T \\ y3 &= [y3_1, y3_2, ..., y3_M]^T \end{aligned} \qquad (3.13)$$

$$\begin{aligned} S1 &= [S1_1, S1_2, ..., S1_k]^T \\ S2 &= [S2_1, S2_2, ..., S2_L]^T \\ S3 &= [S3_1, S3_2, ..., S3_M]^T \end{aligned} \qquad (3.14)$$

F es una matriz diagonal de nuevo (3.15) compuestas por los vectores de activación de entrada a cada capa, descritos en (3.14):

$$\begin{aligned} F1 &= diag[F1_1(S1_1), F1_2(S1_2), ..., F1_k(S1_k)]^T \\ F2 &= diag[F2_1(S2_1), F2_2(S2_2), ..., F2_L(S2_L)]^T \\ F3 &= diag[F3_1(S3_1), F3_2(S3_2), ..., F3_M(S3_M)]^T \end{aligned} \qquad (3.15)$$

Retomando la expresión (3.1), una salida y_3 cualquiera se expresa como:

$$y3 = F3(w3F2(w2F1(w1x + B1) + B2) + B3) \qquad (3.16)$$

Por tanto, a partir de las entradas x, y los pesos como parámetros, podemos determinar la salida.

Notar que para llevar esto a la práctica, mediante un aprendizaje, es necesario aplicar alguna técnica como la ya nombrada “propagación hacia atrás” o “Backpropagation” del error, lo que viene a decir, es que los pesos se van recalculando en base a este error. Mas adelante usaremos esta técnica y la explicaremos detalladamente.

Tras conocer cuál es la estructura básica de una red neuronal artificial, podemos ver que puede haber montones de estructuras distintas, y que cada una tiene unas cualidades, que determinan en que tarea de aprendizaje puede desenvolverse mejor.

3.3. Sistemas de logica difusa (FLS)

Basandonos en los aproximadores difusos que explicamos en el apartado 3.1, hablaremos de los sistemas de logica difusa, que se tartan de aproximadores de ese tipo pero que pueden usarse para el control de sistemas de forma inteligente.

3.3.1. Introducción. Aplicaciones

Como comentamos en 3.1, existen distintos tipos de sistemas difusos, Mamdani, Tsukamoto y Sugeno, que son los más conocidos. A parte existen otros tantos, que siguen la estructura de 3 fases que vimos en los aproximadores, aunque no todos tienen necesariamente que cumplirla. Estas eran la “fuzzicacion”, una transformación dependiente de la base de reglas y conocimiento y la “defuzzificacion”.

Realmente un sistema lógico difuso (FLS), es un aproximador difuso genérico cuyo objetivo es emular otro sistema, dando a unas entradas el mismo resultado que obtendríamos de dicho sistema.

Las aplicaciones de los FLS son amplia, orientadas sobre todo a maquinas eléctricas y motores, su principal ventaja frente a un control PID clásico, radica en que, un FLS puede adaptarse a situaciones cambiantes o incluso puede utilizarse para hallarlas constantes de un controlador clásico siendo más rápido que otras técnicas. Se distinguen realmente dos tipos de sistemas difusos, lo denominados “Fuzzified” y los controladores de lógica difusa (FLC).

La principal diferencia, la acabamos de comentar, un PI se denomina "Fuzzified", cuando la lógica difusa solo se encarga de realizar el "tunning" de las constantes de control Kp y Ki, un FLC directamente se encarga del control. ¿Qué ventaja existe frente a los controladores clásicos?, no es necesario un modelo matemático de forma estricta ni tener conocimientos de control, lo que facilita una implementación rápida para investigación y prototipado.

3.3.2. Conceptos básicos de los FLS

A los conceptos que vimos en el apartado 3.1, vamos a añadir conceptos que facilitan la compresión de este tipo de sistemas, que se unen a los básicos de lógica difusa.

3.3.2.1. Conjuntos difusos y funciones de pertenencia

Se define conjunto difuso, como un conjunto A' que contiene una colección de elementos x, pertenecientes a un universo X, el cual queda descrito por la función de pertenencia $\mu^{A'}(x)$ (que toma valores entre 0-1 exclusivamente)

Por tanto, el conjunto A queda definido matemáticamente como se indica en (3.17)

$$A' = \left\{ (x, \mu^{A'}(x)) \mid x \in X \right\} \qquad (3.17)$$

Quedando definido el grado de pertenencia de un elemento x al conjunto A', lo que indica la cercanía que tiene el valor x con el resto de los valores del conjunto A'.

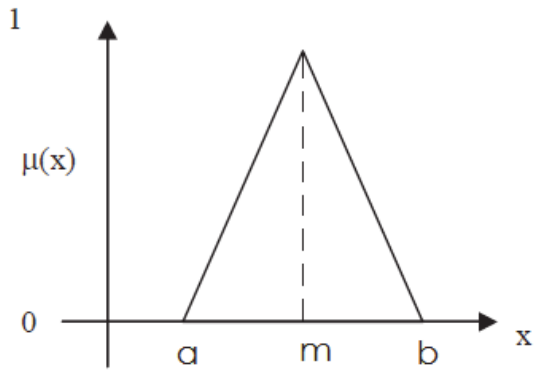


Figura 3-8. Función de pertenencia triangular [8].

Existen múltiples tipos de funciones de pertenencia entre las cuales están la gaussiana, de forma de campana y sigmoide.

Entre conjuntos existen operaciones básicas, a continuación, mostramos el resultado de la unión, intersección de los conjuntos A y B. Por último, el complementario del conjunto A.

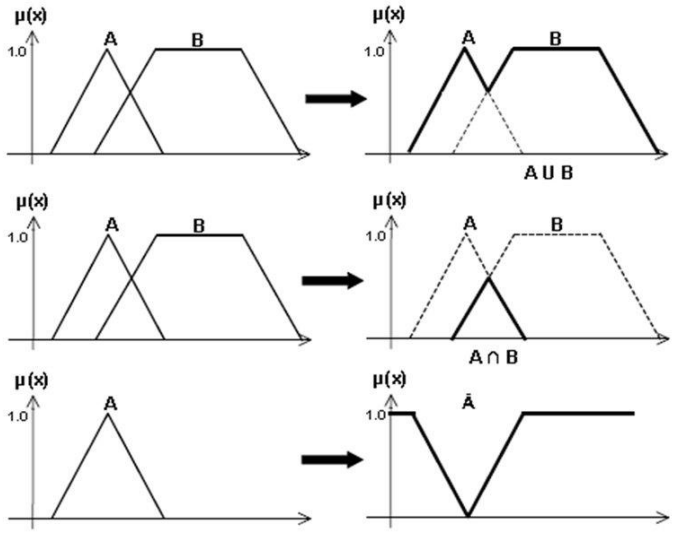


Figura 3-9. Operaciones básicas entre conjuntos difusos [9].

Existen muchas más operaciones, muchas de ellas son modificaciones de las básicas, pero con ligeros cambios.

No se tratarán otros aspectos de los FLSs ya que no es necesario para la correcta comprensión de la estructura que se propondrá para el aprendizaje

3.3.3. FLS de tipo Mamdani

En este apartado se explicará uno de los sistemas de lógica difusa más conocidos, el de tipo Mamdani, tratándolo como controlador.

Este sistema fue propuesto en 1975 por Ebhasim Mamdani, el objetivo era el control de una máquina de vapor usando las reglas elaboradas a partir de los conocimientos de los propios trabajadores.

El FCL contiene 4 partes:

- **Fuzzificador:**

  El Fuzzificador realiza la primera transformación con los valores de entrada, los cuales son nítidos y necesitan ser llevador a valores de tipo difuso, es decir, discretizarlos. Por tanto, acaban entendiéndose por las reglas lingüísticas. Para ello se utiliza funciones de pertenencia, que realizan este mapeado.

- **Base de conocimientos:**

  Se trata de una base de datos y base de reglas acordes al control a realizar, es decir, estas reglas especifican directamente objetivos en el argot del control. El controlador, a partir de los datos de entrada y las reglas IF-THEN, y determina la salida correcta. Esta base de reglas sería necesario determinarlas mediante ANNs, conociendo el sistema a controlar…etc.

- **Motor de toma de decisiones o de razonamiento:**

  Toma las decisiones usando los valores difusos que nos dio la primera transformación y la base de conocimiento anteriormente explicada

- **Defuzzificador:**

  Devuelve los valores difusos devueltos por el motor de inferencia a valores nítidos que, si se entienden en el contexto del control del sistema, para ello se utilizan funciones de pertenencia asociadas a las reglas. Existen distintas técnicas que explicaremos más adelante.

A continuación, esquemático de las fases explicadas (FLS de tipo Mamdani)

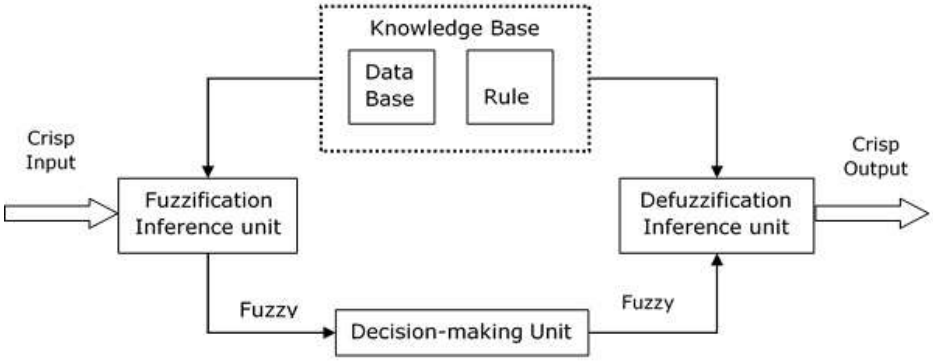


Figura 3-10. Esquemático de un FLC de tipo Mamdani [10].

3.3.3.1. Técnicas de Defuzzicación

Las principales técnicas son:

1) Método COG (Centro de gravedad):

El valor Defuzzificado se corresponde con el centroide del conjunto difuso de salida (Expresión 3.18)

$$z^{*COG} = \frac{\sum_{k=1}^{n} z_k \mu(z_k)}{\sum_{k=1}^{n} z_k} \qquad (3.18)$$

2) Método H (De la altura)

El valor Defuzzificado se corresponde con el (3.19)

$$z^{*H} = \frac{\sum_{k=1}^{n} p_k c_k}{\sum_{k=1}^{n} p_k} \qquad (3.19)$$

3) Método MOM (Media min-max)

El valor defuzzificado se corresponde con la media del valor máximo más pequeño y el valor máximo más grande (3.20)

$$z^{*MOM} = \frac{z_1 + z_2}{2} \qquad (3.20)$$

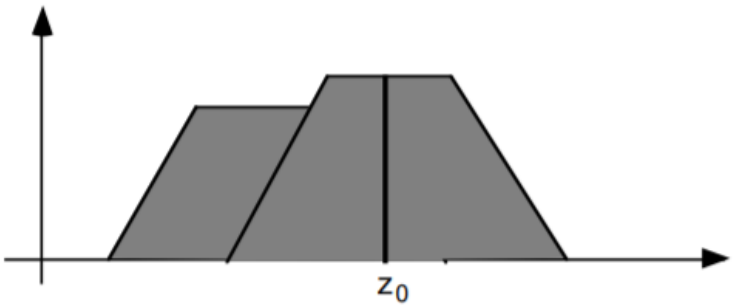


Figura 3-11. Método MOM Fuente [11].

4) Método FOM (First of máxima)

El elemento defuzzificado se corresponde con el valor máximo más pequeño (3.21)

$$z^{*FOM} = z_0 \tag{3.21}$$

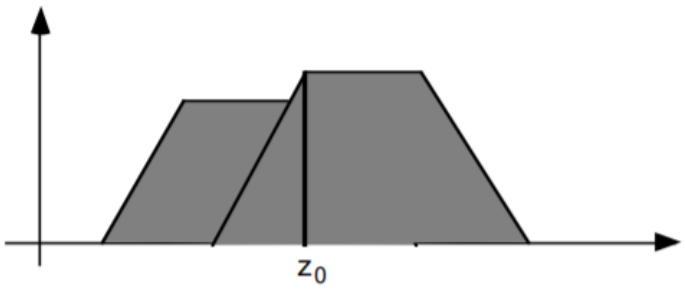


Figura 3-12. Método FOM [11].

3.3.3.2. Mamdani como FLC Concepto

Existen varios tipos de FLCs, que dependen del número de entradas y la forma de la señal de salida. Si tenemos en cuenta que en los controladores clásicos la entrada es un error entre señales, es decir, la referencia y la salida real, entonces se podría decir que un FLC típico tiene dos entradas, las cuales son el error “E” y el cambio de error “CE”.

Si nos centramos en un PI, cuya salida es:

$$u = K_p e + K_i \int e dt \tag{3.22}$$

Las reglas lingüísticas de un FLC-PI tendrían la forma:

“IF E es X y CE es Z, ENTONCES CU es y” Siendo x, y, z conjuntos difusos.

A el proceso que une los resultados de aplicar las reglas lingüísticas a los valores Fuzzificador de los valores de entrada (nítidos), se le denomina composición, ya que valga la redundancia se trata de una composición típicamente de dos operadores difusos.

Existe varios tipos de composiciones, la más conocida la composición denominada "sup-min", que consiste en que por cada regla se obtiene una función de pertenencia combinación de las existentes en la regla aplicando el operador mínimo, es decir, la función de pertenencia es la intersección de las funciones. A las restantes (una por regla) le aplicamos el operador supremo, que coge el valor mínimo de todos los máximos.

Otro método conocido es el "Max-Min", que es parecido al anterior, pero que en el segundo paso aplica la operación máximo, que es la unión de las funciones de pertenencias resultantes de aplicar el mínimo en cada regla.

A continuación, para cerrar el apartado, en la siguiente figura se muestra de forma sencilla el proceso que sigue un FLS de tipo Mamdani, que además en la defuzzificacion aplica el método COG.

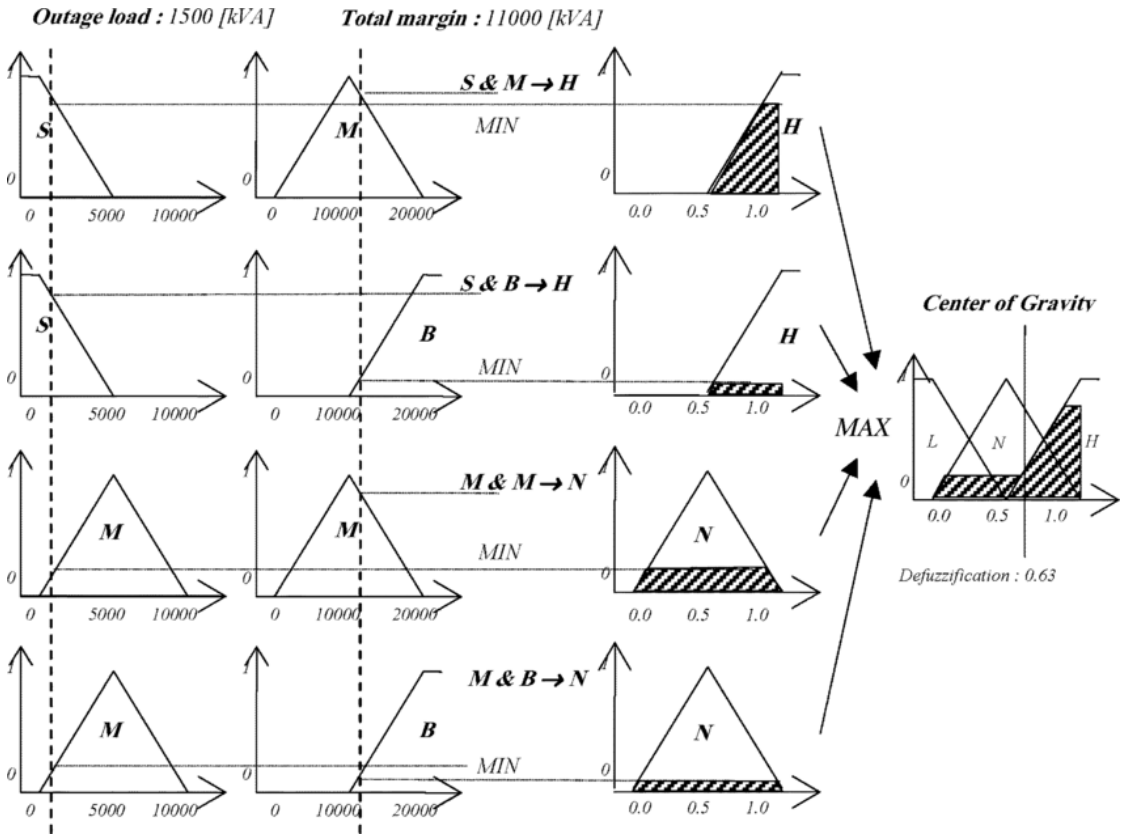


Figura 3-13. Etapas de un FLC tipo Mamdani aplicando COG [12].

Podríamos continuar profundizando en los FLS con los otros tipos más conocidos, Sugeno y Tsukamoto.

3.4. Sistemas difusos-neuronales (FNS)

Los sistemas difusos-neuronales, o como denominaremos de aqui en adelante, “Neurofuzzy”, resultan como hemos mencionado en otros apartados, de la combinación de las ANNs y los FLSs, cogiendo las caracteristicas que mas interesan de cada uno.

Se definen dos tipos de FNS, los sistemas difusos “neuralizados”, y los sistemas neuronales “fuzzificados”. En los primeros, el mapeado lo realiza un FLS para llevarlo a una ANN, en el Segundo, los conceptos de las FLS se introducen en las propias ANN.

En Resumen, la hibridacíon entre cada parte se realiza de forma que sea ventajosa para la tarea que se quiera llevar a cabo, es decir, si necesitamos una base de conocimiento de nuestro Sistema, pero no no es accessible, se puede crear de cero a partir de multiples de datos de entrada.

Los tipos de FNS mas conocidos, al igual que en los sistemas difusos, son los de Tsukamoto, Mamdani,Sugeno y Wang.

En este Subcapitulo, explicaremos el de tipo Mamdani para poder concoer mas de cerca como funcionan los Neurofuzzy, para posteriormente en el apartado 3.7, explicar el de tipo Wang, que será el utilizado para el control del convertidor desarrollado en el primer capitulo de este trabajo (DAB).

3.4.1. Mamdani FNS

A continuación, explicaremos lo más básico que es necesario conocer sobre los FNS usando como ejemplo uno de tipo Mamdani.

3.4.1.1. Introducción

Los Neurofuzzy, como hemos explicado a lo largo de este capítulo, debido a contener la unión de las cualidades más ventajosas de los sistemas difusos y las redes neuronales, son como aproximadores una muy buena opción, y por ende como controladores, que como dijimos, frente a un controlador clásico poseen varias ventajas. Si hablamos de los controladores Neurofuzzy, existen multitud de esquemas e implementaciones, al igual que métodos de aprendizaje y tuneado de los mismos. Se habla de entrenamientos supervisados, sin supervisar, híbridos etc.

3.4.1.2. Controlador usando FNS Mamdani

Este tipo de controlador implementa una red neuronal multicapa Feedfoward como las que vimos en el apartado 3.2.1.3 y está basado en un FLS tipo Mamdani.

Pasemos a explicar la red neuronal que lo forma. (Figura 3-15)

El ejemplo está compuesto por 5 capas:

Los pesos entre los nodos de cada capa previa y posterior se denominarán w_{ji}^{k} , aunque realmente, la mayoría son de valor la unidad, son más bien una forma representativa, para comprender el esquema respecto a la teoría que hemos visto de ANNs.

1. **Capa de entrada:**

   Esta capa contiene un nodo por cada variable del controlador, cada nodo distribuye sus salidas a los distintos nodos de la siguiente capa.

2. **Capa de fuzzicacion:**

   Cada nodo de la segunda capa, la que realiza la defuzzificacion, se corresponde con una función de

pertenencia de tipo Gaussiana distinta. El número de nodos en esta capa depende del número de conjuntos difusos asociados a las variables de entrada. En la expresión (3.23), se calcula la salida de los nodos de esta capa[10].

$$\hat{y} = e^{\frac{-(x^2-m)^2}{\sigma^2}} \qquad (3.23)$$

Siendo m y σ denotan el centro y el ancho de la campana de Gauss.

3. Capa AND

La tercera capa, se denomina así ya que aplica el operador difuso AND (intersección), tal que:

$$y^{-i} = \min(x_1, x_2, ..., x_k) \qquad (3.24)$$

4. Capa OR:

La cuarta capa realiza una suma acotada de las entradas a sus nodos aplicando el operador difuso OR (unión). Mediante la expresión (3.25) se calcula la salida de dicha capa.

El número de nodos de esta capa depende de los conjuntos difusos asociados a las variables de salida.

$$y = \min(1, \sum_{i=1}^{k} x_i)\Sigma \qquad (3.25)$$

5. **Capa de defuzzificacion:**

La última capa de la red, para realizar la defuzzificacion utiliza el método COG explicado en el apartado 3.3.3.1, el número de nodos de salida dependerá del número de variables de salida del controlador.

Este caso es el de la única capa donde los pesos con su capa antecedente son distintos de la unidad e igual al producto del centro y ancho de la gaussiana de la función de pertenencia que introduce esta capa. La salida que incluye dichos pesos se calcula con la siguiente expresión:

$$\bar{y} = \frac{\sum_{i=1}^{k} m_i \sigma_i x_i}{\sum_{i=1}^{k} \sigma_i x_i} \qquad (3.26)$$

[10] "x" denota siempre la entrada a la capa, que depende de la salida de la anterior y el peso entre ambas capas

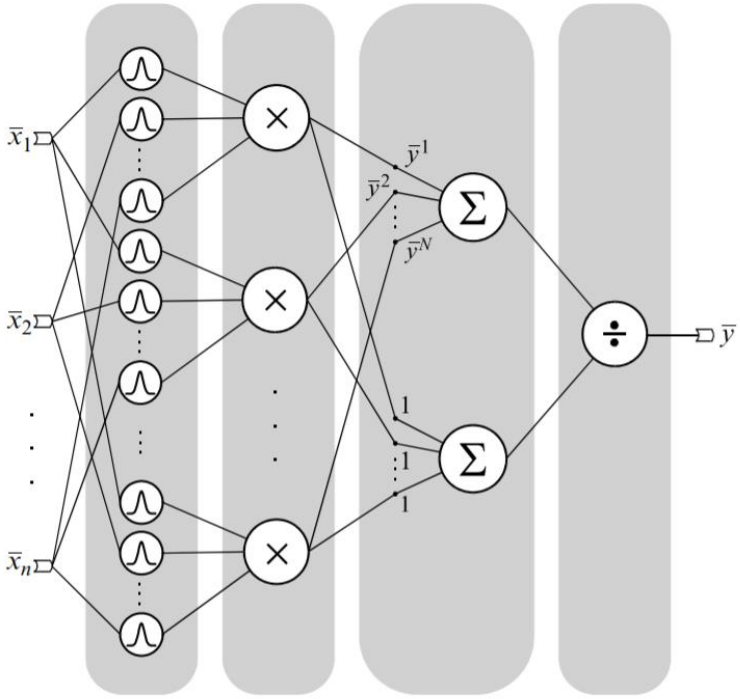


Figura 3-14 *Esquema de una red Mamdani Neurofuzzy* [13].

Típicamente el entrenamiento del FNC, se hace en dos fases, una primera que permite inicializar los pesos parámetros de las Gaussianas, aplicando entrenamientos supervisados. Y una segunda fase, donde se suele utilizar el algoritmo de Backpropagation para ajustar todos los parámetros de forma precisa, se utiliza el mismo set de entrenamiento que en la primera fase.

En el apartado 3.6 se introducirá el Neurofuzzy de tipo Wang de forma detallada. Antes se explicará en detalle el algoritmo de Backpropagation.

3.5. Algoritmo de “Backpropagation” del error

Una vez introducidos las redes neuronales artificiales y los sistemas de logica difusa, camino de explicar la estructura “Neurofuzzy” utilizada en el proyecto, es necesario hacer un inciso, y explicar en que consisted de forma mas detallada este algoritmo.

3.5.1. Aprendizaje por “Backpropagation”

El algoritmo pretende modificar los pesos de la red de forma que el error entre los distintos valores de los datos de entrada y los valores de diseño sea mínimo.

El error total de la red se expresa mediante (3.27)

$$E = \frac{1}{2}\sum_{k=1}^{P}\sum_{j=1}^{K}(d_{kj} - o_{kj})^2 \qquad (3.27)$$

P identifica al número de patrones en el set de entrenamiento, y K el número de entradas.

Los términos d_{kj} y o_{kj}, son respectivamente el valor de diseño del dato de entrada, y dicho dato. El algoritmo se basa en el gradiente decreciente del error, tal que la variación del peso entre los nodos j e i es igual a:

$$\Delta w_{ji} \approx -\frac{\partial E}{\partial w_{ji}} \tag{3.28}$$

Por tanto, podemos observar en primer lugar, que en el aprendizaje de una ANN usando Backpropagation del error, el error se introduce en la misma solo y únicamente a través de los pesos, que a la vez son parámetro de la salida de la red.

Ahora mediante uno sencillos pasos, vamos a transformar la expresión (3.29) para que pueda ser implementada en un algoritmo de aprendizaje.

