CONTROL DE UN SISTEMA DE SEGUNDO ORDEN BASADO EN REDES NEURONALES

SECOND-ORDER SYSTEM CONTROL BASED ON NEURAL NETWORKS

Mario Borja[1],Rudolph Molero[2], Nilton Cuellar[3], Martin Montes[4], Drago Separovich[4]

RESUMEN

El presente trabajo muestra la simulación e implementación de un "Neurocontrolador" en una planta de segundo orden. El controlador neuronal, también conocido como Neurocontrolador, fue implementado con una red multicapa, donde la retropropagación del error fue desarrollada mediante el algoritmo "Backprogation". La red multicapa, compuesta por una capa oculta y una capa de salida, fue simulada primero en Matlab para conseguir los parámetros de variación, luego fue simulada en Visual C++ para lograr la optimización. La arquitectura de esta red multicapa fue variando muchas veces hasta llegar a una forma óptima que se mostrará como la arquitectura final. Seguidamente, se hizo la simulación en LabVIEW 8.4, corroborando las simulaciones en Visual C++. Finalmente, se probó el controlador neural desarrollado en LabVIEW en tiempo real, mostrando gratificantes resultados y comprobando su efectividad a pesar de cambios simultáneos en los parámetros.

Palabras clave.- Neurocontrolador, Control con redes neuronales, Neurocontrolador por refuerzo, Controlador neuronal.

ABSTRACT

In this work we show the simulation and implementation of a "Neurocontroller" for a second-order plant. The neural controller, more known as Neurocontroller, was implemented in a Multi-Layer Network, where the feedback error was developed using the Backpropagation Algorithm. The Multi-Layer Network, composed by a hidden layer and output layer, was simulated in Matlab and C++ in order to get the variation in the parameters and to reach the optimization, respectively. The architecture of the Multi-Layer Network was changing several times until the optimal structure is found (final architecture). Afterwards, a simulation in LabVIEW 8.4 was performed, corroborating the simulation in Visual C++. Finally, the Neural Network developed in LabVIEW was proved in real time, showing good results and checking its effectiveness in spite of simultaneous changes in parameters.

Key words.-Neuro-controller, Neural network control, Neuro-controller through reinforcement learning, Neural controller.

INTRODUCCIÓN

El control de sistemas en general y el de procesos industriales en particular, constituyen un área de trabajo en plena evolución. Los controladores basados en técnicas "clásicas" llevan mucho

---

[1]M.Sc. Docente investigador de la Facultad de Ingeniería Mecánica de la Universidad Nacional de Ingeniería,
[2]Alumno investigador de la Facultad de Ingeniería Mecánica de la Universidad Nacional de Ingeniería,
[3]Alumno investigador de la Facultad de Ingeniería Mecánica de la Universidad Nacional de Ingeniería,
[4]Alumno investigador de la Facultad de Ingeniería Mecánica de la Universidad Nacional de Ingeniería.

tiempo funcionando en la industria, lo que constituye una prueba evidente de su capacidad y fiabilidad. Sin embargo, estas técnicas, en las que a menudo es necesario disponer de un modelo del proceso a controlar para diseñar el controlador, muestran sus limitaciones cuando se abordan problemas de control con determinadas características frecuentes en los procesos industriales. Entre ellas destacan la imprecisión del modelo disponible, ya sea por motivos de incertidumbre o de complejidad, el incumplimiento de ciertas premisas impuestas por las técnicas "clásicas" de diseño de controladores, como la *linealidad*, y muy habitualmente la variación en el tiempo de la dinámica del proceso.

Por ello, se han venido desarrollando diferentes técnicas capaces de enfrentarse a este tipo de problemas, muchas de las cuales se podrían considerar ya clásicas también, puesto que llevan tiempo dando buenos resultados en la industria. Cabe destacar los métodos de *Control no lineal*, el *Control Robusto* y el *Control Adaptativo*, capaces de solucionar parte de los problemas planteados, y con ciertos inconvenientes en su utilización.

Pero la complejidad de los sistemas a controlar es cada vez mayor, las especificaciones de diseño impuestas al comportamiento del sistema controlado son más exigentes, y el conocimiento previo de la planta y su entorno más impreciso, con un alto grado de incertidumbre. Estas circunstancias llevaron a un replanteamiento de las técnicas de control existentes y a la aparición de otras nuevas en este campo, que podrían enmarcarse dentro del llamado *Control Inteligente*.

