Universitat Autònoma
de Barcelona

REDES NEURONALES MULTIMODELO APLICADAS AL CONTROL DE SISTEMAS

Memòria del projecte
d'Enginyeria Tècnica en
Informàtica de Sistemes
realitzat per
Sergio Torrubia Caravaca
i dirigit per
Asier Ibeas Hernández

Escola d'Enginyeria
Sabadell, Setembre de 2010

[El/La] sotasignat, ***Asier Ibeas Hernández***,
professor[/a] de l'Escola d'Enginyeria de la UAB,

CERTIFICA:

Que el treball al que correspon la present memòria ha estat realitzat sota la seva direcció per en ***Sergio Torrubia Caravaca.*** I per a que consti firma la present.
Sabadell, ***Setembre*** de ***2010***

Signat: ***Asier Ibeas Hernández***

Agradecimientos

Quiero dedicar un agradecimiento a las personas que han contribuido a que este proyecto se haya podido completar. A Asier Ibeas, director de este proyecto, por su ayuda, consejos y tiempo dedicado. A Cristina, por su paciencia y por ayudarme a superar la procrastinación. Y a todas las personas que han contribuido anónimamente para que hoy presente este trabajo.

Índice general

1. Introducción

En este primer apartado se describe el objeto del proyecto, así como la metodología utilizada y un resumen de lo que se verá en cada apartado en la estructura de la memoria.

1.1. Objeto del proyecto

La ingeniería de control es una disciplina que se focaliza en modelizar matemáticamente una gama diversa de sistemas dinámicos y el diseño de controladores que harán que estos sistemas se comporten de la manera deseada. En este aspecto, el correcto diseño del controlador es de suma importancia para el buen funcionamiento del sistema de control, ya que un buen sistema de control debe ser robusto (con una alta tolerancia a fallos) y eficiente. Para realizar esta tarea es necesario conocer la estructura del sistema a controlar, ya que el controlador debe ser la inversa de este sistema para la obtención de los resultados esperados. En cambio, en casos donde existen perturbaciones que alteran el sistema puede ser muy complicado conocer cual será la salida.

Las redes neuronales artificiales presentan una serie de ventajas, entre las que se incluyen su capacidad de aprendizaje y una gran tolerancia a fallos. Estas características las hacen muy adecuadas en la implementación de un controlador. Además de que en el aprendizaje de una red neuronal sólo nos es necesario conocer la entrada y salida del sistema, con lo que no es necesario conocer la estructura completa del sistema y pueden ser implementadas en sistemas con perturbaciones. Sin embargo, la fase de aprendizaje de la red neuronal puede ser a veces algo lenta, y en ese periodo de aprendizaje el sistema de control no ofrece unos resultados satisfactorios.

En este proyecto se plantea un nuevo modelo de red neuronal llamado multimodelo. Si en un modelo convencional de red neuronal obtenemos la

salida a partir de una serie de parámetros, en un sistema multimodelo recalculamos la salida a partir de los mismos valores de entrada, pero variando los parámetros. Con esto se consigue una mayor muestra de salidas, que permiten decidir cuales son los parámetros más acertados (los que devuelven una salida más cercana a la salida deseada del sistema de control).

1.2. Descripción de la metodología utilizada

Para llevar a cabo este proyecto se ha hecho uso de Matlab 7.3 (R2006b). Este programa integra un entorno de desarrollo con un lenguaje de programación propio (lenguaje M), con el que se han implementado todas las funciones del proyecto.

Entre sus prestaciones básicas se hallan:

- Manipulación de matrices.
- Representación de datos y funciones.
- Implementación de algoritmos.
- Creación de interfaces de usuario (GUI)

Matlab es un programa de cálculo numérico orientado a matrices. Por tanto, será más eficiente si se diseñan los algoritmos en términos de matrices y vectores.

1.3. Estructura de la memoria

En este estudio se presenta un sistema multimodelo basado en la multiplicación de salidas de una red neuronal artificial con el fin de mejorar la velocidad de aprendizaje de la misma, aplicándola luego a un sistema de control adaptativo.

Se indican a continuación los contenidos de los capitulos que siguen, en los que se describe el trabajo realizado.

– En el capitulo 2 se estudia la viabilidad del proyecto. Se describen los objetivos del estudio, separandolos en tres objetivos principales. Se valoran los riesgos, tareas y costes del proyecto para finalmente concluir la viabilidad técnica del mismo.

– En el capitulo 3 se presenta conceptos básicos sobre las redes neuronales biológicas y artificiales, así como una introducción a los tipos de aprendizaje utilizados en el desarrollo del proyecto.

– En el capitulo 4 se introducen brevemente los sistemas de control de lazo abierto y lazo cerrado.

– En el capitulo 5 se unen los conceptos de los dos apartados anteriores para hablar sobre la aplicación de las redes neuronales artificiales al control de sistemas. También se muestran las desventajas que supone esta adaptación y como el sistema multimodelo presentado en este proyecto puede paliar estas desventajas.

– El capitulo 6 profundiza en la estructura de control del modelo utilizado. Se argumenta y desarrolla el uso de un sistema de control en lazo cerrado y porque se utiliza un control adaptativo basado en redes neuronales artificiales. También se muestra la implementación de las redes neuronales en el control adaptativo, así como la implementación y métodos utilizados por el sistema multimodelo. Finalmente se explica y muestra la importancia del tiempo de residencia y de como este influye en la estabilidad del sistema multimodelo.

– En el capitulo 7 se muestra la implementación del proyecto a partir del programa Matlab. Se definen los requesitos del sistema, casos de uso, así como las variables y funciones más importantes.

– En el capitulo 8 se pueden comprobar los resultados finales obtenidos de la simulación de las funciones implementadas. Se muestran los gráficos de las simulaciones de redes neuronales artificiales, así como los resultados obtenidos en la simulación de un sistema de control adaptativo implementado con redes neuronales artificiales y con redes neuronales artificiales multimodelo.

– En el capitulo 9 se presentan las conclusiones generales del proyecto.

– En el capitulo 10 se presenta la bibliografía usada en el trabajo.

2. Estudio de viabilidad

Este proyecto esta basado en el diseño y desarrollo de una aplicación creada con Matlab R2006b con la que simulamos el funcionamiento de un sistema de control gracias a las capacidades de las redes neuronales artificiales (RNA).

La aplicación consta inicialmente de una serie de funciones que nos permite trabajar con distintos tipos de redes neuronales, donde podemos parametrizar su topología, entradas, funciones de activación, salida objetivo, pesos, tipo de aprendizaje, etc...

A continuación se implementan distintos tipos de sistemas de control en los que tanto el controlador como el estimador consisten en sistemas de RNA.

Por último mejoramos la velocidad de aprendizaje de las RNA del controlador y el estimador recalculando las salidas de las RNA para distintas parametrizaciones aleatorias, por ejemplo, tomando distintos valores de pesos, y utilizando una lógica que conmuta de forma que intenta elegir la mejor salida.

2.1. Objetivos del estudio

2.1.1. Implementar en primera persona una herramienta que permita simular diferentes tipos de red neuronal

Matlab R2006b incluye una “Toolbox” (librería) llamada “Neural Network Toolbox” que contiene decenas de funciones destinadas a la creación y calculo de las RNA. A pesar de disponer de estas herramientas se ha optado por obviarlas, de forma que sea necesario implementarlas desde su forma más básica y así conocerlas en mayor profundidad, ya que al inicio del proyecto no se tenía ningún conocimiento, tan siquiera fundamental, acerca de las RNA. Por lo tanto el primer objetivo marcado en el proyecto es adquirir conocimientos sólidos sobre las RNA que después podamos utilizar en el desarrollo del resto de la aplicación. Además, muchas de las funciones que se utilizan para el

diseño de sistemas de control basados en RNA son funciones especificas que permiten garantizar posteriormente la estabilidad del lazo cerrado, por lo que aunque la “toolbox” de Matlab contenga muchas funciones, en algunos casos se hace necesario implementarlas por cuenta propia, por lo que optar por un desarrollo propio se convierte en una alternativa aceptable del uso de la RNA.

Para conseguir este objetivo ha sido necesaria una labor de recopilación de información acerca de las redes neuronales biológicas y artificiales para adquirir un conocimiento general de las mismas.

También ha sido necesario conocer y entender los algoritmos que se utilizan para computar los resultados de las RNA. Se han tenido que adaptar estos algoritmos a funciones y operaciones que trabajaran con matrices con el objetivo de simplificar cálculos y mejorar el rendimiento del sistema, ya que el entorno Matlab es mucho más eficiente operando con matrices, de hecho Matlab esta diseñado para ello.

2.1.2. Estudio del posible uso de las RNA aplicadas al control de sistemas

El segundo objetivo del proyecto ha sido aplicar los conocimientos adquiridos en la primera fase para implementar un controlador y un estimador neuronal en distintos tipos de sistemas de control. Como ya se explicó en el apartado 1.2 las características y cualidades de las que disponen las RNA son muy apropiadas para la incorporación de las mismas en sistemas de control en lazo cerrado.

Primeramente se ha diseñado y desarrollado una función que simulara un sistema de control discreto lineal con un controlador y un estimador que funcionan sin RNA. De esta forma comprobamos que el diseño del sistema de control original es correcto antes de introducir las RNA.

A continuación se han sustituido tanto el controlador como el estimador común por un controlador neuronal y un estimador neuronal utilizando las herramientas desarrolladas en la primera parte del proyecto.

Por último se han utilizado distintos tipos de sistemas de control (lineal y no lineal), distintos tipos de entrenamientos para las RNA incluidas en el controlador y en el estimador (Adaline o Widrow-Hoff y Backpropagation), así como diferentes lógicas de calculo de las redes, como por ejemplo de tipo on-line (las RNA's utilizan la salida de la red para realimentarse mientras aprenden) u off-line (las RNA's no conocen la salida del sistema y por lo tanto su aprendizaje es más lento). Utilizando y combinando estos sistemas se ha comprobado la respuesta del sistema procurando encontrar la más fiable y rápida para el aprendizaje del sistema.

2.1.3. Estudiar el efecto del sistema multimodelo en la convergencia y utilidad en el control de las RNA

Como se explicará de forma más detallada en el punto 5 la utilización de RNA aplicadas en el sistema de control puede ocasionar problemas a la hora de hacer que la señal obtenida por la planta y la señal deseada (objetivo) converjan. Puede ser un problema de velocidad de convergencia o en el peor caso un problema de divergencia entre las dos señales, con lo que los resultados pueden ser nefastos. Por lo tanto, el objetivo final es paliar este problema.

Para conseguir este objetivo se pretende multiplicar el número de salidas de la red utilizando diferentes pesos para cada una de las salidas y de esta forma acelerar la adaptación de la red a la salida objetivo, es decir, mejorar la velocidad de aprendizaje de la red.

Se ha diseñado y desarrollado un sistema que es capaz de seleccionar la mejor de las salidas multiplicadas. Para considerar cual es la mejor salida se

pueden seleccionar diferentes criterios basados en el análisis de los errores que produce cada salida, como por ejemplo, el error más bajo en cada iteración o la media de error más baja tomando un determinado número de iteraciones de muestra. De esta forma podremos utilizar distintos criterios y utilizar el más adecuado para cada situación.

2.2.¿Es viable técnicamente?

El proyecto es viable técnicamente ya que existe documentación donde se utilizan las RNA para el control de sistemas de control (ver Anexo 1) y también se ha demostrado en otros casos sencillos que es posible mejorar la convergencia del aprendizaje de las RNA utilizando sistemas multimodelo [5].

2.3.Material necesario

Para el desarrollo del proyecto han sido necesarios:

Material humano

–Programador: necesario para el desarrollo del software simulador, así como para la búsqueda de información necesaria para tal desarrollo.

Material físico

–OpenOffice 3.2

–OpenProj 1.4

–Matlab 7.3 (R2006b)

Requisitos mínimos:

- –Procesador tipo AMD o Intel Pentium III o superior.
- –Sistema operativo Windows XP (Service Pack 1, 2 o 3), Windows 2000 (Service pack 2 o 3) o Windows Server 2003.
- –510MB de espació libre en disco.
- –512MB de RAM (se recomiendan 1024MB)
- –Teclado

–Ratón

–Monitor

–DVD/CR-ROM

–Tarjeta Ethernet 10/100

2.4. Tareas

A continuación se muestra una tabla con un desglose de las distintas tareas desarrolladas en el transcurso del proyecto, así como la duración de cada una.

	Nombre	Duración	Inicio	Terminado	Prec
1	**Proyecto**	123 days	**1/09/09 8:00**	**18/02/10 20:30**	
2	Reunión con el tutor	0,25 days	1/09/09 8:00	1/09/09 20:30	
3	Análisi de requerimientos	2 days	2/09/09 8:00	5/09/09 20:00	2
4	**Estudio de viabilidad**	13 days	**8/09/09 8:00**	**24/09/09 20:30**	**3**
5	Planificación	1 day	8/09/09 8:00	11/09/09 20:30	
6	Analisis de software capaz de implementar el sistema	1 day	14/09/09 8:00	17/09/09 20:30	5
7	Analisis de costes	1 day	18/09/09 8:00	19/09/09 16:00	6
8	Analisis de riesgos	1 day	21/09/09 8:00	24/09/09 20:30	7
9	**Imp. de la herram. de sim. RNA**	26 days	**25/09/09 8:00**	**31/10/09 16:00**	**4**
10	Busqueda bibliográfica	4 days	25/09/09 8:00	2/10/09 20:30	
11	Diseño	3 days	5/10/09 8:00	11/10/09 13:00	10
12	Desarrollo	6 days	12/10/09 8:00	24/10/09 18:00	11
13	Pruebas locales y corrección de errores	2 days	26/10/09 8:00	31/10/09 16:00	12
14	**Imp. herram. de sim. de sist. control**	19 days	**3/11/09 8:00**	**28/11/09 18:00**	**9**
15	Busqueda bibliográfica	2 days	3/11/09 8:00	7/11/09 18:00	
16	Diseño	3 days	9/11/09 8:00	15/11/09 13:00	15
17	Desarrollo	4 days	16/11/09 8:00	23/11/09 20:30	16
18	Pruebas locales y corrección de errores	2 days	24/11/09 8:00	28/11/09 18:00	17
19	**Imp. herram. sim. RNA aplicadas a sist. control**	34 days	**1/12/09 8:00**	**16/01/10 16:00**	**14**
20	Busqueda bibliográfica	2 days	1/12/09 8:00	5/12/09 18:00	
21	Diseño	6 days	7/12/09 8:00	19/12/09 18:00	20
22	Desarrollo	10 days	21/12/09 8:00	9/01/10 20:00	21
23	Pruebas locales y corrección de errores	2 days	11/01/10 8:00	16/01/10 16:00	22
24	Documentación	16 days	18/01/10 8:00	18/02/10 20:30	19

Ilustración 2.1: Lista de tareas con su respectiva duración

2.5. Planificación

El diagrama de Gantt nos muestra la planificación de cada tarea. La primera columna del diagrama indica a que tarea pertenece.

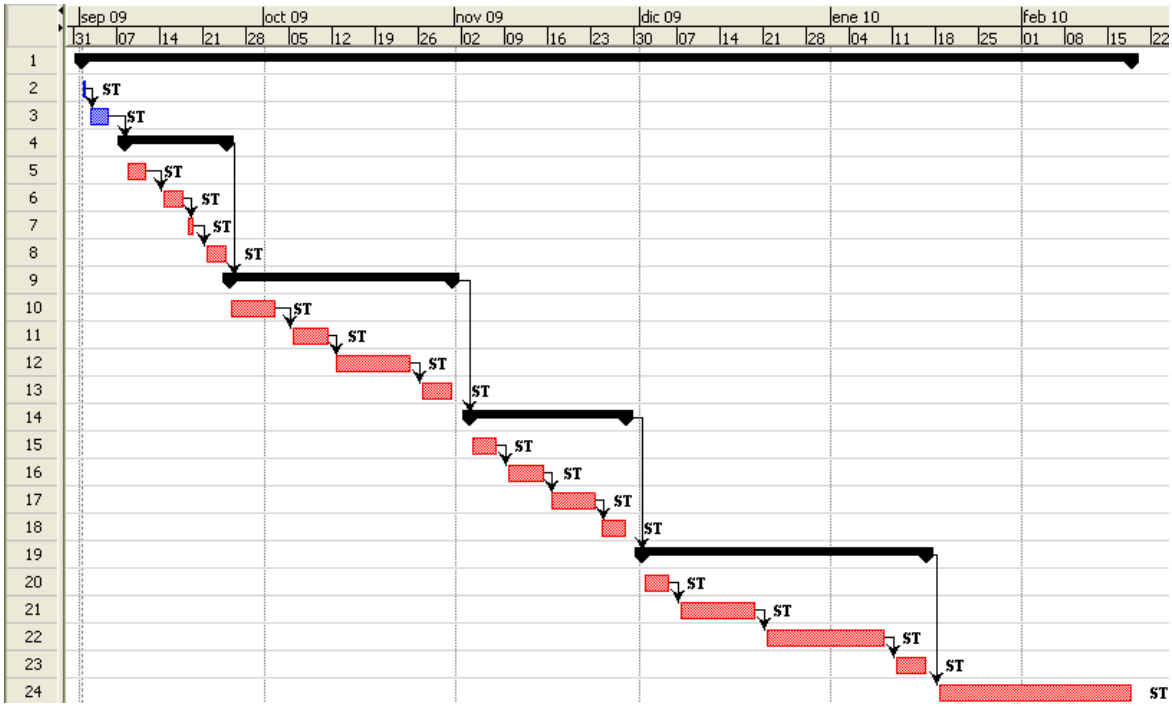

Ilustración 2.2: Diagrama de Gantt

2.6. Coste del proyecto

Gracias a las siguientes tablas podemos realizar un cálculo del coste del proyecto, teniendo en cuenta el coste de los recursos materiales y humanos.

