ESTUDIO Y ANALISIS DE NEUROCONTROLADORES APLICADOS A SERVOSISTEMAS

Autores

ALVARO LOBELO DIAZ
JAMES ESCOBAR MEZA

CORPORACIÓN UNIVERSITARIA TECNOLÓGICA DE BOLÍVAR
FACULTAD DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA, Y MECATRONICA
CARTAGENA DE INDIAS
2003

ESTUDIO Y ANALISIS DE NEUROCONTROLADORES APLICADOS A SERVOSISTEMAS

ALVARO LOBELO DIAZ
JAMES ESCOBAR MEZA

Monografía, presentado para optar al título de Ingeniero Electricista

Director

EDUARDO GOMEZ VASQUEZ

Magíster en Ciencias Computacionales

Asesor

ORLANDO TINOCO ARAQUE

Ingeniero Electrónico

CORPORACIÓN UNIVERSITARIA TECNOLÓGICA DE BOLÍVAR
FACULTAD DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA, Y MECATRONICA
CARTAGENA DE INDIAS
2003

Nota de aceptación

Firma de presidente del jurado

Firma del Jurado

Firma del jurado

Cartagena de Indias 30 de Mayo 2003

CONTENIDO

		Pág
	INTRODUCCIÓN	
1.	**FUNDAMENTACION DEL NEUROCONTROLADOR**	1
1.1	INTRODUCCIÓN A LOS NEUROCONTROLADORES	1
1.2	CLASIFICACIÓN DE LOS NEUROCONTROLADORES	3
1.2.1	Modelado básico o directo de la planta	4
1.2.2	Modelo inverso neuronal	5
1.2.3	Parametrizador de controladores basado en RNA	7
1.2.4	Modelado del controlador	8
1.2.5	Neurocontrolador libre de modelo	10
1.2.6	Control predictivo	11
1.3	COMPARACIÓN ENTRE EL NEUROCONTROLADOR Y EL PID	15
2.	**DESARROLLO DEL NEUROCONTROLADOR DE UN SERVOSISTEMA**	17
2.1	EXPLICACION	17
2.2	CARACTERÍSTICAS DEL CONTROLADOR PID DEL MOTOR DC	19
2.3	ENTRENAMIENTO DE LA RED NEURONAL	25
2.3.1	Adaline	25
2.3.2	Backpropagation	29
2.3.3	Programacion de las redes neuronales	31
2.4	DETERMINACIÓN DE LA TOPOLOGÍA DE LA RED NEURONAL	38
2.4.1	Variando número de neuronas para una sola capa oculta	39
2.4.2	Variando el error esperado en el entrenamiento	42
2.3.3	Variando el de número de neuronas para dos capas ocultas	45
2.4.4	Variando el número de iteraciones	46

Pág

2.4.5	Entrenamiento de la red Adaline	48
2.5	COMPROBACION	49
2.6	OBSERVACIONES DE LA ETAPA DE COMPROBACION	60
2.7	VALIDACIÓN DEL NEUROCONTROLADOR ELEGIDO	61
3.	**APLICACIONES DE LOS NEUROCONTROLADORES**	64
3.1	Neurocontrolador para un sistemas mecánicos	64
3.2	Neurocontrol como herramienta en la atención Farmacéutica	74
3.3	Modelamiento y control neuronal en tiempo real de una planta tipo cónica para el control de nivel	82
3.4	Predicción de demanda de transporte por carretera mediante el empleo de redes neuronales artificiales	96
3.5	Control de servosistemas no lineales usando redes Neuronales	106
3.6	Aplicación de redes neuronales artificiales en la modelización del tratamiento térmico de alimentos	129
3.7	Sensores virtuales mediante redes neuronales Artificiales	146
4.	CONCLUSIONES	158
	BIBLIOGRAFÍA	160
	ANEXOS	162

LISTA DE FIGURA

Pág.

Figura 1.1 Modelamiento directo de la planta 4
Figura 1.2 Modelamiento inverso de la planta. 6
Figura 1.3 Parametizador del controlador. 7
Figura 1.4 Modelado del controlador. 9
Figura 1.5 Neurocontrol libre de modelo. 11
Figura 1.6 Control Predictivo. 13
Figura 2.1 Diagrama de bloque del servosistema. 18
Figura 2.2 Señal del set-point. 19
Figura 2.3 Señal de salida del servosistema con el PID. 20
Figura 2.4 Diagrama de bloque del servosistema utilizado en MATLAB. 21
Figura 2.5 Salida del servosistema sin el PID 22
Figura 2.6 Estructura de la red Adaline 25
Figura 2.7 ADALINE de una neurona y dos entradas. 25
Figura 2.8 Característica de decisión de una red tipo Adaline 26
Figura 2.9 Estructura de una red Backpropagation. 28
Figura 2.9 Grafica de entrenamiento de la primera red neuronal. 32
Figura 2.10 Grafica de entrenamiento de la segunda red neuronal. 32
Figura 2.11 Grafica de entrenamiento de la tercera red neuronal. 33
Figura 2.12 Grafica de entrenamiento de la cuarta red neuronal. 33
Figura.2.13 Variación del no de neuronas vs error cuadrado 38
Figura 2.14 Variación del No Neuronas VS No Iteraciones 39
Figura 2.15 Variación Del Error 42

Pág.

Figura 2.16 Salida del servosistema con el PID 48

Figura 2.17 Diagrama de bloque del neurocontrol 49

Figura 2.18 Salida servosistema, datos sin normalizar, error propuesto 0.1, No neurona 5 capa oculta 50

Figura 2.19 Salida del servosistema, No neuronas capa oculta 2, error propuesto 0.0001 52

Figura 2.20 Salida del servosistema, No neuronas capa oculta 4, error propuesto 0.0001 53

Figura 2.21 Salida servosistema, No neuronas capa oculta 12, error propuesto 0.00001 54

Figura 2.22 Salida servosistema, No neuronas 1 Capa 8, 2 Capa 4, error propuesto 0.00001 55

Figura 2.23 Salida servosistema, No neuronas 8, No iteraciones 400 55

Figura 2.24 Salida del servosistema, sin normalizar los datos de entrenamiento. 56

Figura 2.25 Salida del servosistema, Red Adaline, con datos normalizados para el entrenamiento. 57

Figura 2.26 Señal de referencia del servosistema 59

Figura 2.27 Salida del servosistema con el Neurocontrol 60

Figura 2.28 Salida del servosistema con el PID 60

Figura 3.1 Modelo de un robot caminante. 63

Figura 3.2 Pendulo Invertido 65

Figura 3.3 Equipo del péndulo invertido 66

Figura 3.4 Red neuronal de tres capas. 69

Figura 3.5 Esquema general del controlador 70

Pág.

Figura 3.6 Error y razón de aprendizaje de MNN 71

Figura 3.7 Posición angular 71

Figura 3.8 Posición del carro. 71

Figura 3.9 Aplicaciones de las redes neuronales. 73

Figura 3.10 Variabilidad farmacológica. 74

Figura 3.11 Diagrama esquemático del neurocontrolador en base al modelo inverso. 82

Figura 3.12 Estructura de la red neuronal. 82

Figura 3.13 Esquema de adquisición de datos de entrenamiento. 83

Figura 3.14 Diagrama P&ID del sistema. 84

Figura 3.15 Fotografía del sistema implementado. 84

Figura 3.16 Pantalla del sistema de control en Labview. 86

Figura 3.17 Control con modelo neuronal inverso. 88

Figura 3.18 Control mediante un controlador PID. 89

Figura 3.19 Arquitectura de una red Backpropagation. 94

Figura.3.20 Función que realiza una neurona artificial. 95

Figura 3.21 Número total de operaciones de transporte público y privado con destino la Comunidad Autónoma de Aragón. (Período 93-96) 96

Figura.3.22 Ajuste obtenido en la estimación del modelo predictivo (Método Holt-Winters). 97

Figura 3.23 Topología seleccionada**.** 99

Figura 3.24 Variación del error durante el aprendizaje 100

Figura 3.25 Comparación de resultados 101

Figura 3.26 Red neuronal de tres capas 106

Figura 3.27 Esquema general del sistema implementado 107

Pág.

Figura 3.28 Esquema general del driver implementado. 108

Figura 3.29 Esquema de la planta a controlar. 109

Figura 3.30 Modelo de identificación serie-paralelo 112

Figura 3.31 La primera figura muestra la velocidad de salida sin carga en el motor y la segunda con carga. 117

Figura 3.32 La primera figura muestra el error cuadrático promedio durante el entrenamiento y la segunda la fricción de Coulumb identificada 118

Figura 3.34 Error por muestra al final del entrenamiento y error Cuadrático promedio durante el entrenamiento. 121

Figura 3.35 Compensación neuronal del motor con y sin carga 122

Figura 3.36 Control PID de un motor sin carga. 123

Figura 3.37 Control PID de un motor con carga. 124

Figura 3.38 Error en el seguimiento de la trayectoria para un motor con Carga 125

Figura 3.39 Perfil de temperatura del autoclave 134

Figura 3.40 Perfil de temperatura del autoclave con perturbación 134

Figura 3.41 Error cuadrático en numero de nodos de las capas ocultas. 136

Figura 3.42 Suma del error cuadrático en función del número de Iteraciones 139

Figura 3.44 Validación simulados versus predichos. 140

Figura 3.45 Temperatura del autoclave en función del tiempo. 140

Figura 3.46 Perfil de temperatura 142

Figura 3.47 RNA de fusión. 149

Figura 3.48 Resultados de la estimación de biomasa con la red de fusión para la fermentación F181 151

Figura 3.49 Estimación de biomasa del sistema de análisis temporal para la fermentación F181. 151

LISTA DE TABLAS

Pág.

Tabla 2.1 Entrenamiento de redes 34

Tabla 2.2 Variación del no de neuronas 37

Tabla 2.3 Variación Del Error 41

Tabla 2.4 Variación del no de neuronas para dos capas 43

Tabla 2.5 Variación Del No Iteraciones 45

Tabla 2.6 Validación 48

Tabla 3.1 Conjuntos de datos 77

Tabla 3.2 Estimación de los parámetros del modelo Holt-Winters 98

Tabla 3.3 Errores cuadráticos medios de test para las distintas topologías (Con 12 entradas). 99

Tabla 3.4 Comparación de errores cuadráticos medios 102

Tabla 3.5 Motor Error cuadrático 125

Tabla 3.6 Condiciones operacionales 142

Tabla 3.7 Resumen de resultados de biomasa. 151

Tabla 3.8 Resumen de los resultados de la primera y segunda etapas de estimación para las cuatro corridas de la fermentación. 153

INTRODUCCIÓN

En recientes años, ha habido un creciente incremento en la aplicación de las redes neuronales en el control automático Adicionalmente la información en el idioma español sobre este tema es poca en nuestro medio, y paralelamente está técnica de control ha adquirido una gran relevancia como tecnología emergente. Por tal motivo se considera pertinente realizar una investigación bibliográfica a nivel de monografía sobre los neurocontroladores en servosistemas, y otros campos del control automático.
Para el desarrollo del neurocontrolador se hizo una clasificación de los distintos neurocontroladores, tomando como referencia el campo y la forma en que se aplica en el control automático.

Esta investigación tiene como fin mostrar como se logra diseñar un neurocontrolador utilizando el toolbox de redes neuronales de MATLAB, partiendo de la simulación de un control PID del servosistema. Al final se simula el neurocontrolador con el servosistema, con el fin de hacer la comparación entre estos dos controles.

Adicionalmente se creo una página web para que presentara en una forma didáctica las principales aplicaciones de los neurocontroladores, y la forma en que se diseño el neurocontrolador del servosistema, y al final se presenta las ventajas que presenta neurocontrol sobre el PID.

1. FUNDAMENTACION DEL NEUROCONTROLADOR

1.1 INTRODUCCIÓN A LOS NEUROCONTROLADORES

El neurocontrol es un dinámico campo de investigación que en los últimos años ha atraído una considerable atención en la comunidad científica y de ingeniería de control. En recientes años, ha habido un creciente incremento en la aplicación de las redes neuronales en identificación de la dinámica (modelado), predicción y control de sistemas complejos. 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 sistemas complejos sea modelada y un control exacto sea logrado a través del entrenamiento, sin tener información a priori sobre los parámetros del sistema[1].

Este método de control se hace necesario cuando el control clásico no presenta un buen desempeño, lo que tradicionalmente ocurre en proceso multivariables o tipo no lineal, como las redes neuronales tienen características de realizar un mapeo exacto de funciones altamente no lineales pueden ser una gran solución para este tipo de problemas, considerando que esto no es una tarea fácil.

El problema del control de ciertas variables físicas en un proceso industrial, consiste en la determinación de variables de control y dispositivos que permitan mantenerlo operando en los niveles deseados y en una forma adecuada, a

[1] FREEMAN, James y SKAPURA, David. Redes Neuronales: Algoritmos, aplicaciones y técnicas de programación. Delaware E.U.A; Addison Wesley Iberoamericana S.A. 1993

pesar de influencias externas (perturbaciones) que actúen en sentido contrario o tiendan a desestabilizarlo. Sin embargo al control de un proceso industrial, no se llega en forma arbitraria o casual, por el contrario, se llega a él, por la necesidad de satisfacer de la mejor forma posible los múltiples objetivos que pueda perseguir una industria (beneficio económico, seguridad del proceso o del personal, conservación energética, etc.) ya sea a nivel de un subproceso aislado o de una planta completa o por que la naturaleza del proceso así lo exige.

El pionero en la investigación sobre la identificación y control de sistemas usando redes neuronales ha sido K.S. Narendra quien describe en el año de 1990 como la dinámica de sistemas no lineales pueden ser modeladas mediante redes neuronales, tanto redes tipo feedforward como redes recurrentes[2].

Los principales uso que se le han dado al neurocontrolador son los de modelamiento de la planta, predicción y control de sistemas complejos, estos aspectos se discutirán mas adelante.

[2] K.S., Nerendra. Identification and control of dynamic systems using neural networks. IEEE Trans. On neural networks. Vol 1, No. 1. 1990.

1.2 CLASIFICACIÓN DE LOS NEUROCONTROLADORES

La investigación de las redes neuronales aplicadas a control ha implementado un importante número de soluciones en el control industrial. Las aplicaciones exitosas más relevantes incluyen desde elementos que operan como identificadores hasta aquellos que trabajan como controladores de optimización utilizando modelos del proceso.

Este trabajo tiene como objetivo generar un marco referencial para el desarrollo de controladores basados en redes neuronales artificiales. Dado el acelerado crecimiento de las propuestas en los últimos dos decenios, la clasificación siguiente no puede ser tomada como absoluta, sino por el contrario, muchos ajustes podrán hacerse en virtud de nuevas arquitecturas, modificaciones a las existentes ó nuevos modelos híbridos. En dicha distribución se ha tratado de proponer una clasificación óptima para los sistemas de neurocontrol. Es importante señalar que dependiendo de la escuela o el autor de la clasificación, es común encontrar distintas denominaciones para una estructura similar, ó bien un mismo nombre para diferentes estructuras de control.

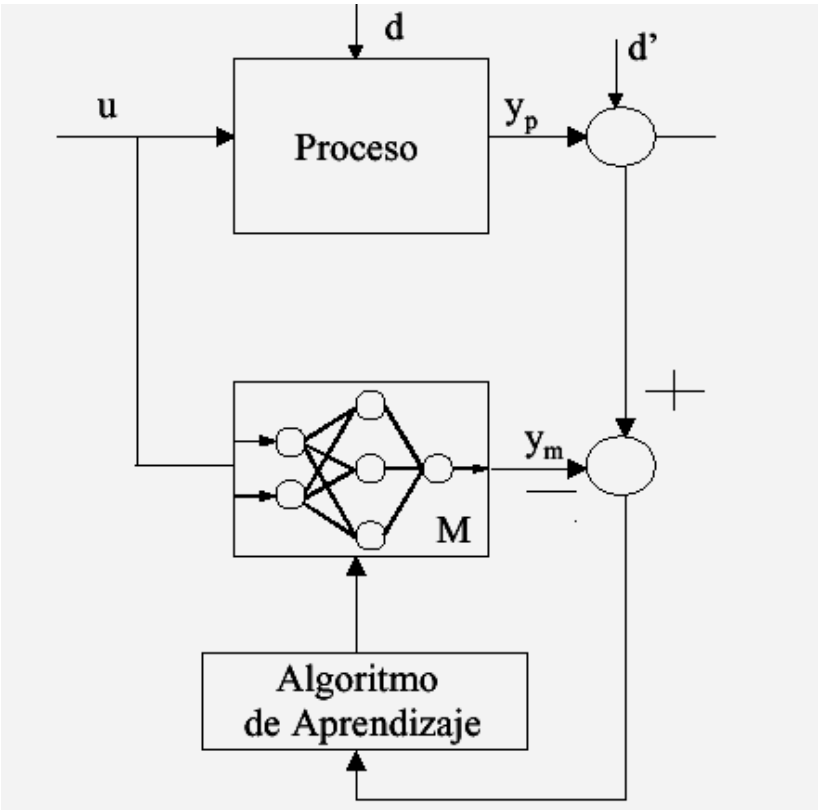
Los neurocontroladores más sobresaliente son los siguientes:

- ✓ · Modelado básico o directo de la planta.
- ✓ · Modelo inverso neuronal.
- ✓ · Parametizador de controladores basado en Redes Neuronales Artificiales.
- ✓ · Modelado del controlador.
- ✓ · Neurocontrolador libre de modelo.
- ✓ · Control predictivo

1.2.1 Modelado básico o directo de la planta.

En este caso, se entrena una red neuronal de manera de obtener la dinámica directa de la planta. En este esquema se conecta en paralelo la RNA a la planta, el error de identificación entre la salida de la red y la salida de la planta, es la señal que se utiliza para el algoritmo de aprendizaje de la red neuronal para ajustar sus pesos con la finalidad de que la salida de la red se aproxime al de la planta, tal como se muestra en la siguiente figura:

Figura 1.1 Modelamiento directo de la planta



El modelo arriba descrito, puede ser representado por la siguiente ecuación, para un sistema no lineal (proceso):

$$Yp(t+1) = F\,[Yp(t),...,Yp(t-n+1),U(t),\ldots,U(t-n+1)] \qquad 1.1$$

Donde Yp(t) y U(t) representa la salida y entrada de la planta en el tiempo t, y n es un numero entero mayor que cero.

Par el modelamiento de la red neuronal se trabaja con la siguiente ecuación:

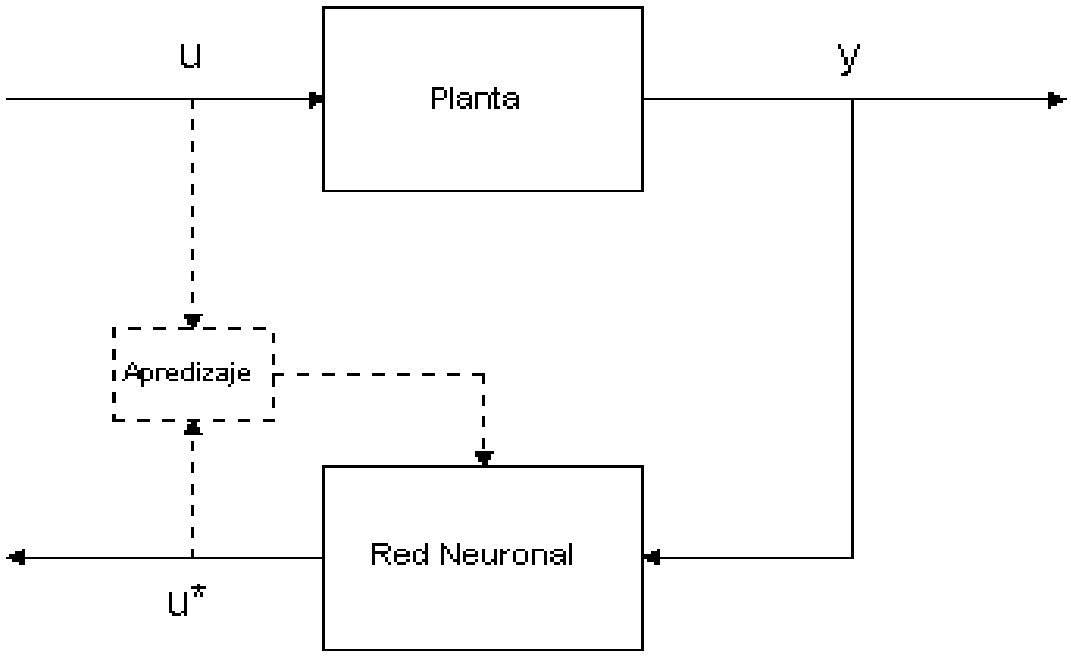
$$Ym(t+1) = F'\,[Yp(t),...,Yp(t-n+1),U(t),\ldots,U(t-n+1)]. \qquad 1.2$$

Donde F' es la relación de entrada y salida por la red neuronal, luego de un tiempo adecuado de entrenamiento se tiene $Ym \approx Yp$ de esta manera la red se independiza de la planta, obteniendo la modelación directa de la planta.

1.2.2 Modelo inverso neuronal. 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. De esta manera la red actúa directamente como controlador. Este tipo de esquemas es común en aplicaciones de robótica. Claramente la confiabilidad de este esquema, radica en la fidelidad del modelo inverso[3], la siguiente figura muestra el diagrama de este método:

[3] Doris Saes (Marzo, 2002). Apuntes II. Control basados en redes neuronales. Seminario AADECA-UBA, Buenos Aires.

Figura 1.2 Modelamiento inverso de la planta.



Donde u * representa la entrada deseada a la planta para una salida deseada.

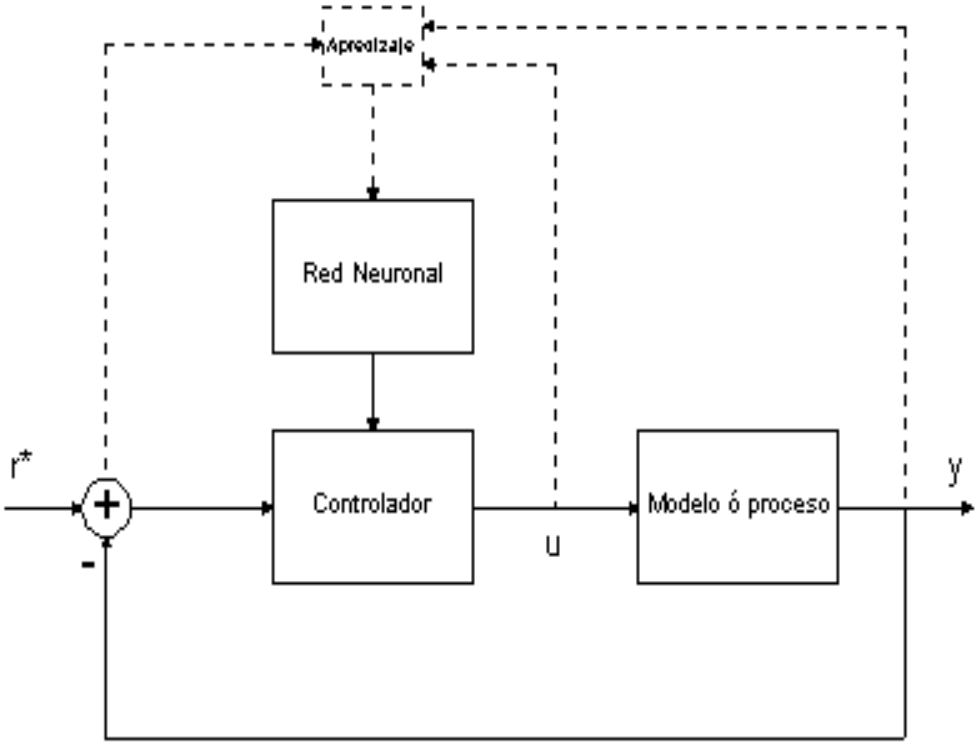
Como se puede observar la red neuronal es entrenada en línea, en este caso, no existe un conjunto finito de datos de entrenamiento La red es entrenada permanentemente mientras es ejecutada con una secuencia de datos potencialmente infinita, y los parámetros de la red son actualizados en cada instante de muestreo. Por esta razón, normalmente el sistema converge rápidamente luego de las primeras iteraciones.

Típicamente el modelo inverso es una estructura de estado estable, la cual puede ser usada para control. Dado un punto de operación deseado **y***, la señal apropiada de control **u*** en estado estable para este punto de operación debe de ser inmediatamente conocida.

1.2.3 Parametizador de controladores basado en redes neuronales artificiales. Para este caso la red se utiliza para mejora el controlador (PD, PID) o en su efecto 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 autoajustador ó parametizador[4]. En la siguiente figura se muestra el diagrama de bloque:

Figura 1.3 Parametizador del controlador.

[4] Diseño y simulación de un sistema de control mediante métodos de espacio de estado, Universidad De Guadalajara.
http://proton.ucting.udg.mx/materias/moderno/modulo5.htm



El ajuste del controlador PID es actualmente un procedimiento manual, a veces largo. 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 controlador PID.

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. 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.

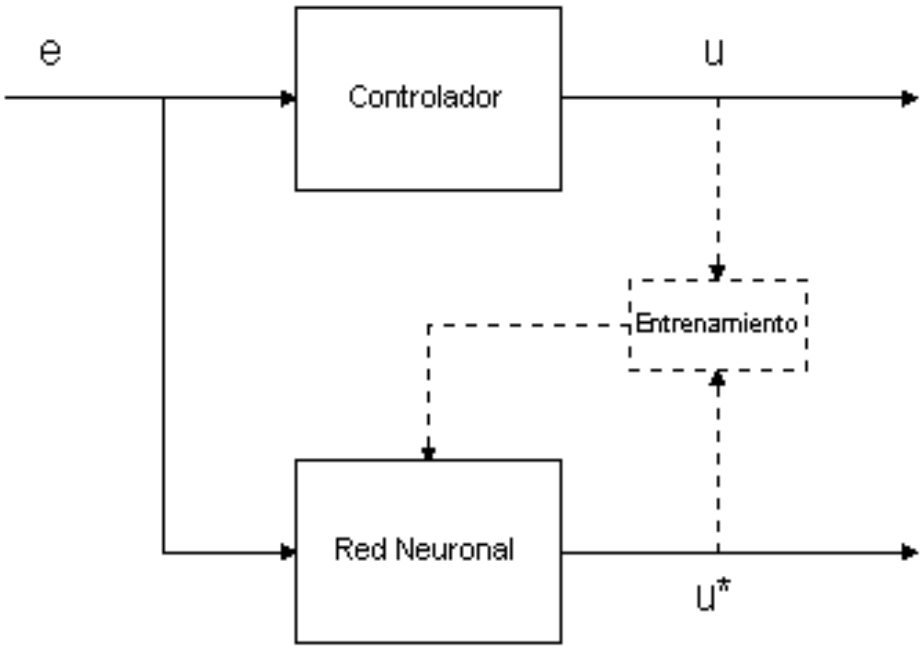
La mayoría de los trabajos realizados para autoajustar ó parametiizar controladores se han desarrollado directamente para PID, debido a que actualmente son utilizados

en una amplia gama de aplicaciones. Valores apropiados en las ganancias del PID son esenciales para que el sistema de lazo cerrado trabaje adecuadamente.

1.2.4 Modelado del controlador. Este modelo consiste en usar una red neuronal para modelar un controlador existente 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.

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 ARX (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. Para un mejor entendimiento se puede observar la siguiente figura:

Figura 1.4 Modelado del controlador.



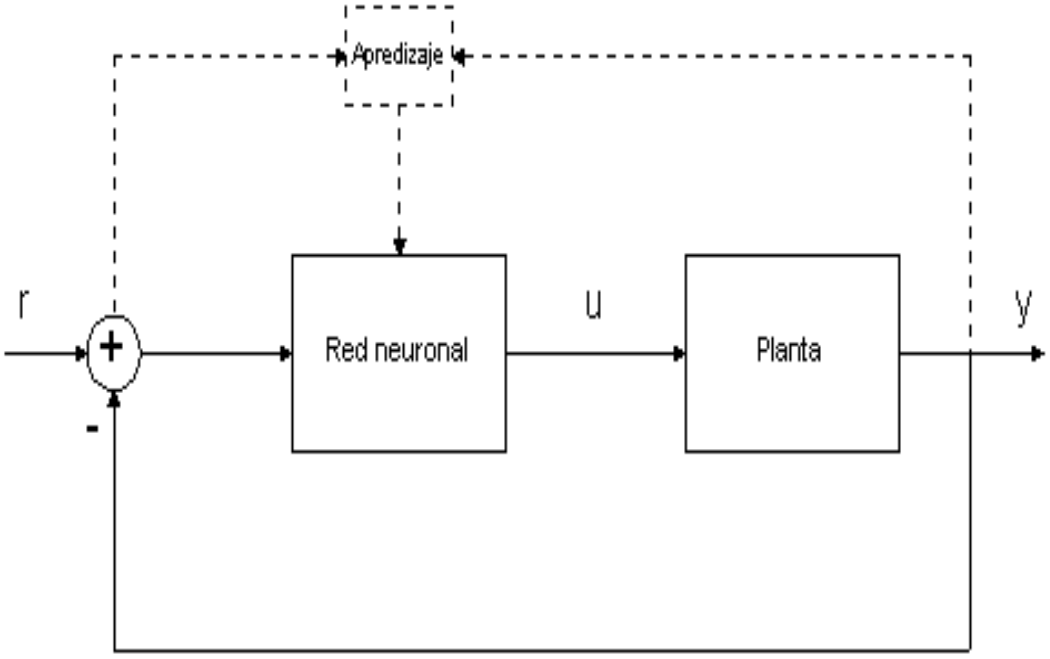
En general este tema puede resultar en controladores que son rápidos y/ó baratos en comparación con los controladores existentes. Por ejemplo usando este tipo de controladores Pomerleau presenta una aplicación intrigante donde una red neuronal es usada para reemplazar a un controlador existente: el operador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

Hay que resaltar que este método 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.

Observando la importancia de este modelo de neurocontrolador se desarrollo en esta investigación de un controlador de un servosistema, esta parte se discutirá con mayor detalle mas adelante.

1.2.5 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. Para este caso requerimos conocimiento sobre el proceso de interés, Si el modelo del procesó no es disponible, uno puede primero entrenar una segunda red neuronal para modelar la dinámica de planta como se presentó en el MODELO NEURONAL DIRECTO. 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. El modelo es el que se muestra en la siguiente figura:

Figura 1.5 Neurocontrol libre de modelo



Para tener en cuenta los neurocontroladores adoptivos 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.

1.2.6 Control predictivo. Aunque en el pasado podía considerarse que el único objetivo del control consistía en mantener una operación estable del proceso, actualmente muchos de los problemas de control industrial no son simples y existe una progresiva complejidad de los procesos industriales con múltiples requisitos para sus sistemas de control, estos problemas constituyen un incentivo para el desarrollo de novedosas y sofisticadas estrategias de control que han dado origen a los denominados sistemas de control avanzado tales como el neurocontrolador y el

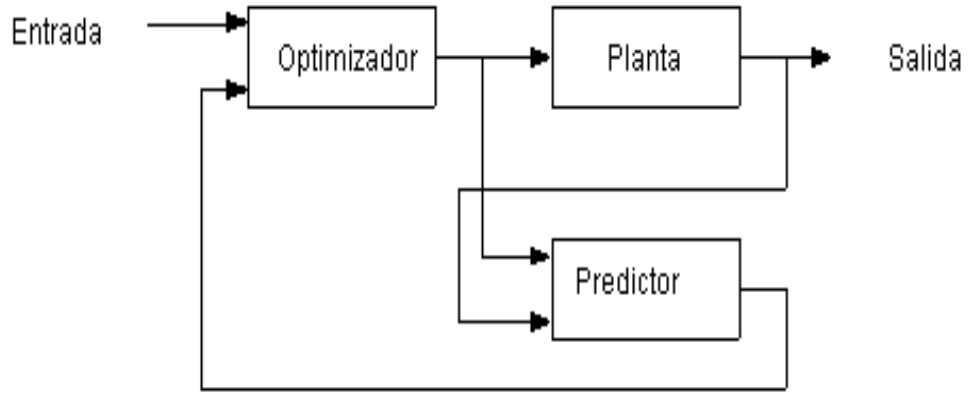
control predictivo[5]. El control predictivo fue desarrollado fundamentalmente en base a dos líneas principales. Por un lado diversos algoritmos que surgieron a finales de los años 70, los cuales usaban explícitamente un modelo dinámico del proceso para predecir el efecto de futuras acciones de control en la salida, y por otro sobre la base de las ideas del control adaptativo. Existen diversas estrategias de control predictivo, tales como el Control Predictivo Basado en modelos (MPC) que aparece a finales de los 70, el Control Predictivo Generalizado (GPC) desarrollado a mediados de los 80 por el grupo del profesor D. W. Clarke en la Universidad de Oxford, o el Dynamic Matrix Control (DMC) desarrollado a fines de los años 70 por ingenieros de Shell Oil Co. Todas estas estrategias son muy similares y se basan en los mismos principios diferenciándose esencialmente en los objetivos de control, el tipo de procesos a controlar y las funciones de coste.

En resumen el control predictivo busca 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 sobre un horizonte específico.

La arquitectura requiere el desarrollo de un modelo de la planta o modelo de preedición, y una técnica de optimización la cual puede determinar la mejor acción de control. Este esquema se muestra en la siguiente figura:

[5] D**iseño e implementación de un sistema de control avanzado para** procesos complejos. Autor: José Gallardo Arancibia, U Católica Del Norte Chile.
http://isisu02.usal.es/~airene/capit9.pdf

Figura 1.6 Control predictivo



Las redes neuronales son usualmente usadas para obtener el modelo de predicción del proceso a ser controlado. Esta opción se basa en la bien conocida capacidad de las redes neuronales para memorizar funciones no lineales, una vez que han sido entrenadas. El tipo de arquitectura de red a utilizar por el esquema de control predictivo es una clásica red perceptron multicapa, donde la dinámica del sistema, se ve representada por ventanas temporales pasadas, tanto de entradas, como de salidas del sistema que son alimentadas a la entrada de la red neuronal. Es así que el modelo neuronal para el sistema desconocido, puede ser expresado por la siguiente ecuación:

$$Yp(t+1) = F[Y(t), Y(t-1),.....,Y(t-n+1),u(t),u(t-1),.....u(t-m+1)] \qquad 1.3$$

Donde $Yp(t+1)$ es la salida predicha, f representa el mapeo “entrada-salida” no lineal de la red, $[u(t), y(t)]$ representan el par [entrada, salida] del proceso y 'n', 'm' la cantidad de retardos en la salida y entrada respectivamente.

Asumiendo que redes multicapa 'feedforward', con al menos una capa oculta, tienen la capacidad de aproximar bastante bien cualquier función no lineal, se ha elegido una clásica red 'backpropagation' con una capa oculta y funciones de activación 'tangente hiperbólica' en la capa oculta y lineal en la capa de salida, para representar el proceso a controlar. De esta forma, la red que asume el papel de predictor puede ser representada por la siguiente ecuación:

$$y_p(t+1) = w_2[tanh(w_1 p + b_1)] + b_2 \qquad 1.4$$

Donde w1, w2, b1 y b2 son las matrices de pesos y vías y 'p' es el vector de entrada a la red definido por:

P= [Y(t), Y(t-1),.....,Y(t-n+1),u(t),u(t-1),…..u(t-m+1)] 1.5

En cuanto al optimizador el tiene como misión determinar la mejor acción del control, además el optimizador tiene otra funciones tales como se muestra en el modelo presentado, es la de enviar una señal de control que alimenta la planta y también esta señal se aprovecha para alimentar a una red neuronal que a futuro pueda desempeñar la función de red controladora, ya sea como respaldo. La red neuronal así entrenada, sería una aproximación del modelo inverso de la planta, dependiente de la función de coste a minimizar.

1.3 COMPARACIÓN ENTRE EL NEUROCONTROLADOR Y EL PID

El neurocontrolador es una nueva tecnología aplicada a la ingeniería de control, para mirar las ventajas y desventajas que ofrece esta técnica con respecto a las antes utilizadas tal como el PID, se recopilo información de autores y trabajos desarrollados en redes neuronales como herramienta de control automático, además en la siguiente sección se desarrollo un controlador de un servosistema en matlab para demostrar la importancia de esta técnica.

Como primer aspecto a discutir cuando se desea elegir un controlador, se necesita conocer la dinámica o función de transferencia del sistema a controlar sea esta mecánica, eléctrica, o química, etc. Pero si el sistema es muy complejo de determinar o no hay leyes físicas que modelen el sistema a controlar se hace difícil elegir un controlador que pueda brindar un control deseado, este inconveniente se viene presentando en el tratamiento térmico continuo de alimentos, o en el control del PH. El neurocontrol es una buena solución cuando se presenta estos inconvenientes, ya que se puede usar una red neuronal para modelar la planta, luego se recopila las distintas necesidades del sistema y con estos parámetros se crea un neurocontrolador, el neurocontrolador libre de modelo recrea esta aplicación.

Otra observación importante es cuando se desea sintonizar el PID, y el proceso se ve sometido a cambios, o cuando hay un cambio en el setpoint del mismo, los parámetros de sintonización puede que no provean una óptima recuperación del sistema ante la presencia de perturbaciones, en algunos casos, la ganancia del proceso puede aún variar con la magnitud de la perturbación, también puede variar si hay una desviación del valor de la variable controlada con respecto al setpoint. Si el neurocontrolador en su etapa de entrenamiento recibe un buen numero de ejemplos, el neurocontrolador puede brindar un control deseado aun cuando es sometido a las variaciones ya citadas.

En el área de los sistemas mecánicos se ha presentado como un problema de control el caso cuando el número de entradas (actuadores) es menor que el número de salidas (grados de libertad). Un ejemplo de ello son los robots subactuados (sistemas con menos controles que variables a controlar), el péndulo invertido es un ejemplo clásico en donde la velocidad lineal del carro representa la entrada del sistema, la posición angular del péndulo y lineal del carro son las correspondientes salidas medibles del sistema. Las redes neuronales puede se una herramienta útil para darle una solución a este problema, Esta nueva salida es diseñada mediante técnicas de redes neuronales que garantizan el desempeño del controlador propuesto.

Para los sistemas no lineales es evidente y esperado que el uso de controladores PID no es la mejor alternativa para controlar un sistema no lineal, si el sistema no es linealizado en torno a un cierto punto de operación, el uso de un controlador PID produce sobreimpulsos transientes de muy elevada magnitud que pueden dañar los sensores del sistema si esto no se prevé utilizando protectores. Es aquí cuando entran en juego las redes neuronales artificiales, que tienen la característica de realizar un mapeo exacto de funciones altamente no lineales, por lo que su aplicación en el control automático es inmediata.

Entre las desventajas que ofrecen las redes neuronales es cuando los datos que se utilizan para el entrenamiento de la red no son generalizados o no se tiene un amplio rango operacional de datos, entonces la red probablemente no responderá con la eficacia adecuada para su aplicación, esto se presenta para casos en los cuales en la planta no se le puede hacer un modelamiento matemático, porque en el caso en que este modelamiento se pudiera realizar la red se podría entrenar con datos hechos en simulación matemática.

2 DESARROLLO DEL NEUROCONTROLADOR DE UN SERVOSISTEMA

El servosistema consiste en un controlador PID, el cual controla la velocidad angular de un motor DC, por medio del voltaje de armadura. Este sistema pude ser usado para mover el brazo de un robot, o controlar una banda transportadora, etc.

2.1 EXPLICACIÓN

Se desea controlar la velocidad angular a un motor DC shunt por medio de su voltaje de armadura, para lo cual se realizara una simulación en MATLAB. Mediante la ecuación de transferencia 2.4 y el diagrama de bloque de la figura 2.1. En donde se realiza la simulación de un controlador PID en el cual podemos leer y registrar los datos en la entrada y la salida de este controlador, para que posteriormente estos datos sean normalizados y así entrenar los neurocontroladores con unos datos bastante generalizados.

Para la modelizacion del controlador PID de un servosistema, se diseñaron diferentes redes backpropagation y una sola red ADALINE. Los datos de entrenamiento de la red es el error obtenido entre la señal de referencia y la salida del servosistema y la señal de respuesta del PID, además fue necesario introducir perturbaciones para lograr un mejor entrenamiento de la red; en total se obtuvieron

4000 datos para el entrenamiento, Para la realización de esta investigación se utilizo Matlab, tanto para el entrenamiento de la red como para la simulación del servosistema.

Para el entrenamiento de la red neuronal se recopilo información de los datos de entrada y salida del PID, luego estos datos fueron normalizados para que la red neuronal tuviera un mejor comportamiento cuando se estuviera entrenado, adicionalmente se entrenaron cinco redes con datos no normalizados de las cuales cuatro son Backpropagation y una Adaline con el fin de hacer una comparación entre las redes que fueron entrenadas con datos normalizados y no normalizados.

