



Implementación de esquema de control de un convertidor Multicelda en Cascada para inyección de energía fotovoltaica a la red en plataforma de control BRAIn, incorporando pruebas en módulo HIL

DANIEL ELÍAS ARRIAGADA FARÍAS

2024

Requisito parcial para obtener el título de:
Ingeniero Civil Electricista

Profesor Guía:
Dr. Pablo Andrés Lezana Illesca (UTFSM)

Comisión:
Dr. Pablo Andrés Lezana Illesca (UTFSM)
Dr. Pablo Fernando Acuña Ríos (U. de Talca)

Valparaíso, 30 de Junio de 2024.



Implementación de esquema de control de un convertidor Multicelda en Cascada para inyección de energía fotovoltaica a la red en plataforma de control BRAIn, incorporando pruebas en módulo HIL

DANIEL ELÍAS ARRIAGADA FARÍAS

2024

*“No te rindas, por favor no cedas, aunque el frío queme, aunque el miedo muerda,
aunque el sol se esconda y se calle el viento, aún hay fuego en tu alma, aún hay vida en
tus sueños”*

— Mario Benedetti

Agradecimientos

Primero que todo quiero agradecer a mi abuelita Lucy, que me acompañó siempre muy de cerca, se interesó por los temas que me interesaban, escuchó cada palabra que con entusiasmo le transmitía, pero que lamentablemente no alcanzó a vivir este momento. Para mí ella fue, sinceramente, lo más cercano a un ángel que he conocido. Agradezco a mi abuelita María, de la que más que probablemente heredé mi perseverancia, y que siempre se asombró por cada pequeño paso que daba. Agradezco a mi madre, que, aunque sobre-protectora, siempre se preocupó por mí cada vez que toqué fondo, aunque a veces coincidía conmigo por la gran empatía con la que fue bendecida. Agradezco a mi padre, que siempre me impulsó hacia adelante, como si el sendero que tomé fuera simple, pero que siempre fue una inspiración para no bajar los brazos y terminar este difícil camino. Agradezco a mi tata, que siempre me ayudó en todo lo que necesitaba, quitando las pequeñas piedras que habían en el camino. Agradezco a todos los que no nombré: mi hermano, mis hermanas, mis tíos, mi tías, mi prima. Estos agradecimientos no están escritos en orden: cada uno de ustedes me apoyó a su manera y son una parte importante de mi formación para llegar a donde estoy. Para mí la suerte sí existe, y es tenerlos a ustedes.

Índice de Contenidos

Índice de Contenidos	I
Índice de Figuras	III
Índice de Tablas	IV
Resumen	1
1. Introducción	2
2. Control de un convertidor MC con esquema de control SHM-MPC.	5
2.1. Convertidor Multicelda en Cascada	5
2.1.1. Topología	5
2.1.2. Modulación	6
2.1.2.a. PS-PWM	6
2.1.2.b. SHE y SHM	8
2.2. Control SHE Predictivo	10
2.2.1. Control Predictivo	10
2.2.1.a. Cálculo de la corriente de referencia	11
2.2.1.b. Cálculo de la predicción de corriente en $k + 1$	11
2.2.1.c. Cálculo del voltaje de salida en estado estacionario	12
2.2.1.d. Lectura de la Tabla SHM Discretizada	13
2.2.1.e. Región óptima de operación	14
2.2.1.f. Optimización de la función de costo	16
2.2.2. PLL	17
3. Modelación Convertidor Multicelda en Cascada en FPGA.	20
4. Optimización de esquema de control en DSP	25
5. Ensayos en módulo HIL	37
5.1. Implementación de Convertidor Multicelda en Cascada en el FPGA . . .	37
5.2. Escalón magnitud de corriente de referencia	38
5.2.1. Simulación	38
5.2.2. Implementación	38
5.3. Escalón ángulo de corriente de referencia	41
5.3.1. Simulación	41
5.3.2. Implementación	41
5.4. Análisis de THD respecto de índice de modulación	41

5.4.1. Simulación	41
5.4.2. Implementación	44
6. Conclusiones.	47
Apéndices	49
A. Código esquema de control SHM-MPC, figuras y tablas	49
A.1. Code Composer	49
A.1.1. Archivo config.c	49
A.1.2. Archivo app.c	52
A.1.3. Archivo main.c	53
A.2. Gráficos	71
A.2.1. Gráfico Convertidor Multicelda en Cascada de 5 niveles	71
A.2.2. Gráfico Modulación PS-PWM Convertidor Multicelda en Cascada 5 niveles y tensión de salida.	72
A.2.3. Gráfico Modulación SHE Convertidor Multicelda 3 niveles.	73
A.2.4. Gráfico Algoritmo SHE	75
A.2.5. Gráfico Algoritmo SHE optimizado	77
A.2.6. Gráfico Ensayo escalón magnitud de corriente implementado	79
A.2.7. Gráfico Ensayo escalón magnitud de corriente simulado	81
A.2.8. Gráfico Ensayo escalón fase de corriente implementado	83
A.2.9. Gráfico Ensayo escalón fase de corriente simulado	85
A.2.10. Gráfico THD y espectro de frecuencia de corriente implementado	87
A.2.11. Gráfico THD y espectro de frecuencia de corriente simulado	94
A.2.12. Gráfico Vectores de tensión plano $\alpha\beta$ 7 niveles	99
A.2.13. Gráfico Sub-héxagono plano $\alpha\beta$ 7 niveles	100
A.2.14. Ángulos patrón SHM	101
A.2.15. Comparación hexágonos	101
A.2.16. Algoritmo forma SHM	103
A.3. Tablas	105
A.3.1. Tensión de salida	105
A.3.2. Tensión v_{an}	106
A.3.3. Tabla lógica XOR	106
A.3.4. Parámetros	107
Bibliografía	108

Índice de Figuras

1.1.	Paneles fotovoltaicos junto con topología CHB.	3
1.2.	DSP y FPGA.	4
2.1.	Convertidor Multicelda en Cascada de 5 niveles.	6
2.2.	Convertidor Multicelda en Cascada de $2x+1$ niveles.	7
2.3.	Modulación PS-PWM Convertidor Multicelda en Cascada 5 niveles. . . .	8
2.4.	Tensión de salida Convertidor Multicelda en Cascada de 5 niveles. . . .	8
2.5.	Modulación SHE Convertidor Multicelda 3 niveles.	9
2.6.	Circuito fase “a”.	12
2.7.	Límites actuales de distorsión para sistemas con una tensión nominal de 120V a 69kV.	13
2.8.	Algoritmo forma SHM.	14
2.9.	Vectores de tensión plano $\alpha\beta$ 7 niveles.	15
2.10.	Sub-héxagono plano $\alpha\beta$ 7 niveles.	16
2.11.	Esquema de control PLL.	18
3.1.	Paneles fotovoltaicos junto con topología CHB.	21
3.2.	Circuito fase “x”.	22
3.3.	Conversión medición de corriente.	23
3.4.	Conversión medición de tensión.	24
4.1.	Ángulos patrón SHM.	26
4.2.	Algoritmo SHE.	30
4.3.	Algoritmo SHE optimizado.	32
4.4.	(a) Héxagono tradicional; (b) Héxagono modificado para trabajar con números enteros.	34
5.1.	Ensayo escalón magnitud de corriente simulado.	39
5.2.	Ensayo escalón magnitud de corriente implementado.	40
5.3.	Ensayo escalón fase de corriente simulado.	42
5.4.	Ensayo escalón fase de corriente implementado.	43
5.5.	Espectro de frecuencia $i_{g.a.k}$ ($m.i. = 0,41$).	44
5.6.	Espectro de frecuencia $i_{g.a.k}$ - Vista Ampliada ($m.i. = 0,41$).	44
5.7.	THD de corriente vs índice de modulación.	45
5.8.	Espectro de frecuencia $i_{g.a.k}$ ($m.i. = 0,41$).	45
5.9.	Espectro de frecuencia $i_{g.a.k}$ - Vista Ampliada ($m.i. = 0,41$).	45
5.10.	THD de corriente vs índice de modulación.	45

Índice de Tablas

3.1.	Tensión de salida	21
3.2.	Tensión v_{an}	22
4.1.	Tabla lógica XOR	32
4.2.	Optimización	35
5.1.	Parámetros	37

Nomenclatura.

x	: Valor temporal o variable de la cantidad x .
X	: Valor medio de la cantidad x .
x^*	: Valor de referencia para la cantidad x .
x_y	: Valor de la cantidad x referida al secundario.
x_Y	: Valor de la cantidad x referida al primario.
x_s	: Variable referida a la entrada, p. ej.: ω_s frecuencia de entrada.
x_o	: Variable referida a la salida, p. ej.: ω_o frecuencia de salida.
a, b, c	: Sub-índices utilizados para variables de entrada.
u, v, w	: Sub-índices utilizados para variables de salida.
N	: Neutro de la alimentación.
n	: Neutro de la carga.
o	: Neutro del inversor.

Resumen

Se realizó un estudio en la Universidad de Talca con el objetivo de determinar una alternativa de modulación para Convertidores Multicelda en Cascada. Este estudio desarrolló una técnica de modulación llamada SHM-MPC (Selective Harmonic Mitigation Model Predictive Control), basada en la modulación SHE (Selective Harmonic Elimination), SHM (Selective Harmonic Mitigation) y en la técnica FCS-MPC (Finite Control Set Model Predictive Control). Esta técnica de modulación permite tener una baja frecuencia de conmutación, lo cual es favorable para aplicaciones de alta potencia. Además, mediante su forma específica de conmutación permite eliminar componentes de bajo orden.

El problema de esta técnica es su alto costo computacional, afectando la calidad de los datos obtenidos en la práctica, haciendo que hasta ahora solo haya podido ser implementada para una celda por fase.

En este trabajo lo que se buscó fue implementar este esquema de control desarrollado en una plataforma de control digital llamada BRAIn (Board for Research Academic and Industrial Applications) desarrollada en el Centro Avanzado de Ingeniería Eléctrica y Electrónica (AC3E), mediante la optimización de este esquema. La optimización se realizó mediante un conteo de los ciclos de reloj del software Code Composer, a través del cual se controlaba el DSP de la tarjeta. Además, se implementó un modelo en el FPGA de la plataforma BRAIn que buscó emular un Convertidor Multicelda en Cascada de 3 celdas por fase junto con un modelo de la línea y de la red. Esto permitió transformar la plataforma de forma que se pudiera trabajar en HIL (Hardware in the Loop).

El objetivo planteado se logró satisfactoriamente luego de la propuesta de distintas alternativas a las distintas secciones del código, principalmente en las secciones del algoritmo SHE y en el FCS-MPC utilizado para obtener la mejor configuración de conmutación para el convertidor. La implementación se verificó mediante el módulo HIL mencionado anteriormente a través de una comparación de resultados respecto a una simulación en MATLAB-Simulink. Un punto importante que permitió la implementación fue la actualización de los registros de configuración de la plataforma, lo que permitió un aumento de la frecuencia de reloj del DSP de 300 MHz a 456 MHz.

Capítulo 1

Introducción

En el contexto de la incorporación exponencial de las Energías Renovables No Convencionales (ERNC), los convertidores para aplicación de energía solar fotovoltaica se han vuelto muy importantes. Debido a este avance, cada vez existe más exigencia en cuanto al diseño y control de los convertidores.

Los convertidores multicelda surgen como una opción atractiva a la hora de aumentar el nivel de potencia con la que trabajan los sistemas ERNC. Lo anterior se debe a la posibilidad de trabajar con mayores tensiones, y, además, una mayor cantidad de niveles. Mientras más niveles tenga este convertidor, mayor es la disminución de armónicos de voltaje y corriente, y, por lo tanto, mejor es la calidad de energía. De las topologías multicelda existentes, la topología Cascaded [1] H-Bridge (CHB) [4,5,7] resulta interesante para trabajar con sistemas fotovoltaicos, dado que esta topología requiere de fuentes de tensión de corriente continua aisladas entre sí. De manera natural cada panel (o conjunto de paneles) cumplen con esta condición, tal como se muestra en la Figura 1.1.

Para adaptarse a los nuevos requerimientos de las plantas fotovoltaicas, como la alta potencia y eficiencia (sobre el 98 %) [9], se han desarrollado nuevas técnicas de modulación. Entre estas se encuentran Phase-Shifted PWM (PS-PWM) para topologías multicelda como CHB y Flying Capacitor Converter (FCC), o Space Vector Modulation (SVM). Estas técnicas generan corrientes de gran calidad, pero al conmutar a alta frecuencia generan pérdidas que pueden ser prohibitivas para aplicaciones de alta potencia. Para aplicaciones que requieren baja frecuencia de conmutación (y por ende menores pérdidas de conmutación), una de las técnicas de modulación utilizada es Selective Harmonic Elimination (SHE), que utiliza unas pocas conmutaciones en ángulos específicos con el fin de generar la componente fundamental deseada y a su vez eliminar ciertos armónicos específicos de bajo orden.

Como alternativa a la modulación SHE, se ha desarrollado la técnica llamada Selective Harmonic Mitigation (SHM), que tiene como objetivo reducir los armónicos en lugar de eliminarlos por completo. Esta técnica permite ajustar convenientemente los armónicos de acuerdo con la norma IEEE Std. 519-2022, que es un estándar que establece los objetivos y los límites de distorsión armónica para el diseño de sistemas eléctricos que incluyen cargas lineales y no lineales para aplicaciones desde los 120V hasta por sobre los 161kV.

Un problema de los esquemas SHE y SHM es su baja dinámica, por lo que en conjunto con el profesor Pablo Acuña de la Universidad de Talca, se ha desarrollado un esquema de

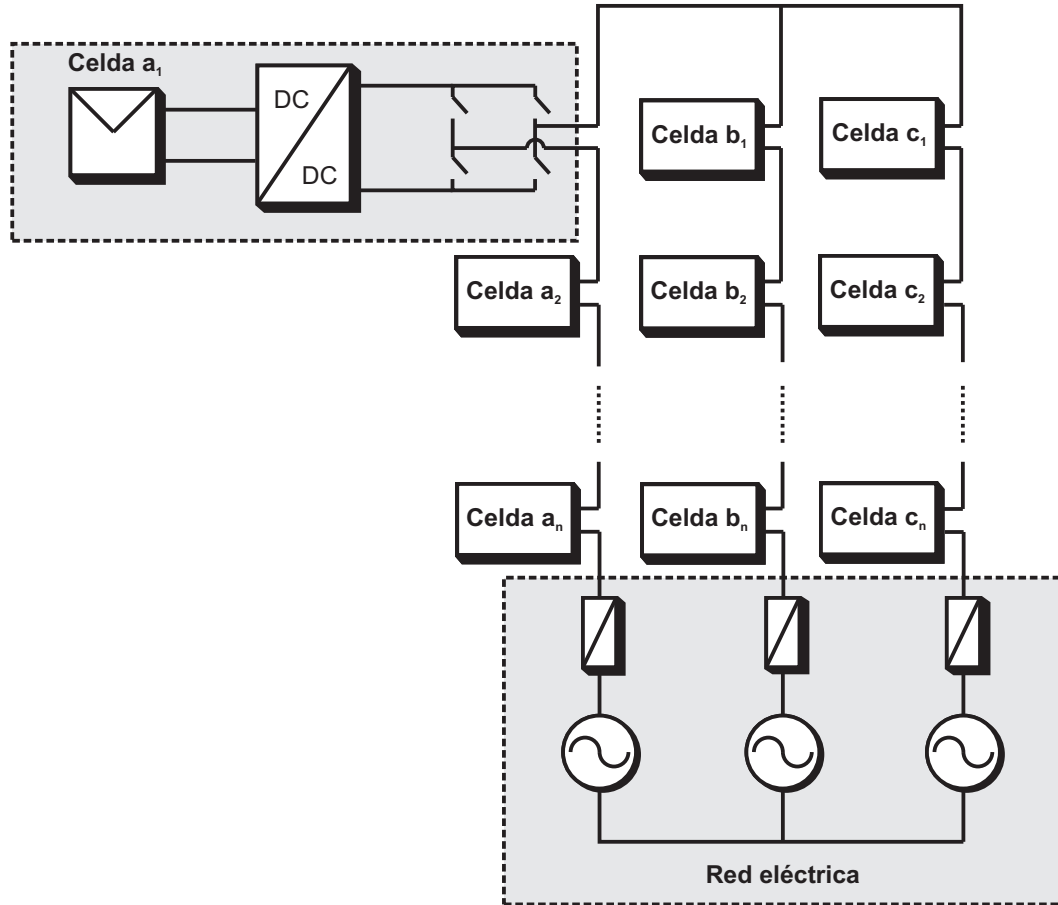


Figura 1.1. Paneles fotovoltaicos junto con topología CHB.

control predictivo basado en modelo (MPC) que permita mejorar los resultados dinámicos sin alterar de manera significativa los patrones SHE o MPC precalculados. Esta técnica se ha denominado SHE-MPC [8]. Un inconveniente de esta técnica es el alto costo computacional, lo que limita su frecuencia de muestreo y, por tanto, la calidad de los resultados obtenidos hasta ahora, ya que sólo se ha podido implementar para sistemas con una celda por fase.

En el Centro Avanzado de ingeniería eléctrica y electrónica (AC3E) de nuestra Universidad, se ha desarrollado una plataforma digital de control llamada BRAIn (Board for Research Academic and Industrial Applications). Esta tarjeta utiliza la electrónica digital para hacer un procesamiento de señales en tiempo real. Su diseño permite una óptima flexibilidad al dar acceso completo a diversos componentes clave. También es compatible con tarjetas comerciales estándar y con diseños a medida. Esta plataforma altamente configurable tiene dos circuitos integrados de procesamiento para facilitar el desarrollo rápido de esquemas de control complejos: DSP y FPGA [6].

El DSP suele hacer los cálculos para determinar la mejor actuación según un esquema de control definido y lo envía a la FPGA, en la que se han programado varios moduladores por ancho de pulsos (PWM). Además, la FPGA se ocupa de gestionar el funcionamiento de los convertidores análogo/digitales y de otros periféricos que pueden ser útiles para la aplicación.

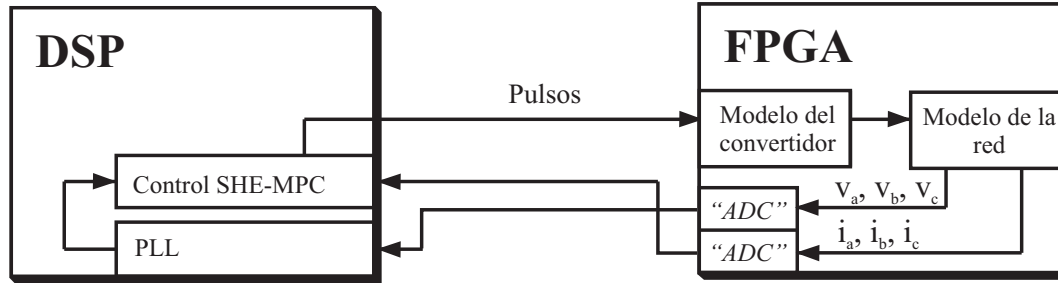


Figura 1.2. *DSP y FPGA.*

El objetivo de este trabajo es implementar y optimizar la técnica de modulación SHE-MPC en la plataforma BRAIn trabajando en conjunto con un estudiante de doctorado de la Universidad de Talca.

Para ello, el enfoque principal de este trabajo se centra en la integración de la técnica SHE-MPC en el DSP de la plataforma BRAIn. Para lograrlo, se optimizará el código desarrollado, disminuyendo los ciclos de reloj con el fin de poder aumentar la frecuencia de muestreo del esquema y así poder integrar más celdas al convertidor.

Para comprobar el buen funcionamiento del esquema de control se implementará un modelo del Convertidor Multicelda en Cascada en el FPGA de la plataforma BRAIn, utilizándola como un sistema HIL (Hardware in the Loop). Este modelo funciona a una alta velocidad (varios Megahertz), por lo que el sistema se ejecuta en tiempo real desde el punto de vista del DSP. El DSP ejecutará el sistema de control, enviando los pulsos de disparo a la FPGA. Estos pulsos llegarán al modelo del convertidor programado en la FPGA, los que se utilizarán para calcular los valores de las corrientes de la red y los almacenará en registros similares a los de los conversores AD. Así, cuando el DSP solicite un valor de medición, se le entregará un valor de corriente calculada en lugar de uno real, tal como se muestra en la Figura 1.2.

Una vez que la implementación en el DSP de la plataforma BRAIn y en el modelo del Convertidor Multicelda en Cascada en el FPGA estén completos, se procederá a realizar una serie de ensayos en el módulo HIL. Estos ensayos tienen como objetivo verificar y validar el correcto funcionamiento de todo el sistema implementado.

Capítulo 2

Control de un convertidor MC con esquema de control SHM-MPC.

2.1. Convertidor Multicelda en Cascada

2.1.1. Topología

La unidad básica de construcción de un Convertidor Multicelda en Cascada (CHB) se llama puente H (H-Bridge). Este tipo de convertidores están compuestos por varios puentes H dispuestos en serie (cascada), los cuales se pueden controlar de manera independiente. Cada uno de estos puentes H puede generar tensiones v_{cc} , 0 o $-v_{cc}$, los cuales al estar en serie se suman para generar distintos niveles de tensión. Este nivel de modularidad simplifica tanto la modulación como la protección del sistema, debido a que si un módulo falla es posible desconectarlo sin afectar a los demás módulos. En general, mientras más celdas en serie mejor el comportamiento de los armónicos dado que esto provoca una mejor aproximación a una sinusoidal, además esto permite reducir la frecuencia de conmutación de las celdas, reduciendo las pérdidas.

Los puentes H pueden funcionar ya sea mediante transistores IGBT's como por MOSFET's, ambos con diodo antiparalelo. La elección entre ambos tipos de transistores depende en mayor medida de la frecuencia de conmutación necesaria para el esquema de control y la tensión del dc-link. Existe una relación directa entre la cantidad de celdas de este tipo de convertidor con la cantidad máxima de niveles de modulación. Esta relación corresponde a $L = 2n + 1$, donde n corresponde al número de celdas del convertidor por fase y L la cantidad máxima de niveles posibles de modulación. La cantidad de niveles a la salida dependen de la modulación y el índice de modulación.

En la figura 2.1 se muestra un ejemplo de un Convertidor Multicelda en Cascada de cinco niveles con una modulación cualquiera, con el objetivo de mostrar el funcionamiento en cascada.

Las tensiones v_1 , v_2 y v_0 son generadas a partir de la transformación de las tensiones continuas v_{cc} , mediante el manejo del arreglo de MOSFET's del puente H a través de las señales S_{11} , S_{13} , S_{21} y S_{23} . Observando los gráficos de la derecha en la figura 2.1 se pueden contar los niveles de tensión de v_1 y v_2 , los cuales claramente son tres. Estas tensiones, tal como se comentó anteriormente, se suman, dado que están en serie, y generan la tensión

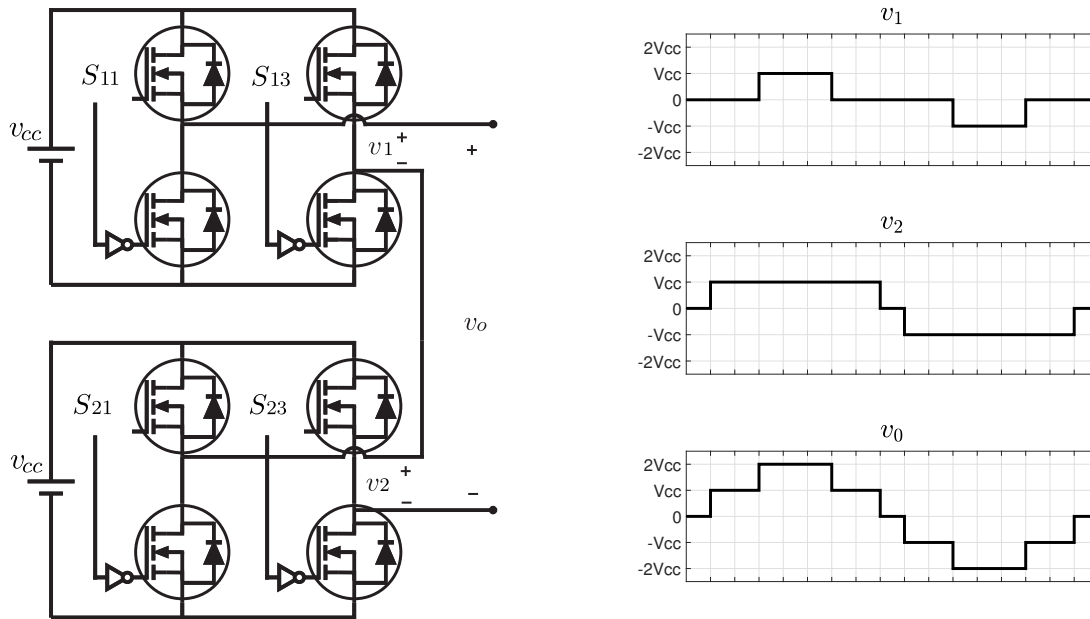


Figura 2.1. Convertidor Multicelda en Cascada de 5 niveles.

v_o , la cual posee cinco niveles que se pueden verificar al examinar la figura. Este ejemplo muestra claramente la relación detallada anteriormente, dado que para el caso de un solo convertidor se tiene $2 \cdot 1 + 1 = 3$ niveles de conmutación, y para el caso de dos convertidores en cascada se pueden tener hasta $2 \cdot 2 + 1 = 5$ niveles en serie.

En la figura 2.2 se observa el caso general de un Convertidor Multicelda en Cascada trifásico. Este convertidor está representado mediante “x” celdas en cascada por fase, conectadas en un nodo n común. Este convertidor trifásico está conectado a la red eléctrica a través de una línea con carga RL, únicamente con propósitos esquemáticos, dado que esta carga puede ser de cualquier tipo. En general, este tipo de convertidores se controlan mediante mediciones de la tensión y corriente, las cuales son muestreadas tanto en tiempos como en magnitudes discretas, y se hacen pasar por controles con distintos fines en específico.

En las siguientes secciones se explicará algunos tipos de modulación de este tipo de convertidor.

2.1.2. Modulación

2.1.2.a. PS-PWM

La técnica PWM (Pulse Width Modulation) es una técnica de modulación que permite controlar diversos parámetros de una carga a través de una variación en el ancho de los pulsos de una señal. Existen diferentes formas de adaptar la técnica PWM a un Convertidor Multicelda en Cascada. Una de las estrategias más comunes corresponde a la llamada PS-PWM, la que utiliza señales triangulares de alta frecuencia llamadas carrier, las cuales se comparan con una señal de referencia (típicamente sinusoidal) con el objetivo de tener

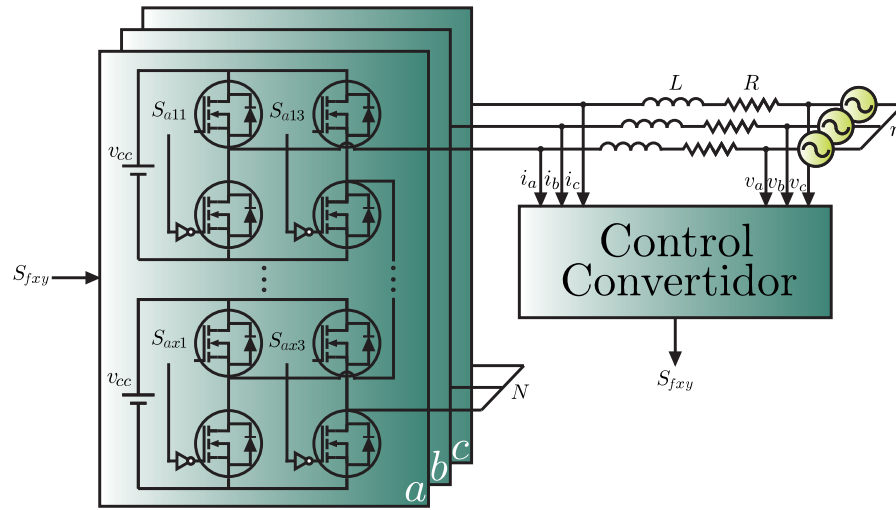


Figura 2.2. *Convertidor Multicelda en Cascada de $2x+1$ niveles.*

una tensión de salida lo más parecido a la forma de la señal sinusoidal.

Este tipo de modulación se genera a partir de señales carrier desplazadas según la cantidad de niveles del convertidor. Los niveles del convertidor tienen una relación directa con la cantidad de puentes H por fase, tal como se había comentado en la sección anterior. La cantidad de carriers requeridos dependen de la cantidad de niveles de la tensión total del convertidor por fase a través de la relación $L - 1$, por lo que si se tienen 5 niveles se requieren de 4 señales de carrier. Estas señales de carrier están desfasadas entre ellas según la relación $360^\circ / (L - 1)$. El desfase de estas señales de carrier se reparten entre las $2N$ piernas de las N celdas en cascada. Dentro de una misma celda las 2 señales de carrier repartidas en cada una de las piernas están desfasadas entre ellas por 180° , por lo que la primera mitad de las señales contiguas de carrier se generan en la pierna de la izquierda de cada una de las celdas y la segunda mitad de las señales contiguas de carrier se generan en la pierna de la derecha de cada una de las celdas.

La tensión de salida modulada por cada una de las fases tiene una parte de la magnitud total requerida en la carga. Por ejemplo, si se requiere V en un Convertidor Multicelda en Cascada de 5 niveles las tensiones de salida moduladas por cada una de las celdas deberán ser de $V/2$ dado que son 2 celdas en serie.

En la figura 2.3 se observa un ejemplo de un esquema del funcionamiento de una fase de un Convertidor Multicelda en Cascada de 5 niveles modulado a partir de la técnica de modulación PS-PWM. A la izquierda de la figura mencionada, se tiene un esquema de control básico de modulación PS-PWM en el cual, si bien pareciera que se utiliza una sola señal de carrier, esta solo es una referencia dado que a partir del desfase de 90° y la multiplicación por -1 se generan las 4 señales de carrier necesarias para el funcionamiento del convertidor, las cuales están desfasadas en 90° entre ellas. A la derecha de la figura se observa los gráficos de las señales de carrier para cada celda (v_{carrier} y $-v_{\text{carrier}}$), la señal de control sinusoidal para la comparación con las señales de carrier (v_{control}) y la tensión de salida generada a partir de esta modulación para la celda 1 y 2. La diferencia entre estas tensiones de salida es relativamente fácil de distinguir para este caso, considerando

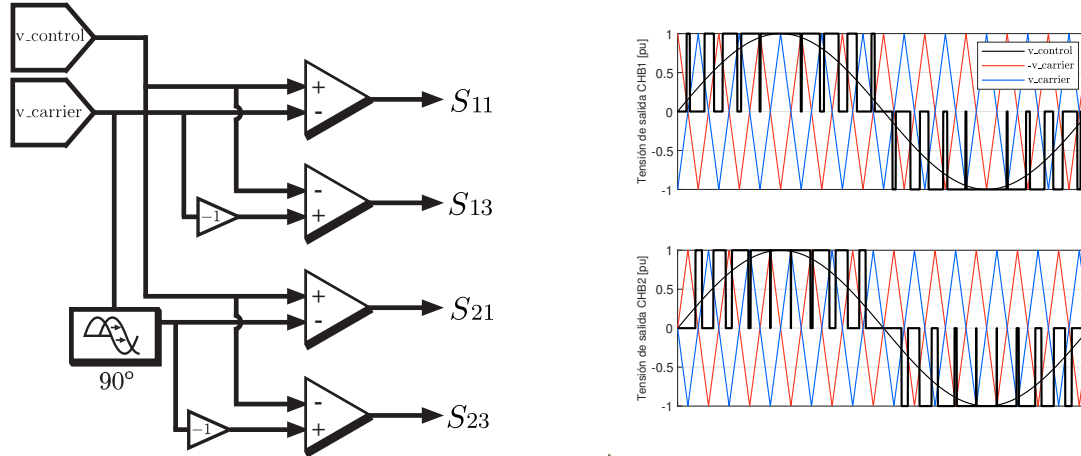


Figura 2.3. Modulación PS-PWM Convertidor Multicelda en Cascada 5 niveles.

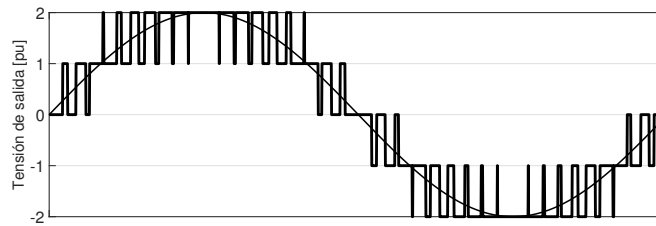


Figura 2.4. Tensión de salida Convertidor Multicelda en Cascada de 5 niveles.

la baja frecuencia de la señal de carrier con respecto a la señal de control sinusoidal, sin embargo, cuando se trata de altas frecuencias de la señal de carrier, lo cual es lo común, la diferencia entre las tensiones de salida es muy difícil de distinguir.

La figura 2.4 muestra la tensión de salida resultante del convertidor en p.u. con tensión base equivalente igual a la tensión continua v_{cc} que resultó de la modulación comentada anteriormente. Los armónicos de esta tensión resultante tienen la particularidad de encontrarse en una zona alrededor de $(L-1)f_s$ donde f_s es la frecuencia de la señal de carrier. Mientras mayor la cantidad de niveles, mayor la frecuencia en que se ubican los armónicos de alta frecuencia lo que facilita la eliminación de estos mediante un filtro pasabajos.

2.1.2.b. SHE y SHM

La técnica de modulación SHE (Selective Harmonic Elimination), como su nombre lo indica, es una metodología que permite seleccionar y eliminar armónicos específicos a través de la obtención de ángulos de conmutación. Esta técnica se basa en el método de conmutación llamado Staircase, por lo que tiene una forma bastante similar. Un ejemplo de como se vería este tipo de modulación para un Convertidor Multicelda en Cascada de 3 niveles se observa en la figura 2.5.

