Implementación de neurocontroladores en línea. Tres configuraciones, tres plantas[1]

The Implementation of On-Line Neurocontrollers: Three Configurations, Three Plants[2]

Implementação de neurocontroladores em linha. Três configurações, três plantas[3]

Código SICI: 0123-2126(201206)16:1<163:IDNELT>2.3.TX;2-Z

José Danilo Rairán-Antolines[4]
Diego Fernando Chiquiza-Quiroga[5]
Miguel Ángel Parra-Pachón[6]

---

[1] Fecha de recepción: 7 de junio de 2011. Fecha de aceptación: 18 de noviembre de 2011. Este artículo se deriva de un proyecto de investigación denominado *Planteamiento de un algoritmo de control de sistemas dinámicos nuevo, basado en el proceso de toma de decisiones humano*. Número de registro: 1-72-176-09. Desarrollado por el Grupo de Investigación en Control Electrónico (GICE) de la Universidad Distrital Francisco José de Caldas, Bogotá, Colombia.

[2] Submitted on June 7, 2011. Accepted on November 18, 2011. This article is the result of the research project *Development of an Algorithm for the Control of Dynamic Systems Based on the Process of Human Decision Making*. Developed by the research group Control Electrónico (GICE), Universidad Distrital Francisco José de Caldas, Bogotá, Colombia. Registration number 1-72-176-09.

[3] Data de recepção: 7 de junho de 2011. Data de aceitação: 18 de novembro de 2011. Este artigo deriva de um projeto de pesquisa denominado *Proposta de um algoritmo de controle de sistemas dinâmicos novo, baseado no processo humano de tomada de decisões*. Número de registro: 1-72-176-09. Desenvolvido pelo Grupo de Pesquisa em Controle Eletrônico (GICE) da Universidade Distrital Francisco José de Caldas, Bogotá, Colômbia.

[4] Ingeniero electricista, Universidad Nacional de Colombia, Bogotá, Colombia. Magíster en Automatización Industrial, Universidad Nacional de Colombia. Docente, Universidad Distrital Francisco José de Caldas, Bogotá, Colombia. Correo electrónico: drairan@udistrital.edu.co.

[5] Estudiante de Tecnología en Electricidad, Universidad Distrital Francisco José de Caldas, Bogotá, Colombia. Correo electrónico: gice@udistrital.edu.co.

[6] Estudiante de Tecnología en Electricidad, Universidad Distrital Francisco José de Caldas, Bogotá, Colombia. Correo electrónico: gice@udistrital.edu.co.

Resumen

En este artículo se implementa el algoritmo de aprendizaje *backpropagation* en línea para el entrenamiento de redes neuronales tipo *feedforward*. Así mismo, se implementan tres neurocontroladores para tres sistemas: el circuito resistivo capacitivo (RC), un motor de corriente continua emulado electrónicamente y un sistema esfera-tubo. La primera estrategia que se prueba para todos los sistemas es un controlador Proporcional Integral Derivativo (PID clásico), utilizado para comparar el desempeño de los otros controladores. El primer neurocontrolador comparte la responsabilidad de comandar al sistema con un PID; el siguiente es entrenado en línea y trabaja solo; el último es un controlador PID neuronal que cambia las ganancias del PID para hacerlo adaptable a la dinámica de la planta. El control se realiza en tiempo real, por medio de Simulink, junto con una tarjeta de adquisición de datos PCI 6024E. En el artículo se muestran los resultados obtenidos de cada sistema.

Palabras clave

Redes neurales, sistemas de control en tiempo real, sistemas híbridos, propagación inversa (inteligencia artificial).

Abstract

In this paper we develop a back-propagation learning algorithm for feedforward neural networks trained online. Three neurocontrollers are designed for three systems. Those systems are an RC circuit, a DC motor (electronically emulated) and a sphere-tube system. The first implemented strategy is a standard PID controller, which is used in order to compare the performance of the neurocontrollers. The first neurocontroller leads the system in parallel with a PID; the next one is trained online to work alone, and the last one is a neural PID, which strives to make the controller adaptable to the dynamics of the plant trough changes on the PID gains. The control is carried out in real time by using Simulink and a PCI 6024E data acquisition card. The results for each system are also included.

Key words

Neural networks, real-time control, hybrid systems, back propagation (artificial intelligence).

Resumo

Neste artigo implementa-se o algoritmo de aprendizagem *backpropagation* em linha para o treinamento de redes neurais tipo *feedforward*. Assim como, programam-se três neurocontroladores para três sistemas: o circuito resistivo capacitivo (RC), um motor de corrente contínua emulado eletronicamente e um sistema esfera-tubo. A primeira estratégia é testada para todos os sistemas é um controlador Proporcional Integral Derivativo (PID clássico), utilizado para comparar o desempenho dos outros controladores. O primeiro neurocontrolador compartilha a responsabilidade de comandar o sistema com um PID; o seguinte é treinado em linha e trabalha sozinho; o último é um controlador PID neural que muda os ganhos do PID para fazê-lo adaptável à dinâmica da planta. O controle realiza-se em tempo real, por meio de Simulink, junto com um cartão de aquisição de dados PCI 6024E. No artigo mostram-se os resultados obtidos de cada sistema.

Palavras chave

Redes neurais, sistemas de controle em tempo real, sistemas híbridos, propagação inversa (inteligência artificial).

Introducción

En su forma más básica, una red neuronal artificial está compuesta por neuronas y conexiones entre neuronas. El trabajo de cada neurona es comprimir el valor de todas las entradas en un único valor de salida, por medio de lo que se conoce como función de activación. A la conexión entre cada par de neuronas se le asocia un peso, el cual puede indicar el grado de importancia que tiene esa conexión en comparación con las demás.

Hay tres características que hace a las redes neuronales especiales: 1) las funciones de activación son no lineales; 2) por la forma en que se conectan, conocida como la estructura, realizan computación en paralelo, y 3) existen algoritmos, llamados de entrenamiento, mediante los cuales se ajustan los pesos para que la red lleve a cabo una tarea determinada. Con toda seguridad, la estructura de red más popular es la conocida como *feedforward*, del inglés relacionado con conexiones hacia delante. Esta estructura se caracteriza por la configuración en capas, una de entrada; cero, una o más capas intermedias, y una capa de salida. La información fluye de la entrada a las capas intermedias para, finalmente, generar una o varias salidas en la capa de salida. En cada capa, excepto en la de entrada, existe un número de neuronas, cada una de las cuales recibe conexiones, ponderadas por pesos, de la capa anterior. Las conexiones son procesadas por las funciones de activación, y el resultado de cada neurona pasa a todas las neuronas de la capa siguiente.

