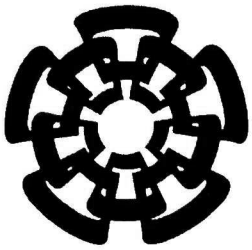
XX(116556.1)



CINVESTAV

Centro de Investigación y de Estudios Avanzados del I.P.N.

Unidad Guadalajara

Entrenamiento de redes neuronales con el filtro de Kalman

Tesis que presenta:

Alma Yolanda Alanís García

para obtener el grado de:

Maestro en Ciencias

en la especialidad de:

Ingeniería Eléctrica

Directores de Tesis

Dr. Edgar Nelson Sánchez Camperos

Dr. Jesús Rico Melgoza

Guadalajara, Jal., Julio del 2004.

Entrenamiento de redes neuronales con el filtro de Kalman

Tesis de Maestría en Ciencias
Ingeniería Eléctrica

Por:
Alma Yolanda Alanís García
Ingeniero Eléctrico
Instituto Tecnológico de Durango 1997-2002

Becario del CONACyT, expediente no. **171362**

Directores de Tesis
Dr. Edgar Nelson Sánchez Camperos
Dr. Jesús Rico Melgoza

CINVESTAV del IPN Unidad Guadalajara, Julio del 2004.

AGRADECIMIENTOS.

Dedico esta tesis con todo mi cariño y respeto a mis padres y a mi hermano, por su apoyo incondicional en todo momento.

Y en especial a Gilberto por todo el amor, paciencia y comprensión que me ha brindado todos estos años.

Agradezco al Dr. Edgar por su apoyo, por todo el tiempo y la dedicación brindados para la realización del presente trabajo.

Agradezco al Dr. Jesús Rico por la atención prestada a este trabajo y por proporcionarle un enfoque practico al mismo.

Agradezco al Dr. Loukianov y al Dr. Ramírez Arredondo por sus comentarios siempre apropiados, los cuales contribuyeron a enriquecer el contenido de este trabajo.

Agradezco a mis compañeros: Carlos Alberto, Carlos Iván, Flavio, Isaac, Juan, Raymundo, Tonatiuh, Víctor y Víctor Jesús, por su amistad y apoyo en todo momento.

Al CINVESTAV unidad Guadalajara por el apoyo otorgado durante el desarrollo del presente trabajo de investigación.

Al CONACyT por apoyarme económicamente durante toda mi estancia en la maestría.

Índice General.

Índice de figuras.

Capítulo 1.

INTRODUCCIÓN.

En esta tesis se presentan los resultados del uso de algoritmos de entrenamiento, para redes neuronales recurrentes, basados en el Filtro de Kalman. Así mismo las redes neuronales recurrentes entrenadas con el algoritmo aquí propuesto, son aplicadas para la predicción de series de tiempo en sistemas eléctricos de potencia así como para la reproducción de sistemas caóticos discretos. El uso del Filtro de Kalman en el entrenamiento de redes neuronales se ha incrementado en gran medida; esto debido a los excelentes resultados obtenidos en diversas aplicaciones [7], [10], [29], [35], [44], [39], [42].

La investigación en redes neuronales, desde su resurgimiento en la década de los ochenta del siglo pasado ha suscitado un creciente interés, en muy diversas áreas de la ingeniería. Existen muchas clasificaciones para las redes neuronales, por el momento sólo se hará la distinción entre las redes neuronales estáticas y las redes neuronales recurrentes o dinámicas.

Es bien conocido que las redes neuronales estáticas son capaces de aproximar cualquier función continua [3], [10], [31]. Sin embargo las redes neuronales recurrentes poseen un rico repertorio de arquitecturas, lo cual las habilita para realizar diversas aplicaciones [10], que no son posibles con las redes neuronales estáticas; algunas de estas aplicaciones son: predicción no lineal, modelado, control, representaciones en espacio de estado, etc.

El interés principal para los casos de estudio propuestos en esta tesis, reside en realizar predicción no lineal a corto plazo; también se presentan resultados interesantes en la identificación y modelado de sistemas caóticos discretos.

La aplicación de las redes neuronales para la predicción de series de tiempo en sistemas eléctricos de potencia tiene un gran impacto en la planeación de los mercados de este ramo, ya que aún y cuando se utilizan otras técnicas para realizar la predicción de series de tiempo [3], [23], [38]; los resultados que se muestran en este trabajo representan una importante alternativa. El objetivo de las predicciones es la reducción de riesgos en la toma de decisiones, a través de estimaciones de eventos futuros [38]. La predicción en general tiene aplicaciones en varias áreas.

Los siguientes ejemplos describen algunas de las aplicaciones de las predicciones de series de tiempo en sistemas eléctricos de potencia.

- Administración de inventarios
- Planeación de producción
- Control de procesos
- Planeación financiera
- Planeación de personal
- Planeación de recursos

Para el caso particular de este trabajo, la predicción de series de tiempo en sistemas eléctricos de potencia tiene aplicación directa en la planeación de generación y suministro de energía eléctrica e influencia directa en los mercados eléctricos [38].

La industria de la energía eléctrica, en muchos países, se torna más competitiva día a día. Este hecho explica la importancia de conocer con anticipación el comportamiento del mercado, para poder proveer un servicio de mejor calidad y menor costo tanto para el productor como para el consumidor.

Las series de tiempo en sistemas eléctricos de potencia presentan las siguientes características [2], [23]:

- Alta frecuencia
- Media y varianza variables
- Ciclos o ritmos (semanales y diarios)
- Efecto del calendario (Fines de semana, días festivos, etc.)
- Alta volatilidad
- Evoluciones irregulares

Las series de tiempo caso de estudio en este trabajo, son las relacionadas a la demanda eléctrica y a los precios de la energía eléctrica. Por otra parte, los sistemas no lineales tienen comportamientos dinámicos muy variados, que incluyen el *caos*. El comportamiento caótico es caracterizado por su alta sensibilidad a la variación de parámetros, perturbaciones externas y particularmente a cambios pequeños en las condiciones iniciales.

El *caos* presenta algunas propiedades importantes que en los últimos años han dado lugar a aplicaciones poco tradicionales. En los ejemplos que se presentarán en el *capítulo 6*, la identificación de las series de tiempo se realizará bajo la suposición de que los datos de las mismas fueron obtenidos a partir de un sistema *caótico* presumiblemente desconocido.

1.2 Organización de la tesis.

La presente tesis está organizada de la siguiente manera:

En el *Capítulo 2,* se realiza la revisión de los antecedentes de redes neuronales, partiendo de las redes neuronales biológicas hasta culminar con los principales tipos de redes neuronales artificiales.

El *Capítulo 3* está dedicado al Filtro de Kalman, su deducción y sus modalidades como son: el Filtro de Kalman Extendido (*FKE*), el Filtro de Kalman Extendido Libre de Derivadas (*FKELD*) y el Filtro de Kalman Extendido con Factor de Olvido (*FKEFO*).

Estos dos primeros capítulos presentan los conocimientos que, si bien ya existen son necesarios para la comprensión de la tesis. Los siguientes capítulos presentan la principal aportación de esta tesis.

En el *Capítulo 4,* se presenta la formulación del *FKE* y el *FKELD* aplicados al entrenamiento de redes neuronales recurrentes tipo perceptrón multicapa (*RMLP*), así como los resultados obtenidos en la predicción de las series de tiempo en sistemas eléctricos de potencia.

En el *Capítulo 5,* se introducen las redes neuronales recurrentes de alto orden (*RHONN, por su nombre en ingles "Recurrent High Order Neural Networks"*) entrenadas con algoritmos basados en el Filtro de Kalman

En el *Capítulo 6,* se hace la aplicación de las *RHONN* entrenadas con el Filtro de Kalman para la reproducción de sistemas caóticos discretos, así mismo se realiza el análisis de las propiedades caóticas de las series de tiempo en sistemas eléctricos de potencia.

En el *Capítulo 7,* se presentan las conclusiones generales de la investigación desarrollada en la presente tesis, así como las perspectivas de trabajo futuro.

En el *apéndice 1,* se incluye un resumen de las estructuras de identificación para modelos lineales, como complemento a su contraparte no lineal presentadas en el *capítulo 4.* Así mismo en el *apéndice 2,* se muestran algunos otros resultados obtenidos durante el desarrollo de la investigación concerniente a esta tesis; finalmente en el *apéndice 3,* se anexan los artículos aceptados para publicación en los congresos CLCA'04, IJCNN'04 y APWCCS'04, respectivamente; los cuales están basados en la presente tesis.

Capítulo 2

ANTECEDENTES DE REDES NEURONALES.

El conocimiento sobre redes neuronales se ha incrementado de manera significativa en los últimos años, llegando a ser una metodología bien establecida para modelado y control, ésta permite resolver algunos de los problemas de control más dificiles, como identificación y control de sistemas no lineales complejos. En particular el uso de redes neuronales recurrentes, para el modelado y la reproducción de transformaciones entrada-salida, se ha incrementado en años recientes.

2.1 Redes Neuronales Biológicas.

Las neuronas son las células que forman el sistema nervioso central incluyendo el cerebro; tienen tres componentes principales: las dendritas, el cuerpo de la célula o soma y el axón. Las dendritas son el elemento receptor; son como fibras nerviosas que cargan de señales eléctricas al cuerpo de la célula. El cuerpo de la célula realiza la suma de esas señales de entrada. El axón es una fibra larga que lleva la señal desde el cuerpo de la célula hacia otras neuronas.

El punto de contacto entre el axón de una célula y la dendrita de otra célula es llamado sinapsis; la comunicación entre las neuronas tiene lugar como resultado de la liberación de unas sustancias llamadas neurotransmisores por parte de la célula presináptica y la absorción de las mismas por la célula postsináptica. Así la sinapsis convierte una señal eléctrica de la neurona presináptica en un proceso químico en la sinapsis y después se reconvierte en una señal eléctrica en la neurona postsináptica. En la figura 2.1 se muestra un esquema simplificado de la interconexión de dos neuronas biológicas [37].

Todas las neuronas conducen la información de forma similar; ésta viaja a lo largo de axones en breves impulsos eléctricos, denominados potenciales de acción. Los potenciales de acción alcanzan una amplitud máxima de unos 100 mV y dura 1 ms. La membrana en reposo mantiene un gradiente de potencial eléctrico de –70 mV. Los axones tienen una cubierta que se denomina vaina de mielina, la cual funciona como una capa aislante; ésta es interrumpida en varios puntos por los nodos de Ranvier [37].

Las fibras nerviosas en si son malos conductores. La transmisión del potencial de acción a lo largo del axón es el resultado de una serie de despolarizaciones que tienen lugar en los nodos de Ranvier. Cuando uno de los nodos se despolariza, se desencadena la despolarización del siguiente nodo. El potencial de acción viaja a lo largo de la fibra en forma discontinua de un nodo a otro. Una vez que un potencial de acción ha pasado por un cierto punto, ese punto no puede volver a ser excitado durante aproximadamente 1 ms, que es el tiempo que tarda en volver a su potencial de reposo. Este periodo refractario limita la frecuencia de transmisión de los impulsos nerviosos.

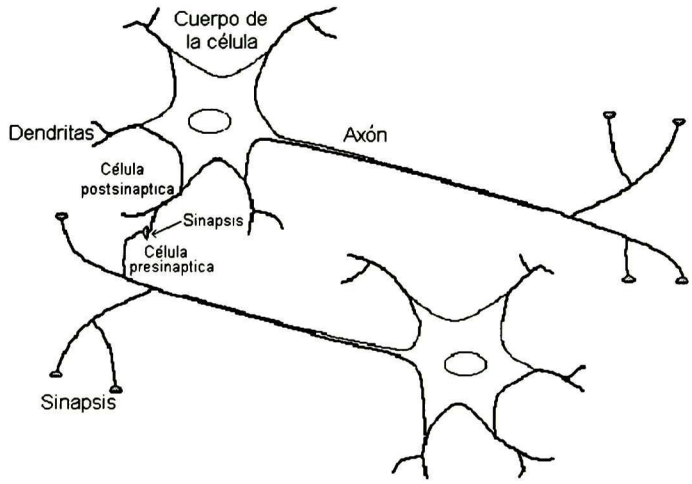


Figura 2.1. Neuronas biológicas.

En algunas neuronas los impulsos se inician en la unión entre el axón y el soma y luego se transmiten a lo largo del axón a otras células nerviosas. Cuando el axón está cerca de sus células destino, se divide en muchas ramificaciones que forman sinapsis con el soma o axones de otras células. Las sinapsis pueden ser excitatorias o inhibitorias según el neurotransmisor que se libere. Cada neurona recibe de 10,000 a 100,000 sinapsis y su axón realiza una cantidad similar de sinapsis. Las sinapsis se clasifican según su posición en la superficie de la neurona receptora en tres tipos: axo-somática, axo-dendrítica y axo-axónica [34].

Las redes neuronales biológicas están constituidas por un gran número de neuronas conectadas en forma masiva. Conforman el sistema nervioso y el cerebro. El cerebro puede contener 10^{11} neuronas y 10^{15} interconexiones [34].

2.2 Las redes neuronales artificiales y su relación con las biológicas.

Las redes neuronales artificiales son modelos simplificados de las redes neuronales biológicas. Tratan de extraer las excelentes capacidades del cerebro para resolver ciertos problemas complejos como: visión, reconocimiento de patrones o control moto-sensorial. La red neuronal artificial es un procesador paralelo distribuido, la cual puede almacenar conocimiento experimental y mantenerlo disponible para su uso [10].

Las redes neuronales artificiales no alcanzan la complejidad del cerebro; sin embargo existen dos aspectos similares entre redes neuronales biológicas y artificiales: primero los bloques de construcción de ambas redes son elementos computacionales sencillos altamente interconectados (aunque las redes neuronales artificiales son mucho más simples que las biológicas); segundo, las conexiones entre las neuronas determinan la función de la red.

El modelo de una neurona artificial es una imitación del proceso de una neurona biológica. Una neurona artificial es la unidad fundamental para la operación de una red neuronal artificial. En la figura 2.2 se observa una neurona artificial en su forma general y su similitud con una neurona biológica.

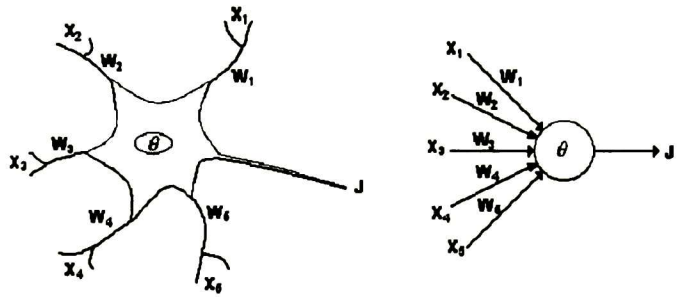


Figura 2.2.- De la neurona biológica a la neurona artificial.

De la observación detallada del proceso biológico, se han hallado las siguientes analogías con el sistema artificial.

- Las entradas X_i representan las señales que provienen de otras neuronas y que son capturadas por las dendritas. Para las neuronas artificiales, las entradas pueden ser variables continuas o discretas, mientras que en las neuronas biológicas son pulsos discretos.
- Los pesos W_i representan la intensidad de la sinapsis que conecta dos neuronas; tanto X_i como W_i son valores reales
- θ es la función umbral que la neurona debe sobrepasar para activarse; este proceso ocurre biológicamente en el cuerpo de la célula.

A partir de la siguiente sección se omitirá la especificación de la palabra artificial ya que se tratarán exclusivamente este tipo de neuronas.

2.3 El modelo matemático de una neurona

Aún no ha sido establecida una notación matemática única para redes neuronales artificiales, ya que sus aplicaciones son útiles en muchos campos: ingeniería, física, sociología, matemáticas, etc. En este trabajo se adopta la convención utilizada en [10].

La neurona es la unidad de proceso de información fundamental en una red neuronal. En la figura 2.3 se muestra el modelo de una neurona; éste forma la base para el diseño de una red neuronal artificial. Aquí se pueden identificar tres elementos básicos del modelo neuronal.

- Un conjunto de *sinapsis*, cada una de las cuales está caracterizada por un *peso* o *ganancia* sináptica. Específicamente, una señal x_j a la entrada de la sinapsis j conectada a la neurona k es multiplicada por un peso sináptico w_{kj} Es importante notar que el primer subíndice corresponde a la neurona en cuestión, mientras que el segundo subíndice corresponde a la entrada
- Un *sumador* o punto de suma para sumar las señales de entrada pesadas por su respectivo peso sináptico; las operaciones hasta aquí descritas constituyen combinaciones lineales.
- Una *función de activación* para limitar la amplitud de la salida de la neurona. La función de activación típicamente está normalizada dentro de un intervalo cerrado [0,1] o alternativamente [-1,1].

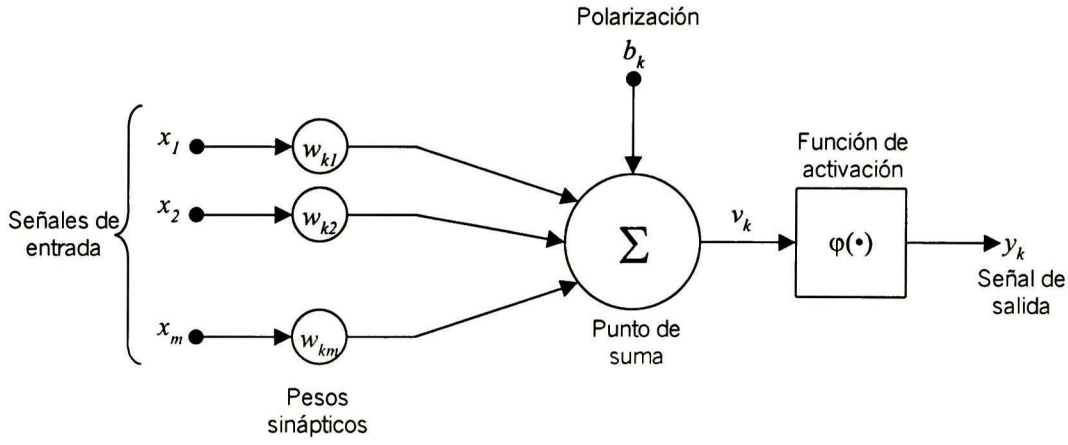


Figura 2.3 Modelo de una neurona artificial.

El modelo neuronal de la figura 2.3 también incluye una polarización externa, denotada por b_k. La polarización b_k tiene el efecto de incrementar o reducir la entrada total a la función de activación, dependiendo de sí la polarización es positiva o negativa, respectivamente.

En términos matemáticos, es posible describir la neurona k de la figura 2.3, por el siguiente par de ecuaciones.

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{2.1}$$

y

$$y_k = \varphi(u_k + b_k) \tag{2.2}$$

donde $x_1, x_2, \cdots, x_m$ son las señales de entrada; $w_{k1}, w_{k2}, \cdots, w_{km}$ son los pesos sinápticos de la neurona k; u_k es la combinación lineal de las entradas ponderadas; b_k es la polarización; $\varphi(\bullet)$ es la función de activación; finalmente y_k es la señal de salida de la neurona.

La polarización es un parámetro externo de la neurona k, pero es posible considerarla como parte de las señales de entrada, de tal forma que si se combinan (2.1) y (2.2) se tiene:

$$v_k = \sum_{j=0}^{m} w_{kj} x_j \tag{2.3}$$

y

$$y_k = \varphi(v_k) \tag{2.4}$$

En la ecuación (2.3) se ha agregado una nueva sinapsis. Su entrada es:

$$x_0 = +1$$

y el peso correspondiente es:

$$w_{k0} = b_k$$

De tal forma que el modelo neuronal presentado en la figura 2.3 se puede reformular como en la figura 2.4. Estos modelos pueden tener una apariencia diferente, pero matemáticamente son iguales.

Los esquemas presentados en las figuras 2.3 y 2.4, respectivamente, no son la forma más común de representar gráficamente una neurona. Una representación más común es la que se muestra en la figura 2.5 y ésta se basa en la teoría de grafos orientados [10].

Hasta este punto solo se ha nombrado a la función de activación, sin embargo, no se ha definido formalmente. A continuación se analizarán algunas de las funciones de activación más utilizadas en redes neuronales.

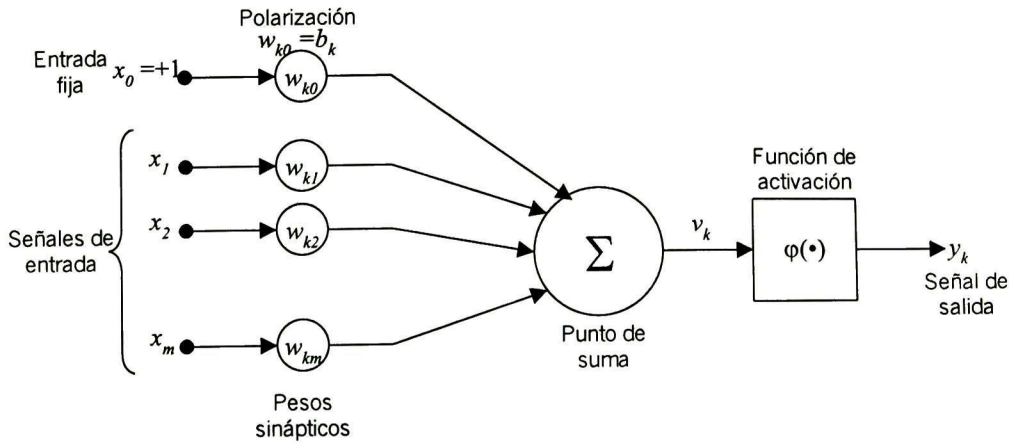


Figura 2.4 Otro modelo neuronal

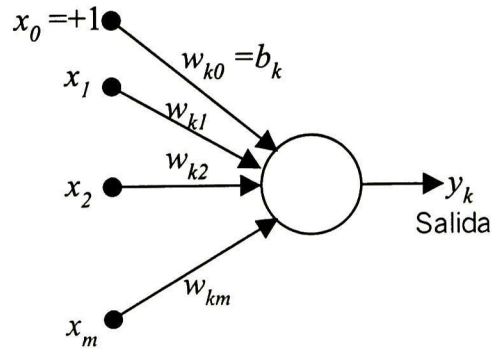


Figura 2.5 Gráfico de una neurona

2.3.1 Tipos de funciones de activación.

Las funciones de activación, denotadas por $\varphi(v)$, definen la salida de la neurona en términos de un campo local inducido v. Aquí se analizarán tres de los tipos básicos de las funciones de activación:

1. *Función escalón o umbral.* Para este tipo de función de activación, descrita en la figura 2.6(a), se tiene:

$$\varphi(v)=\begin{cases}1 & \text{if } v\geq 0\\ 0 & \text{if } v<0\end{cases} \tag{2.5}$$

Correspondientemente, la salida de la neurona k empleando esta función de activación queda expresada como:

$$y_k = \begin{cases} 1 \text{ if } v_k \geq 0 \\ 0 \text{ if } v_k < 0 \end{cases} \tag{2.6}$$

con v_k como en (2.3).

2. *Función lineal a tramos.* Para este tipo de función descrita en la figura 2.6(b) se tiene:

$$\varphi(v) = \begin{cases} 1 & v \geq +\frac{1}{2} \\ v & +\frac{1}{2} > v > -\frac{1}{2} \\ 0 & v \leq -\frac{1}{2} \end{cases} \tag{2.7}$$

donde el factor de amplificación dentro de la región de operación se supone igual a la unidad.

3. *Función sigmoidal.* Ésta es la función más comúnmente utilizada en redes neuronales artificiales. Es definida estrictamente creciente, con un comportamiento asintótico. Un ejemplo de la función sigmoidal es la función logística:

$$\varphi(v) = \frac{1}{1+e^{-av}} \tag{2.8}$$

donde a es el parámetro que determina la pendiente de la función sigmoidal

Las funciones de activación hasta aquí descritas toman valores en el intervalo cerrado [0,1], sin embargo tienen variantes que pueden tomar valores en el intervalo cerrado [-1,1]. La función escalón (2.6) es ahora definida como:

$$\varphi(v) = \begin{cases} 1 & \text{if } v > 0 \\ 0 & \text{if } v = 0 \\ -1 & \text{if } v < 0 \end{cases} \tag{2.9}$$

la cual es comúnmente llamada función signo. Para la forma correspondiente de la función sigmoidal se puede usar la función tangente hiperbólica, definida como:

$$\varphi(v) = \tanh(v) \tag{2.10}$$

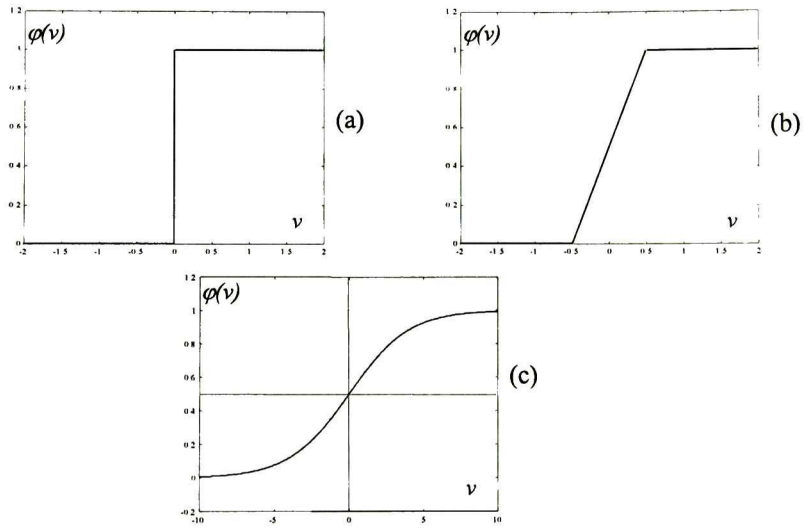


Figura 2.6 Funciones de activación (a) Función *Escalón*. (b) Función *Lineal a tramos*. (c) Función *Sigmoidal*.

2.4 Arquitecturas neuronales.

En general, es posible identificar tres clases diferentes de arquitecturas neuronales. Las cuales se describen a continuación:

2.4.1 Redes neuronales unicapa.

Generalmente en las redes neuronales, las neuronas están organizadas por capas. En el caso más simple, una red neuronal unicapa, la capa de entrada se conecta directamente a la capa de neuronas de salida por medio de las sinapsis. Esto es ilustrado en la figura 2.7(a) para el caso de tres nodos tanto en la capa de entrada como en la capa de salida. Se le da la designación de unicapa, porque sólo tiene una capa con nodos computacionales; para esta designación no se toma en cuenta la capa que contiene los nodos de entrada.

2.4.2 Redes neuronales multicapa.

Las redes neuronales multicapa se distinguen por tener una o más capas de neuronas ocultas, cuyos nodos computacionales son llamados neuronas ocultas o unidades ocultas. La red neuronal mostrada en la figura 2.7(b) se dice totalmente conectada, en el sentido de que todos los nodos en cada capa de la red están conectado a todos los otros nodos en la siguiente capa. En el caso de que falte alguna de las de la red, entonces se dice que la red es parcialmente conectada.

2.4.3 Redes neuronales recurrentes.

Las redes neuronales recurrentes, se distinguen de las anteriores, en que por lo menos tienen un lazo de retroalimentación. Por ejemplo, una red recurrente puede consistir de una sola capa oculta con cada una de sus neuronas retroalimentando las señales de salida a la entrada de otras neuronas, como se ilustra en la figura 2.7(c), donde se introduce el operador de retardo unitario z^{-1} En la estructura mostrada no existen lazos de retroalimentación en la red. La red neuronal representada en la figura 2.7(c) tampoco tiene neuronas ocultas. En la figura 2.7(d) se ilustra otra clase de red neuronal recurrente con neuronas ocultas. La presencia de una estructura recurrente tiene un profundo impacto en la capacidad de aprendizaje de la red neuronal. [10].

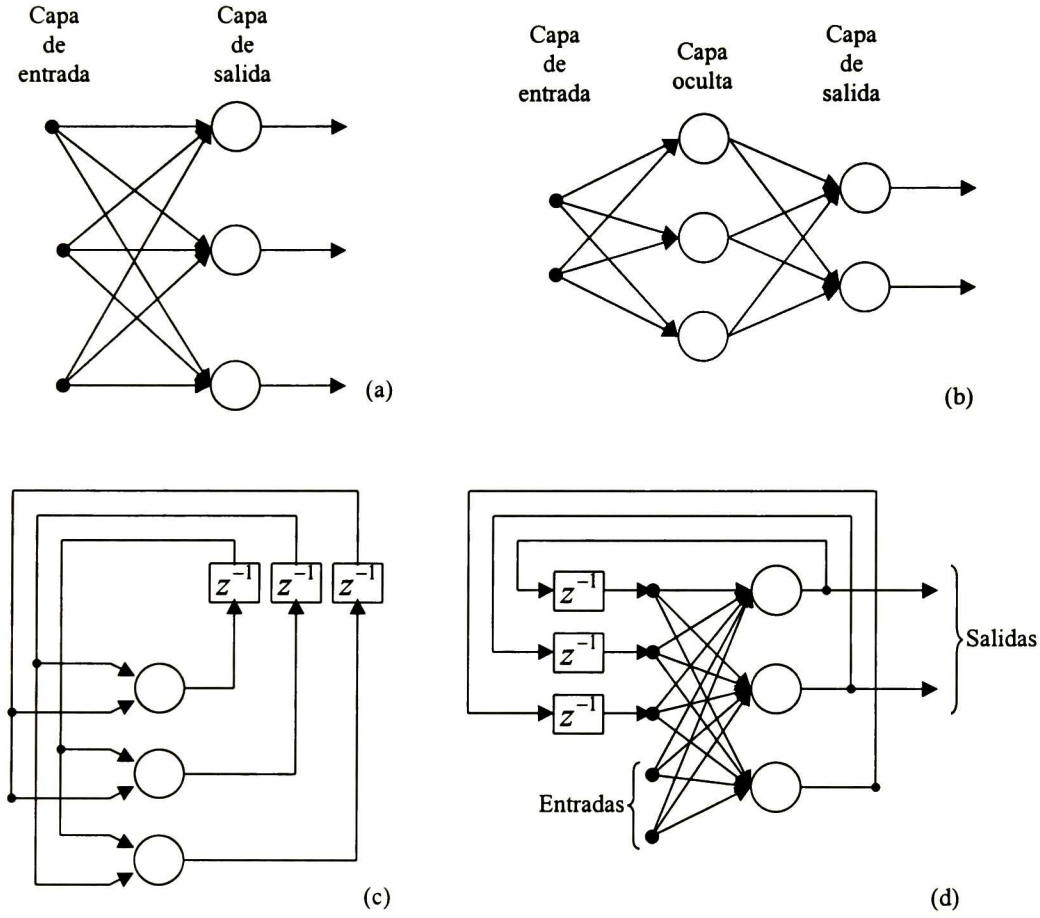


Figura 2.7 Estructuras neuronales; (a) Red neuronal *Unicapa*; (b) Red neuronal *multicapa*; (c) Red neuronal *recurrente sin autolazos*; (d) Red neuronal *recurrente con neuronas ocultas*.

2.5 El Perceptrón

La red tipo Perceptrón fue propuesta por el psicólogo Rosenblatt en el año 1958 [30]. Su intención era ilustrar algunas propiedades fundamentales de los sistemas inteligentes en general, sin entrar en mayores detalles con respecto a condiciones específicas y desconocidas para organismos biológicos concretos [10].

El Perceptrón es la forma más simple de una red neuronal usada para la clasificación de patrones que sean linealmente separables. Básicamente, consiste en una neurona con pesos sinápticos ajustables y una polarización. El Perceptrón de Rosenblatt fue el primer modelo de aprendizaje supervisado. Así mismo Rosenblatt probó que si los patrones utilizados para entrenar el Perceptrón pertenecían a dos clases linealmente separables, entonces el algoritmo del Perceptrón converge y posiciona la superficie de decisión como un hiperplano entre las dos clases [10].

El Perceptrón es construido con base en el modelo neuronal McCulloc-Pitts, según se muestra en la figura 2.8, donde la salida está dada por:

$$\begin{aligned} y &= \varphi(v) = sign(v) \\ v &= \sum_{i=0}^{m} w_i x_i \end{aligned} \tag{2.11}$$

Para la adaptación de los pesos sinápticos del Perceptrón $w_1, w_2, \cdots, w_m$ se puede utilizar la regla de corrección de error propuesta por Rosenblatt como sigue:

$$\begin{aligned} &w(k+1) = w(k) \qquad \text{si } w^T x \geq 0, \ \forall x \in C_1 \text{ o si } w^T x < 0, \ \forall x \in C_2 \\ &w(k+1) = w(k) - \eta(k) x(k) \qquad \text{si } w^T x \geq 0, \ \forall x \in C_2 \\ &w(k+1) = w(k) + \eta(k) x(k) \qquad \text{si } w^T x < 0, \ \forall x \in C_1 \end{aligned} \tag{2.12}$$

donde $\eta(k)$ es la razón de aprendizaje; C_1 y C_2 son clases linealmente separables. Considerando la salida del Perceptrón y la salida deseada, es posible definir el error como:

$$\begin{aligned} e(k) &= d(k) - y(k) \\ d(k) &= \begin{cases} +1 & x(k) \in C_1 \\ -1 & x(k) \in C_2 \end{cases} \end{aligned} \tag{2.13}$$

con $d(k)$ la salida deseada y $x(k)$ el vector de entradas. Entonces la ley de adaptación de pesos queda como:

$$w(k+1) = w(k) + \eta(k) e(k) x(k) \tag{2.14}$$

donde $w(k)$ es el vector de pesos. Esta adaptación de pesos puede lograrse utilizando un algoritmo de mínimos cuadrados o un algoritmo de gradiente descendiente, entre otros [10].

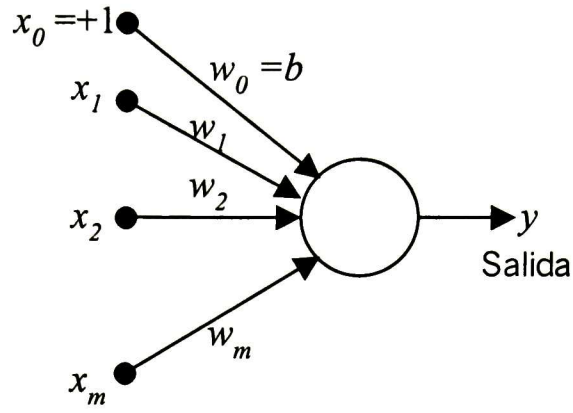


Figura 2.8 El Perceptrón

2.6 El Perceptrón Multicapa

En 1969 Minsky y Papert publicaron su libro *"Perceptrons: An introduction to computational geometry"* [22], el cual significó un gran estancamiento en la teoría de redes neuronales. En él, se presentaba un análisis detallado del Perceptrón, en términos de sus capacidades y limitaciones, en especial en cuanto a las restricciones que existen para los problemas que una red tipo Perceptrón puede resolver; la mayor desventaja de este tipo de redes es su incapacidad para resolver problemas de clasificación que no sean linealmente separables [10].

El Perceptrón multicapa, inicialmente desarrollado por P. Werbos (1974) permite resolver este problema [41]. Posee una estructura con al menos una capa oculta; y su algoritmo de entrenamiento es del tipo corrección de error. Implementa el gradiente distribuido en los diferentes componentes de la red.

Las redes neuronales tipo Perceptrón multicapa (*MLP*, por su nombre en ingles "Multilayer Perceptron"), han sido aplicadas satisfactoriamente para resolver muy diversos y difíciles problemas por medio del algoritmo conocido como *retropropagación* este algoritmo consta de dos etapas:

1. *Etapa hacia adelante.*
Parámetros de la red fijos. Se presenta la señal de entrada a la red, que se propaga hacia adelante para producir la salida.

2. *Etapa hacia atrás.*
El error entre la salida deseada y la red se propaga hacia atrás. Los parámetros de la red se modifican para minimizar el cuadrado de dicho error.

El *MLP* tiene tres características distintivas:

1. El modelo de cada neurona en la red incluye una función de activación no lineal. Lo importante aquí es que la no-linealidad es suave (diferenciable en cualquier punto), lo opuesto a la función signo utilizada en el Perceptrón de Rosenblatt. Una de las funciones de activación más utilizadas es la logística (2.8) [10]. Más aún, el uso de esta función de activación tiene motivos biológicos.
2. La red contiene una o más capas ocultas que no son parte de las entradas o las salidas de la red. Estas neuronas ocultas permiten que la red "aprenda" tareas complejas por la extracción progresiva de las características principales de los patrones de entrada
3. La red presenta altos grados de conectividad, determinados por las sinapsis de la red.

La combinación de estas características junto a la habilidad de aprender de la experiencia a través del entrenamiento del *MLP* da como resultado un gran potencial de computación. En la figura 2.9 se presenta una red tipo *MLP*.

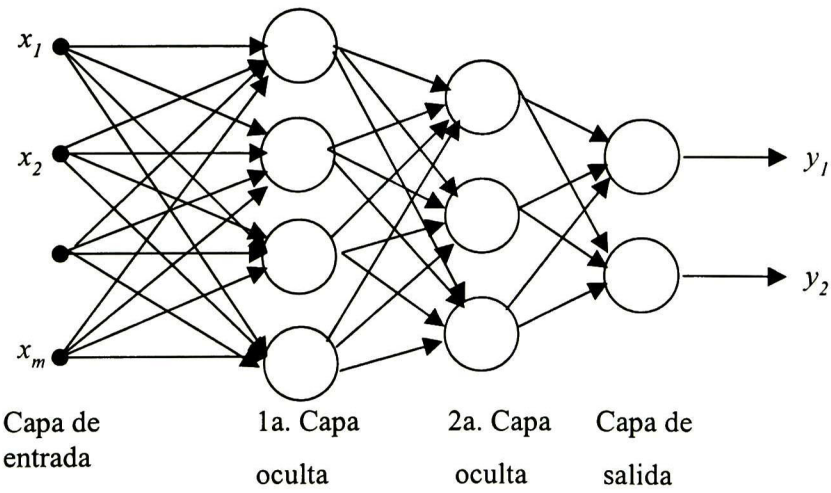


Figura 2.9 Red neuronal tipo *MLP*.

Numerosos estudios han demostrado que las redes neuronales multicapa con solo una capa oculta son capaces de aproximar cualquier función uniformemente continua sobre un dominio compacto por la simple adaptación de los pesos sinápticos, de tal manera que se minimice un criterio deseado: error entre la salida de la red neuronal y la salida deseada [3], [31]

2.7 Redes Neuronales de Base Radial

Las funciones de base radial fueron introducidas primero para la solución de problemas de interpolación multivariable. El trabajo pionero en este tema fue de Powell (1985) [28]. Broomhead y Lowe (1988) [1] fueron los primeros en explotar el uso de funciones de base radial en el diseño de redes neuronales [10].

La justificación matemática la da el teorema de Cover [4], que establece que un problema de *clasificación* es más probable que sea *linealmente separable*, si se transforma en uno de dimensión mayor que la dimensión del espacio en que fue planteado dicho problema [10].

Esta estructura de redes neuronales consta de tres capas bien definidas:

- La capa de nodos de entrada
- La capa oculta de neuronas que proveen una transformación no lineal de la entrada por medio de funciones de base radial
- La capa de salida que es una combinación lineal de la salida de la capa oculta

Una función de base radial es una función multidimensional cuya salida depende de la distancia entre el vector de entradas x y un centro c previamente definido. Usualmente, esta distancia se formula como:

$$d = \sqrt{(x-c)^T (x-c)} \tag{2.15}$$

Existen distintas funciones de base radial, tales como la función cúbica, función Gaussiana, la multicuadrática y la multicuadrática inversa, entre otras [33]. Cuando una red neuronal de base radial (*RBF* por su nombre en ingles "Radial Basis Functions") es usada para la clasificación, resuelve el problema llevándolo a un espacio de dimensión mayor. Un esquema general de *RBF* es presentado en la figura 2.10. Su modelo matemático está dado por:

$$\begin{aligned} y = F(x) &= \sum_{i=1}^{m} w_i \varphi(\mathrm{x}) \\ \varphi(\mathrm{x}) &= G\left(\|x - c_i\|_{r_i}^2\right) \end{aligned} \tag{2.16}$$

donde w_i son los pesos; $G(\bullet)$ es la función de base radial; $x = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}^T$ es la entrada; $c_i \in \Re^n$ son los centros; $r_i \in \Re$ son los radios y $\|x - c_i\|_{r_i}^2 = (x-c)^T (x-c)/r_i$.

El algoritmo de aprendizaje incluye leyes de adaptación tanto de pesos como de centros; leyes que están basadas en minimizar el error medio cuadrático en función de estos parámetros.