Donde para este caso específico se cuenta con 3 ángulos α_1 , α_2 y α_3 los cuales determinan la forma de la tensión de salida del convertidor. Se puede observar que esta tensión tiene una forma particular, sin embargo, no es muy distinta a una onda sinusoidal en

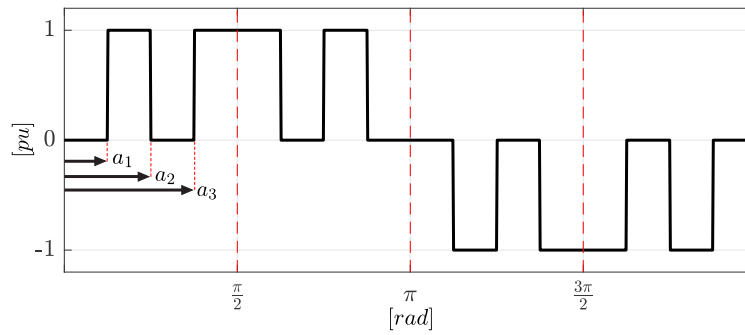


Figura 2.5. Modulación SHE Convertidor Multicelda 3 niveles.

términos de simetría. Dada la forma de la tensión se cumple que cada cuarto de onda de la señal es un reflejo tanto horizontal, vertical o ambas del primer cuarto de la señal, lo cual es ventajoso tanto en el análisis de la señal como en la forma de programar la señal a través de las señales que llegan a los semiconductores de potencia.

Una de las principales ventajas de este tipo de modulación es que mantiene un número fijo de conmutaciones por período, lo que corresponde al mínimo número de conmutaciones necesario. Esto resulta en pérdidas por conmutación reducidas al mínimo. La selección de los ángulos se realiza a partir de un análisis de la serie de Fourier de la señal donde posteriormente se obtiene un sistema de ecuaciones que permite calcular el valor de estos ángulos. Este cálculo se hace de manera *offline*, debido a su complejidad, por lo que no afecta el tiempo de ejecución del control. Sin embargo, el costo computacional de este cálculo aumenta a medida que se incrementa el número de niveles de voltaje de referencia. La eliminación de armónicos se realiza principalmente para armónicos de bajo orden, ya que los armónicos de mayor orden suelen ser filtrados debido a la característica pasabajos de la mayoría de las cargas.

De acuerdo con la representación de Fourier, toda señal periódica puede reescribirse como:

$$v(t) = a_0 + \sum_{n=1}^{\infty} [a_n \cos(n\omega t) + b_n \sin(n\omega t)]. \quad (2.1)$$

Donde el valor a_0 corresponde al valor medio de la señal $v(t)$ y a_n y b_n son coeficientes de la serie de Fourier para $v(t)$. Aquí, n representa el n -ésimo armónico, y ωt corresponde a la fase de la red.

Considerando que hay simetría impar en la forma de la señal, la componente de corriente continua y los armónicos pares son cero. Por lo que la representación de la tensión $v(t)$ en p.u. queda de la forma

$$v(t) = \sum_{n=1}^{\infty} \frac{a_n}{v_{cc}} \sin(n\omega t). \quad (2.2)$$

con

$$a_n = \frac{4}{n\pi} \sum_{k=1}^N (-1)^{k+1} \cos(n\alpha_k) \quad (2.3)$$

y N la cantidad de ángulos y α_k el k -ésimo ángulo de conmutación.

Al aplicar la serie de Fourier al caso específico de tres ángulos ($N = 3$), considerando la simetría se obtienen N ecuaciones linealmente independientes del tipo:

$$H_n = \frac{a_n}{v_{cc}} = \frac{4}{n\pi} [\cos(n\alpha_1) - \cos(n\alpha_2) + \cos(n\alpha_3)] \quad (2.4)$$

la ecuación relacionada con $n=1$ (fundamental) se iguala al voltaje deseado en la carga, mientras que las $N-1$ ecuaciones restantes pueden utilizarse para ajustar la magnitud de las armónicas a valores deseados. Por lo que, para eliminar los armónicos previamente seleccionados, es necesario igualar las ecuaciones correspondientes a cero y resolver el sistema resultante. Uno de los métodos más eficaces para abordar esta tarea es el método de Newton-Raphson, ampliamente utilizado en la resolución de sistemas de ecuaciones no lineales.

Una variante al método SHE es el Selective Harmonic Mitigation. En este caso, en lugar de igualar las $N-1$ ecuaciones restantes a 0, se busca que la magnitud de las armónicas se encuentre por debajo de un límite, típicamente fijado por alguna normativa como la IEEE Std. 519-2022.

Aunque los métodos SHE y SHM ofrecen numerosas ventajas, también presentan algunas desventajas que son importantes considerar.

Las dos desventajas principales de trabajar con el método SHE en un sistema de control en lazo cerrado son la lentitud de la respuesta dinámica y las dificultades relacionadas con la resolución de las ecuaciones no lineales. La lentitud se debe a que se debe mantener la simetría de la forma de onda SHE/SHM durante el periodo fundamental de la señal, lo que puede hacer que los ajustes sean lentos y menos reactivos a los cambios. Por otro lado, las ecuaciones no lineales del método SHE/SHM no pueden resolverse en línea garantizando la convergencia. Sin embargo, esta limitación puede mitigarse mediante el uso de look-up tables (LUT), que permiten calcular los valores de manera *offline* y almacenar las soluciones para diferentes condiciones, facilitando su aplicación en tiempo real [2].

2.2. Control SHE Predictivo

2.2.1. Control Predictivo

El MPC (Model Predictive Control) es un tipo de algoritmo de control retroalimentado que utiliza un modelo del sistema a controlar con el objetivo de hacer predicciones del comportamiento de éste. Este algoritmo se utiliza en una amplia gama de aplicaciones, destacándose su éxito en el control de convertidores. Una de las variantes de MPC aplicadas a convertidores es el SHM-MPC, mencionada brevemente al inicio de esta memoria. Esta variante combina la modulación SHE y SHM con la técnica FCS-MPC (Finite Control Set Model Predictive Control). A continuación, se describe el algoritmo SHM-MPC.

2.2.1.a. Cálculo de la corriente de referencia

Para determinar la corriente de referencia, primero se calcula $\omega t(k+n)$, el cual se define como:

$$\omega t(k+n) = \omega t(k) + n\omega_0 T_s(k) \quad (2.5)$$

En donde n corresponde al muestreo n -ésimo, ω_0 es la velocidad angular nominal ($2\pi f_0$, con $f_0 = 50$ [Hz]) y $T_s(k)$ es equivalente al periodo de muestreo. Para las predicciones de corriente en $k+1$ y $k+2$, este algoritmo utiliza $\omega t(k+1)$ y $\omega t(k+2)$, respectivamente.

A partir de esta definición y considerando un ángulo de referencia de corriente respecto a la red ϕ_i , se obtiene la ecuación que describe la fase de la corriente:

$$\omega_i t(k) = \omega t(k) + \phi_i \quad (2.6)$$

Por lo tanto, el vector de la corriente de referencia se describe como:

$$\begin{aligned} \mathbf{i}_g^*(k) &= [i_{ga}^*(k) \quad i_{gb}^*(k) \quad i_{gc}^*(k)] \\ &= I^* [\cos(\omega_i t(k)) \quad \cos(\omega_i t(k) - \frac{2\pi}{3}) \quad \cos(\omega_i t(k) + \frac{2\pi}{3})] \end{aligned} \quad (2.7)$$

Con I^* correspondiente a la magnitud de corriente de referencia.

2.2.1.b. Cálculo de la predicción de corriente en $k+1$

De forma adicional se calcula la predicción de la corriente en el instante $k+1$. Para esto, se obtiene la ecuación diferencial que describe la relación de tensiones y corrientes para la fase “a”:

$$v_{La}(t) = v_{aN}(t) - v_{Ra}(t) - v_{ga}(t) - v_{oN}(t) \quad (2.8)$$

$$\frac{di_{ga}(t)}{dt} = \frac{-R}{L} i_{ga}(t) + \frac{1}{L} (v_{aN}(t) - v_{ga}(t) - v_{oN}(t)) \quad (2.9)$$

Donde $v_{aN}(t)$, $v_{Ra}(t)$, $v_{La}(t)$, $v_{ga}(t)$ y v_{oN} son las tensiones a la salida del convertidor, de resistencia R e inductancia de línea L , de la red ac y de modo común respectivamente. La representación esquemática de estas tensiones se pueden observar en el circuito de la figura 2.6.

Luego, se utiliza Euler hacia adelante para aproximar la derivada de forma discreta. De esta forma se obtiene la predicción de corriente en el instante $(k+1)$:

$$i_{ga}(k+1) = \left(1 - \frac{RT_s(k)}{L}\right) i_{ga}(k) + \frac{T_s(k)}{L} (v_{aN}(k) - v_{ga}(k) - v_{oN}(t)) \quad (2.10)$$

Este proceso es reproducido para la fase “b”. En cuanto a la predicción de corriente para la fase “c” esta se obtiene fácilmente a partir de la relación

$$i_{gc}(k+1) = -i_{ga}(k+1) - i_{gb}(k+1). \quad (2.11)$$

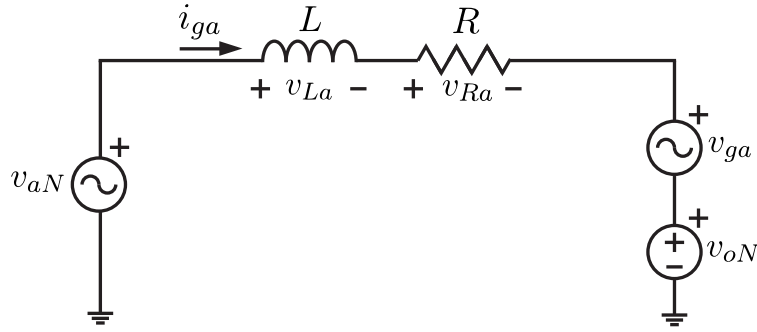


Figura 2.6. Circuito fase “a”.

2.2.1.c. Cálculo del voltaje de salida en estado estacionario

Esta sección del algoritmo está diseñada para obtener el módulo y la fase del voltaje de referencia trifásico en la salida del conversor en estado estacionario, denotado como $\mathbf{v}_N^*$. Dado que se trata de un valor de referencia, se utiliza la corriente de referencia $\mathbf{i}_g^*$ en el cálculo.

El procedimiento es desarrollado para la fase “a”. Sin embargo, en el algoritmo estos cálculos se deben realizar para todas las fases. Entonces, el primer paso consiste en reformular la expresión (2.8) en:

$$v_{aN}^*(k) = v_{Ra}^*(k) + v_{La}^*(k) + v_{ga}(k) \quad (2.12)$$

En donde:

$$v_{Ra}^*(k) = R i_{ga}^*(k) \quad (2.13)$$

$$v_{La}^*(k) = L \frac{di_{ga}^*}{dt} = -\omega L I^* \sin(\omega_i t(k)) \quad (2.14)$$

Por otro lado, $v_{ga}(k)$ corresponde a el valor muestreado de tensión de la red en la fase “a” en el instante k.

Cabe destacar que para obtener el valor de v_{aN} en $k + 1$, se realiza la aproximación:

$$v_{aN}(k + 1) \approx v_{aN}(k) \quad (2.15)$$

Luego de obtener la tensión de referencia del convertidor para todas las fases se obtiene el módulo mediante una transformación de un marco de referencia en el tiempo (abc) a uno rotatorio ($dq0$) mediante la expresión:

$$\begin{bmatrix} x_d(k) \\ x_q(k) \\ x_0(k) \end{bmatrix} = \frac{2}{3} \begin{bmatrix} \cos(\omega t(k)) & \cos(\omega t(k) - \frac{2\pi}{3}) & \cos(\omega t(k) + \frac{2\pi}{3}) \\ -\sin(\omega t(k)) & -\sin(\omega t(k) - \frac{2\pi}{3}) & -\sin(\omega t(k) + \frac{2\pi}{3}) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} x_a(k) \\ x_b(k) \\ x_c(k) \end{bmatrix} \quad (2.16)$$

Maximum harmonic current distortion in percent of I_L						
Individual harmonic order ^b						
I_{sc}/I_L	$2 \leq h < 11^a$	$11 \leq h < 17$	$17 \leq h < 23$	$23 \leq h < 35$	$35 \leq h \leq 50$	TDD
$< 20^c$	4.0	2.0	1.5	0.6	0.3	5.0
$20 < 50$	7.0	3.5	2.5	1.0	0.5	8.0
$50 < 100$	10.0	4.5	4.0	1.5	0.7	12.0
$100 < 1000$	12.0	5.5	5.0	2.0	1.0	15.0
> 1000	15.0	7.0	6.0	2.5	1.4	20.0

Figura 2.7. Límites actuales de distorsión para sistemas con una tensión nominal de 120V a 69kV.

En donde $\mathbf{x}_{abc}(k) = [x_a(k) \ x_b(k) \ x_c(k)]^T$ corresponde a un vector de una señal trifásica abc y $\mathbf{x}_{dq0} = [x_d(k) \ x_q(k) \ x_0(k)]^T$ corresponde al mismo vector transformado a un marco de referencia rotatorio.

Entonces, para obtener el módulo del voltaje v_{aN} se usa la expresión:

$$|v_N^*(k+1)| = \sqrt{v_{Nd}^{*2}(k+1) + v_{Nq}^{*2}(k+1)} \quad (2.17)$$

Como también se obtiene la fase para este voltaje de referencia:

$$\delta(k+1) = \arctan\left(\frac{v_{Nq}^*(k+1)}{v_{Nd}^*(k+1)}\right) \quad (2.18)$$

Es importante señalar que la tensión de modo común no aparece en la ecuación (2.12) como en la ecuación para la predicción de corriente (2.8). Esto se debe a que, tal como se vio en esta sección, la solución en estado estacionario es calculada en el plano $\alpha\beta$. Por lo tanto, el voltaje de nodo común corresponde a la componente $x_0(k)$, la cual no es considerada para el cálculo de la tensión en estado estacionario.

2.2.1.d. Lectura de la Tabla SHM Discretizada

La Tabla SHM contiene distintos ángulos en función del índice de modulación. Estos ángulos están determinados a partir de la norma IEEE Std. 519-2022. En la figura 2.7 [3] se observa la tabla de la norma recién mencionada que establece las restricciones para la construcción de la Tabla SHM.

Una vez obtenida la amplitud y la fase del voltaje de referencia $\mathbf{v}_N$, el siguiente paso consiste en obtener el índice de modulación m^* utilizando la siguiente fórmula:

$$m^*(k+1) = \frac{|v_N^*(k+1)|}{v_{cc}} \quad (2.19)$$

A continuación, se define una nueva referencia para la fase calculada para la lectura de la Tabla SHM:

$$\omega_{v_N} t(k+1) = \omega t(k+1) + \delta(k+1) \quad (2.20)$$

Con estas nuevas variables es posible formular el voltaje de referencia ($\mathbf{v}_{\text{SHM}}^*(k+1) = [v_{\text{SHM}_a}^*(k+1) \ v_{\text{SHM}_b}^*(k+1) \ v_{\text{SHM}_c}^*(k+1)]$) que es usada en la lectura de la tabla del patrón SHM. Esta corresponde a:

$$\mathbf{v}_{\text{SHM}}^*(k+1) = m^* \begin{bmatrix} \cos(\omega_{v_N} t(k+1)) & \cos(\omega_{v_N} t(k+1) - \frac{2\pi}{3}) & \cos(\omega_{v_N} t(k+1) + \frac{2\pi}{3}) \end{bmatrix} \quad (2.21)$$

El proceso de lectura contempla una comparación entre $\omega_{v_N} t(k+1)$, variando desde 0 a 2π , y los ángulos leídos de una tabla SHM calculada de forma *offline* para un cierto índice de modulación $m^*(k+1)$. El resultado de esta comparación corresponde a una referencia del nivel de voltaje $\mathbf{v}_{\text{SHM}}^*$ en el instante $k+1$. Un ejemplo de esto para el primer cuarto de onda con un índice de modulación $m = 0,82$ se puede observar en la figura 2.8.

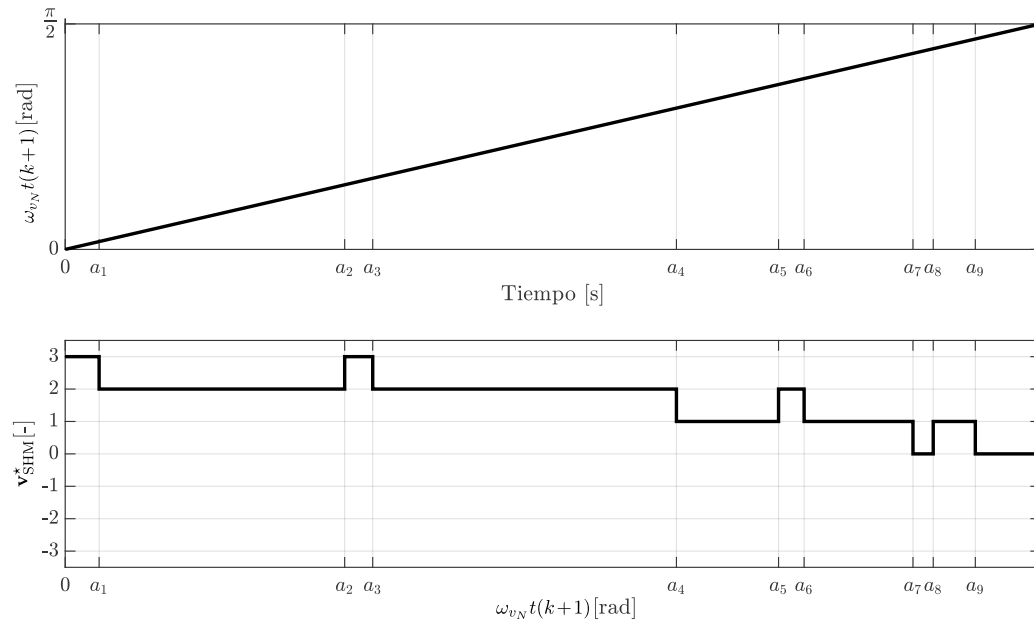


Figura 2.8. Algoritmo forma SHM.

2.2.1.e. Región óptima de operación

Esta etapa del algoritmo se utiliza el vector $\mathbf{v}_{\text{SHM}_N}^*(k+1)$ para obtener una representación en el plano imaginario ($\alpha\beta$) de esta referencia. Para llevar cada uno de los L^3 vectores al plano $\alpha\beta$ se utiliza la transformada de Clarke:

$$\begin{bmatrix} x_\alpha \\ x_\beta \\ x_\gamma \end{bmatrix} = \frac{2}{3} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} x_a \\ x_b \\ x_c \end{bmatrix} \quad (2.22)$$

Con $\mathbf{x}_{\alpha\beta\gamma} = [x_\alpha \ x_\beta \ x_\gamma]^T$ como un vector con sus componentes en un marco de referencia estacionario ($\alpha\beta$) y $\mathbf{x}_{abc} = [x_a \ x_b \ x_c]^T$ corresponde al mismo vector pero en un marco de referencia trifásico abc . La ventaja de usar este plano es que permite distinguir los

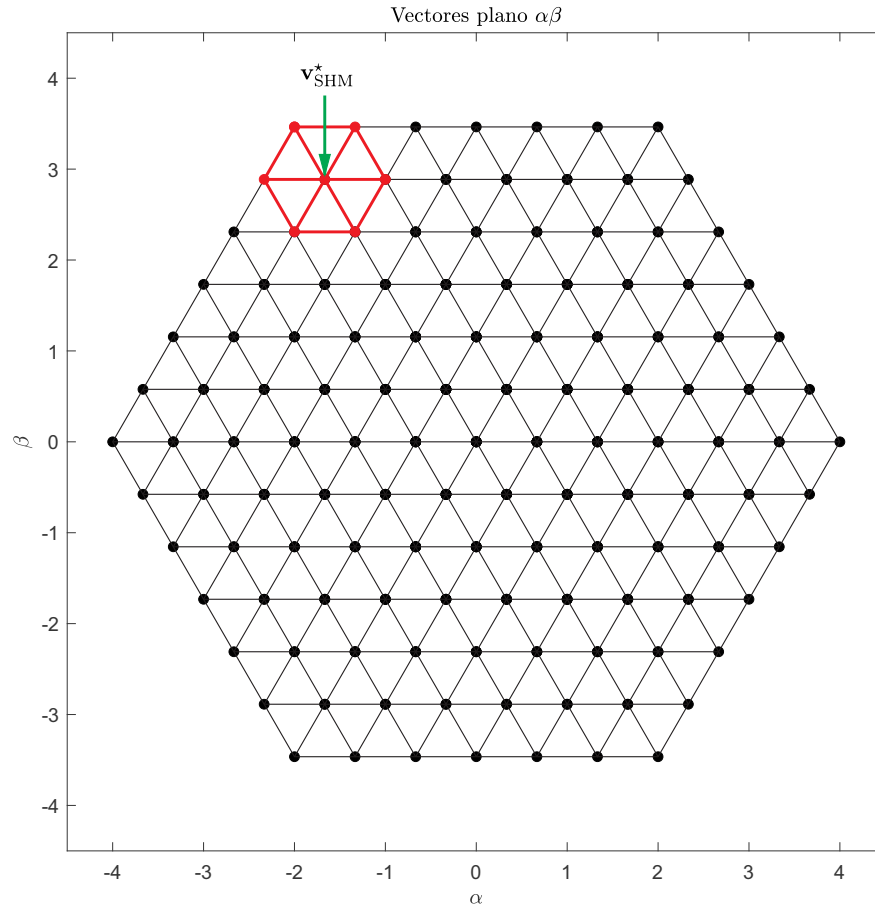


Figura 2.9. Vectores de tensión plano $\alpha\beta$ 7 niveles.

vectores de voltaje que son redundantes y que, por lo tanto, pueden ser descartables en la evaluación de la función de costo.

La propuesta de la reducción de costo computacional es seleccionar una región óptima en estado estacionario cuyo centro es el vector $\mathbf{v}_{\text{SHM}}^*(k+1)$. Además de este vector se seleccionan los vectores adyacentes que forman un sub-hexágono. Ahora bien, dentro de este sub-hexágono que tiene como centro el vector de referencia SHM, solo se requiere evaluar los vectores de voltaje que permiten una única transición de estado entre el voltaje de referencia, que se encuentra al centro, y los que se encuentran en el vértice del sub-hexágono. Así, un algoritmo de preselección permite reducir el conjunto de soluciones, utilizado en la optimización de la función de costo, de L^3 (L la cantidad de niveles del convertidor)(considerando todos los estados conmutación) a solo siete (el vector de voltaje de referencia central y los seis que conforman el sub-hexágono).

Un ejemplo de esto con $L = 7$ se observa en la figura 2.9.

En este caso los $L^3 = 343$ vectores posibles en el plano $\alpha\beta$ se traducen en 127 vectores

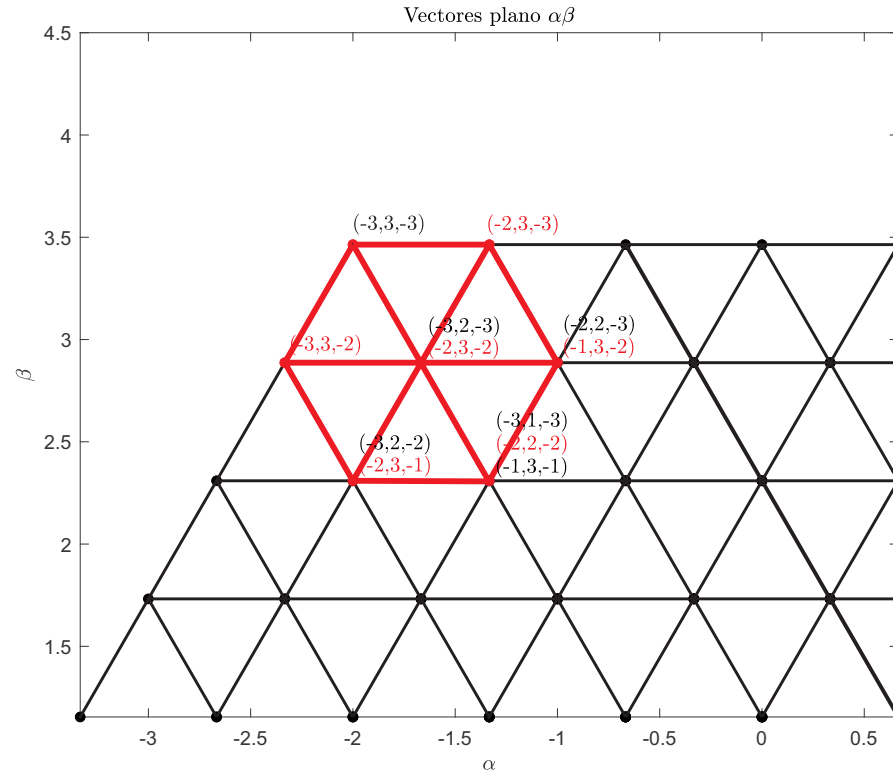


Figura 2.10. Sub-héxagono plano $\alpha\beta$ 7 niveles.

posibles en el plano $\alpha\beta$. Un ejemplo de un sub-héxagono para la combinación $(-2,3,-2)$ está demarcado en rojo. Este sub-héxagono se observa con más detalle en la figura 2.10. En este caso el centro del sub-héxagono viene dado por los vectores en el plano abc $(-3,2,-3)$ o $(-2,3,-2)$. Para este caso en particular se observa 6 posibles vectores que se evaluarán en la función de costo, considerando una única transición de estado. En algoritmo se analiza siempre para 7 casos. Por lo tanto, para este caso en particular, se repite el caso $(-3,3,-2)$ para completar los 7.

Cabe destacar que en esta selección de la región óptima no se realiza ninguna transformación de los vectores de voltaje ya que este trabajo es realizado de forma *offline*.

Ahora, en el algoritmo solo se busca, por medio de comparaciones, en cuál de los L^3 hexágonos se encuentra en el centro el voltaje SHM de referencia para posteriormente seleccionar los siete vectores que se evaluarán en la parte del FCS-MPC.

2.2.1.f. Optimización de la función de costo

En esta parte se busca minimizar la función de costo basada en operaciones con normas euclidianas, la cual está separada en tres partes. A continuación se definirán las partes de la función de costo en términos de la fase “a”, dado que el proceso se repite para las otras fases.

La primera parte de la función de costo consiste en la reducción del costo de corriente en el tiempo $k+2$

$$J_{i_a(k+2)} = \|i_{ga}(k+2) - i_{ga}^*(k+2)\|_2^2 \quad (2.23)$$

Donde $i_{ga}(k+2)$ se obtiene de forma análoga a como se obtuvo $i_{ga}(k+1)$.

La segunda parte de la función de costo consiste en la expresión que permite el seguimiento del patrón SHM según:

$$J_{v_a(k+1)} = \sigma(k+1) \cdot \|v_{aN}(k+1) - V_{dc} \cdot v_{SHM_a}(k+1)\|_2^2 \quad (2.24)$$

La función de costo para el seguimiento del patrón SHM considera un factor de peso σ variable en $k+1$. Este factor de peso permite disminuir el seguimiento del patrón en estado transitorio para priorizar el seguimiento de la corriente de referencia, similar al comportamiento de un FCS-MPC

$$\sigma(k+1) = \max(\sigma_{\max} - \lambda_\sigma \cdot \frac{(i_{g,d}(k+1) - i_{g,d}^*(k+1))^2}{I^*}, \sigma_{\min}) \quad (2.25)$$

En donde λ_σ corresponde al factor de decaimiento de σ , cuyo valor se selecciona de forma heurística, de la misma forma que $\sigma_{\max}$. Debido a que este error puede tender teóricamente a un valor negativo, se define un límite inferior cercano a 0 ($\sigma_{\min}$) obtenido heurísticamente.

La tercera parte considera la penalización al cambio de nivel de voltaje. Con esto se resuelve el problema de los pulsos espurios introducidos al voltaje v_{aN} cuando se considera un sistema conectado a la red.

$$J_{u_a(k+1)} = \lambda \|v_{aN}(k+1) - v_{aN}(k)\|_2^2 \quad (2.26)$$

En donde λ corresponde a un factor de peso de la función de costo $J_{u(k+1),a}$ cuyo valor se define de forma heurística.

Por lo tanto, la función de costo que es implementada en esta estrategia de control corresponde a

$$J_a = J_{i_{ga}(k+2)} + J_{v_a(k+1)} + J_{u_a(k+1)} \quad (2.27)$$

Finalmente, la función de costo que se debe minimizar corresponde a la sumatoria de las funciones de costo de cada fase:

$$J = J_a + J_b + J_c \quad (2.28)$$

Esto permite escoger los estados de conmutación de los transistores de potencia. Finalmente, se actualiza el valor del estado óptimo de conmutación para la siguiente iteración.

2.2.2. PLL

Un PLL (Phase-Locked Loop) es un sistema que se utiliza para sincronizar la fase y la frecuencia de una señal generada internamente con las de una señal externa. Este

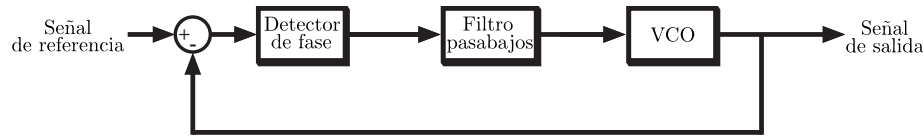


Figura 2.11. Esquema de control PLL.

proceso se logra generalmente mediante un bucle de realimentación negativa que ajusta continuamente la señal generada hasta que coincide con la señal de referencia. El sistema está compuesto por un detector de fase, un filtro pasabajos y un VCO (Voltage Controller Oscillator), tal como se muestra en la figura 2.11.

El detector de fase generalmente produce una señal pulsante donde el ancho de los pulsos es proporcional a la diferencia de fase entre la señal de referencia y la señal generada. El filtro pasabajos transforma esta señal pulsante en una señal más constante, adecuada para ser utilizada como entrada al VCO. La frecuencia de salida del VCO depende de manera no lineal del voltaje de entrada, permitiendo que el sistema ajuste y mantenga la sincronización de fase y frecuencia con precisión.

Por otro lado, un VSF-PLL (Variable Sample Frequency Phase-Locked Loop) es una variante de este tipo de controlador pero que funciona de manera un poco distinta. Este sincronizador cambia la frecuencia de muestreo de una señal en vez de cambiar la frecuencia de la señal misma.

La forma en que opera es mediante el sensado de una tensión externa definida como

$$\mathbf{u}_g(t) = U_g [\sin(\theta_g(t)) \quad \sin(\theta_g(t) - 2\pi/3) \quad \sin(\theta_g(t) + 2\pi/3)]^T \quad (2.29)$$

Donde U_g representa la magnitud de la tensión de la señal con la que se quiere sincronizar y $\theta_g(t)$ la fase de la tensión a sincronizar. Por otro lado, se tiene una variable interna definida como

$$\mathbf{u}_i(t) = [\cosine(n_0(k)) \quad \cosine(n_{-2\pi/3}(k)) \quad \cosine(n_{+2\pi/3}(k))]^T \quad (2.30)$$

Donde $\cosine()$ es una tabla discretizada de la función coseno con M valores correspondientes a la cantidad de muestras por periodo. $n_0(k)$, $n_{-2\pi/3}(k)$, $n_{+2\pi/3}(k)$ con valores iniciales ($k = 0$) de 1, $2M/3 + 1$ y $M/3 + 1$ respectivamente. Estos valores se actualizan sumando la unidad por cada muestreo, reiniciando la cuenta al llegar a $k = M$.

Al hacer el producto punto entre la señal externa (2.29) e interna (2.30) se obtiene la expresión

$$u(k) = \langle \mathbf{u}_g(t), \mathbf{u}_i(k) \rangle = (3/2)U_g \sin(\theta_g(t) - 2\pi n_0(k)/M) \quad (2.31)$$

donde para que la señal esté en sincronía se requiere que $2\pi n_0(k)/M = \theta_g(t) + 2m\pi$ con $m \in \mathbb{Z}$. Esta condición se muestra en

$$\theta_i(k) = \theta_g(k) + 2m\pi \implies \langle \mathbf{u}_g^{abc}(k), \mathbf{u}_i^{abc}(k) \rangle = 0. \quad (2.32)$$

Se puede filtrar la señal en caso de tener ruido mediante

$$u_f(k) = u_f(k-1) + \alpha(u(k) - u_f(k-1)) \quad (2.33)$$

Donde α corresponde a la frecuencia de corte del filtro pasa-bajos y u_f la señal filtrada. Para lograr que $u_f = 0$, tal como se describe en la expresión (2.32), dado que u_f corresponde a la misma señal filtrada, se utiliza un control PI con función de transferencia en el plano z , de la forma

$$c(z) = k_c \frac{z - a_c}{z - 1} = \frac{T_s(z)}{e(z)} = \frac{T_s(z)}{0 - u_f(z)} = \frac{T_s(z)}{-u_f(z)}. \quad (2.34)$$

Este control se elige porque permite respuestas rápidas a las diferencias con respecto a los cambios en el error y, además, elimina el error a largo plazo, logrando que en estado estacionario $u_f = 0$.

Por lo que la expresión para el periodo de muestreo $T_s(k)$ tiene la forma

$$T_s(k) = T_s(k-1) - k_c u_f(k) + k_c a_c u_f(k-1) \quad (2.35)$$

Este tiempo de muestreo se utiliza para la proxima vez que se mida la señal.

Capítulo 3

Modelación Convertidor Multicelda en Cascada en FPGA.

Para la modelación del Convertidor Multicelda en Cascada se utilizó el software Vitis HLS junto con Vivado para modelar tanto el convertidor, la línea y la red. El Convertidor Multicelda consta de tres celdas en cascada por fase, que corresponden a nueve celdas en total. Cada celda cuenta con cuatro MOSFET con diodo antiparalelo. Este convertidor está conectado a la red mediante una línea de resistencia e inductancia por fase de 1Ω y $0,02H$ respectivamente. Considerando lo anterior, se tienen 36 entradas lógicas S_{fxy} para las conmutaciones que equivalen a la cantidad de MOSFET's que forman parte del convertidor. La tensión de la red se fijó en $100V_{peak}$ y se generó mediante un contador que se incrementaba en una unidad cada vez que se ejecutaba el modelo del convertidor, el cuál lo hacía cada $1,11 \cdot 10^{-6}$ segundos. Para que esto funcionara correctamente, se multiplicó el tiempo de ejecución por el contador y la frecuencia de la red para simular los 50 Hz a tiempo real.

Para la obtención de las tensiones de salida de cada fase del convertidor, primero se analizó una única celda, dado que todas eran idénticas. El esquema de la celda se observa en la figura 3.1. Se consideró las caídas de tensión tanto en el MOSFET como en los diodos, lo que derivó en la obtención de la tabla 3.1, que representa los valores de salidas según los valores de corriente i_o del convertidor, dado que algunos estados tienen dependencia de esta.

