UNIVERSIDAD MICHOACANA DE SAN NICOLAS DE HIDALGO

DIVISION DE ESTUDIOS DE POSGRADO

FACULTAD DE INGENIERIA QUIMICA

USO DE REDES NEURONALES PARA CONTROLAR LA TEMPERATURA EN UN REACTOR BATCH

TESIS presentada por:

FRANCISCO JAVIER SANCHEZ RUIZ

A la División de Estudios de Posgrado de la Facultad de Ingeniería Química como requisito parcial para obtener el grado de:

MAESTRO EN CIENCIAS

EN

INGENIERIA QUIMICA

Morelia, Mich. Abril 2007

RESUMEN

USO DE REDES NEURONALES PARA CONTROLAR LA TEMPERATURA EN UN REACTOR BATCH

Por

Francisco Javier Sánchez Ruiz

Abril del 2007

Maestro en Ciencias en Ingeniería Química

Asesor: Dr. Luis Ignacio Salcedo Estrada

Coasesor: Dr. Agustín Jaime Castro Montoya

Las redes neuronales son una alternativa para sistemas del tipo batch (por lotes), teniendo un entrenamiento adecuado para la comprensión total de la dinámica del proceso. En el presente trabajo se da una alternativa en el Neurocontrol para un sistema de dos reacciones paralelas, formándose un producto no deseado que depende directamente del incremento de temperatura, siendo esta la variable a controlar en el proceso. La red propuesta para el control es una red neuronal del tipo NARMAX- L2 que proporciona el simulador comercial de Matlab en su librería de Simulink con una estructura de la red neuronal con 4 capas ocultas; además de una capa de entrada y una capa de salida, con 19 neuronas en cada capa interna de la red neuronal, delimitada a la entrada entre un intervalo de temperatura de 60 a 95 ºC; siendo una red neuronal del tipo recurrente, con un entrenamiento del tipo acelerado Levenberg-Marquardt. El error presente con este tipo de entrenamiento nos evita que la red neuronal presente sobreaprendizaje. El análisis del control con la red neuronal, despliega resultados mejores en comparación con el control basado en un Modelo Genérico de Control (GMC por sus siglas en inglés) y el convencional proporcional integral derivativo (PID) en un intervalo de 20-120 ºC.

A MIS PADRES.

A TI MADRE:

Por la vida que me has dado, por el apoyo
Que me has dado incondicionalmente,
Por los consejos, la paciencia que has tenido,
Para ti madre este simple trabajo que no es nada,
Pero eres parte de el, pues me has enseñado
Que luchando por las cosas que uno quiere
Las puede alcanzar, quisiera dedicarte mi vida
Como tu me la has dedicado, Gracias.

A TI PADRE:

También padre te agradezco por la vida y doy
Gracias a dios por haberme dado un padre
Tan ejemplar, con tu ejemplo y apoyo has sido parte
Fundamental de mi formación, tú me has dado consejos
Y me has enseñado que la paciencia es una virtud que
Tú practicas con el ejemplo, eres parte de cada palabra
Escrita aquí, Gracias.

CONTENIDO

Lista de Figuras

Lista de Tablas

AGRADECIMIENTOS

Debo agradecer en primer lugar a Dios por prestarme vida y alcanzar una meta, a mi asesor y coasesor de este trabajo: Dr. Luis Ignacio Salcedo Estrada y Dr. Agustín Jaime Castro Montoya, gracias por cada una de las aportaciones que me sirvieron para la realización de este trabajo.

Agradezco también a los Drs. Medardo Serna González y al Juan Gabriel Segovia Hernández por sus aportaciones hacia este trabajo, así mismo quiero agradecer de manera especial al M. C. Rodolfo Ruiz Hernández por sus comentarios para este trabajo, y por sus consejos durante mi periodo de estudios de Licenciatura y de Postgrado, gracias ustedes son parte de este trabajo.

A la Universidad Michoacana de San Nicolás de Hidalgo, por haberme albergado durante todo este tiempo, orgullosamente Nicolaita, a la Facultad de Ingeniería Química y a la División de Estudios de Postgrado por haberme formado en una carrera tan satisfactoria, Gracias.

A mis compañeros de Postgrado Teoxahual, Gladis, Xiomara, Martín, Beatriz, Pedro, gracias por haber compartido todos esos momentos difíciles y divertidos.

A mis amigos que han estado conmigo en las buenas y en las malas ustedes saben que son parte de esto, y de manera especial a ti Eunice sabes que lugar ocupas, Gracias.

Por ultimo a las personas mas importantes de mi vida, a ustedes hermanos tengo mucho que agradecerles, en especial a ti Pedro Sánchez me has enseñado muchas cosas y te lo agradezco, a ustedes dos mis hermanas que son las mujeres que me hacen luchar, a mi cuñada y a mis sobrinos por darme fuerzas y enseñarme que uno puede lograr lo que quiere.

Aquí termina una página de mi vida,
Habrá de darle la vuelta y comenzar una
Nueva en donde nuevos sueños y nuevas
Metas se vean posibles de alcanzar.
Todo es posible con perseverancia y fe.

CAPITULO 1.
INTRODUCCIÓN

Se establecen las generalidades sobre los reactores químicos, y la importancia de trabajar en el control de estos sistemas altamente no lineales. Así también se define el objetivo a alcanzar con el trabajo, planteándose la hipótesis a comprobar

1.1 Generalidades

El control de procesos tiene una función vital en el avance de la ingeniería y de la ciencia, ya que permite la producción con la confiabilidad y calidad que las altas exigencias del proceso y del mercado continuamente imponen.

En lo referente a la industria química, el valor radica principalmente en la importancia de controlar variables de estado como: presión, temperatura y flujos, siendo estas la principales variables manipulables para el control.

De los diversos equipos con los que se cuenta en las plantas de proceso son los reactores químicos, sin lugar a dudas, los que representan un mayor reto para poder implementar un sistema de control, ya que son sistemas altamente complejos.

Dentro de la gran variedad de reactores químicos, los de tipo por lotes (batch), constituyen un ejemplo de reactores donde su modelo matemático exhibe no linealidad. El modelo de dichos reactores se asemeja bastante al modelo obtenido en reactores continuos de mezcla completa (CSTR, en sus siglas en ingles), aunque con la limitante de que en los reactores por lotes no existen salidas de producto, por lo cual su complejidad hacia el control se ve incrementada (Galván et al, 1999).

Para obtener los parámetros de los controladores de estos sistemas altamente no lineales, algunos investigadores han propuesto que el proceso se caracterice mediante un modelo de primer orden o de segundo orden con tiempo muerto (Smith y Corripio, 1996).

Sin embargo hay muy pocas propuestas sobre lo referente a la sintonización del control de temperatura de reactores por lotes ya que no es un problema trivial, y la utilización de métodos tradicionales genera respuestas pronunciadamente oscilatorias e inestables en muchos de los casos creando así un problema de sintonización muy complejo en comparación con otros reactores (Luyben, 1998).

En la industria química, para este tipo de reactores los controladores más comúnmente usados son los controladores humanos, pues requieren de cierta experiencia para su control. Usando control automático, los controladores usados son controladores clásicos como el control proporcional integral (PI) o proporcional integral derivativo (PID), pero en gran medida los parámetros de ajuste para el controlador son obtenidos industrialmente con sintonización empírica, a fin de alcanzar coeficientes razonables de amortiguamiento a lazo cerrado, por no contar con métodos convencionales para su determinación (Galván et al, 1999).

Para estos equipos, el objetivo primordial es mantener la conversión en un valor deseado; sin embargo esto se realiza normalmente manteniendo el control de la temperatura. Cuando la temperatura esta bien controlada, las variaciones en la conversión del reactivo van muy de acuerdo con la variaciones de la temperatura. El control de la conversión no es tan convencional, en muy pocos casos se maneja como variable a controlar por su alto costo de control.

Para el presente trabajo, se han elegido dos reacciones irreversibles y exotérmicas de primer orden (Macchietto y Cott, 1989).

1.2 Objetivo

Diseñar un sistema de control basado en el uso de redes neuronales, para el control de temperatura de un reactor por lotes.

1.3 Justificación

En la actualidad el control de procesos es una rama de la ingeniería en donde se está desarrollando el mayor número de investigaciones enfocadas a lo que es control de procesos continuos. Sin embargo, también son importantes los procesos por lotes, ya que son procesos altamente no lineales y su solución en cuanto al control no ha sido resuelto en su totalidad (Galván et al, 1999), por lo que es necesario plantearse nuevos caminos para poder controlar estos sistemas con mayor precisión.

Expuesto lo anterior, se debe plantear nuevas alternativas tecnológicas, tal como el uso de redes neuronales, una nueva propuesta novedosa para el control de

un reactor por lotes, y generar una nueva vertiente de investigación en lo referente a los procesos por lotes.

Puesto que la modelación matemática de un reactor por lotes está basado en la formulación de balances de energía y masa, escritos en función de ecuaciones algebraicas diferenciales, deben ser capaces de predecir el comportamiento dinámico del proceso por lotes.

1.4 Alcance

Se plantea la posibilidad de que el diseño del controlador usando redes neuronales sea capaz de estabilizar el reactor por lotes, manifieste suficiente robustez ante cambios en el punto de referencia (set point), haciendo mas fácilmente su control y mejorar el control convencional.

1.5 Hipótesis

Se considera que los controladores que hacen uso de redes neuronales son una alternativa real para el control de reactores químicos por lotes utilizando un entrenamiento de la red neuronal de acuerdo al comportamiento dinámico del proceso y que mejore el control convencional.

CAPITULO 2

ANTECEDENTES

El control de procesos mediante un enfoque de los sistemas expertos, como las redes neuronales, han cobrado un mayor interés en la actualidad. Los sistemas altamente no lineales no han sido resueltos en su totalidad, tal es el caso de sistemas del tipo batch (por lotes), enfocándose actualmente parte del estudio de control a estos procesos.

2.1 Reactores por lotes.

Un reactor batch, también conocido como intermitente o por lotes, tiene la característica de que una reacción química se efectúa mediante la adición de los reactantes dentro del mismo y éstos se dejan reaccionar hasta que ha pasado un tiempo determinado en el cual se calcula que el avance ha sido lo suficiente para poder descargar la mezcla totalmente. Siempre un reactor intermitente es el primer equipo con el que se encuentra para una reacción, a veces por la carencia de material para la experimentación y muchas de las veces por el tamaño pequeño de la muestra y buen control de la reacción en el laboratorio o en planta. Un reactor por lotes no tiene entrada ni salida de reactantes o productos mientras se está realizando una reacción (Figura 2.1).

Figura. 2.1 Reactor por lotes nivel laboratorio

2.2 Redes neuronales

Las Redes Neuronales Artificiales (RNA) surgieron de la observación del funcionamiento del cerebro y de su comparación con la forma de trabajar de los

ordenadores digitales. Los elementos estructurales constituyentes del cerebro, las neuronas biológicas (Ramón y Cajal, 1911), son unas seis órdenes de magnitud más lentas que las puertas lógicas de silicio, ofreciendo tiempos de respuesta del orden del milisegundo frente a los vertiginosos tiempos de respuesta del orden del nanosegundo que tienen ciertos dispositivos digitales actuales.

Esta relativa lentitud no impide al cerebro realizar ciertas funciones habituales, como son las tareas de reconocimiento, percepción y control motriz, con una eficacia y rapidez fuera del alcance del mejor ordenador digital. Esta supremacía le viene dada por la alta complejidad de su estructura, formada por un elevado número de células nerviosas (neuronas) masivamente interconectadas y funcionando en paralelo. Se estima que el número de neuronas en el córtex humano es del orden de diez millares de millones (10^{10}), y que el número de sinapsis o conexiones es del orden de 60 billones ($60x\ 10^{12}$). La alta complejidad de esta estructura biológica no sería operativa si no fuese además eficiente. La eficiencia energética del cerebro es aproximadamente de 10^{-16} J por operación por segundo, mientras que los mejores ordenadores no superan los 10^{-6} J por operación por segundo (Faggin, 1991).

Al nacer, el cerebro ya tiene gran parte de su estructura formada, y lo que es más importante, está dotado de los mecanismos necesarios para construirse sus propias reglas a través de lo que conocemos como “experiencia”. Este proceso de aprendizaje se extiende a lo largo de los años, aunque el desarrollo más espectacular tiene lugar durante los dos primeros años de vida, durante los cuales se forman alrededor de un millón de sinapsis por segundo. La adaptación del sistema nervioso a su entorno sigue dos mecanismos distintos en un cerebro adulto: la creación de nuevas conexiones sinápticas y la modificación de las conexiones existentes.

Podemos pues concluir que el cerebro humano es una estructura compleja, no lineal y paralela de tratamiento de información que almacena su conocimiento en las conexiones que ligan a sus elementos de proceso (neuronas), y que es capaz de adaptarse a su entorno. Las RNA están inspiradas de la estructura del cerebro, y fueron concebidas para resolver cierto tipo de problemas especialmente mal resueltos por las técnicas de programación tradicionales. Formalmente, computación neuronal es la disciplina tecnológica que trata sistemas paralelos y adaptativos de procesamiento de información distribuida, que desarrollan sus capacidades bajo su exposición a un entorno de información.

Esta definición muestra claramente el paralelismo entre la estructura cerebral biológica y las RNA. Este paralelismo ha motivado a muchos investigadores de diversos campos de la ciencia a profundizar en la interpretación biológica de las RNA, proponiendo nuevas estructuras conexionistas y estrategias de aprendizaje directamente inspiradas de la modelización del cerebro. Estos estudios han dado resultados interesantes en el campo de las RNA, pero desde un punto de vista subjetivo, lo más importante de ellos es el conocimiento que directa o indirectamente está aportando para esclarecer los complejos procesos de aprendizaje, percepción y gestión del conocimiento que tienen lugar en el cerebro humano.

Una definición más concreta de RNA es la propuesta por Robert Hecht Nielsen (Hecht-Nielsen, 1990), y que dice así:

“Una RNA es una estructura de procesamiento paralelo de información distribuida, bajo forma de grafo orientado (Figura 2.2), con las siguientes subdefiniciones y restricciones:

• Los nodos del grafo son llamados *elementos de proceso* o *neuronas*.

• Las uniones del grafo son llamadas *conexiones* (unidireccionales e instantáneas).

• Se admite cualquier número de conexiones de entrada.

• Cada elemento de proceso sólo puede tener una señal de salida, que puede ramificarse en cualquier número de conexiones de salida.

• Los elementos de proceso pueden tener *memoria local*.

• Cada elemento de proceso tiene una *función de transferencia* que puede utilizar y alterar la memoria local, puede usar las señales de entrada, y produce la señal de salida. La función de transferencia puede actuar de forma continua o discreta en el tiempo. Si actúa de forma discreta, existirá una entrada de estímulo que active la aplicación de la función de transferencia proveniente de la unidad de control de la RNA”.

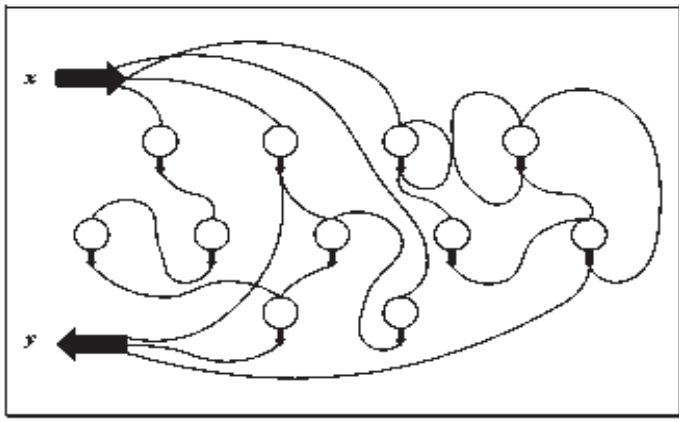


Figura 2.2 Estructura general de una RNA

La Figura 2.3 muestra el esquema general de un elemento de proceso, que consta de dos funciones, una de transferencia y una memoria local, donde la función de transferencia es la parte principal del modelo neuronal para el proceso.

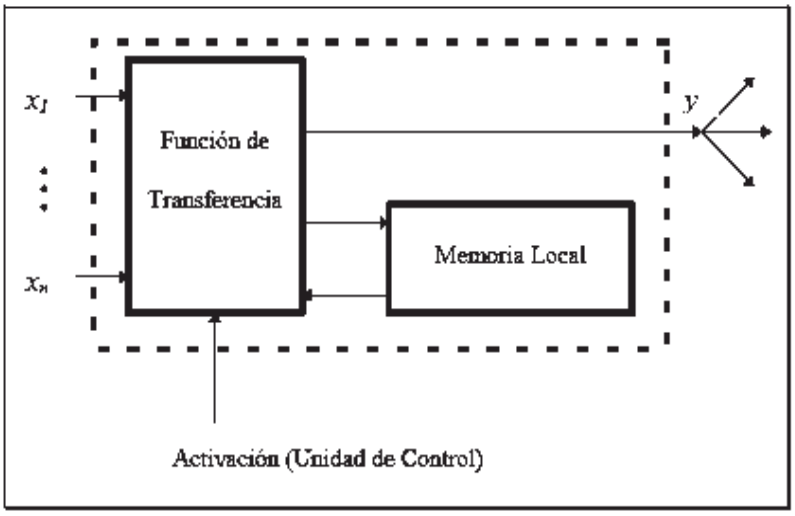


Figura 2.3 Estructura de un elemento de proceso

Se puede concluir que las RNA son estructuras adaptativas de procesamiento de información inspiradas en la estructura cerebral, donde el procesamiento se lleva a cabo mediante la interconexión de elementos de proceso muy sencillos a los que se llaman neuronas. Esta arquitectura da lugar a estructuras altamente paralelizables donde el flujo de información no sigue un camino secuencial, sino que se distribuye a través de las conexiones de los elementos de proceso donde la información es tratada.

Durante la fase de aprendizaje, la RNA se expone un entorno de información para que pueda adaptar sus pesos (parámetros libres que dan forma a las funciones de transferencia de sus elementos de proceso) y posiblemente también su estructura.

Durante la fase de evaluación, los pesos de la red se mantienen fijos y la RNA se limita a tratar la información de entrada que se le suministra.

Breve recorrido histórico

A continuación se muestra la evolución del estudio de redes neuronales (Anderson, 1999):

Los comienzos

- S. XIX Se plantea a nivel teórico la posibilidad de modelar la fisiología del cerebro, teniendo como objetivo la creación de un modelo capaz de reproducir procesos del razonamiento humano.

- 1943 Warren Mc Culloch y Walter Pitts publican la formalización de la descripción de una neurona artificial que sigue siendo hoy en día elemento constitutivo de las RNA más complejas. No contemplaron sin embargo aplicaciones prácticas.

- 1949 Donald Hebb publica "La organización del comportamiento", donde afirma que el condicionamiento psicológico es una propiedad de las neuronas a nivel individual. Propone una ley de aprendizaje que le permitió explicar cualitativamente algunos ejemplos experimentales de carácter psicológico.

- 1951 Marvin Minsky construye el primer neurocomputador ("Snark"). Resultó un éxito desde el punto de vista técnico, pues ajustaba automáticamente sus pesos, pero no llegó a darle ninguna aplicación práctica.

Primeros éxitos:

- 1957-1958 Frank Rosenblatt, Charles Whightman y otros, construyen el primer neurocomputador ("Mark I Perceptron") con aplicación práctica al reconocimiento de patrones. Rosenblatt publica "Principios de Neurodinámica".

- 1960 Bernard Widrow y Marcian E. Hoff desarrollan la ADALINE y la dotan de una potente ley de aprendizaje. Su aplicación práctica fue extensa. Widrow funda también la "Memistor Corporation".

Etapa de crisis

La falta de rigor analítico que se produjo al basar todos los trabajos en la mera experimentación, unido a un entusiasmo descontrolado, fueron el origen de esta etapa.
- 1969 Minsky y Papert publican "Perceptrons", donde se prueba que un perceptrón no puede resolver problemas linealmente no separables como es el caso de la función XOR. Llevan a cabo una verdadera campaña anti RNA.

Momentos de silencio

- 1967-1982 La investigación en este campo en los EEUU se ve acallada por el libro "Perceptrons". Sin embargo se consolidan importantes pilares de las RNA bajo los nombres de proceso adaptativo de señales, reconocimiento de patrones y modelado biológico. Pertenecen a esta época Shun-ichi Amari, James Anderson, Kunihiko Fukushima, Stephen Grossberg, Harry Klopf, Teuvo Kohonen y David Willshaw.

El despegue

- Principios de los 80.- Se produce el resurgimiento de la computación neuronal en EEUU, en parte financiado por la "Defense Advanced Research Projects Agency" (DARPA).
Por otro lado Hopfield, físico de gran reputación, se dedica entre 1983 y 1986 a relanzar la computación neuronal mediante publicaciones y conferencias.
- 1986 Publicación de "PDP Books" (Parallel Distributed Processing, Vol. I and II) editados por David Rumelhart y James Mc Clelland que supone un verdadero acontecimiento por la presentación del método de retropropagación (*"backpropagation"*).
- 1987 IEEE International Conference on Neural Networks con 1700 participantes (San Diego). Se crea la International Neural Networks Society (INNS)
- 1988 Publicación de la revista "Neural Networks" por el INNS.
- 1989 Publicación de la revista "Neural Computation"
- 1990 Publicación de la revista "Transactions on Neural Networks" por el IEEE.

Ventajas de las RNA

Las RNA deben su capacidad de procesamiento de información a su estructura distribuida y paralela (la información queda almacenada en los elementos de proceso de la red de forma no centralizada) y a su capacidad de aprendizaje y por tanto de generalización (en contraposición con la memorización).

Estas dos capacidades de procesamiento hacen que las RNA sean capaces de resolver cierto tipo de problemas muy complejos (tanto por su tamaño como por su estructura) que hasta el momento no habían quedado resueltos de forma satisfactoria.

Hay que dejar claro que las RNA no son una nueva metodología de ingeniería para la resolución global de problemas. Son herramientas de tratamiento de información que pueden integrarse fácilmente en arquitecturas modulares, para resolver de forma muy eficaz aquellas subtareas precisas del problema global que mejor se adaptan a sus capacidades. Entre estas subtareas cabe citar procedimientos de reconocimiento de patrones, de aproximación funcional y de memorias asociativas.

Las propiedades o características de las RNA que suelen ser más útiles son:

- No linealidad: las neuronas son elementos de proceso generalmente no lineales. La interconexión de estos elementos genera estructuras de transformación de datos donde este carácter no lineal queda distribuido a lo largo y ancho de la red. Esta característica permite modelar procesos intrínsecamente no lineales (como por ejemplo el reconocimiento del discurso hablado) pero complica también los métodos de análisis de las estructuras resultantes, impidiendo la aplicación de técnicas de análisis bien establecidas como son las de los sistemas lineales.

- Modelado de relaciones de entrada/salida: un paradigma de aprendizaje especialmente extendido y útil para el objetivo del llamado aprendizaje supervisado. En este tipo de aprendizaje se dispone de un conjunto de muestras de la relación entrada/salida a modelar, formado por pares (entradas, salidas deseadas), que permite optimizar los pesos de la red de tal forma que se espera que la relación de entrada/salida generada sea capaz de reproducir casos no representados en el conjunto de datos original. Esta capacidad de generalización se obtiene utilizando estructuras de aproximación funcional con capacidad de representación universal, y estrategias de aprendizaje, que cuidan de un modo especial el no sobrepasar el límite del sobre-entrenamiento.

- Adaptabilidad: las RNA son por definición estructuras adaptativas capaces de ajustar sus pesos, y por tanto su función de transferencia, a cambios en el entorno.
Estas características las hace particularmente útiles en el tratamiento de procesos no estacionarios, donde pueden diseñarse estrategias de aprendizaje en tiempo real para que el modelo conexionista se vaya adaptando de forma continua a los cambios del proceso en cuestión. En el diseño de sistemas adaptativos hay que tener siempre en cuenta las constantes de tiempo del sistema de adaptación y las del sistema bajo estudio: un sistema adaptativo con constantes de tiempo demasiado bajas puede responder demasiado rápido de tal forma que no sea capaz de ignorar perturbaciones espúreas y se haga inestable. En el caso concreto del diagnóstico basado en modelos conexionistas de funcionamiento normal esta capacidad de adaptación permitirá al sistema de diagnóstico el irse acomodando al lento proceso de envejecimiento de los componentes. Otros campos de aplicación relevantes son el reconocimiento adaptativo de patrones, el procesamiento adaptativo de señales y el control adaptativo.

- Respuesta evidencial: en el ámbito de aprendizaje supervisado (para aproximación funcional y clasificación), una RNA puede, además de estimar la salida deseada, dar una medida de la fiabilidad de la estimación. Esta información puede ser utilizada para rechazar patrones de entrada, completando de esta forma el proceso de estimación.

- Tolerancia frente a fallos: una red neuronal realizada en *"hardware"* tiene la capacidad de seguir respondiendo de forma no catastrófica cuando parte de su estructura está dañada. Esto es debido al tratamiento distribuido de la información y a la redundancia implícita en su estructura.

- Realización en VLSI: la naturaleza masivamente paralela de las RNA las hace potencialmente eficaces para la realización de ciertas tareas complejas. Esta misma característica, junto con la uniformidad de su estructura, las hace especialmente adecuadas para su construcción en tecnología de integración VLSI por sus siglas en ingles *"Very Large Scale Integration"*. Esto permite construir estructuras extremadamente complejas (en cuanto a su tamaño) que de otro modo serían inviables.

Estructura de la Neurona Artificial

Una neurona artificial es una unidad de procesamiento de información de redes neuronales. El modelo de neurona más conocido es de McCulloch-Pitts.

Puede observarse que N señales de entrada son representadas por las variables x_1 x_2... x_N las cuales están asociadas a pesos que son representados por las variables w_{ji} los cuales determinan el nivel de influencia de la neurona j para la neurona i.

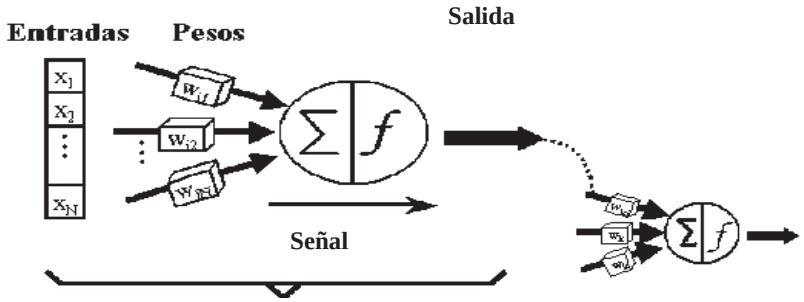


Figura 2.4 Estructura de una red neuronal

Existen dos etapas de procesamiento para cada neurona: suma y activación.

En la primera etapa, las señales de entrada x_j y los pesos w_{ij} son combinadas por el sumatoria:

$$y_i = \sum_{j=1}^{N} w_{ij} x_j \qquad (2.1)$$

Donde y_i es llamado estado interno de la neurona i. En la segunda etapa, la salida de la neurona es generada a través de la aplicación de una función llamada función de activación.

$$x_i = f(y_i) \qquad (2.2)$$

Donde la salida de la neurona es representado por x_i y f corresponde a la función de activación aplicada al estado interno de la neurona, que tiene como objetivo limitar el nivel de activación de entre [-1 ,1] o [0 1], en el caso de x_i sea un valor continuo y si x_i es discreto entonces el puede ser {-1,1} o {0,1}.

Existen varios tipos de función de activación. La Figura 2.5 muestra dos funciones de activación más usadas: la función de grado y la tangente hiperbólica. Como se vio en la Figura 2.4 la salida de una neurona puede ser la entrada de otra. Generalmente, una red neuronal se forma por muchas neuronas de alguna forma acoplados.

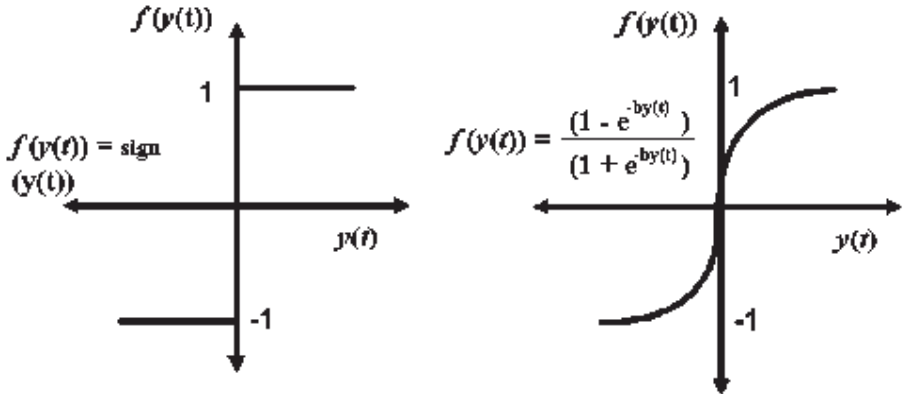


Figura 2.5 Funciones de activación

Principales estructuras de redes neuronales.

Dado el enorme abanico de estructuras conexionistas existentes, se va a limitar en este apartado a enumerar las más relevantes al enfoque de este trabajo, haciendo un especial hincapié en sus respectivos campos de aplicación (Maren et al, 1990).

Para cada una de ellas se especificarán sus creadores o desarrolladores más decisivos, su fecha de aparición o desarrollo, las referencias bibliográficas más relevantes, una breve descripción de su estructura, sus aplicaciones, y sus ventajas e inconvenientes.

Desde un punto de vista práctico, cabe mencionar el paquete de simulación de redes neuronales de MatlabTM, desarrollado por la compañía Math Works, Inc. (Demuth & Beale, 1992). Este paquete permite obtener, de una forma muy sencilla, una primera experiencia práctica en este campo, y viene dotado de una serie de programas de demostración que ilustran gráficamente las capacidades de esta herramienta.

A un nivel más avanzado existen varias herramientas de simulación de RNA de dominio público, generalmente desarrolladas en el seno de universidades europeas y americanas (Luzzy & Dengel, 1993). Entre estas herramientas cabe destacar la

desarrollada por la universidad de Stuttgart ("*Stuttgart Neural Net Simulator*") que está disponible vía *ftp* anónimo y funciona bajo entornos *X Windows*.

A continuación se presentan las estructuras neuronales más relevantes:

1) Perceptrón Multicapa: PM

(*"Multilayer Perceptron"*)

Creadores: P.J. Werbos
D. Parker
D. Rumelhart

Fecha: 1974-1986

Referencias: *(Werbos, 1974) (Werbos, 1988) (Werbos, 1989)*
(Parker, 1985) (Parker, 1987)
(Rumelhart et al., 1986a) (Rumelhart et al., 1986b)

Estructura:

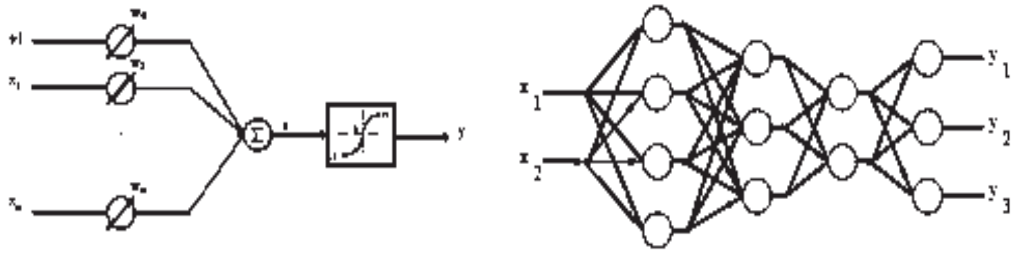


Figura 2.6 Preceptrón multicapa

Aplicaciones:
- Aproximación funcional
- Reconocimiento de patrones
- Filtrado de señales
- Eliminación de ruido
- Segmentación de imágenes y señales
- Control adaptativo
- Compresión de datos
- etc.

