Universidad de Pamplona
Programa de Ingeniería Eléctrica
Facultad de Arquitectura e Ingenierías
Departamento de Eléctrica, Electrónica, Sistemas y Telecomunicaciones

SIMULADOR DE EVENTOS DE LA CALIDAD DE LA ENERGÍA

Autora:

MELISSA CAROLINA PATERNINA JIMÉNEZ

PAMPLONA, NORTE DE SANTANDER
19 DE JUNIO DE 2020

Universidad de Pamplona
Programa de Ingeniería Eléctrica
Facultad de Arquitectura e Ingenierías
Departamento de Eléctrica, Electrónica, Sistemas y Telecomunicaciones

SIMULADOR DE EVENTOS DE LA CALIDAD DE LA ENERGÍA

Autora:

MELISSA CAROLINA PATERNINA JIMÉNEZ

Director: Msc. Edison Andrés Caicedo Peñaranda

Co-director: MSc. Luis David Pabón Fernández

TRABAJO DE GRADO PARA OPTAR POR EL TÍTULO DE INGENIERO ELÉCTRICO

PAMPLONA, NORTE DE SANTANDER

19 DE JUNIO DE 2020

“A mis padres, a mi novio y a los incontables tazas de café que me han apoyado a lo largo de este camino y continuarán haciéndolo al día de hoy”

— Melissa Paternina Jiménez

Agradecimientos

A cada uno de mis docentes que hicieron parte de mi formación a nivel profesional, en especial a mi director y co-director de tesis, el profesor Edison Caicedo y Luis David Pabón, por su paciencia ante mi inconsistencia, por su dedicación, apoyo y ayuda a lo largo del desarrollo de ese proyecto.

A mi familia por darme su confianza, apoyo y motivación a lo largo de esta etapa de mi vida.

A todos mis amigos y compañeros con los que compartí este camino, en especial a mi novio por que siempre creyó en mí y estuvo presente a lo largo de este proyecto.

Índice

Índice de tablas

Índice de figuras

Resumen

En este proyecto de investigación se ha planteado una herramienta de simulación que proporciona información para simular los eventos de la calidad de la energía estipulados por el estándar IEEE 1159 del 1995, la construcción de este simulador está hecha en base al software MATLAB y su blockset SIMULINK. Se simulan diferentes eventos de la calidad de la energía, los cuales poseen la facilidad de modificar los parámetros generales de cada uno y las características constructivas del convertidor.

Para dar facilidad al usuario se ha elaborado una interfaz gráfica amigable de uso fácil e intuitivo, la cual recoge los parámetros y características de cada evento de la calidad de la energía y visualiza la forma de onda del evento con los parámetros dados, para tal motivo se utilizó el algoritmo PSO (Particle Swarm Optimization) para obtener los ángulos de disparo necesarios para construir las ondas resultantes de los eventos que involucran la magnitud del voltaje RMS y los armónicos. Para finalizar se validaron cada uno de los eventos simulados con ayuda de blocksets de la librería de Power System y/o bloques que se crearon para el presente proyecto.

Palabras clave: Simulador, Eventos, Calidad de la energía, IEEE 1159, PSO.

Abstract

In this research project, a simulation tool has been proposed that provides information to simulate the power quality events stipulated by the IEEE 1159 standard of 1995, the construction of this simulator is based on the MATLAB software and its SIMULINK blockset. . Different power quality events are simulated, which have the facility to modify the general parameters of each one and the constructive characteristics of the converter.

To make the user easier, a user-friendly and intuitive graphical interface has been developed, which collects the parameters and characteristics of each event of the power quality and displays the waveform of the event with the given parameters, for this reason The PSO (Particle Swarm Optimization) algorithm was used to obtain the firing angles necessary to construct the waves resulting from the events involving the magnitude of the RMS voltage and harmonics. Finally, each of the simulated events was validated with the help of Power System library blocksets and blocks that were created for the present project.

Keywords: Simulator, Events, Power Quality, IEEE 1159, PSO

Introducción

Uno de los elementos que hacen parte de la complejidad del estudio de la calidad de la energía es la amplia gama de factores que contribuyen al deterioro de ésta, partiendo desde los componentes no lineales que se encuentran en aparatos de uso doméstico y los de uso industrial, además de los dispositivos usados por las fuentes de energía no convencionales. Otro elemento que aumenta la dificultad en el campo de investigación de la calidad de la energía es la imposibilidad de validar las soluciones propuestas por los investigadores debido a que se hace imposible obtener estos eventos directamente de la red eléctrica [1].

Para adquirir una solución a la mejora de la calidad de la energía se requiere un análisis puntual y específico de cada evento, por tanto, desde la academia se hace necesaria la posibilidad de apreciar y determinar estos eventos que perturban la calidad de la energía, para que de esta manera se proporcionen soluciones puntuales a cada uno. Sin embargo, los equipos especializados capaces de simular estos eventos raramente se encuentran disponibles comercialmente y poseen un precio elevado [2, 3].

Debido a lo anteriormente mencionado, con el presente proyecto se realizó el diseño de un simulador el cual cuenta con la capacidad de generar diversos eventos de la calidad de la energía de manera precisa en términos de tiempo y magnitud.

Primer Capítulo

Problema y justificación

La discusión sobre la regulación de la calidad en el suministro del fluido eléctrico es un área de importancia, debido a que los eventos asociados a la operación de sistemas eléctricos puede ocasionar un mal funcionamiento en diferentes equipos o cargas altamente sensibles a perturbaciones electromagnéticas, una operación inadecuada en las redes eléctricas, y pueden llegar a conducir a fallas e incrementos en los costos de operación, lo cual acarrearía en pérdidas para las compañías encargadas del suministro de energía [4]. A causa de esto, se han empezado a realizar investigaciones que buscan elaborar equipos acondicionadores que sean capaces de mitigar los eventos de la calidad de la energía que se encuentran presentes en las redes eléctricas, y de esta forma mejorar las condiciones de alimentación de los equipos y dispositivos con el objetivo de salvaguardar su vida útil [5].

Para la validación del funcionamiento óptimo de los equipos mencionados anteriormente, se requiere el uso de aparatos especializados [2] que generen diversos eventos de la calidad de la energía para tener la certeza que el equipo propuesto mitiga correctamente el o los eventos asociados. Hoy en día, la disponibilidad de estos simuladores de eventos de calidad de la energía es reducida debido a los altos costos [3], lo que los hacen poco accesibles para la investigación en ambientes académicos.

En virtud a lo señalado anteriormente, la realización de este proyecto tiene como finalidad el diseño de un simulador de eventos de la calidad de la energía con tecnología asequible para su futura implementación con el que se pueda realizar pruebas a los equipos acondicionadores de potencia, electrodomésticos e incluso equipos industriales. También permitirá realizar estudios de correlación

entre calidad de la energía y eficiencia energética, un tema que posee una gran trascendencia en el ámbito científico actualmente [6]. Por otra parte, se podrán realizar estudios detallados de investigación referentes a la calidad de la energía de forma experimental en aparatos reales.

Segundo Capítulo

Delimitación

2.1. Objetivos

2.1.1. Objetivo General

Diseñar un simulador de eventos de la calidad de la energía presentes en el estándar IEEE 1159 de 1995.

2.1.2. Objetivos Específicos

- Determinar los eventos de calidad de la energía a generar en el simulador según el estándar IEEE 1159 de 1995.
- Programar los algoritmos que permitan la generación de los eventos de la calidad de la energía mediante el simulador diseñado.
- Diseñar la interfaz gráfica del simulador.
- Validar el funcionamiento del simulador de eventos de calidad de la energía.

2.2. Acotaciones

- La totalidad de los eventos a generar por el simulador realizado se determinará durante el desarrollo de la investigación y de acuerdo a la viabilidad en la consecución de cada uno de ellos.
- Se excluirán los eventos transitorios tanto impulsivos como oscilatorios.
- Se utilizarán en primera instancia los softwares disponibles en el programa de ingeniería eléctrica y el grupo de investigación *Sistemas Energéticos*.

Tercer Capítulo

Marco referencial

3.1. Estado del arte

En los últimos años el tema sobre la calidad de la energía ha adquirido una gran importancia en el sector eléctrico, por tanto el número de investigaciones con información sobre su base teórica, normativa, análisis en los sistemas de potencia, equipos para mitigar los problemas de calidad de la energía, entre otros ha ido en aumento; sin embargo, son escasos los estudios asociados al diseño o creación de sistemas que generen diferentes eventos de calidad de la energía para poder probar los equipos encargados de mitigar dichos eventos.

A continuación, se contemplan algunas de las investigaciones más relevantes de la bibliografía disponible sobre el tema de estudio, que permite recopilar diversos puntos de vista conceptuales y metodológicos con el propósito de garantizar un proceso analítico, comparativo y constructivo en este proyecto.

Kezunovic Mladen y Liao Yuan. *"A novel method for equipment sensitivity study during power quality events"* (2000). En esta investigación se propone un método de simulación para distintos eventos de calidad de la energía con el fin de estudiar cómo afectan dichos eventos en las características operacionales de un equipo, en este caso un variador de velocidad VSD, de esta manera el software creado puede explicar por qué falló la carga cuando se presentó un evento, o predecir qué tan bien funcionaría durante un evento en particular. Para esto se diseñó una librería en la blockset Simulink de MATLAB utilizando técnicas de procesamiento digital de señales como la transformada de Fourier y el análisis de Wavelet para generar varios tipos de eventos de calidad de la energía: sags, swells, interrupciones, impulsos, muescas y flickers. Una característica peculiar

del método propuesto es que los eventos pueden ser generados por medio de ecuaciones algebraicas o a través de la reproducción de formas de onda previamente grabadas. De cualquier modo, los parámetros de cada evento pueden ser ajustados por el usuario a los valores deseados. El estudio evidencia que el VSD posee mayor sensibilidad a los eventos de sag, swell e interrupción; y que el método propuesto es flexible y factible para aplicaciones prácticas [7]

León Carlos, Montaño Juan Carlos, Ropero Jorge y Elena José Manuel. *"A system for the generation and detection of electrical disturbances"* (2004). En este trabajo se describe un sistema desarrollado por el Instituto de Recursos Naturales y Agrobiología del Consejo de Investigación Científica (CSIC) junto al Departamento de Tecnología Electrónica de la Universidad de Sevilla, el cual tiene como objetivo generar y detectar distintas perturbaciones o eventos de la calidad de la energía (sag, swell, sobrevoltaje, bajovoltaje y desviación de la frecuencia). Para esto se hizo uso de una red neuronal encargada de detectar y clasificar las perturbaciones, y herramientas matemáticas como la Transformada de Fourier y Wavelet para la generación de las características de la señal que servirán como entradas a la red, usan también una técnica llamada análisis de resolución múltiple con la que cada característica única que representa cada evento de la calidad de la energía proporcionada por la transformada, se representa a diferentes resoluciones. Con este estudio se logró el desarrollo de un generador de patrones eléctricos capaz de generar eventos comunes que se pueden encontrar en una línea eléctrica y la primera versión de un sistema clasificador basado en redes neuronales obteniendo un 93.83% de detección correcta [8].

Eloy-García Joaquin, Vasquez Juan Carlos, Guerrero Josep M. *"Grid simulator for power quality assessment of micro-grids""* (2013). En este estudio, se presenta un simulador de red basado en una topología de inversor back-to-back con controladores resonantes. El simulador puede producir voltajes trifásicos para cierto rango de amplitudes y frecuencias con diferentes tipos de perturbaciones, tales como sag, swell, voltajes desbalanceados en estado estacionario, armónicos de orden bajo y flicker. Este equipo tiene como finalidad probar el desempeño de un sistema durante tales eventos. El prototipo del simulador implementado

consta de dos inversores back-to-back conectados a 380 V y que alimenta una micro red formada por generadores distribuidos conectados a dos inversores y una carga crítica. El simulador de red obtuvo un desempeño realmente bueno siendo capaz de rastrear su referencia incluso en presencia de corrientes muy distorsionadas [9].

Nahoum Pamela, Yammine Emile, Karam Elie, Najjar Maged B. y El Hassan Moustapha. *"Real generation of power quality disturbances"* (2015). En este artículo se presenta un generador de eventos de la calidad de la energía probado en el electrocardiograma BIOPAC, quien tendrá el papel de la carga. Los eventos que se generaron fueron: sags, swells, armónicos, y una combinación de sags y swells con armónicos, estos disturbios fueron generados en el software LabView. El propósito de este estudio es investigar si la señal ECG se verá afectada por la introducción de los disturbios mencionados anteriormente. Aunque el experimento no mostró ningún cambio perceptible en la señal grabada del corazón, los autores se abstienen de concluir que estos eventos no afectan directamente a este tipo de equipo biomédico por lo que usaron equipos de bajo costo [10].

Cheng-I Chen, Chieh-Yin Cheng y Yeong-Chin Chen. *"Design of programmable power-quality signal generator for power disturbance testing of consumer electronics"* (2015). En el artículo presentado por Cheng-I Chen et al. se describe todo el proceso de diseño y los métodos utilizados para la puesta en marcha de un generador de señal de calidad de la energía programable el cual será usado como apoyo a la realización de pruebas referentes a los eventos (perturbaciones) de la energía en la electrónica de consumo. La parte programable se da de la forma en que la onda arbitraria puede ser generada y proporcionar una salida de potencia de corriente alterna (CA). Esta salida de potencia con forma de onda arbitraria puede ser usada para emular una fuente de alimentación de contaminación de armónicos de potencia, flicker o parpadeo, muescas, sags (depresiones), swells (elevaciones), y otro tipo de eventos de calidad de la energía conocidos. El sistema creado puede utilizarse para realizar la capacitación para el diseño de un filtro que sirva para filtrar una fuente de energía contaminada (perturbada) en la onda sinusoidal pura o limpia, o puede ser usado también, para probar un

equipo en busca de interferencia de mala calidad de energía. La potencia nominal de la fuente de alimentación se estableció en 300 vatios, se usó un amplificador de potencia de clase D como núcleo del diseño, y un circuito de refuerzo para proporcionar una fuente de alimentación para que el amplificador eleve el voltaje de salida. A partir de los experimentos realizados por los autores se logró evidenciar que las perturbaciones de calidad de la energía comúnmente vistas pueden ser generadas con precisión y realizar el respectivo suministro de energía [11].

Inci Mustafa, Demirdelen Tugce, Tan Adnan, Köroglu Tahsin, Cuma M. Ugras, Bayindir K. Cagatay y Tümay Mehmet. *"A novel low cost sag/swell generator"* (2015). En este estudio se plantea una nueva topología de generador de sag/swell capaz de generar diversos problemas de la calidad de la energía como la elevación de voltaje (swell), depresión de voltaje (sag) e interrupciones. El generador está basado en un conmutador bidireccional de tipo puente para generar diferentes condiciones de falla. La principal ventaja de esta topología es su estructura más simple, de menor costo y un control más fácil en comparación de otros tipos de topologías basadas en transformadores convencionales. Con el desarrollo de esta topología se pudo realizar pruebas a dispositivos de alimentación personalizados como DVR, UPS y STS. El rendimiento del generador planteado se probó para una caída de tensión del 50% y un aumento de tensión del 20% [12].

Shen Zhuoxuan, Duan Tong y Dinavahi Venkata. *"Design and Implementation of Real-Time MPSoC-FPGA-Based Electromagnetic Transient Emulator of CIGRÉ DC Grid for HIL Application"* (2018). Este proyecto de investigación realizado por el grupo de trabajo CIGRÉ en cabeza de Shen Zhuoxuan, propone un sistema de prueba de red de CC, que cubre diferentes configuraciones de HVDC y despliega múltiples niveles modulares convertidores (MMC) en la malla (grid). De manera específica, este trabajo se centra en la solución eficiente del emulador en tiempo real de la red de CC que proporciona resultados precisos y detallados. Para el diseño e implementación de la red CIGRÉ CC se utilizó una plataforma híbrida MPSoC-FPGA quien es la encargada de realizar la sinergia entre el dispositivo Xilinx Vitrex UltraScale+ FPGA, que contiene una gran cantidad de

recursos lógicos, y el dispositivo Xilinx Zynq UltraScale+ MPSoC, que contiene el sistema de procesamiento multinúcleo ARM y recursos FPGA en un solo chip. Para presentar los resultados detallados a nivel de dispositivo del equipo local y los resultados precisos a nivel de sistema de las interacciones globales de la red de CC se hizo uso de una metodología de modelado híbrido que utiliza el modelo electrotérmico a nivel de dispositivo, el modelo de circuito equivalente y el modelo de valor promedio para los convertidores. Los resultados en tiempo real fueron validados con herramientas de simulación comercial PSCAD/EMTDC y SaberRD. En este artículo se llegó a la conclusión que la mejora de los esquemas de modelado y las técnicas de implementación pueden ampliar la funcionalidad y el alcance del estudio EMT y proporcionar análisis integrales de la red AC-DC y que, además, el emulador propuesto puede ser beneficioso para el estudio de control y protección de la red de CC [13].

Ashourianjozdani Mohammadhossein, Lopes Luiz A.C. y Pillay Pragasen. *"Power electronic converter based PMSG emulator: a testbed for renewable energy experiments"* (2018). En este trabajo se esboza un emulador de generador síncrono de imán permanente (PMSG) basado en convertidor como banco de pruebas para diseñar, analizar y probar la interfaz electrónica de potencia del generador y su sistema de control. El modelo PMSG está formulado en un simulador digital en tiempo real. Se presenta como un algoritmo de interfaz un modelo de transformador ideal de tipo de voltaje combinado con una impedancia virtual. También, se utiliza un convertidor de fuente de voltaje de seis interruptores como amplificador de potencia para imitar el comportamiento del PMSG que suministra cargas lineales y no lineales, y se propone un controlador proporcional-integral más resonante como controlador de bucle de voltaje para el seguimiento de una señal de referencia de voltaje de salida distorsionada. La precisión del emulador propuesto se postula para los armónicos de voltaje fundamental y de bajo orden. Como resultados se obtuvo que el controlador proporcionó un seguimiento de voltaje adecuado no solo para el componente fundamental de frecuencia variable, sino también para los componentes armónicos quinto y séptimo que aparecen cuando el PMSG suministra cargas no lineales. Asimismo, los resultados demostraron que el emulador PMSG propuesto funciona bien con alta precisión para

condiciones de cargas tanto lineales como no lineales. Por lo tanto, el emulador PMSG propuesto por los autores se puede utilizar como banco de pruebas para probar y desarrollar sistemas de conversión de energía eólica/hidrocinética. [14].

Parizad Ali, Mohamadian Sobhan, Iranian Mohamad Esmaeil, Guerrero Josep M. *"Power system real-time emulation: a practical virtual instrumentation to complete electric power system modeling"* (2019). En este artículo, se propone una emulación en tiempo real de un sistema completo de energía eléctrica, este sistema incluye generador, regulador de turbina, sistema de excitación, líneas de transmisión, transformador, red externa y cargas relacionadas y se implementó en un entorno MATLAB/Simulink; mientras que las diferentes páginas de instrumentos virtuales están modeladas en el lenguaje de programación gráfico de LabVIEW. Además, un sistema de excitación real de 1518 kW se considera como un caso de prueba para el sistema HIL introducido, este equipo se conecta al software LabVIEW por medio de una tecnología National Instrument PXI. Se simulan diferentes escenarios (frecuencia eléctrica/cambio de potencia activa, respuesta de paso de voltaje, entre otros) en el emulador de sistema de potencia (PSE) diseñado, y se verifica la validez del modelo implementado encontrando una buena correspondencia entre los resultados de simulación MATLAB y HIL [15].

Kenichiro Saito y Hirofumi Akagi. *"A real-time real-power emulator of a medium-voltage high-speed induction motor loaded with a centrifugal compressor"* (2019). En este documento se muestra el diseño e implementación de un emulador de energía real a tiempo real basado en un rectificador DSCC trifásico de un motor de inducción de alta velocidad, media tensión y alta potencia cargado con un compresor centrífugo. El emulador en cuestión posee un inductor auxiliar por fase instalado en el lado de CA y un inductor de modo común (realizado manualmente) en el lado de CC del rectificador DSCC. El valor de inductancia del inductor auxiliar por fase es igual a la inductancia de fuga por fase referida al primario del motor de inducción emulado. Las formas de ondas obtenidas por el experimento confirman que el emulador es capaz de reproducir el rendimiento operativo eléctrico y mecánico del motor de inducción trifásico cargado con un compresor centrífugo, lo cual lo hace un emulador de alta fidelidad, además su

uso trae ahorros de tiempo y costos [16].

Cuarto Capítulo

Marco teórico

Dado que la mira central de este proyecto de investigación estará puesta en los diferentes eventos de la calidad de la energía y en el diseño de un sistema capaz de simular los eventos de la calidad de la energía, se hace indispensable aclarar algunos conceptos o fundamentos teóricos esenciales para comprender a cabalidad las consecuencias de este trabajo y vislumbrar el impacto de las propuestas aquí planteadas.

4.1. Calidad de la energía eléctrica

A continuación se muestran algunas definiciones para el término calidad de la energía eléctrica:

Para la norma IEEE 1159 de 1995: *"El término se refiere a una amplia variedad de eventos electromagnéticos que caracterizan la tensión y la corriente eléctrica, en un tiempo dado y en una ubicación dada en el sistema de potencia"* [5].

La CREG en Colombia en la Resolución 070 de 1998 lo definió como: *"El término calidad de la potencia suministrada se refiere a las perturbaciones y variaciones de estado estacionario de la tensión y corriente suministrada por el operador de red. El término calidad del servicio prestado se refiere a los criterios de confiabilidad del servicio"* [17].

Según la norma IEEE 1100 de 2005, se entiende por calidad de la potencia eléctrica: *"El concepto de alimentar y poner a tierra equipo electrónico de manera que sea adecuado para la operación de dicho equipo y compatible con el sistema de alambrado del local y con otro equipo conectado"* [18].

De acuerdo a la norma IEC 61000-2-2/4 la calidad de la energía eléctrica es: *"Una característica física del suministro de electricidad, la cual debe llegar al cliente en condiciones normales, sin producir perturbaciones ni interrupciones en los procesos del mismo"* [19].

Hasta el momento no existe una definición completamente aceptada o universal para el término "calidad de la energía eléctrica" o "calidad del suministro eléctrico", pero generalmente se utiliza para referirse al estándar de calidad que debe poseer el suministro de corriente alterna en las instalaciones eléctricas, la cual debería tener tensiones equilibradas, sinusoidales y de amplitudes y frecuencias constantes [20].

4.2. Pérdida de la calidad de la energía

La pérdida de la calidad de la energía implica el deterioro de las señales de voltaje y corriente en lo concerniente a la forma de onda, frecuencia e interrupciones que llevan a la disminución o parada de procesos ocasionando averías o daños de equipos del consumidor [20].

4.3. Importancia del estudio de la calidad de la energía

El estudio de la calidad de la energía eléctrica es el paso más importante para identificar y solucionar problemas en el sistema de potencia. Los problemas eléctricos pueden influir negativamente en el comportamiento del equipo y disminuir su confiabilidad, productividad y rentabilidad e inclusive puede poner en riesgo la seguridad del personal si no son corregidos. Este tipo de estudios para plantas industriales, empresas de energía y empresas privadas, incluyendo auditorías energéticas y revisiones mecánicas, térmicas y eléctricas conducen a reducir los desperdicios de energía y administrar eficientemente los recursos

energéticos [21].

La principal razón por la que hay un gran interés en los estudios de calidad de la energía es de tipo monetario. El número de cargas sensibles a los problemas de suministro de energía eléctrica se ha incrementado en estos últimos años. Esto no solo afecta a consumidores domésticos o comerciales, sino también a los consumidores industriales y a las mismas empresas que suministran y distribuyen la energía eléctrica que utilizan, en la mayoría de los casos, los avances de la electrónica de potencia [22].

4.4. Problemas que influyen en una buena calidad de la energía

Cualquier alteración en el voltaje de una fuente de energía se puede considerar como materia referente a la calidad de la energía eléctrica. Los problemas de la calidad de la energía pueden ser sucesos de mucha velocidad como impulsos de voltaje/transitorios, sonido de alta frecuencia, fallas en la onda eléctrica, incrementos y caídas de voltaje y pérdida total de la energía. Los equipos se verán afectados de forma particular debido a los diversos eventos de la calidad de la energía [21].

La gran parte de los problemas referentes a la calidad de la energía se encuentran relacionados con problemas internos de las edificaciones y no con el suministro eléctrico como tal, así como que el 90% de los problemas en la calidad de la energía suceden dentro de las edificaciones; problemas de puesta a tierra, violación de normas y generación de disturbios en la energía eléctrica interna son algunos casos típicos. Por lo que se considera que existen dos tipos básicos de problemas en la calidad de la energía [21]:

- Los que crean la interrupción de cargas eléctricas o de circuitos enteros.
- Los que causan la interacción del equipo eléctrico y el sistema de suministro eléctrico.

4.5. Tipos de cargas

4.5.1. Cargas lineales

Cuando un voltaje sinusoidal es aplicado directamente a cargas tales como resistencias, inductancias, capacitores o una combinación de ellos, se produce una corriente proporcional a la magnitud del voltaje que también es sinusoidal, por lo que se les denominan cargas lineales (Figura 4.1) [23].

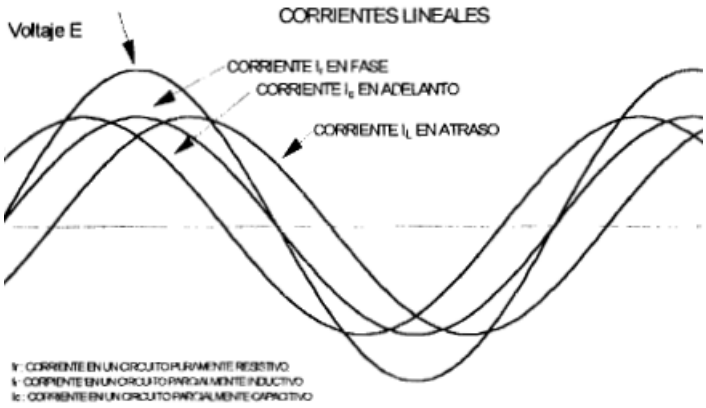


Figura 4.1: Corrientes y voltajes en cargas lineales. *Fuente: Juan Carlos Herrera Heredia, "Determinación de la potencia de transformadores para alimentar cargas no lineales" (1997). Escuela Politécnica Nacional.*

4.5.2. Cargas no lineales

Una carga no lineal es una en la cual la corriente de carga no es proporcional al voltaje instantáneo (Figura 4.2). Muchas veces, la corriente de carga no es continua. Puede ser conmutada en solo una parte del ciclo, tal como en un circuito de tiristores; o la corriente puede ser pulsada, como en un circuito rectificador controlado, un computador, o derivada hacia un UPS. El más grande efecto de las cargas no lineales es el crear una considerable distorsión armónica en el sistema y, además, muchas poseen un bajo factor de potencia, incrementando el costo de utilización de la energía cuando se llega a un factor de potencia penalizado [24].

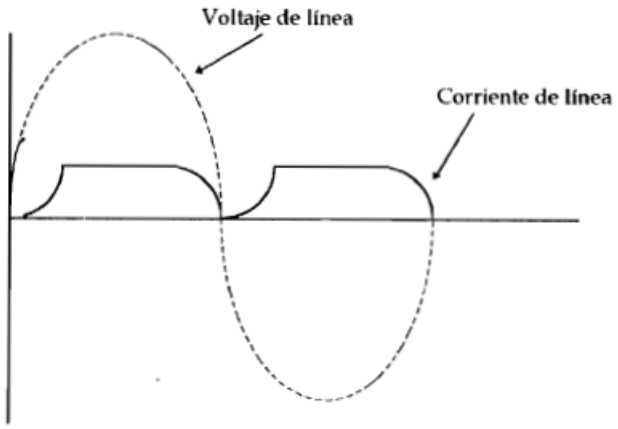


Figura 4.2: Forma de corriente en una carga no lineal. *Fuente: Juan Carlos Herrera Heredia, "Determinación de la potencia de transformadores para alimentar cargas no lineales" (1997). Escuela Politécnica Nacional.*

Las corrientes de carga no lineales son no sinusoidales, y aún cuando la fuente de voltaje sea una onda sinusoidal perfecta, las cargas no lineales distorsionarán esta onda de voltaje haciéndola no sinusoidal [24].

4.5.3. Cargas sensibles

Una carga sensible es aquella que necesita de un suministro de alta calidad, es decir, libre de disturbios [18]. Un ejemplo son los equipos electrónicos.

4.5.4. Cargas críticas

Las cargas críticas son aquellas de cuyo funcionamiento incorrecto o inapropiado se pueden derivar grandes perjuicios económicos o poner en peligro la seguridad del personal [25].

4.6. Eventos que afectan la calidad de la energía

A continuación se expondrán las definiciones de cada uno de los eventos de la calidad de la energía que se establecen en la norma IEEE de 1159 de 1995 [26]:

4.6.1. Variaciones de larga duración

Estas comprenden desviaciones del valor RMS a frecuencia de potencia (60 Hz) por un tiempo superior a 1 minuto. Las variaciones de larga duración pueden ser sobrevoltajes (overvoltages) o bajos voltajes (undervoltages), y usualmente son producidas por variaciones en la carga o por acciones de cierre/apertura de interruptores, es decir, acciones de switcheo.

4.6.2. Variaciones de corta duración

Estas variaciones son originadas por condiciones de fallas, comúnmente cortocircuitos en el sistema. Dependiendo de la duración de la variación según lo visto en la Tabla 5.1, esta se puede clasificar como instantánea, momentánea o temporal.

