UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA

FACULTAD DE INGENIERIA

IDENTIFICACIÓN DE SISTEMAS DINÁMICOS MEDIANTE UNA RED NEURODIFUSA RECURRENTE

T E S I S

QUE PARA OPTAR POR EL GRADO DE

MAESTRO EN INGENIERÍA

INGENIERÍA ELÉCTRICA – CONTROL

P R E S E N T A :

MARCOS ANGEL GONZÁLEZ OLVERA

TUTOR:
DR. YU TANG XU

2005

Jurado:

Presidente: Dr. Gerardo René Espinosa Pérez

Secretario: Dr. Héctor Benítez Pérez

Vocal: Dr. Yu Tang Xu

1er Suplente: Dr. Luis Agustín Álvarez-Icaza Longoria

2do Suplente: Dr. Leonid Fridman

Lugar o lugares donde se realizó la tesis: Facultad de Ingeniería

TUTOR DE TESIS

Dr. Yu Tang Xu

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Motivación

En muchas ramas del conocimiento encontramos problemas en los cuales el poseer un modelo del sistema estudiado puede ser de gran utilidad. Por ejemplo, en economía es útil poseer modelos que describan el comportamiento de variables tales como tasas de interés, precios, niveles de oferta, etcétera. Por otro lado, en los diversos campos de la ingeniería los modelos son muy útiles para poder analizar y controlar los comportamientos de los sistemas analizados.

Existen varios tipos de modelos que pueden ser empleados: probabilísticos, analíticos, heurísticos, etcétera. La principal utilidad de ellos es que nos pueden ayudar a conocer más a fondo el comportamiento del sistema y así facilitar su análisis y descripción, e incluso ser auxiliares para la predicción del comportamiento del sistema. Es más, el poseer un modelo del sistema incluso nos facilita el proponer esquemas prescriptivos, es decir métodos mediante los cuales podamos obtener un desempeño deseado del sistema modelado, ya sea a través de realimentación o sin ésta.

Este es precisamente el objetivo, aunque no privativo, de la Teoría del Control, en donde por lo general un modelo de un sistema dinámico nos ayuda en gran medida para poder proponer un sistema de control tal que permita obtener una respuesta deseada de un sistema.

Así como en el Control, también en otros campos de la ingeniería los modelos son útiles para solucionar problemas relacionados con la detección de fallas, diseño de nuevos procesos, optimización y pruebas mediante simulación, entre otros. La calidad de la solución propuesta en cada uno de estos campos de estudio depende típicamente del modelo que se

haya empleado, y es ahí en donde surge la necesidad de contar con esquemas de modelado e identificación que permitan lograr mejores desempeños en las soluciones.

Modelos lineales contra no lineales

De acuerdo con la forma en la cual afecten las señales y los parámetros al comportamiento de un sistema, será el tipo de esquema o modelo que se empleará. En general los esquemas considerados para atacar el problema de modelado e identificación se pueden dividir en dos tipos: Lineales y No lineales. La elección de alguno de estos tipos de modelos depende del problema que se esté atacando, y la correcta elección de alguno de ellos determinará la calidad del modelo final.

En la determinación de los modelos, se debe considerar si se tomará en cuenta la evolución del comportamiento del sistema con respecto al tiempo o no. En primer lugar, se pueden considerar estructuras "estáticas" representadas por funciones sin memoria, en donde la salida de la función depende únicamente de la entrada actual. Por otro lado se tienen las estructuras "dinámicas" o "recurrentes", representadas por funciones con memoria, donde la salida actual depende tanto de la entrada actual como de valores pasados de la entrada y la salida. Este tipo de esquemas son los más utilizados para modelar sistemas dinámicos, en donde los valores de las señales de interés dependen de valores pasados de éstas.

Al tratar con sistemas dinámicos por lo general la primera opción a considerar para modelarle puede ser un sistema lineal, en vista de que la teoría bajo este esquema es más sencilla y se encuentra mucho más estudiada al compararse con el caso de sistemas no lineales.

Sin embargo, si el empleo de un modelo lineal no satisface los requisitos de exactitud deseados, el siguiente paso será migrar a un esquema no lineal con parámetros lineales. En vista de que para modelar un sistema no lineal no existe una única estructura, la topología de la función que le modele puede ser obtenida de varias formas: mediante desarrollo analítico de los elementos que componen al sistema y las interacciones entre ellos, por funciones genéricas (desarrollo en series de Taylor, representación polinomial, *onduletas*[*], etcétera) o modelos heurísticos.

Una de las topologías que en últimas fechas ha cobrado mayor atención es la basada en redes neuronales y redes neurodifusas. Las primeras son estructuras matemáticas basadas en estructuras biológicas neuronales, en donde elementos muy sencillos de cómputo son

[*]Mejor conocidas por su nombre en inglés: *wavelets*

capaces de lograr procesamientos de información muy eficientes en paralelo con base en las interconexiones entre ellos. Por otra parte, las segundas están inspiradas en el mismo modelo, pero además consideran conceptos de la teoría de conjuntos difusos creada por Lofti Zadeh[27], inspirada en los procesos de pensamiento y toma de decisiones humanos.

La ventaja de utilizar este tipo de estructuras es que, así como los modelos biológicos en los cuales están inspiradas, son capaces de ser sujetas a un proceso de "aprendizaje" o "entrenamiento", tras el cual podrán reproducir el comportamiento dinámico del proceso para el cual fueron entrenadas. Sobre este último punto existe un compromiso entre la complejidad de la red, la dificultad de la tarea a realizar y la capacidad que tengamos de entrenarla: para tareas sencillas y con un entrenamiento modesto, una red de baja complejidad podrá desempeñarse bien; mientras que para procesos complejos donde sea necesario un buen desempeño, será necesario contar con redes más amplias, así como procesos de entrenamiento más intensivos.

Modelos estáticos contra dinámicos

Para modelar sistemas dinámicos existen dos esquemas: modelo paralelo y serie-paralelo [24][17]. En la Figura 1.2 se muestra la implementación de este tipo de modelos. El modelo serie-paralelo es típicamente una función estática, utilizada como un "predictor" en donde se estimará (tras ser entrenado con datos obtenidos del sistema real) con base en las señales de salida y entrada previas del sistema real, la o las señales de salida que dará el sistema en un futuro. Este tipo de esquema de modelado puede ser empleado en procesos en los cuales el modelo sea empleado como una referencia con respecto al comportamiento del sistema real, con la desventaja de no poder ser utilizado de forma independiente; esto es, que permita "simular" el comportamiento del sistema ante una cierta excitación externa sin necesidad de excitar al sistema real. En contraste, un modelo paralelo, de naturaleza dinámica, permite que con base en únicamente la información de la entrada de la planta, se pueda reproducir (o aproximar) el comportamiento de ésta.

En muchos casos, como se comentaba al principio, es de gran utilidad (cuando no necesario) contar con este tipo de *modelos de simulación* que permitan evaluar cómo se comportará el sistema real ante una cierta excitación, y muy especialmente cuando se trata de procesos en los cuales esta condición de excitación es difícil de ser aplicada. Por ejemplo, si deseamos diseñar un control para un reactor químico, siempre será mejor contar con el

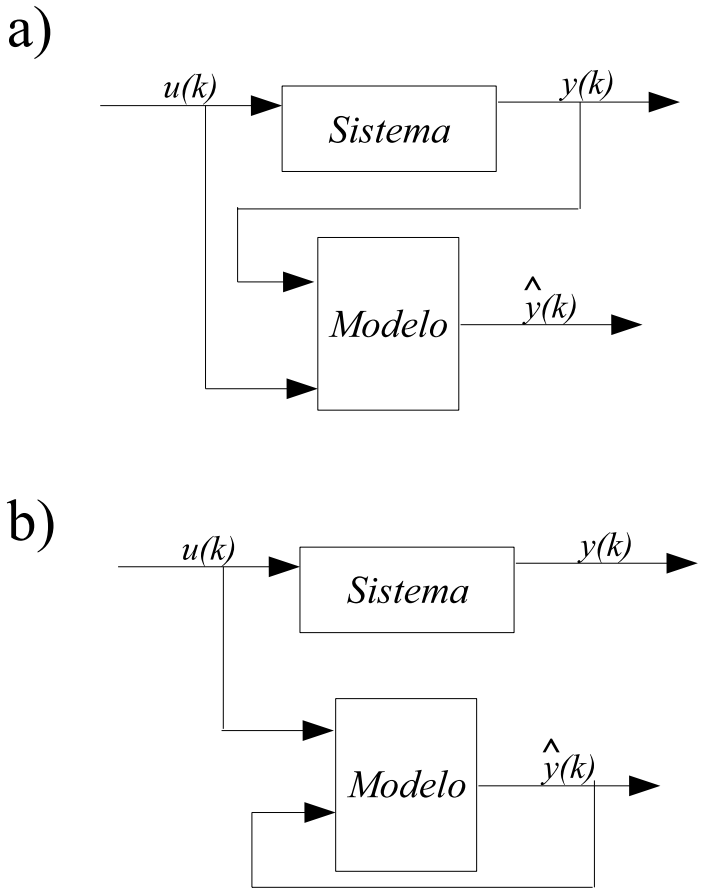


Figura 1.1: Tipos de esquemas: a)Serie-Paralelo. b)Paralelo [18]

modelo matemático del sistema, evaluar las posibles respuestas de éste ante diversos esquemas de control dado por medio de simulaciones que evaluar dichos esquemas en el propio reactor, donde incluso excitación mal elegida puede provocar un daño al sistema.

Existen varias estrategias para lograr obtener un modelo útil para simulaciones, en especial en el caso de sistemas no lineales. Una de las primeras es obtener un modelo analítico del sistema a partir del análisis de las relaciones constitutivas de elementos que le componen y las conexiones entre ellos. Este método está generalmente sujeto al conocimiento de las funciones que representan dichas relaciones, así como de la complejidad de éstas y a sus parámetros. Para sistemas en los cuales dichas funciones y sus parámetros son conocidos, o donde al menos se cuente con una aproximación, se deberá evaluar si su desempeño es satisfactorio, esto es, si cumple con los requisitos mínimos de aproximación y campo de acción establecidos.

En caso contrario se deberá reevaluar el tipo de funciones empleadas, así como sus parámetros. Si es posible emplear funciones más complejas para representar a los elementos y obtener un desempeño satisfactorio, podremos considerar como concluido el problema de modelado. Sin embargo, a mayor complejidad del modelo, más complicado resultará su análisis y el diseño de estrategias de control sobre éste, volviendo a entrar en el compromiso entre complejidad y desempeño.

Otra estrategia utilizada para la obtención de dichos modelos es la obtención de modelos lineales locales. La ventaja de este método es que permite modelar un sistema no lineal como una composición de sistemas lineales, lo cual simplifica la tarea de modelado e identificación, en especial si el análisis del sistema se reduce a la zona que modela dicho modelo lineal. Para sistemas no lineales en los cuales requiramos una representación más amplia de su espacio de operación, generalmente será necesario incrementar el número de modelos lineales locales. Esto último puede llegar a convertirse en una desventaja de este tipo de representaciones, ya que si los requisitos de aproximación así lo demandan el modelo puede llegar a ser demasiado complejo, dificultando su análisis.

Es entonces cuando se debe considerar el uso de modelos no lineales para la representación de la dinámica de sistemas, en vista de que una función no lineal puede, dependiendo del tipo de sistema y de la función no lineal considerada, proveer una representación más simple y general que una función lineal.

El problema entonces radica en elegir o identificar cuál es esa función no lineal tal que

logre modelar al sistema con una mejor aproximación. En la literatura se pueden encontrar una gran cantidad de modelos que permiten describir sistemas no lineales que poseen características muy específicas y relativamente conocidas (*i.e* fricción seca, fenómenos de histéresis), pero son pocos los que proveen de una representación más generalizada. Esto se debe al hecho de que analíticamente cada sistema no lineal es estructuralmente distinto a otro.

Cuando un sistema puede ser modelado de forma analítica y los valores de los parámetros pueden ser determinados, el problema de identificación se reduce a estimar el valor de dichos parámetros, en vista de que la topología estará prácticamente dada. Generalmente el valor de los parámetros se determina con base en el análisis de las señales entrada-salida del sistema real y aplicando un algoritmo de identificación.

El problema radica en qué tipo de modelo emplear si su desempeño resulta ser pobre; o cuando es incluso difícil establecerlo en esta forma, dado el poco conocimiento que se tenga del sistema real.

La motivación central del presente trabajo radica en este último punto: cómo obtener un modelo en paralelo de un sistema pese a desconocer su modelo analítico cuando se dispone de información proveniente de señales entrada-salida de la planta.

1.2. Antecedentes

Algunas aproximaciones clásicas a este problema han sido las funciones basadas en polinomios, donde la señal de salida del sistema es una función polinomial dependiente de retrasos de la señal de entrada y de salida. Entre las estrategias que han sido utilizadas se encuentran los modelos polinomiales de Kolmogorov-Gabor[18], que tienen la ventaja de ofrecer un modelo basado en formas lineales y cuadráticas, pero con la desventaja de ser sensibles al ruido y ser prácticos únicamente para predicciones de un paso adelante (es decir, como predictores), y no como modelos de simulación, dado que tienden a ser inestables.

Otra estrategia han sido los modelos en series de Volterra, los cuales únicamente consideran modelos sin realimentación de la salida, siendo únicamente función de retrasos en la entrada; lo cual les otorga ventajas de estabilidad, pero que limita su aplicabilidad.

Para combinar los dos modelos anteriores, surge otro tipo de modelo conocidos como modelos paramétricos en series de Volterra[18], el cual considera un modelo polinomial cuadrático para los retrasos de la entrada, pero con una realimentación lineal de retrasos

en la señal de salida. Este modelo permite evaluar la estabilidad del sistema al referirse únicamente a la parte lineal de realimentación y permite modelar algunos tipos de sistemas no lineales, con la desventaja de la pérdida de generalidad para la representación de sistemas, en donde generalmente la señal de salida es una función no lineal de los retrasos de ésta.

Entre los modelos más ampliamente conocidos se encuentran los modelos de Hammerstein y de Wiener[18], los cuales consideran combinaciones entre funciones lineales y no lineales polinomiales para lograr un modelo. Estos modelos han mostrado ser útiles, especialmente para sistemas con no-linealidades estáticas (como lo son los fenómenos de saturación, histéresis, zona muerta, etcétera).

En últimas fechas los modelos basados en redes neuronales y sistemas difusos han sido empleados para atacar el problema de identificación, dado que han mostrado una gran capacidad para aproximar funciones no lineales desconocidas, además de que ofrecen una estructura general tan compleja o sencilla como el problema lo demande. Por un lado, las redes neuronales son estructuras matemáticas inspiradas en las estructuras biológicas compuestas por neuronas, en donde a través de elementos relativamente simples (llamados *neuronas*) que efectúan operaciones y procesos muy sencillos logran, mediante interconexiones, realizar procesos complejos y procesamiento de información en paralelo. En general se consideran redes neuronales conformadas por "capas" de neuronas, donde una capa procesa la información recibida en paralelo y la envía a la siguiente capa de neuronas. Por otro lado, los sistemas difusos son también estructuras matemáticas, aunque éstos se encuentran inspirados en los procesos de pensamiento y toma de decisiones llevados a cabo por los humanos, así como en la teoría de conjuntos difusos propuesta por Lofti Zadeh [27]. En general un sistema de inferencia difuso se compone de reglas difusas de la siguiente forma

$$R_i: \text{ Si } x \text{ es } A_{i1} \quad \text{ y ... y } \quad x_n \text{ es } A_{in} \text{ entonces } y^i = a_{i0} + \sum_{j=1}^{n} a_{in} x_j$$

en el caso de una arquitectura Takagi-Sugeno[23], donde x_i son las señales de entrada, y^i es la salida del sistema difuso de acuerdo con la i-ésima regla, a_{ij} son constantes y A_i son conjuntos difusos; mientras que una arquitectura tipo Mamdani las reglas toman la forma:

$$R_i: \text{ Si } x \text{ es } A_{i1} \quad \text{ y ... y } \quad x_n \text{ es } A_{in} \text{ entonces } y^i \text{ es } B_i$$

donde B_i es un conjunto difuso. En ambos casos, la salida y del sistema difuso se define

como una ponderación entre los resultados de las reglas. En el Capítulo 2 se describe con más detalle la teoría de redes neuronales y sistemas difusos.

Una de las primeras propuestas presentadas para atacar este problema mediante redes neuronales fue propuesta por Hopfield [8], que considera redes neuronales en donde la salida de cada una de las neuronas que componen dicha red es función de un retraso de esta señal. En otras palabras, a cada neurona se le asocia una "memoria" de un tiempo de retraso. Esta estructura funciona relativamente bien con estructuras simples, pero a medida que el problema tiende a requerir un mayor número de neuronas, la estructura y el entrenamiento de la red tienden a ser más complejas, dado que el número de parámetros a determinar también crece.

Un proceso similar fue propuesto por Sastry [21], en donde a cada neurona se le asocia una neurona con memoria. Esta solución ha probado ser efectiva, aunque presenta problemas similares a los de las redes Hopfield. Entre ellos se encuentra el hecho de que el número de parámetros crece rápidamente a medida que los requisitos de aproximación se vuelven más estrictos.

Otro tipo de aproximaciones son los propuestos por Lee y Teng [14], con una capa de neuronas con memoria; la propuesta de Kosmatopoulos *et al* [13], en donde se consideran conexiones recurrentes entre neuronas de mayor orden, no solamente lineales; y la propuesta por Billings [1], en donde los retrasos de las neuronas de capas internas son realimentados tanto a la capa de neuronas que recibe la señal de excitación externa como a la última capa de neuronas.

Sin embargo todas las estructuras mencionadas anteriormente tienen un problema fundamental: dada la estructura propia de las redes neuronales suele ser difícil extraer información de la red para obtener algún tipo de información práctica, así como analizar al sistema identificado a partir de la red que le emula.

