UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

PROGRAMA DE MAESTRIA Y DOCTORADO EN INGENIERIA

FACULTAD DE INGENIERIA

DESARROLLO DE UN OBSERVADOR Y CONTROLADOR NEURODIFUSO PARA UN SISTEMA INTERCONECTADO

T E S I S

QUE PARA OPTAR POR EL GRADO DE:

MAESTRO EN INGENIERIA

ELECTRICA – CONTROL

P R E S E N T A :

PAUL ERICK MENDEZ MONROY

TUTOR:
DR. HÉCTOR BENÍTEZ PÉREZ

2007

JURADO ASIGNADO:

Presidente:	Dr. Álvarez Icaza Longoría Luis A.
Secretario:	Dr. Espinosa Pérez Gerardo René
Vocal:	Dr. Benítez Pérez Héctor
1er. Suplente:	Dr. Tang Yu Xu
2do. Suplente:	Dr. Arteaga Pérez Marco

Lugar donde se realizo la tesis:

INSTITUTO DE INVESTIGACIONES EN MATEMÁTICAS APLICADAS Y SISTEMAS (IIMAS)

TUTOR DE TESIS:

Dr. BENÍTEZ PÉREZ HÉCTOR

_____

FIRMA

AGRADECIMIENTOS

Esta tesis se la dedico a toda mi familia por su apoyo y comprensión durante la realización de esta maestría, a mi madre por estar conmigo en todo momento, sus preocupaciones de q siempre este bien, por sus desvelos; a mi padre por ser un ejemplo de persona responsable, por dedicación hacia mi familia y por su amor incondicional. A mis hermanos por su apoyo y empuje hacia ser siempre mejores, por darme ánimos y simplemente porque me amen, a todos ustedes gracias por compartir su vida conmigo.

Un agradecimiento muy especial a Héctor Benítez por su apoyo, sus dedicación y más por su amistad durante esta etapa de mi vida, por ser más que un profesor un amigo, por siempre impulsarme a cumplir las metas y dar un poco mas. Un agradecimiento para todos los profesor del posgrado por su dedicación para mi formación académica y personal.

Un agradecimiento especial a todos mis amigos de la LBS, Laura, Johana, Gabriela, Ulises, Alex, por apoyarme durante todo este tiempo. Por sus atenciones y sonrisas por estar siempre cuando se les necesita. Por haber conocido y querido al huevo, que también se le extraña, por todo gracias.

Un agradecimiento a todos mis amigos de la UNAM por permitirme se parte de sus vidas y formar parte de la mía, por todos sus consejos, gracias a todos, de psicología, química, ciencias, iimas, etc.

Un agradecimientos al personal del IIMAS-DISCA por permitirme el uso de las instalaciones y brindarme su amistad.

Por ultimo pero no menos importante a todos mis amigos desde vocacional, por seguir apoyándome y darme ánimos a continuar en mi camino hacia la felicidad.

INDICE

Capítulo 1

1 Introducción

Durante los últimos años el estudio en redes neuronales y en sistemas difusos ha sido ampliamente desarrollado. Sin embargo, debido a sus orígenes y paradigmas se han desarrollado de forma independiente. Es conocido que la lógica difusa ofrece un marco muy poderoso para el razonamiento aproximado intentando modelar el proceso de razonamiento humano en un nivel cognitivo. Los sistemas difusos adquieren el conocimiento gracias al dominio del experto el cual es codificado dentro del algoritmo en términos de un conjunto de reglas difusas IF-THEN. Los sistemas difusos emplean este enfoque basado en reglas e interpolan el razonamiento para responder a escenarios desconocidos.

En contraste, las redes neuronales ofrecen una arquitectura altamente estructurada, con capacidades de aprendizaje y generalización, que intenta imitar el mecanismo neurológico del cerebro. Una red neuronal almacena conocimiento de una manera distribuida a través del ajuste de sus pesos, los cuales han sido determinados por medio de un entrenamiento (aprendizaje) con escenarios conocidos. La habilidad de generalización para nuevos escenarios está basada en la estructura algebraica inherente de la red neuronal. Ha sido demostrado que ambas tecnologías tienen capacidades de aproximadores universales [8].

Recientemente ha habido un interés considerable en la integración de las redes neuronales y la lógica difusa, las dos tecnologías pueden ser vistas como complementarias y como consecuencia, esta combinación proporciona un marco de razonamiento que tiene capacidades de aprendizaje y generalización. La tecnología combinada cubre algunas desventajas de estas técnicas como por ejemplo la poca transparencia de la red neuronal y la capacidad de aprendizaje limitada de los sistemas difusos [17].

En los últimos años se han realizado múltiples estudios sobre sistemas neurodifusos reportados en la literatura enfocados a modelado y control de sistemas dinámicos. En general estos son sistemas con una estructura fija que interpretan un sistema difuso en

términos de una red neuronal tal que cada paso en el sistema difuso es equivalente a por lo menos una capa de la red [13]. Así, la mayoría de estas arquitecturas tienen al menos una capa correspondiente a difusión, intersección, implicación y desdifusión. Las arquitecturas difieren de una red neuronal convencional en términos de la uniformidad tanto de los nodos de procesamiento como de la estrategia de la interconexión resultando con frecuencia en algoritmos de entrenamiento modificados. Sin embargo, estas arquitecturas no proporcionan un procedimiento sistemático para determinar el número de reglas del sistema de inferencia difuso.

Hay arquitecturas que fijan el número de reglas difusas antes de realizar el entrenamiento de la red [5] cuya tarea es dejada a los expertos. Dicha tarea no es fácil, por lo tanto, determinar el número de reglas a utilizar representa un problema para los sistemas neurodifusos.

Algunos sistemas neurodifusos llamados autoorganizados consideran al número de reglas como un parámetro de diseño más, el cual puede ser determinado utilizando un algoritmo de agrupamiento de datos (*clustering*), generalmente los datos son la entrada y la salida del sistema [10][13][27]. Estas técnicas de agrupamiento proporcionan el planteamiento para encontrar la estructura del sistema neurodifuso que caracteriza un conjunto de datos, que define el número de reglas difusas en el sistema neurodifuso. Los métodos de agrupamiento desarrollados en [13][27] son adecuados para procesar los datos fuera de línea, también se han desarrollado estudios que emplean técnicas de agrupamiento que procesan los datos en línea para poder modelar sistemas dinámicos vía sistemas neurodifusos en tiempo real [11].

La desventaja de estos sistemas neurodifusos es que no presentan un análisis de estabilidad, característica principal para modelar o controlar un sistema dinámico. En [4] se propone un modelo neurodifuso para identificación de sistemas dinámicos con análisis de estabilidad basado en el segundo método de Lyapunov, este modelo lo realiza mediante dos redes neurodifusas, una estática y otra dinámica, pero el número de reglas es fijo lo cual representa una desventaja en esta técnica. Resulta interesante el desarrollar técnicas para la generación de sistemas neurodifusos que permita el análisis de estabilidad, para el modelo y control de sistemas dinámicos.

1.1 Motivación

El control automático ha desempeñado una función vital en el área de la ingeniería, es una parte esencial e integral de los procesos industriales. Esto ha permitido el desarrollo de la teoría de control, cuyo objetivo es aportar los medios para obtener un desempeño óptimo de sistemas dinámicos, mejorar la productividad, disminuir los riesgos y fallas de operación, así como otras funciones.

Dentro de la teoría de control clásico se han desarrollado múltiples métodos para el modelado de sistemas dinámicos [12][19] pero la mayoría requiere información del sistema para poder ser modelado y/o controlado. no siempre es factible disponer de información del sistema, tales como sistemas dinámicos complejos o de gran escala que cuentan con poca o nula información, sin embargo, se tiene la necesidad de modelarlo y/o controlarlo, por lo que estudios de nuevas técnicas que emplean solo información de las señales medibles del sistema se han realizado en años recientes [1][16].

Esto ha motivado a una parte de la comunidad de control al estudio de los sistemas neurodifusos que tienen la capacidad de ser diseñados con poca información del sistema. A partir de datos de entrada-salida del sistema y el diseño de algunos parámetros del sistema, se puede obtener un modelo neurodifuso con alguna exactitud deseada [5][10][13] [25][27]. Ninguna de estas técnicas tiene una estructura capaz de proporcionar un análisis de estabilidad que es necesario para el modelado y control de un sistema, por lo que es una área de investigación que resulta de interés desarrollar.

La motivación principal es obtener un sistema neurodifuso para modelado y control con la capacidad de obtener una estructura y aprendizaje del sistema a partir de señales de entrada-salida del sistema, con la posibilidad de realizar un análisis de estabilidad, para garantizar el desempeño de funciones de observación y control.

1.2 Objetivo de la tesis

En esta tesis se propone un sistema neurodifuso SISO para el modelado y control de sistemas dinámicos capaz de generar la estructura del sistema a partir de las señales de entrada-salida donde la identificación de la estructura como el aprendizaje de parámetros se realizan fuera de línea. Se propone una estructura del sistema neurodifuso con la capacidad

de interconectarse para poder modelar y controlar un sistema dinámico MIMO a partir de múltiples modelos neurodifusos SISO.

1.3 Resumen de tesis

El presente trabajo presenta una nueva estructura de una red neurodifusa recurrente con representación en espacio de estados, para ser utilizado como modelo para predicción o simulación de un sistema real a partir de datos entrada-salida. A partir del modelo neurodifuso recurrente se diseña un observador para asegurar estabilidad del modelo, y una estrategia de control para el sistema real.

La estructura propuesta es recurrente en las variables de estado propuestas por el diseñador. El número de reglas difusas de la red es establecido por un método de cluster. Se presenta el análisis de estabilidad para los tres elementos (modelo, observador y controlador) basado en la teoría de Lyapunov. Los parámetros de la red son obtenidos mediante un algoritmo de entrenamiento basado en el método del gradiente (RTRL[1]).

El Capítulo 2 presenta los antecedentes para definir los elementos de un modelo matemático, también presenta un resumen de los sistemas difusos y neuronales, así como su combinación. El Capítulo 3 presenta la nueva estructura de la red neurodifusa para el modelado de un sistema real a partir de datos entrada-salida, partiendo de que no se tienen reglas difusas; se presentan los algoritmos de generación de reglas y de actualización de parámetros para la red neurodifusa, así como el análisis de estabilidad. El Capítulo 4 presenta el diseño del observador y controlador basados en el modelo neurodifuso propuesto, también presenta el análisis de estabilidad para ambos. El Capítulo 5 presenta dos sistemas no lineales para probar la efectividad la estructura propuesta para modelado, observación y control. Finalmente se presentan las conclusiones obtenidas después de la realización del trabajo, así como el trabajo futuro a realizar.

[1] RTRL Real Time Recurrent Learning

Capítulo 2

2 Modelado

El modelado de sistemas es una de las funciones más importantes en la que se basa la teoría de control. La calidad del modelo generalmente determina un límite de la calidad de un objetivo final, como puede ser robustez del sistema, detectar fallas, etc.

Los modelos matemáticos son usados particularmente en las disciplinas de las ciencias naturales y de ingeniería tales como física, biología e ingeniería eléctrica, pero también las ciencias sociales tales como economía, sociología y ciencias políticas.

2.1 Modelo matemático

Un modelo matemático usualmente describe un sistema por un conjunto de variables y un conjunto de ecuaciones que establecen relaciones entre las variables. Los valores de las variables pueden ser prácticamente cualquiera; números reales o enteros, valores boléanos o caracteres, etc. Las variables representan algunas propiedades del sistema, por ejemplo, las salidas del sistema medidas como señales, datos de tiempo, contadores, ocurrencia de eventos, etc. El modelo es el conjunto de funciones que describen las relaciones entre las diferentes variables [15].

Hay seis grupos básicos de variables: variables de decisión, variables de entrada, variables de estado, variables exógenas, variables aleatorias, y variables de salida. Puede haber variables que pertenezcan a múltiples grupos, las variables son generalmente representadas por vectores.

Las variables de decisión son comúnmente conocidas como variables independientes. Las variables exógenas son conocidas como parámetros o constantes. Las variables no son necesariamente independientes de estar en otros grupos; por ejemplo, las variables de estado son dependientes de las variables de decisión, entrada, aleatorias y exógenas; Además, las variables de salida son dependientes del estado del sistema (representado por las variables de estado).

Los objetivos del sistema pueden ser representados como funciones de las variables de salida o variables de estado. Las funciones objetivo dependerán de la perspectiva del modelo. Dependiendo del contexto, una función objetivo es también conocida como un índice de desempeño, siendo esta la medida de mayor interés para el usuario. La calidad de un modelo es generalmente medida en términos de una función de error entre la salida del sistema y la salida del modelo, donde el error es utilizado para ajustar los parámetros del modelo.

Se pueden seguir ciertas tareas para lograr un modelo exitoso que cumpla con representar la conducta de un sistema lo más cercanamente posible y permita realizar un objetivo final [12][14].

Tarea 1: Elección de las señales de entrada al modelo. Es una tarea que se realiza por prueba y error y en función del conocimiento del sistema. Entre menos conocimiento del sistema o mayor complejidad de este es más difícil identificar las variables que influyen en el sistema. Es decir contemplar entradas actuales del sistema, entradas retrasadas, salidas retrasadas, perturbaciones, etc.

Tarea 2: Elección de las señales de excitación. Requiere conocimiento a priori sobre el sistema y la propuesta del modelo. La conducta del sistema que no es representada con el conjunto de datos puede no ser descrita por el modelo a menos que un conocimiento a priori sea explícitamente incorporado. Si el sistema no puede ser excitado, se emplean las mediciones que mejor representen la conducta del sistema.

Tarea 3: Elección de la arquitectura del modelo. Es posiblemente la decisión más difícil y subjetiva. Depende de ciertos criterios como el *tipo de problema*, si es clasificación, aproximación o identificación, etc. El *uso previsto*, se refiere a las propiedades que deben tener el modelo definidas por el objetivo final. La *cantidad y calidad de los datos disponibles*, el *problema de dimensionalidad*, *aprendizaje fuera de línea o en línea*, *experiencia del usuario*, entre las más importantes. En gran parte esta tarea define la representación dinámica y el orden que tendrá el modelo.

Cada arquitectura tiene ventajas y desventajas y su elección depende principalmente del sistema que se trate y de las circunstancias especificas del objetivo a cumplir. Uno de los aspectos más importantes para la elección de una arquitectura es la *complejidad* del modelo que depende de la *precisión* que se requiera del modelo. Se debe establecer un

equilibrio entre estas dos, a mayor precisión requerida de un modelo, mayor será la complejidad. Lo más conveniente es obtener un modelo matemático con la suficiente precisión para lograr el objetivo final que se tiene en consideración sin que haya un exceso de complejidad en el mismo. De acuerdo a su complejidad los modelos se pueden dividir básicamente en dos categorías: los modelos lineales y los modelos no lineales.

Los modelos lineales identifican sistemas dinámicos generalmente ignorando ciertas propiedades inherentes al sistema que no afectan considerablemente la respuesta del sistema, como ciertas no linealidades y parámetros distribuidos. Dichos modelos son válidos solamente en ciertas regiones de operación. La ventaja de usar un modelo lineal es que la teoría de control ha sido estudiada tanto en modelado como en otros objetivos.

Los modelos no lineales identifican sistemas dinámicos contemplando que la respuesta es afectada por ciertas no linealidades y/o parámetros distribuidos. Debido ha esto su modelo matemático es más complejo y algunos modelos son específicos cumpliendo con objetivos particulares a determinados sistemas. La desventaja de usar un modelo no lineal es que la teoría no ha sido ampliamente estudiada y es más complicada que la de los modelos lineales.

El primer paso para elegir una arquitectura es realizar un modelo lineal, si no cumple con el objetivo final para el cual se diseñó se considera el diseño de un modelo no lineal que permita cumplir el objetivo. El siguiente paso es decidir entre un modelo estático o dinámico, considerar si el modelo tendrá un comportamiento en función del tiempo o no. Si el modelo del sistema establece que la respuesta depende únicamente de las entradas actuales del sistema es llamado un modelo estático. Si el sistema es modelado estableciendo que la respuesta no solo depende de las entradas actuales, sino también de ciertas entradas y salidas retrasadas en el tiempo, el modelo es llamado dinámico .

En los sistemas recurrentes existen dos esquemas de identificación: un esquema en paralelo y un esquema serie-paralelo [24]. En el esquema paralelo (2.1) la salida del modelo de identificación es retroalimentada al modelo y se pueden definir como un modelo con recurrencia interna. Es utilizado principalmente como modelo de simulación para determinar el comportamiento de la planta (Figura 2.1). En el esquema serie-paralelo (2.2) la salida de la planta es retroalimentada al modelo y se puede definir como un modelo con

recurrencia externa, es utilizado principalmente como predictor de la salida de la planta (Figura 2.2). Los modelos son respectivamente de la forma

$$\hat{y}(k+1) = \hat{f}(\hat{y}(k),\ldots,\hat{y}(k-n+1);u(k),\ldots,u(k-m+1)) \tag{2.1}$$

$$\hat{y}(k+1) = \hat{f}(y(k),\ldots,y(k-n+1);u(k),\ldots,u(k-m+1)) \tag{2.2}$$

Donde:

$\hat{y}$	Salida estimada
y	Salida medida
u	Entrada de control
$\hat{f}$	Función de aproximación
k	Instante de muestreo
n	Número de salidas retroalimentadas
m	Número de entradas retroalimentadas

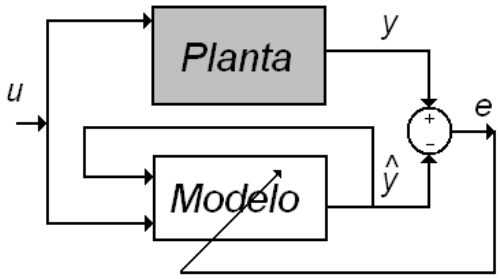

Figura 2.1 Identificación de un modelo recurrente paralelo

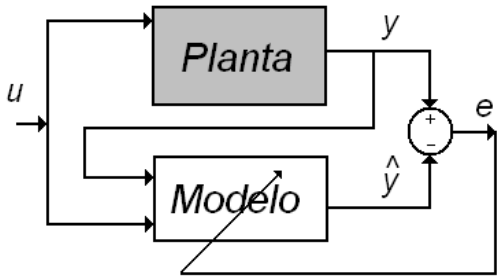

Figura 2.2 Identificación de un modelo recurrente serie-paralelo

Tarea 4: Elección de la estructura del modelo y sus parámetros. Esta tarea implica decidir que tipo de estructura debe tener el modelo para que puedan ser aplicables los

métodos de aprendizaje para obtener o actualizar los parámetros del modelo. De acuerdo a esto existen muchas opciones: polinomios, espacio de estados, ecuaciones diferenciales parciales, etc.

Tarea 5: Validación del modelo. Una vez realizadas las tareas anteriores y de obtener el modelo es necesario revisar el desempeño del sistema. El criterio específico para revisar el desempeño depende en gran medida del objetivo final que se tiene. La forma más fácil de validar la calidad de un modelo es primeramente con datos de entrenamiento y posteriormente con datos de prueba no presentados en el entrenamiento.

Estas son algunas tareas básicas que realizadas apropiadamente pueden llevar a obtener un modelo exitoso de un sistema dinámico.

2.2 Sistemas neurodifusos

Los sistemas neurodifusos forman parte de una nueva tecnología llamada computación flexible (*soft computing*), la cual tiene su origen en la teoría de la inteligencia artificial [19]. La computación flexible engloba un conjunto de técnicas comunes que tienen relación con el mundo real (por ejemplo: reconocimiento de formas, clasificación, toma de decisiones, etc.). En algunos casos, las técnicas de computación flexible pueden ser combinadas para aprovechar sus ventajas individuales, algunas de estas técnicas son:

- Lógica difusa.
- Redes neuronales.
- Algoritmos evolutivos o algoritmos genéticos.
- Teoría del caos

En los últimos años las investigaciones en redes neuronales y en sistemas difusos han mostrado grandes avances en el modelado de sistemas dinámicos debido a su propiedad de ser aproximadores universales con la condición de que estén disponibles suficientes neuronas ocultas o reglas difusas [8]. Hoy es ampliamente conocido que la lógica difusa ofrece un marco muy poderoso para el razonamiento aproximado que intenta modelar el proceso de razonamiento humano en un nivel cognoscitivo. Los sistemas difusos adquieren el conocimiento gracias al dominio del experto el cual es codificado dentro del algoritmo en términos de un conjunto de reglas difusas IF-THEN. Los sistemas difusos emplean este enfoque basado en reglas e interpolan el razonamiento para responder a entradas nuevas.

Una ventaja de usar un modelo difuso es que puede incluir el conocimiento a priori del sistema, pero no es posible aprender del sistema. En contraste, las redes neuronales ofrecen una arquitectura altamente estructurada, con capacidades de aprendizaje y generalización, que intenta imitar el mecanismo neurológico del cerebro. Una red neuronal almacena conocimiento de una manera distribuida con el ajuste de sus pesos, los cuales han sido determinados por medio del entrenamiento (aprendizaje) con datos conocidos. La habilidad de generalización para entradas nuevas está basada en la estructura algebraica inherente de la red neuronal. Una ventaja de usar un modelo neuronal es que tiene la capacidad de aprendizaje, pero no se puede incluir conocimiento a priori dentro del modelo.

Resultados recientes muestran que el procedimiento de fusión de estas dos técnicas parece ser muy efectivo para una amplia categoría de sistemas complejos no lineales, cuando no se tiene la información completa de la planta [14][19][24]. Los modelos neurodifusos cubren las desventajas de los sistemas difusos y neuronales, debido a que tienen la posibilidad de incluir conocimiento a priori y la capacidad de razonamiento y de aprender del sistema.

2.2.1 Sistemas difusos.

El sistema difuso es una estructura computacional basada en los conceptos de la teoría difusa, en reglas del tipo IF-THEN y en métodos de inferencia difusa. Los sistemas difusos, actualmente han encontrado diversas aplicaciones exitosas dentro de una gran variedad de áreas tales como el control automático, la clasificación de datos, el análisis de decisiones, los sistemas expertos, la predicción de series de tiempo, la robótica y en el reconocimiento de patrones [1][14][17][24]. A causa de su naturaleza multidisciplinaria, los sistemas difusos son conocidos como sistemas experto, modelos difusos o controladores lógicos difusos.

Existen muchas variantes de los modelos difusos para modelado de sistemas dinámicos desde principio de los años ochentas. En una manera general la configuración básica de un modelo difuso (Figura 2.3) tiene componentes básicos como *preprocesamiento de datos, difusión, base de reglas, mecanismo de inferencia, desdifusión y postprocesamiento*. [1].

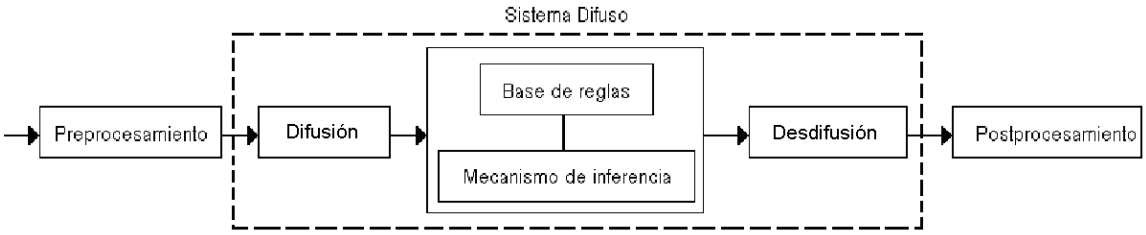

Figura 2.3 Estructura de un sistema difuso

Preprocesamiento de datos. Los valores físicos de las variables de entrada del sistema difuso pueden diferenciar significativamente en magnitud, por lo que es necesario mapear estos a dominios normalizados apropiados vía escalamiento (también para interpretabilidad), lo que permite trabajar con señales de la misma magnitud, lo que es deseable desde un punto de vista de modelado.