Ventajas: Capacidad de representación funcional universal. Gran rapidez de procesamiento. Genera buenas representaciones internas de las características de los

datos de entrada. Ampliamente estudiada. Es la estructura conexionista que más se ha aplicado en la práctica.

Desventajas: Tiempo de aprendizaje elevado para estructuras complejas.

2) Perceptrón Multicapa recurrente

Creadores: Almeida
Pineda

Fecha: 1987

Referencias: (*Almeida, 1987) (Almeida, 1988)*
(Pineda, 1987)(Pineda, 1988)

Estructura:

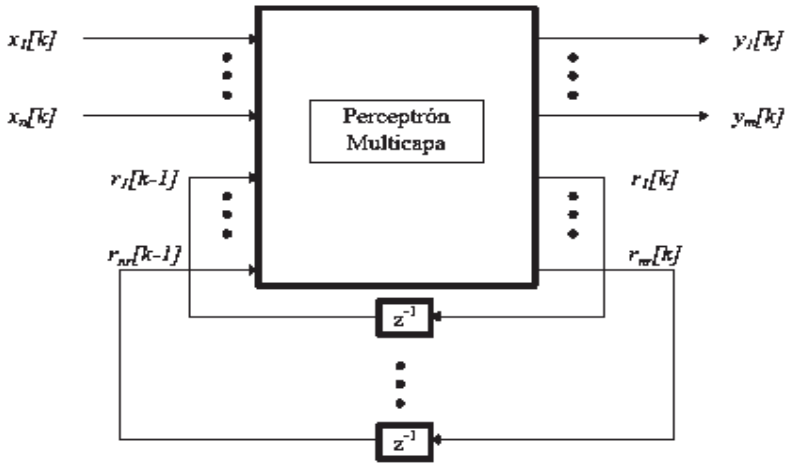


Figura 2.7 Preceptrón multicapa recurrente

Esta estructura recurrente permite estimar los valores de las salidas *y* en el instante *k*, a partir de los valores de las entradas externas *x* en el mismo instante *k*, y de los valores de lo que podríamos llamar variables de estado internas ***r*** en el instante *(k-1)*, entre las que podría haberse incluido algunas de las variables de salida.

Aplicaciones: **-** Control
- Reconocimiento del habla
- Predicción de secuencias

Ventajas: Capaz de tratar información temporal

Desventajas: Estructuras muy complicadas. El aprendizaje puede resultar muy difícil.

2.3 Control mediante redes neuronales (Neurocontroladores)

Neurocontroladores Indirectos.

- Controlador neuronal basado en modelo completo.
- Controlador neuronal basado en modelo paramétrico ó parcial.
- Modelo inverso neuronal.
- Parametrizador de controladores basado en Redes Neuronales Artificiales.

Neurocontroladores Directos.

- Modelado del controlador.
- Neurocontrolador libre de modelo.
- Neurocontrolador basado en modelo.
- Neurocontrol robusto.

Son tres familias de arquitecturas para la aplicación de las RNA en sistemas de modelado y control. En primer lugar se tienen las arquitecturas de modelado de sistemas, donde se utiliza una red neuronal para imitar el comportamiento de un elemento físico real. Con este objetivo se desarrollan cuatro propuestas.

En el esquema de neurocontroladores indirectos, la red neuronal no envía señales de control directamente al proceso. En lugar de esto, la red indica algunas características dinámicas de relevancia en el proceso. Este indicador puede ser un modelo que se comporta de forma similar al proceso ó bien un parametrizador que ajusta a un controlador produciendo configuraciones apropiadas basadas en el comportamiento de dicho proceso.

En el esquema de un neurocontrol directo, una red neuronal es empleada como un controlador de retroalimentación que envía señales directamente al proceso. Dependiendo del concepto del diseño, los elementos del neurocontrolador directo se categorizan principalmente atendiendo su dependencia al modelo del sistema.

Arquitecturas para el modelado de sistemas.

Existen principalmente cuatro arquitecturas de RNA para el modelado de plantas (Agarwal, 1997) como son: Modelado básico, modelado inverso de la planta, modelado inverso especializado y modelado del operador.

Para el modelado especializado inverso, el error se forma de la salida de la planta y la salida de la red es la entrada a la planta.

Modelado básico o directo de la planta se muestra en la Figura 2.8.

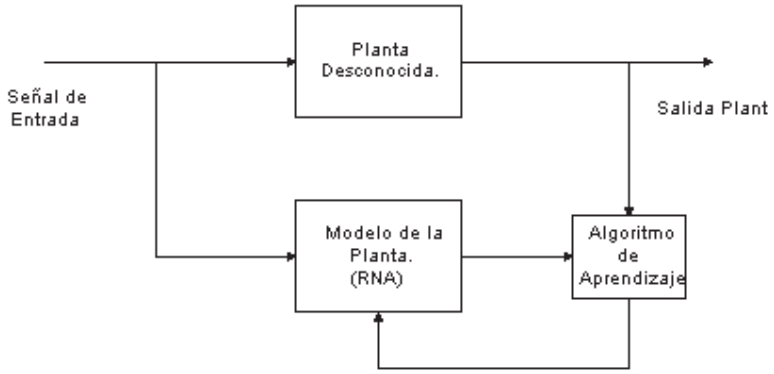


Figura 2.8 Estructura para el modelado básico o directo de la planta

Su arquitectura básica es muy simple, tanto la señal de control como los estados de la planta son muestreados y conforman el vector de entrada a la red. La salida de la red se calcula y se compara con las mediciones de la salida de la planta. La diferencia entre esas dos cantidades es usada para ajustar el vector de ajuste de pesos tendiente a reducir el error de la red.

Modelado directo inverso de la planta.

El objetivo con este modelado es formular un controlador tal, que junto con la arquitectura de la planta se conforme una sola función de transferencia. Inevitablemente los errores del modelado perturban dicha función, por lo que se recurre al uso de un precompensador en adición del controlador lineal, lo cual generalmente provee buen rendimiento para una amplia gama de plantas no lineales.

Para definir lo mejor posible el modelo inverso (Figura 2.9), los ejemplos de entrenamiento deberán ser únicos. Esto se cumple cuando la planta es invertible, es decir, que los datos de entrenamiento de la planta no lineal son contenidos en un área restringida del espacio de entradas.

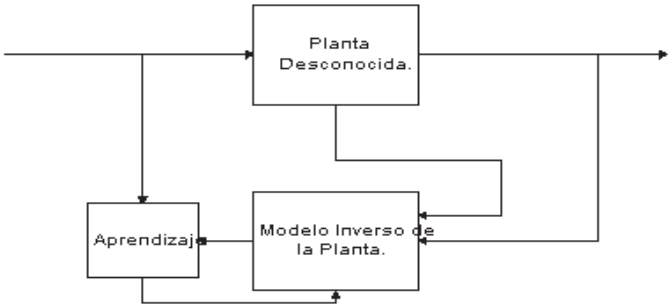


Figura 2.9 Modelado directo inverso de la planta

Modelado especializado inverso de la planta. Este tipo de modelado (Figura 2.10) utiliza una estructura similar al modelado inverso, donde existe una función de transferencia única, solo que el método propuesto es diferente (Agarwal, 1997). Un modelo de la planta básico es construido como primer paso, la diferencia entre la respuesta de la planta y la salida deseada es usada para generar la señal de error la cual es pasada hacia atrás desde el modelo básico con el objeto de ajustar los parámetros del nuevo modelo inverso.

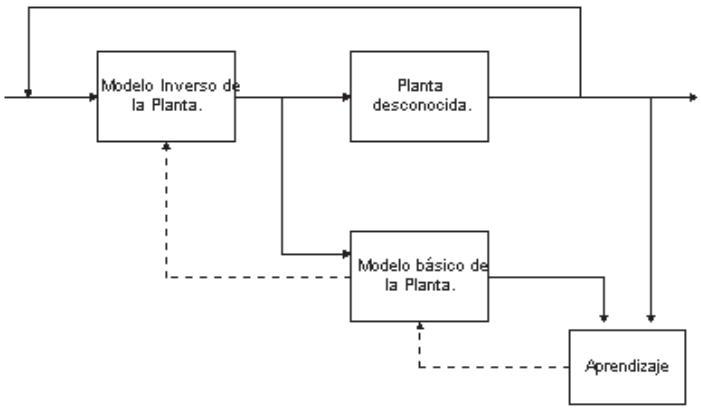


Figura 2.10 Modelado especializado inverso de la planta

Modelado de Operador.

Sintetizar un controlador por aprendizaje a partir de un experto tiene aplicaciones potenciales en el campo de la inteligencia artificial. El algoritmo de aprendizaje esta ejecutándose en paralelo con una planta conducida por un operador experto y sus respuestas forman la salida deseada de la red. (Figura 2.11) Estas

señales de entrenamiento típicamente contienen altas cantidades de ruido dado que el operador usa diferentes acciones para entradas similares, por lo que la señal deberá filtrarse antes de que los algoritmos convencionales de aprendizaje para la red sean aplicados.

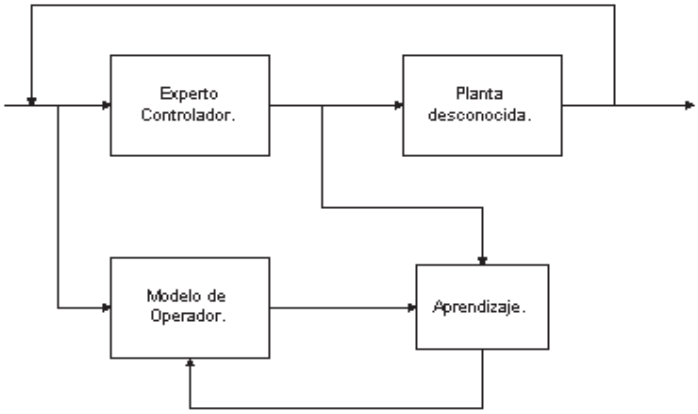


Figura 2.11 Arquitectura para el modelado del operador

Neurocontroladores Indirectos.

Control neuronal basado en modelo completo, la aplicación más popular de los sistemas de control basados en redes neuronales es utilizar una red como modelo de entrada y salida del proceso. Estos elementos utilizan aprendizajes manejados por lo datos supervisados como por ejemplo una red neuronal tratando de imitar un proceso existente a partir de la exposición a datos de dicho proceso. La estructura de modelo más comúnmente adoptada para este propósito es la Autoregresiva No Lineal de Promedio del Movimiento con Entradas Exógenas conocida como NARMAX ó su denominación más simple NARX (Galván y Zaldivar, 1999). Esta familia de modelos Narmax es una función de transferencia no lineal discreta en el tiempo (Galván y Zaldivar, 1999). Alternativamente, uno puede elegir identificar modelos continuos en el tiempo con una red neuronal como una red recurrente (Diron et. al., 1996). Esto conduce a una estructura del modelo y a una estrategia de control,

Cada modelo se desarrolla y luego puede implementarse un control basado en modelo. Sin embargo, en la etapa de implementación, el modelo de la red neuronal no puede ser usado solo. Este debe ser incorporado con un esquema de control basado en modelo. Dichos esquemas pueden ser diseño de auto ajustadores basados en

redes neuronales, modelado de controladores basados en modelo y el diseño de neurocontroladores basados en modelo.

Control neuronal basado en modelo parcial ó paramétrico.

En algunos casos, cierto grado de conocimiento sobre el proceso pudiera estar disponible, como puede ser la estructura del modelo ó un fenómeno físico particular bien entendido. En este caso, no es deseable un modelo de caja negra. Por ejemplo, si en la estructura del modelo están disponibles valores para los parámetros asociados, estos pueden ser determinados por una red neuronal. Ejemplo de esos parámetros pueden ser las constantes de tiempo, ganancias, retardos y parámetros físicos, rangos de difusión ó coeficiente de transmisión del calor, etc. (Maren et al., 1990) emplearon este modelo como un alimentador para un biorreactor. En otros casos donde la estructura del modelo fue especialmente conocida, las redes neuronales pueden ser utilizadas como un modelo parcial con el cual puede mejorar el modelo.

Control neuronal por modelo inverso.

Una red neuronal puede ser entrenada para desarrollar un modelo inverso de la planta. La entrada a la red es la salida del proceso, y la salida de la red es la correspondiente entrada al proceso (Figura 2.12).

Aplicaciones exitosas del modelado inverso son discutidas en las publicaciones hechas por Werbos en años recientes. Obviamente, un modelo inverso existe solamente cuando el proceso se comporta monótonamente como una función hacia adelante en el estado estable. Si no, este concepto es inaplicable. También pueden encontrarse unos cuantos artículos señalando conceptos similares al modelo inverso usando una red neuronal inversa no estática para el control (Werbos, 1988).

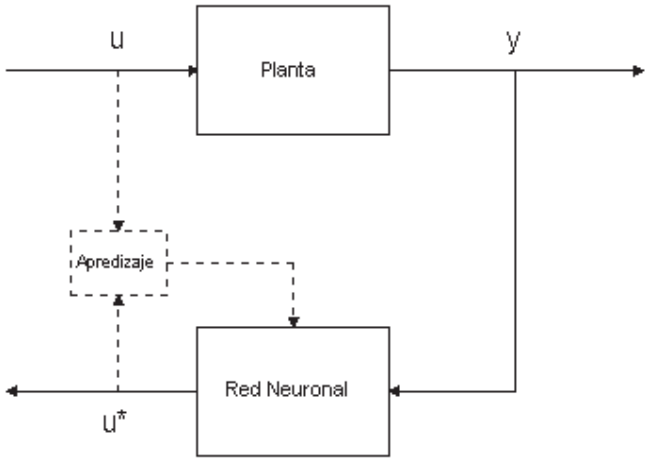


Figura 2.12 Arquitectura de control neuronal por modelo inverso

En principio, un modelo inverso basado en redes neuronales puede aprender la dinámica inversa bajo algunas restricciones (por ejemplo la fase mínima y causalidad son requeridas). En la práctica para modelos dinámicos de tipo discreto, el modelo inverso en especial puede no estar disponible para aprender la dinámica inversa deseada. Por ejemplo Psichogios y Ungar en 1999 entrenaron un modelo inverso y luego desarrollaron algunos casos de estudio. El modelo inverso fallo con referencia a las expectactivas que se tenían. En algunos casos el proceso inverso es un factor no causal aún cuando el proceso se comporta monótonamente. La no causalidad de un proceso inverso puede resultar del retardo de transporte ó de los tiempos muertos ó bien de la discretización de un proceso continuo en un sistema de datos muestreado. Si el modelo inverso existe, el uso del modelo dinámico inverso como controlador de retroalimentación no resultará en un sistema de control apropiado. Las propiedades estrictas son esenciales en el diseño de un sistema de control.

En algunos artículos de la literatura abierta, el modelado inverso se usa algunas veces para referirse al entrenamiento de un modelo de red neuronal en un ambiente de lazo cerrado (Mujtaba, 2004). Este modelado inverso no coincide con el modelo que invierte la función hacia delante de proceso/planta. Por ejemplo, el proceso opera como una función que direcciona las variables de entrada a variables de salida, mientras que el "modelo inverso de lazo cerrado" no direcciona las mismas variables de salida ó las mismas variables de entrada. Aún así, frecuentemente se direccionan las diferencias entre las variables de salida y el objetivo ó los puntos de operación hacia las variables de entrada. Desde la perspectiva de este capítulo, estos elementos

caen dentro de otras categorías para el diseño de neurocontroladores y serán discutidos en secciones posteriores.

Parametrizador neuronal de controladores.

Como en el caso anterior donde una red puede ser usada para estimar los parámetros de un modelo conocido, es posible utilizar esas mismas estructuras neuronales para estimar parámetros de ajuste para un controlador cuya estructura sea conocida a priori. El estimador de ajuste para los parámetros del controlador es conocido frecuentemente como un auto ajustador ó parametrizador. La entrada a la red puede comprimir datos del proceso muestreado ó características extraídas de este. Sin embargo esos parámetros no pueden determinarse únicamente desde las características del proceso. Dependen también de las características del sistema deseado para el control de lazo cerrado. Lo realmente atractivo de estas estructuras es que la red neuronal puede ser entrenada en simulación. El entrenamiento con datos reales del proceso no es necesario. Para un auto ajuste de lazo abierto, una simulación de lazo abierto es suficiente; de lo contrario una simulación de lazo cerrado es necesaria (Figura 2.13). El entrenamiento deberá ser conducido sobre un espacio de entradas que contenga distintos modelos del proceso. Idealmente, este espacio deberá cubrir todos los procesos que serán encontrados durante la fase de operación.

La mayoría de los trabajos realizados para auto ajustar ó parametrizar controladores se han desarrollado directamente para PID, debido a que actualmente son utilizados en una amplia gama de aplicaciones. Valores apropiados en las ganancias del PID son esenciales para que el sistema de lazo cerrado trabaje adecuadamente. El ajuste del PID es actualmente un procedimiento manual, a veces largo, y frecuentemente cae sobre la heurística desarrollada medio siglo atrás. Distintos auto-ajustadores están comercialmente disponibles, pero son aun necesarias algunas mejoras. Para el diseño de un auto-ajustador basado en redes neuronales, los modelos lineales de bajo orden con rangos adecuados para los parámetros, son generalmente suficientes para la mayoría de las aplicaciones que utilizan PID.

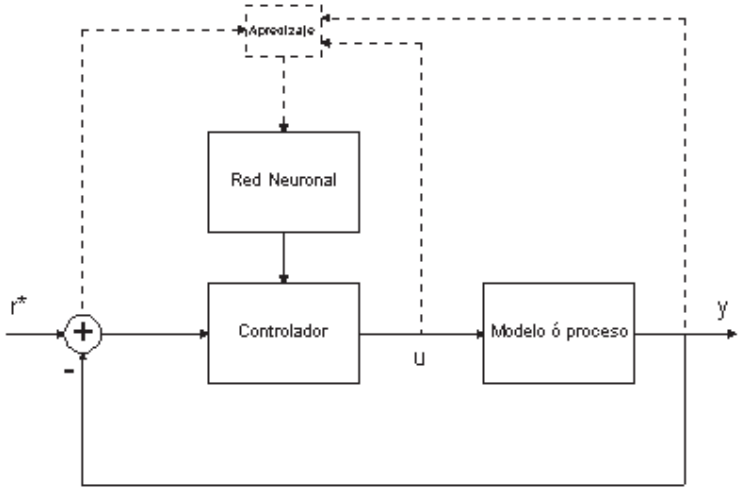


Figura 2.13 Parametrizador neuronal de lazo cerrado

Desarrollos de este concepto son discutidos por Swiniarski y Ruano en 2001. En el primero la red es entrenada usando un método clásico para determinar las ganancias de PID. Durante la operación, la entrada a la red son 128 muestras del proceso en lazo abierto. También el auto-ajustador resultante requiere la operación del proceso en lazo abierto. En contraste, Ruano en 2002 describe un método para el auto ajuste de lazo abierto ó lazo cerrado, acompañado por un preproceso de datos de entrada y de salida. Las ganancias óptimas para el PID en el proceso de entrenamiento son calculadas con un algoritmo de optimización de Newton, inicializado con valores determinísticos. El criterio de optimización es la minimización de la integral del tiempo multiplicada por el error absoluto (ITAE). Los autores demuestran como un auto-ajustador de lazo cerrado puede efectivamente adaptar los parámetros del PID en línea, en respuesta a cambios en el punto de operación para el lazo de control.

Neurocontroladores Directos.

Modelado del controlador. De los esquemas de neurocontrol directo que se presentan, él más simple de ellos consiste en usar una red neuronal para modelar un controlador existente (Figura 2.14). La entrada al controlador existente es la entrada de entrenamiento de la red y la salida del controlador sirve como la salida deseada para dicha red neuronal. Este elemento es similar al modelado de la red neuronal discutido en la sección Control Neuronal basado en modelo completo excepto por que

en aquel caso los valores deseados a la salida de la red no eran los valores del controlador

Como un modelo del proceso, el controlador es generalmente un sistema dinámico y frecuentemente contiene integradores y diferenciadores. Si una red algebraica de conexiones hacia adelante es usada para modelar el controlador existente, la información dinámica debe de ser explícitamente provista como una entrada a la red. Esta información dinámica puede ser dispuesta ya sea como integrales apropiadas y derivadas ó bien como señales empaquetadas de retardo con datos del proceso. Por ejemplo para modelar un controlador PID, una red neuronal algebraica requiere no solamente el error instantáneo entre el punto de operación y la salida del proceso, requiere además la integral y la derivada del error. Alternativamente, uno puede entrenar una red neuronal con series de esos errores y/ó las salidas del controlador en el tiempo previo. Esta ultima afirmación es similar a la desarrollada para las redes RAX (Red Autoregresiva con entradas exógenas) excepto porque que las entradas y las salidas del proceso se reemplazan con errores de retroalimentación y salidas de controlador.

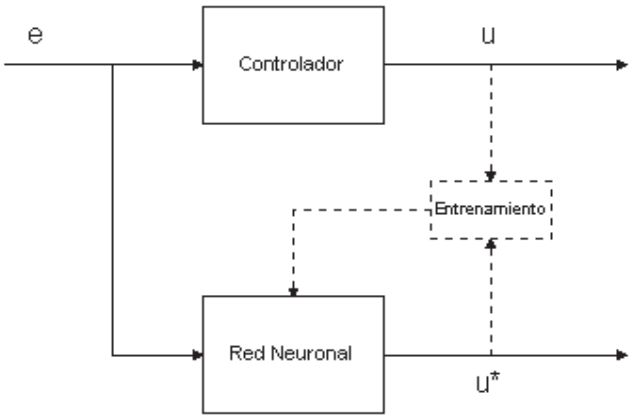


Figura 2.14 Estructura para utilizar una red neuronal para modelar un controlador existente

En general este tema puede resultar en controladores que son rápidos y/ó baratos en comparación con los controladores existentes. Por ejemplo usando este tipo de controladores Pomerleau en 1999 presenta una aplicación intrigante donde una red neuronal es usada para reemplazar a un controlador existente: el operador humano. La universidad de Carnige Mellon equipó una camioneta con una

videocámara y un reconocedor de caminos láser, operándola tal como lo haría un conductor humano a 6 millas por hora durante 5 minutos. Con entradas sensoriales adicionales ha sido posible alcanzar nuevas capacidades como supresión de colisiones ó navegación nocturna.

Mientras los beneficios de este tipo de controladores son aparentes cuando se trata del controlador humano, su utilidad puede ser limitada. Es aplicable solamente cuando un controlador existente está disponible, lo cual no sucede en muchas aplicaciones. Una red neuronal se entrena para comportarse como un controlador existente y luego se refina en coordinación con el modelo del proceso.

Neurocontrolador libre de modelo.

Ante la ausencia de un controlador existente, algunos investigadores han desarrollado la opción para que un controlador aprenda como actuaría un controlador humano con poco ó nada de conocimiento detallado sobre la dinámica de proceso. También se ha intentado diseñar controladores que por adaptación y aprendizaje puedan resolver problemas de control difíciles en la ausencia de los modelos del proceso ó de experiencia humana al respecto. La clave en esta estructura de control adaptivo es que el modelo del proceso no es conocido previamente y tampoco es muy explícito durante el diseño del controlador. Esta técnica para el diseño de controladores es frecuentemente denominada como aprendizaje reforzado. Sin embargo, a esta clase de controladores como neurocontroladores libres de modelo, ya que desde la óptica particular es más apropiado en el contexto. La Figura 2.15 es una representación clásica de esta clase de controladores.

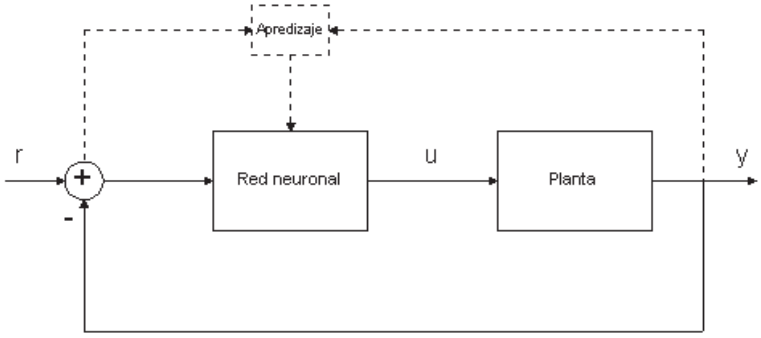


Figura 2.15 Arquitectura para el diseño de controladores libres de modelo

El primer trabajo en esta área fue el algoritmo de adaptivo crítico propuesto por Werbos. Este algoritmo puede ser visto como una versión aproximada de la programación dinámica. En este trabajo se desarrolla el problema del péndulo invertido mostrando el concepto del diseño de control (Werbos, 1988). En esta clase de diseño la información es limitada ó pobre, y es frecuentemente adoptada como un indicador para los criterios rendimiento del sistema. Por ejemplo, el objetivo del problema del péndulo invertido es simplemente mantener el péndulo cercano a su posición vertical balanceada. La retroalimentación instruccional es limitada a una señal de falla cuando el controlador no logra mantener el péndulo en la posición vertical. El problema de balancear el péndulo invertido ha llegado a ser popular como cama de prueba para explorar nuevos conceptos de diseño para controladores libres de modelo.

La solución propuesta por Werbos en 1988 fue construir un esquema de control el cual esta compuesto por dos elementos adaptivos: un Elemento de Búsqueda Asociativa (ASE por sus siglas en ingles) y un Elemento Crítico Adaptivo (ACE por sus siglas en ingles). El Elemento de Búsqueda trata de reproducir la señal de control optimo que satisface los objetivos de rendimiento definidos, mientras que el Elemento Crítico trata de monitorear el rendimiento interno del controlador y proveer una señal de reforzamiento interna la cual es usada para entrenar al ASE por sus siglas en ingles como se ve en la Figura 2.16.

El ACE por sus siglas en ingles es entrenado utilizando señales de fallas/éxitos externas. Este entrenamiento interno continuo del elemento de control pretende implementar mejoras en el rendimiento de todo el sistema.

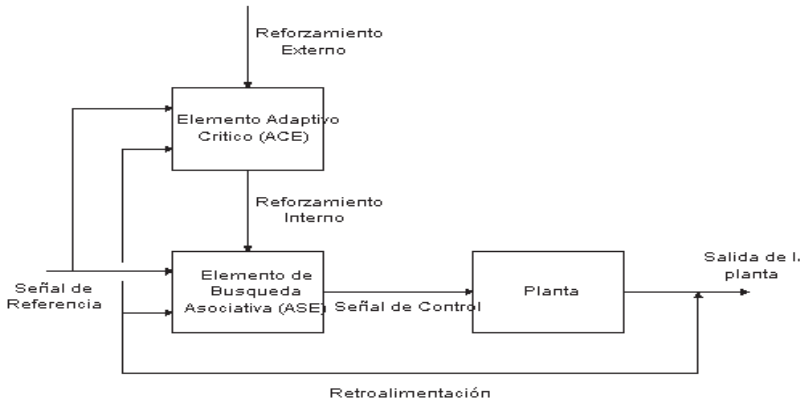


Figura 2.16 Arquitectura de aprendizaje reforzado planteada por Werbos, utilizando el concepto de ACE y ASE.

Aún con su importancia histórica y sus conceptos intuitivos, los neurocontroladores adaptivos libres de modelo no son apropiados para la mayoría de las aplicaciones del mundo real. La planta deberá de pasar por estados fuera de control durante el proceso de aprendizaje y pocos procesos industriales pueden tolerar esa larga cantidad de fallas, necesarias para adaptar el controlador.

Neurocontrolador basado en modelo

Desde una perspectiva práctica, se prefiere dejar las fallas para un ambiente simulado utilizando un modelo más que una planta real, aún cuando las fallas no sean desastrosas ó no causen pérdidas sustantivas en las etapas tempranas del aprendizaje neuronal. Esta clase de neurocontroladores es llamada frecuentemente "neurocontroladores basados en modelo".

Si el modelo del proceso no es disponible, uno puede primero entrenar una segunda red neuronal para modelar la dinámica de planta como se presentó en la sección de introducción. En el curso del modelado de la planta esta debe de ser operada "normalmente" en lugar de llevarla fuera de control. Después de la etapa de modelado, el modelo puede ser usado para el diseño de controladores. Si un modelo de la planta puede ser desarrollado, entonces en una simulación, en la cual las fallas no pueden causar ninguna pérdida más de allá del tiempo de cómputo, un controlador basado en redes neuronales puede instalarse en el sistema de control real, después de un extenso entrenamiento en alguna plataforma de simulación (Figura 2.17).

En efecto, estos diseños de neurocontroladores basados en modelo, no solamente han probado su eficiencia en distintos estudios (Diron et. al, 1996) sino que también ya han probado generar beneficios económicos notables. Dichos elementos pueden ser usados para diseños fuera de línea ó en adaptaciones en línea.

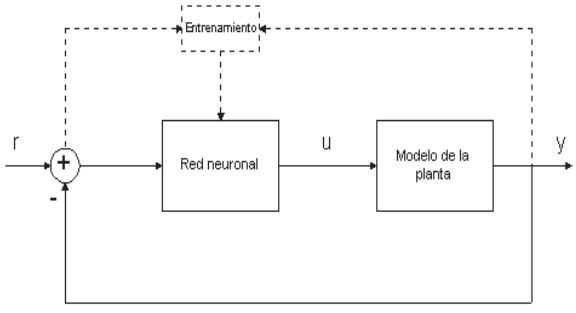


Figura 2.17 Un modelo reemplaza a la planta en el sistema de aprendizaje durante la fase de diseño del controlador

Demostraciones exitosas han sido desarrolladas para el clásico problema de control del vuelo multivariable. Quizás el mejor éxito comercial del neurocontrol a la fecha es aquella propuesta que trata de un horno de arco inteligente, el cual utiliza una red neuronal para regular la posición del electrodo en hornos eléctricos de arco. Publicaciones comerciales reportan ahorros típicos de sobre 2 millones de dólares por horno por tamaño (Sanz, 2005). El controlador inteligente del horno de arco incluye una interesante combinación para el desarrollo de un neurocontrolador. Inicialmente un controlador neuronal es entrenado para actuar como un controlador exitoso para la planta. Después de entrenar, la red neuronal reemplaza dicho controlador. En esta ultima etapa, una segunda red neuronal pre-entrenada se utiliza como modelo del proceso y el controlador neuronal continúa adaptándose en línea para contener las principales fallas de la planta.

Neurocontrolador robusto basado en modelo.

El tipo de neurocontrolador descrito en la sección pasada tiene todavía una desventaja común hasta ahora: Una red neuronal debe entrenarse cada vez que se tenga una nueva aplicación. El re-entrenamiento de la red es necesario aún cuando son pequeños los cambios del criterio de control, tanto para cambios en los pesos de la red relativos a la energía del control como para la respuesta de seguimiento, ó si el controlador va a utilizarse para procesos homólogos pero distintos. Con el objetivo de evitar estas desventajas, el concepto de controlador robusto es naturalmente incluido dentro del diseño de neurocontroladores. En el diseño de sistemas de control neuronal basados en un modelo robusto, se consideran familias de los modelos del proceso en lugar de un solo modelo. (Figura 2.18). Frecuentemente, una familia se agrupa por modelos que presentan los mismos niveles de ruido o bien pueden agruparse atendiendo el criterio del número de parámetros del proceso. Dos aspectos acerca de que tan robusto es un sistema se señalan frecuentemente: La estabilidad que refiere si un sistema es estable cualitativamente y el rendimiento robusto que opera con respecto a criterios cuantitativos (Werbos, 1989). Optimizar un controlador neuronal basado en un modelo preciso, puede alcanzarse un alto rendimiento, tanto como el proceso permanece invariable, pero con altos costos de fragilidad. Un procedimiento para el diseño robusto, por otro lado, no es capaz de alcanzar el mismo nivel de rendimiento nominal pero será menos sensible a estados obsoletos del proceso, ruido y otras fuentes de error en la relación modelo-proceso.

