Universidad Zaragoza
1542

Trabajo Fin de Máster

RECONOCIMIENTO NO INTRUSIVO DE CARGAS ELÉCTRICAS DOMÉSTICAS MEDIANTE REDES NEURONALES PARA IMPLEMENTACIÓN EN DISPOSITIVOS DE BAJO CONSUMO

NON INTRUSIVE APPLIANCE RECOGNITION BY MEANS OF NEURAL NETWORKS FOR THE IMPLEMENTATION INTO LOW POWER DEVICES

Autor

Álvaro Valerio RAMÍREZ LADISLAO

Director

Roberto José CASAS NEBRA

ESCUELA DE INGENIERÍA Y ARQUITECTURA, UNIVERSIDAD DE ZARAGOZA.
AÑO 2016

RECONOCIMIENTO NO INTRUSIVO DE CARGAS ELÉCTRICAS DOMÉSTICAS MEDIANTE REDES NEURONALES PARA IMPLEMENTACIÓN EN DISPOSITIVOS DE BAJO CONSUMO

RESUMEN:

El presente trabajo aborda un enfoque novedoso a la hora de la identificación de cargas eléctricas de forma no invasiva, centrándose en cuantificar las prestaciones que una red neuronal artificial puede alcanzar para el caso de usar dispositivos computacionales de bajo consumo y medias prestaciones.

Basándose en la impronta que cualquier carga eléctrica fija en la corriente que consume se idea un sistema electrónico capaz de, mediante una red neuronal, clasificar dicha carga únicamente capturando la onda de corriente en la acometida de cualquier usuario.

Empezando con una adquisición detallada de los datos, se eligen aquellas características que se extraerán de la corriente así medida y se analizan las posibles técnicas de ahorro operacional y de adquisición de datos

Posteriormente se realizan comparativas entre diversos tipos de redes neuronales artificiales en función de sus requerimientos de memoria frente a sus resultados clasificatorios.

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Álvaro Valerio RAMÍREZ LADISLAO,

con nº de DNI 17744396B en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) MÁSTER EN INGENIERÍA ELECTRÓNICA, (Título del Trabajo) RECONOCIMIENTO NO INTRUSIVO DE CARGAS ELÉCTRICAS DOMÉSTICAS MEDIANTE REDES NEURONALES PARA IMPLEMENTACIÓN EN DISPOSITIVOS DE BAJO CONSUMO

, es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, a 25 de Octubre de 2016

TABLA DE CONTENIDOS

1. INTRODUCCIÓN

1.1. MOTIVACIONES

Desde siempre me ha interesado la producción, conversión y generación de energía eléctrica. Mi proyecto final de carrera versaba sobre cálculos de centrales hidroeléctricas, obteniendo el caudal óptimo instalado según el perfil del caudal disponible. Cuando tuve que elegir el trabajo fin de máster intenté profundizar sobre estos temas: energía, optimización y cálculos, aplicando los nuevos conocimientos y técnicas adquiridas.

Fruto de estas inquietudes surge este estudio que pretende avanzar un modesto paso más en el control de la energía.

1.2. OBJETO

El presente trabajo fin de máster parte de estudios y técnicas para el reconocimiento de cargas eléctricas mediante el análisis de la impronta que éstas generan sobre determinados parámetros eléctricos, medibles de forma no invasiva (por sus siglas en ingles N.I.L.M, *non intrusive load monitoring*). De esta manera es posible conocer las cargas conectadas a un nodo eléctrico, sin necesidad de intervenir aguas abajo de dicho nodo.

Esta información puede resultar útil tanto para el usuario final, que en cualquier momento puede saber el consumo no sólo de su hogar, sino también de cada dispositivo tanto en tiempo real como histórico, como para las compañías distribuidoras de energía eléctrica que podrían guiarse por la información así obtenida para mejorar el servicio (obviando distintos matices éticos referidos a la intimidad de las personas).

Como se verá más adelante, estos sistemas han sido ya desarrollados por muchos investigadores, con diversas implementaciones de los algoritmos de reconocimiento (sistemas estadísticos, redes neuronales, lógica difusa, etc.).

Todos tienen en común un esquema muy similar al de la figura 1

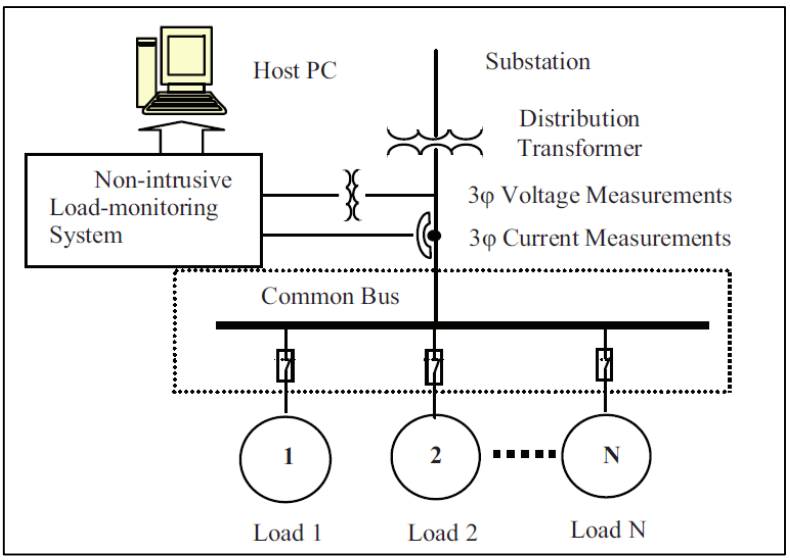

Fig. 1 Esquema genérico de un sistema N.I.L.M.

Estos sistemas aprovechan la gran capacidad y rapidez de los ordenadores personales para ejecutar los algoritmos de reconocimiento en cualquiera de sus versiones y obtener y almacenar los resultados.

Sin embargo, estos sistemas son voluminosos, caros y ellos mismos consumen una energía considerable; por ello resulta ventajoso que el sistema de reconocimiento sea pequeño, consuma poca energía y sea capaz de obtener resultados similares. Es en este matiz del bajo consumo en el que se centrará este estudio, por lo que los algoritmos y funcionamiento se orientarán hacia este objetivo: conseguir un sistema de adquisición, procesado y un algoritmo eficiente energéticamente hablando.

El acondicionamiento, ensayos y obtención de algoritmos se han realizado mediante scripts en MatLab, anexados a este documento.

TDC

2. ESTADO DE LA IDENTIFICACION DE CARGAS

Los sistemas NILM desde su concepción [HART 92] han tratado de obtener el consumo, y por tanto la potencia, desde la acometida o desde puntos de distribución, y no midiendo directamente en el electrodoméstico. Esto plantea el inconveniente de que la corriente que se puede medir en esto lugares centralizados es una suma de corrientes de todas las cargas allí conectadas. Discernir qué cargas están conectadas y qué porción de la corriente medida corresponde a cada cuál es el centro de las investigaciones en este campo.

En este punto se realizará un estudio de las diversas técnicas empleadas desde sus comienzos hasta nuestros días, haciendo especial hincapié en las técnicas relacionadas con las redes neuronales, y su contraposición a técnicas estadísticas o más clásicas y a algoritmos heurísticos.

El sistema de identificación de cargas (dado que no se mide cada carga individualmente, sino el conjunto de ellas) se sustenta en la teoría de que cada carga imprime en la corriente consumida o tensión soportada una “huella” que permite distinguirla del resto (disgregación). Es la búsqueda de esta huella única que permita la identificación, la que ha generado tan abundantes y variados sistemas.

2.1. MÉTODOS DE IDENTIFICACIÓN

El sistema de identificación propiamente dicho es el mecanismo por el cual al observar unas determinadas medidas, se obtiene cierta información de lo que está conectado aguas abajo del sistema de captación.

Esta información es extraída mediante diferentes métodos, que podemos clasificar en tres grupos estudiados en los siguientes puntos:

- Métodos heurísticos.
- Sistemas estadísticos.
- Redes Neuronales y lógica difusa.

2.1.1. MÉTODOS HEURÍSTICOS

Se basan principalmente en la investigación del funcionamiento de ciertas cargas, cómo se comportan, y cómo son usadas normalmente. Suelen usar reglas lógicas de uso y funcionamiento. Debido al estudio del modo de funcionamiento, el resultado de los algoritmos no es en tiempo real, sino que necesita de cierto tiempo de observación del funcionamiento. La frecuencia de muestreo suele ser baja. Usan pocos parámetros para la identificación.

[HART 92] propone el uso de un plano P-Q (potencia activa-potencia reactiva respectivamente; figura 2).

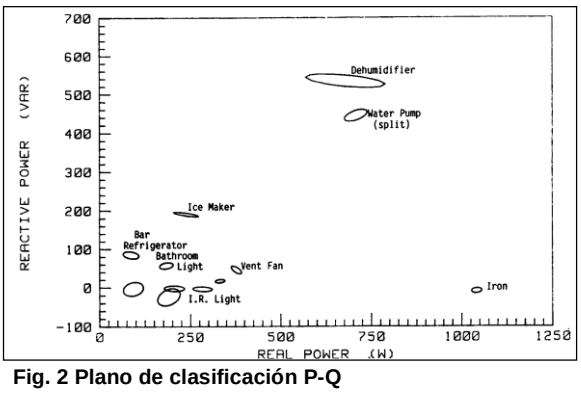

Fig. 2 Plano de clasificación P-Q

Y asemeja las cargas a MEF (Maquinas de Estados Finitos, figura 3) que siguen unas reglas

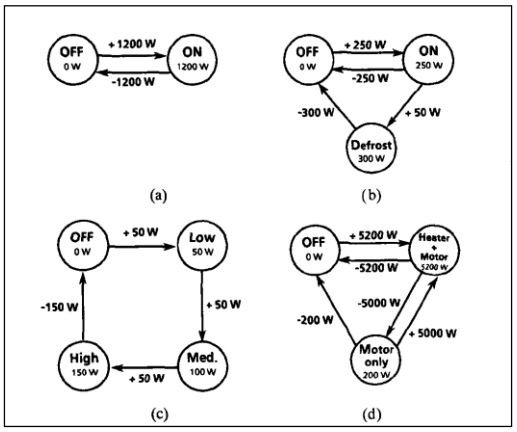

Fig. 3 MEF de algunas cargas.

De esta forma se hacen corresponder los cambios producidos en la potencia con los estados de la máquina. (50W+50W+50W-150W corresponde a la carga “c”)

[NORFORD 96] incluye el transitorio a ON como signatura a tener en cuenta en la clasificación, y propone usar las órdenes de los buses de comunicación de la automatización, si la hubiere, para conocer qué aparato se va a conectar.

Así mismo, el sistema tolera cierto solapamiento (simultaneidad) de transitorios (figura 4)

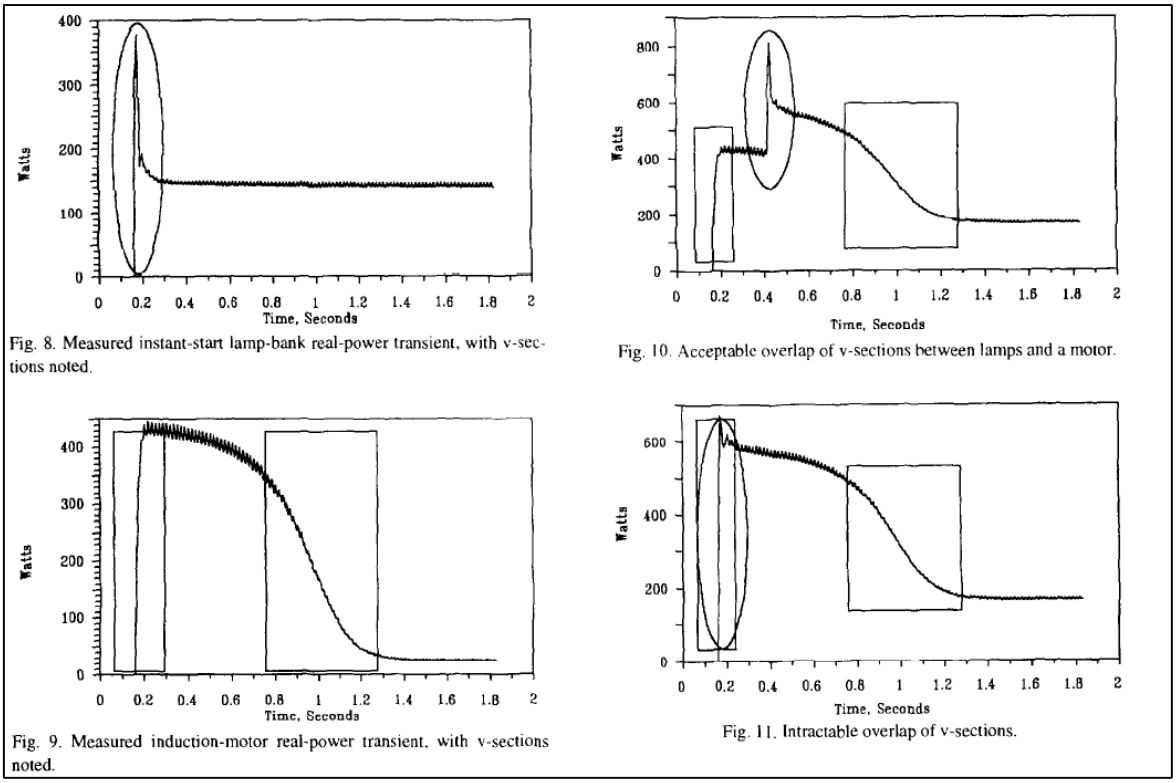

Fig. 4 Solapamiento en los transitorios

Este sistema no es capaz de solucionar el problema de cargas lentamente variables. Para solucionarlo propone medir los fenómenos responsables de ésta variación en la carga (temperatura, presión, caudal…).

Mejorando el sistema de reconocimiento basado en el régimen permanente y para separar más las clases que se identifiquen, [LAUGHMAN 03] amplia el plano P-Q a un espacio tridimensional, figura 5 con la inclusión de una dimensión relacionada con el contenido armónico.

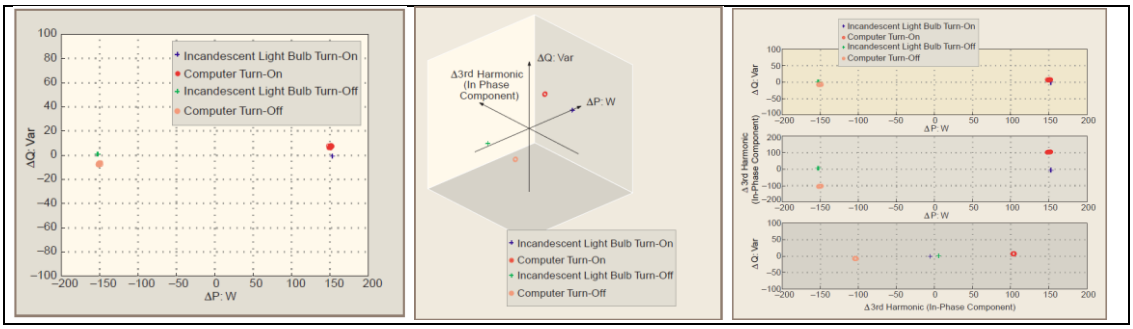

Fig. 5 Ampliación del plano P-Q a espacio P-Q-D

[GILREATH 06] resuelve una técnica para cuantificar los armónicos que producen cargas no lineales, evitando la transformada de Fourier, aliviando los cálculos y memoria necesarios. Para ello usa una modificación de la transformada de Concordia, en la que el centroide (figura 6) se desplaza del origen.

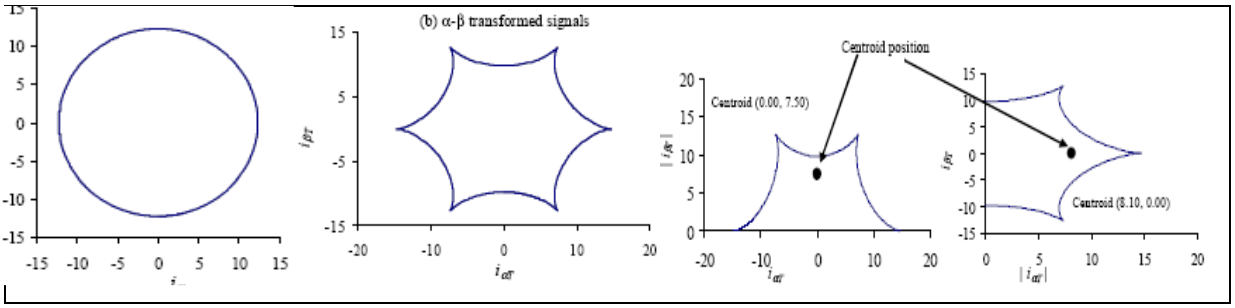

Fig. 6 Resultado de la modificación de la trasformada de Concordia.

Este sistema no es un clasificador propiamente dicho, pero se incluye en este estudio debido a que podría ser un complemento de los sistemas no intrusivos.

En esta tabla 1 se consigna un resumen de los métodos expuestos:

	Signaturas	Régimen de análisis	Distinción cargas múltiples	Distinción cargas conexión simultánea	Distinción cargas lentamente variables	Modo Identificación
[HART 92]:	Admitancia, Armónicos	Permanente	Sí	No	No	Reglas heurísticas
[NORFORD 96]:	P,Q	Permanente/Transitorio	Sí	Sí	No	Cambio de media / distancias vectoriales
[LAUGHMAN 03]:	P,Q,D	Permanente/Transitorio	Sí	Sí	Sí	Mínimos cuadrados
[GILREATH 06]:	I neutro	Permanente	---	---	---	Transformada Concordia

Tab. 1 Resumen métodos heurísticos.

2.1.2. MÉTODOS ESTADÍSTICOS

Los métodos estadísticos difieren respecto de los vistos; mientras que en los ejemplos anteriores el reconocimiento de las cargas se realiza basándose en algoritmos guiados por la experimentación, usan reglas más o menos empíricas y analizan las curvas temporales de las signaturas, los métodos estadísticos emplean una fuerte carga de cómputo matemático, y funciones más o menos actuales. Hacen uso del desarrollo que han experimentado las ramas de la estadística y la probabilidad en el campo de la minería de datos.

En [LIN 10] el reconocimiento se produce usando un filtro bayesiano (figura 7) que calcula la posibilidad de que se dé el estado At dadas las observaciones desde O_1 hasta O_t. Así mismo compara los resultados con diferentes versiones del mismo y con otras técnicas como KNN (K nearest neighbors), .Naïve Bayes, SVM (Supported Vector Machine).

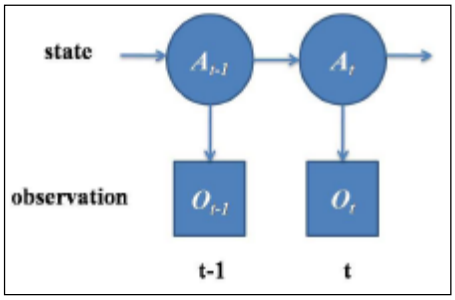

Fig. 7 Esquema filtro bayesiano

El resultado arrojado por el método del filtro bayesiano discretizado es superior en una comparación general (OA) al resto presentado con un 86.4%.

[RAHIMI 11] Experimenta con la distancia de Mahalanobis para el reconocimiento de una única carga, en régimen permanente y sin que ésta pueda variar lentamente. La precisión es del 100%.

[WANG 12] toma la corriente eficaz y la asemeja a figuras geométricas simples como se aprecia en la figura 8, de forma que almacenar y tratar esa información resulta más liviano, desde el punto de vista de la computación. La frecuencia de muestreo es muy baja, 10Hz.

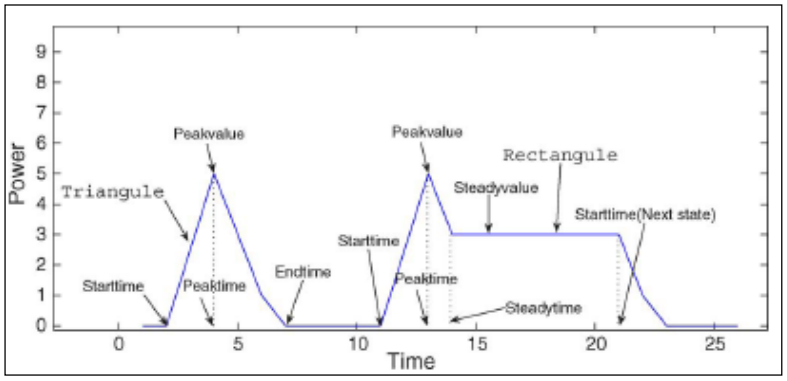

Fig. 8 Segmentación de la corriente en triángulos y rectángulos

También plantea una primera clasificación en cargas muy variables, variables y poco variables.

El siguiente cuadro (Tab 2) resume las técnicas estadísticas

	Signaturas	Régimen de análisis	Distinción cargas múltiples	Distinción cargas conexión simultánea	Distinción cargas lentamente variables	Modo Identificación
[LIN 10]	$Potencia_{i-7}$ P_{med} P_{max} P_{rms} $P_{Desviación\ estandar}$ $P_{Factor\ Cresta}$ P_{max}/P_{med} Lugar de P_{max}	Permanente	Sí	No	No	Filtro bayesiano
[RAHIMI 11]	P,Q	Permanente	No	No	No	Distancia Mahalanobis
[WANG 12]	I_{rms},V_{rms}, P	Permanente/Transitorio	No	No	No	Mean-shift
[YUN 12]	P	Permanente	No	No	No	Fuzzy Logic

Tab. 2 Resumen técnicas estadísticas.

2.1.3. REDES NEURONALES

Por último, veremos los sistemas de reconocimiento de cargas que usan redes neuronales. El sistema es parecido a lo ya visto, salvo que el procedimiento de clasificación es realizado por una red neuronal.

Contrariamente a lo que pueda parecer, los sistemas basados en redes neuronales no son sistemas recientes, sino que ya desde los primeros momentos en los que aparecen los estudios para la identificación de cargas, año 1992 en el que aparece la génesis de estos sistemas con la publicación de [HART 92], se empiezan a usar estas redes para la identificación, como demuestra la publicación del trabajo en 1994, tan solo dos años después, de [ROOS 94]

[ROOS 94] utiliza un sistema tal y como hemos comentado de recolección de datos en la acometida y son tratados siguiendo estos procesos.

- Captura de datos en formato vector.
- Preprocesado y filtrado; en caso necesario se separa en trozos más significativos.
- Extracción de información, eliminando redundancias y disminuyendo así la dimensión.
- Clasificación por la red neuronal previamente entrenada.

La red neuronal usada para la clasificación es una red de perceptrones multicapa, entrenados mediante la técnica de back propagation, de forma tal, que se realiza una clasificación en cascada según el esquema de la figura 9:

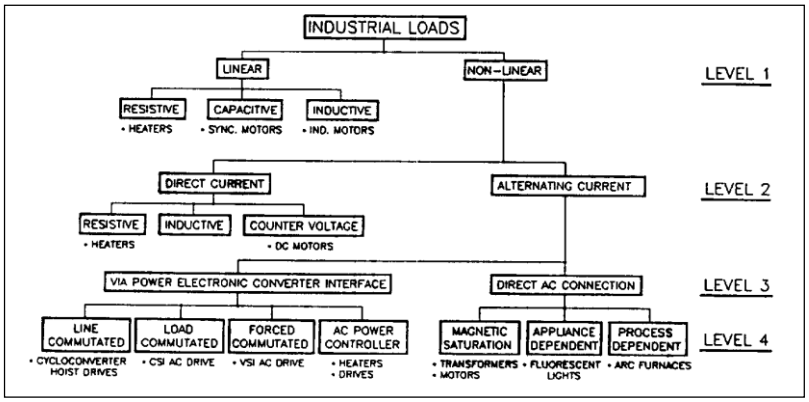

Fig. 9 Red en cascada de perceptrones clasificadores en niveles

A cada nivel le correspondería un perceptrón.

En el año 2007 se publica [PATEL 07]. La adquisición se realiza en cualquier punto de circuito eléctrico sensando únicamente la tensión. La signatura utilizada para la detección es el ruido eléctrico que se produce en el encendido y en el apagado de los dispositivos. Este ruido se propaga por todo el circuito y es capturado en cualquier toma de corriente perteneciente al mismo capturando un vector de frecuencias y amplitudes asociadas.

Este vector se preprocesa mediante FFT (Fast Fourier Transform) y se aplica a una red neuronal del tipo SVM (Support Vector Machine) entrenada. El número de ejemplares para el aprendizaje ronda los 3000.

[YANG 07] comparan dos métodos de aprendizaje neuronales para obtener cuál es el óptimo para la clasificación de cargas:

- Learning Vector Quantization.
- Back Propagation

ambos aplicados a un Perceptrón Multicapa. El resultado es una supremacía del aprendizaje Back Propagation sobre el primero; otra conclusión obtenida es que la energía del transitorio a ON se mantiene prácticamente constante en cada conexión de la carga y para la conexión con cualquier ángulo de la tensión de red.

[TSAI 11] optimiza una faceta de la identificación. Hasta ahora, las medidas que se presentaban al sistema de identificación eran elegidas por el diseñador en base a pruebas realizadas y la propia intuición. En este artículo, esta tarea se realiza usando GA (algoritmos genéticos). Así de una población de posibles medidas (corriente, potencia, pico de corriente, armónicos…) se extraen aquellos individuos que proporcionaran mayor información para el reconocimiento. La función fitness de este GA es el criterio de Fisher.

También realiza una comparación entre los sistemas de identificación KNNR (K Nearest Neigbor Rule) MLP con BP y MLP con LVQ. El elegido es el KNNR, debido a su sencillez y eficacia (98%).

[LIN 12] es el último artículo relacionado con la identificación de cargas. En este artículo, el sistema de identificación es un reconocedor de patrones neuro fuzzy con Linguisitc Hedges.

Se expone a continuación la tabla 3 como resumen de las redes neuronales.

	Signaturas	Régimen de análisis	Distinción cargas múltiples	Distinción cargas conexión simultánea	Distinción cargas lentamente variables	Modo Identificación
[ROOS 94]	I,P,Contorno Impedancia, armónicos de I, THD de P	Permanente	Sí	No	No	Red Perceptrones
[PATEL 07]	V	Transitorio	Sí	Depende tiempo entre evento	Sí	SVM
[YANG 07]	P,Q,,V,I,V_{HD},I_{HD}, V_{THD}, I_{THD},U_S,U_D	Permanente/Transitorio	Sí	Sí	Sí	Perceptrón +BP
[CHANG 10]	P,Q,U_s	Permanente/Transitorio	Sí	Sí	Sí	Perceptrón +BP
[TSAI 11]	I_{RMS}, I_{PP}, P_{trans}	Transitorio	Sí	No	No	K-NNR
[LIN 12]	$I_{Factor\ Cresta}$, Periodo Transitorio	Transitorio	Sí	No	No	NeuroFuzzy+LH

Tab. 3 Resumen redes neuronales.

2.2. ELECCION DE LA LÍNEA DE INVESTIGACION

Se han visto las principales líneas de investigación concernientes a los sistemas de monitorización de cargas de forma no intrusiva y los logros obtenidos por cada uno de ellas. A la vista de los resultados, no se puede decir que ninguno de los tres principales métodos (heurístico, estadístico, redes neuronales) posea una supremacía sobre los otros. Sin embargo, ajustándonos a los requerimientos de bajo consumo expuestos podemos declinarnos por uno de ellos.

Los sistemas heurísticos a priori requieren de menos carga computacional, menor frecuencia en el muestreo de las características que adquieren; por el contrario, precisan de una mayor intervención externa para fijar reglas lógicas o introducción de normas de funcionamiento, siendo menos versátiles. Los sistemas estadísticos requieren de una fuerte carga computacional para la obtención de resultados, siendo semejantes en cuanto al número de signaturas a las redes neuronales. Las redes neuronales, por su parte, requieren de una carga computacional media (dependiente de la red y la configuración) permitiendo ser fácilmente implementadas en computadores de rango medio, de propósito general. Se elegirán las redes neuronales como modelo de reconocimiento.

Otros parámetros importantes que definen el sistema son las signaturas elegidas y la frecuencia de muestreo que se tratará más adelante sobre ellas.

Un pilar fundamental sobre el que se basa este trabajo es el realizado por [TSAI 11], debido a que usa gran cantidad de parámetros extraídos de la forma de onda de la corriente, el uso de algoritmos genéticos en la elección de signaturas y su comparativa entre redes neuronales.

TDC

3. SISTEMA ADQUISICIÓN

El sistema físico de adquisición sigue un diagrama de bloques como el descrito en la figura 10: adquisición, acondicionamiento, digitalización y envío para el almacenamiento. Según los sistemas vistos, estos bloques varían poco en cuanto a su funcionalidad, diferenciándose en las prestaciones conforme el paso de tiempo.

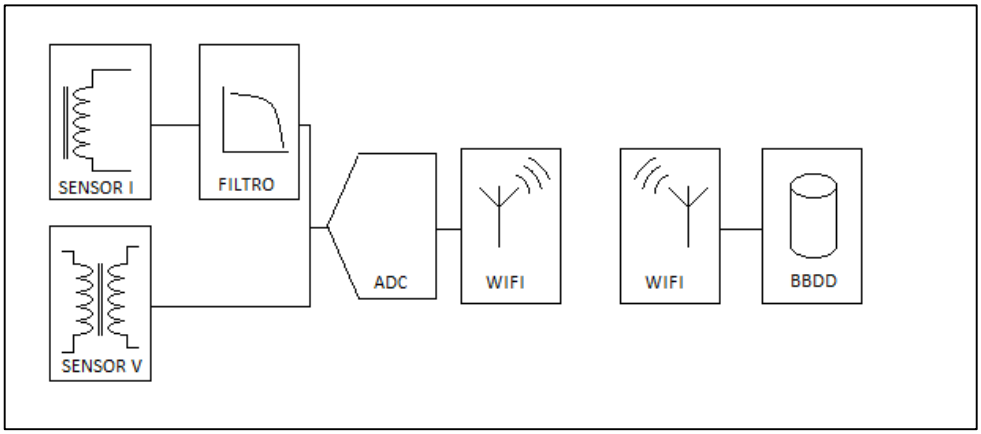

Fig. 10 Diagrama de bloques del sistema.

Es necesario obtener ciertos parámetros que constituirán las entradas a la red neuronal. Debido al compromiso de realizar un sistema de medias prestaciones, se usará como base el mismo sistema de adquisición que se usará para implementar el procesamiento, acercándose al sistema real y evitando la incertidumbre que podría provocar el paso del entorno de laboratorio al entorno real.

A continuación describiremos el sistema físico completo utilizado.

3.1. ADQUISICIÓN

La onda de corriente será obtenida mediante transformador de corriente tipo “pinza amperimétrica” modelo SCT0400-025 de MAGNELAB; proporciona una tensión de 0.333V a 25 Amperios de fondo de escala y fue sometida a una prueba de atenuación de señal, ya que el fabricante especifica la frecuencia máxima de operación en 400Hz por debajo de la de adquisición como se verá.

De la tensión, sólo se capturarán los pasos por cero de la misma mediante un transformador 230V/3.7V 4.8VA modelo ACP-7E de NOKIA para determinar el desfase corriente-tensión de la carga.

3.1.1. ENSAYO TRANSFORMADOR DE CORRIENTE

MATERIAL:

- Fuente de potencia
- Transformador de corriente SCT0400-25 (MAGNELAB)
- Resistencia de potencia
- Osciloscopio

DESCRIPCION:

Debido a que la frecuencia máxima de trabajo de la fuente es de 500Hz, por debajo de 2500Hz, la frecuencia de muestreo (como se verá más a delante), resulta imposible establecer una relación de ganancia directamente. Por ello, se excitará el esquema de la figura 11 en dos fases tal como se explica:

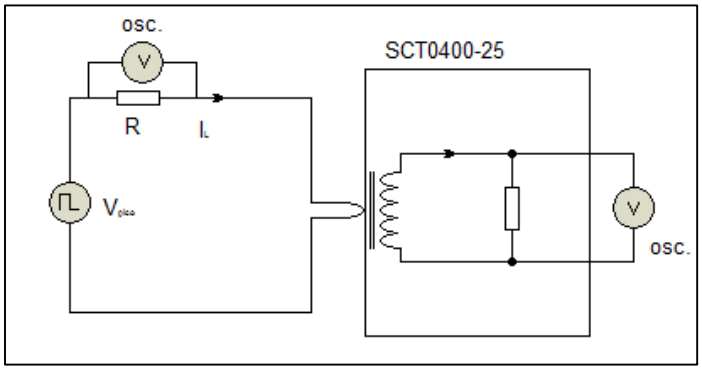

Fig 11 Ensayo del transformador de corriente.

Primera fase: excitación mediante senoides de frecuencias 50, 100, 500Hz

- V_{fuente}:110Vpico senoidal.
- F_{fuente}: 50,100Hz y 500Hz
- R: Resistencia cerámica de 10Ω 5%
- I_L:11Apico.

Se obtiene los siguientes valores de pico de la tabla 4:

Frec.(Hz)	Vsensor (mV)
50	146
100	146
500	146

Tab. 4 Resultado del ensayo del transformador de corriente primera fase.

No se aprecia atenuación.

Segunda fase: Excitación mediante una onda cuadrada de tensión comprobando que la respuesta frecuencial de la onda obtenida en el sensor no contiene atenuaciones en la banda de interés (despreciando la inductancia real de la resistencia).

Con los siguientes datos:

- Vfuente:110Vpico
- Ffuente: 500Hz
- R: Resistencia cerámica de 10Ω 5%
- IL:11Apico.

Se obtienen los siguientes valores, mostrados en Tabla 5, para las frecuencias armónicas de la fundamental en valores normalizados:

Frec.(Hz)	Vfuente	Vsensor	Vfuente/Vsensor
500	1.0000	1.0000	1.0000
1500	0.3133	0.3043	0.9713
2500	0.1500	0.1408	0.9387
3500	0.1129	0.1033	0.9149
4500	0.0931	0.0821	0.8819
5500	0.0689	0.0549	0.7968
6500	0.0619	0.0419	0.6770
7500	0.0537	0.0277	0.5155
8500	0.0438	0.0138	0.3154

Tab. 5 Resultado del ensayo del transformador de corriente.

La figura 12 muestra el resultado del ensayo

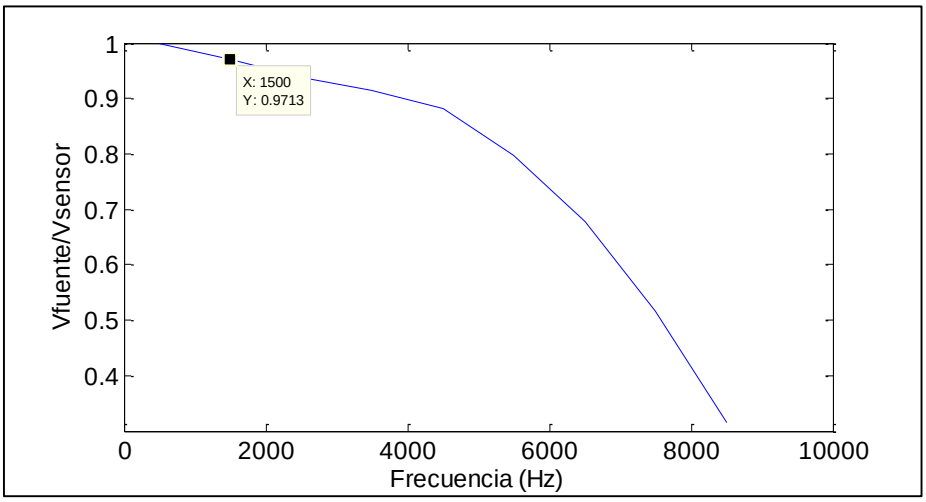

Fig. 12 Resultado del ensayo del transformador de corriente.

Como se puede apreciar existe una ligera atenuación hasta la frecuencia de 4500Hz pronunciándose a partir de la misma. Para la frecuencia de Niquist la ganancia resulta ser superior al 0.9

Tercera fase: excitación mediante senoide de frecuencia 50Hz, para determinar el desfase entre la onda de excitación y la onda resultante; el resultado es un desfase de 6.3º.

3.1.2. ENSAYO TRANSFORMADOR DE TENSIÓN

MATERIAL:

- Transformador de tensión ACP-7E (NOKIA)
- Osciloscopio

DESCRIPCION:

Se somete al transformador a un ensayo de desfase entre bobinados según el esquema de la figura 13.

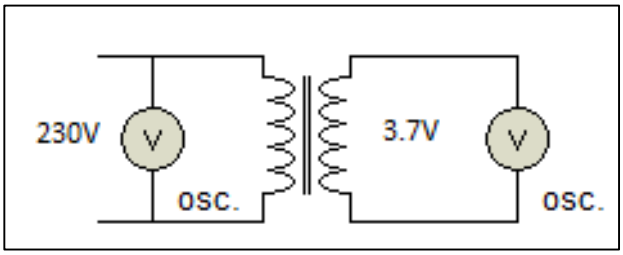

Fig. 13 Ensayo del transformador de tensión.

El resultado es un desfase de 4.95 grados

3.2. ACONDICIONAMIENTO, DIGITALIZACIÓN Y ENVÍO

El sistema se basa en la placa comercial *Flyport WIFI* de *OpenPICUS*, que consta de un microcontrolador, un regulador de tensión, y un módulo wifi; a esta base hardware se le dota de una placa aneja (figura 14) que es la encargada del acondicionamiento de señal. El Flyport realiza la conversión digital, y la comunicación a PC para el almacenamiento de los datos. Mediante el transformador de campo magnético a tensión conectado a J2 se realiza el muestreo de la forma de onda de la corriente; un transformador de tensión conectado a J1 registra la onda de tensión:

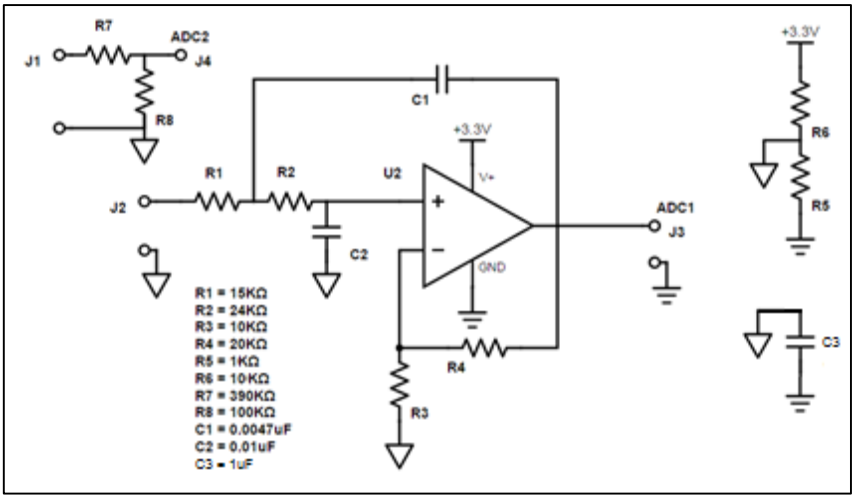

Fig. 14 Esquema de la placa acondicionadora de señal.

El amplificador operacional está configurado como filtro Sallen Key de segundo orden con fc=1223.5 Hz y amplificación de 3 (figura 15). La masa virtual necesaria para elevar las señales provenientes del sensor (±0.333V_{pico}) y que sean positivas a fin de usar el amplificador con alimentación simple resulta del punto medio del divisor resistivo formado por R6 y R5, con un valor de 0.3V sobre la masa de alimentación. Esto permite usar el sistema para cargas de hasta 5.18Kw.

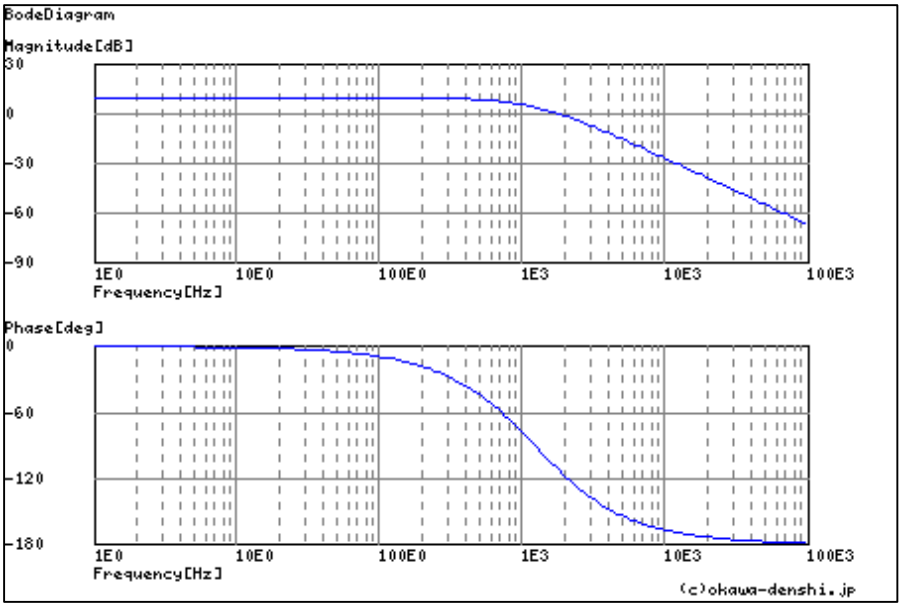

Fig. 15 Diagrama de Bode del filtro.

La digitalización la realiza el conversor de aproximaciones sucesivas de 10 bits incorporado en el microcontrolador de la placa FlyPort; la referencia es externa al microcontrolador y de 2.048V. La frecuencia de muestreo se fija en 2500Hz, máxima frecuencia de muestreo que ha soportado el sistema sin pérdida de datos en la conexión WiFi entre el sistema de adquisición y el almacenamiento.

3.3. ALMACENAMIENTO

Para la recepción de los datos enviados mediante conexión WiFi, se dispone de un ordenador PC con el programa MatLab encargado de la recepción y almacenamiento de todas las ondas capturadas. Posteriormente se procederá al análisis de estas señales (ANEXO II y siguientes). La tabla 6 resume los parámetros característicos del sistema físico.

Sensor de corriente	Fondo escala del sensor de corriente	0.333V a 25 A
	Ganancia 50Hz-1250Hz	>0.9
	Desfase	6.3º
Transformador de tensión	V primario	230V
	V secundario	3.7 Vrms
	Desfase	4.95º
Acondicionamiento	Frecuencia corte del filtro	1223.5Hz
	Margen de fase	68.2º a 1780Hz
	Ganancia filtro	3
	Masa virtual	0.3V
	Potencia máxima medible	5.18Kw
Digitalización	Nº Bits ADC	10
	Referencia	2.048V
	Frecuencia muestreo	2500Hz

Tab. 6 Parámetros del sistema.

TDC

4. ANÁLISIS DE LOS DATOS

Una vez descrito el sistema de adquisición, se realiza una base de datos con las señales capturadas para su posterior análisis.

4.1. BASE DE DATOS

La base de datos consta de diversos ficheros en los que se han capturado la forma de onda de corriente y pasos por cero de la tensión de los electrodomésticos testados (tabla 7):

Designación de carga	Nº Cargas simultáneas	Potencia Nominal(W)	Observaciones
Bat1	1	750	Batidora
Bat2	1	750	Batidora
Bat3	1	750	Batidora
Expr	1	20	Exprimidor
Micr	1	800	Microondas
Sand	1	750	Sandwichera
Ven1	1	60	Ventilador
MicrOnYSandOn	2	1550	Agregada de anteriores
SandOnYBat3On	2	1500	Agregada de anteriores
Ven1OnYBat3On	2	810	Agregada de anteriores
Ven1OnYMicrOnYSandOn	3	1610	Agregada de anteriores

Tab. 7 Población de cargas.

Notas:

- Las cargas denominadas Bat1, Bat2 y Bat3 se refieren a la misma batidora regulada mediante mando deslizante y triac, en las posiciones más baja, a la mitad de recorrido del mando y en la posición más alta, respectivamente en vacío.
- El exprimidor es usado normalmente (no en vacío) por lo que el par resistente es dependiente de la fuerza ejercida en cada instante.
- El microondas, a la vista del perfil de la corriente, sigue una secuencia de dos fases al aplicar toda su potencia. Es usado en modo Quick (toda la potencia durante el máximo tiempo posible).
- Sandwichera usada normalmente.
- Ventilador usado normalmente a la velocidad 3 (máxima velocidad).
- Las cargas agregadas son una sucesión de conexiones y desconexiones en el orden indicado por el nombre.

4.1.1. PROCEDIMIENTO DE CAPTURA

Se realizan 10 trials por cada carga siguiendo el siguiente protocolo:

Protocolo:

- Comienzo de grabación de datos.
- Espera de 3 segundos
- Conexión de la carga a la red.
- Espera de 3 segundos.
- Conexión de segunda o sucesivas cargas cada 3 segundos.
- Desconexión de la última carga conectada (orden inverso).
- Espera de 3 segundos.
- Desconexión de penúltima o predecesoras cargas cada 3 segundos (orden inverso).
- Finalización de grabación de datos.

Como resultado se obtienen 110 ficheros de datos, con al menos una conexión y desconexión por carga. Ejemplo de un fichero es el de la figura 16 y detalle de la conexión de una carga agregada la figura 17.

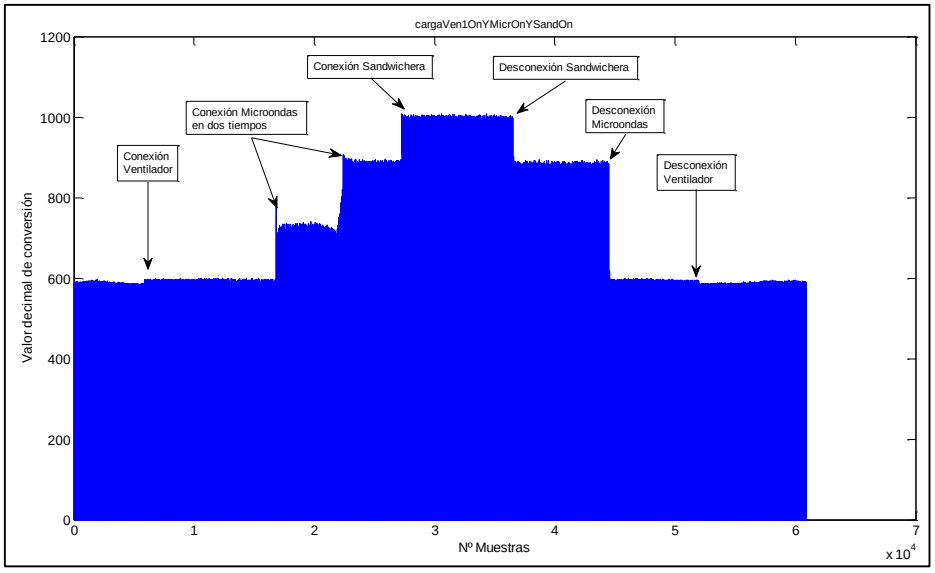

Fig. 16 Corriente obtenida de un trial de la carga Ven1OnYMicrOnYSandOn.

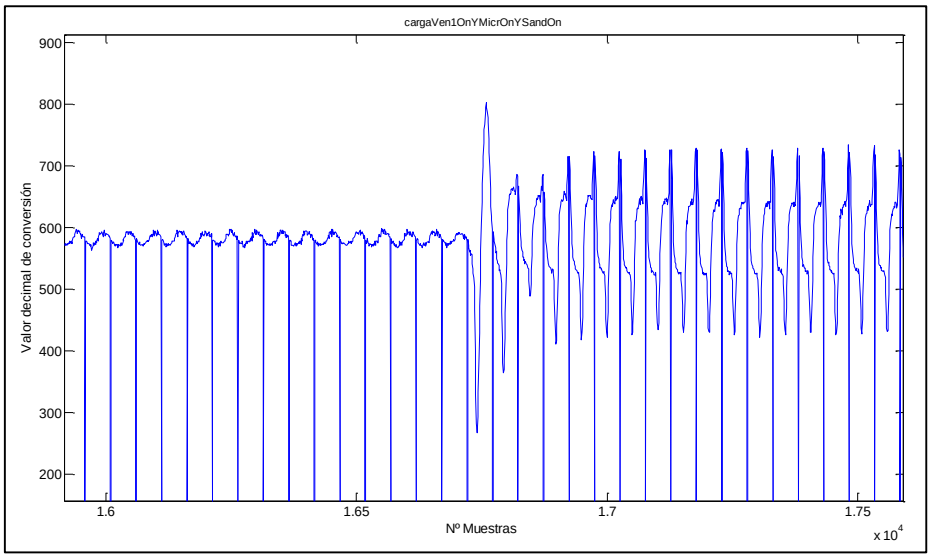

Fig. 17 Detalle de la conexión del microondas agregada a la señal del ventilador.

Las muestras con valor cero son ficticias y se han intercalado durante el muestreo para indicar el momento del paso por cero de la tensión. Posteriormente serán separadas, marcando así el desfase corriente-tensión.

Como se observa del protocolo, no se han tenido en cuenta **conexiones** simultáneas de cargas.

Otro aspecto a destacar es la diferencia del transitorio al régimen permanente, lo que proporciona información valiosa del electrodoméstico conectado como constatan [NORFORD 96], [LAUGHMAN 03] [WANG 12] [YANG 07] [TSAI 11] [LIN 12].

4.2. PROCESADO

Las ondas así capturadas, se someterán a un acondicionamiento de la señal para posteriormente obtener las características para las entradas de la red neuronal. El código de esta simulación de procesado puede consultarse a partir de la línea 150 del ANEXO II.

4.2.1. EXTRACCIÓN DEL PERIODO DE FUNCIONAMIENTO

Como se ha visto en la descripción del protocolo, existe un periodo entre el comienzo de captura y la conexión de 3 segundos (al igual que en la desconexión) y de funcionamiento en régimen estacionario antes de la conexión de otro dispositivo. Hay que identificar el momento de éstos eventos a fin de extraer los intervalos en los que únicamente están la o las cargas de interés. En el caso de cargas únicas, el intervalo de interés es el correspondiente al estado de conexión (incluyendo el transitorio); en el caso de cargas agregadas, el intervalo es el correspondiente a estar todas las cargas conectadas a la vez. Se siguió inspección visual para determinar el momento de la conexión y desconexión. En el ejemplo de la carga batidora en la velocidad media (Bat2) se puede señalar como momento de la conexión el señalado en la figura 18. Aplicando el mismo criterio a las cargas no agregadas y agregadas, se obtienen el periodo de ON (ANEXO II líneas 150-164).

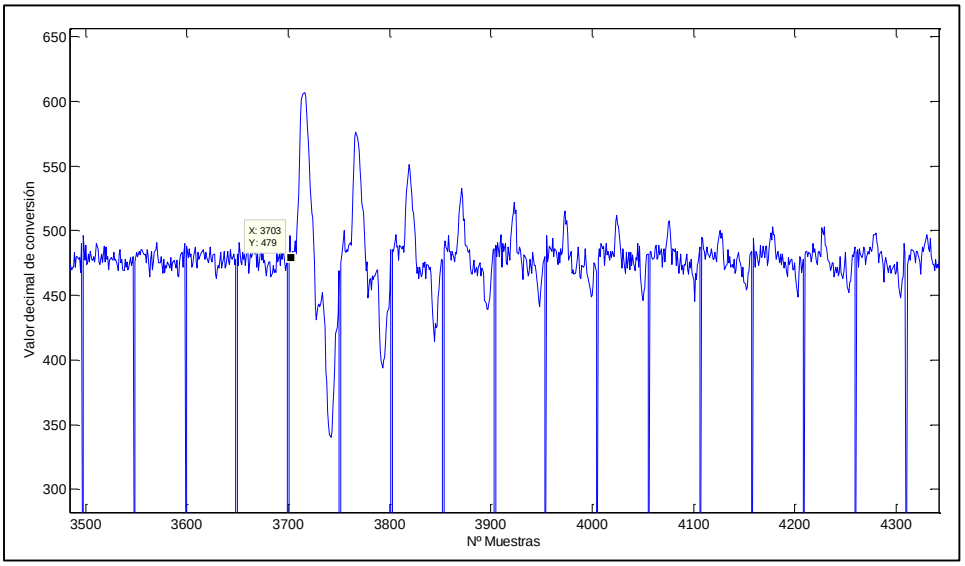

Fig. 18 Detalle de la conexión de la carga Bat2.

4.2.2. EXTRACCIÓN DE LOS PASOS POR CERO DE LA TENSIÓN.

Como se ha mencionado, para no sobrecargar el canal WiFi, no se envía la señal completa de tensión, sino que mediante un algoritmo de comparación y filtrado en el microcontrolador se determina el momento del cruce por cero con tendencia decreciente de la misma, momento en el que se incluye un valor cero en los datos de la onda de corriente. De esta manera queda registrado y sincronizado el cruce por cero de la tensión. Por lo cual, ahora es necesario eliminar ese registro, guardando el valor que ocupaba en la secuencia para posteriormente obtener el corriente-tensión tal y como se ilustra en el siguiente esquema de vectores (tabla 8) (ANEXO II líneas 373-413):

Índice	n	n+1	n+2	n+3	n+4	n+5	…	…	k	k+1	k+2	k+3	k+4
Valor onda	454	485	0	487	492	500	…	…	450	462	0	470	478

Índice	n	n+1	n+2	n+3	n+4	n+5	…	k	k+1	k+2	k+3	k+4	…
Valor onda	454	485	487	492	500	…	…	450	462	470	478	…	…

Pasos por cero	…	n+2	k+2	…

Tab. 8 Esquema de la extracción de pasoso por cero.

4.2.3. EXTRACCIÓN DE LOS PASOS POR CERO DE LA CORRIENTE.

Para la determinar cuándo se producen los pasos por cero de la corriente se comienza eliminando la componente continua de la señal, mediante un filtro de mediana, el cual resta a cada punto, la mediana de la totalidad de ella misma. A la vista de los resultados obtenidos en la figura 19 para la carga Bat1, determinar el momento del paso por cero resulta imposible debido a que la señal puede cambiar de signo varias veces en un periodo. Se opta por someter a la señal a un filtrado digital, que permita determinar el momento del cruce por cero con pendiente negativa (ANEXO II líneas 417-496).