4.6.3. Transitorios

En ingeniería eléctrica los transitorios hacen referencia a esos eventos subcíclicos indeseables que perturban la forma de onda sinusoidal pura. Estos pueden ser de grandes magnitudes en pocos milisegundos (impulsivos) o variaciones rápidas a frecuencias mayores a las del sistema de potencia (oscilatorios). Ambos tipos de transitorios pueden ser medidos con o sin la componente a frecuencia fundamental incluida.

4.6.3.1. Transitorio impulsivo

Es un disturbio en el voltaje de alimentación que no dura más de medio ciclo y que al comienzo posee la misma polaridad que el voltaje normal, de tal forma que el disturbio se suma a la forma de onda nominal (Figura 4.3). Estos transitorios son provocados por maniobras con interruptores y por descargas atmosféricas, por lo cual también son llamados impulsos atmosféricos.

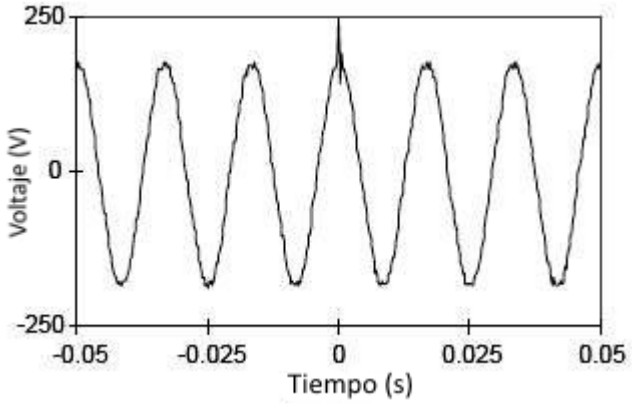


Figura 4.3: Impulso transitorio de voltaje. *Fuente: [26]*

4.6.3.2. Transitorio oscilatorio

Es una variación repentina a frecuencia diferente de la de potencia (60 Hz) en la condición de estado estacionario de voltaje, la corriente o de ambos, que implica tanto valores de polaridad positiva como negativa (Figura 4.4). Un transitorio oscilatorio consiste de un voltaje o corriente cuyo valor instantáneo varía de polaridad rápidamente y se describe por su contenido espectral (frecuencia predominante), magnitud y duración. Las subclases de contenido espectral son alta, baja y media frecuencia (ver Tabla 5.1).

4.6.4. Muesca

Una muesca es un disturbio transitorio en el voltaje de alimentación con duración menor a medio ciclo y que, inicialmente, posee una polaridad opuesta al voltaje normal, de tal modo que el disturbio se resta a la forma de onda (Figura 4.5).

Estas son originadas por cortocircuitos entre fases por la conmutación de diodos y de tiristores (SCR) en rectificadores trifásicos.

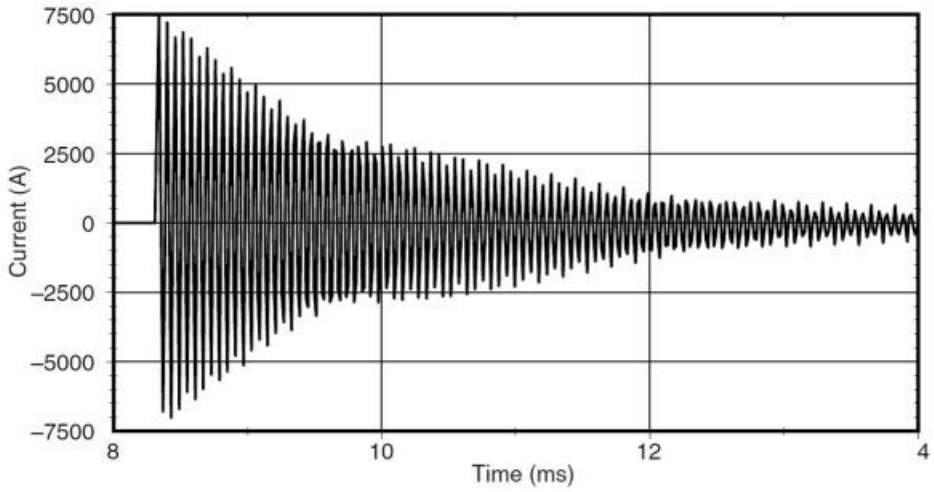


Figura 4.4: Corriente transitoria oscilatoria causada por la maniobra de un banco de capacitores. *Fuente: Ali KÖSE, "Corrección del coeficiente de energía con reactor controlado por tiristor de condensador fijo" (2014). Gazi Üniversitesi.*

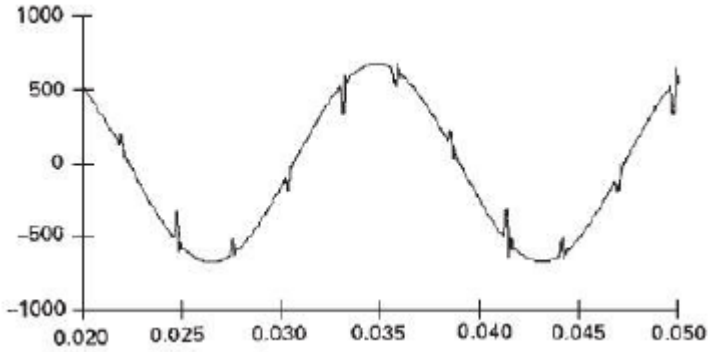


Figura 4.5: Muescas (notches) en el voltaje. *Fuente: [26].*

4.6.5. Reducciones y aumentos temporales en el voltaje

4.6.5.1. Depresión (*Sag o dip*)

Se define como la reducción momentánea del valor RMS del voltaje o corriente de CA (a valores entre 0.1 pu y 0.9 pu) a la frecuencia de potencia (60 Hz) con duración entre medio ciclo (8.33 ms) y un minuto (Figura 4.6).

Este evento puede ocasionar que el voltaje caiga por debajo del nivel estándar por varios ciclos y hacer que las cargas críticas salgan de operación. Además,

para equipos controlados electrónicamente, un voltaje por debajo del 20% del valor normal resultará en salida de operación.

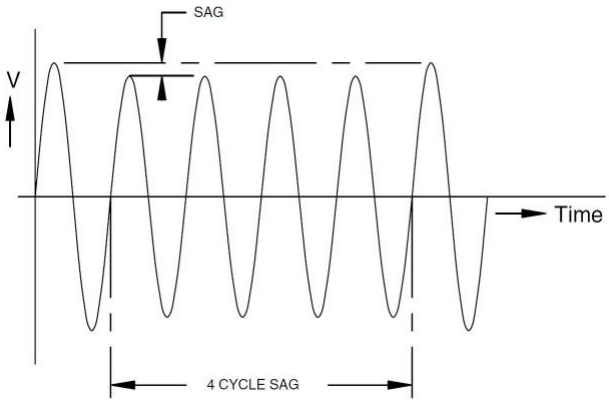


Figura 4.6: Depresión temporal del voltaje. *Fuente: [26].*

4.6.5.2. Elevación (*Swell*)

Se refiere al incremento momentáneo del valor RMS del voltaje o de la corriente de CA (entre 1.1 pu y 1.8 pu) a la frecuencia de potencia con duración entre medio ciclo (8.33 ms) y 1 minuto (Figura 4.7).

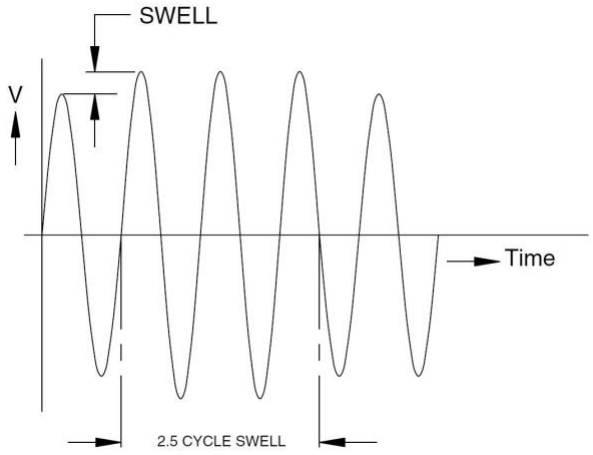


Figura 4.7: Elevación temporal del voltaje. *Fuente: [26]*

4.6.5.3. Bajo voltaje (*undervoltage*)

Es la disminución del valor RMS del voltaje de alimentación a menos del 90% del voltaje nominal de CA a frecuencia de potencia (60 Hz) que dura más de un minuto. La diferencia entre *undervoltage* y *sag* solo radica en el tiempo de duración.

4.6.5.4. Sobrevoltaje (*overvoltaje*)

Es el aumento en el valor RMS del voltaje de alimentación (superior al 110% del voltaje nominal) de CA a frecuencia de potencia con duración de más de un minuto. La diferencia entre *overvoltage* y *swell* solo radica en el tiempo de duración.

Los sobrevoltajes y bajos voltajes son variaciones de larga duración y usualmente no resultan de fallas en el sistema, sino más bien por variaciones en la carga y por operaciones de conmutación (switcheo) en el sistema.

4.6.6. Interrupción

La interrupción tiene lugar cuando el voltaje de alimentación o la corriente de carga se reducen a menos de 0.1 pu (10% del voltaje nominal) (Figura 4.8). Estas son medidas por su duración y se clasifican en: instantáneas (entre 0.5 ciclos y 30 ciclos), momentáneas (entre 30 ciclos y 3 segundos) o sostenidas (superiores a 1 minuto).

Las interrupciones pueden ser causados por fallas en el sistema de potencia, mal funcionamiento de los controles o fallas en el equipo.

4.6.7. Fluctuaciones de voltaje, parpadeo o *"Flicker"*

Según la IEC, *"las fluctuaciones de voltaje son impresiones de inestabilidad de la sensación visual inducida por una fuente luminosa, cuya distribución del contenido espectral fluctúa con el tiempo dentro de unos límites preestablecidos"* [27].

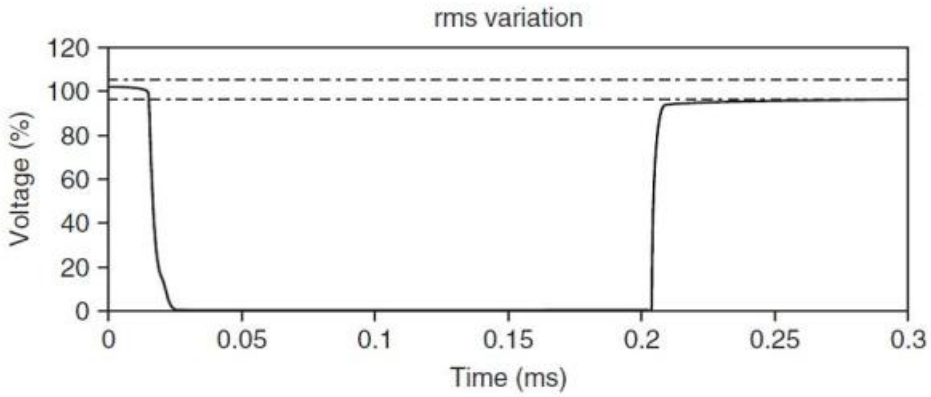


Figura 4.8: Interrupción del suministro eléctrico. *Fuente: Ali KÖSE, "Corrección del coeficiente de energía con reactor controlado por tiristor de condensador fijo" (2014). Gazi Üniversitesi.*

El parpadeo o *flicker* es el efecto más notorio de las variaciones de voltaje. Es un evento de baja frecuencia en el que la magnitud del voltaje presenta variaciones que llegan a ser observables al ojo humano. De manera general, las fluctuaciones oscilan entre el 0.1% y el 7% de la tensión nominal con un contenido espectral típico inferior a 25 Hz intermitente. El parpadeo se debe, por lo general, a la energización de cargas que requieren de corrientes en el arranque. Ejemplos de estas cargas lo son elevadores, soldadoras y hornos de arco.

Las fluctuaciones del voltaje son cambios sistemáticos de la envolvente del voltaje o una serie de variaciones aleatorias en el voltaje cuya magnitud no excede unos límites preestablecidos (Figura 4.9).

4.6.8. Desbalance de tensión

Corresponde al desequilibrio entre fases en un sistema polifásico. El desequilibrio se puede dar en magnitud o en desfase.

La fuente principal de desbalances menores al 2% en el voltaje, es tener cargas monofásicas en un circuito trifásico, o también, puede ser el resultado de que se fundan los fusibles en una fase de bancos de capacitores trifásicos.

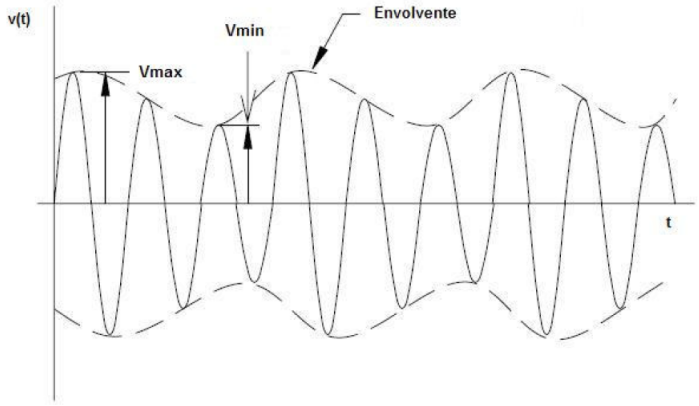


Figura 4.9: Voltaje de alimentación típico en un horno de arco indicando la fluctuación de voltaje a una frecuencia *flicker* de 3 Hz. *Fuente: [26].*

4.6.9. Distorsión de la forma de onda

Es definida como una distorsión en estado estacionario de una sinusoide ideal de frecuencia de potencia que se caracteriza principalmente por el contenido espectral de la desviación. Existen cinco tipos:

- Offset de CD (Figura 4.10)
- Armónicas (Figura 4.12)
- Interamónicas
- Muescas (Figura 4.5)
- Ruido (Figura 4.11)

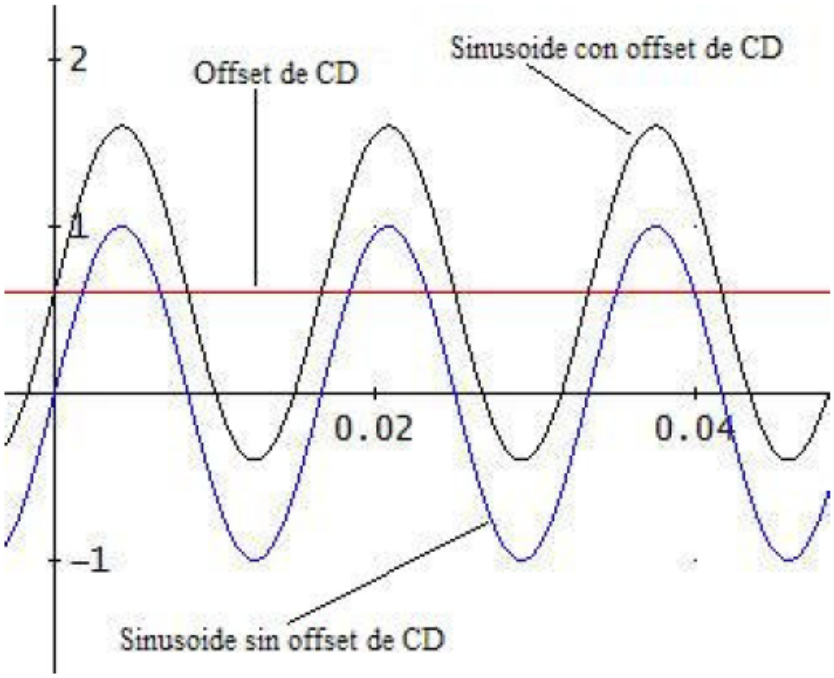
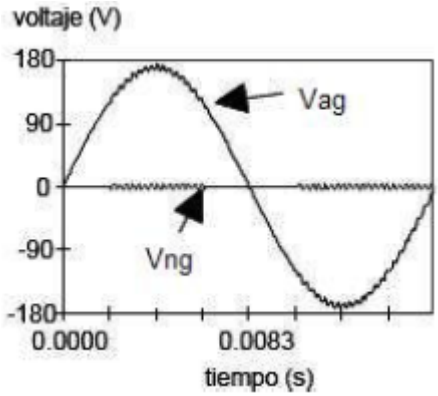
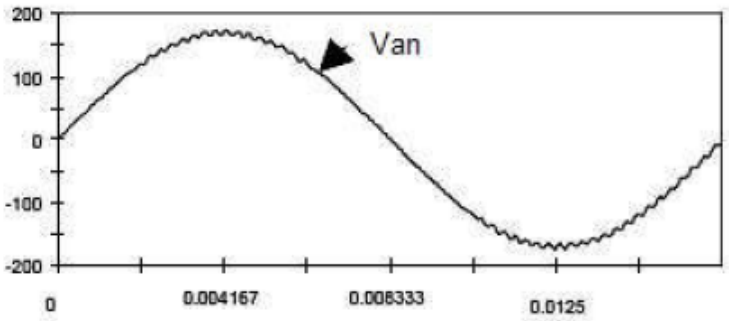


Figura 4.10: Onda sinusoidal con offset de CD. *Fuente: [26].*



(a) Ruido de modo común



(b) Ruido de modo normal.

Figura 4.11: Ondas de voltaje con ruido. *Fuente: [26].*

4.6.10. Distorsión armónica

Se define como la representación cuantitativa de la distorsión a partir de una forma de onda sinusoidal pura. La distorsión armónica es debida a cargas no lineales, o a cargas en las que la forma de onda de la corriente no conforma a la forma de onda del voltaje de alimentación (Figura 4.12).

Los voltajes o corrientes que tienen componentes de frecuencia que no son múltiplos enteros de la frecuencia fundamental son llamados **interarmónicos**. Estos se pueden encontrar en redes de cualquier nivel de tensión, siendo las fuentes principales de distorsión de forma de onda interarmónica los convertidores de frecuencia estáticos, los cicloconvertidores, los hornos de inducción y los dispositivos de arco.

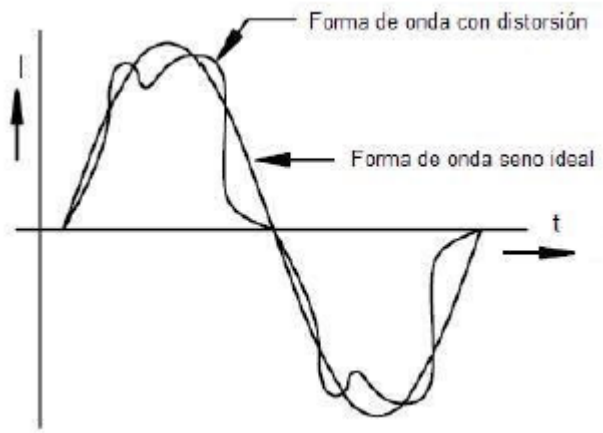


Figura 4.12: Forma de onda con distorsión armónica. *Fuente: [26].*

4.6.11. Variación de la frecuencia

Hace referencia a la desviación de la frecuencia fundamental del sistema de potencia de su valor nominal especificado (idealmente de 50 Hz o 60 Hz) (Figura 4.13).

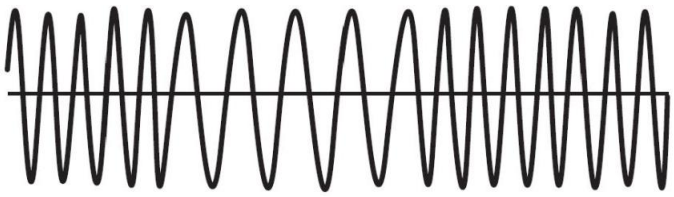


Figura 4.13: Variación de frecuencia. *Fuente: [26].*

4.7. Convertidor multinivel en cascada

En la Figura 4.14 se muestra el diagrama general de un inversor multinivel en cascada con puente-H *(H-bridge Cascaded Multilevel Inverter, CMLI)*. La forma de onda de salida vendría a ser la suma de las salidas de cada puente-H [28].

La topología de este convertidor puede dividirse en dos categorías que dependen de la relación de los voltajes en cada puente. Estas categorías son: simétricas y asimétricas [29].

La topología es simétrica cuando los voltajes en todos los puentes son iguales, mientras que la topología es asimétrica cuando los voltajes son diferentes, en este tipo son utilizados comúnmente relaciones de transformación de 1:2 o 1:3 [29].

Dependiendo de la manera de obtener el voltaje de cada puente, se pueden describir dos sub-topologías: una *fuente de CD separada*, en la que todos los puentes reciben alimentación de diferentes fuentes de voltaje (Figura 4.14) y *una sola fuente de CD*, en la que todos los puentes se encuentran alimentados desde la misma fuente de voltaje. La diferencia en los voltajes y el aislamiento eléctrico se logra por medio del uso de transformadores (Figura 4.15) [29].

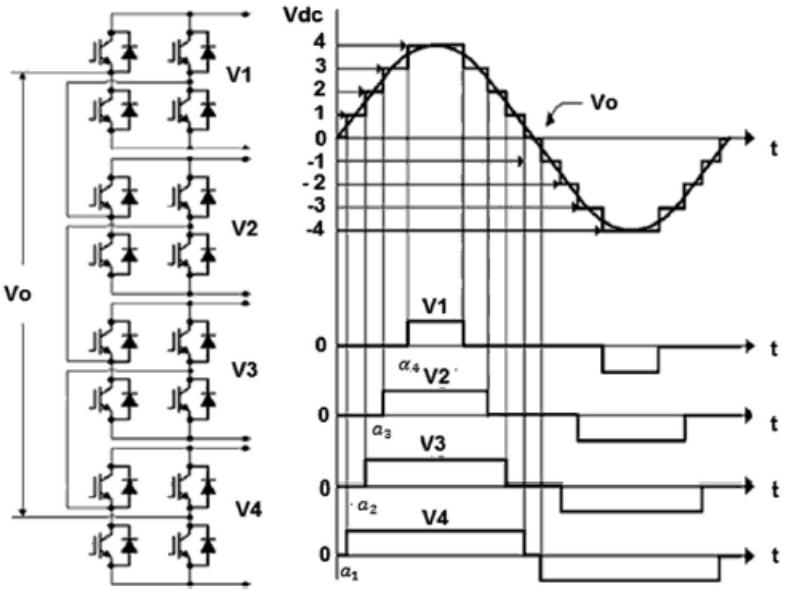


Figura 4.14: Convertidor multinivel puente-H, formas de onda y salida del convertidor. *Fuente: [30].*

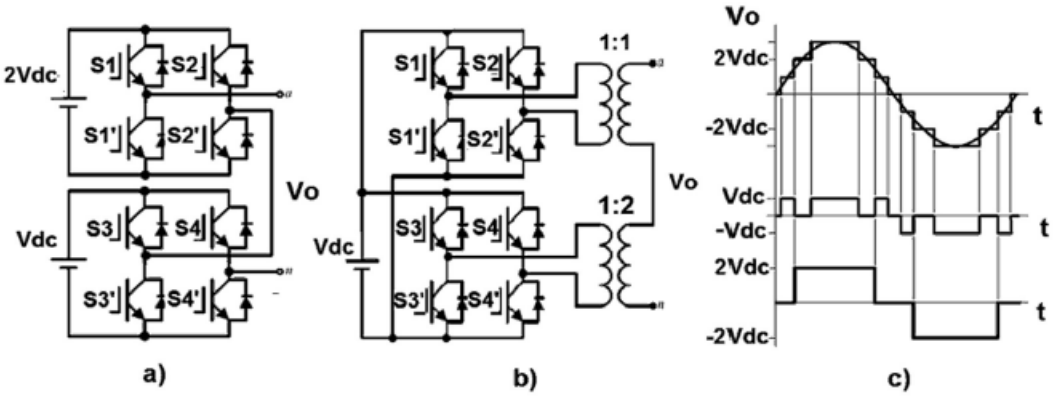


Figura 4.15: Sub-topologías asimétricas de un convertidor multinivel en cascada tipo puente-H. (a) Fuente de DC separada, (b) Una sola fuente de DC, (c) Voltaje de salida. *Fuente: [30].*

Quinto Capítulo

Determinación de los eventos

Para la elección de los eventos de calidad de la energía que se pueden escoger en el simulador, se tuvo en cuenta la clasificación dada por el estándar IEEE 1159 - 1995 (ver Tabla 5.1).

Tabla 5.1: CategorÍas y caracterÍsticas de fenómenos electromagnéticos (IEEE 1159-1995). *Fuente: Adaptado de [26].*

CategorÍa	**Contenido TÍpico Espectral**	**Duración TÍpica**	**Magnitud TÍpica del Voltaje**
1. Transitorios			
1.1. Impulsivos			
1.1.1. Nanosegundos	5 ns de elevación	<50 ns	
1.1.2. Microsegundos	1 μs de elevación	50 ns - 1 μs	
1.1.3. Milisegundos	0.1 ms de elevación	>1 ms	
1.2. Oscilatorios			
1.2.1. Baja frecuencia	<5 kHz	0.3 - 50 ms	0 - 4 pu
1.2.2. Media frecuencia	5 - 500 kHz	20 μs	0 - 8 pu
1.2.3. Alta frecuencia	0.5 - 5 MHz	5 μs	0 - 4 pu
2. Variaciones de corta duración			
2.1. Instantáneas			
2.1.1. Sag		0.5 - 30 ciclos	0.1 - 0.9 pu

Continúa en la siguiente página.

Categoría	Contenido Típico Espectral	Duración Típica	Magnitud Típica del Voltaje
2.1.2. Swell		0.5 - 30 ciclos	1.1 - 1.8 pu
2.2. Momentáneas			
2.2.1. Interrupción		0.5 - 3s	<0.1
2.2.2. Sag		30 ciclos-3s	0.1 - 0.9 pu
2.2.3. Swell		30 ciclos-3s	1.1 - 1.4 pu
2.3. Temporal			
2.3.1. Interrupción		3s - 1 min	<0.1 pu
2.3.2. Sag		3s - 1 min	0.1 - 0.9 pu
2.3.3. Swell		3s - 1 min	1.1 - 1.2 pu
3. Variaciones de larga duración			
3.1. Insterrup. sostenida		>1 min	0.0 pu
3.2. Bajo voltaje		>1 min	0.8 - 0.9 pu
3.3. Sobrevoltaje		>1 min	1.1 - 1.2 pu
4. Desbalance de voltaje		Est. Estable	0.5 - 2%
5. Distorsión de forma de onda			
5.1. Componente de CD		Est. Estable	0 - 0.1%
5.2. Contenido armónico	De la armónica 0 a la 100	Est. Estable	0 - 20%
5.3. Interamónicas	0 - 6 kHz	Est. Estable	0 - 2%
5.4. Muescas en el voltaje		Est. Estable	
5.5. Ruido banda amplia		Est. Estable	0 - 1%
6. Fluctuaciones de voltaje	<25 Hz	Intermitente	0.1 - 7%

Continúa en la siguiente página.

Categoría	Contenido Típico Espectral	Duración Típica	Magnitud Típica del Voltaje
7. Variaciones de frecuencia		<10s	

En la Tabla 5.2 se pueden observar los eventos de calidad de la energía que se incluyen en el simulador desarrollado en el presente trabajo.

Tabla 5.2: Eventos electromagnéticos a simular. *Fuente: Elaboración propia.*

Eventos	Contenido Típico Espectral	Duración Típica	Magnitud Típica del Voltaje
Elevación *(Swell)*		0.5 - 1 min	0.1 - 0.9 pu
Depresión *(Sag,dip)*		0.5 - 1 min	1.1 - 1.8 pu
Bajo voltaje		>1 min	0.8 - 0.9 pu
Sobrevoltaje		>1 min	1.1 - 1.2 pu
Interrupción instantánea		0.5 ciclos - 30 ciclos	<0.1 pu
Interrupción momentánea		30 ciclos - 3 s	<0.1 pu
Interrupción temporal		3 s - 1 min	<0.1 pu
Interrupción sostenida		>1 min	<0.1 pu
Variación de frecuencia		<10s	
Armónicos	h = 0-50	Est. Estable	0 - 20%

Los anteriores eventos son en su mayoría los que se encuentran relacionados con magnitudes media cuadráticas *(Root-Mean Square - RMS)* y de duración de tiempo como lo son la depresión de voltaje *(sag)*, la elevación de voltaje *(swell)*, entre otros; y también los eventos de variaciones de frecuencia y contenido armónicos.

Para el simulador de eventos de calidad de la energía, se excluyen los eventos o fenómenos transitorios, tanto impulsivos como oscilatorios, lo anterior debido

a la dificultad por la parte del muestreo para la determinación de fenómenos oscilatorios de media y alta frecuencia [30]. Al igual que se dejan por fuera los eventos como el ruido, muescas en el voltaje y *offset* de CD, entre otros. Todo lo anterior dada la dificultad de encontrar una expresión o formulación de una ecuación que dé inicio a la realización de un algoritmo.

Cabe resaltar que los métodos propuestos para simular cada uno de los eventos presentes en la Tabla 5.2 no llegan a abarcar el problema para su validación en línea.

Sexto Capítulo

Métodos para la generación de los eventos de calidad de la energía

6.1. Algoritmo metaheurístico PSO *(Particle Swarm Optimization)*

El método utilizado como estrategia para determinar los ángulos de disparos necesarios para la creación de los eventos que involucran magnitudes media cuadráticas *(Root-Mean Square - RMS)* y duración de tiempo, para unos parámetros y condiciones específicas dadas por el usuario, fue con la implementación de un algoritmo de optimización por enjambre de partículas PSO *(Particle Swarm Optimization)*.

