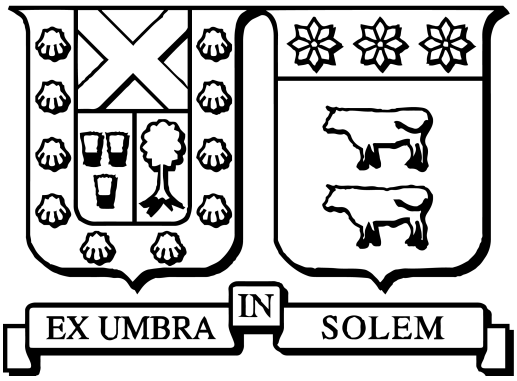
**Universidad Técnica Federico Santa María**
Departamento de Ingeniería Eléctrica
Valparaíso, Chile




# Diseño e implementación de una red neuronal de entrenamiento mixto para el control de un Flying Capacitor trifásico de tres celdas

**SOFIA ANETTE DAWSON ABD-EL-KADER**

**2025**

Requisito parcial para obtener el título de:
Ingeniero Electricista

Profesor Guía:
Dra. Margarita A. Norambuena Valdivia (UTFSM)

Profesor Co-guía:
Dr. Pablo A. Lezana Illesca (UTFSM)

Valparaíso, Julio 2025.

# CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

## 1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

**Tipo de monografía (marcar una opción):** ☒ Memoria o trabajo de título; ☐ Tesis de Postgrado;

**Título del trabajo:** Diseño e implementación de una red neuronal de entrenamiento mixto para el control de un Flying Capacitor trifásico de tres celdas

**Nombre del candidato(a):** Sofía Anette Dawson Abd-El-Kader

**Carrera / Grado:** Ingeniería Eléctrica

**Campus:** Casa Central Valparaiso **;** **Departamento**: Departamento de Ingeniería Eléctrica

## 2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Margarita Andréa Norambuena Valdivia, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución

## 3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL

☒ El trabajo **NO contiene información que amerite confidencialidad** y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por:

☐ 6 meses; ☐ 12 meses; ☐ 2 años; ☐ 3 años; ☐ 5 años; ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

## 4.- FIRMAS

**Profesor(a) guía o director(a) de memoria o tesis:**

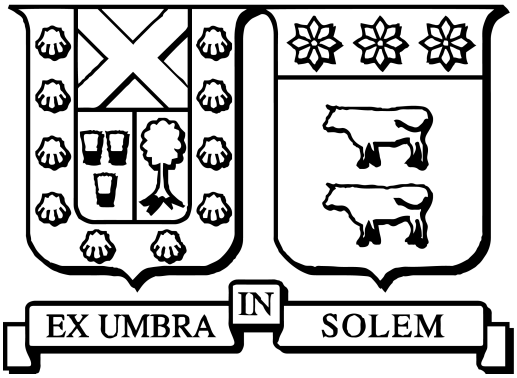
**Fecha: 04/08/2025** ; **Firma:** _____

**Estudiante o Candidato(a):**

**Fecha:** 03/08/2025 ; **Firma:** _____

*Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.*

**Universidad Técnica Federico Santa María**
Departamento de Ingeniería Eléctrica
Valparaíso, Chile




# Diseño e implementación de una red neuronal de entrenamiento mixto para el control de un Flying Capacitor trifásico de tres celdas

**SOFIA ANETTE DAWSON ABD-EL-KADER**

**2025**

*"El futuro pertenece a quienes creen en la belleza de sus sueños."*
*— Eleanor Roosevelt*

# Agradecimientos

Este logro es el resultado de un proceso que recorrí acompañada, y quiero agradecer sinceramente a quienes, de distintas formas, fueron parte de este camino.

Quiero agradecer profundamente a mis papás, Frank y Michelle, por estar siempre presentes, en los momentos buenos y en los más complejos. Gracias por su apoyo incondicional, por confiar en mí, y por enseñarme con el ejemplo lo que significan el esfuerzo, la responsabilidad y el compromiso. Nada de esto habría sido posible sin la estabilidad, el cariño y la confianza que me entregan cada día.

A mis hermanos, Pablo y Antonio, gracias por estar siempre atentos a mis logros y tropiezos, por su paciencia, su humor y su compañía a lo largo de este proceso. A mis tíos, Cristian y Carola, por sus palabras de aliento, su disposición y por recibirme en su casa con tanto cariño.

Al Beto, que estuvo en todo este proceso apoyándome y dándome ánimos en el día a día, recordándome siempre que sí podía.

A la profe Margarita, por haberme guiado de la mejor manera en este camino. Gracias por su paciencia, por creer en mí y por ser un referente no solo en lo académico, sino también en lo humano. También a todos los profesores que me acompañaron estos años con sus enseñanzas, exigencia y compromiso. Gracias por formar parte de mi formación.

A mis compañeros y amigos de eléctrica: Enrique, Dani, Benja, Jhon, Curicó, Gato, Cami, Pablo, Andrés y el Oyarzo. Gracias por las largas jornadas de estudio, las conversaciones, las risas, los desahogos y el compañerismo que marcó esta etapa. A la Luna, con quien compartí horas de tesis y con quien tuve el privilegio de acompañarnos en este cierre. Sé que todos tienen por delante un camino profesional lleno de logros, y me alegra haber coincidido con ustedes en esta etapa.

A la comunidad de Mujeres Eléctricas, por el apoyo constante, tanto académico como emocional, a lo largo de estos años. Fue un espacio de encuentro y crecimiento, donde encontré referentes, contención y amistades valiosas.

Y finalmente, a quienes ya no están, como mi abuela Mabel, pero dejaron una huella profunda en mi vida. Este logro también refleja todo lo que me enseñaron con su amor, sus valores y la confianza que depositaron en mí desde siempre. Gracias.

# Índice de contenidos

# Índice de figuras

# Índice de tablas

# Nomenclatura

| | | |
|---|---|---|
| $x$ | : | Valor temporal o variable de la cantidad $x$. |
| $x^k$ | : | Valor en el instante $k$ de la cantidad $x$. |
| $x^*$ | : | Valor de referencia para la cantidad $x$. |
| $\mathbf{x}$ | : | Vector de valores de $x$. |
| $a,\ b,\ c$ | : | Subíndices utilizados para las fases $a$, $b$ y $c$. |
| $o$ | : | Neutro de la carga. |
| $N$ | : | Neutro del inversor. |
| $R$ | : | Resistencia de la carga. |
| $L$ | : | Inductancia de la carga. |
| $C_{ij}$ | : | Capacitancia del capacitor de la celda $i$ de la fase $j$. |
| $v_{ij}$ | : | Tensión en el capacitor de la celda $i$ de la fase $j$. |
| $m_i$ | : | Índice de modulación de la fase $i$. |
| $S_{ij}$ | : | Estado de conmutación de la celda $i$ de la fase $j$. |
| $SS_{ij}$ | : | Estado de conmutación parametrizado de la celda $i$ de la fase $j$. |
| $V_{dc}$ | : | Tensión principal del enlace de corriente continua (dc-link). |
| $v_{xN}$ | : | Tensión entre la fase $x$ y el neutro ($x \in \{a, b, c\}$). |
| $i_x$ | : | Corriente de salida por fase ($x \in \{a, b, c\}$). |
| $i_{ref,x}$ | : | Corriente de referencia para la fase $x$. |
| $v_{ref,x}$ | : | Tensión de referencia para la fase $x$. |
| $v_{1x},\ v_{2x}$ | : | Tensiones de los capacitores flotantes de las celdas de la fase $x$. |
| $\Delta H$ | : | Banda de histéresis aplicada al control. |
| $\eta$ | : | Eficiencia del sistema. |
| $\tau$ | : | Constante de tiempo asociada a la respuesta del sistema. |
| $E_{loss}$ | : | Energía disipada en pérdidas. |
| $\alpha,\ \beta$ | : | Coordenadas del sistema de referencia $\alpha\beta$. |
| $\phi$ | : | Ángulo de fase. |
| $\omega$ | : | Frecuencia angular del sistema. |
| $f_{sw}$ | : | Frecuencia de conmutación del convertidor. |
| $J[k]$ | : | Función de costo en el instante $k$. |
| $\mathbf{P}$ | : | Matriz de pesos utilizada en la función de costo. |

# Resumen

En la electrónica de potencia, los convertidores multinivel, como el de capacitores flotantes (FCC, del inglés Flying Capacitor Converter), destacan por su eficiencia y alta calidad en las formas de onda, aunque presentan desafíos complejos en su control, como la necesidad de mantener el balance de tensiones entre capacitores en tiempo real, evitando condiciones que comprometan su estabilidad. Las estrategias tradicionales, tales como el Control Proporcional-Integrativo (PI) y el Control Predictivo basado en Modelo (MPC, del inglés: Model Predictive Control), aunque son efectivas, enfrentan limitaciones en adaptabilidad y costo computacional, pues el control PI resulta poco adecuado para sistemas multivariables con dinámicas rápidas, mientras que el MPC, si bien ofrece una mejor dinámica de respuesta, presenta error en estado estacionario y además exige una capacidad de cómputo considerable para resolver optimizaciones en línea, lo que restringe su implementación en entornos con recursos limitados. Esto ha motivado la exploración de enfoques de control más flexibles e inteligentes, como las redes neuronales artificiales (ANN, del inglés: Artificial Neural Network).

El presente trabajo se centra en el diseño e implementación de una estrategia de control avanzada basada en ANN para un convertidor FCC de tres celdas. A partir del estudio de controles tradicionales como el PI y el MPC, se desarrollaron dos enfoques novedosos: el primero constó de la integración de dos redes neuronales basadas en control lineal y control MPC, mediante un control por bandas de histéresis, y el segundo en una única red neuronal dual que combina características de ambos controles. Estos métodos fueron evaluados en simulaciones y en un prototipo experimental, evaluando su desempeño en términos de comportamiento en transitorios y estado estacionario, mediante el análisis de las corrientes y tensiones de salida.

Los resultados muestran que la propuesta con ANN Dual se destaca por sobre la propuesta de unión de ANN mediante histéresis, principalmente debido a su adaptabilidad en el entorno experimental y a las limitantes de las redes neuronales utilizadas para el control de unión de ANN con histéresis. Los resultados confirman que la ANN Dual cumple con los objetivos planteados, mostrando un desempeño competitivo tanto en simulación como en aplicaciones reales. Sin embargo, se identificaron áreas de mejora, principalmente en el proceso de entrenamiento de la red, para aumentar su adaptabilidad a convertidores con parámetros variables y mejorar la calidad de las señales de salida. Este enfoque demuestra el potencial de las ANN para reemplazar y optimizar los controles tradicionales, reduciendo el costo computacional y ofreciendo sistemas más robustos y versátiles para aplicaciones avanzadas en la electrónica de potencia.

# Capítulo 1

# Introducción

La electrónica de potencia es fundamental para adaptar y convertir la energía eléctrica, permite hacer uso de tecnologías tales como la generación fotovoltaica, los bancos de baterías, la transmisiónen HVDC y el control de máquinas eléctricas. Los convertidores de potencia son dispositivos capaces de alterar las características de tensión y corriente de entrada y/o salida, con tal de transformar la energía de manera optimizada entre AC-DC, DC-DC, AC-AC o DC-AC según los usos que sean requeridos. Éstos han sido altamente estudiados en las últimas décadas con la finalidad de obtener desempeños cada vez más favorables, tanto con mejores mecanismos de control como con topologías más eficientes[1]. Los convertidores capaces de realizar la conversión de DC a AC se denominan inversores. Uno de los inversores más utilizados para obtener formas de tensión y/o corriente con mayor calidad y con una característica más cercana a la sinusoidal es el de tecnología multinivel; en estos, la tensión de salida se construye mediante varios escalones discretos, cuya cantidad depende del número de niveles del convertidor utilizado [2].

En el campo de la electrónica de potencia, la evolución de las estrategias de control ha sido crucial para mejorar la eficiencia y estabilidad de los convertidores. Existen numerosas estrategias de control para convertidores, siendo una de las más habituales el control lineal con Modulación por Ancho de Pulsos (PWM, por sus siglas en inglés: Pulse Width Modulation)[2] que, por ejemplo, para controles del tipo PI tienen una alta ganancia a continua, asegurando un desempeño óptimo en estado estacionario al presentar error estacionario nulo incluso ante errores en el modelo [3].
Por otro lado, se tiene uno de los tipos de control más influyentes de los últimos años, el Control MPC con Conjunto de Estados Finitos (FCS, del inglés: Finite Control Set) que se caracteriza por su simplicidad conceptual, sencillo tratamiento de no-linealidades y buen seguimiento de la referencia, teniendo un comportamiento ventajoso en transitorios. Dentro de sus limitaciones se tiene que considera todos los estados de conmutación posibles en la implementación del algoritmo, por lo que puede implicar alto costo computacional según la topología del convertidor a controlar [3]-[4].

Últimamente se han desarrollado estrategias que apuntan a tener sistemas de control inteligentes y capaces de adaptarse a diversos escenarios, presentándose el desafío de diseñar sistemas de control eficientes y aptos para controlar múltiples variables con dinámicas de respuesta más rápidas. Una de las posibles soluciones es la implementación de redes neuronales artificiales basadas en sistemas de control, para lograr una herramienta

robusta tanto en comportamientos dinámicos como estacionarios[5]. Presentando ciertas ventajas respecto a los controladores tradicionales, tales como el bajo costo computacional que requiere para evaluar los posibles estados de conmutación y también, que para realizar el control no requiere de modelos matemáticos[4]. Sin embargo, se necesita de un sistema patrón que se utiliza para el proceso de entrenamiento de la red neuronal el cuál sí requiere del uso de modelos matemáticos.

En este contexto, las redes neuronales artificiales emergen como una alternativa prometedora. Su capacidad para aprender y generalizar comportamientos sin necesidad de un modelo explícito del sistema, junto con su adaptabilidad, las hace ideales para implementar estrategias de control avanzadas en convertidores de potencia. Este trabajo de memoria se centra en la aplicación de ANN para emular y optimizar las estrategias de control PI y FCS-MPC de manera dual en un convertidor trifásico de condensadores flotantes de cuatro niveles.

Se exploran dos posibles enfoques para implementar un control dual. El primero consiste en combinar dos redes neuronales artificiales: una basada en el control PI y otra en el control FCS-MPC. Estas redes se coordinan mediante un esquema de bandas de histéresis, que selecciona el control óptimo dependiendo de si el sistema se encuentra en estado estacionario o transitorio. El segundo enfoque implica el uso de una única RNA entrenada de manera conjunta, incorporando ambos tipos de control en su proceso de aprendizaje.

A través de la simulación y la implementación en un prototipo de laboratorio, se evalúa la capacidad de la ANN para reemplazar y mejorar las técnicas de control tradicionales, con el objetivo de reducir el costo computacional y mantener un rendimiento óptimo en el control de un convertidor FCC.

# Capítulo 2

# Convertidor Flying Capacitor de cuatro niveles

Los inversores son equipos capaces de transformar la entrada de energía con corriente continua (DC) en una salida de corriente alterna (AC), siendo esenciales en áreas de energía solar, sistemas de respaldo y electro-movilidad. A lo largo del tiempo, se han desarrollado múltiples topologías, destacando dentro de la tecnología multinivel el uso del Flying Capacitor (FCC) [6].

El FCC destaca por su capacidad de generar tensiones escalonadas mediante celdas conectadas en tandem. Cada celda, la unidad básica del FCC, está compuesta por un capacitor flotante y un par de semiconductores que operan de manera complementaria de modo de evitar cortocircuitos en los capacitores [5]. A medida que se añaden más celdas en tandem, el número de niveles de tensión aumenta en proporción a la relación: *niveles = celdas + 1*. Esto se traduce en una mejora significativa en la calidad de la forma de onda de salida, reduciendo también la distorsión armónica de la misma.

Entre las principales ventajas de esta topología destacan su capacidad para gestionar tanto la potencia activa como la reactiva, además de no requerir diodos de interconexión o clamped, a diferencia de la topología Neutral Point Clamped (NPC) con un número similar de niveles. Los condensadores en cada celda permiten almacenar más energía, lo que mejora la respuesta ante situaciones transitorias. Por otro lado, las desventajas incluyen el aumento en el número y tamaño de los capacitores a medida que se incrementan los niveles, lo que genera mayores costos y riesgos de fallos. Asimismo, el sistema de control debe ser especialmente cuidadoso para equilibrar la tensión de los capacitores flotantes, y el capacitor principal debe estar diseñado para soportar la tensión máxima del FCC, por lo que en casos de tensiones elevadas se requiere de un arreglo de capacitores [5]-[7].

En este trabajo, se utilizó un FCC de tres celdas por fase, como se muestra en la figura 2.1. El FCC trifásico está conectado a una carga $R$-$L$ configurada en estrella, y alimentado por un enlace de corriente continua (dc-link), tal como se observa en la figura 2.2. Su principio de funcionamiento se basa en la imposición de una tensión continua $+V_{dc}$, junto con la implementación de un esquema de control en los semiconductores. Esto genera diversas combinaciones de estados, dependiendo del balance de tensiones en los

capacitores, lo que permite la carga y descarga cíclica de los mismos. La tensión de salida, que presenta cuatro niveles, se genera entre los puntos $x$ y $N$.

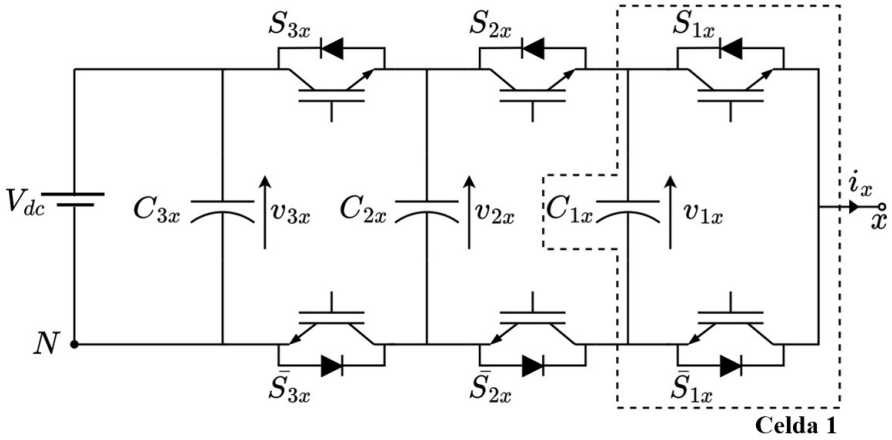



Figura 2.1: FCC de 4 niveles[7].

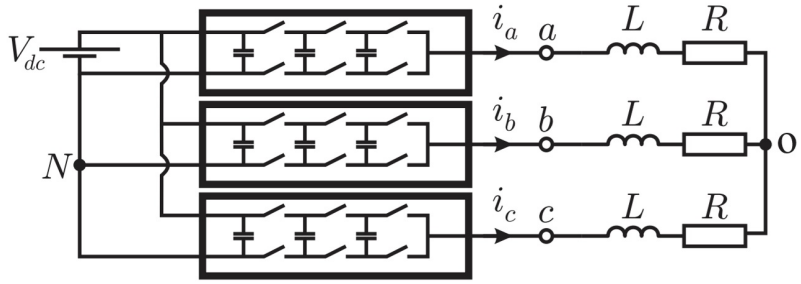



Figura 2.2: FCC trifásico de 4 niveles con carga $R$-$L$ configurada en estrella[3].

Para distribuir uniformemente la tensión de bloqueo entre los semiconductores, las tensiones de los capacitores flotantes deben cumplir:

$$v_{3x} : v_{2x} : v_{1x} = 3 : 2 : 1, \qquad x \in \{a, b, c\} \tag{2.1}$$

con $v_{3x} = V_{dc}$.

La tabla 2.1 muestra las 8 combinaciones posibles de conmutación para un convertidor FCC de tres celdas. Si se cumple la ecuación (2.1), $v_{xN} \in 0, \frac{1}{3}V_{dc}, \frac{2}{3}V_{dc}, V_{dc}$, es decir, la tensión de salida puede alcanzar hasta cuatro niveles, mientras que cada semiconductor bloquea $\frac{1}{3}V_{dc}$. Por lo tanto, los objetivos de control para esta topología son las tensiones de los capacitores flotantes $v_{1x}$, $v_{2x}$ y la corriente de salida $i_x$ (típicamente sinusoidal), lo que resulta en un total de tres variables por fase en este caso[5]. Así, la desviación del sistema es:

$$J[k] = \begin{bmatrix} v_{1x}^{ref} - v_{1x} \\ v_{2x}^{ref} - v_{2x} \\ i_x^{ref} - i_x \end{bmatrix}^T P \begin{bmatrix} v_{1x}^{ref} - v_{1x} \\ v_{2x}^{ref} - v_{2x} \\ i_x^{ref} - i_x \end{bmatrix} \tag{2.2}$$

Donde $P$ es una matriz diagonal de números positivos que corresponden a factores de ponderación usados para distribuir el esfuerzo de control entre las distintas variables de interés.

La tensión instantánea de la carga para el convertidor de la figura 2.2 se puede calcular de la siguiente manera:

$$v_{xN}(t) = Ri_x(t) + L\frac{di_x(t)}{dt} + v_{oN}(t) \tag{2.3}$$

Donde la tensión de modo común está descrita por:

$$v_{oN}(t) = \frac{v_{aN}(t) + v_{bN}(t) + v_{cN}(t)}{3} \tag{2.4}$$

Los valores de $v_{xN}$ se obtienen mediante la siguiente ecuación:

$$v_{xN}(t) = S_{3x}(t)V_{dc}(t) + \Big(S_{2x}(t) - S_{3x}(t)\Big)v_{2x}(t) + \Big(S_{1x}(t) - S_{2x}(t)\Big)v_{1x}(t) \tag{2.5}$$

Con las tensiones de los capacitores:

$$v_{1x}(t) = \frac{1}{C_{1x}}\int_0^t i_{1x}(\tau)d\tau + v_{1x}(0) \tag{2.6}$$

$$v_{2x}(t) = \frac{1}{C_{2x}}\int_0^t i_{2x}(\tau)d\tau + v_{2x}(0) \tag{2.7}$$

Finalmente, la corriente en cada capacitor flotante se describe por la relación:

$$i_{1x}(t) = i_x(t)\Big(S_{2x}(t) - S_{1x}(t)\Big) \tag{2.8}$$

$$i_{2x}(t) = i_x(t)\Big(S_{3x}(t) - S_{2x}(t)\Big) \tag{2.9}$$

Es importante destacar que las ecuaciones anteriores describen un sistema no lineal debido a la interacción de los estados del sistema (corrientes y tensiones flotantes) y los estados de los interruptores.

Tabla 2.1: Estados de conmutación posibles para el FCC de tres celdas.

| $S_{3x}$ | $S_{2x}$ | $S_{1x}$ | $v_{xN}(t)$ | $i_{1x}(t)$ | $i_{2x}(t)$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | $v_{1x}(t)$ | $-i_x(t)$ | 0 |
| 0 | 1 | 0 | $v_{2x}(t)$ - $v_{1x}(t)$ | $i_x(t)$ | $-i_x(t)$ |
| 0 | 1 | 1 | $v_{2x}(t)$ | 0 | $-i_x(t)$ |
| 1 | 0 | 0 | $V_{dc} - v_{2x}(t)$ | 0 | $i_x(t)$ |
| 1 | 0 | 1 | $V_{dc} - v_{2x}(t) + v_{1x}(t)$ | $-i_x(t)$ | $i_x(t)$ |
| 1 | 1 | 0 | $V_{dc} - v_{1x}(t)$ | $i_x(t)$ | 0 |
| 1 | 1 | 1 | $V_{dc}$ | 0 | 0 |

# Capítulo 3

# Estrategias de control

En este capítulo se exploran los fundamentos de cuatro tipos de control relevantes para esta memoria, a modo de controlar el FCC: control PI, control FCS-MPC, control dual (PI sumado a FCS-MPC) y control basado en redes neuronales (ANN).

## 3.1. Control PI

El control lineal es una técnica que utiliza ecuaciones lineales para modelar y regular el comportamiento de sistemas dinámicos. En este sentido, la relación entre las variables de entrada y salida se representa mediante funciones lineales, lo que facilita el análisis y diseño del sistema. Dentro de la familia de los controladores PID *(Proporcional-Integral-Derivativo)* se encuentra el control PI a considerar en este trabajo.

Considerando el diagrama de control en la figura 3.1, la expresión estándar para el control PI se describe mediante la función de transferencia entre el error relativo $e(s) = r(s) - y(s)$ y la salida del controlador $u(s)$[7].

$$C_{PI}(s) = K_p \left(1 + \frac{1}{T_r \cdot s}\right) \tag{3.1}$$

donde $K_p$ es la constante proporcional, y $T_r$ es el tiempo de reinicio o constante de acción integral del control.

La componente integral garantiza un error en estado estacionario nulo para valores constantes de $r(s)$.

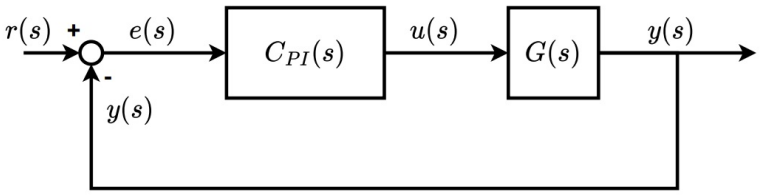



Figura 3.1: Esquema básico del control PI[7].

Los controladores lineales requieren un modelo de sistema lineal para ser ajustados. Como se mencionó anteriormente, el FCC es un sistema no lineal, sin embargo, (2.3)

establece una relación lineal entre la tensión de salida del convertidor y la corriente de salida. Luego, para usar controladores PI, el sistema trifásico definido por (2.3) puede reescribirse en el marco de referencia $dq$ según las siguientes ecuaciones [8].

$$v_d(t) = Ri_d(t) + L\frac{di_d(t)}{dt} \Leftrightarrow G_d(s) = \frac{I_d(s)}{V_d(s)} = \frac{1}{Ls + R} \tag{3.2}$$

$$v_q(t) = Ri_q(t) + L\frac{di_q(t)}{dt} \Leftrightarrow G_q(s) = \frac{I_q(s)}{V_q(s)} = \frac{1}{Ls + R} \tag{3.3}$$

donde $s$ es el operador de Laplace.

Los controladores lineales generan salidas de control continuas, que deben transformarse en estados de conmutación mediante moduladores de ancho de pulso (PWM) para generar las tensiones de salida deseadas. La técnica más común para el FCC es la modulación de ancho de pulso con desplazamiento de fase (PS-PWM, del inglés: Phase Shifted-PWM), ya que es capaz de mantener el balance natural de tensión en los capacitores flotantes [9]. El PS-PWM requiere señales portadoras triangulares, una para cada celda, de igual amplitud y frecuencia pero desplazadas entre sí. El desfase entre triangulares viene dado según la expresión:

$$\theta = \frac{2\pi}{n} \tag{3.4}$$

Donde $n$ es el número de celdas.

Para el FCC a estudiar se tiene que $\theta_{1x} = 0$, $\theta_{2x} = 2\pi \cdot 1/3$, y $\theta_{3x} = 2\pi \cdot 2/3$, quedando las señales de la figura 3.2.

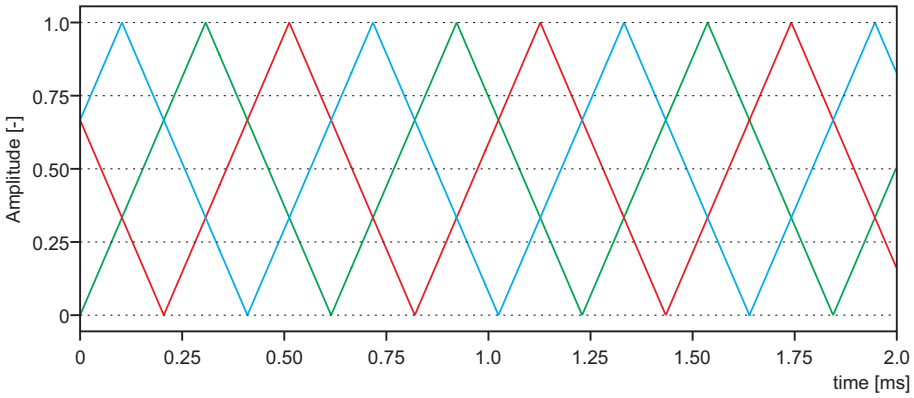



Figura 3.2: Arreglo de señales portadoras triangulares.

El esquema del control lineal aplicado al FCC de estudio se presenta en la figura 3.3 en tiempo discreto, incluyendo un esquema antienrollamiento y un limitador de tensión.

El FCC se modela como un retardo de primer orden con una constante de tiempo $h$, siendo la función de transferencia de la forma:

$$G_{PWM}(s) = e^{-sh}\frac{V_{dc}}{2} \tag{3.5}$$

Mientras que la función de transferencia de la carga se modela como:

$$G(s) = \frac{1}{sL + R} \tag{3.6}$$

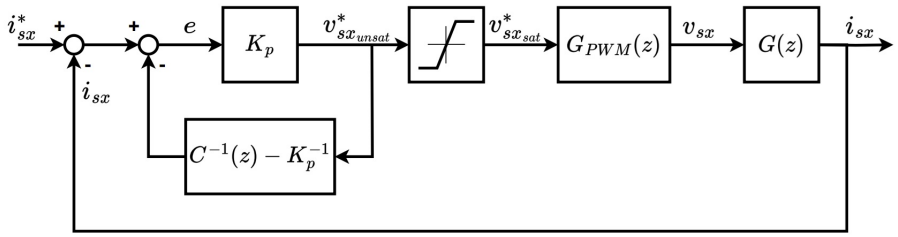



Figura 3.3: Esquema del control PI aplicado[7].

## 3.2. Control FCS-MPC

El control predictivo basado en modelo es una técnica avanzada de control que utiliza un modelo matemático del sistema para predecir su comportamiento futuro y optimizar su rendimiento. A diferencia de los métodos de control tradicionales, el MPC calcula la acción de control actual anticipando cómo las decisiones afectarán al sistema en un horizonte de tiempo definido, permitiendo manejar restricciones y mejorar el desempeño en sistemas complejos y multivariables.

El FCS-MPC es un tipo de estrategia de control que opera prediciendo el comportamiento futuro de un sistema para todas las posibles combinaciones finitas de conmutación y determinando la acción de control óptima que se mantendrá durante todo el período de muestreo. Por lo tanto, no se requiere de un modulador externo[3].

El algoritmo evalúa todos los estados de conmutación posibles hasta escoger el primer vector de estados de conmutación que minimiza la función de costo. Esta función de costo define el comportamiento objetivo del sistema comparando los valores predichos con los valores de referencia. Para el FCC de 3 celdas, la función de costo se define como:

$$g^k = \sum_{x=a}^{c} \left( (i_x^\star - i_x^{k+2})^2 + \lambda_1 (v_{c1}^\star - v_{c1x}^{k+2})^2 + \lambda_2 (v_{c2}^\star - v_{c2x}^{k+2})^2 \right) \tag{3.7}$$

donde $\lambda_1$ y $\lambda_2$ son factores de ponderación que distribuyen el esfuerzo de control entre las tensiones flotantes y las corrientes de salida, $i_x^{k+2}$, $v_{c2x}^{k+2}$ y $v_{c1x}^{k+2}$ son los valores predichos de las corrientes de salida y las tensiones de los capacitores flotantes, respectivamente. Finalmente, $i_x^\star$ son las referencias de corriente de salida, mientras que $v_{c2x}^\star$ y $v_{c1x}^\star$ son las referencias de tensión flotante según la proporción definida en (2.1), aunque podrían ser seleccionadas otras razones[10].

Para realizar las predicciones, las corrientes de salida y las tensiones flotantes se miden en el instante $k$. Estas mediciones y la combinación óptima de conmutación predefinida se utilizan para estimar la corriente del sistema y las tensiones flotantes en $k+1$ a través de la versión discretizada de (2.3)-(2.9):

$$v_{xN}^{k} = S_{3x}^{k} V_{dc} - (S_{3x}^{k} - S_{2x}^{k}) v_{2x}^{k} - (S_{2x}^{k} - S_{1x}^{k}) v_{1x}^{k} \tag{3.8}$$

$$v_{oN}^{k+1} = \frac{v_{aN}^{k+1} + v_{bN}^{k+1} + v_{cN}^{k+1}}{3} \tag{3.9}$$

$$i_{x}^{k+1} = \left(v_{xN}^{k} - v_{oN}^{k}\right) \frac{T}{L} + i_{x}^{k} \left(1 - R\frac{T}{L}\right) \tag{3.10}$$

$$v_{1x}^{k+1} = \frac{T}{2C_{1x}} (i_{x}^{k+1} + i_{x}^{k})(S_{2x}^{k} - S_{1x}^{k}) + v_{1x}^{k} \tag{3.11}$$

$$v_{2x}^{k+1} = \frac{T}{2C_{2x}} (i_{x}^{k+1} + i_{x}^{k})(S_{3x}^{k} - S_{2x}^{k}) + v_{2x}^{k} \tag{3.12}$$

Donde $T$ es el intervalo de muestreo.

Una vez que se calculan los valores estimados en $k+1$, los valores del sistema pueden predecirse para $k+2$ para todas las combinaciones posibles del convertidor, adaptando (3.8)-(3.12):

$$v_{xN}^{k+1} = S_{3x}^{k+1} V_{dc} - (S_{3x}^{k+1} - S_{2x}^{k+1}) v_{2x}^{k+1} - (S_{2x}^{k+1} - S_{1x}^{k+1}) v_{1x}^{k+1} \tag{3.13}$$

$$v_{oN}^{k+2} = \frac{v_{aN}^{k+2} + v_{bN}^{k+2} + v_{cN}^{k+2}}{3} \tag{3.14}$$

$$i_{x}^{k+2} = \left(v_{xN}^{k+1} - v_{oN}^{k+1}\right) \frac{T}{L} + i_{x}^{k+1} \left(1 - R\frac{T}{L}\right) \tag{3.15}$$

$$v_{1x}^{k+2} = \frac{T}{2C_{1x}} (i_{x}^{k+2} + i_{x}^{k+1})(S_{2x}^{k+1} - S_{1x}^{k+1}) + v_{1x}^{k+1} \tag{3.16}$$

$$v_{2x}^{k+2} = \frac{T}{2C_{2x}} (i_{x}^{k+2} + i_{x}^{k+1})(S_{3x}^{k+1} - S_{2x}^{k+1}) + v_{2x}^{k+1} \tag{3.17}$$

Es relevante notar que las ecuaciones (3.13)-(3.17) deben evaluarse hasta 512 veces debido a la alta cantidad de combinaciones posibles, ya que existen 8 estados posibles por fase y, al tratarse de un sistema trifásico, se generan $8^3 = 512$ configuraciones distintas; esto provoca que la carga computacional sea considerablemente elevada. Además, es importante destacar que el FCS-MPC seleccionará los estados de conmutación óptimos para minimizar la función de costo (3.7) en cada tiempo de muestreo. Esto a menudo resulta en patrones de conmutación de alta dispersión, lo que a su vez conduce a un espectro de señales más amplio y una distribución desigual de las pérdidas en los semiconductores.

## 3.3. Control Dual con Histéresis

Considerando que el control PI disminuye su rendimiento en sistemas con dinámicas complejas, no lineales y durante transitorios, teniendo limitaciones para adaptarse a variaciones bruscas en la referencia, y que, por otro lado, el control MPC tiene la capacidad para predecir el comportamiento futuro del sistema, ofreciendo una respuesta dinámica óptima pero con desventajas tales como su alto costo computacional y la tendencia a generar armónicos más dispersos, se propone el enfoque de la estrategia de control dual en los trabajos [3]-[5].

Esta estrategia combina el control FCS-MPC con el control PI, con el objetivo de aprovechar las fortalezas de ambos: una alta respuesta dinámica (FCS-MPC) y un excelente comportamiento en estado estacionario (PI-PWM) en cuanto a estabilidad y eficiencia del sistema. Se propone el uso de una banda de histéresis para alternar entre ambos esquemas, basada en la desviación del sistema definida en (3.18).

$$J^k = \sum_{x=a}^{c} \Big[ \gamma_i (i_x^\star - i_x^{k+2})^2 + \gamma_v \big( (v_{c1}^\star - v_{c1x}^{k+2})^2 + (v_{c2}^\star - v_{c2x}^{k+2})^2 \big) \Big] \tag{3.18}$$

donde $\gamma_i$ y $\gamma_v$ son constantes de ponderación para la desviación de la corriente y la tensión, respectivamente.

El mayor desafío de este enfoque es lograr una transición suave entre ambos controladores, especialmente la transición de FCS-MPC a PI-PWM. Para ello, el PI se implementa en su forma de retroalimentación, de manera que los estados internos del controlador se actualizen permanentemente con la actuación del FCS-MPC[3], lo cual es un aspecto clave de esta propuesta. La figura 3.4 muestra el esquema completo del control dual PI-MPC.

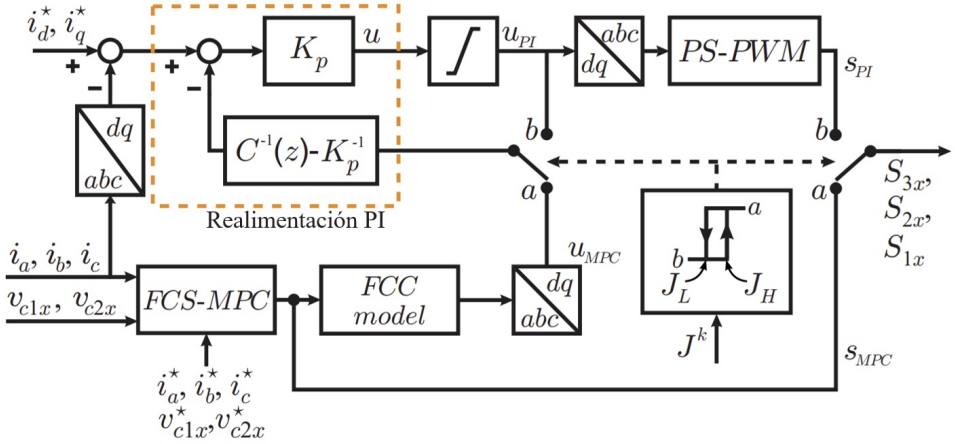



Figura 3.4: Esquema del control dual: PI-MPC.

Siendo $J_H$ la banda superior y $J_L$ la banda inferior, se tiene que si $J^k > J_H$, lo que indica una alta desviación del sistema debido a un cambio en la referencia o una perturbación, se utiliza FCS-MPC para mover rápidamente el sistema hacia la vecindad de la referencia. Una vez que $J^k < J_L$, el sistema es controlado por el esquema PI-PWM, volviendo suavemente a la operación en estado estacionario, con un error en estado estacionario nulo. Entre bandas, el sistema sigue el comportamiento mostrado en la figura 3.5, donde el control activo corresponde al de la última banda que el sistema cruzó.

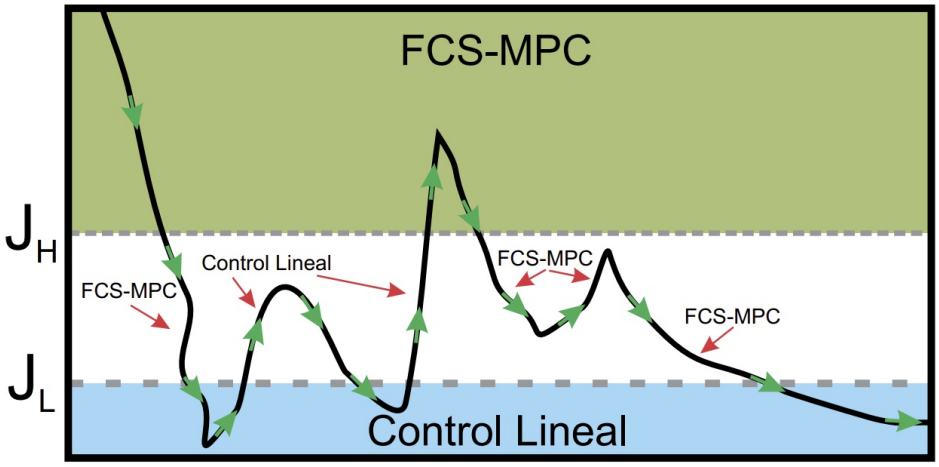



Figura 3.5: Bandas $J_H$ y $J_L$ para el selector por histéresis[5].

## 3.4. Redes Neuronales

Las redes neuronales artificiales (ANN, por sus siglas en inglés: Artificial Neural Network) son una tecnología emergente que ha generado un impacto significativo en diversas áreas de aplicación científica e ingenieril. Destacan por su capacidad para resolver problemas complejos, como el reconocimiento de patrones y el procesamiento de imágenes, que resultan difíciles de resolver mediante métodos tradicionales en computadoras digitales. Su capacidad de aprendizaje y adaptación las hace especialmente valiosas en entornos donde las soluciones convencionales son insuficientes o ineficientes[11].

### 3.4.1. Estructura y modelo

La neurona artificial es la unidad básica de la ANN y es fundamental para el procesamiento de información. Cada neurona recibe una serie de entradas que combina de forma ponderada, aplicando una función de activación, y produciendo así una salida según lo mostrado en la figura 3.6.

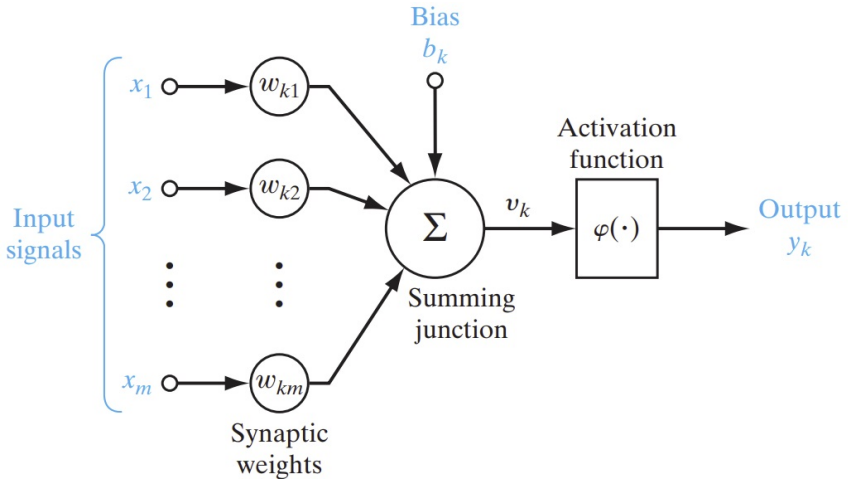



Figura 3.6: Modelo no lineal de una neurona artificial[12].

El modelo neuronal de la figura 3.6 también incluye un sesgo, o también llamado bias, aplicado externamente denotado por $b_k$. Este tiene el efecto de aumentar o disminuir la entrada neta de la función de activación. En términos matemáticos, se puede describir la neurona $k$ mediante las siguientes ecuaciones:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{3.19}$$

$$y_k = \varphi(u_k + b_k) \tag{3.20}$$

donde $x_1, x_2, \ldots, x_m$ son las señales de entrada; $w_{k1}, w_{k2}, \ldots, w_{km}$ son los respectivos pesos de la neurona $k$, que ajustan la importancia de cada entrada de la neurona; $u_k$ (no mostrado en la figura 3.6) es la salida combinada lineal debido a las señales de entrada; $b_k$ es el sesgo; $\varphi(\cdot)$ es la función de activación; e $y_k$ es la señal de salida de la neurona[12].