Difusión. La difusión mapea los valores normales de las entradas del modelo preprocesadas dentro de conjuntos difusos convenientes representados por funciones de membresía. El grado de membresía de una variable normal a un conjunto difuso es evaluado usando una función de membresía. Para cada variable de entrada se definen dos o más conjuntos difusos usualmente. La función de membresía gaussiana μ_{ij} (2.3) es una de las más empleadas para diseñar sistemas difusos esto debido a su propiedad de ser diferenciable, esta es la función de membresía de la *i* entrada de la *j* regla (Figura 2.4). Dado por

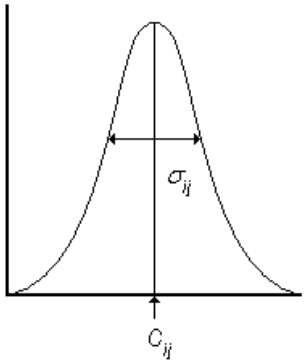

Figura 2.4 Función de membresía gaussiana

$$\mu_{ij}(x_i) = \exp\left\{-\frac{(x_i - c_{ij})^2}{2\sigma_{ij}^2}\right\} \tag{2.3}$$

Base de reglas. La base de reglas es la parte principal del sistema difuso. El conocimiento del experto es almacenado en una base de reglas difusas, esto a través de un número de reglas IF-THEN de forma general

$$IF \quad [antecedente] \quad THEN \quad [consecuente] \tag{2.4}$$

Siendo la parte consecuente la que diferencia la estructura de la base de reglas, la cual distingue los diferentes modelos difusos como son los modelos Mamdani y Takagi Sugeno Kang (TSK).

Mecanismo de inferencia. El mecanismo de inferencia o motor de inferencia es el método computacional que calcula el grado de disparo de cada regla para un patrón dado de entrada difuso. El grado de disparo o fuerza de disparo w_j de la *j* regla es determinado por el mecanismo que es usado para implementar la expresión de la regla difusa antecedente, un ejemplo de este mecanismo es el producto de los grados de membresía.

$$w_j = \prod_{i=1}^{n} \mu_{ij} \tag{2.5}$$

Donde μ_{ij} define la función de membresía de la entrada *i* usada en la regla *j*, siendo *n* variables de entrada.

Desdifusión. Un desdifusor compila la información provista por cada una de las reglas y toma una decisión que aplica a la parte consecuente. El método del *centroide* es uno de los más empleados como desdifusor (2.6). Los modelos difusos TSK [21] son los más empleados para el modelado y control de sistemas dinámicos (2.7).

$$y = \frac{\sum_{j=1}^{r} w_j f_j(x)}{\sum_{j=1}^{r} w_j} \tag{2.6}$$

Donde *r* es el número de reglas difusas y $f_j(x)$ es la función de la parte consecuente. Este método de desdifusión toma la suma pesada de la parte consecuente $f_j(x)$ de las reglas de acuerdo al grado de disparo.

Postprocesamiento. Este paso proporciona la salida del sistema en señal normal. Comúnmente un escalamiento de la salida.

Uno de los modelos difusos más empleados es el sistema difuso TSK [21] que ha provisto una solución eficiente y poderosa para el modelado de sistemas no lineales. Debido a que su estructura de modelos múltiples da la capacidad de aproximar modelos dinámicos con incertidumbres y fuertes no linealidades en un conjunto compacto y con alguna exactitud [24]. Una regla difusa en el modelo TSK tiene la forma:

$$if \quad x \quad is \quad A \quad \& \quad y \quad is \quad B \quad then \quad z = f(x, y) \tag{2.7}$$

Donde A y B son conjuntos difusos en la parte antecedente, mientras $z = f(x, y)$ es una función común en la parte consecuente. Usualmente $f(x, y)$ es un polinomio lineal con variables de entrada x y y, pero este puede ser cualquier función mientras pueda describir apropiadamente la salida del modelo dentro de la región difusa especificada por la regla del antecedente. Cuando $f(x, y)$ es un polinomio de primer orden, el sistema resultante de inferencia difusa es llamado modelo difuso TSK de primer orden. Cuando f es una constante, entonces tenemos un modelo difuso TSK de orden cero, el cual puede ser visto como un caso especial de la regla base de Mamdani, o un caso especial del modelo Tsukamoto [24].

El sistema TSK fue propuesto para modelar procesos dinámicos en tiempo continuo, y fue desarrollado posteriormente por Sugeno y Kang [20]. La versión en tiempo discreto del modelo TSK fue analizada por Tanaka y Sugeno [22][23].

Tanaka parte de que se puede construir un modelo TSK sí la descripción local de la planta esta disponible en términos de modelos locales lineales dado por r reglas difusas

$$x(k+1) = A_j x(k) + B_j u(k) \quad j = 1,2,\ldots,r \tag{2.8}$$

Donde el vector de estado $x(k) \in R^n$, la entrada de control $u(k) \in r^m$, y las matrices A_j y B_j son de dimensiones apropiadas.

La regla base IF-THEN tiene una parte antecedente con estados como entradas y funciones de membresía gaussianas $\mu_{ij}(x)$ y una parte consecuente que es un modelo local lineal en variables de estado.

La j regla tiene la forma siguiente:

$$\begin{aligned} &regla \quad j: \quad if \quad x_1(k) \quad is \quad \mu_{1j} \quad \& \quad \ldots \quad \& \quad x_n(k) \quad is \quad \mu_{nj} \\ &then \quad x^j(k+1) = A_j x(k) + B_j u(k) \end{aligned} \tag{2.9}$$

Con $j = 1, 2, \ldots, r$

El mecanismo de inferencia del modelo difuso TSK es el producto de los grados de membresía de cada entrada del sistema difuso con grado de disparo w_j (2.10).

$$w_j = w_j(x) = \prod_{i=1}^{n} \mu_{ij}(x_i) \tag{2.10}$$

El defusificador del modelo difuso TSK más común es el centroide $x(k+1)$, visto como la salida del sistema difuso TSK (2.11).

$$x(k+1) = \frac{\sum_{j=1}^{r} w_j \left(A_j x(k) + B_j u(k)\right)}{\sum_{j=1}^{r} w_j} \tag{2.11}$$

Representando este sistema como un sistema en variables de estado con matrices dinámicas (2.12).

$$x(k+1) = \sum_{j=1}^{r} \alpha_j \left(A_j x(k) + B_j u(k)\right) = \sum_{j=1}^{r} \alpha_j x^j(k) \tag{2.12}$$

$$\alpha_j = \frac{w_j}{\sum_{j=1}^{r} w_j} \tag{2.13}$$

Aquí α_j (2.13) es visto como un componente de normalización de la salida, teniendo la propiedad de que para cada regla difusa tiene un valor entre 0 y 1 (2.14) y la suma de todos los componentes es igual a uno (2.15) con $j = 1, 2, \ldots, r$.

$$\alpha = [\alpha_1 \quad \alpha_2 \quad \ldots \quad \alpha_r]^T \in [0,1]^r \tag{2.14}$$

$$\sum_{j=1}^{r} \alpha_j = 1 \tag{2.15}$$

2.2.2 Redes neuronales

Las Redes Neuronales Artificiales (RNAs) fueron originalmente motivadas por la estructura biológica del cerebro humano, que es extremadamente poderoso para algunas tareas como procesamiento de información, aprendizaje y adaptación [14]. Las redes neuronales son usadas para resolver una variedad de problemas, incluyendo aspectos del control de sistemas dinámicos complejos. Con el control estándar, se requiere un modelo

matemático; con control difuso, se usa un conjunto de reglas. Cuando la información disponible sobre la conducta de un sistema es principalmente de datos numéricos, la metodología de redes neuronales es muy útil.

En la literatura de redes neuronales existen muchas definiciones de una red neuronal, pero difieren del tipo de red, para muchos tipos de redes neuronales es aceptable la siguiente definición: Es un sistema de elementos de procesamiento simple (neuronas) que son conectadas dentro de una red por un conjunto de pesos (sinapsis). La función de la red es determinada por la arquitectura de la red, la magnitud de los pesos y el modo de operación de los elementos de procesamiento [16].

Las características más importantes de las redes neuronales son:

- Un gran número de unidades simples
- Unidades altamente paralelas
- Unidades fuertemente interconectadas
- Robustez contra fallas de unidades simples
- Aprendizaje desde datos

McCulloch y Pitts [14] construyeron un modelo básico de neurona artificial, como un elemento simple a base de un sumador y una función de activación (Figura 2.5). Las conexiones de entrada x_i de la neurona son pesadas por los w_{ij} y sumados, la neurona es evaluada en la función de activación φ dando como resultado la salida de la neurona siempre y cuando esta suma supere un valor de umbral θ_j (2.16).

$$y_j = \varphi\left(\sum_{i=0}^{n} w_{ij} x_i + \theta_j\right) \tag{2.16}$$

Donde w_{ij} es el peso que pondera la entrada x_i de la neurona j y θ_j es el umbral. φ es la función de activación de la neurona y n es el número total de pesos conectados a la entrada de la neurona.

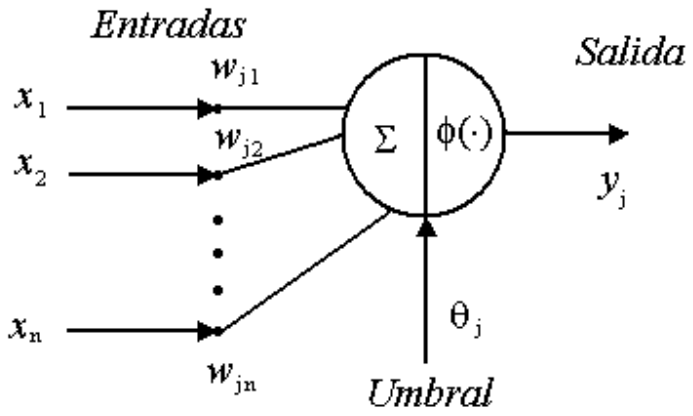

Figura 2.5 Representación de una neurona artificial

Como función de activación se puede usar muchas funciones, pero las más usadas para los casos no lineales son las sigmoides, aunque existen también funciones lineales, estas tienen como característica que son completamente diferenciables en todo su dominio y se usan principalmente en redes neuronales prealimentadas (feedforward neural networks) [16].

Si bien la neurona es la unidad principal, no menos importante es como se interconectan entre ellas para formar una red que procese la información. La *arquitectura de la red* es la forma en que se conectar las unidades básicas. Puede existir un sin número de posibles combinaciones de las conexiones entre ellas, sin embargo se pueden definir algunas arquitecturas fundamentales.

La arquitectura fundamental más común en la literatura es la *red perceptron multicapa* (*MLP, Multilayer Perceptron*) [16]. Dentro de estas se han derivado múltiples arquitecturas, la más conocida es la *red neuronal feedforward* [16] está construida ordenando las neuronas en capas, dejando que cada neurona en una capa tome como entrada solo las salidas de las neuronas de la capa previa o entradas externas. Un ejemplo es la arquitectura de una red feedforward que contiene 2 capas ocultas y una capa de salida Figura 2.6. Normalmente, las entradas se conectan a la primera capa oculta y se dice que esta red esta *conectada completamente* debido a que todas las entradas ó todas las neuronas en una capa están conectadas a todas las neuronas de la siguiente capa.

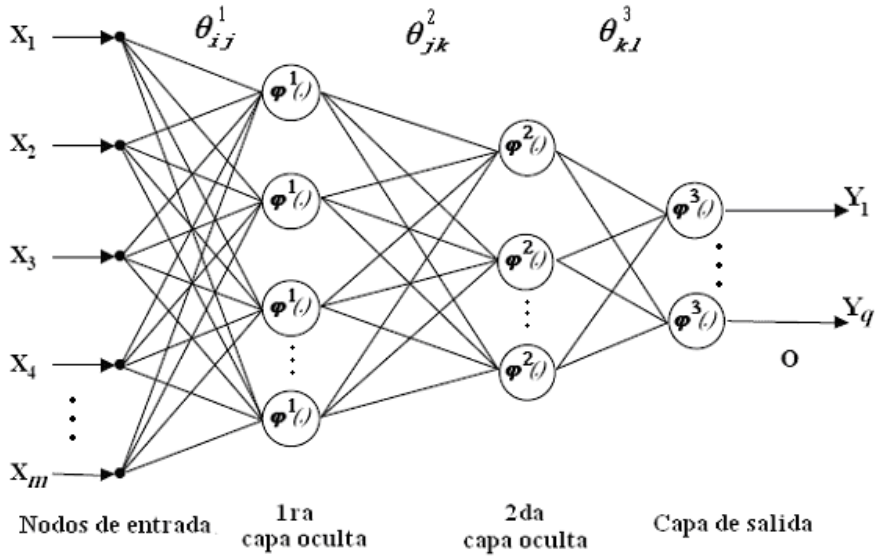

Figura 2.6 Red neuronal feedforward

En las redes neuronales los *pesos* $\{\theta_{ij}^1 \quad \theta_{jk}^2 \quad \theta_{kl}^3\}$ son los que contienen la información que se obtienen a partir de los datos. La fórmula matemática que expresa la red feedforward de 2 capas ocultas tiene la forma:

$$\hat{y}_l(k) = F[x,\theta] = \varphi_l^3\left\{\sum_{k=1}^{p}\theta_{kl}^3\varphi_k^2\left[\sum_{j=1}^{n}\theta_{jl}^2\varphi_j^1\left(\sum_{i=1}^{m}\theta_{ij}^1 x_i\right)\right]\right\} \tag{2.17}$$

θ especifica el vector paramétrico, que contiene todos los parámetros ajustables de la red y $\varphi(\cdot)$ es la función de activación de cada neurona. Se tienen *m* entradas, *n* neuronas en la primera capa oculta con pesos θ^1, *p* neuronas en la segunda capa oculta con pesos θ^2 y *q* neuronas en la capa de salida con conexiones θ^3.

Para determinar los valores de los pesos se deben tener un conjunto de ejemplos de cómo las salidas $\hat{y}_l$ deben relacionar a las entradas x_j. La tarea de determinar los pesos desde los ejemplos es llamado *entrenamiento* o *aprendizaje*, es básicamente un problema de estimación convencional. Esto es, los pesos son estimados desde los ejemplos de tal forma que la red, de acuerdo a una métrica, modela la relación entrada-salida tan precisamente como sea posible [15].

Los algoritmos de entrenamiento más utilizados para el aprendizaje de las redes feedforward son los *algoritmos de retropropagación* (*Backpropagation*) que son algoritmos de entrenamiento iterativo para neuronas con funciones de activación diferenciables [17]. Estos algoritmos son llamados así porque retro propagan el error entre las salidas deseadas y actuales de la red neuronal con el propósito de ajustar los pesos para reducir el error. Un algoritmo de retropropagación típico consiste de dos pasos: un paso hacia delante, las entradas son aplicadas a la red, y las salidas actuales de la red son calculadas. Luego, el error entre las salidas actuales y deseadas es calculado. En el paso hacia atrás el gradiente del error con respecto a los pesos de la red es calculado para propagar el error al revés a través de la red. Una vez que el gradiente del error es calculado, los pesos son ajustados usando, por ejemplo, un método de gradiente descendiente [19].

El algoritmo de retropropagación más sencillo es el método del descenso en la mayor inclinación (steepest descent). En este algoritmo se define una funcional de costo $J(\cdot)$ que se utiliza para definir el método de optimización; puede ser por ejemplo el error cuadrático instantáneo (2.18).

$$J(\theta) = \frac{1}{2}\left(\hat{y}_i(k) - y_i(k)\right)^2 = \frac{1}{2}\left(\hat{y}_i(k,\theta) - y_i(k)\right)^2 \tag{2.18}$$

Donde θ son los parámetros a estimar, y_i es la salida deseada y $\hat{y}_i$ es la salida de la red en el instante *k*. En general, el gradiente se entiende como la derivada parcial del funcional de costo con respecto a los parámetros (2.19).

$$\nabla J(\theta) = \frac{\partial J(\theta)}{\partial \theta} \tag{2.19}$$

El objetivo del algoritmo es que a cada tiempo *k*, se varíen los parámetros con una razón de aprendizaje (η) en la dirección del gradiente rotado $\nabla J(\theta)$ para obtener el valor de θ en el tiempo *k*+1 (2.20). La razón de aprendizaje η tiene un valor fijo η_0.

$$\begin{aligned} &\theta(k+1) = \theta(k) - \eta R(k)\frac{\partial J(\theta)}{\partial \theta} \\ &R(k) = I \\ &\eta = \eta_0 \end{aligned} \tag{2.20}$$

Con lo que se busca que en cada paso de tiempo se reduzca el valor de la funcional de costo. La suposición general en este método es que el funcional de costo es convexo con respecto a los parámetros. Si el sistema es no lineal no es posible garantizar esta suposición por lo que puede que existan mínimos locales y la búsqueda pueda quedar atrapada en uno de ellos si se inicia la búsqueda muy cerca de este.

Existen otros métodos para el entrenamiento de redes MLP basados en el gradiente como el método de Newton o método de gradiente conjugado, que básicamente difieren en la forma de definir *R(k)* y η [15].

Las redes MLP no necesariamente tiene la estructura feedforward mostrada en la Figura 2.6. En si, en el contexto de modelos para sistemas dinámicos es común ver que la arquitectura de red MLP es aumentada con lazos retroalimentados (Figura 2.7), en este caso la red es referida como una *red neuronal recurrente*. Contrario a una red feedforward, donde hay una relación algebraica entre la entrada y la salida, la red recurrente contiene memoria, es decir, es un sistema dinámico. La manera matemática que expresa la red recurrente tiene la forma:

$$\hat{y}_l(k|\theta) = F[x(k),\theta,k] = \varphi_l^3\left\{\sum_{k=1}^{p}\theta_{kl}^3\varphi_k^2(\cdot,k)\right\} \tag{2.21}$$

Desglosando esta ecuación (2.21) en la capa de retroalimentación aparece un término que depende de un instante anterior $\varphi_j^1(\cdot,k-1)$ (2.22).

$$\hat{y}_l(k|\theta) = \varphi_l^3\left\{\sum_{k=1}^{p}\theta_{kl}^3\varphi_k^2\left[\sum_{j=1}^{n}\theta_{jl}^2\varphi_j^1(\cdot,k)+\sum_{j=1}^{n}\theta_{jl}^2\varphi_j^1(\cdot,k-1)\right]\right\} \tag{2.22}$$

Desarrollando la función retrasada en el tiempo se obtiene una expresión que depende de la entrada actual y una entrada retrasada un instante *k* (2.23).

$$\hat{y}_l(k|\theta) = \varphi_l^3\left\{\sum_{k=1}^{p}\theta_{kl}^3\varphi_k^2\left[\sum_{j=1}^{n}\theta_{jl}^2\varphi_j^1\left(\sum_{i=1}^{m}\theta_{ij}^1x_i(k)\right)+\sum_{j=1}^{n}\theta_{jl}^2\varphi_j^1\left(\sum_{i=1}^{m}\theta_{ij}^1x_i(k-1)\right)\right]\right\} \tag{2.23}$$

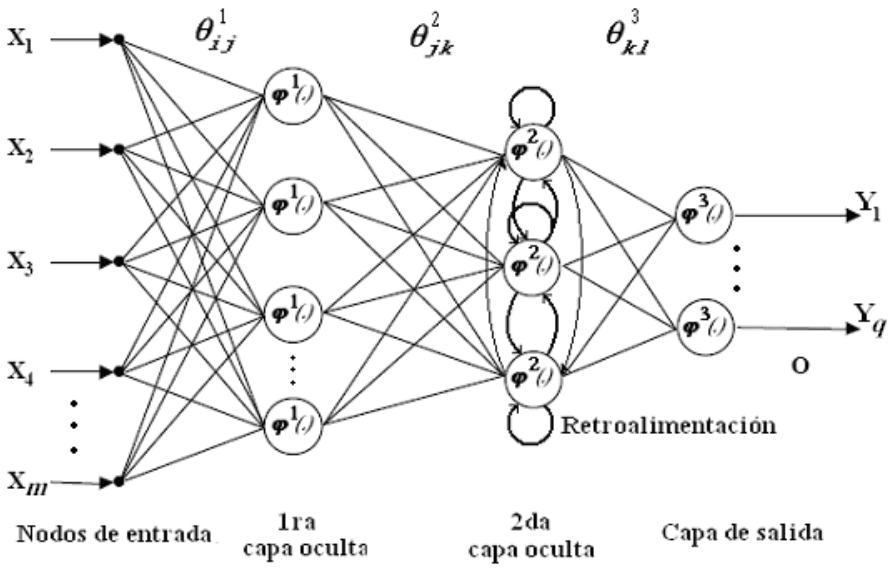

Figura 2.7 Red neuronal recurrente

La recursividad puede ser implementada en muchas formas, la estructura mostrada en la Figura 2.7 es solo un ejemplo. Pero puede haber también retroalimentación de las salidas como otra variante. Entre las estructuras más conocidas están los modelos NOE[1] y NARMAX[2] [15].

El entrenamiento de estructuras feedforward es equivalente al entrenamiento de modelos estáticos. Pero, entrenar estructuras recurrentes es más complicado debido a que las retroalimentaciones tienen que ser tomadas en cuenta. En particular, entrenar modelos recurrentes es siempre un problema de optimización no lineal independientemente de sí la arquitectura del modelo recurrente utilizado es lineal o no lineal en los parámetros [15]. Básicamente, 2 estrategias para el entrenamiento de estructuras recurrentes pueden distinguirse, ambas basadas en el algoritmo de retropropagación para redes feedforward. Pero contrario a los modelos estáticos, el cálculo del gradiente de modelos recurrentes depende de estados pasados del modelo. Los dos algoritmos generales utilizados para entrenar redes recurrentes son: el algoritmo de *retropropagación a través del tiempo* (BPTT, *Backpropagation through time*) y el algoritmo de *aprendizaje recurrente en tiempo*

[1] Por sus siglas en ingles: Nonlinear Output Error

[2] Por sus siglas en ingles: Nonlinear AutoRegressive, Moving Average, eXternal input

real (*real time recurrent learning*) [15]. Estos difieren en la forma de procesar la información sobre los estados anteriores de la red.

El algoritmo de retropropagación a través del tiempo [18] es aplicado a estructuras de modelos recurrentes desdoblando el modelo a través del pasado, donde el cálculo del gradiente del error es desglosado no solo en el instante de tiempo actual, sino se toman en cuenta los retrasos de tiempo de las retroalimentaciones internas y/o externas de la red recurrente. Por ejemplo, suponiendo un modelo recurrente entrada-salida y que los parámetros no dependen de la entrada.

$$\hat{y}(k) = f\left(\theta(k), \hat{y}(k-1), \ldots, \hat{y}(k-r), u(k-1)\right) \tag{2.24}$$

Es decir, un sistema recurrente con una entrada y r salidas retrasadas (retroalimentación), θ es el vector de parámetros. Como en el algoritmo de retropropagación para redes feedforward se requiere un funcional de costo (2.18), para calcular el gradiente con respecto a los parámetros del modelo. Este gradiente depende de las derivadas de la salida del modelo con respecto a los parámetros θ (2.25).

$$\frac{\partial \hat{y}(k)}{\partial \theta(k)} = \frac{\partial f(\cdot)}{\partial \theta(k)} + \frac{\partial f(\cdot)}{\partial \hat{y}(k-1)} \frac{\partial \hat{y}(k-1)}{\partial \theta(k)} + \cdots + \frac{\partial f(\cdot)}{\partial \hat{y}(k-r)} \frac{\partial \hat{y}(k-r)}{\partial \theta(k)} \tag{2.25}$$

El primer término corresponde al gradiente de la salida del algoritmo de retropropagación estándar, los siguientes términos corresponden a los gradientes de cada una de las conexiones de retroalimentación de salida del modelo. La evaluación de los términos adiciones requiere el cálculo de las derivadas de la salida del modelo retrasadas r muestras previas con respecto a los parámetros actuales (2.26).

$$\begin{aligned} &\hat{y}(k-1) = f\left(\theta(k), \hat{y}(k-2), \ldots, \hat{y}(k-r-1), u(k-2)\right) \\ &\vdots \\ &\hat{y}(k-r) = f\left(\theta(k), \hat{y}(k-r-1), \ldots, \hat{y}(k-2r), u(k-r)\right), \end{aligned} \tag{2.26}$$

donde cada una de las derivadas del modelo dependen de derivadas previas (2.27).

$$\begin{aligned}
&\frac{\partial \hat{y}(k-1)}{\partial \theta(k)} = \frac{\partial f(\cdot)}{\partial \theta(k)} + \frac{\partial f(\cdot)}{\partial \hat{y}(k-2)} \frac{\partial \hat{y}(k-2)}{\partial \theta(k)} + \cdots + \frac{\partial f(\cdot)}{\partial \hat{y}(k-r-1)} \frac{\partial \hat{y}(k-r-1)}{\partial \theta(k)} \\
&\vdots \\
&\frac{\partial \hat{y}(k-r)}{\partial \theta(k)} = \frac{\partial f(\cdot)}{\partial \theta(k)} + \frac{\partial f(\cdot)}{\partial \hat{y}(k-r-1)} \frac{\partial \hat{y}(k-r-1)}{\partial \theta(k)} + \cdots + \frac{\partial f(\cdot)}{\partial \hat{y}(k-2r)} \frac{\partial \hat{y}(k-2r)}{\partial \theta(k)}
\end{aligned} \tag{2.27}$$

Si se sigue calculando estas derivadas se pueden realizar hasta el instante k=0 con un valor inicial y_0 (2.28).

$$\hat{y}(0) = y_0 \tag{2.28}$$

Donde dicha derivada no depende de los parámetros (2.29).

$$\frac{\partial \hat{y}(0)}{\partial \theta(k)} = 0 \tag{2.29}$$

Este método tiene la ventaja de calcular de forma exacta cada una de las derivadas $\partial \hat{y}(\cdot)/\partial \theta(k)$ a través del tiempo. Sin embargo, el algoritmo tiene que ser repetido para todos los instantes de tiempo $k = 1,2,\ldots,N$ donde N es el número de datos de entrenamiento. Esto significa que para el último dato de entrenamiento $k = N$, tienen que ser calculadas $N \cdot r$ derivadas para cada parámetro. Debido a que el algoritmo es iterativo y el número de datos de entrenamiento N es grande, el costo computacional es demasiado y los requerimientos de memoria altos que hace del algoritmo poco práctico.