Las características ya mencionadas de las redes neuronales han hecho que estas sean atractivas para proponer soluciones inteligentes en control. Por ejemplo en (Weerasooriya y El-Sharkawi, 1993) se utiliza una red tipo perceptrón multicapa para controlar un motor de corriente continua (DC), el cual vale la pena mencionar, dado que ese motor es una de las plantas que se trabaja aquí. En (Rubaai et ál., 2001) controlan un motor de inducción y utilizan un algoritmo de aprendizaje en línea; pero no para hacer el control, sino para la construcción del modelo de la planta (mediante cinco redes neuronales que trabajan en

paralelo), el cual se requiere para regular corriente con un control adaptable. Un trabajo más está en (Kadwane et ál., 2006), en el cual controlan un conversor de voltaje con una red neuronal artificial, entrenada en línea, utilizando Simulink. Si bien los resultados son interesantes, no son concluyentes en cuanto al rendimiento de ese controlador en otras plantas, por ejemplo, las utilizadas en este artículo.

Un trabajo más, dedicado a exponer las ventajas de los neurocontroladores en línea, está en (Noh, Lee y Jung, 2008). Sus autores proponen utilizar redes neuronales de base radial para controlar la trayectoria de un péndulo en movimiento, en el cual el objetivo es mantener el equilibrio, mientras se traza una trayectoria circular. Así mismo, con respecto al control de temperatura, esta vez en (Hedjar, 2007), se prueban configuraciones con redes neuronales para identificar y controlar, con resultados satisfactorios. Este trabajo es interesante también porque, además de las redes con conexiones hacia delante, utiliza redes recurrentes, como se hará hacia el final de este artículo.

La organización de este artículo es la siguiente: en la primera sección se trata el entrenamiento en línea de una red neuronal artificial; en la siguiente se explican tres estrategias de control neuronal; luego se presenta la implementación de los tres controladores sobre dos sistemas, un circuito RC y un motor DC; enseguida se trabaja con un sistema de complejidad alta (el sistema esfera-tubo), y finalmente están las conclusiones y recomendaciones.

1. Entrenamiento en línea de una red neuronal

El algoritmo de entrenamiento para las redes con conexiones hacia delante es conocido como *backpropagation*, del inglés para propagación hacia atrás. El algoritmo tiene dos partes: en la primera, cada grupo o vector de entradas es presentado a la red, y esta, usualmente inicializada con pesos aleatorios, genera algunas salidas. Estas salidas de la red se comparan con las salidas que se espera que la red asocie a esas entradas, y se calcula la diferencia o error. Enseguida se presenta el segundo vector de entrada, también conocido como ejemplo; nuevamente se propaga la red, y se evalúa el error entre la salida de la red y las salidas que se esperarían. El error debido a todos los ejemplos es acumulado, y se repite esto con todos los ejemplos que incluya el conjunto de datos que se tenga.

En la segunda parte, con base en el error acumulado, se empiezan a cambiar los pesos de cada conexión en la capa de salida, de manera que el error disminuya —para mayores detalles respecto al algoritmo de aprendizaje ir a (Kadwane et ál., 2006)—. Luego se cambian también los pesos de todas las otras capas.

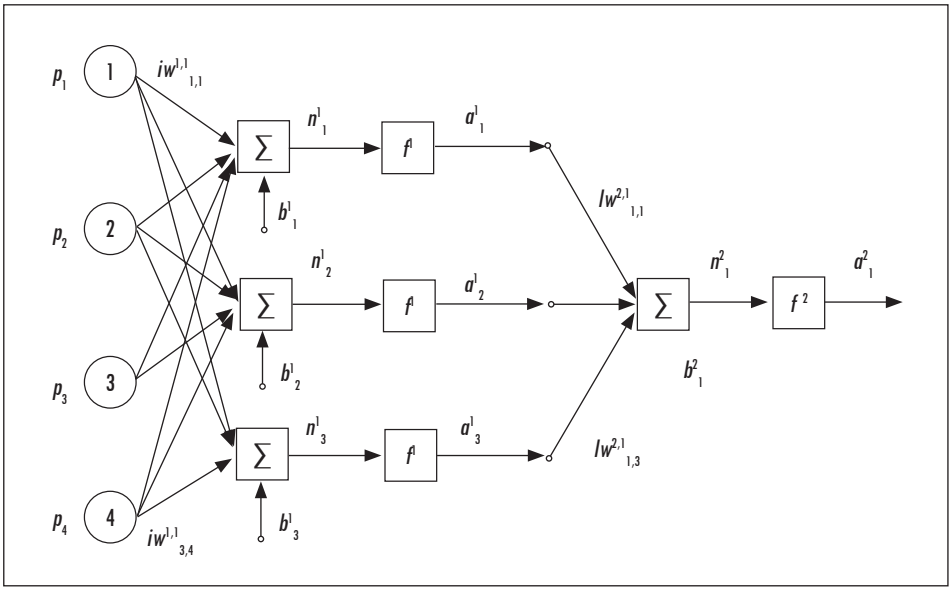
Cuando se han realizado estas dos partes, se dice que se ha implementado una época, luego de lo cual se repite la parte uno, luego la dos, y se repite el procedimiento por tantas épocas como sea necesario, hasta que el error sea menor o igual a una cota que se considere adecuada.

El proceso ya descrito se conoce como *entrenamiento en lote*, o fuera de línea, porque se tienen todos los ejemplos y se dispone de tiempo para ajustar los pesos por medio del algoritmo de aprendizaje, y una vez se considera que la red tiene el conjunto de pesos adecuado, se detiene el entrenamiento y se utiliza la red. En este artículo se trabaja un problema relacionado, pero más difícil, pues es obligatorio que el aprendizaje se haga en línea. La red neuronal debe aprender, debe hacerlo rápido y, además, debe hacerlo a partir de un ejemplo por época.

En su forma más simple, lo que la red aprende es la dinámica inversa de la planta, como en (Nouri et ál., 2006). Los ejemplos que se le presentan a la red, compuestos por duplas de datos correspondientes a la entrada y la salida de la planta, se intercambian, de tal manera que la entrada de la red es la salida de la planta y la salida de la red es la entrada de la planta. En general, en la capa de entrada de la red pueden tenerse n conexiones; pero para simplificar la explicación en la figura 1 se suponen solo cuatro: p_1 a p_4.