La función de activación transforma la suma ponderada de las entradas en una salida, que generalmente es no lineal, induciendo así no linealidades en el modelo. Esto es fundamental para que la red neuronal pueda aprender y caracterizar relaciones complejas entre las entradas y las salidas. Aunque la función de activación puede ser lineal, el verdadero potencial radica en su carácter no lineal, ya que esto evita que la ANN se convierta en un sistema linealmente dependiente. La introducción de no linealidades asegura que las neuronas no se reduzcan a una sola combinación lineal, y permite que cada neurona se especialice en aprender diferentes características de los datos[7].

Entre las funciones de activación más populares se tiene la *sigmoide* y la *tangente hiperbólica (tanh)*, que comprimen las salidas en un rango acotado y son útiles en redes profundas o de múltiples capas ocultas. La *ReLU (Rectified Linear Unit)*, por otro lado, ha ganado popularidad debido a su simplicidad y eficacia en disminuir el problema del desvanecimiento del gradiente. Además, existen variantes como *Leaky ReLU* o *ELU* que abordan las limitaciones de la *ReLU* clásica[12].

Las ANN se estructuran en capas, siendo las más comunes: a) Capa de entrada: donde se reciben los datos del sistema a modelar, con cada nodo representando una variable de entrada (por ejemplo, valores de corriente). b) Capas ocultas: ubicadas entre la capa de entrada y la de salida, están compuestas por múltiples neuronas que procesan las entradas utilizando pesos, sesgos (biases) y funciones de activación. Son responsables de captar las relaciones complejas entre los datos. c) Capa de salida: proporciona los resultados finales del procesamiento, y el número de neuronas en esta capa depende del número de salidas que se desee obtener del modelo. En la figura 3.7 se muestra un diagrama básico de la estructura de una red neuronal.

Las redes se pueden clasificar en varias estructuras fundamentales según la forma en que están organizadas sus neuronas y cómo se propagan los datos y señales de entrenamiento a través de la red. Las clases más comunes son las de tipo *feedforward (FFNN)* y las *recurrentes (RNN)*. Las redes *feedforward* son las más simples, donde la información fluye en una sola dirección desde la entrada hasta la salida, siendo convenientes para tareas de clasificación y regresión. Por otro lado las redes neuronales *recurrentes* incluyen conexiones internas que contienen lazos de retroalimentación entre capas, lo que las hace ideales para procesar secuencias de datos, como en el análisis de series temporales y el procesamiento de lenguaje natural[7]-[13].

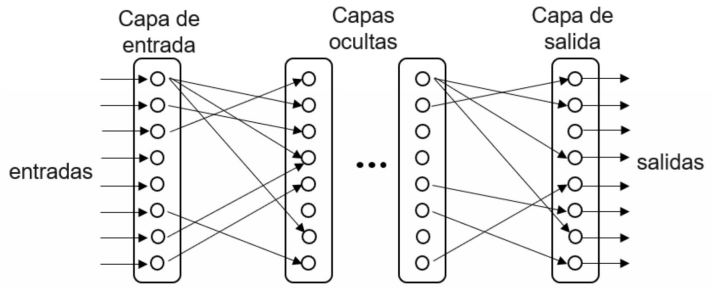



Figura 3.7: Esquema de una ANN.

### 3.4.2. Entrenamiento

El entrenamiento de una ANN se puede definir como el proceso en que la red ajusta sus parámetros internos (pesos y biases) para minimizar el error entre las salidas esperadas y las reales. Este se realiza utilizando un conjunto de datos de entrenamiento y sigue un principio iterativo por determinada cantidad de épocas, donde la red aprende a modelar las relaciones entre las entradas y las salidas. Durante cada época, además de ajustarse los pesos, la red evalúa su rendimiento en el conjunto de entrenamiento. A medida que se desarrolla el entrenamiento, la red actualizada debería mejorar progresivamente e ir reduciendo el error.

El algoritmo *Backpropagation* es un método utilizado en ANN de tipo *feedforward* para calcular de manera eficiente el gradiente de una función de costo con respecto a los parámetros del modelo. Es ampliamente empleado para entrenar redes multicapa [13], y ha encontrado aplicaciones relevantes en el campo de la electrónica de potencia y los sistemas de accionamientos [14].

La información se transmite desde los nodos de la capa de entrada hacia las capas ocultas y de salida, considerando el procesamiento con los pesos y biases según lo mostrado anteriormente en la figura 3.6. Al final de esta fase la red genera una salida que se compara con la salida esperada para calcular el error. Por otro lado, se tiene la retropropagación del error, donde el error calculado en la capa de salida se propaga hacia atrás a través de la red, pasando por las capas ocultas en dirección a la capa de entrada, actualizando los pesos para mejorar la precisión[13] [14].

Los paradigmas de aprendizaje definen diferentes formas en las que los modelos adquieren conocimiento a partir de los datos. Cada paradigma tiene sus propias características y aplicaciones específicas, donde la elección del paradigma adecuado a utilizar depende de los datos y la naturaleza del problema.

Existen tres paradigmas de aprendizaje en *Machine Learning*: a) Supervisado: entrena la red proporcionando datos de entrada $x$ y datos de salida $t$ que se compone de etiquetas u objetivos de manera que el algoritmo aprende a generar una salida $y$, que es similar a $t$, dada una entrada $x$. b) No supervisado: entrena proporcionando datos de la entrada $x$, definiendo solo la cantidad de características de salida, de esta manera se busca que la red pueda sintetizar estas características a partir los datos $x$, tales como; patrones, secuencias y relaciones, identificando patrones y estructuras sin una guía previa. c) Reforzado: consiste de una red subordinada cuya salida es evaluada por otra red la cual proporciona una señal correctiva (de recompensa o castigo) en función de mejorar el desempeño de la red subordinada[7]-[12].

Existen diversas formas de evaluar el desempeño de una ANN, las cuales permiten analizar tanto el proceso de entrenamiento como la calidad de las predicciones. Durante el entrenamiento, se evalúan métricas como el gradiente, que indica los ajustes realizados a los pesos, la tasa de aprendizaje, que refleja la velocidad de convergencia, y el error cuadrático medio (MSE), que mide la precisión del modelo al minimizar la diferencia entre las salidas predichas y los valores reales. Para analizar la calidad de las predicciones, se utilizan herramientas como la matriz de confusión, que resume las predicciones correctas e incorrectas por clase, y la curva ROC, que mide la capacidad del modelo para distinguir entre las distintas categorías. Adicionalmente, los histogramas de errores permiten visualizar la distribución de las desviaciones entre las predicciones y los valores reales, identificando posibles fallos en el ajuste del modelo. Estas herramientas en conjunto proporcionan una visión integral del desempeño de la red, permitiendo identificar áreas de mejora.

### 3.4.3. ANN como estrategia de control

En los últimos años, las redes neuronales han surgido como una herramienta altamente eficaz para el diseño de controladores en sistemas complejos, particularmente en aquellos con comportamiento no lineal y alta incertidumbre. A diferencia de enfoques tradicionales como los métodos PI o MPC, las ANN pueden aprender dinámicamente del comportamiento del sistema, lo que les permite adaptarse rápidamente a cambios en las condiciones operativas o a perturbaciones.

En el control de convertidores de potencia, las ANN han demostrado ser especialmente útiles. Convertidores como el FCC o los de tipo multinivel presentan dinámicas altamente no lineales y complejas de modelar con técnicas tradicionales. Las redes neuronales pueden ser entrenadas para aprender estas dinámicas no lineales y proporcionar señales de control precisas. Por ejemplo, en lugar de utilizar un controlador PI que requiere ajustes manuales de parámetros, una ANN puede aprender la relación entre las variables del sistema, como las tensiones de los capacitores o las corrientes de fase, y generar las señales de control óptimas para minimizar el error o maximizar la eficiencia energética.

Además, al combinar redes neuronales con métodos como el MPC, es posible diseñar esquemas de control híbridos donde la red neuronal se utiliza para predecir el comportamiento del convertidor, mientras que el MPC ajusta las decisiones de control para optimizar la respuesta dinámica del sistema. Los enfoques duales permiten mejorar la robustez y la capacidad de respuesta, especialmente en aplicaciones de alta potencia y alta frecuencia.

A pesar de los desafíos en su implementación, como la necesidad de hardware especializado para ejecutar redes neuronales en tiempo real, su capacidad de aprender dinámicas complejas y adaptarse a condiciones cambiantes hace que las redes neuronales sean una solución prometedora para el control avanzado de convertidores de potencia.

# Capítulo 4

# Estrategia de control propuesta: Dual-ANN

En este capítulo se describen las dos propuestas de estrategias de control Dual-ANN, que integran el uso de redes neuronales en dos configuraciones distintas para optimizar el rendimiento de un sistema de control híbrido, que imita el comportamiento del control lineal y MPC según corresponda. Las estrategias constan de:

1. La unión de dos ANN independientes, una basada en el control lineal y otra basada en el FCS-MPC, unidas mediante un control de bandas por histéresis.
2. Una única ANN Dual, que contempla el comportamiento del control PI, FCS-MPC e histéresis.

## 4.1. Unión de ANN con histéresis

La unión de ANN con histéresis, es una extensión del trabajo de memoria [7] realizado por Cristóbal Cortés, en el cual se desarrollaron redes neuronales específicas para ejecutar las funciones de control lineal y FCS-MPC independientemente. En la sección 4.1.1 y 4.1.2 se expone un resumen de la estructura y las características de las ANN creadas para aquel trabajo de memoria; redes que luego son utilizadas en este trabajo para agregar el enfoque de la histéresis.

El mecanismo de histéresis expuesto en la sección 4.1.3 permite la transición fluida entre los modos de control. La histéresis actúa como un mediador entre los sistemas de control, activando cada ANN en función de las condiciones del sistema y reduciendo cambios abruptos entre los modos.

### 4.1.1. ANN control PI

La estructura de la red neuronal para la estrategia de control lineal es una red de tipo *backpropagation network* compuesta de una capa oculta y una capa de salida. Siendo las entradas de la ANN: las corrientes de referencia $i_x^*$, las corrientes medidas $i_x$, el error entre ambas $(i_x^* - i_x)$ y el índice de modulación del tiempo anterior $m_x^{k-1}$. En cuanto a las

salidas, estas se definieron como los índices de modulación $m_x^k$. En la figura 4.1 se muestra un esquema de su estructura.

La red neuronal busca establecer una relación entre las corrientes y el índice de modulación correspondiente, desempeñándose como una ANN de regresión.

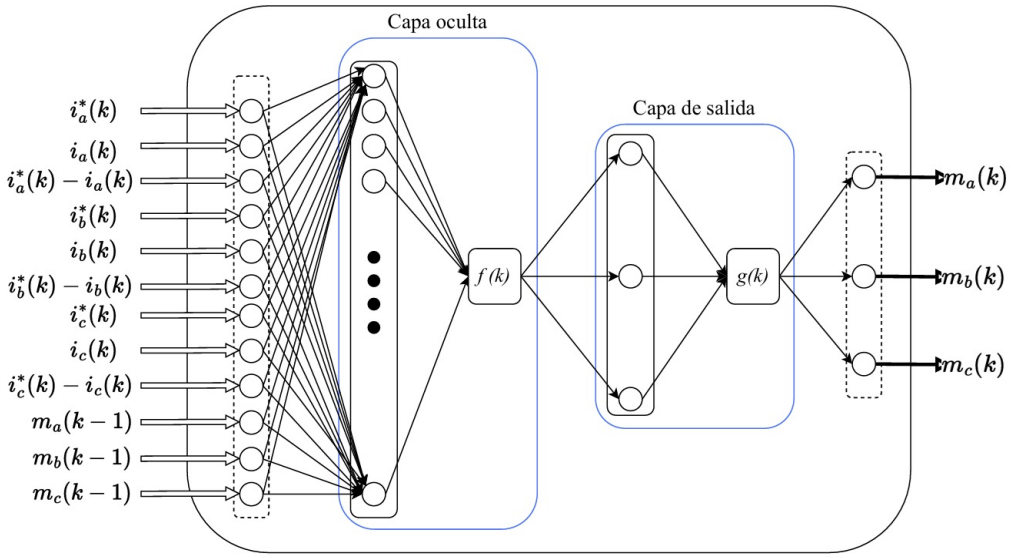



Figura 4.1: Diagrama de la ANN para el control lineal[7].

El esquema de control en base a redes neuronales aplicada a la estrategia de control lineal se muestra en la figura 4.2.

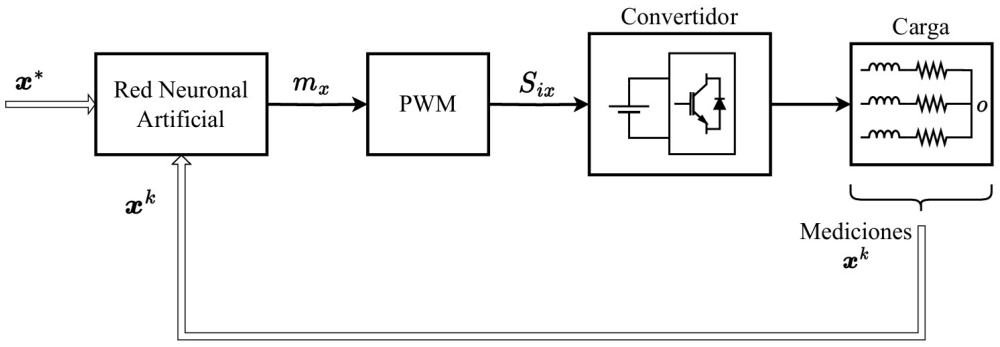
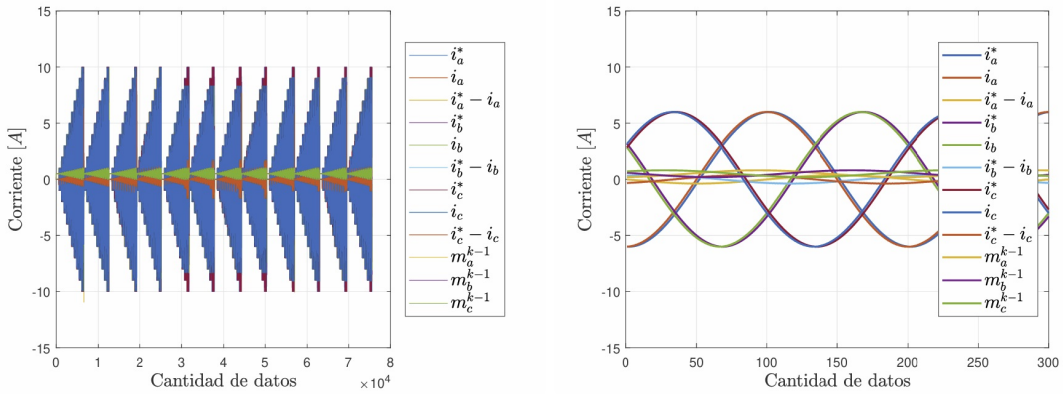



Figura 4.2: Esquema de control en base a ANN aplicada a la estrategia de control lineal[7].

Los datos de entrenamiento para la red neuronal consisten en las entradas y salidas de la ANN, que son datos extraídos de la operación de un FCC con control PI y modulación PS-PWM alimentando una carga $RL$ con inductor de $10[mH]$ y resistencia de $15[\Omega]$. Siendo la frecuencia de muestreo del controlador PI de $10[kHz]$ y la del modulador PWM de $1{,}666[kHz]$. El set de datos utilizado para la entrada es el mostrado en la figura 4.3 , donde se varió la magnitud de la corriente. Para la salida se tienen los índices de modulación de la figura 4.4.

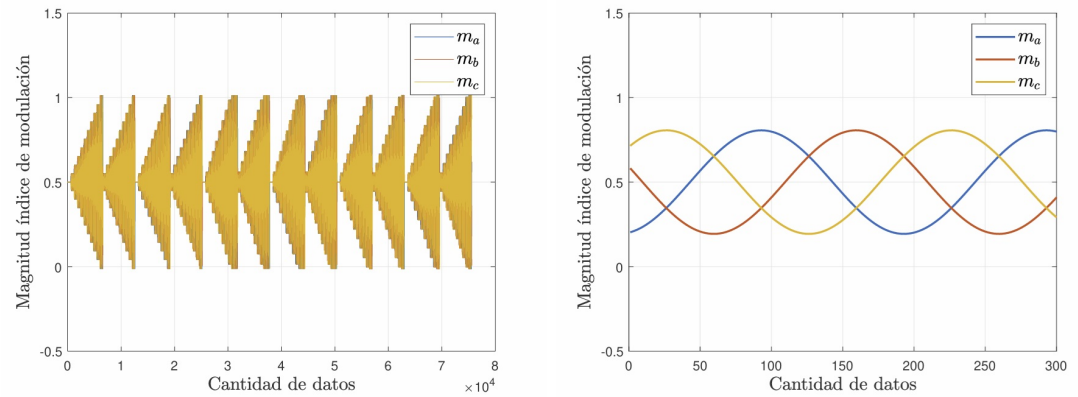
La red neuronal fue entrenada usando la *Deep Learning Toolbox*, una caja de herramientas de software de *MATLAB*, mientras que los datos de entrenamiento se obtuvieron

de la simulación del control lineal, realizada en el software *PLECS*. En la tabla 4.1 se muestra el detalle de los parámetros de entrenamiento de la red.




(a) Conjunto completo de datos (b) Detalle de los datos de entrada

Figura 4.3: Datos de entrada para el entrenamiento de la ANN lineal[7].




(a) Conjunto completo de datos de salida (b) Detalle de los datos

Figura 4.4: Datos de salida para el entrenamiento de la ANN lineal[7].

Es relevante destacar que una de las conclusiones del trabajo de memoria [7] señala que la ANN lineal desarrollada no logró capturar completamente la acción integral característica del controlador PI. Además, presenta la limitación de generar corrientes con amplitudes ligeramente superiores a las de referencia establecidas. No obstante, a pesar de estas restricciones, el uso de estas ANN representan una gran oportunidad para explorar estrategias avanzadas de control con histéresis.

Tabla 4.1: Parámetros de entrenamiento de la ANN Lineal.

| **Parámetro** | **Especificación** |
|---|---|
| Nodos en capas de: entrada/oculta/salida | 12 / 12 / 3 |
| Función de activación en capa oculta | Tanh |
| Función de activación en capa de salida | Lineal |
| Rango de las corrientes | -15, 15 [A] |
| Rango del error de corrientes | -2, 2 |
| Rango índices de modulación | 0, 1 |
| Algoritmo de entrenamiento | Gradient descent with adaptive learning rate backpropagation |
| Función de desempeño | Error cuadrático medio (MSE) |
| Número de épocas | 200,000 |
| Error de desempeño | 0.00001 |
| Proporción de datos de entrenamiento | 95 % |
| Proporción de datos de validación | 0 % |
| Proporción de datos de prueba | 5 % |

### 4.1.2. ANN control MPC

Su estructura es una red del tipo *backpropagation network* compuesta de una capa oculta y una capa de salida. La estructura de la red se muestra en la figura 4.5, siendo en las entradas: $i_x$ las corrientes medidas, $i_x^*$ las corrientes de referencia, $v_{1x}$ y $v_{2x}$ las tensiones de los capacitores medidas, $v_{c1}^*$ y $v_{c2}^*$ las tensiones de referencia de los capacitores, y $v_{xN}$ la tensión de salida del convertidor por fase con respecto al neutro del enlace de tensión continua.

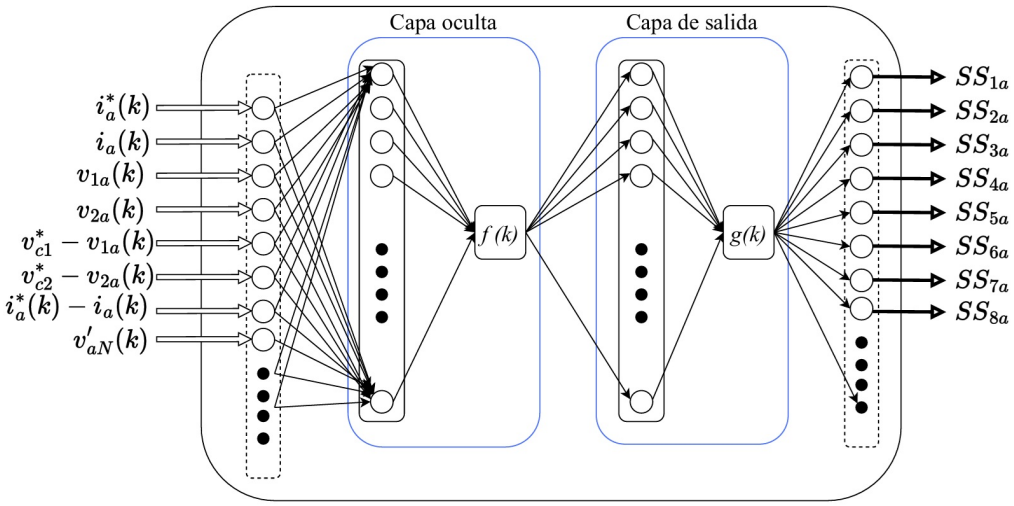



Figura 4.5: Diagrama de la ANN para el control FCS-MPC[7].

En cuanto a la salida, se realizó una parametrización de los nueve estados de conducción $S_{1a}$, $S_{2a}$, $S_{3a}$, $S_{1b}$, $S_{2b}$, $S_{3b}$, $S_{1c}$, $S_{2c}$, $S_{3c}$ a 24 estados de conducción $SS_{ix}$ los cuales corresponden a las 8 combinaciones posibles de conducción por fase de $S_{1x}$, $S_{2x}$ y $S_{3x}$ según lo mostrado en la tabla 4.2. Notar que así las variables $SS_{ix}$ son variables binarias

que representan los estados de conducción para las fases.

Tabla 4.2: Parametrización de los estados de conducción por fase.

| $S_{1x}$ | $S_{2x}$ | $S_{3x}$ | |
|---|---|---|---|
| 0 | 0 | 0 | $SS_{1x}$ |
| 0 | 0 | 1 | $SS_{2x}$ |
| 0 | 1 | 0 | $SS_{3x}$ |
| 0 | 1 | 1 | $SS_{4x}$ |
| 1 | 0 | 0 | $SS_{5x}$ |
| 1 | 0 | 1 | $SS_{6x}$ |
| 1 | 1 | 0 | $SS_{7x}$ |
| 1 | 1 | 1 | $SS_{8x}$ |

La red neuronal requiere encontrar una relación entre las corrientes y tensiones de referencia y medidas, con el respectivo estado de conducción parametrizado, por ende, la red se desempeña como una ANN de *clasificación* multi-clase multi-etiqueta, es decir, clasifica entre ocho clases y etiqueta las tres fases a la vez.

El esquema de control en base a redes neuronales aplicada a la estrategia de control FCS-MPC se muestra en la figura 4.6.

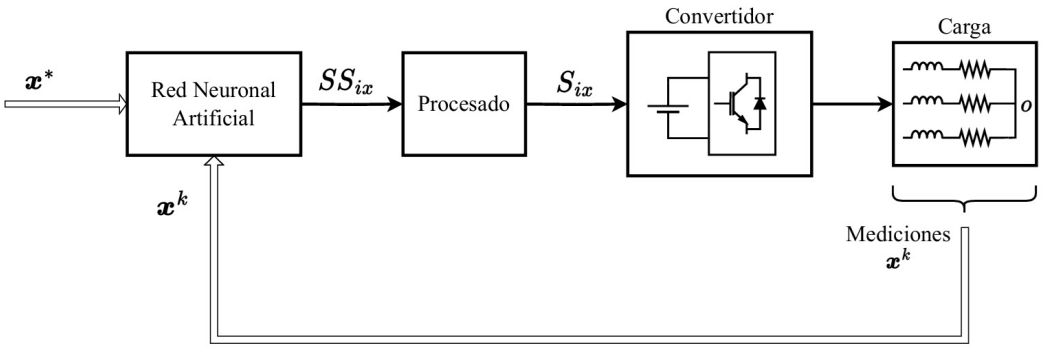
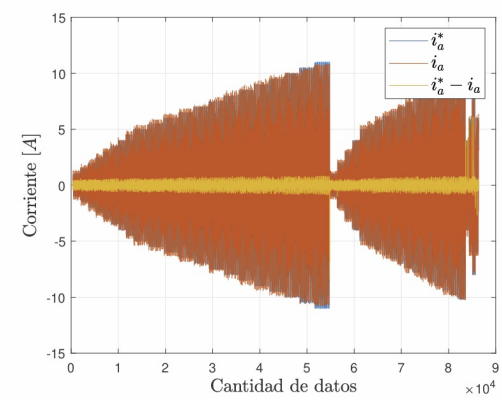



Figura 4.6: Esquema de control en base a ANN aplicada a la estrategia de control FCS-MPC[7].

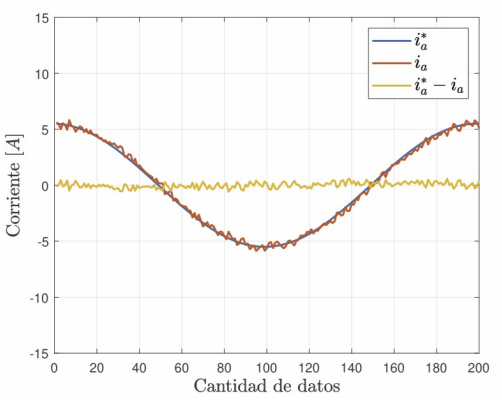
Los datos de entrenamiento para la red neuronal consisten en las entradas y salidas de la ANN, que son datos extraídos de la operación de un FCC con control FCS-MPC alimentando una carga $RL$ con inductor de $10[mH]$ y resistencia de $15[\Omega]$. Siendo la frecuencia de muestreo del control de $10[kHz]$.

El set de datos utilizado para la entrada es el mostrado en la figura 4.7, donde se varió la magnitud de la corriente, y al final para la tensión de los capacitores. Para la salida se tienen los switches de la figura 4.8. Por simplicidad, se muestran los datos para la fase A.

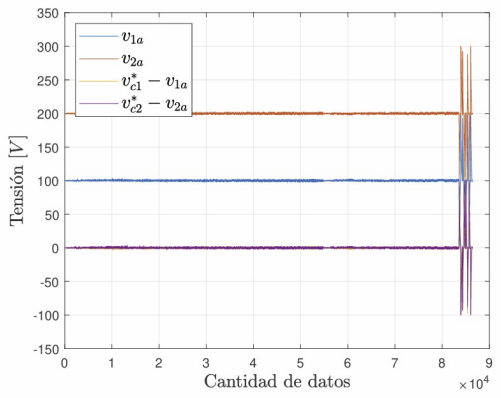
En la tabla 4.3 se muestra el detalle de los parámetros de entrenamiento de la red.



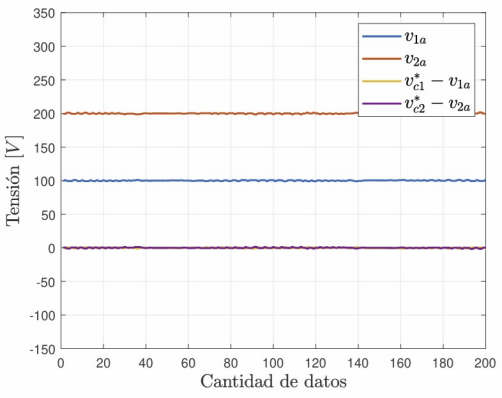

(a) Corrientes fase $a$




(b) Detalle corrientes fase $a$



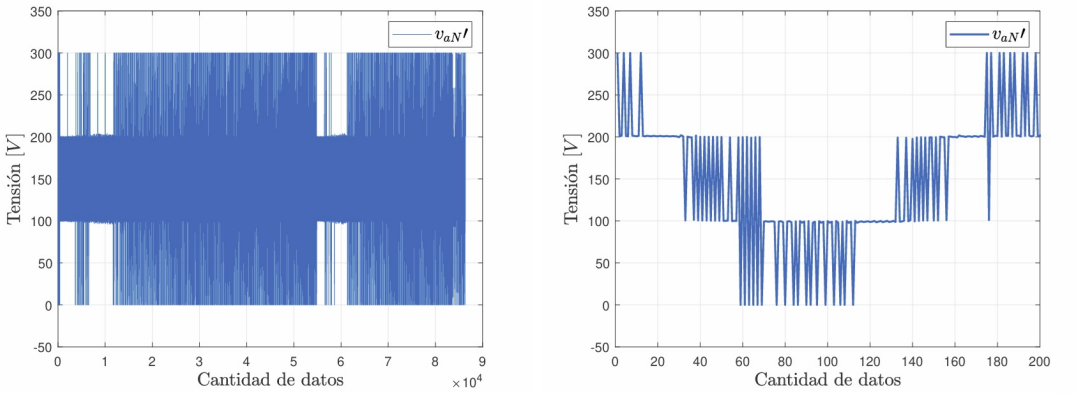

(c) Tensiones fase $a$




(d) Detalle tensiones fase $a$

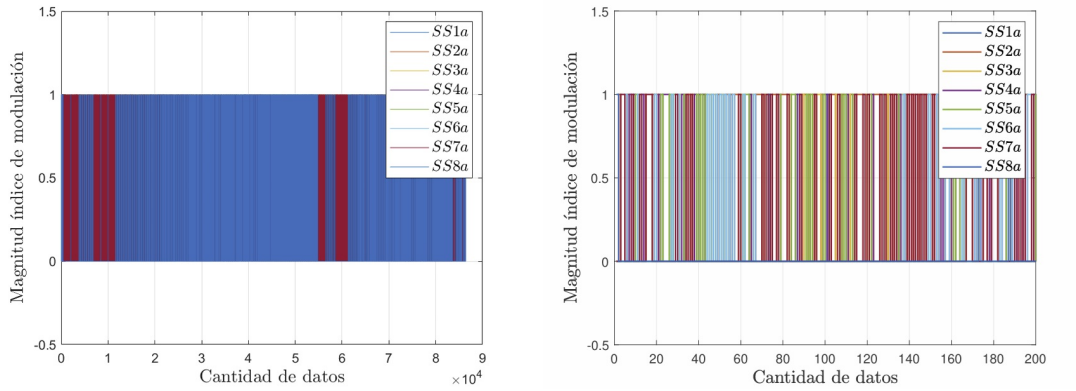
Tabla 4.3: Parámetros de entrenamiento de la ANN FCS-MPC.

| **Parámetro** | **Especificación** |
|---|---|
| Nodos en capas de: entrada/oculta/salida | 24 / 36 / 24 |
| Función de activación en capa oculta | Tanh |
| Función de activación en capa de salida | Softmax |
| Rango de corrientes | -15, 15 [A] |
| Rango de tensiones | 0, 300 [V] |
| Rango del error de corrientes | -2, 2 |
| Rango del error de tensiones | -100, 100 |
| Algoritmo de entrenamiento | Gradient descent with adaptive learning rate backpropagation |
| Función de desempeño | Error cuadrático medio (MSE) |
| Número de épocas | 1,000,000 |
| Error de desempeño | 0.001 |
| Gradiente mínimo del error | 1e-15 |
| Proporción de datos de entrenamiento | 90 % |
| Proporción de datos de validación | 0 % |
| Proporción de datos de prueba | 10 % |




(e) Tensión $v'_{aN}$ (f) Detalle tensión $v'_{aN}$

Figura 4.7: Datos de entrada para el entrenamiento de la ANN FCS-MPC[7].




(a) Datos de salida de la fase $a$ (b) Detalle de los datos de la salida de la fase $a$

Figura 4.8: Datos de salida para el entrenamiento de la ANN FCS-MPC[7].

### 4.1.3. Implementación de las bandas de histéresis

Para realizar esta implementación, se utilizó el entorno de simulación *PLECS*, en el cual se integraron las dos redes neuronales expuestas en 4.1.1 y 4.1.2. Ambas redes previamente entrenadas fueron exportadas de *MATLAB* con sus matrices de pesos y biases, para así incluirlas luego en el entorno de simulación. Los resultados de la simulación serán expuestos en el capítulo 5.1.

Considerando la teoría expuesta en el capítulo 3.3, para la implementación de las bandas de histéresis en este sistema de control dual, las bandas inferior y superior se definieron como $J_H = 2800$ y $J_L = 26$, estos valores se establecieron al probar valores de forma iterativa, hasta lograr el control deseado: activando la ANN de control lineal en estado estacionario y la ANN de control FCS-MPC en estado transitorio, transicionando suavemente entre un control y otro. Por otro lado, en línea con la ecuación 3.18, se especificaron los factores de peso para la corriente de carga y la tensión de los capacitores en $\gamma_i = 30$ $\gamma_v = 1$.

Adicionalmente, para reducir las oscilaciones de corriente en ciertas fases cuando el control cambia de FCS-MPC a PI, se implementa un código adicional. Este código facilita

la transición de manera que ocurra cuando la corriente en cada fase se aproxime a valores cercanos a cero de forma independiente. De este modo, se extiende el control FCS-MPC para asegurar que el cambio se produzca en un momento óptimo posterior. Este método tiene como objetivo suavizar la transición y minimizar las perturbaciones, manteniendo así la estabilidad y el rendimiento del sistema durante el cambio.

Este tipo de propuesta representa un avance significativo en la optimización del control de convertidores de capacitor flotante, ya que al emplear redes neuronales para determinados controles, se logra reducir el tiempo de cálculo, principalmente porque la ANN recorre su matriz de pesos y sesgos una vez independientemente del número de niveles del convertidor. Comparando con controles más complejos como el FCS-MPC, el uso de una ANN para controlar el convertidor viene a desligar el cálculo iterativo de los estados de conmutación mejorando así la eficiencia y carga computacional del sistema. La unión de estas dos redes neuronales permite así, en teoría, una implementación ágil y escalable, especialmente útil en aplicaciones de alta exigencia donde se requiere una rápida respuesta y estabilidad en el sistema.

## 4.2. ANN Dual

En línea con el mismo objetivo buscado en 4.1, se crea una ANN conjunta incluyendo en su entrenamiento el comportamiento del control PI, control FCS-MPC e histéresis. Con la motivación de posteriormente analizar qué control dual responde de mejor manera.

De manera iterativa se fueron entrenando redes hasta encontrar una óptima basándose en el conocimiento previo del trabajo [7] y [15]. Finalmente la ANN Dual expuesta en este trabajo se realizó utilizando las mismas herramientas computacionales y mismos parámetros del FCC para el set de datos de entrenamiento; y tomando como base o punto de partida los parámetros de la ANN lineal expuesta en 4.1.1.

### 4.2.1. Estructura y parámetros

La estructura de la red neuronal dual es una red de tipo *backpropagation network* compuesta de una capa oculta y una capa de salida. Siendo las entradas: las corrientes de referencia $i_x^*$, las corrientes medidas $i_x$, el error entre ambas $(i_x^* - i_x)$ y el índice de modulación del tiempo anterior $m_x^{k-1}$. En cuanto a las salidas, éstas se definieron como los índices de modulación $m_x^k$. En la figura 4.9 se muestra un esquema de su estructura. El esquema de control aplicado en esta estrategia corresponde al mostrado en la figura 4.2, que es el mismo utilizado para la ANN lineal.

La red neuronal busca establecer una relación entre las corrientes y el índice de modulación correspondiente, desempeñándose como una ANN de regresión.

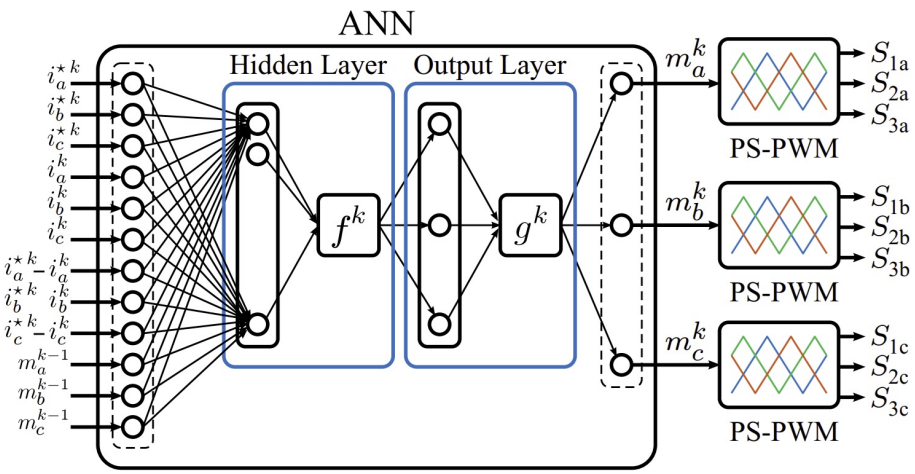



Figura 4.9: Esquema de la ANN propuesta.

### 4.2.2. Set de datos

Los vectores de entrada y salida fueron obtenidos a partir de la simulación realizada en *PLECS* del FCC gobernado por el control dual con histéresis, explicado en 3.3.

El detalle de los parámetros del sistema se muestran en la tabla 4.4, mientras que en las figuras 4.10 y 4.11 se exponen los vectores de entrada y salida utilizados para entrenar la ANN, con un zoom para ver el comportamiento de las variables a detalle.

Tabla 4.4: Parámetros del sistema.

| Símbolo | Parámetro | Valor |
|---|---|---|
| $h$ | Paso de muestreo | 0.0001 |
| $f_s$ | Frecuencia de muestreo | 50 Hz |
| $f_t$ | Frecuencia triangular | 10/6 kHz |
| $V_{dc}$ | Tensión de enlace de CC | 300 V |
| $C_{1x}$ | Valor del capacitor 1 | 330 $\mu$F |
| $C_{2x}$ | Valor del capacitor 2 | 330 $\mu$F |
| $L_l$ | Inductancia de carga | 10 mH |
| $R_l$ | Resistencia de carga | 15 $\Omega$ |

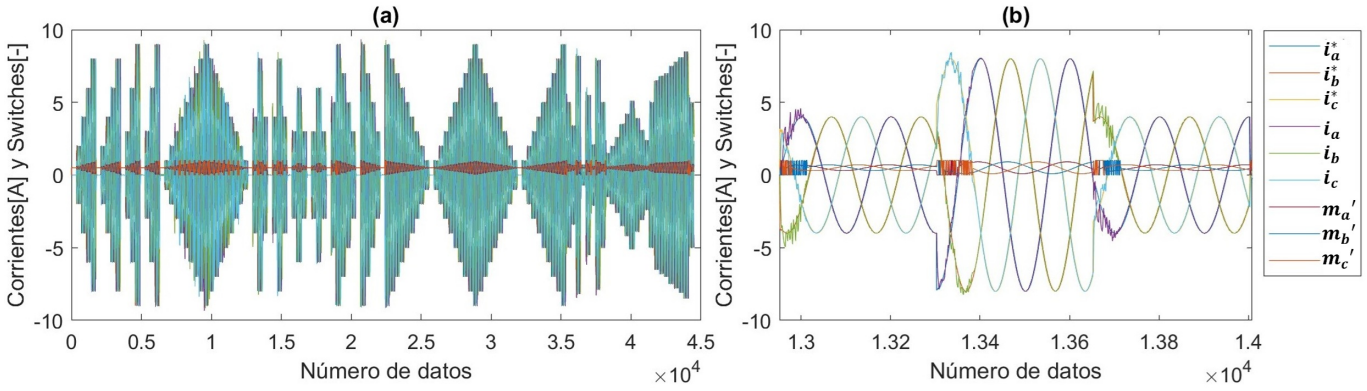



Figura 4.10: Datos de entrada del entrenamiento de la ANN Dual: (a) Set de datos completo. (b) Zoom de las variables.

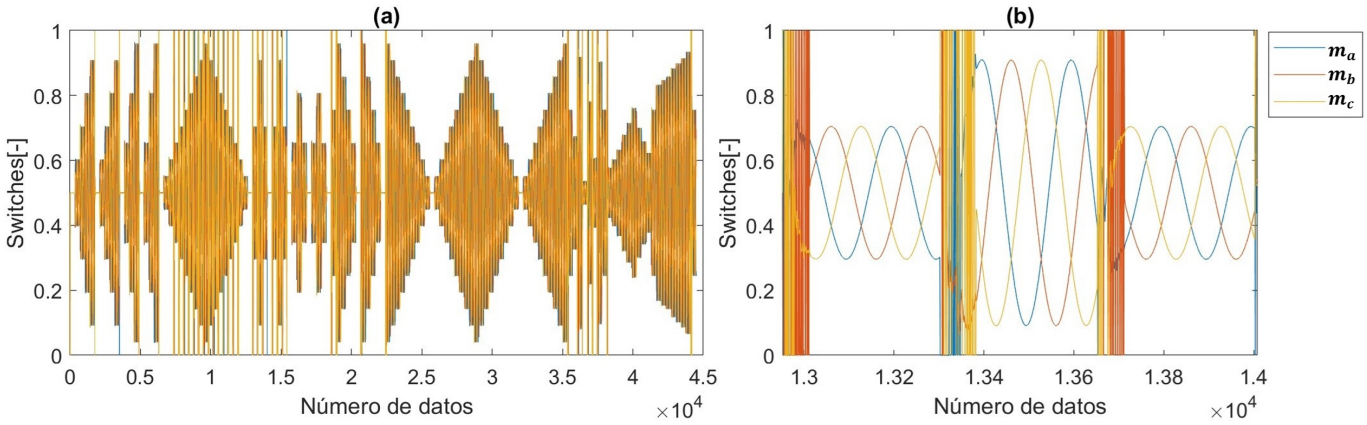



Figura 4.11: Datos de salida del entrenamiento de la ANN Dual: (a) Set de datos completo. (b) Zoom de las variables.

### 4.2.3. Entrenamiento de la ANN

Los parámetros modificados con respecto a la ANN lineal para entrenar la red fueron: el número de capas ocultas, el número de épocas y el conjunto de datos de entrenamiento. Este último resultó ser el factor más determinante en cuanto a lograr un control efectivo con la red, ya que no fue suficiente un set de datos en el que se incrementara progresivamente la amplitud de corriente; fue necesario variar la amplitud de diversas maneras para que la red pudiera aprender y replicar el comportamiento del control dual PI-MPC con histéresis.

En la tabla 4.5 se muestran los parámetros de entrenamiento para la ANN Dual. Cabe destacar que este fue un proceso iterativo de prueba y error, donde se probó modificando de distintas maneras los parámetros de la tabla, y que fue de suma importancia el tener un set de datos óptimo ya que era fundamental que la red obtuviera suficientes datos del fenómeno transitorio, para que éste no sea ignorado en el aprendizaje. El entrenamiento se detiene una vez se alcanza el número de épocas o el error objetivo expuesto.

Notar que en *MATLAB* existen funciones que permiten realizar algoritmos de entrenamiento de forma generalizada. El entrenamiento fue realizado con la función *train*, que recibe como argumentos la ANN, creada con las herramientas del paquete *Deep Learning Toolbox*, el vector de entrada y el vector de salida.