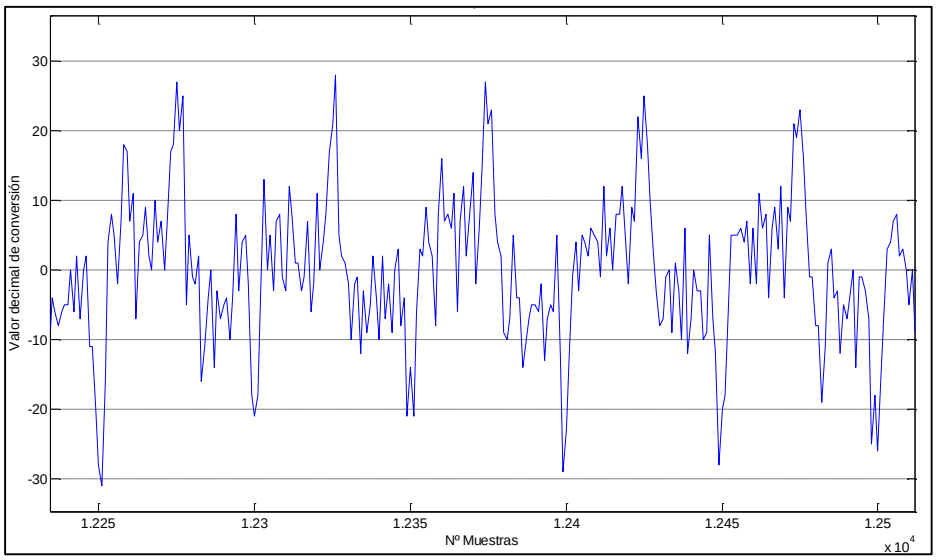

Fig. 19 Detalle de los pasos por cero de la corriente.

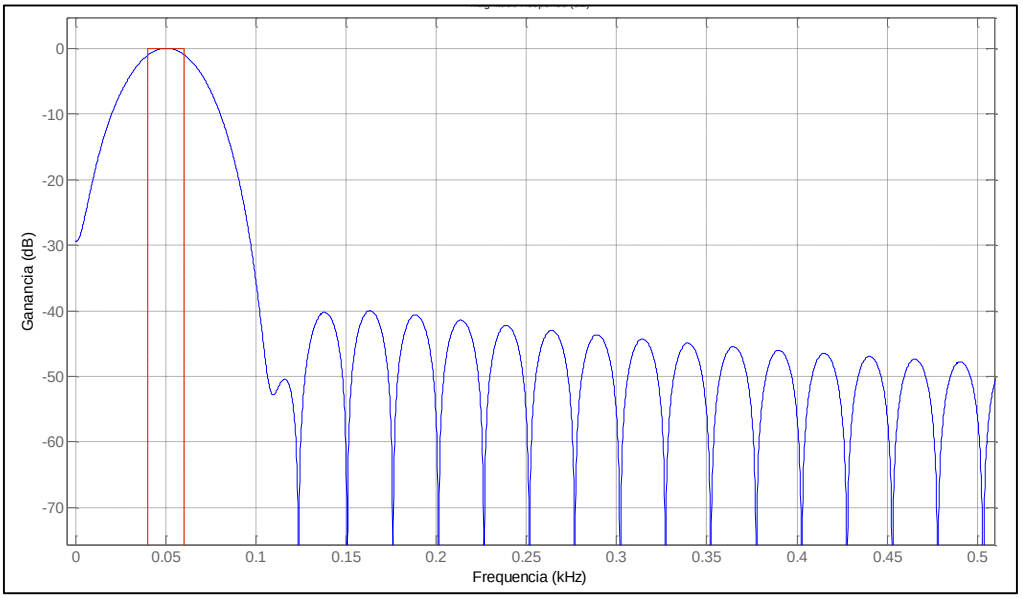

Fig. 20 Ganancia del filtro digital.

En la figura 20 se observa el diagrama de Bode de la ganancia del filtro; se ha optado por un filtro paso banda con frecuencias de corte 40 y 60Hz tal y como se aprecia para obtener la componente correspondiente a los 50 Hz. El resto de parámetros del filtro se pueden consultar en el Anexo I.

El resultado de aplicar tal filtro a la carga BAT1 puede observarse en la figura 21 donde se ha corregido el retardo de grupo correspondiente a 50 muestras. De esta señal filtrada puede extraerse el cruce por cero.

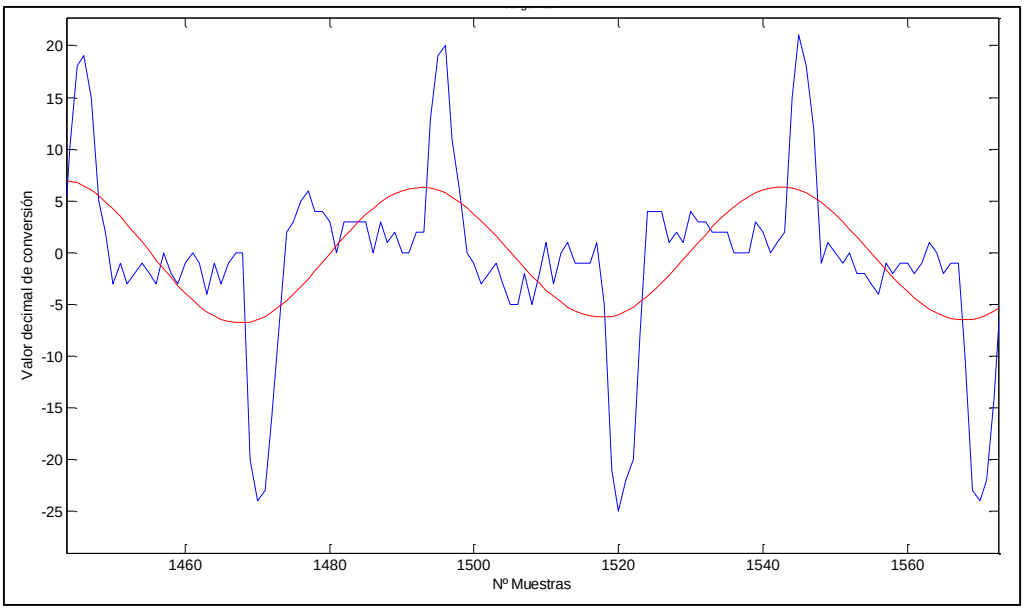

Fig. 21 Corriente en crudo y corriente filtrada.

Para este tipo de cargas activadas mediante interruptores controlados, hay que notar que el resultado no corresponde con el factor de potencia ni el ángulo de la impedancia debido a que las formas de onda de la corriente alimentadas mediante estos interruptores (tipo triac, etc) rompen con la excitación permanente sinusoidal, condición necesaria para que el factor de potencia coincida con el desfase entre tensión y corriente. Por tanto, este parámetro es otra impronta de la corriente de la carga, sin ser estrictamente un parámetro convencional.

A partir de estos datos, se obtiene el desfase entre las ondas de tensión y corriente restando ambos pares de ordenadas (posiciones del paso por cero de la corriente y de la tensión).

4.3. EXTRACIÓN DE CARACTERÍSTICAS

A partir de este momento, la onda se encuentra preparada para extraer las características que servirán como entrada a la red neuronal; han sido seleccionadas en su mayoría siguiendo el criterio de [TSAI 11], otras debido a que han sido elegidas por la mayoría de las fuentes consultadas, mientras que algunas han sido escogidas a criterio del autor del presente.

- Valor eficaz.
- Centroide del patrón de Concordia modificado.
- Amplitudes de espectro del primer, tercer y quinto armónico.
- Valor de pico.
- Valor a cuarto de onda.
- Desfase corriente-tensión.

El archivo resultante del acondicionamiento de la señal tras haber obtenido el periodo de funcionamiento, extraídos los pasos por cero de la tensión y eliminada la componente continua se divide en ventanas de ciclos completos con el fin de obtener por una parte, una cantidad de señal significativa sobre la que operar al ser mayor de un único ciclo, y por otra parte, obtener una gran cantidad de muestras o individuos.

Se ha fijado un ancho de ventana de 10 ciclos completos de señal.

4.3.1. CALCULO DEL VALOR EFICAZ

El valor RMS de la onda se calcula mediante la siguiente fórmula:

$$I_{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}$$

Siendo N el número de muestras por ventana e igual a 500. (Frecuencia de muestreo dividido por 50 ciclos por segundo de la red y multiplicado por el ancho de ventana: 10) (ANEXO II líneas 893-1164).

4.3.2. CENTROIDE DE CONCORDIA

Haciendo uso de las características del centroide de los patrones de la Conversión de Concordia modificada, [GILREATH 06], se puede obtener información del contenido armónico de la onda.

Para ello es necesario tener un sistema trifásico equilibrado; puesto que no es el caso de estudio, se genera un sistema trifásico ficticio, triplicando la onda de corriente, posteriormente desfasándolas y eliminando las muestras sobrantes del desfase (Figura 22) (ANEXO II líneas 893-1164).

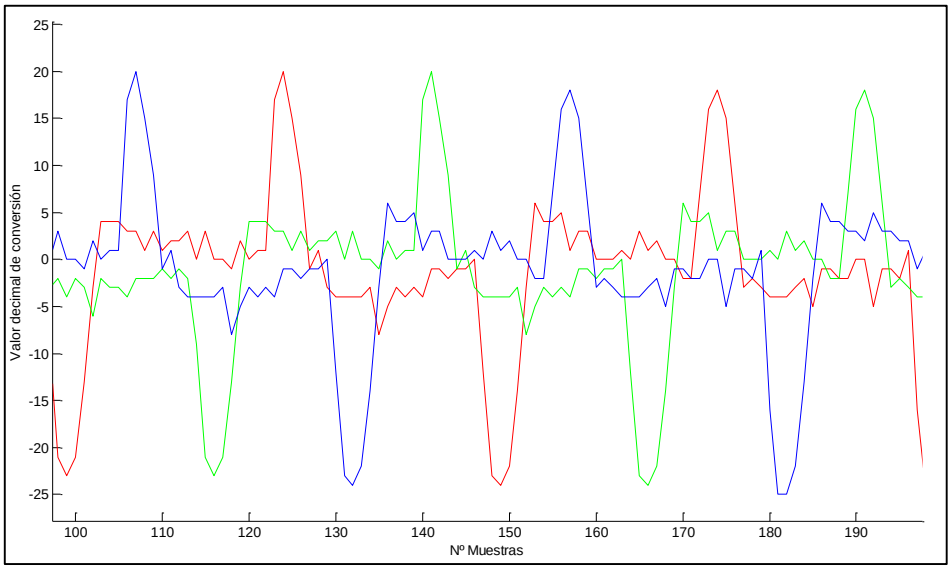

Fig. 22 Sistema trifásico ficticio.

Posteriormente se calcula la transformada de Concordia según estas fórmulas:

$$i_\alpha = \sqrt{\frac{2}{3}} i_a - \frac{1}{\sqrt{6}} i_b - \frac{1}{\sqrt{6}} i_c$$

$$i_\beta = \frac{1}{\sqrt{2}} i_b - \frac{1}{\sqrt{2}} i_c$$

Siendo i_a, i_b e i_c la onda de corriente original y las desfasadas.

Se realiza la modificación de la transformada propuesta en el artículo, tomando valor absoluto alguna de las coordenadas i_α o i_β, obteniendo así los pares $[i_\alpha, |i_\beta|]$ o bien $[|i_\alpha|, i_\beta]$.

En la figura 23 se puede ver una comparativa entre los contornos obtenidos mediante este método para la carga agregada de ventilador, microondas y sandwichera (a) y la teórica mostrada en el artículo mencionado.

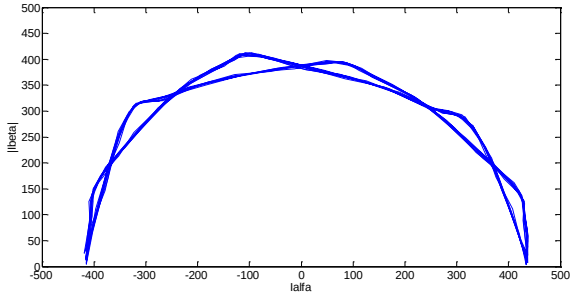

Fig 23 a)

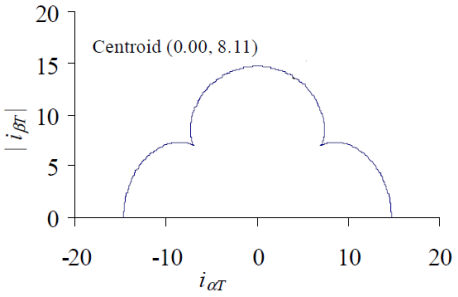

Fig 23 b)

Fig. 23 Comparativa entre contornos

A continuación se obtiene el centroide de estos contornos mediante las formulas:

$$C\alpha = \frac{\sum_{i=1}^{N} i_{\alpha i}}{N}$$

$$C\beta = \frac{\sum_{i=1}^{N} |i_{\beta i}|}{N}$$

Donde Cα y Cβ (Cbeta) son las coordenadas del centroide para los ejes α y β respectivamente, i_α e i_β los valores instantáneos de la onda capturada y N el total de muestras para una captura.

Estos pares de coordenadas del centroide serán las correspondientes entradas de la red neuronal. El espacio creado por todos los individuos centorides para las diferentes cargas puede observarse en la figura 24

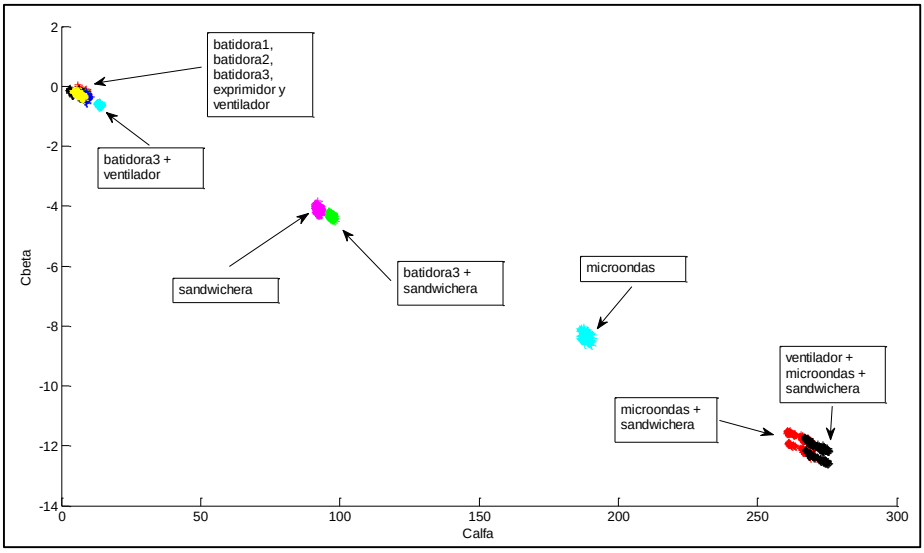

Fig. 24 Espacio de centroides

4.3.3. TRANSFORMADA DE FOURIER

Otras características que se obtienen de la señal es la potencia de los armónicos. Para ello se calcula la transformada de Fourier para el armónico fundamental o primero, tercer y quinto. En la figura 25 se puede ver como la carga microondas (a) presenta cierto contenido armónico de tercer y quinto armónico, mientras que en la carga batidora2 (b) el contenido es mucho mayor, con escasa diferencia entre quinto y tercero (ANEXO II líneas 893-1164).

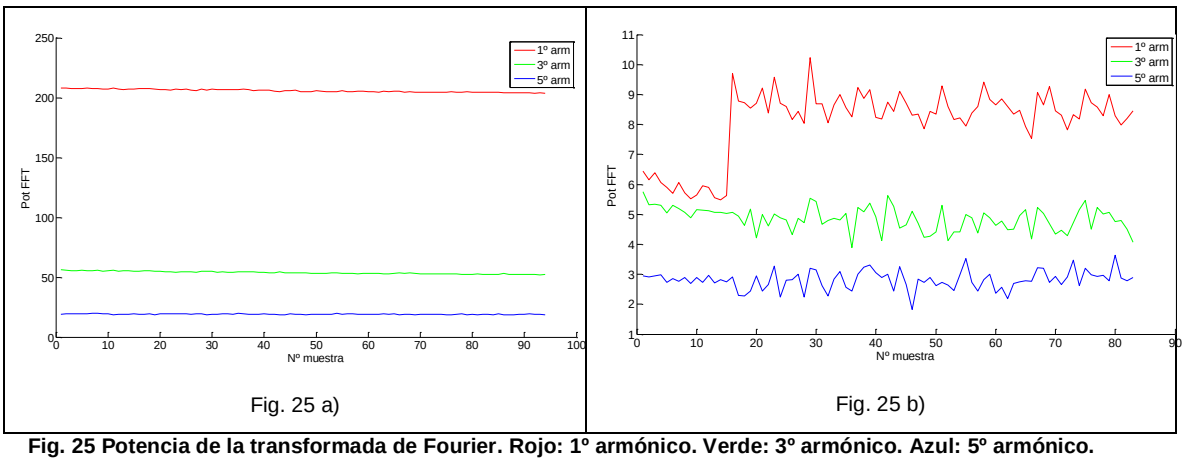

Fig. 25 a) Fig. 25 b)

Fig. 25 Potencia de la transformada de Fourier. Rojo: 1º armónico. Verde: 3º armónico. Azul: 5º armónico.

Por cargas, podemos apreciar las potencias del primer, tercer y quinto armónico en las figuras 26 27 y 28 respectivamente.

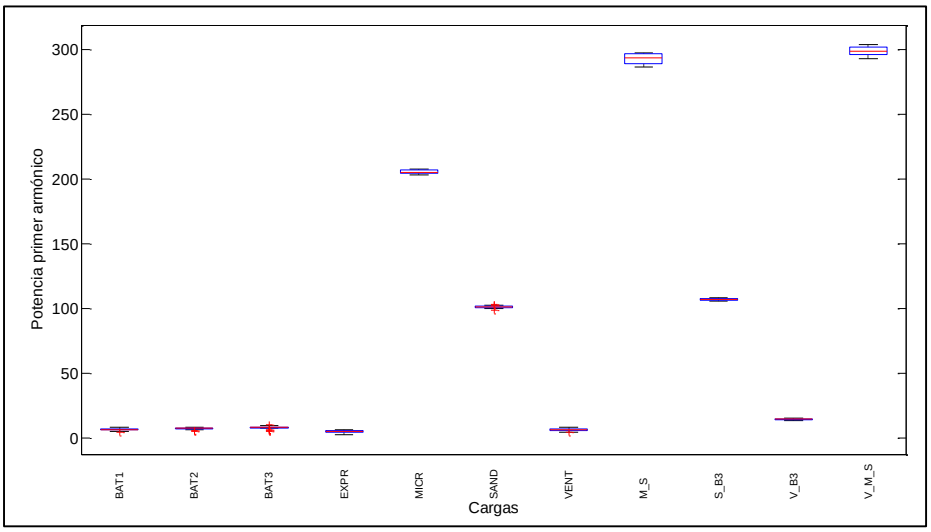

Fig. 26 Boxplot potencia del primer armónico por cargas.

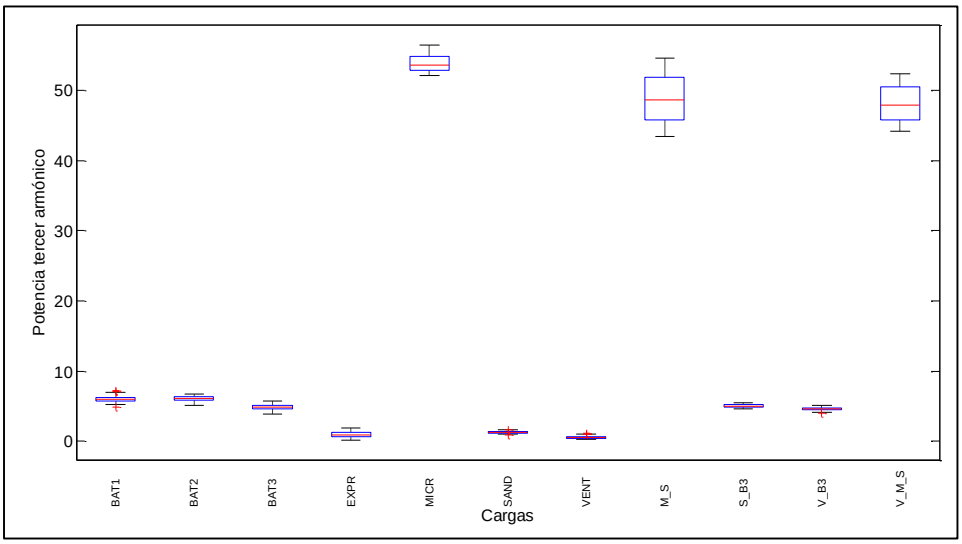

Fig. 27 Boxplot potencia del tercer armónico por cargas.

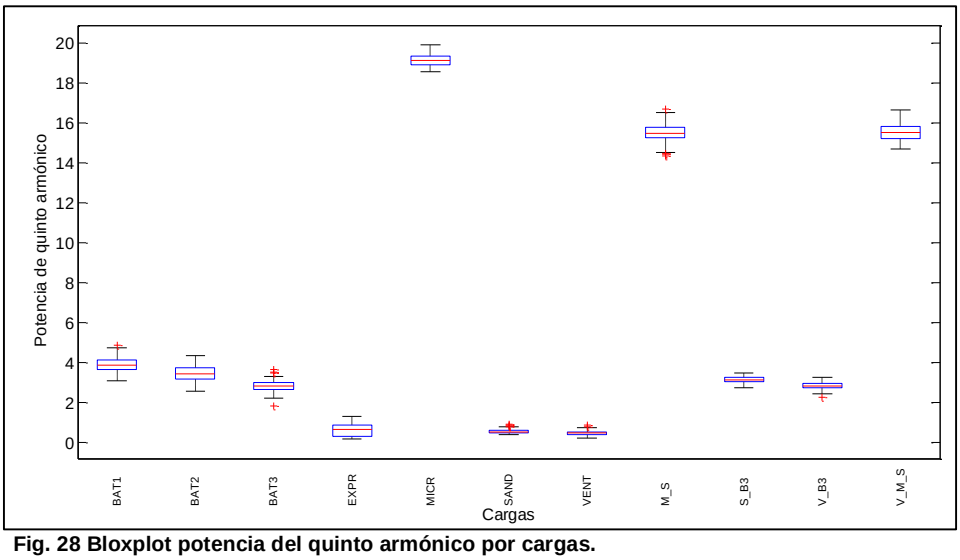

Fig. 28 Bloxplot potencia del quinto armónico por cargas.

4.3.4. VALOR DE PICO

El valor de pico se elige como el máximo valor de cada ventana o individuo. En la figura 29 se puede observar la variabilidad de los resultados (ANEXO II líneas 1165-1364).

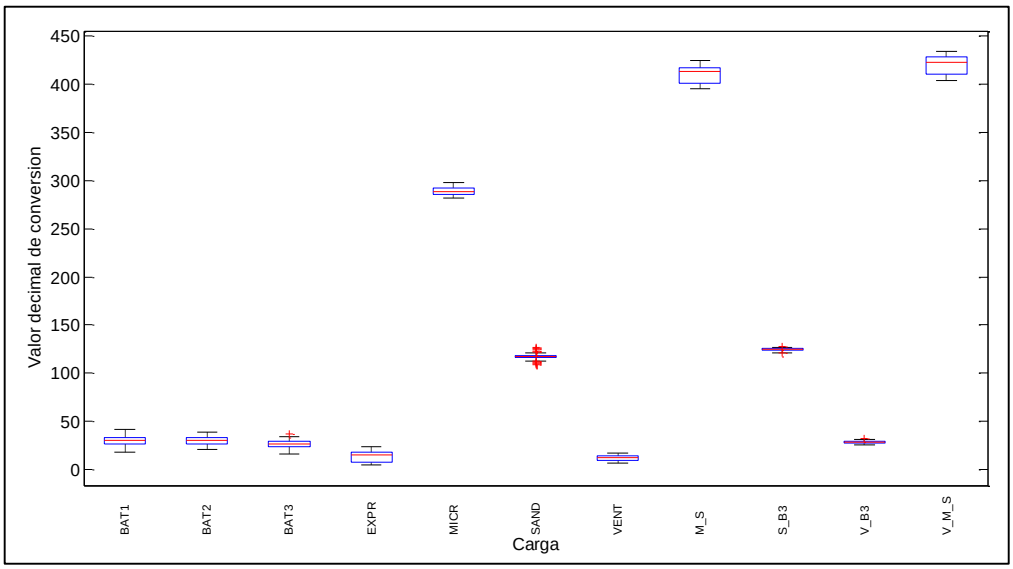

Fig 29 Boxplot del valor de pico referido a cada carga.

4.3.5. VALOR DE INTER SEMIPERIODO

Este parámetro es el valor de la onda de corriente en la posición correspondiente a la máxima diferencia que existe entre una onda sinusoidal pura, y otra sinusoidal afectada por una distorsión de tercer armónico igual a una onda cuadrada. Para calcular esta posición, se resta punto a punto (figura 30(b)) una sinusoide de amplitud unidad, menos la composición de esa misma sinusoide más el tercer armónico de una onda cuadrada de amplitud unidad (figura 30(a)). Es decir, a un décimo aproximadamente del período completo; intenta ser una medida de la distorsión producida por el tercer armónico, de forma que ese valor diferirá del valor de una sinusoide pura en tanto en cuanto aumente el contenido de ese tercer armónico(ANEXO II líneas 1165-1364).

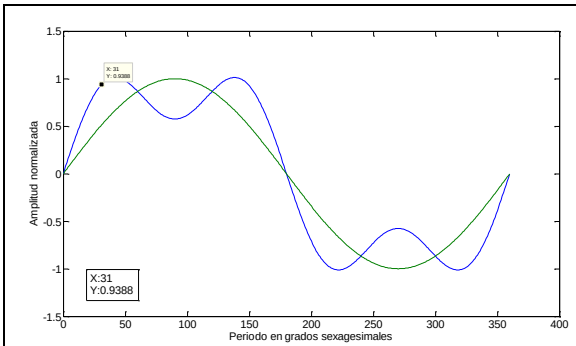

Fig. 30a)

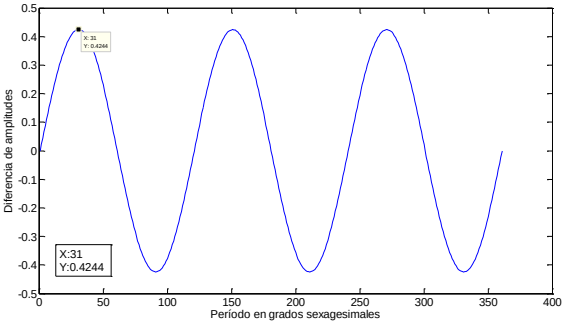

Fig. 30b)

Fig. 30 Elección del valor inter semiperiodo.

Para el caso de las señales de corriente, este valor en posición obtenida es dividido por el valor de pico obteniendo estos resultados de la figura 31. El valor negativo proviene de haber elegido el paso por cero en el momento decreciente, y por consiguiente el semiperiodo negativo.

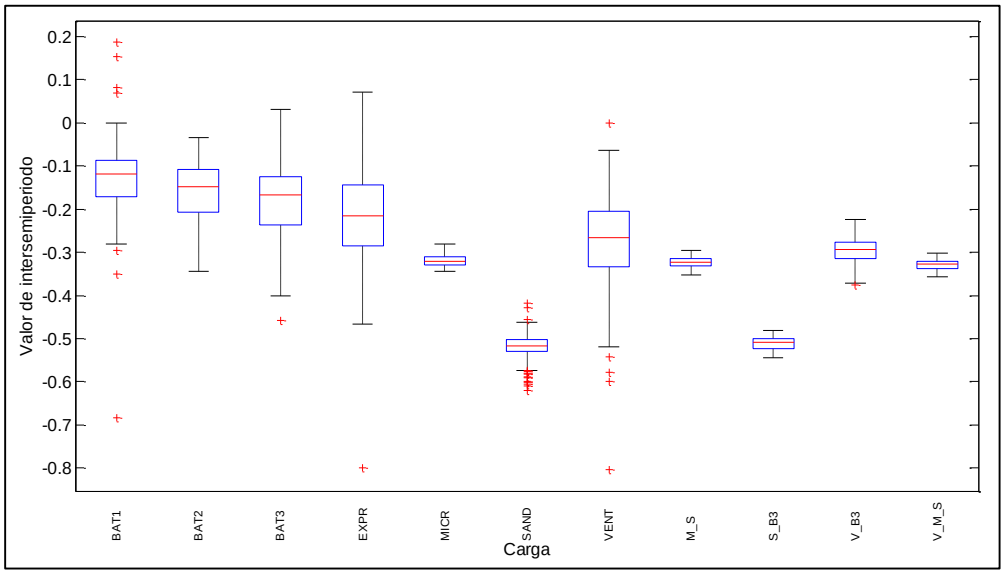

Fig. 31 Boxplot del valor de intersemiperiodo referido a cada carga.

4.3.6. DESFASE TENSION CORRIENTE

Como se ha mencionado con anterioridad, se capturaron los pasos por cero de la tensión, y se sometió a la señal de corriente a un filtrado para obtener un único punto de cruce por cero. El desfase de tensión y corriente se obtiene restando uno a uno cada punto así obtenido. La figura 32 recoge los resultados (ANEXO II líneas 500-891).

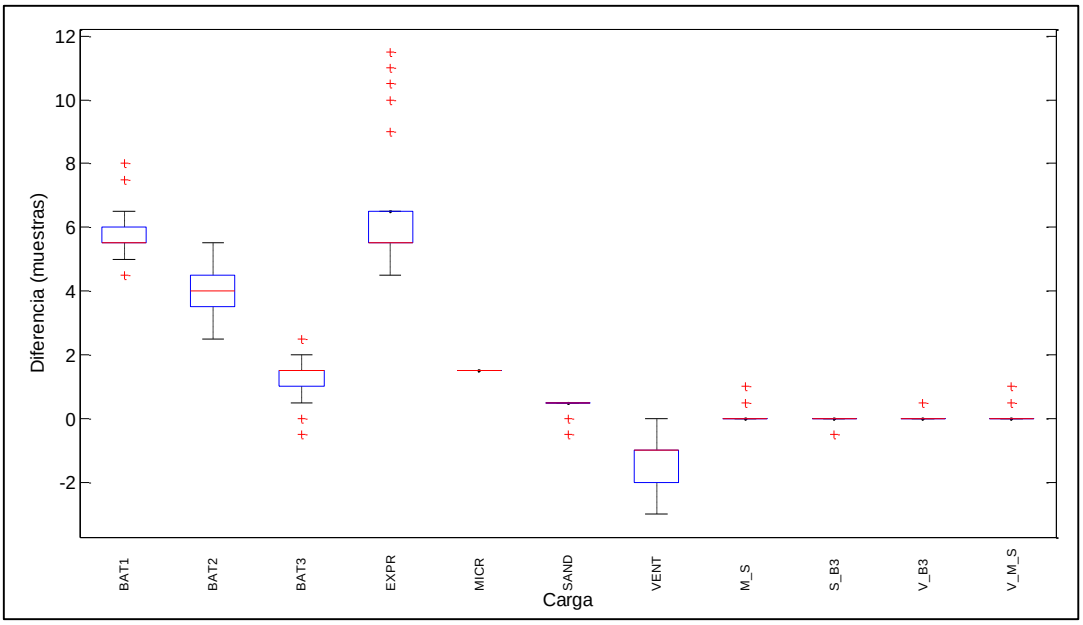

Fig. 32 Boxplot de la diferencia en muestras entre tensión y corriente.

4.4. EVALUACIÓN DE LAS CARACTERÍSTICAS ELEGIDAS

Hasta este momento, se ha descrito el método utilizado para acondicionar y extraer características de las señales capturadas dentro de conjunto de electrodomésticos elegidos. Y tal como se ha comentado, la clasificación se realizará mediante una red neuronal. Sin embargo, antes de realizar la selección de la red neuronal para elegir aquella que más se ajuste a los requerimientos, se evalúa el conjunto de características, a fin de tener una línea base sobre la que comparar las modificaciones y tratamientos que se realizara al sistema.

Para ello, el conjunto de datos obtenido como se ha mencionado hasta ahora se presenta a una red neuronal que sirva de primera medida de la posible calidad del sistema; esta red es un mapa auto organizado. Todo el conjunto de características se divide en dos grupos: uno de entrenamiento y otro de verificación. El conjunto de entrenamiento es el que se usa para generar las regiones o clusters de neuronas y calcular los pesos de los enlaces entre neuronas vecinas. El grupo de verificación servirá para calcular cómo de buena ha sido la clasificación, mediante el parámetro kappa, que varía de 0 a 1 (con 1 como clasificación perfecta) (ANEXO II líneas 1634-1747).

El conjunto de datos se dividió al 80% como grupo de entrenamiento y 20% como grupo de verificación. Esta división tuvo en cuenta la proporción de las clases dentro del conjunto, puesto que no hay igual cantidad de individuos en cada clase; así se extrajo el mencionado porcentaje de cada clase.

El tamaño de la red neuronal se eligió de 26x13 neuronas.

El resultado de la clasificación del grupo de validación se expone en la tabla 9 con formato de matriz de confusión

	Bat1	Bat2	Bat3	Expr	Micr	Sand	Ven1	MicrOnYSandOn	SandOnYBat3On	Ven1OnYBat3On	Ven1OnYMicrOnYSandOn
Bat1	20	8	0	0	0	0	0	0	0	0	0
Bat2	1	8	6	0	0	0	0	0	0	0	0
Bat3	0	0	11	0	0	0	0	0	0	16	0
Expr	2	0	0	39	0	0	0	0	0	0	0
Micr	0	0	0	0	19	0	0	0	0	0	0
Sand	0	0	0	0	0	99	0	0	0	0	0
Ven1	0	0	0	0	0	0	42	0	0	0	0
MicrOnYSandOn	0	0	0	0	0	0	0	30	0	0	0
SandOnYBat3On	0	0	0	0	0	0	0	0	22	0	0
Ven1OnYBat3On	0	0	0	0	0	0	0	0	0	6	0
Ven1OnYMicrOnYSandOn	0	0	0	0	0	0	0	0	0	0	33
TOTAL INDIVIDUOS	23	16	17	39	19	99	42	30	22	22	33

Tab. 9. Primera evaluación del sistema clasificador.

El resultado de aplicar el mencionado coeficiente kappa, que nos indica la bondad del clasificador es:

$$Kappa = 0.9081$$

Lo que resulta en un clasificador muy bueno.

4.5. REDUCCIÓN DEL COSTE COMPUTACIONAL

Sin embargo, como se mencionó en el objeto de este trabajo, se intentará reducir el coste computacional (y por tanto energético) del sistema usando diferentes técnicas. En base a esta primera aproximación del clasificador, se puede tener una idea de cómo pueden influir estas técnicas en el resultado final del posible clasificador por comparación con el primero.

4.5.1. REDUCCION DE LA FRECUENCIA DE MUESTREO.

Una medida para reducir el consumo consiste en reducir la tasa de muestreo, e intentar mantener el sistema en estado de muy bajo consumo (dormido o sleep) el tiempo inter muestras. Cabe esperar que esta medida empeore la clasificación debido a un empeoramiento de las características elegidas.

Se simula una disminución de la tasa de muestreo mediante la técnica de subsampleo, en la que de la señal original se conservan muestras en proporción una de cada dos, una de cada tres, una de cada cuatro,… de forma correlativa, con lo que el efecto es como dividir la frecuencia de muestreo por dos, tres, cuatro,… respectivamente (ANEXO II líneas 1634-1747).

El resultado se puede apreciar en la figura 33

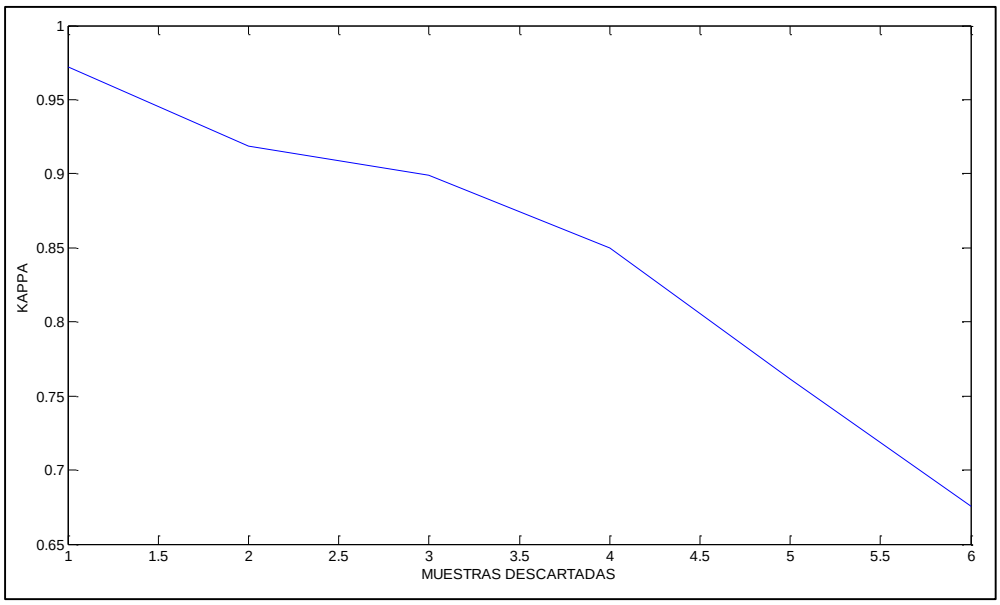

Fig. 33 Degradación del valor Kappa al disminuir la frecuencia de muestreo.

4.5.2. REDUCCION DEL NUMERO DE CARÁCTERÍSTICAS.

Igualmente se han elegido unas características de la onda de corriente que son las encargadas de definir cada una de las cargas. Reducir el número de características también empeoraría el clasificador, sin embargo, reduciría el coste energético de calcularlas y acondicionarlas. Sin embargo, esta reducción o eliminación de entradas de la red neuronal no puede hacerse de forma sistemática; la forma de ver si una característica contiene mucha información útil para la red, es probando. Por ello se ha utilizado un algoritmo iterativo de comprobación, por el cual a una red tipo mapa auto organizado se presentan subconjuntos del total de las características calculadas, obteniendo las combinaciones que cuentan con un kappa mayor. Para esta reducción, no se ha reducido la frecuencia de muestreo, puesto que el resultado buscado es el subconjunto de entradas que menos degrada el valor kappa cualitativamente y no cuantitativamente (ANEXO II líneas 1750-1825).

El resultado puede verse en la tabla 10:

KAPPA	RMS	CCDA	CCDB	FFT1	FFT2	FFT3	FP	PICO	CUARTO
0.9638	0	1	1	1	1	1	1	1	1
0.9666	1	1	1	1	1	1	1	0	1
0.9749	1	1	1	1	1	1	1	1	0
0.9777	0	1	1	1	1	1	1	1	0
0.9833	1	1	1	1	1	0	1	1	0
0.9833	1	0	1	1	1	0	1	1	0
0.9861	1	1	1	0	1	0	1	1	0

Tab. 10 Indice Kappa en función de las características presentadas

De esta tabla se puede deducir que hay características que posiblemente introduzcan ruido al sistema como puede ser la entrada “cuarto” ya que el algoritmo la ha descartado en varias ocasiones.

Para reducir el coste computacional y a la vista que hay características que podrían introducir ruido se eliminarán:

- Valor de Inter Semiperiodo (CUARTO).
- Valores de la transformada de Fourier (FFT1, FFT2, FFT3).

4.5.3. DETECCION DE EVENTOS

Siguiendo con el propósito de reducir el coste computacional, reduciendo la frecuencia de muestreo, También se ha estudiado el efecto de la modulación de muestreo. Esta modulación consiste en una vez elegida la frecuencia de muestreo, en lugar de realizar un muestreo continuo se modula el tiempo de durante el que se muestrea y el tiempo durante el que no se realiza acción alguna, permitiendo al sistema cambiar a un modo de muy bajo consumo; esta modulación tiene su base en la técnica PWM en el que se fija un tiempo de trabajo (adquisición a la frecuencia seleccionada) y un tiempo de espera (bajo consumo), pero en este caso los tiempos son mucho mayores que el periodo de la señal.

Se han desechado características propias de los transitorios, las cuales podrían suceder inter muestreo. A pesar de ello, es necesario evaluar un algoritmo de detección de eventos, que inicie el proceso de captura, cálculo y clasificación.

El algoritmo resta a cada valor de la ventana de muestras del tamaño elegido (teniendo en cuenta también el número de muestras variable en función del subsampleo aplicado) el valor de la mediana de la propia ventana, eliminado la componente continua. Posteriormente se suman los valores absolutos de la totalidad de las muestras de la ventana, comparándolas con el resultado de la ventana anterior y según que el resultado de esta comparación esté por encima del valor positivo de un umbral o por debajo del valor negativo de ese umbral se habrá producido una conexión o una desconexión de alguna carga, respectivamente.

Podría darse el caso de que el periodo de captura coincidiera con el transitorio, por lo que al extraerse características únicamente del periodo permanente, habrá que elegir otra ventana temporal como muestra. Esto se realiza mediante una comparación con gradiente, comparando una ventana con la siguiente de forma que se supone que la señal ha alcanzado el régimen estable si entre dos ventanas consecutivas la diferencia de su suma es menor que un valor denominado gradiente (ANEXO V).

En la figura 34 se muestra un ejemplo de la captura discontinua enventanada, en la que cada rectángulo amarillo corresponde a un periodo de muestreo. Para el caso de una detección de evento conexión, la ventana quedaría en color verde y de color rojo para la desconexión.

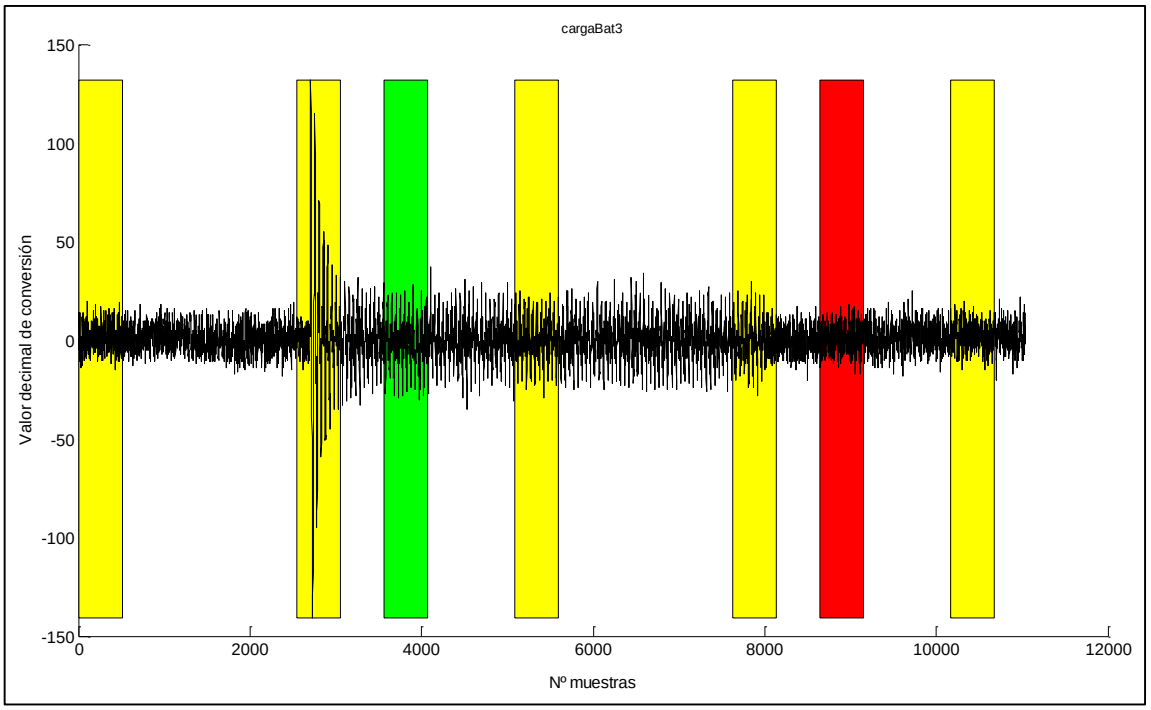

Fig. 34 Ejemplo de muestreo discontinuo, carga simple.

Puede observarse que la ventana elegida como muestra en la conexión, no es la correspondiente por modulación, sino que se encuentra en la zona de onda ya estabilizada para los dos casos de evento conexión (verde) y desconexión (roja).

Igualmente sucede con otros eventos en cargas agregadas. La figura 35 recoge estos eventos.

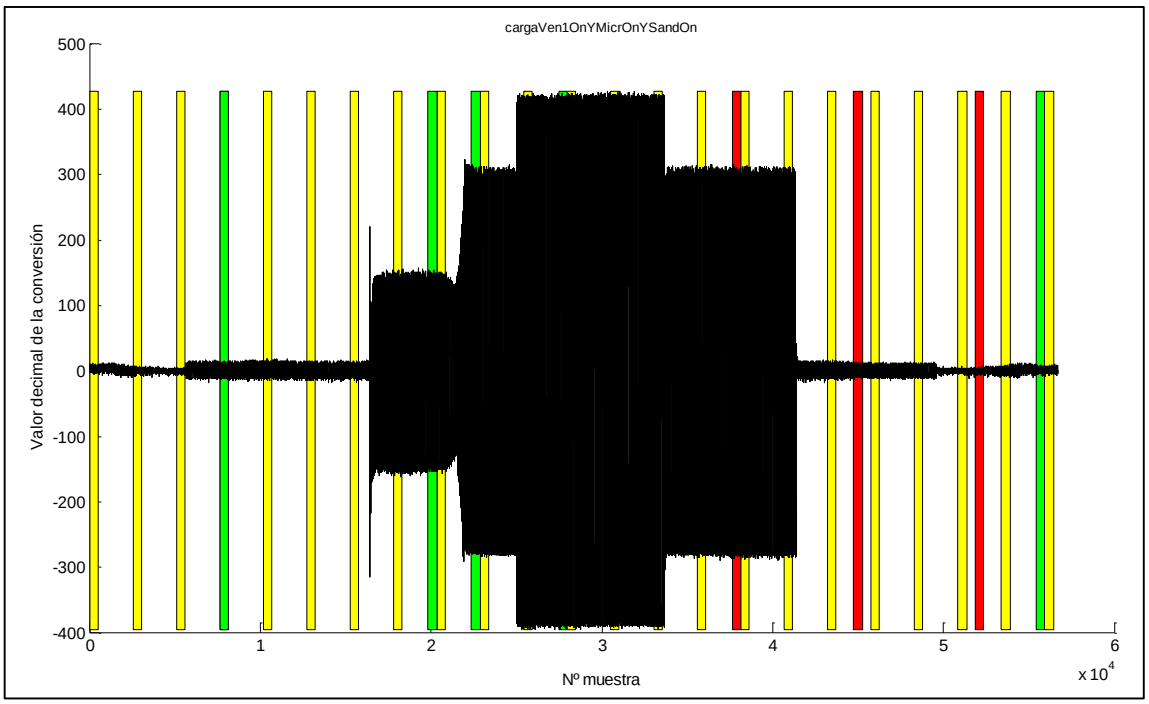

Fig. 35 Ejemplo de muestreo discontinuo, carga compuesta.

En este caso la tercera ventana de conexión corresponde al segundo estado de funcionamiento del microondas (el segundo evento detectado correspondería al primer estado de conexión del microondas).

También se pueden observar una falsa conexión al final del primer tercio de la onda correspondiente a la conexión de la sandwichera y otra falsa conexión al finalizar la captura del trial.

4.6. SOLUCION ADOPTADA

Tras presentar las medidas adoptadas encaminadas al uso de un sistema de bajo consumo y baja complejidad computacional, se eligen las soluciones y parámetros para evaluar las posibles redes neuronales (tabla 11).

FRECUENCIA MUESTREO	625 Hz
SUBSAMPLEO	1/4
TAMAÑO DE INDIVIDUO	10 CICLOS
FRECUENCIA INTERMUESTREO	1 Hz
VENTANA DETECTOR EVENTOS	10 CICLOS=130MUESTRAS
THRESHOLD EVENTO	270 UDS. CONVERSION
GRADIENTE INTER VENTANAS	70 UDS. CONVERSION
DATA SET ENTRENAMIENTO/VERIFICACION	80% / 20%
VALOR RMS	INCLUIDA
CENTROIDE CONCORDIA COMPONENTE α	INCLUIDA
CENTROIDE CONCORDIA COMPONENTE β	INCLUIDA
PRIMER ARMONICO	DESECHADA
TERCER ARMONICO	DESECHADA
QUINTO ARMONICO	DESECHADA
DESFASE TENSION CORRIENTE	INCLUIDA
VALOR PICO	INCLUIDA
VALOR INTER SEMIPERIODO	DESECHADA

Tab. 11 Configuracion de los parámetros de extracción de datos.

TDC

5. REDES NEURONALES

Los datos han sido procesados, elegidas las características que serán la entrada de la red neuronal y configurado el entorno de adquisición así como definido el parámetro que nos evaluará las redes neuronales puestas a prueba.

Las redes evaluadas son:

- SOM (Self Organizing Maps).
- PERCEPTRON LINEAL.
- PERCEPTRON MULTICAPA (MLP).
- APRENDIZAJE DE CUANTIFICACIÓN VECTORIAL(LVQ).

5.1. RED MAPA AUTO ORGANIZADO(SOM)

Un mapa auto-organizado (SOM por sus siglas en inglés) es un tipo de red neuronal artificial, que es entrenada usando aprendizaje no supervisado para producir una representación discreta del espacio de las muestras de entrada, llamado mapa (figura 36) (ANEXO II líneas 1831-1836).

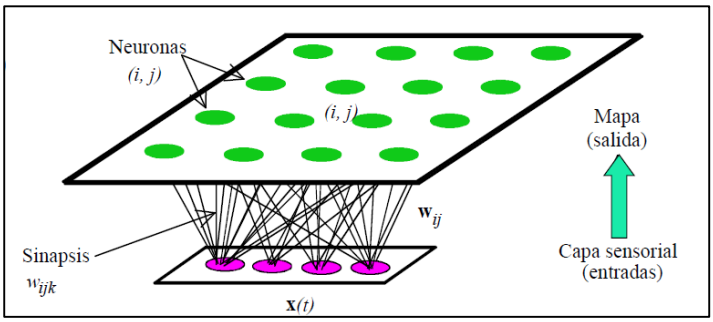

Fig. 36 Esquema de una red SOM.

Este mapa una vez entrenado dispondrá de diferentes grupos de neuronas capaz de activarse en función de las entradas mostradas, generando zonas definas para entradas similares. De esta forma, ante una entrada concreta se activarán las neuronas más afines que identificarán qué tipo de entrada es.

Las características de la red neuronal usada son:

- Inicialización de los pesos aleatoria.
- Modo de entrenamiento batch.
- Tamaño del mapa 26x20 neuronas.
- Disposición hexagonal.

Una vez entrenado el SOM con el grupo de individuos de entrenamiento, elegidos aleatoriamente con la proporción ya indicada se etiquetan los grupos de neuronas de forma que una vez generados los dominios se pueda saber a qué clase pertenecen las neuronas activadas y se presentan los individuos de verificación. Los resultados se exponen a continuación.

En la figura 37 se aprecia la topología del mapa, con cada celda representando una neurona. Las líneas más oscuras tanto de la U-matrix como de la D-matrix, definen regiones, grupos de neuronas que se activan ante cierto tipo de clases. Los marcadores de tamaño y la similitud por coloración representan el mismo fenómeno, pero con otra representación.

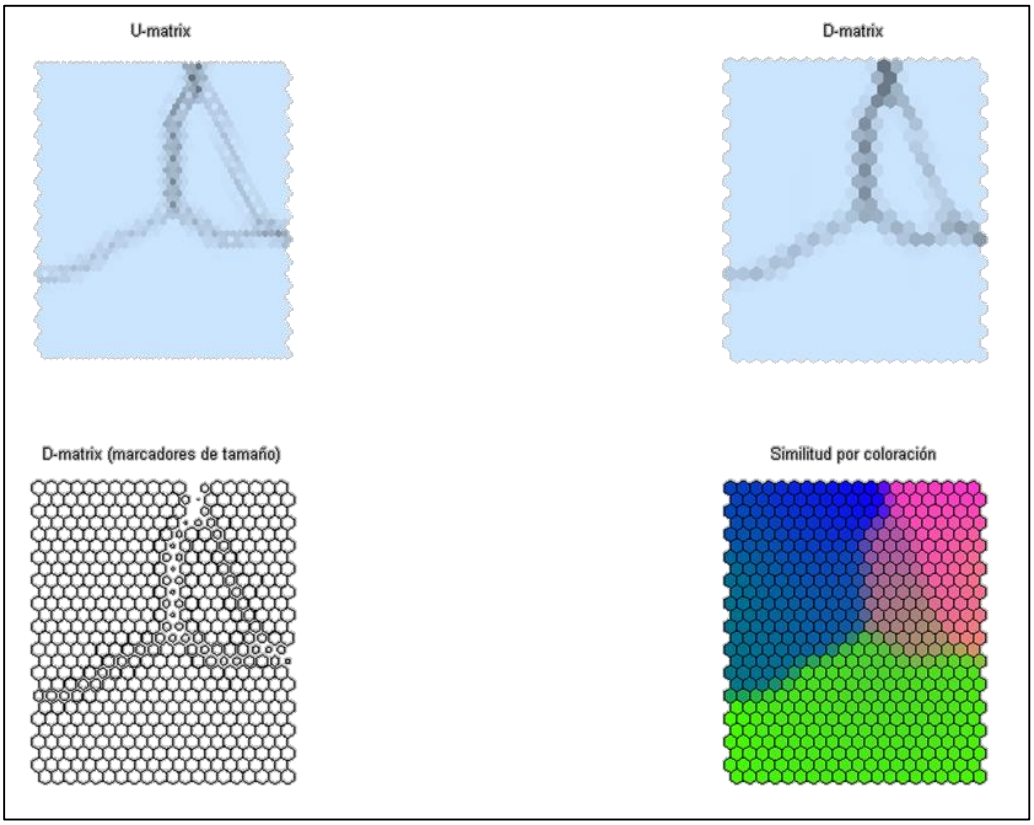

Fig. 37 Dominios del mapa auto organizado en diferentes vistas

En la figura 38 Se observa la activación por las clases de individuos. Las clases más similares (BAT1,BAT2 y BAT3 por ejemplo) se encuentran en la misma zona, sin separación bien definida.

Sin embargo, las clases compuestas MicroOnySandOn no activan la unión de las neuronas de las clases primitivas (Micr y Sand).