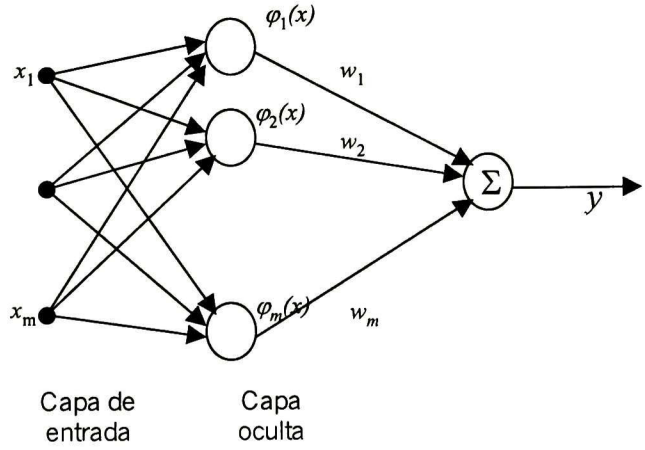


Figura 2.10 Red neuronal de base radial.

Capítulo 3.

EL FILTRO DE KALMAN.

En este capítulo, se presenta una introducción al filtro de Kalman, para fundamentar su aplicación en los capítulos siguientes. El tan elogiado filtro de Kalman, formulado como un sistema dinámico lineal en ecuaciones en espacio de estado, provee una solución al problema lineal de filtrado óptimo. Éste se aplica tanto en ambientes estacionarios como en no estacionarios. La solución es recursiva, de tal forma que cada actualización del estado estimado es calculada a partir del estimado anterior y del nuevo dato de entrada, así solo se requiere almacenar el estimado anterior. Esto significa que no es necesario almacenar todos los datos pasados.

3.1 Conceptos introductorios.

Considérese el sistema dinámico lineal, en tiempo discreto, mostrado en la figura 3.1. El concepto de *estado* es fundamental para esta descripción. El *vector de estado* o simplemente *estado*, denotado por w_k es definido como el conjunto mínimo de datos suficientes para describir el comportamiento dinámico del sistema; el subíndice k denota el instante de muestreo. En otras palabras, el estado es la cantidad mínima de datos del comportamiento pasado del sistema que es necesaria para *predecir* el comportamiento futuro [11]. Típicamente el vector de estados w_k es *desconocido*. Para estimarlo, se usa un conjunto de datos *medidos*, denotados por y_k. En términos matemáticos, el diagrama a bloques de la figura 3.1 involucra el siguiente par de ecuaciones:

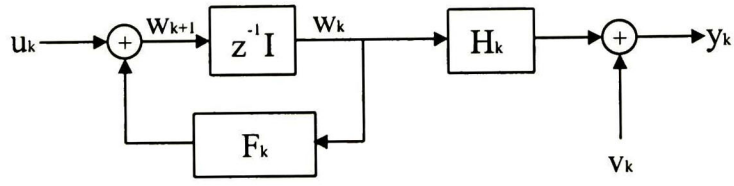


Figura 3.1 Sistema dinámico lineal en tiempo discreto.

Ecuación del proceso.

$$w_{k+1} = F_{k+1,k} w_k + u_k \tag{3.1}$$

donde:

$F_{k+1} \in \Re^{L \times L}$ es la matriz de transición de estados del tiempo k al tiempo $k+1$; $w_k \in \Re^L$ es el vector de estados al tiempo k; $u_k \in \Re^L$ representa el vector del ruido del proceso (Gaussiano con media cero).

La matriz de covarianza del ruido de proceso se define como :

$$E\left\{u_n u_k^T\right\} = \begin{cases} Q_k \text{ para n = k} \\ 0 \text{ para n} \neq \text{k} \end{cases} \tag{3.2}$$

Ecuación de la salida.

$$y_k = H_k w_k + v_k \tag{3.3}$$

donde:

$H_k \in \Re^{m \times L}$ es la matriz de medición; $y_k \in \Re^m$ es el vector de salida del proceso (observable al tiempo k); $v_k \in \Re^m$ representa el vector del ruido de medición (Gaussiano con media cero).

La matriz de covarianza del ruido de medición se define como :

$$E\left\{v_n v_k^T\right\} = \begin{cases} R_k \text{ para n = k} \\ 0 \text{ para n} \neq \text{k} \end{cases} \tag{3.4}$$

Además el ruido de medición está descorrelacionado del ruido del proceso.

Estimación óptima.

Antes de derivar las ecuaciones del filtro de Kalman, se considera útil revisar algunos conceptos básicos de estimación óptima. Considérese la siguiente ecuación:

$$y_k = w_k + v_k$$

donde w_k es una señal desconocida y v_k es un ruido aditivo. Sea $\hat{w}_k$ el estimado a posteriori de la señal w_k, dadas las mediciones $y_1, y_2, ..., y_k$. En general, el estimado $\hat{w}_k$ es diferente de la señal desconocida w_k [11], [16].

Para deducir este estimado de una forma óptima, se necesita una función de costo (pérdida) [11], que debe satisfacer los siguientes requisitos:

- La función de costo es no-negativa
- La función de costo es una función no-decreciente del error de estimación, definido por:

$$\tilde{w}_k = w_k - \hat{w}_k$$

Estos dos requisitos se satisfacen por la esperanza del error al cuadrado, definido por:

$$J_k = E\left\{\left(w_k - \hat{w}_k\right)^2\right\}$$
$$J_k = E\left\{\tilde{w}_k^2\right\}$$

donde E es el operador del valor esperado. La dependencia de la función de costo J_k del tiempo k (instante de muestreo), enfatiza la naturaleza no estacionaria del proceso recursivo de estimación.

Para deducir un valor óptimo para el estimado $\hat{w}_k$, se utilizan los siguientes teoremas [16].

Teorema 1.1. Estimador de esperanza condicional. Si los procesos estocástico $\{w_k\}$ y $\{y_k\}$ son Gaussianos, entonces el estimado óptimo $\hat{w}_k$ que minimiza el error medio cuadrático J_k es el estimador de esperanza condicional:

$$\hat{w}_k = E\left\{w_k \middle| y_1, y_2, ..., y_k\right\}$$

Teorema 1.2. Principio de ortogonalidad. Sean los procesos estocástico $\{w_k\}$ y $\{y_k\}$ con media cero, tal que:

$$E\{w_k\} = E\{y_k\} = 0 \quad \forall \ k$$

Y ademas

i) Los procesos estocásticos $\{w_k\}$ y $\{y_k\}$ son Gaussianos; o

ii) El estimado óptimo $\hat{w}_k$ esta restringido a ser una función lineal de las mediciones $y_1, y_2, ..., y_k$ y de la función de costo del error medio cuadrático,

iii) Entonces el estimado óptimo $\hat{w}_k$, dadas las mediciones $y_1, y_2, ..., y_k$, es ortogonal a la proyección de w_k en el espacio generado por dichas mediciones.

3.2 Filtro de Kalman.

Se desea usar la información contenida en la nueva medición y_k para actualizar el estimado del estado desconocido w_k. Sea $\hat{w}_k^-$ el estimado a priori del estado que está disponible al tiempo k [11]. Podemos expresar el estimado a posteriori $\left(\hat{w}_k\right)$ como una combinación lineal del estimado a priori y la nueva medición, como sigue:

$$\hat{w}_k = L_k \hat{w}_k^- + K_k y_k \tag{3.5}$$

donde L_k y K_k son matrices cuyos elementos son valores constantes. El vector de error de estimación se define como:

$$\tilde{w}_k = w_k - \hat{w}_k \tag{3.6}$$

Aplicando el principio de ortogonalidad podemos escribir:

$$E\left\{\tilde{w}_k y_i^T\right\} = 0 \text{ para i = 1,2, ... ,k-1} \tag{3.7}$$

Usando (3.3), (3.5) y (3.6) en (3.7) se tiene:

$$E\left\{\left(w_k - L_k \hat{w}_k^- - K_k H_k w_k - K_k v_k\right) y_i^T\right\} = 0 \text{ para i = 1,2, ... ,k-1} \tag{3.8}$$

La solución a la ecuación de estado (3.1) al tiempo i es:

$$w_i = F_{i,0} w_0 + \sum_{j=1}^{i-1} F_{i,j+1} u_j \tag{3.9}$$

donde w_0 es el valor inicial del estado. De la ecuación (3.9) se tiene que w_i es una combinación lineal de w_0 y u_1 , u_2 ... , u_{i-1}.

Por hipótesis, el vector de ruido v_i es descorrelacionado tanto del vector de estado inicial como del vector de ruido u_i. Multiplicando ambos lados de (3.9) por v_k^T y tomando los valores esperados, se deduce que:

$$E\left\{w_i v_k^T\right\} = 0 \tag{3.10}$$

Así mismo, a partir de la ecuación (3.3). Multiplicando ambos lados de dicha ecuación por v_k^T tomando los valores esperados y considerando (3.10) y (3.4) se tiene:

$$E\left\{y_i v_k^T\right\} = 0 \tag{3.11}$$

Usando esta relación y reacomodando términos, la ecuación (3.8) se puede escribir como:

$$E\left\{\left(I - K_k H_k - L_k\right) w_k y_i^T + L_k\left(w_k - \hat{w}_k^-\right) y_i^T\right\} = 0 \tag{3.12}$$

donde I es la matriz identidad. Ahora del principio de ortogonalidad:

$$E\left\{\left(w_k - \hat{w}_k^-\right) y_i^T\right\} = 0$$

De acuerdo a la expresión anterior, (3.12) se puede simplificar como:

$$\left(I - K_k H_k - L_k\right) E\left\{w_k y_i^T\right\} = 0 \quad \text{para i = 1,2, ... , k-1} \tag{3.13}$$

Para valores arbitrarios de w_k y y_i, la ecuación (3.13) sólo puede ser satisfecha si los factores L_k y K_k son tales que:

$$I - K_k H_k - L_k = 0$$

o equivalentemente:

$$L_k = I - K_k H_k \tag{3.14}$$

Sustituyendo (3.14) en (3.5) se puede expresar el estado estimado a posteriori al tiempo k como:

$$\hat{w}_k = \hat{w}_k^- + K_k\left(y_k - H_k \hat{w}_k^-\right) \tag{3.15}$$

Debido a esto K_k, es llamada la ganancia de Kalman. Ahora sólo queda por resolver el problema de determinar una fórmula explícita para K_k.

A partir del principio de ortogonalidad se tiene:

$$E\left\{\left(w_k - \hat{w}_k\right) y_k^T\right\} = 0 \tag{3.16}$$

de lo cual se sigue:

$$E\left\{\left(w_k - \hat{w}_k\right) \hat{y}_k^T\right\} = 0 \tag{3.17}$$

donde $\hat{y}_k$ es un estimado de y_k dadas las mediciones anteriores $y_1, y_2, ..., y_{k-1}$. Definiendo las innovaciones del proceso como:

$$\tilde{y}_k = y_k - \hat{y}_k \tag{3.18}$$

la innovación del proceso representa una medición de la nueva información contenida en y_k, esto puede ser expresado como:

$$\begin{aligned} \tilde{y}_k &= y_k - H_k \hat{w}_k^- \\ \tilde{y}_k &= H_k w_k + v_k - H_k \hat{w}_k^- \\ \tilde{y}_k &= H_k \tilde{w}_k^- + v_k \end{aligned} \tag{3.19}$$

con $\tilde{w}_k^- = w_k - \hat{w}_k^-$

Entonces sustrayendo (3.17) de (3.16) y después usando (3.18), podemos escribir:

$$E\left\{\left(w_k - \hat{w}_k\right)\tilde{y}_k^T\right\} = 0 \tag{3.20}$$

Usado las ecuaciones (3.3) y (3.15), podemos expresar el vector de error de predicción como:

$$\begin{aligned} w_k - \hat{w}_k &= \tilde{w}_k^- - K_k\left(H_k \tilde{w}_k^- + v_k\right) \\ w_k - \hat{w}_k &= \left(I - K_k H_k\right)\tilde{w}_k^- - K_k v_k \end{aligned} \tag{3.21}$$

con $\tilde{w}_k^- = w_k - \hat{w}_k^-$

Entonces sustituyendo (3.19) y (3.21) en (3.20) se tiene:

$$E\left\{\left[\left(I - K_k H_k\right)\tilde{w}_k^- - K_k v_k\right]\left(H_k \tilde{w}_k^- + v_k\right)^T\right\} = 0 \tag{3.22}$$

Debido a que el ruido de medición v_k es independiente del estado w_k y por lo tanto del error $\tilde{w}_k^-$, la esperanza de la ecuación (3.22) se reduce a:

$$\left(I - K_k H_k\right)E\left\{\tilde{w}_k^- \tilde{w}_k^{-T}\right\}H_k^T - K_k E\left\{v_k v_k^T\right\} = 0 \tag{3.23}$$

Definiendo la matriz de covarianza a priori:

$$\begin{aligned} P_k^- &= E\left\{\left(w_k - \hat{w}_k^-\right)\left(w_k - \hat{w}_k^-\right)^T\right\} \\ P_k^- &= E\left\{\tilde{w}_k^- \tilde{w}_k^{-T}\right\} \end{aligned} \tag{3.24}$$

Entonces involucrando las definiciones de covarianza de (3.4) y (3.24) podemos rescribir (3.23) como:

$$(I - K_k H_k) P_k^- H_k^T - K_k R_k = 0$$

Resolviendo esta ecuación para K_k, obtenemos la fórmula.

$$K_k = P_k^- H_k^T \left[H_k P_k^- H_k^T + R_k \right]^{-1} \tag{3.25}$$

La ecuación (3.25) es la fórmula para calcular la ganancia de Kalman K_k, la cual es definida en términos de la matriz de covarianza a priori P_k^-

Para completar el proceso recursivo de estimación, debemos considerar la propagación de la covarianza del error, lo que describe los efectos del tiempo en las matrices de covarianza de los errores de estimación. Esta propagación involucra dos pasos de cálculo [11].

Paso 1. La matriz de covarianza a priori P_k^- al tiempo *k* esta definida por la ecuación (3.24) Dada P_k^- calcular la matriz de covarianza a posteriori P_k la cual al tiempo *k* esta definida por:

$$\begin{aligned} P_k &= E\left\{ \tilde{w}_k \tilde{w}_k^T \right\} \\ P_k &= E\left\{ (w_k - \hat{w}_k)(w_k - \hat{w}_k)^T \right\} \end{aligned} \tag{3.26}$$

Paso2. Dada la matriz de covarianza a posteriori anterior P_{k-1}, calcular la matriz de covarianza a priori actualizada P_k^-

Para proceder con el paso 1, sustituimos (3.21) en (3.26) y dado que el ruido de medición v_k es independiente del error de estimación a priori $\tilde{w}_k^-$. Se obtiene:

$$\begin{aligned} P_k &= (I - K_k H_k) E\left\{ \tilde{x}_k^- \tilde{x}_k^{-T} \right\} (I - K_k H_k)^T + K_k E\left\{ v_k v_k^T \right\} K_k^T \\ P_k &= (I - K_k H_k) P_k^- (I - K_k H_k)^T + K_k R_k K_k^T \end{aligned} \tag{3.27}$$

Expandiendo términos en la ecuación (3.27) y usando la ecuación (3.25), podemos reformular la dependencia de la matriz de covarianza a posteriori P_k, de la matriz de covarianza a priori P_k^-, en la siguiente forma simplificada:

$$\begin{aligned} P_k &= (I - K_k H_k) P_k^- - (I - K_k H_k) P_k^- H_k^T K_k^T + K_k R_k K_k^T \\ P_k &= (I - K_k H_k) P_k^- - K_k R_k K_k^T + K_k R_k K_k^T \\ P_k &= (I - K_k H_k) P_k^- \end{aligned} \tag{3.28}$$

Para el segundo paso de la propagación de la covarianza del error, primero reconocemos que el estimado del estado a priori es definido en términos del estimado a posteriori anterior, como sigue:

$$\hat{w}_k^- = F_{k,k-1}\hat{w}_{k-1} \tag{3.29}$$

Ahora podemos expresar el error de estimación a priori como:

$$\begin{aligned} \tilde{w}_k^- &= w_k - \hat{w}_k^- \\ \tilde{w}_k^- &= \left(F_{k,k-1}w_{k-1} + u_{k-1}\right) - \left(F_{k,k-1}\hat{w}_{k-1}\right) \\ \tilde{w}_k^- &= F_{k,k-1}\left(w_{k-1} - \hat{w}_{k-1}\right) + u_{k-1} \\ \tilde{w}_k^- &= F_{k,k-1}\tilde{w}_{k-1} + u_{k-1} \end{aligned} \tag{3.30}$$

Usando (3.30) en (3.24) y notando que el ruido del proceso u_k es independiente de $\tilde{w}_{k-1}$ se tiene:

$$\begin{aligned} P_k^- &= F_{k,k-1}E\left\{\tilde{w}_{k-1}\tilde{w}_{k-1}^T\right\}F_{k,k-1}^T + E\left\{u_{k-1}u_{k-1}^T\right\} \\ P_k^- &= F_{k,k-1}P_{k-1}F_{k,k-1}^T + Q_{k-1} \end{aligned} \tag{3.31}$$

Lo que define la dependencia de la matriz de covarianza a priori P_k^- de la matriz de covarianza a posteriori anterior P_{k-1}. En ausencia de datos medidos para $k=0$, podemos escoger el estado estimado inicial como:

$$\hat{w}_0 = E\left\{w_0\right\} \tag{3.32}$$

y el valor inicial de la matriz de covarianza a posteriori como :

$$P_0 = E\left\{\left(w_0 - E\left\{w_0\right\}\right)\left(w_0 - E\left\{w_0\right\}\right)^T\right\} \tag{3.33}$$

Esta selección para las condiciones iniciales no es solo intuitivamente satisfactoria, tiene la ventaja de mantener el estado sin polarización [11].

3.2.1 Resumen del filtro de Kalman.

- *Modelo en espacio de estado*

  $$\begin{aligned} w_{k+1} &= F_{k+1,k}w_k + u_k \\ y_k &= H_k w_k + v_k \end{aligned}$$

  donde u_k y v_k son ruidos independientes, Gaussianos, con matrices de covarianza Q_k y R_k respectivamente.

- *Inicialización para k=0:*

$$\hat{w}_0 = E\{w_0\}$$
$$P_0 = E\left\{\left(w_0 - E\{w_0\}\right)\left(w_0 - E\{w_0\}\right)^T\right\}$$

- *Cálculo:* para *k=1,2,... calcular:*

  - Propagación del estado estimado
  $$\hat{w}_k^- = F_{k,k-1}\hat{w}_{k-1}^-$$
  Propagación de la covarianza del error
  $$P_k^- = F_{k,k-1}P_{k-1}F_{k,k-1}^T + Q_{k-1}$$
  - Matriz de ganancia de Kalman
  $$K_k = P_k^- H_k^T \left[H_k P_k^- H_k^T + R_k\right]^{-1}$$
  - Actualización del estado estimado
  $$\hat{w}_k = \hat{w}_k^- + K_k\left(y_k - H_k\hat{w}_k^-\right)$$
  - Actualización de la covarianza del error
  $$P_k = \left(I - K_k H_k\right)P_k^-$$

3.3 Filtro de Kalman Extendido (*FKE*).

El filtro de Kalman estudiado hasta aquí, supone un modelo lineal de un sistema dinámico. Sin embargo usualmente, el modelo es no lineal; a continuación se extenderá el uso del filtro de Kalman a través de un procedimiento de linealización. Este filtro de Kalman resultante es conocido como Filtro de Kalman Extendido (*FKE*). Tal extensión es factible en virtud de que el filtro de Kalman es descrito en términos de ecuaciones en diferencias, en el caso de sistemas en tiempo discreto [11].

Considérese un sistema dinámico no lineal descrito por el siguiente modelo en espacio de estado.

$$w_{k+1} = f\left(k, w_k\right) + u_k \tag{3.34}$$
$$y_k = h\left(k, w_k\right) + v_k \tag{3.35}$$

Como antes u_k y v_k son ruidos independientes, Gaussianos con matrices de covarianza R_k y Q_k respectivamente. $f\left(k, w_k\right)$ denota la función matricial no lineal de transición, posiblemente variante en el tiempo y $h\left(k, w_k\right)$ denota la función matricial de medición no lineal que también puede ser variante en el tiempo.

La idea básica del Filtro de Kalman Extendido es linealizar el modelo en espacio de estado de las ecuaciones (3.34) y (3.35) a cada instante de tiempo alrededor del estado estimado más reciente, el cual puede ser tomado como $\hat{w}_k$ o $\hat{w}_k^-$, respectivamente. Una vez que el modelo lineal es obtenido, las ecuaciones del Filtro de Kalman son aplicadas [11].

Más explícitamente, la aproximación procede en dos pasos:

Paso 1. Se construyen las siguientes matrices:

$$F_{k+1,k} = \left.\frac{\partial f(k, w_k)}{\partial w}\right|_{w=\hat{w}_k} \tag{3.36}$$

$$H_k = \left.\frac{\partial h(k, w_k)}{\partial w}\right|_{w=\hat{w}_k^-} \tag{3.37}$$

Paso 2. Una vez que las matrices $F_{k+1,k}$ y H_k son evaluadas, se emplean en una aproximación en series de Taylor de primer orden para las funciones no lineales $f(k, w_k)$ y $h(k, w_k)$ alrededor de $\hat{w}_k$ y $\hat{w}_k^-$ respectivamente. Específicamente, se aproximan como sigue:

$$f(k, w_k) \approx f(w, \hat{w}_k) + F_{k+1,k}(w, \hat{w}_k) \tag{3.38}$$

$$h(k, w_k) \approx h(w, \hat{w}_k^-) + H_{k+1,k}(w, \hat{w}_k^-) \tag{3.39}$$

Con (3.38) y (3.39), procedemos a aproximar las ecuaciones de estado no lineales (3.34) y (3.35) como se muestra:

$$w_{k+1} \approx F_{k+1,k} w_k + u_k + d_k$$

$$\bar{y}_k \approx H_k w_k + v_k$$

Ahora introduciremos dos nuevas variables:

$$\bar{y}_k = y_k - \left\{h(w, \hat{w}_k^-) - H_k \hat{w}_k^-\right\} \tag{3.40}$$

$$d_k = f(w, \hat{w}_k) - F_{k+1,k} \hat{w}_k \tag{3.41}$$

Dadas las ecuaciones del modelo en espacio de estado linealizadas, se procede a realizar el mismo procedimiento que en el caso del filtro de Kalman [11].

3.3.1 Resumen del filtro de Kalman Extendido.

- *Modelo en espacio de estado.*

$$w_{k+1} = f(k, w_k) + u_k$$
$$y_k = h(k, w_k) + v_k$$

donde u_k y v_k son ruidos independientes, Gaussianos, con matrices de covarianza Q_k y R_k respectivamente.

- *Definiciones.*

$$F_{k+1,k} = \left.\frac{\partial f(k, w_k)}{\partial w}\right|_{w=\hat{w}_k}$$
$$H_k = \left.\frac{\partial h(k, w_k)}{\partial w}\right|_{w=\hat{w}_k^-}$$

- *Inicialización para k=0:*

$$\hat{w}_0 = E\{w_0\}$$
$$P_0 = E\left\{(w_0 - E\{w_0\})(w_0 - E\{w_0\})^T\right\}$$

- *Cálculos:* Para *k= 1,2,.... calcular*

  - Propagación del estado estimado
  $$\hat{w}_k^- = f(k, \hat{w}_{k-1})$$
  - Propagación de la covarianza del error
  $$P_k^- = F_{k,k-1} P_{k-1} F_{k,k-1}^T + Q_{k-1}$$
  Matriz de la ganancia de Kalman
  $$K_k = P_k^- H_k^T \left[H_k P_k^- H_k^T + R_k\right]^{-1}$$
  - Actualización del estado estimado
  $$\hat{w}_k = \hat{w}_k^- + K_k y_k - h(k, \hat{w}_k^-)$$
  Actualización de la covarianza del error
  $$P_k = (I - K_k H_k) P_k^-$$

3.3.2 El FKE Simplificado.

El uso del *FKE* en la estimación de estados es bien conocido, ([9], [16], [24], [35], entre otros). En ocasiones el algoritmo presentado en la sección anterior resulta bastante complicado de implementar, por lo cual en trabajos como: [15], [24], [29], [39], [44], etc, se recomienda replantearlo, para lo cual proponen la siguiente simplificación:

$$\hat{w}_k^- = \hat{w}_{k-1}$$
$$F_{k,k-1} = I \tag{3.42}$$

Tomando cuenta (3.42) en las ecuaciones del *FKE*, éste queda como:

$$K_k = P_k^- H_k^T \left[H_k P_k^- H_k^T + R_k \right]^{-1} \tag{3.43}$$
$$\hat{w}_k = \hat{w}_k^- + K_k \left[y_k - \hat{y}_k \right] \tag{3.44}$$
$$P_k = P_{k-1} - K_{k-1} H_{k-1} P_{k-1} + Q_{k-1} \tag{3.45}$$

donde $P_k \in \Re^{LxL}$ y $P_{k-1} \in \Re^{LxL}$ representan la matriz de covarianza del error de predicción al tiempo k y k-1 respectivamente; $w \in \Re^L$ es el vector de estados; L es el número total de estados; $y_k \in \Re^m$ es el vector de salidas, donde m es el número total de salidas; $\hat{y} \in \Re^m$ es la salida estimada del sistema; $K \in \Re^{Lxm}$ es la matriz de ganancia de Kalman; $Q \in \Re^{LxL}$ es la matriz de covarianza del ruido del proceso; $R \in \Re^{mxm}$ representa la matriz de covarianza del ruido de medición; finalmente $H \in \Re^{mxL}$ es la matriz que contiene las derivadas de cada salida del sistema estimado ($\hat{y}_k$), con respecto a cada uno de los parametros w_k, tal y como se define a continuación:

$$H_k = \left[\frac{\partial \hat{y}_k}{\partial w_k} \right]_{w_k = \hat{w}_{k+1}} \tag{3.46}$$

3.4 Estabilidad del filtro de Kalman.

Considérese el sistema dinámico lineal en tiempo discreto mostrado en la figura 3.1, [9], que se representa por:

Ecuación del proceso.

$$w_{k+1} = F_k w_k + u_k \tag{3.47}$$

Ecuación de la salida.

$$y_k = H_k w_k + v_k \tag{3.48}$$

Ecuación del filtro de Kalman.

$$\hat{w}_k = \hat{w}_k^- + K_k \left(y_k - H_k \hat{w}_k^- \right) \tag{3.49}$$

donde:

F_k = Matriz de transición de estado del tiempo k al tiempo $k+1$
w_k = Vector de estado

u_k = Ruido del proceso (Gaussiano, Media cero)

y_k = Salida del proceso al tiempo k

H_k = Matriz de medición

v_k = Ruido de medición (Gaussiano, Media cero)

Análisis de estabilidad del filtro de Kalman discreto.

Tomando (3.49) y reemplazando $\hat{w}_k^-$ con $F_{k-1}\hat{w}_{k-1}$ se tiene:

$$\begin{aligned} \hat{w}_k &= F_{k-1}\hat{w}_{k-1} + K_k\left(y_k - H_k F_{k-1}\hat{w}_{k-1}\right) \\ \hat{w}_k &= \left(F_{k-1} - K_k H_k F_{k-1}\right)\hat{w}_{k-1} + K_k y_k \end{aligned} \tag{3.50}$$

Calculando la transformada **z** a ambos lados de (3.50) y recordando que retardar $\hat{w}_k$ un paso en el tiempo equivale a multiplicarlo por $\mathbf{z^{-1}}$, en el dominio de **z** resulta:

$$\hat{W}(z) = \left(F_k - K_k H_k F_k\right) z^{-1}\hat{W}(z) + K_k Y_k$$

Reacomodando términos:

$$\left[zI - \left(F_k - K_k H_k F_k\right)\right]\hat{W}(z) = zK_k Y_k \tag{3.51}$$

Finalmente el polinomio característico es:

$$\left[zI - \left(F_k - K_k H_k F_k\right)\right] \tag{3.52}$$

y las raíces de este polinomio determinan la estabilidad del filtro de Kalman discreto, que además corresponden a los valores propios de :

$$\left(F_k - K_k H_k F_k\right) \tag{3.53}$$

Por lo tanto la condición de estabilidad queda establecida si los valores propios de (3.53) están dentro del círculo unitario.

3.5 El Filtro de Kalman Extendido Libre de Derivadas (*FKELD*).

El *FKELD* es derivado reemplazando una expansión en series de Taylor, en la vecindad de un vector de estados, por una expansión basada en la fórmula de interpolación de Stirling [24].

Sean los operadores μ y δ, con h denotando la longitud del intervalo seleccionado.

$$\delta f(w) = f\left(w+\frac{h}{2}\right) - f\left(w-\frac{h}{2}\right),$$
$$\mu f(w) = \frac{1}{2}\left(f\left(w+\frac{h}{2}\right) + f\left(w-\frac{h}{2}\right)\right) \tag{3.54}$$

Con estos operadores e introduciendo una variable adimensional p, definida como la distancia desde w en unidades de h (3.55), la fórmula de interpolación de Stirling usada alrededor del punto $w = \bar{w}$ (3.56).

$$p = \frac{w-\bar{w}}{h} \tag{3.55}$$

Por lo tanto la *fórmula de interpolación de Stirling* puede expresarse como:

$$f(w) = f(\bar{w}+ph) = f(\bar{w}) + p\mu\delta f(\bar{w}) + \frac{p^2}{2!}\delta^2 f(\bar{w}) + \binom{p+1}{3}\mu\delta^3 f(\bar{w}) + \frac{p^2(s^2-1)}{4!}\delta^4 f(\bar{w}) + \binom{p+2}{5}\mu\delta^5 f(\bar{w}) + \cdots \tag{3.56}$$

Para este caso particular, serán consideradas aproximaciones polinomiales de primer y segundo orden como sigue:

$$f(w) \approx f(\bar{w}) + f'_{DD}(\bar{w})(w-\bar{w}) + \frac{f''_{DD}(\bar{w})}{2!}(w-\bar{w})^2 \tag{3.57}$$

donde:

$$f'_{DD}(\bar{w}) = \frac{f(\bar{w}+h) - f(\bar{w}-h)}{2h}$$
$$f''_{DD}(\bar{w}) = \frac{f(\bar{w}+h) + f(\bar{w}-h) - 2f(\bar{w})}{h^2} \tag{3.58}$$

Nótese que h define el intervalo $\left[\bar{x}-\frac{h}{2}, \bar{x}+\frac{h}{2}\right]$, para el cual la aproximación es satisfecha[13].

La ecuación (3.59) es la extensión multidimensional de la fórmula de interpolación (3.57), [24], [13].

$$y \approx f(\bar{w}) + \tilde{D}_{\Delta w} f + \frac{1}{2!}\tilde{D}^2_{\Delta w} f \tag{3.59}$$

con:

$$\tilde{D}_{\Delta w} f = \frac{1}{h}\left(\sum_{p=1}^{n} \Delta w_p \mu_p \delta_p\right) f(\bar{w})$$
$$\tilde{D}^2_{\Delta w} f = \frac{1}{h^2}\left(\sum_{p=1}^{n} \Delta w_p^2 \mu_p^2 \delta_p^2 + \sum_{p=1}^{n}\sum_{\substack{q=1 \\ q\neq p}}^{n} \Delta w_p \Delta w_q \left(\mu_p \delta_p\right)\left(\mu_q \delta_q\right)\right) f(\bar{w}) \tag{3.60}$$

donde δ_p es un operador de diferencia parcial, μ_p es un operador de promedio y e_p es un vector unitario de longitud p.

$$\delta_p f(\bar{w}) = f\left(\bar{w} + \frac{h}{2} e_p\right) - f\left(\bar{w} - \frac{h}{2} e_p\right)$$
$$\mu_p f(\bar{w}) = \frac{1}{2}\left[f\left(\bar{w} + \frac{h}{2} e_p\right) + f\left(\bar{w} - \frac{h}{2} e_p\right)\right] \tag{3.61}$$

Considérese la siguiente transformación lineal de w.

$$z = S^{-1} w \tag{3.62}$$

donde S es una transformación lineal y la función $\tilde{f}$ está definida como:

$$\tilde{f}(z) \triangleq f(Sz) = f(w) \tag{3.63}$$

Entonces, se puede definir la siguiente expresión:

$$2\mu_p \delta_p \tilde{f}(\bar{z}) = \tilde{f}\left(\bar{z} + h e_p\right) - \tilde{f}\left(\bar{z} - h e_p\right)$$
$$2\mu_p \delta_p \tilde{f}(\bar{z}) = f\left(\bar{w} + h s_p\right) - f\left(\bar{w} - h s_p\right) \tag{3.64}$$

donde s_p denota la p-ésima columna de S. A partir de $\tilde{D}_{\Delta w} f$ y $\tilde{D}^2_{\Delta w} f$ es posible obtener $\tilde{D}_{\Delta z} \tilde{f}$ y $\tilde{D}^2_{\Delta z} \tilde{f}$, respectivamente.

La transformación (3.62) es seleccionada como un factor de Cholesky con $z = S_w^{-1} w$ tal que $P_w = S_w S_w^T$. La transformación seleccionada [24] involucra que los elementos en z sean descorrelacionados y:

$$E\left[\left(z - E[z]\right)\left(z - E[z]\right)^T\right] = I$$

Sea w un vector de variables estocásticas para el cual se define:

$$\bar{w} = E[w] \qquad P_w = E\left[\left(w - \bar{w}\right)\left(w - \bar{w}\right)^T\right] \tag{3.65}$$

$$\bar{y} \triangleq E\left[f(w)\right] \tag{3.66}$$

$$P_y \triangleq E\left[\left(f(w)-\bar{y}\right)\left(f(w)-\bar{y}\right)^T\right] \tag{3.67}$$

$$P_{wy} \triangleq E\left[\left(w-w\right)\left(f(w)-\bar{y}\right)\right] \tag{3.68}$$

También considérense las siguientes expresiones:

$$\Delta z \triangleq z - E\left[z\right] \tag{3.69}$$

Por hipótesis Δz tiene media cero y distribución Gaussiana [24].

Como f es una función no lineal se utilizará su aproximación.

Contémplese una *aproximación de primer orden*:

$$y = \tilde{f}\left(\bar{z}+\Delta z\right) \approx \tilde{f}\left(\bar{z}\right) + \tilde{D}_{\Delta z}\tilde{f} \tag{3.70}$$

donde $\bar{z} = S^{-1}\bar{w}$ y como $E\left[\Delta z\right] = 0$, entonces el valor esperado de (3.70) es:

$$\begin{aligned}
&\bar{y} \triangleq E\left[\tilde{f}\left(\bar{z}\right) + \tilde{D}_{\Delta z}\tilde{f}\right] \\
&\bar{y} = E\left[\tilde{f}\left(\bar{z}\right)\right] + E\left[\tilde{D}_{\Delta z}\tilde{f}\right] \\
&\tilde{D}_{\Delta z} f = \frac{1}{h}\left(\sum_{p=1}^{n} \Delta z_p \mu_p \delta_p\right) f\left(\bar{x}\right) \text{ y como } E\left[\Delta z\right] = 0 \\
&\bar{y} = \tilde{f}\left(\bar{z}\right) = f\left(\bar{x}\right)
\end{aligned} \tag{3.71}$$

Ahora es posible determinar un estimado de la covarianza (3.67). Por las propiedades de la transformación elegida [24]:

$$E\left[\Delta z_i \Delta z_j\right] = 0 \quad \text{con } i \neq j$$

$$\begin{aligned}
P_y &\triangleq E\left[\left(f(w)-\bar{y}\right)\left(f(w)-\bar{y}\right)^T\right] \\
P_y &= E\left[\left(\tilde{f}\left(\bar{z}\right) + \tilde{D}_{\Delta z}\tilde{f} - \tilde{f}\left(\bar{z}\right)\right)\left(\tilde{f}\left(\bar{z}\right) + \tilde{D}_{\Delta z}\tilde{f} - \tilde{f}\left(\bar{z}\right)\right)^T\right] \\
P_y &= E\left[\left(\tilde{D}_{\Delta z}\tilde{f}\right)\left(\tilde{D}_{\Delta z}\tilde{f}\right)^T\right]
\end{aligned}$$

Sustituyendo (3.60) se tiene:

$$P_y = E\left[\left(\sum_{p=1}^{n}\Delta z_p \mu_p \delta_p \tilde{f}(\bar{z})\right)\left(\sum_{p=1}^{n}\Delta z_p \mu_p \delta_p \tilde{f}(\bar{z})\right)^T\right]\frac{1}{h^2}$$

Denótese el i-ésimo momento de Δz como σ_i

$$P_y = \frac{\sigma_2}{h^2}\sum_{p=1}^{n}\left(\mu_p \delta_p \tilde{f}(\bar{z})\right)\left(\mu_p \delta_p \tilde{f}(\bar{z})\right)^T$$

Considerando el caso específico en que $\sigma_2 = 1$ y sustituyendo (3.64), entonces:

$$2\mu_p \delta_p \tilde{f}(\bar{z}) = \tilde{f}\left(\bar{z} + he_p\right) - \tilde{f}\left(\bar{z} - he_p\right)$$

$$P_y = \frac{1}{4h^2}\sum_{p=1}^{n}\left(\tilde{f}\left(\bar{z} + he_p\right) - \tilde{f}\left(\bar{z} - he_p\right)\right)\times\left(\tilde{f}\left(\bar{z} + he_p\right) - \tilde{f}\left(\bar{z} - he_p\right)\right)^T \qquad (3.72)$$

Retomando (3.64) donde $s_{x,p}$ es la p-ésima columna de S_x, (3.72) se puede rescribir como:

$$2\mu_p \delta_p \tilde{f}(\bar{z}) = f\left(\bar{w} + hs_p\right) - f\left(\bar{w} - hs_p\right)$$

$$P_y = \frac{1}{4h^2}\sum_{p=1}^{n}\left(f\left(\bar{w} + hs_{w,p}\right) - f\left(\bar{w} - hs_{w,p}\right)\right)\times\left(f\left(\bar{w} + hs_{w,p}\right) - f\left(\bar{w} - hs_{w,p}\right)\right)^T \qquad (3.73)$$

Un estimado de la matriz de covarianza cruzada puede deducirse de manera similar:

$$P_{wy} \triangleq E\left[\left(w - \bar{w}\right)\left(f(w) - \bar{y}\right)^T\right]$$

$$P_{wy} = E\left[\left(w - \bar{w}\right)\left(\tilde{f}(\bar{z}) + \tilde{D}_{\Delta z}\tilde{f} - \tilde{f}(\bar{z})\right)^T\right]$$

$$P_{wy} = E\left[\left(w - \bar{w}\right)\left(\tilde{D}_{\Delta z}\tilde{f}\right)^T\right]$$

$$P_{wy} = E\left[\left(S_w \Delta z\right)\left(\tilde{D}_{\Delta z}\tilde{f}\right)^T\right]$$

Sustituyendo (3.60):

$$P_{wy} = E\left[\sum_{p=1}^{n} s_{w,p}\Delta z_p\left(\sum_{p=1}^{n}\Delta z_p \mu_p \delta_p \tilde{f}(z)\right)^T\right]\frac{1}{h}$$

Denótese el i-ésimo momento de Δz como σ_i

$$P_{wy} = \sigma_2 \left[\sum_{p=1}^{n} s_{w,p} \left(\mu_p \delta_p \tilde{f}(z) \right)^T \right] \frac{1}{h}$$

Sustituyendo (3.64) en la ecuación anterior, se tiene:

$$P_{wy} = \frac{1}{2h} \sum_{p=1}^{n} s_{w,p} \left(\tilde{f}\left(\bar{z} + he_p\right) - \tilde{f}\left(\bar{z} - he_p\right) \right)^T \tag{3.74}$$

Retomando (3.64) donde $s_{w,p}$ es la p-ésima columna de S_w, (3.74) se puede rescribir como:

$$P_{wy} = \frac{1}{2h} \sum_{p=1}^{n} s_{w,p} \left(\tilde{f}\left(\bar{w} + hs_{w,p}\right) - \tilde{f}\left(\bar{w} - hs_{w,p}\right) \right)^T \tag{3.75}$$

Para la aproximación de segundo orden:

$$y \approx \tilde{f}(\bar{z}) + \tilde{D}_{\Delta z}\tilde{f} + \frac{1}{2}\tilde{D}_{\Delta z}^2 \tilde{f} \tag{3.76}$$

Se tiene:

$$P_y = \frac{1}{4h^2} \sum_{p=1}^{n} \left[f(\bar{w} + hs_{w,p}) - f(\bar{w} - hs_{w,p}) \right] \times \left[f(\bar{w} + hs_{w,p}) - f(\bar{w} - hs_{w,p}) \right]^T$$
$$+ \frac{h^2 - 1}{4h^2} \sum_{p=1}^{n} \left[f\left(\bar{w} + hs_{w,p}\right) + f\left(\bar{w} - hs_{w,p}\right) - 2f\left(\bar{w}\right) \right] \times \left[f\left(\bar{w} + hs_{w,p}\right) + f\left(\bar{w} - hs_{w,p}\right) - 2f\left(\bar{w}\right) \right]^T \tag{3.77}$$

$$P_{wy} = \frac{1}{2h} \sum_{p=1}^{n} s_{w,p} \left(f\left(\bar{w} + hs_{w,p}\right) - f\left(\bar{w} - hs_{w,p}\right) \right)^T \tag{3.78}$$

Considérese un sistema no lineal cuya salida está definida por:

$$y_k = f\left(w_k, u_k\right) \tag{3.79}$$

con f no lineal, v_k y u_k se suponen ruidos, Gaussianos, aditivos, independientes y $v_k \sim \left(\bar{v}_k, Q(k)\right)$ $u_k \sim \left(\bar{u}_k, R(k)\right)$; además w es el vector de estados a estimar, cuya dimensión es L.