Con el fin de determinar cual va ser la red neuronal que va modelar el PID, se tiene que analizar como es el comportamiento del servosistema cuando trabaja con el neurocontrolador, la mejor forma de evaluar el funcionamiento del neurocontrolador consiste en aplicar los mismo valores usados para entrenar la red neuronal con el fin de comparar como es la respuesta del servosistema trabajando con el PID y el neurocontrolador.

Una vez escogida la red neuronal que va modelizar el comportamiento del PID, a esta red se le introducen nuevos datos en la señal de referencia, para observar como es el comportamiento a estos nuevos eventos, al final se compara con el PID y se determina si el neurocontrol exhibe un mejor desempeño en el servosistema que el mostrado por el PID, la señal que se va a comparar no es la presentada en la salida del PID o neurocontrolador si no en la señal de salida del servosistema debido

a que es mas fácil de interpretar la señal de salida del servosistema que la mostrada por el PID

2.2 CARACTERÍSTICAS DEL CONTROLADOR PID DEL MOTOR DC

El motor DC tiene como función de transferencia la siguiente ecuación[6]:

$S(Js+b)\ \theta(s) = K\ I(s)$ 2.1

$(Ls+R)I(s) = V-Ks\,\theta(s)$ 2.2

$$\frac{\hat{\theta}}{V} = \frac{K}{(Js+b)(Ls+R)+K^2}$$ 2.3

Donde:

$\theta(s)$ = velocidad angular

I(s) = corriente de armadura

K = constante del par

b = coeficiente de amortiguamiento

J = momento de inercia

V = voltaje de armadura

R = resistencia armadura

L = inductancia de armadura

Los valores utilizados para la representación del motor fueron los siguientes valores:

[6] CHAPMAN, Stephen J. Máquinas Eléctricas. Santafé de Bogotá Colombia. McGraw Hill. 1997

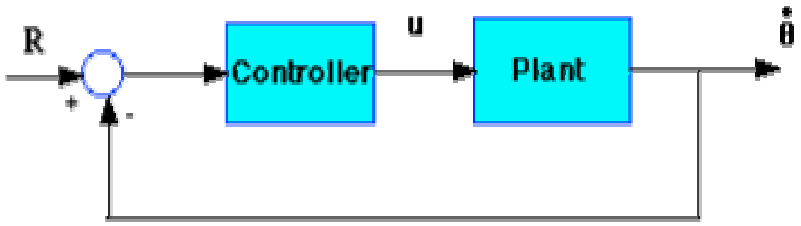
J=0.01
b=0.1
K=0.01
R=1 Ω
L=0.5mH

La función de transferencia de transferencia queda de la siguiente forma:

$$\frac{0.01}{0.005s^2+0.06s+0.1001} \qquad 2.4$$

El diagrama de bloque del servosistema es el siguiente:

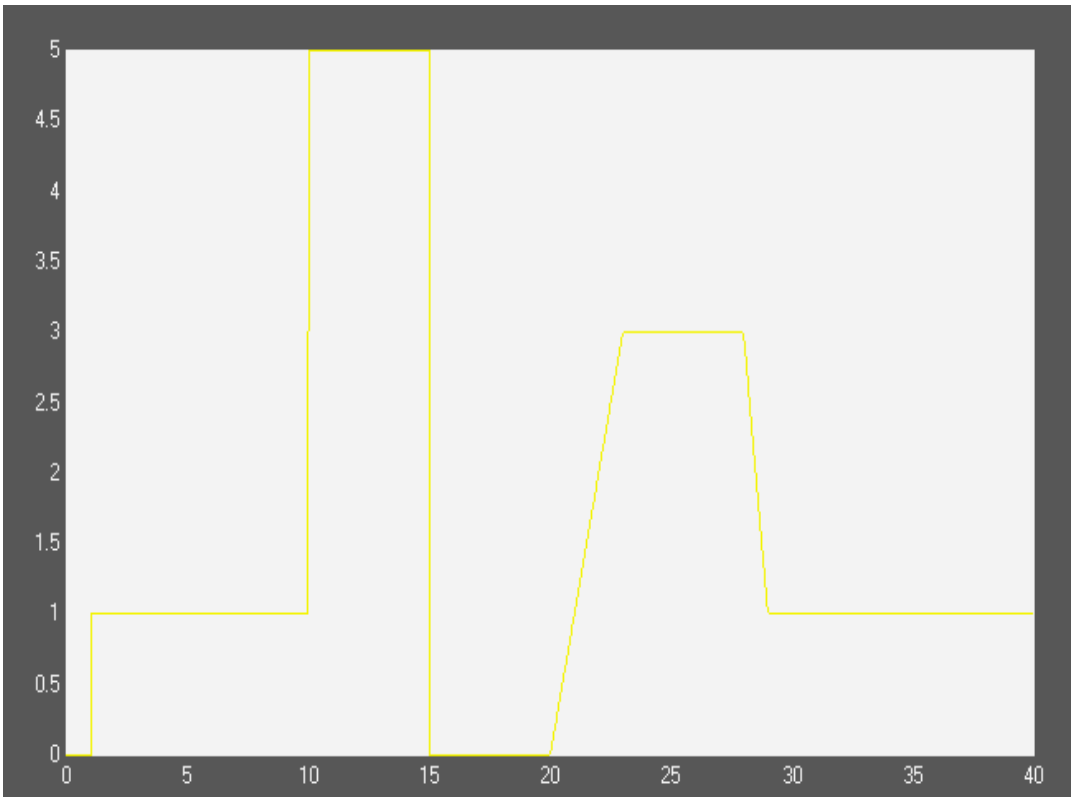
Figura 2.1 Diagrama de bloque del servosistema



R es la señal de referencia para el sistema, los valores de las constante proporcionar, intregal y derivativa del PID son 100, 200, y 4.5 respectivamente.

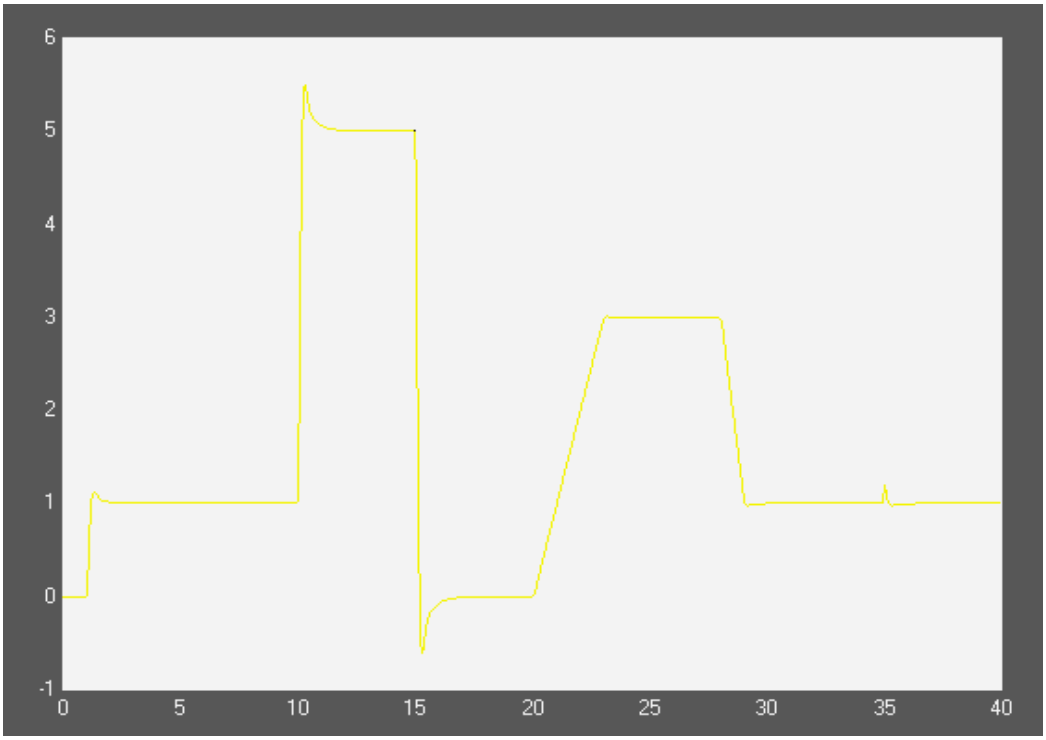
Para hacer la simulación y obtención de datos para el entrenamiento de la red neuronal se realizo en matlab, para obtener un buen entrenamiento de la red neuronal la señal de referencia R tuvo variaciones en el tiempo, la señal de de referencia es la que se muestra en la siguiente figura:

Figura 2.2 Señal del set-point.



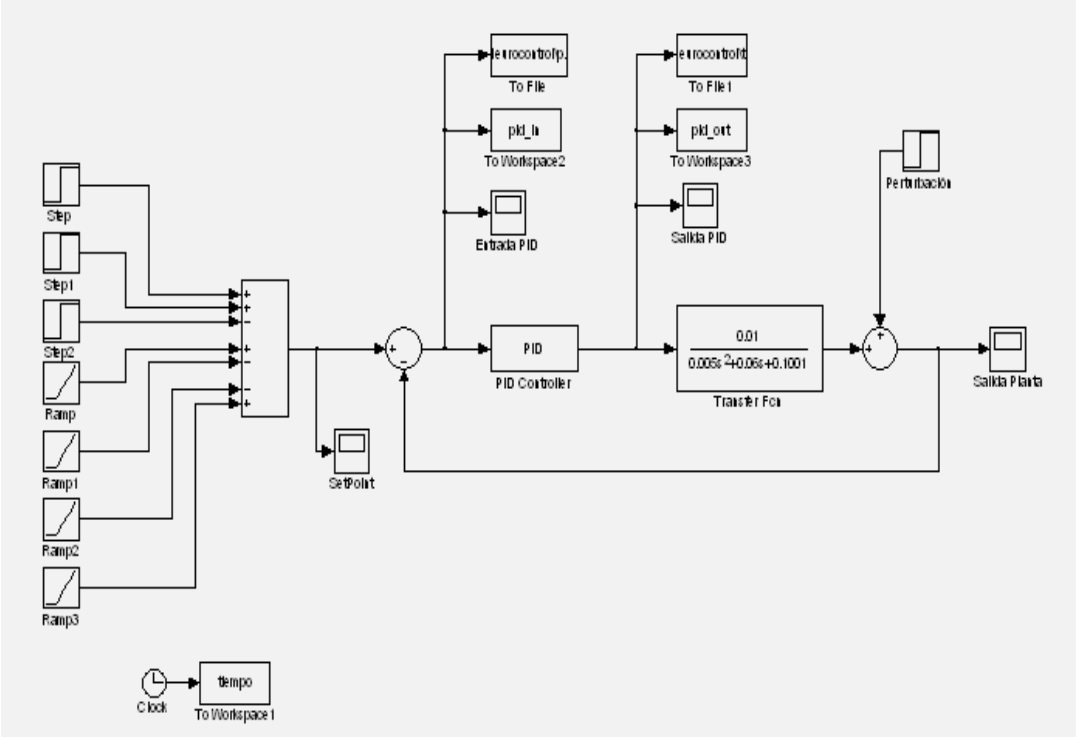
Las señales de referencia son las escalón y rampa, además a los 35 segundos se introduce una perturbación externa al servosistema, dicha señal es una escalón con una amplitud de 0.2, como se puede observar en la figura la señal de transferencia es continua en el tiempo. En la salida del servosistema el comportamiento es el que se muestra en la siguiente figura:

Figura 2.3 Señal de salida del servosistema con el PID



Para hacer la simulación en matlab se utilizo la siguiente configuración:

Figura 2.4. Diagrama de bloque del servosistema utilizado en MATLAB.



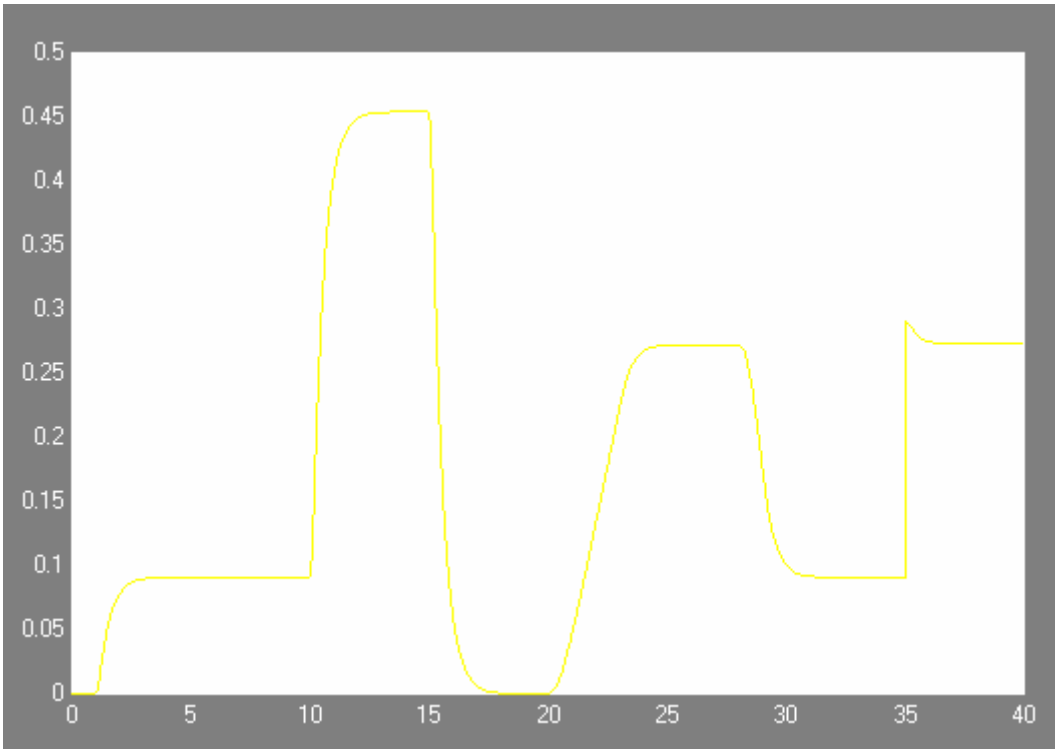
Los datos para el entrenamiento de la RNA son tomados a la entrada y salida del PID, hay que recordar que la red neuronal va a simular el comportamiento del PID.

Antes de empezar con la etapa de entrenamiento hay que analizar como es el comportamiento del servosistema sin el controlador PID, con el fin de mirar si la planta es lo suficientemente estable para trabajar sola, esta labor se hace para mirar si el controlador PID realiza la labor de control o si es necesario que el

controlador PID se encuentre como herramienta de control, porque si la planta logra tener un buen desempeño sin el controlador ya sea este PID o neurocontrolador no justificaria la realización de un neurocontrol para que controlara la velocidad angular del motor DC (servosistema) debido a que el neurocontrolador se estaría subutilizado.

En el simulink de MATLAB se simulo el servosistema sin el controlador PID y se obtuvo la siguiente grafica en la salida del servosistema para evaluar el desempeño del servosistema sin el PID:

Figura 2.5. Salida del servosistema sin el PID



Por medio de esta grafica podemos sacar como conclusión que el servosistema sin el PID no pude realizar un comportamiento adecuado ante las variaciones que se ve sometido, ósea que si el servosistema no tiene asociado un controlador a el

servosistema no puede ser tomado en cuenta para ninguna aplicación a nivel de la ingeniería de control.

2.3 ENTRENAMIENTO DE LA RED NEURONAL

Antes de determinar la arquitectura de la red cabe hacer una breve descripción de las redes usadas en el entrenamiento, la Backpropagetion y Adaline son las que se utilizaron para encontrar el modelo del PID del servosistema. En la siguiente sección se define las redes trabajadas en esta investigación.

2.3.1 Adaline. La red Adaline es similar al Perceptrón, excepto en su función de transferencia, la cual es una función de tipo lineal en lugar de un limitador fuerte como en el caso del Perceptrón. La red Adaline presenta la misma limitación del Perceptrón en cuanto al tipo de problemas que pueden resolver, ambas redes pueden solo resolver problemas linealmente separables, sin embargo el algoritmo LMS es más potente que la regla de aprendizaje del Perceptrón ya que minimiza el error medio cuadrático, la regla sirvió de inspiración para el desarrollo de otros algoritmos, este es el gran aporte de esta red[7].

El término Adaline es una sigla, sin embargo su significado cambió ligeramente a finales de los años sesenta cuando decayó el estudio de las redes neuronales, inicialmente se llamaba ADAptive LInear NEuron (Neurona Lineal Adaptiva), para pasar después a ser Adaptive LInear Element (Elemento Lineal Adaptivo), este

[7]Tutorial del grupo de circuito de la Universidad Politécnica de Madrid-UPM, bajo la dirección del Dr. Diego Andina de la Fuente.
/http://www.gc.ssr.upm.es/inves/neural/ann2/anntutorial.html.

cambio se debió a que la Adaline es un dispositivo que consta de un único elemento de procesamiento, como tal no es técnicamente una red neuronal.

El elemento de procesamiento realiza la suma de los productos de los vectores de entrada y de pesos, y aplica una función de salida para obtener un único valor de salida, el cual debido a su función de transferencia lineal será +1 si la sumatoria es positiva o –1 si la salida de la sumatoria es negativa. En términos generales la salida de la red está dada por

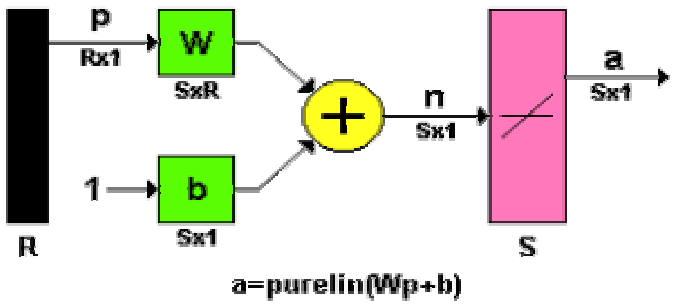
$$a = W^T p \qquad 2.5$$

En este caso, la salida es la función unidad al igual que la función de activación; el uso de la función identidad como función de salida y como función de activación significa que la salida es igual a la activación, que es la misma entrada neta al elemento.

El Adaline es **AD**aptivo en el sentido de que existe un procedimiento bien definido para modificar los pesos con objeto de hacer posible que el dispositivo proporcione el valor de salida correcto para la entrada dada; el significado de correcto para efectos del valor de salida depende de la función de tratamiento de señales que esté siendo llevada a cabo por el dispositivo. El Adaline es **LI**neal porque la salida es una función lineal sencilla de los valores de la entrada. Es una **NE**urona tan solo en el sentido (muy limitado) del PE. También se podría decir que el Adaline es un Elemento Lineal, evitando por completo la definición como **NE**urona.

La estructura general de la red tipo Adaline puede visualizarse en la siguiente figura.

Figura 2.6 Estructura de la red ADALINE.



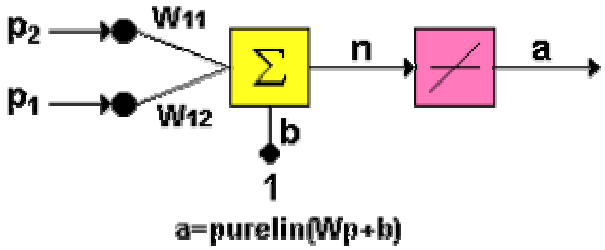
Estructura de una red Adaline

La salida de la red está dada por:

$$a = purelin(Wp + b) = Wb + b \qquad 2.6$$

Para una red Adaline de una sola neurona con dos entradas el diagrama corresponde a la figura 2.7

Figura 2.7 Adaline de una neurona y dos entradas

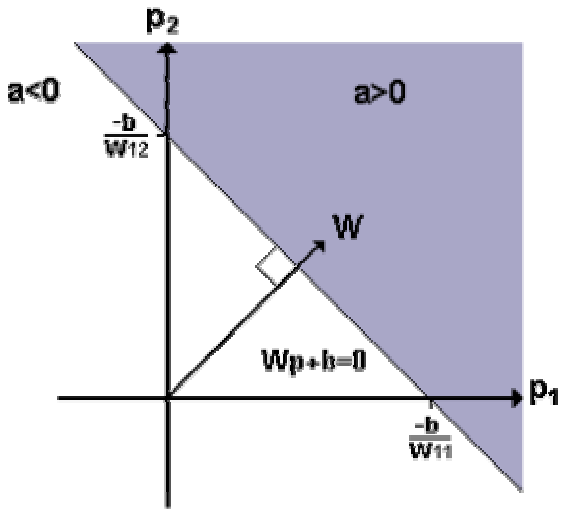


En similitud con el Perceptrón, el límite de la característica de decisión para la red Adaline se presenta cuando n = 0, por lo tanto:

$$W^{t} p + b = 0 \qquad 2.7$$

Específica la línea que separa en dos regiones el espacio de entrada, como se muestra en la figura 2.8

Figura 2.8 Característica de decisión de una red tipo Adaline



La salida de la neurona es mayor que cero en el área gris, en el área blanca la salida es menor que cero. Como se mencionó anteriormente, la red Adaline puede clasificar correctamente patrones linealmente separables en dos categorías.

Al igual que el Perceptrón, la red Adaline es una red de aprendizaje supervisado que necesita conocer de antemano los valores asociados a cada entrada. Los pares de entrada/salida tienen la siguiente forma:

$$\{p_1,t_1\},\{p_2,t_2\},\ldots,\{p_Q,t_Q\} \qquad 2.8$$

Donde P_Q es la entrada a la red y t_Q es su correspondiente salida deseada, cuando una entrada *p* es presentada a la red, la salida de la red es comparada con el valor de *t* que le es asociado.

2.3.2 BACKPROPAGATION. La Backpropagation es un tipo de red de aprendizaje supervisado, que emplea un ciclo propagación – adaptación de dos fases. Una vez que se ha aplicado un patrón a la entrada de la red como estímulo, este se propaga desde la primera capa a través de las capas superiores de la red, hasta generar una salida[8]. 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 de 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 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 de entrenamiento.

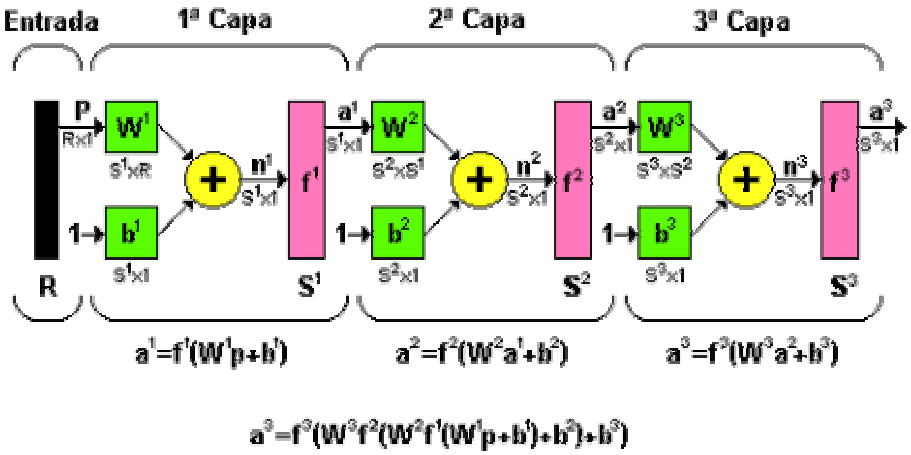
La red Backpropagation hace un procesamiento en paralelo, además el sistema debe ser capas de concentrarse en las características de una entrada arbitraria que

[8] HILERA, José y MARTINEZ, Víctor J. Redes Neuronales Artificiales. Fundamentos, modelos y aplicaciones . Madrid: Ra-ma Editorial. 1995

se asemeje a otros patrones vistos previamente, sin que ninguna señal de ruido lo afecte.

La estructura típica de una red multicapa se observa en la siguiente figura:

Figura 2.8 Estructura de una red Backpropagation.



Como puede observarse cada capa puede tener diferente número de neuronas, e incluso distinta función de transferencia. $\boldsymbol{W^1}$ representa la matriz de pesos para la primera capa, $\boldsymbol{W^2}$ los pesos de la segunda y así similarmente para todas las capas que incluya una red. S representa el número de neuronas y el exponente representa la capa a la cual la neurona corresponde**, b** es la ganancia de cada neurona, **f** es la variable de la función de transferencia de cada neurona, **a** es la señal de salida.

El algoritmo Backpropagation para redes multicapa es una generalización del algoritmo LMS (Least Mean Square), ambos algoritmos realizan su labor de actualización de pesos y ganancias con base en el error medio cuadrático. La red Backpropagation trabaja bajo aprendizaje supervisado y por tanto necesita un set de entrenamiento que le describa cada salida y su valor de salida esperado de la siguiente forma:

$$\{p_1,t_1\},\ \{p_2,t_2\},\ \ldots,\{p_Q,\ t_Q\} \qquad 2.9$$

Donde p_Q es una entrada a la red y t_Q es la correspondiente salida deseada para el patrón q-ésimo. El algoritmo debe ajustar los parámetros de la red para minimizar el error medio cuadrático.

El entrenamiento de una red neuronal multicapa se realiza mediante un proceso de aprendizaje, para realizar este proceso se debe inicialmente tener definida la topología de la red esto es: número de neuronas en la capa de entrada el cual depende del número de componentes del vector de entrada, cantidad de capas ocultas y número de neuronas de cada una de ellas, número de neuronas en la capa de la salida el cual depende del número de componentes del vector de salida o patrones objetivo y funciones de transferencia requeridas en cada capa, con base en la topología escogida se asignan valores iniciales a cada uno de los parámetros que conforma la red.

Es importante recalcar que no existe una técnica para determinar el número de capas ocultas, ni el número de neuronas que debe contener cada una de ellas para un problema especifico, esta elección es determinada por la experiencia del diseñador, el cual debe cumplir con las limitaciones de tipo computacional.

Cada patrón de entrenamiento se propaga a través de la red y sus parámetros para producir una respuesta en la capa de salida, la cual se compara con los patrones objetivo o salidas deseadas para calcular el error en el aprendizaje, este error marca el camino mas adecuado para la actualización de los pesos y ganancias que al final del entrenamiento producirán una respuesta satisfactoria a todos los patrones de entrenamiento, esto se logra minimizando el error medio cuadrático en cada iteración del proceso de aprendizaje.

2.3.3 Programacion de las redes neuronales. Primero se creo en matlab las variables de entrada y salida del PID, *"p"* y *"t"* respectivamente por medio el comando *"To Workspace"*, y para guardar las matrices de entrenamiento se utilizo el comando *"To File"* el cual carga los valores en el archivo *"Neurocontrol\p.mat"* y *"Neurocontrol\t.mat"*, esta parte se realizo en el simulink de matlab.

Los datos de entrada y salida del PID tuvieron que ser normalizados, para evitar el error elevado en el entrenamiento de la red neuronal, un ejemplo de esto fue cuando se entrenaron las cuatros redes Backpropagation dichas redes presentaron un gran error, aun cuando se programo estas redes con un error esperado de 0.1, y un numero de iteraciones de 400[9].

La programación que se utilizo en matlab fue la siguiente:

```
load ('C:\Neurocontrol\p.mat');
load ('C:\Neurocontrol\t.mat');
p=p(2,:);        %Eliminar la fila de tiempo
t=t(2,:);        %Eliminar la fila de tiempo

net=newff(minmax(p),[5 1],{'tansig' 'purelin'},'trainlm');
net.trainParam.epochs=400;
net.trainParam.goal=0.1;
net.trainParam.show=1;
[net,tr] = train(net,p,t);
pause
gensim(net,-1)
```

Las funciones *"Load ('C:\Neurocontrol\p.mat')"* y *"load ('C:\Neurocontrol\t.mat')"* cargan las matrices de entrada y salida del PID.

Las funciones *"p=p(2,:)"* y *"t=t(2,:)",* eliminan la fila de tiempo de la matriz, mientras que la función *"net=newff(minmax(p),[5 1],{'tansig' 'purelin'},'trainlm')"* , crea una red

[9] **MATLAB®**. versión 5.3, Toolbox de redes neuronales usando MATLAB.

neuronal tipo Feed Forward Backpropagation, con 5 neuronas en la capa oculta y una en la capa de salida, *"'trainlm'"* especificándole a la red que el algoritmo de aprendizaje es el Levenberg – Marquard[9].

El algoritmo que se utilizo para el entrenamiento es el Levenberg – Marquard, este algoritmo se caracteriza por ser un algoritmo iterativo de optimización en el que el método de iteraron representa una leve modificación sobre el método tradicional de newton[10], para utilizar este algoritmo en la minimización de una función solo es necesario dar una rutina que calcule la función a minimizar.

La función *"net.trainParam.goal=0.1",* corresponde a la meta del error que se desea que alcance la red durante el entrenamiento.

La función *"net.trainParam.epochs=400"*, especifica el número de iteraciones que debe hacer la red neuronal.

La siguiente función *"gensim(net,-1)"*, crea en simulink el neurocontrolador que a entrenado la red neuronal.

En total fueron entrenadas 5 redes neuronales sin normalizar los datos de entrenamiento, a continuación se presenta las cinco graficas de error vs entrenamiento que se obtuvieron en matlab, en ellas se puede observar el error alcanzado en el entrenamiento:

[10] Tutorial del grupo de circuito de la Universidad Politécnica de Madrid-UPM, bajo la dirección del Dr. Diego Andina de la Fuente.
/http://www.gc.ssr.upm.es/inves/neural/ann2/anntutorial.html.

Figura 2.9 Grafica de entrenamiento de la primera red neuronal.

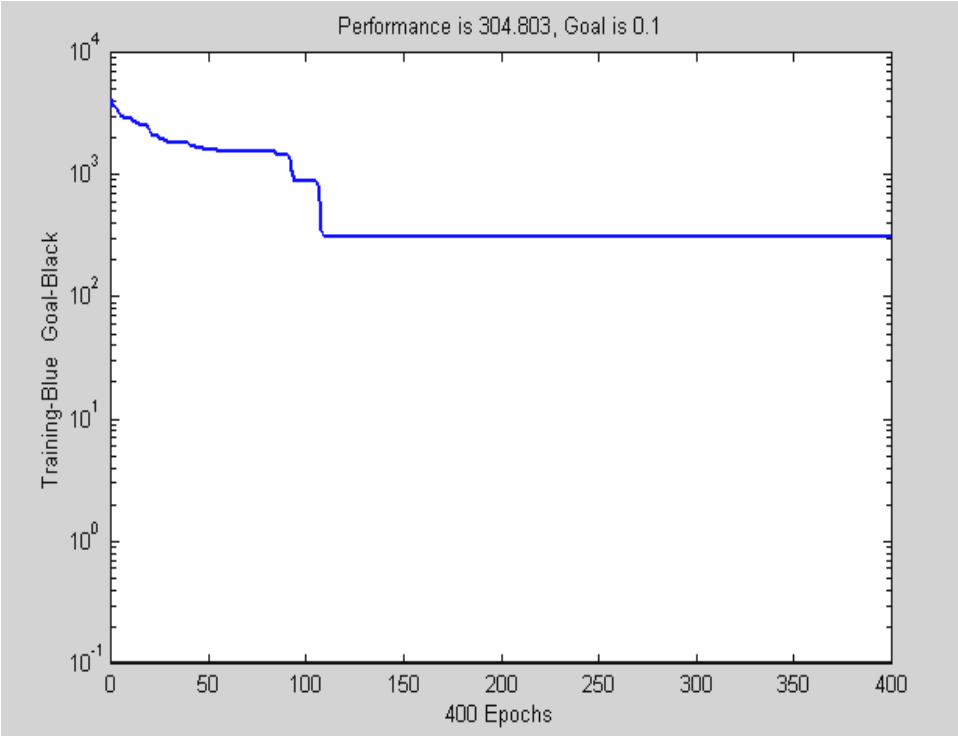


Figura2.10 Grafica de entrenamiento de la segunda red neuronal.

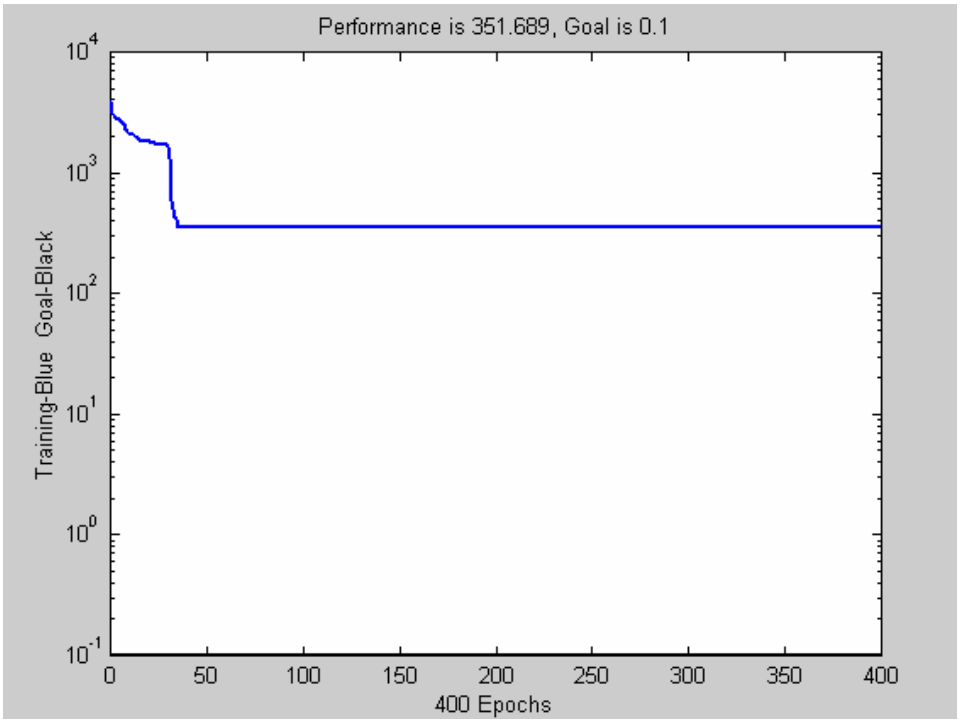


Figura 2.11 grafica de entrenamiento de la tercera red neuronal.

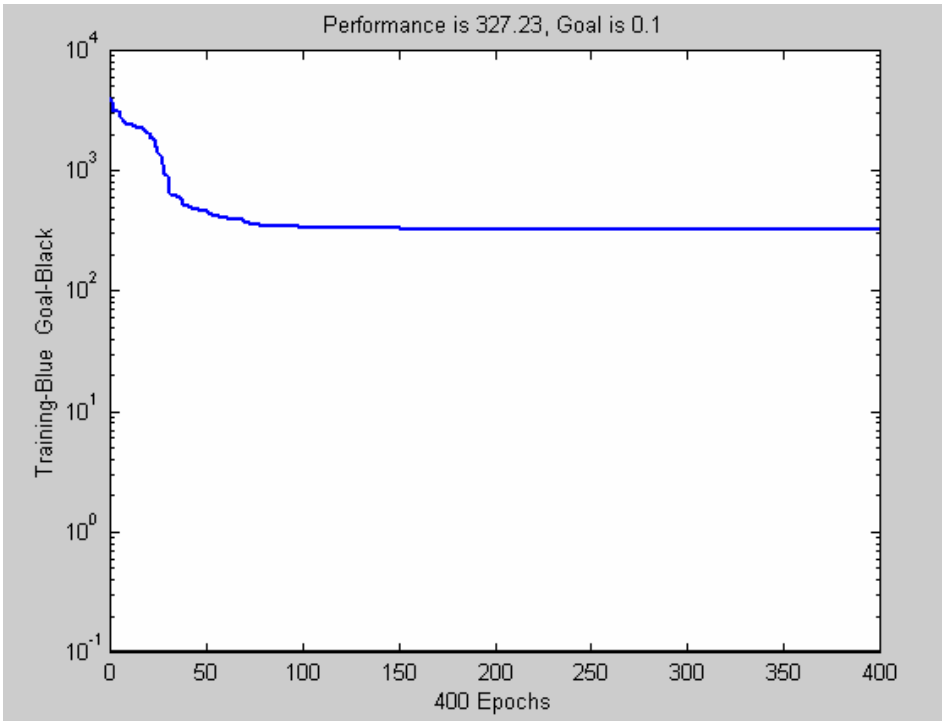
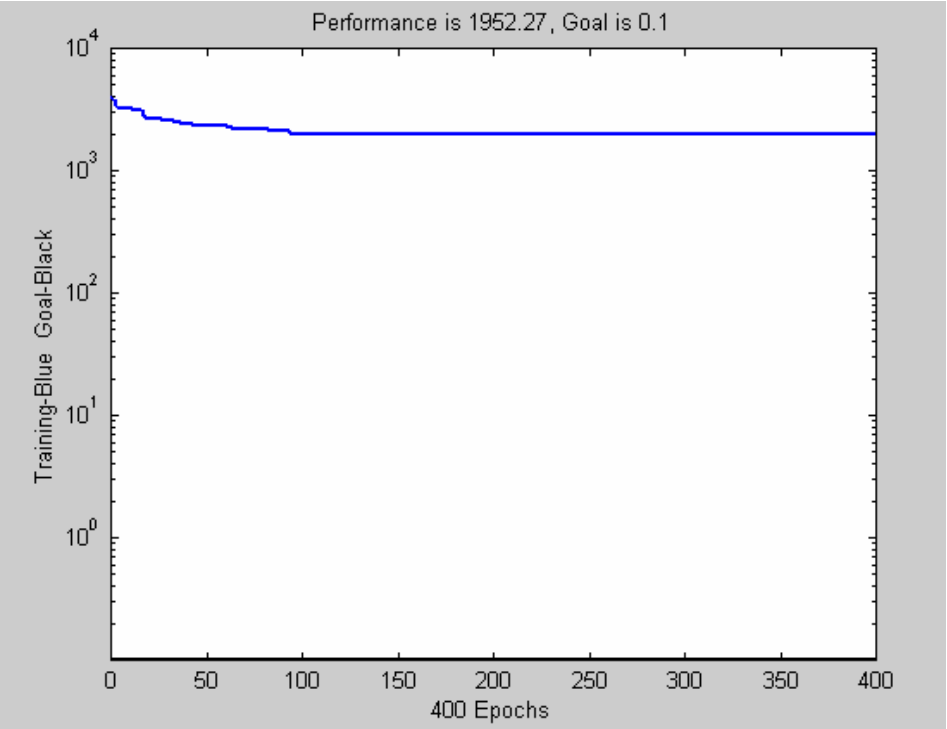


Figura 2.12 grafica de entrenamiento cuarta red neuronal.



Como se puede observar en las anteriores graficas los errores alcanzados fueron de gran magnitud, y la meta del error propuesto no fue alcanzada en el entrenamiento, esto se debe a que los valores usados en el entrenamiento no se normalizaron pero de todas formas no se puede decir que estas redes no pueden ser usadas para modelar el PID, para hacer tal afirmación hay que validar estas redes, la red que se valido fue la que presento el menor error, la primera red, la siguiente tabla hace un resumen de las redes entrenadas y de los resultados obtenido en el entrenamiento. Cabe resaltar, que para la salida de la red neuronal se usó una sola neurona.

Tabla 2.1 Entrenamiento de redes

Red Entrenada	No Neuronas 1 Capa Oculta	No Neuronas 2 Capa Oculta	Error Alcanzado	Gradiente Alcanzado
Primera	5	X	304,80	1,89*10^10
Segunda	6	X	351,69	9368,31*10^10
Tercera	3	X	327,23	8215,85*10^10
Cuarta	6	3	1735,89	9456,14*10^10

Para la normalización de los datos primero se tuvo que identificar la magnitud del valor máximo que se obtuvo de los datos de entrada y salida del PID, este proceso se realizó en matlab con la función "*minmax(abs(p))*", para esto se utilizo la siguiente programación:

```
load ('C:\Neurocontrol\p.mat');
load ('C:\Neurocontrol\t.mat');
p=p(2,:);        %Eliminar la fila de tiempo
t=t(2,:);        %Eliminar la fila de tiempo
minmax(abs(p))
ans =

     0    5.0000
```

Como se puede observar el valor máximo obtenido en la matriz de entrada del PID es 5, para la salida del PID se realiza el mismo procedimiento y el máximo valor obtenido 2700.

Una vez obtenido el máximo valor de la matriz "p" y "t", se divide todo los valores de la matriz por el valor máximo de la misma, con el fin de obtener valores que oscilen entre -1 y 1, de tal forma que cuando se esté entrenando la red neuronal esta logren alcanzar un error menor que uno.