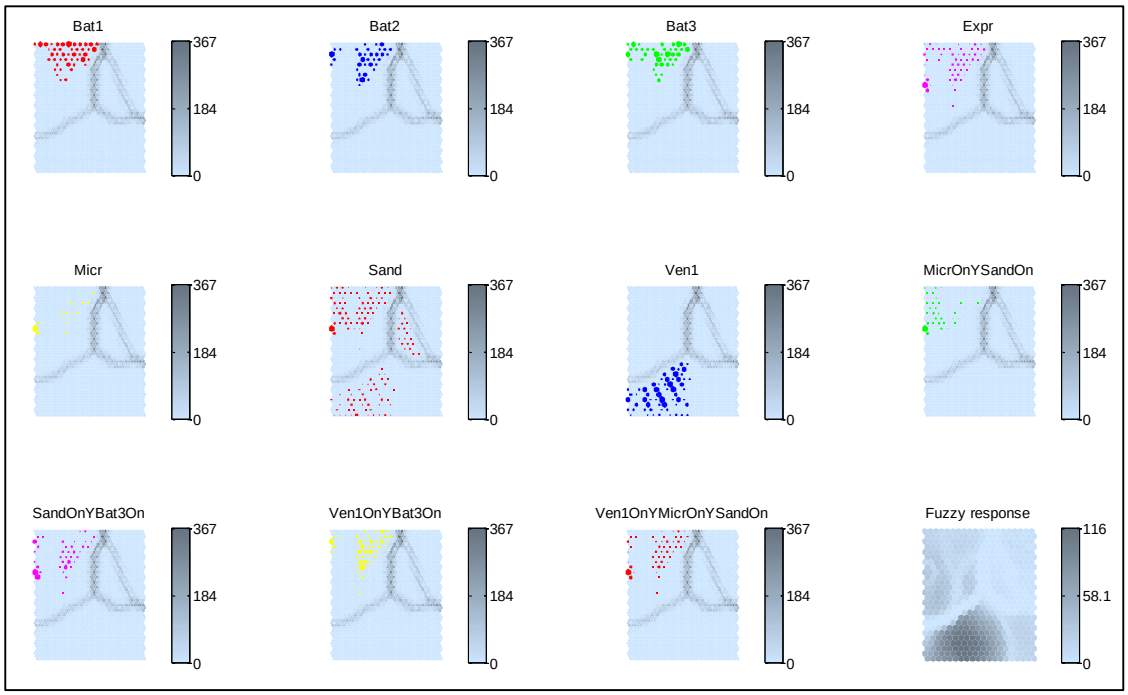

Fig. 38 Activación por clases de individuos.

En cuanto a la activación por características, se puede observar el resultado en la figura 39. Las zonas de activación por características son muy similares.

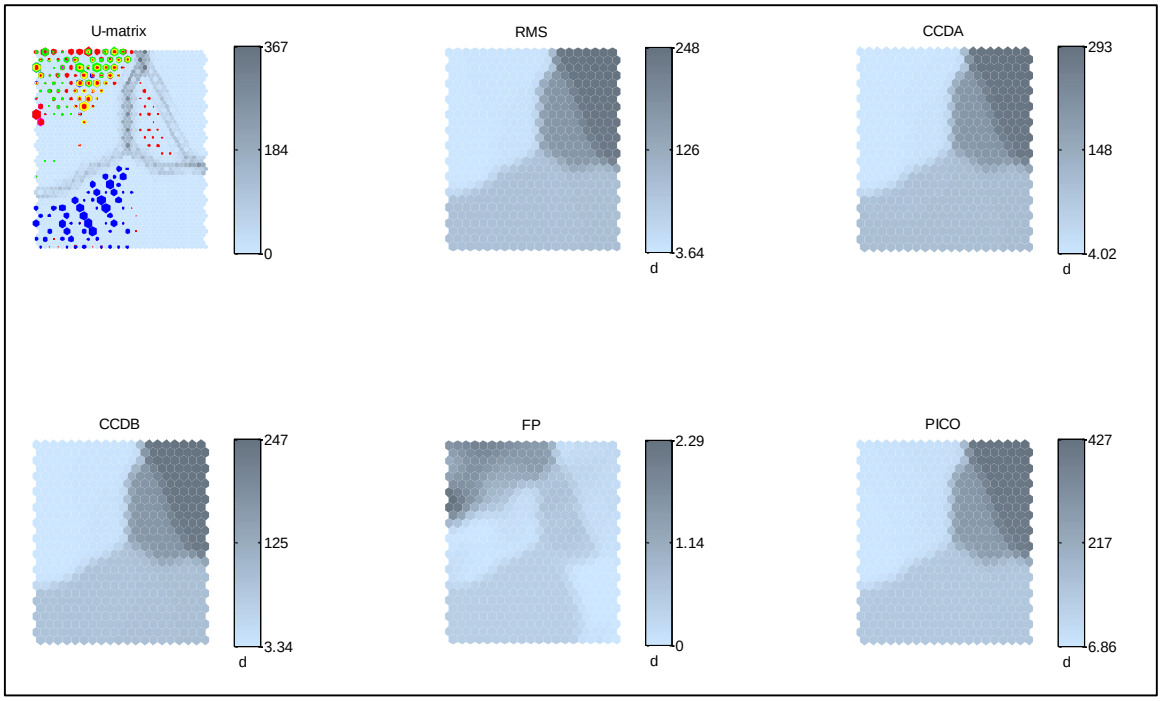

Fig. 39 Activación por características.

Para esta red neuronal se ha extraído la matriz de confusión en la tabla 12:

	Bat1	Bat2	Bat3	Expr	Micr	Sand	Ven1	MicrOnYSandOn	SandOnYBat3On	Ven1OnYBat3On	Ven1OnYMicrOnYSandOn
Bat1	15	4	0	1	0	0	0	0	0	0	0
Bat2	4	11	0	0	0	0	0	0	0	0	0
Bat3	1	0	15	1	0	0	1	0	0	0	0
Expr	2	0	0	36	0	0	0	0	0	0	0
Micr	0	0	0	0	19	0	0	0	0	0	0
Sand	0	0	0	0	0	91	0	0	7	0	0
Ven1	0	0	1	0	0	0	33	0	0	4	0
MicrOnYSandOn	0	0	0	0	0	0	0	20	0	0	22
SandOnYBat3On	0	0	0	0	0	8	0	0	14	0	0
Ven1OnYBat3On	0	0	0	0	0	0	8	0	0	17	0
Ven1OnYMicrOnYSandOn	0	0	0	0	0	0	0	9	0	0	11
TOTAL INDIVIDUOS	**68%**	**73%**	**94%**	**95%**	**100%**	**92%**	**79%**	**69%**	**67%**	**81%**	**33%**

Tab. 12 Configuración de los parámetros de extracción de datos.

Se puede observar la similitud de clases en los errores cometidos en la clasificación, al observar la simetría respecto de la diagonal, marcados con flechas dobles, en las que aunque las cantidades no coinciden dan una idea de que individuos que debían estar marcados como una clase son marcados como la otra y viceversa.

Para este caso, el índice Kappa obtenido es: 0.7937

5.2. RED LINEAL PERCEPTRON

La red lineal de perceptores se caracteriza por disponer de dos capas (figura 40), que procesan secuencialmente. La primera de ellas correspondería a las entradas con los pesos correspondientes y la segunda capa la capa de salida que en este caso clasificarían a cada individuo, realizada con neuronas logarítmicas sigmoideas (ANEXO II líneas 1865-1898).

Es claro que al estar fijadas el número de entradas y salidas, la red es invariante respecto al número de neuronas.

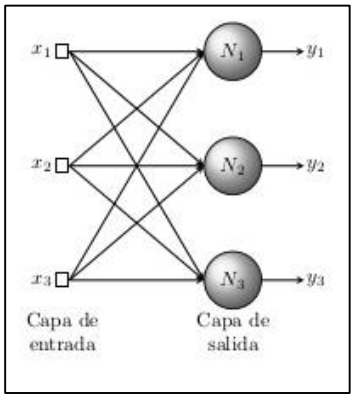

Fig. 40 Esquema red neuronal lineal.

Una peculiaridad de estas redes es que existe el peligro del sobre entrenamiento. Esto ocurre por un exceso de presentaciones de los mismos patrones a la red, lo que produce un sobre ajuste y memorización de los patrones presentados a la red, empeorando la capacidad generalizadora de la misma. Para evitar esto se usa la técnica de la “parada anticipada” que previene este hecho. Más adelante se usará esta técnica.

En primera instancia, para seleccionar el tipo de entrenamiento, se elige una red que se someterá a diversos tipos de entrenamientos.

Las características de la red neuronal usada son:

- Inicialización de los pesos aleatoria.
- Esquema 5-11 (nºNeuronasEntrada - nºNeuronasCapaSalida)
- Modo de entrenamiento: variable
- Tipos de neuronas: Capa salida: logarítmica sigmoidea.
- Numero de ciclos:500.
- Error objetivo: 5e-4.

Mediante un algoritmo de repetición se entrena y valida la red neuronal variando el algoritmo de entrenamiento. A pesar de ser pocas iteraciones, el resultado será suficiente para poder elegir el entrenamiento. Para conseguir un valor fiable, con cada tipo de entrenamiento se repite el proceso 5 veces obteniendo el resultado en la figura 41.

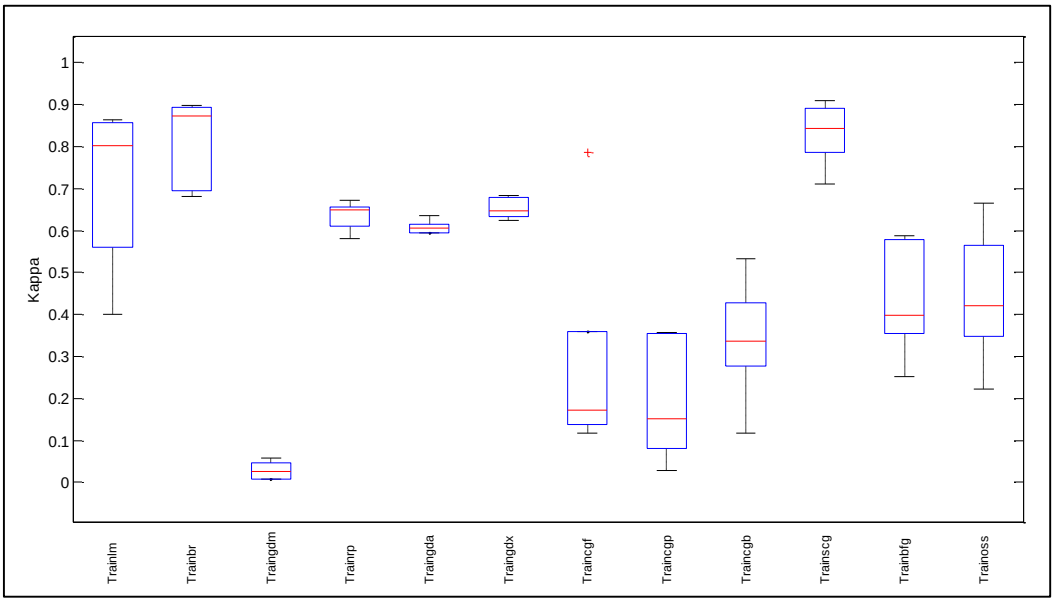

Fig. 41 Boxplot del indice Kappa en función del tipo de entrenamiento.

La equivalencia de entrenamientos es la siguiente.

Trainlm:	Levenberg-Marquardt
Trainbr:	Regularización Bayesiana
Traingdm:	Descenso del gradiente con momento
Trainrp:	Descenso del gradiente con derivadas signo (DG flexible)
Traingda:	Descenso del gradiente con factor de aprendizaje adaptable
Traingdx:	Descenso del gradiente con factor de aprendizaje adaptable y momento
Traincgf:	Descenso del gradiente conjugado, método de Fletcher-Reeves
Traincgp:	Descenso del gradiente conjugado, método de Polack-Riviere
Traincgb:	Descenso del gradiente conjugado, método de Powell-Beale
Trainscg:	Descenso del gradiente conjugado, método de escalado
Trainbfg:	Método cuasi-Newton BFGS
Trainoss:	Método cuasi-Newton de secante a un paso

Obtenido el resultado se escoge como método de entrenamiento Levenberg-Marquardt. A la vista de la varianza de los resultados obtenidos, se opta por obtener un pool de redes entrenadas y escoger la mejor de ellas (figura 42). En este caso sí que se ha tenido en cuenta el sobre entrenamiento y se ha aplicado la parada anticipada, con un valor máximo 10 ciclos de entrenamiento seguidos donde el conjunto de validación genere errores crecientes o constantes.

Para poder realizar esta técnica el dataset se ha divido en tres grupos como sigue:

- Grupo entrenamiento:60%.
- Grupo validación: 20%.
- Grupo test:20%

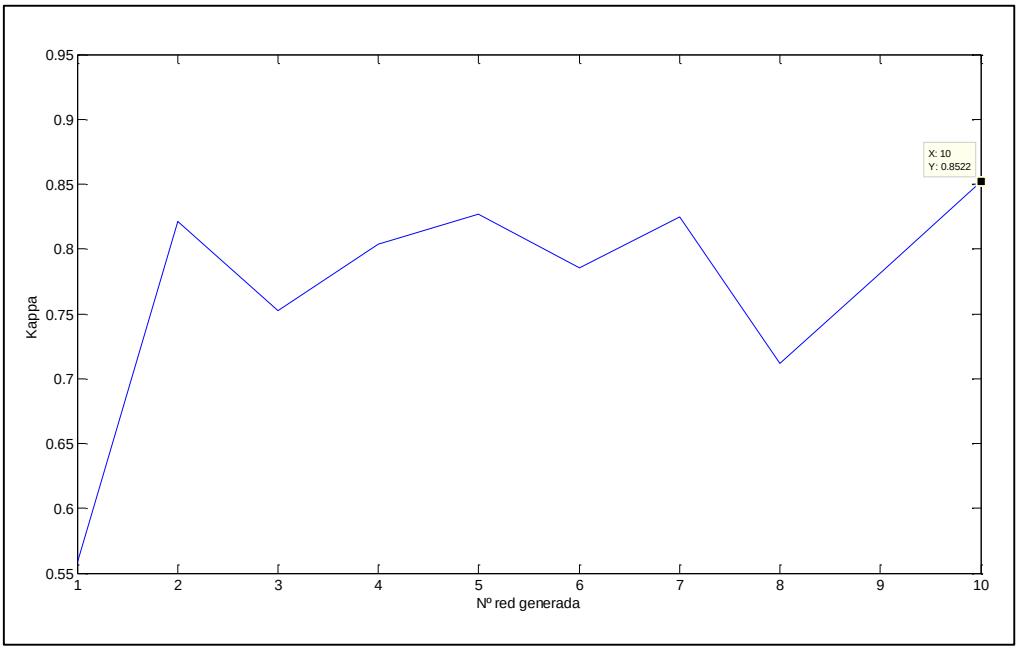

Fig. 42 Pool de redes generadas.

La red con mejores resultados es la número 10 con un Kappa de 0.8522 de la que se expone la matriz de confusión en la figura 43.

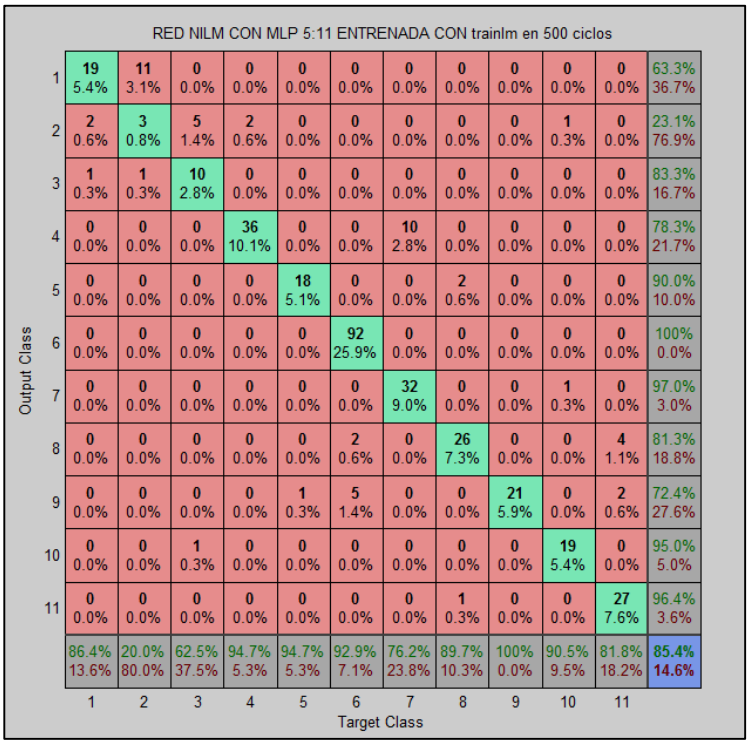

Fig. 43 Matriz de confusión de la red escogida.

En este caso concreto se ha detenido el entrenamiento debido a que la red ha alcanzado el valor de ciclos previsto por la parada anticipada. En la figura 44 se observa el error de la red para los conjuntos de entrenamiento, validación y test; en el ciclo 12 se observa el menor error del grupo de validación. A partir de ese ciclo, el error se mantiene constante o aumenta, indicado sobre entrenamiento

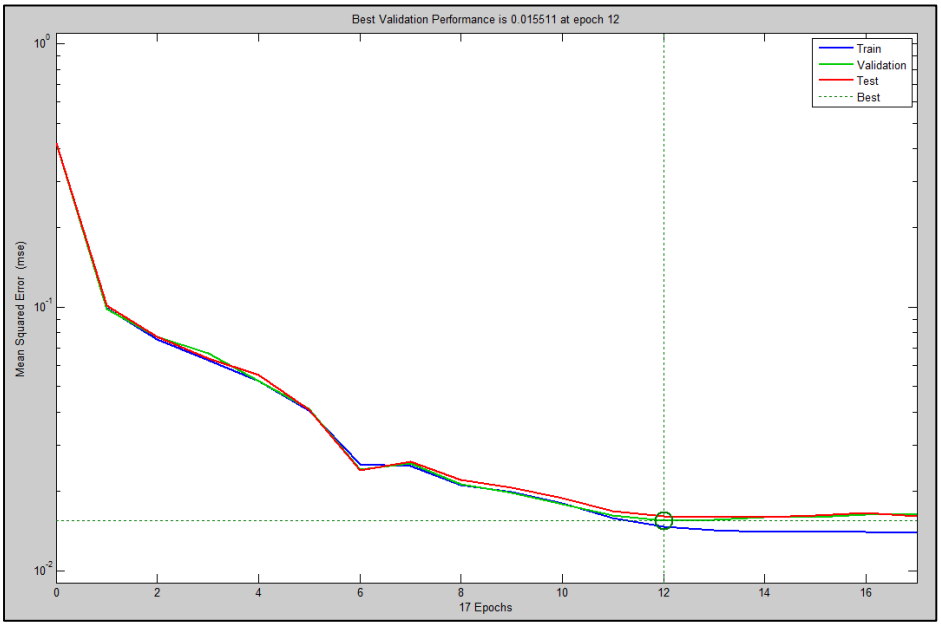

Fig. 44 Error generado por los tres grupos de datos.

5.3. RED PERCEPTRON MULTICAPA (MLP)

Otra red neuronal que se prueba es el perceptrón multicapa, red neuronal de aprendizaje supervisado, en la que las neuronas se disponen tal y como indica la figura 45, pudiendo existir varias capas intermedias. La capa de entrada tiene por dimensión (número de neuronas) igual a las características electas, las capas intermedias tiene un número no definido de neuronas y la de salida igual a las clases que esperamos clasificar. En este caso el número es conocido, puesto que es supervisado. (ANEXO II líneas 1901-1940)

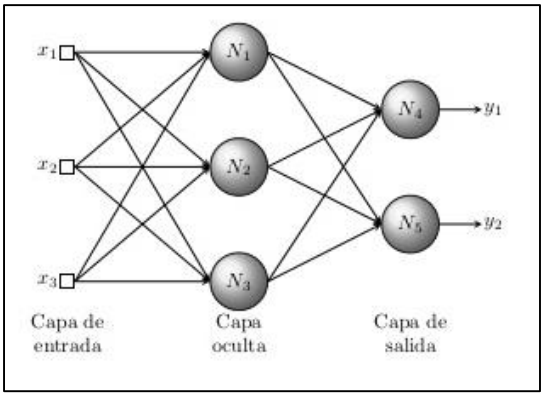

Fig. 45 Esquema de perceptron multicapa.

Las características de la red neuronal usada son:

- Inicialización de los pesos aleatoria.
- Esquema 5-XX-11 (nºNeuronasEntrada-nºNeuronasCapaOculta-nºNeuronasCapaSalida)
- Modo de entrenamiento: backpropagation con función de entrenamiento: Levenberg-Marquardt.
- Tipos de neuronas: Capa intermedia: tangente sigmoidea; capa salida: lineal.
- Numero de ciclos:1000.
- Error objetivo: 5e-4.
- Parada anticipada:5

Según [XX] un número de neuronas para la única capa intermedia estaría alrededor del doble de la de neuronas en la capa de salida. Se realizan pruebas para determinarlo mediante iteraciones de entrenamiento de la siguiente manera.

Mediante un algoritmo de repetición se entrena y valida la red neuronal variando el número de neuronas ocultas, con un máximo de 1000 iteraciones de entrenamiento Levenberg-Marquardt. Se incluye nuevamente la parada anticipada para el resultado será suficiente para poder elegir configuración. Para conseguir un valor fiable, con cada valor de neuronas en la capa oculta se repite el entrenamiento y validación 10 veces obteniendo una mediana de los distintos valores. El resultado puede observarse en la figura 46

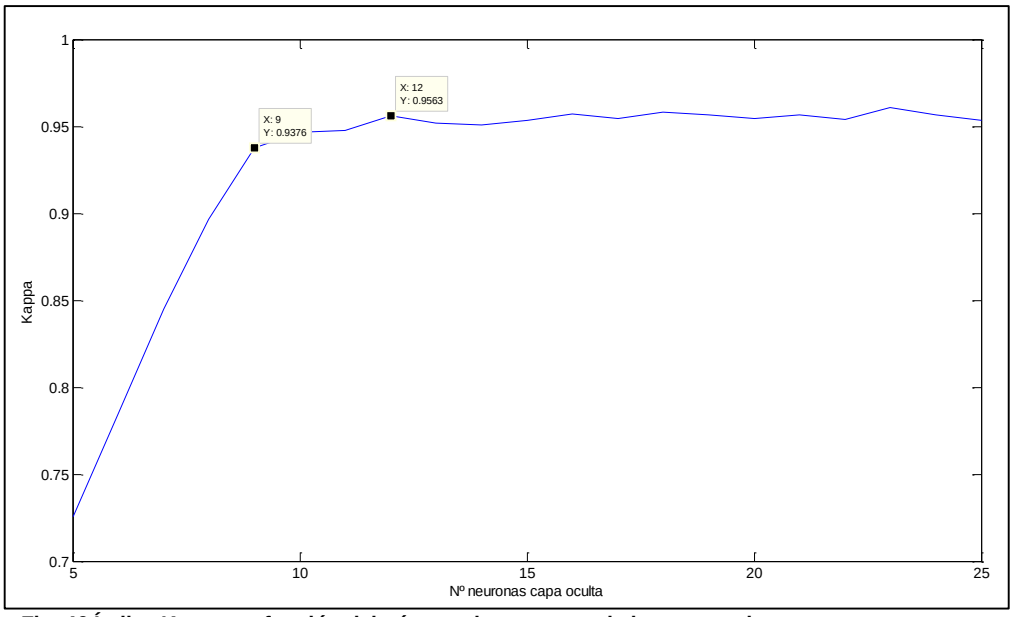

Fig. 46 Índice Kappa en función del número de neuronas de la capa oculta.

Se aprecia una tendencia clara conforme el aumento de neuronas ocultas hasta un valor cercano a los 10, tal y como avanzaba [XX], lo que indica que tras alcanzar un valor de unas 10 neuronas, sobrecargar la red neuronal con una capa oculta más compleja, no mejora mucho la clasificación.

Definido el mínimo de neuronas ocultas para un clasificación aceptable, se determina si con algunas neuronas más el sistema estaría sobre dimensionado. El entrenamiento por regularización bayesiana nos indicará cuantos pesos realmente aportan información a la clasificación; de esta manera se puede ver si el sistema está sobredimensionado ya que informaría de un bajo número de pesos usados. El proceso se repite 5 veces y se obtiene la mediana; el resultado obtenido informa que de 181 pesos en la red, se usan 172, lo que indica que no hay sobredimensionamiento. Si el número de pesos efectivos fuera mucho menor que el de pesos totales se podría hablar de red sobredimensionada.

A la vista de los resultados se elige la configuración 5-10-11 (5 entradas, 10 neuronas en la capa oculta y 11 neuronas de salida). Con la configuración elegida, se entrena nuevamente un pool de redes MLP con las características antes descritas, y con 1000 iteraciones. Extrapolando los resultados de entrenamientos del punto anterior, se elige el mismo tipo de entrenamiento, Levenberg-Marquardt. El resultado de este pool es el de la figura 47.

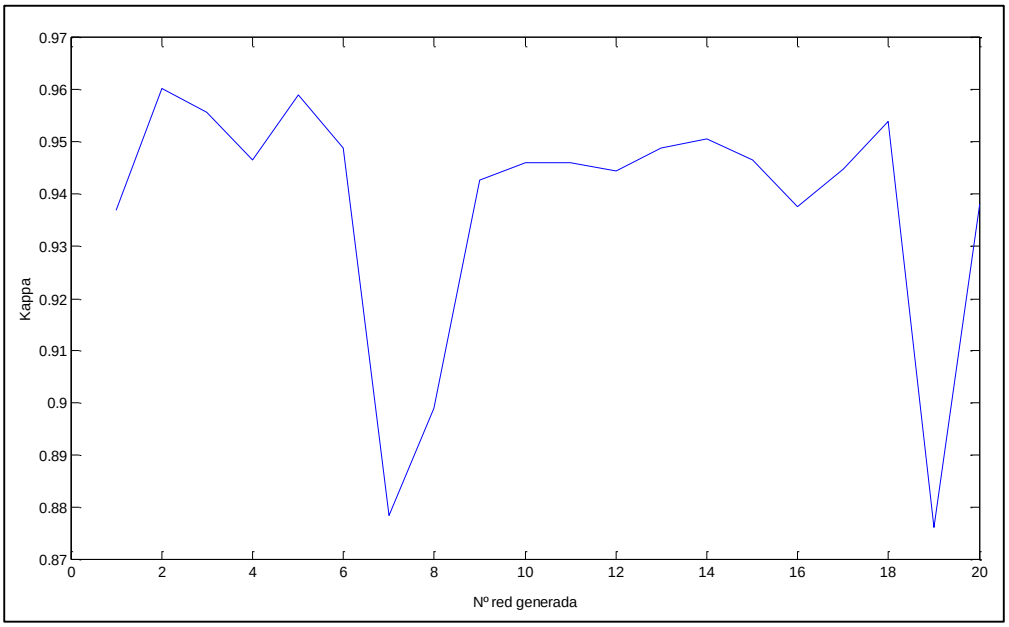

Fig. 47 Resultados del pool de redes generadas

Para obtener un valor aproximado y comparable de la capacidad de clasificación de este pool, es necesario promediar el resultado, lo que da un kappa de 0.9461.

Se muestra a continuación en la figura 48 la matriz de confusión de la red de perceptrones elegida y el diagrama de barras en la figura 49 para cada clase e individuo.

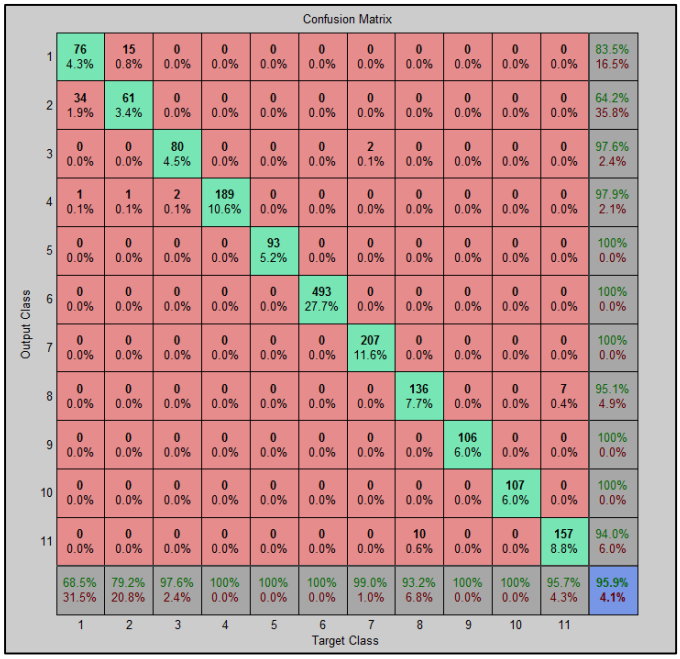

Fig. 48 Matriz de confusión de la red MLP 5-10-11.

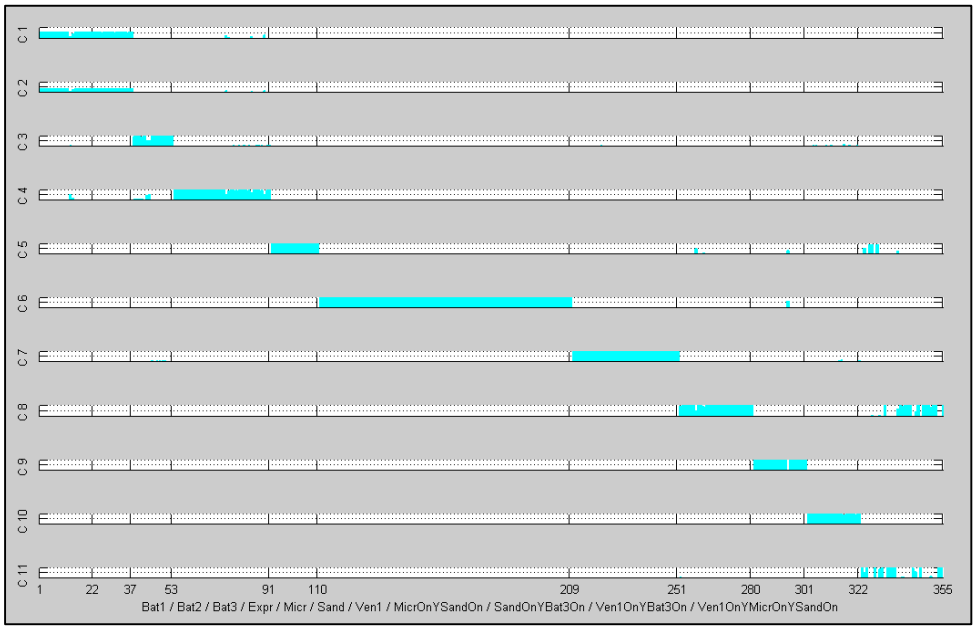

Fig. 49 Grafico de barras de la red MLP 5-10-11.

La figura del error conforme se ha ido entrenando la red puede verse en la figura 50

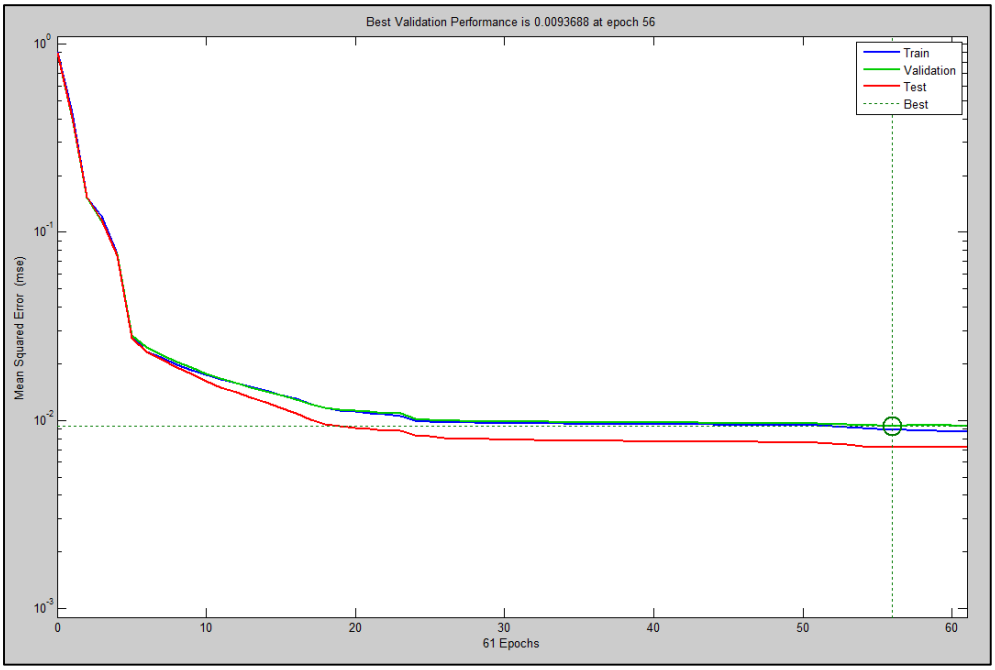

Fig. 50 Error de la red conforme avanza el entrenamiento.

La parada anticipada puede observarse en la siguiente figura (figura 51)

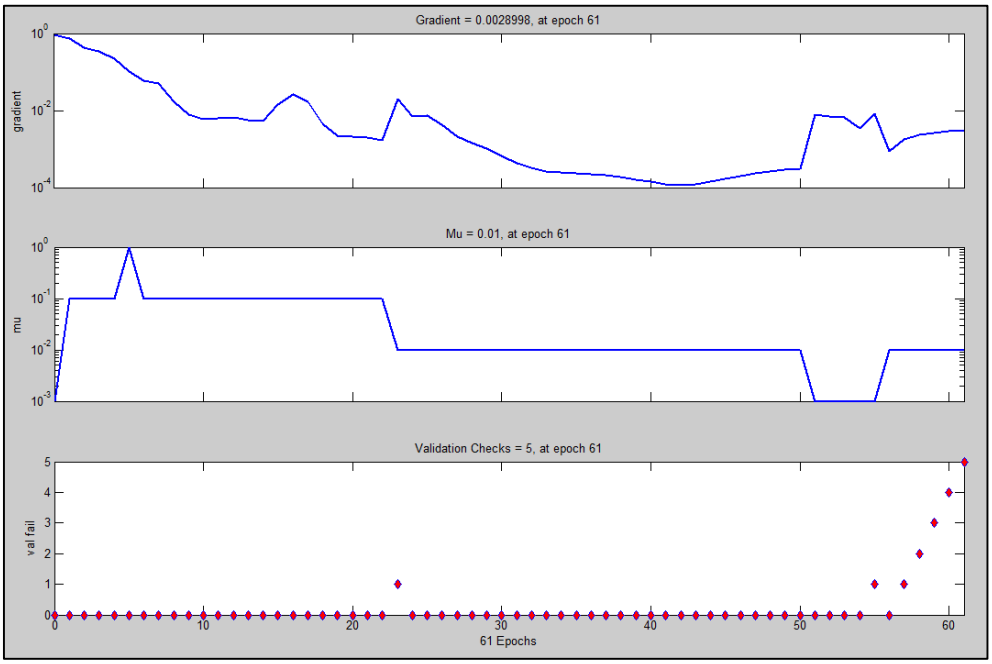

Fig. 51 Evolución de parámetros de la red.

Puede observarse cómo el gradiente va disminuyendo hasta que a partir de la iteración 56 (aumento del número de iteraciones con gradiente no descendiente) no continua el descenso, por lo que se decide parar el entrenamiento.

5.4. RED LEARNING VECTOR QUANTIZATION (LVQ)

Una red LVQ es parecida a una red SOM en cuanto a su funcionamiento y aprendizaje, pero está más orientada a la clasificación y entrenamiento supervisado, necesitando menos neuronas, siendo por ello un modelo más sencillo y eficaz (ANEXO II líneas 1942-1961).

El esquema de la red es similar al perceptrón simple (red lineal) de la figura 52:

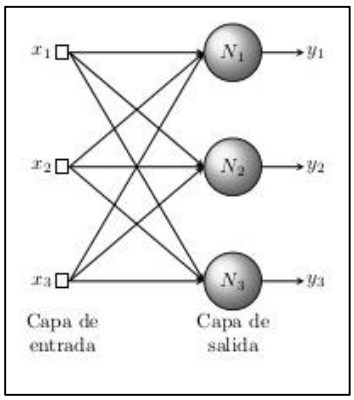

Fig. 52 Esquema red LVQ.

Esta red presenta también el inconveniente de la elección del número de neuronas de salida, puesto que aunque podría funcionar con una neurona de salida por clase, aumentar el número de neuronas por clase, mejora la clasificación.

Por ello se realiza una repetición sistemática de resultados de la red variando el número de neuronas de salida con el objetivo de seleccionar dicho número. Para esta aproximación se realizan 3 repeticiones medianadas en la red con los siguientes parámetros:

- Inicialización de los pesos aleatoria.
- Esquema 5-XX (nºNeuronasEntrada-nºNeuronasCapaSalida)
- Modo de entrenamiento: Entrenamiento aleatorio de orden incremental.
- Función de aprendizaje:LVQ1
- Numero de ciclos:20.
- Factor aprendizaje: 0.2
- Error objetivo: 5e-4.

El resultado puede observarse en la siguiente gráfica (figura 53)

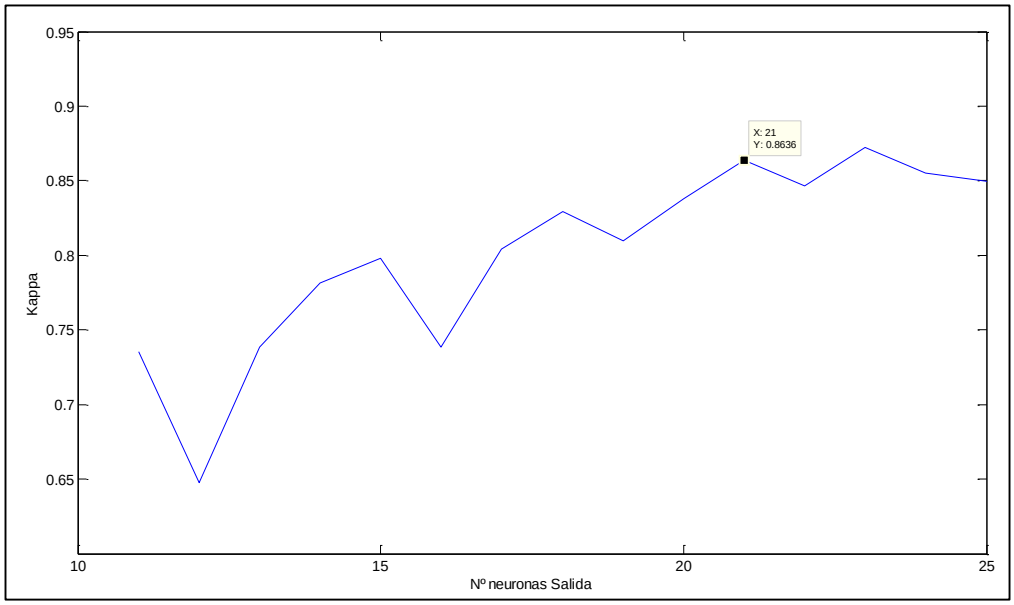

Fig. 53 Kappa en función de neuronas de salida.

Puede observarse el incremento del índice Kappa al aumentar el número de neuronas. Indicar que por debajo de 20 neuronas hay clases que la red no habría podido clasificar ningún individuo, por lo que se elige un valor superior, 21.

A la vista de los resultados, se entrena una red LVQ con los parámetros descritos y esquema 5-21. Según expone [KOHONEN 95] el entrenamiento mediante el algoritmo LVQ1 no debe superar de 30 a 50 veces el muestrario de entrenamiento (el 80% del total de las muestras, o sea 1422 muestras). En este caso se opta por una posición conservadora eligiendo 300 ciclos de entrenamiento.

Se expone la matriz de confusión (figura 54) y el historial del error a lo largo del entrenamiento (figura 55).

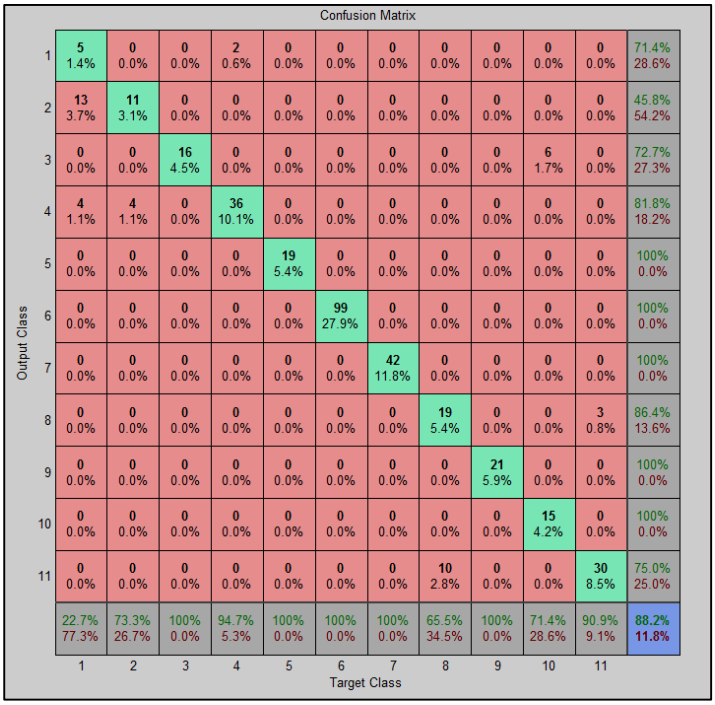

Fig. 54 Matriz de confusión de la red LVQ.

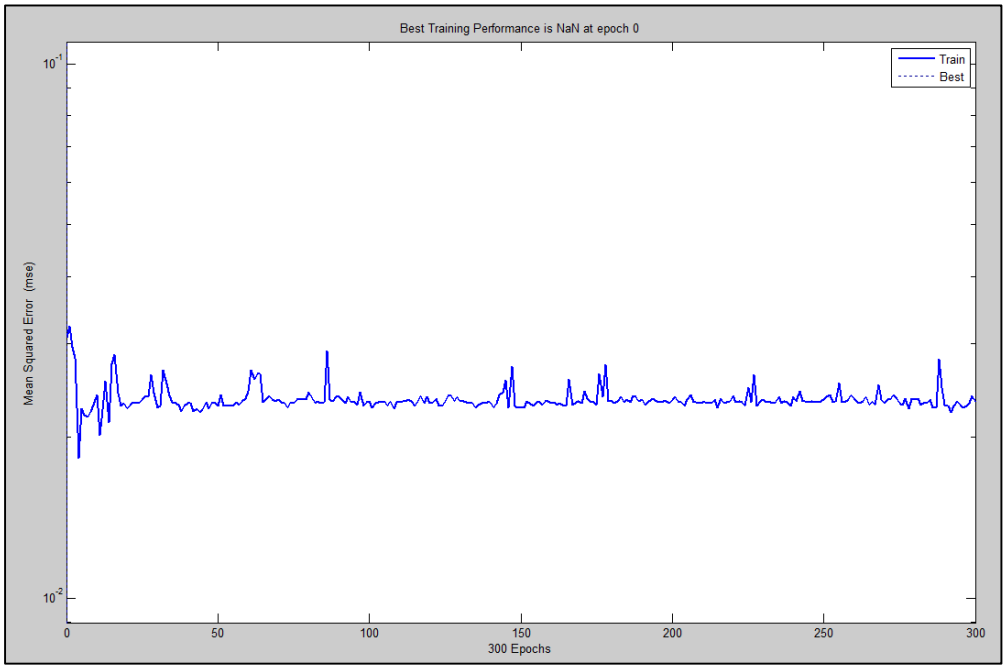

Fig. 55 Evolución del error durante el entrenamiento.

Con un Kappa de 0.8807

5.5. GASTO COMPUTACIONAL

Cada neurona responde a un esquema como el de la figura 56.

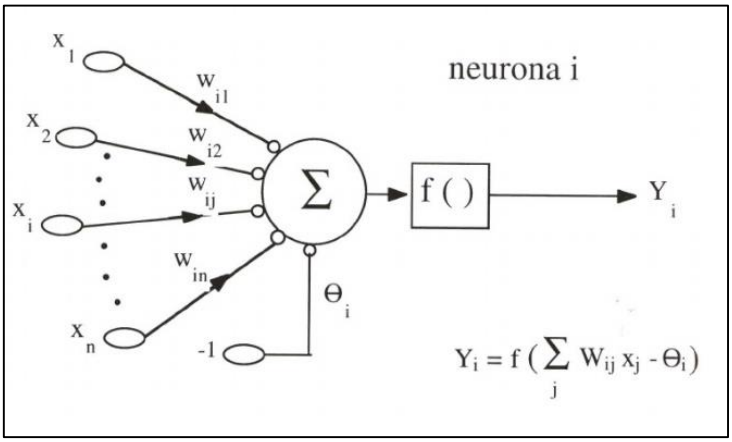

Fig 56 Esquema de computación de una neurona genérica.

Al resultado de multiplicar las entradas por sus pesos más el bías o valor de activación, se le aplica la función de activación, tal y como indica la ecuación:

$$y(t) = f_i\left(\sum_j w_{ij}x_j - \theta_i\right)$$

Donde

- i: es la neurona
- j:es la entrada correspondiente
- w: es el valor del peso
- x: es el valor de la entrada
- θ: es el valor de activación
- f: función de activación (lineal, sigmoidea, etc.)

Por simplificación en la representación, muchas veces el bias es considerado una entrada más, con el signo negativo correspondiente.

Dentro del funcionamiento de la red, hay una relación directa entre el número de pesos y el número de operaciones que debe realizar el sistema. En esta tabla 13 se resumen las redes neuronales desde el punto de vista computacional y la relación Kappa-número de pesos, que nos indica el rendimiento de cada red

RED	ESQUEMA	NEURONAS	PESOS	KAPPA	KAPPA/PESOS
SOM	26x20	520	3120	0.7937	2.54 e^{-4}
PERCEPTRON LINEAL	5-11	16	60	0.8522	142 e^{-4}
MLP	5-10-11	26	181	0.9461	52.27 e^{-4}
LVQ	5-21	26	126	0.8807	69.89 e^{-4}

Tab. 13 Redes neuronales y rendimiento computacional.

Dado que por cada peso se ha de realizar una multiplicación, desde el punto de vista de rendimiento energético la red que mejor resuelve el problema con menor coste es el perceptrón lineal.

Por el contrario, el algoritmo de entrenamiento puede requerir incluso mayores recursos que la propia red. En el caso del perceptrón multicapa, este entrenamiento superaría los requisitos de cualquier microprocesador de rango medio, quedando excluido el entrenamiento de la integración en el dispositivo, debiéndose entrenar en un PC para luego clonar la red entrenada en el dispositivo.

TDC

6. CONCLUSIONES

6.1. RESUMEN

En este trabajo fin de máster se ha partido de los sistemas no intrusivos para el reconocimiento de cargas eléctricas, que hasta ahora han sido sistemas de grandes prestaciones, voluminosos, caros y de grandes requerimientos energéticos, usados y afinados para obtener los mejores resultados clasificatorios, para avanzar en la dirección de las tendencias actuales de los sistemas de monitorización sensado y procesado: nodos cada vez más independientes, muy eficientes energéticamente hablando y distribuidos.

Con la premisa de usar sistemas de bajo consumo, pequeño volumen y bajo costo, se han elegido características de la onda de corriente que necesiten de poco cómputo, y se han ajustado parámetros del funcionamiento del sistema como la frecuencia de muestreo y el algoritmo de detección de eventos, obteniendo un compromiso entre energía consumida y exactitud en la clasificación.

Se han evaluado nuevas características capaces de suplantar la potente información proporcionada por la Transformada de Fourier, más compleja de obtener, y se han evitado conversiones a unidades físicas, usando siempre las unidades directas de conversión del propio conversor analógico digital. También se han detectado características que, bien no aportan nueva información, bien introducen ruido al sistema.

Se han elegido ciertas redes neuronales que por su simplicidad de implementación o por su potencia de clasificación resultan muy convenientes para estos sistemas descritos, y se han obtenido rendimientos de clasificación frente a computación

6.2. TRABAJOS FUTUROS

Existe un colectivo de personas centradas en esta faceta del reconocimiento de cargas de forma no intrusiva [NILM 16], que celebra congresos y disponen de data sets de libre acceso para investigadores. Es por tanto un área de estudio en vigencia.

Esta posible línea de trabajo queda abierta a muchos más proyectos, siguiendo la idea de un sistema clasificador no intrusivo de bajo consumo: desde la propia implementación de la red en el microcontrolador hasta optimizaciones funcionales de las redes.

Un siguiente paso muy concreto, siguiendo con el trabajo aquí expuesto, se centraría en que el sistema fuera capaz de detectar las cargas compuestas como sumas de dos o tres cargas, en lugar de como una única carga, disminuyendo la memoria necesaria, al reducir el número de clases (una carga compuesta no sería una nueva clase, sino combinación de las ya conocidas).

Existen muchos datos sobre el uso y consumo de electrodomésticos en media, por habitante, por país, etc. Sin embargo, es en la proximidad del individuo donde más útil resultará este tipo de información.

TDC

7. RESEÑAS.

[HART 92]: GEORGE W. HART, NONINTRUSIVE APPLIANCE LOAD MONITORING (1992). Proceedings of the IEEE

[NORFORD 96]: LESLIE K. NORFORD, NON-INTRUSIVE ELECTRICAL LOAD MONITORING IN COMMERCIAL BUILDINGS BASED ON STEADY-STATE AND TRANSIENT LOAD-DETECTION ALGORITHMS (17 Septiembre 1995). Energy and Buildings, Volume 24, Issue 1, 1996, Pages 51–64

[LAUGHMAN 03]: CHRISTOPHER LAUGHMAN, POWER SIGNATURE ANALYSIS (Abril 2003). IEEE power & energy magazine

[GILREATH 06]: PHIL GILREATH, A NOVEL TECHNIQUE FOR IDENTIFICATION AND CONDITION MONITORING OF NONLINEAR LOADS IN POWER SYSTEMS (2006). Power Electronics, Drives and Energy Systems, 2006. PEDES '06. International Conference on.

[LIN 10]: GU-YUAN LIN, APPLYING POWER METERS FOR APPLIANCE RECOGNITION ON THE ELECTRIC PANEL (Junio 2010). Industrial Electronics and Applications (ICIEA), 2010 the 5th IEEE Conference on.

[RAHIMI 11]: SABA RAHIMI, USAGE MONITORING OF ELECTRICAL DEVICES IN A SMART HOME (Agosto 2011). 33rd Annual International Conference of the IEEE EMBS Boston, Massachusetts USA.

[WANG 12]: ZHENYU WANG, RESIDENTIAL APPLIANCES IDENTIFICATION AND MONITORING BY A NONINTRUSIVE METHOD. IEEE transactions on smart grid, vol. 3, no. 1, March 2012

[YUN 12]: GAO YUN, NON-INTRUSIVE LOAD IDENTIFICATION BY FUZZY CLUSTER ANALYSIS BASED ON ACTIVE POWER (MARZO 2012). Power and Energy Engineering Conference (APPEEC), 2012 Asia-Pacific

[ROOS 94]: JG ROOS, USING NEURAL NETWORKS FOR NON-INTRUSIVE MONITORING OF INDUSTRIAL ELECTRICAL LOADS (Mayo 1994) Instrumentation and Measurement Technology Conference, 1994. IMTC/94. Conference Proceedings. 10th Anniversary. Advanced Technologies in I & M., 1994 IEEE

[PATEL 07]: SHWETAK N. PATEL, AT THE FLICK OF A SWITCH: DETECTING AND CLASSIFYING UNIQUE ELECTRICAL EVENTS ON THE RESIDENTIAL POWER LINE. 9th International Conference, UbiComp 2007, Innsbruck, Austria, September 16-19, 2007.

[YANG 07]: HONG-TZER YANG, DESIGN A NEURAL NETWORK FOR FEATURES SELECTION IN NON-INTRUSIVE MONITORING OF INDUSTRIAL ELECTRICAL LOADS (2007). Proceedings of the 2007 11th International Conference on Computer Supported Cooperative Work in Design.

[CHANG 10]: HSUEH-HSIEN CHANG, LOAD IDENTIFICATION IN NONINTRUSIVE LOAD MONITORING USING STEADY-STATE AND TURN-ON TRANSIENT ENERGY ALGORITHMS (Abril 2010). Proceedings of the 2010 14th International Conference on Computer Supported Cooperative Work in Design.

[TSAI 11] DEVELOPMENT OF A NON-INTRUSIVE MONITORING TECHNIQUE FOR APPLIANCE IDENTIFICATION IN ELECTRICITY ENERGY MANAGEMENT. The International Conference on Advanced Power System Automation and Protection (2011)

[ALAHMAD 11]: MAHMOUD ALAHMAD, NON-INTRUSIVE ELECTRICAL LOAD MONITORING AND PROFILING METHODS FOR APPLICATIONS IN ENERGY MANAGEMENT SYSTEMS (2011). Architectural Engineering -- Faculty Publications University of Nebraska.

[LIN 12]: YU-HSIU LIN, APPLICATION OF NEURO-FUZZY PATTERN RECOGNITION FOR NON-INTRUSIVE APPLIANCE LOAD MONITORING IN ELECTRICITY ENERGY CONSERVATION. WCCI 2012 IEEE World Congress on Computational Intelligence June, 10-15, 2012 - Brisbane, Australia.

[KOHONEN 95]: THE LEARNING VECTOR QUANTIZATION PROGRAM PACKAGE VERSION 3.1 (APRIL 7, 1995) Archivo de ayuda del programa LVQ_PAK de la Universidad de Helsinki

[NILM 16]: 3rd International Workshop on Non-Intrusive Load Monitoring (May 14 2016) Vancouver, Canada (http://nilmworkshop.org/2016/)

TDC

ANEXO I

PARÁMETROS DEL FILTRO DIGITAL

Se exponen a continuación los parámetros del filtro digital aplicado a la señal de corriente.