El algoritmo PSO pertenece a las técnicas denominadas *optimización inteligente* y se clasifica como un algoritmo estocástico de optimización basado en población. A esta clasificación igualmente pertenecen los Algoritmos Genéticos (AG). Los anteriores algoritmos se consideran adecuados comparados con los métodos clásicos de optimización cuando el problema de optimización es complejo, estocástico o no lineal con múltiples mínimos locales [31].

Una de las ventajas atribuidas a estas técnicas inteligentes de optimización son: su paralelismo intrínseco, su capacidad para resolver problemas complejos, de gran tamaño, y con un mínimo conocimiento del sistema que se está identificando [31].

Este algoritmo de búsqueda basado en población fue propuesto por *Kennedy*

y *Eberhart* en 1995 [32] como un modelo de las actividades sociales de insectos, pájaros y peces.

Lo que pretende el algoritmo PSO es representar el proceso natural de comunicación grupal para compartir conocimiento individual cuando grupos de animales se desplazan, migran o cazan. Si un miembro detecta un camino deseable para desplazarse, el resto de la colonia lo sigue inmediatamente. En PSO, este comportamiento animal es imitado por partículas con ciertas posiciones y velocidades en un espacio de búsqueda, donde la población es llamada *swarm*, y cada miembro del *swarm* es llamado partícula. La población inicial se determina aleatoriamente y cada partícula se desplaza a través del espacio de búsqueda y recuerda la mejor posición que ha encontrado. Cada partícula comunica las buenas posiciones a las demás y dinámicamente ajustan su propia posición y su velocidad con base en las buenas posiciones. La velocidad se ajusta con el comportamiento histórico de las partículas. De esta forma, las partículas tienden a dirigirse hacia un mejor espacio de búsqueda en el proceso de minimización de la función objetivo, también llamada función de costo [33].

El procedimiento de implementación del algoritmo PSO se ilustra en el diagrama de flujo mostrado en la Figura 6.1. El proceso de optimización puede ser dividido en seis pasos generales como se describe a continuación [31]:

1. *Inicialización:* Durante este paso se determinan los límites de la posición y velocidad de las partículas; la población inicial (la cual se calcula aleatoriamente) y el valor $pbest_i$ correspondiente; los parámetros de necesarios para el procedimiento de búsqueda y la condición de parada del algoritmo.

2. *Evaluación de la población inicial:* En este paso, el costo para todas las partículas en la población inicial es evaluado de acuerdo con la función objetivo; y se selecciona la mejor partícula global, *gbest*.

3. *Actualización de posición y velocidad:* La posición y velocidad se actualizan de acuerdo con una ecuación de razón de cambio de la posición de cada partícula;

si la posición y la velocidad de las partículas están por fuera de los correspondientes límites, éstos se ajustan a los valores establecidos.

4. *Evaluación de la población actualizada:* Similar al paso 2, la posición actualizada de las partículas es evaluada de acuerdo al valor de la función objetivo; las partículas *gbest* y la *pbest* serán actualizadas si es necesario.

5. *Verificación del cumplimiento de la condición de parada:* Esta condición puede corresponder con el número de iteraciones o con el mínimo valor de la función objetivo. Si la condición de parada no ha sido cumplida, el proceso de actualización del paso 3 será repetido; de lo contrario, el proceso de optimización finalizará.

6. *Resultados de salida:* La mejor solución obtenida durante el proceso de optimización, *gbest*, es la salida en este paso, que corresponderían a los ángulos de disparo α.

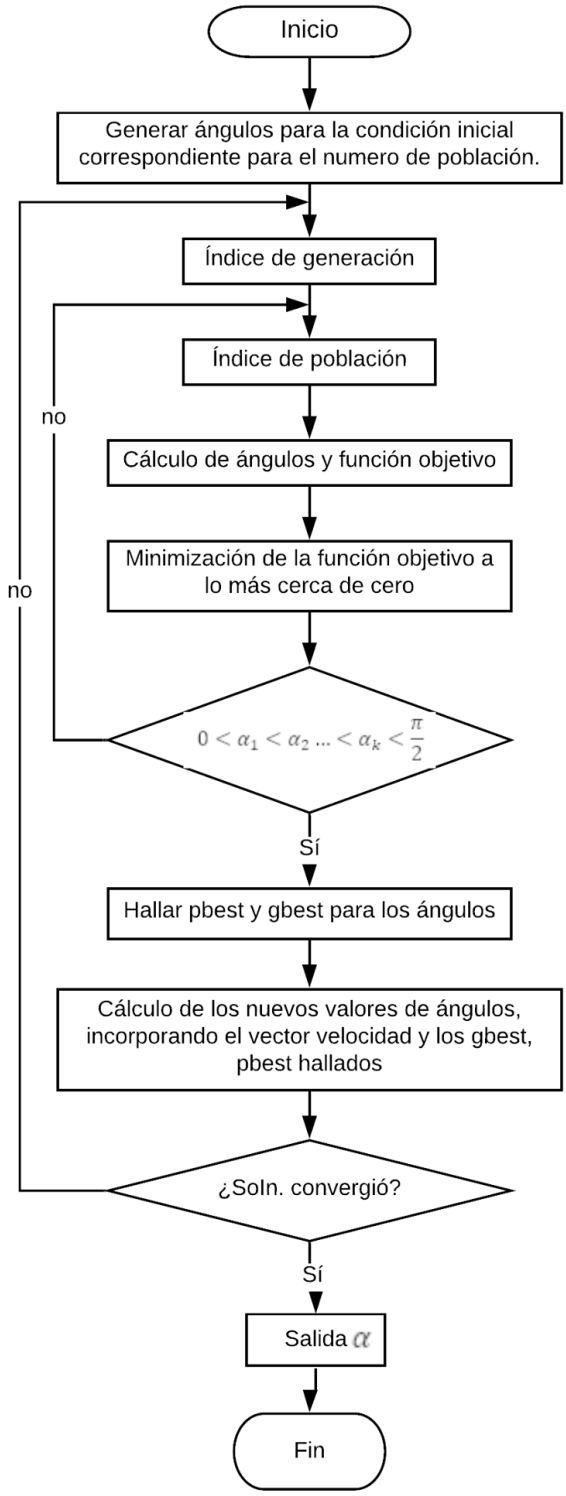


Figura 6.1: Diagrama de flujo del algoritmo PSO. *Fuente: Adaptado de [33].*

6.2. Método para la generación de *sag*, *swell*, bajovoltaje y sobrevoltaje

En primera instancia, para estos eventos que involucran magnitudes media cuadráticas *(Root-Mean Square - RMS)* y de duración de tiempo, se hace uso de la ecuación 6.1 del voltaje RMS para modulación PWM, la cual se extrajo de [34]:

$$VRMS = \sqrt{\sum_{n=1}^{hmax} \frac{1}{2} \frac{4Vdc}{\pi n} \left(\sum_{i=1}^{k-1/2} \sum_{j=1}^{Li} (-1^{j-1} \cos n\alpha_{ij}) \right)^2} \tag{6.1}$$

Donde:

V_{RMS}: Valor del voltaje RMS.

Vdc: Valor de voltaje del escalón.

n: Número del armónico.

i: Indicador del escalón.

j: Indicador del número de ángulo de disparo en el escalón i.

α_{ij}: Ángulo de encendido o apagado en cada escalón i y ángulo j.

L: Vector que contiene la información de cuantos ángulos hay en cada escalón.

Li: Número de ángulos de disparo para el escalón i en el primer cuarto de onda.

$hmax$: Armónico máximo.

k: Indicador del número de escalones.

La ecuación 6.1 es la clave para obtener los eventos mencionados anteriormente, por tanto, se hizo necesaria la creación de una función en MATLAB que cumplirá el papel de función objetivo para el algoritmo metaheurístico PSO *(Particle Swarm Optimization)*. En este caso, la función necesitará unos valores α que corresponden a los ángulos de disparo de cada escalón del convertidor para calcular el valor de voltaje RMS que desee el usuario y así generar la gráfica del evento que se desea.

El código de la función mencionada se puede encontrar en el Anexo A.

Luego de realizada la anterior función, se procedió a programar y parametrizar el algoritmo PSO, así como la función de costo o función objetivo que se empleará para la determinación de los ángulos α para cada uno de los eventos presentados en esta sección.

En el algoritmo PSO realizado, se toma como salida los ángulos de disparo α, es decir, el resultado que se espera que arroje el algoritmo serán cada uno de estos ángulos. Para ello se deben cumplir las condiciones presentadas a continuación:

$$0 < \alpha_1 < \alpha_2 < ...\alpha_m < \frac{\pi}{2} \tag{6.2}$$

Con la expresión 6.2 se quiere dar a entender que los ángulos resultantes del algoritmo PSO deben estar en un rango de entre 0 y 90°, y deben a su vez ir de forma ascendente continua.

Dado que el propósito de este algoritmo es minimizar o reducir el error de la ecuación presentada como función objetivo, se tomó como ésta la ecuación del error absoluto del V_{RMS} dado por el usuario (valor deseado) menos el V_{RMS} calculado por el algoritmo (valor estimado). Esta ecuación es presentada en 6.3.

$$VRMS_{error} = VRMS_{Usuario} - VRMS_{PSO} \tag{6.3}$$

Donde:

$VRMS_{error}$: Valor del error absoluto, el cual se espera que sea cero.

$VRMS_{Usuario}$: Valor deseado (ingresado por el usuario).

$VRMS_{PSO}$: Valor estimado (el calculado por el algoritmo PSO).

El código de la función objetivo creada se puede encontrar en el Anexo B.

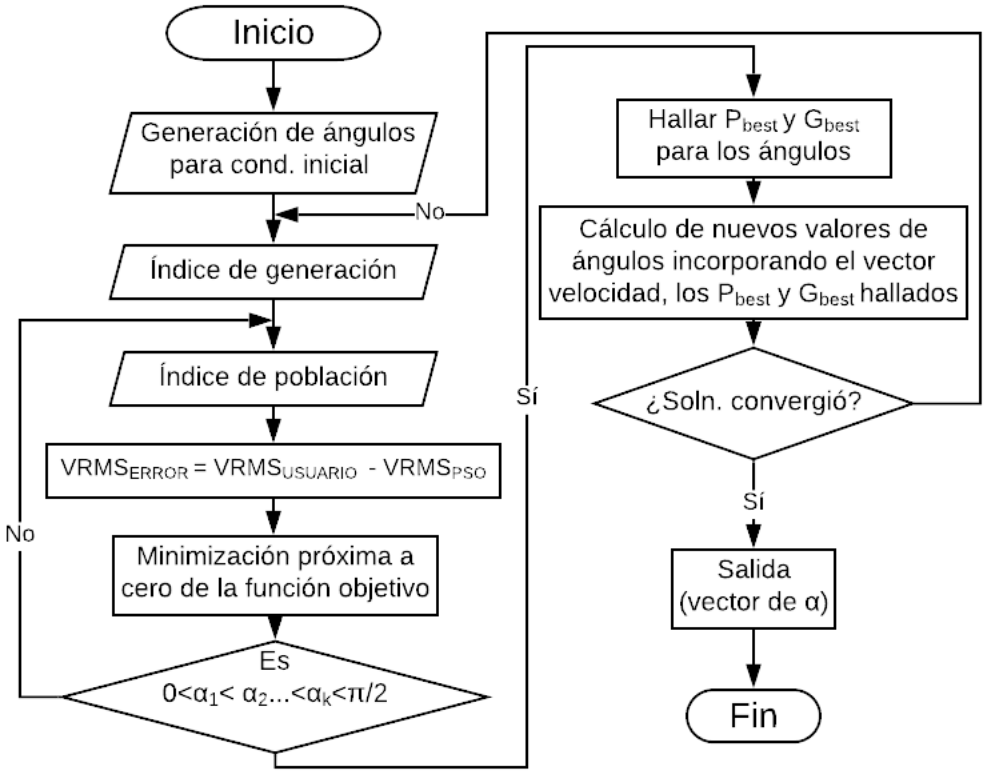


Figura 6.2: Diagrama de flujo del algoritmo de generación de los eventos. *Fuente: Elaboración propia.*

En la Figura 6.2 se muestra el diagrama de flujo del algoritmo para los eventos de depresión de voltaje, elevación, bajo voltaje y sobrevoltaje.

Posteriormente hallados los ángulos de disparo que dan el error más pequeño para el V_{RMS} pedido, se procede a la obtención de los datos en X (tiempo) y en Y (Magnitud) del primer ciclo de la gráfica escalonada del evento y de la onda sin presencia de éste. Lo anterior se realiza a través de una función a la que se le entregan los ángulos α resultantes del algoritmo PSO y se puede encontrar a más detalle en el Anexo C.

Por último se procede a realizar la simulación del evento en la blockset SIMULINK de MATLAB (ver Figura 6.3), en donde se crean en los primeros cinco ciclos una onda sin el evento, luego se introduce la gráfica escalonada del evento reconstruida con la duración digitada por el usuario en la interfaz, y por último veinte ciclos adicionales de la gráfica sin el evento.

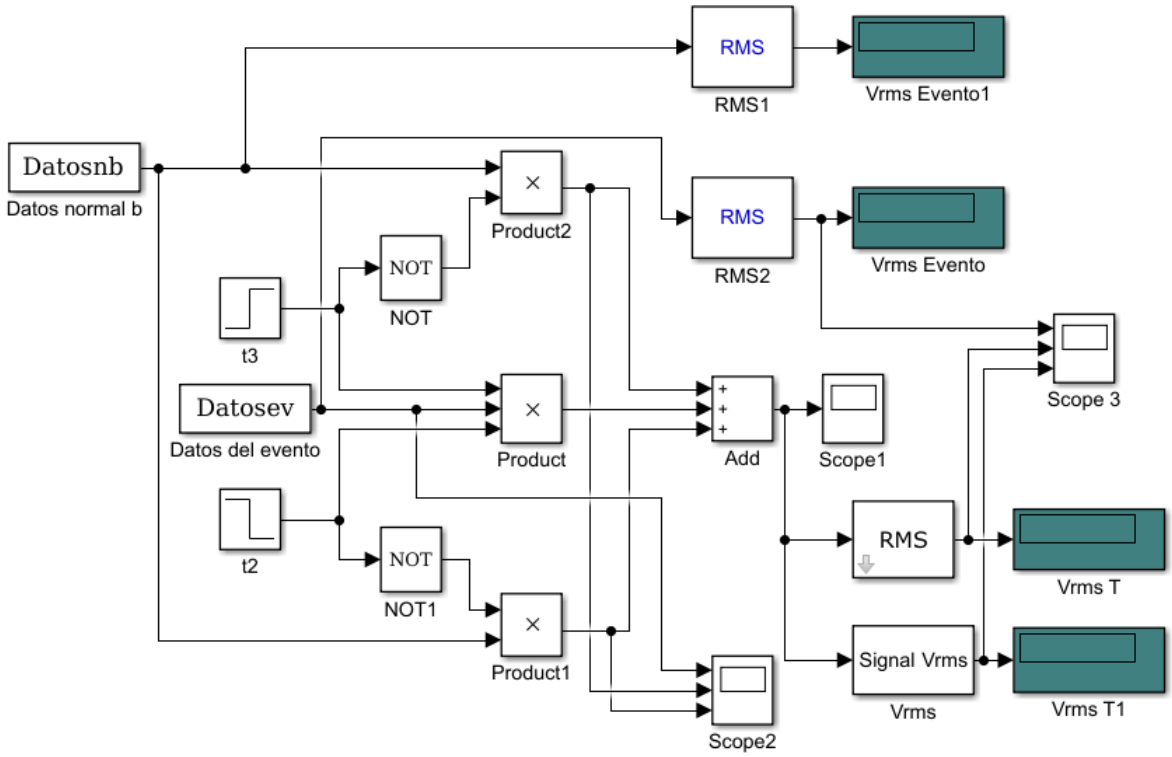


Figura 6.3: Programa en SIMULINK para la creación de la gráfica del evento. *Fuente: Elaboración propia.*

La creación de las gráficas para los eventos simulados sigue el proceso descrito en el diagrama de flujo mostrado en la Figura 6.4. Por un lado se tienen los datos

de la onda sin el evento, y por el otro, los datos del evento; t_2 y t_3 se describen con las ecuaciones 6.4 y 6.5 respectivamente.

$$t_2 = \frac{5}{f} + duración_{evento} \tag{6.4}$$

$$t_3 = \frac{5}{f} \tag{6.5}$$

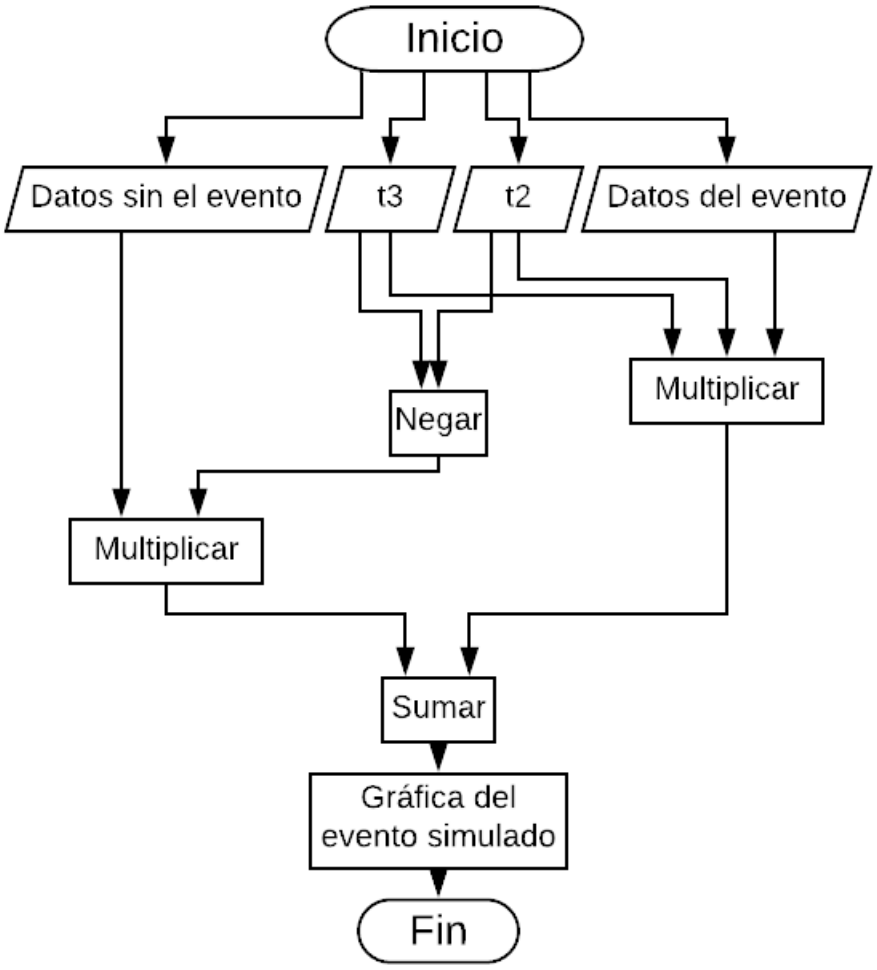


Figura 6.4: Diagrama de flujo del proceso de simulación de la gráfica del evento. *Fuente: Elaboración propia.*

Entonces, se multiplican los datos del evento por t_3 que se encuentra en alto y por t_2 que está en bajo, al hacer esto se logra que el evento se genere a partir de t_3 hasta t_2. Los datos sin el evento se multiplican al inicio por t_3 negado, es decir, que antes de t_3 la onda sin el evento se generará. Además, datos sin el evento se multiplica también por el t_2 negado, es decir que después de t_2 aparecerá de nuevo la onda sin el evento. Y al final, el resultado de cada uno de estos productos se suman para formar de esta manera la gráfica del evento seleccionado.

El procedimiento mencionado anteriormente se realiza de la misma manera para los eventos presentados en esta sección, las diferencias que se encuentran son las siguientes:

- Para la elevación de voltaje *(swell)* los límites de la magnitud corresponderán entre 0.1 - 0.9 pu y la duración que se podrá escoger será entre 0.5 - 1 min.
- Para la depresión de voltaje *(sag)* los límites de la magnitud corresponderán entre 1.1 - 1.8 pu y la duración que se podrá escoger será la misma que la del *swell* al ser ambas variaciones de corta duración.
- Para el bajo voltaje *(undervoltage)* los límites de la magnitud corresponderán entre 0.8 - 0.9 pu y la duración que se podrá escoger será mayor a 1 min.
- Para el sobrevoltaje *(overvoltage)* los límites de la magnitud corresponderán entre 1.1 - 1.2 pu y la duración que se podrá escoger será la misma que la del bajo voltaje al ser ambos variaciones de larga duración.

6.3. Método para la generación de interrupciones

Para el caso de las interrupciones se usa el mismo procedimiento mencionado en la Sección 6.2, salvo que en esta ocasión la magnitud en por unidad (pu) que puede elegir el usuario estará entre 0 y 0.1 pu, y la duración de éstas dependerá del tipo de interrupción que se desee simular: si es instantánea estaría entre

0.5 ciclos a 30 ciclos, si es momentánea estaría entre 30 ciclos a 3 segundos, si es temporal el tiempo estaría en el rango de 3 segundos a 1 minuto, y si es una interrupción sostenida la duración debe ser mayor a 1 minuto.

En la Figura 6.5 se muestra el diagrama de flujo del algoritmo para el evento de los cuatro tipos de interrupciones.

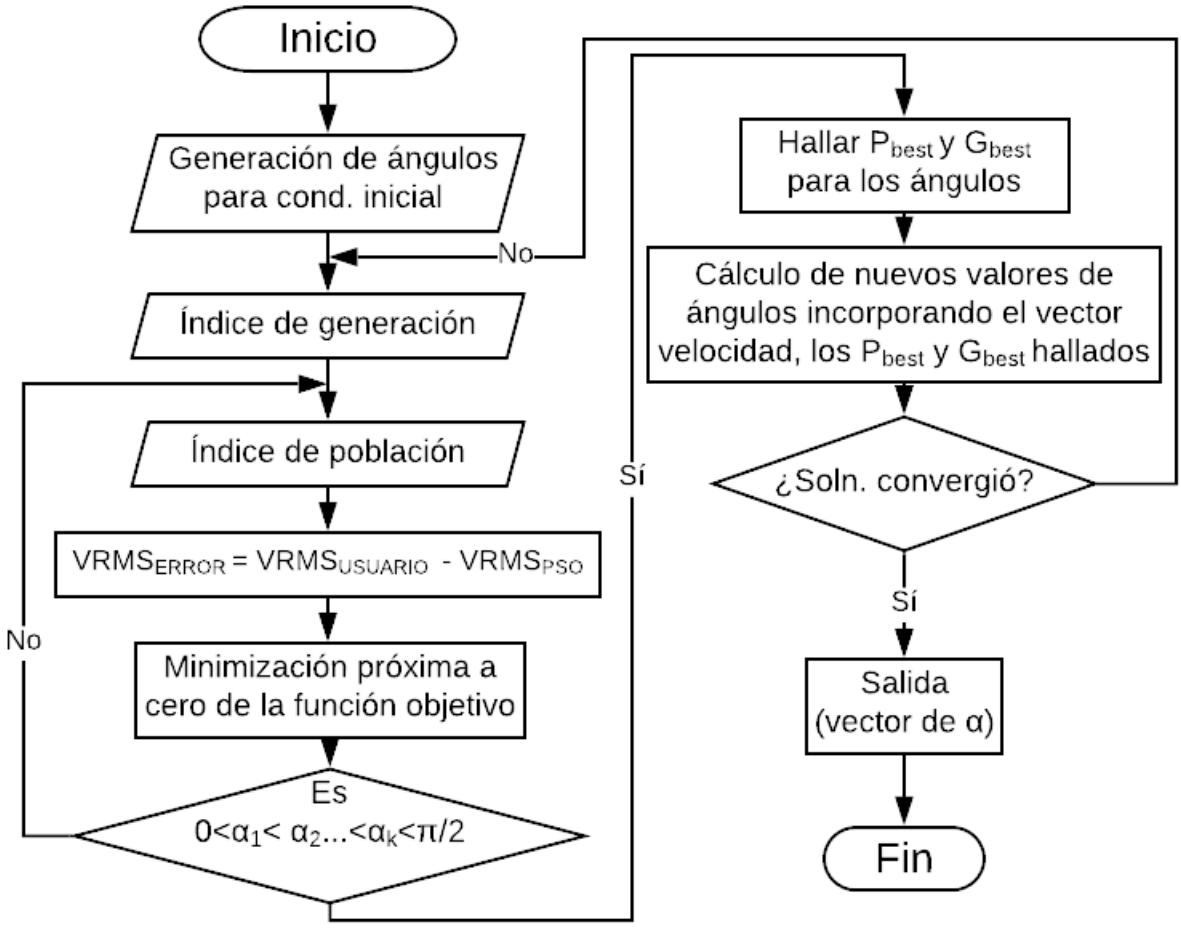


Figura 6.5: Diagrama de flujo del algoritmo de generación de las interrupciones. *Fuente: Elaboración propia.*

6.4. Método para la generación de variaciones de frecuencia

Para este método el usuario deberá registrar el voltaje nominal con el que se desea trabajar, la frecuencia fundamental (50 Hz o 60 Hz), el factor de variación que se desea para la frecuencia y la duración del evento que estará entre 0 a 10 segundos.

Para este caso, el algoritmo PSO tendrá como función objetivo la ecuación (6.1), donde hallará los ángulos de disparo α para la frecuencia fundamental ingresada.

Seguidamente, al momento de la obtención de los datos en X (tiempo) y Y (magnitud) de las gráficas con ayuda de la función del Anexo C, se obtendrá una gráfica sin el evento, la cual en la parte donde se pasan los ángulos a tiempo con la ecuación 6.6 se multiplicarán por la frecuencia fundamental ingresada por el usuario, y una gráfica del evento, la cual en vez de multiplicar por la frecuencia fundamental se multiplicará por la frecuencia a la que se desea llegar (ver ecuación 6.7). Esto se puede ver sintetizado en la Figura 6.6.

$$Vect_tiempo = \frac{Vect_{Ángulos}}{f * 360} \tag{6.6}$$

$$f_{variada} = f_{fund.} * Factordevariación \tag{6.7}$$

El diagrama de flujo del algoritmo para este evento se puede ver en la Figura 6.5.

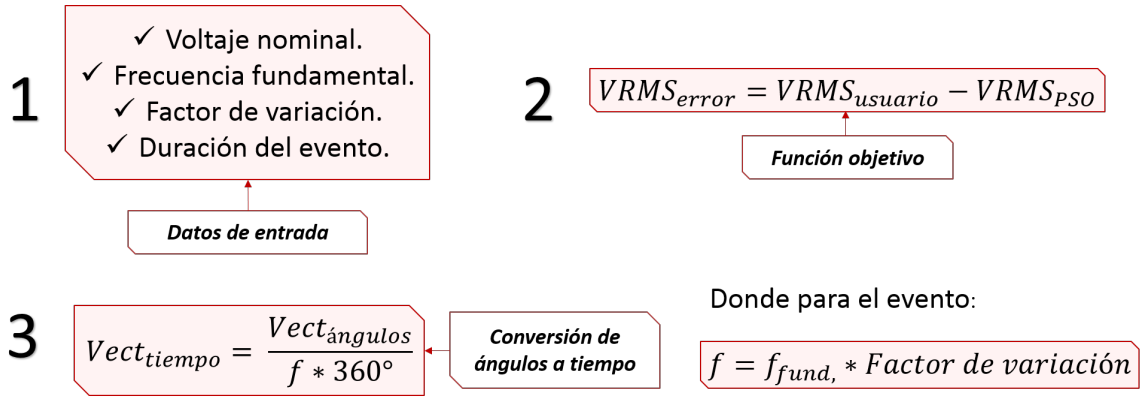


Figura 6.6: Método para la generación de variaciones de frecuencia. *Fuente: Elaboración propia.*

6.5. Método para la generación de armónicos

Para el evento de generación de armónicos se hizo uso de la ecuación **??**, extraída de [34] que, utilizando los valores obtenidos por el algoritmo PSO en las iteraciones previas al resultado final y los datos deseados ingresados por el usuario para cada uno de los armónicos que se quieran incluir hasta máximo el armónico 50, y solo teniendo en cuenta los armónicos impares, determina el error acumulado en todos los armónicos. Este algoritmo obtiene en primera instancia, el valor de la magnitud entregada en el primer armónico por el algoritmo PSO en ese ciclo, posterior a esto compara el valor objetivo, que en este caso sería el porcentaje del armónico en la señal con la relación de la magnitud del armónico n sobre la magnitud del primer armónico calculado por el algoritmo PSO. Este paso se repite entre todos los armónicos establecidos por el usuario y acumula el error para luego entregárselo al algoritmo PSO y este a su vez encuentra los ángulos de disparo α con el mínimo error.

$$b_n = \frac{4Vdc}{\pi n}\left(\sum_{i=1}^{k-1/2}\sum_{j=1}^{Li}(-1^{j-1}\cos n\alpha_{ij})\right)^2 \tag{6.8}$$

Donde:

b_n : Magnitud del armónico n.

Vdc : Valor de voltaje del escalón.

n : Número del armónico.

i : Indicador del escalón.

j : Indicador del número de ángulo de disparo en el escalón i.

α_{ij} : Ángulo de encendido o apagado en cada escalón i y ángulo j.

L : Vector que contiene la información de cuantos ángulos hay en cada escalón.

Li : Número de ángulos de disparo para el escalón i en el primer cuarto de onda.

k : Indicador del número de escalones.

La ecuación presentada en 6.9 es la función objetivo que utiliza el algoritmo de la generación de contenido armónico.

$$bn_{error} = |bn_{Usuario} - \frac{bn_{PSO}}{b1_{PSO}} \times 100| \tag{6.9}$$

Donde:

bn : Magnitud del armónico n.