Tabla 4.5: Parámetros de entrenamiento de la ANN Dual.

| **Parámetro** | **Especificación** |
|---|---|
| Nodos en capas de: entrada/oculta/salida | 12 / 14 / 3 |
| Función de activación en capa oculta | Tanh |
| Función de activación en capa de salida | Lineal |
| Rango de las corrientes | -15, 15 [A] |
| Rango del error de corrientes | -2, 2 |
| Rango índices de modulación | 0, 1 |
| Algoritmo de entrenamiento | Gradient descent with adaptive learning rate backpropagation |
| Función de desempeño | Error cuadrático medio (MSE) |
| Número de épocas | 450,000 |
| Error objetivo | 0.00001 |
| Proporción de datos de entrenamiento | 95 % |
| Proporción de datos de validación | 0 % |
| Proporción de datos de prueba | 5 % |

A continuación se muestra el código utilizado para entrenar la ANN Dual, que se divide en una etapa de formulación, donde se adaptaron los vectores de entrada y salida, y otra etapa de entrenamiento, en donde se ingresaron los parámetros de la tabla 4.5. Es importante mencionar que la red neuronal del tipo *fitnet* es aquella que su capa oculta tiene una función de activación *tangencial* y que su capa de salida tiene una función de activación *lineal*.

```
%%----------Formulación:----------
InputFeaturesSamples = cell(1, 1);
TargetsSamples = cell(1, 1);
load('data_dual.mat');
InputFeaturesSamples{1}=[input.iar input.iA input.iar-input.iA input.ibr
↪  input.iB input.ibr-input.iB input.icr input.iC input.icr-input.iC
↪  input.s1at input.s1bt input.s1ct];
TargetsSamples{1}=[output.s1A output.s1B output.s1C];
% From cell to matrix conversion, the proper form for NN training.
T = cell2mat(TargetsSamples);
X = cell2mat(InputFeaturesSamples);
% nnstart
T = transpose(T);
X = transpose(X);

%%----------Entrenamiento:----------
x = X;  % X - input data.
t = T;  % T - target data.
% Choose a Training Function
trainFcn = 'traingda';
% Create a Pattern Recognition Network
hiddenLayerSize = 14;
net = fitnet(hiddenLayerSize, trainFcn);
% Inicializacion de la red neuronal
net.layers{:}.initFcn = 'initwb';
```

```
net.inputWeights{1}.initFcn = 'rands';
net.layerWeights{2}.initFcn = 'rands';
net.biases{:}.initFcn = 'rands';
net = init(net);
net.inputs.range = {[-15, 15; -15, 15; -2,2; -15, 15; -15, 15; -2,2; -15, 15;
↪  -15, 15; -2,2; 0,1; 0,1; 0,1]};
net.outputs.range = {[],[0,1; 0,1; 0,1]};
% Setup Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 95/100;
net.divideParam.valRatio = 0/100;
net.divideParam.testRatio = 5/100;
net.trainParam.epochs=450000;
net.trainParam.goal=1e-6;
net.trainParam.min_grad = 1e-9;
net.performFcn = 'mse';
% Train the Network
[net,tr] = train(net,x,t);
% Test the Network
y = net(x);
e = gsubtract(t,y);
performance = perform(net,t,y);
tind = vec2ind(t);
yind = vec2ind(y);
percentErrors = sum(tind ~= yind)/numel(tind);
% View the Network
view(net)
% Plots
figure, plotperform(tr)
figure, plottrainstate(tr)
figure, ploterrhist(e)
figure, plotconfusion(t,y)
figure, plotroc(t,y)
```

El mejor desempeño del entrenamiento ocurrió en la época 245953, a continuación, en la tabla 4.6 y figuras 4.12 y 4.13 se muestran los resultados de entrenamiento.

Tabla 4.6: Resultados de entrenamiento.

| | ANN |
|---|---|
| Tiempo | 03:14 [h] |
| Performance | 0.0056242 |
| Gradiente | 0.0039207 |
| Validation Checks | 0 |

La matriz de confusión expone un buen desempeño general del modelo, especialmente en las clases 1 y 2, con una alta precisión en la clasificación de estas clases. La clase 1 tiene un recall del 96.9 % y una especificidad del 84.3 %, mientras que la clase 2 presenta un recall de 94.7 % y especificidad de 85.2 %. Esto indica que el modelo clasifica correctamente la mayoría de los ejemplos de estas clases y pocas predicciones son erróneas. Por otro lado, la clase 3 tiene un recall más bajo, de 78.7 %, sugiriendo que muchos ejemplos de esta clase se clasifican incorrectamente como otras, aunque mantiene una alta especificidad de

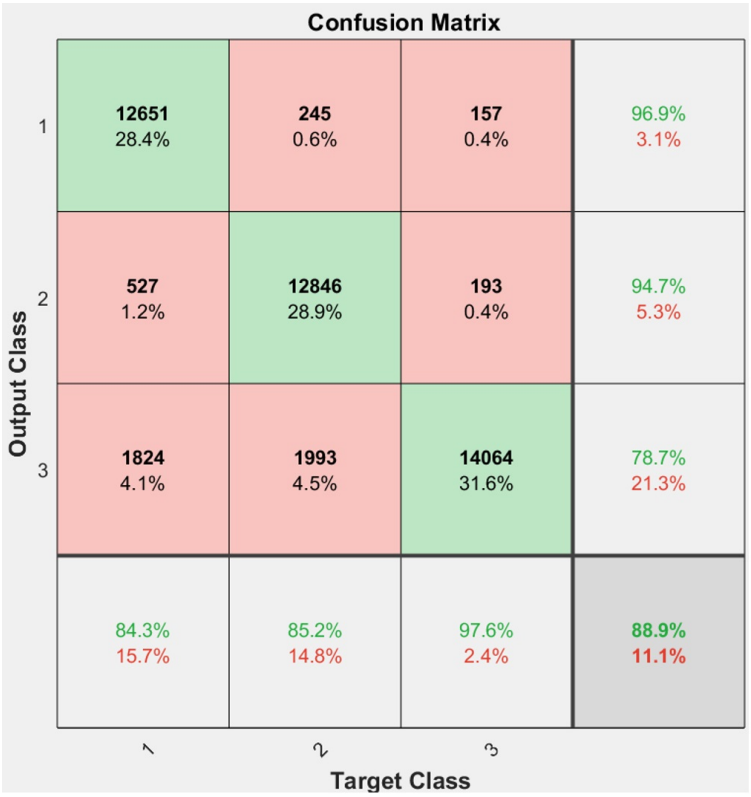



Figura 4.12: Matriz de confusión.

97.6 %, lo que indica que, cuando se clasifica como clase 3, suele ser correcto.

Las gráficas de rendimiento y la matriz de confusión muestran que la ANN para el control dual tiene un desempeño estable y preciso en el entrenamiento. El gradiente bajo indica que el modelo está convergiendo correctamente y que la ANN ha logrado reducir el error sin grandes variaciones. Validation checks en cero sugieren que el modelo no está sobreajustado y que generaliza bien en el conjunto de validación. La tasa de aprendizaje constante, cercana a 8, mantiene la estabilidad del entrenamiento. El histograma muestra que la mayoría de los valores están cerca de cero, lo que significa que las predicciones de la red son precisas y que el error promedio es bajo. La gráfica de MSE indica que el mejor resultado (0.0056242) se alcanzó en una época intermedia, lo que confirma que el modelo logró una buena precisión sin necesidad de ajustes adicionales al final.

La gráfica ROC alineada con la diagonal indica que el modelo no distingue bien entre clases, lo que normalmente reflejaría un mal desempeño. Sin embargo, en este caso, la explicación sugiere que esto se debe al uso de un control con salidas continuas, para las cuales el ROC no es una métrica adecuada. La pequeña curva inicial refleja la activación temporal del control MPC, que sí genera salidas discretas evaluables con el ROC. Por lo que la gráfica no indica un mal funcionamiento, sino una incompatibilidad metodológica al usar ROC para un sistema mayormente continuo.

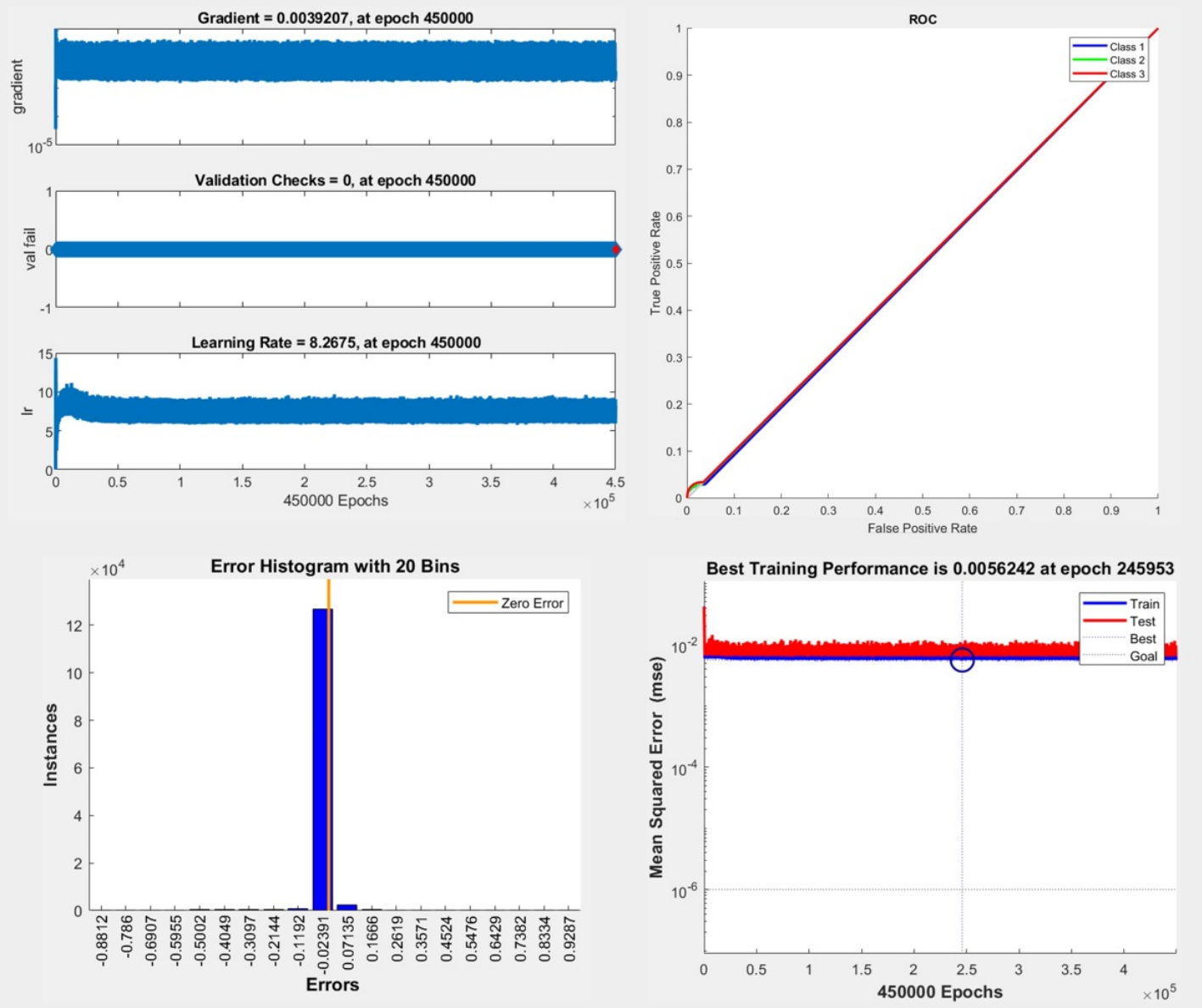



Figura 4.13: Gráficas de gradiente, histograma, RIC y MSE para la ANN Dual.

# Capítulo 5

# Análisis y comparativa de estrategias de control

## 5.1. Simulaciones

En esta sección se presentan los resultados obtenidos a partir de las simulaciones realizadas en *PLECS*. Se incluyen gráficos de las corrientes, el estado de conmutación de los switches (considerando únicamente $s1A$, $s1B$ y $s1C$ para simplificar la representación), y las tensiones, tanto las internas de los capacitores como la tensión de salida $v_{aN}$. Los casos analizados son los siguientes: (1) Control PI, (2) Control FCS-MPC, (3) Control PI+ FCS-MPC con histéresis, (4) Control basado en la unión de ANN con histéresis, y (5) Control con ANN Dual.

Se estudia el comportamiento de los controles al someterlos a un cambio abrupto de referencia de -3 [A] a 7 [A], con la intención de analizar su desempeño dinámico. Es importante mencionar que las corrientes fueron graficadas con un trazo punteado para facilitar la visualización de la componente fundamental. No obstante, en la figura 1 en el Apéndice A se presentan estas gráficas con líneas continuas, lo que permite observar con mayor claridad el ripple.

Por otro lado, para evaluar el desempeño en estado estacionario, se calcula el THD (del inglés: Total Harmonic Distortion) y se obtiene el espectro de frecuencias para la corriente $i_a$ y la tensión de salida $v_{aN}$; en los casos en que la corriente de referencia tiene -3 [A] y 7 [A] de amplitud.

### 5.1.1. Control PI

En la figura 5.1 se observa que, durante el transitorio, las corrientes (e índices de modulación) presentan oscilaciones iniciales con un leve sobreimpulso mientras se ajustan a la nueva referencia. La respuesta es suave y progresiva, con las curvas intentando seguir la referencia sin cambios abruptos.

Las tensiones internas permanecen constantes en sus valores esperados, mientras que la tensión de salida $v_{aN}$ adopta cuatro niveles definidos una vez que la corriente alcanza una amplitud de 7 [A], teniendo antes solo dos niveles.

Considerando el tiempo de asentamiento como el intervalo desde el momento en que ocurre el cambio de referencia hasta que las tres fases de corriente se estabilizan, se obtiene:

$$t_a = 3{,}2[ms] \tag{5.1}$$

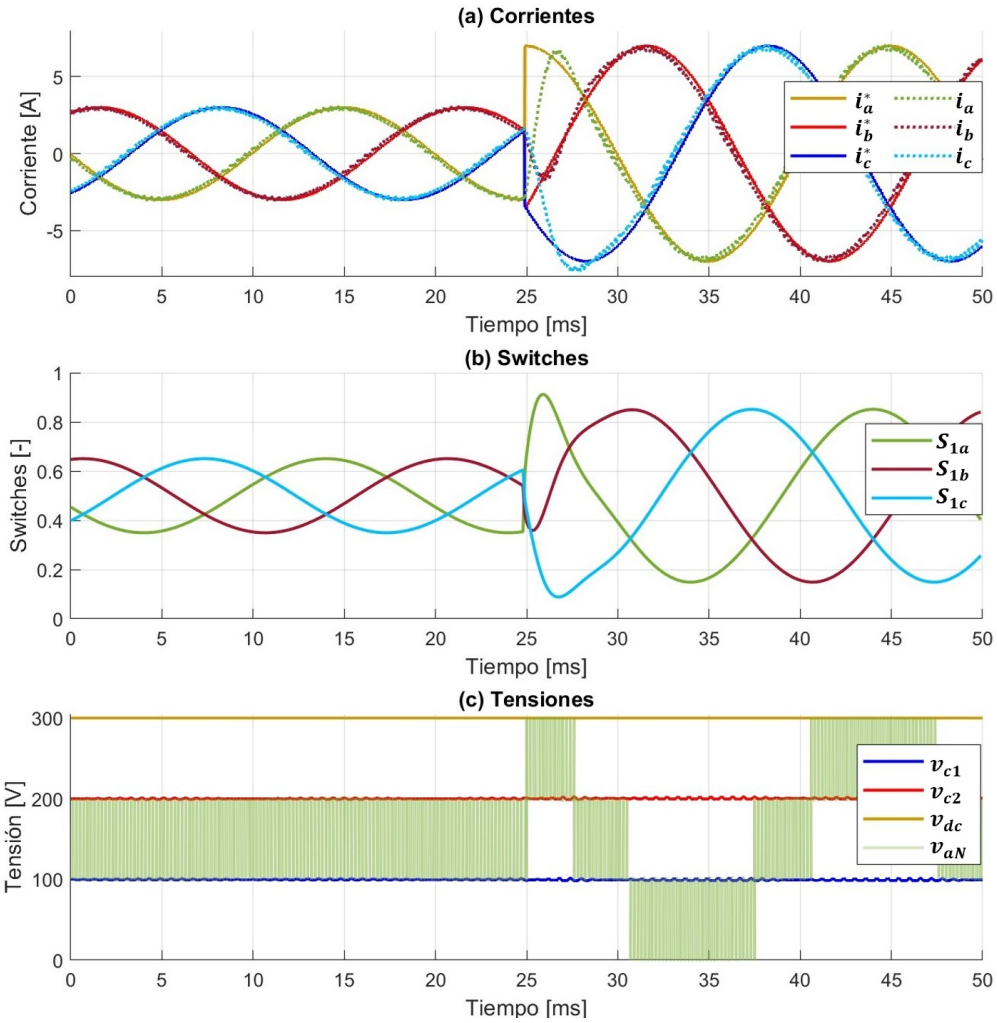



Figura 5.1: Corrientes, índices de modulación y tensiones para control PI en simulación.

En cuanto al estado estacionario, se analizan las curvas de la figura 5.2, donde se ve que para ambas amplitudes se sigue a la referencia de corriente de manera precisa y estable, demostrando un excelente desempeño en estado estacionario.

Se calculó el THD mostrado en la tabla 5.1 a partir de las mismas gráficas. Los valores de THD están dentro de los rangos óptimos.

Tabla 5.1: THD para diferentes valores de corriente para control PI.

| Amplitud | -3 [A] | 7 [A] |
|---|---|---|
| $i_a$ | 0.78 % | 0.77 % |
| $v_{aN}$ | 0.87 % | 0.80 % |

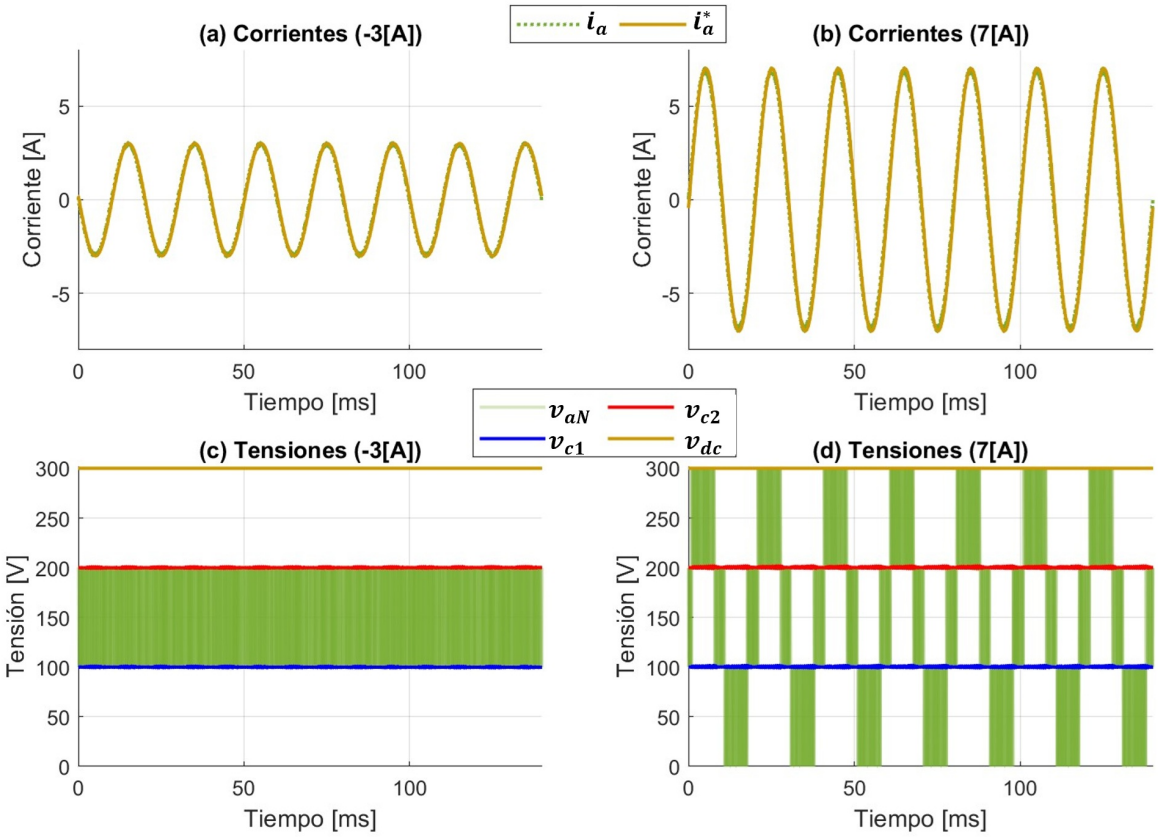



Figura 5.2: Corriente y tensiones de la fase A en estado estacionario con -3 [A] y 7 [A] para control PI en simulación.

Como era de esperar, los espectros de frecuencia de la figura 5.3 y 5.4 muestran armónicos en 50 [Hz] para la corriente y en 0 y 50 [Hz] para la tensión. También se presentan armónicos en torno a los 5000[Hz] y los 10000[Hz], sobre todo para las tensiones.

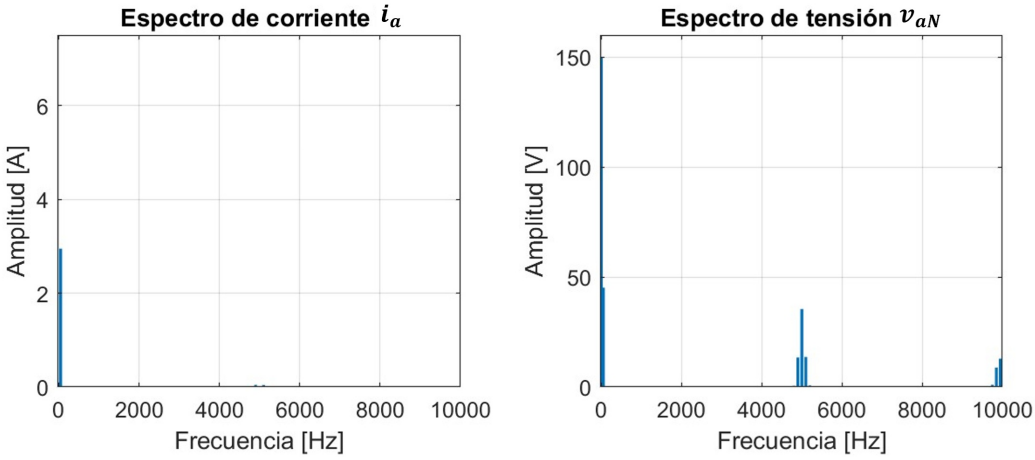



Figura 5.3: Espectro de corriente $ia$ y tensión de salida $v_{aN}$ con -3 [A] para control PI en simulación.

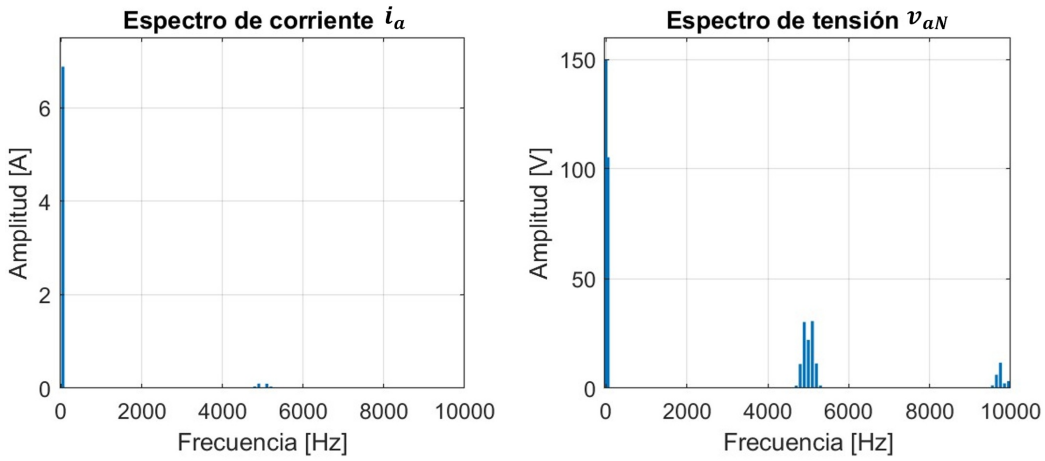



Figura 5.4: Espectro de corriente $ia$ y tensión de salida $v_{aN}$ con 7 [A] para control PI en simulación.

### 5.1.2. Control FCS-MPC

Analizando el transitorio, en la figura 5.5 se observa que a pesar de la característica ruidosa del control FCS-MPC, en todo momento las corrientes siguen efectivamente a la referencia. El tiempo de asentamiento mostrado a continuación resultó ser pequeño.

$$t_a = 0{,}9[ms] \tag{5.2}$$

Respecto a las tensiones, $v_{aN}$ no se ve tan definida debido a los efectos de la modulación del control, que realiza conmutaciones rápidas para mantener las variables de control dentro de los rangos deseados. Cabe destacar que el control FCS-MPC no solo controla las corrientes, sino también las tensiones de los capacitores, las cuales permanecieron constantes bajo este esquema de control.

En cuanto al estado estacionario, de la figura 5.6 (a) se obtiene que el control FCS-MPC logra seguir la referencia de corriente pero de manera ruidosa. A partir de estos datos se calculó el THD de la tabla 5.2, donde se refleja que el control FCS-MPC esta optimizado para priorizar la forma de onda de la corriente, lo que genera mayores distorsiones en la tensión.

Tabla 5.2: THD para diferentes valores de corriente para control FCS-MPC.

| Amplitud | -3 [A] | 7 [A] |
|---|---|---|
| $i_a$ | 6.60 % | 2.74 % |
| $v_{aN}$ | 49.32 % | 17.13 % |

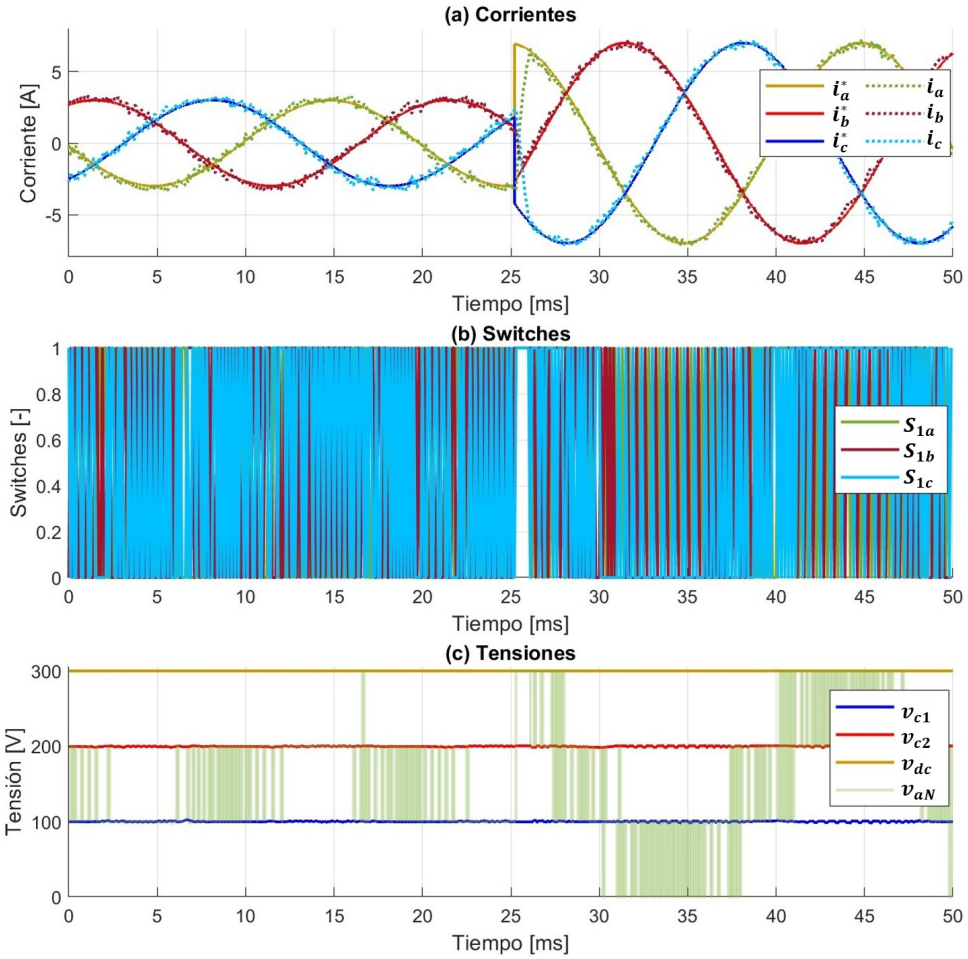



Figura 5.5: Corrientes, switches y tensiones para control FCS-MPC en simulación.

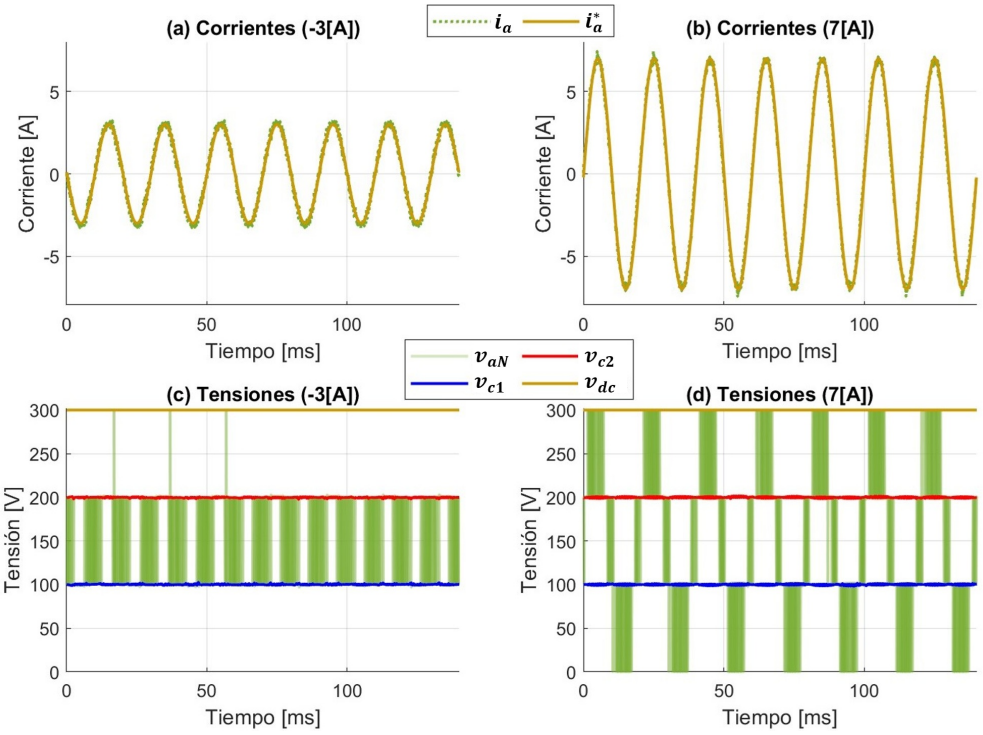



Figura 5.6: Corriente y tensiones de la fase A en estado estacionario con -3 [A] y 7 [A] para control FCS-MPC en simulación.

Los espectros de frecuencia de la figura 5.7 y 5.8 muestran armónicos significativos en 50 [Hz] para la corriente, y en 0 y 50 [Hz] para la tensión. También, hay numerosas armónicas de magnitudes pequeñas en las tensiones.

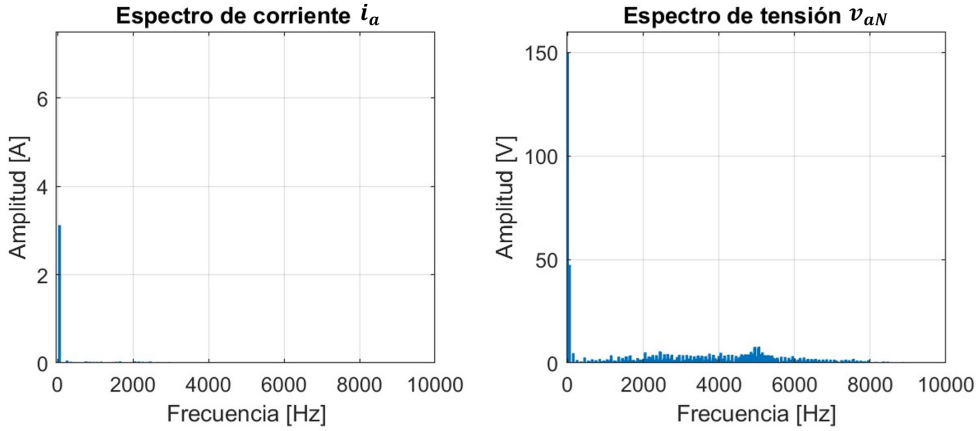



Figura 5.7: Espectro de corriente $i_a$ y tensión de salida $v_{aN}$ con -3 [A] para control FCS-MPC en simulación.

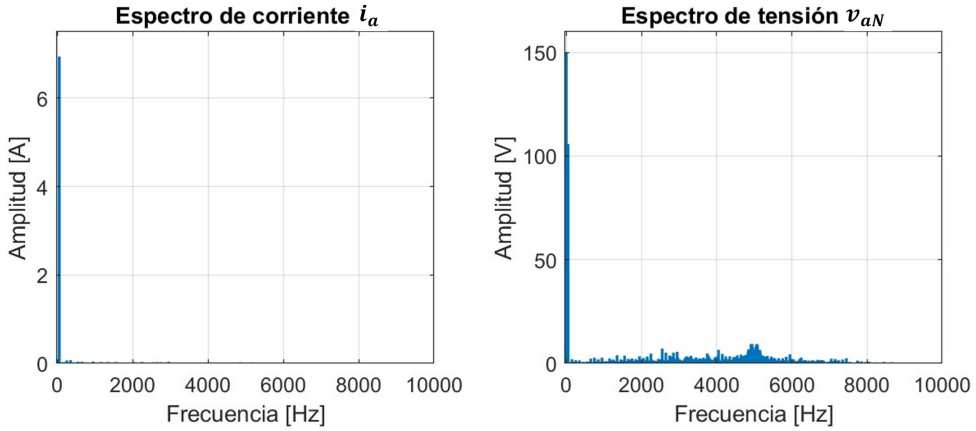



Figura 5.8: Espectro de corriente $i_a$ y tensión de salida $v_{aN}$ con 7 [A] para control FCS-MPC en simulación.

### 5.1.3. Histeresis: Control PI + FCS-MPC

En la Figura 5.9 se observa que el control sigue a la referencia, con el control FCS-MPC activándose durante el transitorio y el control PI manteniendo el régimen en estado estacionario. Este control dual no se encuentra optimizado, ya que se conservaron los parámetros utilizados en los controles del trabajo de memoria [7], con el fin de facilitar la comparación con los demás esquemas estudiados. Las oscilaciones observadas se deben a la alta velocidad de respuesta del control PI; al conmutar desde el control FCS-MPC al PI, este último requiere tiempo para actualizar sus valores internos, lo que genera dichas oscilaciones. El tiempo de asentamiento se expone a continuación:

$$t_a = 10{,}15[ms] \tag{5.3}$$

En cuanto a los switches, se puede observar que la activación del control PI desde el control FCS-MPC se realizó de manera selectiva por fase. Esto es para minimizar los sobre-impulsos indeseados, llevándola a cabo en los instantes en que las corrientes se encuentran cerca de valores nulos.

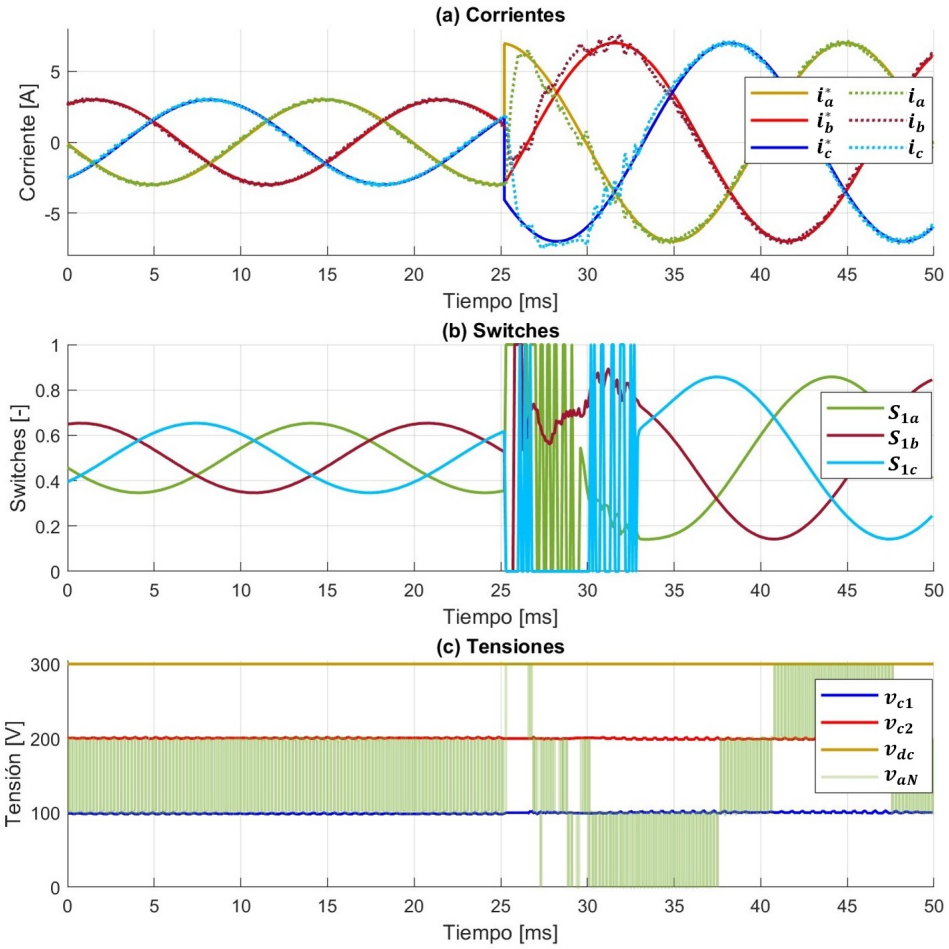



Figura 5.9: Corrientes, switches y tensiones para control con histeresis en simulación.

Respecto al estado estacionario, el comportamiento es el mismo que en 5.1.1, puesto que se activa el control PI.

### 5.1.4. Unión de ANN con histéresis

Para esta simulación se utilizan las ANN expuestas en los puntos 4.1.1 y 4.1.2, con las mismas bandas de histéresis del punto anterior, mencionadas en 4.1.3.

En la figura 5.10 se observa que este control no resulta del todo efectivo, pues hay problemas con el seguimiento de la referencia, especialmente con la ANN del control PI. Es importante mencionar que aunque se mantuvieron los valores de las bandas, se ve que el control FCS-MPC se activa menos tiempo que en el punto anterior.

A continuación se muestra el tiempo de asentamiento, que resultó ser mayor que en el control PI expuesto en 5.1, por lo que no se cumple el objetivo.

$$t_a = 5{,}7[ms] \tag{5.4}$$

En este control la tensión $v_{aN}$ mantiene cuatro niveles para la corriente de -3[A], a diferencia de los tres controles anteriores que mantuvo solo dos niveles.

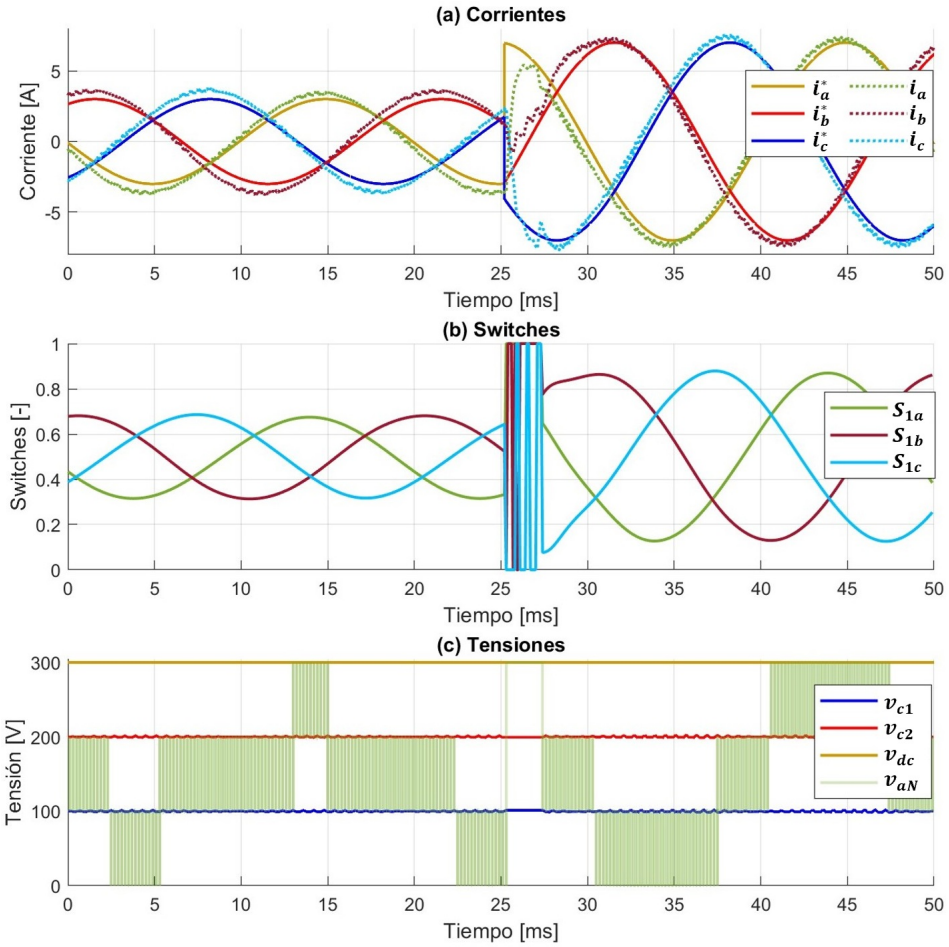



Figura 5.10: Corrientes, switches y tensiones para control con unión de ANN mediante histéresis en simulación.

En cuanto al estado estacionario, de la figura 5.11 se observa que el control no logra seguir la referencia. Esto ocurre porque, en estado estacionario, se activa la ANN control PI, que presenta una limitación en su entrenamiento y no es capaz de corregir el error, tal como se indica en [7]. El error máximo para $i_a$ en estado estacionario, para las referencias de -3 [A] y 7 [A], es de 0.84 [A] y 0.95 [A], respectivamente. En la tabla 5.3 se presentan los valores calculados de THD.