La siguiente programación fue la utilizada en matlab:

```
load ('C:\Neurocontrol\p.mat');
load ('C:\Neurocontrol\t.mat');
```

```
p=p(2,:);        %Eliminar la fila de tiempo
t=t(2,:);        %Eliminar la fila de tiempo
p=p/5;           %Normaliza el valor de entrada
t=t/2700;        %Normaliza el valor de salida
net=newff(minmax(p),[N N],{'tansig' 'purelin'},'trainlm');
net.trainParam.epochs=400;
net.trainParam.goal=0.1;
net.trainParam.show=1;
[net,tr] = train(net,p,t);
pause
gensim(net,-1)
```

2.4 DETERMINACIÓN DE LA TOPOLOGÍA DE LA RED NEURONAL.

Para la determinación de la topología de la red se entrenaron un numero total de 37 redes neuronales de las cuales cinco se entrenaron sin normalizar los datos de entrenamiento, y dos redes entrenadas son de tipo Adaline y las treinta y cinco restante son Backpropagetion, además se hicieron cambios en las distintas variables de entrenamiento como por ejemplo: la meta del error, número de neuronas, número de capas ocultas y número de iteraciones.

Al final se validaron las redes que presentaron un mejor comportamiento en el aprendizaje, o aquellas que alcanzaron el menor error durante el entrenamiento, los datos que se usaron para validar la red fueron los mismos que se usaron para entrenar la red, una ves determinada la red neuronal que se va utilizar para modelar el PID a esta se le introduce nuevos datos para observar como es su comportamiento ante nuevos eventos.

Inicialmente se escogieron topología al azar, primeramente se escogió una sola capa oculta de dos neuronas y un número de iteraciones igual a 100, para empezar la modelación, y con un error esperado de 0.0001. El proceso de que se llevo a cabo para encontrar la topología de la red es el que se muestra en las siguientes secciones.

2.4.1 Variando número de neuronas para una sola capa oculta. Una vez escogida la red neuronal inicial, el número de neurona se comenzó a variar de dos en dos hasta alcanzar el valor 20 neuronas en la capa oculta, para todas las redes entrenadas la capa de salida solamente tiene una sola neurona.

La red se entreno con un error esperado de 0.0001 y un número total de iteraciones igual a 1000. En la siguiente tabla se muestra los resultados obtenidos en la etapa de entrenamiento de las distintas redes entrenadas:

No iteraciones = 100

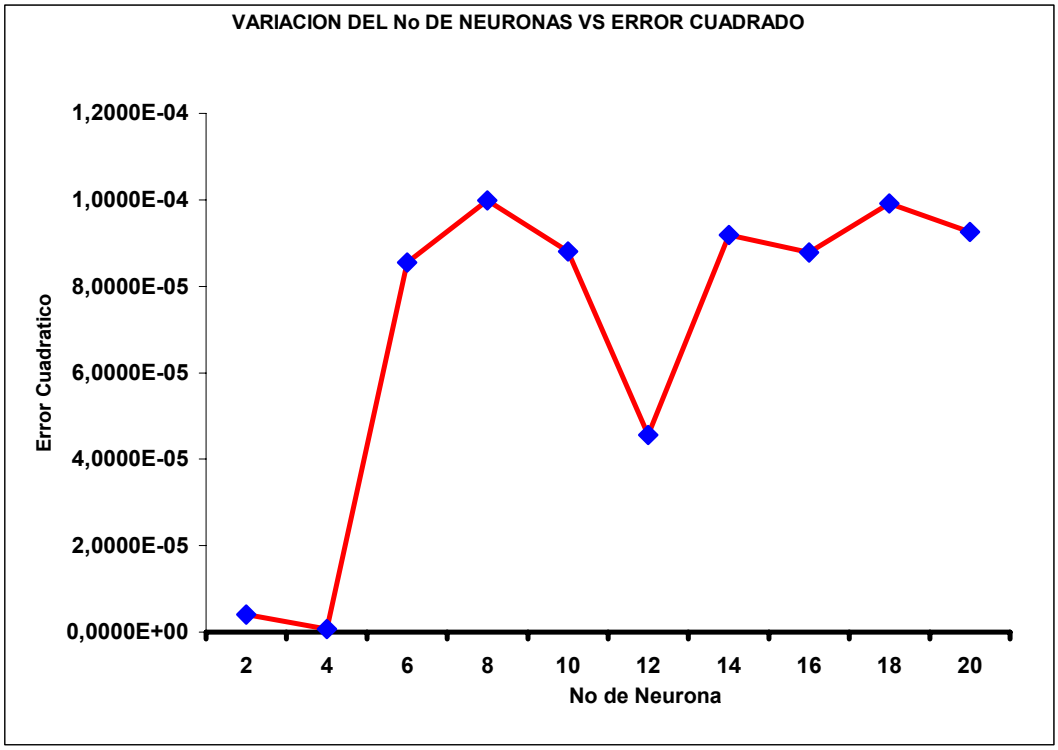
Tabla 2.2 Variación del no de neuronas

VARIACIÓN DEL No DE NEURONAS			
Red Entrenada	**No de Neurona**	**Error Alcanzado**	**No Iteraciones**
1	**2**	**4,0824E-06**	**1**
2	**4**	**7,4041E-07**	**2**
3	**6**	**8,5483E-05**	**17**
4	**8**	**9,9889E-05**	**25**
5	**10**	**8,8040E-05**	**30**
6	**12**	**4,5578 E-05**	**30**
7	**14**	**9,1898E-05**	**52**
8	**16**	**8,7801E-05**	**22**
9	**18**	**9,9170E-05**	**28**
10	**20**	**9,2579E-05**	**5**

A simple vista se logra observar que ninguna de las redes alcanzo el número de iteraciones propuesto, y el error logrado por el entrenamiento de las redes fue menor que 0.0001.

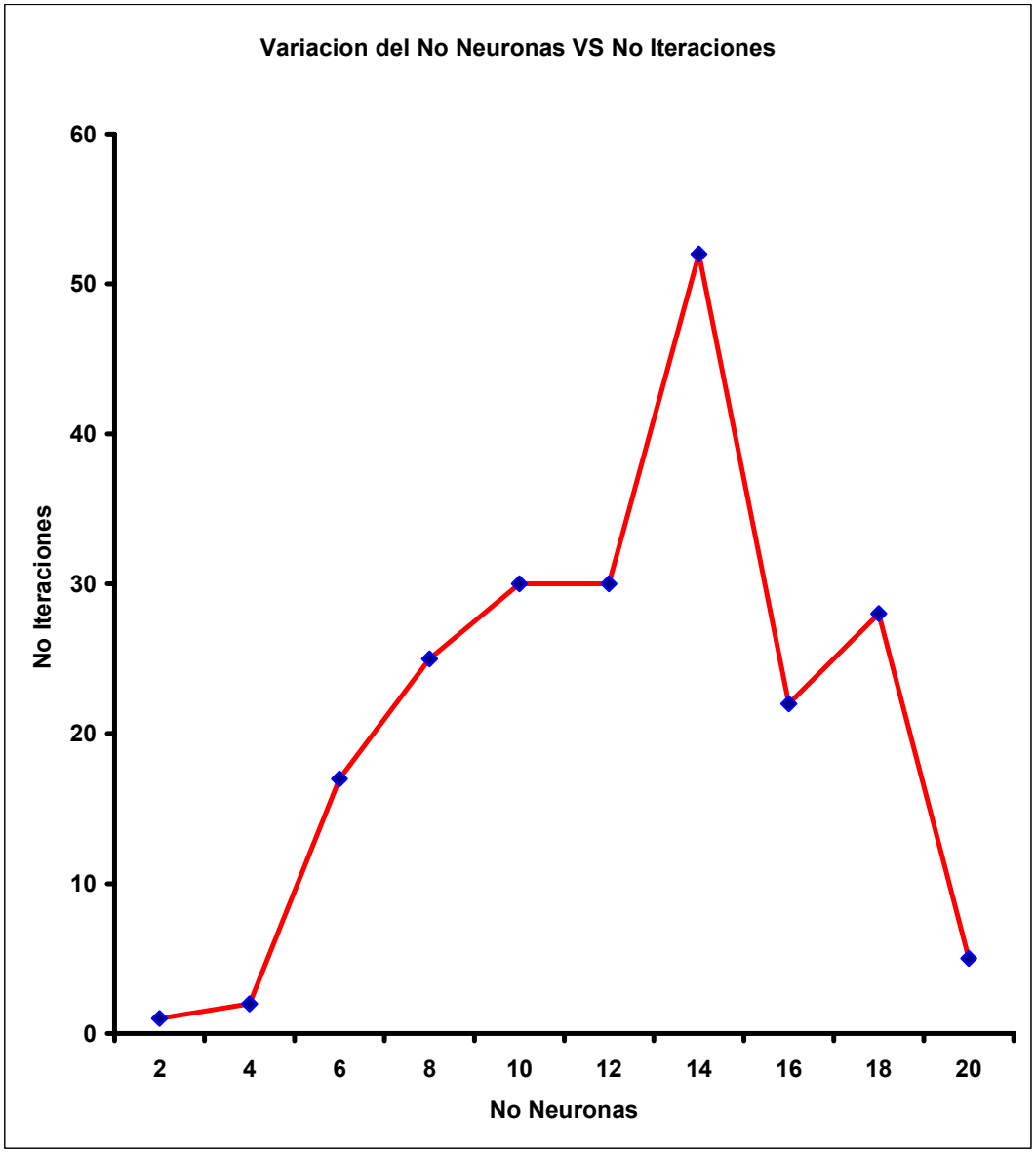
Continuando con el análisis a continuación se presenta una grafica en la cual se aprecia mejor como fue el comportamiento del error cuadrado con respecto a la variación del número de neuronas:

Figura.2.13 VARIACIÓN DEL No DE NEURONAS VS ERROR CUADRADO



El menor error cuadrático medio fue obtenido por la segunda red, la cual tiene cuatro neuronas en la capa oculta, cabe recordar que este error se logro con dos iteraciones, hay que tener cuidado porque la red pudo haber quedado en un mínimo local en la etapa de entrenamiento, en la grafica se puede observar que al aumentar el numero de neuronas el error que lograban en el entrenamiento i aumenta a acepción de la red que tiene dos doce neuronas en la capa oculta.

Figura 2.14 Variación del No Neuronas VS No Iteraciones



En esta grafica se puede notar que las redes que obtuvieron el menor error en el entrenamiento son aquellas que lograron la meta del error propuesto con un mínimo número de iteraciones durante el entrenamiento, las redes fueron la primera y la segunda, casualmente dichas redes son las que tienen el menor numero de neuronas en la capa oculta y esto quizás se debe al poco tiempo empleado por esta

redes para el procesamiento de información recibidas por las redes durante la etapa de entrenamiento.

Ahora, observando la última red que se entreno se puede ver que alcanzo un error de 9,2579E-05 en cinco iteraciones, esto contradice la anterior hipótesis, esto demuestra que para entrenar una red neuronal no existen reglas o pasos específicos para entrenar una red neuronal, esto se logra a manera de prueba y error.

Como resultado final las redes que se van a utilizar para la validación son aquellas que presentaron un menor error; la red 1 y 2, el error alcanzado por estas redes fue 4,0824E-067 y 4041E-07 respectivamente.

2.4.2 Variando el error esperado en el entrenamiento. Para esta parte se trabajo con una sola capa oculta y una neurona en la capa de salida, como en la parte anterior el numero de neurona de la capa oculta se comenzó a modificar de dos en dos hasta alcanzar doce neurona, adicionalmente se trabajo con dos errores esperado para cada numero de neuronas los valores usados son 0.00001 y 0.000001.

Cuando se entreno las primeras redes se observo que al disminuir la meta del error de programación, el error que alcanzaba la red en el entrenamiento aumentaba por tal razón solo se trabajo las primeras cuatro redes con un error esperado de 0.000001.

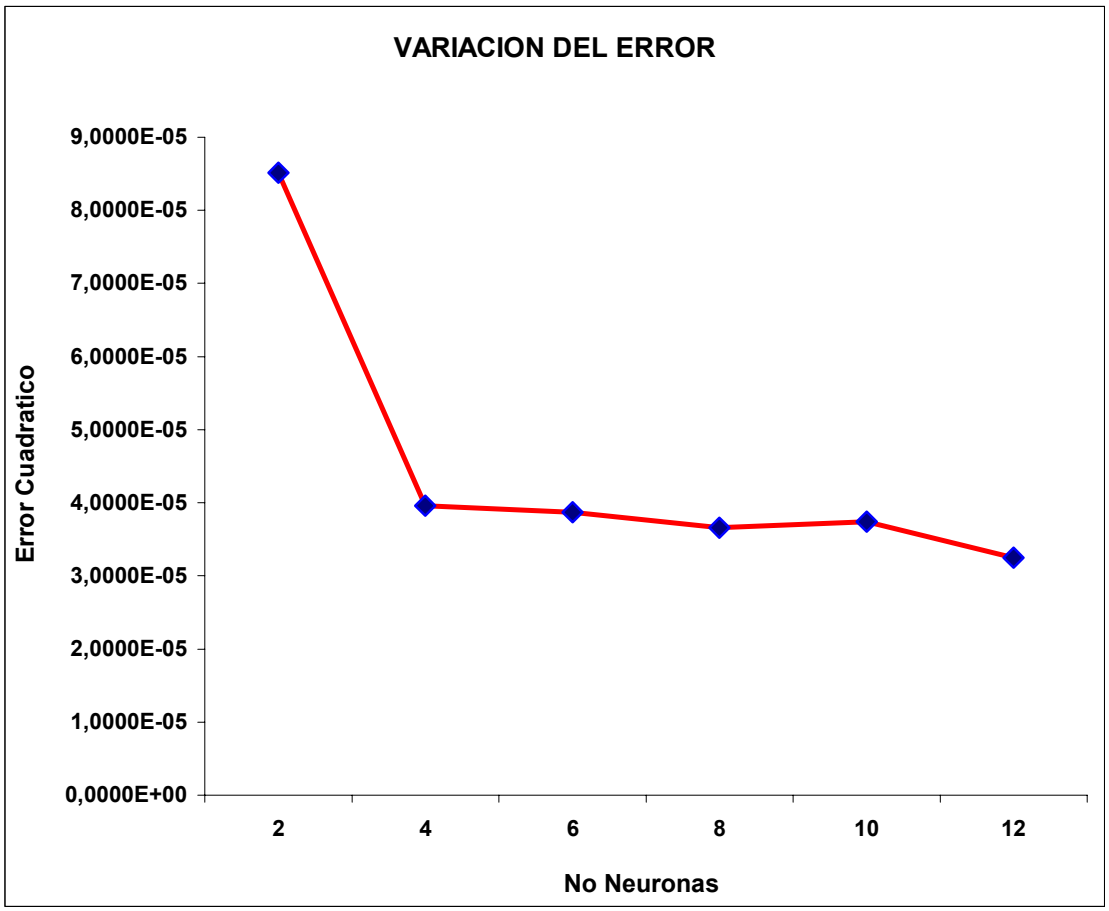
Los resultados obtenidos en el entrenamientote la red se presentan en la siguiente tabla:

Tabla 2.3 Variación Del Error

VARIACIÓN DEL ERROR			
Red Entrenada	**No De Neurona**	**Error Programado**	**Error Alcanzado**
1	2	1,00E-05	8,5136E-05
2	2	1,00E-06	3,2402E-04
3	4	1,00E-05	3,9610E-05
4	4	1,00E-06	2,0145E-04
5	6	1,00E-05	3,8725E-05
6	8	1,00E-05	3,6620E-05
7	10	1,00E-05	3,7379E-05
8	12	1,00E-05	3,2489E-05

La siguiente grafica muestra un mejor enfoque del entrenamiento de la red neuronal, la grafica No 2.15 presenta la variación del error cuadrático medio con respecto al numero de neuronas, en esta grafica se trabajo con un error esperado de 1,00E-05, el error 1,00E-06 no se tuvo en cuenta para la realización de la grafica porque el error que se lograba en el entrenamiento es muy elevados comparado cuando se entrenaron las redes con un error de 1,00E-6, entonces no hay la necesidad de presentar una grafica para hacer un análisis.

Figura 2.15 Variación Del Error



Las redes que mostraron el menor error son la 6, 7 y 8, estas redes tienen 8, 10 y 12 neuronas en la capa oculta respectivamente, para comprobar cual de todas presenta la mejor opción para trabajar como neurocontrolador se selecciono la octava red, esta presenta 12 neuronas en la capa oculta, la validación de esta red se hace mas adelante.

2.4.3 Variando el de número de neuronas para dos capas ocultas. Siguiendo con el proceso de entrenamiento para esta parte se trabajo con dos capas oculta y a esta se la fue cambiando el número de neuronas de cada capa ocultas, el único dato que se mantuvo constante fue el numero de iteraciones el valor seleccionado fue 200, además se hicieron variaciones de la meta error propuesto en la programación, aunque solo para la primera red se entreno con un error de 0.0001 y las demás redes se entrenaron con un error de 0.00001, cuando se disminuyo la meta del error del entrenamiento las redes mostraron un mejor aprendizaje, en la siguiente tabla se muestra los datos alcanzados en el entrenamiento:

No iteraciones = 200

Tabla 2.4 Variación del no de neuronas para dos capas

Variación Del No De Neuronas Para Dos Capas				
Red Entrenada	**No Neurona 1 Capa**	**No Neurona 2 Capa**	**Error Programado**	**Error Alcanzado**
1	4	2	1,00E-04	1,3965E-04
2	6	2	1,00E-05	3,8151E-05
3	8	2	1,00E-05	3,7412E-05
4	8	4	1,00E-05	3,1025E-05
5	6	3	1,00E-05	3,9228E-05
6	10	4	1,00E-05	3,7164E-05

La variación del error no tuvo una relación directa con la modificación del número de neuronas para cada capa oculta, esto dificulto la escogencia de un patrón para determinar una topología de la red, por tal razón la variación del numero de neurona se hizo a la suerte de encontrar una topología que mostrara el menor error posible, como aspecto para resalar los errores alcanzado por las redes neuronales oscilaron dentro un mismo rango, esto demuestra que la adición de otra capa no disminuye o

presenta una mejoría en el entrenamiento para la red neuronal, esto demuestra una vez mas que no existen reglas básicas para entrenar una red neuronal.

Como se alcanza apreciar en la tabla No 2.4 la red que presento el menor error de entrenamiento fue la cuarta red, la cual tiene ocho neuronas en la primera capa oculta y cuatro neuronas en la segunda capa oculta, para la etapa de validación solamente se utilizara esta red para determinar si desempeña un buen control para el servosistema.

2.4.4 Variando el número de iteraciones. Aunque para esta parte solamente se entrenaron 6 redes neuronales, casi todos los parámetros de arquitectura de la red y programación de aprendizaje se le hicieron los suficientes cambios para la realización de un buen análisis, el único valor que se mantuvo constante fue el error propuesto de 0.00001.

Entre los aspectos más importantes que se alcanzaron a observar en el entrenamiento de las redes neuronales se puede destacar que cuando se aumento el numero de iteraciones con el numero de neuronas para una sola capa oculta el error que se obtenía en el entrenamiento disminuye de igual forma, caso contrario ocurre cuando se trabajo con dos capas ocultas, pues para este caso al aumentar el numero de iteraciones el error obtenido en el entrenamiento aumentaba, por tal motivo solo se entrenaron dos redes con dos capas ocultas. En la siguiente tabla se muestra en resumen el trabajo hecho en el entrenamiento:

Error propuesto = 0.00001

Tabla 2.5 Variación Del No Iteraciones

Variación Del No Iteraciones				
Red Entrenada	**No Neurona 1 Cap**	**No Neurona 2 Cap**	**No Iteraciones**	**Error Alcanzado**
1	2	X	200	3,2374E-04
2	4	X	200	3,9610E-05
3	4	2	400	4,6215E-05
4	4	2	600	4,8523E-05
5	6	X	400	3,9610E-05
6	8	X	400	3,8365E-05

La red que mostró el mayor error es la primera, esta red tiene dos neuronas en la capa oculta pero esta misma red ya se había entrenado anteriormente con un número de iteraciones igual a 100 el error que alcanzó en el entrenamiento fue de 4,08238E-06.

En la tabla No 2.5 se alcanza a mirar la red que presenta el menor error la red 6, esta red tiene una sola capa oculta con ocho neuronas, como esta presento el menor error de las redes entrenadas esta se escogió para ser validada.

2.4.5 Entrenamiento de la red Adaline. Basados en la información recopilada de las distintas aplicaciones del neurocontrol en la ingeniería del control, se decidió entrenar una red Adaline con una sola neurona, pero para obtener un buen entrenamiento de le aplico unos retardos a los datos que recibe la red neuronal durante el entrenamiento, esto se realizó con el fin de que la red pueda mantener una información de los datos que recibe de ciclos anteriores este proceso es parecido al utilizados en los modelos neuronales predictivos.

Primero se entreno la red sin normalizar los datos de entrenamiento, después se normalizó los datos de entrenamiento y se entreno nuevamente esta red para así poder hacer una comparación entre estos dos tipos de entrenamiento.

Las funciones utilizadas en matlab fue la siguiente:

```
load ('C:\Neurocontrol\p.mat');
load ('C:\Neurocontrol\t.mat');
p1=p(2,:);      %Eliminar la fila de tiempo
q=size(p1);
q=q(2);
p=zeros(4,q);
p(1,1:q)=p1(1,1:q);
p(2,2:q)=p1(1,1:(q-1));
p(3,3:q)=p1(1,1:(q-2));
p(4,4:q)=p1(1,1:(q-3));
t=t(2,:);         %Eliminar la fila de tiempo
net=newlind(p,t);
pause
gensim(net,-1)
```

La variable “q“ almacena los valores de entrada “p“, la función “p(1,1:q)=p1(1,1:q) “ especifica los retardos que se van hacer a la señal de entrada p.

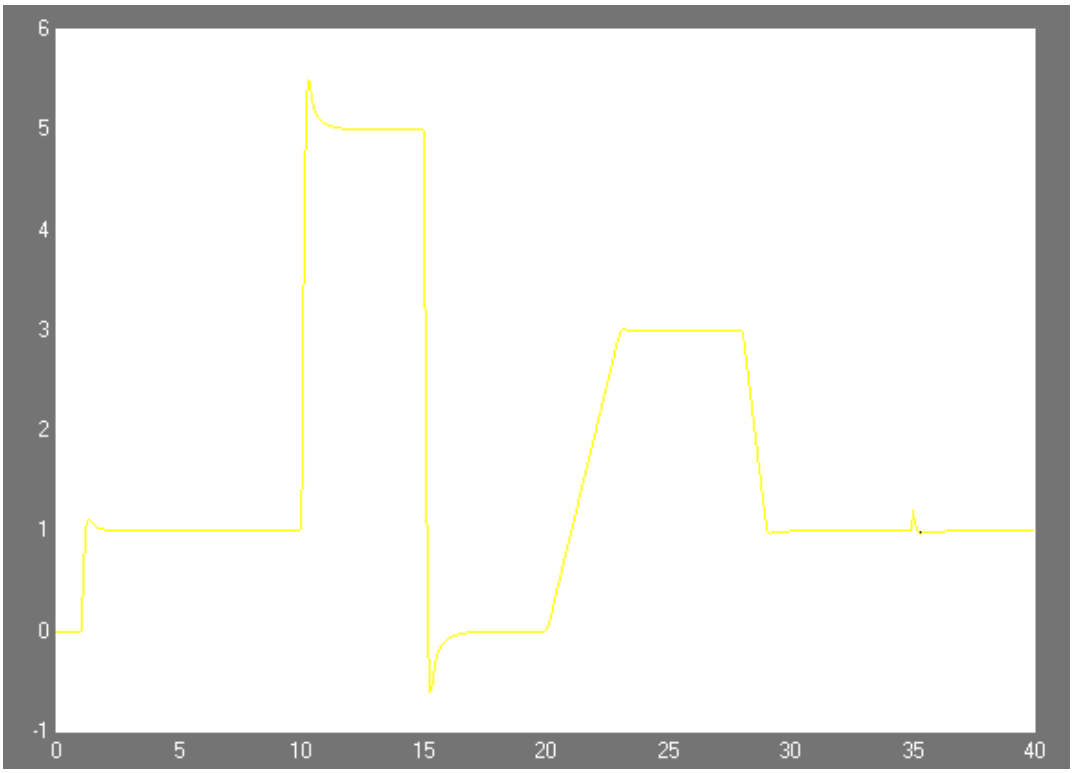
La función “net=newlind(p,t) “; crea una red neuronal tipo ADALINE, pero este comando permite crear dicha red sin la necesidad de especificarles el numero de neurona, rata de aprendizaje, los valores máximos y mínimo de los valores de la entrada, etc. Ya que esta función diseña por si sola todos estos parámetros.

El entrenamiento de esta red fue muy rápido, y los resultados se muestran en la etapa de validación. Para la normalización de los datos de entrad del PID se realizo el mismo procedimiento que el hecho cuando se entrenaron las anteriores redes.

2.5 COMPROBACION.

Para determinar cual va ser la red neuronal que va modelar el PID, se tiene que analizar como es el comportamiento del servosistema cuando trabaja con el neurocontrolador, la mejor forma de evaluar el funcionamiento del neurocontrolador consiste en aplicar los mismo valores usados para entrenar la red neuronal con el fin de comparar como es la respuesta del servosistema trabajando con el PID y el neurocontrolador, pero la señal que se va a comparar no es la presentada en la salida del PID o neurocontrolador si no en la señal de salida del servosistema debido a que es mas fácil de interpretar la señal de salida del servosistema que la mostrada por el PID. Como primer aspecto se muestra en la siguiente grafica la salida del servosistema trabajando con el PID:

Figura 2.16 Salida del servosistema con el PID



Las redes que van a ser comprobadas son aquellas que presentaron el menor error alcanzado en su proceso de entrenamiento, en la siguiente tabla presenta las principales características de las redes que van a ser comprobadas.

Tabla 2.6 VALIDACIÓN

COMPROBACION					
Parámetro Variado	**No Neurona 1 Cap**	**No Neurona 2 Cap**	**No Iteraciones**	**Error Programado**	**Error Alcanzado**
Datos no Normalizados	5	X	400	0,1	300,8
No neuronas	2	X	100	1,00E-04	4,082E-06
No neuronas	4	X	100	1,00E-04	7,404E-07
Error programado	12	X	100	1,00E-05	3,249E-05
No neuronas	8	4	200	1,00E-05	3,103E-05
No iteraciones	8	X	400	1,00E-05	3,836E-05

Adicionalmente a las redes tipo Backpropagation hay que validar una red tipo Adaline esta red consta de una sola neurona, pero hay que hacer la comprobacion para dos tipo de rango los normalizados y los no normalizado.

Antes de empezar con la comprovacion las redes neuronales con las redes que fueron entrenada con datos normalizados, hay que aplica el proceso contrario que se hizo para normalizar la red neuronal, ósea que la entrada del neurocontrolador se dividiría entre 5 y la salida del neurocontrolador se multiplicara por 2700, el diagrama de bloque quedaría de la siguiente forma para el neurocontrol:

Figura 2.17 Diagrama de bloque del neurocontrol

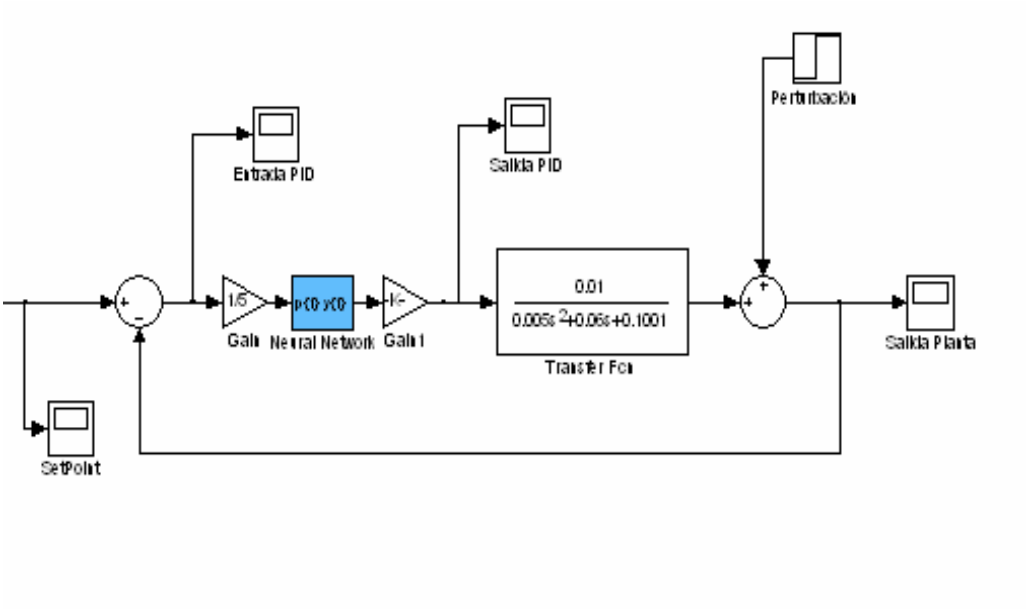
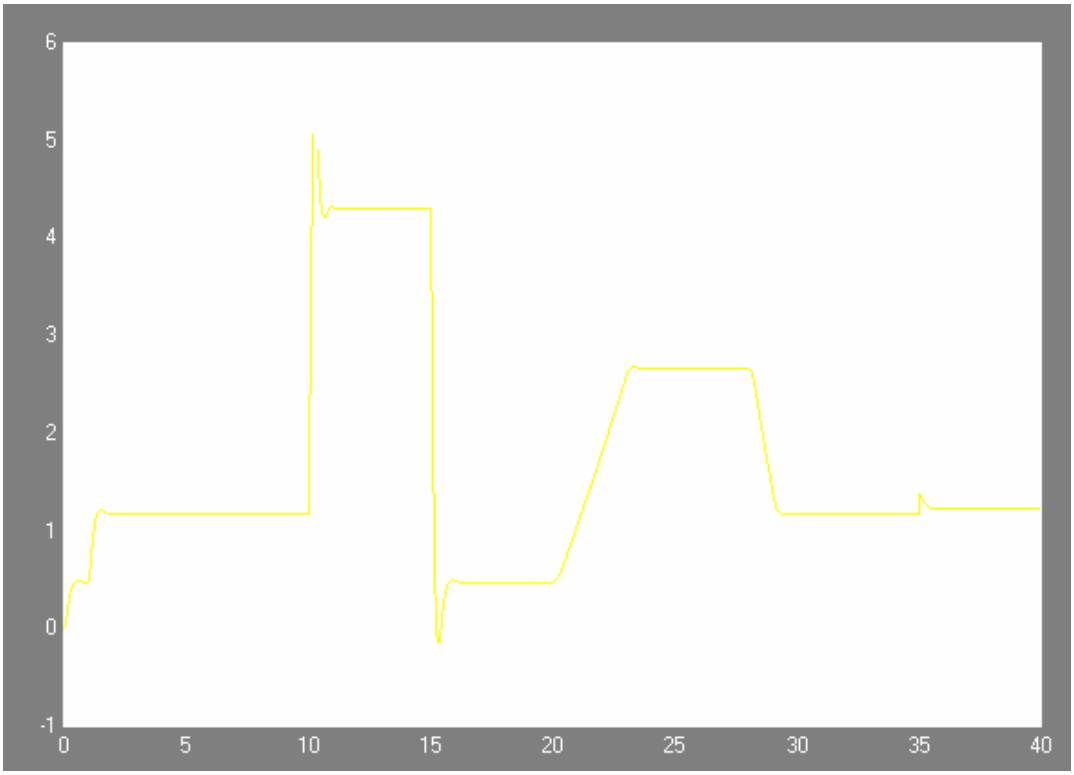


Figura 2.18 Salida servosistema, datos sin normalizar, error propuesto 0.1, No neurona 5 capa oculta.



Observando la graficas se puede ver a simple vista que la señal de salida del neurocontrolador describe el mismo comportamiento, que la presentada por el PID, además las variaciones a la que es sometida el servosistema por la señal de referencia son bien asimiladas por el neurocontrolador en el mismo instante de tiempo en que cambia la señal de entrada, lo cual demuestra que hace un buen seguimiento en el tiempo y una respuesta rápida.

Lo insólito para este neurocontrolador es que el error que se obtuvo la red neuronal al ser entrenada fue muy elevado y uno esperaría que al reemplazarse el neurocontrolador por el PID el comportamiento del servosistema no fuera el adecuado, hay que recordar que el error que se obtuvo en el entrenamiento fue de 300,8 dicho error se debe principalmente a que los datos de entrenamiento no se normalizaron.

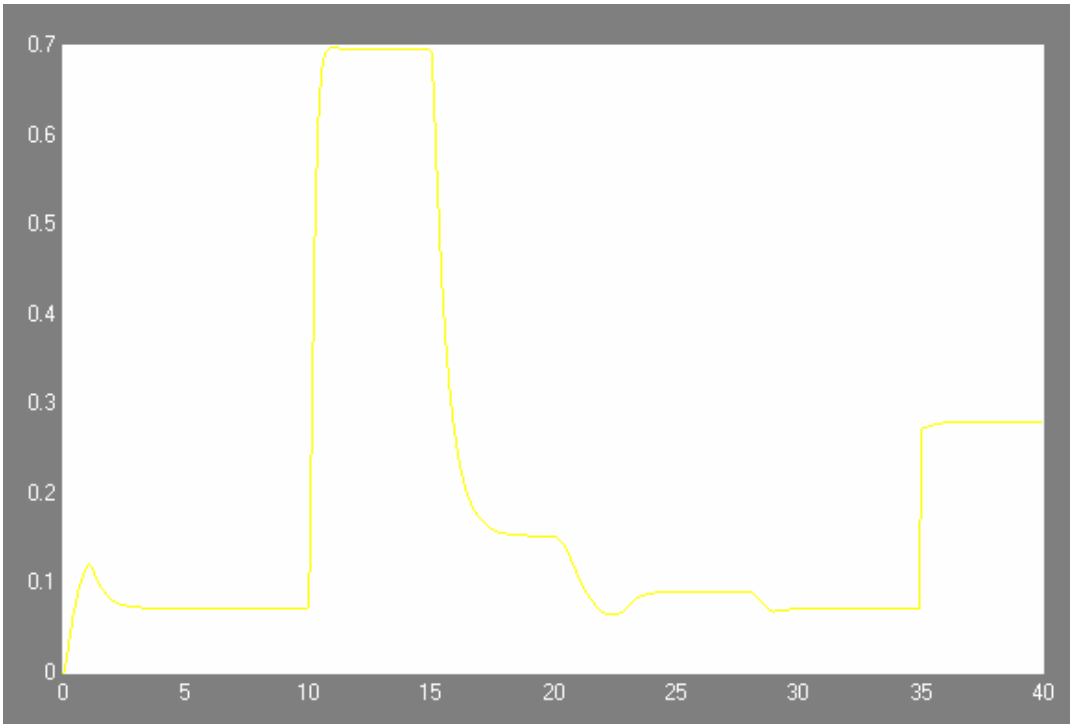
Quizás estemos al frente de una anomalía de las redes neuronales porque se supone que si una red neuronal para modelar o simular un sistema el error que se logra en el aprendizaje debe ser lo mas cercano a cero entonces surge la incertidumbre de que si tal afirmación es del todo cierta, ahora hay que observar como es el comportamiento de las siguientes redes para poder hacer una comparación de esta red con las redes que se entrenaron con datos validados.

Nosotros creemos que como esta red se entreno con datos reales y no con datos que de alguna forma fueron afectados por un factor, al sustituir el PID el control que ejercería el neurocontrolador sobre el servosistema mostraría un comportamiento parecido o mejorado al mostrado por el PID, a pesar de que el error presentado en el entrenamiento fuera elevado.

Otro aspecto a destacar es el comportamiento de la señal del neurocontrolador presenta siempre una desviación o un error en estado estacionario, lo cual se asemeja a un controlador proporcionar, además los sobreimpulsos que muestra el

servosistema cuando trabaja con el neurocontrolador son de mayor amplitud que los mostrado por el PID.

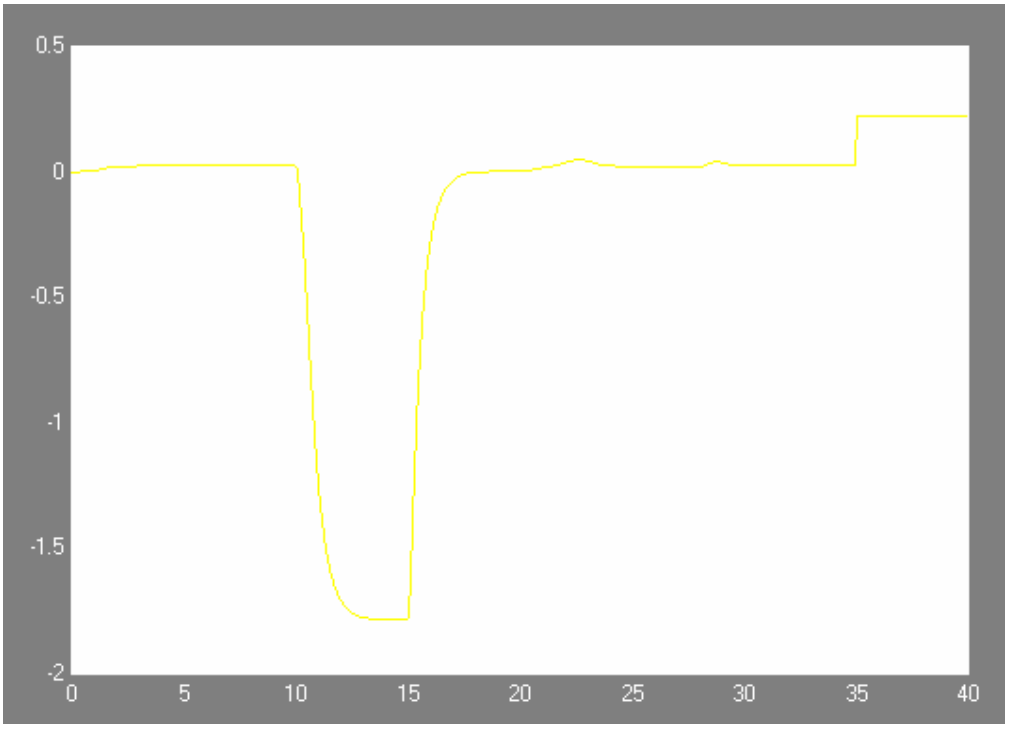
Figura 2.19 Salida del servosistema, No neuronas capa oculta 2, error propuesto 0.0001



Esta red fue una de las que presento uno de los mas bajos errores de entrenamiento, pero como se puede observar en la anterior figura la salida del servosistema con la red neuronal no muestra un buen comportamiento, primero los valores de la amplitud de la grafica es menor que uno y la máxima que se obtiene con el PID es cinco, como segundo aspecto cuando se coloco la perturbación externa al neurocontrolador la respuesta de el fue un desastre.

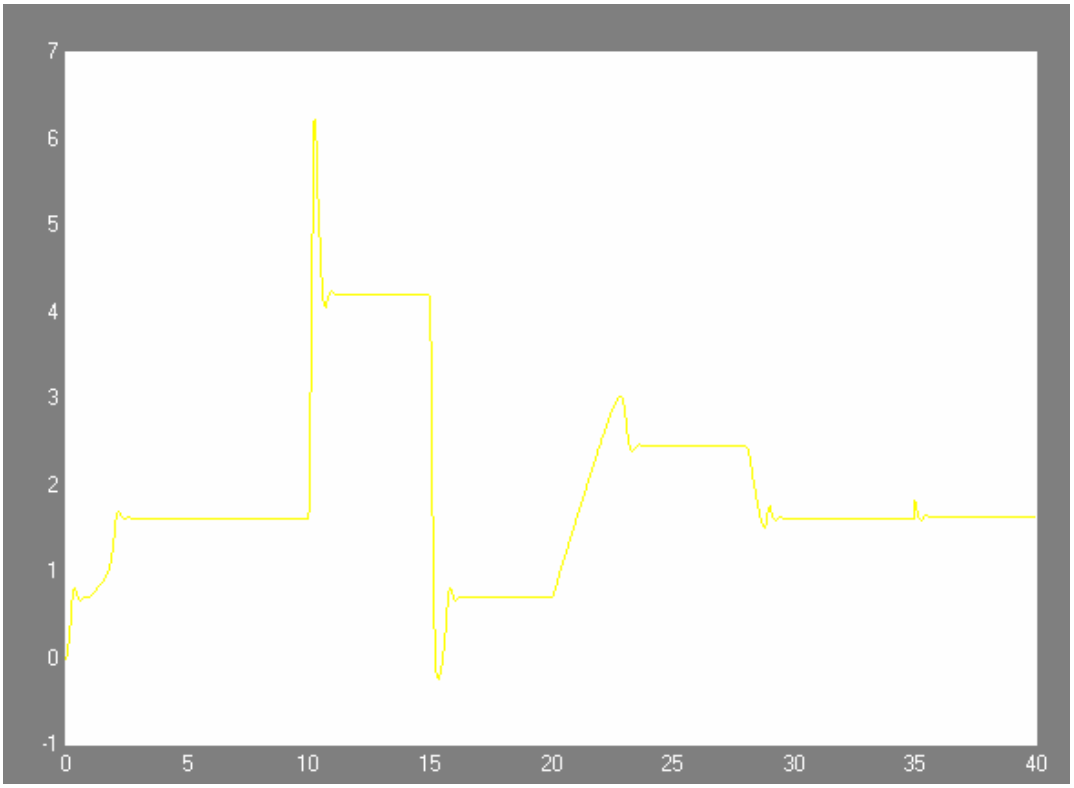
Podemos sacar como conclusión que a pesar de que el error logrado en el entrenamiento por esta red fue unos de los mas bajos presentado por las 35 redes backpropagation entrenadas, no se puede garantizar que la red que muestre el menor error de entrenamiento es la que mejor va a realizar la simulación del PID, quizás el error que encontró esta red fue un mínimo local.