$bn_{Usuario}$: Magnitud del armónico n ingresada por el usuario.

bn_{PSO} : Magnitud del armónico n calculado por el algoritmo PSO.

$b1_{PSO}$: Magnitud del armónico 1 calculado por el algoritmo PSO.

El diagrama de flujo del algoritmo para este evento se puede ver en la Figura 6.7.

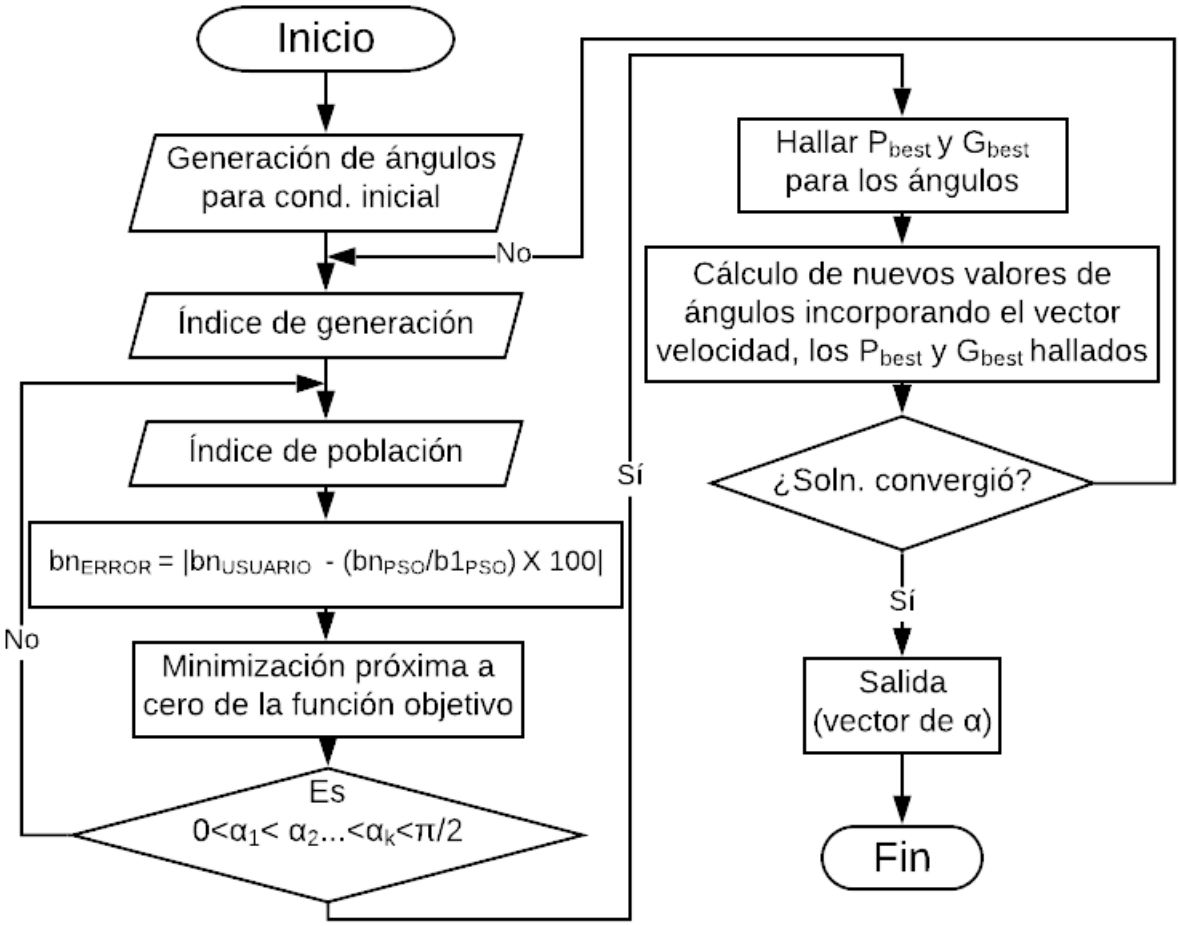


Figura 6.7: Diagrama de flujo para la generación de armónicos. *Fuente: Elaboración propia.*

La función objetivo para los armónicos se puede encontrar en más detalle en el Anexo D.

Séptimo Capítulo

Interfaz gráfica (HMI)

La interfaz gráfica se realizó en GUIDE/MATLAB y se puede observar en la Figura 7.1. Ésta se encuentra programada de tal manera que el panel de "Parámetros del evento" irá variando de manera dinámica según la opción del evento seleccionado por el usuario.

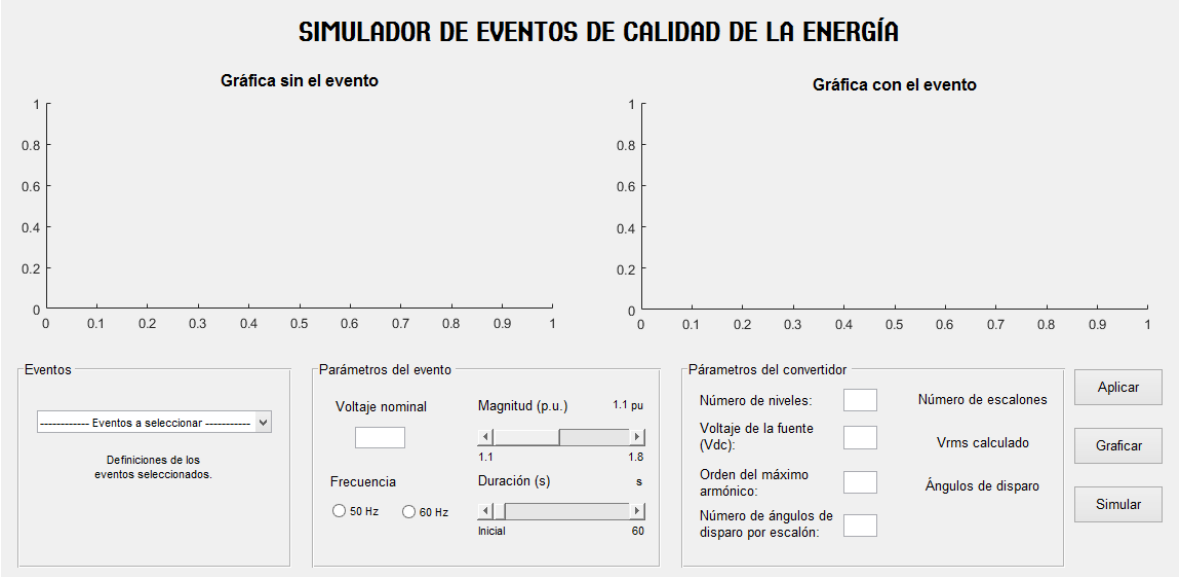


Figura 7.1: Interfaz gráfica del simulador de eventos de la calidad de la energía. *Fuente: Elaboración propia.*

La programación de la interfaz creada se puede encontrar en más detalle en el Anexo D.

7.1. Descripción general

La Figura 7.2 presenta el diagrama de caso de uso a utilizar para el diseño de la interfaz gráfica, el cual se encuentra orientado bajo la metodología UML.

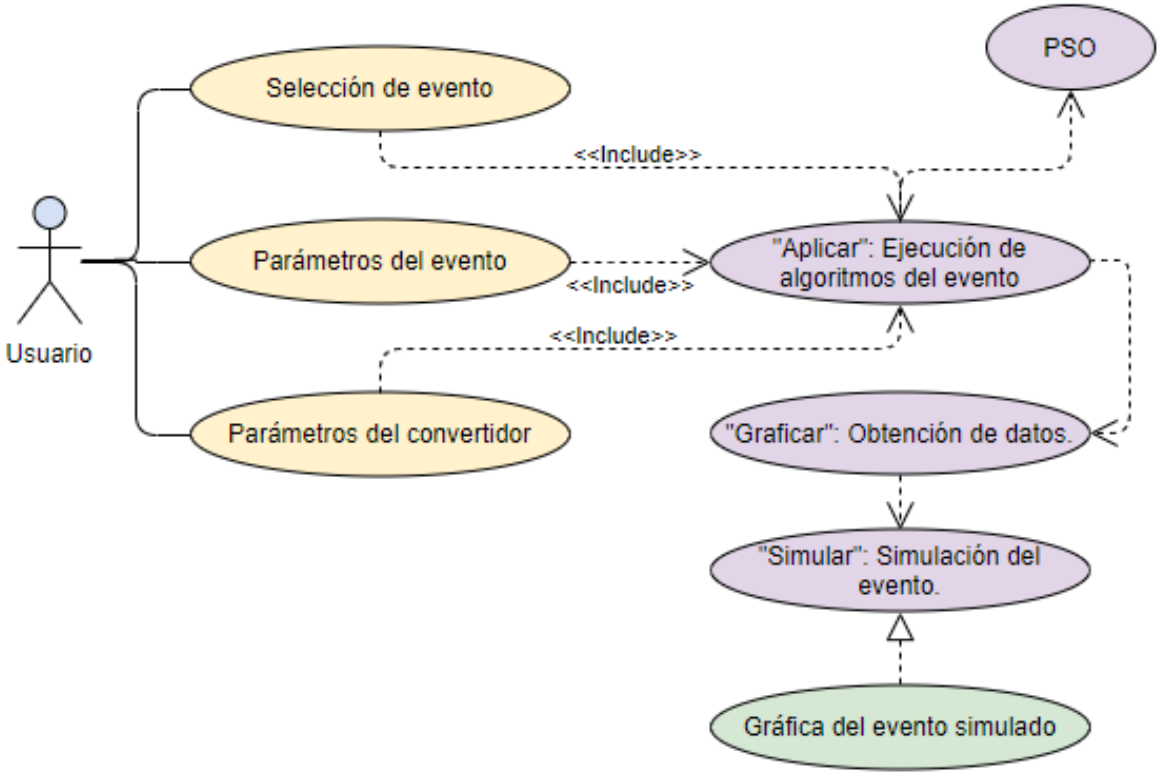


Figura 7.2: Diagrama de caso de uso de la interfaz gráfica. *Fuente: Elaboración propia.*

La interfaz cuenta con dos gráficas donde aparecerán el primer ciclo de onda sin el evento y con el evento. Posee una casilla donde el usuario puede seleccionar los eventos que quiere simular y se muestra una breve definición del evento seleccionado. Se encuentra otra casilla donde se ponen las características del evento, la cual cambia los parámetros según el evento seleccionado. Cuenta además con una casilla donde se ingresan los parámetros del convertidor deseado, en esta se mostrará el número de escalones del convertidor de acuerdo a los niveles escogidos, el V_{RMS} calculado y los ángulos de disparo seleccionados por el algoritmo PSO. Y por último, cuenta con tres botones: Aplicar para ingresar

todos los datos, Graficar para construir las gráficas de cada evento y obtener los datos de tiempo y magnitud, los cuales son necesarios para el siguiente botón que es Simular, el cual envía los datos obtenidos de Graficar a un programa de Simulink en donde se simula y se hace la respectiva verificación del evento simulado.

7.2. Depresión de voltaje *(sag)*

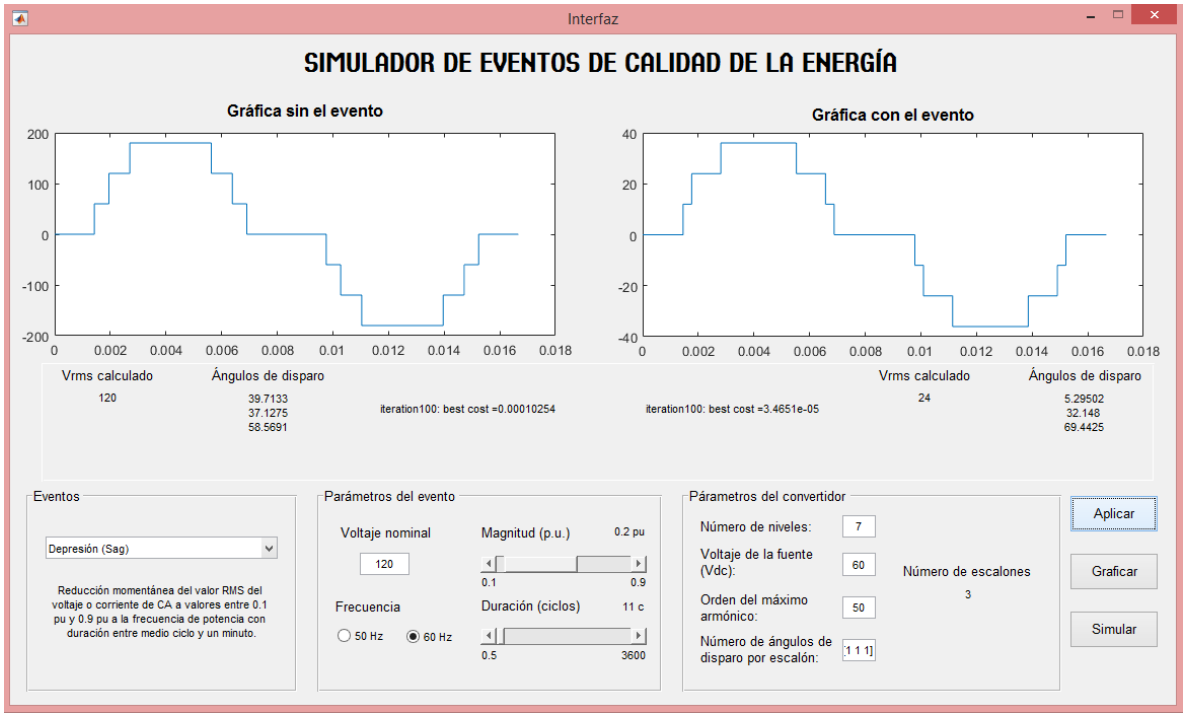


Figura 7.3: Interfaz gráfica del simulador de eventos para *sag*. *Fuente: Elaboración propia.*

En la Figura 7.3 se muestra un ejemplo de la forma en la que se deben llenar las casillas para este evento en específico. Se observa que los límites para magnitud (pu) y duración (ciclos) son entre 0.1 a 0.9 pu y 0.5 a 3600 ciclos (1 min) respectivamente como lo estipula el el estándar IEEE 1159 de 1995.

En el ejemplo expuesto, se puede observar que el error arrojado para la gráfica sin el evento y la del con el evento da 0.00010254 y 3.2851e-05 respectivamente,

lo cual es muy cercano a cero. Por tanto el voltaje RMS calculado para la primera gráfica es de 120, y para la segunda gráfica sería 24, el cual es el resultado deseado debido a que se escogió una magnitud del 20% del voltaje nominal.

7.3. Elevación de voltaje *(swell)*

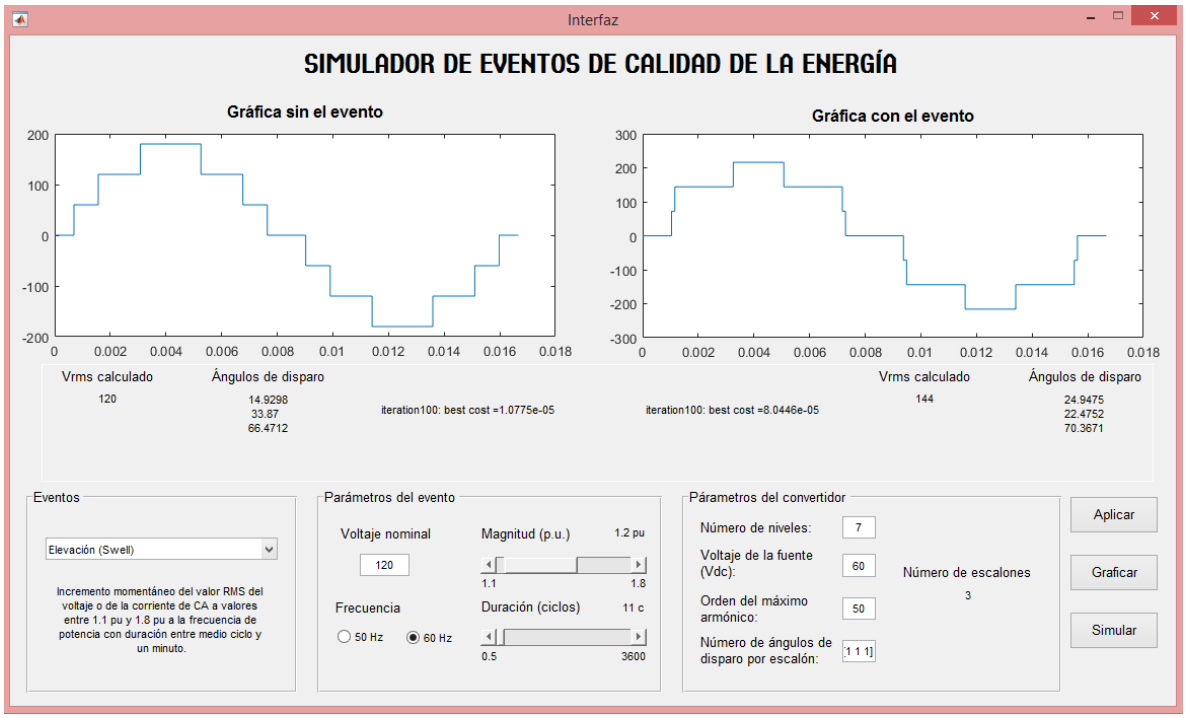


Figura 7.4: Interfaz gráfica del simulador de eventos para *swell*. *Fuente: Elaboración propia.*

En la Figura 7.4 se muestra un ejemplo de la forma en la que se deben llenar las casillas para este evento en específico. Se observa que los límites para magnitud (pu) y duración (ciclos) son entre 1.1 a 1.8 pu y 0.5 a 3600 ciclos (1 min) respectivamente como lo estipula el estándar IEEE 1159 de 1995.

En el ejemplo expuesto, se puede observar que el error arrojado para la gráfica sin el evento y la del con el evento da 1.0775e-05 y 8.0446e-05 respectivamente, lo cual es muy cercano a cero. Por tanto el voltaje RMS calculado para la primera

gráfica es de 120, y para la segunda gráfica sería 144, el cual es el resultado deseado debido a que se escogió una magnitud del 120% del voltaje nominal.

7.4. Bajo voltaje *(undervoltage)*

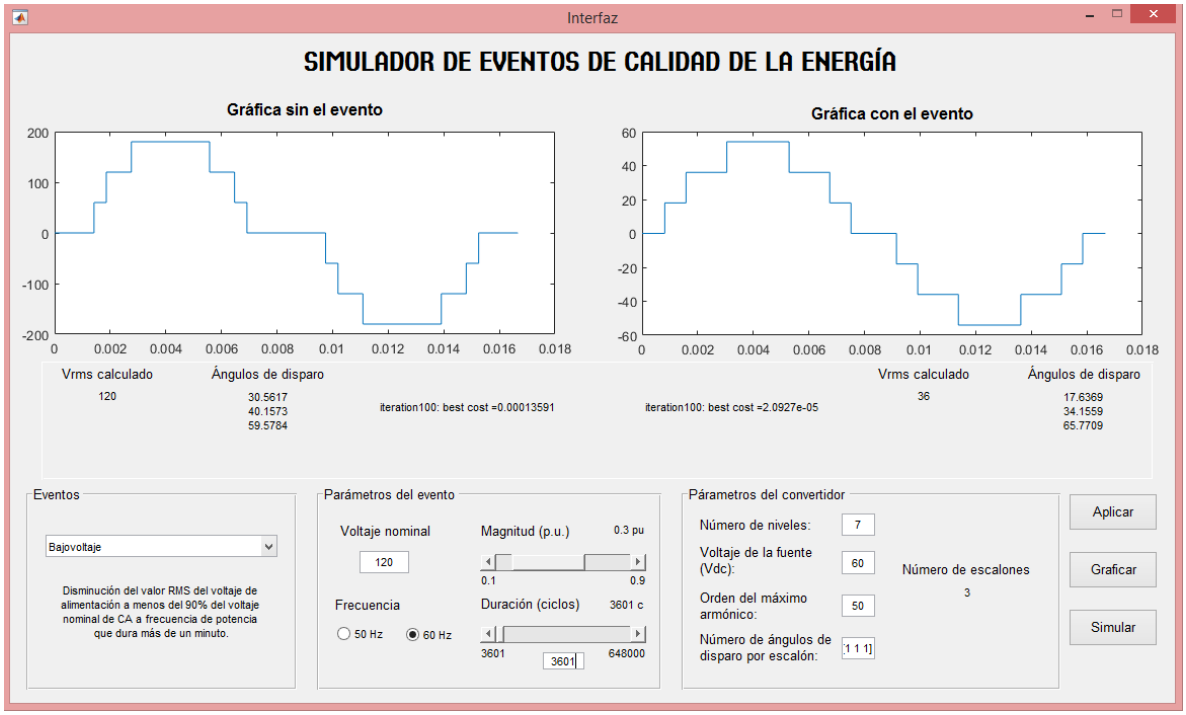


Figura 7.5: Interfaz gráfica del simulador de eventos para un bajo voltaje. *Fuente: Elaboración propia.*

En la Figura 7.5 se muestra un ejemplo de la forma en la que se deben llenar las casillas para este evento en específico. Se observa que los límites para magnitud y duración son entre 0.1 a 0.9 pu y valores mayores a 3600 ciclos (1 min) respectivamente como lo estipula el estándar IEEE 1159 de 1995.

En el ejemplo expuesto, se puede observar que el error arrojado para la gráfica sin el evento y la del con el evento da como resultado 0.00013591 y 2.0927e-05 respectivamente, lo cual es muy cercano a cero. Por tanto el voltaje RMS calculado para la primera gráfica es de 120, y para la segunda gráfica sería 36,

el cual es el resultado deseado debido a que se escogió una magnitud del 30% del voltaje nominal.

7.5. Sobrevoltaje *(overvoltage)*

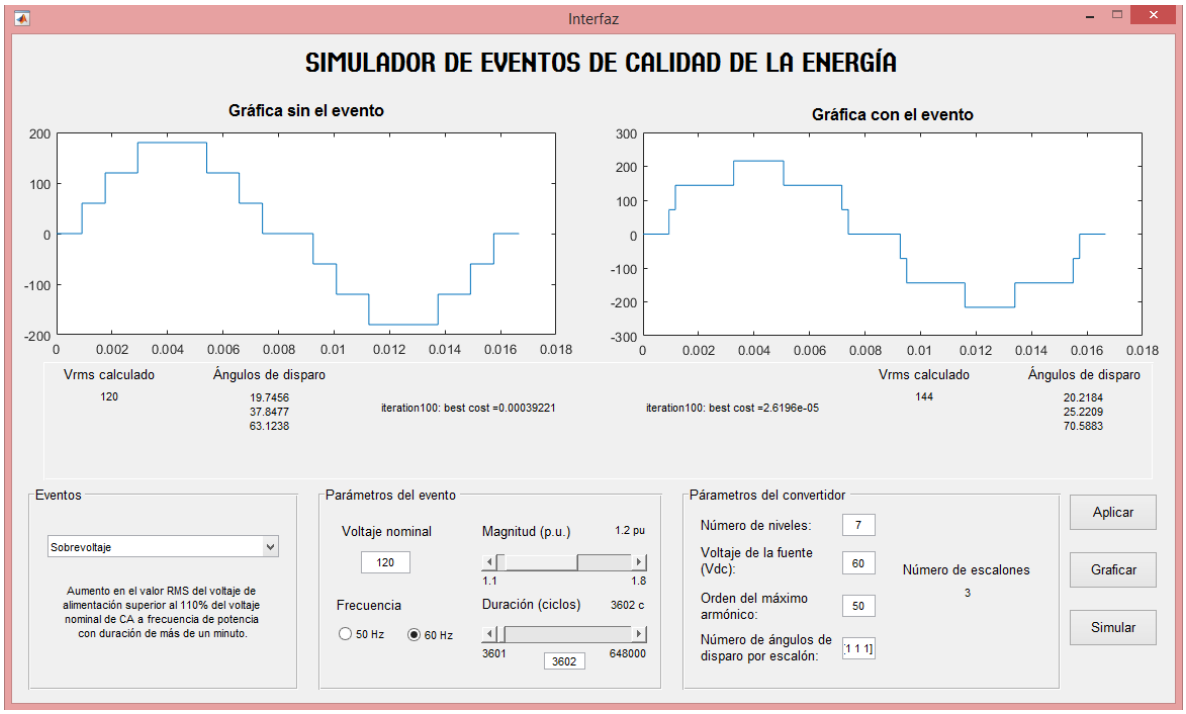


Figura 7.6: Interfaz gráfica del simulador de eventos para un sobrevoltaje. *Fuente: Elaboración propia.*

En la Figura 7.6 se muestra un ejemplo de la forma en la que se deben llenar las casillas para este evento en específico. Se observa que los límites para magnitud y duración son entre 1.1 a 1.8 pu y valores mayores a 3600 ciclos (1 min) respectivamente, como lo estipula el estándar IEEE 1159 de 1995.

En el ejemplo expuesto, se puede observar que el error arrojado para la gráfica sin el evento y la del con el evento da como resultado 0.00039221 y 2.6196e-05 respectivamente, lo cual es muy cercano a cero. Por tanto el voltaje RMS calculado para la primera gráfica es de 120, y para la segunda gráfica sería 144,

el cual corresponde al valor deseado debido a que se escogió una magnitud del 120% del voltaje nominal.

7.6. Interrupciones

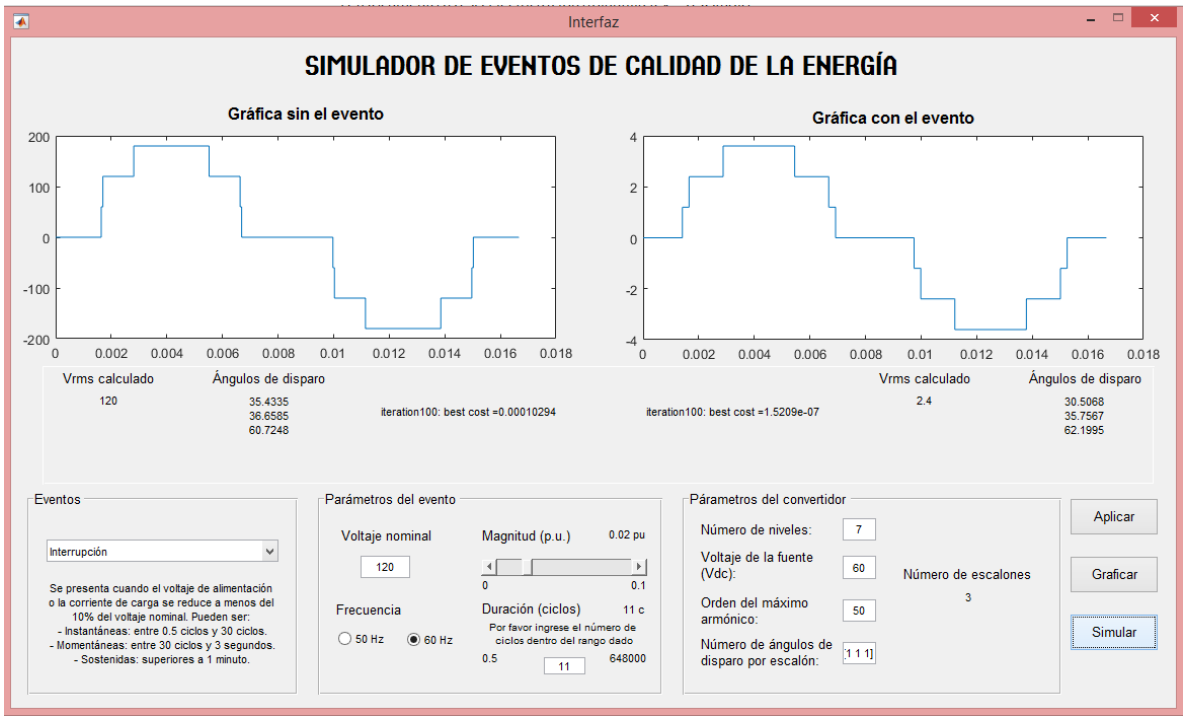


Figura 7.7: Interfaz gráfica del simulador de eventos para una interrupción instantánea. *Fuente: Elaboración propia.*

En la Figura 7.7 se muestra un ejemplo de la forma correcta en la que se deben llenar las casillas para este evento en específico. Se observa que los límites para magnitud y duración son entre 0 a 0.1 pu y valores de 0.5 a 1 min respectivamente como lo estipula el estándar IEEE 1159 de 1995.

En el ejemplo expuesto, se puede observar que el error arrojado para la gráfica sin el evento y la del con el evento da como resultado 0.00010294 y 1.5209e-07 respectivamente, lo cual es muy cercano a cero. Por tanto el voltaje RMS calculado para la primera gráfica es de 120, y para la segunda gráfica sería 2.4,

esto debido a que se escogió una interrupción instantánea (11 ciclos) con una magnitud del 2% del voltaje nominal.