Debido a los defectos de costo computacional y memoria presentes en el algoritmo BPTT se desarrolló el algoritmo de *aprendizaje recurrente en tiempo real* (*Real Time Recurrent Learning RTRL*) que es mucho más eficiente que BPTT puesto que evita desdoblar el modelo hacia el pasado [26] . El algoritmo de aprendizaje recurrente en tiempo real considera que los parámetros del modelo no cambian durante un entrenamiento completo de los datos de entrenamiento, esto es, $\theta(k) = \theta(k-1) = \ldots = \theta(1)$, esta condición es cubierta para una adaptación *batch* (cada época) donde los parámetros se actualizan una vez que se ha simulado el modelo para todos los datos de entrenamiento, pero no es satisfecha para una adaptación por muestras donde en cada instante de tiempo se actualizan los parámetros [15]. Sin embargo, para aprendizaje recurrente en tiempo real en modo muestreo se hace la suposición de que los cambios en los parámetros son despreciados, es

decir, $\theta(k) \approx \theta(k-1) \approx \ldots \approx \theta(1)$, esto se logra cuando la razón de aprendizaje es pequeña. Con esta suposición las derivadas obtenidas en el gradiente para el algoritmo BPTT debidas a las retroalimentaciones, son aproximadamente equivalentes (2.30).

$$\frac{\partial \hat{y}(k-1)}{\partial \theta(k)} = \frac{\partial \hat{y}(k-1)}{\partial \theta(k-1)} \cdots \frac{\partial \hat{y}(k-r)}{\partial \theta(k)} = \frac{\partial \hat{y}(k-r)}{\partial \theta(k-r)} \tag{2.30}$$

Con estas derivadas el gradiente del algoritmo BPTT puede ser reescrito, dejando las derivadas solo en función del cálculo de la derivada anterior.

$$\frac{\partial \hat{y}(k)}{\partial \theta(k)} = \frac{\partial f(\cdot)}{\partial \theta(k)} + \frac{\partial f(\cdot)}{\partial \hat{y}(k-1)} \frac{\partial \hat{y}(k-1)}{\partial \theta(k-1)} + \cdots + \frac{\partial f(\cdot)}{\partial \hat{y}(k-r)} \frac{\partial \hat{y}(k-r)}{\partial \theta(k-r)} \tag{2.31}$$

La propiedad explotada por el algoritmo de aprendizaje en tiempo real es que las derivadas equivalentes son los gradientes del modelo previo, que fueron obtenidas en el instante *k-1*. Se debe tener cuidado de que el gradiente durante el aprendizaje cumpla con las cotas del gradiente (2.32) porque en otro caso, la actualización del gradiente seria inestable. Por lo que hay que asegurarse de que estas no excedan el valor de uno, si lo hacen se puede reducir el valor de η la razón de aprendizaje como una opción para continuar con el entrenamiento.

$$\left|\partial f(\cdot) / \partial \hat{y}(k-1)\right| < 1 \quad \ldots \quad \left|\partial f(\cdot) / \partial \hat{y}(k-r)\right| < 1 \tag{2.32}$$

Igual que en el algoritmo BPTT las derivadas del modelo son iguales a cero para valores iniciales y_0 (2.28), que no depende de los parámetros (2.29).

Comparado con BPTT, el aprendizaje recurrente en tiempo real es mucho más simple y rápido, requiere menos memoria, y lo más importante su complejidad no depende del número de muestras para el entrenamiento. Aunque en comparación con retropropagación estática, requiere un poco más de demanda computacional y de implementación.

2.2.3 Sistemas neurodifusos

Para el diseño de un sistema difuso, se requiere de un buen conocimiento del sistema obtenido por los expertos y representado por las reglas difusas, pero debido a que los

sistemas tienen componentes dinámicas y los conocimientos de los expertos no son exactos, esto resulta en mucho tiempo en el diseño y sintonización de las funciones de membresía, surgió la idea de fusionar las redes neuronales y los sistemas difusos en un sistema que permita representar el conocimiento de los expertos en una representación que tenga una interpretación lógica y permita realizar el aprendizaje del sistema.

Las redes neurodifusas son modelos difusos que no solamente son diseñados por expertos sino al menos parcialmente aprenden desde los datos. Normalmente, los modelos difusos son representados en una estructura neuronal, y los métodos de aprendizaje ya establecidos en el contexto de las redes neuronales son aplicados a las redes neurodifusas.

La red neurodifusa ANFIS (Adaptive Neural Fuzzy Inference Systems) diseñada por Jang [5] es una de las más conocidas en la comunidad, y permite dar una explicación de cómo funcionan los sistemas neurodifusos. El sistema se basa en un modelo difuso TSK con dos entradas $x=(x_1,x_2)$, con un número de reglas fijo, por ejemplo dos reglas, de la siguiente forma:

$$\begin{aligned} R_1:&\quad x_1(k)\quad es\quad \mu_{11}\quad \&\quad x_2(k)\quad es\quad \mu_{21}\quad entonces\quad y=f_1(x)\\ R_2:&\quad x_1(k)\quad es\quad \mu_{12}\quad \&\quad x_2(k)\quad es\quad \mu_{22}\quad entonces\quad y=f_2(x)\end{aligned} \tag{2.33}$$

Donde μ son funciones de membresía difusas, con parte antecedente funciones polinomiales de primer orden (2.34).

$$\begin{aligned} f_1(x)&=z_{11}x_1+z_{21}x_2+z_{01}\\ f_2(x)&=z_{12}x_1+z_{22}x_2+z_{02}\end{aligned} \tag{2.34}$$

Con esto el sistema ANFIS produce la salida del modelo siguiente:

$$\hat{y}=\frac{\mu_{11}(x_1)\mu_{21}(x_2)f_1(x)+\mu_{12}(x_1)\mu_{22}(x_2)f_2(x)}{\mu_{11}(x_1)\mu_{21}(x_2)+\mu_{12}(x_1)\mu_{22}(x_2)}=\frac{\sum_{j=1}^{2}\left(\prod_{i=1}^{2}\mu_{ij}(x_i)f_j(x)\right)}{\sum_{j=1}^{2}\prod_{i=1}^{2}\mu_{ij}(x_i)} \tag{2.35}$$

Simplificando la salida estimada se definen *a* y *b* (2.36) para un mejor entendimiento en la obtención de las leyes de entrenamiento.

$$a = \sum_{j=1}^{2} \left(\prod_{i=1}^{2} \mu_{ij}(x_i) f_j(x) \right)$$
$$b = \sum_{j=1}^{2} \prod_{i=1}^{2} \mu_{ij}(x_i) \tag{2.36}$$

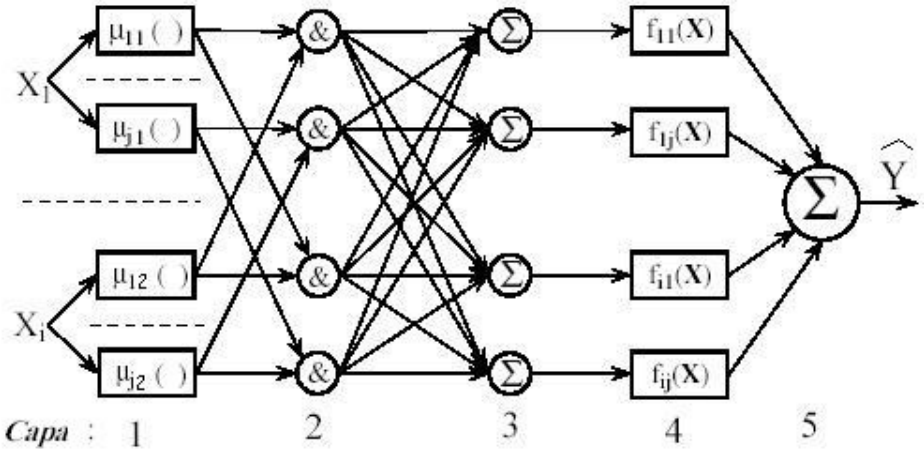

Figura 2.1 Estructura ANFIS

El sistema ANFIS es visto como una estructura neuronal como en la Figura 2.1 donde $\mu_{ij}(x_i)$ (2.37) es la función de membresía de tipo gaussiana de la entrada *i* con respecto a la regla *j*.

$$\mu_{ij}(x_i) = \exp\left\{ -\frac{(x_i - c_{ij})^2}{2\sigma_{ij}^2} \right\} \tag{2.37}$$

En una estructura ANFIS los parámetros (c_{ij}, σ_{ij}) de la parte antecedente, junto con los parámetros (z_{ij}, z_{0j}) de la parte consecuente juegan el papel de los pesos de una red neuronal. El algoritmo de aprendizaje utilizado para ajustar los pesos es el algoritmo de retropropagación para redes feedforward, la funcional de costo a minimizar es el error cuadrático instantáneo definido como

$$J = \frac{1}{2}(\hat{y} - y)^2 \tag{2.38}$$

Donde *y* es la salida del sistema a identificar y $\hat{y}$ es la salida del modelo ANFIS, los pesos a adaptar son $w = \{z_{ij}, z_{0j}, c_{ij}, \sigma_{ij}\}$ usando el algoritmo del gradiente se tiene la ley de adaptación (2.39) para cada uno de los parámetros de *w*.

$$w(k+1) = w(k) - \eta \nabla J(w(k)) \tag{2.39}$$

Para poder utilizar el algoritmo es necesario calcular las derivadas parciales de *J* con respecto a cada uno de los elementos de *w*. Para esto primero fijamos los índices *i* y *j*.

Primero se obtienen las leyes de aprendizaje para la parte consecuente, se establece la ley de aprendizaje con respecto a z_{ij} como

$$z_{ij}(k+1) = z_{ij}(k) - \eta \frac{\partial J(w)}{\partial z_{ij}} \tag{2.40}$$

Obteniendo la derivada parcial de *J* con respecto a z_{ij}

$$\frac{\partial J(w)}{\partial z_{ij}} = \frac{\partial}{\partial z_{ij}} \left(\frac{1}{2} (\hat{y} - y)^2 \right) \tag{2.41}$$

$$\frac{\partial J(w)}{\partial z_{ij}} = (\hat{y} - y) \frac{\partial \hat{y}}{\partial z_{ij}} \tag{2.42}$$

$$\frac{\partial J(w)}{\partial z_{ij}} = \frac{(\hat{y} - y)}{b} \frac{\partial (a)}{\partial z_{ij}} \tag{2.43}$$

$$\frac{\partial J(w)}{\partial z_{ij}} = \frac{(\hat{y} - y)}{b} \prod_{i=1}^{2} \mu_{ij}(x_i) \frac{\partial f_j(x)}{\partial z_{ij}} \tag{2.44}$$

$$\frac{\partial J(w)}{\partial z_{ij}} = \frac{(\hat{y} - y)}{b} \prod_{i=1}^{2} \mu_{ij}(x_i) x_i \tag{2.45}$$

Ahora bien, se obtiene la ley de aprendizaje con respecto a z_{0j}

$$z_{0j}(k+1) = z_{0j}(k) - \eta \frac{\partial J(w)}{\partial z_{0j}} \tag{2.46}$$

Obteniendo la derivada parcial de *J* con respecto a z_{0j}

$$\frac{\partial J(w)}{\partial z_{0j}} = \frac{\partial}{\partial z_{0j}}\left(\frac{1}{2}(\hat{y}-y)^2\right) = (\hat{y}-y)\frac{\partial \hat{y}}{\partial z_{0j}} \tag{2.47}$$

$$\frac{\partial J(w)}{\partial z_{0j}} = \frac{(\hat{y}-y)}{b}\frac{\partial(a)}{\partial z_{0j}} \tag{2.48}$$

$$\frac{\partial J(w)}{\partial z_{0j}} = \frac{(\hat{y}-y)}{b}\prod_{i=1}^{2}\mu_{ij}(x_i)\frac{\partial f_j(x)}{\partial z_{0j}} \tag{2.49}$$

$$\frac{\partial J(w)}{\partial z_{0j}} = \frac{(\hat{y}-y)}{b}\prod_{i=1}^{2}\mu_{ij}(x_i) \tag{2.50}$$

Calculando las leyes de aprendizaje de los pesos antecedentes del sistema difuso, se obtiene la ley de aprendizaje para c_{ij}

$$c_{ij}(k+1) = c_{ij}(k) - \eta\frac{\partial J(w)}{\partial c_{ij}} \tag{2.51}$$

Obteniendo la derivada parcial de J con respecto a c_{ij}

$$\frac{\partial J(w)}{\partial c_{ij}} = \frac{\partial}{\partial c_{ij}}\left(\frac{1}{2}(\hat{y}-y)^2\right) = (\hat{y}-y)\frac{\partial \hat{y}}{\partial c_{ij}} \tag{2.52}$$

$$\frac{\partial J(w)}{\partial c_{ij}} = (\hat{y}-y)\frac{\frac{\partial(a)}{\partial c_{ij}}b - \frac{\partial(b)}{\partial c_{ij}}a}{b^2} \tag{2.53}$$

$$\frac{\partial J(w)}{\partial c_{ij}} = (\hat{y}-y)\left(\frac{f_j b - a}{b^2}\right)\prod_{i=1}^{2}\mu_{ij}(x_i)\frac{\partial}{\partial c_{ij}}\left(\frac{(x_i-c_{ij})^2}{2\sigma_{ij}^2}\right) \tag{2.54}$$

$$\frac{\partial J(w)}{\partial c_{ij}} = (\hat{y}-y)\left(\frac{f_j b - a}{b^2}\right)\prod_{i=1}^{2}\mu_{ij}(x_i)\frac{(x_i-c_{ij})}{\sigma_{ij}^2} \tag{2.55}$$

Por último se desarrolla la ley de aprendizaje para σ_{ij}

$$\sigma_{ij}(k+1) = \sigma_{ij}(k) - \eta\frac{\partial J(w)}{\partial \sigma_{ij}} \tag{2.56}$$

Obteniendo la derivada parcial de J con respecto a σ_{ij}

$$\frac{\partial J(w)}{\partial \sigma_{ij}} = \frac{\partial}{\partial \sigma_{ij}}\left(\frac{1}{2}(\hat{y}-y)^2\right) = (\hat{y}-y)\frac{\partial \hat{y}}{\partial \sigma_{ij}} \tag{2.57}$$

$$\frac{\partial J(w)}{\partial \sigma_{ij}} = (\hat{y}-y)\frac{\frac{\partial(a)}{\partial \sigma_{ij}}b - \frac{\partial(b)}{\partial \sigma_{ij}}a}{b^2} \tag{2.58}$$

$$\frac{\partial J(w)}{\partial \sigma_{ij}} = (\hat{y}-y)\left(\frac{f_j b - a}{b^2}\right)\prod_{i=1}^{2}\mu_{ij}(x_i)\frac{\partial}{\partial \sigma_{ij}}\left(\frac{(x_i - c_{ij})^2}{2\sigma_{ij}^2}\right) \tag{2.59}$$

$$\frac{\partial J(w)}{\partial \sigma_{ij}} = (\hat{y}-y)\left(\frac{f_j b - a}{b^2}\right)\prod_{i=1}^{2}\mu_{ij}(x_i)\frac{(x_i - c_{ij})^2}{\sigma_{ij}^3} \tag{2.60}$$

Las leyes de aprendizaje (2.45, 2.50, 2.55, 2.60) permiten la actualización del sistema ANFIS.

2.2.4 Sistemas autoorganizados neurodifusos

Con ANFIS, el número de reglas difusas es definido por adelantado, y los parámetros de la parte consecuente y antecedente son aprendidos desde los datos disponibles. En algunos casos para los expertos no es fácil definir el número de reglas difusas apropiado para el sistema difuso, por lo que las reglas difusas pueden ser definidas por algún método de agrupamiento (*clustering*) , a este tipo de sistemas es llamado sistemas neurodifusos autoorganizados [10][11][13][27].

El objetivo del método de agrupamiento es formar grupos de datos que son similares en alguna característica particular. Cada grupo representa una regla difusa IF-THEN, donde las funciones de membresía inician con información obtenida de los datos.

Recientemente, algunos grupos se han enfocado a las redes neurodifusas autoorganizadas [10][13][27]. Estos métodos se basan en un modelo neurodifuso ANFIS, utilizan métodos de agrupamiento de tipo jerárquico donde el número de reglas es definido principalmente por 2 criterios, el criterio del error y el criterio de *e-completeness* [27]. La combinación de estos dos criterios define si se crea o actualiza una regla difusa, o si se fusionan una o más reglas en una regla difusa nueva.

El criterio del error se basa en el error entre la salida del modelo $\hat{y}$ y la salida del sistema y (2.61), se establece un umbral de error K_e, donde, se realiza una acción determinada ya sea que se cumpla o no el criterio (2.62). El umbral puede ser fijo o puede variar en alguna forma de acuerdo al criterio del diseñador.

$$e(k) = |y(k) - \hat{y}(k)| \tag{2.61}$$

$$e(k) > K_e \tag{2.62}$$

El criterio de *e-completeness (EC)* establece que para cualquier entrada al sistema que este dentro del rango de operación existe al menos una de las reglas difusas cuyo grado de disparo w_j (2.63) es mayor que un umbral K_d (2.64).

$$D = \max_j(w_j) \tag{2.63}$$

$$D < K_d \tag{2.64}$$

Las posibles combinaciones de estos dos criterios definen que acción debe realizarse. Para adicionar una regla difusa es común definir los parámetros de las funciones de membresía a partir de la información del dato y los parámetros de la parte consecuente de una manera aleatoria, para la actualización de las reglas se emplean variantes del algoritmo de retropropagación, o algún método de mínimos cuadrados [15] y sus variantes. En algunos diseños de sistemas neurodifusos autoorganizados es posible fusionar las reglas difusas para evitar un incremento en su número de reglas que provoque un costo computacional elevado o para evitar reglas similares, estas reglas difusas son fusionadas principalmente promediando los parámetros, por ejemplo los centros de las gaussianas y las desviaciones estándar así como los parámetros de la parte consecuente [10].

Capítulo 3

3 Redes Neurodifusas Recurrentes en Espacio de Estados

En este Capítulo se presenta un modelo neurodifuso autoorganizado propuesto para el modelado y control de sistemas dinámicos no lineales. Partiendo de las tareas propuestas para el modelado de sistemas. El esquema para diseñar la red neurodifusa parte de que no se dispone de información alguna del sistema y que la red inicia con cero reglas difusas. El diseño se realiza con datos de entrenamiento obtenidos del sistema dinámico y se realiza un análisis de estabilidad.

3.1 Estructura de la red

La estructura del modelo neurodifuso propuesto es llamado Red Neurodifusa en Espacio de Estados (SSFNN) (Figura 3.1) donde u_c es la entrada del sistema; u_s es un vector de mediciones del sistema que intervienen en la dinámica del sistema y [*y(k-1), y(k-2), ..., y(k-s)*] son *s* salidas retrasadas del sistema, en conjunto forman los estados $x(k) \in R^n$ que serán realimentados al modelo neurodifuso. $\overline{A} \in R^{nxn}$ es la matriz de estado y $\overline{B} \in R^{nx1}$ es el vector de entrada, $C \in R^{1xn}$ es el vector de salida del modelo neurodifuso, que conforman un sistema en espacio de estados en ciertas regiones del espacio del sistema; z^{-1} es un retraso en tiempo de una muestra *k*.

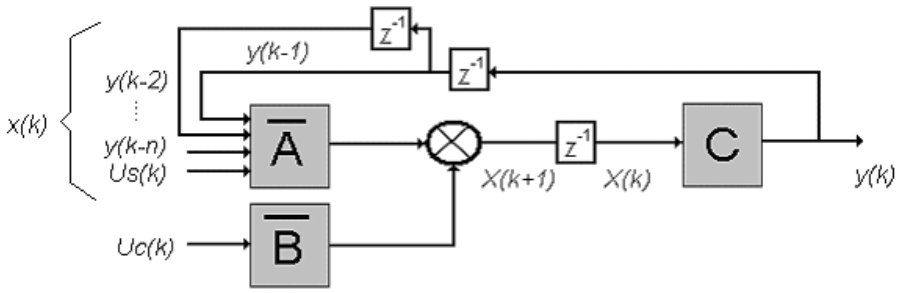

Figura 3.1 Representación de la red propuesta.

Donde la *j* regla difusa en espacio de estados es definida por

$$R_j : si \quad x_i(k) \quad es \quad \mu_{ij} \quad entonces \quad x^j(k+1) = A_j x(k) + B_j u(k)$$
$$i = 1, \cdots, n \quad j = 1, \cdots, r \tag{3.1}$$

Viendo el sistema difuso como la estructura de una red neuronal pueden definirse cinco capas. La capa 1 es una capa de entrada (3.2) formada por los estados del sistema.

$$z = x(k) \in R^n \tag{3.2}$$

La capa 2 es una capa de difusión (3.3) formada por funciones de membresía gaussianas, donde μ_{ij} es la función de membresía gaussiana de la entrada *i* y la regla *j*. Cuenta con dos parámetros de diseño, el centro de la función c_{ij} y la desviación estándar σ_{ij}.

$$\mu_{ij} = e\left(-\frac{(x_i - c_{ij})^2}{2\sigma_{ij}^{\ 2}} \right) \tag{3.3}$$

La capa 3 es llamada la capa de reglas (3.4) tiene una operación producto o T-norma [15], que calcula el grado de disparo w_j de la regla *j* difusa con *n* entradas al modelo.

$$w_j = \prod_{i=1}^{n} \mu_{ij} \tag{3.4}$$

La capa 4 es una capa de normalización (3.5) de las reglas difusas a valores entre 0 y 1 (3.6), con la propiedad de que la suma de todas las reglas normalizadas R_j es igual a 1 (3.7), esta propiedad permite realizar el análisis de estabilidad.

$$R_j = \frac{w_j}{\sum_{k=1}^{r} w_k} \tag{3.5}$$

$$0 < R_j \leq 1 \tag{3.6}$$

$$\sum_{j=1}^{r} R_j(z) = 1 \tag{3.7}$$

La capa 5 es la capa de salida (3.8) que representa la parte consecuente del sistema difuso, es representado por un sistema en variables de estado para cada regla con $A_j \in R^{nxn} \quad B_j \in R^n \quad x \in R^n \quad u(k) \in R.$

$$x(k+1) = \sum_{j=1}^{r} R_j \left(A_j x(k) + B_j u(k) \right) \tag{3.8}$$

El sistema neurodifuso para el modelado de un sistema dinámico (3.9) es visto como múltiples sistemas en espacio de estados, cada uno define el comportamiento del sistema dinámico en una región de acuerdo a la partición del espacio de entrada que define la combinación de las funciones de membresía definidas en cada una de las reglas difusas.

$$\begin{aligned} \hat{x}(k+1) &= \overline{A}\hat{x}(k) + \overline{B}u(k) \\ \hat{y} &= \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix} x(k) \end{aligned} \tag{3.9}$$

Donde $\overline{A}$ (3.10) y $\overline{B}$ (3.11) son matrices dinámicas definidas por el grado de disparo normalizado R_j.

$$\overline{A} = \sum_{j=1}^{r} R_j A_j \tag{3.10}$$

$$\overline{B} = \sum_{j=1}^{r} R_j B_j \tag{3.11}$$

3.2 Entrenamiento del modelo difuso

Dado que la estructura propuesta inicialmente no tiene definida ninguna regla difusa, es necesario realizar el entrenamiento de la estructura simultáneamente con el entrenamiento de los parámetros, el entrenamiento de la estructura es realizado en función

del criterio de e-completeness (2.63, 2.64), mientras que el entrenamiento de los parámetros es realizado en función del criterio del error (2.61, 2.62).