Cada una de las tensiones de las celdas en cascada se suman por fase para obtener la tensión que irá hacia la red v_{xn} . Esta tensión se genera a partir de la configuración de los valores S_{fxy} seleccionados a partir del algoritmo FCS-MPC comentado en la sección final del capítulo anterior. En la tabla 3.2 se observa la configuración posible de interruptores que se programó para generar la tensión v_{an} . La mitad de los interruptores no están representados en la tabla debido a que, tal como mostró la figura 2.2, son el valor lógico negado del interruptor de la misma pierna del puente H correspondiente.

Luego, haciendo las ecuaciones mediante las leyes de Kirchhoff para el lado ac del convertidor, considerando tensión de la red v_{rx} , la tensión de nodo común v_{on} y la caída de tensión en la línea, tal como se muestra en la figura 3.2, se obtiene la ecuación diferencial

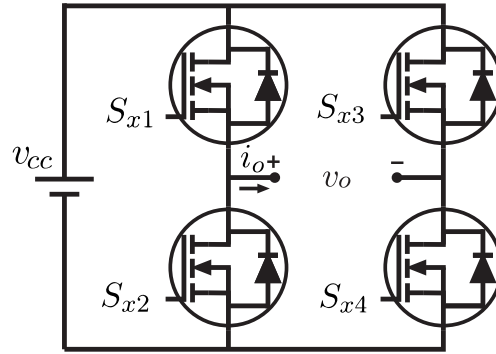
**Figura 3.1.** Paneles fotovoltaicos junto con topología CHB.

Tabla 3.1
TENSIÓN DE SALIDA

S_{x1}	S_{x2}	S_{x3}	S_{x4}	Voltaje de Salida $v_o(i_o > 0)$	Voltaje de Salida $v_o(i_o < 0)$
0	0	0	0	$-v_{cc}$	v_{cc}
0	0	0	1	0	v_{cc}
0	0	1	0	$-v_{cc}$	0
0	1	0	0	$-v_{cc}$	0
1	0	0	0	0	v_{cc}
1	0	1	0	0	0
1	0	0	1	v_{cc}	v_{cc}
0	1	1	0	$-v_{cc}$	$-v_{cc}$
0	1	0	1	0	0

$$v_{xn} - v_{rx} - v_{on} = i_x R + \frac{di_x}{dt} L, \quad (3.1)$$

que, al utilizar el operador diferencial p , se deriva

$$v_{xn} - v_{rx} - v_{on} = i_x (R + Lp). \quad (3.2)$$

Donde v_{on} está definida como el promedio de la suma de las tensiones de salida por fase del convertidor.

De la ecuación anterior se obtiene la función de transferencia que relaciona la salida, correspondiente a la corriente i_x , con la entrada, correspondiente a la tensión $v_{xn} - v_{rx} - v_{on}$. Esta función de transferencia corresponde a

$$H(s) = \frac{\frac{1}{R}}{1 + \frac{L}{R}s}, \quad (3.3)$$

Tabla 3.2
TENSIÓN v_{an}

S_{a11}	S_{a13}	S_{a21}	S_{a23}	S_{a31}	S_{a33}	v_{an}
1	0	1	0	1	0	$3v_{cc}$
1	0	1	0	0	0	$2v_{cc}$
1	0	0	0	0	0	v_{cc}
0	0	0	0	0	0	0
0	1	0	0	0	0	$-v_{cc}$
0	1	0	1	0	0	$-2v_{cc}$
0	1	0	1	0	1	$-3v_{cc}$

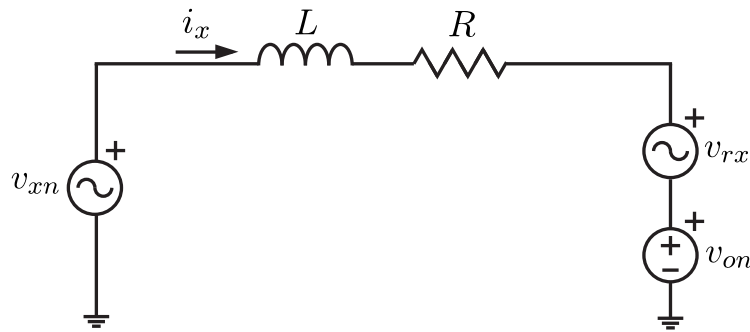


Figura 3.2. Circuito fase “x”.

donde L y R corresponde a la inductancia y resistencia de la línea.

Mediante una aproximación por Euler hacia adelante a través de la relación

$$s = \frac{z - 1}{T}, \quad (3.4)$$

se determina que la función de transferencia en términos de z corresponde a

$$H(z) = \frac{\frac{T_s}{L}}{z - (1 - \frac{RT_s}{L})}. \quad (3.5)$$

Donde T_s corresponde al tiempo en el cual se demora en simular el modelo del convertidor, el cual corresponde a $1,11 \cdot 10^{-6}$ segundos.

Reemplazando los valores correspondientes de resistencia, inductancia y tiempo de simulación se obtiene la función de transferencia

$$H(z) = \frac{5,5498 \cdot 10^{-5}}{z - 0,9999}. \quad (3.6)$$

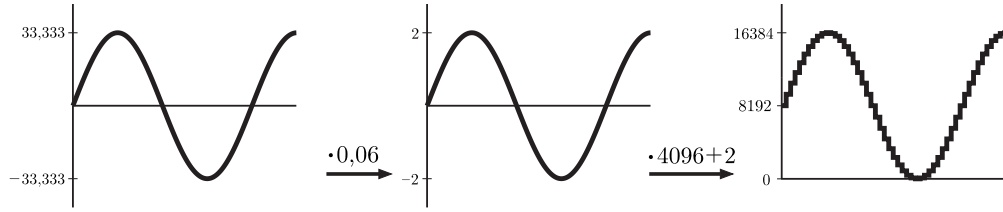


Figura 3.3. *Conversión medición de corriente.*

Esta ecuación permite modelar por fase la relación entre la corriente y las tensiones mediante un simple cálculo iterativo en el FPGA. Las ecuaciones (3.7), (3.8), (3.9) y (3.10) muestran como se puede derivar el cálculo iterativo a partir de la función de transferencia discreta antes mencionada.

$$H(z) = \frac{I_{out}(z)}{V_{in}(z)} \quad (3.7)$$

$$zI_{out}(z) - 0,9999I_{out}(z) = 5,5498 \cdot 10^{-5}V_{in}(z) \quad (3.8)$$

$$i_x(t+h) - 0,9999i_x(t) = 5,5498 \cdot 10^{-5}(v_{xn}(t) - v_{rx}(t) - v_{on}(t)) \quad (3.9)$$

$$i_x(t) = 0,9999i_x(t-h) + 5,5498 \cdot 10^{-5}(v_{xn}(t-h) - v_{rx}(t-h) - v_{on}(t-h)) \quad (3.10)$$

Este valor de corriente se reitera para cada fase, dado que todas las líneas son idénticas.

Para obtener las mediciones de corriente que irán al DSP, primero se multiplica las corrientes por 0,06 y luego se le suma 2 y, finalmente, se multiplica por 4096 dado que la salida es de 14 bits de resolución lo que equivale a un valor máximo de $2^{14} = 16384$. Esto se observa en la figura 3.3, donde la corriente máxima que se logra medir mediante esta conversión corresponde a 33,333. De forma esquemática se discretizó la señal con una menor cantidad de bits que la real para representar la forma de la señal. Este valor máximo de corriente que es posible medir se eligió con el objetivo tener un margen de medición para los ensayos posteriores, además de acercarse a la mayor resolución posible, usando la mayor cantidad de bits.

Para obtener la medición de la tensión de la red se realizó un procedimiento similar al descrito en el párrafo anterior: se multiplicó por 0,008 y se le sumó 1 y, finalmente, se multiplicó por 8192, considerando la misma salida de 14 bits de resolución. Esto se observa en la figura 3.4 donde, si bien la tensión peak de la red corresponde a 100[V], la tensión máxima que se logra medir con esta conversión corresponde a 125[V]. Este valor se eligió para tener un margen de medición, a pesar de que en este caso la tensión no varía su valor peak en absoluto. La diferencia en los valores de conversión no tiene ningún propósito más allá de mostrar 2 formas de conversión distintas.

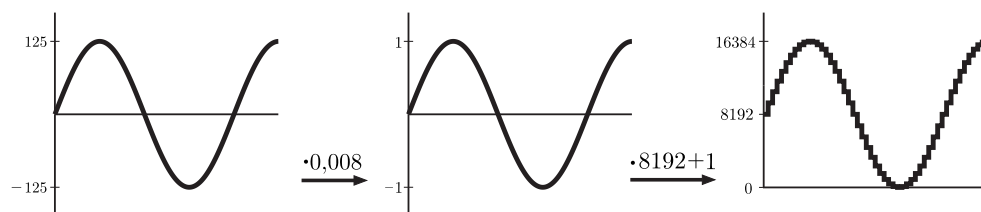


Figura 3.4. *Conversión medición de tensión.*

Finalmente, se realiza la conversión inversa para procesar los valores en el esquema de control.

Capítulo 4

Optimización de esquema de control en DSP

A continuación se describirá la optimización de los distintos trozos del algoritmo SHM-MPC. La optimización del código del algoritmo en cuestión ejecutado en la tarjeta a través del DSP *TMS320C6748* con el software Code Composer Studio 9.1 se realizó mediante comparaciones de las distintas formas alternativas en que se podía escribir las secciones sin alterar la finalidad de estas. Estas comparaciones se realizaron con el software Code Composer Studio 3.1 a través de una simulación del DSP *TMS320C6713*. Se eligió esta versión más antigua junto con la simulación de la versión de ese DSP debido principalmente a que no se encontró una forma relativamente simple de estimar el costo computacional en el 6748. En cambio, al correr las secciones en la versión más antigua del software, se podía verificar los ciclos de reloj que demoraban las secciones en ejecutarse. Estos ciclos de reloj representan un intervalo de tiempo equivalente a $1/225\text{MHz}$.

El DSP puede trabajar con distintos tipos de variables. Las variables en cuestión corresponden a las del tipo “int” (enteros de 32 bits), “long” (enteros de 64 bits), “float” (flotantes de 32 bits) y “double” (flotantes de 64 bits). Uno de los puntos más importantes, si no el más importante, a considerar para la optimización es utilizar en mayor medida las variables de tipo int y float, debido a que son las que requieren una menor cantidad de ciclos para su ejecución. Entre estos dos tipos de variables, lo ideal es utilizar las del tipo “int”, por lo que varias de las optimizaciones que se muestran más adelante están basadas en esta idea.

El algoritmo SHM-MPC que se optimizó utilizó un máximo de 7 niveles y funcionó a partir de 9 ángulos elegidos de tal forma que se siguiera un patrón SHM. Este patrón aseguró que los armónicos cumplieran con la norma IEEE Std. 519 2022, específicamente que se cumpliera con la primera fila de la tabla de la figura 2.7. Esta fila se eligió debido a que se correspondía a valores donde la razón entre la corriente de cortocircuito en PCC y la corriente de demanda máxima en el PCC era menor a 20. Estos ángulos se determinaron en función del índice de modulación, tal como se muestra en la figura 4.1.

La sección del código destinada a la sincronización, es decir, el VSF-PLL no requirió ningún ajuste debido a que esta sección contenía principalmente operaciones aritméticas y comparaciones. Como detalle adicional, en una de las distintas versiones del código se requirió el uso de un buffer que optimizó bastante esta sección, sin embargo, para la versión

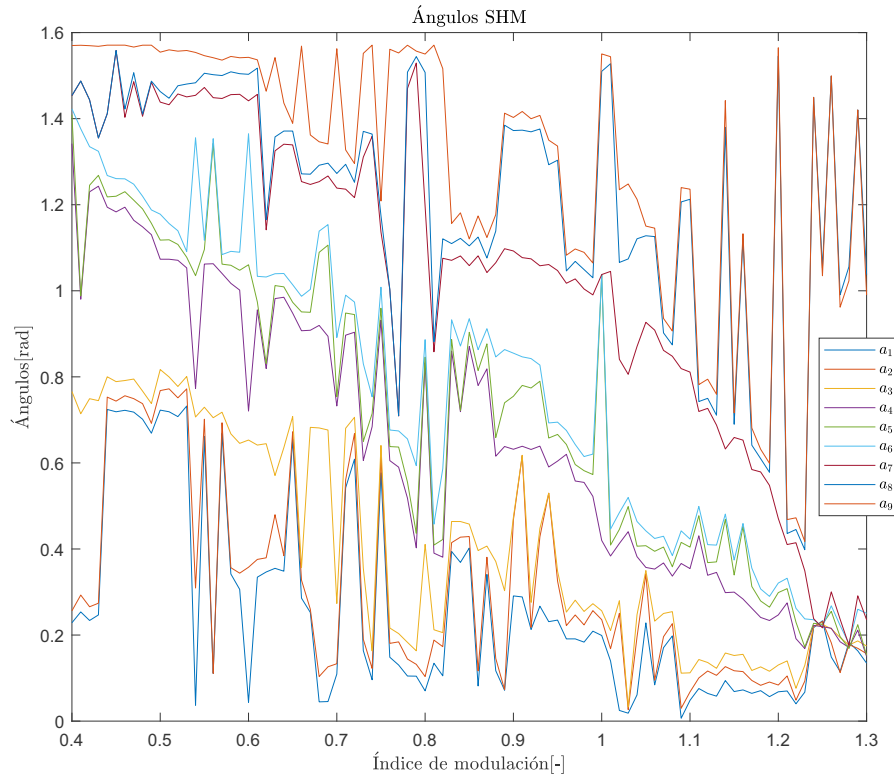


Figura 4.1. *Ángulos patrón SHM.*

final no fue necesario.

La sección destinada al algoritmo SHM-MPC, en cambio, requirió bastantes ajustes de optimización. A continuación se describirán las optimizaciones realizadas.

Con respecto al cálculo de la corriente de referencia 2.2.1.a y las tensiones de referencia para el cálculo del voltage de salida en estado estacionario 2.2.1.c, el código original que generaba las referencias de corriente en el tiempo k y $k + 2$ correspondientes a $i_{g_a_r_k}$, $i_{g_b_r_k}$, $i_{g_a_r_k2}$, $i_{g_b_r_k2}$ y las tensiones de referencia $v_{L_a_k}$, $v_{L_b_k}$, $v_{L_c_k}$, es el siguiente:

```
i_g_a_r_k = ref_in*cos(wt_a_k + phi_r_k);
i_g_b_r_k = ref_in*cos(wt_b_k + phi_r_k);
i_g_c_r_k = - i_g_a_r_k - i_g_b_r_k;
```

```
i_g_a_r_k2 = ref_in*cos(wt_a_k2 + phi_r_k);
i_g_b_r_k2 = ref_in*cos(wt_b_k2 + phi_r_k);
i_g_c_r_k2 = - i_g_a_r_k2 - i_g_b_r_k2;
```

```
v_L_a_k = ref_in*wL*cos(wt_a_k + phi_r_k - PI.2);
v_L_b_k = ref_in*wL*cos(wt_b_k + phi_r_k - PI.2);
v_L_c_k = ref_in*wL*cos(wt_c_k + phi_r_k - PI.2);
```

Este código usa principalmente la función “cos” que viene en la librería genérica math.h, la cual contiene funciones más lentas que las que se encuentran definidas en la librería especializada mathlib.lib. Es por esto que la mejor alternativa es utilizar la

función “cosp” de la librería `mathlib.lib`, la cual corresponde a la versión optimizada de tipo flotante de la función coseno. El código utilizando esta función optimizada se observa en el siguiente código:

```
i_g_a_r_k = ref_in*cosp(wt_a_k + phi_r_k);
i_g_b_r_k = ref_in*cosp(wt_b_k + phi_r_k);
i_g_c_r_k = - i_g_a_r_k - i_g_b_r_k;

i_g_a_r_k2 = ref_in*cosp(wt_a_k2 + phi_r_k);
i_g_b_r_k2 = ref_in*cosp(wt_b_k2 + phi_r_k);
i_g_c_r_k2 = - i_g_a_r_k2 - i_g_b_r_k2;
```

```
v_L_a_k = ref_in*wL*cosp(wt_a_k + phi_r_k - PI.2);
v_L_b_k = ref_in*wL*cosp(wt_b_k + phi_r_k - PI.2);
v_L_c_k = ref_in*wL*cosp(wt_c_k + phi_r_k - PI.2);
```

El costo computacional se puede reflejar en la cantidad de ciclos de reloj que demoran ambos procesos. Para el primer caso correspondiente al código con las funciones “cos” demoró un total de 3482 ciclos de reloj en ejecutarse, mientras que para el código con las funciones “cosp” se demoró un total de 837 (un 76 % menor al original). Se puede notar que para la comparación no es necesario tomar en consideración $i_{g-c-r-k}$ ni $i_{g-c-r-k2}$, dado que están en ambas versiones del código.

Con respecto al cálculo de la predicción de corriente en $k + 1$ 2.2.1.b, no se realizó ninguna optimización dado que este trozo del código se componía únicamente de operaciones aritméticas.

Siguiendo el cálculo del voltaje de salida en estado estacionario 2.2.1.c, en el código original escrito en Matlab se utiliza funciones especiales para el cálculo de $v_{ab-dq-mod-k}$ y de $v_{ab-dq-ang-k}$. Estas funciones corresponden a `hypot()` y a `atan2()` para el módulo y el ángulo de v_{ab-dq} respectivamente. De estas funciones solo la segunda estaba disponible para usar en Code Composer, por lo que, para el cálculo del módulo, se hizo el cálculo normal y se utilizó una función similar a `sqrt()`. Se analizaron dos casos, uno utilizando las funciones `sqrt()` y `atan2()` de la librería `math.h` y otro caso utilizando las funciones `sqrtsp()` y `atan2sp()` de la librería especializada `mathlib.lib`. El primer caso se muestra a continuación:

```
v_ab_dq_mod_k = sqrt(v_ab_d_k*v_ab_d_k + v_ab_q_k*v_ab_q_k);
v_ab_dq_ang_k = atan2(v_ab_q_k, v_ab_d_k);
```

El cual necesitó 2010 ciclos de reloj para su ejecución. Para el segundo caso, donde se utilizó la librería `mathlib.lib`, se muestra a continuación:

```
v_ab_dq_mod_k = sqrtsp(v_ab_d_k*v_ab_d_k + v_ab_q_k*v_ab_q_k);
v_ab_dq_ang_k = atan2sp(v_ab_q_k, v_ab_d_k);
```

Para este caso se utilizaron 175 ciclos de reloj (un 91 % menor que el original) para su ejecución. Por lo que claramente se utilizó la segunda opción en el código, dado que es más de 10 veces más rápido comparado al primer caso.

En el código original para el cálculo de wt_{xk1} se hacían dos transformaciones. Este código adaptado a Code Composer se muestra a continuación:

```

float ang_aux_a1 = wt_a_k1 + PI_2 + v_ab_dq_ang_k;
float ang_aux_b1 = wt_b_k1 + PI_2 + v_ab_dq_ang_k;
float ang_aux_c1 = wt_c_k1 + PI_2 + v_ab_dq_ang_k;

if (ang_aux_a1 < 0) ang_aux_a1 += PI2;
else if (ang_aux_a1 >= PI2) ang_aux_a1 -= PI2;

if (ang_aux_b1 < 0) ang_aux_b1 += PI2;
else if (ang_aux_b1 >= PI2) ang_aux_b1 -= PI2;

if (ang_aux_c1 < 0) ang_aux_c1 += PI2;
else if (ang_aux_c1 >= PI2) ang_aux_c1 -= PI2;

ang_aux_a2 = ang_aux_a1 + PI_2 - PI;
ang_aux_b2 = ang_aux_b1 + PI_2 - PI;
ang_aux_c2 = ang_aux_c1 + PI_2 - PI;

if (ang_aux_a2 < 0) ang_aux_a2 += PI2;
else if (ang_aux_a2 >= PI2) ang_aux_a2 -= PI2;

if (ang_aux_b2 < 0) ang_aux_b2 += PI2;
else if (ang_aux_b2 >= PI2) ang_aux_b2 -= PI2;

if (ang_aux_c2 < 0) ang_aux_c2 += PI2;
else if (ang_aux_c2 >= PI2) ang_aux_c2 -= PI2;

wt_she_a_k1      = ang_aux_a2;
wt_she_b_k1      = ang_aux_b2;
wt_she_c_k1      = ang_aux_c2;

```

Esta parte del código requirió 776 ciclos de reloj para su ejecución, sin embargo, para propósitos de la optimización no era necesario hacer dos transformaciones. Esto se ve reflejado en el código optimizado que solo se utiliza una transformación, el cual se muestra a continuación:

```

float ang_aux_a = wt_a_k1 + PI_2 + v_ab_dq_ang_k;
float ang_aux_b = wt_b_k1 + PI_2 + v_ab_dq_ang_k;
float ang_aux_c = wt_c_k1 + PI_2 + v_ab_dq_ang_k;

if (ang_aux_a < 0) ang_aux_a += PI2;
else if (ang_aux_a >= PI2) ang_aux_a -= PI2;

if (ang_aux_b < 0) ang_aux_b += PI2;
else if (ang_aux_b >= PI2) ang_aux_b -= PI2;

if (ang_aux_c < 0) ang_aux_c += PI2;
else if (ang_aux_c >= PI2) ang_aux_c -= PI2;

wt_she_a_k1      = ang_aux_a;
wt_she_b_k1      = ang_aux_b;
wt_she_c_k1      = ang_aux_c;

```

Este código, en cambio, requirió 307 ciclos de reloj (un 60% menor que el original), debido claramente a que se utilizó solo una transformación en vez de las dos originales.

Posteriormente, para el caso optimizado, se acotó entre 0 a 2π para obtener *wt_she_x_k1* y se multiplicó por 10000 para luego redondearlo al entero y obtener *wt_sheint_x_k1*. Sin embargo, esto último no se tomó en consideración para la obtención del número de ciclos de reloj dado que se considerará en lo que se detallará a continuación.

Respecto de la lectura de la tabla SHM discretizada 2.2.1.d, el algoritmo para la obten-

ción de la forma SHM de la tensión se determina originalmente a partir de un código que tiene como entradas los ángulos eléctricos expresados como dientes de sierra *wt_she_x_k1*. Este código, únicamente para la fase “a” dado su cantidad de líneas, se muestra a continuación:

```

if (wt_she_a_k1 <= t2){
    if (wt_she_a_k1 >= 0 && wt_she_a_k1 <= a1){
        v_she_a_k1 = C;}
    if (wt_she_a_k1 > a1 && wt_she_a_k1 <= a2){
        v_she_a_k1 = U;}
    if (wt_she_a_k1 > a2 && wt_she_a_k1 <= a3){
        v_she_a_k1 = C;}
    if (wt_she_a_k1 > a3 && wt_she_a_k1 <= a4){
        v_she_a_k1 = U;}
    if (wt_she_a_k1 > a4 && wt_she_a_k1 <= a5){
        v_she_a_k1 = D;}
    if (wt_she_a_k1 > a5 && wt_she_a_k1 <= a6){
        v_she_a_k1 = U;}
    if (wt_she_a_k1 > a6 && wt_she_a_k1 <= a7){
        v_she_a_k1 = D;}
    if (wt_she_a_k1 > a7 && wt_she_a_k1 <= a8){
        v_she_a_k1 = T;}
    if (wt_she_a_k1 > a8 && wt_she_a_k1 <= a9){
        v_she_a_k1 = D;}
    if (wt_she_a_k1 > a9 && wt_she_a_k1 <= b1){
        v_she_a_k1 = T;}

    if (wt_she_a_k1 > b1 && wt_she_a_k1 <= b2){
        v_she_a_k1 = D;}
    if (wt_she_a_k1 > b2 && wt_she_a_k1 <= b3){
        v_she_a_k1 = T;}
    if (wt_she_a_k1 > b3 && wt_she_a_k1 <= b4){
        v_she_a_k1 = D;}
    if (wt_she_a_k1 > b4 && wt_she_a_k1 <= b5){
        v_she_a_k1 = U;}
    if (wt_she_a_k1 > b5 && wt_she_a_k1 <= b6){
        v_she_a_k1 = D;}
    if (wt_she_a_k1 > b6 && wt_she_a_k1 <= b7){
        v_she_a_k1 = U;}
    if (wt_she_a_k1 > b7 && wt_she_a_k1 <= b8){
        v_she_a_k1 = C;}
    if (wt_she_a_k1 > b8 && wt_she_a_k1 <= b9){
        v_she_a_k1 = U;}
    if (wt_she_a_k1 > b9 && wt_she_a_k1 <= t2){
        v_she_a_k1 = C;}
} else{
    if (wt_she_a_k1 > t2 && wt_she_a_k1 <= c1){
        v_she_a_k1 = C;}
    if (wt_she_a_k1 > c1 && wt_she_a_k1 <= c2){
        v_she_a_k1 = MU;}
    if (wt_she_a_k1 > c2 && wt_she_a_k1 <= c3){
        v_she_a_k1 = C;}
    if (wt_she_a_k1 > c3 && wt_she_a_k1 <= c4){
        v_she_a_k1 = MU;}
    if (wt_she_a_k1 > c4 && wt_she_a_k1 <= c5){
        v_she_a_k1 = MD;}
    if (wt_she_a_k1 > c5 && wt_she_a_k1 <= c6){
        v_she_a_k1 = MU;}
    if (wt_she_a_k1 > c6 && wt_she_a_k1 <= c7){
        v_she_a_k1 = MD;}
    if (wt_she_a_k1 > c7 && wt_she_a_k1 <= c8){
        v_she_a_k1 = MT;}
    if (wt_she_a_k1 > c8 && wt_she_a_k1 <= c9){
        v_she_a_k1 = MD;}
    if (wt_she_a_k1 > c9 && wt_she_a_k1 <= d1){
        v_she_a_k1 = MT;}
}

```

```

if (wt_she_a_k1 > d1 && wt_she_a_k1 <= d2){
    v_she_a_k1 = MD;}
if (wt_she_a_k1 > d2 && wt_she_a_k1 <= d3){
    v_she_a_k1 = MT;}
if (wt_she_a_k1 > d3 && wt_she_a_k1 <= d4){
    v_she_a_k1 = MD;}
if (wt_she_a_k1 > d4 && wt_she_a_k1 <= d5){
    v_she_a_k1 = MU;}
if (wt_she_a_k1 > d5 && wt_she_a_k1 <= d6){
    v_she_a_k1 = MD;}
if (wt_she_a_k1 > d6 && wt_she_a_k1 <= d7){
    v_she_a_k1 = MU;}
if (wt_she_a_k1 > d7 && wt_she_a_k1 <= d8){
    v_she_a_k1 = C;}
if (wt_she_a_k1 > d8 && wt_she_a_k1 <= d9){
    v_she_a_k1 = MU;}
if (wt_she_a_k1 > d9 && wt_she_a_k1 <= t4){
    v_she_a_k1 = C;}
}

```

Dependiendo de en que valor se ubique $wt_she_x_k1$, el cual, como se dijo anteriormente, está acotado entre 0 a 2π , se obtienen los valores deseados del patrón SHE, $v_she_x_k1$. Tal como se muestra en la figura 4.2, en la cual se tomaron ángulos igualmente espaciados para mostrar de forma clara el algoritmo. Si bien es un algoritmo simple, no es lo óptimo, dado que su costo computacional se puede mejorar bastante. Este código demoró 3685 ciclos de reloj en ejecutarse para todas las fases.

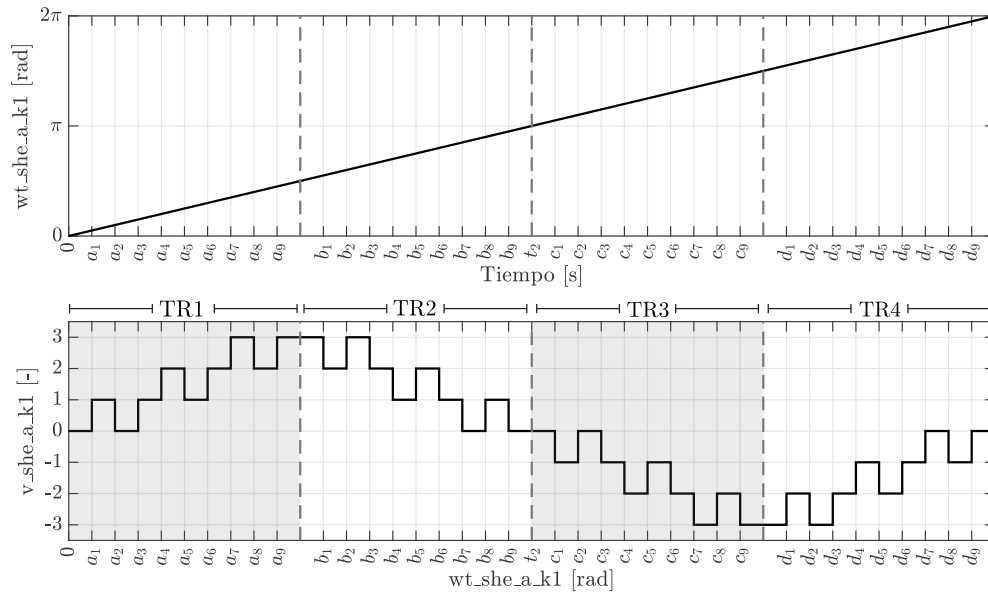


Figura 4.2. *Algoritmo SHE.*

Para poder mejorar el costo computacional de la obtención de la forma SHM para la tensión se modificó el algoritmo para que incorporara entradas en variables de tipo entero en vez de variables de tipo flotante. Esto se logró mediante el uso de la variable $wt_sheint_x_k1$ mencionada anteriormente. Además de esto, se modificó la tabla calculada de manera *offline* que contenía los ángulos para la modulación SHM, multiplicando todos

los ángulos de la tabla por 10000 y redondeándolos a su entero más cercano. La estructura de este algoritmo también se modificó, en este caso se aprovechó su naturaleza simétrica para utilizar un solo cuadrante en vez de los 4 originales. Este algoritmo optimizado se muestra a continuación:

```

TRa= (wt_sheint_a_k1>0)+(wt_sheint_a_k1>INTPI.2)+(wt_sheint_a_k1>INTPI)+(
wt_sheint_a_k1>INTPI23);

wt_sheint_a_k1 = wt_sheint_a_k1-(TRa - 1)*INTPI.2;

if (TRa == 4 || TRa == 2){
    Va = (wt_sheint_a_k1>INTPI.2-a9)+(wt_sheint_a_k1>INTPI.2-a8)+(
wt_sheint_a_k1>INTPI.2-a7)+(wt_sheint_a_k1>INTPI.2-a6)+(wt_sheint_a_k1>
INTPI.2-a5)+(wt_sheint_a_k1>INTPI.2-a4)+(wt_sheint_a_k1>INTPI.2-a3)+(
wt_sheint_a_k1>INTPI.2-a2)+(wt_sheint_a_k1>INTPI.2-a1);
}
else{
    Va = (wt_sheint_a_k1>a1)+(wt_sheint_a_k1>a2)+(wt_sheint_a_k1>a3)+(
wt_sheint_a_k1>a4)+(wt_sheint_a_k1>a5)+(wt_sheint_a_k1>a6)+(
wt_sheint_a_k1>a7)+(wt_sheint_a_k1>a8)+(wt_sheint_a_k1>a9);
}

v_she_a_k1=data[Va];

if (TRa==3){
    v_she_a_k1 = -1*v_she_a_k1;
}
else if (TRa==2){
    v_she_a_k1 = v_she_a_k1 ^ 3;
}
else if (TRa==4){
    v_she_a_k1 = -1*(v_she_a_k1 ^ 3);
}

```

Primero se determinó el cuadrante o tramo mediante una variable que tomaba valores entre 1 a 4 dependiendo del cuadrante de $wt_sheint_x_k1$. Luego, se actualizó la variable $wt_sheint_x_k1$, manteniéndola entre 0 a $10^4 \cdot \frac{\pi}{2}$ independientemente de en que tramo se ubicaba originalmente. Dado que los tramos 1 y 3 son las imágenes reflejadas horizontalmente de los tramos 2 a 4, se utilizaron los ángulos complementarios en el caso de que los tramos almacenados en la variable que guardaba el número del tramo fuera 2 o 4. Posteriormente se almacenó un valor entre 0 a 8 dependiendo de en que subtramo determinado por los ángulos (o complemento de ángulos) de la matriz alpha se encontraba $wt_sheint_x_k1$. Este valor se usó como índice de una matriz que toma de referencia el primer tramo para determinar los valores de $v_she_a_k1$. Este valor de $v_she_a_k1$ se actualizó dependiendo de en que tramo se encontraba. Para el caso de que se ubique en el primer tramo el valor se mantiene igual. Para el segundo tramo se le aplica una compuerta XOR entre el valor $v_she_a_k1$ y 3, esto resulta en la tabla 4.1, lo que permite hacer un efecto de espejo con los valores obtenidos en el primer tramo. Para el tercer tramo solo se multiplica por -1 la señal del primer cuadrante, dado que se refleja verticalmente. Por último, para el cuarto tramo, se refleja tanto horizontalmente como verticalmente, por lo que se le aplica una compuerta XOR entre ese valor y 3, y se multiplica el valor resultante por -1 . Esto se ve mejor reflejado en la figura 4.3

Tabla 4.1
TABLA LÓGICA XOR

A	B	A XOR B
000	011	011
001	011	010
010	011	001
011	011	000

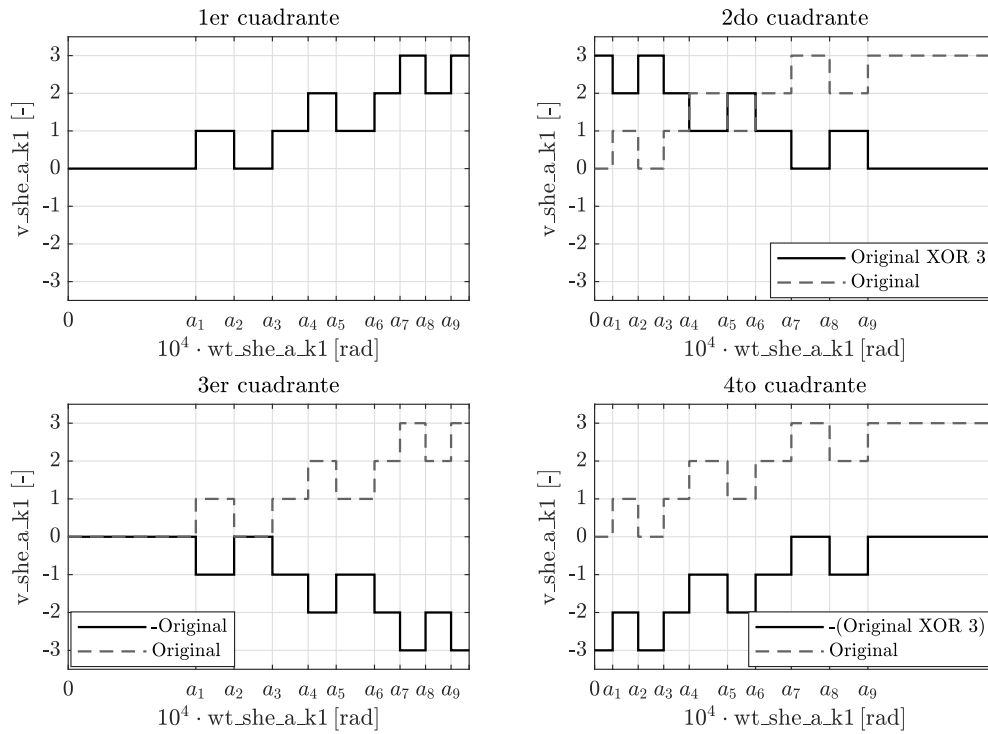


Figura 4.3. Algoritmo SHE optimizado.