Sin entrar en la polémica de las características que debe o no presentar un controlador inteligente, lo que es indiscutible es la necesidad de un alto grado de *autonomía* en los controladores que vayan a enfrentarse a procesos complejos, y esto significa que hay que dotar a los controladores de la facultad de adaptarse sobre la marcha y, sobretodo, de la capacidad de aprender.

Entre las técnicas que se han enmarcado habitualmente en el campo del control inteligente encontramos los sistemas expertos, el control borroso, el control basado en algoritmos genéticos y las redes neuronales. Los dos primeros no presentan en sí las características de autonomía y capacidad de aprendizaje citadas, aunque últimamente las técnicas de control borroso se han mezclado con las redes neuronales o los algoritmos genéticos para diseñar controladores híbridos que sí reúnen esos requisitos [1].

El trabajo realizado en el presente proyecto afronta el problema de diseño de controladores desde la técnica de las redes neuronales artificiales. Entre las ventajas de las redes neuronales podemos citar su arquitectura paralela y distribuida; además de presentar una alta tolerancia a fallas. Pero su gran potencial en el campo del control de procesos se encuentra, por una parte, en su gran capacidad de aproximar casi cualquier función no lineal, lo que permite tanto modelar sistemas muy complejos así como sustituir controladores adecuados para los mismos.

El presente documento científico remarca la capacidad de aprendizaje proponiendo redes neuronales, que basándose en la técnica de Aprendizaje por Refuerzo aprenda a controlar el proceso en marcha (on-line) a partir de su propia experiencia, es decir, sin necesidad de un maestro que instruya sobre las acciones de control a tomar. De este modo se puede obtener un controlador capaz de aprender a controlar un sistema en principio desconocido y de reaccionar, adaptándose a los posibles cambios en la dinámica del proceso a lo largo del tiempo, sin más información que la obtenida de su propia experiencia con el proceso, siguiendo la metodología similar a la conocida en psicología como prueba y error.

DISEÑO DEL CONTROLADOR

Control basado en aprendizaje por refuerzo

En el aprendizaje por refuerzo, no se indica a la red cuáles son las acciones correctas. La idea es dejar que la red perturbe el sistema a controlar y observe el resultado de sus acciones sobre algún índice de resultados (definido en función de los objetivos buscados) que le permita obtener información para su aprendizaje.

El aprendizaje mediante refuerzo es un método de entrenamiento muy general que puede ser aplicado cuando no existe una información detallada sobre la salida deseada, por lo menos para los ejemplos patrones. Dado que existen situaciones donde es posible evaluar los resultados de una acción

ejecutada con un simple valor cualitativo *éxito* o *fracaso*, el modelo pretende añadir una función de referencia la cual está basado en el diseño clásico de controladores. La idea de este modelo es la utilización de una función de referencia que hace de *crítico* en lugar de una información detallada sobre las salidas deseadas.

Cualquier problema de control que requiera aprendizaje se puede formular como un problema de aprendizaje por refuerzo, donde el objetivo será maximizar un índice de mejoras.

Es claro que este método es justificable cuando no exista el conocimiento necesario; pero cuando existe información disponible, es recomendable utilizar métodos especializados de aprendizaje supervisados, para sacarle mayor beneficio al conocimiento disponible.

En la práctica es difícil conseguir una buena función evaluadora. Se suele implementar mediante una o dos redes neuronales que realicen primeramente la selección de una de las posibles acciones dentro del espacio de acciones y finalmente la generación propiamente dicha de la señal de evaluación (refuerzo).

Planteamiento del problema

Se tiene una planta lineal, la cual será controlada con una red multicapa (*1-N-1*). El sistema lineal puede ser un motor DC, o una planta de segundo orden. El problema radica en sintonizar a la red neuronal para que dado un sistema de referencia la planta siga a este modelo bajo un cierto índice de error. Para probar nuestro Controlador Neuronal vamos a simular nuestro sistema de referencia y a nuestra planta de manera que pueda reflejar fielmente las pruebas que se le pudieran someter en tiempo real. Para esto nos vamos a apoyar del Visual C++ y de Matlab, para finalmente implementarlo en el LabVIEW. El modelo de referencia utilizado tiene las siguientes características:

$$\begin{aligned} Mp &= 10\% \\ ts &= 0.1\ s \end{aligned} \qquad (1)$$

Luego el sistema de referencia de segundo orden queda así:

$$Mp = e^{-\frac{\xi * \pi}{\sqrt{1-\xi^2}}} \quad \wedge \quad \omega_n = \frac{5}{ts * \xi} \qquad (2)$$

$$Gr(s) = \frac{\omega_n{}^2}{s^2 + 2 \cdot \xi \cdot \omega_n \cdot s + \omega_n{}^2} \qquad (3)$$

$$\begin{aligned} &\omega_n = 84.58\ rad/s \quad \wedge \quad \xi = 0.5912 \\ &Gr(s) = \frac{out(s)}{u(s)} = \frac{7153.8}{s^2 + 100 * s + 7153.8} \end{aligned} \qquad (4)$$

El sistema de referencia mostrado en (4) es el que se ha usado para realizar el entrenamiento por refuerzo, cabe señalar que para poder usar esta función de transferencia se ha tenido que convertir a espacio de estados.

La transformación en el espacio de estados trae consigo la siguiente expresión:

$$\begin{aligned} \dot{x}_1 &= -100 * x_1 - 7153.8 * x_2 + u \\ \dot{x}_2 &= x_1 \\ out &= 7153.8 * x_2 \end{aligned} \qquad (5)$$

La expresión (5) es posible implementarla a nivel de simulación en Matlab o en Visual C++. Ahora bien, para definir en primera instancia la arquitectura neuronal se utilizó dos neuronas para la entrada y salida, y tres neuronas en la capa oculta, luego se empezaron a realizar las pruebas.

En la Fig. 1 podemos ver la arquitectura o topología de aprendizaje utilizado en el cual se aprecia de qué manera se ha realizado el aprendizaje por refuerzo; nuestra topología cuenta con dos entradas para darle mayor información acerca del desempeño de la red en el tiempo.

Seguidamente, en la Fig. 2 podemos ver el flujo de señales que se dan en la arquitectura propuesta, la nomenclatura de cada nodo se muestra para un mayor entendimiento del funcionamiento de la red.