Esquema de control de aprendizaje predictivo

Los esquemas de control indirecto de aprendizaje predictivo buscan formular una estrategia tomando en consideración los efectos de sus acciones por muchos instantes de tiempo en el futuro y seleccionan la acción de control óptima, cuando es aplicada a la planta.

La arquitectura requiere el desarrollo de un modelo de la planta, una función de rendimiento para evaluar los efectos de cada acción de control y una técnica de optimización la cual puede determinar la mejor acción de control. Esto es ilustrado en la Figura 2.18, donde un elemento de control de aprendizaje está también incluido, para que después de suficiente entrenamiento, el cálculo total de la optimización no sea necesario y los recursos de cómputo puedan ser destinados a otras tareas.

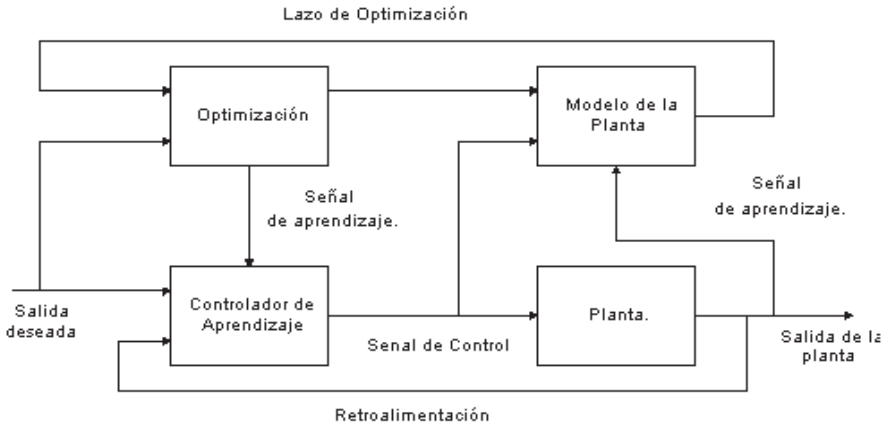


Figura 2.18 Arquitectura de control de aprendizaje predectivo

Si la planta es variable en el tiempo, el modelo es generalmente adaptivo. Si el modelo de proceso es pobre, es posible que el controlador ofrezca un control de lazo cerrado muy pobre.

Neurocontrolador por modelo adaptivo de referencia.

La arquitectura de aprendizaje de modelo de referencia ha sido ampliamente usada en el campo del control lineal adaptivo, su estructura se esquematiza en la Figura 2.19. El objetivo del controlador es ajustar la señal de control de una forma estable para que la salida de la planta asintótica mente tienda a la salida del modelo de referencia. El rendimiento de este algoritmo depende de la elección de un modelo de referencia correcto y la derivación de un buen mecanismo de aprendizaje.

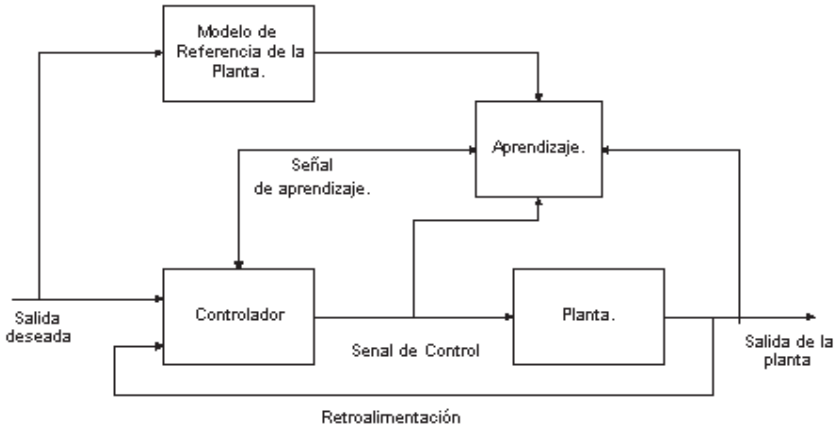


Figura 2.19 Arquitectura de control adaptivo por modelo de referencia

Control neuronal por modelo interno.

El controlador de modelo interno (Mujtaba, 2004), usa una estructura similar al esquema de control de aprendizaje predictivo (Figura 2.20). Un elemento de aprendizaje es usado para modelar el proceso directamente, recibiendo la señal de control en lugar de la señal de referencia. El error entre la salida del modelo y la respuesta de la planta es usado como señal de retroalimentación, la cual es alimentada al controlador.

Generalmente, el controlador de modelo interno se diseña como modelo inverso de la planta (cuando este existe), por lo que un esquema de modelado inverso descrito en puntos anteriores puede usarse para sintetizar el controlador adecuado.

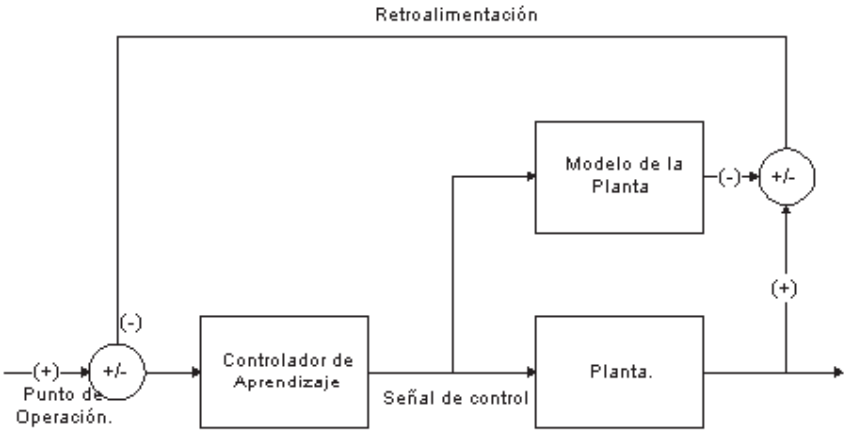


Figura 2.20 Neurocontrolador basado en un modelo interno

Muchos resultados teóricos acerca de la estabilidad de este tipo de controladores esta disponible en los demos de MathWorks, aunque generalmente son resultados para la estabilidad de lazo abierto, modelado exacto y/o modelado inverso.

Entrenamiento de una red neuronal

Las redes neuronales son modelos matemáticos diseñados para emular algunas capacidades del cerebro humano y están compuestos por muchos elementos computacionales simples llamados nodos o neuronas (que imitan el comportamiento de la neurona biológica), y que están altamente conectados entre si. Las redes neuronales pueden aproximar arbitrariamente bien cualquier función no lineal, dentro de un conjunto compacto y acotado, y son por tanto aproximadores universales (Cybenco, 1989).

Los modelos neuronales están especificados por una topología (diseño estructural de conexiones y nodos) de la red, características de los nodos, y reglas o algoritmos de entrenamiento o aprendizaje. El entrenamiento, utilizando datos de operación de un dado sistema, permite especificar el conjunto de pesos o conexiones sinápticas de la red, para dar a esta capacidad de representar al sistema en cuestión. La red constituye entonces un modelo empírico del sistema, que permite predicción de sus variables de salida, o en el caso de un sistema dinámico, la inferencia de su comportamiento a lo largo del tiempo. Esta habilidad de representar sistemas es una de las características más importantes de las redes neuronales y es utilizada para la identificación de sistemas y para la síntesis de controladores (Hunt et al., 1992).

Las redes neuronales pueden tener factores de pesos fijos y adaptables. Las que tienen pesos adaptables emplean leyes de aprendizaje para ajustar el valor de la fuerza de una interconexión con otras neuronas. Si las neuronas utilizan pesos fijos, entonces su tarea deberá estar previamente definida. Por otra parte, los pesos adaptables son esenciales si no se conocen previamente cual deberá ser su valor correcto.

El aprendizaje de la neurona lo podemos clasificar en dos tipos un aprendizaje supervisado y otro aprendizaje no supervisado, el primero ocurre cuando se le proporciona a la red tanto la entrada como la salida correcta, y la red ajusta sus pesos tratando de minimizar el error de su salida calculada. Este tipo de entrenamiento se aplica por ejemplo en el reconocimiento de patrones. El entrenamiento no supervisado se presenta cuando a la red se le proporcionan únicamente los estímulos, y la red

ajusta sus interconexiones basándose únicamente en estímulos y la salida de la propia red. Las leyes de aprendizaje determinan como la red ajustara sus pesos utilizando una función de error o algún otro criterio. La ley de aprendizaje adecuada se determina en base a la naturaleza del problema que se intente resolver (Hopfield, 1982).

La estructura de la red neuronal puede variar esto depende de las necesidades de control requeridas y de los datos con los que se cuenta para el entrenamiento de la red neuronal, las más comúnmente usadas son de dos tipos:

a) Redes Multicapas.

b) Redes Recurrentes.

a) Redes Multicapas.

Las redes multicapas, son las más conocidas y usadas, aun cuando existen redes que pueden representar de mejor manera la capacidad de conocer el funcionamiento dinámico de un proceso (Diron et al, 1996).

Las redes multicapas tienen la gran desventaja de que no se conoce la topología de las capas escondidas (Figura 2.21), siendo esto una búsqueda de la red óptima algo tedioso y lento, que debe hacerse, en general por ensayo y error. Esto para cuando no se tienen resultados reales de la planta, para este tipo de casos lo mas comúnmente usado para el entrenamiento o introducción de datos es el algoritmo de propagación hacia tras (backpropagation).

En este tipo de algoritmo, la red se interconectan varias unidades de procesamiento en capas, las neuronas de cada capa no se interconectan entre si. Sin embargo, cada neurona de una capa proporciona una entrada a cada una de las neuronas de las siguientes capas.

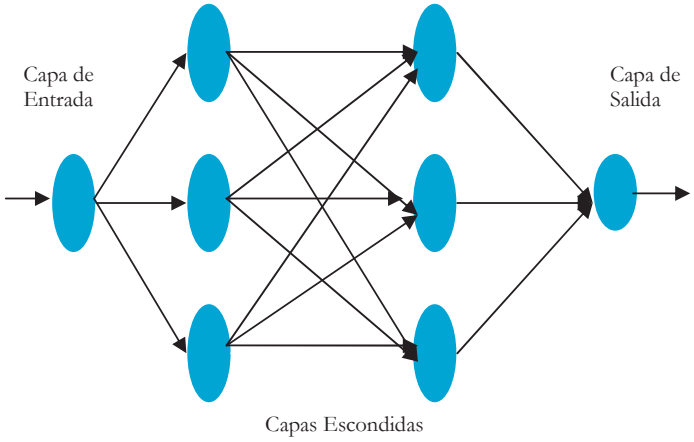


Figura 2.21 Arquitectura de red multicapa

Dirion et al, en 1996, sintetizaron una red neuronal usando el algoritmo de multicapas obteniendo resultados en el control de un reactor por lotes buenos, en donde la desviación con los resultados obtenidos de planta es muy pequeña.

Nahas et al, en 1992**,** sintetizaron una red neuronal multicapas, para el control de temperatura en un reactor CTRS por sus siglas en ingles, entrenando la red con datos de simulaciones y datos de planta, mostrando así una disminución del error y un mejor control de la reacción.

b) Redes Recurrentes.

Este tipo de redes son mas difíciles de estructurar y el entrenamiento mas lento, por esto ha sido mucho menos utilizadas, aunque para la estructuración de dichas redes se pueden estructurar de acuerdo a los resultados que deseemos obtener esto utilizando lazos de retroalimentación sobre los nodos internos (Drakopoulos et al, 2005).

Una de las ventajas de la redes recurrentes es que tienen una mejor capacidad predictora que la red multicapa, ya que su diseño es independiente al sistema y por lo tanto no acarrea la acumulación del error de ajuste y esto la hace mas adecuada para calcular la evolución temporal de las variables de salida de un proceso dadas sus entradas (Figura 2.22).

Una de las estructuras mas utilizadas en las redes recurrentes es la red recurrente diagonal (EDR por sus siglas en ingles).

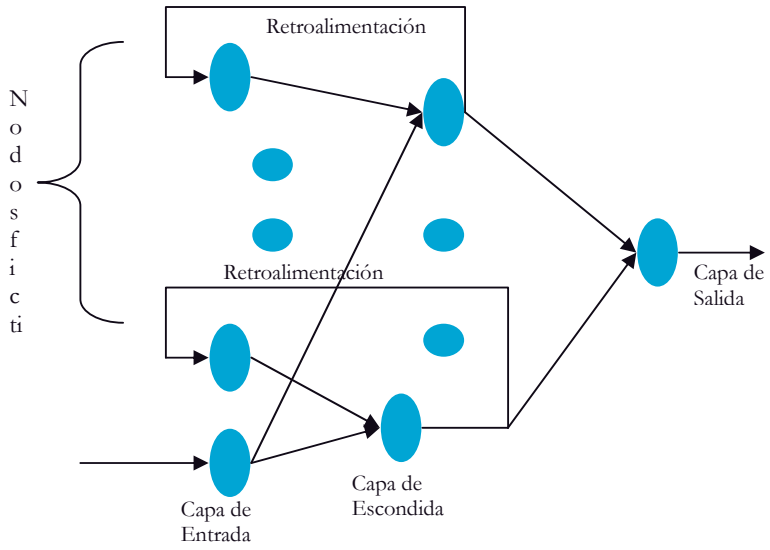


Figura 2.22 Red recurrente diagonal (EDR por sus siglas en ingles)

Luus, en 1994, sintetizó una red recurrente modificada para el control de pH en un reactor de saponificación, obteniendo buenos resultados comparados con trabajos semejantes en donde se utilizaron datos de planta y rede multicapas.

Daosud et. al., en 2005, han utilizado las redes neuronales recurrentes para el control de temperatura en procesos que presentan una reacción exotérmica y además para linealizar el modelo matemático obteniendo resultados muy satisfactorios y mejores que usando otro tipo de controlador.

Propagación hacia atrás.

Uno de los grandes avances logrados con la propagación hacia atrás es que esta red aprovecha la naturaleza paralela de la redes neuronales para reducir el tiempo requerida para el procesamiento secuencial para determinar lo patrones entre un punto dado. Además el tiempo de desarrollo de cualquier sistema que se este tratando de analizar se puede reducir como secuencia de que la red puede aprender el algoritmo correcto sin que alguien tenga que deducir por anticipado el algoritmo en cuestión.

La propagación hacia atrás es un tipo de red de aprendizaje supervisado, que emplea un ciclo de propagación – adaptación de dos fases una vez que se ha aplicado un patrón a la entrada de la red como estimulo, este se propaga desde la primera capa a través de las capas superiores de la red, hasta generar una salida. La señal de salida se compara con la salida deseada y se calcula una señal de error para cada una de las salidas.

Las salidas del error se propagan hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de la capa oculta que contribuyen directamente a la salida. Sin embargo las neuronas de la capa oculta solo reciben una fracción de la señal total del error, basándose aproximadamente en la contribución relativa que haya aportado cada neurona a la salida original. Este proceso se repite capa por capa, hasta que todas las neuronas de la red hayan recibido una señal de error que describa su contribución relativa al error total. Basándose en la señal de error que percibida, se actualizan los pesos de conexión de cada neurona, para hacer que la red converja hacia un estado que permita clasificar correctamente todos los patrones del entrenamiento.

La importancia de este proceso consiste en que, a medida que se entrena la red, las neuronas de las capas intermedias se organizan a si mismas de tal modo que

las distintas neuronas aprenden a reconocer distintas características del espacio total de entrada. Después del entrenamiento, cuando se les presente un patrón arbitrario de entrada que contenga ruido o que este incompleto, las neuronas de la capa oculta de la red responderán con una salida activa si la nueva entrada contiene un patrón que se asemeje a aquella característica que las neuronas individuales hayan aprendido a reconocer.

Varias investigaciones han demostrado que, durante el proceso de entrenamiento, la red de propagación hacia atrás tiende a desarrollar relaciones internas entre neuronas con el fin de organizar los datos de entrenamiento por clases. Esta tendencia se puede extrapolar, para llegar a la hipótesis consiste en que todas las unidades de la capa oculta de una propagación hacia atrás son asociadas de alguna manera a características específicas del patrón de entrada como consecuencia del entrenamiento. Lo que sea o no exactamente asociado puede no resultar evidente para el observador humano, lo importante es que la red ha encontrado una representación interna que le permite generar las salidas deseadas cuando se le dan las entradas, en el proceso de entrenamiento. Esta misma representación interna se puede aplicar a entradas que la red no haya visto antes, y la red clasificará estas entradas según las características que comportan con los ejemplos de entrenamiento (Galván, 1999).

El diseño de un controlador neuronal basado en el entrenamiento de propagación hacia atrás, puede considerarse como una arquitectura compleja en donde el principal factor de peso es el entrenamiento del neuron, en donde, se ha demostrado que un buen entrenamiento puede llevar a un tiempo de computo menor que en el caso de utilizar un entrenamiento diferente al usado con propagación hacia atrás.

El *Backpropagation* fue creado mediante la generalización de la regla de aprendizaje Widrow-Hoff para redes multicapas y las funciones de transferencia diferenciables no lineales. Los vectores de entrada y los correspondientes vectores deseados se usan para entrenar una red hasta cuando ella pueda aproximar una función, asociando los vectores de entrada con los vectores de salida específicos, ó clasificar los vectores de entrada de una manera apropiada tal como nosotros la definimos. Las redes con bases, una capa sigmoide, y una capa de salida lineal son competentes para aproximar cualquier función con un número finito de discontinuidades.

El backpropagation estándar es un algoritmo de gradiente descendente, como lo es la regla de aprendizaje Widrow-Hoff, en la cual los pesos de la red son movidos a lo largo del negativo del gradiente de la función de ejecución. El término *backpropagation* se refiere a la manera como el gradiente es calculado para redes multicapa no lineales. Existe un cierto número de variaciones en el algoritmo básico, las cuales están basadas en otras técnicas de optimización, tales como el gradiente conjugado y los métodos de Newton. Con el Toolbox de Redes Neuronales podemos implementar estas variaciones.

Las redes *backpropagation* entrenadas de manera apropiada, se orientan a dar respuestas razonables cuando se les presentan entradas que aún no han sido consideradas. Típicamente, una nueva entrada conduce a una salida similar a la salida correcta para vectores de entrada que se han empleado en el entrenamiento, los cuales a su vez son similares a las nuevas entradas que están siendo presentadas. Esta generalización con propiedad hace posible entrenar una red sobre un conjunto representativo de las parejas entrada / salida deseada, y se obtienen buenos resultados sin entrenar la red sobre todas las posibles parejas de entrada / salida. Existen dos características del Toolbox de Redes Neuronales las cuales están diseñadas para mejorar la generalización y regularización de la red y la detención remota.

2.4 Caja de herramientas de redes neuronales en Matlab.

Matlab cuenta con diferentes estructuras de redes neuronales para el uso en el control las cuales describiremos a continuación:

Modelo de control predectivo: este controlador usa una red neuronal de tipo de modelo de predicción a una futura respuesta a las señales del control. Tiene así mismo un algoritmo de optimización para el mejoramiento del modelo de la planta. La red neuronal es entrenada fuera de línea esto significa que dicho entrenamiento no requiere de datos provenientes de planta. Aunque en algunas de las ocasiones el algoritmo de optimización requiere de ajustes para su mejoramiento, este aplica el algoritmo de entrenamiento de propagación hacia atrás. En la caja de herramientas de simulink aparece de la manera mostrada en la Figura 2.23.

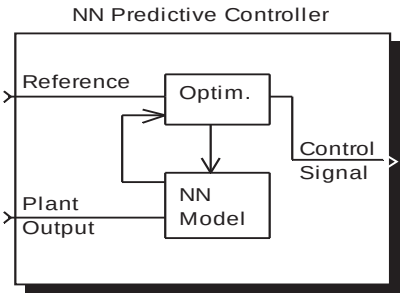


Figura 2.23 Controlador Predectivo

Control NARMA-L2: este controlador requiere de largos periodos de computación esto a consecuencia de la simple arquitectura de la red interna del controlador , el entrenamiento de esta red se lleva acabo fuera de línea aunque por prioridad es necesario conocer aunque sea una salida de la planta. Dicha estructura fue diseñada alrededor de los años 80, este tipo de control es más aplicado a controladores de tipo de entrenamiento supervisado. El control NARMA-L2 de la caja de herramientas se muestra en la Figura 2.24.

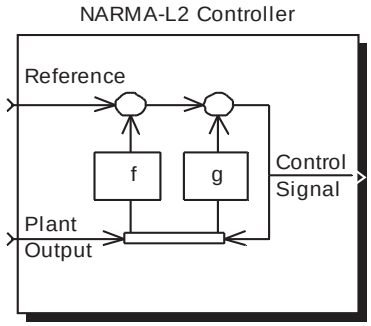


Figura 2.24 Control NARMA-L2

Control con un modelo de referencia: este tipo de control es semejante al control proporcionado por el control de NARMA-L2, aunque con la limitante de que el control con un modelo de referencia es muy poco usado en la industria debido a que necesita tener un modelo del proceso como referencia. El entrenamiento se hace fuera de línea como los dos mencionados anteriormente, este presenta un inconveniente debido a que también se debe hacer un entrenamiento para la red con el modelo de referencia. En la caja de herramientas de simulink aparece como la Figura 2.25 este tipo de control.

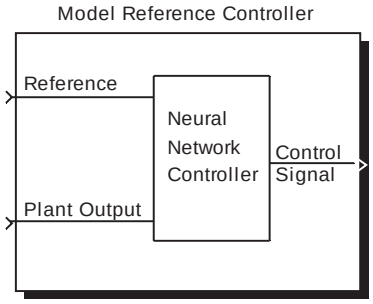


Figura 2.25 Control con modelo de referencia

Además Matlab cuenta con las funciones necesarias para generar una nueva red neuronal para la aplicación en cualquier campo.

CAPITULO 3

METODOLOGIA

En casi todos los reactores por lotes, cuanto más tiempo permanezca un reactivo en el reactor, más de él se convertirá en producto hasta que se llegue al equilibrio o bien se agote el reactivo. Por tanto, en los sistemas por lotes la conversión es función del tiempo que los reactivos permanecen en el reactor

3.1 Ecuaciones generales de balance

Para realizar un balance de masa en cualquier sistema, primero hay que especificar las fronteras del sistema. Se llama *volumen del sistema* al volumen encerrado entre dichas fronteras (Fogler, 2001). Se realiza un balance de moles de la especie *j* en un volumen del sistema, donde la especie *j* representa la especie química que interesa (Figura 3.1).

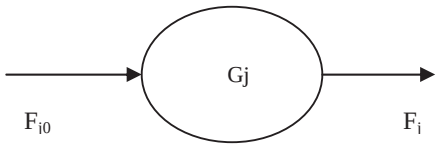


Figura 3.1 Balance en el volumen del sistema

Balance de masa

Un balance de masa de la especie *j* en cualquier instante da la siguiente ecuación:

$$\left[\begin{array}{l}\text{Velocidad de fujo}\\ \text{de j hacia el sistema}\end{array}\right] + \left[\begin{array}{l}\text{Velocidad de}\\ \text{generacion de j}\\ \text{por reaccion}\\ \text{quimica dentro}\\ \text{del sistema.}\end{array}\right] - \left[\begin{array}{l}\text{Velocidad de}\\ \text{flujo de j desde}\\ \text{el sistema}\end{array}\right] = \left[\begin{array}{l}\text{Velocidad de}\\ \text{acumulacio n de}\\ \text{j dentro del sistema.}\end{array}\right] \qquad (3.1a)$$

En un reactor por lotes de manera general se obtiene la siguiente ecuación:

$$\frac{dM_j}{dt} = r_j V \qquad (3.1b)$$

Donde:

M_j = Masa en el reactor del componente j (moles o kg).

t = tiempo (hr, min., o seg.).

V = volumen del reactor (m^3, ft^3)

r_j = rapidez de reacción del componente j.

Balance de energía

Planteando un balance de energía en estado no estacionario para un sistema abierto de n especies todas las cuales entran y salen del sistema con sus respectivas velocidades de flujo de entrada y con sus respectivas energías.

$$\begin{bmatrix}\text{Velocidad de}\\ \text{acumulacio n}\\ \text{de energia}\\ \text{dentro del}\\ \text{sistema}\end{bmatrix} = \begin{bmatrix}\text{Velocidad de}\\ \text{flujo de}\\ \text{calor del}\\ \text{entorno}\\ \text{al sistema}\end{bmatrix} - \begin{bmatrix}\text{Tasa de trabajo}\\ \text{efectuado por}\\ \text{el sistema}\\ \text{sobre el}\\ \text{entorno}\end{bmatrix} + \begin{bmatrix}\text{Tasa de adicion}\\ \text{de energia al}\\ \text{sistema por flujo}\\ \text{de masa hacia}\\ \text{dentro del sistema}\end{bmatrix} - \begin{bmatrix}\text{Tasa de perdida}\\ \text{de energia del}\\ \text{sistema por flujo}\\ \text{de masa fuera}\\ \text{del sistema}\end{bmatrix} \qquad (3.2a)$$

De manera general:

$$Q - W_s + \sum F_{i0} H_{i0} - \sum F_i H_i = \frac{d\hat{E}_{sis}}{dt} \qquad (3.2b)$$

Para un reactor por lotes tenemos la siguiente ecuación:

$$\frac{dT}{dt} = \frac{Q - W_s + (-\Delta H)(-r_i V)}{\sum N_i Cp_i} \qquad (3.3)$$

Donde:

T = temperatura dentro del reactor.

Q = calor añadido o removido.

W_s = trabajo efectuado sobre el sistema.

ΔH = entalpía de la reacción.

V = volumen del reactor.

r_i = rapidez de reacción del componente.

N_i = cantidad de masa o moles.

Cp_i = capacidad calorífica de cada especie.

3.2 Modelo matemático de un reactor por lotes

Considerando el siguiente caso de estudio para el control de temperatura en un reactor batch propuesto por Macchietto et. al., en 1989 (Figura 3.2).

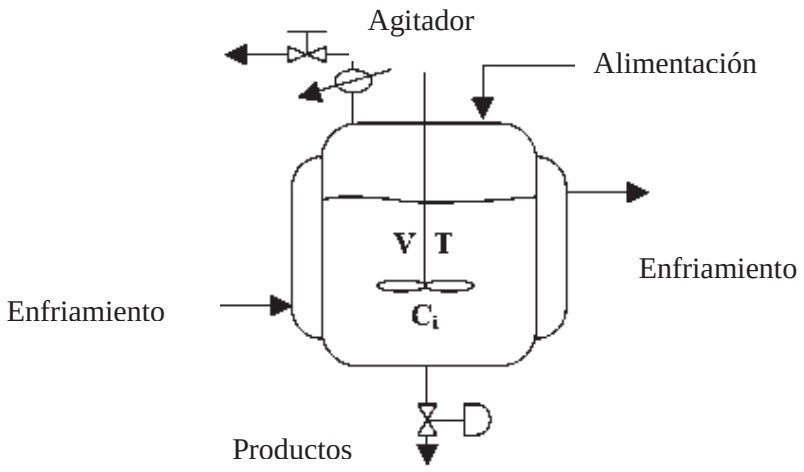


Figura 3.2 Reactor batch con enfriamiento

Siendo las siguientes reacciones que presenta el proceso:

$$A + B \rightarrow C$$
$$A + C \rightarrow D \qquad (3.4)$$

En donde A y B son reactivos, C y D productos, siendo el de mayor importancia el producto C, por consiguiente el producto D es no deseado.

Planteando balance de masa para cada componente, haciendo las siguientes consideraciones:

- Reacciones irreversibles.
- Reacciones elementales.
- Mezclado perfecto.
- Volumen constante.

Por tanto se obtienen los siguientes balances de masa y energía.

Balances de masa individuales

$$\frac{dM_A}{dt} = -R_1 - R_2 \qquad (3.5)$$

$$\frac{dM_B}{dt} = -R_1 \tag{3.6}$$

$$\frac{dM_C}{dt} = R_1 - R_2 \tag{3.7}$$

$$\frac{dM_D}{dt} = R_2 \tag{3.8}$$

Balance de energía

$$\frac{dT_r}{dt} = \frac{UA\left(T_j - T_r\right) - \Delta H_1 R_1 - \Delta H_2 R_2}{M_r Cp_r} \tag{3.9}$$

$$\frac{dT_j}{dt} = \frac{F_j \rho_j Cp_j \left(T_{jsp} - T_j\right) \pm UA\left(T_j - T_r\right)}{V_j \rho_j Cp_j} \tag{3.10}$$

Ecuaciones adicionales

$$R_1 = k_1 M_A M_B \tag{3.11}$$

$$R_2 = k_2 M_A M_C \tag{3.12}$$

$$k_1 = \exp\left(k_1^1 - \frac{k_1^2}{\left(T_r + 273.15\right)}\right) \tag{3.13}$$

$$k_2 = \exp\left(k_2^1 - \frac{k_2^2}{\left(T_r + 273.15\right)}\right) \tag{3.14}$$

$$M_r = M_A + M_B + M_C + M_D \tag{3.15}$$

$$Cp_R = \frac{Cp_A M_A + Cp_b M_B + Cp_C M_C + Cp_D M_D}{M_R} \tag{3.16}$$

Con los siguientes valores de las constantes (Tabla 3.1):

Tabla 3.1. Constantes para modelo.

Cp_A=75.31 kJ/(kmol ºC)	k_2^1= 38.9057	U=40.84 kW/(m^2 ºC)
Cp_B=167.36 kJ/(kmol ºC)	k_2^2= 17 000 K	ρ_j = 1000 kg/m^3
Cp_C=217.57 kJ/(kmol ºC)	ΔH_1=-41840 kJ/kmol	Cp_j = 1.8828 kJ/(kg ºC)
Cp_D=334.73 kJ/(kmol ºC)	ΔH_2=-25105 kJ/kmol	τ_i = 3
k_1^1=20.9057	ρ = 1000 kg/m^3	V_j = 0.6912 m^3
k_1^2= 10 000 K	r=0.5 m	F_j = 0.0058 kg/min.
W_r = 1560 kg	A = 6.24 m^2	

Con las siguientes condiciones iniciales M_{A0}=12, M_{B0}=12, M_{C0}=0, M_{D0}=0, T_{R0}=20, T_{j0}=20.

Resolviendo el modelo en Simulink (Figura 3.3).