Este código optimizado demoró 788 ciclos de reloj (un 79% menor al original) en ejecutarse para todas las fases, siendo claramente mucho más rápido que el código original.

Para el algoritmo del hexágono, el cual tenía como objetivo, tal como se comentó en la sección 2.2.1.e, disminuir la cantidad de iteraciones en el FCS-MPC, se tenía originalmente el siguiente código:

```

v_she_Alpha_k1 = DOS.TER*v_she_a_k1 - UN.TER*v_she_b_k1 - UN.TER*v_she_c_k1;
v_she_Beta_k1 = UN.RAIZTER*v_she_b_k1 - UN.RAIZTER*v_she_c_k1;
q_opt_k1 = 1;

for (k = 1; k < 128; k++){ //128
    float X1 = v_she_Alpha_k1 - HEX1[0][k-1];
    float X2 = v_she_Beta_k1 - HEX1[1][k-1];
    if (X1 < 0){
        X1 = -X1;
    }
}

```

```

        if (X2 < 0){
            X2 = -X2;
        }
        if (X1 < 0.01 && X2 < 0.01){
            q_opt_k1 = k;
        }
    }

    p_opt_k1 = 1;
    HEX2.ind2 = q_opt_k1-1;
    for (k = 1; k < 8; k++){
        int HEX2.ind1 = k-1;
        int HEX3.ind = HEX2[HEX2.ind1][HEX2.ind2]-1;
        float X1 = v_she_a_k1 - HEX3[HEX3.ind][1];
        float X2 = v_she_b_k1 - HEX3[HEX3.ind][2];
        float X3 = v_she_c_k1 - HEX3[HEX3.ind][3];
        if (X1 < 0){
            X1 = -X1;
        }
        if (X2 < 0){
            X2 = -X2;
        }
        if (X3 < 0){
            X3 = -X3;
        }
        if (X1 < 0.5 && X2 < 0.5 && X3 < 0.5){
            p_opt_k1 = HEX3[HEX3.ind][0];
        }
    }
}

```

Lo primero que se busca hacer es pasar desde coordenadas a, b, c a coordenadas α, β a través de la transformación ya comentada. Luego se hace una búsqueda a través de un bucle para obtener el valor q_opt_k1 , el cual es parte de un índice asociado a cada uno de los 127 vectores posibles en el plano α, β formados por el Convertidor Multicelda en Cascada de 7 niveles. Este índice se utiliza para buscar dentro de los posibles 7 vectores (peor caso) en el plano a, b, c que pudieron dar como resultado ese vector obtenido en el plano α, β . Este proceso da como resultado la obtención del índice p_opt_k1 , el cuál contiene información acerca del vector en el plano a, b, c y que se utiliza junto con la matriz HEX4 en la búsqueda local de los 7 vectores que se utilizarán en el algoritmo FCS-MPC.

Si bien este algoritmo no parece tener posibilidades de optimización, si hay unos pequeños detalles que pueden hacer una gran diferencia. A continuación se muestra el algoritmo optimizado:

```

int v_she_Alpha_k1 = 2*v_she_a_k1 - v_she_b_k1 - v_she_c_k1;
int v_she_Beta_k1 = v_she_b_k1 - v_she_c_k1;
int q_opt_k1 = 1;

int k;
for (k = 1; k < 128; k++){ //128
    if (v_she_Alpha_k1 == HEX1[0][k-1] && v_she_Beta_k1 == HEX1[1][k-1]){
        q_opt_k1 = k;
        break;
    }
}

int p_opt_k1 = 1;
int HEX2.ind2 = q_opt_k1-1;
for (k = 1; k < 8; k++){
    int HEX2.ind1 = k-1;
    int HEX3.ind = HEX2[HEX2.ind1][HEX2.ind2]-1;
    if (v_she_a_k1 == HEX3[HEX3.ind][1] && v_she_b_k1 == HEX3[HEX3.ind][2] &&

```

```

        v_she_c_k1 == HEX3[HEX3_ind][3]) {
        p_opt_k1 = HEX3[HEX3_ind][0];
        break;
    }
}

```

Se observa en primer lugar que la transformación desde el plano a,b,c al plano α,β es distinta, debido a que lo que se hizo fue normalizar los valores α,β con respecto a $1/3$ y $1/\sqrt{3}$ respectivamente. La normalización también se realizó de manera adecuada en la tabla HEX1 con el objetivo de no afectar el objetivo del código. Esta normalización no se visualiza en el código porque se realizó de manera inmediata en la definición de `v_she_Alpha_k1` y `v_she_Beta_k1`, manifestandose en la eliminación de los factores `DOS_TER`, `UN_TER` y `UN_RAIZER`. La ventaja de esto es que ahora se puede trabajar simplemente con valores enteros en la búsqueda de los índices `q_opt_k1` y `p_opt_k1` haciendo más rápida su ejecución dado que ya no se requiere de un intervalo que permita determinar si el valor coincide con el de las tablas, si no que simplemente basta con una igualdad. Además de esta optimización se añadió un `break` que disminuyó significativamente la cantidad de iteraciones realizadas antes de terminar la búsqueda. Se puede observar el hexágono normal en contraste con el normalizado usado para la optimización en la figura 4.4, donde el hexágono se expande debido la normalización para que ocupe únicamente números enteros, tal como se describió anteriormente.

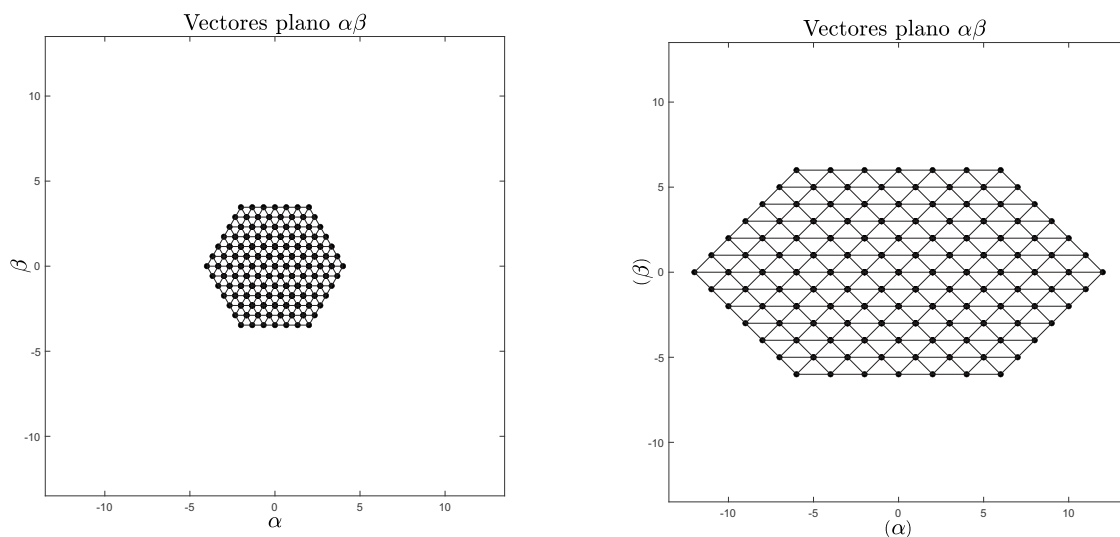


Figura 4.4. (a) *Héxagono tradicional;* (b) *Héxagono modificado para trabajar con números enteros.*

Esta optimización logra que esta sección del algoritmo pase de 35222 ciclos en el peor caso, es decir, que el algoritmo itere ambos bucles completamente, a 12593 ciclos (64% menos). Este caso se da cuando `v_she_a_k1`, `v_she_b_k1` y `v_she_c_k1` valen -3,-3 y 3 respectivamente. Si bien este es el peor caso, no quiere decir necesariamente que la cantidad de ciclos será de esa magnitud ni que la razón entre los ciclos de reloj de esta sección del algoritmo no optimizado con el optimizado será la misma. Por ejemplo, tomando aleatoriamente los valores 2, 0 y -1 se obtienen 5745 ciclos para el caso no optimizado y 2269 para el caso optimizado, lo que le da sentido a lo anteriormente dicho.

El uso de la memoria también jugó un papel importante en la optimización del tiempo

de ejecución del código. En este caso, debido principalmente al tamaño de las matrices HEX2, HEX3 y HEX4, correspondientes a 7×127 , 343×4 y 2401×3 respectivamente, fue necesario almacenar estas matrices en otro tipo de memoria diferente a la que se usa por defecto. Esto no fue problema, debido a que la configuración de la tarjeta BRain hace sencillo el cambio en el almacenamiento de variables. La idea principal es que las variables que menos se utilicen se desplacen a otras memorias no tan rápidas (externas al chip del *TMS320C6748*), en este caso la variable HEX4 fue elegida para el cambio de memoria debido a que participa relativamente pocas veces en el código dado que forma parte del algoritmo FCS-MPC que solo itera 7 veces. Además, su tamaño hace que no sea necesario involucrar más variables en el cambio de almacenamiento. La memoria elegida para su almacenamiento fue DSPL2RAM, dado que corresponde a una memoria interna que, si bien es más lenta con respecto a la memoria SHRAM donde viene el resto del programa y datos, no es tan lenta con respecto a la memoria externa DDR2, por lo que se consideró una elección adecuada.

Finalmente, se determinó que el nivel de optimización más rápido correspondía al nivel 2. Por otro lado, para la configuración de trade-off entre velocidad y tamaño, se eligió el nivel 4, ya que, por experiencia, esta opción permitía que el código se ejecutara de manera más eficiente.

En la tabla 4.2 se muestra un resumen de las modificaciones realizadas a lo largo de las distintas secciones del código. Es importante notar de que todas las optimizaciones realizadas logran una mejora mayor o igual al 60 %.

Tabla 4.2
OPTIMIZACIÓN

Modificación	Ciclos código original	Ciclos código optimizado	% Mejora
Cálculo referencias de corriente en k y $k+2$ y tensión inductor	3482	837	76 %
Cálculo $v_{ab_dq_k_mod}$ y $v_{ab_dq_k_ang}$	2010	175	91 %
Cálculo de wt_x_k1	776	307	60 %
Forma SHM	3685	788	79 %
Algoritmo Héxagono	35222	12593	64 %
Total	45175	14700	67 %

En cuanto a la frecuencia máxima de muestreo en la tarjeta BRAIn ésta fue de 29400 Hz, la cual corresponde a casi el doble con respecto a la frecuencia de muestreo que se llegó en la Universidad de Talca de 15000 Hz en la tarjeta F28379D. Esto consideró la ejecución del código completo, es decir, la etapa del algoritmo PLL junto con la técnica SHM-MPC, con la particularidad de que el tiempo de muestreo se estableció como constante. Esto demuestra tanto la alta capacidad de procesamiento de la tarjeta BRAIn como la

optimización del esquema de control demostrado a lo largo de este capítulo.

Capítulo 5

Ensayos en módulo HIL

Para la comprobación de los resultados obtenidos de la ejecución del Convertidor Multicelda en Cascada junto con el esquema de control SHM-MPC, se realizaron tres ensayos relacionados al seguimiento de la corriente de referencia, tanto para su magnitud como su fase, y los armónicos de corriente representados mediante el llamado THD (Total harmonic distortion). Estos resultados se compararon con la simulación en MATLAB-Simulink donde se ubicaban los archivos originales del código.

Los parámetros seleccionados para la implementación y simulación corresponden a $\alpha_{LF_PI} = 0,1190$, $kp = 2,494 \cdot 10^{-4}$ y $a = 0,98$ para la sección del PLL, y $Sgm_{mx_k} = 1 \cdot 10^{-3}$, $Lmb_k = 0,005$, $Sgm_{mn_k} = 1 \cdot 10^{-5}$ y $Lmb_u_k1 = 0,1 \cdot 10^{-5}$ para el control predictivo. En la tabla 5.1 se muestran los parámetros que se utilizaron para la tensión v_{cc} , la resistencia de línea y la inductancia de línea.

Tabla 5.1
PARÁMETROS

v_{cc} [V]	R [Ω]	L [H]
50	1	0.02

A continuación se mostrará una breve explicación de como se llevó a cabo la implementación del modelo del convertidor en el FPGA y, posteriormente, los distintos ensayos tanto de la implementación en la tarjeta BRAIn como en la simulación en MATLAB-Simulink:

5.1. Implementación de Convertidor Multicelda en Cascada en el FPGA

La implementación en el FPGA comienza en el software Vitis HLS, donde se programó el convertidor en código C++, tal como se detalló en el capítulo 3. Este programa permite transformar un código de alto nivel como C++ en una descripción de hardware en RTL (Register transfer level).

El archivo RTL generado por Vitis HLS se exportó para ser utilizado en el software Vivado. En Vivado se realizaron las configuraciones necesarias para poder generar un archivo wrapper, el cual encapsula el diseño RTL para su uso dentro de la tarjeta BRAIn. Este archivo wrapper se incorporó en el firmware proporcionado por el departamento, permitiendo

así que el diseño del convertidor se comunicara correctamente con otros componentes de la tarjeta.

Posteriormente, en Vivado, se realizó la síntesis del diseño, convirtiendo el diseño RTL en una red de puertas lógicas. Luego, se llevó a cabo la etapa de implementación, que incluye la colocación y el enrutamiento del diseño en el FPGA, determinando la ubicación física de cada componente y las rutas de interconexión.

Finalmente, se generó el bitstream, un archivo binario que contiene la configuración final del FPGA. Este bitstream se cargó en el FPGA mediante un cable JTAG. Este proceso finalizó la implementación, permitiendo que el FPGA funcione según el diseño del convertidor programado en Vitis HLS.

5.2. Escalón magnitud de corriente de referencia

Este ensayo se realizó con una referencia de fase de corriente igual a cero y un cambio instantáneo de la referencia de corriente desde el valor en el cual se tenía el máximo nivel del índice de modulación (m.i.) correspondiente a 1,15, es decir, cuando se tenía una referencia de corriente peak de $20[A]$ hasta un 25 % de la corriente máxima, es decir, cuando la corriente peak era $5[A]$.

5.2.1. Simulación

En el caso de la simulación, para el ensayo de escalón de magnitud de corriente de referencia se obtuvieron los resultados de la figura 5.1.

5.2.2. Implementación

En el caso de la implementación, para el ensayo de escalón de magnitud de corriente de referencia se obtuvieron los resultados de la figura 5.2.

Comparando los resultados con la simulación se observa que tiene un comportamiento muy similar, salvo por algunas diferencias que probablemente sean debidas a la diferencia en el modelo del convertidor, dado que en la simulación se consideran algunos detalles que no fueron representados en el modelo implementado en el FPGA. Esto se deduce a partir del gráfico de la tensión de salida del convertidor, donde no existe diferencia entre la tensión de salida de la implementación y de la simulación.

Una de diferencias que se logran ver se muestra en la corriente de línea para el caso de la simulación, dado que se tiene un pequeño retraso respecto a la corriente de referencia de aproximadamente $0,000323[s]$. Otra diferencia notoria se puede ver en la corriente en el eje “q” i-g-q-k de la simulación, donde antes del cambio de referencia no seguía completamente la referencia, sino que se desviaba $2,146[A]$ aproximadamente. Se puede notar también una diferencia en el tiempo T_{s_k} de la simulación donde el tiempo se desvía levemente del tiempo de referencia en $0,04844 \cdot 10^{-5}[s]$. Esto último puede deberse simplemente a una malinterpretación del valor obtenido en la simulación, considerando que el algoritmo del PLL es el mismo. Las diferencias en los resultados se pueden deber además a que, si bien se intentó hacer la implementación lo más parecido a la simulación, se entregaron varias versiones con distintas modificaciones en el código, por lo que quedaron

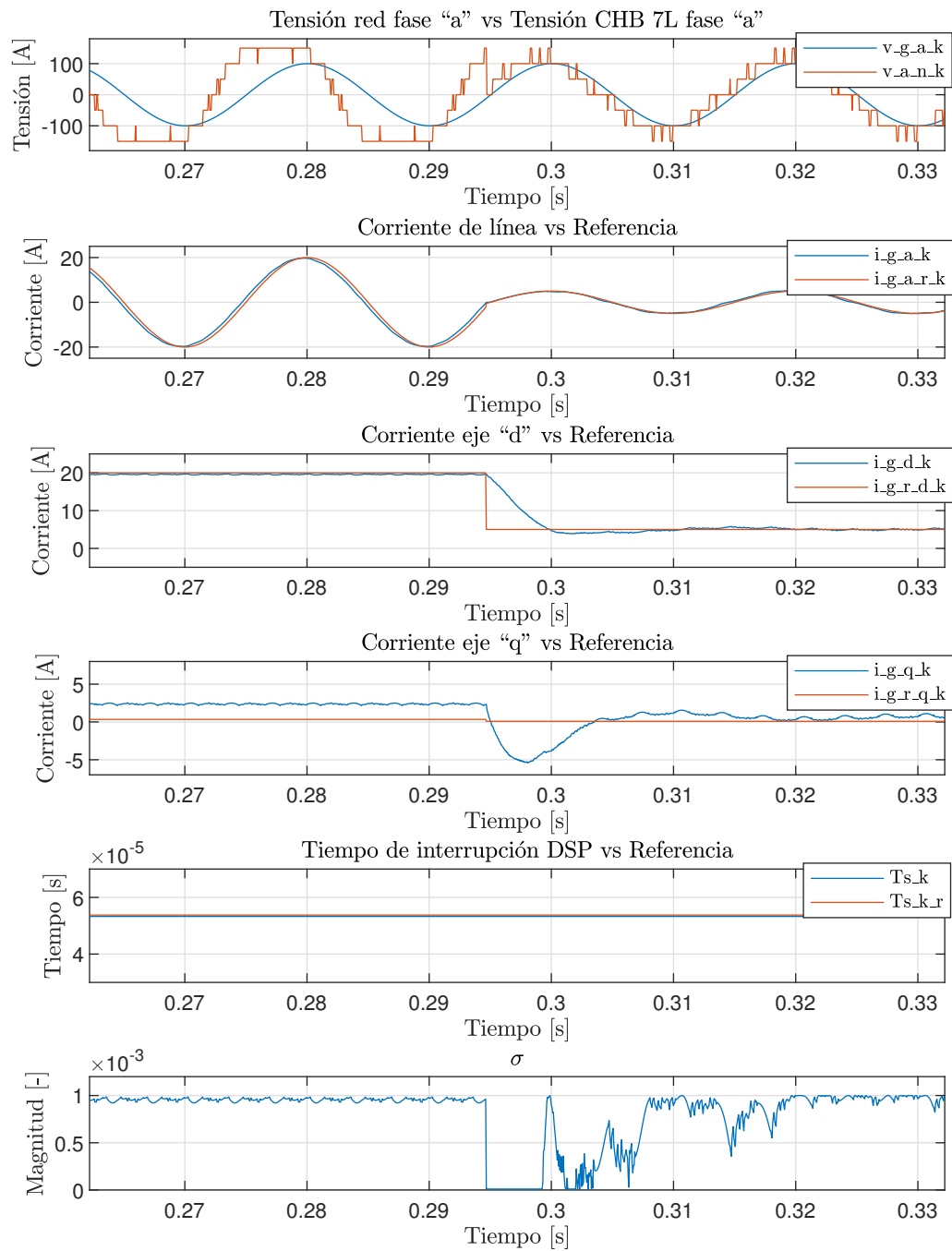


Figura 5.1. Ensayo escalón magnitud de corriente simulado.

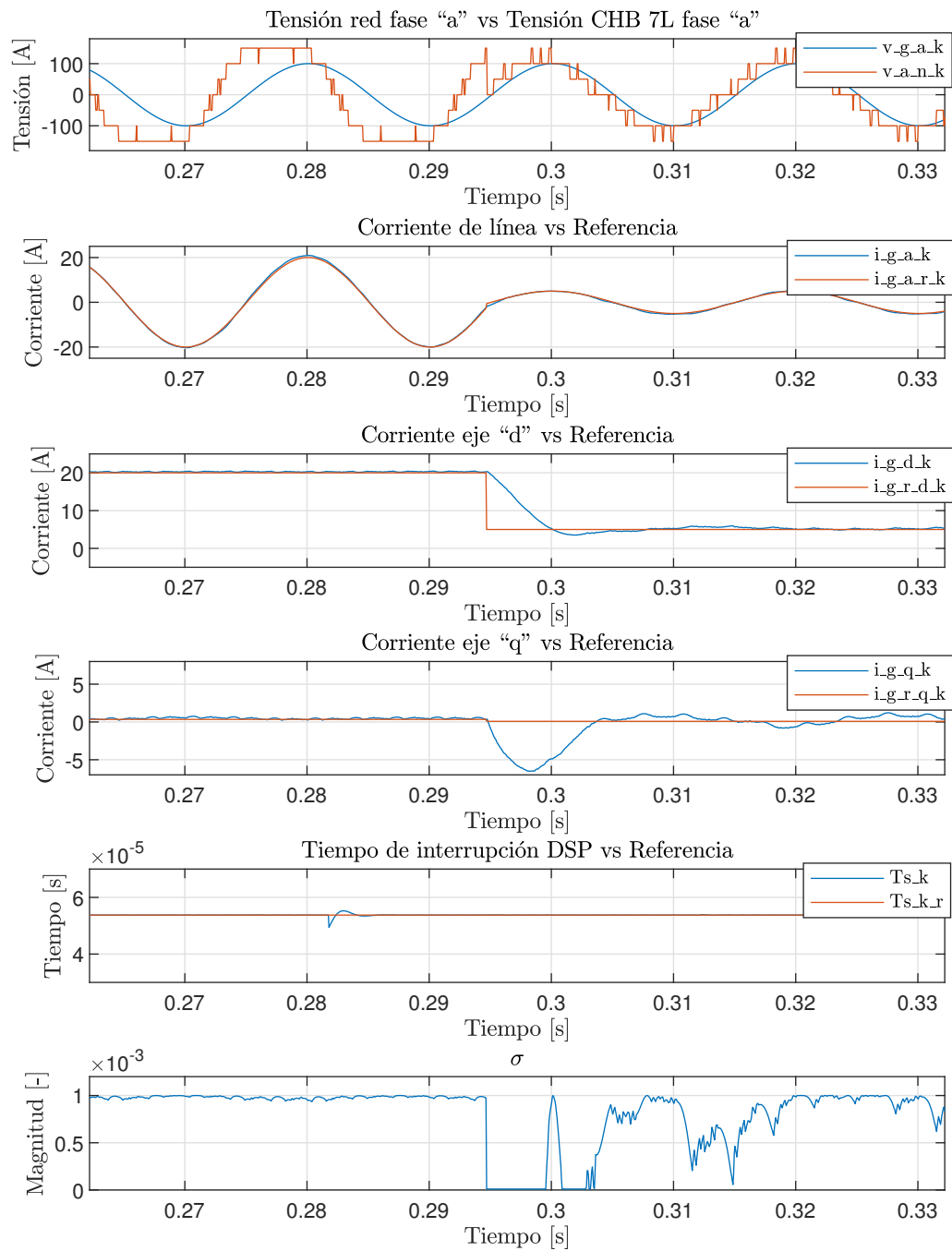


Figura 5.2. Ensayo escalón magnitud de corriente implementado.

algunas modificaciones de códigos anteriores. Aún así, la implementación muestra un gran desempeño.

5.3. Escalón ángulo de corriente de referencia

En este ensayo, se utilizó inicialmente una referencia de magnitud de corriente de $20[A]$ para alcanzar el nivel máximo de modulación con una referencia de fase de corriente de 0° . Posteriormente, se realizó un cambio instantáneo en la referencia de fase de corriente desde 0° hasta 70° , manteniendo la corriente de referencia en $20[A]$. Este cambio resultó en una disminución del índice de modulación de 1.15 a 0.4.

5.3.1. Simulación

En el caso de la simulación, para el ensayo de escalón de fase de corriente de referencia se obtuvieron los resultados de la figura 5.3.

5.3.2. Implementación

En el caso de la implementación, para el ensayo de escalón de fase de corriente de referencia se obtuvieron los resultados de la figura 5.4.

Al igual que los resultados del primer ensayo se observa que hay muy poca diferencia respecto de la simulación. En este ensayo la tensión de salida del convertidor sigue siendo igual tanto en la implementación como en la simulación.

Las diferencias observadas son las mismas que el caso del ensayo de escalón de magnitud de corriente de referencia con respecto a la corriente $i_{g_a_k}$ y el tiempo T_{s_k} . Sin embargo, en este caso la corriente en el eje “q” se desvía $2,12133[A]$ de la referencia, aunque es un valor prácticamente igual al obtenido en el otro ensayo. Las diferencias obtenidas se deben a las mismas razones ya mencionadas.

5.4. Análisis de THD respecto de índice de modulación

Este ensayo partió desde las mismas condiciones iniciales que la sección anterior, es decir, con una magnitud y una fase de la corriente de referencia de $20[A]$ y 0° respectivamente. Esto, tal como se dijo anteriormente, equivale a un índice de modulación de 1,15. La idea de este ensayo fue ir aumentando la referencia de fase de tal forma que el índice de modulación fuera disminuyendo en pasos de 0,02, con el objetivo de ir midiendo los THD's de cada corriente medida.

Tanto para la simulación como para la implementación, se compararán los resultados conforme a la norma IEEE Std. 519 2022, teniendo en cuenta lo expuesto al inicio del capítulo 4.

5.4.1. Simulación

En el caso de la simulación, se obtuvieron los siguientes resultados ilustrados en las figuras 5.5, 5.6 y 5.7.

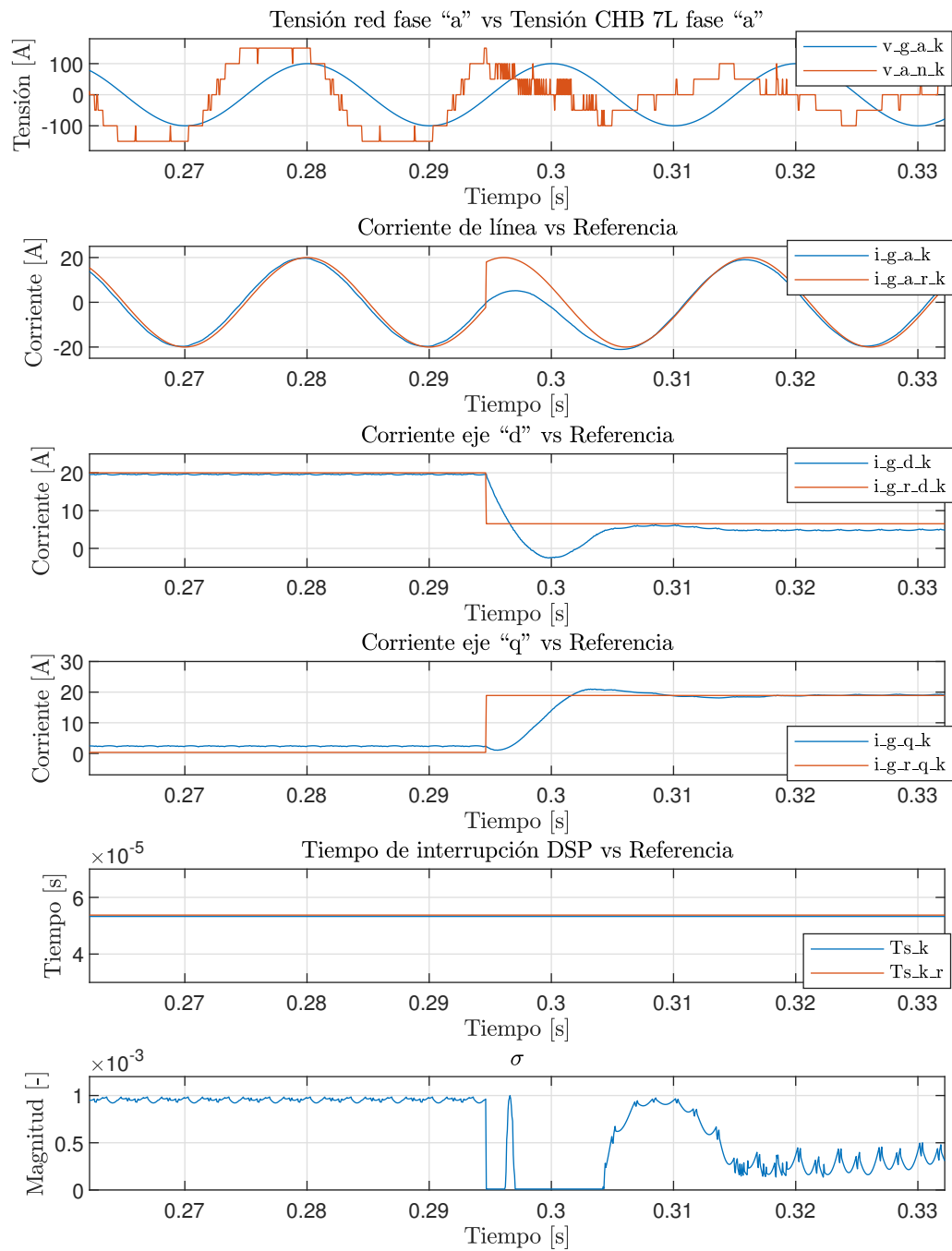


Figura 5.3. Ensayo escalón fase de corriente simulado.

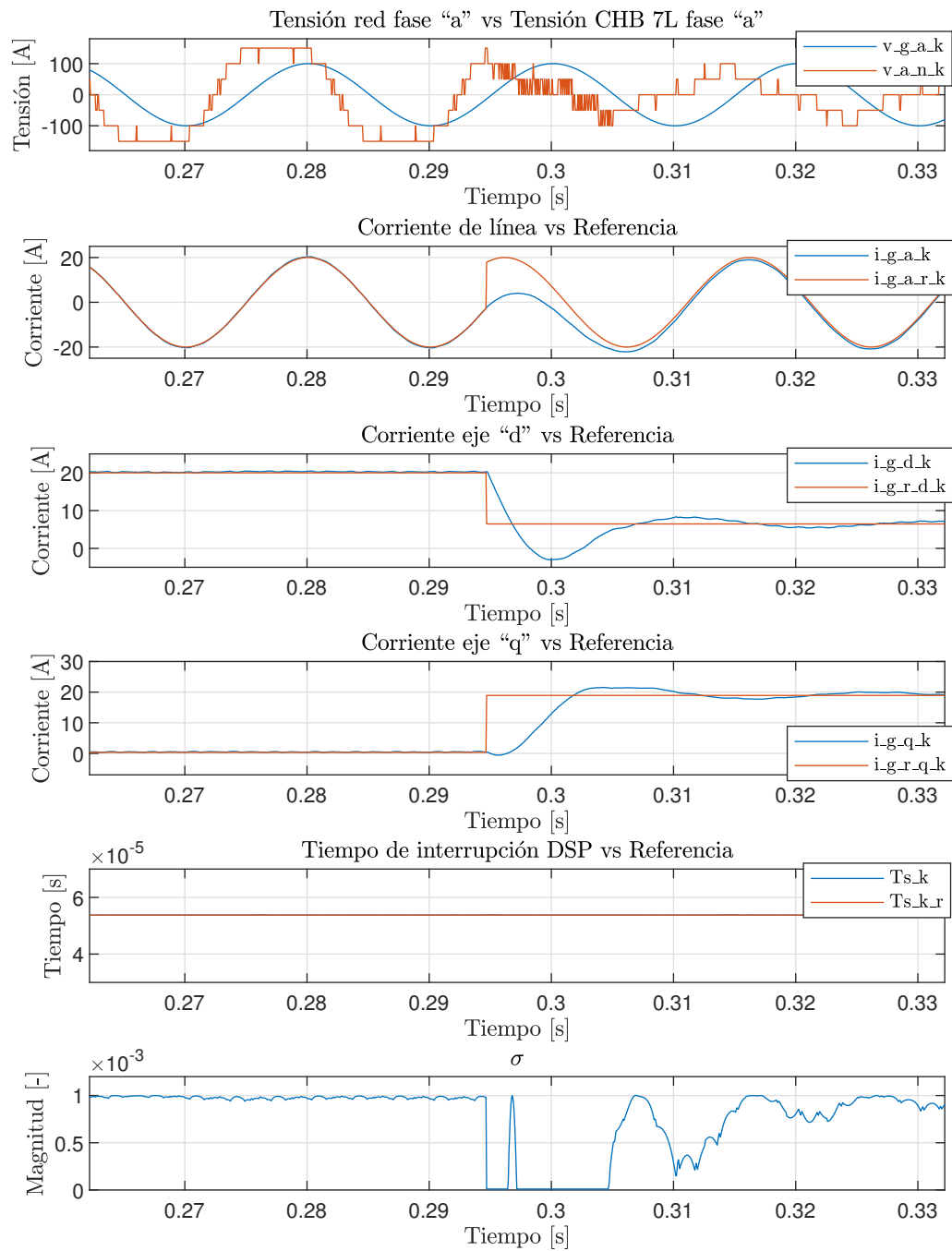


Figura 5.4. *Ensayo escalón fase de corriente implementado.*

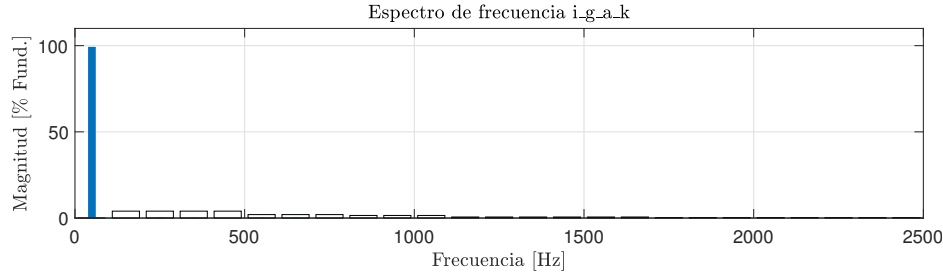


Figura 5.5. Espectro de frecuencia $i_{g_a_k}$ ($m.i. = 0,41$).