Recurso	Coste
Matlab 7.3 (R2006b)	65 €
OpenOffice 3.2	0 €
OpenProj 1.4	0 €
TOTAL	65 €

Tareas	Duración (horas)
Análisis de requerimientos	16
Estudio de viabilidad	32
Imp. herr. sim. RNA	120
Imp. herr. sim. sist. control	88
Imp. herr. sim. RNA aplicadas a sist. control	160
Documentación	128
TOTAL	544

–Coste del programador: 7€/hora.

Coste total del proyecto (Recursos+Tareas)	**3.873 €**

2.7. Evaluación de riesgos

Riesgo

Avería del PC de desarrollo

–El PC de desarrollo es susceptible de sufrir averías de hardware (daños en disco duro, roturas por golpes o caídas, problemas en la placa base, etc.) con lo que perdería toda su funcionalidad.

–También son posibles averías de software (fallos en el SO, infección del equipo por virus, problemas en la instalación de MATLAB, etc.) que podrían provocar ralentización en el desarrollo del proyecto o como en el caso anterior perdida total de la funcionalidad del equipo.

Plan de contingencia

–Disponer de un segundo equipo de desarrollo minimizará la perdida de tiempo hasta que el equipo principal este reparado.

–Para minimizar perdidas de datos se efectuarán copias de todos los datos al

finalizar cada jornada de trabajo en un dispositivo de almacenamiento externo (p.e. Memoria Flash) o subiendo los datos en formato comprimido a un servidor externo en Internet (p.e. GMail).

Riesgo

–No se dispone de vehículo para acudir al lugar de trabajo.

Plan de contingencia

–Se utilizará transporte público.

–Al utilizarse un PC portátil como equipo de desarrollo existe la posibilidad de continuar el trabajo en el hogar.

2.8. Conclusiones

Finalmente, valorando todos los puntos estudiados en el estudio desarrollado en este capitulo, podemos concluir que la totalidad del proyecto es viable tanto a nivel técnico, como a nivel de recursos materiales y humanos.

3. Redes neuronales

En este apartado se hace una breve introducción a las redes neuronales, tanto biológicas como artificiales. También se verá un ejemplo de cálculo de una RNA para comprender mejor su funcionamiento.

3.1. Introducción

Ya en 1911, con los primeros estudios de Santiago Ramón y Cajal, las neuronas se definieron como el componente básico del sistema nervioso, incluyendo al cerebro, el cual esta formado por un enorme número de ellas. Así mismo, existen numerosos estudios que muestran que estas neuronas están conectadas de forma masiva entre sí, y aunque esta cualidad está presente en todos los animales es mucho más acentuada y evidente en los seres humanos, característica que nos permite ser inteligentes y nos discrimina del resto de seres vivos.

Y aunque aún en la actualidad el cerebro es quizás el órgano que más se desconoce en la anatomía humana, el estudio de la inteligencia ha fascinado a filósofos y científicos, siendo un tema recurrente en tratados y libros.
Una de las capacidades más relevantes de nuestro cerebro es la capacidad que tiene para aprender gracias a la interacción con el medio ambiente. Por este motivo la representación de las redes neuronales biológicas por medio de un modelo formal resulta de sumo interés para diversas disciplinas: neurociencia, matemáticas, estadística, física, ciencias de la computación, ingeniería, entre otras.

De este interés surgen las redes neuronales artificiales, las cuales tratan de extraer las excelentes capacidades del cerebro para resolver algunos de los problemas más complejos de las disciplinas antes mencionadas, como: visión, reconocimiento de patrones, control moto-sensorial o procesamiento de señales.

Es precisamente en el problema del procesamiento de señales el que se tratara en este proyecto, ya que este se basa en la simulación de sistemas de control mediante la utilización de redes neuronales.

Como se ha dicho anteriormente, el proyecto consiste en la simulación de un sistema de control utilizando las capacidades de aprendizaje de las que disponen las redes neuronales artificiales y su capacidad para aproximar arbitrariamente bien cualquier función continua.

A continuación se explicará qué es una red neuronal, qué es un sistema de control y la relación que se ha establecido entre los dos en el programa.

3.2. Redes neuronales biológicas

Para entender el funcionamiento del programa es necesario entender el funcionamiento de las redes neuronales artificiales, y para entender estas es necesario también entender las redes neuronales biológicas.

El cerebro esta compuesto de células nerviosas llamadas neuronas. Estas neuronas tienen ramas diminutas que se extienden y se conectan con otras neuronas para formar una red neuronal. El cerebro construye todos sus conceptos por la memoria asociativa, lo que significa nuevas conexiones entre neuronas. Por ejemplo, las ideas, los pensamientos y los sentimientos están todos construidos e interconectados en esta red neuronal y gracias a esta interconexión el cerebro es capaz de resolver problemas de pensamiento no analíticos tales como la percepción, el razonamiento, la memoria asociativa, la decisión, la creatividad, la planificación de tareas, el control, y sobre todo, el lenguaje natural y el aprendizaje.

Estas redes neuronales están compuestas por neuronas de entrada (sensores), conectados a una red compleja de neuronas “calculadoras”

conectadas a las neuronas de salida, las cuales se pueden encargar, por ejemplo de controlar la activación de los músculos.

Fisiológicamente, las neuronas que se disparan juntas se conectan. Si practicas algo una y otra vez esas células nerviosas tienen una relación a largo plazo. Si por el contrario dejas de practicar una actividad durante un largo tiempo, estas células empiezan a interrumpir su relación a largo plazo. Esto es lo que conocemos como aprendizaje.

Se estima que en un sólo milímetro cúbico de nuestro cerebro hay unas 40.000 neuronas (más de cien mil millones en todo nuestro cerebro) y 1.000 conexiones de fibras nerviosas.

La misión de estas neuronas comprende generalmente cinco funciones parciales:

–Recogen información que llega a ellas en forma de impulsos procedentes de otras neuronas o receptores.

–La integran en un código de activación propio de la célula.

–La transmiten codificada en forma de frecuencia de impulsos a través de su axón.

–A través de sus ramificaciones el axón efectúa la distribución espacial de los mensajes.

–En sus terminales transmite los impulsos a las neuronas subsiguientes o a las células efectoras.

La Ilustración 3.1 muestra un diagrama de una célula nerviosa típica.

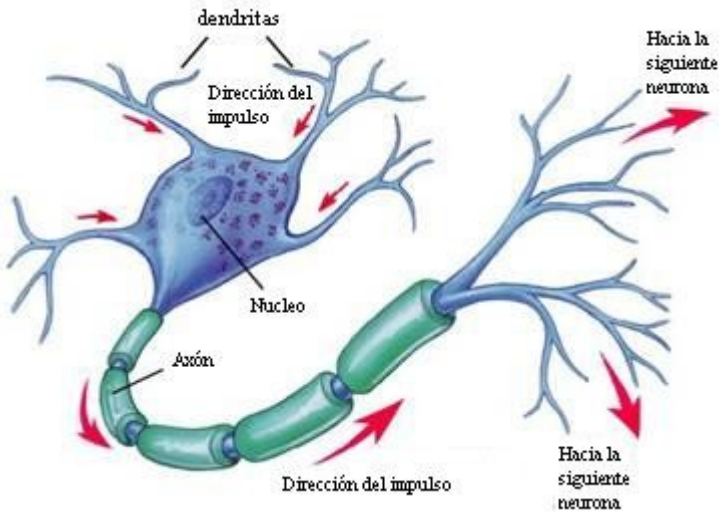

Ilustración 3.1: Neurona biológica

En este esquema se aprecia que, al igual que el resto de las células del organismo, la neurona consta de un núcleo. Además del núcleo la neurona cuenta también con algunos elementos específicos. En primer lugar está el axón, que es una ramificación de salida de la neurona. A través de él se propagan una serie de impulsos electro-químicos. La neurona también cuenta con un gran número de ramificaciones de entrada, llamadas dendritas, que se encargan de propagar la señal al interior de la neurona.

La propagación de estas señales funciona de la siguiente forma. Las dendritas recogen información electro-química procedente de las células vecinas a las que la célula en cuestión está conectada; esta información llega al núcleo donde se procesa hasta generar una respuesta que es propagada por el axón. Más tarde, esta señal se ramifica y llega a dendritas de otras células a través de lo que se denomina sinapsis. Las sinapsis son los elementos de unión entre axón y dendritas. Consiste en un espacio líquido donde existen determinadas concentraciones de elementos ionizados que hacen que el espacio intersináptico posea ciertas propiedades de conductividad que activen o impidan en mayor o menor grado el paso del impulso eléctrico. Por lo tanto las sinapsis funcionan como potenciadores o inhibidores de la señal procedente de

los axones. La concentración iónica de estas sinapsis es modificada sucesivamente dentro de la enorme malla de neuronas. Esta concentración iónica es muy importante, ya que las neuronas no son elementos lineales, funcionan por saturación y sólo producen una señal de activación si la señal recibida supera un cierto umbral, en caso contrario permanecen inhibidas sin transmitir información a otras neuronas.

3.3. Redes neuronales artificiales

Las redes neuronales artificiales (RNA) intentan extraer las grandes capacidades que tiene nuestro cerebro para resolver problemas complejos que de otra forma serían muy difíciles de resolver.

Las RNA ofrecen algunas ventajas como:

–No linealidad: gracias a la sinapsis el procesador neuronal biológico no es lineal, el procesador neuronal artificial también utiliza la sinapsis y sólo se activa si la señal supera el umbral de activación, lo que también lo convierte en no lineal. Si la neurona es no lineal la red neuronal también lo es.

–Transformación entrada-salida: El proceso de aprendizaje de la red consiste en mostrarle un ejemplo y modificar sus pesos sinápticos de acuerdo con su respuesta. De esta forma la red tiene la capacidad de aprender y adaptarse a la transformación entrada-salida.

–Adaptabilidad: Una red neuronal es capaz de adaptar sus parámetros incluso en tiempo real.

–Tolerancia a fallas: Al existir una conectividad masiva la falla de un procesador no altera seriamente la operación.

–Uniformidad en el análisis y diseño, lo que permite garantizar características precisas.

–Analogía con las redes biológicas, lo que permite utilizar conocimientos de ambas áreas, de forma que podemos conocer mejor el funcionamiento de las redes neuronales biológicas y mejorar los sistemas de redes neuronales artificiales.

Una red neuronal artificial esta formada por una capa de entrada, un determinado número de capas ocultas (que también puede ser cero) y una capa de salida. Entre cada una de las capas que forman la red neuronal existen matrices de pesos sinápticos, que son los que le otorgan a la red neuronal su potencial de cálculo y capacidad de aprendizaje, ya que son estos los que varían a medida que el sistema aprende.

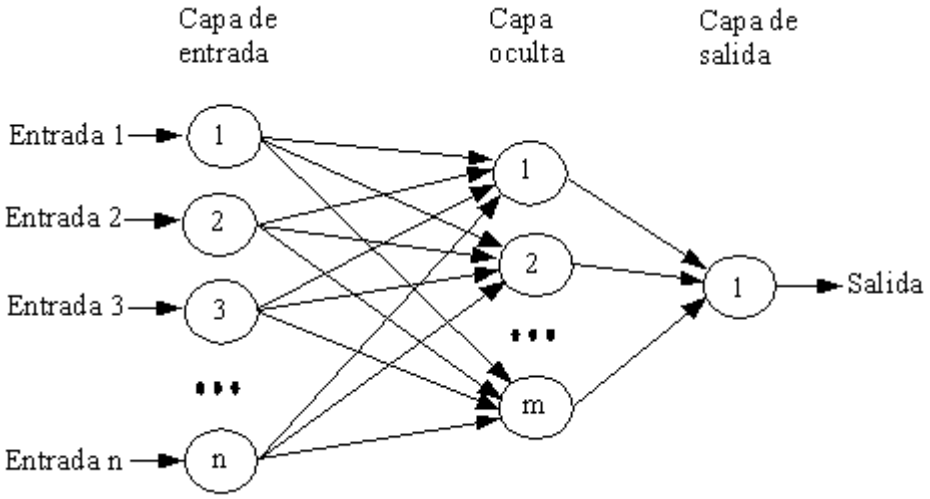

Ilustración 3.2: Modelo de una red neuronal artificial.

Como vemos en la Ilustración 3.2 cada una de las capas esta formada por nodos interconectados entre si de forma masiva (cada una de las neuronas de una capa esta conectada con todas las demás neuronas de las capas adyacentes), cada uno de estos nodos son neuronas artificiales y son el elemento básico de una red neuronal.

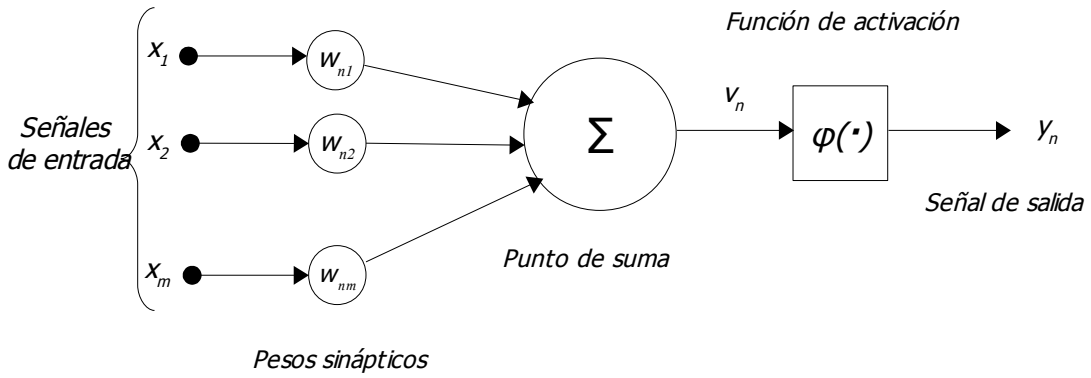

Ilustración 3.3: Modelo de una neurona artificial.

En el modelo de neurona artificial mostrado en la Ilustración 3.3 se pueden identificar tres elementos:

–Enlaces de conexión: Parametrizados por los pesos sinápticos w_{ij}. El primer subíndice se refiere a la neurona receptora y el segundo a la neurona emisora. Si w_{ij} es positivo se trata de una conexión excitadora, si es negativo entonces es inhibidora. En una neurona biológica la dendrita y el axón funcionan como la neurona receptora y emisora respectivamente en una red neuronal artificial.

–Sumador (Σ): Suma cada uno de los componentes de las señales de entrada multiplicadas por w_{ij}. El sumador realiza la misma función de procesamiento de datos que el núcleo de una neurona biológica.

–Función de activación (φ): Se trata de una función de trasformación generalmente no lineal. Este elemento es equivalente a la sinapsis en la neurona biológica, la cual otorgaba a la red neuronal biológica la capacidad de cálculo no lineal (si la salida es no-lineal).

En términos matemáticos podemos describir la relación entrada-salida de la neurona *i* de la Ilustración 3.3 por la siguiente ecuación:

$$y_i = \varphi\left(\sum_{j=1}^{m} w_{ij} x_j\right) \qquad (3.1)$$

Con el siguiente ejemplo se podrá interpretar con mayor claridad la ecuación (3.1).

Tomemos como muestra la neurona número 5 de la red neuronal artificial representada en la Ilustración 3.4.