Discrete-Time FIR Filter (real)

Filter Structure : Direct-Form FIR
Filter Length : 101
Stable : Yes
Linear Phase : Yes (Type I)

Design Method Information
Design Algorithm : window

Design Options
ScalePassband : true
Window : hamming

Design Specifications
Sampling Frequency : 2.542 kHz
Response : Bandpass
Specification : N,Fc1,Fc2
FilterOrder : 100
Fcutoff1 : 40 Hz
Fcutoff2 : 60 Hz

Measurements
Sampling Frequency : 2.542 kHz
First Stopband Edge : Unknown
First 6-dB Point : 26.2517 Hz
First 3-dB Point : 33.0918 Hz
First Passband Edge : Unknown
Second Passband Edge : Unknown
Second 3-dB Point : 67.2323 Hz
Second 6-dB Point : 73.8605 Hz
Second Stopband Edge : Unknown
First Stopband Atten. : Unknown
Passband Ripple : Unknown
Second Stopband Atten. : Unknown
First Transition Width : Unknown
Second Transition Width : Unknown
Group delay response : 50 samples

Implementation Cost
Number of Multipliers : 101
Number of Adders : 100
Number of States : 100
Multiplications per Input Sample : 101
Additions per Input Sample : 100

TDC

ANEXO II

SCRIPT CAPTURA, ACONDICIONAMIENTO Y SIMULACION

```
clc;
clear all;
close all;



%% CONSTANTES
%%%%%
%%MASCARA ELECCION DE 11 ELECTRODOMESTICOS
%%%        BAT1 BAT2 BAT3 EXPR MICR SAND VENT  M_S  S_B3 V_B3 V_M_S
MASCARA_ELECTR=[1   1   1   1   1   1   1    1    1   1    1];

CARACTERISTICAS = {'RMS','CCDA','CCDB','FFT1','FFT2','FFT3','FP','PICO','CUARTO'};


Kappatemp=[];
subsampleVsKappa=[];
for subsamples=4:4                          %repeticiones con cambio de sampleo
                                  %(1=sin subsampleo)(4=> subsampleo = FS/4)
   for veces=1:1                        %repeticiones para media en la reduccion de entradas
      %adquisicion
      TRIALS=10;
      CLASES=11;
      %names = {'RMS','CCDA','CCDB','FFT1','FFT2','FFT3','FP','PICO','CUARTO'};
      VENTANA_EST=10;                   %número de ciclos que componenen un muestra
      %COMP_CONTINUA=600;
      % FS=2542; %Frecuencia de sampleo original
      SUBSAMPLE=subsamples;             %se guarda una de cada SUBSAMPLE muestras
      FS=round(2542/SUBSAMPLE);          %Frecuencia de sampleo
      MUESTRAS_CICLO=round(FS*0.02);      %número de adquisiciones en cada ciclo
      INC_VENT=VENTANA_EST*MUESTRAS_CICLO;   %desplazamiento

      %FFT
      T=1/FS;                           %periodo
      NFFT = 2^nextpow2(INC_VENT);          % Next power of 2 from length of y
      FRC_BUSQ_FFT=[40 60 120 180 200 300];  %rango frecuencias para buscar las CARACTERISTICAS FFT
      VECTOR_FRECUENCIA = FS/2*linspace(0,1,NFFT/2+1);


      %filtro paso banda
      N=100;          %orden del filtro
      FC1=40;          %Frecuencia de corte inferior
      FC2=60;          %Frecuencia de corte superior
      INICIO_FILTR=50;
      DESF_FILTRO=50;    %muestras de desfase entre señal original y filtrada.

      %Medida tension
      DESFASE_TRAFO_TENSION=0;            %desfase que introduce el trafo de medida de tension
      SEP_PASOS_0=round(MUESTRAS_CICLO/3.33); %umbral para aceptar pasos por cero
      MAY_MEN_0=0;

      %evento conexion-desconexion
      ANCHO_VENTANA=3*MUESTRAS_CICLO;                 %ancho ventana de evento a on
      THRESHOLD=120;                                          %umbral para considerar un evento a ON
      CUARTO_SEMIPERIODO= round (MUESTRAS_CICLO/11.6129);    %donde más se nota el 3º armonico, con 5º y 7º 10% del 3º
      VALOR_CUARTO_SEMIPERIODO_NORMALIZADO=0.4226;          %Tanto por uno del valor de pico que se obtendría en un
```

```
                                                % seno puro en el cuarto del segundo semiperiodo
PORC_ENTRENAMIENTO=80/100;                      %porcentaje del data set destinado al entrenamiento


%indices para recortar el fichero visualmente

inicioCargaBat1=[7449,5690,7410,4324,5272,4552,4192,3348,3962,7233,5232,3802,2829,3704,2859];
finCargaBat1=[13827,12302,12227,10597,12226,9589,11776,10693,9398,12556,11560,11375,7912,8637,7144];

inicioCargaBat2=[6228,4135,4921,4036,4153,3532,2820,1932,3812,3547,3325,3388,2844,2858,3862];
finCargaBat2=[14459,8466,9500,6876,8633,7632,6830,5611,7456,7548,7258,8170,8422,9084,8843];

inicioCargaBat3=[6142,4009,2984,4210,3188,3048,3694,4386,3372,3710,3297,2941,3570,2730,4218];
finCargaBat3=[14274,7892,6445,8744,7288,7207,7690,9216,7829,8591,8577,8470,8650,7460,9496];

inicioCargaExpr=[535,6210,1961,1157,2462,1529,1907,2151,1761,3037,102928,3228,5863,8080,3394];
finCargaExpr=[34486,15672,9335,10589,9787,7808,7831,9058,8935,10392,122613,9052,13974,16991,16840];

inicioCargaMicr=[7227,7261,7487,7120,6242,6041,7640,6068,6510,7082,10599,9468,8420,7638,7805,];
finCargaMicr=[12861,11299,13171,12554,9207,11353,14619,11035,12415,12217,16421,15290,14790,14655,13180,];

inicioCargaSand=[121872,11550,9588,11444,5729,9019,7459,7310,6713,6837,5822,5234,3808,4801,10800];
finCargaSand=[256228,18393,21697,26456,19607,22046,21506,23103,21360,22576,13787,14142,11177,12417,19251];

inicioCargaVenI=[12585,7160,5809,7362,7904,7394,6886,6759,6280,7526,7800,7369,6905,6840,6182];
finCargaVenI=[22158,18468,18074,19423,20356,20353,16606,18519,15745,17491,19092,17787,20162,18256,17478];

inicioCargaMicrOnYSandOffTramo1=[14925,14380,9807,11733,12483,8217,12366,11434,10657,8858];
finCargaMicrOnYSandOffTramo1=[18075,18245,14536,17937,16142,12895,17349,16505,15130,13281];

inicioCargaMicrOnYSandOn=[18683,18548,14942,18188,16447,13097,17604,16808,15589,13635];
finCargaMicrOnYSandOn=[26103,26128,22314,28455,24886,21590,25279,24839,21433,21920];

inicioCargaMicrOnYSandOffTramo2=[26358,26128,22569,28864,25089,21794,25584,25248,25295,22277];
finCargaMicrOnYSandOffTramo2=[34334,33140,23992,35576,32511,28253,32854,31450,32414,25528];

inicioCargaSandOnYBat3OffTramo1=[7754,6033,6137,4960,6465,5395,5478,5142,6127,7047];
finCargaSandOnYBat3OffTramo1=[19491,14120,14573,14621,14908,13583,13660,11191,13346,13859];

inicioCargaSandOnYBat3On=[20491,15052,15234,15586,15618,14344,14320,12868,14160,14672];
finCargaSandOnYBat3On=[27062,20577,21640,20978,21873,20039,20728,18460,19447,20470];

inicioCargaSandOnYBat3OffTramo2=[28638,21544,21790,21384,22381,20497,21234,19375,20007,20977];
finCargaSandOnYBat3OffTramo2=[38953,27902,29620,29114,29551,26039,27945,27865,26313,27993];


inicioCargaVenIOnYBat3OffTramo1=[6857,6307,6431,6218,5535,5270,5332,5608,5487,5756];
finCargaVenIOnYBat3OffTramo1=[17072,17792,16442,17096,15748,14982,15501,15982,16218,16386];

inicioCargaVenIOnYBat3On=[17936,19116,18019,18326,16815,16250,16579,16899,17135,17352];
finCargaVenIOnYBat3On=[23679,24647,23813,23917,22863,21392,23137,23318,23647,23251];

inicioCargaVenIOnYBat3OffTramo2=[23882,25314,24174,24259,23115,22657,23385,23663,25729,23605];
finCargaVenIOnYBat3OffTramo2=[32117,33647,34329,33612,23115,30843,32788,33425,33460,33824];


inicioCargaVenIOnYMicrOffYSandOffTramo1=[5921,5994,5459,4824,5618,8271,5485,6004,6734,5549];
finCargaVenIOnYMicrOffYSandOffTramo1=[16497,16720,15269,16312,16343,16909,15954,17541,15175,16173];

inicioCargaVenIOnYMicrOnYSandOffTramo1=[22456,22470,20865,21602,21685,22298,21497,23135,21073,21510];
finCargaVenIOnYMicrOnYSandOffTramo1=[25450,27199,25134,23993,27632,28145,27292,28216,28089,26541];

inicioCargaVenIOnYMicrOnYSandOn=[25548,27349,25233,24143,27782,28296,27492,28370,29089,26695];
finCargaVenIOnYMicrOnYSandOn=[34295,36451,33571,33239,33189,37285,37647,37210,39070,36251];
```

```
122 inicioCargaVenIOnYMicrOnYSandOffTramo2=[34549,27349,33727,33392,37542,37849,38323,37413,39274,36455];
123 finCargaVenIOnYMicrOnYSandOffTramo2=[42125,44433,42213,42948,46126,46642,45034,45242,46948,44033];
124
125 inicioCargaVenIOnYMicrOffYSandOffTramo2=[42677,44882,42413,43202,46432,46945,45388,45598,47354,44386];
126 finCargaVenIOnYMicrOffYSandOffTramo2=[50509,51903,50552,51028,55121,55789,54439,53834,49591,53592];
127
128
129 %% CARGA FICHEROS Y ACONDICIONAMIENTO
130
131
132 %Carga ficheros
133 for i=1:TRIALS
134    cargaBat1{i}=load(strcat('PICUS_BAT1_TRIAL',num2str(i),'.txt'));
135    cargaBat2{i}=load(strcat('PICUS_BAT2_TRIAL',num2str(i),'.txt'));
136    cargaBat3{i}=load(strcat('PICUS_BAT3_TRIAL',num2str(i),'.txt'));
137    cargaExpr{i}=load(strcat('PICUS_EXPR_TRIAL',num2str(i),'.txt'));
138    cargaMicr{i}=load(strcat('PICUS_MICR_TRIAL',num2str(i),'.txt'));
139    cargaSand{i}=load(strcat('PICUS_SAND_TRIAL',num2str(i),'.txt'));
140    cargaVen1{i}=load(strcat('PICUS_VEN1_TRIAL',num2str(i),'.txt'));
141 %    cargaVen2{i}=load(strcat('PICUS_VEN2_TRIAL',num2str(i),'.txt'));
142    cargaVen1{i}=load(strcat('PICUS_VEN1_TRIAL',num2str(i),'.txt'));
143    cargaMicrOnYSandOn{i}=load(strcat('PICUS_MICR_SAND_SAND_MICRO_TRIAL',num2str(i),'.txt'));
144    cargaSandOnYBat3On{i}=load(strcat('PICUS_SAND_BAT3_BAT3_SAND_TRIAL',num2str(i),'.txt'));
145    cargaVenIOnYBat3On{i}=load(strcat('PICUS_VEN1_BAT3_BAT3_VENTI_TRIAL',num2str(i),'.txt'));
146    cargaVenIOnYMicrOnYSandOn{i}=load(strcat('PICUS_VEN1_MICR_SAND_SAND_MICR_VEN1_TRIAL',num2str(i),'.txt'));
147 end
148
149
150 %recorte de la señal para tener solo la onda de ON.
151 for i=1:TRIALS
152    cargaBat1{1,i}=cargaBat1{1,i}(inicioCargaBat1(i):finCargaBat1(i));
153    cargaBat2{1,i}=cargaBat2{1,i}(inicioCargaBat2(i):finCargaBat2(i));
154    cargaBat3{1,i}=cargaBat3{1,i}(inicioCargaBat3(i):finCargaBat3(i));
155    cargaExpr{1,i}=cargaExpr{1,i}(inicioCargaExpr(i):finCargaExpr(i));
156    cargaMicr{1,i}=cargaMicr{1,i}(inicioCargaMicr(i):finCargaMicr(i));
157    cargaSand{1,i}=cargaSand{1,i}(inicioCargaSand(i):finCargaSand(i));
158    cargaVen1{1,i}=cargaVen1{1,i}(inicioCargaVen1(i):finCargaVen1(i));
159 %    cargaVen2{1,i}=cargaVen1{1,i}(inicioCargaVen2(i):finCargaVen2(i));
160    cargaMicrOnYSandOn{1,i}=cargaMicrOnYSandOn{1,i}(inicioCargaMicrOnYSandOn(i):finCargaMicrOnYSandOn(i));
161    cargaSandOnYBat3On{1,i}=cargaSandOnYBat3On{1,i}(inicioCargaSandOnYBat3On(i):finCargaSandOnYBat3On(i));
162    cargaVenIOnYBat3On{1,i}=cargaVenIOnYBat3On{1,i}(inicioCargaVenIOnYBat3On(i):finCargaVenIOnYBat3On(i));
163    cargaVenIOnYMicrOnYSandOn{1,i}=cargaVenIOnYMicrOnYSandOn{1,i}(inicioCargaVenIOnYMicrOnYSandOn(i):finCargaVenIOnYMicrOnYSandOn(i));
164 end
165
166
167 %Subsampleo
168 %recorremos el vector de uno en uno y si encontramos un cero se guarda
169 %como el cero no es parte de la onda, incremento el indice de subsampleo a
170 %la siguiente muestra. Si coincide muestra leida con el indice subsampleo
171 %guardo esa muestra y e incremento el indiceSub en SUBSAMPLE
172 for i=1:TRIALS
173    cargaBat1Temp{i}=[];
174    cargaBat2Temp{i}=[];
175    cargaBat3Temp{i}=[];
176    cargaExprTemp{i}=[];
177    cargaMicrTemp{i}=[];
178    cargaSandTemp{i}=[];
179    cargaVen1Temp{i}=[];
180   % cargaVen2Temp{1}=[];
181    cargaMicrOnYSandOnTemp{i}=[];
182    cargaSandOnYBat3OnTemp{i}=[];
183    cargaVenIOnYBat3OnTemp{i}=[];
184    cargaVenIOnYMicrOnYSandOnTemp{i}=[];
185
186
```

```
187             indiceSub=1;
188             j=1;
189             while (j<size(cargaBat1{1,i},2))
190                 if(cargaBat1{1,i}(j)==0)
191                     cargaBat1Temp{1,i}(end+1)=0;
192                     indiceSub=indiceSub+1;
193                 end
194                 if (j==indiceSub)
195                     cargaBat1Temp{1,i}(end+1)=cargaBat1{1,i}(j);
196                     indiceSub=indiceSub+SUBSAMPLE;
197                 end
198                 j=j+1;
199             end
200
201             indiceSub=1;
202             j=1;
203             while (j<size(cargaBat2{1,i},2))
204                 if(cargaBat2{1,i}(j)==0)
205                     cargaBat2Temp{1,i}(end+1)=0;
206                     indiceSub=indiceSub+1;
207                 end
208                 if (j==indiceSub)
209                     cargaBat2Temp{1,i}(end+1)=cargaBat2{1,i}(j);
210                     indiceSub=indiceSub+SUBSAMPLE;
211                 end
212                 j=j+1;
213             end
214
215             indiceSub=1;
216             j=1;
217             while (j<size(cargaBat3{1,i},2))
218                 if(cargaBat3{1,i}(j)==0)
219                     cargaBat3Temp{1,i}(end+1)=0;
220                     indiceSub=indiceSub+1;
221                 end
222                 if (j==indiceSub)
223                     cargaBat3Temp{1,i}(end+1)=cargaBat3{1,i}(j);
224                     indiceSub=indiceSub+SUBSAMPLE;
225                 end
226                 j=j+1;
227             end
228
229             indiceSub=1;
230             j=1;
231             while (j<size(cargaExpr{1,i},2))
232                 if(cargaExpr{1,i}(j)==0)
233                     cargaExprTemp{1,i}(end+1)=0;
234                     indiceSub=indiceSub+1;
235                 end
236                 if (j==indiceSub)
237                     cargaExprTemp{1,i}(end+1)=cargaExpr{1,i}(j);
238                     indiceSub=indiceSub+SUBSAMPLE;
239                 end
240                 j=j+1;
241             end
242
243             indiceSub=1;
244             j=1;
245             while (j<size(cargaMicr{1,i},2))
246                 if(cargaMicr{1,i}(j)==0)
247                     cargaMicrTemp{1,i}(end+1)=0;
248                     indiceSub=indiceSub+1;
249                 end
250                 if (j==indiceSub)
251                     cargaMicrTemp{1,i}(end+1)=cargaMicr{1,i}(j);
```

```
            indiceSub=indiceSub+SUBSAMPLE;
        end
        j=j+1;
    end

    indiceSub=1;
    j=1;
    while (j<size(cargaSand{1,i},2))
        if(cargaSand{1,i}(j)==0)
            cargaSandTemp{1,i}(end+1)=0;
            indiceSub=indiceSub+1;
        end
        if (j==indiceSub)
            cargaSandTemp{1,i}(end+1)=cargaSand{1,i}(j);
            indiceSub=indiceSub+SUBSAMPLE;
        end
        j=j+1;
    end

    indiceSub=1;
    j=1;
    while (j<size(cargaVenl{1,i},2))
        if(cargaVenl{1,i}(j)==0)
            cargaVenlTemp{1,i}(end+1)=0;
            indiceSub=indiceSub+1;
        end
        if (j==indiceSub)
            cargaVenlTemp{1,i}(end+1)=cargaVenl{1,i}(j);
            indiceSub=indiceSub+SUBSAMPLE;
        end
        j=j+1;
    end

    indiceSub=1;
    j=1;
    while (j<size(cargaMicrOnYSandOn{1,i},2))
        if(cargaMicrOnYSandOn{1,i}(j)==0)
            cargaMicrOnYSandOnTemp{1,i}(end+1)=0;
            indiceSub=indiceSub+1;
        end
        if (j==indiceSub)
            cargaMicrOnYSandOnTemp{1,i}(end+1)=cargaMicrOnYSandOn{1,i}(j);
            indiceSub=indiceSub+SUBSAMPLE;
        end
        j=j+1;
    end

    indiceSub=1;
    j=1;
    while (j<size(cargaSandOnYBat3On{1,i},2))
        if(cargaSandOnYBat3On{1,i}(j)==0)
            cargaSandOnYBat3OnTemp{1,i}(end+1)=0;
            indiceSub=indiceSub+1;
        end
        if (j==indiceSub)
            cargaSandOnYBat3OnTemp{1,i}(end+1)=cargaSandOnYBat3On{1,i}(j);
            indiceSub=indiceSub+SUBSAMPLE;
        end
        j=j+1;
    end

    indiceSub=1;
    j=1;
    while (j<size(cargaVenlOnYBat3On{1,i},2))
        if(cargaVenlOnYBat3On{1,i}(j)==0)
```

```
                cargaVenIOnYBat3OnTemp{1,i}(end+1)=0;
                indiceSub=indiceSub+1;
            end
            if (j==indiceSub)
                cargaVenIOnYBat3OnTemp{1,i}(end+1)=cargaVenIOnYBat3On{1,i}(j);
                indiceSub=indiceSub+SUBSAMPLE;
            end
            j=j+1;
        end

        indiceSub=1;
        j=1;
        while (j<size(cargaVenIOnYMicrOnYSandOn{1,i},2))
            if(cargaVenIOnYMicrOnYSandOn{1,i}(j)==0)
                cargaVenIOnYMicrOnYSandOnTemp{1,i}(end+1)=0;
                indiceSub=indiceSub+1;
            end
            if (j==indiceSub)
                cargaVenIOnYMicrOnYSandOnTemp{1,i}(end+1)=cargaVenIOnYMicrOnYSandOn{1,i}(j);
                indiceSub=indiceSub+SUBSAMPLE;
            end
            j=j+1;
        end

        cargaBat1{1,i}=cargaBat1Temp{1,i};
        cargaBat2{1,i}=cargaBat2Temp{1,i};
        cargaBat3{1,i}=cargaBat3Temp{1,i};
        cargaExpr{1,i}=cargaExprTemp{1,i};
        cargaMicr{1,i}=cargaMicrTemp{1,i};
        cargaSand{1,i}=cargaSandTemp{1,i};
        cargaVen1{1,i}=cargaVen1Temp{1,i};
    %   cargaVen2{1,i}=cargaVen1Temp{1,i};
        cargaMicrOnYSandOn{1,i}=cargaMicrOnYSandOnTemp{1,i};
        cargaSandOnYBat3On{1,i}=cargaSandOnYBat3OnTemp{1,i};
        cargaVenIOnYBat3On{1,i}=cargaVenIOnYBat3OnTemp{1,i};
        cargaVenIOnYMicrOnYSandOn{1,i}=cargaVenIOnYMicrOnYSandOnTemp{1,i};

    end

        clear cargaBat1Temp{1,i};
        clear cargaBat2Temp{1,i};
        clear cargaBat3Temp{1,i};
        clear cargaExprTemp{1,i};
        clear cargaMicrTemp{1,i};
        clear cargaSandTemp{1,i};
        clear cargaVen1Temp{1,i};
    %   clear  cargaVen2Temp{1,i};
        clear cargaMicrOnYSandOnTemp{1,i};
        clear cargaSandOnYBat3OnTemp{1,i};
        clear cargaVenIOnYBat3OnTemp{1,i};
        clear cargaVenIOnYMicrOnYSandOnTemp{1,i};
        clear indiceSub;




    %extraccion de los pasos por cero.
    %el paso por cero es en la muestra actual menos el número de
    %pasos por cero anteriores.
    for i=1:TRIALS
        pasosCeroVBat1{i}=find(cargaBat1{1,i}==0)+DESFASE_TRAFO_TENSION;
        pasosCeroVBat1{1,i}=pasosCeroVBat1{1,i}-[0:size(pasosCeroVBat1{1,i},2)-1];

        pasosCeroVBat2{i}=find(cargaBat2{1,i}==0)+DESFASE_TRAFO_TENSION;
        pasosCeroVBat2{1,i}=pasosCeroVBat2{1,i}-[0:size(pasosCeroVBat2{1,i},2)-1];
```

```
382
383                 pasosCeroVBat3{i}=find(cargaBat3{1,i}==0)+DESFASE_TRAFO_TENSION;
384                 pasosCeroVBat3{1,i}=pasosCeroVBat3{1,i}-[0:size(pasosCeroVBat3{1,i},2)-1];
385
386                 pasosCeroVExpr{i}=find(cargaExpr{1,i}==0)+DESFASE_TRAFO_TENSION;
387                 pasosCeroVExpr{1,i}=pasosCeroVExpr{1,i}-[0:size(pasosCeroVExpr{1,i},2)-1];
388
389                 pasosCeroVMicr{i}=find(cargaMicr{1,i}==0)+DESFASE_TRAFO_TENSION;
390                 pasosCeroVMicr{1,i}=pasosCeroVMicr{1,i}-[0:size(pasosCeroVMicr{1,i},2)-1];
391
392                 pasosCeroVSand{i}=find(cargaSand{1,i}==0)+DESFASE_TRAFO_TENSION;
393                 pasosCeroVSand{1,i}=pasosCeroVSand{1,i}-[0:size(pasosCeroVSand{1,i},2)-1];
394
395                 pasosCeroVVenI{i}=find(cargaVenI{1,i}==0)+DESFASE_TRAFO_TENSION;
396                 pasosCeroVVenI{1,i}=pasosCeroVVenI{1,i}-[0:size(pasosCeroVVenI{1,i},2)-1];
397
398             %   pasosCeroVVen2{i}=find(cargaVen2{1,i}==0)+DESFASE_TRAFO_TENSION;
399             %   pasosCeroVVen2{1,i}=pasosCeroVVen2{1,i}-[0:size(pasosCeroVVen2{1,i},2)-1];
400
401                 pasosCeroVMicrOnYSandOn{i}=find(cargaMicrOnYSandOn{1,i}==0)+DESFASE_TRAFO_TENSION;
402                 pasosCeroVMicrOnYSandOn{1,i}=pasosCeroVMicrOnYSandOn{1,i}-[0:size(pasosCeroVMicrOnYSandOn{1,i},2)-1];
403
404                 pasosCeroVSandOnYBat3On{i}=find(cargaSandOnYBat3On{1,i}==0)+DESFASE_TRAFO_TENSION;
405                 pasosCeroVSandOnYBat3On{1,i}=pasosCeroVSandOnYBat3On{1,i}-[0:size(pasosCeroVSandOnYBat3On{1,i},2)-1];
406
407                 pasosCeroVVenIOnYBat3On{i}=find(cargaVenIOnYBat3On{1,i}==0)+DESFASE_TRAFO_TENSION;
408                 pasosCeroVVenIOnYBat3On{1,i}=pasosCeroVVenIOnYBat3On{1,i}-[0:size(pasosCeroVVenIOnYBat3On{1,i},2)-1];
409
410                 pasosCeroVVenIOnYMicrOnYSandOn{i}=find(cargaVenIOnYMicrOnYSandOn{1,i}==0)+DESFASE_TRAFO_TENSION;
411                 pasosCeroVVenIOnYMicrOnYSandOn{1,i}=pasosCeroVVenIOnYMicrOnYSandOn{1,i}-[0:size(pasosCeroVVenIOnYMicrOnYSandOn{1,i},2)-1];
412
413             end
414
415
416
417             %eliminacion de los pasos por cero y de la comp. continua y filtrado
418
419             f2=fdesign.bandpass('n,fc1,fc2',N,FC1,FC2,FS);
420             filtro=design(f2);
421             %fvtool(filtro)
422
423             for i=1:TRIALS
424                 cargaBatI{1,i}=nonzeros(cargaBatI{1,i});
425                 cargaBatI{1,i}=(cargaBatI{1,i})'-median(cargaBatI{1,i});
426                 cargaBatIFiltr{1,i}=filter(filtro,cargaBatI{1,i});
427                 cargaBatIFiltr{1,i}=cargaBatIFiltr{1,i}(INICIO_FILTR:end); %quitamos las INICIO_FILTR muestas
428                 cargaBatI{1,i}= cargaBatI{1,i}(INICIO_FILTR:end); %de la filtrada y del la original
429
430                 cargaBat2{1,i}=nonzeros(cargaBat2{1,i});
431                 cargaBat2{1,i}=(cargaBat2{1,i})'-median(cargaBat2{1,i});
432                 cargaBat2Filtr{1,i}=filter(filtro,cargaBat2{1,i});
433                 cargaBat2Filtr{1,i}=cargaBat2Filtr{1,i}(INICIO_FILTR:end);
434                 cargaBat2{1,i}= cargaBat2{1,i}(INICIO_FILTR:end);
435
436                 cargaBat3{1,i}=nonzeros(cargaBat3{1,i});
437                 cargaBat3{1,i}=(cargaBat3{1,i})'-median(cargaBat3{1,i});
438                 cargaBat3Filtr{1,i}=filter(filtro,cargaBat3{1,i});
439                 cargaBat3Filtr{1,i}=cargaBat3Filtr{1,i}(INICIO_FILTR:end);
440                 cargaBat3{1,i}= cargaBat3{1,i}(INICIO_FILTR:end);
441
442                 cargaExpr{1,i}=nonzeros(cargaExpr{1,i});
443                 cargaExpr{1,i}=(cargaExpr{1,i})'-median(cargaExpr{1,i});
444                 cargaExprFiltr{1,i}=filter(filtro,cargaExpr{1,i});
445                 cargaExprFiltr{1,i}=cargaExprFiltr{1,i}(INICIO_FILTR:end);
446                 cargaExpr{1,i}= cargaExpr{1,i}(INICIO_FILTR:end);
```

```
447
448                 cargaMicr{l,i}=nonzeros(cargaMicr{l,i});
449                 cargaMicr{l,i}=(cargaMicr{l,i})'-median(cargaMicr{l,i});
450                 cargaMicrFiltr{l,i}=filter(filtro,cargaMicr{l,i});
451                 cargaMicrFiltr{l,i}=cargaMicrFiltr{l,i}(INICIO_FILTR:end);
452                 cargaMicr{l,i}= cargaMicr{l,i}(INICIO_FILTR:end);
453
454                 cargaSand{l,i}=nonzeros(cargaSand{l,i});
455                 cargaSand{l,i}=(cargaSand{l,i})'-median(cargaSand{l,i});
456                 cargaSandFiltr{l,i}=filter(filtro,cargaSand{l,i});
457                 cargaSandFiltr{l,i}=cargaSandFiltr{l,i}(INICIO_FILTR:end);
458                 cargaSand{l,i}= cargaSand{l,i}(INICIO_FILTR:end);
459
460                 cargaVenI{l,i}=nonzeros(cargaVenI{l,i});
461                 cargaVenI{l,i}=(cargaVenI{l,i})'-median(cargaVenI{l,i});
462                 cargaVenIFiltr{l,i}=filter(filtro,cargaVenI{l,i});
463                 cargaVenIFiltr{l,i}=cargaVenIFiltr{l,i}(INICIO_FILTR:end);
464                 cargaVenI{l,i}= cargaVenI{l,i}(INICIO_FILTR:end);
465
466             %   cargaVen2{l,i}=nonzeros(cargaVen2{l,i});
467             %   cargaVen2{l,i}=(cargaVen2{l,i})'-median(cargaVen2{l,i});
468             %   cargaVen2Filtr{l,i}=filter(filtro,cargaVen2{l,i});
469             %   cargaVen2Filtr{l,i}=cargaVen2Filtr{l,i}(INICIO_FILTR:end);
470             %   cargaBatI{l,i}= cargaBatI{l,i}(INICIO_FILTR:end);
471
472                 cargaMicrOnYSandOn{l,i}=nonzeros(cargaMicrOnYSandOn{l,i});
473                 cargaMicrOnYSandOn{l,i}=(cargaMicrOnYSandOn{l,i})'-median(cargaMicrOnYSandOn{l,i});
474                 cargaMicrOnYSandOnFiltr{l,i}=filter(filtro,cargaMicrOnYSandOn{l,i});
475                 cargaMicrOnYSandOnFiltr{l,i}=cargaMicrOnYSandOnFiltr{l,i}(INICIO_FILTR:end);
476                 cargaMicrOnYSandOn{l,i}= cargaMicrOnYSandOn{l,i}(INICIO_FILTR:end);
477
478                 cargaSandOnYBat3On{l,i}=nonzeros(cargaSandOnYBat3On{l,i});
479                 cargaSandOnYBat3On{l,i}=(cargaSandOnYBat3On{l,i})'-median(cargaSandOnYBat3On{l,i});
480                 cargaSandOnYBat3OnFiltr{l,i}=filter(filtro,cargaSandOnYBat3On{l,i});
481                 cargaSandOnYBat3OnFiltr{l,i}=cargaSandOnYBat3OnFiltr{l,i}(INICIO_FILTR:end);
482                 cargaSandOnYBat3On{l,i}= cargaSandOnYBat3On{l,i}(INICIO_FILTR:end);
483
484                 cargaVenIOnYBat3On{l,i}=nonzeros(cargaVenIOnYBat3On{l,i});
485                 cargaVenIOnYBat3On{l,i}=(cargaVenIOnYBat3On{l,i})'-median(cargaVenIOnYBat3On{l,i});
486                 cargaVenIOnYBat3OnFiltr{l,i}=filter(filtro,cargaVenIOnYBat3On{l,i});
487                 cargaVenIOnYBat3OnFiltr{l,i}=cargaVenIOnYBat3OnFiltr{l,i}(INICIO_FILTR:end);
488                 cargaVenIOnYBat3On{l,i}= cargaVenIOnYBat3On{l,i}(INICIO_FILTR:end);
489
490                 cargaVenIOnYMicrOnYSandOn{l,i}=nonzeros(cargaVenIOnYMicrOnYSandOn{l,i});
491                 cargaVenIOnYMicrOnYSandOn{l,i}=(cargaVenIOnYMicrOnYSandOn{l,i})'-median(cargaVenIOnYMicrOnYSandOn{l,i});
492                 cargaVenIOnYMicrOnYSandOnFiltr{l,i}=filter(filtro,cargaVenIOnYMicrOnYSandOn{l,i});
493                 cargaVenIOnYMicrOnYSandOnFiltr{l,i}=cargaVenIOnYMicrOnYSandOnFiltr{l,i}(INICIO_FILTR:end);
494                 cargaVenIOnYMicrOnYSandOn{l,i}= cargaVenIOnYMicrOnYSandOn{l,i}(INICIO_FILTR:end);
495
496             end
497
498
499
500             %% FACTOR POTENCIA, PASO POR CERO
501
502             %valor desfase V-I
503             %calculamos el desfase entre tension e intensiadad, flanco bajada de
504             %corriente
505             %con el vector pasosCeroVxxx y calculando el paso por cero
506             %de la onda de corriente filtrada(primer paso por cero, obviando los rebotes)
507
508             pasosCeroIBatI=cell(1,TRIALS);
509
510             for i=1:TRIALS
511                 k=1;% indice de fila de los ceros de corriente
```

```
512             pasosCerolBat1{1,i}(k)=-SEP_PASOS_0;
513             for j=1:size(cargaBat1Filtr{1,i},2)-1%j recorre toda la onda de corriente
514                 if (cargaBat1Filtr{1,i}(j)>MAY_MEN_0)&&(cargaBat1Filtr{1,i}(j+1)<=0)%si los pasos por cero
515                     if j-pasosCerolBat1{1,i}(k)>=SEP_PASOS_0; %estan separados, se añade
516                         if abs(cargaBat1Filtr{1,i}(j))<abs(cargaBat1Filtr{1,i}(j+1))%el que está mas cerca de cero
517                             pasosCerolBat1{1,i}(k+1)=j;
518                         else
519                             pasosCerolBat1{1,i}(k+1)=j+1;
520                         end
521                         k=k+1;
522                     end
523                 end
524             end
525             pasosCerolBat1{1,i}=pasosCerolBat1{1,i}(2:end);%eliminamos el primer valor
526         end                                 %que es cero
527
528
529
530         pasosCerolBat2=cell(1,TRIALS);
531
532         for i=1:TRIALS
533             k=1;% indice de fila
534             pasosCerolBat2{1,i}(k)=-SEP_PASOS_0;
535             for j=1:size(cargaBat2Filtr{1,i},2)-1
536                 if (cargaBat2Filtr{1,i}(j)>MAY_MEN_0)&&(cargaBat2Filtr{1,i}(j+1)<=0)
537                     if j-pasosCerolBat2{1,i}(k)>=SEP_PASOS_0; %si los pasos por cero
538                         if abs(cargaBat2Filtr{1,i}(j))<abs(cargaBat2Filtr{1,i}(j+1))
539                             pasosCerolBat2{1,i}(k+1)=j;%estan separados, se añade
540                         else
541                             pasosCerolBat2{1,i}(k+1)=j+1;%el que está mas cerca de cero
542                         end
543                         k=k+1;
544                     end
545                 end
546             end
547             pasosCerolBat2{1,i}=pasosCerolBat2{1,i}(2:end);%eliminamos el primer valor
548         end                                 %que es cero
549
550
551         pasosCerolBat3=cell(1,TRIALS);
552
553         for i=1:TRIALS
554             k=1;% indice de fila
555             pasosCerolBat3{1,i}(k)=-SEP_PASOS_0;
556             for j=1:size(cargaBat3Filtr{1,i},2)-1
557                 if (cargaBat3Filtr{1,i}(j)>MAY_MEN_0)&&(cargaBat3Filtr{1,i}(j+1)<=0)
558                     if j-pasosCerolBat3{1,i}(k)>=SEP_PASOS_0; %si los pasos por cero
559                         if abs(cargaBat3Filtr{1,i}(j))<abs(cargaBat3Filtr{1,i}(j+1))
560                             pasosCerolBat3{1,i}(k+1)=j;%estan separados, se añade
561                         else
562                             pasosCerolBat3{1,i}(k+1)=j+1;%el que está mas cerca de cero
563                         end
564                         k=k+1;
565                     end
566                 end
567             end
568             pasosCerolBat3{1,i}=pasosCerolBat3{1,i}(2:end);%eliminamos el primer valor
569         end                                 %que es cero
570
571
572         pasosCerolExpr=cell(1,TRIALS);
573
574         for i=1:TRIALS
575             k=1;% indice de fila
576             pasosCerolExpr{1,i}(k)=-SEP_PASOS_0;
```

```
577             for j=1:size(cargaExprFiltr{1,i},2)-1
578                 if (cargaExprFiltr{1,i}(j)>MAY_MEN_0)&&(cargaExprFiltr{1,i}(j+1)<=0)
579                     if j-pasosCero1Expr{1,i}(k)>=SEP_PASOS_0; %si los pasos por cero
580                         if abs(cargaExprFiltr{1,i}(j))<abs(cargaExprFiltr{1,i}(j+1))
581                             pasosCero1Expr{1,i}(k+1)=j;%estan separados, se añade
582                         else
583                             pasosCero1Expr{1,i}(k+1)=j+1;%el que está mas cerca de cero
584                         end
585                         k=k+1;
586                     end
587                 end
588             end
589             pasosCero1Expr{1,i}=pasosCero1Expr{1,i}(2:end);%eliminamos el primer valor
590         end                                 %que es cero
591
592
593         pasosCero1Micr=cell(1,TRIALS);
594
595         for i=1:TRIALS
596             k=1;% indice de fila
597             pasosCero1Micr{1,i}(k)=-SEP_PASOS_0;
598             for j=1:size(cargaMicrFiltr{1,i},2)-1
599                 if (cargaMicrFiltr{1,i}(j)>MAY_MEN_0)&&(cargaMicrFiltr{1,i}(j+1)<=0)
600                     if j-pasosCero1Micr{1,i}(k)>=SEP_PASOS_0; %si los pasos por cero
601                         if abs(cargaMicrFiltr{1,i}(j))<abs(cargaMicrFiltr{1,i}(j+1))
602                             pasosCero1Micr{1,i}(k+1)=j;%estan separados, se añade
603                         else
604                             pasosCero1Micr{1,i}(k+1)=j+1;%el que está mas cerca de cero
605                         end
606                         k=k+1;
607                     end
608                 end
609             end
610             pasosCero1Micr{1,i}=pasosCero1Micr{1,i}(2:end);%eliminamos el primer valor
611         end                                 %que es cero
612
613
614         pasosCero1Sand=cell(1,TRIALS);
615
616         for i=1:TRIALS
617             k=1;% indice de fila
618             pasosCero1Sand{1,i}(k)=-SEP_PASOS_0;
619             for j=1:size(cargaSandFiltr{1,i},2)-1
620                 if (cargaSandFiltr{1,i}(j)>MAY_MEN_0)&&(cargaSandFiltr{1,i}(j+1)<=0)
621                     if j-pasosCero1Sand{1,i}(k)>=SEP_PASOS_0; %si los pasos por cero
622                         if abs(cargaSandFiltr{1,i}(j))<abs(cargaSandFiltr{1,i}(j+1))
623                             pasosCero1Sand{1,i}(k+1)=j;%estan separados, se añade
624                         else
625                             pasosCero1Sand{1,i}(k+1)=j+1;%el que está mas cerca de cero
626                         end
627                         k=k+1;
628                     end
629                 end
630             end
631             pasosCero1Sand{1,i}=pasosCero1Sand{1,i}(2:end);%eliminamos el primer valor
632         end                                 %que es cero
633
634
635         pasosCero1Ven1=cell(1,TRIALS);
636
637         for i=1:TRIALS
638             k=1;% indice de fila
639             pasosCero1Ven1{1,i}(k)=-SEP_PASOS_0;
640             for j=1:size(cargaVen1Filtr{1,i},2)-1
641                 if (cargaVen1Filtr{1,i}(j)>MAY_MEN_0)&&(cargaVen1Filtr{1,i}(j+1)<=0)
```

```
642                 if j-pasosCero1Ven1{1,i}(k)>=SEP_PASOS_0; %si los pasos por cero
643                     if abs(cargaVen1Filtr{1,i}(j))<abs(cargaVen1Filtr{1,i}(j+1))
644                         pasosCero1Ven1{1,i}(k+1)=j;%estan separados, se añade
645                     else
646                         pasosCero1Ven1{1,i}(k+1)=j+1;%el que está mas cerca de cero
647                     end
648                     k=k+1;
649                 end
650             end
651         end
652         pasosCero1Ven1{1,i}=pasosCero1Ven1{1,i}(2:end);%eliminamos el primer valor
653     end                                     %que es cero
654 
655     %
656     % pasosCero1Ven2=cell(1,TRIALS);
657     %
658     % for i=1:TRIALS
659     %     k=1;% indice de fila
660     %     pasosCero1Ven2{1,i}(k)=-SEP_PASOS_0;
661     %     for j=1:size(cargaVen2Filtr{1,i},2)-1
662     %         if (cargaVen2Filtr{1,i}(j)>MAY_MEN_0)&&(cargaVen2Filtr{1,i}(j+1)<=0)
663     %             if j-pasosCero1Ven2{1,i}(k)>=SEP_PASOS_0; %si los pasos por cero
664     %                 if abs(cargaVen2Filtr{1,i}(j))<abs(cargaVen2Filtr{1,i}(j+1))
665     %                     pasosCero1Ven2{1,i}(k+1)=j;%estan separados, se añade
666     %                 else
667     %                     pasosCero1Ven2{1,i}(k+1)=j+1;%el que está mas cerca de cero
668     %                 end
669     %                 k=k+1;
670     %             end
671     %         end
672     %     end
673     %     pasosCero1Ven2{1,i}=pasosCero1Ven2{1,i}(2:end);%eliminamos el primer valor
674     % end                                     %que es cero
675     %
676 
677     pasosCero1MicrOnYSandOn=cell(1,TRIALS);
678 
679     for i=1:TRIALS
680         k=1;% indice de fila
681         pasosCero1MicrOnYSandOn{1,i}(k)=-SEP_PASOS_0;
682         for j=1:size(cargaMicrOnYSandOnFiltr{1,i},2)-1
683             if (cargaMicrOnYSandOnFiltr{1,i}(j)>MAY_MEN_0)&&(cargaMicrOnYSandOnFiltr{1,i}(j+1)<=0)
684                 if j-pasosCero1MicrOnYSandOn{1,i}(k)>=SEP_PASOS_0; %si los pasos por cero
685                     if abs(cargaMicrOnYSandOnFiltr{1,i}(j))<abs(cargaMicrOnYSandOnFiltr{1,i}(j+1))
686                         pasosCero1MicrOnYSandOn{1,i}(k+1)=j;%estan separados, se añade
687                     else
688                         pasosCero1MicrOnYSandOn{1,i}(k+1)=j+1;%el que está mas cerca de cero
689                     end
690                     k=k+1;
691                 end
692             end
693         end
694         pasosCero1MicrOnYSandOn{1,i}=pasosCero1MicrOnYSandOn{1,i}(2:end);%eliminamos el primer valor
695     end                                     %que es cero
696 
697     pasosCero1SandOnYBat3On=cell(1,TRIALS);
698 
699     for i=1:TRIALS
700         k=1;% indice de fila
701         pasosCero1SandOnYBat3On{1,i}(k)=-SEP_PASOS_0;
702         for j=1:size(cargaSandOnYBat3OnFiltr{1,i},2)-1
703             if (cargaSandOnYBat3OnFiltr{1,i}(j)>MAY_MEN_0)&&(cargaSandOnYBat3OnFiltr{1,i}(j+1)<=0)
704                 if j-pasosCero1SandOnYBat3On{1,i}(k)>=SEP_PASOS_0; %si los pasos por cero
705                     if abs(cargaSandOnYBat3OnFiltr{1,i}(j))<abs(cargaSandOnYBat3OnFiltr{1,i}(j+1))
706                         pasosCero1SandOnYBat3On{1,i}(k+1)=j;%estan separados, se añade
```

```
                    else
                        pasosCeroISandOnYBat3On{1,i}(k+1)=j+1;%el que está mas cerca de cero
                    end
                    k=k+1;
                end
            end
        end
        pasosCeroISandOnYBat3On{1,i}=pasosCeroISandOnYBat3On{1,i}(2:end);%eliminamos el primer valor
    end                                 %que es cero

    pasosCeroIVenIOnYBat3On=cell(1,TRIALS);

    for i=1:TRIALS
        k=1;% indice de fila
        pasosCeroIVenIOnYBat3On{1,i}(k)=-SEP_PASOS_0;
        for j=1:size(cargaVenIOnYBat3OnFiltr{1,i},2)-1
            if (cargaVenIOnYBat3OnFiltr{1,i}(j)>MAY_MEN_0)&&(cargaVenIOnYBat3OnFiltr{1,i}(j+1)<=0)
                if j-pasosCeroIVenIOnYBat3On{1,i}(k)>=SEP_PASOS_0; %si los pasos por cero
                    if abs(cargaVenIOnYBat3OnFiltr{1,i}(j))<abs(cargaVenIOnYBat3OnFiltr{1,i}(j+1))
                        pasosCeroIVenIOnYBat3On{1,i}(k+1)=j;%estan separados, se añade
                    else
                        pasosCeroIVenIOnYBat3On{1,i}(k+1)=j+1;%el que está mas cerca de cero
                    end
                    k=k+1;
                end
            end
        end
        pasosCeroIVenIOnYBat3On{1,i}=pasosCeroIVenIOnYBat3On{1,i}(2:end);%eliminamos el primer valor
    end                                 %que es cero

    pasosCeroIVenIOnYMicrOnYSandOn=cell(1,TRIALS);

    for i=1:TRIALS
        k=1;% indice de fila
        pasosCeroIVenIOnYMicrOnYSandOn{1,i}(k)=-SEP_PASOS_0;
        for j=1:size(cargaVenIOnYMicrOnYSandOnFiltr{1,i},2)-1
            if (cargaVenIOnYMicrOnYSandOnFiltr{1,i}(j)>MAY_MEN_0)&&(cargaVenIOnYMicrOnYSandOnFiltr{1,i}(j+1)<=0)
                if j-pasosCeroIVenIOnYMicrOnYSandOn{1,i}(k)>=SEP_PASOS_0; %si los pasos por cero
                    if abs(cargaVenIOnYMicrOnYSandOnFiltr{1,i}(j))<abs(cargaVenIOnYMicrOnYSandOnFiltr{1,i}(j+1))
                        pasosCeroIVenIOnYMicrOnYSandOn{1,i}(k+1)=j;%estan separados, se añade
                    else
                        pasosCeroIVenIOnYMicrOnYSandOn{1,i}(k+1)=j+1;%el que está mas cerca de cero
                    end
                    k=k+1;
                end
            end
        end
        pasosCeroIVenIOnYMicrOnYSandOn{1,i}=pasosCeroIVenIOnYMicrOnYSandOn{1,i}(2:end);%eliminamos el primer valor
    end                                 %que es cero



    %diferencias
    medFacPotBat1=0;
    medFacPotBat2=0;
    medFacPotBat3=0;
    medFacPotExpr=0;
    medFacPotMicr=0;
    medFacPotSand=0;
    medFacPotVenI=0;
    medFacPotMicrOnYSandOn=0;
    medFacPotSandOnYBat3On=0;
    medFacPotVenIOnYBat3On=0;
    medFacPotVenIOnYMicrOnYSandOn=0;
```

```
772
773             for i=1:TRIALS
774                 sizeFacPotBat1(i)=min(size(pasosCero1Bat1{1,i},2),size(pasosCeroVBat1{1,i},2));
775                 FacPotBat1{1,i}=pasosCero1Bat1{1,i}(1:sizeFacPotBat1(i))-pasosCeroVBat1{1,i}(1:sizeFacPotBat1(i));
776                 FacPotBat1{1,i}(find(FacPotBat1{1,i}<0))=FacPotBat1{1,i}(find(FacPotBat1{1,i}<0))+MUESTRAS_CICLO/2;
777                 FacPotBat1{1,i}(find(FacPotBat1{1,i}>(MUESTRAS_CICLO/4)))=FacPotBat1{1,i}(find(FacPotBat1{1,i}>(MUESTRAS_CICLO/4)))-MUESTRAS_CICLO/2;
778                 for j=1:VENTANA_EST:idivide(int32(sizeFacPotBat1(i)),int32(VENTANA_EST))*VENTANA_EST
779                     medFacPotBat1=[medFacPotBat1 median(FacPotBat1{1,i}(j:j+VENTANA_EST-1))];
780                 end
781
782                 sizeFacPotBat2(i)=min(size(pasosCero1Bat2{1,i},2),size(pasosCeroVBat2{1,i},2));
783                 FacPotBat2{1,i}=pasosCero1Bat2{1,i}(1:sizeFacPotBat2(i))-pasosCeroVBat2{1,i}(1:sizeFacPotBat2(i));
784                 FacPotBat2{1,i}(find(FacPotBat2{1,i}<0))=FacPotBat2{1,i}(find(FacPotBat2{1,i}<0))+MUESTRAS_CICLO/2;
785                 FacPotBat2{1,i}(find(FacPotBat2{1,i}>(MUESTRAS_CICLO/4)))=FacPotBat2{1,i}(find(FacPotBat2{1,i}>(MUESTRAS_CICLO/4)))-MUESTRAS_CICLO/2;
786                 for j=1:VENTANA_EST:idivide(int32(sizeFacPotBat2(i)),int32(VENTANA_EST))*VENTANA_EST
787                     medFacPotBat2=[medFacPotBat2 median(FacPotBat2{1,i}(j:j+VENTANA_EST-1))];
788                 end
789
790                 sizeFacPotBat3(i)=min(size(pasosCero1Bat3{1,i},2),size(pasosCeroVBat3{1,i},2));
791                 FacPotBat3{1,i}=pasosCero1Bat3{1,i}(1:sizeFacPotBat3(i))-pasosCeroVBat3{1,i}(1:sizeFacPotBat3(i));
792                 FacPotBat3{1,i}(find(FacPotBat3{1,i}<0))=FacPotBat3{1,i}(find(FacPotBat3{1,i}<0))+MUESTRAS_CICLO/2;
793                 FacPotBat3{1,i}(find(FacPotBat3{1,i}>(MUESTRAS_CICLO/4)))=FacPotBat3{1,i}(find(FacPotBat3{1,i}>(MUESTRAS_CICLO/4)))-MUESTRAS_CICLO/2;
794                 for j=1:VENTANA_EST:idivide(int32(sizeFacPotBat3(i)),int32(VENTANA_EST))*VENTANA_EST
795                     medFacPotBat3=[medFacPotBat3 median(FacPotBat3{1,i}(j:j+VENTANA_EST-1))];
796                 end
797
798                 sizeFacPotExpr(i)=min(size(pasosCero1Expr{1,i},2),size(pasosCeroVExpr{1,i},2));
799                 FacPotExpr{1,i}=pasosCero1Expr{1,i}(1:sizeFacPotExpr(i))-pasosCeroVExpr{1,i}(1:sizeFacPotExpr(i));
800                 FacPotExpr{1,i}(find(FacPotExpr{1,i}<0))=FacPotExpr{1,i}(find(FacPotExpr{1,i}<0))+MUESTRAS_CICLO/2;
801                 FacPotExpr{1,i}(find(FacPotExpr{1,i}>(MUESTRAS_CICLO/4)))=FacPotExpr{1,i}(find(FacPotExpr{1,i}>(MUESTRAS_CICLO/4)))-MUESTRAS_CICLO/2;
802                 for j=1:VENTANA_EST:idivide(int32(sizeFacPotExpr(i)),int32(VENTANA_EST))*VENTANA_EST
803                     medFacPotExpr=[medFacPotExpr median(FacPotExpr{1,i}(j:j+VENTANA_EST-1))];
804                 end
805
806                 sizeFacPotMicr(i)=min(size(pasosCero1Micr{1,i},2),size(pasosCeroVMicr{1,i},2));
807                 FacPotMicr{1,i}=pasosCero1Micr{1,i}(1:sizeFacPotMicr(i))-pasosCeroVMicr{1,i}(1:sizeFacPotMicr(i));
808                 FacPotMicr{1,i}(find(FacPotMicr{1,i}<0))=FacPotMicr{1,i}(find(FacPotMicr{1,i}<0))+MUESTRAS_CICLO/2;
809                 FacPotMicr{1,i}(find(FacPotMicr{1,i}>(MUESTRAS_CICLO/4)))=FacPotMicr{1,i}(find(FacPotMicr{1,i}>(MUESTRAS_CICLO/4)))-MUESTRAS_CICLO/2;
810                 for j=1:VENTANA_EST:idivide(int32(sizeFacPotMicr(i)),int32(VENTANA_EST))*VENTANA_EST
811                     medFacPotMicr=[medFacPotMicr median(FacPotMicr{1,i}(j:j+VENTANA_EST-1))];
812                 end
813
814                 sizeFacPotSand(i)=min(size(pasosCero1Sand{1,i},2),size(pasosCeroVSand{1,i},2));
815                 FacPotSand{1,i}=pasosCero1Sand{1,i}(1:sizeFacPotSand(i))-pasosCeroVSand{1,i}(1:sizeFacPotSand(i));
816                 FacPotSand{1,i}(find(FacPotSand{1,i}<0))=FacPotSand{1,i}(find(FacPotSand{1,i}<0))+MUESTRAS_CICLO/2;
817                 FacPotSand{1,i}(find(FacPotSand{1,i}>(MUESTRAS_CICLO/4)))=FacPotSand{1,i}(find(FacPotSand{1,i}>(MUESTRAS_CICLO/4)))-
818     MUESTRAS_CICLO/2;
819                 for j=1:VENTANA_EST:idivide(int32(sizeFacPotSand(i)),int32(VENTANA_EST))*VENTANA_EST
820                     medFacPotSand=[medFacPotSand median(FacPotSand{1,i}(j:j+VENTANA_EST-1))];
821                 end
822
823                 sizeFacPotVen1(i)=min(size(pasosCero1Ven1{1,i},2),size(pasosCeroVVen1{1,i},2));
824                 FacPotVen1{1,i}=pasosCero1Ven1{1,i}(1:sizeFacPotVen1(i))-pasosCeroVVen1{1,i}(1:sizeFacPotVen1(i));
825                 FacPotVen1{1,i}(find(FacPotVen1{1,i}<0))=FacPotVen1{1,i}(find(FacPotVen1{1,i}<0))+MUESTRAS_CICLO/2;
826                 FacPotVen1{1,i}(find(FacPotVen1{1,i}>(MUESTRAS_CICLO/4)))=FacPotVen1{1,i}(find(FacPotVen1{1,i}>(MUESTRAS_CICLO/4)))-MUESTRAS_CICLO/2;
827                 for j=1:VENTANA_EST:idivide(int32(sizeFacPotVen1(i)),int32(VENTANA_EST))*VENTANA_EST
828                     medFacPotVen1=[medFacPotVen1 median(FacPotVen1{1,i}(j:j+VENTANA_EST-1))];
829                 end
830
831                 sizeFacPotMicrOnYSandOn(i)=min(size(pasosCero1MicrOnYSandOn{1,i},2),size(pasosCeroVMicrOnYSandOn{1,i},2));
832                 FacPotMicrOnYSandOn{1,i}=pasosCero1MicrOnYSandOn{1,i}(1:sizeFacPotMicrOnYSandOn(i))-
833     pasosCeroVMicrOnYSandOn{1,i}(1:sizeFacPotMicrOnYSandOn(i));
834
835     FacPotMicrOnYSandOn{1,i}(find(FacPotMicrOnYSandOn{1,i}<0))=FacPotMicrOnYSandOn{1,i}(find(FacPotMicrOnYSandOn{1,i}<0))+MUESTRAS_CICLO/2;
```

```
836
837     FacPotMicrOnYSandOn{1,i}(find(FacPotMicrOnYSandOn{1,i}>(MUESTRAS_CICLO/4)))=FacPotMicrOnYSandOn{1,i}(find(FacPotMicrOnYSandOn{1,i}>(MUESTRAS
838     _CICLO/4)))-MUESTRAS_CICLO/2;
839                 for j=1:VENTANA_EST:idivide(int32(sizeFacPotMicrOnYSandOn(i)),int32(VENTANA_EST))*VENTANA_EST
840                     medFacPotMicrOnYSandOn=[medFacPotMicrOnYSandOn median(FacPotMicrOnYSandOn{1,i}(j:j+VENTANA_EST-1))];
841                 end
842
843                 sizeFacPotSandOnYBat3On(i)=min(size(pasosCero1SandOnYBat3On{1,i},2),size(pasosCeroVSandOnYBat3On{1,i},2));
844                 FacPotSandOnYBat3On{1,i}=pasosCero1SandOnYBat3On{1,i}(1:sizeFacPotSandOnYBat3On(i))-
845     pasosCeroVSandOnYBat3On{1,i}(1:sizeFacPotSandOnYBat3On(i));
846
847     FacPotSandOnYBat3On{1,i}(find(FacPotSandOnYBat3On{1,i}<0))=FacPotSandOnYBat3On{1,i}(find(FacPotSandOnYBat3On{1,i}<0))+MUESTRAS_CICLO/2;
848
849     FacPotSandOnYBat3On{1,i}(find(FacPotSandOnYBat3On{1,i}>(MUESTRAS_CICLO/4)))=FacPotSandOnYBat3On{1,i}(find(FacPotSandOnYBat3On{1,i}>(MUESTRA
850     S_CICLO/4)))-MUESTRAS_CICLO/2;
851                 for j=1:VENTANA_EST:idivide(int32(sizeFacPotSandOnYBat3On(i)),int32(VENTANA_EST))*VENTANA_EST
852                     medFacPotSandOnYBat3On=[medFacPotSandOnYBat3On median(FacPotSandOnYBat3On{1,i}(j:j+VENTANA_EST-1))];
853                 end
854
855                 sizeFacPotVenIOnYBat3On(i)=min(size(pasosCero1VenIOnYBat3On{1,i},2),size(pasosCeroVVenIOnYBat3On{1,i},2));
856                 FacPotVenIOnYBat3On{1,i}=pasosCero1VenIOnYBat3On{1,i}(1:sizeFacPotVenIOnYBat3On(i))-
857     pasosCeroVVenIOnYBat3On{1,i}(1:sizeFacPotVenIOnYBat3On(i));
858
859     FacPotVenIOnYBat3On{1,i}(find(FacPotVenIOnYBat3On{1,i}<0))=FacPotVenIOnYBat3On{1,i}(find(FacPotVenIOnYBat3On{1,i}<0))+MUESTRAS_CICLO/2;
860
861     FacPotVenIOnYBat3On{1,i}(find(FacPotVenIOnYBat3On{1,i}>(MUESTRAS_CICLO/4)))=FacPotVenIOnYBat3On{1,i}(find(FacPotVenIOnYBat3On{1,i}>(MUESTRAS_
862     CICLO/4)))-MUESTRAS_CICLO/2;
863                 for j=1:VENTANA_EST:idivide(int32(sizeFacPotVenIOnYBat3On(i)),int32(VENTANA_EST))*VENTANA_EST
864                     medFacPotVenIOnYBat3On=[medFacPotVenIOnYBat3On median(FacPotVenIOnYBat3On{1,i}(j:j+VENTANA_EST-1))];
865                 end
866
867                 sizeFacPotVenIOnYMicrOnYSandOn(i)=min(size(pasosCero1VenIOnYMicrOnYSandOn{1,i},2),size(pasosCeroVVenIOnYMicrOnYSandOn{1,i},2));
868                 FacPotVenIOnYMicrOnYSandOn{1,i}=pasosCero1VenIOnYMicrOnYSandOn{1,i}(1:sizeFacPotVenIOnYMicrOnYSandOn(i))-
869     pasosCeroVVenIOnYMicrOnYSandOn{1,i}(1:sizeFacPotVenIOnYMicrOnYSandOn(i));
870
871     FacPotVenIOnYMicrOnYSandOn{1,i}(find(FacPotVenIOnYMicrOnYSandOn{1,i}<0))=FacPotVenIOnYMicrOnYSandOn{1,i}(find(FacPotVenIOnYMicrOnYSandOn{1,i}
872     <0))+MUESTRAS_CICLO/2;
873
874     FacPotVenIOnYMicrOnYSandOn{1,i}(find(FacPotVenIOnYMicrOnYSandOn{1,i}>(MUESTRAS_CICLO/4)))=FacPotVenIOnYMicrOnYSandOn{1,i}(find(FacPotVenIOn
875     YMicrOnYSandOn{1,i}>(MUESTRAS_CICLO/4)))-MUESTRAS_CICLO/2;
876                 for j=1:VENTANA_EST:idivide(int32(sizeFacPotVenIOnYMicrOnYSandOn(i)),int32(VENTANA_EST))*VENTANA_EST
877                     medFacPotVenIOnYMicrOnYSandOn=[medFacPotVenIOnYMicrOnYSandOn median(FacPotVenIOnYMicrOnYSandOn{1,i}(j:j+VENTANA_EST-1))];
878                 end
879             end
880
881             medFacPotBat1=medFacPotBat1(2:end);
882             medFacPotBat2=medFacPotBat2(2:end);
883             medFacPotBat3=medFacPotBat3(2:end);
884             medFacPotExpr=medFacPotExpr(2:end);
885             medFacPotMicr=medFacPotMicr(2:end);
886             medFacPotSand=medFacPotSand(2:end);
887             medFacPotVenI=medFacPotVenI(2:end);
888             medFacPotMicrOnYSandOn=medFacPotMicrOnYSandOn(2:end);
889             medFacPotSandOnYBat3On=medFacPotSandOnYBat3On(2:end);
890             medFacPotVenIOnYBat3On=medFacPotVenIOnYBat3On(2:end);
891             medFacPotVenIOnYMicrOnYSandOn=medFacPotVenIOnYMicrOnYSandOn(2:end);
892
893             %% RMS-FFT-CCDA-CUARTO
894
895             %valor RMS
896             %recortamos la señal en longitud para otener un multiplo
897             %de un periodo asi el valor rms está promediado y tomamos
898             %ventanas de VENTANA_EST numero de ciclos.
899
900
```

```
901         rmsBat1=[];
902         rmsBat2=[];
903         rmsBat3=[];
904         rmsExpr=[];
905         rmsMicr=[];
906         rmsSand=[];
907         rmsVenl=[];
908         rmsMicrOnYSandOn=[];
909         rmsSandOnYBat3On=[];
910         rmsVenlOnYBat3On=[];
911         rmsVenlOnYMicrOnYSandOn=[];
912
913         CalfBat1=[];
914         CbetBat1=[];
915         CalfBat2=[];
916         CbetBat2=[];
917         CalfBat3=[];
918         CbetBat3=[];
919         CalfExpr=[];
920         CbetExpr=[];
921         CalfMicr=[];
922         CbetMicr=[];
923         CalfSand=[];
924         CbetSand=[];
925         CalfVenl=[];
926         CbetVenl=[];
927         CalfMicrOnYSandOn=[];
928         CbetMicrOnYSandOn=[];
929         CalfSandOnYBat3On=[];
930         CbetSandOnYBat3On=[];
931         CalfVenlOnYBat3On=[];
932         CbetVenlOnYBat3On=[];
933         CalfVenlOnYMicrOnYSandOn=[];
934         CbetVenlOnYMicrOnYSandOn=[];
935
936         fft1Bat1=[];
937         fft2Bat1=[];
938         fft3Bat1=[];
939         fft1Bat2=[];
940         fft2Bat2=[];
941         fft3Bat2=[];
942         fft1Bat3=[];
943         fft2Bat3=[];
944         fft3Bat3=[];
945         fft1Expr=[];
946         fft2Expr=[];
947         fft3Expr=[];
948         fft1Micr=[];
949         fft2Micr=[];
950         fft3Micr=[];
951         fft1Sand=[];
952         fft2Sand=[];
953         fft3Sand=[];
954         fft1Venl=[];
955         fft2Venl=[];
956         fft3Venl=[];
957         fft1MicrOnYSandOn=[];
958         fft2MicrOnYSandOn=[];
959         fft3MicrOnYSandOn=[];
960         fft1SandOnYBat3On=[];
961         fft2SandOnYBat3On=[];
962         fft3SandOnYBat3On=[];
963         fft1VenlOnYBat3On=[];
964         fft2VenlOnYBat3On=[];
965         fft3VenlOnYBat3On=[];
```

```
fftVenIOnYMicrOnYSandOn=[];
fft2VenIOnYMicrOnYSandOn=[];
fft3VenIOnYMicrOnYSandOn=[];

picoCargaBatI=[];
picoCargaBat2=[];
picoCargaBat3=[];
picoCargaExpr=[];
picoCargaMicr=[];
picoCargaSand=[];
picoCargaVenI=[];
picoCargaMicrOnYSandOn=[];
picoCargaSandOnYBat3On=[];
picoCargaVenIOnYBat3On=[];
picoCargaVenIOnYMicrOnYSandOn=[];

medPicoBatI=[];
medPicoBat2=[];
medPicoBat3=[];
medPicoExpr=[];
medPicoMicr=[];
medPicoSand=[];
medPicoVenI=[];
medPicoMicrOnYSandOn=[];
medPicoSandOnYBat3On=[];
medPicoVenIOnYBat3On=[];
medPicoVenIOnYMicrOnYSandOn=[];

cuartoCargaBatI=[];
cuartoCargaBat2=[];
cuartoCargaBat3=[];
cuartoCargaExpr=[];
cuartoCargaMicr=[];
cuartoCargaSand=[];
cuartoCargaVenI=[];
cuartoCargaMicrOnYSandOn=[];
cuartoCargaSandOnYBat3On=[];
cuartoCargaVenIOnYBat3On=[];
cuartoCargaVenIOnYMicrOnYSandOn=[];

medCuartoBatI=[];
medCuartoBat2=[];
medCuartoBat3=[];
medCuartoExpr=[];
medCuartoMicr=[];
medCuartoSand=[];
medCuartoVenI=[];
medCuartoMicrOnYSandOn=[];
medCuartoSandOnYBat3On=[];
medCuartoVenIOnYBat3On=[];
medCuartoVenIOnYMicrOnYSandOn=[];

fftIndicesFrec=[];
for i=1:2:size(FRC_BUSQ_FFT,2)
   Indices=find((FRC_BUSQ_FFT(i)<VECTOR_FRECUENCIA)&(VECTOR_FRECUENCIA<FRC_BUSQ_FFT(i+1)));
   fftIndicesFrec=[fftIndicesFrec [Indices(1);Indices(end)]];
end


for i=1:TRIALS

   sizecargaBatI(i)=size(cargaBatI{1,i},2);
   for j=1:VENTANA_EST:size(pasosCerolBatI{1,i},2)-VENTANA_EST-1;%ventana desde paso por cero, hasta ancho ventana.
      cargaBatIVentana=cargaBatI{1,i}(pasosCerolBatI{1,i}(j):pasosCerolBatI{1,i}(j+VENTANA_EST));
      %rmsBatI=[rmsBatI ((trapz(cargaBatIVentana.^2)/size (cargaBatIVentana,2))^0.5)];
```

```
            rmsBat1=[rmsBat1 ((sum(cargaBat1Ventana.^2)/size (cargaBat1Ventana,2))^0.5)];
            %concordia
            ia=cargaBat1Ventana(1,18:end-17);%desfasamos la corriente para crear stma. trif.
            ib=cargaBat1Ventana(1,1:end-34);
            ic=cargaBat1Ventana(1,35:end);
            ialf=abs(ia*(2/3)^0.5-ib*(1/6)^0.5-ic*(1/6)^0.5);
            ibet=abs(ib*(1/2)^0.5-ic*(1/2)^0.5);
            CalfBat1=[CalfBat1 sum(ialf)/size(ialf,2)];
            CbetBat1=[CbetBat1 sum(ibet)/size(ibet,2)];
            %FFT
            y = fft(cargaBat1Ventana,NFFT)/INC_VENT;
            fft1Bat1=[fft1Bat1 max(2*abs(y(fftIndicesFrec(1,1):fftIndicesFrec(2,1))))];
            fft2Bat1=[fft2Bat1 max(2*abs(y(fftIndicesFrec(1,2):fftIndicesFrec(2,2))))];
            fft3Bat1=[fft3Bat1 max(2*abs(y(fftIndicesFrec(1,3):fftIndicesFrec(2,3))))];
            %PICO Y CUARTO
            for k=1:VENTANA_EST
                pico=max(cargaBat1{1,i}((pasosCero1Bat1{1,i}(j):(pasosCero1Bat1{1,i}(j+k)))));
                cuartoCargaBat1=[cuartoCargaBat1 cargaBat1{1,i}(pasosCero1Bat1{1,i}(j+k-1)+CUARTO_SEMIPERIODO)/pico];
                picoCargaBat1=[picoCargaBat1 pico];
            end
            medPicoBat1=[medPicoBat1 median(picoCargaBat1)];
            medCuartoBat1=[medCuartoBat1 median(cuartoCargaBat1)];
            cuartoCargaBat1=[];
            picoCargaBat1=[];
        end

        sizecargaBat2(i)=size(cargaBat2{1,i},2);
        for j=1:VENTANA_EST:size(pasosCero1Bat2{1,i},2)-VENTANA_EST-1;%ventana desde paso por cero, hasta ancho ventana.
            cargaBat2Ventana=cargaBat2{1,i}(pasosCero1Bat2{1,i}(j):pasosCero1Bat2{1,i}(j+VENTANA_EST));
%           rmsBat2=[rmsBat2 ((trapz(cargaBat2Ventana.^2)/size (cargaBat2Ventana,2))^0.5)];
            rmsBat2=[rmsBat2 ((sum(cargaBat2Ventana.^2)/size (cargaBat2Ventana,2))^0.5)];
            %concordia
            ia=cargaBat2Ventana(1,18:end-17);%desfasamos la corriente para crear stma. trif.
            ib=cargaBat2Ventana(1,1:end-34);
            ic=cargaBat2Ventana(1,35:end);
            ialf=abs(ia*(2/3)^0.5-ib*(1/6)^0.5-ic*(1/6)^0.5);
            ibet=abs(ib*(1/2)^0.5-ic*(1/2)^0.5);
            CalfBat2=[CalfBat2 sum(ialf)/size(ialf,2)];
            CbetBat2=[CbetBat2 sum(ibet)/size(ibet,2)];
            %FFT
            y = fft(cargaBat2Ventana,NFFT)/INC_VENT;
            fft1Bat2=[fft1Bat2 max(2*abs(y(fftIndicesFrec(1,1):fftIndicesFrec(2,1))))];
            fft2Bat2=[fft2Bat2 max(2*abs(y(fftIndicesFrec(1,2):fftIndicesFrec(2,2))))];
            fft3Bat2=[fft3Bat2 max(2*abs(y(fftIndicesFrec(1,3):fftIndicesFrec(2,3))))];
            %PICO Y CUARTO
            for k=1:VENTANA_EST
                pico=max(cargaBat2{1,i}((pasosCero1Bat2{1,i}(j):(pasosCero1Bat2{1,i}(j+k)))));
                cuartoCargaBat2=[cuartoCargaBat2 cargaBat2{1,i}(pasosCero1Bat2{1,i}(j+k-1)+CUARTO_SEMIPERIODO)/pico];
                picoCargaBat2=[picoCargaBat2 pico];
            end
            medPicoBat2=[medPicoBat2 median(picoCargaBat2)];
            medCuartoBat2=[medCuartoBat2 median(cuartoCargaBat2)];
            cuartoCargaBat2=[];
            picoCargaBat2=[];
        end

        sizecargaBat3(i)=size(cargaBat3{1,i},2);
        for j=1:VENTANA_EST:size(pasosCero1Bat3{1,i},2)-VENTANA_EST-1;%ventana desde paso por cero, hasta ancho ventana.
            cargaBat3Ventana=cargaBat3{1,i}(pasosCero1Bat3{1,i}(j):pasosCero1Bat3{1,i}(j+VENTANA_EST));
%           rmsBat3=[rmsBat3 ((trapz(cargaBat3Ventana.^2)/size (cargaBat3Ventana,2))^0.5)];
            rmsBat3=[rmsBat3 ((sum(cargaBat3Ventana.^2)/size (cargaBat3Ventana,2))^0.5)];
            %concordia
            ia=cargaBat3Ventana(1,18:end-17);%desfasamos la corriente para crear stma. trif.
            ib=cargaBat3Ventana(1,1:end-34);
            ic=cargaBat3Ventana(1,35:end);
```

```
            ialf=abs(ia*(2/3)^0.5-ib*(1/6)^0.5-ic*(1/6)^0.5);
            ibet=abs(ib*(1/2)^0.5-ic*(1/2)^0.5);
            CalfBat3=[CalfBat3 sum(ialf)/size(ialf,2)];
            CbetBat3=[CbetBat3 sum(ibet)/size(ibet,2)];
            %FFT
            y = fft(cargaBat3Ventana,NFFT)/INC_VENT;
            fftlBat3=[fftlBat3 max(2*abs(y(fftIndicesFrec(1,1):fftIndicesFrec(2,1))))];
            fft2Bat3=[fft2Bat3 max(2*abs(y(fftIndicesFrec(1,2):fftIndicesFrec(2,2))))];
            fft3Bat3=[fft3Bat3 max(2*abs(y(fftIndicesFrec(1,3):fftIndicesFrec(2,3))))];
            %PICO Y CUARTO
            for k=1:VENTANA_EST
                pico=max(cargaBat3{1,i}((pasosCerolBat3{1,i}(j):(pasosCerolBat3{1,i}(j+k)))));
                cuartoCargaBat3=[cuartoCargaBat3 cargaBat3{1,i}(pasosCerolBat3{1,i}(j+k-1)+CUARTO_SEMIPERIODO)/pico];
                picoCargaBat3=[picoCargaBat3 pico];
            end
            medPicoBat3=[medPicoBat3 median(picoCargaBat3)];
            medCuartoBat3=[medCuartoBat3 median(cuartoCargaBat3)];
            cuartoCargaBat3=[];
            picoCargaBat3=[];
        end

        sizecargaExpr(i)=size(cargaExpr{1,i},2);