Figura 2.20 Salida del servosistema, No neuronas capa oculta 4, error propuesto 0.0001



La red que presento el menor error de entrenamiento es esta, pero al sustituir el PID por la red neuronal, la respuesta del servosistema es una locura, como en la anterior red quizás el error que se logro en el entrenamiento fue un mínimo local.

Figura 2.21 Salida servosistema, No neuronas capa oculta 12, error propuesto 0.00001



Para esta red se puede observar un mejor comportamiento de la salida del servosistema cuando es controlada por el neurocontrolador, pero se puede observar que cuando la señal de referencia tiene cambio en su comportamiento o tipo de señal, el sobreimpulso que se muestra a causa del cambio de señal del set-point es de gran amplitud.

Lo que hay que resaltar para este neurocontrolador es el buen comportamiento que tiene ante perturbaciones externas, como la que se muestra en la grafica, la velocidad de respuesta es rápida y se estabiliza en un lapso de tiempo corto.

La salida del servosistema presenta un error estacionario de 0.5 el cual limita las aplicaciones de este neurocontrolador.

Figura 2.22 Salida servosistema, No neuronas 1 Capa 8, 2 Capa 4, error propuesto 0.00001

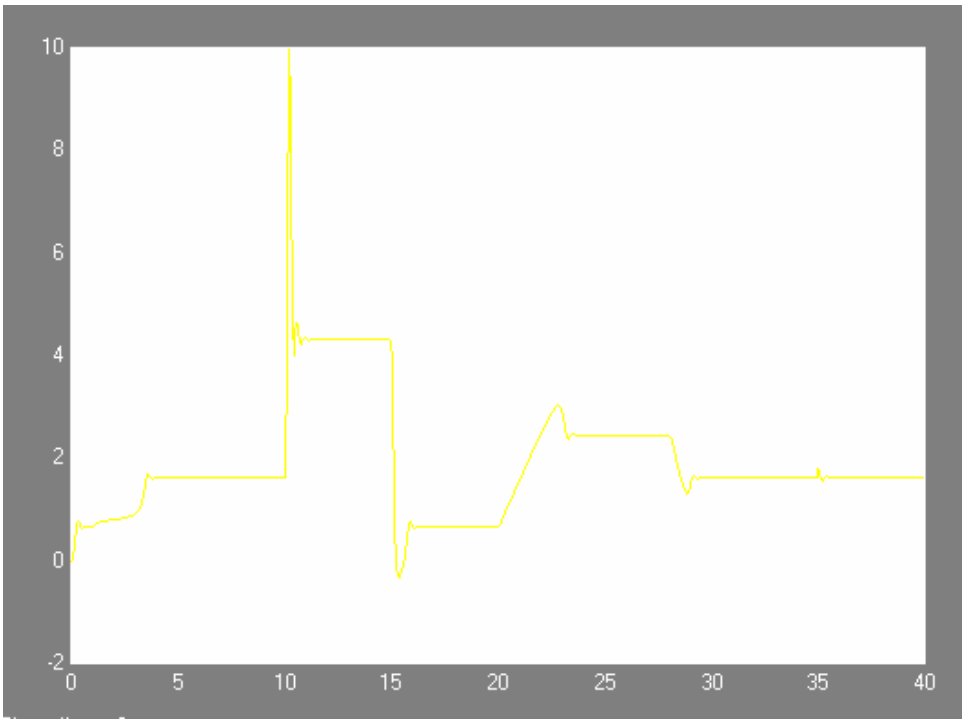
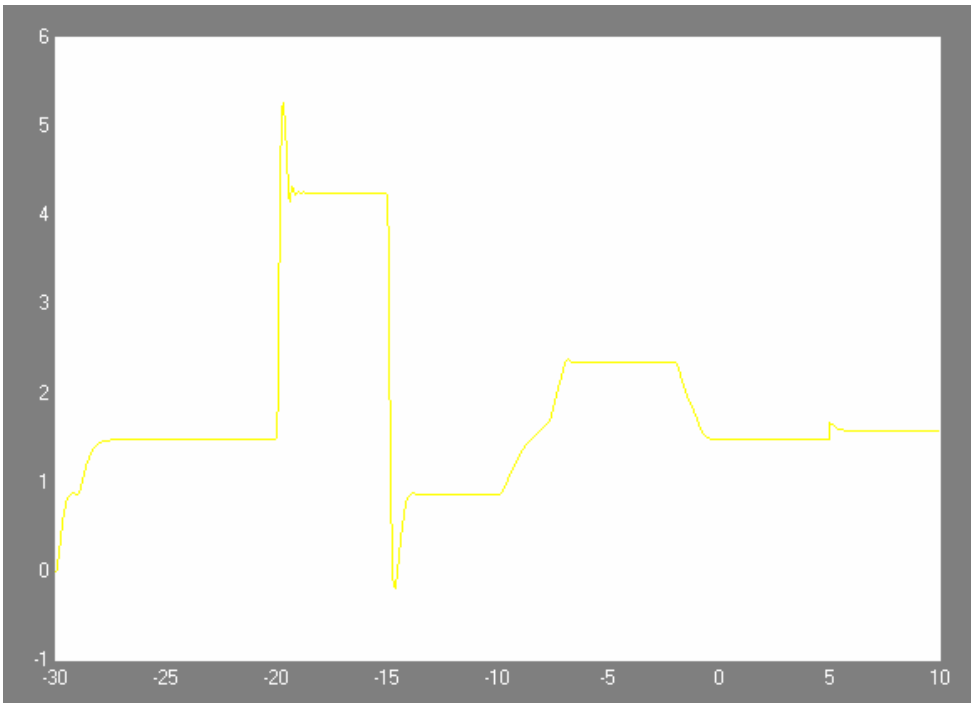


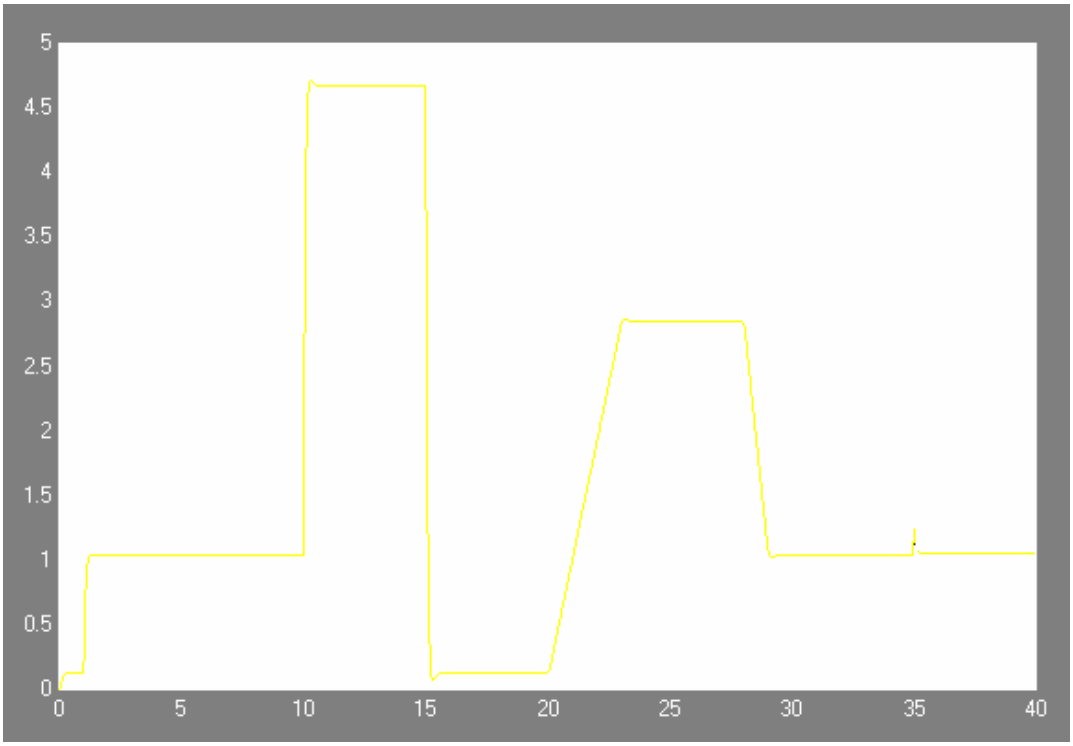
Figura 2.23 Salida servosistema, No neuronas 8, No iteraciones 400



Para las dos graficas mostradas se puede descartar estas redes para la modelación del PID, debido a la amplitud elevada de los sobreimpulso, además el seguimiento de la señal de referencia es probé, aunque ambas dieron un buen comportamiento ante perturbaciones externas.

Para la red Adaline

Figura 2.24 Salida del servosistema, sin normalizar los datos de entrenamiento.

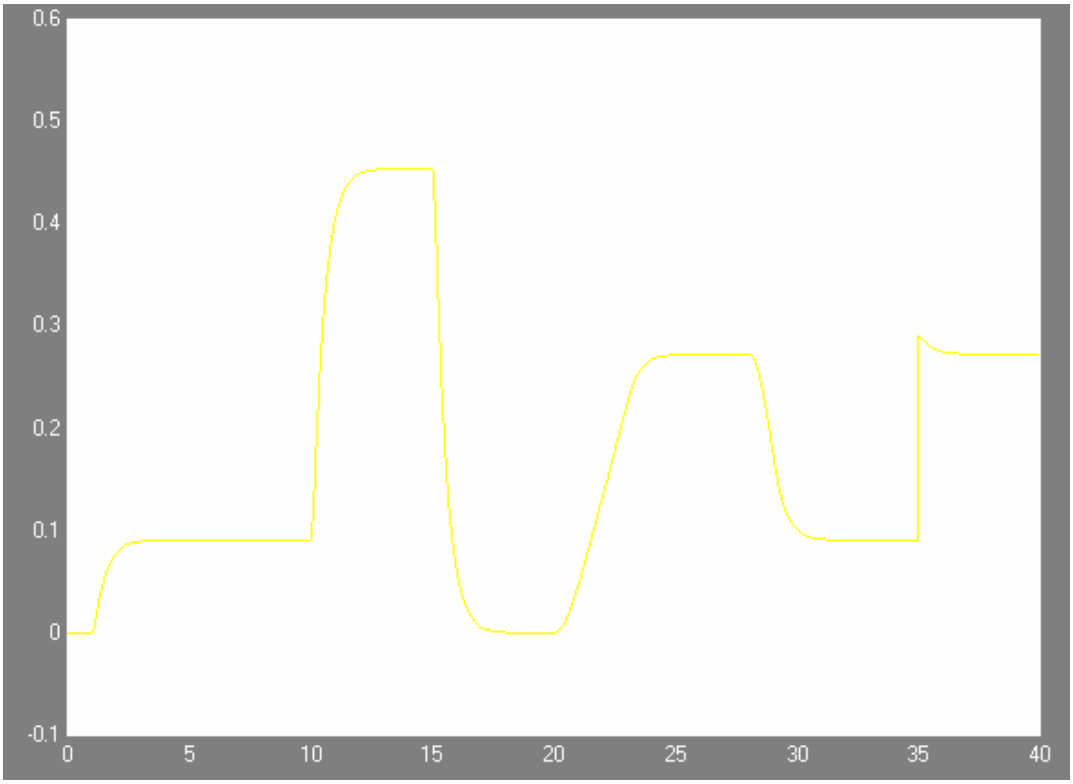


A simple vista se puede observar un mejor comportamiento del servosistema trabajando con este neurocontrolador, entre los aspectos mas importantes podemos citar que el valor del sobre impulso en la señal de salida es prácticamente nulo y comparado con los mostrados por el PID tiene un mejor comportamiento, también el tiempo de respuesta del neurocontrolador es superior que el mostrado por el PID dando una respuesta mas rápidas ante las variaciones en que se ve sometida el servosistema , y con respecto a la perturbación externa se observo que el

neurocontrolador tuvo un mejor comportamiento ya que el tiempo de estabilización fue mas rápido que el PID.

Como aspecto negativo se pudo observar que la señal de salida presenta un error estacionario, este valor oscila entre 0.04 y 0.3, pero en algunas aplicaciones del control automático este error puede aproximarse a cero.

Figura 2.25 Salida del servosistema, Red Adaline, con datos normalizados para el entrenamiento.



A simple vista se puede ver el comportamiento desastroso que presenta esta red cuando es usada para controlar el servosistema, en resumen la red no esta realizando ningún tipo de control deseable.

2.6 OBSERVACIONES DE LA ETAPA DE COMPROBACION

Entre los aspectos mas sobresalientes que se obtuvieron en la etapa de validación de las distintas redes que se escogieron para modelar el PID, podemos citar:

- ✓ Las redes que se entrenaron con datos no normalizados presentaron un mejor comportamiento, que las redes que se entrenaron con datos normalizados.

- ✓ Para las redes que lograron el menor error en la etapa de entrenamiento, como dato contradictorio fueron las que tuvieron el peor comportamiento cuando se utilizaron en el servosistema.

- ✓ Todas las redes presentaron un error estacionario en la salida del servosistema.

- ✓ La red que se escogió para reemplazar el PID, presento un mejor control que el realizado por el mismo PID, dicha red es la Adaline esta red se entreno con datos no normalizados.

2.7 VALIDACIÓN DEL NEUROCONTROLADOR ELEGIDO

La red que se escogió para reemplazar el PID, fue la red Adaline que se entreno con datos no normalizados, la siguiente grafica muestra la nueva señal de referencia que se introduce al servosistema:

Figura 2.26 Señal de referencia del servosistema

.

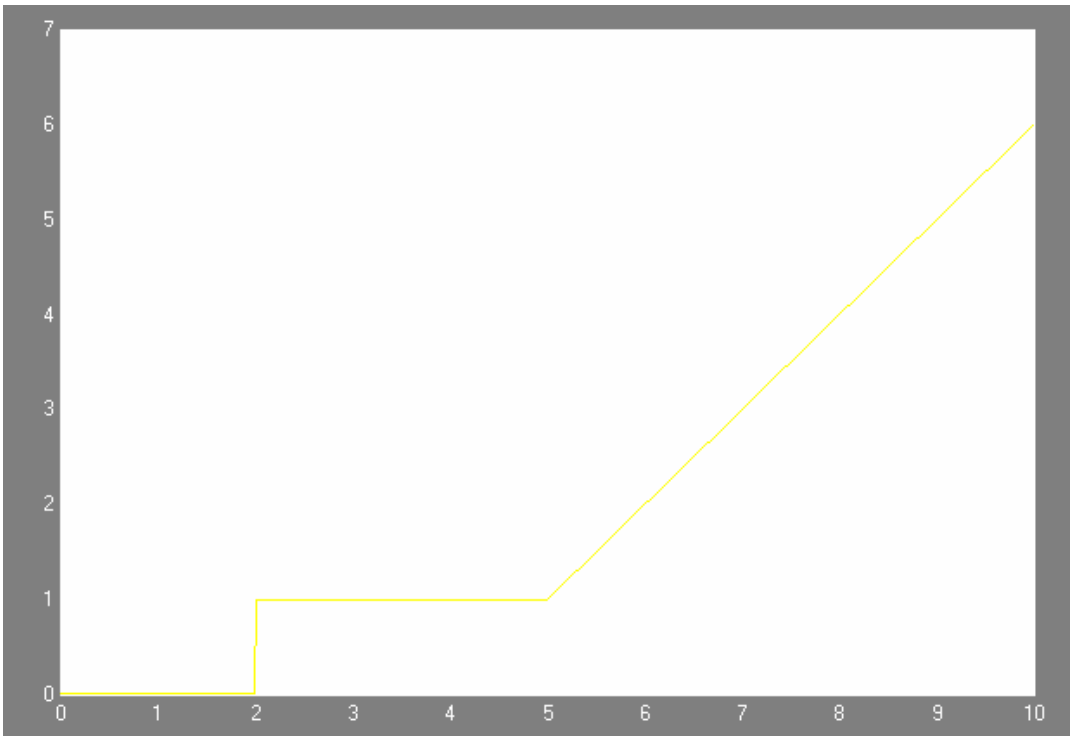


Figura 2.27 Salida del servosistema con el Neurocontrol

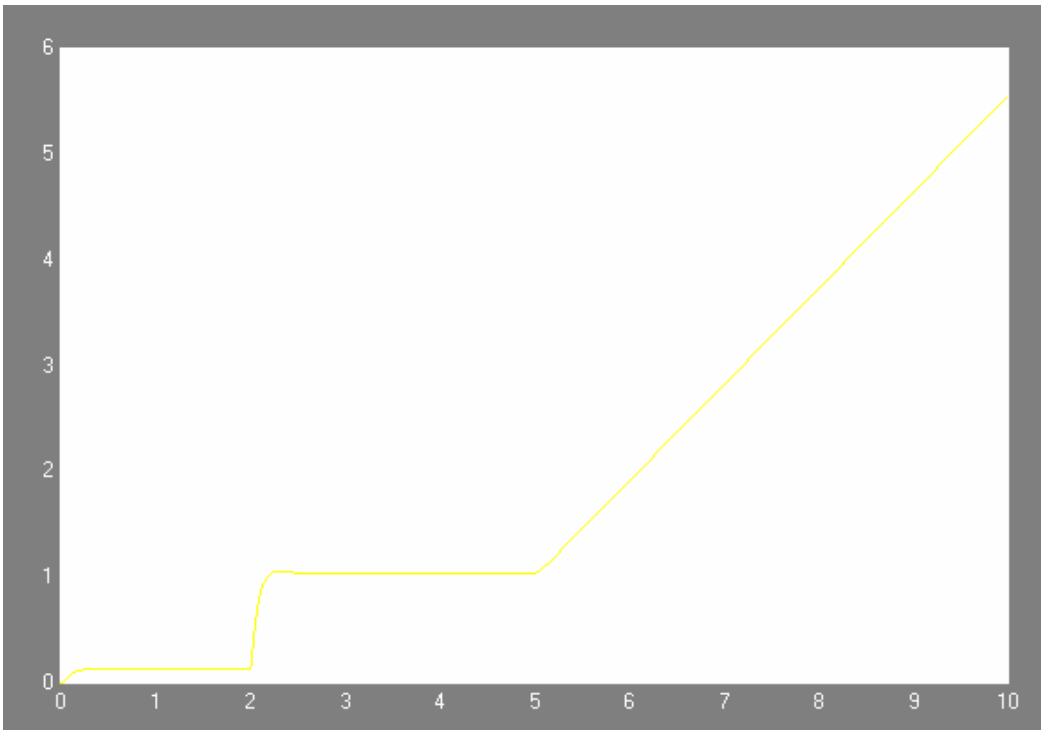
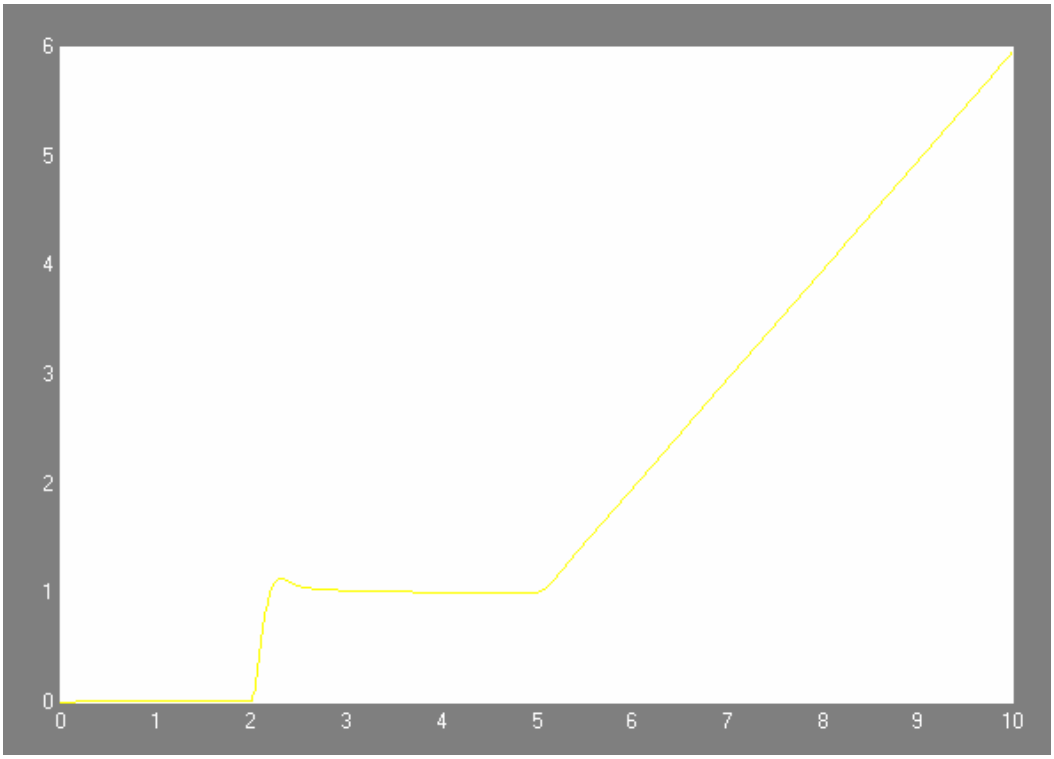


Figura 2.28 Salida del servosistema con el PID



Comparando estas dos graficas se puede observar que el neurocontrolador prácticamente hace disminuir los sobreimpulso que se generan a los cambios que se ve sometido el servosistema, caso contrario ocurre con el PID.

El neurocontrolador hace un mejor seguimiento de la señal del set-point que el realizado por el PID, debido a que el tiempo de respuesta del neurocontrolador es superior al mostrado por el PID.

La salida del servosistema presenta un pequeño error estacionario igual a 0.04, pero para algunas aplicaciones este error no presenta complicaciones para el sistema a controlar

3 APLICACIONES DE LOS NEUROCONTROLADORES

3.1 NEUROCONTROLADOR PARA SISTEMAS MECÁNICOS

Martha I. Aguilera Hernández
Jesús de León Morales

Resumen:
Este artículo presenta un estudio experimental de un controlador basado en redes neuronales diseñado para sistemas mecánicos que tienen menos actuadores que grados de libertad. El controlador es aplicado a un equipo de péndulo invertido para mostrar su desempeño.

INTRODUCCIÓN

Los controladores se utilizan en muy diversas áreas, básicamente en aquellas en donde la automatización es el elemento primordial para el desarrollo de sus aplicaciones.

Estos controladores deben ser robustos para asegurar estabilidad y mantener un desempeño aceptable bajo condiciones adversas de operación.

En el área de los sistemas mecánicos se ha presentado como un problema de control el caso cuando el número de entradas (actuadores) es menor que el número de salidas (grados de libertad). Un ejemplo de ello son los robots subactuados

(sistemas con menos controles que variables a controlar), el péndulo invertido en donde la velocidad lineal del carro representa la entrada del sistema, la posición angular del péndulo y lineal del carro son las correspondientes salidas medibles del sistema.

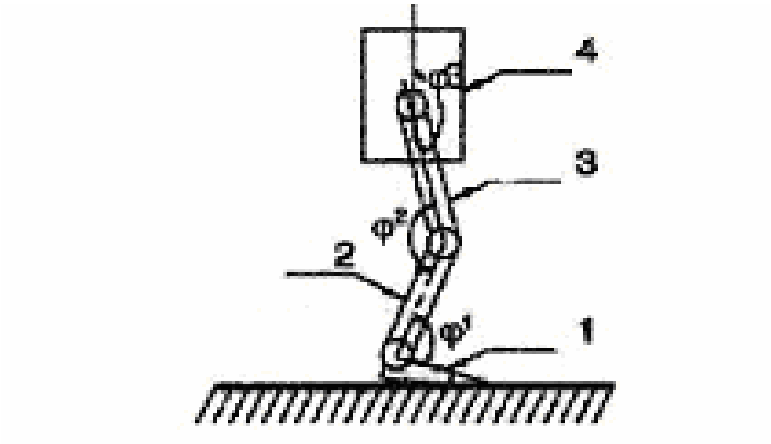
El péndulo invertido es un ejemplo clásico usado como prototipo de prueba. Es un sistema mecánico inestable cuando se encuentra en posición vertical, y el estudio de sus dinámicas es importante para el análisis de sistemas que tienen que mantenerse próximos a un punto de equilibrio inestable.

Ejemplo de ello son los sistemas robóticos móviles con patas, sistemas de navegación o antenas espaciales.

En la literatura, se pueden encontrar diferentes enfoques para resolver este problema de diseño de controladores. Por ejemplo, la linealización por retroalimentación de estado[11], pasividad, linealización de entrada-salidas técnicas basadas en métodos de Lyapunov. Sin embargo, el problema no se ha resuelto completamente, sobretodo para los sistemas subactuados, y el estudio continúa para encontrar caminos alternativos.

Figura 3.1 Modelo de un robot caminante.

[11] A. Isidori, Nonlinear Control Systems, Springer Verlag, N. Y :Segunda edicion, Ed. 1989



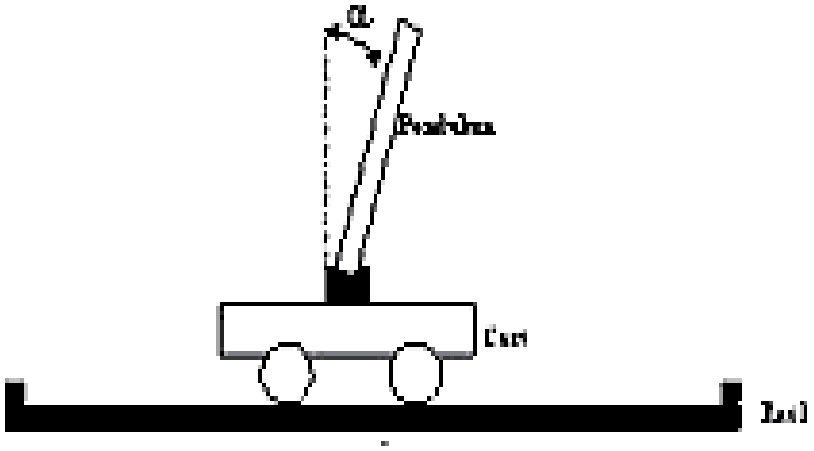
Por otra parte, las redes neuronales han sido utilizadas en la última década en diferentes aplicaciones, principalmente en la adaptación de los parámetros de un sistema cuando el modelo se desconoce. Entonces, vale la pena preguntarse ¿es posible utilizar las técnicas de redes neuronales para resolver el problema de control de sistemas subactuados? En este artículo se presenta una propuesta en donde se introduce un método para diseñar un controlador basado en redes neuronales que, combinado con técnicas de control lineal, resuelve el problema de control que se presenta en sistemas menos entradas que salidas. En este procedimiento, la idea principal consiste en definir una nueva salida que iguale el número de entradas y salidas. Esta nueva salida es diseñada mediante técnicas de redes neuronales que garantizan el desempeño del controlador propuesto. Además, se presenta el análisis de estabilidad del sistema en lazo cerrado.

El artículo está organizado como sigue: En la modelación del péndulo invertido, se presenta el modelo matemático. El controlador basado en redes neuronales será derivado el Diseño del controlador basado en RNA. También se muestran los resultados de simulación y de experimentación obtenidos. Finalmente, se presentan las conclusiones de este trabajo.

Modelo del péndulo invertido. El péndulo invertido consiste de una barra (péndulo) colocada en la parte superior de un carro con ruedas el cual se mueve a través de

un riel. Este sistema se muestra en la Fig. 3.1 El carro y el péndulo están restringidos a tener un movimiento horizontal.

Figura 3.2 Pendulo Invertido

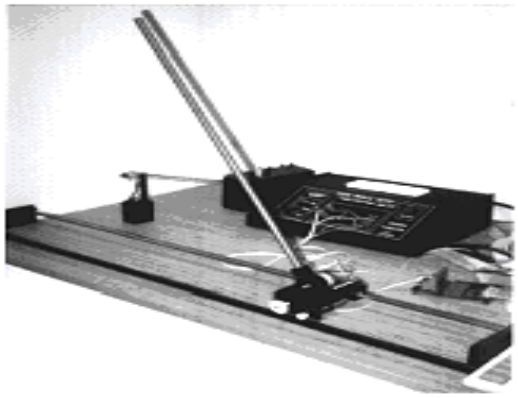


El equipo de instrumentación consiste de un actuador que controla la velocidad lineal del carro y dos sensores que miden la velocidad lineal del carro y la posición angular del péndulo. El objetivo de control es mantener en equilibrio al péndulo en la posición vertical y colocar al carro en una posición deseada. La velocidad lineal y angular puede ser determinada en forma indirecta por medio de la medición de la posición lineal del carro y el ángulo del péndulo. El equipo de péndulo invertido utilizado para estos experimentos se muestra en la Fig. 3.2.

El carro está equipado con un motor y un potenciómetro y se desliza por un riel. Estos están acoplados a un mecanismo que introduce la fuerza al sistema y mide la posición del carro y la posición del péndulo. Los datos se retroalimentan a una

computadora 80486 por medio de una tarjeta de adquisición de datos. En esta computadora se realiza el tratamiento de los datos y la programación del control utilizando lenguaje C.

Figura 3.3 Equipo del péndulo invertido.



El tiempo de muestreo es de 4 m/s, la masa del carro es de 455 grms., la máxima velocidad es de 1.09m/s, la máxima aceleración es de 3.0 m/s, la longitud del riel es de 0.914m., el diámetro del péndulo es de 1.27 cm., la longitud de 0.61 m., la masa es de 210 grms y el motor tiene una velocidad máxima de 6000 rpm.

El modelo matemático del péndulo invertido está dado por el siguiente sistema de ecuaciones diferenciales, basadas en consideraciones Lagrangianas.

$$\sum_p : \left\{ \begin{array}{l} (m_1 + m_2)\bar{z}_2 + m_2 l\alpha\cos(\bar{\alpha})^2 - m_2 l\alpha\sin(\bar{\alpha}) = u \\ m_2 l\alpha\cos(\alpha) - \frac{4}{3} m_2 l^2\alpha - m_2 gl\sin(\alpha) = 0 \end{array} \right\} \qquad 3.1$$

Donde z representa la posición lineal, u = fuerza de entrada al carro (N), m1=masa de la barra (péndulo, unidades Kg), m2= masa del carro (Kg), l = longitud del péndulo, a = ángulo de desviación con respeto a la posición vertical, g = fuerza gravitacional.

Diseño del controlador basado en redes neuronales. Las señales de referencia consideradas pueden ser modeladas por medio de un sistema dinámico denominado exosistema, que está representado por:

$$\sum_e : \left\{ \dot{\omega} = s\omega \right\} \qquad 3.2$$

Donde $\omega \in R^r$, es el conjunto de variables de entrada exógenas. El exosistema tiene la propiedad de Estabilidad Neutral, es decir, todos los eigen-valores se encuentran en el eje imaginario alrededor de w = 0. La utilidad básica del exosistema es la de proveer una señal de referencia “persistente” al sistema.

El problema de regulación de salida consiste en probar la estabilidad en lazo cerrado del sistema. $\sum_l$ usando el control dado por: U = L + Kx, Donde K y L son matrices que se eligen tal que el error de seguimiento e definido como: e = Qw + Hx tienda a cero cuando el tiempo tiende a infinito, para todo estado inicial y para toda entrada exógena.

Definiendo una nueva entrada a ser controlada como y = Hx. (donde H es una combinación lineal de estados conocida) se procede a diseñar una ley de control que siga una señal de referencia tal que el error de seguimiento tienda a cero exponencialmente. Para esto, primeramente se consideran las siguientes suposiciones.

Suposición 1.
El exosistema es estable neutralmente.

Suposición 2.
El par (A, B) es estabilizable.

Suposición 3.
Todas las componentes del vector de estado de la planta, Hy las variables exógenas son conocidas.

Lema 1.
Bajo las suposiciones 1 a la 3. Considere el sistema lineal $\sum_l$ con un controlador de la forma U= Kx + L w.

$$\sum_c : \left\{ x = Ax + Bu \;\; ; \;\; yc = Hx \right\} \; con \; H \in \; R^{m*n} \qquad 3.3$$

y un exosistema descrito por:

$$\sum_e : \left\{ \dot{\omega} = s\omega \;\; ; \;\; y_{ref} = -Q\omega \right\} \qquad 3.4$$

Donde $\omega \in R^r$ es el conjunto de variables exógenas de entrada. Si se define el error de seguimiento como e =Qw+Hx Entonces, el error de seguimiento tiende exponencialmente a cero cuando el tiempo tiende a infinito si

$$H(A+BK) = -a_0 H; HBL = QS - a_0 Q; a_0 \in R^{m*n} \qquad 3.5$$

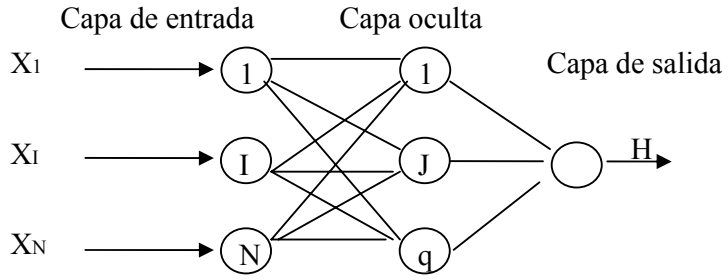
Para resolver este caso, se utiliza una Red Neuronal Multicapa (Multilayered Neural Network (MNN)) para identificar los coeficientes de H[12]. Unas de las arquitecturas de redes más populares es la de Propagación hacia atrás. Una red de este tipo con tres capas es utilizada y permite aproximar una gran variedad de funciones lineales y no lineales con una precisión aceptable El cálculo de Husando la Red de Tres Capas (MNN) se muestra en la Fig. 3.1, donde x representa los estados, h representa los pesos de la MNN, que presenta las cantidades en la capa escondida.
El vector de entrada dado por $X = (x_1, x_2 ..., x_N)^T$ se aplica a la capa de entrada.

Las unidades de entrada distribuyen estos valores a las unidades escondidas. Sin pérdida de generalidad, se asume que la activación del nodo es igual a la entrada de la capa, por lo tanto se tiene que la salida de la capa escondida es dada por x_i^T.

El algoritmo de aprendizaje (es decir encontrar el conjunto de pesos adecuados) que se utilizó es una técnica denominada de descenso acelerado

Figura 3.4 red neuronal de tres capas.

[12] FREEMAN. James y SKAPURA. David. Redes Neuronales: Algoritmos, aplicaciones y técnicas de programación. Delaware E.U.A: Addison Wesley Iberoamericana S.A. 1993



El objetivo de utilizar MNN es minimizar el error de seguimiento dado por e= Yc-Yref usando la técnica de descanso acelerado.

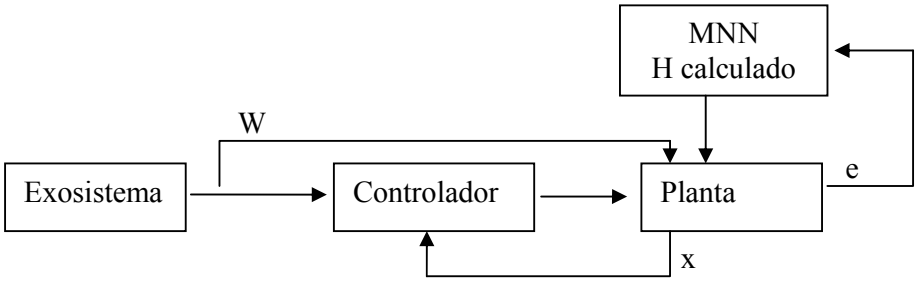
Donde el factor positivo ζ <1es llamado el parámetro de razón de aprendizaje y el superíndice “o” se utiliza para definir los valores de salida.

Entonces, se tiene que el gradiente de está dado por

$$\frac{DH}{DT} = -\xi e x^T$$

Ahora, considere el sistema lineal controlado por la entrada u y teniendo como salida y donde los coeficientes de Hson determinados mediante la ley de adaptación dada por $\xi e x^T$, tal que el error de seguimiento e tienda a cero exponencialmente. El esquema general se muestra en la Fig. 3.5

Figura 3.5 Esquema general del controlador



El sistema de ecuaciones que describen al péndulo invertido fueron simuladas con los siguientes valores nominales.

G= 9.8, l=0.6, m1=0.455, m2 = 0.210

El valor del parámetro ζ utilizado en la ley de adaptación fue de 0.005. Los valores iniciales de los pesos se fijaron a 0.5. Las simulaciones se desarrollaron con Matlab. La razón de aprendizaje de la MNN se muestra en 3.6 Se puede observar como la red MNN se entrena hasta que el error se reduce a una valor menor de 0.001. Las gráficas 3.7 a 3.8 muestran el comportamiento obtenido en simulación y experimentación del sistema en lazo cerrado. La Fig. 3.7 muestra la posición del ángulo a, donde a pesar de mostrar unas oscilaciones al inicio, éstas se van haciendo más pequeñas hasta que alcanza el objetivo. El tiempo en que lo logra es menor a 4 segundos. La posición del carro y la velocidad lineal se muestra en las Fig. 3.7 y 3.8 Respectivamente.

Figura 3.6 Error y razon de aprendizaje de MNN

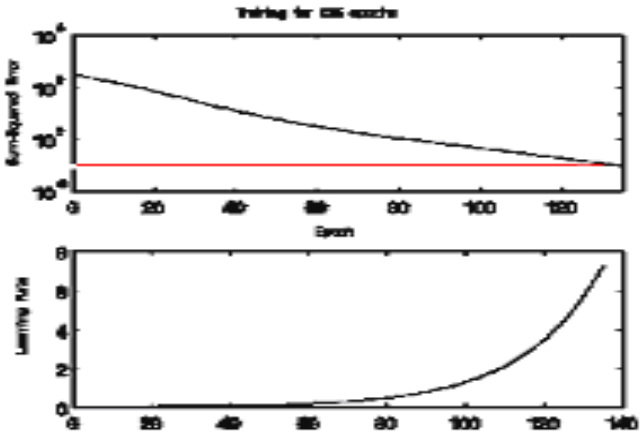


Figura 3.7 Posición angular

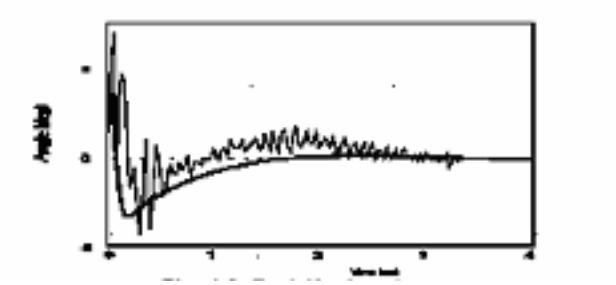
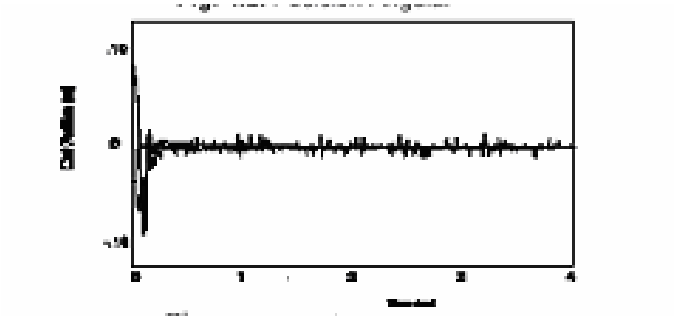


Figura 3.8 Posicion del carro.



3.2 NEUROCONTROL COMO HERRAMIENTA EN LA ATENCIÓN FARMACÉUTICA.

Emilio Soria Olivas (1), Antonio J. Serrano López (1), Gustavo Camps Valls (1), Jose D. Martín Guerrero (1), N. Víctor Jiménez Torres(2)-(3)

(1) Grupo de Procesado Digital de Señales
Dpto Ingeniería Electrónica, Universitat de València.

(2)Servicio de Farmacia del Hospital Universitario Dr. Peset de Valencia;

(3)Dpto Farmacia y Tecnología Farmacéutica, Universitat de València.

Resumen.

Se describen las redes neuronales artificiales y su potencial aplicación en la predicción de la morbilidad relacionada con los medicamentos en los pacientes. Estos sistemas usados en otros campos del conocimiento como elementos de ayuda no se han introducido aún en el campo de la farmacoterapia donde se usan otras herramientas matemáticas (regresión logística) menos potentes. El objetivo de este trabajo es introducir el uso de estos sistemas en la prevención de problemas relacionados con los medicamentos en los pacientes a nivel individual.