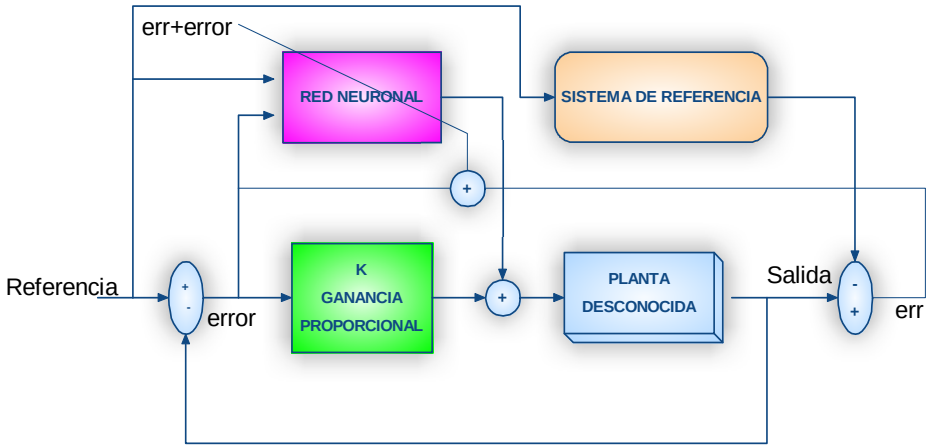

Fig. 1 *Diagrama de bloques del sistema de control neuronal.*

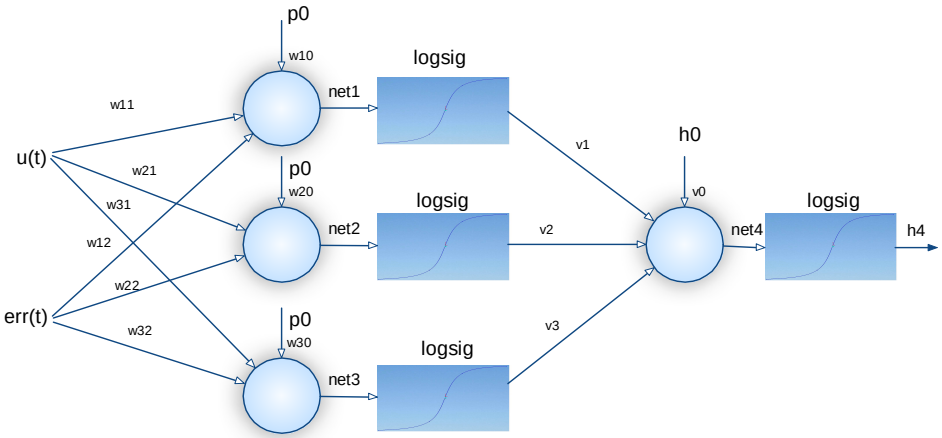

Fig. 2 *Esquema de la arquitectura de red utilizada.*

FUNCIONAMIENTO DE LA ARQUITECTURA

La ganancia proporcional le da ya una estabilidad en lazo cerrado; por tanto, se aleja la idea de divergencia en proceso de entrenamiento. La entrada de referencia alimenta a la red neuronal la cual en base a esa entrada y a unos pesos aleatorios, nos da una respuesta de manera que excita a la planta, dándonos un resultado final el cual es comparado con la referencia. Por otro lado, la entrada alimenta al sistema de referencia dando un resultado, y es comparado con la respuesta de la planta. Por tanto tendríamos dos errores, los cuales se quiere que sean cero, por un lado garantizamos que nuestro sistema llegue a igualarse a la referencia y por otro lado forzamos a nuestra planta a responder como un sistema de referencia y de este forma hacer que nuestra red llegue al objetivo de control bajo una cierta dinámica [2].

A continuación vamos a describir el algoritmo backpropagation:

$$net_i = \sum_{j=1}^{n} w_{ij} * p_j + w_{i0} * p_0 \Big| i = 1,2,3 \qquad (6)$$

$$h_i = \sigma\left(\sum_{j=1}^{n} w_{ij} * p_j + w_{i0} * p_0 \right) \qquad (7)$$

$$\sigma(x) = logsig = \frac{1}{1+e^{-x}}$$

$$net_4 = O_l = \sum_{i=1}^{n} v_{li} * h_i + v_{l0} * h_0 \Big| l = 1 \qquad (8)$$

$$h_4 = O = \sigma\left(\sum_{i=1}^{n} v_i * h_i + v_0 * h_0 \right) \qquad (9)$$

$$\sigma(x) = \text{logsig} = \frac{1}{1+e^{-x}} \quad i = 1,2,3$$

Luego calculamos los errores Feedback

$$\delta_4 = (error^*) * \sigma'(net_4)$$
$$\delta_4 = (error^*) * (1 - h_4) * h_4 \quad (10)$$

Luego el error en cada neurona oculta será:

$$\delta_i = (\delta_4) * v_i * \sigma'(net_i)$$
$$\delta_i = (\delta_4) * v_i * (1 - h_i) * h_i \quad (11)$$
$$i = 1,2,3$$

Calculamos la variación de los pesos para las capas externas:

$$\Delta v_i = \alpha * \delta_4 * h_i \quad (12)$$
$$i = 0,1,2,3$$

De igual manera para las neuronales internas:

$$\Delta w_{ij} = \alpha * \delta_i * p_j$$
$$i = 0,1,2 \quad (13)$$
$$j = 0,1,2$$

Finalmente actualizamos los pesos de todas las neuronas

$$w = w + \Delta w$$
$$v = v + \Delta v \quad (14)$$

PRUEBAS DEL ENTRENAMIENTO

En esta sección vamos a probar nuestro controlador con una planta lineal de segundo orden, bajo un sistema de referencia que se ha detallado en la primera sección.

La planta a usar es un sistema electrónico de segundo orden, el cual fue implementado con dispositivos básicos como OPAM's, resistencias y condensadores; por tanto, el proyecto está basado en el control de un sistema real. El proceso de identificación se ha realizado en Matlab, para lo cual se ha tomado muestras con la tarjeta de adquisición de datos *daqUSB-6008*

En la Fig. 3 podemos apreciar la respuesta del sistema identificado con respecto a lo que se tomo como muestra, note que el seguimiento es impecable; por tanto, éste gráfico valida nuestra identificación, es decir, las simulaciones posteriores que se hagan van a reflejar muy fielmente la respuesta real.