Entre la capa de entrada y la primera capa oculta existen conexiones entre todas las entradas y todas las neuronas, y cada conexión tiene asociado un peso, $iw^{1,1}$, denotado así por enlazar la capa de entrada con la primera capa oculta. Por otra parte, cada peso tiene un subíndice, el cual indica la neurona destino y la entrada conectada por la conexión. Para el ejemplo, en la figura 1 uno de los subíndices es 3,4. Cada neurona tiene un valor de ajuste: por ejemplo, en la primera neurona es b^1_1, lo cual indica que es la primera neurona en la primera capa oculta.

Figura 1. Estructura de una red neuronal artificial *feedforward* con una capa de entradas



Fuente: presentación propia de los autores.

La entrada de la función de activación de la capa oculta 1, es decir f^1, está en la ecuación 1.

$$n^1_1 = \sum_{R=1}^{4} iw^{1,1}_{1,R} * P_R + b^1_1 \quad (1)$$

Por lo general, la función de activación es sigmoidea, aunque en realidad puede ser cualquier función diferenciable:

$$a^1 1 = f^1 \left(\sum_{R=1}^{4} iw^{1,1} 1, R * P_R + b^1 1 \right) \quad (2)$$

La salida de la capa oculta es llamada a^1_1, como se indica en la figura 1. Este valor es una de las entradas para la capa de salida, en la cual se suman las contribuciones de las neuronas de la capa oculta, como se indica en la ecuación 3, y finalmente se calcula la salida, utilizando f^2.

$$n^2_1 = \sum_{S=1}^{3} lw^{2,1}_{1,S} * a^1_S + b^2_1 \quad (3)$$

$$a^2_1 = f^2 \left(n^2_1 \right) \quad (4)$$

En la ecuación 4, a^2_1 es la salida de la red, la cual se compara con la salida deseada, y_d, para determinar la desviación entre la red y la salida esperada.

$$e = \frac{\left(y_d - a^2_1\right)^2}{2} \tag{5}$$

En la ecuación 5, e es el error cuadrático medio, según (Kadwane et ál., 2006), el cual se busca minimizar mediante el cambio de los pesos de la red. En ese proceso de minimización, las variables δ^1 y δ^2 indican la contribución de cada peso al error general, según se muestra en las ecuaciones 6 y 7.

$$\delta^2_1 = \left(y_d - a^2_1\right) * f^{2'}\left(n^2_1\right) \tag{6}$$

$$\delta^1_1 = f^{1'}\left(n^1_1\right) * \delta^2_1 * lw^{2,1}_{1,1} \tag{7}$$

$f^{1'}$ y $f^{2'}$ son las derivadas de las funciones de activación. Finalmente, el valor de los pesos corregidos es el que se indica en las ecuaciones 8 y 9.

$$lw^{2,1}_{1,1}(t+1) = lw^{2,1}_{1,1}(t) + \alpha * \delta^2_1 * a^1_1 \tag{8}$$

$$iw^{1,1}_{1,1}(t+1) = iw^{1,1}_{1,1}(t) + \alpha * \delta^1_1 * P_1 \tag{9}$$

La tasa de aprendizaje, α, es la encargada de definir la velocidad de aprendizaje. Para mayor detalle sobre la deducción del algoritmo *backpropagation* en línea ir a (Kadwane et ál., 2006).

2. Tres configuraciones para el controlador

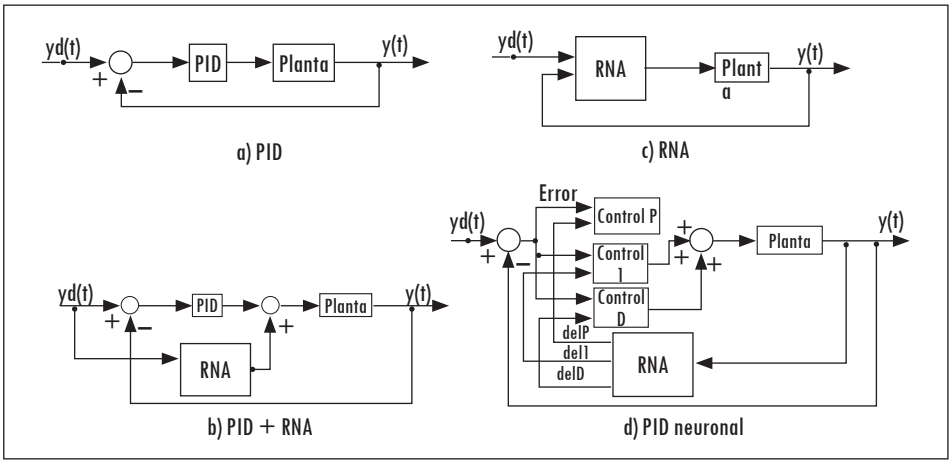
En esta sección se explican las tres configuraciones que se probarán. Estas no son escogidas al azar, sino que responden a un proceso heurístico de diseño de controladores, en el cual se parte de una configuración conocida, y se incrementa el grado de complejidad de la arquitectura del controlador, hasta alcanzar el objetivo del diseño. Este incremento en la complejidad también se verá en el tipo de planta que se va a controlar, con tres casos igualmente, como se explica en la sección 3, a lo cual se adiciona un controlador Proporcional Integral Derivativo (PID) clásico, utilizado como base de comparación.

De esta manera se tienen doce experimentos, mediante los cuales se busca conocer el comportamiento de las plantas y lograr el patronamiento de los controladores. La primera estrategia es un lazo de control, comandado por un PID clásico, como se observa en la figura 2a. En la segunda estrategia, en la figura 2b, se incluye el neurocontrolador, pero la responsabilidad de comandar la planta es

compartida con un PID. En la figura 2c está la tercera opción: una red neuronal entrenada en línea para minimizar el error, entendido este como la diferencia entre la salida deseada *yd(t)* y la que se tiene *y(t)*. Aplicaciones similares se pueden ver en (Guo et ál., 2007; Kamalasadan y Ghandakly, 2007).

La cuarta configuración fue denominada PID neuronal, como en (Omatu, 2009), donde se aplicó al control de un vehículo. En esta las ganancias *Kp*, *Ki* y *Kd* del PID convencional son reemplazadas por variables que dependen del error, según se indica en la figura 2d. El propósito de esta última prueba es utilizar los mejores resultados del control PID clásico (entre los que se está el tener en cuenta el valor actual, la historia y una predicción del error para controlar), y combinar esos resultados con las ventajas de control neuronal en línea.