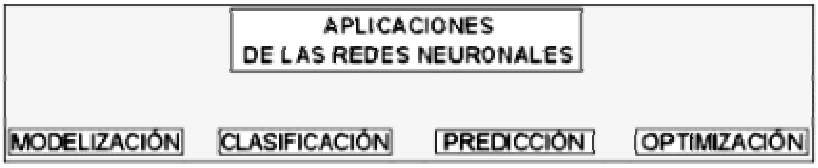
INTRODUCCIÓN.

El uso de las redes neuronales artificiales en diferentes campos del conocimiento (control, procesado de la señal, sistemas expertos, predicción en series temporales, etc) han sufrido un crecimiento exponencial en los últimos años[13]. A modo de ejemplo, una búsqueda realizada en 1999 en Internet con el buscador Infoseek

[13] Haykin, S. Neural Networks: a Comprehensive Foundation. Prentice-Hall, 1998

usando como palabras clave ‘neural networks’ proporciona un total de ¡26.161.995 referencias!. Este número tan elevado se explica por el enorme campo de aplicación de las denominadas redes neuronales artificiales, figura 3.9:

Figura 3.9 Aplicaciones de las redes neuronales.



Las redes neuronales son preferibles a otros métodos matemáticos cuando se presentan las siguientes características:

1. Es muy difícil encontrar las reglas que definen la variable a modelizar en relación a las variables independientes consideradas para el modelo.

2. Los datos son imprecisos o contienen perturbaciones estadísticas (ruido).

3. El problema necesita para ser definido un gran número de variables dependientes (presenta una alta dimensionalidad).

4. El modelo a resolver es no lineal.

5. Se dispone de un gran número de datos.

6. El entorno de trabajo es variable.

Las características precedentes son representativas de la variabilidad en farmacología, figura 2, por lo que la aplicación de estos elementos presentará ventajas sobre los actuales, sin que ello signifique no reconocer sus limitaciones[14]. En farmacoterapia hay necesidad de determinar modelos para la prevención de problemas relacionados con la morbilidad asociada a los medicamentos, en

[14] Fleiss, J.L. “Analysis of Data from Multiclinic Trials”. Controlled clinical Trials, vol 7, pp 267-275, 1986.

particular cuando se analizan los factores que influyen en la respuesta farmacológica (figura 3.10):

Figura 3.10. Variabilidad farmacológica.

De acuerdo con lo antedicho, y en concordancia con la filosofía que sustenta la atención farmacéutica [3.6], el valor y la aplicación de las redes neuronales en este campo es obvio por cuanto:

1. El cuerpo humano y sus interacciones entre diferentes elementos de su entorno es uno de los sistemas más complejos que existen. En principio es lógico pensar que estas relaciones son no lineales. En la figura 2 los procesos que aparecen

sombreados y que pueden ser no lineales en algún momento constituyen etapas claves en la secuencia definida.

2. Existen muchas variables para definir el comportamiento de un fármaco determinado. En consecuencia cuanto más se simplifique el problema más errores contendrá el modelo. Existe pues un problema no lineal con una alta dimensionalidad.

3. Las hojas de recogida de datos de los pacientes sobre una determinada patología pueden estar incompletas o presentar errores de medida.

4. Los datos clínicos, aumentan con el tiempo por lo que un sistema que se adapte, ofreciendo validez y fiabilidad a la nueva información obtenida sería la mejor elección.

En este contexto, el campo de la atención farmacoterapéutica es ideal para la aplicación de estos métodos. Sin embargo existen muy pocos trabajos en la bibliografía especializada donde se hayan usado, circunstancia que se debe al recelo que despiertan el uso de herramientas conocidas como cajas negras; es decir, dan buenos resultados pero se piensa que es fruto de la “casualidad”. Esta impresión no es real por cuanto que:

- Existen libros de redes neuronales escritos por matemáticos estadísticos de gran prestigio internacional[15].
- En libros de análisis multivariante clásicos se da como camino a seguir las redes neuronales.

[15] B.D. Ripley, Pattern Recognition & Neural Networks. Cambridge University Press, 1996.

- Programas informáticos usualmente usados en análisis de datos, por ejemplo el famoso y ampliamente extendido SPSS, incluyen módulos de redes neuronales.

- Existen multitud de demostraciones de la convergencia de los valores de salida de un tipo de red neuronal, el perceptrón multicapa, hacia los valores definidos por el Teorema de Bayes; herramienta de uso común en problemas de clasificación.

Existen, por tanto, herramientas muy potentes contrastadas por ingenieros y matemáticos pero que no se aplican al campo de la atención farmacéutica donde la búsqueda de modelos es una constante para predecir posibles efectos adversos de los medicamentos. Este trabajo pretende dar a conocer estos métodos para que los diferentes especialistas las empiecen a aplicar mejorando sus modelos definidos por otras técnicas, y, por tanto, mejorando la calidad de vida de los pacientes que, en definitiva, es nuestro objetivo final.

Predicción de la emesis tras la terapia antineoplásica altamente emetógena. El fallo en el control de la emesis (nauseas y vómitos) posterior quimioterapia disminuye la efectividad de los tratamientos de cáncer y empeora la calidad de vida del paciente. La disponibilidad de una herramienta capaz de modelizar la relación entre la medicación y el tipo de protección del paciente frente a emesis permitiría, sin duda, optimizar el tratamiento. Con esta intención se desarrolló una red neuronal con el objetivo de discriminar entre pacientes con protección mayor (<3 vómitos) y menor (o fracaso) frente a emesis durante las 24 horas siguientes al tratamiento, etapa de mayor riesgo, en función de las características fisiológicas del paciente, la pauta de los citostáticos y de los antieméticos empleados. Para este problema se utilizó un

perceptrón multicapa totalmente conectado con 19 neuronas de entrada correspondientes a las diferentes variables independientes escogidas por expertos del hospital y una neurona en la salida que determina el grupo del paciente en función del signo de su valor de salida.

El conjunto de entrenamiento y generalización estaba formado por 256 y 63 casos respectivamente. Con estos conjuntos de datos se consiguieron los siguientes resultados:

Tabla 3.1 Conjuntos de datos

Entrenamiento	**PME**	**PMA**	**Generalización**	**PME**	**PMA**
Test PME	**54**	**16**	**Test PME**	**16**	**6**
Test PMA	**7**	**179**	**Test PMA**	**2**	**39**

Sensibilidad 91% Sensibilidad 86%
Especificidad 88% Especificidad 88%
Acierto 91% Acierto 87,3%

Si se utiliza un modelo de regresión logística los mejores resultados obtenidos, en cuanto a sensibilidad y especificidad fueron:

ENTRENAMIENTO GENERALIZACIÓN
Sensibilidad: 75% Sensibilidad: 75%
Especificidad: 66% Especificidad: 69%

Comparando los resultados obtenidos la superioridad del modelo definido por la red neuronal es evidente.

Predicción del riesgo de intoxicación por digoxina[16]. En esta aplicación el objetivo era predecir si la concentración en sangre de esta sustancia era mayor o menor que 2 ng/ml. Esta clasificación se hacía en función de datos antropométricos, tratamiento con amiodarona, aclaramiento de creatinina y la pauta y vía en la administración de este fármaco. Los conjuntos de entrenamiento estaban formados por 75 y 33 casos respectivamente.

Al igual que en el caso anterior se usó el método de regresión logística para este problema. También se definieron dos conjuntos, entrenamiento y validación, sin embargo los resultados eran tan bajos para el conjunto de validación que se usaron todos los datos para construir el modelo (no se validó por tanto). Los resultados obtenidos fueron:

	CP > 2ng/ml	CP < 2ng/ml
Test +	20	24
Test -	6	58

Sensibilidad: 77%
Especificidad: 70%
Aciertos: 72,2%

Otra vez (y ahora de forma más evidente) la red neuronal demuestra su superioridad.

Para estos problemas se han desarrollado unas aplicaciones amigables e intuitivas de manejar. Son herramientas de ayuda a la decisión, nunca de sustitución, para el

[16]A.J, Serrano. Uso de redes neuronales en el problema de la emesis posquimioterapia. Tesis de Licenciatura, Valencia 1998

profesional clínico. Una descripción más completa de esta aplicación se encuentra en la página WEB http://imf.es/

En la actualidad los autores de este trabajo junto a otros profesionales investigan con la subvención de un proyecto FEDER otros problemas que son susceptibles de ser resueltos con estas herramientas.

CONCLUSIONES.

En este trabajo se exponen brevemente las bases de una disciplina que ha sufrido una producción científica en los últimos años. De forma sorprendente, estos métodos, ampliamente usados en otros campos del conocimiento, no han sido aplicados de forma generalizada en el ámbito clínico como elementos de ayuda. No obstante, el perceptrón multicapa, junto con su algoritmo de aprendizaje, el backpropagation han proporcionado resultados clínicos altamente satisfactorios.

3.3 MODELAMIENTO Y CONTROL NEURONAL EN TIEMPO REAL DE UNA PLANTA TIPO CÓNICA PARA EL CONTROL DE NIVEL

José Gallardo, Rodrigo Arriagada, Danilo Bassi

Departamento de Ingeniería de Sistemas y Computación,
Facultad de Ingeniería. Universidad Católica del Norte.
Av. Angamos 0610. Antofagasta - Chile.
e-mail: jgallard@socompa.ucn.cl

Departamento de Ingeniería Informática,
Facultad de Ingeniería. Universidad de Santiago de Chile.
Av. Ecuador 3659. Santiago - Chile.
e-mail: dbassi@diinf.usach.cl

Introducción

La problemática del control consiste en lograr que un sistema, (colección de elementos que interactúan entre sí), produzca una salida que se mantenga en ciertos niveles preestablecidos aun cuando existan influencias externas (perturbaciones) que actúen negativamente, en sentido contrario al del control, y cambios en las condiciones de operación.

Esto es lo que se define como la acción de control.
Considerando que las redes neuronales artificiales tienen la capacidad de almacenar conocimiento basado en ejemplos, si se aplican en un lazo de control enseñándoles la forma adecuada de controlar, estas pueden producir el efecto deseado.

El uso de las redes neuronales para diseñar sistemas de control dio origen a una disciplina denominada comúnmente 'neurocontrol'. Ésta es una forma de realizar

control avanzado. Con el término control avanzado se designa a la filosofía de control automático que trata de mejorar el control clásico,(control PID) empleando técnicas innovadoras tales como los sistemas expertos, la lógica difusa, visión artificial, control predictivo, control adaptivo y control usando redes neuronales artificiales.

Estos métodos de control son necesarios cuando el control clásico no presenta un buen desempeño, lo que tradicionalmente ocurre en procesos multivariables o de tipo no lineal. Se debe considerar que la mayoría de los procesos reales son no lineales. El control clásico trabaja bien sólo en torno a puntos de operación cuya vecindad es lineal.

Es aquí cuando entran en juego las redes neuronales artificiales, que tienen la característica de realizar un mapeo exacto de funciones altamente no lineales, por lo que su aplicación en el control automático es inmediata.
En este artículo, se describen el diseño e implementación de un sistema de neurocontrol de un proceso real de tipo no lineal y se consignan los resultados obtenidos.

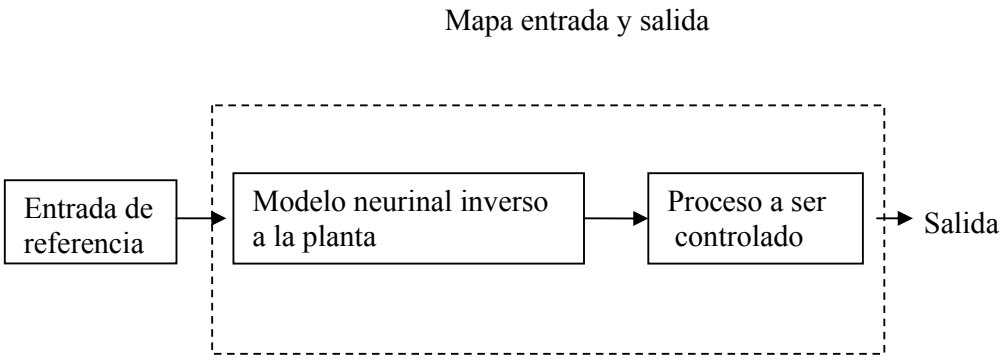
Diseño del controlador neuronal. En la figura 4.1 se consigna el diagrama en bloques del sistema de control neuronal a implementar. Como se puede observar, el control es llevado a cabo por una red neuronal que representa el modelo inverso de la planta o proceso a controlar.

Este modelo inverso neuronal es obtenido en lazo abierto aplicando un entrenamiento de tipo fuera de línea. Una vez entrenada la red, que representa el modelo inverso del sistema, ésta se conecta simplemente en cascada con el proceso a controlar.

El conjunto red neuronal inversa - proceso (descrito por el bloque en línea punteada) representa un mapeo identidad entre la entrada (referencia) y la salida de la planta

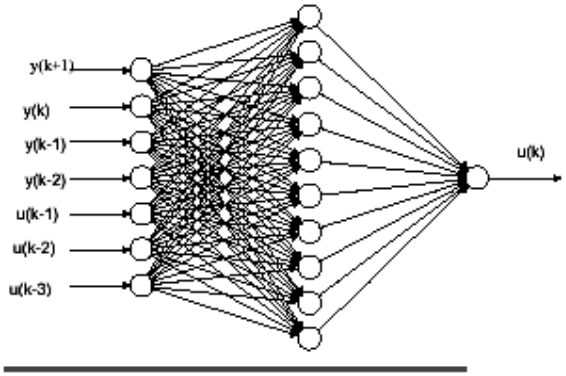
(variable controlada) y por tanto, al representar este conjunto un mapa identidad, evidentemente la salida deberá ser idéntica a la entrada, con lo cual se consigue el objetivo de control deseado. Las únicas limitaciones para el buen desempeño de este esquema de control son que el modelo neuronal inverso de la planta debe representar la dinámica inversa del proceso en forma exacta, y además, la planta debe cumplir con las condiciones de invertibilidad.

Figura 3.11 Diagrama esquemático del neurocontrolador en base al modelo inverso.



La red neuronal que representa el modelo inverso de la planta se ilustra en la figura 3.11,

Figura 3.12 Estructura de la red neuronal.

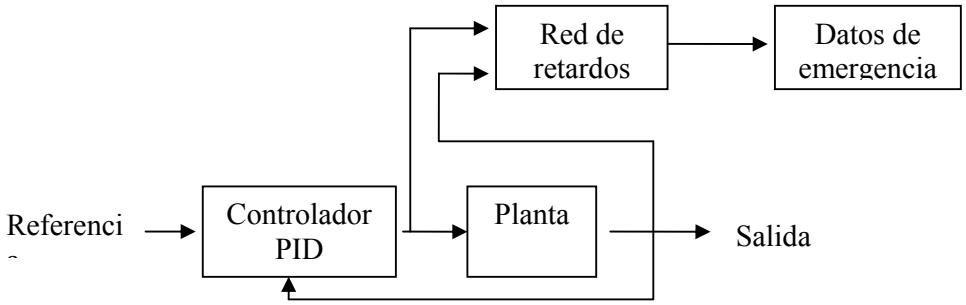


Como se puede observar, se utiliza una red neuronal con una sola capa oculta de 10 neuronas con función de activación tangente hiperbólica y función de activación lineal en la capa de salida, dado que, cualquier sistema dinámico no lineal puede ser modelado con un grado de exactitud deseado por una red de una sola capa oculta, de un número adecuado de neuronas[17]. De acuerdo al teorema de Kolmogorov [Hecht-Nielsen'88], se asegura la convergencia eligiendo un número de neuronas para la capa oculta igual a 2*N+1, donde N representa el número de entradas a la red. Para la red en discusión se requerirían entonces 15 neuronas, puesto que el número de entradas es igual a 7, sin embargo, para acelerar la velocidad de convergencia, se puede establecer un compromiso entre exactitud y complejidad, y es así que se utilizaron sólo diez neuronas, probándose empíricamente que con este número se lograba un buen grado de exactitud (bajo error en la convergencia).

La figura 3.13 muestra el esquema implementado para obtener los datos requeridos para entrenar la red neuronal presentada en la figura 3.11.

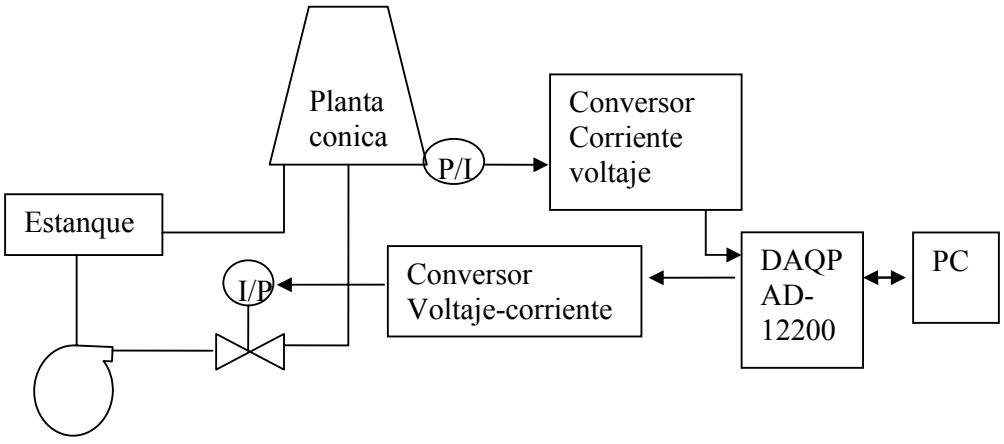
[17] Zamarreño J.M. , Vega P.,Neural predictive control: application to a hightly non-linear process, World Congress International Federation of Automatic Control, vol. C: Control Design I, 19-24 , 1996, IFAC'96.

Figura 3.13 Esquema de adquisición de datos de entrenamiento.



Esquema general del sistema. La figura 3.14 muestra el diagrama P&ID (Process and Instrumentation Diagram) que representa el sistema de control neuronal en base al modelo inverso de una planta cónica no lineal de 2º orden

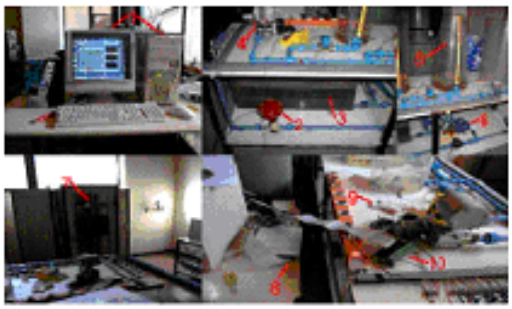
Figura 3.14 Diagrama P&ID del sistema.



Este diagrama a su vez se ilustra con una fotografía del sistema de control en tiempo real, que fue implementado en el laboratorio de control automático del Departamento de Ingeniería Eléctrica de la Universidad de Santiago de Chile. En esta imagen se

pueden apreciar todos los componentes que conforman el sistema de control, esto es:

Figura 3.15 Fotografía del sistema implementado.



1-Computador tipo PC, el cual almacena y ejecuta el algoritmo de control

2 -Actuador, válvula de control tipo diafragma que realiza la manipulación de la variable de salida.

3 -Estanque, que almacena el fluido saliente de la planta para recircularlo con la bomba.

4 -Conversor presión – corriente, el cual convierte la presión de la columna de líquido de la planta en corriente, para digitalizarla.

5 -Planta cónica no lineal a controlar.

6 -Bomba, que permite circular el fluido.

7 -Conversor corriente – presión, que controla la presión a ingresar a la válvula de control.

8 -Adquisidor de datos DAQPad 1200, el cual digitaliza la salida de la planta y entrega la señalen voltaje de la actuación necesaria.

9 -Conversor voltaje – corriente.

10 -Bornera de conexiones CB – 50.

Descripción de los módulos del controlador. Para ejecutar el sistema de control, éste se divide básicamente dos módulos: uno encargado del pre - tratamiento de los datos que ingresan a la red (normalización), el cual se llamará modulo de adquisición de datos, y otro módulo llamado módulo de ejecución, el cual se encarga de ejecutar la red neuronal controladora, con los datos obtenidos por el módulo de adquisición, y así generar la señal de control.

Dichos módulos se detallan a continuación:

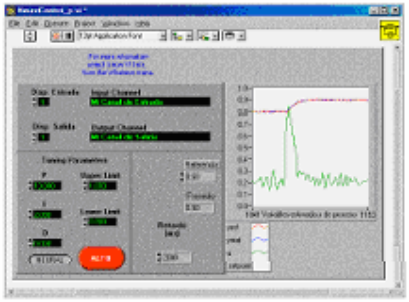
✓ Módulo de adquisición de datos

Este módulo esta constituido por software y por hardware. En cuanto al software, éste recibe los datos normalizados y los retarda con el uso de registros de desplazamiento, en cuanto al hardware, se posee un par de circuitos conversores los cuales se encargan de convertir las señales, tanto de salida como de entrada, a los niveles permitidos. Ya que el adquisidor sólo funciona con señales de voltaje, por lo que la corriente de salida del conversor P/I debe ser convertida a voltaje, y la señal de voltaje de salida del convertidor digital análogo del adquisidor de datos debe ser convertida a corriente antes de ingresar al convertidor I/P.

✓ Módulo de ejecución

Este módulo consiste básicamente en la ejecución de la red neuronal diseñada anteriormente (figura 3.12), en la cual sus parámetros (pesos) ya fueron determinados, por tanto se encarga sólo de generar la acción de control requerida para el vector de entrada necesario.

Figura 3.16 Pantalla del sistema de control en Labview.



Presentación de resultados. En forma previa a la presentación gráfica de los resultados que se obtuvieron al aplicar un controlador neuronal para controlar el sistema de control de nivel, es importante destacar las consideraciones bajo las cuales se obtuvieron los datos para entrenar la red controladora.

Considerando lo expuesto en[18], los datos para entrenar la red neuronal de control deberán ser obtenidos a partir de un sistema controlado por algún controlador convencional. La razón que justifica tal determinación es que es conveniente entrenar una red de control con datos provenientes de un controlador para que el dominio de datos con los que deba operar la red controladora se restrinja al universo de datos reales provenientes de una planta controlada. Esto permitirá un mejor desempeño de la red neuronal de control, por tanto, para el presente estudio y considerando que se cuenta con un controlador PID para controlar el proceso en cuestión, los datos de entrenamiento fueron obtenidosa partir del sistema controlado por un controlador PID.

---

[18] Zamarreño J.M. , Vega P.,Neural predictive control: application to a hightly non-linear process, World Congress International Federation of Automatic Control, vol. C: Control Design I, 19-24 , 1996, IFAC'96.

Otro punto importante a considerar es que para efectos de estabilidad numérica, los datos de entrenamiento y los de la posterior ejecución de la red neuronal, deben ser normalizados entre 0 y 1. Para estos efectos el algoritmo empleado fue el siguiente:

Para los datos de entrada se usó la ecuación 3.6.

$$\bar{y} = \frac{Y - V_L}{V_H - V_L} \qquad 3.6$$

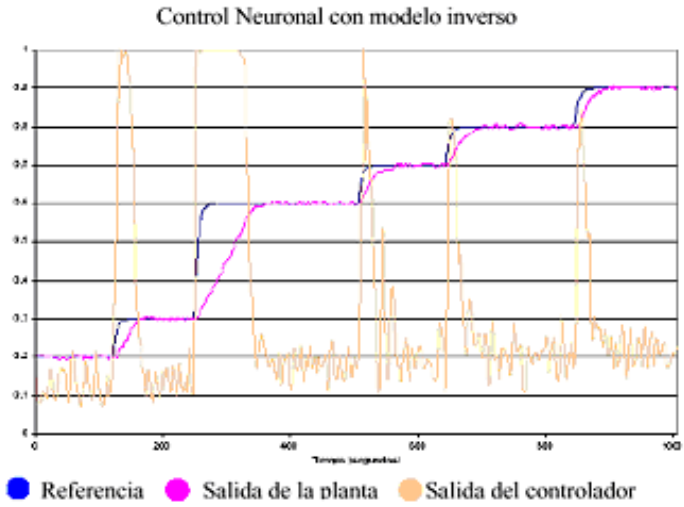
Donde, 'y'es la salida de la planta, 'V_L'el límite inferior del voltaje que entrega el conversor corriente– voltaje, 'V_H ' el límite superior que entrega dicho conversor, e 'y' es la salida normalizada de la planta La señal de control que se obtiene de la ejecución de la red, por motivos de entrenamiento también es una señal normalizada entre 0 a 1, por lo que se debe desnormalizar con la ecuación 3.7.

$$u = \bar{u}(U_H - U_L) + U_L \qquad 3.7$$

Donde 'u' es la salida de la planta, 'U_L' es el límite inferior del voltaje que necesita el conversor voltaje - corriente, 'U_H' el límite superior, y u es la salida normalizada de la red de control.

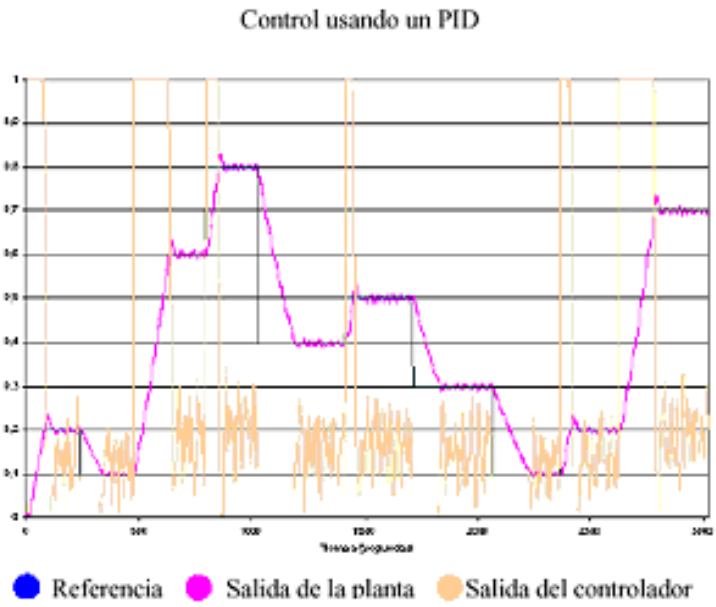
Hechas las consideraciones ya mencionadas, es posible mostrar los resultados producto del controlen tiempo real de una planta de nivel no lineal cónica de 2º orden. Estos son consignados en la figura 3.17.

Figura 3.17 Control con modelo neuronal inverso.



Para propósitos de evaluación del sistema de control neuronal implementado se ha estimado adjuntar, los resultados obtenidos al controlar la planta de nivel mediante un controlador convencional. En este caso éste es representado por un tradicional controlador PID. Los resultados que se obtuvieron fueron logrados con similares condiciones en cuanto a los cambios en el punto de referencia. Estos resultados se muestran en la figura 3.18.:

Figura 3.18 Control mediante un controlador PID.



DISCUSIÓN DE RESULTADOS. Los resultados logrados producto de la aplicación de un esquema de control basado en el modelo inverso del sistema, han sido bastante satisfactorios, como se puede observar en las curvas consignadas en la figura 3.17 Como era de esperar, el control neuronal implementado tiene un mejor desempeño en el control de la planta cónica de nivel (descrita por un sistema no lineal de 2º orden), que un controlador PID. Esto es evidente y esperado por cuanto el uso de controladores PID no es la mejor alternativa para controlar un sistema no lineal, si el sistema no es linealizado en torno a un cierto punto de operación. Como se puede observar en la figura 3.17 el uso de un controlador PID produce.

Sobreimpulsos transciendes de muy elevada magnitud que pueden dañar los sensores del sistema si esto no se prevé utilizando protectores ad-hoc. Sin embargo, el controlador neuronal tuvo la capacidad de eliminar en la etapa transciende de respuesta dichos sobreimpulsos. También se puede observar que los tiempos de

establecimiento son bastante pequeños, y los errores de estado estacionario despreciables.

Otro punto importante de destacar es que la implementación de un neurocontrolador basado en el modelo inverso del sistema no es una tarea simple, por cuanto inicialmente el modelo inverso que se obtuvo fue el modelo inverso de la planta en lazo abierto. Como resultado de esta primera aproximación se pudo constatar experimentalmente que el esquema así implementado no funciona correctamente, es excesivamente lento y esto es debido a que la red no fue entrenada con el universo esperado de señales de control. Este problema constituye una limitante en la aplicación de este esquema de control, por cuanto el sistema deberá contar con un sistema de control previamente.

CONCLUSIONES.

En forma inicial y previa a la implementación de un sistema de control en tiempo real, de acuerdo a los diversos estudios realizados sobre esquemas de control neuronal desarrollados en trabajos de investigación, se pudo advertir que uno de los elementos claves en el desarrollo de esquemas de control neuronal es el modelamiento del proceso a controlar, y una de las herramientas más adecuadas para estos efectos es el uso de las redes neuronales, razón por la cual se estudiaron las arquitecturas de redes más adecuadas para el modelamiento de procesos, concluyéndose que estas arquitecturas son redes realimentadas y redes feedforward con dinámica externa. Sin embargo, se utilizaron redes feedforward tipo perceptrón multicapa, en consideración a: su simplicidad, el amplio conocimiento de estas y por lo expresado en [Nerrand'94], quien establece que cualquier red realimentada puede ser expresada como una red feedforward con dinámica externa.

Se ensayaron distintas topologías, pudiéndose establecer experimentalmente, que redes con una sola capa oculta y un número adecuado de neuronas en dicha capa, modelan adecuadamente cualquier sistema dinámico no lineal. También se experimentaron con los dos típicos métodos de entrenamiento, "en línea" y "fuera de línea", lográndose establecer que un esquema de control con entrenamiento fuera de línea, para el modelo del proceso, es más adecuado que el uso de entrenamiento en línea, si se requiere utilizar el modelo en esquemas de control no adaptativos. Por cuanto, en un esquema con entrenamiento fuera de línea, se logra una convergencia en forma mucho más rápida.

También, en forma previa a la implementación en tiempo real del esquema de control seleccionado, se efectuaron una serie de ensayos de simulación con diversos esquemas de control neuronal, concluyéndose que el esquema más adecuado para efectuar una primera aproximación de control neuronal de procesos reales, es el control directo con uso del modelo inverso del proceso, por su buen desempeño en el control de diversos sistemas lineales y no lineales, y su relativa simplicidad Luego de experimentar el esquema elegido, en el control en tiempo real de una planta compleja (planta cónica, tipo no lineal 2º orden), se pudo observar la diferencia que existe entre experimentar un determinado esquema de control, mediante ensayos de simulación, y un control en tiempo real.

La migración de un esquema de control diseñado y experimentado mediante simulación no es simple ni de aplicación inmediata en un sistema real, por cuanto se deben tomar en cuenta muchos factores que no se consideran en la simulación, como por ejemplo: limitaciones físicas en las variables manipuladas, esto es, no cualquier magnitud en la señal de control puede lograrse en un sistema real por problemas de saturación, tampoco la rapidez de cambio en las señales de control de esquemas simulados es posible lograr en sistemas reales.
Otro aspecto importante a considerar es que las señales provenientes de los distintos equipos (actuación y sensores) deben ser adecuadas para su procesamiento digital y normalizadas antes de alimentar la red neuronal.

Al efectuar las pruebas del sistema implementado, como se observa en la figura 3.17, los resultados de la aplicación de un esquema de control neuronal directo en base al modelo inverso, en el control de la planta cónica de nivel, son muy superiores a los logrados con un controlador convencional PID. Esto es de esperar, por cuanto un controlador PID no es el más adecuado para el control de un sistema dinámico no lineal, si éste no es linealizado en torno a un cierto punto de operación.

3.4 PREDICCIÓN DE DEMANDA DE TRANSPORTE POR CARRETERA MEDIANTE EL EMPLEO DE REDES NEURONALES ARTIFICIALES

RESUMEN

Este artículo describe como se puede emplear una RNA del tipo backpropagation para predecir una serie temporal y la aplicación al caso concreto del número de operaciones totales de transporte que tienen como destino la comunidad autónoma de Aragón. Los datos empleados son una extrapolación de los remitidos por el Ministerio de Fomento en su encuesta permanente de Transporte de Mercancías por Carretera. Finalmente se compara el resultado obtenido con el producido por otras técnicas estadísticas.

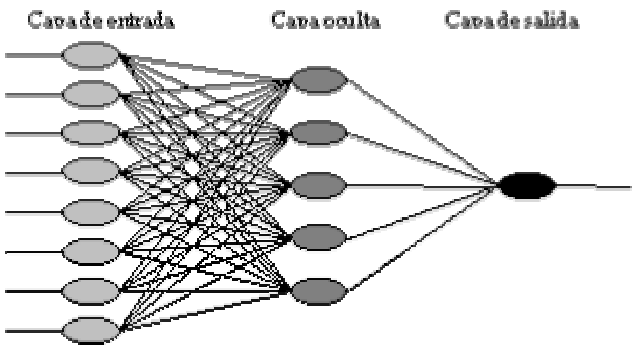
INTRODUCCIÓN

En resultados obtenidos en encuestas como la Encuesta Permanente de Transporte de Mercancías por Carretera, publicada por el Ministerio de Fomento, se pretenden conocer, mediante la investigación de las operaciones de transporte efectuadas por vehículos pesados, los flujos de mercancías realizados en este modo de transporte, consiguiendo de esta manera la evaluación del nivel de actividad del sector. Se define operación de transporte como el desplazamiento de mercancías sobre un vehículo a una distancia determinada, teniendo en cuenta también las operaciones en vacío. En este artículo en concreto se han analizado los resultados provenientes de las operaciones de transporte realizadas en la comunidad autónoma de Aragón

por vehículos pesados autorizados cuya carga útil sea superior a 3.5 toneladas y cuyo peso máximo autorizado supere las 6 toneladas. No se consideran aquellas operaciones que no supongan para el vehículo el cambio de término municipal.

Descripción de la red Backpropagation. La red neuronal artificial de tipo Backpropagation consta de una capa de neuronas de entrada, otra de neuronas de salida y, opcionalmente, una o varias capas de neuronas ocultas (Fig.3.19). El número de neuronas que constituyen las capas de entrada y de salida está determinado por la aplicación para la que se dimensione la red. El número de capas ocultas y el de neuronas en cada capa oculta no dependen de la naturaleza de la aplicación, sino que deben ensayarse múltiples combinaciones para obtener la mejor generalización[19].

Figura 3.19 Arquitectura de una red Backpropagation.



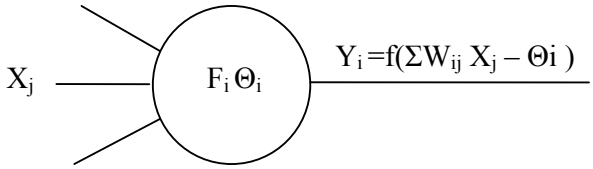
.

Cada una de las neuronas de la red (excepto las de entrada) produce una salida al aplicar una función de transferencia a la suma ponderada de las entradas por los

[19] WIDROW, Bernard y LEHR, Michael A. 30 years of adaptive neural networks: Perceptron, Madaline, and Backpropagation. Procedimiento de l IEEE, vol 78 #9, Sptiembre 1990, pp 1415-1442

pesos sinápticos (Fig.3.20). Cada neurona recibe la información procedente de las neuronas que se encuentran en una capa más cerca de la capa de entrada, y envía la salida hacia una capa que está más cerca de la capa de salida. De este modo la información fluye siempre hacia adelante.

Figura.3.20 Función que realiza una neurona artificial.



El modo de operación de la red consta de una fase de aprendizaje y una de recuerdo. La fase de aprendizaje es un proceso iterativo en el que se presentan ejemplos significativos al sistema y la salida que debería obtenerse, ajustando mediante un algoritmo las conexiones o pesos sinápticos, siendo en los valores finales de éstos donde se almacena el conocimiento. Tras esta fase, en el posterior recuerdo se pueden introducir ejemplares nunca vistos en el sistema obteniendo salidas satisfactorias.

Aplicación de técnicas predictivas. Obtención de resultados y análisis. Realizamos una extrapolación de los datos hasta obtenerlos mensuales, por lo tanto contamos con 48 datos históricos a partir de los que realizar este estudio comparativo. Como primera elección, consideraremos dentro del análisis los 42 primeros, dejando el último semestre del año 1996 como conjunto de datos con el que validar las predicciones realizadas a partir de las distintas técnicas (Figura 3.21).

Figura 3.21 Número total de operaciones de transporte público y privado con destino la Comunidad Autónoma de Aragón. (Período 93-96)

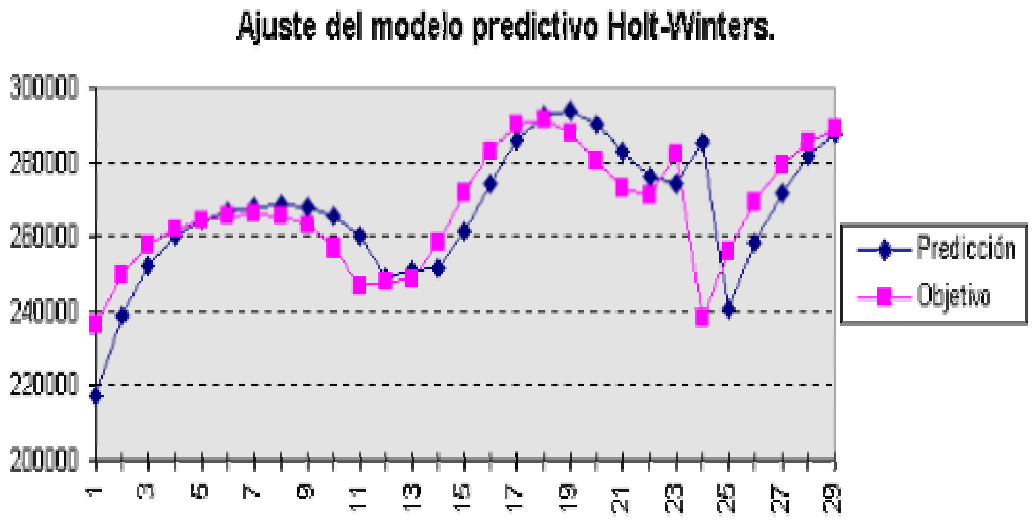
Aplicación del método estadístico tradicional. Una observación detenida de los datos, muestra un claro comportamiento estacional, así como una cierta tendencia al alza en los mismos. Puesto que el efecto estacional aumenta con el valor medio de los mismos, nos encontramos ante una estacionalidad del tipo multiplicativa. Para este tipo de series el método más idóneo, por precisión y sencillez es el método **Holt-Winters.** Es un método de alisado exponencial, con tres parámetros a estimar. Para su construcción hemos utilizado los 42 primeros datos, de forma que los parámetros se ajustan para minimizar el error entre los valores reales y la predicción suministrada. El modelo construido queda de la siguiente forma (Tabla 1):

Tabla 3.2. Estimación de los parámetros del modelo Holt-Winters

Parámetro	a nivel medio	g tendencia	d estacionalidad.
Estimación	0.999	0.0001	0.999

Los resultados obtenidos son los que se muestran en la Figura 3.22.

Figura.3.22 Ajuste obtenido en la estimación del modelo predictivo. (Método Holt-Winters).



Aplicación de la red Backpropagation. La red neuronal más adecuada en este caso es el **perceptrón multicapa** con aprendizaje por retropropagación de error (Backpropagation). Al ser nuestro problema una serie temporal, las entradas serán

los datos correspondientes a los meses anteriores y las salidas la predicción para los siguientes.

Tras probar otras topologías consideraremos una red de 12 entradas y 1 salida. El número óptimo de capas ocultas se obtiene tras ensayar la red para distinto número de capas y neuronas por capa (Tabla 3.3).

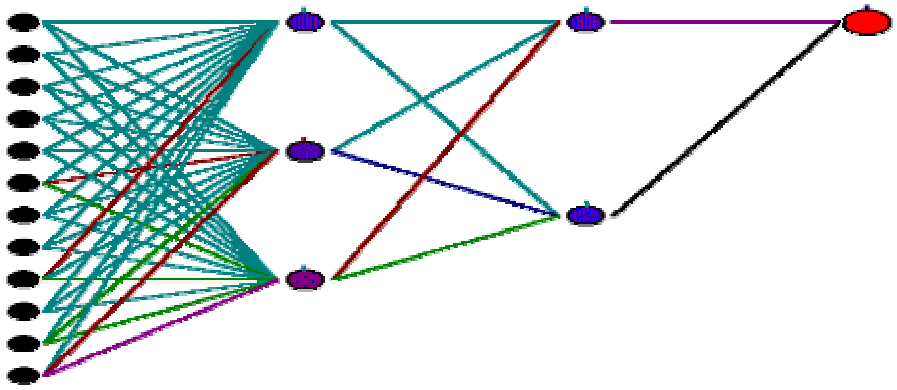
Para la realización de las pruebas se ha entrenado la red con distintos números de patrones de aprendizaje, y de test, finalizándose el entrenamiento cuando el error de test alcanza su valor mínimo, para garantizar que la red no "memoriza" los datos de entrada, sino que conserva su capacidad de generalización.