Definimos el sumatorio de cada peso que relaciona cada entrada con la salida del nodo (3.29)

$$S_j = \sum_{i=1}^{J} w_{ji} y_i \tag{3.29}$$

Con J igual al número de entradas, y w_{ji} el peso entre el nodo j que pertenece a la capa de salida (posterior), y el nodo i de la capa previa, cuya señal es y_i.

En (3.30) se define la salida del nodo j-esimo.

$$o_j = f_j(S_j) = f(S_j) \tag{3.30}$$

Para resolver la derivada parcial, es necesario aplicar la regla de la cadena, teniendo en cuenta (3.31), ya que S_j depende de los pesos y estos del cambio del error previamente definido, lo que resulta en la expresión (3.31)

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial S_j} \frac{\partial S_j}{\partial w_{ji}} \tag{3.31}$$

Podemos identificar el segundo termino de (3.31) como una constante debido a que es la derivada de la suma ponderada de pesos, antes descrita, pues no cambia ante un mismo patrón a la entrada de nuestra red:

$$\frac{\partial S_j}{\partial w_{ji}} = y_i \tag{3.32}$$

Como E es función compuesta de S_j, es decir:

$$E = E[o_j(S_j)] \tag{3.33}$$

El primer termino de (3.31) ahora mediante la regla de la cadena aplicada de nuevo se expresa como:

$$\frac{\partial E}{\partial S_j} = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial S_j} \tag{3.34}$$

Hablando de la expresión (3.34) cuyo segundo término es la derivada de la función de activación, por tanto, simplificamos (3.35)

Y que el primer termino de (3.34), realmente es la diferencia con signo negativo entre el dato de diseño y el que tenemos a la entrada (3.36).

$$\frac{\partial o_j}{\partial S_j} = f_j^{'}(S_j) \tag{3.35}$$

$$\frac{\partial E}{\partial o_j} = -(d_j - o_j) \tag{3.36}$$

De forma que la expresión inicial que teníamos sobre la variación de los pesos (3.28), queda como se muestra en (3.37)

$$\Delta w_{ji} \approx y_i (d_j - o_j) f_j^{'}(S_j) \tag{3.37}$$

Si además definimos el siguiente termino que está contenido en (3.37), que sería el error equivalente del nodo j propagado hacia atrás, es decir, el error de la señal de entrada por la derivad de su correspondiente función de activación.

$$\delta_j = (d_j - o_j) f_j^{'}(S_j) \tag{3.38}$$

Si además añadimos una constante que se suele denominar en la jerga de las ANNs como paso de aprendizaje o factor de aceleración, tenemos en (3.39) la variación necesaria en cada iteración k, del peso ji-esimo.

$$\Delta w_{ji} \approx \eta y_i \delta_j \tag{3.39}$$

$$w_{ji}(k+1) = w_{ji}(k) + \Delta w_{ji} = w_{ji}(k) + \eta y_i \delta_j \tag{3.40}$$

El cálculo de los pesos para los nodos de las capas ocultas es similar al anterior, pero teniendo en cuenta que si los pesos se referían al nodo anterior (j-esimo) y posterior (i-esimo), ahora los pesos que se pretenden cambiar se refieren al nodo anterior (h-esimo), que puede referirse a otra capa oculta anterior, y al nodo posterior (j-esimo):

Quedando en este caso la variación de cada peso como muestra (3.41):

$$\Delta w_{ih} = -\frac{\partial E}{\partial w_{ih}} = \eta \delta_{yi} y_h = \eta f_i^{'}(S_i)\left(\sum_{j=1}^{K} \delta_j w_{ji}\right) y_h \tag{3.41}$$

Que, por tanto, ahora depende de los pesos antes calculados, es decir los calculados para la capa de salida, de la expresión (3.38), de la salida del nodo h-esimo y de la derivada de la función de activación para los nodos i-esimo.

En otras palabras, como se trata de una propagación hacia atrás del error, los pesos de una capa previa a otra oculta o a la de salida, necesitan de los pesos de la capa posterior y de la derivada de la función de activación entre ambas capas.

3.6. FNS de tipo Wang

Realmente este tipo de Neurofuzzy es una ligera variación del Mamdani explicado en 3.5

3.6.1. Arquitectura

Antes de explicar la arquitectura de este Neurofuzzy, como hemos dicho varía respecto al Mamdani, debido a que se suponen los siguientes aspectos:

- El operador AND se aplica como el producto algebraico.
- Hay el mismo número de funciones de pertenencia a la entrada que reglas.

En el resto de los aspectos es igual al Mamdani explicado.

Las reglas lingüísticas del FNS, teniendo en cuenta M reglas, n entradas, éstas serán de la forma:

Regla 1: "SI x_1 ***es*** F_1^1 ***Y*** x_2 ***ES*** F_2^1 ***Y … Y…*** x_n ***es*** F_n^1 ***ENTONCES*** y ***es*** G^1 ***"***

Regla 2: "SI x_1 ***es*** F_1^2 ***Y*** x_2 ***ES*** F_2^2 ***Y … Y…*** x_n ***es*** F_n^2 ***ENTONCES*** y ***es*** G^2 ***"***

.

.

.

Regla k: "SI x_1 ***es*** F_1^k ***Y*** x_2 ***es*** F_2^k ***Y … Y…*** x_n ***es*** F_n^k ***ENTONCES*** y ***ES*** G^k ***"***

Siendo G^k, F^k conjuntos difusos.

La salida de la red queda de la siguiente manera:

$$y = \frac{\left[\sum_{i=1}^{M} y_{ctr}^{i} \prod_{j=1}^{n} \mu F_j^i(x_j)\right]}{\left[\sum_{i=1}^{M} \prod_{j=1}^{n} \mu F_j^i(x_j)\right]} \qquad (3.42)$$

Siendo x_j la entrada j a la red, y_{ctr}^i denominado centroide, es el centro de la función de pertenencia de cada regla asociada a la salida.

$\mu F_j^i(x_j)$ se trata de la función de pertenencia de la entrada j, cuya expresión se trata de la campana de Gauss:

$$\mu F_j^i = \mu^i(x_j) = a_j^i e^{-\left\{\left(\frac{x_j - x_j^i}{\sigma_j^i}\right)^2\right\}} \qquad (3.43)$$

Cuyos parámetros son a_j, x_j^i y σ_j^i, el primero se ha tomado con valor la unidad, lo que significan que la campana devolverá valores entre 0 y 1. El centro y el ancho, varían y corresponden según la entrada j y el nodo i.

A continuación, mostramos el esquema de la red Neurofuzzy Wang de dos entradas y 1 salida, con el objetivo de explicar sus capas.

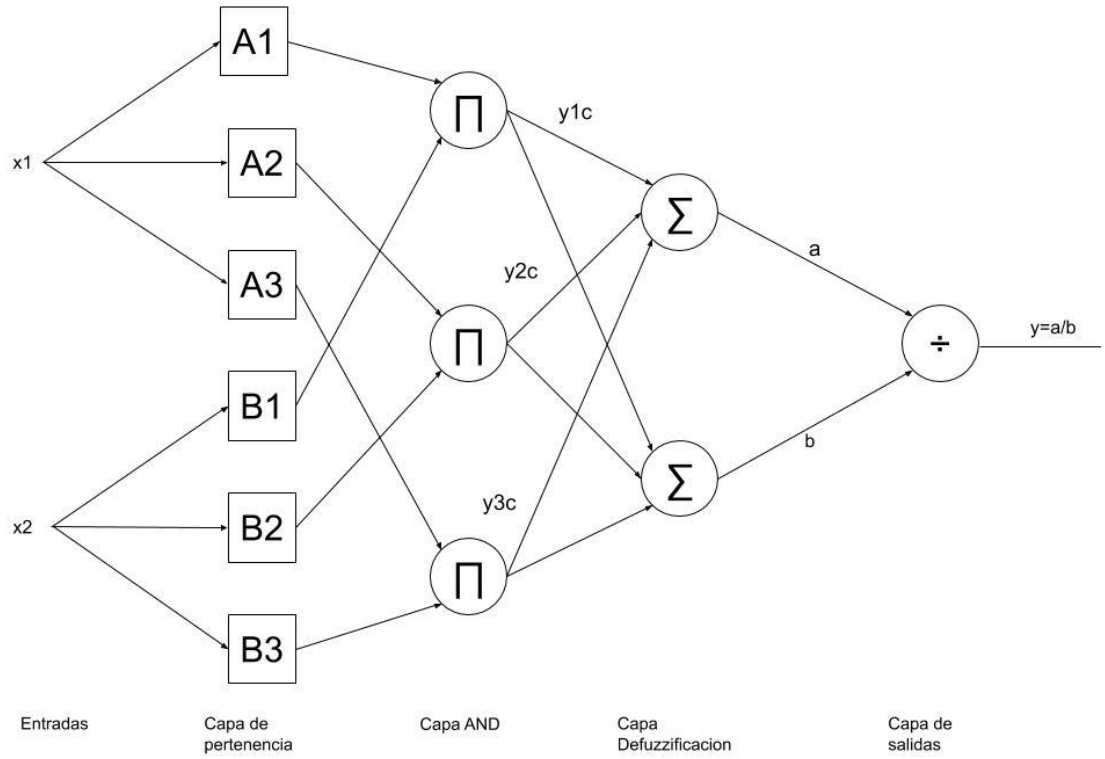


Figura 3-15. Esquemático de un Neurofuzzy de tipo Wang.

La primera capa, es la capa de pertenencia, donde cada nodo es la función de pertenencia μ^{Ak} o μ^{Bk}, conectada a la capa AND que es la siguiente, los pesos que unen ambas capas son de valor unidad.

La segunda capa, es la capa AND, que realizar el producto de las salidas de las funciones de pertenencia dos a dos (esto es debido a que tenemos dos entradas, cuyos valores difusos una vez transformados)

Cada nodo de esta capa calcula la expresión $z^i = \mu^{Ak}\mu^{Bk}$, cuya salida tiene la expresión $z^i y_{ctr}^i$ para las salidas impares y z^i para las salidas pares.

La tercera capa es la capa de defuzzificacion, formada por dos nodos, el primer recibe cada de una de las salidas de cada nodo de la capa AND, con un peso que denominaremos y_{ctr}^{i} , de aquí en adelante centroide o centro. El segundo nodo recibe las mismas salidas que el primero, pero el peso de interconexión es de valor la unidad. Por tanto, la salida de cada nodo, denominada a y b respectivamente son:

$$\begin{aligned} a &= z^i y_{ctr}^i = z^1 y_{ctr}^1 + z^2 y_{ctr}^2 + z^3 y_{ctr}^3 \\ b &= z^1 + z^2 + z^3 \end{aligned} \tag{3.44}$$

La última capa, denominada de salida, tiene un solo nodo, que realiza la división de a y b, cuya expresión a la salida, que es igualmente la del neurofuzzy es:

$$y = \frac{a}{b} = \frac{\left(\sum_{i=1}^{M=3} z^i y_{ctr}^i \right)}{\sum_{i=1}^{M=3} z^i} \tag{3.45}$$

Como dijimos al principio del apartado, el neurofuzzy de tipo Wang, usa como Defuzzificador el método COG (Técnicas de Defuzzicación), pero tal y como mostramos la arquitectura, la defuzzificacion se hace así, pero entre las dos últimas capas.

3.6.2. Aprendizaje del Neurofuzzy Wang

Explicaremos como se realiza el aprendizaje del neurofuzzy hasta ahora mostrado. Para ello partiendo de lo que se vio en 3.5 del algoritmo de Backpropagation, deduciremos las expresiones necesarias para realizar el aprendizaje. El erro cuadrático medio tomando como $f = f\left(x_p\right)$ la salida del neurofuzzy, y d_p como el valor diseñado de la salida para cada par de datos p introducido a la red, se define en (3.46).

$$e_p = \frac{1}{2}\left[f\left(x_p\right) - d_p \right]^2 \tag{3.46}$$

El objetivo es calcular los parámetros variables del Neurofuzzy minimizando este error, que tal como vimos, utilizando el algoritmo de BP, se necesita hacer la derivada parcial del error definido, respecto a cada uno de estos parámetros.

En la expresión se define el valor del centroide en la siguiente iteración del aprendizaje como el valor actual restando una variación, que depende, de α , denominado paso de aprendizaje, y de la derivada parcial del error respecto al centroide:

$$y_c^i(k+1) = y_c^i(k) - \alpha \left[\frac{\partial e_p}{\partial y_c^i} \right]_k \tag{3.47}$$

De (3.46) y teniendo en cuenta $f\left(x_p\right) = f = a/b$ la dependencia del error respecto al centroide se produce debido a la salida del primer nodo de la capa de defuzzificacion (3.44), entonces, la derivada parcial de (3.47), se calcula como:

$$\frac{\partial e_p}{\partial y_c^i} = (f - d_p)\frac{\partial f}{\partial a}\frac{\partial a}{\partial y_c^i} = \left(\frac{f - d_p}{b} \right) z^i \tag{3.48}$$

$$y_c^i(k+1) = y_c^i(k) - \alpha \left(\frac{f - d_p}{b} \right) z^i \tag{3.49}$$

Quedando la ecuación que determina el valor de cada centroide en cada instante k+1 como muestra (3.49).

Que depende del valor de cada regla, que denominaremos valor activación (para cada combinación de entradas dos a dos), de la diferencia entre la salida y su valor diseñado, y de b, definido en 3.44 como el sumatorio de las activaciones, dicha expresión $(f - d_p)/b$, se conoce como error normalizado, que es realmente el que se propaga hacia atrás en el neurofuzzy.

El valor de activación o regla i-esima se define como:

$$z^i = \prod_{j=1}^{n} e^{-\left\{\left[\frac{\left(x_j - x_j^i\right)}{\sigma_j^i}\right]^2\right\}} \tag{3.50}$$

De forma similar calculamos el valor en cada instante k+1 para el centro de la Gaussiana de cada función de pertenencia en (3.51)

$$x_j^i(k+1) = x_j^i(k) - \alpha \left[\frac{\partial e_p}{\partial x_j^i}\right]_k \tag{3.51}$$

En el caso de x_j^i, la dependencia a la hora de derivar por el sumatorio de activaciones b, definido en (3.44), quedando la derivada como:

$$\frac{\partial e_p}{\partial x_j^i} = (f - d_p)\frac{\partial f}{\partial z^i}\frac{\partial z^i}{\partial x_j^i} = \left(\frac{f - d_p}{b}\right)(y_c^i - f)z^i 2\frac{\left[x_{jp} - x_j^i(k)\right]}{\left(\sigma_j^i\right)^2} \tag{3.52}$$

Por tanto, el valor del centro de la campana de Gauss en cada instante k+1 es como muestra (3.53)

$$x_j^i(k+1) = x_j^i(k) - \alpha\left(\frac{f - d_p}{b}\right)(y_c^i - f)z^i 2\frac{\left[x_{jp} - x_j^i(k)\right]}{\left(\sigma_j^i\right)^2} \tag{3.53}$$

Expresión que depende al igual que antes de $(f - d_p)/b$, de la activación, y ahora del error entre cada centroide y la salida del FNS. Además, el decremento sufrido depende del propio centro normalizado a cada valor de entrada y al ancho de la Gaussiana.

Nos queda calcular el valor en cada instante k+1 para el ancho de la campana de Gauss de cada función de pertenencia, de forma similar a (3.51) es:

$$\sigma_j^i(k+1) = \sigma_j^i(k) - \alpha \left[\frac{\partial e_p}{\partial \sigma_j^i}\right]_k \tag{3.54}$$

Despejando directamente la derivada, el valor en cada instante k+1 del ancho es:

$$\sigma_j^i(k+1) = \sigma_j^i(k) - \alpha\left(\frac{f - d_p}{b}\right)(y_c^i - f)z^i 2\frac{\left[x_{jp} - x_j^i(k)\right]^2}{\left(\sigma_j^i\right)^3} \tag{3.55}$$

No hemos saltado el paso intermedio realizado en (3.52) ya que, a diferencia del valor de los centros, la segunda derivada parcial (de sigma respecto al error) es igual a la de x_j^i, pero sigma queda elevado a 3.

El proceso de aprendizaje para un vector de entrada x_p, calcula la salida del neurofuzzy, es decir, se parte de valores conocidos. Una vez obtenida la salida, $f = a / b$, se calculan los parámetros, centroide, centro y ancho de la Gaussiana. De forma que queda reflejado el algoritmo de Backpropagation, pues lo que se denominó como error normalizado y el erro entre centros y salida multiplicado por las activaciones $(y_c^i - f)z^i$, que ya se conoce gracias al cálculo de la salida, se lleva "hacia atrás" para calcular estos parámetros.

Este proceso se repite durante varios ciclos para entrenar el neurofuzzy.

3.6.3. Aplicación en el proyecto

En este proyecto, hemos implementado para controlar el DAB un neurofuzzy de tipo Wang, en primer lugar, de 1 entrada, con 7 centroides, y e n segundo lugar un Neurofuzzy de 2 entradas, con el objetivo de capturar la dinámica del convertidor.

Posteriormente, si los resultados son satisfactorios, se pretende implementar distintas estructuras de control, con el fin de demostrar la utilidad del Neurofuzzy.

Para ello y como se explicará más detalladamente en el capítulo 5, en el aprendizaje se pretende que la estructura Wang aprenda la función inversa del convertidor, es decir, a partir de una potencia concreta de referencia, nos devuelva como salida el desfase entre los puentes que habría que aplicar para que el DAB transfiera dicha potencia. Para ello, damos como entrada al convertidor una serie de datos aleatorios de desfases en un rango elegido previamente, dando lugar a una potencia medida en el propio convertidor. Esta

4. IMPLEMENTACIÓN DEL CONVERTIDOR EN SIMULINK

Si los hechos no encajan en la teoría, cambie los hechos.

- Albert Einstein, Padre de la física cuántica -

El propósito de este 4º capitulo, es documentar y explicar de forma detallada, la implementación del convertidor DAB propuesto en el software Simulink, perteneciente a MATLAB.

Dicho convertidor como se tratará ahora sigue la topología prevista en el capítulo 2, y concretamente, para dimensionar el convertidor a simular, hemos utilizado como referencia un DAB que se encuentra montado físicamente en los laboratorios del departamento de Ingeniería Electrónica de la Escuela.

Ya que, propuesto el proyecto, su último objetivo, y que lo hace más interesante para una posible ampliación, es la posibilidad de implementar el neurofuzzy en el DAB real con el objetivo de realizar pruebas y llegar finalmente a controlarlo.

4.1. Convertidor DAB de referencia

Para similar el convertidor, partimos como hemos dicho de uno que está montado en laboratorio, apoyandonos en el documento [14]. El objetivo era simularlo para poder entrenar una red neuronal que pueda controlar el DAB.

Este diseño al que nos referimos está orientado a sistemas distribuidos en hogares o industria, donde existen distintos dispositivos y se quiere implementar una Fuente removable de energia y se quiere crear un Sistema de alimentación ininterrumpida (UPS)

El siguiente esquematico, nos ayudará a explicar la utilidad del diseño y el context donde funciona:

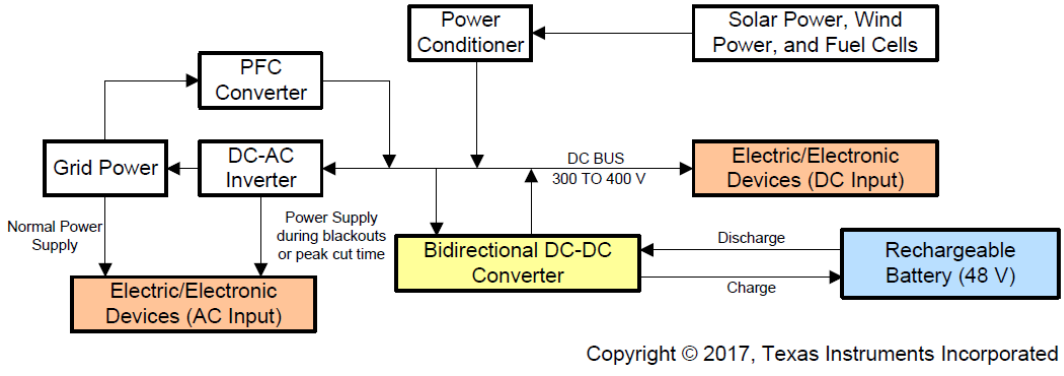


Figura 4-1. Esquemático de un UPS incluyendo el diseño de convertidor bidireccional [14].

Este diseño encaja cuando se parte de que dicha Fuente de energia, por ejemplo un array fotovoltaico nos da una tension DC de salida, dicho array está conectado a un BUS de corriente continua, donde posiblemente tenemos un inversor para devolver a la red la energia Sobrante asi como alimentar la instalación existente de alterna. El convertidor queda conectado entre el BUS y un banco de baterias o de almacenamiento de electricidad, de forma que todo el Sistema en colaboración con el diseño que tenemos, guardar la energia Sobrante en momentos donde la producción es mayor que el consume, y ademas en el caso de que la bateria esté completa, mantener si asi se quiere la tension en el BUS aportando energia.

Por tanto, el convertidor en el primer funcionaria como convertidor reductor, y en el Segundo como elevafor. El DAB, entonces es una pieza clave en sistemas que hoy en dia empiezan a cobrar sentido por el facil acceso a energia fotovoltaica.etc

El diagrama de bloques de este diseño es:

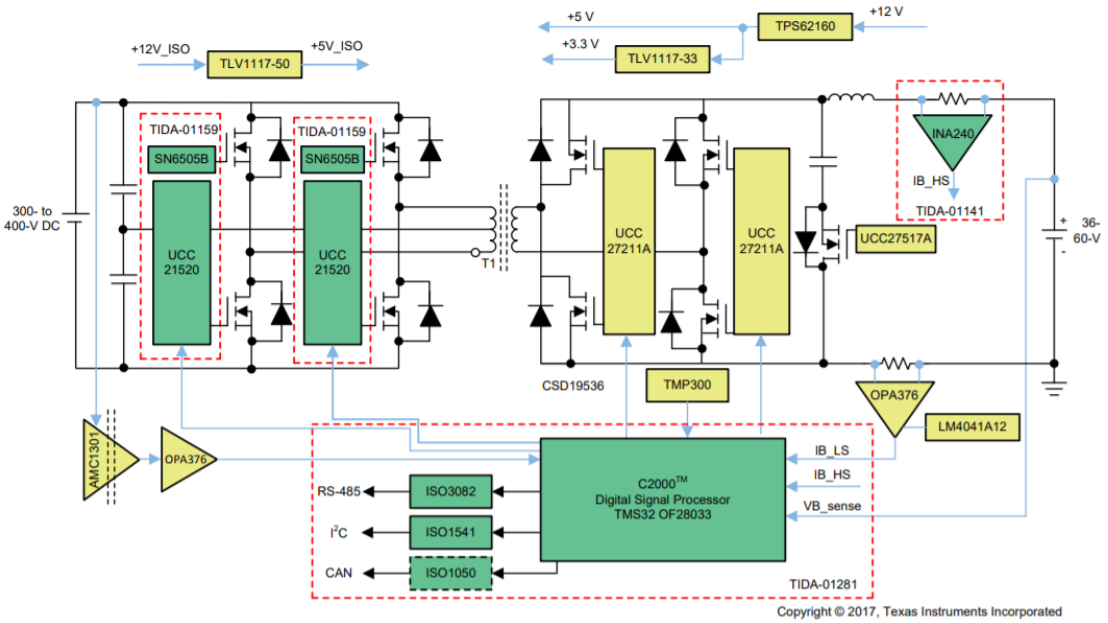


Figura 4-2. Diagrama de bloques del convertidor [14].

Que consta de los siguientes elementos electrónicos:

- ***Fuente de alimentación DC variable 300-400 V:***

  Representa el BUS de corriente continua al que estaría conectado el convertidor, cuyo voltaje puede ser variable debido a las condiciones de funcionamiento del sistema completo.

- ***Puente de entrada o de alta tensión:***

  Se trata de un puente completo formado por 4 MOSFETs de potencia de canal tipo N. Conectado al BUS de DC.

- ***Transformador:***

  Con una relación de transformación aproximada de 1:8.33

- ***Puente de salida o de baja tensión:***

  También se trata de un puente completo formado por 4 MOSFETs de potencia (Vds aprox 100V) de canal tipo N. En este caso está conectado a la batería.

- ***Active clamp:***

  Formado por un condensador y un MOSFET similar al mencionado para los puentes, cuya función es actuar como snubber, de forma que se facilita la conmutación ZVS.

- ***Fuente de alimentación DC variable 36-60V:***

  Representa a una batería cuya tensión nominal es de 48V aproximadamente y una intensidad máxima de 16A.

  El diseño puede funcionar como dijimos tanto de elevador (Boost) como reductor (Buck), en el primer caso funciona como un convertidor Voltage-Fed, el segundo como Current-Fed.

El resto de los bloques del diseño, para el caso por ahora no nos interesan. Son distintos dispositivos que permiten el control del dispositivo, sensores, drivers, amplificadores. Por otro lado, tenemos el microprocesador que recibe las señales de los sensores y ejecuta la modulación PWM. En un capítulo posterior se tratará la implementación y las posibles ampliaciones del estudio.

4.2. Implementación en Simulink

En este apartado, vamos a explicar el proceso que ha supuesto implementar el convertidor asi como su control mediante un PI clasico.