Figura 2. Estrategias de control: a) PID, b) PID + RNA, c) RNA y d) PID neuronal



Fuente: presentación propia de los autores.

El cambio de las ganancias del PID neuronal consiste en definir incrementos (deltas) de ganancia, las cuales son la salida de la red neuronal, y esos valores se suman a las ganancias actuales en cada instante, como se bosqueja en la figura 2d. Es importante anotar que la red neuronal ajusta sus pesos internos en función del error, por el método conocido como *backpropagation* en línea, explicado en la sección anterior. Este algoritmo permite que el proceso de ajuste de las ganancias se repita incluso hasta mil veces por segundo.

3. Implementación de los controladores

En esta sección se implementan las estrategias de control descritas en la sección anterior. Primero en un circuito eléctrico RC y, enseguida, en un circuito doble

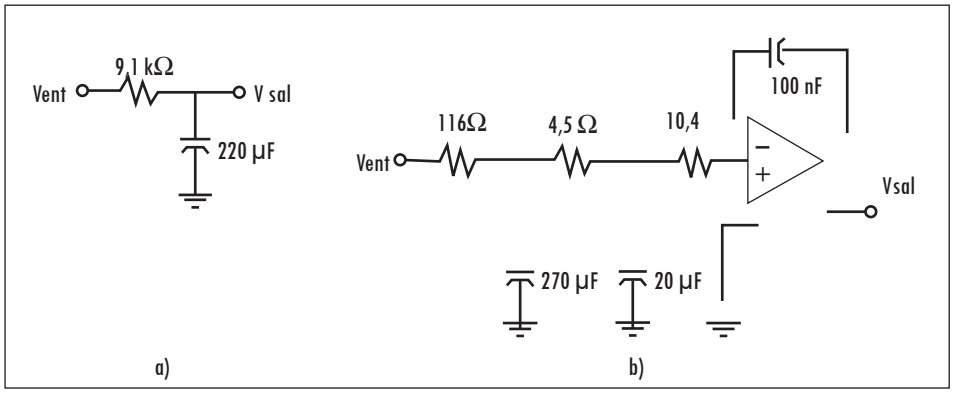
RC en cascada, acoplado con un amplificador operacional, lo cual genera una función de transferencia equivalente a la de un motor DC.

3.1. Circuito resistivo capacitivo

La primera opción cuando se prueba una estrategia de control es implementarla sobre un sistema lineal de orden uno. El hecho de que este tenga solo un polo, estable, hace que sea fácil dirigir su dinámica hacia cualquier región del plano S. La opción más generalizada para controlar un sistema lineal consiste en interconectar el sistema con un sensor y un controlador, para que al formar un lazo cerrado, el sistema en su conjunto tenga los polos y los ceros que se requieran para alcanzar la dinámica deseada.

El sistema de orden uno pudo haber sido emulado de muchas formas, y de ellas se prefirió la implementación física, debido a que las ecuaciones son solo abstracciones matemáticas del comportamiento físico y, por lo tanto, siempre dejan algo de lado, lo cual puede ser importante para la red neuronal con aprendizaje en línea. El sistema de orden uno utilizado es un circuito eléctrico RC serie, como en la figura 3. La selección de R y C define la constante de tiempo. En este caso se seleccionó una constante de tiempo de 2 s.

Figura 3. Circuitos eléctricos: a) circuito RC y b) circuito equivalente a un motor D

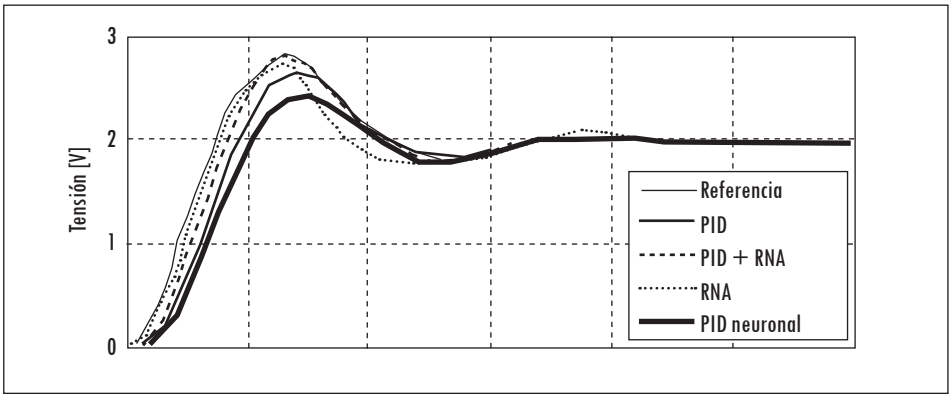


Fuente: presentación propia de los autores.

En la figura 4 se presenta el resultado del control del circuito RC. La referencia es la salida de un circuito de orden dos, con coeficiente de amortiguamiento de 0,5 y, por ende, oscilatorio. La primera prueba se efectuó con un PID con ganancias $Kp = 10$, $Ki = 5$, $Kd = 0{,}1$. La red neuronal para la segunda prueba, como indica la figura 2b, tiene cinco entradas: tres neuronas en cada una de las

dos capas ocultas, y una neurona lineal de salida, la cual es combinada con el PID ya definido. En la tercera prueba se deja a la red neuronal sola para que trabaje como controlador, según se muestra en la figura 2c.

Figura 4. Respuesta del circuito RC ante las cuatro estrategias de control



Fuente: presentación propia de los autores.

En el caso del controlador neuronal, RNA en la figura 4, se utilizó una tasa de aprendizaje variable *t.a(t)*, de la forma *t.a(t)* = 9,9exp(-t/2,8) + 0,1. Esto permite que el algoritmo aprenda la dinámica del sistema cuando la tasa es alta; y deja que la neurona siga aprendiendo, aunque más lento, cuando la tasa es baja. Una ventaja adicional de la tasa pequeña es que se reduce el riesgo de oscilaciones, las cuales pueden presentarse por búsquedas inestables alrededor de mínimos locales por parte del algoritmo de aprendizaje.