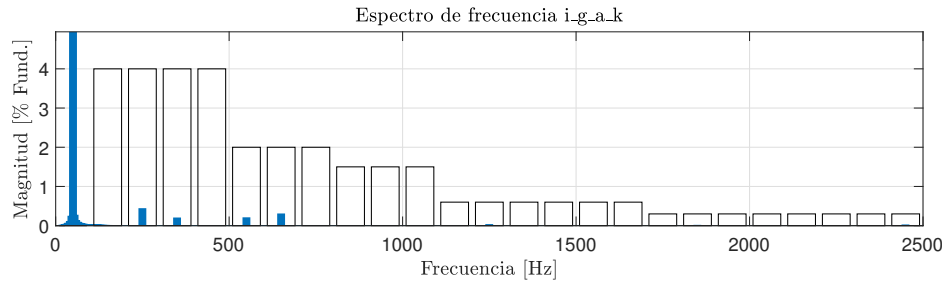


Figura 5.6. Espectro de frecuencia $i_{g_a_k}$ - Vista Ampliada ($m.i. = 0,41$).

La figura 5.5 corresponde al espectro de corriente $i_{g_a_k}$ con un índice de modulación igual a 0,41, lo cual se traduce en una referencia de desfase de $69,67^\circ$. Las barras azules corresponden al valor obtenido y las barras transparentes con bordes negros el límite establecido para los armónicos de acuerdo a la norma antes mencionada.

La figura 5.6 es simplemente una vista ampliada de los armónicos con el objetivo de ver en mayor detalle los armónicos de mayor orden. Para este caso específico se observa una predominancia de los armónicos $250Hz$, $350Hz$, $550Hz$ y $650Hz$, sin embargo, están muy por debajo del límite establecido por la norma.

Por otro lado, la figura 5.7 corresponde a la variación del THD de corriente tal como se comentó anteriormente calculado mediante la formula

$$THD_I = 100 \frac{\sqrt{\sum_{n=2}^N I_{nRMS}^2}}{I_{1RMS}}. \quad (5.1)$$

Se observa que para cada índice de modulación el perfil de armónicos se mantiene relativamente constante. Además, en todos los casos se cumple con la norma.

5.4.2. Implementación

En el caso de la implementación, se obtuvieron los siguientes resultados ilustrados en las figuras 5.8, 5.9 y 5.10.

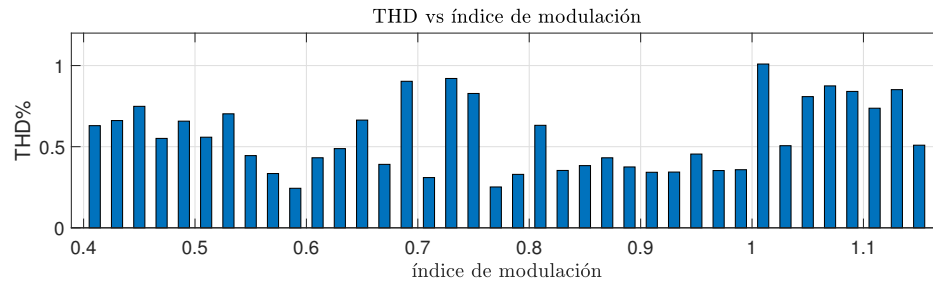


Figura 5.7. *THD de corriente vs índice de modulación.*

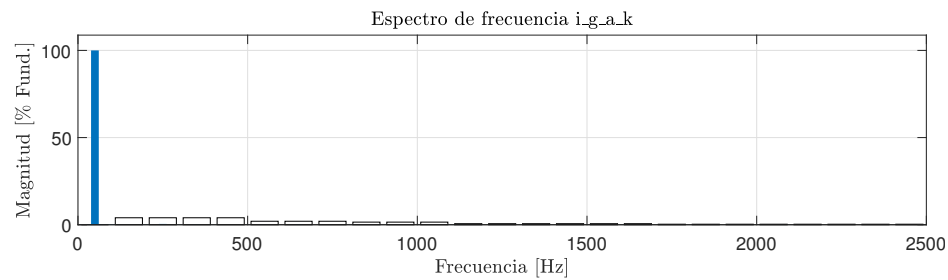


Figura 5.8. *Espectro de frecuencia $i_{g_a_k}$ ($m.i. = 0,41$).*

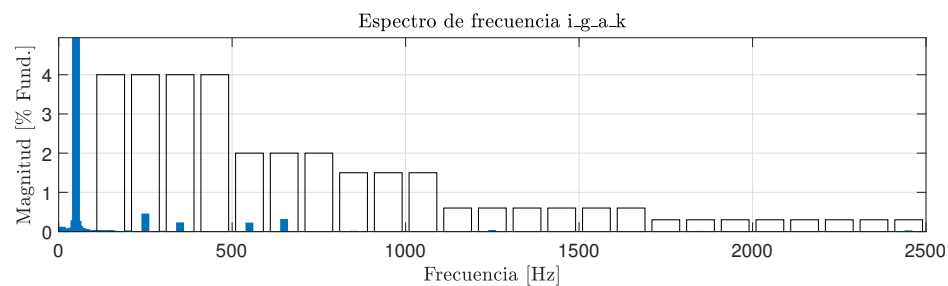


Figura 5.9. *Espectro de frecuencia $i_{g_a_k}$ - Vista Ampliada ($m.i. = 0,41$).*

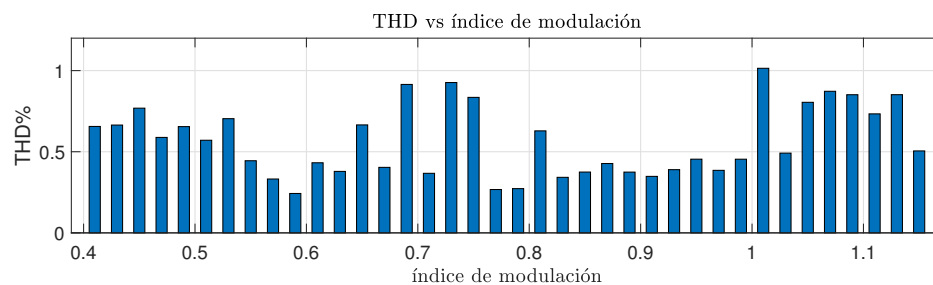


Figura 5.10. *THD de corriente vs índice de modulación.*

La figura 5.8 es representativa de lo mismo dicho de la sección anterior, pero en la implementación. Esta figura es prácticamente idéntica a la obtenida en la simulación.

La figura 5.9 es, al igual que en la sección anterior, una vista ampliada de los armónicos con el objetivo de ver en mayor detalle los armónicos de mayor orden. Para este caso, al igual que antes, se observa una predominancia de los armónicos 250Hz , 350Hz , 550Hz y 650Hz , los cuales siguen estando bastante por debajo de la norma. Esta figura no tiene

una diferencia observable con respecto a la obtenida en la simulación.

Por otro lado, la figura 5.10 corresponde a la variación del THD de corriente tal como se comentó anteriormente calculado mediante la formula (5.1). Esta figura, al igual que la obtenida en la simulación mantiene el THD sin grandes cambios a medida que se variaba el índice de modulación. Además, los gráficos tanto de simulación como de implementación son prácticamente idénticos. Como se observa, también se cumple con la norma para cada uno de los índices de modulación.

Capítulo 6

Conclusiones.

La solución propuesta originalmente consistió en hacer bastantes ajustes al código del esquema de control utilizado, tal como quedó demostrado a lo largo de esta memoria. Uno de las secciones que más disminuyó el tiempo de ejecución del esquema de control fue la región óptima de operación mediante los sub-hexágonos, la cual al comienzo solo estaba desarrollada para el caso de una celda por fase haciendo que el esquema de control fuera imposible de implementar debido a que el FCS-MPC se debía evaluar para cada posible combinación de conmutación, haciendo que el costo computacional fuera excesivamente alto. Sin embargo, a lo largo de esta memoria se logró desarrollar para el caso de tres celdas por fase, lo que permitió acercarse a tiempos de simulación que cumplieran con las condiciones mínimas requeridas para la ejecución del control. Esta sección además fue la última en optimizarse, haciendo que se acabaran las posibilidades de optimización y, por lo tanto, de implementación del esquema de control. Sin embargo, gracias a la actualización de los registros de configuración de la plataforma realizados desde el Centro Avanzado de ingeniería eléctrica y electrónica, que permitieron aumentar la frecuencia de reloj del DSP desde 300 MHz a 456 MHz, se logró implementar sin problemas aún con condiciones imprevistas en el control. El tiempo de muestreo máximo obtenido por la tarjeta BRAIn fue de 29400 Hz, lo cual deja la posibilidad de conmutar a mayores frecuencias y, por lo tanto, reducir la cantidad de contenido armónico.

Los resultados obtenidos de la implementación del esquema de control fueron bastante satisfactorios. Si bien los resultados de la simulación e implementación no son exactamente iguales, esto se debió a que los modelos del Convertidor Multicelda en Cascada fueron ligeramente distintos, pero considerando que el objetivo de este trabajo no estuvo enfocado en un modelo adecuado del convertidor, sino en la correcta implementación y optimización del esquema de control con un modelo de convertidor que reflejara de manera aproximada la dinámica de uno real, es que esto no es considerado importante en la implementación. Dicho esto, la comparación de los resultados se aproxima lo bastante, considerando además que uno de los parámetros más importantes a considerar es la salida de tensión por fase del convertidor, la cual es prácticamente igual, salvo por algunas fluctuaciones despreciables. Observando únicamente los resultados obtenidos de la implementación se observa que sigue sin problemas en todos los ensayos cada una de las referencias de corriente, el tiempo de simulación y la tensión de salida por fase del convertidor sigue sin problemas la forma de la técnica SHM. Los resultados obtenidos del THD y espectro de frecuencia de corriente fueron consistentes comparados a la simulación, lo que reafirmó la correcta implementación

del esquema de control. Además, estos resultados cumplieron con la norma, que era uno de los objetivos más importantes de este esquema de control.

Si bien se logró la implementación del esquema de control para tres celdas en cascada, la implementación de un número mayor requiere superar un obstáculo. Este obstáculo a considerar es el aumento exponencial en la cantidad posible de combinaciones de las celdas que pueden ser candidatos para el algoritmo FCS-MPC, si bien esto se puede mejorar mediante la región óptima de operación mediante los sub-hexágonos, esto requiere un mayor uso de la memoria, lo que puede derivar en el uso de memorias más lentas o directamente la incapacidad de almacenamiento de tal cantidad de datos. Además la búsqueda se hace más compleja haciendo que posiblemente aún con esta búsqueda no se logre ejecutar el algoritmo en el tiempo deseado.

La implementación realizada en este trabajo puede ser relativamente fácil de lograr para el caso del convertidor real. Esto se lograría mediante el uso de las entradas ADC de la plataforma, algunos cambios en la forma que se implementa el modelo del convertidor en el FPGA y cambiando levemente el algoritmo de forma que se procesen correctamente estas señales. Esta facilidad de implementación justamente es una de las ventajas de la plataforma BRAIn.

Apéndice A

Código esquema de control SHM-MPC, figuras y tablas

A continuación se muestra el código utilizado para el esquema de control SHM-MPC. Además se incluyen los códigos en MATLAB para la obtención de los gráficos mostrados a lo largo de toda la tesis y los códigos para crear las tablas escritas en este documento.

A.1. Code Composer

A.1.1. Archivo config.c

```
#include "aceC6748.h"

#include "ace_interrupt.h"

#include "PWM_app.h"

extern void Init_Platform(void)
{
    ACE_PLL_initPllOut0(DSP456MHz);
    ACE_EMIF_init(EMIFA_SOURCE_PLLOSYSCLK3);
}

extern void Init_Interrupts(void)
{
    ACE_INT_initDspIntController();

    ACE_EMIF_initFpgaSyncInterruption(PWM_fpgaSyncInterrupt_isr);
    ACE_INT_enableDspGlobalInt();

    INTOENABLE = INTO_ADCO_MASK;

    INTOCLRFLG = 0xFFFF;
    INT1CLRFLG = 0xFFFF;
```

```

    INT2CLRFLG = 0xFFFF;
}

extern void Init_Mod(MO_MAX)
{
    CLKSTBDIV0  = 10000;

    CLKSTBDIV1  = 20;

    CLKSTBDIV2  = 1;

    MODNEGLOG012 = 0xFFFF;
    MODNEGLOG345 = 0xFFFF;
    MOCTRL      = MOD_MODE_UD | MOD_CLKSEL_2 | MOD_INT_MINMAX;
    MOMAXCNT    = MO_MAX;
    MOPWMCTRL   = MOD_PWM_EN | MOD_PWM_DT(0.5);
    MOLOAD      = 0;

    M1CTRL = MOD_INISLO_UP | MOD_SYNC_TO_MASTER | MOD_MODE_UD
        | MOD_CLKSEL_2;
    M1MAXCNT = MO_MAX;
    M1PWMCTRL = MOD_PWM_EN | MOD_PWM_DT(0.5);
    M1LOAD = 0;

    M2CTRL = MOD_INISLO_UP | MOD_SYNC_TO_MASTER | MOD_MODE_UD
        | MOD_CLKSEL_2;
    M2MAXCNT = MO_MAX;
    M2PWMCTRL = MOD_PWM_EN | MOD_PWM_DT(0.5);
    M2LOAD = 0;

    M3CTRL = MOD_INISLO_UP | MOD_SYNC_TO_MASTER | MOD_MODE_UD
        | MOD_CLKSEL_2;
    M3MAXCNT = MO_MAX;
    M3PWMCTRL = MOD_PWM_EN | MOD_PWM_DT(0.5);
    M3LOAD = 0;

    M4CTRL = MOD_INISLO_UP | MOD_SYNC_TO_MASTER | MOD_MODE_UD
        | MOD_CLKSEL_2;
    M4MAXCNT = MO_MAX;
    M4PWMCTRL = MOD_PWM_EN | MOD_PWM_DT(0.5);
    M4LOAD = 0;

    M5CTRL = MOD_INISLO_UP | MOD_SYNC_TO_MASTER | MOD_MODE_UD
        | MOD_CLKSEL_2;
    M5MAXCNT = MO_MAX;
    M5PWMCTRL = MOD_PWM_EN | MOD_PWM_DT(0.5);
    M5LOAD = 0;

```

```
}

extern void Init_Adc(void)
{
    ADCOCTRL = ADC_ST_MOMAX | ADC_ST_MOMIN | ADC_EOCINT_EN
              | ADC_SEQCNV;

    ADC1CTRL = ADC_ST_MOMIN | ADC_ST_MOMAX | ADC_SEQCNV ;

    ADC3CTRL = ADC_ST_MOMIN | ADC_ST_MOMAX | ADC_SEQCNV ;

    ADCOCLRFLG = 0xFFFF;
    ADCOCLRFLG = 0xFFFF;
    ADCOCLRFLG = 0xFFFF;
}

extern void Init_Iop(void)
{
    IOPODIR = 0x0010;

    IOP0EEN = 0x0001;

    IOP0EVT = 0xFFFF;

    IOP0OUT = 0xAAAA;

    IOP1DIR = 0x0000;
    IOP2DIR = 0x0000;
    IOP3DIR = 0x0000;
    IOP4DIR = 0x0000;
    IOP5DIR = 0x0000;
    IOP6DIR = 0x0000;
    IOP7DIR = 0x0000;
    IOP8DIR = 0x0000;

    IOP1EEN = 0x0000;
    IOP2EEN = 0x0000;
    IOP3EEN = 0x0000;
    IOP4EEN = 0x0000;
    IOP5EEN = 0x0000;
    IOP6EEN = 0x0000;
    IOP7EEN = 0x0000;
    IOP8EEN = 0x0000;

    ACE_PINMUX_ESS_writeRegister(ACE_PINMUX_ESS_REG_00, 0x5555);
    ACE_PINMUX_ESS_writeRegister(ACE_PINMUX_ESS_REG_01, 0x5555);
    ACE_PINMUX_ESS_writeRegister(ACE_PINMUX_ESS_REG_02, 0x5555);
    ACE_PINMUX_ESS_writeRegister(ACE_PINMUX_ESS_REG_03, 0x5555);
}
```

```

    ACE_PINMUX_ESS_writeRegister(ACE_PINMUX_ESS_REG_04, 0x5555);

    ACE_PINMUX_ESS_writeRegister(ACE_PINMUX_ESS_REG_05, 0x0000);
    ACE_PINMUX_ESS_writeRegister(ACE_PINMUX_ESS_REG_06, 0x0000);
    ACE_PINMUX_ESS_writeRegister(ACE_PINMUX_ESS_REG_07, 0x0000);
    ACE_PINMUX_ESS_writeRegister(ACE_PINMUX_ESS_REG_08, 0x0000);
    ACE_PINMUX_ESS_writeRegister(ACE_PINMUX_ESS_REG_09, 0x0000);

    ACE_PINMUX_ESS_writeRegister(ACE_PINMUX_ESS_REG_10, 0x5555);
    ACE_PINMUX_ESS_writeRegister(ACE_PINMUX_ESS_REG_11, 0x5555);
    ACE_PINMUX_ESS_writeRegister(ACE_PINMUX_ESS_REG_12, 0x5555);
    ACE_PINMUX_ESS_writeRegister(ACE_PINMUX_ESS_REG_13, 0x5555);
    ACE_PINMUX_ESS_writeRegister(ACE_PINMUX_ESS_REG_14, 0x5555);
}

extern void Init_DS_LED(void)
{
    ACE_USRIO_setupDs0();
    ACE_USRIO_setupDs1();
    ACE_USRIO_setupDs2();
    ACE_USRIO_setupDs3();

    ACE_USRIO_setupLd0();
    ACE_USRIO_setupLd1();
    ACE_USRIO_setupLd2();
    ACE_USRIO_setupLd3();
}

```

A.1.2. Archivo app.c

```

#include "stdint.h"

extern void PWM_fpgaSyncInterrupt_isr(void);

#define PI 3.1415926535897932384626433832795
#define DESFASE 2.0943951023931954923084289221863
#define DOS_TER 0.66666666666666666666666666666666
#define UN_TER 0.33333333333333333333333333333333
#define UN_RAIZTER 0.5773502691896257645091487805019

#define W_BASE_1 0.0031830988618379066866503901422902
#define W_BASE 314.15926535897932384626433832795

#define I_RESTA 2
#define I_MUL 0.000244140625
#define V_RESTA 1
#define V_MUL 0.0001220703125

```

```

#define W_RESTA                2
#define W_MUL                   0.000244140625

#define VDC_MUL                 0.037874597259831041529132278355974

#define DATOS                   20000

#define BUFFER_SIZE             93
#define CONVRADDEG              118.411277660370129812049519949150

```

A.1.3. Archivo main.c

```

#pragma DATA_SECTION(HEX4, ".mi_secc")
#include "aceC6748.h"
#include "assert.h"
#include "PWM_app.h"
#include "atan2sp.h"
#include "sqrtsp.h"
#include "sinsp.h"
#include "cosp.h"

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

void main(void){
    Init_Platform();
    Init_Mod(MO_MAX);
    Init_Adc();
    Init_Iop();
    Init_DS_LED();
    Init_Interrupts();

    ACE_USRIO_turnOffLd(ACE_USRIO_LD0);
    ACE_USRIO_turnOffLd(ACE_USRIO_LD1);
    ACE_USRIO_turnOffLd(ACE_USRIO_LD2);
    ACE_USRIO_turnOffLd(ACE_USRIO_LD3);
    MODTRIP012      = 0xFFFF;
    RST_CONV         = 0xFFFF;
    START_CONV       = 0x0000;
    TRIGGER_CONV     = 0x0000 ;

    while(1)
    {
    }
}

```

```

}

extern void PWM_fpgaSyncInterrupt_isr(void)
{
    ACE_INT_disableDspGlobalInt();

    interrupcion++;

    ia_read  = IA_CONV;
    ib_read  = IB_CONV;
    ic_read  = IC_CONV;
    va_read  = VA_RED;
    vb_read  = VB_RED;

    v_g_a_k = ((int)va_read * V_MUL - V_RESTA)*1.25;
    v_g_b_k = ((int)vb_read * V_MUL - V_RESTA)*1.25;

    float cos_wt_a_k, cos_wt_b_k, cos_wt_c_k, sin_wt_a_k,
          sin_wt_b_k, sin_wt_c_k, Ts_k;
    float wt_a_k;
    float v_g_c_k;
    float cos0_k;
    float cos120_k;
    float cos240_k;
    float sin0_k;
    float sin120_k;
    float sin240_k;
    float Y_k;
    float Yf_k;

    float v_g_a_SP_k;
    float v_g_b_SP_k;
    float v_g_c_SP_k;

    v_g_c_k = - v_g_a_k - v_g_b_k;

    cos0_k = sin_N[indexPLlc0_k - 1];
    cos120_k = sin_N[indexPLlc120_k - 1];
    cos240_k = sin_N[indexPLlc240_k - 1];
    sin0_k = sin_N[indexPLls0_k - 1];
    sin120_k = sin_N[indexPLls120_k - 1];
    sin240_k = sin_N[indexPLls240_k - 1];

    Y_k = v_g_a_k*cos0_k + v_g_b_k*cos120_k + v_g_c_k*cos240_k;

    Yf_k = Yf_k1 + alpha_LF_PI*(Y_k - Yf_k1);
    Ts_k = Ts_1k - kp*Yf_k + a*kp*Yf_k1;

```

```

if(Ts_k > MAX_ACT){
    Ts_k = MAX_ACT;
}

if(Ts_k < MIN_ACT){
    Ts_k = MIN_ACT;
}

Ts_1k = Ts_k;
Yf_k1 = Yf_k;

MO_MAX = Ts_1k * 100000000;

indexPLLC0_k = indexPLLC0_k + 1.0f;
indexPLLC120_k = indexPLLC120_k + 1.0f;
indexPLLC240_k = indexPLLC240_k + 1.0f;
indexPLLS0_k = indexPLLS0_k + 1.0f;
indexPLLS120_k = indexPLLS120_k + 1.0f;
indexPLLS240_k = indexPLLS240_k + 1.0f;

if(indexPLLC0_k == P_1){
    indexPLLC0_k = 1;
}

if(indexPLLC120_k == P_1){
    indexPLLC120_k = 1;
}

if(indexPLLC240_k == P_1){
    indexPLLC240_k = 1;
}

if(indexPLLS0_k == P_1){
    indexPLLS0_k = 1;
}

if(indexPLLS120_k == P_1){
    indexPLLS120_k = 1;
}

if(indexPLLS240_k == P_1){
    indexPLLS240_k = 1;
}

cos_wt_b_k = sin120_k;

```

```

cos_wt_c_k = sin240_k;
sin_wt_a_k = -cos0_k;
sin_wt_b_k = -cos120_k;
sin_wt_c_k = -cos240_k;
wt_a_k = (PI2*(indexPLLc0_k-1)*u_tsd-PI);

//*****//
//*****TERMINO PLL*****//
//*****//

i_g_a_k = (((int)ia_read) * I_MUL - I_RESTA )*16.666666;
i_g_b_k = (((int)ib_read) * I_MUL - I_RESTA )*16.666666;
i_g_c_k = (((int)ic_read) * I_MUL - I_RESTA )*16.666666;

float TL;
float TRL;
float wL;
float wT2;
float wT;
float v_g_d_k;

float v_g_a_k1;
float v_g_b_k1;
float v_g_c_k1;

float wt_b_k;
float wt_c_k;

float wt_a_k1;
float wt_b_k1;
float wt_c_k1;

float wt_a_k2;
float wt_b_k2;

float i_g_a_r_k;
float i_g_b_r_k;
float i_g_c_r_k;

float i_g_a_r_k2;
float i_g_b_r_k2;
float i_g_c_r_k2;

float i_g_d_k;
float i_g_r_d_k;
float i_g_q_k;

```

```
float i_g_r_q_k;
float i_g_d_k1;
float i_g_r_d_k1;

float I_r_k1;

float v_r_a_k;
float v_r_b_k;
float v_r_c_k;

float v_L_a_k;
float v_L_b_k;
float v_L_c_k;

float v_an_k;
float v_bn_k;
float v_cn_k;

float v_a_n_k;
float v_b_n_k;
float v_c_n_k;

float v_o_n_k;

float i_g_a_k1;
float i_g_b_k1;
float i_g_c_k1;

float delta_i_k1;
float sigma_d_k1;

float v_ab_d_k;
float v_ab_q_k;
float v_ab_dq_mod_k;
float v_ab_dq_ang_k;

float M_she_k1;
float max_m;
float min_m;

int v_she_a_k1, v_she_b_k1, v_she_c_k1;

float K1;
float K2;
float K3;
float K4;
float K5;
float K6;
```

```
float J_op_k;  
float J_i_g_abc_op_k;  
float J_v_g_abc_op_k;  
float J_u_g_abc_op_k;  
int Sa_op_k1;  
int Sb_op_k1;  
int Sc_op_k1;
```

```
float v_a_n_k1;  
float v_b_n_k1;  
float v_c_n_k1;
```

```
float v_o_n_k1;
```

```
float K7;  
float K8;  
float K9;
```

```
float i_g_a_k2;  
float i_g_b_k2;  
float i_g_c_k2;
```

```
float J_i_g_a_k;  
float J_i_g_b_k;  
float J_i_g_c_k;
```

```
float J_i_g_abc_k;
```

```
float J_v_g_a_k;  
float J_v_g_b_k;  
float J_v_g_c_k;
```

```
float J_v_g_abc_k;
```

```
float J_u_g_a_k;  
float J_u_g_b_k;  
float J_u_g_c_k;
```

```
float J_u_g_abc_k;
```

```
float J_k;
```

```
int indcp1;  
int indcp2;  
int indcp3;  
int indcp4;  
int indcp5;
```

```

    int indcp6;
    int indcp7;

    //PRUEBA CAMBIO MAGNITUD CORRIENTE
/*
    if (interrupcion > 5572){ //20 a 5 y
        //ref_in = 20;
        //phi_r_k = 70*PI/180;
    }
*/
    //phi_r_k = interrupcion*6.7875854610059477246562333986522e-5;
    //0.003889*PI/180;

    TL = Ts_0k/L_1;
    TRL = TL*R_1;
    wL = L_1*w_0;
    wT2 = Ts_0k*w_2;
    wT = Ts_0k*w_0;

    v_g_d_k = V_g_fn;

    v_g_a_k = v_g_d_k*cos_wt_a_k;
    v_g_b_k = v_g_d_k*cos_wt_b_k;
    v_g_c_k = v_g_d_k*cos_wt_c_k;

    v_g_a_k1 = v_g_a_k;
    v_g_b_k1 = v_g_b_k;
    v_g_c_k1 = v_g_c_k;

    wt_b_k = wt_a_k - PI23;
    wt_c_k = wt_a_k + PI23;

    wt_a_k1 = wt_a_k + wT;
    wt_b_k1 = wt_b_k + wT;
    wt_c_k1 = wt_c_k + wT;

    wt_a_k2 = wt_a_k + wT2;
    wt_b_k2 = wt_b_k + wT2;

    i_g_a_r_k = ref_in*cossp(wt_a_k + phi_r_k);
    i_g_b_r_k = ref_in*cossp(wt_b_k + phi_r_k);
    i_g_c_r_k = - i_g_a_r_k - i_g_b_r_k;

    i_g_a_r_k2 = ref_in*cossp(wt_a_k2 + phi_r_k);
    i_g_b_r_k2 = ref_in*cossp(wt_b_k2 + phi_r_k);
    i_g_c_r_k2 = - i_g_a_r_k2 - i_g_b_r_k2;

    i_g_d_k = DOS_TER*(i_g_a_k*cos_wt_a_k + i_g_b_k*cos_wt_b_k

```

```

    + i_g_c_k*cos_wt_c_k);
i_g_r_d_k = DOS_TER*(i_g_a_r_k*cos_wt_a_k + i_g_b_r_k*cos_wt_b_k
    + i_g_c_r_k*cos_wt_c_k);
i_g_q_k = DOS_TER*(- i_g_a_k*sin_wt_a_k - i_g_b_k*sin_wt_b_k
    - i_g_c_k*sin_wt_c_k);
i_g_r_q_k = DOS_TER*(-i_g_a_r_k*sin_wt_a_k - i_g_b_r_k*sin_wt_b_k
    - i_g_c_r_k*sin_wt_c_k);

i_g_d_k1 = i_g_d_k;
i_g_r_d_k1 = i_g_r_d_k;

I_r_k1 = ref_in;

v_r_a_k = i_g_a_r_k*R_1;
v_r_b_k = i_g_b_r_k*R_1;
v_r_c_k = i_g_c_r_k*R_1;

v_L_a_k = ref_in*wL*sinsp(wt_a_k1 + phi_r_k);
v_L_b_k = ref_in*wL*sinsp(wt_b_k1 + phi_r_k);
v_L_c_k = ref_in*wL*sinsp(wt_c_k1 + phi_r_k);

v_an_k = v_r_a_k - v_L_a_k + v_g_a_k;
v_bn_k = v_r_b_k - v_L_b_k + v_g_b_k;
v_cn_k = v_r_c_k - v_L_c_k + v_g_c_k;

v_a_n_k = V_dc * Sa_op_k;
v_b_n_k = V_dc * Sb_op_k;
v_c_n_k = V_dc * Sc_op_k;

v_o_n_k = UN_TER*(v_a_n_k + v_b_n_k + v_c_n_k);

i_g_a_k1 = (1-TRL)*i_g_a_k + TL*(v_a_n_k - v_g_a_k - v_o_n_k);
i_g_b_k1 = (1-TRL)*i_g_b_k + TL*(v_b_n_k - v_g_b_k - v_o_n_k);
i_g_c_k1 = (1-TRL)*i_g_c_k + TL*(v_c_n_k - v_g_c_k - v_o_n_k);

delta_i_k1 = (i_g_d_k1 - i_g_r_d_k1)*(i_g_d_k1 - i_g_r_d_k1)/I_r_k1;
sigma_d_k1 = Sgm_mx_k - Lmb_k*delta_i_k1;

if (sigma_d_k1 < Sgm_mn_k){
    sigma_d_k1 = Sgm_mn_k;
}

v_ab_d_k = DOS_TER*(v_an_k*cos_wt_a_k + v_bn_k*cos_wt_b_k
    + v_cn_k*cos_wt_c_k);
v_ab_q_k = DOS_TER*(-v_an_k*sin_wt_a_k - v_bn_k*sin_wt_b_k
    - v_cn_k*sin_wt_c_k);
v_ab_dq_mod_k = sqrtsp(v_ab_d_k*v_ab_d_k + v_ab_q_k*v_ab_q_k);

```

```

v_ab_dq_ang_k = atan2sp(v_ab_q_k, v_ab_d_k);
M_she_k1 = v_ab_dq_mod_k*V_dc_inv*UN_TER;

max_m = 1.3;
min_m = 0.4;

if (M_she_k1 >= max_m){
    M_she_k1 = max_m;
}
if (M_she_k1 <= min_m){
    M_she_k1 = min_m;
}

float wt_she_a_k1;
float wt_she_b_k1;
float wt_she_c_k1;
float G;
int a1;
int a2;
int a3;
int a4;
int a5;
int a6;
int a7;
int a8;
int a9;

int TRa;
int Va;

int TRb;
int Vb;

int TRc;
int Vc;

float ang_aux_a = wt_a_k1 + PI_2 + v_ab_dq_ang_k;
float ang_aux_b = wt_b_k1 + PI_2 + v_ab_dq_ang_k;
float ang_aux_c = wt_c_k1 + PI_2 + v_ab_dq_ang_k;

if (ang_aux_a < 0) ang_aux_a += PI2;
else if (ang_aux_a >= PI2) ang_aux_a -= PI2;

if (ang_aux_b < 0) ang_aux_b += PI2;
else if (ang_aux_b >= PI2) ang_aux_b -= PI2;

if (ang_aux_c < 0) ang_aux_c += PI2;
else if (ang_aux_c >= PI2) ang_aux_c -= PI2;

```

```

wt_she_a_k1      = ang_aux_a;
wt_she_b_k1      = ang_aux_b;
wt_she_c_k1      = ang_aux_c;

v_she_a_k1       = 0;
v_she_b_k1       = 0;
v_she_c_k1       = 0;

G = dxdy*M_she_k1+c;

int intG = (int)G;
if (G > intG){
    intG = intG + 1;
} else{
    intG = intG;
}

a1      = alpha[intG-1][0];
a2      = alpha[intG-1][1];
a3      = alpha[intG-1][2];
a4      = alpha[intG-1][3];
a5      = alpha[intG-1][4];
a6      = alpha[intG-1][5];
a7      = alpha[intG-1][6];
a8      = alpha[intG-1][7];
a9      = alpha[intG-1][8];

int data[]={C,U,C,U,D,U,D,T,D,T};

int wt_sheint_a_k1 = wt_she_a_k1*10000;
int wt_sheint_b_k1 = wt_she_b_k1*10000;
int wt_sheint_c_k1 = wt_she_c_k1*10000;

TRa= (wt_sheint_a_k1>0)+(wt_sheint_a_k1>INTPI_2)
+(wt_sheint_a_k1>INTPI)+(wt_sheint_a_k1>INTPI23);

wt_sheint_a_k1 = wt_sheint_a_k1-(TRa - 1)*INTPI_2;

if (TRa == 4 || TRa == 2){
    Va = (wt_sheint_a_k1>INTPI_2-a9)+(wt_sheint_a_k1>INTPI_2-a8)
+(wt_sheint_a_k1>INTPI_2-a7)+(wt_sheint_a_k1>INTPI_2-a6)
+(wt_sheint_a_k1>INTPI_2-a5)+(wt_sheint_a_k1>INTPI_2-a4)
+(wt_sheint_a_k1>INTPI_2-a3)+(wt_sheint_a_k1>INTPI_2-a2)
+(wt_sheint_a_k1>INTPI_2-a1);
}
else{
    Va = (wt_sheint_a_k1>a1)+(wt_sheint_a_k1>a2)+(wt_sheint_a_k1>a3)

