



UNIVERSIDAD NACIONAL
AUTÓNOMA DE
MÉXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

**DETECCIÓN DE ERRORES CON BASE EN
WAVELETS UTILIZANDO MAPAS
AUTOORGANIZADOS**

T E S I S

QUE PARA OBTENER EL GRADO DE:

**INGENIERO EN ELÉCTRICA ELECTRÓNICA
(MÓDULO DE SISTEMAS DIGITALES)**

P R E S E N T A:

LEÓN FELIPE PALAFOX NOVACK

DIRECTOR DE TESIS:

DR. HÉCTOR BENÍTEZ PÉREZ

México, D.F.

2005.

Agradecimiento

Al proyecto PAPIIT IN105305 por la beca recibida para poder llevar a cabo esta tesis de licenciatura.

Agradezco a mi madre, por su apoyo, sus noches en vela y por cada vez
que me deseó suerte cuando salía a un examen en la madrugada.

A mi padre, por estar ahí cuando lo necesitaba y nunca dejarme caminar
sólo.

A mis maestros como Pablo y César que siempre confiaron en mi.

A Héctor y Francisco por darme un lugar y una oportunidad para realizar
mis sueños.

A todos mis amigos por permitirme vivir en esta Facultad de Ingeniería de
la manera más humana posible.

Y a todas aquellas personas que en algún momento, en algún lugar,
mostraron apoyo y un par de saludos.

Índice

1. Introducción	1
1.1 Objetivos.....	3
1.2 Alcances.....	3
1.3 Mapa de la tesis.....	3
2. Generalidades	4
2.1 Onduletas.....	4
2.1.1 Introducción.....	4
2.1.2 Transformada discreta de Fourier y FFT.....	7
2.1.3 Propiedades de la DFT.....	8
2.1.4 Transformada rápida de Fourier.....	9
2.1.5 Transformada de Onduleta.....	10
2.1.6 Onduletas en la detección de errores.....	12
2.2 Redes Neuronales Artificiales – Mapas Autoorganizados.....	14
2.2.1 Introducción.....	14
2.2.2 Disposición básica de una neurona.....	14
2.2.3 Funcionamiento de la neurona artificial.....	16
2.2.4 Mapas Autoorganizados.....	18
2.3 Detección de errores.....	21
2.3.1 Introducción.....	21
2.3.2 Modelos en la detección de errores.....	22
2.3.3 Algoritmo de detección de falla.....	24
2.3.4 Detección de fallas usando mapeos autoorganizados.....	25
2.4 Conclusiones.....	25
3. Algoritmo Propuesto.....	27
3.1 Unión de Wavelets y Mapas autoorganizados.	28
3.2 Diagrama de flujo del sistema.....	28
3.3 Metodología del algoritmo.	29
3.2.1 Datos de entrada.....	29
3.2.2 Arquitectura Onduleta – SOM.....	32
3.2.3 Desarrollo en línea.....	33
3.4 Conclusiones.....	35
4. Parámetros del algoritmo.....	36
4.1 Introducción.....	37
4.2 Número de neuronas.....	37

4.3	Valor de σ	39
4.4	Tamaño de las muestras.....	45
4.5	Conclusiones.....	46
5.	Caso de estudio.....	48
5.1	Introducción.....	49
5.2	Levitador electromagnético.....	49
5.3	Conclusiones.....	52
6.	Resultados	53
6.1	Introducción.....	54
6.2	Entrenamiento de la señal sin falla.....	54
6.3	Sistema en línea.....	57
6.4	Conclusiones.....	62
7.	Conclusiones.....	64
8.	Apéndice.....	66
8.1	Programas realizado para llevar a cabo la arquitectura.....	67
9.	Bibliografía.....	76

Uno de los grandes retos en la industria es lograr la completa automatización de los procesos para la manufactura o desarrollo de algún producto. En las últimas décadas se ha hecho un gran esfuerzo por lograr alcanzar esta meta.

Sin embargo existe un punto clave que no se ha logrado automatizar, la detección y corrección de fallas. Esta tarea se le ha relegado a operadores humanos, debido a que las fallas no pueden ser predichas y se requiere de un proceso de clasificación que sólo se logra mediante la experiencia de fallas ya conocidas.

El problema de clasificación de fallas se centra entonces en encontrar una forma de categorizar las fallas, no importando la naturaleza de éstas, y lograr agruparlas de cierta manera, para que fallas similares tengan respuestas similares. De esta forma se active la misma alarma si ocurren 2 fallas que en esencia son la misma, pero que en un sistema de cómputo puedan determinarse como distintas.

Para categorizar las fallas, se puede utilizar el análisis multidimensional de las señales. Es decir realizar una descomposición de algún tipo de señales para obtener sus características esenciales, y que sean las que el sistema de control aprenda. De esta forma podríamos catalogar a las fallas no por su comportamiento en el tiempo, sino por sus características frecuenciales.

Una de estas herramientas resultan ser las onduletas, que como se describe detalladamente en el capítulo 2 nos otorgan una descomposición de la señal en niveles de detalle, y no sólo eso, sino que estos niveles de detalle centran toda su energía en las fallas.

Las fallas necesitan categorizarse, por lo que se requiere una herramienta que no necesite de una supervisión constante de un operador. Una herramienta que categorice a los elementos por si misma. Al igual que en el caso anterior existen varias soluciones, como puede ser la estadística Bayessiana, que implica cálculo matemático muy fuerte. Otra herramienta muy usada y que ha mostrado ser confiable y útil son los mapas autoorganizados. Esta arquitectura hace uso del principio de las redes neuronales para su funcionamiento, herramienta que por su simplicidad y su demostrada eficacia en muchos casos anteriores, como la agrupación de elementos en un alfabeto fonético [Freeman, 1992] , es la utilizada en el desarrollo de esta tesis.

1.1 Objetivo

Definición de un algoritmo para la clasificación de fallas utilizando Mapas Autoorganizados (SOM) y Onduletas con el objeto de tener una arquitectura que detecte fallas con distintas respuestas en el campo de la frecuencia.

1.2 Alcances

Esta tesis pretende diseñar una herramienta que permita detectar las fallas y aislarlas dependiendo de su naturaleza en el campo de las frecuencias para que de esta manera, dentro de un sistema físico, la arquitectura sea capaz de reconocer cuando ocurre una falla y al mismo tiempo reajustar el sistema y aislar la falla de tal manera que si se presenta de nuevo en un futuro el sistema la identifique y tome una acción preventiva para corregir la misma.

1.3 Mapa de la tesis

La tesis esta organizada de tal manera que al principio se detallan los elementos teóricos de las herramientas a utilizar. El capítulo 2 define conceptos básicos de onduletas, redes neuronales y detección de fallas, para comprender la naturaleza, fortaleza y utilidad del algoritmo propuesto. El capítulo 3 describe el algoritmo, su funcionalidad, así como las variables que intervienen en la arquitectura de Onduleta-SOM. Las variables del algoritmo son evaluadas mediante diversas pruebas de rendimiento y se detallaran las mismas junto con sus resultados en el capítulo 4. En el capítulo 5 se explican a groso modo las componentes del sistema que se va a analizar y se aplicará nuestra arquitectura(un levitador magnético). Y por último el capítulo 6 presenta los resultados que obtuvimos en este sistema físico, se comparan errores contra aciertos y se reportan los tiempos que tarda en detectar una falla el sistema.

Finalmente se presentan las conclusiones generales del trabajo, incluyendo la viabilidad de la arquitectura.

Detección de errores con base en wavelets utilizando mapas autoorganizados

Resumen:

Para poder realizar la detección de fallas se cuenta con varias herramientas como son los modelos basados en el conocimiento, los modelos basados en el sistema físico, los modelos heurísticos, y los modelos basados en redes neuronales con información en múltiples resoluciones de las señales. En el desarrollo de esta tesis se utilizara este ultimo, utilizando como herramienta de análisis de la señal, la transformada Wavelet, y la arquitectura de red neuronal de Mapas Autoorganizados, esto debido a que el análisis de Wavelet nos proporciona un análisis multidimensional que se enfoca principalmente en las regiones donde existen cambios drásticos de la señal. Mientras que los mapas autoorganizados nos permiten clasificar de manera no supervisada a distintas señales que se encuentren dentro de un universo dado.

la arquitectura consiste en realizar una descomposición de wavelet a la señal que se le va a alimentar a la red neuronal, esta descomposición es alimentada al mapa autoorganizado para así crear diversas zonas de conocimiento. Una vez entrenado el mapa, se le alimentara a este la señal sin ser procesada por la transformación de wavelet.

2.1 Onduletas – Wavelets

2.1.1 Introducción

En este capítulo se explican los conceptos básicos y fundamentales del análisis de onduletas. Un análisis que, debido a sus diversas propiedades, ha encontrado una gran cantidad de aplicaciones, como son el procesamiento de señales, filtrado y compresión. [Marcos et Al,1998]

Las onduletas son usualmente referidas como el análisis de Fourier del siglo XXI. Estas explotan el análisis multiresolución de la misma forma que los microscopios lo hacen en la microbiología. El nacimiento de las onduletas fue en 1910 cuando A. Haar propuso la aproximación de escalera de una función, utilizando pequeñas constantes, denominadas Onduletas Haar [A.Haar, 1910]. Dando origen a una serie de contribuciones:

- Paley-Littlewood propusieron una frecuencia de agrupamiento en ordenes de 2 en 1938 [Paley et al 1937]
- Shannon derivó la teoría del muestreo en 1948 [Shannon, 1948]
- Calderon empleó la descomposición atómica de distribuciones en los espacios parabólicos H^p en 1977 [Calderon, 1977]
- Stromberg mejoró los sistemas de Haar en 1981 [Stromberg, 1981]
- Grossman y Morlet descompusieron las funciones de Hardy en onduletas cuadradas integrables para señales sísmicas en 1984 [Grossman et al, 1984]
- Meyer construyó bases ortogonales en L^2 (espacio de Hilbert) con retardo y traslación de una función suave en 1986 [Meyer,1986]
- Mallat introdujo el análisis multiresolución (MRA) en 1988 y unificó las construcciones individuales de onduletas de Strombreg, Battle y Meyer [Mallat, 1988]
- Daubechies es la primera en construir un sistema ortogonal de onduletas compacto en 1987 [Daubechies, 1987]

Mientras que en el análisis de Fourier, se trata de representar una señal por medio de una combinación lineal de señales senoidales, en el análisis de onduletas, se utilizan como base generadora la señal denominada onduleta madre. Las onduletas permiten descomponer o proyectar la señal sobre bases con distintas características.

La palabra “onduleta” es utilizada en matemáticas para denotar una base dentro del campo de los complejos, en el espacio de Hilbert definido de la siguiente forma $\{x = (x_i) \mid x_i \in \mathbb{C}, i \in \mathbb{N}, \sum_{i=1}^{\infty} x_i^2 < \infty\}$, la suma que tiene como condición es una característica que nos indica que las señales en el espacio contienen energía.

El valor promedio de una onduleta es cero, además de que tienen poca duración efectiva, en contraste con una señal senoidal cuya duración efectiva se extiende al infinito. [Härdle, 1997]

Gráficamente podemos apreciar una onduleta de tipo “Daubechies” en la figura 2.1

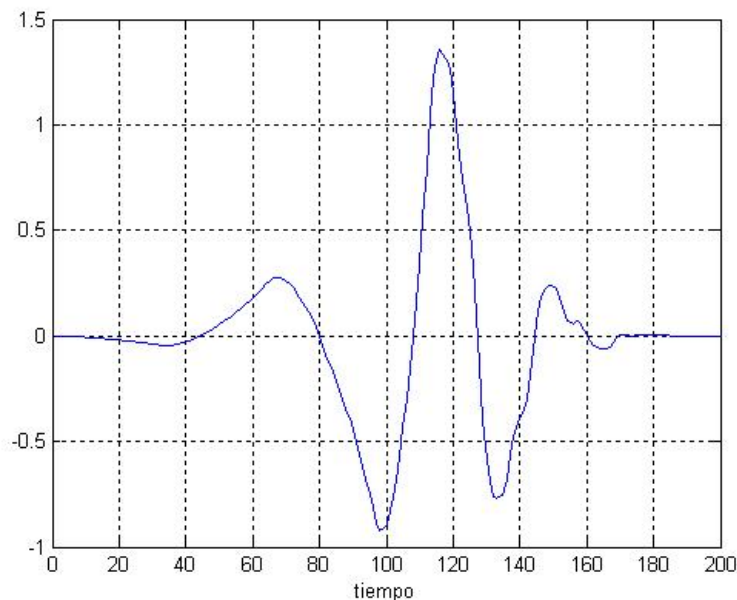


Figura 2.1 Representación gráfica de una onduleta

Una de las razones primordiales para utilizar este análisis es que la transformación que da la onduleta es en tiempo y frecuencia, en contraste con la transformada de Fourier que sólo permite ver el comportamiento en frecuencia o en tiempo. Aunado a esto en el análisis de Fourier, todas las componentes tienen la misma magnitud en el dominio de la frecuencia. Esta diferencia, nos otorga con el análisis de onduletas una herramienta que procesa sólo los puntos donde haya cambios drásticos en la señal, como por ejemplo, en la transmisión de video, se puede realizar un análisis de las

componentes de la imagen que cambien, como pueden ser personas u objetos en movimiento, mientras que el fondo, ya no tendría que ser reprocesado a cada momento de la transmisión, otros ejemplos son fallas en la geografía terrestre, o alguna falla en algún material. [Frazier,1999]

El interés en la teoría de onduletas y sus aplicaciones se ha visto acelerado con la introducción de algoritmos eficientes para el análisis, compresión, estimación y aproximación de funciones y señales. El más popular de estos algoritmos es la transformada de onduleta discreta (DWT), que usa bases de propósito general, capaces de representar varios tipos de funciones. Las bases usadas en el algoritmo de la DWT son especialmente aptas para datos uniformemente muestreados.

2.1.2 Transformada discreta de Fourier y FFT

Cabe mencionar que funcionalmente, la DWT es muy similar a la transformada discreta de Fourier, en el hecho que la función de transformación para ambas es ortogonal, dependiendo de la onduleta madre que se utilice. Ambas transformadas son reversibles y en que la señal de entrada se da por hecho que es un conjunto de muestras discretas. Ambas transformadas son al final convoluciones. [Frazier,1999]

Para realizar el procesamiento digital de señales se hace uso de computadoras, o hardware digital especialmente diseñado, el filtrado de señales o algún tratamiento en el dominio de la frecuencia de alguna señal, se debe transformar la señal del dominio del tiempo al dominio de la frecuencia, la transformada de Fourier ya ofrece esta ventaja, pero al ser una transformada continua en el tiempo, no resulta computacionalmente factible, por lo que se hace uso de la transformada discreta de Fourier (DFT).

La transformada de Fourier representa una señal lo mas fielmente posible dentro del dominio de la frecuencia, expresando a la señal como una combinación lineal de senos y cosenos, esto debido a que en el dominio de la frecuencia, una señal senoidal se ve representada como una espiga con posición y magnitud, así que se trata de hacer una combinación de estas espigas, es decir de señales senoidales de distinta magnitud y frecuencia. [Proakis,1998]

En la figura 2.2 se muestran algunos ejemplos de como se pueden representar gráficamente diversas funciones por medio de señales senoidales.

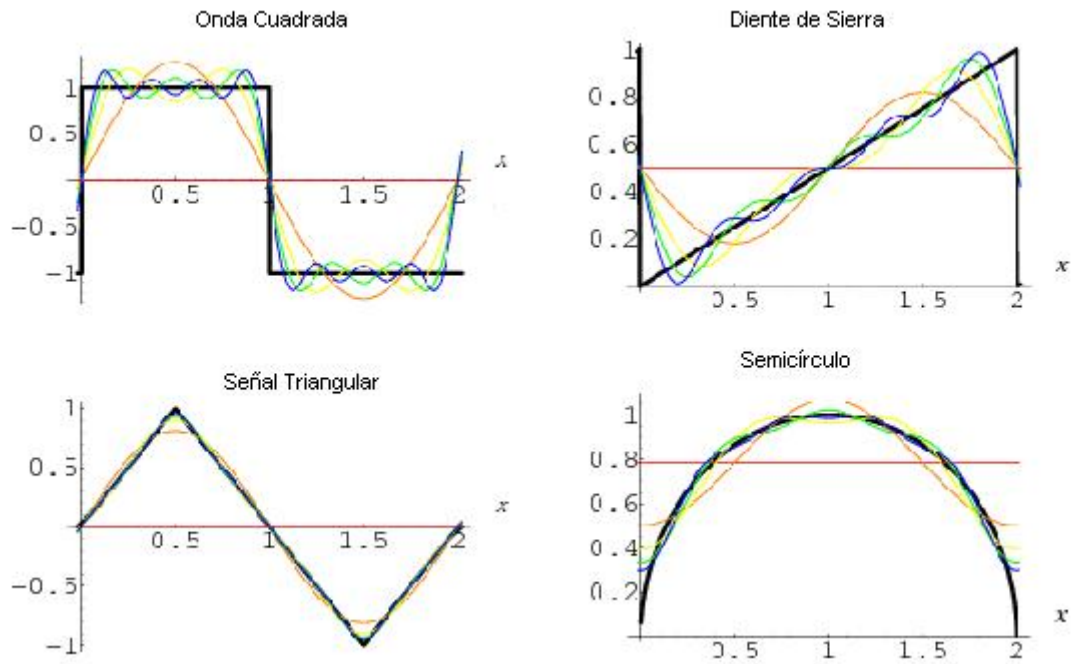


Figura 2.2 Representación gráfica de algunas señales como una composición de senoidales

2.1.3 Propiedades de la DFT

Periodicidad: Al aplicar la DFT a una señal, se obtiene una señal periódica en el dominio de la frecuencia.