4.2.1. Esquemático

Para simular el convertidor (figura 4-3), hemos creado los siguientes bloques/subsistemas:

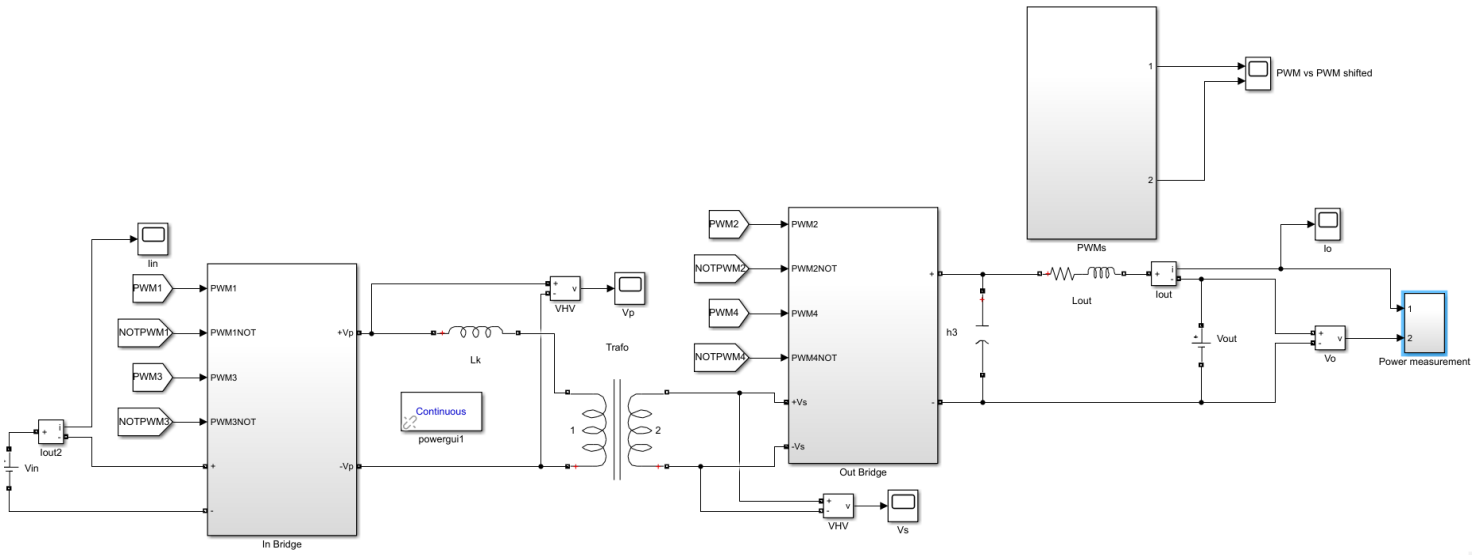


Figura 4-3. Esquema convertidor implementado en Simulink.

4.2.1.1 Puentes de transistores

Para cada puente hemos utilizado el bloque de MOSFET que trae Simulink, los parámetros que hemos modificado, han sido “FET resistance Ron”, que como se indica en el datasheet [15] de los MOSFET usados en [14], su valor típico para Vgs=6V es 2.4Ω, y el “Internal diode forward voltage Vf”, que de igual manera, el datasheet nos fija 1.1V como valor máximo.

Debido a que en [14] no se indica, se han fijado los mismo parámetros para los MOSFETs del puente de alta tensión (HV). Esto se ha decidido así, ya que los parámetros que se han modificado dependen del voltaje de la puerta (en caso de Ron) o no afecta a la decisión (Vf).

Se muestra en la figura 4-4 el esquemático del puente del lado de baja tensión.

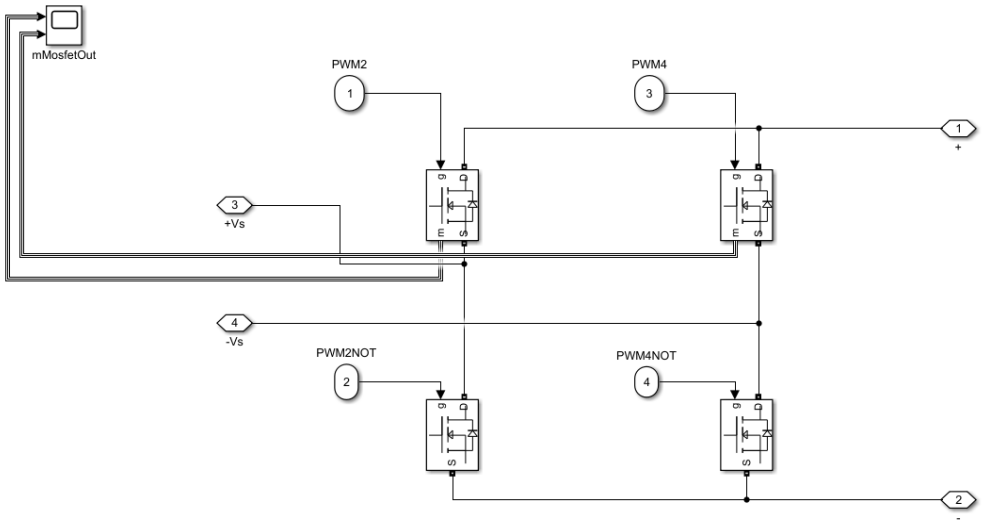


Figura 4-4. Puente de transistores del lado de baja tensión.

4.2.1.2 Transformador

Hemos elegido un transformador de dos devanados a diferencia del que se muestra en el diseño de la figura

4-2. Además, para simular la inductancia interna equivalente del mismo, se colocó una a la entrada, que hemos denominado Lk, cuyo valor se ha fijado en 5 μF .

Los valores del trasformador escogidos para el bloque ("Linear transformer") son:

1. Frecuencia: 100 kHz
2. Relación de transformación: 400/48
3. Potencia nominal: 2.3Kw
4. Resistencia de magnetización: $1x10^6$
5. Inductancia de magnetización: ∞

4.2.1.3 Señales de disparo de los transistores

Como vimos en el apartado 2.1, en los DAB, la frecuencia de conmutación toma valores altos, del orden de decenas de kHz o más, en nuestro caso la frecuencia como hemos mencionado para el transformador es 100kHz, que será la frecuencia de las ondas cuadradas que usaremos como modulación. El duty cicle queda fijo e igual a 0.5, por ese motivo, aunque hablamos de PWMs, ya que el ancho del pulso no variará en las simulaciones, debido a que el parámetro de interés es el desfase entre puentes, usaremos otros términos.

Para generar la onda cuadrada que será la que se desfase respecto a una de referencia, cuyo desfase pueda ser modificado en tiempo de simulación, se ha creado un m-function, que recibe como parámetros, la frecuencia en Hz, el tiempo de simulación y el desfase phi.

La figura 4-5 muestra la implementación de los PWMs:

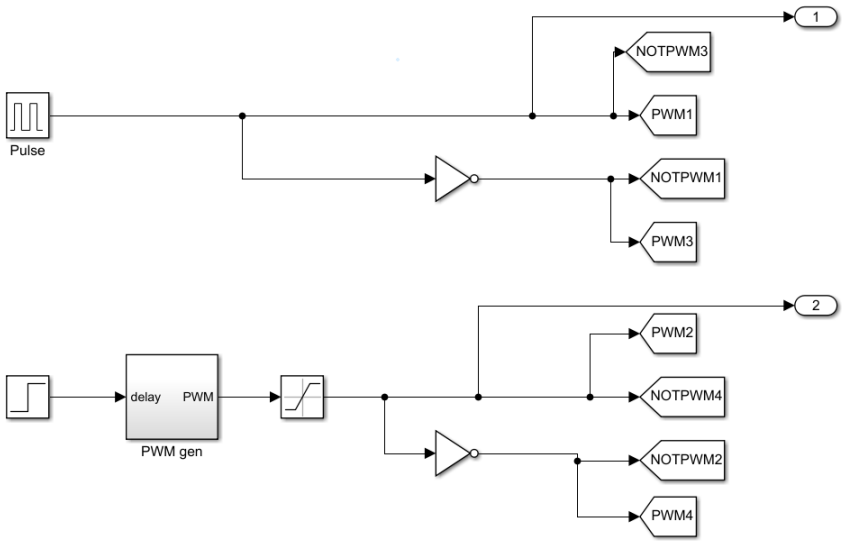


Figura 4-5. Subsistema "PWMs".

Como puede verse, hay dos ondas independientes, siendo la de la parte de abajo la que se desfasa y que sirve a los MOSFET del puente de lado de baja tensión.

De forma resumida, si tomamos como referencia la primera rama (izquierda) del puente del lado de alta tensión, cada rama de la derecha de cada puente recibe la misma onda que la rama izquierda pero desfasada 180º (equivalente a la operación NOT). Por otro lado, el puente de la derecha recibe las mismas ondas que el lado izquierdo, pero desfasadas un ángulo phi.

4.2.1.4 Resto de elementos.

A la entrada como a la salida, para empezar, usaremos dos fuentes de alimentación DC, bloque "DC Voltage Source" de valor 400V en el lado de alta tensión, y 48V en el lado de baja.

Como estabilizador de la tensión en el lado de baja, un condensador de valor 1mF.

Además, para poder medir la intensidad correctamente y suavizar la corriente de salida, se ha puesto una impedancia RL de valor $0.1\,\Omega$ y $4.5\,\mu F$

4.2.2. Control PI

Para controlar la potencia que el DAB transfiere de un lado a otro, se ha usado un controlador PI clásico y discreto. El rango de control que se ha fijado es de -2.4 kW a 2.4kW, que equivalen a -0.05776 rad y 0.05812 rad (no es totalmente simétrico).

Para hallar los valores de las constantes integral y proporcional adecuados, se ha utilizado la herramienta "PID tunning" que incorpora Simulink, ya que el diseño por otros métodos clásicos no era satisfactorio y el diseño del control no es el aspecto más importante.

Hemos obtenido dos funciones de transferencia (4.1) distintas sometiendo a la planta a una señal escalón, la primera (G1) asumiendo que el convertidor se comporta como un sistema de primer orden, y otra (G2) como de segundo orden. Llevando estas funciones al "tunning", obtenemos las constantes de control para un PI (4.2), y quedan fijados el tiempo de subida y la agresividad de cada control (4.3)

La conclusión es que, una vez probado cada controlador, detectamos que uno de ellos sobreoscila ligeramente y es más rápido, y otro es más tímido en la respuesta, y por tanto tarda más. Posteriormente en 4.2.3.2 analizaremos mejor el resultado.

$$G_1 = \frac{4.1328x10^4}{0.0001s+1} \tag{4.1}$$

$$G_2 = \frac{1.085x10^{13}}{s^2 + 2.437x10^4 s + 2.626xe^8} \tag{4.2}$$

$$\begin{aligned} Ki_1 &= 2.5401x10^{-5} \\ Kp_1 &= 0.2542 \end{aligned} \tag{4.3}$$

$$\begin{aligned} Ki_2 &= 1.715628^{-5} \\ Kp_2 &= 0.171562 \end{aligned} \tag{4.4}$$

$$T_{s1} = 0.00019, \delta_1 = 0.9 \tag{4.5}$$

$$T_{s2} = 0.0001821, \delta_2 = 0.8 \tag{4.6}$$

4.2.3. Resultados de la simulación

En este apartado, mostraremos las gráficas de interés que reflejan el modo de funcionamiento del convertidor, así como que lo hace correctamente. Luego analizaremos brevemente el control PI mencionado antes.

4.2.3.1. Funcionamiento DAB

Teniendo en cuenta la topología del DAB implementado, se espera en primer lugar, que se rija por la formula obtenida en el apartado 2.1 de este proyecto (denominada entonces expresión 2.10) que refrescamos:

$$P_o = \frac{V_i^2}{\omega L} d\phi \left[1 - \frac{\phi}{\pi} \right] \tag{4.7}$$

Recordemos que dicha expresión depende del desfase aplicado entre puentes (en radianes) y la tensión de entrada v_i que puede variar.

A continuación, en la figura 4-6, mostramos la relación entre la expresión (2.10), la cual se ha implementado en Simulink con bloques, para poder compararla fácilmente con la verdadera potencia de salida del DAB y la salida. Para ello se ha proporcionado un desfase de 0.0 rad, seguido de uno de 0.0556 equivalente a una potencia de 2.3kW y otro de -0.0119, equivalente a -0.5kW.

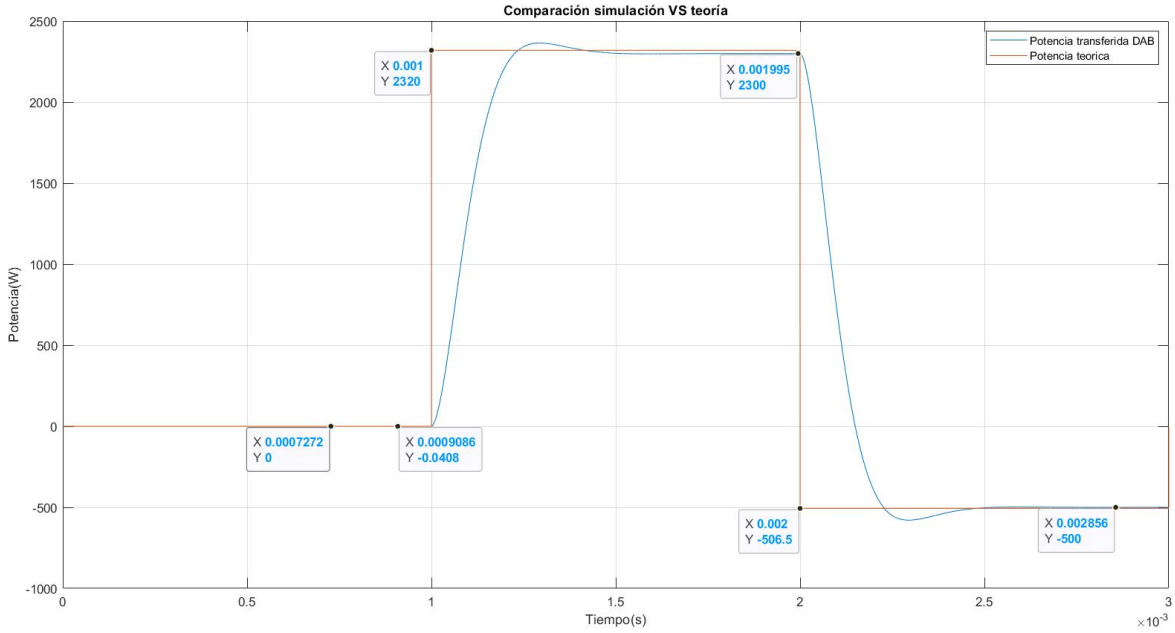


Figura 4-6. Potencia teórica VS potencia a la salida del DAB.

Mediante marcadores que hemos colocado para ver es el valor exacto de la potencia real, podemos comprobar como de fiel es la implementación del convertidor respecto a la expresión de la potencia que conocemos. Por tanto, podemos comprobar que es correcto el montaje, y el posible error que existe ni siquiera es importante, ya que teniendo en cuenta que es una simulación aproximada, no se pretende una precisión absoluta.

Ahora mostraremos la evolución de la corriente:

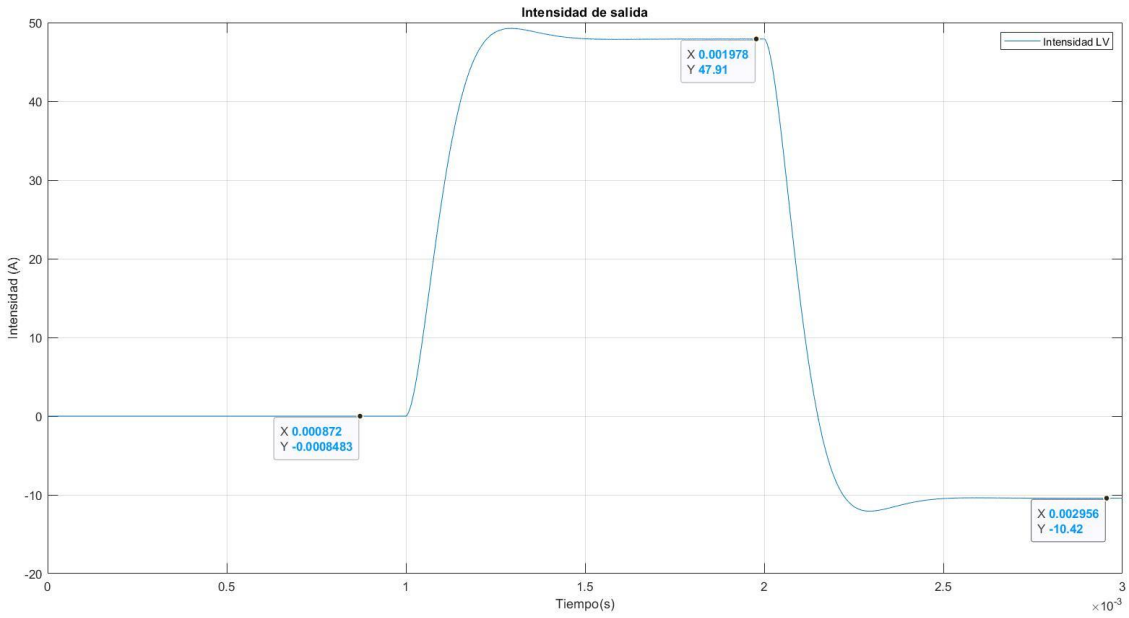


Figura 4-7. Evolución de la intensidad en el lado de baja tensión.

Dicha figura de la intensidad nos demuestra el funcionamiento bidireccional del convertidor, la corriente para un desfase negativo (o que la onda de referencia es la que está desfasada de la otra), arroja valores negativos.

Funcionando a la potencia nominal, en torno a 2.3kW, la intensidad alcanza unos 50 A.

Para comprobar que se existe desfase entre las ondas cuadradas que le damos a los MOSFET, tenemos la figura 4-8. Dicha figura comprende 30us.

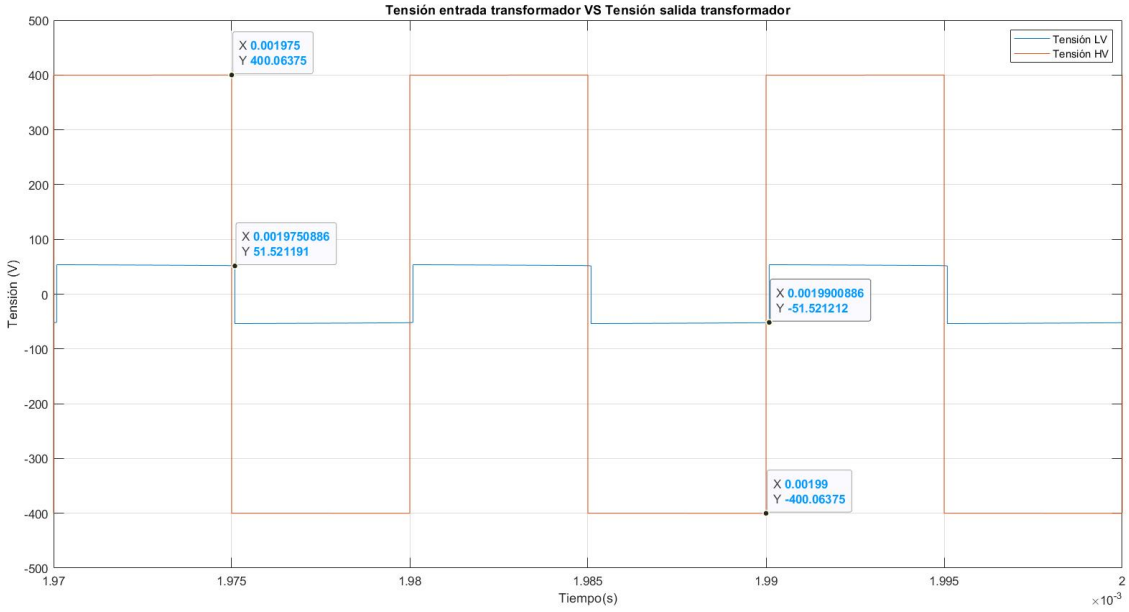


Figura 4-8. Tensión de entrada al transformados frente a la de salida de este.

Dicha gráfica muestra los voltajes Vp y Vs, en un periodo de 30us, momento en el que se está aplicando un desfase de 0.0556 rad como dijimos.

Primero debemos comprobar que la frecuencia de 100kHz es correcta, si calculamos el tiempo que transcurre entre dos puntos de la onda cuadrada tenemos que:

T=0.00198-0.00199=0.00001, f=1/T, obtenemos 100kHz que es correcto.

La diferencia de tiempo que arrojan los marcadores colocados, entre la onda de referencia (naranja) y la onda desfasada (azul) es $0.00199 - 0.0019900886 = 0.0000000886s = 8.86x10^{-8}\,s$

Si pasamos el desfase a unidades de tiempo mediante la frecuencia (100kHz), $t_\varphi = \frac{\varphi}{2\pi f} = \frac{0.0556}{2\pi 100x10^3} = 8.849x10^{-8}\,s$, que es prácticamente igual tiempo medido.

Igualmente podemos ver que los valores de las tensiones son correctos, 400V y 48V aproximadamente.

La siguiente figura, muestra la corriente de drenador (Id) y la tensión drenador-sustrato (Vgs) que soportan los MOSFETs del puente de alta tensión:

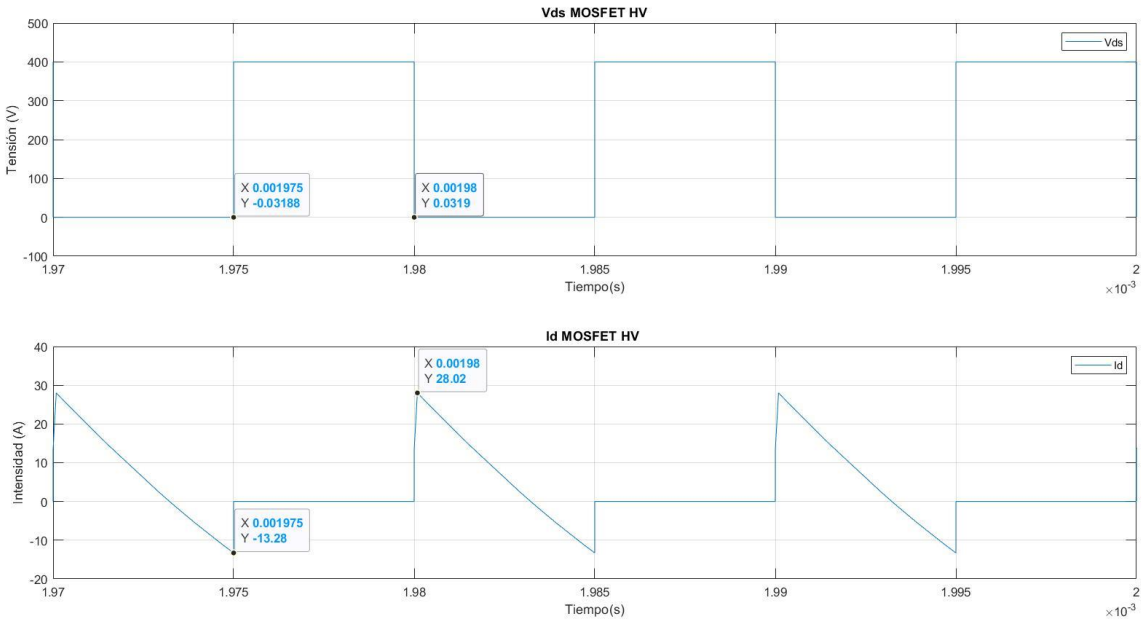


Figura 4-9. Tensión Vds e intensidad Id de un MOSFET del lado HV.

Podemos observar que, por un lado, la corriente toma como valores extremos 28A y -13A en este caso.

Y la tensión 400V correspondiente a la tensión de entrada.

Y la misma gráfica, pero de un MOSFET del puente de salida:

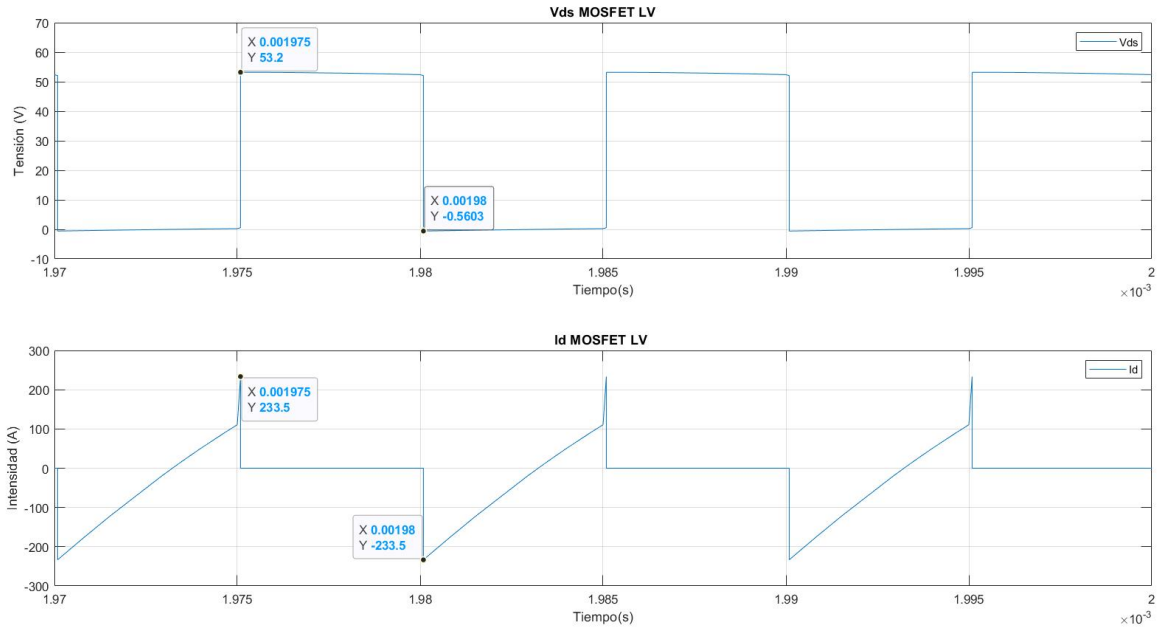


Figura 4-10. Tensión Vds e Intensidad Id de un MOSFET del lado LV.