3.2.1 Entrenamiento de estructura

El entrenamiento de estructura es de tipo jerárquico donde se realiza la adición de reglas difusas, tal que cubran el espacio de entrada del sistema, para esto, se emplea el criterio de e-completeness el cual contempla que al menos una de las reglas difusas existentes cuyo grado de disparo w_j (3.4) tiene un valor mayor del umbral de disparo K_d para cada entrada del modelo. El umbral K_d para efectos de diseño es fijo y establecido de acuerdo al número de reglas que se quieran tener en el modelo, si el valor de K_d es pequeño las reglas creadas serán pocas, y si K_d es un valor grande entonces el número de reglas difusas será grande.

Primera regla

La primera regla difusa se crea en el instante k=1 con el primer dato de entrada al sistema difuso con la forma

$$R_1 : si \quad x_i(k) \quad es \quad \mu_{i1} \quad entonces \quad x^1(k+1) = A_1 x(k) + B_1 u(k) \tag{3.12}$$

Donde los centros de cada función de membresía $\mu_{i1}(c_{i1}, \sigma_{i1})$ de la primera regla son definidos como los datos de entrada $x_i(1)$, y las desviaciones estándar son definidas por un parámetro preestablecido σ_0, las matrices A_1 y B_1 son matrices aleatorias con eigenvalores de magnitud menor que uno de dimensiones apropiadas.

$$\begin{aligned} C_{i1} &= x_i(1) \\ \sigma_{ij} &= \sigma_0 \\ A_1 &= \begin{bmatrix} a_1^{11} & \cdots & a_1^{1n} \\ \vdots & \ddots & \vdots \\ a_1^{n1} & \cdots & a_1^{nn} \end{bmatrix} \\ B_1 &= \begin{bmatrix} b_1^1 \\ \vdots \\ b_1^n \end{bmatrix} \end{aligned} \tag{3.13}$$

Adición de una regla difusa

Si en cualquier instante *k* durante el barrido de los datos de entrenamiento no se cumple con el criterio de e-completeness (2.64) se toma la decisión de adicionar una nueva regla difusa R_N (3.14). Es necesario revisar si existe alguna función de membresía que cubra cada entrada del sistema difuso, esto se logra con un criterio de difusión (3.15), donde el valor mínimo de difusión permitido es $K_f = \sqrt[n]{K_d}$ para cada entrada al sistema difuso con *n* estados, tal que al calcular el grado de disparo este sea mayor a dicho umbral. Si el valor de difusión no es mayor a K_f significa que no existe ninguna función de membresía que represente esa entrada, por lo que se crea una nueva función de membresía para esa entrada cuyo centro es el valor de entrada $x_i(k)$ con desviación estándar preestablecida σ_0.

$$R_N : si \quad x_i(k) \quad es \quad \mu_{iN} \quad entonces \quad x^N(k+1) = A_N x(k) + B_N u(k) \tag{3.14}$$

$$\begin{aligned}
&for \quad i = 1 \ldots n+1 \\
&\quad f = \max_j(\mu_{ij}) \\
&\quad if \quad f > \sqrt[n]{K_d} \quad then \\
&\qquad c_{iN} = c_f \\
&\qquad \sigma_{iN} = \sigma_f \\
&\quad else \\
&\qquad c_{iN} = x_i \\
&\qquad \sigma_{iN} = \sigma_0
\end{aligned} \tag{3.15}$$

Donde las matrices A_N y B_N son matrices aleatorias estables

$$A_N = \begin{bmatrix} a_N^{11} & \dots & a_N^{1n} \\ \vdots & \ddots & \vdots \\ a_N^{n1} & \dots & a_N^{nn} \end{bmatrix}$$
$$B_N = \begin{bmatrix} b_N^{1} \\ \vdots \\ b_N^{n} \end{bmatrix} \tag{3.16}$$

La ventaja de definir de esta forma las reglas difusas es que se realiza una partición del espacio de entrada en un número reducido de reglas difusas, debido a que en cada nueva regla se pueden emplean las funciones de membresía existentes que cubran dicho espacio de entrada y no solo se crean nuevas funciones de membresía como en otros sistemas neurodifusos [10][13][27].

3.2.2 Entrenamiento de parámetros

Una vez que se tiene una estructura definida para el sistema difuso, es necesario realizar el entrenamiento de los parámetros que lo conforman para obtener un modelo lo más convincente para cumplir el objetivo de modelado. El entrenamiento de aprendizaje se realiza si se cumple con el criterio del error (2.62). Si el criterio no se cumple significa que el error (2.61) es menor de lo permitido y no se requiere el aprendizaje del sistema difuso. El umbral de error K_e (3.17) varia como una línea recta en función del la época actual (s), con un total de épocas (ep), entre un valor máximo de error e_{max} permitido al principio del entrenamiento (s=1) y un valor mínimo de error e_{min} (s=ep) en la última época de entrenamiento.

$$K_e = -\frac{e_{\max} + e_{\min}}{ep - 1}(s-1) + e_{\max} \tag{3.17}$$

Leyes de aprendizaje

En la estructura del sistema difuso (3.1) los parámetros $\{c_{ij}, \sigma_{ij}\}$ de la parte antecedente, junto con los parámetros del sistema en variables de estado $\{A_j, B_j\}$ serán actualizados de acuerdo al algoritmo RTRL (2.31). Definiendo la funcional de costo a minimizar (3.18).

$$J = \frac{1}{2}\left\|\hat{x} - x\right\|^2 = \frac{(\hat{x}_1 - x_1)^2 + \ldots + (\hat{x}_p - x_p)^2}{2} \tag{3.18}$$

Donde $p = 1 \ldots n$ con n estados del sistema. Los pesos a adaptar son $w = \{A_j, B_j, c_{ij}, \sigma_{ij}\}$ usando el algoritmo del gradiente se tiene la ley de adaptación (3.19) para cada uno de los parámetros de *w*.

$$w(k+1) = w(k) - \eta \nabla J(w(k)) \tag{3.19}$$

Para poder utilizar el algoritmo es necesario calcular las derivadas parciales de *J* con respecto a cada uno de los elementos de *w* para cada una de las reglas difusas del modelo.

Parámetros $a_j^{q,p}$

Primero se obtienen las leyes de aprendizaje para la parte consecuente, se establece la ley de aprendizaje con respecto a cada uno de los elementos de las matrices de estado A_j ($a_j^{q,p}$) (3.20), con $p, q = 1 \ldots n$, teniendo n estados del sistema y $j = 1 \ldots r$ reglas difusas existentes.

$$a_j^{q,p}(k+1) = a_j^{q,p}(k) - \eta \frac{\partial J(w)}{\partial a_j^{q,p}} \tag{3.20}$$

Obteniendo la derivada parcial de *J* con respecto a A_j

$$\frac{\partial J}{\partial a_j^{q,p}} = \left(\hat{x}_p(k+1) - x_p(k+1)\right) \frac{\partial \hat{x}_p{}^{+}(k+1)}{\partial a_j^{q,p}} \tag{3.21}$$

La derivada parcial total ∂^+ se calcula con el algoritmo de RTRL

$$\frac{\partial \hat{x}_p{}^{+}(k+1)}{\partial a_j^{q,p}(k)} = \frac{\partial \hat{x}_p(k+1)}{\partial a_j^{q,p}(k)} + \frac{\partial \hat{x}_p(k+1)}{\partial \hat{x}_p(k)} \frac{\partial \hat{x}_p^{+}(k)}{\partial a_j^{q,p}(k-1)} \tag{3.22}$$

Calculando las derivadas, la ley de aprendizaje queda en función del cálculo de la derivada anterior

$$\frac{\partial \hat{x}_p^{+}(k+1)}{\partial a_j^{q,p}(k)} = R_j x_p + R_j a_j^{q,p} \frac{\partial \hat{x}_p^{+}(k)}{\partial a_j^{q,p}(k-1)} \tag{3.23}$$

Parámetros b_j^p

Se establece la ley de aprendizaje con respecto a cada uno de los elementos de las matrices de entrada B_j (b_j^p) (3.24), con $p = 1 \ldots n$, teniendo n estados del sistema y $j = 1 \ldots r$ reglas difusas existentes.

$$b_j^p(k+1) = b_j^p(k) - \eta \frac{\partial J(w)}{\partial b_j^p} \tag{3.24}$$

Obteniendo la derivada parcial de J con respecto a B_j (3.25).

$$\frac{\partial J}{\partial b_j^p} = -\left(x_p(k+1) - \hat{x}_p(k+1)\right) \frac{\partial \hat{x}_p^{+}(k+1)}{\partial b_j^p} \tag{3.25}$$

La derivada parcial total ∂^+ se calcula con el algoritmo de RTRL (3.26).

$$\frac{\partial \hat{x}_p^{+}(k+1)}{\partial b_j^p(k)} = \frac{\partial \hat{x}_p(k+1)}{\partial b_j^p(k)} + \frac{\partial \hat{x}_p(k+1)}{\partial \hat{x}_p(k)} \frac{\partial \hat{x}_p^{+}(k)}{\partial b_j^p(k-1)} \tag{3.26}$$

Calculando las derivadas, la ley de aprendizaje queda en función del cálculo de la derivada anterior (3.27).

$$\frac{\partial \hat{x}_p^{+}(k+1)}{\partial b_j^p(k)} = R_j u(k) + R_j a_j^{q,p} \frac{\partial \hat{x}_p^{+}(k)}{\partial b_j^p(k-1)} \tag{3.27}$$

Parámetros $c_{i,j}$

Se establece la ley de aprendizaje con respecto a cada uno de los centros de las funciones de membresía de las entradas difusas $c_{i,j}$ (3.28), con $i = 1 \ldots n+1$, teniendo n estados del sistema y $j = 1 \ldots r$ reglas difusas existentes.

$$c_{i,j}(k+1) = c_{i,j}(k) - \eta \frac{\partial J(w)}{\partial c_{i,j}} \tag{3.28}$$

Obteniendo la derivada parcial de *J* con respecto a $c_{i,j}$

$$\frac{\partial J}{\partial c_{i,j}} = \left(\hat{x}(k+1) - x(k+1)\right)^T \frac{\partial \hat{x}^+(k+1)}{\partial c_{i,j}} \tag{3.29}$$

La derivada parcial total ∂^+ se calcula con el algoritmo de RTRL (3.30), la ley de aprendizaje queda en función del cálculo de la derivada del sistema visto sin recurrencia y de la derivada anterior.

$$\frac{\partial \hat{x}^+(k+1)}{\partial c_{i,j}(k)} = \frac{\partial \hat{x}(k+1)}{\partial c_{i,j}(k)} + \frac{\partial \hat{x}(k+1)}{\partial \hat{x}(k)} \frac{\partial \hat{x}^+(k)}{\partial c_{i,j}(k-1)} \tag{3.30}$$

Calculando la primera de las derivadas se aplica la regla de la cadena para obtener las derivadas. (3.31).

$$\frac{\partial \hat{x}(k+1)}{\partial c_{i,j}(k)} = \left(A_j \hat{x}(k) + B_j u(k)\right) \frac{\frac{\partial \psi_j}{\partial c_{ij}}\left(\sum \psi_l\right) - \frac{\partial\left(\sum \psi_l\right)}{\partial c_{ij}} \psi_j}{\left(\sum \psi_l\right)^2} \tag{3.31}$$

Obteniendo las derivadas de los ψ y su sumatoria, falta obtener la derivada de las funciones de membresía μ

$$\frac{\partial \hat{x}(k+1)}{\partial c_{i,j}(k)} = \left(A_j \hat{x}(k) + B_j u(k)\right) \frac{\left(\frac{\psi_j}{\mu_{ij}}\left(\sum \psi_l\right) - \frac{\psi_j}{\mu_{ij}} \psi_j\right) \frac{\partial \mu_{ij}}{\partial c_{ij}}}{\left(\sum \psi_l\right)^2} \tag{3.32}$$

$$\frac{\partial \hat{x}(k+1)}{\partial c_{i,j}(k)} = \left(A_j \hat{x}(k) + B_j u(k)\right)\left(R_j - R_j^{\,2}\right)\left(\frac{z_i - c_{ij}}{\sigma_{ij}^{\,2}}\right) \tag{3.33}$$

La derivada debido a las retroalimentaciones internas del sistema difuso permiten obtener la ley de aprendizaje total para el sistema recurrente

$$\frac{\partial \hat{x}(k+1)}{\partial \hat{x}(k)} = \frac{\partial\left(\sum_{j=1}^{r}\left(A_j \hat{x}(k) + B_j u(k)\right)\right)}{\partial \hat{x}(k)} \tag{3.34}$$

La derivada queda en función de las matrices de la dinámica interna

$$\frac{\partial \hat{x}(k+1)}{\partial \hat{x}(k)} = \sum_{j=1}^{r} R_j A_j \tag{3.35}$$

La ley de aprendizaje total para los centros de las funciones de membresía es de la siguiente forma:

$$\frac{\partial \hat{x}^{+}(k+1)}{\partial c_{i,j}(k)} = \left(A_j \hat{x}(k) + B_j u(k)\right)\left(R_j - {R_j}^2\right)\left(\frac{z_i - c_{ij}}{{\sigma_{ij}}^2}\right) + \sum_{j=1}^{r} R_j A_j \frac{\partial \hat{x}^{+}(k)}{\partial c_{i,j}(k-1)} \tag{3.36}$$

Parámetros $\sigma_{i,j}$

Se establece la ley de aprendizaje con respecto a cada una de las desviaciones estándar de las funciones de membresía de las entradas difusas $\sigma_{i,j}$ (3.37), con $i = 1 \ldots n+1$, teniendo *n* estados del sistema y $j = 1 \ldots r$ reglas difusas existentes.

$$\sigma_{i,j}(k+1) = \sigma_{i,j}(k) - \eta \frac{\partial J(w)}{\partial \sigma_{i,j}} \tag{3.37}$$

Obteniendo la derivada parcial de *J* con respecto a $\sigma_{i,j}$

$$\frac{\partial J}{\partial \sigma_{i,j}} = \left(\hat{x}(k+1) - x(k+1)\right)^T \frac{\partial \hat{x}^{+}(k+1)}{\partial \sigma_{i,j}} \tag{3.38}$$

La derivada parcial total ∂^{+} se calcula con el algoritmo de RTRL

$$\frac{\partial \hat{x}^{+}(k+1)}{\partial \sigma_{i,j}(k)} = \frac{\partial \hat{x}(k+1)}{\partial \sigma_{i,j}(k)} + \frac{\partial \hat{x}(k+1)}{\partial \hat{x}(k)} \frac{\partial \hat{x}(k)}{\partial \sigma_{i,j}(k-1)} \tag{3.39}$$

La ley de aprendizaje queda en función del cálculo de la derivada del sistema visto sin recurrencia y de la derivada anterior.

$$\frac{\partial \hat{x}(k+1)}{\partial \sigma_{i,j}(k)} = \left(A_j \hat{x}(k) + B_j u(k)\right) \frac{\frac{\partial \psi_j}{\partial \sigma_{ij}} \left(\sum \psi_l\right) - \frac{\partial \left(\sum \psi_l\right)}{\partial \sigma_{ij}} \psi_j}{\left(\sum \psi_l\right)^2} \tag{3.40}$$

Obteniendo las derivadas de los ψ y su sumatoria, falta obtener la derivada de las funciones de membresía μ

$$\frac{\partial \hat{x}(k+1)}{\partial \sigma_{i,j}(k)} = \left(A_j \hat{x}(k) + B_j u(k)\right) \frac{\left(\frac{\psi_j}{\mu_{ij}} \left(\sum \psi_l\right) - \frac{\psi_j}{\mu_{ij}} \psi_j\right) \frac{\partial \mu_{ij}}{\partial \sigma_{ij}}}{\left(\sum \psi_l\right)^2} \tag{3.41}$$

$$\frac{\partial \hat{x}(k+1)}{\partial \sigma_{i,j}(k)} = \left(A_j \hat{x}(k) + B_j u(k)\right) \left(R_j - R_j^{\,2}\right) \left(\frac{\left(z_i - c_{ij}\right)^2}{\sigma_{ij}^{\,3}}\right) \tag{3.42}$$

La ley de aprendizaje total para los centros de las funciones de membresía es de la siguiente forma:

$$\frac{\partial \hat{x}^+(k+1)}{\partial \sigma_{i,j}(k)} = \left(A_j \hat{x}(k) + B_j u(k)\right) \left(R_j - R_j^{\,2}\right) \left(\frac{\left(z_i - c_{ij}\right)}{\sigma_{ij}^{\,3}}\right) + \sum_{j=1}^{r} R_j A_j \frac{\partial \hat{x}^+(k)}{\partial \sigma_{i,j}(k-1)} \tag{3.43}$$

3.2.3 Análisis de estabilidad

Una vez obtenido un modelo neurodifuso recurrente, es importante determinar la estabilidad del modelo, el método más empleado para analizar la estabilidad de un sistema no linear es determinar la estabilidad en el sentido de Lyapunov, para emplear este método es necesario considerar al sistema como un modelo en respuesta libre (3.45), es decir considerar al modelo difuso (3.44) con una entrada $u(k) = 0$. La estabilidad del sistema de la forma (3.45) puede ser verificado por el Teorema 3.1

$$x(k+1) = \sum_{j=1}^{r} R_j \left(A_j x(k) + B_j u(k) \right) \tag{3.44}$$

$$x(k+1) = \left(\sum_{j=1}^{r} R_j A_j \right) x(k) \tag{3.45}$$

Lema 3.1 Sea $A, B, P \in R^{nxn}$. Si $P = P^T > 0$ tal que

$$A^T PA - P < 0 \text{ y } B^T PB - P < 0 \tag{3.46}$$

Entonces

$$A^T PB + B^T PA - 2P < 0 \tag{3.47}$$

Teorema 3.1

El estado de equilibrio $x_e = 0$, del modelo (3.45) es global asintóticamente estable si existe una matriz definida positiva $P = P^T > 0$ tal que para $i = 1,2,\ldots r$,

$$A_j^T PA_j - P < 0 \tag{3.48}$$

Prueba: Suponiendo que existe una matriz simétrica positiva definida $P = P^T > 0$ tal que para $i = 1,2,\ldots r$ desigualdades dadas por (3.48) son satisfechas. Considerar la función candidata de Lyapunov

$$V(k) = x^T(k) P x(k) \tag{3.49}$$

Para probar el Teorema es necesario mostrar que

$$\Delta V(k) < 0 \tag{3.50}$$

Se tiene que:

$$\begin{aligned}\Delta V(k) &= V(k+1) - V(k) \\ &= x^T(k+1)Px(k+1) - x^T(k)Px(k) \\ &= x^T(k)\left(\sum_{j=1}^{r} R_j A_j^T\right)Px(k)\left(\sum_{j=1}^{r} R_j A_j\right) - x^T(k)Px(k) \\ &= x^T(k)\left(\left(\sum_{j=1}^{r} R_j A_j^T\right)P\left(\sum_{j=1}^{r} R_j A_j\right) - P\right)x(k)\end{aligned} \tag{3.51}$$

Para proceder se representa (3.51) en una forma equivalente como

$$\Delta V(k) = x^T(k)\left(\left(\sum_{i=1}^{r} R_i A_i^T\right)P\left(\sum_{j=1}^{r} R_j A_j\right) - P\right)x(k) \tag{3.52}$$

Teniendo

$$\sum_{i=1}^{r} R_i \sum_{j=1}^{r} R_j = \sum_{i=1}^{r}\sum_{j=1}^{r} R_i R_j = 1 \tag{3.53}$$

Sustituyendo (3.53) en (3.52) se tiene

$$\begin{aligned}\Delta V(k) &= x^T(k)\left(\left(\sum_{i=1}^{r} R_i A_i^T\right)P\left(\sum_{j=1}^{r} R_j A_j\right) - \sum_{i=1}^{r}\sum_{j=1}^{r} R_i R_j P\right)x(k) \\ &= x^T(k)\left(\sum_{i=1}^{r}\sum_{j=1}^{r} R_i R_j \left(A_i^T P A_j - P\right)\right)x(k)\end{aligned} \tag{3.54}$$

Re organizando (3.54) para conseguir

$$\begin{aligned}\Delta V(k) &= x^T(k)\sum_{j=1}^{r} R_j^2\left(A_j^T P A_j - P\right)x(k) + x^T(k)\sum_{\substack{i=1 \\ i\neq j}}^{r}\sum_{j=1}^{r} R_i R_j \left(A_j^T P A_i - P\right)x(k) \\ &= x^T(k)\sum_{j=1}^{r} R_j^2\left(A_j^T P A_j - P\right)x(k) \\ &\quad + x^T(k)\sum_{i>j}^{r}\sum_{j=1}^{r} R_i R_j \left(A_j^T P A_i + A_i^T P A_j - 2P\right)x(k)\end{aligned} \tag{3.55}$$

Por lo tanto

$$\Delta V(k) < 0 \tag{3.56}$$

El primer término en (3.54) es negativo definido por suposición de que $P = P^T > 0$ y cumpliendo con (3.6 y 3.7). El segundo término en (3.54) es negativo definido por el lema 3.1 (3.47). Por lo tanto, $V = x^T Px$ es una función de Lyapunov para el modelo (3.45), y por el Teorema de Lyapunov el estado de equilibrio $x_e = 0$ del modelo es global y asintóticamente estable. El Teorema queda demostrado.

3.3 Conclusiones

El uso de un modelo neurodifuso SSFNN tiene múltiples ventajas entre las que destacan:

- No requiere información alguna del sistema a modelar solo las mediciones entrada - salida.
- Puede incorporar mediciones que intervengan en el sistema sin conocer su relación pero que afectan la respuesta del sistema y estas pueden estar disponibles o no en línea.
- Se puede realizar un análisis de estabilidad del modelo para efectos de tareas de control.
- Para modelado de sistemas MIMO se pueden interconectar múltiples modelos SSFNN cuyo resultado es un modelo SSFNN estable.

Los modelos SSFNN solo son válidos para sistemas estables (inherentes o controlados).

Capítulo 4

1 Controlador basado en un observador difuso

1.1 Observador Difuso

En la práctica las variables de estado que no están disponibles para su medición, suelen ser estimas para efectos de control. Un observador de estado estima las variables de estado con base en las mediciones de las variables de salida y de control. Un observador de estado de orden completo estima todas las variables de estado, sin importar si algunas están disponibles para una medición directa, esto es el vector de estado observado $\tilde{x} \in R^n$ tiene el mismo orden que el vector de estado $x \in R^n$.

1.1.1 Estructura del observador

El observador difuso (4.3), (4.4) de orden completo emplea el modelo SSFNN de la planta (4.1, 4.2). para obtener las matrices A_j y B_j.

$$x(k+1) = \sum_{j=1}^{r} R_j A_j x(k) + \sum_{j=1}^{r} R_j B_j u(k) = \overline{A}x(k) + \overline{B}u(k) \tag{4.1}$$

$$\hat{y}(k) = Cx(k) \tag{4.2}$$

$$\tilde{x}(k+1) = \sum_{j=1}^{r} R_j A_j \tilde{x}(k) + \sum_{j=1}^{r} R_j B_j u(k) + \sum_{j=1}^{r} R_j L_j \left(y(k) - \tilde{y}(k)\right) \tag{4.3}$$

$$\tilde{y}(k) = C\tilde{x}(k) \tag{4.4}$$

donde $\tilde{x}$ denota el vector de estado observado, $\tilde{y}$ es la salida del observador, A_j, B_j son las matrices de los modelos locales del SSFNN y R_j es el peso de la parte antecedente del modelo SSFNN. L_j es la matriz del observador local de la *j* regla.