Siendo $x(n)$ y $X(k)$ un par de transformadas DFT de N puntos, entonces

$$x(n) \xrightarrow{DFT} X(k)$$

$$x(n) + N = x(n) \quad \forall \quad n \in [0, N-1] \dots\dots\dots (2.1)$$

$$X(k + N) = X(k) \quad \forall \quad k \in [0, N-1] \dots\dots\dots (2.2)$$

Linealidad : Sean 2 pares de transformadas

$$x_1(n) \rightarrow X_1(k)$$

$$x_2(n) \rightarrow X_2(k)$$

Entonces para cualesquiera constantes reales o complejas a_1 y a_2

$$a_1x_1 + a_2x_2 \rightarrow a_1X_1 + a_2X_2 \dots\dots\dots(2.3)$$

Esto quiere decir que una composición lineal en el dominio del tiempo se transforma no sólo como una composición lineal en el dominio de la frecuencia, sino que además con los mismos coeficientes.

Después de ver las propiedades pasaremos a definir matemáticamente la transformada de Fourier discreta, que en términos de la simbología anterior se define de la siguiente manera:

Una señal $x(n)$ de N puntos con $n = 0,1,\dots,N-1$ tiene como DFT la siguiente ecuación:

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N} \quad k = 0,1,2,\dots,N-1 \dots\dots\dots(2.4)$$

Observamos entonces que entre mayor sea el número de puntos en la señal, la cantidad de cálculos se incrementa exponencialmente por lo que resulta computacionalmente inconveniente. Los algoritmos de la DFT, herramienta fundamental en el procesamiento digital de señales, han sido desarrollados para disminuir los tiempos de procesamiento de las señales. Ejemplos de esto son los algoritmos de Goertzel y la transformada rápida de Fourier.

2.1.4 Transformada rápida de Fourier (FFT)

El principio fundamental de la FFT es el principio “divide y vencerás”, donde se divide la señal original, y lo que queda se procesa por separado, y después se une, con el fin de disminuir los tiempos de procesamiento. [Proakis, 1998]

En esencia al disminuir el orden, se tienen que cambiar las bases ya que dependen directamente de la N o el número de puntos de las señales a transformar, y este cambio de base se debe realizar sucesivamente conforme se vaya disminuyendo la señal, la FFT tiene como principal limitante el hecho de sólo poder ser aplicada a

señales cuya dimensión sea de 2^n , debido a que como se observa en la figura 2.3, se debe realizar con pares de señales.

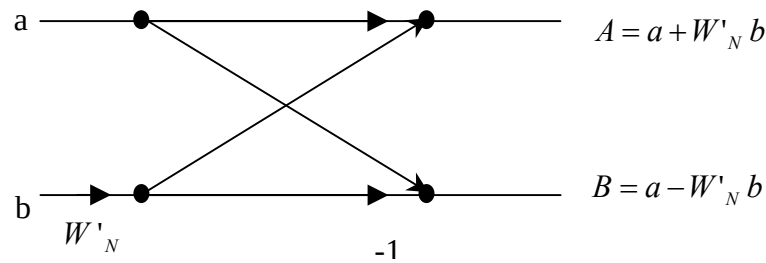


Figura 2.3 Estructura básica del diagrama de mariposa

Donde

$$W'_N = e^{-2j\frac{\pi}{N}} \dots\dots\dots(2.5)$$

A estos se les denomina diagramas de mariposa, y son una representación grafica del algoritmo de la FFT y computacionalmente hablando se le denomina el algoritmo básico de esta transformada

2.1.5 Transformada de Onduleta

Una vez explicado el principio de la DFT, es fácil saber que lo que se necesita es cambiar una base no localizada en el tiempo, por una base localizada, y la respuesta a esto es la onduleta.

Se busca una base computable por medio de los algoritmos de la FFT, ya que han demostrado ser efectivos y rápidos, se hacen pruebas hasta que se encuentra una base adecuada, sin embargo, para que cumpla con las propiedades de ortogonalidad, se deben de crear 2 bases, logrando así que el vector generador de la transformada onduleta es en realidad la unión de 2 bases, cada una computable por los algoritmos de la transformada de Fourier. Es decir, cada una realiza una convolución con el vector de entradas. [Frazier , 1999]

El principio básico de estas convoluciones es el filtrado, como se muestra en la figura 2.4 cada una de estas convoluciones tendrá a la salida 2 vectores distintos de la señal de entrada s , las frecuencias bajas y las frecuencias altas, en las señales F y G respectivamente se verán estas salidas, con esto, la transformada onduleta también muestra ser una efectiva herramienta para el filtrado de la señal.

Después de esto, a cada una de estas señales se le realiza una decimación, esto reduce su tamaño a la mitad, obteniendo una reducción en frecuencia, teniendo al final 2 elementos, el que obtenemos del filtro paso bajas y el que obtenemos del filtro paso altas, el que obtenemos del filtro paso bajas es una señal que es el promedio de la señal original, tan sólo que con una cantidad menor de valores, y lo que nos interesa en este trabajo, es lo que obtenemos del filtro paso altas, que es llamado nivel de detalle, que contiene los detalles o cambios que sufre la señal, y es colocado en un vector de misma longitud al vector de compresión. Este nivel de detalle, conforme se van realizando mas y más transformaciones, se va centrando sobre cambios bruscos dentro de la frecuencia o el tiempo, mientras que todo aquello que permanezca prácticamente igual, no le realiza ninguna operación. [Frazier, 1999]

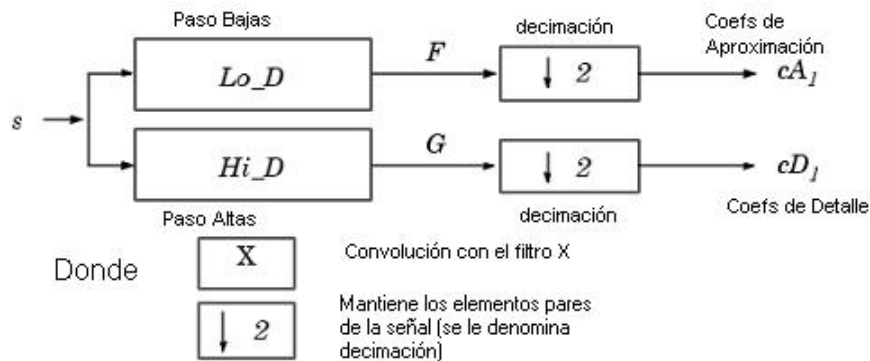


Figura 2.4 Diagrama de flujo de la transformada de onduleta

Donde

$$cA_1 = D(LoD * s)$$

$$cD_1 = D(HiD * s)$$

El operador $D^{(*)}$ se refiere a la decimación de quitar todos los elementos pares de la señal

2.1.6 Onduletas en la detección de errores

La onduleta, como ya se ha mencionado es una descomposición que resulta útil cuando se desea analizar los instantes en los que una señal ha sufrido un cambio brusco [Frazier 1999], para ejemplificar ésto en la figura 2.5 tenemos una señal senoidal (2.5a) junto con la misma señal, pero que recibe una perturbación en un momento del tiempo (2.5b).

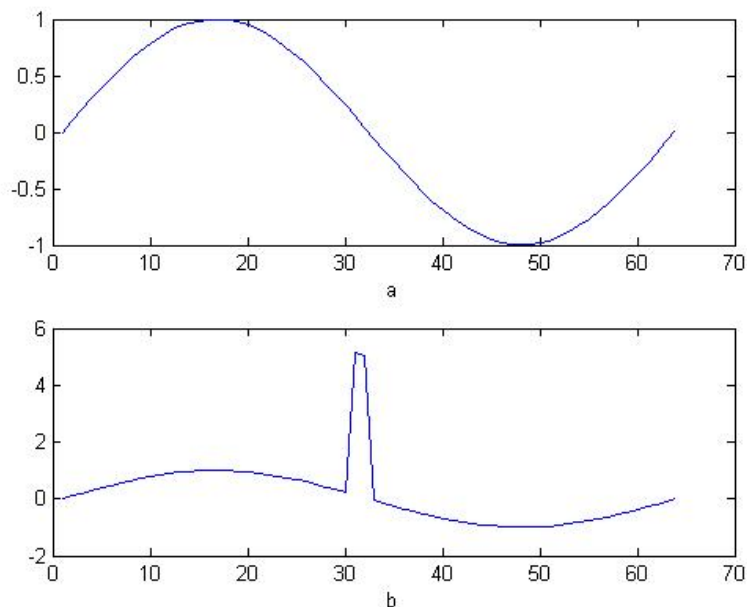


Figura 2.5 Señales senoidal normal y con una perturbación: a) Señal Senoidal y b) Señal con perturbación

Al transformar estas señales y realizar su descomposición con la transformada onduleta, obtenemos señales del siguiente tipo (figura 2.6), donde las señales debajo de las originales son las descomposiciones.

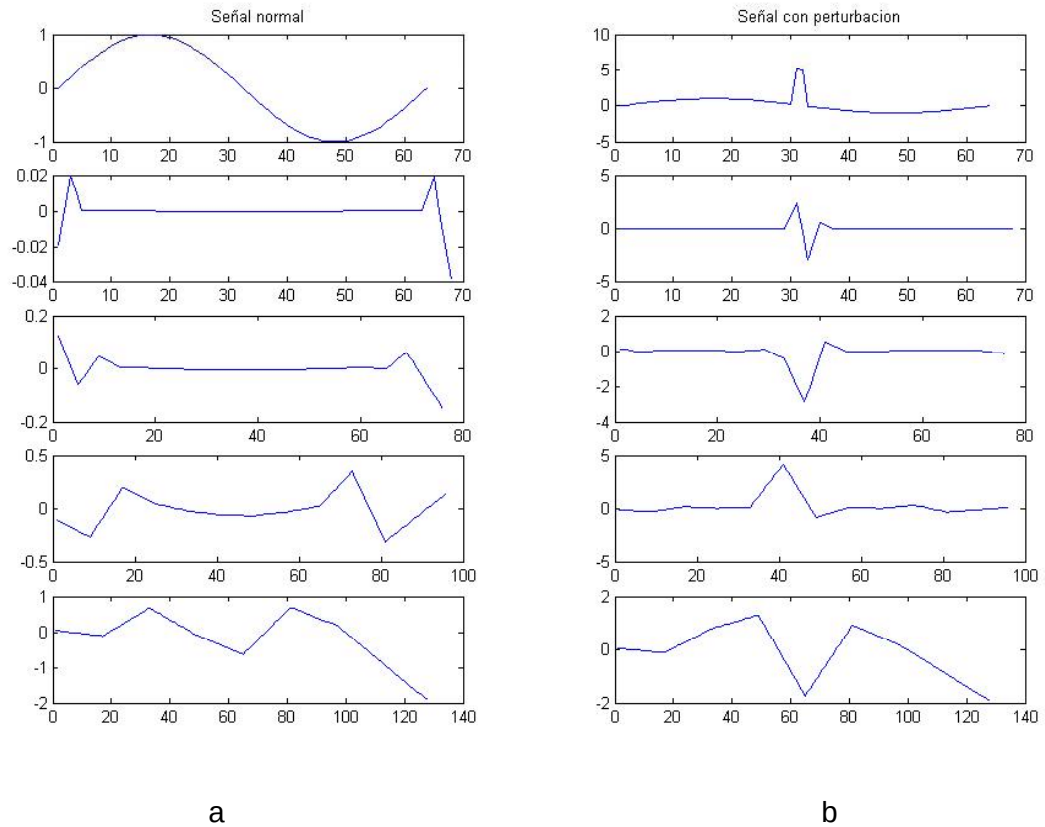


Figura 2.6 Señales y sus descomposiciones

Observamos en la figura 2.6b que la energía en la señal “b” se concentra donde está el cambio brusco (alrededor del valor 30), observamos con esta señal entonces que las onduletas son una gran herramienta para la detección de errores, logrando así tratar vectores donde toda la energía se concentra en las fallas y con esta información categorizar las mismas con alguna herramienta de organización.

2.2 Redes Neuronales

2.2.1 Introducción

En 1943, empezó el furor por las redes neuronales, presentadas por McCulloch y Pitts. Estas redes presentaron un modelo artificial de la neurona, como componentes conceptuales capaces de realizar diversas tareas en un circuito.

En 1969, Minsky y Papert publicaron un libro denominado *Perceptrón* [Minsky 1969], que pese a lo que se pudiera pensar no hacía mas que demostrar las deficiencias que tenía este componente. Esto causo que la mayoría de los científicos abandonaran el campo, y las redes se encontraran en un abandono total, y no fue sino años después, a los principios de los 80, que gracias a los esfuerzos de algunos teóricos como Teuvo Kohonen, Stephen Grossberg, James Anderson, y Kunihiko Fukushima que las redes neuronales tomaron un nuevo auge, debido a que estos personajes lograron realizar diversas aplicaciones practicas que utilizaban las redes neuronales. [Freeman, 1992]

Las redes neuronales se definen como modelos computacionales, que tiene como características las de aprender, generalizar , empaquetar y organizar información, mediante unidades básicas, en un procesamiento paralelo. [Fausett, 1994]

2.2.2 Disposición básica de una neurona

Una neurona es un elemento biológico, que entra dentro de la categoría de célula, la neurona es la encargada de transmitir la información al cerebro, en forma de impulsos eléctricos.

El ser humano posee por naturaleza propia, un pensamiento donde las neuronas trabajan en paralelo para procesar cada impulso que recibe del exterior, de cualquiera de las entradas sensoriales, como puede ser la vista, el oído, el tacto, el gusto y el olfato, cada uno de estos sentidos tienen neuronas específicas que se encargan de traducir todas estas señales del mundo externo en impulsos eléctricos.

La neurona cuenta con muchas partes, pero debido a la naturaleza de este trabajo, sólo nos enfocaremos en las principales; las dendritas, los axones y el cuerpo o soma, dichas partes son mostradas en la figura 2.7.

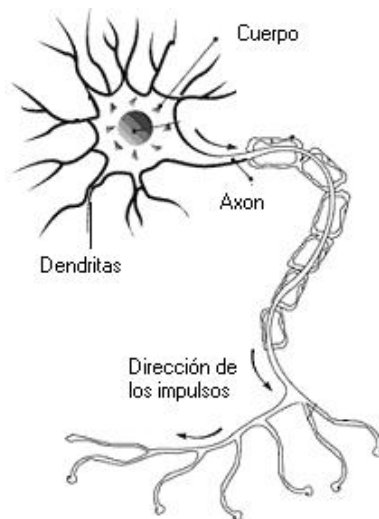


Figura 2.7 Disposición básica de una neurona

Las dendritas son las encargadas de recibir la información y enviarla al núcleo, se realiza algún proceso en el mismo y después el resultado se envía a las demás neuronas por medio del axón. Cada dendrita físicamente tiene pequeños componentes microscópicos denominados neurotransmisores, éstos se van modificando a sí mismos dependiendo del tipo de impulso que se transmita por esta dendrita, si es un impulso que está asociado a la neurona, cambiarían de tal manera que el pulso tenga la mayor energía posible para entrar a la misma, si es un pulso que no está realmente asociado con los procesos que la neurona tiene asociados, las dendritas cambiarían para que este pulso tenga poca energía al momento de entrar. [Freeman, 1992]

Una fortaleza muy grande del pensamiento neuronal es su gran robustez ante el daño, esto significa que si se muere una neurona, la reacción a las excitaciones no varía mucho, por no decir que nada, esto considerando que tenemos aproximadamente 10^{12} neuronas en el cerebro, número que varía de autor a autor, pero al parecer es el más constante en la bibliografía.

El modelo artificial de la neurona es un modelo muy sencillo de entender, una salida, y varias entradas. Cada una de las entradas se ve afectada por una variable, que denominaremos pesos.

2.2.3 Funcionamiento de la neurona artificial

Una de las principales funciones de la neurona artificial, si no es que la principal es la de categorización de elementos, y como con diversos arreglos se puede realizar no sólo una identificación, si no también la identificación de objetos conociendo sólo poca información de estos.

Son varias las operaciones que se pueden realizar dentro de la neurona, pero la más usada es el producto interno entre las entradas y los pesos de la neurona. Que en realidad no nos permite categorizar la entrada, por lo que recurrimos a la denominada función de activación, una función que se aplicará al resultado del producto interno para acotar el resultado a un margen deseado. Gráficamente se expresa en la figura 2.8.