En el lado LV, la tensión como era esperable toma un valor aproximado de 50V y la intensidad oscila entre los -200A y 230A.

4.2.3.2. Control en bucle cerrado PI

Se ha simulado el esquema ya presentado introduciendo un bloque PI discreto que es ahora quien da el desfase en cada instante al DAB. Se han probado los dos controladores que se mencionaron, cuya entrada (Potencia de referencia) es una escalera de valores recorriendo todo el rango del convertidor. Esto luego nos será útil para conocer el valor exacto de cada centroide en el neurofuzzy.

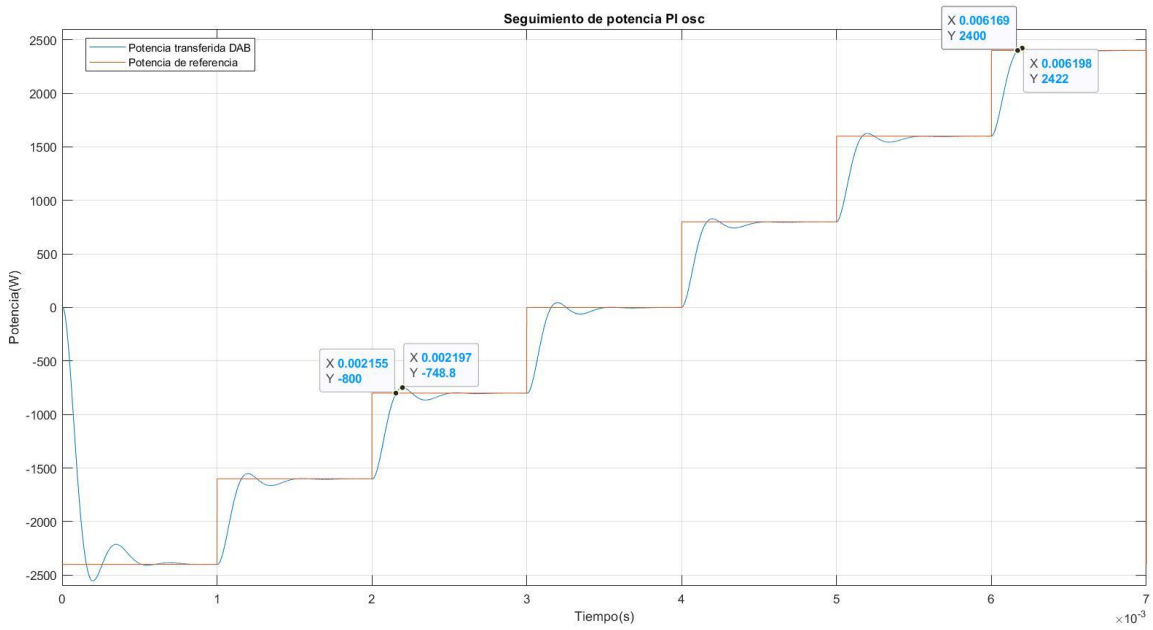


Figura 4-11. Seguimiento de la potencia de referencia PI "OSC".

La figura 4-11, muestra el seguimiento que hace el convertidor de la potencia de referencia que se introduce al PI. Hemos calculado la sobreoscilación (SO%), en cada escalón, así como el tiempo de subida.

Siendo SO%, la diferencia del valor de pico y el valor de régimen permanente relativa al propio valor de régimen permanente. Y siendo el tiempo de subida, el que tarda la respuesta es alcanzar la primera vez el valor de régimen permanente desde que comienza su evolución.

A partir de los marcadores de la gráfica, que corresponden con los valores máximos y mínimos de SO y Ts, obtenemos dichos valores:

$T_{s\min} = 15.5\text{ms}$ (Pref = -800W)

$T_{s\max} = 16.9\text{ms}$ (Pref = 2400W)

$SO_{\min}(\%) = 0.8\%$ (Pref= 2400W)

$SO_{\max}(\%) = 6.8\%$ (Pref= -800W)

Correspondiendo el valor mínimo del tiempo de subida con el escalón donde se dá la máxima SO y viceversa.

Como explicamos en el apartado 4.2.2, habíamos denominado uno de los controladores, algo más agresivo y rápido, lo que lleva a que se produzca SO, se trata de este, correspondiente a las constantes de la expresión 4.3.

En la siguiente figura (Figura 4-12) se muestra la evolución de la salida del PI, desfase (rad), los valores se han señalado con marcadores como dijimos ya que serán útiles.

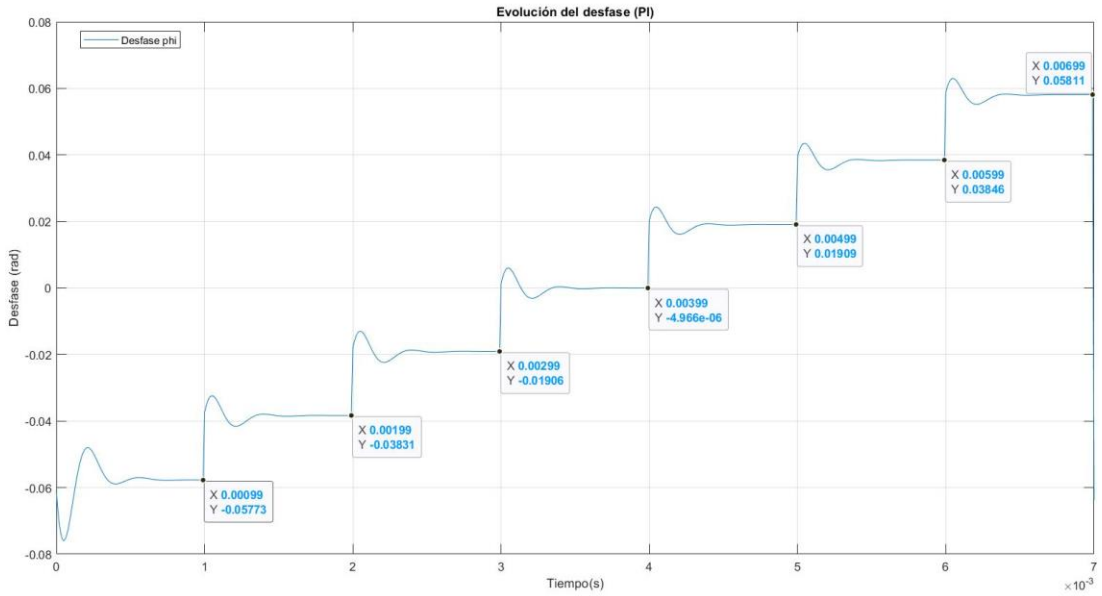


Figura 4-12. Evolución desfase proporcionado por PI "osc".

A continuación, el seguimiento de la potencia de referencia, usando el controlador de la expresión (4.1) cuyas constantes son las de la expresión (4.4):

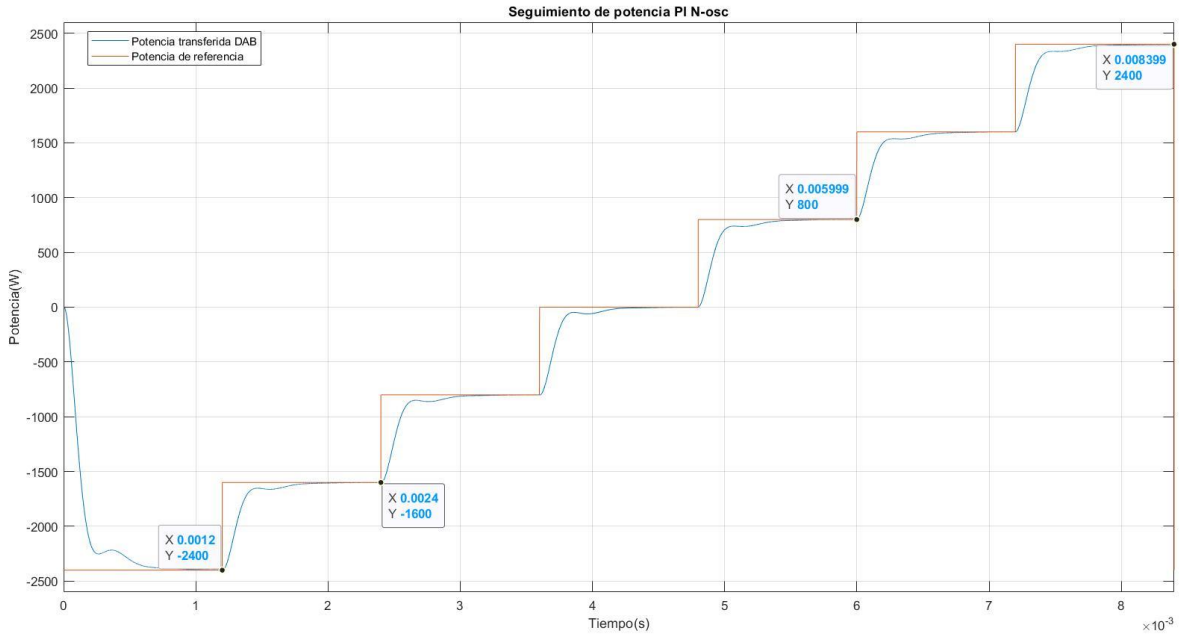


Figura 4-13. Seguimiento de potencia PI "Nosc".

Y por último la evolución del desfase con el mismo PI:

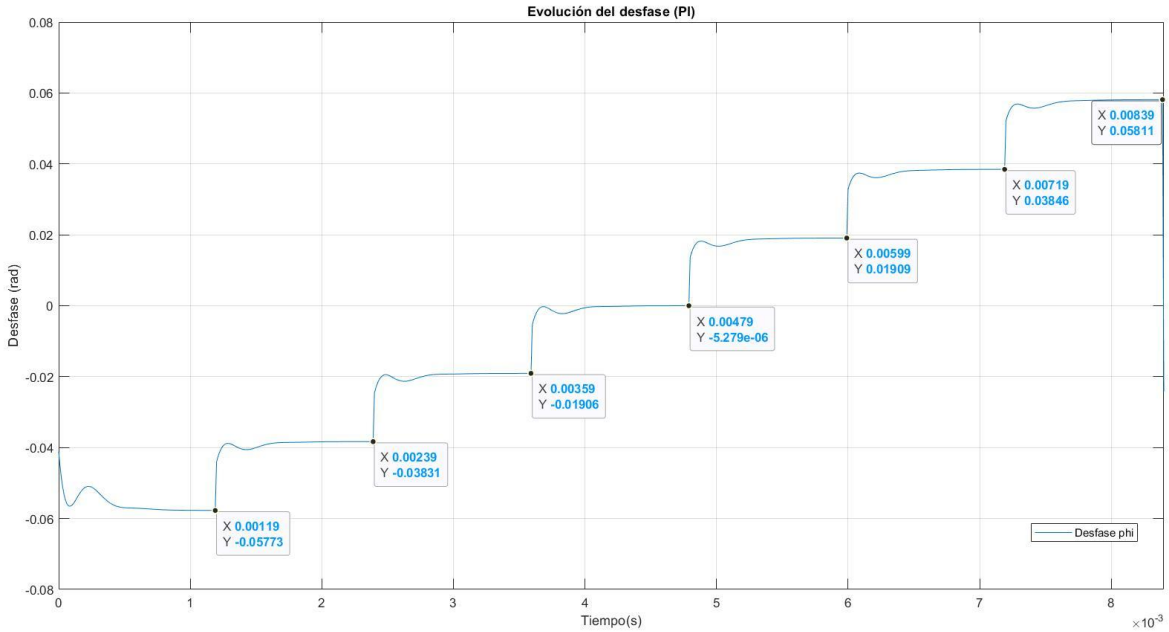


Figura 4-14. Evolución desfase proporcionado por PI "Nosc".

Hasta aquí las simulaciones básicas del convertidor, si se quisiese, podrían probarse otros experimentos, por ejemplo, cargando una batería en el lado LV tal y como reza la principal aplicación de este convertidor. Además de variar el voltaje de entrada simulando oscilaciones en el BUS dc.

En el próximo capitulo se analizarán las simulaciones referidas al aprendizaje del neurofuzzy, así como las estructuras usadas.

5. IMPLEMENTACIÓN DE LA ESTRUCTURA NEUROFUZZY EN SIMULINK

Lo único que importa es el esfuerzo.

- Antoine de Saint-Exupery, autor de "El principito" -

En este capítulo, se implementa el Neurofuzzy de tipo Wang visto en el capítulo 3 y se procede a realizar el aprendizaje de la función inversa del DAB con dicha estructura. Posteriormente se prueban varias estrategias de control ANNs

5.1. Implementación en Simulink

Para poder simular el neurofuzzy y llevar a cabo el entrenamiento, se ha utilizado un bloque "S-function" que nos permite usar lenguaje C para simulaciones dentro del propio Simulink. Además, nos da ventaja a la hora llevar el control resultante al microcontrolador. Dicho código se divide en 2 partes:

1) Un primer bucle for, que calcula z_i a partir de los valores de potencia del convertidor.

2) Un segundo bucle "FOR" que recalcula los parámetros y^i_c, σ y x (los centroides, el ancho de la campana de Gauss de cada función de pertenencia y el centro de la campana respectivamente), a partir del error en desfase.

En cada tiempo de muestreo de la S-Function, se enseña un dato de potencia al algoritmo de aprendizaje y a su vez el equivalente en potencia dado por el convertidor.

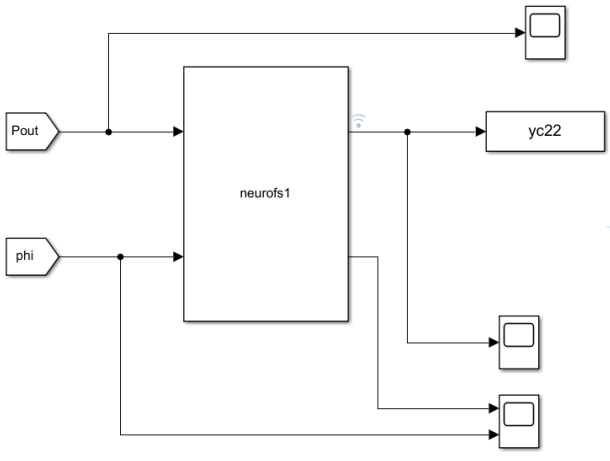


Figura 5-1. Esquemático del implementación del Neurofuzzy (Simulink).

5.2. Estructuras de aprendizaje

5.2.1. Estructura Aprendizaje Neurofuzzy 1 entrada

El primer esquema utilizado en las simulaciones se muestra en la figura 5-1, el objetivo es identificar la ANN inversa, es decir, entrenar una red neuronal que aprenda la función inversa del convertidor $f = \varphi(P)$.

El procedimiento es el siguiente:

1. Se proporciona como entrada al Neurofuzzy una serie de valores de potencias $x(k)$ que vienen del convertidor, correspondientes a unos valores de desfase $u_c(k)$ que se usarán durante el entrenamiento (salida deseada de la ANN).
2. Se calcula la salida $u_c^{'}$ (desfase), y con dicho valor calculamos el error en desfase que se propaga hacia atrás en el neurofuzzy, calculando (en el siguiente paso de integración) los centroides, el centro o media de la Gaussiana y la desviación o sigma.
3. Iterativamente se van mostrando datos que proporcionan un error que hay que propagar de tal forma que los parámetros acaben convergiendo en unos valores al cabo de varios set de entrada.

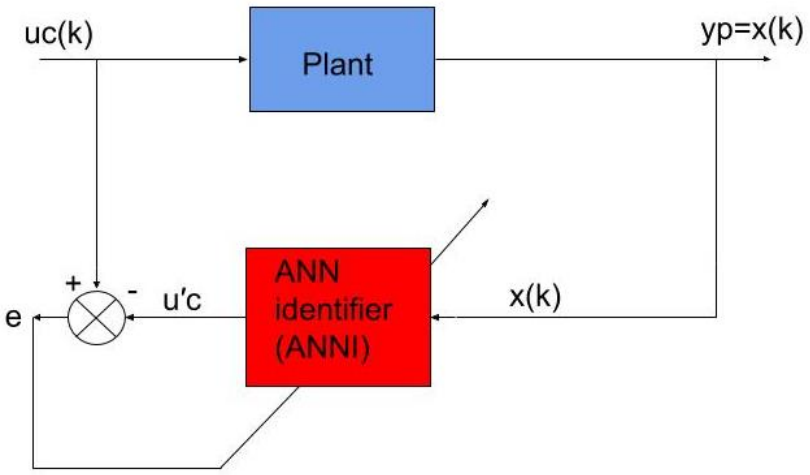


Figura 5-2. Esquema de identificación ANN.

5.2.2 Simulaciones neurofuzzy 1 entrada

Hemos realizado las siguientes simulaciones/entrenamientos, cada uno con una características que se detallarán.

5.2.2.1. Aprendizaje de los centroides con μ fija y σ =500 W

La primera simulación que probamos es la más sencilla y su objetivo es comprobar el buen funcionamiento del aprendizaje, fijando los parámetros de las distintas funciones de pertenencia y partiendo de 0 con los 7 centroides.

Como hemos explicado en el apartado el rango de aprendizaje elegido es desde -2900 a 2900 kW y se divide en 7 centroides, lo que dejando equidistantes las campanas, se define el vector de medias de las campanas como:

$$\mu = [-2400, -1600, -800, 0, 800, 1600, 2400] \text{ (W)} \qquad (5.1)$$

Y de la misma manera el vector de los anchos de las campanas:

$$\sigma = [300, 300, 300, 300, 300, 300, 300] \text{ (W)} \qquad (5.2)$$

Por tanto, de esta manera la distribución de campanas de Gauss en el rango de actuación de la potencia se muestra en la figura 5-2.

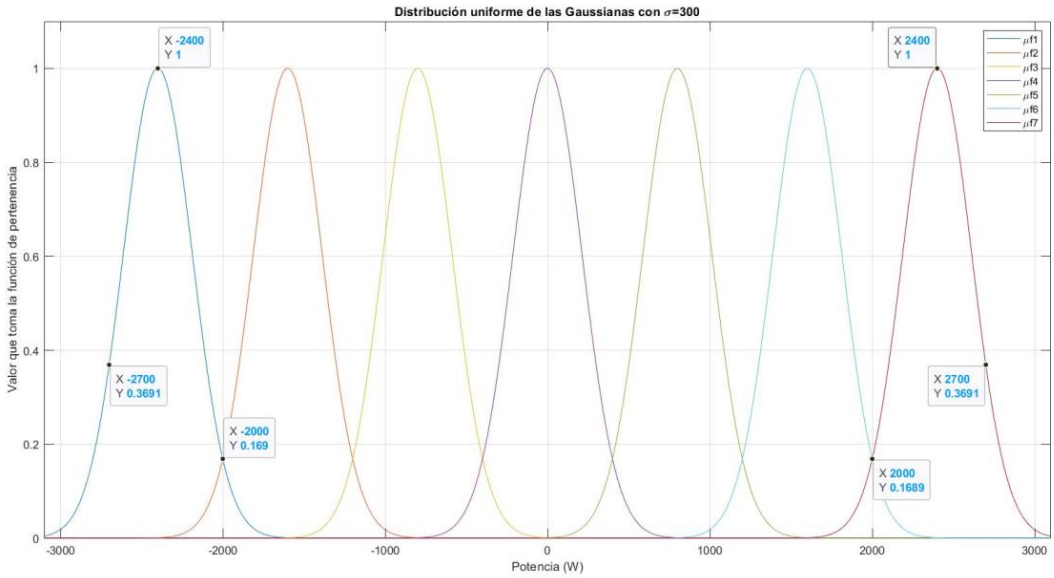


Figura 5-3. Distribución de las campanas de Gauss con Sigma=500.

Con estos valores iniciamos el aprendizaje exclusivo de los centroides denominados y_c^i

Tras 3 simulaciones, la evolución de los centroides ha sido la siguiente:

Tabla 5-1 Evolución de los centroides Sigma=500

Centroides (rad)	y_c^1	y_c^2	y_c^3	y_c^4	y_c^5	y_c^6	y_c^7
1	-0.0630933	-0.0287328	-0.0199558	-0.0010076	0.0195680	0.0311421	0.0002288
2	-0.0577295	-0.0359762	-0.0200372	0.0011926	0.0180272	0.0397233	0.0569398
3	-0.0612917	-0.0359698	-0.0180567	0.0017173	0.0211484	0.0352904	0.0593713

Las gráficas asociadas al proceso se detallan a continuación con cada experimento:

- ***Experimento 1:***

  Para este experimento se ha utilizado un paso de aprendizaje $\alpha = 0.007$ y 20 datos de entrada.

  En esta primera aproximación, se han obtenido las siguientes gráficas.

  En la figura 5-4 se muestra la evolución de los centroides.

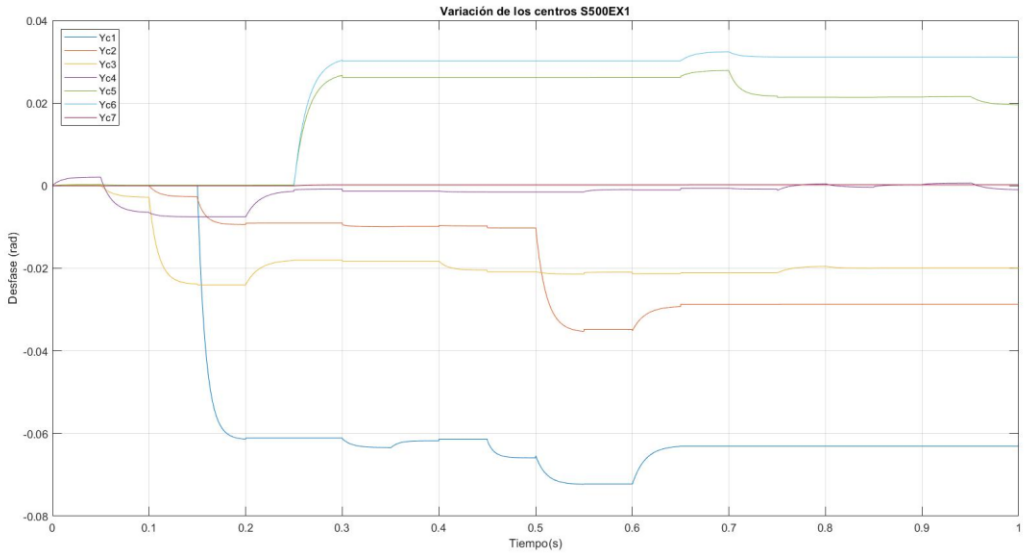


Figura 5-4. Evolución centroides Sigma=500-Exp 1.

Apoyándonos en la tabla y la figura, podemos ver que el experimento 1 sirve para colocar los centros a lo largo del rango sin mucha precisión. Los valores que hemos obtenidos muestran que fácilmente con un numero de datos pequeño, algunos centros pueden posicionar su valor en un rango correcto.

- ***Experimento 2:***

  Para este experimento se ha utilizado un paso de aprendizaje $\alpha = 0.07$ y se ha duplicado el número de datos.

  En la figura 5-6 se muestra la evolución de los centroides.

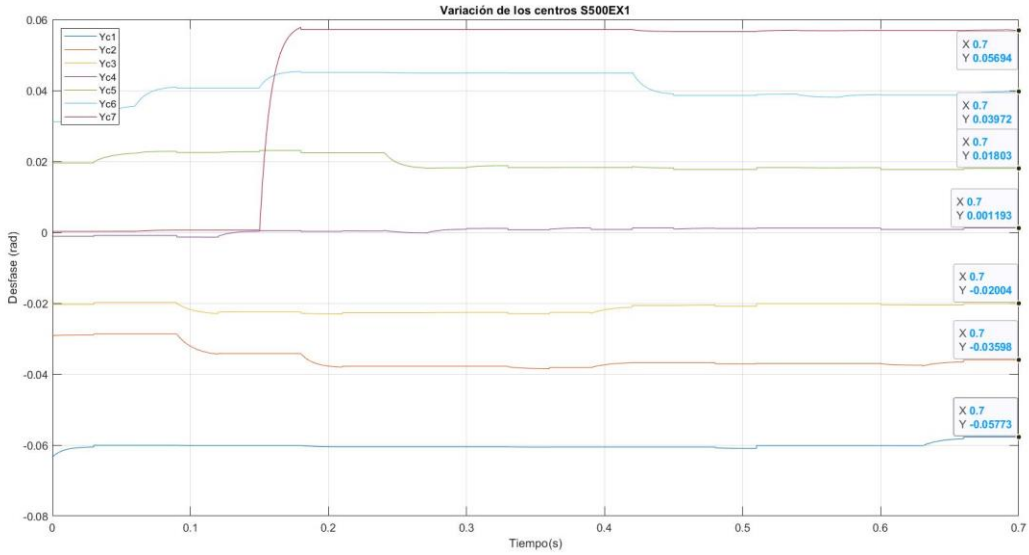


Figura 5-5. Evolución centroides Sigma=500-Exp 2.