7.7. Variación de frecuencia

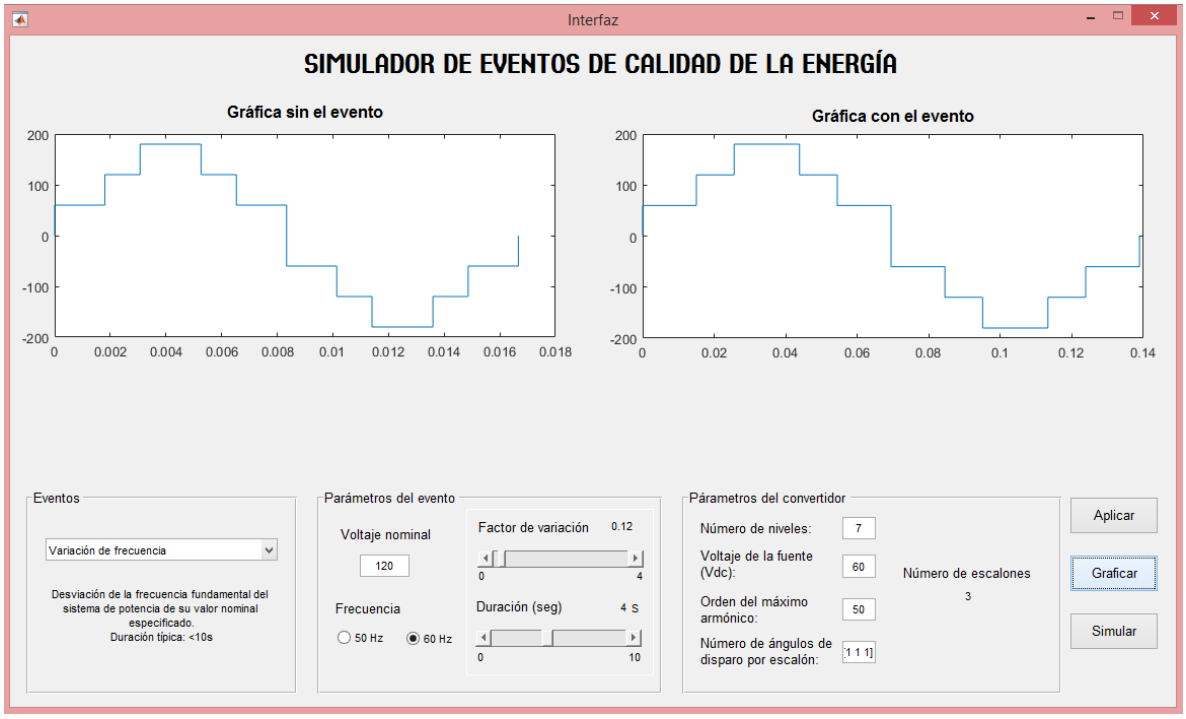


Figura 7.8: Interfaz gráfica del simulador de eventos para una variación de frecuencia. *Fuente: Elaboración propia.*

En la Figura 7.8 se muestra un ejemplo de la forma correcta en la que se deben llenar las casillas para este evento en específico. Se observa que solo se puede escoger de cero hasta cuatro veces la frecuencia fundamental, esto con la finalidad de no dar pie a valores exagerados de variaciones de frecuencia en la simulación, y la duración tiene un rango de cero a diez segundos como lo estipula el estándar IEEE 1159 de 1995.

En el ejemplo expuesto, se puede observar que al variar la frecuencia a una más pequeña, en este caso a 7.2 Hz, la onda se verá afectada por un incremento en la duración de los ciclos.

7.8. Armónicos

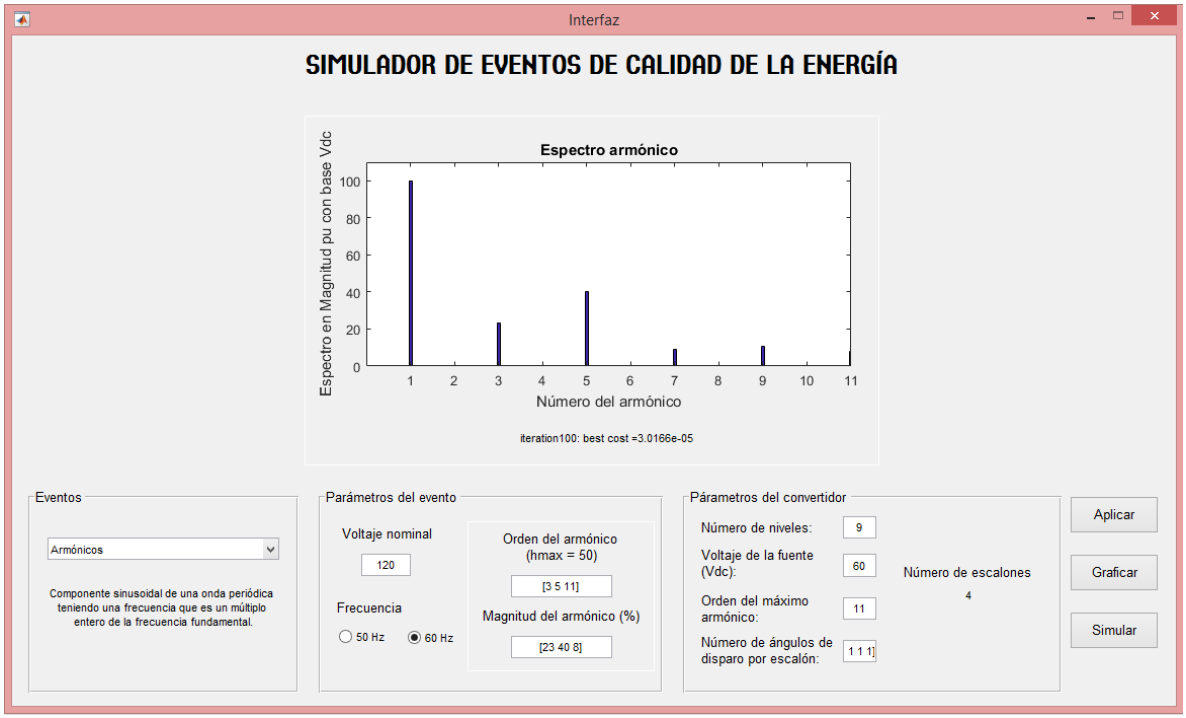


Figura 7.9: Interfaz gráfica del simulador de eventos para contenidos armónicos. *Fuente: Elaboración propia.*

En la Figura 7.9 se muestra un ejemplo de la forma correcta en la que se deben llenar las casillas para este evento en específico. Se observa que los parámetros de entrada del evento es el voltaje nominal, la frecuencia fundamental, el vector de los números de los armónicos que se quieren crear, y el vector de las magnitudes porcentuales de cada uno de los armónicos ingresados anteriormente, así como también se podrá llenar los parámetros del convertidor con valores por *default* o si, por el contrario, el usuario quiera unos parámetros específicos podrá hacerlo llenando cada uno de los *edit text*.

En el ejemplo expuesto, se puede observar que el error acumulado de las magnitudes para cada armónico fue de 3.0166e-05, lo que se traduce a que obtuvo ángulos de disparos que generarán el contenido armónico deseado.

Octavo Capítulo

Validación de los eventos simulados

En el presente capítulo se mostrarán los resultados obtenidos para cada uno de los eventos de la calidad de la energía simulados y su respectiva validación.

8.1. Depresión de voltaje (sag)

En la Figura 8.1 se muestra la generación resultante de una depresión de voltaje con una magnitud de 24 V que corresponde al 20% del voltaje nominal (120 V), una frecuencia fundamental de 60 Hz y una duración de 11 ciclos (0.1833 segundos).

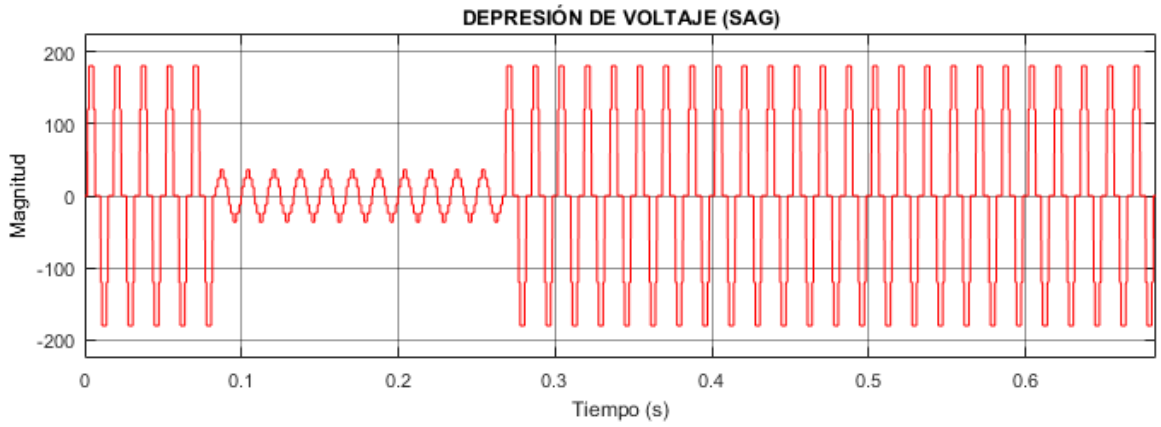


Figura 8.1: Gráfica resultante para la generación de una depresión de voltaje *(sag)*. *Fuente: Elaboración propia.*

Para la validación de este evento se hizo uso del bloque RMS de la toolbox de Power Systems en el blockset de Simulink, así como también de un subsistema creado para verificar el valor RMS (ver Figura 8.2), el cual consiste en sumar las magnitudes de cada uno de los armónicos hasta el armónico máximo (ingresado

por el usuario, por *default* es 50), multiplicar la sumatoria por una ganancia de 1/2 y sacarle la raíz cuadrada (ecuación 6.1).

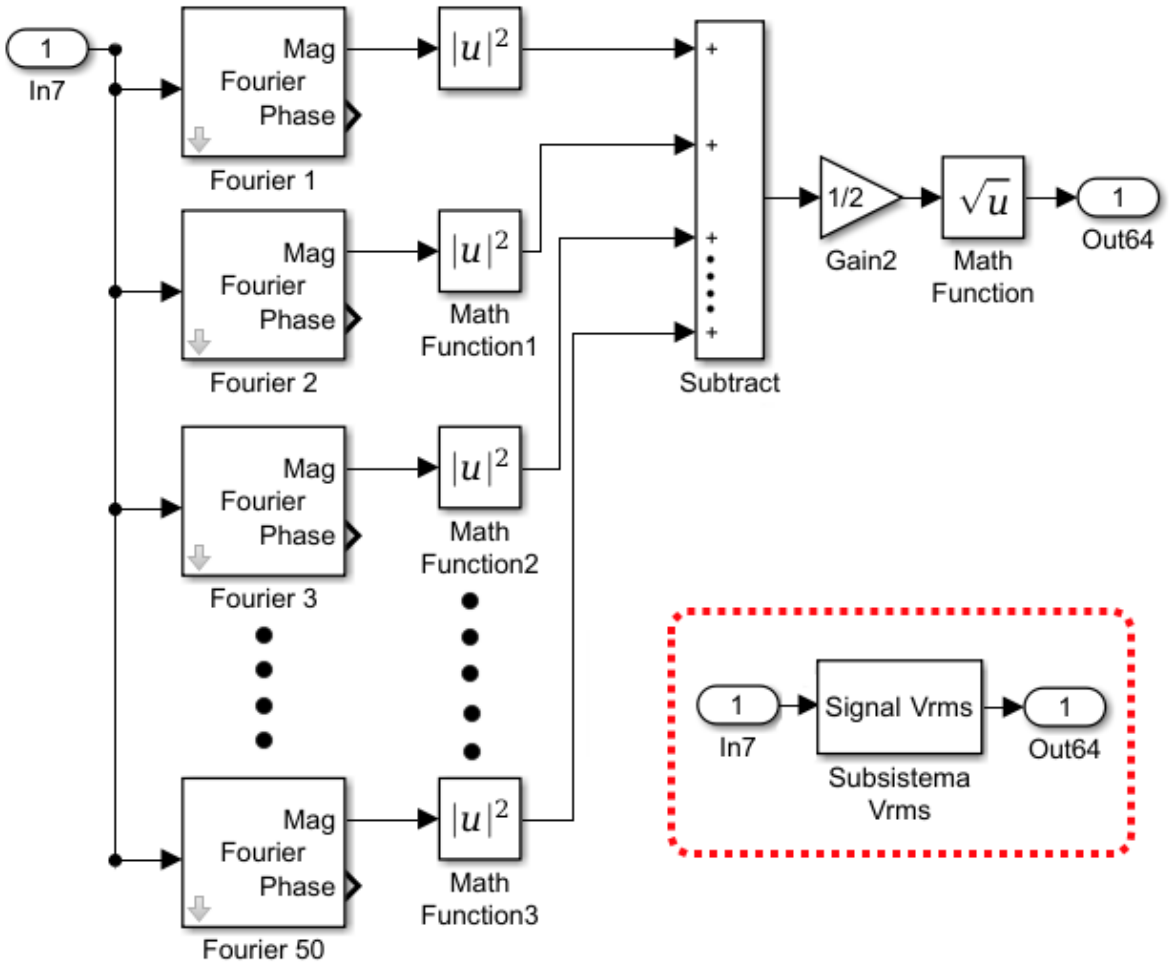


Figura 8.2: Subsistema creado para la validación de los eventos relacionados al VRMS. *Fuente: Elaboración propia.*

En la Figura 8.3 se puede observar que el VRMS de la depresión de voltaje simulada es 24.03 V, lo cual es un valor muy aproximado al deseado (24 V), dando un error relativo bastante bajo de 0.125 %. Los límites de duración en la ocurrencia del evento fueron: 0.0833 y 0.2666 segundos, lo cual da un tiempo de duración del evento de 0.1833 segundos que traducido a ciclos correspondería aproximadamente a 11 ciclos, el cual fue el valor ingresado por el usuario.

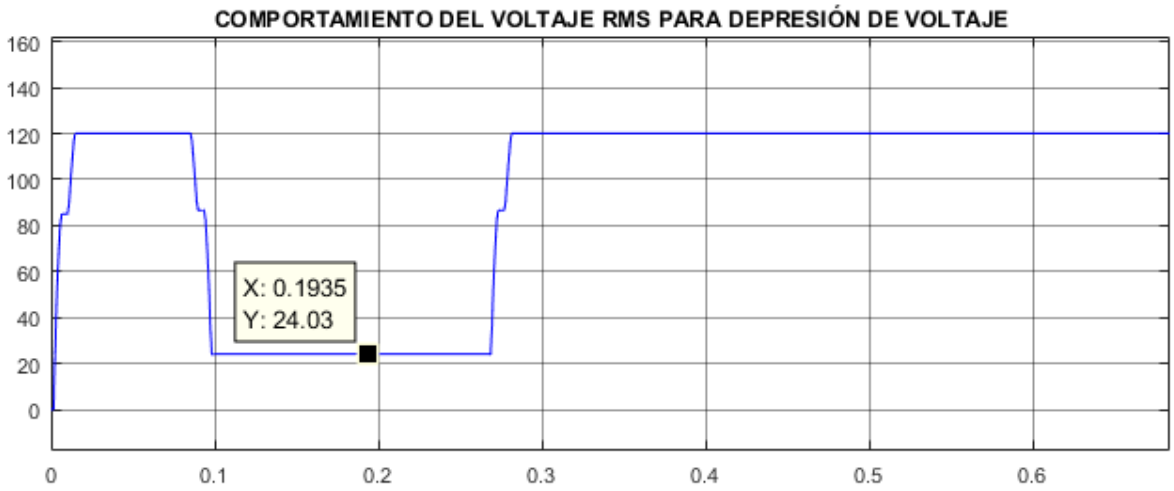


Figura 8.3: Validación del *sag* simulado con el bloque RMS de la librería Power System. *Fuente: Elaboración propia.*

8.2. Elevación de voltaje *(swell)*

En la Figura 8.4 se muestra la generación resultante de una elevación de voltaje con una magnitud de 144 V que corresponde al 120% del voltaje nominal (120 V), una frecuencia fundamental de 60 Hz y una duración de 11 ciclos (0.1833 segundos).

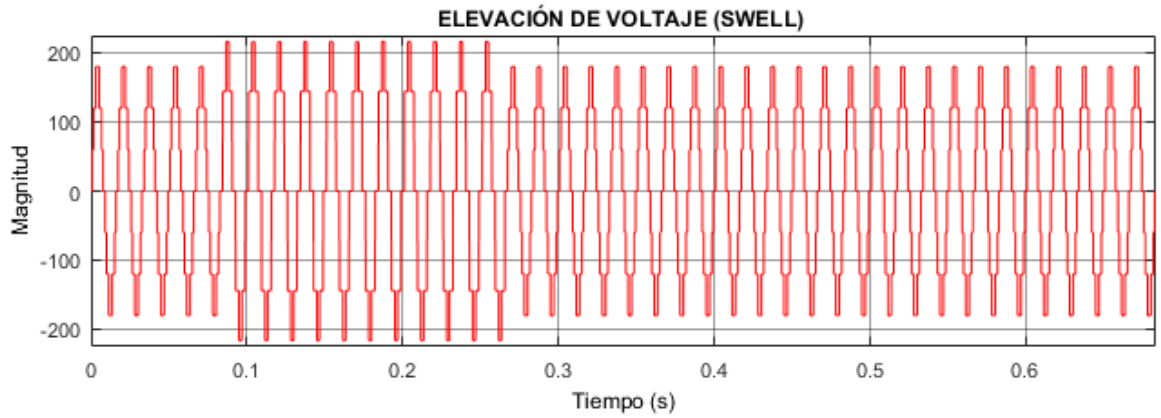


Figura 8.4: Gráfica resultante para la generación de una elevación de voltaje *(swell)*. *Fuente: Elaboración propia.*

Para la validación de este evento se hizo uso del bloque RMS de la toolbox de

Power Systems en el blockset de Simulink, así como también de un subsistema creado para verificar el valor RMS (ver Figura 8.2), el cual consiste en sumar las magnitudes de cada uno de los armónicos hasta el armónico máximo (ingresado por el usuario, por *default* es 50), multiplicar la sumatoria por una ganancia de 1/2 y sacarle la raíz cuadrada (ecuación 6.1).

Figura 8.5: Validación del *swell* simulado con el bloque RMS de la librería Power System. *Fuente: Elaboración propia.*

En la Figura 8.5 se puede observar que el VRMS de la depresión de voltaje simulada es 144.2 V, lo cual es un valor muy aproximado al deseado (144 V), dando un error relativo bastante bajo de 0.138 %. Los límites de duración en la ocurrencia del evento fueron: 0.0833 y 0.2666 segundos, lo cual da un tiempo de duración del evento de 0.1833 segundos que traducido a ciclos correspondería aproximadamente a 11 ciclos, el cual fue el valor ingresado por el usuario.

8.3. Bajo voltaje *(undervoltage)*

En la Figura 8.6 se muestra el bajo voltaje simulado con una magnitud de 36 V que corresponde al 30 % del voltaje nominal (120 V), una frecuencia fundamental de 60 Hz y una duración de 3601 ciclos (aproximadamente 1 minuto).

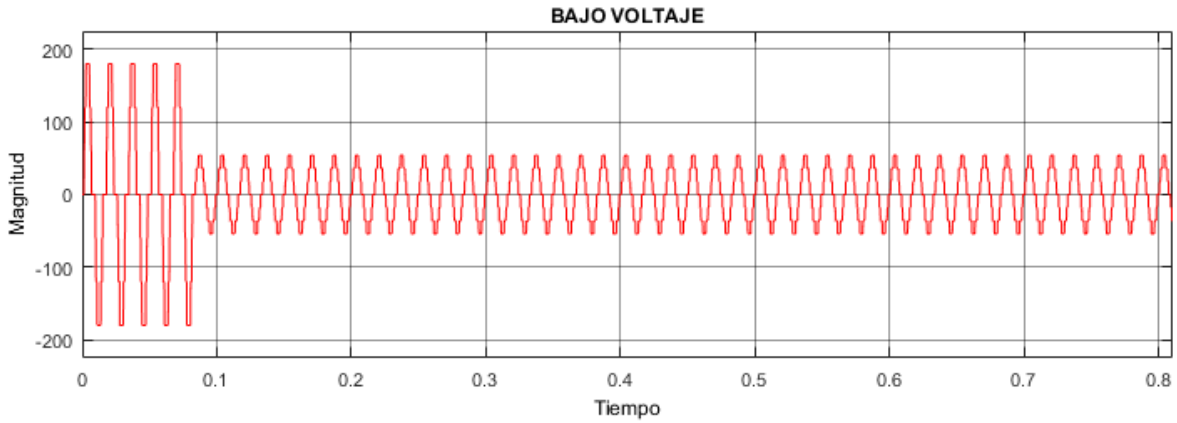


Figura 8.6: Gráfica resultante para la generación de un bajo voltaje *(undervoltage)*. *Fuente: Elaboración propia.*

Para la validación de este evento se hizo uso del bloque RMS de la toolbox de Power Systems en el blockset de Simulink, así como también de un subsistema creado para verificar el valor RMS (ver Figura 8.2), el cual consiste en sumar las magnitudes de cada uno de los armónicos hasta el armónico máximo (ingresado por el usuario, por *default* es 50), multiplicar la sumatoria por una ganancia de 1/2 y sacarle la raíz cuadrada (ecuación 6.1).

Figura 8.7: Validación del bajo voltaje simulado con el bloque RMS de la librería Power System. *Fuente: Elaboración propia.*

En la Figura 8.7 se puede observar que el VRMS de la depresión de voltaje

simulada es 36.05 V, lo cual es un valor muy aproximado al deseado (36 V), dando un error relativo bastante bajo de 0.138%. El tiempo de duración de este evento fue de aproximadamente 1 minuto que traducido a ciclos correspondería a 3601 ciclos, el cual fue el valor ingresado por el usuario.

8.4. Sobrevoltaje *(overvoltage)*

En la Figura 8.8 se muestra la generación resultante de un bajo voltaje con una magnitud de 144 V que corresponde al 120% del voltaje nominal (120 V), una frecuencia fundamental de 60 Hz y una duración de 3602 ciclos (aproximadamente 1 minuto).

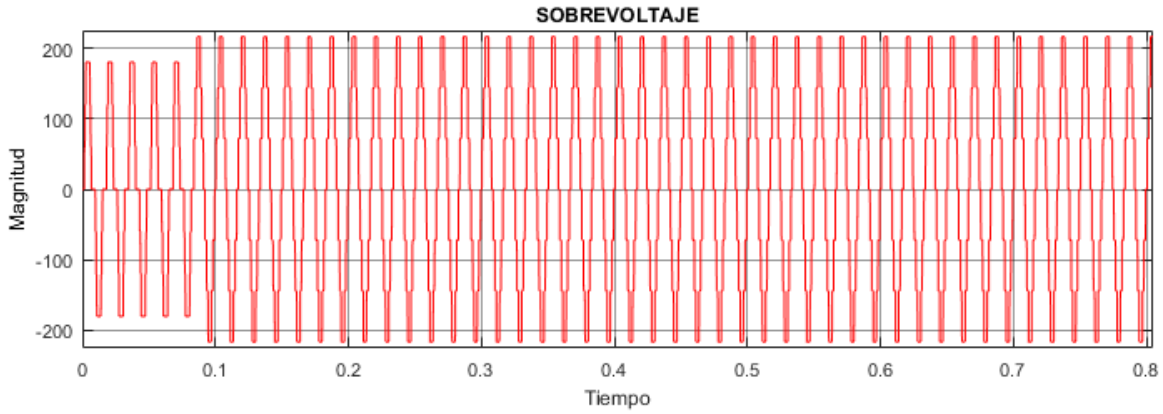


Figura 8.8: Gráfica resultante para la generación de un sobrevoltaje *(overvoltage)*. *Fuente: Elaboración propia.*

Para la validación de este evento se hizo uso del bloque RMS de la toolbox de Power Systems en el blockset de Simulink, así como también de un subsistema creado para verificar el valor RMS (ver Figura 8.2), el cual consiste en sumar las magnitudes de cada uno de los armónicos hasta el armónico máximo (ingresado por el usuario, por *default* es 50), multiplicar la sumatoria por una ganancia de 1/2 y sacarle la raíz cuadrada (ecuación 6.1).

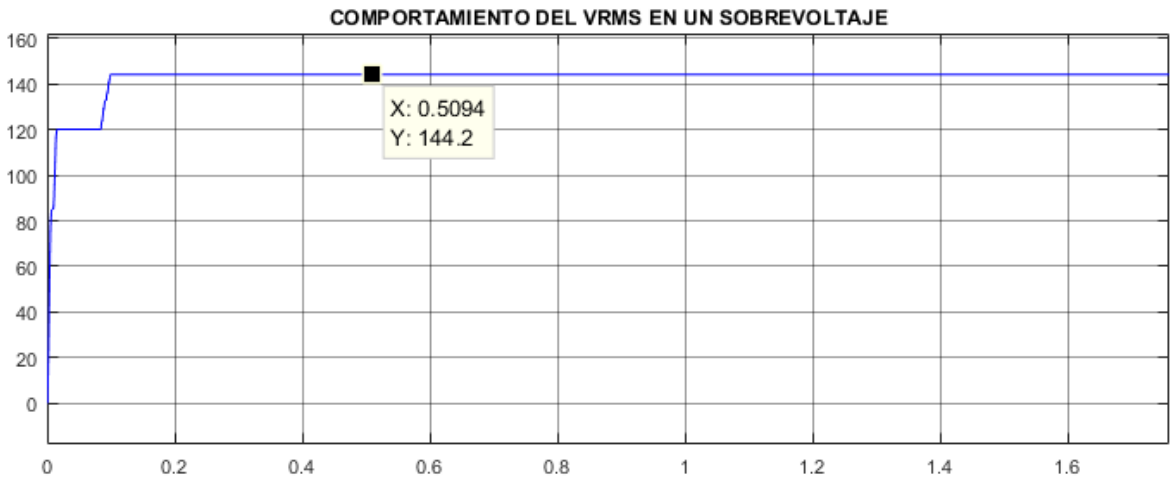


Figura 8.9: Validación del sobrevoltaje simulado con el bloque RMS de la librería Power System. *Fuente: Elaboración propia.*

En la Figura 8.9 se puede observar que el VRMS de la depresión de voltaje simulada es 144.2 V, lo cual es un valor muy aproximado al deseado (144 V), dando un error relativo bastante bajo de 0.138%. El tiempo de duración de este evento fue de aproximadamente 1 minuto que traducido a ciclos correspondería a 3602 ciclos, el cual fue el valor ingresado por el usuario.

8.5. Interrupciones

En la Figura 8.10 se muestra la generación resultante de un bajo voltaje con una magnitud de 2.4 V que corresponde al 2% del voltaje nominal (120 V), una frecuencia fundamental de 60 Hz y una duración de 11 ciclos (0.1833 segundos).

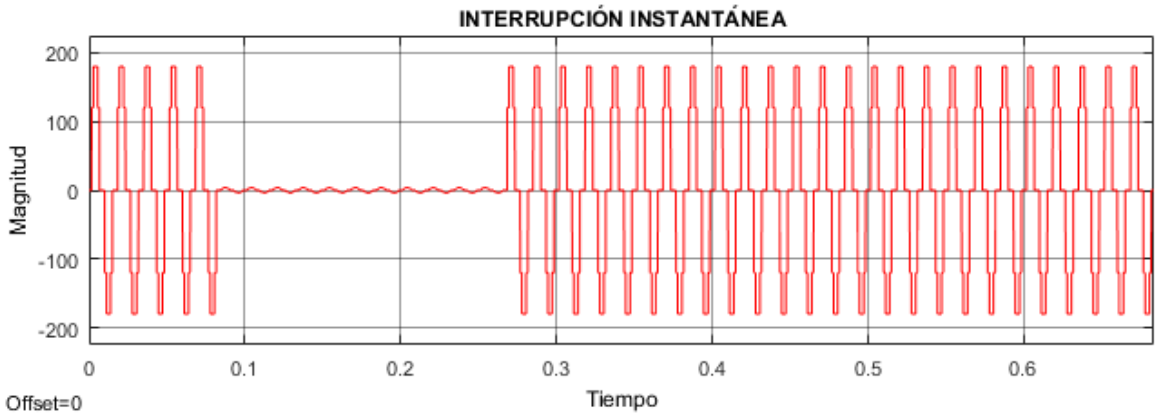


Figura 8.10: Gráfica resultante para la generación de una Interrupciones (tipo instantánea). *Fuente: Elaboración propia.*

Para la validación de este evento se hizo uso del bloque RMS de la toolbox de Power Systems en el blockset de Simulink, así como también de un subsistema creado para verificar el valor RMS (ver Figura 8.2), el cual consiste en sumar las magnitudes de cada uno de los armónicos hasta el armónico máximo (ingresado por el usuario, por *default* es 50), multiplicar la sumatoria por una ganancia de 1/2 y sacarle la raíz cuadrada (ecuación 6.1).

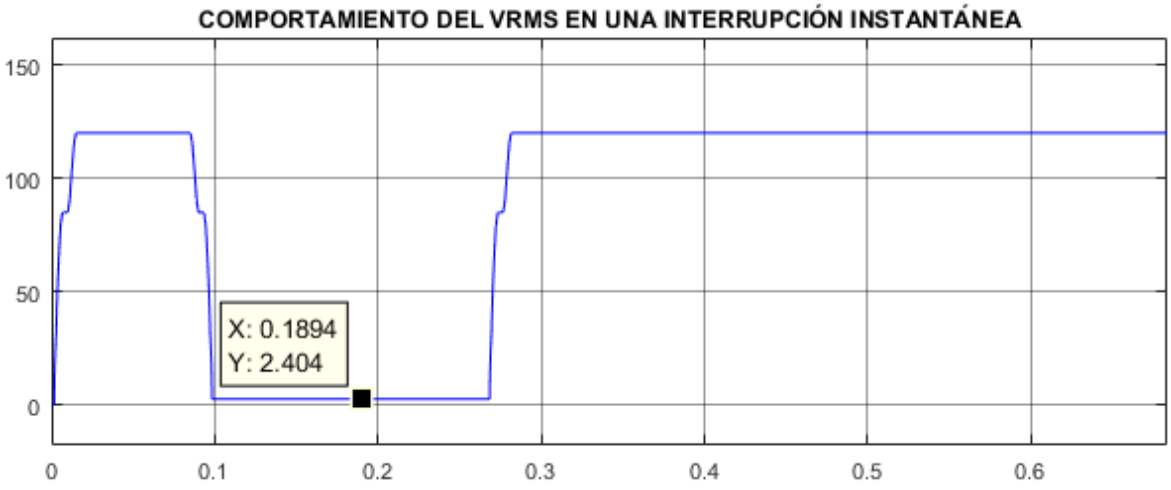


Figura 8.11: Validación de una interrupción de tipo instantánea simulada con el bloque RMS de la librería Power System. *Fuente: Elaboración propia.*

En la Figura 8.11 se puede observar que el VRMS de la depresión de voltaje

simulada es 2.404 V, lo cual es un valor muy aproximado al deseado (2.4 V), dando un error relativo bastante bajo de 0.166%. Los límites de duración en la ocurrencia del evento fueron: 0.0833 y 0.2666 segundos, lo cual da un tiempo de duración del evento de 0.1833 segundos que traducido a ciclos correspondería aproximadamente a 11 ciclos, el cual fue el valor ingresado por el usuario.