Tabla 3.3 Errores cuadráticos medios de test para las distintas topologías (con 12 entradas).

Topología	Error de test
3-0	0,55
5-0	0,49
7-0	0,58
9-0	0,6
11-0	0,55
3-2	0,375
5-3	0,484
7-5	0,61
9-7	0,465
11-9	0,485

El resultado de las pruebas nos da la óptima topología para una red de 12 entradas, 1 de salida, y dos capas ocultas con 3 y 2 neuronas respectivamente (Figura 3.23).

Figura 3.23 Topología seleccionada**.**



Para el entrenamiento, prescindiremos de los 6 últimos datos que será con los que comprobemos la capacidad de predicción de la red.

Número de patrones = 42-(12+1-1) = 30 patrones.

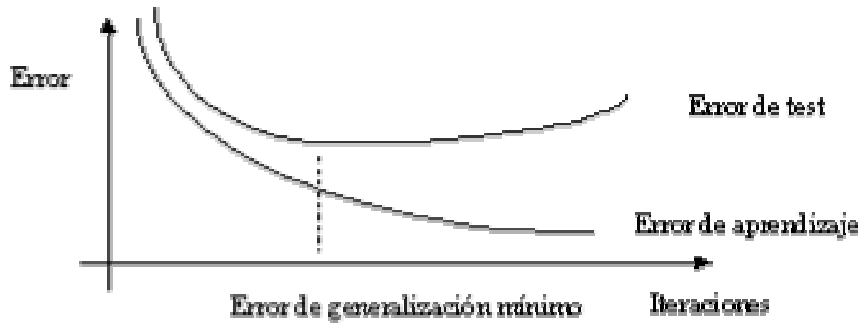
El primer paso es preprocesar los datos para adecuarlos a su tratamiento por la red neuronal. En nuestro caso realizaremos un estandarizado, que transforma las variables de forma que presenten media cero y varianza unidad. De esta forma se consiguen rangos de valores parecidos para todas las entradas y acotados cerca del intervalo de trabajo de la función de activación de la red (que tomamos del tipo tangente hiperbólica, entre -1 y 1).

Para el entrenamiento de la red, dividimos el conjunto de patrones en patrones de **aprendizaje** y patrones de **test.** Con los primeros la red ajusta sus pesos internos de forma que el error entre la salida obtenida y la esperada sea mínimo. Con los segundos controlamos la capacidad de generalización de la red, frente a datos con los que no ha sido entrenada. Un bajo error de aprendizaje indica que la red

responde bien frente a las entradas con las que se le ha entrenado. No obstante si es muy bajo, puede que la red haya "memorizado" los datos de partida, perdiendo capacidad de generalización frente a otras entradas. Un bajo error de test, por contra asegura que la red ha aprendido el "*mapping"* subyacente en los datos, y responde adecuadamente frente a series de datos con los que no ha sido entrenada.

Consideraremos 24 patrones para el aprendizaje y 6 para el test. Se entrena la red con los patrones de aprendizaje, y se va comprobando su capacidad de generalización con los patrones de test, cesando el aprendizaje cuando el error de test es mínimo, para evitar el sobre ajuste de la red (Figura 3.24.).

Figura 3.24 Variación del error durante el aprendizaje

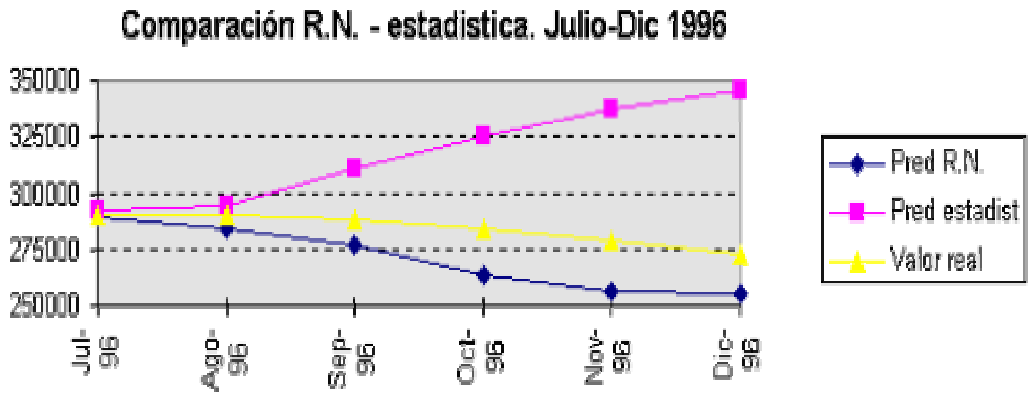


Tras ensayar con distintos pesos iniciales, la configuración óptima proporciona los siguientes resultados.

Resultados.

Comparando los resultados obtenidos a partir de las dos técnicas (Fig.7 y Tabla 3), se observan errores de un orden de magnitud superiores en el caso del método estadístico, además de la necesidad de realizar un estudio de la naturaleza de la serie para seleccionar el método estadístico.

Figura 3.25 Comparación de resultados



.Tabla 3.4Comparación de errores cuadráticos medios.

	RMS
Red neuronal	1443664763
Mét. Estadístico	11167000000

CONCLUSIONES

La red neuronal permite obtener mejores soluciones sin necesidad de conocer en profundidad la naturaleza del problema, aspecto muy común en el estudio del transporte donde las variables dependen de multitud de factores a veces muy difícilmente evaluables (situación económica local, nacional, mundial, de los distintos

sectores, huelgas,…). Por otro lado, los métodos estadísticos pueden resultar más idóneos en situaciones donde se conozca con precisión el problema. Sin embargo, el entrenamiento y la predicción de la red neuronal son procesos independientes, por lo que una vez entrenada, su uso como herramienta predictiva resulta muy sencillo para los usuarios (transportistas, empresas del sector…).

3.5 CONTROL DE SERVO SISTEMAS NO LINEALES USANDO REDES NEURONALES

Luis A. Ponce Dioses, Arturo Rojas Moreno, Luis G. Herrera Bendezú

Universidad Nacional de Ingeniería

Facultad de Ingeniería Eléctrica y Electrónica

RESUMEN

En este trabajo se plantea el problema de identificación y seguimiento de trayectoria de sistemas no lineales empleando redes neuronales. Los resultados teóricos se verifican mediante un estudio experimental en el cual el sistema no lineal está conformado por un servomotor DC de imán permanente con reducción que además tiene una varilla acoplada al eje del motor, a manera de un Brazo Robótico de un Grado de Libertad (BR1L), en donde la varilla es capaz de moverse en un plano perpendicular al eje del motor y es controlada por la tensión de armadura aplicada al servomotor.

INTRODUCCIÓN

En este artículo se plantea el problema de identificación y seguimiento de trayectoria de sistemas no lineales empleando redes neuronales. La solución al problema propuesto se ha dividido en dos etapas. La primera etapa consiste en la identificación de los elementos no lineales del BR1L que son la fricción estática y la fricción de Coulomb, y la componente sinusoidal debido a la presencia de la varilla, que además es función de la posición angular del eje del servomotor. Estas dos no linealidades son identificadas por dos redes neuronales multicapa de alimentación directa, entrenadas mediante el algoritmo de retropropagación (backpropagation). El objetivo de la identificación es linealizar el sistema BR1L utilizando una compensación por realimentación con la red neuronal. El sistema utilizado puede ser considerado como un sistema de primer orden si se desprecia el efecto de la inductancia de la armadura. Esta consideración se utiliza con el fin de tener acceso a los elementos no lineales del sistema BR1L, mediante una simple traslación de los mismos hacia la tensión de entrada del servomotor, pudiendo éstos eliminarse mediante una realimentación de compensación.

La segunda etapa consiste en la realización del control. Una vez linealizado el sistema, podemos diseñar cualquier tipo de control clásico. El controlador utilizado en este trabajo fue un PID.

El sistema experimental BR1L esta constituido, además del servomotor, por un sensor óptico de posición, un decodificador de cuadratura diseñado en un PLD (Dispositivo Lógico Programable), que también integra contadores y la interfaz a la

PC para medir la posición absoluta del BR1L. El sistema es controlado mediante un driver de potencia PWM (Pulse Width Modulation) con configuración tipo H. Para sintetizar al sistema se utilizó una PC con un procesador Pentium de 100 Mhz y una tarjeta de adquisición de datos. El Software se desarrolló en lenguaje C y consistió de los siguientes módulos: (1) identificación y control utilizando redes neuronales; (2) algoritmo de entrenamiento de la red neuronal (retropropagación); (3) visualización en tiempo real de los parámetros identificados.

En la sección 3.5.1 de este artículo se describen las redes neuronales utilizadas. La sección 3.5.3 describe el sistema a controlar. En la sección3.5.4 se deduce el algoritmo de identificación de los parámetros lineales y no lineales del sistema, y en la sección V se muestran los resultados.

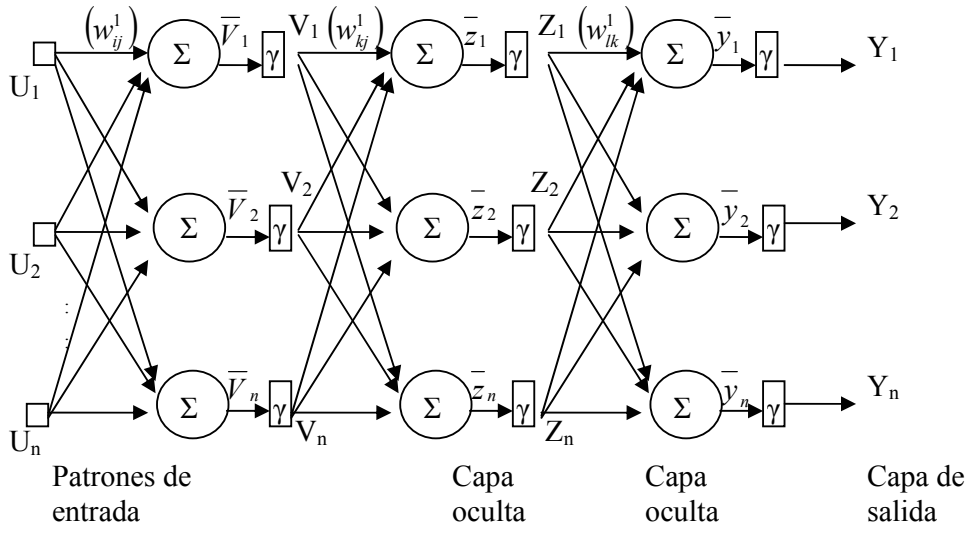
Redes neuronales. Para la identificación de los elementos no lineales del motor se puede usar una red multicapa, con una capa de entrada, una de salida y dos capas ocultas, tal como se muestra en la Fig. 3.26 En la salida de cada elemento de una capa se empleó la función no lineal sigmoidal.

$$\gamma = \frac{1 - e^{-x}}{1 + e^{-x}} \qquad 3.8$$

En la práctica, esta red se usa para la identificación de sistemas estáticos. Los pesos de la red son ajustados con el algoritmo de retropropagación[20], el cual consiste en minimizar una adecuada función de error **e** entre la salida **y** de la red y una salida deseada y_d.

[20] FREEMAN, James A. y SKAPURA, David M. Redes Neuronales. Algoritmos, aplicaciones y técnicas de programación. Addison Wesley Iberoamericana, S. A. 1993.

Figura 3.26 Red neuronal de tres capas

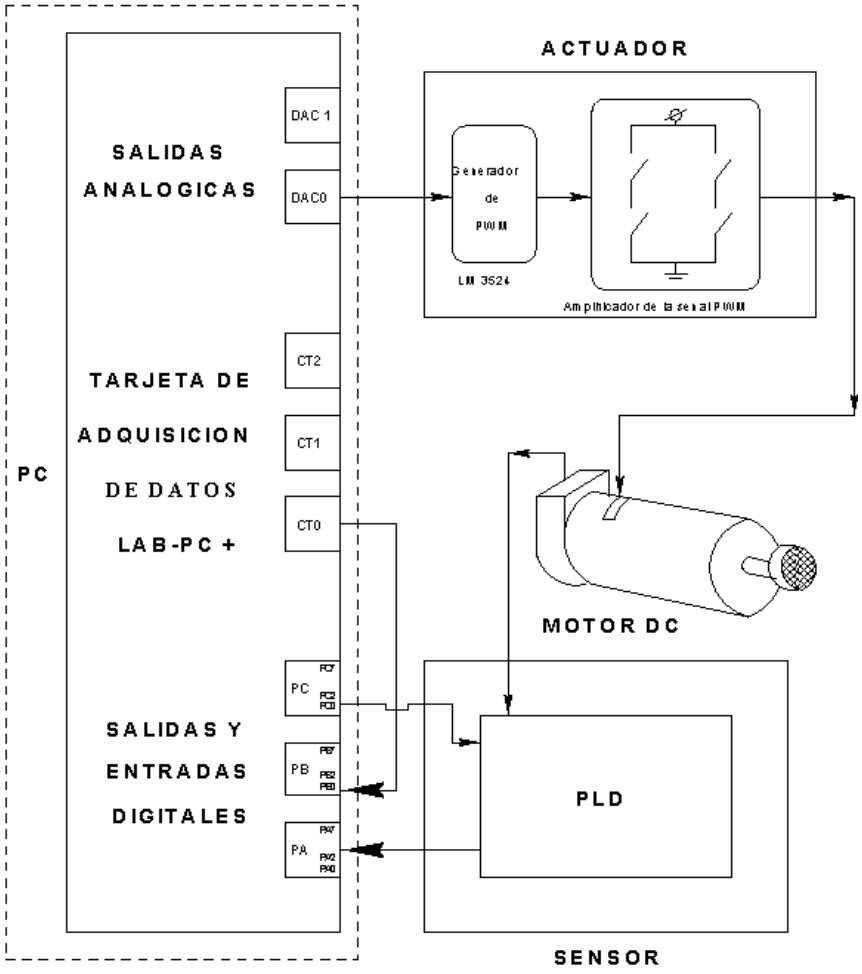


Las rutinas de entrenamiento de la red fueron programado en "C" . Las operaciones con matrices fueron implementadas en base a rutinas optimizadas.

Las redes neuronales utilizadas poseen un elemento en la capa de entrada, 20 elementos en la primera capa oculta, 10 elementos en la segunda capa oculta y 1 elemento en la capa de salida.

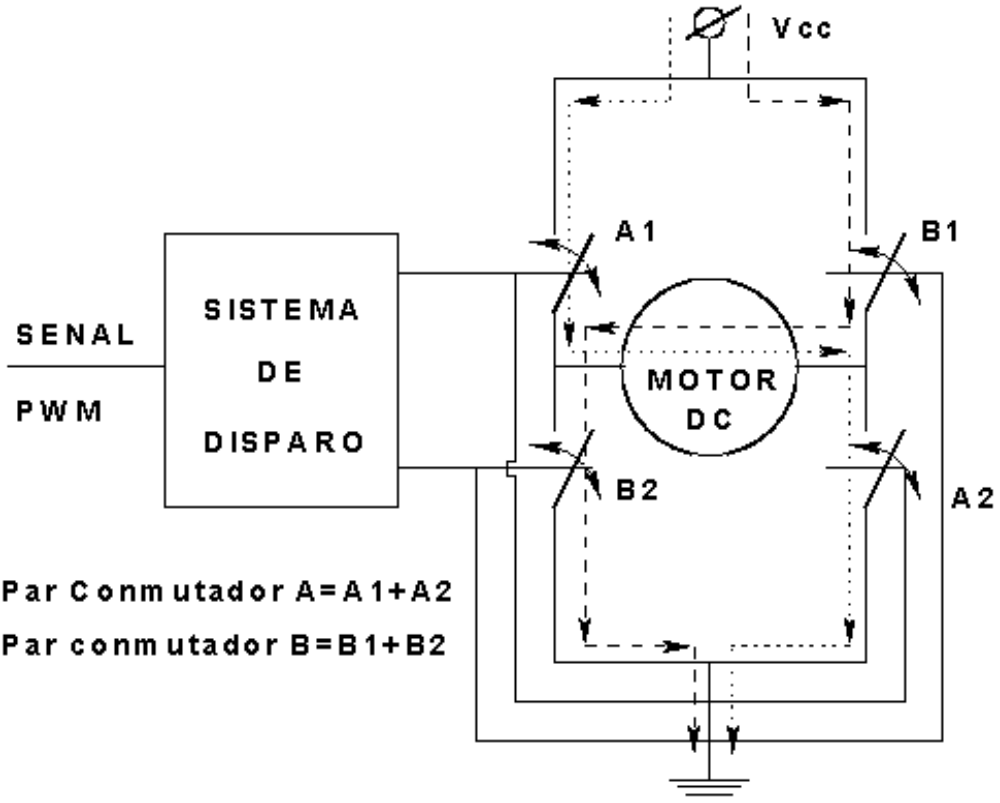
Descripción del sistema a controlar. En esta sección se presenta una descripción del sistema implementado, el cual se muestra en la figura 3.27. También se realiza el modelamiento matemático de la planta. Para el control del BR1L se empleó un actuador que consiste de un servomotor DC de campo magnético permanente con escobillas conmutadas.

Figura 3.27 Esquema general del sistema implementado



Para accionar al actuador se construyó un driver PWM, cuya etapa de potencia consiste de 4 conmutadores en configuración H. El esquema general del sistema de disparo y los conmutadores se muestran en la figura 3.28. Cuando el sistema de disparo cierra el conmutador A y abre el B, el sentido de la corriente sigue la línea de puntos, induciendo de esta forma una tensión $^{+V_{cc}}$en el motor. Si el sistema de disparo abre el conmutador A y cierra el B, el sentido de la corriente sigue la línea segmentada, induciendo así una tensión $^{-V_{cc}}$en el motor. Por consiguiente, el motor ve en sus bornes una onda de voltaje cuadrada, variando entre $^{\pm V_{cc}}$, y la corriente que pueda absorber dependerá de los conmutadores.

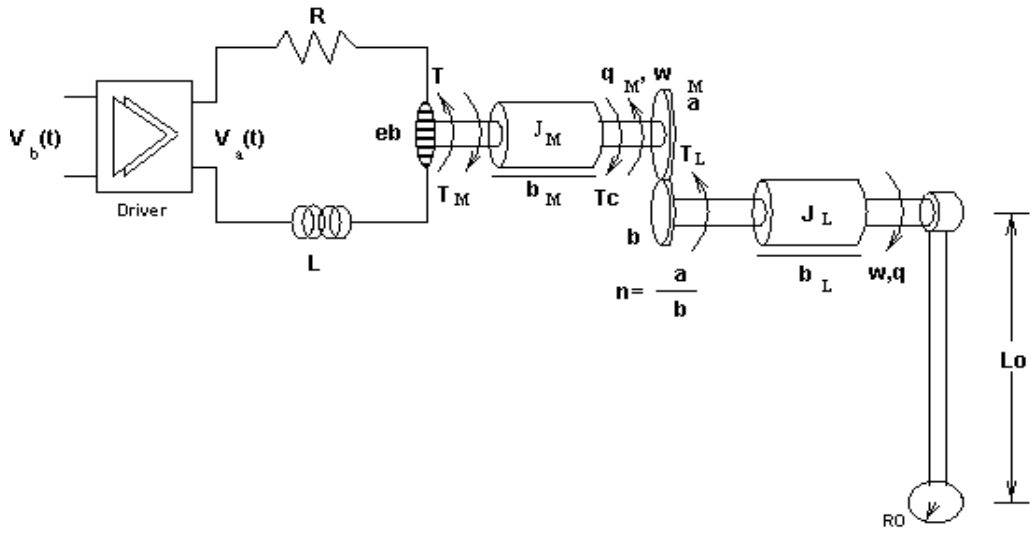
Figura 3.28 Esquema general del driver implementado



Para realizar el sensado de posición y sentido de giro del motor se usó un codificador óptico montado en el motor. La información sensada se envía, a través de dos salidas seriales de tren de pulsos desfasados 90 grados, a un PLD EPM7128E de la familia MAX 7000, el cual tiene integrado un circuito sensor de posición que consiste de un LS7083 y 4 contadores de la serie 74193. Una tarjeta LAB-PC+ se usó para la adquisición de datos y manipuleo de la señal de control.

En la figura 3.29 se muestra un esquema general del BR1L. En este sistema se observan dos partes importantes: (1) un sistema mecánico y (2) un sistema eléctrico.

Figura 3.29 Esquema de la planta a controlar



J_M, J_L	Momento de inercia del motor y de la carga.
b_M, b_L	Coeficiente de fricción viscosa del motor y la carga.
K_{AC}	Ganancia del actuador.
m, L_0	Masa y longitud de la varilla

$f(\bullet)$	Modela el efecto de la fricción estática y de Coulomb.
$\pm V_a$	Tensión de entrada en los bornes del motor.
R, L	Resistencia e inductancia de armadura
e_b	Fuerza contra - electromotriz.
K	Constante del par motriz.
q	Posición de la varilla
w	Velocidad angular de la varilla
n = a / b	Relación de engranajes o velocidades

Considerando como positivo el movimiento de la varilla en sentido antihorario y despreciando la inductancia del motor (L=0), la ecuación diferencial que modela este sistema es la siguiente:

$$[J_{eff} + \frac{mL_0^2}{3n}]\dot{w} = \frac{V_b K K_{act}}{R} - [b_{eff} + \frac{nEK}{R}]w - \\ - \frac{gL_0 m}{2n}\sin(q) - f(nw) \cdots (1)$$

$$J_{eff} = nJ_M + \frac{J_L}{n} \quad y \quad b_{eff} = nb_M + \frac{b_L}{n} \qquad 3.8$$

Identificación de parámetros. En esta sección se realiza la deducción matemática del algoritmo de identificación de los parámetros lineales y no lineales de motor. Primero consideraremos el caso de un motor sin carga usado para identificar la fricción estática y la de Coulomb usando una red neuronal NNT. Esta red identificada servirá como dato para la identificación de los parámetros de un motor con carga.

El problema de identificación consiste en encontrar un conveniente modelo de identificación para ajustar los parámetros del modelo vía la optimización de una función de costo basado en el error entre la planta y la salida del modelo de identificación.

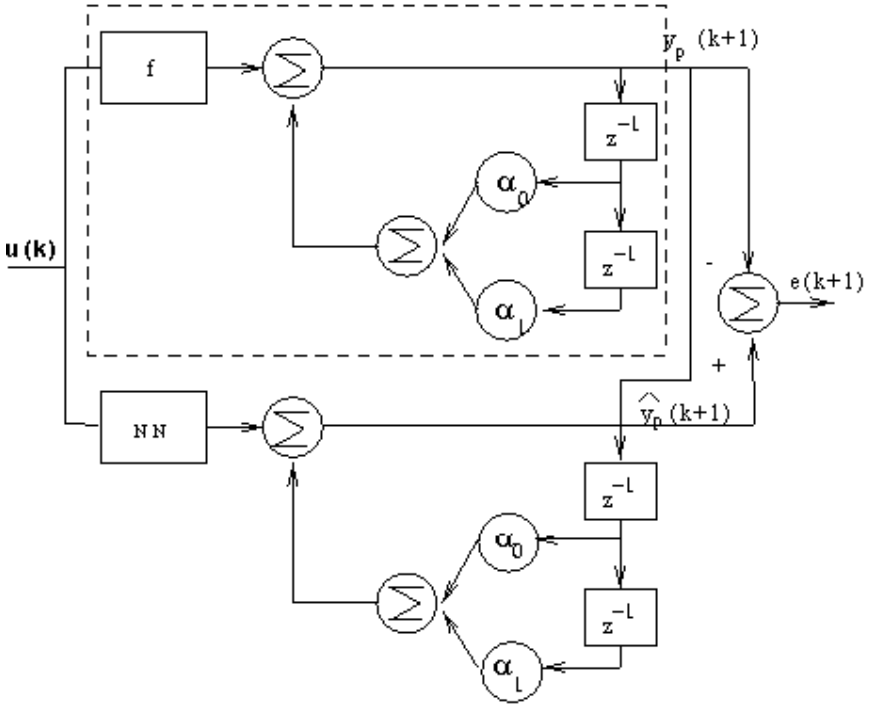
Se muestran dos modelos de identificación[21]: el paralelo y el serie-paralelo. El modelo serie-paralelo se ilustra en la figura 3.30 y se diferencia de la configuración en paralelo en que la salida de la planta es realimentada al modelo de identificación. Para la figura 3.30, el modelo de identificación tiene la forma:

$$\hat{y}_p(k+1) = \hat{a}_0 y_p(k) + a_1 y_p(k-1) + NN[u(k)] \qquad 3.10$$

El modelo serie - paralelo tiene varias ventajas sobre el modelo paralelo. Desde que la planta se asume que es estable y con salida y entrada acotada, entonces todas las señales usadas en el procedimiento de identificación (por ejemplo, entradas a la red neuronal) serán acotadas. Además, desde que no existe un lazo de realimentación en el modelo, puede usarse el algoritmo de entrenamiento retropropagación estática para ajustar los parámetros de la red, reduciendo así substancialmente los cálculos computacionales. Sin embargo, si se asume que el valor de salida tiende asintoticamente a pequeños valores, el modelo serie - paralelo puede ser reemplazado por el paralelo.

Figura 3.30 Modelo de identificación serie-paralelo

[21] NARENDRA, Kumpati S. y PARTHASARATHY Kannan. Identification and Control of Dynamical Systems Using Neural Networks, IEEE Transactions on Neural Networks. Vol. I. No. 1, March 1990.



Identificación de los parámetros lineales y no lineales del motor sin carga. Asumiendo despreciable la inductancia del motor y considerando la masa de la varilla igual a cero para el motor sin carga, la ecuación diferencial de este sistema es la siguiente:

$$J_{eff}\,\dot{\omega} = \frac{V_b K K_{act}}{R} - \left(b_{eff} + \frac{nEK}{R}\right)\omega - f(n \qquad 3.11$$

Discretizando (3.11) se obtiene la ecuación:

$$\omega_{k+1} = \left[1 - \frac{T_S}{J_{eff}}\left(b_{eff} + \frac{nEK}{R}\right)\right]\omega_k + \\ + \frac{KK_{act}T_S}{RJ_{eff}}\left[V_{b\,k} - \frac{Rf(n\,\omega_k)}{KK_{act}}\right] \qquad 3.12$$

Donde f(nw_k) se modela como C Sgn(nw_k) para las simulaciones. La ecuación (3.12) se puede escribir en una versión simplificada, tal como se muestra en la ecuación (3.13).

$$\omega_{k+1} = G_1\omega_k + G_2\{y_{b_k} + G_3 Sgn(n\omega_k \qquad 3.13$$

Ahora asumiremos que no se conocen los parámetros lineales G_1 y G_2 del modelo matemático del motor. Entonces definiremos $\hat{G}_{1k}$, $\hat{G}_{2k}$ y $\hat{\omega}_k$ como los parámetros lineales y la velocidad angular respectivamente, estimados en el k-ésimo instante de tiempo, los cuales se encuentran relacionados por la siguiente ecuación:

$$\hat{\omega}_{k+1} = \hat{G}_{1_k}\omega_k + \hat{G}_{2_k}\{V_{b_k} + NNT(n\omega_k \qquad 3.14$$

Donde NNT(nw_k) es la salida de la red neuronal para una entrada discreta nw_k. La función de NNT es identificar a $\frac{Rf(nwk}{KK_{act}}$. Si el error en el instante **k** se define como la diferencia entre la velocidad angular real y la estimada:

$$e_k = \hat{\omega}_k - \omega_k \qquad 3.15$$

Entonces el objetivo de identificación se reduce a minimizar el error e_k utilizando algún método de optimización. El método seleccionado, por su simplicidad, fue el ***algoritmo del gradiente***. Este algoritmo minimiza la función de costo $J = \frac{e_k^2}{2}$ en funció de los parámetros estimados:

$$\Theta_{k+1} = \begin{bmatrix} G_{1_{k+1}} \\ G_{2_{k+1}} \end{bmatrix} \qquad 3.16$$

Donde $\Theta_k = \left[\hat{G}_{1k}\ \hat{G}_{2k}\right]^T$ es el vector de los parámetros identificados. La ley de identificación del vector Θ_k es de forma tal que la función de costo se minimiza en cada k-ésimo instante de tiempo, de modo que

$$\begin{bmatrix}\hat{G}_{1k}\\ \hat{G}_{2k}\end{bmatrix} = \begin{bmatrix}\hat{G}_{1k-1}\\ \hat{G}_{2k-1}\end{bmatrix} - \begin{bmatrix}g_1 & 0\\ 0 & g_2\end{bmatrix} \frac{\partial \mathcal{J}}{\partial \Theta_{k-1}} \qquad 3.17$$

Donde $\begin{bmatrix}g_1 & 0\\ 0 & g_2\end{bmatrix}$ es la matriz de velocidad de identificación y $g_1, g_2 \rangle 0$ son los parámetros de velocidad de identificación que permiten variar la velocidad de convergencia. Si:

$$\frac{w}{\partial \Theta_{k-1}} = \frac{w}{\partial e_k}\frac{\partial e_k}{\partial \Theta_{k-1}} = e_k \frac{\partial e_k}{\partial \Theta_{k-1}} = -e_k\left[V_{bk-1} + \overset{wk-1}{NNT}(nw_{k-1})\right] \quad 3.18$$

se obtiene entonces la siguiente ley de identificación discreta:

$$\begin{bmatrix}\hat{G}_{1k}\\ \hat{G}_{2k}\end{bmatrix} = \begin{bmatrix}\hat{G}_{1k-1}\\ \hat{G}_{2k-1}\end{bmatrix} + \begin{bmatrix}g_1 & 0\\ 0 & g_2\end{bmatrix} \cdot \begin{bmatrix}\omega_{k-1}\\ V_{bk-1} + NNT(n\omega_{k-1})\end{bmatrix} \epsilon \quad 3.19$$

Identificación de los parámetros lineales y no lineales del motor con carga. El modelo matemático de este sistema se muestra en la ecuación (3.19) y la discretización de este modelo se presenta en la ecuación (3.20):

$$\omega_{k+1} = \left[1 - \frac{T_S\left(b_{eff} + \frac{nEK}{R}\right)}{J_{eff} + \frac{mL_0^{\ 2}}{3n}}\right]\omega_k + \frac{T_S KK_{act}}{R\left[J_{eff} + \frac{mL_0^{\ 2}}{3n}\right]} \cdot \left[V_{bk} - \frac{RgL_0 m}{2nKK_{act}}\sin(q_k) - \frac{RC}{KK_{act}}Sgn(n\omega_k)\right] \qquad 3.20$$

Para la identificación de los parámetros lineales se debe deducir un ley de identificación para G_1 y G_2 que es similar a la que se hizo en la sección IV-A. Reescribiendo la ecuación (3.21) como:

$$\omega_{k+1} = G_1 \omega_k + G_2 \left\{ V_{\delta k} + G_4 sin(q_k) + G_3 f(n\omega) \right. \qquad 3.21$$

se observa que existen dos no linealidades: $G_4 sin(q_k)$ y $J = \frac{e_k^2}{2}$, de las cuales la segunda ha sido identificada en la sección anterior por la red NNT. Ahora se debe encontrar una red NNT1 que identifique la primera no linealidad. La velocidad angular estimada $\hat{\omega}_{k+1}$ se puede escribir como sigue:

$$\hat{\omega}_{k+1} = \hat{G}_{1k} \omega_k + \hat{G}_{2k} \left\{ V_{\delta k} + NNT1(q_k) + NNT(n\omega_k) \right. \qquad 3.22$$

Si comparamos las ecuaciones (3.21) y (3.22) con (1.13) y (1.14) respectivamente, se deduce fácilmente la siguiente ley de identificación:

$$\begin{bmatrix} \hat{G}_{1k} \\ \hat{G}_{2k} \end{bmatrix} = \begin{bmatrix} \hat{G}_{1k-1} \\ \hat{G}_{2k-1} \end{bmatrix} + \begin{bmatrix} g_1 & 0 \\ 0 & g_2 \end{bmatrix} \cdot \begin{bmatrix} \omega_{k-1} \\ V_{\delta k-1} + NNT1(q_{k-1}) + NNT(n\omega_{k-1}) \end{bmatrix} \qquad 3.22$$

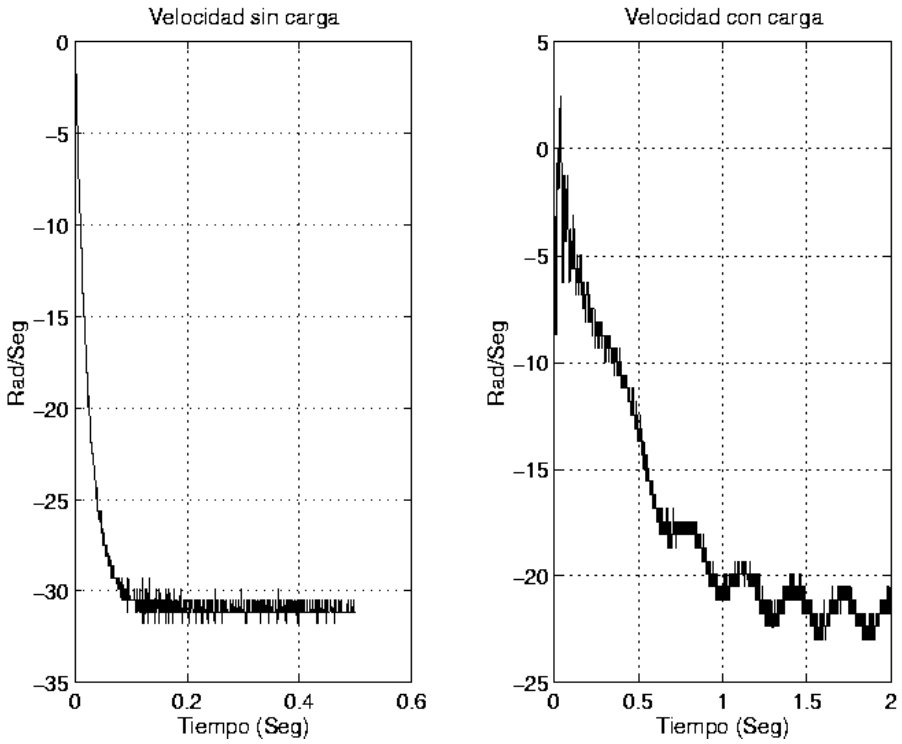
Resultados experimentales. En esta sección se presentan los resultados experimentales obtenidos, los cuales son comparados con modelos matemáticos deducidos.

Para entrenar a la red neuronal hay que excitar el sistema con una señal que realice un mapeo total. Es decir, se debe elegir un conjunto de entrada **I** que produzca un conjunto de salidas **S**, de tal forma que **I** y **S** se distribuyan uniformemente en el

conjunto de entradas y salidas del sistema, respectivamente. Para nuestro sistema, la tensión de entrada se encuentra entre [-1.53, 1.37] Volt. Se realizaron experiencias con tres tipos de señales: (1) escalones con amplitudes variando dentro del rango de entrada y tiempo de duración variable para cada escalón; (2) escalones con amplitudes variando dentro del rango de entrada y tiempo de duración fijo; (3) sinusoides con amplitudes variables y periodo fijo. Para el entrenamiento se eligió el segundo tipo de señal; el criterio de elección fue la velocidad de aprendizaje de la red y el periodo de las señales de entrenamiento se eligió por el método de prueba y error, teniendo como valor inicial la frecuencia de resonancia del BR1L, linealizado alrededor de q=0.

Se eligió 30 valores fijos de tensiones uniformemente distribuidos para el conjunto **I**, los cuales produjeron un conjunto de velocidades $S \in [-37,37]\, Rad\ s^{-1}$ que es el rango de salida. Por consiguiente, la señal de excitación resulta un tren de escalones cuyas amplitudes se encuentran aleatoriamente distribuidas en cada conjunto **I**. Para la identificación de $J = \frac{e_k^2}{2}$, cada escalón tienen una duración de 0.5 seg. y para $G_4 sin(q_k)$ los escalones tiene una duración de 2 seg., produciendo una velocidad que incluye un estado transitorio y uno estacionario, tal como se muestra en la figura 3.31.

Figura 3.31 La primera figura muestra la velocidad de salida sin carga en el motor y la segunda con carga.



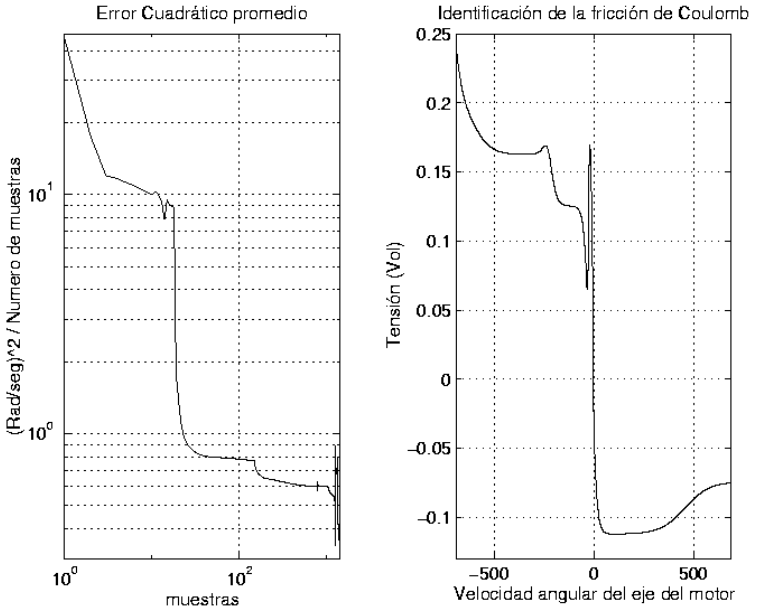
Para comprobar el aprendizaje de la red, se excitó al sistema, sin actualizar los pesos de la red, con un tren de escalones con amplitudes distribuidas en un conjunto de entrada **I** (distinto al de entrenamiento. Podemos definir un error cuadrático promedio por fase como:

$$E_f = \frac{\sum_{k=1}^{N} e_k^{\,2}}{N} \qquad 3.24$$

Donde N es el número de muestras por fase y e_k es el error por muestra.

Identificación de la fricción Estática y de Coulomb. Utilizando la ley de identificación de la ecuación (3.19) se obtuvieron los resultados que a continuación se detallan.

Figura 3.32 La primera figura muestra el error cuadrático promedio durante el entrenamiento y la segunda la fricción de Coulumb identificada.



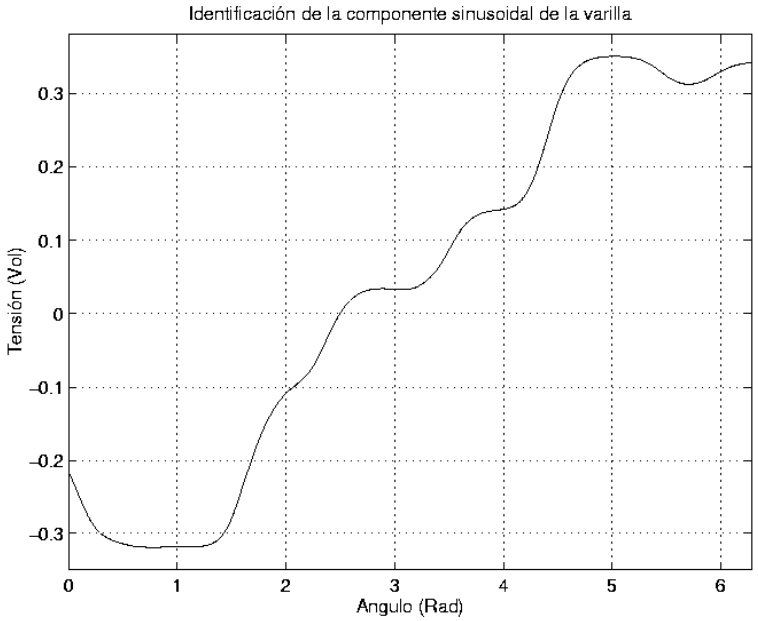
En la Figura 3.32 se observa que el error cuadrático por muestra se reduce hasta un valor de 0.3, en donde se ha tomado en cuenta un ruido de cuantificación cuando se calculó la velocidad a partir de la posición (ver Fig. 3.32). El modelo de identificación tiene como salida el promedio de la velocidad a identificar. En la Fig. 3.32 se muestra también la identificación de la fricción de Coulomb; es claro observar que el torque de fricción tiene distintos valores para cada velocidad. También se muestra que la red es capaz de identificar el offset de la señal de entrada, observando la asimetría en la señal identificada.