El observador de estado es visto como el resultado de la implementación de *r* reglas difusas de la forma

$$R: si \quad \tilde{x}_i(k) \quad es \quad \mu_{ij} \quad entonces \quad \tilde{x}_j(k+1) = A_j\tilde{x}(k) + B_j u(k) + L_j(y - \hat{y})$$
$$i = 1, \cdots, m \quad j = 1, \cdots, r \tag{4.5}$$

Para obtener la ecuación de error del observador, se resta la ecuación del modelo (4.1) y la ecuación del observador (4.3)

$$\begin{aligned} e(k+1) &= x(k+1) - \tilde{x}(k+1) \\ &= \sum_{j=1}^{r} R_j \left[\left(A_j x(k) + B_j u(k) \right) - \left(A_j \tilde{x}(k) + B_j u(k) + L_j \left(y(k) - \tilde{y}(k) \right) \right) \right] \end{aligned} \tag{4.6}$$

Eliminando los términos comunes y sustituyendo las ecuación (4.2) y (4.4) en (4.6)

$$\begin{aligned} e(k+1) &= \sum_{j=1}^{r} R_j A_j x(k) - \sum_{j=1}^{r} R_j L_j C x(k) - \sum_{j=1}^{r} R_j A_j \tilde{x}(k) + \sum_{j=1}^{r} R_j L_j C \tilde{x}(k) \\ &= \left(\sum_{j=1}^{r} R_j A_j - \sum_{j=1}^{r} R_j L_j C \right) x(k) - \left(\sum_{j=1}^{r} A_j - \sum_{j=1}^{r} R_j L_j C \right) \tilde{x}(k) \\ &= \left(\sum_{j=1}^{r} R_j \left(A_j - L_j C \right) \right) \left(x(k) - \tilde{x}(k) \right) \end{aligned} \tag{4.7}$$

La dinámica del error es definida por (4.8).

$$e(k+1) = \left(\sum_{j=1}^{r} R_j \left(A_j - L_j C \right) \right) e(k) \tag{4.8}$$

1.1.2 Análisis de estabilidad

La estabilidad del observador de la forma (4.3) puede ser verificada por el Teorema 4.1, que esta basada en la estabilidad de Lyapunov para sistemas no lineales.

Teorema 4.1

Suponiendo que los pares (A_j, C_j) son observables, el error de estimación del observador de la forma (4.3) es asintóticamente estable a cero $e \to 0 \; si \; k \to \infty$ en la región

$\Omega = \{\forall \tilde{x}(k) \mid \max(\psi_j) > K_d\}$, si existe una matriz definida positiva $P = P^T > 0$ tal que para $j = 1,2,\ldots r$,

$$(A_j - K_j C)^T P(A_j - K_j C) - P < 0 \tag{4.9}$$

Prueba: Suponiendo que existe una matriz simétrica positiva definida $P = P^T > 0$ tal que para $j = 1,2,\ldots r$ desigualdades dadas por (4.9) son satisfechas. Considerar la función candidata de Lyapunov (4.10).

$$V(k) = e^T(k) P e(k) \tag{4.10}$$

Para probar el Teorema es necesario mostrar que

$$\Delta V(k) < 0 \tag{4.11}$$

Se tiene que:

$$\begin{aligned}
\Delta V(k) &= V(k+1) - V(k) \\
&= e^T(k+1) P e(k+1) - e^T(k) P e(k) \\
&= e^T(k) \left(\sum_{j=1}^{r} R_j (A_j - L_j C) \right)^T P \left(\sum_{j=1}^{r} R_j (A_j - L_j C) \right) e(k) - e^T(k) P e(k) \\
&= e^T(k) \left(\left(\sum_{j=1}^{r} R_j (A_j - L_j C) \right)^T P \left(\sum_{j=1}^{r} R_j (A_j - L_j C) \right) - P \right) e(k)
\end{aligned} \tag{4.12}$$

Para proceder se representa (4.12) en una forma equivalente como

$$\Delta V(k) = e^T(k) \left(\left(\sum_{i=1}^{r} R_i (A_i - L_i C) \right)^T P \left(\sum_{j=1}^{r} R_j (A_j - L_j C) \right) - P \right) e(k) \tag{4.13}$$

Teniendo en cuenta que

$$\sum_{i=1}^{r} R_i \sum_{j=1}^{r} R_j = \sum_{i=1}^{r} \sum_{j=1}^{r} R_i R_j = 1 \tag{4.14}$$

Sustituyendo (4.14) en (4.13) se tiene

$$\begin{aligned}\Delta V(k) &= e^T(k)\left(\left(\sum_{j=1}^{r} R_j\left(A_j - L_jC\right)^T\right)P\left(\sum_{j=1}^{r} R_j\left(A_j - L_jC\right)\right) - \sum_{i=1}^{r}\sum_{j=1}^{r} R_iR_jP\right)e(k) \\ &= e^T(k)\left(\sum_{i=1}^{r}\sum_{j=1}^{r} R_iR_j\left(\left(A_j - L_jC\right)^T P\left(A_j - L_jC\right) - P\right)\right)e(k)\end{aligned} \tag{4.15}$$

Re organizando (4.15)

$$\begin{aligned}\Delta V &= e^T(k)\sum_{j=1}^{r} R_j^2\left(\left(A_j - L_jC\right)^T P\left(A_j - L_jC\right) - P\right)e(k) \\ &\quad + e^T(k)\sum_{\substack{i=1 \\ i\neq j}}^{r}\sum_{j=1}^{r} R_iR_j\left(\left(A_i - L_iC\right)^T P\left(A_j - L_jC\right) - P\right)e(k) \\ &= e^T(k)\left(\sum_{j=1}^{r} R_j^2\left(\left(A_j - L_jC\right)^T P\left(A_j - L_jC\right) - P\right)e(k)\right. \\ &\quad \left. + \sum_{i>j}^{r}\sum_{j=1}^{r} R_iR_j\left(\left(A_j - L_jC\right)^T P\left(A_i - L_iC\right) + \left(A_i - L_iC\right)^T P\left(A_j - L_jC\right) - 2P\right)\right)e(k)\end{aligned} \tag{4.16}$$

Por lo tanto

$$\Delta V(k) < 0 \tag{4.17}$$

El primer término en (4.16) es negativo definido por suposición de $P = P^T > 0$ y cumpliendo con las ecuaciones (3.6) y (3.7). El segundo término en (4.16) es negativo definido por el lema 3.1 ecuación (3.47). Por lo tanto, $V = e^T Pe$ es una función de Lyapunov para el observador (4.3), y por el Teorema de Lyapunov el error de estimación del observador es asintóticamente estable a cero $e \to 0\, si\, k \to \infty$ en la región Ω. El Teorema queda demostrado.

1.2 Controlador

1.2.1 Ley de control

Considerando un sistema no lineal de la forma:

$$x(k+1) = f(x(k)) + g(x(k))u \tag{4.18}$$

$$y(k) = Cx(k) \tag{4.19}$$

donde f y g son funciones discretas desconocidas, $u \in R$ y $y \in R$ son la entrada y salida del sistema, respectivamente. $x = (x_1, x_2, \ldots, x_n)^T \in R^n$ es el vector de estado del sistema que se asume esta disponible para medición y $C = [1 \quad 0 \quad \ldots \quad 0] \in R^n$. El objetivo de control es forzar a que la salida de la planta $y(k)$ siga una señal de referencia acotada $y_d(k)$. El error de seguimiento debe ser el más pequeño posible $e = y_d(k) - y(k)$.

Si las funciones $f(x(k))$ y $g(x(k))$ son conocidas, entonces la ley de control es:

$$u(k) = g(x)^{+}\left[-f(x) + h(x)^{+}\left(y_d(k+1) + \beta e_c(k)\right)\right] \tag{4.20}$$

Pero debido a que no se conocen las funciones $f(x)$, $g(x)$ y $h(x)$ son sustituidas por sus estimados.

$$u(k) = \tilde{g}(\tilde{x})^{+}\left[-\tilde{f}(\tilde{x}) + C^{+}\left(y_d(k+1) + \beta e_c(k)\right)\right] \tag{4.21}$$

donde las funciones estimadas se componen del observador difuso (4.3, 4.4),

$$\tilde{f}(x) = \bar{A}\tilde{x}(k) + \bar{L}\left(y(k) - \tilde{y}(k)\right) \tag{4.22}$$

$$\tilde{g}(\tilde{x}) = \bar{B} \tag{4.23}$$

donde $\bar{L} = \sum_{j=1}^{r} L_j$ es la matriz dinámica formada por las matrices locales de los observadores. y_d es la salida deseada, y es la salida actual y $\tilde{y}$ es la salida del observador.

La función $\tilde{f}(x)$ es definida con la retro del observador debido a que tras el entrenamiento del modelo neurodifuso no es posible asegurarse que este es estable por lo que se asegura una estabilidad practica, ya que en una forma convencional $\tilde{f}(x)$ se define con la dinámica del modelo y al no ser estable $\bar{A}$ la señal de control seria inestable.

La ley de control queda expresada en función de las matrices dinámicas $\bar{A}$, $\bar{B}$, $\bar{L}$ y del error de seguimiento e_c.

$$u_c(\tilde{x}) = (\bar{B})^{+}\left[-\bar{A}\tilde{x}(k) - \bar{L}(y(k) - \tilde{y}(k)) + C^{+}(y_d(k+1) + \beta e_c(k))\right] \tag{4.24}$$

1.2.2 Análisis de estabilidad

Re escribiendo (4.21) se tiene que:

$$y_d(k+1) = C(\tilde{f}(\tilde{x}) + \tilde{g}(\tilde{x})u_c(k)) - \beta e_c(k) \tag{4.25}$$

La ecuación dinámica del error de control es

$$\begin{aligned} e_c(k+1) &= y_d(k+1) - y(k+1) \\ &= C(\tilde{f}(\tilde{x}) + \tilde{g}(\tilde{x})u_c(k)) - \beta e_c - C(f(x) + g(x)u_c(k)) \\ &= C(\tilde{f}(\tilde{x}) + \tilde{g}(\tilde{x})u_c(k) - f(x) + g(x)u_c(k)) - \beta e_c \\ &= C(\tilde{x}(k+1) - x(k+1)) - \beta e_c(k) \end{aligned} \tag{4.26}$$

Si se define el error de observación como

$$e_m(k+1) = x(k+1) - \tilde{x}(k+1) \tag{4.27}$$

Por el Teorema 4.1 se tiene que $\lim_{k\to\infty} e_m \to 0$, por lo que la dinámica del error de seguimiento depende de β. Si $0 < \beta < 1$ el error de seguimiento converge a cero $\lim_{k\to\infty} e_c \to 0$.

1.3 Conclusiones

El observador basado en SSFNN al ser diseñado se puede asegurar que es estable y con el comportamiento deseado, más aún, teniendo múltiples observadores SSFNN estables pueden ser interconectados teniendo como resultado un observador MIMO. La ventaja de diseñar el observador es que cada uno de los modelos locales se diseña por separado y con técnicas existentes para sistemas lineales.

El controlador difuso solo requiere de un parámetro β para ajustar el control del sistema, el problema es que su elección debe ser a prueba y error, pero una vez teniendo el

observador SSFNN puede ser simulada la respuesta del sistema y hacer una elección de la β más adecuada para el sistema.

Capítulo 5

1 Caso de estudio

En este Capítulo se muestran los resultados obtenidos al aplicar los algoritmos de modelado de sistemas no lineales mostrados en el Capítulo 3, y el diseño del observador y control difusos mostrados en el Capítulo 4, con el objetivo de mostrar la capacidad de generación de las estructuras y el algoritmo de aprendizaje para el modelo difuso, el observador propuesto y el controlador difuso. Primero se muestras los resultado para el modelado de un sistema no lineal comúnmente usado como benchmark y se muestran los resultados comparando con otros métodos de redes neurodifusas. Como segundo ejemplo se modela un sistema no lineal y se realiza el diseño del observador y controlador difuso. En ambos casos se muestra un análisis de estabilidad, para asegurar estabilidad asintótica y global.

1.1 Sistema de Narendra

El sistema de Narendra consiste de una planta discreta no lineal con múltiples retrasos de tiempo descrita por (5.1)

$$\begin{aligned} y_p(k+1) &= f\left(y_p(k), y_p(k-1), y_p(k-2), u(k), u(k-1)\right) \\ &= \frac{y_p(k)y_p(k-1)u(k-1)\left(y_p(k-2)-1\right)+u(k)}{1+\left(y_p(k)\right)^2+\left(y_p(k-2)\right)^2} \end{aligned} \tag{5.1}$$

Este sistema es empleado comúnmente para comparar el desempeño de sistemas de modelado recurrentes, las comparaciones son mostradas en la Tabla 5.1.

La red SSFNN propuesta en el Capítulo 3 toma en cuenta solo la salida y entrada de la planta actuales $\lfloor y_p(k) \quad u(k) \rfloor$ para modelar el sistema (5.1), se utilizó un patrón de entrenamiento aleatorio con media cero y desviación estándar uno (Figura 5.1), en otros métodos se propone un patrón de entrenamiento donde la mitad de los datos es una señal aleatoria y la otra mitad una señal sinusoidal [6][7],

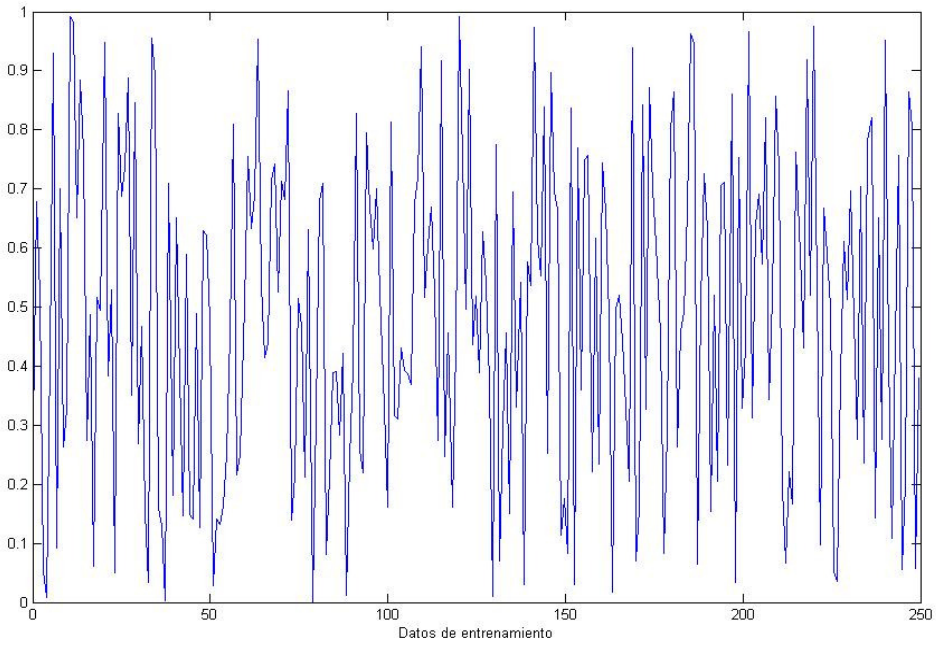

Figura 5.1 Datos de entrenamiento para el sistema no lineal

Los parámetros de entrenamiento son los que obtuvieron un error mínimo de modelado con respecto a los métodos de comparación [4][6][7][9]. Los parámetros de entrenamiento son $\eta_a = 0.09$, $\eta_b = 0.009$, $\eta_c = 0.0001$ y $\eta_\sigma = 0.0001$ que permitieron un entrenamiento estable con respecto al método del gradiente, los parámetros correspondientes a la generación de reglas y a la toma de decisión para entrenar o no el sistema son: $K_d = 0.3$, $\sigma_0 = 0.2$, $K_e = [0.1 : 0.00001]$. Se realizaron 100 épocas de entrenamiento.

En este caso solo se decidió usar un estado interno $x(k) = y(k)$, después del entrenamiento se generaron 3 reglas difusas con una partición del rango del estado interno en tres.

$$\begin{aligned} R_1 &: Si\ x(k)\ es\ \mu_1\ entonces\ x^1(k+1) = A_1 x(k) + B_1 u(x) \\ R_2 &: Si\ x(k)\ es\ \mu_2\ entonces\ x^2(k+1) = A_2 x(k) + B_2 u(x) \\ R_3 &: Si\ x(k)\ es\ \mu_3\ entonces\ x^3(k+1) = A_3 x(k) + B_3 u(x) \end{aligned} \tag{5.2}$$

donde

$$\mu = \begin{bmatrix} gauss(0.512,0.192) \\ gauss(0.1747,0.223) \\ gauss(0.971,0.168) \end{bmatrix} \tag{5.3}$$

donde $gauss(c,\sigma)$ es una función de membresía gaussiana con centro c y desviación estándar σ. La partición del espacio del estado interno del a red SSFNN se muestra en la Figura 5.2

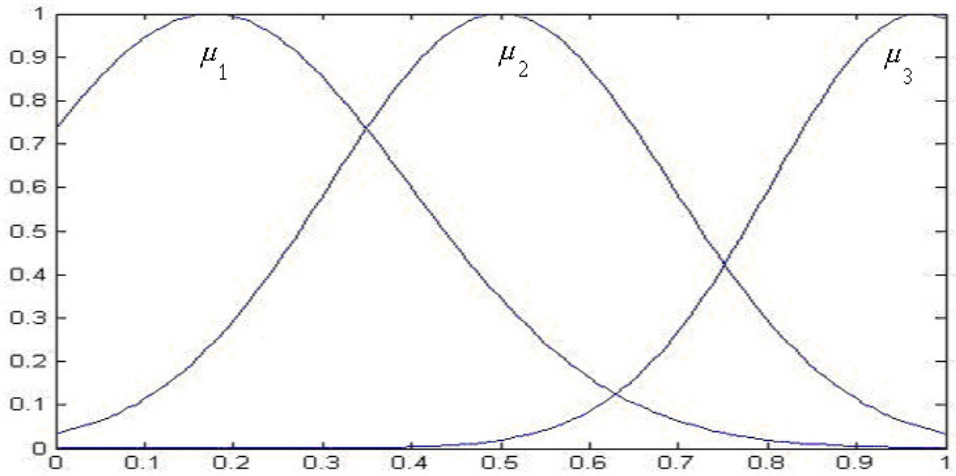

Figura 5.2 Funciones de membresía generadas para el estado interno

La parte consecuente del modelo difuso se compone por las matrices A y B compuestas por:

$$\begin{bmatrix} A_1 \\ A_2 \\ A_3 \end{bmatrix} = \begin{bmatrix} -0.0814 \\ 0.7139 \\ 0.4554 \end{bmatrix} \qquad \begin{bmatrix} B_1 \\ B_2 \\ B_3 \end{bmatrix} = \begin{bmatrix} 1.0237 \\ 0.5826 \\ 0.4200 \end{bmatrix} \tag{5.4}$$

El la Figura 5.3a se muestra el disparo de reglas normalizado para el patrón de entrenamiento, donde, dependiendo del color es el grado de disparo de cada regla siendo un valor bajo cercano a cero es un color casi blanco y un valor cercano a uno es un color negro; la Figura 5.3b muestra el error entre la salida de la planta y la salida del modelo donde el error se mantiene en una banda $e \in [-0.03,0.01]$. El error cuadrático medio

obtenido durante las 100 épocas se muestra en la Figura 5.3c, con un error final $e_{rms} = 0.0073$.

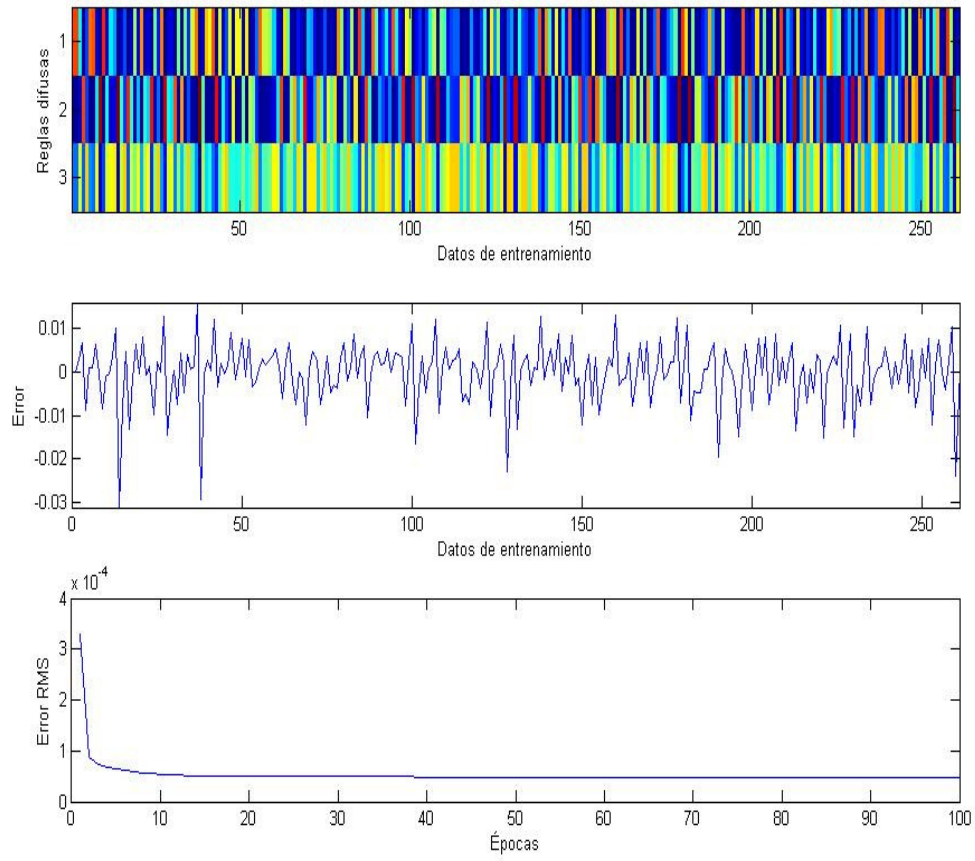

Figura 5.3 Desempeño del modelo, a) Distribución de reglas, b) Error de modelado, c) Error RMS

La Figura 5.4 muestra el resultado del modelado del patrón de entrenamiento, donde se observa que el error es casi imperceptible.

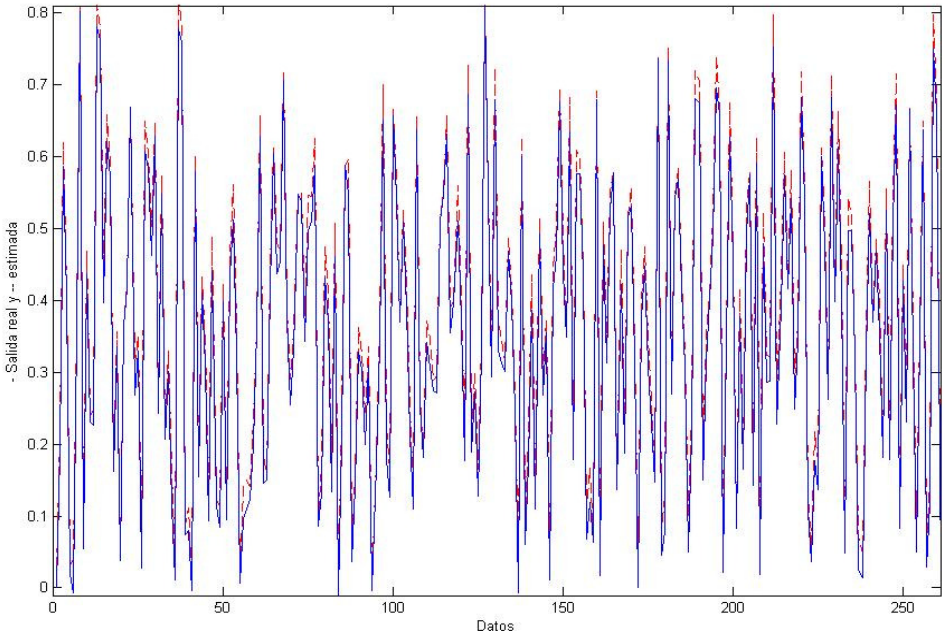

Figura 5.4 Respuesta del modelo, (–) salida deseada, (– –) salida del modelo

En la Tabla 5.1 se muestra la generación de reglas y el error RMS del modelado, variando solo el umbral del disparo K_d (Ec.), los parámetros constantes son $\eta_a = 0.09$, $\eta_b = 0.009$, $\eta_c = 0.0001$ y $\eta_\sigma = 0.0001$, $\sigma = 0.2$, $K_e = [0.1 : 0.00001]$, con 100 épocas de entrenamiento.