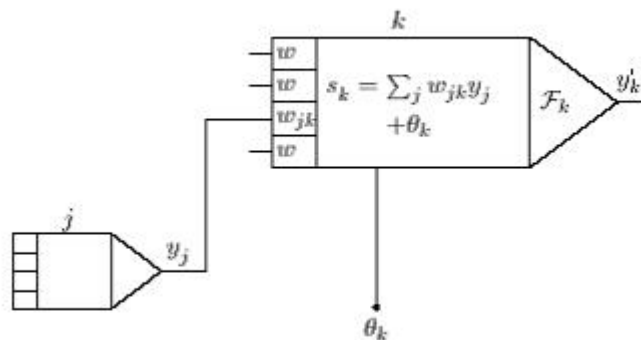


Figura 2.8 Esquema de la neurona artificial

Matemáticamente se define de la siguiente manera:

Tenemos un vector de pesos w_k definido de la siguiente manera:

$$w_k = (w_1, w_2, \dots, w_j, \dots, w_n)$$

donde cada uno asocia a la neurona k con alguna otra neurona

A su vez, también tenemos un vector de entradas, que se asocia a la salida de cada neurona conectada a la neurona k

$$y_k = (y_{1k}, y_{2k}, \dots, y_{jk}, \dots, y_{nk})$$

Para una neurona k que tiene n entradas, definimos como una de estas a la salida de otra neurona j , de tal manera que w_{jk} se refiera al peso que conecta a la neurona j con la neurona k

Para realizar el procesamiento, como ya se mencionó, se realiza el producto interno entre estos 2 vectores.

$$s'_k = w_1 y_1 + w_2 y_2 + \dots + w_n y_n = \sum_{i=1}^n w_i y_{ik} \dots \dots \dots (2.6)$$

Después de realizar el procesamiento, se fija el limite de activación denominado factor de umbral. Este factor lo denominaremos θ_k que será el factor de activación para la neurona k . De esta forma tenemos la siguiente ecuación:

$$s_k = w_1 y_1 + w_2 y_2 + \dots + w_n y_n + \theta_k = \sum_{i=1}^n w_i y_{ik} + \theta_k \dots \dots \dots (2.7)$$

A continuación se le debe aplicar la llamada función de activación, definiendo la forma de la salida de la neurona:

$$y'(k) = F(s_k) \dots \dots \dots (2.8)$$

En muchos casos se aplica simplemente la función escalón, es decir si la función es negativa, la salida es 0, y si es positiva es uno. Este escalón se puede mover con lo que denominamos el umbral de la neurona. Este valor nos permite variar la “sensibilidad” de nuestra neurona a las entradas externas. Otra función muy común es la función sigmoide cuya ecuación es:

$$y'_k = F(s_k) = \frac{1}{1 + e^{-s_k}} \dots \dots \dots (2.9)$$

Si se hacen pruebas se graficará con la forma de una “s” alargada y si a esta función se le hace la siguiente modificación:

$$y'_k = F(s_k) = \frac{1}{1 + e^{-\alpha s_k}} \dots\dots\dots(2.10)$$

La curva cambiará su forma para distintos valores de alfa, e inclusive en un caso extremo se parecerá prácticamente a la función escalón.

Algunas funciones de activación son las siguientes:

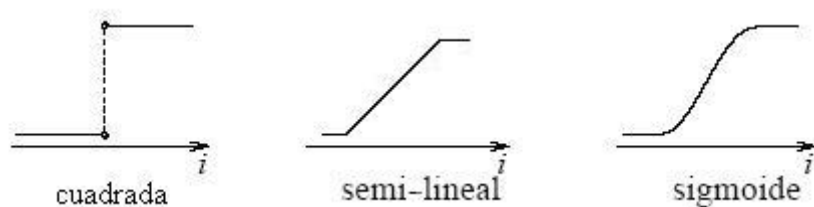


Figura 2.9 Distintas funciones de activación

Algunas se utilizan para aplicaciones en especial, pero las mas usadas son la sigmoide y por simplicidad el escalón.

Para poder hacer que una neurona aprenda algún comportamiento, lo que se debe hacer es variar los pesos de la misma, de tal manera que las operaciones matemáticas dentro de ésta den las salidas deseadas a las entradas, para esto existen diversos métodos de entrenamiento denominados como aprendizaje supervisado y aprendizaje no supervisado. [Freeman 1992]

2.2.4 Mapas Autoorganizados

Existen diversas arquitecturas de redes neuronales, como el perceptrón multicapa o los mapas de Hopfield, pero la que nos concierne en este momento es la denominada mapeos autoorganizados

De acuerdo a las evidencias de la ciencia, se dice que el cerebro humano se divide en zonas, cada zona esta enfocada a algún proceso dentro del cuerpo humano. Bajo esta

teoría existe una zona dedicada únicamente a la audición, y sus neuronas están entrenadas para ser especialmente sensibles y reaccionar a impulsos eléctricos que se asocian a las entradas recibidas por los oídos, y así para las diferentes funciones sensoriales del cuerpo humano existirá una zona dedicada.

En 1982, Teuvo Kohonen desarrolló una arquitectura de redes neuronales que trataba de hacer una analogía con esta organización del cerebro, el resultado fueron los Mapas Autoorganizados, que reciben su nombre del hecho de que es una arquitectura generalmente representada en un mapa de $N \times M$ neuronas (figura 2.10) y que no requiere de una supervisión externa para su aprendizaje, es una arquitectura de aprendizaje no supervisado. [Freeman,1992]. Esta arquitectura tiene como principal característica que puede mapear en un espacio bidimensional a un espacio de N dimensiones, logrando así una visualización gráfica de la similitud de elementos dentro de un espacio de 2 dimensiones.

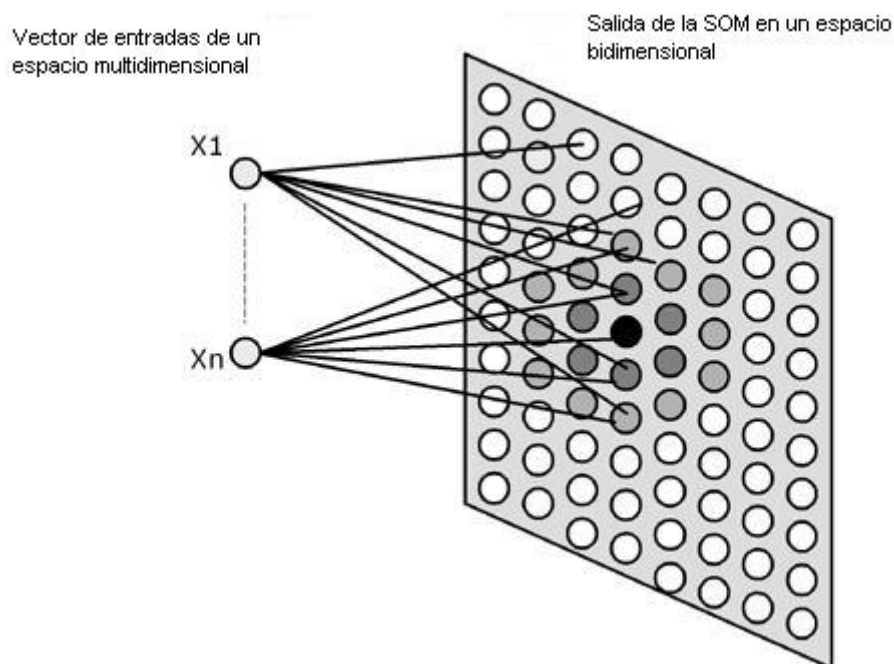


Figura 2.10 Representación grafica de una SOM

Esta arquitectura trata de actualizar los pesos de las neuronas dependiendo de las diversas entradas, de tal manera que en el mapa se vayan creando zonas que respondan a estímulos similares, o de la misma naturaleza. Es decir, en cada zona estarán neuronas cuyos vectores de pesos sean semejantes, y cuyas salidas respondan de manera similar a excitaciones similares. [Kohonen 1994].

Para realizar los mapas, se hace uso de la teoría de la neurona ganadora, consiste en dar una entrada igual a cada neurona del mapa, y la que tenga una salida más alta con respecto de las demás será la neurona ganadora. Para determinar la neurona ganadora se mide la distancia del vector de pesos con el vector de entradas correspondiente. Se usan diversas métricas, pero la más usual es la distancia euclidiana, que se presenta en la ecuación (2.11):

Para un vector de entradas x de longitud n , la distancia entre este y una neurona de coordenadas (i,j) esta dada por:

$$d^2(w_{ij}, x) = \sum_{k=1}^n (w_{ijk} - x_k)^2 \dots\dots\dots(2.11)$$

Aquella neurona cuya distancia sea menor al vector de entrada será la ganadora. Los pesos de esta se actualizan, buscando que la distancia sea aun menor. La actualización de pesos se puede hacer de muy diversas maneras, por conveniencia se usa la siguiente

$$w_{ij}(t+1) = w_{ij}(t) + \alpha(x - w_{ij}(t)) \dots\dots\dots(2.12)$$

donde α es el factor de aprendizaje, y puede ser variable en el tiempo, dependiendo de nuestras necesidades. De esta manera se observa que la tendencia de la red es que el vector de pesos sea lo más similar posible al de entradas.

La variación que otorga el mapa autoorganizado es el hecho de que no sólo actualiza una neurona, si no que actualiza una vecindad. Las neuronas circundantes a la ganadora tendrán un cierto parecido al vector de entradas, aunque sus respuestas no serán tan altas. Para conseguirlo, basta con aumentar un factor de vecindad h en el segundo término del sumando de la ecuación 2.12.

$$w_i(t+1) = w_i(t) + \alpha h_{ij}(x - w_i(t)) \dots\dots\dots(2.13)$$

Este factor tiene diversas representaciones, la mas sencilla es declararlo 1 si la neurona esta en la vecindad y 0 si no lo esta, y otras como el sombrero mexicano, o la curva normal gaussiana.

Para el desarrollo de la tesis se utiliza esta última, cuya expresión matemática es la siguiente:

Para una neurona de coordenadas i y j , tenemos:

$$h = e^{\left(-\frac{(i-rw)^2 + (j-cw)^2}{\sigma^2} \right)} \dots\dots\dots(2.14)$$

donde rw y cw se refieren al renglón y columna de la neurona ganadora. Y σ es la varianza de la curva normal que va a limitar a la vecindad

Un problema importante del mapa autoorganizado es el hecho de que conforme se elevan el número de neuronas el poder computacional requerido para el entrenamiento y la evaluación se va elevando cuadráticamente, y al mismo tiempo, la cantidad de patrones que la SOM puede almacenar depende de la cantidad de neuronas que ésta tenga, si queremos categorizar una gran cantidad de patrones, debemos tener una gran cantidad de neuronas, haciendo procesos muy lentos.

2.3 Detección de errores

2.3.1 Introducción

Para mantener un alto nivel de seguridad, desarrollo y confiabilidad en los procesos controlados, es importante que los errores de sistema, falla de componentes y operaciones anormales del sistema sean detectadas oportunamente y que la fuente y severidad de cada disfunción sea diagnosticada para que una acción correctiva pueda ser llevada a cabo. Un operador humano puede corregir anomalías del sistema, como por ejemplo cerrando una válvula o terminando un proceso. Por el otro lado, una reconfiguración automática de la planta o una adaptabilidad propia del sistema de control pueden ser aceptables, pero con una autoridad limitada. Esto debido a un problema que comentaremos mas adelante.

Las fallas en el equipo o instrumentos, o en el proceso en si, se reflejan en productos que no cumplan las especificaciones, aumento en los costos, la posibilidad de que la línea cese de operar, e inclusive, algún impacto ambiental. Por esto, la pronta detección de fallas y diagnósticos de disfunciones del sistema son estratégicamente importantes, debido a las demandas económicas y ambientales que exigen las compañías para poder seguir compitiendo en el mercado. [Patton 2000]

Dentro de estos ambientes automatizados que mencionamos, ya observamos que algoritmos capaces de detectar y clasificar funcionamiento anormal en la planta son críticamente importantes desde el punto de vista económico, e inclusive de seguridad, por las razones expuestas anteriormente. Estos algoritmos utilizan información extraída de análisis previos de la planta para poder predecir el funcionamiento cuando sólo datos nuevos estan entrando a la planta. Estas tareas se denominan detección de errores. Claramente esto tiene similitudes con otros campos, como los son el comercio electrónico (detección de fraudes), seguridad en redes (detección de intrusos) y comunicaciones inalámbricas (detección de señales), en donde en cada uno de ellos lo indicado representa una anomalía para el funcionamiento usual del sistema. Entonces no es de sorprender que algoritmos de cada una de estas áreas puedan resultar muy útiles para la detección de errores [Hrishikesh 1997].

La actividad de detección de errores junto con el diagnostico de fallas y la optimización de procesos caen en la categoría de las aplicaciones de la detección de patrones. Normalmente realizadas por un operador humano basados en observaciones a lo largo del proceso. Dicha tarea ha sido históricamente relegada a los humanos, debido a lo complicado que resulta la automatización de la misma. Esto representa hoy en día un cuello de botella en lo que se denomina la tecnología de la “manufacturación inteligente”, usada en los procesos industriales. El cuello de la botella, desde un punto de vista de tecnología de la información, esta representado en la imposibilidad de convertir los datos numéricos de los sensores en información simbólica útil. Se requiere de un interprete que le “diga” a la computadora el estado actual del sistema o del producto, con base a lo medido, de la misma forma en la que el cerebro interpreta las señales que obtenemos por los medios sensoriales,[S. Lee et al 2003].

2.3.2 Modelos en la detección de errores

La principal tarea de la detección de errores es realizar una interpretación confiable de la planta, y lograr que a cada estado inesperado de la misma, el controlador reaccione de una manera satisfactoria, para de esta forma, correr el proceso automáticamente sin la intervención humana, reduciendo así costos y tiempos de ejecución.

Para realizar la detección de errores existe 2 vertientes principales:

- Los modelos cuantitativos (ecuaciones diferenciales, variables de estado, funciones de transferencia, etc). Estos utilizan primordialmente resultados que

se obtienen del campo de la teoría del control, que se basan en conceptos de la estimación de parámetros, estimación de estados y paridad de espacios. La filosofía de esta aproximación es que una falla repercutirá en un cambio en algún parámetro físico y en las medidas, que al mismo tiempo conducirá a un cambio en ciertos parámetros o estados del modelo. Monitoreando los estados o parámetros estimados, es posible detectar y aislar cada falla. Con esta aproximación, es esencial tener información *a priori* acerca de la relación del sistema y sus fallas y sus parámetros o estados, es decir, cómo las fallas van a afectar al sistema en realidad, para poder categorizar a cada una que se presente. Sin embargo, los modelos teóricos de sistemas complejos son difíciles de obtener y en algunas ocasiones imposibles de derivar. Los modelos matemáticos llegan a consumir enormes cantidades de tiempo y en raras ocasiones logran representar fielmente cada etapa del sistema

- La segunda aproximación se basa en el campo de la inteligencia artificial (redes neuronales , lógica difusa). Hay en la actualidad una gran cantidad de trabajos en la detección de fallas utilizando estas técnicas, basadas en razonamiento cualitativo y modelado cualitativo. Para la simulación cualitativa se utilizan modelos cualitativos de los procesos para predecir su comportamiento operando en condiciones normales y así también durante varias fallas. La detección de fallas es comparar el comportamiento predicho con las observaciones actuales. Estos métodos en general son muy atractivos ya que no requieren modelos matemáticos explícitos de la planta que se está modelando, estas técnicas generan modelos implícitos de la planta, es decir modelos que se basan en los datos obtenidos anteriormente de la misma. Como ya lo vimos con las redes neuronales, la información del sistema se mantiene almacenada en la forma de pesos neuronales. Y con la lógica difusa, tenemos una aproximación más cercana a la teoría de conjuntos, pero con alguna variantes.

2.3.3 Algoritmo de detección de falla

El procedimiento de detección de fallas está determinado por el momento en que ocurre una falla. La detección oportuna es acompañada de alarmas de emergencia que pueden ser de distintos tipos (sonoras, luminosas, etc), que se activan al presentarse una falla para prevenir al personal operativo de la planta y en algunas ocasiones, permiten disminuir el riesgo de un daño severo en el proceso o en el sistema. Como se mencionó, la tarea de detección consiste en identificar las variables de observación más relevantes que nos ayuden a diagnosticar la falla. El propósito de este procedimiento es enfocar la atención de los operadores e ingenieros de la planta sobre los subsistemas pertinentes para el diagnóstico de la falla, de tal manera que el efecto de la falla pueda ser eliminado de una manera más eficiente. [Isermann, 1984].

El procedimiento de restauración del proceso también llamado de intervención, tiene la tarea de eliminar los efectos de la falla y es el esquema que cierra el lazo en el esquema o diagrama de flujo que sigue el monitoreo de procesos, ver fig 2.11. Cada vez que una falla es detectada, se realizarán los 4 procedimientos para el correcto tratamiento de la misma, en caso contrario, el proceso simplemente estará en el lazo cerrado de detección de falla todo el tiempo.

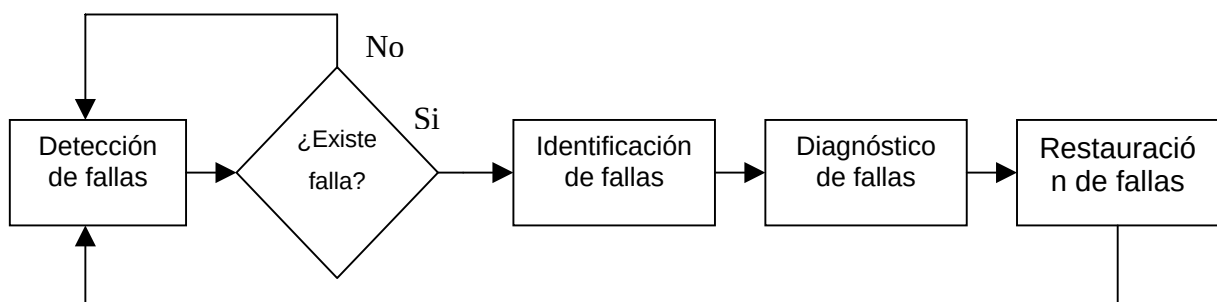


Figura 2.11 Proceso de detección y corrección de fallas

El proceso por completo no necesita ser automatizado, además como ya se mencionó se considera un problema de dificultad, así que en muchos casos, el único procedimiento que se automatiza es el de detección de la falla, ya que éste es más que necesario para poder dar indicadores a los operadores sobre las acciones preventivas que deben de tomar.

Podemos decir que los sistemas de detección e identificación de fallas (FDI) son sistemas diseñados para la supervisión y monitoreo de procesos capaces de localizar e identificar fallas automáticamente.