Identificación de la componente sinusoidal, efecto de la varilla. Utilizando la ley de identificación de la ecuación (3.23) y la red neuronal NNT de la sección anterior, se realizó la identificación de los componentes lineales de la ecuación (3.22) y de la

red neuronal (NNT1) que identificará a la componente sinusoidal, debido al efecto de la dinámica de la varilla. Se probaron 3 posibles señales de entrada para entrenar a la red NNT1. Primero, una señal escalón de periodo 2 seg., cuyas amplitudes se encuentran distribuidas en dos conjuntos de entrada **I** mencionados en la sección V. Segundo, una señal sinusoidal, cuyas frecuencias, se encuentran distribuidas en un ancho de banda alrededor de la frecuencia de resonancia del sistema linealizado alrededor de $\theta = 0$. Se tomó el criterio de - 3dB para el cálculo de las frecuencias iniciales y finales de este ancho de banda. Las amplitudes se encuentran distribuidas en el primer conjunto. Tercero, se combinaron las señales anteriores, para generar una nueva señal de entrada. Esto se realizó intercalando el primer conjunto con el segundo. Con el primer conjunto de entrenamiento se obtuvo mejores resultados con respecto a la velocidad de convergencia en la identificación de los parámetros lineales y no lineales.

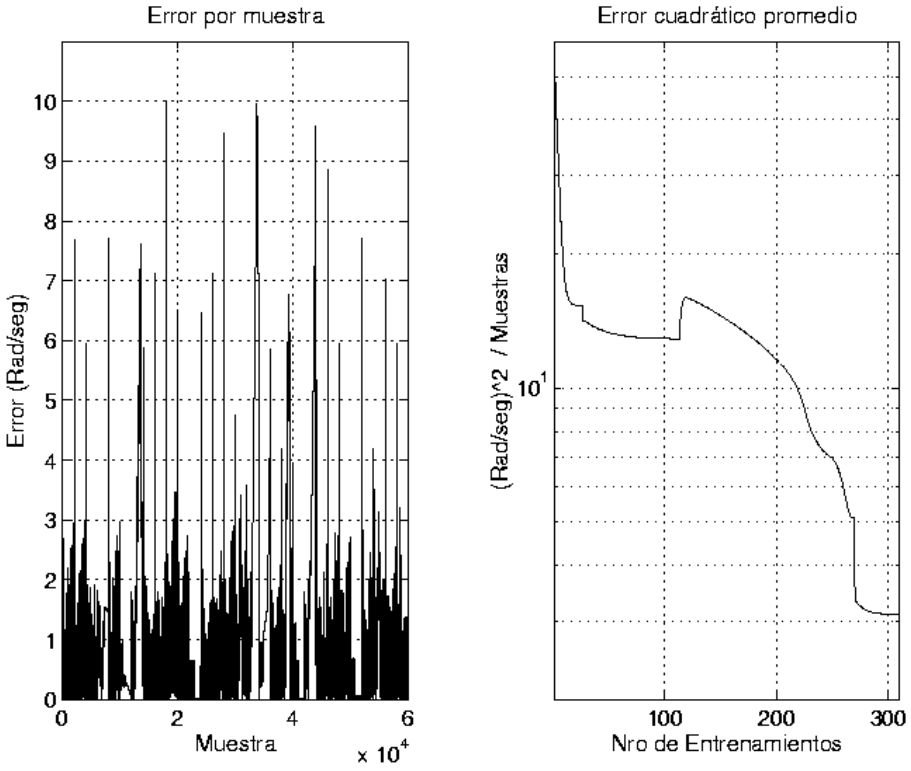
Otro punto importante que hay que tomar en cuenta es la señal de entrada a NNT1. El método de actualización de pesos empleado (retropropagación) sirve sólo para sistemas estáticos. En nuestro caso, a partir del modelo matemático del sistema se deduce fácilmente que la no linealidad a identificar $\left(\frac{RgL_0 m}{2nKK_{act}} \sin(q_k) \right)$ es una función estática en el rango de **[0.2 π]**. En forma práctica, la posición es derivada a partir de la velocidad del eje de la carga. Por tanto, se puede aprovechar la periodicidad de la función sinusoidal para acotar la posición del eje de la carga en el rango de **[0.2 π]**; es decir, limitarla. En la figura 8 se muestra la señal identificada por la red neuronal NNT1; la señal de entrada a esta red es la posición del eje de la carga limitada entre **[0.2 π]**. Esta red identificó la no linealidad producida por la varilla en el eje de la carga. En este gráfico se observa el efecto de NNT (red que identificó la fricción de Coulomb) sobre la identificación de NNT1. Aquí se observa qué tan sensible es la identificación de NNT1 ante disturbios externos.

Figura 3.33 Señal identificada por la red neuronal NNT1.



En el primer gráfico de la figura 3.33 se muestra el error por muestra al final del entrenamiento, que se realizó sin actualizar los pesos de la red. Aquí se observan unos picos de error de hasta 10 rad/seg; esto se debe a que la red ha promediado un ruido transitorio debido al cambio de posición de la varilla de una posición a otra. Este ruido transitorio se puede observar en los primeros segundos del segundo gráfico de la figura 3.32. En el segundo gráfico de la figura 3.33 se observa el error cuadrático promedio que se calcula después de cada fase de entrenamiento, sin actualizar los pesos de la red, utilizando la ecuación (3.24).

Figura 3.34 En la primera figura se muestra el error por muestra al final del entrenamiento. En la segunda figura, se observa el error cuadrático promedio durante el entrenamiento.

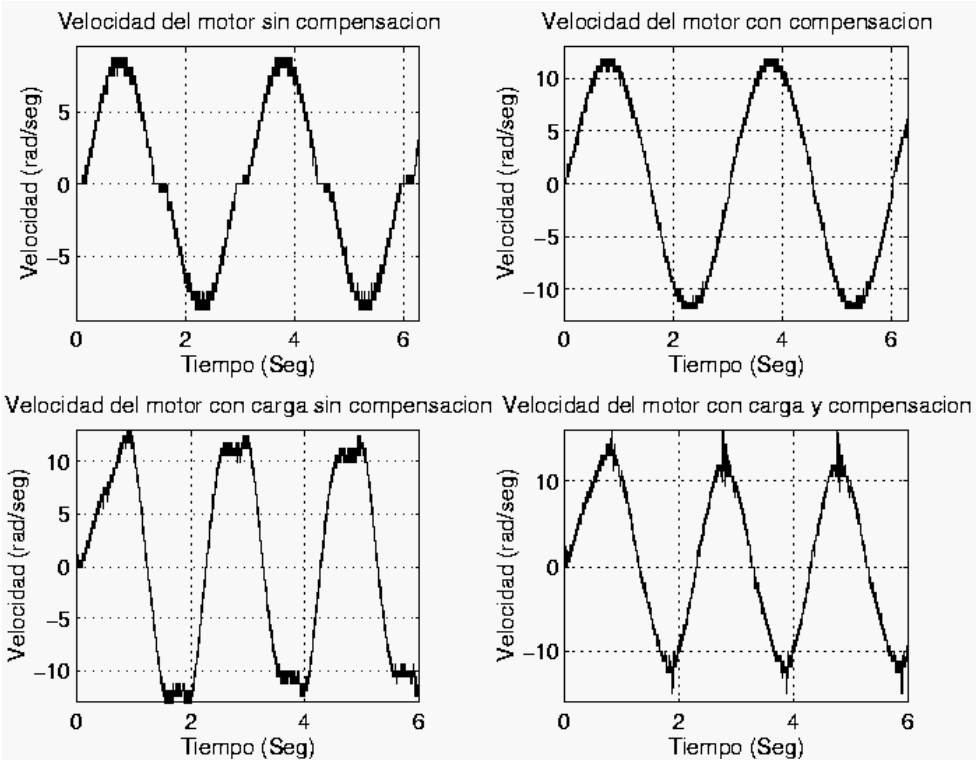


CONTROL PID. Es claro observar en las ecuaciones (3.13) y (3.21) que los torques no lineales $G_4 sin(q_k)$ y $J = \frac{e_k^2}{2}$ han sido trasladados hacia la tensión de entrada del sistema para tener acceso a ellos. Entonces, la eliminación de estas no linealidades puede ser realizada por una realimentación de compensación. Es decir, generando una tensión de igual magnitud que $J = \frac{e_k^2}{2}$ y $G_4 sin(q_k)$ pero de signo contrario, de tal manera que al adicionarlas a la tensión de entrada, elimine los torques de fricción no

lineal. Las redes neuronales NNT y NNT1 se utilizaron para realizar esta realimentación de compensación.

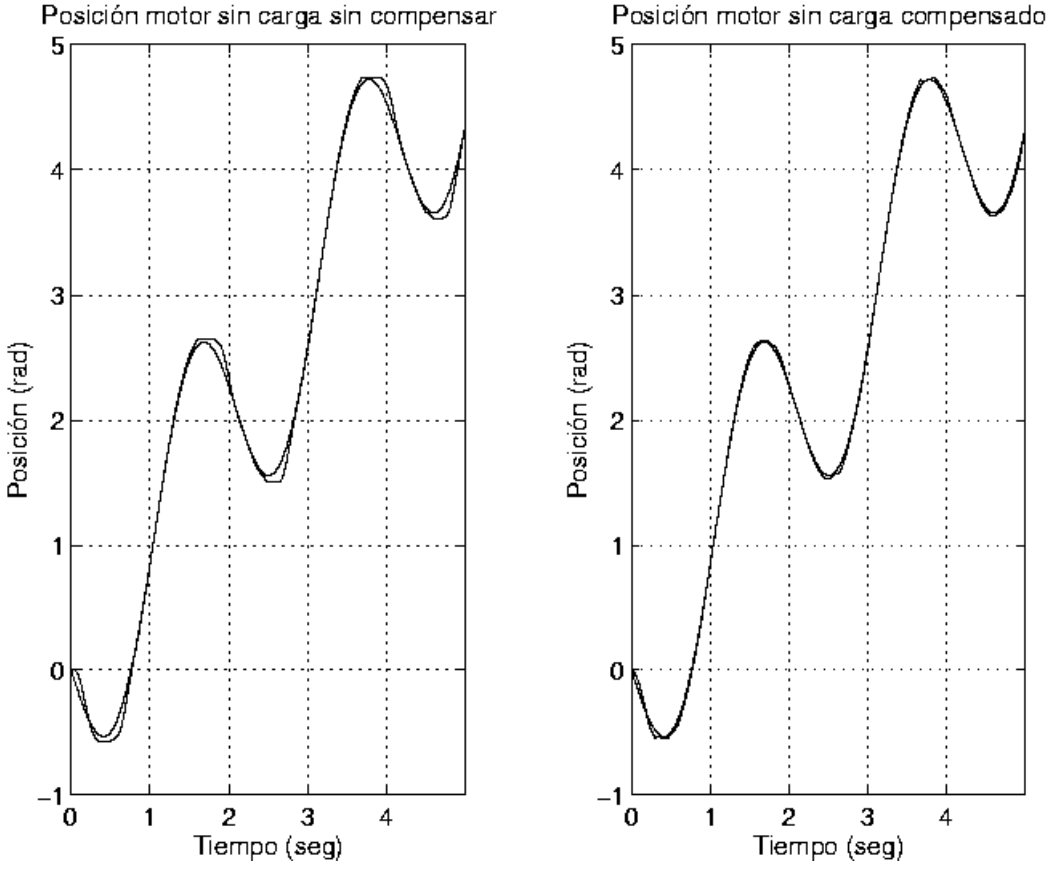
El primer gráfico de la figura 10 muestra la velocidad del motor sin carga para una entrada $0.5\sin\left(\frac{2\pi t}{3}\right)$ sin utilizar compensación; el segundo gráfico compensa esta no linealidad. Se observa que la velocidad aumenta y desaparece la zona muerta del motor. En el tercer gráfico de la figura 10 se muestra la velocidad del motor con carga para una entrada $sin(\pi t)$ sin compensar; en el cuarto gráfico se muestra el efecto de la compensación del sistema.

Figura 3.35 Compensación neuronal del motor con y sin carga



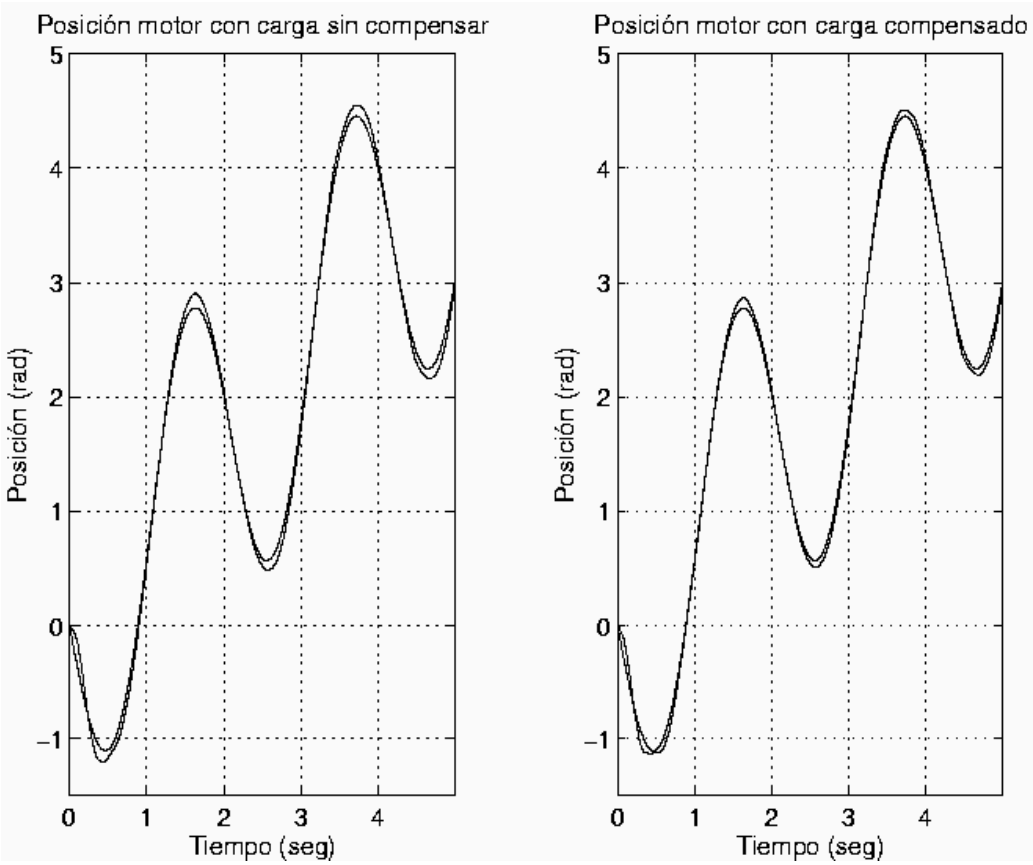
Control PID del motor sin carga. En el gráfico de la figura 3.35 se muestra la respuesta del motor sin carga. En el primer gráfico se observa el efecto de la zona muerta del motor y en el segundo se muestra su compensación. La trayectoria a seguir es $r = 0.8t - 3sin(3t)$ para el tiempo $0 \le t \le 5$. Los parámetros lineales convergieron a los valores $G_1 = 0.9604$ y $G_2 = 0.9546$.

Figura 3.36 Control PID de un motor sin carga.



Control PID del motor con carga. En el gráfico de la figura 12 se muestra la respuesta del control del motor con carga. En el primer gráfico se muestra la respuesta sin compensación y en el segundo gráfico se muestra su compensación. La trayectoria a seguir es $r = 0.8t - 2.5sin(3t)$ para el tiempo $0 \le t \le 5$. Para el control se sintonizaron los parámetros del controlador en ambas experiencias. Los parámetros lineales convergieron en valores $G_1 = 0.9973$ y $G_2 = 0.0416$.

Figura 3.37 Control PID de un motor con carga.

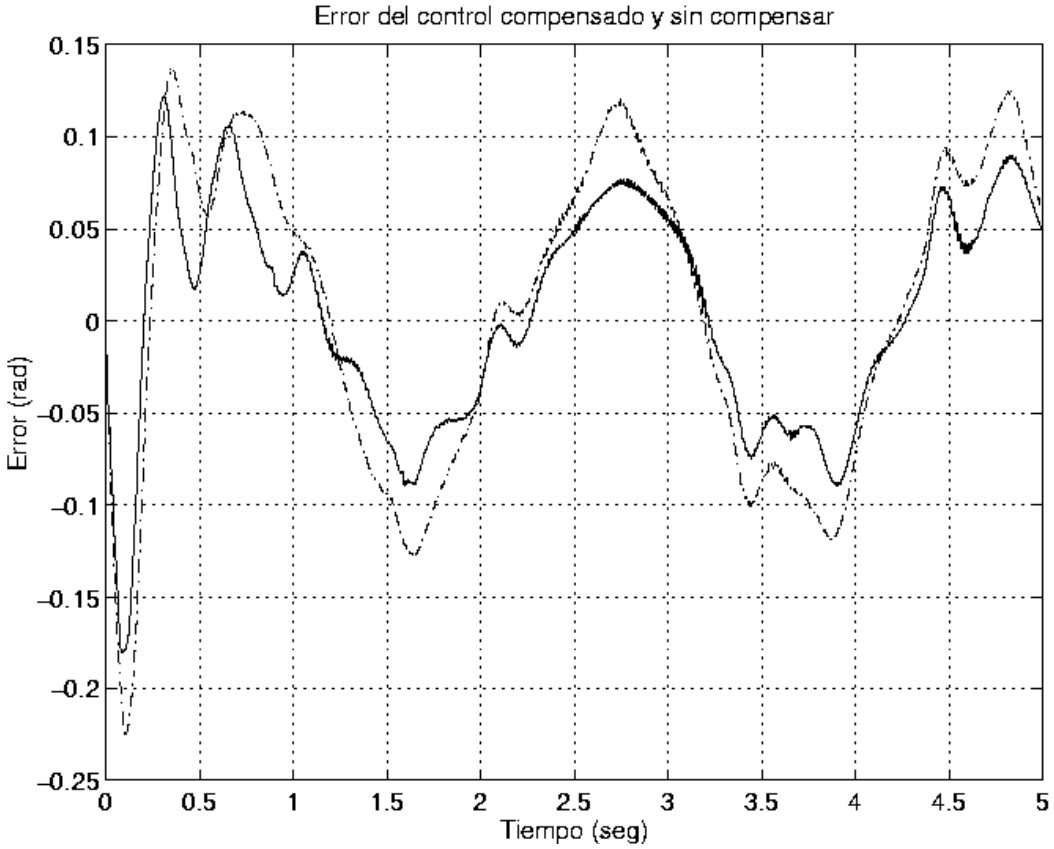


La compensación disminuyo el error cuadrático medio de seguimiento de la trayectoria en la mitad, tal como se observa en la tabla 3.5 En la figura 3.37 se muestran estos errores.

Tabla 3.5. Motor Error cuadrático.

Motor	Error cuadrático medio
Sin compensación	0.0071
Con compensación	0.0036

Figura 3.38 Error en el seguimiento de la trayectoria para un motor con carga.



La línea continua muestra el error utilizando compensación y la punteada sin compensación.

CONCLUSIONES

En este trabajo se ha demostrado que las redes neuronales son capaces de identificar elementos no lineales tales como la fricción de Coulomb y la componente sinusoidal debido a la varilla Se demostró también que la velocidad puede ser hallada a partir de la posición, a pesar del error de cuantificación (resolución) del sensor. Como hemos visto, la red neuronal promedia este error cuando se realiza la identificación. De los parámetros identificados, se verificó que la dinámica introducida por la inductancia de la armadura del motor se pudo despreciar en el diseño, lo cual redujo en uno el órden del modelo.

3.6 APLICACIÓN DE REDES NEURONALES ARTIFICIALES EN LA MODELIZACIÓN DEL TRATAMIENTO TÉRMICO DE ALIMENTOS.

Luera Peña, W. E.: Minim, L. A.

Resumen

En este trabajo se ha empleado la técnica de modelización por redes neuronales para predecir el perfil de temperatura en el centro de un producto acondicionado en latas durante el procesamiento térmico, como función de la temperatura del autoclave, temperatura inicial del producto y tiempo de proceso. Se construyó una red de tipo recurrente, compuesta de cinco nodos en la capa de entrada, diez nodos en la capa interna y un nodo en la capa de salida. Se uso el algoritmo "JE-backpropagation" para el proceso de aprendizaje. Los datos usados para la etapa de aprendizaje, validación y generalización fueron obtenidos a través de la implementación computacional de un modelo matemático resultante de la aplicación del balance diferencial de energía en envases cilíndricos conteniendo el alimento. Inicialmente, el proceso se analizó usándose datos de tiempo-temperatura con temperatura de autoclave constante.

Posteriormente, se emplearon datos de tiempo-temperatura con temperatura de autoclave variable, simulando de esta manera un proceso más realista. Para todas

las situaciones, la técnica de redes neuronales tuvo un buen desempeño en la predicción del perfil de temperatura para el centro del producto, indicando que es una alternativa viable para modelar procesos dinámicos.

El proceso térmico tradicional consiste en calentar el alimento dentro de su propio envase, usándose para ello un autoclave presurizada durante un periodo de tiempo y temperatura preestablecidos por ingenieros del proceso.

Tales parámetros son calculados tomando como base el conocimiento de las resistencias térmicas de los microorganismos y de los componentes nutricionales, de modo que sean obtenidos alimentos seguros desde el punto de vista de salud pública y estable a la temperatura de almacenamiento. Son dos las áreas básicas relacionadas con los cálculos de los procesos térmicos: la bacteriología, que engloba los conceptos de inactivación térmica de microorganismos y degradación de nutrientes; y la transferencia de calor, responsable de los factores que gobiernan el calentamiento del producto.

La modelización de la operación de esterilización de productos alimenticios en general tiene gran interés para estudios de optimización, diseño y operación de los equipos utilizados. Mediante el uso de estos modelos matemáticos en adecuados sistemas informáticos es posible conocer la evolución de la temperatura en el centro del producto evaluando así el tratamiento térmico aplicado y el efecto letal de éste. La visualización, automatización y control de procesos permite disponer en todo momento de información acerca de los parámetros de interés que afectan a la operación[22].

[22] Virseda, P.; Abril, Y.J. 1997. Simulación numérica en estado no estacionario de espárragos durante el proceso de esterilización. Alimentaría 3: 43-46

La modelización del procesamiento térmico de alimentos no es un proceso reciente. Varios modelos para transferencia de calor se desarrollaron por investigadores[23]. Los primeros modelos matemáticos para tratamiento térmico continuo de alimentos en partículas aparecen al inicio de los años 70 (Ruyter y Brunet, 1973; Manson y Cullen, 1974).

Posteriormente surgieron modelos para el análisis del coeficiente de transferencia de calor entre un líquido y una superficie sólida.

A pesar de que las técnicas de modelización matemática aplicadas al proceso de esterilización sean muy utilizadas, es necesario disponer de datos estables sobre las propiedades térmicas del producto, las cuales al no permanecer constantes durante el tratamiento térmico debido a la variación con la composición y la temperatura dificultan la modelización. En este contexto, el empleo de redes neuronales sería adecuado, toda vez que no son necesarias tal información, como variación de las propiedades físicas del alimento durante el proceso, pues tienen la capacidad de relacionar datos de entrada y salida del proceso sin conocimiento anterior de la relación entre ambos, a través de sucesivos entrenamientos. Puede afirmarse que las redes neuronales artificiales tienen la habilidad de aprender con los ejemplos.

En este trabajo se propone el uso de la técnica de redes neuronales para desarrollar un modelo matemático predictivo del procesamiento térmico de alimentos que permita determinar la temperatura del punto frío de un producto acondicionado en un recipiente metálico, a partir de la temperatura o presión del autoclave y del tiempo de proceso. Una vez que la temperatura del punto frío sea conocida, a partir de las condiciones del autoclave, la definición del proceso podrá ser realizada de forma “on-line”, y problemas debido a fallas en el proceso serán corregidos automáticamente.

---

[23] INCROPERA, P. F.; De Witt, D.P. 1992. Fundamentos de transferencia de calor e masa. Rio de Janeiro: Guanabara Koogan, 455 p.

Materiales y métodos. Modelo en base al principio de conservación de energía. La ecuación (3.25) representa el modelo de transferencia de calor por conducción unidimensional en estado no estacionario, sin generación de calor interna, expresada en coordenadas cilíndricas considerándose el sentido radial

$$\frac{\partial T}{\partial t} = \alpha\left[\frac{\partial T^2}{\partial r^2} + \frac{1}{r}\frac{\partial T}{\partial r}\right] \quad 3.25$$

Con las siguientes condiciones de contorno:

$$t = 0 \Rightarrow T = T_0, \forall r$$

$$t = 0 \Rightarrow \begin{cases} \dfrac{\partial T}{\partial r} = 0, \ r = 0 \\ \\ k\left(\dfrac{\partial T}{\partial r}\right) = h(T_\infty - T), \quad r = R \end{cases}$$

donde: T = temperatura; t = tiempo; α = difusividad térmica; r = posición radial del cilindro; k = conductividad térmica; h = coeficiente de transferencia de calor; R = radio del cilindro; T_∞ = temperatura en el interior del autoclave;

Donde: T = temperatura; t = tiempo; a = difusividad térmica; r = posición radial del cilindro; k = conductividad térmica; h = coeficiente de transferencia de calor; R = radio del cilindro; T8= temperatura en el interior del autoclave; la ecuación 3.25 puede ser expresada en la forma de diferencias finitas para su resolución numérica como:

$$T_i^{p+1} = T_i^p + \alpha\Delta t\left[\frac{T_{i+1}^p - 2T_i^p + T_{i-1}^p}{\Delta r^2}\right] + \alpha\Delta t\left[\frac{1}{r}\frac{T_{i+1}^p - T_i^p}{2\Delta r}\right] \quad 3.26$$

Donde: Δt, Δr = incrementos discretos en tiempo y radio; i = secuencia de los incrementos radiales.

De esta forma se implementó un programa computacional para obtener datos de tiempo-temperatura.

Para que estos datos sean más reales se introdujeron ciertos factores de desequilibrio e inestabilidad como variación tipo senoidal de la temperatura del autoclave debido a purgas del equipo. De la misma forma, se consideraron perturbaciones y caídas de temperatura en forma de pasos en la temperatura y presión del autoclave, simulando así un fallo en la producción de vapor. Se simularon 18 procesos diferentes, siendo los datos de tiempo-temperatura registrados cada 2 minutos.

Para determinar la distribución de temperatura en un cuerpo, solucionando la ecuación de conducción de calor, deben ser conocidas las propiedades físicas del material, así como las condiciones de contorno existentes (Incropera y de Witt, 1992). Las propiedades físicas de algunos alimentos pueden ser estimadas usando las siguientes ecuaciones:

- Condutividad térmica del alimento:

$$k = k_w x_w + k_{gra} x_{gra} + k_{prot} x_{prot} \quad (3)$$

$$k_w = 0{,}594 + 9{,}57 \times 10^{-4} T \quad 3.28$$

$$k_{gra} = 0{,}179 - 2{,}23 \times 10^{-4} T \quad 3.29$$

$$k_{prot} = 0{,}172 + 2{,}81 \times 10^{-4} T \quad 3.30$$

Donde k (w/m°C) es la conductividad térmica del alimento; k_w (w/m°C) es la conductividad térmica del água; k_{gra} (w/m°C) es la conductividad térmica de la grasa; k_{prot} (w/m°C) es la conductividad térmica de la proteína; T (°C) es la temperatura del alimento; x_w es la fracción de composición porcentual del agua; x_{prot} es la fracción de la proteína y x_{gra} es la fracción de la grasa.

Calor específico:

$$Cp = 1670 + 25\,X \quad 3.31 \quad (7)$$

Donde Cp (J/kg K) es el calor específico, X (% base húmeda) es el contenido de humedad del alimento.

- Densidad del alimento (Lewis, 1993):

$$\rho = \frac{1}{\frac{x_w}{\rho_{água}} + \frac{x_{prot}}{\rho_{prot}} + \frac{x_{gra}}{\rho_{gra}}} \quad 3.32$$

Donde: $\rho_{água}$ es la densidad del agua; ρ_{prot} es la densidad de la proteína y ρ_{gra} es la densidad de la grasa; x_w, x_{prot} y x_{gra} son los mismos valores de la ecuación 3.

Redes neuronales artificiales. La evolución de la inteligencia artificial se remonta a los años 40, cuando McCulloch y Pitts (1943) presentaron una discusión sofisticada de redes lógicas y de elementos de decisión, umbral de comportamiento y memoria (Pádua et al., 2000). Posteriormente, Hebb (1949) propuso la técnica de aprendizaje en redes neuronales sustentándose en correlaciones de las activaciones entre las neuronas (Russel y Norving, 1995). Años después, Frank Rosenblatt (1962) estudió el modelo del perceptron; Hopfield (1986) analizó estudios de asociación de memoria; Hinton (1984) realizó estudios sobre la máquina de Boltzmann y Rumelhart (1986) realizó investigaciones con redes de retropropagación[24].

Existen numerosas formas de definir lo que es una red neuronal artificial, desde las definiciones genéricas hasta las que intentan explicar más detalladamente lo que significa una red neuronal o computación neuronal. Las redes neuronales artificiales están inspiradas en la arquitectura de los sistemas nerviosos biológicos y en la analogía matemática que consisten de un gran número de sistemas relativamente simples, que funcionan en paralelo para tomar decisiones rápidamente. De forma semejante las redes neuronales consisten en un grande número de elementos computacionales primitivos, los cuales están dispuestos en una estructura compactamente paralela.

Tales elementos conocidos como nodos son conectados por medio de sinapsis artificiales, simbolizadas por una matriz de números, los cuales pueden ser ajustados por medio de un proceso de.

[24] HILERA, José R y MARTINEZ, Víctor J. Redes Neuronales Artificiales. Fundamentos, modelos y aplicaciones. Madrid: Ra-ma Editorial. 1995

Para la modelización utilizando redes neuronales se construyó una red de tipo recurrente ya que el tratamiento térmico de alimentos es un proceso de tiempo dependiente. Los parámetros de entrada fueron: tiempo de proceso, temperatura de autoclave y temperatura en el centro del producto para el tiempo presente ti y tiempos pasados ti-1, ti-2 y como parámetro de salida la temperatura del centro del producto para el tiempo futuro ti+1. De esta forma la red fue compuesta por cinco nodos en la capa de entrada y uno en la capa de salida. Para esto se usó el simulador SNNS (Stuttgart Neural Network Simulador) desarrollado para entorno UNIX. Se obtuvieron 2051 datos de tiempo-temperatura para el centro del recipiente cilíndrico correspondiente a las diferentes condiciones de los 18 procesos. Estos datos se usaron para el aprendizaje de la red. Adicionalmente, se preparó otro conjunto de 360 datos que se usaron para la validación de la red. En la etapa de aprendizaje la red se alimentó con datos de las variables operacionales obtenidos a través de simulación, usándose el programa computacional escrito para la solución de la ecuación (3.25).

A través de este aprendizaje la red neuronal modifica sus pesos en respuesta a una información de entrada. Los cambios que se producen durante esta etapa se reducen a la destrucción, modificación y creación de conexiones entre las neuronas. En los modelos de redes

Figura 3.39 Perfil de temperatura del autoclave

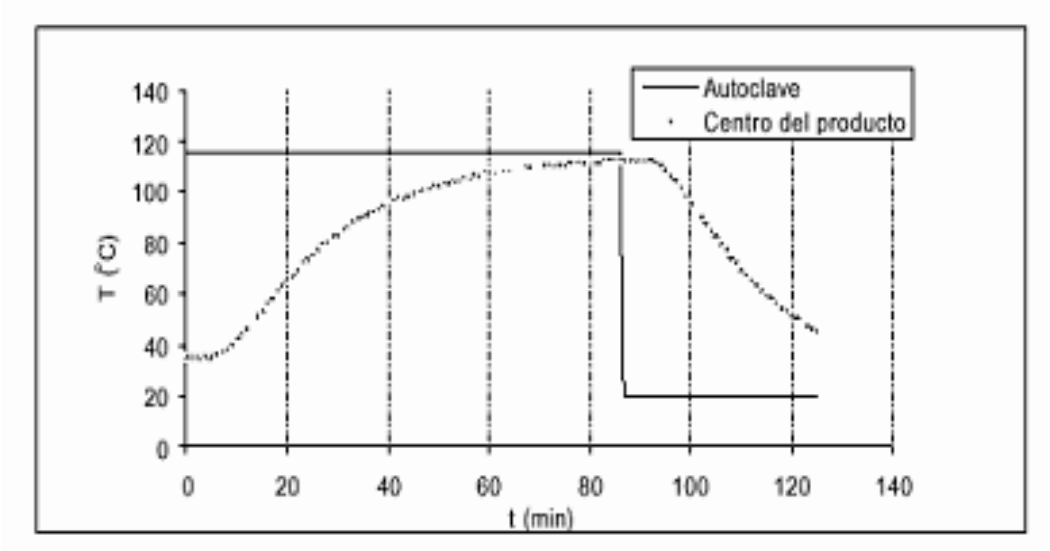
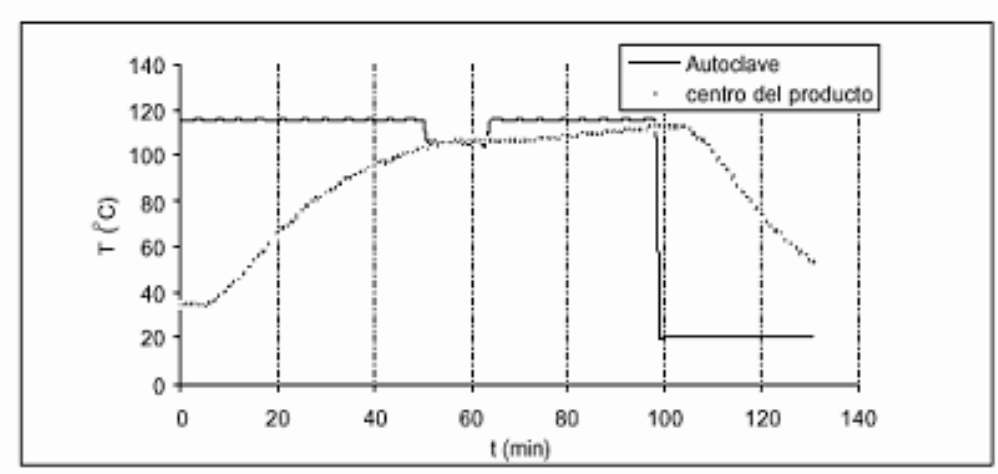


Figura 3.40 Perfil de temperatura del autoclave con perturbación



Neuronales artificiales, la creación de una nueva conexión implica que el peso wij de la misma pasa a tener un valor diferente de cero. De la misma forma una conexión se destruye cuando su peso pasa a ser cero. Durante este proceso de aprendizaje, los pesos de las conexiones de la red sufren modificaciones[25]. Es decir que el error entre el valor de salida de la red y el proporcionado a ella sea mínimo, consiguiendo

[25] Noriega, D. M. J. 1999. Aplicación de las redes neuronales en el análisis sensorial de alimentos. Alimentaria 9: 67-72.

de esta forma mejores predicciones. Para esto se uso un algoritmo de aprendizaje para redes recurrentes “JE-BP:standar backpropagation”, que es una modificación del “back-propagation” y usa los mismos parámetros que la versión “feed-forward” (Sep et al., 1995). Simultáneamente al aprendizaje se realizó la validación.

La topología o arquitectura de redes consiste en la organización y disposición de las neuronas formando capas o agrupaciones de neuronas más o menos alejadas de la entrada y salida de la red (Noriega 1999). En este trabajo la tipología de red fue definida determinándose el número de capas internas y el número de nodos en cada capa, la forma de la red fue del tipo Jordan (Zell et al., 1995), donde los nodos de la capa de entrada están totalmente conectados a la capa oculta y esta a su vez está conectada a cada nodo de la capa de salida. Esas unidades de salida están conectadas a unidades de contexto una a una (conexión recurrente) y finalmente cada unidad de contexto se conecta a cada unidad de la capa oculta. Esta tipología usada se determinó basándose en el método de crecimiento, es decir partiendo de un pequeño número de nodos se analizaron los valores de los errores a medida que el número de nodos se incrementaba. Esto se realizó tanto para una como para dos capas internas. Escogiéndose de esta manera el número de capas internas con su correspondiente número de nodos que resulte en el menor error de validación. Después de haber sido definida la tipología de la red para el problema en estudio, se construyó una curva de validación y aprendizaje, determinándose el número de iteraciones mínimas para lo cual el error cuadrado medio de validación era mínimo.

Posteriormente, la red entrenada fue usada para predecir el perfil de temperatura temporal para nuevas condiciones de entrada, comprobando de esta manera su generalización.

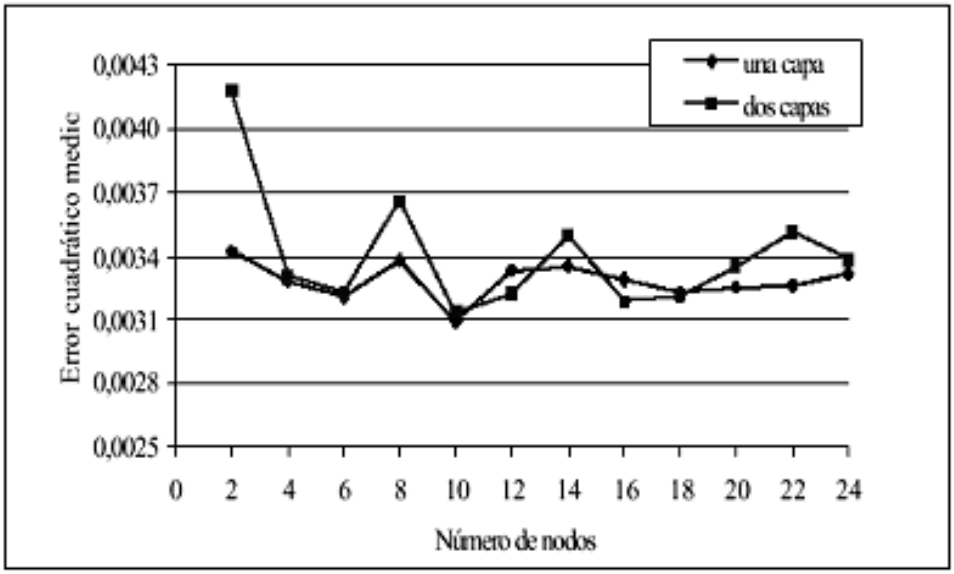
RESULTADOS Y DISCUSIÓN

Para la obtención de los perfiles de temperatura fue utilizada la solución de la ecuación (3.25). Los valores de las propiedades físicas consideradas constantes

durante el proceso se determinaron mediante las ecuaciones 3.27-3.32, resultando en: densidad del producto: 1058 kg/m; calor específico: 3170 J/Kg K y conductividad térmica: 0,469 W/m K.

Las Figuras 3.39 y 3.40 representan los perfiles de temperatura para el centro del producto y del autoclave en función del tiempo, para algunas de las situaciones

Figura 3.41 Error cuadrático en numero de nodos de las capas ocultas



Estudiadas. Como puede observarse, cuando fueron introducidas perturbaciones de tipo paso (Figura 3.40), el tiempo del proceso aumentó afectando a la curva de tiempo-temperatura de esterilización, en comparación con el perfil de temperatura para autoclave constante (Figura 3.29). Esto puede ser el caso en un proceso a escala industrial.

Estas perturbaciones plantean predecir fallos en el procesamiento térmico real, debido a variaciones de temperatura en el autoclave provocada por la caída de presión de vapor de la caldera.

Desarrollo de la red neuronal

Determinación de la topología o arquitectura Teóricamente una vez entrenada y validada, el desempeño de la red puede ser comprobado, empleándose otro grupo de datos escogidos al azar (Huang y Mujudar, 1993). Sin embargo, para obtener un buen proceso de aprendizaje, es necesario determinar una configuración óptima de la red y también, el número de iteraciones, para obtenerse el menor error de validación. Inicialmente se utilizó un número de iteraciones para aprendizaje escogidas aleatoriamente igual a 6000. El número de neuronas en la capa interna fue variando de 2 en 2 hasta alcanzar 24. Se comprobaron redes con una y dos capas internas. El error asociado calculado fue el error cuadrático medio, obtenido con los datos de validación.

La Figura 3.41, representa la variación del error cuadrático medio en función del número de nodos, con una o dos capas internas. Se observó que el menor error se obtuvo para un valor de 10 nodos y que había convergencia tanto para una como para dos capas internas. El aumento del número de nodos aumentó el error de validación, además de necesitar mayor tiempo y esfuerzo computacional para una red con dos capas internas. Así, se escogió una red con apenas una capa interna puesto que los errores mínimos eran relativamente. Por lo tanto, la topología de red para el proceso fue una capa de entrada con cinco nodos, una capa interna con diez nodos y una capa de salida con un nodo.

3.6.5 Tasa de aprendizaje

Para determinar la tasa de aprendizaje adecuada, se comprobaron tres valores de tasa de aprendizaje, usándose la topología de red determinada anteriormente. La Figura 3.42 representa la curva de validación para los tres valores de tasa de aprendizaje.