A continuación se definirán las siguientes expresiones:

$$P = S_w S_w^T$$
$$Q = S_v S_v^T \tag{3.80}$$
$$R = S_u S_u^T$$

Además:

$$S_{w\hat{w}}^{(1)}(k+1) \triangleq S_w(k)$$
$$S_{wv}^{(1)} \triangleq Q_0^{1/2}$$
$$S_{vu}^{1} \triangleq R_0^{1/2} \tag{3.81}$$
$$\left(S_{yw}^{(1)}\right)_{ij} \triangleq \frac{1}{2h}\left(f_j\left(w+hs_{w,i}\right)-f_j\left(w-hs_{w,i}\right)\right)$$
$$\left(S_{yw}^{(2)}\right)_{ij} \triangleq \frac{\left(h^2-1\right)^{1/2}}{2h^2}\left(f_j\left(w+hs_{w,i}\right)+f_j\left(w-hs_{w,i}\right)-2f_j(w)\right)$$

Utilizando (3.77) y (3.81)

$$P_y = S_y S_y^T \tag{3.82}$$

con

$$S_y = \left[S_{yw}^{(1)} \quad S_{yu}^{(1)} \quad S_{yw}^{(2)}\right] \tag{3.83}$$

y utilizando (3.78) y (3.81)

$$P_{wy} = S_w S_{yw}^{(1)\,T} \tag{3.84}$$

Entonces, de acuerdo con [20], la materiz P_{wy} es equivalente a la matriz PH^T del *FKE* y la matriz P_y es equivalente a la matriz $\left[HPH^T + R\right]$ del *FKE*, entonces:

$$K_k(k) = P_{wy}(k)P_y^{-1}(k)$$

Por lo tanto las ecuaciones del *FKELD* quedan como:

$$K_k(k) = P_{wy}(k)P_y^{-1}(k) \tag{3.85}$$
$$w(k+1) = w(k) + K(k)\left[y(k) - \bar{y}(k)\right] \tag{3.86}$$
$$P(k+1) = P(k) - K(k)P_y(k)K^T(k) + Q \tag{3.87}$$

con:

$$\bar{y}_j = \frac{h^2-L}{h^2}f_j(w) + \frac{1}{2h^2}\sum_{i=1}^{L}\left[f_j\left(w+hs_{w,i}\right)+f_j\left(w-hs_{w,i}\right)\right] \tag{3.88}$$

donde y es la medición de la salida del sistema real, $\bar{y}$ es el estimado de la salida. El vector de estados está denotado por $w \in \Re^L$; $\bar{S}_w \in \Re^{LxL}$ está definida por una descomposición de Cholesky de P como se muestra en (3.80): P, Q y R están definidas igual que en el *FKE*. La i-ésima columna de $\bar{S}_w$ es denotada por $s_{x,i}$. Para cada i-ésima columna correspondiente se forman dos versiones del vector w: $w + hs_{x,i}$ y $w - hs_{x,i}$. Para cada una de éstas, se debe calcular la salida correspondiente. La j-ésima salida del sistema estimado para los estados nominales es definida como: $f_j(w)$; así mismo para cada una de las variantes de w corresponden $f_j(w + hs_{x,i})$ y $f_j(w - hs_{x,i})$ respectivamente. Las matrices $S_{wv}^{(1)} \in \Re^{LxL}$ y la matriz $S_{yu}^{(1)} \in \Re^{mxm}$ son matrices diagonales cuyos elementos no-cero son iguales a $Q_0^{1/2}$ y $R_0^{1/2}$ respectivamente. Los elementos de la matriz $S_{yw}^{(1)} \in \Re^{mxL}$ están definidos en (3.81). La matriz de ganancia del *FKELD* está dada por (3.85); La actualización de los estados se define con (3.86); y finalmente la actualización de la matriz de covarianza del error se define en (3.87) [7].

3.6 El Filtro de Kalman Extendido con Factor de Olvido (*FKEFO*).

En el algoritmo presentado en la sección 3.3, toda la información (Mediciones nuevas y pasadas) es tratada con la misma importancia. Esto puede tener ciertas complicaciones; Si la planta cambia sus parámetros rápidamente, la información pasada puede retardar la convergencia del vector de estados al nuevo valor. Así que el algoritmo es modificado con el objeto de que la información reciente tenga mayor peso mientras que la información pasada sea olvidada. Esto se puede lograr al incluir un *factor de olvido*. Las ecuaciones del *FKE* con Factor de olvido (*FKEFO*) están dadas por:

$$K_k = P_k^- H_k^T \left[H_k P_k^- H_k^T + \lambda_k R_k \right]^{-1} \tag{3.89}$$

$$\hat{w}_k = \hat{w}_k^- + K_k \left[y_k - \hat{y}_k \right] \tag{3.90}$$

$$P_k = \lambda_k^{-1} P_{k-1} - \lambda_k^{-1} K_{k-1} H_{k-1} P_{k-1} + Q_{k-1} \tag{3.91}$$

donde λ_k es el factor de olvido y todas las otras expresiones están definidas igual que en el *FKE*. Aquí λ_k es una constante positiva, cercana pero menor a uno, cuando se toma igual a uno, se tiene el *FKE* ordinario [12].

Los factores de olvido comúnmente son utilizados cuando los sistemas tienen dinámicas que varían rápidamente, en algoritmos de identificación recursiva y control adaptable, en este aspecto, la inclusión del factor de olvido permite una convergencia más rápida [21], [12].

Capítulo 4

APROXIMACIÓN DE SERIES DE TIEMPO.

La literatura es muy amplia sobre las capacidades de las redes neuronales, tipo perceptrón multicapa (MLP), para reproducir relaciones no lineales entre conjuntos de datos. Las redes neuronales recurrentes poseen un rico repertorio de arquitecturas, lo cual las habilita para realizar diversas aplicaciones [10], que no son posibles con las redes neuronales estáticas; algunas de estas aplicaciones son: predicción no lineal, modelado, control, representaciones en espacio de estado, etc. En este capítulo se presenta la primer aportación importante de esta tesis: La aproximación de series de tiempo usando redes neuronales recurrentes tipo perceptrón multicapa.

4.1 El *MLP* entrenado con el *FKE*.

Diversos investigadores han propuesto algoritmos de entrenamiento para redes neuronales basados en el Filtro de Kalman Extendido (*FKE*). Dicha aplicación se basa en que el entrenamiento del *MLP* puede ser interpretado como un problema de estimación para un sistema no lineal, el cual pueda ser resuelto con el *FKE* [7], [10], [11], [29], [42].

El uso de un algoritmo basado en el *FKE* se basa, en que el algoritmo de gradiente descendiente, el de mínimos cuadrados recursivo, el de retropropagación y otros, solo son casos específicos del Filtro de Kalman, [12], [32], [42]. En la aplicación del Filtro de Kalman al entrenamiento de redes neuronales, los pesos del perceptrón multicapa son los estados que el Filtro de Kalman estima, y la salida de la red es la medición usada por el Filtro de Kalman. Como el *MLP* es un sistema no lineal, es necesario utilizar el *FKE*.

Es por ello que surge la alternativa de visualizar el entrenamiento de redes neuronales como un problema de filtrado óptimo, la solución del cual sea recursiva, utilice la información actual y no necesite almacenar toda la evolución de los pesos; ésta es la esencia del Filtro de Kalman aplicado al entrenamiento de redes neuronales. La estimación se realiza recursivamente; esto es, cada actualización del estimado (pesos sinápticos) es calculada a partir del estimado anterior y de los datos actuales. Así sólo el estimado anterior requiere ser almacenado. El entrenamiento se realiza a partir de un conjunto de N mediciones entrada-salida. El objetivo es encontrar los pesos óptimos que minimicen el error de predicción.

Acorde a lo establecido en la sección 3.3.2, se tiene que las ecuaciones necesarias para el desarrollo del algoritmo basado en el *FKE* son:

$$\begin{aligned} K(k) &= P(k)H^T(k)\left[R(k)+H(k)P(k)H^T(k)\right]^{-1} \\ w(k+1) &= w(k)+K(k)\left[y(k)-\hat{y}(k)\right] \\ P(k+1) &= P(k)-K(k)H(k)P(k)+Q(k) \end{aligned} \tag{4.1}$$

donde $P(k)\in\Re^{LxL}$ y $P(k+1)\in\Re^{LxL}$ representan la matriz de covarianza del error de predicción al tiempo k y $k+1$ respectivamente; $w\in\Re^{L}$ es el vector de pesos (estados); L es el número total de pesos de la red neuronal; $y\in\Re^{m}$ es el vector de salidas, donde m es el número total de salidas; $\hat{y}\in\Re^{m}$ es la salida de la red neuronal; $K\in\Re^{Lxm}$ es la matriz de ganancia de Kalman; $Q\in\Re^{LxL}$ es la matriz de covarianza del ruido de proceso; y $R\in\Re^{mxm}$ es la matriz de covarianza del ruido de medición. $H\in\Re^{m\times L}$ es la matriz que contiene las derivadas de cada salida de la red neuronal $(\hat{y}_i)$ con respecto a cada uno de los pesos (w_j), tal y como se define en (4.2)

$$H_{ij}(k)=\left[\frac{\partial\hat{y}_i(k)}{\partial w_j(k)}\right]_{w(k)=\hat{w}(K+1)},\quad i=1\cdots m,\ j=1\cdots L \tag{4.2}$$

Considérese el *MLP* mostrado en la figura 4.1, con p entradas y *noc* neuronas ocultas el vector de pesos está dado por:

$$w=\left[w_{10}^{(1)}\quad w_{11}^{(1)}\quad\cdots\quad w_{1p}^{(1)}\quad w_{20}^{(1)}\quad w_{21}^{(1)}\quad\cdots\quad w_{nocp}^{(1)}\quad w_{10}^{(2)}\quad w_{11}^{(2)}\cdots\quad w_{1noc}^{(2)}\right]^T \tag{4.3}$$

L es el número total de pesos y se define como:

$$L=(p+1)\,\mathrm{x}\,noc+(noc+1)\,\mathrm{x}\,m \tag{4.4}$$

donde m es el número total de salidas de la red neuronal.

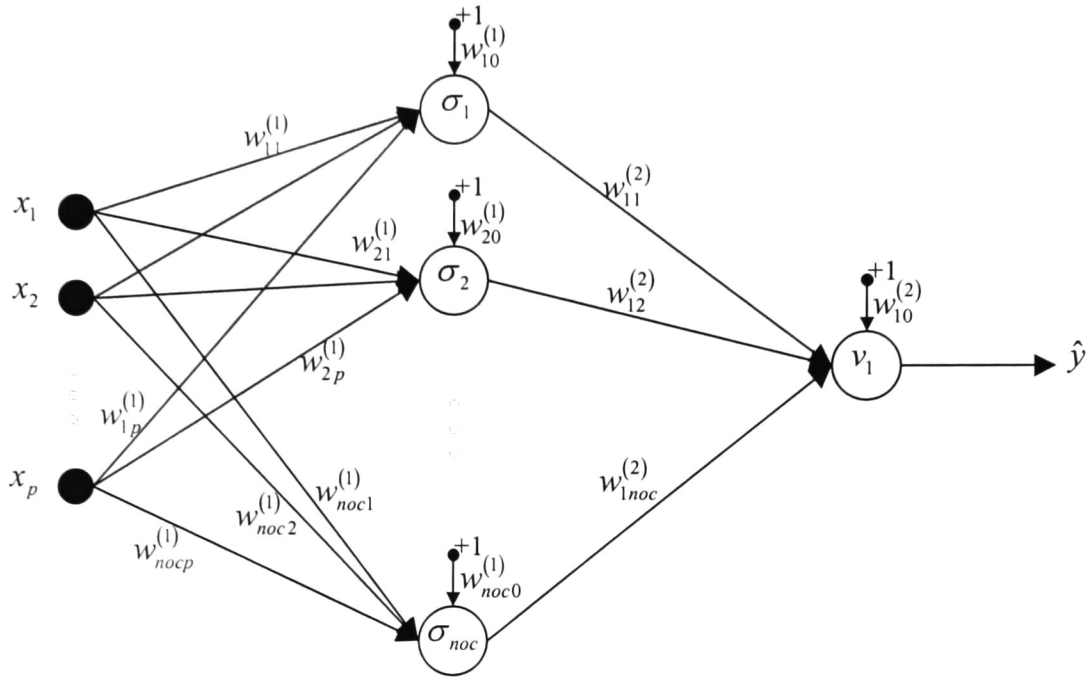


Figura 4.1.- Esquema del *MLP*.

La salida del *MLP* mostrado en la figura 4.1 esta definida por (4.5).

$$\begin{aligned} \sigma_i &= \frac{1}{1+e^{-n_i}} \quad i=1\cdots noc \\ n_i &= \sum_{j=0}^{p} w_{ij}^{(1)} x_j \quad x_0 = +1 \\ v_1 &= \sum_{k=0}^{noc} w_{1k}^{(2)} \sigma_k \quad \sigma_0 = +1 \\ \hat{y} &= v_1 \end{aligned} \tag{4.5}$$

Por lo tanto para el *MLP* mostrado en la figura 4.1, la ecuación (4.2) se puede expresar como:

$$\frac{\partial \hat{y}}{\partial w} = \left[\frac{\partial \hat{y}}{\partial w_{10}^{(1)}} \quad \frac{\partial \hat{y}}{\partial w_{11}^{(1)}} \quad \cdots \quad \frac{\partial \hat{y}}{\partial w_{1noc}^{(2)}} \right] \tag{4.6}$$

donde:

$$\begin{aligned}
&\frac{\partial \hat{y}}{\partial w_{10}^{(1)}} = \frac{w_{11}^{(2)} e^{-n_1}}{\left(1-e^{-n_1}\right)^2} x_0; \quad \frac{\partial \hat{y}}{\partial w_{11}^{(1)}} = \frac{w_{11}^{(2)} e^{-n_1}}{\left(1-e^{-n_1}\right)^2} x_1; \quad \cdots \quad \frac{\partial \hat{y}}{\partial w_{1p}^{(1)}} = \frac{w_{11}^{(2)} e^{-n_1}}{\left(1-e^{-n_1}\right)^2} x_p; \\
&\frac{\partial \hat{y}}{\partial w_{20}^{(1)}} = \frac{w_{12}^{(2)} e^{-n_2}}{\left(1-e^{-n_2}\right)^2} x_0; \quad \frac{\partial \hat{y}}{\partial w_{21}^{(1)}} = \frac{w_{11}^{(2)} e^{-n_2}}{\left(1-e^{-n_2}\right)^2} x_1; \quad \cdots \quad \frac{\partial \hat{y}}{\partial w_{2p}^{(1)}} = \frac{w_{11}^{(2)} e^{-n_2}}{\left(1-e^{-n_2}\right)^2} x_p; \\
&\vdots \\
&\frac{\partial \hat{y}}{\partial w_{noc0}^{(1)}} = \frac{w_{11}^{(2)} e^{-n_{noc}}}{\left(1-e^{-n_{noc}}\right)^2} x_0; \quad \frac{\partial \hat{y}}{\partial w_{noc1}^{(1)}} = \frac{w_{11}^{(2)} e^{-n_{noc}}}{\left(1-e^{-n_{noc}}\right)^2} x_1; \quad \cdots \quad \frac{\partial \hat{y}}{\partial w_{nocp}^{(1)}} = \frac{w_{11}^{(2)} e^{-n_{noc}}}{\left(1-e^{-n_{noc}}\right)^2} x_p; \\
&\frac{\partial \hat{y}}{\partial w_{10}^{(2)}} = 1; \quad \frac{\partial \hat{y}}{\partial w_{11}^{(2)}} = \frac{1}{1+e^{-n_1}} \quad \frac{\partial \hat{y}}{\partial w_{12}^{(2)}} = \frac{1}{1+e^{-n_2}} \quad \cdots \quad \frac{\partial \hat{y}}{\partial w_{1noc}^{(2)}} = \frac{1}{1+e^{-n_{noc}}}
\end{aligned} \tag{4.7}$$

con x_0 y n_i como en (4.5). Definiendo la expresión (4.8) como:

$$\gamma(n_i) = \frac{w_{1i}^{(2)} e^{-n_i}}{\left(1+e^{-n_i}\right)^2} \quad i = 1 \cdots noc \tag{4.8}$$

Finalmente, considerando las expresiones (4.5) y (4.7), H puede expresarse como:

$$H = \left[\gamma(n_1)x_0 \quad \cdots \quad \gamma(n_1)x_p \quad \gamma(n_2)x_0 \quad \cdots \quad \gamma(n_{noc})x_p \quad \sigma_0 \quad \sigma_1 \quad \cdots \quad \sigma_{noc}\right] \tag{4.9}$$

La expresión (4.9) es valida para el *MLP* mostrado en la figura 4.1 con funciones de activación sigmoidal para las neuronas de la capa oculta y lineal para la neurona de salida.

4.2 El *MLP* entrenado con el *FKELD*.

Uno de los principales inconvenientes del *FKE* es la necesidad del cálculo de derivadas; para tratar de subsanar esto en [7] y [24], proponen un método alternativo, el *FKELD*, el cual se presentó en la sección 3.5 y ahora se plantea para el entrenamiento de redes neuronales.

Definanse las siguientes expresiones:

$$P = S_w S_w^T$$
$$Q = S_v S_v^T$$
$$R = S_u S_u^T$$

$$S_{ww}^{(1)}(k+1) \triangleq S_w(k) \tag{4.10}$$
$$S_{wv}^{(1)} \triangleq Q_0^{1/2}$$
$$S_{yu}^{1} \triangleq R_0^{1/2}$$
$$\left(S_{yw}^{(1)}\right)_{ij} \triangleq \frac{1}{2h}\left(f_j\left(w + hs_{w,j}\right) - f_j\left(w - hs_{w,j}\right)\right)$$
$$\left(S_{yw}^{(2)}\right)_{ij} \triangleq \frac{\left(h^2-1\right)^{1/2}}{2h^2}\left(f_j\left(w + hs_{w,j}\right) + f_j\left(w - hs_{w,j}\right) - 2f_j(w)\right)$$

Además considérese lo siguiente:

$$P_y = S_y S_y^T$$
$$S_y = \left[S_{yw}^{(1)} \quad S_{yu}^{(1)} \quad S_{yw}^{(2)}\right] \tag{4.11}$$
$$P_{wy} = S_w S_{yw}^{(1)\,T}$$

Teniendo en cuenta lo establecido en la sección 3.5, las ecuaciones del *FKELD* quedan como:

$$K(k) = P_{wy}(k) P_y^{-1}(k)$$
$$w(k+1) = w(k) + K(k)\left[y(k) - \bar{y}(k)\right] \tag{4.12}$$
$$P(k+1) = P(k) - K(k) P_y(k) K^T(k) + Q(k)$$

con:

$$\bar{y}_j = \frac{h^2 - L}{h^2} f_j(w) + \frac{1}{2h^2}\sum_{i=1}^{L}\left[f_j\left(w + hs_{w,i}\right) + f_j\left(w - hs_{w,i}\right)\right] \tag{4.13}$$

donde $\overline{y}_j$ representa la j-ésima columna del estimado de la salida de la red neuronal; y es la medición de la salida del sistema real; el vector de pesos está denotado por $w \in \Re^L$; $\overline{S}_w \in \Re^{LxL}$ está definida por una descomposición de Cholesky de P como se muestra en (4.10): P, Q y R están definidas igual que en el *EKF* La i-ésima columna de $\overline{S}_w$ es denotada por $s_{x,i}$. Para cada i-ésima columna correspondiente se forman dos versiones del vector w: $w+hs_{x,i}$ y $w-hs_{x,i}$. Para cada una de estas versiones se debe calcular la salida correspondiente. La j-ésima salida de la red neuronal para los pesos está definida como: $f_j(w)$, así mismo para cada una de las variantes de w corresponden $f_j(w+hs_{x,i})$ y $f_j(w-hs_{x,i})$, respectivamente. Las matrices $S_{ww}^{(1)} \in \Re^{LxL}$ y la matriz $S_{yu}^{(1)} \in \Re^{mxm}$ son matrices diagonales cuyos elementos no-cero son iguales a $Q_0^{1/2}$ y $R_0^{1/2}$ respectivamente. Los elementos de la matriz $S_{yw}^{(1)} \in \Re^{mxL}$ están definidos en (4.10). K es la matriz de ganancia del *FKELD* [7].

4.3 Estructuras de modelos no lineales basadas en redes neuronales.

Cuando se tiene como objetivo el modelado de la dinámica de un sistema no lineal, el problema de seleccionar la estructura del modelo, se incrementa en dificultad. Es bien conocida que los *MLP* tienen la capacidad de aproximar relaciones no lineales a partir de un conjunto de datos. Así que en la búsqueda de una familia de estructuras de modelos convenientes para el modelado de sistemas dinámicos no lineales, es natural pensar en redes neuronales tipo *MLP* [25]. Haciendo esta elección, la estructura del modelo básicamente se reduce a lo siguiente:

- Seleccionar las entradas de la red
- Seleccionar la arquitectura interna de la red

Una aproximación muy común es usar las estructuras de los modelos lineales *(Apéndice 1)*, pero con una estructura interna de red neuronal tipo *MLP* Esto tiene algunas ventajas como:

- Es una extensión natural de modelos lineales bien conocidos.
- La arquitectura interna puede expandirse gradualmente con una gran flexibilidad, que es necesaria para modelar relaciones no lineales complejas.
- Las decisiones estructurales requeridas por el usuario son reducidas a un nivel razonable de manejar.

Las estructuras de modelos no lineales, en contraparte a las lineales, que se presentan en el *Apéndice 1*, son denotadas por:

$$y(t) = g\left[\varphi(t,\theta),\theta\right] + e(t)$$

o en forma de predictor:

$$\hat{y}(t|\theta) = g[\varphi(t,\theta),\theta]$$

$\varphi(t,\theta)$ es el vector de regresión, mientras que θ es el vector que contiene los parámetros ajustables en la red neuronal (pesos), g es la función realizada por la red neuronal. Dependiendo del tipo de vector de regresión que se seleccione, surgen diferentes estructuras no lineales. Si el vector de regresión es seleccionado como para un modelo *ARX*, la estructura del modelo es llamada *NNARX* como el acrónimo para *"Neural Network ARX"*. De igual modo para *NNFIR, NNARMAX y NNOE.*

NNFIR y NNARX .

Como en el caso lineal, estos predictores son siempre estables, debido a que son puramente una relación algebraica entre la predicción, entradas pasadas y salidas pasadas. Esto es particularmente importante en el caso no lineal, debido a que el estudio de la estabilidad es mucho más complejo que para sistemas lineales. En la figura 4.2 se muestra el diagrama a bloques correspondiente a estas estructuras. La ausencia de problemas relacionados con la estabilidad, hace que estas estructuras, en particular la estructura *NNARX*, sea la preferida cuando el sistema es determinístico o el nivel de ruido es insignificante [25].

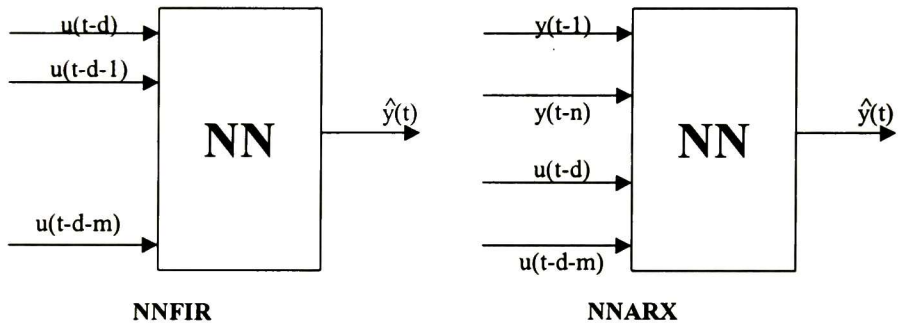
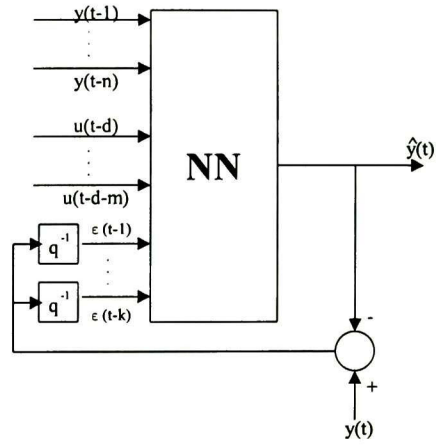


Figura 4.2 Diagramas a bloques de las estructuras *NNFIR* y *NNARX*

NNARMAX.

De acuerdo a la figura 4.3 el vector de regresión de la estructura *NNARMAX* esta compuesto por salidas y entradas pasadas así como de los errores de predicción pasados, estos últimos dependen de la salida del modelo y consecuentemente establecen una retroalimentación. Lo cual puede provocar problemas de estabilidad. Típicamente es relevante considerar la estabilidad como una propiedad local. Esto significa que un modelo *NNARMAX* es estable cuando opera en un régimen por calcular, mientras que puede ser inestable en otros [25].

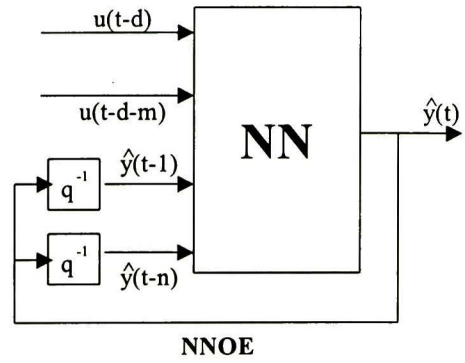


NNARMAX

Figura 4.3.- Diagrama a bloques de la estructura *NNARMAX*.

NNOE.

Algunos de los regresores en la estructura *NNOE* son predicciones de salidas pasadas y por lo tanto esta sujeto a los mismos problemas que en la estructura *NNARMAX* [25].



NNOE

Figura 4.4.- Diagrama a bloques de la estructura *NNOE*.

Existen muchas variantes y/o estructuras híbridas, originadas de la combinación de estructuras lineales y no lineales pero su análisis sale del propósito de este trabajo.

4.4 Predicción de la serie de tiempo de la demanda eléctrica.

4.4.1 Descripción de la serie de tiempo de la demanda eléctrica.

En esta sección se describen algunas características de la serie de tiempo de la demanda eléctrica, usando como caso práctico datos del sistema eléctrico del estado de California, EEUU. Los datos fueron obtenidos de la pagina de Internet de CALIFORNIA ISO *(Independent System Operator)* [14] y comprenden el periodo del 1 de septiembre de 2000 al 14 de octubre de 2002. Esta base de datos cuenta con 18,591 valores de mediciones horarias que corresponden a 24 meses y 14 días. Comenzaremos por definir serie de tiempo.

Se llama serie de tiempo a un conjunto de mediciones de cierto fenómeno o experimento registradas secuencialmente en el tiempo [27].

En la figura 4.5 se muestra la serie de tiempo de la demanda durante el periodo mencionado. En la figura 4.6, se muestra el comportamiento diario y anual de la demanda eléctrica del estado de California. En la gráfica de la figura 4.5 es posible observar ciertos efectos estaciónales de los cuales se desprende que los periodos de mayor consumo de energía eléctrica son:

- Hora 1 a 900 que corresponde a un periodo de septiembre-octubre (verano 2000)
- Hora 6000 a 9500 que corresponde al periodo de mayo-octubre (primavera-verano 2001)
- Hora 15000 a 18951 que corresponde al periodo mayo octubre (primavera – verano 2002)

La razón del alto consumo de energía eléctrica durante el periodo primavera-verano está asociado con el uso de aparatos de refrigeración y sistemas de bombeo para el regadío [38]. Otras características de la curva de la demanda son los ciclos o ritmos semanales y diarios. Estos ciclos son debidos al ritmo normal de las actividades económicas y sociales [39]. Muchas actividades económicas ocurren de lunes a viernes, mientras que los sábados y domingos se interrumpen porque son días no laborales. Este comportamiento se puede observar en la figura 4.7, donde el primer día es un domingo.

El ritmo diario, resulta del comportamiento de las personas durante el día, la mayoría de la gente duerme de noche y por consecuencia, el consumo de energía disminuye. También durante el día se desarrollan muchas actividades de manera simultanea. Este efecto puede observarse en la figura 4.8.

Según [27], otra característica importante en el análisis de series de tiempo es la presencia de movimientos irregulares, los cuales pueden ser aleatorios o bien se deben a fuerzas naturales esporádicas. Tales fluctuaciones son completamente impredecibles aunque se pueden identificar o reconocer fácilmente. En la serie de tiempo de la demanda que se esta analizando en este trabajo, no se identifica ningún efecto irregular.

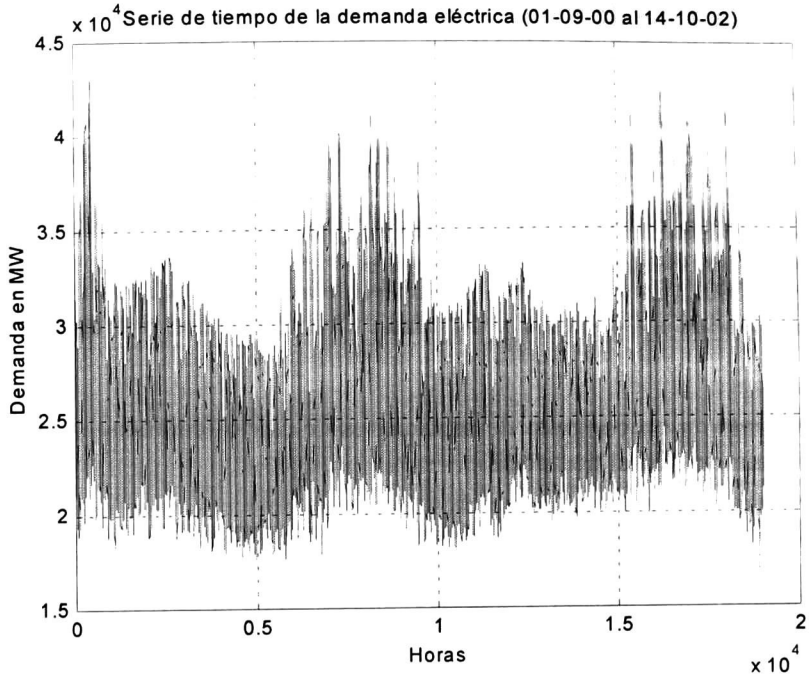


Figura 4.5.- Serie de tiempo de la demanda eléctrica del estado de California.

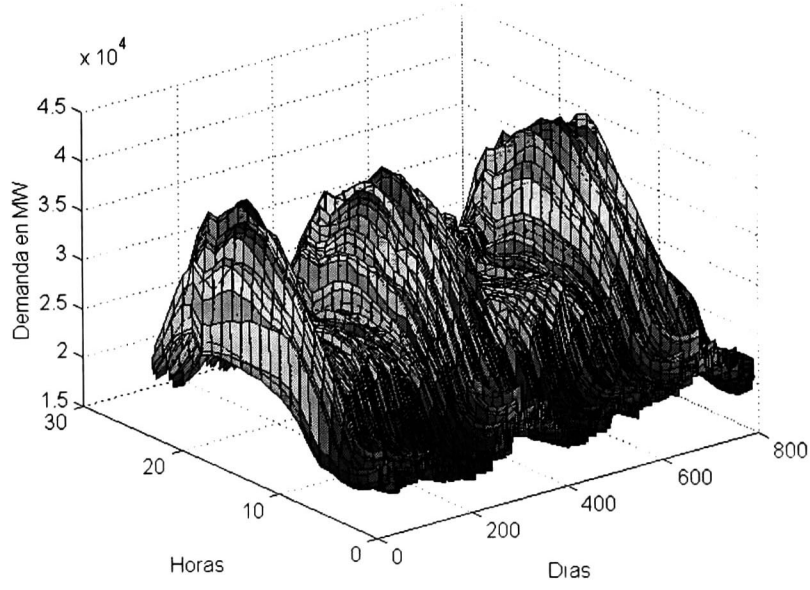


Figura 4.6.- Comportamiento diario de la demanda eléctrica.

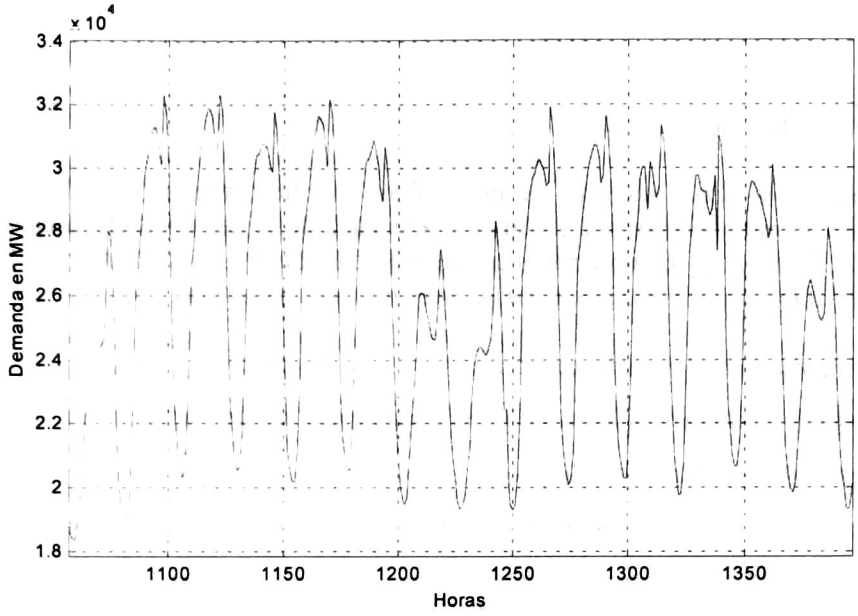


Figura 4.7.- Ritmo semanal de la demanda eléctrica.

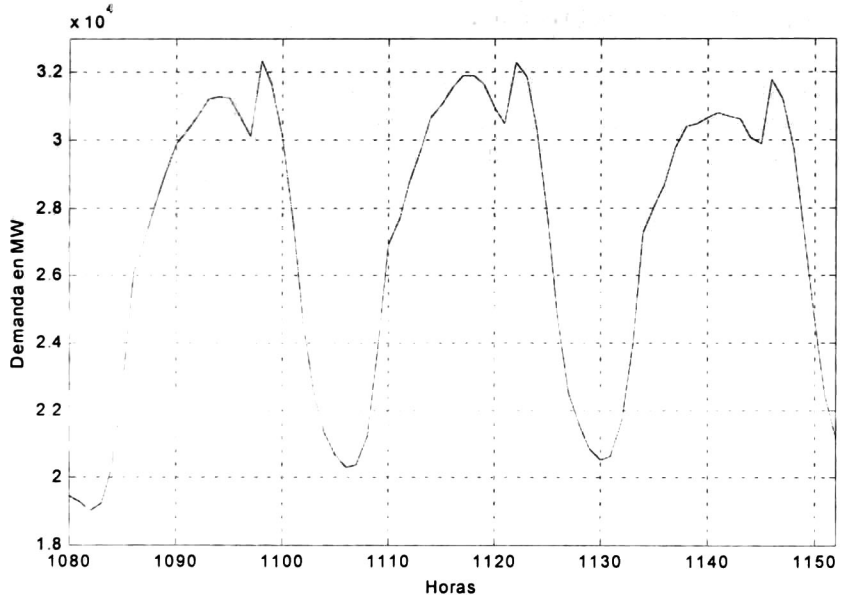


Figura 4.8.- Ritmo diario de la demanda eléctrica

Un sistema de energía eléctrica debe abastecer de energía a todos los puntos de carga con buena calidad de servicio, lo cual implica continuidad en el servicio, eficiencia, costos mínimos, etc. Dichas características pueden garantizarse por medio de una planeación del sistema, que permita conocer no sólo su estado actual sino también las medidas que deben adoptarse para condiciones futuras.

El objetivo de la presente sección es desarrollar una herramienta, basada en redes neuronales, que permita obtener una preedición adecuada de la demanda eléctrica con una anticipación de 24 horas y con un error absoluto promedio del 5% o inferior. La predicción a corto plazo *(desde varias horas hasta varias semanas)* es usada para:

- Planeación del despacho económico
- Análisis costo beneficio de los programas de gestión de cargas
- Planes de compra de combustibles
- En algunos casos el establecimiento del precio de compra de energía eléctrica.

Es por ello que de las estructuras analizadas en la sección 4.3, se selecciono la *NNOE* o *PARALELO*, ya que dicha estructura permite dejar los pesos fijos después del entrenamiento y obtener una evolución autónoma de la red neuronal, lo que proporciona la predicción de la serie de tiempo de la demanda eléctrica. Como ya se tienen los datos de la serie de tiempo (figura 4.5), sólo queda por determinar el orden del sistema.

4.4.2 Determinación del orden del sistema.

La determinación correcta del orden del sistema es por demás importante, ya que si se selecciona mal puede traer diversos efectos perjudiciales en el entrenamiento de la red neuronal. Si el orden seleccionado es menor al real, se pueden perder características importantes de la serie de tiempo, lo cual provoca a su vez una mala predicción; el caso contrario no es tan perjudicial, sin embargo la redundancia tampoco es aconsejable, principalmente por el tiempo que se consume innecesariamente.

En [25] se sugiere encontrar el orden del sistema basándose en los Coeficientes de Lipschitz (para mayor información consúltese [25]); Norgaard proporciona una herramienta computacional que determina el orden de una serie de tiempo. Para el desarrollo del presente trabajo se utilizó dicha herramienta arrojando que para la serie de tiempo del caso de estudio el orden es 7; esto se obtiene en base a la gráfica de la figura 4.9, en la cual se puede observar que la inflexión de la misma se da en 7.

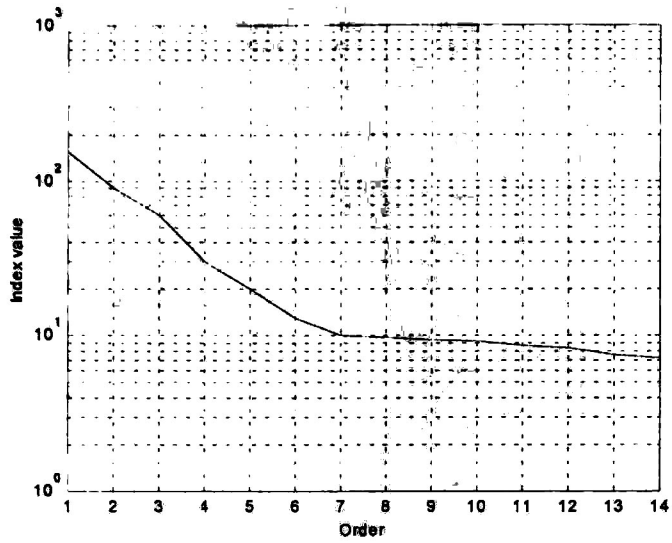


Figura 4.9.- Grafica para determinar el orden del sistema.

4.4.3 Predicción de la demanda eléctrica usando redes neuronales recurrentes entrenadas con el FKE.

La red neuronal utilizada para esta aplicación es una *RMLP* (Por su nombre en inglés "Recurrent Multilayer Perceptron") y la configuración de la misma es como se muestra en la figura 4.10. Se utilizaron 15 neuronas en la capa oculta y una neurona en la capa de salida; para las neuronas de la capa oculta se usó como función de activación una del tipo logística como la mostrada en la ecuación (2.8), con $a=1$; y para la neurona de la capa de salida una función tipo lineal. Los valores iniciales para las matrices de covarianza son: $R_0 = Q_0 = P_0 = 10000$; $1e^{-4}$ es el error medio cuadrático deseado el cual se alcanzó en 35 iteraciones para el día 535 y en 43 iteraciones para el día 778. El error absoluto promedio durante las primeras 24 horas es del 2.9% para la predicción del día 535 y del 1.42% para la predicción del día 778, lo cual resulta satisfactorio ya que se desea obtener un error absoluto promedio del 5%. El vector de regresión utilizado es el que se define en (4.14).

$$\varphi = \left[\hat{y}(t-1) \quad \hat{y}(t-2) \quad \cdots \quad \hat{y}(t-7) \quad u_1(k) \quad u_2(k)\right]^T \tag{4.14}$$

En (4.14) se utilizan siete retardos de la salida estimada, ya que éste es el orden del sistema. Además fue necesario agregar dos datos externos $u_1(k)$ y $u_2(k)$, los cuales corresponden al día y la hora respectivos; esto fue necesario para obtener mejores resultados. El entrenamiento se realizó fuera de línea. Una vez finalizado el entrenamiento de la red neuronal (cuando se alcanza el error medio cuadrático deseado), los pesos se mantienen fijos y se realiza la predicción.