Además, se analizó la respuesta ante referencias tipo escalón y senoisal. También se experimentó con señales de referencia aleatorias y se estudió la respuesta ante disturbios, entendidos como señales aleatorias de voltaje en algunos puntos del circuito. En todos los casos se llegó a resultados satisfactorios.

$$H(S) = \frac{Vp(S)}{Vf(S)} = \frac{253.782}{S \cdot (S^2 + 41 \cdot S + 263)} \tag{10}$$

El hecho de que las estrategias de control logren su objetivo con un circuito de orden uno no es un logro muy grande, pero sí indica que se va por buen camino. Para incrementar la complejidad del problema, en esta sección se implementa una planta con un polo inestable, en el origen, más dos polos estables, como

se presenta en la ecuación 10. El modelo equivale al cambio de posición de un motor de corriente directa, como se explica en (Rairán y Fonseca, 2011).

3.2. Control de posición de un motor DC

El denominador de la función de transferencia fue obtenido mediante la herramienta de Matlab para identificación de parámetros aplicada a un motor de imán permanente de marca Faulhaber con velocidad sin carga de 8100 revoluciones por minuto (rpm), el cual indica constantes de tiempo de 30 ms y 125 ms. El valor del numerador fue seleccionado para que una entrada tipo escalón de amplitud 10 produzca una salida de 360 en un tiempo aproximado de 210 ms, como sería en el motor real para la entrada en voltios y la salida en grados, respectivamente, cuando la velocidad inicial es cero. El circuito eléctrico que emula la función de transferencia en la ecuación 10 está en la figura 3.

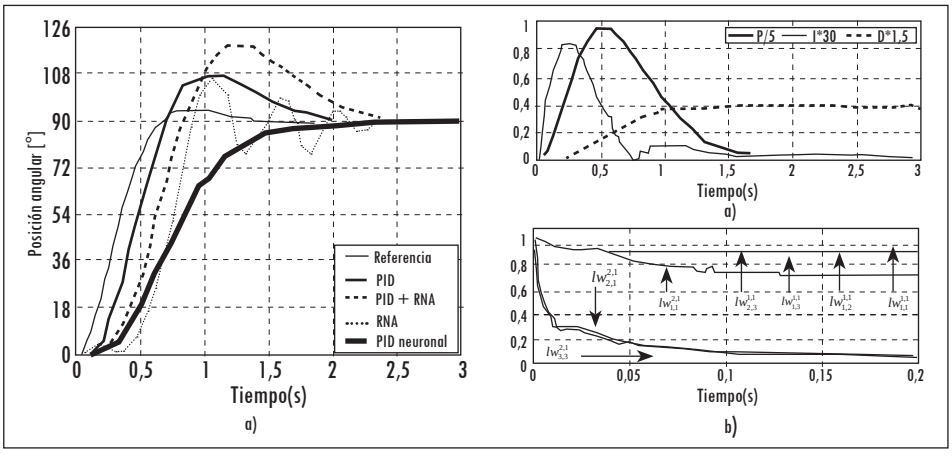
El orden de los experimentos es el indicado en la figura 2. Primero se sintonizó un controlador PID, por medio de Sisotool en Matlab, utilizando como modelo de la planta el resultado de un proceso de identificación, el cual se alimentó con datos de salida del circuito ante entradas aleatorias en amplitud y duración. Con el controlador PID se logró el comportamiento indicado en la figura 5, que presenta un sobrepico menor al 30% y un tiempo de estabilización de alrededor de 3 s. Las ganancias son $Kp = 5$, $Ki = 6$, $Kd = 0$. En la siguiente prueba, es decir PID + RNA en la figura 5, se diseñó una red neuronal con cinco entradas, dos capas ocultas y una capa de salida. Las funciones de transferencias son tangentes sigmoidales, excepto en la capa de salida, donde es una línea con pendiente unitaria. Las entradas de la red son: el instante actual y el anterior de la señal de referencia; el valor anterior y diez veces anterior de la salida de la planta, y la variable de control, filtrada con una función de transferencia de orden uno con constante de tiempo de 0,2 s, lo cual se hizo para reducir oscilaciones.

Como tercer experimento, la red neuronal anterior se deja sola para que controle el sistema, el resultado está en la figura 5a, bajo el título RNA. El sistema presenta oscilaciones alrededor del punto de equilibrio —una indicación de que puede hacerse más trabajo en la definición de esa red—. La experimentación demostró que la tasa de aprendizaje es determinante en el algoritmo. Los mejores resultados se logran con la aplicación de reglas empíricas consignadas en reglas difusas, en las cuales el valor de la tasa depende de la diferencia entre la referencia y la salida del sistema, o error. Esta tasa va de 0,5 a 0,001. A mayor error mayor tasa. Esta lógica permite que cuando se presenta un incremento de error, los pesos de la neurona cambien con rapidez para que el sistema se

reajuste, y cuando se fija un valor bajo en la tasa es porque el error está dentro de un rango que se supone aceptable, para este caso dentro del 1% respecto al valor de referencia.

El último controlador en esta sección es denominado PID neuronal. La red tiene tres entradas: la referencia actual, la anterior y el valor anterior de salida del sistema controlado. Con base en los datos de entrada se incrementa o reduce el valor de las ganancias, las cuales multiplican el error, la acumulación del error y la tendencia de cambio del error. Esta estrategia presenta la curva más rápida y sin sobrepico, como se observa en la figura 5a. En la figura 5b se presenta la evolución de las ganancias y se muestra la forma particular como cambian algunos de los pesos dentro de la red neuronal. En la parte *i* se observan las ganancias del controlador PID neuronal escalizadas por objeto de comparación; en la parte *ii* está la variación individual de algunos de los pesos en la red neuronal que forma parte del controlador PID neuronal. La notación utilizada para la identificación de los pesos corresponde a la explicada en la figura 1.

Figura 5. Respuesta de los controladores: a) posición de un motor DC y b) variación de los pesos



Fuente: presentación propia de los autores.

4. Sistema esfera-tubo

A diferencia de los anteriores, este sistema —también conocido como levitador neumático— no es la emulación eléctrica de una planta, sino que es una planta real, construida para verificar la efectividad de las estrategias de control explicadas en la sección 2. Este ajusta la velocidad de un ventilador, y como resultado

se obtiene un flujo de aire, que eleva y mantiene una esfera liviana en un nivel determinado. Por lo general, la esfera está confinada en un tubo, lo que le da el nombre a la configuración (Ouyang et ál., 2007).

A continuación se expone el montaje y se describe el proceso de linealización de los sensores, debido a que durante el desarrollo del trabajo se observó que el control depende en buena medida de la calidad de la medición y se muestran los resultados más sobresalientes.