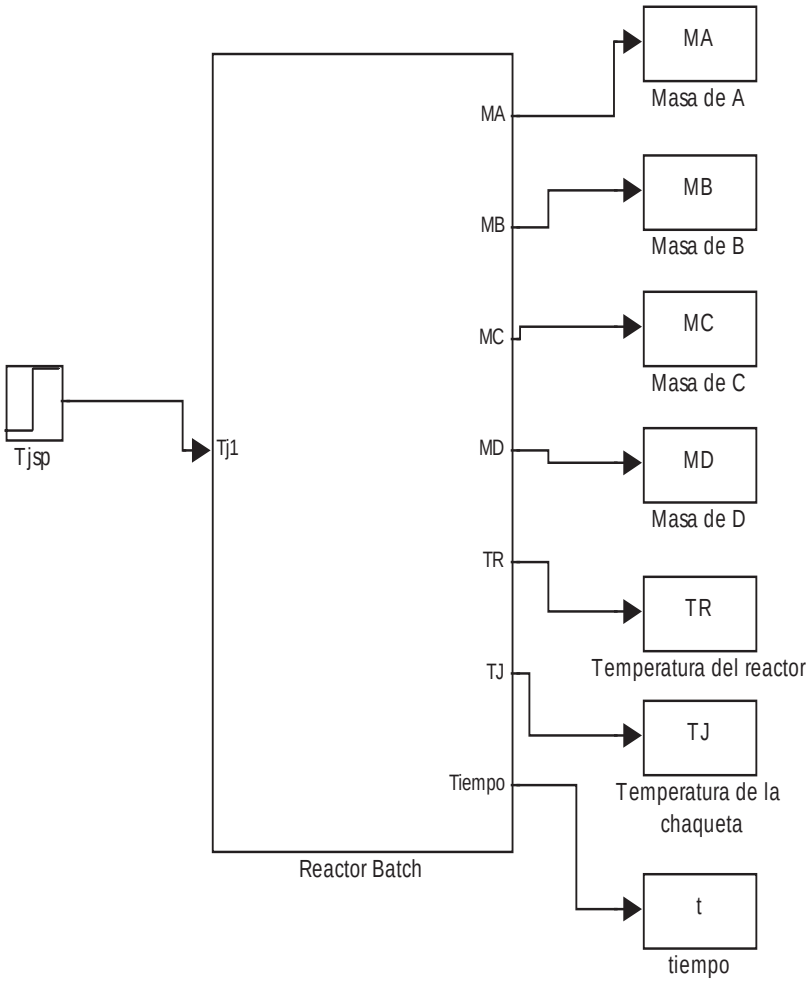


Figura 3.3 Diagrama enmascarado a lazo abierto de un reactor batch.

El diagrama de la solución no enmascarado es el mostrado en la Figura 3.4 para el simulador comercial Simulink de Matlab en su versión 7:

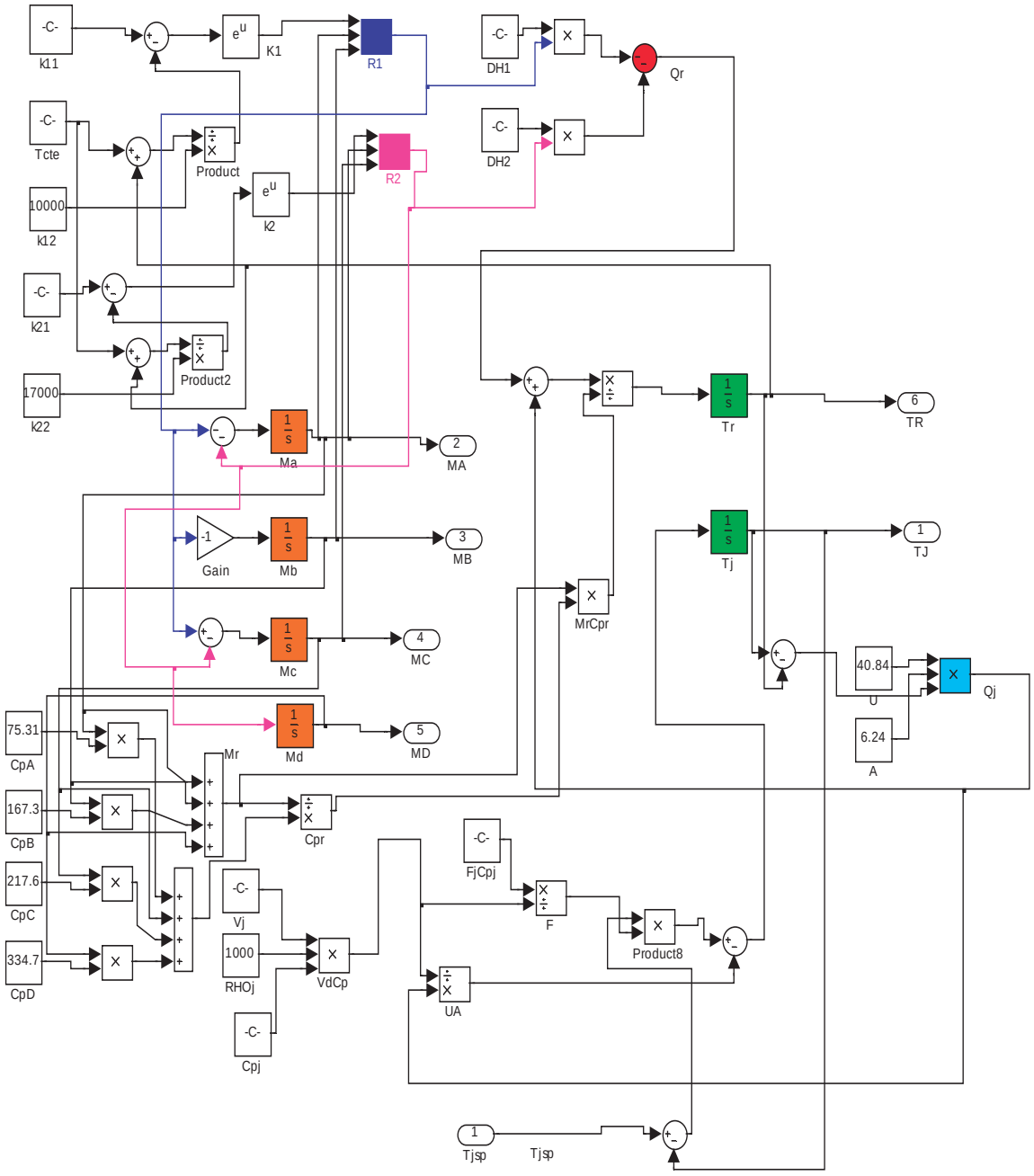


Figura 3.4 Diagrama a lazo abierto de un reactor batch en Simulink

Obteniendo los resultados para las temperaturas de la chaqueta y del reactor Figura 3.5.

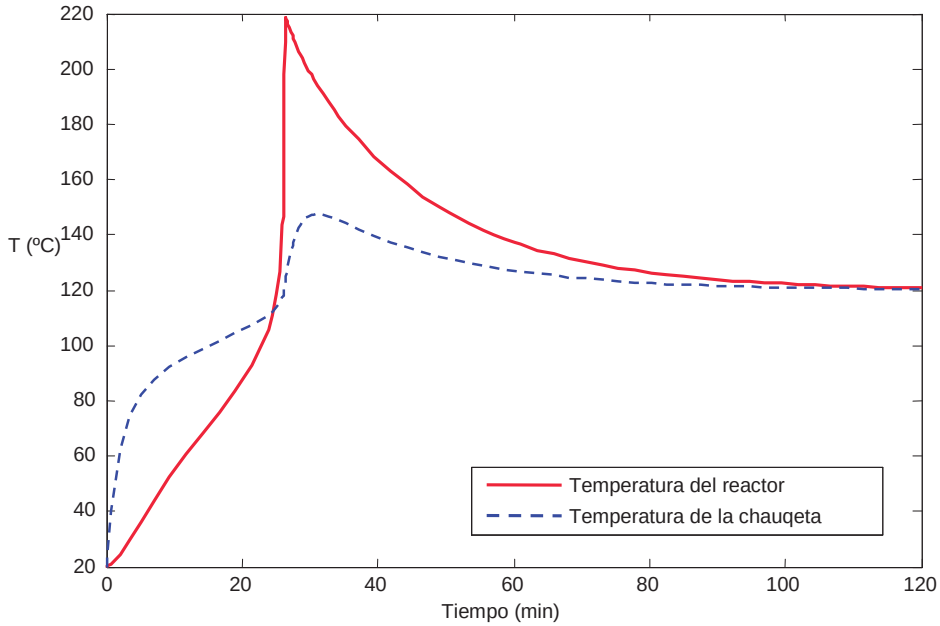


Figura 3.5 Perfiles de temperatura a lazo abierto

Observando el consumo de los reactivos y formación de los productos con respecto al tiempo (Figura 3.6 y 3.7).

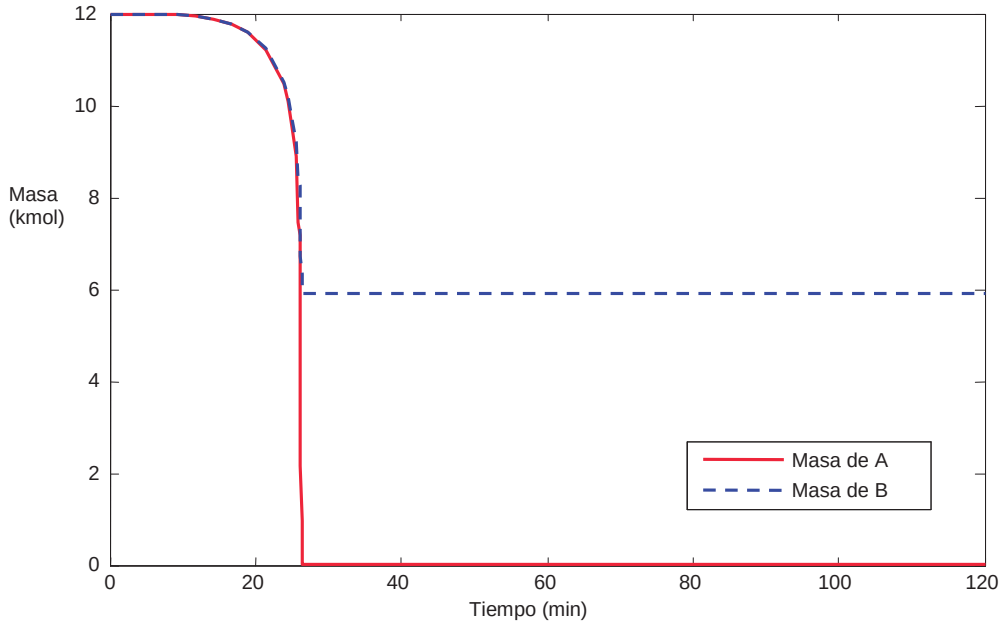


Figura 3.6 Perfiles de consumo de masa

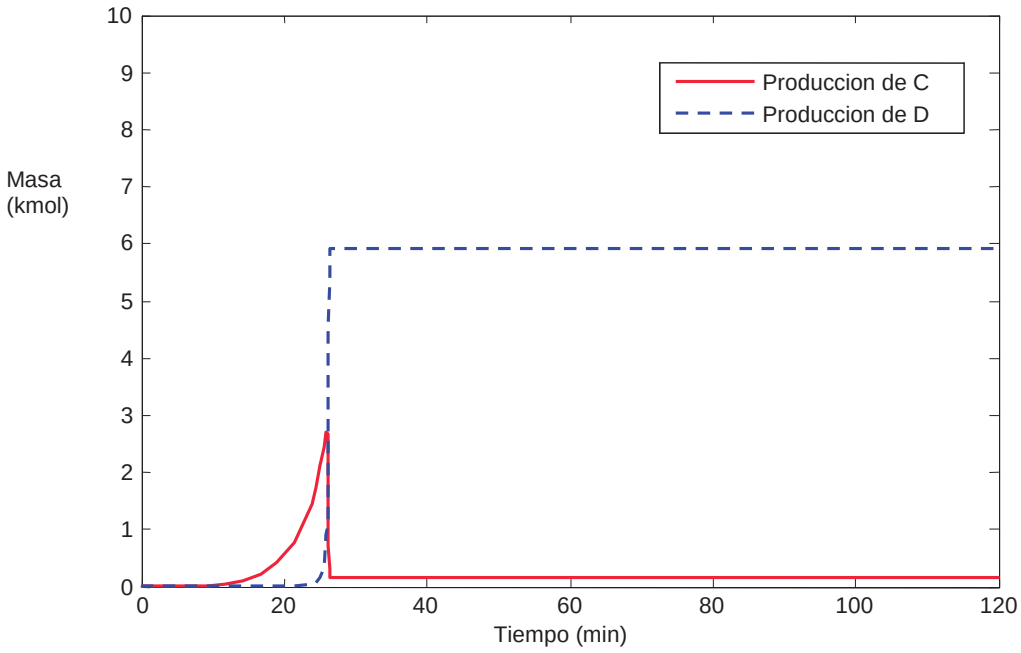


Figura 3.7 Perfil de producción

3.3 Control con Redes Neuronales.

Las redes neuronales se caracterizan por su habilidad de aprender de los ejemplos en lugar de tener que programarse en un sentido convencional. Su uso posibilita que la dinámica de los sistemas complejos sea modelada y un control preciso sea logrado a través del entrenamiento, sin tener información a priori sobre los parámetros del sistema.

Modelo general de neurona

Se denomina procesador elemental o neurona a un dispositivo simple de cálculo que, a partir de un vector de entrada procedente del exterior o de otras neuronas, proporciona una única respuesta o salida. Los elementos que constituyen una neurona son los siguientes:

- Conjunto de entradas
- Pesos sinápticos
- Regla de propagación.
- Función de activación.
- Función de salida.

De este modo la operación de una neurona i puede expresarse como:

$$y_i(t) = F_i\left(f_i\left[a_i\left(t-1\right), \sigma_i\left(w_{ij}, x_j(t)\right)\right]\right) \tag{3.17}$$

Este modelo de neurona formal se inspira en la operación de la biológica, en el sentido se integrar una serie de entradas y proporciona cierta respuesta, que se propaga por el axón.

Entradas y salidas.

Las variables de entrada y salida pueden ser binarias (digitales) o continuas (analógicas), dependiendo del modelo y aplicación. Por ejemplo, un preceptor multicapa (usado para este trabajo) admite ambos tipos de señales, así que para tareas de clasificación poseería señale digitales (0,1) mientras que para un problema de ajuste funcional de una aplicación multivariable continua, se utilizaría salidas continuas pertenecientes a un cierto intervalo.

Dependiendo del tipo de salida, las neuronas suelen recibir nombres específicos (Sanz, 2005). Así las neuronas convencionales cuya salida sólo puede tomar los valores de 0 o 1 se suelen denominar genéricamente neuronas del tipo McColloch-Pitts, mientras que únicamente pueden tener por salidas -1 o +1 se suelen denominar neuronas tipo Ising (debido al paralelismo con los modelos físicos de las partículas del spin que pueden adoptar únicamente dos estados de excitación). Si puede adoptar valores discretos en la salida, se dice que se trata de una neurona de tipo Potts.

Regla de propagación.

La regla de propagación permite obtener, a partir de las entradas y los pesos, el valor potencial postsináptico h_i de la neurona.

$$h_i(t) = \sigma_i\left(w_{ij}, x_j(t)\right) \tag{3.18}$$

La función más habitual es de tipo lineal, y se basa en la suma ponderada de las entradas con los pesos sinápticos.

$$h_i(t) = \sum_j w_{ij} x_j \tag{3.19}$$

Que formalmente también puede interpretarse como el producto escalar de los vectores de entrada y pesos.

$$h_i(t) = \sigma_i\left(w_{ij}, x_j(t)\right) = W_i^T . x \tag{3.20}$$

El peso sináptico w_{ij} define en este caso la intensidad de interacción entre la neurona presináptica j y la post sináptica i. dada una entrada positiva (procedente de un sensor o simplemente la salida de otra neurona), si el peso es positivo tendera a excitar a la neurona post sináptica, el peso es negativo tendera a inhibirla. Así se habla de sinapsis excitadoras (de peso positivo) e inhibidoras (de peso negativo).
Una regla de tipo lineal, de uso mas limitad, es la siguiente:

$$h_i(t) = \sum_{j1\,j2\ldots jp} w_{ij1\,j2\ldots jp} x_{j1} x_{j2} \ldots x_p \tag{3.21}$$

que implica una interacción de tipo multiplicativo entre las entradas de la neurona (como se ha observado realmente en determinadas sinapsis biológicas). El uso de esta última regla de propagación determina que una neurona se denomine de orden superior o neurona sigma-pi (Rumelhart,1986a), por emplear sumas y productos e implica una mayor complejidad, tanto en el estudio de la dinámica de la red neuronal.

Función de transferencia o función de activación.

La función de transferencia o activación proporciona el estado de activación actual $a_i(t)$ a partir del potencial postsináptico $h_i(t)$ y del propio estado de activación anterior $a_i(t-1)$

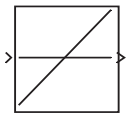
$$a_i(t) = f_i\left(a_i\left(t-1\right), h_i(t)\right) \tag{3.22}$$

Sin embargo, en muchos modelos de redes neuronales se considera que el estado actual de la neurona no depende de su estado anterior, sino únicamente del actual.

$$a_i(t) = f_i\left(h_i(t)\right) \tag{3.23}$$

La función de activación se suele considerar determinista, y en la mayor parte de los modelos es monótona creciente y continua, como se observa habitualmente en las neuronas biológicas.

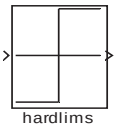
Alguna de las funciones de activación más comunes y usadas son las siguientes:


Identidad

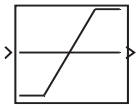
La más simple de todas es la función identidad (que se puede generalizar al caso de una función lineal cualquiera), empleada, por ejemplo en la Adelina.

El algoritmo es el siguiente:

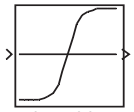
$$y(n) = n \tag{3.24}$$


hardlims

Es una función de transferencia simétrica que fuerza a la salida del neuron a dos tipos de respuestas entre el intervalo de 1 y -1 algunas de las veces este tipo de función puede llamarse de decisión.


Lineal a tramos

Esta función de transferencia calcula la salida de la red neuronal con solo conocer una de las entradas de la neurona teniendo un truncamiento en los umbrales de la excitación de la neurona en los puntos de 1 y -1.


Sigmoidea

La función sigmoidea es muy comúnmente usada en la redes de retropropagación ya que dan un intervalo mas amplio de excitación esto para alcanzar mas rápidamente los umbrales de la neurona.

Su algoritmo es el siguiente:

$$n = 2/(1+\exp(-2*n))-1 \tag{3.25}$$

Donde n es la entrada de la neurona o los pesos de entrenamiento de la neurona.

En le toolbox de Matlab dicha función de transferencia esta construida de la forma mostrada en la Figura 3.8:

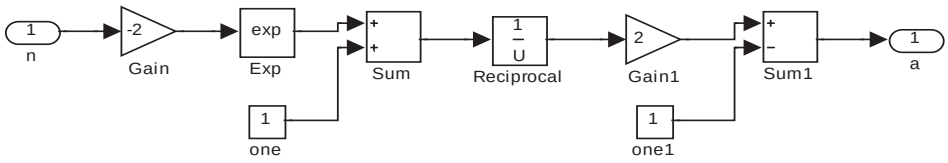


Figura 3.8 Estructura interna de la función sigmoidea

Las estructuras de la funciones de transferencia son las más usadas comúnmente existen mas funciones de transferencia que pueden ser usadas para la excitación de la neurona.

Con lo anterior podemos hacer un modelo general de una neurona estándar:

$$y_i(t) = f_i\left(\sum_j w_{ij} x_j - \theta_i\right) \tag{3.26}$$

En la ecuación anterior introducimos un término nuevo como lo es θ_i que s prácticamente el umbral de la neurona (el máximo en donde la neurona tiende a excitarse en exceso confundiendo a la neurona) y fi es la función de transferencia de la neurona para lograr la excitación de la misma.

Arquitecturas de redes neuronales.

Se denomina arquitectura a la topología, estructura o patrón de conexionado de una red neurona. En una sistema neuronal los nodos se conectan por medio de sinapsis, esta estructura de conexiones sinápticas determina el comportamiento de la red. Las conexiones sinápticas son direccionales, es decir, la información solamente puede propagarse en un único sentido. En general, las neuronas se suelen agrupar en unidades estructurales que se denominan capas.

Algunas formas generalizadas de la estructura de las redes neuronales son las redes monocapas que son aquellas compuestas por una única capa de neuronas. Las redes multicapa son aquellas cuyas neuronas se organizan en varias capas.

Así mismo atendiendo al flujo de datos en la red neuronal, podemos hablar de redes unidireccionales, la información va en un solo sentido desde la neurona de entrada

hasta la neurona de salida. En las redes recurrentes o retroalimentadas la información puede circular entre las capas en cualquier sentido, incluido el de entrada y salida.

Teniendo definida la estructura de la red neuronal, es necesaria que dicha red sea entrenada, algunos de los métodos más comunes de entrenamiento son los siguientes:

Entrenamiento de Hebb.

Se trata de uno de los modelos clásicos de entrenamiento de redes neuronales. Donald Hebb en 1949 postuló un mecanismo de aprendizaje para la neurona biológica, cuya idea básica consiste en que cuando un axón presináptico causa la activación de cierta neurona post sináptica, la eficacia de la sinapsis que las relaciona se refuerza (Sanz, 2005).

De una manera general, se denomina entrenamiento hebbiano a aquellas formas de entrenamiento que involucran una modificación en los pesos proporcional al producto de una entrada j por la salida i de la neurona.

$$\Delta w_{ij} = \varepsilon y_i x_i \tag{3.27}$$

Donde ε es el ritmo de aprendizaje, que suele ser una cantidad entre 0 y 1. En donde la representación matemática es la siguiente considerando una función de transferencia lineal.

$$\Delta w_{ij}{}^{\mu} = t_i^{\mu} x_j^{\mu} \tag{3.28}$$

$$w_{ij}^{nuevo} = w_{ij}^{viejo} + \Delta w_{ij}^{\mu} \tag{3.29}$$

Si los pesos de partida son nulos, el valor final de W para las p asociaciones será:

$$W\mathrm{x}^{\mu} = \left(t^1 x^{1T} + \ldots + t^p x^{pT}\right) x^{\mu} \tag{3.30}$$

Eliminando la condición de ortogonalidad

$$W\mathrm{x}^{\mu} = t^{\mu} + \sum_{\nu \neq \mu} t^{\nu} \left(x^{\nu T} x^{\mu}\right) = t^{\mu} + \delta \tag{3.31}$$

La expresión anterior es denominada expansión señal ruido, pues proporciona la salida deseada más un término adicional, que interpretamos como el ruido superpuesto a la señal.

El entrenamiento de un precéptron (modelo de red neuronal retropropagación), es un algoritmo de entrenamiento de los denominados por corrección de errores. Los algoritmos de este tipo ajustan los pesos en proporción a la diferencia existente entre la salida actual de la red y la salida deseada con el objetivo de minimizar el error actual de la red.

Sea un conjunto de p patrones $\mathbf{x}^{\mu}$, μ=1,…,p, con sus salidas deseadas $\mathbf{t}^{\mu}$. Tanto las entradas como las salidas solamente pueden tomar los valores de -1 o 1. Si se tiene una arquitectura de preceptron simple, con pesos iniciales aleatorios, y se requiere que se clasifiquen correctamente todos los patrones del conjunto de entrenamientos (lo cual es solamente posible si son separables linealmente). Se actúa de un modo ante la presencia del patrón μ-ésimo, si la respuesta que proporciona el preceptron es correcta, no actualiza los pesos; si es incorrecta se modifican según la regla de Hebb.

$$\Delta w_{ij}^{\mu}(t) = \begin{cases} 2\varepsilon.t_i^{\mu}x_j^{\mu}, & si \quad y_i^{\mu} \neq t_i^{\mu} \\ 0 \qquad\quad , & si \quad y_i^{\mu} = t_i^{\mu} \end{cases} \tag{3.32}$$

$$\Delta w_{ij}^{\mu}(t) = \varepsilon.\left(t_i^{\mu} - y_i^{\mu}\right)x_j^{\mu} \tag{3.33}$$

Habrá que decir que la forma anterior es para el calculo de los gradientes de los pesos en punto actual de la salida o entrada de la red neuronal y como ya se había mencionado con anterioridad este entrenamiento se basa en la estimación de las salida por lo tanto es necesario el conocimiento de la salida, esto se logra introduciendo la siguiente ecuación para el entrenamiento para un conjunto de iteraciones t.

$$w_{ij}(t+1) = w_{ij}(t) + \sum_{\mu=1}^{p} \Delta w_{ij}^{\mu}(t) \tag{3.34}$$

Rosemblatt demostró que si la función a representar es linealmente separable, este algoritmo siempre converge en un tiempo finito y con independencia de los pesos de partida. Por otra parte si la función es no linealmente separable, el proceso de entrenamiento oscilara.

Si se añaden capas intermedias (ocultas) a un perceptron multicapa o MLP. Esta arquitectura suele entrenarse mediante el algoritmo denominado retropropagación de errores o bien haciendo uso de alguna de sus variantes o derivados, motivo por lo que en muchas ocasiones el conjunto arquitectura MLP + aprendizaje Backprpagation suele denominarse red de retropropagación.

Una solución al problema de entrenar los nodos de las capas ocultas pertenecientes a arquitecturas multicapa la proporciona el algoritmo de retropropagación de errores (Rumelhart, 1986a, 1986b, Hecht-Nielsen, 1990).

Dado un patrón de entradas $\mathbf{x}^{\mu}$, (μ=1,…, p), en donde la operación global de esta arquitectura se expresa del siguiente modo.

$$z_k^{\mu} = \sum w_{kj}^{'} y_j^{\mu} - \theta_k^{'} = \sum_j w_{kj}^{'} f\left(\sum_i w_{ji} x_i^{\mu} - \theta_j \right) - \theta_k^{'} \tag{3.35}$$

Las funciones de transferencia de las neuronas son del tipo sigmoideo, con h el potencial postsináptico o local. Como en el caso de la Adalina, la función de costo de la que es parte el error cuadrático medio.

$$E\left(w_{ji}, \theta_j, w_{kj}^{'}, \theta_k^{'}\right) = \left(\frac{1}{2}\right) \sum_{\mu} \sum_{k} \left[t_k^{\mu} - f\left(\sum_j w_{kj}^{'} y_j^{\mu} - \theta_k^{'} \right) \right]^2 \tag{3.36}$$

La minimización se lleva acabo mediante descenso por el gradiente, pero en esta ocasión habrá un gradiente respecto de los pesos de la capa de salida y otro con respecto de los de la oculta.

$$\delta w_{ji} = -\varepsilon \frac{\partial E}{\partial w_{ji}} \tag{3.37}$$

$$\delta w_{kj}^{'} = -\varepsilon \frac{\partial E}{\partial w_{kj}^{'}} \tag{3.38}$$

Las expresiones de actualización de los pesos se obtienen sólo con derivar, teniendo en cuenta las dependencias funcionales y aplicando adecuadamente la regla de la cadena.

$$\delta w'_{kj} = \varepsilon \sum_{\mu} \Delta'^{\mu}_{k} y^{\mu}_{j} \tag{3.39}$$

$$\delta w_{ji} = \varepsilon \sum_{\mu} \Delta^{\mu}_{j} x^{\mu}_{i} \tag{3.40}$$

$$\Delta'^{\mu}_{k} = \left[t^{\mu}_{k} - f\left(\nu'^{\mu}_{k}\right) \right] \frac{\partial f\left(\nu'^{\mu}_{k}\right)}{\partial \nu'^{\mu}_{k}} \tag{3.41}$$

$$\Delta^{\mu}_{j} = \left(\sum_{k} \Delta'^{\mu}_{k} w'_{kj} \right) \frac{\partial f\left(\nu^{\mu}_{j}\right)}{\partial \nu^{\mu}_{j}} \tag{3.42}$$

La actualización de los umbrales se realiza haciendo uso las ecuaciones anteriores, considerando que el umbral es un caso particular de peso sináptico, cuya entrada es una constante igual a -1.

En las ecuaciones anteriores esta implícito el concepto de propagación hacia atrás de los errores que da nombre al algoritmo. En primer lugar se calcula la expresión Δ'^{μ}_{k} que se denomina señal de error, por ser proporcional al error de la salida actual de la red, con el calculamos la actualización $\delta w'_{kj}$ de los pesos de la capa de salida. A continuación se propaga hacia atrás los errores Δ'^{μ}_{k} a través de las sinapsis, proporcionando así las señales de error Δ^{μ}_{j}, correspondientes a las sinapsis de la capa oculta; con éstas se calcula la actualización δw_{kj} de las sinapsis ocultas. El algoritmo puede extenderse fácilmente a arquitecturas con más de una capa oculta siguiendo el mismo esquema.

En resumen el procedimiento para entrenar mediante retropropagación una arquitectura multicapa (Figura 3.9):

1) Establecer aleatoriamente los pesos y umbrales iniciales (t:=0)
2) Para cada patrón μ del conjunto de aprendizaje.

   2.1) Llevar a cabo una fase de ejecución para obtener la respuesta de la red ante el patrón μ-ésimo (3.35).

   2.2) Calcular las señales de error asociadas Δ'^{μ}_{k} y Δ^{μ}_{j} según (3.39 -42)

   2.3) Calcular el incremento parcial de los pesos y umbrales debidos a cada patrón (μ).

3) Calcular el incremento total (para todos los patrones) actual de los pesos $\delta w'_{kj}$ y δw_{kj} según (3.39 - 42). Hacer lo mismo para los umbrales.
4) Actualizar pesos y umbrales.

5) Calcular el error actual (b) t:=t+1, y volver a 2) si todavía no es satisfactorio.

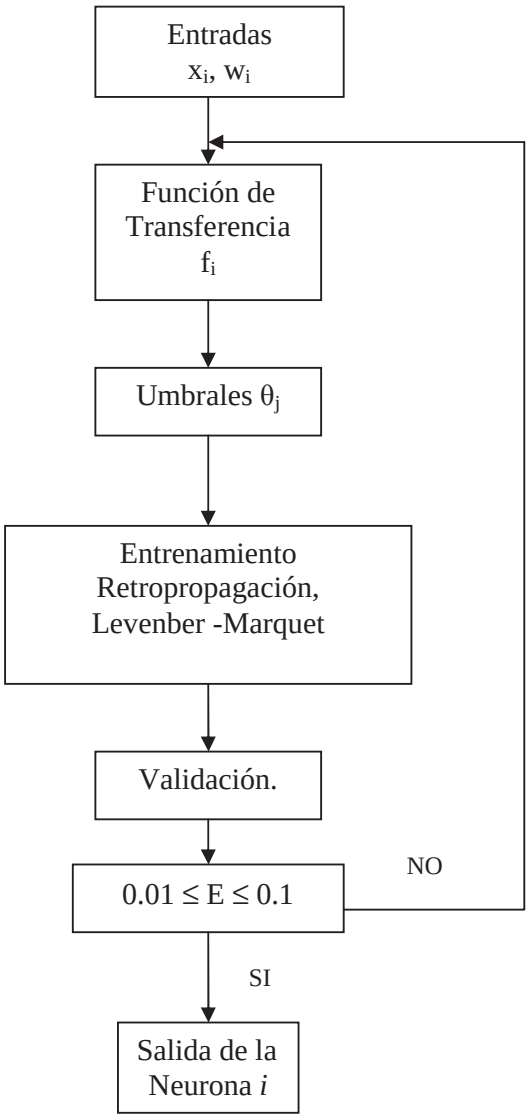


Figura 3.9 Diagrama de flujo de una neurona simple

Para resolver algunos de los inconvenientes de la retropropagación se plantea continuamente correcciones o variantes. Buena parte de estas modificaciones tratan de resolver el problema de su lenta convergencia, mientras que otras se centran en conseguir una mejor generalización. Una aceleración del entrenamiento lo da la siguiente expresión:

$$\delta w'_{kj}(t+1) = -\varepsilon \left. \frac{\partial E}{\partial w'_{kj}} \right|_t + \alpha.\delta w'_{kj}(t-1) \qquad (3.43)$$

$$\delta w_{ji}(t+1) = -\varepsilon \left.\frac{\partial E}{\partial w_{ji}}\right|_t + \alpha.\delta w_{ji}(t-1) \tag{3.44}$$

Existen otras técnicas que permiten acelerar el entrenamiento, consistentes en procesar las entradas, asignar un ritmo de aprendizaje diferente a cada peso, ritmos adoptivos, etc. (Sanz, 2005).