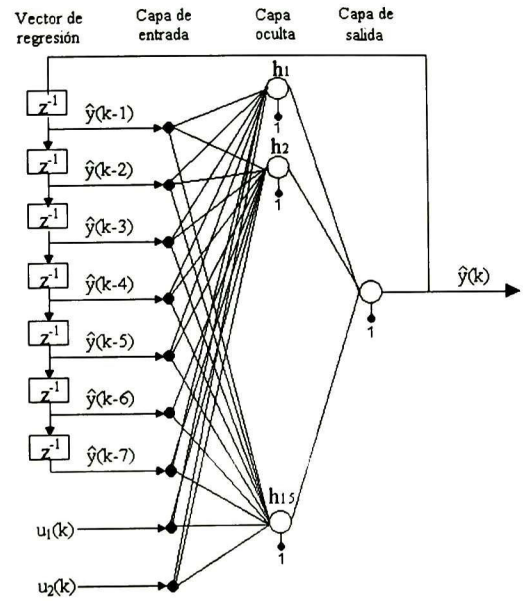


Figura 4.10.- Estructura de la red neuronal utilizada.

Los resultados obtenidos, al utilizar el algoritmo de entrenamiento basado en el *FKE*, se muestran en las figuras 4.11, 4.12 y 4.13 . En éstas, se observa la predicción para la semana que corresponde a los días 535-542 de la serie de tiempo caso de estudio de esta sección, el detalle para el día 535 es decir para las primeras 24 horas de la generalización después del entrenamiento y la evolución del error medio cuadrático, durante el entrenamiento.

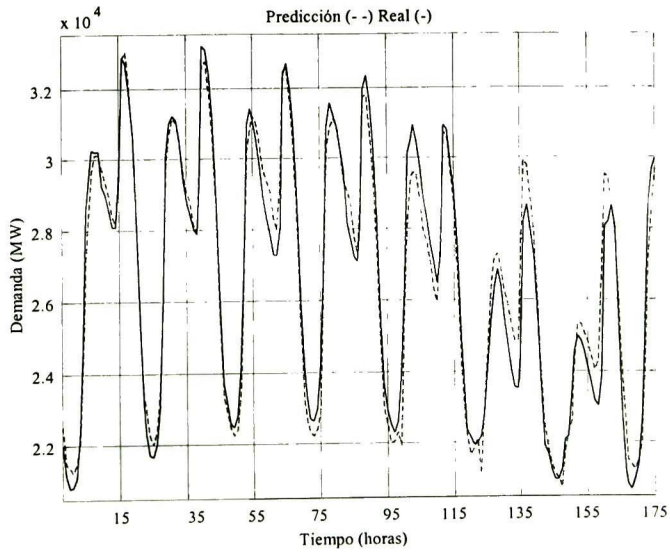


Figura 4.11.- Comparación de la demanda eléctrica real y la predicción durante una semana.

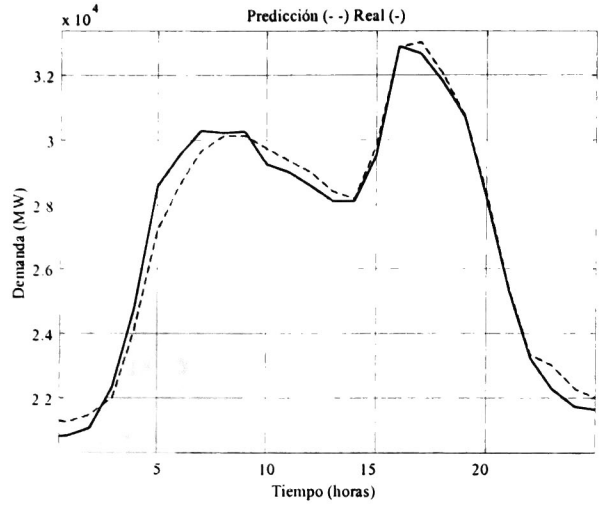


Figura 4.12.- Comparación de la demanda eléctrica real y la predicción para las primeras 24 horas.

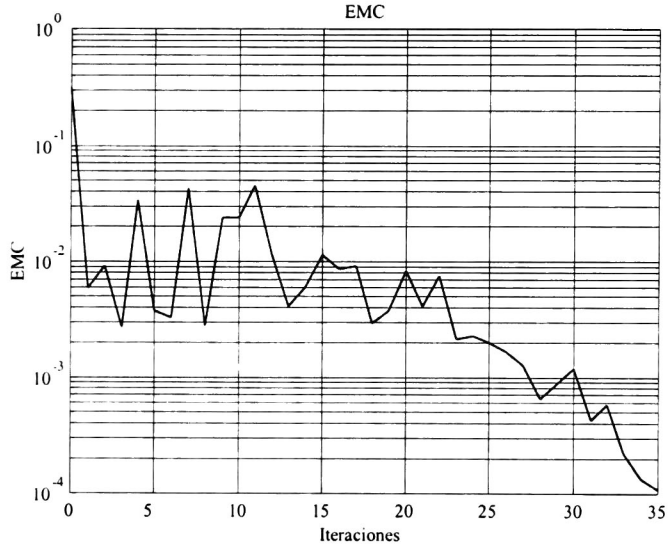


Figura 4.13.- Evolución del error medio cuadrático durante el entrenamiento de la red neuronal.

Las figuras 4.14 y 4.15 muestran la predicción para la semana que corresponde a los días 778-785 de la serie de tiempo caso de estudio de esta sección y el detalle para el día 778 es decir para las primeras 24 horas de la generalización después del entrenamiento.

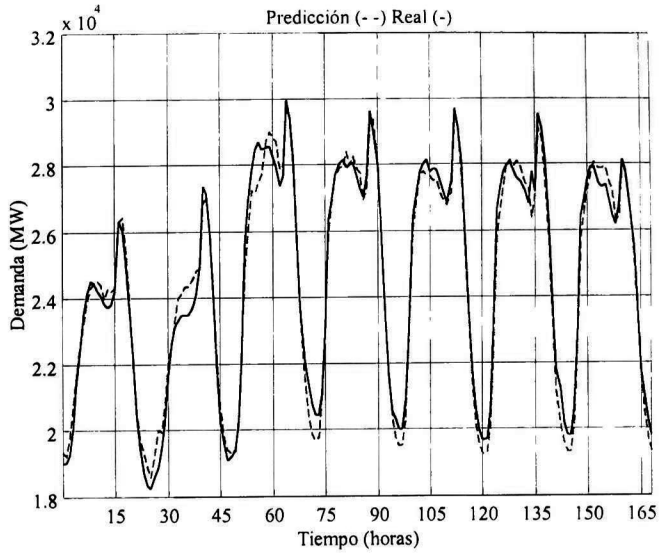


Figura 4.14.- Comparación de la demanda eléctrica real y la predicción durante una semana.

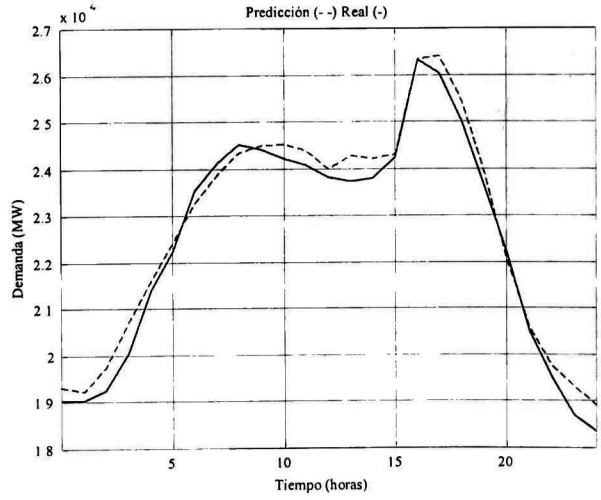


Figura 4.15.- Comparación de la demanda eléctrica real y la predicción para las primeras 24 horas.

4.4.4 Predicción de la demanda eléctrica usando redes neuronales recurrentes entrenadas con el FKELD.

En esta sección se presentan los resultados obtenidos al entrenar una red neuronal tipo *RMLP* como la mostrada en la figura 4.10. Se utilizaron 15 neuronas en la capa oculta y una neurona en la capa de salida; para las neuronas de la capa oculta se usó como función de activación una del tipo logística como la mostrada en la ecuación (2.8), con $a=1;$ y para la neurona de la capa de salida una función tipo lineal. Los valores iniciales para las matrices de covarianza son: $P_0 = 5e^{-6}$ $R_0 = 5e^{-6}$ $Q_0 = 5e^{-10}$; se utilizó el vector de regresión definido en (4.16).

Se realizó un entrenamiento de 100 iteraciones, sin embargo no fue posible alcanzar el error medio cuadrático deseado. El error absoluto promedio para las primeras 24 horas fue del 8.5% para la predicción del dia 535. Este algoritmo presenta un parámetro de diseño extra, h, que representa la longitud del intervalo donde la aproximación se satisface, h se seleccionó igual a $\sqrt{3}$, de acuerdo a lo recomendado en [7] y [24].

Los resultados obtenidos, al utilizar el algoritmo de entrenamiento basado en el *FKELD*, se muestran en las figuras 4.16, 4.17 y 4.18. En éstas, se observa la predicción de una semana, el detalle de un día y la evolución del error medio cuadrático durante el entrenamiento de la red neuronal.

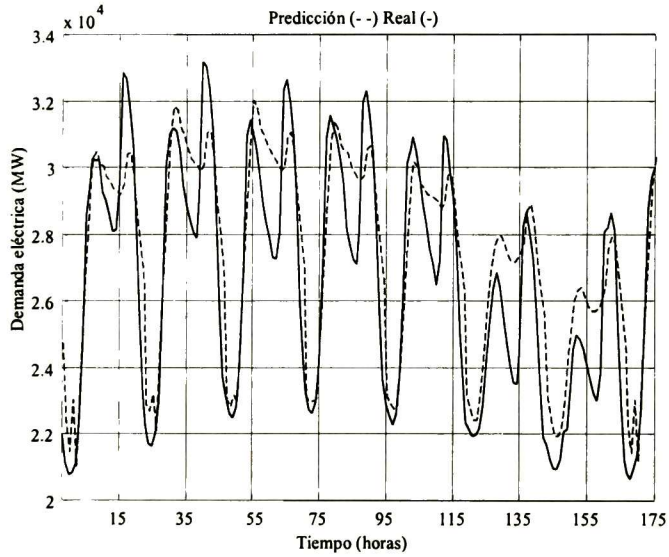


Figura 4.16.- Comparación de la demanda eléctrica real y la predicción durante una semana.

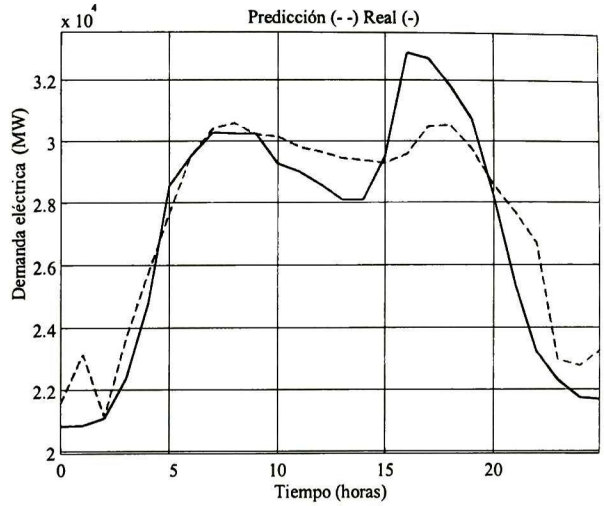


Figura 4.17.- Comparación de la demanda eléctrica real y la predicción para las primeras 24 horas.

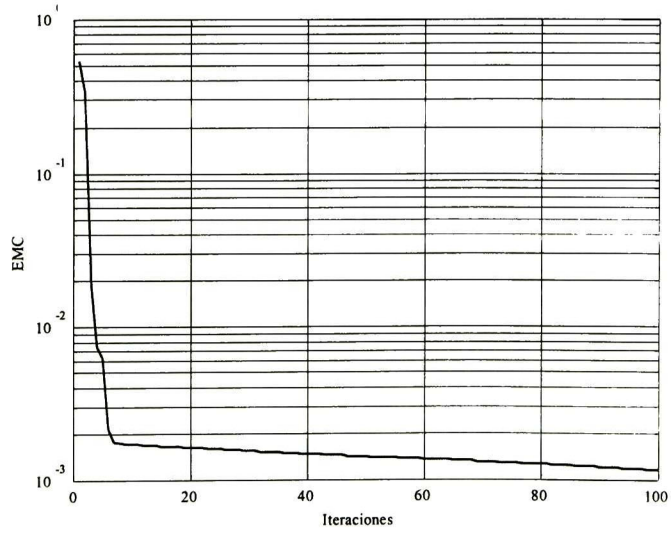


Figura 4.18.- Evolución del error medio cuadrático durante el entrenamiento de la red neuronal.

4.5 Predicción de la serie de tiempo de los precios de la energía eléctrica.

4.5.1 Descripción de la serie de tiempo de los precios de la energía eléctrica

En el ámbito de los mercados eléctricos competitivos, los productores de energía y los consumidores necesitan herramientas precisas para predecir el precio de la energía eléctrica. La predicción de precios involucra información crucial tanto para productores como para consumidores cuando ambos desean planear estrategias de mercado con el objetivo de maximizar beneficios y utilidades respectivamente.

Al igual que en el caso de la demanda eléctrica, la serie de tiempo de los precios de la energía eléctrica presenta ciclos diarios y ciclos semanales. En la figura 4.19 se presenta la serie de tiempo de los precios de la energía eléctrica del sistema Español; estos datos son del dominio público y están disponibles en la página de Internet de OMEL (*Compañía operadora del mercado eléctrico español*) [26]. En la figura 4.19 se presentan los precios por hora de la energía eléctrica para el mes de agosto de 2000 y en ella se pueden observar los ciclos ya mencionados.

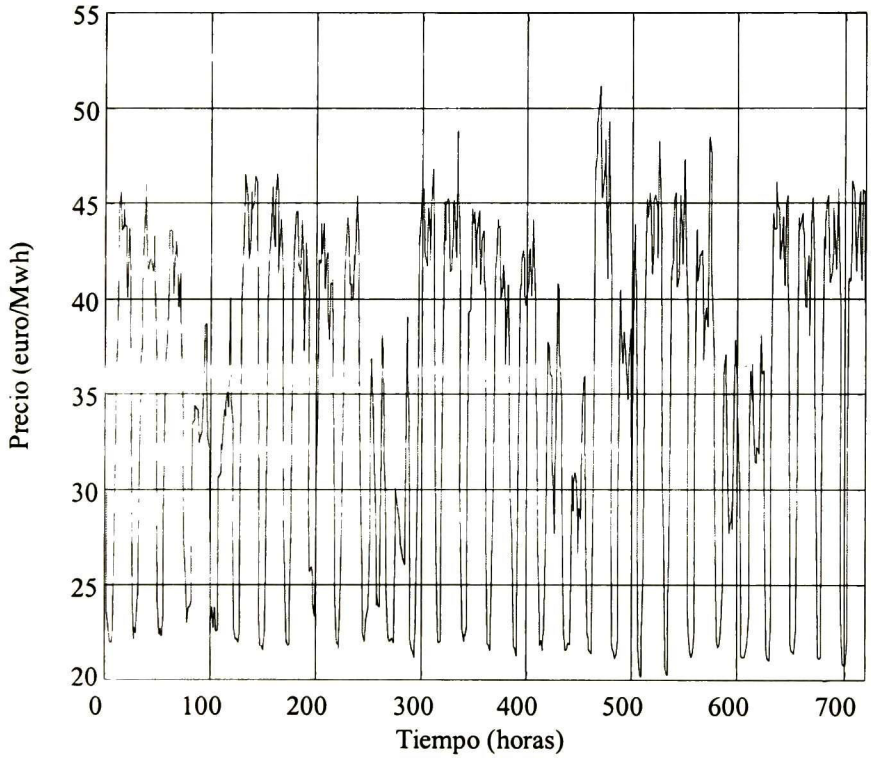


Figura 4.19.- Serie de tiempo de los precios de le energía eléctrica del sistema Español.

Si un productor de energía eléctrica tiene un buen predictor de precios para el siguiente día, puede desarrollar estrategias que maximicen su beneficio. Esto también es cierto para el consumidor, quien puede preparar sus estrategias de mercado. El objetivo de esta sección es desarrollar una herramienta, basada en redes neuronales, que permita obtener una adecuada predicción de los precios de la energía eléctrica con 24 horas de anticipación con un error absoluto promedio del 5% o inferior.

Con base en los resultados obtenidos en la sección 4.4, para este caso de estudio se seleccionó una estructura *NNOE* o *PARALELO*, así como un algoritmo de entrenamiento para la red neuronal basado en *FKE* exclusivamente.

4.5.2 Determinación del orden del sistema.

Al igual que en el caso anterior, para la determinación del orden del sistema se utiliza el método propuesto en [25]. Para la serie de tiempo mostrada en la figura 4.19, el orden del sistema es 10; esto se obtiene en base a la gráfica de la figura 4.20 en la cual se puede observar que la inflexión de la misma se da en 10.

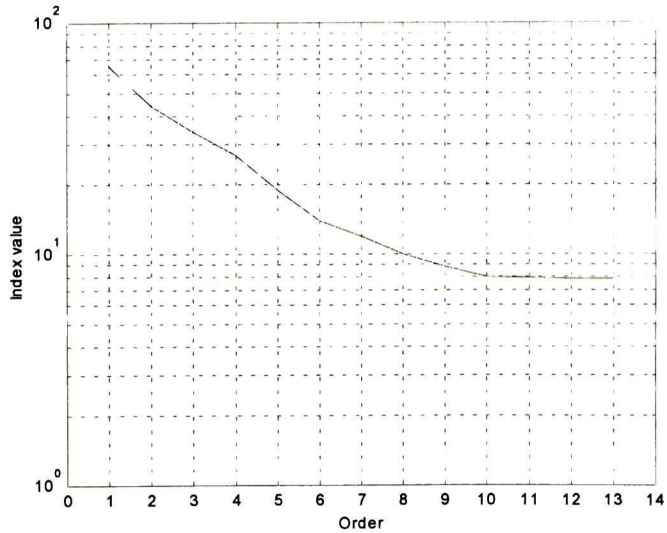


Figura 4.20.- Gráfica para determinar el orden del sistema.

4.5.3 Predicción de los precios de la energía eléctrica usando redes neuronales recurrentes entrenadas con el FKE.

La red neuronal utilizada para esta aplicación es una red tipo *RMLP* y la configuración de la misma es como se muestra en la figura 4.21. Se utilizaron 25 neuronas en la capa oculta y una neurona en la capa de salida; para las neuronas de la capa oculta se usó como función de activación una del tipo logística como la mostrada en la ecuación (2.8), con $a=1$; y para la neurona de la capa de salida una función tipo lineal. Los valores iniciales para las matrices de covarianza son: $R_0 = Q_0 = P_0 = 10000$; $1e^{-4}$ es el error medio cuadrático deseado el cual se alcanzó en 105 iteraciones para la predicción del día el 9 de agosto del 2000 y en 98 iteraciones para la predicción del día el 13 de noviembre del 2000 El error absoluto promedio durante las primeras 24 horas es del 3.1% para el 9 de agosto y del 2.8% para el 13 de noviembre, lo cual resulta satisfactorio ya que se desea obtener un error absoluto promedio del 5%. El vector de regresión utilizado es el que se define en (4.15).

$$\varphi = \left[\hat{y}(t-1) \quad \hat{y}(t-2) \quad \cdots \quad \hat{y}(t-10) \quad u_1(k) \quad u_2(k)\right]^T \tag{4.15}$$

El entrenamiento se realizó fuera de línea. Una vez finalizado el entrenamiento de la red neuronal (cuando se alcanza el error medio cuadrático deseado), los pesos se mantienen fijos y se realiza la predicción.

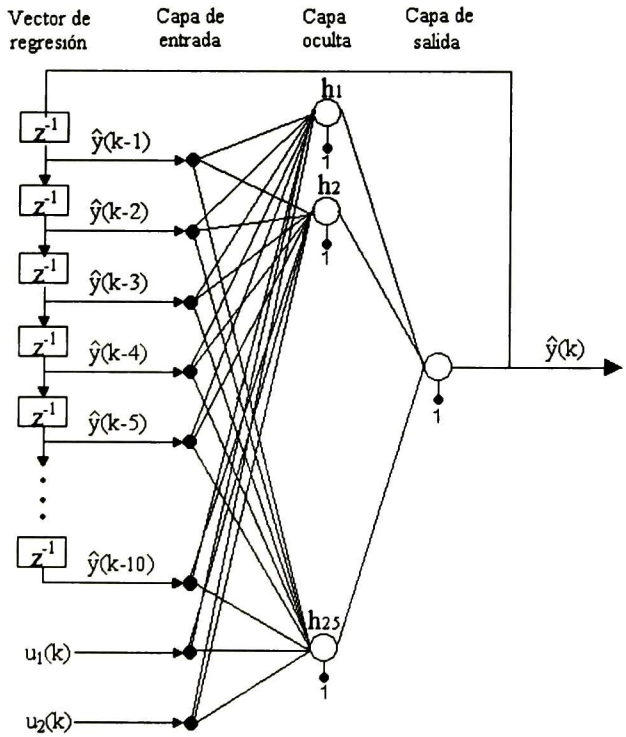


Figura 4.21.- Estructura de la red neuronal utilizada.

Los resultados obtenidos al utilizar el algoritmo de entrenamiento basado en el *FKE*, se muestran en las figuras 4.22, 4.23 y 4.24. En éstas, se observa la predicción para la semana que corresponde a los días 9-13 de agosto del 2000 de la serie de tiempo caso de estudio de esta sección, el detalle para el día 9 de agosto es decir para las primeras 24 horas de la generalización después del entrenamiento y la evolución del error medio cuadrático, durante el entrenamiento de la red.

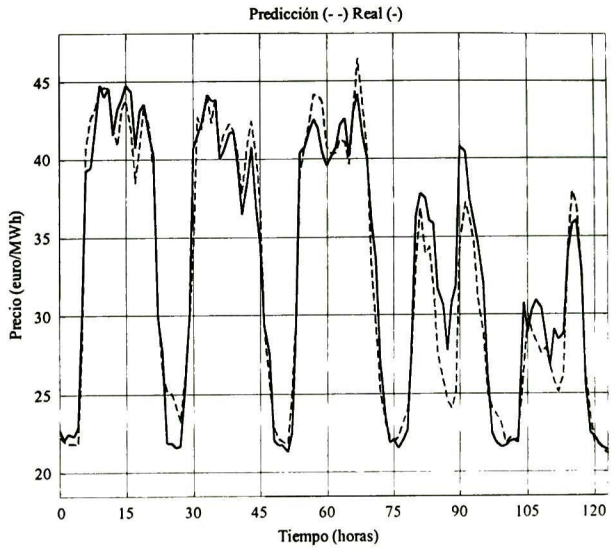


Figura 4.22 Comparación del precio real y la predicción durante una semana.

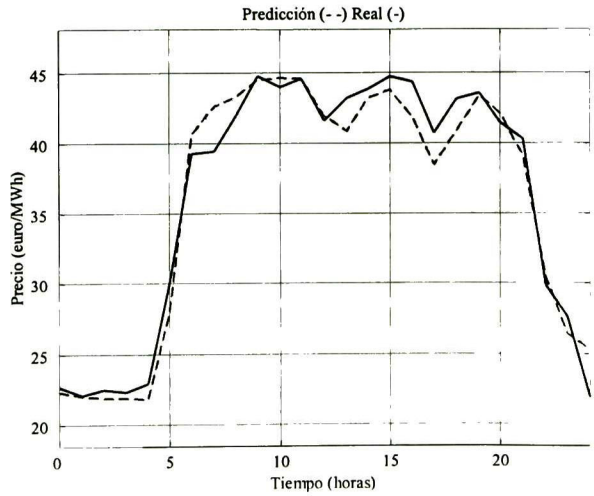


Figura 4.23.- Comparación del precio real y la predicción para las primeras 24 horas.

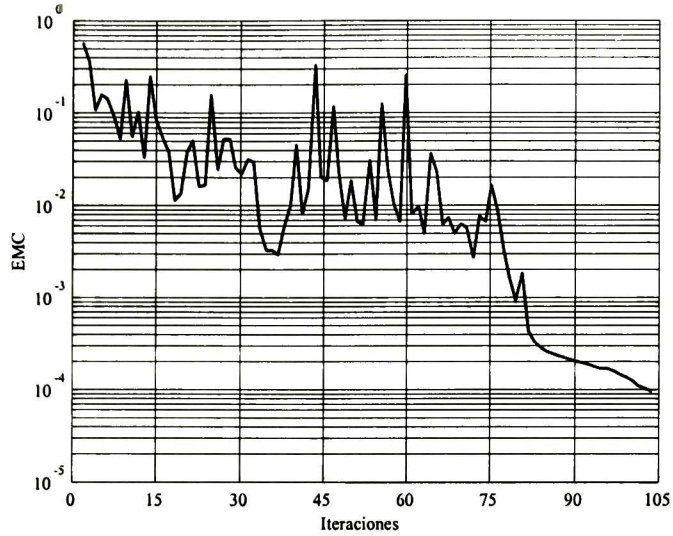


Figura 4.24.- Evolución del error medio cuadrático durante el entrenamiento de la red neuronal.

Las figuras 4.25 y 4.26 muestran la predicción para la semana que corresponde a los días 13-17 de noviembre del 2000 de la serie de tiempo caso de estudio de esta sección y el detalle para el día 13 de noviembre es decir para las primeras 24 horas de la generalización después del entrenamiento.

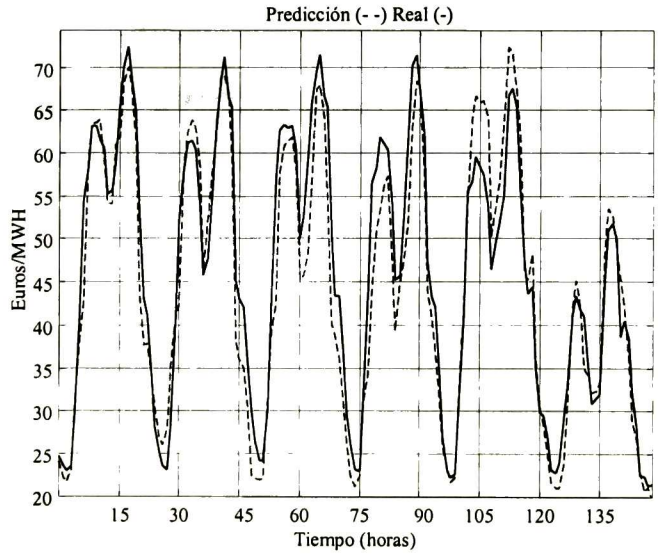


Figura 4.25.- Comparación del precio real y la predicción durante una semana.

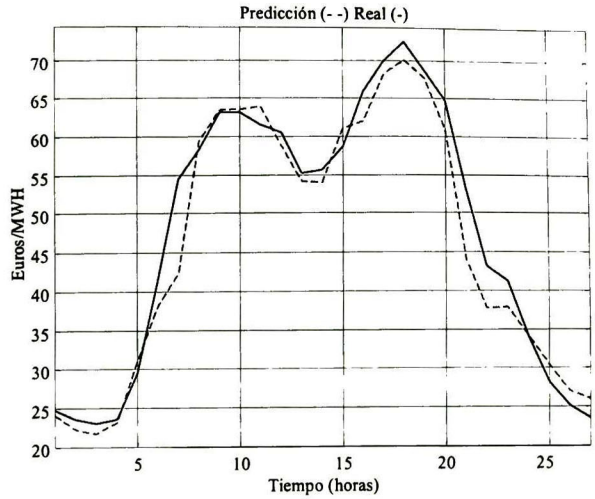


Figura 4.26.- Comparación del precio real y la predicción para las primeras 24 horas.

Capítulo 5.

LAS REDES NEURONALES RECURRENTES DE ALTO ORDEN DISCRETAS.

En el capítulo 4 se analizaron las redes neuronales recurrentes tipo RMLP, dichas redes son redes neuronales recurrentes entrada-salida esto porque solo se retroalimenta la salida de la red neuronal hacia la entrada, sin embargo la recurrencia también puede ser interna, es decir que la salida de una neurona oculta puede retroalimentarse así misma o a otras de la misma capa o de diferente capa; algunos modelos de este tipo son las redes tipo Hopfield y las redes tipo Cohen-Grossberg, entre otras. Sin embargo, el caso más general y el que analiza este capítulo son las redes neuronales de alto orden (RHONN por sus nombre en ingles "Recurrent High Order Neural Networks"). En este capítulo se presenta la segunda aportación importante de esta tesis "Las redes neuronales recurrentes de alto orden discretas"

5.1 Introducción a la *RHONN* continua.

En el caso más simple, el estado de cada neurona se puede determinar por una ecuación diferencial de la forma:

$$x_i = -a_i x_i + b_i \sum_j w_{ij} y_j \tag{5.1}$$

donde x_i es el estado de la i-ésima neurona, a_i, b_i son constantes, w_{ij} es el peso sináptico que conecta la j-ésima entrada a la i-ésima neurona y y_j puede ser una entrada externa o el estado de una neurona pasado a través de una función sigmoidal, es decir $y_j = S(x_j)$ donde $S(\bullet)$ representa una función sigmoidal [18].

El comportamiento dinámico y las propiedades de las redes neuronales de la forma presentada en la ecuación (5.1) han sido extensamente estudiados, dichos estudios han mostrado que este tipo de redes neuronales ha dado muy buenos resultados en diversas aplicaciones, sin embargo también han revelado limitaciones debido a la simplicidad de su modelo.

En una red neuronal de segundo orden, la entrada total a una neurona no es sólo la combinación lineal de los componentes y_j, también se incluyen sus productos $y_j y_k$; más aún, siguiendo esta línea es posible obtener interacciones de alto orden. Esta clase de red forma una red neuronal recurrente de alto orden (*RHONN*) [18].

Considérese una red con n neuronas y m entradas. El estado de cada neurona está gobernado por una ecuación diferencial de la siguiente forma:

$$x_i = -a_i x_i + \sum_{k=1}^{L} w_{ik} \prod_{j \in I_k} y_j^{d_j(k)} \tag{5.2}$$

donde x_i es el estado de la i-ésima neurona; L es el número de conexiones de alto orden; $\{I_1, I_2, \cdots, I_L\}$ es una colección de L subconjuntos no ordenados $\{1, 2, \cdots, m+n\}$; a_i son coeficientes reales; w_{ik} son los coeficientes reales ajustables de la red neuronal; $d_j(k)$ son enteros no negativos y y es un vector construido por las entradas a cada neurona, definido como:

$$y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \\ y_{n+1} \\ \vdots \\ y_{n+m} \end{bmatrix} = \begin{bmatrix} S(x_1) \\ \vdots \\ S(x_n) \\ S(u_1) \\ \vdots \\ S(u_m) \end{bmatrix} \tag{5.3}$$

con $u = [u_1 \quad u_2 \quad \cdots \quad u_m]^T$ como el vector de entradas a la red neuronal. La función $S(\bullet)$ es una sigmoide suave, monótonamente creciente de la forma:

$$S(x) = \frac{\mu}{1 + e^{-\beta x}} + \varepsilon \tag{5.4}$$

donde β y μ son constantes positivas y ε es un número real positivo pequeño. Obviamente $S(x) \in [\varepsilon, \varepsilon + 1]$.

Para simplificar aún más el modelo de la *RHONN*, se define el vector z_I de dimensión L, tal que:

$$z_I(x,u) = \begin{bmatrix} z_{I1} \\ z_{I2} \\ \vdots \\ z_{IL} \end{bmatrix} = \begin{bmatrix} \prod_{j\in I_1} y_j^{d_j(1)} \\ \prod_{j\in I_2} y_j^{d_j(2)} \\ \vdots \\ \prod_{j\in I_L} y_j^{d_j(L)} \end{bmatrix} \tag{5.5}$$

y de esta manera (5.2) se expresa como sigue:

$$x_i = -a_i x_i + \sum_{k=1}^{L} w_{ik} z_{Ik} \tag{5.6}$$

Más aún, si se define el vector de pesos como $w_i = \begin{bmatrix} w_{i1} & w_{i2} & \cdots & w_{iL} \end{bmatrix}^T$ entonces:

$$x_i = -a_i x_i + w_i^T z_I(x,u) \tag{5.7}$$

donde $a_i > 0$, $i = 1,2,\cdots,n$; por lo tanto el modelo de la *RHONN*, queda como sigue:

$$x = Ax + W^T z_i(x,u) \tag{5.8}$$

donde:

$x = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}^T \in \Re^n$ $W = \begin{bmatrix} w_1 & \cdots & w_n \end{bmatrix} \in \Re^{nxL}$ $A = diag\{-a_1 \quad -a_2 \quad \cdots \quad -a_n\} \in \Re^{nxn}$

y se supone que $a_i > 0$.

Considérese el problema de aproximación de un sistema dinámico no lineal de la forma:

$$\chi = F(\chi,u) \tag{5.9}$$

donde $\chi \in \Re^n$ es el estado del sistema; $u \in \Re^m$ es la entrada al sistema y $F : \Re^{n+m} \to \Re^n$ es un campo vectorial suave definido en un conjunto compacto $\mathcal{Y} \in \Re^{n+m}$ En [19] se ha demostrado que para cualquier función continua (campo vectorial) F como (5.9), para cualquier conjunto compacto $\mathcal{Y} \in \Re^{n+m}$ y para cualquier $\varepsilon > 0$, existe un entero L y un conjunto de pesos $W^* \in \Re^{Lxn}$ tal que el modelo (5.8) con L conexiones de alto orden y pesos $W = W^*$ satisface:

$$\sup_{\chi,u\in y} \left\| F(\chi,u) + A\chi - W^{*T} z(\chi,u) \right\| < \varepsilon \tag{5.10}$$

En otras palabras, el modelo (5.8) puede aproximar cualquier sistema de la forma de (5.9) con el grado de precisión deseado, siempre y cuando el número de conexiones L sea seleccionado adecuadamente [18].

5.2 Discretización de la *RHONN*.

Frecuentemente los modelos discretos pueden ser vistos como una discretización numérica de un modelo continúo. Se desea que las propiedades del modelo continuo trasciendan la discretización y pasen al modelo discreto. En [40] se analizan, desde el punto de vista dinámico, las condiciones bajo las cuales los modelos neuronales discretos heredan el comportamiento dinámico de su contraparte continua.

Ciertamente, es deseable que el modelo discreto, cuando es derivado a partir de una aproximación numérica de un modelo continuo, conserve las características dinámicas de este último. Una vez establecido esto, el modelo discreto puede usarse por completo sin pérdida de similitud funcional del modelo continuo y preservando las características biológicas que el modelo continuo posea.

En [40] se establece un criterio general llamado *"el criterio de la consistencia asintótica"* el cual es una condición suficiente y necesaria para que el modelo discreto resultante de la aplicación del método de *Euler*, tenga un comportamiento asintóticamente consistente con su modelo continuo correspondiente. Los resultados de [40] están basados en un análisis local.

5.2.1 Discretización de un modelo neuronal.

Considérese el siguiente modelo continuo:

$$x = F(x) \quad x \in \Re^n \quad F \in C^1 \quad F : \Re^n \to \Re^n \tag{5.11}$$

La discretización de (5.11) por el método de *Euler*, con una constante de tiempo $\delta > 0$, queda como:

$$x(k+\delta) = x(k) + \delta F(x) = f_\delta(x) \qquad f_\delta(x) : \Re^n \to \Re^n \tag{5.12}$$

Para un punto fijo $x = x^*$ los Jacobianos de F y f_δ respectivamente son: $DF(x^*)$ y $Df_\delta(x^*)$ donde:

$$Df_\delta(x^*) = I + \delta DF(x^*) \tag{5.13}$$

Los valores propios Λ de $DF(x^*)$ y λ de $Df_\delta(x^*)$ están relacionados por:

$$\lambda = 1 + \delta\Lambda \tag{5.14}$$

Considérese el siguiente modelo neuronal:

$$F(x) = -x + G(Wx + J) \tag{5.15}$$

donde $G(z) = \left[g_{\mu_1}(z_1) \quad \cdots \quad g_{\mu_n}(z_n) \right]^T$; W es la matriz que contiene los pesos de la red neuronal; x son los estados de la red neuronal y J son las entradas externas a la red neuronal. La aproximación numérica de (5.15) da lugar al siguiente modelo discreto:

$$f_\delta(x) = x + \delta F(x) = (1-\delta)x + \delta G(Wx + J) \tag{5.16}$$

Cada función de activación g_{μ_i}, se supone que toma la forma $g_{\mu_i}(z_i) = \Psi(\mu_i, z_i)$, donde Ψ es una función C^1 que satisface:

- $\Psi(z) \to \pm 1$ cuando $z \to \pm\infty$;
- $\Psi'(z) > 0$;
- $\Psi'(z) \to 0$ cuando $z \to \pm\infty$;
- $\Psi'(z)$ toma un valor máximo local en 1 cuando $z = 0$, tal que $g'_{\mu_i}(z)$ logra su valor máximo μ_i en el origen $z = 0$;

Ejemplos típicos de Ψ son la tangente hiperbólica y la función logística. Para la función logística se tiene:

$$\begin{aligned} \Psi(z) &= \frac{1}{1+e^{-z}} \\ g_{\mu_i}(z) &= \frac{\mu_i}{1+e^{-z}} \end{aligned} \tag{5.17}$$

En el punto fijo $x = x^*$ los jacobianos de F y f_δ respectivamente son:

$$\begin{aligned} DF(x^*) &= -I + G'(Wx^* + J)W \\ Df_\delta(x^*) &= I + \delta DF(x^*) = (1-\delta)I + \delta G'(Wx^* + J)W \end{aligned} \tag{5.18}$$

donde I es la matriz identidad y $G'(z) = diag\{g'_{\mu_i}(z_i)\}$ es la derivada de G en z. Los valores propios λ y Λ satisfacen la relación establecida en (5.14).

Teorema 5.1. Suponiendo que $x = x^*$ es un punto fijo para el modelo contínuo definido en (5.15) y para el modelo discreto (5.16) y que todos los valores propios de $DF(x^*)$ son reales. Si x^* es un punto fijo asintóticamente estable de F, entonces el modelo discreto es asintóticamente consistente con el modelo continuo en x^* si y solo si δ es tal que la matriz

$$N(W,\delta) \equiv M_{\tau^*} W + \frac{2-\delta}{\delta} I$$

tiene solo valores propios positivos, donde:

$$M_{\tau^*} = G'(Wx^* + J) = diag\left\{g'_{\mu_i}\left(\sum_{j=1}^{n} w_{ij} x_i^* + J_i\right)\right\}$$

Nota: Si x^* es un punto fijo asintóticamente inestable del modelo continuo, el modelo discreto es asintóticamente consistente con el modelo continuo en $x = x^*$ para toda δ

La prueba de este teorema se puede encontrar en [40].

Teorema 5.2. Para redes neuronales con W en forma triangular (posiblemente después de renombrar algunas neuronas), una condición suficiente para que el modelo discreto definido en (5.16) sea asintóticamente consistente con el modelo continuo (5.15) en $x = x^*$ es:

$$\mu_i w_{ii} + \frac{2-\delta}{\delta} > 0$$

La condición también es necesaria cuando todas las entradas externas $J = 0$.

Prueba

Del teorema 5.1 sabemos que para que (5.16) sea asintóticamente consistente con (5.15), la matriz

$$N(W,\delta) \equiv M_{\tau^*} W + \frac{2-\delta}{\delta} I$$

debe tener valores propios positivos; para el caso particular que aquí se plantea, se tiene:

$$N(W,\delta) = \begin{bmatrix} g'_{\mu_1}(z_1) & & & \\ & g'_{\mu_2}(z_2) & & \\ & & \ddots & \\ & & & g'_{\mu_n}(z_n) \end{bmatrix} \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ 0 & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & w_{nn} \end{bmatrix} + \frac{2-\delta}{\delta} \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}$$

$$N(W,\delta)=\begin{bmatrix} g'_{\mu_1}(z_1)w_{11}+\dfrac{2-\delta}{\delta} & g'_{\mu_1}(z_1)w_{12} & \cdots & g'_{\mu_1}(z_1)w_{1n} \\ 0 & g'_{\mu_2}(z_2)w_{22}+\dfrac{2-\delta}{\delta} & \cdots & g'_{\mu_2}(z_2)w_{2n} \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & g'_{\mu_n}(z_n)w_{nn}+\dfrac{2-\delta}{\delta} \end{bmatrix}$$

como el valor máximo de $g'_{\mu_i}(z_i)$ es μ_i cuando $z=0$, se tiene:

$$N(W,\delta)\le N_1(W,\delta)$$

con:

$$N_1(W,\delta)=\begin{bmatrix} \mu_1 w_{11}+\dfrac{2-\delta}{\delta} & \mu_1 w_{12} & \cdots & \mu_1 w_{1n} \\ 0 & \mu_1 w_{22}+\dfrac{2-\delta}{\delta} & \cdots & \mu_1 w_{2n} \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \mu_1 w_{nn}+\dfrac{2-\delta}{\delta} \end{bmatrix}$$

por la forma de $N_1(W,\delta)$, sus valores propios son los elementos de la diagonal, entonces del teorema 5.1 se tiene que la condición es:

$$\mu_i w_{ii}+\frac{2-\delta}{\delta}>0$$

cuando $\delta=1$ la condición se reduce a:

$$\mu_i w_{ii}+1>0 \quad \therefore \quad \mu_i w_{ii}>-1$$

5.2.2 Discretización de la RHONN.

En el modelo de la *RHONN* (5.8), si se considera que $A=-I$ donde I es la matriz identidad, el modelo (5.8) se reduce al modelo presentado en (5.15) y su discretización por el método de *Euler* queda como en (5.16), y para $\delta=1$ se tiene:

$$x(k+1)=G\left(W(k)x(k)+J(k)\right)=W^T(k)z_I\left(x(k),u(k)\right) \tag{5.19}$$

o bien para los estados de la *RHONN* se tiene:

$$x_i(k+1) = w_i^T(k) z_i(x(k), u(k)) \tag{5.20}$$

con x, w, z y u como en (5.7). La condición establecida por el *criterio de consistencia asintótica* [40] queda determinada por el teorema 5.1 para el caso general y para el caso de W triangular, queda determinada por el teorema 5.2, con $\delta = 1$.