        for j=1:VENTANA_EST:size(pasosCerolExpr{1,i},2)-VENTANA_EST-1;%ventana desde paso por cero, hasta ancho ventana.
            cargaExprVentana=cargaExpr{1,i}(pasosCerolExpr{1,i}(j):pasosCerolExpr{1,i}(j+VENTANA_EST));
%           rmsExpr=[rmsExpr ((trapz(cargaExprVentana.^2)/size (cargaExprVentana,2))^0.5)];
            rmsExpr=[rmsExpr ((sum(cargaExprVentana.^2)/size (cargaExprVentana,2))^0.5)];
            %concordia
            ia=cargaExprVentana(1,18:end-17);%desfasamos la corriente para crear stma. trif.
            ib=cargaExprVentana(1,1:end-34);
            ic=cargaExprVentana(1,35:end);
            ialf=abs(ia*(2/3)^0.5-ib*(1/6)^0.5-ic*(1/6)^0.5);
            ibet=abs(ib*(1/2)^0.5-ic*(1/2)^0.5);
            CalfExpr=[CalfExpr sum(ialf)/size(ialf,2)];
            CbetExpr=[CbetExpr sum(ibet)/size(ibet,2)];
            %FFT
            y = fft(cargaExprVentana,NFFT)/INC_VENT;
            fftlExpr=[fftlExpr max(2*abs(y(fftIndicesFrec(1,1):fftIndicesFrec(2,1))))];
            fft2Expr=[fft2Expr max(2*abs(y(fftIndicesFrec(1,2):fftIndicesFrec(2,2))))];
            fft3Expr=[fft3Expr max(2*abs(y(fftIndicesFrec(1,3):fftIndicesFrec(2,3))))];
            %PICO Y CUARTO
            for k=1:VENTANA_EST
                pico=max(cargaExpr{1,i}((pasosCerolExpr{1,i}(j):(pasosCerolExpr{1,i}(j+k)))));
                cuartoCargaExpr=[cuartoCargaExpr cargaExpr{1,i}(pasosCerolExpr{1,i}(j+k-1)+CUARTO_SEMIPERIODO)/pico];
                picoCargaExpr=[picoCargaExpr pico];
            end
            medPicoExpr=[medPicoExpr median(picoCargaExpr)];
            medCuartoExpr=[medCuartoExpr median(cuartoCargaExpr)];
            cuartoCargaExpr=[];
            picoCargaExpr=[];
        end

        sizecargaMicr(i)=size(cargaMicr{1,i},2);
        for j=1:VENTANA_EST:size(pasosCerolMicr{1,i},2)-VENTANA_EST-1;%ventana desde paso por cero, hasta ancho ventana.
            cargaMicrVentana=cargaMicr{1,i}(pasosCerolMicr{1,i}(j):pasosCerolMicr{1,i}(j+VENTANA_EST));
%           rmsMicr=[rmsMicr ((trapz(cargaMicrVentana.^2)/size (cargaMicrVentana,2))^0.5)];
            rmsMicr=[rmsMicr ((sum(cargaMicrVentana.^2)/size (cargaMicrVentana,2))^0.5)];
            %concordia
            ia=cargaMicrVentana(1,18:end-17);%desfasamos la corriente para crear stma. trif.
            ib=cargaMicrVentana(1,1:end-34);
            ic=cargaMicrVentana(1,35:end);
            ialf=abs(ia*(2/3)^0.5-ib*(1/6)^0.5-ic*(1/6)^0.5);
            ibet=abs(ib*(1/2)^0.5-ic*(1/2)^0.5);
            CalfMicr=[CalfMicr sum(ialf)/size(ialf,2)];
            CbetMicr=[CbetMicr sum(ibet)/size(ibet,2)];
            %FFT
```

```
            y = fft(cargaMicrVentana,NFFT)/INC_VENT;
            fftIMicr=[fftIMicr max(2*abs(y(fftIndicesFrec(1,1):fftIndicesFrec(2,1))))];
            fft2Micr=[fft2Micr max(2*abs(y(fftIndicesFrec(1,2):fftIndicesFrec(2,2))))];
            fft3Micr=[fft3Micr max(2*abs(y(fftIndicesFrec(1,3):fftIndicesFrec(2,3))))];
            %PICO Y CUARTO
            for k=1:VENTANA_EST
                pico=max(cargaMicr{1,i}((pasosCeroIMicr{1,i}(j):(pasosCeroIMicr{1,i}(j+k)))));
                cuartoCargaMicr=[cuartoCargaMicr cargaMicr{1,i}(pasosCeroIMicr{1,i}(j+k-1)+CUARTO_SEMIPERIODO)/pico];
                picoCargaMicr=[picoCargaMicr pico];
            end
            medPicoMicr=[medPicoMicr median(picoCargaMicr)];
            medCuartoMicr=[medCuartoMicr median(cuartoCargaMicr)];
            cuartoCargaMicr=[];
            picoCargaMicr=[];
        end

        sizecargaSand(i)=size(cargaSand{1,i},2);
        for j=1:VENTANA_EST:size(pasosCeroISand{1,i},2)-VENTANA_EST-1;%ventana desde paso por cero, hasta ancho ventana.
            cargaSandVentana=cargaSand{1,i}(pasosCeroISand{1,i}(j):pasosCeroISand{1,i}(j+VENTANA_EST));
%           rmsSand=[rmsSand ((trapz(cargaSandVentana.^2)/size (cargaSandVentana,2))^0.5)];
            rmsSand=[rmsSand ((sum(cargaSandVentana.^2)/size (cargaSandVentana,2))^0.5)];
            %concordia
            ia=cargaSandVentana(1,18:end-17);%desfasamos la corriente para crear stma. trif.
            ib=cargaSandVentana(1,1:end-34);
            ic=cargaSandVentana(1,35:end);
            ialf=abs(ia*(2/3)^0.5-ib*(1/6)^0.5-ic*(1/6)^0.5);
            ibet=abs(ib*(1/2)^0.5-ic*(1/2)^0.5);
            CalfSand=[CalfSand sum(ialf)/size(ialf,2)];
            CbetSand=[CbetSand sum(ibet)/size(ibet,2)];
            %FFT
            y = fft(cargaSandVentana,NFFT)/INC_VENT;
            fftISand=[fftISand max(2*abs(y(fftIndicesFrec(1,1):fftIndicesFrec(2,1))))];
            fft2Sand=[fft2Sand max(2*abs(y(fftIndicesFrec(1,2):fftIndicesFrec(2,2))))];
            fft3Sand=[fft3Sand max(2*abs(y(fftIndicesFrec(1,3):fftIndicesFrec(2,3))))];
            %PICO Y CUARTO
            for k=1:VENTANA_EST
                pico=max(cargaSand{1,i}((pasosCeroISand{1,i}(j):(pasosCeroISand{1,i}(j+k)))));
                cuartoCargaSand=[cuartoCargaSand cargaSand{1,i}(pasosCeroISand{1,i}(j+k-1)+CUARTO_SEMIPERIODO)/pico];
                picoCargaSand=[picoCargaSand pico];
            end
            medPicoSand=[medPicoSand median(picoCargaSand)];
            medCuartoSand=[medCuartoSand median(cuartoCargaSand)];
            cuartoCargaSand=[];
            picoCargaSand=[];
        end

        sizecargaVenI(i)=size(cargaVenI{1,i},2);
        for j=1:VENTANA_EST:size(pasosCeroIVenI{1,i},2)-VENTANA_EST-1;%ventana desde paso por cero, hasta ancho ventana.
            cargaVenIVentana=cargaVenI{1,i}(pasosCeroIVenI{1,i}(j):pasosCeroIVenI{1,i}(j+VENTANA_EST));%
%           rmsVenI=[rmsVenI ((trapz(cargaVenIVentana.^2)/size (cargaVenIVentana,2))^0.5)];
            rmsVenI=[rmsVenI ((sum(cargaVenIVentana.^2)/size (cargaVenIVentana,2))^0.5)];
            %concordia
            ia=cargaVenIVentana(1,18:end-17);%desfasamos la corriente para crear stma. trif.
            ib=cargaVenIVentana(1,1:end-34);
            ic=cargaVenIVentana(1,35:end);
            ialf=abs(ia*(2/3)^0.5-ib*(1/6)^0.5-ic*(1/6)^0.5);
            ibet=abs(ib*(1/2)^0.5-ic*(1/2)^0.5);
            CalfVenI=[CalfVenI sum(ialf)/size(ialf,2)];
            CbetVenI=[CbetVenI sum(ibet)/size(ibet,2)];
            %FFT
            y = fft(cargaVenIVentana,NFFT)/INC_VENT;
            fftIVenI=[fftIVenI max(2*abs(y(fftIndicesFrec(1,1):fftIndicesFrec(2,1))))];
            fft2VenI=[fft2VenI max(2*abs(y(fftIndicesFrec(1,2):fftIndicesFrec(2,2))))];
            fft3VenI=[fft3VenI max(2*abs(y(fftIndicesFrec(1,3):fftIndicesFrec(2,3))))];
            %PICO Y CUARTO
```

```
                for k=1:VENTANA_EST
                    pico=max(cargaVen1{1,i}((pasosCero1Ven1{1,i}(j):(pasosCero1Ven1{1,i}(j+k))));
                    cuartoCargaVen1=[cuartoCargaVen1 cargaVen1{1,i}(pasosCero1Ven1{1,i}(j+k-1)+CUARTO_SEMIPERIODO)/pico];
                    picoCargaVen1=[picoCargaVen1 pico];
                end
                medPicoVen1=[medPicoVen1 median(picoCargaVen1)];
                medCuartoVen1=[medCuartoVen1 median(cuartoCargaVen1)];
                cuartoCargaVen1=[];
                picoCargaVen1=[];
            end

            sizecargaMicrOnYSandOn(i)=size(cargaMicrOnYSandOn{1,i},2);
            for j=1:VENTANA_EST:size(pasosCero1MicrOnYSandOn{1,i},2)-VENTANA_EST-1;%ventana desde paso por cero, hasta ancho ventana.
                cargaMicrOnYSandOnVentana=cargaMicrOnYSandOn{1,i}(pasosCero1MicrOnYSandOn{1,i}(j):pasosCero1MicrOnYSandOn{1,i}(j+VENTANA_EST));
        %       rmsMicrOnYSandOn=[rmsMicrOnYSandOn ((trapz(cargaMicrOnYSandOnVentana.^2)/size (cargaMicrOnYSandOnVentana,2))^0.5)];
                rmsMicrOnYSandOn=[rmsMicrOnYSandOn ((sum(cargaMicrOnYSandOnVentana.^2)/size (cargaMicrOnYSandOnVentana,2))^0.5)];
                %concordia
                ia=cargaMicrOnYSandOnVentana(1,18:end-17);%desfasamos la corriente para crear stma. trif.
                ib=cargaMicrOnYSandOnVentana(1,1:end-34);
                ic=cargaMicrOnYSandOnVentana(1,35:end);
                ialf=abs(ia*(2/3)^0.5-ib*(1/6)^0.5-ic*(1/6)^0.5);
                ibet=abs(ib*(1/2)^0.5-ic*(1/2)^0.5);
                CalfMicrOnYSandOn=[CalfMicrOnYSandOn sum(ialf)/size(ialf,2)];
                CbetMicrOnYSandOn=[CbetMicrOnYSandOn sum(ibet)/size(ibet,2)];
                %FFT
                y = fft(cargaMicrOnYSandOnVentana,NFFT)/INC_VENT;
                fftlMicrOnYSandOn=[fftlMicrOnYSandOn max(2*abs(y(fftIndicesFrec(1,1):fftIndicesFrec(2,1))))];
                fft2MicrOnYSandOn=[fft2MicrOnYSandOn max(2*abs(y(fftIndicesFrec(1,2):fftIndicesFrec(2,2))))];
                fft3MicrOnYSandOn=[fft3MicrOnYSandOn max(2*abs(y(fftIndicesFrec(1,3):fftIndicesFrec(2,3))))];
                %PICO Y CUARTO
                for k=1:VENTANA_EST
                    pico=max(cargaMicrOnYSandOn{1,i}((pasosCero1MicrOnYSandOn{1,i}(j):(pasosCero1MicrOnYSandOn{1,i}(j+k))));
                    cuartoCargaMicrOnYSandOn=[cuartoCargaMicrOnYSandOn                cargaMicrOnYSandOn{1,i}(pasosCero1MicrOnYSandOn{1,i}(j+k-
1)+CUARTO_SEMIPERIODO)/pico];
                    picoCargaMicrOnYSandOn=[picoCargaMicrOnYSandOn pico];
                end
                medPicoMicrOnYSandOn=[medPicoMicrOnYSandOn median(picoCargaMicrOnYSandOn)];
                medCuartoMicrOnYSandOn=[medCuartoMicrOnYSandOn median(cuartoCargaMicrOnYSandOn)];
                cuartoCargaMicrOnYSandOn=[];
                picoCargaMicrOnYSandOn=[];
            end

            sizecargaSandOnYBat3On(i)=size(cargaSandOnYBat3On{1,i},2);
            for j=1:VENTANA_EST:size(pasosCero1SandOnYBat3On{1,i},2)-VENTANA_EST-1;%ventana desde paso por cero, hasta ancho ventana.
                cargaSandOnYBat3OnVentana=cargaSandOnYBat3On{1,i}(pasosCero1SandOnYBat3On{1,i}(j):pasosCero1SandOnYBat3On{1,i}(j+VENTANA_EST));
        %       rmsSandOnYBat3On=[rmsSandOnYBat3On ((trapz(cargaSandOnYBat3OnVentana.^2)/size (cargaSandOnYBat3OnVentana,2))^0.5)];
                rmsSandOnYBat3On=[rmsSandOnYBat3On ((sum(cargaSandOnYBat3OnVentana.^2)/size (cargaSandOnYBat3OnVentana,2))^0.5)];
                %concordia
                ia=cargaSandOnYBat3OnVentana(1,18:end-17);%desfasamos la corriente para crear stma. trif.
                ib=cargaSandOnYBat3OnVentana(1,1:end-34);
                ic=cargaSandOnYBat3OnVentana(1,35:end);
                ialf=abs(ia*(2/3)^0.5-ib*(1/6)^0.5-ic*(1/6)^0.5);
                ibet=abs(ib*(1/2)^0.5-ic*(1/2)^0.5);
                CalfSandOnYBat3On=[CalfSandOnYBat3On sum(ialf)/size(ialf,2)];
                CbetSandOnYBat3On=[CbetSandOnYBat3On sum(ibet)/size(ibet,2)];
                %FFT
                y = fft(cargaSandOnYBat3OnVentana,NFFT)/INC_VENT;
                fftlSandOnYBat3On=[fftlSandOnYBat3On max(2*abs(y(fftIndicesFrec(1,1):fftIndicesFrec(2,1))))];
                fft2SandOnYBat3On=[fft2SandOnYBat3On max(2*abs(y(fftIndicesFrec(1,2):fftIndicesFrec(2,2))))];
                fft3SandOnYBat3On=[fft3SandOnYBat3On max(2*abs(y(fftIndicesFrec(1,3):fftIndicesFrec(2,3))))];
                %PICO Y CUARTO
                for k=1:VENTANA_EST
                    pico=max(cargaSandOnYBat3On{1,i}((pasosCero1SandOnYBat3On{1,i}(j):(pasosCero1SandOnYBat3On{1,i}(j+k))));
                    cuartoCargaSandOnYBat3On=[cuartoCargaSandOnYBat3On                cargaSandOnYBat3On{1,i}(pasosCero1SandOnYBat3On{1,i}(j+k-
1)+CUARTO_SEMIPERIODO)/pico];
```

```
                picoCargaSandOnYBat3On=[picoCargaSandOnYBat3On pico];
            end
            medPicoSandOnYBat3On=[medPicoSandOnYBat3On median(picoCargaSandOnYBat3On)];
            medCuartoSandOnYBat3On=[medCuartoSandOnYBat3On median(cuartoCargaSandOnYBat3On)];
            cuartoCargaSandOnYBat3On=[];
            picoCargaSandOnYBat3On=[];
        end

        sizecargaVenIOnYBat3On(i)=size(cargaVenIOnYBat3On{1,i},2);
        for j=1:VENTANA_EST:size(pasosCerolVenIOnYBat3On{1,i},2)-VENTANA_EST-1;%ventana desde paso por cero, hasta ancho ventana.
            cargaVenIOnYBat3OnVentana=cargaVenIOnYBat3On{1,i}(pasosCerolVenIOnYBat3On{1,i}(j):pasosCerolVenIOnYBat3On{1,i}(j+VENTANA_EST));
    %       rmsVenIOnYBat3On=[rmsVenIOnYBat3On ((trapz(cargaVenIOnYBat3OnVentana.^2)/size (cargaVenIOnYBat3OnVentana,2))^0.5)];
            rmsVenIOnYBat3On=[rmsVenIOnYBat3On ((sum(cargaVenIOnYBat3OnVentana.^2)/size (cargaVenIOnYBat3OnVentana,2))^0.5)];
            %concordia
            ia=cargaVenIOnYBat3OnVentana(1,18:end-17);%desfasamos la corriente para crear stma. trif.
            ib=cargaVenIOnYBat3OnVentana(1,1:end-34);
            ic=cargaVenIOnYBat3OnVentana(1,35:end);
            ialf=abs(ia*(2/3)^0.5-ib*(1/6)^0.5-ic*(1/6)^0.5);
            ibet=abs(ib*(1/2)^0.5-ic*(1/2)^0.5);
            CalfVenIOnYBat3On=[CalfVenIOnYBat3On sum(ialf)/size(ialf,2)];
            CbetVenIOnYBat3On=[CbetVenIOnYBat3On sum(ibet)/size(ibet,2)];
            %FFT
            y = fft(cargaVenIOnYBat3OnVentana,NFFT)/INC_VENT;
            fftIVenIOnYBat3On=[fftIVenIOnYBat3On max(2*abs(y(fftIndicesFrec(1,1):fftIndicesFrec(2,1))))];
            fft2VenIOnYBat3On=[fft2VenIOnYBat3On max(2*abs(y(fftIndicesFrec(1,2):fftIndicesFrec(2,2))))];
            fft3VenIOnYBat3On=[fft3VenIOnYBat3On max(2*abs(y(fftIndicesFrec(1,3):fftIndicesFrec(2,3))))];
            %PICO Y CUARTO
            for k=1:VENTANA_EST
                pico=max(cargaVenIOnYBat3On{1,i}((pasosCerolVenIOnYBat3On{1,i}(j):(pasosCerolVenIOnYBat3On{1,i}(j+k)))));
                cuartoCargaVenIOnYBat3On=[cuartoCargaVenIOnYBat3On                cargaVenIOnYBat3On{1,i}(pasosCerolVenIOnYBat3On{1,i}(j+k-
1)+CUARTO_SEMIPERIODO)/pico];
                picoCargaVenIOnYBat3On=[picoCargaVenIOnYBat3On pico];
            end
            medPicoVenIOnYBat3On=[medPicoVenIOnYBat3On median(picoCargaVenIOnYBat3On)];
            medCuartoVenIOnYBat3On=[medCuartoVenIOnYBat3On median(cuartoCargaVenIOnYBat3On)];
            cuartoCargaVenIOnYBat3On=[];
            picoCargaVenIOnYBat3On=[];
        end

        sizecargaVenIOnYMicrOnYSandOn(i)=size(cargaVenIOnYMicrOnYSandOn{1,i},2);
        for j=1:VENTANA_EST:size(pasosCerolVenIOnYMicrOnYSandOn{1,i},2)-VENTANA_EST-1;%ventana desde paso por cero, hasta ancho ventana.

cargaVenIOnYMicrOnYSandOnVentana=cargaVenIOnYMicrOnYSandOn{1,i}(pasosCerolVenIOnYMicrOnYSandOn{1,i}(j):pasosCerolVenIOnYMicrOnYSandOn{1,i}
(j+VENTANA_EST));
    %                                   rmsVenIOnYMicrOnYSandOn=[rmsVenIOnYMicrOnYSandOn     ((trapz(cargaVenIOnYMicrOnYSandOnVentana.^2)/size
(cargaVenIOnYMicrOnYSandOnVentana,2))^0.5)];
            rmsVenIOnYMicrOnYSandOn=[rmsVenIOnYMicrOnYSandOn                                ((sum(cargaVenIOnYMicrOnYSandOnVentana.^2)/size
(cargaVenIOnYMicrOnYSandOnVentana,2))^0.5)];
            %concordia
            ia=cargaVenIOnYMicrOnYSandOnVentana(1,18:end-17);%desfasamos la corriente para crear stma. trif.
            ib=cargaVenIOnYMicrOnYSandOnVentana(1,1:end-34);
            ic=cargaVenIOnYMicrOnYSandOnVentana(1,35:end);
            ialf=abs(ia*(2/3)^0.5-ib*(1/6)^0.5-ic*(1/6)^0.5);
            ibet=abs(ib*(1/2)^0.5-ic*(1/2)^0.5);
            CalfVenIOnYMicrOnYSandOn=[CalfVenIOnYMicrOnYSandOn sum(ialf)/size(ialf,2)];
            CbetVenIOnYMicrOnYSandOn=[CbetVenIOnYMicrOnYSandOn sum(ibet)/size(ibet,2)];
            %FFT
            y = fft(cargaVenIOnYMicrOnYSandOnVentana,NFFT)/INC_VENT;
            fftIVenIOnYMicrOnYSandOn=[fftIVenIOnYMicrOnYSandOn max(2*abs(y(fftIndicesFrec(1,1):fftIndicesFrec(2,1))))];
            fft2VenIOnYMicrOnYSandOn=[fft2VenIOnYMicrOnYSandOn max(2*abs(y(fftIndicesFrec(1,2):fftIndicesFrec(2,2))))];
            fft3VenIOnYMicrOnYSandOn=[fft3VenIOnYMicrOnYSandOn max(2*abs(y(fftIndicesFrec(1,3):fftIndicesFrec(2,3))))];
            %PICO Y CUARTO
            for k=1:VENTANA_EST
                pico=max(cargaVenIOnYMicrOnYSandOn{1,i}((pasosCerolVenIOnYMicrOnYSandOn{1,i}(j):(pasosCerolVenIOnYMicrOnYSandOn{1,i}(j+k)))));
```

```
                cuartoCargaVenIOnYMicrOnYSandOn=[cuartoCargaVenIOnYMicrOnYSandOn
cargaVenIOnYMicrOnYSandOn{1,i}(pasosCeroIVenIOnYMicrOnYSandOn{1,i}(j+k-1)+CUARTO_SEMIPERIODO)/pico];
                picoCargaVenIOnYMicrOnYSandOn=[picoCargaVenIOnYMicrOnYSandOn pico];
            end
            medPicoVenIOnYMicrOnYSandOn=[medPicoVenIOnYMicrOnYSandOn median(picoCargaVenIOnYMicrOnYSandOn)];
            medCuartoVenIOnYMicrOnYSandOn=[medCuartoVenIOnYMicrOnYSandOn median(cuartoCargaVenIOnYMicrOnYSandOn)];
            cuartoCargaVenIOnYMicrOnYSandOn=[];
            picoCargaVenIOnYMicrOnYSandOn=[];
        end
    end



    %% MATRICES ENTRADAS-TARGETS

    %ordenacion de los datos en matrices entradas y targets
    sizeMatEntradasBatI=min([size(rmsBatI,2) size(CalfBatI,2) size(CbetBatI,2) size(fftIBatI,2) size(fft2BatI,2) size(fft3BatI,2) size(medFacPotBatI,2)
size(medPicoBatI,2) size(medCuartoBatI,2)]);
    sizeMatEntradasBat2=min([size(rmsBat2,2) size(CalfBat2,2) size(CbetBat2,2) size(fftIBat2,2) size(fft2Bat2,2) size(fft3Bat2,2)
size(medFacPotBat2,2) size(medPicoBat2,2) size(medCuartoBat2,2)]);
    sizeMatEntradasBat3=min([size(rmsBat3,2) size(CalfBat3,2) size(CbetBat3,2) size(fftIBat3,2) size(fft2Bat3,2) size(fft3Bat3,2)
size(medFacPotBat3,2) size(medPicoBat3,2) size(medCuartoBat3,2)]);
    sizeMatEntradasExpr=min([size(rmsExpr,2) size(CalfExpr,2) size(CbetExpr,2) size(fftIExpr,2) size(fft2Expr,2) size(fft3Expr,2) size(medFacPotExpr,2)
size(medPicoExpr,2) size(medCuartoExpr,2)]);
    sizeMatEntradasMicr=min([size(rmsMicr,2) size(CalfMicr,2) size(CbetMicr,2) size(fftIMicr,2) size(fft2Micr,2) size(fft3Micr,2) size(medFacPotMicr,2)
size(medPicoMicr,2) size(medCuartoMicr,2)]);
    sizeMatEntradasSand=min([size(rmsSand,2) size(CalfSand,2) size(CbetSand,2) size(fftISand,2) size(fft2Sand,2) size(fft3Sand,2)
size(medFacPotSand,2) size(medPicoSand,2) size(medCuartoSand,2)]);
    sizeMatEntradasVenI=min([size(rmsVenI,2) size(CalfVenI,2) size(CbetVenI,2) size(fftIVenI,2) size(fft2VenI,2) size(fft3VenI,2) size(medFacPotVenI,2)
size(medPicoVenI,2) size(medCuartoVenI,2)]);
    sizeMatEntradasMicrOnYSandOn=min([size(rmsMicrOnYSandOn,2) size(CalfMicrOnYSandOn,2) size(CbetMicrOnYSandOn,2)
size(fftIMicrOnYSandOn,2) size(fft2MicrOnYSandOn,2) size(fft3MicrOnYSandOn,2) size(medFacPotMicrOnYSandOn,2) size(medPicoMicrOnYSandOn,2)
size(medCuartoMicrOnYSandOn,2)]);
    sizeMatEntradasSandOnYBat3On=min([size(rmsSandOnYBat3On,2) size(CalfSandOnYBat3On,2) size(CbetSandOnYBat3On,2)
size(fftISandOnYBat3On,2) size(fft2SandOnYBat3On,2) size(fft3SandOnYBat3On,2) size(medFacPotSandOnYBat3On,2) size(medPicoSandOnYBat3On,2)
size(medCuartoSandOnYBat3On,2)]);
    sizeMatEntradasVenIOnYBat3On=min([size(rmsVenIOnYBat3On,2) size(CalfVenIOnYBat3On,2) size(CbetVenIOnYBat3On,2) size(fftIVenIOnYBat3On,2)
size(fft2VenIOnYBat3On,2) size(fft3VenIOnYBat3On,2) size(medFacPotVenIOnYBat3On,2) size(medPicoVenIOnYBat3On,2)
size(medCuartoVenIOnYBat3On,2)]);
    sizeMatEntradasVenIOnYMicrOnYSandOn=min([size(rmsVenIOnYMicrOnYSandOn,2) size(CalfVenIOnYMicrOnYSandOn,2)
size(CbetVenIOnYMicrOnYSandOn,2) size(fftIVenIOnYMicrOnYSandOn,2) size(fft2VenIOnYMicrOnYSandOn,2) size(fft3VenIOnYMicrOnYSandOn,2)
size(medFacPotVenIOnYMicrOnYSandOn,2) size(medPicoVenIOnYMicrOnYSandOn,2) size(medCuartoVenIOnYMicrOnYSandOn,2)]);

    %ecualizacion de entradas==todas con el mismo numero de muestras

    % maxMuestras=max([sizeMatEntradasBatI sizeMatEntradasBat2 sizeMatEntradasBat3 sizeMatEntradasExpr...
    %     sizeMatEntradasMicr sizeMatEntradasSand sizeMatEntradasVenI sizeMatEntradasMicrOnYSandOn...
    %     sizeMatEntradasSandOnYBat3On sizeMatEntradasVenIOnYBat3On sizeMatEntradasVenIOnYMicrOnYSandOn]);
    %
    %
    % unosBatI=ones(1,idivide(int32(maxMuestras),int32(sizeMatEntradasBatI)));
    % unosBat2=ones(1,idivide(int32(maxMuestras),int32(sizeMatEntradasBat2)));
    % unosBat3=ones(1,idivide(int32(maxMuestras),int32(sizeMatEntradasBat3)));
    % unosExpr=ones(1,idivide(int32(maxMuestras),int32(sizeMatEntradasExpr)));
    % unosMicr=ones(1,idivide(int32(maxMuestras),int32(sizeMatEntradasMicr)));
    % unosSand=ones(1,idivide(int32(maxMuestras),int32(sizeMatEntradasSand)));
    % unosVenI=ones(1,idivide(int32(maxMuestras),int32(sizeMatEntradasVenI)));
    % unosMicrOnYSandOn=ones(1,idivide(int32(maxMuestras),int32(sizeMatEntradasMicrOnYSandOn)));
    % unosSandOnYBat3On=ones(1,idivide(int32(maxMuestras),int32(sizeMatEntradasSandOnYBat3On)));
    % unosVenIOnYBat3On=ones(1,idivide(int32(maxMuestras),int32(sizeMatEntradasVenIOnYBat3On)));
    % unosVenIOnYMicrOnYSandOn=ones(1,idivide(int32(maxMuestras),int32(sizeMatEntradasVenIOnYMicrOnYSandOn)));


    % matriz entradas sin ecualizar
```

```
matEntradasOr=[rmsBat1(1:sizeMatEntradasBat1) rmsBat2(1:sizeMatEntradasBat2) rmsBat3(1:sizeMatEntradasBat3)
rmsExpr(1:sizeMatEntradasExpr) rmsMicr(1:sizeMatEntradasMicr) rmsSand(1:sizeMatEntradasSand) rmsVen1(1:sizeMatEntradasVen1)
rmsMicrOnYSandOn(1:sizeMatEntradasMicrOnYSandOn) rmsSandOnYBat3On(1:sizeMatEntradasSandOnYBat3On)
rmsVen1OnYBat3On(1:sizeMatEntradasVen1OnYBat3On) rmsVen1OnYMicrOnYSandOn(1:sizeMatEntradasVen1OnYMicrOnYSandOn);...
CalfBat1(1:sizeMatEntradasBat1) CalfBat2(1:sizeMatEntradasBat2) CalfBat3(1:sizeMatEntradasBat3)
CalfExpr(1:sizeMatEntradasExpr) CalfMicr(1:sizeMatEntradasMicr) CalfSand(1:sizeMatEntradasSand) CalfVen1(1:sizeMatEntradasVen1)
CalfMicrOnYSandOn(1:sizeMatEntradasMicrOnYSandOn) CalfSandOnYBat3On(1:sizeMatEntradasSandOnYBat3On)
CalfVen1OnYBat3On(1:sizeMatEntradasVen1OnYBat3On) CalfVen1OnYMicrOnYSandOn(1:sizeMatEntradasVen1OnYMicrOnYSandOn);...
CbetBat1(1:sizeMatEntradasBat1) CbetBat2(1:sizeMatEntradasBat2) CbetBat3(1:sizeMatEntradasBat3)
CbetExpr(1:sizeMatEntradasExpr) CbetMicr(1:sizeMatEntradasMicr) CbetSand(1:sizeMatEntradasSand) CbetVen1(1:sizeMatEntradasVen1)
CbetMicrOnYSandOn(1:sizeMatEntradasMicrOnYSandOn) CbetSandOnYBat3On(1:sizeMatEntradasSandOnYBat3On)
CbetVen1OnYBat3On(1:sizeMatEntradasVen1OnYBat3On) CbetVen1OnYMicrOnYSandOn(1:sizeMatEntradasVen1OnYMicrOnYSandOn);...
fftlBat1(1:sizeMatEntradasBat1) fftlBat2(1:sizeMatEntradasBat2) fftlBat3(1:sizeMatEntradasBat3) fftlExpr(1:sizeMatEntradasExpr)
fftlMicr(1:sizeMatEntradasMicr) fftlSand(1:sizeMatEntradasSand) fftlVen1(1:sizeMatEntradasVen1)
fftlMicrOnYSandOn(1:sizeMatEntradasMicrOnYSandOn) fftlSandOnYBat3On(1:sizeMatEntradasSandOnYBat3On)
fftlVen1OnYBat3On(1:sizeMatEntradasVen1OnYBat3On) fftlVen1OnYMicrOnYSandOn(1:sizeMatEntradasVen1OnYMicrOnYSandOn);...
fft2Bat1(1:sizeMatEntradasBat1) fft2Bat2(1:sizeMatEntradasBat2) fft2Bat3(1:sizeMatEntradasBat3)
fft2Expr(1:sizeMatEntradasExpr) fft2Micr(1:sizeMatEntradasMicr) fft2Sand(1:sizeMatEntradasSand) fft2Ven1(1:sizeMatEntradasVen1)
fft2MicrOnYSandOn(1:sizeMatEntradasMicrOnYSandOn) fft2SandOnYBat3On(1:sizeMatEntradasSandOnYBat3On)
fft2Ven1OnYBat3On(1:sizeMatEntradasVen1OnYBat3On) fft2Ven1OnYMicrOnYSandOn(1:sizeMatEntradasVen1OnYMicrOnYSandOn);...
fft3Bat1(1:sizeMatEntradasBat1) fft3Bat2(1:sizeMatEntradasBat2) fft3Bat3(1:sizeMatEntradasBat3)
fft3Expr(1:sizeMatEntradasExpr) fft3Micr(1:sizeMatEntradasMicr) fft3Sand(1:sizeMatEntradasSand) fft3Ven1(1:sizeMatEntradasVen1)
fft3MicrOnYSandOn(1:sizeMatEntradasMicrOnYSandOn) fft3SandOnYBat3On(1:sizeMatEntradasSandOnYBat3On)
fft3Ven1OnYBat3On(1:sizeMatEntradasVen1OnYBat3On) fft3Ven1OnYMicrOnYSandOn(1:sizeMatEntradasVen1OnYMicrOnYSandOn);...
medFacPotBat1(1:sizeMatEntradasBat1) medFacPotBat2(1:sizeMatEntradasBat2) medFacPotBat3(1:sizeMatEntradasBat3)
medFacPotExpr(1:sizeMatEntradasExpr) medFacPotMicr(1:sizeMatEntradasMicr) medFacPotSand(1:sizeMatEntradasSand)
medFacPotVen1(1:sizeMatEntradasVen1) medFacPotMicrOnYSandOn(1:sizeMatEntradasMicrOnYSandOn)
medFacPotSandOnYBat3On(1:sizeMatEntradasSandOnYBat3On) medFacPotVen1OnYBat3On(1:sizeMatEntradasVen1OnYBat3On)
medFacPotVen1OnYMicrOnYSandOn(1:sizeMatEntradasVen1OnYMicrOnYSandOn);...
medPicoBat1(1:sizeMatEntradasBat1) medPicoBat2(1:sizeMatEntradasBat2) medPicoBat3(1:sizeMatEntradasBat3)
medPicoExpr(1:sizeMatEntradasExpr) medPicoMicr(1:sizeMatEntradasMicr) medPicoSand(1:sizeMatEntradasSand)
medPicoVen1(1:sizeMatEntradasVen1) medPicoMicrOnYSandOn(1:sizeMatEntradasMicrOnYSandOn)
medPicoSandOnYBat3On(1:sizeMatEntradasSandOnYBat3On) medPicoVen1OnYBat3On(1:sizeMatEntradasVen1OnYBat3On)
medPicoVen1OnYMicrOnYSandOn(1:sizeMatEntradasVen1OnYMicrOnYSandOn);...
medCuartoBat1(1:sizeMatEntradasBat1) medCuartoBat2(1:sizeMatEntradasBat2) medCuartoBat3(1:sizeMatEntradasBat3)
medCuartoExpr(1:sizeMatEntradasExpr) medCuartoMicr(1:sizeMatEntradasMicr) medCuartoSand(1:sizeMatEntradasSand)
medCuartoVen1(1:sizeMatEntradasVen1) medCuartoMicrOnYSandOn(1:sizeMatEntradasMicrOnYSandOn)
medCuartoSandOnYBat3On(1:sizeMatEntradasSandOnYBat3On) medCuartoVen1OnYBat3On(1:sizeMatEntradasVen1OnYBat3On)
medCuartoVen1OnYMicrOnYSandOn(1:sizeMatEntradasVen1OnYMicrOnYSandOn);...
];


%% matriz entradas ecualizada
% matEntradas=[rmsBat1(1:sizeMatEntradasBat1)*unosBat1 rmsBat2(1:sizeMatEntradasBat2)*unosBat2
rmsBat3(1:sizeMatEntradasBat3)*unosBat3 rmsExpr(1:sizeMatEntradasExpr)*unosExpr rmsMicr(1:sizeMatEntradasMicr)*unosMicr
rmsSand(1:sizeMatEntradasSand)*unosSand rmsVen1(1:sizeMatEntradasVen1)*unosVen1
rmsMicrOnYSandOn(1:sizeMatEntradasMicrOnYSandOn)*unosMicrOnYSandOn
rmsSandOnYBat3On(1:sizeMatEntradasSandOnYBat3On)*unosSandOnYBat3On
rmsVen1OnYBat3On(1:sizeMatEntradasVen1OnYBat3On)*unosVen1OnYBat3On
rmsVen1OnYMicrOnYSandOn(1:sizeMatEntradasVen1OnYMicrOnYSandOn)*unosVen1OnYMicrOnYSandOn;...
% CalfBat1(1:sizeMatEntradasBat1)*unosBat1 CalfBat2(1:sizeMatEntradasBat2)*unosBat2 CalfBat3(1:sizeMatEntradasBat3)*unosBat3
CalfExpr(1:sizeMatEntradasExpr)*unosExpr CalfMicr(1:sizeMatEntradasMicr)*unosMicr CalfSand(1:sizeMatEntradasSand)*unosSand
CalfVen1(1:sizeMatEntradasVen1)*unosVen1 CalfMicrOnYSandOn(1:sizeMatEntradasMicrOnYSandOn)*unosMicrOnYSandOn
CalfSandOnYBat3On(1:sizeMatEntradasSandOnYBat3On)*unosSandOnYBat3On
CalfVen1OnYBat3On(1:sizeMatEntradasVen1OnYBat3On)*unosVen1OnYBat3On
CalfVen1OnYMicrOnYSandOn(1:sizeMatEntradasVen1OnYMicrOnYSandOn)*unosVen1OnYMicrOnYSandOn;...
% CbetBat1(1:sizeMatEntradasBat1)*unosBat1 CbetBat2(1:sizeMatEntradasBat2)*unosBat2 CbetBat3(1:sizeMatEntradasBat3)*unosBat3
CbetExpr(1:sizeMatEntradasExpr)*unosExpr CbetMicr(1:sizeMatEntradasMicr)*unosMicr CbetSand(1:sizeMatEntradasSand)*unosSand
CbetVen1(1:sizeMatEntradasVen1)*unosVen1 CbetMicrOnYSandOn(1:sizeMatEntradasMicrOnYSandOn)*unosMicrOnYSandOn
CbetSandOnYBat3On(1:sizeMatEntradasSandOnYBat3On)*unosSandOnYBat3On
CbetVen1OnYBat3On(1:sizeMatEntradasVen1OnYBat3On)*unosVen1OnYBat3On
CbetVen1OnYMicrOnYSandOn(1:sizeMatEntradasVen1OnYMicrOnYSandOn)*unosVen1OnYMicrOnYSandOn;...
% fftlBat1(1:sizeMatEntradasBat1)*unosBat1 fftlBat2(1:sizeMatEntradasBat2)*unosBat2 fftlBat3(1:sizeMatEntradasBat3)*unosBat3
fftlExpr(1:sizeMatEntradasExpr)*unosExpr fftlMicr(1:sizeMatEntradasMicr)*unosMicr fftlSand(1:sizeMatEntradasSand)*unosSand
fftlVen1(1:sizeMatEntradasVen1)*unosVen1 fftlMicrOnYSandOn(1:sizeMatEntradasMicrOnYSandOn)*unosMicrOnYSandOn
```

```
fftlSandOnYBat3On(l:sizeMatEntradasSandOnYBat3On)*unosSandOnYBat3On
fftlVenlOnYBat3On(l:sizeMatEntradasVenlOnYBat3On)*unosVenlOnYBat3On
fftlVenlOnYMicrOnYSandOn(l:sizeMatEntradasVenlOnYMicrOnYSandOn)*unosVenlOnYMicrOnYSandOn;...
        %       fft2Batl(l:sizeMatEntradasBatl)*unosBatl    fft2Bat2(l:sizeMatEntradasBat2)*unosBat2    fft2Bat3(l:sizeMatEntradasBat3)*unosBat3
fft2Expr(l:sizeMatEntradasExpr)*unosExpr    fft2Micr(l:sizeMatEntradasMicr)*unosMicr    fft2Sand(l:sizeMatEntradasSand)*unosSand
fft2Venl(l:sizeMatEntradasVenl)*unosVenl    fft2MicrOnYSandOn(l:sizeMatEntradasMicrOnYSandOn)*unosMicrOnYSandOn
fft2SandOnYBat3On(l:sizeMatEntradasSandOnYBat3On)*unosSandOnYBat3On
fft2VenlOnYBat3On(l:sizeMatEntradasVenlOnYBat3On)*unosVenlOnYBat3On
fft2VenlOnYMicrOnYSandOn(l:sizeMatEntradasVenlOnYMicrOnYSandOn)*unosVenlOnYMicrOnYSandOn;...
        %       fft3Batl(l:sizeMatEntradasBatl)*unosBatl    fft3Bat2(l:sizeMatEntradasBat2)*unosBat2    fft3Bat3(l:sizeMatEntradasBat3)*unosBat3
fft3Expr(l:sizeMatEntradasExpr)*unosExpr    fft3Micr(l:sizeMatEntradasMicr)*unosMicr    fft3Sand(l:sizeMatEntradasSand)*unosSand
fft3Venl(l:sizeMatEntradasVenl)*unosVenl    fft3MicrOnYSandOn(l:sizeMatEntradasMicrOnYSandOn)*unosMicrOnYSandOn
fft3SandOnYBat3On(l:sizeMatEntradasSandOnYBat3On)*unosSandOnYBat3On
fft3VenlOnYBat3On(l:sizeMatEntradasVenlOnYBat3On)*unosVenlOnYBat3On
fft3VenlOnYMicrOnYSandOn(l:sizeMatEntradasVenlOnYMicrOnYSandOn)*unosVenlOnYMicrOnYSandOn;...
        %       medFacPotBatl(l:sizeMatEntradasBatl)*unosBatl    medFacPotBat2(l:sizeMatEntradasBat2)*unosBat2
medFacPotBat3(l:sizeMatEntradasBat3)*unosBat3    medFacPotExpr(l:sizeMatEntradasExpr)*unosExpr    medFacPotMicr(l:sizeMatEntradasMicr)*unosMicr
medFacPotSand(l:sizeMatEntradasSand)*unosSand    medFacPotVenl(l:sizeMatEntradasVenl)*unosVenl
medFacPotMicrOnYSandOn(l:sizeMatEntradasMicrOnYSandOn)*unosMicrOnYSandOn
medFacPotSandOnYBat3On(l:sizeMatEntradasSandOnYBat3On)*unosSandOnYBat3On
medFacPotVenlOnYBat3On(l:sizeMatEntradasVenlOnYBat3On)*unosVenlOnYBat3On
medFacPotVenlOnYMicrOnYSandOn(l:sizeMatEntradasVenlOnYMicrOnYSandOn)*unosVenlOnYMicrOnYSandOn;...
        %       medPicoBatl(l:sizeMatEntradasBatl)*unosBatl    medPicoBat2(l:sizeMatEntradasBat2)*unosBat2
medPicoBat3(l:sizeMatEntradasBat3)*unosBat3    medPicoExpr(l:sizeMatEntradasExpr)*unosExpr    medPicoMicr(l:sizeMatEntradasMicr)*unosMicr
medPicoSand(l:sizeMatEntradasSand)*unosSand    medPicoVenl(l:sizeMatEntradasVenl)*unosVenl
medPicoMicrOnYSandOn(l:sizeMatEntradasMicrOnYSandOn)*unosMicrOnYSandOn
medPicoSandOnYBat3On(l:sizeMatEntradasSandOnYBat3On)*unosSandOnYBat3On
medPicoVenlOnYBat3On(l:sizeMatEntradasVenlOnYBat3On)*unosVenlOnYBat3On
medPicoVenlOnYMicrOnYSandOn(l:sizeMatEntradasVenlOnYMicrOnYSandOn)*unosVenlOnYMicrOnYSandOn;...
        %       medCuartoBatl(l:sizeMatEntradasBatl)*unosBatl    medCuartoBat2(l:sizeMatEntradasBat2)*unosBat2
medCuartoBat3(l:sizeMatEntradasBat3)*unosBat3    medCuartoExpr(l:sizeMatEntradasExpr)*unosExpr    medCuartoMicr(l:sizeMatEntradasMicr)*unosMicr
medCuartoSand(l:sizeMatEntradasSand)*unosSand    medCuartoVenl(l:sizeMatEntradasVenl)*unosVenl
medCuartoMicrOnYSandOn(l:sizeMatEntradasMicrOnYSandOn)*unosMicrOnYSandOn
medCuartoSandOnYBat3On(l:sizeMatEntradasSandOnYBat3On)*unosSandOnYBat3On
medCuartoVenlOnYBat3On(l:sizeMatEntradasVenlOnYBat3On)*unosVenlOnYBat3On
medCuartoVenlOnYMicrOnYSandOn(l:sizeMatEntradasVenlOnYMicrOnYSandOn)*unosVenlOnYMicrOnYSandOn;...
        %       ];

        %Target no ecualizado
        %
        matTargetOr=[ones(1,sizeMatEntradasBatl)    zeros(1,sizeMatEntradasBat2)    zeros(1,sizeMatEntradasBat3)    zeros(1,sizeMatEntradasExpr)
zeros(1,sizeMatEntradasMicr)    zeros(1,sizeMatEntradasSand)    zeros(1,sizeMatEntradasVenl)    zeros(1,sizeMatEntradasMicrOnYSandOn)
zeros(1,sizeMatEntradasSandOnYBat3On) zeros(1,sizeMatEntradasVenlOnYBat3On) zeros(1,sizeMatEntradasVenlOnYMicrOnYSandOn); ...
                zeros(1,sizeMatEntradasBatl)    ones(1,sizeMatEntradasBat2)    zeros(1,sizeMatEntradasBat3)    zeros(1,sizeMatEntradasExpr)
zeros(1,sizeMatEntradasMicr)    zeros(1,sizeMatEntradasSand)    zeros(1,sizeMatEntradasVenl)    zeros(1,sizeMatEntradasMicrOnYSandOn)
zeros(1,sizeMatEntradasSandOnYBat3On) zeros(1,sizeMatEntradasVenlOnYBat3On) zeros(1,sizeMatEntradasVenlOnYMicrOnYSandOn); ...
                zeros(1,sizeMatEntradasBatl)    zeros(1,sizeMatEntradasBat2)    ones(1,sizeMatEntradasBat3)    zeros(1,sizeMatEntradasExpr)
zeros(1,sizeMatEntradasMicr)    zeros(1,sizeMatEntradasSand)    zeros(1,sizeMatEntradasVenl)    zeros(1,sizeMatEntradasMicrOnYSandOn)
zeros(1,sizeMatEntradasSandOnYBat3On) zeros(1,sizeMatEntradasVenlOnYBat3On) zeros(1,sizeMatEntradasVenlOnYMicrOnYSandOn); ...
                zeros(1,sizeMatEntradasBatl)    zeros(1,sizeMatEntradasBat2)    zeros(1,sizeMatEntradasBat3)    ones(1,sizeMatEntradasExpr)
zeros(1,sizeMatEntradasMicr)    zeros(1,sizeMatEntradasSand)    zeros(1,sizeMatEntradasVenl)    zeros(1,sizeMatEntradasMicrOnYSandOn)
zeros(1,sizeMatEntradasSandOnYBat3On) zeros(1,sizeMatEntradasVenlOnYBat3On) zeros(1,sizeMatEntradasVenlOnYMicrOnYSandOn); ...
                zeros(1,sizeMatEntradasBatl)    zeros(1,sizeMatEntradasBat2)    zeros(1,sizeMatEntradasBat3)    zeros(1,sizeMatEntradasExpr)
ones(1,sizeMatEntradasMicr)    zeros(1,sizeMatEntradasSand)    zeros(1,sizeMatEntradasVenl)    zeros(1,sizeMatEntradasMicrOnYSandOn)
zeros(1,sizeMatEntradasSandOnYBat3On) zeros(1,sizeMatEntradasVenlOnYBat3On) zeros(1,sizeMatEntradasVenlOnYMicrOnYSandOn); ...
                zeros(1,sizeMatEntradasBatl)    zeros(1,sizeMatEntradasBat2)    zeros(1,sizeMatEntradasBat3)    zeros(1,sizeMatEntradasExpr)
zeros(1,sizeMatEntradasMicr)    ones(1,sizeMatEntradasSand)    zeros(1,sizeMatEntradasVenl)    zeros(1,sizeMatEntradasMicrOnYSandOn)
zeros(1,sizeMatEntradasSandOnYBat3On) zeros(1,sizeMatEntradasVenlOnYBat3On) zeros(1,sizeMatEntradasVenlOnYMicrOnYSandOn); ...
                zeros(1,sizeMatEntradasBatl)    zeros(1,sizeMatEntradasBat2)    zeros(1,sizeMatEntradasBat3)    zeros(1,sizeMatEntradasExpr)
zeros(1,sizeMatEntradasMicr)    zeros(1,sizeMatEntradasSand)    ones(1,sizeMatEntradasVenl)    zeros(1,sizeMatEntradasMicrOnYSandOn)
zeros(1,sizeMatEntradasSandOnYBat3On) zeros(1,sizeMatEntradasVenlOnYBat3On) zeros(1,sizeMatEntradasVenlOnYMicrOnYSandOn); ...
                zeros(1,sizeMatEntradasBatl)    zeros(1,sizeMatEntradasBat2)    zeros(1,sizeMatEntradasBat3)    zeros(1,sizeMatEntradasExpr)
zeros(1,sizeMatEntradasMicr)    zeros(1,sizeMatEntradasSand)    zeros(1,sizeMatEntradasVenl)    ones(1,sizeMatEntradasMicrOnYSandOn)
zeros(1,sizeMatEntradasSandOnYBat3On) zeros(1,sizeMatEntradasVenlOnYBat3On) zeros(1,sizeMatEntradasVenlOnYMicrOnYSandOn); ...
```

```
                zeros(1,sizeMatEntradasBat1)  zeros(1,sizeMatEntradasBat2)  zeros(1,sizeMatEntradasBat3)  zeros(1,sizeMatEntradasExpr)
zeros(1,sizeMatEntradasMicr)  zeros(1,sizeMatEntradasSand)  zeros(1,sizeMatEntradasVen1)  zeros(1,sizeMatEntradasMicrOnYSandOn)
ones(1,sizeMatEntradasSandOnYBat3On)  zeros(1,sizeMatEntradasVen1OnYBat3On) zeros(1,sizeMatEntradasVen1OnYMicrOnYSandOn); ...
                zeros(1,sizeMatEntradasBat1)  zeros(1,sizeMatEntradasBat2)  zeros(1,sizeMatEntradasBat3)  zeros(1,sizeMatEntradasExpr)
zeros(1,sizeMatEntradasMicr)  zeros(1,sizeMatEntradasSand)  zeros(1,sizeMatEntradasVen1)  zeros(1,sizeMatEntradasMicrOnYSandOn)
zeros(1,sizeMatEntradasSandOnYBat3On) ones(1,sizeMatEntradasVen1OnYBat3On)  zeros(1,sizeMatEntradasVen1OnYMicrOnYSandOn); ...
                zeros(1,sizeMatEntradasBat1)  zeros(1,sizeMatEntradasBat2)  zeros(1,sizeMatEntradasBat3)  zeros(1,sizeMatEntradasExpr)
zeros(1,sizeMatEntradasMicr)  zeros(1,sizeMatEntradasSand)  zeros(1,sizeMatEntradasVen1)  zeros(1,sizeMatEntradasMicrOnYSandOn)
zeros(1,sizeMatEntradasSandOnYBat3On) zeros(1,sizeMatEntradasVen1OnYBat3On) ones(1,sizeMatEntradasVen1OnYMicrOnYSandOn) ; ...
                ];


        matTarget=(matTargetOr'*diag(MASCARA_ELECTR))';%aplica la mascara de electrodomesticos a Target

        matEntradas=matEntradasOr*diag(sum(matTarget,1)); %aplica la mascara a entradas
        vecTemp=(sum(matEntradas,1));          %suma por columnas, se obtiene vector
        matEntradas=matEntradas(:,find(vecTemp));%los valores =0 del vecTemp no se escogen

        vecTemp=(sum(matTarget,1));          %suma por columnas, se obtiene vector
        matTarget=matTarget(:,find(vecTemp));%los valores =0 del vecTemp no se escogen
        vecTemp=(sum(matTarget,2));
        matTarget=matTarget(find(vecTemp),:);

        % %Target  ecualizado
        %     matTarget=[ones(1,sizeMatEntradasBat1)*unosBat1  zeros(1,sizeMatEntradasBat2)*unosBat2  zeros(1,sizeMatEntradasBat3)*unosBat3
zeros(1,sizeMatEntradasExpr)*unosExpr  zeros(1,sizeMatEntradasMicr)*unosMicr  zeros(1,sizeMatEntradasSand)*unosSand
zeros(1,sizeMatEntradasVen1)*unosVen1  zeros(1,sizeMatEntradasMicrOnYSandOn)*unosMicrOnYSandOn
zeros(1,sizeMatEntradasSandOnYBat3On)*unosSandOnYBat3On  zeros(1,sizeMatEntradasVen1OnYBat3On)*unosVen1OnYBat3On
zeros(1,sizeMatEntradasVen1OnYMicrOnYSandOn)*unosVen1OnYMicrOnYSandOn; ...
        %  zeros(1,sizeMatEntradasBat1)*unosBat1  ones(1,sizeMatEntradasBat2)*unosBat2  zeros(1,sizeMatEntradasBat3)*unosBat3
zeros(1,sizeMatEntradasExpr)*unosExpr  zeros(1,sizeMatEntradasMicr)*unosMicr  zeros(1,sizeMatEntradasSand)*unosSand
zeros(1,sizeMatEntradasVen1)*unosVen1  zeros(1,sizeMatEntradasMicrOnYSandOn)*unosMicrOnYSandOn
zeros(1,sizeMatEntradasSandOnYBat3On)*unosSandOnYBat3On  zeros(1,sizeMatEntradasVen1OnYBat3On)*unosVen1OnYBat3On
zeros(1,sizeMatEntradasVen1OnYMicrOnYSandOn)*unosVen1OnYMicrOnYSandOn; ...
        %  zeros(1,sizeMatEntradasBat1)*unosBat1  zeros(1,sizeMatEntradasBat2)*unosBat2  ones(1,sizeMatEntradasBat3)*unosBat3
zeros(1,sizeMatEntradasExpr)*unosExpr  zeros(1,sizeMatEntradasMicr)*unosMicr  zeros(1,sizeMatEntradasSand)*unosSand
zeros(1,sizeMatEntradasVen1)*unosVen1  zeros(1,sizeMatEntradasMicrOnYSandOn)*unosMicrOnYSandOn
zeros(1,sizeMatEntradasSandOnYBat3On)*unosSandOnYBat3On  zeros(1,sizeMatEntradasVen1OnYBat3On)*unosVen1OnYBat3On
zeros(1,sizeMatEntradasVen1OnYMicrOnYSandOn)*unosVen1OnYMicrOnYSandOn; ...
        %  zeros(1,sizeMatEntradasBat1)*unosBat1  zeros(1,sizeMatEntradasBat2)*unosBat2  zeros(1,sizeMatEntradasBat3)*unosBat3
ones(1,sizeMatEntradasExpr)*unosExpr  zeros(1,sizeMatEntradasMicr)*unosMicr  zeros(1,sizeMatEntradasSand)*unosSand
zeros(1,sizeMatEntradasVen1)*unosVen1  zeros(1,sizeMatEntradasMicrOnYSandOn)*unosMicrOnYSandOn
zeros(1,sizeMatEntradasSandOnYBat3On)*unosSandOnYBat3On  zeros(1,sizeMatEntradasVen1OnYBat3On)*unosVen1OnYBat3On
zeros(1,sizeMatEntradasVen1OnYMicrOnYSandOn)*unosVen1OnYMicrOnYSandOn; ...
        %  zeros(1,sizeMatEntradasBat1)*unosBat1  zeros(1,sizeMatEntradasBat2)*unosBat2  zeros(1,sizeMatEntradasBat3)*unosBat3
zeros(1,sizeMatEntradasExpr)*unosExpr  ones(1,sizeMatEntradasMicr)*unosMicr  zeros(1,sizeMatEntradasSand)*unosSand
zeros(1,sizeMatEntradasVen1)*unosVen1  zeros(1,sizeMatEntradasMicrOnYSandOn)*unosMicrOnYSandOn
zeros(1,sizeMatEntradasSandOnYBat3On)*unosSandOnYBat3On  zeros(1,sizeMatEntradasVen1OnYBat3On)*unosVen1OnYBat3On
zeros(1,sizeMatEntradasVen1OnYMicrOnYSandOn)*unosVen1OnYMicrOnYSandOn; ...
        %  zeros(1,sizeMatEntradasBat1)*unosBat1  zeros(1,sizeMatEntradasBat2)*unosBat2  zeros(1,sizeMatEntradasBat3)*unosBat3
zeros(1,sizeMatEntradasExpr)*unosExpr  zeros(1,sizeMatEntradasMicr)*unosMicr  ones(1,sizeMatEntradasSand)*unosSand
zeros(1,sizeMatEntradasVen1)*unosVen1  zeros(1,sizeMatEntradasMicrOnYSandOn)*unosMicrOnYSandOn
zeros(1,sizeMatEntradasSandOnYBat3On)*unosSandOnYBat3On  zeros(1,sizeMatEntradasVen1OnYBat3On)*unosVen1OnYBat3On
zeros(1,sizeMatEntradasVen1OnYMicrOnYSandOn)*unosVen1OnYMicrOnYSandOn; ...
        %  zeros(1,sizeMatEntradasBat1)*unosBat1  zeros(1,sizeMatEntradasBat2)*unosBat2  zeros(1,sizeMatEntradasBat3)*unosBat3
zeros(1,sizeMatEntradasExpr)*unosExpr  zeros(1,sizeMatEntradasMicr)*unosMicr  zeros(1,sizeMatEntradasSand)*unosSand
ones(1,sizeMatEntradasVen1)*unosVen1  zeros(1,sizeMatEntradasMicrOnYSandOn)*unosMicrOnYSandOn
zeros(1,sizeMatEntradasSandOnYBat3On)*unosSandOnYBat3On  zeros(1,sizeMatEntradasVen1OnYBat3On)*unosVen1OnYBat3On
zeros(1,sizeMatEntradasVen1OnYMicrOnYSandOn)*unosVen1OnYMicrOnYSandOn; ...
        %  zeros(1,sizeMatEntradasBat1)*unosBat1  zeros(1,sizeMatEntradasBat2)*unosBat2  zeros(1,sizeMatEntradasBat3)*unosBat3
zeros(1,sizeMatEntradasExpr)*unosExpr  zeros(1,sizeMatEntradasMicr)*unosMicr  zeros(1,sizeMatEntradasSand)*unosSand
zeros(1,sizeMatEntradasVen1)*unosVen1  ones(1,sizeMatEntradasMicrOnYSandOn)*unosMicrOnYSandOn
zeros(1,sizeMatEntradasSandOnYBat3On)*unosSandOnYBat3On  zeros(1,sizeMatEntradasVen1OnYBat3On)*unosVen1OnYBat3On
zeros(1,sizeMatEntradasVen1OnYMicrOnYSandOn)*unosVen1OnYMicrOnYSandOn; ...
```

```
    %        zeros(1,sizeMatEntradasBat1)*unosBat1   zeros(1,sizeMatEntradasBat2)*unosBat2   zeros(1,sizeMatEntradasBat3)*unosBat3
zeros(1,sizeMatEntradasExpr)*unosExpr        zeros(1,sizeMatEntradasMicr)*unosMicr        zeros(1,sizeMatEntradasSand)*unosSand
zeros(1,sizeMatEntradasVen1)*unosVen1        zeros(1,sizeMatEntradasMicrOnYSandOn)*unosMicrOnYSandOn
ones(1,sizeMatEntradasSandOnYBat3On)*unosSandOnYBat3On        zeros(1,sizeMatEntradasVen1OnYBat3On)*unosVen1OnYBat3On
zeros(1,sizeMatEntradasVen1OnYMicrOnYSandOn)*unosVen1OnYMicrOnYSandOn; ...
    %        zeros(1,sizeMatEntradasBat1)*unosBat1   zeros(1,sizeMatEntradasBat2)*unosBat2   zeros(1,sizeMatEntradasBat3)*unosBat3
zeros(1,sizeMatEntradasExpr)*unosExpr        zeros(1,sizeMatEntradasMicr)*unosMicr        zeros(1,sizeMatEntradasSand)*unosSand
zeros(1,sizeMatEntradasVen1)*unosVen1        zeros(1,sizeMatEntradasMicrOnYSandOn)*unosMicrOnYSandOn
zeros(1,sizeMatEntradasSandOnYBat3On)*unosSandOnYBat3On        ones(1,sizeMatEntradasVen1OnYBat3On)*unosVen1OnYBat3On
zeros(1,sizeMatEntradasVen1OnYMicrOnYSandOn)*unosVen1OnYMicrOnYSandOn; ...
    %        zeros(1,sizeMatEntradasBat1)*unosBat1   zeros(1,sizeMatEntradasBat2)*unosBat2   zeros(1,sizeMatEntradasBat3)*unosBat3
zeros(1,sizeMatEntradasExpr)*unosExpr        zeros(1,sizeMatEntradasMicr)*unosMicr        zeros(1,sizeMatEntradasSand)*unosSand
zeros(1,sizeMatEntradasVen1)*unosVen1        zeros(1,sizeMatEntradasMicrOnYSandOn)*unosMicrOnYSandOn
zeros(1,sizeMatEntradasSandOnYBat3On)*unosSandOnYBat3On        zeros(1,sizeMatEntradasVen1OnYBat3On)*unosVen1OnYBat3On
ones(1,sizeMatEntradasVen1OnYMicrOnYSandOn)*unosVen1OnYMicrOnYSandOn ; ...
    %    ];
    %

    numeroMuestras=sum(matTarget');


    %% SOM REDUCCION ENTRADAS


    %variando que entradas se tienen en cuenta, se usa un SOM para
    %detectar aquellas que producen kappas más altos



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
    % Esta máscara decide qué CARACTERISTICAS de entrada se van a utilizar en el problema

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
    %     'RMS',  'CCDA' 'CCDB' 'FFT1'  'FFT2'   'FFT3'  'FP'   'PICO' 'CUARTO'
    %mascara = [ 1     1     1     1     1      1     1     1     1  ];


    tamanoMatEntradas=size(matEntradas,2);

    %Separacion test-entrenamiento
    incrementoColumna=0;
    columnaAleatoria=zeros(1,tamanoMatEntradas); %Mascara para separar individuos test de entrenamieno
    limiteRand=0;
    for i=1:size(matTarget,1)
      tamanoColumnaAleatoria(i)=round(numeroMuestras(i)*PORC_ENTRENAMIENTO);
      incrementoColumna=[incrementoColumna incrementoColumna(i)+numeroMuestras(i)];
      limiteRand=[limiteRand incrementoColumna(i)+numeroMuestras(i)];
      while sum(columnaAleatoria(:,incrementoColumna(i)+1:incrementoColumna(i)+numeroMuestras(i)))<tamanoColumnaAleatoria(i)
        aleatorio=round((limiteRand(i)+1)+(limiteRand(i+1)-(limiteRand(i)+1))*rand);
        if columnaAleatoria(aleatorio);%si el numero aleatorio ya esta en la lista
        else                              %no hacer nada
          columnaAleatoria(aleatorio)=1;% si no está añadir
        end
      end
    end

    matEntradasEntrenaSom=matEntradas(: ,find(columnaAleatoria));%Se extraen las columnas en "1"
    matEntradasTestSom=matEntradas(: ,find(not(columnaAleatoria)));%el resto son de test

    matTargetEntrenaSom=matTarget(: ,find(columnaAleatoria));
    matTargetTestSom=matTarget(: ,find(not(columnaAleatoria)));
```

```
1678            clase = matTargetEntrenaSom';
1679            clase_t=matTargetTestSom';
1680
1681            numclases = size(matTargetEntrenaSom,1); % numero de clases
1682
1683            nombre_clases = char('Bat1', 'Bat2', 'Bat3', 'Expr', 'Micr', 'Sand', 'Venl', 'MicrOnYSandOn', 'SandOnYBat3On', 'VenlOnYBat3On',
1684        'VenlOnYMicrOnYSandOn');
1685            nombre_clases=nombre_clases(find(MASCARA_ELECTR),:);
1686            comp_names = CARACTERISTICAS; % Nombres de las CARACTERISTICAS de entrada
1687
1688
1689            names=CARACTERISTICAS;
1690
1691
1692            %Primer SOM con todas las caracteristicas para obtener funcion
1693            %de eficacia vs subsample
1694            'Primer SOM con todas las caracteristicas y todas las muestras'
1695            nombre_mapa = 'SOM NILM';
1696            data = matEntradasEntrenaSom; % guardamos la data original y separamos entrenamiento de test
1697            clase = matTargetEntrenaSom;
1698            clase=vec2ind(clase)';
1699            data_t=matEntradasTestSom;
1700            clase_t=matTargetTestSom;
1701            clase_t=vec2ind(clase_t)';
1702
1703            lattice = 'hexa';
1704            [ ndata npat ] = size(data); % npat = numero de patrones (pacientes), ndata = numero de variables
1705
1706            %           'RMS',  'CCDA' 'CCDB' 'FFT1'  'FFT2'   'FFT3'  'FP'   'PICO' 'CUARTO'
1707            %mascara_ini = [ 1     1     1     0     0     0     1     1     1  ];
1708
1709            %mascara_ini = ones( 1, ndata);     % Mascara incluyendo todas las variables ( 1 = incluir / 0 = suprimir)
1710            %%%%%%%%%%%%% GENERA LA STRUCT DE DATOS ENTRENAMIENTO DEL SOM  %%%%%%%%%%%%%%%%%%%%%%%%%%%
1711            sD = som_data_struct( data' , 'name' , nombre_mapa , 'comp_names' , comp_names , 'labels' , num2str( clase ) );
1712            % Normalizamos cada variable de entrada independientemente en el rango [0 1]
1713            sD = som_normalize( sD , 'range' );
1714
1715            %%%%%%%%%%%%% GENERA LA STRUCT DE DATOS TEST DEL SOM  %%%%%%%%%%%%%%%%%%%%%%%%%%%
1716            sD2 = som_data_struct( data_t' , 'name' , nombre_mapa , 'comp_names' , comp_names , 'labels' , num2str( clase_t ) );
1717            % Normalizamos cada variable de entrada independientemente en el rango [0 1]
1718            sD2 = som_normalize( sD2 , 'range' );
1719
1720
1721            %%%%%%%%%%%%% ENTRENAR EL SOM  %%%%%%%%%%%%%%%%%%%%%%%%%%%
1722            %sM = som_make(sD, 'lattice', lattice); % SOM entrenado con el tamaño del mapa introducido
1723            sM = som_make(sD,'randinit','msize',[26 13], 'lattice', lattice); % SOM entrenado con el tamaño del mapa introducido
1724
1725            %%%%%%%%%%%%% ETIQUETAMOS LAS NEURONAS DEL MAPA A PARTIR DE LOS DATOS %%%%%%%%%%%
1726            [ unit_class_index , unit_class_label] = som_unit_classification( sM , sD , clase);
1727            sM.labels = unit_class_label;
1728            %%%%%%%%%%%%% ETIQUETAMOS LOS DATOS A PARTIR DE LA ACTIVACION DEL MAPA %%%%%%%%%%%%
1729            [ data_class_index , data_class_label ] = som_data_classification( sM , sD2 , unit_class_index );
1730            %%%%%%%%%%%%% CALCULANDO EL KAPPA INDEX %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1731            [ conf_mat , Kappa ] = confussion_matrix( data_class_index , clase_t )
1732             disp('espera un poco...')
1733
1734             %plotconfusion(data_class_index',clase_t')
1735
1736            Kappatemp=[Kappatemp Kappa];
1737
1738          end
1739          Kappatemp=mean(Kappatemp);
1740          subsampleVsKappa=[subsampleVsKappa Kappatemp];
1741          Kappatemp=[];
1742        end
```

```
subsampleVsKappa
figure;
plot(subsampleVsKappa)
disp('pulsa una tecla')
%pause;


%obtenido el subsample optimo, se reducen las entradas
% Testea comp SOM automat
clase = matTargetEntrenaSom;
clase_t=matTargetTestSom;
nombre_mapa = 'SOM NILM';
% Determinando el tamaño del mapa
mapsize = [26 13];
lattice = 'hexa';
data_ini = matEntradasEntrenaSom; % guardamos la data original
data_test=matEntradasTestSom;
[ ndata npat ] = size(data_ini); % npat = numero de patrones (pacientes), ndata = numero de variables

mascara_ini = ones( 1, ndata);    % Mascara incluyendo todas las variables ( 1 = incluir / 0 = suprimir)

maximokappa = 0;          % ultimo kappa maximo
mascara_maximo_matrix =[]; % cada fila sera una mascara que dio kappa maximo
kappa_values_maximos = []; % vector conteniendo los sucesivos kappa maximos
clase=vec2ind(clase)';
clase_t=vec2ind(clase_t)';
while (1)
  kappa_values = zeros( 1 , ndata );
  for k = 1:ndata              % bucle de enmascaramiento para cada componente por separado
    mascara = mascara_ini;    % recargamos en mascara la anterior
    Kappa = 0;
    if mascara(k)            % solo ejecutamos el mapa si la componente de mascara es 1
      mascara(k) = 0;                    % enmascaramos solo la componente k
      indices = find( (mascara == 1) );    % indices de variables NO enmascaradas
      data = data_ini( indices , : );      % Extraemos solamente las variables NO enmascaradas
      data_t=data_test( indices , : );
      comp_names = { names{[indices]} };    % nombres de variables NO enmascaradas
      %%%%%%%%%%%%%% GENERA LA STRUCT DE DATOS DEL SOM  %%%%%%%%%%%%%%%%%%%%%%%%%
      sD = som_data_struct( data' , 'name' , nombre_mapa , 'comp_names' , comp_names , 'labels' , num2str( clase ) );
      % Normalizamos cada variable de entrada independientemente en el rango [0 1]
      sD = som_normalize( sD , 'range' );
      %%%%%%%%%%%%%% ENTRENAR EL SOM  %%%%%%%%%%%%%%%%%%%%%%%%
      sM = som_make(sD, 'msize', mapsize, 'lattice', lattice); % SOM entrenado con el tamaño del mapa introducido

   %validar el som
      %%%%%%%%%%%%%% GENERA LA STRUCT DE DATOS DEL SOM  %%%%%%%%%%%%%%%%%%%%%%%%%
      sD2 = som_data_struct( data_t' , 'name' , nombre_mapa , 'comp_names' , comp_names , 'labels' , num2str( clase_t ) );
      % Normalizamos cada variable de entrada independientemente en el rango [0 1]
      sD2 = som_normalize( sD2 , 'range' );

      %%%%%%%%%%%%%% ETIQUETAMOS LAS NEURONAS DEL MAPA A PARTIR DE LOS DATOS %%%%%%%%%%%%
      [ unit_class_index , unit_class_label] = som_unit_classification( sM , sD , clase);
      sM.labels = unit_class_label;
      %%%%%%%%%%%%%% ETIQUETAMOS LOS DATOS A PARTIR DE LA ACTIVACION DEL MAPA %%%%%%%%%%%%%%
      [ data_class_index , data_class_label ] = som_data_classification( sM , sD2 , unit_class_index );
      %%%%%%%%%%%%% CALCULANDO EL KAPPA INDEX %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      [ conf_mat , Kappa ] = confussion_matrix( data_class_index , clase_t )
    end
    kappa_values(k) = Kappa;
    if ( Kappa >= maximokappa )  % si el nuevo kappa es maximo, guardamos kappa y la mascara
      mascara_maximo_matrix = [ mascara_maximo_matrix ; mascara ];
      kappa_values_maximos = [ kappa_values_maximos Kappa ];
      maximokappa = Kappa;
    end
  end
```

```
        indices_maximos = [ 1:1:ndata ];
        % indices de CARACTERISTICAS cuya eliminacion generaba un kappa maximo
        maximokappa
        indices_maximos = indices_maximos( ( maximokappa == kappa_values ) )
        if ( isempty( indices_maximos ) )
            break;
        else
            mascara_ini( indices_maximos ) = 0; % anulamos todas las CARACTERISTICAS con kappa maximo
        end
        fprintf('\nPulsa cualquier tecla para continuar ...');
        %pause;
    end

    resultados=[kappa_values_maximos', mascara_maximo_matrix];
    disp('    KAPPA      RMS      CCDA      CCDB      FFT1      FFT2      FFT3      FP      PICO     CUARTO')
    disp(resultados)
    k;
    %pause;

    %         'RMS',  'CCDA' 'CCDB' 'FFT1'  'FFT2'   'FFT3'  'FP'   'PICO' 'CUARTO'
    mascara = [ 1      1      1      0      0      0      1      1      0  ];


    %% SOM FINAL

    %prueba definitiva de un SOM con los parametros antes obtenidos
    SOM_Helsinki2.m

    [ conf_mat , Kappa ] = confussion_matrix( data_class_index , clase_t )


     %% PERCEPTRONES FINAL

     %prueba definitiva de perceptron lineal y perceptron multicapa
     %con los parametros antes obtenidos

    indices = find( mascara == 1 ); % indices de entradas no enmascaradas

    matEntradasMLP = matEntradas( indices,: );  % se incluyen sólo las componentes con mascara a 1
    matTargetMLP = matTarget;%( : , indices ); % extraemos las columnas de componentes seleccionadas en la mascara



     %MLP
     entrenos={'trainlm' 'trainbr' 'traingdm' 'trainrp' 'traingda' 'traingdx'...
         'traincgf' 'traincgp' 'traincgb' 'trainscg' 'trainbfg' 'trainoss' }

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
    % Esta máscara decide qué CARACTERISTICAS de entrada se van a utilizar en el problema
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%

    mascaraMlp=mascara;



    % MLP LINEAL

    for i=1:1
        net = newpr(matEntradasMLP,matTargetMLP);
        net.trainFcn='trainlm';
        net.layers{1}.transferFcn = 'tansig';
        %net.layers{2}.transferFcn = 'purelin';
        %net = newff( minmax( data ) , [ numclases ] , { 'logsig' } , funtrain );
```

```
1873            net.divideParam.trainRatio = 60/100;  % Adjust as desired
1874            net.divideParam.valRatio = 20/100;  % Adjust as desired
1875            net.divideParam.testRatio = 20/100;  % Adjust as desired
1876            net.trainParam.max_fail=5;
1877            % Train and Apply Network
1878            [net,tr] = train(net,matEntradasMLP,matTargetMLP);
1879            outputs = sim(net,matEntradasMLP);
1880
1881            % Plot
1882            plotperf(tr)
1883            plotconfusion(matTargetMLP,outputs)
1884
1885            vecTarget=vec2ind(matTargetMLP)';
1886            [Dummy vecOut]=max(outputs);
1887            %vecOut=vec2ind(I)';
1888            vecOut=vecOut';
1889
1890            [ conf_mat , Kappa ] = confussion_matrix( vecTarget' , vecOut );
1891
1892            string=strcat(num2str(sum(mascara)),'_',num2str(nhunit2),'_',num2str(numclases),'_',num2str(funtrain),'_',num2str(i));
1893            tablaMlp{1,end+1}= string;
1894            tablaMlp{2,end}=Kappa;
1895
1896        end
1897
1898        tablaMlp
1899
1900
1901        %MLP MULTILAYER
1902        % Create Network
1903
1904
1905        minNeurOcultas=10;
1906        maxNeuronasOcultas=10;
1907        tablaMlp={};
1908        for nhunit2=minNeurOcultas:maxNeuronasOcultas
1909            for i=1:1
1910                net = newpr(matEntradasMLP,matTargetMLP,nhunit2);
1911                net.trainFcn='trainlm';
1912                net.layers{1}.transferFcn = 'tansig';
1913                net.layers{2}.transferFcn = 'purelin';
1914                %net = newff( minmax( data ) , [ numclases ] , { 'logsig' } , funtrain );
1915                net.divideParam.trainRatio = 60/100;  % Adjust as desired
1916                net.divideParam.valRatio = 20/100;  % Adjust as desired
1917                net.divideParam.testRatio = 20/100;  % Adjust as desired
1918                net.trainParam.max_fail=5;
1919                % Train and Apply Network
1920                [net,tr] = train(net,matEntradasMLP,matTargetMLP);
1921                outputs = sim(net,matEntradasMLP);
1922
1923                % Plot
1924                plotperf(tr)
1925                plotconfusion(matTargetMLP,outputs)
1926
1927                vecTarget=vec2ind(matTargetMLP)';
1928                [Dummy vecOut]=max(outputs);
1929                %vecOut=vec2ind(I)';
1930                vecOut=vecOut';
1931
1932                [ conf_mat , Kappa ] = confussion_matrix( vecTarget' , vecOut );
1933
1934                string=strcat(num2str(sum(mascara)),'_',num2str(nhunit2),'_',num2str(numclases),'_',num2str(funtrain),'_',num2str(i));
1935                tablaMlp{1,end+1}= string;
1936                tablaMlp{2,end}=Kappa;
1937
```

```
1938            end
1939        end
1940        tablaMlp'
1941
1942        %% LVQ FINAL
1943
1944         %prueba definitiva de perceptron lineal y perceptron multicapa
1945         %con los parametros antes obtenidos
1946
1947
1948        tablaLVQ=[];
1949        numEpoch=300;
1950        fa=0.2;
1951        minNeurOcultas=21;
1952        maxNeuronasOcultas=21;
1953        for nhidden=minNeurOcultas:1:maxNeuronasOcultas
1954            for i=1:1
1955                levequ
1956                save(sprintf('LVQ Concordia Ocultas %d.mat ',Unidades))
1957                tablaLVQ=[tablaLVQ Kappa];
1958            end
1959        end
1960
1961        tablaLVQ
1962
1963
1964
```