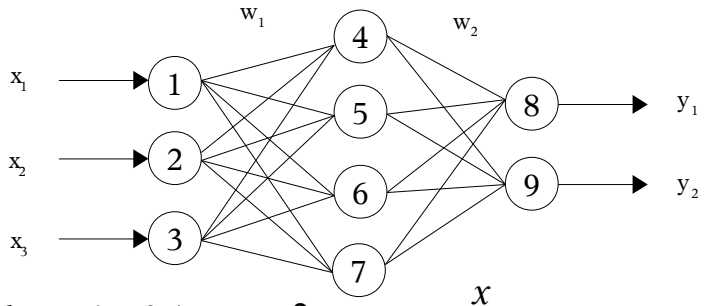

Ilustración 3.4 $\varphi_1 = 2x$ $\varphi_2 = \frac{x}{4}$

Donde:

$$W_1 = \begin{pmatrix} 3 & 2 & 3 & 4 \\ 1 & 2 & 5 & 2 \\ 2 & 1 & 6 & 1 \end{pmatrix} \qquad (3.2)$$

y

$$W_2 = \begin{pmatrix} 2 & 3 \\ 1 & 2 \\ 1 & 1 \\ 3 & 1 \end{pmatrix} \qquad (3.3)$$

Estas matrices se obtienen a partir de w_{ij} para cada conexión entre el nodo i de la capa anterior y el nodo j de la capa posterior, por lo que si tenemos que en la matriz W_1 el valor de w_{23} = 5 quiere decir que el peso sináptico entre el segundo nodo de la capa anterior (número 2 en la Ilustración 3.4) y el tercer

nodo en la capa posterior (número 6 en la Ilustración 3.4) es igual a 5.

Se comprueba que el número de filas de las matrices de pesos corresponde con el número de neuronas de la capa anterior y el número de columnas corresponde con el número de neuronas de la capa posterior.

Una vez presentada la red neuronal de ejemplo calculamos el vector de salida a partir del vector de entrada X = (-3, 1, 4) utilizando la ecuación (3.1).

$$\vec{B} = \vec{X} \cdot W_1 = \begin{pmatrix} -3 & 1 & 4 \end{pmatrix} \cdot \begin{pmatrix} 3 & 2 & 3 & 4 \\ 1 & 2 & 5 & 2 \\ 2 & 1 & 6 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 20 & -6 \end{pmatrix} \qquad (3.4)$$

$$\vec{Y} = \varphi(\varphi(\vec{B}) \cdot W_2) = \begin{pmatrix} 0 & 0 & 40 & -12 \end{pmatrix} \cdot \begin{pmatrix} 2 & 3 \\ 1 & 2 \\ 1 & 1 \\ 3 & 1 \end{pmatrix} = \varphi(4 \quad 28) = \begin{pmatrix} 1 & 7 \end{pmatrix} \qquad (3.5)$$

Como vemos en la fórmula *(3.5)* las funciones de activación (φ) se calculan elemento a elemento cuando se aplican a vectores.

Si bien en el ejemplo se han utilizado funciones de activación lineales para simplificar el proceso hemos de saber que todas las redes neuronales cuentan con funciones de activación no lineales, ya que de esta forma el potencial de la red para solucionar problemas de forma genérica es mucho más elevado. Cualquier red de más de una capa que cuente con funciones de activación lineales se puede simplificar en una red equivalente de una sola capa.

Las funciones de activación (φ) definen la salida de la neurona en función del

potencial de activación. Para conseguirlo se establece el valor de la salida, x_j, a partir del potencial postsináptico calculado por la función de combinación, net_j. Existen muchas funciones diferentes pero estas son las más usuales:

–lineal: no se realiza ningún tipo de operación sobre el potencial, salvo una ponderación por un valor constante ŋ.

$$x_i = \eta \cdot net_i \tag{3.6}$$

Esta es la función utilizada en el ejemplo anterior.

–paso:

$$x_i = \begin{cases} \beta \; si \; net_i > \theta \\ \Upsilon \; si \; net_i = \theta \\ \delta \; si \; net_i < \theta \end{cases} \tag{3.7}$$

Donde β, γ, δ y θ son constantes, normalmente iguales para todas las neuronas de la red neuronal.

–sigmoide:

$$x_j = \left(1 + \exp\left(-\frac{net_i}{T} \right)^{-1} \right) \tag{3.8}$$

Donde T es un parámetro que controla la pendiente de la función en el origen: si $T \rightarrow \infty$ la función sigmoide tiende a la función escalón. Es la función más utilizada ya que el rango de salida es continuo, aunque acotado, y es derivable.

A veces se utiliza como variante de esta función la tangente hiperbólica:

$$x_i = \tanh(net_i) = 2\,sigmoide(net_i) - 1 = \frac{1 - e^{-netj/T}}{1 + e^{-netj/T}} \quad (3.9)$$

En este caso, cuando T → ∞, la función tangente hiperbólica tiende a la función signo.

Un aproximación que se suele utilizar en vez de la función sigmoidal, por ser más fácil de calcular, derivable, con derivada continua, y monótonamente creciente, es

–rampa:

$$x_j \begin{cases} \dfrac{net_j^2}{net_j^2 + 1} & si\ net_j > 0 \\ 0 & en\ otro\ caso \end{cases} \quad (3.10)$$

$$x_j = \begin{cases} \Upsilon\ si\ net_j \geq \Upsilon \\ net_i\ si\ \delta < net_j < \Upsilon \\ \delta\ si\ net_j \leq \delta \end{cases} \quad (3.11)$$

–gaussiana: utilizada con la función de combinación media-varianza en las redes neuronales de funciones de base radial, con $x_j \in [0,1]$:

$$x_j = \exp\left(\frac{-net_j^2}{2}\right) \quad (3.12)$$

3.3.1. Aprendizaje de las redes neuronales artificiales

Se puede decir que las características más importantes de las redes neuronales son:

–El aprendizaje por interacción con el medio ambiente.

–El mejoramiento de su desempeño por medio de este aprendizaje.

Una red neuronal aprende variando sus parámetros, en particular los pesos sinápticos (en nuestro ejemplo mostrado en la Ilustración 3 estos pesos son w_1 y w_2). El aprendizaje es un proceso por el cual estos parámetros se adaptan por la interacción continua con el medio ambiente. El tipo de aprendizaje está determinado por la forma en que se realiza dicha adaptación. Este proceso implica la siguiente secuencia de eventos:

–La red neuronal es estimulada por el medio ambiente.

–La red neuronal ajusta sus parámetros.

–La red neuronal genera una nueva respuesta.

La fórmula general que determina el mecanismo de ajuste de los pesos en la red neuronal es la siguiente:

$$w_{ij}(k+1)=w_{ij}\,k+\Delta\, w_{ij}(k) \qquad (3.13)$$

donde k es la iteración actual y $\Delta w_{ij}(k)$ es el aumento o disminución de los pesos. Podríamos definir esta fórmula de la siguiente forma: los pesos sinápticos de la próxima iteración es igual a los pesos sinápticos de la iteración actual más una corrección aditiva. Esta corrección viene determinada por el calculo del aprendizaje. Cuanto mayor sea el valor absoluto de $\Delta w_{ij}(k)$ más variará el peso sináptico de la neurona net_{ij}.

Los algoritmos de aprendizaje son el conjunto de reglas para resolver el

problema de aprendizaje y difieren en cómo determinar $\Delta w_{ij}(k)$.

Veamos dos de los algoritmos de aprendizaje que se utilizan en el programa desarrollado para este proyecto.

3.3.1.1. Regla delta o Widrow-Hoff

Tipica de las redes Adaline y Madaline. Se trata de minimizar el error definido como la distancia euclídea entre el vector de salida de la red Y y el vector de salida deseado A. Este error se calcula como

$$E = |A - Y|^2 = 1/2 \sum_i (a_i - y_i)^2 \tag{3.14}$$

En este tipo de algoritmo de aprendizaje sólo hay una capa con pesos modificables, lo que significa que sólo puede aprender funciones linealmente separables. Según esto último tampoco podemos utilizar este tipo de aprendizaje para redes neuronales de más de una capa.

3.3.1.2. Algoritmo backpropagation (perceptrón multicapa)

El algoritmo backpropagation es un algoritmo iterativo que permite entrenar redes multicapa. En una red multicapa podemos definir el error total como la suma de los cuadrados de los errores cometidos, en este tipo de algoritmo se pretende minimizar dicho error. Para conseguirlo se debe:

–Calcular el error en la salida para cada patrón utilizando la fórmula 3.14.

–Ajustar los pesos de la capa de salida para reducir el error.

–Propagar los errores hacia la capa de entrada, ajustando los pesos de las capas ocultas.

Este proceso se repite de forma iterativa para todos los valores de entrada.

4. Sistemas de control

Los sistemas de control son aquellos dedicados a obtener la salida deseada de un sistema o proceso. En un sistema general se tienen una serie de entradas que provienen del sistema a controlar, llamado planta, y se diseña un sistema para que, a partir de estas entradas, modifique ciertos parámetros en el sistema planta, con lo que las señales anteriores volverán a su estado normal ante cualquier variación.

Existen diferentes tipos de sistemas de control, pero todos deben cumplir los siguientes requisitos:

1. Deben ser estables y robustos frente a perturbaciones y errores en los modelos.

2. Ser eficientes según un criterio preestablecido evitando comportamientos bruscos e irreales.

3. Ser fácilmente implementables en un sistema computerizado.

4.1. Sistema de control en lazo abierto

El objetivo de un sistema de control es poder obtener una o más salidas a través de una o más entradas. Para conseguir esto se incluye en el sistema un controlador que sea el inverso de la función de transferencia del sistema, consiguiendo así que la función de transferencia de todo el sistema sea igual a 1. Esta idea se corresponde con un sistema de control en lazo abierto. Algunos de los inconvenientes son:

–Nunca se conoce perfectamente la planta, por lo que no se puede conseguir el inverso perfecto.

–No se puede implementar en sistema inestables.

–No compensa perturbaciones en el sistema.

Ilustración 4.1: Esquema de sistema de control en lazo abierto.

4.2. Sistema de control en lazo cerrado

Para eliminar algunos de los problemas que dan los sistema de control en lazo abierto se puede utilizar la realimentación, dando lugar a un sistema de control en lazo cerrado. Los sistema de control en lazo cerrado se definen como aquellos en los que existe una realimentación de la señal de salida, con lo que la señal de salida tiene efecto sobre la acción de control.

El encargado de realizar este ajuste en el sistema de control en lazo cerrado es el controlador, el cual ajusta la acción de control basándose en la salida del sistema. Para este controlador la salida del sistema general es su propia entrada, y la salida del controlador se convierte en la acción de control que envía al sistema general. Esto es lo que se conoce como retroalimentación, ya que gracias al controlador la salida del sistema influye activamente en la entrada.

Algunas de las ventajas que tienen los sistemas de control en lazo cerrado son:

–Pueden controlar sistemas inestables.

–Pueden compensar perturbaciones en el sistema.

–Pueden controlar sistemas que contengan errores en el modelado.

Estas ventajas hacen que los sistemas de control en lazo cerrado sean los más implementados en la practica, a pesar de que son más complejos y suponen un mayor coste económico (ya que por ejemplo deben utilizarse sensores para compensar las perturbaciones del sistema y medir la salida de la planta). Utilizar la realimentación también implica un problema de ruido en el sistema al hacer la medición del mismo.

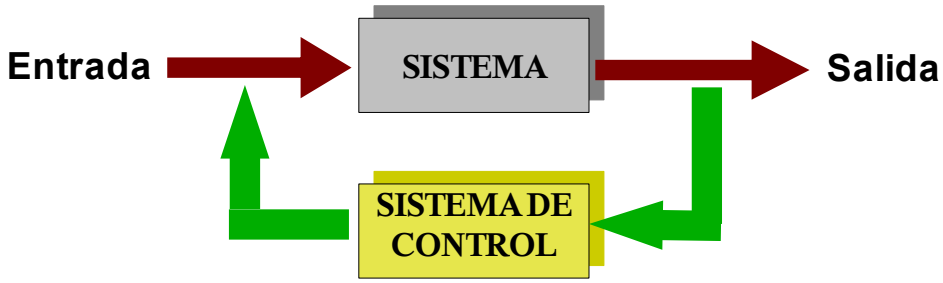

Ilustración 4.2: Esquema de sistema de control en lazo cerrado

5. Aplicación de las redes neuronales al control

Hemos visto que en los sistema de control en lazo cerrado el controlador juega un papel muy importante en el ajuste de la planta. Este controlador ajusta la acción de control en función de la salida. Por ejemplo, en un sistema de control de crucero de un vehículo el controlador será el encargado de administrar más o menos gasolina a consumir por el vehículo dependiendo de la velocidad del mismo. O en un sistema climatizador se ajustará la salida de aire frío o caliente en función de la temperatura actual.

Normalmente para diseñar un controlador es necesario conocer la función de transferencia de la planta. Esta función de transferencia de la planta puede ser totalmente desconocida, con lo que sólo podríamos conocer la entrada y salida del sistema.

Las redes neuronales tienen la capacidad de aprender, por lo que si conocemos la entrada del sistema, la salida objetivo (consigna) y la salida real del sistema ante una entrada dada, un controlador integrado por una red neuronal sería capaz de modificar sus parámetros aprendiendo la dinámica de la planta hasta conseguir un sistema fiable. A efectos prácticos, en el estado estacionario, los resultados obtenidos por la red neuronal serían muy similares a la función de transferencia desconocida del controlador. Se convertiría por tanto en un sistema de control variable en el tiempo ya que la red iría cambiando su estructura con el paso del tiempo y aprendiendo la dinámica del sistema, lo que recibe el nombre de un controlador adaptable. Los controladores adaptables se utilizan para diseñar sistemas de control para plantas que son desconocidas a priori y configuran controladores variables en el tiempo.

5.1. Desventajas

Al inicializar los parámetros de una red neuronal no podemos saber cuales van a ser los más adecuados para que el sistema minimice el error de la forma más rápida posible, por eso se inicializan estos parámetros de forma aleatoria. Es posible por tanto, que los parámetros elegidos aleatoriamente converjan muy lentamente. Durante el intervalo en el que aprende se esta controlando muy mal el sistema y por tanto la salida es mala.

Adicionálmente, la velocidad de convergencia de la red depende de la RNA utilizada y en los casos en los que se utilizan RNA estándares pueden surgir velocidades de convergencia lentas, lo que hace que sea un problema en la práctica.

5.2. Sistema multimodelo

Para solucionar el problema de velocidad de convergencia se propone un sistema capaz de multiplicar el número de salidas de la red neuronal para que esta aprenda de forma paralela, y que utilizando un sistema de conmutación se escoja siempre la mejor salida. Para conseguirlo se debe calcular la salida para tantos grupos independientes de parámetros aleatorios como factor de multiplicación se utilice en el sistema. Por ejemplo, imaginemos un sistema compuesto por una red neuronal con una sola capa oculta, y por lo tanto, con dos matrices de pesos; y con 3 neuronas en la capa de salida. Si quisiéramos multiplicar por 3 las salidas deberíamos manejar un total de 6 matrices de pesos y 9 salidas.

Multiplicando la capa de salida e inicializando los parámetros de salida de los pesos de forma aleatoria podemos compensar una desviación inicial del error más rápidamente. Para aprovechar esta característica de forma correcta se utiliza una lógica que elige en cada intervalo de tiempo el conjunto de parámetros con el error entre la salida y la salida deseada más pequeño, por lo

que existe una conmutación entre las salidas adicionales. Esto no quiere decir que en todos los sistemas multimodelo se conmute entre todas las salidas, si en un sistema las salidas multiplicadas adicionales devuelven errores mayores a la salida original de la red no existirá ninguna conmutación.

En términos de calculo de un sistema multimodelo estaremos multiplicando las operaciones prácticamente como tantas salidas estemos multiplicando, por lo que el sistema necesitará un mayor número de operaciones. Dado que se procesa en paralelo se hacen más cálculos pero no requieren más tiempo. Por otro lado estamos ganando una velocidad de aprendizaje mucho mayor, consiguiendo que el sistema se adapte mucho más rápido a la salida que deseamos obtener.

6. Estructura de control

6.1. Ventajas del sistema de control en lazo cerrado

Ya vimos en el apartado 4 las ventajas que tenía utilizar un sistema de control en lazo cerrado (realimentado) frente a un sistema de control en lazo abierto.
Los sistemas realimentados tienen varias propiedades importantes ya que el comportamiento global del sistema viene determinado por las propiedades de la interacción de cada una de las partes. La realimentación permite estabilizar sistemas inestables, hace que los sistemas sean más robustos ante variaciones en algunas partes del sistema, y también permite atenuar perturbaciones externas que no puedan ser cuantificadas.

Para deducir el principal beneficio de la realimentación debemos considerar las incertidumbres en el comportamiento de la planta. Existen dos razones por las que la salida de la planta *y*, para una entrada *u*, genera una trayectoria que no está completamente determinada previamente. En primer lugar, la dinámica de la planta es desconocida, por lo que sólo podemos hacer una consideración aproximada del modelo real. En segundo lugar, la presencia de perturbaciones desconocidas pueden alterar el comportamiento del sistema, por lo que la salida *y* no es sólo la respuesta a la señal de control *u*, sino también a la perturbación *d*, la cual es generalmente indeterminada.