Y en la figura 5-7 se encuentra la evolución de la salida del neurofuzzy que se corresponde con el desfase (rad) determinado en cada ajuste de lo centroides.

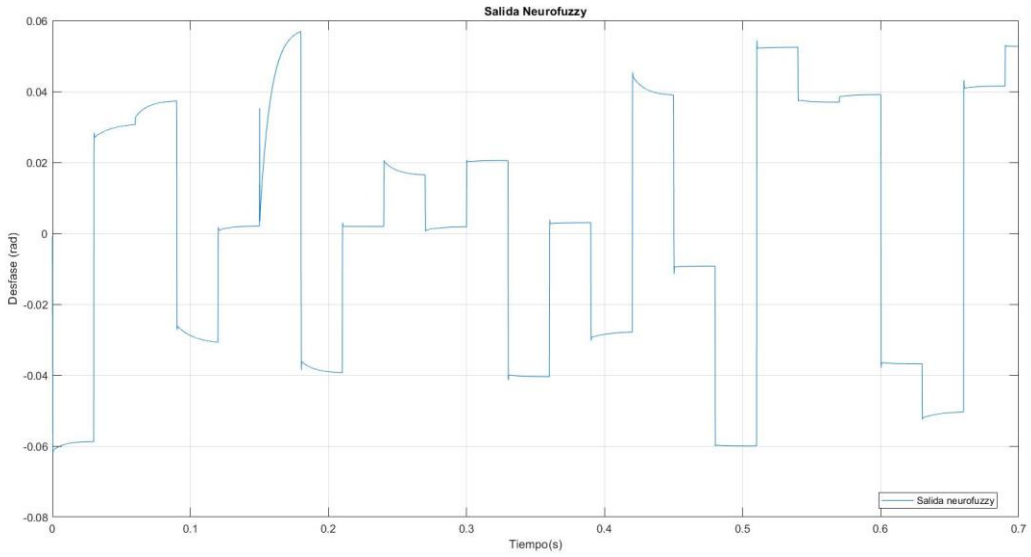


Figura 5-6. Evolución salida neurofuzzy Sigma=500-Exp 2.

El segundo experimento, permite que los valores de los centros se ajusten con precisión y que como nos muestra la figura 5-6, el aprendizaje es suave y permite al neurofuzzy converger a valores de los centroides con mejor rendimiento que el primer experimento.

- ***Experimento 3:***

  Para este experimento se ha utilizado un paso de aprendizaje $\alpha = 0.007$, se mantiene el número de datos del experimento 2.

  En la figura 5-7 se muestra la evolución de los centroides.

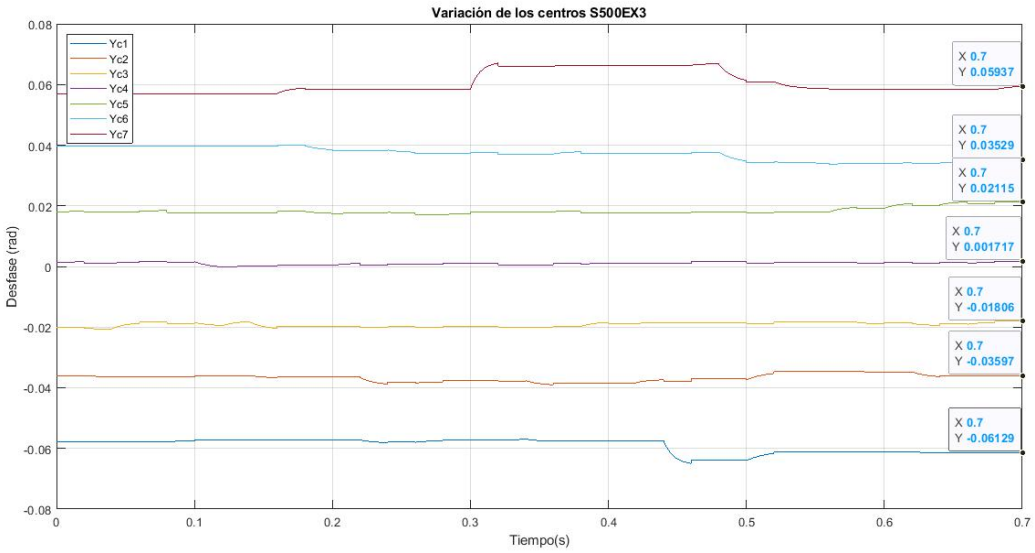


Figura 5-7. Evolución centroides Sigma=500-Exp 3.

En este último experimento, ya no se aprecia una mejora significativa, los centroides se mantienen estables, excepto 1, que se modifica ligeramente. Digamos que cuando se estabilizan los valores, solo queda que alguno de los que no ha podido converger tan bien durante el proceso, lo haga en este último golpe.

5.2.2.2. Aprendizaje de los centroides con μ fija y σ =300

Ahora se han repetido los experimentos del anterior apartado 5.2.2.1, pero usando como ancho de las funciones de pertenencia un valor bastante más pequeño, que respecto al usado antes (500) se considera que individualiza aún más la parte del rango de la señal que se reparte cada centroide. Inicialmente los centroides parten de 0.

Por tanto, el vector de anchos utilizado:

$$\sigma = [300, 300, 300, 300, 300, 300, 300] \qquad (5.3)$$

De la misma manera la distribución de campanas de Gauss en el rango de actuación de la potencia se muestra en la figura 5-8.

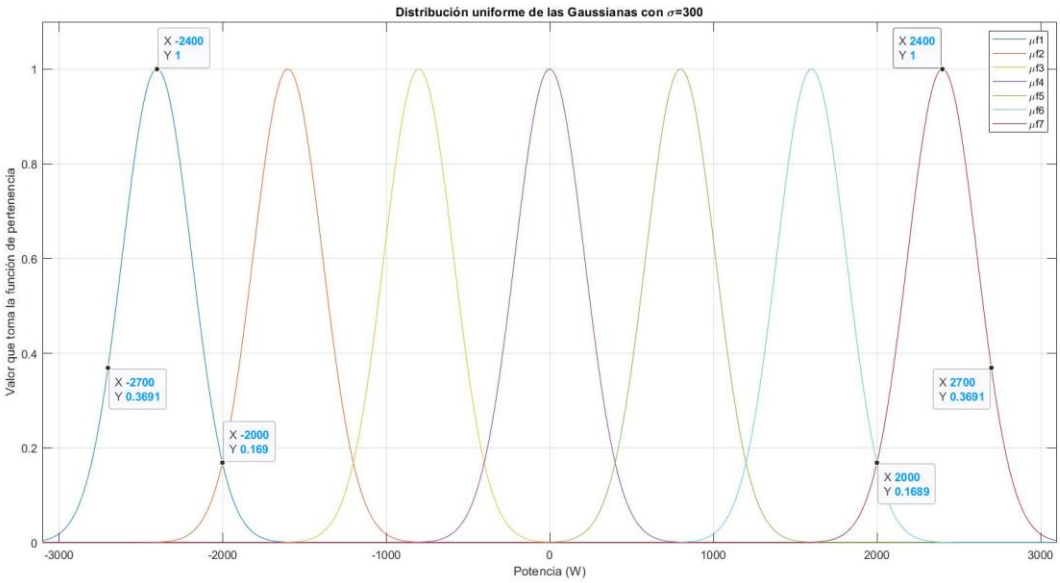


Figura 5-8. Distribución de las campanas de Gauss con Sigma=300.

Tras las simulaciones realizadas, que se detallaran, los valores obtenidos de los centroides han sido:

Tabla 5-2. Evolución centroides Sigma=300.

Centroides (rad)	y_c^1	y_c^2	y_c^3	y_c^4	y_c^5	y_c^6	y_c^7
1	-0.05826235	-0.0327218	-0.0195204	0.000729529	0.0217278	0.0365697	0.06177312
2	-0.0588023	-0.0402563	-0.0199136	0.00538925	0.0145763	0.0368283	0.0601946
3	-0.0585449	-0.0426605	-0.0150259	-0.0068512	0.0201516	0.0362181	0.0659072

Las gráficas asociadas al proceso se detallan a continuación:

- ***Experimento 1:***

  Para este experimento se ha utilizado un paso de aprendizaje $\alpha = 0.008$

  En la figura 5.8 se muestra la evolución de los centroides.

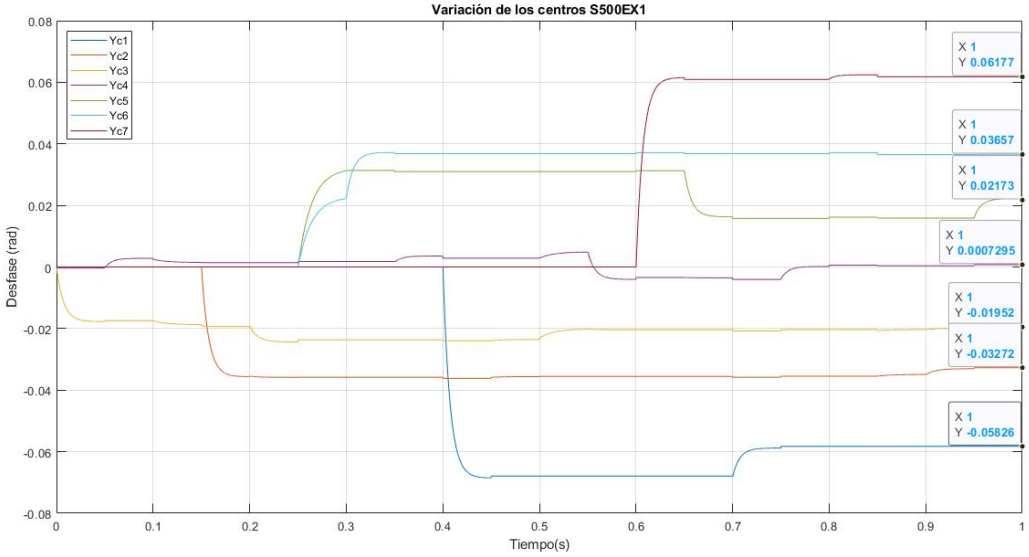
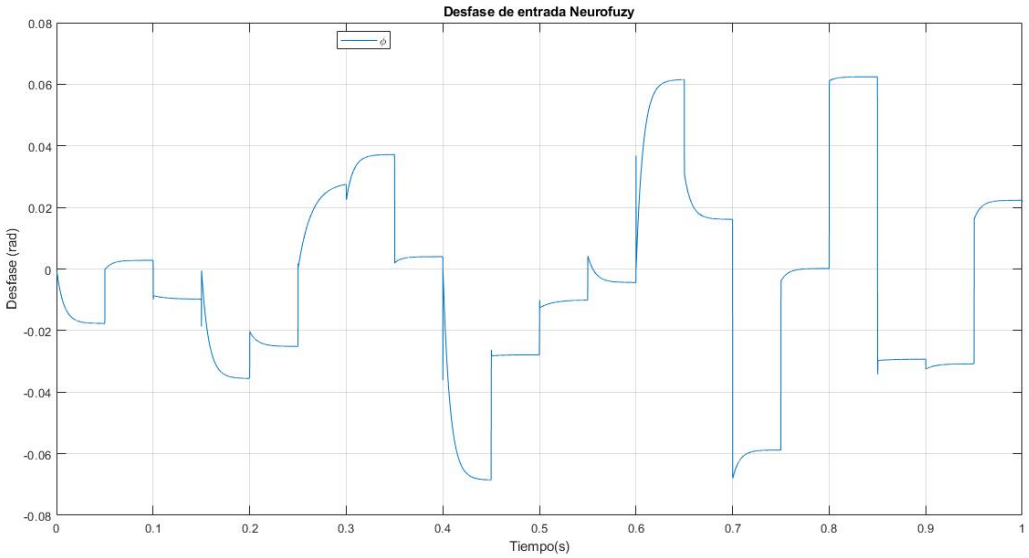


Figura 5-9. Evolución de los centroides Sigma=300-Exp 1.

En este caso, en el primer experimento, los valores de los centroides acaban con valores bastante distribuidos.

En la figura 5-10, tenemos la salida del neurofuzzy a lo largo del aprendizaje. Que indica, que, en la mayoría de los datos de entrada, la convergencia es suave, solo en aquellas diferencias entre datos grandes (1/2 del rango, por ejemplo) el algoritmo se ve obligado a llevar más velocidad, podemos darnos cuenta entre las distintas figuras que se han colocado que muestran la evolución del desfase de diseño, que inicialmente la convergencia es rápida, y a medida que el erro se reduce, se hace mucho más lenta la aproximación al valor.



F*igura 5-10. Evolución del desfase de diseño Sigma=300-Exp 1.*

- ***Experimento 2:***

  Para este experimento se ha utilizado un paso de aprendizaje $\alpha = 0.008$, se mantiene el número de datos respecto al experimento 1.

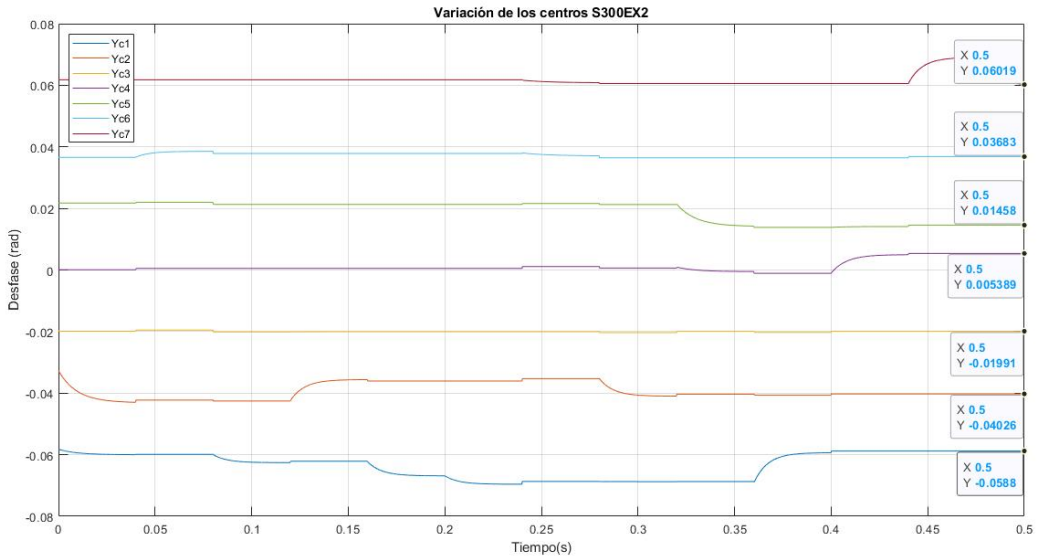


Figura 5-11. Evolución de los centroides Sigma=300-Exp 2.

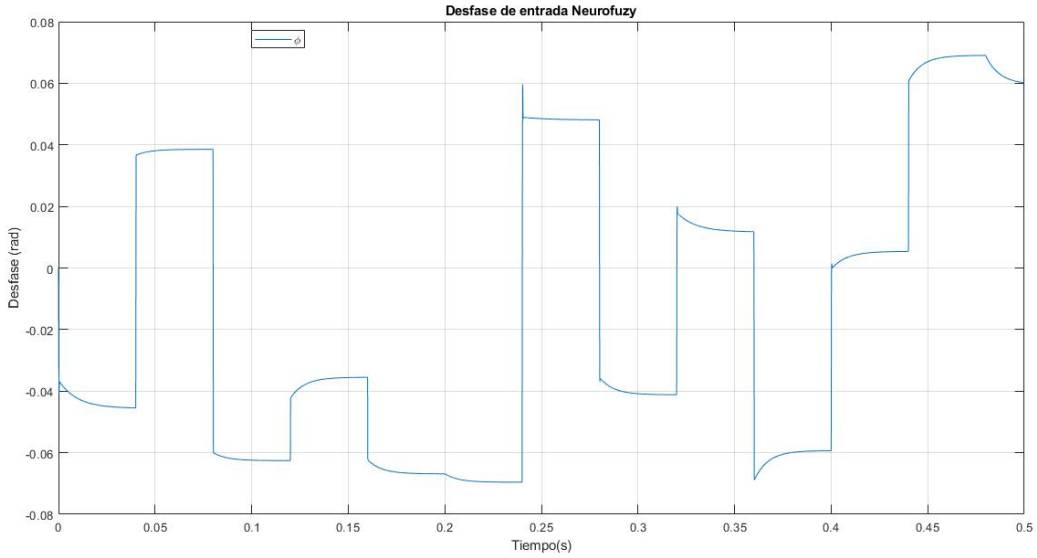


Figura 5-12. Evolución del desfase de diseño Sigma=300-Exp 2.

En el experimento 2, como ya se habían alcanzado valores de los centros bien distribuidos la primera vez, este entrenamiento ha servido para acabar de estabilizar los valores.

- ***Experimento 3:***

  Para este experimento se ha utilizado un paso de aprendizaje $\alpha = 0.009$, se aumenta el número de datos del set de aprendizaje ligeramente.

  En la figura 5-13, se muestran los centroides a lo largo de la simulación de este segundo experimento.

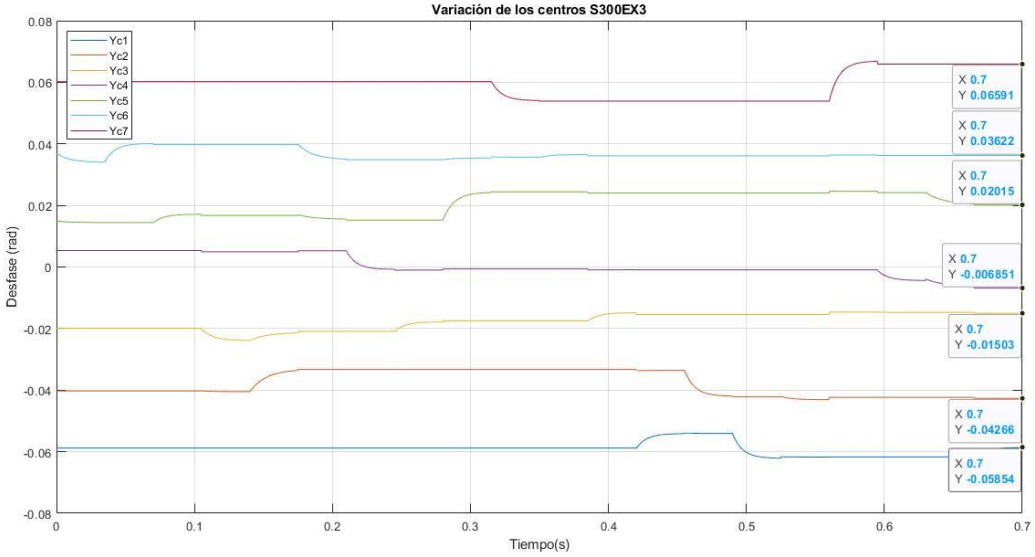


Figura 5-13. Evolución centroides Sigma=300-Exp 3.

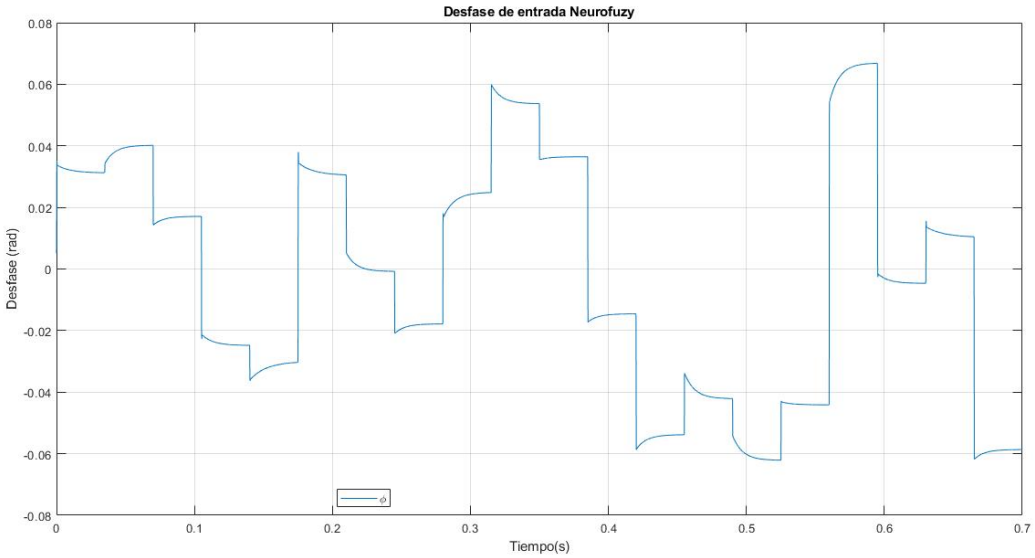


Figura 5-14. Evolución del desfase de diseño Sigma=300-Exp 3.

Esta vez el proceso de entrenamiento ha sido ligeramente distinto al de caso de Sigma=500, por un lado, como no se utiliza un número excesivo de datos de entrenamiento, y estos son aleatorios, puede ocurrir que no siempre haya datos suficientes para entrenar cada centro.

5.2.3. Estructura aprendizaje Neurofuzzy 2 entradas

El segundo esquema (Figura 5.15) utilizado en las simulaciones permite cuyo objetivo es identificar la ANN inversa, pero esta vez se pretende usar dos entradas, la potencia que devuelve el convertidor actual y con un ligero retraso para obtener la potencia anterior. El objetivo es poder captar la dinámica de la planta, que como explicamos al inicio del capítulo, existen ciertas no linealidades que en el caso de un controlador PI clásico, quedan anuladas.

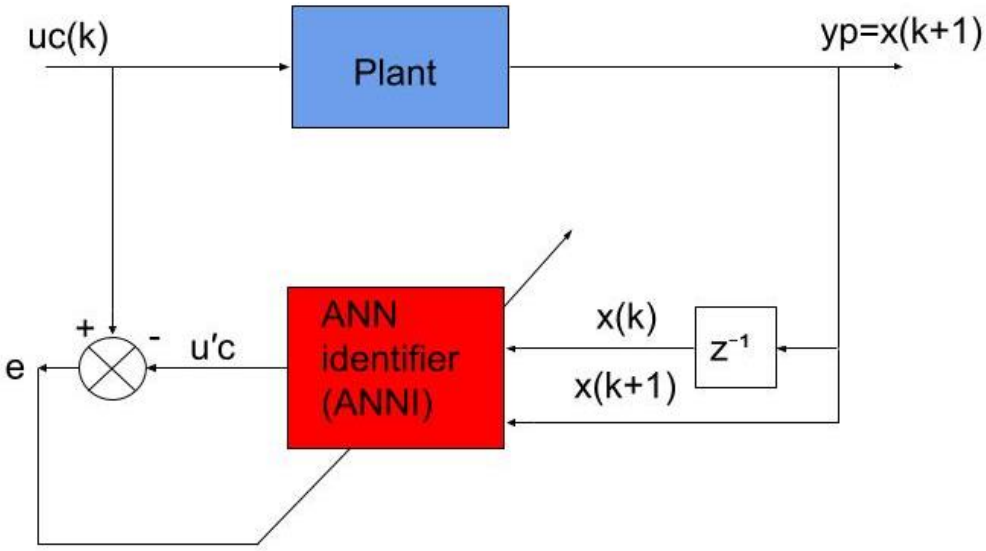


Figura 5-15. Esquema de identificación ANN 2 entradas.

Ahora debido a que tenemos 2 entradas, tenemos 7^2 =49 centroides ya que cada uno representa cada posible combinación de las entradas, teniendo en cuenta que su rango en potencia se divide al igual que antes en 7 tramos, por tanto, combinando dos a dos, tenemos definido lo que se ha denominado en el capítulo 3, universo de entrada y por tanto las reglas son de la forma descrita en el apartado 3.6.1.

La diferencia principal para los cálculos respecto a lo detallado en el neurofuzzy de 1 entrada es que cada función de activación ahora es producto de dos funciones de pertenencia, de forma, que $z(i)$ mide la pertenencia por cada par de valores de potencia recibidos a cada regla.

5.2.4 Simulaciones neurofuzzy 2 entradas

5.2.4.1. Aprendizaje de los centroides con μ fija y σ =500 W

Para este aprendizaje, debido a la cantidad de centros, se optado por tabular también la evolución de estos, pero como puede verse, no queda tan claro como en el neurofuzzy de 1 entrada, se empieza a tratar a los centroides de una ANN como meros pesos que definen la respuesta a una entrada que previamente se ha entrenado. Como antes se ha realizado para un valor del ancho de la campana de Gauss igual a 500. Y se han dejado fijos los centros de la Gaussiana. Tenemos que los valores de partida son.

$$\mu = [-2400, -1600, -800, 0, 800, 1600, 2400] \text{ (W)} \qquad (5.4)$$

$$\sigma = [500, 500, 500, 500, 500, 500, 500] \text{ (W)} \qquad (5.5)$$

Y a continuación los valores tabulados de los centroides a lo largo de los 3 experimentos realizados:

Tabla 5-3. Evolución centroides experimento 1.