Tabla 5.1 Número de reglas y error RMS para diferentes umbrales de disparo K_d

K_d	Reglas	e_{rms}
0.1	3	0.0136
0.2	3	0.0146
0.3	3	0.0154
0.4	3	0.0160
0.5	4	0.0105
0.6	5	0.0101
0.7	5	0.0102
0.8	7	0.0100

0.9	8	0.0105
0.95	10	0.0100
0.99	22	0.0116

En la Tabla 5.2se muestra la relación que hay en la generación de reglas y el error RMS con respecto a la elección de las variaciones estándar iniciales σ_0 de las funciones de membresía. los parámetros constantes son $\eta_a = 0.09$, $\eta_b = 0.009$, $\eta_c = 0.0001$ y $\eta_\sigma = 0.0001$, $K_d = 0.3$, $K_e = [0.1 : 0.00001]$, con 100 épocas de entrenamiento.

Tabla 5.2 Número de reglas y error RMS para diferentes desviaciones estándar iniciales σ_o

σ	Reglas	e_{rms}
0.01	39	0.0087
0.05	10	0.0100
0.1	5	0.0112
0.2	3	0.0154
0.3	3	0.0162
0.4	1	0.0105
0.5	1	0.0105
0.6	1	0.0105
0.7	1	0.0105
0.8	1	0.0105
0.9	1	0.0105

Una vez entrenada la red neurodifusa se prueba la eficiencia del algoritmo con respecto a un patrón de prueba (5.5) cuyos resultados se muestran en la Figura 5.5.

$$u(k) = \begin{cases} \sin\left(\frac{nk}{25}\right) & 0 < k \le 250 \\ 1 & 250 < k \le 500 \\ -1 & 501 < k \le 750 \\ 0.3\sin\left(\frac{nk}{25}\right) + 0.1\sin\left(\frac{nk}{32}\right) + 0.6\sin\left(\frac{nk}{10}\right) & 751 < k \le 1000 \end{cases} \tag{5.5}$$

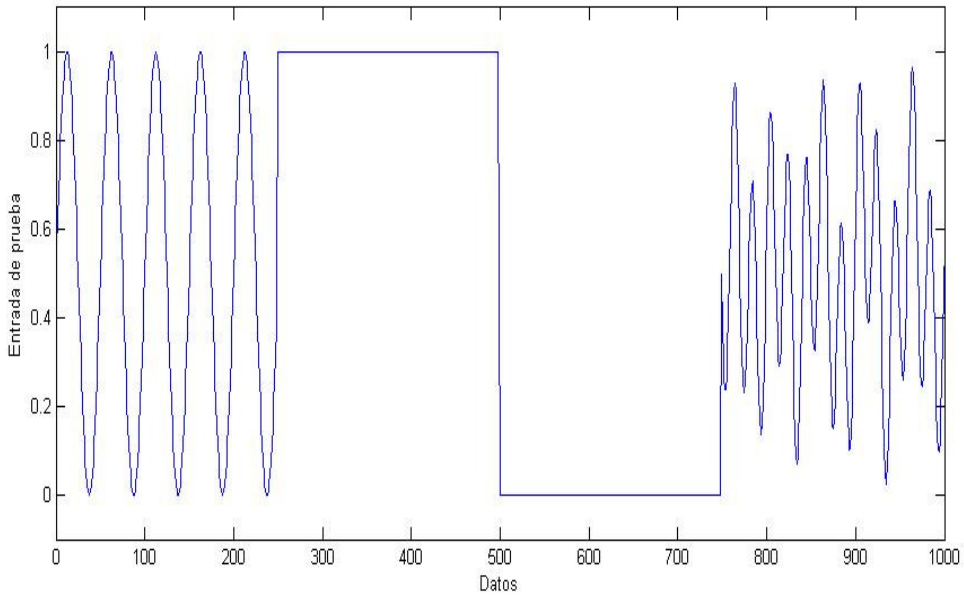

Figura 5.5 Datos de prueba para observar el desempeño del modelo

La Figura 5.6 muestra la comparación de la salida del sistema (-) y la salida del modelo (--), donde se observa que aunque el patrón no fue mostrado durante el entrenamiento, el sistema puede seguir la dinámica aun con componentes de alta frecuencia.

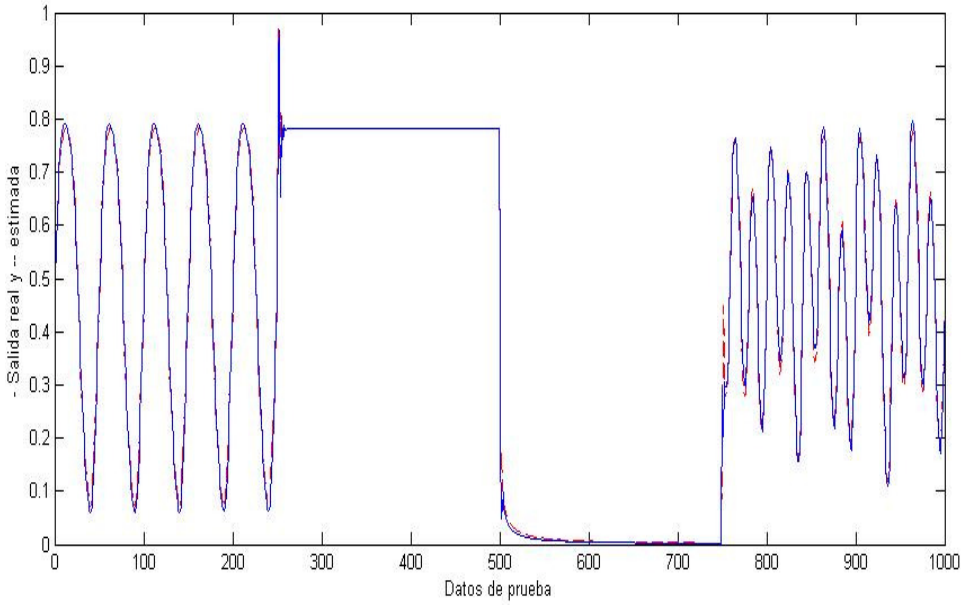

Figura 5.6 Respuesta del modelo al patrón de prueba, (–) salida del sistema, (– –) salida del modelo

El error RMS para un total de 1000 muestras fue $e_{rms} = 0.0202$, el error de modelado se muestra en la Figura 5.7a. La Figura 5.7b muestra la distribución de las reglas de acuerdo al espacio de estados internos del modelo neurodifuso recurrente, donde se observa que todas las reglas difusas son funcionales en cierto subespacio de estado.

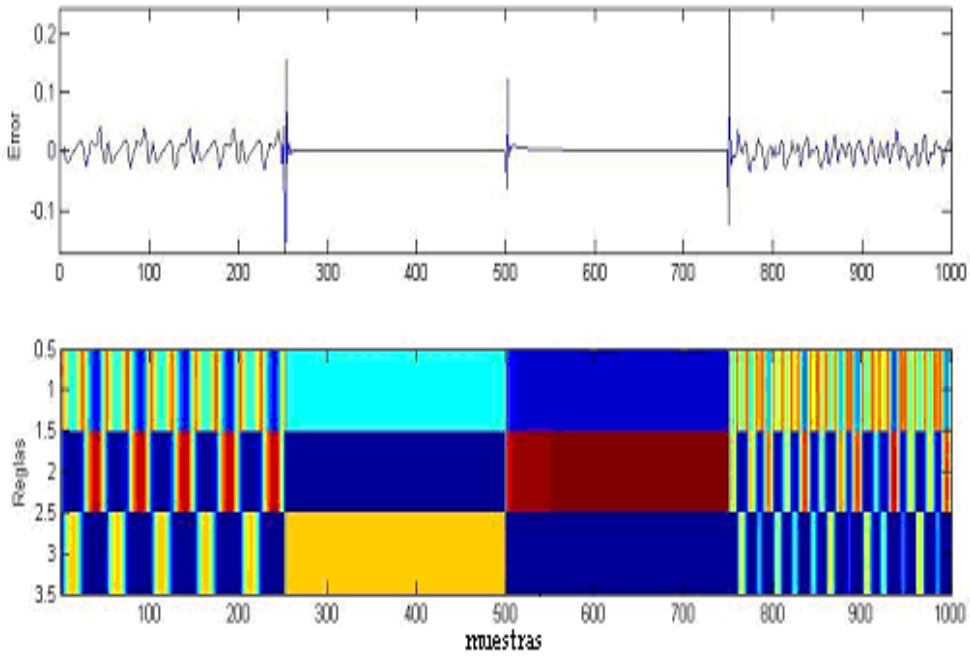

Figura 5.7 a) Error para el patrón de prueba, b) distribución del disparo de las reglas difusas.

Una vez obtenido el desempeño del modelo con respecto a un patrón de entrenamiento y uno de prueba, es necesario comparar este desempeño con respecto a otros métodos ya existentes. En la Tabla 5.3 se muestra la comparación del algunas estructuras recurrentes propuestas en la literatura. La primera estructura es una red neurodifusa recurrente RSONFIN propuesta por Juang [7] que utiliza una retroalimentación de variables lingüísticas, donde las variables se incrementan conforme se aumentan las reglas difusas. La segunda estructura es una red neurodifusa recurrente propuesta por Lee [9] que considera la capa de variables lingüísticas con una interconexión de retroalimentación, y un algoritmo de aprendizaje mediante retropropagación. La tercera estructura está dividida en dos redes neurodifusas una estática y otra dinámica con una parte consecuente en variables de estado y proporciona un análisis de estabilidad en el sentido de Lyapunov [4]. La última

estructura muestra una red neurodifusa modificada de la red SONFIN [6] con un algoritmo de entrenamiento por algoritmos genéticos.

Observando la Tabla 5.3 se concluye que el número de parámetros total del modelo SSFNN (parámetros de las funciones membresía y parámetros de los sistemas en espacio de estados locales) a entrenar es más bajo con respecto a los otros métodos. También tanto el error RMS de entrenamiento como de prueba son los más bajos con el mínimo de pasos. Esta tabla muestra la efectividad de dicha estructura. Es importante hacer mención de que existen diferencias estructurales entre los modelos donde el método SSFNN toma ventaja, en comparación a los demás, a partir del análisis de estabilidad, el número de parámetros, y otros tales como el entrenamiento. Aun cuando este último no necesariamente presenta una ventaja.

Tabla 5.3 Comparación del desempeño del método propuesto con otros métodos.

	RSONFIN Juang [7]	RFNN Lee [9]	González-Tang [4]	TRFN-S Juang [6]	SSFNN
Patrones	900	900	-	900	250
Parámetros	36	112	51	33	12
Épocas	100	100	-	10	100
Pasos de entrenamiento	90000	90000	1000	9000	2500
$e_{rms}(entrenamiento)$	0.0248	0.00013	-	0.0084	0.0154
$e_{rms}(prueba)$	0.078	0.00057	0.0418	0.0346	0.0203

1.2 Sistema de tres tanques interconectados

Para verificar el algoritmo del observador y controlador propuesto se utilizará un sistema simulado de tres tanques de agua interconectados por la base, cada tanque cuenta con la medición de altura del agua, alimentados con dos bombas en los tanques uno y tres, y una salida en el tanque tres (Figura 5.8).

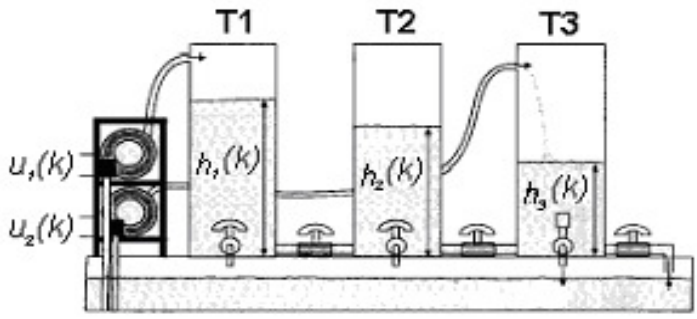

Figura 5.8 Sistema de tres tanques interconectados

Los parámetros a tomar en cuenta para la simulación de los tanques son los siguientes:

$Hm = 60\,cm$	Altura máxima del tanque
$Q_{1,3} = 10^{-3}\,\frac{m^3}{s}$	Flujo de las bombas
$A = 0.0154\,m^2$	Área de sección transversal
$S_n = 0.5\,cm^2$	Diámetro de interconexión
$a_1 = 0.45$	Desgaste entre el tanque 1 y 2
$a_2 = 0.46$	Desgaste entre el tanque 2 y 3
$a_3 = 0.61$	Desgaste de salida

Las ecuaciones de estado del sistema están en función de los flujos de entrada – salida de los tanques.

$$\begin{aligned} A\frac{\partial H_1}{\partial t} &= Q_1 - Q_{12} \\ A\frac{\partial H_2}{\partial t} &= Q_{12} - Q_{23} \\ A\frac{\partial H_3}{\partial t} &= Q_3 + Q_{23} - Q_S \end{aligned} \tag{5.7}$$

Donde:

$$\begin{aligned} Q_{12} &= a_1 S_n \,\mathrm{sgn}(h_1 - h_2)\sqrt{2g|h_1 - h_2|} \\ Q_{23} &= a_2 S_n \,\mathrm{sgn}(h_2 - h_3)\sqrt{2g|h_2 - h_3|} \\ Q_S &= a_3 S_n \,\mathrm{sgn}(h_3)\sqrt{2gh_3} \end{aligned} \tag{5.8}$$

Para modelar el sistema completo a partir solo de las mediciones de las alturas y los flujos de entrada, se requirieron tres redes neurodifusas SSFNN (3.6) definidas a partir del método presentado en la Sección 3.1:

$$\begin{aligned} h_1(k) &= f\left([h_1(k-1) \quad h_2(k-1)], \quad Q_1(k-1)\right) \\ h_2(k) &= f\left([h_2(k-1) \quad h_1(k-1) \quad h_3(k-1)]\right) \\ h_3(k) &= f\left([h_3(k-1) \quad h_2(k-1)], \quad Q_3(k-1)\right) \end{aligned} \tag{5.9}$$

Definiendo las alturas como los estados del sistema y en un esquema serie-paralelo.

$$\begin{aligned} x_1(k+1) &= f_1(x_1(k), x_2(k), u_1(k)) \\ x_2(k+1) &= f_2(x_2(k), x_1(k), x_3(k)) \\ x_3(k+1) &= f_3(x_3(k), x_2(k), u_2(k)) \end{aligned} \tag{5.10}$$

Los datos de entrenamiento son obtenidos mediante la simulación del sistema con un control PID estable buscando llevar al sistema a una región de diseño deseable. (Figura 5.9)

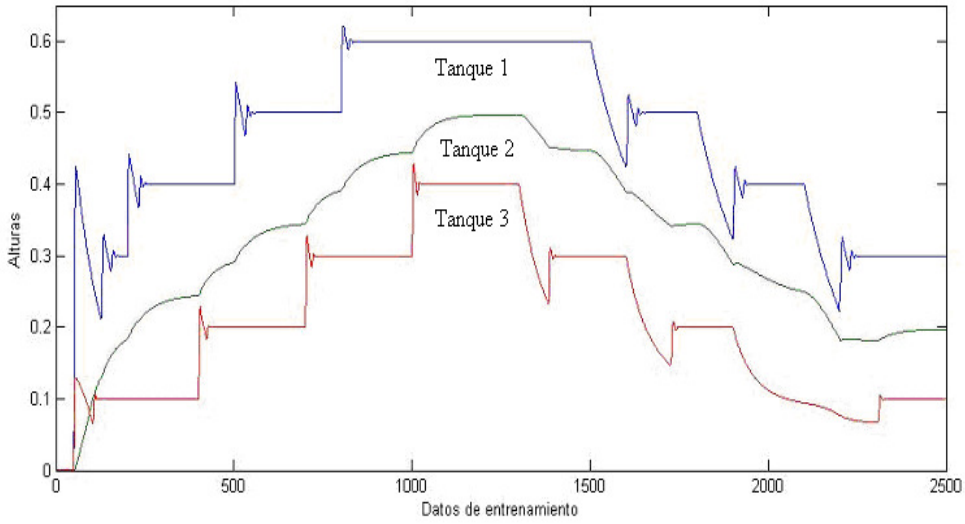

Figura 5.9 Datos de entrenamiento para las tres redes neurodifusas

Los parámetros de entrenamiento que dieron los mejores resultados para el aprendizaje de las tres redes neurodifusas son: el umbral de error $K_e = [0.1 : 0.00001]$, umbral de disparo $K_d = 0.6$, desviación estándar inicial $\sigma_0 = 0.3$, razón de aprendizaje

para las matrices A_j y B_j $N_{a,b} = 0.002$, razón de aprendizaje para los centros de las funciones gaussianas $N_m = 0.0015$ y razón de aprendizaje para las desviaciones estándar $N_s = 0.0015$, entrenadas en 100 épocas. Cabe mencionar que los valores de estos parámetros son heurísticos con base a prueba y error durante el entrenamiento.

Una vez entrenadas las redes, las matrices de la parte consecuente y los parámetros de las funciones gaussianas para los tres tanques son obtenidas. Para el primer tanque se muestran a continuación:

$$\begin{aligned} A_1 &= \begin{bmatrix} 0.9338 & 0.1209 \\ 0.1470 & 0.7266 \end{bmatrix} \qquad B_1 = \begin{bmatrix} 0.00691 \\ -0.00038 \end{bmatrix} \\ A_2 &= \begin{bmatrix} 1.113 & -0.1178 \\ -0.0669 & 1.1867 \end{bmatrix} \quad B_2 = \begin{bmatrix} 0.00598 \\ -0.00048 \end{bmatrix} \\ A_3 &= \begin{bmatrix} 0.8669 & 0.0246 \\ 0.0472 & 0.836 \end{bmatrix} \qquad B_3 = \begin{bmatrix} 0.00649 \\ -0.00017 \end{bmatrix} \end{aligned} \tag{5.11}$$

$$\mu = \begin{bmatrix} gauss(9.2\times10^{-3},0.343) & gauss(0.359,0.324) & gauss(0.596,0.315) \\ gauss(4.1\times10^{-3},0284) & gauss(0.278,0.312) & gauss(0.456,0.320) \end{bmatrix} \tag{5.12}$$

La Figura 5.10 muestra la distribución de las funciones de membresía en el espacio de estado de la red neurodifusa.

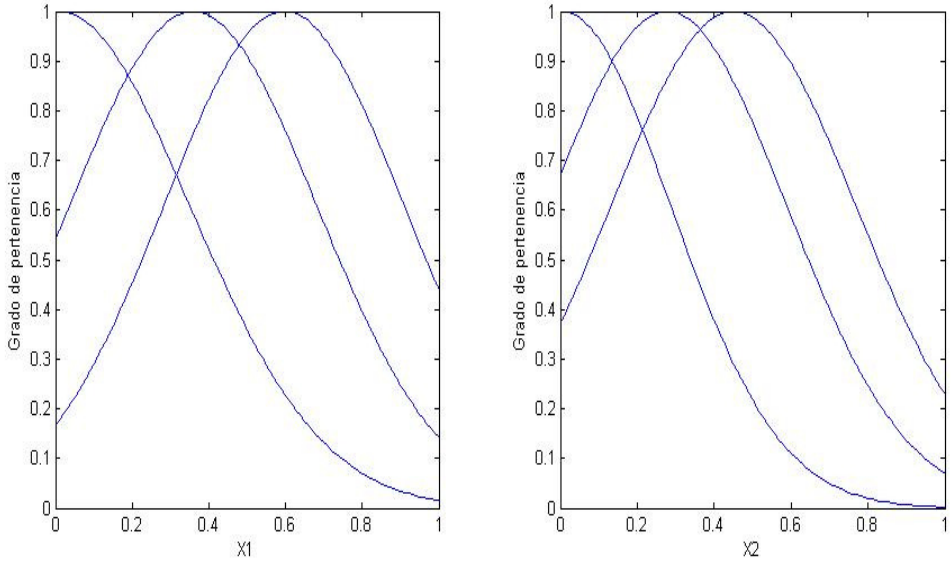

Figura 5.10 funciones de membresía para la red que modela el tanque 1.

El resultado del modelado para el tanque 1 es mostrado en comparación con la altura esperada Figura 5.11a. La distribución de las reglas muestra que todas las reglas son disparadas con una importancia significativa frente a los datos de entrenamiento Figura 5.11b. El error de modelado es mostrado en la Figura 5.11c donde se observa una banda de error del 0.008. En la Figura 5.11d muestra el progreso del error RMS durante el entrenamiento de la red neurodifusa con un error RMS final $e_{rms} = 0.0089$.

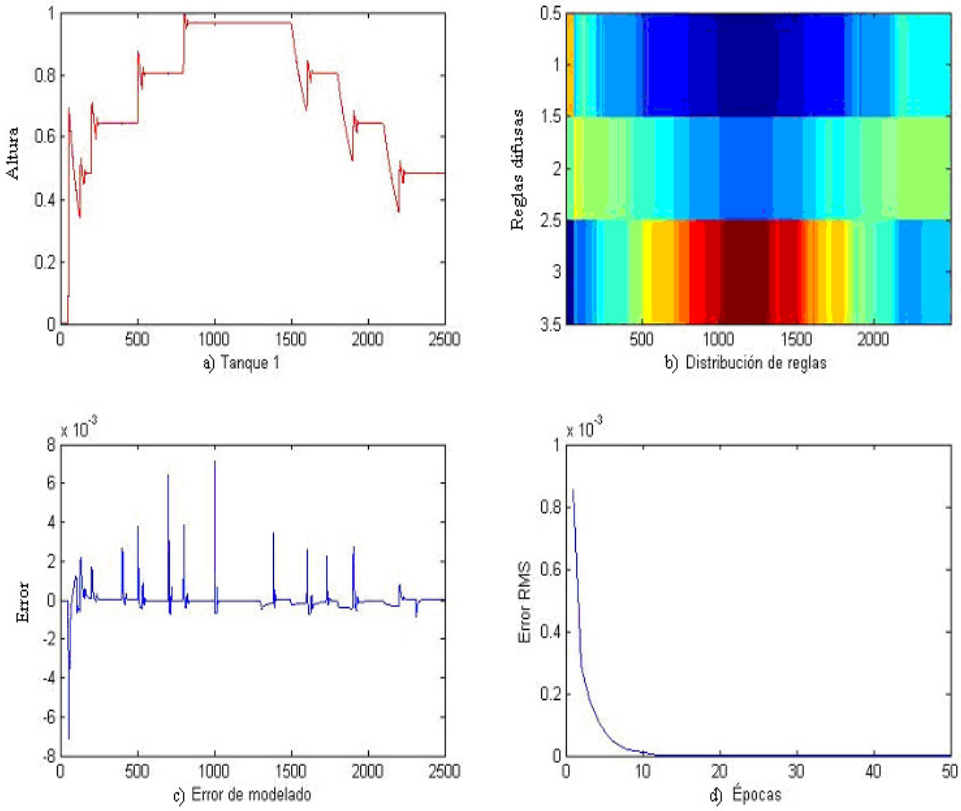

Figura 5.11 Desempeño de la red para el modelado del tanque 1.

La Figura 5.12 muestra los datos de entrenamiento (–) y los datos obtenidos durante el modelo (– –), donde casi no se observa diferencia aun cuando hay componentes de alta frecuencia.

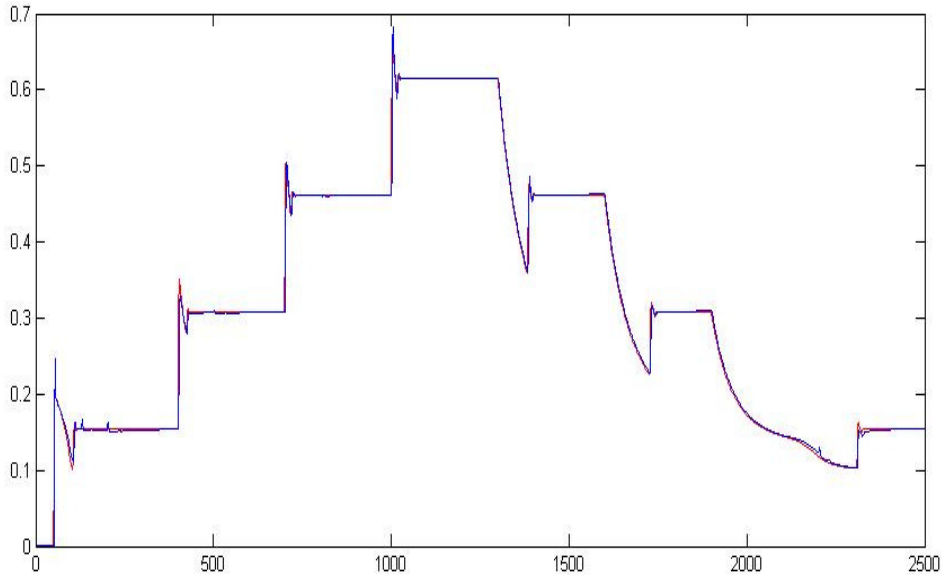

Figura 5.12 Salida deseada (–) y salida modelada (– –) para el tanque 1.