Puede observarse que, para valores muy bajos como 0,01, el proceso de aprendizaje fue muy lento, siendo necesario un número alto de iteraciones para disminuir el error de validación y para valores de tasa de aprendizaje demasiado altos como 5,0 el comportamiento de la curva de validación fue en cierto modo irregular, alcanzando inicialmente un valor mínimo local para después aumentar. Para un valor intermediario, tal como 0,3, el comportamiento de la curva de validación fue él más adecuado llegando a alcanzar el valor mínimo de validación con 2300 iteraciones. Por tanto, para la simulación se utilizó una tasa de aprendizaje de 0,3.

Aprendizaje y validación

Una vez establecida la topología de red 5-10-1 y tasa de aprendizaje de 0,3 se realizó el entrenamiento con los 2051 datos y simultáneamente la validación con los 360 datos adicionales. La Figura 3.42 representa las curvas de aprendizaje y validación, donde se observa que el error de aprendizaje y

Validación disminuyen a medida que el número de iteraciones aumenta, siendo ese su comportamiento característico. Pero se observa que la curva de validación muestra un punto mínimo en el valor del error cuadrático medio cuando el número de iteraciones alcanza 2300. Un entrenamiento con número de iteraciones superior a este significa la incorporación

Figura 3.42 Suma del error cuadrático en función del número de iteraciones.

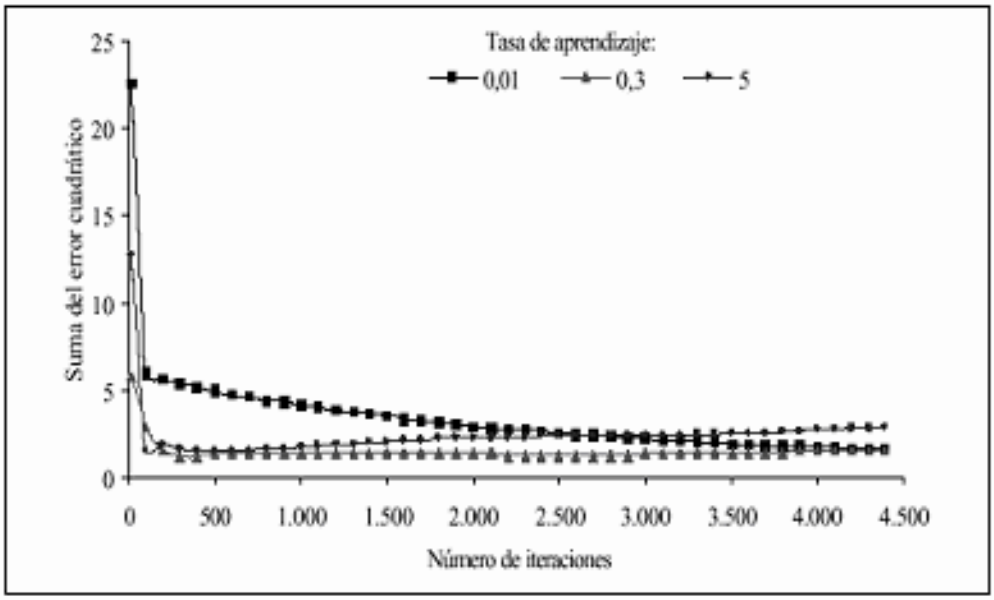


Figura 3.43 Suma del error cuadrático en función del número de iteraciones

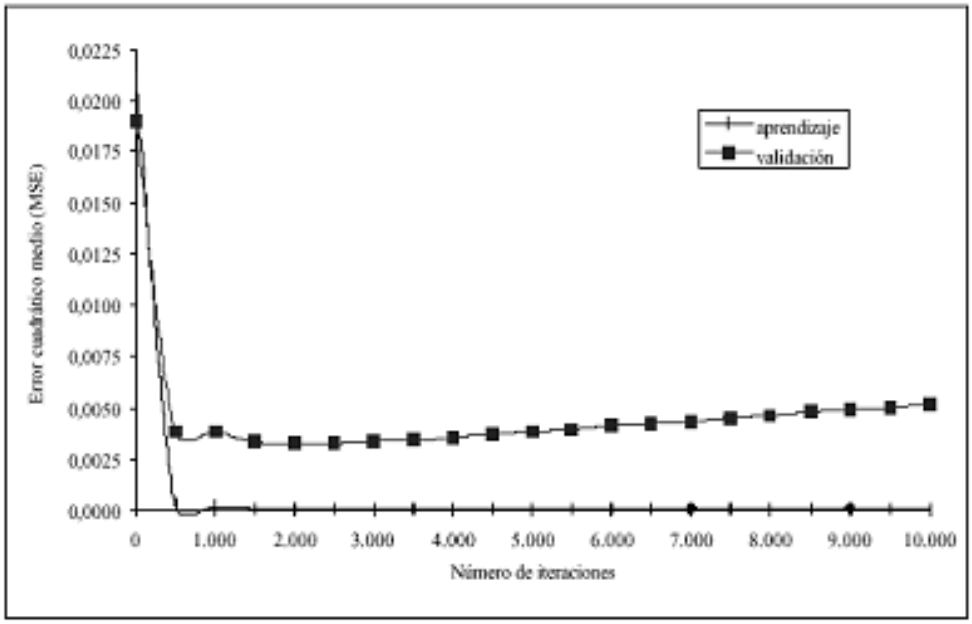


Figura 3.44 Validación simulados versus predichos

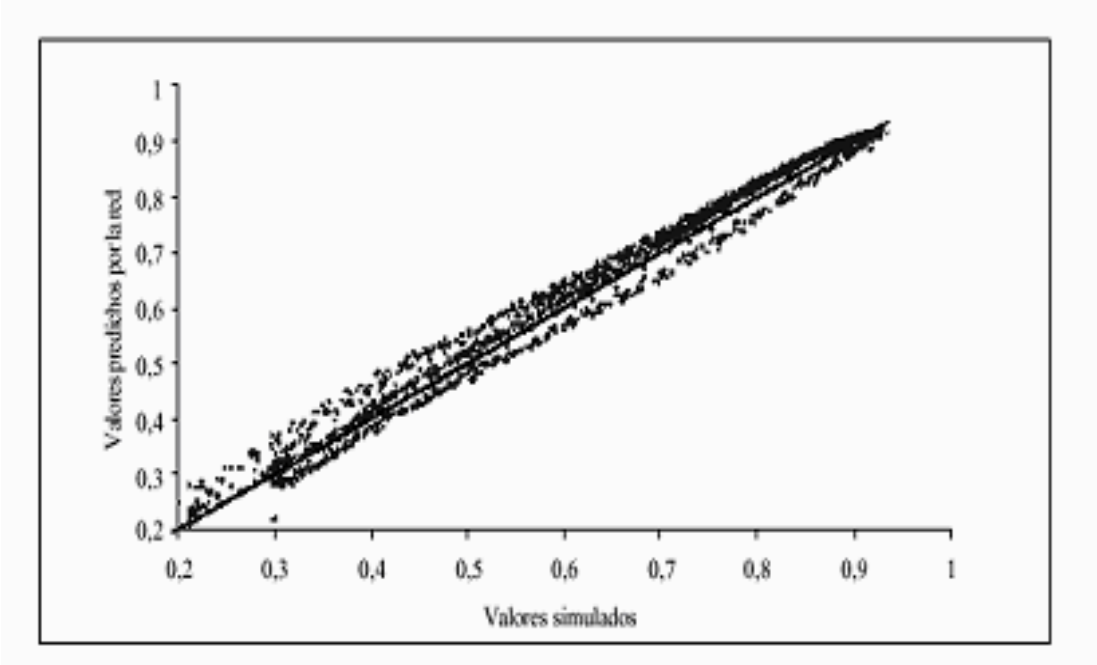
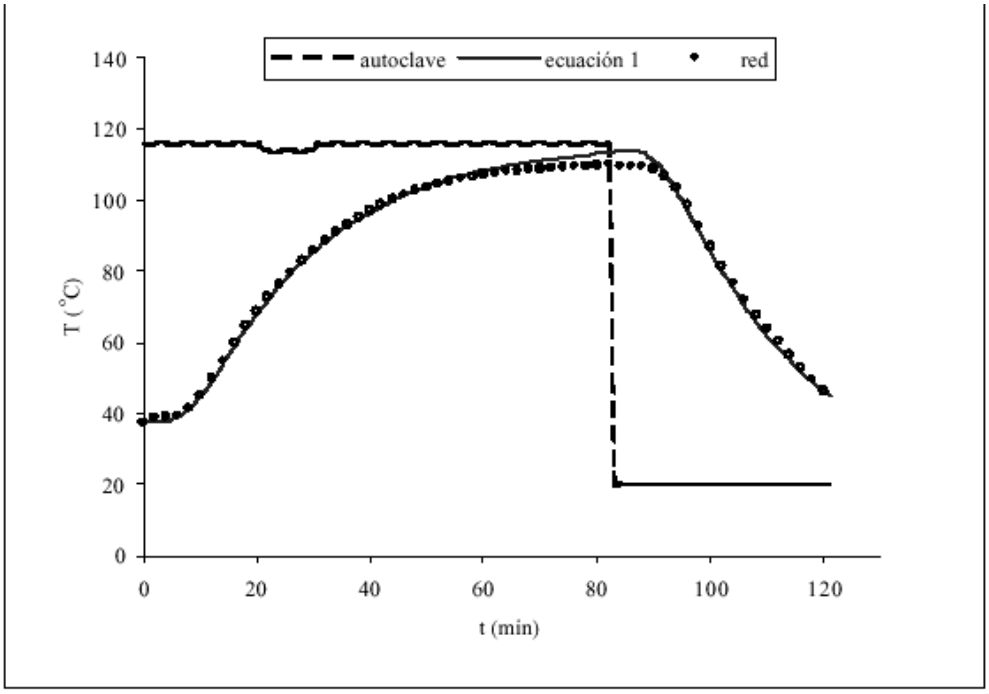


Figura 3.45 Temperatura del autoclave en función del tiempo.



de ruidos en el modelo, perdiendo así su característica de filtración. El error mínimo alcanzado fue de 0,0031. El análisis de residuales, es decir los valores predichos por la red versus los valores simulados por la ecuación 1 están representados en la Figura 6, con una distribución aleatoria a lo largo de la recta x-y.

Generalización

Para comprobar el desempeño de la red desarrollada y entrenada, se prepararon procesos desconocidos por ella, presentados en la Tabla 2. Es decir, condiciones que no fueron usadas en las etapas de aprendizaje y validación.

En las Figuras 3.45 y 3.44 se representa el perfil de temperatura en el centro del producto en función del tiempo para dos casos de las diferentes situaciones estudiadas.

Las curvas indican que la red consiguió generalizar bien esas nuevas situaciones presentadas, prediciendo el perfil de temperatura para esos casos. El error cuadrático medio obtenido fue del orden de 10-4.

CONCLUSIONES

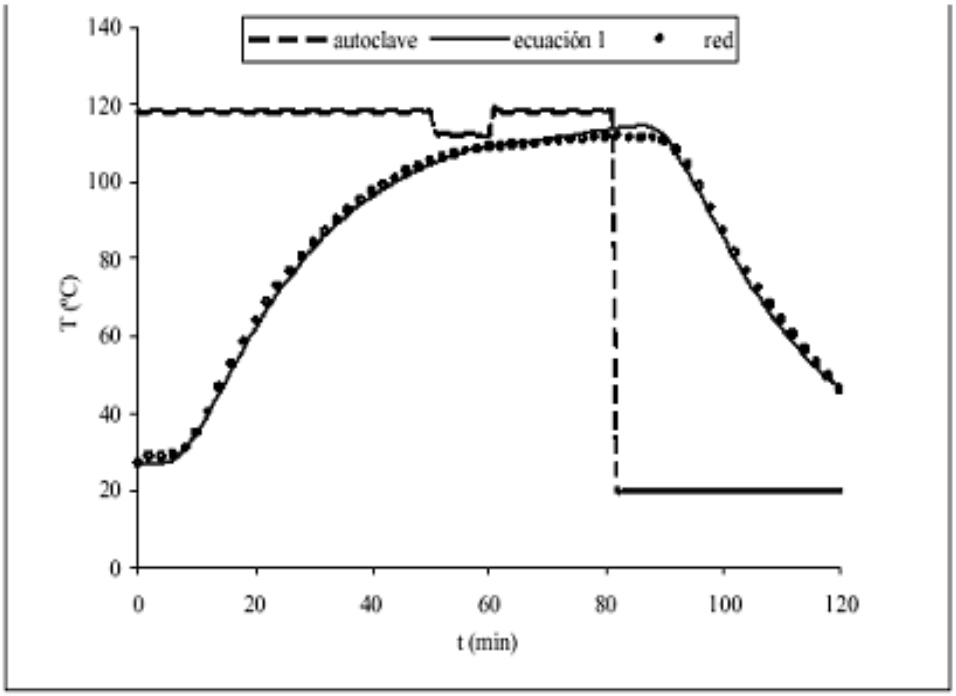
Se ha propuesto un sistema de aplicación de una red neuronal artificial, que se ha demostrado útil para simular el perfil de temperatura para el centro de un producto, usando como base los datos generados por un balance diferencial de energía durante el tratamiento térmico de esterilización.

Tabla 3.6. Condiciones operacionales

Proceso	Condiciones Operacionales	
	Temperatura del autoclave (°C)	Temperatura inicial del producto (°C)
1	116 ± 0,5	38
2	118 ± 0,5	27
3	120 ± 0,5	31
4	121 ± 0,5	42

Redes neuronales es una técnica y herramienta útil en la modelización de sistemas complejos dependientes del tiempo, como es el caso del tratamiento térmico de alimentos, cuyas propiedades físicas del producto son difíciles de predecir. Esta técnica no requiere de tales informaciones y sí de datos de entrada y salida representativos del proceso

Figura 3.46 Perfil de temperatura.



3.7 SENSORES VIRTUALES MEDIANTE REDES NEURONALES ARTIFICIALES.

Raúl R. Leal Ascencio y Enrique Herrera.*
Departamento de Electrónica, Sistemas e Informática, ITESO. Periférico Sur 8585, CP 45090, Tlaquepaque,
Jalisco, México. Correo Electrónico:
rleal@iteso.mx
. Tel: (3) 669 3598, Fax: (3) 669 3511.
** CIATEJ. Centro de Investigación y Asistencia Técnica del Estado de Jalisco, A.Cñ̃. Av. Normalistas 800 Apdo.*
Postal 2-191, CP 44270. Guadalajara, Jalisco, México

INTRODUCCIÓN.

Cualquier sistema de control o de monitoreo requiere de elementos de interfase con el mundo real, el mundo físico. Así, un proceso industrial requiere de una diversidad de sensores para observar e identificar su estado actual y tomar decisiones de control. Para un robot móvil es esencial tener información de su alrededor para saber hacia donde moverse o desarrollar estrategias para elegir la trayectoria más apropiada. En un proceso de producción es de vital importancia saber si el sistema de producción en realidad está produciendo la calidad para la que fue diseñado, por lo tanto se requieren sensores para verificarlo. Dado que las áreas de aplicación de los sensores son muy diversas, existen sensores de muchos tipos, que se eligen

dependiendo de la naturaleza de la variable a medir y de las condiciones de la aplicación. En el proceso industrial antes mencionado puede haber distintos tipos de sensores midiendo distintas variables que son relevantes en el desarrollo del proceso. El cómo procesar los datos de los sensores en este caso y hacer deducciones del proceso basados en esa información es motivo de muchos estudios y existen ya teorías de control para aplicar la información según la naturaleza del proceso. Para un caso como el descrito se usarían modelos multivariable del proceso y teorías de control adecuadas para esos modelos.

En el caso del robot y el sistema de control de calidad la situación del manejo de los datos de los sensores puede tornarse bastante distinta. En este caso lo más común es que el interés sea una sola variable (como la dirección a moverse de tal manera que no haya colisiones ó la brillantez del producto) pero en cambio se tienen varios sensores midiendo la misma variable. En este caso el objetivo sería usar sensores que actúen bajo distintos principios para medir la misma variable, de tal manera que la información sea más confiable. Se busca redundancia para obtener confiabilidad. Ahora el problema de procesamiento de la información es muy distinto ya que se requiere de la fusión de la información de fuentes distintas (distintos tipos de sensor, distintos niveles de las variables, etc.) para obtener el valor de una sola variable.

Existen situaciones en las que el sensor adecuado no existe o es prohibitivamente caro por lo que se desearía una manera de estimar la información de esa variable. En muchos casos la información sobre esa variable inmensurable existe pero en otras formas. Por ejemplo, se sabe que en todo proceso la presión y temperatura están ligadas, por lo que sabiendo una, se podría inferir la otra mediante el procedimiento adecuado. En el caso de procesos de fermentación el monitoreo de la variable de biomasa es esencial, sin embargo, no existe un sensor para medirla. Existen básicamente dos métodos para medir esta variable: a) el método automático a través de un cromatógrafo o densidad óptica y b) el método manual a través de muestras tomadas a mano y analizadas por un especialista. El método a) cuesta al menos $8,000.00 dólares por lo tanto se prefiere el método b). Éste método no

provee muestras periódicas ni de suficiente frecuencia por lo tanto no es completamente satisfactorio.

Basados en esta idea de inferencia de información han nacido los llamados sensores por software o soft-sensors o sensores virtuales. Estos son programas que son los encargados de hacer la inferencia tomando la información existente. Los programas pueden consistir en un modelo matemático de cómo hacer la inferencia, en un modelo heurístico o un 'modelo inteligente'.

Un método usado como modelo inteligente es el de las redes neuronales artificiales (RNA). Las RNA son un conjunto de elementos procesadores adaptables que simulan burdamente el funcionamiento de una neurona animal. En una aplicación como esta, las RNA harían un modelo inferencial de la variable en cuestión tomando como base la información de las variables medidas. En el área de procesos industriales las variables del proceso siempre están ligadas de una u otra manera. En ocasiones la interrelación entre variables es compleja y altamente no lineal, sin embargo, las RNA y han demostrada ser capaces de aprender a base de entrenamiento estas dinámicas.

SENSORES VIRTUALES

En muchos tipos de procesos industriales y en otros casos donde se tiene multiplicidad de sensores y monitoreo o control por computadora es posible implementar un sensor virtual. Ya sea porque es difícil medir la variable en cuestión, porque el sensor es muy caro, demasiado lento o inexacto los sensores virtuales se hacen necesarios y ahora son cada vez más populares. Se está desarrollando ya la primera propuesta comercial de un sensor virtual múltiple de este tipo para la industria automotriz. En esta aplicación se usa un modelo neuronal del motor del automóvil para hacer asociaciones altamente complejas, no lineales y multidimensionales.

Las asociaciones que hace el modelo neuronal se generan mediante entradas de los sensores estándar de los motores actuales y las salidas deseadas, que en este caso son par del motor, emisiones tóxicas y consumo de combustible en todo el rango de operación. El modelo neuronal se obtiene mediante entrenamiento en una estructura de prueba del motor con un dinamómetro o entrenamiento en línea con el automóvil funcionando. En base a un entrenamiento de aproximadamente 30 minutos en un motor caliente bajo pruebas altas en transientes el modelo del sensor virtual ha dado resultados de 1 a 3% de error con respecto a los valores reales de las variables en cuestión. En el caso de las emisiones de gases este nivel de exactitud es equivalente al que proveen los instrumentos analizadores de emisiones usados para obtener los datos de entrenamiento. Este sistema provee además niveles altos de generalización debido a los entrenamientos intensos en niveles de operación distintos y a la capacidad natural de generalización que proveen las redes neuronales.

En los procesos que nosotros hemos estudiado las interrelaciones entre las variables y la naturaleza del proceso en sí se presumen más complejas que el caso automotriz. Las fermentaciones nunca son iguales aun con los mas estrictos controles de condiciones iniciales, substratos e instrumentación. Los microorganismos son siempre muy sensibles al conjunto de pequeñas variaciones en las distintas variables, por lo tanto se requiere de un sistema de estimación altamente robusto y con muy buena capacidad de generalización. Una de las razones por la que los sensores virtuales son muy robustos en su funcionamiento es su característica de redundancia intrínseca sin incurrir en gastos adicionales de hardware o más sensores. El hecho de tener como entradas datos de una multiplicidad de variables y que éstas puedan ser dependientes unas de otras o estar relacionadas de alguna manera, provee esta redundancia intrínseca que evita fallas drásticas aun en el evento de fallas en los sensores.

Presentamos el planteamiento y algunos resultados de dos casos de estudio en los que hemos estado involucrados.

En el primer caso que analizamos se trata de la estimación de biomasa en un proceso de producción de un antibiótico como producto secundario. En el segundo, se produce un pigmento rojo altamente cotizado.

PREPROCESAMIENTO

Los datos del proceso fueron normalizados antes de ser presentados a las RNA de tal manera que el rango de todas las variables se redujera a entre 0.1 y 0.9 para obtener una convergencia más rápida en el entrenamiento de las redes[26]. También se aplicó interpolación lineal a los datos históricos de la biomasa para obtener una razón de muestreo periódica y uniforme para todas las variables de interés. Basados en experimentos previos, se decidió aplicar un periodo de muestreo constante de dos horas para todas las variables. Las variables en uso como entradas a las RNA son la razón de evolución del dióxido de carbono (CER, por sus siglas en inglés), oxígeno disuelto, tiempo transcurrido en la fermentación y pH. Se ha llegado a la elección de estas variables, considerando también reportes de otros investigadores.

LAS REDES NEURONALES ARTIFICIALES.

Estamos usando redes de alimentación hacia adelante con una capa escondida. El número de entradas a cada red es igual al número de variables que se ha escogido para cada una y que se presenta más adelante. Basados en experimentos previos, el número de neuronas en la capa escondida de cada red será de 6. La salida de todas las redes es siempre la biomasa (sólo una salida). Hemos encontrado para

[26] FREEMAN, James y SKAPURA, David. Redes Neuronales: Algoritmos, aplicaciones y técnicas de programación. Delaware E.U.A: Addison Wesley Iberoamericana S.A. 1993

esta aplicación que si se quiere mantener una buena capacidad de generalización de las redes, el criterio de convergencia (la figura de error) debe ser holgado. Para este caso elegimos un valor de alrededor de 5% del rango de los datos normalizados. Las funciones de activación de la capa escondida son tipo sigmoide y en la salida se tienen funciones de activación lineales. El algoritmo de entrenamiento es 'retropropagación' (Backpropagation) con regla de ajuste Levenberg-Marquardt. La figura de error que se usa para evaluar el desempeño de las redes es el Error Absoluto Medio[27].

$$EAM \propto \sum_{i}^{n} abs(\hat{Y}_i - T_i) \Big/ n \qquad 3.33$$

Donde Y es el vector de estimación, Tes el vector de valores deseados, el número de valores de entrada iy abs(*) indica el valor absoluto. El factor de proporcionalidad se debe a que el valor del EAM es un porcentaje del rango de los datos normalizados.

La Aplicación.

Como primer paso después de este preprocesamiento se entrenan cuatro RNA para estimar el valor de la biomasa, teniendo cada una un grupo distinto de entradas. Los grupos de entradas son los siguientes:

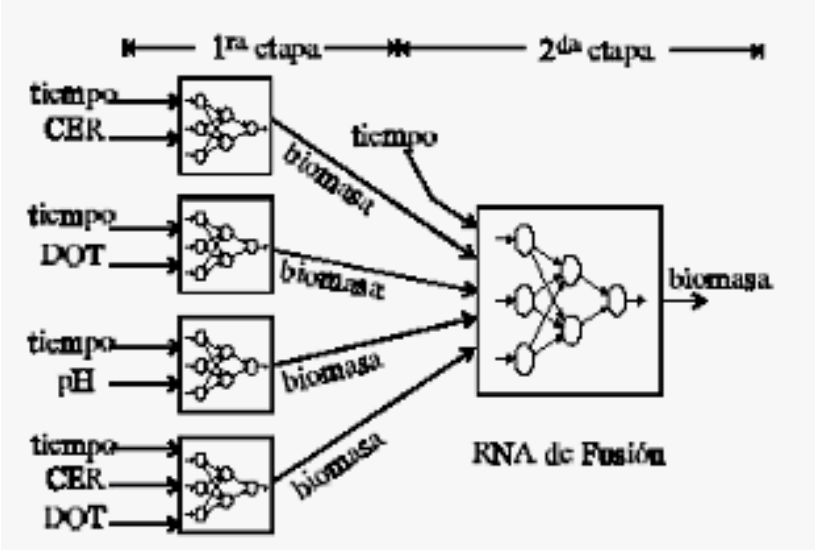
1) CER y tiempo transcurrido (CT).
2) DOT y tiempo transcurrido (DT).
3) pH y tiempo transcurrido (PT).
4) CER, DOT y tiempo transcurrido (CDT).

A las estimaciones de biomasa de estas RNA les llamamos resultados de la primera etapa. Existe una segunda etapa de procesamiento que llamamos etapa de fusión,

[27] HILERA, José R y MARTINEZ Víctor J. Redes Neuronales Artificiales. Fundamentos, modelos y aplicaciones. Madrid: Ra-ma Editorial. 1995.

para la que proponemos una de dos opciones, como se explica adelante. En la primera se usa una RNA a la que le llamamos 'red de fusión'. Para ésta, las estimaciones de las cuatro RNA de la primera etapa, más la señal de tiempo transcurrido se usan como entradas para entrenar la red de fusión para estimar biomasa nuevamente, como se ilustra en la Fig. 2. La arquitectura de la red de fusión es igual a las demás con una capa escondida y 6 neuronas en esa capa.

Figura 3.47 RNA de fusión



Una segunda forma de implementar la fusión se logra mediante un programa que elige las mejores estimaciones de las proporcionadas por la primera etapa. Así, para cada muestra, se elige la mejor estimación con respecto a una referencia. Cómo generar esa referencia es un problema interesante ya que la referencia óptima es precisamente la variable que se está tratando de generar.

La solución que proponemos es tomar esta decisión basados en un pronóstico de un paso hacia adelante. Este es una RNA que, entrenada con datos de biomasa en el tiempo ty el tiempo transcurrido (dos datos que se tienen), genera una predicción de la biomasa en el tiempo t+1. Esta predicción es bastante buena ya que sólo se predice un paso de tiempo por lo tanto sirve bien para el efecto que se desea. Este método también provee información de qué conjunto de entradas es más importante

en cada paso de tiempo. Este segundo método de fusión está ilustrado en la Fig. 3.47 Se ha observado que errores por fallas de los sensores y ruido son disminuidos en gran medida usando esta técnica. A este procedimiento le hemos llamado Sistema de Análisis Temporal (SAT).

Pruebas sin perturbaciones.

Para estos dos métodos de procesamiento, se entrenan las redes con el grupo de datos de una fermentación y se prueba el desempeño con tres grupos distintos de fermentaciones del mismo tipo, repetidas. En la siguiente tabla se muestra un resumen de los resultados de las pruebas.

Los errores están en términos de EAM y presentados en porcentaje del rango normalizado de datos (EAM/0.8).

Tabla 3.7. Resumen de resultados de biomasa.

	F181	F187	F253	F256	**Media**
FUS	4.52%	1.19%	9.62%	5.90%	**5.31%**
SAT	2.65%	2.41%	5.75%	7.49%	**4.58%**

En la tabla de resultados F187 se refiere a los datos de la fermentación que se usó para entrenamiento de las redes, por eso presenta los errores más bajos. En términos de promedios, las estimaciones de los dos métodos están alrededor de 5% lo cual puede no parecer muy impresionante, aunque se está demostrando una capacidad de generalización en realidad muy importante. Lo que es más impresionante es la demostrada robustez de ambos sistemas. Las figuras Fig. 3.49 y

Fig. 3.50, muestran gráficas de los resultados para una de las fermentaciones de prueba, F181. En Fig. 5, el símbolo en el recuadro indica el origen de la estimación.

Figura 3.48 Resultados de la estimación de biomasa con la red de fusión para la fermentación F181.

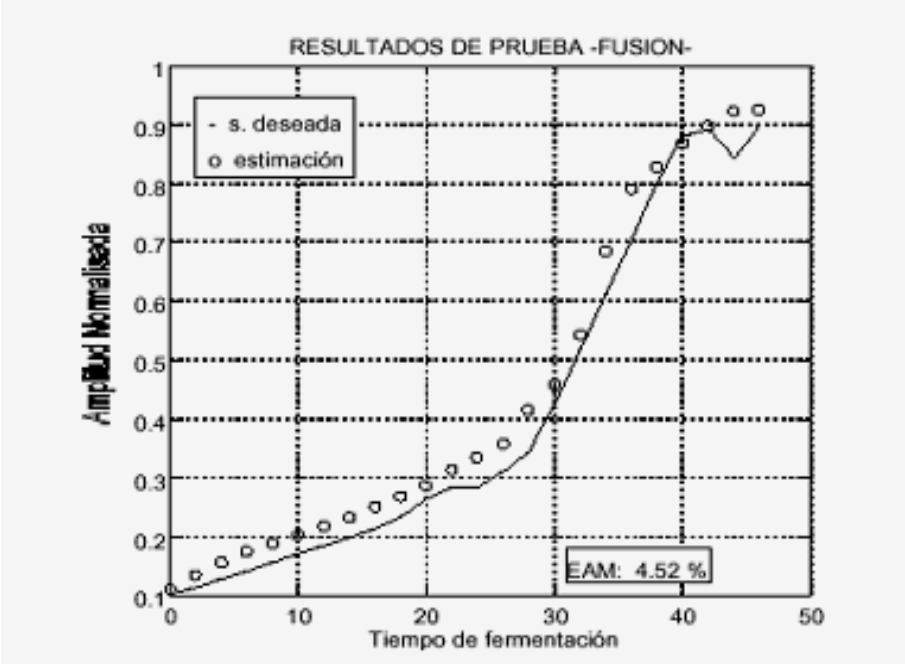
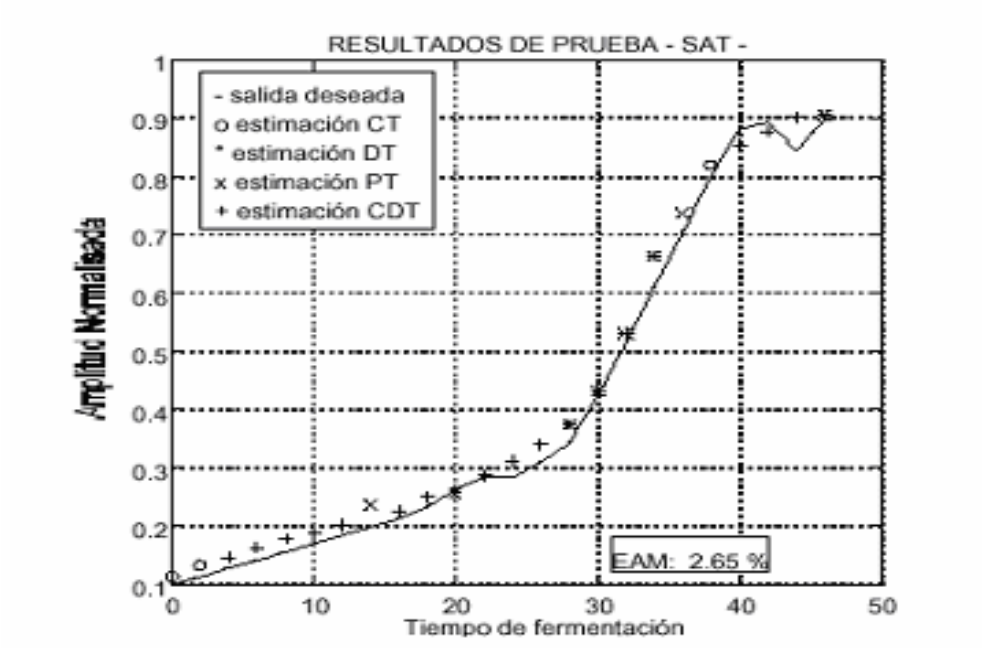


Figura 3.50 Estimación de biomasa del sistema de análisis temporal para la fermentación F181.



Pruebas con perturbaciones.

Con el propósito de probar la robustez de los estimadores, investigamos su comportamiento ante varios tipos de perturbaciones. Un tipo de perturbación que probamos fue ruido aleatorio. Estas pruebas se hicieron añadiendo un vector de ruido a la señal que nos interesaba afectar. La magnitud de la señal de ruido es igual a la de la variable afectada pero el ruido se aplica mediante la siguiente ecuación:

señal _ ruidosa (i)=señal (i)+señal (i)*ruido (i) 3.34

La razón de ser de Ec. 3.34 es que creemos que modela bien una situación real donde la magnitud del ruido es proporcional a la magnitud de la señal. Esta perturbación se aplica solamente a la señal CER porque es la más importante. Para las otras variables se usan los datos reales del proceso. La Tabla 2 muestra un resumen de los resultados de esta prueba

Tabla 3.8. Resumen de los resultados de la primera y segunda etapas de estimación para las cuatro corridas de la fermentación.

	1ra. Etapa				2da Etapa	
	CT	DT	PT	CDT	FUS	SAT
F181	5.6%	3.37%	5.65%	5.28%	3.94%	3.14%
F187	6.65%	2.15%	2.13%	9.15%	2.99%	2.13%
F253	6.95%	5.81%	10.25%	5.85%	6.34%	5.15%
F256	9.96%	8.57%	10.09%	11.8%	6.8%	8.73%
Media	7.29%	4.98%	7.03%	8.02%	5.02%	4.79%

CONCLUSIONES

Hemos demostrado que una red neuronal artificial es capaz de aprender la inerrelación entre la biomasa y algunas de las variables más importantes del proceso en su conceptualización como sensor virtual. Se propone una mejora en las estimaciones a través de un método de dos etapas promoviendo redundancia de datos. En la primera etapa se generan varias estimaciones de biomasa a partir de varios agrupamientos de variables de entrada. En la segunda etapa se fusionan estas estimaciones para obtener una estimación mejorada de biomasa. Para esta fusión se proponen dos métodos. El primero consiste en emplear una RNA de fusión para estimar la biomasa tomando como entradas las estimaciones de la etapa anterior. El segundo método consiste en entrenar un RNA tipo TDNN (Time Delay Neural Network) para hacer un pronóstico de biomasa un paso adelante para elegir la mejor de las estimaciones de la primera etapa. A éste último método le llamamos sistema de análisis temporal (SAT). En ambos casos de estudio se ha observado que, dado el entrenamiento apropiado, la metodología de fusión de datos a través de

RNA es capaz de estimar la biomasa con una precisión comparable a errores de instrumentación.

Más aun, los resultados muestran que las estimaciones son casi insensibles a ruido y a fallas en los sensores debido a la integración y redundancia disponibles. Esta será una de las principales ventajas que provean los sensores virtuales.

4. CONCLUSIONES

El neurocontrolador que se obtuvo en esta investigación da un buen comportamiento al servosistema, cuando se ve sometido a las distintas variaciones de la señal de referencia, y comparado con el PID el tiempo de respuesta del neurocontrolador es superior a la del PID, y a perturbaciones externa la velocidad de estabilización del neurocontrol es mas rápida. Otro aspecto interesante del neurocontrolador es que este hace un buen mapeo de la señal de referencia (set-point) en la salida del servosistema.

Como principio básico cuando se va a entrenar una red neuronal los datos de entrenamiento tienen que ser normalizados, para que al entrenarse puedan alcanzar un error bajo, con esto se justificaría que la red logra un entrenamiento optimo, pero tal afirmación no sucedió en esta investigación, debido a que las redes que obtuvieron el mayor error de entrenamiento son las que hicieron la mejor modelizacion del PID, y casualmente esta redes fueron entrenadas con datos sin normalizar.

El neurocontrolador demostró lo estable que son las redes neuronales a los distintos cambio en que son sometidas, por ejemplo la amplitud de los sobreimpulsos que muestra el neurocontrolador en la señal de salida es mucho menor que el mostrado por el PID, brindando de esta forma un servosistema estable a condiciones cambiante de su entrono.

En el aspecto económico los neurocontroladores son mas baratos en comparación con el neurocontroladores existentes, además si las condiciones de la planta son cambiadas o se desea hacer algunos cambio en la función de transferencia de la planta el neurocontrolador podría ser entrenado nuevamente y el proceso de adaptación del neurocontrolador será mas rápido y eficiente que el que presentase el PID.

El error estacionario que presenta el neurocontrolador elegido podría decirse que fue el único aspecto negativo que mostró este neurocontrolador, pero hay que recordar que el valor de este error no es muy grande y para algunos procesos no presentaría ningún tipo de problema la aplicación de este neurocontrolador.

Se logro recopilar información importante sobre las aplicaciones de los neurocontroladores en varios campos, no solamente en el control automático a nivel industrial si no en otros campos tales como en la medicina, en la preedición de demanda de transporte por cierta vía etc. Además se pudo hacer una clasificación de los neurocontroladores dependiendo la forma en que se van a utilizar.

Se aprendió que las redes neuronales es un tema de mucha investigación, dedicación y comprobación de resultados, dado que es una tecnología emergente la cual se viene desarrollando en nuestro medio, los resultados obtenidos en esta monografía no son del todo absoluto pero al final se encontró con un comportamiento no esperado por las redes neuronales. Este fue el caso mostrado por la red backpropagation que presento el error mas alto en el entrenamiento la cual desempeño un mejor comportamiento en el servosistema que el mostrado por las otras redes backpropagation, o sea que para asegurar los resultados aquí encontrados habrá que hacer una investigación mas exhaustiva y poder asegurar su veracidad.

BIBLIOGRAFÍA

A.J, Serrano. Uso de redes neuronales en el problema de la emesis posquimioterapia. Tesis de Licenciatura, Valencia 1998

CHAPMAN, Stephen J. Máquinas Eléctricas. Santafé de Bogotá Colombia: McGraw Hill. 1997.

Diseño e implementación de un sistema de control avanzado para procesos complejos. Autor: José Gallardo Arancibia, U Católica Del Norte Chile.
http://www.isisu02.usal.es/~airene/capit9.pdf

Diseño e implementación de un neurocontrolador aplicado a un servosistema no lineal.
http://fiee.uni.edu.pe/publicaciones/TecniaEsp/02art/

Diseño y simulación de un sistema de control mediante métodos de espacio de estado, Universidad De Guadalajara.
http://proton.ucting.udg.mx/materias/moderno/modulo5.htm

FREEMAN, James y SKAPURA, David. Redes Neuronales: Algoritmos, aplicaciones y técnicas de programación. Delaware E.U.A: Addison Wesley Iberoamericana S.A. 1993.

HILERA, José R y MARTINEZ, Víctor J. Redes Neuronales Artificiales. Fundamentos, modelos y aplicaciones. Madrid: Ra-ma Editorial. 1995

INCROPERA, P. F.; De Witt, D.P. 1992. Fundamentos de transferencia de calor e masa. Rio de Janeiro: Guanabara Koogan, 455 p.

KATSUHIKO, Ogata. Ingeniería de control moderna. Englewood Cliffs, N.J.: Prentice Hall, 1993.

K.S., Nerendra. Identification and control of dynamic systems using neural networks. IEEE Trans. On neural networks. Vol 1, No. 1. 1990.

M. SAERENS Y A. SOQUET. Neural controller based on back-propagation algorithm IEE Proceedings-F, Vol. 138, No.1, Febrero de 1991.

M. KOSTENKO Y L. PIOTROVSKY. Maquinas eléctricas. Barcelona, Editorial montaner y simón, S.A.

NARENDRA, Kumpati S. y PARTHASARATHY Kannan. Identification and Control of Dynamical Systems Using Neural Networks, IEEE Transactions on Neural Networks. Vol. I. No. 1, March 1990.

Noriega, D. M. J. 1999. Aplicación de las redes neuronales en el análisis sensorial de alimentos. Alimentaria 9: 67-72.

Russell, J.S, Norving, P. Artificial intelligence a modern approach. New Jersey. Prentice Hall:Upper Saddle River, 1995.

SAES, Doris (Marzo, 2002). Apuntes II. Control basado en redes neuronales. Seminario AADECA-UBA, Buenos Aires.

Tutorial del grupo de circuito de la Universidad Politécnica de Madrid-UPM, bajo la dirección del Dr. Diego Andina de la Fuente.
/http://www.gc.ssr.upm.es/inves/neural/ann2/anntutorial.html.

Virseda, P.; Abril, Y.J. 1997. Simulación numérica en estado no estacionario de espárragos durante el proceso de esterilización. Alimentaría 3: 43-46.

WIDROW, Bernard y LEHR, Michael A. 30 years of adaptive neural networks: Perceptron, Madaline, and Backpropagation. Procedimiento de l IEEE, vol 78 #9, Sptiembre 1990, pp 1415-1442.

Zamarreño J.M. , Vega P.,Neural predictive control: application to a hightly non-linear process, World Congress International Federation of Automatic Control, vol. C: Control. Design I, 19-24 , 1996, IFAC'96.

ANEXOS