2.3.4 Detección de fallas usando mapas autoorganizados

La mayoría de los procesos industriales de gran escala en el mundo real, no están precisamente definidos dentro del espacio de medición sensible. Dentro de una región definida, cualquier proceso puede seguir uno de los muchos caminos, dependiendo de un gran número de factores desconocidos [Hrishikesh et al 1996]. Pueden existir muchas regiones, posiblemente sin elementos en común entre ellas, debido a diversos factores como combinaciones de características de entrada, cambios en la naturaleza deseada de la salida, variaciones en las condiciones ambientales, etc. (Figura 1a.). Los modelos basados en la clusterización, (agrupación <<<<de elementos), aproximan estos modos de operación complejos y multivariables como regiones en el espacio sensible y no derivando una relación funcional precisa, y son entonces muy adecuados para el diagnóstico en procesos industriales. [Whiteley et al 1996], [Kavuri et al 1993]. La arquitectura de mapas autoorganizados nos ofrece una clusterización que nos representa en dimensiones menores todo el espacio de entrada. [Vesanto 2000]. Los procesos típicos del mundo real, por lo general, cambian su comportamiento de un régimen operativo a otro, explorando nuevos equilibrios. En respuesta a esto, los mapas autoorganizados se reentrenan para adaptarse a los cambios.

2.4 Conclusiones

En este capítulo se explican los principios básicos del análisis de onduletas, redes neuronales, mapas autoorganizados y detección de errores, y en cada apartado se resaltan las conveniencias de utilizar cada una de las herramientas para la implementación de una arquitectura que se encargue de la detección de fallas.

Las onduletas resultan ser una excelente herramienta para la detección de cambios bruscos en la señal, en el tiempo, y las redes neuronales, en especial la arquitectura de mapas autoorganizados, son bastante convenientes para realizar tareas de clasificación utilizando un aprendizaje no supervisado.

Finalmente se plantean los principios de la detección de errores para comprender la necesidad de un algoritmo como el que se presenta en el capítulo 3 para tratar de resolver este problema.

3.1 Unión de Onduletas y Mapas Autoorganizados.

Las redes neuronales son una herramienta para la categorización de un espacio de datos, sin embargo, las redes neuronales muestran una debilidad cuando se le alimenta con datos no contemplados durante el entrenamiento. Es decir, cuando la planta o el espacio de datos de entrada se comportan de una manera no prevista en el proceso de entrenamiento de la red. Cuando esto sucede, en el mejor de los casos, se espera obtener resultados coherentes, sin embargo, en muchos casos esto no sucede, debido a su principio heurístico de funcionamiento, que se basa en ordenar los datos de entrenamiento de una manera tal que estén más cerca los que se parecen. Esto no nos asegura que la red deba comportarse correctamente con datos no utilizados en su entrenamiento.

En los mapeos autoorganizados, cuando se le introduce al mapa un vector no dado en el entrenamiento, si es similar a algún patrón de entrenamiento, el mapa lo categorizara correctamente, sin embargo esto no es seguro. Esto hace necesario el desarrollo de una herramienta con la que podamos entrenar a la red no con patrones, sino con características de la señal, elementos que sean comunes para señales comunes, como son las herramientas que existen para descomposición del espectro en frecuencia, que para nuestro caso particular serán las onduletas.

La transformada de onduleta es una herramienta matemática de análisis espectral localizado en frecuencia y en tiempo, esta característica, nos permite analizar los puntos donde una señal se comporta de una manera distinta a lo precedente. Dicha cualidad nos otorga un análisis en varios niveles, siendo cada nivel un conjunto de características para esa señal, en un momento dado. Entonces, al descomponer la señal que obtenemos del caso de estudio en sus distintos niveles de detalles obtenemos una serie de características propias de ese error, características que serán muy similares con las del mismo error en una muestra totalmente distinta.

3.2 Diagrama de flujo del sistema

Primero definiremos un bloque denominado bloque de entrenamiento (figura 3.1) que es el bloque que se ejecuta fuera de línea, tanto para la señal sin falla como para la actualización de la red cuando se encuentra la falla.

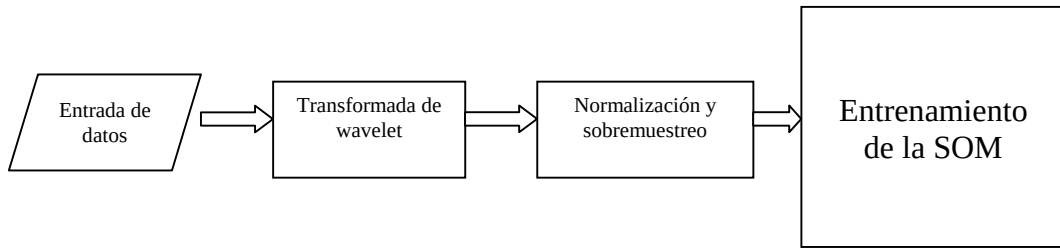


Figura 3.1 Bloque de entrenamiento(fuera de línea)

El sistema funcionando en línea entonces, sólo activa el bloque de entrenamiento cuando haya recibido un dato que no conocía, y entonces lo categoriza y lo aísla para su reconocimiento en futuras pasadas.

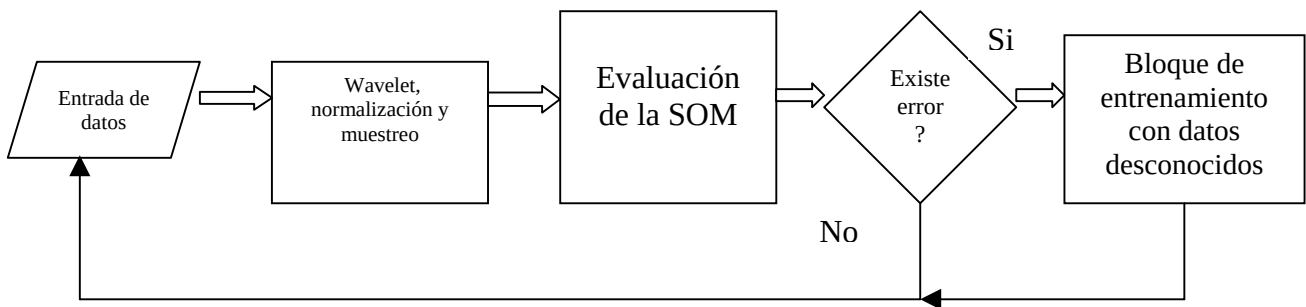


Figura 3.2 Sistema en línea

3.3 Metodología del algoritmo

A continuación detallaremos los pasos del algoritmo, se pueden apreciar claramente 4 etapas para el desarrollo del mismo:

- Datos de entrada
- Arquitectura Onduleta - Mapa autoorganizado
- Desarrollo en línea

3.3.1 Datos de entrada

Los datos de entrada para la transformada de onduleta, son vectores de longitud N, para el correcto procesamiento de la señal dentro de nuestra arquitectura, el vector es

preprocesado de tal manera que su longitud sea una potencia de dos a la entrada. Para este procesamiento, se hacen dos cosas: se encuentra la potencia de dos mas cercana a N hacia arriba, y sobremuestrear la señal, debido a que la energía perdida por un submuestreo de la señal podría alterar el experimento.

La señal con la que se trabajó, en algunos casos tiene una N muy grande, esto retrasa de manera considerable los tiempos de procesamiento, algo que como sabemos no es deseable en los sistemas de detección de fallas , donde se requiere una pronta detección y corrección de la misma De acuerdo a algunos experimentos, la detección se tarda hasta 15 minutos en ser notada con los datos sin comprimir, sin embargo se tarda tan sólo dos segundos con los datos comprimidos. Entonces, lo que se realiza es una reducción de la señal para trabajar constante a $N = 512$.

Otra solución para el problema del tamaño de la señal es el muestreo de la misma en segmentos, sin embargo, esto merece una mención especial.

Si se desea muestrear la señal, se debe tener mucho cuidado con la frecuencia de la misma en relación a la frecuencia de la falla. Como apreciamos en la figura 3.3, una falla en especial puede tener una frecuencia muy grande, y si la frecuencia de muestreo fuese mayor que ésta, como se observa (figura 3.4), la falla llega a perderse.

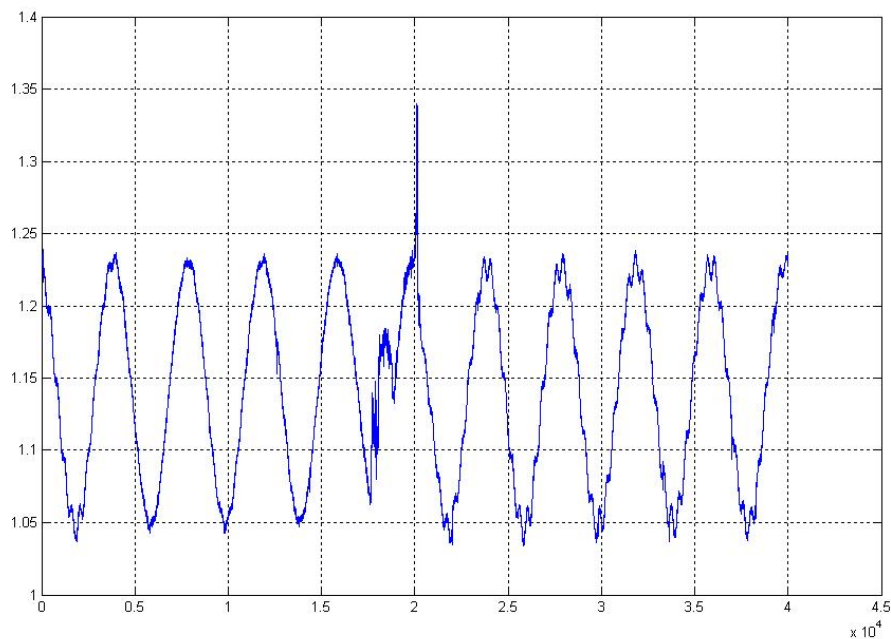


Figura 3.3 Falla a una frecuencia muy alta

Los datos de entrada en este caso no son señales completas, sino muestras de las mismas que se van tomando a lo largo del tiempo, y estas son las que se usan en todos los pasos que se describen más adelante.

Si realizamos un muestreo donde la longitud de la muestra sea menor a la de la falla, por ejemplo, ventanas de longitud de 1% de la señal original, tenemos señales de la siguiente forma (figura 3.4):

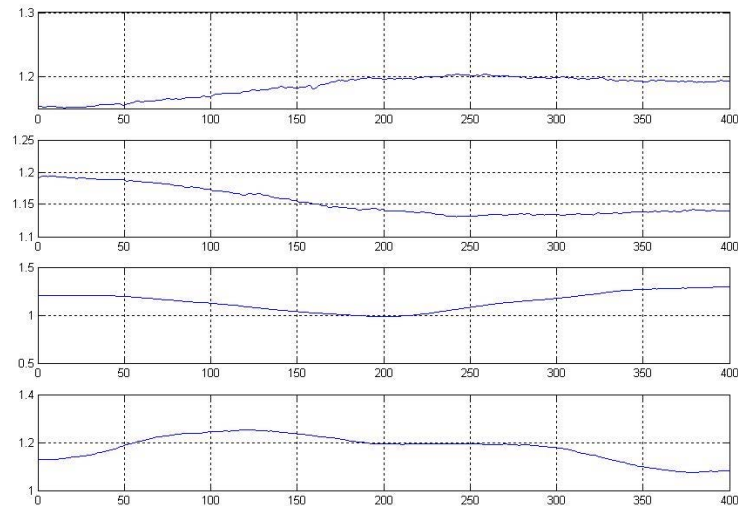


Figura 3.4 :4 Muestras de la falla mostrada en la figura 3.1

Estas muestras una vez normalizadas se asemejan a una señal sin falla, pero, las muestras inferiores, son las correspondientes a la falla, el sistema no reconocerá las fallas que se encuentran en la señal debido a la normalización, por lo cual se debe cuidar que la frecuencia de muestreo sea lo suficientemente chica, para que la falla sea reconocible.

En el capítulo 6 se darán los resultados para esta aproximación del algoritmo.

Como lo menciona Kohonen [Kohonen 1994], la señal de entrada debe ser acotada para asegurar el espacio que se genera en la SOM, entre el procesamiento de onduleta y el mapa se hace una normalización de los datos de entrada, así como un desplazamiento, de tal manera que todos los datos se localicen entre 0 y 1.

La normalización se realizó con respecto a la siguiente ecuación:

$$I = \frac{S}{\|S\|} \dots\dots\dots(3.1)$$

donde I es la señal que entra al mapa, S es el vector de datos de entrada de longitud N y

$$\|S\| = \sqrt{s_1^2 + s_2^2 + \dots + s_n^2} \dots\dots\dots(3.2)$$

3.3.2 Arquitectura Onduleta – SOM

Lo primero que se utiliza es la DWT, descomponiendo así la señal en 4 niveles, utilizando la onduleta “daubechies 2” [Daubechies ,1987], que después de algunas pruebas de desempeño mostradas en el capítulo 4, es la más adecuada para el caso de estudio. Se eligen 4 niveles, debido a que como se menciono antes, el trabajo de cómputo de un mapa autoorganizado se ve elevado exponencialmente conforme el número de patrones a entrenar van aumentando, además, una descomposición de 4 niveles presenta buenos elementos para categorizar una señal adecuadamente. Esto se da porque en el cuarto nivel, si es que existe un error en la señal, la mayor cantidad de la energía se ve enfocada hacia el mismo, dejando prácticamente como ceros aquellas señales donde no se presente un cambio drástico

Para el entrenamiento del mapa se utiliza un espacio de entrada de 2 elementos, 2 señales del sistema funcionando adecuadamente. Cabe recordar que cada una de éstas será expandida en 4 niveles lo que nos genera un espacio de 8 entradas al mapa.

Al entrar la señal a la arquitectura, lo primero que le sucede es que tiene 4 nodos que la descomponen en sus 4 niveles de onduleta, después de esto, cada uno de estos niveles es conectado exhaustivamente con cada una de las neuronas del mapa para realizar el entrenamiento.

Para cada nivel de detalle, existe una neurona ganadora, entonces para cada señal de entrada existirán 4 neuronas ganadoras, que se agruparan relativamente cerca una de las otras, dado su parecido en el espacio de la frecuencia. Para hacer la actualización de la vecindad se utiliza una vecindad de tipo Gaussiano, y dependiendo del caso de

estudio, se fija un valor para la vecindad, que se contempla estática, al contrario de lo que establece Kohonen [Kohonen 1994] que se debe de estar variando para obtener un mapa cuyas fronteras no sean muy distintas las unas de las otras, pero este efecto se espera de que no se están categorizando patrones si no características.

Después de pasar todas las señales de entrenamiento por la arquitectura, se vuelven a pasar, pero en esta ocasión una cantidad mayor de veces y con una varianza muy chica, esto esperando simplemente fortalecer la neurona ganadora de cada zona, para que no exista lugar a errores. [Kohonen 1994]

Este último pasó exige computacionalmente hablando una gran cantidad de tiempo de máquina debido a que la carga numérica para reforzar los datos adecuadamente requiere demasiadas iteraciones del sistema. Esto último entonces muestra que el sistema aumenta mucho sus tiempos de procesamiento al aumentar el número de neuronas o el número de señales a entrenar.

3.3.3 Desarrollo en línea

Después de obtener un mapa entrenado con las características obtenidas de nuestra herramienta de análisis de elementos de frecuencia, se maneja el sistema en línea simplemente utilizando la primera descomposición de onduleta de la señal, esto con el fin de ahorrar tiempo, y también para prevenir el caso de que la señal pueda contener ruido del exterior, ya que la señal reconstruida con onduleta, como ya se mencionó, funciona como un efectivo filtro. Después de probar constantemente con los 4 niveles de la transformada en distintas muestras y valorando las categorizaciones realizadas con cada nivel por medios cualitativos. Se llega a la conclusión de que es suficiente con la primera descomposición de la señal para realizar una correcta categorización de la misma. Ya que en esta descomposición, como se muestra en la figura 3.5, hay suficiente información de la diferencia de una señal sin perturbaciones contra otra con alguna perturbación.

El sistema se prueba antes de correrse en línea mediante una evaluación para ver si responde correctamente a señales previamente entrenadas. En este caso será la señal sin perturbaciones la que se utiliza como prueba.