Como se ha visto, el entrenamiento de la retropropagación estándar esta basada en la técnica de descenso por el gradiente, es decir, calculando las derivadas de la superficie de error respecto de cada peso, se obtiene la dirección de máxima pendiente de la superficie la cual se sigue para actualizar los pesos; sin embargo, nadie garantiza que sea ese precisamente el camino mas rápido hacia el mínimo. Los denominados métodos de segundo orden se basan en realizar el descenso utilizando también la información por el ritmo de cambio de la pendiente.

$$H = \frac{\partial^2 E(W)}{\partial w_{ij} \partial_{kj}} \tag{3.45}$$

Los algoritmos de gradientes conjugados, gradientes conjugados escalados y Levenberg-Marquardt son ejemplos de ellos.

Este último está en función de la matriz Hessiana, el algoritmo de este tipo de entrenamiento es el siguiente:

$$x_{k+1} = x_k - \left[J^T J + \mu I \right]^{-1} J^T e \tag{3.46}$$

Son técnicas mas robustas que la retropropagación, que pueden acelerar en uno o dos ordenes de magnitud la convergencia, aunque como contrapartida son mucho más complejas de implementar y precisan más recursos de cálculo.

Se debe dejar en claro que desgraciadamente ninguno de los algoritmos o recetas descritos puede considerarse superior en general: un buen rendimiento en una puede proporcionar un rendimiento pobre para otra.

No obstante Matlab proporciona una gama importante de la implementación de dichos entrenamientos en donde el fundamento básico es el entrenamiento de Hebb.

Implementación de la caja de herramientas de MATLAB para el control.

Basados en el modelo matemático ya planteado se propone el siguiente diagrama para el control de la temperatura en un reactor por lotes como se muestra en la Figura 3.10.

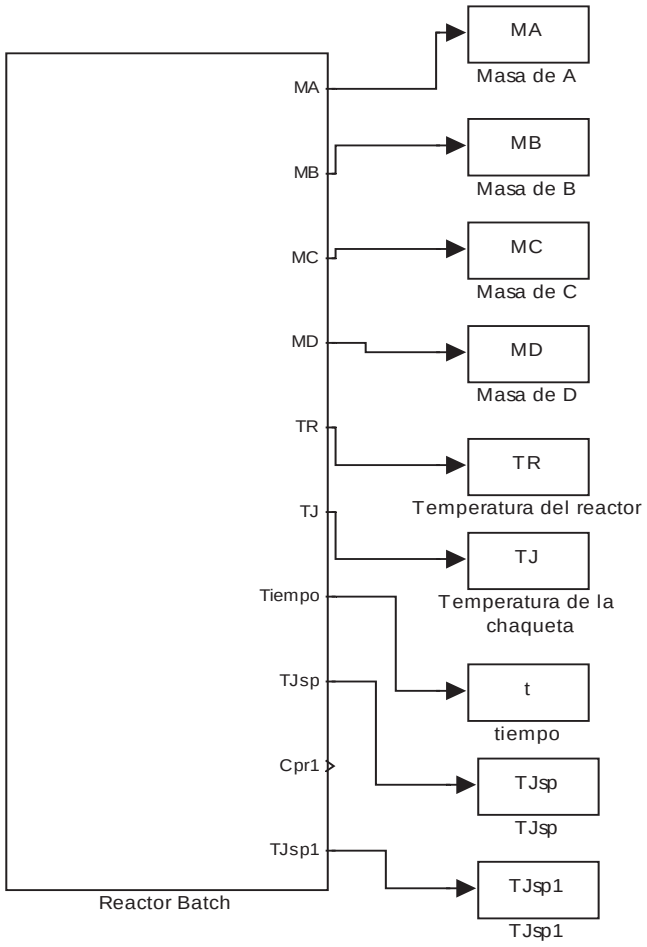


Figura 3.10 Modelo enmascado control con redes neuronales

El modelo de forma detallada se muestra en la Figura 3.11.

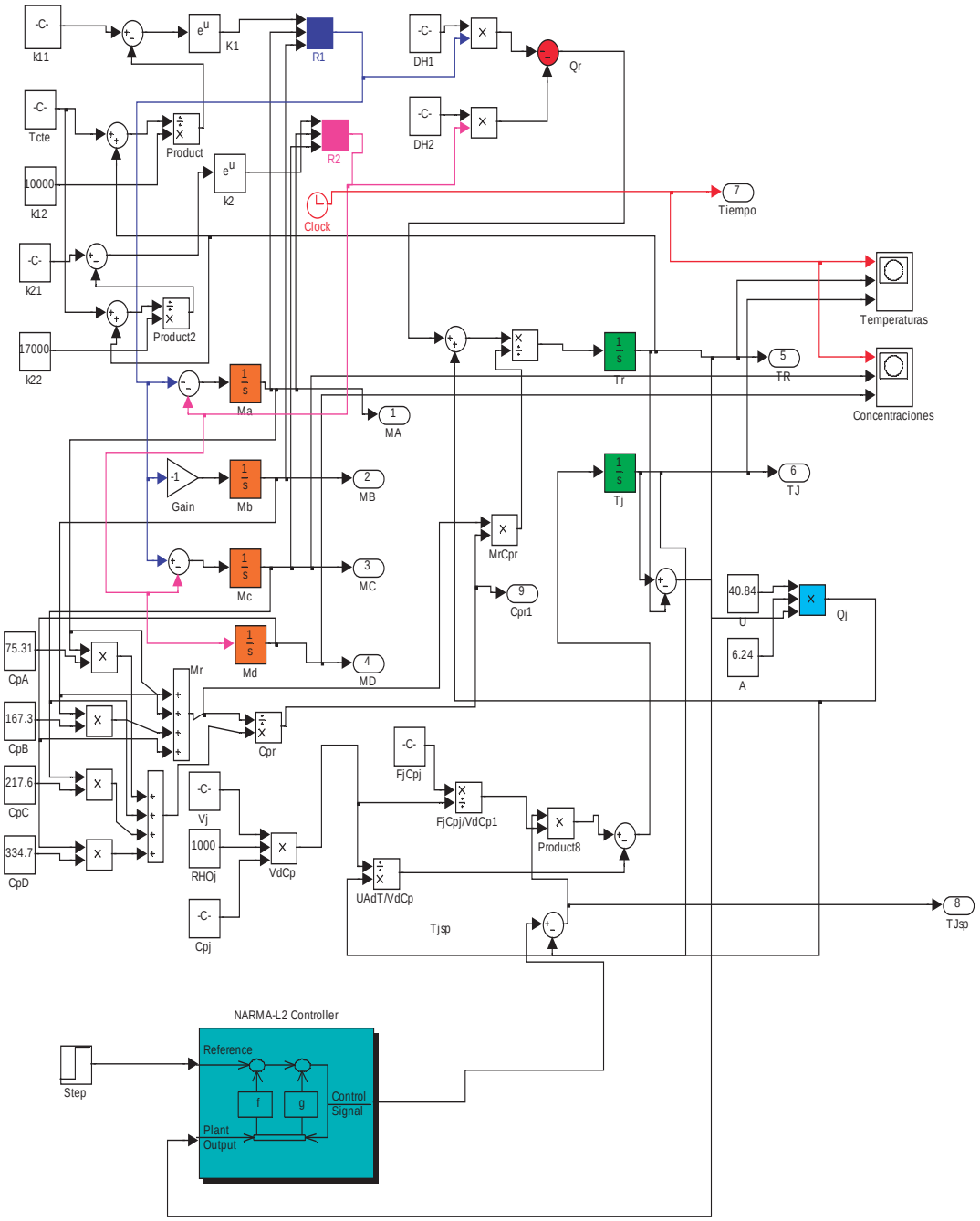


Figura 3.11 Control con redes neuronales

La estructura interna del controlador NARMA L-2 que en este trabajo se utilizó es la siguiente:

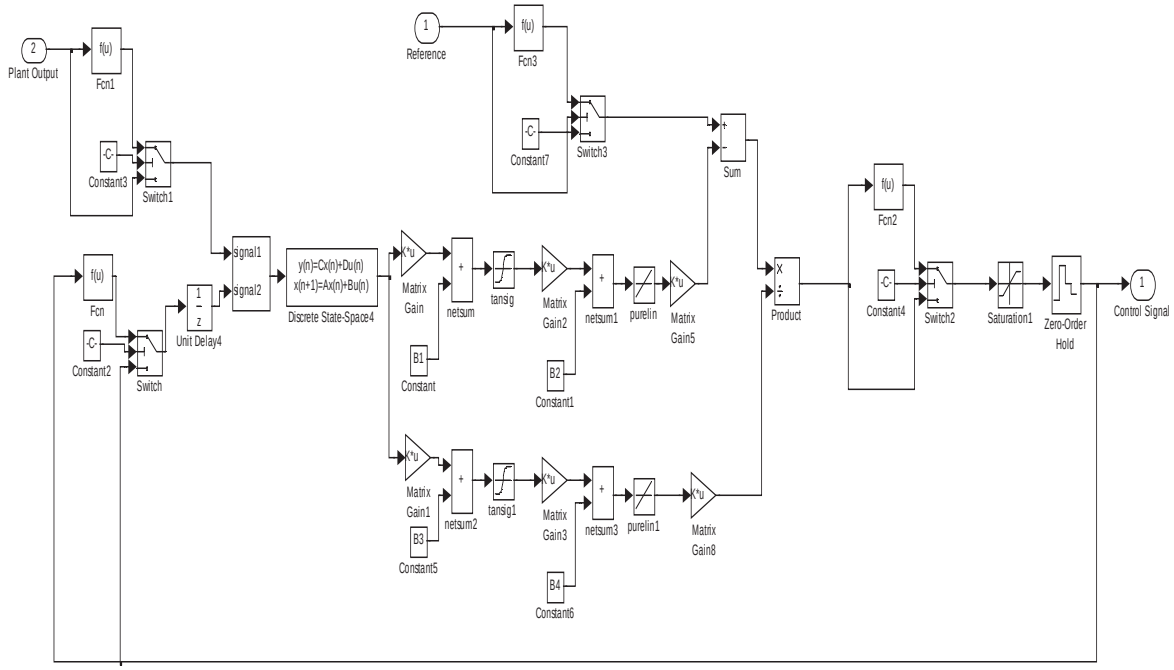


Figura 3.12 Estructura interna del controlador NARMA L-2.

En la Figura 3.12 se muestra la estructura interna del controlador propuesto en este trabajo, en donde, se tiene dos entradas y una sola salida, el controlador esta estructurado de la siguiente manera:

1. Un receptor de señal con 2 entradas y una salida; estructura realizada por Simulink (Figura 3.13).

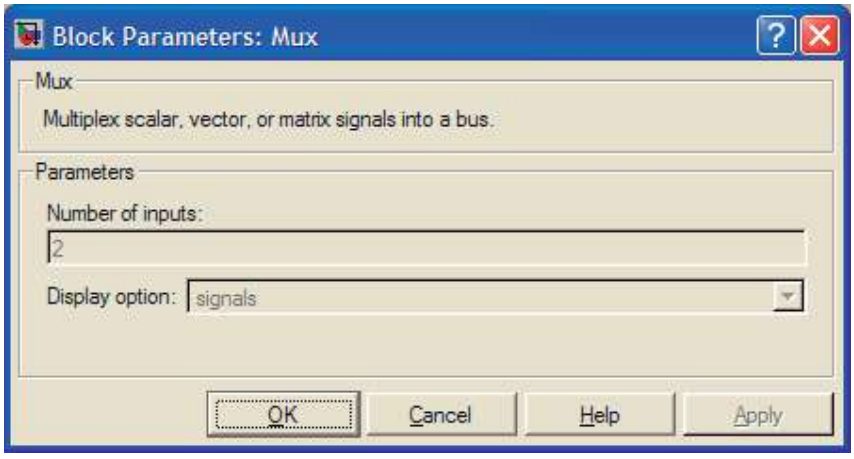


Figura 3.13 Receptor de dos señales

2. Una función de discretización que tiene la principal función de generar los pesos sinápticos y los vectores de entrada a la red neuronal (Figura 3.14).

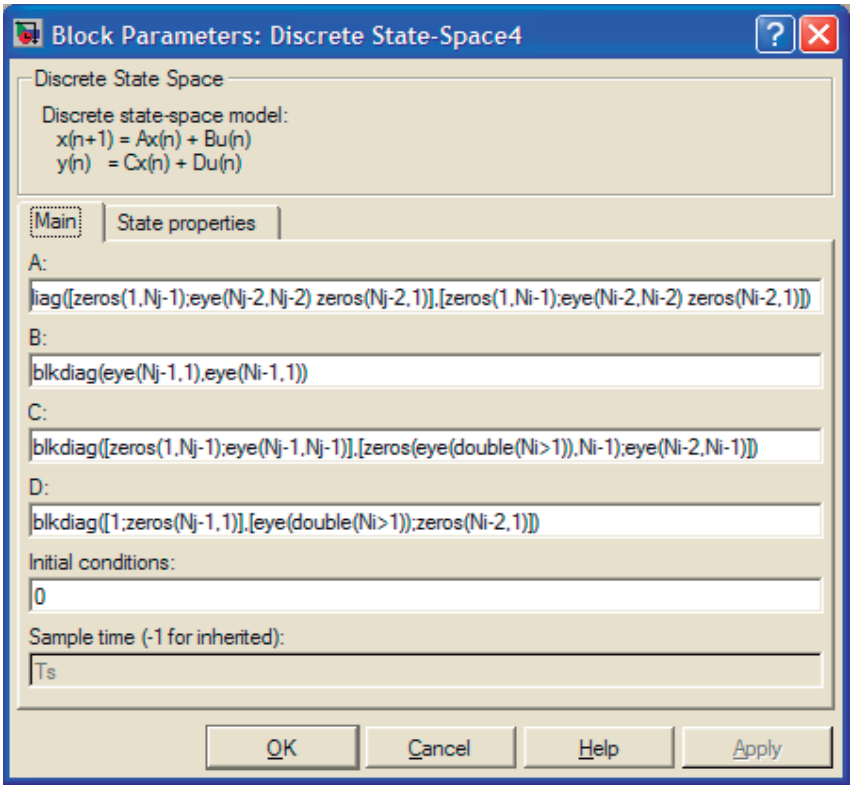


Figura 3.14 Función de discretización

3. Cuatro sumadores de neuronas (Figura 3.15).

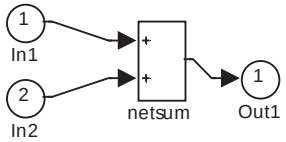


Figura 3.15 Sumador de neuronas

4. Dos funciones de transferencia (excitación) de tipo sigmoidea (Figura 3.16).

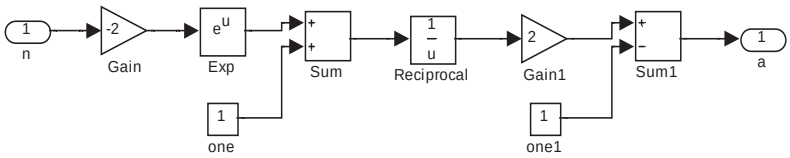


Figura 3.16 Función de excitación

5. Dos funciones de transferencia del tipo lineal (Figura 3.17).

3.17 Función lineal

6. Una función de saturación que tiene el destino de imponer limites para el entrenamiento de la red neuronal, en las salidas como en las entradas (Figura 3.18).

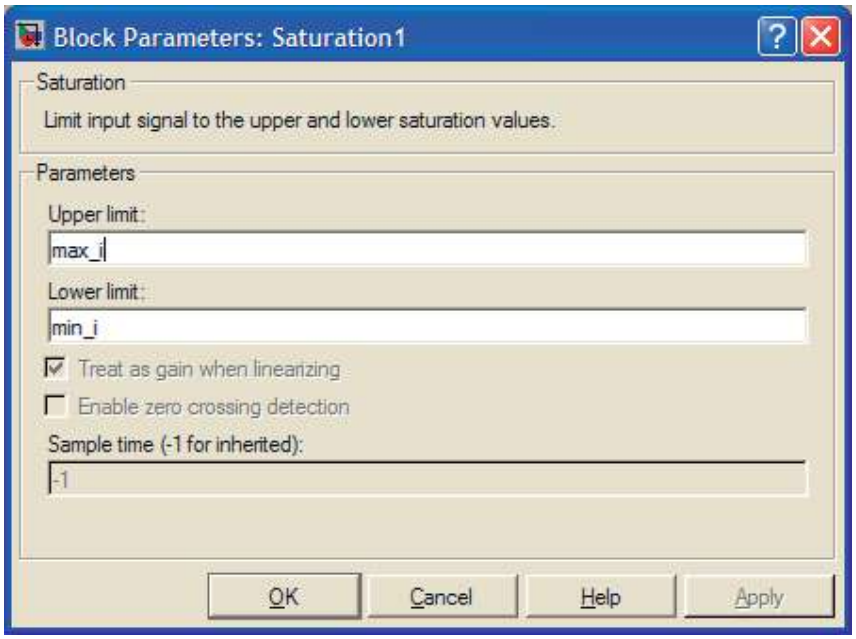


Figura 3.18 Función de saturación

Para la simulación es necesario introducir datos en la red neuronal elegida para el control, los parámetros necesarios para la simulación son los siguientes (Figura 3.19):

- Una red neuronal de 7-22 neuronas internas.
- Dos capas ocultas de las cuales cada una contiene 10 neuronas.
- Un entrenamiento en línea de 4000 muestras.
- Un máximo de entrada, un mínimo de entrada de 93 y 60 ºC respectivamente.
- Un máximo y mínimo de salida de 93 y 20 ºC respectivamente.
- Un modelo de referencia.

- Un total de 200 muestras de mejoramiento en el entrenamiento esto para la disminución del error.
- Un entrenamiento de propagación hacia atrás.

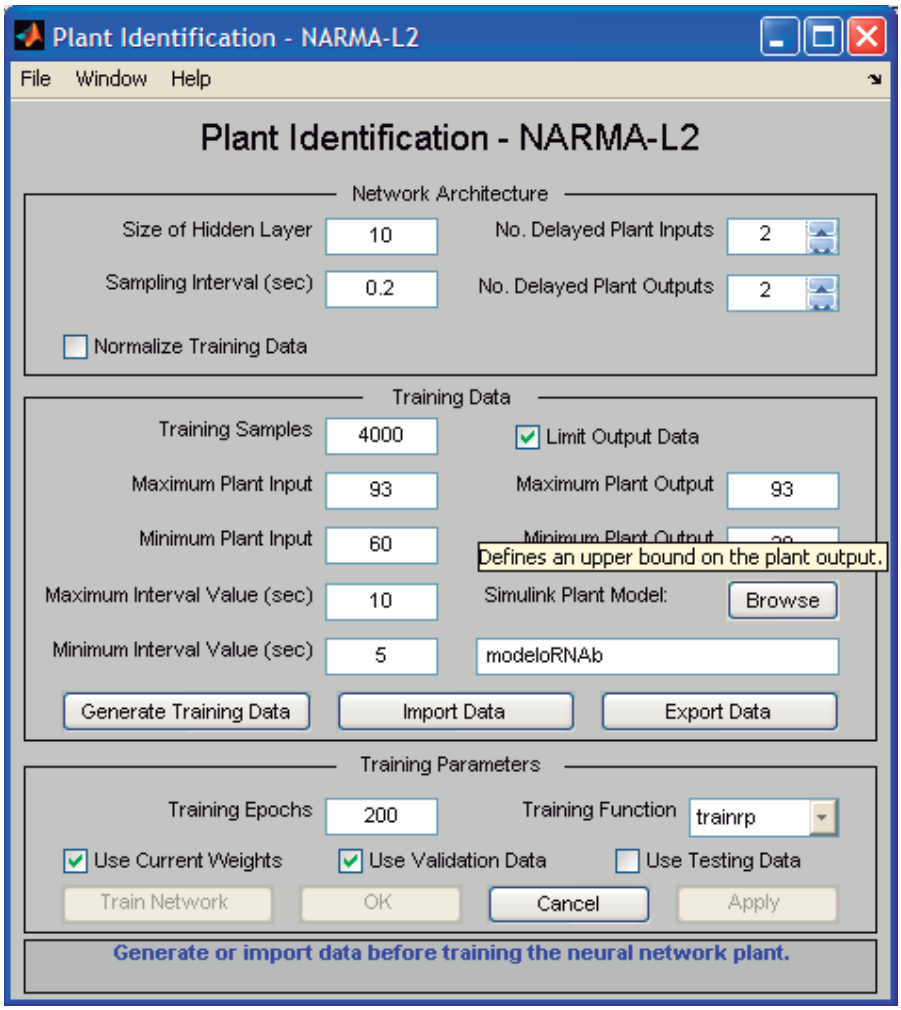


Figura 3.19 Parametrización de la red neuronal NARMA-L2.

Para fines de comparación se plantea el control GMC por sus siglas en ingles (Generic Model Control) y dándole solución en Simulink.

Controlador GMC ley de control (Lee et. al., 1988).

$$u = K_1\left(x^* - x\right) + K_2\int\left(x^* - x\right)dt \qquad (3.47)$$

Para este caso de estudio se tiene:

$$T_{jsp} = T_r + \frac{WCp_j}{UA}\left(K_1\left(T_{rsp} - T_r\right) + K_2 \int_0^t \left(T_{rsp} - T_r\right) dt \right) - \frac{Q_r}{UA} \qquad (3.48)$$

En este tipo de control como en que se plantea con redes neuronales la variable a controlar es la temperatura de la chaqueta que propiamente es la temperatura de la chaqueta en el set-point.

Las constantes K_1 y K_2 se obtienen usando las siguientes ecuaciones.

$$K_1 = \frac{2\xi}{\tau} \qquad (3.49)$$

$$K_2 = \frac{1}{\tau^2} \qquad (3.50)$$

Donde ξ y τ son constantes determinadas mediante optimización del error en donde los valores correspondientes son 10.0 y 80 respectivamente (Macchietto, Cott, 1975), obteniéndose los valores de las constantes de K_1= 0.2, K_2= 1.00×10^{-4}, introduciendo las constantes en la ley de control, resolviendo con Simulink se obtiene el siguiente diagrama (Figura 3.20).

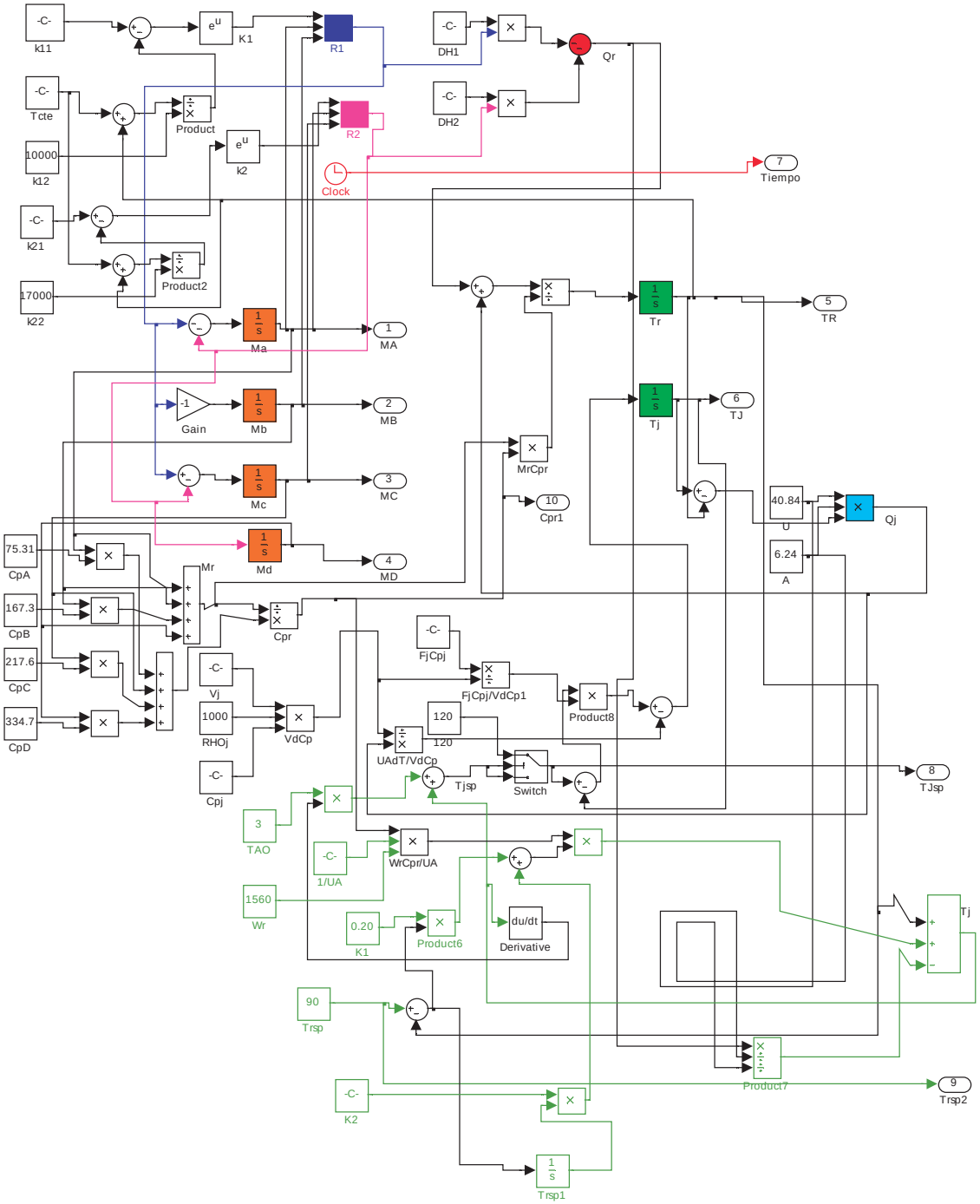


Figura 3.20 Diagrama de control GMC

CAPITULO 4

RESULTADOS

Se presentan los resultados obtenidos para un sistema de control que hace uso de las redes neuronales, para un sistema de dos reacciones paralelas, en donde se obtiene de manera no deseada un producto que depende directamente de la temperatura, siendo esta la variable que se debe controlar en el proceso.

4.1 Red Neuronal

Las simulaciones realizadas se obtuvieron introduciendo en el diagrama de Simulink los parámetros necesarios para el entrenamiento de la red neuronal como una cantidad de siete neuronas con sólo dos capas ocultas, una capa de entrada y una capa de salida. Una de las variables en la red neuronal que se modifica es la cantidad de neuronas en un intervalo de siete hasta veintidós neuronas para determinar la mejor estructura de la red neuronal (Figura 4.1).

Proponiendo un cambio escalón de 20 a 60 ºC con lo parámetros mencionados anteriormente. El algoritmo de entrenamiento fue el de propagación hacia atrás, el entrenamiento se hace en incrementos, esto para que la red neuronal logre comprender la dinámica del proceso, obteniéndose los resultados del entrenamiento (Figura 4.2) así como la validación del entrenamiento (Figura 4.3) y la obtención de los perfiles de temperatura ya implementando la red neuronal como control.

La Figura 4.2 muestra cuatro gráficos, dos de los cuales son tanto la entrada y la salida del proceso en general. El error también es mostrado, como se observa es mínimo, aunque debido a este tipo de entrenamiento, el error no alcanza a minimizarse en su totalidad. La salida de la red neuronal es muy semejante a la salida del proceso, esto significa que la red neuronal comprendió el proceso y logra controlarlo.

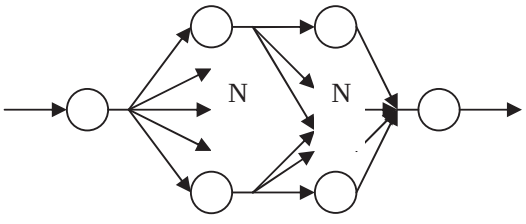


Figura 4.1 Estructura de la red neuronal

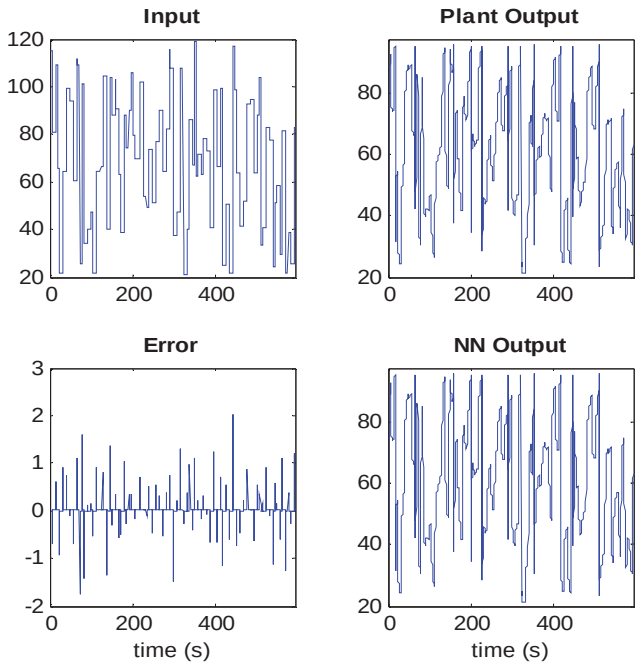


Figura 4.2 Entrenamiento de la red neuronal (incremento de 20 ºC – 60 ºC)

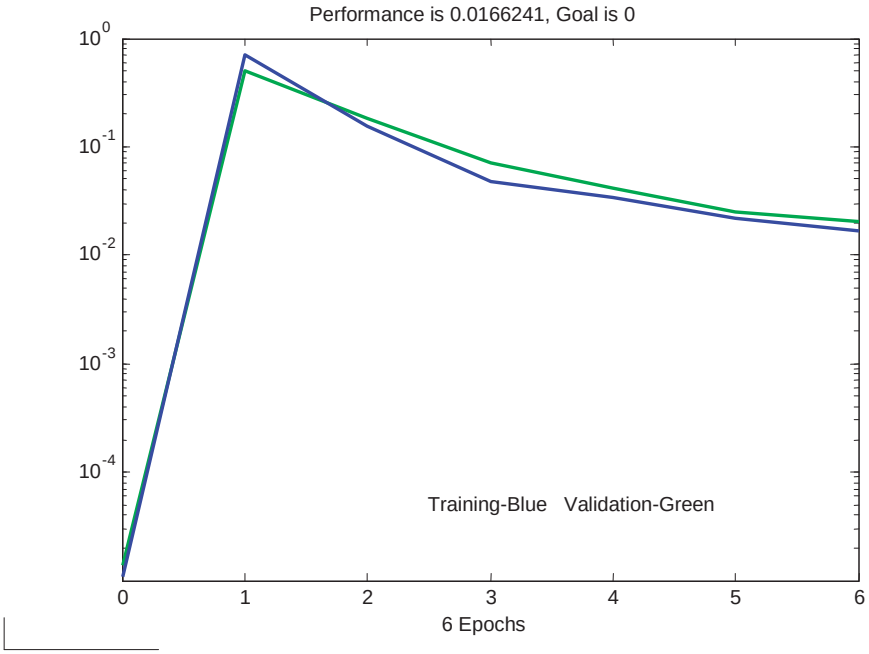


Figura 4.3 Validación del entrenamiento

Con el entrenamiento de propagación hacia atrás se obtiene el perfil de temperatura (Figura 4.4), este presenta oscilación siendo semejante al control on-off.