```

```

        +(wt_sheint_a_k1>a4)+(wt_sheint_a_k1>a5)+(wt_sheint_a_k1>a6)
        +(wt_sheint_a_k1>a7)+(wt_sheint_a_k1>a8)+(wt_sheint_a_k1>a9);
    }

    v_she_a_k1=data[Va];

    if (TRa==3){
        v_she_a_k1 = -1*v_she_a_k1;
    }
    else if (TRa==2){
        v_she_a_k1 = v_she_a_k1^3;
    }
    else if (TRa==4){
        v_she_a_k1 = -1*(v_she_a_k1^3);
    }

    TRb= (wt_sheint_b_k1>0)+(wt_sheint_b_k1>INTPI_2)
    +(wt_sheint_b_k1>INTPI)+(wt_sheint_b_k1>INTPI23);

    wt_sheint_b_k1 = wt_sheint_b_k1-(TRb - 1)*INTPI_2;

    if (TRb == 4 || TRb == 2){
        Vb = (wt_sheint_b_k1>INTPI_2-a9)+(wt_sheint_b_k1>INTPI_2-a8)
        +(wt_sheint_b_k1>INTPI_2-a7)+(wt_sheint_b_k1>INTPI_2-a6)
        +(wt_sheint_b_k1>INTPI_2-a5)+(wt_sheint_b_k1>INTPI_2-a4)
        +(wt_sheint_b_k1>INTPI_2-a3)+(wt_sheint_b_k1>INTPI_2-a2)
        +(wt_sheint_b_k1>INTPI_2-a1);
    }
    else{
        Vb = (wt_sheint_b_k1>a1)+(wt_sheint_b_k1>a2)+(wt_sheint_b_k1>a3)
        +(wt_sheint_b_k1>a4)+(wt_sheint_b_k1>a5)+(wt_sheint_b_k1>a6)
        +(wt_sheint_b_k1>a7)+(wt_sheint_b_k1>a8)+(wt_sheint_b_k1>a9);
    }

    v_she_b_k1=data[Vb];

    if (TRb==3){
        v_she_b_k1 = -1*v_she_b_k1;
    }
    else if (TRb==2){
        v_she_b_k1 = v_she_b_k1^3;
    }
    else if (TRb==4){
        v_she_b_k1 = -1*(v_she_b_k1^3);
    }

    TRc= (wt_sheint_c_k1>0)+(wt_sheint_c_k1>INTPI_2)
    +(wt_sheint_c_k1>INTPI)+(wt_sheint_c_k1>INTPI23);

```

```

wt_sheint_c_k1 = wt_sheint_c_k1-(TRc - 1)*INTPI_2;

if (TRc == 4 || TRc == 2){
    Vc = (wt_sheint_c_k1>INTPI_2-a9)+(wt_sheint_c_k1>INTPI_2-a8)
    +(wt_sheint_c_k1>INTPI_2-a7)+(wt_sheint_c_k1>INTPI_2-a6)
    +(wt_sheint_c_k1>INTPI_2-a5)+(wt_sheint_c_k1>INTPI_2-a4)
    +(wt_sheint_c_k1>INTPI_2-a3)+(wt_sheint_c_k1>INTPI_2-a2)
    +(wt_sheint_c_k1>INTPI_2-a1);
}
else{
    Vc = (wt_sheint_c_k1>a1)+(wt_sheint_c_k1>a2)+(wt_sheint_c_k1>a3)
    +(wt_sheint_c_k1>a4)+(wt_sheint_c_k1>a5)+(wt_sheint_c_k1>a6)
    +(wt_sheint_c_k1>a7)+(wt_sheint_c_k1>a8)+(wt_sheint_c_k1>a9);
}

v_she_c_k1=data[Vc];

if (TRc==3){
    v_she_c_k1 = -1*v_she_c_k1;
}
else if (TRc==2){
    v_she_c_k1 = v_she_c_k1^3;
}
else if (TRc==4){
    v_she_c_k1 = -1*(v_she_c_k1^3);
}

int v_she_Alpha_k1 = 2*v_she_a_k1 - v_she_b_k1 - v_she_c_k1;
int v_she_Beta_k1 = v_she_b_k1 - v_she_c_k1;
int q_opt_k1 = 1;

int k;
for (k = 1; k < 128; k++){
    int X1 = v_she_Alpha_k1 - HEX1[0][k-1];
    int X2 = v_she_Beta_k1 - HEX1[1][k-1];
    if (X1 == 0 && X2 == 0){
        q_opt_k1 = k;
        break;
    }
}

int p_opt_k1 = 1;
int HEX2_ind2 = q_opt_k1-1;
for (k = 1; k < 8; k++){
    int HEX2_ind1 = k-1;
    int HEX3_ind = HEX2[HEX2_ind1][HEX2_ind2]-1;
    int X1 = v_she_a_k1 - HEX3[HEX3_ind][1];

```

```

    int X2 = v_she_b_k1 - HEX3[HEX3_ind][2];
    int X3 = v_she_c_k1 - HEX3[HEX3_ind][3];
    if (X1 == 0 && X2 == 0 && X3 == 0){
        p_opt_k1 = HEX3[HEX3_ind][0];
        break;
    }
}

K1 = (1 - TRL)*i_g_a_k1 - TL*v_g_a_k1;
K2 = (1 - TRL)*i_g_b_k1 - TL*v_g_b_k1;
K3 = (1 - TRL)*i_g_c_k1 - TL*v_g_c_k1;

K4 = V_dc*v_she_a_k1;
K5 = V_dc*v_she_b_k1;
K6 = V_dc*v_she_c_k1;

J_op_k = 1e10;
Sa_op_k1 = 0;
Sb_op_k1 = 0;
Sc_op_k1 = 0;

for (k = 1; k < 8; k++) {

    int HEX4_ind = (p_opt_k1-1)*7+k;

    v_a_n_k1 = V_dc*HEX4[HEX4_ind-1][0]; //ESTA BIEN
    v_b_n_k1 = V_dc*HEX4[HEX4_ind-1][1];
    v_c_n_k1 = V_dc*HEX4[HEX4_ind-1][2];

    v_o_n_k1 = UN_TER*(v_a_n_k1 + v_b_n_k1 + v_c_n_k1);

    K7 = TL*(v_a_n_k1 - v_o_n_k1);
    K8 = TL*(v_b_n_k1 - v_o_n_k1);
    K9 = TL*(v_c_n_k1 - v_o_n_k1);

    i_g_a_k2 = K1 + K7;
    i_g_b_k2 = K2 + K8;
    i_g_c_k2 = K3 + K9;

    J_i_g_a_k = (i_g_a_k2-i_g_a_r_k2)*(i_g_a_k2-i_g_a_r_k2);
    J_i_g_b_k = (i_g_b_k2-i_g_b_r_k2)*(i_g_b_k2-i_g_b_r_k2);
    J_i_g_c_k = (i_g_c_k2-i_g_c_r_k2)*(i_g_c_k2-i_g_c_r_k2);

    J_i_g_abc_k = J_i_g_a_k + J_i_g_b_k + J_i_g_c_k;

    J_v_g_a_k = (v_a_n_k1 - K4)*(v_a_n_k1 - K4);
    J_v_g_b_k = (v_b_n_k1 - K5)*(v_b_n_k1 - K5);
    J_v_g_c_k = (v_c_n_k1 - K6)*(v_c_n_k1 - K6);

```

```

J_v_g_abc_k = sigma_d_k1*(J_v_g_a_k + J_v_g_b_k + J_v_g_c_k);

J_u_g_a_k = (v_a_n_k1 - v_a_n_k)*(v_a_n_k1 - v_a_n_k);
J_u_g_b_k = (v_b_n_k1 - v_b_n_k)*(v_b_n_k1 - v_b_n_k);
J_u_g_c_k = (v_c_n_k1 - v_c_n_k)*(v_c_n_k1 - v_c_n_k);

J_u_g_abc_k = Lmb_u_k1*(J_u_g_a_k + J_u_g_b_k + J_u_g_c_k);

J_k = J_v_g_abc_k + J_u_g_abc_k + J_i_g_abc_k;

if (J_k < J_op_k){
    J_op_k      = J_k;
    Sa_op_k1    = HEX4[HEX4_ind-1][0];
    Sb_op_k1    = HEX4[HEX4_ind-1][1];
    Sc_op_k1    = HEX4[HEX4_ind-1][2];
}
}

Sa_op_k = Sa_op_k1;
Sb_op_k = Sb_op_k1;
Sc_op_k = Sc_op_k1;

switch (Sa_op_k1){
    case 3:
        M1CMPR1 = (int)M0_MAX;
        M1CMPR2 = (int)0;
        M2CMPR1 = (int)M0_MAX;
        M2CMPR2 = (int)0;
        M3CMPR1 = (int)M0_MAX;
        M3CMPR2 = (int)0;
        break;
    case 2:
        M1CMPR1 = (int)M0_MAX;
        M1CMPR2 = (int)0;
        M2CMPR1 = (int)M0_MAX;
        M2CMPR2 = (int)0;
        M3CMPR1 = (int)0;
        M3CMPR2 = (int)0;
        break;
    case 1:
        M1CMPR1 = (int)M0_MAX;
        M1CMPR2 = (int)0;
        M2CMPR1 = (int)0;
        M2CMPR2 = (int)0;
        M3CMPR1 = (int)0;
        M3CMPR2 = (int)0;

```

```

        break;
    case 0:
        M1CMPR1 = (int)0;
        M1CMPR2 = (int)0;
        M2CMPR1 = (int)0;
        M2CMPR2 = (int)0;
        M3CMPR1 = (int)0;
        M3CMPR2 = (int)0;
        break;
    case -1:
        M1CMPR1 = (int)0;
        M1CMPR2 = (int)MO_MAX;
        M2CMPR1 = (int)0;
        M2CMPR2 = (int)0;
        M3CMPR1 = (int)0;
        M3CMPR2 = (int)0;
        break;
    case -2:
        M1CMPR1 = (int)0;
        M1CMPR2 = (int)MO_MAX;
        M2CMPR1 = (int)0;
        M2CMPR2 = (int)MO_MAX;
        M3CMPR1 = (int)0;
        M3CMPR2 = (int)0;
        break;
    case -3:
        M1CMPR1 = (int)0;
        M1CMPR2 = (int)MO_MAX;
        M2CMPR1 = (int)0;
        M2CMPR2 = (int)MO_MAX;
        M3CMPR1 = (int)0;
        M3CMPR2 = (int)MO_MAX;
        break;
    default:
        M1CMPR1 = (int)0;
        M1CMPR2 = (int)0;
        M2CMPR1 = (int)0;
        M2CMPR2 = (int)0;
        M3CMPR1 = (int)0;
        M3CMPR2 = (int)0;
}

switch (Sb_op_k1){
    case 3:
        M0CMPR1 = (int)MO_MAX;
        M0CMPR2 = (int)0;
        M5CMPR1 = (int)MO_MAX;
        M5CMPR2 = (int)0;

```

```
        M4CMPR1 = (int)MO_MAX;
        M4CMPR2 = (int)0;
        break;
    case 2:
        M0CMPR1 = (int)MO_MAX;
        M0CMPR2 = (int)0;
        M5CMPR1 = (int)MO_MAX;
        M5CMPR2 = (int)0;
        M4CMPR1 = (int)0;
        M4CMPR2 = (int)0;
        break;
    case 1:
        M0CMPR1 = (int)MO_MAX;
        M0CMPR2 = (int)0;
        M5CMPR1 = (int)0;
        M5CMPR2 = (int)0;
        M4CMPR1 = (int)0;
        M4CMPR2 = (int)0;
        break;
    case 0:
        M0CMPR1 = (int)0;
        M0CMPR2 = (int)0;
        M5CMPR1 = (int)0;
        M5CMPR2 = (int)0;
        M4CMPR1 = (int)0;
        M4CMPR2 = (int)0;
        break;
    case -1:
        M0CMPR1 = (int)0;
        M0CMPR2 = (int)MO_MAX;
        M5CMPR1 = (int)0;
        M5CMPR2 = (int)0;
        M4CMPR1 = (int)0;
        M4CMPR2 = (int)0;
        break;
    case -2:
        M0CMPR1 = (int)0;
        M0CMPR2 = (int)MO_MAX;
        M5CMPR1 = (int)0;
        M5CMPR2 = (int)MO_MAX;
        M4CMPR1 = (int)0;
        M4CMPR2 = (int)0;
        break;
    case -3:
        M0CMPR1 = (int)0;
        M0CMPR2 = (int)MO_MAX;
        M5CMPR1 = (int)0;
        M5CMPR2 = (int)MO_MAX;
```

```

        M4CMPR1 = (int)0;
        M4CMPR2 = (int)MO_MAX;
        break;
default:
    M0CMPR1 = (int)0;
    M0CMPR2 = (int)0;
    M5CMPR1 = (int)0;
    M5CMPR2 = (int)0;
    M4CMPR1 = (int)0;
    M4CMPR2 = (int)0;
}

switch (Sc_op_k1){
    case 3:
        M0CMPR3 = (int)MO_MAX;
        M1CMPR3 = (int)0;
        M2CMPR3 = (int)MO_MAX;
        M3CMPR3 = (int)0;
        M4CMPR3 = (int)MO_MAX;
        M5CMPR3 = (int)0;
        break;
    case 2:
        M0CMPR3 = (int)MO_MAX;
        M1CMPR3 = (int)0;
        M2CMPR3 = (int)MO_MAX;
        M3CMPR3 = (int)0;
        M4CMPR3 = (int)0;
        M5CMPR3 = (int)0;
        break;
    case 1:
        M0CMPR3 = (int)MO_MAX;
        M1CMPR3 = (int)0;
        M2CMPR3 = (int)0;
        M3CMPR3 = (int)0;
        M4CMPR3 = (int)0;
        M5CMPR3 = (int)0;
        break;
    case 0:
        M0CMPR3 = (int)0;
        M1CMPR3 = (int)0;
        M2CMPR3 = (int)0;
        M3CMPR3 = (int)0;
        M4CMPR3 = (int)0;
        M5CMPR3 = (int)0;
        break;
    case -1:
        M0CMPR3 = (int)0;
        M1CMPR3 = (int)MO_MAX;

```

```

        M2CMPR3 = (int)0;
        M3CMPR3 = (int)0;
        M4CMPR3 = (int)0;
        M5CMPR3 = (int)0;
        break;
    case -2:
        M0CMPR3 = (int)0;
        M1CMPR3 = (int)M0_MAX;
        M2CMPR3 = (int)0;
        M3CMPR3 = (int)M0_MAX;
        M4CMPR3 = (int)0;
        M5CMPR3 = (int)0;
        break;
    case -3:
        M0CMPR3 = (int)0;
        M1CMPR3 = (int)M0_MAX;
        M2CMPR3 = (int)0;
        M3CMPR3 = (int)M0_MAX;
        M4CMPR3 = (int)0;
        M5CMPR3 = (int)M0_MAX;
        break;
    default:
        M0CMPR3 = (int)0;
        M1CMPR3 = (int)0;
        M2CMPR3 = (int)0;
        M3CMPR3 = (int)0;
        M4CMPR3 = (int)0;
        M5CMPR3 = (int)0;
}

Init_Mod(M0_MAX);

graf2[ind_arr] = i_g_a_k;
vgraf[ind_arr] = i_g_a_r_k;

Vdc_mas_int = V_MAXIMO_VDC * 102.4;

ind_arr++;
if (ind_arr >= DATOS){
    ind_arr = 0;
}

VCC_CONV    = (int)Vdc_mas_int;

MODTRIP012  = 0X0000;
START_CONV  = 0xFFFF;
RST_CONV    = 0X0000;

```

```

    ACE_INT_enableDspGlobalInt();
    INTOCLRFLG = INTO_ADC0_MASK;
}

```

A.2. Gráficos

A.2.1. Gráfico Convertidor Multicelda en Cascada de 5 niveles

```

subplot(3,1,1)
x=[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17];
y=[0,0,0,1,1,1,0,0,0,0,0,-1,-1,-1,0,0,0,0];
stairs(x,y,'LineWidth', 1.5,'Color',[0 0 0])
ylim([-2.5,2.5])
xlim([0,17])
grid on
set(gca, 'YTick', [-2,-1,0,1,2])
set(gca, 'YTickLabel', {'$-2V_{cc}$','$-V_{cc}$','$0$','$V_{cc}$',
'$2V_{cc}$'}, 'TickLabelInterpreter', 'latex')
set(gca, 'XTick', [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17])
set(gca, 'XTickLabel', {'','','','','','','','','','','','','','','',''})
title('$v_1$', 'Interpreter', 'latex', 'FontSize', 14)

```

```

subplot(3,1,2)
x=[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17];
y=[0,1,1,1,1,1,1,0,-1,-1,-1,-1,-1,-1,-1,0,0];
stairs(x,y,'LineWidth', 1.5,'Color',[0 0 0])
ylim([-2.5,2.5])
xlim([0,17])
grid on
set(gca, 'YTick', [-2,-1,0,1,2])
set(gca, 'YTickLabel', {'$-2V_{cc}$','$-V_{cc}$','$0$','$V_{cc}$',
'$2V_{cc}$'}, 'TickLabelInterpreter', 'latex')
set(gca, 'XTick', [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17])
set(gca, 'XTickLabel', {'','','','','','','','','','','','','','','',''})
title('$v_2$', 'Interpreter', 'latex', 'FontSize', 14)

```

```

subplot(3,1,3)
x=[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17];
y=[0,1,1,2,2,2,1,1,0,-1,-1,-2,-2,-2,-1,-1,0,0];
stairs(x,y,'LineWidth', 1.5,'Color',[0 0 0])
ylim([-2.5,2.5])
xlim([0,17])
grid on
set(gca, 'YTick', [-2,-1,0,1,2])
set(gca, 'YTickLabel', {'$-2V_{cc}$','$-V_{cc}$','$0$','$V_{cc}$',
'$2V_{cc}$'}, 'TickLabelInterpreter', 'latex')

```

```

set(gca, 'XTick', [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17])
set(gca, 'XTickLabel', {'','','','','','','','','','','','','','',''})
title('$v_0$', 'Interpreter', 'latex', 'FontSize', 14)

```

A.2.2. Gráfico Modulación PS-PWM Convertidor Multicelda en Cascada 5 niveles y tensión de salida.

```

x = linspace(0, 10, 2000);

y11 = mod(-10 * x, -10) + 5;
y11(y11 < 0) = 0;

y12 = mod(10 * x, 10) - 5;
y12(y12 < 0) = 0;

y1 = (y11 + y12) * (2 / 5) - 1;
y2 = -y1;

sin1 = sin(2 * pi * x / 10);
cmas = sin1 - y1 > 0;
cmenos = y2 - sin1 > 0;
cef = cmas - cmenos;

h1 = subplot(2,1,1)
plot(x, sin1, 'LineWidth', 0.8, 'Color', [0 0 0])
hold on;
plot(x, y1, 'LineWidth', 0.8, 'Color', [249/256 56/256 34/256])
plot(x, y2, 'LineWidth', 0.8, 'Color', [1/256 101/256 252/256])

plot(x, cef, 'LineWidth', 1.5, 'Color', [0 0 0]);
legend('v\_control', 'v\_carrier', 'v\_carrier', 'Interpreter', 'latex',
'FontSize', 8)
set(h1, 'XTick', [0])
set(h1, 'XTickLabel', {''})
set(h1, 'YTick', [-1,-0.5,0,0.5,1])
ylabel('Tensi\{o\}n de salida CHB1 [pu]', 'Interpreter', 'latex', 'FontSize', 8)
grid on;
ylim([-1,1])

h2 = subplot(2,1,2)

y21 = mod(-10 * (x-0.25), -10) + 5;
y21(y21 < 0) = 0;

y22 = mod(10 * (x-0.25), 10) - 5;
y22(y22 < 0) = 0;

```

```

y3 = (y21 + y22) * (2 / 5) - 1;
y4 = -y3;

cmas1 = sin1 - y3 > 0;
cmenos1 = y4 - sin1 > 0;
cef1 = cmas1 - cmenos1;

plot(x, sin1,'LineWidth', 0.8,'Color',[0 0 0])
hold on;
plot(x,y3,'LineWidth', 0.8,'Color',[249/256 56/256 34/256])

plot(x, y4,'LineWidth', 0.8,'Color',[1/256 101/256 252/256])

plot(x, cef1,'LineWidth', 1.5,'Color',[0 0 0]);
set(h2, 'XTick', [0])
set(h2, 'XTickLabel', {''})
set(h2, 'YTick', [-1,-0.5,0,0.5,1])
ylabel('Tensi\''{o}n de salida CHB2 [pu'],'Interpreter','latex','FontSize',8)
grid on;
ylim([-1,1])

figure(2)
plot(x,cef+cef1,'LineWidth', 1.5,'Color',[0 0 0])
set(gca, 'XTick', [0])
set(gca, 'XTickLabel', {''})
set(gca, 'YTick', [-2,-1,0,1,2])
grid on;
hold on
plot(x, 2*sin1,'LineWidth', 0.8,'Color',[0 0 0])
ylabel('Tensi\''{o}n de salida [pu'],'Interpreter','latex','FontSize',8)

```

A.2.3. Gráfico Modulación SHE Convertidor Multicelda 3 niveles.

```

t = linspace(0,16,1000);
a1 = 1;
a2 = 2;
a3 = 3;
b1 = 8 - a3;
b2 = 8 - a2;
b3 = 8 - a1;
c1 = 8 + a1;
c2 = 8 + a2;
c3 = 8 + a3;
d1 = 16 - a3;
d2 = 16 - a2;
d3 = 16 - a1;

```

```

mag = 1;
for i = 1:1000
    if t(i) < a1
        y(i) = 0;
    elseif t(i) >= a1 && t(i) < a2
        y(i) = mag;
    elseif t(i) >= a2 && t(i) < a3
        y(i) = 0;
    elseif t(i) >= a3 && t(i) < b1
        y(i) = mag;
    elseif t(i) >= b1 && t(i) < b2
        y(i) = 0;
    elseif t(i) >= b2 && t(i) < b3
        y(i) = mag;
    elseif t(i) >= b3 && t(i) < c1
        y(i) = 0;
    elseif t(i) >= c1 && t(i) < c2
        y(i) = -mag;
    elseif t(i) >= c2 && t(i) < c3
        y(i) = 0;
    elseif t(i) >= c3 && t(i) < d1
        y(i) = -mag;
    elseif t(i) >= d1 && t(i) < d2
        y(i) = 0;
    elseif t(i) >= d2 && t(i) < d3
        y(i) = -mag;
    elseif t(i) >= d3 && t(i) <= 16
        y(i) = 0;
    end
end

plot(t,y,'LineWidth', 1.5,'Color',[0 0 0])
grid on
ylim([-1.2, 1.2])
xlim([0,16])
set(gca, 'XTick', [0, 4, 8, 12])
set(gca, 'XTickLabel', {'', '$\frac{\pi}{2}$', '$\pi$', '$\frac{3\pi}{2}$'},
    'TickLabelInterpreter', 'latex', 'FontSize', 12)
set(gca, 'YTick', [-1, 0, 1])
set(gca, 'YTickLabel', {'-1', '0', '1'}, 'TickLabelInterpreter', 'latex',
    'FontSize', 12)
hold on
lineax1 = 4*ones(1000,1);
lineay1 = linspace(-1.2,1.2,1000);
lineax2 = 8*ones(1000,1);
lineay2 = linspace(-1.2,1.2,1000);
lineax3 = 12*ones(1000,1);
lineay3 = linspace(-1.2,1.2,1000);

```

```

plot(lineax1,lineay1,'--','Color',[1,0,0])
plot(lineax2,lineay2,'--','Color',[1,0,0])
plot(lineax3,lineay3,'--','Color',[1,0,0])

```

A.2.4. Gráfico Algoritmo SHE

```

t = linspace(0, 1/372, 1000);
y = linspace(0, 2*pi, 1000);

subplot(2,1,1)
plot(t,y, 'LineWidth', 1, 'Color', [0 0 0])
grid on
hold on
xlim([0,1/372])
ylim([0,2*pi])
xlabel('Tiempo [s]', 'Interpreter', 'latex')
ylabel('wt\_she\_a\_k1 [rad]', 'Interpreter', 'latex')
linea1x = (1/372)*(1/4)*ones(1,100);
linea2x = (1/372)*(2/4)*ones(1,100);
linea3x = (1/372)*(3/4)*ones(1,100);
lineay = linspace(0,2*pi,100);
plot(linea1x,lineay,'--', 'LineWidth', 1, 'Color',1/255*[100 100 100])
plot(linea2x,lineay,'--', 'LineWidth', 1, 'Color',1/255*[100 100 100])
plot(linea3x,lineay,'--', 'LineWidth', 1, 'Color',1/255*[100 100 100])

set(gca, 'YTick', [0,pi,2*pi])
set(gca, 'YTickLabel', {'$0$', '$\pi$', '$2\pi$'}, 'TickLabelInterpreter',
'latex');

set(gca, 'XTick', [0,6.72043010752690e-05,0.000134408602150538,
0.000201612903225806,0.000268817204301075,
0.000336021505376344,0.000403225806451613,
0.000470430107526883,0.000537634408602152,
0.000604838709677421,0.000739247311827959,
0.000806451612903228,0.000873655913978497,
0.000940860215053766,0.00100806451612903,
0.00107526881720430,0.00114247311827957,
0.00120967741935484,0.00127688172043011,
0.00134408602150538,0.00141129032258064,
0.00147849462365591,0.00154569892473118,
0.00161290322580645,0.00168010752688172,
0.00174731182795699,0.00181451612903226,
0.00188172043010753,0.00194892473118280,
0.00208333333333333,0.00215053763440860,
0.00221774193548387,0.00228494623655914,
0.00235215053763441,0.00241935483870968,
0.00248655913978495,0.00255376344086022,

```

```

0.00262096774193549,0.00268817204301075]);

set(gca, 'XTickLabel', {0,'$a_1$', '$a_2$', '$a_3$', '$a_4$', '$a_5$', '$a_6$',
'$a_7$', '$a_8$', '$a_9$', '$b_1$', '$b_2$', '$b_3$', '$b_4$', '$b_5$', '$b_6$',
'$b_7$', '$b_8$', '$b_9$', '$t_2$', '$c_1$', '$c_2$', '$c_3$', '$c_4$', '$c_5$',
'$c_6$', '$c_7$', '$c_8$', '$c_9$', '$d_1$', '$d_2$', '$d_3$', '$d_4$', '$d_5$',
'$d_6$', '$d_7$', '$d_8$', '$d_9$', ''}, 'TickLabelInterpreter', 'latex');

PI2 = 2*pi;
PI = pi;
T_0 = PI2;
t2 = PI;
t4 = T_0;

a = linspace(0,pi/2,11);
a(11) = [];
b = t2 - a;
c = t2 + a;
d = t4 - a;
hold on

C = 0;
U = 1;
D = 2;
T = 3;

abcdt = sort([a,b]);
subplot(2,1,2)
abcdres = [C,U,C,U,D,U,D,T,D,T,T,D,T,D,U,D,U,C,U,C,C,C,-U,C,-U,-D,-U,-D,
-T,-D,-T,-T,-D,-T,-D,-U,-D,-U,C,-U,C,C];
stairs([abcdt(1:10),abcdt(10:20),pi+abcdt(1:10),pi+abcdt(10:20)],
transpose(abcdres),'LineWidth', 1, 'Color', [0 0 0])
ylim([-3.5,3.5])
grid on
hold on
xlim([0,2*pi])
xlabel('wt\_she\_a\_k1 [rad]', 'Interpreter', 'latex')
ylabel('v\_she\_a\_k1 [-]', 'Interpreter', 'latex')
linea1x = (2*pi)*(1/4)*ones(1,100);
linea2x = (2*pi)*(2/4)*ones(1,100);
linea3x = (2*pi)*(3/4)*ones(1,100);
lineay = linspace(-3.5,3.5,100);
plot(linea1x,lineay,'--', 'LineWidth', 1, 'Color',1/255*[100 100 100])
plot(linea2x,lineay,'--', 'LineWidth', 1, 'Color',1/255*[100 100 100])
plot(linea3x,lineay,'--', 'LineWidth', 1, 'Color',1/255*[100 100 100])

set(gca, 'YTick', [-3,-2,-1,0,1,2,3], 'TickLabelInterpreter', 'latex')

```

```
set(gca, 'XTick', [0,0.157079632679490,0.314159265358979,0.471238898038469,
0.628318530717959,0.785398163397448,0.942477796076938,
1.09955742875643,1.25663706143592,1.41371669411541,
1.72787595947439,1.88495559215388,2.04203522483337,
2.19911485751286,2.35619449019235,2.51327412287183,
2.67035375555132,2.82743338823081,2.98451302091030,
3.14159265358979,3.29867228626928,3.45575191894877,
3.61283155162826,3.76991118430775,3.92699081698724,
4.08407044966673,4.24115008234622,4.39822971502571,
4.55530934770520,4.86946861306418,5.02654824574367,
5.18362787842316,5.34070751110265,5.49778714378214,
5.65486677646163,5.81194640914112,5.96902604182061,
6.12610567450010,6.28318530717959]);
```

```
set(gca, 'XTickLabel', {0,'$a_1$', '$a_2$', '$a_3$', '$a_4$', '$a_5$', '$a_6$',
'$a_7$', '$a_8$', '$a_9$', '$b_1$', '$b_2$', '$b_3$', '$b_4$', '$b_5$', '$b_6$',
'$b_7$', '$b_8$', '$b_9$', '$t_2$', '$c_1$', '$c_2$', '$c_3$', '$c_4$', '$c_5$',
'$c_6$', '$c_7$', '$c_8$', '$c_9$', '$d_1$', '$d_2$', '$d_3$', '$d_4$', '$d_5$',
'$d_6$', '$d_7$', '$d_8$', '$d_9$'}, 'TickLabelInterpreter', 'latex');
```

A.2.5. Gráfico Algoritmo SHE optimizado

```
PI2 = 2*pi;
PI = pi;
T_0 = PI2;
t2 = PI;
t4 = T_0;

subplot(2,2,1)

a = [0, 0.5,0.65,0.8,0.94,1.05,1.2,1.3,1.4,1.5, pi/2];
pa = sort(pi/2-a);
b = t2 - a;
c = t2 + a;
d = t4 - a;

C = 0;
U = 1;
D = 2;
T = 3;
abcdres = [C,U,C,U,D,U,D,T,D,T,T];
nabcdres = fliplr(abcdres);
nabcdres(1) = [];
nabcdres(11) = nabcdres(10);

stairs(a,transpose(abcdres),'LineWidth', 1, 'Color', [0 0 0])
```

```

set(gca, 'YTick', [-3,-2,-1,0,1,2,3], 'TickLabelInterpreter', 'latex')
set(gca, 'XTick', a)
set(gca, 'XTickLabel', {0, '$a_1$', '$a_2$', '$a_3$', '$a_4$', '$a_5$',
'$a_6$', '$a_7$', '$a_8$', '$a_9$', ''}, 'TickLabelInterpreter', 'latex')
grid on
ylim([-3.5,3.5])
xlim([0,pi/2])
title('1er cuadrante', 'Interpreter', 'latex')
ylabel('v\_{\mathrm{wt}}\_{\mathrm{a}}\_{\mathrm{k1}} [-]', 'Interpreter', 'latex')
xlabel('$10^4 \cdot \mathrm{wt}\_{\mathrm{a}}\_{\mathrm{k1}} \backslash, \mathrm{[rad]}$',
'Interpreter', 'latex')

subplot(2,2,2)

stairs(pa,transpose(nabcdres),'LineWidth', 1, 'Color', [0 0 0])
hold on
stairs(pa,transpose(abcdres),'--','LineWidth', 1,'Color',1/255*[100 100
100])
set(gca, 'YTick', [-3,-2,-1,0,1,2,3], 'TickLabelInterpreter', 'latex')
set(gca, 'XTick', pa)
set(gca, 'XTickLabel', {0, '$a_1$', '$a_2$', '$a_3$', '$a_4$', '$a_5$',
'$a_6$', '$a_7$', '$a_8$', '$a_9$', ''}, 'TickLabelInterpreter', 'latex')
grid on
ylim([-3.5,3.5])
xlim([0,pi/2])
title('2do cuadrante', 'Interpreter', 'latex')
ylabel('v\_{\mathrm{wt}}\_{\mathrm{a}}\_{\mathrm{k1}} [-]', 'Interpreter', 'latex')
xlabel('$10^4 \cdot \mathrm{wt}\_{\mathrm{a}}\_{\mathrm{k1}} \backslash, \mathrm{[rad]}$',
'Interpreter', 'latex')
legend('Original XOR 3','Original', 'Interpreter', 'latex')

subplot(2,2,3)

stairs(a,-1.*transpose(abcdres),'LineWidth', 1, 'Color', [0 0 0])
hold on
stairs(a,transpose(abcdres),'--','LineWidth', 1,'Color',1/255*[100 100
100])
set(gca, 'YTick', [-3,-2,-1,0,1,2,3], 'TickLabelInterpreter', 'latex')
set(gca, 'XTick', a)
set(gca, 'XTickLabel', {0, '$a_1$', '$a_2$', '$a_3$', '$a_4$', '$a_5$',
'$a_6$', '$a_7$', '$a_8$', '$a_9$', ''}, 'TickLabelInterpreter', 'latex')
grid on
ylim([-3.5,3.5])
xlim([0,pi/2])
title('3er cuadrante', 'Interpreter', 'latex')
ylabel('v\_{\mathrm{wt}}\_{\mathrm{a}}\_{\mathrm{k1}} [-]', 'Interpreter', 'latex')
xlabel('$10^4 \cdot \mathrm{wt}\_{\mathrm{a}}\_{\mathrm{k1}} \backslash, \mathrm{[rad]}$',