Los sistemas basados en redes neurodifusas logran en cierta forma atacar este problema, además de que han mostrado mejores desempeños que las anteriores. Este tipo de redes se basan en el concepto de sistemas difusos recurrentes propuesto por Gorrini [7], con la diferencia de que su implementación añade elementos de redes neuronales. Por ejemplo, entre los modelos de predicción basados en sistemas difusos se encuentra el propuesto por Chiu [2], así como *ANFIS*, propuesto por Jang [9], que han mostrado poseer una gran capacidad de aproximación para funciones no lineales, conservando cierta información de

la planta. Asimismo se han propuesto estructuras para simulación y control basadas en sistemas difusos, tales como los presentados en el libro de Wang[24], donde modelan una parte del sistema a partir de una descripción lineal, y otra es descrita mediante un sistema difuso.

El uso de redes neuronales inspiradas en sistemas difusos fue propuesto en primer lugar por Zhang [28], en donde una red neurodifusa es utilizada como un predictor para controlar un proceso no lineal mediante una representación entrada-salida del sistema.

Una de las aproximaciones más interesantes es la propuesta por Juang [10], en donde a partir de una red neurodifusa parcialmente recurrente, mostrada en la Figura 1.2, y un algoritmo de entrenamiento basado tanto en el algoritmo de gradiente como en métodos genéticos, logra desempeños de identificación muy satisfactorios al considerar una suerte de estados internos que modelan la dinámica del sistema (parte recurrente).

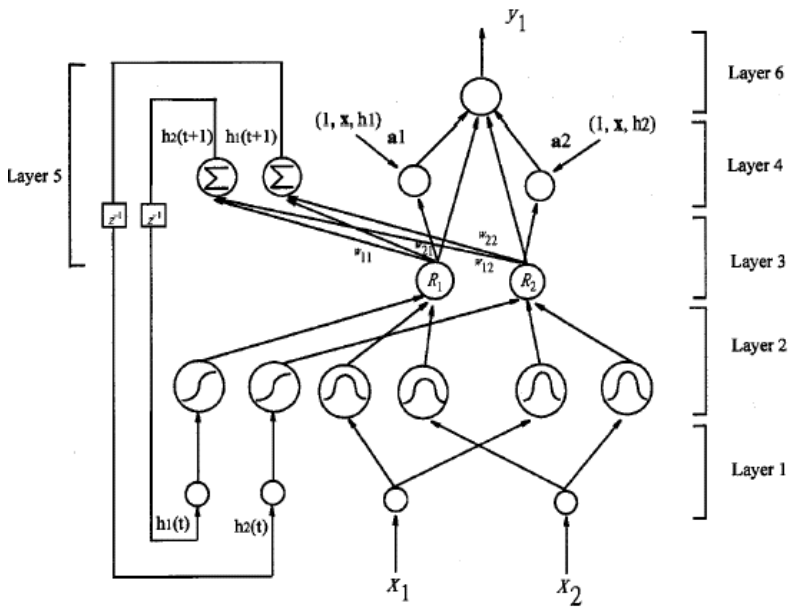


Figura 1.2: Red Neurodifusa propuesta por Juang [10]

En general las redes neurodifusas que han mostrado ser más eficientes, de acuerdo con [10], han sido aquellas basadas en una arquitectura tipo Takagi-Sugeno.

Las estructuras arriba indicadas ofrecen representaciones que han mostrado su efectivi-

dad en modelar cierto tipo de sistemas no lineales, aunque la definición de este tipo de sistemas ha estado de cierta forma alejado de la representación en espacio de estados, tan utilizada en control.

Si bien se han mencionado las principales estructuras utilizadas para atacar el problema de identificación mediante estructuras recurrentes, no hemos mencionado cómo se buscan los parámetros de este tipo de redes. Generalmente los algoritmos diseñados para encontrar los parámetros asociados a sistemas estáticos son insuficientes para encontrar aquellos de sistemas recurrentes. Entre los algoritmos más usados para entrenar este tipo de sistemas se encuentran el algoritmo de retropropagación a través del tiempo o BTT[**][20], que considera un algoritmo de retropropagación inspirado en aquél de redes estáticas, pero tomando en cuenta los efectos de la recurrencia. Este algoritmo tiene la desventaja de volverse lento y muy pesado (en términos computacionales) para casos en los que se tengan muchos datos de entrenamiento. Para emplear los conceptos del método anterior, pero para facilitar su empleo, Williams y Zipser [26] propusieron el *método de aprendizaje en tiempo real* o *RTRL*[***], basado en el mismo concepto pero considerando aproximaciones tales que permiten su fácil implementación y uso en problemas de tiempo real.

1.3. Resumen del trabajo

El presente trabajo presenta una nueva estructura de redes neurodifusas recurrentes basada en una representación en espacio de estados, la cual tiene la ventaja de proporcionar un más fácil manejo e interpretación que las descritas en la literatura, y que permite obtener un modelo útil para simulaciones obtenido a partir de datos entrada-salida de un sistema real.

La estructura propuesta asigna la tarea de manejar la dinámica del sistema a una red recurrente a través de las variables de estado, con tantas reglas como estados internos considerados; mientras que la parte estática del sistema es modelada mediante una red neurodifusa sin memoria. Se presenta también un análisis de estabilidad basado en la teoría de Lyapunov, aplicada a sistemas difusos recurrentes, y se presentan las condiciones bajo las cuales el sistema obtenido es asintóticamente estable.

Para determinar el valor de los parámetros de la red se propone un algoritmo de entre-

[**]Por sus siglas en inglés: *Back-propagation- Through-Time*
[***]Por sus siglas en inglés: *Real-Time Recurrent Learning*

namiento basado en el método del gradiente donde además la inicialización de los parámetros se encuentra basada en el método ANFIS [9] para después ser sujetas a un algoritmo de entrenamiento en tiempo real.

Con el objetivo de determinar la efectividad del método para identificar y generar un modelo cuyo comportamiento sea lo suficientemente cercano al del sistema real, se muestran dos ejemplos. El primero considera la identificación, a nivel de simulación, de un sistema no lineal muy empleado en evaluaciones con respecto a la efectividad de otros métodos[****]. El segundo considera un sistema escalado de laboratorio, en donde las señales entrada-salida están sujetas a ruido y perturbaciones reales, a diferencia del primero, donde no se consideran este tipo de fenómenos. Ambos resultados son satisfactorios, aunque sujetos de ser mejorados.

Por otro lado, a las redes que modelan a los sistemas se les aplica el análisis de estabilidad mencionado anteriormente, encontrándose que los modelos encontrados son asintóticamente estables.

El trabajo presentado se organiza en los siguientes capítulos: El Capítulo 2 comienza presentando en forma resumida el problema de identificación. Más adelante se presenta el Teorema Universal de Aproximación para redes neurodifusas, el cual, junto con la teoría para sistemas recurrentes (tanto de estructura como de entrenamiento, desarrollada en este mismo capítulo), permiten justificar el uso de una red neurodifusa recurrente para atacar el problema de modelado e identificación; por lo cual en el Capítulo 3 se propone una estructura para una red neurodifusa, así como el algoritmo de entrenamiento. En el Capítulo 4 se emplea el identificador propuesto para modelar dos sistemas no lineales: uno cuyos datos son obtenidos mediante una simulación, y el segundo mediante datos obtenidos de forma experimental a partir de un sistema real; y se muestran los resultados obtenidos. Finalmente, en el Capítulo 5 se detallan las conclusiones obtenidas a partir de este trabajo, así como el trabajo futuro a realizar.

[****]Es decir, es un problema tipo *benchmark*

Capítulo 2

Conceptos Básicos y Planteamiento del Problema

2.1. Introducción

Como se mencionó anteriormente, en muchas disciplinas y campos del conocimiento se encuentran problemas que involucran sistemas dinámicos. Específicamente en problemas consistentes en diseñar e implementar sistemas de control los sistemas a operar suelen ser de naturaleza dinámica (motores, torres de destilación, circuitos eléctricos, etcétera), en donde es de gran utilidad el poseer un modelo del sistema que nos permita analizar sus propiedades e incluso simular y predecir su comportamiento antes de implementar un esquema de control sobre él. Entre "mejor" sea el modelo empleado, más seguridad tendremos de que el diseño que hemos desarrollado tendrá un comportamiento similar al momento de ser implementado en el sistema físico real en comparación con el modelo simulado mediante una computadora y un paquete de software.

El concepto de cuál es "*el mejor modelo*"depende de lo que deseemos obtener del sistema y de cuáles serán sus puntos o regiones de operación. Por ejemplo, podemos poseer el modelo matemático de un resorte que describa su comportamiento desde su etapa de máxima compresión hasta la etapa de ruptura. Dicho modelo (de naturaleza no lineal) será excelente para simular cualquier tipo de condición de operación a la que sea sometido el resorte, con el costo de que su descripción analítica será más complicada que un modelo lineal enfocado a describir el comportamiento de este sistema en un intervalo corto, y existirá un mayor número de parámetros involucrados.

Es en este último punto donde la correcta elección de un modelo es crucial, ya que entre más exacto deseemos que sea un modelo generalmente el número de parámetros involucrados dentro de éste será mayor. El problema de identificación gira en torno a este último punto, ya que el modelo o estructura debe tener los suficientes grados de libertad como para obtener una buena representación del sistema, pero sin ser demasiado complejo tal que complique su análisis o identificación.

Entonces podemos entender el problema de identificación como que dada una cierta estructura del modelo (un polinomio, una serie de Fourier, una red neuronal), encontrar los parámetros que logren que la diferencia entre las señales de salida entre el modelo y el sistema real dada una misma excitación sea la menor posible, teniendo como información datos entrada-salida del sistema real.

En el caso de sistemas lineales el problema de identificación es ya una materia madura, y existen varios métodos mediante los cuales se pueden encontrar modelos lineales que logran desempeños satisfactorios, claro que dentro de las limitaciones inherentes a este tipo de sistemas.

Sin embargo los sistemas lineales no siempre son la mejor opción, especialmente si requerimos de un modelo que opere dentro de regiones para las cuales el desempeño de un modelo lineal es poco preciso, o cuando el objeto de estudio es precisamente la acción de las no linealidades, por lo que tenemos que considerar entonces el utilizar estructuras no lineales.

Al considerar estructuras no lineales nos topamos con una de sus características inherentes: no existe una única estructura para representar a la totalidad de los sistemas no lineales. En el caso de los sistemas lineales de una única salida se cuenta con una estructura general expresada en forma matricial en la forma

$$\dot{\mathbf{x}} = A(t)\mathbf{x} + B(t)\mathbf{u} \tag{2.1}$$

$$y = C(t)\mathbf{x} + D(t)\mathbf{u} \tag{2.2}$$

donde $\mathbf{u} \in \Re^m$ la entrada del sistema, $\mathbf{x} \in \Re^n$ el *vector de estados*, relacionado con los elementos dinámicos del sistema, y $y \in \Re$ la salida del sistema, $A(t) \in \Re^{n\times n}$, $B(t) \in \Re^{n\times m}$, $C(t) \in \Re^{1\times n}$ y $D(t) \in \Re^{1\times m}$ matrices variantes en el tiempo. Para sistemas no lineales tenemos que su representación es mucho más general, del tipo $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t)$, $y = h(\mathbf{x}, \mathbf{u}, t)$, con $\mathbf{f}$ y h campos vectoriales y t el tiempo.

Un modo de proponer la estructura de las funciones $\mathbf{f}$ y h es realizar un análisis del sistema atendiendo a las relaciones e interconexiones que posean los elementos de éste, y establecer los valores de los parámetros de dichas funciones mediante el conocimiento de los parámetros de los elementos que los componen. Otro modo de determinar el valor de los parámetros es mediante un análisis de los datos entrada-salida del sistema a modelar, y luego mediante algoritmos de identificación de parámetros, determinar el valor de éstos para una estructura dada de las funciones $\mathbf{f}(\cdot,\cdot)$ y $h(\cdot,\cdot)$, lo cual no siempre es fácil, en especial cuando los parámetros no son lineales con respecto al modelo elegido; esto es, que los parámetros θ no puedan ser representados en la forma $\mathbf{f}(\mathbf{x},\mathbf{u}) = \xi(\mathbf{x},\mathbf{u})\theta$, donde $\xi(\mathbf{x},\mathbf{u})$ es llamado *regresor*.

Un problema más radica en que la estructura que obtengamos para un cierto sistema de naturaleza no lineal no tiene por qué ser útil para otro sistema no lineal. En el caso de los sistemas lineales, si tenemos disponible un algoritmo de identificación de parámetros para un sistema de orden n y grado relativo r, con mínimos cambios puede servir para otro sistema lineal de distinta naturaleza de orden m y grado relativo ρ. Esto no necesariamente sucede con los sistemas no lineales, donde prácticamente cada sistema es estructuralmente distinto a otro.

Es entonces cuando surge la necesidad de plantear estructuras y modelos que puedan ser utilizados por un número mayor de sistemas no lineales, pudiendo además aprovechar el trabajo de desarrollar algoritmos más generales. Entre las estructuras que pueden ser empleadas se encuentran las redes neuronales y los sistemas difusos, cuya combinación es conocida como *red neurodifusa*.

2.2. Redes Neuronales y Sistemas Difusos

Las redes neuronales y los sistemas difusos son dos estrategias que han sido utilizadas con bastante éxito, como se explicó en el Capítulo 1, para capturar y emular algunas de las características que hacen de las estructuras biológicas pensantes sistemas muy eficientes de procesamiento y manejo de información abstracta o incierta.

En particular, las redes neuronales se inspiran en el hecho biológico de que la interconexión de elementos muy sencillos (neuronas) con otras similares logra que con pocas unidades de procesamiento elemental se generen estructuras más complejas que pueden procesar información en paralelo con relativamente pocas operaciones.

Por otro lado los sistemas difusos son estructuras que tratan de emular la forma de razonamiento humano, y concretamente aluden al hecho de que para tomar una decisión no solo se toma en cuenta una única "opinión", sino que se buscan varias y con ellas se busca llegar a una decisión final, pesando cada "opinión"de acuerdo con la importancia que se le de a cada una.

2.2.1. Redes Neuronales

El elemento fundamental dentro de una red neuronal es la *neurona*, la cual se puede apreciar gráficamente en la Figura 2.1. Matemáticamente una neurona se puede entender como un elemento que depende de dos parámetros: la entrada $u \in \Re^m$ y el "umbral"(*threshold*) $b \in \Re^m$. La función f se conoce como *función de activación*, donde esta suele tener las formas mostradas en la Figura 2.2, en donde se puede apreciar el efecto del valor umbral. En otros casos, la neurona puede efectuar únicamente la función de suma, donde $y = \sum_i u_i$.

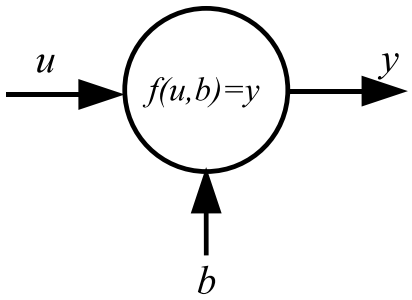


Figura 2.1: Representación gráfica de una neurona

Gráficamente una red neuronal se puede apreciar en la Figura 2.3, en donde cada flecha representa un valor de "peso" ω_{ij} que es asignado a cada señal u_i, para su posterior procesamiento por las neuronas de las siguientes "capas" j.

2.2.2. Sistemas Difusos

Por otro lado, los sistemas difusos se basan en la lógica difusa, creada por Zadeh en 1965 como una extensión de la lógica de Boole. En este tipo de sistemas se considera que el "verdadero" y "falso" en una frase no son absolutamente ciertos (1) o falsos (0), sino que considera valores intermedios en los cuales los enunciados pueden ser en parte verdaderos y

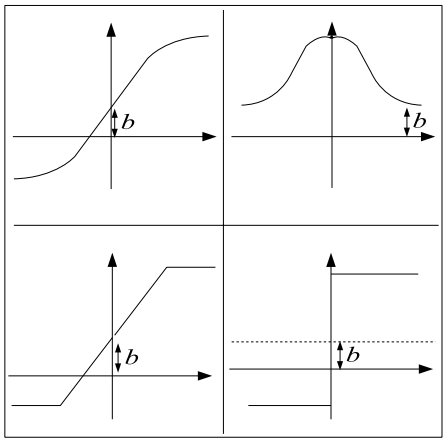


Figura 2.2: Funciones de activación

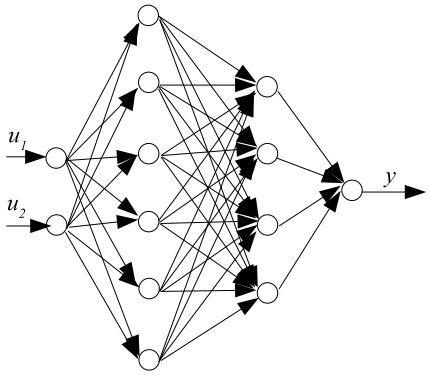


Figura 2.3: Representación gráfica de una red neuronal

# de regla	Si	entonces
1	hace frío	usar ropa gruesa
2	está templado	usar ropa normal
3	hace calor	usar ropa ligera

Tabla 2.1: Ejemplos de reglas utilizadas en lógica difusa

en parte falsos. En general se basa en reglas del tipo *Si... entonces*, en donde dependiendo de qué tan verdadera sea la parte del *Si* (también conocida como *antecedente*), será el peso que se le de en la decisión final a la parte *entonces* (o parte *consecuente*). Por ejemplo, podemos tomar decisiones acerca de la ropa que usaremos en función de la temperatura ambiente por medio de las reglas indicadas en la Tabla 2.1.

Estas reglas nos dan una noción acerca del funcionamiento de los sistemas basados en lógica difusa, en donde la decisión final de la ropa a usar depende de qué tan cierto sea que *haga frío*, el clima *esté templado* o que *haga calor*. Nosotros no poseemos sensores que nos digan con exactitud en unidades del Sistema Internacional la temperatura ambiente, pero sí contamos con una *percepción* del frío y del calor, conforme a los cuales tomaremos la decisión final. Puede que sintamos que el clima está entre templado y caluroso, en ese caso tenemos la libertad de combinar ropa normal con ropa ligera. Si sentimos que hace entre frío y calor, podemos escoger el llevar ropa normal con una prenda gruesa, en vez de usar únicamente ropa térmica.