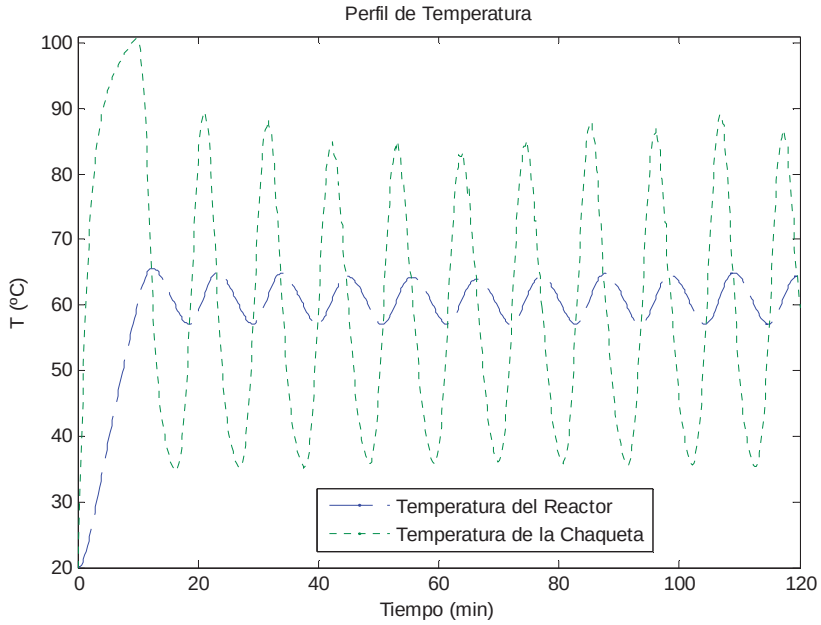


Figura 4.4 Perfil de Temperatura (incremento 20 ºC – 60 ºC)

Proponiendo un cambio escalón de 20 ºC a 80 ºC. Se observa que el error se mantiene con una tendencia constante con respecto al entrenamiento anterior entre el intervalo de 2 y -2 siendo esto los umbrales de la red neuronal (Figura 4.5). La validación del entrenamiento con respecto a los pesos de entrada y de salida se muestra en la Figura 4.6.

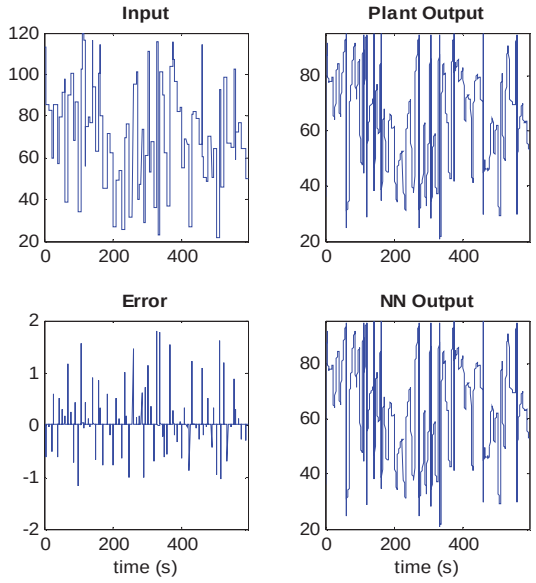


Figura 4.5 Entrenamiento de la red neuronal (incremento 20 ºC – 80 ºC)

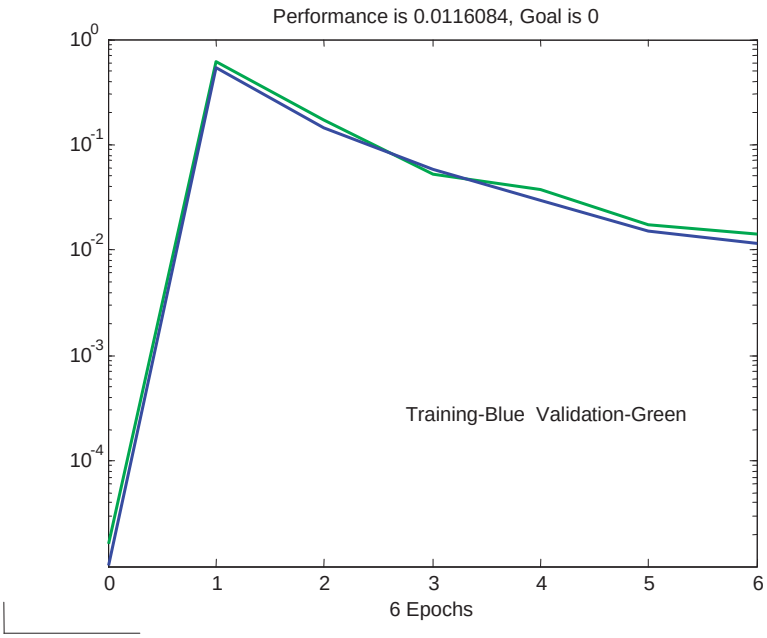


Figura 4.6 Validación del entrenamiento

El perfil de temperatura se muestra en la Figura 4.7, y se puede observar que también presenta oscilaciones.

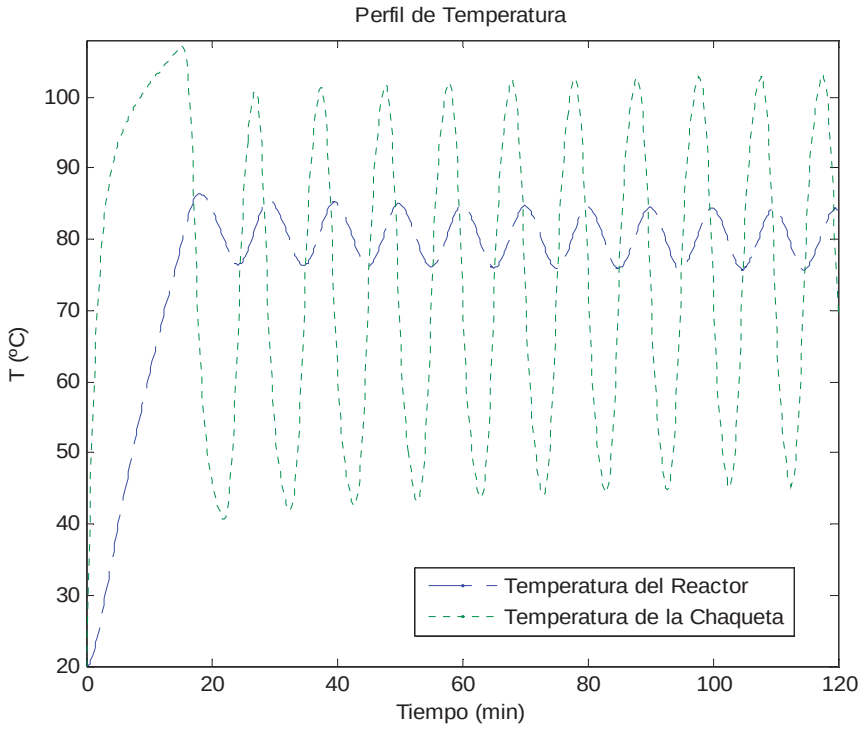


Figura 4.7 Perfil de Temperatura (Incremento 20 ºC – 80 ºC)

El entrenamiento debe de hacerse de una manera de incrementos pequeños, esto debido que en un incremento abrupto la red neuronal tiende a confundirse (hay que decir que se esta tratando con inteligencia artificial), pues el paso anterior queda asociada en la memoria de la red neuronal, debido a que toda red neuronal tiene como principio de entrenamiento la regla de entrenamiento de Hebb que es un aprendizaje semejante al del ser humano.

Proponiendo un cambio escalón de 20 ºC a 90 ºC, el error se minimiza estando entre un intervalo de 0.02, -0.01 siendo este considerado como aceptable (Figura 4.8), en la validación del modelo no se observa discrepancia entre ambas líneas (Figura 4.9).

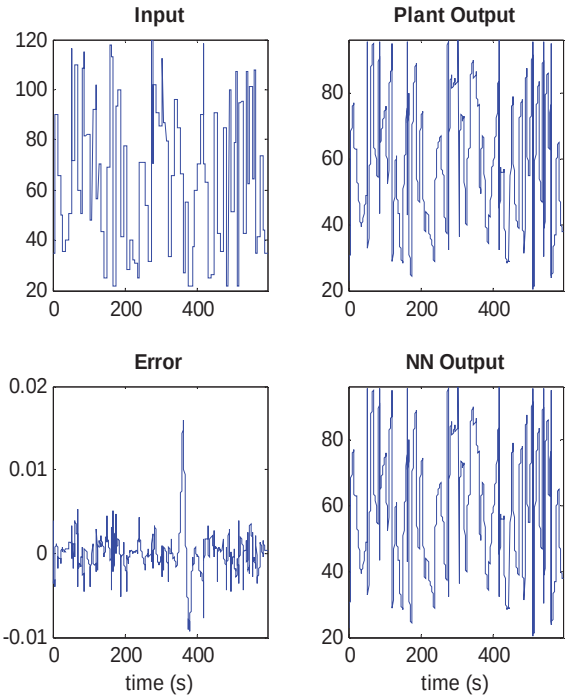


Figura 4.8 Entrenamiento de la red neuronal (incremento 20 ºC – 90 ºC)

El perfil de temperatura para este incremento es mostrado en la Figura 4.10. Como se puede observar el perfil de temperatura tiene la misma tendencia que presenta los perfiles anteriores, de los incrementos de temperatura siendo una tendencia que la red neuronal a comprendido la dinámica del proceso implementando un control del tipo on-off.

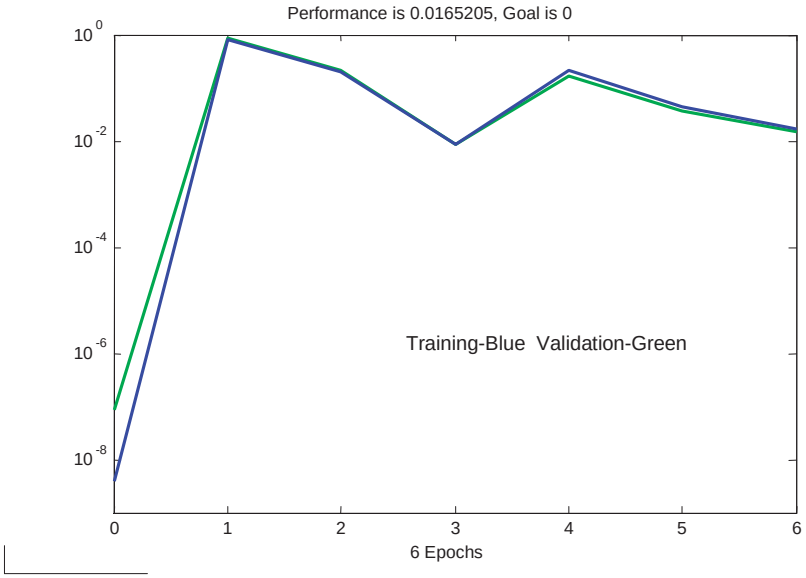


Figura 4.9 Validación del entrenamiento

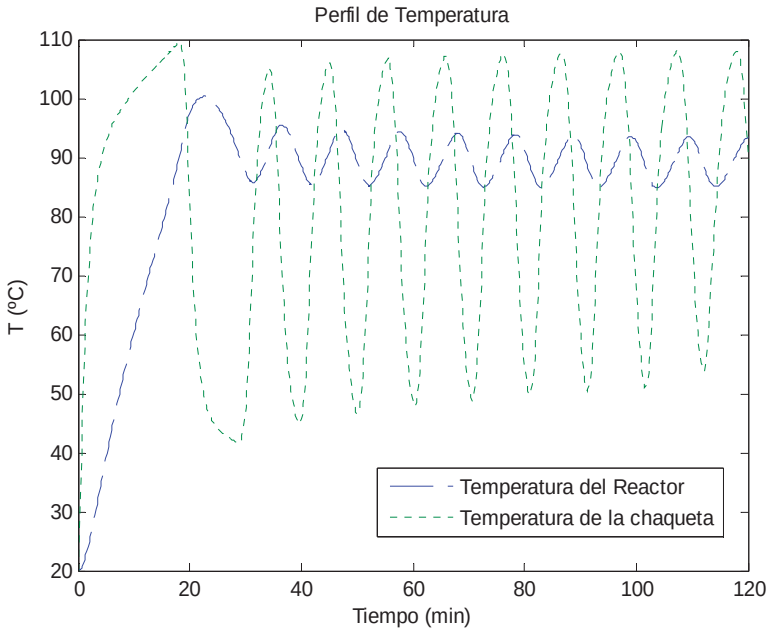


Figura 4.10 Perfil de Temperatura (incremento 20 ºC – 90 ºC)

De los cambios anteriores se aprecia que en el entrenamiento de la red neuronal la disminución del error no se logra, provocando la oscilación del control de la temperatura, los incrementos para el entrenamiento hacen evidente la poca comprensión de la dinámica del proceso.

Ahora de manera similar se hace un nuevo incremento de 90 ºC a 92.46 ºC esto debido a que en la literatura encontrada se observa que el punto de estabilidad del reactor es a una temperatura de 92.46 ºC. En este incremento no es necesario el entrenamiento de toda la red pues sólo son 2.46 ºC, la red a comprendido la dinámica del proceso obteniendo el perfil de temperatura (Figura 4.11).

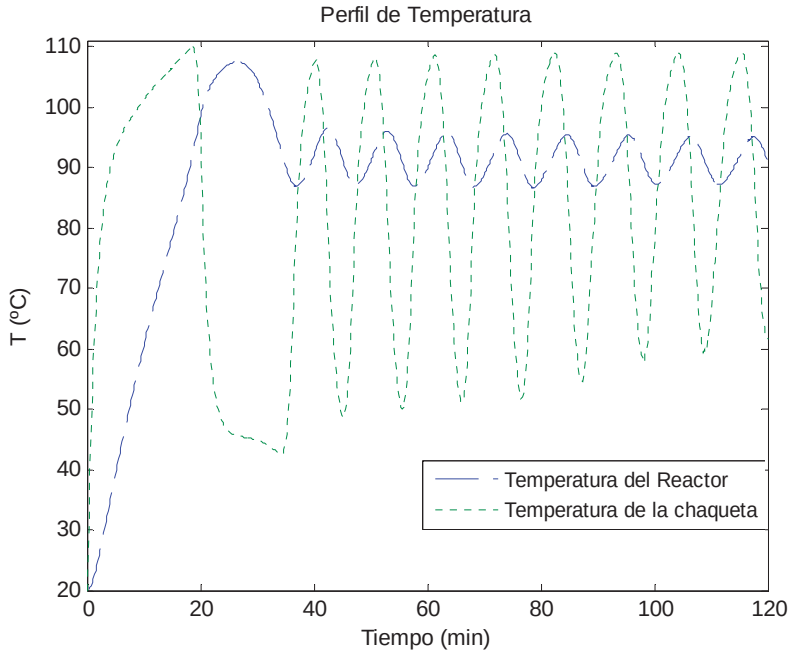


Figura 4.11 Perfil de Temperatura set point

Al hacer el siguiente incremento de 92 ºC a 93 ºC la temperatura se hace incontrolable esto debido a que el límite de salida de planta se fijo en una temperatura de 95 ºC que seria el máximo.

Por lo anterior al encontrar los puntos tanto del máximo como mínimo de la red neuronal, se incrementa el número de neuronas incrementando de siete a diez neuronas, con dos capas ocultas y el mismo tipo de entrenamiento de propagación hacia atrás.

La Figura 4.12 muestra el perfil de temperatura con un número de neuronas incrementado, el entrenamiento se hace con el mismo algoritmo de entrenamiento que para un número de neuronas menor, para un cambio escalón de 20 ºC a 60 ºC.

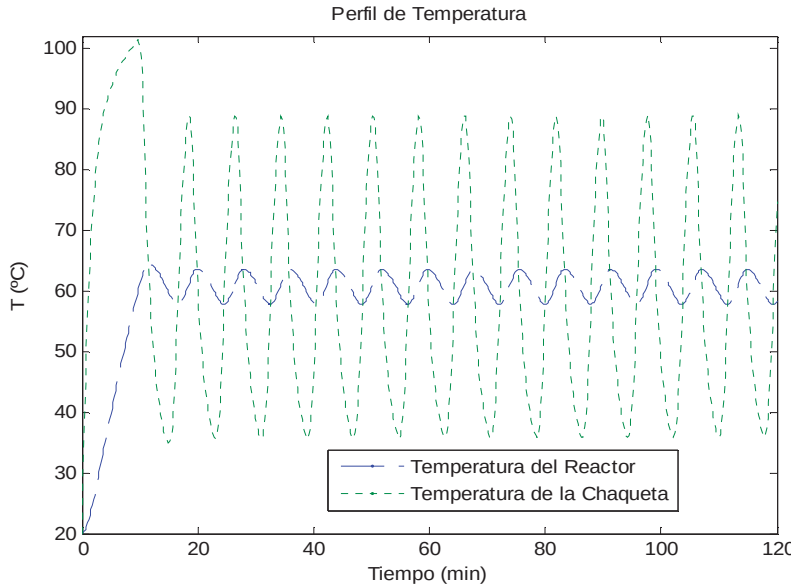
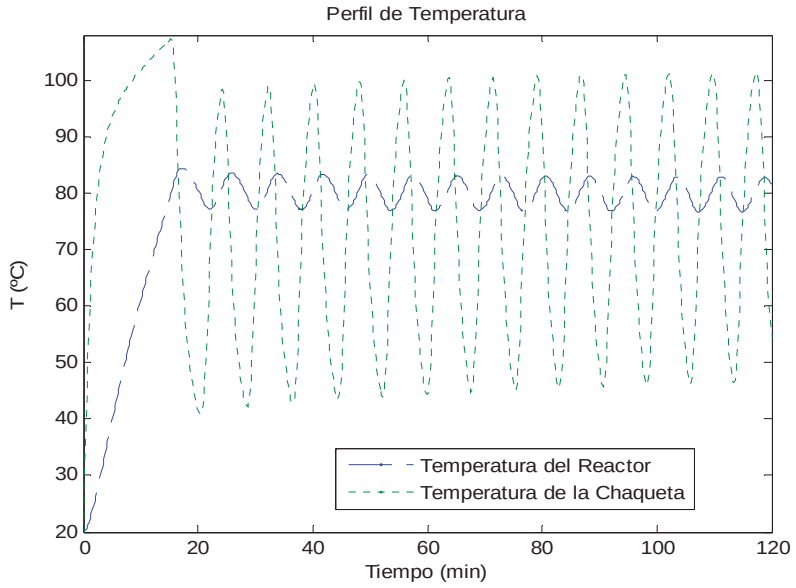


Figura 4.12 Perfil de Temperatura

Después se hace el siguiente incremento en un cambio escalón de 20 a 80 ºC obteniendo el perfil de temperatura de la Figura 4.13.

El control se mantiene por debajo del set point dado, llegándose a un control aceptable, presentando aun oscilaciones similares a un control tipo on-off.



.

Figura 4.13 Perfil de Temperatura

El siguiente incremento es de 20 a 90 ºC (Figura 4.14). Se continúa obteniendo el mismo perfil de los incrementos anteriores

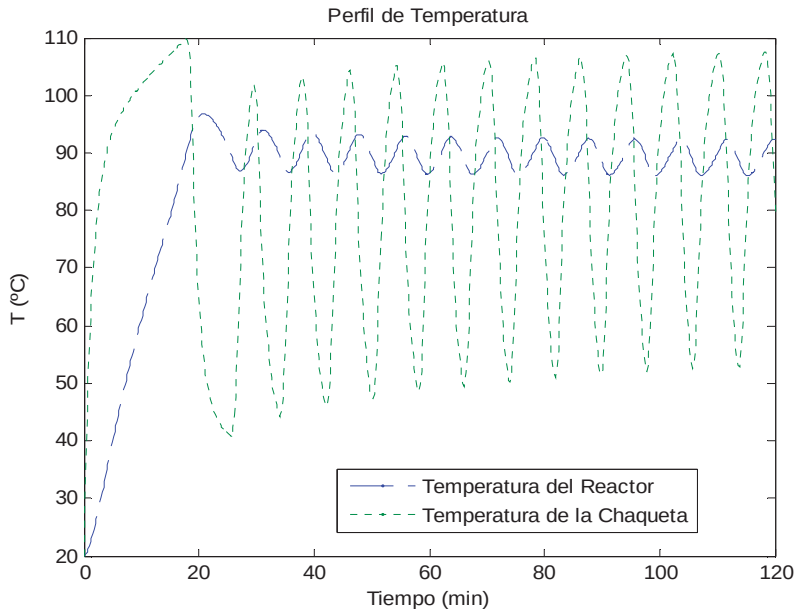


Figura 4.14 Perfil de Temperatura.

Al hacer incrementos mayores, la red al principio se adapta progresivamente al conjunto de aprendizaje acomodándose al problema y mejorando la generalización. Sin embargo, en un momento dado el sistema se ajusta demasiado a las particularidades de los patrones empleados en el entrenamiento, aprendiendo incluso del ruido en ellos presente, por lo que crece el error que comete ante patrones diferentes a los empleados en el entrenamiento (error de generalización). En ese momento la red no ajusta correctamente el mapping, si no que simplemente esta memorizando los patrones del conjunto de aprendizaje, lo que técnicamente se denomina sobre aprendizaje o sobre ajuste, pues la red esta aprendiendo demasiado (incluso el ruido presente en los patrones).

Con base a que la red propuesta anteriormente no se ajusta adecuadamente, ahora se propone, una red de 13 neuronas y con un entrenamiento de propagación hacia atrás, obteniendo el perfil de temperatura de la Figura 4.16, el cual no es muy bueno aun, como se puede observar en el error (Figura 4.15)

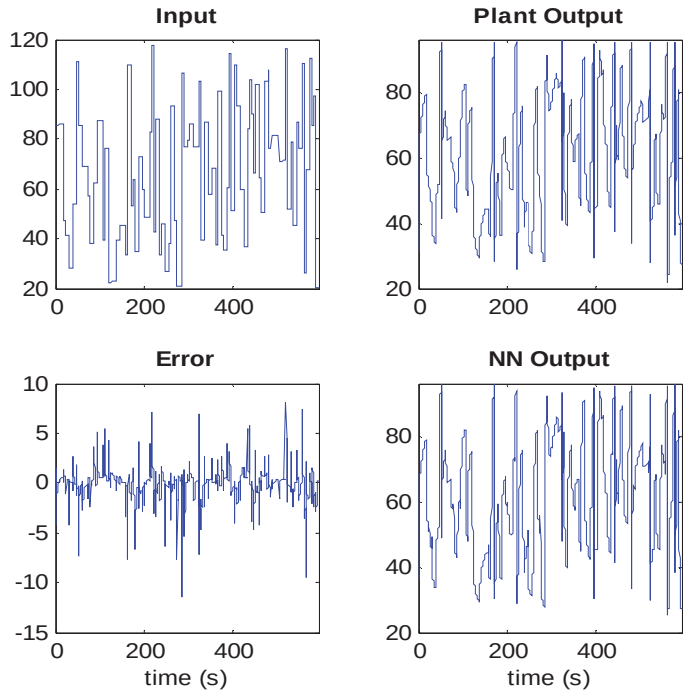


Figura 4.15 Entrenamiento de la red con 13 neuronas en cada capa

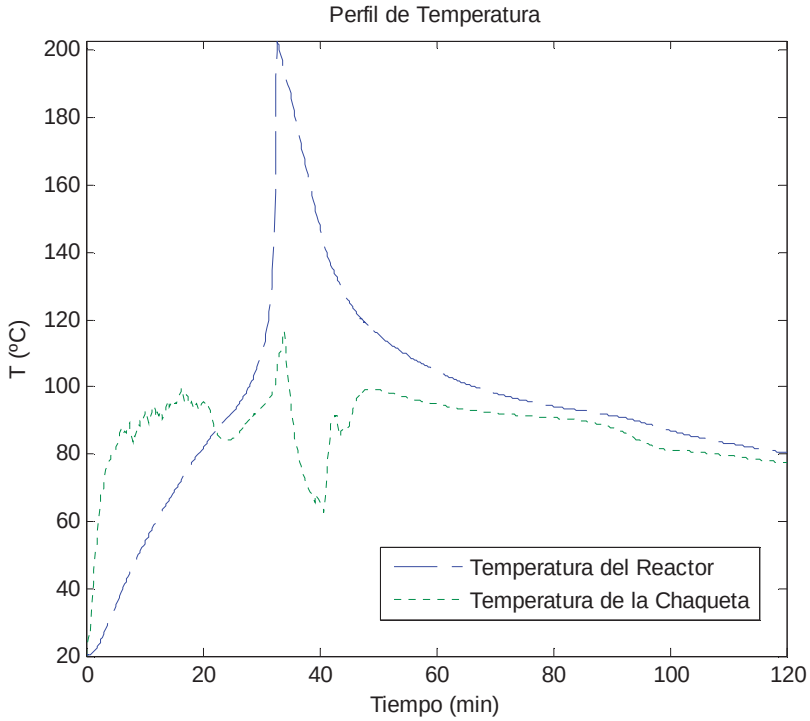


Figura 4.16 Perfil de Temperatura con una red de 13 neuronas en cada capa.

De lo anterior se ve la necesidad de incrementar la cantidad de neuronas nuevamente de 13 a 16 en cada capa interna en la red neuronal, obteniéndose resultados de las Figuras 4.17 y 4.18 para un cambio de 20 a 80 ºC, observándose que no se ajustan muy bien los resultados, pero existiendo poca oscilación.

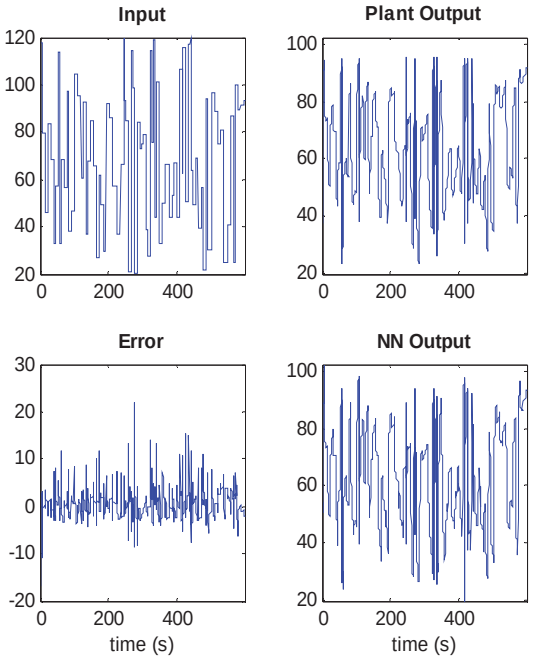


Figura 4.17 Entrenamiento de la red neuronal con 16 neuronas en cada capa

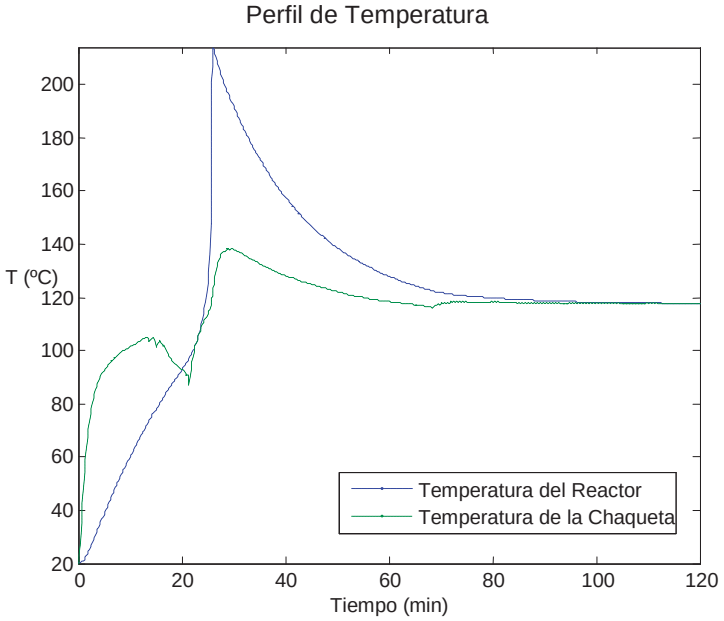


Figura 4.18 Perfil de Temperatura con una red de 16 neuronas en cada capa

Después se incrementó de 16 a 19 neuronas en cada capa de la red neuronal mostrando los perfiles del entrenamiento y el perfil de temperatura (Figura 4.19 y 4.20). También existe oscilación.

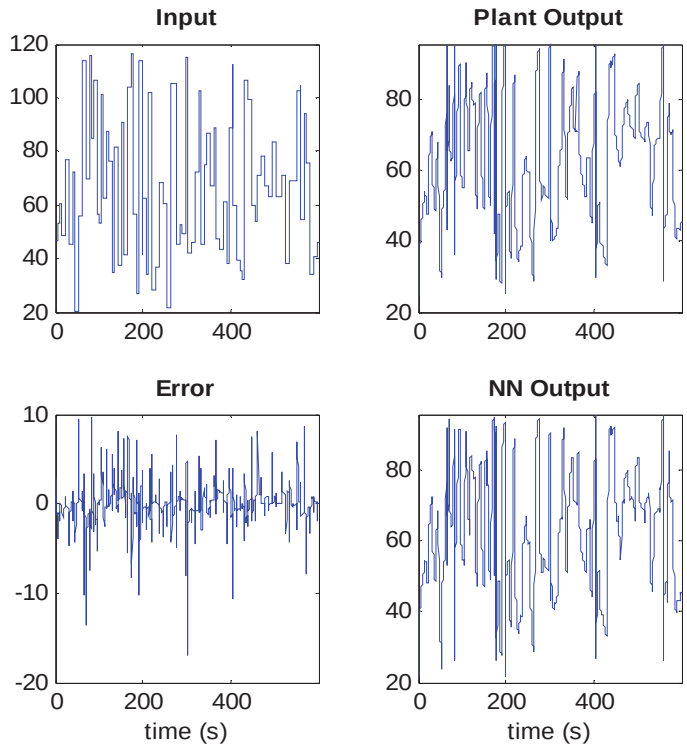


Figura 4.19 Entrenamiento de la red neuronal con 19 neuronas en cada capa

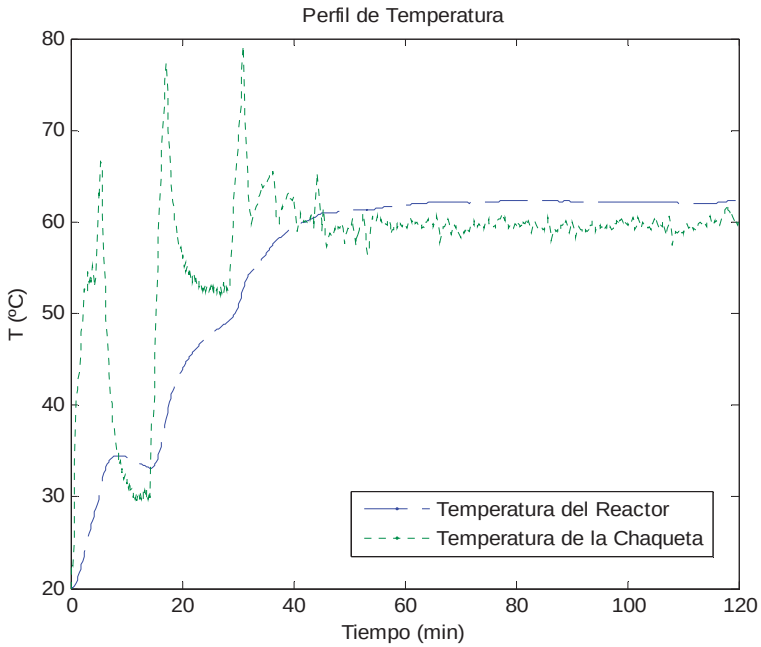


Figura 4.20 Perfil de Temperatura con una red de 19 neuronas en cada capa

Finalmente se incremento la cantidad de neuronas de 19 a 22 por cada capa oculta proporciona los resultados de la Figura 4.21 y 4.22. Se observa que las oscilaciones aumentan.