```

```

    'Interpreter', 'latex')
legend('-Original', 'Original',
    'Interpreter', 'latex')

subplot(2,2,4)

stairs(pa,-1.*transpose(nabcdres),'LineWidth', 1, 'Color', [0 0 0])
hold on
stairs(pa,transpose(abcdres),'--','LineWidth', 1,'Color',1/255*[100    100
    100])
set(gca, 'YTick', [-3,-2,-1,0,1,2,3], 'TickLabelInterpreter', 'latex')
set(gca, 'XTick', pa)
set(gca, 'XTickLabel', {0, '$a_1$', '$a_2$', '$a_3$', '$a_4$', '$a_5$',
    '$a_6$', '$a_7$', '$a_8$', '$a_9$', ''}, 'TickLabelInterpreter', 'latex')
grid on
ylim([-3.5,3.5])
xlim([0,pi/2])
ylabel('v\_{\rm{she}}\_{a\_k1} [-]', 'Interpreter', 'latex')
xlabel('$10^4 \cdot \mathrm{wt\_{\rm{she}}\_{a\_k1}} \backslash, \mathrm{[rad]}$',
    'Interpreter', 'latex')
title('4to cuadrante', 'Interpreter', 'latex')
legend('-(Original XOR 3)', 'Original', 'Interpreter', 'latex')

```

A.2.6. Gráfico Ensayo escalón magnitud de corriente implementado

```

Ts_k_x = load('imp_mag_corr.mat','Tsk');
i_g_a_k = load('imp_mag_corr.mat','igak');
i_g_a_r_k = load('imp_mag_corr.mat','igark');
v_g_a_k = load('imp_mag_corr.mat','vgak');
v_a_n_k = load('imp_mag_corr.mat','vank');
i_g_d_k = load('imp_mag_corr.mat','igdk');
i_g_r_d_k = load('imp_mag_corr.mat','igrdk');
i_g_q_k = load('imp_mag_corr.mat','igqk');
i_g_r_q_k = load('imp_mag_corr.mat','igrqk');
Ts_k_y = load('imp_mag_corr.mat','Tsk');
sigma = load('imp_mag_corr.mat','sigma');

dataTs_k_x = Ts_k_x.(char(fieldnames(Ts_k_x)));
datai_g_a_k = i_g_a_k.(char(fieldnames(i_g_a_k)));
datai_g_a_r_k = i_g_a_r_k.(char(fieldnames(i_g_a_r_k)));
datav_g_a_k = v_g_a_k.(char(fieldnames(v_g_a_k)));
datav_a_n_k = v_a_n_k.(char(fieldnames(v_a_n_k)));
datai_g_d_k = i_g_d_k.(char(fieldnames(i_g_d_k)));
datai_g_r_d_k = i_g_r_d_k.(char(fieldnames(i_g_r_d_k)));
datai_g_q_k = i_g_q_k.(char(fieldnames(i_g_q_k)));
datai_g_r_q_k = i_g_r_q_k.(char(fieldnames(i_g_r_q_k)));
dataTs_k_y = Ts_k_y.(char(fieldnames(Ts_k_y)));

```

```

datasigma = sigma.(char(fieldnames(sigma)));

datos = 8000;
Ts = cumsum(dataTs_k_x);

figure7 = figure(7);

set(figure7, 'Position', [0, 0, 600, 1000]);

subplot(6,1,1)
plot(Ts(1:datos,2), datav_g_a_k(:,2));
xlim([0.2622,0.3322]); %Ts(datos,2)];
ylim([-180,180]);
title('Tensi\'\'on red fase \'\'a" vs Tensi\'\'on CHB 7L fase \'\'a" ',
'Interpreter','latex');
xlabel('Tiempo [s]','Interpreter','latex');
ylabel('Tensi\'\'on [A]','Interpreter','latex');
grid on
hold on
plot(Ts(1:datos,2), datav_a_n_k(:,2));
legend('v\_g\_a\_k','v\_a\_n\_k','FontSize', 6,'Interpreter','latex')

subplot(6,1,2)
plot(Ts(1:datos,2), datai_g_a_k(:,2));
xlim([0.2622,0.3322]);
ylim([-25,25]);
title('Corriente de l\'\'inea vs Referencia','Interpreter','latex');
xlabel('Tiempo [s]','Interpreter','latex');
ylabel('Corriente [A]','Interpreter','latex');
grid on
hold on
plot(Ts(1:datos,2), datai_g_a_r_k(:,2));
legend('i\_g\_a\_k','i\_g\_a\_r\_k','FontSize', 6,'Interpreter','latex')

subplot(6,1,3)
plot(Ts(1:datos,2), datai_g_d_k(:,2));
xlim([0.2622,0.3322]);
ylim([-5,25]);
title('Corriente eje \'\'d" vs Referencia','Interpreter','latex');
xlabel('Tiempo [s]','Interpreter','latex');
ylabel('Corriente [A]','Interpreter','latex');
grid on
hold on
plot(Ts(1:datos,2), datai_g_r_d_k(:,2));
legend('i\_g\_d\_k','i\_g\_r\_d\_k','FontSize', 6,'Interpreter','latex')

subplot(6,1,4)
plot(Ts(1:datos,2), datai_g_q_k(:,2));

```

```

xlim([0.2622,0.3322]);
ylim([-7,8]);
title('Corriente eje "q" vs Referencia','Interpreter','latex');
xlabel('Tiempo [s]','Interpreter','latex');
ylabel('Corriente [A]','Interpreter','latex');
grid on
hold on
plot(Ts(1:datos,2), datai_g_r_q_k(:,2));
legend('i\g\_q\_k','i\g\_r\_q\_k','FontSize', 6,'Interpreter','latex')

subplot(6,1,5)
plot(Ts(1:datos,2), dataTs_k_x(:,2));
xlim([0.2622,0.3322]);
ylim([3e-5,7e-5]);
title('Tiempo de interrupci\''on DSP vs Referencia','Interpreter','latex');
xlabel('Tiempo [s]','Interpreter','latex');
ylabel('Tiempo [s]','Interpreter','latex');
grid on
hold on
plot(Ts(1:datos,2), 1/(372*50)*ones(1,8000));
legend('Ts\_k','Ts\_k\_r','FontSize', 6,'Interpreter','latex')

subplot(6,1,6)
plot(Ts(1:datos,2), datasigma(:,2));
xlim([0.2622,0.3322]);
ylim([0,1.2e-3]);
title('$\sigma$','Interpreter','latex');
xlabel('Tiempo [s]','Interpreter','latex');
ylabel('Magnitud [-]','Interpreter','latex');
grid on
hold on

```

A.2.7. Gráfico Ensayo escalón magnitud de corriente simulado

```

datos = load('sim_mag_corr.mat', 'sim_mag_corr');
sim_mag_corr = datos.sim_mag_corr;
limdatos = 8000;
dataTs_k_x = sim_mag_corr.time(1:limdatos);
Ts = dataTs_k_x;
datai_g_a_k = sim_mag_corr.signals.values(1:limdatos,2);
datai_g_a_r_k = sim_mag_corr.signals.values(1:limdatos,3);
datav_g_a_k = sim_mag_corr.signals.values(1:limdatos,1);
datav_a_n_k = sim_mag_corr.signals.values(1:limdatos,10);
datai_g_d_k = sim_mag_corr.signals.values(1:limdatos,4);
datai_g_r_d_k = sim_mag_corr.signals.values(1:limdatos,5);
datai_g_q_k = sim_mag_corr.signals.values(1:limdatos,6);
datai_g_r_q_k = sim_mag_corr.signals.values(1:limdatos,7);

```

```

dataTs_k_y = sim_mag_corr.signals.values(1:limdatos,9);
datasigma = sim_mag_corr.signals.values(1:limdatos,8);

datos = 8000;

figure7 = figure(7);

set(figure7, 'Position', [0, 0, 600, 1000]);

subplot(6,1,1)
plot(Ts, datav_g_a_k);
xlim([0.2622,0.3322]);
ylim([-180,180]);
title('Tensi\'\'on red fase \'\'a" vs Tensi\'\'on CHB 7L fase \'\'a"',
'Interpreter','latex');
xlabel('Tiempo [s]','Interpreter','latex');
ylabel('Tensi\'\'on [A]','Interpreter','latex');
grid on
hold on
plot(Ts, datav_a_n_k);
legend('v\_g\_a\_k','v\_a\_n\_k','FontSize', 6,'Interpreter','latex')

subplot(6,1,2)
plot(Ts, datai_g_a_k);
xlim([0.2622,0.3322]);
ylim([-25,25]);
title('Corriente de l\'\'inea vs Referencia','Interpreter','latex');
xlabel('Tiempo [s]','Interpreter','latex');
ylabel('Corriente [A]','Interpreter','latex');
grid on
hold on
plot(Ts, datai_g_a_r_k);
legend('i\_g\_a\_k','i\_g\_a\_r\_k','FontSize', 6,'Interpreter','latex')

subplot(6,1,3)
plot(Ts, datai_g_d_k);
xlim([0.2622,0.3322]);
ylim([-5,25]);
title('Corriente eje \'\'d" vs Referencia','Interpreter','latex');
xlabel('Tiempo [s]','Interpreter','latex');
ylabel('Corriente [A]','Interpreter','latex');
grid on
hold on
plot(Ts, datai_g_r_d_k);
legend('i\_g\_d\_k','i\_g\_r\_d\_k','FontSize', 6,'Interpreter','latex')

subplot(6,1,4)
plot(Ts, datai_g_q_k);

```

```

xlim([0.2622,0.3322]);
ylim([-7,8]);
title('Corriente eje "q" vs Referencia','Interpreter','latex');
xlabel('Tiempo [s]','Interpreter','latex');
ylabel('Corriente [A]','Interpreter','latex');
grid on
hold on
plot(Ts, datai_g_r_q_k);
legend('i\g\_q\_k','i\g\_r\_q\_k','FontSize', 6,'Interpreter','latex')

subplot(6,1,5)
plot(Ts, dataTs_k_y);
xlim([0.2622,0.3322]);
ylim([3e-5,7e-5]);
title('Tiempo de interrupci\''on DSP vs Referencia','Interpreter','latex');
xlabel('Tiempo [s]','Interpreter','latex');
ylabel('Tiempo [s]','Interpreter','latex');
grid on
hold on
plot(Ts, 1/(372*50)*ones(1,limdatos));
legend('Ts\_k','Ts\_k\_r','FontSize', 6,'Interpreter','latex')

subplot(6,1,6)
plot(Ts, datasigma);
xlim([0.2622,0.3322]);
ylim([0,1.2e-3]);
title('$\sigma$','Interpreter','latex');
xlabel('Tiempo [s]','Interpreter','latex');
ylabel('Magnitud [-]','Interpreter','latex');
grid on
hold on

```

A.2.8. Gráfico Ensayo escalón fase de corriente implementado

```

Ts_k_x = load('imp_ang_corr.mat','Tsk');
i_g_a_k = load('imp_ang_corr.mat','igak');
i_g_a_r_k = load('imp_ang_corr.mat','igark');
v_g_a_k = load('imp_ang_corr.mat','vgak');
v_a_n_k = load('imp_ang_corr.mat','vank');
i_g_d_k = load('imp_ang_corr.mat','igdk');
i_g_r_d_k = load('imp_ang_corr.mat','igrdk');
i_g_q_k = load('imp_ang_corr.mat','igqk');
i_g_r_q_k = load('imp_ang_corr.mat','igrqk');
Ts_k_y = load('imp_ang_corr.mat','Tsk');
sigma = load('imp_ang_corr.mat','sigma');

dataTs_k_x = Ts_k_x.(char(fieldnames(Ts_k_x)));
datai_g_a_k = i_g_a_k.(char(fieldnames(i_g_a_k)));

```

```

datai_g_a_r_k = i_g_a_r_k.(char(fieldnames(i_g_a_r_k)));
datav_g_a_k = v_g_a_k.(char(fieldnames(v_g_a_k)));
datav_a_n_k = v_a_n_k.(char(fieldnames(v_a_n_k)));
datai_g_d_k = i_g_d_k.(char(fieldnames(i_g_d_k)));
datai_g_r_d_k = i_g_r_d_k.(char(fieldnames(i_g_r_d_k)));
datai_g_q_k = i_g_q_k.(char(fieldnames(i_g_q_k)));
datai_g_r_q_k = i_g_r_q_k.(char(fieldnames(i_g_r_q_k)));
dataTs_k_y = Ts_k_y.(char(fieldnames(Ts_k_y)));
datasigma = sigma.(char(fieldnames(sigma)));

datos = 8000;
Ts = cumsum(dataTs_k_x);

figure7 = figure(7);

set(figure7, 'Position', [0, 0, 600, 1000]);
subplot(6,1,1)
plot(Ts(1:datos,2), datav_g_a_k(:,2));

xlim([0.2622,0.3322]);
ylim([-180,180]);
title('Tensi\'\'on red fase \'a" vs Tensi\'\'on CHB 7L fase \'a"',
'Interpreter','latex');
xlabel('Tiempo [s]','Interpreter','latex');
ylabel('Tensi\'\'on [A]','Interpreter','latex');
grid on
hold on
plot(Ts(1:datos,2), datav_a_n_k(:,2));
legend('v\_g\_a\_k','v\_a\_n\_k','FontSize', 6,'Interpreter','latex')

subplot(6,1,2)
plot(Ts(1:datos,2), datai_g_a_k(:,2));
xlim([0.2622,0.3322]);
ylim([-25,25]);
title('Corriente de l\'\'inea vs Referencia','Interpreter','latex');
xlabel('Tiempo [s]','Interpreter','latex');
ylabel('Corriente [A]','Interpreter','latex');
grid on
hold on
plot(Ts(1:datos,2), datai_g_a_r_k(:,2));
legend('i\_g\_a\_k','i\_g\_a\_r\_k','FontSize', 6,'Interpreter','latex')

subplot(6,1,3)
plot(Ts(1:datos,2), datai_g_d_k(:,2));
xlim([0.2622,0.3322]);
ylim([-5,25]);
title('Corriente eje \'d" vs Referencia','Interpreter','latex');
xlabel('Tiempo [s]','Interpreter','latex');

```

```

ylabel('Corriente [A]', 'Interpreter', 'latex');
grid on
hold on
plot(Ts(1:datos,2), datai_g_r_d_k(:,2));
legend('i\_g\_d\_k', 'i\_g\_r\_d\_k', 'FontSize', 6, 'Interpreter', 'latex')

subplot(6,1,4)
plot(Ts(1:datos,2), datai_g_q_k(:,2));
xlim([0.2622,0.3322]);
ylim([-7,30]);
title('Corriente eje "q" vs Referencia', 'Interpreter', 'latex');
xlabel('Tiempo [s]', 'Interpreter', 'latex');
ylabel('Corriente [A]', 'Interpreter', 'latex');
grid on
hold on
plot(Ts(1:datos,2), datai_g_r_q_k(:,2));
legend('i\_g\_q\_k', 'i\_g\_r\_q\_k', 'FontSize', 6, 'Interpreter', 'latex')

subplot(6,1,5)
plot(Ts(1:datos,2), dataTs_k_x(:,2));
xlim([0.2622,0.3322]);
ylim([3e-5,7e-5]);
title('Tiempo de interrupci\''on DSP vs Referencia', 'Interpreter', 'latex');
xlabel('Tiempo [s]', 'Interpreter', 'latex');
ylabel('Tiempo [s]', 'Interpreter', 'latex');
grid on
hold on
plot(Ts(1:datos,2), 1/(372*50)*ones(1,8000));
legend('Ts\_k', 'Ts\_k\_r', 'FontSize', 6, 'Interpreter', 'latex')

subplot(6,1,6)
plot(Ts(1:datos,2), datasigma(:,2));
xlim([0.2622,0.3322]);
ylim([0,1.2e-3]);
title('$\sigma$', 'Interpreter', 'latex');
xlabel('Tiempo [s]', 'Interpreter', 'latex');
ylabel('Magnitud [-]', 'Interpreter', 'latex');
grid on
hold on

```

A.2.9. Gráfico Ensayo escalón fase de corriente simulado

```

datos = load('sim_ang_corr.mat', 'sim_ang_corr');
sim_ang_corr = datos.sim_ang_corr;
limdatos = 8000;
dataTs_k_x = sim_ang_corr.time(1:limdatos);
Ts = dataTs_k_x;

```

```

datai_g_a_k = sim_ang_corr.signals.values(1:limdatos,2);
datai_g_a_r_k = sim_ang_corr.signals.values(1:limdatos,3);
datav_g_a_k = sim_ang_corr.signals.values(1:limdatos,1);
datav_a_n_k = sim_ang_corr.signals.values(1:limdatos,10);
datai_g_d_k = sim_ang_corr.signals.values(1:limdatos,4);
datai_g_r_d_k = sim_ang_corr.signals.values(1:limdatos,5);
datai_g_q_k = sim_ang_corr.signals.values(1:limdatos,6);
datai_g_r_q_k = sim_ang_corr.signals.values(1:limdatos,7);
dataTs_k_y = sim_ang_corr.signals.values(1:limdatos,9);
datasigma = sim_ang_corr.signals.values(1:limdatos,8);

datos = 8000;

figure7 = figure(7);

set(figure7, 'Position', [0, 0, 600, 1000]);
subplot(6,1,1)
plot(Ts, datav_g_a_k);
xlim([0.2622,0.3322]);
ylim([-180,180]);
title('Tensi\'\'on red fase \'\'a" vs Tensi\'\'on CHB 7L fase \'\'a"',
'Interpreter','latex');
xlabel('Tiempo [s]','Interpreter','latex');
ylabel('Tensi\'\'on [A]','Interpreter','latex');
grid on
hold on
plot(Ts, datav_a_n_k);
legend('v\_g\_a\_k','v\_a\_n\_k','FontSize', 6,'Interpreter','latex')

subplot(6,1,2)
plot(Ts, datai_g_a_k);
xlim([0.2622,0.3322]);
ylim([-25,25]);
title('Corriente de l\'\'inea vs Referencia','Interpreter','latex');
xlabel('Tiempo [s]','Interpreter','latex');
ylabel('Corriente [A]','Interpreter','latex');
grid on
hold on
plot(Ts, datai_g_a_r_k);
legend('i\_g\_a\_k','i\_g\_a\_r\_k','FontSize', 6,'Interpreter','latex')

subplot(6,1,3)
plot(Ts, datai_g_d_k);
xlim([0.2622,0.3322]);
ylim([-5,25]);
title('Corriente eje \'\'d" vs Referencia','Interpreter','latex');
xlabel('Tiempo [s]','Interpreter','latex');
ylabel('Corriente [A]','Interpreter','latex');

```

```

grid on
hold on
plot(Ts, datai_g_r_d_k);
legend('i\_g\_d\_k', 'i\_g\_r\_d\_k', 'FontSize', 6, 'Interpreter', 'latex')

subplot(6,1,4)
plot(Ts, datai_g_q_k);
xlim([0.2622,0.3322]);
ylim([-7,30]);
title('Corriente eje "q" vs Referencia', 'Interpreter', 'latex');
xlabel('Tiempo [s]', 'Interpreter', 'latex');
ylabel('Corriente [A]', 'Interpreter', 'latex');
grid on
hold on
plot(Ts, datai_g_r_q_k);
legend('i\_g\_q\_k', 'i\_g\_r\_q\_k', 'FontSize', 6, 'Interpreter', 'latex')

subplot(6,1,5)
plot(Ts, dataTs_k_y);
xlim([0.2622,0.3322]);
ylim([3e-5,7e-5]);
title('Tiempo de interrupci\''on DSP vs Referencia', 'Interpreter', 'latex');
xlabel('Tiempo [s]', 'Interpreter', 'latex');
ylabel('Tiempo [s]', 'Interpreter', 'latex');
grid on
hold on
plot(Ts, 1/(372*50)*ones(1,limdatos));
legend('Ts\_k', 'Ts\_k\_r', 'FontSize', 6, 'Interpreter', 'latex')

subplot(6,1,6)
plot(Ts, datasigma);
xlim([0.2622,0.3322]);
ylim([0,1.2e-3]);
title('$\sigma$', 'Interpreter', 'latex');
xlabel('Tiempo [s]', 'Interpreter', 'latex');
ylabel('Magnitud [-]', 'Interpreter', 'latex');
grid on
hold on

```

A.2.10. Gráfico THD y espectro de frecuencia de corriente implementado

```

Ts_k_x = load('imp_ang_corr.mat', 'Tsk');
i_g_a_k_041 = load('imp_Mshe.mat', 'igak041');
i_g_a_k_043 = load('imp_Mshe.mat', 'igak043');
i_g_a_k_045 = load('imp_Mshe.mat', 'igak045');
i_g_a_k_047 = load('imp_Mshe.mat', 'igak047');
i_g_a_k_049 = load('imp_Mshe.mat', 'igak049');
i_g_a_k_051 = load('imp_Mshe.mat', 'igak051');

```

```
i_g_a_k_053 = load('imp_Mshe.mat','igak053');
i_g_a_k_055 = load('imp_Mshe.mat','igak055');
i_g_a_k_057 = load('imp_Mshe.mat','igak057');
i_g_a_k_059 = load('imp_Mshe.mat','igak059');
i_g_a_k_061 = load('imp_Mshe.mat','igak061');
i_g_a_k_063 = load('imp_Mshe.mat','igak063');
i_g_a_k_065 = load('imp_Mshe.mat','igak065');
i_g_a_k_067 = load('imp_Mshe.mat','igak067');
i_g_a_k_069 = load('imp_Mshe.mat','igak069');
i_g_a_k_071 = load('imp_Mshe.mat','igak071');
i_g_a_k_073 = load('imp_Mshe.mat','igak073');
i_g_a_k_075 = load('imp_Mshe.mat','igak075');
i_g_a_k_077 = load('imp_Mshe.mat','igak077');
i_g_a_k_079 = load('imp_Mshe.mat','igak079');
i_g_a_k_081 = load('imp_Mshe.mat','igak081');
i_g_a_k_083 = load('imp_Mshe.mat','igak083');
i_g_a_k_085 = load('imp_Mshe.mat','igak085');
i_g_a_k_087 = load('imp_Mshe.mat','igak087');
i_g_a_k_089 = load('imp_Mshe.mat','igak089');
i_g_a_k_091 = load('imp_Mshe.mat','igak091');
i_g_a_k_093 = load('imp_Mshe.mat','igak093');
i_g_a_k_095 = load('imp_Mshe.mat','igak095');
i_g_a_k_097 = load('imp_Mshe.mat','igak097');
i_g_a_k_099 = load('imp_Mshe.mat','igak099');
i_g_a_k_101 = load('imp_Mshe.mat','igak101');
i_g_a_k_103 = load('imp_Mshe.mat','igak103');
i_g_a_k_105 = load('imp_Mshe.mat','igak105');
i_g_a_k_107 = load('imp_Mshe.mat','igak107');
i_g_a_k_109 = load('imp_Mshe.mat','igak109');
i_g_a_k_111 = load('imp_Mshe.mat','igak111');
i_g_a_k_113 = load('imp_Mshe.mat','igak113');
i_g_a_k_115 = load('imp_Mshe.mat','igak115');

i_g_a_r_k_041 = load('imp_Mshe.mat','igark041');
i_g_a_r_k_043 = load('imp_Mshe.mat','igark043');
i_g_a_r_k_045 = load('imp_Mshe.mat','igark045');
i_g_a_r_k_047 = load('imp_Mshe.mat','igark047');
i_g_a_r_k_049 = load('imp_Mshe.mat','igark049');
i_g_a_r_k_051 = load('imp_Mshe.mat','igark051');
i_g_a_r_k_053 = load('imp_Mshe.mat','igark053');
i_g_a_r_k_055 = load('imp_Mshe.mat','igark055');
i_g_a_r_k_057 = load('imp_Mshe.mat','igark057');
i_g_a_r_k_059 = load('imp_Mshe.mat','igark059');
i_g_a_r_k_061 = load('imp_Mshe.mat','igark061');
i_g_a_r_k_063 = load('imp_Mshe.mat','igark063');
i_g_a_r_k_065 = load('imp_Mshe.mat','igark065');
i_g_a_r_k_067 = load('imp_Mshe.mat','igark067');
i_g_a_r_k_069 = load('imp_Mshe.mat','igark069');
```

```

i_g_a_r_k_071 = load('imp_Mshe.mat','igark071');
i_g_a_r_k_073 = load('imp_Mshe.mat','igark073');
i_g_a_r_k_075 = load('imp_Mshe.mat','igark075');
i_g_a_r_k_077 = load('imp_Mshe.mat','igark077');
i_g_a_r_k_079 = load('imp_Mshe.mat','igark079');
i_g_a_r_k_081 = load('imp_Mshe.mat','igark081');
i_g_a_r_k_083 = load('imp_Mshe.mat','igark083');
i_g_a_r_k_085 = load('imp_Mshe.mat','igark085');
i_g_a_r_k_087 = load('imp_Mshe.mat','igark087');
i_g_a_r_k_089 = load('imp_Mshe.mat','igark089');
i_g_a_r_k_091 = load('imp_Mshe.mat','igark091');
i_g_a_r_k_093 = load('imp_Mshe.mat','igark093');
i_g_a_r_k_095 = load('imp_Mshe.mat','igark095');
i_g_a_r_k_097 = load('imp_Mshe.mat','igark097');
i_g_a_r_k_099 = load('imp_Mshe.mat','igark099');
i_g_a_r_k_101 = load('imp_Mshe.mat','igark101');
i_g_a_r_k_103 = load('imp_Mshe.mat','igark103');
i_g_a_r_k_105 = load('imp_Mshe.mat','igark105');
i_g_a_r_k_107 = load('imp_Mshe.mat','igark107');
i_g_a_r_k_109 = load('imp_Mshe.mat','igark109');
i_g_a_r_k_111 = load('imp_Mshe.mat','igark111');
i_g_a_r_k_113 = load('imp_Mshe.mat','igark113');
i_g_a_r_k_115 = load('imp_Mshe.mat','igark115');

dataTs_k_x = Ts_k_x.(char(fieldnames(Ts_k_x)));
datai_g_a_k_041 = i_g_a_k_041.(char(fieldnames(i_g_a_k_041)));
datai_g_a_k_043 = i_g_a_k_043.(char(fieldnames(i_g_a_k_043)));
datai_g_a_k_045 = i_g_a_k_045.(char(fieldnames(i_g_a_k_045)));
datai_g_a_k_047 = i_g_a_k_047.(char(fieldnames(i_g_a_k_047)));
datai_g_a_k_049 = i_g_a_k_049.(char(fieldnames(i_g_a_k_049)));
datai_g_a_k_051 = i_g_a_k_051.(char(fieldnames(i_g_a_k_051)));
datai_g_a_k_053 = i_g_a_k_053.(char(fieldnames(i_g_a_k_053)));
datai_g_a_k_055 = i_g_a_k_055.(char(fieldnames(i_g_a_k_055)));
datai_g_a_k_057 = i_g_a_k_057.(char(fieldnames(i_g_a_k_057)));
datai_g_a_k_059 = i_g_a_k_059.(char(fieldnames(i_g_a_k_059)));
datai_g_a_k_061 = i_g_a_k_061.(char(fieldnames(i_g_a_k_061)));
datai_g_a_k_063 = i_g_a_k_063.(char(fieldnames(i_g_a_k_063)));
datai_g_a_k_065 = i_g_a_k_065.(char(fieldnames(i_g_a_k_065)));
datai_g_a_k_067 = i_g_a_k_067.(char(fieldnames(i_g_a_k_067)));
datai_g_a_k_069 = i_g_a_k_069.(char(fieldnames(i_g_a_k_069)));
datai_g_a_k_071 = i_g_a_k_071.(char(fieldnames(i_g_a_k_071)));
datai_g_a_k_073 = i_g_a_k_073.(char(fieldnames(i_g_a_k_073)));
datai_g_a_k_075 = i_g_a_k_075.(char(fieldnames(i_g_a_k_075)));
datai_g_a_k_077 = i_g_a_k_077.(char(fieldnames(i_g_a_k_077)));
datai_g_a_k_079 = i_g_a_k_079.(char(fieldnames(i_g_a_k_079)));
datai_g_a_k_081 = i_g_a_k_081.(char(fieldnames(i_g_a_k_081)));
datai_g_a_k_083 = i_g_a_k_083.(char(fieldnames(i_g_a_k_083)));
datai_g_a_k_085 = i_g_a_k_085.(char(fieldnames(i_g_a_k_085)));

```

```

datai_g_a_k_087 = i_g_a_k_087.(char(fieldnames(i_g_a_k_087)));
datai_g_a_k_089 = i_g_a_k_089.(char(fieldnames(i_g_a_k_089)));
datai_g_a_k_091 = i_g_a_k_091.(char(fieldnames(i_g_a_k_091)));
datai_g_a_k_093 = i_g_a_k_093.(char(fieldnames(i_g_a_k_093)));
datai_g_a_k_095 = i_g_a_k_095.(char(fieldnames(i_g_a_k_095)));
datai_g_a_k_097 = i_g_a_k_097.(char(fieldnames(i_g_a_k_097)));
datai_g_a_k_099 = i_g_a_k_099.(char(fieldnames(i_g_a_k_099)));
datai_g_a_k_101 = i_g_a_k_101.(char(fieldnames(i_g_a_k_101)));
datai_g_a_k_103 = i_g_a_k_103.(char(fieldnames(i_g_a_k_103)));
datai_g_a_k_105 = i_g_a_k_105.(char(fieldnames(i_g_a_k_105)));
datai_g_a_k_107 = i_g_a_k_107.(char(fieldnames(i_g_a_k_107)));
datai_g_a_k_109 = i_g_a_k_109.(char(fieldnames(i_g_a_k_109)));
datai_g_a_k_111 = i_g_a_k_111.(char(fieldnames(i_g_a_k_111)));
datai_g_a_k_113 = i_g_a_k_113.(char(fieldnames(i_g_a_k_113)));
datai_g_a_k_115 = i_g_a_k_115.(char(fieldnames(i_g_a_k_115)));

datai_g_a_r_k_041 = i_g_a_r_k_041.(char(fieldnames(i_g_a_r_k_041)));
datai_g_a_r_k_043 = i_g_a_r_k_043.(char(fieldnames(i_g_a_r_k_043)));
datai_g_a_r_k_045 = i_g_a_r_k_045.(char(fieldnames(i_g_a_r_k_045)));
datai_g_a_r_k_047 = i_g_a_r_k_047.(char(fieldnames(i_g_a_r_k_047)));
datai_g_a_r_k_049 = i_g_a_r_k_049.(char(fieldnames(i_g_a_r_k_049)));
datai_g_a_r_k_051 = i_g_a_r_k_051.(char(fieldnames(i_g_a_r_k_051)));
datai_g_a_r_k_053 = i_g_a_r_k_053.(char(fieldnames(i_g_a_r_k_053)));
datai_g_a_r_k_055 = i_g_a_r_k_055.(char(fieldnames(i_g_a_r_k_055)));
datai_g_a_r_k_057 = i_g_a_r_k_057.(char(fieldnames(i_g_a_r_k_057)));
datai_g_a_r_k_059 = i_g_a_r_k_059.(char(fieldnames(i_g_a_r_k_059)));
datai_g_a_r_k_061 = i_g_a_r_k_061.(char(fieldnames(i_g_a_r_k_061)));
datai_g_a_r_k_063 = i_g_a_r_k_063.(char(fieldnames(i_g_a_r_k_063)));
datai_g_a_r_k_065 = i_g_a_r_k_065.(char(fieldnames(i_g_a_r_k_065)));
datai_g_a_r_k_067 = i_g_a_r_k_067.(char(fieldnames(i_g_a_r_k_067)));
datai_g_a_r_k_069 = i_g_a_r_k_069.(char(fieldnames(i_g_a_r_k_069)));
datai_g_a_r_k_071 = i_g_a_r_k_071.(char(fieldnames(i_g_a_r_k_071)));
datai_g_a_r_k_073 = i_g_a_r_k_073.(char(fieldnames(i_g_a_r_k_073)));
datai_g_a_r_k_075 = i_g_a_r_k_075.(char(fieldnames(i_g_a_r_k_075)));
datai_g_a_r_k_077 = i_g_a_r_k_077.(char(fieldnames(i_g_a_r_k_077)));
datai_g_a_r_k_079 = i_g_a_r_k_079.(char(fieldnames(i_g_a_r_k_079)));
datai_g_a_r_k_081 = i_g_a_r_k_081.(char(fieldnames(i_g_a_r_k_081)));
datai_g_a_r_k_083 = i_g_a_r_k_083.(char(fieldnames(i_g_a_r_k_083)));
datai_g_a_r_k_085 = i_g_a_r_k_085.(char(fieldnames(i_g_a_r_k_085)));
datai_g_a_r_k_087 = i_g_a_r_k_087.(char(fieldnames(i_g_a_r_k_087)));
datai_g_a_r_k_089 = i_g_a_r_k_089.(char(fieldnames(i_g_a_r_k_089)));
datai_g_a_r_k_091 = i_g_a_r_k_091.(char(fieldnames(i_g_a_r_k_091)));
datai_g_a_r_k_093 = i_g_a_r_k_093.(char(fieldnames(i_g_a_r_k_093)));
datai_g_a_r_k_095 = i_g_a_r_k_095.(char(fieldnames(i_g_a_r_k_095)));
datai_g_a_r_k_097 = i_g_a_r_k_097.(char(fieldnames(i_g_a_r_k_097)));
datai_g_a_r_k_099 = i_g_a_r_k_099.(char(fieldnames(i_g_a_r_k_099)));
datai_g_a_r_k_101 = i_g_a_r_k_101.(char(fieldnames(i_g_a_r_k_101)));
datai_g_a_r_k_103 = i_g_a_r_k_103.(char(fieldnames(i_g_a_r_k_103)));