TDC

ANEXO III

SCRIPT FUNCION SOM_HELSINKI2

```
%clear
%clc
%echo on;
% Este archivo entrena un mapa autoorganizado con la clasificación del Zoo2 (ver Zoo2.m)
% cuya selección de datos de entrada (componentes) es más acertada que en Zoo
% Permite visualizar los mapas generados mediante U-matrix y componentes
% con hits de color diferente para cada clase de animales.
% Se puede incluir un análisis de PCA para averiguar el mínimo número de componentes principales
% que mantiene una adecuada discriminación entre clases.
%echo off;

%% LECTURA DATOS SOM HELSINKI

tamanoMatEntradas=size(matEntradas,2);

%normalizacion
% maxEntradas=max(matEntradas');
% minEntradas=min(matEntradas');
% semiRangoEntradas=((maxEntradas-minEntradas)/2)';
% mitadEntradas=((maxEntradas+minEntradas)/2)';
% a=(matEntradas-(mitadEntradas*(ones(1,tamanoMatEntradas))));
% b=semiRangoEntradas*(ones(1,tamanoMatEntradas));
% matEntradasNorm=a./b;
% matEntradas=matEntradasNorm;


%division dataset

%rellenaremos un vector aleatorio con la columna de la matEntradas que
%formaran parte del entrenamiento y de Test.

%
% incrementoColumna=0;
% columnaAleatoria=zeros(1,tamanoMatEntradas); %Mascara para separar individuos test de entrenamieno
% limiteRand=0;
% for i=1:size(matTarget,1)
%     tamanoColumnaAleatoria(i)=round(numeroMuestras(i)*PORC_ENTRENAMIENTO);
%     incrementoColumna=[incrementoColumna incrementoColumna(i)+numeroMuestras(i)];
%     limiteRand=[limiteRand incrementoColumna(i)+numeroMuestras(i)];
%     while sum(columnaAleatoria(:,incrementoColumna(i)+1:incrementoColumna(i)+numeroMuestras(i)))<tamanoColumnaAleatoria(i)
%         aleatorio=round((limiteRand(i)+1)+(limiteRand(i+1)-(limiteRand(i)+1))*rand);
%         if columnaAleatoria(aleatorio);%si el numero aleatorio ya esta en la lista
%         else                                %no hacer nada
%             columnaAleatoria(aleatorio)=1;% si no está añadir
%         end
%     end
% end
%
% matEntradasEntrenaSom=matEntradas(: ,find(columnaAleatoria));%Se extraen las columnas en "1"
% matEntradasTestSom=matEntradas(: ,find(not(columnaAleatoria)));%el resto son de test
%
% matTargetEntrenaSom=matTarget(: ,find(columnaAleatoria));
% matTargetTestSom=matTarget(: ,find(not(columnaAleatoria)));
%
% %permuta entrenadores
% indicesAleatorios=randperm(size(matEntradasEntrenaSom,2));%se mezclan por igual
```

```
57        % matEntradasEntrenaSom=matEntradasEntrenaSom(indicesAleatorios); %los entrenadores y target
58        % matTargetEntrenaSom=matTargetEntrenaSom(indicesAleatorios);
59        %
60        % clear indicesAleatorios
61        % %permuta test
62        % indicesAleatorios=randperm(size(matEntradasTestSom,2));
63        % matEntradasTestSom=matEntradasTestSom(indicesAleatorios);
64        % matTargetTestSom=matTargetTestSom(indicesAleatorios);
65        %
66        %
67        % clear tamanoColumnaAleatoria columnaAleatoria aleatorio tamanoMatEntradas incrementoColumna...
68        %     indicesAleatorios;
69        %
70
71
72
73
74
75        incrementoColumna=0;
76        columnaAleatoria=zeros(1,tamanoMatEntradas); %Mascara para separar individuos test de entrenamieno
77        limiteRand=0;
78        for i=1:size(matTarget,1)
79           tamanoColumnaAleatoria(i)=round(numeroMuestras(i)*PORC_ENTRENAMIENTO);
80           incrementoColumna=[incrementoColumna incrementoColumna(i)+numeroMuestras(i)];
81           limiteRand=[limiteRand incrementoColumna(i)+numeroMuestras(i)];
82           while sum(columnaAleatoria(:,incrementoColumna(i)+1:incrementoColumna(i)+numeroMuestras(i)))<tamanoColumnaAleatoria(i)
83              aleatorio=round((limiteRand(i)+1)+(limiteRand(i+1)-(limiteRand(i)+1))*rand);
84              if columnaAleatoria(aleatorio);%si el numero aleatorio ya esta en la lista
85              else                                  %no hacer nada
86                 columnaAleatoria(aleatorio)=1;% si no está añadir
87              end
88           end
89        end
90
91        matEntradasEntrenaSom=matEntradas(: ,find(columnaAleatoria));%Se extraen las columnas en "1"
92        matEntradasTestSom=matEntradas(: ,find(not(columnaAleatoria)));%el resto son de test
93
94        matTargetEntrenaSom=matTarget(: ,find(columnaAleatoria));
95        matTargetTestSom=matTarget(: ,find(not(columnaAleatoria)));
96
97
98
99
100
101
102       rasgos = matEntradasEntrenaSom';
103       rasgos_t=matEntradasTestSom';
104
105       clase = matTargetEntrenaSom';
106       clase_t=matTargetTestSom';
107
108       %[ rasgos , minp , maxp ] = premnmx( rasgos );
109       %[ rasgos_t , minp , maxp ] = premnmx( rasgos_t );
110
111       numclases = size(matTarget,1); % numero de clases
112       numpatclases = [ sizeMatEntradasBat1 sizeMatEntradasBat2 sizeMatEntradasBat3 sizeMatEntradasExpr sizeMatEntradasMicr sizeMatEntradasSand
113    sizeMatEntradasVen1 sizeMatEntradasMicrOnYSandOn sizeMatEntradasSandOnYBat3On sizeMatEntradasVen1OnYBat3On
114    sizeMatEntradasVen1OnYMicrOnYSandOn ]; % numero de muestras en cada clase
115
116       %clasem = [ min( clase ) ; max( clase ) ]';
117
118
119       nombre_clases = char('Bat1', 'Bat2', 'Bat3', 'Expr', 'Micr', 'Sand', 'Ven1', 'MicrOnYSandOn', 'SandOnYBat3On', 'Ven1OnYBat3On', 'Ven1OnYMicrOnYSandOn');
120
121       etiquetas = nombre_clases( clase * ( [ 1:numclases ]' ) , : ); % etiquetas de clases
```

```
etiquetas2 = nombre_clases( clase_t * ( [ 1:numclases ]' ) , : ); % etiquetas de clases

componentes = {'RMS','CCDA','CCDB','FFT1','FFT2','FFT3','FP','PICO','CUARTO'};
x_label = '1-RMS/2-CCDA/3-CCDB/4-FFT1/5-FFT2/6-FFT3/7-FP/8-PICO/9-CUARTO';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
% Esta máscara decide qué componentes de entrada se van a utilizar en el problema
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
%        'RMS',  'CCDA' 'CCDB' 'FFT1'  'FFT2'   'FFT3'  'FP'   'PICO' 'CUARTO'
%mascara = [ 1      1      1      0      0       0      1      1      0  ];

indices = find( mascara == 1 ); % indices de entradas no enmascaradas

componentes = componentes( indices );  % se incluyen sólo las componentes con mascara a 1
rasgos = rasgos( : , indices ); % extraemos las columnas de componentes seleccionadas en la mascara
rasgos_t=rasgos_t( : , indices );
%rasgosm = [ min( rasgos ) ; max( rasgos ) ]'; % valores minimos y maximos en las entradas
%[ npat ndat ] = size( rasgos ); % npat = numero de patrones; ndat = numero de entradas o componentes;


%% ENTRENAMIENTO Y VISUALIZACION


%%%%%%%%%%%%%% GENERA LA STRUCT DE DATOS DEL SOM  %%%%%%%%%%%%%%%%%%%%%%%%
sD = som_data_struct( rasgos , 'name' , nombre_mapa,  'comp_names' , componentes , 'labels' , etiquetas );
sD2= som_data_struct( rasgos_t , 'name' , nombre_mapa,'comp_names' , componentes , 'labels' , etiquetas2 );

%mapsize = [];
%mapsize = input('\nIntroduce el tamaño del mapa entre corchetes (por ejemplo, [6 5]): ');
%lattice = input('\nIntroduce el tipo de mapa (0 = rectangular, 1 = hexagonal): ');
mapsize=[26 20];
lattice = 'hexa';

%%%%%%%%%%%%%% ENTRENAR EL SOM  %%%%%%%%%%%%%%%%%%%%%%%%
sM = som_make( sD , 'randinit','msize' , mapsize , 'lattice' , lattice ); % SOM entrenado con el tamaño del mapa introducido

%%%%%%%%%%%%%% ETIQUETAMOS LAS NEURONAS DEL MAPA A PARTIR DE LOS DATOS %%%%%%%%%%%
[ unit_class_index , unit_class_label] = som_unit_classification( sM , sD , vec2ind(clase'));
sM.labels = unit_class_label;

%%%%%%%%%%%%%% ETIQUETAMOS LOS DATOS A PARTIR DE LA ACTIVACION DEL MAPA %%%%%%%%%%%%%
[ data_class_index , data_class_label ] = som_data_classification( sM , sD2 , unit_class_index );

[ conf_mat , Kappa ] = confussion_matrix( data_class_index , vec2ind(clase_t') )


colores =  flipdim(repmat(linspace(0.5,1,64)',1,3).*repmat([0.8 0.9 1],64,1),1); % 'blueish'

warning off;

%%%%%%%%%%%%%%%%%% Representamos el mapa en cuatro formas diferentes  %%%%%%%%%%%%%

Hfigure = figure; % Handle de la figura
set( Hfigure , 'NumberTitle' , 'off' );
set( Hfigure , 'Name' , 'VISUALIZACIONES DE LA U-MATRIX Y D-MATRIX' ); % Ponemos nomnbre a la figura
U = som_umat( sM ); % calcula U-matrix
colormap( colores );

subplot( 2 , 2 , 1 ); % pinta la U-Matrix
h = som_cplane( [ sM.topol.lattice , 'U' ] , sM.topol.msize , U(:) );
set( h , 'Edgecolor' , 'none' );
title('U-matrix');
```

```
subplot( 2 , 2 , 2 ); % pinta la D-matrix
Um = U( 1:2:size( U , 1 ) , 1:2:size( U , 2 ) ); % extrae las componentes impares de la U-Matrix
h = som_cplane( sM , Um(:) );
set( h , 'Edgecolor' , 'none' );
title('D-matrix');

subplot( 2 , 2 , 3 ); % pinta la D-matrix con hexágonos de tamaño variable
som_cplane( sM , 'none' , 1 - Um(:) / max( Um(:) ) );
title('D-matrix (marcadores de tamaño)');

[ Pd , V , me ] = pcaproj( sD.data , 2 );  % obtiene matriz de proyeccion PCA de los datos en dos dimensiones
Pm = pcaproj( sM.codebook , V , me );      % proyecta con esa matriz los pesos para la paleta de color
subplot( 2 , 2 , 4 );  % pinta la D-matrix con la paleta de colores las distancias
C = som_colorcode( Pm );  % Paleta de colores
som_cplane( sM , C );
title('Similitud por coloración');

% Pinta la U-matrix con las activaciones de las muestras de las clases en colores
% Se representan conjuntamente los mapas de la distribución de valores de cada componente de entrada
Hfigure = figure; % Handle de la figura
temp=0;
for i=1:CLASES
   Dsample{i}=sD.data(temp+1:temp+numpatclases(i) , :);
   temp=numpatclases(i);
end
% Dsample1 = sD.data( 1:numpatclases(1) , : );   % muestras de mamíferos
% Dsample2 = sD.data( numpatclases(1)+1:numpatclases(1)+numpatclases(2) , : );  % muestras de aves
% Dsample3 = sD.data( numpatclases(2)+1:numpatclases(3) , : );  % muestras de aves
% Dsample4 = sD.data( numpatclases(3)+1:numpatclases(4) , : );  % muestras de aves
% Dsample5 = sD.data( numpatclases(4)+1:numpatclases(5) , : );  % muestras de aves
% temp=numpatclases(1)+numpatclases(2);
% Dsample6 = sD.data( numpatclases(5)+1:numpatclases(6) , : );  % muestras de aves
% temp=numpatclases(1)+numpatclases(2);
% Dsample7 = sD.data( numpatclases(6)+1:numpatclases(7) , : );  % muestras de aves
% Dsample8 = sD.data( numpatclases(7)+1:numpatclases(8) , : );  % muestras de aves
% Dsample9 = sD.data( numpatclases(8)+1:numpatclases(9) , : );  % muestras de aves
% Dsample10 = sD.data( numpatclases(9)+1:numpatclases(10) , : );  % muestras de aves
% Dsample11= sD.data( numpatclases(10)+1:numpatclases(11) , : );  % muestras de aves
%

h1 = som_hits( sM , Dsample{1} ); % obtenemos los histogramas de activación para las muestras de mamíferos
h2 = som_hits( sM , Dsample{2} ); % para las muestras de aves
h3 = som_hits( sM , Dsample{3} ); % para las muestras de reptiles
h4 = som_hits( sM , Dsample{4} ); % para las muestras de peces
h5 = som_hits( sM , Dsample{5} ); % para las muestras de artrópodos
h6 = som_hits( sM , Dsample{6} ); % para las muestras de artrópodos
h7 = som_hits( sM , Dsample{7} ); % para las muestras de artrópodos
h8 = som_hits( sM , Dsample{8} ); % para las muestras de artrópodos
h9 = som_hits( sM , Dsample{9} ); % para las muestras de artrópodos
h10 = som_hits( sM , Dsample{10} ); % para las muestras de artrópodos
h11 = som_hits( sM , Dsample{11} ); % para las muestras de artrópodos

% Respuesta difusa calculada sumando 1./(1+(q/a)^2)
% para cada muestra, donde 'q' es un vector que contiene
% la distancia de cada muestra hasta cada prototipo de las unidades del mapa
% y 'a' es el promedio del error de cuantización
hf = som_hits( sM , sD.data , 'fuzzy' );

% Representamos los histogramas en U-matrices separadas
colormap(colores);
som_show( sM , 'umat' , {'all','Bat1'} , 'umat' , {'all','Bat2'}...
   , 'umat' , {'all','Bat3'} , 'umat' , {'all','Expr'}...
   , 'umat' , {'all','Micr'}, 'umat' , {'all','Sand'}...
   , 'umat' , {'all','Ven1'}, 'umat' , {'all','MicrOnYSandOn'}...
   , 'umat' , {'all','SandOnYBat3On'}, 'umat' , {'all','Ven1OnYBat3On'}...
```

```
252         , 'umat' , {'all','VenIOnYMicrOnYSandOn'}, 'color' , {hf,'Fuzzy response'} );
253     som_show_add( 'hit' , h1 , 'Subplot' , 1 , 'Markercolor' , 'r');  % mamíferos en rojo
254     som_show_add( 'hit' , h2 , 'Subplot' , 2 , 'Markercolor' , 'b');  % aves en azul
255     som_show_add( 'hit' , h3 , 'Subplot' , 3 , 'Markercolor' , 'g');  % reptiles en verde
256     som_show_add( 'hit' , h4 , 'Subplot' , 4 , 'Markercolor' , 'm');  % peces en magenta
257     som_show_add( 'hit' , h5 , 'Subplot' , 5 , 'Markercolor' , 'y');  % artrópodos en amarillo
258     som_show_add( 'hit' , h6 , 'Subplot' , 6 , 'Markercolor' , 'r');  % artrópodos en amarillo
259     som_show_add( 'hit' , h7 , 'Subplot' , 7 , 'Markercolor' , 'b');  % artrópodos en amarillo
260     som_show_add( 'hit' , h8 , 'Subplot' , 8 , 'Markercolor' , 'g');  % artrópodos en amarillo
261     som_show_add( 'hit' , h9 , 'Subplot' , 9 , 'Markercolor' , 'm');  % artrópodos en amarillo
262     som_show_add( 'hit' , h10 , 'Subplot' , 10 , 'Markercolor' , 'y');  % artrópodos en amarillo
263     som_show_add( 'hit' , h11 , 'Subplot' , 11 , 'Markercolor' , 'r');  % artrópodos en amarillo
264
265
266
267
268     set( Hfigure , 'NumberTitle' , 'off' );
269     set( Hfigure , 'Name' , 'VISUALIZACION DE LA ACTIVACION DETALLADA' ); % Ponemos nomnbre a la figura
270
271     % Representamos la U-matrix y las matrices de distancias para cada componente de entrada
272     Hfigure = figure; % Handle de la figura
273     colormap(colores);
274     som_show( sM );
275     % Añadimos los histogramas de activación sobre la U-matrix
276     som_show_add( 'hit' , h1 , 'Subplot' , 1 , 'Markercolor' , 'r' );  % mamíferos en rojo
277     som_show_add( 'hit' , h2 , 'Subplot' , 1 , 'Markercolor' , 'b' );  % aves en azul
278     som_show_add( 'hit' , h3 , 'Subplot' , 1 , 'Markercolor' , 'g' );  % reptiles en verde
279     som_show_add( 'hit' , h4 , 'Subplot' , 1 , 'Markercolor' , 'm' );  % peces en magenta
280     som_show_add( 'hit' , h5 , 'Subplot' , 1 , 'Markercolor' , 'y' );  % artrópodos en amarillo
281     som_show_add( 'hit' , h6 , 'Subplot' , 1 , 'Markercolor' , 'r' );  % artrópodos en amarillo
282     som_show_add( 'hit' , h7 , 'Subplot' , 1 , 'Markercolor' , 'b' );  % artrópodos en amarillo
283     som_show_add( 'hit' , h8 , 'Subplot' , 1 , 'Markercolor' , 'g' );  % artrópodos en amarillo
284     som_show_add( 'hit' , h9 , 'Subplot' , 1 , 'Markercolor' , 'm' );  % artrópodos en amarillo
285     som_show_add( 'hit' , h10 , 'Subplot' , 1 , 'Markercolor' , 'y' );  % artrópodos en amarillo
286     som_show_add( 'hit' , h11 , 'Subplot' , 1 , 'Markercolor' , 'r' );  % artrópodos en amarillo
287     set( Hfigure , 'NumberTitle' , 'off' );
288     set( Hfigure , 'Name' , 'VISUALIZACION DE LAS COMPONENTES' ); % Ponemos nomnbre a la figura
289
290
```