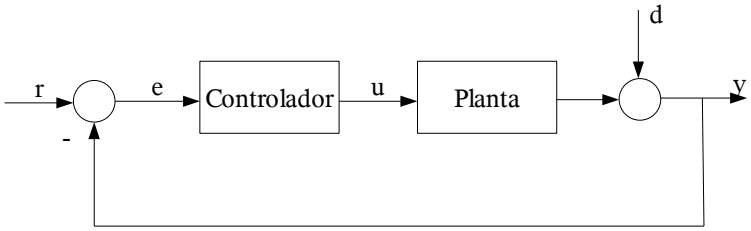

Ilustración 6.1: Sistema de control en lazo cerrado con perturbación

Generalmente este tipo de indeterminaciones están presentes en mayor o menor medida en todos los problemas de control, porque a causa del proceso

de modelado y diseño, el sistema a ser controlado debe considerarse fuera de su entorno. Usualmente no es fácil saber qué fenómenos tienen que considerarse como parte de la planta o como conexiones entre la planta y su entorno y cuales no. Teniendo esto en cuenta debemos saber que las principales propiedades de los sistemas realimentados dependen solo débilmente de tales indeterminaciones. Como la señal de control *u* es calculada por el controlador dependiendo del valor actual de *y*, hay que tener en cuenta el efecto de las señales de control y de las perturbaciones.

Esta propiedad fundamental de los sistemas realimentados se hace evidente por la comparación de las configuraciones de los sistemas de control en lazo abierto y lazo cerrado. En el caso del lazo abierto el control está basado en la exactitud del modelo de la planta, ya que la señal de control se calcula en base a este modelo. En cambio, en el control en lazo cerrado, se compara el valor actual de la salida *y* con la señal de referencia *r*, y la entrada a la planta *u* se calcula en base a la señal de error *e*. Además de la información que la planta nos da en forma de modelo matemático, el control realimentado utiliza el conocimiento acerca del comportamiento actual de la planta y de las perturbaciones existentes, todo esto suministrado de forma implícita por el valor de *y*.

Estas ventajas hacen que los sistemas de control en lazo cerrado sean los más adecuados para nuestro sistema de control adaptado con redes neuronales, ya que la utilización de estas no previene que el sistema pueda contener errores en el modelado, provoque perturbaciones o haga el sistema inestable. De forma más estricta, el uso de la realimentación puede no estar justificado si no hay indeterminaciones en el sistema, ya que en el control en lazo abierto un sistema sin perturbaciones produce la misma o probablemente mejor respuesta.

6.2. Sensibilidad en lazo abierto y en lazo cerrado

Sea un sistema cuya función de transferencia en lazo abierto es G(s). Si la función de transferencia en lazo cerrado es T(s) se trendrá que en el dominio de frecuencias de interés, es decir, para $\omega \in [0, \Omega]$, siendo *Ω* la frecuencia angular que define el ancho de banda para el que se proyecta el sistema, se tendrá que,

$$T(s) \approx 1 \qquad (6.1)$$

es decir, se pretende que en el anterior dominio de frecuencias la reproducción de las señales sea lo más fiel posible.

Se sabe que la relación que liga a *G(s)* y a *T(s)* es

$$T(s) = \frac{G(s)}{1 + G(s)} \qquad (6.2)$$

Por lo tanto, puesto que T ≈ 1 para todas las frecuencias de interés, se tendrá que

$$G \gg 1 \; para \; \omega \in [0, \Omega] \qquad (6.3)$$

Se define la sensibilidad de un sistema, dado por su función de transferencia *T(s)*, como la variación que sufre esta función de transferencia como consecuencia de la variación de uno de los parámetros *p* que interviene en la misma. Se denota por S_p^T la sensibilidad de la función de transferencia *T(s)* con respecto a variaciones del parámetro *p*. Formalmente se escribe:

$$S_p^T = \frac{\%\ variación\ de\ T(s)}{\%\ variación\ de\ p} \qquad (6.4)$$

lo que a su vez puede escribirse como

$$S_p^T = \frac{dT/T}{dp/p} = \frac{dT}{dp} \cdot \frac{p}{T} \qquad (6.5)$$

Supóngase, que uno de los parámetros que aparecen en *G(s)* sufre una variación (por envejecimiento, efecto de perturbaciones externas, etc.). La sensibilidad de la función de transferencia en lazo cerrado *T(s)* a la variación de un parámetro de *G(s)* será:

$$S_G^T = \frac{1}{(1+G)^2} \cdot \frac{G}{T} = \frac{1}{1+G} \qquad (6.6)$$

lo que habida cuenta de la expresión 6.3 conduce a $S_G^T \ll 1$ es decir, un sistema en bucle cerrado tiene una sensibilidad a las variaciones de los elementos que constituyen la cadena en lazo abierto mucho menor que 1.

Este resultado debe compararse con la sensibilidad de un sistema en lazo abierto $G = G_1 G_2$, que puede escribirse,

$$S_{G_1}^G = 1 \qquad (6.7)$$

Se concluye de lo anterior que la introducción de la realimentación reduce notablemente la sensibilidad del sistema con respecto a la variación de los parámetros. Es éste un resultado fundamental que justifica sobradamente la introducción de la realimentación en el diseño de sistemas de control.

6.3. Análisis y diseño en presencia de incertidumbres

La mayoría de los métodos de análisis y diseño para sistemas realimentados presuponen que se dispone de un modelo suficientemente exacto del proceso a controlar o del sistema en lazo cerrado, respectivamente. En este modelo

está fijado, tanto la estructura del modelo como los parámetros. Aunque algunos de los métodos hacen algunas consideraciones sobre el efecto de las posibles incertidumbres, el tratamiento y atenuación de las perturbaciones no son su principal razón de ser.

Sin embargo, un enfoque más realista para el control por realimentación tiene que tener en cuenta las imprecisiones del modelo, lo cual es la principal razón para la utilización de la realimentación. Otra forma de abordar el problema es tratar con principios y métodos que nos permitan considerar explícitamente las discrepancias entre el modelo y el proceso real. Este es, intentar analizar y diseñar el controlador sin tener un modelo matemático preciso de la planta. Intentar obtener resultados que no solo sean válidos para el modelo aproximado, sino que lo sean para un rango de modelos de la planta dados y en consecuencia para el proceso real.

La conveniencia de investigaciones teóricas sobre el análisis y diseño de sistemas robustos pone de manifiesto que la robustez es una cuestión fundamental en los sistemas realimentados. Sin embargo, sin un profundo conocimiento de los beneficios de la realimentación, el campo de aplicaciones de los controladores lineales invariantes en el tiempo, para el caso de desconocimiento del proceso, variable con el tiempo o no lineal, debe quedar mucho más limitado. La realimentación debe estar relacionada con la clase y cantidad de incertidumbre del modelo y con las propiedades que serán preservadas.

Desde el punto de vista de la ingeniería, las investigaciones teóricas sobre la robustez de los sistemas realimentados, deben estar encaminadas a:

– determinar las capacidades fundamentales de la realimentación lineal para controlar las plantas con incertidumbres, y

– elaborar procedimientos de análisis y diseño prácticos.

La primera tarea concierne a preguntas como: ¿Qué incertidumbre del modelo puede ser tolerada por el controlador lineal? ¿Qué estructura debe tener el controlador? ¿Bajo qué condiciones debe ser diseñado el controlador para que asegure la estabilidad en el caso de fallo del sensor o actuador? La segunda tarea es claro que va hacia la explotación sistemática de la robustez.

6.4. Posibles planteamientos

Principalmente, para el diseño de controladores a partir de modelos de la planta que son lineales e invariantes con el tiempo, hay dos posibles enfoques para contemplar las incertidumbre en el modelo del sistema y las perturbaciones sobre este. El primero de ellos es considerar las incertidumbres del sistema en el diseño de un controlador fijo, lo cuál lleva a un esquema de control robusto, que es más insensible a las variaciones en los parámetros y a las perturbaciones.

El segundo enfoque es utilizar un controlador adaptativo, el cuál estima los parámetros y calcula la señal de control basándose en dichos parámetros. La metodología de los reguladores autoajustables es la adecuada en este caso, pero conlleva el diseño en línea, con el esfuerzo de cálculo necesario para ello, mucho mayor que en el caso de un regulador más simple.

El uso del controlador adaptativo también hace que sea necesario conocer la estructura del sistema, saber si se trata de un sistema lineal o no lineal y conocer el orden del sistema en concreto.

Sin embargo, con la ayuda de las redes neuronales artificiales podemos paliar estos inconvenientes, ya que gracias a estas conseguiremos prescindir de métodos de calculo complejos y costosos, minimizando así el tiempo de calculo, además de poder obviar la estructura del sistema, ya que las RNA son

capaces de adaptarse a cualquier tipo de estructura sin necesidad de conocer sus características.

Además, las RNA poseen una propiedad fundamental a la hora de aproximar funciones llamada *propiedad de aproximación universal*.

Para que se cumpla esta propiedad es imprescindible que la RNA disponga de al menos dos capas. En algunos trabajos (ver, por ejemplo, Hornik, K [3]) se ha demostrado que una red bicapa con capa oculta sigmoide y capa de salida lineal es capaz de aproximar con precisión arbitraria cualquier función no lineal suave con un numero finito de discontinuidades.

6.5. Control adaptativo

El término adaptativo significa cambiar el comportamiento conforme a nuevas circunstancias. Un regulador adaptativo es un regulador que puede modificar su comportamiento en respuesta a cambios en la dinámica del sistema y a las perturbaciones. Este mismo objetivo es el de la inclusión de la realimentación en el bucle de control, por lo que surge la pregunta de cuál es la diferencia entre control realimentado y control adaptativo.

Existen muchas definiciones de control adaptativo, siendo una de las más aceptadas, que control adaptativo es un tipo especial de control no lineal en el que el estado del proceso puede ser separado en dos escalas de tiempo que evolucionan a diferente velocidad. La escala lenta corresponde a los cambios de los parámetros y por consiguiente a la velocidad con la cual los parámetros del regulador son modificados, y la escala rápida que corresponde a la dinámica del bucle ordinario de realimentación.

El esquema básico de control adaptativo, según puede verse en la Ilustración 6.2, está compuesto de un bucle principal de realimentación negativa, en el que actúa al igual que en los sistemas convencionales un regulador y de otro bucle

en el que se mide un cierto índice de funcionamiento, el cual es comparado con el índice deseado y se procesa el error en un mecanismo de adaptación que ajusta los parámetros del regulador y en algunos casos actúa directamente sobre la señal de control.

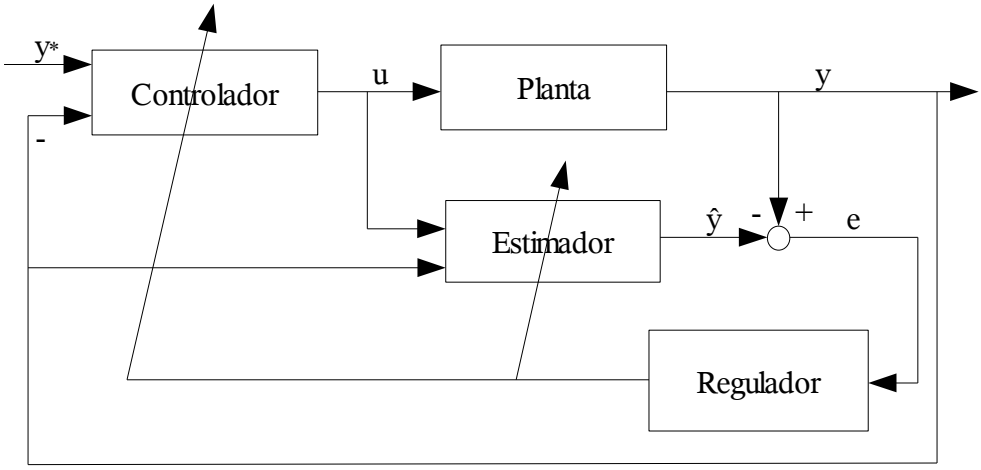

Ilustración 6.2. Sistema adaptable de control.

El control adaptativo actúa como un control secundario (lazo secundario) influenciando a los parámetros del control primario.

El lazo primario debe ser de respuesta más rápida que el lazo secundario, en caso contrario el sistema tendería a saturarse porque se estaría obligando al sistema a reaccionar más rápidamente de lo que puede. Por tanto, el lazo secundario debe variar más lentamente que el lazo primario.

6.6. Adaptación de las RNA a un sistema de control adaptativo

Como ya vimos en el punto 6.4. el uso de las redes neuronales artificiales a la hora de implementar un sistema de control esta más que justificado. Además, en las últimas dos décadas ha existido una gran actividad de investigación acerca de las aplicaciones de las redes neuronales para el diseño de sistemas de control, incluyendo problemas de control adaptativo (vease, por ejemplo, S. Leuseur [6]).

Dos enfoques básicos han sido referenciados en la documentación de las redes neuronales para abordar el problema del control adaptativo. En el primer caso algunos parámetros del diseño son aprendidos “off-line”, midiendo las señales de entrada-salida y observando el comportamiento de la planta en momentos clave. En el segundo enfoque el aprendizaje adaptativo se implementa y la entrada de control se determina “on-line” como la salida de la red neuronal, la señal de control que genera la red es la que genera la salida del sistema que se usa para adaptar los parámetros.

Tomaremos como muestra el segundo enfoque mencionado arriba. De forma más especifica se usa una red neuronal de dos capas como modelo de la planta desconocida. Esta red neuronal aprende “on-line” la dinámica del sistema, que en este ejemplo utilizará la regla Widrow-Hoff delta, la cual minimiza la diferencia entre la señal de salida actual de la planta y la salida pronosticada por la red. Este aprendizaje es usado como controlador. El objetivo del control es conseguir que la salida de la red *y* se convierta en la señal de referencia *y**.

6.6.1. Modelo de la planta

Con objeto de mostrar una primera aproximación al problema y metodología planteada, se considera un primer caso lineal que permite extenderlo al caso no-lineal.

Tratamos con una planta lineal que cumple la siguiente igualdad:

$$A(q^{-1})\,y(k) = B(q^{-1})\,u(k) \qquad (6.8)$$

Donde $A(q^{-1})$ y $B(q^{-1})$ son polinomios definidos como:

$$A(q^{-1}) = 1 + a_1 q^{-1} + \ldots + a_{n'} q^{-n'} \qquad (6.9)$$
$$B(q^{-1}) = b_1 q^{-1} + \ldots + b_{m'} q^{-m'}$$

siendo q^{-1} el operador de desplazamiento hacia atrás (p.e.: $q^{-1}y(k)=y(k-1)$), y u(k), y(k) la entrada y salida de la planta respectivamente.

Debemos asumir estas premisas para la planta:

A1: Los limites superiores (n y m, respectivamente) de los ordenes n', m' son conocidos.

A2: $B(q^{-1})$ es un polinomio estable.

A3: El coeficiente $b_1 \neq 0$.

6.6.2. Modelo neuronal de la planta

Usaremos una red neuronal de dos capas como modelo de la planta. La red aprende la dinámica de la planta usando la regla Widrow-Hoff. La capa de entrada esta compuesta de n+m elementos. Estas n+m entradas son las señales de entrada-salida calculadas en instantes previos en la planta. La capa de salida tiene sólo un elemento, y su salida es la salida pronosticada de la planta. Esta red pretende aprender la dinámica de la planta, por lo tanto su salida es una salida estimada del verdadero sistema.