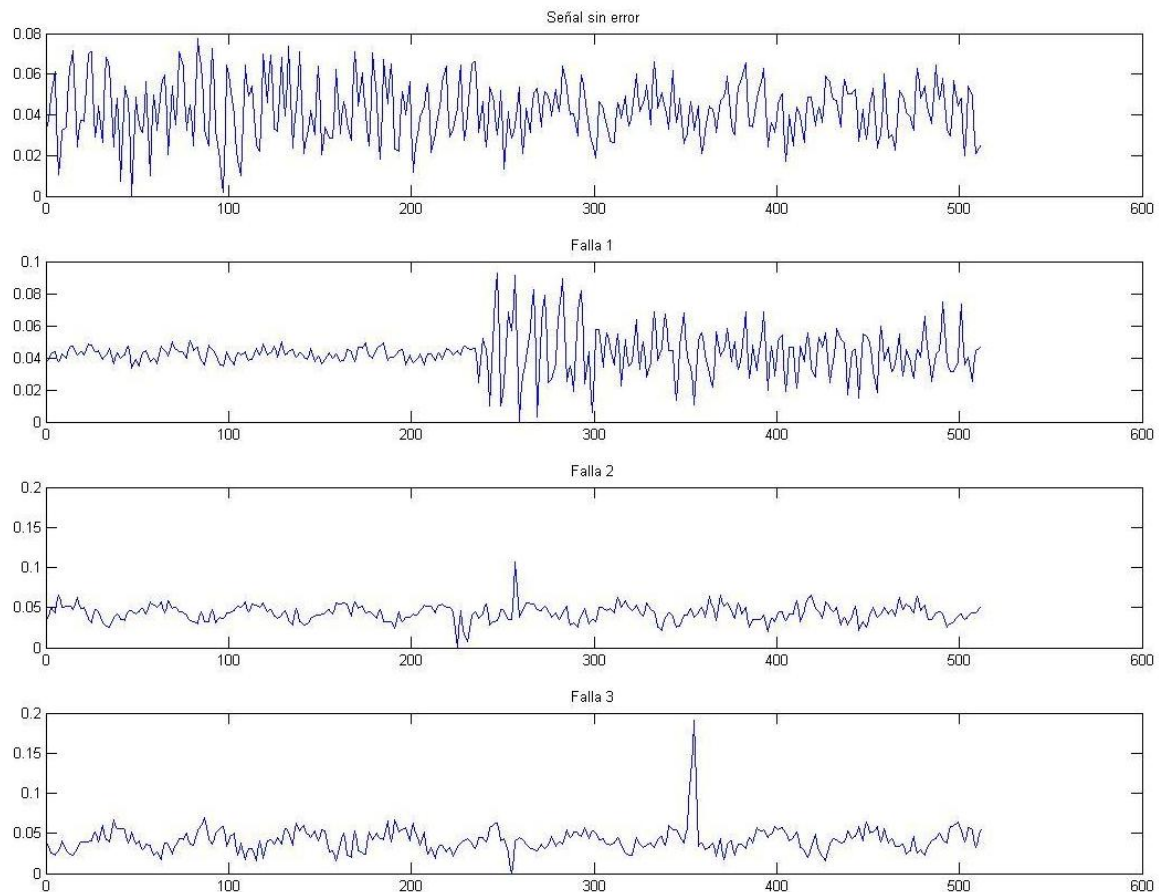


Figura 3.5 Descomposición en 1er nivel de onduletas de una señal normal y sus errores

Con la señales pasando por la arquitectura, se detectara en que momento esta señal no corresponde con los patrones de entrada dados, esto se hace mediante una comparación de la neurona resultante de una evaluación de la señal que esta entrando contra todo el arreglo de neuronas que el sistema ya tiene identificadas, al no encontrar que la señal pertenezca a una zona, quiere decir que se ha presentado un error o un comportamiento imprevisto del sistema, entonces se saca a la arquitectura de línea y se le entrena con esta nueva entrada. Cabe mencionar que si se tienen una gran cantidad de entradas, posiblemente el tamaño del mapa resulte ineficiente para poder categorizar correctamente a todas, es por esto que se hicieron pruebas de desempeño detalladas en el capítulo 4 en el cual se vio que para 3 errores el tamaño ideal del mapa era de 10 x 10, sin embargo para más errores puede necesitarse de un mapa más grande.

El objetivo de esta detección es categorizar los errores en pequeños bloques de tal manera que errores próximos de características muy similares, sean catalogados en las mismas zonas del mapa que los errores previamente entrenados, para evitar así el tener que estar cambiando el tamaño de la red.

3.4 Conclusiones

El algoritmo presenta dos variantes, una tomando muestras de la señal, que presenta algunas dificultades, como el tamaño de la muestra, en comparación con la otra aproximación, en la cual tan sólo se toma la señal completa y se le hace un submuestreo. Como detalle, se debe tener en cuenta que en el sistema físico se debe realizar entonces un sobremuestreo para que al momento de realizar este procedimiento en el procesamiento de la señal, no se pierda energía importante y necesaria para la identificación de la falla.

La arquitectura tendrá dos pasos, uno en el cual se le entrena con la señal sin fallas, ya sea una o un conjunto de las mismas, esto para que el sistema sea capaz de detectar señales sin fallas. Existen muchas variables dentro del algoritmo que requieren especial atención, como son el tipo de onduleta a utilizar, el número de neuronas, y en la aproximación por muestreos, el tamaño de las muestras. Estas variables son discutidas en el siguiente capítulo.

4.1 Introducción

El sistema presentado en el capítulo anterior tiene una gran cantidad de variables que se pueden modificar para obtener distintos resultados, como son el tamaño de σ en la función normal, o el tipo de onduleta a utilizar, se llegó a la conclusión de que las más significativas eran 2, una el número de neuronas del Mapa y la otra el valor de σ en la función de vecindad del algoritmo. Se eligieron estas 2, ya que son las variables que repercuten directamente en la creación de las zonas de reconocimiento dentro de la red, para después evaluar en línea los resultados de la misma.

4.2 Número de neuronas

Como se vió en el capítulo 2 el número de neuronas de la SOM es un parámetro directamente proporcional al número de patrones a entrenar, el problema, es que al igual que para la mayoría de los modelos con redes neuronales, el número ideal de neuronas no es determinado exactamente por algún modelo matemático. En cambio, es una determinación que se debe llevar a cabo de una manera empírica, realizando experimentos para distintos números de neuronas en el mapa.

Cabe mencionar que los tamaños de las redes, por facilidad de cálculo y simpleza en el modelo fue determinado por arreglos cuadrados de $N \times N$ neuronas.

Bajo la filosofía previamente dicha, se realizan pruebas con diversos tamaños de redes. Debido a causas obvias, no se pueden utilizar tamaños relativamente chicos, es decir, se comenzó con un arreglo de 5×5 neuronas, debido a que arreglos más chicos podían llegar a mezclar muy fácilmente la información.

De acuerdo a lo que se expresó en el capítulo anterior, se busca un modelo donde se puedan entrenar nuevos patrones, una vez fija la red. Cada prueba se realiza de tal manera que se le entrenan patrones a la red y después se hace un reconocimiento en línea reentrenando los nuevos patrones que se pudiesen presentar, y después evaluar si los patrones anteriormente entrenados se ven o no afectados por este nuevo entrenamiento. Para esta comparación se utilizó un nivel de error para comparar la localización entre el patrón previamente entrenado y su nueva localización, así como otra evaluación con la cercanía y lejanía con el patrón mas reciente que se entrena. Estos errores son expresados de la siguiente manera:

$$E_{\text{renglón}} = \frac{|r_{\text{actual}} - r_{\text{anterior}}|}{r_{\text{anterior}}} \times 100 \dots\dots\dots(4.1)$$

$$E_{\text{columna}} = \frac{|c_{\text{actual}} - c_{\text{anterior}}|}{c_{\text{anterior}}} \times 100 \dots\dots\dots(4.2)$$

$$E_{\text{total}} = E_{\text{renglón}} + E_{\text{columna}} \dots\dots\dots(4.3)$$

$$Cercania = \sqrt{(r_{pn} - r_{pa})^2 + (c_{pn} - c_{pa})^2} \dots\dots\dots(4.4)$$

donde

r_{actual} = renglón actual

r_{anterior} = renglón anterior

c_{actual} = columna actual

c_{anterior} = columna anterior

r_{pn} = renglón patrón nuevo

r_{pa} = renglón patrón anterior

c_{pn} = columna patrón nuevo

c_{pa} = columna patrón anterior

Como se observa claramente, la cercanía no es mas que la distancia euclidiana en el espacio de R^2 en los números reales, esto debido a que el espacio en la red siempre fue evaluado bajo la misma métrica que los números reales, es decir el entrenamiento y la razón de distancia.

Como se mencionó se utilizaron arreglos a partir de 5x5 neuronas aumentando en cada evaluación un renglón y una columna, hasta lograr un resultado satisfactorio cuando se presentara un error nuevo dentro de la red previamente entrenada. (Tabla 4.1).

Arreglo	Cercanía	$E_{\text{renglón}}$	E_{columna}	E_{total}
5x5	2	0	66.6666667	66.6666667
6x6	2.236067977	20	66.6666667	86.6666667
7x7	2.236067977	20	40	60
8x8	1	20	0	20
9x9	2.236067977	16.6666667	40	56.6666667
10x10	0	0	0	0
11x11	0	0	0	0

Tabla 4.1 Evaluación del tamaño óptimo de la red

Se observa como después de cada iteración los errores van disminuyendo, de tal forma, como se menciono anteriormente, a un mapa más grande se pueden entrenar más parámetros sin afectar los previamente entrenados. Al final se determinó que el arreglo de 10x10 fue el que ofreció mayor flexibilidad a nuestra arquitectura, esto claro, teniendo una idea previa de los errores que se pueden presentar, y que sea razonable que los nuevos errores sean similares al menos en ciertos aspectos a los errores ya presentados, para que después de algunas actualizaciones, los nuevos errores sean localizados sin necesidad de un nuevo entrenamiento de la red.

4.3 Valor de σ

Una vez determinado el numero de neuronas es posible evaluar el valor de σ .

Como se mencionó en el capítulo anterior, la vecindad de la SOM fue definida por la función gaussiana, la cual tiene 2 variables principales, una es la media, y la otra es la desviación estándar (σ). Para el problema de la SOM, la media, en un espacio de 2 dimensiones, siempre será la neurona ganadora, como se observa en la figura 4.1 y la sigma será entonces un parámetro de vital importancia, ya que va a determinar que tan grande o que tan chica va a ser la vecindad de la red.

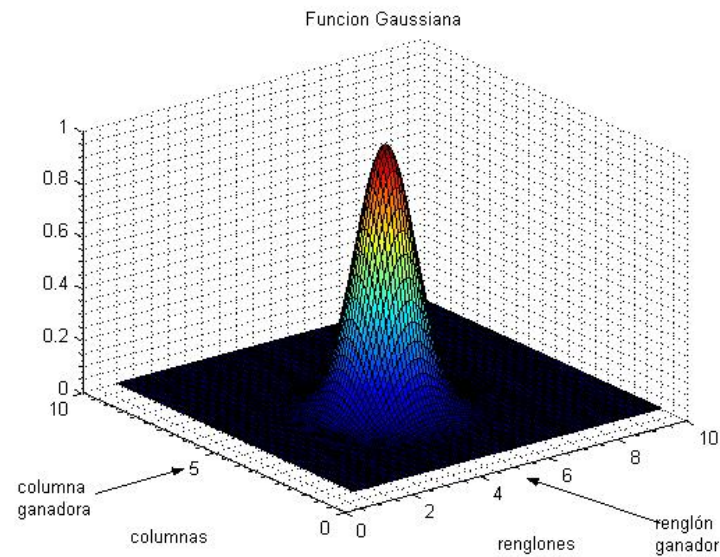


Figura 4.1 Representación gráfica de la vecindad de una neurona

De acuerdo a lo anterior, la sigma se comporta de manera exponencial al tamaño de la vecindad, es decir, a neuronas más lejanas el valor de la misma va a resultar menor, de acuerdo a la ecuación 2.14 significa que va a aprender menos que las neuronas con una σ grande. Así, dependiendo del tamaño de la red, será la magnitud de la σ a utilizar, gráficamente, para un arreglo de 10x10 neuronas, las siguientes son algunas de las sigmas posibles a utilizar.

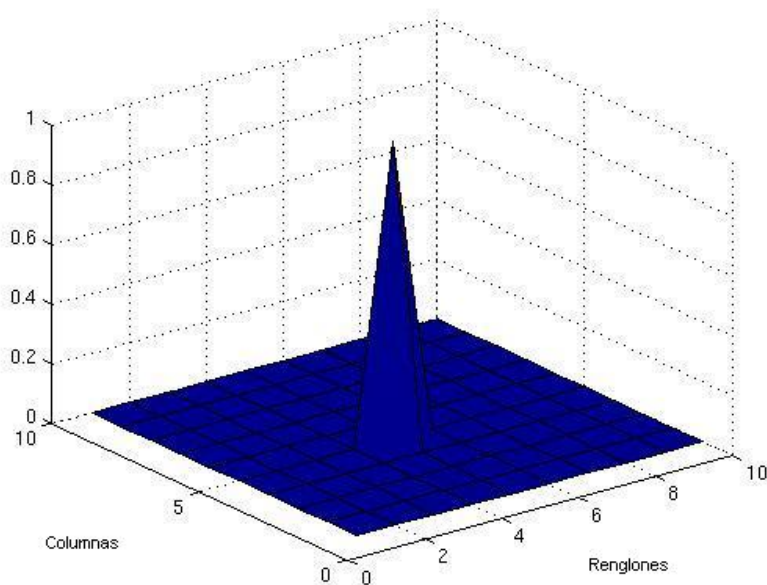


Figura 4.2 Vecindad Gaussiana con una $\sigma = 0.1$

Como podemos apreciar en la figura 4.2, con una σ pequeña, prácticamente no existe una vecindad, por que la función h (ecuación 2.13) que nos determina el tamaño de la vecindad va a tener toda su energía concentrada en ese valor. Esta aproximación se denomina el método de los vectores cuantizados, [Kohonen 1994] donde sólo se actualiza una neurona, la ganadora. Para nuestra aplicación requerimos que se generen zonas no puntos de reconocimiento, por lo que el valor de σ resulta poco útil.

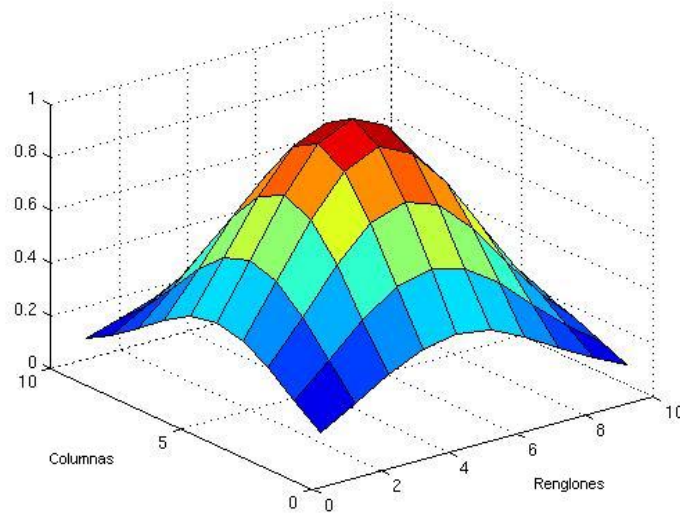


Figura 4.3 Vecindad Gaussiana con $\sigma = 4$

Al contrario como se denota en la figura 4.3 , para una σ grande, tenemos una actualización total de la red a cada patrón nuevo que se le introduce. Esto, tampoco es útil para nuestro algoritmo debido a que si obedecemos a esta σ , los únicos patrones entrenados serán los mas recientes, olvidando así por completo los patrones anteriores, debido a que la función h va a tener un valor muy grande y va a hacer que la ultima actualización sea la que vale para el sistema, por lo que una σ tan grande tampoco resulta útil para el algoritmo.

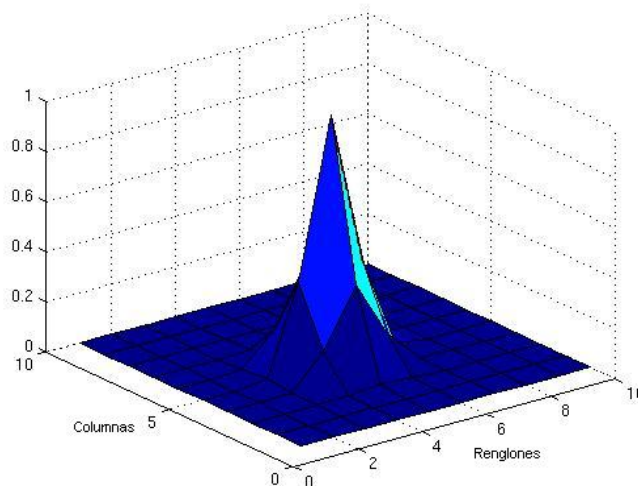


Figura 4.4 Vecindad Gaussiana con $\sigma = 1$

Observamos claramente en la figura 4.4 que una $\sigma = 1$ parece ser una buena opción, sin embargo, al no tener datos numéricos, no sabemos si existe un valor cercano que nos pueda servir mejor que ésta, se diseñó un experimento donde se varía el valor de la σ y se realiza el entrenamiento, realizado con varias señales de prueba, y al final, al igual que en el caso anterior, se logra un parámetro de error, que determina que tan cerca están a las zonas que les corresponden las señales entrenadas originalmente.

El parámetro de error se determina por la distancia euclidiana (ecuación 3.4) que se utiliza para determinar la cercanía entre neuronas en el apartado anterior. Esta distancia es calculada entre la señal entrenada y otra similar que se le introduce a la red para cada patrón entrenado, y después, se obtiene la distancia promedio, para tener un solo parámetro de efectividad (Eq 4.5). En la tabla 4.2 observamos las variaciones de esta distancia total promedio a diferentes valores de σ .

$$DTP = \frac{\sum D}{\# Patrones} \dots\dots\dots(4.5)$$

Donde DTP: Distancia total promedio

D : Distancia entre patrón entrenado y nuevo patrón

Y el número de patrones para nuestro caso particular fue de 5

Se elige este parámetro debido a que la mejor forma de evaluar la eficacia en un mapa autoorganizado es observando que tan bien definidos están las zonas entrenadas, y la

única manera de definir esto, es midiendo la cercanía entre neuronas no entrenadas que debieran pertenecer a una zona cercana y neuronas entrenadas pertenecientes a esa zona.

σ	Distancia total promedio
0.2	4.50783117
0.4	2.86305202
0.6	4.35453981
0.8	4.68328157
1.0	6.26146765
1.2	7.21052641
1.4	7.7692936
1.6	7.44547559
1.8	8.22347064
2.0	7.84328885

Tabla 4.2 Variación de la distancia total a diferentes sigmas

En la tabla observamos como a $\sigma \rightarrow 0$ hay más cercanía entre los patrones y sus distancias totales son menores, sin embargo, hay que cuidar no utilizar una σ muy pequeña para el tamaño de la red. Se utilizará la sigma de 0.8 que muestra un error pequeño, no el menor, sin embargo, cabe recordar que tan sólo se entrenaran 2 patrones y sus expansiones de onduletas, ayudando a generar zonas más grandes para cada señal nueva .

Otra forma para evaluar el mejor valor de σ es de manera cualitativa, observando los mapas que se forman con diversos valores de entrenamiento de σ , y elegir el mapa que presente una distribución más uniforme al momento de entrenar el patrón.