Tabla 5.3: THD para diferentes valores de corriente para control con unión de ANN mediante histéresis.

| Amplitud | -3 [A] | 7 [A] |
|---|---|---|
| $i_a$ | 1.24 % | 1.37 % |
| $v_{aN}$ | 19.14 % | 11.52 % |

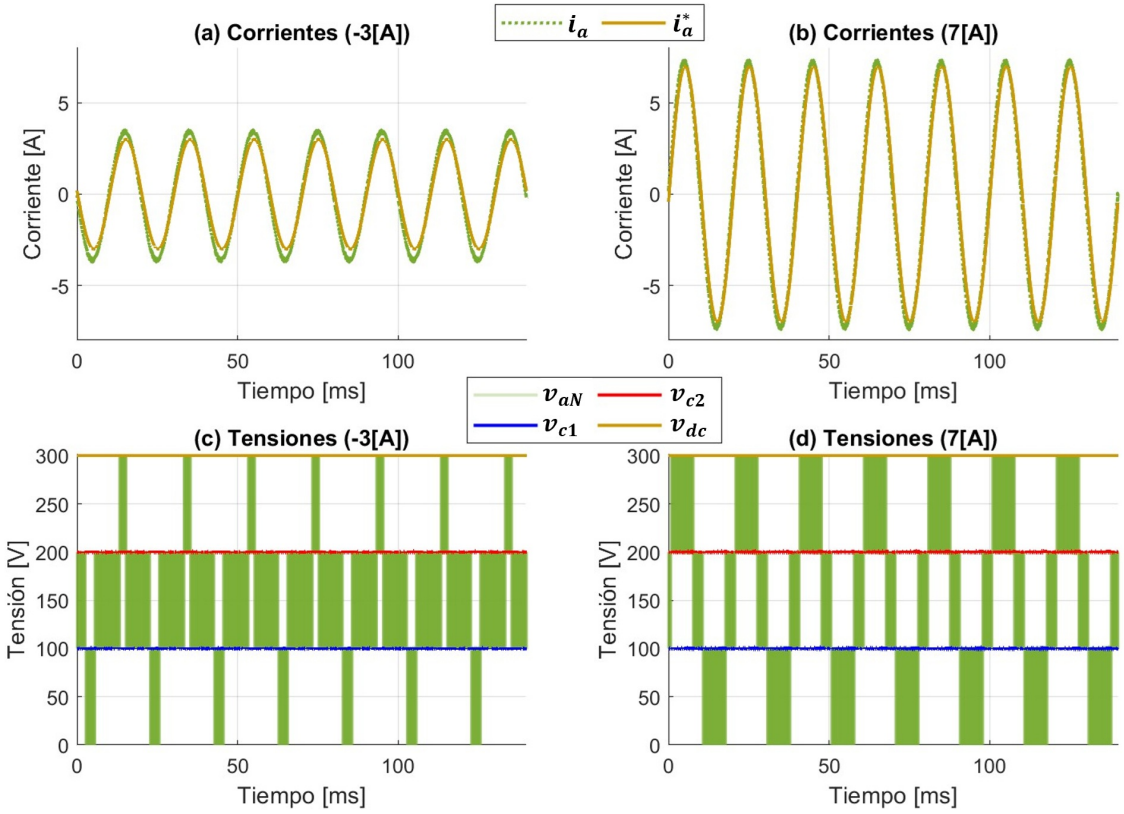



Figura 5.11: Corriente y tensiones de la fase A en estado estacionario con -3 [A] y 7 [A] para control con unión de ANN mediante histéresis en simulación.

Los espectros, al igual que los controles anteriores, muestran predominancia en la fundamental y en la de frecuencia cero para la tensión. Los armónicos de este control son equivalentes a los del control PI.

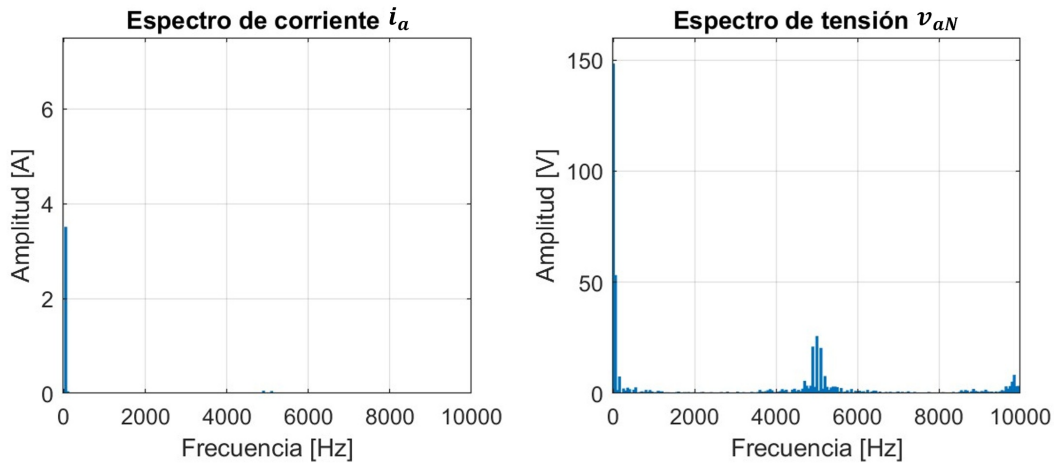



Figura 5.12: Espectro de corriente $i_a$ y tensión de salida $v_{aN}$ con -3 [A] para control con unión de ANN mediante histéresis en simulación.

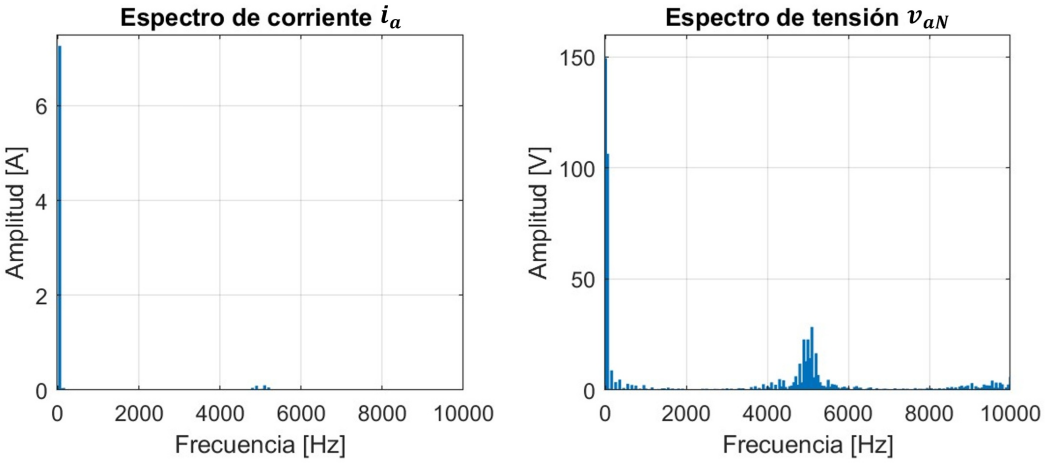



Figura 5.13: Espectro de corriente $i_a$ y tensión de salida $v_{aN}$ con 7 [A] para control con unión de ANN mediante histéresis en simulación.

### 5.1.5. ANN Dual

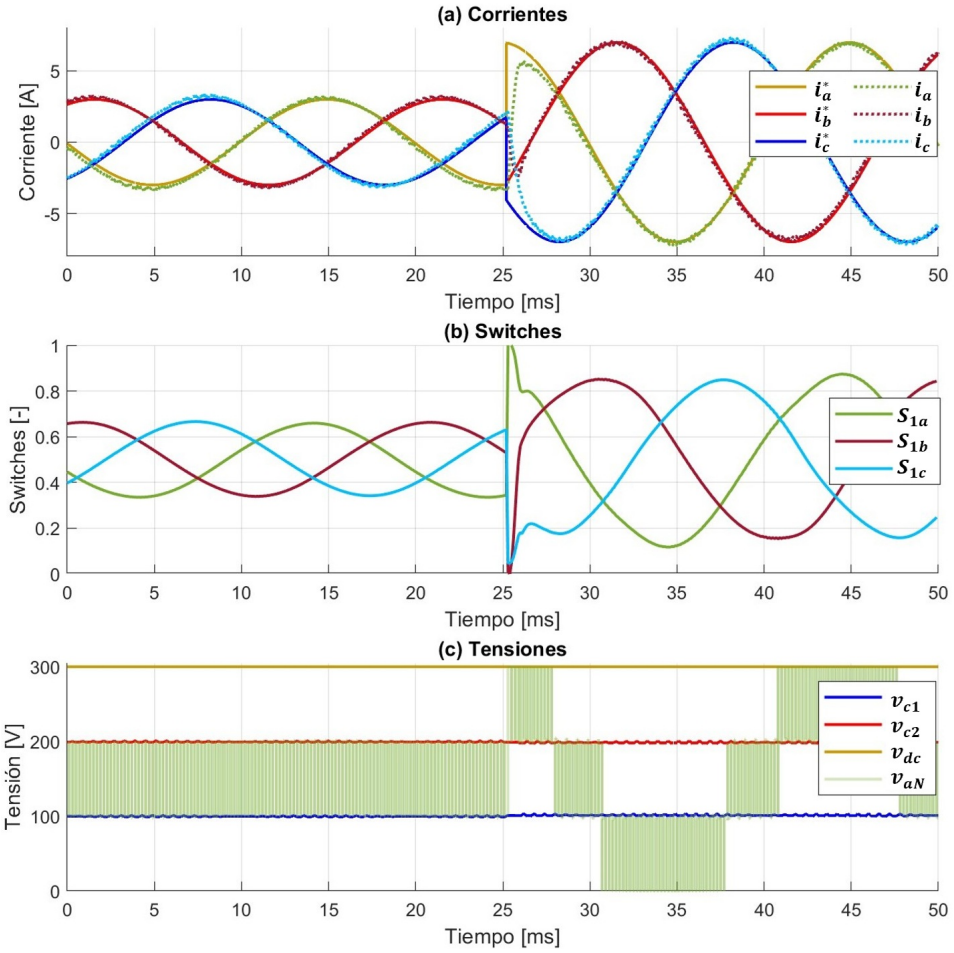



Figura 5.14: Corrientes, switches y tensiones para control ANN Dual en simulación.

Para este control, en la figura 5.14 se observa que si hay un seguimiento de la referencia, siendo las curvas suaves en todo momento. La red neuronal conjunta logra imitar al control FCS-MPC en el transitorio realizando en los switches unos peaks, que finalmente logran mejorar el comportamiento en relación al control PI.

El tiempo de asentamiento es de:

$$t_a = 2{,}4[ms] \tag{5.5}$$

En cuanto al estado estacionario, de la figura 5.15 se observa que el control con la ANN Dual logra corrientes cercanas a las referencias, aunque en los puntos máximos y mínimos estas sobrepasan ligeramente los valores de referencia. El error en estado estacionario de la corriente $i_a$ es de 0.43 [A] y 0.58 [A] para las referencias de -3 [A] y 7 [A], respectivamente, lo que representa una mejora significativa frente al control de unión de ANN mediante histéresis (o ANN control PI, por estar en estado estacionario), reduciendo los errores en aproximadamente un 50 %.

También, se calculó el THD de la tabla 5.4, donde se puede apreciar que los THD de

corriente son similares a los obtenidos con el control PI. Por otro lado, los de $v_{aN}$ son más altos que en el control PI pero tiene sentido, ya que al incluir el comportamiento del control FCS-MPC, se prioriza el que la corriente siga la referencia a costa de las formas de onda de tensión.

Tabla 5.4: THD para diferentes valores de corriente para control con ANN Dual.

| Amplitud | -3 [A] | 7 [A] |
|---|---|---|
| $i_a$ | 0.96 % | 1.72 % |
| $v_{aN}$ | 9.50 % | 11.47 % |

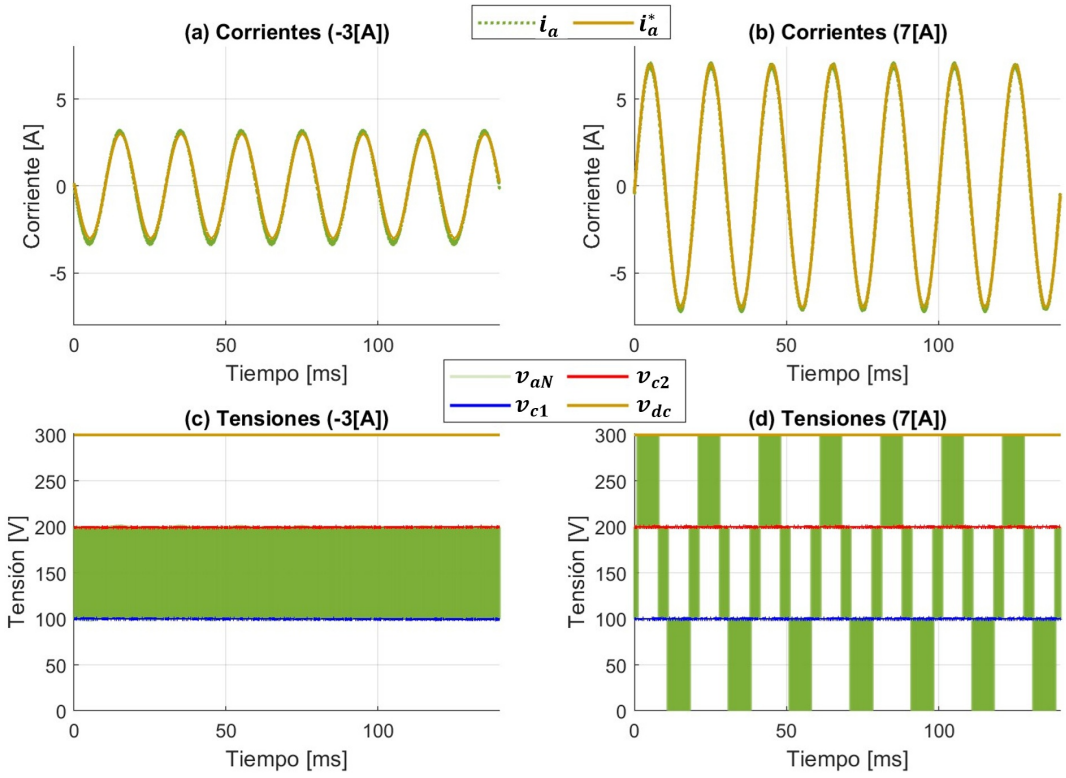



Figura 5.15: Corriente y tensiones de la fase A en estado estacionario con -3 [A] y 7 [A] para control con ANN Dual en simulación.

En las figuras 5.16 y 5.16 se muestra que para las corrientes, los espectros muestran una dominancia clara en la fundamental, con armónicos superiores atenuados. Los espectros de $v_{aN}$ muestran componentes armónicas en valores superiores, especialmente en valores cercanos a 5000[Hz] y 10000[Hz].

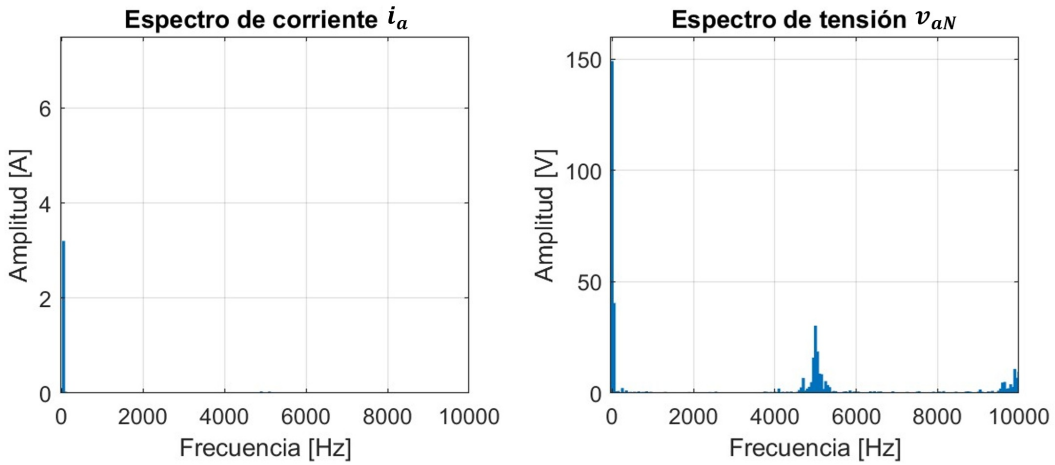



Figura 5.16: Espectro de corriente $i_a$ y tensión de salida $v_{aN}$ con -3 [A] para control con ANN Dual en simulación.

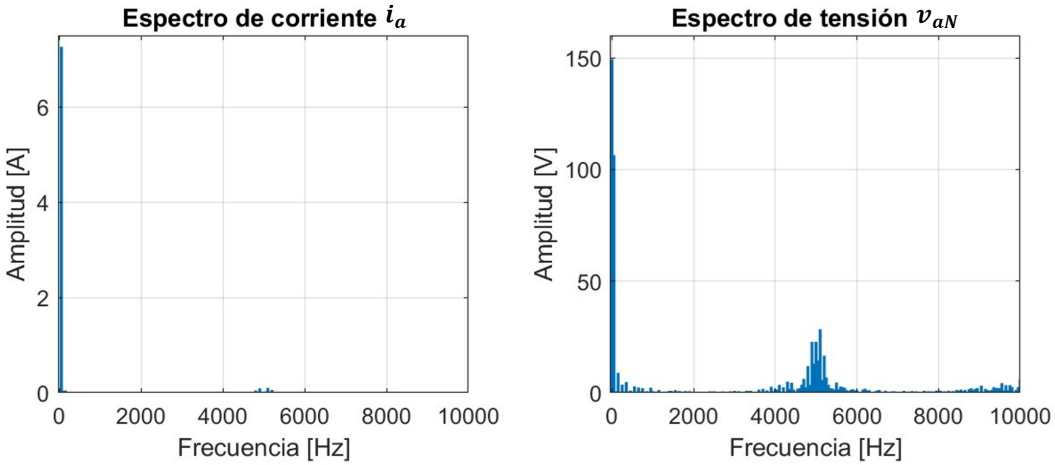



Figura 5.17: Espectro de corriente $i_a$ y tensión de salida $v_{aN}$ con 7 [A] para control con ANN Dual en simulación.

### 5.1.6. Análisis Comparativo de los controles en simulación

En la figura 5.18 se muestran los transitorios de corriente de todos los controles. Analizando el transitorio, como se preveía, el control lineal presenta mayores dificultades para seguir los cambios abruptos, mostrando sobreoscilación. El control con histéresis tiene un buen ajuste en el transitorio, tendiendo rápido a la referencia pero manteniendo oscilaciones en un tiempo prolongado por la rapidez del control PI utilizado, resultando tener un comportamiento similar al control ANN con Histéresis en términos de oscilaciones. Por otro lado, la ANN Dual logra transiciones más suaves y con menor ruido comparado con otros controles, lo cual sugiere una mejor capacidad para manejar las variaciones rápidas sin comprometer la estabilidad. También, se puede ver que el ruido del control FCS-MPC es menor que en los casos con histéresis, lo que evidencia el efecto de las bandas.

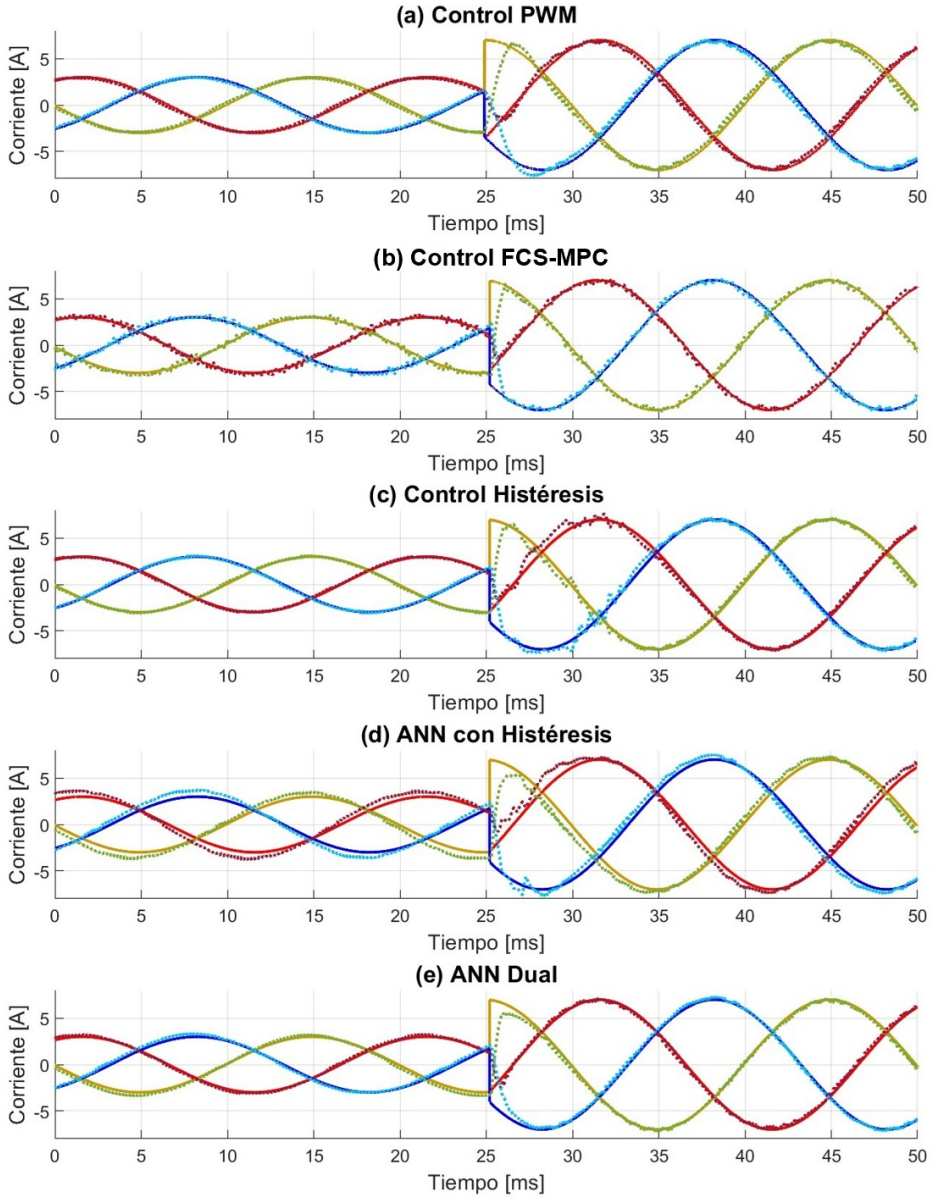



Figura 5.18: Gráficas del transitorio de corriente para todos los controles en simulación.

A continuación, en la tabla 5.5, se presentan los tiempos de asentamiento para todos los controles estudiados. Los tiempos de asentamiento más pequeños se obtienen con los controles FCS-MPC y ANN Dual, ambos alcanzando un valor de 0.9 y 2.4 [ms] respectivamente, lo que demuestra una respuesta dinámica altamente eficiente. El control Lineal también presenta un desempeño notable, con un tiempo de asentamiento de 3.2 [ms], mejorando respecto al ANN con histéresis, aunque sin igualar la rapidez de FCS-MPC o ANN Dual. Por otro lado, el control con Histéresis muestra el mayor tiempo de asentamiento, con 10.15 [ms], lo que indica una menor velocidad para alcanzar el estado estacionario.

Tabla 5.5: Tiempos de asentamiento $t_a$ para distintos tipos de control en simulación, ordenados de menor a mayor.

| **Control** | FCS-MPC | ANN Dual | Lineal | ANN con histéresis | Histéresis |
|---|---|---|---|---|---|
| $t_a$ **[ms]** | 0.9 | 2.4 | 3.2 | 5.7 | 10.2 |

En la tabla 5.6 se presentan los valores de THD para las variables de corriente y tensión bajo diferentes tipos de control y condiciones de referencia. Notar que estos valores se obtienen en estado estacionario, por lo que los resultados del control lineal y de histéresis son iguales, ya que, como se mencionó anteriormente, se activa el control lineal en estado estacionario.

El análisis de estos resultados revela que el control FCS-MPC presenta los valores de THD más elevados, especialmente para la tensión $v_{aN}$, alcanzando un 49.32 % para una corriente de referencia de -3 [A]. Estos resultados sugieren que, a pesar de la buena respuesta dinámica que ofrece el FCS-MPC, la calidad de la señal resultante se ve comprometida significativamente debido a la alta distorsión armónica.

Por otro lado, el control lineal (y de histéresis) muestra una calidad de señal superior, con valores de THD notablemente bajos para todas las condiciones, manteniéndose consistentemente por debajo del 1 %. Estos valores indican una buena capacidad para limitar la distorsión armónica y, por lo tanto, garantizar una señal más limpia y estable. El control basado en ANN con histéresis y el control ANN Dual se encuentran en una posición intermedia respecto a la calidad de señal. Estos controles basados en ANN ofrecen una solución intermedia que balancea entre velocidad de respuesta y una reducción parcial de la distorsión armónica, aunque sin llegar al nivel del control lineal.

Tabla 5.6: THD para distintos tipos de control en simulación.

| | **Control** | | | | |
|---|---|---|---|---|---|
| **Variables** | Lineal | Histéresis | ANN con histéresis | ANN Dual | FCS-MPC |
| $i_a$ **(-3[A])** | 0.78 % | 0.78 % | 1.24 % | 0.96 % | 6.60 % |
| $i_a$ **(7[A])** | 0.77 % | 0.77 % | 1.37 % | 1.72 % | 2.74 % |
| $v_{aN}$ **(-3[A])** | 0.87 % | 0.87 % | 19.14 % | 9.50 % | 49.32 % |
| $v_{aN}$ **(7[A])** | 0.80 % | 0.80 % | 11.52 % | 11.47 % | 17.13 % |

En la tabla 5.7 se presenta la amplitud de error estacionario de $i_a$ para los cinco tipos de control analizados, considerando las referencias de -3 [A] y 7 [A]. Se observa que, en

todos los casos, el error es mayor para la referencia de 7 [A]. El control basado en ANN con histéresis muestra consistentemente el mayor error estacionario en ambas referencias, mientras que el control Lineal y ANN Dual presentan los menores. El control FCS-MPC presenta valores parecidos a estos controles pero levemente más altos, aunque menores al ANN con Histéresis.

Tabla 5.7: Amplitud de error estacionario para la corriente $i_a$ para distintos tipos de control en simulación.

| | **Control** | | | | |
|---|---|---|---|---|---|
| **Variables** | Lineal | Histéresis | ANN Dual | FCS-MPC | ANN con histéresis |
| $i_a$ **(-3[A])** | 0.36[A] | 0.36[A] | 0.43[A] | 0.55[A] | 0.84[A] |
| $i_a$ **(7[A])** | 0.84[A] | 0.84[A] | 0.58[A] | 0.72[A] | 0.95[A] |

En resumen, en simulación el control con ANN Dual ofrece el mejor balance en términos de seguimiento y respuesta tanto en el régimen estacionario como en el transitorio.

## 5.2. Experimental

En esta sección se presenta el prototipo de laboratorio y los resultados experimentales obtenidos al implementar diferentes estrategias de control sobre el convertidor FCC de tres celdas. Las pruebas se realizaron para evaluar el desempeño de tres enfoques de control: el control PI, el control de unión de ANN con histéresis, y el control con ANN Dual.

El control PI se utilizó como referencia, ya que es una metodología de control clásica ampliamente utilizada en aplicaciones industriales. Se compara con las dos propuestas de control basadas en redes neuronales para identificar las ventajas y limitaciones de cada enfoque, proporcionando una comparación que contribuya a la selección del método de control más adecuado. Cada método fue evaluado bajo condiciones operativas similares, con el objetivo de comparar y contrastar los resultados en cuanto a transitorios, estado estacionario y carga computacional.

### 5.2.1. Prototipo de laboratorio

En la figura 5.19 se muestra el *setup* para realizar las mediciones experimentales. Se puede observar que el FCC consta de tres placas, una para cada fase. En cada placa, además de las tres celdas, se encuentran las conexiones al circuito de medición, como se muestra en la figura 5.20 (b), y al circuito de disparo mediante una tarjeta receptora, según lo mostrado en la figura 5.20 (a). Ambas conexiones se dirigen a la plataforma de control *BRAIn (Board for Research, Academic and Industrial applications)* mostrada en 5.20 (c), que permite la programación de las estrategias de control en un DSP (Digital Signal Processor) y la integración de módulos periféricos a través de una FPGA (Field Programmable Gate Array), los cuales se utilizan tanto para la generación de señales como para la medición mediante ADCs (Analog-to-Digital Converters).

En el diagrama de la figura 5.21 se puede entender mejor la comunicación entre la *BRAIn* y el FCC. Esta se realiza mediante un enlace de fibra óptica para enviar los pulsos de disparo, utilizando la tarjeta de expansión *PowerEloBase* que se encuentra en la placa inferior de la *BRAIn*, desde donde se envían las señales de fibra óptica a las tarjetas receptoras acopladas a cada placa del convertidor. Las estrategias de control se programaron utilizando el software *Code Composer Studio*, en lenguaje de programación *C*. Este software, al ser un compilador, permite transformar el código en *C* a *Assembler*.

Las mediciones de corriente de salida y tensiones de los capacitores provenientes de las tarjetas receptoras del convertidor se reciben mediante la misma tarjeta de expansión *PowerEloBase* hacia la plataforma *BRAIn*. La tarjeta en la parte superior de la *BRAIn* permite la comunicación con el software *MATLAB*, para así poder medir y almacenar los datos para su posterior análisis. También se utilizan dos osciloscopios para ver el comportamiento de las variables en tiempo real y almacenar los datos de corrientes en uno, y de tensiones de los capacitores y salida $v_{aN}$ en otro.

La plataforma *BRAIn* cuenta con un DSP TMS320C6748 de *Texas Instruments*, una FPGA XCA35T, 120 puertos GPIO y 12 canales analógicos, entre otras cualidades que la hacen adecuada para usos académicos e industriales. Para más detalle sobre su funcionamiento y características consultar [16].

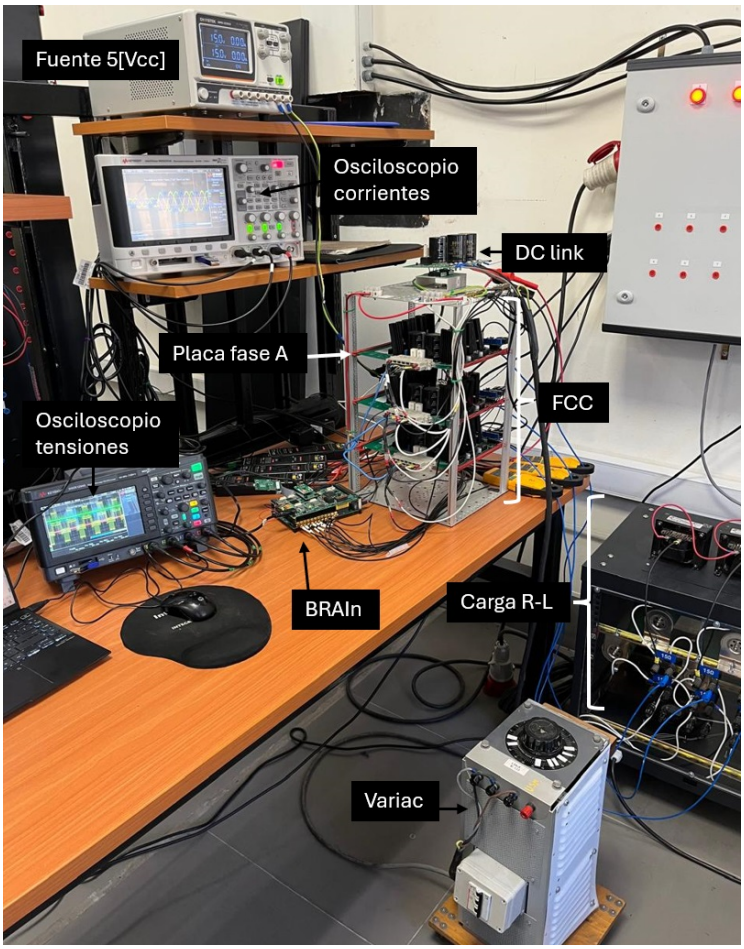



Figura 5.19: Setup laboratorio.

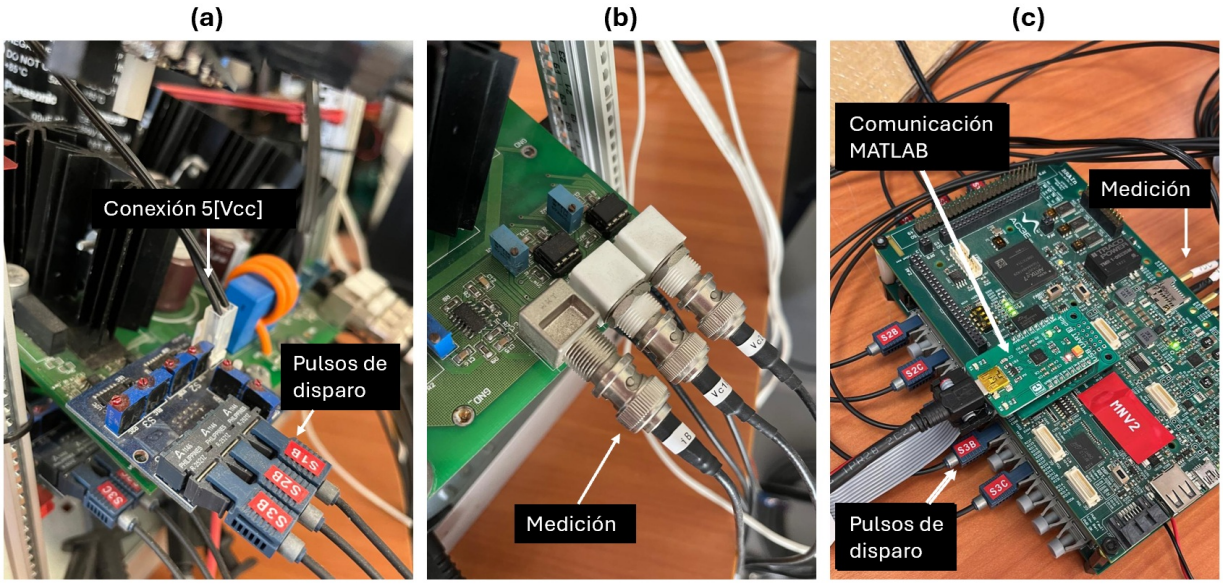



Figura 5.20: Zoom del *setup* para: (a) Receptor pulsos de disparo en la placa B. (b) Medición de la placa B del FCC. (c) BRAIn.

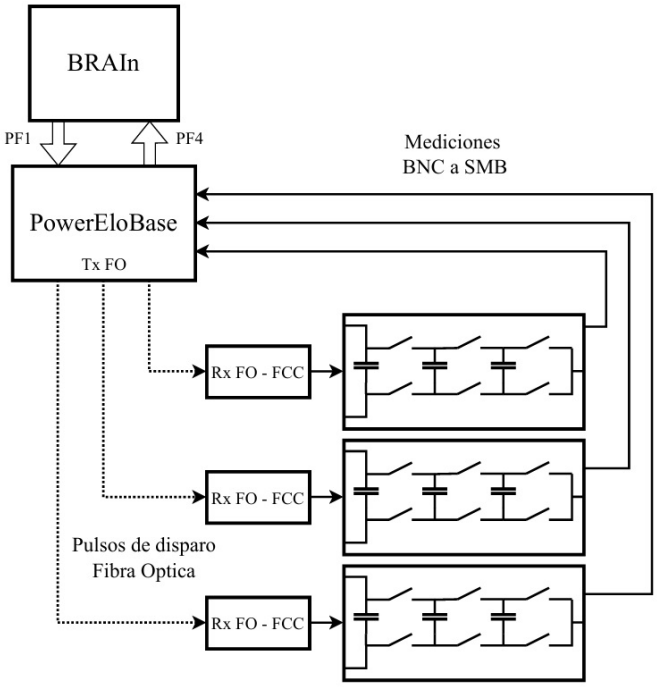



Figura 5.21: Diagrama de comunicación BRAIn-FCC. [7]

Los valores de los parámetros del FCC y la carga se intentaron igualar a los utilizados en simulación. En la tabla 5.8 se muestran los valores medidos.

Tabla 5.8: Parámetros experimentales de la carga y placa.

| **Elemento** | **Parámetro** | **Valor** |
|---|---|---|
| Placa capacitor flotante | $C_1$ | 330 [$\mu$F] |
| | $C_2$ | 330 [$\mu$F] |
| | $C_3$ | 900 [$\mu$F] |
| Carga | $R_1$ | 15.7 [$\Omega$] |
| | $L_1$ | 10.1 [mH] |
| | $R_2$ | 15.3 [$\Omega$] |
| | $L_2$ | 10.3 [mH] |
| | $R_3$ | 15.4 [$\Omega$] |
| | $L_3$ | 9.9 [mH] |

En los puntos 5.2.2, 5.2.3, y 5.2.4 se analizan los tres tipos de control probados experimentalmente, de la misma manera que en la sección anterior, a modo de poder luego comparar los comportamientos entre sí. Como se mencionó previamente, el control PI se toma como referencia para el análisis de los otros controles propuestos basados en redes neuronales.

### 5.2.2. Control PI

Al igual que en simulación, se analizó el transitorio al variar la amplitud de corriente de -3[A] a 7[A]. En la figura 5.22 se observa que durante el transitorio las corrientes tienen notables problemas para seguir a la referencia. Las tensiones de los capacitores se muestran estables gracias a la modulación PS-PWM a pesar de aumentar la magnitud de la corriente, lo que devela un buen control. En contraste, la tensión $V_{dc}$ cae al momento de hacer el cambio de referencia debido a que el sistema consume potencia desde el Variac (que no tiene regulación de tensión) por lo que al aumentar la corriente a 7[A] baja la tensión del dc-link.

El tiempo de asentamiento, en que las tres fases logran estabilizarse, es de:

$$t_a = 9,1[ms] \tag{5.6}$$

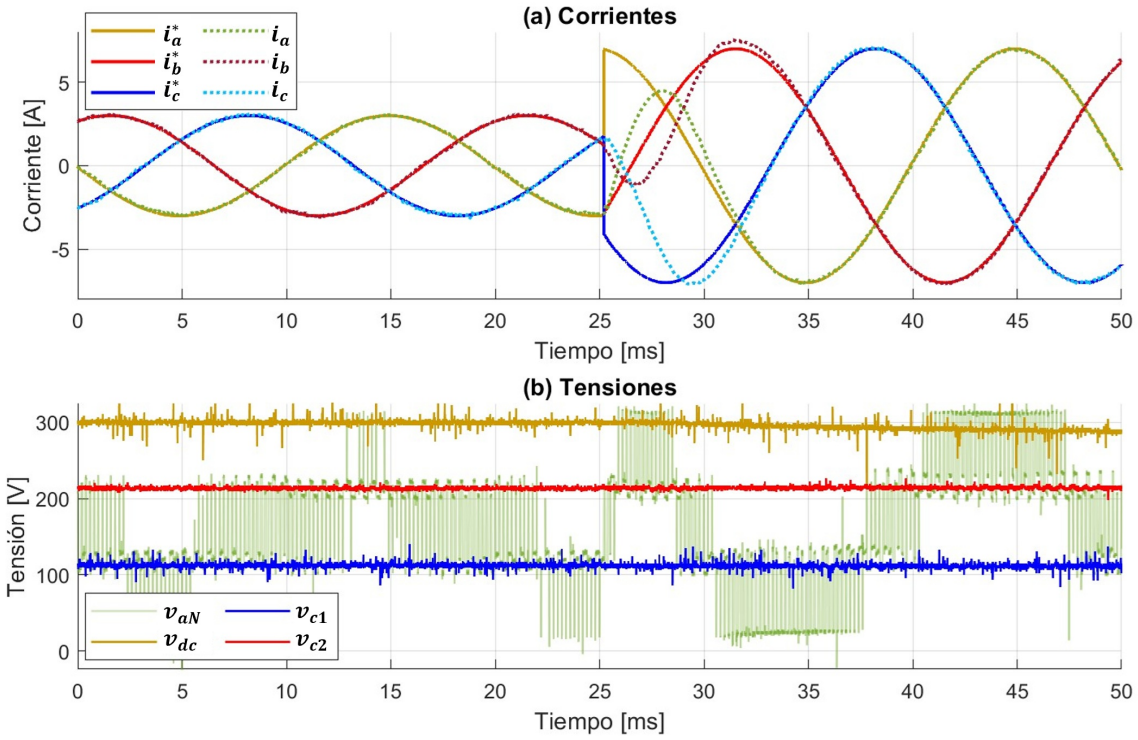



Figura 5.22: Corrientes y tensiones para control PI experimental.

En cuanto al estado estacionario, se analizan las curvas de la figura 5.23, donde se ve que para ambas amplitudes se sigue a la referencia de corriente de manera precisa y estable, demostrando un excelente desempeño en estado estacionario. La tensión $v_{aN}$ muestra cuatro niveles para las dos amplitudes de corriente.

Se calculó el THD mostrado en la tabla 5.9 y el espectro de las figuras 5.24 y 5.25.

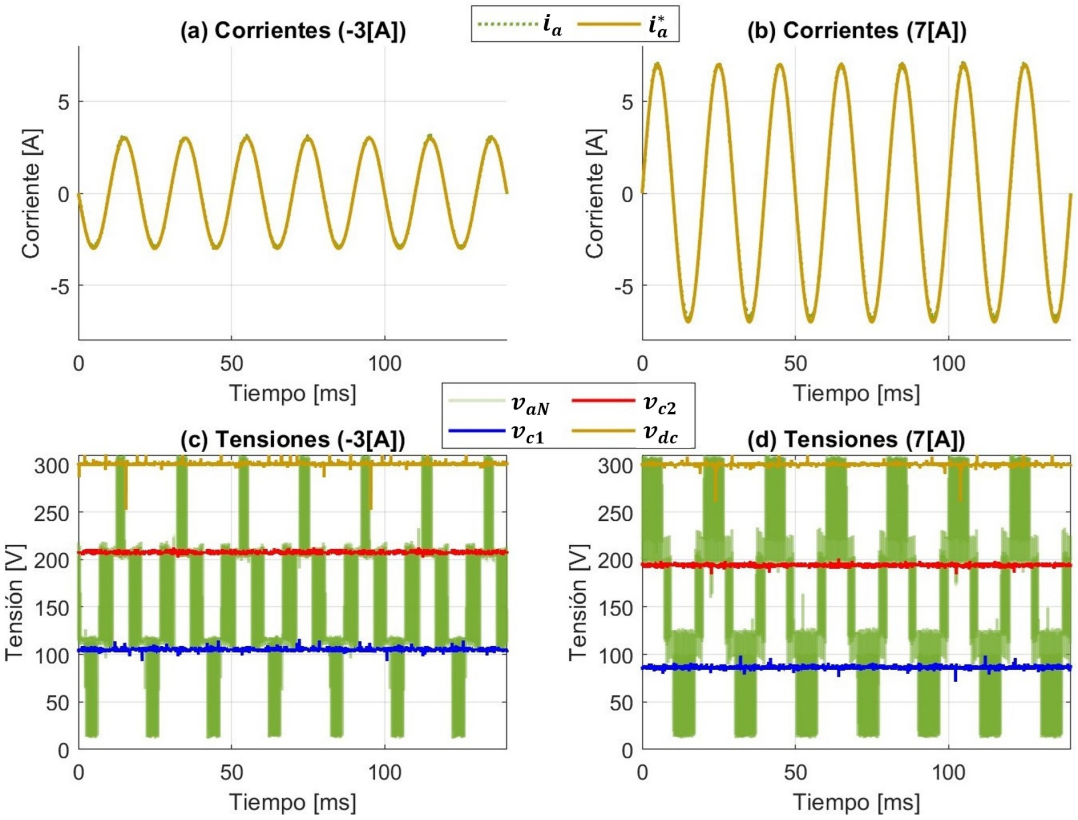



Figura 5.23: Corriente y tensiones de la fase A en estado estacionario con -3 [A] y 7 [A] para control PI experimental.