Para llevar este tipo de conocimiento y reglas a una estructura matemática y/o de control debemos indicar qué rango de temperaturas consideramos como “frías”, “templadas” y “calurosas”, y qué tan cierto consideraremos que una temperatura de 10° es templada, o que 20° es fría. Un ejemplo se puede apreciar en la Figura 2.4, donde a cada temperatura situada en el eje de las abscisas corresponde un *grado de verdad* o *grado de membresía* correspondiente a cada uno de los *conjuntos* definidos como el conjunto de las temperaturas que consideramos “frías”, “templadas”o “calientes”. La ponderación de estas condiciones en cada regla indicará cuál es el peso que se considerará en la elección de la ropa (y en último término, qué tan “gruesa” o “ligera” deberá ser). A estos conjuntos se les conoce como *conjuntos difusos*, dado que se permiten valores de verdad o *pertenencia* reales situados entre 0 y 1.

Ahora bien, el *grado de pertenencia* de un cierto elemento a un conjunto difuso suele estar dado por una función matemática muy similar a las funciones de activación de una

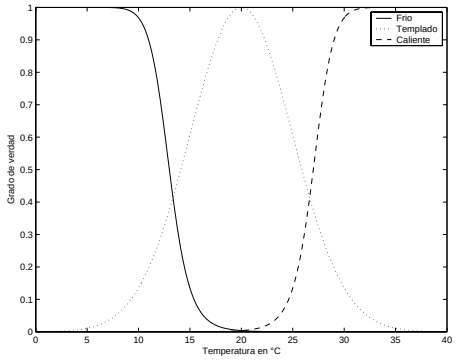


Figura 2.4: Funciones de membresía para temperatura

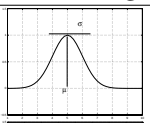
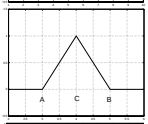
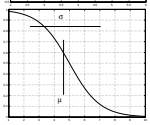
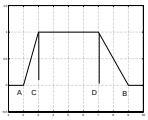
Nombre	Función	Representación gráfica
Gaussiana	$f(x) = e^{-\sigma^2(x-\mu)^2}$	
Triangular	$f(x) = \begin{cases} \frac{x-A}{C-A} & \text{si} \quad x \in [A, C) \\ \frac{B-x}{B-C} & \text{si} \quad x \in [B, C] \\ 0 & \text{si} \quad x \notin [A, B] \end{cases}$	
Sigmoide	$f(x) = \frac{1}{(1+e^{-\sigma(x-\mu)})}$	
Trapezoidal	$f(x) = \begin{cases} \frac{x-A}{C-A} & \text{si} \quad x \in [A, C) \\ \frac{B-x}{B-D} & \text{si} \quad x \in [B, D] \\ 1 & \text{si} \quad x \in [C, D] \\ 0 & \text{si} \quad x \notin [A, D] \end{cases}$	

Tabla 2.2: Funciones de membresía comunes para definir conjuntos difusos

neurona, como las mostradas en la Figura 2.2. Las funciones de activación más comunes se muestran en la Tabla 2.2.

Con base en estas funciones se puede definir el *peso* de cada regla en una forma matemática. Por ejemplo, para las reglas mostradas en la Tabla 2.1 podemos definir el peso de cada regla como el grado de pertenencia que tenga la temperatura con respecto al conjunto difuso indicado por la parte antecedente en dicha regla.

En el caso de múltiples entradas y una salida la definición del peso de cada regla depende del grado de pertencia que tenga cada entrada con respecto al conjunto difuso definido en la parte antecedente. Por ejemplo, podemos considerar el ejemplo de la acción de conducir un automóvil que se topa ante una luz roja. Un ejemplo de definición de reglas lo podemos ver en la Tabla 2.3, donde se consideran tres conjuntos difusos para la velocidad del automóvil (Alta, Media y Baja), dos para la distancia (Cercana, Lejana) y cuatro para el grado de

# de regla	Si la VELOCIDAD es	y la DISTANCIA al paso cebra u otro auto es	entonces FRENAR
1	alta	lejana	MEDIO
2	alta	cercana	FONDO
3	media	lejana	POCO
4	media	cercana	MEDIO
5	baja	lejana	NULO
6	baja	cercana	MEDIO

Tabla 2.3: Ejemplo: Conducción de un automóvil

frenado (NULO, POCO, MEDIO, FONDO).

Llamemos a los conjuntos difusos de la velocidad v Alta, Media y Baja como A_1, A_2 y A_3 respectivamente, mientras que para la distancia d definamos como B_1 al conjunto difuso que representa a la distancia lejana y B_2 a la cercana. Finalmente, digamos que la presión que ejercemos sobre el freno será definida por conjuntos difusos. Entonces el peso de cada regla dependerá del grado de pertenencia que tengan ambas entradas con respecto a sus conjuntos difusos definidos en la parte antecedente de la regla. Es decir, el peso de la regla 1 $R_1(v, d)$ dependerá de el grado de verdad que tenga el enunciado "*Si la Velocidad es Alta y la Distancia es lejana*". La conjunción y implica que el grado de verdad de la regla 1 estará en función de que se cumplan las condiciones 'Velocidad Alta' y 'Distancia cercana'. La conjunción y implica una operación lógica entre conjuntos difusos, y su definición se toma como un caso más general de la implicación lógica ordinaria de la lógica bivaluada. La operación de conjunción en lógica difusa se conoce como *t-norma* y se toma de distintas formas, siendo las más comunes (si se toma a v y d como las variables de entrada y los valores de membresía a los conjuntos difusos $A_i(v)$ y $B_1(d)$) las mostradas en la Tabla 2.4. Así como existe la operación de conjunción, existe también la operación de disyunción o. Para definir el grado de verdad que tiene la parte antecedente de una regla difusa definida en la forma *Si* v *es* A_i *O* d *es* B_i se cuenta también con varias definiciones, las cuales se conocen como *t-conormas* y se muestran en la misma Tabla 2.4.

Una vez establecidos los conjuntos difusos y las reglas, conocidas también como *base de conocimiento*, será necesario tomar en cuenta la parte consecuente de cada regla y con base en ellas tomar una decisión final. Dependiendo de la estructura de la parte consecuente de las reglas será el método que se usará para tomar la decisión final, con lo que además se definirá el *tipo* del sistema difuso. Si la parte consecuente implica un conjunto difuso, el sistema difuso se define como *tipo Mamdani* o lingüístico. Si la parte consecuente implica

T-normas	
Mínimo:	$A_i(v)$ y $B_i(d) = \text{mín}(A_i(v), B_i(d))$
Producto:	$A_i(v)$ y $B_i(d) = A_i(v)B_i(d)$
Diferencia acotada:	$A_i(v)$ y $B_i(d) = \text{máx}(0, A_i(v) + B_i(d) - 1)$
T-conormas	
Máximo	$A_i(v)$ *ó* $B_i(d) = \text{máx}(A_i(v), B_i(d))$
Suma algebraica	$A_i(v)$ *ó* $B_i(d) = A_i(v) + B_i(d) - A_i(v)B_i(d)$
Suma acotada	$A_i(v)$ *ó* $B_i(d) = \text{mín}(1, A_i(v) + B_i(d))$

Tabla 2.4: Tipos de T-normas u operaciones *... y ...*, y T-conormas u operaciones *... o...* con conjuntos difusos

una función matemática que sea función de las variables de entrada, se definirá como un sistema difuso *tipo Takagi-Sugeno*. Esto es, un sistema difuso tipo Mamdani para el sistema de frenado expuesto anteriormente, considera que la definición de los términos lingüísticos NULO, POCO, MEDIO y FONDO se expresará en términos de conjuntos difusos, donde la decisión final de qué presión aplicar al freno radicará tanto del grado de verdad de cada regla así como de los conjuntos difusos considerados en la parte consecuente. Existen varios métodos para obtener una salida numérica concreta a partir de sistemas difusos tipo Mamdani, los cuales no serán considerados en el presente trabajo, pero que se pueden consultar en [12] y [18] entre otros.

Consideremos ahora un sistema difuso tipo Takagi-Sugeno para atacar el problema de frenado señalado anteriormente. En este caso la parte consecuente estará asociada a una función matemática. Es decir, el frenado a aplicar $f_f(v, u)$ será función, en cada regla, de la velocidad v y de la distancia d. En general, un sistema tipo Takagi-Sugeno considera reglas en la forma:

$$R^i: \text{ Si } x_1 \text{ es } A_{1i} \text{ y... y } x_n \text{ es } A_{ni} \text{ entonces } f_i = a_{0i} + a_{1i}x_1 + \ldots + a_{ni}x_n \tag{2.3}$$

donde $a_{ij} \in \Re \; \forall \; i, j$.

Existen varios métodos para obtener una decisión final, siendo la más común el promedio ponderado, el cual pesa la parte consecuente de cada regla de acuerdo con el grado de verdad de ésta. Si llamamos $R_i(\mathbf{x})$ al peso de la i-ésima regla, la salida del sistema difuso será

$$f_f = \frac{\sum_i R_i(\mathbf{x}) f_i(\mathbf{x})}{\sum_i R_i(\mathbf{x})} = \sum_i w_i(\mathbf{x}) f_i(\mathbf{x}) \tag{2.4}$$

donde

$$w_i(\mathbf{x}) = \frac{R_i(\mathbf{x})}{\sum_i R_i(\mathbf{x})} \tag{2.5}$$

2.2.3. Redes neurodifusas

Las *redes neurodifusas* son redes neuronales en donde las funciones de activación pueden ser interpretadas como conjuntos difusos, en las cuales las acciones de ponderación y evaluación de el grado de pertenencia son llevadas por medio de redes neuronales, lo que permite emplear más fácilmente las técnicas usuales de entrenamiento existentes para éstas (tal como la retropropagación), como se indica en [24]. En la Figura 2.5 se muestra la similitud entre un sistema difuso y una red neuronal.

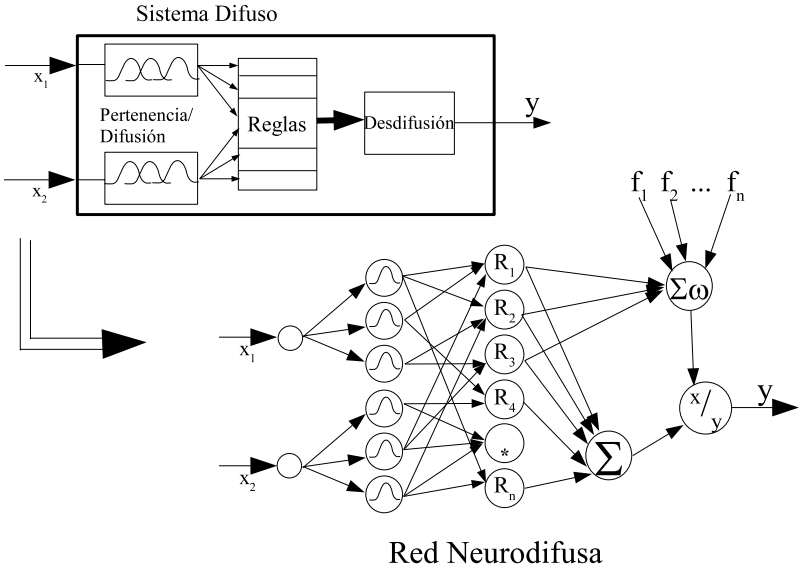


Figura 2.5: Conversión de un Sistema Difuso a una red Neurodifusa

2.2.4. Redes Neuronales y Sistemas Difusos como Aproximadores Universales

De acuerdo con el *Teorema Universal de Aproximación* [24], supongamos un sistema difuso o red neuronal de n entradas tal que esté representado por la función

$$f(\mathbf{u}) = \frac{\sum_{l=1}^{M} \bar{y}^l \left[\prod_{i=1}^{n} a_i^l e^{-\left(\frac{u_i - \bar{u}_i^l}{\sigma_i^l}\right)^2} \right] / \delta^l}{\sum_{l=1}^{M} \left[\prod_{i=1}^{n} a_i^l e^{-\left(\frac{u_i - \bar{u}_i^l}{\sigma_i^l}\right)^2} \right] / \delta^l} \tag{2.6}$$

donde $u \in \Re^n$, M el número de reglas, $\bar{u}_i^l$ y σ_i^l el centro y ancho de las funciones gaussianas que representan a los conjuntos difusos de la entrada i y la regla l respectivamente, a_i^l es el peso que se le da a cada conjunto difuso y δ^l es el peso que tendrá predeterminada cada regla l. En este caso se postula el siguiente Teorema [24]:

Teorema 2.1 (Teorema Universal de Aproximación) *Para cualquier función real y continua g definida en un conjunto compacto $U \subset \Re^n$ y un número $\varepsilon \in \Re$, $\varepsilon > 0$ existe un sistema difuso en la forma (2.6) tal que*

$$\sup_{\bar{w} \subset U} |f(\bar{w}) - g(\bar{w})| < \varepsilon \tag{2.7}$$

El siguiente corolario indica otro tipo de norma que puede utilizarse para definir el error entre la función aproximada dada por el sistema difuso y por la función real.

Corolario 2.1 ([24]) *Para cualquier función $g \in \mathcal{L}_2(U)^*$ y $\varepsilon > 0$ existe un sistema difuso f en la forma (2.6) tal que*

$$\left(\int_U |f(\bar{w}) - g(\bar{w})|^2 d\bar{w} \right)^{\frac{1}{2}} < \varepsilon \tag{2.8}$$

donde $U \subset \Re^n$ es compacto, $\mathcal{L}_\in(U) = [g : U \to R| \int_U |g(\bar{w}^2)| d\bar{w}| < \infty]$ con las integrales en el sentido de Lebesgue.

Este teorema y su corolario indican entonces que siempre podemos aproximar de forma arbitraria una función tal que cumpla con las características indicadas, aunque se debe notar que no hace referencia al grado de complejidad que el sistema difuso ha de tener ni el cómo obtener dichas funciones; por esto es que debemos entender a este teorema y su corolario como una justificación del uso de sistemas difusos para atacar el problema de identificación de funciones no lineales.

2.3. Algoritmos de identificación

Con una topología definida mediante la cual se identificará una determinada función $f(\cdot)$, es necesario determinar un método mediante el cual sintonizar los parámetros de la red neurodifusa $\hat{f}(\cdot)$. La elección del método de entrenamiento dependerá del uso que se le desee

*Un espacio $\mathcal{L}_2$ se define aquel que contiene a todas las funciones u tales que cumplan con que su norma $||u||_{\mathcal{L}_2} = \sqrt{\int_0^\infty u(t)^T u(t) dt} < \infty$

dar a la red: ya sea como red recurrente o como red estática. Esto último es importante, ya que los métodos desarrollados para sistemas recurrentes son concebidos tomando en cuenta dicha propiedad, lo que provoca que su entrenamiento sea más complejo que en el caso de redes estáticas.

2.3.1. Algoritmos para sistemas estáticos

Entenderemos como sistemas estáticos aquellos que pueden ser representados mediante una función sin memoria

$$y = f(\mathbf{x}, \bar{\theta}) \tag{2.9}$$

donde $\mathbf{x} \in \Re^n$ son las entradas de la función y $\bar{\theta} \in \Re^{n_\theta}$ los parámetros.

El objetivo entonces será el minimizar una *función de costo* $J(\cdot)$ que dependa del error entre la función real y la estimada

$$E = f(\cdot) - \hat{f}(\cdot)$$

con base en el cálculo de los parámetros $\hat{\bar{\theta}}$ tal que optimicen dicha función (o en dado caso, que sean sub-óptimos). De la definición de la función de costo dependerá el tipo de método de optimización a emplear, así como del conocimiento que se tenga para comenzar dicha búsqueda.

Métodos basados en el gradiente

Este tipo de métodos busca cambiar en cada paso de entrenamiento el valor de los parámetros θ con base en la información de la topología local de la función de costo en función de los parámetros, con el fin de dirigirse hacia el mínimo local con información del gradiente.

Los métodos basados en el gradiente son los más comunes dentro de las técnicas de optimización local para funciones no lineales [18], y esto se debe a su relativa facilidad de uso y velocidad de convergencia, pero poseen la desventaja de que los resultados solo pueden ser garantizados de forma local; esto es, que es generalmente difícil determinar si el valor de los parámetros $\hat{\bar{\theta}}$ provoca un mínimo local o global de la función de costo $J(\cdot)$, la cual

puede ser tomada como el error cuadrático instantáneo.

$$J(k) = \frac{1}{2}(\hat{f}(k) - f(k))^2 \tag{2.10}$$

En general entenderemos al gradiente como la derivada parcial

$$\mathbf{g} = \frac{\partial J(\hat{\bar{\theta}})}{\partial \hat{\bar{\theta}}} \tag{2.11}$$

y el objetivo de los métodos basados en gradiente será el determinar la forma de variar los parámetros $\hat{\bar{\theta}}$, en cada tiempo k, de forma proporcional a un paso η_θ en una dirección $\mathbf{p}$ dada por el gradiente $\mathbf{g}$ rotado y escalado por una matriz de dirección R, es decir, calcular para el tiempo $k+1$ el valor de $\hat{\bar{\theta}}$ mediante

$$\hat{\bar{\theta}}(k+1) = \hat{\bar{\theta}}(k) - \eta_\theta \mathbf{p}(k) \tag{2.12}$$

$$\mathbf{p}(k) = R(k)\mathbf{g} = R(k)\frac{\partial J(\hat{\bar{\theta}}(k))}{\partial \hat{\bar{\theta}}} \tag{2.13}$$

Con lo que se busca que en cada paso de entrenamiento se reduzca el valor de la función de costo. La suposición general en este tipo de métodos es que la función de costo será convexa con respecto a los parámetros. Cuando el sistema es no lineal esto generalmente no se puede garantizar, y generalmente pueden existir un gran número de *mínimos locales*. Si el algoritmo de búsqueda sitúa ésta cerca de uno de estos mínimos, suele ser difícil que encuentre un mínimo con mejor desempeño(en caso de existir), con lo que se dice que queda *atrapado en un mínimo local*.

Los diversos métodos basados en gradiente se diferencian básicamente por la elección de la matriz $R(k)$ y el paso de entrenamiento η_θ. Generalmente este tipo de métodos suele ser relativamente sensible al ruido, así como a la magnitud de la ganancia de adaptación η_θ. A continuación se mencionan algunos de estos métodos:

1. **Descenso en la mayor inclinación (*steepest descent*)**

   Esta es la forma más sencilla de establecer la ecuación (2.12), ya que se toma $R(k)$ como la matriz identidad $R(k) = I$ y η_θ se toma constante. Entre las ventajas de este caso particular se encuentra su relativamente fácil implementación, comprensión y desempeño en las primeras iteraciones. Sin embargo, algunas desventajas que presenta son una lenta convergencia al transcurrir varias iteraciones, lo que ocasiona que no se

encuentre el óptimo local (a menos que se considere un número infinito de pasos de entrenamiento). Generalmente no se busca el valor del óptimo, sino un valor *subóptimo* tal que satisfaga un determinado criterio de aproximación (*i.e.* una cota del error).