Centroides (rad)	y_c^1	y_c^2	y_c^3	y_c^4	y_c^5	y_c^6	y_c^7
y_{1c}	-0.06195593	-0.01837267	-0.00041891	-1.44E-06	-9.60E-10	-3.80E-14	-3.14E-20
y_{2c}	-0.01229621	-0.04373007	-0.02590908	-0.00060742	-3.08E-06	-4.47E-10	-3.03E-14
y_{3c}	-5.72E-05	-0.01298226	-0.01909085	-0.01167587	-0.00025473	-2.29E-06	-1.15E-09
y_{4c}	2.97E-06	0.00053628	0.00261523	0.00482031	-0.00450611	-0.00053381	-3.30E-06
y_{5c}	1.13E-09	1.26E-06	0.00020875	0.00964888	0.0217072	-0.00196545	-0.00032967
y_{6c}	1.09E-14	1.65E-10	2.42E-06	0.00109405	0.01306483	0.05173737	0.0207101
y_{7c}	7.17E-21	3.92E-14	3.08E-09	2.98E-06	0.00050321	0.02989748	0.06558195

Tabla 5-4. Evolución centroides experimento 2.

Centroides (rad)	y_c^1	y_c^2	y_c^3	y_c^4	y_c^5	y_c^6	y_c^7
y_{1c}	-0.06119213	-0.04416412	-0.0009039	-4.12E-06	-3.76E-09	-1.22E-13	-1.06E-19
y_{2c}	-0.02382111	-0.05824613	-0.03813752	-0.00160266	-7.59E-06	-1.40E-09	-1.89E-13
y_{3c}	0.0004616	0.005428	-0.02176431	-0.02707314	-0.00103366	-9.46E-06	-5.19E-09
y_{4c}	7.47E-06	0.00120186	0.01971746	-0.00066436	-0.01913513	-0.00173647	-7.25E-06
y_{5c}	1.85E-09	5.08E-06	0.00105062	0.02439947	0.03619798	-0.00679198	-0.00093844
y_{6c}	3.74E-14	1.26E-09	5.39E-06	0.00203397	0.03428235	0.04141859	0.0234156
y_{7c}	6.21E-20	6.38E-14	4.21E-09	5.48E-06	0.0007078	0.045271	0.05634268

Tabla 5-5. Evolución centroides experimento 3.

Centroides (rad)	y_c^1	y_c^2	y_c^3	y_c^4	y_c^5	y_c^6	y_c^7
y_{1c}	-0.05905447	-0.05818981	-0.00133974	-7.41E-06	-5.46E-09	-2.28E-13	-2.60E-19
y_{2c}	-0.01577292	-0.03574881	-0.06685082	-2.78E-03	-1.65E-05	-3.76E-09	-2.43E-13
y_{3c}	0.00166109	0.0168742	-0.02369864	-0.06084177	-2.53E-03	-1.58E-05	-7.80E-09
y_{4c}	1.66E-05	0.00307954	0.04327114	0.00606169	-0.03849992	-3.52E-03	-1.45E-05
y_{5c}	6.66E-09	1.25E-05	0.00241197	0.05328847	0.02379496	-0.02066576	-0.00188614
y_{6c}	1.27E-13	3.37E-09	9.55E-06	0.00301922	0.06894958	0.04270739	0.01841638
y_{7c}	2.39E-19	8.86E-14	5.20E-09	9.47E-06	0.001255	0.06393214	0.05835874

Hemos decidido de las gráficas de evolución de centroides, mostrar 3 de cada 7 (y_{1c}, y_{4c}, y_{7c})por experimento para poder evaluar cómo cambian los valores.

- ***Experimento 1:***

  Para este experimento se ha utilizado un paso de aprendizaje $\alpha = 0.1$

  En las figuras 5-16, 17 y 18 se muestra la evolución de los centroides correspondientes a y_{1c}, y_{4c}, y_{7c} .

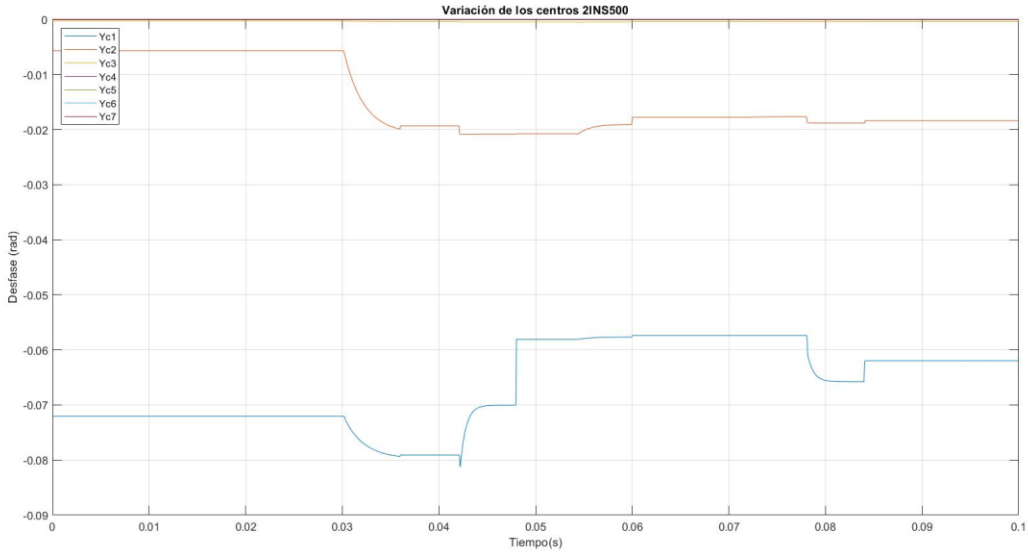


Figura 5-16 Evolución centroides y_{1c} Sigma=500. Exp 1

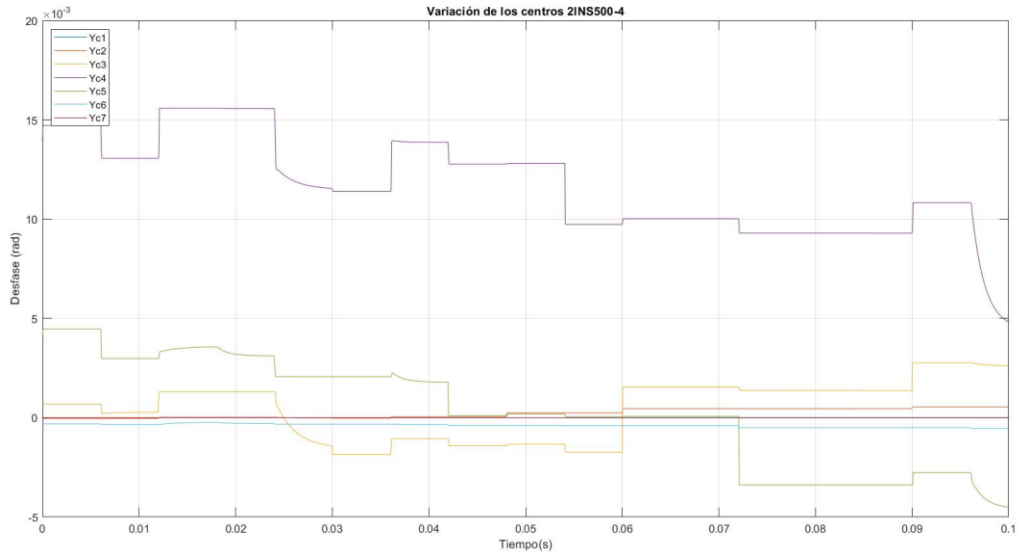


Figura 5-17. Evolución centroides y_{4c} Sigma=500-Exp 1.

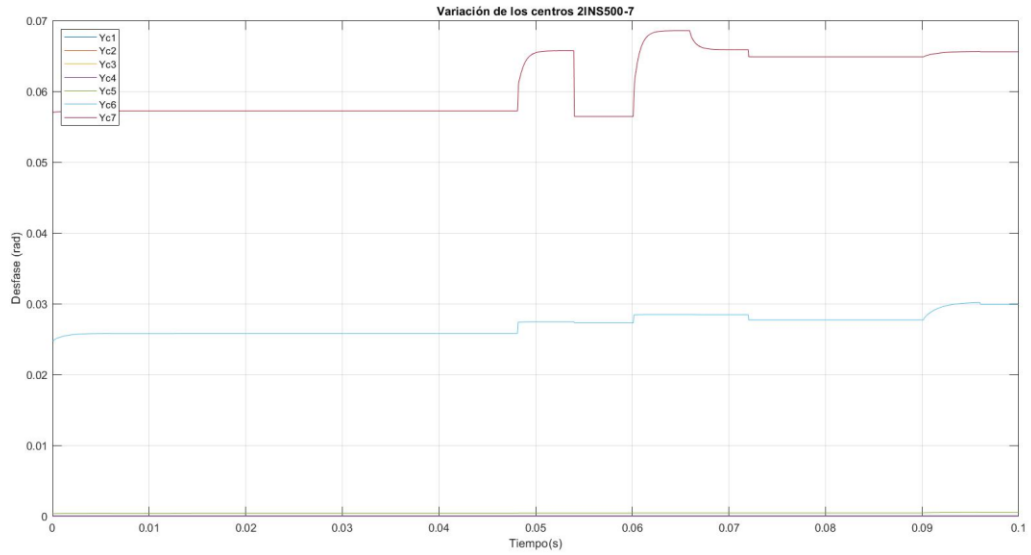


Figura 5-18. Evolución centroides y_{7c} Sigma=500-Exp 1.

Podemos ver que una parte importante de los centroides no han cambiado mucho su valor de 0, en la tabla se puede apreciar mejor, esto creemos que se debido a que como hemos explicado, hay distintas combinaciones entre ambas entradas, y es que algunas no llegan a darse, estamos suponiendo que los parámetros de las funciones de pertenencia son fijos y de valor equidistante entre ellos.

- ***Experimento 2:***

  Para este experimento se ha utilizado un paso de aprendizaje α = 0.09 y alrededor de 5 veces más de datos que el primer experimento.

  En las figuras 5-19,20 y 21 se muestra la evolución de los centroides correspondientes a y_{1c}, y_{4c}, y_{7c} .

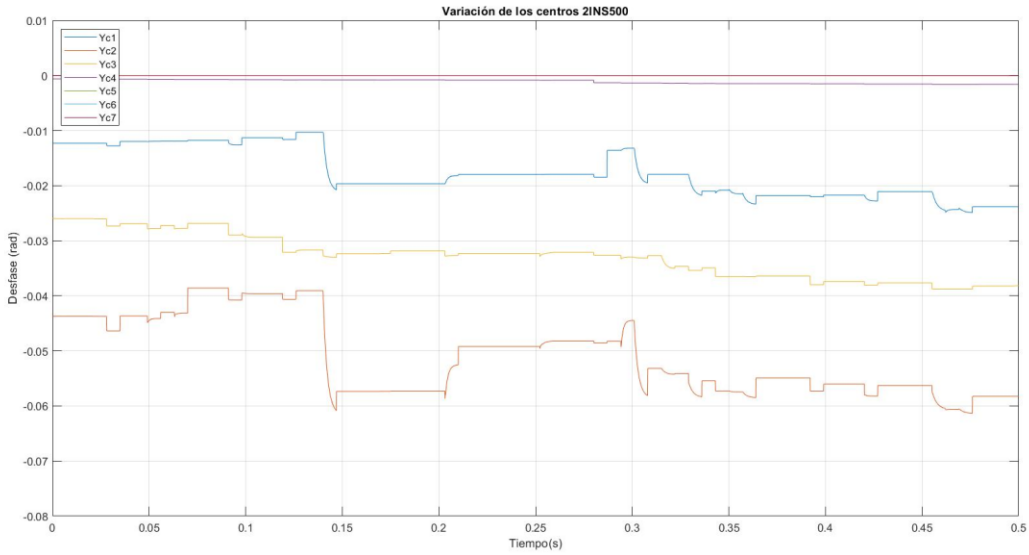


Figura 5-19. Evolución centroides y_{1c} Sigma=500-Exp 2.

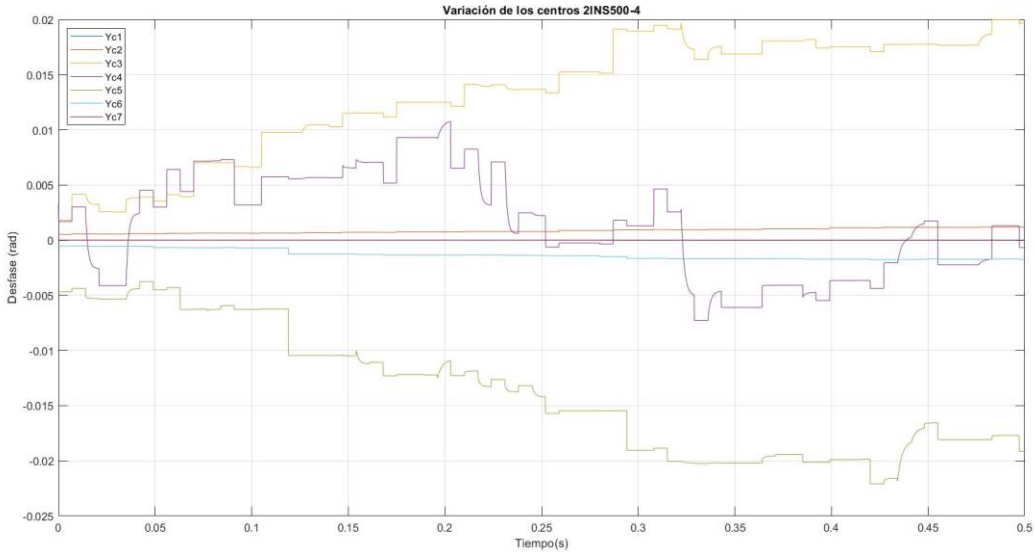


Figura 5-20. Evolución centroides y_{4c} Sigma=50-Exp 2.

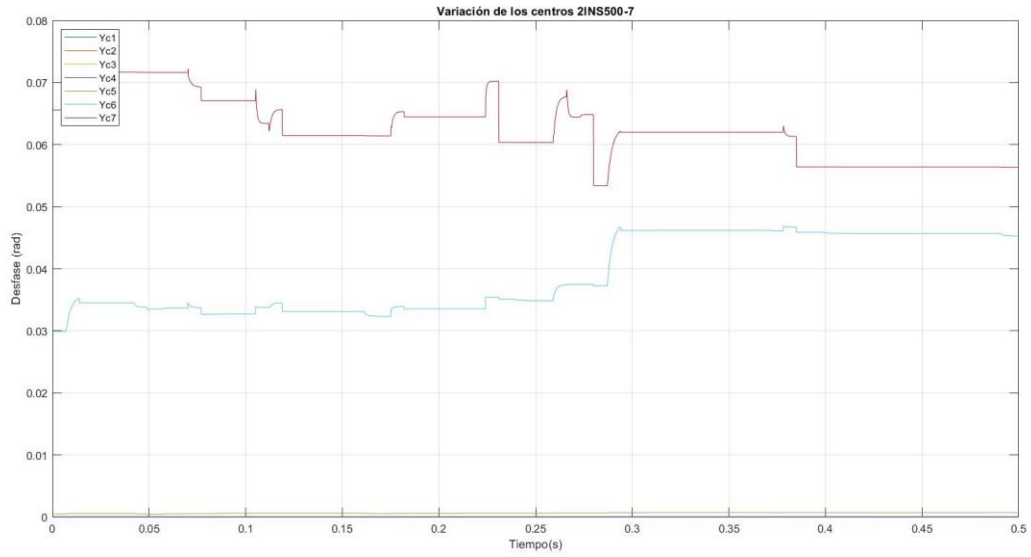


Figura 5-21. Evolución centroides y_{7c} Sigma=500-Exp 2.

El segundo experimento tiene como objetivo al usar muchos más datos, es una vez estimados valores para los centros entorno a un valor, luego dejar que puedan converger más precisamente a sus valores correctos, esta forma de proceder es útil desde el punto de vista, de que no es lo mismo para el algoritmo de aprendizaje partir de 0 que tener ya un valor de referencia dentro del mismo orden de magnitud e incluso cercano.

- ***Experimento 3:***

  Para este experimento se ha utilizado un paso de aprendizaje $\alpha = 0.09$ y se mantiene el mismo número de datos a usar.

  En las figuras 5-22, 23 y 24 se muestra la evolución de los centroides correspondientes a y_{1c}, y_{4c}, y_{7c}.

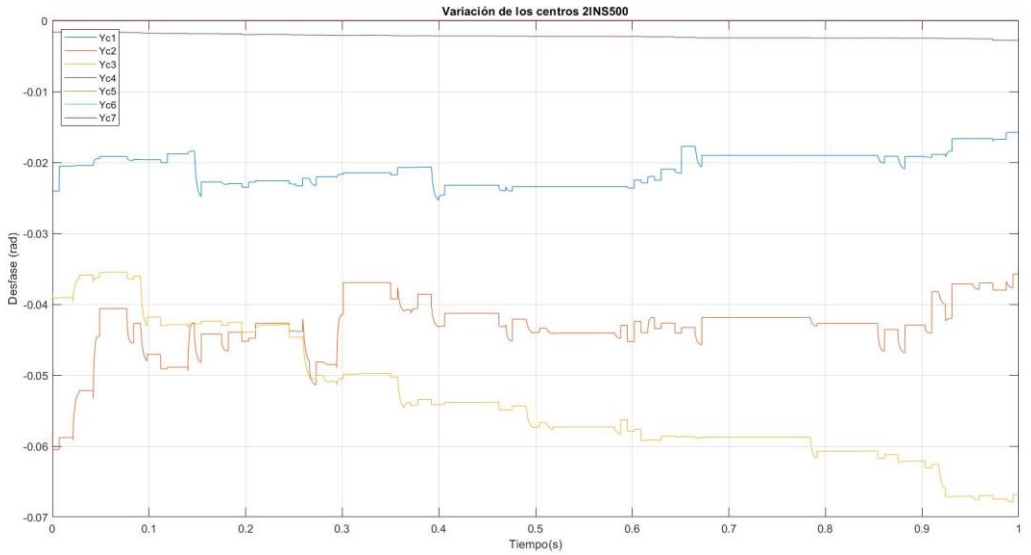


Figura 5-22. Evolución centroides y_{1c} Sigma=500-Exp 3.

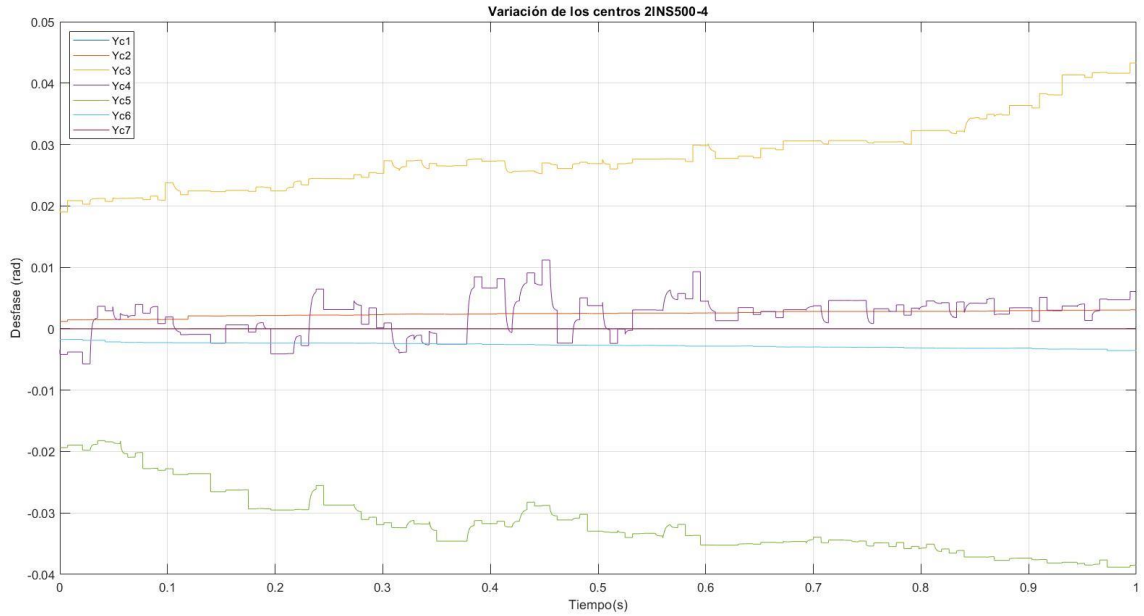


Figura 5-23. Evolución centroides y_{4c} Sigma=500-Exp 3.

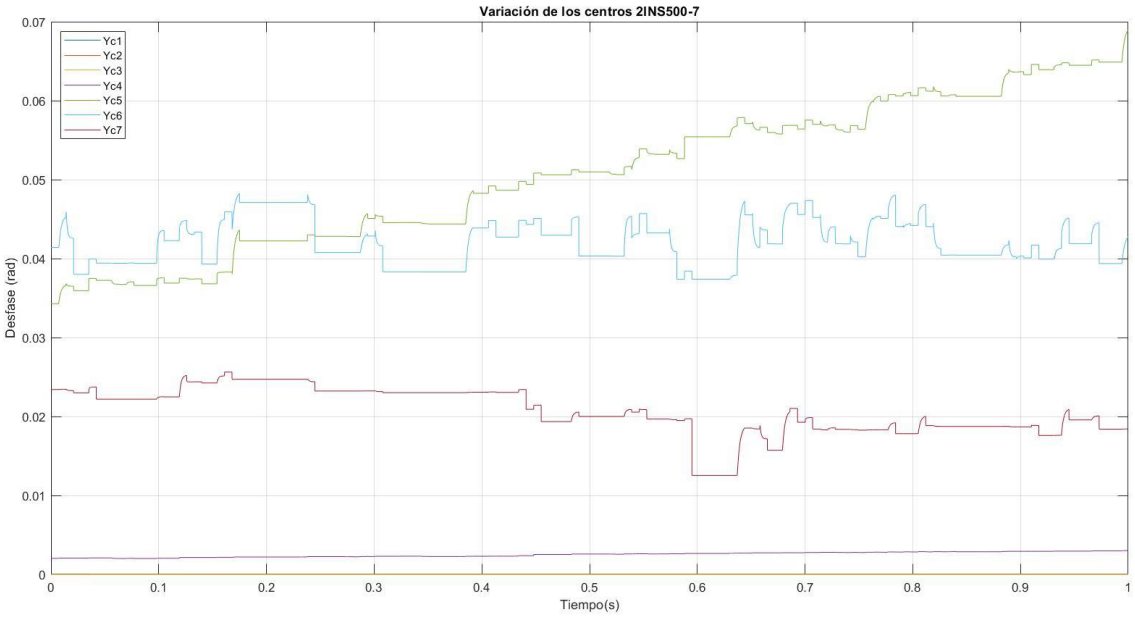


Figura 5-24. Evolución centroides y_{7c} Sigma=500-Exp3.

En el último experimento se vuelve a mantener el número de datos con el objetivo de consolidar el aprendizaje, aunque no se ha podido mejorar la situación de una parte de los centros, los cuales no han variado lo suficiente.

5.3. Neurofuzzy como control del DAB. Simulaciones

En este apartado haremos diversas propuestas de esquemas de control usando el Neurofuzzy entrenado. De la misma manera, podremos comprobar la bondad del aprendizaje con las simulaciones de control y cuantificar cual es la respuesta del Neurofuzzy a la hora de llevar el seguimiento de potencia.

5.3.1 Esquema control en bucle abierto Neurofuzzy 1 entrada

En este caso, se propone como primera validación del aprendizaje, evaluar la salida del Neurofuzzy ante distintos valores de entrada de la potencia, que en este caso actuará como referencia. Lo que se hace es directamente calcular el desfase de salida, no se realiza aprendizaje alguno.

5.3.1.1. Testeo Neurofuzzy σ=500

Por simplicidad, la referencia en potencia que mostramos al Neurofuzzy es una escalera que recorre todo el rango de diseño. Dicha escalera de valores es:

$$P = [-2400, -1600, -800, 0, 800, 1600, 2400]$$

Los valores de los centros utilizados son:

$$y_c = [-0.06129177, -0.03596988, -0.01805676, 0.0017173, 0.0211484, 0.03529046, 0.05937131]$$

En la figura mostrada a continuación podemos ver el seguimiento de la potencia de referencia por parte del neurofuzzy:

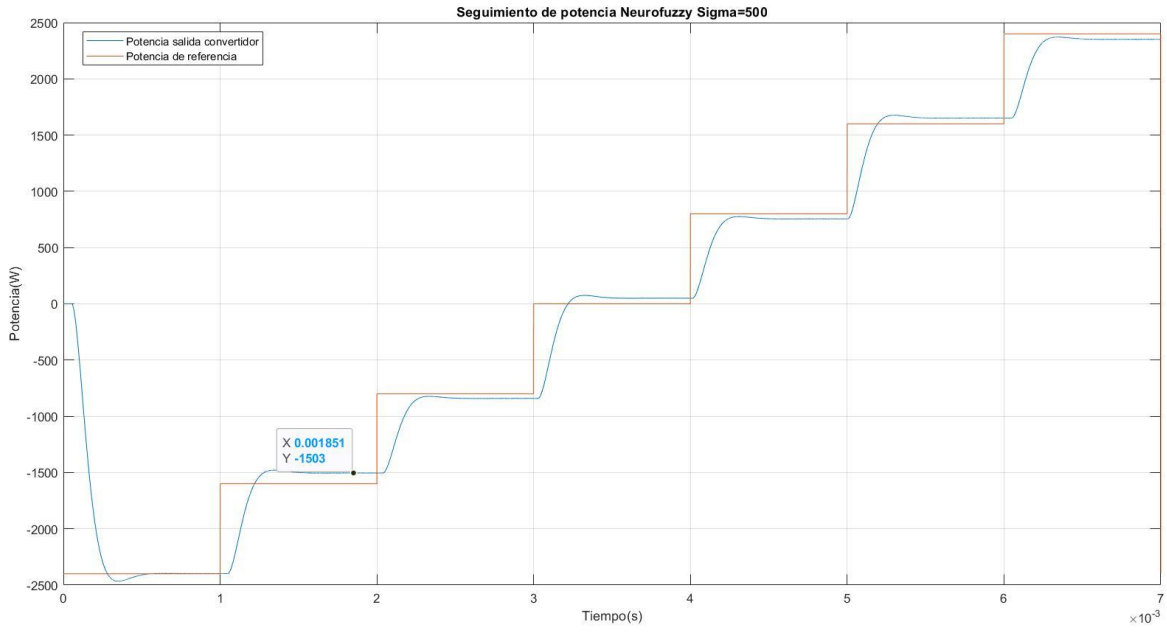


Figura 5-25. Seguimiento de la potencia Sigma=500.