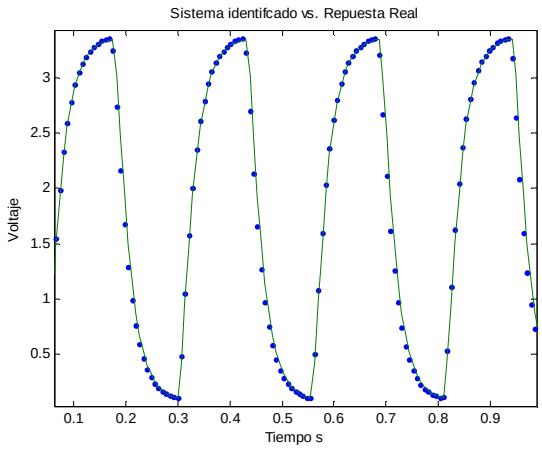

Fig. 3 *Validación del proceso de identificación de la planta.*

En la expresión 15, se aprecia la función de transferencia obtenida luego del proceso de identificación. El proceso se llevó a cabo utilizando el *toolbox* para Identificación de Sistemas de Matlab. El modelo resultante es un sistema de 2do orden.

$$G(s) = \frac{-0.2856 \cdot s + 5980}{s^2 + 192.9 \cdot s + 5993} \quad (15)$$

La expresión 16 muestra la representación en el espacio de estados de la planta utilizada, siendo *out* la salida del sistema, x_1 y x_2, las variables de estado del sistema.

$$\dot{x}_1 = -165.77 * x_1 - 5061.81 * x_2 + uc$$
$$\dot{x}_2 = x_1 \quad (16)$$
$$out = 2.328 * x_1 + 5030.31 * x_2$$

RESULTADOS DE SIMULACIONES EN VISUAL C++

La Fig. 4 muestra la simulación de la respuesta de la planta (onda roja) bajo el efecto del controlador neuronal. El sistema de referencia (verde) en todo momento responde de la misma forma, mientras que la planta por el hecho de tener un carácter cognitivo aprende de sus errores respecto al

sistema de referencia, y a medida que transcurre el tiempo de simulación de su respuesta va mejorando.

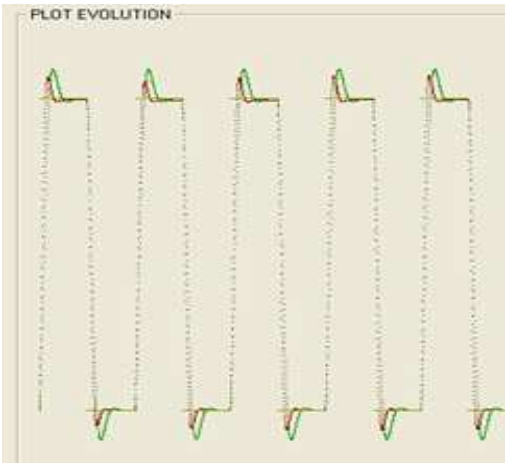

Fig. 4 *Simulación de la respuesta de la planta.*

Variando condiciones de simulación

En las Fig. 5 y 6 podemos ver las simulaciones para un cambio de referencia, es decir, en un momento dado alteramos el sistema de referencia, y vemos como la respuesta de la planta con el controlador neuronal se adapta al nuevo nivel de referencia mejorando su respuesta a medida que transcurre el tiempo de simulación. Este tipo de respuesta adaptativa es propia de los controladores neuronales por refuerzo.

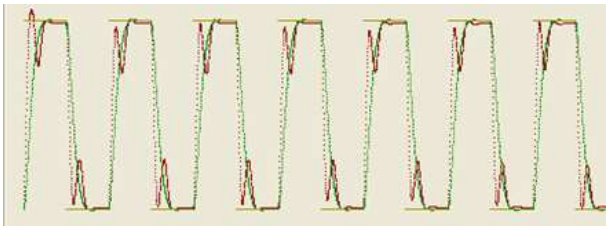

Fig. 5 *Respuesta a cambios en la referencia.*

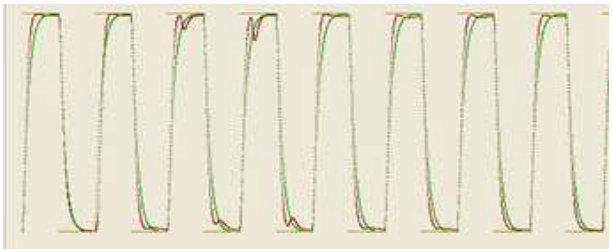

Fig. 6 *Respuesta con mejora en los parámetros de aprendizaje.*

En la Fig. 7, se ha considerado una entrada sinusoidal (señal en verde cactus), mientras que la señal de referencia (señal en verde) sigue a la señal de entrada de acuerdo a las características en el estado transitorio preestablecidas al inicio. La señal de respuesta de la planta (señal roja) sigue al modelo de referencia. El seguimiento de ésta se hace de manera progresiva, es decir, tiene un comportamiento más similar a la referencia a medida que transcurre el tiempo de simulación.