2. **Método de Newton**. En este método, la matriz de rotación se toma como la inversa del Hessiano

$$R(k) = H^{-1}(k) = \left(\frac{\partial^2 \mathbf{g}}{\partial \theta^2}\right)^{-1} \tag{2.14}$$

Este método cuenta con la ventaja de poseer una muy buena velocidad de convergencia, especialmente al situarnos cerca del óptimo. Sin embargo sus desventajas saltan a la vista: requiere del cálculo de la matriz Hessiana, la cual puede ser difícil de calcular, requiere que dicha matriz sea positiva definida para converger (no siempre esperado estando relativamente lejos del óptimo) y en general resulta computacionalmente más pesado dada la necesidad de calcular la inversa de una matriz. Para sortear estos obstáculos suelen usarse formas aproximadas del Hessiano tales como

$$\hat{H}(k) \quad = \quad \hat{H}(k-1) + Q(k-1) \tag{2.15}$$

o mediante métodos basados en la fórmula de Broyden-Fletcher-Goldfarb-Shanno [22], tomando como valor inicial del Hessiano $H(0) = P$,

$$\begin{aligned}
\hat{H}(k) &= (P - \Gamma(k-1))\, H^{-1}(k-1)\, (P - \Gamma(k-1))^T + \Theta(k-1) && (2.16)\\
\Gamma(k) &= \frac{\Delta\theta(k)\Delta\mathbf{g}^T(k)}{\Delta\theta^T(k)\Delta\mathbf{g}(k)} && (2.17)\\
\Theta(k) &= \frac{\Delta\theta(k)\Delta\theta^T(k)}{\Delta\theta^T(k)\Delta\mathbf{g}(k)} && (2.18)\\
\Delta\theta(k) &= \theta(k+1) - \theta(k) && (2.19)\\
\Delta\mathbf{g}(k) &= \mathbf{g}(k+1) - \mathbf{g}(k) && (2.20)
\end{aligned}$$

3. **Métodos conjugados**. Este tipo de métodos suelen emplear a su vez aproximaciones de los métodos anteriores. Si bien los resultados no son tan buenos como en éstos, ésto se paga con el peso computacional, ya que la complejidad de los cálculos disminuye, con lo que además se permite atacar problemas con una mayor cantidad de parámetros.

Los métodos conjugados se pueden describir como una actualización de parámetros

dada por

$$\theta(k) = \theta(k-1) - \eta_\theta \mathbf{p}(k-1) \tag{2.21}$$

donde

$$\mathbf{p}(k) = \mathbf{g}(k) - \beta(k)\mathbf{p}(k-1) \tag{2.22}$$

y el término β está dado por

$$\beta(k) = \frac{\mathbf{g}^T(k)\mathbf{g}(k)}{\mathbf{g}^T(k-1)\mathbf{g}(k-1)} \tag{2.23}$$

Este tipo de aproximación tiene la ventaja de no requerir del cálculo del Hessiano, lo que le permite ser aplicado a problemas de gran número de parámetros, además de una buena velocidad de convergencia y requerimientos computacionales bajos al ser comparado con los métodos de Newton (puesto que no es necesario el cálculo de matrices inversas).

Métodos basados en mínimos cuadrados

Este tipo de métodos se distingue por no hacer mayores suposiciones sobre la topología de la función de costo, a no ser el suponer que es una función suave. La función de costo más comúnmente usada es la suma de los errores al cuadrado ponderados

$$J(\theta) = \sum_{i=1}^{N} q_i \left(f_i - \hat{f}_i \right)^2 \tag{2.24}$$

donde q_i es la ponderación de cada error. Típicamente $q_i = 1$. Esto se puede escribir también como

$$J = \Phi^T \Phi \tag{2.25}$$

donde $\Phi = [\Phi(1,\theta)\ \Phi(2,\theta)\ \ldots\ \Phi(N,\theta)]^T$ es la matriz que contiene los errores de los datos generados por el modelo dados por $\hat{f}$ con respecto a los datos obtenidos f_i. En general para aplicar este método se supone que el modelo $\hat{f}$ es lineal con respecto a sus parámetros, es decir

$$\hat{y} = \hat{f}(\mathbf{x}, \bar{\theta}) = \phi(\mathbf{x})\bar{\theta} \tag{2.26}$$

Con esta suposición, se puede determinar que el valor de los parámetros que minimizan la función de costo (2.24) es

$$\hat{\hat{\theta}} = (\phi^T \phi)^{-1} \phi^T f \tag{2.27}$$

Dado que el cálculo de la matriz $(\phi^T \phi)^{-1}$ puede llevar a singularidades, suele emplearse el método recursivo para mínimos cuadrados, el cual considera incluso su utilización en sistemas en tiempo real mediante la expresión de adaptación de parámetros

$$\hat{\hat{\theta}}(k) = \hat{\hat{\theta}}(k-1) + \bar{\gamma}(k) E(k) \tag{2.28}$$

donde

$$E(k) = y(k) - \hat{y} = y(k) - \phi^T(k) \hat{\hat{\theta}}(k-1) \tag{2.29}$$

$$\gamma(k) = \frac{P(k-1)\phi(k)}{\phi^T(k) P(k-1) \phi(k) + 1} \tag{2.30}$$

$$P(k) = (I - \gamma(k)\phi^T(k)) P(k-1) \tag{2.31}$$

Por lo general, se suele iniciar el valor de P con $P(0) = \alpha I$, donde $\alpha >> 1$.

2.3.2. Algoritmos para sistemas recurrentes

Un sistema recurrente se diferencia de un sistema estático en que en el primero, a diferencia del segundo, la salida depende de valores anteriores de tanto la salida como la entrada. Si llamamos $\mathbf{x}(k)$ a la entrada del sistema que es realimentada al sistema, $\mathbf{u}(k)$ a la entrada externa, con k como el tiempo, la salida del sistema recurrente definida por la función $\mathbf{f}(\cdot, \cdot)$ estará dada por

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k)) \tag{2.32}$$

Si la función recurrente es definida mediante un sistema difuso, en forma general, el sistema difuso recurrente se define por medio de m reglas en la forma

$$R_i: \quad \text{Si } \mathbf{x} \text{ es } A_i(\mathbf{x}) \quad \text{y} \quad \mathbf{u} \text{ es } B_i(\mathbf{u}) \quad \text{entonces} \quad \mathbf{x}^i(k+1) = C_i \mathbf{x}(k) + D_i \mathbf{u}(k) \tag{2.33}$$

donde $i = 1 \ldots m$, $m = \dim(\mathbf{u})$, $n = \dim(\mathbf{x})$, $C_i \in \Re^{n \times n}$, $D_i \in \Re^{n \times m}$; y con el peso de

cada regla, considerando la multiplicación como la t-norma de la conjunción, definido como

$$\begin{aligned} R_i(\mathbf{x}(k)) &= A_{1i}(x_1(k))A_{2i}(x_2(k))\dots A_{ni}(x_n(k))B_{1i}(x_1(k))\dots B_{mi}(x_m(k)) \\ &= \prod_{j=1}^{n} A_{ji}(\mathbf{x}(k)) \prod_{j=1}^{m} B_{ji}(\mathbf{u}(k)) \end{aligned} \tag{2.34}$$

finalmente el valor de $\mathbf{x}(k+1)$ estará dado por el promedio ponderado

$$\mathbf{x}(k+1) = \frac{\sum_{j}^{m} R_j(\mathbf{x}(k))(C_j\mathbf{x}(k) + D_j\mathbf{u}(k))}{\sum_{j} R_j(\mathbf{x}(k))} = \sum_{j} p_j(\mathbf{x}(k))(C_j\mathbf{x}(k) + D_j\mathbf{u}(k)) \tag{2.35}$$

donde

$$p_j(\mathbf{x}(k)) = \frac{R_i(\mathbf{x}(k))}{\sum_{r} R_r(\mathbf{x}(k))} \tag{2.36}$$

Algoritmo de retropropagación a través del tiempo

El algoritmo de retropropagación a través del tiempo [25] [**] es una extensión del algoritmo de retropropagación aplicado a estructuras recurrentes. Por ejemplo, tómese el modelo entrada-salida

$$\hat{y}(k) = f(\theta(k), \hat{y}(k-1), u(k-1)) \tag{2.37}$$

donde θ es el vector de parámetros del modelo. Dicho vector de parámetros se variará de acuerdo con el algoritmo de gradiente (2.12). Dado que será necesario calcular la derivada de la salida con respecto al vector de parámetros, se tendrá que

$$\frac{\partial \hat{y}(k)}{\partial \theta(k)} = \frac{\partial f}{\partial \theta(k)} + \frac{\partial f}{\partial \hat{y}(k-1)} \frac{\partial \hat{y}(k-1)}{\partial \theta(k)} \tag{2.38}$$

donde se puede observar que el primer sumando corresponde al término normal del algoritmo de retropropagación para sistemas estáticos, mientras que la segunda parte involucra la expresión de la derivada parcial $\partial\hat{y}/\partial\theta$ pero tomando un retraso de la señal de salida. Si seguimos calculando esta derivada, obtendremos que, para el caso $k - N$

$$\frac{\partial \hat{y}(k-N)}{\partial \theta(k)} = \frac{\partial f}{\partial \theta(k)} + \frac{\partial f}{\partial \hat{y}(k-N-1)} \frac{\partial \hat{y}(k-N-1)}{\partial \theta(k)} \tag{2.39}$$

[**] En inglés: *Backpropagation-Through-Time*

donde, si $k - N = 0$, es decir, que lleguemos hasta el tiempo inicial, se tendrá que

$$\begin{aligned} \hat{y}(0) &= 0 \\ \frac{\partial \hat{y}(0)}{\partial \theta(k)} &= 0 \end{aligned}$$

Este método cuenta con 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, es necesario observar que a medida que el número de datos de entrenamiento se incremente, así lo hará el número de derivadas que será necesario calcular para cada parámetro considerado en el vector θ; y dado que el número de muestras necesarias para entrenar una red generalmente es grande, los requerimientos computacionales de este algoritmo suelen ser demasiado demandantes para ser aplicados en forma práctica.

Aprendizaje Recurrente en Tiempo Real [26]

Este método está basado en el anterior, pero éste considera que los parámetros θ cambiarán relativamente poco a través del tiempo, por lo que se puede considerar que

$$\theta(k) \approx \theta(k-1) \approx \ldots \approx \theta(1) \tag{2.40}$$

con lo que la derivada (2.39) se puede expresar como

$$\frac{\partial \hat{y}(k-1)}{\partial \theta(k)} = \frac{\partial \hat{y}(k-1)}{\partial \theta(k-1)} \tag{2.41}$$

$$\frac{\partial \hat{y}(k)}{\partial \theta(k)} = \frac{\partial f}{\partial \theta(k)} + \frac{\partial f}{\partial \hat{y}(k-1)} \frac{\partial \hat{y}(k-1)}{\partial \theta(k-1)} \tag{2.42}$$

lo cual permite realizar un cálculo mucho más sencillo de esta derivada, ya que en cada paso de entrenamiento se tomará la derivada anterior para efectuar el cálculo de la nueva derivada, en lugar de "desdoblarla" a través del tiempo, como ocurre en el método de retropropagación mostrado en la sección 2.3.2.

Cabe mencionar que los dos algoritmos mostrados previamente sufren de los mismos inconvenientes que sus símiles para sistemas estáticos: asumen que la función de costo será convexa y suave con respecto a los parámetros y suelen presentar una relativa sensibilidad al ruido y al valor de la ganancia de adaptación. En caso de que no se pueda garantizar la convexidad global, se hace la suposición de que el algoritmo comenzará la búsqueda en la

vecindad de un *buen* mínimo local, por lo que en el caso de funciones de costo relativamente complejas será igualmente necesario contar con un algoritmo de inicialización de parámetros que le sitúe cerca de algún mínimo, idealmente el *mínimo global*.

2.4. Estabilidad de sistemas difusos recurrentes

Una vez establecido un sistema difuso recurrente, es importante contar con un método para determinar la estabilidad en el sentido de Lyapunov del sistema sin excitación [11]. Si se considera la operación del sistema (2.35) con $u = 0$, obtendremos que el sistema quedará definido por la ecuación recurrente

$$\mathbf{x}(k+1) = \left(\sum_{j+1}^{m} p_j(\mathbf{x}(k)) C_j \right) \mathbf{x}(k) \tag{2.43}$$

Ahora bien, para analizar la estabilidad del sistema (2.43), se propone la candidata a función de Lyapunov

$$L(k) = \mathbf{x}^T(k) P \mathbf{x}(k) \tag{2.44}$$

donde P es una matriz positiva definida.

Para que el sistema (2.43) sea estable es necesario que $\Delta L(k) \leq 0$. Desarrollando esta expresión, se observa que:

$$\begin{aligned} \Delta L(k) &= L(k+1) - L(k) \\ &= \mathbf{x}^T(k+1) P \mathbf{x}(k+1) - \mathbf{x}^T(k) P \mathbf{x}(k) \\ &= \left(\sum_{j=1}^{m} p_j \mathbf{x}^T(k) C_j^T \right) P \left(\sum_{j=1}^{m} p_j C_j \mathbf{x}(k) \right) - \mathbf{x}^T(k) P \mathbf{x}(k) \\ &= \mathbf{x}^T(k) \left(\left(\sum_{j=1}^{m} p_j C_j^T \right) P \left(\sum_{j=1}^{m} p_j C_j \right) - P \right) \mathbf{x}(k) \end{aligned} \tag{2.45}$$

Por simplicidad de notación se considerará $\mathbf{x} = \mathbf{x}(k)$. Si se desarrolla la suma de pesos de reglas

$$\sum_j p_j = \frac{R_1}{\sum_r R_r} + \ldots + \frac{R_m}{\sum_r R_r} = \frac{\sum_r R_r}{\sum_r R_r} = 1 \tag{2.46}$$

Entonces

$$\overbrace{\left(\sum_j^m p_j\right)}^{=1}\overbrace{\left(\sum_i^m p_i\right)}^{=1} = 1 \tag{2.47}$$

por lo que se puede aplicar la igualdad

$$\sum_i \sum_j p_i p_j = 1 \tag{2.48}$$

En vista de lo anterior, es posible multiplicar P por la ecuación (2.48), con lo que en la parte derecha de la ecuación (2.45) queda establecida como

$$\begin{aligned}
\Delta L &= \mathbf{x}^T \left(\left(\sum_i^m p_i C_i^T \right) P \left(\sum_j^m p_j C_j \right) - \sum_i \sum_j p_i p_j P \right) \mathbf{x} \\
&= \mathbf{x}^T \left(\sum_i^m p_i C_i^T P \sum_j p_j C_j - p_i \sum_j p_j P \right) \mathbf{x} \\
&= \mathbf{x}^T \left(\sum_i \sum_j p_i C_i^T P p_j C_j - p_i p_j P \right) \mathbf{x} \\
&= \mathbf{x}^T \left(\sum_i \sum_j p_i p_j (C_i^T P C_j - P) \right) \mathbf{x}
\end{aligned} \tag{2.49}$$

Si desarrollamos la suma, especificando el término $i = j$

$$\Delta L = \mathbf{x}^T \left(\sum_j p_j^2 (C_j^T P C_j - P) + \sum_{i \neq j} \sum_j p_i p_j (C_i^T P C_j - P) \right) \mathbf{x} \tag{2.50}$$

Ahora, supongamos que la matriz P es tal que $\forall\ j$

$$C_j^T P C_j - P \leq 0 \tag{2.51}$$

Entonces el primer sumando de la ecuación (2.50) será negativo, con lo que únicamente queda analizar el segundo sumando para encontrar la condición de estabilidad. Dado que sabemos que $C_j^T P C_i = C_i^T P C_j$, entonces

$$\begin{aligned}
2(C_i^T P C_j - P) &= C_i^T P C_j - P + C_j^T P C_i - P \\
&= C_i^T P C_j + C_j^T P C_i - C_i^T P C_i - C_j^T P C_j + C_i^T P C_i + C_j^T P C_j - P - P \\
&= -[C_i - C_j]^T P [C_i - C_j] + (C_i^T P C_i - P) + (C_j^T P C_j - P)
\end{aligned} \tag{2.52}$$

De donde, atendiendo a la suposición dada por (2.51) para el primer elemento de la ecuación (2.52) y la suposición (2.51) para el segundo, el resultado será que

$$\Delta L(k) < 0 \tag{2.53}$$

lo que implicará una estabilidad global y asintótica si p_j está definido para toda $\mathbf{x}$. En caso contrario, el resultado será local.

Con lo que podemos postular el siguiente teorema [12]:

Teorema 2.2 *Estabilidad de sistemas difusos recurrentes*

El sistema difuso recurrente generalizado sin entrada externa dado por la ecuación (2.43) es global y asintóticamente estable si existe una matriz P positiva definida común tal que se cumpla

$$C_j^T P C_j - P < 0 \;\; \forall j = 1, \; \ldots, \; m \tag{2.54}$$

y que $p_j(\mathbf{x})$ dado por la ecuación (2.36) esté definido para toda $\mathbf{x} \in \Re^n$. En caso contrario el resultado será local.

Cabe mencionar que este este teorema es un tanto restrictivo con respecto al espacio de búsqueda, ya que además de requerir que todas las matrices C_i sean estables requiere de que exista una $P > 0$ común positiva definida que cumpla con (2.51); además de que requiere ser aplicado al sistema difuso una vez entrenado.