TDC

ANEXO IV

SCRIPT FUNCIÓN LEVEQU

```
% Este fichero entrena una red LVQ
% y ejecuta la rutina newlvq para crear la red
clc


%% LECTURA DATOS LVQ


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%
% Esta máscara decide qué componentes de entrada se van a utilizar en el problema
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%

%        'RMS',  'CCDA' 'CCDB' 'FFT1'  'FFT2'   'FFT3'  'FP'   'PICO' 'CUARTO'
%mascara = [ 1     1     1     0     0      0     1     1     1  ];



tamanoMatEntradas=size(matEntradas,2);

incrementoColumna=0;
columnaAleatoria=zeros(1,tamanoMatEntradas); %Mascara para separar individuos test de entrenamieno
limiteRand=0;
for i=1:size(matTarget,1)
   tamanoColumnaAleatoria(i)=round(numeroMuestras(i)*PORC_ENTRENAMIENTO);
   incrementoColumna=[incrementoColumna incrementoColumna(i)+numeroMuestras(i)];
   limiteRand=[limiteRand incrementoColumna(i)+numeroMuestras(i)];
   while sum(columnaAleatoria(:,incrementoColumna(i)+1:incrementoColumna(i)+numeroMuestras(i)))<tamanoColumnaAleatoria(i)
      aleatorio=round((limiteRand(i)+1)+(limiteRand(i+1)-(limiteRand(i)+1))*rand);
      if columnaAleatoria(aleatorio);%si el numero aleatorio ya esta en la lista
      else                                    %no hacer nada
         columnaAleatoria(aleatorio)=1;% si no está añadir
      end
   end
end

matEntradasEntrenaLvq=matEntradas(: ,find(columnaAleatoria));%Se extraen las columnas en "1"
matEntradasTestLvq=matEntradas(: ,find(not(columnaAleatoria)));%el resto son de test

matTargetEntrenaLvq=matTarget(: ,find(columnaAleatoria));
matTargetTestLvq=matTarget(: ,find(not(columnaAleatoria)));

% %permuta entrenadores
% indicesAleatorios=randperm(size(matEntradasEntrenaLvq,2));%se mezclan por igual
% matEntradasEntrenaLvq=matEntradasEntrenaLvq(:,indicesAleatorios); %los entrenadores y target
% matTargetEntrenaLvq=matTargetEntrenaLvq(:,indicesAleatorios);
%
% clear indicesAleatorios
% %permuta test
% indicesAleatorios=randperm(size(matEntradasTestLvq,2));
% matEntradasTestLvq=matEntradasTestLvq(:,indicesAleatorios);
% matTargetTestLvq=matTargetTestLvq(:,indicesAleatorios);
```

```
57        clear tamanoColumnaAleatoria columnaAleatoria aleatorio tamanoMatEntradas incrementoColumna...
58           indicesAleatorios;
59
60        indices = find((mascara == 1));          % indices de variables NO enmascaradas
61        vino = matEntradasEntrenaLvq( indices , : );              % Extraemos solamente las variables NO enmascaradas
62        %clases=matTargetEntrenaLvq;
63        clases=vec2ind(matTargetEntrenaLvq);
64        clases=clases';
65        clases=ind2vec(clases);
66        %clases=clases;
67        matEntradasTestReducidoLvq=matEntradasTestLvq( indices , : );
68
69
70        numpatclases = full( sum( clases , 1 ) ); % numero de patrones de cada clase
71        numclases = length( numpatclases ); % numero de clases
72        [ numinp , npat ] = size( vino ); % numero de entradas y de patrones
73        prob_priori = numpatclases / npat; % probabilidad a priori de las clases
74        names_variables = {'Bat1', 'Bat2', 'Bat3', 'Expr', 'Micr', 'Sand', 'Ven1', 'Micr0nYSand0n', 'Sand0nYBat30n', 'Ven10nYBat30n', 'Ven10nYMicr0nYSand0n'};
75        %numpatclases = [ sizeMatEntradasBat1 sizeMatEntradasBat2 sizeMatEntradasBat3 sizeMatEntradasExpr sizeMatEntradasMicr sizeMatEntradasSand
76     sizeMatEntradasVen1 sizeMatEntradasMicr0nYSand0n sizeMatEntradasSand0nYBat30n sizeMatEntradasVen10nYBat30n
77     sizeMatEntradasVen10nYMicr0nYSand0n ]; % numero de muestras en cada clase
78
79
80        %% LVQ
81        % flagesc = input('\nEscalado (0=no/1=si, por defecto 1): ');
82        % % Escalamos todas las entradas para que tengan variación entre -1 y 1
83        % if isempty( flagesc );flagesc = 1;end
84        % if (flagesc == 0)
85        %    data = vino;
86        % else
87        %    [data , minp , maxp ] = premnmx( vino );
88        % end
89
90        [data , minp , maxp ] = premnmx( vino );
91
92        % odim = input('\nDimensión del espacio de proyección de PCA (1-4, por defecto NO PCA): ');
93        % if isempty( odim )
94        %     datapca = data;
95        % else
96        % [ P , V , me , L ] = pcaproj( data' , odim );
97        % datapca = P';
98        % end
99        datapca = data;
100
101       [ dim , npat ] = size( datapca );
102
103       % Generando una red LVQ
104       % nhidden = input('\nNúmero de unidades ocultas (por defecto 24): ');
105       % if isempty( nhidden ); nhidden = 24; end
106       % fa = input('\nFactor de aprendizaje (por defecto 0.2): ');
107       % if isempty( fa ); fa = 0.2; end
108       %out = full( clases )';
109       out = full( clases );
110       claspat = sum( out' , 1 ); % sumamos los indices de clases
111       nclass = length( claspat );
112       PerC = claspat / npat; % porcentajes de clases en los patrones
113
114       net = newlvq( minmax( datapca ) , nhidden , PerC , fa );
115       % Escribe el tamaño de la capa de unidades con kernel
116       fprintf('\nLa red generada utiliza:');
117       [ Unidades , Datos ] = size( net.IW{ 1 , 1 } )
118
119       % Usamos como método de inicialización de pesos la selección aleatoria de muestras
120       w2 = net.LW{ 2 , 1 }; % pesos desde unidades de salida a capa competitiva UsalidaxUhidden
121       [ nclases , nhidden ] = size( w2 );
```

```
122 rep = 0;  % los centroides se adjudican sin repetir muestras
123 if ( nhidden > npat ); rep = 1; end;
124 [ w , ind ] = initwpat( datapca' , nhidden , out' , rep );% obtiene una matriz w con patrones
125 net.IW{ 1 , 1 } = w;
126
127 % Dibujamos los datos en el subespacio proyectado
128 % if ( odim >= 3)
129 %    ind = input('Tres índices de componentes para visualizar (por defecto [1 2 3]): ');
130 %    if isempty( ind ); ind = [ 1 2 3 ]; end;
131 %    H = drawkernel( ind , datapca , out , net );
132 % else if (odim == 2)
133 %       H = drawkernel( [ 1 2 ] , datapca , out , net );
134 %    end
135 % end
136 % title('VALORES INICIALES DE LOS CENTROIDES DE LA RED LVQ');
137 % fprintf('\nPulsar cualquier tecla para iniciar el entrenamiento.');
138 % pause;
139
140
141 % Entreno
142 net.trainfcn='trainr'; % el entrenamiento se hace por iteraciones; esta sentencia pone epochs a 100 por defecto
143 net.trainParam.show = 100;
144 %net.trainParam.epochs = 3000;%3000
145 net.trainParam.epochs = numEpoch;
146 net.adaptParam.passes = 10000;
147 net.inputWeights{ 1 , 1 }.learnFcn = 'learnlvl';
148 % net.inputWeights{1,1}.learnFcn = 'learnlv2';
149 net = train( net , datapca , out );
150 % net = adapt( net , datapca , out);
151
152
153 % Test
154
155 vino = matEntradasTestLvq( indices , : );            % Extraemos solamente las variables NO enmascaradas
156 [data , minp , maxp ] = premnmx( vino );
157 %datapca=matEntradasTestReducidoLvq;
158 datapca=data;
159 Y = sim( net , datapca );
160 numpatclases=sum(matTargetTestLvq');
161 % funcion que pinta graficos de barras para cada unidad con el conjunto de respuestas
162 base = 'Bat1 / Bat2 / Bat3 / Expr / Micr / Sand / Ven1 / MicrOnYSandOn / SandOnYBat3On / Ven1OnYBat3On / Ven1OnYMicrOnYSandOn';
163 xticks = [ 1 ];
164    for n = 1:size(matTargetTestLvq,1)
165       xticks = [ xticks ; sum( numpatclases( 1:n ) ) ]; % Puntos de eje X de las graficas
166    end
167 H = barcompout( Y , 'U' , base , xticks );
168
169 % Dibujamos los datos en el subespacio proyectado
170 % if (odim >= 3)
171 %    H = drawkernel( ind , datapca , out , net );
172 % else if (odim == 2)
173 %       H = drawkernel( [ 1 2 ] , datapca , out , net );
174 %    end
175 % end
176 %title('VALORES FINALES DE LOS CENTROIDES DE LA RED LVQ');
177 [ conf_mat , Kappa ] = confussion_matrix( vec2ind(matTargetTestLvq) , vec2ind(Y) );
178 plotconfusion(matTargetTestLvq,Y)
179
180
```

TDC

ANEXO V

SCRIPT DETECCIÓN DE EVENTOS

```
1
2       clc;
3       clear all;
4       %close all;
5
6       % for i=1:TRIALS
7       %     figure;
8       %     plot(FacPotBatI{1,i}(1,1:end),'DisplayName','cargaBat3{1,1}(1,1:15360)','YDataSource','cargaBat3{1,1}(1,1:15360)');figure(gcf)
9       %
10      % end
11
12
13      %% CONSTANTES
14      Kappatemp=[];
15      subsampleVsKappa=[];
16      %for subsamples=4:4
17      subsamples=4;
18      %    for veces=1:1
19      TRIALS=10;
20      CLASES=11;
21      names = {'RMS','CCDA','CCDB','FFT1','FFT2','FFT3','FP','PICO','CUARTO'};
22      VENTANA_EST=10;%50 ciclos de tamano de muestra
23      COMP_CONTINUA=600;
24      % FS=2542; %Frecuencia de sampleo original
25      SUBSAMPLE=subsamples; %se guarda una de cada SUBSAMPLE muestras
26      FS=round(2542/SUBSAMPLE); %Frecuencia de sampleo
27      MUESTRAS_CICLO=round(FS*0.02); %número de muestras en cada ciclo por
28
29      AMP_POR_UNIDAD=0; %0.15amperios por unidad de matlab
30      %50 muestras por periodo
31
32      INC_VENT=VENTANA_EST*MUESTRAS_CICLO;
33      %
34      % %FFT
35      % T=1/FS; %periodo
36      % NFFT = 2^nextpow2(INC_VENT); % Next power of 2 from length of y
37      % FRC_BUSQ_FFT=[40 60 120 180 200 300]; %rango frecuencias para buscar las componentes FFT
38      % VECTOR_FRECUENCIA = FS/2*linspace(0,1,NFFT/2+1);
39      %
40      %
41      %filtro
42      N=100;
43      FC1=40;
44      FC2=60;
45      INICIO_FILTR=1;%50
46      DESF_FILTRO=50; %muestras de desfase entre señal original y filtrada.
47      %
48      % DESFASE_TRAFO_TENSION=0;%desfase que introduce el trafo de medida de tension
49      % SEP_PASOS_0=round(MUESTRAS_CICLO/3.33);%15;%46
50      % MAY_MEN_0=0;
51
52      WAKE_UP=FS*1; %muestras (tiempo) entre despertar y despertar (1 segundo)
53      VENTANA_EVENTO=10*MUESTRAS_CICLO;%ancho ventana de evento a on
54      THRESHOLD=270;% umbral para considerar un evento a ON
55      GRADIENTE=70;% diferencia < entre ventanas para considerar fin transitorio
56      %MAX_TRANS=4;%
```

```
57
58      % %CUARTO_SEMIPERIODO= round (MUESTRAS_CICLO/8);
59      % CUARTO_SEMIPERIODO= round (MUESTRAS_CICLO/14); %donde más se nota el 3º armonico, con 5º y 7º 10% del 3º
60      % VALOR_CUARTO_SEMIPERIODO_NORMALIZADO=0.4226;%Tanto por uno del valor de pico
61      % %VALOR_CUARTO_SEMIPERIODO_NORMALIZADO=-0.648228395307788;%Tanto por uno del valor de pico
62      %       %que se obtendría en un seno puro en el cuarto del segundo semiperiodo
63      % PORC_ENTRENAMIENTO=80/100;%porcentaje del data set destinado al entrenamiento
64
65
66      %% CARGA FICHEROS Y ACONDICIONAMIENTO
67
68
69      %Carga ficheros
70      for i=1:TRIALS
71          cargaBat1{i}=load(strcat('PICUS_BAT1_TRIAL',num2str(i),'.txt'));
72          cargaBat2{i}=load(strcat('PICUS_BAT2_TRIAL',num2str(i),'.txt'));
73          cargaBat3{i}=load(strcat('PICUS_BAT3_TRIAL',num2str(i),'.txt'));
74          cargaExpr{i}=load(strcat('PICUS_EXPR_TRIAL',num2str(i),'.txt'));
75          cargaMicr{i}=load(strcat('PICUS_MICR_TRIAL',num2str(i),'.txt'));
76          cargaSand{i}=load(strcat('PICUS_SAND_TRIAL',num2str(i),'.txt'));
77          cargaVen1{i}=load(strcat('PICUS_VEN1_TRIAL',num2str(i),'.txt'));
78      %    cargaVen2{i}=load(strcat('PICUS_VEN2_TRIAL',num2str(i),'.txt'));
79          cargaMicrOnYSandOn{i}=load(strcat('PICUS_MICR_SAND_SAND_MICRO_TRIAL',num2str(i),'.txt'));
80          cargaSandOnYBat3On{i}=load(strcat('PICUS_SAND_BAT3_BAT3_SAND_TRIAL',num2str(i),'.txt'));
81          cargaVen1OnYBat3On{i}=load(strcat('PICUS_VEN1_BAT3_BAT3_VENTI_TRIAL',num2str(i),'.txt'));
82          cargaVen1OnYMicrOnYSandOn{i}=load(strcat('PICUS_VEN1_MICR_SAND_SAND_MICR_VEN1_TRIAL',num2str(i),'.txt'));
83      end
84
85
86
87
88
89      %Subsampleo
90      %recorremos el vector de uno en uno y si encontramos un cero se guarda
91      %como el cero no es parte de la onda, incremento el indice de subsampleo a
92      %la siguiente muestra. Si coincide muestra leida con el indice subsampleo
93      %guardo esa muestra y e incremento el indiceSub en SUBSAMPLE
94
95
96
97      for i=1:TRIALS
98          cargaBat1Temp{i}=[];
99          cargaBat2Temp{i}=[];
100         cargaBat3Temp{i}=[];
101         cargaExprTemp{i}=[];
102         cargaMicrTemp{i}=[];
103         cargaSandTemp{i}=[];
104         cargaVen1Temp{i}=[];
105         % cargaVen2Temp{1}=[];
106         cargaMicrOnYSandOnTemp{i}=[];
107         cargaSandOnYBat3OnTemp{i}=[];
108         cargaVen1OnYBat3OnTemp{i}=[];
109         cargaVen1OnYMicrOnYSandOnTemp{i}=[];
110
111
112         indiceSub=1;
113         j=1;
114         while (j<size(cargaBat1{1,i},2))
115             if(cargaBat1{1,i}(j)==0)
116                 cargaBat1Temp{1,i}(end+1)=0;
117                 indiceSub=indiceSub+1;
118             end
119             if (j==indiceSub)
120                 cargaBat1Temp{1,i}(end+1)=cargaBat1{1,i}(j);
121                 indiceSub=indiceSub+SUBSAMPLE;
```

```
122             end
123             j=j+1;
124         end
125
126         indiceSub=1;
127         j=1;
128         while (j<size(cargaBat2{1,i},2))
129             if(cargaBat2{1,i}(j)==0)
130                 cargaBat2Temp{1,i}(end+1)=0;
131                 indiceSub=indiceSub+1;
132             end
133             if (j==indiceSub)
134                 cargaBat2Temp{1,i}(end+1)=cargaBat2{1,i}(j);
135                 indiceSub=indiceSub+SUBSAMPLE;
136             end
137             j=j+1;
138         end
139
140         indiceSub=1;
141         j=1;
142         while (j<size(cargaBat3{1,i},2))
143             if(cargaBat3{1,i}(j)==0)
144                 cargaBat3Temp{1,i}(end+1)=0;
145                 indiceSub=indiceSub+1;
146             end
147             if (j==indiceSub)
148                 cargaBat3Temp{1,i}(end+1)=cargaBat3{1,i}(j);
149                 indiceSub=indiceSub+SUBSAMPLE;
150             end
151             j=j+1;
152         end
153
154         indiceSub=1;
155         j=1;
156         while (j<size(cargaExpr{1,i},2))
157             if(cargaExpr{1,i}(j)==0)
158                 cargaExprTemp{1,i}(end+1)=0;
159                 indiceSub=indiceSub+1;
160             end
161             if (j==indiceSub)
162                 cargaExprTemp{1,i}(end+1)=cargaExpr{1,i}(j);
163                 indiceSub=indiceSub+SUBSAMPLE;
164             end
165             j=j+1;
166         end
167
168         indiceSub=1;
169         j=1;
170         while (j<size(cargaMicr{1,i},2))
171             if(cargaMicr{1,i}(j)==0)
172                 cargaMicrTemp{1,i}(end+1)=0;
173                 indiceSub=indiceSub+1;
174             end
175             if (j==indiceSub)
176                 cargaMicrTemp{1,i}(end+1)=cargaMicr{1,i}(j);
177                 indiceSub=indiceSub+SUBSAMPLE;
178             end
179             j=j+1;
180         end
181
182         indiceSub=1;
183         j=1;
184         while (j<size(cargaSand{1,i},2))
185             if(cargaSand{1,i}(j)==0)
186                 cargaSandTemp{1,i}(end+1)=0;
```

```
            indiceSub=indiceSub+1;
        end
        if (j==indiceSub)
            cargaSandTemp{1,i}(end+1)=cargaSand{1,i}(j);
            indiceSub=indiceSub+SUBSAMPLE;
        end
        j=j+1;
    end

    indiceSub=1;
    j=1;
    while (j<size(cargaVenI{1,i},2))
        if(cargaVenI{1,i}(j)==0)
            cargaVenITemp{1,i}(end+1)=0;
            indiceSub=indiceSub+1;
        end
        if (j==indiceSub)
            cargaVenITemp{1,i}(end+1)=cargaVenI{1,i}(j);
            indiceSub=indiceSub+SUBSAMPLE;
        end
        j=j+1;
    end

    indiceSub=1;
    j=1;
    while (j<size(cargaMicrOnYSandOn{1,i},2))
        if(cargaMicrOnYSandOn{1,i}(j)==0)
            cargaMicrOnYSandOnTemp{1,i}(end+1)=0;
            indiceSub=indiceSub+1;
        end
        if (j==indiceSub)
            cargaMicrOnYSandOnTemp{1,i}(end+1)=cargaMicrOnYSandOn{1,i}(j);
            indiceSub=indiceSub+SUBSAMPLE;
        end
        j=j+1;
    end

    indiceSub=1;
    j=1;
    while (j<size(cargaSandOnYBat3On{1,i},2))
        if(cargaSandOnYBat3On{1,i}(j)==0)
            cargaSandOnYBat3OnTemp{1,i}(end+1)=0;
            indiceSub=indiceSub+1;
        end
        if (j==indiceSub)
            cargaSandOnYBat3OnTemp{1,i}(end+1)=cargaSandOnYBat3On{1,i}(j);
            indiceSub=indiceSub+SUBSAMPLE;
        end
        j=j+1;
    end

    indiceSub=1;
    j=1;
    while (j<size(cargaVenIOnYBat3On{1,i},2))
        if(cargaVenIOnYBat3On{1,i}(j)==0)
            cargaVenIOnYBat3OnTemp{1,i}(end+1)=0;
            indiceSub=indiceSub+1;
        end
        if (j==indiceSub)
            cargaVenIOnYBat3OnTemp{1,i}(end+1)=cargaVenIOnYBat3On{1,i}(j);
            indiceSub=indiceSub+SUBSAMPLE;
        end
        j=j+1;
    end