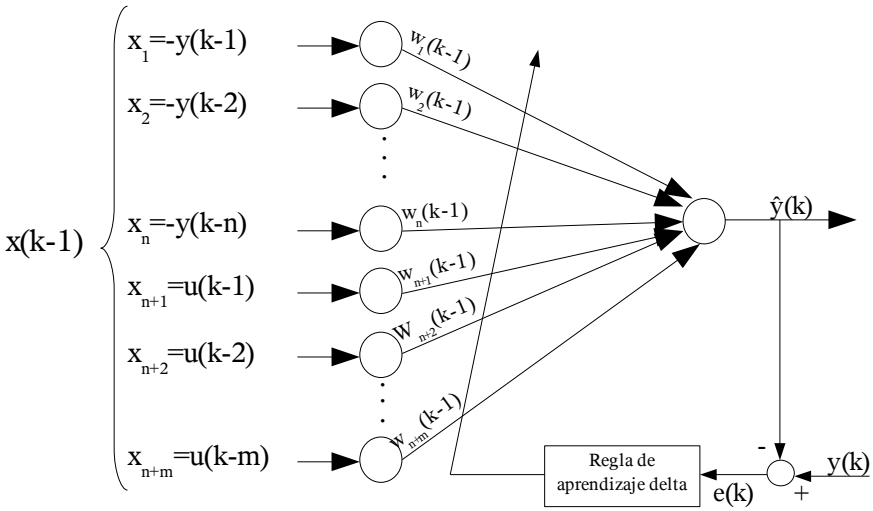

Ilustración 6.5: Modelo neuronal de la planta

La Ilustración 6.5. muestra el estimador de la planta neuronal. El vector de pesos en el instante k, definido por:

$$w(k)=[w_1(k),\ldots,w_n(k),w_{n+1}(k),\ldots,w_{n+m}(k)]^T \qquad (6.10)$$

se actualiza usando la regla de Widrow-Hoff:

$$w(k+1)=w(k)+\frac{\alpha\, e(k+1)x(k)}{\varepsilon+x^T(k)x(k)} \qquad (6.11)$$

donde x(k) es el vector de entrada de la red, el cual se define por:

$$\begin{aligned} x(k)&=[x_1(k),\ldots,x_n(k),x_{n+1}(k),\ldots,x_{n+m}(k)]^T \\ x(k)&=[-y(k),\ldots,-y(k-n+1),u(k),\ldots,u(k-m+1)]^T \end{aligned} \qquad (6.12)$$

y α perteneciente al intervalo (0,2) es el factor de reducción. La constante ε se define cercana a cero y sólo se incluye para evitar la división por cero en (6.11) si $x^T(k)\cdot x(k) = 0$. Como se muestra en la Ilustración 6.5, la señal de error e(k) es la diferencia entre la respuesta real de la planta y(k) y la salida $\hat{y}(k)$ pronosticada por el modelo neuronal. Esta salida pronosticada se calcula linealmente como sigue:

$$\hat{y}(k)=\sum_{i=1}^{n+m} w_i(k-1)x_i(k-1) \qquad (6.13)$$

y el algoritmo de adaptación (6.11) minimiza el error.

Desde un punto de vista matemático, la regla de adaptación de pesos (6.11) puede ser considerada como un caso particular de un algoritmo de estimación de parámetros adaptativo general con una convergencia bien definida y unas buenas propiedades de estabilidad. Esta analogía debería permitirnos aplicar las técnicas de control adaptativo para analizar las propiedades de algunos tipos de redes neuronales.

6.6.3. Controlador neuronal

El aprendizaje de la dinámica de la planta gracias al estimador neuronal mostrado arriba se usa para ajustar los pesos de conexión de un controlador neuronal que genera la señal de control u(k). Esta señal de control, cuando se aplica a la entrada de la planta, pretende conseguir que la salida de la planta *y(k)* sea la señal objetivo *y*(k)*. El esquema de control en lazo cerrado se muestra en la Ilustración 3.6.

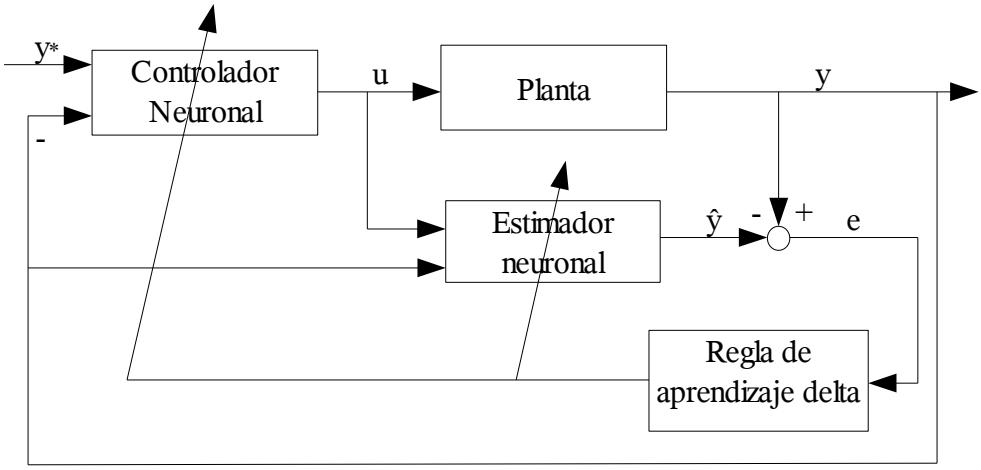

Ilustración 6.6: Esquema de control neuronal adaptativo

El controlador neuronal esta compuesto de una segunda red neuronal de dos capas con n+m elementos en la capa de entrada y un elemento en la capa de salida. El vector de entrada en el instante k z(k) se define por:

$$z(k)=[y*(k+1),-x_1(k),-x_2(k),\ldots,-x_{n+2}(k),\ldots,-x_{n+m}(k)]^T \quad (6.14)$$

y el vector de pesos de conexión w'(k) se define como la función de pesos ajustable w_i, i=1,...,n+m, (ver ecuación 6.10), del modelo neuronal de la planta como sigue:

$$w'(k)=\frac{1}{w_{n+1}(k)}[1, w_1(k), w_2(k), \ldots, w_n(k), w_{n+2}(k), \ldots, w_{n+m}(k)]^T \quad (6.15)$$

La Ilustración 6.7. muestra el controlador neuronal. La señal de control se genera como una combinación lineal de los elementos del vector de entrada z(k), usando los nuevos pesos $w'_i(k)$:

$$u(k)=\sum_{i=1}^{n+m} w'_i(k) z_i(k) \quad (6.16)$$

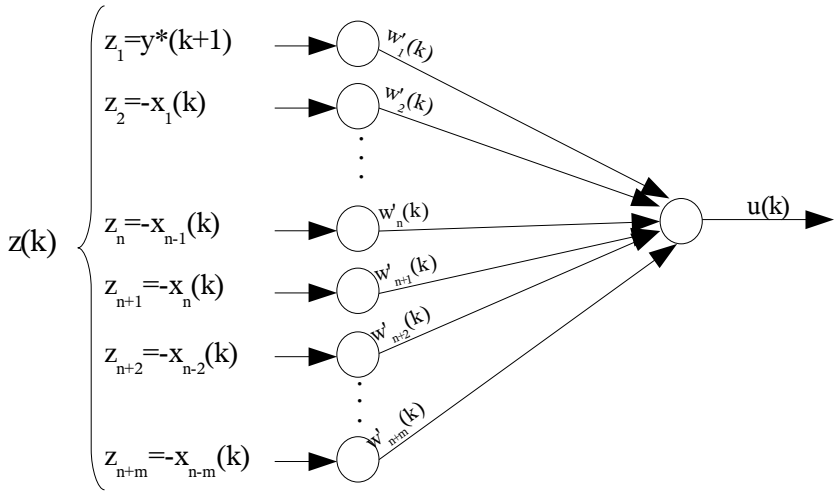

Ilustración 6.7: Controlador neuronal

La secuencia de acciones en el ciclo de control en el instante k sería el siguiente:

a) Calcular y*(k+1) y y(k)

b) Utilizar el modelo de la planta para calcular la salida estimada $\hat{y}(k)$ usando los pesos anteriores $w_i(k-1)$

c) Calcular las señales de error $e(k) = y(k) - \hat{y}(k)$, y usar la regla delta para calcular los nuevos pesos $w_i(k)$

d) Actualizar los pesos del controlador neuronal

e) Usar el controlador neuronal para generar la señal de control u(k).

6.8. Estructura del controlador multimodelo

Como hemos visto en el apartado anterior el uso de RNA en un sistema de control supone un mal transitorio, por lo que el aprendizaje será lento. Para evitar este problema modelamos un sistema capaz de decidir el mejor "camino" para reducir al máximo el tiempo de aprendizaje del sistema y por lo tanto estabilizar e igualar antes la señal de salida del sistema a la señal objetivo.

6.8.1. Solución propuesta

Este sistema se basa en multiplicar los nodos de salida y seleccionar según un criterio la mejor salida, esta es normalmente la que hace converger la señal más rápidamente. Es un proceso que selecciona la mejor salida en tiempo de calculo, por lo que salida que se selecciona varía en el tiempo, pudiendo existir una conmutación constante. Esta selección se realiza según una muestra y criterio, como por ejemplo, tomando los errores de las últimas cinco iteraciones y seleccionando la salida con la media de error más baja.

6.8.2. Procedimiento

Conceptualmente el procedimiento para utilizar el sistema multisalidas es sencillo.

Inicialmente tenemos una red neuronal con una o varias salidas y debemos multiplicar estas salidas por una constante, calculando de forma paralela para cada grupo de salidas y utilizando diferentes pesos sinápticos. Al realizar estos cálculos de forma paralela los resultados que se obtienen en un grupo de salidas no afectan a los resultados obtenidos en los demás grupos de salidas.

Tomemos como ejemplo la RNA de la Ilustración 6.9:

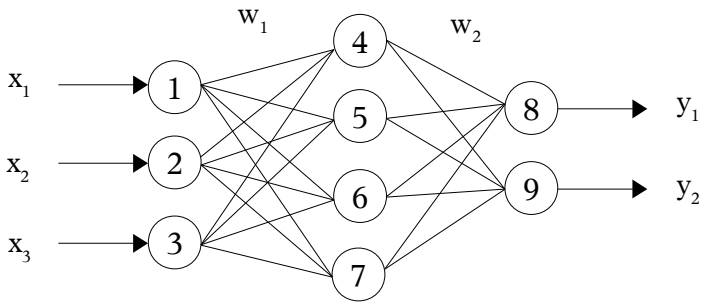

Ilustración 6.9: RNA sin multisalidas

Adaptaremos esta RNA para un sistema multisalidas con un multiplicador de salidas x2, quedando la RNA como en la Ilustración 6.10.

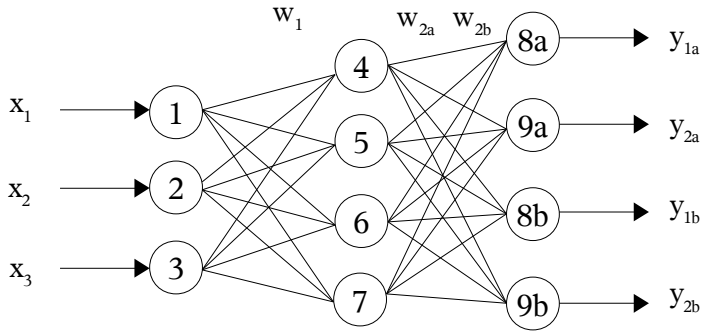

Ilustración 6.10: RNA con multisalidas (x2)

Para adaptar una RNA a un sistema multisalidas además de multiplicar por la constante multiplicadora del sistema (M) las salidas, también debemos multiplicar por M la matriz de pesos sinápticos de la capa de salida, ya que cada grupo de salidas dispone de sus propias matrices independientes de pesos.

Para entender mejor la conversión de una RNA a un modelo multisalidas la siguiente tabla muestra los parámetros de la red del ejemplo (Ilustración 6.9 y 6.10).

	Multiplicador	**Nº Capas**	**Entradas**	**Nodos capa oculta**	**Salidas**	**Tamaño W_1**	**Tamaño W_2**
RNA	N/A	2	3	4	2	2x3	4x2
RNA multimodelo	**2**	2	3	4	**4**	2x3	**4x4**

	Nº Iteraciones	**Tamaño vector objetivo**	**Aprendizaje**
RNA	200	2x200	Backpropagation
RNA multimodelo	200	**4x200**	Backpropagation

De igual forma que disponemos de matrices de pesos y salidas independientes para cada grupo en el sistema, también obtenemos vectores de error diferentes. Estos vectores de error son los que determinan, según un criterio preestablecido, cuales son las mejores salidas del sistema multisalidas.

6.8.3. Criterios de conmutación

A la hora de seleccionar la salida más “adecuada” podemos hacerlo según diferentes criterios. Todos ellos se basan en los valores de error que devuelve el sistema. En el proyecto se han implementado cuatro criterios distintos descritos a continuación.

6.8.3.1. Mejor salida en cada iteración

Se trata del criterio más básico. Comprueba en cada iteración el error obtenido y selecciona aquella salida con el error más bajo.

Con este criterio se corre el riesgo de obtener una conmutación constante en el sistema, la cual se traduciría en un mayor riesgo de divergencia de la señal obtenida.

6.8.3.2. Mejor peor caso

Este criterio comprueba los valores de error según una muestra dada por el usuario. En este caso para cada salida multiplicada se guarda el mayor error de cada muestra y de todos estos máximos errores se selecciona la salida con el más pequeño.

Por ejemplo, en una red con una única salida multiplicada por tres, y tomando una muestra de 4 iteraciones obtenemos los 3 vectores de error siguientes.

e_1= […,-4 , -2, 1.25, 0,75, ...]
e_2 = […,-3, 3.25, -2.75, 2, …]
e_3 = […, -0.15, 0.9, -4,25, 3, …]

Los máximos errores de e_1, e_2 y e_3 son 4, 3.25 y -4,25 respectivamente, por lo que según este criterio la mejor es la correspondiente al vector de error e_2 (Se han de tomar siempre los valores de error como valores absolutos, ya que no importa si el valor es positivo o negativo).

6.8.3.3. Media de error más baja

En este caso calculamos la media de error según una muestra y seleccionamos la media más baja.

Tomando el mismo ejemplo del apartado anterior, las medias de los valores absolutos de cada vector de error son:
e_1 = 2;
e_2 = 2,75;
e_3 = 2,075;

En este caso la mejor salida calculada es la correspondiente al vector de error e_1.

6.8.3.4. Error cuadrático medio con factor de olvido

Se calcula error cuadrático medio con factor de olvido para cada vector aplicando la fórmula (6.17) y seleccionando el valor más bajo.

$$\sqrt{\frac{\sum_{k=1}^{n}(0.995^{n}\cdot|e_k|^2)}{n}} \qquad (6.17)$$

Gracias al factor de olvido se da más importancia a los valores más próximos a la iteración actual y menos importancia a los que se calcularon anteriormente.

Se calcula el error cuadrático medio con factor de olvido para los vectores de error e_1, e_2 y e_3:

e1 = 2.3307

e2 = 2.7701

e3 = 2.6295

Al igual que en el apartado anterior la mejor salida es la correspondiente al vector de error e_1.

6.8.4. Tiempo de residencia

Se define el tiempo de residencia como la frecuencia con la que calcula la mejor salida del sistema multimodelo, por lo que este tiempo dependerá directamente de la muestra que se toma en cada criterio de selección.

De este tiempo de residencia (TR) depende la estabilidad del sistema frente a la conmutación. Por lo que si no tomamos un número de valores de muestra adecuados el sistema puede ser inestable y llegar a divergir.

A continuación se muestra un sistema discreto multimodelo con criterio de selección de error cuadrático medio con factor de olvido con distintos tiempos de residencia.