Con una sigma muy baja (figura 4.5) las zonas son totalmente puntuales al momento de ser entrenadas, no nos proporciona un mapeo conveniente debido a que toda la energía se encuentra concentrada en esos puntos. Con valores altos de la σ el mapeo no presenta mucha uniformidad en la distribución de los datos (figura 4.6), debido a que el área que se ve afectada en el entrenamiento del mapa es muy grande.

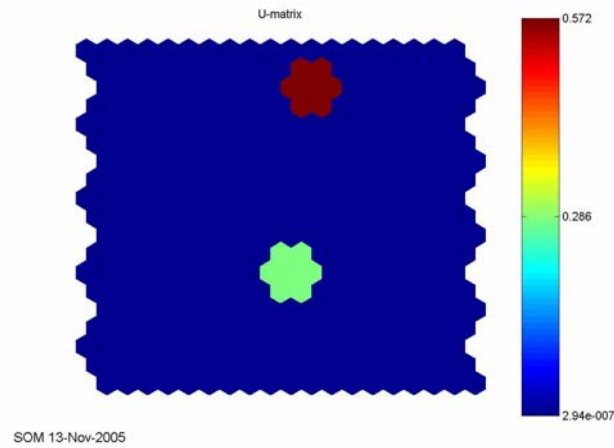


Figura 4.5 Mapa entrenado para una $\sigma = 0.1$

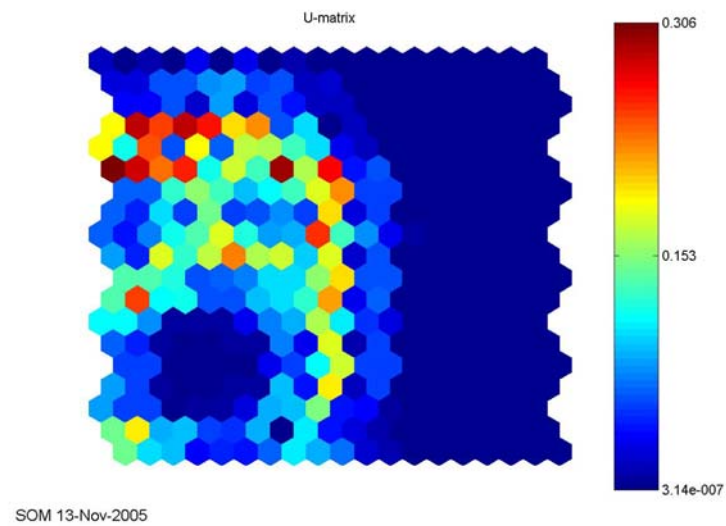


Figura 4.6 Mapa entrenado para una $\sigma = 1.6$

Para un valor medio de la σ (figura 4.7) observamos que los datos se encuentran acumulados de manera uniforme y cercana, estos valores oscilan entre 0.6 y 1.4, con estos resultados, el valor obtenido de σ mediante la evaluación anterior se ve reforzado.

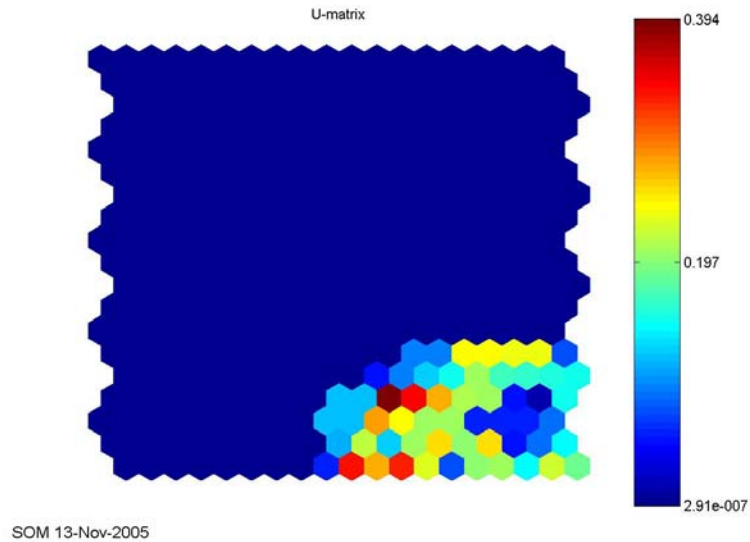


Figura 4.7 Mapa entrenado para una $\sigma = 0.8$

4.4 Tamaño de las muestras

Para determinar el mejor periodo de muestreo, se tuvo que recurrir a las señales de prueba, mostrando sus fallas en la figura 4.8. Como observamos la falla 2 tiene una frecuencia muy alta en comparación con la señal, mientras que la falla 3 es un desplazamiento de la misma, lo cual nos dice que el muestreo debe ser hecho de tal manera que esta sección sea comparable en magnitud con la señal original, debido a que si una muestra fuese totalmente de la sección con desplazamiento, al momento de realizar la normalización, parece una señal sin falla, y no se detectaría ningún problema.

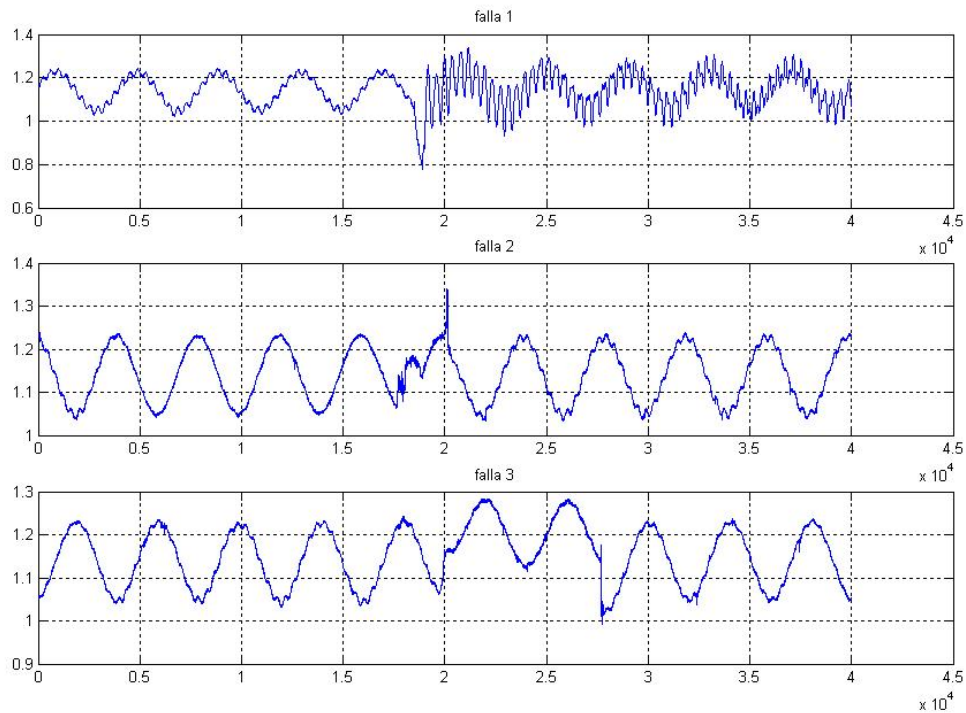


Figura 4.8 Señales con falla

Por lo anterior, elegimos una frecuencia de muestreo que sea adecuada para las 3 fallas, esto porque elegir distintas frecuencias de muestreo no resulta viable, por falta de información previa del sistema, entonces, la longitud de la sección con offset resulta de 21% la longitud de la señal original, debido a esto, y para facilitar el muestreo en secciones iguales se tomaran muestras de 25 % para que sí existe una falla como la 3, no se pierda debido al tamaño de la muestra.

Un muestreo de 25% implica 4 muestras por señal, en la siguiente sección compararemos esto con la aproximación de tomar la señal entera en cuestión de tiempos de procesamiento y factibilidad del sistema, por lo pronto, una ventaja de este procesamiento de la señal, es que se pierde menos información del sistema y no se requiere realizar un sobremuestreo de la señal original, ya que se puede trabajar perfectamente con los datos que se tienen.

4.5 Conclusiones

En conclusión, las variables en juego para el sistema como ya se mencionó son muchas, pero las más importantes las podemos reducir a 4 (en el caso de que se use la aproximación de muestras, de lo contrario serán 3), las cuales son tipo de onduleta,

tamaño de la vecindad, que se traduce en el valor de σ , tamaño del mapa y tamaño de la muestra.

La primer variable, debido a que sólo nos interesa la detección de la falla, no es de esencial importancia para el sistema, ya que los distintos tipos de transformada onduleta nos ofrecen distintos tipos de resolución, pero finalmente detectan una falla. Llegamos al resultado de que el valor óptimo de $\sigma = 0.8$, valor estático que demuestra un mejor desempeño para el aislamiento adecuado de señales en el mapa, y a su vez que el valor del mapa mínimo que obtiene mejores resultados fue de 10x10 neuronas, sin embargo es posible que el tamaño de la red pueda aumentar dependiendo de las demandas del sistema físico.

Por último el tamaño de la muestra que también depende del caso de estudio resulta ser de 25% del tamaño de la señal original, debido a que de usar un muestreo más chico se perderían las características esenciales de cada falla e inclusive en el peor de los casos la catalogaría como una señal sin falla debido a la normalización que se le hace a la señal.

5. Caso de estudio

5.1 Introducción

El algoritmo desarrollado puede ser aplicado a diversos sistemas, sin embargo, para cada uno de éstos se deben de tener consideraciones en particular, con respecto a las variables de la arquitectura que estamos implementando.

El caso de estudio es un levitador electromagnético, para controlar mediante diversos algoritmos, la posición de una esfera de metal inmersa dentro de un campo magnético, esto anipulando la variable física de la corriente eléctrica que se hace circular a través de la bobina generadora del campo.

5.2 Levitador electromagnético

El levitador electromagnético es un sistema dinámico sensible a perturbaciones del ambiente, este levitador tiene una interfaz para conectarse a la computadora, para el control del mismo y mantener la bola metálica siempre a flote.

El levitador magnético, mostrado en la figura 5.1 se divide en 3 secciones encerradas en el contenedor rectangular. La primera sección contiene un electromagneto formado por un solenoide con núcleo de acero. La siguiente sección consiste en una cámara interna donde la suspensión magnética toma lugar. Uno de los polos del electromagneto esta frente a la punta de un poste negro donde hay una esfera de una pulgada. La levitación de la bola es medida con un fotosensor embebido en el poste. El poste esta diseñado de tal forma que cuando la pelota se encuentra descansando, el espacio de aire entre ésta y el polo superior es de 14 mm. El poste también provee condiciones iniciales repetibles para el desarrollo del sistema de control. Finalmente la tercera parte del levitador contiene la circuitería necesaria para que el sistema funcione como por ejemplo el controlador de intensidad luminosa del sensor. [Manuales de Quanser]



Figura 5.1 Levitador Electromagnético

El experimento del levitador electromagnético no consiste simplemente en hacer flotar una pelota en el aire, sino diseñar un sistema de control que haga uso del electromagneto. El controlador puede seguir la posición de la bola en una trayectoria deseada.

Para realizar la conexión a una computadora de este sistema dinámico, son necesarios algunos componentes:

Amplificador de potencia: Quanser UPM 2405. (Figura 5.2)

Este amplificador es necesario para energizar todos los actuadores de la marca Quanser, el modelo utilizado tiene un voltaje máximo de salida de 24 Volts y una corriente continua máxima de salida de 5 Amperes, cuenta con una salida y su voltaje a la misma es lineal.

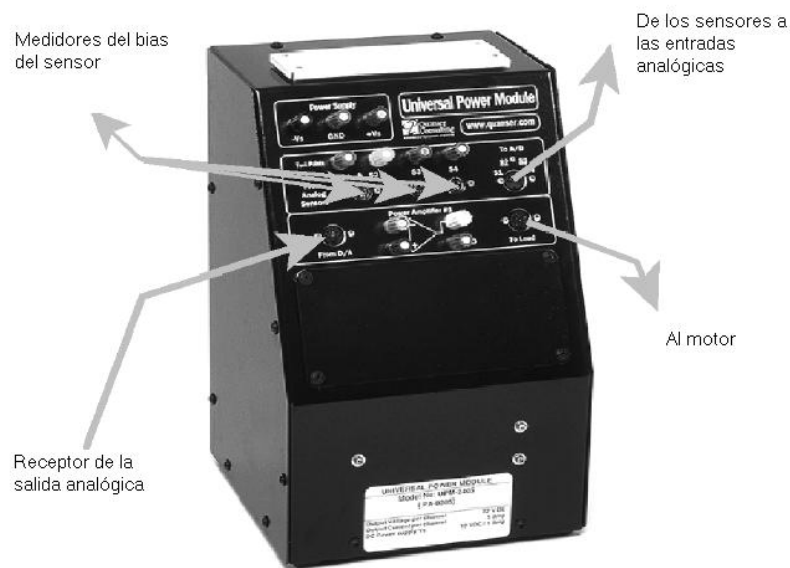
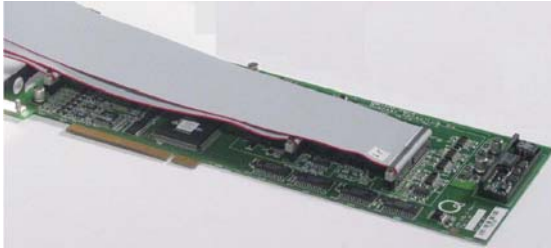


Figura 5.2 Módulo UPM

Tarjeta de adquisición de datos: Se utiliza la tarjeta Q8 de Quanser (Figura 5.3), con las siguientes características:

- Alta resolución – 14 bits de entrada
- Velocidad de muestreo de hasta 350 kHz
- Muestreo simultáneo de entradas A/D, digitales y decodificadores.
- Entradas analógicas programables de 8 x 14 bits
- 32 canales digitales programables de entrada-salida

Esta tarjeta consta de 2 partes: i) tarjeta de adquisición de datos que es instalada dentro de la computadora (figura 5.3a), donde se encuentran los procesadores y ii) tarjeta de terminales (figura 5.3b) donde se localizan los conectores para los controladores físicos.



(a)

Tarjeta de adquisición de datos



(b)

Tarjeta de terminales

Figura 5.3 Sistema de adquisición de datos

5.3 Conclusiones

Observamos que el levitador electromagnético es un sistema que nos permite controlar mediante una interfaz gráfica de usuario-máquina (como puede ser Matlab) el comportamiento del mismo, y realizar diversos experimentos de control y detección de fallas.

6.Resultados

6.1 Introducción

Los resultados presentados en este capítulo son los entregados al aplicar la herramienta a nuestro sistema físico, con las variables que ya habíamos determinado, $\sigma = 0.8$, mapa de 10x10 neuronas, onduleta “daubechies 2”. Cabe mencionar que la selección de las variables es dependiente del sistema y necesita realizarse previamente a la implementación del mismo.

Los resultados reportados fueron obtenidos utilizando una PC, con sistema operativo Windows XP Profesional, un procesador Intel Pentium 4, reloj de 2.66 Ghz y 512 M en memoria RAM.

6.2 Entrenamiento de la señal sin falla

La señal con la que se entrena a la red es la descomposición de onduleta (figura 6.1) de una señal sin falla que proviene del levitador electromagnético.

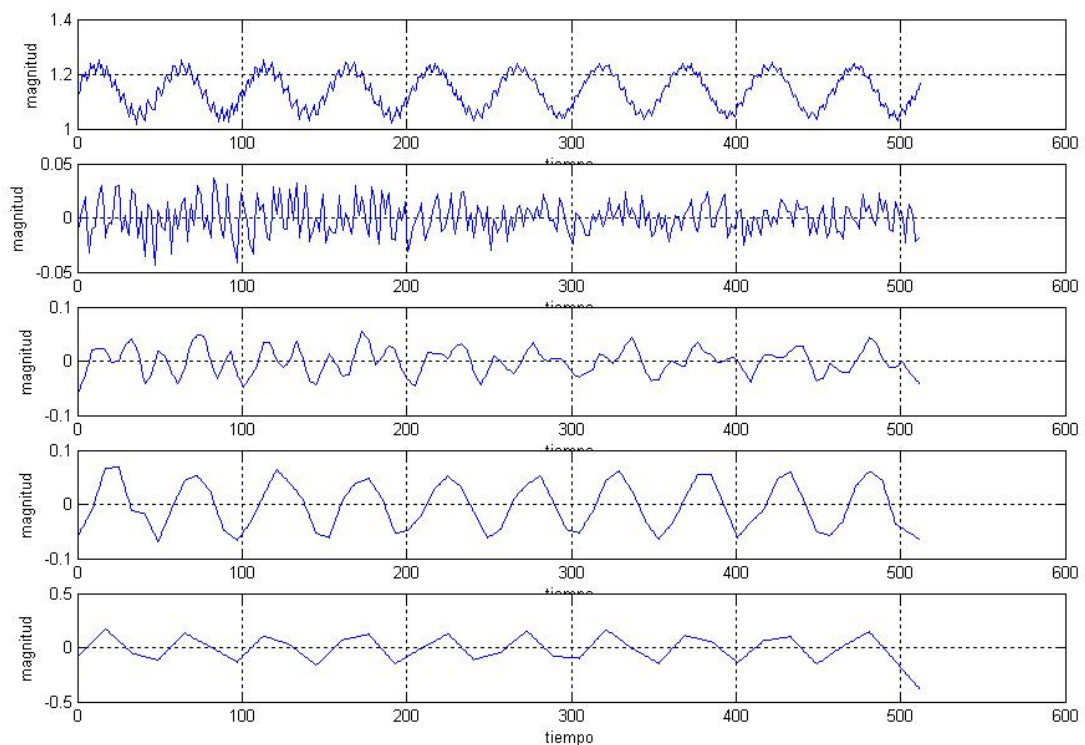


Figura 6.1 Señal sin falla con sus 4 descomposiciones de onduleta

Podemos observar en la figura 6.1 como las descomposiciones cuentan con muy poca energía, debido a que la señal no tiene cambios bruscos, como pueden ser perturbaciones drásticas; cambios intensos en la armónica principal de la señal.