8.6. Variación de frecuencia

En la Figura 8.12 se muestra la generación resultante de una variación de frecuencia con un factor de variación de 0.12 veces la frecuencia fundamental (60 Hz), es decir con una frecuencia de 7.2 Hz, un voltaje nominal de 120 V y una duración de 4 segundos.

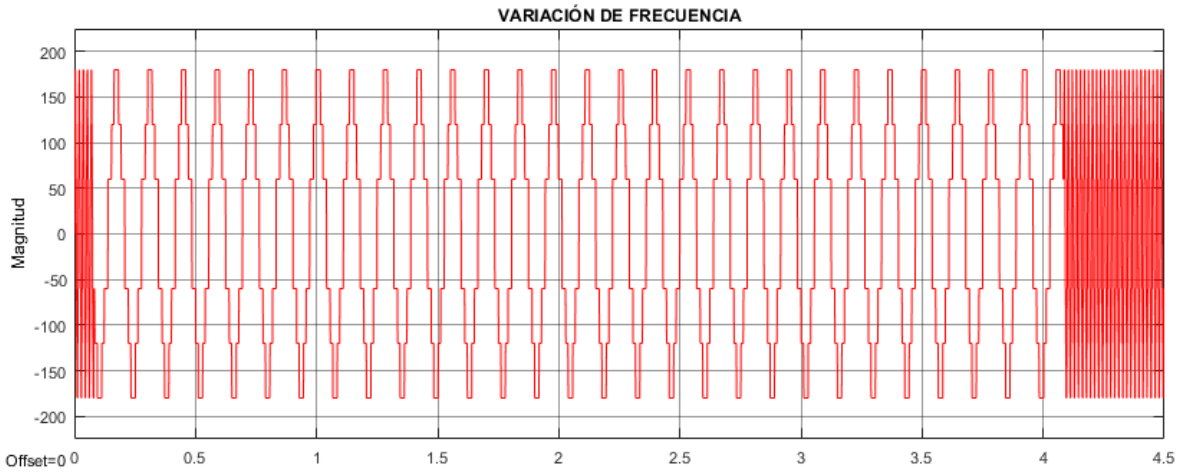


Figura 8.12: Gráfica resultante del evento para una variación de frecuencia de 0.12 veces la frecuencia fundamental (60 Hz) con duración de 4 segundos. *Fuente: Elaboración propia.*

La validación de este evento se puede observar en la Figura 8.13. Para esto con ayuda de la herramienta *"Cursor Measurements"* se tomaron dos puntos en dos picos contiguos de la onda del evento con el fin de hallar el periodo, teniendo el periodo se obtiene la frecuencia con la ecuación 8.1 dando como resultado lo resaltado en color rojo 7.2 Hz, el cual es el valor al que el usuario varió la frecuencia.

$$f = \frac{1}{\Delta T} \tag{8.1}$$

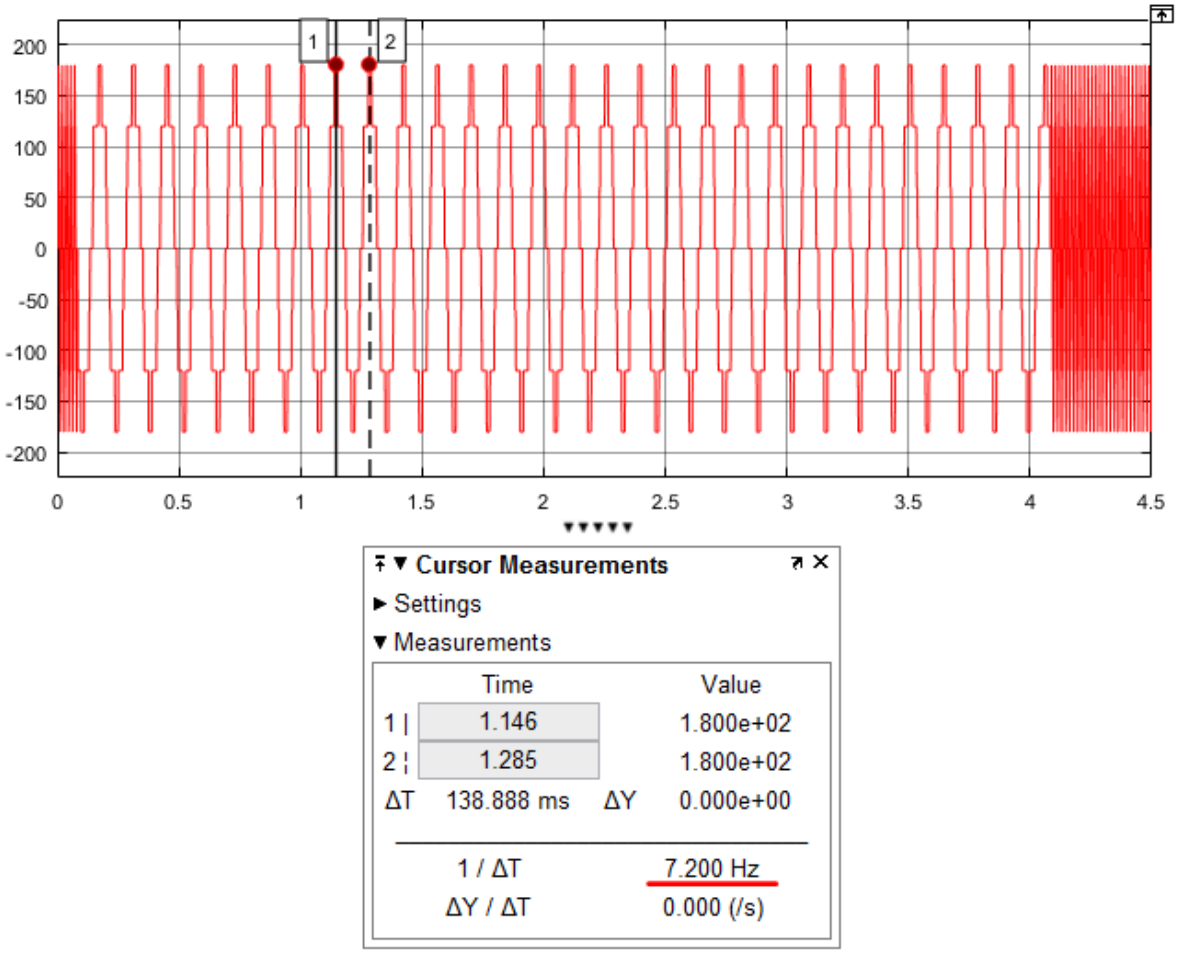


Figura 8.13: Validación de una variación de frecuencia simulada. *Fuente: Elaboración propia.*

8.7. Armónicos

En la Figura 8.14 se muestra la generación del armónico 3, 5 y 11 con magnitudes del 23%, 40% y 18% de la fundamental respectivamente, un voltaje nominal de 120 V y frecuencia fundamental de 60 Hz.

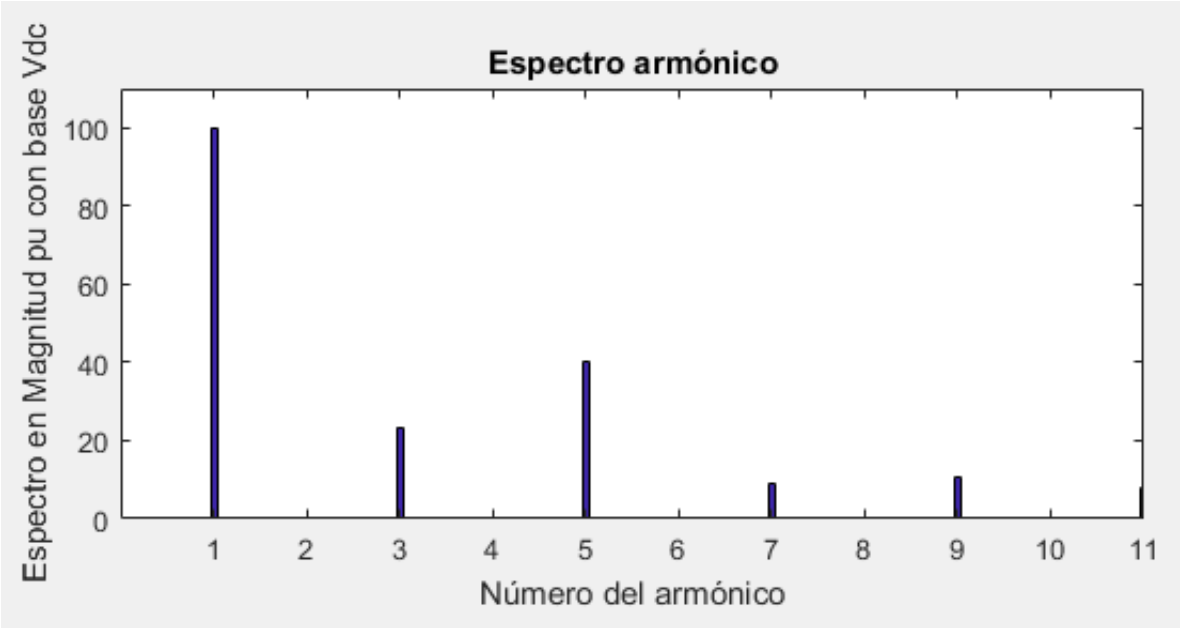


Figura 8.14: Gráfica resultante del contenido armónico para los datos ingresados por el usuario. *Fuente: Elaboración propia.*

Al no verificar la existencia de otros armónicos en el algoritmo estos aparecerán, puesto que el algoritmo lo que hace es asegurarse de que la magnitud de los armónicos deseados se encuentren en el valor ingresado por el usuario. Los armónicos por fuera de los seleccionados no entrarán en la función objetivo del algoritmo PSO, por tanto la magnitud de ellos no se hace relevante para éste.

La validación de este evento consistió en obtener los datos de la generación del contenido armónico simulado (vector tiempo y vector magnitud) y evaluarlos en un bloque de análisis de la Transformada Rápida de Fourier *FFT* del programa Simulink.

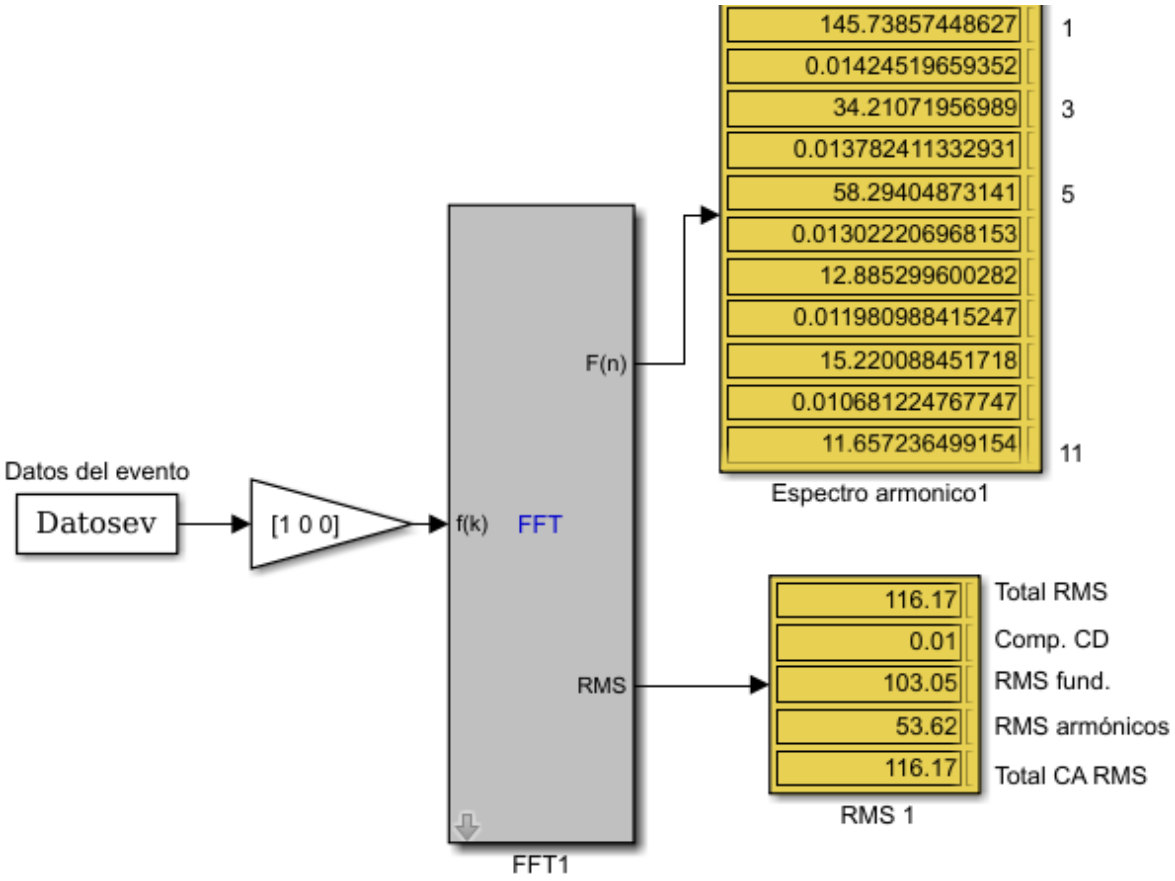


Figura 8.15: Validación del contenido armónico. *Fuente: Elaboración propia.*

En la Figura 8.15 se muestra el esquema en Simulink utilizado para la validación; la entrada son los datos del evento y la salida son los análisis dados por el bloque ***FFT***, junto con los valores RMS. Se puede observar que la magnitud de la fundamental es de 145.7385, la del tercer armónico es 34.2107, la del quinto armónico es 58,2940 y la del onceavo armónico es de 11.6572. Tomando como el 100% la magnitud de la fundamental, el porcentaje para los anteriores armónicos con respecto al armónico uno se pude calcular con la expresión de la ecuación 8.2, dando como resultado 23.474% para el armónico 3, 39.999% para el armónico 5 y 7.998% para el armónico 11, por lo tanto los valores resultantes no se alejan demasiado de los ingresados.

$$Magnitud\% = \frac{Magnidud_{h1}}{Magnidud_{hn}} * 100 \tag{8.2}$$

En la Figura 8.16 se muestra el espectro armónico arrojado por el bloque *FFT*, en el cual se observa una gráfica muy similar a la obtenida en la Figura 8.14, con la salvedad que en ésta las magnitudes no se encuentran expresadas en porcentaje sino en base al parámetro *"Base peak"*.

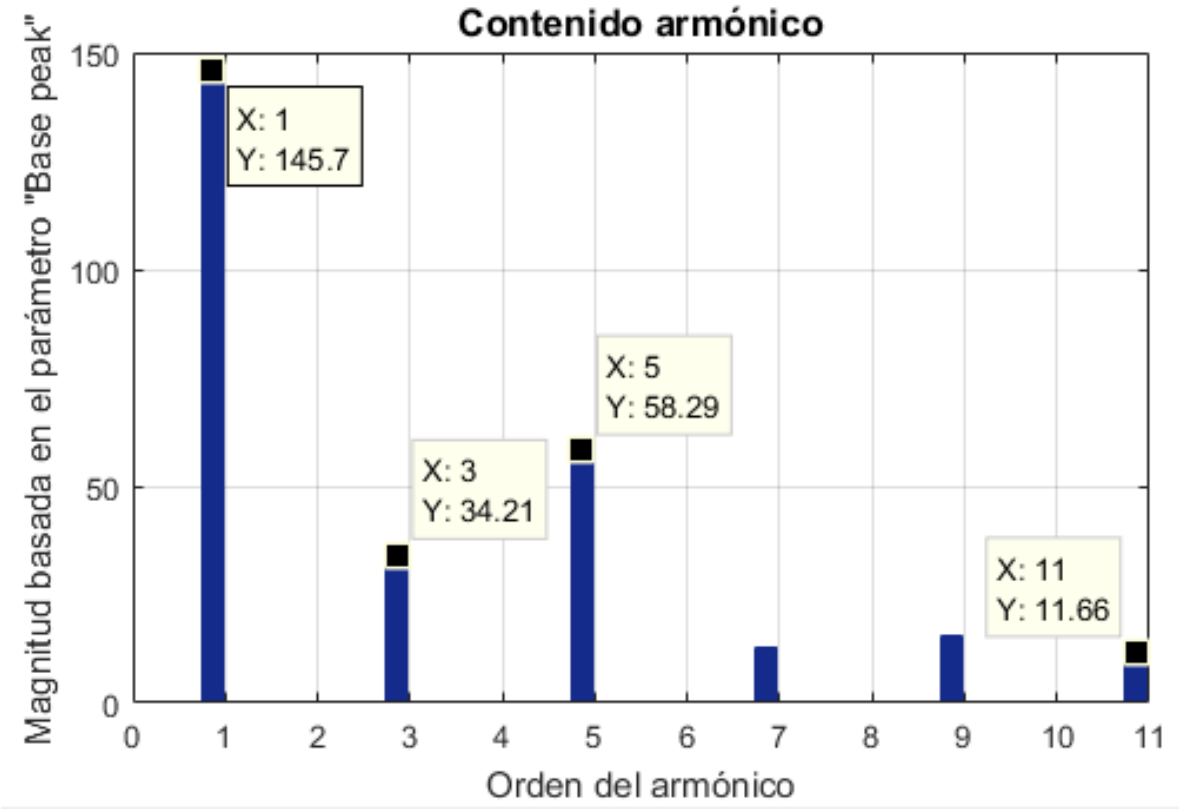


Figura 8.16: Espectro armónico obtenido de la validación del contenido armónico. *Fuente: Elaboración propia.*

Noveno Capítulo

Conclusiones

- Con este simulador se propuso una herramienta para estudiar diferentes eventos de calidad de la energía que servirá de guía para que en implementaciones futuras se pueda observar el comportamiento de diferentes dispositivos ante tales eventos.

- Se valida el algoritmo PSO como una estrategia alternativa con resultados positivos en cuanto a la estimación de ángulos de disparos para convertidores con valores de voltaje RMS ya dados.

- Para el contenido armónico se logró simular los armónicos dados por el usuario, sin embargo no se primó para que solo aparecieran estos, por lo cual con los ángulos hallados se pueden encontrar armónicos diferentes a los deseados.

- Como trabajo a futuro se pueden agregar más de una función objetivo para cada uno de los eventos simulados, pero se tendría que tener en cuenta que el algoritmo PSO requerirá más tiempo en hallar los valores de ángulo de disparo, debido a que necesita más espacio de almacenamiento; y además, puede que en algunos casos no llegue a converger con un error mínimo.

Bibliografía

[1] Bingham, Richard P.: *Measurement instruments for power quality monitoring*. En *Transmission and Distribution Exposition Conference: 2008 IEEE PES Powering Toward the Future, PIMS*, páginas 1–3, Chicago, USA, 2008. IEEE, ISBN 9781424419036.

[2] IEEE: *IEEE Recommended Practice and Requirements for Harmonic Control in Electric Power Systems*. IEEE Std 519-2014 (Revision of IEEE Std 519-1992), páginas 1–29, 2014.

[3] OPAL-RT TECHNOLOGIES: *Online Store: Electrical Real-Time Simulation Systems*, 2018. `https://www.opal-rt.com/online-store/?category=hil-offers`.

[4] Kezunovic, Mladen y Yuan Liao: *A novel software implementation concept for power quality study*. IEEE Transactions on Power Delivery, 17(2):544–549, 2002.

[5] IEEE: *IEEE Recommended Practice for Monitoring Electric Power Quality*. IEEE Std 1159-2009 (Revision of IEEE Std 1159-1995), páginas 1–81, 2009.

[6] Carrillo-Rojas, Galo, Juan Andrade-Rodas, Antonio Barragán-Escandón y Ana Astudillo-Alemán: *Impacto de programas de eficiencia energética eléctrica, estudio de caso: Empresas alimentarias en Cuenca, Ecuador*. DYNA (Colombia), 81(184):41–48, 2014, ISSN 00127353.

[7] Kezunovic, Mladen y Yuan Liao: *A Novel Method for Equipment Sensitivity Study During Power Quality Events*. En *IEEE Power Engineering Society Winter Meeting*, páginas 993–998, Singapore, Singapore, 2000. IEEE.

[8] Monedero, Iñigo, Carlos León, Jorge Ropero, José Luis De La Vega, Juan C. Montaño y José Manuel Elena: *A system for the generation and detection of*

electrical disturbances. En *Fourth IASTED International Conference Power And Energy System*, páginas 180–185, Rhodes, Grecia, 2004.

[9] Eloy-García, Joaquin, Juan Carlos Vasquez y Josep M. Guerrero: *Grid simulator for power quality assessment of micro-grids*. IET Power Electronics, 6(4):700–709, 2013, ISSN 17554535.

[10] Nahoum, Pamela, Emile Yammine, Elie Karam, Maged B. Najjar y Moustapha El Hassan: *Real generation of power quality disturbances*. En *Third International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE)*, páginas 224–229, Beirut, Líbano, 2015. IEEE.

[11] Chen, Cheng I., Chieh Yin Cheng y Yeong Chin Chen: *Design of programmable power-quality signal generator for power disturbance testing of consumer electronics*. En *2015 IEEE 4th Global Conference on Consumer Electronics, GCCE*, páginas 436–437, Osaka, Japón, 2015. IEEE, ISBN 9781479987511.

[12] Inci, Mustafa, Tugce Demirdelen, Adnan Tan, Tahsin Köroglu, M. Ugras Cuma, K. Cagatay Bayindir y Mehmet Tümay: *A novel low cost sag/swell generator*. En *2015 IEEE 6th International Symposium on Power Electronics for Distributed Generation Systems, (PEDG)*, páginas 1–4, Aachen, Alemania, 2015. IEEE, ISBN 9781479985869.

[13] Shen, Zhuoxuan, Tong Duan y Venkata Dinavahi: *Design and Implementation of Real-Time Mpsoc-FPGA-Based Electromagnetic Transient Emulator of CIGRÉ DC Grid for HIL Application*. IEEE Power and Energy Technology Systems Journal, 5(3):104–116, 2018, ISSN 2332-7707.

[14] Ashourianjozdani, Mohammadhossein, Luiz A.C. Lopes y Pragasen Pillay: *Power Electronic Converter Based PMSG Emulator: A Testbed for Renewable Energy Experiments*. IEEE Transactions on Industry Applications, 54(4):3626–3636, 2018, ISSN 00939994.

[15] Parizad, Ali, Sobhan Mohamadian, Mohamad Esmaeil Iranian y Josep M. Guerrero: *Power System Real-Time Emulation: A Practical Virtual Instru-*

mentation to Complete Electric Power System Modeling. IEEE Transactions on Industrial Informatics, 15(2):889–900, 2019, ISSN 15513203.

[16] Saito, Kenichiro y Hirofumi Akagi: *A Real-Time Real-Power Emulator of a Medium-Voltage High-Speed Induction Motor Loaded With a Centrifugal Compressor*. IEEE Transactions on Industry Applications, 55(5):4821–4833, 2019, ISSN 0093-9994.

[17] Comisión de Regulación de Energía y Gas CREG: *Resolución CREG 070-98. Reglamento de distribución de energía eléctrica*, 1998.

[18] IEEE: *IEEE Recommended Practice for Powering and Grounding Sensitive Electronic Equipment*. IEEE Std 1100-2005 (Revision of IEEE Std 1100-1999), página 10, 2005.

[19] IEC: *Electromagnetic compatibility (EMC) - Part 2-4: Environment - Compatibility levels in industrial plants for low-frequency conducted disturbances*. IEC Std 61000-2-4:2002, páginas 1–75, 2002.

[20] Holguin, Marcos y David Gomezcoello: *Análisis de la calidad de energía eléctrica en el "Nuevo Campus"de la Universida Politécnica Salesiana*. Tesis, Universida Politécnica Salesiana, 2010. https://dspace.ups.edu.ec/bitstream/123456789/2110/13/UPS-GT000145.pdf.

[21] Saucedo, Daniel, José Texis y Zoar Flores: *Factores que afectan la calidad de la energía y su solución*. Tesis, Instituto Politécnico Nacional, 2008. http://tesis.ipn.mx/bitstream/handle/123456789/429/FINALsauicedomtz.pdf?sequence=1.

[22] Cuevas Bravo, David: *Calidad de la energía en los sistemas eléctricos de potencia*. Tesis, Universidad Nacional Autónoma de México, 2011. https://es.scribd.com/document/389163629/Calidad-de-La-Energia-en-Los-Sistemas-Electricos-de-Potencia.

[23] Ruggero, Bruno y M Sánchez: *Incidencias de Cargas no Lineales en Transformadores de Distribución*. Revista Científica de la UCSA, 1(1):33–51, 2014.

[24] Herrera Heredia, Juan Carlos: *Determinación de la potencia de cargas no lineales*. Tesis, Escuela Politécnica Nacional, 1997. https://bibdigital.epn.edu.ec/bitstream/15000/5992/1/T290.pdf.

[25] Martínez García, Salvador: *Nociones básicas sobre las cargas eléctricas*. En *Alimentación de equipos informáticos y otras cargas críticas*, capítulo 1, páginas 1–3. McGraw-Hill, Madrid, España, primera edición, 1992, ISBN 84-7615-920-X.

[26] Sánchez Cortés, Miguel: *Calidad de la energía eléctrica*. Instituto Tecnológico de Puebla, Puebla, México, primera edición, 2009.

[27] Zbigniew, H. y A. Bién: *Power Quality Application Guide*. En *Voltage Disturbances: Flicker Measurement*, página 12. Copper Development Association, Cracovia, Polonia, 2005.

[28] Kim, Yong Jung y Hyosung Kim: *Optimal inductance ratio of LCL filter for grid connected inverters considering with low order harmonics*. En *IECON Proceedings (Industrial Electronics Conference)*, páginas 2355–2360, Florencia, Italia, 2016. IEEE, ISBN 9781509034741.

[29] Rodriguez, J L D, L D P Fernandez y ...: *THD improvement of a PWM cascade multilevel power inverters using genetic algorithms as optimization method*. WSEAS Transactions on Power Systems, 10:46–54, 2015.

[30] Diaz Rodriguez, Jorge Luis, Luis David Pabón Fernández y Jorge Luis Contreras Peña: *Low-cost platform for the evaluation of single phase electromagnetic phenomena of power quality according to the IEEE 1159 standard*. DYNA, 82(194):119–129, dec 2015, ISSN 2346-2183. http://www.revistas.unal.edu.co/index.php/dyna/article/view/46922.

[31] Duarte, César y Jabid Quiroga: *Algoritmo PSO para identificación de parámetros en un motor DC*. Revista Facultad de Ingenieria, (55):116–124, 2010, ISSN 01206230.

[32] Ali, Syed Saad Azhar, Ramani Kannan y M. Suresh Kumar: *Exploration of Modulation Index in Multi-level Inverter using Particle Swarm Optimiza-*

tion Algorithm. Procedia Computer Science, 105(December 2016):144–152, 2017, ISSN 18770509. http://dx.doi.org/10.1016/j.procs.2017.01.194https://linkinghub.elsevier.com/retrieve/pii/S1877050917302120.

[33] Kennesy, James y Russell Eberhart: *Particle Swarm Optimization*. En *Proceedings of ICNN'95 - International Conference on Neural Networks*, páginas 1942–1948, Perth, Australia, 1995. IEEE, ISBN 0780327683.

[34] Diaz Rodriguez, Jorge Luis, Luis David Pabon Fernandez y Edison Andres Caicedo Peñaranda: *Multiobjective Genetic Algorithm to Minimize the THD in Cascaded Multilevel Converters with V/F Control*. En *Applied Computer Sciences in Engineering. Communications in Computer and Information Science.*, páginas 456–468. Cartagena, Colombia, 2017. http://link.springer.com/10.1007/978-3-319-66963-2{_}41.