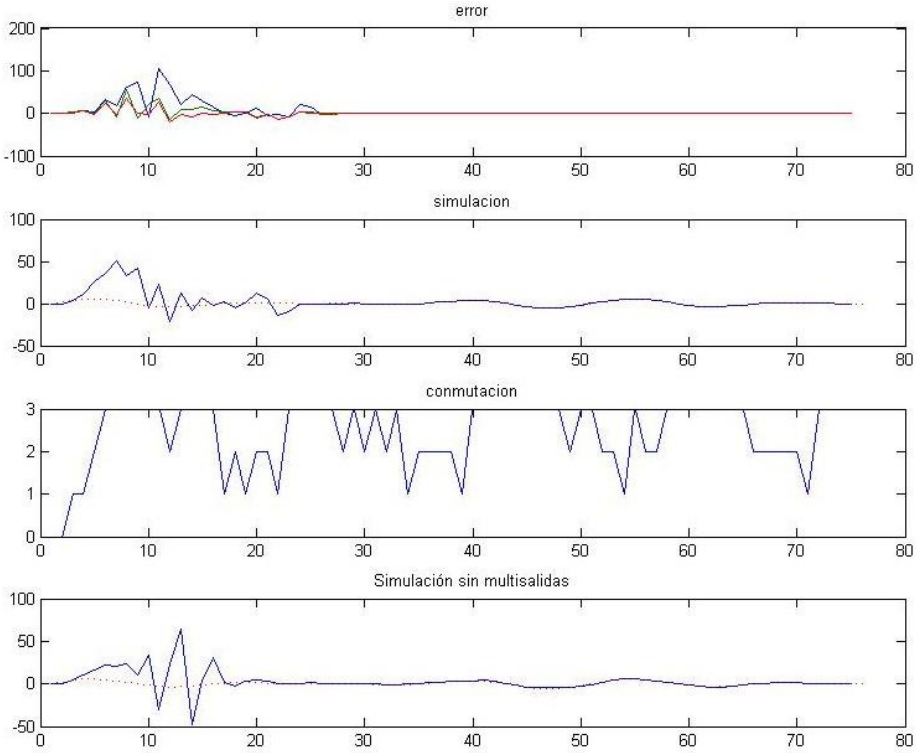

Ilustración 6.13: Sistema discreto multimodelo con TR = 1

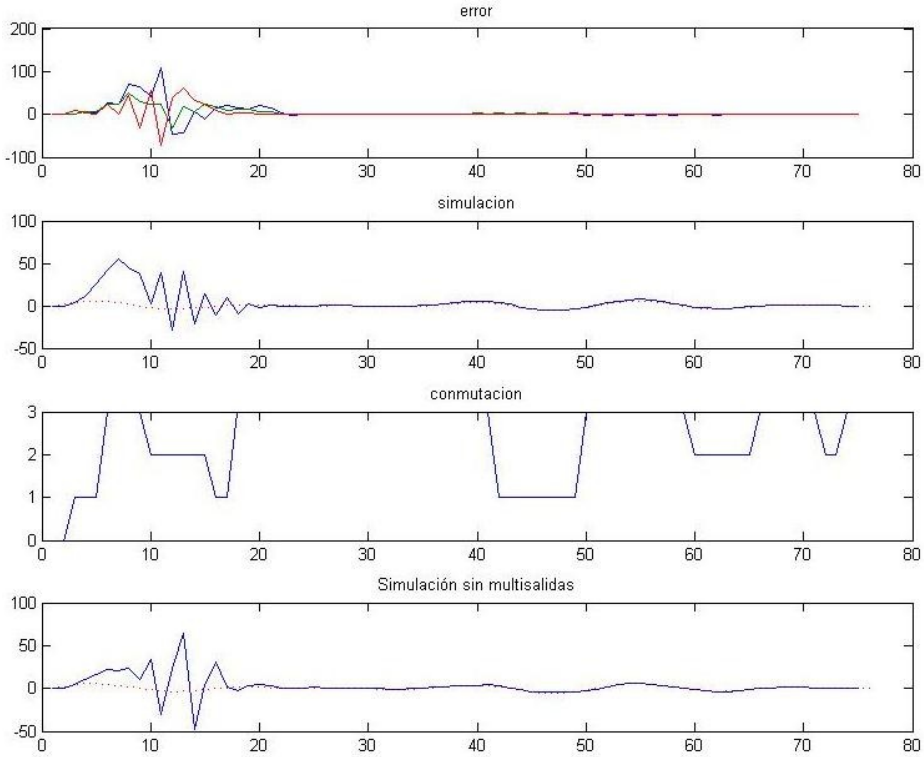

Ilustración 6.14: Sistema discreto multimodelo con TR = 2

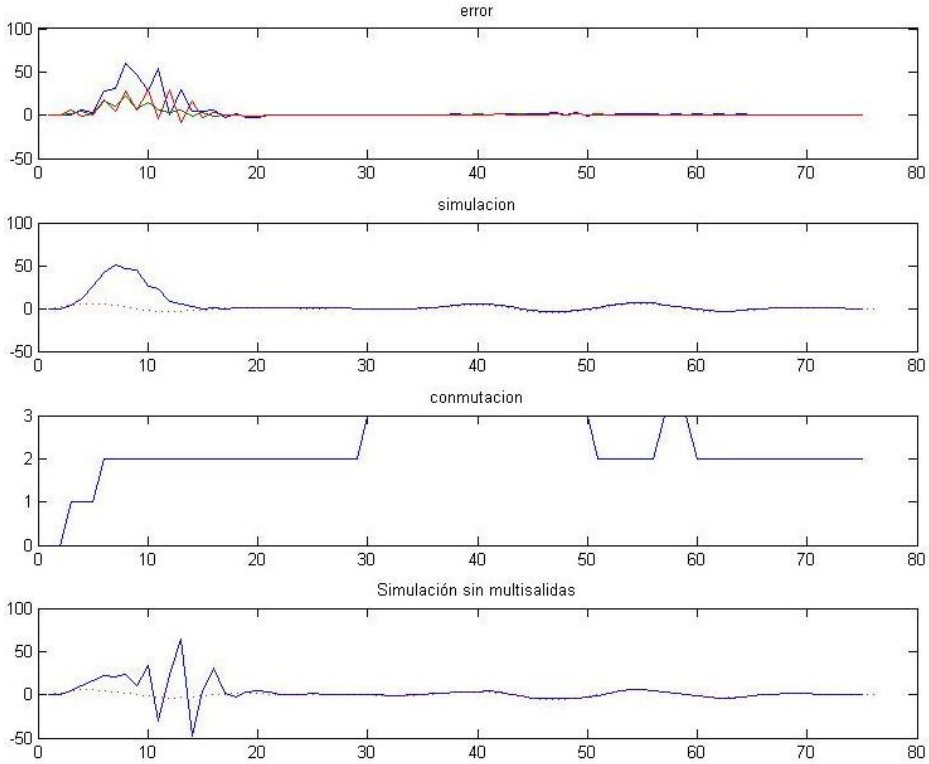

Ilustración 6.15: Sistema discreto multimodelo con TR = 3

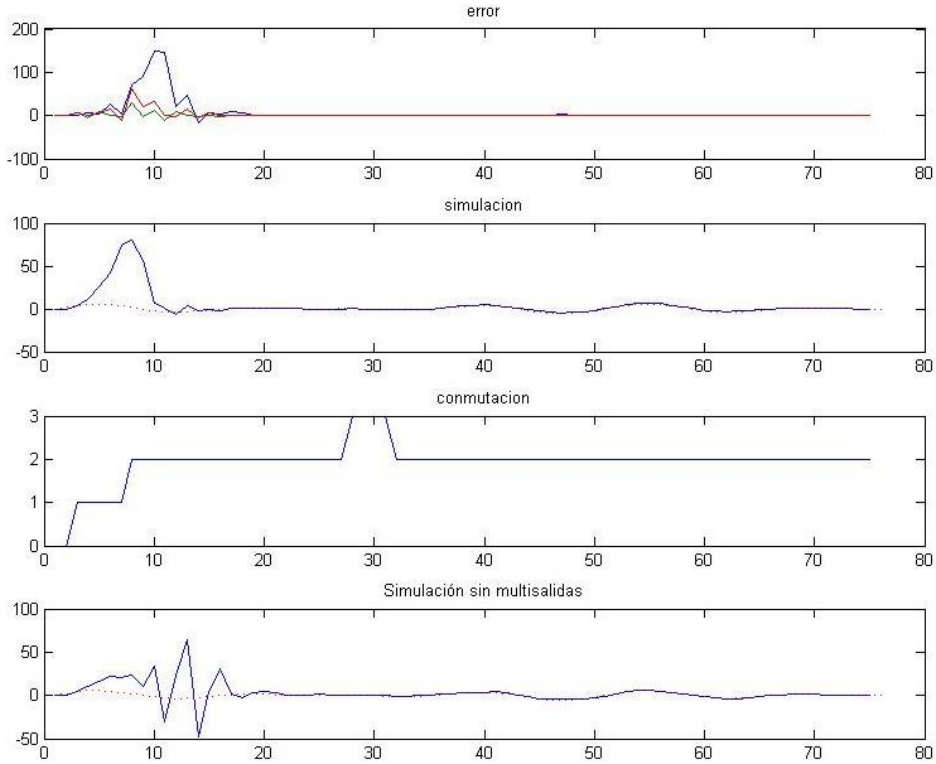

Ilustración 6.16: Sistema discreto multimodelo con TR = 4

Se comprueba que en las Ilustraciones 6.13 y 6.14 el sistema discreto multimodelo con TR = 1 y TR = 2 no mejora respecto al sistema discreto original. En cambio en las Ilustraciones 6.15 y 6.16, con tiempos de residencia de 3 y 4 iteraciones se mejoran los resultados frente al sistema discreto sin multimodelo.

En el articulo Ibeas. A [11] se calcula este tiempo de residencia mínimo según la ecuación (6.18):

$$TR_{min} > \frac{lnC}{|\ln \rho|} \qquad (6.18)$$

donde la constante C debe cumplir C > 0 y la constante ρ debe cumplir 1 > ρ > 0.

7. Implementación del sistema

En este apartado se explican los puntos más importantes a la hora de realizar la implantación del sistema de simulación de RNA utilizado .

El lenguaje de programación que se ha utilizado ha sido Matlab script. Este lenguaje dispone de muy pocas sentencias, ya que funciona mucho por funciones ya integradas en el sistema. Esto lo convierte en un lenguaje muy sencillo.

La herramienta que se ha utilizado es Matlab 7.3.0 (R2006b), la cual incluye una serie de funciones orientadas al calculo de redes neuronales artificiales, en cambio, se ha optado por implementar estas funciones por los motivos que ya se explicaron en el punto 2.2.1.

A nivel matemático la implementación ha sido complejo. Ha sido necesario estudiar y comprender la metodología del cálculo de las redes neuronales artificiales, así como los distintos algoritmos de aprendizaje que estas utilizan.

7.2. Requisitos

Los dos puntos siguientes describen los requisitos de desarrollo y de usuario.

7.2.1. Requisitos de desarrollo

Para el desarrollo de la aplicación sólo es necesario un ordenador y el software Matlab. El ordenador en cuestión no debe ser un equipo muy potente, sólo cumplir con los requisitos mínimos que necesita la instalación del software.

7.2.2. Requisitos de usuario

Al haberse desarrollado las funciones en Matlab script se hace necesaria la instalación del software Matlab para el uso de las funciones.

Aunque no es imprescindible, a la hora de ejecutar algunas de las funciones desarrolladas es interesante disponer de un procesador potente, ya que el tiempo de cálculo se puede reducir de forma considerable.

Ya que el usuario debe introducir los datos con los que el sistema trabaja es necesario que disponga de un conocimiento básico de las redes neuronales y los sistemas de control.

7.3. Casos de uso

El siguiente esquema muestra los casos en los que se puede encontrar un usuario. Existe un único actor, el cual puede crear y simular una red.

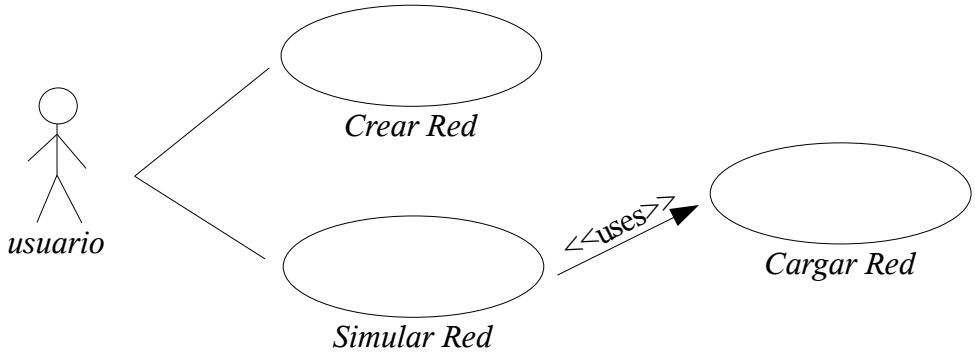

Ilustración 7.1: Casos de uso

Estas funciones simplemente simulan una red neuronal y muestra una figura con los resultados de la simulación.

Teniendo en cuenta la complejidad y las diferencias que existen entre los distintos sistemas de control que utilizan redes neuronales y que se utilizan en esta memoria para comprobar el funcionamiento del sistema con múltiples salidas se ha optado por crear una función para cada tipo de sistema de control, quedando como resultado las siguientes funciones:

- sistema_discreto
- sistema_discreto_multi
- sistema_control_back_y
- sistema_control_back_multi_y
- sistema_control_back_u
- sistema_control_back_u_multi
- sistema_control_back_union
- sistema_control_back_union_online
- sistema_control_back_union_online_multi
- sistema_control_no_lineal
- sistema_control_no_lineal_backpropagation
- sistema_control_no_lineal_multi
- sistema_control_no_lineal_sinred

7.4. Variables

A continuación se describen las variables más importantes que definen el tipo de red neuronal que el sistema simula. Estas variables deben estar definidas a la hora de utilizar la función “crear_red” en el caso que se desee simular sólo una red, si se quiere simular alguno de los sistemas de control estas variables están definidas dentro del código de cada una de las funciones.

nombre

Es el nombre que el usuario puede darle a su red. Al utilizar la función “crear_red” se creara una carpeta con el mismo nombre, donde se crearan una serie de ficheros que contendrán la información del resto de variables. Este nombre es también el utilizado para la función “simulacion”.

Esta variable es de tipo “string”, por lo que en Matlab se define con comillas simples (ejemplo: 'nombre1').

topologia

Es un vector con n elementos para una red de n-1 capas. Cada valor indica el numero de nodos (neuronas) que hay en la capa n-1, siendo la entrada la capa 0. Por ejemplo, si la variable topologia = [2 4 1] entonces se tratara de una red con dos nodos de entrada, cuatro nodos en la capa oculta y un único nodo de salida.

El programa nos permite simular redes de una a tres capas sin entrenamiento, redes de una o dos capas para el entrenamiento *backpropagation* y redes de una capa para el entrenamiento *ADALINE*.

entrada

Es una matriz con n filas y m columnas donde n corresponde al número de nodos de la entrada y m son los valores de cada entrada. Por ejemplo, el valor de la tercera columna en la segunda fila corresponderá al valor de entrada de la segunda salida en la tercera iteración de la simulación.

salida

Es una matriz de n filas y 10 columnas donde n corresponde al número de nodos de la capa de salida. El primer valor de cada fila indica el tipo de función que leerá e interpretará la función “lista_funciones”. El resto de valores de cada fila son los parámetros del tipo de función.

func_act

Funciona de idéntica forma que la variable salida, pero en este caso el número de filas debe corresponder con el número de capas de la red.

pesos

Existen tres variables para los pesos: pesos1, pesos2 y pesos3.

Son matrices de n filas y m columnas donde n corresponde al número de

nodos de la capa predecesora y m al número de nodos de la capa sucesora. Por lo tanto, a una red con 3 nodos de entrada y 2 de salida le corresponderá una única matriz de pesos de tamaño 3x2.

Se deberán utilizar tantas de estas variables como capas tenga la red, por lo que sólo en el caso de una red de 3 capas se utilizarán las 3 variables de pesos.

entrenamiento

Es un vector con 10 elementos donde el primer valor corresponde al tipo de entrenamiento y el resto de valores son los parámetros del tipo de entrenamiento.

7.4.1. Ejemplo práctico

Veamos como se definen las variables con una red neuronal de ejemplo:

Tomemos la red de la Ilustración 7.2:

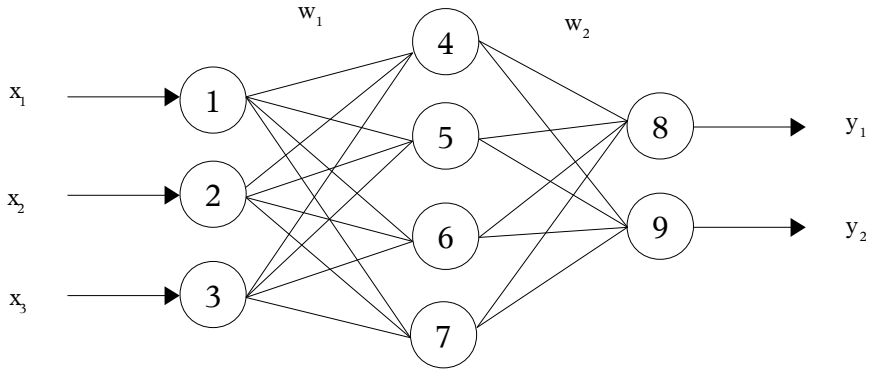

Ilustración 7.2: Red neuronal artificial de 2 capas

```
nombre = 'red2capas';
topologia = [3 4 2];
entrada = [0:0.01:1.5;0:0.02:3;0:0.04:6];
salida =[0 sym('((x1^2)*(1/4))') 0 0 0 0 0 0 0 0;5 3 0 0 0 0 0 0 0 0];
func_act = [6 2 0 0 0 0 0 0 0 0;4 0 0 0 0 0 0 0 0 0];
pesos1 = [-0.2 0.5 1 0.4;0 1 0.3 -0.1;-0.3 0.4 -1 -0.6];
pesos2 = [0.1 -0.2;0.3 0.5; 0.7 0.2; 0.6 -0.5];
```

```
pesos3 = 0;
entrenamiento = [2 0.1 0 0 0 0 0 0 0 0];
```

Interpretando los valores de todas las variables obtenemos una red de 2 capas con 3 entradas, 4 nodos en la capa oculta y 2 nodos en la capa de salida. Tenemos una muestra de 151 valores de entrada para la simulación con una salida objetivo para el primer nodo con la función f = x^2 * ¼ para el primer nodo de salida y una función Gaussiana con sigma = 3 y c = 0 para el segundo nodo de salida.

La función de activación de la primera capa es la función lineal f = 2x u la función de activación de la segunda capa es la función tangencial sigmoidal.