```

```
252             indiceSub=1;
253             j=1;
254             while (j<size(cargaVen1OnYMicrOnYSandOn{1,i},2))
255                 if(cargaVen1OnYMicrOnYSandOn{1,i}(j)==0)
256                     cargaVen1OnYMicrOnYSandOnTemp{1,i}(end+1)=0;
257                     indiceSub=indiceSub+1;
258                 end
259                 if (j==indiceSub)
260                     cargaVen1OnYMicrOnYSandOnTemp{1,i}(end+1)=cargaVen1OnYMicrOnYSandOn{1,i}(j);
261                     indiceSub=indiceSub+SUBSAMPLE;
262                 end
263                 j=j+1;
264             end
265
266             cargaBat1{1,i}=cargaBat1Temp{1,i};
267             cargaBat2{1,i}=cargaBat2Temp{1,i};
268             cargaBat3{1,i}=cargaBat3Temp{1,i};
269             cargaExpr{1,i}=cargaExprTemp{1,i};
270             cargaMicr{1,i}=cargaMicrTemp{1,i};
271             cargaSand{1,i}=cargaSandTemp{1,i};
272             cargaVen1{1,i}=cargaVen1Temp{1,i};
273         %    cargaVen2{1,i}=cargaVen1Temp{1,i};
274             cargaMicrOnYSandOn{1,i}=cargaMicrOnYSandOnTemp{1,i};
275             cargaSandOnYBat3On{1,i}=cargaSandOnYBat3OnTemp{1,i};
276             cargaVen1OnYBat3On{1,i}=cargaVen1OnYBat3OnTemp{1,i};
277             cargaVen1OnYMicrOnYSandOn{1,i}=cargaVen1OnYMicrOnYSandOnTemp{1,i};
278
279         end
280
281             clear cargaBat1Temp{1,i};
282             clear cargaBat2Temp{1,i};
283             clear cargaBat3Temp{1,i};
284             clear cargaExprTemp{1,i};
285             clear cargaMicrTemp{1,i};
286             clear cargaSandTemp{1,i};
287             clear cargaVen1Temp{1,i};
288         %   clear  cargaVen2Temp{1,i};
289             clear cargaMicrOnYSandOnTemp{1,i};
290             clear cargaSandOnYBat3OnTemp{1,i};
291             clear cargaVen1OnYBat3OnTemp{1,i};
292             clear cargaVen1OnYMicrOnYSandOnTemp{1,i};
293             clear indiceSub;
294
295
296         %filtrado de señal
297         % f=fdesign.lowpass('N,Fc',10,70,FS);
298         % filtroLP = design(f,'butter');
299
300         f2=fdesign.bandpass('n,fc1,fc2',N,FC1,FC2,FS);
301         filtroPB=design(f2);
302
303         for i=1:TRIALS
304
305             cargaBat1{1,i}=nonzeros(cargaBat1{1,i});
306             cargaBat1Filtr{1,i}=filter(filtroPB,cargaBat1{1,i});
307             cargaBat1Filtr{1,i}=cargaBat1Filtr{1,i}(INICIO_FILTR:end); %quitamos las INICIO_FILTR muestas
308
309             cargaBat2{1,i}=nonzeros(cargaBat2{1,i});
310             cargaBat2Filtr{1,i}=filter(filtroPB,cargaBat2{1,i});
311             cargaBat2Filtr{1,i}=cargaBat2Filtr{1,i}(INICIO_FILTR:end); %quitamos las INICIO_FILTR muestas
312
313             cargaBat3{1,i}=nonzeros(cargaBat3{1,i});
314             cargaBat3Filtr{1,i}=filter(filtroPB,cargaBat3{1,i});
315             cargaBat3Filtr{1,i}=cargaBat3Filtr{1,i}(INICIO_FILTR:end); %quitamos las INICIO_FILTR muestas
316
```

```
317             cargaExpr{l,i}=nonzeros(cargaExpr{l,i});
318             cargaExprFiltr{l,i}=filter(filtroPB,cargaExpr{l,i});
319             cargaExprFiltr{l,i}=cargaExprFiltr{l,i}(INICIO_FILTR:end); %quitamos las INICIO_FILTR muestas
320
321             cargaMicr{l,i}=nonzeros(cargaMicr{l,i});
322             cargaMicrFiltr{l,i}=filter(filtroPB,cargaMicr{l,i});
323             cargaMicrFiltr{l,i}=cargaMicrFiltr{l,i}(INICIO_FILTR:end); %quitamos las INICIO_FILTR muestas
324
325             cargaSand{l,i}=nonzeros(cargaSand{l,i});
326             cargaSandFiltr{l,i}=filter(filtroPB,cargaSand{l,i});
327             cargaSandFiltr{l,i}=cargaSandFiltr{l,i}(INICIO_FILTR:end); %quitamos las INICIO_FILTR muestas
328
329             cargaVenl{l,i}=nonzeros(cargaVenl{l,i});
330             cargaVenlFiltr{l,i}=filter(filtroPB,cargaVenl{l,i});
331             cargaVenlFiltr{l,i}=cargaVenlFiltr{l,i}(INICIO_FILTR:end); %quitamos las INICIO_FILTR muestas
332
333             cargaMicrOnYSandOn{l,i}=nonzeros(cargaMicrOnYSandOn{l,i});
334             cargaMicrOnYSandOnFiltr{l,i}=filter(filtroPB,cargaMicrOnYSandOn{l,i});
335             cargaMicrOnYSandOnFiltr{l,i}=cargaMicrOnYSandOnFiltr{l,i}(INICIO_FILTR:end); %quitamos las INICIO_FILTR muestas
336
337             cargaSandOnYBat3On{l,i}=nonzeros(cargaSandOnYBat3On{l,i});
338             cargaSandOnYBat3OnFiltr{l,i}=filter(filtroPB,cargaSandOnYBat3On{l,i});
339             cargaSandOnYBat3OnFiltr{l,i}=cargaSandOnYBat3OnFiltr{l,i}(INICIO_FILTR:end); %quitamos las INICIO_FILTR muestas
340
341             cargaVenlOnYBat3On{l,i}=nonzeros(cargaVenlOnYBat3On{l,i});
342             cargaVenlOnYBat3OnFiltr{l,i}=filter(filtroPB,cargaVenlOnYBat3On{l,i});
343             cargaVenlOnYBat3OnFiltr{l,i}=cargaVenlOnYBat3OnFiltr{l,i}(INICIO_FILTR:end); %quitamos las INICIO_FILTR muestas
344
345             cargaVenlOnYMicrOnYSandOn{l,i}=nonzeros(cargaVenlOnYMicrOnYSandOn{l,i});
346             cargaVenlOnYMicrOnYSandOnFiltr{l,i}=filter(filtroPB,cargaVenlOnYMicrOnYSandOn{l,i});
347             cargaVenlOnYMicrOnYSandOnFiltr{l,i}=cargaVenlOnYMicrOnYSandOnFiltr{l,i}(INICIO_FILTR:end); %quitamos las INICIO_FILTR muestas
348         end
349
350
351         %% EVENTO ON
352         %deteccion evento a on
353         %elegimos una ventana de un ciclo, le restamos la mediana de esa ventana(DC OFF)
354         %sumamos los valores absolutos y si son mayores que un threshold,
355         % => Evento ON
356         % en ventanaEvento se guarda el valor de la ventana corriespondiente al
357         % wakeup,si hay evento, tras el gradiente(ventana roja o verde)
358
359
360
361
362
363         onBatl=zeros(TRIALS,l);
364         onBatl=zeros(TRIALS,l);
365         vBatl={};
366         onBat2=zeros(TRIALS,l);
367         offBat2=zeros(TRIALS,l);
368         vBat2={};
369         onBat3=zeros(TRIALS,l);
370         offBat3=zeros(TRIALS,l);
371         vBat3={};
372         onExpr=zeros(TRIALS,l);
373         offExpr=zeros(TRIALS,l);
374         vExpr={};
375         onMicr=zeros(TRIALS,l);
376         offMicr=zeros(TRIALS,l);
377         vMicr={};
378         onSand=zeros(TRIALS,l);
379         offSand=zeros(TRIALS,l);
380         vSand={};
381         onVenl=zeros(TRIALS,l);
```

```
382         offVenI=zeros(TRIALS,1);
383         vVenI={};
384         onMicrOnYSandOn=zeros(TRIALS,1);
385         offMicrOnYSandOn=zeros(TRIALS,1);
386         vMicrOnYSandOn={};
387         onSandOnYBat3On=zeros(TRIALS,1);
388         offSandOnYBat3On=zeros(TRIALS,1);
389         vSandOnYBat3On={};
390         onVenIOnYBat3On=zeros(TRIALS,1);
391         offVenIOnYBat3On=zeros(TRIALS,1);
392         vVenIOnYBat3On={};
393         onVenIOnYMicrOnYSandOn=zeros(TRIALS,1);
394         offVenIOnYMicrOnYSandOn=zeros(TRIALS,1);
395         vVenIOnYMicrOnYSandOn={};
396
397         for i=1:TRIALS
398
399             cargaBatI{1,i}=nonzeros(cargaBatI{1,i})';
400             cargaBatI{1,i}=(cargaBatI{1,i})-median(cargaBatI{1,i});
401             j=1;
402             k=1;
403             m=1;
404             vTemp=[];
405             for indice=1:WAKE_UP:size(cargaBatI{1,i},2)-VENTANA_EVENTO;
406                 ventanaEvenBatI(i,j)=sum(abs(cargaBatI{1,i}(indice:indice+VENTANA_EVENTO-1)-median(cargaBatI{1,i}(indice:indice+VENTANA_EVENTO-1))));
407                 vTemp=[vTemp indice ];
408                 if j>1
409                     if ventanaEvenBatI(i,j)>ventanaEvenBatI(i,j-1)+THRESHOLD %hay evento
410                         ventanaTempAnt=ventanaEvenBatI(i,j);
411                         for n=1:round(WAKE_UP/VENTANA_EVENTO) %miramos las siguientes ventanas hasta una diferencia menor que GRADIENTE
412                             if indice+(n+1)*VENTANA_EVENTO>size(cargaBatI{1,i},2)-1
413                                 onBatI(i,k)=indice+(n-1)*VENTANA_EVENTO;
414                                 ventanaEvenBatI(i,j)=ventanaTempAct;
415                                 break;
416                             end
417                             if n==round(WAKE_UP/VENTANA_EVENTO)
418                                 onBatI(i,k)=indice+(n-1)*VENTANA_EVENTO;
419                                 ventanaEvenBatI(i,j)=ventanaTempAct;
420                                 break;
421                             end
422                             ventanaTempAct=sum(abs(cargaBatI{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1)-
423     median(cargaBatI{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1))));
424                             if abs(ventanaTempAnt-ventanaTempAct)<=GRADIENTE
425                                 onBatI(i,k)=indice+(n-1)*VENTANA_EVENTO;
426                                 ventanaEvenBatI(i,j)=ventanaTempAct;
427                                 break;
428                             else
429                                 ventanaTempAnt=ventanaTempAct;
430                             end
431                         end
432                         k=k+1;
433                     end
434                     if ventanaEvenBatI(i,j)<ventanaEvenBatI(i,j-1)-THRESHOLD
435                         ventanaTempAnt=ventanaEvenBatI(i,j);
436                         for n=1:round(WAKE_UP/VENTANA_EVENTO) %miramos las siguientes ventanas hasta una diferencia menor que GRADIENTE
437                             if indice+(n+1)*VENTANA_EVENTO>size(cargaBatI{1,i},2)-1
438                                 offBatI(i,m)=indice+(n-1)*VENTANA_EVENTO;
439                                 ventanaEvenBatI(i,j)=ventanaTempAct;
440                                 break;
441                             end
442                             if n==round(WAKE_UP/VENTANA_EVENTO)
443                                 offBatI(i,k)=indice+(n-1)*VENTANA_EVENTO;
444                                 ventanaEvenBatI(i,j)=ventanaTempAct;
445                                 break;
446                             end
```

```
                    ventanaTempAct=sum(abs(cargaBat1{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1)-
median(cargaBat1{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1))));
                    if abs(ventanaTempAnt-ventanaTempAct)<=GRADIENTE
                        offBat1(i,m)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenBat1(i,j)=ventanaTempAct;
                        break;
                    else
                        ventanaTempAnt=ventanaTempAct;
                    end
                end
                m=m+1;
            end
        end
        j=j+1;
    end;
    vBat1{i}=vTemp;

    cargaBat2{1,i}=nonzeros(cargaBat2{1,i})';
    cargaBat2{1,i}=(cargaBat2{1,i})-median(cargaBat2{1,i});
    j=1;
    k=1;
    m=1;
    vTemp=[];
    for indice=1:WAKE_UP:size(cargaBat2{1,i},2)-VENTANA_EVENTO;
        ventanaEvenBat2(i,j)=sum(abs(cargaBat2{1,i}(indice:indice+VENTANA_EVENTO-1)-median(cargaBat2{1,i}(indice:indice+VENTANA_EVENTO-1))));
        vTemp=[vTemp indice ];
        if j>1
            if ventanaEvenBat2(i,j)>ventanaEvenBat2(i,j-1)+THRESHOLD %hay evento
                ventanaTempAnt=ventanaEvenBat2(i,j);
                for n=1:round(WAKE_UP/VENTANA_EVENTO) %miramos las siguientes ventanas hasta una diferencia menor que GRADIENTE
                    if indice+(n+1)*VENTANA_EVENTO>size(cargaBat2{1,i},2)-1
                        onBat2(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenBat2(i,j)=ventanaTempAct;
                        break;
                    end
                    if n==round(WAKE_UP/VENTANA_EVENTO)
                        onBat2(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenBat2(i,j)=ventanaTempAct;
                        break;
                    end
                    ventanaTempAct=sum(abs(cargaBat2{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1)-
median(cargaBat2{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1))));
                    if abs(ventanaTempAnt-ventanaTempAct)<=GRADIENTE
                        onBat2(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenBat2(i,j)=ventanaTempAct;
                        break;
                    else
                        ventanaTempAnt=ventanaTempAct;
                    end
                end
                k=k+1;
            end
            if ventanaEvenBat2(i,j)<ventanaEvenBat2(i,j-1)-THRESHOLD
                ventanaTempAnt=ventanaEvenBat2(i,j);
                for n=1:round(WAKE_UP/VENTANA_EVENTO) %miramos las siguientes ventanas hasta una diferencia menor que GRADIENTE
                    if indice+(n+1)*VENTANA_EVENTO>size(cargaBat2{1,i},2)-1
                        offBat2(i,m)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenBat2(i,j)=ventanaTempAct;
                        break;
                    end
                    if n==round(WAKE_UP/VENTANA_EVENTO)
                        offBat2(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenBat2(i,j)=ventanaTempAct;
                        break;
                    end
```

```
                    ventanaTempAct=sum(abs(cargaBat2{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1)-
median(cargaBat2{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1))));
                    if abs(ventanaTempAnt-ventanaTempAct)<=GRADIENTE
                        offBat2(i,m)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenBat2(i,j)=ventanaTempAct;
                        break;
                    else
                        ventanaTempAnt=ventanaTempAct;
                    end
                end
                m=m+1;
            end
        end
        j=j+1;
    end;
    vBat2{i}=vTemp;

    cargaBat3{1,i}=nonzeros(cargaBat3{1,i})';
    cargaBat3{1,i}=(cargaBat3{1,i})-median(cargaBat3{1,i});
    j=1;
    k=1;
    m=1;
    vTemp=[];
    for indice=1:WAKE_UP:size(cargaBat3{1,i},2)-VENTANA_EVENTO;
        ventanaEvenBat3(i,j)=sum(abs(cargaBat3{1,i}(indice:indice+VENTANA_EVENTO-1)-median(cargaBat3{1,i}(indice:indice+VENTANA_EVENTO-1))));
        vTemp=[vTemp indice ];
        if j>1
            if ventanaEvenBat3(i,j)>ventanaEvenBat3(i,j-1)+THRESHOLD %hay evento
                ventanaTempAnt=ventanaEvenBat3(i,j);
                for n=1:round(WAKE_UP/VENTANA_EVENTO) %miramos las siguientes ventanas hasta una diferencia menor que GRADIENTE
                    if indice+(n+1)*VENTANA_EVENTO>size(cargaBat3{1,i},2)-1
                        onBat3(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenBat3(i,j)=ventanaTempAct;
                        break;
                    end
                    if n==round(WAKE_UP/VENTANA_EVENTO)
                        onBat3(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenBat3(i,j)=ventanaTempAct;
                        break;
                    end
                    ventanaTempAct=sum(abs(cargaBat3{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1)-
median(cargaBat3{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1))));
                    if abs(ventanaTempAnt-ventanaTempAct)<=GRADIENTE
                        onBat3(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenBat3(i,j)=ventanaTempAct;
                        break;
                    else
                        ventanaTempAnt=ventanaTempAct;
                    end
                end
                k=k+1;
            end
            if ventanaEvenBat3(i,j)<ventanaEvenBat3(i,j-1)-THRESHOLD
                ventanaTempAnt=ventanaEvenBat3(i,j);
                for n=1:round(WAKE_UP/VENTANA_EVENTO) %miramos las siguientes ventanas hasta una diferencia menor que GRADIENTE
                    if indice+(n+1)*VENTANA_EVENTO>size(cargaBat3{1,i},2)-1
                        offBat3(i,m)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenBat3(i,j)=ventanaTempAct;
                        break;
                    end
                    if n==round(WAKE_UP/VENTANA_EVENTO)
                        offBat3(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenBat3(i,j)=ventanaTempAct;
                        break;
                    end
```

```
                    ventanaTempAct=sum(abs(cargaBat3{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1)-
median(cargaBat3{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1))));
                    if abs(ventanaTempAnt-ventanaTempAct)<=GRADIENTE
                        offBat3(i,m)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenBat3(i,j)=ventanaTempAct;
                        break;
                    else
                        ventanaTempAnt=ventanaTempAct;
                    end
                end
                m=m+1;
            end
        end
        j=j+1;
    end;
    vBat3{i}=vTemp;

    cargaExpr{1,i}=nonzeros(cargaExpr{1,i})';
    cargaExpr{1,i}=(cargaExpr{1,i})-median(cargaExpr{1,i});
    j=1;
    k=1;
    m=1;
    vTemp=[];
    for indice=1:WAKE_UP:size(cargaExpr{1,i},2)-VENTANA_EVENTO;
        ventanaEvenExpr(i,j)=sum(abs(cargaExpr{1,i}(indice:indice+VENTANA_EVENTO-1)-median(cargaExpr{1,i}(indice:indice+VENTANA_EVENTO-1))));
        vTemp=[vTemp indice ];
        if j>1
            if ventanaEvenExpr(i,j)>ventanaEvenExpr(i,j-1)+THRESHOLD %hay evento
                ventanaTempAnt=ventanaEvenExpr(i,j);
                for n=1:round(WAKE_UP/VENTANA_EVENTO) %miramos las siguientes ventanas hasta una diferencia menor que GRADIENTE
                    if indice+(n+1)*VENTANA_EVENTO>size(cargaExpr{1,i},2)-1
                        onExpr(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenExpr(i,j)=ventanaTempAct;
                        break;
                    end
                    if n==round(WAKE_UP/VENTANA_EVENTO)
                        onExpr(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenExpr(i,j)=ventanaTempAct;
                        break;
                    end
                    ventanaTempAct=sum(abs(cargaExpr{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1)-
median(cargaExpr{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1))));
                    if abs(ventanaTempAnt-ventanaTempAct)<=GRADIENTE
                        onExpr(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenExpr(i,j)=ventanaTempAct;
                        break;
                    else
                        ventanaTempAnt=ventanaTempAct;
                    end
                end
                k=k+1;
            end
            if ventanaEvenExpr(i,j)<ventanaEvenExpr(i,j-1)-THRESHOLD
                ventanaTempAnt=ventanaEvenExpr(i,j);
                for n=1:round(WAKE_UP/VENTANA_EVENTO) %miramos las siguientes ventanas hasta una diferencia menor que GRADIENTE
                    if indice+(n+1)*VENTANA_EVENTO>size(cargaExpr{1,i},2)-1
                        offExpr(i,m)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenExpr(i,j)=ventanaTempAct;
                        break;
                    end
                    if n==round(WAKE_UP/VENTANA_EVENTO)
                        offExpr(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenExpr(i,j)=ventanaTempAct;
                        break;
                    end
```

```
                    ventanaTempAct=sum(abs(cargaExpr{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1)-
median(cargaExpr{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1))));
                    if abs(ventanaTempAnt-ventanaTempAct)<=GRADIENTE
                        offExpr(i,m)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenExpr(i,j)=ventanaTempAct;
                        break;
                    else
                        ventanaTempAnt=ventanaTempAct;
                    end
                end
                m=m+1;
            end
        end
        j=j+1;
    end;
    vExpr{i}=vTemp;

    cargaMicr{1,i}=nonzeros(cargaMicr{1,i})';
    cargaMicr{1,i}=(cargaMicr{1,i})-median(cargaMicr{1,i});
    j=1;
    k=1;
    m=1;
    vTemp=[];
    for indice=1:WAKE_UP:size(cargaMicr{1,i},2)-VENTANA_EVENTO;
        ventanaEvenMicr(i,j)=sum(abs(cargaMicr{1,i}(indice:indice+VENTANA_EVENTO-1)-median(cargaMicr{1,i}(indice:indice+VENTANA_EVENTO-1))));
        vTemp=[vTemp indice ];
        if j>1
            if ventanaEvenMicr(i,j)>ventanaEvenMicr(i,j-1)+THRESHOLD %hay evento
                ventanaTempAnt=ventanaEvenMicr(i,j);
                for n=1:round(WAKE_UP/VENTANA_EVENTO) %miramos las siguientes ventanas hasta una diferencia menor que GRADIENTE
                    if indice+(n+1)*VENTANA_EVENTO>size(cargaMicr{1,i},2)-1
                        onMicr(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenMicr(i,j)=ventanaTempAct;
                        break;
                    end
                    if n==round(WAKE_UP/VENTANA_EVENTO)
                        onMicr(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenMicr(i,j)=ventanaTempAct;
                        break;
                    end
                    ventanaTempAct=sum(abs(cargaMicr{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1)-
median(cargaMicr{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1))));
                    if abs(ventanaTempAnt-ventanaTempAct)<=GRADIENTE
                        onMicr(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenMicr(i,j)=ventanaTempAct;
                        break;
                    else
                        ventanaTempAnt=ventanaTempAct;
                    end
                end
                k=k+1;
            end
            if ventanaEvenMicr(i,j)<ventanaEvenMicr(i,j-1)-THRESHOLD
                ventanaTempAnt=ventanaEvenMicr(i,j);
                for n=1:round(WAKE_UP/VENTANA_EVENTO) %miramos las siguientes ventanas hasta una diferencia menor que GRADIENTE
                    if indice+(n+1)*VENTANA_EVENTO>size(cargaMicr{1,i},2)-1
                        offMicr(i,m)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenMicr(i,j)=ventanaTempAct;
                        break;
                    end
                    if n==round(WAKE_UP/VENTANA_EVENTO)
                        offMicr(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenMicr(i,j)=ventanaTempAct;
                        break;
                    end
```

```
                    ventanaTempAct=sum(abs(cargaMicr{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1)-
median(cargaMicr{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1))));
                    if abs(ventanaTempAnt-ventanaTempAct)<=GRADIENTE
                        offMicr(i,m)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenMicr(i,j)=ventanaTempAct;
                        break;
                    else
                        ventanaTempAnt=ventanaTempAct;
                    end
                end
                m=m+1;
            end
        end
        j=j+1;
    end;
    vMicr{i}=vTemp;

    cargaSand{1,i}=nonzeros(cargaSand{1,i})';
    cargaSand{1,i}=(cargaSand{1,i})-median(cargaSand{1,i});
    j=1;
    k=1;
    m=1;
    vTemp=[];
    for indice=1:WAKE_UP:size(cargaSand{1,i},2)-VENTANA_EVENTO;
        ventanaEvenSand(i,j)=sum(abs(cargaSand{1,i}(indice:indice+VENTANA_EVENTO-1)-median(cargaSand{1,i}(indice:indice+VENTANA_EVENTO-1))));
        vTemp=[vTemp indice ];
        if j>1
            if ventanaEvenSand(i,j)>ventanaEvenSand(i,j-1)+THRESHOLD %hay evento
                ventanaTempAnt=ventanaEvenSand(i,j);
                for n=1:round(WAKE_UP/VENTANA_EVENTO) %miramos las siguientes ventanas hasta una diferencia menor que GRADIENTE
                    if indice+(n+1)*VENTANA_EVENTO>size(cargaSand{1,i},2)-1
                        onSand(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenSand(i,j)=ventanaTempAct;
                        break;
                    end
                    if n==round(WAKE_UP/VENTANA_EVENTO)
                        onSand(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenSand(i,j)=ventanaTempAct;
                        break;
                    end
                    ventanaTempAct=sum(abs(cargaSand{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1)-
median(cargaSand{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1))));
                    if abs(ventanaTempAnt-ventanaTempAct)<=GRADIENTE
                        onSand(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenSand(i,j)=ventanaTempAct;
                        break;
                    else
                        ventanaTempAnt=ventanaTempAct;
                    end
                end
                k=k+1;
            end
            if ventanaEvenSand(i,j)<ventanaEvenSand(i,j-1)-THRESHOLD
                ventanaTempAnt=ventanaEvenSand(i,j);
                for n=1:round(WAKE_UP/VENTANA_EVENTO) %miramos las siguientes ventanas hasta una diferencia menor que GRADIENTE
                    if indice+(n+1)*VENTANA_EVENTO>size(cargaSand{1,i},2)-1
                        offSand(i,m)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenSand(i,j)=ventanaTempAct;
                        break;
                    end
                    if n==round(WAKE_UP/VENTANA_EVENTO)
                        offSand(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenSand(i,j)=ventanaTempAct;
                        break;
                    end
```

```
                    ventanaTempAct=sum(abs(cargaSand{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1)-
median(cargaSand{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1))));
                    if abs(ventanaTempAnt-ventanaTempAct)<=GRADIENTE
                        offSand(i,m)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenSand(i,j)=ventanaTempAct;
                        break;
                    else
                        ventanaTempAnt=ventanaTempAct;
                    end
                end
                m=m+1;
            end
        end
        j=j+1;
    end;
    vSand{i}=vTemp;

    cargaVenI{1,i}=nonzeros(cargaVenI{1,i})';
    cargaVenI{1,i}=(cargaVenI{1,i})-median(cargaVenI{1,i});
    j=1;
    k=1;
    m=1;
    vTemp=[];
    for indice=1:WAKE_UP:size(cargaVenI{1,i},2)-VENTANA_EVENTO;
        ventanaEvenVenI(i,j)=sum(abs(cargaVenI{1,i}(indice:indice+VENTANA_EVENTO-1)-median(cargaVenI{1,i}(indice:indice+VENTANA_EVENTO-1))));
        vTemp=[vTemp indice ];
        if j>1
            if ventanaEvenVenI(i,j)>ventanaEvenVenI(i,j-1)+THRESHOLD %hay evento
                ventanaTempAnt=ventanaEvenVenI(i,j);
                for n=1:round(WAKE_UP/VENTANA_EVENTO) %miramos las siguientes ventanas hasta una diferencia menor que GRADIENTE
                    if indice+(n+1)*VENTANA_EVENTO>size(cargaVenI{1,i},2)-1
                        onVenI(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenVenI(i,j)=ventanaTempAct;
                        break;
                    end
                    if n==round(WAKE_UP/VENTANA_EVENTO)
                        onVenI(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenVenI(i,j)=ventanaTempAct;
                        break;
                    end
                    ventanaTempAct=sum(abs(cargaVenI{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1)-
median(cargaVenI{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1))));
                    if abs(ventanaTempAnt-ventanaTempAct)<=GRADIENTE
                        onVenI(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenVenI(i,j)=ventanaTempAct;
                        break;
                    else
                        ventanaTempAnt=ventanaTempAct;
                    end
                end
                k=k+1;
            end
            if ventanaEvenVenI(i,j)<ventanaEvenVenI(i,j-1)-THRESHOLD
                ventanaTempAnt=ventanaEvenVenI(i,j);
                for n=1:round(WAKE_UP/VENTANA_EVENTO) %miramos las siguientes ventanas hasta una diferencia menor que GRADIENTE
                    if indice+(n+1)*VENTANA_EVENTO>size(cargaVenI{1,i},2)-1
                        offVenI(i,m)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenVenI(i,j)=ventanaTempAct;
                        break;
                    end
                    if n==round(WAKE_UP/VENTANA_EVENTO)
                        offVenI(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenVenI(i,j)=ventanaTempAct;
                        break;
                    end
```

```
                    ventanaTempAct=sum(abs(cargaVenI{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1)-
median(cargaVenI{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1))));
                    if abs(ventanaTempAnt-ventanaTempAct)<=GRADIENTE
                        offVenI(i,m)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenVenI(i,j)=ventanaTempAct;
                        break;
                    else
                        ventanaTempAnt=ventanaTempAct;
                    end
                end
                m=m+1;
            end
        end
        j=j+1;
    end;
    vVenI{i}=vTemp;

    cargaMicrOnYSandOn{1,i}=nonzeros(cargaMicrOnYSandOn{1,i})';
    cargaMicrOnYSandOn{1,i}=(cargaMicrOnYSandOn{1,i})-median(cargaMicrOnYSandOn{1,i});
    j=1;
    k=1;
    m=1;
    vTemp=[];
    for indice=1:WAKE_UP:size(cargaMicrOnYSandOn{1,i},2)-VENTANA_EVENTO;
        ventanaEvenMicrOnYSandOn(i,j)=sum(abs(cargaMicrOnYSandOn{1,i}(indice:indice+VENTANA_EVENTO-1)-
median(cargaMicrOnYSandOn{1,i}(indice:indice+VENTANA_EVENTO-1))));
        vTemp=[vTemp indice ];
        if j>1
            if ventanaEvenMicrOnYSandOn(i,j)>ventanaEvenMicrOnYSandOn(i,j-1)+THRESHOLD %hay evento
                ventanaTempAnt=ventanaEvenMicrOnYSandOn(i,j);
                for n=1:round(WAKE_UP/VENTANA_EVENTO) %miramos las siguientes ventanas hasta una diferencia menor que GRADIENTE
                    if indice+(n+1)*VENTANA_EVENTO>size(cargaMicrOnYSandOn{1,i},2)-1
                        onMicrOnYSandOn(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenMicrOnYSandOn(i,j)=ventanaTempAct;
                        break;
                    end
                    if n==round(WAKE_UP/VENTANA_EVENTO)
                        onMicrOnYSandOn(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenMicrOnYSandOn(i,j)=ventanaTempAct;
                        break;
                    end
                    ventanaTempAct=sum(abs(cargaMicrOnYSandOn{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1)-
median(cargaMicrOnYSandOn{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1))));
                    if abs(ventanaTempAnt-ventanaTempAct)<=GRADIENTE
                        onMicrOnYSandOn(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenMicrOnYSandOn(i,j)=ventanaTempAct;
                        break;
                    else
                        ventanaTempAnt=ventanaTempAct;
                    end
                end
                k=k+1;
            end
            if ventanaEvenMicrOnYSandOn(i,j)<ventanaEvenMicrOnYSandOn(i,j-1)-THRESHOLD
                ventanaTempAnt=ventanaEvenMicrOnYSandOn(i,j);
                for n=1:round(WAKE_UP/VENTANA_EVENTO) %miramos las siguientes ventanas hasta una diferencia menor que GRADIENTE
                    if indice+(n+1)*VENTANA_EVENTO>size(cargaMicrOnYSandOn{1,i},2)-1
                        offMicrOnYSandOn(i,m)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenMicrOnYSandOn(i,j)=ventanaTempAct;
                        break;
                    end
                    if n==round(WAKE_UP/VENTANA_EVENTO)
                        offMicrOnYSandOn(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenMicrOnYSandOn(i,j)=ventanaTempAct;
                        break;
```

```
                    end
                    ventanaTempAct=sum(abs(cargaMicrOnYSandOn{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1)-
median(cargaMicrOnYSandOn{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1))));
                    if abs(ventanaTempAnt-ventanaTempAct)<=GRADIENTE
                        offMicrOnYSandOn(i,m)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenMicrOnYSandOn(i,j)=ventanaTempAct;
                        break;
                    else
                        ventanaTempAnt=ventanaTempAct;
                    end
                end
                m=m+1;
            end
        end
        j=j+1;
    end;
    vMicrOnYSandOn{i}=vTemp;

    cargaSandOnYBat3On{1,i}=nonzeros(cargaSandOnYBat3On{1,i})';
    cargaSandOnYBat3On{1,i}=(cargaSandOnYBat3On{1,i})-median(cargaSandOnYBat3On{1,i});
    j=1;
    k=1;
    m=1;
    vTemp=[];
    for indice=1:WAKE_UP:size(cargaSandOnYBat3On{1,i},2)-VENTANA_EVENTO;
        ventanaEvenSandOnYBat3On(i,j)=sum(abs(cargaSandOnYBat3On{1,i}(indice:indice+VENTANA_EVENTO-1)-
median(cargaSandOnYBat3On{1,i}(indice:indice+VENTANA_EVENTO-1))));
        vTemp=[vTemp indice ];
        if j>1
            if ventanaEvenSandOnYBat3On(i,j)>ventanaEvenSandOnYBat3On(i,j-1)+THRESHOLD %hay evento
                ventanaTempAnt=ventanaEvenSandOnYBat3On(i,j);
                for n=1:round(WAKE_UP/VENTANA_EVENTO) %miramos las siguientes ventanas hasta una diferencia menor que GRADIENTE
                    if indice+(n+1)*VENTANA_EVENTO>size(cargaSandOnYBat3On{1,i},2)-1
                        onSandOnYBat3On(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenSandOnYBat3On(i,j)=ventanaTempAct;
                        break;
                    end
                    if n==round(WAKE_UP/VENTANA_EVENTO)
                        onSandOnYBat3On(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenSandOnYBat3On(i,j)=ventanaTempAct;
                        break;
                    end
                    ventanaTempAct=sum(abs(cargaSandOnYBat3On{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1)-
median(cargaSandOnYBat3On{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1))));
                    if abs(ventanaTempAnt-ventanaTempAct)<=GRADIENTE
                        onSandOnYBat3On(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenSandOnYBat3On(i,j)=ventanaTempAct;
                        break;
                    else
                        ventanaTempAnt=ventanaTempAct;
                    end
                end
                k=k+1;
            end
            if ventanaEvenSandOnYBat3On(i,j)<ventanaEvenSandOnYBat3On(i,j-1)-THRESHOLD
                ventanaTempAnt=ventanaEvenSandOnYBat3On(i,j);
                for n=1:round(WAKE_UP/VENTANA_EVENTO) %miramos las siguientes ventanas hasta una diferencia menor que GRADIENTE
                    if indice+(n+1)*VENTANA_EVENTO>size(cargaSandOnYBat3On{1,i},2)-1
                        offSandOnYBat3On(i,m)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenSandOnYBat3On(i,j)=ventanaTempAct;
                        break;
                    end
                    if n==round(WAKE_UP/VENTANA_EVENTO)
                        offSandOnYBat3On(i,k)=indice+(n-1)*VENTANA_EVENTO;
                        ventanaEvenSandOnYBat3On(i,j)=ventanaTempAct;
```

```
                    break;
                end
                ventanaTempAct=sum(abs(cargaSandOnYBat30n{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1)-
median(cargaSandOnYBat30n{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1))));
                if abs(ventanaTempAnt-ventanaTempAct)<=GRADIENTE
                    offSandOnYBat30n(i,m)=indice+(n-1)*VENTANA_EVENTO;
                    ventanaEvenSandOnYBat30n(i,j)=ventanaTempAct;
                    break;
                else
                    ventanaTempAnt=ventanaTempAct;
                end
            end
            m=m+1;
        end
    end
    j=j+1;
end;
vSandOnYBat30n{i}=vTemp;

cargaVenlOnYBat30n{1,i}=nonzeros(cargaVenlOnYBat30n{1,i})';
cargaVenlOnYBat30n{1,i}=(cargaVenlOnYBat30n{1,i})-median(cargaVenlOnYBat30n{1,i});
j=1;
k=1;
m=1;
vTemp=[];
for indice=1:WAKE_UP:size(cargaVenlOnYBat30n{1,i},2)-VENTANA_EVENTO;
    ventanaEvenVenlOnYBat30n(i,j)=sum(abs(cargaVenlOnYBat30n{1,i}(indice:indice+VENTANA_EVENTO-1)-
median(cargaVenlOnYBat30n{1,i}(indice:indice+VENTANA_EVENTO-1))));
    vTemp=[vTemp indice ];
    if j>1
        if ventanaEvenVenlOnYBat30n(i,j)>ventanaEvenVenlOnYBat30n(i,j-1)+THRESHOLD %hay evento
            ventanaTempAnt=ventanaEvenVenlOnYBat30n(i,j);
            for n=1:round(WAKE_UP/VENTANA_EVENTO) %miramos las siguientes ventanas hasta una diferencia menor que GRADIENTE
                if indice+(n+1)*VENTANA_EVENTO>size(cargaVenlOnYBat30n{1,i},2)-1
                    onVenlOnYBat30n(i,k)=indice+(n-1)*VENTANA_EVENTO;
                    ventanaEvenVenlOnYBat30n(i,j)=ventanaTempAct;
                    break;
                end
                if n==round(WAKE_UP/VENTANA_EVENTO)
                    onVenlOnYBat30n(i,k)=indice+(n-1)*VENTANA_EVENTO;
                    ventanaEvenVenlOnYBat30n(i,j)=ventanaTempAct;
                    break;
                end
                ventanaTempAct=sum(abs(cargaVenlOnYBat30n{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1)-
median(cargaVenlOnYBat30n{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1))));
                if abs(ventanaTempAnt-ventanaTempAct)<=GRADIENTE
                    onVenlOnYBat30n(i,k)=indice+(n-1)*VENTANA_EVENTO;
                    ventanaEvenVenlOnYBat30n(i,j)=ventanaTempAct;
                    break;
                else
                    ventanaTempAnt=ventanaTempAct;
                end
            end
            k=k+1;
        end
        if ventanaEvenVenlOnYBat30n(i,j)<ventanaEvenVenlOnYBat30n(i,j-1)-THRESHOLD
            ventanaTempAnt=ventanaEvenVenlOnYBat30n(i,j);
            for n=1:round(WAKE_UP/VENTANA_EVENTO) %miramos las siguientes ventanas hasta una diferencia menor que GRADIENTE
                if indice+(n+1)*VENTANA_EVENTO>size(cargaVenlOnYBat30n{1,i},2)-1
                    offVenlOnYBat30n(i,m)=indice+(n-1)*VENTANA_EVENTO;
                    ventanaEvenVenlOnYBat30n(i,j)=ventanaTempAct;
                    break;
                end
                if n==round(WAKE_UP/VENTANA_EVENTO)
                    offVenlOnYBat30n(i,k)=indice+(n-1)*VENTANA_EVENTO;
```

```
                    ventanaEvenVenIOnYBat3On(i,j)=ventanaTempAct;
                    break;
                end
                ventanaTempAct=sum(abs(cargaVenIOnYBat3On{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1)-
median(cargaVenIOnYBat3On{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1))));
                if abs(ventanaTempAnt-ventanaTempAct)<=GRADIENTE
                    offVenIOnYBat3On(i,m)=indice+(n-1)*VENTANA_EVENTO;
                    ventanaEvenVenIOnYBat3On(i,j)=ventanaTempAct;
                    break;
                else
                    ventanaTempAnt=ventanaTempAct;
                end
            end
            m=m+1;
        end
    end
    j=j+1;
end;
vVenIOnYBat3On{i}=vTemp;

cargaVenIOnYMicrOnYSandOn{1,i}=nonzeros(cargaVenIOnYMicrOnYSandOn{1,i})';
cargaVenIOnYMicrOnYSandOn{1,i}=(cargaVenIOnYMicrOnYSandOn{1,i})-median(cargaVenIOnYMicrOnYSandOn{1,i});
j=1;
k=1;
m=1;
vTemp=[];
for indice=1:WAKE_UP:size(cargaVenIOnYMicrOnYSandOn{1,i},2)-VENTANA_EVENTO;
    ventanaEvenVenIOnYMicrOnYSandOn(i,j)=sum(abs(cargaVenIOnYMicrOnYSandOn{1,i}(indice:indice+VENTANA_EVENTO-1)-
median(cargaVenIOnYMicrOnYSandOn{1,i}(indice:indice+VENTANA_EVENTO-1))));
    vTemp=[vTemp indice ];
    if j>1
        if ventanaEvenVenIOnYMicrOnYSandOn(i,j)>ventanaEvenVenIOnYMicrOnYSandOn(i,j-1)+THRESHOLD %hay evento
            ventanaTempAnt=ventanaEvenVenIOnYMicrOnYSandOn(i,j);
            for n=1:round(WAKE_UP/VENTANA_EVENTO) %miramos las siguientes ventanas hasta una diferencia menor que GRADIENTE
                if indice+(n+1)*VENTANA_EVENTO>size(cargaVenIOnYMicrOnYSandOn{1,i},2)-1
                    onVenIOnYMicrOnYSandOn(i,k)=indice+(n-1)*VENTANA_EVENTO;
                    ventanaEvenVenIOnYMicrOnYSandOn(i,j)=ventanaTempAct;
                    break;
                end
                if n==round(WAKE_UP/VENTANA_EVENTO)
                    onVenIOnYMicrOnYSandOn(i,k)=indice+(n-1)*VENTANA_EVENTO;
                    ventanaEvenVenIOnYMicrOnYSandOn(i,j)=ventanaTempAct;
                    break;
                end
                ventanaTempAct=sum(abs(cargaVenIOnYMicrOnYSandOn{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1)-
median(cargaVenIOnYMicrOnYSandOn{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1))));
                if abs(ventanaTempAnt-ventanaTempAct)<=GRADIENTE
                    onVenIOnYMicrOnYSandOn(i,k)=indice+(n-1)*VENTANA_EVENTO;
                    ventanaEvenVenIOnYMicrOnYSandOn(i,j)=ventanaTempAct;
                    break;
                else
                    ventanaTempAnt=ventanaTempAct;
                end
            end
            k=k+1;
        end
        if ventanaEvenVenIOnYMicrOnYSandOn(i,j)<ventanaEvenVenIOnYMicrOnYSandOn(i,j-1)-THRESHOLD
            ventanaTempAnt=ventanaEvenVenIOnYMicrOnYSandOn(i,j);
            for n=1:round(WAKE_UP/VENTANA_EVENTO) %miramos las siguientes ventanas hasta una diferencia menor que GRADIENTE
                if indice+(n+1)*VENTANA_EVENTO>size(cargaVenIOnYMicrOnYSandOn{1,i},2)-1
                    offVenIOnYMicrOnYSandOn(i,m)=indice+(n-1)*VENTANA_EVENTO;
                    ventanaEvenVenIOnYMicrOnYSandOn(i,j)=ventanaTempAct;
                    break;
                end
                if n==round(WAKE_UP/VENTANA_EVENTO)
```

```
                    offVentOnYMicrOnYSandOn(i,k)=indice+(n-1)*VENTANA_EVENTO;
                    ventanaEvenVentOnYMicrOnYSandOn(i,j)=ventanaTempAct;
                    break;
                end
                ventanaTempAct=sum(abs(cargaVentOnYMicrOnYSandOn{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1)-
median(cargaVentOnYMicrOnYSandOn{1,i}(indice+n*VENTANA_EVENTO:indice+(n+1)*VENTANA_EVENTO-1))));
                if abs(ventanaTempAnt-ventanaTempAct)<=GRADIENTE
                    offVentOnYMicrOnYSandOn(i,m)=indice+(n-1)*VENTANA_EVENTO;
                    ventanaEvenVentOnYMicrOnYSandOn(i,j)=ventanaTempAct;
                    break;
                else
                    ventanaTempAnt=ventanaTempAct;
                end
            end
            m=m+1;
        end
    end
    j=j+1;
end;
vVentOnYMicrOnYSandOn{i}=vTemp;

end;
clear ventana  ventanaTempAnt ventanaTempAct j k m n ;


%% VER EVENTOS
for z=1:TRIALS
    figure;
    hold;
    %Marca Ventana
    X=ones(1,2)*max(cargaBatt{1,z});
    Y=ones(1,2)*max(cargaBatt{1,z});
    for x=1:size(vBatt{z},2)
        h=area([vBatt{z}(x) vBatt{z}(x)+VENTANA_EVENTO],[max(cargaBatt{1,z}) max(cargaBatt{1,z})],...
            'FaceColor','y');
        set(h,'BaseValue',min(cargaBatt{1,z}))
    end
    %Marca On
    for x=1:size(onBatt,2)
        %h=area([onBatt(z,x) onBatt(z,x)+VENTANA_EVENTO],[max(cargaBatt{1,z}) max(cargaBatt{1,z})],...
        if onBatt(z,x) %si es distinto de cero se pinta
            h=area([onBatt(z,x) onBatt(z,x)+VENTANA_EVENTO],X,...
                'FaceColor','g');
            set(h,'BaseValue',min(cargaBatt{1,z}))
        end
    end
    %Marca Off
    for x=1:size(offBatt,2)
        %h=area([offBatt(z,x) offBatt(z,x)+VENTANA_EVENTO],[max(cargaBatt{1,z}) max(cargaBatt{1,z})],...
        if  offBatt(z,x)
            h=area([offBatt(z,x) offBatt(z,x)+VENTANA_EVENTO],Y,...
                'FaceColor','r');
            set(h,'BaseValue',min(cargaBatt{1,z}))
        end
    end
    plot(cargaBatt{1,z},'k')
    title('cargaBatt')

end
pause;
close all;

for z=1:TRIALS
    figure;
    hold;
```

```
1162            %Marca Ventana
1163            X=ones(1,2)*max(cargaBat2{1,z});
1164            Y=ones(1,2)*max(cargaBat2{1,z});
1165            for x=1:size(vBat2{z},2)
1166               h=area([vBat2{z}(x) vBat2{z}(x)+VENTANA_EVENTO],[max(cargaBat2{1,z}) max(cargaBat2{1,z})],...
1167                     'FaceColor','y');
1168               set(h,'BaseValue',min(cargaBat2{1,z}))
1169            end
1170            %Marca On
1171            for x=1:size(onBat2,2)
1172               %h=area([onBat2(z,x) onBat2(z,x)+VENTANA_EVENTO],[max(cargaBat2{1,z}) max(cargaBat2{1,z})],...
1173               if onBat2(z,x) %si es distinto de cero se pinta
1174                  h=area([onBat2(z,x) onBat2(z,x)+VENTANA_EVENTO],X,...
1175                     'FaceColor','g');
1176                  set(h,'BaseValue',min(cargaBat2{1,z}))
1177               end
1178            end
1179            %Marca Off
1180            for x=1:size(offBat2,2)
1181               %h=area([offBat2(z,x) offBat2(z,x)+VENTANA_EVENTO],[max(cargaBat2{1,z}) max(cargaBat2{1,z})],...
1182               if  offBat2(z,x)
1183                  h=area([offBat2(z,x) offBat2(z,x)+VENTANA_EVENTO],Y,...
1184                        'FaceColor','r');
1185                  set(h,'BaseValue',min(cargaBat2{1,z}))
1186               end
1187            end
1188            plot(cargaBat2{1,z},'k')
1189            title('cargaBat2')
1190
1191         end
1192         pause;
1193         close all;
1194
1195         for z=1:TRIALS
1196            figure;
1197            hold;
1198            %Marca Ventana
1199            X=ones(1,2)*max(cargaBat3{1,z});
1200            Y=ones(1,2)*max(cargaBat3{1,z});
1201            for x=1:size(vBat3{z},2)
1202               h=area([vBat3{z}(x) vBat3{z}(x)+VENTANA_EVENTO],[max(cargaBat3{1,z}) max(cargaBat3{1,z})],...
1203                     'FaceColor','y');
1204               set(h,'BaseValue',min(cargaBat3{1,z}))
1205            end
1206            %Marca On
1207            for x=1:size(onBat3,2)
1208               %h=area([onBat3(z,x) onBat3(z,x)+VENTANA_EVENTO],[max(cargaBat3{1,z}) max(cargaBat3{1,z})],...
1209               if onBat3(z,x) %si es distinto de cero se pinta
1210                  h=area([onBat3(z,x) onBat3(z,x)+VENTANA_EVENTO],X,...
1211                     'FaceColor','g');
1212                  set(h,'BaseValue',min(cargaBat3{1,z}))
1213               end
1214            end
1215            %Marca Off
1216            for x=1:size(offBat3,2)
1217               %h=area([offBat3(z,x) offBat3(z,x)+VENTANA_EVENTO],[max(cargaBat3{1,z}) max(cargaBat3{1,z})],...
1218               if  offBat3(z,x)
1219                  h=area([offBat3(z,x) offBat3(z,x)+VENTANA_EVENTO],Y,...
1220                        'FaceColor','r');
1221                  set(h,'BaseValue',min(cargaBat3{1,z}))
1222               end
1223            end
1224            plot(cargaBat3{1,z},'k')
1225            title('cargaBat3')
1226
```

```
1227        end
1228        pause;
1229        close all;
1230
1231        for z=1:TRIALS
1232           figure;
1233           hold;
1234           %Marca Ventana
1235           X=ones(1,2)*max(cargaExpr{1,z});
1236           Y=ones(1,2)*max(cargaExpr{1,z});
1237           for x=1:size(vExpr{z},2)
1238              h=area([vExpr{z}(x) vExpr{z}(x)+VENTANA_EVENTO],[max(cargaExpr{1,z}) max(cargaExpr{1,z})],...
1239                    'FaceColor','y');
1240              set(h,'BaseValue',min(cargaExpr{1,z}))
1241           end
1242           %Marca On
1243           for x=1:size(onExpr,2)
1244              %h=area([onExpr(z,x) onExpr(z,x)+VENTANA_EVENTO],[max(cargaExpr{1,z}) max(cargaExpr{1,z})],...
1245              if onExpr(z,x) %si es distinto de cero se pinta
1246                 h=area([onExpr(z,x) onExpr(z,x)+VENTANA_EVENTO],X,...
1247                    'FaceColor','g');
1248                 set(h,'BaseValue',min(cargaExpr{1,z}))
1249              end
1250           end
1251           %Marca Off
1252           for x=1:size(offExpr,2)
1253              %h=area([offExpr(z,x) offExpr(z,x)+VENTANA_EVENTO],[max(cargaExpr{1,z}) max(cargaExpr{1,z})],...
1254              if  offExpr(z,x)
1255                 h=area([offExpr(z,x) offExpr(z,x)+VENTANA_EVENTO],Y,...
1256                       'FaceColor','r');
1257                 set(h,'BaseValue',min(cargaExpr{1,z}))
1258              end
1259           end
1260           plot(cargaExpr{1,z},'k')
1261           title('cargaExpr')
1262
1263        end
1264        pause;
1265        close all;
1266
1267        for z=1:TRIALS
1268           figure;
1269           hold;
1270           %Marca Ventana
1271           X=ones(1,2)*max(cargaMicr{1,z});
1272           Y=ones(1,2)*max(cargaMicr{1,z});
1273           for x=1:size(vMicr{z},2)
1274              h=area([vMicr{z}(x) vMicr{z}(x)+VENTANA_EVENTO],[max(cargaMicr{1,z}) max(cargaMicr{1,z})],...
1275                    'FaceColor','y');
1276              set(h,'BaseValue',min(cargaMicr{1,z}))
1277           end
1278           %Marca On
1279           for x=1:size(onMicr,2)
1280              %h=area([onMicr(z,x) onMicr(z,x)+VENTANA_EVENTO],[max(cargaMicr{1,z}) max(cargaMicr{1,z})],...
1281              if onMicr(z,x) %si es distinto de cero se pinta
1282                 h=area([onMicr(z,x) onMicr(z,x)+VENTANA_EVENTO],X,...
1283                    'FaceColor','g');
1284                 set(h,'BaseValue',min(cargaMicr{1,z}))
1285              end
1286           end
1287           %Marca Off
1288           for x=1:size(offMicr,2)
1289              %h=area([offMicr(z,x) offMicr(z,x)+VENTANA_EVENTO],[max(cargaMicr{1,z}) max(cargaMicr{1,z})],...
1290              if  offMicr(z,x)
1291                 h=area([offMicr(z,x) offMicr(z,x)+VENTANA_EVENTO],Y,...
```

```
                    'FaceColor','r');
                set(h,'BaseValue',min(cargaMicr{1,z}))
            end
        end
        plot(cargaMicr{1,z},'k')
        title('cargaMicr')

    end
    pause;
    close all;

    for z=1:TRIALS
        figure;
        hold;
        %Marca Ventana
        X=ones(1,2)*max(cargaSand{1,z});
        Y=ones(1,2)*max(cargaSand{1,z});
        for x=1:size(vSand{z},2)
            h=area([vSand{z}(x) vSand{z}(x)+VENTANA_EVENTO],[max(cargaSand{1,z}) max(cargaSand{1,z})],...
                'FaceColor','y');
            set(h,'BaseValue',min(cargaSand{1,z}))
        end
        %Marca On
        for x=1:size(onSand,2)
            %h=area([onSand(z,x) onSand(z,x)+VENTANA_EVENTO],[max(cargaSand{1,z}) max(cargaSand{1,z})],...
            if onSand(z,x) %si es distinto de cero se pinta
                h=area([onSand(z,x) onSand(z,x)+VENTANA_EVENTO],X,...
                'FaceColor','g');
                set(h,'BaseValue',min(cargaSand{1,z}))
            end
        end
        %Marca Off
        for x=1:size(offSand,2)
            %h=area([offSand(z,x) offSand(z,x)+VENTANA_EVENTO],[max(cargaSand{1,z}) max(cargaSand{1,z})],...
            if  offSand(z,x)
                h=area([offSand(z,x) offSand(z,x)+VENTANA_EVENTO],Y,...
                    'FaceColor','r');
                set(h,'BaseValue',min(cargaSand{1,z}))
            end
        end
        plot(cargaSand{1,z},'k')
        title('cargaSand')

    end
    pause;
    close all;

    for z=1:TRIALS
        figure;
        hold;
        %Marca Ventana
        X=ones(1,2)*max(cargaVenI{1,z});
        Y=ones(1,2)*max(cargaVenI{1,z});
        for x=1:size(vVenI{z},2)
            h=area([vVenI{z}(x) vVenI{z}(x)+VENTANA_EVENTO],[max(cargaVenI{1,z}) max(cargaVenI{1,z})],...
                'FaceColor','y');
            set(h,'BaseValue',min(cargaVenI{1,z}))
        end
        %Marca On
        for x=1:size(onVenI,2)
            %h=area([onVenI(z,x) onVenI(z,x)+VENTANA_EVENTO],[max(cargaVenI{1,z}) max(cargaVenI{1,z})],...
            if onVenI(z,x) %si es distinto de cero se pinta
                h=area([onVenI(z,x) onVenI(z,x)+VENTANA_EVENTO],X,...
                'FaceColor','g');
                set(h,'BaseValue',min(cargaVenI{1,z}))
```

```
            end
        end
        %Marca Off
        for x=1:size(offVent,2)
            %h=area([offVent(z,x) offVent(z,x)+VENTANA_EVENTO],[max(cargaVent{1,z}) max(cargaVent{1,z})],...
            if  offVent(z,x)
                h=area([offVent(z,x) offVent(z,x)+VENTANA_EVENTO],Y,...
                    'FaceColor','r');
                set(h,'BaseValue',min(cargaVent{1,z}))
            end
        end
        plot(cargaVent{1,z},'k')
        title('cargaVent')

    end
    pause;
    close all;

    for z=1:TRIALS
        figure;
        hold;
        %Marca Ventana
        X=ones(1,2)*max(cargaMicrOnYSandOn{1,z});
        Y=ones(1,2)*max(cargaMicrOnYSandOn{1,z});
        for x=1:size(vMicrOnYSandOn{z},2)
            h=area([vMicrOnYSandOn{z}(x) vMicrOnYSandOn{z}(x)+VENTANA_EVENTO],[max(cargaMicrOnYSandOn{1,z}) max(cargaMicrOnYSandOn{1,z})],...
                'FaceColor','y');
            set(h,'BaseValue',min(cargaMicrOnYSandOn{1,z}))
        end
        %Marca On
        for x=1:size(onMicrOnYSandOn,2)
            %h=area([onMicrOnYSandOn(z,x) onMicrOnYSandOn(z,x)+VENTANA_EVENTO],[max(cargaMicrOnYSandOn{1,z}) max(cargaMicrOnYSandOn{1,z})],...
            if onMicrOnYSandOn(z,x) %si es distinto de cero se pinta
                h=area([onMicrOnYSandOn(z,x) onMicrOnYSandOn(z,x)+VENTANA_EVENTO],X,...
                    'FaceColor','g');
                set(h,'BaseValue',min(cargaMicrOnYSandOn{1,z}))
            end
        end
        %Marca Off
        for x=1:size(offMicrOnYSandOn,2)
            %h=area([offMicrOnYSandOn(z,x) offMicrOnYSandOn(z,x)+VENTANA_EVENTO],[max(cargaMicrOnYSandOn{1,z}) max(cargaMicrOnYSandOn{1,z})],...
            if  offMicrOnYSandOn(z,x)
                h=area([offMicrOnYSandOn(z,x) offMicrOnYSandOn(z,x)+VENTANA_EVENTO],Y,...
                    'FaceColor','r');
                set(h,'BaseValue',min(cargaMicrOnYSandOn{1,z}))
            end
        end
        plot(cargaMicrOnYSandOn{1,z},'k')
        title('cargaMicrOnYSandOn')

    end
    pause;
    close all;

    for z=1:TRIALS
        figure;
        hold;
        %Marca Ventana
        X=ones(1,2)*max(cargaSandOnYBat3On{1,z});
        Y=ones(1,2)*max(cargaSandOnYBat3On{1,z});
        for x=1:size(vSandOnYBat3On{z},2)
            h=area([vSandOnYBat3On{z}(x) vSandOnYBat3On{z}(x)+VENTANA_EVENTO],[max(cargaSandOnYBat3On{1,z}) max(cargaSandOnYBat3On{1,z})],...
                'FaceColor','y');
            set(h,'BaseValue',min(cargaSandOnYBat3On{1,z}))
        end
```

```
1422            %Marca On
1423            for x=1:size(onSandOnYBat3On,2)
1424               %h=area([onSandOnYBat3On(z,x) onSandOnYBat3On(z,x)+VENTANA_EVENTO],[max(cargaSandOnYBat3On{1,z}) max(cargaSandOnYBat3On{1,z})],...
1425               if onSandOnYBat3On(z,x) %si es distinto de cero se pinta
1426                  h=area([onSandOnYBat3On(z,x) onSandOnYBat3On(z,x)+VENTANA_EVENTO],X,...
1427                     'FaceColor','g');
1428                  set(h,'BaseValue',min(cargaSandOnYBat3On{1,z}))
1429               end
1430            end
1431            %Marca Off
1432            for x=1:size(offSandOnYBat3On,2)
1433               %h=area([offSandOnYBat3On(z,x) offSandOnYBat3On(z,x)+VENTANA_EVENTO],[max(cargaSandOnYBat3On{1,z}) max(cargaSandOnYBat3On{1,z})],...
1434               if  offSandOnYBat3On(z,x)
1435                  h=area([offSandOnYBat3On(z,x) offSandOnYBat3On(z,x)+VENTANA_EVENTO],Y,...
1436                        'FaceColor','r');
1437                  set(h,'BaseValue',min(cargaSandOnYBat3On{1,z}))
1438               end
1439            end
1440            plot(cargaSandOnYBat3On{1,z},'k')
1441            title('cargaSandOnYBat3On')
1442
1443         end
1444         pause;
1445         close all;
1446
1447         for z=1:TRIALS
1448            figure;
1449            hold;
1450            %Marca Ventana
1451            X=ones(1,2)*max(cargaVenlOnYBat3On{1,z});
1452            Y=ones(1,2)*max(cargaVenlOnYBat3On{1,z});
1453            for x=1:size(vVenlOnYBat3On{z},2)
1454               h=area([vVenlOnYBat3On{z}(x) vVenlOnYBat3On{z}(x)+VENTANA_EVENTO],[max(cargaVenlOnYBat3On{1,z}) max(cargaVenlOnYBat3On{1,z})],...
1455                     'FaceColor','y');
1456               set(h,'BaseValue',min(cargaVenlOnYBat3On{1,z}))
1457            end
1458            %Marca On
1459            for x=1:size(onVenlOnYBat3On,2)
1460               %h=area([onVenlOnYBat3On(z,x) onVenlOnYBat3On(z,x)+VENTANA_EVENTO],[max(cargaVenlOnYBat3On{1,z}) max(cargaVenlOnYBat3On{1,z})],...
1461               if onVenlOnYBat3On(z,x) %si es distinto de cero se pinta
1462                  h=area([onVenlOnYBat3On(z,x) onVenlOnYBat3On(z,x)+VENTANA_EVENTO],X,...
1463                     'FaceColor','g');
1464                  set(h,'BaseValue',min(cargaVenlOnYBat3On{1,z}))
1465               end
1466            end
1467            %Marca Off
1468            for x=1:size(offVenlOnYBat3On,2)
1469               %h=area([offVenlOnYBat3On(z,x) offVenlOnYBat3On(z,x)+VENTANA_EVENTO],[max(cargaVenlOnYBat3On{1,z}) max(cargaVenlOnYBat3On{1,z})],...
1470               if  offVenlOnYBat3On(z,x)
1471                  h=area([offVenlOnYBat3On(z,x) offVenlOnYBat3On(z,x)+VENTANA_EVENTO],Y,...
1472                        'FaceColor','r');
1473                  set(h,'BaseValue',min(cargaVenlOnYBat3On{1,z}))
1474               end
1475            end
1476            plot(cargaVenlOnYBat3On{1,z},'k')
1477            title('cargaVenlOnYBat3On')
1478
1479         end
1480         pause;
1481         close all;
1482
1483         for z=1:TRIALS
1484            figure;
1485            hold;
1486            %Marca Ventana
```

```
1487            X=ones(1,2)*max(cargaVenIOnYMicrOnYSandOn{1,z});
1488            Y=ones(1,2)*max(cargaVenIOnYMicrOnYSandOn{1,z});
1489            for x=1:size(vVenIOnYMicrOnYSandOn{z},2)
1490                h=area([vVenIOnYMicrOnYSandOn{z}(x)                    vVenIOnYMicrOnYSandOn{z}(x)+VENTANA_EVENTO],[max(cargaVenIOnYMicrOnYSandOn{1,z})
1491    max(cargaVenIOnYMicrOnYSandOn{1,z})],...
1492                    'FaceColor','y');;
1493                set(h,'BaseValue',min(cargaVenIOnYMicrOnYSandOn{1,z}))
1494            end
1495            %Marca On
1496            for x=1:size(onVenIOnYMicrOnYSandOn,2)
1497                %h=area([onVenIOnYMicrOnYSandOn(z,x)                    onVenIOnYMicrOnYSandOn(z,x)+VENTANA_EVENTO],[max(cargaVenIOnYMicrOnYSandOn{1,z})
1498    max(cargaVenIOnYMicrOnYSandOn{1,z})],...
1499                if onVenIOnYMicrOnYSandOn(z,x) %si es distinto de cero se pinta
1500                    h=area([onVenIOnYMicrOnYSandOn(z,x) onVenIOnYMicrOnYSandOn(z,x)+VENTANA_EVENTO],X,...
1501                        'FaceColor','g');
1502                    set(h,'BaseValue',min(cargaVenIOnYMicrOnYSandOn{1,z}))
1503                end
1504            end
1505            %Marca Off
1506            for x=1:size(offVenIOnYMicrOnYSandOn,2)
1507                %h=area([offVenIOnYMicrOnYSandOn(z,x)                    offVenIOnYMicrOnYSandOn(z,x)+VENTANA_EVENTO],[max(cargaVenIOnYMicrOnYSandOn{1,z})
1508    max(cargaVenIOnYMicrOnYSandOn{1,z})],...
1509                if  offVenIOnYMicrOnYSandOn(z,x)
1510                    h=area([offVenIOnYMicrOnYSandOn(z,x) offVenIOnYMicrOnYSandOn(z,x)+VENTANA_EVENTO],Y,...
1511                        'FaceColor','r');;
1512                    set(h,'BaseValue',min(cargaVenIOnYMicrOnYSandOn{1,z}))
1513                end
1514            end
1515            plot(cargaVenIOnYMicrOnYSandOn{1,z},'k')
1516            title('cargaVenIOnYMicrOnYSandOn')
1517
1518        end
1519
1520        close all;
1521
1522
1523
1524
1525
```

TDC

ANEXO VI

SCRIPT PARA GRAFICADO

```
%% CARGAS

for z=1:TRIALS
    figure
   hold;
   plot(cargaBat1{1,z},'b')
   title('cargaBat1')
    plot(cargaBat1Filtr{1,z},'r')

end
pause;
close all;

for z=1:TRIALS
   figure
 hold;
  plot(cargaBat2{1,z})
   title('cargaBat2')

%    figure
   plot(cargaBat2Filtr{1,z},'r')
   title('cargaBat2Filtrada')

end
pause;
close all;

for z=1:TRIALS
   figure
   plot(cargaBat3{1,z})
   title('cargaBat3')
end
pause;
close all;

for z=1:TRIALS
   figure
   plot(cargaExpr{1,z})
   title('cargaExpr')
end
pause;
close all;

for z=1:TRIALS
   figure
   hold;
   plot(cargaMicr{1,z})
   title('cargaMicr')
   plot(cargaMicrFiltr{1,z},'r')
end
pause;
close all;

for z=1:TRIALS
   figure
   hold;
```

```
    plot(cargaSand{1,z})
    title('cargaSand')
    plot(cargaSandFiltr{1,z},'r')
    %title('cargaBat2Filtrada')


  end
  pause;
  close all;

  for z=1:TRIALS
    figure
    plot(cargaVent{1,z})
    title('cargaVent')
  end
  pause;
  close all;

  for z=1:TRIALS
    figure
    plot(cargaMicrOnYSandOn{1,z})
    title('cargaMicrOnYSandOn')
  end
  pause;
  close all;

  for z=1:TRIALS
    figure
    plot(cargaSandOnYBat3On{1,z})
    title('cargaSandOnYBat3On')
  end
  pause;
  close all;

  for z=1:TRIALS
    figure
    plot(cargaVentOnYBat3On{1,z})
    title('cargaVentOnYBat3On')
  end
  pause;
  close all;

  for z=1:TRIALS
    figure
    plot(cargaVentOnYMicrOnYSandOn{1,z})
    title('cargaVentOnYMicrOnYSandOn')
  end
  pause;
  close all;

  clear z;




  %% RMS


  figure
  plot(rmsBat1)
  title('rmsBat1')
  pause;
  close all;
```

```
figure
plot(rmsBat2)
title('rmsBat2')
pause;
close all;

figure
plot(rmsBat3)
title('rmsBat3')
pause;
close all;

figure
plot(rmsExpr)
title('rmsExpr')
pause;
close all;

figure
plot(rmsMicr)
title('rmsMicr')
pause;
close all;

figure
plot(rmsSand)
title('rmsSand')
pause;
close all;

figure
plot(rmsVenl)
title('rmsVenl')
pause;
close all;

figure
plot(rmsMicrOnYSandOn)
title('rmsMicrOnYSandOn')
pause;
close all;

figure
plot(rmsSandOnYBat3On)
title('rmsSandOnYBat3On')
pause;
close all;

figure
plot(rmsVenlOnYBat3On)
title('rmsVenlOnYBat3On')
pause;
close all;

figure
plot(rmsVenlOnYMicrOnYSandOn)
title('rmsVenlOnYMicrOnYSandOn')
pause;
close all;

TODOS=[rmsBatl rmsBat2 rmsBat3 rmsExpr rmsMicr ...
   rmsSand rmsVenl rmsMicrOnYSandOn rmsSandOnYBat3On ...
   rmsVenlOnYBat3On rmsVenlOnYMicrOnYSandOn];
GROUP=[ones(1,length(rmsBatl)), 2*ones(1,length(rmsBat2)), 3*ones(1,length(rmsBat3))...
, 4*ones(1,length(rmsExpr)), 5*ones(1,length(rmsMicr)),6*ones(1,length(rmsSand))...
```

```
, 7*ones(1,length(rmsVenl)), 8*ones(1,length(rmsMicrOnYSandOn)),9*ones(1,length(rmsSandOnYBat3On))...
, 10*ones(1,length(rmsVenlOnYBat3On)), 11*ones(1,length(rmsVenlOnYMicrOnYSandOn))];
nombresfigura={'BAT1' 'BAT2' 'BAT3' 'EXPR' 'MICR' 'SAND' 'VENT' 'M_S' 'S_B3' 'V_B3' 'V_M_S'}
figure;
boxplot(TODOS',GROUP','labels',nombresfigura,'labelorientation','inline');




%% CONCORDIA

figure
plot(CalfBat1,CbetBat1,'+')
title('CCDBat1')
pause

figure
plot(CalfBat2,CbetBat2,'+')
title('CCDBat2')
pause

figure
plot(CalfBat3,CbetBat3,'+')
title('CCDBat3')
pause

figure
plot(CalfExpr,CbetExpr,'+')
title('CCDExpr')
pause

figure
plot(CalfMicr,CbetMicr,'+')
title('CCDMicr')
pause

figure
plot(CalfSand,CbetSand,'+')
title('CCDSand')
pause

figure
plot(CalfVenl,CbetVenl,'+')
title('CCDVenl')
pause

figure
plot(CalfMicrOnYSandOn,CbetMicrOnYSandOn,'+')
title('CCDMicrOnYSandOn')
pause

figure
plot(CalfSandOnYBat3On,CbetSandOnYBat3On,'+')
title('CCDSandOnYBat3On')
pause

figure
plot(CalfVenlOnYBat3On,CbetVenlOnYBat3On,'+')
title('CCDVenlOnYBat3On')
pause

figure
plot(CalfVenlOnYMicrOnYSandOn,CbetVenlOnYMicrOnYSandOn,'+')
title('CCDVenlOnYMicrOnYSandOn')
```

```
252         pause
253
254         close all
255
256         %% FFT
257
258         figure
259         hold;
260         plot(fft1Batl,'color','r')
261         plot(fft2Batl,'color','g')
262         plot(fft3Batl,'color','b')
263         title('FFTBatl r=1, g=2, b=3')
264         pause;
265         close all;
266
267         figure
268         hold;
269         plot(fft1Bat2,'color','r')
270         plot(fft2Bat2,'color','g')
271         plot(fft3Bat2,'color','b')
272         title('FFTBat2 r=1, g=2, b=3')
273         pause;
274         close all;
275
276         figure
277         hold;
278         plot(fft1Bat3,'color','r')
279         plot(fft2Bat3,'color','g')
280         plot(fft3Bat3,'color','b')
281         title('FFTBat3 r=1, g=2, b=3')
282         pause;
283         close all;
284
285         figure
286         hold;
287         plot(fft1Expr,'color','r')
288         plot(fft2Expr,'color','g')
289         plot(fft3Expr,'color','b')
290         title('FFTExpr r=1, g=2, b=3')
291         pause;
292         close all;
293
294         figure
295         hold;
296         plot(fft1Micr,'color','r')
297         plot(fft2Micr,'color','g')
298         plot(fft3Micr,'color','b')
299         title('FFTMicr r=1, g=2, b=3')
300         pause;
301         close all;
302
303         figure
304         hold;
305         plot(fft1Sand,'color','r')
306         plot(fft2Sand,'color','g')
307         plot(fft3Sand,'color','b')
308         title('FFTSand r=1, g=2, b=3')
309         pause;
310         close all;
311
312         figure
313         hold;
314         plot(fft1Venl,'color','r')
315         plot(fft2Venl,'color','g')
316         plot(fft3Venl,'color','b')
```

```
317         title('FFTVenI r=1, g=2, b=3')
318         pause;
319         close all;
320
321         figure
322         hold;
323         plot(fftIMicrOnYSandOn,'color','r')
324         plot(fft2MicrOnYSandOn,'color','g')
325         plot(fft3MicrOnYSandOn,'color','b')
326         title('FFTMicrOnYSandOn r=1, g=2, b=3')
327         pause;
328         close all;
329
330         figure
331         hold;
332         plot(fftISandOnYBat3On,'color','r')
333         plot(fft2SandOnYBat3On,'color','g')
334         plot(fft3SandOnYBat3On,'color','b')
335         title('FFTSandOnYBat3On r=1, g=2, b=3')
336         pause;
337         close all;
338
339         figure
340         hold;
341         plot(fftIVenIOnYBat3On,'color','r')
342         plot(fft2VenIOnYBat3On,'color','g')
343         plot(fft3VenIOnYBat3On,'color','b')
344         title('FFTVenIOnYBat3On r=1, g=2, b=3')
345         pause;
346         close all;
347
348         figure
349         hold;
350         plot(fftIVenIOnYMicrOnYSandOn,'color','r')
351         plot(fft2VenIOnYMicrOnYSandOn,'color','g')
352         plot(fft3VenIOnYMicrOnYSandOn,'color','b')
353         title('FFTVenIOnYMicrOnYSandOn r=1, g=2, b=3')
354         pause;
355         close all;
356
357         TODOS=[fftIBatI  fftIBat2  fftIBat3  fftIExpr  fftIMicr  ...
358            fftISand fftIVenI fftIMicrOnYSandOn fftISandOnYBat3On ...
359            fftIVenIOnYBat3On fftIVenIOnYMicrOnYSandOn];
360         GROUP=[ones(1,length(fftIBatI)), 2*ones(1,length(fftIBat2)), 3*ones(1,length(fftIBat3))...
361         , 4*ones(1,length(fftIExpr)), 5*ones(1,length(fftIMicr)),6*ones(1,length(fftISand))...
362         , 7*ones(1,length(fftIVenI)), 8*ones(1,length(fftIMicrOnYSandOn)),9*ones(1,length(fftISandOnYBat3On))...
363         , 10*ones(1,length(fftIVenIOnYBat3On)), 11*ones(1,length(fftIVenIOnYMicrOnYSandOn))];
364         nombresfigura={'BAT1'  'BAT2'  'BAT3'  'EXPR'  'MICR'  'SAND'  'VENT'  'M_S'  'S_B3'  'V_B3'  'V_M_S'}
365         figure;
366         boxplot(TODOS',GROUP','labels',nombresfigura,'labelorientation','inline');
367
368
369         TODOS=[fft2BatI  fft2Bat2  fft2Bat3  fft2Expr  fft2Micr  ...
370            fft2Sand fft2VenI fft2MicrOnYSandOn fft2SandOnYBat3On ...
371            fft2VenIOnYBat3On fft2VenIOnYMicrOnYSandOn];
372         GROUP=[ones(1,length(fft2BatI)), 2*ones(1,length(fft2Bat2)), 3*ones(1,length(fft2Bat3))...
373         , 4*ones(1,length(fft2Expr)), 5*ones(1,length(fft2Micr)),6*ones(1,length(fft2Sand))...
374         , 7*ones(1,length(fft2VenI)), 8*ones(1,length(fft2MicrOnYSandOn)),9*ones(1,length(fft2SandOnYBat3On))...
375         , 10*ones(1,length(fft2VenIOnYBat3On)), 11*ones(1,length(fft2VenIOnYMicrOnYSandOn))];
376         nombresfigura={'BAT1'  'BAT2'  'BAT3'  'EXPR'  'MICR'  'SAND'  'VENT'  'M_S'  'S_B3'  'V_B3'  'V_M_S'}
377         figure;
378         boxplot(TODOS',GROUP','labels',nombresfigura,'labelorientation','inline');
379
380
381         TODOS=[fft3BatI  fft3Bat2  fft3Bat3  fft3Expr  fft3Micr  ...
```

```
382         fft3Sand fft3VenI fft3MicrOnYSandOn fft3SandOnYBat3On ...
383         fft3VenIOnYBat3On fft3VenIOnYMicrOnYSandOn];
384     GROUP=[ones(1,length(fft3Bat1)), 2*ones(1,length(fft3Bat2)), 3*ones(1,length(fft3Bat3))...
385     , 4*ones(1,length(fft3Expr)), 5*ones(1,length(fft3Micr)),6*ones(1,length(fft3Sand))...
386     , 7*ones(1,length(fft3VenI)), 8*ones(1,length(fft3MicrOnYSandOn)),9*ones(1,length(fft3SandOnYBat3On))...
387     , 10*ones(1,length(fft3VenIOnYBat3On)), 11*ones(1,length(fft3VenIOnYMicrOnYSandOn))];
388     nombresfigura={'BAT1' 'BAT2' 'BAT3' 'EXPR' 'MICR' 'SAND' 'VENT' 'M_S' 'S_B3' 'V_B3' 'V_M_S'}
389     figure;
390     boxplot(TODOS',GROUP','labels',nombresfigura,'labelorientation','inline');
391
392
393     %% FACTOR POTENCIA
394
395     figure
396     plot(medFacPotBat1)
397     title('medFacPotBat1')
398     pause;
399     close all;
400
401
402     figure
403     plot(medFacPotBat2)
404     title('medFacPotBat2')
405     pause;
406     close all;
407
408     figure
409     plot(medFacPotBat3)
410     title('medFacPotBat3')
411     pause;
412     close all;
413
414     figure
415     plot(medFacPotExpr)
416     title('medFacPotExpr')
417     pause;
418     close all;
419
420     figure
421     plot(medFacPotMicr)
422     title('medFacPotMicr')
423     pause;
424     close all;
425
426     figure
427     plot(medFacPotSand)
428     title('medFacPotSand')
429     pause;
430     close all;
431
432     figure
433     plot(medFacPotVenI)
434     title('medFacPotVenI')
435     pause;
436     close all;
437
438     figure
439     plot(medFacPotMicrOnYSandOn)
440     title('medFacPotMicrOnYSandOn')
441     pause;
442     close all;
443
444     figure
445     plot(medFacPotSandOnYBat3On)
446     title('medFacPotSandOnYBat3On')
```

```
pause;
close all;

figure
plot(medFacPotVentIOnYBat3On)
title('medFacPotVentIOnYBat3On')
pause;
close all;

figure
plot(medFacPotVentIOnYMicrOnYSandOn)
title('medFacPotVentIOnYMicrOnYSandOn')
pause;
close all;

TODOS=[medFacPotBatI  medFacPotBat2  medFacPotBat3  medFacPotExpr  medFacPotMicr  ...
   medFacPotSand medFacPotVentI medFacPotMicrOnYSandOn medFacPotSandOnYBat3On ...
   medFacPotVentIOnYBat3On medFacPotVentIOnYMicrOnYSandOn];
GROUP=[ones(1,length(medFacPotBatI)), 2*ones(1,length(medFacPotBat2)), 3*ones(1,length(medFacPotBat3))...
, 4*ones(1,length(medFacPotExpr)), 5*ones(1,length(medFacPotMicr)),6*ones(1,length(medFacPotSand))...
, 7*ones(1,length(medFacPotVentI)), 8*ones(1,length(medFacPotMicrOnYSandOn)),9*ones(1,length(medFacPotSandOnYBat3On))...
, 10*ones(1,length(medFacPotVentIOnYBat3On)), 11*ones(1,length(medFacPotVentIOnYMicrOnYSandOn))];
nombresfigura={'BATI'  'BAT2'  'BAT3'  'EXPR'  'MICR'  'SAND'  'VENT'  'M_S'  'S_B3'  'V_B3'  'V_M_S'}
figure;
boxplot(TODOS',GROUP','labels',nombresfigura,'labelorientation','inline');



%% PICO

figure
plot(medPicoBatI)
title('medPicoBatI')
pause;
close all;


figure
plot(medPicoBat2)
title('medPicoBat2')
pause;
close all;

figure
plot(medPicoBat3)
title('medPicoBat3')
pause;
close all;

figure
plot(medPicoExpr)
title('medPicoExpr')
pause;
close all;

figure
plot(medPicoMicr)
title('medPicoMicr')
pause;
close all;

figure
plot(medPicoSand)
title('medPicoSand')
pause;
```

```
close all;

figure
plot(medPicoVenI)
title('medPicoVenI')
pause;
close all;

figure
plot(medPicoMicrOnYSandOn)
title('medPicoMicrOnYSandOn')
pause;
close all;

figure
plot(medPicoSandOnYBat3On)
title('medPicoSandOnYBat3On')
pause;
close all;

figure
plot(medPicoVenIOnYBat3On)
title('medPicoVenIOnYBat3On')
pause;
close all;

figure
plot(medPicoVenIOnYMicrOnYSandOn)
title('medPicoVenIOnYMicrOnYSandOn')
pause;
close all;

%BOXPLOTS

TODOS=[medPicoBatI  medPicoBat2  medPicoBat3 medPicoExpr  medPicoMicr  ...
   medPicoSand medPicoVenI medPicoMicrOnYSandOn medPicoSandOnYBat3On ...
   medPicoVenIOnYBat3On medPicoVenIOnYMicrOnYSandOn];
GROUP=[ones(1,length(medPicoBatI)), 2*ones(1,length(medPicoBat2)), 3*ones(1,length(medPicoBat3))...
, 4*ones(1,length(medPicoExpr)), 5*ones(1,length(medPicoMicr)),6*ones(1,length(medPicoSand))...
, 7*ones(1,length(medPicoVenI)), 8*ones(1,length(medPicoMicrOnYSandOn)),9*ones(1,length(medPicoSandOnYBat3On))...
, 10*ones(1,length(medPicoVenIOnYBat3On)), 11*ones(1,length(medPicoVenIOnYMicrOnYSandOn))];
nombresfigura={'BATI'  'BAT2'  'BAT3'  'EXPR'  'MICR'  'SAND'  'VENT'  'M_S'  'S_B3'  'V_B3'  'V_M_S'}
figure;
boxplot(TODOS',GROUP','labels',nombresfigura,'labelorientation','inline');

%% CUARTO SEMIPERIODO

figure
plot(medCuartoBatI)
title('medCuartoBatI')
pause;
close all;


figure
plot(medCuartoBat2)
title('medCuartoBat2')
pause;
close all;


figure
plot(medCuartoBat3)
title('medCuartoBat3')
pause;
```

```
577         close all;
578
579
580         figure
581         plot(medCuartoExpr)
582         title('medCuartoExpr')
583         pause;
584         close all;
585
586
587         figure
588         plot(medCuartoMicr)
589         title('medCuartoMicr')
590         pause;
591         close all;
592
593
594         figure
595         plot(medCuartoSand)
596         title('medCuartoSand')
597         pause;
598         close all;
599
600
601         figure
602         plot(medCuartoVenI)
603         title('medCuartoVenI')
604         pause;
605         close all;
606
607         %
608         % figure
609         % plot(medCuartoVen2)
610         % title('medCuartoVen2')
611         % pause;
612         % close all;
613
614
615         figure
616         plot(medCuartoMicrOnYSandOn)
617         title('medCuartoMicrOnYSandOn')
618         pause;
619         close all;
620
621
622         figure
623         plot(medCuartoSandOnYBat3On)
624         title('medCuartoSandOnYBat3On')
625         pause;
626         close all;
627
628
629         figure
630         plot(medCuartoVenIOnYBat3On)
631         title('medCuartoVenIOnYBat3On')
632         pause;
633         close all;
634
635
636         figure
637         plot(medCuartoVenIOnYMicrOnYSandOn)
638         title('medCuartoVenIOnYMicrOnYSandOn')
639         pause;
640         close all;
641
```

```
642 TODOS=[medCuartoBatI  medCuartoBat2  medCuartoBat3  medCuartoExpr  medCuartoMicr  ...
643    medCuartoSand medCuartoVenI medCuartoMicrOnYSandOn medCuartoSandOnYBat3On ...
644    medCuartoVenIOnYBat3On medCuartoVenIOnYMicrOnYSandOn];
645 GROUP=[ones(1,length(medCuartoBatI)), 2*ones(1,length(medCuartoBat2)), 3*ones(1,length(medCuartoBat3))...
646 , 4*ones(1,length(medCuartoExpr)), 5*ones(1,length(medCuartoMicr)),6*ones(1,length(medCuartoSand))...
647 , 7*ones(1,length(medCuartoVenI)), 8*ones(1,length(medCuartoMicrOnYSandOn)),9*ones(1,length(medCuartoSandOnYBat3On))...
648 , 10*ones(1,length(medCuartoVenIOnYBat3On)), 11*ones(1,length(medCuartoVenIOnYMicrOnYSandOn))];
649 nombresfigura={'BATI'  'BAT2'  'BAT3'  'EXPR'  'MICR'  'SAND'  'VENT'   'M_S'   'S_B3'  'V_B3'  'V_M_S'}
650 figure;
651 boxplot(TODOS',GROUP','labels',nombresfigura,'labelorientation','inline');
652
653
654 %% NUMERO MUESTRAS
655
656 % disp(['','BatI', 'Bat2', 'Bat3', 'Expr', 'Micr', 'Sand', 'VenI', 'MicrOnYSandOn', 'SandOnYBat3On', 'VenIOnYBat3On', 'VenIOnYMicrOnYSandOn'])
657 % disp(['Train' 'Test'];
658 disp('total   Train   Test')
659 disp([sum(matTarget');sum(matTargetEntrenaSom');sum(matTargetTestSom')]')
660 % disp(sum(matTargetEntrenaSom));
661 % disp(sum(matTargetTestSom));
662 % tabla2{1,1}=cab;
663 % tabla2{2,1}=sum(matTargetEntrenaSom');
664 % tabla2{3,1}=sum(matTargetTestSom');
665
666
667
668
669
670
671
672
673
674
```

TDC