El tiempo promedio necesario para entrenar el sistema fuera de línea, con 2 muestras de señales de entrada sin ruido es alrededor de 430 [s].

Para una prueba el sistema coloca a la señal sin falla en la neurona (10,10), y todas las señales sin falla que se presentan al sistema estarán alrededor de esta. Probando confiabilidad en los resultados (Tabla 6.1). Las señales sin falla usadas son escaladas en 5, 10 y 50 veces la señal original (5x, 10x, 50x).

Magnitud	Posición
5x	10,10
10x	9,10
50x	10,9

Tabla 6.1 Colocación de señales a distintas magnitudes

El sistema está preparado para entrar al proceso en línea, toda señal que no detecte como zona sin falla (figura 6.2) es considerada error.

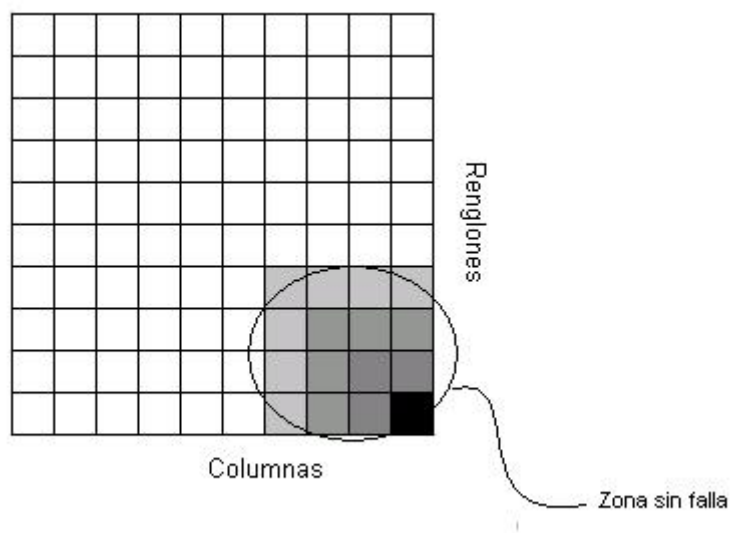


Figura 6.2 Representación gráfica del mapa resultante

La prueba anterior (figura 6.2) fue realizado utilizando la señal completa, si se muestrea la señal, se generaran 4 sub-señales a partir de la original, que entrenaran a su vez a 4 secciones del mapa (figura 6.3).

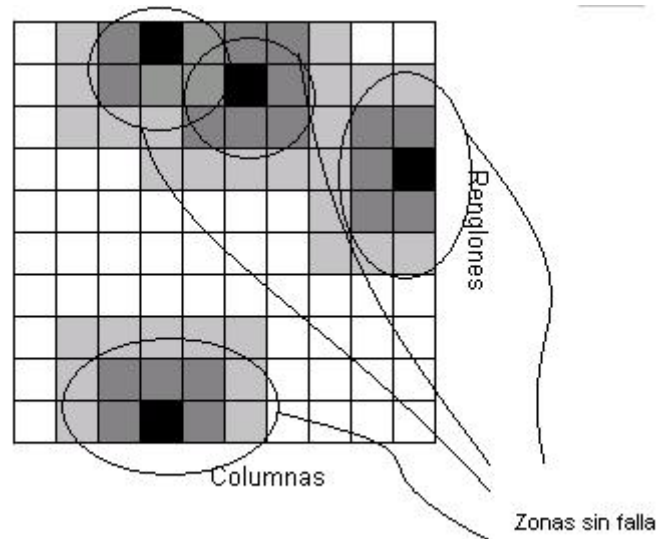


Figura 6.3 Representación gráfica de un mapa con 4 señales sin falla.

En este caso tenemos 4 zonas (Tabla 6.2) en las cuales si la señal esta lo suficientemente cerca, es decir a una distancia de uno, medida con la media euclídeana típica en los espacios de 2 dimensiones, se puede considerar la misma como una señal sin falla..

Muestra	Neurona
1	(2,6)
2	(10,4)
3	(4,10)
4	(1,4)

Tabla 6.2 Asignación de las muestras de la señal sin falla

La muestra 1 y 4 se encuentran considerablemente cerca entre ellas (distancia euclídeana ≤ 1), es decir se asemejan mucho, esto tiene como consecuencia que se le asigne una mayor zona a ese tipo de señal sin falla.

Ahora procederemos a realizar un análisis de los resultados cuando se prueba el algoritmo en línea.

6.3 Sistema en línea

Las señales de prueba con falla son: una señal con un aumento repentino del ruido en la señal (figura 6.3b), otra con una perturbación en un momento(figura 6.3c) y otra con un aumento de la energía promedio de la señal (figura 6.3d).

El sistema en línea utilizando la señal completa detecta la falla en la muestra que tiene falla, al detectarla el sistema sale de línea y realiza una actualización del mismo. Para detectar la falla, los tiempos son mínimos (Tabla 6.3)

Falla	Tiempo [s]
1	1.6
2	1.58
3	1.6

Tabla 6.3 Tiempos de detección de falla

Utilizando un muestreo de la señal, las fallas se detectaron de la siguiente manera:

Falla	Muestra(s)	Tiempo [s]
1	2,3,4	4.8
2	1,2,3,4	6.4
3	2,3	3.2

Tabla 6.4 Detección de las fallas en términos de muestras

Vemos como en la falla 1 (figura 6.3) a partir de que ocurre ese cambio en la frecuencia el sistema no detecta ninguna de las señales subsecuentes como adecuadas, en la falla 2, ninguna de las señales las considera como sin falla, esto posiblemente debido a que la frecuencia de la señal es la misma que la señal sin falla, entonces en este caso se debe de realizar una valoración de la señal y entrenarla como una señal sin falla, mientras que en la última falla, si se detectan únicamente las fallas en los segmentos de señal que tienen ese aumento en la señal de frecuencia cero.

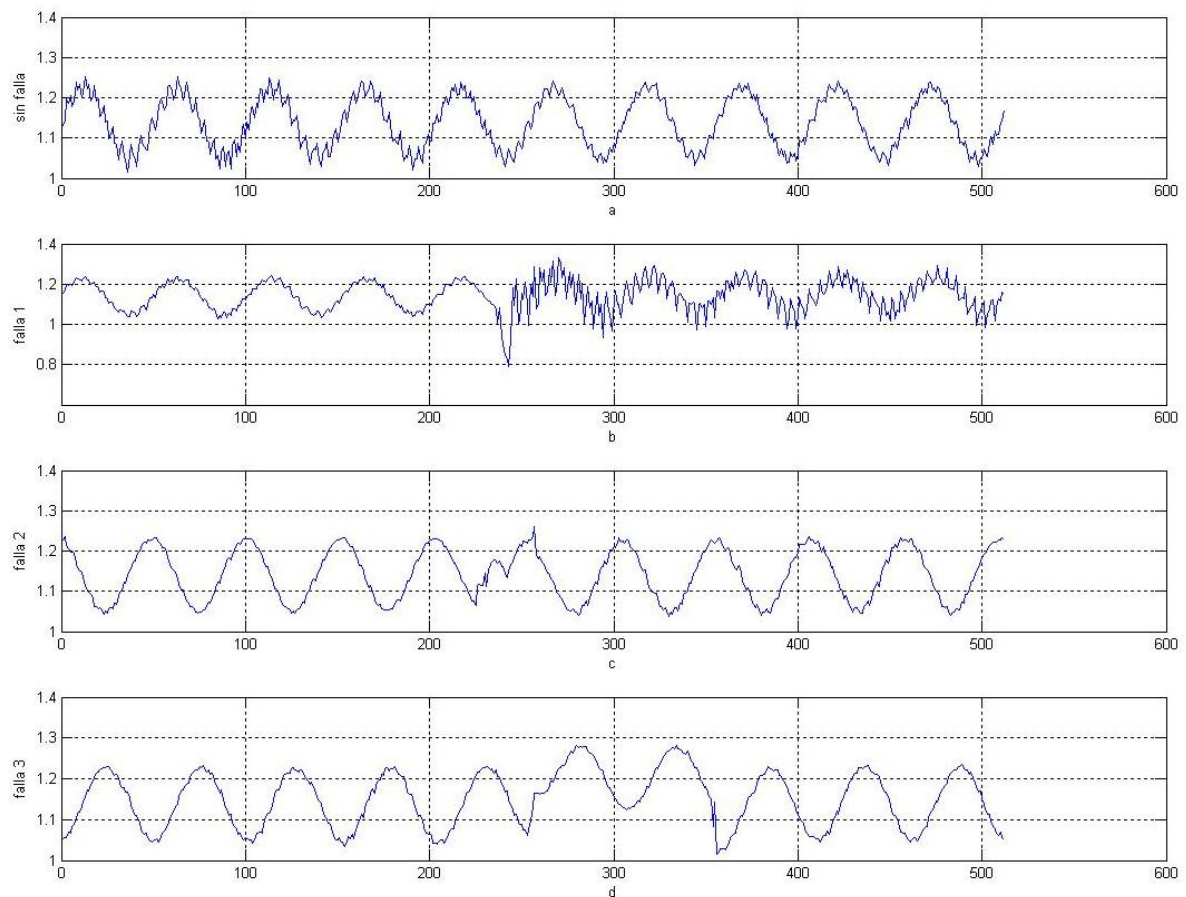


Figura 6.3 Señal de entrada con fallas

Cada una de las fallas presenta una descomposición de onduletas, lo que nos permite poder diferenciarla de la señal de entrenamiento.

En la fase de detección de falla, la aproximación en el cual se utiliza la señal completa (aproximación A) mostró ser superior a aquella en el cual se muestrea la señal (aproximación B), ya que en este ultimo, se presentaron errores al momento de detectar una falla y una señal sin falla, colocando en zonas no entrenadas a la señal sin falla, y acercando considerablemente señales sin falla con señales con falla.(Tabla 6.5)

Falla	Sistema A		Sistema B	
	Errores	Aciertos	Errores	Aciertos
1	0	1	0	4
2	0	1	2	2
3	0	1	0	4

Tabla 6.5 Comparación de los algoritmos con respecto a sus aciertos y errores.

Se trató de aumentar el número de neuronas para la aproximación B, sin embargo, no beneficio su desempeño.

Para entrenar cada nueva señal en la aproximación A, esta descompone en sus 4 niveles y es así como se reentrena al sistema, con el mismo algoritmo con el que se entrenó a la entrada sin falla, la actualización se tarda aproximadamente 215 [s], es decir la mitad de lo que se tardó en entrenar por primera vez al sistema fuera de línea. Esto quiere decir que para que el sistema pueda reconocer las 3 fallas presentadas se necesitan aproximadamente 645 [s] con el sistema fuera de línea.

En el sistema B, se tarda el triple con la falla 1, debido a que tiene que actualizar 3 señales, el doble con el error 2 y el doble con el error 3, lo anterior nos muestra que los tiempos de procesamiento de la aproximación B, dependen totalmente del tipo de falla que se presente, si es una falla puntual, se tarda poco, pero si es una falla que afecte por completo la señal, se tardará n-veces las muestras que afecte. Donde n es el número de muestras.

Debido a estos resultados, enfocaremos nuestros esfuerzos a la actualización mediante la aproximación A, ya que es la que presenta un mejor desempeño en relación a la identificación de fallas.

A continuación, colocamos figuras donde se muestra cada descomposición en onduletas de las fallas presentadas, que son a su vez las señales con las que se reentrena el sistema una vez presentada la falla.