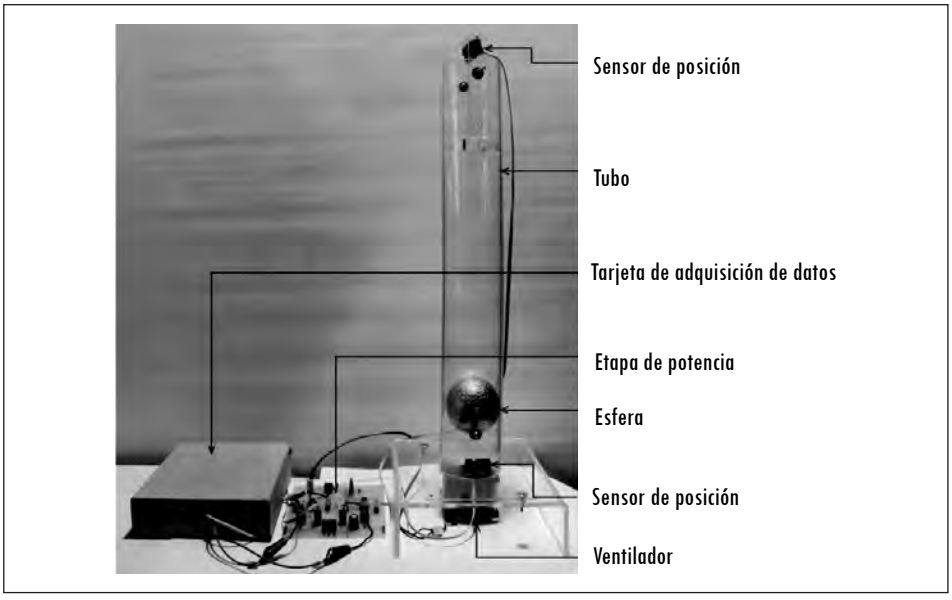
4.1. Descripción del montaje

En la planta hay tres componentes principales: un ventilador, un tubo y una esfera. El ventilador genera una corriente de aire para elevar una esfera de icopor. Con el fin de simplificar el control, ese aire es confinado en un tubo. Así, la tarea de control se reduce a ajustar la velocidad del ventilador para que la esfera no caiga ni se pegue arriba, sino que se mantenga a cierta altura, llamada referencia.

Además de los tres componentes básicos, el montaje requiere sensores para medir la posición de la esfera. Si bien uno sería suficiente, en este trabajo se utiliza uno en la parte superior (con referencia comercial GP2D12) y otro en la inferior (GP2D120). El resultado es que la ventana de observación es más amplia, pero el precio es que la combinación de las dos lecturas exige un trabajo especial, como se explica en la sección siguiente. En la figura 6 puede verse la ubicación de los sensores y del resto del montaje. Un componente más de la planta es la etapa de potencia. Esta recibe una señal proveniente de la tarjeta de adquisición de datos y la amplifica en potencia, para hacer girar el motor del ventilador. La salida de la tarjeta es un tren de pulsos, que alimenta al puente H con referencia L298N. Este puente puede manejar hasta 4 A.

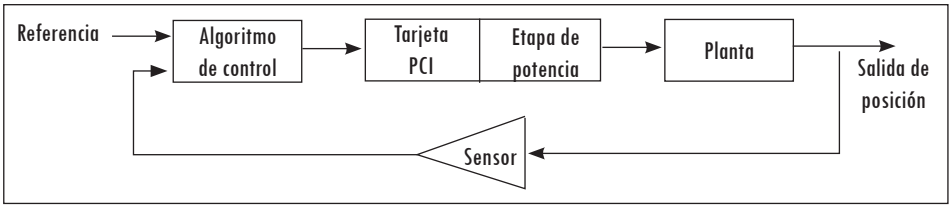
La parte de programación del montaje se hizo en Simulink por dos razones: permite la ejecución del algoritmo de control en tiempo real y es versátil en el manejo de bloques. Esto último facilita la implementación de la red neuronal con aprendizaje en línea. De acuerdo con el esquema en la figura 7, el primer bloque es el algoritmo de control, el cual puede ser PID, PID + RNA, RNA, PID neuronal, según se ha trabajado en todo el artículo.

Figura 6. Montaje de laboratorio del sistema esfera-tubo



Fuente: presentación propia de los autores.

Figura 7. Esquema general de control



Fuente: presentación propia de los autores.

La conexión entre la parte física y la de programación es una tarjeta de adquisición de datos de National Instrumentes, la PCI-6024E, que cuenta con entradas y salidas análogas y digitales. El computador en que se trabajó fue un Intel Pentium 4, de 3 Ghz, con 512 Mb de RAM, lo cual permite un tiempo de discretización de 1 ms. Este tiempo resulta adecuado para la aplicación.

4.2. Linealización de los sensores

En esta sección se explica el proceso de adecuación y linealización de la lectura de los sensores que se utilizaron para medir la posición de la esfera. El problema inicial fue el rango de medida de los sensores comerciales, dado que en el mejor

de los casos este llegó a 14 cm, mientras se requieren 24 cm. La solución fue unir la lectura de un sensor en la parte superior con la lectura de otro en la parte inferior.

La adecuación de la lectura de los sensores se realiza en tres etapas: 1) se comienza con un escalamiento, realizado con amplificadores operacionales; 2) enseguida se utiliza una red neuronal, para linealizar los datos, y 3) se integran las dos lecturas. La primera etapa, el escalamiento, consiste en ajustar la lectura de manera que el mínimo corresponda a cero voltios, y el máximo a algún valor cercano a diez voltios, como se observa en la figura 8.

En la etapa de linealización, dos redes neuronales con aprendizaje supervisado son alimentadas con el voltaje escalizado. Las redes son entrenadas para que sus salidas correspondan a la posición real de la esfera, desde arriba y desde abajo, según corresponda. Las redes son de tipo *feedforward*, con una única capa oculta, con treinta neuronas con función de transferencia tangente sigmoidal y una salida con función lineal pura. El algoritmo de entrenamiento es Levenberg-Marquardt, conocido en Matlab como *trainlm*, el cual es fuera de línea, a la manera que se expone en (Rairán et *á*l., 2009).