```

```

datai_g_a_r_k_105 = i_g_a_r_k_105.(char(fieldnames(i_g_a_r_k_105)));
datai_g_a_r_k_107 = i_g_a_r_k_107.(char(fieldnames(i_g_a_r_k_107)));
datai_g_a_r_k_109 = i_g_a_r_k_109.(char(fieldnames(i_g_a_r_k_109)));
datai_g_a_r_k_111 = i_g_a_r_k_111.(char(fieldnames(i_g_a_r_k_111)));
datai_g_a_r_k_113 = i_g_a_r_k_113.(char(fieldnames(i_g_a_r_k_113)));
datai_g_a_r_k_115 = i_g_a_r_k_115.(char(fieldnames(i_g_a_r_k_115)));

datos = 8000;
Ts = cumsum(dataTs_k_x);

%% FFT PARTE 1
Mshe = input('Ingrese Mshe (desde 0.41 a 1.15 impares): ');

Mshe_scaled = round(Mshe * 100);

Mshe_str = sprintf('%03d', Mshe_scaled);

data_variable_name = ['datai_g_a_k_' Mshe_str];
data = eval(data_variable_name);

V = data(2156:6248, 2);

f = input('Ingrese resoluci\''{o}'n (mayor o igual a 1): ');

Fs = 50 * 372;
T = 1 / Fs;
cdt = length(V);
t = (0:cdt-1) * T;

L = f * cdt;

t = (0:L-1) * T;

A = zeros(L, 1);
A(1:cdt) = V;

Y = fft(A);
P2 = abs(Y / cdt);
P1 = P2(1:L/2+1);
P1(2:end-1) = 2 * P1(2:end-1);
frequencies = Fs * (0:(L/2)) / L;

figure3 = figure(3);

set(figure3, 'Position', [400, 400, 1200*0.6, 300*0.6]);

bar(frequencies, P1, 1)
xlim([0 3000])

```

```

ylim([0 22])
title('Espectro de frecuencia i\g\_a\_k','Interpreter','Latex')
ylabel('Magnitud [-]','Interpreter','Latex')
xlabel('Frecuencia [Hz]','Interpreter','Latex')
grid on
hold on

carpetaDestino = 'C:\Users\Danie\Documents\Escritura\Formato para memorias
y tesis\imag\';
nombreArchivo = ['fft_imp.eps'];

rutaCompleta = fullfile(carpetaDestino, nombreArchivo);

print(rutaCompleta, '-depsc');
hold on
figure4 = figure(4);

set(figure4, 'Position', [400, 400, 1200*0.6, 300*0.6]);

bar(frequencies, P1, 1)
xlim([0 3000])
ylim([0 0.15])
title('Espectro de frecuencia i\g\_a\_k','Interpreter','Latex')
ylabel('Magnitud [-]','Interpreter','Latex')
xlabel('Frecuencia [Hz]','Interpreter','Latex')
grid on
hold on
carpetaDestino = 'C:\Users\Danie\Documents\Escritura\Formato para memorias
y tesis\imag\';
nombreArchivo = ['fft_imp_zoom.eps'];

rutaCompleta = fullfile(carpetaDestino, nombreArchivo);

print(rutaCompleta, '-depsc');
k = 1;
f = input('Ingrese resoluci\''{o}n (mayor o igual a 1): ');
for i = 0.41:0.02:1.15

    Mshe = i;

    Mshe_scaled = round(Mshe * 100);

    Mshe_str = sprintf('%03d', Mshe_scaled);

    data_variable_name = ['datai_g_a_k_' Mshe_str];
    data = eval(data_variable_name);

```

```

V = data(2156:6248,2);

Fs = 50 * 372;
T = 1 / Fs;
cdt = length(V);
t = (0:cdt-1) * T;

L = f * cdt;

t = (0:L-1) * T;

A = zeros(L, 1);
A(1:cdt) = V;

Y = fft(A);
P2 = abs(Y / cdt);
P1 = P2(1:L/2+1);
P1(2:end-1) = 2 * P1(2:end-1);
frequencies = Fs * (0:(L/2)) / L;

f_CRITICAS = 0;
for j = 1:60
    f_CRITICAS(j) = j*50;
end

for j = 1:length(f_CRITICAS)
    centro = f_CRITICAS(j);
    delta = 5;
    indice_en_rango = (frequencies >= (centro - delta)) &
        (frequencies <= (centro + delta));

    x_rango = frequencies(indice_en_rango);
    y_rango = P1(indice_en_rango);

    [y_max, idx_max] = max(y_rango);

    x_max = x_rango(idx_max);

    Val_max_f_CRITICAS(j) = y_max;
end

THDsuma = 0;
for j = 2:length(f_CRITICAS)
    THDsuma = THDsuma + Val_max_f_CRITICAS(j)^2;
end
THD(k) = 100*sqrt(THDsuma)/Val_max_f_CRITICAS(1);
k = k + 1;
end

```

```

figure5 = figure(5);

set(figure5, 'Position', [400, 400, 1200*0.6, 300*0.6]);

m = 0.41:0.02:1.15;
bar(m, THD, 0.5)
ylabel('THD%', 'Interpreter', 'Latex')
ylim([0, 1.2])
xlabel('\''indice de modulaci\''on', 'Interpreter', 'Latex')
title('THD vs \''indice de modulaci\''on', 'Interpreter', 'Latex')
grid on
hold on
carpetaDestino = 'C:\Users\Danie\Documents\Escritura\Formato para memorias
y tesis\imag\';
nombreArchivo = ['THD_imp.eps'];

rutaCompleta = fullfile(carpetaDestino, nombreArchivo);

print(rutaCompleta, '-depsc');

```

A.2.11. Gráfico THD y espectro de frecuencia de corriente simulado

```

datos = load('sim_Mshe.mat');
tiempo = datos.sim_fft_corr041.time;

igak041 = datos.sim_fft_corr041.signals.values(:,1);
igak043 = datos.sim_fft_corr043.signals.values(:,1);
igak045 = datos.sim_fft_corr045.signals.values(:,1);
igak047 = datos.sim_fft_corr047.signals.values(:,1);
igak049 = datos.sim_fft_corr049.signals.values(:,1);
igak051 = datos.sim_fft_corr051.signals.values(:,1);
igak053 = datos.sim_fft_corr053.signals.values(:,1);
igak055 = datos.sim_fft_corr055.signals.values(:,1);
igak057 = datos.sim_fft_corr057.signals.values(:,1);
igak059 = datos.sim_fft_corr059.signals.values(:,1);
igak061 = datos.sim_fft_corr061.signals.values(:,1);
igak063 = datos.sim_fft_corr063.signals.values(:,1);
igak065 = datos.sim_fft_corr065.signals.values(:,1);
igak067 = datos.sim_fft_corr067.signals.values(:,1);
igak069 = datos.sim_fft_corr069.signals.values(:,1);
igak071 = datos.sim_fft_corr071.signals.values(:,1);
igak073 = datos.sim_fft_corr073.signals.values(:,1);
igak075 = datos.sim_fft_corr075.signals.values(:,1);
igak077 = datos.sim_fft_corr077.signals.values(:,1);
igak079 = datos.sim_fft_corr079.signals.values(:,1);
igak081 = datos.sim_fft_corr081.signals.values(:,1);
igak083 = datos.sim_fft_corr083.signals.values(:,1);

```

```
igak085 = datos.sim_fft_corr085.signals.values(:,1);
igak087 = datos.sim_fft_corr087.signals.values(:,1);
igak089 = datos.sim_fft_corr089.signals.values(:,1);
igak091 = datos.sim_fft_corr091.signals.values(:,1);
igak093 = datos.sim_fft_corr093.signals.values(:,1);
igak095 = datos.sim_fft_corr095.signals.values(:,1);
igak097 = datos.sim_fft_corr097.signals.values(:,1);
igak099 = datos.sim_fft_corr099.signals.values(:,1);
igak101 = datos.sim_fft_corr101.signals.values(:,1);
igak103 = datos.sim_fft_corr103.signals.values(:,1);
igak105 = datos.sim_fft_corr105.signals.values(:,1);
igak107 = datos.sim_fft_corr107.signals.values(:,1);
igak109 = datos.sim_fft_corr109.signals.values(:,1);
igak111 = datos.sim_fft_corr111.signals.values(:,1);
igak113 = datos.sim_fft_corr113.signals.values(:,1);
igak115 = datos.sim_fft_corr115.signals.values(:,1);

igark041 = datos.sim_fft_corr041.signals.values(:,2);
igark043 = datos.sim_fft_corr043.signals.values(:,2);
igark045 = datos.sim_fft_corr045.signals.values(:,2);
igark047 = datos.sim_fft_corr047.signals.values(:,2);
igark049 = datos.sim_fft_corr049.signals.values(:,2);
igark051 = datos.sim_fft_corr051.signals.values(:,2);
igark053 = datos.sim_fft_corr053.signals.values(:,2);
igark055 = datos.sim_fft_corr055.signals.values(:,2);
igark057 = datos.sim_fft_corr057.signals.values(:,2);
igark059 = datos.sim_fft_corr059.signals.values(:,2);
igark061 = datos.sim_fft_corr061.signals.values(:,2);
igark063 = datos.sim_fft_corr063.signals.values(:,2);
igark065 = datos.sim_fft_corr065.signals.values(:,2);
igark067 = datos.sim_fft_corr067.signals.values(:,2);
igark069 = datos.sim_fft_corr069.signals.values(:,2);
igark071 = datos.sim_fft_corr071.signals.values(:,2);
igark073 = datos.sim_fft_corr073.signals.values(:,2);
igark075 = datos.sim_fft_corr075.signals.values(:,2);
igark077 = datos.sim_fft_corr077.signals.values(:,2);
igark079 = datos.sim_fft_corr079.signals.values(:,2);
igark081 = datos.sim_fft_corr081.signals.values(:,2);
igark083 = datos.sim_fft_corr083.signals.values(:,2);
igark085 = datos.sim_fft_corr085.signals.values(:,2);
igark087 = datos.sim_fft_corr087.signals.values(:,2);
igark089 = datos.sim_fft_corr089.signals.values(:,2);
igark091 = datos.sim_fft_corr091.signals.values(:,2);
igark093 = datos.sim_fft_corr093.signals.values(:,2);
igark095 = datos.sim_fft_corr095.signals.values(:,2);
igark097 = datos.sim_fft_corr097.signals.values(:,2);
igark099 = datos.sim_fft_corr099.signals.values(:,2);
igark101 = datos.sim_fft_corr101.signals.values(:,2);
```

```

igark103 = datos.sim_fft_corr103.signals.values(:,2);
igark105 = datos.sim_fft_corr105.signals.values(:,2);
igark107 = datos.sim_fft_corr107.signals.values(:,2);
igark109 = datos.sim_fft_corr109.signals.values(:,2);
igark111 = datos.sim_fft_corr111.signals.values(:,2);
igark113 = datos.sim_fft_corr113.signals.values(:,2);
igark115 = datos.sim_fft_corr115.signals.values(:,2);

%% FFT PARTE 1
Mshe = input('Ingrese Mshe (desde 0.41 a 1.15 impares): ');

Mshe_scaled = round(Mshe * 100);

Mshe_str = sprintf('%03d', Mshe_scaled);

data_variable_name = ['datos.sim_fft_corr' Mshe_str '.signals.values'];
data = eval(data_variable_name);

%% FFT PARTE 2

V = data(2156:6248, 1);

f = input('Ingrese resoluci\''{o}'n (mayor o igual a 1): ');

Fs = 50 * 372;
T = 1 / Fs;
cdt = length(V);
t = (0:cdt-1) * T;

L = f * cdt;

t = (0:L-1) * T;

A = zeros(L, 1);
A(1:cdt) = V;

Y = fft(A);
P2 = abs(Y / cdt);
P1 = P2(1:L/2+1);
P1(2:end-1) = 2 * P1(2:end-1);
frequencies = Fs * (0:(L/2)) / L;

figure3 = figure(3);

set(figure3, 'Position', [400, 400, 1200*0.6, 300*0.6]);

bar(frequencies, P1, 1)
xlim([0 3000])

```

```

ylim([0 22])
title('Espectro de frecuencia i\g\_a\_k','Interpreter','Latex')
ylabel('Magnitud [-]','Interpreter','Latex')
xlabel('Frecuencia [Hz]','Interpreter','Latex')
grid on
hold on

carpetaDestino = 'C:\Users\Danie\Documents\Escritura\Formato para memorias
y tesis\imag\';
nombreArchivo = ['fft_sim.eps'];

rutaCompleta = fullfile(carpetaDestino, nombreArchivo);

print(rutaCompleta, '-depsc');

figure4 = figure(4);

set(figure4, 'Position', [400, 400, 1200*0.6, 300*0.6]);

bar(frequencies, P1, 1)
xlim([0 3000])
ylim([0 0.15])
title('Espectro de frecuencia i\g\_a\_k','Interpreter','Latex')
ylabel('Magnitud [-]','Interpreter','Latex')
xlabel('Frecuencia [Hz]','Interpreter','Latex')
grid on

carpetaDestino = 'C:\Users\Danie\Documents\Escritura\Formato para memorias
y tesis\imag\';
nombreArchivo = ['fft_sim_zoom.eps'];

rutaCompleta = fullfile(carpetaDestino, nombreArchivo);

print(rutaCompleta, '-depsc');

k = 1;
f = input('Ingrese resoluci\''{o}'n (mayor o igual a 1): ');
for i = 0.41:0.02:1.15

    Mshe = i;

    Mshe_scaled = round(Mshe * 100);

    Mshe_str = sprintf('%03d', Mshe_scaled);

    data_variable_name = ['datos.sim_fft_corr' Mshe_str '.signals.values'];
    data = eval(data_variable_name);

```

```

V = data(2156:6248,1);

Fs = 50 * 372;
T = 1 / Fs;
cdt = length(V);
t = (0:cdt-1) * T;

L = f * cdt;

t = (0:L-1) * T;

A = zeros(L, 1);
A(1:cdt) = V;

Y = fft(A);
P2 = abs(Y / cdt);
P1 = P2(1:L/2+1);
P1(2:end-1) = 2 * P1(2:end-1);
frequencies = Fs * (0:(L/2)) / L;

f_CRITICAS = 0;
for j = 1:60
    f_CRITICAS(j) = j*50;
end

for j = 1:length(f_CRITICAS)
    centro = f_CRITICAS(j);
    delta = 5;
    indice_en_rango = (frequencies >= (centro - delta)) &
        (frequencies <= (centro + delta));

    x_rango = frequencies(indice_en_rango);
    y_rango = P1(indice_en_rango);

    [y_max, idx_max] = max(y_rango);

    x_max = x_rango(idx_max);

    Val_max_f_CRITICAS(j) = y_max;
end

THDsuma = 0;
for j = 2:length(f_CRITICAS)
    THDsuma = THDsuma + Val_max_f_CRITICAS(j)^2;
end
THD(k) = 100*sqrt(THDsuma)/Val_max_f_CRITICAS(1);

```

```

    k = k + 1;
end

figure5 = figure(5);

set(figure5, 'Position', [400, 400, 1200*0.6, 300*0.6]);

m = 0.41:0.02:1.15;
bar(m, THD, 0.5)
ylabel('THD%', 'Interpreter', 'Latex')
ylim([0, 1.2])
xlabel('\'\'indice de modulaci\'\'on', 'Interpreter', 'Latex')
title('THD vs \'\'indice de modulaci\'\'on', 'Interpreter', 'Latex')
grid on

carpetaDestino = 'C:\Users\Danie\Documents\Escritura\Formato para memorias
y tesis\imag\';
nombreArchivo = ['THD_sim.eps'];

rutaCompleta = fullfile(carpetaDestino, nombreArchivo);

print(rutaCompleta, '-depsc');
```

A.2.12. Gráfico Vectores de tensión plano $\alpha\beta$ 7 niveles

```

valores = [-3,-2,-1, 0, 1, 2, 3];

[A, B, C] = ndgrid(valores, valores, valores);

combinaciones = [A(:), B(:), C(:)];

indices1 = [];
indices2 = [];
indices3 = [];
indices4 = [];
indices5 = [];
indices6 = [];
indices7 = [];

for i = 1:length(combinaciones(:,1))
    Ares(i) = 2/3*(A(i) - 1/2 * B(i) - 1/2 * C(i));
    Bres(i) = 2/3*(0*A(i) + sqrt(3)/2 * B(i) - sqrt(3)/2 * C(i));
    Cres(i) = 1/2*(1/2*A(i) + 1/2 * B(i) + 1/2 * C(i));
    if (abs(Ares(i) - (-2)) < 0.01 && abs(Bres(i) - 3.46410161513775)
        < 0.01)
        indices1 = [indices1; i];
    end
    if (abs(Ares(i) - (-1.3333)) < 0.01 && abs(Bres(i) - 3.46410161513775)
```

```

    < 0.01)
        indices2 = [indices2; i];
    end
    if (abs(Ares(i) - (-2.3333)) < 0.01 && abs(Bres(i) - 2.88675) < 0.01)
        indices3 = [indices3; i];
    end
    if (abs(Ares(i) - (-1.6666)) < 0.01 && abs(Bres(i) - 2.88675) < 0.01)
        indices4 = [indices4; i];
    end
    if (abs(Ares(i) - (-1)) < 0.01 && abs(Bres(i) - 2.88675) < 0.01)
        indices5 = [indices5; i];
    end
    if (abs(Ares(i) - (-2)) < 0.01 && abs(Bres(i) - 2.3094) < 0.01)
        indices6 = [indices6; i];
    end
    if (abs(Ares(i) - (-1.3333)) < 0.01 && abs(Bres(i) - 2.3094) < 0.01)
        indices7 = [indices7; i];
    end
end

figure1 = figure(1);
set(figure1, 'Position', [0, 0, 1200*0.6, 1200*0.6]);
plot(Ares, Bres, 'o', 'MarkerSize', 5, 'MarkerEdgeColor',
'k', 'MarkerFaceColor', 'k');
axis equal
ylim([-4.5,4.5])
xlim([-4.5,4.5])
xlabel('$\alpha$', 'Interpreter', 'latex')
ylabel('$\beta$', 'Interpreter', 'latex')
title('Vectores plano $\alpha\beta$', 'Interpreter', 'latex')

```

A.2.13. Gráfico Sub-héxagono plano $\alpha\beta$ 7 niveles

```

vector1 = [A(indices1),B(indices1),C(indices1)]
vector2 = [A(indices2),B(indices2),C(indices2)]
vector3 = [A(indices3),B(indices3),C(indices3)]
vector4 = [A(indices4),B(indices4),C(indices4)]
vector5 = [A(indices5),B(indices5),C(indices5)]
vector6 = [A(indices6),B(indices6),C(indices6)]
vector7 = [A(indices7),B(indices7),C(indices7)]

figure2 = figure(2);
set(figure2, 'Position', [0, 0, 1200*0.6, 1200*0.6]);
plot(Ares, Bres, 'o', 'MarkerSize', 5, 'MarkerEdgeColor',
'k', 'MarkerFaceColor', 'k');
axis equal

```

```
ylim([1.1547,4.5])
xlim([-3.3333,0.6666])
xlabel('$\alpha$', 'Interpreter', 'latex')
ylabel('$\beta$', 'Interpreter', 'latex')
title('Vectores plano $\alpha\beta$', 'Interpreter', 'latex')
```

A.2.14. Ángulos patrón SHM

```
load('alpha.mat')
m = alpha(:,10);
a1 = alpha(:,1);
a2 = alpha(:,2);
a3 = alpha(:,3);
a4 = alpha(:,4);
a5 = alpha(:,5);
a6 = alpha(:,6);
a7 = alpha(:,7);
a8 = alpha(:,8);
a9 = alpha(:,9);
plot(m,a1)
hold on
plot(m,a2)
plot(m,a3)
plot(m,a4)
plot(m,a5)
plot(m,a6)
plot(m,a7)
plot(m,a8)
plot(m,a9)
xlim([0.4,1.3])
legend('$a_1$', '$a_2$', '$a_3$', '$a_4$', '$a_5$', '$a_6$', '$a_7$', '$a_8$',
'$a_9$', 'Interpreter', 'latex')
xlabel('\''Indice de modulaci\''on[-]', 'Interpreter', 'latex')
ylabel('\''Angulos[rad]', 'Interpreter', 'latex')
title('\''Angulos SHM', 'Interpreter', 'latex')
```

A.2.15. Comparación hexágonos

```
% Definir los valores posibles
valores = [-3,-2,-1, 0, 1, 2, 3];

% Generar todas las combinaciones posibles
[A, B, C] = ndgrid(valores, valores, valores);

% Convertir las combinaciones en una matriz
```

```

combinaciones = [A(:), B(:), C(:)];

indices1 = [];
indices2 = [];
indices3 = [];
indices4 = [];
indices5 = [];
indices6 = [];
indices7 = [];

for i = 1:length(combinaciones(:,1))
    Ares(i) = 2*A(i) - 1 * B(i) - 1 * C(i);
    Bres(i) = 1 * B(i) - 1 * C(i);
    Cres(i) = 1/2*(1/2*A(i) + 1/2 * B(i) + 1/2 * C(i));
end

figure1 = figure(1);
set(figure1, 'Position', [0, 0, 1200*0.6, 1200*0.6]);
plot(Ares, Bres, 'o', 'MarkerSize', 5, 'MarkerEdgeColor',
'k', 'MarkerFaceColor', 'k');
axis equal
ylim(3*[-4.5,4.5])
xlim(3*[-4.5,4.5])
xlabel('$\alpha$', 'Interpreter', 'latex')
ylabel('$\beta$', 'Interpreter', 'latex')
title('Vectores plano $\alpha\beta$', 'Interpreter', 'latex')

% Definir los valores posibles
valores = [-3,-2,-1, 0, 1, 2, 3];

% Generar todas las combinaciones posibles
[A, B, C] = ndgrid(valores, valores, valores);

% Convertir las combinaciones en una matriz
combinaciones = [A(:), B(:), C(:)];

indices1 = [];
indices2 = [];
indices3 = [];
indices4 = [];
indices5 = [];
indices6 = [];
indices7 = [];

for i = 1:length(combinaciones(:,1))
    Ares(i) = 2/3*(A(i) - 1/2 * B(i) - 1/2 * C(i));
    Bres(i) = 2/3*(0*A(i) + sqrt(3)/2 * B(i) - sqrt(3)/2 * C(i));
    Cres(i) = 1/2*(1/2*A(i) + 1/2 * B(i) + 1/2 * C(i));

```

```

end

figure1 = figure(1);
set(figure1, 'Position', [0, 0, 1200*0.6, 1200*0.6]);
plot(Ares, Bres, 'o', 'MarkerSize', 5, 'MarkerEdgeColor', 'k',
     'MarkerFaceColor', 'k');
axis equal
ylim(3*[-4.5,4.5])
xlim(3*[-4.5,4.5])
xlabel('$\alpha$', 'Interpreter', 'latex')
ylabel('$\beta$', 'Interpreter', 'latex')
title('Vectores plano $\alpha\beta$', 'Interpreter', 'latex')

```

A.2.16. Algoritmo forma SHM

```

t = linspace(0, 1/372, 1000);
y = linspace(0, 2*pi, 1000);

subplot(2,1,1)
plot(t,y, 'LineWidth', 1, 'Color', [0 0 0])
grid on
hold on
xlim([0,1/372])
ylim([0,2*pi])
xlabel('Tiempo [s]', 'Interpreter', 'latex')
ylabel('wt\_she\_a\_k1 [rad]', 'Interpreter', 'latex')
linea1x = (1/372)*(1/4)*ones(1,100);
linea2x = (1/372)*(2/4)*ones(1,100);
linea3x = (1/372)*(3/4)*ones(1,100);
lineay = linspace(0,2*pi,100);
plot(linea1x,lineay,'--', 'LineWidth', 1, 'Color',1/255*[100 100
100])%1/255*[255 102 102])%[0.8500 0.3250 0.0980])
plot(linea2x,lineay,'--', 'LineWidth', 1, 'Color',1/255*[100 100
100])%[0.8500 0.3250 0.0980])
plot(linea3x,lineay,'--', 'LineWidth', 1, 'Color',1/255*[100 100
100])%[0.8500 0.3250 0.0980])

set(gca, 'YTick', [0,pi,2*pi])
set(gca, 'YTickLabel', {'$0$', '$\pi$', '$2\pi$'}, 'TickLabelInterpreter',
'latex');

set(gca, 'XTick', [0,6.72043010752690e-05,0.000134408602150538,
0.000201612903225806,0.000268817204301075,
0.000336021505376344,0.000403225806451613,
0.000470430107526883,0.000537634408602152,
0.000604838709677421,0.000739247311827959,
0.000806451612903228,0.000873655913978497,

```

```

0.000940860215053766,0.00100806451612903,
0.00107526881720430,0.00114247311827957,
0.00120967741935484,0.00127688172043011,
0.00134408602150538,0.00141129032258064,
0.00147849462365591,0.00154569892473118,
0.00161290322580645,0.00168010752688172,
0.00174731182795699,0.00181451612903226,
0.00188172043010753,0.00194892473118280,
0.00208333333333333,0.00215053763440860,
0.00221774193548387,0.00228494623655914,
0.00235215053763441,0.00241935483870968,
0.00248655913978495,0.00255376344086022,
0.00262096774193549,0.00268817204301075]);
% Asignar etiquetas personalizadas
set(gca, 'XTickLabel', {0,'$a_1$', '$a_2$', '$a_3$', '$a_4$', '$a_5$',
'$a_6$', '$a_7$', '$a_8$', '$a_9$', '$b_1$', '$b_2$', '$b_3$', '$b_4$', '$b_5$',
'$b_6$', '$b_7$', '$b_8$', '$b_9$', '$t_2$', '$c_1$', '$c_2$', '$c_3$', '$c_4$',
'$c_5$', '$c_6$', '$c_7$', '$c_8$', '$c_9$', '$d_1$', '$d_2$', '$d_3$', '$d_4$',
'$d_5$', '$d_6$', '$d_7$', '$d_8$', '$d_9$', ''}, 'TickLabelInterpreter', 'latex');

PI2 = 2*pi;
PI = pi;
T_0 = PI2; % Periodo
t2 = PI; % Mitad de periodo
t4 = T_0; % Periodo completo

a = [0, 0.228528527609550,0.256312349851493,0.765833961128487,
1.34190099359899,1.40926675328107,1.42280402645180,
1.45331255832309,1.45392877494162,1.56993386513436, pi/2];
%a = linspace(0,pi/2,11);
a(11) = [];
b = t2 - a;
c = t2 + a;
d = t4 - a;
hold on

C = 0;
U = 1;
D = 2;
T = 3;

abcdt = sort([a,b]);
subplot(2,1,2)
abcdres = [C,U,C,U,D,U,D,T,D,T,T,D,T,D,U,D,U,C,U,C,C,C,-U,C,-U,-D,-U,
-D,-T,-D,-T,-T,-D,-T,-D,-U,-D,-U,C,-U,C,C];
stairs([abcdt(1:10),abcdt(10:20),pi+abcdt(1:10),pi+abcdt(10:20)],
transpose(abcdres),'LineWidth', 1, 'Color', [0 0 0])
ylim([-3.5,3.5])

```

```

grid on
hold on
xlim([0,2*pi])
xlabel('wt\_she\_a\_k1 [rad]', 'Interpreter', 'latex')
ylabel('v\_she\_a\_k1 [-]', 'Interpreter', 'latex')
linea1x = (2*pi)*(1/4)*ones(1,100);
linea2x = (2*pi)*(2/4)*ones(1,100);
linea3x = (2*pi)*(3/4)*ones(1,100);
lineay = linspace(-3.5,3.5,100);
plot(linea1x,lineay,'--', 'LineWidth', 1, 'Color',1/255*[100 100
100])%[0.8500 0.3250 0.0980])
plot(linea2x,lineay,'--', 'LineWidth', 1, 'Color',1/255*[100 100
100])%[0.8500 0.3250 0.0980])
plot(linea3x,lineay,'--', 'LineWidth', 1, 'Color',1/255*[100 100
100])%[0.8500 0.3250 0.0980])

set(gca, 'YTick', [-3,-2,-1,0,1,2,3], 'TickLabelInterpreter', 'latex')

set(gca, 'XTick', [0,0.157079632679490,0.314159265358979,0.471238898038469,
0.628318530717959,0.785398163397448,0.942477796076938,
1.09955742875643,1.25663706143592,1.41371669411541,
1.57079632679490,1.72787595947439,1.88495559215388,2.04203522483337,
2.19911485751286,2.35619449019235,2.51327412287183,
2.67035375555132,2.82743338823081,2.98451302091030,
3.14159265358979,3.29867228626928,3.45575191894877,
3.61283155162826,3.76991118430775,3.92699081698724,
4.08407044966673,4.24115008234622,4.39822971502571,
4.55530934770520,4.71238898038469,4.86946861306418,5.02654824574367,
5.18362787842316,5.34070751110265,5.49778714378214,
5.65486677646163,5.81194640914112,5.96902604182061,
6.12610567450010,6.28318530717959]);
% Asignar etiquetas personalizadas
set(gca, 'XTickLabel', {0,'$a_1$', '$a_2$', '$a_3$', '$a_4$', '$a_5$',
'$a_6$', '$a_7$', '$a_8$', '$a_9$', '$b_1$', '$b_2$', '$b_3$', '$b_4$',
'$b_5$', '$b_6$', '$b_7$', '$b_8$', '$b_9$', '$t_2$', '$c_1$', '$c_2$',
'$c_3$', '$c_4$', '$c_5$', '$c_6$', '$c_7$', '$c_8$', '$c_9$', '$d_1$',
'$d_2$', '$d_3$', '$d_4$', '$d_5$', '$d_6$', '$d_7$', '$d_8$', '$d_9$'}, 'TickLabelInterpreter', '

```

A.3. Tablas

A.3.1. Tensión de salida

```

\begin{table}[htbp]
\renewcommand{\arraystretch}{1.3}
\centering

```

```

\caption{Tensi\'{o}n de salida}\label{tab:tabulas}
\begin{tabular}{|c|c|c|c|c|c|}
\hline
 $S_{x1}$  &  $S_{x2}$  &  $S_{x3}$  &  $S_{x4}$  & & \\
& Voltaje de Salida  $v_o$  ( $i_o > 0$ ) & & & & \\
& Voltaje de Salida  $v_o$  ( $i_o < 0$ ) & & & & \\
\hline\hline
0 & 0 & 0 & 0 &  $-v_{cc}$  &  $v_{cc}$  \\
0 & 0 & 0 & 1 & 0 &  $v_{cc}$  \\
0 & 0 & 1 & 0 &  $-v_{cc}$  & 0 \\
0 & 1 & 0 & 0 &  $-v_{cc}$  & 0 \\
1 & 0 & 0 & 0 & 0 &  $v_{cc}$  \\
1 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 &  $v_{cc}$  &  $v_{cc}$  \\
0 & 1 & 1 & 0 &  $-v_{cc}$  &  $-v_{cc}$  \\
0 & 1 & 0 & 1 & 0 & 0 \\
\hline
\end{tabular}
\end{table}

```

A.3.2. Tensión v_{an}

```

\begin{table}[htbp]
\renewcommand{\arraystretch}{1.3}
\centering
\caption{Tensi\'{o}n  $v_{an}$ }\label{tab:vxn}
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline
 $S_{a11}$  &  $S_{a13}$  &  $S_{a21}$  &  $S_{a23}$  &  $S_{a31}$  &  $S_{a33}$  &  $v_{an}$  \\
\hline\hline
1 & 0 & 1 & 0 & 1 & 0 &  $3v_{cc}$  \\
1 & 0 & 1 & 0 & 0 & 0 &  $2v_{cc}$  \\
1 & 0 & 0 & 0 & 0 & 0 &  $v_{cc}$  \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 &  $-v_{cc}$  \\
0 & 1 & 0 & 1 & 0 & 0 &  $-2v_{cc}$  \\
0 & 1 & 0 & 1 & 0 & 1 &  $-3v_{cc}$  \\
\hline
\end{tabular}
\end{table}

```

A.3.3. Tabla lógica XOR

```

\begin{table}[htbp]
\renewcommand{\arraystretch}{1.3}

```

```

\centering
\caption{Tabla 1\'}{o}gica XOR}\label{tab:xor}
\begin{tabular}{|c|c|c|}
\hline A & B & A XOR B \\
\hline\hline
000 & 011 & 011 \\
001 & 011 & 010 \\
010 & 011 & 001 \\
011 & 011 & 000 \\
\hline
\end{tabular}
\end{table}

```

A.3.4. Parámetros

```

\begin{table}[htbp]
\renewcommand{\arraystretch}{1.3}
\centering
\caption{Par\'}{a}metros}\label{tab:parametros}
\begin{tabular}{|c|c|c|}
\hline $v_{cc}$ [V] & $R$ [$\Omega$] & $L$ [H] \\
\hline\hline
50 & 1 & 0.02 \\
\hline
\end{tabular}
\end{table}

```

Bibliografía

- [1] B. P. McGrath and D. G. Holmes, “Multicarrier PWM Strategies for Multilevel Inverters,” *IEEE Trans. Ind. Applicat.*, vol. 49, Aug. 2002.
- [2] J. Cheng, T. Xu, D. Chen, and G. Chen, “Dynamic and steady state response analysis of selective harmonic elimination in high power inverters,” *IEEE Access*, vol. 9, pp. 75 588–75 598, 2021.
- [3] Institute of Electrical and Electronics Engineers, “Ieee standard 519-2022: Recommended practices and requirements for harmonic control in electrical power systems,” *IEEE Standards*, 2022.
- [4] M. Marchesoni, M. Mazzucchelli, and S. Tenconi, “A non conventional power converter for plasma stabilization,” in *Proc. IEEE 19th Power Electron. Spec. Conf.*, Apr. 11–14 1998, pp. 122–129.
- [5] P. Kouro, M. Malinowski, K. Gopakumar, J. Pou, L. G. Franquelo, B. Wu, J. Rodriguez, M. A. Pérez, and J. I. Leon, “Recent Advances and Industrial Applications of Multilevel Converters,” *IEEE Trans. Ind. Applicat.*, vol. 57, 2010.
- [6] P. Lezana, “Manejo de la Plataforma Digital DSP+FPGA para el control de Accionamientos,” Departamento de Electricidad Universidad Técnica Federico Santa María, Valparaíso, Chile 2020.
- [7] P. W. Hammond, “A New Approach to Enhance Power Quality for Medium Voltaje AC Drives,” *IEEE Trans. Ind. Applicat.*, vol. 33, Jan./Feb. 1997.
- [8] Ricardo P. Aguilera, Pablo Acuña, Pablo Lezana, Georgios Konstantinou, Bin Wu, Steffen Bernet, and Vassilios G. Agelidis, “Selective harmonic elimination model predictive control for multilevel power converters,” *IEEE Transactions on Power Electronics*, vol. 32, no. 3, pp. 2416–2426, 2017.
- [9] Samir Kouro, Jose I. Leon, Dimitri Vinnikov, and Leopoldo G. Franquelo, “Grid-connected photovoltaic systems: An overview of recent research and emerging pv converter technology,” *IEEE Industrial Electronics Magazine*, vol. 9, no. 1, pp. 47–61, 2015.