En el segundo tanque se contempla que no se tiene entrada al modelo solo tres estados internos, por lo cual solo se tienen las matrices A_j para las tres reglas difusas generadas, junto con sus respectivas funciones de membresía μ.

$$A_1 = \begin{bmatrix} 0.7868 & 0.1208 & 0.1061 \\ 0.0825 & 0.8626 & 0.1847 \\ 0.0389 & 0.2244 & 0.2866 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 1.1616 & -0.0824 & -0.1010 \\ -0.0908 & 1.0881 & -0.0261 \\ -0.0306 & -0.0727 & 1.2471 \end{bmatrix} \qquad (5.13)$$

$$A_3 = \begin{bmatrix} 0.7845 & 0.1039 & 0.1296 \\ 0.0554 & 0.9212 & 0.0353 \\ -0.0047 & 0.0505 & 0.8708 \end{bmatrix}$$

$$\mu = \begin{bmatrix} gauss(6.3\times10^{-3},0.249) & gauss(0.274,0.245) & gauss(0.543,0.322) \\ gauss(7.2\times10^{-3},0.329) & gauss(0.323,0.289) & gauss(0.612,0.287) \\ gauss(4.5\times10^{-3},0.363) & gauss(0.196,0.329) & gauss(0.391,0.325) \end{bmatrix} \quad (5.14)$$

La Figura 5.13 muestra la distribución de las funciones de membresía en el espacio de estado de la red neurodifusa.

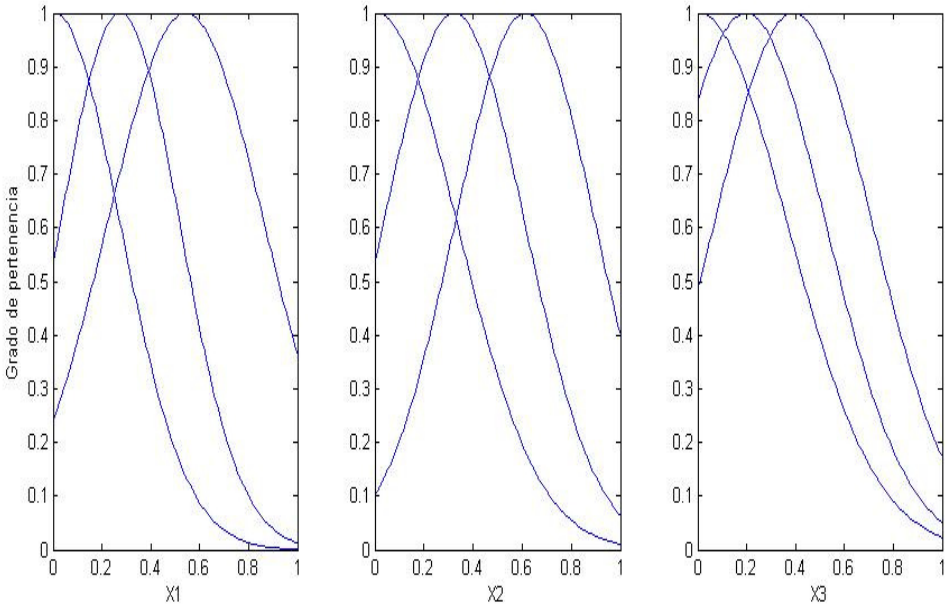

Figura 5.13 funciones de membresía para la red que modela el tanque 2.

El resultado del modelado para el tanque 2 es mostrado en comparación con la altura esperada Figura 5.14a. La distribución de las reglas muestra que todas las reglas son disparadas con una importancia significante frente a los datos de entrenamiento Figura 5.14b. El error de modelado es mostrado en la Figura 5.14c donde se observa una banda de error del 0.005. En la Figura 5.14d muestra el progreso del error RMS durante el entrenamiento de la red neurodifusa con un error RMS final $e_{rms} = 0.0023$.

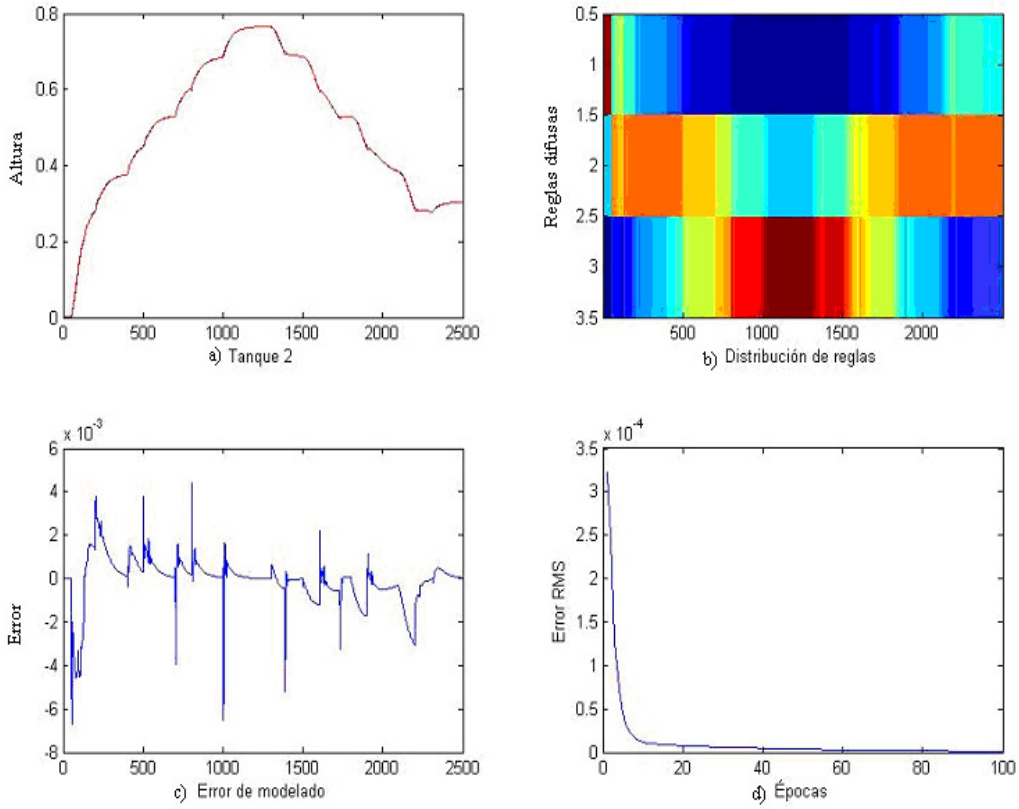

Figura 5.14 Desempeño de la red para el modelado del tanque 2.

La muestra los datos de entrenamiento (–) y los datos obtenidos durante el modelo (– –), donde casi no se observa diferencia aun cuando hay componentes de alta frecuencia.

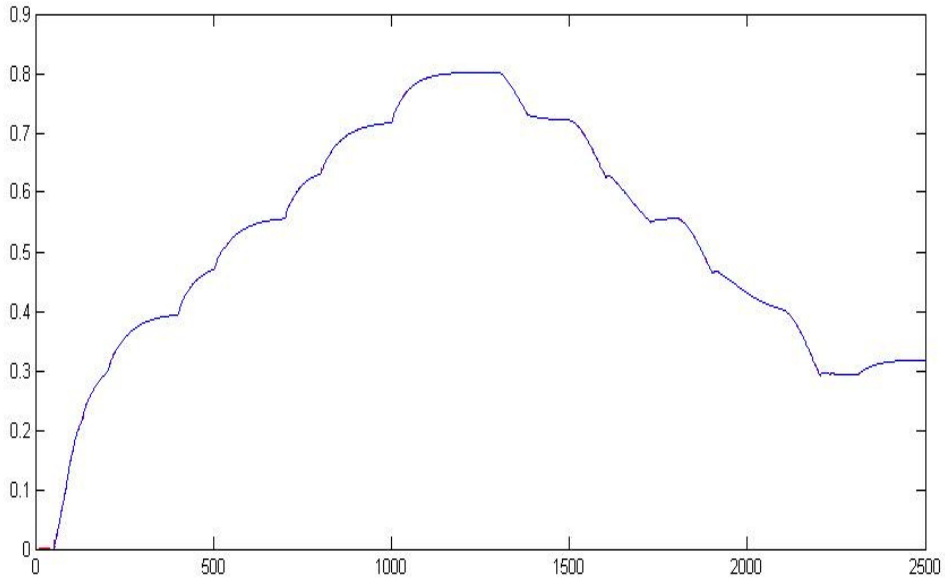

Figura 5.15 Salida deseada (–) y salida modelada (– –) para el tanque 2.

Para el tanque 3 también se generaron tres reglas difusas donde la distribución de las funciones de membresía se muestra en la Figura 5.16.

$$A_1 = \begin{bmatrix} 0.9338 & 0.1209 \\ 0.1470 & 0.7266 \end{bmatrix} \qquad B_1 = \begin{bmatrix} 0.00691 \\ -0.00038 \end{bmatrix}$$
$$A_2 = \begin{bmatrix} 1.113 & -0.1178 \\ -0.0669 & 1.1867 \end{bmatrix} \quad B_2 = \begin{bmatrix} 0.00598 \\ -0.00048 \end{bmatrix} \tag{5.15}$$
$$A_3 = \begin{bmatrix} 0.8669 & 0.0246 \\ 0.0472 & 0.836 \end{bmatrix} \qquad B_3 = \begin{bmatrix} 0.00649 \\ -0.00017 \end{bmatrix}$$

$$\mu = \begin{bmatrix} gauss(9.2\times10^{-3},0.343) & gauss(0.359,0.324) & gauss(0.596,0.315) \\ gauss(4.1\times10^{-3},0284) & gauss(0.278,0.312) & gauss(0.456,0.320) \end{bmatrix} \tag{5.16}$$

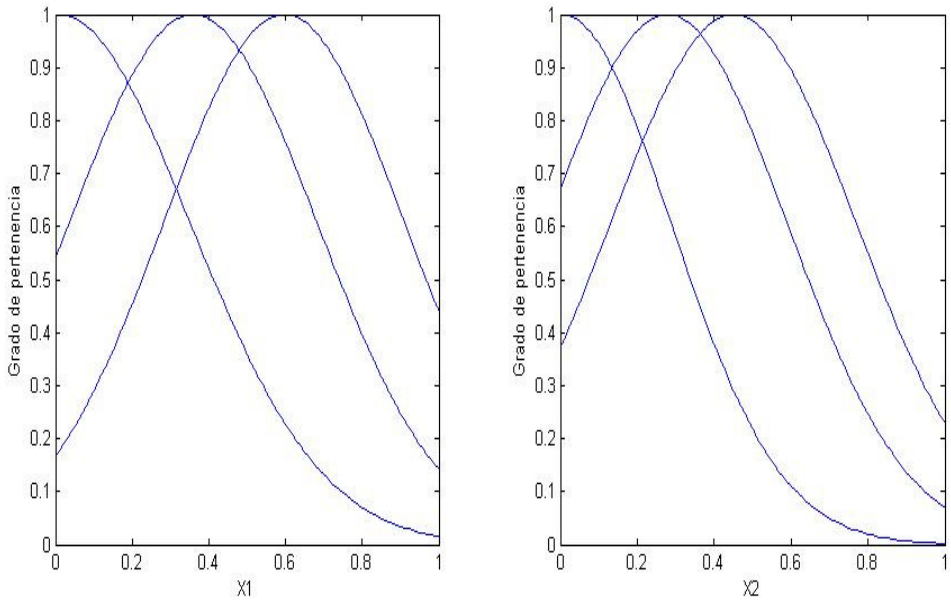

Figura 5.16 funciones de membresía para la red que modela el tanque 1.

El resultado del modelado para el tanque 3 es mostrado en comparación con la altura esperada Figura 5.17a. La distribución de las reglas muestra que todas las reglas son disparadas con una importancia significativa frente a los datos de entrenamiento Figura 5.17b. El error de modelado es mostrado en la Figura 5.17c donde se observa una banda de error del 0.01. En la Figura 5.17d muestra el progreso del error RMS durante el entrenamiento de la red neurodifusa con un error RMS final $e_{rms} = 0.0274$.

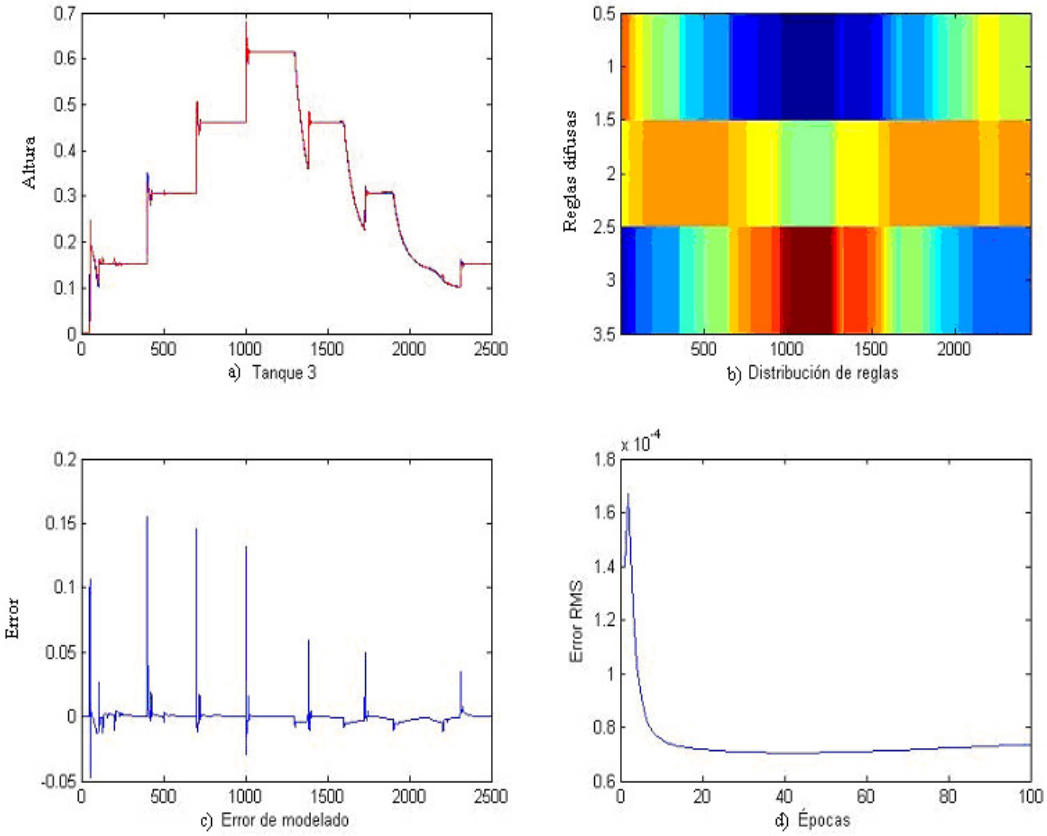

Figura 5.17 Desempeño de la red para el modelado del tanque 3.

La Figura 5.18 muestra los datos de entrenamiento (–) y los datos obtenidos durante el modelo (– –), donde casi no se observa diferencia aun cuando hay componentes de alta frecuencia.

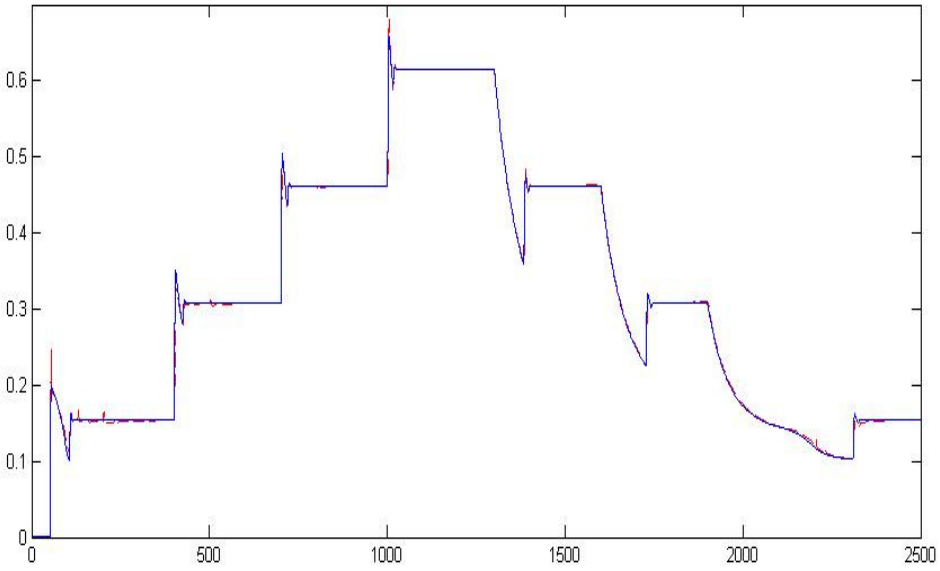

Figura 5.18 Salida deseada (–) y salida modelada (– –) para el tanque 3.

Análisis de estabilidad

Las magnitudes de los valores característicos para todas las matrices A_j para los tres tanques, no cumplen con el Teorema 4.1 para el análisis de estabilidad asintótica del las tres redes neurodifusas. Esto debido a que al menos una de las matrices para cada uno de los tres tanques es mayor a uno.

$$\begin{aligned} Tanque\,1 &= \{0.9996 \quad 1.2495 \quad 0.8906\} \\ Tanque\,2 &= \{0.9966 \quad 0.9960 \quad 1.2836\} \\ Tanque\,3 &= \{0.8945 \quad 1.8404 \quad 0.9237\} \end{aligned} \tag{5.17}$$

1.2.1 Observador

Una vez diseñado los modelos difusos para el sistema, estos al ser entrenado en un esquema serie-paralelo no necesitan tener la propiedad de ser estable para que identifiquen al sistema, pero si los modelos difusos se requieren para tareas de simulación o para ser

empleados en un esquema paralelo es necesario que los modelos sean estables para que los estados internos tiendan a los estados reales.

Debido a que los parámetros de las matrices A_j son entrenados sin tener en cuenta si las matrices resultantes son estables o inestables, el modelado del sistema tiene que repetirse si al menos una matriz es inestable o no se cumple el Teorema 4.1.

Una alternativa es el diseño del observador difuso (4.3), (4.4) tal que cumpla con el Teorema 4.1 en función de los modelos obtenidos en la Sección 3.1, Este observador permite que los estados internos sean asintóticamente estables. Dando posibilidad de utilizar el observador difuso para simulación o para realizar alguna acción de control.

Para el caso de los tres tanques interconectados los tres modelos neurodifusos obtenidos son inestables, por lo que es necesario el diseño de observadores tal que cumplan con el Teorema 4.1 y permitan realizar la interconexión entre observadores y simular el sistema MIMO a partir de tres SISO (Figura 5.19).

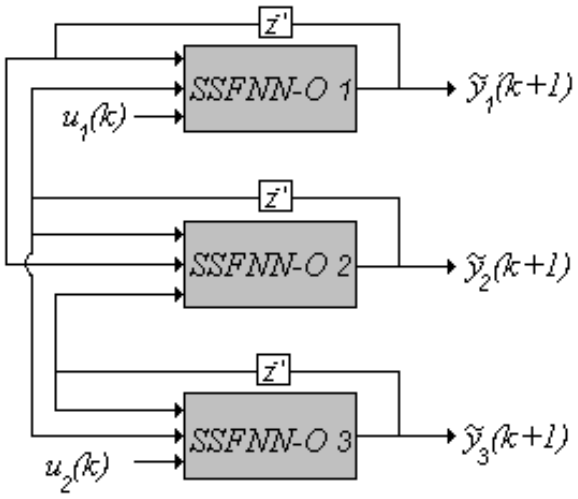

Figura 5.19 Interconexión de observadores.

Una vez obtenidas las matrices A_j para los tres tanques se diseñaron los observadores para que la dinámica tuviera eigenvalores que cumplieran con el Teorema 4.1.

Tabla 5.4 Eigenvalores propuesto para la dinámica de los observadores

	Reglas 1	*Reglas* 2	*Reglas* 3
λ_1	$(0.061 \quad 0.199)$	$(0.524 \quad 0.523)$	$(-0.188 \quad -0.114)$
λ_2	$(0.72 \quad 0.49 \quad 0.22)$	$(0.6 \quad 0.65 \quad 0.67)$	$(0.62 \quad 0.67 \quad 0.68)$
λ_3	$(0.48 \quad 0.58)$	$(0.4 \quad 0.5)$	$(0.33 \quad 0.49)$

Las matrices de retroalimentación L_j para cada uno de los modelos son diseñadas a partir de la formula de Ackerman de tal forma que tengan una respuesta satisfactoria para el seguimiento de los estados del sistema.

Tabla 5.5 Ganancias de los observadores

	Regla 1	*Regla* 2	*Regla* 3
L_1	$(1.4 \quad 3)$	$(1.22 \quad -3.62)$	$(2 \quad 39.6)$
L_2	$(0.5 \quad 0.65 \quad 0.23)$	$(1.59 \quad 6.11 \quad -13.6)$	$(0.6 \quad 1.66 \quad 0.01)$
L_3	$(0.61 \quad 1.36)$	$(1.46 \quad -8.6)$	$(0.8 \quad 4.18)$

De acuerdo a la colocación de los eigenvalores de todas las matrices A_j se cumple con el Teorema 4.1 y por lo tanto los tres observadores son estables asintóticamente en todo el rango del espacio de los estados internos.

una vez diseñados los observadores se interconectan y se realiza la simulación del sistema. Las referencias (Figura 5.21a) son aplicadas al controlador PID, las señales de control son aplicadas al modelo y al sistema en un esquema paralelo. Las respuestas de los observadores son estables y siguen a las salidas del sistema (Figura 5.21b); siendo los errores de observación pequeños $e \in [-0.1, 0.1]$ (Figura 5.21c) con errores RMS para los tres observadores de $e^1_{rms} = 0.0123$, $e^2_{rms} = 0.0016$ y $e^3_{rms} = 0.0072$.

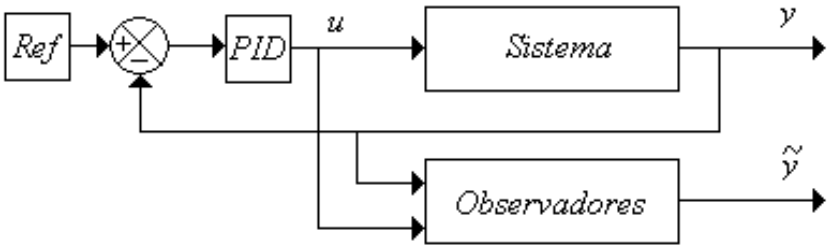

Figura 5.20 Esquema paralelo para los observadores interconectados

Una vez probados los observadores con los datos de entrenamiento como se muestra en la Figura 5.20 se proporcionan datos de prueba para mostrar la eficiencia de los observadores a escenarios desconocidos. El escenario de prueba es mostrado en la Figura 5.22a donde se contemplan diferentes puntos de equilibrio del sistema a los de entrenamiento, la respuesta de los observadores es mostrada en la Figura 5.22b donde el error de observación esta en un rango de $e \in [-0.1,0.1]$, con errores RMS de $e^1_{rms} = 0.0102$, $e^2_{rms} = 0.0015$ y $e^3_{rms} = 0.0044$.