Tabla 5.9: THD para diferentes valores de corriente para control PI experimental.

| Amplitud | -3 [A] | 7 [A] |
|---|---|---|
| $i_a$ | 2.29 % | 1.61 % |
| $v_{aN}$ | 19.41 % | 14.11 % |

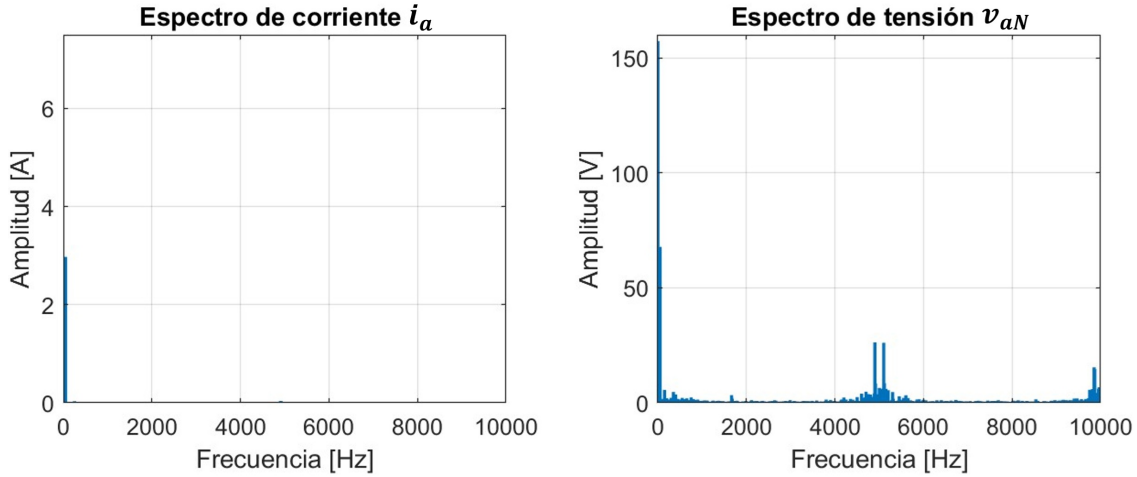



Figura 5.24: Espectro de corriente *ia* y tensión de salida $v_{aN}$ con -3 [A] para control PI experimental.

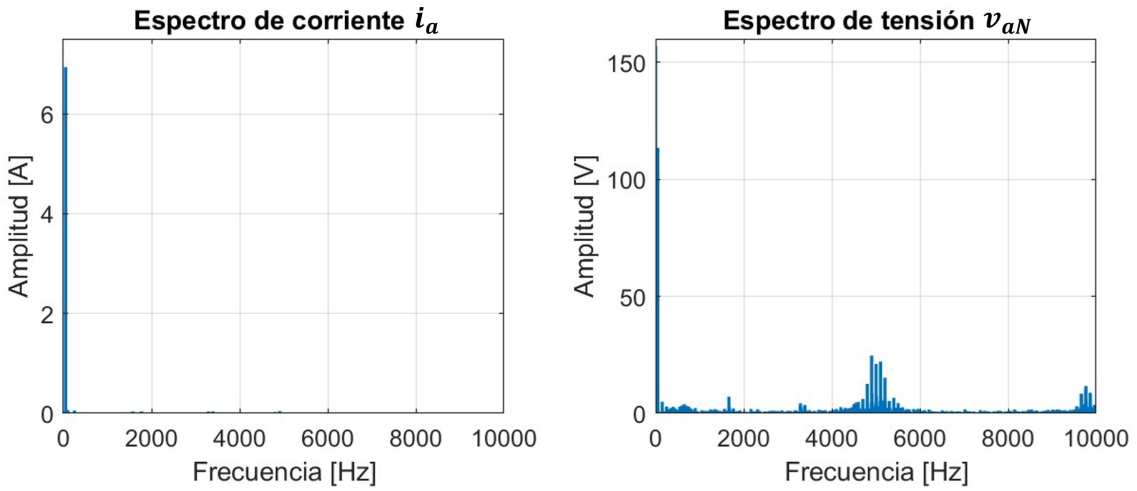



Figura 5.25: Espectro de corriente $i_a$ y tensión de salida $v_{aN}$ con 7 [A] para control PI experimental.

Se muestra que el control PI regula de forma efectiva la corriente $i_a$, con un bajo THD, sin embargo, el THD de la tensión $v_{aN}$ es significativamente más alto indicando mayor presencia de armónicos, lo que también se muestra en las gráficas de espectro. Los espectros revelan un peak dominante en la frecuencia fundamental en ambas señales, y para las tensiones también en frecuencia cero por el valor medio de la señal.

Operar con mayor amplitud de corriente mejoró el THD tanto en la corriente como en la tensión, sugiriendo mayor eficiencia a mayores niveles de corriente.

### 5.2.3. Unión de ANN con histéresis

Al implementar experimentalmente la unión de ANN PI con ANN FCS-MPC mediante bandas de histéresis, se tiene a partir de la figura 5.26 que, durante el transitorio las corrientes intentan seguir a la referencia al activarse la ANN que imita el control FCS-MPC, aunque su comportamiento es sumamente ruidoso. También se puede apreciar que hay problemas de seguimiento a la referencia cuando la amplitud de corriente es -3[A], concretamente con el comportamiento de la ANN lineal. Las tensiones de los capacitores presentan oscilaciones en el transitorio.

Se consideró el tiempo de asentamiento hasta el momento en que efectivamente se sigue la referencia una vez que se vuelve al control con la ANN lineal, este se muestra a continuación:

$$t_a = 16[ms] \tag{5.7}$$

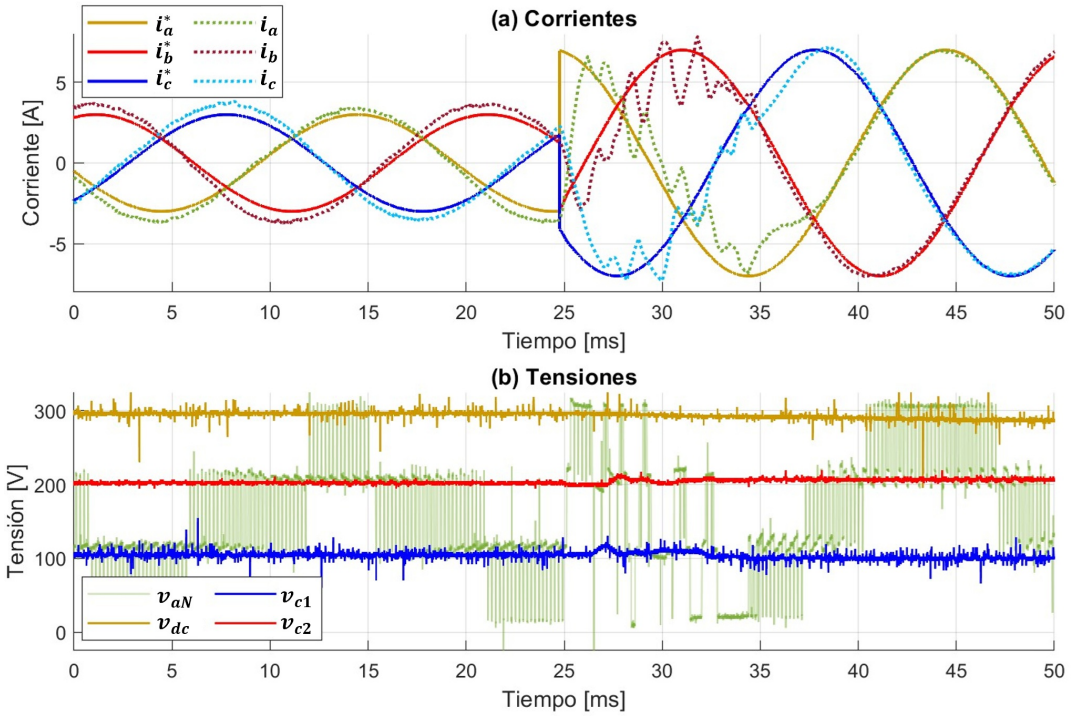



Figura 5.26: Corrientes y tensiones para control con unión de ANN mediante histéresis experimental.

En cuanto al estado estacionario, al igual que en simulación, en las gráficas de la figura 5.27 se ve que para la amplitud de corriente de -3[A] el control no es efectivo ya que no hay un seguimiento de la corriente de referencia, mientras que para 7[A] el control tuvo un mejor desempeño. Respecto a las tensiones de los capacitores, estas se mantienen estables. La tensión $vaN$ muestra cuatro niveles para las dos amplitudes.

En la tabla 5.10 se muestra el THD calculado a partir de las gráficas de la figura 5.27.

Tabla 5.10: THD para diferentes valores de corriente para control de unión de ANN con histéresis.

| Amplitud | -3 [A] | 7 [A] |
|---|---|---|
| $i_a$ | 3.10 % | 2.14 % |
| $v_{aN}$ | 14.04 % | 4.88 % |

En el caso de la corriente $i_a$, el THD es bajo, este resultado indica que las corrientes presentan en general una calidad aceptable de señal a pesar de no seguir a la referencia, con una baja contribución de armónicos respecto a la componente fundamental. En los espectros de corriente se observa que la frecuencia fundamental domina, mientras que las componentes armónicas superiores tienen una amplitud insignificante, lo que respalda el bajo valor del THD.

En cuanto a la tensión $v_{aN}$, el THD es mayor y el espectro de tensión también muestra un claro predominio de la frecuencia fundamental a 50[Hz] y de la armónica en frecuencia cero. Las componentes armónicas presentes, como en 1,67[kHz] y superiores, contribuyen al THD observado, pero su magnitud es menor respecto a la fundamental.

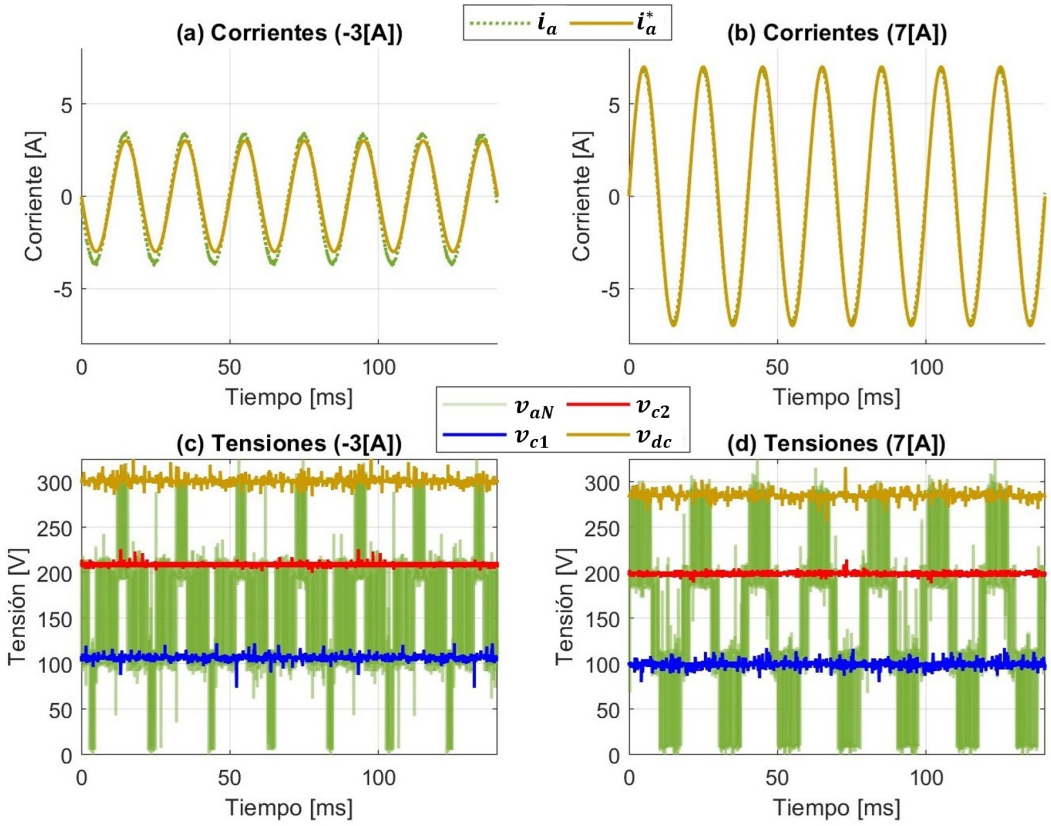



Figura 5.27: Corriente y tensiones de la fase A en estado estacionario con -3 [A] y 7 [A] para control de unión de ANN con histéresis experimental.

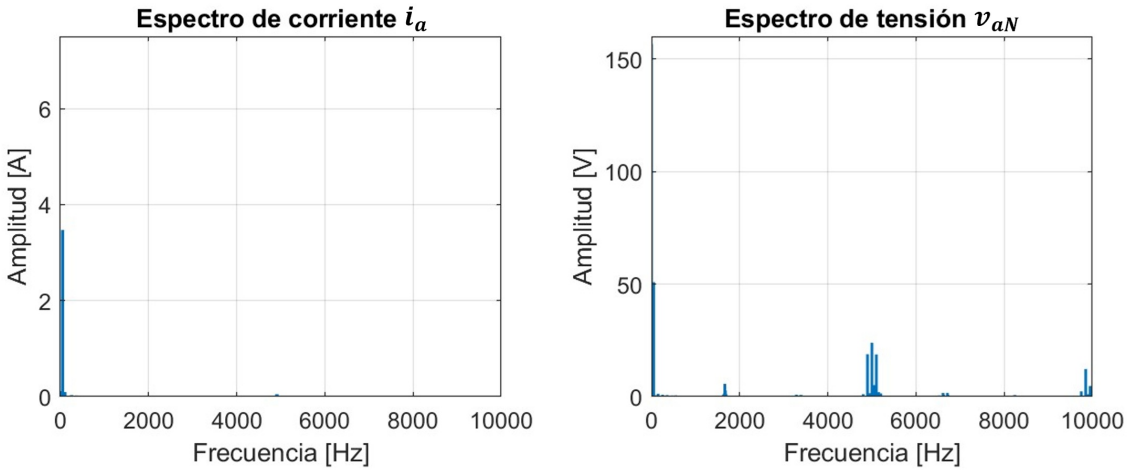



Figura 5.28: Espectro de corriente $i_a$ y tensión de salida $v_{aN}$ con -3 [A] para control de unión de ANN con histéresis experimental.

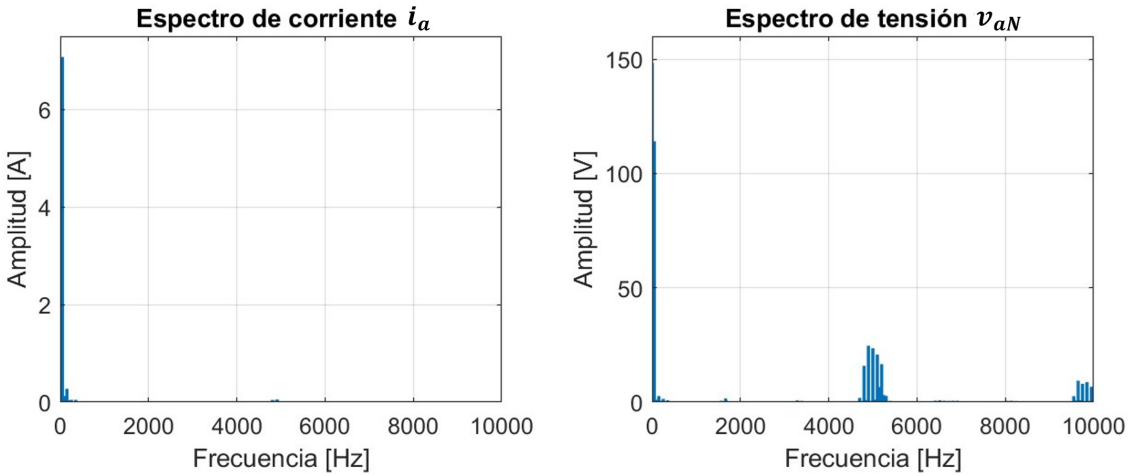



Figura 5.29: Espectro de corriente $i_a$ y tensión de salida $v_{aN}$ con 7 [A] para control de unión de ANN con histéresis experimental.

### 5.2.4. ANN Dual

La figura 5.30 muestra que, si bien durante el transitorio las corrientes no logran seguir a la referencia como lo haría un control FCS-MPC, este control con ANN Dual tiene curvas suaves que estabilizan rápidamente las corrientes, presentando un tiempo de asentamiento menor al del control PI expuesto en 5.6, siendo éste de:

$$t_a = 5,6[ms] \tag{5.8}$$

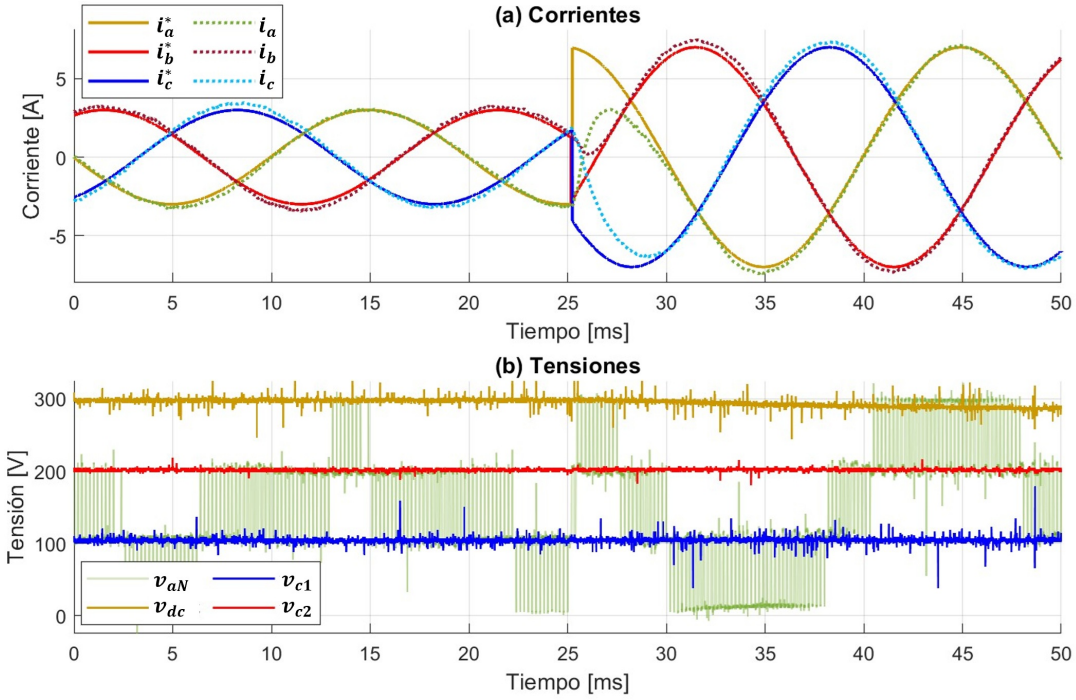



Figura 5.30: Corrientes y tensiones para control ANN Dual experimental.

Es importante mencionar que las corrientes obtenidas a partir del control de la ANN Dual tienen algunas fases que presentan leves desplazamientos verticales o bien, máximos o mínimos mas grandes que la referencia, que se dejan en evidencia en la imagen anterior.

Respecto al estado estacionario, en las gráficas de la figura 5.31 se puede ver que las corrientes tienen un buen seguimiento de la referencia. Las tensiones de los capacitores se mantienen estables. La tensión $v_{aN}$ muestra cuatro niveles para las dos amplitudes.

En la tabla 5.10 se muestra el THD, y en las figuras 5.32 y 5.33 el espectro calculado a partir de las gráficas de la figura 5.31.

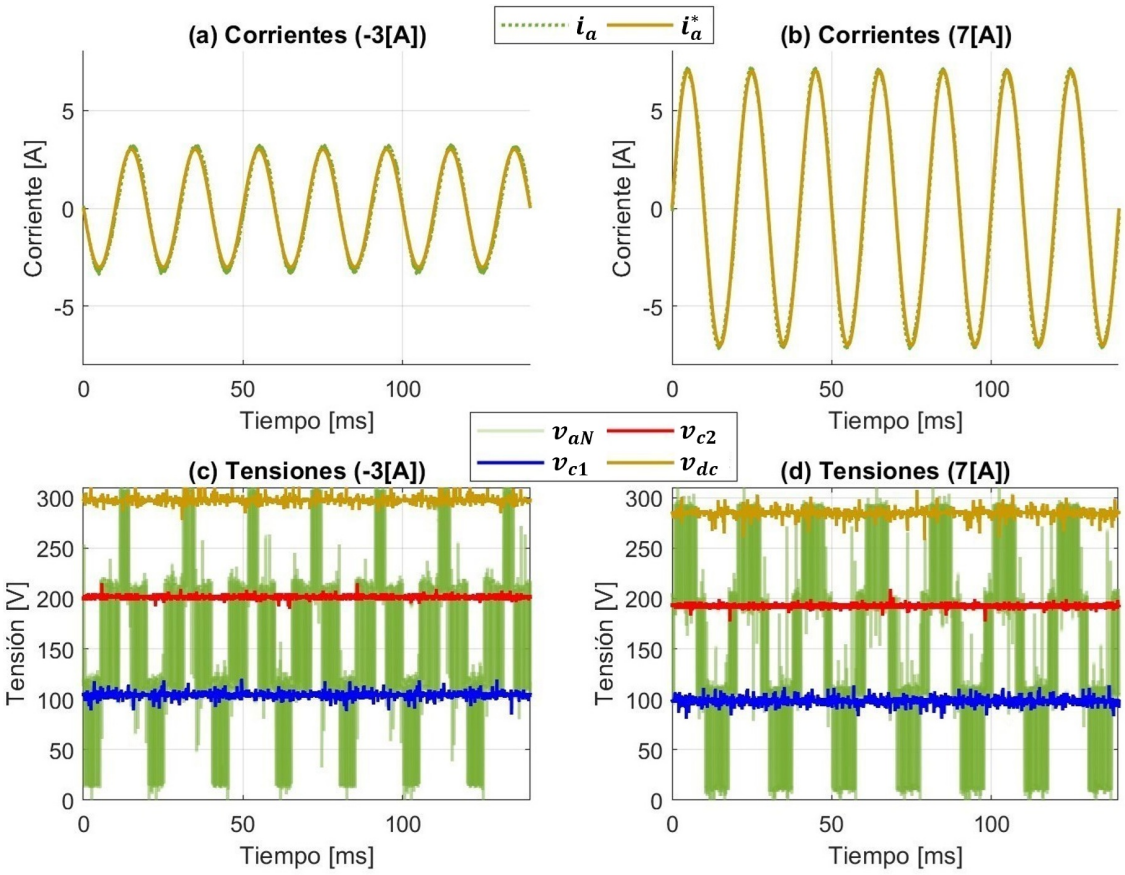



Figura 5.31: Corriente y tensiones de la fase A en estado estacionario con -3 [A] y 7 [A] para control ANN Dual experimental.

Tabla 5.11: THD para diferentes valores de corriente para control ANN Dual.

| Amplitud | -3 [A] | 7 [A] |
|---|---|---|
| $i_a$ | 3.78 % | 4.83 % |
| $v_{aN}$ | 8.92 % | 3.67 % |

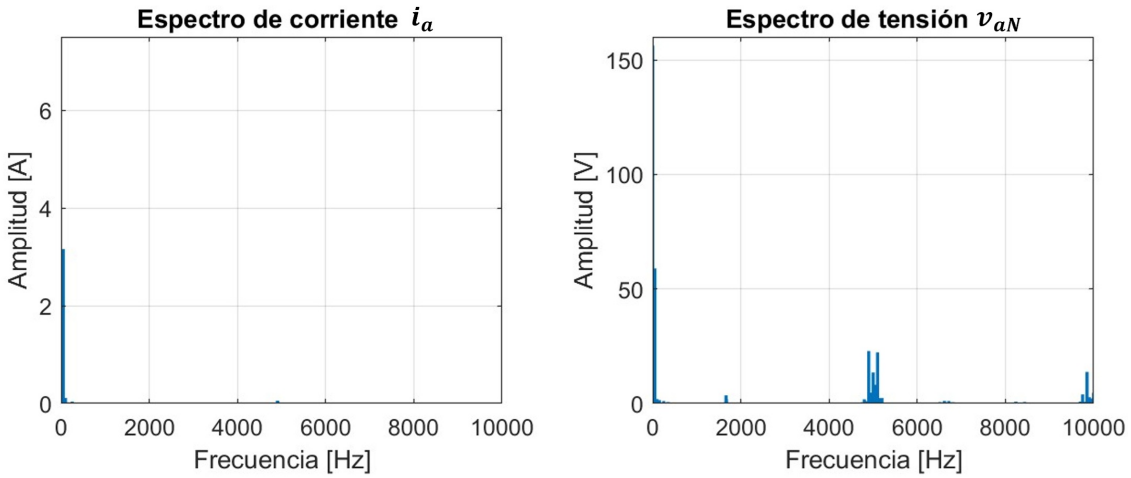



Figura 5.32: Espectro de corriente $i_a$ y tensión de salida $v_{aN}$ con -3 [A] para control ANN Dual experimental.

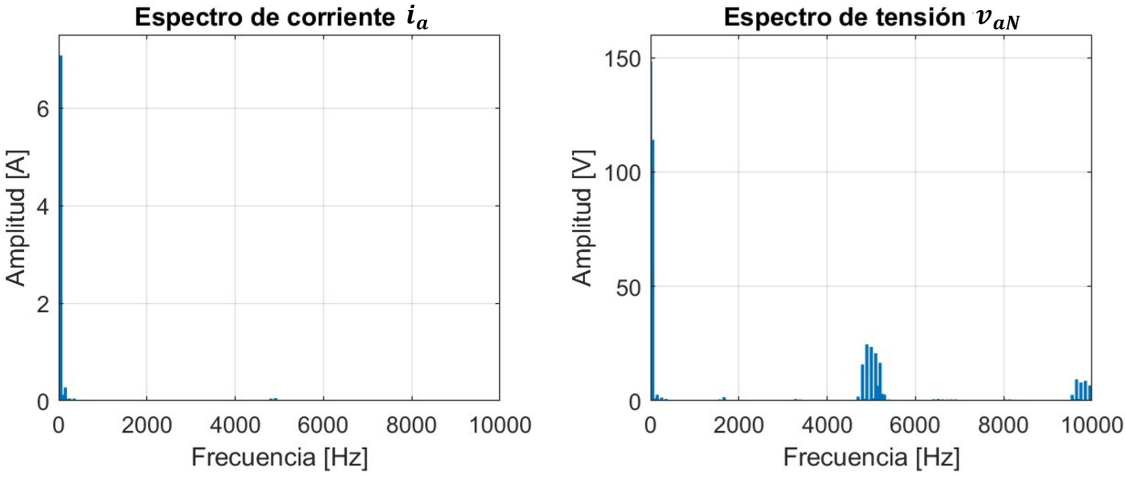



Figura 5.33: Espectro de corriente $i_a$ y tensión de salida $v_{aN}$ con 7 [A] para control ANN Dual experimental.

A grandes rasgos se evidencia un comportamiento eficiente del sistema tanto en las corrientes como en las tensiones. Para la corriente $i_a$, el THD es del 3.78 % con una amplitud de -3[A] y aumenta a 4.83 % para 7[A]. Los espectros muestran un claro predominio de la frecuencia fundamental a 50[Hz], con armónicos superiores de amplitud baja, lo que indica una señal bien controlada y con bajas distorsiones.

En cuanto a la tensión $v_{aN}$, el THD es del 8.92 % para -3[A] y disminuye a 3.67 % para 7[A], reflejando una mayor calidad de la señal bajo mayores niveles de corriente. Los espectros de tensión también están dominados por la frecuencia fundamental a 50 , pero presentan armónicos con una amplitud mayor que en las corrientes.

### 5.2.5. Análisis Comparativo de los controles experimentales

En la figura 5.34 se muestran los transitorios de corriente de todos los controles probados experimentalmente.

Se ve que el control con ANN Dual es el que presenta el mejor desempeño, destacándose por su rapidez, precisión y estabilidad. En comparación con el control PI, este tiene un comportamiento equivalente en estado estacionario pero en transitorio logra mejorar el desempeño al tener una respuesta más rápida y parecida a la de la referencia, evitando sobreimpulsos. Por lo anterior se infiere que esta propuesta de control tiene mejoras con respecto al control de referencia (control PI). Por otro lado, el control de ANN con Histéresis actuó de manera más rápida al cambio de amplitud, intentando seguir la referencia antes que la ANN Dual pero de una manera sumamente ruidosa.

Si bien el control PI tiene evidentes limitaciones en el transitorio, el control de ANN con Histéresis no cumple el objetivo de mejorar el control, principalmente porque hay una limitante con la ANN Lineal que no logra seguir a la referencia para determinadas corrientes de referencia (como el caso de -3[A]) y también, porque en el transitorio al activarse la ANN MPC, el ruido es excesivo.

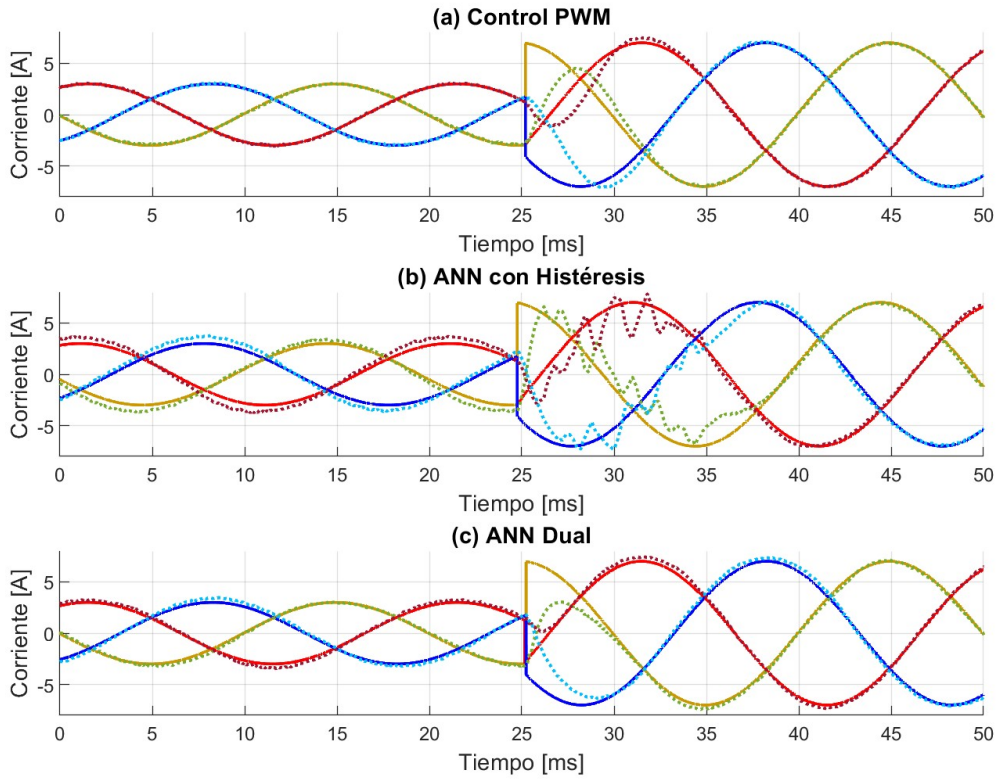



Figura 5.34: Gráficas del transitorio de corriente para todos los controles experimentales.

A continuación, en la tabla 5.12, se presentan los tiempos de asentamiento para los tres controles, siendo el menor el del control con ANN Dual y el mayor el del control de ANN con Histéresis.

Tabla 5.12: Tiempos de asentamiento $t_a$ para distintos tipos de control experimental.

| **Control** | ANN Dual | Lineal | ANN con Histéresis |
|---|---|---|---|
| $t_a$ **[ms]** | 5.6 | 9.1 | 16 |

En la tabla 5.13 se presentan los valores de THD para los controles lineal, ANN con histéresis y ANN Dual.

El control lineal se destaca por tener el menor THD en las corrientes $i_a$ , lo que indica estabilidad y bajo contenido armónico en estas variables. Se puede ver que el control con ANN Dual presenta el mayor contenido armónico de corriente respecto el resto de los controles, aunque el menor en cuanto a la tensión $v_{aN}$. De todas formas, el THD de corriente de todos los controles es menor al 5 %, lo que se considera aceptable.

Tabla 5.13: THD para distintos tipos de control experimentales.

| | **Control** | | |
|---|---|---|---|
| **Variables** | Lineal | ANN con Histéresis | ANN Dual |
| $i_a$ **(-3[A])** | 2.29 % | 3.10 % | 3.78 % |
| $i_a$ **(7[A])** | 1.61 % | 2.14 % | 4.83 % |
| $v_{aN}$ **(-3[A])** | 19.40 % | 14.04 % | 8.92 % |
| $v_{aN}$ **(7[A])** | 14.11 % | 4.88 % | 3.67 % |

En la tabla 5.14 se expone la amplitud de error estacionario de $i_a$ para los tres tipos de control analizados, considerando las referencias de -3 [A] y 7 [A]. Se observa que el control con menor error estacionario corresponde al Lineal, seguido por el ANN Dual. En contraste, el mayor error se presenta con el control ANN con Histéresis, el cual se diferencia por tener error más alto para la referencia de -3[A], en comparación con el resto de los controles.

Tabla 5.14: Amplitud de error estacionario para la corriente $i_a$ para distintos tipos de control experimentales.

| | **Control** | | |
|---|---|---|---|
| **Variables** | Lineal | ANN Dual | ANN con Histéresis |
| $i_a$ **(-3[A])** | 0.16[A] | 0.54[A] | 0.98[A] |
| $i_a$ **(7[A])** | 0.30[A] | 0.65[A] | 0.66[A] |

En cuanto al costo computacional, se evaluó midiendo el número de ciclos de reloj que cada rutina requiere para ejecutarse en el DSP. Para garantizar la validez de la comparación, tanto el código como las variables asociadas se ubicaron en una memoria cercana al procesador, minimizando los tiempos de acceso y asegurando condiciones homogéneas. Asimismo, se diseñaron las rutinas con estructuras de código similares para reducir las variaciones atribuibles a la implementación. Las mediciones se realizaron utilizando herramientas del entorno *Code Composer Studio*, como *Profile Clock* y *Count Events*, que permitieron registrar de manera precisa los ciclos de reloj entre puntos clave del código.

En la tabla 5.15 se muestra el número de ciclos de reloj utilizados y el tiempo correspondiente en microsegundos. Se puede ver, que de los controles estudiados en este trabajo, el control lineal es el más eficiente, mientras que la ANN con histéresis tiene el mayor costo computacional. La ANN Dual presenta un desempeño intermedio, mostrando una mejora significativa en comparación con la ANN con histéresis, pero menos eficiente que el control lineal.

Es importante destacar que la carga computacional de la red neuronal propuesta se considera un éxito, dado que la ANN Dual incorpora el efecto del control FCS-MPC. En el trabajo previo descrito en [7], el control FCS-MPC presentó una carga computacional de 21046 ciclos, mientras que la red neuronal ANN FCS-MPC alcanzó una carga de 23467 ciclos. Esto resalta la eficiencia de la ANN Dual al integrar las características del FCS-MPC de manera simple, con un impacto computacional competitivo.

Tabla 5.15: Costo computacional para distintos tipos de control experimental.

| **Control** | Lineal | ANN Dual | FCS-MPC | ANN con Histéresis | ANN FCS-MPC |
|---|---|---|---|---|---|
| **Ciclos de reloj** | 6878 | 10795 | 21046 | 21731 | 23467 |
| **Tiempo** [$\mu$**s**] | 22.93 | 35.98 | 70.00 | 72.44 | 78.20 |

## 5.3. Comparación resultados simulación - experimental

En esta sección, se presenta una comparación entre los resultados obtenidos mediante simulación y aquellos registrados en pruebas experimentales para los tres tipos de control implementados en el FCC. El objetivo principal de esta comparación es evaluar la precisión y consistencia del modelo teórico desarrollado frente al comportamiento real del sistema.

El análisis del transitorio se centra en los tiempos de asentamiento y la comparación de las gráficas de corriente y tensión obtenidas en simulación y experimentación para los tres controles estudiados.

En la tabla 5.16 se observan diferencias significativas entre los tiempos de asentamiento en simulación y experimentación. Los tiempos en experimentación son consistentemente mayores debido a factores no modelados en la simulación, como el ruido, las pérdidas en los componentes y la latencia de los dispositivos de medición y control.

El control lineal muestra el menor tiempo de asentamiento en simulación (3.2[ms]), pero en experimentación este aumenta significativamente a 9.1[ms], reflejando que, aunque efectivo en condiciones ideales, es sensible a perturbaciones en el entorno real. Por otro lado, el enfoque basado en ANN con histéresis presenta el mayor tiempo de asentamiento tanto en simulación como en experimentación (5.7[ms] y 16.0[ms], respectivamente), lo que evidencia una mayor complejidad en su comportamiento frente a cambios en el sistema. Finalmente, el control ANN Dual destaca con el mejor desempeño en simulación (2.4[ms]) y un tiempo intermedio en experimentación (5.6[ms]), aunque en la práctica los retrasos asociados al hardware afectan su desempeño de forma no despreciable.

Tabla 5.16: Tiempos de asentamiento $t_a$ para distintos tipos de control en simulación y experimentación.

| | **Control** | | |
|---|---|---|---|
| **Tipo** | ANN Dual | Lineal | ANN con Histéresis |
| Simulación | 2.4 [ms] | 3.2 [ms] | 5.7 [ms] |
| Experimentación | 5.6 [ms] | 9.1 [ms] | 16.0 [ms] |

Las figuras 5.35, 5.36 y 5.37 muestran las gráficas de corriente y tensión. En todos los casos, se observan diferencias entre simulación y experimentación, siendo las más notorias para el control ANN con histéresis.

En el caso del control lineal en la figura 5.35 se ve que hay diferencias en la forma de la tensión $v_{aN}$ entre simulación y experimental cuando la amplitud de corriente es de -3[A], presentando dos niveles para la simulación y cuatro para el experimental. En cuanto al comportamiento transitorio de las corrientes, las curvas experimentales alcanzan amplitudes menores a pesar de que las constantes de control del filtro se mantuvieron iguales, lo que sugiere discrepancias en la respuesta del sistema real frente al modelo teórico. Estas observaciones indican que, si el control de referencia experimenta cambios notables al pasar de simulación a experimental, es probable que los controles basados en redes neuronales también presenten adaptaciones similares.

En el control ANN con histéresis de la figura 5.36, las tensiones presentan un comportamiento más oscilatorio en el caso experimental en comparación con la simulación, destacándose que la ANN basada en FCS-MPC extendió su tiempo de control más allá de lo observado en la simulación. Aunque en simulación esta estrategia mostró un alto potencial para mantener un seguimiento preciso de la referencia en todo momento, en el entorno experimental no se logró replicar este desempeño con la misma eficacia. Esto puede atribuirse a que la ANN FCS-MPC fue entrenada con datos provenientes de una simulación que utiliza parámetros específicos, los cuales difieren ligeramente de los valores reales en el sistema experimental, lo que provoca discrepancias en el comportamiento de la red neuronal al adaptarse a las condiciones prácticas.

Para el control ANN Dual mostrado en la figura 5.37, al igual que en el control lineal, la tensión $v_{aN}$ muestra cuatro niveles en el caso experimental para una amplitud de corriente de -3 [A], mientras que en simulación se observan únicamente dos niveles; no obstante, el comportamiento general es equivalente. En el análisis del transitorio de las corrientes, las curvas experimentales presentan un desempeño similar al de la simulación, destacándose por su rápida estabilización, aunque con amplitudes menores respecto a las simulaciones. Las diferencias observadas son menos pronunciadas en comparación con los otros controles, lo que resalta la robustez y adaptabilidad de este enfoque frente a las discrepancias entre el entorno simulado y experimental.

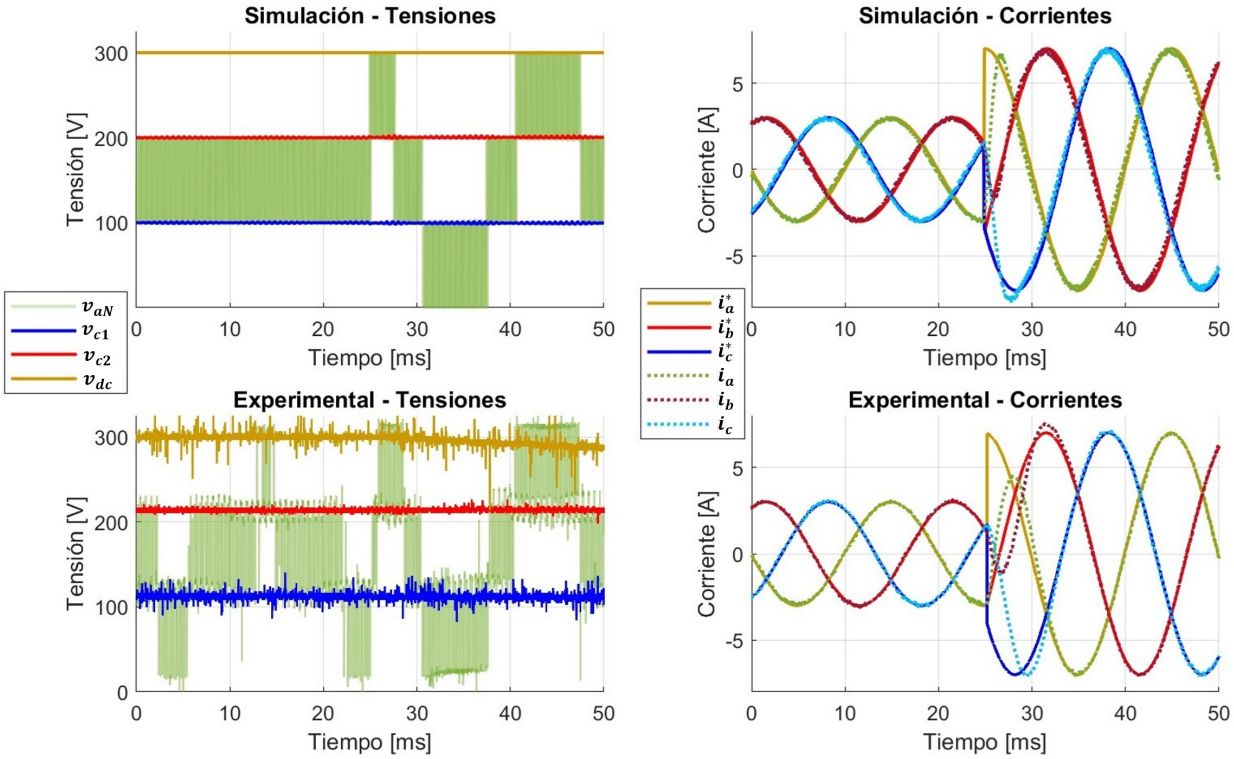



Figura 5.35: Comparación de corrientes y tensiones de resultados simulados y experimentales para el control PI.