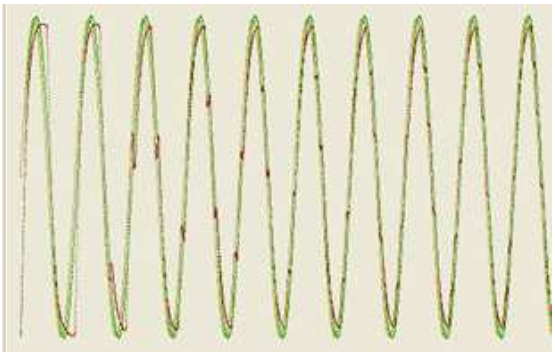

Fig.7 *Respuesta de la planta para una referencia senoidal.*

IMPLEMENTACIÓN EN LABVIEW

La implementación en LabVIEW es básicamente una traducción del código C al lenguaje gráfico de LabVIEW; en consecuencia, los pasos hechos en el código fuente son similares. Las secuencias del algoritmo de *Backprogation* se han hecho con el *Flat Sequence* de LabVIEW.

El proceso se ha separado en cinco etapas:

i. Primera etapa.- se evalúan las entradas por sus respectivos pesos.
ii. Segunda etapa.- se calcula el error feedback de la capa de salida.
iii. Tercera etapa.- se calculan los errores feedback en la capa oculta.
iv. Cuarta etapa.-se evalúan los cambios de pesos.
v. Quinta etapa.-se actualizan los pesos.

Así como se implemento en visual C++, aquí también usamos la representación en el espacio de estados para la planta y para el sistema de referencia. De esa forma los evaluamos en la etapa correspondiente según la simulación previa.

En la Fig. 8, podemos apreciar el *front pannel* del programa principal. En él vemos controles para inicializar parámetros de simulación como para *k*, α, y para los pesos en general. Los pesos han sido inicializados por primera vez como *0.1* o en todo caso números aleatorios, y luego se prosiguió esperar a que el controlador converja a los valores óptimos. Los valores óptimos varían de acuerdo a qué situación corresponde el estado actual del nivel y sistema de referencia. Por eso, cuando se realiza la simulación, estos valores, al llegar la planta al valor deseado, simplemente los pesos dejan de actualizarse, culminando el proceso de entrenamiento.

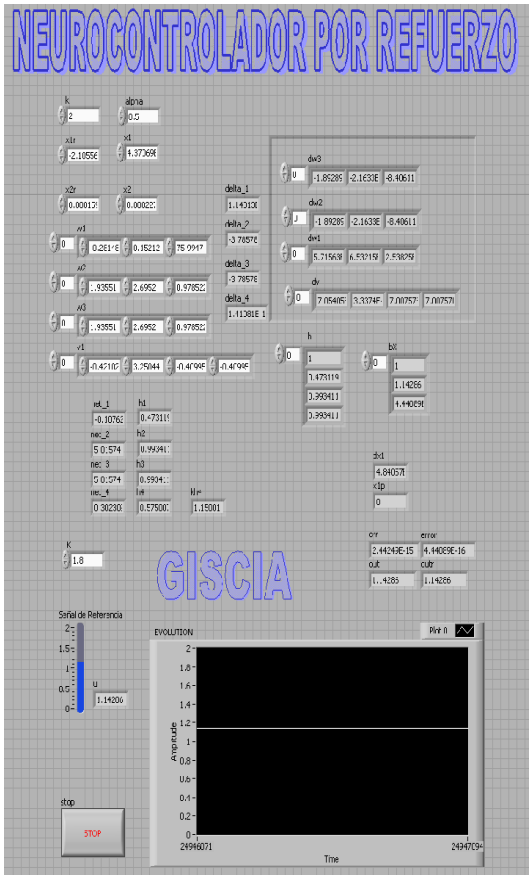

Fig. 8 *Panel frontal del sistema de control en LabVIEW.*

Por otro lado en la Fig. 9 se muestra el programa en sí. Podemos apreciar las cinco etapas programadas. Cada etapa corresponde a cada uno de los procesos de actualización descrito en el algoritmo *backpropagation*. En el *time delay* que se muestra en esta figura se especifica el tiempo de muestreo que se usó para la simulación, correspondiente a 2 kHz.