Las variables pesos1 y pesos2 corresponden a las siguientes matrices de pesos:

$$W_1 = \begin{pmatrix} -0.2 & 0.5 & 1 & 0.4 \\ 0 & 1 & 0.3 & -0.1 \\ -0.3 & 0.4 & -1 & -0.6 \end{pmatrix} \qquad (7.1)$$

$$W_2 = \begin{pmatrix} 0.1 & -0.2 \\ 0.3 & 0.5 \\ 0.7 & 0.2 \\ 0.6 & -0.5 \end{pmatrix} \qquad (7.2)$$

Por último, la variable entrenamiento índica que se utilizará el aprendizaje *Backpropagation* con alpha = 0.1.

7.5. Funciones

A continuación se describirán algunas de las funciones implementadas comentando los aspectos más importantes de cada una de ellas.

7.5.1. Función “crear_red (nombre, topologia, entrada, salida, func_act, pesos1, pesos2, pesos3, entrenamiento)”

Esta función toma los valores de las variables descritas en el punto 5.4. para guardar toda esa información en un directorio con el nombre que se haya definido en la variable “nombre”. Dentro de este directorio se generan 6 ficheros, cada uno de ellos con una extensión que índica el tipo de información que contiene:

- nombre.af (función de activación)
- nombre.ann (topologia)
- nombre.in (valores de la/s entrada/s)
- nombre.out (valores de la/s salida/s referencia)
- nombre.trf (tipo de entrenamiento)
- nombre.wnm (pesos sinápticos)

Se utiliza de la forma crear_red(nombre, topologia, entrada, salida, func_act, pesos1, pesos2, pesos3, entrenamiento).

Antes de crear los ficheros la función crear_red se encarga de comprobar que no existan discordancias en los parámetros, como por ejemplo que los pesos no coincidan con la topología de la red o que no coincidan el número de funciones de activación con el número de capas.

7.5.2. Función “simulacion('nombre_red')”

Se utiliza de la forma simulacion('nombre_red') y simula la red “nombre_red” utilizando las variables que la función “cargar_red” haya cargado previamente.

Al finalizar la simulación se muestra unas gráficas que variarán según el tipo de entrenamiento utilizado. En el caso de no utilizar entrenamiento se muestra una gráfica con los valores de entrada y una gráfica con los valores calculados de

salida. Para las simulaciones que utilizan los entrenamientos ADALINE y *Backpropagation* se muestran 3 gráficas: la primera con el progreso del error, la segunda con los valores de la salida objetivo/referencia y la última con los valores reales calculados por la red neuronal.

7.5.3. Función "sistema_control_back_union"

Esta función y las de los apartados 7.5.4 y 7.5.5 están basadas en el sistema de control neuronal adaptativo descrito en el apartado 6.6 y que corresponde a la Ilustración 7.3.

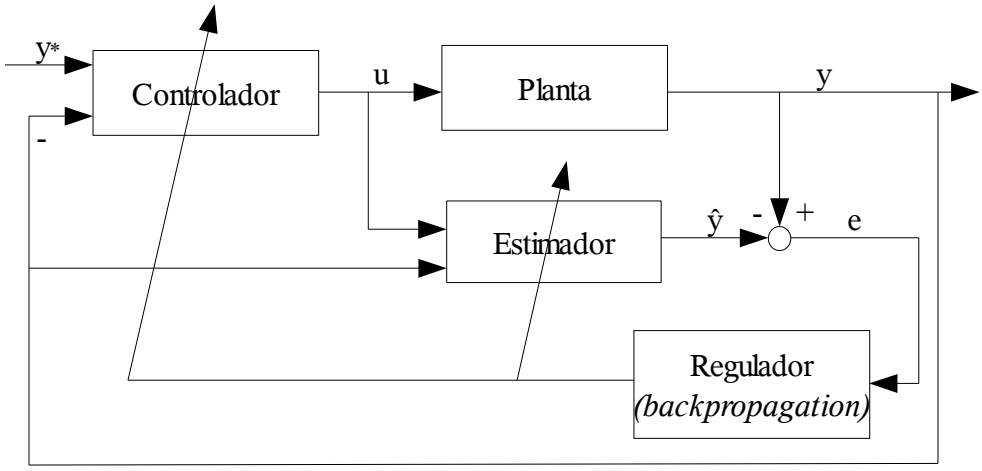

Ilustración 7.3. Sistema adaptable de control.

Recordemos que como se describía en el apartado 3.6 el regulador utilizaba la regla de aprendizaje Widrow-Hoff (ADALINE), en el sistema de control adaptativo simulado se utiliza la regla *Backpropagation*, los que nos permite utilizar redes de hasta 3 capas, ya que la regla de aprendizaje ADALINE esta limitada a redes de una única capa.

Las funciones "sistema_control_back_u" y "sistema_control_back_y" son el controlador neuronal y el modelo neuronal de la planta respectivamente. Cada una de estas funciones calcula los pesos para toda la muestra de valores de entrada que se obtienen de la función "sistema_control_no_lineal_sinred", de esta última función también obtenemos los valores de la salida referencia con

los que trabaja el estimador.

Al trabajar de forma *offline* primero se obtienen los valores de los pesos para el controlador y la planta de forma independiente, una vez obtenidos todos los pesos para toda la muestra estos se utilizan para la simulación de todo el sistema de control.

Por lo tanto, la secuencia de acciones a seguir para el cálculo del sistema de control adaptativo *offline* con aprendizaje backpropagation es el siguiente:

1. Obtenemos los valores de entrada y salida referencia a partir de la función “sistema_control_no_lineal_sinred”.

2. Utilizando el aprendizaje Backpropagation obtenemos los pesos del controlador neuronal (función “sistema_control_back_u”) y el modelo neuronal de la planta (función “sistema_control_back_y“).

3. Simulamos el sistema de control usando los pesos obtenidos en anterior apartado (función “sistema_control_back_union”).

7.5.4. Función “sistema_control_back_union_online”

Como ya se dijo en el punto anterior esta función se basa en el esquema de control de la Ilustración 7.3. La única diferencia con la función descrita anteriormente es que esta trabaja de forma online. Esto significa que los pesos del controlador neuronal y del modelo neuronal de la planta se calculan conjuntamente.

En este caso, la secuencia de acciones para el cálculo del sistema de control adaptativo online con aprendizaje backpropagation es el siguiente:

1. Obtenemos los valores de entrada y salida referencia a partir de la función “sistema_control_no_lineal_sinred”).

2. Tomamos el primer valor de entrada y actualizamos los pesos del controlador y el modelo neuronal de la planta.

3. Simulamos el sistema de control con los pesos obtenidos en el punto 2.

4. Repetimos los puntos 2 y 3 hasta haber simulado todos los valores de entrada.

7.5.5. Función “sistema_control_back_union_online_multi”

La función “sistema_de_control_back_union_online_multi” utiliza una simulación online como la función del apartado anterior, en cambio, tanto el controlador neuronal como el modelo neuronal de la planta se aprovechan de las ventajas de las redes neuronal multimodelo descritas en el apartado 6.8.

Al utilizar un sistema multimodelo se utilizan las funciones “sistema_control_back_u_multi” y “sistema_control_back_y_multi” para el calculo de *u* e *y* respectivamente.

8. Resultados de la simulación

En este apartado se muestran los resultados de las simulaciones de control de las funciones descritas en el apartado anterior.

8.1. Función “simulacion”

La función “simulacion” nos permite simular infinitos modelos de redes neuronales, con o sin aprendizaje y de una hasta tres capas (dependiendo del tipo de aprendizaje). En este apartado sólo mostraremos los resultados de los principales modelos de red neuronal que serán una red tipo ADALINE y una red de dos capas con aprendizaje *backpropagation*.

8.1.1. Red neuronal ADALINE

Como ya se ha mencionado con anterioridad el aprendizaje ADALINE sólo puede utilizarse con redes neuronales de una única capa.

Para la simulación se toma la red de la Ilustración 8.1.

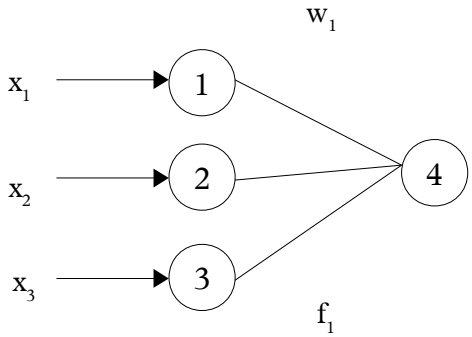

Ilustración 8.1: Red tipo ADALINE

Los valores de entrada son una muestra de 201 valores:

x_1 = [0:0.5:100]

x_2 = [1:1:201]

x_3 = [0:3:600]

La salida deseada se corresponde con la función 8.1:

$$y'_1 = x_1 - 4x_2 + x_3 - 3 \qquad (8.1)$$

La función de activación es:

$$f_1 = 2x \qquad (8.2)$$

Los pesos iniciales son:

$$W_1 = \begin{pmatrix} 0.5 \\ 1 \\ -0.2 \end{pmatrix} \qquad (8.3)$$

El gráfico resultante es el mostrado en la Ilustración 8.2:

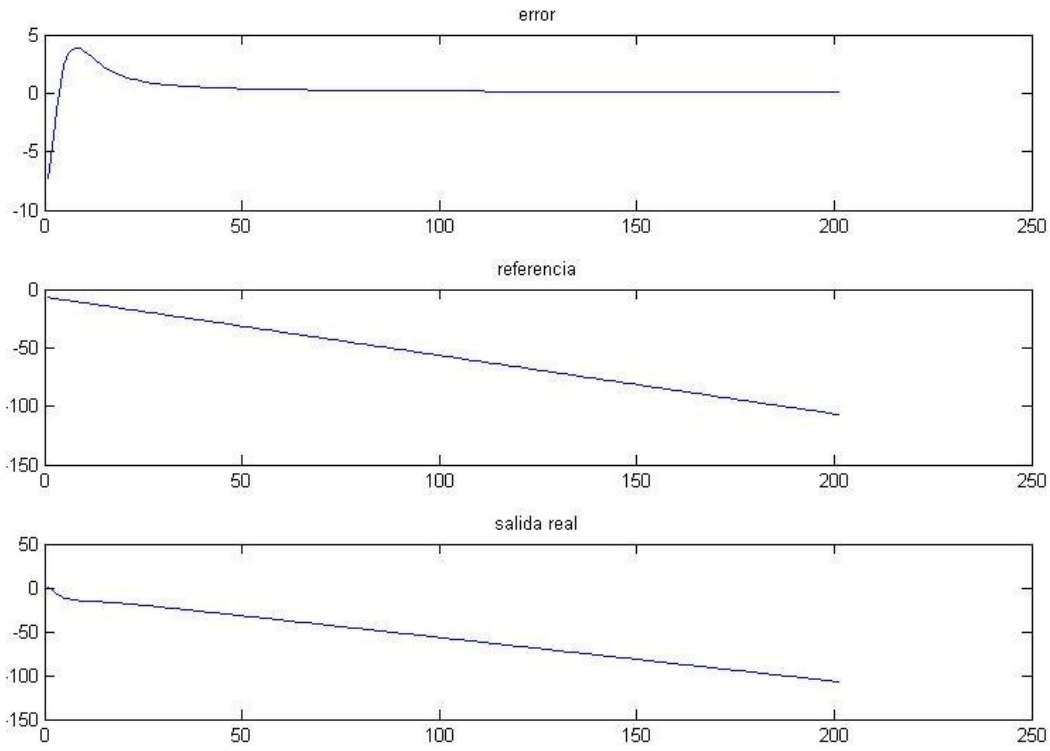

Ilustración 8.2: Simulación red ADALINE

Al finalizar la simulación la matriz de pesos W_1 se estabiliza en los valores:

$$W_1 = \begin{pmatrix} 0.6319 \\ -1.9436 \\ 0.3674 \end{pmatrix} \qquad (8.4)$$

8.1.2. Red de 2 capas con aprendizaje Backpropagation

La red a simular es la mostrada en la Ilustración 8.3.

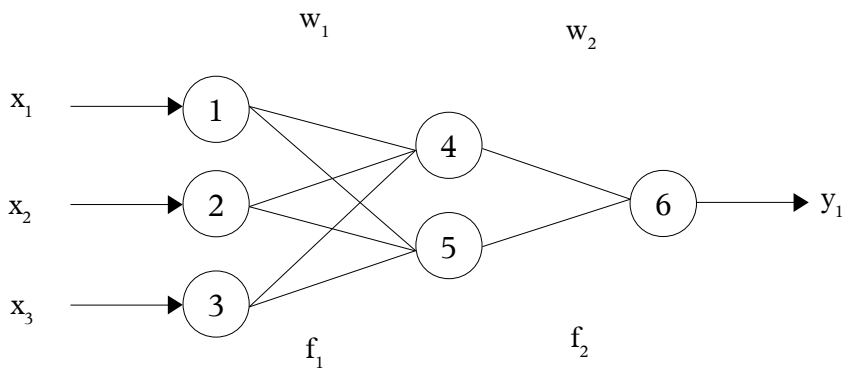

Ilustración 8.3: Red neuronal de dos capas

La muestra de entrada tiene 151 valores:

x_1 = [0:0.01:1.5]

x_2 = [-1:0.02:2]

x_3 = [1:0.04:7]

Descartamos el valor $x_3 = 0$ para evitar la división por cero en la función referencia *(8.9)*.

La salida de referencia tomada para el aprendizaje Backpropagation es:

$$y'_1 = \frac{2 \cdot x_1^2 + x_2}{2 \cdot x_3} \qquad (8.5)$$

Tomamos como funciones de activación una función lineal y la función

sigmoidal tangencial respectivamente:

$$f_1 = 2x \tag{8.6}$$

$$f_2 = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{8.7}$$

Para el entrenamiento utilizamos el valor α = 0.1 y los pesos iniciales W_1 y W_2:

$$W_1 = \begin{pmatrix} -0.2 & 0.5 \\ 0 & 1 \\ -0.3 & 0.4 \end{pmatrix} \tag{8.8}$$

$$W_2 = \begin{pmatrix} 0.1 \\ 0.3 \end{pmatrix} \tag{8.9}$$

Al finalizar la simulación obtenemos la siguiente gráfica:

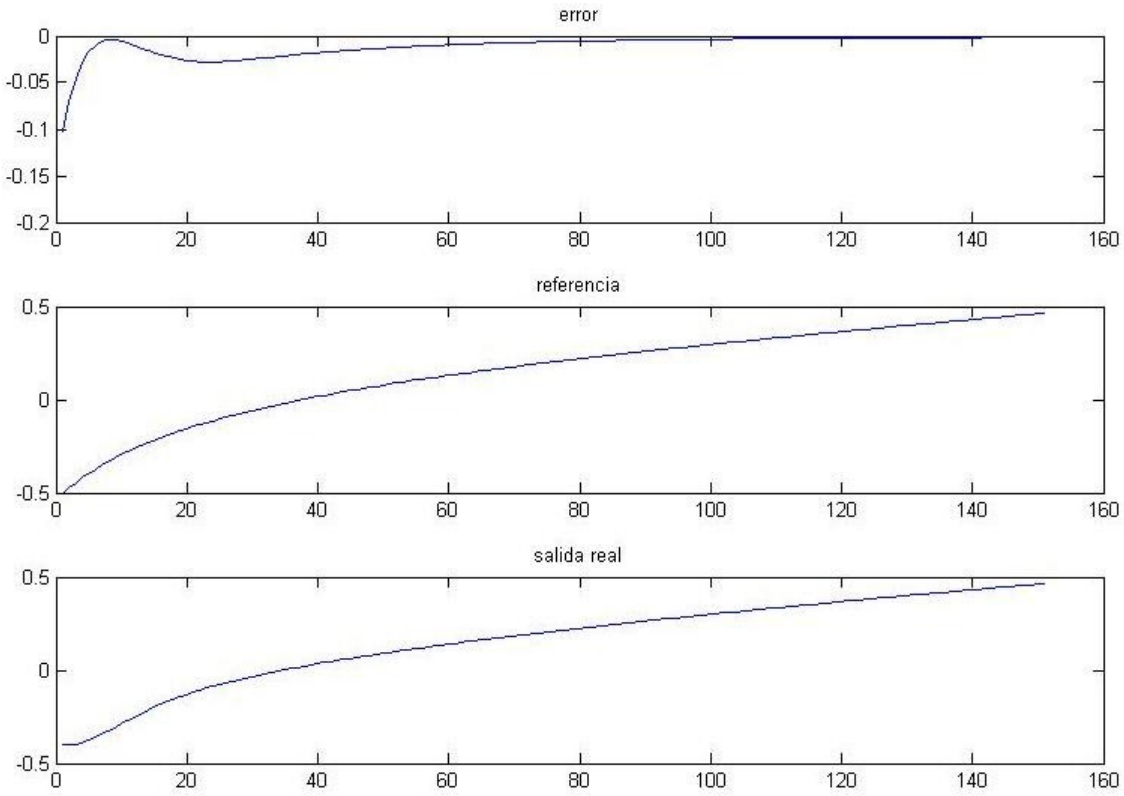

Ilustración 8.4: Simulación red neuronal con aprendizaje Backpropagation

Al finalizar el entrenamiento obtenemos los nuevos valores de W_1 y W_2:

$$W_1=\begin{pmatrix} -0.2282 & 0.4259 \\ 0.0158 & 1.0419 \\ -0.485 & -0.0868 \end{pmatrix} \quad (8.10)$$

$$W_2=\begin{pmatrix} 0.1089 \\ 0.3089 \end{pmatrix} \quad (8.11)$$

8.2. Sistema de control adaptativo online con aprendizaje backpropagation

Esta simulación se basa en el esquema de control de la Ilustración 7.3.