5.3 La *RHONN* entrenada con el *FKE*.

La ecuaciones del *FKE* para el entrenamiento de redes neuronales, se establecieron en el capítulo anterior en la sección 4.1 y son:

$$\begin{aligned} K(k) &= P(k) H^T(k) \left[R(k) + H(k) P(k) H^T(k) \right]^{-1} \\ w(k+1) &= w(k) + K(k) \left[y(k) - \hat{y}(k) \right] \\ P(k+1) &= P(k) - K(k) H(k) P(k) + Q(k) \end{aligned} \tag{5.21}$$

con:

$$H_{ij}(k) = \left[\frac{\partial \hat{y}_i(k)}{\partial w_j} \right]_{w_j = w_j(k)}, i = 1 \cdots m, \; j = 1 \cdots L \tag{5.22}$$

Para el caso de una red neuronal en representación en espacio de estados se tiene:

$$\hat{y}(k) = \phi(x, u) \tag{5.23}$$

donde ϕ es función de los estados de la red neuronal y en el caso más general también es función de las entradas externas de la red neuronal, así que H se debe calcular de manera diferente a como se calculó en el capítulo anterior. Por la regla de la cadena se tiene que:

$$\frac{\partial \hat{y}(k)}{\partial w} = \frac{\partial \hat{y}(k)}{\partial x(k)} \frac{\partial x(k)}{\partial w} \tag{5.24}$$

donde x es el estado de la neurona, rescribiendo (5.19) como (5.25); de tal forma que en este caso la linealización se logra con una rutina de derivadas recurrentes [11], [43], dando como resultado un sistema dinámico en la forma de (5.26).

$$x(k+1) = T(x(k), u(k), w(k)) \tag{5.25}$$

Consecuentemente:

$$\frac{\partial x(k+1)}{\partial w} = \frac{\partial T\left(x(k),u(k),w(k)\right)}{\partial x(k)}\frac{\partial x(k)}{\partial w} + \frac{\partial T\left(x(k),u(k),w(k)\right)}{\partial w} \tag{5.26}$$

donde $T(\bullet,\bullet,\bullet)$ es una función no lineal de $x(k)$, $u(k)$ y $w(k)$ que determina la transición del estado de la neurona. Debido a que se supone que el estado inicial de la red neuronal no tiene una dependencia funcional de los pesos [43], se tiene que $H_{ij}(0) = 0$.

5.3.1 Predicción a un paso de la serie de tiempo de la demanda eléctrica utilizando el FKE como algoritmo de entrenamiento para una RHONN.

Como no se tiene una representación en espacio de estados para la serie de tiempo de la demanda eléctrica, en este trabajo se considerará que dicha serie de tiempo puede ser representada por la forma canónica:

$$\begin{aligned}
\chi_1(k+1) &= \chi_2(k)\\
\chi_2(k+1) &= \chi_3(k)\\
\chi_3(k+1) &= \chi_4(k)\\
\chi_4(k+1) &= \chi_5(k)\\
\chi_5(k+1) &= \chi_6(k)\\
\chi_6(k+1) &= \chi_7(k)\\
\chi_7(k+1) &= \alpha\left(\chi_1(k),\cdots,\chi_7(k)\right)\\
y(k) &= \chi_1(k)
\end{aligned} \tag{5.27}$$

donde α es una función no lineal de los estados y de las entradas, se consideran 7 estados, ya que en el capítulo anterior se determinó que ese es el orden del sistema. La red neuronal que se propone es:

$$\begin{aligned}
x_1(k+1) &= w_{11}S\left(x_1(k)\right) + w_{12}(k)x_2(k)\\
x_2(k+1) &= w_{21}S\left(x_2(k)\right) + w_{22}(k)x_3(k)\\
x_3(k+1) &= w_{31}S\left(x_3(k)\right) + w_{32}(k)x_4(k)\\
x_4(k+1) &= w_{41}S\left(x_4(k)\right) + w_{42}(k)x_5(k)\\
x_5(k+1) &= w_{51}S\left(x_5(k)\right) + w_{52}(k)x_6(k)\\
x_6(k+1) &= w_{61}S\left(x_6(k)\right) + w_{62}(k)x_7(k)\\
x_7(k+1) &= w_{71}S\left(x_7(k)\right)\\
\hat{y}(k) &= x_1(k)
\end{aligned} \tag{5.28}$$

donde $S(\bullet)$ es una función sigmoidal definida como en (5.4). El estado de la red neuronal está definido como x_1, mientras que χ_1 es el estado de la serie de tiempo de la demanda eléctrica; por lo tanto el error que se pretende minimizar con el *FKE*, está dado por:

$$e_1(k) = \chi_1(k) - x_1(k) \tag{5.29}$$

Considerando la notación anteriormente, (5.29) puede rescribirse como:

$$e_1(k) = y_1(k) - \hat{y}_1(k) \tag{5.30}$$

con:

$$\begin{aligned} \hat{y}_1(k) &= x_1(k) \\ y_1(k) &= \chi_1(k) \end{aligned} \tag{5.31}$$

En el caso de las *RHONN* es común que la ley de adaptación de pesos se calcule de manera separada para cada estado de la red neuronal, [18], [31],[8]. Para seguir esa práctica en este trabajo se planteará un *FKE* para cada estado de la red neuronal. Para ejemplificar esto se tomará el primer estado de la red neuronal planteada en (5.28). Cabe destacar que también es posible plantear un solo *FKE* para adaptar todos los pesos.

$$\begin{aligned} K_1(k) &= P_1(k) H_1^T(k) \left[R_1(k) + H_1(k) P_1(k) H_1^T(k) \right]^{-1} \\ W_1(k+1) &= W_1(k) + K_1(k) \left[y_1(k) - \hat{y}_1(k) \right] \\ P_1(k+1) &= P_1(k) - K_1(k) H_1(k) P_1(k) + Q_1(k) \end{aligned} \tag{5.32}$$

donde H_1 se calcula como:

$$H_1(k) = \left[\frac{\partial \hat{y}_1(k)}{\partial w} \right]_{w=w(k)} \tag{5.33}$$

con:

$$\frac{\partial \hat{y}_1(k)}{\partial w} = \frac{\partial \hat{y}_1(k)}{\partial x_1(k)} \frac{\partial x_1(k)}{\partial w} \tag{5.34}$$

tomando en cuenta (5.31):

$$\frac{\partial \hat{y}_1(k)}{\partial x_1(k)} = 1 \tag{5.35}$$

y de (5.28) se tiene:

$$x_1(k+1) = T(x(k), w) = W_1^T z_1(x(k)) \tag{5.36}$$

con

$$W_1 = [w_{11} \quad w_{12}]$$

$$z_1 = \begin{bmatrix} \dfrac{\mu_1}{1+e^{-\beta_1 x_1}} + \varepsilon \\ x_2 \end{bmatrix}$$

$$\begin{aligned} \frac{\partial x_1(k+1)}{\partial w} &= \frac{\partial T(x(k), w)}{\partial x_1(k)} \frac{\partial x_1(k)}{\partial w} + \frac{\partial T(x(k), w)}{\partial w} \\ \frac{\partial x_1(k+1)}{\partial w} &= W_1^T z_1'(x(k)) \frac{\partial x_1(k)}{\partial w} + z_1(x(k)) \end{aligned} \tag{5.37}$$

donde $z'_1(x(k))$ es la derivada de $z_1(x(k))$ con respecto a $x_1(k)$:

$$z'_1(x(k)) = \begin{bmatrix} \dfrac{\mu_1 \beta_1 e^{-\beta_1 x_1}}{\left(1+e^{-\beta_1 x_1}\right)^2} \\ 0 \end{bmatrix} \tag{5.38}$$

Se realiza el mismo procedimiento para cada estado de la red neuronal. Los parámetros utilizados para esta aplicación son: $\beta_i = 2e^{-3}$ $\mu_i = 1$ ε=0

Los resultados obtenidos al entrenar la red neuronal (5.28) con el *FKE* se muestran en las figuras (5.1) y (5.2). En éstas se muestran la serie de tiempo y los pesos de la red neuronal, respetivamente.

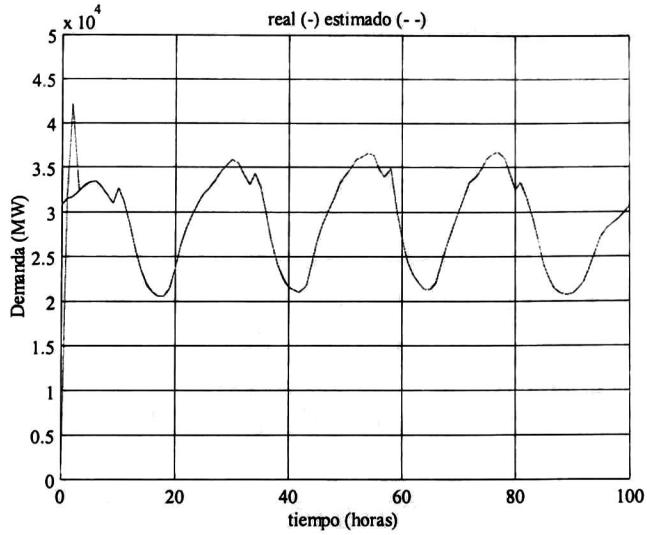


Figura 5.1 Comparación de la serie de tiempo real y la predicción a un paso para la demanda eléctrica con el *FKE*.

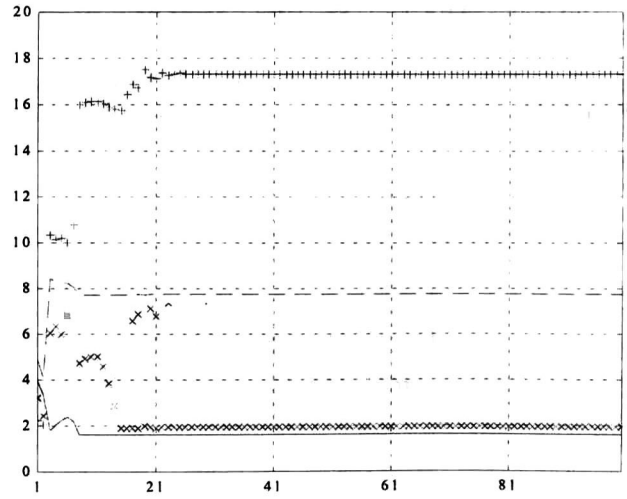


Figura 5.2.- Evolución de los pesos durante el entrenamiento con el *FKE*.

5.4 La *RHONN* entrenada con el *FKELD*.

Para el caso de conexiones de primer orden, la implementación del *FKE* es relativamente fácil, sin embargo, cuando se desean realizar conexiones de alto orden estas incrementa en gran medida la dificultad del cálculo de H que de por si ya es difícil. Es por ello que para el caso de conexiones de alto orden, es recomendable utilizar un algoritmo de entrenamiento basado en el *FKELD;* las ecuaciones relativas a este algoritmo se presentaron en la sección 4.2. Para este algoritmo las ecuaciones quedan determinadas exactamente igual que para el *RMLP*; esto muestra la facilidad de implementación del *FKELD*. Para cada estado de la red neuronal se tiene:

$$\begin{aligned} K(k) &= P_{wy}(k)P_y^{-1}(k) \\ w(k+1) &= w(k) + K(k)\left[y(k) - \bar{y}(k)\right] \\ P(k+1) &= P(k) - K(k)P_y(k)K^T(k) + Q(k) \end{aligned} \tag{5.39}$$

tal y como se definieron en la ecuación (4.12).

5.4.1 Predicción a un paso de la serie de tiempo de la demanda eléctrica utilizando el FKELD como algoritmo de entrenamiento para una RHONN.

Para esta aplicación se considera la misma representación en espacio de estado de la serie de tiempo de la demanda eléctrica mostrada en (5.26), y se propone la siguiente red neuronal:

$$\begin{aligned} x_1(k+1) &= w_{11}S\left(x_1(k)\right)^{d_1} + w_{12}(k)x_2(k) \\ x_2(k+1) &= w_{21}S\left(x_2(k)\right)^{d_2} + w_{22}(k)x_3(k) \\ x_3(k+1) &= w_{31}S\left(x_3(k)\right)^{d_3} + w_{32}(k)x_4(k) \\ x_4(k+1) &= w_{41}S\left(x_4(k)\right)^{d_4} + w_{42}(k)x_5(k) \\ x_5(k+1) &= w_{51}S\left(x_5(k)\right)^{d_5} + w_{52}(k)x_6(k) \\ x_6(k+1) &= w_{61}S\left(x_6(k)\right)^{d_6} + w_{62}(k)x_7(k) \\ x_7(k+1) &= w_{71}S\left(x_7(k)\right)^{d_7} \\ \hat{y}(k) &= x_1(k) \end{aligned} \tag{5.40}$$

con $d_i = 8$; $S(\bullet)$ es una función sigmoidal definida como en (5.4). Con el *FKE* es difícil tener conexiones de alto orden, ya que la derivada que ya de por si es complicada se complica más; sin embargo, en el caso del *FKELD* resulta muy fácil de programar. Los parámetros utilizados en esta aplicación son: $\beta_i = 2e^{-3}$ $\mu_i = 1$ ε=0

Los resultados obtenidos al entrenar la red neuronal (5.40) con el *FKE* se muestran en las figuras (5.3) y (5.4). En las que se muestran la serie de tiempo y los pesos de la red neuronal, respectivamente.

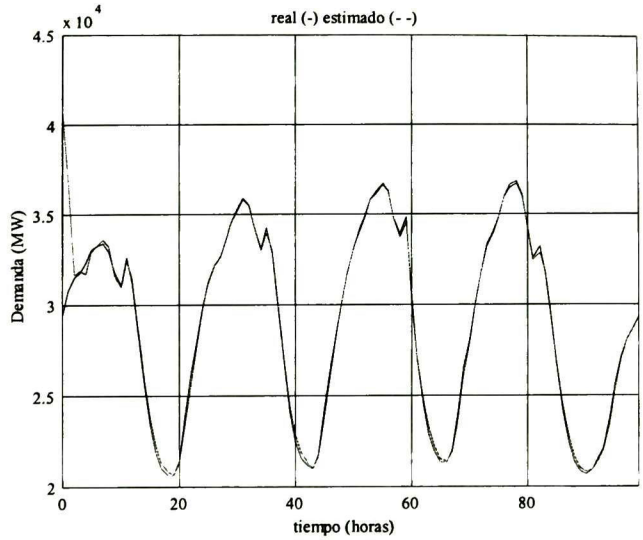


Figura 5.3.- Comparación de la serie de tiempo real y la predicción a un paso para la demanda eléctrica con el *FKELD*.

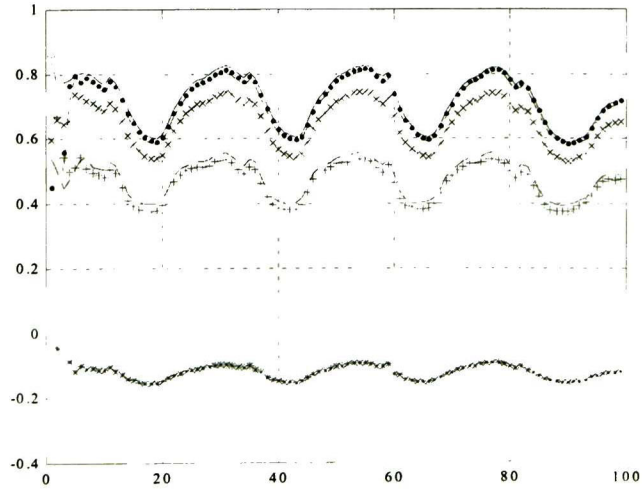


Figura 5.4.- Evolución de los pesos durante el entrenamiento con el *FKELD*.

5.5 La *RHONN* entrenada con el *FKEFO*.

Otra modalidad del *FKE* surge al agregar un factor de olvido, el *FKE* como se presentó en el *capítulo 3* en la sección 3.6. Ahora se plantea para el entrenamiento de redes neuronales. Las ecuaciones del *FKEFO* están dadas por:

$$\begin{aligned} K(k) &= P(k)H^T(k)\left[\lambda(k)R(k)+H(k)P(k)H^T(k)\right]^{-1} \\ w(k+1) &= w(k)+K(k)\left[y(k)-\hat{y}(k)\right] \\ P(k+1) &= \lambda^{-1}(k)P(k)-\lambda^{-1}(k)K(k)H(k)P(k)+Q(k) \end{aligned} \tag{5.41}$$

donde $\lambda(k)$ es el factor de olvido; $P(k)\in\Re^{L\times L}$ y $P(k+1)\in\Re^{L\times L}$ representan la matriz de covarianza del error de predicción al tiempo k y $k+1$ respectivamente; $w\in\Re^{L}$ es el vector de pesos (estados); L es el número total de pesos de la red neuronal; $y\in\Re^{m}$ es el vector de salidas, donde m es el número total de salidas; $\hat{y}\in\Re^{m}$ es la salida de la red neuronal; $K\in\Re^{L\times m}$ es la matriz de ganancia de Kalman; $Q\in\Re^{L\times L}$ es la matriz de covarianza del ruido de proceso; y $R\in\Re^{m\times m}$ es la matriz de covarianza del ruido de medición. $H\in\Re^{m\times L}$ es la matriz que contiene las derivadas de cada salida de la red neuronal $(\hat{y}_i)$ con respecto a cada uno de los pesos (w_j), tal y como se define en (5.22).

5.5.1 Predicción a un paso de la serie de tiempo de la demanda eléctrica utilizando el FKEFO como algoritmo de entrenamiento para una RHONN.

La representación utilizada es la misma que se utilizó para el *FKE* (5.26). La red neuronal propuesta también es la misma del *FKE* (5.28), las ecuaciones del *FKEFO* se presentaron en (5.41), y se aplican igual que las del *FKE* con H calculada de la misma forma (5.33). Los parámetros utilizados para esta aplicación son: $\beta_i = 2e^{-3}$ $\mu_i = 1$ ε=0 $\lambda(k) = 0.83$

Los resultados obtenidos al entrenar la red neuronal (5.40) con el *FKE* se muestran en las figuras (5.5) y (5.6), en las cuales se muestra la serie de tiempo y los pesos de la red neuronal.

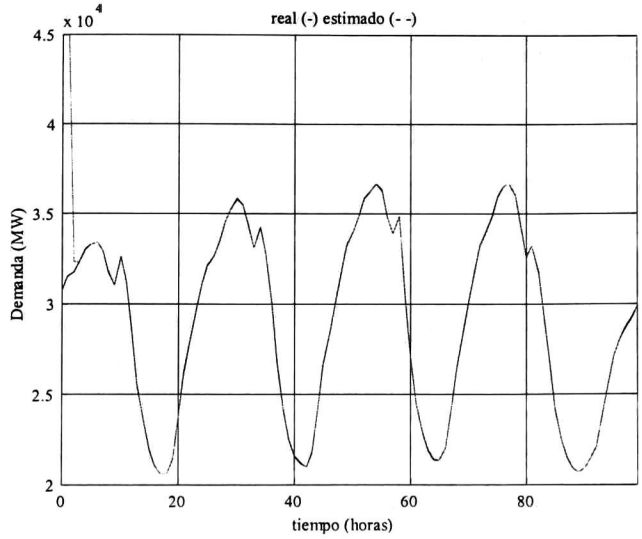


Figura 5.5.- Comparación de la serie de tiempo real y la predicción a un paso para la demanda eléctrica con el *FKEFO*.

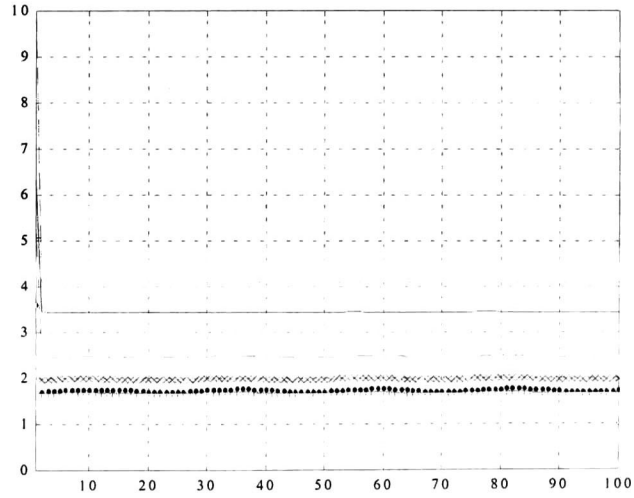


Figura 5.6.- Evolución de los pesos durante el entrenamiento con el *FKEFO*.

5.5.2 Predicción a un paso de la serie de tiempo de los precios de la energía eléctrica utilizando el FKEFO como algoritmo de entrenamiento para una RHONN.

Al igual que para la serie de tiempo de la demanda eléctrica, para esta serie de tiempo tampoco se tiene una representación en espacio de estados, en este trabajo se considerará que dicha serie de tiempo puede ser representada por la forma canónica:

$$\begin{aligned}
\chi_1(k+1) &= \chi_2(k) \\
\chi_2(k+1) &= \chi_3(k) \\
&\vdots \\
\chi_9(k+1) &= \chi_{10}(k) \\
\chi_{10}(k+1) &= \alpha\left(\chi_1(k),\cdots,\chi_{10}(k)\right) \\
y(k) &= \chi_1(k)
\end{aligned} \tag{5.42}$$

donde α es una función no lineal de los estados y de las entradas, se consideran 10 estados, ya que en el capítulo anterior se determinó que ese es el orden del sistema. La red neuronal que se propone es:

$$\begin{aligned}
x_1(k+1) &= w_{11}S\left(x_1(k)\right) + w_{12}x_2(k) \\
x_2(k+1) &= w_{21}S\left(x_2(k)\right) + w_{22}x_3(k) \\
&\vdots \\
x_9(k+1) &= w_{91}S\left(x_9(k)\right) + w_{92}x_{10}(k) \\
x_{10}(k+1) &= w_{101}S\left(x_{10}(k)\right) \\
\hat{y}(k) &= x_1(k)
\end{aligned} \tag{5.43}$$

donde $S(\bullet)$ es una función sigmoidal definida como en (5.4). El estado de la red neuronal esta definido como x, mientras que χ es el estado de la serie de tiempo de los precios de la energía eléctrica.

Las ecuaciones del *FKEFO* se presentaron en (5.41), y se aplican igual que las del *FKE* con H calculada de la misma forma (5.33). Los parámetros utilizados para esta aplicación son: $\beta_i = 2e^{-3}$ $\mu_i = 1$ ε=0 $\lambda(k) = 0.85$

Los resultados obtenidos al entrenar la red neuronal (5.43) con el *FKE* se muestran en las figuras (5.7) y (5.8), en las cuales se muestra la serie de tiempo y los pesos de la red neuronal.

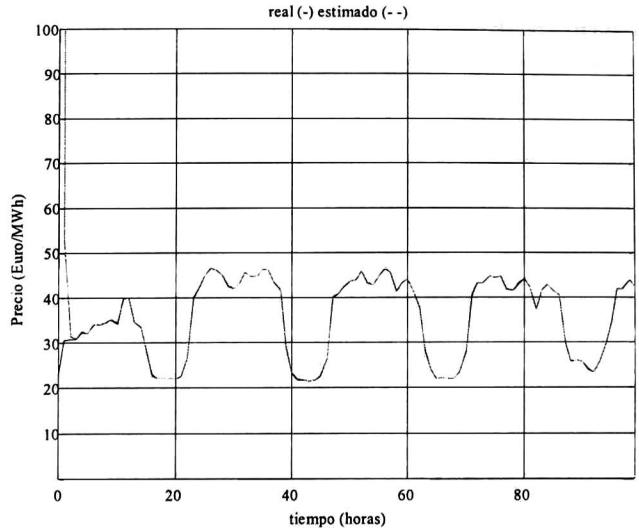


Figura 5.7.- Comparación de la serie de tiempo real y la predicción a un paso para los precios de la energía eléctrica con el *FKEFO*.

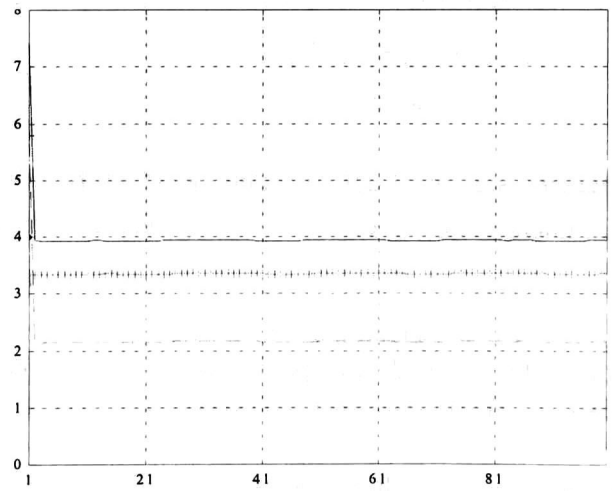


Figura 5.8.- Evolución de los pesos durante el entrenamiento con el *FKEFO*.

Capítulo 6

REPRODUCCIÓN NEURONAL DE CAOS.

En este capítulo se presenta la tercera aportación de esta tesis: la reproducción neuronal de caos. Los sistemas no lineales tienen ricos comportamientos dinámicos, que incluyen caos. El comportamiento caótico es caracterizado por su alta sensibilidad a la variación de parámetros, perturbaciones externas y particularmente a los pequeños cambios en las condiciones iniciales. El caos presenta algunas importantes propiedades que en los últimos años han dado lugar a aplicaciones poco tradicionales. En los ejemplos que se presentan en este capítulo, la identificación de las series de tiempo se realizará bajo la suposición de que los datos de las mismas fueron obtenidos a partir de un sistema (caótico) presumiblemente desconocido.

6.1 Introducción.

En la literatura no existe una definición unificada de caos. Hablando estrictamente, la característica principal del caos es el aparente comportamiento aleatorio de un sistema dinámico que por naturaleza es determinístico. El caos se puede definir como un comportamiento en estado estable acotado, pero que no es un punto de equilibrio, ni es periódico y tampoco es cuasi periódico. El caos es un fenómeno exclusivo de los sistemas no lineales [5], [17].

Una característica del caos es la extrema sensibilidad del sistema dinámico a sus condiciones iniciales; las trayectorias que parten de valores iniciales muy cercanos se comportan de forma diferente, acercándose y alejándose de forma impredecible. Esto también es conocido como el efecto *mariposa* un término utilizado en estudios atmosféricos, que significa que el simple aleteo de una mariposa puede alterar las condiciones iniciales del sistema climático y así pueden darse cambios significativos en los patrones climáticos futuros. Poincaré fue quien observó este fenómeno [5]: "Puede suceder que pequeñas diferencias en las condiciones iniciales produzcan grandes diferencias en el fenómeno final. *La predicción se hace imposible*"

Otras características del caos incluyen, la existencia de un conjunto denso de órbitas periódicas inestables en su régimen, exponentes de Lyapunov positivos, una entropía de Kolomogorov-Sinai finita y espectro de potencia continuo [5].

De entre todas las características de caos, el *exponente positivo de Lyapunov* es quizá el más fácil de verificar. Sea $x(t)$ la trayectoria de un sistema dinámico, cuya estado inicial es $x(0)$. Si dicho estado evoluciona a una velocidad exponencial, entonces éste se puede aproximar como: $|x(t)| \approx |x(0)|e^{\lambda t}$ donde $\lambda < 0$ si la trayectoria es afectada por algunos *atractores* (puntos, órbitas, conjuntos o regiones estables) y $\lambda > 0$ si es divergente y muy sensitiva a pequeñísimas perturbaciones del estado inicial. Esta característica permite definir el concepto del exponente de Lyapunov, definido como sigue:

Considérese una transformación escalar en tiempo discreto:

$$x_{k+1} = f(x_k) \tag{6.1}$$

con dos condiciones iniciales cercanas x_0 y y_0. Sean:

$$x_k = f(x_{k-1}) = \cdots = f^k(x_0) \text{ y } y_k = f^k(y_0) \tag{6.2}$$

Si estas dos posiciones, x_k y y_k, se separan a velocidad exponencial en las siguientes iteraciones, entonces:

$$|y_k - x_k| = |y_0 - x_0|e^{k\lambda} \quad (\lambda > 0) \tag{6.3}$$

y

$$\frac{1}{k}\ln|y_k - x_k| \to \lambda \text{ cuando } k \to \infty \tag{6.4}$$

En el caso de que el movimiento sea dentro de una región acotada, la separación exponencial, ocurre independientemente de que las dos posiciones iniciales sean muy cercanas entre sí. Así que $|y_0 - x_0| \to 0$ antes de tomar el limite $k \to \infty$, para definir la siguiente constante:

$$\begin{aligned} \lambda &= \lim_{k\to\infty}\frac{1}{k}\lim_{|y_0-x_0|\to 0}\ln\left|\frac{|y_k - x_k|}{|y_0 - x_0|}\right| = \lim_{k\to\infty}\frac{1}{k}\lim_{|y_0-x_0|\to 0}\ln\left|\frac{|f^k(y_0) - f^k(x_0)|}{|y_0 - x_0|}\right| \\ \lambda &= \lim_{k\to\infty}\frac{1}{k}\ln\left|\frac{df^k(x_0)}{x_0}\right| = \lim_{k\to\infty}\frac{1}{k}\sum_{i=0}^{k-1}\ln\left|\frac{df^k(x_i)}{dx_i}\right| \end{aligned} \tag{6.5}$$

λ es llamado exponente de Lyapunov para la trayectoria $x_k = f^k(x_0)$, $k = 0,1,\cdots$ Para el caso continuo, se usa la trayectoria del vector **x** definido de manera similar:

$$\lambda = \lim_{t\to\infty} \sup \frac{1}{t} \ln|x(t)| \tag{6.6}$$

El exponente de Lyapunov provee una medida para la velocidad media de convergencia (o divergencia) de las trayectorias del sistema dinámico. Para un sistema *n*-dimensional, dependiendo de la dirección (pero no de la posición) del vector de estado inicial, los *n* exponentes de Lyapunov $\lambda_1 \geq \cdots \geq \lambda_n$ describen diferentes tipos de atractores [5].

Algunos sistemas dinámicos que son caóticos o pueden presentar caos son:

- Transformaciones matemáticas típicas y sistemas físicos como: el mapa logístico, mapa de Hénon, mapa Lozi, mapa Ikeda, mapa de Rössler, el movimiento de Bernoulli, la ecuación de Duffing, la ecuación de Lorenz, la ecuación de Navier-Stokes, el circuito de Chua, el circuito de Chen, etc.
- Sistemas Hamiltonianos (de diferentes tipos)
- Filtros digitales, sistemas eléctricos y electrónicos
- Mecanismos celestes (el problema de los tres cuerpos)
- Dinámica de fluidos
- Láser, plasmas, estado sólido y mecanismos cuánticos
- Óptica no lineal
- Reacciones químicas
- Sistemas eléctricos de potencia
- Redes neuronales
- Comportamiento económico
- Sistemas biológicos (corazón, cerebro, población, etc.)

6.2 Análisis de las propiedades caóticas de la curva de la demanda eléctrica.

Tal y como se mencionó anteriormente, algunos sistemas eléctricos de potencia pueden poseer propiedades caóticas. Es por ello que surge la necesidad de investigar las propiedades caóticas de la demanda eléctrica, con base en la serie de tiempo de la misma. Para esto se considera el mismo caso de estudio del *capítulo 4*. Se utilizó una herramienta computacional que permite estudiar algunas propiedades características de los sistemas caóticos, con base en su serie de tiempo; esta herramienta es el *CDA* 1.0 (Chaos Data Analyzer) [36].

El *CDA* es una colección de programas que permite analizar las propiedades caóticas de una serie de tiempo; algunas de estas propiedades son:

- El exponente de Lyapunov más positivo
- Espectro de potencia
- Frecuencias dominantes
- Dimensión Hausdorff

- Dimensión de correlación
- Atractores

Una de las propiedades caóticas de mayor relevancia son los exponentes de Lyapunov, ya que las órbitas caóticas poseen por lo menos uno de estos positivo. Con el *CDA* es posible calcular sólo el exponente de Lyapunov más positivo [36].

Para la serie de tiempo de la demanda eléctrica mostrada en 4.5 [14], se tomaron las primeras 1000 muestras y se analizaron con el *CDA*, dando como resultado que el exponente de Lyapunov más positivo es de 0.361 y la dimensión de la correlación es 2.546 indicando esto que la serie de tiempo de la demanda eléctrica es caótica [36]. El atractor correspondiente a los datos analizados, se muestra en la figura 6.1.

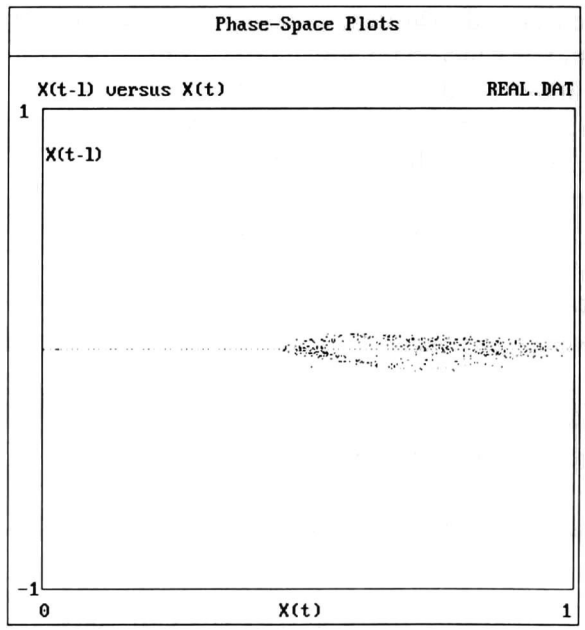


Figura 6.1.- Atractor de la serie de tiempo de la demanda eléctrica.

6.3 Análisis de las propiedades caóticas de los precios de la energía eléctrica.

Los datos de la serie de tiempo de los precios de la energía eléctrica mostrada en la figura 4.17 [26], se analizaron con el *CDA*, dando como resultado que el exponente de Lyapunov más positivo es de 0.476 y la dimensión de la correlación es 3.216 indicando esto que la serie de tiempo de los precios de la energía eléctrica es caótica [36]. El atractor correspondiente a los datos analizados, se muestra en la figura 6.2.

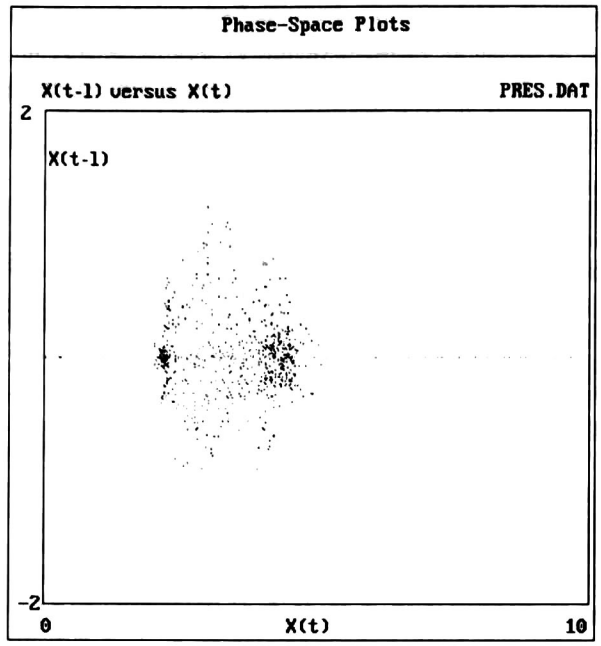


Figura 6.2.- Atractor de la serie de tiempo de los precios de la energía eléctrica

6.4 Reproducción de sistemas caóticos discretos.

Recientemente el control y ordenamiento de caos han recibido bastante atención de diversas comunidades científicas y de ingeniería [6]. El control de sistemas caóticos, al igual que el de cualquier otro sistema no lineal requiere del correcto modelado del mismo. Para muchos sistemas no lineales, especialmente para los sistemas caóticos, es difícil la obtención de modelos matemáticos precisos, esto debido a su compleja estructura física y parámetros ocultos [6].

Las redes neuronales se han desarrollado como una metodología bien establecida para modelado y control, que permite la solución de algunos de los problemas más difíciles de la ingeniería. Esto queda ejemplificado por sus innumerables aplicaciones en la identificación y control de sistemas no lineales complejos. En el *capítulo* anterior se propone el uso de redes neuronales discretizadas, entrenadas con un algoritmo basado en el *FKEFO*; ahora se propone el uso de dichas redes neuronales, para la reproducción de sistemas caóticos discretos, la cual se realiza con base en una predicción a un paso de la serie de tiempo producida por un sistema caótico discreto, presumiblemente desconocido. Los ejemplos de sistemas caóticos que se utilizan en esta sección son:

- Mapa logístico,
- Mapa de Hénon,
- Sistema Ikeda y
- Sistema Lozi.

6.4.1 El mapa logístico.

El mapa logístico fue utilizado, en primera instancia, como un modelo para el crecimiento de la población [11]. Está descrito por (6.1). El mapa logístico presenta propiedades caóticas para $\alpha \in (3.5699, 4]$.

$$\chi(k+1) = \alpha\chi(k)[1-\chi(k)] \tag{6.7}$$

Los parámetros utilizados para esta aplicación son: $\alpha = 4; \chi(0) = 0.5; \lambda(k) = 0.75$ donde $\lambda(k)$ es el factor de olvido. El atractor correspondiente al sistema caótico presentado en (6.7) se muestra en la figura 6.3. La red neuronal entrenada para la reconstrucción de este sistema caótico se propone como:

$$\begin{aligned} x_1(k+1) &= w_{11}S(x_1(k)) + w_{12}x_2(k) \\ x_2(k+1) &= w_{21}S(x_2(k)) \end{aligned} \tag{6.8}$$

donde $\{x_i : i = 1, \cdots, n\}$ son los estados de la red neuronal; $S(\bullet)$ representa la función sigmoidal como en (6.9); y el conjunto $\{w_{ij} : i = 1, \cdots, n, j = 1, \cdots, 2\}$ son los pesos de la red neuronal.

$$S(x) = \frac{1}{1+e^{-\beta x}} + \varepsilon \tag{6.9}$$

donde β es una constante positiva y ε es un número positivo pequeño. Claramente $S(x) \in [\varepsilon, \varepsilon+1]$.

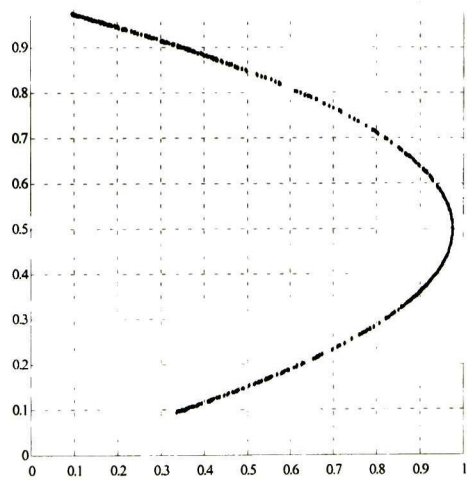


Figura 6.3.- Atractor original del mapa logístico.

Para el entrenamiento de la red neuronal propuesta se utilizaron los datos de la serie de tiempo generados por χ_1. Los resultados se muestran en las figuras 6.4 y 6.5, las cuales presentan el atractor reconstruido por la red neuronal y la serie de tiempo de predicción a un paso lograda por la red neuronal entrenada con el algoritmo del *FKEFO*, respectivamente.

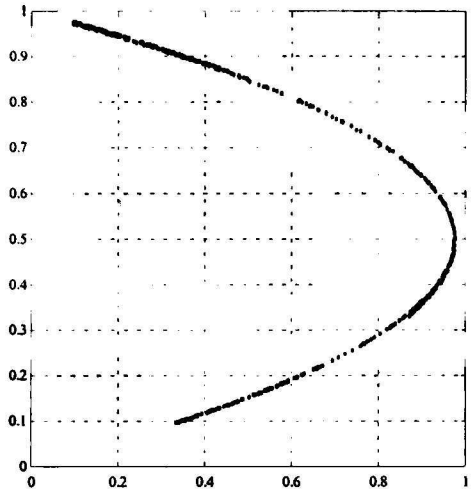


Figura 6.4.- Atractor reproducido por la red neuronal para el mapa logístico.