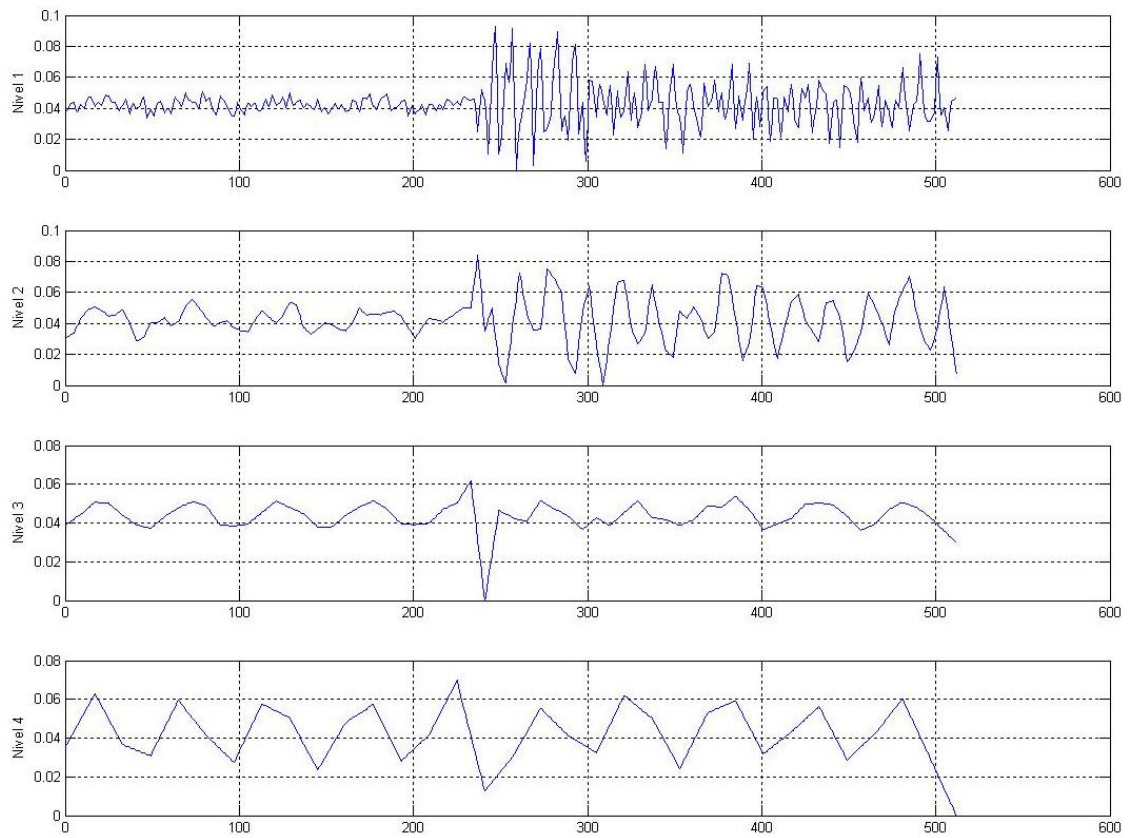


Figura 6.4 Descomposición de la falla 1

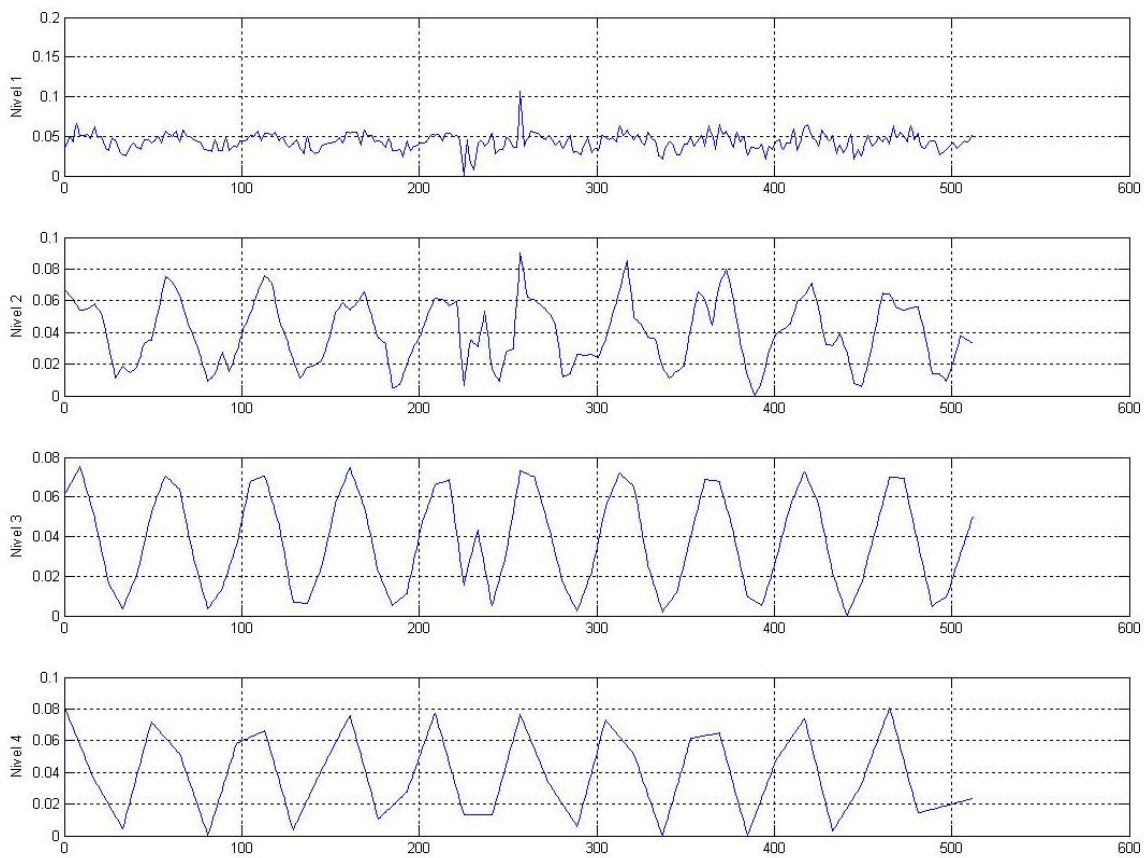


Figura 6.5 Descomposición de la falla 2

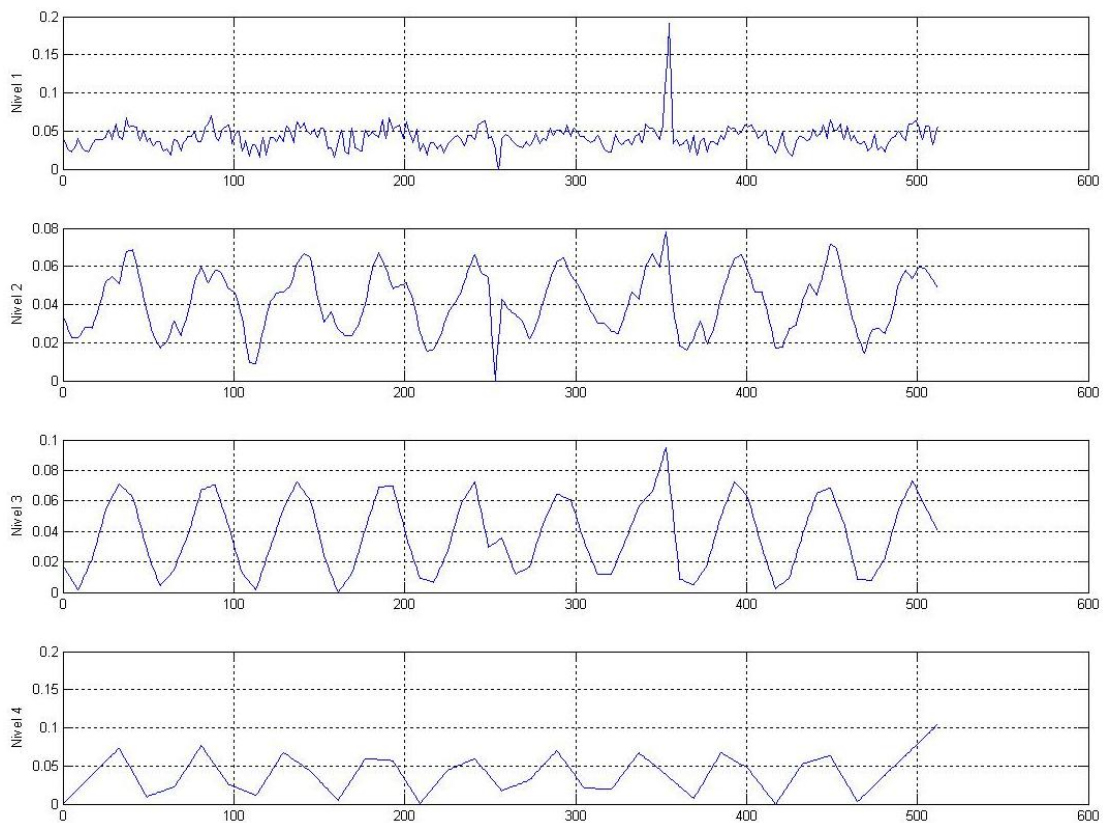


Figura 6.6 Descomposición de la falla 3

El sistema una vez actualizado presenta la siguiente distribución y relación con respecto a la señal que no presenta falla. (Tabla 6.6)

La señal sin falla, se mantuvo siempre en la neurona (10,10), indicando que el algoritmo es bastante efectivo en la detección de fallas nuevas sin tener que reconfigurarse.

Falla	Neurona	Distancia con elemento sin falla
1	(3,8)	7.28
2	(7,1)	9.48
3	(2,2)	11.31

Tabla 6.6 Relación de fallas en el mapa

6.4 Conclusiones

Cabe mencionar que en algunos de los experimentos, la falla 1 por su similitud con la señal entrenada resultaba muy cercana a la neurona ganadora sin falla, haciendo que no fuese catalogada como falla. Para prever esta situación, se tomó en cuenta también el segundo nivel de la transformada de onduletas para así categorizar la falla, el conflicto es que la actualización del sistema mueve invariablemente de lugar a la señal que no tiene falla. Se plantean 3 posibles soluciones.

- En el momento en que se presente la falla cercana a la señal sin falla, reestructurar la red avisando al usuario en que lugar quedó ahora colocada la señal sin falla y actualizar en el algoritmo las neuronas de la señal sin falla, esta manera fue probada y logró alejar a las 2 señales lo suficiente para ser tomadas como independientes la una de la otra. Sin embargo la actualización tomará mucho más tiempo, debido a que debe estar actualizando las 2 señales, y en algún caso podría quedar en un ciclo tal que se tardara demasiado tiempo en lograr dar una respuesta favorable.
- Preentrenar ligeramente las neuronas con las fallas que son mas comunes al sistema, para que de esta forma al momento de presentarse sólo las refuerce y las fallas nuevas que se presenten por el mismo preentrenamiento que se mostró van a estar localizadas en posiciones más alejadas a la señal sin falla.
- Al momento de crear el mapa y realizar el primer entrenamiento, hacer un análisis de las fallas más comunes y ver si no se encuentran demasiado cerca las señales. Si es el caso de que estuvieran muy cerca, se crea un nuevo mapa, todo esto fuera de línea, con un software encargado de realizarlo.

Son muy pocos los casos que se presentan con el error anteriormente descrito, la solución que se utilizó para prever este problema fue la última, ya que es el que presenta un menor tiempo de ejecución para el sistema así como menores recursos del mismo.

La aproximación B presentó múltiples desventajas contra la A, como fueron lentitud en los procesos de entrenamiento y actualización, y mala clasificación de las señales nuevas sin falla. Estos problemas pueden ser tratados con mas señales de

preentrenamiento sin falla y hardware que realice los algoritmos más eficazmente, las ventajas que nos ofrece son que la señal no precisa ser sobremuestreada al momento de tratarla, y también que al tomar muestras nos permite realizar un análisis de la señal mas localizado. Análisis que logra el sistema A mediante la transformada de wavelet a toda la señal, pero que debido al submuestreo se puede llegar a perder información valiosa de la misma.

El presente trabajo presenta un algoritmo que tiene que objetivo detectar e aislar eficazmente fallas distintas que se dan en el campo de la frecuencia, aplicado a un sistema dinámico de levitación magnética.

El enfoque del trabajo es utilizar una herramienta que nos permita un extenso análisis en el campo de la frecuencia, como pueden ser las onduletas, y una herramienta que nos permita identificar señales nuevas sin necesidad de agentes externos que estén valorando las mismas, esta herramienta se presenta en los mapeos autoorganizados.

Por parte de la arquitectura es, determinar las variables dinámicas del sistema, por lo que se propone un trabajo de optimización basado en el proceso físico que se quiera analizar.

El algoritmo cumple con el objetivo principal del trabajo, definir un algoritmo para la detección de fallas, aunque presenta limitantes en lo que respecta a portabilidad a otros sistemas físicos y al número de fallas que es capaz de aprender a lo largo del tiempo. Como lo son el hecho de que para cada sistema físico se requiera una nueva implementación del algoritmo con distintas variables.

Se presenta como trabajo futuro determinar cual debe ser la mejor forma de tratar la señal, ya sea muestreando la misma, o tomando las señales intactas que nos entrega el sistema físico, en tamaños muy bien definidos. A lo largo del trabajo se han presentado los problemas al momento de muestrear la señal, así como las ventajas y defectos que conlleva esto.

También lograr hacer que todo el algoritmo sea capaz de ejecutarse en línea sin necesidad de desperdiciar tiempo al momento de correrlo fuera de línea en el preprocesamiento de la señal, esto se puede lograr probando distintos tipos de preprocesamiento de tal manera que no sea necesario un aprendizaje previo.

Y realizar un algoritmo que sea capaz de detectar los valores óptimos para cada variable de la arquitectura dependiendo del sistema físico que se de.

8.1 Programas realizados para llevar a cabo la arquitectura

```
%Este programa realiza todo el procesamiento de la
%señal que proviene del levitador magnético
%Primero realiza el submuestreo de la señal
%y ajusta el tamaño a 512
%Después genera un arreglo en el cual se encuentran todas las señales
%junto con sus descomposiciones en wavelet
%Finalmente realiza las iteraciones necesarias para entrenar el mapa
%y limpia el espacio de trabajo de las variables
%no necesarias
tic
d=sprintf('train');
load(d);
someva=createmf(10,10,512);
entra=muestra10pc(q_piv_maglev_q4_lc_Resp___A__0__);
ent(1).senial=extent512(entra);
entra=muestra10pc(q_piv_maglev_q4_lc_Resp___A__1__);
ent(2).senial=extent512(entra);
datawav=wavematot(ent,4,'db1');
datawavnorm=normalizaentra(datawav);
evasom
toc
clear i
clear d
clear q_piv_maglev_q4_lc_Resp___A__0__ q_piv_maglev_q4_lc_Resp___A__1__
clear q_piv_maglev_q_xb_Resp___mm__0__ q_piv_maglev_q_xb_Resp___mm__1__;
clear ent entra j k datawav;
```

```
%En este programa se realiza el entrenamiento  
%del mapa por primera vez siendo j el numero  
%de iteraciones que se van a hacer por elemento a entrenar
```

```
tic  
  for k=1:2  
    for i=1:4  
      for j=1:50  
        someva=somleo3(someva,datawavnorm(k).matriz(i).senial,1.4);  
      end  
    end  
  end  
toc
```

%Este programa realiza el algoritmo de la SOM

% requiere para su funcionamiento

function matrix=SOMLEO3(red,invec,sigma)

[nmatriz]=up(invec,red,sigma);

matrix=nmatriz;

%Funcion principal de actualizacion

function [nmatrix]=up(invec,red,sigma)

[winner, rengw, colw]=minim(invec,red);

A=0.3;

rw=rengw;

cw=colw;

vect=size(red);

limitreng=vect(1); % el vect 1 nos da rengl y el 2 column

limitcolu=vect(2);

vect(3)=length(red(1,1).peso);

for j=1 : limitreng

for k=1 : limitcolu

vecind=vec(j,k,rengw,colw,sigma);

for i=1:vect(3)

red(j,k).peso(i)=red(j,k).peso(i)+vecind.*A.*(invec(i)-red(j,k).peso(i));

end

end

end

nmatrix=red;

%Funcion para obtener la neurona ganadora

```
function [min,reng,col] = minim(invec,red)
```

```
winner=0;
```

```
smallest=1000;
```

```
vec=size(red);
```

```
limit1=vec(1);
```

```
limit2=vec(2);
```

```
for j=1 : limit1
```

```
    for i=1 : limit2
```

```
        mag=dist(invec,red,j,i);
```

```
        if mag<smallest
```

```
            winner=red(j,i);
```

```
            rengw=j;
```

```
            colw=i;
```

```
            smallest=mag;
```

```
        end
```

```
    end
```

```
end
```

```
reng=rengw;
```

```
col=colw;
```

```
min = winner;
```

%Funcion para obtener la distancia entre una neurona y el
%vector de entradas

```
function mag = dist(invec,red,j,i)
sum=0;
for k=1:length(invec)
    sum=sum+(invec(k)-red(j,i).peso(k)).^2;
end
mag=sqrt(sum);
```

```
function vecind=vec(reng,col,rengw,colw,sigma)
```

```
a=exp(-(((reng-rengw).^2+(col-colw).^2)/(sigma.^2)));
vecind=a;
```

%Este programa realiza la evaluacion del algoritmo en linea
 %Y señala en que momento existe un error, y al momento de detectarlo
 %realiza la actualizacion, sacando temporalmente al sistema de linea

tic

```
[a b]=evalua(someva,datawavnorm(1).matriz(1).senial);
[c d]=evalua(someva,datawavnorm(1).matriz(2).senial);
reng(1)=a;
col(1)=b;
reng2(1)=c;
col2(1)=d;
masterflag=1;
while masterflag < 4

    for i=1:4
        tic
        d=sprintf('senial%d',i);
        load(d);
        datan=muestra10pc(q_piv_maglev_q4_lc_Resp___A__0__);
        data(i).senial=extent512(datan);
        dataintwav=wavematot(data,2,'db1');
        dataintwavnorm=normalizaentra(dataintwav);
        fprintf('muestra %d\t\n',i)
        [a b]=evalua(someva,dataintwavnorm(i).matriz(1).senial);
        [c d]=evalua(someva,dataintwavnorm(i).matriz(2).senial);
        flag1=distleo(a,b,reng,col)
        flag2=distleo(c,d,reng2,col2)
        if flag1==1
            fprintf('Existe error Actualizando')
            toc
            someva=actualizafin(someva,data,i);
            [reng(i+1) col(i+1)]=evalua(someva,dataintwavnorm(i).matriz(1).senial);
            [reng2(i+1) col2(i+1)]=evalua(someva,dataintwavnorm(i).matriz(2).senial);
```

```

        masterflag=1;
        break
    end
    if flag2==1
        fprintf('Existe error Actualizando')
        toc
        someva=actualizafin(someva,data,i);
        [reng(i+1) col(i+1)]=evalua(someva,dataintwavnorm(i).matriz(1).senal);
        [reng2(i+1) col2(i+1)]=evalua(someva,dataintwavnorm(i).matriz(2).senal);
        masterflag=1;
        break
    end
    if i==1
        fprintf('\nNo hay error')
    else
        fprintf('\nError %d',i)
    end

    toc
    masterflag=i;
end

end
toc
clear i
clear d
clear q_piv_maglev_q4_Ic_Resp___A__0_ q_piv_maglev_q4_Ic_Resp___A__1_
q_piv_maglev_q_xb_Resp___mm__0_ q_piv_maglev_q_xb_Resp___mm__1_;
clear j k;

%Este programa calcula la distancia
%euclideana entre 2 neurona y levanta una bandera
%si es que se encuentran mas lejos que una
%distancia de raiz de 2
function flag1=distleo(a,b,reng,col)
small=sqrt(2);
flag1=0;

```

```
for i=1:length(reng)
    magtemp=sqrt((reng(i)-a)^2+(col(i)-b)^2);
    if magtemp<=small
        break
    end
    if i==length(reng)
        flag1=1;
    end
end
end
```

- M. Campos M. y G. A. Carpenter (1998) "WSOM: Building Adaptive Wavelets with Self Organizing Maps." Department of Cognitive and Neural Systems. Boston University. 1998
- A. Haar (1910) "Zur Theorie der orthogonalen Funktionensysteme," *Math. Annal.*, 69, 331–371, 1910.
- R. E. A. C. Paley y J. E. Littlewood (1937) "Theorems on Fourier series and power series," (I). *J.L.M.S.*, **6**, 230–233, 1931; (II). *P.L.M.S.*, **42**, 52–89, 1936; (III). *ibid*, **43**, 105–126, 1937.
- C. E. Shannon (1948) "Mathematical theory of communication," *Bell Syst. Tech. J.*, **27**, , 1948.
- A. P. Calderon (1977) "An atomic decomposition of distributions in parabolic H^p spaces," *Adv.Math.*, **25**, 1977.
- J. Stromberg (1981) "A modified Franklin system and higher-order systems of R_n as unconditional bases for Hardy spaces," *Conf. Harmonic Analysis in Honor of A. Zygmund*, W. Beckner et al., ed., 475–493 *Wadsworth Math Series*, Wadsworth, Belmont, CA, 1981.
- A. Grossman and J. Morlet (1984) "Decomposition of Hardy functions into square integrable wavelets of constant shape," *SIAM J. Math. Anal.*, **15**, 723–736, 1984.
- Y. Meyer (1986) "Principe d'incertitude, bases Hilbertiennes et algebras d'operateurs", *Bourbaki Seminar*, no. 662, 1985–1986.
- S. Mallat (1989) "Multiresolution approximation and wavelet orthogonal bases of L_2 ," *Trans. AMS*, **315**, 69–87, 1989.
- I. Daubechies (1987) "Orthogonal bases of compactly supported wavelets," *Comm. Pure Appl. Math.*, **41**, 909–996, 1988.
- W. Härdle, G. Kerkycharian, D. Picard, A. Tsybakov (1997) "Seminaire Paris-Berlin, Seminar Berlin-Paris, Wavelets, Approximation and Statistical Applications"
- M. W. Frazier (1999) "An introduction to wavelets through linear algebra" Springer 1999
- J. G. Proakis, D. G. Manolakis (1998) "Tratamiento digital de señales" Prentice Hall 1998, México
- M. L. Minsky, y Papert (1969) "Perceptron", Cambridge MA MIT Pres.
- J. A. Freeman (1992) "Neural networks algorithms applications and programming techniques", CNS 1992.
- L. Fausett (1994) "Fundamentals of neural networks", Prentice Hall, 1994
- K. Teuvo (1994) "Self Organizing Maps", 3ed , Springer, 1994, Nueva York.
- K. Teuvo (1990) "The Self-Organizing map" *Proceedings of the IEEE* 78(9) 1464-1480

- Vesanto (2000) "Clustering of the self organizing map", IEEE Transactions in Neural Networks Vol 11 No.3
- R. J. Patton, Paul M. Frank and Robert N. Clark. (2000) "Issues of fault diagnosis for dynamic systems". Springer, 2000.
- S. Lee et al (2003) "Model based Fault detection and Isolation Meted Using ART2 Neural Network" International Journal of intelligent systems Vol 18 2003
- Hrishikesh et al (1996) "Clustering in wavelet domain: A multiresolution ART Network for anomaly detection"
- I. R. Process (1984) "Fault Detection Based on Modelling and Estimation", Methods Automatica Vol. 20 387-404
- J. Cuellar G. (2003) "Diagnostico en Linea del estado de un medidor de oxigeno disuelto" Tesis, Maestria , Universidad Nacional Autonoma de Mexico Division de EStudios de posgrado , Facultad de ingenieria, 2005
- M. C. Moisen Locena (2005) "Caracterizacion de fallas con ultrasonidos utilizando redes neuronales no supervisadas" Tesis, Licenciatura , Universidad Autonoma de Guerrero, Facultad de Matemáticas, 2005
- Manuales del levitador y la tarjeta marca Quanser, www.quanser.com