Se emula un sistema de control no-lineal (8.12), ya que es en el caso no-lineal cuando este esquema es especialmente efectivo dada la dificultad de controlar sistemas no-lineales.

$$y(k)=-0.73\cdot y(k-1)\cdot|y(k-2)^{\frac{1}{3}}|+0.1\cdot y(k-2)+2\cdot u(k) \quad (8.12)$$

Donde y(k-1) e y(k-2) son los valores anteriores de la misma función y u(k) es la entrada de la planta.

Como señal de referencia utilizamos el siguiente polinomio:

$$r(k)=0.1\cdot\sin(2t)+0.36\cdot\sin(3t)+0.25\cdot\sin(7\pi t) \quad (8.13)$$

El rango de valores de *t* corresponde al vector:

t = [0:0.02:10] *(8.14)*

Este mismo sistema de control es el que queremos aproximar en la simulación, por lo tanto los valores resultantes este sistema para la muestra *t* serán los valores de referencia que se tomaran para el aprendizaje de las controlador

neuronal y el modelo neuronal de la planta.

En la simulación se utilizan un controlador neuronal y un modelo neuronal de la planta, ambos implementados por la red neuronal de la Ilustración 6.6:

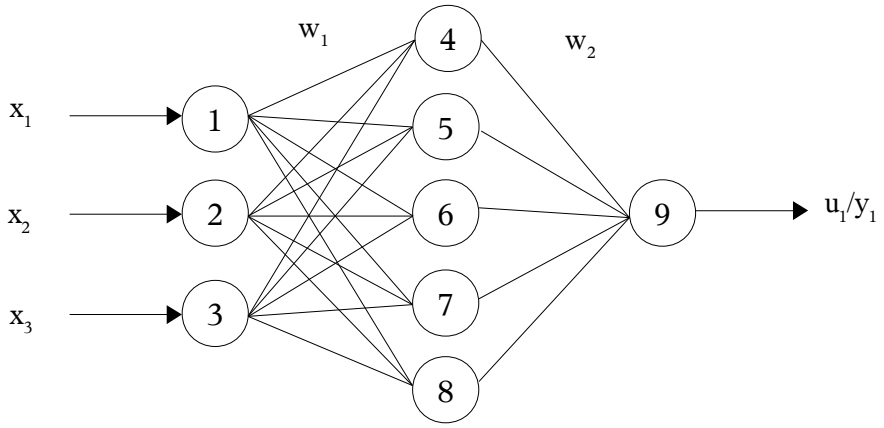

Ilustración 8.5: Modelo neuronal del controlador y la planta

En el controlador neuronal las entradas [x_1, x_2, x_3] corresponden a los valores [y, y^{-1}, y^{-2}], sabiendo que y^{-1} e y^{-2} son y(k-1) e y(k-2) respectivamente.
En el modelo neuronal de la planta las entradas [x_1, x_2, x_3] son [y^{-1}, y^{-2}, u] respectivamente.

Para el controlador la salida obtenida será u(k), para la planta y(k).

En ambos modelos neuronales se inicializan los pesos de W_1 y W_2 con todos los elementos a 1. Las funciones de activación f_1 y f_2 son la función tangencial hiperbólica.

Para el entrenamiento *backpropagation* α = 0.1.

A diferencia del sistema de control offline el sistema de control online modifica los pesos del controlador y la planta neuronal de forma conjunta, lo que se traduce en un mejor aprendizaje y una mayor aproximación al sistema de control referencia (8.13).

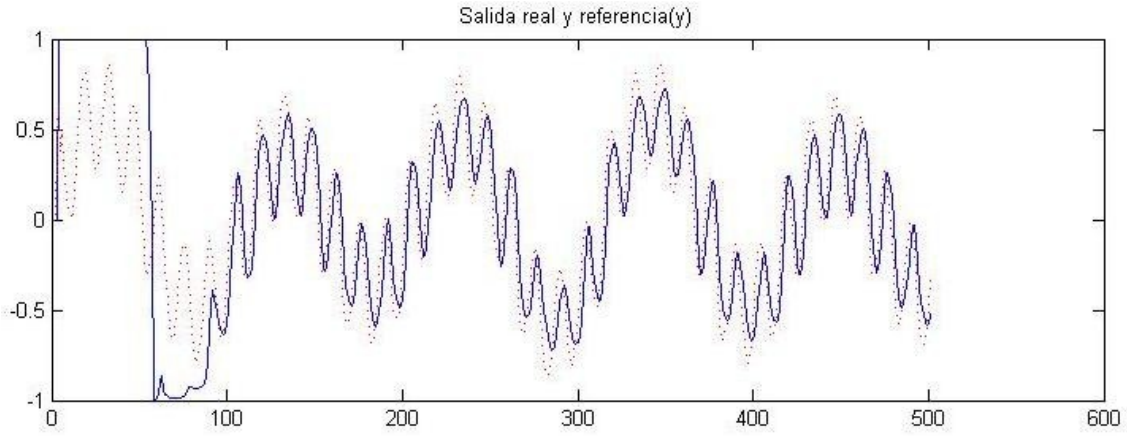

Ilustración 8.6: Simulación del sistema de control adaptativo online

Como se comprueba en la Ilustración 8.7 el transitorio del sistema de control simulado no es muy bueno, ya que hasta aproximadamente la iteración nº 100 la señal real del sistema no se adapta a la señal referencia. Este problema se soluciona con el uso de un sistema multisalidas, simulado en el siguiente apartado.

8.3. Sistema de control adaptativo online utilizando un sistema multimodelo

Esta simulación utiliza los modelos neuronales del controlador y la planta descritos en el apartado 8.2 mejorados con el sistema multisalidas.

En este caso el multiplicador de salidas es 3, con lo que se obtienen dos redes neuronales con 3 salidas y 2 matrices de pesos asociadas a cada salida, con lo que se obtiene un total de 6 matrices de pesos.

Las matrices de pesos asociadas a la primera salida se corresponden con las utilizadas en los dos apartados anteriores, por lo tanto son matrices inicializadas con todos los elementos a 1. Los elementos de las matrices asociadas a las otras dos salidas se inicializan con valores aleatorios

comprendidos en el intervalo [-1,1].

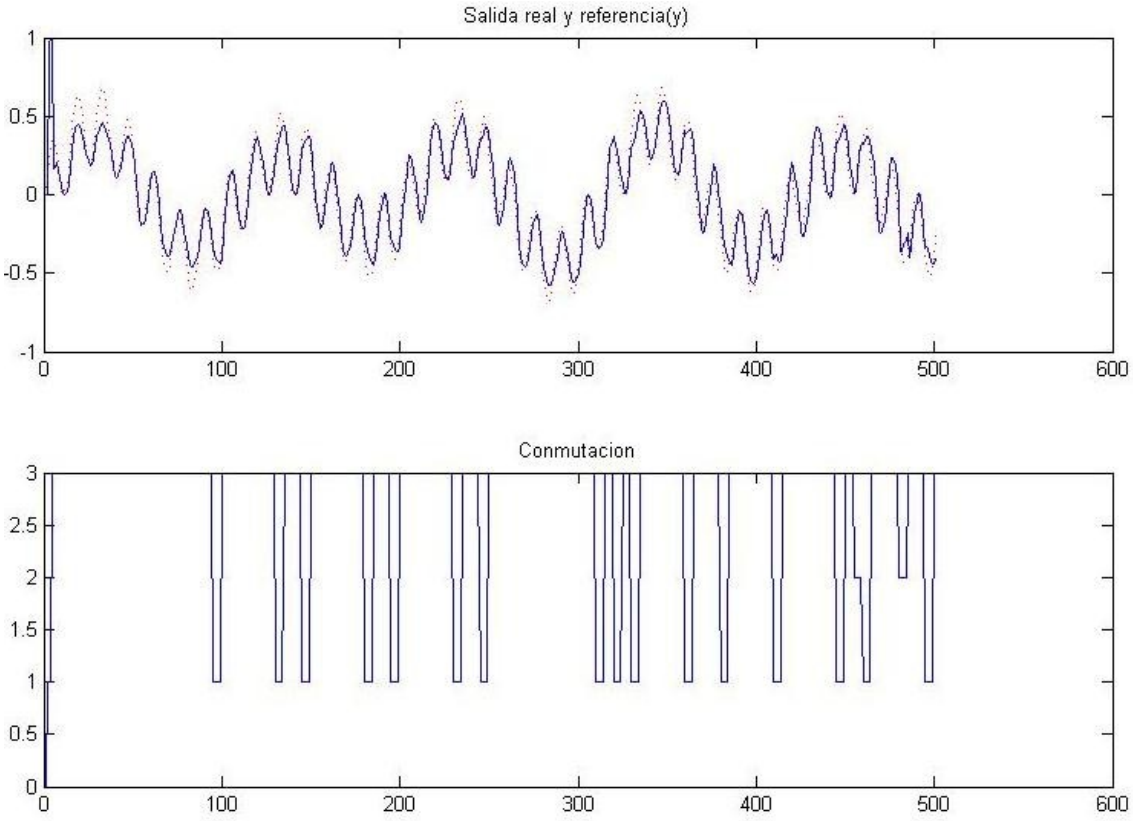

Ilustración 8.7: Simulación del sistema de control adaptativo con multisalidas

La Ilustración 8.7 muestra la salida real (linea continua) y referencia (linea discontinua), el error asociado a cada una de las salidas multiplicadas y la conmutación entre estas salidas.

Como se observa la mayor parte del tiempo esta tomando como mejor salida la número 3 y sólo toma los resultados de la salida 1 de forma esporádica y los de la salida 2 de forma puntual. Al generar los pesos de las salidas 2 y 3 de forma aleatoria la tendencia de la conmutación puede variar.

Comparando la salida real de la gráfica de la Ilustración 8.7 con la gráfica de la Ilustración 8.6 se puede ver claramente como con el sistema multisalidas el transitorio del sistema es mucho mejor, ya que el aprendizaje en este último caso es prácticamente inmediato.

9. Conclusiones

En este proyecto se ha abordado el tema de la aplicación de redes neuronales artificiales multimodelo al control de sistemas.

En las primeras fases del proyecto se han estudiado las redes neuronales artificiales para posteriormente implementarlas en Matlab desde inicio sin ayudarnos de las librerias que el programa incorpora para la simulación de estas, para así profundizar más en el funcionamiento de las mismas.

A continuación y gracias a las funciones desarrolladas en la simulación de redes neuronales artificiales se ha desarrollado un sistema de control que funciona con un controlador y un modelo de planta neuronal. Esta adaptación con RNA nos permite simular sistemas de control donde la estructura de la planta en el sistema es desconocida.

A partir de los resultados obtenidos se observa que el uso de las RNA en el control de sistemas supone velocidad de convergencia lenta. En la práctica, esto supone un problema, ya que existe un periodo de transición en el que el sistema no responde como se desea.

Gracias a un sistema multimodelo de redes neuronales artificiales conseguimos mejorar el aprendizaje de estás de forma muy notable. Este sistema se basa en la simulación de nuevas salidas generadas a partir de las salidas originales. Asimismo, este nuevo desarrollo no supone un incremento notable en el procesamiento de datos, ya que este se realiza de forma paralela.

Este sistema multimodelo puede seguir distintos criterios basados en el número de veces que se multiplican las salidas, el tiempo de conmutación de las salidas o la forma de seleccionar la mejor salida, por lo que en las fases finales del proyecto se simulan los distintos criterios, comprobando la importancia del

tiempo de residencia seleccionado y de como este puede comprometer la estabilidad del sistema.

9.1. Lineas futuras de ampliación

Para finalizar proponemos algunas mejoras a realizar tanto de manera teórica como practica reflejándolas en el conjunto de aplicaciones implementadas:

- Aunque durante la implementación de las aplicaciones se ha intentado conseguir un nivel de eficiencia alto dentro de estas, la naturaleza de los algoritmos usados exige un gran coste de cálculo. De esta manera es interesante la implementación de atajos que aceleren el sistema.
- Uno de las funciones que necesita un mayor tiempo de cálculo es la que implementa el sistema multimodelo. Este sistema funciona para toda la muestra de valores dentro de la simulación. Se propone utilizar el sistema multimodelo sólo hasta que el aprendizaje haya alcanzado un valor satisfactorio, una vez conseguido esto la red se simula normalmente sin utilizar este sistema. Aunque la implementación de esta propuesta puede ser más compleja, la ganancia en tiempo de cálculo puede ser muy notable.
- En la implementación de las redes neuronales artificiales sólo se han tenido en cuenta los sistemas de aprendizaje supervisados *Widrow-Hoff* y *backpropagation*, queda abierta la posibilidad de incluir tipos de aprendizaje no supervisados. Estos algoritmos de aprendizaje no necesitan de un conjunto de datos de entrada cuya respuesta objetivo sea conocida.
- El uso de las funciones desarrolladas esta limitado a la linea de comando de Matlab, lo que hace necesario tener algunos conocimientos acerca del uso de estas. Existe la posibilidad de desarrollar un entorno gráfico en Matlab, lo que se traduciría en un uso más rápido e intuitivo de las funciones.

10. Bibliografía

[1] A. R. Carrera, M. Martínez, “Introducción a Matlab y a la creación de interfaces gráficas”. Servicio Editorial de la Universidad del País Vasco, 2004.

[2] E. N. Sánchez, A. Y. Alanís, “Redes neuronales: conceptos fundamentales y aplicaciones a control automático”. Pearson-Prentice Hall, 2006.

[3] Hornik, K, Stinchcombe, M., White, H. “Multilayer feedforward networks are universal approximators” Neural Networks, vol. 2, pags. 359-366, 1989.

[4] P. Isasi, I. M. Galván, “Redes de neuronas artificiales: un enfoque práctico”. Pearson-Prentice Hall, 2004.

[5] S. Barro, J. Mira, “Computación neuronal”. Universidad de Santiago de Compostela. 1995, Pags. 181-212.

[6] S. Lesueur, D. Massicotte, P. Sicard. “Indirect Inverse Adaptative Control Based on Neural Networks Using Dynamic Back Propagation for Nonlinear Dynamic Systems”. Electrical and Computer Engineering Department, Université du Québec à Trois-Rivières, 2002.

Fuentes electrónicas

[7] “Electrónica de control”. Wikipedia [Online]. Disponible en: http://es.wikipedia.org/

[8] D. J. Horat, J. L. Cañizales. “Backpropagation”. DavidHorat.com [Online]. Disponible en: http://es.davidhorat.com/

[9] F. Rodríguez, M. J. López (1996). “Control adaptativo y robusto”. Google Libros [Online]. Disponible en: http://books.google.es/

[10] A. Ibeas, M. de la Sen y S. Alonso-Quesada. “A supervised discrete adaptive control scheme with multiestimation”. Instituto de Investigación y Desarrollo.de Procesos. Disponible en: http://www.iidp.ehu.es

[11] Ibeas, A. , De La Sen, M., Alonso-Quesada, S. “Stable multi-estimation model for single-input single-output discrete adaptive control systems”, International Journal of Systems Science, (2004). Disponible en: http://www.informaworld.com

[12] J. García de Jalón, J. I. Rodríguez, J. Vidal (2005). “Aprenda Matlab 7.0 como si estuviera en primero”. Disponible en: http://mat21.etsii.upm.es/

[13] “Ingeniería automática”. Wikipedia [Online]. Disponible en: http://es.wikipedia.org/

[14] M. Gómez. “Sistemas de tiempo real”. SlideFinder [Online]. Disponible en: http://www.slidefinder.net/

[15] “Neurona”. Wikipedia [Online]. Disponible en: http://es.wikipedia.org/

[16] P. Coronado. “Control avanzado práctico”. Universidad Autónoma de Madrid [Online]. Disponible en: http://www.uam.es/

[17] R. Zurco. “Control adaptativo con modelo de referencia”. SlideShare [Online]. Disponible en: http://www.slideshare.net/

[18] “Sistema de control”. Wikipedia [Online]. Disponible en: http://es.wikipedia.org/