Capítulo 3

Identificación mediante una red neurodifusa recurrente

3.1. Introducción

Dentro del presente capítulo se analizará el problema general de identificación de sistemas no lineales y se mostrarán las ventajas de una representación de un sistema no lineal mediante un modelo con dinámica interna con respecto a un modelo que únicamente considera la dinámica externa. A continuación se expondrá el esquema propuesto con el que se busca atacar el problema de identificación de sistemas no lineales, el cual está basado en una red neurodifusa con dos etapas: una recurrente encargada de manejar los estados internos, y una estática enfocada a obtener la salida estimada. Con base en el esquema propuesto, se mostrará un algoritmo de inicialización de parámetros basado en el entrenamiento de estructuras estáticas difusas a partir de las cuales se pueda extraer información útil para la red propuesta. Una vez que la red posea parámetros iniciales, se mostrará el algoritmo de entrenamiento mediante el cual se sintonizarán los parámetros para lograr minimizar el error entre la señal de referencia y la obtenida de la red.

3.2. Sistemas no lineales

Cuando se busca representar sistemas lineales, una de las formas más directas de modelarlos es a partir de un esquema entrada/salida en la forma

$$\hat{y}(k) = b_1 u(k-1) + \ldots + b_m u(k-m) - a_1 y(k-1) - \ldots - a_n y(k-n) \qquad (3.1)$$

La ecuación (3.1) es conocida como *modelo autorregesivo con entrada externa y promedio variante* o ARMAX* por sus siglas en inglés.

En el caso de que un sistema lineal no sea suficiente como para describir el comportamiento de un sistema, se debe entonces considerar la representación no lineal, cuyo modelo general es una extensión del modelo lineal dado por una función no lineal $f(\cdot)$ dada por

$$\hat{y}(k) = f(u(k-1), \ldots, u(k-m), y(k-1), \ldots, y(k-n)) \tag{3.2}$$

el cual es conocido como *modelo no lineal autorregresivo con entrada externa* o NARMAX**.

Pese a que los modelos tipo NARMAX y otras variaciones pueden cubrir a un buen número de sistemas no lineales, la realidad es que posiblemente no consideren otro tipo de dinámicas internas del sistema que no se vean directamente reflejadas en la salida.

Un esquema más general es obtenido a partir de una representación en espacio de estados, la cual estará dada en un dominio discreto como

$$\mathbf{x}(k+1) = \mathbf{f}_d(\mathbf{x}(k), \mathbf{u}(k)) \tag{3.3}$$

$$y(k) = h_d(\mathbf{x}(k)) \tag{3.4}$$

donde $\mathbf{x} \in \Re^{\bar{n}}$, $\mathbf{u} \in \Re^m$, $y \in \Re$, $\mathbf{f}_d : \Re^{\bar{n}} \times \Re^m \to \Re^{\bar{n}}$ y $h_d : \Re^{\bar{n}} \to \Re$.

El objetivo es entonces encontrar una red neurodifusa recurrente con una estructura

$$\mathbf{z}(k+1) = \mathbf{f}(\mathbf{z}(k), \mathbf{u}(k)) \tag{3.5}$$

$$\hat{y} = h(\mathbf{z}(k)) \tag{3.6}$$

donde $\mathbf{z} = [z_1\ z_2\ \ldots\ z_n]^T \in \Re^n$, $\hat{y} \in \Re$ tal que logre que el criterio

$$J(k) = \frac{1}{2}(\hat{y(k)} - y(k))^2 \triangleq \frac{1}{2}e^2(k) \tag{3.7}$$

se minimice, donde $i = 1$, ..., T_n con T_n como el número de muestras.

Autoregressive with Exogenous-Input-Moving-Average Model

**Nonlinear Autoregressive with Exogenous-Input-Moving-Average Model*

3.3. Estructura del identificador propuesta

Se propone entonces que la red neurodifusa tenga la estructura que se ve en la Figura 3.1, donde el vector **z** representa al vector de estados que serán realimentados a la función **f**. Las funciones **f** (red recurrente) y h (red estática) se definen mediante dos sistemas difusos, donde la primera toma la forma vista en la ecuación (2.33), mientras que la segunda es un sistema difuso estático tipo Takagi-Sugeno. De esta forma, las reglas toman la forma:

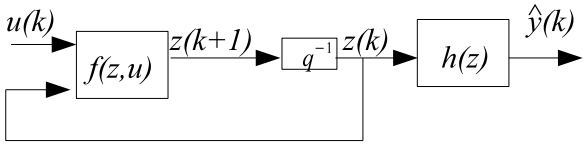


Figura 3.1: Representación gráfica de la red Neurodifusa propuesta

Para $f(\cdot,\cdot)$:

$$R_f^n: \text{ Si } z_n \text{ es } A_n \text{ y } \mathbf{u} \text{ es } B_n \text{ entonces } \mathbf{w}_n(k+1) = C_n\mathbf{z}(k) + D_n\mathbf{u}(k) \tag{3.8}$$

Para $h(\cdot)$:

$$R_h^p: \text{ Si } \mathbf{z} \text{ es } H_n \text{ entonces } \gamma_p(k) = \mathbf{h}_n^T\bar{\mathbf{z}}(k) \tag{3.9}$$

donde $n = 1, 2, \ldots n_{Rf}$, $n_z = \dim(\mathbf{z})$, $n_u = \dim(\mathbf{u})$, $C_n \in \Re^{n_z \times n_z}$, $D_n \in \Re^{n_z \times n_u}$. A_n y B_n son conjuntos difusos asociados a z_n y $\mathbf{u}$ respectivamente; $\mathbf{h}_n = \Re^{(n_z+1)\times 1}$, y n_{Rf} y n_{Rh} representan al número de reglas que definen a los sistemas difusos f y h respectivamente, y se puede observar que está definido de forma tal que $n_z = n_{Rh}$.

Por otro lado,$\bar{\mathbf{z}}(k)$ se define como $\bar{\mathbf{z}}(k) = [1\ \mathbf{z}]^T$, $\mathbf{w}_n(k+1)$ es la propuesta de la n-ésima regla de **f** para el valor de $\mathbf{z}(k)$ y γ_p es la salida de la p-ésima regla de h.

Las funciones de membresía A_n, B_n y H_p se definen como gaussianas en la forma

$$A_n(z_n) = e^{-\sigma_n^2(z_n-\mu_n)^2} \tag{3.10}$$

$$B_n(\mathbf{u}) = e^{-\sum_{i=1}^{n_u} s_{ni}^2(u_i-m_{ni})^2} \tag{3.11}$$

$$H_p(\mathbf{z}) = e^{-\sum_{i=1}^{n_z} \alpha_{pi}^2(z_i-\beta_{pi})^2} \tag{3.12}$$

donde:

- σ_n : Ancho de la n-ésima función de membresía gaussiana relacionada con z_n en $\mathbf{f}$
- μ_n : Centro de la n-ésima función de membresía gaussiana relacionada con z_n en $\mathbf{f}$
- s_{ni} : Ancho de la función de membresía relacionada con la i-ésima entrada en la n-ésima regla
- m_{ni} : Centro de la función de membresía relacionada con la i-ésima entrada y la n-ésima regla
- α_{pi} : Ancho de la i-ésima función de membresía gaussiana relacionada con z_i en la p-ésima regla en h
- β_{pi} : Centro de la i-ésima función de membresía gaussiana relacionada con z_i en la p-ésima regla en h
- $C_n \in \Re^{n_z \times n_z}$
- $D_n \in \Re^{n_z \times n_u}$
- $h_n \in \Re^{(n_z+1)\times 1}$

Se considerará una t-norma producto, por lo que los pesos de las reglas R_f^n y R_h^n se pueden escribir como:

$$\begin{aligned} R_f^n(k) &= A_n(z_n)B_n(\mathbf{u}) \\ &= e^{-\sigma_n^2(z_n-\mu_n)^2-\sum_{j=1}^{n_u} s_{nj}^2(u_j-m_{nj})^2} \qquad (3.13) \\ R_h^p(k) &= H_p(\mathbf{z}) \qquad (3.14) \end{aligned}$$

En el caso SISO[***] (es decir, $n_u = 1$), las variables $\mathbf{z}(k)$, $\mathbf{w}(k)$ y $y(k)$ se definen como

$$\mathbf{z}(k+1) = \frac{\sum_{j=1}^{n_{Rf}} R_f^j(k)\mathbf{w}_j(k+1)}{\sum_{j=1}^{n_Rf} R_f^j(k)} \tag{3.15}$$

$$\begin{aligned}\mathbf{w}_n(k+1) &= \begin{pmatrix} w_{n1}(k+1) \\ \vdots \\ w_{nj}(k+1) \end{pmatrix} = \begin{pmatrix} c_{1n} & c_{2n} & \dots & c_{n_z-1n} & c_{n_zn} \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{pmatrix} \mathbf{z}(k) + \begin{pmatrix} d_n \\ 0 \\ \vdots \\ 0 \end{pmatrix} u(k) \\ &= \begin{pmatrix} \mathbf{c}_n^T & \\ I & \mathbf{0} \end{pmatrix} \mathbf{z}(k) + D_n u(k) \triangleq C_n \mathbf{z}(k) + D_n u(k) \end{aligned} \tag{3.16}$$

$$\hat{y}(k) = \frac{\sum_{i=1}^{n_{Rh}} R_h^i \gamma_i(k)}{\sum_{i=1}^{n_{Rh}} R_h^i} \tag{3.17}$$

donde $\mathbf{c}_n^T = [c_{1n}\ c_{2n}\ \dots\ c_{n_zn}]$. Se define n_{Rf} y n_{Rh} como el número de reglas para las funciones $\mathbf{f}$ y h respectivamente.

Las ventajas que proporciona la definición de la red propuesta se pueden resumir en los siguientes puntos:

- *La estructura de la red permite analizar la estabilidad con métodos ya existentes.* Dado que la red recurrente $\mathbf{f}$ tiene la misma forma que la mostrada en (2.43), una vez entrenada la red es posible efectuar un análisis de estabilidad en el sentido de Lyapunov sobre ella y por lo tanto aplicar el Teorema 2.2.

- *Número reducido de parámetros recurrentes.* La estructura propuesta simplifica el número de parámetros a encontrar con respecto a las matrices C_n, ya que si $C_n \in \Re^{n_z \times n_z}$, el número de parámetros a determinar será $n_{R_f} \times n_z$ (en vez de $n_{R_f} \times n_z \times n_z$). Asimismo, dado que la i-ésima regla de $\mathbf{f}$ requiere información sobre el estado z_i y no de los estados $z_j,\ j \neq i$, el número de parámetros asociado a los conjuntos difusos A_j será $2 \times n_{R_f}$.

- *Incorporación de conocimiento previo.* El modelo propuesto permite incorporar co-

[***] Una entrada, una salida. En inglés *Single-Input-Single-Output*

nocimiento previo, como se analizará en el algoritmo de inicialización de parámetros presentado en la Sección 3.5.1, a diferencia de otro tipo de métodos ([10] y [8]) donde no se describe cómo realizar este proceso.

- *Estructura en forma de variables de estado.* La estructura propuesta considera una representación en forma de variables de estado, lo cual facilita la implementación de esquemas de control basados en ésta.

De la propia estructura podemos notar las siguientes limitaciones:

- *Existencia de un único punto de equilibrio.* En las reglas de la función $\mathbf{f}$ se consideran sistemas lineales. Por esta causa, al ser $\mathbf{u} = 0$ el único punto de equilibrio al que pueden tender (en caso de ser estables en el sentido de Lyapunov) es el origen. Esta condición es una limitante si se desea identificar sistemas no lineales con más de un punto de equilibrio.

- *Parametrización no lineal.* No todos los parámetros de la red propuesta no pueden agruparse como en (2.26), sino que se relacionan de forma igualmente no lineal, con el correspondiente aumento en la complejidad del entrenamiento.

3.4. Entrenamiento de la red

Dado que la estructura propuesta involucra una red recurrente y una estática, el entrenamiento de los parámetros de $\mathbf{f}$ se lleva a cabo mediante el método de aprendizaje recurrente en tiempo real basado en gradiente (*Gradient-Based RT Recurrent Learning* [26][3]), mientras que aquellos de h son sintonizados mediante el método simple de gradiente *steepest descent*. Se consdierará entonces la actualización de parámetros en la forma

$$\theta(k+1) = \theta(k) - \eta_\theta \frac{\partial J(k)}{\partial \theta} \tag{3.18}$$

3.4.1. Identificación de los parámetros asociados a la red estática

Parámetros $\mathbf{h}_n^T$

Estos son los parámetros asociados con la parte consecuente de la n-ésima regla de la red estática h, tal y como se aprecia en (3.9). Para el entrenamiento de estos parámetros se

propone la ley de entrenamiento basada en (2.42), por lo que se obtiene que

$$\mathbf{h}_n^T(k+1) = \mathbf{h}_n^T(k) - \eta_h \frac{\partial J(k)}{\partial \mathbf{h}_n^T} \tag{3.19}$$

donde, desarrollando la derivada parcial para $J(k)$ de acuerdo con (3.7), obtenemos:

$$\frac{\partial J(k)}{\partial \mathbf{h}_n^T} = e(k) \frac{\partial \hat{y}(k)}{\partial \mathbf{h}_n^T} \tag{3.20}$$

en donde, derivando (3.17), obtenemos

$$\frac{\partial \hat{y}(k)}{\partial \mathbf{h}_n^T} = \frac{R_h^n(k)}{\sum_j R_h^j(k)} \bar{\mathbf{z}^T}(k) \tag{3.21}$$

Parámetros α_{ij}

Los parámetros α_{ij} son los parámetros asociados con el ancho de las funciones gaussianas que representan a los conjuntos difusos H_i de la parte antecedente de las reglas de la red estática h, correspondientes a z_j en la i-ésima regla. Para su adaptación, se toma de igual forma una ley de adaptación basada en el gradiente:

$$\alpha_{ij}(k+1) = \alpha_{ij}(k) - \eta_\alpha \frac{\partial J(k)}{\partial \alpha_{ij}} \tag{3.22}$$

en donde las derivadas parciales se calculan con base en (3.9), (3.12) y (3.17):

$$\frac{\partial J(k)}{\partial \alpha_{ij}} = e(k) \frac{\partial \hat{y}(k)}{\partial \alpha_{ij}} \tag{3.23}$$

$$\frac{\partial \hat{y}(k)}{\partial \alpha_{ij}} = \left[\frac{\gamma_i(k)}{\sum_r R_h^r(k)} - \frac{\sum_r R_h^r(k)\gamma_r(k)}{\left(\sum_r R_h^r(k) \right)^2} \right] \frac{\partial R_h^i(k)}{\partial \alpha_{ij}} \tag{3.24}$$

$$\frac{\partial R_h^l}{\partial \alpha_{ij}} = \begin{cases} -2\alpha_{ij}(z_j(k) - \beta_{ij})^2 R_h^l(k) & \text{si } l = i \\ 0 & \text{si } l \neq i \end{cases} \tag{3.25}$$

Parámetros β_{ij}

Los parámetros β_{ij} son los parámetros asociados con el centro de las funciones gaussianas que representan a los conjuntos difusos H_i de la parte antecedente de las reglas de la red

estática h, correspondientes a z_j en la i-ésima regla. Considerando la misma forma de la ley de adaptación para sistemas estáticos, tenemos que

$$\beta_{ij}(k+1) = \beta_{ij}(k) - \eta_\beta \frac{\partial e(k)}{\partial \beta_{ij}} \tag{3.26}$$

calculando las derivadas parciales con base en (3.9), (3.12) y (3.17), se obtiene:

$$\frac{\partial J(k)}{\partial \beta_{ij}} = e(k)\frac{\partial \hat{y}(k)}{\partial \beta_{ij}} \tag{3.27}$$

$$\frac{\partial \hat{y}(k)}{\partial \beta_{ij}} = \left[\frac{\gamma_i(k)}{\sum_r R_h^r(k)} - \frac{\sum_r R_h^r(k)\gamma_r(k)}{\left(\sum_r R_h^r(k)\right)^2} \right] \frac{\partial R_h^i(k)}{\partial \beta_{ij}} \tag{3.28}$$

$$\frac{\partial R_h^l}{\partial \beta_{ij}} = \begin{cases} 2\alpha_{ij}^2(z_j(k) - \beta_{ij})R_h^l(k) & \text{si } l = i \\ 0 & \text{si } l \neq i \end{cases} \tag{3.29}$$

3.4.2. Identificación de los parámetros asociados a la red recurrente

Parámetros $\mathbf{c}_n$

Los parámetros $\mathbf{c}_n$ son aquellos asociados con el primer renglón de las matrices C_n de la cada una de las reglas de la parte consecuente red recurrente f, como se aprecia en la ecuación (3.16). En vista de que estos parámetros están asociados con la red recurrente $\mathbf{f}$,es necesario apelar a esta propiedad para establecer la ley de adaptación. Por lo tanto, se emplea el método de *Aprendizaje Recurrente en Tiempo Real* por medio de la ecuación (2.42), por lo que la ley de adaptación resultante es

$$\mathbf{c}_n(k+1) = \mathbf{c}_n(k) - \eta_c \frac{\partial J(k)}{\partial \mathbf{c}_n} \tag{3.30}$$

en donde

$$\frac{\partial J(k)}{\partial \mathbf{c}_n} = e(k)\frac{\partial \hat{y}(k)}{\partial \mathbf{c}_n} \tag{3.31}$$

Esta derivada se calcula a partir de (3.17), por lo que se obtiene

$$\frac{\partial \hat{y}(k)}{\partial \mathbf{c}_n} = \frac{\sum_r \gamma_r(k) \frac{\partial R_h^r(k)}{\partial \mathbf{c}_n} + R_h^r(k) \bar{\mathbf{h}}^T \frac{\partial \mathbf{z}(k)}{\partial \mathbf{c}_n}}{\sum_r R_h^r(k)} - \frac{\left(\sum_r R_h^r(k)\gamma_r(k)\right)\left(\sum_r \frac{\partial R_h^r(k)}{\partial \mathbf{c}_n}\right)}{\left(\sum_r R_h^r(k)\right)^2} \tag{3.32}$$

con

$$\bar{\mathbf{h}}_r^T = [h_{1r}\ h_{2r}\ \ldots\ h_{1n_z}] \tag{3.33}$$

$$\frac{\partial R_h^r(k)}{\partial \mathbf{c}_n} = -2R_h^r(k)\left(\sum_l \alpha_{rl}^2 (z_l(k) - \beta_{rl}) \frac{\partial z_l(k)}{\partial \mathbf{c}_n}\right) \tag{3.34}$$