Anexo A

Función para el cálculo del voltaje RMS para la modulación

```
function VRMS = fun_VRMS(alpha,V)

load valuestruct.mat

A = 0;
B = 0;
m = (k-1)/2;

for n = 1:2:hmax
        for i = 1:m
                for j = 1:L(i)
                        A = cosd(n*alpha(i))*(-1)^(j-1) + A;
                end
        end
        i=0;
        Amplitud = (A*((4*V)/(pi*n))); %Amplitud
        B = (1/2)*(A*((4*V)/(pi*n)))^2 + B;
        A = 0;
        end

VRMS = B^(1/2);
end
```

Anexo B

Función objetivo para los eventos de la sección 6.2

```
function error = fun_eVRMS(alpha,V)

load valuestruct.mat

A = 0;
B = 0;
m = (k-1)/2;

for n = 1:2:hmax
        for i = 1:m
                for j = 1:L(i)
                        A = cosd(n*alpha(i))*(-1)^(j-1) + A;
                end
        end
        i=0;
        Amplitud = (A*((4*V)/(pi*n)));
        B = (1/2)*(A*((4*V)/(pi*n)))^2 + B;
        A = 0;
        end

eVRMS = B^(1/2);
error = abs(VRMS - eVRMS);
end
```

Anexo C

Función para la generación de las gráficas de los eventos

```
1       function Datos = Const_grafica(alpha)
2
3       load valuestruct.mat
4
5       ciclo = [5 10000];
6       alpha = sort(alpha);
7       acum = 0; a = 1; ac = 1; j = 1; am = 1;
8       tam_L = length(L);
9
10      for i = 1:tam_L
11      acum = acum+L(i);
12      end
13
14      Vnegdesc = zeros(1,acum*2);
15      Vposdesc = zeros(1,acum*2);
16      Vnegasc = zeros(1,acum*2);
17      dob = zeros(1,acum*2);
18
19      %----------------Vector estados/voltajes----------------%
20      for n = 1:tam_L
21      for i = 1:L(n)
22      lvl(a) = rem(i,2) + (n-1);
23      a = a+1;
24      end
25      n=0;
26      end
27
```

```
28          lvl = [0 lvl];
29
30          dup = kron(lvl,ones(2,1));
31          rep_lvl = reshape(dup,1,size(dup,1)*size(dup,2));
32
33          elim = rep_lvl(2:(length(rep_lvl)-2));
34
35          Vect_esc = [rep_lvl fliplr(elim) -elim -fliplr(rep_lvl)];
36          %-------------------------------------------------------%
37
38          %--------------------Vector angulos/tiempos-------------------%
39          for n = 1:tam_L
40          for i = 1:L(n)
41          d = alpha(am); %1er cuarto de onda
42          posdesc(i) = (180-alpha(am)); %2do cuarto de onda
43          negdesc(i) = (180+alpha(am)); %3er cuarto de onda
44          negasc(i) = (360-alpha(am)); %4to cuarto de onda
45
46          dob(j) = d;
47          Vposdesc(j) = posdesc(i);
48          Vnegdesc(j) = negdesc(i);
49          Vnegasc(j) = negasc(i);
50          j = j+1;
51          dob(j) = d;
52          Vposdesc(j) = posdesc(i);
53          Vnegdesc(j) = negdesc(i);
54          Vnegasc(j) = negasc(i);
55          j = j+1;
56          am = am +1;
57          end
58          n = 0;
59          end
60
61          Vect_ang = [0 dob fliplr(Vposdesc) Vnegdesc fliplr(Vnegasc) 360];
62          %-------------------------------------------------------------%
63
64          angcc = zeros(1,(length(Vect_ang)*ciclo(1)));
65          esccc = zeros(1,(length(Vect_esc)*ciclo(1)));
66          %-----------PARA CINCO CICLOS-------------%
```

```
67        for i = 1:ciclo(1)
68        for j = 1:length(Vect_ang)
69        angcc(ac) = Vect_ang(j)+360*(i-1);
70        esccc(ac) = Vect_esc(j);
71        ac = ac +1;
72        end
73        end
74        %-----------------------------------------%
75
76        angbc = zeros(1,(length(Vect_ang)*ciclo(1)));
77        escbc = zeros(1,(length(Vect_esc)*ciclo(1)));
78
79        %----------PARA BASTANTES CICLOS----------%
80        for i = 1:ciclo(2)
81        for j = 1:length(Vect_ang)
82        angbc(ac) = Vect_ang(j)+360*(i-1);
83        escbc(ac) = Vect_esc(j);
84        ac = ac +1;
85        end
86        end
87        %-----------------------------------------%
88
89        %.....Para un ciclo.....%
90        Datos.tc = Vect_ang/(f*360);
91        Datos.esc = Vect_esc;
92        %.....Para 5 ciclos.....%
93        Datos.tcc = angcc/(f*360);
94        Datos.esccc = esccc;
95        %..Para muchos ciclos...%
96        Datos.tbc = angbc/(f*360);
97        Datos.escbc = escbc;
98        end
```

Anexo D

Función objetivo para el evento de contenido armónico

```
1        function eMAG = fun_eMAG_h(alpha)
2
3        load valuestruct.mat
4
5        eAmplitud = zeros(1,length(Amplitud));
6
7        y = 1;
8        A = 0;
9        acum = 0;
10       m = (k-1)/2;
11
12       for n = 1:2:hmax
13               if n == h(y)
14                       for i = 1:m
15                               for j = 1:L(i)
16                                       A = cosd(h(y)*alpha(i))*(-1)^(j
                                             -1) + A;
17                               end
18                       end
19                       i=0;
20                       if n == 1
21                               eAmplitud(y) = (A*((4*Vdc)/(pi*n)));
22                       else
23                               eAmplitud(y) = (A*((4*Vdc)/(pi*n)));
24                               errorp = abs(Amplitud(y)-(eAmplitud(y)/
                                     eAmplitud(1))*100);
25                               acum = acum + errorp;
```

```
26                          end
27                          y = y + 1;
28                          A = 0;
29                  end
30                  eMAG = acum;
31          end
```