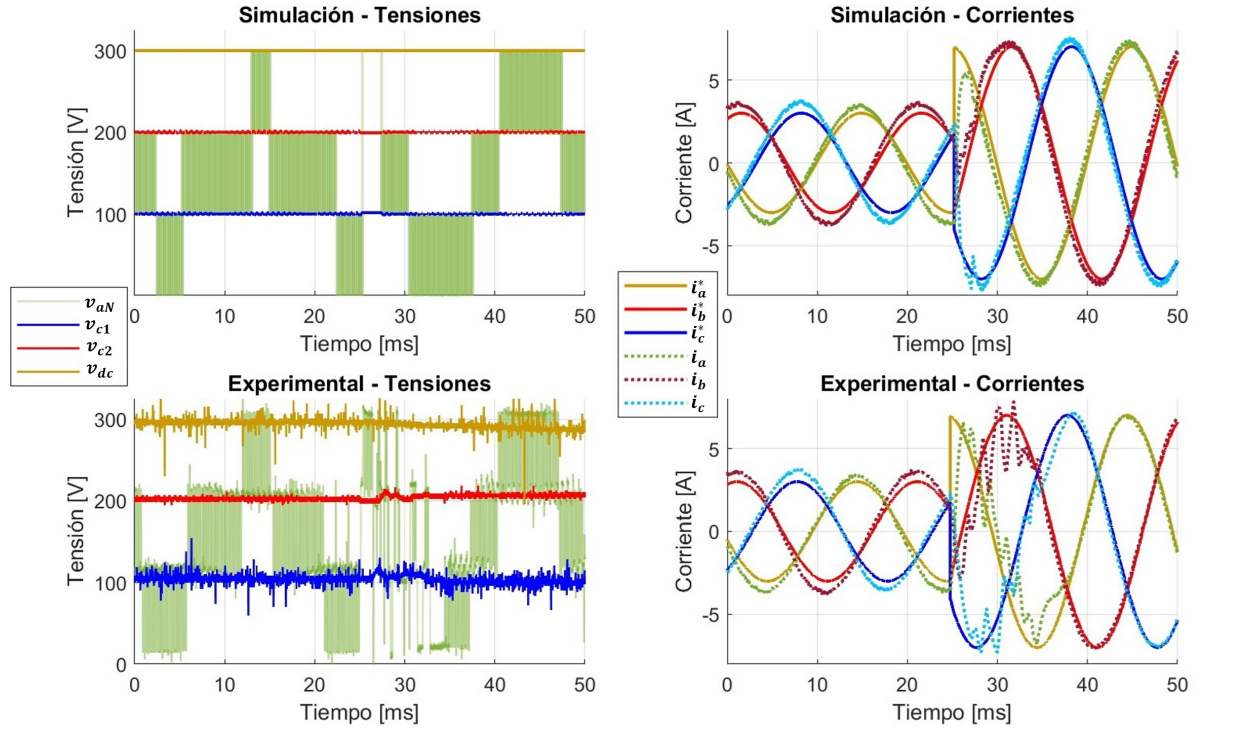



Figura 5.36: Comparación de corrientes y tensiones de resultados simulados y experimentales para el control con unión de ANN mediante histéresis.

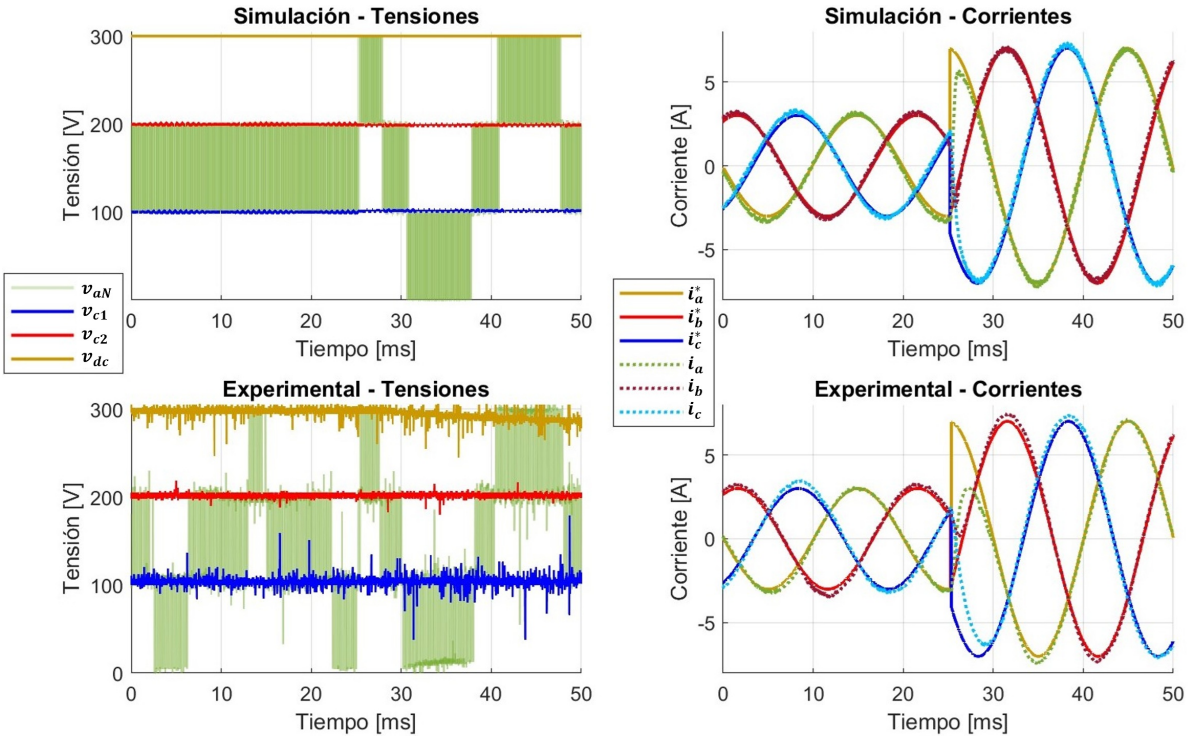



Figura 5.37: Comparación de corrientes y tensiones de resultados simulados y experimentales para el control con ANN Dual.

Para el estado estacionario, se comparan los valores de THD para las corrientes $i_a$ y tensiones $v_{aN}$ en los casos de simulación y experimentación, como se resume en la tabla 5.17.

El THD muestra diferencias notorias entre los casos simulados y en laboratorio, mostrándose que, como era de esperar, para las corrientes experimentales este aumentó para todos los controles. En contraste, para la tensión $v_{aN}$ experimental se mostraron leves disminuciones del THD en los controles con redes neuronales.

El control PI destacó por mantener los menores valores de THD en ambos escenarios. Por su parte, los controles basados en ANN lograron mantener valores de THD de corriente inferiores al 5 % tanto en simulación como en experimentación, lo cual se considera dentro de un rango aceptable.

En la Tabla 5.18 se comparan los valores de amplitud de error en estado estacionario para las corrientes $i_a$ en los casos de simulación y experimentación. El menor error se obtuvo con el control ANN Lineal. Por otro lado, en ambos casos, el control ANN con Histéresis presentó el mayor error, especialmente para la referencia de -3[A]. Esto se atribuye a la limitación de la ANN Lineal, que se activa en estado estacionario.

Tabla 5.17: THD para distintos tipos de control en simulación y experimentación.

| | **Control** | | |
|---|---|---|---|
| **Variables** | Lineal | ANN con Histéresis | ANN Dual |
| **Simulación** | | | |
| ***ia* (-3[A])** | 0.78 % | 1.24 % | 0.96 % |
| ***ia* (7[A])** | 0.77 % | 1.37 % | 1.72 % |
| ***vaN* (-3[A])** | 0.87 % | 19.14 % | 9.50 % |
| ***vaN* (7[A])** | 0.80 % | 11.52 % | 11.47 % |
| **Experimentación** | | | |
| ***ia* (-3[A])** | 2.29 % | 3.10 % | 3.78 % |
| ***ia* (7[A])** | 1.61 % | 2.14 % | 4.83 % |
| ***vaN* (-3[A])** | 19.40 % | 14.04 % | 8.92 % |
| ***vaN* (7[A])** | 14.11 % | 4.88 % | 3.67 % |

Tabla 5.18: Error estacionario de la corriente $i_a$ para distintos tipos de control en simulación y experimentación.

| | **Control** | | |
|---|---|---|---|
| **Variables** | Lineal | ANN Dual | ANN con Histéresis |
| **Simulación** | | | |
| $i_a$ **(-3[A])** | 0.36[A] | 0.43[A] | 0.84[A] |
| $i_a$ **(7[A])** | 0.84[A] | 0.58[A] | 0.95[A] |
| **Experimentación** | | | |
| $i_a$ **(-3[A])** | 0.16[A] | 0.54[A] | 0.98[A] |
| $i_a$ **(7[A])** | 0.30[A] | 0.65[A] | 0.66[A] |

A continuación en las figuras 5.38, 5.39 y 5.40 se muestran los espectros de corriente $i_a$ y tensión de salida $v_{aN}$ para las amplitudes de corriente -3[A] y 7 [A], para los tres controles en simulación y experimental, en el rango de 0-10.000 [Hz].

En la figura 5.38, se observa que para el control PI no se produjo un cambio significativo en los armónicos de las corrientes, manteniéndose un espectro limpio y con una predominancia clara de la frecuencia fundamental. Sin embargo, en las tensiones, tanto para la amplitud de -3[A] como para la de 7[A], se evidenció en los datos experimentales la aparición de armónicos adicionales en comparación con la simulación, y también una baja en la amplitud de la armónica fundamental.

Por otro lado, en la figura 5.39, el control ANN con histéresis muestra un comportamiento similar en las corrientes, con una ligera disminución en la amplitud de la frecuencia fundamental para el caso experimental de 7[A]. En cuanto a las tensiones, se observa que en simulación se distribuyen más los armónicos en comparación con el caso experimental, aunque manteniendo magnitudes relativamente pequeñas.

Para el control ANN Dual en la figura 5.40 se tiene que, en comparación con los otros

controles, se visualizan armónicas de corrientes experimentales para frecuencias pequeñas, sobre todo para el caso de 7[A], lo que no es ideal. En el caso de las tensiones, estas presentan comportamientos similares a los controles anteriores.

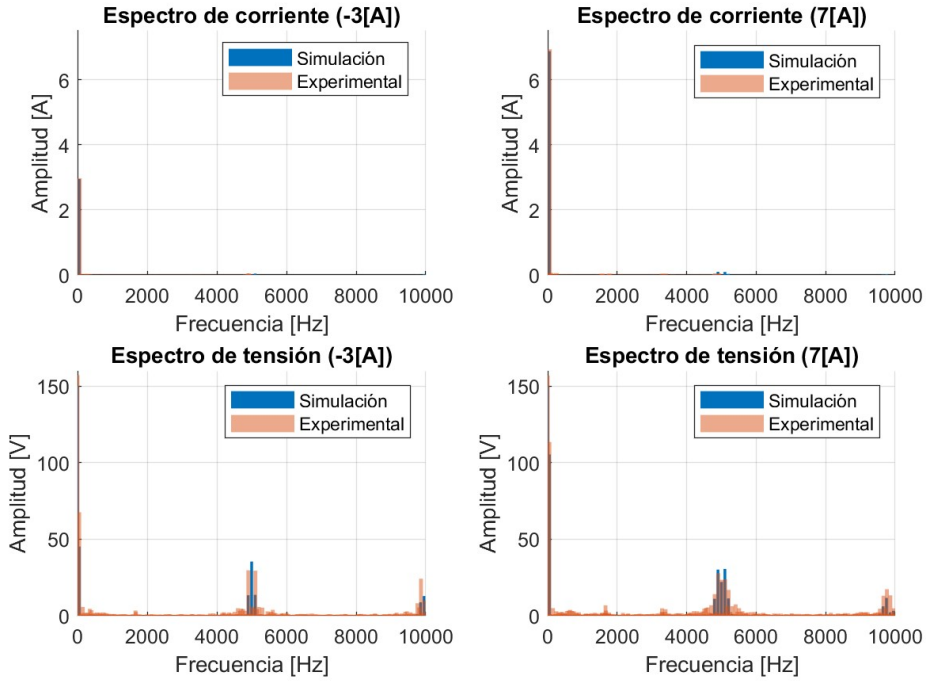



Figura 5.38: Espectro de corriente $i_a$ y tensión de salida $v_{aN}$ con -3[A] y 7 [A] para control PI en simulación y experimental.

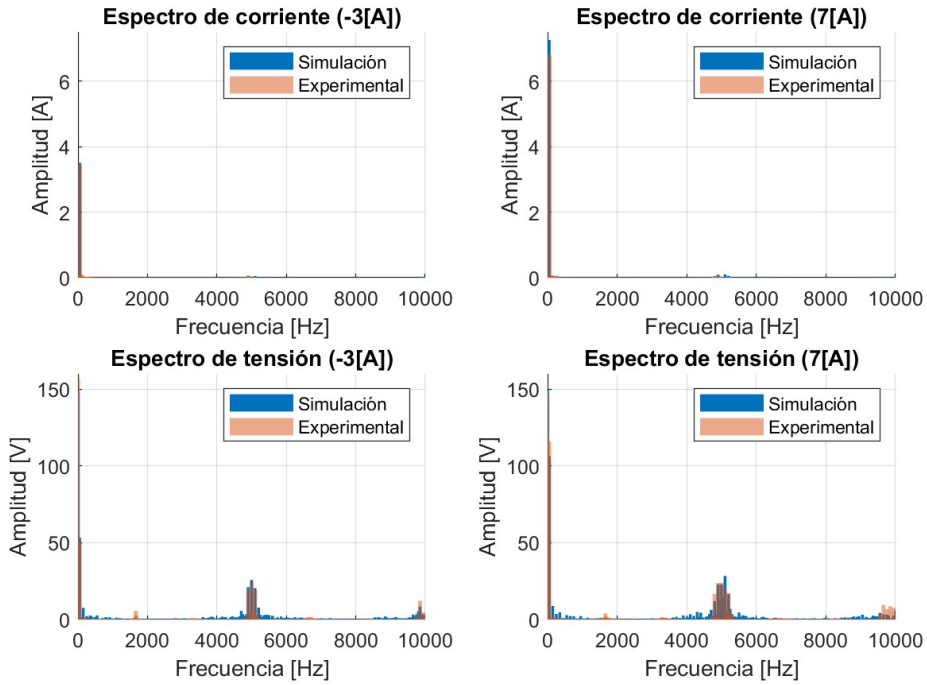



Figura 5.39: Espectro de corriente $i_a$ y tensión de salida $v_{aN}$ con -3[A] y 7 [A] para control de unión de ANN con histéresis en simulación y experimental.

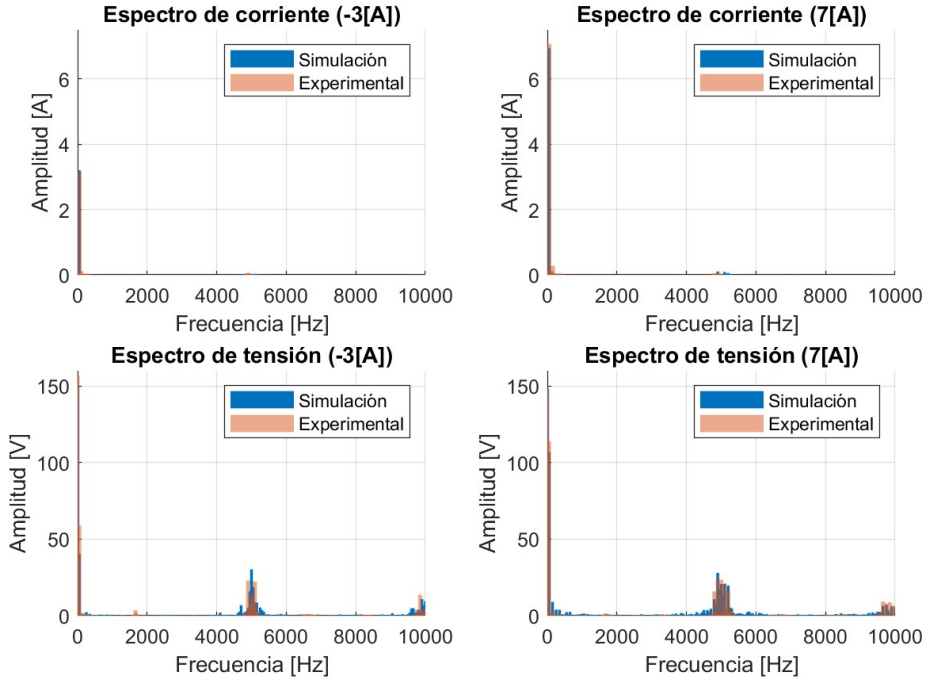



Figura 5.40: Espectro de corriente $i_a$ y tensión de salida $v_{aN}$ con -3[A] y 7 [A] para control ANN Dual en simulación y experimental.

Las diferencias entre los casos de simulación y experimentales resaltan la influencia de las condiciones reales del sistema, como el ruido y las no linealidades, que generan una mayor complejidad armónica en las variables medidas experimentalmente. A modo de análisis, en las figuras 5.41, 5.42 y 5.43 se muestra el *zoom* de las armónicas centrado en $10/6 \approx 1666.6$[Hz], correspondiente a la frecuencia base de las portadoras del PS-PWM.

Para el control PI, se observa que en la simulación no se presentan armónicas significativas en el rango analizado, salvo en torno a 1666[Hz], donde aparecen componentes de baja magnitud tanto en la tensión como en la corriente. En cambio, los resultados experimentales muestran una mayor presencia de armónicas, especialmente en la corriente. En la tensión, estas se concentran en torno a 1666[Hz], mientras que en la corriente se distribuyen principalmente cerca de 1566[Hz] y 1766[Hz].

Para el control de unión ANN con histéresis, al igual que en el control PI, el contenido espectral en torno a 1666[Hz] toma relevancia en las tensiones experimentales, siendo de mayor magnitud en el caso de -3[A] y de menor magnitud en el caso de 7[A], en comparación con el control PI. Esto puede deberse a desbalances en los condensadores del dc-link.

En el control ANN Dual las armónicas están más atenuadas en comparación con los otros dos controles, y en las tensiones se observa menos concentración en 1666[Hz], sobretodo para el caso de 7[A].

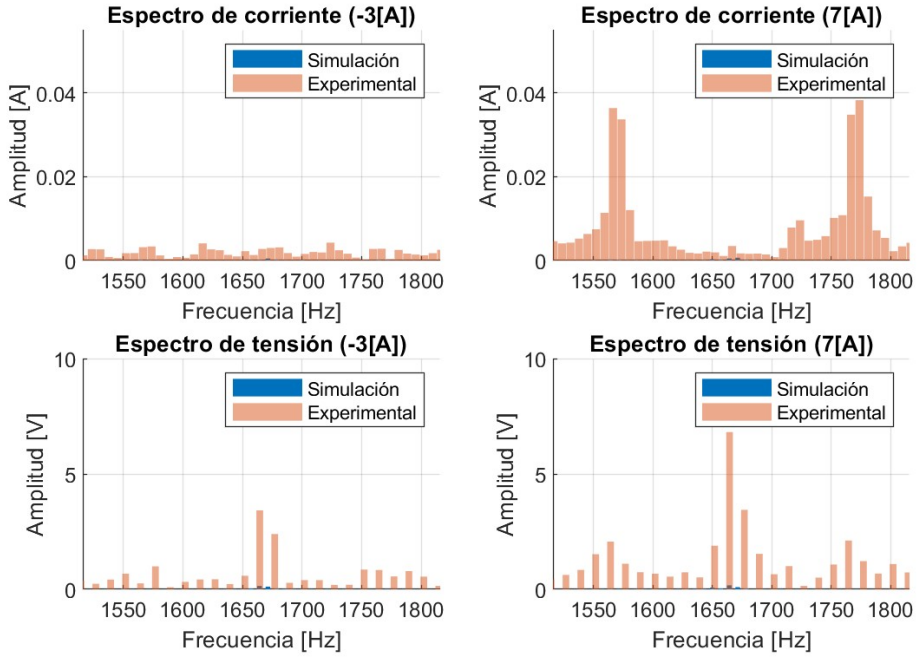



Figura 5.41: Espectro de corriente $i_a$ y tensión de salida $v_{aN}$ con -3[A] y 7 [A] para control PI en simulación y experimental, con *zoom* en 10/6[Hz].

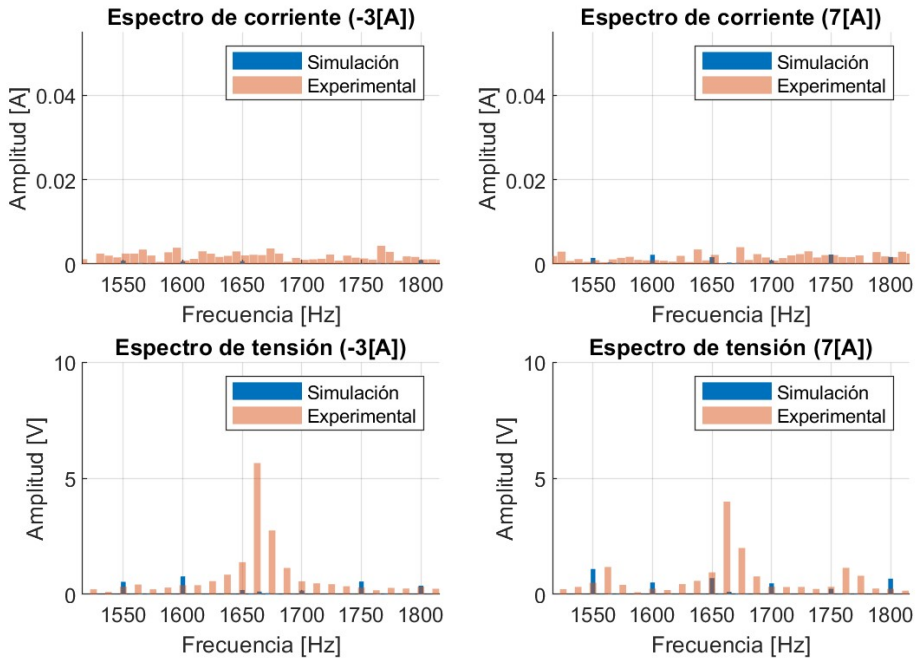



Figura 5.42: Espectro de corriente $i_a$ y tensión de salida $v_{aN}$ con -3[A] y 7 [A] para control de unión de ANN con histéresis en simulación y experimental, con *zoom* en 10/6[Hz].

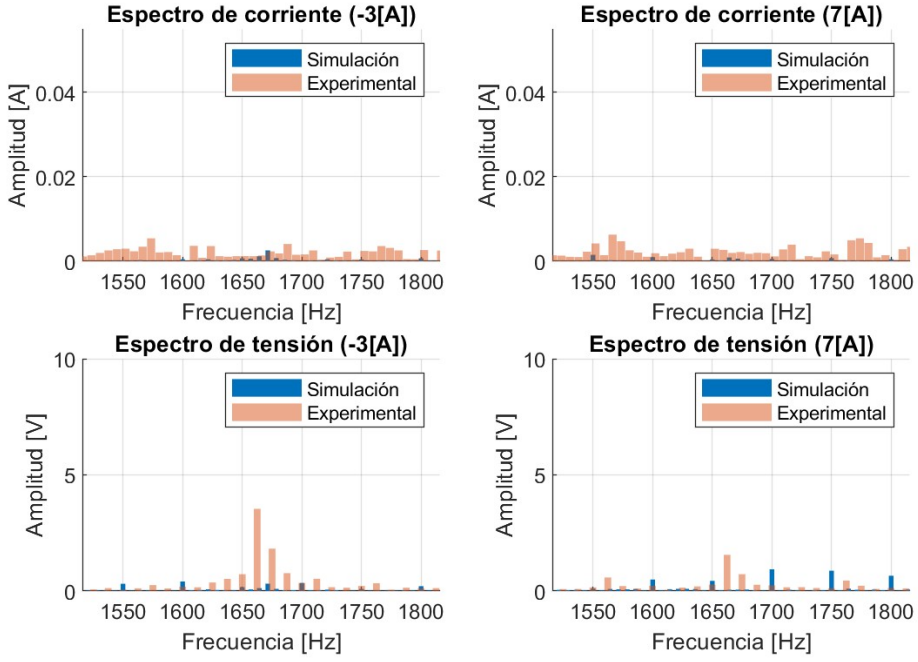



Figura 5.43: Espectro de corriente $i_a$ y tensión de salida $v_{aN}$ con -3[A] y 7 [A] para control ANN Dual en simulación y experimental, con *zoom* en 10/6[Hz].

El análisis comparativo revela que el control ANN Dual sobresale tanto en simulación como en experimentación debido a su capacidad de integrar un buen comportamiento

tanto en estados transitorios como estacionarios, con un tiempo de asentamiento reducido y un THD competitivo. Sin embargo, el control PI tiene ventajas específicas que pueden ser útiles dependiendo del escenario de aplicación.

Es importante destacar que, aunque la ANN Dual cumplió con los objetivos planteados tanto en simulación como en la aplicación experimental, existe un margen considerable para optimizar su proceso de entrenamiento. Esto podría lograrse mediante el uso de datos que representen una mayor diversidad de escenarios o a través de ajustes específicos en los parámetros de entrenamiento. Estas mejoras tienen el potencial de incrementar la adaptabilidad de la red a convertidores con diferentes parámetros, así como de optimizar la calidad de las corrientes y tensiones en la salida, garantizando un desempeño más robusto, eficiente y versátil en una amplia variedad de configuraciones del sistema.

# Capítulo 6

# Conclusiones

La presente memoria demuestra la efectividad y el potencial de las redes neuronales artificiales como una alternativa avanzada para el control de convertidores multinivel, específicamente el convertidor de capacitores flotantes trifásico de tres celdas. A lo largo de este estudio, se desarrollaron y analizaron estrategias innovadoras que integran redes neuronales en diferentes configuraciones, destacando la combinación de ANN con histéresis y el diseño de una ANN Dual. Estas propuestas fueron planteadas como una solución a las limitaciones observadas en los controles tradicionales, como el Proporcional-Integrativo (PI) y el Predictivo Basado en Modelo (MPC), en contextos que exigen alta precisión, adaptabilidad y robustez.

La estrategia de unión de ANN con histéresis fue una extensión de un trabajo previo, en el cual se desarrollaron redes neuronales específicas para ejecutar las funciones de control PI y FCS-MPC de manera independiente. En este trabajo, estas redes fueron utilizadas como base para implementar un esquema de control híbrido mediante bandas de histéresis, que permitió alternar entre los modos de control en función de las condiciones del sistema. La ANN para el control PI, con una estructura de tipo *backpropagation* y 12 neuronas en su capa oculta, fue entrenada para calcular los índices de modulación a partir de las corrientes de referencia, medidas y sus errores. Es importante mencionar que esta red presenta la limitación de no lograr incorporar la acción integral del control PI, lo que conlleva a un error en estado estacionario.

Por su parte, la ANN para el control FCS-MPC, también de tipo *backpropagation*, incluyó 36 neuronas en su capa oculta y fue diseñada para clasificar los estados de conmutación del convertidor en función de las corrientes y tensiones de referencia y medidas. Aunque este enfoque demostró ser efectivo en simulaciones, en el entorno experimental presentó limitaciones relacionadas con la sensibilidad a las condiciones operativas y la transición entre modos.

La ANN Dual, en contraste, representó un desarrollo completamente nuevo que integró ambas estrategias de control en una única red neuronal. Esta red conjunta fue diseñada para aprender de manera simultánea las dinámicas del control PI y FCS-MPC, incluyendo las bandas de histéresis, lo que permitió una transición fluida entre los estados estacionario y transitorio. Su diseño, basado en la ANN para el control lineal, también constó de una estructura *backpropagation*, combinó múltiples entradas representativas del sistema, logrando una mayor capacidad de generalización y una reducción significativa del costo

computacional respecto a el control de ANN con histéresis.

Los resultados mostraron que la ANN Dual no solo superó a la estrategia de unión de ANN con histéresis en simulaciones, sino que también mantuvo un desempeño robusto en el entorno experimental, destacándose por su eficiencia en la reducción del THD, el error en estado estacionario y la calidad de las señales de salida.

Adicionalmente, los resultados demuestran que las ANN son una herramienta efectiva para abordar problemáticas complejas en el control de convertidores multinivel, destacándose por ofrecer ventajas significativas frente a los enfoques tradicionales, especialmente en sistemas de mayor escala, donde las alternativas convencionales suelen implicar una carga computacional considerablemente más alta.

Sería interesante explorar el desarrollo de una red neuronal dual que, además de controlar las corrientes del sistema, incorpore la regulación de las tensiones de los capacitores flotantes. Este enfoque permitiría que el control sea efectivo incluso cuando las tensiones del dc-link difieran de los 300[V] utilizados en este trabajo, ampliando su aplicabilidad a una mayor variedad de configuraciones operativas. Asimismo, una extensión de la red para adaptarse a convertidores de capacitores flotantes con un número variable de celdas proporcionaría un control más flexible y escalable, adecuado para sistemas de mayor complejidad. Esto abriría nuevas posibilidades para implementar estrategias avanzadas de control en aplicaciones que requieran mayor versatilidad y robustez.

Finalmente, este trabajo subraya la importancia de continuar optimizando el entrenamiento de las redes neuronales, especialmente para mejorar su adaptabilidad a convertidores con parámetros variables y perfeccionar la calidad de las señales de salida. Estos avances permitirán maximizar el potencial de las ANN en aplicaciones avanzadas de la electrónica de potencia, consolidándolas como una herramienta clave en el diseño de sistemas más eficientes, robustos y sostenibles.

# Bibliografía


[1] Sadeq Ali Qasem Mohammed y Jin-Woo Jung. «A State-of-the-Art Review on Soft-Switching Techniques for DC–DC, DC–AC, AC–DC, and AC–AC Power Converters». En: *IEEE Transactions on Industrial Informatics* 17.10 (2021), págs. 6569-6582. DOI: 10.1109/TII.2021.3058218.

[2] Daming Wang et al. «Model Predictive Control Using Artificial Neural Network for Power Converters». En: *IEEE Transactions on Industrial Electronics* 69.4 (2022), págs. 3689-3699. DOI: 10.1109/TIE.2021.3076721.

[3] Pablo Lezana, Margarita Norambuena y Ricardo P. Aguilera. «Dual-Stage Control Strategy for a Flying Capacitor Converter Based on Model Predictive and Linear Controllers». En: *IEEE Transactions on Industrial Informatics* 18.4 (2022), págs. 2203-2212. DOI: 10.1109/TII.2021.3096947.

[4] Daming Wang et al. «Model Predictive Control Using Artificial Neural Network for Power Converters». En: *IEEE Transactions on Industrial Electronics* 69.4 (2022), págs. 3689-3699. DOI: 10.1109/TIE.2021.3076721.

[5] Margarita Norambuena. *Control Dual aplicado a un Convertidor Multinivel Flying Capacitor*. 2013.

[6] José Rodriguez et al. «Multilevel Voltage-Source-Converter Topologies for Industrial Medium-Voltage Drives». En: *IEEE Transactions on Industrial Electronics* 54.6 (2007), págs. 2930-2945. DOI: 10.1109/TIE.2007.907044.

[7] Cristobal Cortés. *Aplicación de red neuronal artificial al control de un convertidor trifásico de condensadores flotantes*. 2024.

[8] Margarita Norambuena, Sofía Dawson y Pablo Lezana. *Dual-Model Artificial Neural Network based control for a 4-level Flying Capacitor Converter*. 2024.

[9] Richardt H. Wilkinson, Thierry A. Meynard y Hendrik du Toit Mouton. «Natural Balance of Multicell Converters: The General Case». En: *IEEE Transactions on Power Electronics* 21.6 (2006), págs. 1658-1666. DOI: 10.1109/TPEL.2006.882951.

[10] Margarita Norambuena et al. «Finite Control Set Model Predictive Control reduced computational cost applied to a Flying Capacitor converter». En: *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*. 2017, págs. 4903-4907. DOI: 10.1109/IECON.2017.8216846.

[11] B.K. Bose. «Artificial neural network applications in power electronics». En: *IECON'01. 27th Annual Conference of the IEEE Industrial Electronics Society (Cat. No.37243)*. Vol. 3. 2001, 1631-1638 vol.3. DOI: 10.1109/IECON.2001.975533.

[12] Simon Haykin. *Neural Networks and Learning Machines, 3rd ed.* 2009.

[13] Ian Goodfellow, Yoshua Bengio y Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[14] T. Kamio, S. Tanaka y M. Morisue. «Backpropagation algorithm for logic oriented neural networks». En: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*. Vol. 2. 2000, 123-128 vol.2. DOI: 10.1109/IJCNN.2000.857885.

[15] Cristobal Cortés. *Control en base a RNA al control de convertidor flying capacitor de cuatro niveles con control MPC*. 2023.

[16] Pablo Lezana. *Manejo de la Plataforma Digital BRAIn para el control de Accionamientos*. 2022.

# Apéndice A: Corrientes en el transitorio

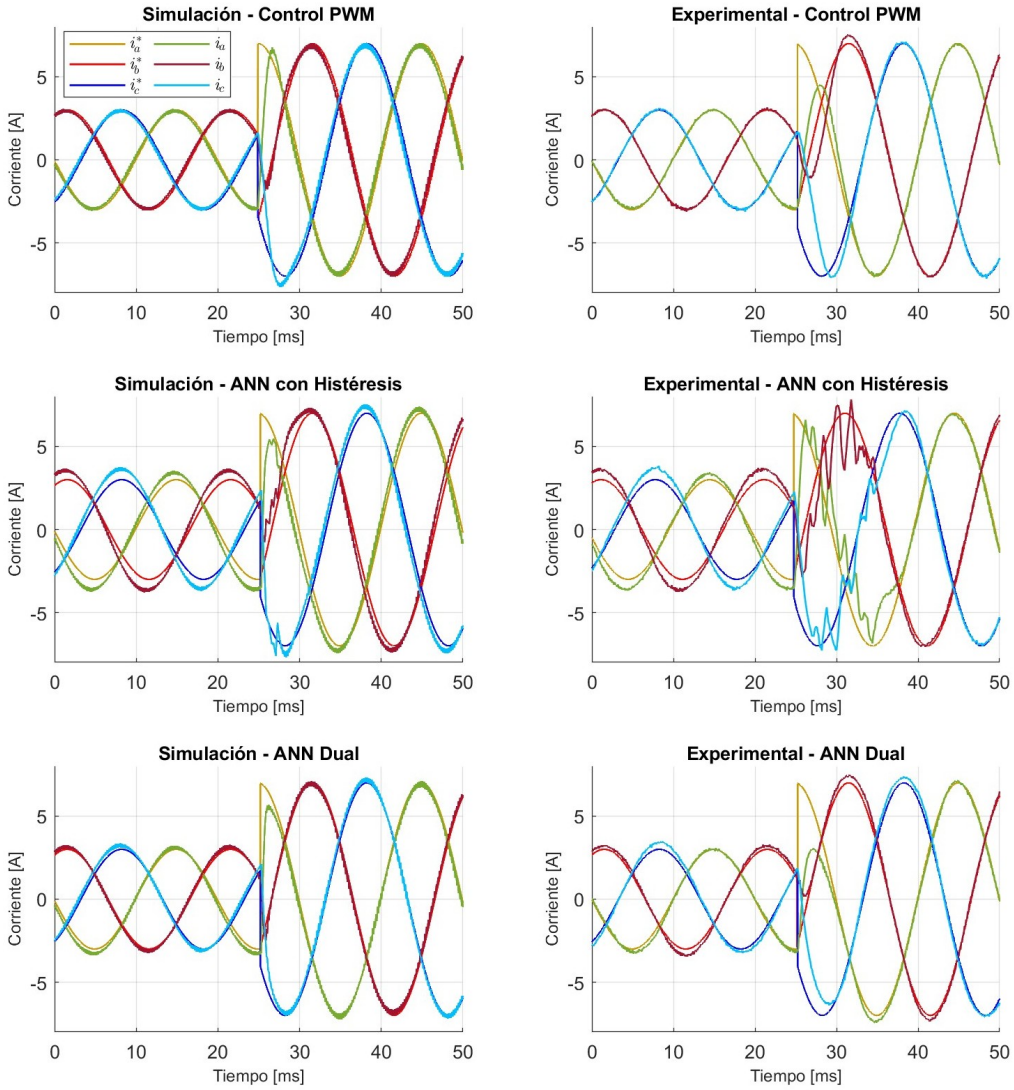



Figura 1: Gráficas de corriente en el transitorio para los controles en simulación y experimentales medidas con la BRAIn (trazos continuos).

Como complemento, a continuación se muestran las gráficas de corrientes experimentales medidas con el osciloscopio.

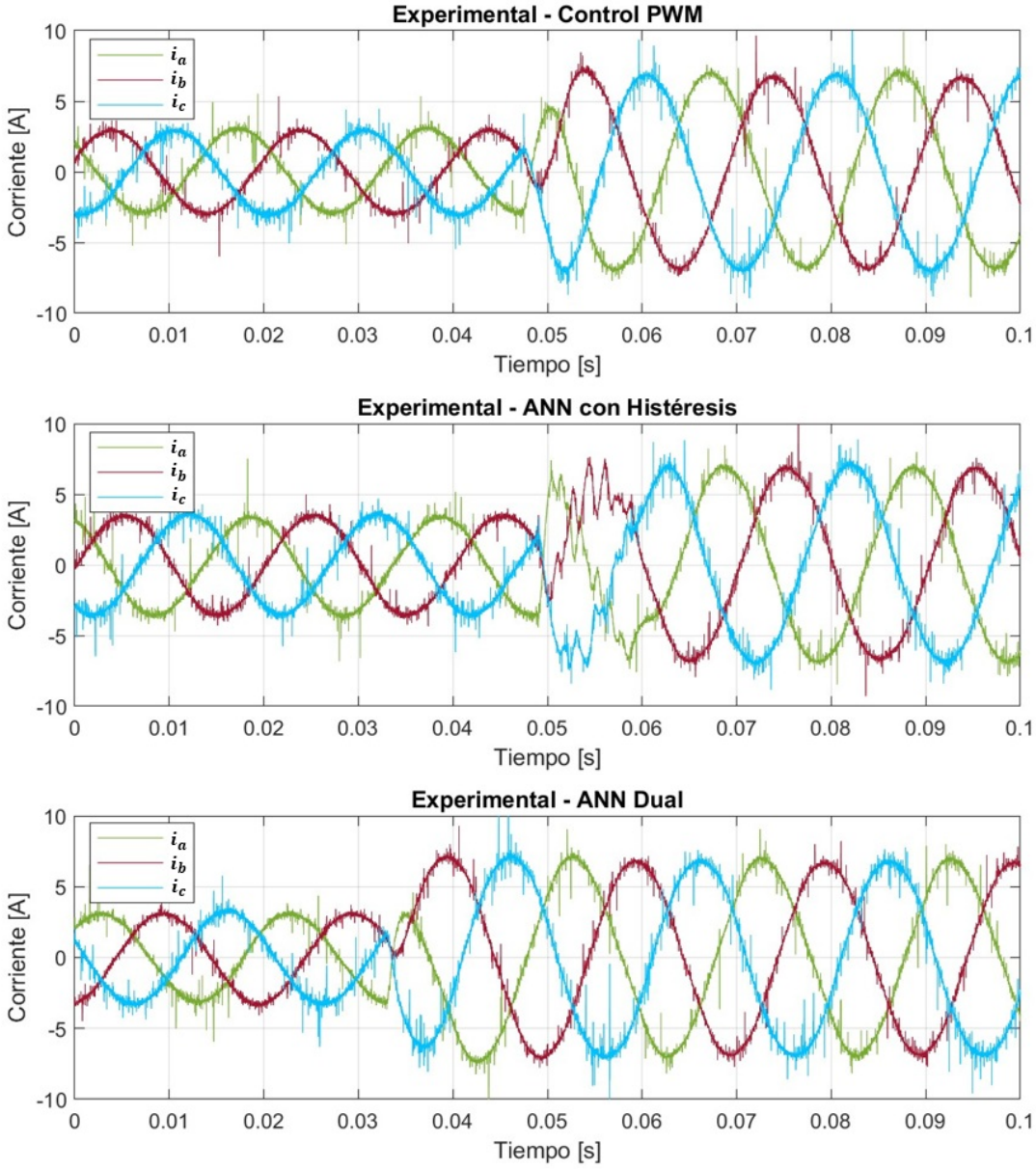



Figura 2: Gráficas de corriente en el transitorio medidas con el osciloscopio.