5.3.1.2. Testeo Neurofuzzy σ =300

Los valores de los centros utilizados son:

$$y_c = [-0.05854490, -0.04266053, -0.01502599, -0.0068512, 0.0201516, 0.036218, 0.065907]$$

En la figura mostrada a continuación podemos ver el seguimiento de la potencia de referencia por parte del neurofuzzy:

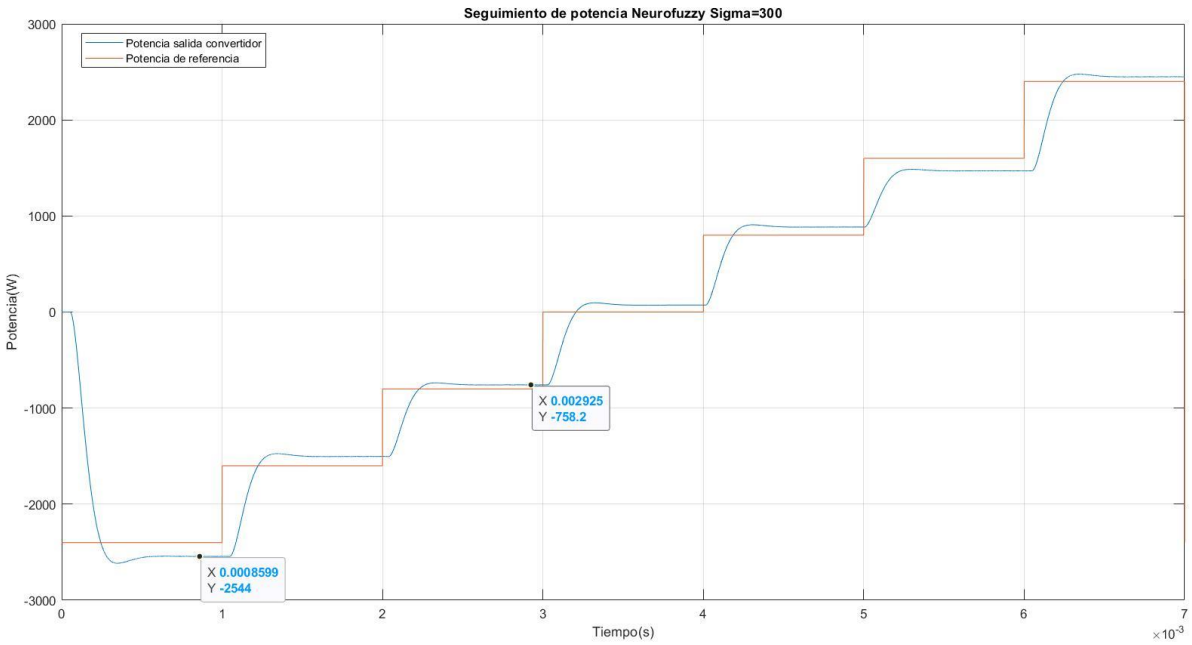


Figura 5-26. Seguimiento de la potencia Sigma=300.

5.3.2 Control en bucle cerrado junto PI

En este caso, una vez asumido el resultado del aprendizaje, ha resultado ser un esbozo de la función inversa del DAB, como solución, se propone ahora si un esquema de control en bucle cerrado que permite el aprendizaje online. Es decir, se hace uso de la idea de modelo de referencia, el cual será un PI, que recibe el error en potencia (igual que el diseñado en el apartado 4.1) y como salida proporciona un desfase que debe ser el mismo que tiene que dar el neurofuzzy al DAB. El neurofuzzy recibe como error la diferencia entre su propia salida y el desfase que el PI proporciona, Por tanto, el Neurofuzzy tiene como objetivo constante ajustar sus centros de manera que arroje como salida el mismo desfase que el PI, que en resumen es el que necita el DAB para alcanzar la referencia de potencia.

Hemos simulado dicha mejora, proporcionando un escalón negativo al neurofuzzy como referencia, y por ende al PI, la figura muestra el seguimiento de esta referencia.

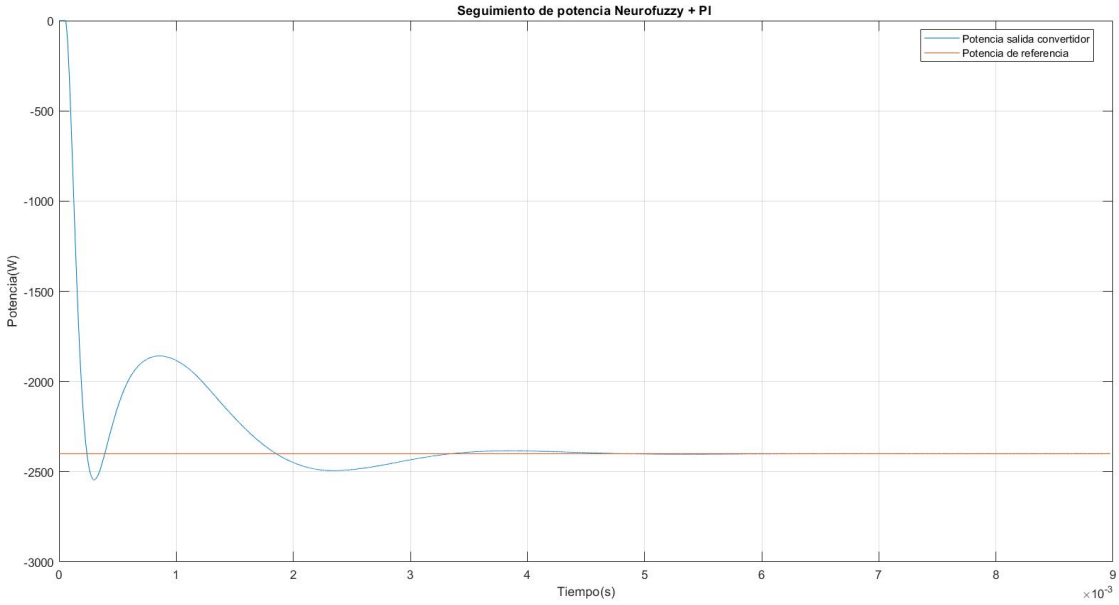


Figura 5-27. Evolución del seguimiento de potencia PI.

A continuación, la figura 5-28 muestra la variación que han sufrido los centros:

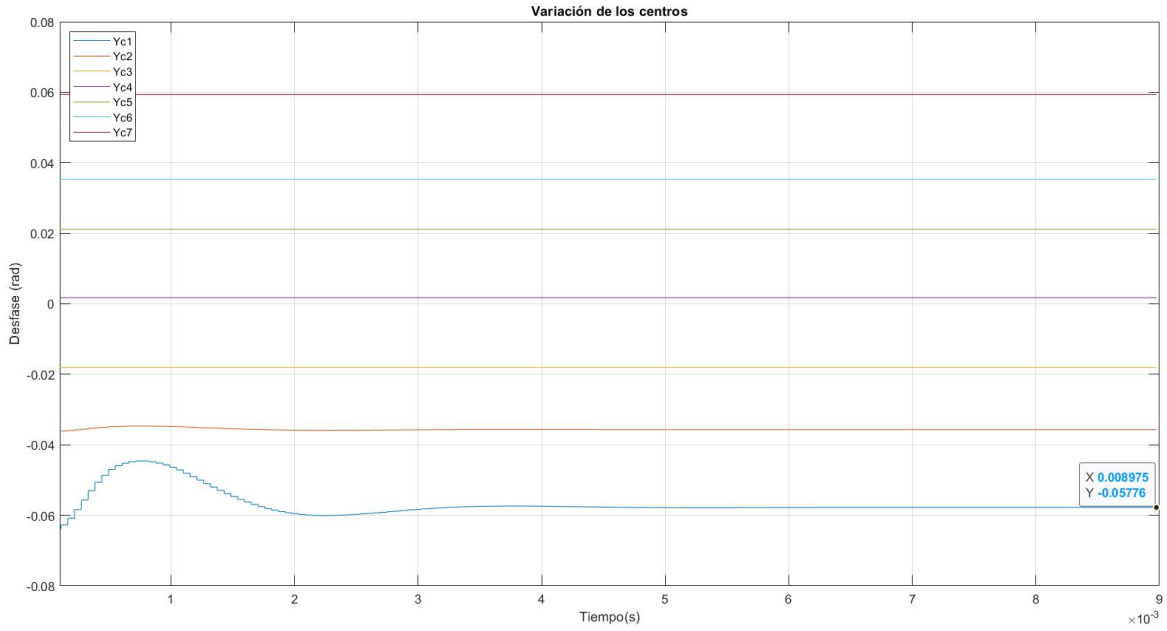


Figura 5-28. Seguimiento de la potencia Sigma=300.

Comprobamos que el control usando referencia, concretamente el PI que utilizamos para llevar a cabo el control del convertidor. Obtenemos dos ventajas con este montaje, por un lado, permite salvar el control si el aprendizaje realizado no ha arrojado valores de los centros precisos, y, por otro lado, simultáneamente al control, modifica los centros afectados por la referencia, haciéndolos converger al verdadero valor que debería tener.

5.4. Control usando Neurofuzzy de 2 entradas

5.4.1 Control "Direct neuro-control"

En el caso del neurofuzzy de 2 entradas para el primer control se propone el siguiente esquema:

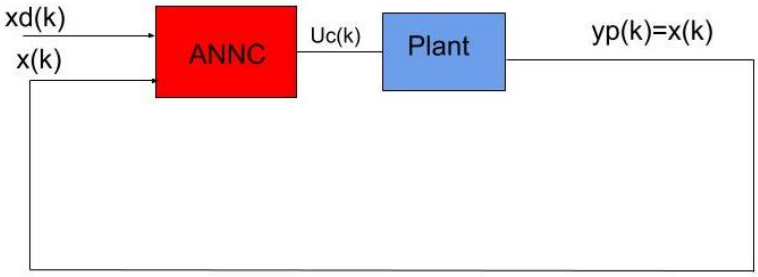


Figura 5-29. Direct neuro-control Neurofuzzy.

En resumen, si antes teníamos como entradas a nuestro neurofuzzy la potencia anterior y la actual, ahora tendremos la potencia de referencia y la actual. Entonces esos valores se pasan al neurofuzzy, para obtener la salida y comprobar los valores aprendidos.

Usando como referencia: $P = [-1500, -700, 0.0, 300, 1200, 2000]$

La figura que muestra el seguimiento a dicha referencia:

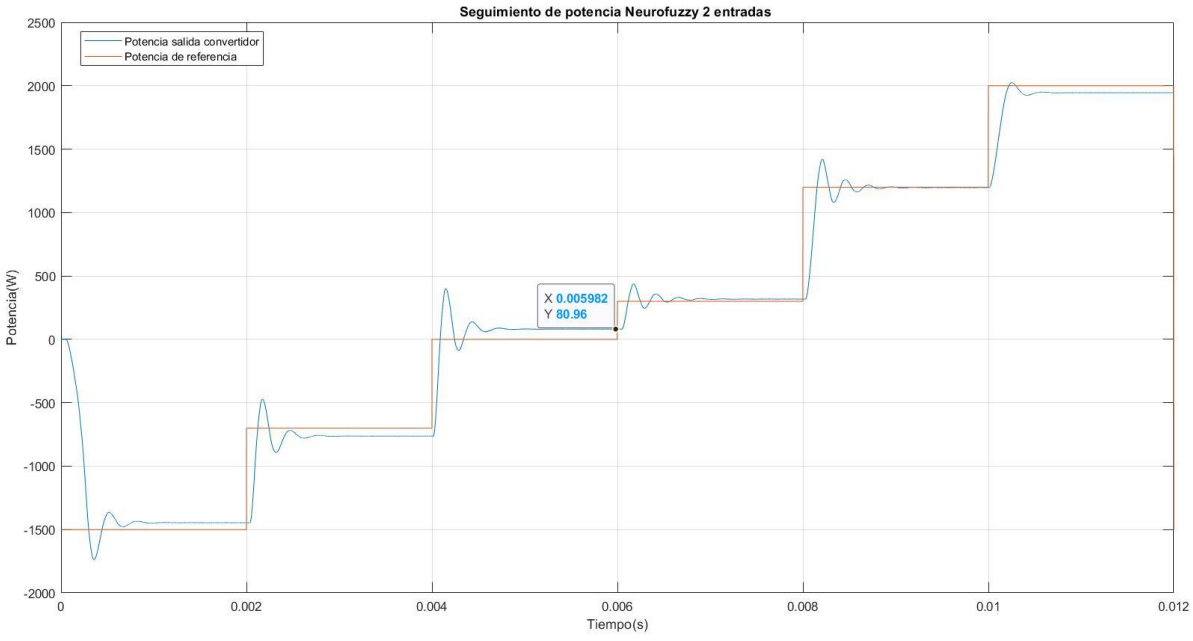


Figura 5-30. Direct neuro-control.

5.4.2 Control "Direct-Inverse neuro-control"

Como último esquema para el control, se propone el que se muestra en la figura, conocido con el nombre "Direct-Inverse neuro-control "en [2]. Dicho esquema, realiza un control con aprendizaje online de tal forma que el bloque "ANNC" representa el neurofuzzy que hemos entrenado previamente y que se encarga de dar la señal de control, el desfase que necesita el DAB para transferir la potencia de referencia xd(k). Mientras tanto el bloque ANNI, es otro neurofuzzy, pero que, a diferencia del primero, se encarga de minimizar el error que hemos definido como la diferencia de desfases, entre el que nos devuelve ANNC y el de ANNI, de tal forma que ANNI está entrenando los parámetros de ANNC, modificándolos de forma que se corrijan los posibles defectos que hay en el control del DAB.

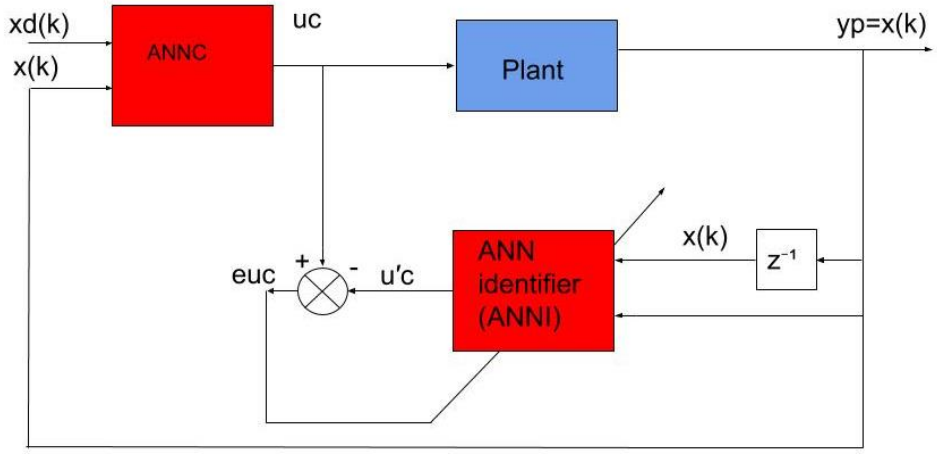


Figura 5-31 Direct-Inverse neuro-control

Los resultados obtenidos, no han sido satisfactorios del todo, pues tras varios ciclos, no se observa una corrección significativa, ni a largo ni a corto plazo de la actuación del neurofuzzy sobre el DAB.

La figura 5-31 muestra un seguimiento de referencia, donde tras repetir la escalera en varias ocasiones, se detectan ciertos cambios de la red para adaptarse, pero no es suficiente.

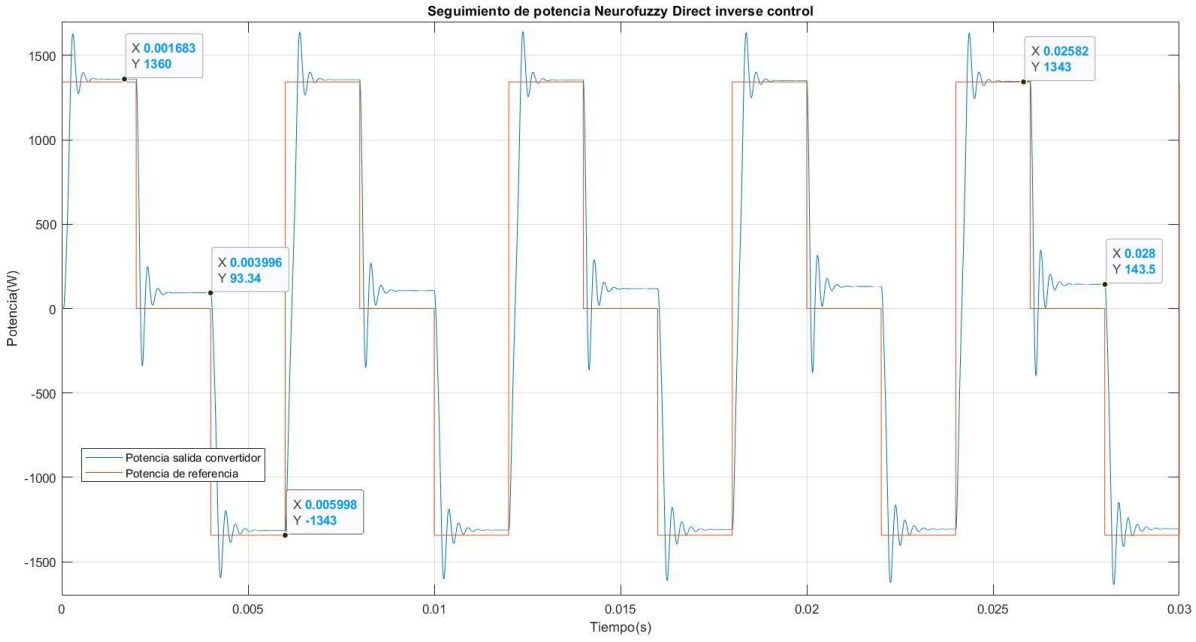


Figura 5-32. Direct-Inverse neuro control.

5.4.3 Control en bucle cerrado junto PI

A parte de las estrategias de control propuestas, se probado el funcionamiento de la idea el apartado 5.3.2 con el neurofuzzy de 2 entradas, arrojando un resultado positivo ante distintas referencias, pues como hemos visto con el direct-neuro control, no existe un mecanismo para reducir el error sobre la referencia.

De la misma manera que con el neurofuzzy de 1 entrada, se ha colocado el PI usado en el capítulo 4, como referencia la red neuronal, de forma que se utiliza el error entre desfase deseado (dado por el PI) y la propia salida del neurofuzzy.

En este caso ha sido necesario modificar el tiempo integral del PI, pues si no, el control era lento y oscilatorio, esto es debido funciona de forma discreta tanto el PI como el neurofuzzy, por tanto, tardaba mucho en coger al PI.

El seguimiento de referencia ha quedado de la siguiente manra:

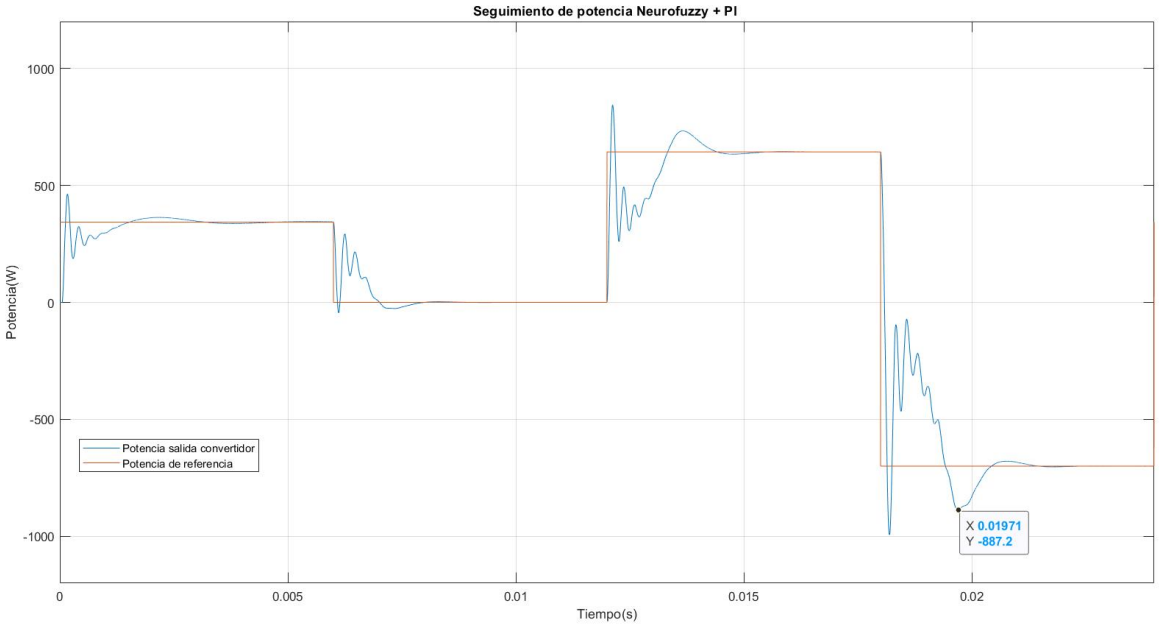


Figura 5-33. Seguimiento de referencia 2-in PI.

6. PROGRAMACIÓN DEL C2000™ F28035 PICCOLO MICROCONTROLLER

El mejor modo de resolver una dificultad es no tratar de soslayarla.

- Noel Clarasó, Escritor español -

EN el presente capitulo se tratará de explicar de forma clara y resumida una breve noción de la programación del microcontrolador F28035, con especial énfasis en los módulos ePWM y HRPWM.

6.1. Introducción

Como se mencionó en el apartado 1.3, uno de las posibles ampliaciones del Proyecto, es el de poder probar en laboratorio los esquemas de control simulados en el convertidor original. Debido a que exisiti dicha posibilidad, se llegó estudiar el funcionamiento de un Microcontrolador concreto. El F28035 de Texas Instruments, perteneciente a la serie "Piccolo" de Microcontroladores. La idea era usar el dispositivo principalmente para generar las señales de modulación de los MOSFETs del convertidor, proporcionando el desfase necesario para la transferencia de potencia.

6.1.1. Características del microcontrolador

El dispositivo en cuestión se trata del microcontrolador de la serie "Piccolo" TMS320F28035 de Texas Instruments, orientado especialmente al control en tiempo real con convertidores de potencia, motores etc.

Dicho MCU, cuenta con una placa de expansión (con aislamiento) que implementa las conexiones que facilitan su uso (Figura 6-1).

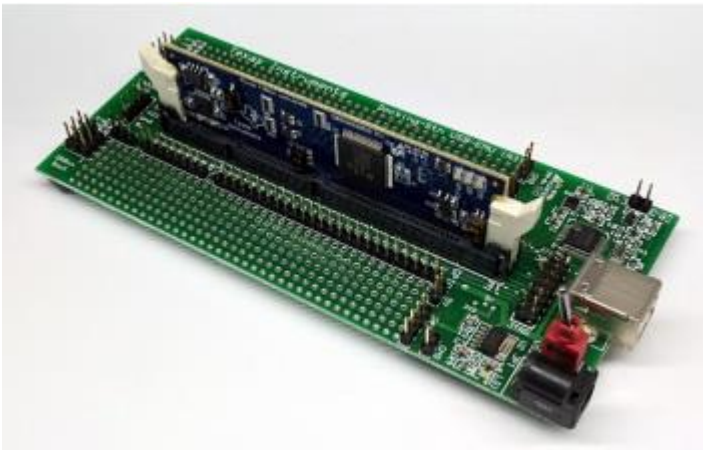


Figura 6-1. Kit de experimentación F28035 [16].

Las cacracteristicas a destacar que posee dicho MCU son:

- CPU de 32-bit (serie C28x)
    - Frecuencia de reloj maxima de 60MHz
    - Arquitectura de BUS Harvard
- CLA (Control Law Accelerator)
    - Acelerador matematico de punto flotante 32-bit
- Bajo consume
- 45 GPIO
- Timers de 16-bit independientes para cada ePWM
- Modulos SCI, I2C, SPI, eCAN
- Modulos de alta resolución (HR) que complementan a los clasicos

6.2. ePWM

En este apartado se explicarán las caracteristicas del modulo PWM del que dispone el Piccolo, el funcionamiento del mismo y se concretará para el Código de generación de disparos para el DAB que se incluirá en un "Anexo" al final de esta memoria.

Los dispositivos de la seria 2803x, cuentan todos con un módulo ePWM de tipo 1 (Fija las características avanzadas que posee o no posee el módulo) y dependiendo del encapsulado puede tener 14,12 u 8 canales. En nuestro caso disponemos de 8 (en la jerga de registros, del "EMPWM1A al EPWM7B"), de los cuales 4 son de tipo HR (High Resolution).