Es importante mencionar que en la generación de datos de entrada y salida para el entrenamiento de las redes se tomaron lecturas cada centímetro, las cuales fueron interpoladas por medio de una ecuación polinomial de orden seis, de la cual se obtuvieron 1800 duplas entrada/salida para el sensor GP2D120 y 1700 para el GP2D12. Así, un número típico de épocas durante el entrenamiento fue 1000.

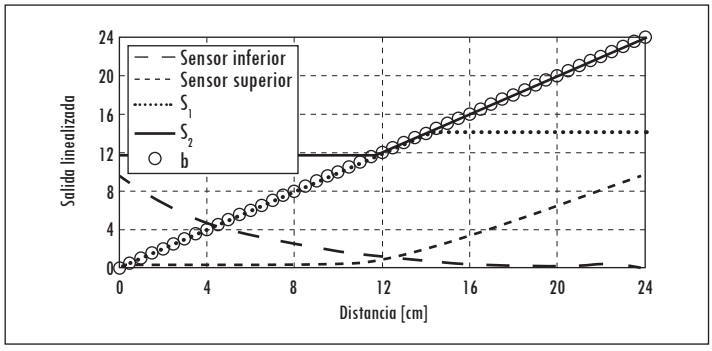
La tercera etapa es la integración de las lecturas, la cual se rige por la expresión siguiente:

$$
\begin{aligned}
b &= c_1 \cdot S_1 + c_2 \cdot S_2 \\
c_1 &= \begin{cases} 1, & S_1 \le 12,75 \\ -S_1 + 13,75, & 12,75 < S_1 < 13,75 \\ 0, & S_1 > 13,75 \end{cases} \\
c_2 &= \begin{cases} 0, & S_2 \le 12,75 \\ S_2 - 12,75, & 12,75 < S_2 < 13,75 \\ 1, & S_2 > 13,75 \end{cases}
\end{aligned}
\qquad (11)
$$

La salida integrada es la señal b; la salida linealizada del sensor en la parte inferior es S_1, y en la parte superior es S_2; c_1 y c_2 son los parámetros que sirven para efectuar la integración.

Siempre y cuando la lectura del sensor inferior sea inferior a 12,75, entonces la salida integrada, b, es igual a S_1. Cuando la lectura del sensor superior es mayor a 13,75, entonces b es igual a S_2. Para valores intermedios se realiza una ponderación lineal. El resultado se observa en la figura 8.

Figura 8. Linealización de los sensores: se linealiza la región con mayor variación en cada sensor



Fuente: presentación propia de los autores.

5. Implementación de los controladores

El levitador neumático es un problema difícil de solucionar, debido a los efectos no lineales que rigen su dinámica. Por ejemplo, experimentalmente se verificó que si las ganancias del controlador son fijas, entonces la respuesta del sistema es distinta en función de la altura de la esfera. Esto significa que un controlador con ganancias que se adapten puede resultar en un control más adecuado, lo cual es una de las motivaciones más fuertes para utilizar la red neuronal con aprendizaje en línea en el control de esta planta.

Una dificultad adicional es el tiempo de respuesta del sistema, el cual se expresa como el retardo entre el momento cuando se indica una referencia hasta cuando la planta responde en realidad. Por ejemplo, el retardo desde cuando se da la orden al ventilador para que gire al momento cuando ya existe un flujo de aire proporcional al comando. Además, existen dificultades durante la construcción, como el flujo turbulento que genera el ventilador y la tendencia de la esfera a girar, según describe la ley de Bernoulli y el efecto Coandă (Popescu et ál., 2009).

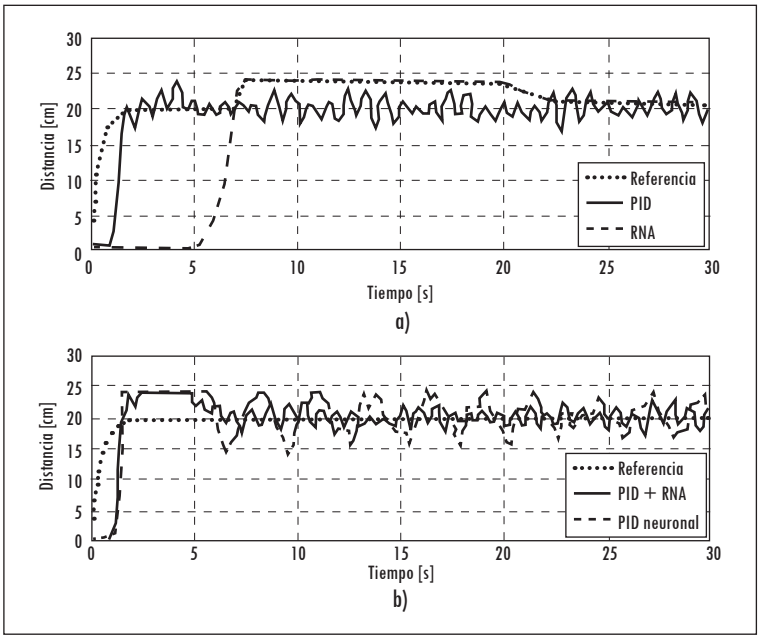
En la figura 9 se observa el comportamiento de las cuatro estrategias utilizadas a lo largo del artículo, pero esta vez controlando el levitador a una referencia de 20 cm. El controlador que presenta la oscilación menor es el PID + RNA, aunque el tiempo para llegar por primera vez a la referencia es mayor que con

el controlador PID, debido al tiempo que la RNA demora en adaptar sus pesos. Puede decirse que el control PID tiene el segundo mejor comportamiento, en cuanto a la amplitud de las oscilaciones, las cuales deberían ser cero. El mayor inconveniente con el PID, como ya se mencionó, es que la utilidad de este algoritmo depende del punto de operación. La estrategia PID neuronal presenta el tercer mejor comportamiento, aunque se esperaba que tuviera el mejor, según se logró con los dos sistemas en la sección 3.

Las ganancias del PID se definen por ensayo-error, de la experiencia en el manejo de la planta. Las ganancias iniciales para el PID neuronal son las mismas que para el PID. En el caso del controlador neuronal todos los pesos iniciales son uno y luego se les deja variar libremente según diga el algoritmo de aprendizaje. En el PID + RNA el PID tiene las mismas ganancias de los otros casos y la red es iniciada con pesos unitarios. Las estrategias PID + RNA y RNA tienen cinco entradas:

- La referencia actual y la referencia del instante anterior.
- La salida de un instante anterior y de dos instantes antes.
- La salida del controlador del instante anterior.