# Apéndice B: Códigos de simulación

## .1. Control Dual con Histéresis

```
// -------- def. variables Parte MPC -----------------
// 1. referencia de corriente
float t1=0;
float i1_r = 0;
float i2_r = 0;
float i3_r = 0;
ia_ref1 = 0;
ib_ref1 = 0;
ic_ref1 = 0;
// 2. Cálculo de Teta
float alfa = 0;
float beta = 0;
float teta = 0;
float tetaf = 0;
// 3. Transformada ABC-DQ de corriente
float Da=0;
float Db=0;
float Dc=0;
float Qa=0;
float Qb=0;
float Qc=0;
// Actualización variables k+1 *
float VoN=0;
float ia_t=0;
float ib_t=0;
float ic_t=0;
float vc1a_t=0;
float vc2a_t=0;
float vc1b_t=0;
float vc2b_t=0;
float vc1c_t=0;
float vc2c_t=0;
float vc1a_t2=0;
float vc2a_t2=0;
float vc1b_t2=0;
float vc2b_t2=0;
float vc1c_t2=0;
float vc2c_t2=0;
// Control MPC k+2
```

```
float ia_t2=0;
float ib_t2=0;
float ic_t2=0;
int   i=0;
float g_min=0;
float g=0;
int S1a=0;
int S2a=0;
int S3a=0;
int S1b=0;
int S2b=0;
int S3b=0;
int S1c=0;
int S2c=0;
int S3c=0;
float VaN_t=0;
float VbN_t=0;
float VcN_t=0;
int act=0;
float ia_tt2=0;
float ib_tt2=0;
float ic_tt2=0;
float vc1a_tt2=0;
float vc2a_tt2=0;
float vc1b_tt2=0;
float vc2b_tt2=0;
float vc1c_tt2=0;
float vc2c_tt2=0;

//------def. variables de control LINEAL-----------------------
// 1. referencia de corriente
float tt1=0;
float hs=Ts;
float ia_ref1p = 0;
float ib_ref1p = 0;
float ic_ref1p = 0;
// 4. Corrientes en eje D y eje Q
float id_refp=0;
float id_medp=0;
float iq_refp=0;
float iq_medp=0;
// 5. Cálculo eje D
float errorD=0;
float xd_t1=0;
float ud_t1=0;
// 6. Cálculo eje Q
float errorQ=0;
float xq_t1=0;
float uq_t1=0;
// 7. Transformada DQ-ABC de tensión
float DET=0;
float DET_1=0;
float Va=0;
float Vb=0;
```

```
float Vc=0;

//------def. variables HISTERESIS ------------------
float Eia=0;
float Eib=0;
float Eic=0;
float EVac1=0;
float EVac2=0;
float EVbc1=0;
float EVbc2=0;
float EVcc1=0;
float EVcc2=0;
float Jc=0;
float Csup=0;
float Cinf=0;
float banda_n=0;
float cambio=0;
float banda=0;
float baja=0;
float sube=0;
float MPC=0;
// probando cosas,,
float Va_pp=0;
float Vb_pp=0;
float Vc_pp=0;
float Vdc=0;
float Vdc_3=0;
float Vdc_2=0;
//------
float s1_Ampc=0;
float s2_Ampc=0;
float s3_Ampc=0;
float s1_Bmpc=0;
float s2_Bmpc=0;
float s3_Bmpc=0;
float s1_Cmpc=0;
float s2_Cmpc=0;
float s3_Cmpc=0;

/************************************/
Vdc=Amp_vdc;
Vdc_3=Vdc*0.33333333333333;
Vdc_2=Vdc*0.5;
vc1_r=Vdc_3;
vc2_r=2*Vdc_3;

// *************PARTE HISTERESIS**************
// Variables necesarias
t=t+hs;

ia_ref=Amp_ref*cos(2*pi*Fc_ref*(t));
ib_ref=Amp_ref*cos(2*pi*Fc_ref*(t)-x2_3*pi);
ic_ref=Amp_ref*cos(2*pi*Fc_ref*(t)+x2_3*pi);
```

```
Eia=ia_ref-ia_med;
Eib=ib_ref-ib_med;
Eic=ic_ref-ic_med;

EVac1=Vdc_3-vc1a_m;
EVac2=(2*Vdc_3-vc2a_m)*0.5;
EVbc1=Vdc_3-vc1b_m;
EVbc2=(2*Vdc_3-vc2b_m)*0.5;
EVcc1=Vdc_3-vc1c_m;
EVcc2=(2*Vdc_3-vc2c_m)*0.5;

C_MPCant=C_MPC;

// Calcular la función de costo
Jc = CP_IL * (Eia * Eia + Eib * Eib + Eic * Eic) + CP_VC * (EVac1 * EVac1 +
↪  EVac2 * EVac2 + EVbc1 * EVbc1 + EVbc2 * EVbc2 + EVcc1 * EVcc1 + EVcc2 *
↪  EVcc2);

if (Jc >= CD_Esup) {
    Csup = 1; //pasa banda superior
    }
else {
    Csup = 0; //no pasa banda superior
    }

if (Jc <= CD_Einf) {
         Cinf = 1; //pasa banda inferior
    }
else {
    Cinf = 0;  //no pasa banda inferior
    }

banda_n=Csup-Cinf;
// banda_n = 1 cuando pasa banda sup
// banda_n = -1 cuando pasa banda inf
// banda_n = 0 cuando no pasa bandas

cambio=(banda_n-banda_a);
// cambio = 0 cuando banda_n == banda_a (si el estado de antes era igual que
↪  ahora no hay cambio)
// cambio =/= 0 cuando banda_n =/= banda_a


banda=fabs(banda_n)*-1+1;
// banda = 0 cuando pasa por alguna de las bandas
// banda = 1 cuando no pasa por las bandas


if (cambio < 0 || cambio > 0) {  // si el cambio es distinto de cero
        auxSB = cambio;
    }

//esta bajando cuando:
if ((banda && auxSB < 0)) {  // si pasa por alguna de las bandas y cambió
```

```
        baja = 1;
        sube = 0;
}
else {
        if ((banda && auxSB > 0)) {
                        sube = 1;
         baja = 0;
        }
      }
// situación en que está en modo MPC:
if ((Jc >= CD_Einf && baja) || (Jc >= CD_Esup)) {
        C_MPC = 1;
        }
else {
        C_MPC = 0;
    }

banda_a=banda_n; // banda_a es un discstate

// variables para analizar comportamiento
J1=banda_n;
J2=cambio;
J3=banda;
J4=sube;
J5=baja;

//-------- N U E V O--------------------------------
if ( (C_MPCant-C_MPC)==1 )  //cambia de 1 a 0 --> MPC a PI Y se activa la
↪  versión extendida y opera mpc en paralelo con PI
{
Ext=1;
PIa=0;
PIb=0;
PIc=0;
}

if ( (PIa==1) && (PIb==1) && (PIc==1) )  // los 3 estan habilitados para usar
↪  pi
{
Ext=0;
}

//-----------------------------------------------------------
/// *************PARTE MPC**************
if ( C_MPC == 1 || Ext==1 )
{

// 1. Referencia corriente
t1=t+2*hs;

i1_r=cos(2*pi*Fc_ref*(t));
i2_r=cos(2*pi*Fc_ref*(t)-x2_3*pi);
i3_r=cos(2*pi*Fc_ref*(t)+x2_3*pi);
// con t1
```

```
ia_ref1=Amp_ref*cos(2*pi*Fc_ref*t1);
ib_ref1=Amp_ref*cos(2*pi*Fc_ref*t1-x2_3*pi);
ic_ref1=Amp_ref*cos(2*pi*Fc_ref*t1+x2_3*pi);
// con t
ia_ref=Amp_ref*cos(2*pi*Fc_ref*(t));
ib_ref=Amp_ref*cos(2*pi*Fc_ref*(t)-x2_3*pi);
ic_ref=Amp_ref*cos(2*pi*Fc_ref*(t)+x2_3*pi);

//*****************************************/
// 2. Cálculo de Teta
alfa = i1_r;
beta = (i2_r-i3_r) * sqrt3_1;
teta = fabs(atan(beta/alfa));

if (alfa<0 && beta>=0)
    tetaf = pi - teta;
else
         {
    if (alfa<0 && beta<0)
        tetaf = pi + teta;
    else
                {
        if (alfa>=0 && beta<0)
            tetaf = 2 * pi-teta;
        else
            if (alfa>=0 && beta>=0)
                tetaf = teta;
                }
          }
/*****************************************/
// 3. Transformada ABC-DQ de corriente
Da = cos(tetaf) * x2_3;
Db = cos(tetaf-g120) * x2_3;
Dc = cos(tetaf+g120) * x2_3;

Qa = -sin(tetaf) * x2_3;
Qb = -sin(tetaf-g120) * x2_3;
Qc = -sin(tetaf+g120) * x2_3;

/*****************************************/
// 4. Corrientes en eje D y eje Q
id_med = Da*ia_med + Db*ib_med + Dc*ic_med;
iq_med = Qa*ia_med + Qb*ib_med + Qc*ic_med;

id_ref = Da*ia_ref1 + Db*ib_ref1 + Dc*ic_ref1;
iq_ref = Qa*ia_ref1 + Qb*ib_ref1 + Qc*ic_ref1;

// Actualización variables en el t=t+1 (fórmulas)
//VoN=(VaN+VbN+VcN)/3;
VoN=150;

ia_t=(VaN-VoN)*k1+ia_med*k2;
ib_t=(VbN-VoN)*k1+ib_med*k2;
ic_t=(VcN-VoN)*k1+ic_med*k2;
```

```
vc1a_t=(ia_t+ia_med)*kC1*(s2_A-s1_A)+vc1a_m;  // kC1, kC2 ctes h/C
vc2a_t=(ia_t+ia_med)*kC2*(s3_A-s2_A)+vc2a_m;

vc1b_t=(ib_t+ib_med)*kC1*(s2_B-s1_B)+vc1b_m;
vc2b_t=(ib_t+ib_med)*kC2*(s3_B-s2_B)+vc2b_m;

vc1c_t=(ic_t+ic_med)*kC1*(s2_C-s1_C)+vc1c_m;
vc2c_t=(ic_t+ic_med)*kC2*(s3_C-s2_C)+vc2c_m;

/*****************************************/
// Predicción en el t=t+2 -->
g_min=1e20;

for (i=0;i<=contX-1;i=i+1) //ojo contX es 512=8x8x8 (archivo mat_1.h)
{
   //Fórmula típica de Vxn evaluando todos los estados posibles (i)
        VaN_t=Vdc*S3A[i]+vc2a_t*(S2A[i]-S3A[i])+vc1a_t*(S1A[i]-S2A[i]);
        VbN_t=Vdc*S3B[i]+vc2b_t*(S2B[i]-S3B[i])+vc1b_t*(S1B[i]-S2B[i]);
        VcN_t=Vdc*S3C[i]+vc2c_t*(S2C[i]-S3C[i])+vc1c_t*(S1C[i]-S2C[i]);

        VoN=(VaN_t+VbN_t+VcN_t)/3;
        //VoN=150;
        ia_t2=(VaN_t-VoN)*k1+ia_t*k2; // misma fórmula pero con ia en t=t+1
        ib_t2=(VbN_t-VoN)*k1+ib_t*k2;
        ic_t2=(VcN_t-VoN)*k1+ic_t*k2;

        // misma fórmula pero en t=t+2
        vc1a_t2=(ia_t2+ia_t)*kC1*(S2A[i]-S1A[i])+vc1a_t;
        vc2a_t2=(ia_t2+ia_t)*kC2*(S3A[i]-S2A[i])+vc2a_t;

        vc1b_t2=(ib_t2+ib_t)*kC1*(S2B[i]-S1B[i])+vc1b_t;
        vc2b_t2=(ib_t2+ib_t)*kC2*(S3B[i]-S2B[i])+vc2b_t;

        vc1c_t2=(ic_t2+ic_t)*kC1*(S2C[i]-S1C[i])+vc1c_t;
        vc2c_t2=(ic_t2+ic_t)*kC2*(S3C[i]-S2C[i])+vc2c_t;

// Se utiliza ( )2 porque es más suave que el | |
        g=((ia_ref1-ia_t2)*(ia_ref1-ia_t2))+w1*((vc1_r-vc1a_t2)*(vc1_r-vc1a_t2))+w2*((vc2_r-vc2a_t2

        if (g<g_min)
        {
                g_min=g;    //quedan datos del con menor error
                act=i;      // actuación
                ia_tt2=ia_t2;
                ib_tt2=ib_t2;
                ic_tt2=ic_t2;
                vc1a_tt2=vc1a_t2;
                vc2a_tt2=vc2a_t2;
                vc1b_tt2=vc1b_t2;
                vc2b_tt2=vc2b_t2;
                vc1c_tt2=vc1c_t2;
                vc2c_tt2=vc2c_t2;
        }
```

```
}
S1a=S1A[act];
S2a=S2A[act];
S3a=S3A[act];
S1b=S1B[act];
S2b=S2B[act];
S3b=S3B[act];
S1c=S1C[act];
S2c=S2C[act];
S3c=S3C[act];
// Valor final tensión de carga
VaN=Vdc*S3a+vc2a_t*(S2a-S3a)+vc1a_t*(S1a-S2a);
VbN=Vdc*S3b+vc2b_t*(S2b-S3b)+vc1b_t*(S1b-S2b);
VcN=Vdc*S3c+vc2c_t*(S2c-S3c)+vc1c_t*(S1c-S2c);

MPC=1;
/*****************************************/
// Salida de variables MPC
s1_Ampc=S1a;
s2_Ampc=S2a;
s3_Ampc=S3a;
s1_Bmpc=S1b;
s2_Bmpc=S2b;
s3_Bmpc=S3b;
s1_Cmpc=S1c;
s2_Cmpc=S2c;
s3_Cmpc=S3c;

s1_A=s1_Ampc;
s2_A=s2_Ampc;
s3_A=s3_Ampc;
s1_B=s1_Bmpc;
s2_B=s2_Bmpc;
s3_B=s3_Bmpc;
s1_C=s1_Cmpc;
s2_C=s2_Cmpc;
s3_C=s3_Cmpc;

dyia_t=ia_tt2;
dyib_t=ib_tt2;
dyic_t=ic_tt2;
dyvc1a_t = vc1a_tt2;
dyvc2a_t = vc2a_tt2;
dyvc1b_t = vc1b_tt2;
dyvc2b_t = vc2b_tt2;
dyvc1c_t = vc1c_tt2;
dyvc2c_t = vc2c_tt2;
//-------------------
// SALIDA EXTRA
s1a_1t = s1_Ax;
s2a_1t = s2_Ax;
s3a_1t = s3_Ax;
s1b_1t = s1_Bx;
s2b_1t = s2_Bx;
```

```
s3b_1t = s3_Bx;
s1c_1t = s1_Cx;
s2c_1t = s2_Cx;
s3c_1t = s3_Cx;

s1_Ax = s1_A;
s2_Ax = s2_A;
s3_Ax = s3_A;
s1_Bx = s1_B;
s2_Bx = s2_B;
s3_Bx = s3_B;
s1_Cx = s1_C;
s2_Cx = s2_C;
s3_Cx = s3_C;
}

//------------ACTUALIZACIÓN ESTADOS LINEAL---------------

if (MPC ==1 ) {
// 1. Referencia corriente
tt1=t-hs;

i1_r=cos(2*pi*Fc_ref*(t));
i2_r=cos(2*pi*Fc_ref*(t)-x2_3*pi);
i3_r=cos(2*pi*Fc_ref*(t)+x2_3*pi);

ia_ref1p=Amp_ref*cos(2*pi*Fc_ref*tt1);
ib_ref1p=Amp_ref*cos(2*pi*Fc_ref*tt1-x2_3*pi);
ic_ref1p=Amp_ref*cos(2*pi*Fc_ref*tt1+x2_3*pi);

/*****************************************/
// 2. Cálculo de Teta
alfa = i1_r;
beta = (i2_r-i3_r) * sqrt3_1;
teta = fabs(atan(beta/alfa));

if (alfa<0 && beta>=0)
    tetaf = pi - teta;
else
         {
    if (alfa<0 && beta<0)
        tetaf = pi + teta;
    else
                {
        if (alfa>=0 && beta<0)
            tetaf = 2 * pi-teta;
        else
            if (alfa>=0 && beta>=0)
                tetaf = teta;
                }
          }
/*****************************************/
// 3. Transformada ABC-DQ de corriente
Da = cos(tetaf) * x2_3;
```

```
Db = cos(tetaf-g120) * x2_3;
Dc = cos(tetaf+g120) * x2_3;

Qa = -sin(tetaf) * x2_3;
Qb = -sin(tetaf-g120) * x2_3;
Qc = -sin(tetaf+g120) * x2_3;
/*****************************************/
// 4. Corrientes en eje D y eje Q
VoN=150;//0.333333*(VaN+VbN+VcN);
ud_t = Da*(VaN-VoN) + Db*(VbN-VoN) + Dc*(VcN-VoN);
uq_t = Qa*(VaN-VoN) + Qb*(VbN-VoN) + Qc*(VcN-VoN);
/*****************************************/
id_medp = Da*ia_med + Db*ib_med + Dc*ic_med;
iq_medp = Qa*ia_med + Qb*ib_med + Qc*ic_med;

id_refp = Da*ia_ref1p + Db*ib_ref1p + Dc*ic_ref1p;
iq_refp = Qa*ia_ref1p + Qb*ib_ref1p + Qc*ic_ref1p;
/*****************************************/
// 5. Cálculo eje D
errorD = id_refp-id_medp;

xd_t1 = (ud_t * ni2) - (xd_t * di2);
ud_t1 = (errorD-xd_t1) * kpi;

if (ud_t1>=Vdc_2)
     ud_t1 = Vdc_2;
else
{
    if(ud_t1<=-Vdc_2)
        ud_t1 = -Vdc_2;
}
/*****************************************/
// 6. Cálculo eje Q
errorQ = iq_refp-iq_medp;

xq_t1 = (uq_t * ni2) - (xq_t * di2);
uq_t1 = (errorQ-xq_t1) * kpi;

if(uq_t1>=Vdc_2)
    uq_t1 = Vdc_2;
else
{
    if(uq_t1<=-Vdc_2)
        uq_t1 = -Vdc_2;
}
/*****************************************/
// 7. Transformada DQ-ABC de tensión
DET = Da * (Qb-Qc) - Db * (Qa-Qc) + Dc * (Qa-Qb);
DET_1 =1/DET;

Va_pp = (((Qb-Qc) * ud_t1 - (Db-Dc) * uq_t1) * DET_1 / Vdc);  //Va_p son lo
↪  distinto
Vb_pp = (((Qc-Qa) * ud_t1 + (Da-Dc) * uq_t1) * DET_1 / Vdc);
Vc_pp = (((Qa-Qb) * ud_t1 - (Da-Db) * uq_t1) * DET_1 / Vdc);
```

```
// variables que se guardan para PI
Va_p = Va_pp;
Vb_p = Vb_pp;
Vc_p = Vc_pp;

xd_t=xd_t1;
ud_t=ud_t1;
xq_t=xq_t1;
uq_t=uq_t1;
}
salidaMPC = C_MPC;
J = Jc;

// *************PARTE LINEAL**************
if( C_MPC == 0)
{
// 1. Referencia corriente
tt1=t-hs;

i1_r=cos(2*pi*Fc_ref*(t));
i2_r=cos(2*pi*Fc_ref*(t)-x2_3*pi);
i3_r=cos(2*pi*Fc_ref*(t)+x2_3*pi);

ia_ref1p=Amp_ref*cos(2*pi*Fc_ref*tt1);
ib_ref1p=Amp_ref*cos(2*pi*Fc_ref*tt1-x2_3*pi);
ic_ref1p=Amp_ref*cos(2*pi*Fc_ref*tt1+x2_3*pi);
/*****************************************/
// 2. Cálculo de Teta
alfa = i1_r;
beta = (i2_r-i3_r) * sqrt3_1;
teta = fabs(atan(beta/alfa));

if (alfa<0 && beta>=0)
    tetaf = pi - teta;
else
         {
    if (alfa<0 && beta<0)
        tetaf = pi + teta;
    else
                {
        if (alfa>=0 && beta<0)
            tetaf = 2 * pi-teta;
        else
            if (alfa>=0 && beta>=0)
                tetaf = teta;
                }
          }
/*****************************************/
// 3. Transformada ABC-DQ de corriente
Da = cos(tetaf) * x2_3;
Db = cos(tetaf-g120) * x2_3;
Dc = cos(tetaf+g120) * x2_3;
```

```
Qa = -sin(tetaf) * x2_3;
Qb = -sin(tetaf-g120) * x2_3;
Qc = -sin(tetaf+g120) * x2_3;

/*****************************************/
// 4. Corrientes en eje D y eje Q
id_medp = Da*ia_med + Db*ib_med + Dc*ic_med;
iq_medp = Qa*ia_med + Qb*ib_med + Qc*ic_med;

id_refp = Da*ia_ref1p + Db*ib_ref1p + Dc*ic_ref1p;
iq_refp = Qa*ia_ref1p + Qb*ib_ref1p + Qc*ic_ref1p;
/*****************************************/
// 5. Cálculo eje D
errorD = id_refp-id_medp;

xd_t1 = (ud_t * ni2) - (xd_t * di2);
ud_t1 = (errorD-xd_t1) * kpi;

if (ud_t1>=Vdc_2)
     ud_t1 = Vdc_2;
else
{
    if(ud_t1<=-Vdc_2)
        ud_t1 = -Vdc_2;
}
/*****************************************/
// 6. Cálculo eje Q
errorQ = iq_refp-iq_medp;

xq_t1 = (uq_t * ni2) - (xq_t * di2);
uq_t1 = (errorQ-xq_t1) * kpi;

if(uq_t1>=Vdc_2)
    uq_t1 = Vdc_2;
else
{
    if(uq_t1<=-Vdc_2)
        uq_t1 = -Vdc_2;
}
/*****************************************/
// 7. Transformada DQ-ABC de tensión
DET = Da * (Qb-Qc) - Db * (Qa-Qc) + Dc * (Qa-Qb);
DET_1 =1/DET;

Va = (((Qb-Qc) * ud_t1 - (Db-Dc) * uq_t1) * DET_1 / Vdc);
Vb = (((Qc-Qa) * ud_t1 + (Da-Dc) * uq_t1) * DET_1 / Vdc);
Vc = (((Qa-Qb) * ud_t1 - (Da-Db) * uq_t1) * DET_1 / Vdc);

Va_p= Va;
Vb_p= Vb;
Vc_p= Vc;

/*****************************************/
// Salida de variables PWM
```

```
xd_t=xd_t1;
ud_t=ud_t1;
xq_t=xq_t1;
uq_t=uq_t1;

id_ref=id_refp;
iq_ref=iq_refp;
id_med=id_medp;
iq_med=iq_medp;

ia_ref1 = ia_ref1p;
ib_ref1 = ib_ref1p;
ic_ref1 = ic_ref1p;

s1_A = Va_p+0.5;
s2_A = Va_p+0.5;
s3_A = Va_p+0.5;
s1_B = Vb_p+0.5;
s2_B = Vb_p+0.5;
s3_B = Vb_p+0.5;
s1_C = Vc_p+0.5;
s2_C = Vc_p+0.5;
s3_C = Vc_p+0.5;

//-------------------
// SALIDA EXTRA

s1a_1t = s1_Ax;
s2a_1t = s2_Ax;
s3a_1t = s3_Ax;
s1b_1t = s1_Bx;
s2b_1t = s2_Bx;
s3b_1t = s3_Bx;
s1c_1t = s1_Cx;
s2c_1t = s2_Cx;
s3c_1t = s3_Cx;

s1_Ax = s1_A;
s2_Ax = s2_A;
s3_Ax = s3_A;
s1_Bx = s1_B;
s2_Bx = s2_B;
s3_Bx = s3_B;
s1_Cx = s1_C;
s2_Cx = s2_C;
s3_Cx = s3_C;
}
//------ MPC -> PI cuando pasa cerca de 0 por fase: -----
if ( Ext==1)
{
if( (fabs(ia_ref1)<=0.8)&&(PIa==0) )
PIa=1;
if( (fabs(ib_ref1)<=0.8)&&(PIb==0) )
PIb=1;
```

```
if( (fabs(ic_ref1)<=0.8)&&(PIc==0) )
PIc=1;
}
//----- A
if ( PIa==0)
{
s1_A = s1_Ampc;
s2_A = s2_Ampc;
s3_A = s3_Ampc;

s1_Ax2 = s1_A;
s2_Ax2 = s2_A;
s3_Ax2 = s3_A;
}
else
{
s1_A = s1_A;
s2_A = s2_A;
s3_A = s3_A;
}
//----- B
if ( PIb==0)
{
s1_B = s1_Bmpc;
s2_B = s2_Bmpc;
s3_B = s3_Bmpc;

s1_Bx2 = s1_B;
s2_Bx2 = s2_B;
s3_Bx2 = s3_B;
}
else
{
s1_B = s1_B ;
s2_B = s2_B ;
s3_B = s3_B ;
}
//----- C
if ( PIc==0)
{
s1_C = s1_Cmpc;
s2_C = s2_Cmpc;
s3_C = s3_Cmpc;

s1_Cx2 = s1_C;
s2_Cx2 = s2_C;
s3_Cx2 = s3_C;
}
else
{
s1_C = s1_C ;
s2_C = s2_C ;
s3_C = s3_C ;
}
```

```
//--------------------
// SALIDA EXTRA
s1a_1t = s1_Ax2;
s2a_1t = s2_Ax2;
s3a_1t = s3_Ax2;
s1b_1t = s1_Bx2;
s2b_1t = s2_Bx2;
s3b_1t = s3_Bx2;
s1c_1t = s1_Cx2;
s2c_1t = s2_Cx2;
s3c_1t = s3_Cx2;

s1_Ax2 = s1_A;
s2_Ax2 = s2_A;
s3_Ax2 = s3_A;
s1_Bx2 = s1_B;
s2_Bx2 = s2_B;
s3_Bx2 = s3_B;
s1_Cx2 = s1_C;
s2_Cx2 = s2_C;
s3_Cx2 = s3_C;
```

## .2. Unión de ANN con Histéresis

```
a_ref = A_r;
//------def. variables HISTERESIS ------------------
float Eia=0;
float Eib=0;
float Eic=0;
float EVac1=0;
float EVac2=0;
float EVbc1=0;
float EVbc2=0;
float EVcc1=0;
float EVcc2=0;
float Jc=0;
float Csup=0;
float Cinf=0;
float banda_n=0;
float cambio=0;
float banda=0;
float baja=0;
float sube=0;
float C_MPC=0;
// -------- def. variables Parte MPC -----------------
ia_r1=0;
ib_r1=0;
ic_r1=0;
int   i=0;
float t1=0;
```

```
float VaN_m;
float VbN_m;
float VcN_m;
float i1_r = 0;
float i2_r = 0;
float i3_r = 0;
// Cálculo de Teta
float alfa = 0;
float beta = 0;
float teta = 0;
float tetaf = 0;
// Transformada ABC-DQ de corriente
float Da=0;
float Db=0;
float Dc=0;
float Qa=0;
float Qb=0;
float Qc=0;
// -------- def. variables Parte PWM ---------------
// referencia de corriente
float tt1=0;
float hs=Ts;
float ia_ref1p = 0;
float ib_ref1p = 0;
float ic_ref1p = 0;

// Corrientes en eje D y eje Q
float s1_Ampc=0;
float s2_Ampc=0;
float s3_Ampc=0;
float s1_Bmpc=0;
float s2_Bmpc=0;
float s3_Bmpc=0;
float s1_Cmpc=0;
float s2_Cmpc=0;
float s3_Cmpc=0;
// ************* PARTE HISTERESIS **********************************
// Variables necesarias
t=t+h;

ia_ref=a_ref*cos(2*pi*Fc_ref*(t));
ib_ref=a_ref*cos(2*pi*Fc_ref*(t)-x2_3*pi);
ic_ref=a_ref*cos(2*pi*Fc_ref*(t)+x2_3*pi);

Eia=ia_ref-ia_m;
Eib=ib_ref-ib_m;
Eic=ic_ref-ic_m;

EVac1=Vdc_3-vc1a_m;
EVac2=(2*Vdc_3-vc2a_m)*0.5;
EVbc1=Vdc_3-vc1b_m;
EVbc2=(2*Vdc_3-vc2b_m)*0.5;
EVcc1=Vdc_3-vc1c_m;
EVcc2=(2*Vdc_3-vc2c_m)*0.5;
```

```
// Calcular la función de costo
Jc = CP_IL * (Eia * Eia + Eib * Eib + Eic * Eic) + CP_VC * (EVac1 * EVac1 +
↪  EVac2 * EVac2 + EVbc1 * EVbc1 + EVbc2 * EVbc2 + EVcc1 * EVcc1 + EVcc2 *
↪  EVcc2);

if (Jc >= CD_Esup) {
    Csup = 1; //pasa banda superior
    }
else {
    Csup = 0; //no pasa banda superior
    }

if (Jc <= CD_Einf) {
         Cinf = 1; //pasa banda inferior
    }
else {
    Cinf = 0;  //no pasa banda inferior
    }

banda_n=Csup-Cinf;
// banda_n = 1 cuando pasa banda sup
// banda_n = -1 cuando pasa banda inf
// banda_n = 0 cuando no pasa bandas

cambio=(banda_n-banda_a);
// cambio = 0 cuando banda_n == banda_a (si el estado de antes era igual que
↪  ahora no hay cambio)
// cambio =/= 0 cuando banda_n =/= banda_a

banda=fabs(banda_n)*-1+1;
// banda = 0 cuando pasa por alguna de las bandas
// banda = 1 cuando no pasa por las bandas


if (cambio < 0 || cambio > 0) {  // si el cambio es distinto de cero
        auxSB = cambio;
    }

//esta bajando cuando
if ((banda && auxSB < 0)) {  // si pasa por alguna de las bandas y cambió
        baja = 1;
        sube = 0;
}
else {
        if ((banda && auxSB > 0)) {
                        sube = 1;
         baja = 0;
        }
      }

// situación en que está en modo MPC:
if ((Jc >= CD_Einf && baja) || (Jc >= CD_Esup)) {
        C_MPC = 1;
```

```
        }
else {
        C_MPC = 0;
    }

banda_a=banda_n; // banda_a es un discstate

//----Para probar MPC y PWM por separado----------------
//C_MPC=1; // MPC
//C_MPC=0; // PWM
/*********************************************/
//----------Corrientes en D Q----------

//if ( C_MPC == 1)
{
t1=t+2*h;

ia_r1=a_ref*cos(2*pi*Fc_ref*t1);
ib_r1=a_ref*cos(2*pi*Fc_ref*t1-x2_3*pi);
ic_r1=a_ref*cos(2*pi*Fc_ref*t1+x2_3*pi);

VaN_m=Vdc*s3a_1t+vc2a_m*(s2a_1t-s3a_1t)+vc1a_m*(s1a_1t-s2a_1t);
VbN_m=Vdc*s3b_1t+vc2b_m*(s2b_1t-s3b_1t)+vc1b_m*(s1b_1t-s2b_1t);
VcN_m=Vdc*s3c_1t+vc2c_m*(s2c_1t-s3c_1t)+vc1c_m*(s1c_1t-s2c_1t);
/*****************************************/
// Para ver como sigue la referencia en dq
i1_r=cos(2*pi*Fc_ref*(t));
i2_r=cos(2*pi*Fc_ref*(t)-x2_3*pi);
i3_r=cos(2*pi*Fc_ref*(t)+x2_3*pi);
//  Cálculo de Teta
alfa = i1_r;
beta = (i2_r-i3_r) * sqrt3_1;
teta = fabs(atan(beta/alfa));

if (alfa<0 && beta>=0)
    tetaf = pi - teta;
else
         {
    if (alfa<0 && beta<0)
        tetaf = pi + teta;
    else
                {
        if (alfa>=0 && beta<0)
            tetaf = 2 * pi-teta;
        else
            if (alfa>=0 && beta>=0)
                tetaf = teta;
                }
          }
//  Transformada ABC-DQ de corriente (IGUAL)
Da = cos(tetaf) * x2_3;
Db = cos(tetaf-g120) * x2_3;
Dc = cos(tetaf+g120) * x2_3;
Qa = -sin(tetaf) * x2_3;
```

```
Qb = -sin(tetaf-g120) * x2_3;
Qc = -sin(tetaf+g120) * x2_3;

//  Corrientes en eje D y eje Q
id_med = Da*ia_m + Db*ib_m + Dc*ic_m;
iq_med = Qa*ia_m + Qb*ib_m + Qc*ic_m;
id_ref = Da*ia_r1 + Db*ib_r1 + Dc*ic_r1;
iq_ref = Qa*ia_r1 + Qb*ib_r1 + Qc*ic_r1;

// ************* PARTE MPC ************************************************
//-------------- comienzo ANN MPC --------------
//ANN
int n_in = 24;
int n_lyr1 = 36;
int n_lyr2 = 24;

// ANN 3f
double input[n_in0]={ia_r1, ia_m, vc1a_m, vc2a_m, vc1_r-vc1a_m, vc2_r-vc2a_m,
↪  ia_r1-ia_m, VaN_m, ib_r1, ib_m, vc1b_m, vc2b_m, vc1_r-vc1b_m, vc2_r-vc2b_m,
↪  ib_r1-ib_m, VbN_m, ic_r1, ic_m, vc1c_m, vc2c_m, vc1_r-vc1c_m, vc2_r-vc2c_m,
↪  ic_r1-ic_m, VcN_m};

double input2[n_in];
for (int i = 0; i < n_in; i++){
        input2[i] = (2/(xmax1[i]-xmin1[i]))*(input[i] - xmin1[i])-1;
}

double outlayer1[n_lyr1];

for (int i = 0; i < n_lyr1; i++) {
        double suma = 0;
        for (int j = 0; j < n_in; j++) {
                suma += iw1[i][j] * input2[j];
        }
        outlayer1[i] = 2/(1+exp(-2*(suma + ib1[i][0])))-1;
}

double outlayer2[n_lyr2];
for (int i = 0; i < n_lyr2; i++) {
        double suma = 0;
        for (int j = 0; j < n_lyr1; j++) {
                suma += ow1[i][j] * outlayer1[j];
        }
        outlayer2[i] = suma + ob1[i][0];
}
double output[n_lyr2];
for (int i = 0; i < n_lyr2; i++){
        output[i] = 1/(1+exp(-outlayer2[i]));
}

double SSmax_a;
int indSSmax_a;

SSmax_a = output[0];
```

```
for (i=0; i<8; i++){
        if (output[i] > SSmax_a){
                SSmax_a = output[i];
                indSSmax_a = i;
        }
}

double SSmax_b;
int indSSmax_b;

SSmax_b = output[8];

for (i=8; i<16; i++){
        if (output[i] > SSmax_b){
                SSmax_b = output[i];
                indSSmax_b = i-8;
        }
}
double SSmax_c;
int indSSmax_c;

SSmax_c = output[16];

for (i=16; i<24; i++){
        if (output[i] > SSmax_c){
                SSmax_c = output[i];
                indSSmax_c = i-16;
        }
}
//------------- fin ANN MPC -----------------
// Salida de variables
s1_Ampc=SS1[indSSmax_a];
s2_Ampc=SS2[indSSmax_a];
s3_Ampc=SS3[indSSmax_a];
s1_Bmpc=SS1[indSSmax_b];
s2_Bmpc=SS2[indSSmax_b];
s3_Bmpc=SS3[indSSmax_b];
s1_Cmpc=SS1[indSSmax_c];
s2_Cmpc=SS2[indSSmax_c];
s3_Cmpc=SS3[indSSmax_c];
}
aux = a_ref;
salidaMPC = C_MPC;
J = Jc;

//*************** PARTE LINEAL ************************************//
//if( C_MPC == 0)
{
tt1=t-hs;

// Referencia corriente
ia_ref1p=a_ref*cos(2*pi*Fc_ref*tt1);
ib_ref1p=a_ref*cos(2*pi*Fc_ref*tt1-x2_3*pi);
```

```
ic_ref1p=a_ref*cos(2*pi*Fc_ref*tt1+x2_3*pi);
//------------ comienzo ANN PWM --------------
double input[12]={ia_ref1p, ia_m, ia_ref1p-ia_m, ib_ref1p, ib_m, ib_ref1p-ib_m,
↪  ic_ref1p, ic_m, ic_ref1p-ic_m, m1at, m1bt, m1ct};

double input2[12];
for (int i = 0; i < 12; i++){
        input2[i] = (2/(xmax2[i]-xmin2[i]))*(input[i] - xmin2[i])-1;
}

double outlayer1[12];

for (int i = 0; i < 12; i++) {
        double suma = 0;
        for (int j = 0; j < 12; j++) {
                suma += iw2[i][j] * input2[j];
        }
        outlayer1[i] = 2/(1+exp(-2*(suma + ib2[i][0])))-1;
}

double outlayer2[3];
for (int i = 0; i < 3; i++) {
        double suma = 0;
        for (int j = 0; j < 12; j++) {
                suma += ow2[i][j] * outlayer1[j];
        }
        outlayer2[i] = suma + ob2[i][0];
}
double output[3];
for (int i = 0; i < 3; i++){
        output[i] = ((ymax2[i]-ymin2[i])/2)*(outlayer2[i] + 1 ) + ymin2[i];
}
//------------- fin ANN PWM -----------------

//*****************************************/
// Salida de variables PWM
s1_A=output[0];
s2_A=output[0];
s3_A=output[0];
s1_B=output[1];
s2_B=output[1];
s3_B=output[1];
s1_C=output[2];
s2_C=output[2];
s3_C=output[2];
} // corchete de comienzo del PWM

/*****************************************/
// Dos ANN funcionando "en paralelo". Salidas correspondientes al MPC:
if ( C_MPC == 1){
s1_A=s1_Ampc;
s2_A=s2_Ampc;
s3_A=s3_Ampc;
s1_B=s1_Bmpc;
```

```
s2_B=s2_Bmpc;
s3_B=s3_Bmpc;
s1_C=s1_Cmpc;
s2_C=s2_Cmpc;
s3_C=s3_Cmpc;
}
VaN_m=Vdc*s3_A+vc2a_m*(s2_A-s3_A)+vc1a_m*(s1_A-s2_A);
VbN_m=Vdc*s3_B+vc2b_m*(s2_B-s3_B)+vc1b_m*(s1_B-s2_B);
VcN_m=Vdc*s3_C+vc2c_m*(s2_C-s3_C)+vc1c_m*(s1_C-s2_C);
m1at = (VaN_m)/Vdc;
m1bt = (VbN_m)/Vdc;
m1ct = (VcN_m)/Vdc;

/************* Control extra *****************/
if((J2-C_MPC)==1){
J1=1;
}
else
J1=0;
if(J1!=0)
{
s1_A=s1_Ampc;
s2_A=s2_Ampc;
s3_A=s3_Ampc;
s1_B=s1_Bmpc;
s2_B=s2_Bmpc;
s3_B=s3_Bmpc;
s1_C=s1_Cmpc;
s2_C=s2_Cmpc;
s3_C=s3_Cmpc;
}
J1=(J2);

if(J1==1&&J3<10){
J2=1;
}
else
J2=C_MPC;

if(J3>10)
J3=0;
J3=J3+1;
```

## .3. ANN Dual

```
// -------- def. variables Parte MPC -----------------
float t1=0;
float i1_r = 0;
float i2_r = 0;
float i3_r = 0;
```

```
// Cálculo de Teta
float alfa = 0;
float beta = 0;
float teta = 0;
float tetaf = 0;
// Transformada ABC-DQ de corriente
float Da=0;
float Db=0;
float Dc=0;
float Qa=0;
float Qb=0;
float Qc=0;
float iD_r=0;
float iD_m=0;
float iQ_r=0;
float iQ_m=0;
//-----------------------------------------------------
// Referencia corriente
aux=aux+h;
t1=aux-1*h;

i1_r=cos(2*pi*50*aux);
i2_r=cos(2*pi*50*aux-x2_3*pi);
i3_r=cos(2*pi*50*aux+x2_3*pi);

ia_r1=A_r*cos(2*pi*50*t1);
ib_r1=A_r*cos(2*pi*50*t1-x2_3*pi);
ic_r1=A_r*cos(2*pi*50*t1+x2_3*pi);

ia_r=A_r*cos(2*pi*50*aux);
ib_r=A_r*cos(2*pi*50*aux-x2_3*pi);
ic_r=A_r*cos(2*pi*50*aux+x2_3*pi);

/*****************************************/
// Cálculo de Teta
alfa = i1_r;
beta = (i2_r-i3_r) * raiz1_3;
teta = fabsf(atanf(beta/alfa));

if (alfa<0 && beta>=0)
    tetaf = pi - teta;
else
    {
    if (alfa<0 && beta<0)
        tetaf = pi + teta;
    else
                {
        if (alfa>=0 && beta<0)
            tetaf = 2 * pi-teta;
        else
            if (alfa>=0 && beta>=0)
                tetaf = teta;
                }
        }
```

```
// Transformada ABC-DQ de corriente
Da = cosf(tetaf) * x2_3;
Db = cosf(tetaf-g120) * x2_3;
Dc = cosf(tetaf+g120) * x2_3;

Qa = -sinf(tetaf) * x2_3;
Qb = -sinf(tetaf-g120) * x2_3;
Qc = -sinf(tetaf+g120) * x2_3;

/*****************************************/
// Corrientes en eje D y eje Q
iD_m = Da*ia_m + Db*ib_m + Dc*ic_m;
iQ_m = Qa*ia_m + Qb*ib_m + Qc*ic_m;
iD_r = Da*ia_r1 + Db*ib_r1 + Dc*ic_r1;
iQ_r = Qa*ia_r1 + Qb*ib_r1 + Qc*ic_r1;
/*****************************************/
id_m = iD_m;
iq_m = iQ_m;
id_r = iD_r;
iq_r = iQ_r;

//*************** PARTE LINEAL ************************************//
int n_in = 12;
int n_lyr1 = 14;
int n_lyr2 = 3;

//ANN
double input[n_in0]={ia_r1, ia_m, ia_r1-ia_m, ib_r1, ib_m, ib_r1-ib_m, ic_r1,
↪  ic_m, ic_r1-ic_m, s1at, s1bt, s1ct};

double input2[n_in];
for (int i = 0; i < n_in; i++){
        input2[i] = (2/(xmax[i]-xmin[i]))*(input[i] - xmin[i])-1;
}
//---------
double outlayer1[n_lyr1];

for (int i = 0; i < n_lyr1; i++) {
        double suma = 0;
        for (int j = 0; j < 12; j++) {
                suma += iw[i][j] * input2[j];
        }
        outlayer1[i] = 2/(1+exp(-2*(suma + ib[i][0])))-1;
}

//---------
double outlayer2[n_lyr2];
for (int i = 0; i < n_lyr2; i++) {
        double suma = 0;
        for (int j = 0; j < n_lyr1; j++) {
                suma += ow[i][j] * outlayer1[j];
        }
        outlayer2[i] = suma + ob[i][0];
}
```

```
//---------
double output[n_lyr2];
for (int i = 0; i < n_lyr2; i++){
        output[i] = ((ymax[i]-ymin[i])/2)*(outlayer2[i] + 1 ) + ymin[i];
}

// Salida de variables
s1_A=output[0];
s2_A=output[0];
s3_A=output[0];
s1_B=output[1];
s2_B=output[1];
s3_B=output[1];
s1_C=output[2];
s2_C=output[2];
s3_C=output[2];

s1at = s1_A;
s1bt = s1_B;
s1ct = s1_C;
```