Como principales características el módulo puede generar 16 PWM independientes o 8 PWM (dual-edge y asimétricos), posibilidad de desfase entre módulos, trip zone programable, y generación de zona muerta. se compone de otros submódulos que a continuación numeramos y que serán explicados:

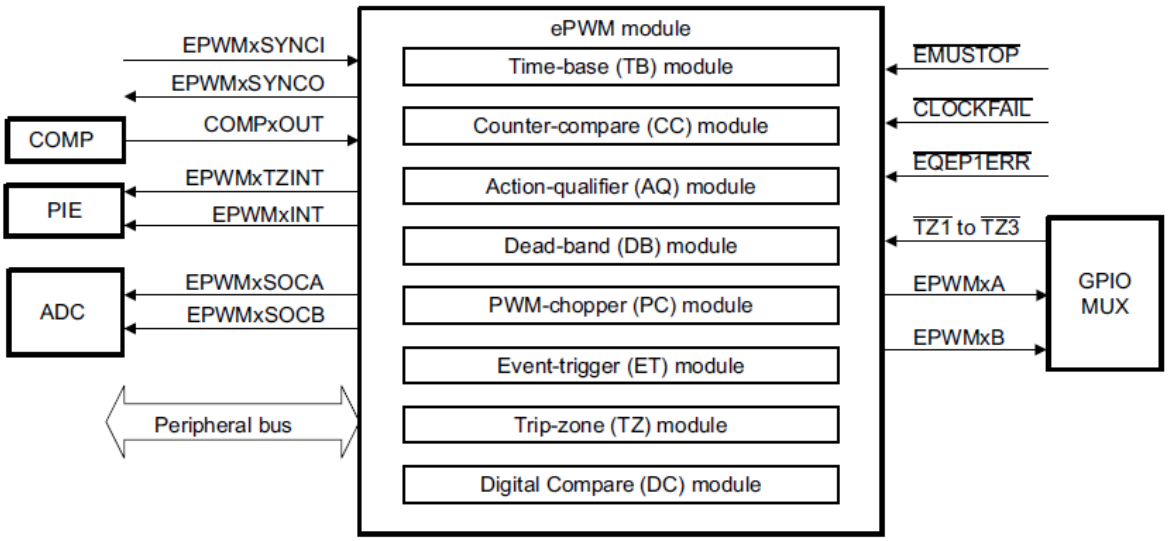


Figura 6-2. Esquemático del módulo ePWM y sus submódulos [17].

Las diferencias entre un ePWM de tipo 1 y 0 son, la posibilidad de alta resolución para variar el periodo del PWM, añade un submódulo de comparación digital, incrementa la resolución de la "zona muerta" y generación SOC.

6.2.1.1 Time base submodule

Es el más importante de los submódulos que componen el "enhanced PWM", pues para generar señales, lo primero es poder gestionar la temporización que proporciona el reloj de cualquier sistema microprocesador.

El módulo TB, permite estas opciones:

- Escalado del reloj del sistema.
- Configurar frecuencia/periodo.
- 3 modos de cuenta (Ascendente, descendente y ambas).
- Relación master-Slave para desfasar la señales de unos módulos con las de otros.

6.2.1.2 Counter compare submodule

El módulo CC, permite estas opciones:

- Especifica el duty cicle de cada PWM independiente (A y B).
- Especifica el tiempo en que se producen los eventos de conmutación para cada PWM (A y B).

6.2.1.3 Action-qualifier submodule

Este submódulo se encarga de todas las tareas asociadas a la forma de onda y flancos de esta.

El módulo AQ, permite estas opciones:

- Especifica la acción a realizar cuando se produce un evento asociado al módulo CC o a él modulo TB.
- Permite configurar el estado de la salida del PWM o configurar la Dead Band mediante software.

6.2.1.4 Dead band submodule

El submódulo facilita la generación de disparo para los convertidores, implementado zonas muertas que son muy necesarias para evitar conmutaciones en un instante erróneo respecto a otra onda complementaria.

El módulo DB, permite estas opciones:

- Control de la zona muerta.
- Especifica los valores de delay en cada flanco de la onda.

6.2.1.5 PWM-chopper submodule

Orientado a los interruptores tipo chopper y control de motores.

El módulo PC, permite estas opciones:

- Crear una frecuencia chopper.
- Controlar el duty cicle de los trenes de pulsos.
- Especifica los valores de delay en cada flanco de la onda.

6.2.1.6 Trip-zone submodule

El módulo TZ, permite estas opciones:

- Configura el módulo de ePWM para actuar ante señales trip-zone y eventos de comparación digitales.
- Especifica la acción a realizar cuando ocurre un fallo.

6.2.1.7 Event-trigger submodule

El módulo ET, permite estas opciones:

- Habilita los eventos que pueden activar una interrupción.
- Permite especificar los comportamientos a otros tipos de interrupciones.

6.2.1.8 Digital-compare submodule

El módulo DC, permite estas opciones:

- Permite que las salidas del módulo comparador y las señales de la trip zone creen eventos y se filtren.

6.2.2. Programación del ePWM. Generación de ondas PWM

Para que el MCU genere los PWM necesarios, debemos programar el módulo ePWM de forma que genere 4 PWM de frecuencia 100KHz y Duty cicle del 50% que siguen el siguiente esquema, la figura 6-3 que muestra las formas de onda de los PWM que son necesarios para hacer funcionar el DAB:

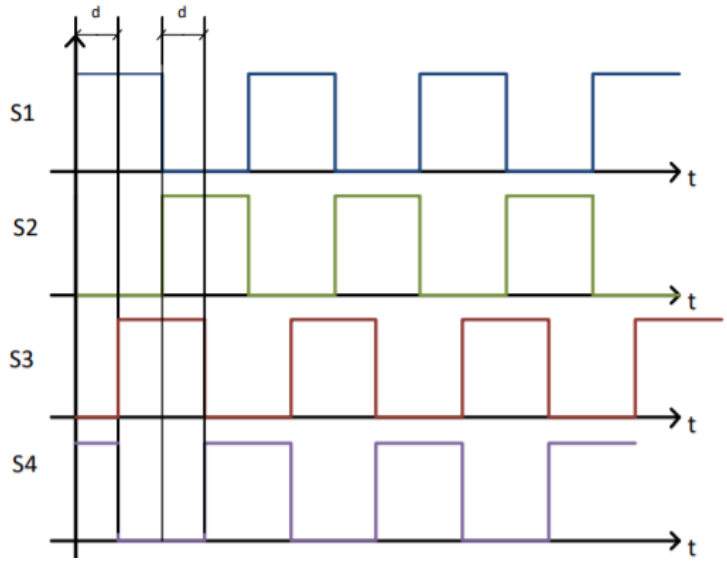


Figura 6-3. Forma de onda de las señales de disparo del convertidor [18].

Tenemos 4 PWM independientes para los 8 MOSFET, ya que el MOSFET izquierdo superior y derecho inferior (Q9 y Q7) del puente de entrada usan S1 como señal de disparo y S3 para los mismos MOSFETs (Q4 y Q2) del puente de salida, siendo S3 una versión de S1 desfasada φ rad. Ocurre lo mismo para los MOSFETs izquierdo inferior y superior derecho de los puentes (Q6 y Q8) y (Q1 y Q3), pero utilizan S2 y S4 que son las formas de onda complementarias de S1 y S3.

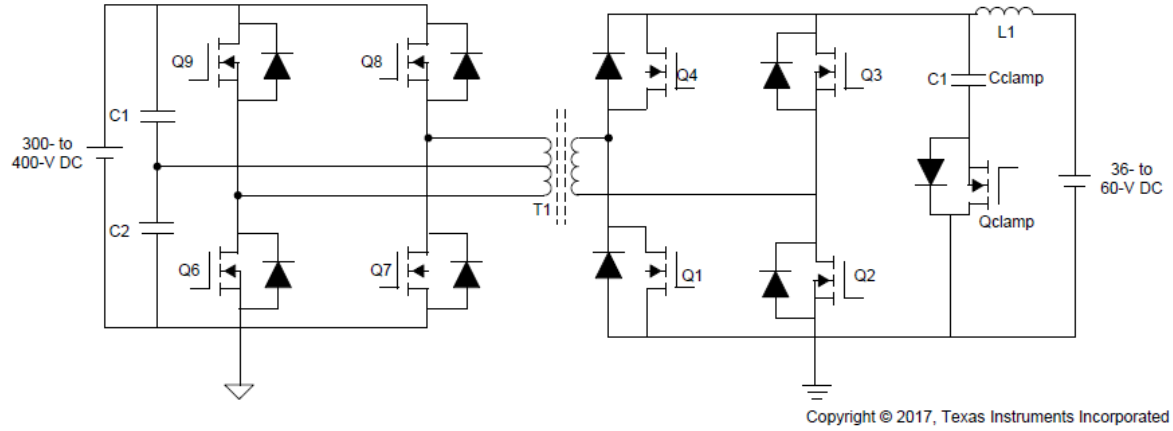


Figura 6-4. Esquema eléctrico DAB (Explicado apartado 4.1) [14].

Ahora pasaremos a explicar los registros que es necesario modificar y en que afectan a la consecución de la onda PWM necesaria.

De los módulos antes mencionados, nos afectan TB, CC, AQ y DB.

Por tanto, los registros que se deben configurar se muestran a continuación.

- ***Registro TBPRD***

  Controla la frecuencia del PWM, almacena el valor hasta el que tiene que llegar la cuenta del contador del TB. Se calcula como:

$$TBPRD = \frac{f_{clk}}{2\,fPWM} \qquad \text{(6.1) PWM ASIMETRICO}$$

Siendo f_{clk} la frecuencia del reloj y f_{PWM} la frecuencia del PWM que quiere generarse.

En nuestro caso, $TBPRD = \frac{60MHz}{2*100KHz} = 300$ es el valor que tenemos que usar.

- ***Registro CMPA/B***

  Controla el duty cicle del PWM, se calcula con (6.22)

$$CMPA/B = (1-D)*TBPRD \qquad (6.2)$$

Siendo Del duty cicle, en nuestro caso es fijo e igual al 50%, por tanto, CMPA/B=TBPRD/2=150

Por tanto, los registros que se deben configurar se muestran a continuación.

- ***Registro TBPHS***

  Controla el desfase del PWM del que se configura respecto al master (referencia que suele ser el PWM previo), se calcula mediante la expresión (6.3)

$$TBPHS = \frac{\varphi(rad)}{2\pi} 2TBPRD \qquad (6.3)$$

Siendo φ el desfase que se quiere aplicar, en nuestro caso será variable e igual a la salida del control.

- ***Registro CMPCTL***

  Se utiliza para habilitar los dobles registros y la carga desde los mismos de los valores de CMPA/B, en nuestro caso vale 0.

  Los registros TBCTL y AQCTLA/B al tener varios campos y opciones, se detallan a continuación en las tabla 6-1 y 6-2.

Tabla 6-1. Configuración Registro TBCTL.

Bit/grupo de bits	Valores posibles	Usado EPW1/2	Descripción campo
15:14 (FREE, SOFT)	00 01…1X	11/11	Configura comportamiento en eventos de emulación. No nos afecta.
13 (PHSDIR)	0 Down 1 Up	0/0	Especifica dirección de la cuenta tras evento de sincronización.
12:10 (CLKDIV)	000 (Sin escalado) 001…111 (1/128)	000/000	Prescalador del reloj (TB) No hace falta
9:7 (HSPCLKDIV)	000 (Sin escalado) 001…111 (1/14)	000/000	Prescalador del reloj (TB) de alta velocidad
6 (SWFSYNC)	0/1	0/0	Sincronización por software.
5:4 (SYNCOSEL)	00 01...11 (Desactivar)	01 (Zero)/ 00 (Usar salida de precedente)	Configura la fuente de sincronización para el módulo PWM
3 (PRDLD)	0/1	0 (Carga valor desde el shadow reg) /0	Activa la carga desde los “registros sombra”
2 (PHSEN)	0/1	0/1	Activa carga del desfase desde registro TBPHS
1:0 (CTRMODE)	00 01…11	10/10 (UP-DOWN)	Modo de cuenta (Up, Down, Up-Down, stop)

Tabla 6-2 . Configuración Registro AQCTLA/B.

Bit/grupo de bits	Valores posibles	Utilizado EPW1A /1B/2A/2B	Descripción campo
15:12 Reservado	----------	----------	----------
11:10 (CBD)	00 (No hacer nada) 01…11	00/00/00/00	Acción cunado contador llega al valor de CMPB decrementándose
9:8 (CBU)	00 01…11	00/00/00/00	Acción cunado contador llega al valor de CMPB incrementándose
7:6 (CAD)	00 01…11	01/01/01/01	Acción cunado contador llega al valor de CMPA decrementándose
5:4 (CAU)	00 01…11	10/10/10/10	Acción cunado contador llega al valor de CMPA incrementándose
3:2 (PRD)	00…11	00/00/00/00	Acción cuando contador llega al valor del periodo
1:0 (ZRO)	0/1	00/00/00/00	Acción cuando contador llega a 0

7. CONCLUSIONES Y ANÁLISIS DE LOS RESULTADOS OBTENIDOS

Cada solución da pie a una nueva pregunta...

- David Hume, filosofo del siglo XVIII -

FInalmente, tras 5 capitulos donde el objetivo era documentar de la mejor manera y dar conocer en detalles lo que ha supuesto este TFG, vamos a hacer algunas reflexiones sobre el trabajo y resumir lo expuesto.

7.1. Análisis final

Para acabar, de forma breve pero clara, queremos analizar el resultado del trabajo y la conclusión a la que se llega.

7.1.1. Neurofuzzy como controlador

El principal objetivo del proyecto era que partiendo de una topología tipo DAB, pudiéramos simularla (reflejada en un montaje existente en laboratorio), para posteriormente implementar una estructura de inteligencia artificial, un neurofuzzy concretamente, y llegar a controlar el convertidor.

En el capítulo 5 hemos documentado el trabajo realizado con dicho neurofuzzy, tanto el entrenamiento como el control. Por el lado del aprendizaje, hemos podido comprobar que, con muy pocos datos de entrada, se puede modelar la función inversa del convertidor fácilmente, y si queremos precisión solo hay que implementar un buen proceso de aprendizaje, por tanto, en este sentido, las ventajas de las que hablamos en el capítulo 3 respecto a los controladores clásicos, han podido comprobarse. Es más, en el apartado 5.3.2, donde hemos usado para el control un PI como referencia, vemos que no solo este tipo de estructuras pueden entrenarse usando un controlador clásico para luego poder adaptarse mejor que los primeros a las mismas situaciones, sino que, además, pueden ser combinados, tomando lo mejor de cada uno para realizar algo así como un post-entrenamiento.

Respecto a la parte de control, por el número de estrategias que pueden llevarse a cabo, que son muchas, y por los experimentos realizados, podemos afirmar que el potencial de un neurofuzzy como controlador adaptativo es bastante amplio, y que lo aquí expuesto en este trabajo puede ampliarse sacando más partido a ese potencial.

7.1.2. Ampliaciones

Como se propuso en el primer capítulo, el último objetivo de este proyecto sería llegar a implementar en el DAB existente en laboratorio este neurofuzzy de forma que se puede extender las ideas que de este proyecto se extraen. Y es que sin realmente conocer un modelo del convertidor para desarrollar un controlador clásico o poder proporcionar los datos de entrenamiento, se puede igualmente emular al DAB primero con una simulación y luego con un "Neurofuzzy" que se apoya en esa simulación para conocer mejor que nadie el comportamiento del convertidor. Por tanto, se podría continuar este TFG para acabar desarrollando en el laboratorio un controlador fiable y basado en distintas rutinas que permitan arrancar el DAB desde el F28035.

Además, una vez conseguido dicho objetivo, sería posible implementar el DAB en un sistema de potencia más grande, y dejar que y tal como mostramos en el apartado 4.1, el convertidor pueda llevar la gestión de energía entre las baterías y la instalación, lo que supone tanto carga un banco de baterías, como mantener la tensión de la que se alimentarían otros convertidores y el resto de la instalación eléctrica.

7.1.3. Conclusión

Como acabo de decir, tras todo un proyecto como este, no cabe duda de que las estructuras basadas en inteligencia artificial como las ANNs, los neurofuzzy y los FLS. Cada día tienen más adeptos, y es obvio, ya que permiten tal flexibilidad en muchas tareas y aplicaciones, que siempre es atractiva la idea de utilizar dichas estructuras.

La conclusión es clara, el proyecto ha servido para desarrollar una gran cantidad de experiencia y comprobaciones que han resultado ser muy positivas, tanto por los conocimientos adquiridos como por los resultados. Por tanto, como se comenta en el apartado 7.1.2, el proyecto puede ser mucho más amplio y continuarse, habría faltado exprimir aún más las formas de aprendizaje y el perfeccionamiento de las estructuras de control, aun así, queda clara la utilidad de lo estudiado y sus futuras posibilidades.

REFERENCIAS

[1] R. W. De Doncker, D. M. Divan y M. H. Kheraluwala, «A Three-Phase Soft-Switched High-Power-Density dc/dc Converter for High-Power Applications,» *IEEE TRANSACTIONS ON INDUSTRY APPLICATIONS,* vol. 27, nº 1, JANUARY/FEBRUARY 1991.

[2] P. Vas, Artificial-Intelligence-Based Electrical Machines and Drives: Application of Fuzzy, Neural, Fuzzy-neural, and Genetic-Algorithm-based Techniques, Oxford University Press, April 1999.

[3] C. Gonzalez Morcillo, «Escuela Superior Informatica, Universidad de Castilla La Mancha,» 2011. [En línea]. Available: http://www.esi.uclm.es/www/cglez/downloads/docencia/2011_Softcomputing/LogicaDifusa.pdf.

[4] F. Sancho Caparrini, «Fernando Sancho Caparrini,Introducción a la Lógica Difusa,» [En línea]. Available: http://www.cs.us.es/~fsancho/?e=97.

[5] C. George Boeree, «Universidad de Shippensburg,» [En línea]. Available: http://webspace.ship.edu/cgboer/genesp/neuronas.html.

[6] K. Nahua, «Towards Data Science,» [En línea]. Available: https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f.

[7] O. Fontenla-Romero, E. Castillo, A. Alonso-Betanzos, J. Principe y D. Erdogmus, «Linear Least-Squares Based Method for Neural Networks Learning,» de *Artificial Neural Networks and Neural Information Processing*, Istanbul, 2003.

[8] H. Bautista Santos, J. L. Martínez Flores, G. Fernández Lambert, M. B. Bernábe Loranca, F. Sánchez-Galván y N. Sablón Cossío, «Integration model of collaborative supply chain,» *Dyna,* vol. 82, nº 192, pp. 145-154, 2015.

[9] . A. Adewunmi, *Selection Of Simulation Variance Reduction Techniques Through A Fuzzy Expert System.,* 2010.

[10] «tutorials point,» [En línea]. Available: https://www.tutorialspoint.com/fuzzy_logic/fuzzy_logic_inference_system.htm.

[11] R. Fuller, Introduction to Neuro-Fuzzy Systems, Physica-Verlag, 2000.

[12] S.-I. Lim, S.-J. Lee, M.-S. Choi, D.-J. Lim y B.-N. Ha, «Service Restoration Methodology for Multiple Fault Case in Distribution Systems,» *IEEE Transactions on Power Systems,,* vol. 21, nº 4, pp. 1638-1644, Noviembre 2006.

[13] L. Rutkowski, Computational Intelligence. Methods and Techinques, Czestochowa: Springer, 2005.

[14] Texas Instruments Designs, *2-kW, 48- to 400-V, >93% Efficiency, Isolated Bidirectional DC-DC Converter Reference Design for UPS,* 2017.

[15] Texas Instrument Designs, *CSD19536KCS 100 V N-Channel NexFET™ Power MOSFET,* JANUARY 2014.

[16] TEXAS INSTRUMENTS, «TEXAS INSTRUMENTS,» [En línea]. Available: https://www.ti.com/tool/TMDSDOCK28035?DCMP=c2x-f2803-6x&HQS=c2x-f2803-6x-pr-tf4-eu.

[17] TEXAS INSTRUMENTS, *TMS320F2803x Piccolo™ Microcontrollers,* JUNE 2020.

[18] L. Estrada Rojo, J. Ortega Alejos y J. E. E. González Duran, «GENERACIÓN DE UN PATRÓN PWM PARA UN CONVERTIDOR CD-CD BIDIRECCIONAL DOBLE PUENTE ACTIVO,» *Pistas Educativas,* vol. 39, nº 125, pp. 112-126, 2017.

[19] P. U. R y A. K. Rathore, «Extended Range ZVS Active-Clamped Current-Fed Full-Bridge Isolated DC/DC Converter for Fuel Cell Aplications: Analysis, Design, and Experimental Results,» *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS,* vol. 60, nº 7, JULY 2013.

[20] F. C. Kunrong Wang y L. Lee and J, «Operation principles of bi-directional full-bridge DC/DC converter with unified soft-switching scheme and soft-starting capability,» de *APEC 2000. Fifteenth Annual IEEE Applied Power Electronics Conference and Exposition*, New Orleans, LA, USA.

[21] B. Z. Q. S. W. L. y Y. S. , «Overview of Dual-Active-Bridge Isolated Bidirectional DC–DC Converter for High-Frequency-Link Power-Conversion System,» *IEEE Transactions on Power Electronics,* vol. 29, nº 8, pp. 4091-4106, Aug 2014.

[22] Q. Z. «Phase-Shifted Full-Bridge Converter with ZVZCS for Adjustable Output Voltage,» de *Intelec 2013; 35th International Telecommunications Energy Conference, SMART POWER AND EFFICIENCY*, Hamburg, Germany, 2013.

[23] E. Person, R. Ayyanar y N. Mohan, «Soft-Switching in DC-DC Converters: Principles, Practical Topologies, Design Techniques, Latest Developments,» 2002. [En línea]. Available: https://aboutme.samexent.com/classes/spring09/ee5741/SoftSwitching_Lecturenotes.pdf.

[24] S. Nagai, M. Nakaoka y A. Okuno, «High-frequency inverter with phase-shifted PWM and load-adaptive PFM control strategy for industrial induction-heating,» *Conference Record of the 1993 IEEE Industry Applications Conference Twenty-Eighth IAS Annual Meeting,* vol. 3, pp. 2165-2172, 1993.

[25] M. H. Rashid, Power Electronics. Deviced, Circuits, and Applications, PEARSON, 2014.

[26] F. Berzal, Redes Neuronales & Deep Learning - Volumen 1: Entrenamiento de redes neuronales artificiales, Independiente, 2019.

[27] R. Phadungthin y J. Haema, «Full bridge resonant inverter for non-ferrous metal induction heating application,» de *2015 IEEE PES Asia-Pacific Power and Energy Engineering Conference*, Brisbane, 2015.

[28] G. Xu y D. Sha, High-Frequency Isolated Bidirectional Dual Active Bridge DC-DC Converters with Wide Voltage Gain, Springer, 2019.

[29] H. W. Koertzen, J. D. Van Wyk y J. A. Ferreira, «Single shot surface heating series resonant converter with ZCS and an uncontrolled DC link voltage,» de *IAS '95. Conference Record of the 1995 IEEE Industry*

Applications Conference Thirtieth IAS Annual Meeting, Orlando, 1995.

[30] R. Pittini y M. A. E. Andersen, «Isolated full bridge boost DC-DC converter designed for bidirectional operation of fuel cells/electrolyzer cells in grid-tie applications,» de *2013 15th European Conference on Power Electronics and Applications (EPE)*, 2013, Lille.

[31] G. G. Koch y e. a. , «Design of a Robust PI Controller for a Dual Active,» de *016 12th IEEE International Conference on Industry Applications (INDUSCON)*, Curitiba, 2016.

[32] O. Yade, J. Gauthier, X. Lin-Shi, M. Gendrin y A. Zaoui, «Modulation strategy for a Dual Active Bridge converter using Model Predictive Control,» de *2015 IEEE International Symposium on Predictive Control of Electrical Drives and Power Electronics (PRECEDE)*, Valparaiso, 2015.

GLOSARIO

ANN: Aritifical Neural Network 11
ANNC:Artifical Neural Network 69
ANNI: Artificial Neural Network Inverse 69
CLA: Control Law Accelerator 72
COG: Center Of Gravity 23
CPU: Central Processing Unit 72
DAB: Dual Active Bridge 1
DC: Direct Current 38
DSP: Digital Signal Processor 2
eCAN: Enhanced Controller Area Network 72
FLC: Fuzzy Logic Controller 20
FLS: Fuzzy Neural Sistem 20
FNC: Fuzzy Neural Controller 28
FOM: First Of MaximaI 24
GPIO: General-Purpose Input Output 72
HV: Hight Voltage 40
I2C: Inter Integrate Circuit 72
IA: Inteligencia Artificial 11
LV: Low Voltage 46
MCU: Microcontroller Unit 71
MOSFET: Metal–Oxide–Semiconductor Field-Effect Transistor 5
PI: Prportional Integral 2
PID: Proportional Integral Derivative 41
PWM: Pulse Width Modulation 1
SCI: Serial Communications Interface 72
SISO: Single Input Single Output 11
SO: Sobreoscilación 47
SOC: Sistem On a Chip 72
SPI: Serial Peripheral Interface 72
UPS: Uninterruptible Power Supply 37
ZVZCS: Zero Voltage Zero Current Switching 5