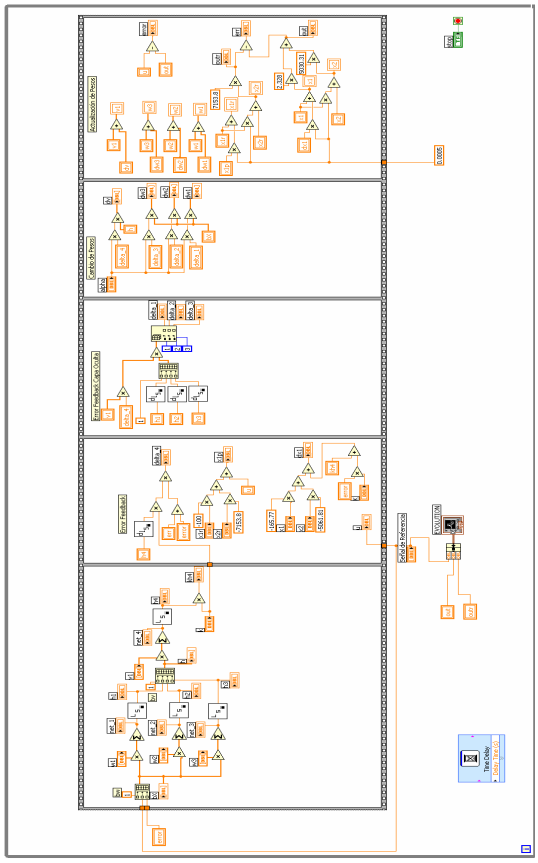

Fig. 9 *Código gráfico del controlador neuronal.*

En la Fig. 10 podemos apreciar el *front pannel* correspondiente a las pruebas reales finales. En el *waveform chart* se muestran los resultados visuales bajo una alteración de los niveles de referencia. Una visualización real podría mostrarnos como varían los pesos en las neuronas del controlador. Sin embargo, este gráfico ilustra muy claramente el potencial de estos controladores neuronales.

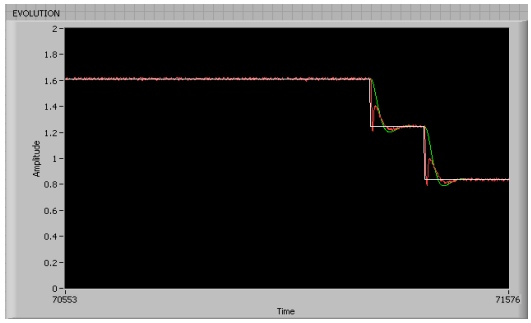

Fig. 10 *Respuesta real de la planta monitoreada en LabVIEW.*

En las siguientes figuras mostraremos las imágenes reales del controlador funcionando para

una señal de referencia escalón, la cual se pudo variar con la barra *slide* de LabView. La Fig. 11 muestra un cambio de nivel de referencia. Note que este resultado real, corresponde a lo predicho en las Fig. 5 o 6.

Fig. 11 *Fotografía de la respuesta de la planta.*

Las Fig. 12 y 13 muestran una comparación de resultados. El primer valor corresponde al nivel de referencia o *setpoint*, mientras que el segundo valor corresponde al obtenido a la salida de la planta tomado con un multímetro *Fluke* de alta precisión. Este error corresponde a aproximadamente *0.0004V*.

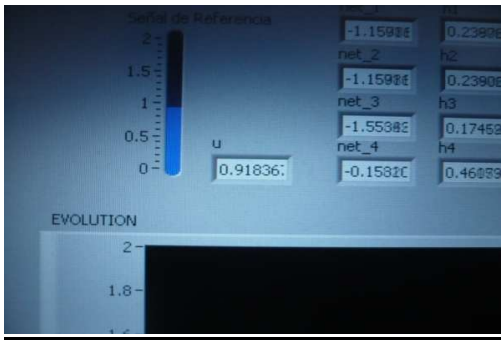

Fig. 12 *Fotografía del set point deseado.*

Fig. 13 *Voltaje de respuesta validando set point de la Fig. 12.*

Algo similar a las simulaciones de la Fig. 6, se muestra en la Fig. 14. En esta figura se ha sometido a la planta a cambios bruscos en el nivel de referencia.