# Apéndice C: Códigos experimentales

## .4. Unión de ANN con Histéresis

```
/*****************************************************************************
**      HEADER FILES
*****************************************************/
/* HW Macros */
/* System Defines */
#include "ANN_app.h"
#include "aceC6748.h"
#include "assert.h"
#include "math.h"
#include "psc.h"
#include "hw_psc_C6748.h"
#include "ace_monitor.h"
#include "soc_C6748.h"

#include "cossp.h"
#include "tanhsp.h"

#include "funciones.h" // lo usa en el ANN MPC

//includes ANN y parametros:
#include
↪  <pesosbias_ts_multiref_pwm_operacion_12n_indmod_PI_iabc_mt_1_6k_R151618.h>
#include <pesosbias_ts_operacion_vxn_SS_3f_ts10khz.h>

#include <tvmono.h>  // Tabla de verdad por fase
#include <param_v3.h> //parametros

/*
 * Monitor.
 */
#define SYSCLK2_FREQ      (150000000u)
#define HMI_UART_BAUDRATE (115200u)
#define HMI_UART_TIMEOUT  (20000)

bool tick_flag = 0;
#pragma DATA_SECTION (array0_A1, ".ddr")
#pragma DATA_SECTION (array0_A2, ".ddr")
#pragma DATA_SECTION (array0_B1, ".ddr")
#pragma DATA_SECTION (array0_B2, ".ddr")
```

```
#pragma DATA_SECTION (array1_A1, ".ddr")
#pragma DATA_SECTION (array1_A2, ".ddr")
#pragma DATA_SECTION (array1_B1, ".ddr")
#pragma DATA_SECTION (array1_B2, ".ddr")
#pragma DATA_SECTION (array2_A1, ".ddr")
#pragma DATA_SECTION (array2_A2, ".ddr")
#pragma DATA_SECTION (array2_B1, ".ddr")
#pragma DATA_SECTION (array2_B2, ".ddr")
#pragma DATA_SECTION (array3_A1, ".ddr")
#pragma DATA_SECTION (array3_A2, ".ddr")

extern void Init_Platform(void);
extern void Init_Interrupts(void);
extern void Init_Adc(void);
extern void Init_Iop(void);
extern void Init_Mod(void);
extern void Init_DS_LED(void);

#define array_length 4000

// Variables del sistema
float s1_A = 0;
float s2_A = 0;
float s3_A = 0;
float s1_B = 0;
float s2_B = 0;
float s3_B = 0;
float s1_C = 0;
float s2_C = 0;
float s3_C = 0;

float s1a_1t;
float s2a_1t;
float s3a_1t;
float s1b_1t;
float s2b_1t;
float s3b_1t;
float s1c_1t;
float s2c_1t;
float s3c_1t;

//bandas:
float auxSB;
float banda_a;
float tx;
float salidaMPC;
float J;

float Eia=0;
float Eib=0;
float Eic=0;
float EVac1=0;
float EVac2=0;
float EVbc1=0;
```

```
float EVbc2=0;
float EVcc1=0;
float EVcc2=0;

float Jc=0;

float C_MPC=0;

//control extra:
float J1 = 100;
float J2 = 0;
float J3 = 0;
float J4 = 0;

float m1at;
float m1bt;
float m1ct;

float mA=0;
float mB=0;
float mC=0;

float ia_r1,ib_r1,ic_r1;

float ia_m;
float ib_m;
float ic_m;

float vc1a_m;
float vc2a_m;
float vc1b_m;
float vc2b_m;
float vc1c_m;
float vc2c_m;

float alfa;
float beta;
float Da, Db, Dc;
float Qa, Qb, Qc;
float id_med,id_ref;
float iq_med,iq_ref;

float teta;
float tetaf;

double t1=0;
double t=0;

float ia_r,ib_r,ic_r;
float ia_ref,ib_ref,ic_ref;
float i1_r,i2_r,i3_r;

float VaN_m;
float VbN_m;
```

```
float VcN_m;

double tt1=0;
float hs=Ts;

float ia_ref1p = 0;
float ib_ref1p = 0;
float ic_ref1p = 0;

float s1_Ampc=0;
float s2_Ampc=0;
float s3_Ampc=0;
float s1_Bmpc=0;
float s2_Bmpc=0;
float s3_Bmpc=0;
float s1_Cmpc=0;
float s2_Cmpc=0;
float s3_Cmpc=0;

// Ganancias y offsets --------------------------------
float gain_ia = -5.82536055417565;
float gain_ib = -6.01639519815913;
float gain_ic = -5.84099191599484;
float offset_i_a = 0;
float offset_i_b = 0;
float offset_i_c = 0;

float gain_v1 = 34.0803110998317;
float gain_v2 = 64.1207956726628;
float offset_v1 = 0;
float offset_v2 = 0;

float gain_vdc = 102.524765155101;
float offset_vdc = 0;

// Variables para cálculo de offsets
float ia_zero = 0;
float ib_zero = 0;
float ic_zero = 0;
float vc1a_zero = 0;
float vc2a_zero = 0;
float Vdc_zero = 0;
int contador = 0;

double aux = 0;
float a_ref = 2;
float a_mpc = 0;
float a_ref1 = 1;
//-------------------------------------------------

int16_t Med_ADC0_A1_16;
int16_t Med_ADC0_A2_16;
int16_t Med_ADC0_B1_16;
int16_t Med_ADC0_B2_16;
```

```
float array0_A1[array_length];
float array0_A2[array_length];
float array0_B1[array_length];
float array0_B2[array_length];

int16_t Med_ADC1_A1_16;
int16_t Med_ADC1_A2_16;
int16_t Med_ADC1_B1_16;
int16_t Med_ADC1_B2_16;
float array1_A1[array_length];
float array1_A2[array_length];
float array1_B1[array_length];
float array1_B2[array_length];

int16_t Med_ADC2_A1_16;
int16_t Med_ADC2_A2_16;

float array2_A1[array_length];
float array2_A2[array_length];
float array2_B1[array_length];
float array2_B2[array_length];

float array3_A1[array_length];
float array3_A2[array_length];

int array_i = 0;
float leer = 0;

int i=0;
int j=0;
float auxiliar;

float Vdc=0;
float Vdc_2=0;
float Vdc_3=0;

//ANN MPC
int n_in = 24;
int n_lyr1 = 36;
int n_lyr2 = 24;

float input[24];
float input2[24];
float outlayer1[36];
float outlayer2[24];
float output[24];

//ANN PI
int n_in_p = 12;
int n_lyr1_p = 12;
int n_lyr2_p = 3;

float input_p[12];
float input2_p[12];
```

```
float outlayer1_p[12];
float outlayer2_p[3];
float output_p[3];

float Vdc_1=0;
float tolerance;
//-----------------------------------------------------------------------------
void main(void)
{

    /* All peripherals to be uses are configured here */

    /* Interruptions must be configured after peripherals configuration */
    Init_Platform();     // Configuración de PLL del DSP y de comunicación con
    ↪  la FPGA
    Init_Mod();          // Configuración de los Timers y moduladores PWM
    Init_Adc();          // Configuración de los ADCs
    Init_Iop();          // Configuración de los puertos IO
    Init_DS_LED();       // Configiración de DipSwitch y LEDs

    Init_Interrupts();   // Configuración de las interrupciones. Debe ser la
    ↪  última función a llamar

    /*
    ACE_USRIO_turnOffLd(ACE_USRIO_LD0);
    ACE_USRIO_turnOffLd(ACE_USRIO_LD1);
    ACE_USRIO_turnOffLd(ACE_USRIO_LD2);
    ACE_USRIO_turnOffLd(ACE_USRIO_LD3);
    */

    /*-------------------------------------------------------------------------*/
    /*  BACKGROUND ROUTINE
    ↪  */
    /*-------------------------------------------------------------------------*/
    /* LOCAL MONITOR CONFIGURATION */
    /* UART configuration */

    ↪  PSCModuleControl(SOC_PSC_1_REGS,HW_PSC_UART2,PSC_POWERDOMAIN_ALWAYS_ON,PSC_MDCTL_NEXT_ENABL
    ACE_DSPUART_PinMuxSetup(SOC_UART_2, FALSE);
    ACE_MON_init(SOC_UART_2_REGS, SYSCLK2_FREQ, HMI_UART_BAUDRATE,
    ↪  HMI_UART_TIMEOUT);

    ACE_MON_addVariableToMonitor(&ia_m, ACE_MON_FLOAT,   sizeof(ia_m),
    ↪  ACE_MON_ACCESS_READONLY, array0_A1, sizeof(array0_A1));
    ACE_MON_addVariableToMonitor(&ib_m, ACE_MON_FLOAT,   sizeof(ib_m),
    ↪  ACE_MON_ACCESS_READONLY, array0_A2, sizeof(array0_A2));
    ACE_MON_addVariableToMonitor(&ic_m, ACE_MON_FLOAT,   sizeof(ic_m),
    ↪  ACE_MON_ACCESS_READONLY, array0_B1, sizeof(array0_B1));

    ACE_MON_addVariableToMonitor(&vc1a_m, ACE_MON_FLOAT,   sizeof(vc1a_m),
    ↪  ACE_MON_ACCESS_READONLY, array0_B2, sizeof(array0_B2));
    ACE_MON_addVariableToMonitor(&vc2a_m, ACE_MON_FLOAT,   sizeof(vc2a_m),
    ↪  ACE_MON_ACCESS_READONLY, array1_A1, sizeof(array1_A1));
```

```
    ACE_MON_addVariableToMonitor(&vc1b_m, ACE_MON_FLOAT,   sizeof(vc1b_m),
    ↪  ACE_MON_ACCESS_READONLY, array1_A2, sizeof(array1_A2));
    ACE_MON_addVariableToMonitor(&vc2b_m, ACE_MON_FLOAT,   sizeof(vc2b_m),
    ↪  ACE_MON_ACCESS_READONLY, array1_B1, sizeof(array1_B1));

    ACE_MON_addVariableToMonitor(&vc1c_m, ACE_MON_FLOAT,   sizeof(vc1c_m),
    ↪  ACE_MON_ACCESS_READONLY, array1_B2, sizeof(array1_B2));
    ACE_MON_addVariableToMonitor(&vc2c_m, ACE_MON_FLOAT,   sizeof(vc2c_m),
    ↪  ACE_MON_ACCESS_READONLY, array2_A1, sizeof(array2_A1));

    ACE_MON_addVariableToMonitor(&ia_r1, ACE_MON_FLOAT,   sizeof(ia_r1),
    ↪  ACE_MON_ACCESS_READONLY, array2_A2, sizeof(array2_A2));
    ACE_MON_addVariableToMonitor(&id_med, ACE_MON_FLOAT,   sizeof(id_med),
    ↪  ACE_MON_ACCESS_READONLY, array2_B1, sizeof(array2_B1));
    ACE_MON_addVariableToMonitor(&iq_med, ACE_MON_FLOAT,   sizeof(iq_med),
    ↪  ACE_MON_ACCESS_READONLY, array2_B2, sizeof(array2_B2));

    ↪  /*************************************************************************/
    ACE_MON_addVariableToMonitor(&Vdc, ACE_MON_FLOAT,   sizeof(Vdc),
    ↪  ACE_MON_ACCESS_READONLY, array3_A1, sizeof(array3_A1));
    ACE_MON_addVariableToMonitor(&a_mpc, ACE_MON_FLOAT,   sizeof(a_mpc),
    ↪  ACE_MON_ACCESS_WRITABLE,0,0);

    ACE_MON_addVariableToMonitor(&leer, ACE_MON_FLOAT,   sizeof(leer),
    ↪  ACE_MON_ACCESS_WRITABLE,0,0);
    ACE_MON_addVariableToMonitor(&a_ref, ACE_MON_FLOAT,   sizeof(a_ref),
    ↪  ACE_MON_ACCESS_WRITABLE,0,0);

    while(1)
    {
        ACE_MON_tickBackground();
    }

}
/*************************************************************************/
extern void ADC_fpgaSyncInterrupt_isr(void){
    //int var_temp_int;
    /* Clear flag that caused the interruption */
    INT0CLRFLG = INT0_ADC0_MASK; // Borra el Flag sobreescribiendo un 1

// comienzo código offsets:---------------------------------
   if (a_ref==0){
        contador=0;
        ia_zero = 0;
        ib_zero = 0;
        ic_zero = 0;
        vc1a_zero = 0;
        vc2a_zero = 0;
        Vdc_zero = 0;
        a_ref=1;
    }
    contador=contador+1;

    if (contador>1000){
```

```
            contador=3000;}
    if (contador<=1000){

        ia_zero = ia_zero+((int)Med_ADC0_A1_16 *
        ↪  ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_ia ;
        ib_zero = ib_zero+((int)Med_ADC0_A2_16 *
        ↪  ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_ib;
        ic_zero = ic_zero+((int)Med_ADC0_B1_16 *
        ↪  ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_ic;
        vc1a_zero = vc1a_zero+((int)Med_ADC0_B2_16 *
        ↪  ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_v1;
        vc2a_zero = vc2a_zero+((int)Med_ADC1_A1_16 *
        ↪  ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_v2;
        Vdc_zero =
        ↪  Vdc_zero+((int)Med_ADC2_A2_16*ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_vdc;
    }
    else{
        offset_i_a=-ia_zero*0.001;
        offset_i_b=-ib_zero*0.001;
        offset_i_c=-ic_zero*0.001;
        offset_v1=-vc1a_zero*0.001;
        offset_v2=-vc2a_zero*0.001;
        offset_vdc=-Vdc_zero*0.001;
    }
//fin código offsets------------------------

//----- cuando hace el cambio de referencia:
   tolerance=1e-6;
   if((leer==1)&&((array_i>=0.45*array_length)&&(fabs(ia_r1 - a_ref1) <
   ↪  tolerance))){
           a_ref1=a_ref;
   }
   //-------------------------------------------
    Med_ADC0_A1_16 =  ADC0CHA1;
    Med_ADC0_A2_16 =  ADC0CHA2;
    Med_ADC0_B1_16 =  ADC0CHB1;
    Med_ADC0_B2_16 =  ADC0CHB2;

    ia_m = -(((int)Med_ADC0_A1_16 *
    ↪  ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_ia + offset_i_a);
    ib_m = -(((int)Med_ADC0_A2_16 *
    ↪  ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_ib + offset_i_b);
    ic_m = -(((int)Med_ADC0_B1_16 *
    ↪  ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_ic + offset_i_c);
    vc1a_m = ((int)Med_ADC0_B2_16 *
    ↪  ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_v1 + offset_v1;

    Med_ADC1_A1_16 =  ADC1CHA1;
    Med_ADC1_A2_16 =  ADC1CHA2;
    Med_ADC1_B1_16 =  ADC1CHB1;
    Med_ADC1_B2_16 =  ADC1CHB2;

    vc2a_m = ((int)Med_ADC1_A1_16 *
    ↪  ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_v2 + offset_v2;
```

```
vc1b_m = ((int)Med_ADC1_A2_16 *
↪  ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_v1 + offset_v1;
vc2b_m = ((int)Med_ADC1_B1_16 *
↪  ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_v2 + offset_v2;
vc1c_m = ((int)Med_ADC1_B2_16 *
↪  ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_v1 + offset_v1;

Med_ADC2_A1_16 =  ADC2CHA1;
Med_ADC2_A2_16 =  ADC2CHA2;

vc2c_m =((int)Med_ADC2_A1_16 *
↪  ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_v2 + offset_v2;
Vdc = ((int)Med_ADC2_A2_16*ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_vdc +
↪  offset_vdc;

Vdc_2=Vdc*0.5;
Vdc_3=Vdc*0.3333333333333333;

tick_flag = ACE_MON_tickForeground();

if (leer==1){
    array0_A1[array_i] = ia_m;
    array0_A2[array_i] = ib_m;
    array0_B1[array_i] = ic_m;
    array0_B2[array_i] = vc1a_m;
    array1_A1[array_i] = vc2a_m;
    array1_A2[array_i] = vc1b_m;
    array1_B1[array_i] = vc2b_m;
    array1_B2[array_i] = vc1c_m;
    array2_A1[array_i] = vc2c_m;
    array2_A2[array_i] = ia_r1;
    array2_B1[array_i] = id_med;
    array2_B2[array_i] = iq_med;
    array3_A1[array_i] = Vdc;
    array3_A2[array_i] = auxiliar;

    array_i = array_i+1;

    if (array_i>= array_length){
        array_i = 0;
        leer=0;
    }
}
// -------------------------
t = t + h;
i1_r=cossp(pi100*t);
i2_r=cossp(pi100*t-x2_3*pi);
i3_r=cossp(pi100*t+x2_3*pi);
// Cálculo de Teta
alfa = i1_r;
beta = (i2_r-i3_r) * raiz1_3;
teta = fabsf(atanf(beta/alfa));

if (alfa<0 && beta>=0)
```

```
        tetaf = pi - teta;
    else
        {
        if (alfa<0 && beta<0)
            tetaf = pi + teta;
        else
            {
            if (alfa>=0 && beta<0)
                tetaf = 2 * pi-teta;
            else
                if (alfa>=0 && beta>=0)
                    tetaf = teta;
            }
        }

    // Transformada ABC-DQ de corriente
    Da = cosf(tetaf) * x2_3;
    Db = cosf(tetaf-g120) * x2_3;
    Dc = cosf(tetaf+g120) * x2_3;

    Qa = -sinf(tetaf) * x2_3;
    Qb = -sinf(tetaf-g120) * x2_3;
    Qc = -sinf(tetaf+g120) * x2_3;

    /****************************************/
    // Corrientes en eje D y eje Q
    id_med = Da*ia_m + Db*ib_m + Dc*ic_m;
    iq_med = Qa*ia_m + Qb*ib_m + Qc*ic_m;
    id_ref = Da*ia_r1 + Db*ib_r1 + Dc*ic_r1;
    iq_ref = Qa*ia_r1 + Qb*ib_r1 + Qc*ic_r1;


    // --------------- HISTERESIS ---------------------------------------------
    ia_ref=a_ref1*cossp(pi100*t);
    ib_ref=a_ref1*cossp(pi100*t - x2_3*pi);
    ic_ref=a_ref1*cossp(pi100*t + x2_3*pi);

    Eia=(ia_ref-ia_m);
    Eib=(ib_ref-ib_m);
    Eic=(ic_ref-ic_m);

    EVac1=Vdc_3-vc1a_m;
    EVac2=(2*Vdc_3-vc2a_m)*0.5;
    EVbc1=Vdc_3-vc1b_m;
    EVbc2=(2*Vdc_3-vc2b_m)*0.5;
    EVcc1=Vdc_3-vc1c_m;
    EVcc2=(2*Vdc_3-vc2c_m)*0.5;

    // Calcular la función de costo
    Jc = CP_IL * (Eia * Eia + Eib * Eib + Eic * Eic) + CP_VC * (EVac1 * EVac1 +
    ↪  EVac2 * EVac2 + EVbc1 * EVbc1 + EVbc2 * EVbc2 + EVcc1 * EVcc1 + EVcc2 *
    ↪  EVcc2);

    if (a_mpc==2){
```

```
    if (Jc >= CD_Esup){
        C_MPC = 1;
        J2=0;
    }
    else if (Jc <= CD_Einf){
        C_MPC = 0;
        J2=1;
    }
    }
    else
    {
        C_MPC=a_mpc;
    }
if(J1<=80){
    C_MPC=1;
}
    //----Para probar MPC y PWM por
    ↪  separado**********************************
    //C_MPC=a_mpc; // MPC: C_MPC=1 ; PWM: C_MPC=0
    // ------------------------------------------------------------------------

// ************* PARTE MPC ************************************************
    if ( C_MPC == 1) {

    t1 = t + h + h;

    ia_r1 = a_ref1*cossp(pi100*t1);
    ib_r1 = a_ref1*cossp(pi100*t1 - x2_3*pi);
    ic_r1 = a_ref1*cossp(pi100*t1 + x2_3*pi);

    VaN_m=Vdc*s3_A+vc2a_m*(s2_A-s3_A)+vc1a_m*(s1_A-s2_A);
    VbN_m=Vdc*s3_B+vc2b_m*(s2_B-s3_B)+vc1b_m*(s1_B-s2_B);
    VcN_m=Vdc*s3_C+vc2c_m*(s2_C-s3_C)+vc1c_m*(s1_C-s2_C);

    //-----ANN MPC -------------------------------------------
    input[0] = ia_r1;
    input[1] = ia_m;
    input[2] = vc1a_m;
    input[3] = vc2a_m;
    input[4] = vc1_r-vc1a_m;
    input[5] = vc2_r-vc2a_m;
    input[6] = ia_r1-ia_m;
    input[7] = VaN_m;
    input[8] = ib_r1;
    input[9] = ib_m;
    input[10] = vc1b_m;
    input[11] = vc2b_m;
    input[12] = vc1_r-vc1b_m;
    input[13] = vc2_r-vc2b_m;
    input[14] = ib_r1-ib_m;
    input[15] = VbN_m;
    input[16] = ic_r1;
    input[17] = ic_m;
    input[18] = vc1c_m;
```

```
input[19] = vc2c_m;
input[20] = vc1_r-vc1c_m;
input[21] = vc2_r-vc2c_m;
input[22] = ic_r1-ic_m;
input[23] = VcN_m;

ajusteminmax(xmin,xmax,input,input2,n_in);

mulmat2(36,24,&iw[0][0], ib, input2,outlayer1);
ccbtanhsp_v(outlayer1, n_lyr1);

mulmat2(24,36,&ow[0][0], ob, outlayer1,outlayer2);

ccblogsigsp_v(outlayer2, output, n_lyr2);

int indSSmax_a;
indSSmax_a = fsmax(0,8,output,output[0]);

int indSSmax_b;
indSSmax_b = fsmax(8,16,output,output[8]);

int indSSmax_c;
indSSmax_c = fsmax(16,24,output,output[16]);

//auxiliar = 100*indSSmax_a + 10*indSSmax_b + indSSmax_c;
auxiliar = a_ref;

// Salida de variables
s1_Ampc=SS1[indSSmax_a];
s2_Ampc=SS2[indSSmax_a];
s3_Ampc=SS3[indSSmax_a];
s1_Bmpc=SS1[indSSmax_b];
s2_Bmpc=SS2[indSSmax_b];
s3_Bmpc=SS3[indSSmax_b];
s1_Cmpc=SS1[indSSmax_c];
s2_Cmpc=SS2[indSSmax_c];
s3_Cmpc=SS3[indSSmax_c];

} //fin mpc

aux = a_ref;
salidaMPC = C_MPC;
J = Jc;

//**************** PARTE LINEAL ************************************//
if( C_MPC == 0)
{

tt1 = t - h;
ia_ref1p = a_ref1*cossp(pi100*tt1);
ib_ref1p = a_ref1*cossp(pi100*tt1 - x2_3*pi);
ic_ref1p = a_ref1*cossp(pi100*tt1 + x2_3*pi);

ia_r1=ia_ref1p;
```

```
    ib_r1=ib_ref1p;
    ic_r1=ic_ref1p;

    //--------------------ANN PI---------------------------
    input_p[0] = ia_ref1p;
    input_p[1] = ia_m;
    input_p[2] = ia_ref1p-ia_m;
    input_p[3] = ib_ref1p;
    input_p[4] = ib_m;
    input_p[5] = ib_ref1p-ib_m;
    input_p[6] = ic_ref1p;
    input_p[7] = ic_m;
    input_p[8] = ic_ref1p-ic_m;
    input_p[9] = m1at;
    input_p[10] = m1bt;
    input_p[11] = m1ct;

    for (i = 0; i < n_in_p; ++i)
    {
        input2_p[i] = (2/(xmax_p[i]-xmin_p[i]))*(input_p[i] - xmin_p[i])-1;
    }

    for (i = 0; i < n_lyr1_p; i++) {
        float suma = 0;
        for (j = 0; j < n_in_p; j++) {
            suma += iw_p[i][j] * input2_p[j];
        }
        outlayer1_p[i] = tanhsp(suma + ib_p[i][0]);
    }

    for (i = 0; i < n_lyr2_p; i++) {
        float suma = 0;
        for (j = 0; j < n_lyr1_p; j++) {
            suma += ow_p[i][j] * outlayer1_p[j];
        }
        outlayer2_p[i] = suma + ob_p[i][0];
    }

    for (i = 0; i < n_lyr2_p; i++){
        output_p[i] = ((ymax_p[i]-ymin_p[i])/2)*(outlayer2_p[i] + 1 ) +
        ↪  ymin_p[i];
    }

    /*****************************************/
    mA = output_p[0];
    mB = output_p[1];
    mC = output_p[2];
    }  //corchete de comienzo del PWM !!!
    /*****************************************/
// nuevo porque esta cuestion no reconoce los switches del pwm
    if ( C_MPC == 0){
    s1_A=mA;
    s2_A=mA;
    s3_A=mA;
```

```
  s1_B=mB;
  s2_B=mB;
  s3_B=mB;
  s1_C=mC;
  s2_C=mC;
  s3_C=mC;
  }

  // Dos ANN funcionando "en paralelo". Salidas correspondientes al MPC:

  if ( C_MPC == 1){
  s1_A=s1_Ampc;
  s2_A=s2_Ampc;
  s3_A=s3_Ampc;
  s1_B=s1_Bmpc;
  s2_B=s2_Bmpc;
  s3_B=s3_Bmpc;
  s1_C=s1_Cmpc;
  s2_C=s2_Cmpc;
  s3_C=s3_Cmpc;
  }

/************* Control extra *****************/
  if((J2-C_MPC)==1){
  J1=1;
  }
  else
  J1=0;
  if(J1!=0)
  {
  s1_A=s1_Ampc;
  s2_A=s2_Ampc;
  s3_A=s3_Ampc;
  s1_B=s1_Bmpc;
  s2_B=s2_Bmpc;
  s3_B=s3_Bmpc;
  s1_C=s1_Cmpc;
  s2_C=s2_Cmpc;
  s3_C=s3_Cmpc;
  }
  J1=(J2);

  if(J1==1&&J3<10){
  J2=1;
  }
  else
  J2=C_MPC;

  if(J3>10)
  J3=0;

  J3=J3+1;
```

```
    // Salida de variables
    M0CMPR1 = s1_A * M0_MAX;
    M0CMPR2 = s1_B * M0_MAX;
    M0CMPR3 = s1_C * M0_MAX;

    M1CMPR1 = s2_A * M0_MAX;
    M1CMPR2 = s2_B * M0_MAX;
    M1CMPR3 = s2_C * M0_MAX;

    M2CMPR1 = s3_A * M0_MAX;
    M2CMPR2 = s3_B * M0_MAX;
    M2CMPR3 = s3_C * M0_MAX;

    auxiliar = a_ref;
    m1at=mA;
    m1bt=mB;
    m1ct=mC;

    M3CMPR1 = M0_MAX/2;
}
/*****************************************************************************/
```

## .5. ANN Dual

```
/*****************************************************************************
**      HEADER FILES
*****************************************/
/* HW Macros */
/* System Defines */
#include "ANN_app.h"
#include "aceC6748.h"
#include "assert.h"
#include "math.h"
#include "psc.h"
#include "hw_psc_C6748.h"
#include "ace_monitor.h"
#include "soc_C6748.h"

#include "cossp.h"
#include "tanhsp.h"

#include "funciones.h" // lo usa en el ANN MPC

//includes ANN y parametros:
#include <pesosbias_b.h>
#include <tvmono.h>  // Tabla de verdad por fase
#include <param_v3.h> //parametros
/*
 * Monitor.
```

```
 */
#define SYSCLK2_FREQ      (150000000u)
#define HMI_UART_BAUDRATE (115200u)
#define HMI_UART_TIMEOUT  (20000)

bool tick_flag = 0;
#pragma DATA_SECTION (array0_A1, ".ddr")
#pragma DATA_SECTION (array0_A2, ".ddr")
#pragma DATA_SECTION (array0_B1, ".ddr")
#pragma DATA_SECTION (array0_B2, ".ddr")
#pragma DATA_SECTION (array1_A1, ".ddr")
#pragma DATA_SECTION (array1_A2, ".ddr")
#pragma DATA_SECTION (array1_B1, ".ddr")
#pragma DATA_SECTION (array1_B2, ".ddr")
#pragma DATA_SECTION (array2_A1, ".ddr")
#pragma DATA_SECTION (array2_A2, ".ddr")
#pragma DATA_SECTION (array2_B1, ".ddr")
#pragma DATA_SECTION (array2_B2, ".ddr")
#pragma DATA_SECTION (array3_A1, ".ddr")
#pragma DATA_SECTION (array3_A2, ".ddr")

extern void Init_Platform(void);
extern void Init_Interrupts(void);
extern void Init_Adc(void);
extern void Init_Iop(void);
extern void Init_Mod(void);
extern void Init_DS_LED(void);

#define array_length 4000

float s1_A = 0;
float s2_A = 0;
float s3_A = 0;
float s1_B = 0;
float s2_B = 0;
float s3_B = 0;
float s1_C = 0;
float s2_C = 0;
float s3_C = 0;

float s1at;
float s1bt;
float s1ct;

float vc1a_m;
float vc2a_m;
float vc1b_m;
float vc2b_m;
float vc1c_m;
float vc2c_m;

float alfa=0;
float beta=0;
float teta=0;
```

```
float tetaf=0;
float Da=0;
float Db=0;
float Dc=0;
float Qa=0;
float Qb=0;
float Qc=0;

float id_m=0;
float id_r=0;
float iq_m=0;
float iq_r=0;
double t1;
double t1x;
double t=0;

float ia_m;
float ib_m;
float ic_m;

float ia_r1,ib_r1,ic_r1;
float ia_r1x,ib_r1x,ic_r1x;
float ia_r1o,ib_r1o,ic_r1o;

float ia_r1d,ib_r1d,ic_r1d;
float i1_r,i2_r,i3_r;

// Ganancias y offsets (nuevo) ---------------------------------
float gain_ia = -5.82536055417565;
float gain_ib = -6.01639519815913;
float gain_ic = -5.84099191599484;
float offset_i_a = 0;
float offset_i_b = 0;
float offset_i_c = 0;

float gain_v1 = 34.0803110998317;
float gain_v2 = 64.1207956726628;
float offset_v1 = 0;
float offset_v2 = 0;

float gain_vdc = 102.524765155101;
float offset_vdc = 0;

// Variables para cálculo de offsets
float ia_zero = 0;
float ib_zero = 0;
float ic_zero = 0;
float vc1a_zero = 0;
float vc2a_zero = 0;
float Vdc_zero = 0;
int contador = 0;

double aux = 0;
float a_ref = 2;
```

```
float a_ref1 = 1;

float mA=0;
float mB=0;
float mC=0;

//-------------------------------------------------
int16_t Med_ADC0_A1_16;
int16_t Med_ADC0_A2_16;
int16_t Med_ADC0_B1_16;
int16_t Med_ADC0_B2_16;
float array0_A1[array_length];
float array0_A2[array_length];
float array0_B1[array_length];
float array0_B2[array_length];

int16_t Med_ADC1_A1_16;
int16_t Med_ADC1_A2_16;
int16_t Med_ADC1_B1_16;
int16_t Med_ADC1_B2_16;
float array1_A1[array_length];
float array1_A2[array_length];
float array1_B1[array_length];
float array1_B2[array_length];

int16_t Med_ADC2_A1_16;
int16_t Med_ADC2_A2_16;

float array2_A1[array_length];
float array2_A2[array_length];
float array2_B1[array_length];
float array2_B2[array_length];

float array3_A1[array_length];
float array3_A2[array_length];

int array_i = 0;
float leer = 0;

int i=0;
int j=0;
float auxiliar;
float Vdc=0;
float Vdc_2=0;
float Vdc_3=0;

//ANN MIA
int n_in = 12;
int n_lyr1 = 14;
int n_lyr2 = 3;

float input[12];
float input2[12];
float outlayer1[14];
```

```
float outlayer2[3];
float output[3];
//-------------
//ANN PI
int n_in_p = 12;
int n_lyr1_p = 12;
int n_lyr2_p = 3;

float input_p[12];
float input2_p[12];
float outlayer1_p[12];
float outlayer2_p[3];
float output_p[3];

double tt1=0;
float ia_ref1p = 0;
float ib_ref1p = 0;
float ic_ref1p = 0;

float Vdc_1=0;

float ea=0;
float eb=0;
float ec=0;
float suma = 0;
float cual=0;

float s1atp=0;
float s1btp=0;
float s1ctp=0;

float tolerance=0;
//-------------------------------------------------------------------------------
void main(void)
{
    /* All peripherals to be uses are configured here */

    /* Interruptions must be configured after peripherals configuration */
    Init_Platform();     // Configuración de PLL del DSP y de comunicación con
    ↪  la FPGA
    Init_Mod();          // Configuración de los Timers y moduladores PWM
    Init_Adc();          // Configuración de los ADCs
    Init_Iop();          // Configuración de los puertos IO
    Init_DS_LED();       // Configiración de DipSwitch y LEDs

    Init_Interrupts();   // Configuración de las interrupciones. Debe ser la
    ↪  última función a llamar
    /*-----------------------------------------------------------------------*/
    /*  BACKGROUND ROUTINE
    ↪  */
    /*-----------------------------------------------------------------------*/
    /* LOCAL MONITOR CONFIGURATION */
    /* UART configuration */
```

```
    ↪ PSCModuleControl(SOC_PSC_1_REGS,HW_PSC_UART2,PSC_POWERDOMAIN_ALWAYS_ON,PSC_MDCTL_NEXT_ENABL
    ACE_DSPUART_PinMuxSetup(SOC_UART_2, FALSE);
    ACE_MON_init(SOC_UART_2_REGS, SYSCLK2_FREQ, HMI_UART_BAUDRATE,
    ↪ HMI_UART_TIMEOUT);
//--------
    ACE_MON_addVariableToMonitor(&ia_m, ACE_MON_FLOAT,   sizeof(ia_m),
    ↪ ACE_MON_ACCESS_READONLY, array0_A1, sizeof(array0_A1));
    ACE_MON_addVariableToMonitor(&ib_m, ACE_MON_FLOAT,   sizeof(ib_m),
    ↪ ACE_MON_ACCESS_READONLY, array0_A2, sizeof(array0_A2));
    ACE_MON_addVariableToMonitor(&ic_m, ACE_MON_FLOAT,   sizeof(ic_m),
    ↪ ACE_MON_ACCESS_READONLY, array0_B1, sizeof(array0_B1));

    ACE_MON_addVariableToMonitor(&vc1a_m, ACE_MON_FLOAT,   sizeof(vc1a_m),
    ↪ ACE_MON_ACCESS_READONLY, array0_B2, sizeof(array0_B2));
    ACE_MON_addVariableToMonitor(&vc2a_m, ACE_MON_FLOAT,   sizeof(vc2a_m),
    ↪ ACE_MON_ACCESS_READONLY, array1_A1, sizeof(array1_A1));
    ACE_MON_addVariableToMonitor(&vc1b_m, ACE_MON_FLOAT,   sizeof(vc1b_m),
    ↪ ACE_MON_ACCESS_READONLY, array1_A2, sizeof(array1_A2));

    ACE_MON_addVariableToMonitor(&vc2b_m, ACE_MON_FLOAT,   sizeof(vc2b_m),
    ↪ ACE_MON_ACCESS_READONLY, array1_B1, sizeof(array1_B1));
    ACE_MON_addVariableToMonitor(&vc1c_m, ACE_MON_FLOAT,   sizeof(vc1c_m),
    ↪ ACE_MON_ACCESS_READONLY, array1_B2, sizeof(array1_B2));
    ACE_MON_addVariableToMonitor(&vc2c_m, ACE_MON_FLOAT,   sizeof(vc2c_m),
    ↪ ACE_MON_ACCESS_READONLY, array2_A1, sizeof(array2_A1));

    ACE_MON_addVariableToMonitor(&ia_r1, ACE_MON_FLOAT,   sizeof(ia_r1),
    ↪ ACE_MON_ACCESS_READONLY, array2_A2, sizeof(array2_A2));
    ACE_MON_addVariableToMonitor(&id_m, ACE_MON_FLOAT,   sizeof(id_m),
    ↪ ACE_MON_ACCESS_READONLY, array2_B1, sizeof(array2_B1));
    ACE_MON_addVariableToMonitor(&iq_m, ACE_MON_FLOAT,   sizeof(iq_m),
    ↪ ACE_MON_ACCESS_READONLY, array2_B2, sizeof(array2_B2));
    //---------------------------
    ACE_MON_addVariableToMonitor(&Vdc, ACE_MON_FLOAT,   sizeof(Vdc),
    ↪ ACE_MON_ACCESS_READONLY, array3_A1, sizeof(array3_A1));

    ACE_MON_addVariableToMonitor(&cual, ACE_MON_FLOAT,   sizeof(cual),
    ↪ ACE_MON_ACCESS_WRITABLE,0,0);
    ACE_MON_addVariableToMonitor(&leer, ACE_MON_FLOAT,   sizeof(leer),
    ↪ ACE_MON_ACCESS_WRITABLE,0,0);
    ACE_MON_addVariableToMonitor(&a_ref, ACE_MON_FLOAT,   sizeof(a_ref),
    ↪ ACE_MON_ACCESS_WRITABLE,0,0);

    while(1)
    {
        ACE_MON_tickBackground();
    }
}
/*****************************************************************************/
extern void ADC_fpgaSyncInterrupt_isr(void){
    //int var_temp_int;
    /* Clear flag that caused the interruption */
    INT0CLRFLG = INT0_ADC0_MASK; // Borra el Flag sobreescribiendo un 1
```

```
    /*-------------------------------------------------------------------------*/
 // comienzo código offsets:---------------------------------
   if (a_ref==0){
        contador=0;
        ia_zero = 0;
        ib_zero = 0;
        ic_zero = 0;
        vc1a_zero = 0;
        vc2a_zero = 0;
        Vdc_zero = 0;
        a_ref=1;
    }
    contador=contador+1;

    if (contador>1000){
            contador=3000;}
    if (contador<=1000){

        ia_zero = ia_zero+((int)Med_ADC0_A1_16 *
        ↪  ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_ia ;
        ib_zero = ib_zero+((int)Med_ADC0_A2_16 *
        ↪  ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_ib;
        ic_zero = ic_zero+((int)Med_ADC0_B1_16 *
        ↪  ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_ic;
        vc1a_zero = vc1a_zero+((int)Med_ADC0_B2_16 *
        ↪  ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_v1;
        vc2a_zero = vc2a_zero+((int)Med_ADC1_A1_16 *
        ↪  ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_v2;
        Vdc_zero =
        ↪  Vdc_zero+((int)Med_ADC2_A2_16*ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_vdc;
    }
    else{
        offset_i_a=-ia_zero*0.001;
        offset_i_b=-ib_zero*0.001;
        offset_i_c=-ic_zero*0.001;
        offset_v1=-vc1a_zero*0.001;
        offset_v2=-vc2a_zero*0.001;
        offset_vdc=-Vdc_zero*0.001;
    }
//fin código offsets-------------------------

 //----- cuando hace el cambio de referencia:
    tolerance=1e-6;
    if((leer==1)&&((array_i>=0.5*array_length)&&(fabs(ia_r1 - a_ref1) <
    ↪  tolerance))){
            a_ref1=a_ref;
    }
 //-------------------------------------------
    t = t + h;
    t1 = t - h;
    t1x = t + 9*h;

    i1_r=cossp(pi100*t);
    i2_r=cossp(pi100*t-x2_3*pi);
```

```
    i3_r=cossp(pi100*t+x2_3*pi);

    ia_r1 = a_ref1*cossp(pi100*t1);
    ib_r1 = a_ref1*cossp(pi100*t1 - x2_3*pi);
    ic_r1 = a_ref1*cossp(pi100*t1 + x2_3*pi);
    //nueva referencia:
    ia_r1x = a_ref1*cossp(pi100*t1x);
    ib_r1x = a_ref1*cossp(pi100*t1x - x2_3*pi);
    ic_r1x = a_ref1*cossp(pi100*t1x + x2_3*pi);

    Med_ADC0_A1_16 =  ADC0CHA1;
    Med_ADC0_A2_16 =  ADC0CHA2;
    Med_ADC0_B1_16 =  ADC0CHB1;
    Med_ADC0_B2_16 =  ADC0CHB2;

    ia_m = -(((int)Med_ADC0_A1_16 *
    ↪  ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_ia + offset_i_a);
    ib_m = -(((int)Med_ADC0_A2_16 *
    ↪  ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_ib + offset_i_b);
    ic_m = -(((int)Med_ADC0_B1_16 *
    ↪  ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_ic + offset_i_c);
    vc1a_m = ((int)Med_ADC0_B2_16 *
    ↪  ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_v1 + offset_v1;

    Med_ADC1_A1_16 =  ADC1CHA1;
    Med_ADC1_A2_16 =  ADC1CHA2;
    Med_ADC1_B1_16 =  ADC1CHB1;
    Med_ADC1_B2_16 =  ADC1CHB2;

    vc2a_m = ((int)Med_ADC1_A1_16 *
    ↪  ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_v2 + offset_v2;
    vc1b_m = ((int)Med_ADC1_A2_16 *
    ↪  ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_v1 + offset_v1;
    vc2b_m = ((int)Med_ADC1_B1_16 *
    ↪  ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_v2 + offset_v2;
    vc1c_m = ((int)Med_ADC1_B2_16 *
    ↪  ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_v1 + offset_v1;

    Med_ADC2_A1_16 =  ADC2CHA1;
    Med_ADC2_A2_16 =  ADC2CHA2;

    vc2c_m =((int)Med_ADC2_A1_16 *
    ↪  ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_v2 + offset_v2;
    Vdc = ((int)Med_ADC2_A2_16*ACE_ADC_ESS_M10P10_CONVERSION_FACTOR)*gain_vdc +
    ↪  offset_vdc;

    Vdc_2=Vdc*0.5;
    Vdc_3=Vdc*0.3333333333333333;

    tick_flag = ACE_MON_tickForeground();

//------ORIGINAL---------
    if (leer==1){
        array0_A1[array_i] = ia_m;
```

```
        array0_A2[array_i] = ib_m;
        array0_B1[array_i] = ic_m;
        array0_B2[array_i] = vc1a_m;
        array1_A1[array_i] = vc2a_m;
        array1_A2[array_i] = vc1b_m;
        array1_B1[array_i] = vc2b_m;
        array1_B2[array_i] = vc1c_m;
        array2_A1[array_i] = vc2c_m;
        array2_A2[array_i] = ia_r1;
        array2_B1[array_i] = id_m;
        array2_B2[array_i] = iq_m;
        array3_A1[array_i] = Vdc;
        array3_A2[array_i] = auxiliar;

    /***********************************/
        array_i = array_i+1;

        if (array_i>= array_length){
            array_i = 0;
            leer=0;
        }
    }
   //-------------------------------
    // Cálculo de Teta
    alfa = i1_r;
    beta = (i2_r-i3_r) * raiz1_3;
    teta = fabsf(atanf(beta/alfa));

    if (alfa<0 && beta>=0)
        tetaf = pi - teta;
    else
        {
        if (alfa<0 && beta<0)
            tetaf = pi + teta;
        else
            {
            if (alfa>=0 && beta<0)
                tetaf = 2 * pi-teta;
            else
                if (alfa>=0 && beta>=0)
                    tetaf = teta;
            }
        }
    // Transformada ABC-DQ de corriente
    Da = cosf(tetaf) * x2_3;
    Db = cosf(tetaf-g120) * x2_3;
    Dc = cosf(tetaf+g120) * x2_3;

    Qa = -sinf(tetaf) * x2_3;
    Qb = -sinf(tetaf-g120) * x2_3;
    Qc = -sinf(tetaf+g120) * x2_3;

    /***************************************/
    // Corrientes en eje D y eje Q
```

```
id_m = Da*ia_m + Db*ib_m + Dc*ic_m;
iq_m = Qa*ia_m + Qb*ib_m + Qc*ic_m;
id_r = Da*ia_r1 + Db*ib_r1 + Dc*ic_r1;
iq_r = Qa*ia_r1 + Qb*ib_r1 + Qc*ic_r1;

//**************** PARTE LINEAL ************************************//
if( cual == 0)
{
//--------------------ANN PI: La tengo solo para probarla de
↪  referencia-------

input_p[0] = ia_r1;
input_p[1] = ia_m;
input_p[2] = ia_r1-ia_m;
input_p[3] = ib_r1;
input_p[4] = ib_m;
input_p[5] = ib_r1-ib_m;
input_p[6] = ic_r1;
input_p[7] = ic_m;
input_p[8] = ic_r1-ic_m;
input_p[9] = s1at;
input_p[10] = s1bt;
input_p[11] = s1ct;

for (i = 0; i < n_in_p; ++i)
{
    input2_p[i] = (2/(xmax_p[i]-xmin_p[i]))*(input_p[i] - xmin_p[i])-1;
}

for (i = 0; i < n_lyr1_p; i++) {
    suma = 0;
    for (j = 0; j < n_in_p; j++) {
        suma += iw_p[i][j] * input2_p[j];
    }
    outlayer1_p[i] = tanhsp(suma + ib_p[i][0]);
}

for (i = 0; i < n_lyr2_p; i++) {
    suma = 0;
    for (j = 0; j < n_lyr1_p; j++) {
        suma += ow_p[i][j] * outlayer1_p[j];
    }
    outlayer2_p[i] = suma + ob_p[i][0];
}

for (i = 0; i < n_lyr2_p; i++){
    output_p[i] = ((ymax_p[i]-ymin_p[i])/2)*(outlayer2_p[i] + 1 ) +
    ↪  ymin_p[i];
}
/*************************************/
mA = output_p[0];
mB = output_p[1];
mC = output_p[2];
```

```
s1atp=mA;
s1btp=mB;
s1ctp=mC;
}  //corchete de comienzo del PWM !!!
/*****************************************/

ia_r1d=ia_r1x*1.45;
ib_r1d=ib_r1x*1.3;
ic_r1d=ic_r1x*1.35;

ia_r1o=ia_r1*1.45;
ib_r1o=ib_r1*1.3;
ic_r1o=ic_r1*1.35;

if( cual == 1)
{
// MI ANN:
input[0] = ia_r1d;
input[1] = ia_m;
input[2] = ia_r1o-ia_m;
input[3] = ib_r1d;
input[4] = ib_m;
input[5] = ib_r1o-ib_m;
input[6] = ic_r1d;
input[7] = ic_m;
input[8] = ic_r1o-ic_m;
input[9] = s1at;
input[10] = s1bt;
input[11] = s1ct;

for (i = 0; i < n_in; i++)
{
    input2[i] = (2/(xmax[i]-xmin[i]))*(input[i] - xmin[i])-1;
}

for (i = 0; i < n_lyr1; i++) {
    suma = 0;
    for (j = 0; j < n_in; j++) {
        suma += iw[i][j] * input2[j];
    }
    //outlayer1[i] = 2/(1+exp(-2*(suma + ib[i][0])))-1;
    outlayer1[i] = tanhsp(suma + ib[i][0]);
}

for (i = 0; i < n_lyr2; i++) {
    suma = 0;
    for (j = 0; j < n_lyr1; j++) {
        suma += ow[i][j] * outlayer1[j];
    }
    outlayer2[i] = suma + ob[i][0];
}

for (i = 0; i < n_lyr2; i++){
    output[i] = ((ymax[i]-ymin[i])/2)*(outlayer2[i] + 1 ) + ymin[i];
```

```
    }
    /****************************************/
    mA = output[0];
    mB = output[1];
    mC = output[2];
    }
    s1_A=mA;
    s2_A=mA;
    s3_A=mA;
    s1_B=mB;
    s2_B=mB;
    s3_B=mB;
    s1_C=mC;
    s2_C=mC;
    s3_C=mC;
//---------------------------------------------------------------------
    // Salida de variables
    M0CMPR1 = s1_A * M0_MAX;
    M0CMPR2 = s1_B * M0_MAX;
    M0CMPR3 = s1_C * M0_MAX;

    M1CMPR1 = s2_A * M0_MAX;
    M1CMPR2 = s2_B * M0_MAX;
    M1CMPR3 = s2_C * M0_MAX;

    M2CMPR1 = s3_A * M0_MAX;
    M2CMPR2 = s3_B * M0_MAX;
    M2CMPR3 = s3_C * M0_MAX;

    auxiliar = a_ref;
    s1at = mA;
    s1bt = mB;
    s1ct = mC;

    M3CMPR1 = 0;
}
```