Calculando esta derivada parcial a partir de (3.15), (3.13) y (3.16) se obtiene la ecuación recurrente

$$\begin{aligned} \frac{\partial \mathbf{z}(k)}{\partial \mathbf{c}_n} &= \frac{\sum_r R_f^r(k-1) \frac{\partial \mathbf{w}^r(k)}{\partial \mathbf{c}_n} + \mathbf{w}^r(k)\left(\frac{\partial R_f^r(k-1)}{\partial \mathbf{c}_n}\right)^T}{\sum_r R_f^r(k-1)} \\ &\quad - \frac{\left(\sum_r R_f^r(k-1)\mathbf{w}^r(k)\right)\left(\frac{\partial R_f^r(k-1)}{\partial \mathbf{c}_n}\right)^T}{\left(\sum_r R_f^r(k-1)\right)^2} \end{aligned} \tag{3.35}$$

$$\frac{\partial R_f^r(k-1)}{\partial \mathbf{c}_n} = -2R_f^r(k-1)\sigma_r^2(z_r(k-1) - \mu_r)\frac{\partial z_r(k-1)}{\partial \mathbf{c}_n} \tag{3.36}$$

$$\frac{\partial w_1^r(k)}{\partial \mathbf{c}_n} = \mathbf{c}_n \frac{\partial \mathbf{z}(k-1)}{\partial \mathbf{c}_n} + \mathbf{z}^T(k-1)\delta_{nr} \tag{3.37}$$

$$\frac{\partial w_j^r(k)}{\partial \mathbf{c}_n} = \frac{\partial z_{j-1}(k-1)}{\partial \mathbf{c}_n},\ \ j = 2,\, 3,\, \ldots,\, n_z \tag{3.38}$$

Parámetros σ_i

Los parámetros σ_i se definen como los anchos de las funciones gaussianas de los conjuntos difusos A_i, correspondientes a la i-ésima regla, como se aprecia en la ecuación (3.10). En vista de que se encuentran asociados con la red recurrente, se vuelve a apelar a la ley de adaptación de parámetros (2.42). De esta forma, la ley de adaptación utilizada queda definida como

$$\sigma_i(k+1) = \sigma_i(k) - \eta_\sigma \frac{\partial J(k)}{\partial \sigma_i} \tag{3.39}$$

en donde, a partir de (3.17) se obtiene

$$\frac{\partial J(k)}{\partial \sigma_i} = e(k)\frac{\partial \hat{y}(k)}{\partial \sigma_i} \tag{3.40}$$

$$\frac{\partial \hat{y}(k)}{\partial \sigma_i} = \frac{\sum_r \gamma_r(k)\frac{\partial R_h^r(k)}{\partial \sigma_i} + R_h^r(k)\bar{\mathbf{h}}_r^T\frac{\partial \mathbf{z}(k)}{\partial \sigma_i}}{\sum_r R_h^r(k)} - \frac{\left(\sum_r R_h^r(k)\gamma_r(k)\right)\left(\sum_r \frac{\partial R_h^r(k)}{\partial \sigma_i}\right)}{\left(\sum_r R_h^r(k)\right)^2} \tag{3.41}$$

Aplicando (3.14), se puede calcular la derivada parcial $\frac{\partial R_h^r}{\partial \sigma_i}$ como

$$\frac{\partial R_h^r(k)}{\partial \sigma_i} = -2R_h^r(k)\left(\sum_l \alpha_{rl}^2(z_l(k)-\beta_{rl})\frac{\partial z_l(k)}{\partial \sigma_i}\right) \tag{3.42}$$

Al calcular las derivadas parciales de (3.13), (3.15) y (3.16) se obtiene la ecuación recurrente dada por:

$$\begin{aligned}\frac{\partial \mathbf{z}(k)}{\partial \sigma_i} = & \frac{\sum_q R_f^q(k-1)\frac{\partial \mathbf{w}^q(k)}{\partial \sigma_i} + \mathbf{w}^q(k)\frac{\partial R_f^q(k-1)}{\partial \sigma_i}}{\sum_p R_f^p(k-1)} \\ & - \frac{\left(\sum_q R_f^q(k-1)\mathbf{w}^q(k)\right)\left(\sum_s \frac{\partial R_f^s(k-1)}{\partial \sigma_i}\right)}{\left(\sum_p R_f^p(k-1)\right)^2}\end{aligned} \tag{3.43}$$

$$\frac{\partial R_f^q}{\partial \sigma_i} = -2R_f^q(k-1)\left(\sigma_i^2(z_i(k-1)-\mu_i)\frac{\partial z_i(k-1)}{\partial \sigma_i} + \sigma_i(z_i(k-1)-\mu_i)^2\right) \tag{3.44}$$

$$\frac{\partial \mathbf{w}^q(k)}{\partial \sigma_i} = C_q\frac{\partial \mathbf{z}(k-1)}{\partial \sigma_i} \tag{3.45}$$

Parámetros μ_i

Los parámetros μ_i se definen como los centros de las funciones gaussianas de los conjuntos difusos A_i, como se aprecia en la ecuación (3.10). La ley de adaptación utilizada es

parecida a la de la sección anterior, ya que está dada por

$$\mu_i(k+1) = \mu_i(k) - \eta_\mu \frac{\partial J(k)}{\partial \mu_i} \tag{3.46}$$

donde

$$\frac{\partial J(k)}{\partial \mu_i} = e(k)\frac{\partial \hat{y}(k)}{\partial \mu_i} \tag{3.47}$$

$$\frac{\partial \hat{y}(k)}{\partial \mu_i} = \frac{\sum_r \gamma_r(k)\frac{\partial R_h^r(k)}{\partial \mu_i} + R_h^r(k)\bar{\mathbf{h}}_r^T \frac{\partial \mathbf{z}(k)}{\partial \mu_i}}{\sum_r R_h^r(k)} - \frac{\left(\sum_r R_h^r(k)\gamma_r(k)\right)\left(\sum_r \frac{\partial R_h^r(k)}{\partial \mu_i}\right)}{\left(\sum_r R_h^r(k)\right)^2} \tag{3.48}$$

$$\frac{\partial R_h^r(k)}{\partial \mu_i} = -2R_h^r(k)\left(\sum_l \alpha_{rl}^2(z_l(k) - \beta_{rl})\frac{\partial z_l(k)}{\partial \mu_i}\right) \tag{3.49}$$

Al determinar las derivadas parciales de (3.13), (3.15) y (3.16) se obtiene la ecuación recurrente:

$$\begin{aligned}\frac{\partial \mathbf{z}(k)}{\partial \mu_i} &= \frac{\sum_q R_f^q(k-1)\frac{\partial \mathbf{w}^q(k)}{\partial \mu_i} + \mathbf{w}^q(k)\frac{\partial R_f^q(k-1)}{\partial \mu_i}}{\sum_p R_f^p(k-1)} \\ &\quad - \frac{\left(\sum_q R_f^q(k-1)\mathbf{w}^q(k)\right)\left(\sum_s \frac{\partial R_f^s(k-1)^T}{\partial \mu_i}\right)}{\left(\sum_p R_f^p(k-1)\right)^2}\end{aligned} \tag{3.50}$$

$$\frac{\partial R_f^q}{\partial \mu_i} = -2R_f^q(k-1)\left(\sigma_i^2(z_i(k-1) - \mu_i)\frac{\partial z_i(k-1)}{\partial \mu_i} - \sigma_i^2(z_j(k-1) - \mu_i)\right) \tag{3.51}$$

$$\frac{\partial \mathbf{w}^q(k)}{\partial \sigma_i} = C_q \frac{\partial \mathbf{z}(k-1)}{\partial \sigma_i} \tag{3.52}$$

Parámetros s_{ij}

En este caso, los parámetros s_{ij} se definen como los anchos relacionados con las funciones gaussianas de los conjuntos difusos B_n relacionados con la i-ésima regla y la j-ésima entrada,

de acuerdo con la ecuación (3.11). Si se aplica la misma ley de adaptación dada por (2.42), se obtiene

$$s_{ij}(k+1) = s_{ij}(k) - \eta_s \frac{\partial J(k)}{\partial s_{ij}} \tag{3.53}$$

en donde derivando (3.17) se obtiene

$$\frac{\partial J(k)}{\partial s_{ij}} = e(k) \frac{\partial \hat{y}(k)}{\partial s_{ij}} \tag{3.54}$$

$$\frac{\partial \hat{y}(k)}{\partial s_{ij}} = \frac{\sum_r \gamma_r(k) \frac{\partial R_h^r(k)}{\partial s_{ij}} + R_h^r(k) \bar{\mathbf{h}}_r^T \frac{\partial \mathbf{z}(k)}{\partial s_{ij}}}{\sum_r R_h^r(k)} - \frac{\left(\sum_r R_h^r(k)\gamma_r(k)\right)\left(\sum_r \frac{\partial R_h^r(k)}{\partial s_{ij}}\right)}{\left(\sum_r R_h^r(k)\right)^2} \tag{3.55}$$

Si se deriva (3.14), resulta en

$$\frac{\partial R_h^r(k)}{\partial s_{ij}} = -2R_h^r(k)\left(\sum_l \alpha_{rl}^2 (z_l(k) - \beta_{rl}) \frac{\partial z_l(k)}{\partial s_{ij}}\right) \tag{3.56}$$

Nuevamente, si se determinan las derivadas parciales de (3.13), (3.15) y (3.16) se obtiene la ecuación recurrente dada por las siguientes ecuaciones

$$\begin{aligned} \frac{\partial \mathbf{z}(k)}{\partial s_{ij}} = & \frac{\sum_q R_f^q(k-1) \frac{\partial \mathbf{w}^q(k)}{\partial \mu_{ij}} + \mathbf{w}^q(k) \frac{\partial R_f^q(k-1)}{\partial s_{ij}}}{\sum_p R_f^p(k-1)} \\ & - \frac{\left(\sum_q R_f^q(k-1)\mathbf{w}^q(k)\right)\left(\sum_s \frac{\partial R_f^s(k-1)^T}{\partial s_{ij}}\right)}{\left(\sum_p R_f^p(k-1)\right)^2} \end{aligned} \tag{3.57}$$

$$\frac{\partial R_f^q}{\partial s_{ij}} = -2R_f^q(k-1)\left(\sigma_q^2 (z_q(k-1) - \mu_q) \frac{\partial z_q(k-1)}{\partial s_{ij}} + s_{ij}(u_j(k-1) - \mu_{ij})^2 \delta_{qi}\right) \tag{3.58}$$

$$\frac{\partial \mathbf{w}^q(k)}{\partial s_{ij}} = C_q \frac{\partial \mathbf{z}(k-1)}{\partial s_{ij}} \tag{3.59}$$

Parámetros m_{ij}

Finalmente, los parámetros m_i se definen como los centros relacionados con las funciones gaussianas de los conjuntos difusos B_n, en la i-ésima regla y para la j-ésima entrada, de acuerdo con la ecuación (3.11). De forma similar al cálculo de los parámetros s_{ij}, la ley de adaptación empleada es

$$m_{ij}(k+1) = m_{ij}(k) - \eta_m \frac{\partial J(k)}{\partial m_{ij}} \tag{3.60}$$

donde

$$\frac{\partial J(k)}{\partial m_{ij}} = e(k)\frac{\partial \hat{y}(k)}{\partial m_{ij}} \tag{3.61}$$

$$\frac{\partial \hat{y}(k)}{\partial m_{ij}} = \frac{\sum_r \gamma_r(k)\frac{\partial R_h^r(k)}{\partial m_{ij}} + R_h^r(k)\bar{\mathbf{h}}_r^T \frac{\partial \mathbf{z}(k)}{\partial m_{ij}}}{\sum_r R_h^r(k)} - \frac{\left(\sum_r R_h^r(k)\gamma_r(k)\right)\left(\sum_r \frac{\partial R_h^r(k)}{\partial s_{ij}}\right)}{\left(\sum_r R_h^r(k)\right)^2} \tag{3.62}$$

$$\frac{\partial R_h^r(k)}{\partial m_{ij}} = -2R_h^r(k)\left(\sum_l \alpha_{rl}^2(z_l(k)-\beta_{rl})\frac{\partial z_l(k)}{\partial m_{ij}}\right) \tag{3.63}$$

Al calcular las derivadas parciales de (3.13), (3.15) y (3.16) se obtiene la ecuación recurrente dada por las siguientes ecuaciones

$$\begin{aligned}\frac{\partial \mathbf{z}(k)}{\partial m_{ij}} &= \frac{\sum_q R_f^q(k-1)\frac{\partial \mathbf{w}^q(k)}{\partial \mu_{ij}} + \mathbf{w}^q(k)\frac{\partial R_f^q(k-1)}{\partial m_{ij}}}{\sum_p R_f^p(k-1)} \\ &\quad - \frac{\left(\sum_q R_f^q(k-1)\mathbf{w}^q(k)\right)\left(\sum_s \frac{\partial R_f^s(k-1)^T}{\partial m_{ij}}\right)}{\left(\sum_p R_f^p(k-1)\right)^2}\end{aligned} \tag{3.64}$$

$$\frac{\partial R_f^q}{\partial s_{ij}} = -2R_f^q(k-1)\left(\sigma_q^2(z_q(k-1)-\mu_q)\frac{\partial z_q(k-1)}{\partial m_{ij}} - s_{ij}^2(u_j(k-1)-\mu_{ij})\delta_{qi}\right) \tag{3.65}$$

$$\frac{\partial \mathbf{w}^q(k)}{\partial s_{ij}} = C_q \frac{\partial \mathbf{z}(k-1)}{\partial s_{ij}} \tag{3.66}$$

3.5. Rutina de inicialización de parámetros

Si bien el algoritmo propuesto permite desplazarse siempre en la dirección en la cual el error decrece con respecto al cambio de parámetros, también es cierto que éste asume que la topología del error es convexa con respecto a éstos, lo cual no es seguro ni tememos garantía de que ocurra para todo el campo de parámetros. Es por ello que debemos de encontrar alguna condición inicial para la red tal que permita situarnos cerca del mínimo global (o al menos cerca de algún *buen mínimo local*[****], para así evitar quedar atrapados en algún mínimo local que tenga un desempeño no satisfactorio para determinado criterio de aproximación, así como poder obtener la mejor aproximación posible de la red recurrente a partir de la información disponible.

El algoritmo propuesto se basa en la idea de entrenar sistemas difusos estáticos de forma tal que, dada su naturaleza difusa, se pueda extraer de ellos información tal que sea útil para conseguir una buena condición inicial.

3.5.1. Algoritmo

El algoritmo propuesto supone que existe una serie de N datos de entrada-salida. Por facilidad de presentación se expondrá el caso SISO (Una entrada-Una salida). El algoritmo que se presenta a continuación no determina el número de estados internos n_z que el sistema difuso tendrá, sino que dicho número debe de ser asignado por adelantado.

Los pasos que componen el algoritmo de inicialización de parámetros son los siguientes:

1. Normalizar a la unidad las señales de entrada u y salida y, y almacenar el valor empleado para normalizar cada señal. Este paso es necesario para que, independientemente de la amplificación que sufran las señales u y y, trabajar con señales cuya cota sea conocida.

2. Entrenar un sistema difuso estático usando n_z reglas y retrasos tanto de la salida como de la entrada mediante el método *ANFIS* [9]. En el caso $n_z = 3$ esto supone entrenar un sistema difuso en donde, siendo $y(k)$ la señal de referencia, $u(k)$ la señal de entrada al sistema, y $\hat{y}(k)$ la salida de la red, entrenar un sistema difuso

$$\hat{y}(k+1) = \hat{f}(y(k), y(k-1), y(k-2), u(k), u(k-1), u(k-2)) \tag{3.67}$$

[****]La definición de *buen mínimo local* debe comprenderse como aquel valor de los parámetros que permita reducir el error entre la señal del sistema real y la del modelo propuesto hasta un nivel satisfactorio dado cierto criterio previamente establecido

tal que $\hat{f}$ tenga una estructura Takagi-Sugeno con tantas reglas como estados n_z considerados. Esto es, reglas en la forma

$$\begin{aligned} R_i \quad : \quad & \text{SI } y(k) \text{ es } \mathcal{A}_{i1} \text{ y } y(k-1) \text{ es } \mathcal{A}_{i2} \text{ y } y(k-2) \text{ es } \mathcal{A}_{i3} \\ & \text{y } u(k) \text{ es } \mathcal{B}_{i1} \text{ y } u(k-1) \text{ es } \mathcal{B}_{i2} \text{ y } u(k-2) \text{ es } \mathcal{B}_{i3}, \text{ ENTONCES} \\ & \hat{y}^i(k+1) = a_{i0} + a_{i1}y(k) + a_{i2}y(k-1) + a_{i3}y(k-2) \\ & + b_{i1}u(k) + b_{i2}u(k-1) + b_{i3}u(k-2) \end{aligned} \tag{3.68}$$

de forma que cada conjunto difuso esté relacionado con una función de membresía gaussiana, es decir

$$F_{ij}(v) = \exp\left\{-\sigma_{ij}^2(v-\mu_{ij})^2\right\} \tag{3.69}$$

y la señal estimada final estará dada por el promedio ponderado

$$\hat{y}(k+1) = \frac{\sum_i R_i(k)\hat{y}^i(k+1)}{\sum_i R_i(k)} \tag{3.70}$$

3. Dado que cada regla escrita en la forma (3.68) puede ser vista como un sistema lineal, al omitir el término constante a_{i0}, se puede aplicar la transformada $\mathcal{Z}\{\cdot\}$ a cada regla y ver a la parte consecuente de cada una de éstas como

$$\hat{Y}^i(\mathfrak{z}) = \frac{b_{i1}\mathfrak{z}^{-1} + b_{i2}\mathfrak{z}^{-2} + b_{i3}\mathfrak{z}^{-3}}{1 - a_{i1}\mathfrak{z}^{-1} - a_{i2}\mathfrak{z}^{-2} - a_{i3}\mathfrak{z}^{-3}} U(\mathfrak{z}) \tag{3.71}$$

Dado que cada polinomio $D(\mathfrak{z}^{-1}) = 1 - a_{i1}\mathfrak{z}^{-1} - a_{i2}\mathfrak{z}^{-2} - a_{i3}\mathfrak{z}^{-3}$ se puede interpretar como aquel que guarda información sobre la dinámica del sistema (polos), y dado que la estructura propuesta para $\mathbf{f}(\cdot)$ está dado en la forma canónica de controlador, se igualan los parámetros para cada regla en la forma

$$a_{ij} = c_{ij} \tag{3.72}$$

4. Para inicializar los parámetros de ancho σ_i y centro μ_i se propone escalar las funciones de membresía del tipo (3.69) mediante el teorema del valor final para sistemas lineales discretos, y tomar a una de ellas como la función de membresía inicial relacionada con

z. La elección de qué función de membresía tomar puede ser aleatoria, ya que dada la naturaleza del FIS estático entrenado y el algoritmo de agrupamiento usado por el método ANFIS, los conjuntos difusos asociados con $y(k)$, $y(k-1)$ y $y(k-2)$ serán muy similares.