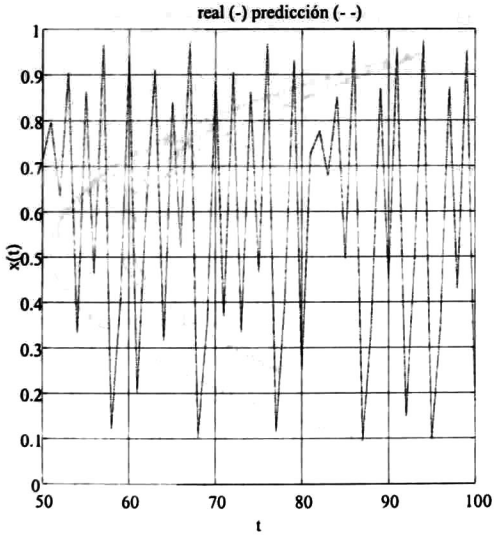


Figura 6.5.- Predicción a un paso para el mapa logístico.

6.4.2 El mapa de Hénon

Otro sistema caótico discreto interesante es el mapa de Hénon [11], el cual está representado por (6.10), con parámetros reales p y q.

$$\begin{aligned} \chi_1(k+1) &= -p\chi_1^2(k) + \chi_2(k) + 1 \\ \chi_2(k+1) &= q\chi_1(n) \end{aligned} \tag{6.10}$$

Los parámetros utilizados para esta aplicación son: $p = 1.4; q = 0.3; \chi_1(0) = 0.4; \chi_2(0) = 0.4; \lambda(k) = 0.78$ donde $\lambda(k)$ es el factor de olvido. El atractor correspondiente al sistema caótico presentado en (6.10) se muestra en la figura 6.6. La red neuronal entrenada para la reconstrucción de este sistema caótico se propone como:

$$\begin{aligned} x_1(k+1) &= w_{11}S(x_1(k)) + w_{12}x_2(k) \\ x_2(k+1) &= w_{21}S(x_2(k)) + w_{22}x_3(k) \\ x_3(k+1) &= w_{31}S(x_3(k)) + w_{32}x_4(k) \\ x_4(k+1) &= w_{41}S(x_4(k)) \end{aligned} \tag{6.11}$$

donde $\{x_i : i = 1, \cdots, n\}$ son los estados de la red neuronal; $S(\bullet)$ representa la función sigmoidal como en (6.9); y el conjunto $\{w_{ij} : i = 1, \cdots, n, j = 1, \cdots, 2\}$ son los pesos de la red neuronal.

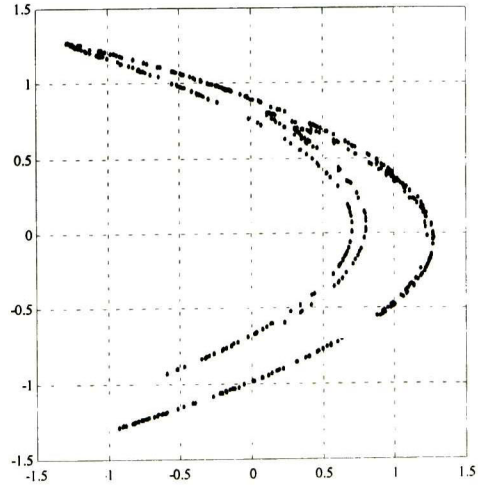


Figura 6.6.- Atractor original para el mapa de Hénon.

Para el entrenamiento de la red neuronal propuesta se utilizaron los datos de la serie de tiempo generados por χ_1. Los resultados se muestran en las figuras 6.7 y 6.8, las cuales presentan el atractor reconstruido por la red neuronal y la serie de tiempo de predicción a un paso lograda por la red neuronal entrenada con el algoritmo del *FKEFO*, respectivamente.

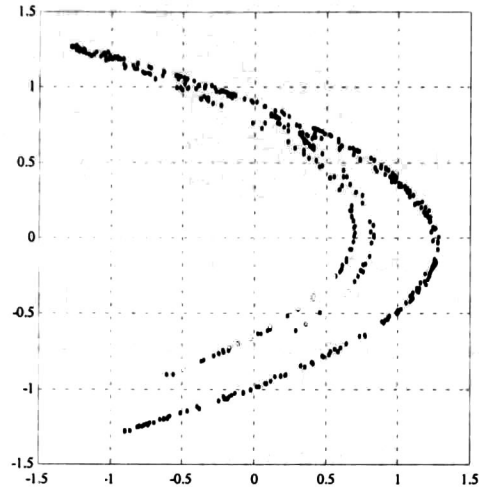


Figura 6.7.- Atractor reproducido por la red neuronal para el mapa de Hénon.

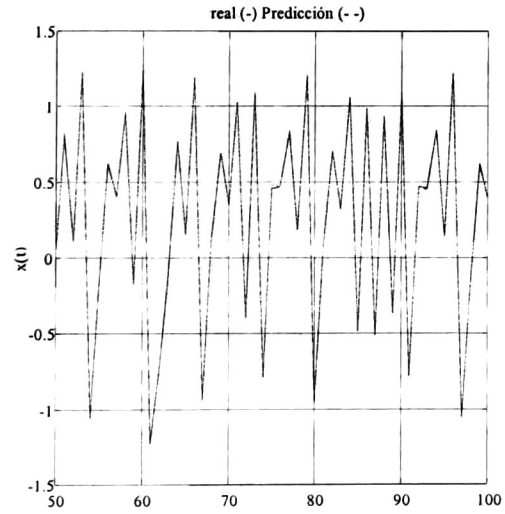


Figura 6.8.- Predicción a un paso para el mapa de Hénon.

6.4.3 El sistema Ikeda.

El sistema Ikeda [11] está descrito por:

$$\begin{aligned}\chi_1(k+1) &= 1+\mu\left\{\chi_1(k)\cos\left[m(k)\right]-\chi_2(k)\sin\left[m(k)\right]\right\}\\ \chi_2(k+1) &= 1+\mu\left\{\chi_1(k)-\chi_2(k)\cos\left[m(k)\right]\right\}\end{aligned} \tag{6.12}$$

con

$$m(k) = 0.4-\frac{6}{1+\chi_1^2(k)+\chi_2^2(k)}$$

Los parámetros utilizados para esta aplicación son: $\mu = 0.9; \chi_1(0) = 0.1; \chi_2(0) = 0.1; \lambda(k) = 0.9$ donde $\lambda(k)$ es el factor de olvido. El atractor correspondiente al sistema caótico presentado en (6.12) se muestra en la figura 6.9. La red neuronal entrenada para la reconstrucción de este sistema caótico se propone como:

$$\begin{aligned}x_1(k+1) &= w_{11}S\left(x_1(k)\right)+w_{12}x_2(k)\\ x_2(k+1) &= w_{21}S\left(x_2(k)\right)+w_{22}x_3(k)\\ x_3(k+1) &= w_{31}S\left(x_3(k)\right)+w_{32}x_4(k)\\ x_4(k+1) &= w_{41}S\left(x_4(k)\right)\end{aligned} \tag{6.13}$$

donde $\{x_i : i = 1,\cdots,n\}$ son los estados de la red neuronal; $S(\bullet)$ representa la función sigmoidal como en (6.9); y el conjunto $\{w_{ij} : i = 1,\cdots,n, j = 1,\cdots,2\}$ son los pesos de la red neuronal.

Para el entrenamiento de la red neuronal propuesta se utilizaron los datos de la serie de tiempo generados por χ_1. Los resultados se muestran en las figuras 6.10 y 6.11 las cuales muestran el atractor reconstruido por la red neuronal y la serie de tiempo de predicción a un paso lograda por la red neuronal entrenada con el algoritmo del *FKEFO*, respectivamente.

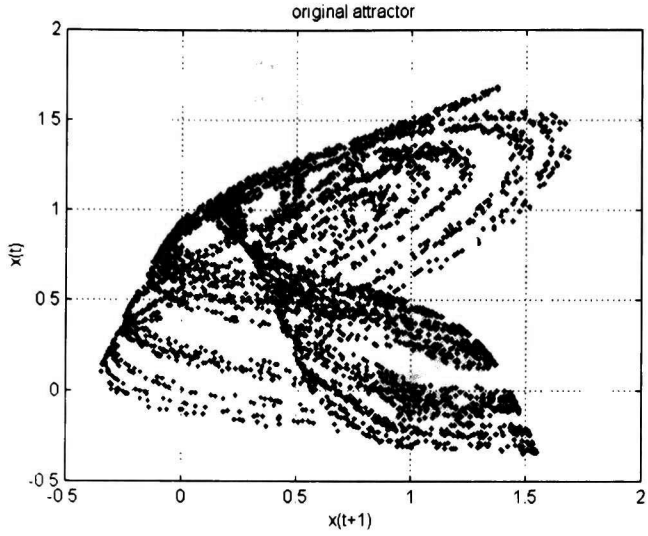


Figura 6.9.- Atractor original del sistema Ikeda.

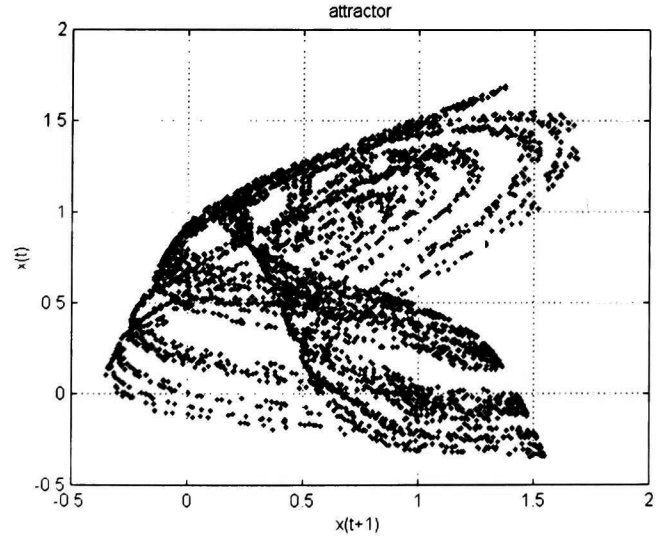


Figura 6.10.- Atractor reproducido por la red neuronal para el sistema Ikeda.

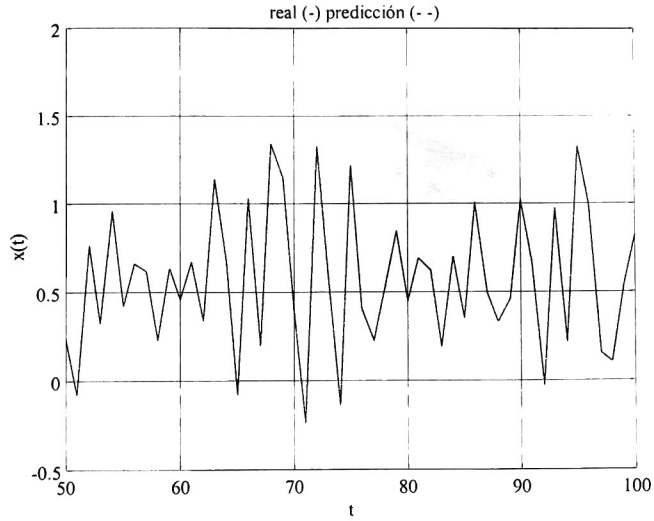


Figura 6.11.- Predicción a un paso para el sistema Ikeda.

6.4.4 El sistema Lozi.

El sistema Lozi [6] está descrito por:

$$\begin{aligned} \chi_1(k+1) &= 1+\mu\left\{\chi_1(k)\cos\left[m(k)\right]-\chi_2(k)\sin\left[m(k)\right]\right\} \\ \chi_2(k+1) &= 1+\mu\left\{\chi_1(k)-\chi_2(k)\cos\left[m(k)\right]\right\} \end{aligned} \tag{6.14}$$

Los parámetros utilizados para esta aplicación son: $p=1.8; q=-1; \chi_1(0)=-5000; \chi_2(0)=5000; \lambda(k)=0.68$ donde $\lambda(k)$ es el factor de olvido. El atractor correspondiente al sistema caótico presentado en (6.14) se muestra en la figura 6.12. La red neuronal entrenada para la reconstrucción de este sistema caótico se propone como:

$$\begin{aligned} x_1(k+1) &= w_{11}S\left(x_1(k)\right)+w_{12}x_2(k) \\ x_2(k+1) &= w_{21}S\left(x_2(k)\right)+w_{22}x_3(k) \\ x_3(k+1) &= w_{31}S\left(x_3(k)\right) \end{aligned} \tag{6.15}$$

donde $\{x_i : i=1,\cdots,n\}$ son los estados de la red neuronal; $S(\bullet)$ representa la función sigmoidal como en (6.9); y el conjunto $\{w_{ij} : i=1,\cdots,n, j=1,\cdots,2\}$ son los pesos de la red neuronal.

Para el entrenamiento de la red neuronal propuesta se utilizaron los datos de la serie de tiempo generados por χ_1. Los resultados se muestran en las figuras 6.13 y 6.14, las cuales presentan el atractor reconstruido por la red neuronal y la serie de tiempo de predicción a un paso lograda por la red neuronal entrenada con el algoritmo del *FKEFO*, respectivamente.

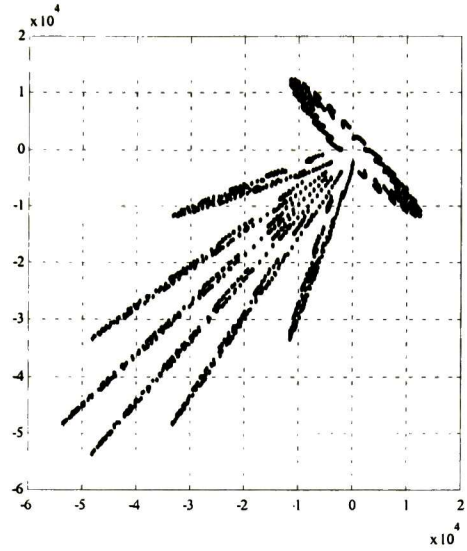


Figura 6.12.- Atractor original del sistema Lozi.

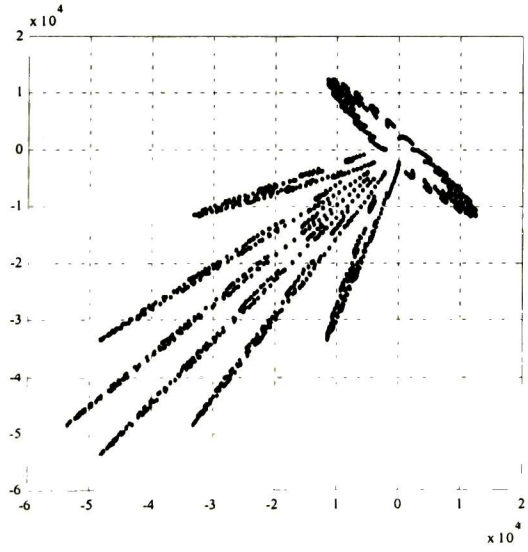


Figura 6.13.- Atractor reproducido por la red neuronal para el sistema Lozi.

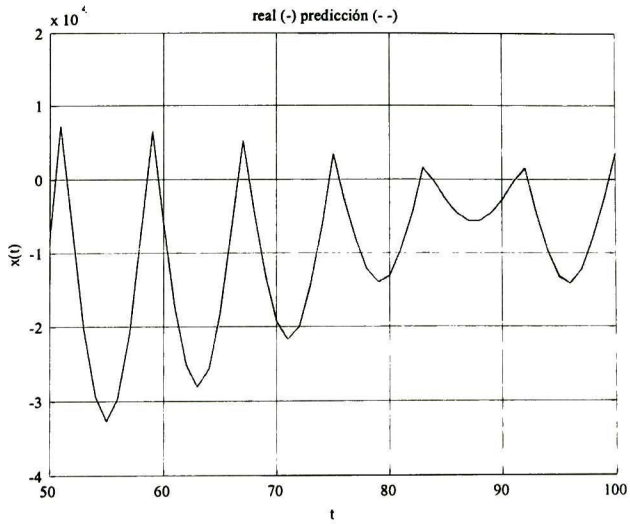


Figura 6.14.- Predicción a un paso para el sistema Lozi.

Capítulo 7

CONCLUSIONES Y TRABAJO FUTURO.

En esta tesis se han propuesto dos algoritmos para el entrenamiento de redes neuronales recurrentes tipo *RMLP*; ambos algoritmos se aplicaron para la predicción de series de tiempo en sistemas eléctricos de potencia; dichas series son la de la demanda eléctrica y la de precios de energía eléctrica, con los resultados presentados en *capítulo 4*. Durante el desarrollo del presente trabajo fue posible observar que los algoritmos aquí propuestos, presentan las siguientes ventajas, sobre otros algoritmos de entrenamiento de redes neuronales: *a)* no son sensibles a los pesos iniciales; *b)* pueden ser usados satisfactoriamente para el modelado de sistemas altamente no lineales; y *c)* no se restringen a redes neuronales de una topología específica, entre otros.

El algoritmo basado en el *FKELD* no alcanzo los resultados obtenidos con el *FKE*, sin embargo su principal atractivo es que posee una complejidad de programación menor, independientemente de la estructura interna de la red neuronal; otro aspecto que destaca al *FKELD* del *FKE* es el no necesitar el cálculo de derivadas.

En conclusión el algoritmo de entrenamiento basado en el *FKE* es el que presenta mejores resultados al ser utilizado como algoritmo de entrenamiento de redes neuronales tipo *RMLP*, mostrando una excelente habilidad de generalización y necesitando menos iteraciones para lograr la exactitud deseada.

Ya que las redes neuronales tipo *RMLP* alcanzaron las expectativas al ser entrenadas con los algoritmos del filtro de Kalman, se procedió a entrenar redes neuronales de alto orden (*RHONN*) con los mismos algoritmos de entrenamiento propuestos para el *RMLP*; además se incluyó otra variante del filtro de Kalman el *FKEFO*, para las *RHONN*, no fue posible obtener buenos resultados en predicción a futuro; sin embargo, fue posible observar que en cuanto a predicción a un paso presenta muy buenos resultados utilizando pocas neuronas.

Como pudo observarse en los ejemplos del *capítulo 5*, la red neuronal propuesta en el mismo, logra realizar predicción a un paso de series de tiempo con muy buena precisión. Nuevamente se mostro que con el algoritmo de entrenamiento para redes neuronales recurrentes basado en el *FKE* es posible obtener buenos resultados y más aún cuando se le agrega el factor de olvido (*FKEFO*), ya que además de obtener buenos resultados, logra estos más rápido que el *FKE* original, pero sobre todo con menos sobreimpulsos *(más suavemente)*.

En lo concerniente al *FKELD* los resultados que presenta son buenos pero no igualan a los obtenidos con otros dos algoritmos (*FKE* y *FKEFO*), sin embargo su aplicación resulta muy sencilla para el caso de que existan conexiones de alto orden. tal y como se observa en los resultados presentados en el *capítulo 5*.

Para los ejemplos presentados en el *capítulo 6*, el número de neuronas utilizado, es menor que el necesario con topologías diferentes [11]. Aunque uno de los principales inconvenientes de la implementación de un algoritmo de entrenamiento basado en el Filtro de Kalman Extendido, se presenta al calcular la linealización, esto puede ser resuelto satisfactoriamente utilizando el cálculo recurrente de las derivadas, tal y como se explicó en el *capítulo 5*. En los ejemplos presentados en el *capítulo 6*, la identificación de las series de tiempo se realizó bajo la suposición de que los datos de las mismas fueron obtenidos a partir de un sistema (caótico) presumiblemente desconocido.

Tal y como se mencionó anteriormente, al utilizar el esquema de las *RHONN* discretizadas, no fue posible obtener buenos resultados en la predicción a 24 horas. Sin embargo fue posible observar, que aún y cuando la predicción no resultó adecuada, si conservó el mismo comportamiento caótico, lo cual puede resultar adecuado para otro tipo de aplicaciones (simuladores, etc) [11].

En lo práctico los pronósticos de demanda eléctrica y de precios de la energía eléctrica son de utilidad para la operación y planeación de redes eléctricas. Los errores de pronóstico presentados en el *capítulo 4* son competitivos con otras alternativas publicadas en la literatura abierta [2], [23], [26], etc. A este respecto, los resultados mostrados en las secciones 4.4.3 y 4.5.3 resultan bastante competitivos ya que el error absoluto promedio que inicialmente se deseaba alcanzar en este trabajo era del 5% o menor para las primeras 24 horas, lo cual fue logrado y superado, ya que para las pruebas realizadas el error absoluto promedio para las primeras 24 horas fue de alrededor del 3% para ambas series de tiempo y además el error absoluto máximo por hora para las pruebas efectuadas en este trabajo fue inferior al 6% para las primeras 24 horas, mientras que en otros trabajos [2], [23], el error absoluto por hora para las primeras 24 horas llega a superar el 10%, y es del orden del 8% o mayor para el error absoluto promedio para las primeras 24 horas.

Es cierto que la demanda eléctrica ha sido objeto de muchos estudios en el pasado. Sin embargo, tales pronósticos han sido restringidos a estimar la demanda en la siguiente media hora o en la siguiente hora. Los retos que imponen los nuevos ambientes de mercado requieren que la demanda eléctrica y los precios de la electricidad sean pronosticados, con buena precisión para las siguientes 24 horas. En este contexto se han probado algunas alternativas pero existe la posibilidad para nuevas contribuciones y más aún para el caso de que la predicción pueda ir mas allá de las primeras 24 horas, como las que se muestran en la tesis.

Los algoritmos sugeridos en la tesis pudiera programarse en un lenguaje como *Fortran* para ser incluido en la biblioteca de funciones de aplicaciones de paquetes de uso industrial, esto con el objetivo de realizar la implementación de los mismos en tiempo real, ya sea para el sistema eléctrico Mexicano o a nivel industrial.

Trabajo futuro

Como trabajo a futuro se consideran importantes los siguientes puntos:

- El análisis teórico de las propiedades de estabilidad de los algoritmos de entrenamiento aquí propuestos, especialmente en el caso de las redes neuronales de alto orden.

- El análisis teórico de las propiedades de convergencia de los algoritmos de entrenamiento aquí propuestos, especialmente en el caso de las redes neuronales de alto orden.

- Establecer las condiciones de convergencia (si existen) de los parámetros al utilizar los algoritmos de entrenamiento basados en el filtro de Kalman, ya que en este aspecto sólo se han encontrado resultados que demuestran convergencia acotada.

- Otro aspecto importante es el probar los algoritmos aquí propuestos con datos del sistema eléctrico Mexicano.

- Programar los algoritmos aquí propuestos en un lenguaje compatible con paquetes de uso industrial, esto con el objetivo de realizar la implementación de los mismos en tiempo real, ya sea para el sistema eléctrico Mexicano o a nivel industrial.

Bibliografía.

[1] D. S. Broomhead, D. Lowe, "Multivariable functional approximation and adaptive networks", *Complex systems*, vol. 2, pp. 321-355, 1988.

[2] J. Contreras, R. Espínola, F. J. Nogales, and A. J. Conejo. "ARIMA models to predict next-day electricity prices", *IEEE Transactions on Power Systems*, vol. 18, no. 3, pp. 1014-1020, 2003.

[3] N. E. Cotter, "The Stone-Weirtrass theorem and its application to neural networks" *IEEE transactions on Neural Networks*, vol. 1, no. 4, pp. 290-295, 1990.

[4] T. M. Cover, "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition", *IEEE Transactions on Electronics Computers*, vol. 14, pp. 326-334, 1965.

[5] G. Chen and X. Dong (eds), "From chaos to order: Methodologies, Perspectives and Applications", World Scientific Pub. Co., Singapore, 1998.

[6] G. Chen, and J. L. Moiola, "An overview of bifurcation, chaos and nonlinear dynamics in control systems" *The Franklin Institute Journal*, vol. 331B, pp. 819-858, 1994.

[7] L. A. Feldkamp, T. M. Feldkamp D. V. Prokhorov, "Neural network training with the nprKF", *Proceeding of International Joint Conference on Neural Networks'01*, pp. 109-114, 2001.

[8] R. A. Félix, *Control Neuronal para Motores Eléctricos*, Tesis de Maestría, CINVESTAV, Campus Guadalajara, México, 2000.

[9] R. Grover and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*, Second edition, Wiley, New York, USA., 1992.

[10] S. Haykin, *Neural Networks: A comprehensive Foundation*, Second edition, Prentice Hall, New Jersey, USA., 1999.

[11] S. Haykin, *Kalman Filtering and Neural Networks*, Wiley, New Jersey, USA., 2001.

[12] S. Haykin, *Adaptive Filter Theory*, Second edition, Prentice Hall, New Jersey, USA., 1991.

[13] F. B. Hildebrand, *Introduction to Numerical Analysis*, Second edition, Dover Publications, Inc., New York, USA, 1987.

[14] ISO California, http://caiso.com

[15] A. H. Jazwinski, *Stochastic Processes and Filtering theory*, Academic Press, New York, 1970.

[16] R. E. Kalman, "A new approach to linear filtering and prediction problems", *Transaction of the ASME, Journal of Basic Engineering*, vol. 82, pp. 35-45, 1960.

[17] H. K. Khalil, *Nonlinear Systems*, Second edition, Prentice Hall, New Jersey, USA., 1996.

[18] E. B. Kosmatopoulus, M. A. Chistodoulou, and P. A. Ioannou, "Dynamical neural networks that ensure exponential identification error convergence", *Neural Networks*, vol. 10, no. 2, pp. 299-314, 1997.

[19] E. B. Kosmatopoulos, M. M. Polycarpou, M. A. Chistodoulou, P. A. Ioannou, "High order neural network structures for identification of dynamical systems", *IEEE Transactions on Neural Networks*, vol. 6, pp. 422-435, 1995

[20] F. L. Lewis, *Optimal estimation*, Wiley, New York, USA, 1986.

[21] L. Ljung, *System Identification: Theory for the User*, Second edition, Prentice Hall, Upper Saddle River, New Jersey, 1999.

[22] M.L. Minisky, S. A. Papert, Perceptrons, Cambridge, MA, MIT Press, 1969.

[23] F. J. Nogales, J. Contreras, A. J. Conejo, and R. Espinola, "Forecasting next-day electricity prices by time series models", *IEEE transactions on Power Systems*, vol. 17, no. 2, pp. 342-348, 2002.

[24] M. Norgaard, N. K. Poulsen, and O. Ravn, "Advances in derivative-free state estimation for nonlinear systems", *Technical Report IMM-REP-1988-15* (revised edition), Technical University of Denmark, 2000.

[25] M. Norgaard, O. Ravn, N. K. Poulsen , L. K. Hansen, *Neural Networks for Modelling and Control of Dynamic Systems*, Springer-Verlag, London, 2000.

[26] OMEL España http://omel.es

[27] D. S. G. Pollock, *A handbook of time-series analysis, signal and dynamics*, Academic Press, San Diego, California, 1999.

[28] M.J.D. Powell, "Radial basis functions for multivariable interpolation: a review", *In proceedings IMA Conference on Algorithms for the approximation of functions and data*, pp. 143-167, Shrivenham, U. K., 1985

[29] G. V. Puskorius, and L. A. Feldkamp, "Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks", *IEEE Transactions on Neural Networks*, vol. 5, No. 2, pp. 279-297, 1994.

[30] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain", *Psychological review*, vol. 65, pp. 396-408,1958.

[31] G. A. Rovitakis and M. A. Chistodoulou, *Adaptive Control with Recurrent High-Order Neural Networks: Theory and Industrial Applications*, Springer-Verlag, London, 2000.

[32] D. W. Ruck, S. K. Rogers, M. Kabrisky, P. S. Maybeck, and M. E. Oxley, "Comparative analysis of backpropagation and the extended Kalman filter for training multilayer perceptrons", *IEEE transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 6, pp. 686-691, 1992.

[33] D. Simon, "Training radial basis neural networks with the extended Kalman filter" *Neurocomputing 1497*, 2001.

[34] P. K. Simpson, *Artificial Neural Systems: Foundation, Paradigms, Applications and Implementations*, Pergamon press, New York, USA., 1990.

[35] S. Singhal and L. Wu, "Training multilayer perceptrons with the extenden Kalman algorithm", in D. S. Touretzky (ed), *Advances in Neural Information Processing Systems 1*, pp. 133-140. San Mateo CA, USA: Morgan Kaufmann, 1989.

[36] J. C. Sprott, G. Rowlands, *CDA user notes*, American Institute of Physics, 1992

[37] C. F. Stevens, "The neuron", *Scientific American*, 241(3), pp. 54-65, 1979.

[38] C. C. Sumila, *Extracción de Patrones Temporales en Base de Datos de Series de Tiempo*, Tesis de Maestría, Universidad Michoacana de San Nicolás de Hidalgo, Facultad de Ingeniería Eléctrica,, Morelia, México, 2001.

[39] E. A. Wan and A. T. Nelson, "Dual extended Kalman filter methods", in S. Haykin (ed), *Kalman Filtering and Neural Networks*, Wiley, New Jersey, USA., 2001.

[40] X. Wang, *"Discrete-time neural networks as dynamical systems"*. Tesis de Doctorado, Departamento de matemáticas, Universidad del Sur de California, California, USA:, 1994.

[41] P. J. Werbos, Beyond regression: New tools for prediction and analysis in the behavioral sciences, Ph.D. Thesis, Harvard University, Cambridge, MA, 1974

[42] R. J. Williams, "Training recurrent networks using the extended Kalman filter", *Proceeding of International Joint Conference on Neural Networks'92*, pp. 241-246, Baltimore, 1992.

[43] R. J. Williams, D. Zipser, "A learning algorithm for continually running fully recurrent neural networks", *Neural Computation 1*, pp. 270-280, 1989.

[44] Y. Zhang and R. Li, "A fast U-D factorization-based learning algorithm with applications to nonlinear system modeling and identification", *IEEE Transactions on Neural Networks*, vol. 10, no. 4,pp. 930-938, 1999.

Apéndice 1.

ESTRUCTURAS DE IDENTIFICACIÓN PARA MODELOS LINEALES.

Considérese un sistema lineal representado por el siguiente modelo:

$$y(t) = G(q^{-1})u(t) + H(q^{-1})e(t)$$

donde G y H son funciones de transferencia en el operador de tiempo q^{-1}, con:

$$q^{-d}x(t) = x(t-d)$$

y *e(t)* es ruido blanco independiente de *u(t)*. Además considérese que se tiene un predictor de varianza mínima con la siguiente estructura:

$$\hat{y}(t|t-1,\theta) = H^{-1}(q^{-1},\theta)G(q^{-1},\theta)u(t) + \left[1 - H^{-1}(q^{-1},\theta)\right]y(t)$$

O escrito en forma recursiva:

$$\hat{y}(t|\theta) = \varphi^{T}(t)\theta$$

donde θ es el vector de parámetros y φ es el vector de regresión, el cual contiene entradas pasadas, salidas pasadas o señales derivadas a partir de entradas y salidas pasadas. A menos que otra cosa sea especificada, en este análisis, solo se considerará la predicción a un paso. Consecuentemente, el condicionante *t-1* será omitido por conveniencia notacional.

La estructura general del modelo se puede rescribir como:

$$A(q^{-1})y(t) = (q^{-d})\frac{B(q^{-1})}{F(q^{-1})}u(t) + \frac{C(q^{-1})}{D(q^{-1})}e(t)$$

donde:

$$A(q^{-1}) = 1 + a_1 q^{-1} + \cdots + a_n q^{-n}$$
$$B(q^{-1}) = b_0 + b_1 q^{-1} + \cdots + b_m q^{-m}$$
$$C(q^{-1}) = 1 + c_1 q^{-1} + \cdots + c_k q^{-k}$$
$$D(q^{-1}) = 1 + d_1 q^{-1} + \cdots + d_l q^{-l}$$
$$F(q^{-1}) = 1 + f_1 q^{-1} + \cdots + f_r q^{-r}$$

A, C, D, E son mónicos

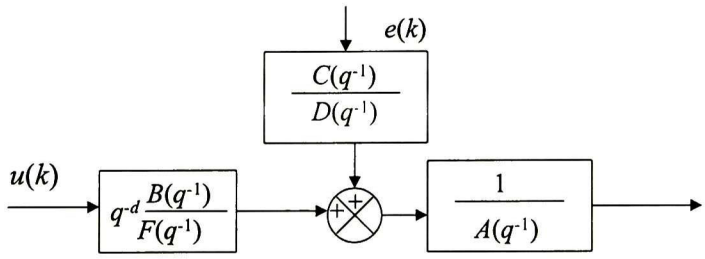


Figura A.1.- Diagrama a bloques para el modelo general.

A1.1 Estructuras del modelo.

Respuesta al impulso finito (FIR). Es el tipo de estructura más simple y corresponde a seleccionar :

$$G(q^{-1}, \theta) = q^{-d} B(q^{-1})$$
$$H(q^{-1}, \theta) = 1$$

en cuyo caso el predictor será:

$$\hat{y}(t|\theta) = q^{-d} B(q^{-1}) u(t)$$

Esto equivale a expresarlo en forma regresiva como:

$$\hat{y}(t|\theta) = \varphi^T(t)\theta$$

con: $\varphi(t) = [u(t-d) \cdots u(t-d-m)]^T$ y $\theta = [b_0 \cdots b_m]^T$

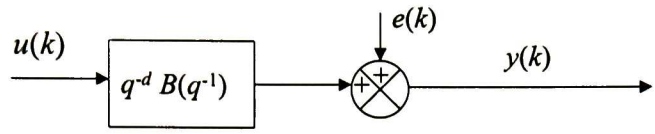


Figura A1.2.- Diagrama a bloques de la estructura *FIR*.

Un sistema con polos, no puede ser descrito exactamente por un modelo *FIR* de orden finito. De cualquier modo si el sistema es estable y la respuesta al impulso decae razonablemente rápido, el sistema podrá ser aproximado por un modelo *FIR* si $B\left(q^{-1}\right)$ es seleccionado como los primeros coeficientes de la respuesta al impulso [25].

ARX (AutoRegressive, eXternal input). La estructura del modelo *ARX* corresponde a seleccionar:

$$G\left(q^{-1},\theta\right)=q^{-d}\frac{B\left(q^{-1}\right)}{A\left(q^{-1}\right)}$$

$$H\left(q^{-1},\theta\right)=\frac{1}{A\left(q^{-1}\right)}$$

Así que el predictor toma la siguiente forma:

$$\hat{y}\left(t|\theta\right)=q^{-d}B\left(q^{-1}\right)u\left(t\right)+\left[1-A\left(q^{-1}\right)\right]y\left(t\right)=\varphi^{T}\left(t\right)\theta$$

con:

$$\varphi\left(t\right)=\left[y\left(t-1\right)\cdots y\left(t-n\right)u\left(t-d\right)\cdots u\left(t-d-m\right)\right]^{T}$$

$$\theta=\left[-a_{1}\cdots-a_{n},b_{0}\cdots b_{m}\right]^{T}$$

La estructura *ARX* también es llamada Controlled AutoRegressive Model (*CAR*), equation error model y Modelo Serie-Paralelo [25].

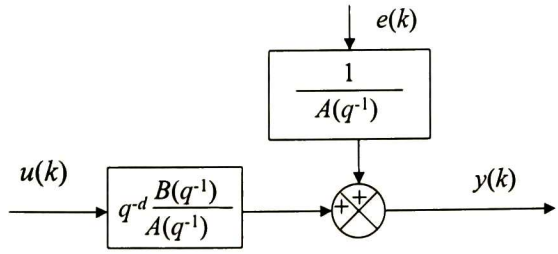


Figura A1.3.- Diagrama a bloques de la estructura *ARX*.

ARMAX (AutoRegressive, Movil Average, eXternal input). Esta estructura siempre es más general que la estructura *ARX*, y corresponde a seleccionar:

$$G\left(q^{-1},\theta\right)=q^{-d}\frac{B\left(q^{-1}\right)}{A\left(q^{-1}\right)}$$

$$H\left(q^{-1},\theta\right)=\frac{C\left(q^{-1}\right)}{A\left(q^{-1}\right)}$$

El predictor es:

$$\hat{y}\left(t|\theta\right)=q^{-d}\frac{B\left(q^{-1}\right)}{C\left(q^{-1}\right)}u\left(t\right)+\left(1-\frac{A\left(q^{-1}\right)}{C\left(q^{-1}\right)}\right)y\left(t\right)$$

$$\hat{y}\left(t|\theta\right)=q^{-d}B\left(q^{-1}\right)u\left(t\right)+\left[1-A\left(q^{-1}\right)\right]y\left(t\right)+\left[C\left(q^{-1}\right)-1\right]\varepsilon\left(t|\theta\right)$$

$$\hat{y}\left(t|\theta\right)=\varphi^{T}\left(t,\theta\right)\theta$$

con:

$\varepsilon\left(t,\theta\right)=y-\hat{y}\left(t|\theta\right)$ que representa el error de predicción y considerando:

$$\varphi\left(t,\theta\right)=\left[y\left(t-1\right)\cdots y\left(t-n\right)u\left(t-d\right)\cdots u\left(t-d-m\right)\varepsilon\left(t,\theta\right)\cdots\varepsilon\left(t-k,\theta\right)\right]^{T}$$

$$\theta=\left[-a_{1}\cdots-a_{n},b_{0}\cdots b_{m},c_{1}\cdots c_{k}\right]^{T}$$

Dada la presencia del polinomial "*C*" el predictor tiene polos. Así que "*C*" debe tener sus raíces dentro del circulo unitario. Los polos también implican que el vector de regresión depende de los parámetros del modelo, lo cual hace más complicada la estimación [25].

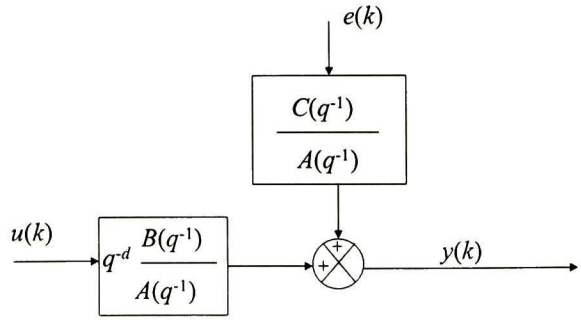


Figura A1.4.- Diagrama a bloques de la estructura *ARMAX*.

OE (Output Error). El modelo *OE* o paralelo se usa cuando el error que afecta a la planta es solo ruido blanco de medición.

$$y(t) = q^{-d}\frac{B(q^{-1})}{F(q^{-1})}u(t) + e(t)$$

Correspondientemente:

$$G(q^{-1},\theta) = q^{-d}\frac{B(q^{-1})}{F(q^{-1})}$$

$$H(q^{-1},\theta) = 1$$

El predictor de este sistema es:

$$\hat{y}(t|\theta) = q^{-d}\frac{B(q^{-1})}{F(q^{-1})}u(t)$$

$$\hat{y}(t|\theta) = q^{-d}B(q^{-1})u(t) + \left[1 - F(q^{-1})\right]\hat{y}(t|\theta) = \varphi^T(t,\theta)\theta$$

donde:

$$\varphi(t,\theta) = \left[\hat{y}(t-1|\theta)\cdots\hat{y}(t-r|\theta), u(t-d)\cdots u(t-d-m)\right]$$

$$\theta = \left[-f_1\cdots - f_r, b_0\cdots b_m\right]^T$$

Para que el predictor sea estable, las raíces de "*F*" deben estar dentro del circulo unitario [25].

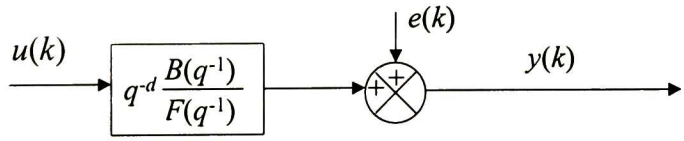


Figura A1.5.- Diagrama a bloques de la estructura *OE*.

Apéndice 2.

OTROS RESULTADOS.

En este apéndice se presentan otros resultados que aún y cuando no son el objetivo inicial de este trabajo, pueden resultar de interés para otras aplicaciones y están basados en el trabajo reportado en esta tesis.

A2.1 Predicción a un paso utilizando una estructura *NNARX*.

Para esta aplicación se utilizó una red neuronal tipo *MLP* en una configuración *NNARX* como la mostrada en la figura 4.2. Como caso de estudio se utilizó la serie de tiempo de la demanda eléctrica mostrada en la figura 4.5. Se utilizaron 10 neuronas en la capa oculta, con función de activación tipo sigmoidal y una en la capa de salida con función de activación lineal, el vector de regresión utilizado es el que se muestra a continuación:

$$\varphi = \left[y(k-1) \quad y(k-1) \quad \cdots \quad y(k-1) \right]^T$$

Se utilizan siete retardos de la salida medida, ya que esté es el orden del sistema. El entrenamiento se realizó fuera de línea. En la figura A2.1 se presenta la estructura de la red neuronal utilizada y los resultados obtenidos, al entrenar dicha red con el algoritmo del *FKE,* se muestran en la figura A2.2.