Figura 9. Comportamiento de la planta ante una señal escalón: a) PID y RNA, b) PID + RNA y PID neuronal



Fuente: presentación propia de los autores.

Esta última entrada convierte a esta configuración en una recurrente. La red cuenta además con dos capas ocultas, una con función de activación *tansig*, y la otra logsig; en cada capa oculta hay tres neuronas. La capa de salida tiene una neurona con función de activación lineal.

Para la estrategia PID neuronal, la red tiene tres entradas: la señal de referencia, la retroalimentación de la planta y, una vez más, la salida de la red. Esta vez se trabaja con una capa oculta con tres neuronas, las cuales tienen función de activación *tansig*. La capa de salida tiene tres neuronas, una por cada ganancia del PID, con función de activación lineal. La tasa de aprendizaje en esta sección fue evaluada a partir de la aplicación de un controlador difuso, lo que hace a esta tasa variable.

La entrada del controlador difuso es generada así: primero se calcula el error medio cuadrático, el cual comprime en un único número las últimas diez diferencias entre la salida deseada y la real; luego, para facilitar la definición de las funciones de membresía, se toma el valor absoluto del error medio cuadrático y se le aplica logaritmo con base 10. Se utilizan cuatro funciones de membresía para la variable de entrada y otras cuatro para la variable de salida. Como resultado se obtiene una relación no lineal, la cual genera una salida que está entre -2 y -6. La tasa de aprendizaje corresponde a diez elevado al valor de la salida del difuso; de esta manera, la tasa de aprendizaje está entre 0,01 y 1e-6. En general, la lógica del controlador difuso consiste en que entre más grande es el error, más grande es la tasa de aprendizaje.

6. Conclusiones

En este artículo se verificó que la red *feedforward* con aprendizaje en línea puede aproximar la dinámica de las tres plantas utilizadas, de tal manera que al conectar cada red con el sistema correspondiente, es posible hacer que cada sistema se estabilice en un valor de referencia dado. Sin embargo, el control del levitador neumático resultó más efectivo al utilizar una conexión recurrente; esto es, con realimentación de la salida de la red a la capa de entrada de la red. Aun así, el número de entradas fue solo cinco. Dos de ellas son realimentaciones de la salida de la planta, otras dos corresponden a la señal de referencia y la última es la conexión recurrente.

El desarrollo de los experimentos demostró que el hecho de estar aprendiendo con base en un único ejemplo, por cada vez que se corre el algoritmo *backpropagation* en línea, en contraste con el aprendizaje por lote o fuera de línea, es compensado de dos maneras: 1) al correrlo cada milisegundo, mientras que las constantes de tiempo de los sistemas están en el orden de los segundos, y 2)

por la tasa de aprendizaje variable. La tasa variable hace que el algoritmo sea rápido en un inicio, que no olvide cuando ya ha logrado aprender lo que se necesita y posibilita el reaprendizaje si las condiciones de operación han cambiado lo suficiente, lo cual hace al algoritmo adaptable.

A partir de la experimentación con las tres plantas no es posible indicar cuál de las tres configuraciones con redes neuronales es la mejor (un PID en paralelo con una red neuronal, una red sola o una red ajustando los pesos de un PID). Sin embargo, en los tres casos sí resultó que el peor comportamiento es el de la red sola, por lo cual puede pensarse que la combinación entre estrategias puede presentar ventajas que no vale la pena desaprovechar en aras de exaltar una estrategia en particular.

Referencias

GUO, C.; SONG, Q. y CAI, W. A Neural Network Assisted Cascade Control System for Air Handling Unit. *IEEE Transactions on Industry Applications*. 2007, vol. 54, núm. 1, pp. 620-628.

HEDJAR, R. Online adaptive control of non-linear plants using neural networks with application to temperature control system. *Journal of King Saud University, Computer & Information Sciences*. 2007, vol. 19, núm. 1, pp. 75-94.

KADWANE, S.; KUMAR, A.; KARAN B. y GHOSE T. Online trained simulation and DSP Implementation of dynamic back propagation neural network for buck converter. *ACSE Journal.* 2006, vol. 6, núm. 1, pp. 27-34.

KAMALASADAN, S. y GHANDAKLY, A. A neural network parallel adaptive controller for dynamic system control. *IEEE Transactions on Instrumentation and Measurement*. 2007, vol. 56, núm. 5, pp. 1786-1796.

NOH, J.; LEE, G. y JUNG, S. Motion control of a mobile pendulum system using neural network. *Advanced Motion Control, 2008. AMC '08. 10th IEEE International Workshop.* 2008, pp. 450-454.

NOURI, K.; DHAOUADI, R. y BRAIEK, N. Nonlinear speed control of a dc motor drive system with online trained recurrent neural network. *9th IEEE international workshop on advanced motion control*. 2006, pp. 704-708.

OMATU, S. Neuro-control and its applications to electric vehicle control. *IWANN'09 Proceedings of the 10th Interntaional Work-Conference on Aritificial Neural Networks: Part II: Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*. 2009, pp. 1-12.

OUYANG, Z.; SCHNELL, M. y WEI, K. The experiment Ball-in-tube with Fuzzy-PID controller based on space. *IEEE International Conference on Systems, Man and Cybernetics*. 2007, pp. 877-881.

POPESCU, M.; BALAS, V.; MUSCA, S. y COSMA, D. Modeling the phenomenon of refreshing air inside the spaces of the Fungi's culture. *ISCIII 2009, 4th International Symposium on Computational Intelligence and Intelligent Informatics*. 2009, Egipto, pp. 85-92.

RAIRÁN, J. y FONSECA, J. Doble lazo de control para regular la posición y la velocidad en un motor de corriente directa. *Ingeniería y Universidad*. 2011, vol. 15, núm. 2, pp. 337-357.

RAIRÁN, J.; PÉREZ, J. y OSORIO, J. Implementación de una red neuronal para la medición indirecta de posición. *Revista Avances en Sistemas e Informática*. 2009, vol. 6, núm. 3, pp. 79-85.

RUBAAI, A.; KOTARU, R. y KANKAM, M. Online training of parallel neural network estimators for control of induction motors. *IEEE Transactions on Industry Applications*. 2001, vol. 37, núm. 5, pp. 1512-1521.

WEERASOORIYA, S. y EL-SHARKAWI, M. Laboratory implementation of a neural network trajectory controller for a DC motor. *IEEE Transactions on Energy Conversion*. 1993, vol. 8, núm. 1, pp. 107-113.