Considerando una entrada escalón

$$u(k) = u_{-1}(k) = \begin{cases} 1, & k \geq 0 \\ 0, & k < 0 \end{cases} \tag{3.73}$$

sabemos que podemos calcular el valor final de cada estado interno $z_i(k)$ mediante el teorema del valor final, que indica que

$$\begin{aligned} Z_{i-final} &\triangleq \lim_{k\to\infty} z_i(k) \\ &= \lim_{\mathfrak{z}\to 1}(1-\mathfrak{z}^{-1})\frac{\mathfrak{z}^{-(i-1)}}{1-a_{i1}\mathfrak{z}^{-1}-a_{i2}\mathfrak{z}^{-2}-a_{i3}\mathfrak{z}^{-3}}U_{-1}(\mathfrak{z}) \end{aligned} \tag{3.74}$$

dado que

$$U_{-1}(\mathfrak{z}) = \mathcal{Z}\left\{u_{-1}(k)\right\} = \frac{1}{1-\mathfrak{z}^{-1}} \tag{3.75}$$

entonces, asumiendo que ningún polo esté en $\mathfrak{z}_p = 1$,

$$Z_{i-final} = \lim_{\mathfrak{z}\to 1}\frac{\mathfrak{z}^{-(i-1)}}{1-a_{i1}\mathfrak{z}^{-1}-a_{i2}\mathfrak{z}^{-2}-a_{i3}\mathfrak{z}^{-3}} = \frac{1}{1-a_{i1}-a_{i2}-a_{i3}} \tag{3.76}$$

Pero de la red estática $\mathbf{f}_s$ entrenada tenemos que para la salida de cada regla, vista también como un sistema lineal, el teorema del valor final se aplicará como

$$Y^i_{final} = \lim_{\mathfrak{z}\to 1}\frac{b_{i1}\mathfrak{z}^{-1}+b_{i2}\mathfrak{z}^{-2}+b_{i3}\mathfrak{z}^{-3}}{1-a_{i1}\mathfrak{z}^{-1}-a_{i2}\mathfrak{z}^{-2}-a_{i3}\mathfrak{z}^{-3}} = \frac{b_{i1}+b_{i2}+b_{i3}}{1-a_{i1}-a_{i2}-a_{i3}} \tag{3.77}$$

Con lo que el factor de escalamiento para la función de membresía relacionada con z_i en cada i-ésima regla será

$$ESCALA_i \triangleq \frac{Z_{i-final}}{Y_{final}} = \frac{1}{b_{i1}+b_{i2}+b_{i3}} \tag{3.78}$$

Cabe mencionar y aclarar que el método de escalamiento propuesto en este punto no busca encontrar la condición definitiva para los anchos y centros de las funciones de membresía relacionadas con cada estado interno, sino únicamente dar una aproxi-

mación de los rangos que éstos pueden tomar.

En el caso de las condiciones iniciales para las funciones de membresía B_i relacionadas con $\mathbf{u}(k)$, podemos tomar los mismos parámetros que las funciones $\mathcal{B}_{i1}$.

5. Para obtener un valor inicial de los parámetros d_i de las matrices D_i se usa el mismo concepto que en el paso anterior. En general se buscará que los valores finales de las variables $\mathbf{z}$ para cada regla se encuentre cercano a la unidad. Es decir, si tratamos a cada una de las reglas como un sistema lineal independiente, tendremos que el valor final de cada estado interno estará dado por

$$Z_{i-final} = \lim_{\mathfrak{z}\to 1} \frac{d_i \mathfrak{z}^{-(i-1)}}{1 - a_{i1}\mathfrak{z}^{-1} - a_{i2}\mathfrak{z}^{-2} - a_{i3}\mathfrak{z}^{-3}} = \frac{d_i}{1 - a_{i1} - a_{i2} - a_{i3}} \tag{3.79}$$

Si deseamos que $Z_{i-final} \to 1$, entonces iniciaremos los valores d_i con

$$d_i = 1 - \sum_j a_{ij} \tag{3.80}$$

6. Con la función difusa $\mathbf{f}(\cdot,\cdot)$ ya inicializada, generar un histórico de los estados internos $\mathbf{z}(k)$ usando como salida deseada $y(k)$ y entrenar la parte estática $h(\cdot)$ usando el algoritmo *ANFIS*, con tantas reglas como estados internos.

Este algoritmo provee entonces de un método sistemático para obtener las condiciones iniciales, previas al entrenamiento. Cabe notar que una de las principales limitaciones que presenta es que no provee de un método de determinar el número de variables de estado que considerará la red, sino que este debe de ser asignado *a priori*.

Capítulo 4

Resultados

4.1. Introducción

En este capítulo se mostrarán los resultados obtenidos al aplicar el algoritmo indicado en el Capítulo 3 para identificar dos sistemas no lineales, con el objetivo de mostrar la efectividad de la estructura, el algoritmo de identificación de parámetros propuesto y analizar la estabilidad de los modelos obtenidos. En primer lugar se muestran los resultados obtenidos mediante simulación al identificar un sistema típicamente usado como un problema de evaluación. Como segundo ejemplo se identifica un sistema físico real de laboratorio. En ambos casos se lleva a cabo un análisis de estabilidad sobre los modelos obtenidos, obteniéndose en ambos casos que son local y asintóticamente estables, en vista de que la función $p(\mathbf{z})$ está definida solo localmente para una vecindad de $\mathbf{z}$.

4.2. Simulación: Sistema de Narendra [17]

Como se ha mencionado, el objetivo es identificar sistemas dinámicos en los cuales existan elementos con "memoria", es decir, donde la salida actual sea función de salidas y entradas pasadas. Para ejemplificar el funcionamiento de la red propuesta en un problema de identificación, se usará la siguiente planta no lineal, dada por la ecuación en diferencias

$$y(k+1) = f_N\left(y(k), y(k-1), y(k-2), u(k), u(k-1)\right) \tag{4.1}$$

donde

$$f(x_1, x_2, x_3, x_4, x_5) = \frac{x_2 x_2 x_3 x_5 (x_3 - 1) + x_4}{1 + x_2^2 + x_3^2} \tag{4.2}$$

Este sistema es comúnmente usado como un sistema de referencia para comparar el desempeño de sistemas de identificación recurrentes, como por ejemplo, en [17], [21], [10] y [14].

Para este ejemplo se utiliza una red neurodifusa con 3 estados internos, donde la parte que maneja la parte dinámica cuenta con 3 reglas, así como la parte estática. Para el entrenamiento se usan únicamente las señales $y(k)$ y $u(k)$, y se emplea el algoritmo descrito en la sección 3. Para realizar el entrenamiento de la red se consideraron los parámetros $\eta_\sigma = \eta_\mu = \eta_s = \eta_m = \eta_c = 10^{-6}$, $\eta_\alpha = \eta_\beta = 2 \cdot 10^{-3}$, $\eta_h = 3 \cdot 10^{-4}$. Estos valores fueron aquellos que, tras una serie de pruebas, lograron sintonizar los parámetros sin que existieran problemas de inestabilidad del algoritmo (recordando que los métodos basados en el cálculo de un gradiente suelen ser sensibles a las ganancias de adaptación), obteniendo un menor error que el logrado por la red cuyos parámetros fueran únicamente determinados por el algoritmo de inicialización. Los parámetros de la red obtenidos tras el entrenamiento fueron los siguientes:

$$\sigma = \begin{bmatrix} 0{,}08583 & 0{,}1471 & 0{,}1048 \end{bmatrix}$$

$$\mu = \begin{bmatrix} -2{,}819 & -0{,}08480 & 2{,}017 \end{bmatrix}$$

$$s = \begin{bmatrix} 2{,}361 \\ 2{,}441 \\ 2{,}187 \end{bmatrix} \quad m = \begin{bmatrix} -1{,}0 \\ -0{,}01338 \\ 0{,}9811 \end{bmatrix}$$

$$C^T = \begin{bmatrix} \mathbf{c}_1^T \\ \mathbf{c}_2^T \\ \mathbf{c}_3^T \end{bmatrix} = \begin{bmatrix} 0{,}7552 & -0{,}1209 & 0{,}1813 \\ 0{,}5970 & -0{,}2267 & -0{,}03667 \\ 0{,}3887 & -0{,}5253 & -0{,}07293 \end{bmatrix}$$

$$\alpha = \begin{bmatrix} 0{,}6762 & 0{,}6705 & 0{,}6669 \\ 0{,}6640 & 0{,}6683 & 0{,}6656 \\ 0{,}6491 & 0{,}6467 & 0{,}6449 \end{bmatrix}$$

$$\beta = \begin{bmatrix} -5{,}467 & -5{,}463 & -5{,}460 \\ -1{,}934 & -1{,}940 & -1{,}945 \\ 1{,}577 & 1{,}579 & 1{,}580 \end{bmatrix}$$

$$\mathbf{g} = \begin{bmatrix} 0{,}5867 & 0{,}07218 & 1{,}323 & -1{,}109 \\ -0{,}2378 & 0{,}3609 & -0{,}2361 & 0{,}01458 \\ 0{,}05001 & 0{,}8317 & -0{,}2848 & 0{,}06962 \end{bmatrix}$$

En la Figura 4.1 se muestra el resultado en simulación, en donde se puede apreciar que ambas señales, la real de la fuente y la estimada por la red neurodifusa, son muy cercanas entre sí, como se puede ver por el valor del error cuadrático promedio logrado con los parámetros indicados anteriormente, que es de 0.0418. En la Tabla 4.1 se puede apreciar la comparación de el error RMS de la estructura propuesta ante otro tipo de modelos recurrentes existentes en la literatura. La red de neuronas con memoria considerada es aquella originalmente propuesta por Sastry [21], en donde cada neurona en cada capa tiene un elemento de memoria. En la segunda columna de la Tabla 4.1 se muestra los resultados para la red propuesta por Lee y Teng [14], que considera una única capa de neuronas con memoria. El resultado para esta última red considerada muestra un error cuadrático medio muy bajo, pero con la desventaja de requerir muchos pasos de entrenamiento, así como de un número de parámetros relativamente alto. La tercer columna muestra los resultados de la red propuesta por Juang [10], en donde una única red neurodifusa recurrente, que considera tantas variables recurrentes como reglas de inferencia difusa, calcula el valor de la salida y maneja los estados internos. En la Figura 1.2 se aprecia la estructura de esta red.

Estructura	Red de neuronas con memoria[21]	Sistema Difuso Recurrente [14]	RSONFIN [10]	Propuesta
# de parámetros	81	112	36	42
Error RMS	0.1521	0.00013	0.0248	0.0418
Pasos de entrenamiento	90000	90000	9000	1000

Tabla 4.1: Tabla comparativa de resultados

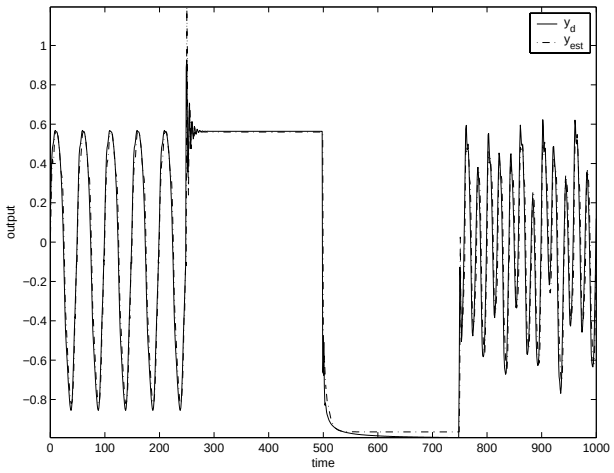


Figura 4.1: Resultado de simulación. La línea continua es la salida de la planta y la punteada la señal entregada por la red neurodifusa

4.2.1. Análisis de estabilidad

Del sistema obtenido podemos extraer las tres matrices correspondientes a cada regla:

$$C_1 = \begin{bmatrix} 0{,}755 & -0{,}121 & 0{,}181 \\ 1{,}0 & 0 & 0 \\ 0 & 1{,}0 & 0 \end{bmatrix} \tag{4.3}$$

$$C_2 = \begin{bmatrix} 0{,}597 & -0{,}227 & -0{,}0367 \\ 1{,}0 & 0 & 0 \\ 0 & 1{,}0 & 0 \end{bmatrix} \tag{4.4}$$

$$C_3 = \begin{bmatrix} 0{,}389 & -0{,}525 & -0{,}0729 \\ 1{,}0 & 0 & 0 \\ 0 & 1{,}0 & 0 \end{bmatrix} \tag{4.5}$$

(4.6)

Cuyos valores característicos son

$$\lambda_{C1} = \{\ 0{,}860, \quad -0{,}0523 - 0{,}456i, \quad -0{,}0523 + 0{,}456i\ \} \tag{4.7}$$

$$\lambda_{C2} = \{\ 0{,}357 - 0{,}428i, \quad 0{,}357 + 0{,}428i, \quad -0{,}118\ \} \tag{4.8}$$

$$\lambda_{C3} = \{\ 0{,}256 - 0{,}723i, \quad 0{,}256 + 0{,}723i, \quad -0{,}124\ \} \tag{4.9}$$

Se puede apreciar que todos los valores característicos se encuentran dentro del círculo unitario y por lo tanto todas las reglas son estables, sin que ello implique que el sistema recurrente total sea estable. Del Teorema 2.2, sabemos que debemos encontrar una matriz simétrica $P > 0$ común a todas las reglas tal que se cumpla que $C_i^T P C_i - P < 0 \; \forall \; i = 1, 2, 3$. Mediante el paquete *LMI* de *Matlab* de solución de sistemas de desigualdades de matrices, que nos permite determinar si el problema tiene solución posible o no y en caso positivo encontrarla, se encuentra que

$$P = \begin{bmatrix} 4{,}28 & -0{,}716 & 0{,}127 \\ -0{,}716 & 2{,}54 & -0{,}173 \\ 0{,}127 & -0{,}173 & 1{,}09 \end{bmatrix} > 0 \tag{4.10}$$

con valores característicos

$$\lambda_P = \{ \; 1{,}07, \quad 2{,}29, \quad 4{,}54 \; \} \tag{4.11}$$

satisface las condiciones requeridas por el teorema, por lo que el sistema resultante es local y asintóticamente estable.

4.3. Experimento: Sistema de tres tanques AMIRA DTS200

Para verificar que el algoritmo propuesto puede ser aplicado a un sistema real con propósitos de identificación, se eligió como sistema a identificar un sistema de tres tanques de líquido como el mostrado en la Figura 4.2.

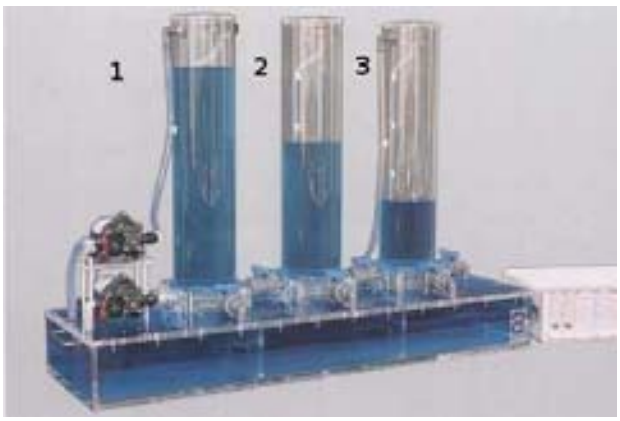


Figura 4.2: Sistema de 3 tanques de líquido AMIRA DTS200

El sistema DTS200 consta de tres tanques de líquido, que se encuentran conectados por sus bases por medio de válvulas regulables manualmente. Cada tanque cuenta con un sensor de altura de líquido y, para el caso del experimento, se consideró que solo el tanque número 1 fuera alimentado externamente con líquido. Un esquema del funcionamiento de los tanques se puede apreciar en la Figura 4.3.