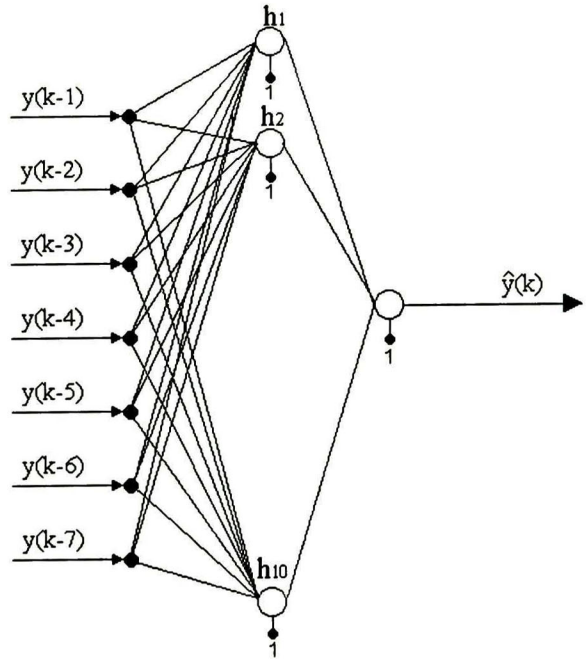


Figura A2.1.- Estructura de la red neuronal utilizada.

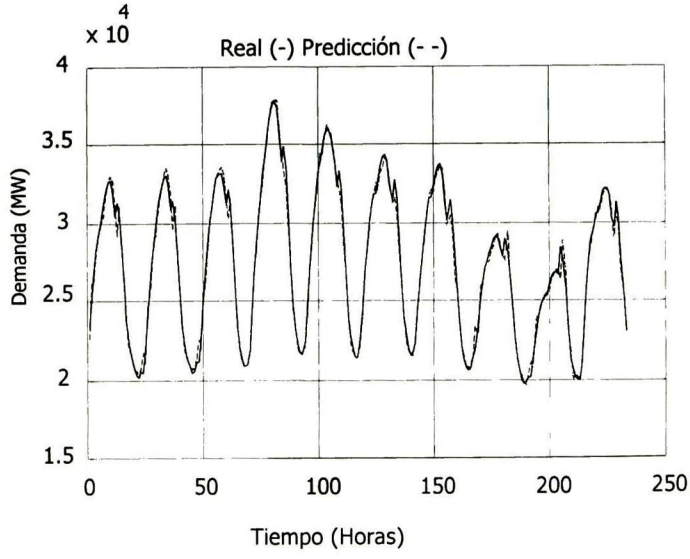


Figura A2.2.- Predicción a un paso con una estructura *NNARX* entrenada con el *FKE*.

A2.2 Resultados obtenidos con el algoritmo de *FKELD* con dos capas ocultas.

Para esta aplicación se utilizó una red neuronal tipo *MLP* con una estructura *NNOE,* como la mostrada en la figura (A2.3), con 15 neuronas en la primer capa oculta, 5 en la segunda capa oculta y una en la capa de salida; las neuronas de la capa oculta tiene una función de activación tipo sigmoidal, mientras que la de salida tiene una función de activación lineal. El vector de regresión utilizado fue el presentado en la ecuación (4.14). Los resultados de entrenar la red neuronal con el algoritmo del *FKELD* se presentan el la figura A2.4.

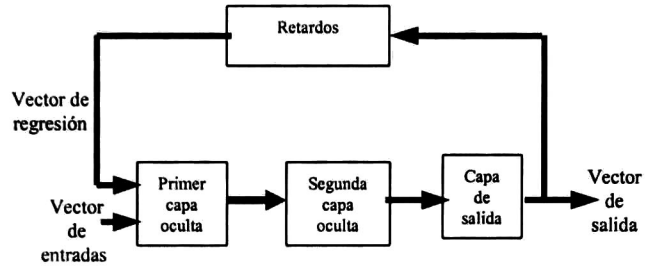


Figura A2.3.- Estructura de la red neuronal utilizada.

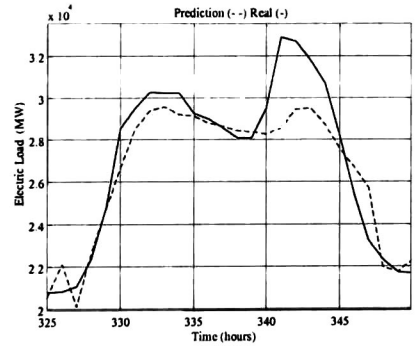


Figura A2.4.- Comparación de la serie de tiempo real y la predicción de la demanda eléctrica para las primeras 24 horas..

Apéndice 3

ARTÍCULOS PUBLICADOS.

Los artículos que a continuación se presentan, están fundamentados en esta tesis.

- Predicción de la demanda eléctrica usando redes neuronales recurrentes entrenadas por filtro de Kalman (*CLCA'04* La Habana Cuba Mayo de 2004)

- Electric Load Demand Prediction using Neural Networks Trained by Kalman Filtering (*UICN* 2004 Budapest Hungria Julio de 2004)

- Recurrent Neural Networks Trained with Kalman Filtering for Discrete Chaos Reconstruction (*APWCCS'04* Melbourne Australia Julio de 2004)

AUTO45.DOC

Estimado colega:

Por medio de la presente le comunico que su trabajo ha sido aceptado para su presentación en el XI Congreso Latinoamericano de Control Automático. Le adjuntamos los resultados de la evaluación.
Le recordamos que el 31 de marzo es la fecha tope para enviar la versión final del trabajo en la cual deben tener en cuenta las sugerencias hechas por los revisores. Si tiene alguna dificultad, por favor, no dude en contactarnos.
En la página www.informaticahabana.com o en nuestra página www.cujae.edu.cu\clca usted podrá obtener todas las informaciones relacionadas con los hospedajes y las agencias de viaje con las que puede contactar para organizar su viaje al Congreso.
Espero que nos veamos en La Habana.
Por favor, confirme la recepción de este mensaje

Atentamente

Orestes

****
Dr. Orestes Llanes-Santiago
Decano
Facultad Ingeniería Eléctrica, ISPJAE
Telf: 53-7-2606778
53-7-2671457
e-mail: orestes@electrica.cujae.edu.cu
****

Aniversario 40 de la CUJAE

Visite:

XII Convencion de Ingenieria y Arquitectura
http://www.cujae.edu.cu/eventos/convencion/

X Confe Internacional y Feria

Informática 2004

Del 10 al 15 de mayo del 2004, La Habana, Palac[illegible] de las Convenciones

Carta de Bienvenida
Información General
Programa General
Programa por Eventos
Auspiciadores
Personalidades
Comité Organizador
Requerimientos Técnicos
Software disponibles
Créditos

Buscar

Eventos

GKT s.a.

citmatel

ISBN: 959-237-117-2

Carta de Bienvenida
Información General
Programa General
Eventos
Auspiciadores
Personalidades
Comité Organizador
Requerimientos Técnicos
Software disponibles
Créditos
Buscar

Programa por Eventos

anterior ...Evento... siguiente

"NEURAL BLOCK CONTROL WITH CONSTRAINED ADAPTIVE PARAMETERS"
RAMON A. FÉLIX
CINVESTAV
AUT080 (Res)
MÉXICO

"IDENTIFICACIÓN DE UN SISTEMA ELECTRO-HIDRÁULICO VÍA REDES NEURONALES RECURRENTES"
EDGAR N. SÁNCHEZ
CINVESTAV
AUT054 (Res)
MÉXICO

"ESTIMADOR DE HORIZONTE MÓVIL NEURONAL EN MOLIENDA SEMIAUTÓGENA"
KARINA CARVAJAL
UNIVERSIDAD DE SANTIAGO DE CHILE
AUT024 (Res)
CHILE

"PREDICCIÓN DE LA DEMANDA ELÉCTRICA USANDO REDES NEURONALES ENTRENADAS POR FILTRO DE KALMAN"
EDGAR N. SÁNCHEZ
CINVESTAV
AUT045 (Res)
MÉXICO

"APRENDIZAGEM AUTÔNOMA DO GERENCIAMENTO DE CONFLITOS EM TRÁFEGO AÉREO"
RICARDO M. MAIA

ISBN: 959-237-117-2

PREDICCIÓN DE LA DEMANDA ELÉCTRICA USANDO REDES NEURONALES ENTRENADAS POR FILTRO DE KALMAN.

Edgar N. Sanchez Alma Y. Alanis Jesus Rico

CINVESTAV, Unidad Guadalajara, Apartado Postal 31-430, Plaza La Luna, Guadalajara, Jalisco C.P. 45091, México, e-mail: sanchez@gdl.cinvestav.mx
Universidad Michoacana de San Nicolas de Hidalgo, Morelia, Michoacan, México.

Resumen: En este artículo se presentan los resultados obtenidos al utilizar redes neuronales artificiales tipo perceptrón multicapa recurrente, para la predicción de la demanda eléctrica. Se usan algoritmos de entrenamiento basados en el Filtro de Kalman Extendido. El objetivo es lograr la predicción a 24 horas de la curva de la demanda eléctrica, utilizándose como caso de estudio, datos de la demanda eléctrica del estado de California, USA.

Palabras clave: Filtro de Kalman, Redes Neuronales Artificiales, Perceptrón Multicapa, Predicción.

1. INTRODUCCIÓN

Un sistema de energía eléctrica debe abastecer a todos los puntos de carga con buena calidad de servicio, lo cual implica continuidad en el servicio, eficiencia, costos mínimos, etc. Dichas características se pueden obtener teniendo un sistema adecuado de predicción, que permita adoptar estrategias para enfrentar condiciones futuras. El objetivo de este trabajo es lograr la predicción de la demanda eléctrica con una anticipación de 24 horas. La predicción a corto plazo sirve para: planeación del despacho económico, análisis costo beneficio de los programas de gestión de cargas, planes de compra de combustibles y en algunos casos el establecimiento del precio de compra de energía eléctrica (Sumila, 2001).

La literatura es muy amplia sobre las habilidades de las redes neuronales, tipo perceptrón multicapa *(MLP)*, para reproducir relaciónes no lineales en conjuntos de datos. Existen diversos algoritmos de entrenamiento para este tipo de redes neuronales; sin embargo presentan diversos problemas, como son: convergencia a mínimos locales, aprendizaje lento, alta sensibilidad a los valores iniciales, entre otros. Por lo tanto se han propuesto diversos algoritmos de entrenamiento, basados en el Filtro de Kalman (Grover, *et al.*, 1992; Haykin, 2001; Singhal, *et al.*,1989). Como para relaciones no lineales, no es posible aplicar directamente el Filtro de Kalman, se hace necesario utilizar el Filtro de Kalman Extendido *(EKF)*. está técnica ha recibido bastante atención en recientes años. La sugerencia propuesta es que el entrenamiento de *MLP* sea interpretado como un problema de estimación para un sistema dinámico no lineal, el cual pueda ser resuelto con el *EKF* (Haykin, 2001).

Las redes neuronales recurrentes poseen un rico repertorio de arquitecturas, lo cual las habilita para realizar diversas aplicaciones (Haykin, 1999), que no son posibles con las redes neuronales estáticas; algunas de estas aplicaciones son: predicción no lineal, modelado, control, representaciones en

espacio de estado, etc. Debido a los beneficios que proveen las redes neuronales recurrentes, en este trabajo sugerimos el uso de redes neuronales tipo perceptrón multicapa recurrentes *(RMLP)*, entrenadas con un algoritmo basado en el *EKF*.

El uso de un algoritmo de entrenamiento basado en el *EKF* involucra el cálculo de derivadas, las cuales en algunos casos no son fáciles de obtener; por ello en (Feldkamp, *et al.*, 2001; Norgaard, *et al.*, 2000), se propone un tratamiento alternativo, para evitar el uso de derivadas. Existen formulas de interpolación que no necesitan derivadas, sólo evaluación de funciones simplificando así su implementación. De esta idea surge el Filtro de Kalman Libre de Derivadas *(EKFDF)*, propuesto en: (Feldkamp, *et al.*, 2001; Norgaard, *et al.*, 2000).

Este artículo se organiza de la manera siguiente. Primero se introducen el *EKF* y el *EKFDF* como algoritmos de entrenamiento para redes neuronales. A continuación se describe la implementación de ambos algoritmos, aplicados en la predicción de la curva de la demanda eléctrica y finalmente se establecen las comparaciones pertinentes entre los algoritmos mencionados, para el caso de estudio propuesto.

2. FORMULACIÓN DEL *EKF* COMO ALGORITMO DE ENTRENAMIENTO DE REDES NEURONALES RECURRENTES.

Esta sección se basa en (Haykin, 1999) y (Haykin, 2001). Básicamente, el Filtro de Kalman intenta estimar el estado de un sistema lineal perturbado por ruido blanco Gaussiano y donde las mediciones disponibles son combinaciones lineales de los estados del sistema perturbados a su vez por ruido blanco Gaussiano. En la aplicación del Filtro de Kalman al entrenamiento de redes neuronales, los pesos del perceptrón multicapa son los estados que el Filtro de Kalman estima, y la salida de la red es la medición usada por el Filtro de Kalman. Como el *MLP* es un sistema no lineal, es necesario utilizar el *EKF*.

Es por ello que surge la alternativa de visualizar el entrenamiento de redes neuronales como un problema de filtrado óptimo, la solución del cual sea recursiva, utilice la información actual y no necesite almacenar toda la evolución de los pesos; ésta es la esencia del Filtro de Kalman aplicado al entrenamiento de redes neuronales.

La estimación se realiza recursivamente; esto es, cada actualización del estimado (pesos sinápticos) es calculada a partir del estimado anterior y de los datos actuales; así sólo el estimado anterior requiere ser almacenado.

El entrenamiento se realiza a partir de un conjunto de N mediciones entrada-salida. El objetivo es encontrar los pesos óptimos que minimicen el error de predicción. Las ecuaciones necesarias para el desarrollo del algoritmo basado en el *EKF* son:

$$K(k) = P(k) H(k) \left[R(k) + H^T(k) P(k) H(k)\right]^{-1} \quad (1)$$

$$x(k+1) = x(k) + K(k) \left[y(k) - \hat{y}(k)\right] \quad (2)$$

$$P(k+1) = P(k) - K(k) H^T(k) P(k) + Q(k) \quad (3)$$

donde $P(k) \in \Re^{LxL}$ y $P(k+1) \in \Re^{LxL}$ representan la matriz de covarianza del error de predicción al tiempo k y $k+1$ respectivamente. $x \in \Re^L$ es el vector de pesos (estados), L es el número total de pesos de la red neuronal; $y \in \Re^m$ es el vector de salidas, donde m es el número total de salidas medidas; $\hat{y} \in \Re^m$ es la salida de la red neuronal; $K \in \Re^{Lxm}$ es la matriz de la ganancia de Kalman; $Q \in \Re^{LxL}$ es la matriz de covarianza del ruido del proceso; y $R \in \Re^{mxm}$ es la matriz de covarianza del ruido de medición. $H \in \Re^{Lxm}$ es la matriz que contiene las derivadas de cada salida de la red neuronal ($\hat{y}_i$), con respecto a cada uno de los pesos (x_j), tal como se define en (4).

$$H_{ij}(k) = \left[\frac{\partial \hat{y}_i(k)}{\partial x_j(k)}\right]_{x(k)=\hat{x}(k+1)} \qquad i = 1...m, \; j = 1...L \quad (4)$$

Usualmente las matrices P, Q y R son inicializadas como matrices diagonales, cuyos elementos de la diagonal son P_0, Q_0 y R_0 respectivamente.

La forma en que se obtiene H en (4) es uno de los principales problemas que se enfrenta al realizar la programación del algoritmo de entrenamiento basado en el Filtro de Kalman; para tratar de subsanar esto, en (Feldkamp, *et al.*, 2001) y (Norgaard, *et al.*, 2000), proponen un método alternativo, el Filtro de Kalman Extendido Libre de Derivadas *EKFDF* el cual se presenta en la siguiente sección.

3. FORMULACIÓN DEL *EKFDF* COMO ALGORITMO DE ENTRENAMIENTO DE REDES NEURONALES RECURRENTES.

Esta sección se basa en los resultados de (Feldkamp, *et al.*, 2001) y (Norgaard, *et al.*, 2000). El *EKFDF* se obtiene al remplazar una expansión en series de Taylor en la vecindad de un vector de pesos, por una expansión basada en la formula de Stirling. Sean los operadores Δ y δ aplicados a f con h denotando la longitud del intervalo seleccionado.

$$\delta f(x) = f\left(x + \frac{h}{2}\right) - f\left(x - \frac{h}{2}\right) \quad (5)$$
$$\mu f(x) = \frac{1}{2} f\left(x + \frac{h}{2}\right) - f\left(x - \frac{h}{2}\right)$$

Con estos operadores, la formula de interpolación de Stirling (Norgaard, *et al.*, 2000), en $x = \overline{x}$, puede expresarse como:

$$f(x) = f(\overline{x} + ph) = f(\overline{x}) + p\mu\delta f(\overline{x}) + \frac{p^2}{2!}\delta^2 f(\overline{x})$$

$$+ \binom{p+1}{3} \mu\delta^3 f(\overline{x}) + \frac{p^2(p^2-1)}{4!}\delta^4 f(\overline{x}) \quad (6)$$

$$\binom{p+2}{5} \mu\delta^5 f(\overline{x}) \quad \ldots$$

Para este caso en particular sólo se consideran aproximaciones polinomiales de primer y segundo orden.

$$f(x) \approx f(\overline{x}) + f_{DD}(\overline{x})[x - \overline{x}] + \frac{f_{DD}(\overline{x})}{2!}[x - \overline{x}]^2 \quad (7)$$

Donde:

$$f_{DD}(\overline{x}) = \frac{f(\overline{x} + h) - f(\overline{x} - h)}{2h} \quad (8)$$
$$f_{DD}(\overline{x}) = \frac{f(\overline{x} + h) + f(\overline{x} - h) - 2f(\overline{x})}{h^2}$$

Nótese que h define el intervalo $[\overline{x} - h, \overline{x} + h]$ para el cual la aproximación se satisface.

Ahora se explicara brevemente la aplicación de la formula de interpolación de Stirling en el entrenamiento de redes neuronales, bajo la suposición de que el ruido de medición es blanco, Gaussiano y aditivo (Feldkamp, *et al.*, 2001).

En primer lugar se definirán algunas matrices necesarias para la realización del algoritmo de entrenamiento basado en el Filtro de Kalman Libre de Derivadas. Al igual que en algoritmo basado en el *EKF* el vector de pesos será denotado por x y es de dimensión L. $\overline{S}_x(k)$ está definida por una descomposición de Cholesky de P como se muestra en (9), $\overline{S}_x(k)$ es una matriz cuadrada de dimensión L. $P(k)$ se define igual que en el *EKF*. La i-ésima columna de $\overline{S}_x(k)$ es denotada por $s_{x,i}(k)$. Por simplicidad se omitirá k.

$$P = \overline{S}_x \overline{S}_x^T \quad (9)$$

Para cada i-ésima columna correspondiente se forman dos variaciones del vector x: $x + hs_{x,i}$ y $x - hs_{x,i}$ Para cada una de estas variaciones se debe calcular la salida correspondiente. La j-ésima salida de la red neuronal para los pesos nominales es definida como: $g_j(x)$, a la variación de $x + hs_{x,i}$ le corresponde $g_j(x + hs_{x,i})$, equivalentemente $g_j(x - hs_{x,i})$ representa la j-ésima salida de la red neuronal para el vector de pesos $x - hs_{x,i}$.

La matriz $S_{xv}^{(1)} \in \Re^{LxL}$ y la matriz $S_{yw}^{(1)} \in \Re^{mxm}$ son matrices diagonales , cuyos elementos no cero son iguales a Q_0^- y R_0^- respectivamente.

Los elementos de la matriz $S_{yx}^{(1)} \in \Re^{mxL}$ están definidos por (10). Finalmente la matriz $S_{yx}^{(2)}$ de la misma dimensión que $S_{yx}^{(1)}$ y sus elementos están definidos por (11).

$$\left(S_{yx}^{(1)}\right)_{ij} = \frac{1}{2h}\left(g_j(x + hs_{x,i}) - g_j(x - hs_{x,i})\right) \quad (10)$$

$$\left(S_{yx}^{(2)}\right)_{ij} = \frac{(h^2 - 1)^-}{2h^2}\left(g_j(x + hs_{x,i}) + g_j(x - hs_{x,i}) - 2g_j(x)\right) \quad (11)$$

S_y es una matriz compuesta de la siguiente forma:

$$S_y = \left[S_{yx}^{(1)} \; S_{yw}^{(1)} \; S_{yx}^{(2)} \right] \quad (12)$$

Una vez definidas estas matrices se procederá a establecer las ecuaciones del Filtro de Kalman Extendido Libre de Derivadas. Sea la matriz $P_y = S_y S_y^T$, la cual es análoga a la matriz $[R + H^T P H]$ del Filtro de Kalman Extendido. La matriz definida en (13), equivale a la matriz originada por el producto PH en el *EKF*. Por lo tanto la Matriz de la Ganancia del filtro de Kalman está dada por (14).

$$P_{xy} = \overline{S}_x S_{yx}^{(1)T} \quad (13)$$

$$K = P_{xy} P_y^{\ 1} \quad (14)$$

La actualización de los pesos está dada por (15), la j-ésima columna de $\overline{y}$ al tiempo k está definida por (16); por último la actualización de la matriz de covarianza se define por (17):

$$x(k+1) = x(k) + K(k)[y(k) - \overline{y}(k)] \quad (15)$$

$$\overline{y}_j = \frac{h^2 - L}{h^2} g_j(x) + \frac{1}{2h^2}\sum_{i=1}^{L} g_j(x + hs_{x,i}) + g_j(x - hs_{x,i}) \quad (16)$$

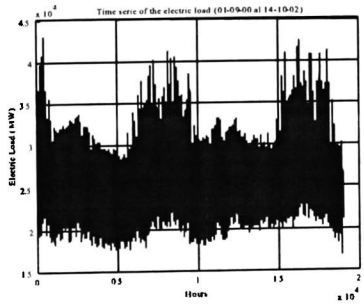


Fig. 1. Sistema eléctrico de California.

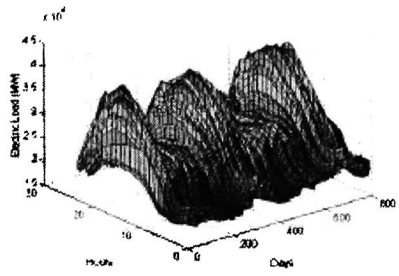


Fig. 2. Comportamiento diario de la demanda eléctrica.

$$P(k+1) = P(k) - K(k) P_y(k) K^T(k) + Q(k) \tag{17}$$

ACCIÓN	*EKF*	*EKFDF*
Inicialización	x_0, P_0, Q_0, R_0	x_0, P_0, Q_0, R_0
H	(4)	
S_y		(10), (11), (12)
Salida	$\widehat{y}$	(16)
$K(k+1)$	(1)	(14)
$w(k+1)$	(2)	(15)
$P(k+1)$	(3)	(17)

Tabla 1. Resumen de las ecuaciones del *EKF* y el *EKFDF*.

4. PREDICCIÓN DE LA DEMANDA ELÉCTRICA USANDO *RMLP*.

La aplicación especifica del presente artículo es la utilización de los algoritmos presentados en las secciones previas, para la predicción de 24 horas de la curva de la demanda eléctrica; para esto se utilizó la base de datos del California ISO (caiso, 2002). En la figura 1 se pueden observar dichos datos; de igual manera en las figuras 3 y 4, se muestran un día típico y una semana típica, respectivamente, de la demanda eléctrica del sistema mencionado.

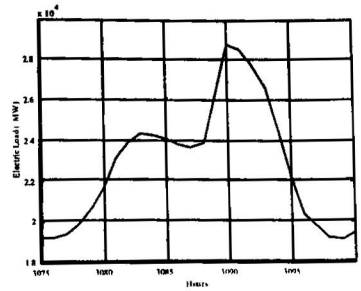


Fig. 3. Ejemplo de un dia típico de la curva de la demanda eléctrica.

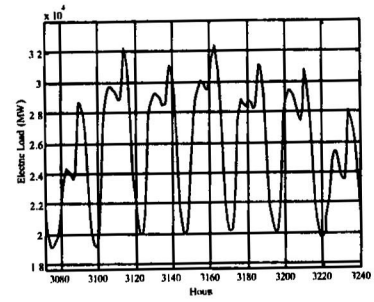


Fig. 4. Ejemplo de una semana típica de la demanda eléctrica.

4.1 Aplicación del algoritmo EKF.

La red utilizada para esta aplicación es una red tipo *MLP* y la configuración de la misma es como se muestra en la figura 5. Se utilizarón 15 neuronas en la capa oculta y una neurona en la capa de salida; para las neuronas de la capa oculta se uso como función de activación una del tipo logística, y para la neurona de la capa de salida una función tipo lineal. Los valores iniciales para las matrices de covarianza son: $R_0 = Q_0 = P_0 = 10000$; el entrenamiento duro 35 iteraciones, con el cual se alcanzó el criterio de error deseado.

La dimensión del vector de regresión es de 7 debido a que esté es el orden del sistema, el mismo que se obtuvo con un algoritmo basado en los coeficientes de Lipschitz.Ademas se agregaron dos datos externos $u_1(k)$ y $u_2(k)$, los cuales corresponden al dia y la hora respectivos. El entrenamiento se realizó fuera de línea, y se utilizó una configuración tipo *serie-paralelo*, entre el sistema real y la red neuronal. Una vez que el Filtro de Kalman logró obtener los pesos óptimos, se congelaron dichos pesos, y se utilizó una configuración en *paralelo* entre el sistema real y el sistema modelado por la red neuronal.

Los resultados obtenidos, al utilizar el algoritmo de entrenamiento basado en el EKF, se muestran en las figuras 6, 7 y 8. En éstas, se observa la predicción de una semana y el detalle de un día,

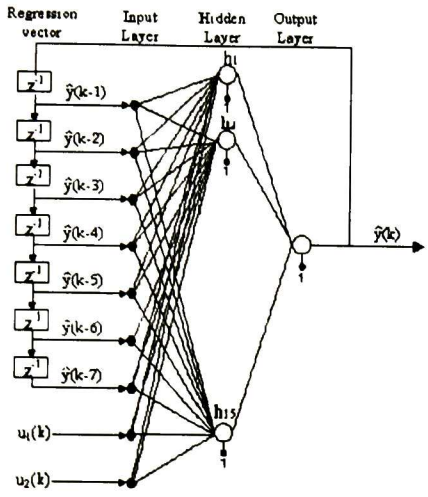


Fig. 5. Estructura de la red neuronal utilizada.

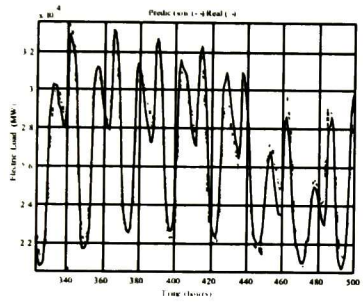


Fig. 6. Comparación de la demanda eléctrica real y la predicción durante una semana.

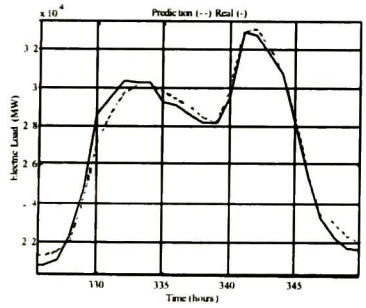


Fig. 7. Comparación de la demanda eléctrica real y la preducción para las primeras 24 horas.

además de la evolución del error medio cuadrático, durante el entrenamiento de la red neuronal.

4.2 Aplicación del algoritmo EKFDF.

En esta sección se presentan los resultados obtenidos con este algoritmo aplicado a la predicción de la curva de la demanda eléctrica. Para esta aplicación se considero una red tipo *RMLP* en una

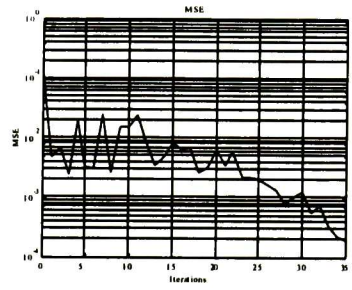


Fig. 8. Evolución del error medio cuadratico durante el entrenamiento de la red neuronal.

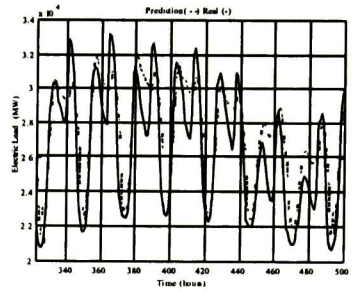


Fig. 9. Comparación de la demanda eléctrica real y la predicción durante una semana.

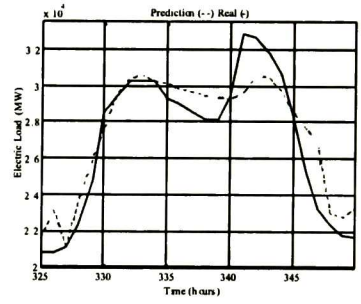


Fig. 10. Comparación de la demanda eléctrica real y la predicción para las primeras 24 horas.

configuración como la que se muestra en la figura 5, es una red recurrente con una capa oculta. Con el método propuesto por (Feldkamp, *et al.*, 2001) y (Norgaard, *et al.*, 2000), el algoritmo resulta muy fácil de programar independientemente del número de capas ocultas en la red neuronal.

Las neuronas de la capa oculta tienen una función de activación tipo logística, mientras que la neurona de la capa de salida tiene una función de activación lineal. El entrenamiento se realizó fuera de línea en una configuración *serie-paralelo*, entre el sistema real y la red neuronal, una vez obtenidos los pesos óptimos, estos se congelan y en una configuración *paralelo* se realizó la predicción con los resultados que se muestran en las figuras 9, 10 y 11.

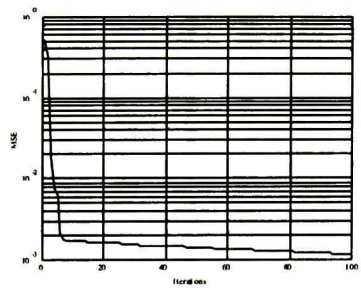


Fig. 11. Evolución del error medio cuadratico durante el entrenamiento de la red neuronal.

En esta aplicación se utilizarón 15 neuronas en la capa oculta y una en la neurona de salida. Los valores iniciales de las matrices de covarianza se tomaron como: $P_0 = 5e-6$, $Q_0 = 5e-10$ y $R_0 = 5e-6$. Se realizó un entrenamiento de 100 iteraciones. Este algoritmo presenta un parámetro de diseño extra, h , que representa la longitud del intervalo donde la aproximación se satisface, h se selecciono igual a $\sqrt{3}$, como se recomienda en (Feldkamp, *et al.*, 2001) y (Norgaard, *et al.*, 2000).

4.3 Comparaciones.

Comparando el algoritmo del *EKF* con el algoritmo de entrenamiento basado en el *EKFDF*, este último requiere mayor número de iteraciones y no iguala la precisión que se obtiene con el algoritmo del *EKF*; el tiempo por iteración de ambos algoritmos es muy similar.

En cuanto a la dificultad de programación, el algoritmo basado en el *EKFDF* es mucho más sencillo de programar que el algoritmo basado en el *EKF*, independientemente de que tan complicada resulte la estructura interna de la red neuronal.

Aunque aqui no se presentan los resultados, también se realizaron comparaciones entre el algoritmo de entrenamiento de *Levenberg-Marquardt* y el algoritmo de entrenamiento basado en el *EKF*; obteniendose mejores resultados para este último.

5. CONCLUSIONES.

En este artículo se han propuesto dos algoritmos para el entrenamiento de redes neuronales; ambos algoritmos se aplicaron al caso de estudio propuesto, con los resultados discutidos en la sección anterior. Los algoritmos aquí propuestos, presentan las siguientes ventajas, sobre otros algoritmos de entrenamiento de redes neuronales: No son sensibles a los pesos iniciales; pueden ser usados satisfactoriamente para el modelado e identificación de sistemas altamente no lineales; y no se restringen a redes neuronales de una topologia especifica, entre otros.

El algoritmo basado en el *EKFDF* no presentó tan buenos resultados como el basado en el *EKF*, sin embargo su principal atractivo es que posee una mínima complejidad de programación independientemente de la estructura interna de la red neuronal; otro aspecto que destaca al *EKFDF* del *EKF* es el no necesitar el cálculo de derivadas.

En conclusión el algoritmo de entrenamiento basado en el *EKF* es el algoritmo que presenta mejores resultados, mostrando una excelente habilidad de generalización y necesitando menos iteraciones, para lograr la exactitud deseada.

Reconocimientos. Los autores agraceden el apoyo del proyecto CONACYT México No. 39866Y.

Bibliografía.

Feldkamp, L. A., Feldkamp, T. M., and Prokhorov D. V. (2001). Neural network training with the nprKF. *Proceedings of International Joint Conference on Neural Networks'01*, 109-114.

Grover, R. and Hwang, P. Y.C. (1992). *Introduction to random signals and applied kalman filtering*, Second edition, Wiley, New York, USA.

Haykin, S. (1999). *Neural Networks. A comprehensive foundation*, Second edition, Prentice Hall, New Jersey, USA.

Haykin, S. (2001). *Kalman Filtering and Neural Networks*, Wiley, New York, USA.

http://caiso.com.(2002).

Norgaard, M., Poulsen, N. K. and Ravn, O. (2000). Advances in Derivative-Free State Estimation for Nonlinear Systems, *Technical Report IMM-REP-1988-15* (revised edition). Technical University of Denmark.

Singhal, S. and Wu, L. (1989). Training multilayer perceptrons with the extended Kalman algorithm, in D. S. Touretzky (ed), *Advances in Neural Information Processing Systems 1* (pp. 133-140). San Mateo, CA, USA: Morgan Kaufmann.

Sumila, C. (2001). *Extracción de patrones temporales en base de datos de series de tiempo.* Universidad Michoacana de San Nicolás de Hidalgo, Facultad de Ingeniería Eléctrica, Morelia. Michoacan, México.

2004 International Joint Conference on Neural Networks
July 25-29, 2004

IJCNN 2004: Paper #1582 Resubmission

Dear Author(s),

Congratulations! On behalf of the IJCNN 2004 Technical Program Committee and the program chairs, we are pleased to inform you that your paper:

Paper ID: 1582
Author(s): Edgar Sanchez, Alma Alanis and Jesus Rico
Title: Electric Load Demand Prediction using Neural Networks Trained by Kalman Filtering

has been accepted for presentation at the 2004 International Joint Conference on Neural Networks and for publication in the conference proceedings published by IEEE. This email provides you with all the information you require to complete your paper and submit it for inclusion in the proceedings. A notification of the timing and the format of your presentation (oral or poster) will follow by the end of March.

Electric Load Demand Prediction using Neural Networks Trained by Kalman Filtering

Edgar N. Sanchez[1], Alma Y. Alanis[1], Jesus Rico[2]
[1]CINVESTAV, Unidad Guadalajara, Apartado Postal 31-438, Plaza La Luna, Guadalajara, Jalisco, C.P. 45091, Mexico, e-mail:sanchez@gdl.cinvestav.mx.
[2]Universidad Michoacana de San Nicolas de Hidalgo, Morelia, Michoacan, Mexico.

Abstract **This paper presents the application of recurrent multilayer perceptron neural netwoks to electric load demad prediction; the respective training is performed extended Kalman ltering. The goal is to obtain a 24 hours horizon. prediction for the electric load demand; data from the state of California, USA, is utilized.**

Keywords **- Recurrent Neural Networks, Multilayer Perceptron, Kalman Filtering, Prediction.**

I. Introduction

An electric power system should provide energy to every load with good quality, which implies reliability, efficiency and minimal costs. For this kind of service, it is required to implement an adequate prediction tool, which will allow to take decisions in advance in order to ful ll future conditions. Along these requirements, the goal of the present paper is to develop such a 24 hours predictor for the electric load demand. This short period prediction will aid to economic dispatch planning, cost-pro t analysis, oil buying planning and electricity pricing, among others [1].

There are many publications about the capabilities of multilayer perceptron neural networks *(MLP)* to determine nonlinear relations from data. There exist different training algorithms for this kind of neural network application; however they are subjected to different problems, such as: local minima, slow learning, high sensitivity to initial conditions, among others. As a viable alternative, new training algorithms, based on Kalman ltering have been proposed. [6], [8],[9]. Due to the fact that training of MLP results on a nonlinear problem, it is required to use Extended Kalman Filtering *(EKF)*, as discussed on [8]. *EKF* requires to calculate some derivatives, which could not be easy; in [4], [5], as an alternative, the so-called Extended Kalman Filtering, Derivative Free (*EKFDF*) algorithm is developed.

There exist many structures for recurrent neural networks, which allow them to be applied for different tasks such as: nonlinear modelling, nonlinear prediction and nonlinear control [7], with better results than the ones obtained with feedforward neural networks.

Take into account all the above facts, in this work, the authors propose to use Recurrent Multilayer Perceptron Neural Networks *(RMLP)*, trained by means of *EKF* and *EKFDF* algorithms, in order to implement a 24 hours predictor for the electric load demand. There are many publications about one-hour ahead load prediction using neural networks [2], [10], however in this paper the goal is to obtain a 24 hours horizon.

The outline of the paper is as follows: For the sake of completeness, rst the *EKF* and *EKFDF* neural network training algorithms are brie y discussed. Then, their application to the above mentioned predictor is presented, including the respective comparison. Finally, some relevant conclusions are established.

II. EKF Training Algorithm

This section is mainly based on [7] and [8]. Kalman ltering estimates the state of a linear system, with additive state and output white noise. For Kalman ltering neural network training, the neural network weights become the states to be estimated, the output is the one of the neural network, with the error between the neural network and the desired output considered as and additive white noise. Due to the fact that the neural network mapping is nonlinear, an *EKF* type algorithm is required.

The neural network training is performed using a set of N input-output measurement pairs. The training goal is to nd the optimal weight values, which minimize the prediction error (the difference between the measured output and the neural network one). The *EKF* training algorithm is based on the following equations

$$K(k) = P(k) H(k) \left[R(k) + H^T(k) P(k) H(k)\right]^{-1} \quad (1)$$

$$x(k+1) = x(k) + K(k) [y(k) - \hat{y}(k)] \quad (2)$$

$$P(k+1) = P(k) - K(k) H^T(k) P(k) + Q(k) \quad (3)$$

where $P(k) \in \Re^{LxL}$ and $P(k+1) \in \Re^{LxL}$ stand for the prediction error covariance matrices at k and $k+1$ respectively; $x \in \Re^L$ is the weight (state) vector, L is the total number of neural network weights; $y \in \Re^m$ is the measured output vector; $\hat{y} \in \Re^m$ is the neural network output; m is the total number of outputs; $K \in \Re^{Lxm}$ is the Kalman gain matrix; $Q \in \Re^{LxL}$ is the state noise covariance matrix; and $R \in \Re^{mxm}$ is the measurement noise covariance matrix. $H \in \Re^{Lxm}$ is a matrix, which each entry (H_{ij}) is the derivative of one of the neural network output $(\hat{y}_i)$ with respect to one neural network weight (x_j), as follows

$$H_{ij}(k) = \left[\frac{\partial \hat{y}_i(k)}{\partial x_j(k)}\right]_{x(k)=\hat{x}(k+1)} \qquad i = 1...m,\ j = 1...L \quad (4)$$

Usually P, Q and R are initialized as diagonal matrices, with entries P_0, Q_0 and R_0, respectively.

To obtain H in (4) is not an easy task. In order to simplify the extended Kalman ltering implementation, in [4] and [5], an alternative algorithm, named as Extended Kalman Filtering, Derivative Free (EKFDF), is proposed. This algorithm is brie y discussed on the next section.

III. *EKFDF* Training Algorithm

This section closely follows [4], [5]. The *EKFDF* algorithm is obtained replacing the Taylor series expansion in the neighborhood of a point, by the Stirling equation expansion.