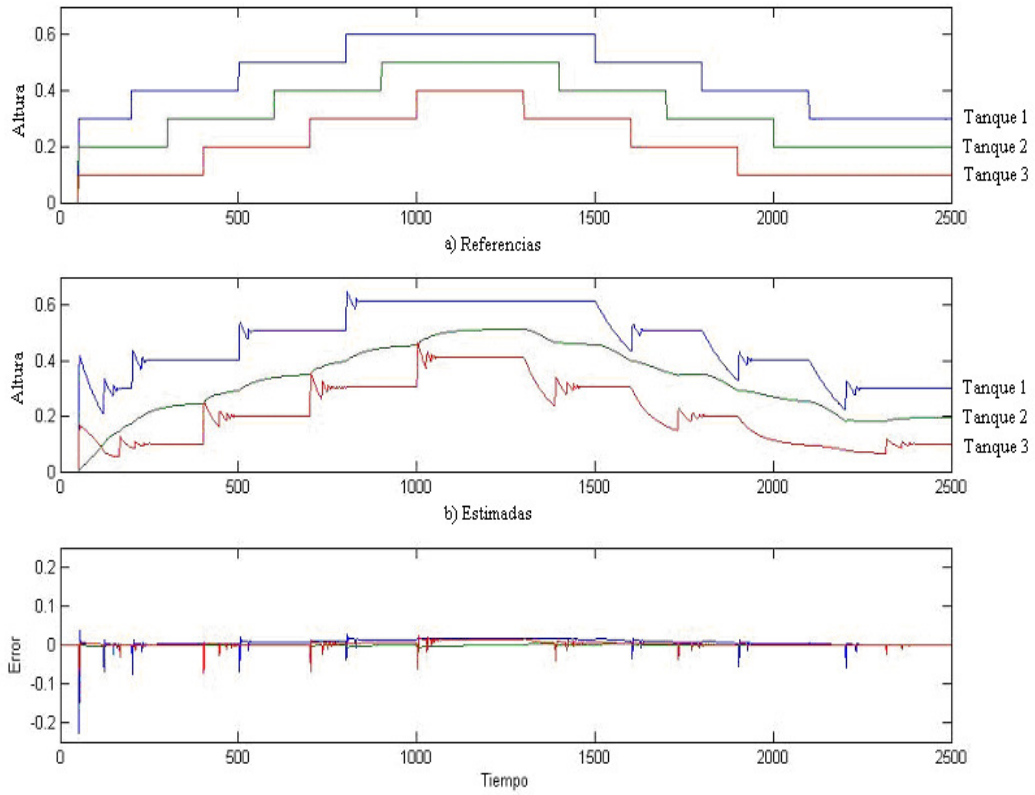

Figura 5.21 Respuesta de los observadores interconectados al escenario de entrenamiento.

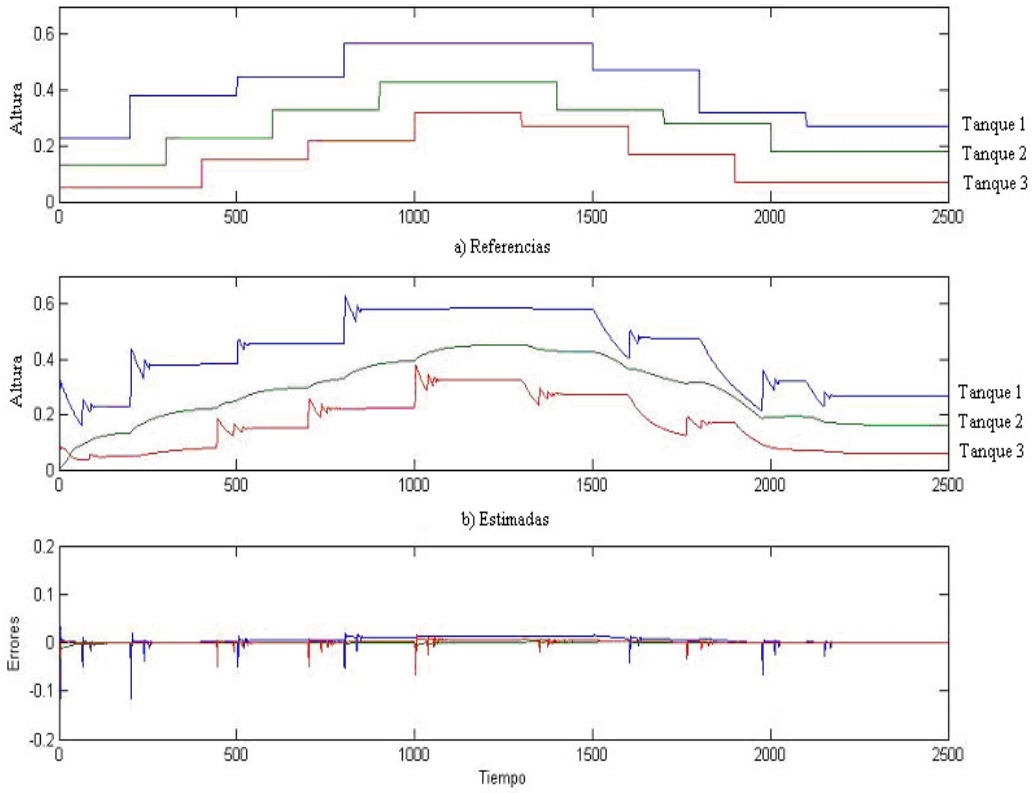

Figura 5.22 Respuesta de los observadores a un escenario desconocido.

1.2.2 Simulación del Controlador

Una vez que se tienen los observadores difusos son empleados para cumplir con un objetivo de control que es el seguimiento de una señal de referencia. Primero se muestra el desempeño del sistema mediante el escenario de entrenamiento para los observadores, y posteriormente se aplica un escenario de prueba para mostrar la efectividad de los observadores.

Se controlan las alturas de los tanques 1 y 3, por lo que se requieren las referencias de estos dos tanques y se controla el sistema con forme a la Figura 5.23.

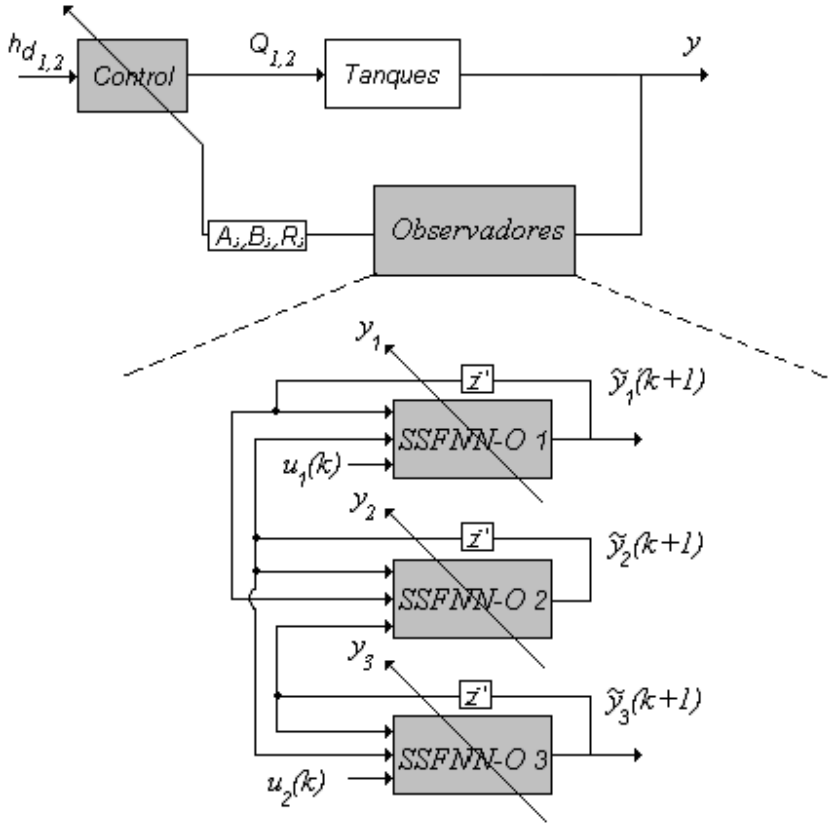

Figura 5.23 Esquema de control

$$u_c(\tilde{x}) = (\overline{B})^{-1}\left[-\overline{A}\tilde{x}(k) - \overline{L}(y(k) - \tilde{y}(k)) + C^{-1}(y_d(k+1) + \beta e_c(k))\right] \tag{5.18}$$

De acuerdo a (5.18) el parámetro de diseño es la elección de una constante $0 < \beta < 1$ para que el sistema sea estable. Dependiendo de la elección de β será la respuesta del sistema. La Figura 5.24 muestra el comportamiento del controlador basado en los observadores SSFNN obtenidos en la sección anterior, tomando una $\beta_1 = 0.8$ para el controlador del tanque 1 y $\beta_2 = 0.87$ para el tanque 3 que dieron los mejores resultados de una serie de pruebas. Se observa que el control es aceptable y estable mejor que la obtenida por el controlador PID que se diseño para obtener los patrones de entrenamiento. Una vez probado el controlador se aplica un patrón de prueba para observar el comportamiento del controlador a un escenario desconocido (Figura 5.25), donde se observa que el comportamiento sigue siendo estable y un seguimiento satisfactorio.

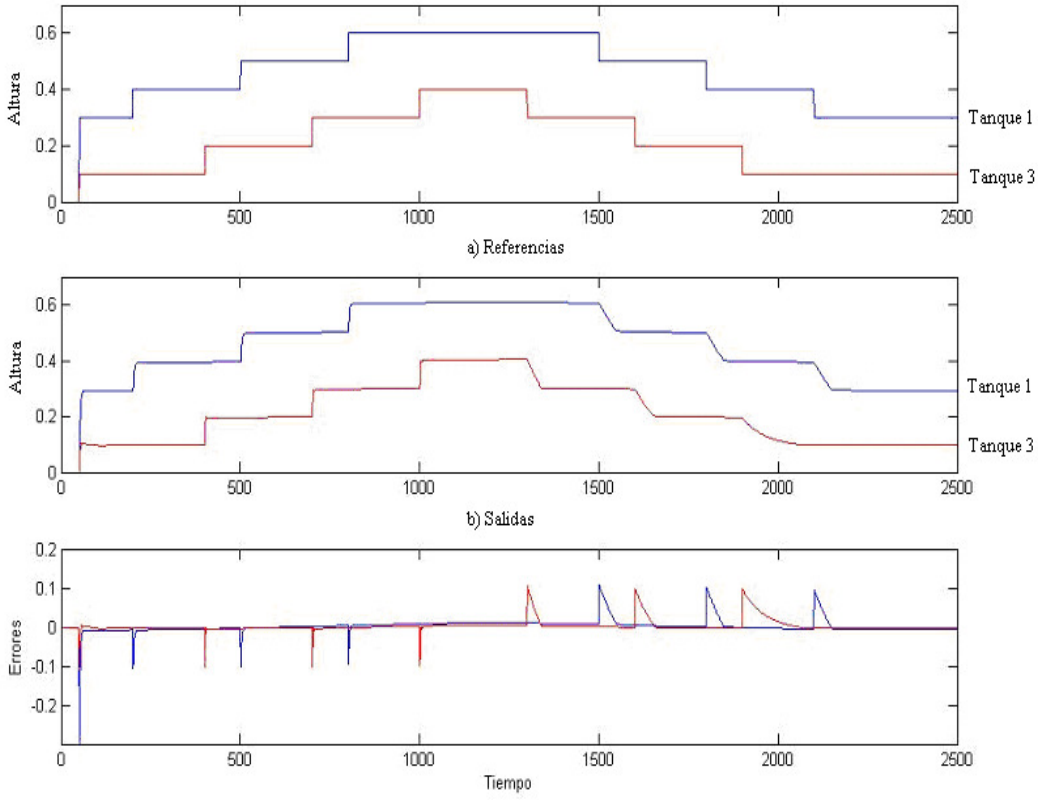

Figura 5.24 Respuesta del sistema al patrón de entrenamiento de los observadores

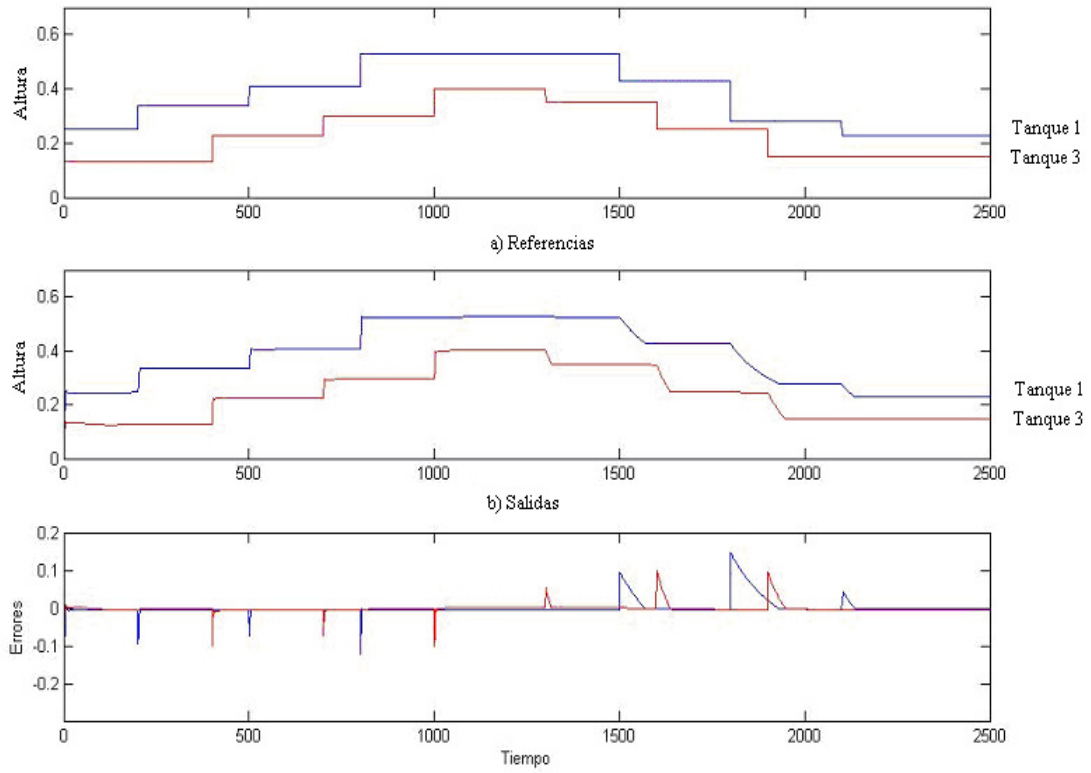

Figura 5.25 Respuesta del sistema a un patrón de prueba.

Una vez probado el sistema a un escenario desconocido solo resta ver el comportamiento del controlador a diversos valores de β (Figura 5.26, Figura 5.27) donde se observa que entre un valor más alto de β se tiene un error en estado estable mínimo y un tiempo de respuesta pequeño pero con la contraparte de que se tiene un sobrepaso, a un valor bajo de β no se tiene sobrepaso pero se tiene un error en estado estable considerable.

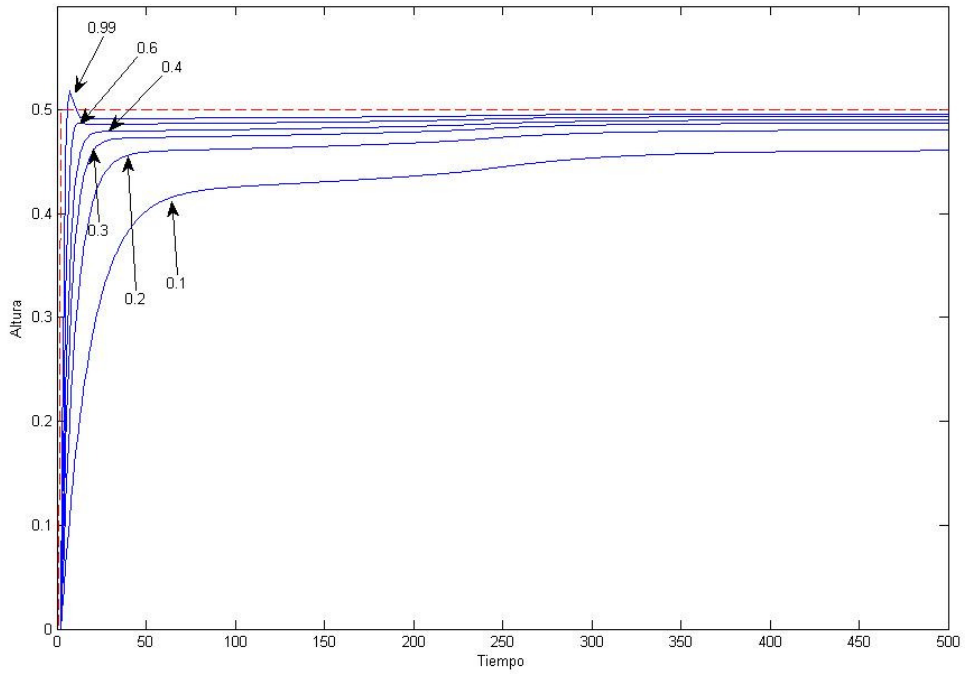

Figura 5.26 Comportamiento del controlador a diversas β para el tanque 1 a una altura de 0.5.

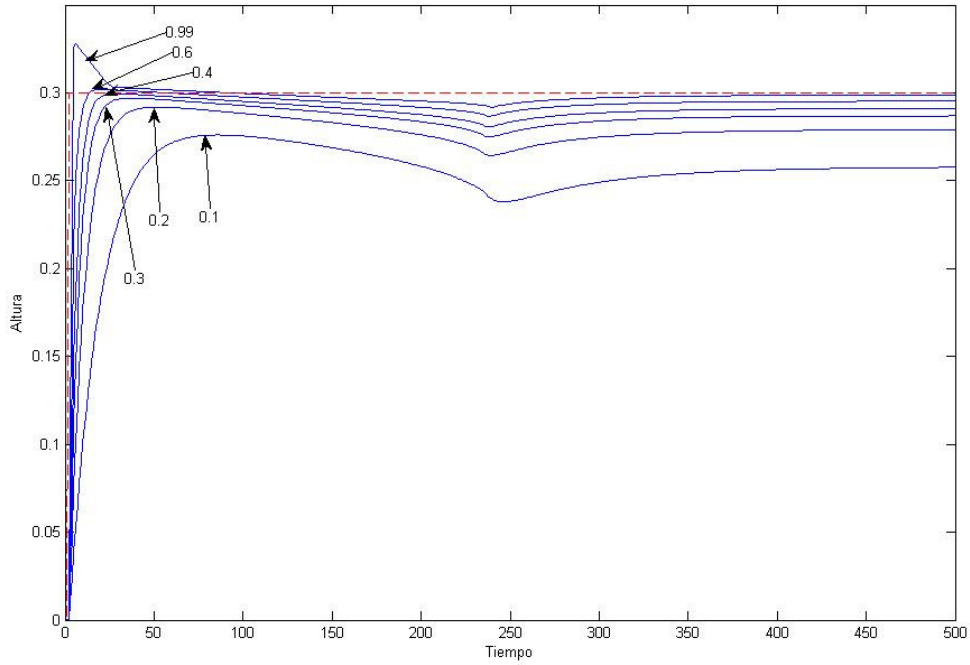

Figura 5.27 Comportamiento del controlador a diversas β para el tanque 3 a una altura de 0.3.

CONCLUSIONES

Se estableció una estructura de una red neurodifusa con la capacidad de obtener modelos locales en variables de estado permitiendo el análisis de estabilidad.

Además, los parámetros iniciales del aprendizaje de la estructura de la red neurodifusa SSFNN son determinados de manera heurística, por lo que no se requiere información del sistema.

Dicha red puede modelar sistemas SISO donde se contemplan señales que afectan al sistema vistas como estados internos de la red y una entrada controlada. Dada está propiedad, se pueden interconectar múltiples redes para poder modelar sistemas MIMO con tantas redes SISO como salidas se requiera.

El análisis de estabilidad del observador interconectado es asegurado y está basado en el primer método de Lyapunov, su implementación puede ser llevada a un problema LMI (*Linear Matrices Inequalities)* con tantas desigualdades como reglas difusas.

Como trabajo futuro es establecer una estructura para la red propuesta modelando un sistema MIMO. Así mismo, establecer mejorías en las actualizaciones y el método de agrupamiento para la generación de reglas de la red neurodifusa en línea.

BIBLIOGRAFIA

[1] Abonyi J., Fuzzy Model Identification for Control, Birkhäuser, 2003.

[2] Chen C. T., Linear System: Theory and Design, Oxford University Press, 1999.

[3] Chen D. S. and Jain R.C. "A robust back propagation learning algorithm for function approximation", IEEE Trans. Neural Networks, vol. 5 (3), pp. 467-479. May. 1994.

[4] González M. A., Tang Y. "Una nueva estructura basada en redes neuronales parcialmente recurrentes para la identificación de sistemas físicos", AMCA 2005, CD-ROM.

[5] Jang J. S. R. "Anfis: Adaptive network based fuzzy inference system", IEEE Trans. on Systems, Man and Cybernetics, vol. 23, pp. 665-685. May. 1993.

[6] Juang F. C. "A TSK-Type Recurrent Fuzzy Network for Dynamic Systems Processing by Neural Networks and Genetic Algorithms", IEEE Trans. On Fuzzy Systems, vol. 10-2, pp. 155-170. Apr. 2002.

[7] Juang C. F., Lin C. T. « A Recurrent Self- Organizing Neural Fuzzy Inference Network", IEEE Trans. On Neural Networks, vol. 10-4, pp.828-845. Jul, 1994.

[8] Kosko B. "Fuzzy systems as universal approximator", IEEE Trans. Computers, vol. 43, pp. 1329-1333. Nov. 1994.

[9] Lee C. H., Teng C. C. "Identification and Control of Dynamic Systems using Recurrent Fuzzy Neural Networks", IEEE Trans. On Fuzzy Systems, vol. 8-4, pp. 349-366. Ago. 2000.

[10] Leng G., Prasad G., McGinnity T. M. "A New Approach to Generate a Self-Organizing Fuzzy Neural Model", IEEE Trans. on Systems, Man and Cybernetics, vol. 4. Oct. 2002.

[11] Leng G., Prasad G., McGinnity T. M. "An On-Line Algorithm for Creating Self-Organizing Fuzzy Neural Networks", Neural networks, vol. 17, pp. 1477-1493. Jul 2004.

[12] Lung L., Glad T., Modeling of Dynamic Systems, PTR Prentice Hall, 1994.

[13] Mastorocostas P. A., Theocharis J. B. “A recurrent fuzzy neural model for dynamic system identification”. IEEE Trans. on Systems, Man and Cybenetics. vol. 32 (2), pp. 176-190. Apr. 2002.

[14] McCulloch W. S., Pitts W. “A logical calculus of the Ideas Immanent in Nervous Activity”, Bull. Of Math. Biophysics, vol.5, pp. 115-133. 1943.

[15] Nelles O., Nonlinear System Identification: from classical approaches to neural networks and fuzzy models, Springer Verlag, 2001.

[16] Norgaard M., Ravn O., Neural Networks for Modeling and Control of Dynamic Systems, Springer Verlag, 2001.

[17] Nugyen H. y Prasad N. R., A first course in fuzzy and neural control, Chapman and Hall/CRC, 2003.

[18] Rummelhardt D.E., Hinton G. E., Williams R. J., Learning Internal Representations by Error Propagation. MIT Press, 1986.

[19] Stanislaw H, Systems and Control, Oxford University Press, 2003.

[20] Sugeno M., Kang G. T. “Structure identification of fuzzy model”. Fuzzy sets and systems, vol. 28 (1), pp. 15-33, Oct. 1988.

[21] Takagi, T., Sugeno M. “Fuzzy identification of systems and its applications to modeling and control” IEEE Trans. on Systems, Man and Cybernetics, vol. 15 (1), pp. 116-132. 1985.

[22] Tanaka K. “Stability and stabilizability of fuzzy-neural-linear control systems”. IEEE Transactions on fuzzy systems, vol. 3 (4), pp.438-447. Nov. 1995.

[23] Tanaka K., Sugeno M. “Stability analysis and design of fuzzy control systems. Fuzzy sets and systems, vol. 45 (2), pp.135-156. Jun. 1992.

[24] Wang L. X., Adaptive fuzzy systems and control: Design and stability analysis. Englewood Cliffs, Prentice Hall, 1994.

[25] Wang Y. C., Chien C. J., Teng C. C. “Takagi Sugeno recurrent fuzzy neural networks for identification and control of dynamic systems”, IEEE International Fuzzy Systems Conference, pp. 537-540, 2001.

[26] Williams R. J., Zipser D. “A learning algorithm for continually running fully recurrent neural networks” Neural computation, vol. 1 (2), pp.270-280, 1989.

[27] Wu S., Er M. J., Gao Y. “A Fast Approach for automatic generation of fuzzy rules by generalized dynamic fuzzy neural networks”, IEEE Trans. Fuzzy Systems, vol. 4 (4), pp.578-598, Ago. 2001.

RECONOCIMIENTOS

Se agradece el apoyo de UNAM-PAPIIT (IN106100 y IN105303).