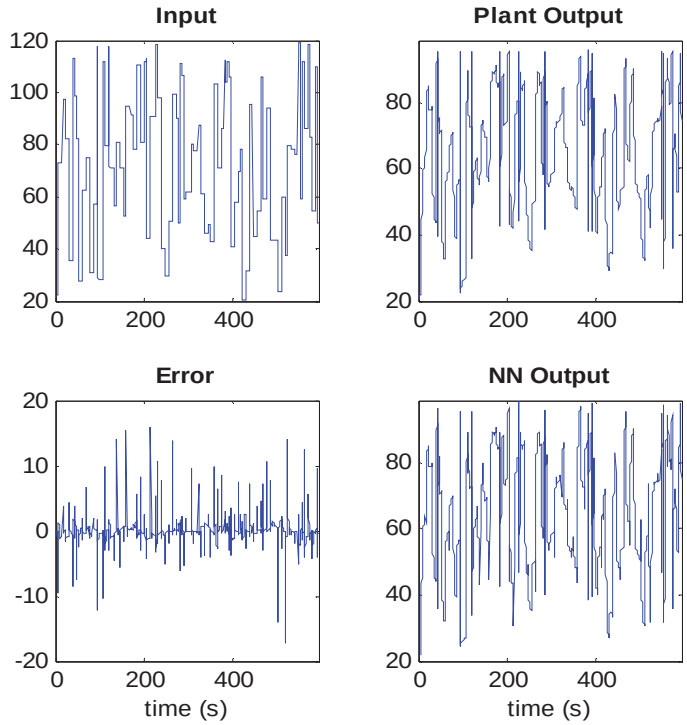


Figura 4.21 Entrenamiento de la red neuronal con 22 neuronas en cada capa

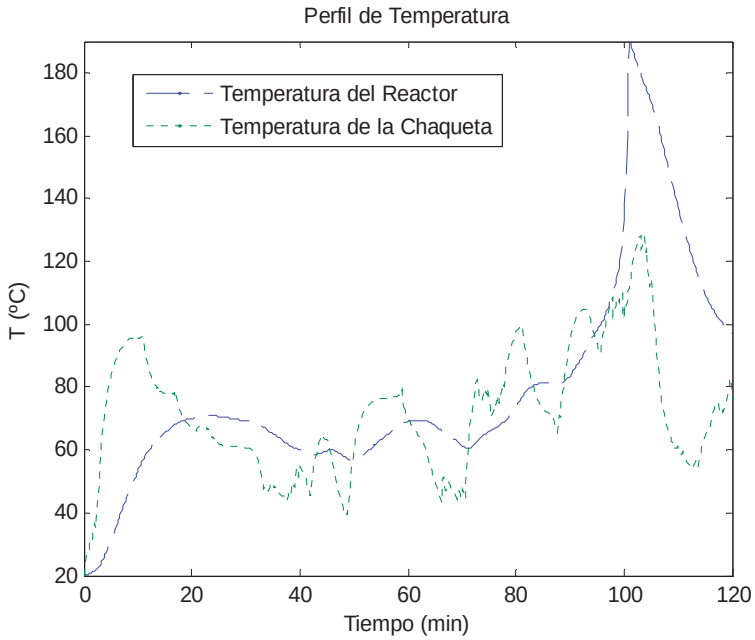


Figura 4.22 Perfil de Temperatura con una red de 22 neuronas en cada capa

Se observó que al aumentar la cantidad de número de neuronas en las capas ocultas se disminuye las oscilaciones, pero sin llegar a un buen control de la variable principal.

Debido a los resultados obtenidos con anterioridad, se plantea el cambio del tipo de entrenamiento del actual de propagación hacia atrás a un entrenamiento del tipo de Levenberg-Marquardt (L-M) que es un entrenamiento acelerado de convergencia y mejor, ya que dicho entrenamiento no acarrea los errores que con lleva el entrenamiento de propagación hacia atrás; en pocas de las veces este aprendizaje tiende a no presentar sobre aprendizaje característica que se presenta en muchas de las ocasiones en el aprendizaje de propagación hacia atrás.

La obtención de resultados se hace únicamente haciendo una modificación en la parametrización de la red neuronal ya que esta comprobado en la literatura que para el tema de investigación la reacción tiene un máximo en la temperatura de 92 ºC por lo tanto la modificación es significativa y presente en el método de entrenamiento Levenberg-Marquardt (L-M). Este cambio se realiza por la razón de que en los resultados obtenidos con anterioridad existe en la mayoría de los casos una oscilación demasiado marcada en la temperatura de la chaqueta comportándose el control como un control on-off.

Una red neuronal de 10 neuronas en las capas ocultas produce el perfil de temperatura mostrado en la Figura 4.23.

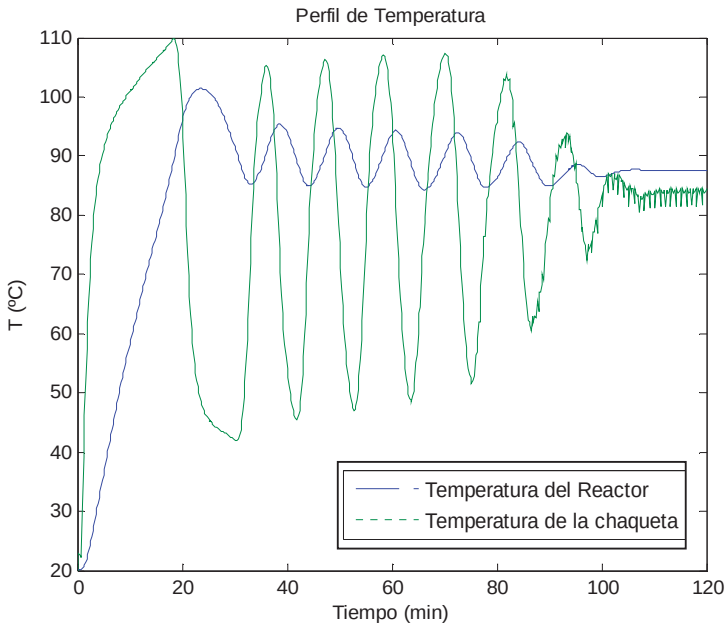


Figura 4.23 Perfil de Temperatura con un entrenamiento L-M

El comportamiento incrementando el número de neuronas internas de 10 a 13 neuronas en las capas ocultas se observa en la Figura 4.24.

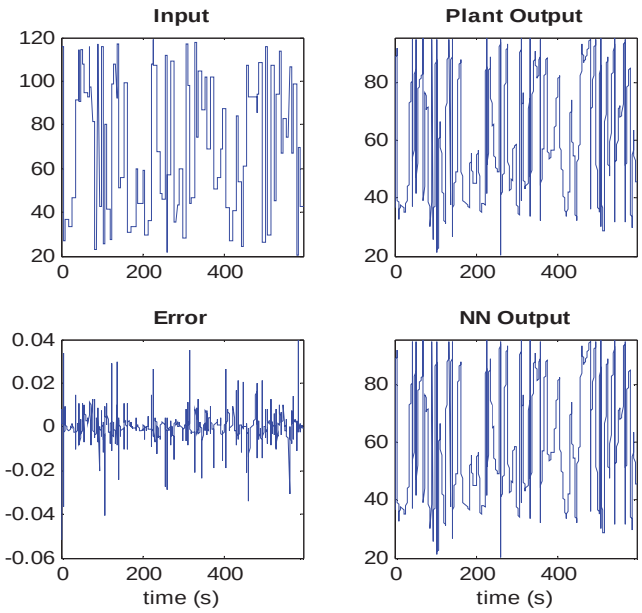


Figura 4.24 Entrenamiento de la red neuronal L-M

Se puede ver que el perfil de temperatura presenta mayor estabilidad y control de la dinámica del proceso en la Figura 4.25.

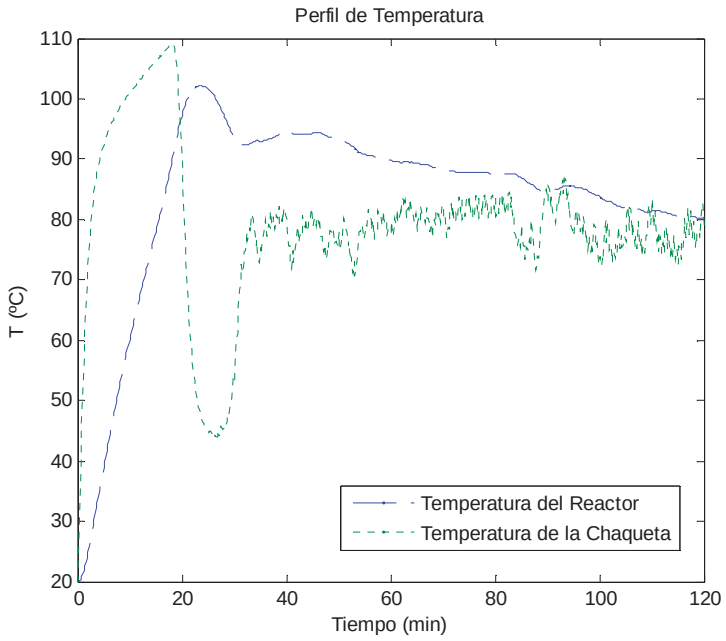


Figura 4.25 Perfil de Temperatura entrenamiento L-M

Si se incrementa de 13 a 16 neuronas en las capas ocultas internas de la red neuronal mostrándose los siguientes resultados de su entrenamiento y el perfil de temperatura se muestra en la Figura 4.26 y 4.27.

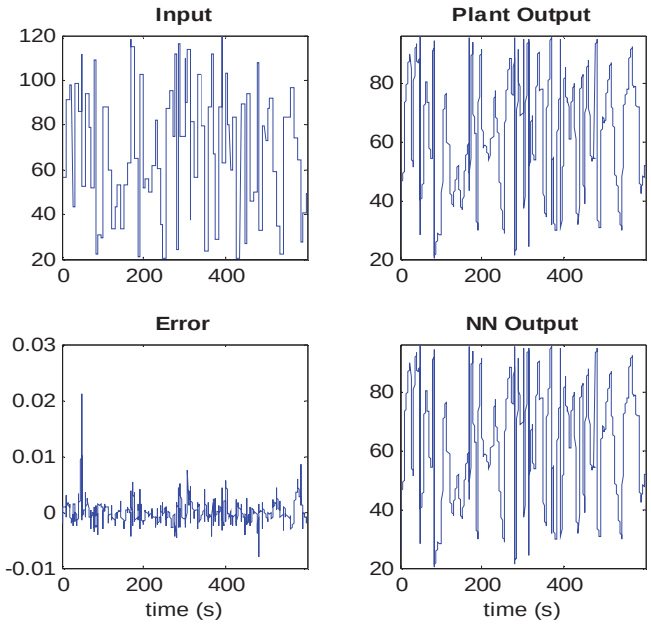


Figura 4.26 Entrenamiento L-M de la red neuronal con 16 neuronas en las capas

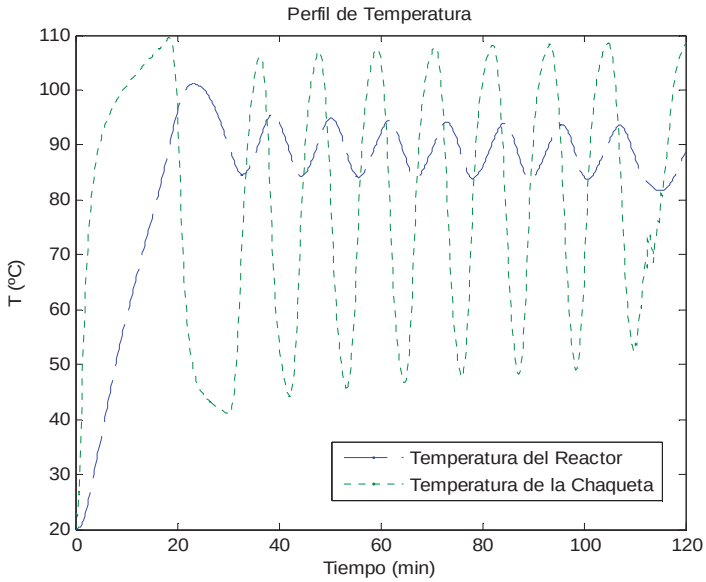


Figura 4.27 Perfil de Temperatura entrenamiento L-M

Si se incrementa el número de neuronas de 16 a 19 en las capas internas los resultados se pueden observar en las Figuras 4.28 y 4.29.

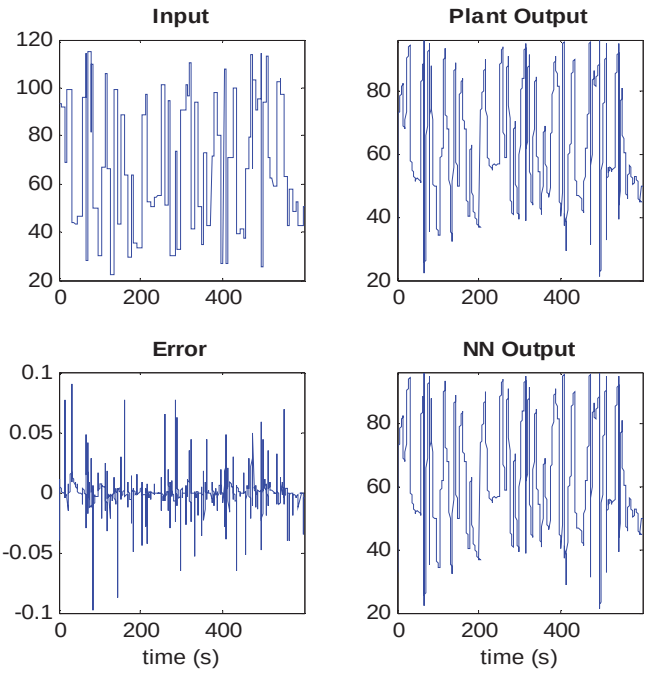


Figura 4.28 Entrenamiento L-M de la red neuronal con 19 neuronas en las capas

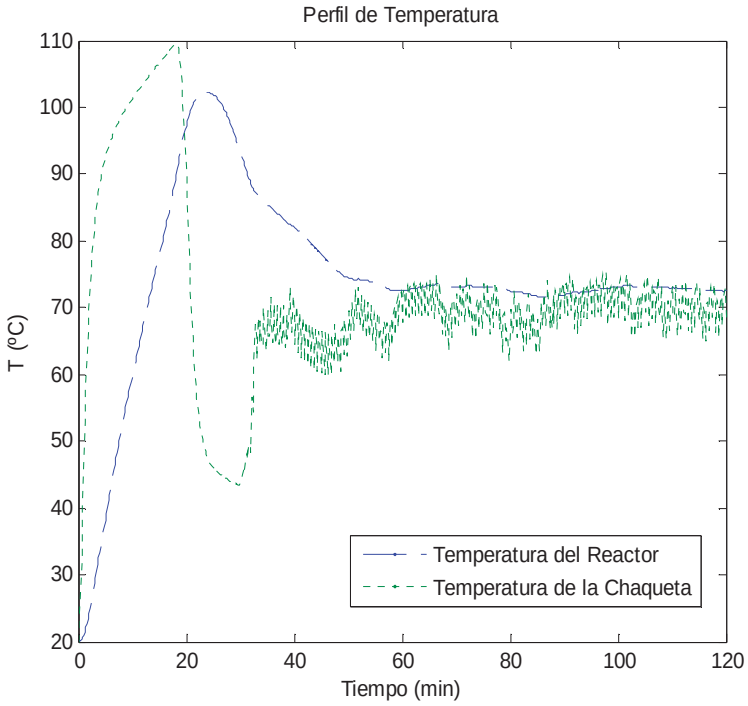


Figura 4.29 Perfil de Temperatura entrenamiento L-M

Realizando un incremento de 19 a 22 neuronas en las capas ocultas de la red neuronal los resultados se ven en la Figura 4.30 y 4.31.

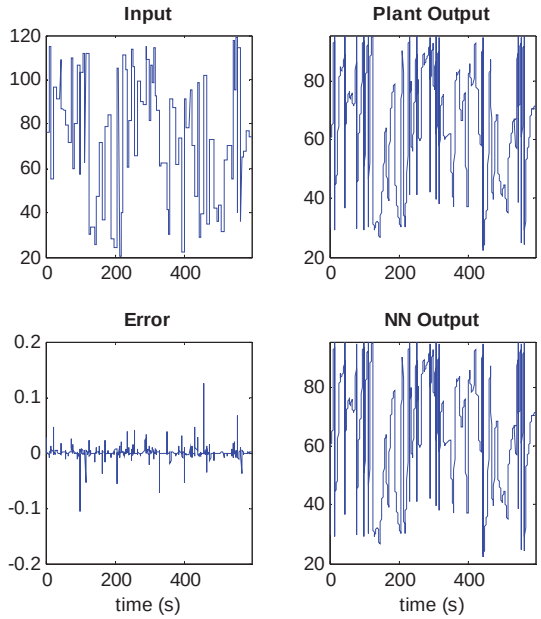


Figura 4.30 Entrenamiento L-M de la red neuronal con 22 neuronas en las capas

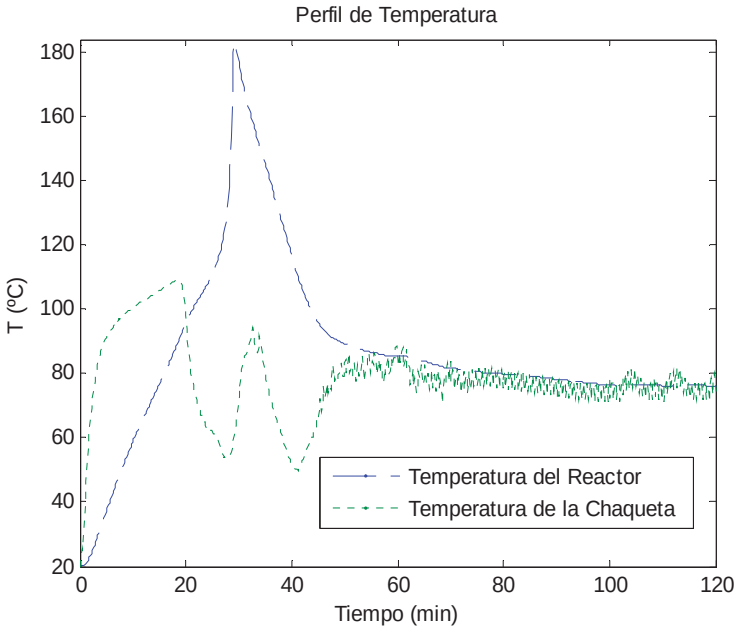


Figura 4.31 Perfil de Temperatura entrenamiento L-M

Se selecciona la red neuronal con una cantidad de 19 neuronas y cuatro capas ocultas, debido a que los resultados presentan una disminución del error y a su vez la disminución de las oscilaciones del control.

4.2 Comportamiento con un set point fijo

Los resultados del controlador GMC para un set-point fijo obteniendo un buen control de 92.46 ºC (Figura 4.32), una producción del producto deseado máxima y una mínima del producto no deseado (Figura 4.33).

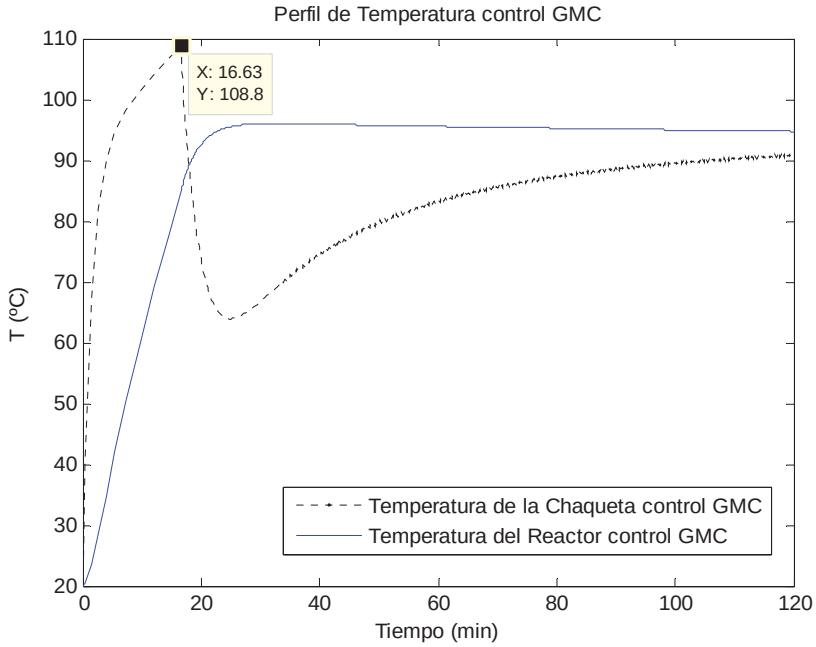


Figura 4.32 Perfil de Temperatura control GMC

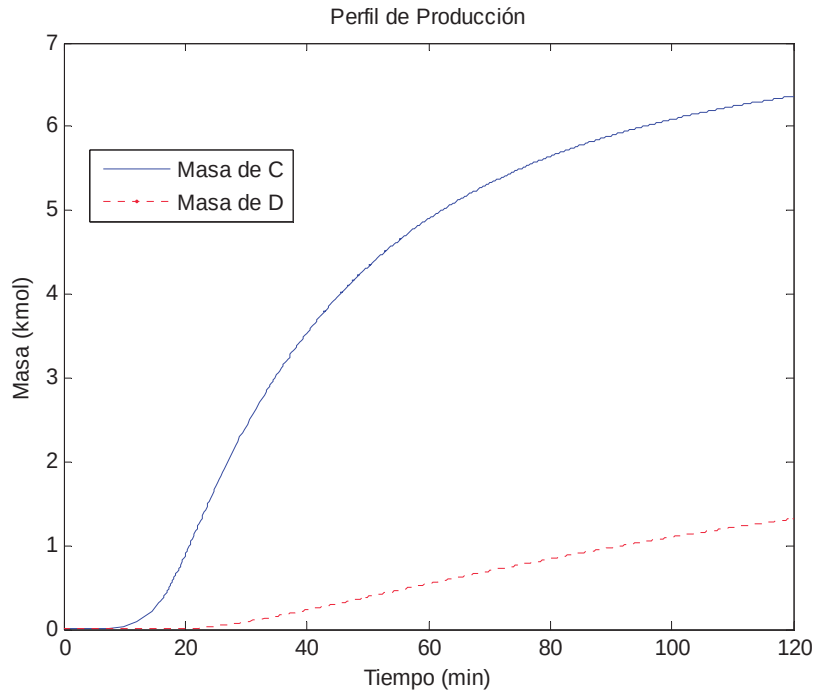


Figura 4.33 Perfil de Producción

El control con redes neuronales obtenidas hasta el momento presenta demasiada oscilación, por lo que es necesario mejorar el control con redes neuronales.

Se modificó la cantidad de neuronas, a un número de 19 neuronas en cada una de las capas que también se modificó a un número de 4 capas ocultas (Figura 4.34). Los límites de salida para el entrenamiento de la red neuronal fueron cambiados también (Figura 4.35), utilizando un entrenamiento acelerado como lo es el entrenamiento L-M esto para la minimización del error y así mismo evitar que nuestra red neuronal pueda presentar un sobre aprendizaje.

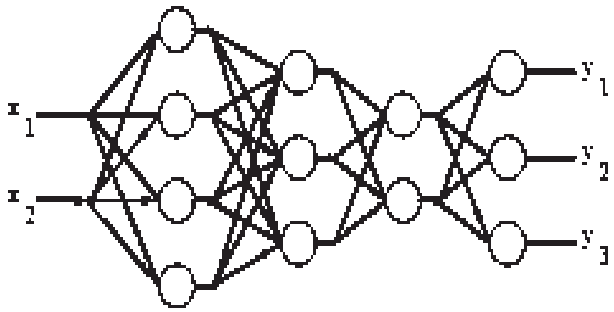


Figura 4.34 Estructura final de la red neuronal

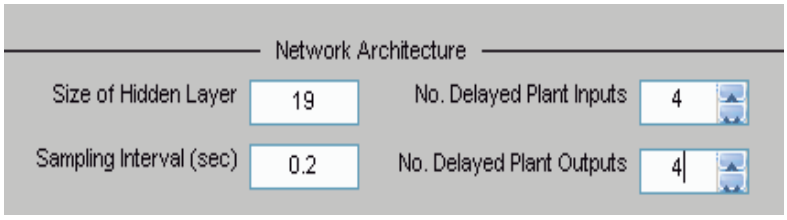


Figura 4.35 Estructura de la red neuronal en Matlab

En los perfiles de temperatura (Figura 4.36 a 4.40) se observa que con estas modificaciones en la red neuronal, comienza a presentar un comportamiento semejante al del control GMC, pero presentando aun en algunos de los perfiles de temperatura oscilaciones comunes en el tipo de control on-off siendo la Figura 4.40 la que proporciona un perfil muy semejante al que presenta el control GMC.

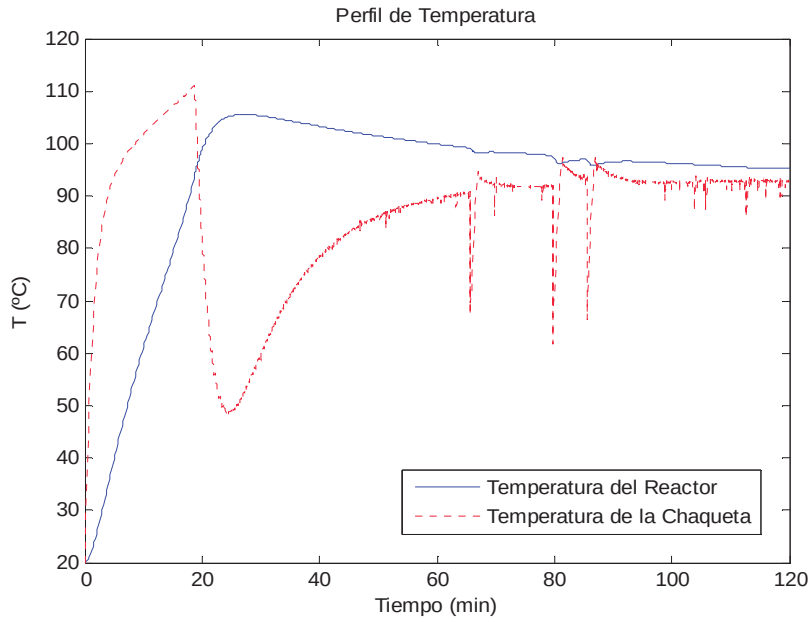


Figura 4.36 Perfil de Temperatura, limites de salida 93 – 90 ºC

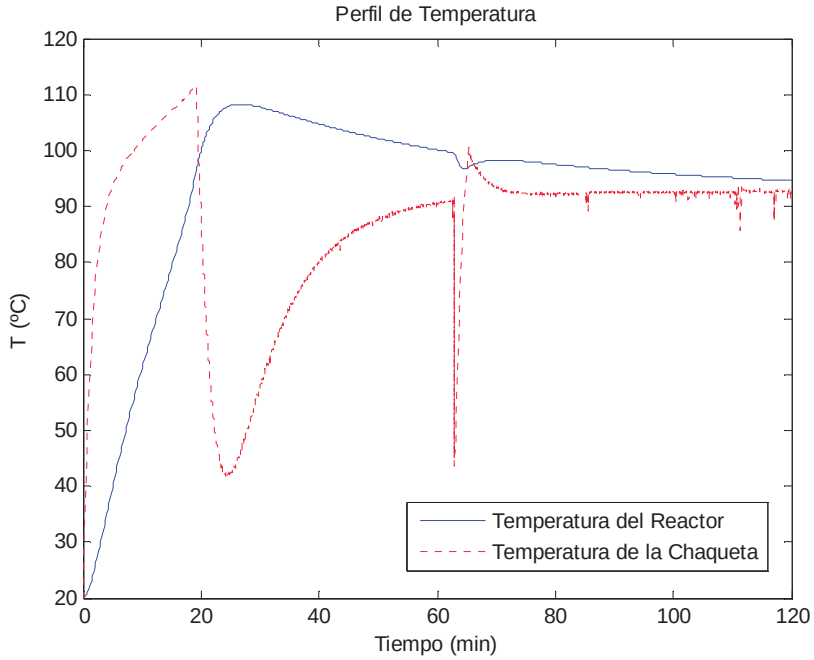


Figura 4.37 Perfil de Temperatura, limites de salida 93 – 70 ºC

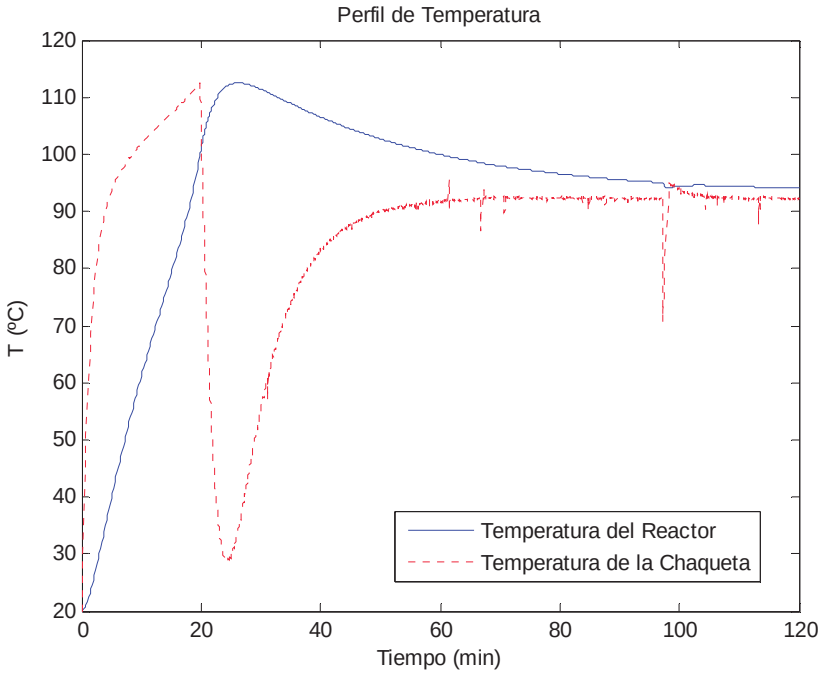


Figura 4.38 Perfil de Temperatura, limites de salida 93 - 40ºC

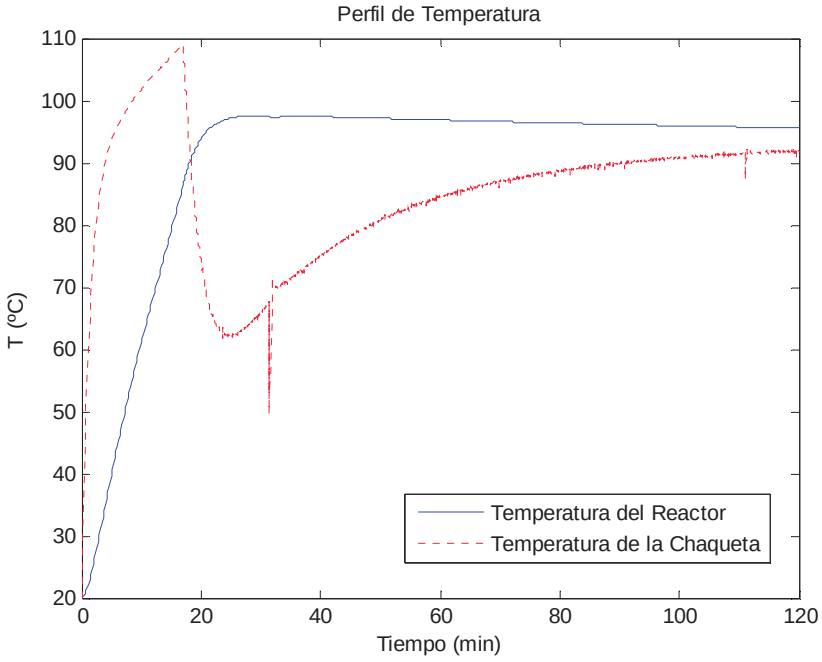


Figura 4.39 Perfil de Temperatura, limites de salida 93 – 20 ºC

El control con redes neuronales con un set point fijo, presenta resultados aceptables (Figura 4.40) observandose que las oscilaciones que se presentaban desaparecieron completamente.