Given the operators □ and δ

$$\begin{aligned} \delta f(x) &= f\left(x+\frac{h}{2}\right) - f\left(x-\frac{h}{2}\right) \\ \square f(x) &= \frac{1}{2} f\left(x+\frac{h}{2}\right) - f\left(x-\frac{h}{2}\right) \end{aligned} \tag{5}$$

with h the selected interval $\left[\overline{x}-\frac{h}{2}, \overline{x}+\frac{h}{2}\right]$ length, then the Stirling interpolation equation, for $x=\overline{x}$, is established as [5]:

$$\begin{aligned} f(x) = f(\overline{x}+ph) = f(\overline{x}) + p\square\delta f(\overline{x}) + \frac{p^2}{2!}\delta^2 f(\overline{x}) \\ + \binom{p+1}{3}\square\delta^3 f(\overline{x}) + \frac{p^2(p^2-1)}{4!}\delta^4 f(\overline{x}) \\ + \binom{p+2}{5}\square\delta^5 f(\overline{x}) + \ldots \end{aligned} \tag{6}$$

If only rst and second order polynomials are used, then.

$$f(x) \approx f(\overline{x}) + f'_{DD}(\overline{x})[x-\overline{x}] + \frac{f''_{DD}(\overline{x})}{2!}[x-\overline{x}]^2 \tag{7}$$

where:

$$\begin{aligned} f'_{DD}(\overline{x}) &= \frac{f(\overline{x}+h) - f(\overline{x}-h)}{2h} \\ f''_{DD}(\overline{x}) &= \frac{f(\overline{x}+h) + f(\overline{x}-h) - 2f(\overline{x})}{h^2} \end{aligned} \tag{8}$$

In order to use this approach to implement the *EKFDF* algorithm [4], the Cholesky factorization ($\overline{S}_x(k) \in \Re^{LxL}$) of the covariance matrix (P) is considered (9). In order to simplify the notation, k is suppressed from here to the end of this section

$$P = \overline{S}_x \overline{S}_x^{\mathsf{T}} \tag{9}$$

The i-th column of $\overline{S}_x(k)$ is denoted by $s_{x,j}(k)$. From each such column vector, two variations of the current weights (x) are formed: $x+hs_{x,i}$ and $x-hs_{x,i}$. For each of these variations, the respective neural network output has to be computed. $g_j(x)$ stands for the j-th neural network output corresponding to x, $g_j(x+hs_{x,i})$ is the respective output for $x+hs_{x,i}$, and $g_j(x-hs_{x,i})$ is the representation of the output for $x-hs_{x,i}$.

Matrices $S_{xv}^{(1)} \in \Re^{LxL}$, $S_{yw}^{(1)} \in \Re^{mxm}$ are diagonal, with non-zero entries $Q_0^{\frac{1}{2}}$ and $R_0^{\frac{1}{2}}$ respectively. The entries of the matrix $S_{yx}^{(1)} \in \Re^{mxL}$ are de ned by (10), and the ones of matrix $S_{yx}^{(2)} \in \Re^{mxL}$ are de ned as (11).

$$\left(S_{yx}^{(1)}\right)_{ij} = \frac{1}{2h}\left(g_j(x+hs_{x,i}) - g_j(x-hs_{x,i})\right) \tag{10}$$

$$\left(S_{yx}^{(2)}\right)_{ij} = \frac{\left(h^2-1\right)^{\frac{1}{2}}}{2h^2}\left(g_j(x+hs_{x,i}) + g_j(x-hs_{x,i}) - 2g_j(x)\right) \tag{11}$$

Finally $S_y \in \Re^{mx(L+m+L)}$ is the concatenated matrix:

$$S_y = \begin{bmatrix} S_{yx}^{(1)} & S_{yw}^{(1)} & S_{yx}^{(2)} \end{bmatrix} \tag{12}$$

On the basis of matrices $\overline{S}_x$, S_y and $S_{yx}^{(1)T}$, it is possible to establish the *EKFDF* algorithm. The outer product matrix $P_y = S_y S_y^{\mathsf{T}}$, is analogous to the matrix $\left[R + H^{\mathsf{T}} P H\right]$ for the *EKF* algorithm, and the matrix (13) is equivalent to the matrix PH for the *EKF* algorithm.

$$P_{xy} = \overline{S}_x S_{yx}^{(1)\mathsf{T}} \tag{13}$$

Hence, the Kalman gain matrix is given by

$$K = P_{xy} P_y^{-1} \tag{14}$$

The update for the weight (state) vector is calculated by (15); the j-th column of $\overline{y}$ at k is de ned as (16). Finally, the covariance matrix is update as (17):

$$x(k+1) = x(k) + K(k)\left[y(k) - \overline{y}(k)\right] \tag{15}$$

$$\overline{y}_j = \frac{h^2-L}{h^2} g_j(x) + \frac{1}{2h^2}\sum_{i=1}^{L} g_j(x+hs_{x,i}) + g_j(x-hs_{x,i}) \tag{16}$$

$$P(k+1) = P(k) - K(k) P_y(k) K^T(k) + Q(k) \tag{17}$$

Table 1.
EKF and *EKFDF* Summary

ACTION	*EKF*	*EKFDF*
Initialization	x_0, P_0, Q_0, R_0	x_0, P_0, Q_0, R_0
Derivative calculations	(4)	
Divided differences		(10), (11)
Output estimate	$\widehat{y}$	(16)
Kalman Gain	(1)	(14)
Weight update	(2)	(15)
Covariance update	(3)	(17)

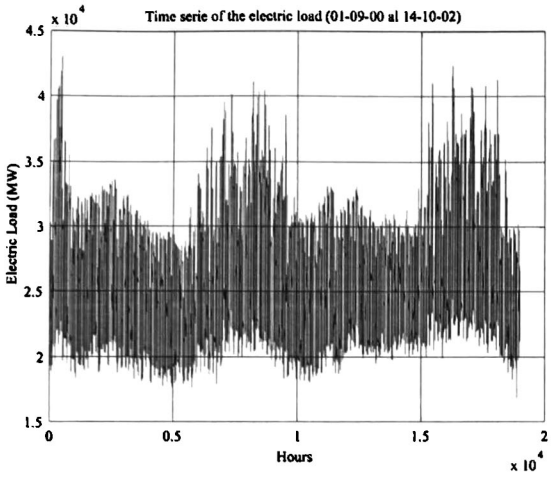


Fig. 1. State of California Electric Load Demand

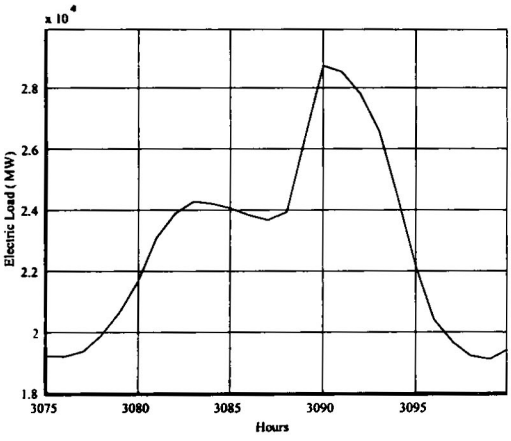


Fig. 2. Electric Load Demand for a Typical Day

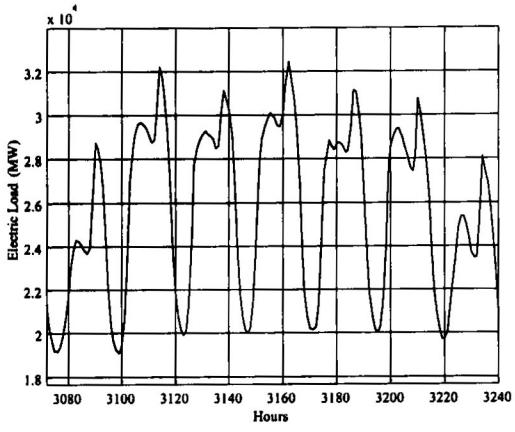


Fig. 3. Electric Load Demand for a Typical Week

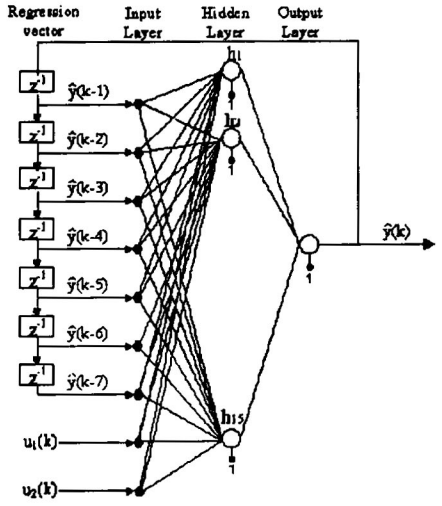


Fig. 4. Neural Network Structure

IV. Electric Load Demand Prediction

The goal of the present paper is to implement a 24 hours neural network predictor for the electric load demand, on the basis of Kalman ltering training. This predictor is developed using data from the State of California, USA [3]. Fig. 1 presents the data corresponding to a time lapse of 26 months; Fig 2 and Fig 3 display a typical day and a typical week, respectively.

A. EKF *Algorithm Application*

The neural network used is a *RMLP*, whose structure is presented in Fig. 4; the hidden layer has 15 neurons, with logistic activation functions, and the output layer is composed by just one neuron, with a linear activation function. The initial values for the covariance matrices (R, Q, P) are $R_0 = Q_0 = P_0 = 10000$. The length of the regression vector is 7 because of that is the order of the system, which was found with an algorithm based in the Lipschitz quotient, and is necessary to add two external signals, they correspond to the day and the hour ($u_1(k)$, $u_2(k)$).

The training is performed off-line, using a series-parallel con guration; for this case the delayed output are taken from the electric load demand. The speci ed target prediction error is 10^{-4}.Once the neural network is trained, its prediction capabilities are tested, with xed weights; using a parallel con guration, with delayed output taken from the neural network output.

The results are presented in Fig. 5, Fig. 6 and Fig. 7, for a week prediction, a prediction of 24 hours and the LMS error, respectively.

B. EKFDF *Algorithm Application*

The structure of the neural network is exactly the same explained above (Fig. 4) for the *EKF*; the following se-

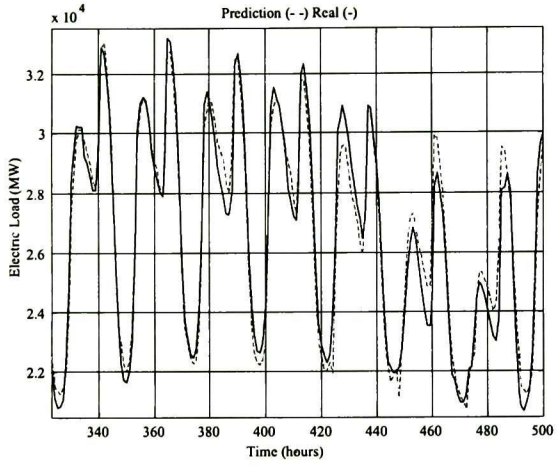


Fig. 5. *EKF* week prediction

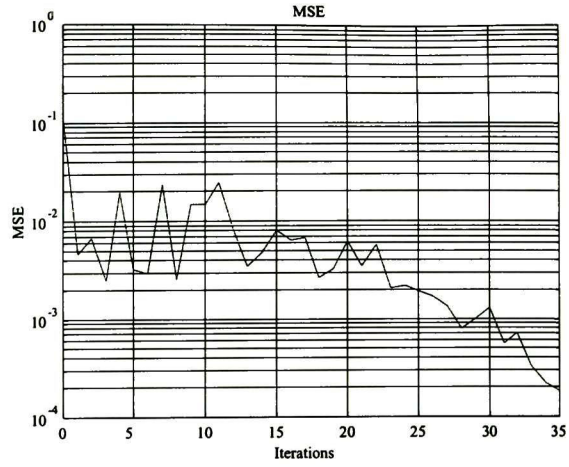


Fig. 7. LMS Error for the *EKF* Training

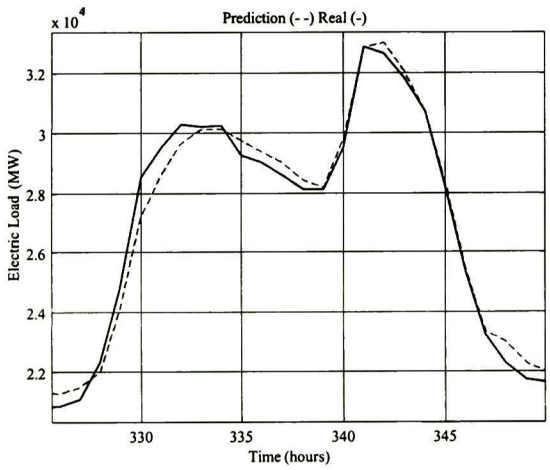


Fig. 6. *EKF* 24 hours prediction

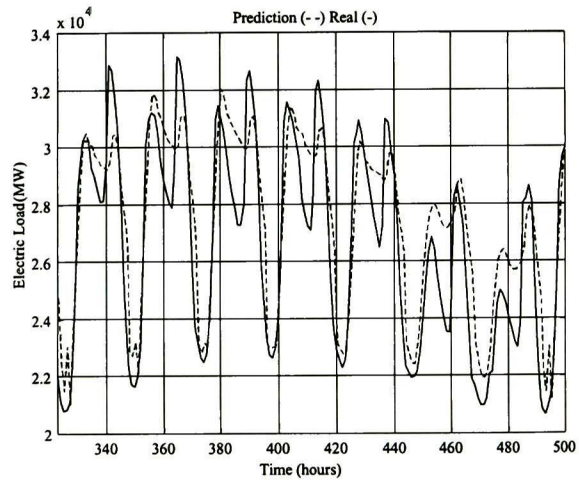


Fig. 8. *EKFDF* week prediction

lection is used: $h = \sqrt{3.}$, and $R_0 = Q_0 = P_0 = 5 * 10^{-6}$ The training and prediction are also performed along the same procedure as for the *EKF*. The respective results are displayed in Fig. 8, Fig. 9, and Fig. 10.

It is worth noting that the computer programming of this algorithm is easier than the former one.

C. Comparison

Even if the *EKFDF* algorithm is simpler to code, the *EKF* one requires less iteration to obtain the speci ed prediction error; the *EKFDF* algorithm never reaches this error. Comparison was also performed for the *Levenberg-Marquardt* backpropagation training algorithm and the *EKF* one, the last one performed much better.

V. CONCLUSIONS

This paper has discussed the application of *EKF* and *EKFDF* algorithms to training of neural networks. These algorithms present the following advantages: their convergence is less sensitive to initial conditions, they do not converge to local minima, they can be used for highly nonlinear mapping, and they can be implemented for almost any neural network structure. Even if the *EKFDF* algorithm is simpler to code, the *EKF* one requires less iteration to obtain the speci ed prediction error; the *EKFDF* algorithm never reaches this error. Taking in account all the above, we can conclude that, for this speci c application the best solution is to use *EKF* training.

The obtained results are very encouraging. Research will be proceed to test the *EKF* algorithm with different sets of data, and to test other kind of neural networks.

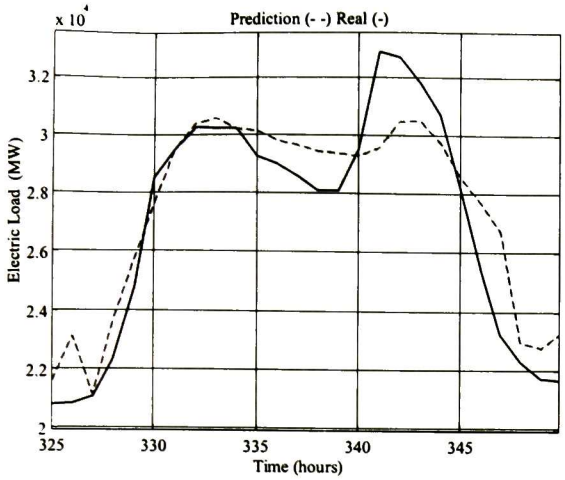


Fig. 9. *EKFDF* 24 hours prediction.

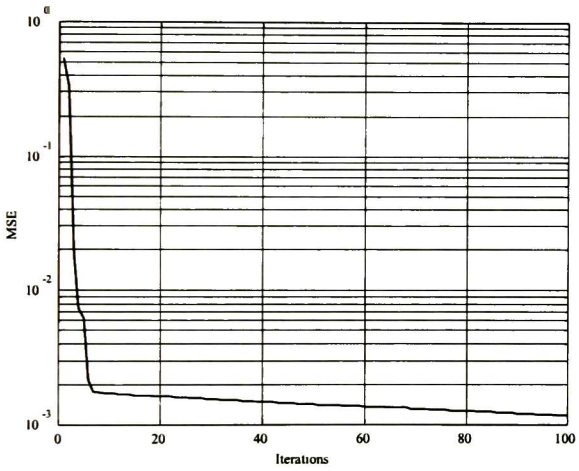


Fig. 10. LMS Error for the *EKFDF* training

The setting of R_0, Q_0 and P_0 varies a lot in the two algorithms because in the *EKFDF* they need to be small by the Cholesky factorization [4], [5].

Acknowledgement 1: The authors thank the support of Conacyt Mexico Project 39866Y.

REFERENCES

[1] C. C. Sumila, *Extracción de patrones temporales en base de datos de series de tiempo* , Universidad Michoacana de San Nicolás de Hidalgo, Facultad de Ingeniería Eléctrica, Morelia, Michoacan, México 2001.

[2] D. C. Park, M. A. El-Sharkawi, R. J. Marks, L. E. Atlas and M. J. Damborg, "Electric Load Forecasting Using an Arti cial Neural Network", *IEEE Transactions on Power Systems*, 1991, vol 6, No. 2, pp. 442-449.

[3] http://caiso.com.

[4] L. A. Feldkamp, T. M. Feldkamp, and D. V. Prokhorov, Neural network training with the nprKF , *Proceedings of International Joint Conference on Neural Networks 01*, 109-114, 2001.

[5] M. Norgaard , N. K. Poulsen , and O. Ravn, Advances in Derivative-Free State Estimation for Nonlinear Systems , *Technical Report IMM-REP-1988-15* (revised edition), Technical University of Denmark, 2000.

[6] R. Grover and P. Y.C. Hwang, *Introduction to random signals and applied kalman ltering* , Second edition, Wiley, New York, USA, 1992.

[7] S. Haykin, *Neural Networks. A comprehensive foundation* second edition, Prentice Hall, New Jersey, USA., 1999.

[8] S. Haykin, *Kalman Filtering and Neural Networks* Wiley, New York, USA, 2001.

[9] S. Singhal and L. Wu, *Training multilayer perceptrons with the extended Kalman algorithm* , in D. S. Touretzky (ed), Advances in Neural Information Processing Systems 1, pp. 133-140. San Mateo, CA, USA: Morgan Kaufmann, 1989.

[10] T. Senjyu, H. Takara, K. Uezato, T. Funabashi, "One-hour-ahead load forecasting using neural network", *IEEE Transactions on Power Systems*, 2002, vol 17, No. 1, pp. 113-118.

Professor Xinghuo Yu PhD, FIEAust, Emeritus Professor (CQU)
Associate Dean, Research & Development
Phone: +61 3 99254492
Fax: , +61 3 99254343
Email: x.yu@rmit.edu.au

RMIT UNIVERSITY

Science, Engineering & Technology Portfolio Office
Royal Melbourne Institute of Technology
Building 96, City Campus
GPO Box 2476V
Melbourne VIC 3001
AUSTRALIA

2 April 2004

Prof Edgar N. Sanchez
CINVESTAV, Unidad Guadalajara
Guadalajara, Jalisco, CP 45091
Mexico

Dear Professor Sanchez,

I am very pleased to inform you that your invited paper entitled

Recurrent Neural Networks Trained with Kalman Filtering for Discrete Chaos Reconstruction

has been accepted for presentation at the 3rd Asian-Pacific Workshop on Chaos Control and Synchronization (APWCCS'04) to be held in Melbourne Australia 19-20 July 2004.

As you are aware, the main purpose of this Workshop is to bring together leading researchers and experts worldwide, especially those from the Asia-Pacific region, to further promote and develop the cutting-edge research on chaos control and synchronization theory and its engineering applications. It will be an exciting event not to miss.

Please note that the acceptance of your paper is conditional on your attendance at the workshop. Please do not to hesitate to let me know if we can be of further assistance.

We are looking forward to seeing you in Melbourne in July 2004.

Yours sincerely

Professor Xinghuo Yu
Co-Chair, Organizing Committee APWCCS'04
http://web.fe.rmit.edu.au/~xinghuo/apwccs04.htm

Recurrent Neural Networks Trained with Kalman Filtering for Discrete Chaos Reconstruction

Edgar N. Sanchez[1], Alma Y. Alanis[1] and Guanrong Chen[2]
[1] CINVESTAV, Unidad Guadalajara, Apartado Postal 31-438, Plaza La Luna, Guadalajara, Jalisco, C.P. 45091, Mexico, e-mail:sanchez@gdl.cinvestav.mx
[2] Department of Electronic Engineering, City University of Hong Kong, P. R. China

Abstract— **This paper reports a new approach to discrete chaos reconstruction using recurrent neural networks. The required network learning is performed using the extended Kalman filter with a forgetting factor. The forgetting factor is used to avoid possible fluctuations throughout the learning process, which is a consequence of incorporating old information in learning. To illustrate the applicability of the proposed scheme, two examples are included: the Ikeda and the Lozi systems.**

Keywords— **Time series, Chaos reconstruction, Recurrent neural networks, Extended Kalman filter, Neural network learning, Forgetting factor.**

I. Introduction.

Nonlinear systems have very rich dynamical behaviors including chaos. The chaotic behavior is characterized by its high sensitivity to parameter variation, external disturbance, and particularly tiny changes of initial conditions [9]. Chaotic orbits are bounded but locally unstable [1]. Chaos presents some important properties that have led to many perhaps not-so-traditional technological applications [2]. On the other hand, however, it implies that reproducing chaos is technically a nontrivial task for systems modelling and control.

Recently, controlling and ordering chaos has received increasing attention from various scientific and engineering communities. To control a chaotic system, similarly to controlling a general nonlinear system, an important requirement usually is to have a good model of the underlying system. For many nonlinear systems, particularly chaotic systems, it is often difficult to obtain their accurate and faithful mathematical models, regarding their physically complex structures and hidden parameters as discussed in [3]. Therefore, system identification becomes important and even necessary before system control can be considered.

Neural networks have grown to be a well-established methodology for modelling and control, which allows for solving some very difficult problems in engineering, as exemplified by their applications to identification and control of general nonlinear and complex systems. In particular, the use of recurrent neural networks for modelling and learning has rapidly increased in recent years [10].

There exist different training algorithms for neural networks, which, however, normally encounter some technical problems such as local minima, slow learning, and high sensitivity to initial conditions, among others. As a viable alternative, new training algorithms, e.g., those based on Kalman filtering, have been proposed [5], [6], [11]. Due to the fact that training a neural network typically results in a nonlinear problem, the Extended Kalman Filter (EKF) is a common tool to use [6].

This paper proposes a learning algorithm based on the EKF with a forgetting factor for recurrent neural network modelling, validated by discrete chaos reproduction. Two examples will be worked out for verification and illustration.

The paper is organized as follows: first, the neural network model to be used is presented; then the Kalman filtering learning algorithm is discussed, followed by the new one using a forgetting factor; to this end, a two discrete chaotic system are presented, which are reproduced by the recurrent neural network model; finally, some relevant conclusions are given.

II. Discrete-time Recurrent Neural Networks

Consider the following discrete-time recurrent neural network:

$$x_i(k+1) = w_i^{\top} z(x(k), u(k)), \quad i = 1, \cdots, n \tag{1}$$

where x_i $(i = 1, 2, \cdots, n)$ is the state of the ith neuron, n is the state dimension, w_i $(i = 1, 2, \cdots, n)$ is the on-line adapted weight vector, and $z(x(k), u(k))$ is given by (2) below, with $d_j(k)$ being a nonnegative integer, and y is defined in (3) below:

$$z(x(k), u(k)) = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_L \end{bmatrix} = \begin{bmatrix} \Pi_{j \in I_1} y_j^{d_j(1)} \\ \Pi_{j \in I_2} y_j^{d_j(2)} \\ \vdots \\ \Pi_{j \in I_{L_i}} y_j^{d_j(L_i)} \end{bmatrix} \tag{2}$$

$$y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \\ y_{n+1} \\ \vdots \\ y_{n+m} \end{bmatrix} = \begin{bmatrix} S(x_1) \\ \vdots \\ S(x_n) \\ S(u_1) \\ \vdots \\ S(u_m) \end{bmatrix} \tag{3}$$

In (3), $u = [u_1, u_2, \ldots, u_m]^{\top}$ is the input vector to the neural network, and $S(\bullet)$ is defined by

$$S(x) = \frac{1}{1 + \exp(-\beta x)} + \varepsilon \tag{4}$$

where β is a positive constant and ε a small positive number. Clearly, $S(x) \in [\varepsilon, \varepsilon + 1]$.

III. The EKF Training Algorithm

Kalman filtering (KF) estimates the state of a linear system with additive state and output white noises [4], [6].

For KF-based neural network training, the network weights become the states to be estimated, with the error between the neural network and the desired output being considered as additive white noise. Due to the fact that the neural network mapping is nonlinear, an EKF-type of algorithm is required.

Neural network training is performed using a set of N input-output measurement pairs. The training goal is to find the optimal weight values that minimize the prediction errors (the differences between the measured outputs and the neural network outputs). The EKF-based training algorithm is based on the following equations:

$$K(k) = P(k) H(k) \left[R(k) + H^{\top}(k) P(k) H(k)\right]^{-1} \quad (5)$$

$$w(k+1) = w(k) + K(k) \left[y(k) - \widehat{y}(k)\right] \quad (6)$$

$$P(k+1) = P(k) - K(k) H^{\top}(k) P(k) + Q(k) \quad (7)$$

where $P(k) \in \Re^{L \times L}$ and $P(k+1) \in \Re^{L \times L}$ are the prediction error covariance matrices at steps k and $k+1$, respectively, $w \in \Re^{L}$ is the weight (state) vector, L is the total number of neural network weights, $y \in \Re^{m}$ is the measured output vector, $\widehat{y} \in \Re^{m}$ is the network output, m is the total number of outputs, $K \in \Re^{L \times m}$ is the Kalman gain matrix, $Q \in \Re^{L \times L}$ is the state noise covariance matrix, $R \in \Re^{m \times m}$, $H \in \Re^{L \times m}$ is a matrix, in which each entry (H_{ij}) is the derivative of one of the neural network output, $(\widehat{y}_i)$, with respect to one neural network weight, (w_j), as follows:

$$H_{ij}(k) = \left[\frac{\partial \widehat{y}_i(k)}{\partial w_j(k)}\right]_{w(k)=\widehat{w}(k+1)} \qquad i = 1, ..., m,\ j = 1, ..., L \quad (8)$$

In this case, $\widehat{y}(k) = x(k)$, so by the chain rule, one has

$$\frac{\partial \widehat{y}(k)}{\partial w(k)} = \frac{\partial \widehat{y}(k)}{\partial x(k)} \frac{\partial x(k)}{\partial w(k)} \quad (9)$$

where x is the state of the neuron. Thus, one can rewrite (1) in the form of (10).

To this end, linearization is applied with the recurrent derivatives routine [6], giving a dynamic system in the form of (11) as follows:

$$x(k+1) = F(x(k), u(k), w(k)) \quad (10)$$

Consequently,

$$\frac{\partial x(k+1)}{\partial w(k)} = \frac{\partial F(x(k), u(k), w(k))}{\partial x(k)} \frac{\partial x(k)}{\partial w(k)} + \frac{\partial F(x(k), u(k), w(k))}{\partial w(k)} \quad (11)$$

where $F(\cdot, \cdot)$ is a nonlinear function of $x(k), u(k)$, and $w(k)$.

Usually P, Q and R are initialized as diagonal matrices, with entries P_0, Q_0 and R_0, respectively.

A. The EKF with a Forgetting Factor

In the above algorithm, all the information (old and new measurements) are treated equally important. This can have some complications: if the plant changes its parameters fast, old information could obstruct the convergence for the weight (state) vector to the new value. Thus, the algorithm is modified in order to weight more the most recent information and to forget the old information. This is done by including a forgetting factor. The EKF with a forgetting factor is given by (12)

$$\begin{aligned} K(k) &= P(k) H(k) [\lambda(k) R(k) \\ &\quad + H^{\top}(k) P(k) H(k)]^{-1} \\ w(k+1) &= w(k) + K(k) [y(k) - \widehat{y}(k)] \\ P(k+1) &= \lambda^{-1}(k) P(k) - \lambda^{-1}(k) K(k) H^{\top}(k) P(k) \\ &\quad + Q(k) \end{aligned} \quad (12)$$

where $\lambda(k)$ is the forgetting factor, and all other expressions are defined as above.

The use of the factor $\lambda(k)$ is intended to ensure that data in the distant past are "forgotten" in order to afford the possibility to track the statistical variations of the observable data when the neural network operates in a nonstationary environment [7]. Here, $\lambda(k)$ is a positive constant, close to but less than 1; when $\lambda(k)$ equals 1, one has the ordinary extended Kalman filter.

Forgetting factors are often used for tracking varying dynamic systems in recursive identification and adaptive control algorithms. In this way, a faster convergence can be achieved [8].

IV. Discrete Chaos

A nonlinear system can have a complicated dynamical behavior known as chaos. Hallmarks of chaos include the extreme sensitivity of the system dynamics to its initial conditions, the existence of a dense set of unstable periodic orbits in its regime, positive Lyapunov exponents, finite KS-entropy, continuous power spectrum, ergodicity and some other limiting properties, etc. [1]

Two examples of discrete chaotic systems are briefly discussed.

A. The Ikeda system

The Ikeda system is described by [6]

$$\begin{aligned} \chi_1(k+1) &= 1 + \mu \{\chi_1(k) \cos[m(k)] - \chi_2(k) \sin[m(k)]\} \\ \chi_2(k+1) &= 1 + \mu \{\chi_1(k) + \chi_2(k) \cos[m(k)]\} \end{aligned} \quad (13)$$

with

$$m(k) = 0.4 - \frac{6}{1 + \chi_1^2(k) + \chi_2^2(k)} \quad (14)$$

where χ_1 and χ_2 are the real and imaginary components, respectively, of χ, and the parameter μ is key to produce chaos.

B. The Lozi system

Another interesting discrete chaotic system is the Lozi system [3], described by

$$\begin{aligned} \lambda_1(k+1) &= -p|\lambda_1(k)| + \lambda_2(k) + 1 \\ \lambda_2(k+1) &= q\lambda_1(n) \end{aligned} \tag{15}$$

where p and q are real parameters. As simple it is, Lozi system is rife with chaotic phenomena.

V. Discrete Chaos Reproduction

In order to reproduce discrete chaos, the above-described recurrent neural network is employed, using its supervised learning performed on-line by the EKF with a forgetting factor. Training data are generated on-line by the underlying (presumably unknown) chaotic system.

A. The Ikeda system

The underlying system is given by (13 and 14). The parameters used are presented in Table 1:

Parameter	Value
μ	0.9
$\chi_1(0)$	0.1
$\chi_2(0)$	0.1
$\lambda(k)$	0.9

Table 1. Parameters used for the Ikeda System.

The phase plot of $\chi_1(k+1)$ versus $\chi_1(k)$ shows a chaotic attractor (Figure 1). The neural network, used for reconstructing this chaotic system, is given by

$$\begin{aligned} x_1(k+1) &= w_{11}S(x_1(k)) + w_{12}x_2(k) \\ x_2(k+1) &= w_{21}S(x_2(k)) + w_{22}x_3(k) \\ x_3(k+1) &= w_{31}S(x_3(k)) + w_{32}x_4(k) \\ x_4(k+1) &= w_{41}S(x_4(k)) \end{aligned} \tag{16}$$

where $\{x_i : i = 1, \cdots, n\}$ are the states of the neural network; $S(\cdot)$ is given as in (4); and the set: $\{w_{ij} : i = 1, \cdots, n,\ j = 1, \cdots 4\}$ are the weights of the network.

The results are presented in Figures 1 to 5. The first figure portrays the original attractor, the second one is the neural-network reproduction, the third is the one-step prediction time series, performed by the trained neural network, the fourth shows the weight evolution, and the last one shows the evolution of covariance P during the training.

B. The Lozi system

The Lozi system is given by (15). The parameters used in this simulation are presented in Table2:

Parameter	Value
p	1.8
q	-1
$\lambda_1(0)$	-5000
$\chi_2(0)$	5000
$\lambda(k)$	0.68

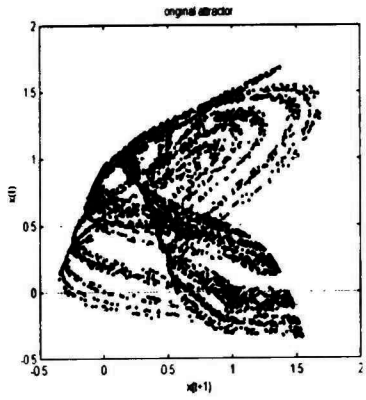


Fig. 1. Original Ikeda attractor

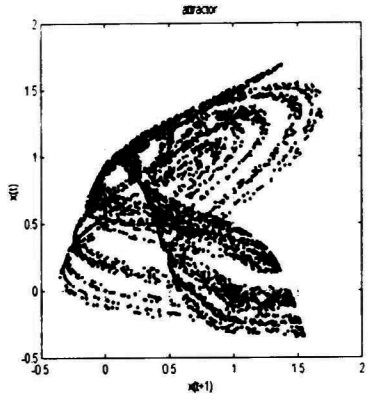


Fig. 2. Neural-network-produced Ikeda attractor

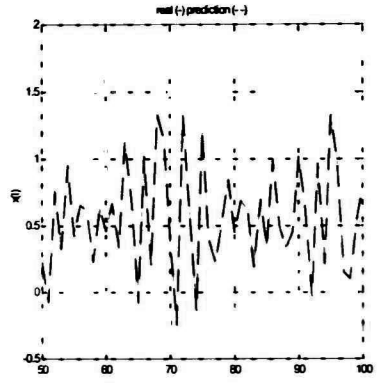


Fig. 3. One-step prediction of the Ikeda attractor.

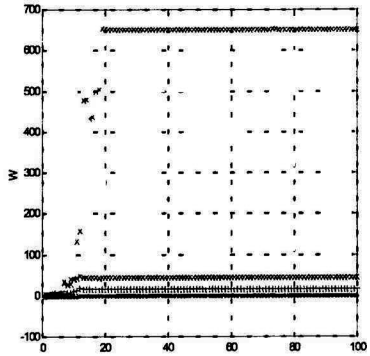


Fig. 4. Weight evolution.

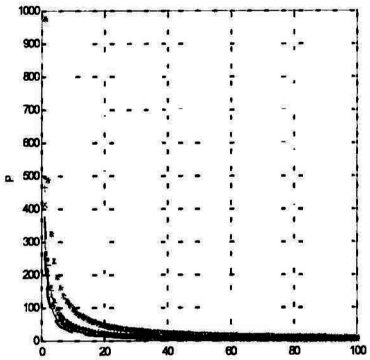


Fig. 5. Covariance evolution.

Table 2. Parameters used for the Lozi System.

The phase plot of $\chi_1(k+1)$ versus $\chi_1(k)$ shows a chaotic attractor (Figure 6). The neural network trained for reconstructing this chaotic system is given by

$$\begin{aligned} x_1(k+1) &= w_{11}S(x_1(k)) + w_{12}x_2(k) \\ x_2(k+1) &= w_{21}S(x_2(k)) + w_{22}x_3(k) \\ x_3(k+1) &= w_{31}S(x_3(k)) \end{aligned} \tag{17}$$

where $\{x_i : i = 1, \cdots, n\}$ are the states of the neural network; $S(\cdot)$ is as in (4); and the set $\{w_{ij} : i = 1, \cdots, n\ , j = 1, \cdots, 3\}$ are the weights of the network.

The results are presented in Figures 1 to 5. The first figure portrays the original attractor, the second one is the neural-network reproduction, the third is the one-step prediction time series, performed by the trained neural network, the fourth shows the weight evolution, and the last one shows the evolution of the covariance P during the training .

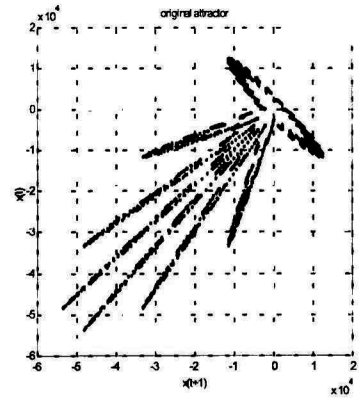


Fig. 6. Original Lozi attractor

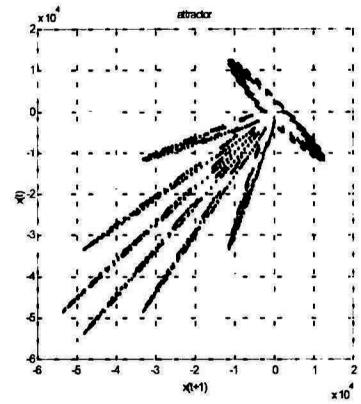


Fig. 7. Neural-network-produced Lozi attractor.

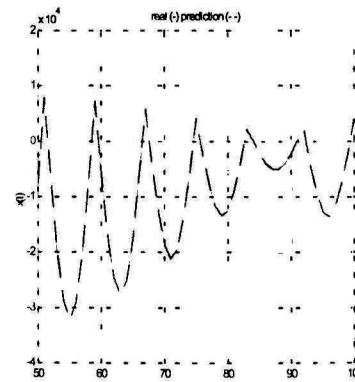


Fig. 8. One-step prediction of Lozi attractor.

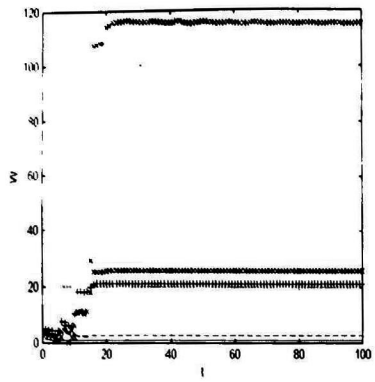


Fig. 9 Weight evolution.

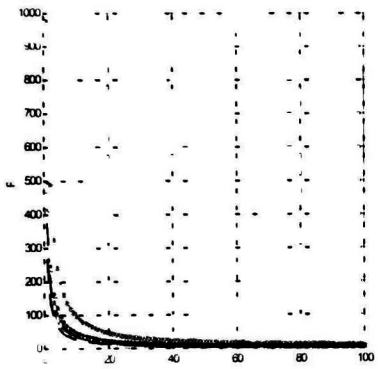


Fig. 10. Covariance evolution.

VI. Conclusions

This paper has given an effective learning scheme that can well reproduce complex dynamics such as chaos, and is also able to implement one-step prediction of chaotic time series The extended Kalman filtering learning algorithm seems to be very suitable for recurrent neural networks, which presents low sensitivity to initial conditions. Together with a forgetting factor. it is possible to achieve a faster convergence. It is impossible to find the "most adequate value" for the forgetting factor, as is well known, since it depends on a lot of issues related to the given system. For chaos reproduction. using recurrent neural networks allows for using a small number of neurons. as demonstrated by the two simulated examples.

Acknowledgement 1: The authors thank the support of CONACYT. Mexico on project 39866Y.

References

[1] G. Chen and J. L. Moiola. "An overview of bifurcation, chaos and nonlinear dynamics in control systems," *The Franklin Institute Journal*. vol. 331B. pp. 819-858. 1994.

[2] G. Chen and X. Yu (eds.), *Chaos Control: Theory and Applications*, Springer-Verlad, Berlin, 2003.

[3] G. Chen and X. Dong (eds.), *From Chaos to Order: Methodologies, Perspectives and Applications*, World Scientific Pub. Co., Singapore, 1998.

[4] C. K. Chui and G. Chen, *Kalman Filtering with Real-Time Applications*, Springer-Verlag, New York, 1st ed. 1987, 3rd ed. 1998.

[5] R. Grover and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*, 2nd ed., Wiley, New York, 1992.

[6] S. Haykin, *Kalman Filtering and Neural Networks*, Wiley, New York, 2001.

[7] S. Haykin, *Adaptive Filter Theory*, Prentice Hall, New Jersey,1991.

[8] L. Ljung, *System Identification, Theory for the User*. Prentice Hall, Upper Saddle River, N. J.,Second edition, 1999.

[9] J. P. Perez, *Recurrent Neural Control: An Inverse Optimality Approach*, Ph. D Thesis (in Spanish), Cinvestav, Unidad Guadalajara, Guadalajara Jalisco Mexico, 2003.

[10] E. N. Sanchez and J. L. Ricalde, "Trajectory tracking via adaptive recurrent neural control with input saturation," *Proceedings of International Joint Conference on Neural Networks'03*, Portland, Oregon, USA, July 2003.

[11] S. Singhal and L. Wu, "Training multilayer perceptrons with the extended Kalman algorithm," in D. S. Touretzky (ed.), *Advances in Neural Information Processing Systems*, vol. 1, pp. 133-140, Morgan Kaufmann, San Mateo, CA, 1989.

Centro de Investigación y de Estudios Avanzados del IPN
Unidad Guadalajara

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, aprobó la tesis:

Entrenamiento de redes neuronales con el filtro de Kalman

del (la) C.

Alma Yolanda ALANIS GARCIA

el día 20 de Agosto de 2004.

Dr. Edgar Nelson SANCHEZ CAMPEROS
Investigador Cinvestav 3C
CINVESTAV GDL
Jalisco

Dr. Alexander GEORGIEVICH LOUKIANOV
Investigador Cinvestav 3C
CINVESTAV GDL
Jalisco

Dr. Juan Manuel RAMÍREZ ARREDONDO
Investigador Cinvestav 3B
CINVESTAV GDL
Jalisco

Dr. J. Jesus RICO MELGOZA
Profesor Investigador Titular C
Universidad Michoacana de San Nicolás de Hidalgo, Estudios de Posgrado
Michoacán