Anexo E

Programación de la interfaz gráfica

```
1   function varargout = Interfaz(varargin)
2   %INTERFAZ MATLAB code file for Interfaz.fig
3   %      INTERFAZ, by itself, creates a new INTERFAZ or raises the
        existing
4   %      singleton*.
5   %
6   %      H = INTERFAZ returns the handle to a new INTERFAZ or the
        handle to
7   %      the existing singleton*.
8   %
9   %      INTERFAZ('Property','Value',...) creates a new INTERFAZ
        using the
10  %      given property value pairs. Unrecognized properties are
        passed via
11  %      varargin to Interfaz_OpeningFcn.  This calling syntax
        produces a
12  %      warning when there is an existing singleton*.
13  %
14  %      INTERFAZ('CALLBACK') and INTERFAZ('CALLBACK',hObject,...)
        call the
15  %      local function named CALLBACK in INTERFAZ.M with the given
         input
16  %      arguments.
17  %
18  %      *See GUI Options on GUIDE's Tools menu.  Choose "GUI
        allows only one
19  %      instance to run (singleton)".
20  %
21  % See also: GUIDE, GUIDATA, GUIHANDLES
22
23  % Edit the above text to modify the response to help Interfaz
```

```
24
25          % Last Modified by GUIDE v2.5 16-Jun-2020 15:00:01
26
27          % Begin initialization code - DO NOT EDIT
28          gui_Singleton = 1;
29          gui_State = struct('gui_Name',       mfilename, ...
30          'gui_Singleton',  gui_Singleton, ...
31          'gui_OpeningFcn', @Interfaz_OpeningFcn, ...
32          'gui_OutputFcn',  @Interfaz_OutputFcn, ...
33          'gui_LayoutFcn',  [], ...
34          'gui_Callback',   []);
35          if nargin && ischar(varargin{1})
36          gui_State.gui_Callback = str2func(varargin{1});
37          end
38
39          if nargout
40          [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
41          else
42          gui_mainfcn(gui_State, varargin{:});
43          end
44          % End initialization code - DO NOT EDIT
45
46
47          % --- Executes just before Interfaz is made visible.
48          function Interfaz_OpeningFcn(hObject, eventdata, handles,
                varargin)
49          % This function has no output args, see OutputFcn.
50          % hObject    handle to figure
51          % eventdata  reserved - to be defined in a future version of
                MATLAB
52          % handles    structure with handles and user data (see GUIDATA)
53          % varargin   unrecognized PropertyName/PropertyValue pairs from
                the
54          %            command line (see VARARGIN)
55
56          % Choose default command line output for Interfaz
57          handles.output = hObject;
58
59          % Update handles structure
```

```
60          guidata(hObject, handles);
61
62          % UIWAIT makes Interfaz wait for user response (see UIRESUME)
63          % uiwait(handles.figure1);
64
65
66          % --- Outputs from this function are returned to the command line
                .
67          function varargout = Interfaz_OutputFcn(hObject, eventdata,
                handles)
68          % varargout  cell array for returning output args (see VARARGOUT)
                ;
69          % hObject    handle to figure
70          % eventdata  reserved - to be defined in a future version of
                MATLAB
71          % handles    structure with handles and user data (see GUIDATA)
72
73          % Get default command line output from handles structure
74          varargout{1} = handles.output;
75
76
77
78          function edit3_Callback(hObject, eventdata, handles)
79          % hObject    handle to edit3 (see GCBO)
80          % eventdata  reserved - to be defined in a future version of
                MATLAB
81          % handles    structure with handles and user data (see GUIDATA)
82
83          % --- Executes during object creation, after setting all
                properties.
84          function edit3_CreateFcn(hObject, eventdata, handles)
85          % hObject    handle to edit3 (see GCBO)
86          % eventdata  reserved - to be defined in a future version of
                MATLAB
87          % handles    empty - handles not created until after all
                CreateFcns called
88
89          % Hint: edit controls usually have a white background on Windows.
90          %       See ISPC and COMPUTER.
```

```
91          if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
                defaultUicontrolBackgroundColor'))
92          set(hObject,'BackgroundColor','white');
93          end
94
95
96          %--- Executes on button press in pushParEv1.
97          function pushParEv1_Callback(hObject, eventdata, handles)
98
99          %---------------ESCOGER VALOR PU, F, DURACION SEGUN EVENTO
                ----------------%
100         contents = get(handles.popupChoice,'String');
101         popChoice = contents{get(handles.popupChoice,'Value')};
102         switch popChoice
103         case {'Depresion (Sag)'}
104         param.pu=round(get(handles.magnitud,'Value'),1);
105         sel = get(get(handles.uibuttongroupf,'SelectedObject'),'String');
106         if sel == '50 Hz'
107         f = 50;
108         min = 0.5;
109         max = 60*f;
110         set(handles.duracion,'SliderStep',[1,1]/(max-min));
111         duracion=round(get(handles.duracion,'Value'));
112         else
113         f = 60;
114         min = 0.5;
115         max = 60*f;
116         set(handles.duracion,'SliderStep',[1,1]/(max-min));
117         duracion=round(get(handles.duracion,'Value'));
118         end
119         case {'Elevacion (Swell)'}
120         param.pu=round(get(handles.magnitud2,'Value'),1);
121         sel = get(get(handles.uibuttongroupf,'SelectedObject'),'String');
122         if sel == '50 Hz'
123         f = 50;
124         min = 0.5;
125         max = 60*f;
126         set(handles.duracion,'SliderStep',[1,1]/(max-min));
127         duracion=round(get(handles.duracion,'Value'));
```

```
128         else
129         f = 60;
130         min = 0.5;
131         max = 60*f;
132         set(handles.duracion,'SliderStep',[1,1]/(max-min));
133         duracion=round(get(handles.duracion,'Value'));
134         end
135         case {'Bajovoltaje'}
136         param.pu=round(get(handles.magnitud,'Value'),1);
137         sel = get(get(handles.uibuttongroupf,'SelectedObject'),'String');
138         if sel == '50 Hz'
139         f = 50;
140         min = (60*f)+1;
141         max = 10800*f;
142         set(handles.duracion2,'SliderStep',[1,1]/(max-min));
143         duracion=round(get(handles.duracion2,'Value'));
144         else
145         f = 60;
146         min = (60*f)+1;
147         max = 10800*f;
148         set(handles.duracion2,'SliderStep',[1,1]/(max-min));
149         duracion=round(get(handles.duracion2,'Value'));
150         end
151         case {'Sobrevoltaje'}
152         param.pu=round(get(handles.magnitud2,'Value'),1);
153         sel = get(get(handles.uibuttongroupf,'SelectedObject'),'String');
154         if sel == '50 Hz'
155         f = 50;
156         min = (60*f)+1;
157         max = 10800*f;
158         set(handles.duracion2,'SliderStep',[1,1]/(max-min));
159         duracion=round(get(handles.duracion2,'Value'));
160         else
161         f = 60;
162         min = (60*f)+1;
163         max = 10800*f;
164         set(handles.duracion2,'SliderStep',[1,1]/(max-min));
165         duracion=round(get(handles.duracion2,'Value'));
166         end
```

```
167         case {'Interrupcion'}
168         param.pu=round(get(handles.magnitudint,'Value'),2);
169         sel = get(get(handles.uibuttongroupf,'SelectedObject'),'String');
170         if sel == '50 Hz'
171         f = 50;
172         duracion = str2double(get(handles.numslider,'String'));
173         else
174         f = 60;
175         duracion = str2double(get(handles.numslider,'String'));
176         end
177         case {'Variacion de frecuencia'}
178         param.pu=round(get(handles.variacionf,'Value'),2);
179         sel = get(get(handles.uibuttongroupf,'SelectedObject'),'String');
180         if sel == '50 Hz'
181         f = 50;
182         min = 0;
183         max = 10; %segundos
184         set(handles.duracionvar,'SliderStep',[1,1]/(max-min));
185         duracion=round(get(handles.duracionvar,'Value'));
186         else
187         f = 60;
188         min = 0;
189         max = 10; %segundos
190         set(handles.duracionvar,'SliderStep',[1,1]/(max-min));
191         duracion=round(get(handles.duracionvar,'Value'));
192         end
193         case {'Armonicos'}
194         sel = get(get(handles.uibuttongroupf,'SelectedObject'),'String');
195         if sel == '50 Hz'
196         f = 50;
197         else
198         f = 60;
199         end
200         end
201         %
              ------------------------------------------------------------------------

202
203         contents = get(handles.popupChoice,'String');
```

```
204         popChoice = contents{get(handles.popupChoice,'Value')};
205         switch popChoice
206         case {'Depresion (Sag)','Elevacion (Swell)','Bajovoltaje','
                Sobrevoltaje', 'Interrupcion'}
207         Vb = str2double(char(get(handles.Vb,'String')));
208
209         param.VRMS = Vb; %Voltaje RMS nominal dado por el usuario
210         param.k = str2double(char(get(handles.k,'String'))); %Niveles del
                conv.
211         param.Vdc = str2double(char(get(handles.Vdc,'String'))); %Vdc
212         param.hmax = str2double(char(get(handles.hmax,'String'))); %
                Numero max. de armonicos
213         param.L = str2num(char(get(handles.L,'String'))); %Vector de
                angulos por escalon
214         param.f = f;
215         param.duracion = duracion;
216         param.popChoice = popChoice;
217
218         save('valuestruct.mat','-struct','param');
219
220         f = waitbar(0,'Please wait...');
221         pause(.5)
222
223         %BOTON APLICAR, PSO PRINCIPAL PARA VRMS NORMAL
224
225         % Definicion del problema
226         acum = 0;
227         problem.costfunction = @(alpha,Vdc) fun_eVRMS(alpha,param.Vdc);
228         m=str2num(char(get(handles.k,'String')));
229         tic
230         m=(m-1)/2;
231
232         set(handles.nesc,'String',m);
233
234         for i = 1:length(param.L)
235         acum = acum+param.L(i);
236         end
237         problem.nvar = acum;
238
```

```
239         problem.varmin = 0;
240         problem.varmax = 90;
241
242         % Parametros para el algoritmo PSO
243         kappa=1;
244         phi1=2.05;
245         phi2=2.05;
246         phi=phi1+phi2;
247         chi=2*kappa/abs(2-phi-sqrt(phi^2-4*phi));
248         params.maxit = 100;
249         params.npop = 100;
250         params.w = chi;
251         params.wdamp=1;
252         params.c1 = chi*phi1;
253         params.c2 = chi*phi2;
254         params.showit=true;
255
256         % Llamar PSO
257         out= PSO(problem,params);
258         textlabel=out.textlabel;
259         set(handles.run, 'String',textlabel);
260         b=out.b;
261         VRMS_calc=fun_VRMS(b.position,param.Vdc);
262         alpha=b.position;
263
264         save('alpha.mat','alpha')
265
266         bestcosts=out.bestcosts;
267
268         set(handles.Vrms_calc,'String',VRMS_calc);
269         set(handles.ang,'String',alpha);
270
271         waitbar(.33,f,'Searching␣the␣best␣cost');
272         pause(1)
273
274         %BOTON APLICAR, PSO PRINCIPAL PARA VRMS DE LOS 5 PRIMEROS EVENTOS
275         param.VRMS = param.pu*Vb; %Voltaje RMS dado para el evento
276         param.Vdc1 = param.Vdc*param.pu;
277         save('valuestruct.mat','VRMS','Vdc1','-append','-struct','param')
```

```
278
279         problem.costfunction = @(alpha,Vdc1) fun_eVRMS(alpha,param.Vdc1);
280
281         % Llamar PSO
282         out= PSO(problem,params);
283         textlabel=out.textlabel;
284         set(handles.runev, 'String',textlabel);
285         b=out.b;
286         VRMS_calc=fun_VRMS(b.position,param.Vdc1);
287         alphaev=b.position;
288
289         save('alphaev.mat','alphaev')
290
291         bestcosts=out.bestcosts;
292
293         %set(handles.run, 'String',y);
294         set(handles.Vrms_calcev,'String',VRMS_calc);
295         set(handles.angev,'String',alphaev);
296
297         waitbar(.67,f,'Searching␣the␣best␣cost');
298         pause(1)
299
300         waitbar(1,f,'Finishing');
301         pause(1)
302
303         close(f)
304
305         case {'Variacion␣de␣frecuencia'}
306         %BOTON APLICAR, PSO PRINCIPAL PARA VRMS NORMAL
307         Vb = str2double(char(get(handles.Vb,'String')));
308
309         param.VRMS = Vb; %Voltaje RMS nominal dado por el usuario
310         param.k = str2double(char(get(handles.k,'String'))); %Niveles del
                conv.
311         param.Vdc = str2double(char(get(handles.Vdc,'String'))); %Vdc
312         param.hmax = str2double(char(get(handles.hmax,'String'))); %
              Numero max. de armonicos
313         param.L = str2num(char(get(handles.L,'String'))); %Vector de
              angulos por escalon
```

```
314         param.f = f;
315         param.duracion = duracion;
316         param.popChoice = popChoice;
317 
318         save('valuestruct.mat','-struct','param');
319 
320         f = waitbar(0,'Please␣wait...');
321         pause(.5)
322 
323         %Definicion del problema
324         acum = 0;
325         problem.costfunction = @(alpha,Vdc) fun_eVRMS(alpha,param.Vdc);
326         m=str2num(char(get(handles.k,'String')));
327         tic
328         m=(m-1)/2;
329 
330         set(handles.nesc,'String',m);
331 
332         for i = 1:length(param.L)
333         acum = acum+param.L(i);
334         end
335         problem.nvar = acum;
336 
337         problem.varmin = 0;
338         problem.varmax = 90;
339 
340         %Parametros para el algoritmo PSO
341         kappa=1;
342         phi1=2.05;
343         phi2=2.05;
344         phi=phi1+phi2;
345         chi=2*kappa/abs(2-phi-sqrt(phi^2-4*phi));
346         params.maxit = 100;
347         params.npop = 100;
348         params.w = chi;
349         params.wdamp=1;
350         params.c1 = chi*phi1;
351         params.c2 = chi*phi2;
352         params.showit=true;
```

```
353
354         % Llamar PSO
355         out= PSO(problem,params);
356         textlabel=out.textlabel;
357         set(handles.run, 'String',textlabel);
358         b=out.b;
359         VRMS_calc=fun_VRMS(b.position,param.Vdc);
360         alpha=b.position;
361
362         save('alpha.mat','alpha')
363
364         bestcosts=out.bestcosts;
365
366         set(handles.Vrms_calc,'String',VRMS_calc);
367         set(handles.ang,'String',alpha);
368
369         waitbar(.33,f,'Searching␣the␣best␣cost');
370         pause(1)
371
372         waitbar(1,f,'Finishing');
373         pause(1)
374
375         close(f)
376
377         case {'Armonicos'}
378         Vb = str2double(char(get(handles.Vb,'String')));
379
380         param.VRMS = Vb; %Voltaje RMS nominal dado por el usuario
381         param.k = str2double(char(get(handles.k,'String'))); %Niveles del
                conv.
382         param.Vdc = str2double(char(get(handles.Vdc,'String'))); %Vdc
383         param.hmax = str2double(char(get(handles.hmax,'String'))); %
              Numero max. de armonicos
384         param.L = str2num(char(get(handles.L,'String'))); %Vector de
              angulos por escalon
385         param.h = str2num(char(get(handles.ordenh,'String'))); %Vector de
                orden de armonicos
386         param.Amplitud = str2num(char(get(handles.magnitudh,'String')));
              %Vector de magnitud de armonicos
```

```
387         param.f = f;
388         param.popChoice = popChoice;
389 
390         save('valuestruct.mat','-struct','param');
391 
392         f = waitbar(0,'Please␣wait...');
393         pause(.5)
394 
395         %BOTON APLICAR, PSO PRINCIPAL PARA VRMS NORMAL
396 
397         % Definicion del problema
398         acum = 0;
399         problem.costfunction = @(alpha) fun_eMAG_h(alpha);
400         m=str2num(char(get(handles.k,'String')));
401         tic
402         m=(m-1)/2;
403 
404         set(handles.nesc,'String',m);
405 
406         for i = 1:length(param.L)
407         acum = acum+param.L(i);
408         end
409         problem.nvar = acum;
410 
411         problem.varmin = 0;
412         problem.varmax = 90;
413 
414         % Parametros para el algoritmo PSO
415         kappa=1;
416         phi1=2.05;
417         phi2=2.05;
418         phi=phi1+phi2;
419         chi=2*kappa/abs(2-phi-sqrt(phi^2-4*phi));
420         params.maxit = 100;
421         params.npop = 100;
422         params.w = chi;
423         params.wdamp=1;
424         params.c1 = chi*phi1;
425         params.c2 = chi*phi2;
```

```
426         params.showit=true;
427
428         % Llamar PSO
429         out= PSO(problem,params);
430         textlabel=out.textlabel;
431         set(handles.runh, 'String',textlabel);
432         b=out.b;
433         [MAG_calc]=fun_MAG_h(b.position);
434         alpha=b.position;
435
436         save('alpha.mat','alpha')
437
438         bestcosts=out.bestcosts;
439
440         %           set(handles.Mag_calc,'String',VRMS_calc);
441         %           set(handles.ang,'String',alpha);
442
443         waitbar(.33,f,'Searching␣the␣best␣cost');
444         pause(1)
445
446         waitbar(1,f,'Finishing');
447         pause(1)
448
449         close(f)
450         end
451
452         %--- Executes on button press in radiobutton60.
453         function radiobutton60_Callback(hObject, eventdata, handles)
454         % hObject    handle to radiobutton60 (see GCBO)
455         % eventdata  reserved - to be defined in a future version of
              MATLAB
456         % handles    structure with handles and user data (see GUIDATA)
457
458         % Hint: get(hObject,'Value') returns toggle state of
              radiobutton60
459
460
461         %--- Executes on button press in radiobutton50.
462         function radiobutton50_Callback(hObject, eventdata, handles)
```

```
463         % hObject    handle to radiobutton50 (see GCBO)
464         % eventdata  reserved - to be defined in a future version of
                MATLAB
465         % handles    structure with handles and user data (see GUIDATA)
466
467         % Hint: get(hObject,'Value') returns toggle state of
                radiobutton50
468
469         % --- Executes during object creation, after setting all
                properties.
470         function popupChoice_CreateFcn(hObject, eventdata, handles)
471         if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
                defaultUicontrolBackgroundColor'))
472         set(hObject,'BackgroundColor','white');
473         end
474
475         % --- Executes on selection change in popupChoice.
476         function popupChoice_Callback(hObject, eventdata, handles)
477         contents = cellstr(get(hObject,'String'));
478         popChoice = contents(get(hObject,'Value'));
479         if (strcmp(popChoice,'------------ Eventos a seleccionar
                ----------'))
480         set(handles.none,'visible','on');
481         elseif (strcmp(popChoice,'Depresion (Sag)'))
482         %---------Definiciones----------------%
483         set(handles.sag,'visible','on');
484         set(handles.none,'visible','off');
485         set(handles.swell,'visible','off');
486         set(handles.bajovoltaje,'visible','off');
487         set(handles.sobrevoltaje,'visible','off');
488         set(handles.flicker,'visible','off');
489         set(handles.varf,'visible','off');
490         set(handles.interrupcion,'visible','off');
491         set(handles.desv,'visible','off');
492         set(handles.muesca,'visible','off');
493         set(handles.offcd,'visible','off');
494         set(handles.armonicos,'visible','off');
495         %--------Panel de setting-------------%
496         %Setting pu
```

```
497         min = 0.1; max = 0.9;
498         set(handles.inicialpu,'String',min);
499         set(handles.finalpu,'String',max);
500         %-----------------------------------%
501         set(handles.duracion,'visible','on');
502         set(handles.magnitud,'visible','on');
503         set(handles.viewevs, 'visible','on');
504         set(handles.graficas,'visible','on');
505         set(handles.duracion2,'visible','off');
506         set(handles.numslider,'visible','off');
507         set(handles.magnitud2,'visible','off');
508         set(handles.magnitudint,'visible','off');
509         set(handles.msn,'visible','off');
510         set(handles.paramvarf,'visible','off');
511         set(handles.paramarm,'visible','off');
512         set(handles.grafarm,'visible','off');
513         set(handles.paramflicker,'visible','off');
514         elseif (strcmp(popChoice,'Elevacion (Swell)'))
515         %----------Definiciones-----------------%
516         set(handles.swell,'visible','on');
517         set(handles.none,'visible','off');
518         set(handles.sag,'visible','off');
519         set(handles.bajovoltaje,'visible','off');
520         set(handles.sobrevoltaje,'visible','off');
521         set(handles.flicker,'visible','off');
522         set(handles.varf,'visible','off');
523         set(handles.interrupcion,'visible','off');
524         set(handles.desv,'visible','off');
525         set(handles.muesca,'visible','off');
526         set(handles.offcd,'visible','off');
527         set(handles.armonicos,'visible','off');
528         %--------Panel de setting-------------%
529         %Setting pu
530         min = 1.1; max = 1.8;
531         set(handles.inicialpu,'String',min);
532         set(handles.finalpu,'String',max);
533         %-----------------------------------%
534         set(handles.duracion,'visible','on');
535         set(handles.magnitud2,'visible','on');
```

```
536         set(handles.viewevs, 'visible','on');
537         set(handles.graficas,'visible','on');
538         set(handles.duracion2,'visible','off');
539         set(handles.numslider,'visible','off');
540         set(handles.magnitud,'visible','off');
541         set(handles.magnitudint,'visible','off');
542         set(handles.msn,'visible','off');
543         set(handles.paramvarf,'visible','off');
544         set(handles.paramarm,'visible','off');
545         set(handles.grafarm,'visible','off');
546         set(handles.paramflicker,'visible','off');
547         elseif (strcmp(popChoice,'Bajovoltaje'))
548         %---------Definiciones----------------%
549         set(handles.bajovoltaje,'visible','on');
550         set(handles.none,'visible','off');
551         set(handles.sag,'visible','off');
552         set(handles.swell,'visible','off');
553         set(handles.sobrevoltaje,'visible','off');
554         set(handles.flicker,'visible','off');
555         set(handles.varf,'visible','off');
556         set(handles.interrupcion,'visible','off');
557         set(handles.desv,'visible','off');
558         set(handles.muesca,'visible','off');
559         set(handles.offcd,'visible','off');
560         set(handles.armonicos,'visible','off');
561         %--------Panel de setting-------------%
562         %Setting pu
563         min = 0.1; max = 0.9;
564         set(handles.inicialpu,'String',min);
565         set(handles.finalpu,'String',max);
566         %----------------------------------%
567         set(handles.duracion2,'visible','on');
568         set(handles.numslider,'visible','on');
569         set(handles.magnitud,'visible','on');
570         set(handles.viewevs, 'visible','on');
571         set(handles.graficas,'visible','on');
572         set(handles.duracion,'visible','off');
573         set(handles.magnitud2,'visible','off');
574         set(handles.magnitudint,'visible','off');
```

```
575         set(handles.msn,'visible','off');
576         set(handles.paramvarf,'visible','off');
577         set(handles.grafarm,'visible','off');
578         set(handles.paramflicker,'visible','off');
579         elseif (strcmp(popChoice,'Sobrevoltaje'))
580         %---------Definiciones----------------%
581         set(handles.sobrevoltaje,'visible','on');
582         set(handles.none,'visible','off');
583         set(handles.sag,'visible','off');
584         set(handles.swell,'visible','off');
585         set(handles.bajovoltaje,'visible','off');
586         set(handles.flicker,'visible','off');
587         set(handles.varf,'visible','off');
588         set(handles.interrupcion,'visible','off');
589         set(handles.desv,'visible','off');
590         set(handles.muesca,'visible','off');
591         set(handles.offcd,'visible','off');
592         set(handles.armonicos,'visible','off');
593         %--------Panel de setting-------------%
594         %Setting pu
595         min = 1.1; max = 1.8;
596         set(handles.inicialpu,'String',min);
597         set(handles.finalpu,'String',max);
598         %----------------------------------%
599         set(handles.duracion2,'visible','on');
600         set(handles.numslider,'visible','on');
601         set(handles.magnitud2,'visible','on');
602         set(handles.viewevs, 'visible','on');
603         set(handles.graficas,'visible','on');
604         set(handles.duracion,'visible','off');
605         set(handles.magnitud,'visible','off');
606         set(handles.paramflicker,'visible','off');
607         set(handles.magnitudint,'visible','off');
608         set(handles.msn,'visible','off');
609         set(handles.paramvarf,'visible','off');
610         set(handles.paramarm,'visible','off');
611         set(handles.grafarm,'visible','off');
612         elseif (strcmp(popChoice,'Interrupcion'))
613         %---------Definiciones----------------%
```

```
614         set(handles.interrupcion,'visible','on');
615         set(handles.none,'visible','off');
616         set(handles.sag,'visible','off');
617         set(handles.swell,'visible','off');
618         set(handles.bajovoltaje,'visible','off');
619         set(handles.sobrevoltaje,'visible','off');
620         set(handles.flicker,'visible','off');
621         set(handles.varf,'visible','off');
622         set(handles.desv,'visible','off');
623         set(handles.muesca,'visible','off');
624         set(handles.offcd,'visible','off');
625         set(handles.armonicos,'visible','off');
626         %--------Panel de setting-------------%
627         %Setting pu
628         min = 0; max = 0.1;
629         set(handles.inicialpu,'String',min);
630         set(handles.finalpu,'String',max);
631         %-----------------------------------%
632         set(handles.magnitudint,'visible','on');
633         set(handles.msn,'visible','on');
634         set(handles.numslider,'visible','on');
635         set(handles.viewevs, 'visible','on');
636         set(handles.graficas,'visible','on');
637         set(handles.duracion,'visible','off');
638         set(handles.paramflicker,'visible','off');
639         set(handles.magnitud2,'visible','off');
640         set(handles.duracion2,'visible','off');
641         set(handles.magnitud,'visible','off');
642         set(handles.paramvarf,'visible','off');
643         set(handles.paramarm,'visible','off');
644         set(handles.grafarm,'visible','off');
645         elseif (strcmp(popChoice,'Variacion de frecuencia'))
646         %-------------Definiciones--------------------%
647         set(handles.varf,'visible','on');
648         set(handles.none,'visible','off');
649         set(handles.sag,'visible','off');
650         set(handles.swell,'visible','off');
651         set(handles.bajovoltaje,'visible','off');
652         set(handles.sobrevoltaje,'visible','off');
```

```
653         set(handles.flicker,'visible','off');
654         set(handles.interrupcion,'visible','off');
655         set(handles.desv,'visible','off');
656         set(handles.muesca,'visible','off');
657         set(handles.offcd,'visible','off');
658         set(handles.armonicos,'visible','off');
659         %--------Panel de setting-------------%
660         set(handles.paramvarf,'visible','on');
661         set(handles.graficas,'visible','on');
662         set(handles.paramflicker,'visible','off');
663         set(handles.paramarm,'visible','off');
664         set(handles.duracion2,'visible','off');
665         set(handles.numslider,'visible','off');
666         set(handles.magnitud2,'visible','on');
667         set(handles.duracion,'visible','off');
668         set(handles.magnitud,'visible','off');
669         set(handles.magnitudint,'visible','off');
670         set(handles.msn,'visible','off');
671         set(handles.viewevs, 'visible','off');
672         set(handles.grafarm,'visible','off');
673         elseif (strcmp(popChoice,'Armonicos'))
674         %---------Definiciones----------------%
675         set(handles.armonicos,'visible','on');
676         set(handles.none,'visible','off');
677         set(handles.sag,'visible','off');
678         set(handles.swell,'visible','off');
679         set(handles.bajovoltaje,'visible','off');
680         set(handles.sobrevoltaje,'visible','off');
681         set(handles.flicker,'visible','off');
682         set(handles.varf,'visible','off');
683         set(handles.interrupcion,'visible','off');
684         set(handles.desv,'visible','off');
685         set(handles.muesca,'visible','off');
686         set(handles.offcd,'visible','off');
687         %--------Panel de setting-------------%
688         set(handles.paramarm,'visible','on');
689         set(handles.grafarm,'visible','on');
690         set(handles.paramflicker,'visible','off');
691         set(handles.duracion2,'visible','off');
```

```
692         set(handles.numslider,'visible','off');
693         set(handles.magnitud2,'visible','on');
694         set(handles.duracion,'visible','off');
695         set(handles.magnitud,'visible','off');
696         set(handles.magnitudint,'visible','off');
697         set(handles.msn,'visible','off');
698         set(handles.paramvarf,'visible','off');
699         set(handles.viewevs, 'visible','off');
700         set(handles.graficas,'visible','off');
701         end
702
703
704         function k_Callback(hObject, eventdata, handles)
705
706
707         %--- Executes during object creation, after setting all
                properties.
708         function k_CreateFcn(hObject, eventdata, handles)
709         % hObject    handle to k (see GCBO)
710         % eventdata  reserved - to be defined in a future version of
                MATLAB
711         % handles    empty - handles not created until after all
                CreateFcns called
712
713         % Hint: edit controls usually have a white background on Windows.
714         %       See ISPC and COMPUTER.
715         if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
                defaultUicontrolBackgroundColor'))
716         set(hObject,'BackgroundColor','white');
717         end
718
719
720         function Vdc_Callback(hObject, eventdata, handles)
721
722
723         %--- Executes during object creation, after setting all
                properties.
724         function Vdc_CreateFcn(hObject, eventdata, handles)
725         % hObject    handle to Vdc (see GCBO)
```

```
726         % eventdata  reserved - to be defined in a future version of
                MATLAB
727         % handles    empty - handles not created until after all
                CreateFcns called
728
729         % Hint: edit controls usually have a white background on Windows.
730         %       See ISPC and COMPUTER.
731         if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
                defaultUicontrolBackgroundColor'))
732         set(hObject,'BackgroundColor','white');
733         end
734
735         function hmax_Callback(hObject, eventdata, handles)
736
737         % --- Executes during object creation, after setting all
                properties.
738         function hmax_CreateFcn(hObject, eventdata, handles)
739         % hObject    handle to hmax (see GCBO)
740         % eventdata  reserved - to be defined in a future version of
                MATLAB
741         % handles    empty - handles not created until after all
                CreateFcns called
742
743         % Hint: edit controls usually have a white background on Windows.
744         %       See ISPC and COMPUTER.
745         if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
                defaultUicontrolBackgroundColor'))
746         set(hObject,'BackgroundColor','white');
747         end
748
749         function L_Callback(hObject, eventdata, handles)
750
751         % --- Executes during object creation, after setting all
                properties.
752         function L_CreateFcn(hObject, eventdata, handles)
753         % hObject    handle to L (see GCBO)
754         % eventdata  reserved - to be defined in a future version of
                MATLAB
755         % handles    empty - handles not created until after all
```

```
    CreateFcns called
756
757 % Hint: edit controls usually have a white background on Windows.
758 %       See ISPC and COMPUTER.
759 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
760 set(hObject,'BackgroundColor','white');
761 end
762
763 % --- Executes on button press in pushParConv.
764 function pushParConv_Callback(hObject, eventdata, handles)
765 % hObject    handle to pushParConv (see GCBO)
766 % eventdata  reserved - to be defined in a future version of
    MATLAB
767 % handles    structure with handles and user data (see GUIDATA)
768
769 % --- STATIC TEXT MAGNITUD PU (SAG and UNDER)
770 function magnitud_Callback(hObject, eventdata, handles)
771 contents = get(handles.popupChoice,'String');
772 popChoice = contents{get(handles.popupChoice,'Value')};
773 switch popChoice
774 case {'Depresion␣(Sag)','Bajovoltaje'}
775 min =0.1; max = 0.9;
776 set(handles.magnitud,'Max',max,'Min',min);
777 end
778
779 pu=round(get(handles.magnitud,'Value'),1);
780 set(handles.vermagnitud,'String',pu);
781
782 % --- Executes during object creation, after setting all
    properties.
783 function magnitud_CreateFcn(hObject, eventdata, handles)
784 % hObject    handle to magnitud (see GCBO)
785 % eventdata  reserved - to be defined in a future version of
    MATLAB
786 % handles    empty - handles not created until after all
    CreateFcns called
787
788 % Hint: slider controls usually have a light gray background.
```

```
789         if isequal(get(hObject,'BackgroundColor'), get(0,'
                defaultUicontrolBackgroundColor'))
790         set(hObject,'BackgroundColor',[.9 .9 .9]);
791         end
792
793         % --- STATIC TEXT MAGNITUD PU (SWELL and OVER)
794         function magnitud2_Callback(hObject, eventdata, handles)
795         contents = get(handles.popupChoice,'String');
796         popChoice = contents{get(handles.popupChoice,'Value')};
797         switch popChoice
798         case {'Elevacion (Swell)','Sobrevoltaje'}
799         min =1.1; max = 1.8;
800         set(handles.magnitud2,'Max',max,'Min',min);
801         end
802
803         pu=round(get(handles.magnitud2,'Value'),1);
804         set(handles.vermagnitud,'String',pu);
805
806         % --- Executes during object creation, after setting all
                properties.
807         function magnitud2_CreateFcn(hObject, eventdata, handles)
808         % hObject    handle to magnitud2 (see GCBO)
809         % eventdata  reserved - to be defined in a future version of
                MATLAB
810         % handles    empty - handles not created until after all
                CreateFcns called
811
812         % Hint: slider controls usually have a light gray background.
813         if isequal(get(hObject,'BackgroundColor'), get(0,'
                defaultUicontrolBackgroundColor'))
814         set(hObject,'BackgroundColor',[.9 .9 .9]);
815         end
816
817
818         % --- SLIDER DE SAG AND SWELL (TIEMPO)
819         function duracion_Callback(hObject, eventdata, handles)
820         contents = get(handles.popupChoice,'String');
821         popChoice = contents{get(handles.popupChoice,'Value')};
822         switch popChoice
```

```
823         case {'Depresion␣(Sag)','Elevacion␣(Swell)'}
824         sel = get(get(handles.uibuttongroupf,'SelectedObject'),'String');
825         if sel == '50␣Hz'
826         f = 50;
827         min = 0.5;
828         max = 60*f;
829         set(handles.duracion,'SliderStep',[1,1]/(max-min));
830         duracion=round(get(handles.duracion,'Value'));
831         else
832         f = 60;
833         min = 0.5;
834         max = 60*f;
835         set(handles.duracion,'SliderStep',[1,1]/(max-min));
836         duracion=round(get(handles.duracion,'Value'));
837         end
838         end
839         set(handles.duracion,'Min',min);
840         set(handles.duracion,'Max',max);
841         set(handles.verduracion,'String',duracion);
842
843
844         %--- Executes during object creation, after setting all
                properties.
845         function duracion_CreateFcn(hObject, eventdata, handles)
846         % hObject    handle to duracion (see GCBO)
847         % eventdata  reserved - to be defined in a future version of
                MATLAB
848         % handles    empty - handles not created until after all
                CreateFcns called
849
850         % Hint: slider controls usually have a light gray background.
851         if isequal(get(hObject,'BackgroundColor'), get(0,'
                defaultUicontrolBackgroundColor'))
852         set(hObject,'BackgroundColor',[.9 .9 .9]);
853         end
854
855         %--- Executes on button press in pushGraficar.
856         function pushGraficar_Callback(hObject, eventdata, handles)
857
```

```
858         load valuestruct.mat
859         load('alpha.mat','alpha')
860         load('alphaev.mat','alphaev')
861
862         contents = get(handles.popupChoice,'String');
863         popChoice = contents{get(handles.popupChoice,'Value')};
864         switch popChoice
865         case {'Depresion (Sag)','Elevacion (Swell)','Bajovoltaje','
                Sobrevoltaje', 'Interrupcion'}
866         Vdc = str2double(char(get(handles.Vdc,'String'))); %Vdc
867
868         axes(handles.axes1);
869         Datos = Const_grafica(alpha);
870         Datos.vc = Datos.esc*Vdc;
871         Datos.vcc = Datos.esccc*Vdc;
872         Datos.vbc = Datos.escbc*Vdc;
873         plot(Datos.tc,Datos.vc);
874         Datosn.time = Datos.tcc';
875         Datosn.signals.values = Datos.vcc';
876         assignin('base','Datosn',Datosn)
877         Datosnb.time = Datos.tbc';
878         Datosnb.signals.values = Datos.vbc';
879         assignin('base','Datosnb',Datosnb)
880
881         %---Evento
882         axes(handles.axes2);
883         Datos = Const_grafica(alphaev);
884         Datos.vc = Datos.esc*Vdc1;
885         Datos.vcc = Datos.esccc*Vdc1;
886         Datos.vbc = Datos.escbc*Vdc1;
887         plot(Datos.tc,Datos.vc);
888         Datosev.time = Datos.tbc';
889         Datosev.signals.values = Datos.vbc';
890         assignin('base','Datosev',Datosev)
891
892         case {'Variacion de frecuencia'}
893         Vdc = str2double(char(get(handles.Vdc,'String'))); %Vdc
894
895         axes(handles.axes1);
```

```
896         Datos = Const_grafica(alpha);
897         Datos.vc = Datos.esc*Vdc;
898         Datos.vcc = Datos.esccc*Vdc;
899         Datos.vbc = Datos.escbc*Vdc;
900         plot(Datos.tc,Datos.vc);
901         Datosn.time = Datos.tcc';
902         Datosn.signals.values = Datos.vcc';
903         assignin('base','Datosn',Datosn)
904         Datosnb.time = Datos.tbc';
905         Datosnb.signals.values = Datos.vbc';
906         assignin('base','Datosnb',Datosnb)
907
908         %---Evento
909         param.f = pu*f; %Frecuencia nueva dada para el evento
910         save('valuestruct.mat','f','-append','-struct','param')
911         axes(handles.axes2);
912         Datos = Const_grafica(alpha);
913         Datos.vc = Datos.esc*Vdc;
914         Datos.vcc = Datos.esccc*Vdc;
915         Datos.vbc = Datos.escbc*Vdc;
916         plot(Datos.tc,Datos.vc);
917         Datosev.time = Datos.tbc';
918         Datosev.signals.values = Datos.vbc';
919         assignin('base','Datosev',Datosev)
920
921         case {'Armonicos'}
922         alpha = sort(alpha); %Ordenar los angulos de forma ascendente
923        m = (k-1)/2;
924         bn = espectrous(alpha,m,L,h,hmax);
925
926         axes(handles.grafh);
927         bar(bn,0.07);
928         title('Espectro␣armonico');
929         xlabel('Numero␣del␣armonico');
930         ylabel('Espectro␣en␣Magnitud␣pu␣con␣base␣Vdc');
931         grid off
932         axis([0 hmax 0 (max(bn)+10) ]);
933
934         Vdc = str2double(char(get(handles.Vdc,'String'))); %Vdc
```

```
935
936         %---Evento
937         Datos = Const_grafica(alpha);
938         %           Datos.vc = Datos.esc*Vdc;
939         %           Datosev.time = Datos.tc';
940         %           Datosev.signals.values = Datos.vc';
941         %           assignin('base','Datosev',Datosev)
942
943         Datos.vbc = Datos.escbc*Vdc;
944         Datosev.time = Datos.tbc';
945         Datosev.signals.values = Datos.vbc';
946         assignin('base','Datosev',Datosev)
947
948         end
949
950         function Vb_Callback(hObject, eventdata, handles)
951
952         %--- Executes during object creation, after setting all
                properties.
953         function Vb_CreateFcn(hObject, eventdata, handles)
954         % hObject    handle to Vb (see GCBO)
955         % eventdata  reserved - to be defined in a future version of
                MATLAB
956         % handles    empty - handles not created until after all
                CreateFcns called
957
958         % Hint: edit controls usually have a white background on Windows.
959         %       See ISPC and COMPUTER.
960         if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
                defaultUicontrolBackgroundColor'))
961         set(hObject,'BackgroundColor','white');
962         end
963
964         %--- Executes on button press in fiftyhz.
965         function fiftyhz_Callback(hObject, eventdata, handles)
966         % hObject    handle to fiftyhz (see GCBO)
967         % eventdata  reserved - to be defined in a future version of
                MATLAB
968         % handles    structure with handles and user data (see GUIDATA)
```

```
969
970         %--- Executes on button press in sixtyhz.
971         function sixtyhz_Callback(hObject, eventdata, handles)
972         % hObject    handle to sixtyhz (see GCBO)
973         % eventdata  reserved - to be defined in a future version of
                MATLAB
974         % handles    structure with handles and user data (see GUIDATA)
975
976         %--- Executes when selected object is changed in uibuttongroupf.
977         function uibuttongroupf_SelectionChangedFcn(hObject, eventdata,
                handles)
978         contents = get(handles.popupChoice,'String');
979         popChoice = contents{get(handles.popupChoice,'Value')};
980         switch popChoice
981         case {'Depresion (Sag)','Elevacion (Swell)'}
982         sel = get(get(handles.uibuttongroupf,'SelectedObject'),'String');
983         if sel == '50 Hz'
984         f = 50;
985         min = 0.5;
986         max = 60*f;
987         else
988         f = 60;
989         min = 0.5;
990         max = 60*f;
991         end
992         set(handles.inicial,'String',min);
993         set(handles.final,'String',max);
994         case {'Bajovoltaje','Sobrevoltaje'}
995         sel = get(get(handles.uibuttongroupf,'SelectedObject'),'String');
996         if sel == '50 Hz'
997         f = 50;
998         min = (60*f) + 1;
999         max = 10800*f;
1000        else
1001        f = 60;
1002        min = (60*f) + 1;
1003        max = 10800*f;
1004        end
1005        set(handles.inicial,'String',min);
```

```
1006        set(handles.final,'String',max);
1007        case {'Interrupcion'}
1008        sel = get(get(handles.uibuttongroupf,'SelectedObject'),'String');
1009        if sel == '50 Hz'
1010        f = 50;
1011        min = 0.5
1012        max = 10800*f;
1013        else
1014        f = 60;
1015        min = 0.5;
1016        max = 10800*f;
1017        end
1018        set(handles.inicial,'String',min);
1019        set(handles.final,'String',max);
1020        end
1021        %--- Executes on button press in radiobutton6.
1022        function radiobutton6_Callback(hObject, eventdata, handles)
1023        % hObject    handle to radiobutton6 (see GCBO)
1024        % eventdata  reserved - to be defined in a future version of
              MATLAB
1025        % handles    structure with handles and user data (see GUIDATA)
1026
1027        %--- Executes on button press in Simular.
1028        function Simular_Callback(hObject, eventdata, handles)
1029        % hObject    handle to Simular (see GCBO)
1030        % eventdata  reserved - to be defined in a future version of
              MATLAB
1031        % handles    structure with handles and user data (see GUIDATA)
1032        load valuestruct.mat
1033
1034        contents = get(handles.popupChoice,'String');
1035        popChoice = contents{get(handles.popupChoice,'Value')};
1036
1037        sel = get(get(handles.uibuttongroupf,'SelectedObject'),'String');
1038        if sel == '50 Hz'
1039        f1 = 50;
1040        else
1041        f1 = 60;
1042        end
```

```
1043
1044        switch popChoice
1045        case {'Depresion (Sag)','Elevacion (Swell)','Bajovoltaje','
                Sobrevoltaje', 'Interrupcion'}
1046        t2 = (5/f) + duracion/f;
1047        ts = t2 + (5/f) + (20/f);
1048
1049        assignin('base','f',f);
1050        assignin('base','t2',t2);
1051        assignin('base','ts',ts);
1052
1053        case {'Variacion de frecuencia'}
1054        t2 = (5/f1) + duracion;
1055        ts = t2 + (5/f1) + (20/f1);
1056
1057        assignin('base','f',f);
1058        assignin('base','f1',f1);
1059        assignin('base','t2',t2);
1060        assignin('base','ts',ts);
1061        end
1062        % find_system('Name','Simulacion');
1063        % open_system('Simulacion');
1064        % set_param('Simulacion/t1','Before',5/f);
1065        % % set_param('Simulacion/t3','Value',f);
1066        % % set_param('Simulacion/t2','Time',tev);
1067        % % set_param('Simulacion/Datos normales','Value',Datosn);
1068        % % set_param('Simulacion/Datos del evento','Value',Datosev);
1069        % % set_param('Simulacion/Datos normal b','Value',Datosnb);
1070        % set_param('Simulacion','StopTime',ts);
1071        % set_param(gcs,'SimulationCommand','inicialpu');
1072
1073        %-----EDIT TEXT PARA OVER AND UNDER VOLTAJE (TIEMPO)
1074        function numslider_Callback(hObject, eventdata, handles)
1075        contents = get(handles.popupChoice,'String');
1076        popChoice = contents{get(handles.popupChoice,'Value')};
1077        switch popChoice
1078        case {'Bajovoltaje','Sobrevoltaje'}
1079        val = str2double(get(hObject,'String'));
1080        set(handles.duracion2,'Value',val);
```

```
1081        set(handles.verduracion,'String',val);
1082        case {'Interrupcion'}
1083        val = str2double(get(hObject,'String'));
1084        set(handles.verduracion,'String',val);
1085        end
1086        %guidata(hObject,handles)
1087
1088        %--- Executes during object creation, after setting all
                properties.
1089        function numslider_CreateFcn(hObject, eventdata, handles)
1090        % hObject    handle to numslider (see GCBO)
1091        % eventdata  reserved - to be defined in a future version of
                MATLAB
1092        % handles    empty - handles not created until after all
                CreateFcns called
1093
1094        % Hint: edit controls usually have a white background on Windows.
1095        %       See ISPC and COMPUTER.
1096        if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
                defaultUicontrolBackgroundColor'))
1097        set(hObject,'BackgroundColor','white');
1098        end
1099
1100
1101        %--- SLIDER DE OVER AND UNDERVOLTAGE (TIEMPO)
1102        function duracion2_Callback(hObject, eventdata, handles)
1103        val = round(get(hObject,'Value'));
1104        set(handles.numslider,'String',num2str(val));
1105        %guidata(hObject,handles)
1106        contents = get(handles.popupChoice,'String');
1107        popChoice = contents{get(handles.popupChoice,'Value')};
1108        switch popChoice
1109        case {'Bajovoltaje','Sobrevoltaje'}
1110        sel = get(get(handles.uibuttongroupf,'SelectedObject'),'String');
1111        if sel == '50 Hz'
1112        f = 50;
1113        min = (60*f)+1;
1114        max = 10800*f;
1115        set(handles.duracion2,'SliderStep',[1,1]/(max-min));
```

```
1116        duracion=round(get(handles.duracion2,'Value'));
1117        else
1118        f = 60;
1119        min = (60*f)+1;
1120        max = 10800*f;
1121        set(handles.duracion2,'SliderStep',[1,1]/(max-min));
1122        duracion=round(get(handles.duracion2,'Value'));
1123        end
1124        end
1125        set(handles.duracion2,'Max',max, 'Min',min);
1126        set(handles.verduracion,'String',duracion);
1127
1128        %--- Executes during object creation, after setting all
               properties.
1129        function duracion2_CreateFcn(hObject, eventdata, handles)
1130        % hObject    handle to duracion2 (see GCBO)
1131        % eventdata  reserved - to be defined in a future version of
               MATLAB
1132        % handles    empty - handles not created until after all
               CreateFcns called
1133
1134        % Hint: slider controls usually have a light gray background.
1135        if isequal(get(hObject,'BackgroundColor'), get(0,'
               defaultUicontrolBackgroundColor'))
1136        set(hObject,'BackgroundColor',[.9 .9 .9]);
1137        end
1138
1139
1140        %--- Executes on slider movement.
1141        function magnitudint_Callback(hObject, eventdata, handles)
1142        contents = get(handles.popupChoice,'String');
1143        popChoice = contents{get(handles.popupChoice,'Value')};
1144        switch popChoice
1145        case {'Interrupcion'}
1146        min =0; max = 0.1;
1147        set(handles.magnitudint,'SliderStep',[0.01,0.01]/(max-min));
1148        end
1149
1150        pu=round(get(handles.magnitudint,'Value'),2);
```

```
1151        set(handles.vermagnitud,'String',pu);
1152
1153        %--- Executes during object creation, after setting all
                properties.
1154        function magnitudint_CreateFcn(hObject, eventdata, handles)
1155        % hObject    handle to magnitudint (see GCBO)
1156        % eventdata  reserved - to be defined in a future version of
                MATLAB
1157        % handles    empty - handles not created until after all
                CreateFcns called
1158
1159        % Hint: slider controls usually have a light gray background.
1160        if isequal(get(hObject,'BackgroundColor'), get(0,'
                defaultUicontrolBackgroundColor'))
1161        set(hObject,'BackgroundColor',[.9 .9 .9]);
1162        end
1163
1164        function ordenh_Callback(hObject, eventdata, handles)
1165
1166
1167        %--- Executes during object creation, after setting all
                properties.
1168        function ordenh_CreateFcn(hObject, eventdata, handles)
1169
1170        if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
                defaultUicontrolBackgroundColor'))
1171        set(hObject,'BackgroundColor','white');
1172        end
1173
1174
1175        function magnitudh_Callback(hObject, eventdata, handles)
1176        % hObject    handle to magnitudh (see GCBO)
1177        % eventdata  reserved - to be defined in a future version of
                MATLAB
1178        % handles    structure with handles and user data (see GUIDATA)
1179
1180        % Hints: get(hObject,'String') returns contents of magnitudh as
                text
1181        %        str2double(get(hObject,'String')) returns contents of
```

```
        magnitudh as a double
1182
1183
1184    % --- Executes during object creation, after setting all
            properties.
1185    function magnitudh_CreateFcn(hObject, eventdata, handles)
1186    % hObject    handle to magnitudh (see GCBO)
1187    % eventdata  reserved - to be defined in a future version of
            MATLAB
1188    % handles    empty - handles not created until after all
            CreateFcns called
1189
1190    % Hint: edit controls usually have a white background on Windows.
1191    %       See ISPC and COMPUTER.
1192    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
            defaultUicontrolBackgroundColor'))
1193    set(hObject,'BackgroundColor','white');
1194    end
1195
1196
1197    % --- SLIDER DURACION DE LA VARIACION DE FRECUENCIA
1198    function duracionvar_Callback(hObject, eventdata, handles)
1199    min = 0;
1200    max = 10; %segundos
1201    set(handles.duracionvar,'SliderStep',[1,1]/(max-min));
1202    duracion=round(get(handles.duracionvar,'Value'));
1203    set(handles.duracionvar,'Max',max,'Min',min);
1204    set(handles.verdurvar,'String',duracion);
1205
1206    % --- Executes during object creation, after setting all
            properties.
1207    function duracionvar_CreateFcn(hObject, eventdata, handles)
1208    % hObject    handle to duracionvar (see GCBO)
1209    % eventdata  reserved - to be defined in a future version of
            MATLAB
1210    % handles    empty - handles not created until after all
            CreateFcns called
1211
1212    % Hint: slider controls usually have a light gray background.
```

```
1213        if isequal(get(hObject,'BackgroundColor'), get(0,'
                defaultUicontrolBackgroundColor'))
1214        set(hObject,'BackgroundColor',[.9 .9 .9]);
1215        end
1216
1217
1218        %--- SLIDER MAGNITUD VARIACION DE FRECUENCIA
1219        function variacionf_Callback(hObject, eventdata, handles)
1220        min =0; max = 4;
1221        set(handles.variacionf,'SliderStep',[0.01,0.01]/(max-min));
1222        set(handles.variacionf,'Max',max,'Min',min);
1223        pc=round(get(handles.variacionf,'Value'),2);
1224        set(handles.vervar,'String',pc);
1225
1226        %--- Executes during object creation, after setting all
                properties.
1227        function variacionf_CreateFcn(hObject, eventdata, handles)
1228        % hObject    handle to variacionf (see GCBO)
1229        % eventdata  reserved - to be defined in a future version of
               MATLAB
1230        % handles    empty - handles not created until after all
                CreateFcns called
1231
1232        % Hint: slider controls usually have a light gray background.
1233        if isequal(get(hObject,'BackgroundColor'), get(0,'
                defaultUicontrolBackgroundColor'))
1234        set(hObject,'BackgroundColor',[.9 .9 .9]);
1235        end
1236
1237
1238
1239        function edit27_Callback(hObject, eventdata, handles)
1240        % hObject    handle to edit27 (see GCBO)
1241        % eventdata  reserved - to be defined in a future version of
               MATLAB
1242        % handles    structure with handles and user data (see GUIDATA)
1243
1244        % Hints: get(hObject,'String') returns contents of edit27 as text
1245        %        str2double(get(hObject,'String')) returns contents of
```

```
%        edit27 as a double
1246
1247
1248    %--- Executes during object creation, after setting all
            properties.
1249    function edit27_CreateFcn(hObject, eventdata, handles)
1250    % hObject    handle to edit27 (see GCBO)
1251    % eventdata  reserved - to be defined in a future version of
            MATLAB
1252    % handles    empty - handles not created until after all
            CreateFcns called
1253
1254    % Hint: edit controls usually have a white background on Windows.
1255    %       See ISPC and COMPUTER.
1256    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
            defaultUicontrolBackgroundColor'))
1257    set(hObject,'BackgroundColor','white');
1258    end
1259
1260
1261
1262    function edit28_Callback(hObject, eventdata, handles)
1263    % hObject    handle to edit28 (see GCBO)
1264    % eventdata  reserved - to be defined in a future version of
            MATLAB
1265    % handles    structure with handles and user data (see GUIDATA)
1266
1267    % Hints: get(hObject,'String') returns contents of edit28 as text
1268    %        str2double(get(hObject,'String')) returns contents of
            edit28 as a double
1269
1270
1271    %--- Executes during object creation, after setting all
            properties.
1272    function edit28_CreateFcn(hObject, eventdata, handles)
1273    % hObject    handle to edit28 (see GCBO)
1274    % eventdata  reserved - to be defined in a future version of
            MATLAB
1275    % handles    empty - handles not created until after all
```

```
    CreateFcns called
1276
1277        % Hint: edit controls usually have a white background on Windows.
1278        %       See ISPC and COMPUTER.
1279        if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
                defaultUicontrolBackgroundColor'))
1280        set(hObject,'BackgroundColor','white');
1281        end
```