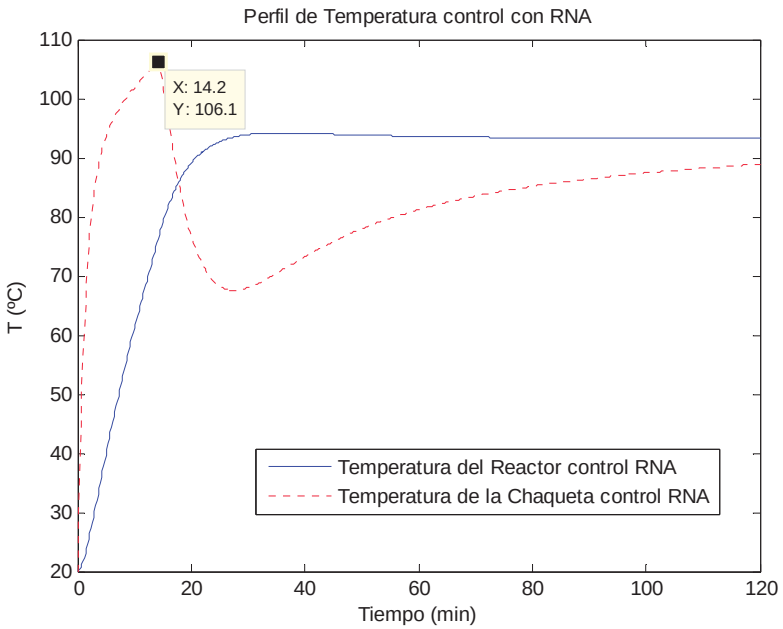


Figura 4.40 Perfil de Temperatura, sin límites a la salida de la red neuronal

4.3 Comportamiento con un set point escalonado

El análisis del control se hará introduciendo cambios en el set point con respecto al tiempo (Figura 4.41), los cambios se dan en tiempos de 40 min (de 92.46 ºC a 91 ºC) y 80 min (de 91 ºC a 93.46 ºC).

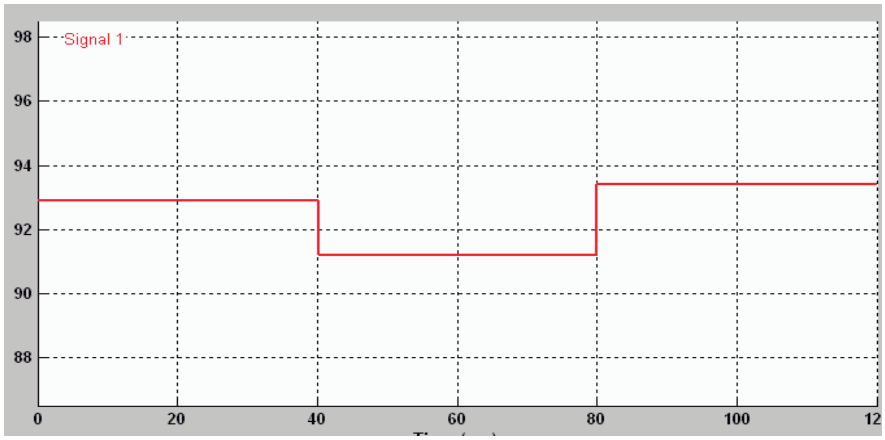


Figura 4.41 Cambio del set point con respecto del tiempo

Al introducir estos cambios en el set point en el controlador en base a redes neuronales se obtiene el de temperatura, en donde, se muestra claramente el control (Figura 4.42a y 4.42b) y los resultados con el control GMC (Figura 4.43a y 4.43b)

siendo este último el que presenta mejores resultados en comparación con el sistema de control con redes neuronales

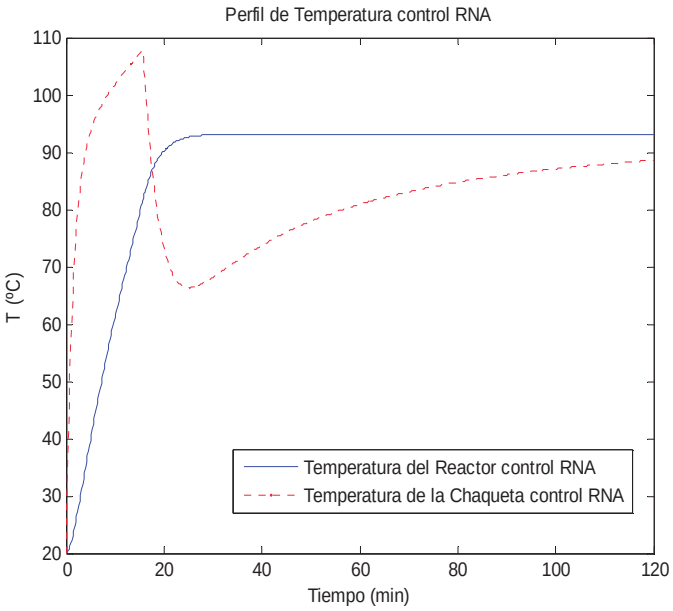


Figura 4.42a Respuesta de control RNA con cambios en el set point

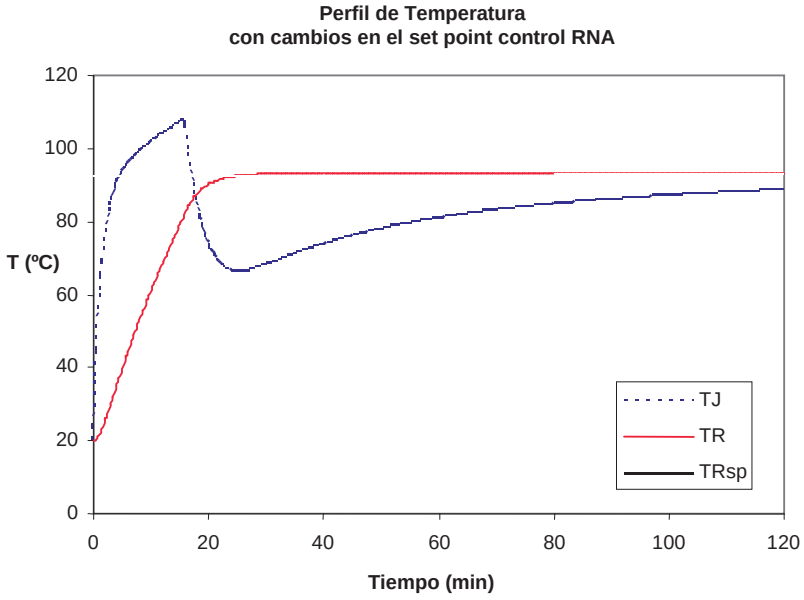


Figura 4.42b Respuesta de control RNA con cambios en el set point

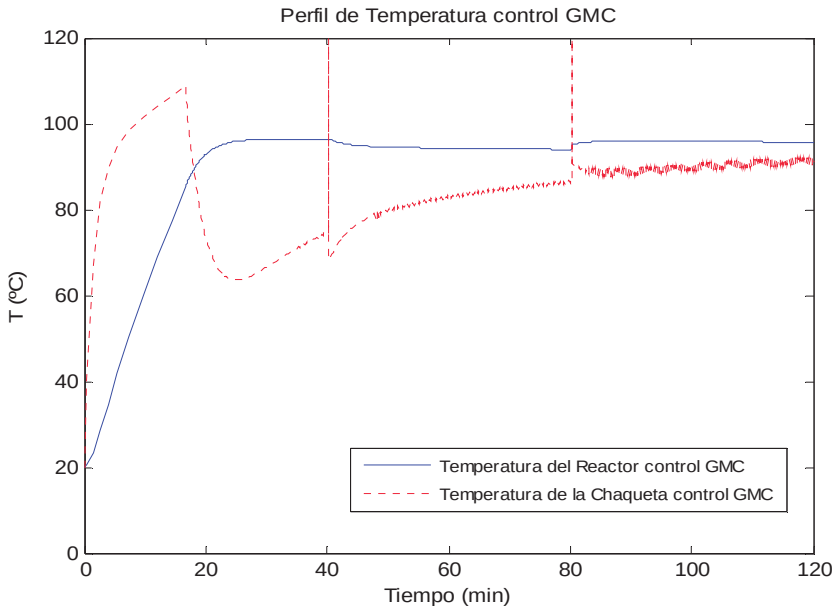


Figura 4.43a Respuesta de control GMC con cambios en el set point

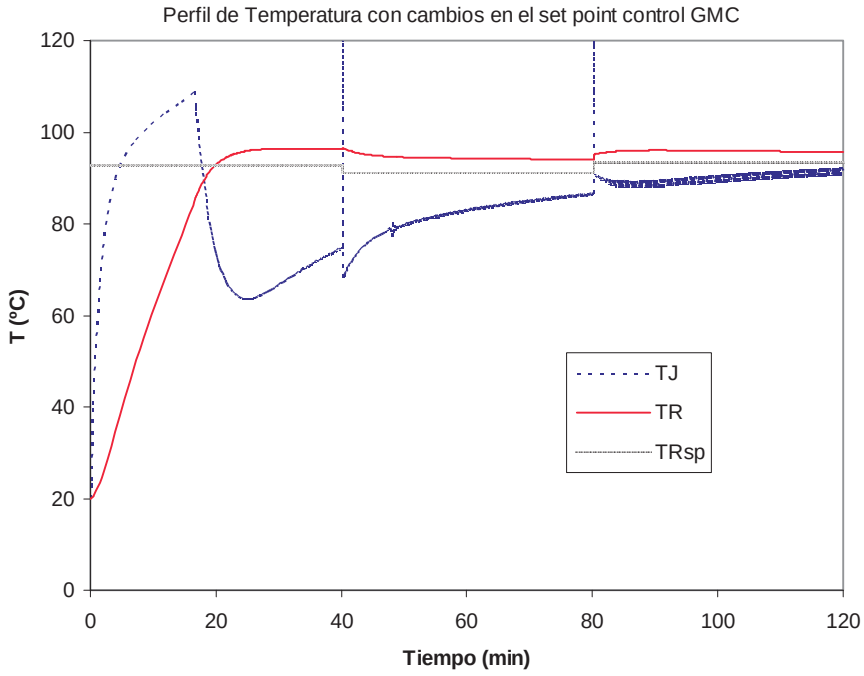


Figura 4.43b Respuesta de control GMC con cambios en el set point

En las Figuras 4.42a y 4.42b los perfiles de temperatura tienen una tendencia en la cual no asimilan los cambios en el set point. El resultado de este perfil se debe

principalmente a los limites de entrenamiento de la red neuronal (60 ºC – 93 ºC), ya que uno de los cambios en el set point sobrepasa el límite de la red neuronal, haciendo un cambio en los límites de la red neuronal de sólo dos grados centígrados (60 ºC – 95 ºC), no modificaron la respuesta del control cuando se tiene el set point fijo pero se mejora los resultados cuando se presenta cambios en el set point con respecto al tiempo (Figura 4.44) y este es mejor que el control GMC.

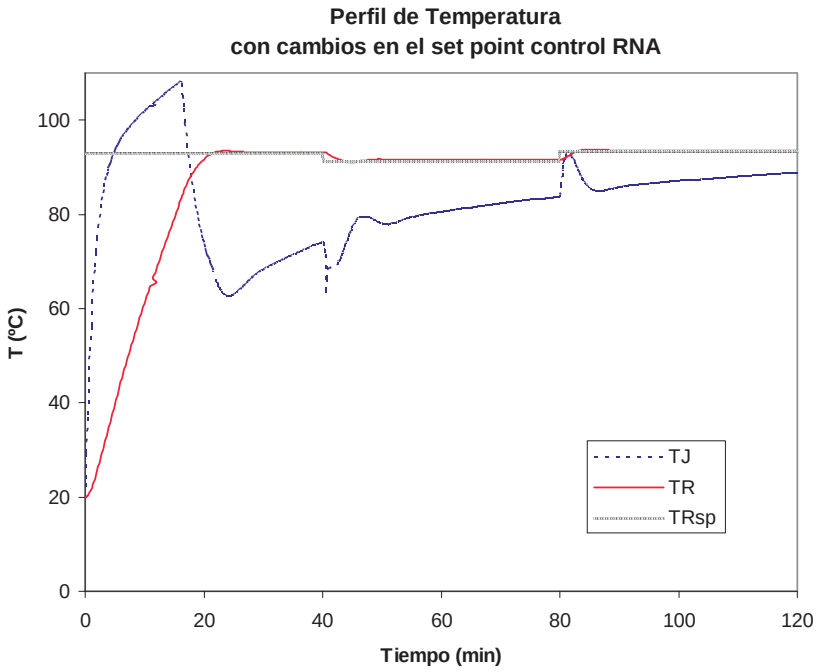


Figura 4.44 Respuesta de control RNA con cambios en el set point

4.4 Comportamiento con un set point rampa

El control con redes neuronales tiene la capacidad de asimilar la dinámica del proceso, un análisis más es el introducir cambios en el set point con cambios mas abruptos como se muestra en la Figura 4.45 (de 92.46 ºC a 64 ºC en un tiempo de 40 min), los cambios realizados en el set point son comprendidos en su totalidad por la red neuronal siguiendo el perfil del set point (Figura 4.46), teniendo un mejor resultado en comparación con el control GMC con los mismos cambios en el set point (Figura 4.47).

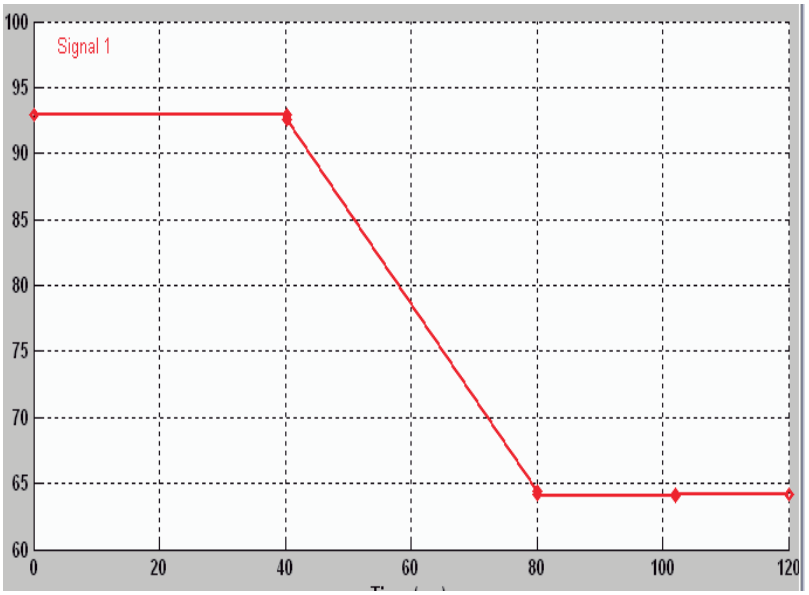


Figura 4.45 Cambios en el set point

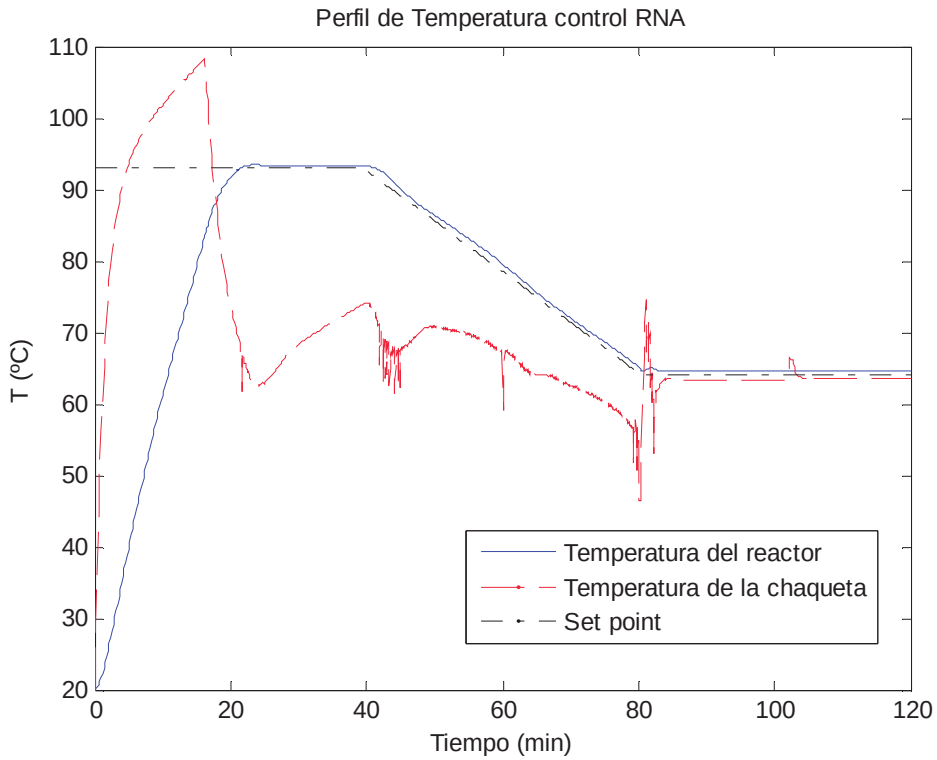


Figura 4.46 Respuesta de control de RNA con cambios en el set point

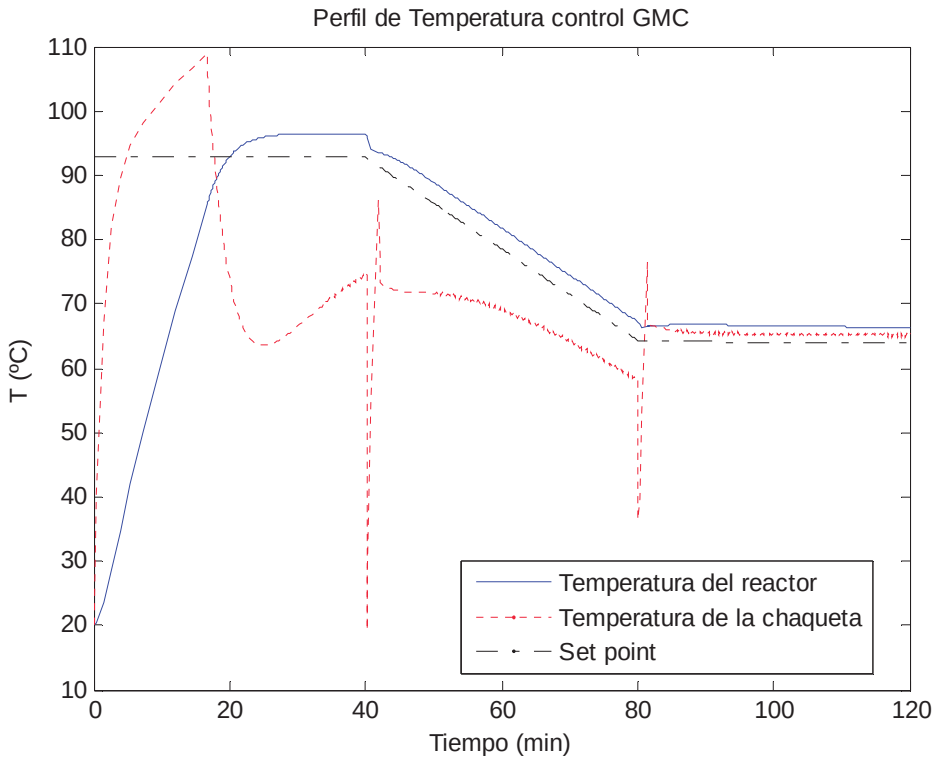


Figura 4.47 Respuesta de control GMC con cambios en el set point

CONCLUSIONES Y RECOMENDACIONES

Se hacen las conclusiones generales del trabajo así como las aportaciones del trabajo en el área de investigación del control mediante redes neuronales, dándose de igual manera una directriz hacia nuevos trabajos.

Conclusiones generales

1) En este trabajo se presentó el uso de redes neuronales para el control de un reactor batch exotérmico, sintetizando la red neuronal en función del número de neuronas internas y el número de capas ocultas, la parametrización de la red neuronal se hizo en el simulador comercial Simunlink de Matlab haciendo uso de una red neuronal del tipo de NARMAX (red neuronal recursiva) que está en función básicamente de una función de transferencia no lineal discreta en el tiempo que es usada por los investigadores para identificar sistemas continuos. En el trabajo presente la estructura de la red neuronal del tipo NARMAX se ha adecuado para un sistema del tipo batch o por lotes demostrando la hipótesis planteada al inició del presente trabajo, en donde los controladores que hacen uso de redes neuronales son una alternativa real para el control de sistemas batch.

2) Debe mencionarse que la elección de la mejor red neuronal se realiza mediante la modificación de los parámetros de la red neuronal implementada y de la respuesta del controlador.

3) Los resultados del uso de la red neuronal como controlador hace evidente que la respuesta del control es una respuesta del tipo on-off (oscilatoria), modificando la red neuronal tanto en el número de neuronas como en el número de capas ocultas se continúa presentando la misma respuesta de control on-off. Se modifica de manera secuencial el tipo de entrenamiento, el entrenamiento usado fue el entrenamiento de propagación hacia atrás (retropropagación) dicho entrenamiento presento un sobre aprendizaje el cual era evidente en la respuesta que el controlador estaba arrojando., Dicho sobre aprendizaje se dio principalmente en la minimización del error significando esto que el entrenamiento de propagación hacia atrás consideraba de manera sustancial los errores aprendiendo de ellos y haciéndolos parte de sus pesos específicos de entrada de la red neuronal.

4) La minimización del error para este tipo de redes neuronales recursivas se hace por medio de la modificación del tipo de entrenamiento. Se modificó el tipo de entrenamiento, de propagación hacia atrás a un entrenamiento del tipo acelerado o entrenamiento del tipo Levenberg-Marquardt (L-M) que tiene la cualidad de no arrastrar el tipo de errores que arrastra el entrenamiento del tipo de propagación hacia atrás, el cálculo del error lo hace mediante la discretización en función del tiempo.

5) Los resultados se mejoraron en gran medida pero aun presentaban oscilaciones; se optó por modificar el número de neuronas dentro de un intervalo de 7 neuronas hasta un número de 22 neuronas y un número de capas ocultas de 2 a 4 capas, se logró obtener un perfil de temperatura aceptable con un número de 19 neuronas y cuatro capas ocultas.

6) La red neuronal que mostró mejores resultados fue la red neuronal con un número de 19 neuronas, 6 capas (una capa de entrada, cuatro capas ocultas y una capa de salida), con limites de entrenamiento de 60 ºC a 95 ºC, como límites de entrada con dichos límites el entrenamiento fue realizado con la formulación de una matriz de pesos discretizados. Los pesos discretizados se obtienen con la simulación del modelo interno de la red neuronal (modelo del proceso).

7) Los resultados de la red neuronal como controlador para un set point fijo (92.46 ºC) es comparado con un controlador del tipo GMC (*Generic Model Control)* planteado por Mujtaba en el 2004. Los resultados de la red neuronal son ampliamente mejorados manteniendo el control en un punto de 93 ºC y un tiempo de respuesta de 14.2 min. a diferencia del control GMC que mantiene el control en una temperatura de 93.84 ºC y una respuesta a un tiempo de 16.63 min. y una disminución del sobre tiro para el control con redes neuronales.

8) La presencia de cambios en el set point pone a prueba con cambios en el control con redes neuronales y al control GMC, presentando mejor resultado el control con redes neuronales; asimilando en su totalidad la dinámica del proceso haciendo los cambios necesarios para seguir el perfil de cambios en el set point, a diferencia que con el control GMC.

Contribución del trabajo

La mayoría de los controladores para sistemas batch son controles del tipo PID (Proporcional Integral Derivativo) que pueden presentar fallas cuando el control PID presenta cambios muy abruptos con respecto al tiempo, ya que dichos controladores pueden controlar cambios muy pequeños con respecto al tiempo. Para cambios grandes el control PID falla. En el presente trabajo se ha demostrado que el control GMC presenta fallas de igual manera para cambios abruptos con respecto del tiempo. Las redes neuronales son sistemas inteligentes que no requiere el conocimiento de sistemas continuos como el control PID. La red neuronal crea su conocimiento del proceso en base a las entradas de la planta y actualizándose de acuerdo con la dinámica del proceso que pueda comprobarse en el trabajo presentado. Se considera que obtenidos los resultados de las simulaciones es posible llevar a aplicaciones el control con redes neuronales pues sólo es necesario, una computadora y una tarjeta de adquisición de datos para que usando datos; estadísticos del proceso sea posible el entrenamiento de la red neuronal. Se a comprobado que la redes neuronales son una alternativa para el control de procesos del tipo batch aun presentado cambios considerables durante la dinámica del proceso.

Recomendaciones para trabajo futuro

1) Las redes neuronales son sistemas considerados dentro de la inteligencia artificial siendo un campo que esta tomando importancia en las investigaciones actuales en el control de procesos. La combinación de las redes neuronales y la lógica difusa esta tomando un repunte importante en los últimos años. El neurocontrol difuso puede ser una alternativa nueva para el control de procesos del tipo batch haciendo un entrenamiento con diferentes funciones de membresía característica del control difuso.

2) La implementación de una nueva estructura de red neuronal que haga uso de un optimizador integrado dentro de la estructura de la red neuronal.

3) La modificación de la estructura interna de la red neuronal, esto para evitar los problemas de sobreaprendizaje común en redes recurrentes o en redes predectivas que hacen uso de un optimizador.

4) Las redes neuronales son sistemas altamente factibles para su uso en sistemas híbridos, siendo esto un nuevo campo de investigación para el control con redes neuronales.

5) Se recomienda que se realice un análisis robusto del sistema de control propuesto en base a redes neuronales.

BIBLIOGRAFIA

Agarwal, 1997, “A systematic calssification of neural networks based control”. IEEE Control System, vol. 4, pp 0272-1708.

Almeida, 1987, “A learning rule for asynchronous perceptrons with feedback in a combinatorial environment” L.B. Almeida IEEE 1st Int. Conf. on Neural Networks, San Diego, CA, vol. 2, pp.609-618

Almeida, 1988, “Backpropagation in perceptrons with feedback” L.B. Almeida Neural Computers (R. Eckmiller, C. von der Malsburg, eds) NATO ASI Ser., New York: Springer Verlag, vol. 1 pp. 199-208.

Anderson, 1999, “Collective computational abilities” J.J. Hopfield Proc. National Academy of Science, USA, vol.79, pp.2554 – 2558.

Cybenko G., 1989, “Continuos Value Neural Networks with two Hidden Layers are Sufficient” , Math Control Signal & Systems, vol. 2, pp. 303-314.

Daosud W., Thitiyasook P., Arpornwichanop A., Kittisupakorn P., 2005, “Neural network inverse model based controller for the control of steel pickling process”. Computers and Chemical Engeneering, vol. 29, pp. 2110-2119.

Demuth & Beale, 1992, “Neural Network Toolbox User’s Guide (For Use with MATLAB™)” H. Demuth, M. Beale The Math Works, Inc. 1992

Dirion J-L., Ettedgui B., Le Lann M. V., 1996, “Development of adaptive neural networks for flexible control batch process”. Chemical Engeneering Journal, pp. 63-65.

Drakopoulos, 2005, “Training neural networks whit heterogeneous data”. Neuronal Networks. Vol. 18. pp. 595-601.

Faggin, 1991, “VLSI implementation of neural networks” F. Faggin International Joint Conference on Neural Networks. Seattle, WA, 1991

Fogler H. Scott, 2001, “Elementos de Ingeniería de las Reacciones Químicas” Ed. Prince Hall, Tercera Edición, pp. 34-35.

Galvan I.M., Zaldivar J.M. 1999, “Control in batch reactors using controllers PID hibrys” . Chemical Engineering and Processing vol. 23.

Hecht-Nielsen, 1990, “On the algebraic structure of feedforward network spaces” Advanced neural computers (R. Eckmiller, ed.) North-Holland, Amsterdam, Netherlands, vol. 29, pp.129- 135.

Hunt. K.J.,Sbarbaro D., Zbikowski R., y Gawthrop P.J., 1992, "Neuronal Networks for Control Systems-A survey", Automatica, vol.28, no 6, pp. 1083-1112.

Lee et al, 1988, "Generic model control", Computers Chemical Engeneering, vol. 12, pp. 573.

Luus, 1994, "Optimal control of reactors by iterative dynamic programming". Journal Process Control, vol. 4, pp. 218.

Luyben W.L., 1998, "Tuning Temperature Controllers on Open loop Unestable Reactors", American Chemical Society, vol. 37, pp. 4322-4331.

Luzzy & Dengel, 1993, "A comparison of neural net simulators" O. Luzzy, A. Dengel IEEE Expert, vol.24, pp. 390.

Macchietto S., Cott B. J., 1989, "Temperature control of exotermic batch reactors using generic model control". Ind. Eng. Chem. Res, vol.28,1117.

Maren J., Abdulkader A., 1990, "Handbook of Neural Computing Applications" A. Maren, C. Harston, R. Pap Academic Press Inc.

Mujtaba, 2004, "Performance of diferent types of controllers in tracking optimal temperature profiles in batch reactors". Computing Chemical Engeneering, vol. 24, pp. 1069.

Nahas E. P., Henson M. A., Seborg D. E., 1992, "Nonlinear internal model control strategy for neuronal networks models". Computers Chemical Engeneering. Vol. 16. Nº 12 pp. 1039-1057.

Parker, 1985, "Learning-logic: Casting the cortex of the human brain in silicon" D.B. Parker Tech. Rep. TR-47, Center for Computational Research in Economincs and Management Science, MIT, Cambridge, MA.

Parker, 1987, "Optimal algorithms for adaptive networks: Second order backpropagation, second order direct propagation and second order Hebbian learning" D.B. Parker IEEE 1st Int. Conf. on Neural Networks, , San Diego, CA,vol.2, pp.593-600.

Pineda, 1987, "Generalization of backpropagation to recurrent neural networks" F.J. Pineda Physical Review Letters, vol. 59, pp.2229-2232.

Pineda, 1988, "Generalization of backpropagation to recurrent and higher order neural networks" F.J. Pineda Neural Information Processing Systems, vol.1, pp. 2290

Ramón y Cajal, 1911, "Histología del sistema nervioso del hombre y de los vertebrados" Santiago Ramón y Cajal Consejo Superior de Investigaciones Científicas Madrid.

Rumelhart D. E., Grakdopoulos N., 1986a, “Learning representations by back-propagating errors” D.E. Rumelhart, G.E. Hinton, R.J. Willliams Nature (London), vol.323, pp.533-536, 1986

Rumelhart D. E., Grakdopoulos N., 1986b, “Learning internal representations by error propagation” D.E. Rumelhart, G.E. Hinton, R.J. Willliams Parallel Distributing Processing: Explorations in the Microstructure of Cognition (D.E. Rumelhart, J.L. McClelland, eds), vol.1, chapter 8, Cambridge, MA: MIT Press, 1986

Sanz M. A., 2005 “Redes Neuronales y Sistemas Difusos” Ed. Alfaomega.

Smith-Corripio, 1996 “Control Automatico de Procesos” Ed. Limusa. p. 277

Werbos, 1974 “Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Science” P. Werbos Ph. D. dissertation, Harvard University, Cambridge, MA, 1974

Werbos, 1988 “Generalization of backpropagation with application to a recurrent gas model” P. Werbos Neural Networks, vol.1, pp.339-356.

Werbos, 1989 “Backpropagation and neurocontrol: A review and prospectus” P. Werbos Proc. Int. Joint Conf. on Neural Networks, Washington DC

ANEXO 1

CONTROL CON REDES NEURONALES PARA UN REACTOR BATCH.