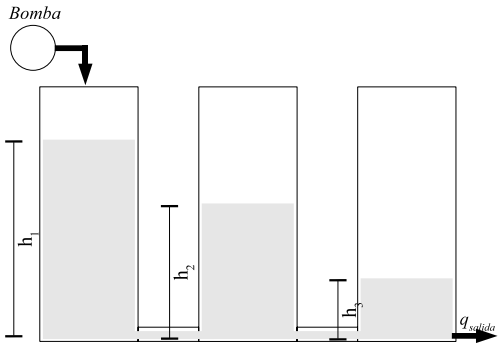


Figura 4.3: Sistema de 3 tanques de líquido AMIRA DTS200

Para realizar la identificación de la planta se tomó como salida deseada y_d a la altura del segundo tanque h_2, y como entrada al flujo de salida de la bomba. El valor máximo de altura que puede alcanzar el segundo tanque es de 60 centímetros, mientras que el flujo máximo que puede entregar la bomba es de $10^{-4}\left[\frac{m^3}{s}\right]$. El algoritmo se aplicó a las señales entrada-salida de este sistema considerando 3 estados internos. En la Figura 4.4 se muestran los resultados obtenidos tras entrenar con las señales y_d y u a la red neurodifusa recurrente propuesta. Las ganancias de adaptación que lograron mejorar la respuesta de la red sin desestabilizar el algoritmo de entrenamiento fueron $\eta_\sigma = \eta_\mu = 2 \cdot 10^{-5}$, $\eta_s = \eta_m = 0$, $\eta_c = 10^{-10}$, $\eta_\alpha = \eta_\beta = \eta_h = 2 \cdot 10^{-4}$. Los valores de los parámetros obtenidos para la red neurodifusa recurrente se enlistan a continuación:

$$\sigma = \begin{bmatrix} 2{,}339 & 2{,}571 & 2{,}114 \end{bmatrix}$$

$$\mu = \begin{bmatrix} 0{,}0005979 & 0{,}4908 & 0{,}9902 \end{bmatrix}$$

$$s = \begin{bmatrix} 4{,}866 & 5{,}342 & 4{,}532 \end{bmatrix} m = \begin{bmatrix} 4{,}866 & 5{,}342 & 4{,}532 \end{bmatrix}$$

$$C^T = \begin{bmatrix} 1{,}260 & 0{,}1584 & -0{,}4194 \\ 1{,}047 & 0{,}2335 & -0{,}2809 \\ 1{,}286 & 0{,}1884 & -0{,}4755 \end{bmatrix}$$

$$\alpha = \begin{bmatrix} 19{,}59 & 19{,}56 & 19{,}50 \\ 9{,}291 & 9{,}364 & 9{,}446 \\ 13{,}11 & 13{,}35 & 13{,}63 \end{bmatrix}$$

$$\beta = \begin{bmatrix} -0{,}02162 & -0{,}02159 & -0{,}02148 \\ 0{,}1853 & 0{,}1845 & 0{,}1837 \\ 0{,}3867 & 0{,}3866 & 0{,}3866 \end{bmatrix}$$

$$\mathbf{g} = \begin{bmatrix} 0{,}002880 & -9{,}617 & -0{,}6084 & 14{,}54 \\ 0{,}1501 & 10{,}03 & -38{,}93 & 30{,}84 \\ 0{,}6362 & 6{,}741 & -30{,}28 & 24{,}51 \end{bmatrix}$$

El error RMS obtenido para este experimento fue de 0.5964. Si se considera una normalización de la señal $y(k)$ a la unidad, el valor RMS resultante es de 0.02; lo que nos permite apreciar que el algoritmo logró obtener un conjunto de parámetros que logran tener un error relativamente bajo, lo cual se puede apreciar en la Figura 4.4, donde se ve que el error permanece dentro de una banda del 5 %.

4.3.1. Análisis de estabilidad

Como en el caso anterior, se utilizó el paquete *LMI* de Matlab para resolver el problema. Las matrices correspondientes a cada regla son:

$$C_1 = \begin{bmatrix} 1{,}26 & 0{,}158 & -0{,}419 \\ 1{,}0 & 0 & 0 \\ 0 & 1{,}0 & 0 \end{bmatrix} \tag{4.12}$$

$$C_2 = \begin{bmatrix} 1{,}05 & 0{,}234 & -0{,}281 \\ 1{,}0 & 0 & 0 \\ 0 & 1{,}0 & 0 \end{bmatrix} \tag{4.13}$$

$$C_3 = \begin{bmatrix} 1{,}29 & 0{,}188 & -0{,}476 \\ 1{,}0 & 0 & 0 \\ 0 & 1{,}0 & 0 \end{bmatrix} \tag{4.14}$$

(4.15)

cuyos eigenvalores son

$$\lambda_{C1} = \{\ -0{,}530,\ \ 0{,}997,\ \ 0{,}793\ \} \tag{4.16}$$

$$\lambda_{C2} = \{\ -0{,}507,\ \ 0{,}999,\ \ 0{,}554\ \} \tag{4.17}$$

$$\lambda_{C3} = \{\ -0{,}561,\ \ 0{,}995,\ \ 0{,}852\ \} \tag{4.18}$$

Es decir, todos son estables. Al aplicar nuevamente el paquete *LMI* de Matlab para determinar la matriz P solución, obtenemos que la matriz P común es

$$P = \begin{bmatrix} 563{,}0 & -293{,}0 & -260{,}0 \\ -293{,}0 & 269{,}0 & 21{,}8 \\ -260{,}0 & 21{,}8 & 235{,}0 \end{bmatrix} \tag{4.19}$$

cuyos valores característicos son $\lambda_P = \{\ 1{,}50 \quad 229{,}0 \quad 836{,}0\ \}$. Con esto determinamos entonces que el sistema recurrente obtenido es asintótica y locamente estable.

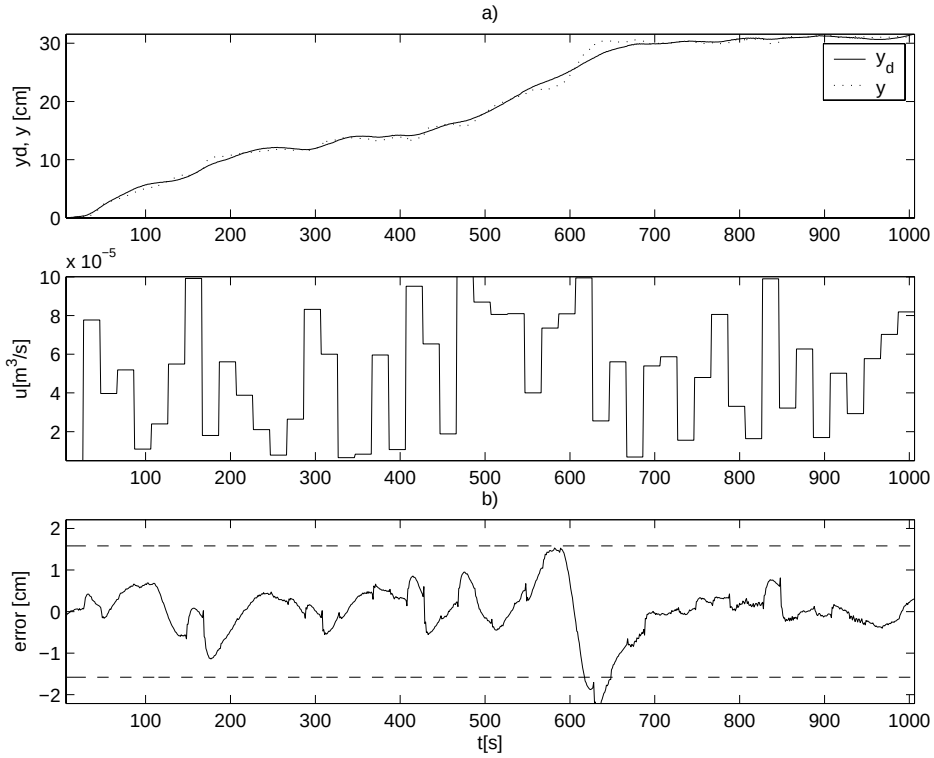


Figura 4.4: Resultados a) Salida de la planta y la red recurrente, b) Entrada a la planta y a la red recurrente c)Gráfica del error en una banda de 5 %

Capítulo 5

Conclusiones

En el presente trabajo se propone una estructura matemática basada en redes neurodifusas recurrentes para atacar el problema de la identificación de sistemas no lineales, así como un algoritmo para que, a partir de datos entrada-salida de un sistema real y un algoritmo de entrenamiento, hacer que la red propuesta logre emular, con propósitos de simulación, a un sistema dinámico no lineal. Además, se presentó un análisis de estabilidad de la red propuesta basada en la teoría de Lyapunov existente para sistemas difusos recurrentes.

La estructura propuesta busca además obtener un tipo de representación basada en redes neurodifusas en donde el número de parámetros a determinar sea reducido, y que permita también realizar un análisis del sistema obtenido con base en las herramientas de la teoría de sistemas no lineales clásica.

Dentro del trabajo se aplicó el algoritmo presentado para identificar dos tipos de sistemas: uno mediante simulaciones y otro para lograr identificar un sistema real, donde entra en juego el tiempo de muestreo así como ruido en las señales de entrenamiento. Los resultados obtenidos fueron satisfactorios y susceptibles de ser mejorados. Los análisis de estabilidad para estos sistemas fueron igualmente satisfactorios, dado que ambos sistemas resultaron ser asintóticamente estables.

Sin embargo, la red propuesta mostró una serie de limitaciones que deben de ser tomadas en cuenta:

- Únicamente se pueden identificar sistemas no lineales con un único punto de equilibrio. En vista de que la red está compuesta por sistemas lineales y dada la estructura de cada regla, el origen es el único punto de equilibrio del sistema (si la red completa es

estable en el sentido de Lyapunov)

- Los sistemas no lineales a identificar se asumen estables. El algoritmo no está diseñado para identificar sistemas no lineales inestables.

- La red es no lineal con respecto a sus parámetros, lo cual dificulta la búsqueda de óptimos locales.

Como trabajo futuro se identifican los siguientes puntos:

- *Desarrollo de esquemas de control basados en la estructura propuesta.* La estructura obtenida permite, en teoría, aplicar técnicas de control basadas en un modelo de referencia.

- *Empleo de otro tipo de estrategias de entrenamiento para la red propuesta.* El algoritmo de inicialización de parámetros, y en particular el método de entrenamiento basado en el gradiente son susceptibles de ser mejorados. El principal problema observado reside en el propio algoritmo de entrenamiento, ya que por estar basado en el gradiente presenta las limitaciones típicas de éstos, las cuales fueron discutidas en el Capítulo 2 (suposiciones relativamente fuertes con respecto a la superficie de la función de costo, posibilidad de quedar *atrapados* en un mínimo local de pobre desempeño, inestabilidad del algoritmo, sensibilidad a las ganancias de adaptación, etcétera). Entre las técnicas de aprendizaje que pueden ser empleadas se pueden considerar algoritmos basados en la Teoría de Aprendizaje (*Q-Learning*, *Reinforcement Learning* [6] [5]), Algoritmos Evolutivos (métodos genéticos, programación genética), entre otros.

- *Análisis de estabilidad del algoritmo de entrenamiento.* El algoritmo de entrenamiento mostró, durante el desarrollo del trabajo, ser susceptible a ruido y sensible a variaciones de las ganancias de adaptación. Si se busca mejorar el algoritmo de gradiente será necesario analizar sus propiedades de estabilidad o establecer un algoritmo en el cual puedan ser determinadas las cotas para las cuales el algoritmo de entrenamiento es estable.

- *Ahondar en el análisis de estabilidad.* El análisis de estabilidad estudiado en el presente trabajo cuenta con la principal desventaja de dejar fuera del análisis de estabilidad un área importante del espacio de búsqueda, dado su planteamiento. Será necesario de-

sarrollar el análisis de estabilidad de la red recurrente propuesta con el fin de encontrar condiciones menos restrictivas de estabilidad.

- *Modificar la estructura para permitir múltiples puntos de estabilidad.* La estructura propuesta no permite aún considerar sistemas no lineales que cuenten con múltiples puntos de estabilidad. Será necesario buscar formas de alterar la estructura de forma que permita esta condición, sin que por ello se complique hasta dejar de ser práctica.

- *Incorporación de un algoritmo de determinación del número de variables de estado* El algoritmo actual requiere que se asigne previamente el número de variables de estado que tendrá la red. Como trabajo futuro se propone crear un algoritmo que lo determine de forma automática.

Apéndice A

Programas realizados en Matlab

A continuación se señalan los programas realizados en Matlab utilizados en la obtención de los resultados del presente trabajo. Los programas se encuentran separados en varios módulos, todos dependientes de un programa central, tal y como se aprecia en la Figura A.1.

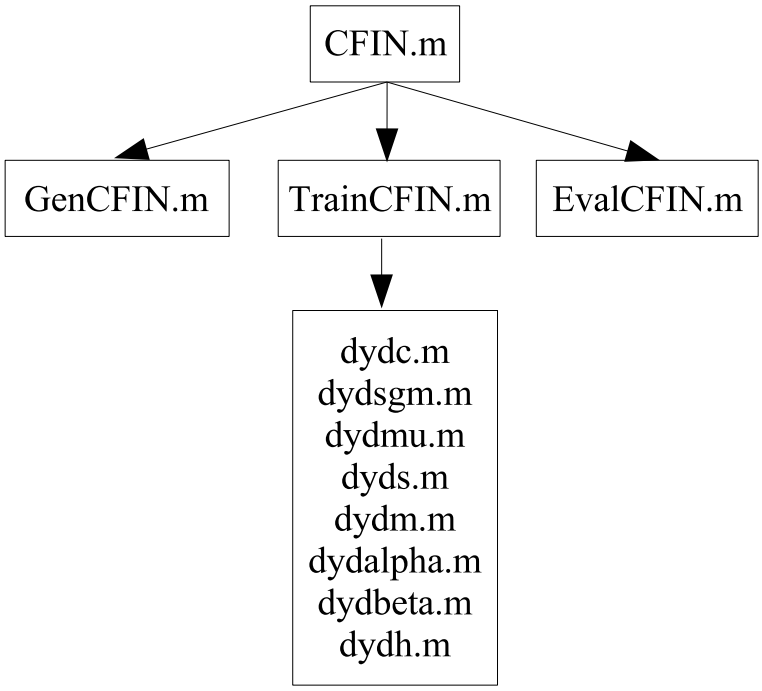


Figura A.1: Estructura del programa realizado en Matlab para la implementación del identificador

La tarea de cada programa es la siguiente:

- `CFIN.m`. Programa principal, el cual llama a las funciones `GENCFIN.m` y `TRAINCFIN.m`, que generan a su vez la estructura inicial para la red recurrente y le aplica el algoritmo de aprendizaje en tiempo real.

- `GENCFIN.m`. Genera la red neurodifusa recurrente y aplica el algoritmo para determinar la condición inicial de los parámetros.

- `TRAINCFIN.m`. Aplica el algoritmo de aprendizaje en tiempo real considerando las ganancias de adaptación definidas.

- `EvalCFIN.m`. Evalúa la salida de la red neurodifusa recurrente ante una cierta entrada u y condición inicial de los estados $\mathbf{z}(0)$ a través del tiempo.

- `EvalCFINLayer.m` y `EvalCFINLayers.m`. Determinan, para el tiempo actual, el valor de una o todas las capas de la red neurodifusa respectivamente. `EvalCFINz` evalúa, para un tiempo k, el valor de $\mathbf{z}(k+1)$.

- Las funciones `dydc.m`, `dydsgm.m`, `dydmu.m`, `dyds.m`, `dydm.m`, `dydalpha.m`, `dydbeta.m` y `dydh.m` calculan las derivadas $\frac{\partial \hat{y}}{\partial \mathbf{c}_n^T}(k)$, $\frac{\partial \hat{y}}{\partial \sigma_i}(k)$, $\frac{\partial \hat{y}}{\partial \mu_i}(k)$, $\frac{\partial \hat{y}}{\partial s_{ij}}(k)$, $\frac{\partial \hat{y}}{\partial m_{ij}}(k)$, $\frac{\partial \hat{y}}{\partial \alpha_{ij}}(k)$, $\frac{\partial \hat{y}}{\partial \beta_{ij}}(k)$ y $\frac{\partial \hat{y}}{\partial \mathbf{h}_n^T}(k)$ respectivamente.

Bibliografía

[1] S.A. Billings and C.F. Fung. Recurrent radial basis function networks for adaptive noise cancellation. *Neural Networks*, 8:273–290, April 1995.

[2] Stephen L. Chiu. Fuzzy model identification based on cluster estimation. *Journal of Intelligent and Fuzzy Systems*, 2:267–278, 1994.

[3] Kenji Doya. Recurrent networks: Learning algorithms. *Handbook of Brain Theory and Neural Networks, 2nd ed. MIT Press*, 2002.

[4] J. L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[5] M.J. Er and C. Dey. Online tuning of fuzzy inference systems using dynamic fuzzy q-learning. *IEEE Trans. on Sys., Man and Cyb.*, 34(3):1479–89, June 2004.

[6] Pierre Yves Glorennec. Fuzzy q-learning and dynamical fuzzy q-learning. *Proc. of the 3rd IEEE Conf. on Fuzzy Syst., World Congress on Comp. Int.*, 1:474–479, June 1994.

[7] V. Gorrini and H. Bersini. Recurrent fuzzy systems. *Proc. IEEE Int. Conf. on Fuzzy Systems*, 1:193–198, June 1994.

[8] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the Natural Academy of Science.*, (79):2554–2558, 1982.

[9] J.-S. R. Jang. Anfis: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man and Cybernetics*, 23:665–685, May-June 1993.

[10] Chia-Feng Juang. A tsk-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms. *IEEE Transactions of Fuzzy Systems*, 10(2):155–170, April 2002.

[11] Hassan K. Khalil. *Nonlinear Systems*. Prentice Hall, 3rd edition, 2002.

[12] Bart Kosko. *Fuzzy Engineering*. Prentice Hall, 1992.

[13] E. Kosmatopoulos, M. Polycarpou, and M. Christodoulou. High-order neural networks for identification of dynamic systems. *IEEE Trans. Neural Networks*, 6:422–431, April 1995.

[14] Ching-Hung Lee and Ching-Cheng Teng. Identification and control of dynamic systems using recurrent fuzzy neural networks. *IEEE Trans. on Fuzzy Systems*, 8(4):349–66, August 2000.

[15] Chin-Teng Lin. *Neural Fuzzy Systems: a neuro-fuzzy synergism to intelligent systems*. Prentice Hall, 1st edition, 1996.

[16] Larry R. Medsker and L.C. Jain. *Recurrent Neural Networks*. CRC Press, 2000.

[17] K.S. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Tras. Neural Networks*, 1:4–27, March 1990.

[18] Oliver Nelles. *Nonlinear System Identification*. Springer, Berlin, 2001.

[19] Kevin M. Passino. *Fuzzy Control*. Addison Wesley Longman, Menlo, CA, 1998.

[20] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. *Learning Internal Representations by Error Propagation*, chapter 8. MIT Press, Cambridge, 1986.

[21] P.S. Sastry, G. Santharam, and K.P. Innikrishnan. Memory neuron networks for identification and control of dynamic systems. *IEEE Trans. Neural Networks*, 5(2):306–319, March 1994.

[22] L.E. Scales. *Introduction to Non-Linear Optimization*. Computer and Science series. McMillan, 1985.

[23] T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Trans. Systems, Man and Cybernetics*, SMC-15:116–132, January 1985.

[24] L.-X. Wang. *Adaptive Fuzzy Systems and Control*. Prentice Hall, 1994.

[25] Paul J. Werbos. Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, Octubre 1990.

[26] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 2(1):270–280, 1989.

[27] Lofti Zadeh. Fuzzy sets. *Information and Control*, (8):338–353, 1965.

[28] J. Zhang and A. J. Morris. Recurrent neuro-fuzzy systems for nonlinear process modeling. *IEEE Trans. on Neural Networks*, 10(2):313–326, March 1999.

Índice alfabético