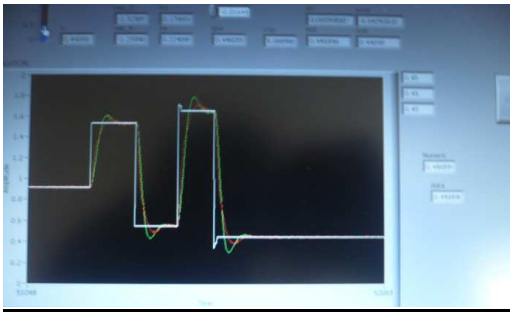

Fig. 14 *Respuesta de la planta sometida a cambios en la referencia.*

La Fig. 15 muestra a la planta de segundo orden utilizado en el presente trabajo. En dicha planta, mediante un potenciómetro se pueden variar los parámetros pasando de ser sub-amortiguada hacia sobre-amortiguada.

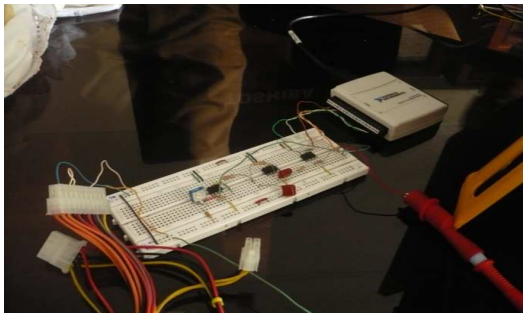

Fig. 15 *Planta de segundo orden utilizado en el proyecto.*

En la Fig. 16, mostramos una vista general del proceso del sistema de control. Se requirió de una computadora portátil y un multímetro Fluke.

Fig. 16 *Elementos usados en la implementación.*

CONCLUSIONES

Se logró comprobar que la arquitectura tomada en cuenta, la que considera dos retroalimentaciones es la más óptima ya que se tuvo en cuenta dos informaciones de nivel y del sistema de referencia; sin embargo, el seguimiento todavía no es del todo perfecto como se puede apreciar en la Fig. 14. Este controlador Neural podrá ser adecuado fácilmente en plantas con fuentes no linealidades y multivariables, de cuya dinámica se tenga poco conocimiento [3, 4].

El controlador es bien práctico para sistemas donde existe variación de parámetros. Esto se debe a que la retroalimentación respecto del sistema de referencia trata en todo momento que la respuesta de la planta siga en todo momento al sistema de referencia, y según Paul j. Werbos este sistema en su conjunto trata de converger hacia un error cero [5].

Se concluye además que la simulación con LabVIEW tiene versatilidad de programación. En el entorno gráfico de LabVIEW se logró sintetizar los algoritmos de entrenamiento para el perceptrón multicapa. Aquí fue posible representar cada fase del entrenamiento y visualizar las variables temporales en cualquier instante.

REFERENCIAS

1. **Valverde, R., Gachet, D., Salichs, M. A.,** "Dynamic systems identification using rbf neural networks", área de ingeniería de sistemas y automática Universidad Carlos III de Madrid, Spain, pp 1-10. España, 1999.
2. **Önder Efe, M., Kaynak, O., Yu, X**., **Wilamowski, B. M**., "Sliding mode control of nonlinear systems using gaussian radial basis function neural networks", pp 1-6. USA 2001.
3. **Valverde, R**., "Neurocontrol of continuous processes through reinforcement learning". Universidad Carlos III de Madrid. C/butarque, 15. 28911, Leganés, pp 1-5. España, 1998.
4. **Valverde, R.**, "Control de sistemas mediante redes neuronales". Aprendizaje por refuerzo, Universidad Carlos III de Madrid, Tesis doctoral, pp 8-16. España, 1999.
5. **Werbos, P. J**., "Neural networks applications". 1997 iop publishing ltd and Oxford University Press, Handbook of Neural Computation release 97/1 f1.9:1-f1.9:10, pp 1 -10. USA 1997.

Correspondencia: rmolero@giscia.com