Control de Posición para Motor de Corriente Directa mediante Redes Neuronales Artificiales

Rubén Eduardo Rodríguez Fuerte
Estudiante de Mecatrónica, Universidad La Salle Noroeste
Cd. Obregón, México
eduardo.rdz112358@gmail.com

Eduardo Núñez Pérez
Área de Ingeniería y Tecnología, Universidad La Salle Noroeste
Cd. Obregón, México
eduardo.nunez@lasallenoroeste.edu.mx

***Resumen*— Una red neuronal artificial es entrenada en base a la metodología del control inverso directo con la finalidad de controlar la posición de un motor de corriente continua, donde se prueban varias estructuras de control, variando la información de entrada para la red neuronal entre cada estructura.**

Palabras clave—Redes neuronales, control inverso directo, control de posición.

I. Introducción

En la industria, el control de posición de motores es un aspecto crucial en una amplia gama de aplicaciones, desde la robótica y automatización industrial hasta dispositivos médicos, donde la precisión y estabilidad son fundamentales [1]. A medida que estos sistemas empiezan a demandar mayor exactitud, el uso de redes neuronales artificiales (RNA) como controladores emerge como una solución a la no linealidad, incertidumbre y complejidad de sistemas [2]. Convirtiéndolas en una herramienta poderosa en el campo del control automático, destacándose particularmente como controladores en sistemas complejos como [3], [4], [5] y [6]. A continuación, se describe la planta que se desea controlar, se habla sobre las redes neuronales artificiales y la idea básica del control inverso directo, se presentan las consideraciones que se toman para el entrenamiento e implementación de la RNA, finalmente se muestran los resultados obtenidos y conclusiones.

II. Descripción de la planta

La planta a controlar es un motor de corriente continua que cuenta con un reductor, su eje está acoplado mediante una polea a un codificador rotativo que captura la posición, Fig. 1.

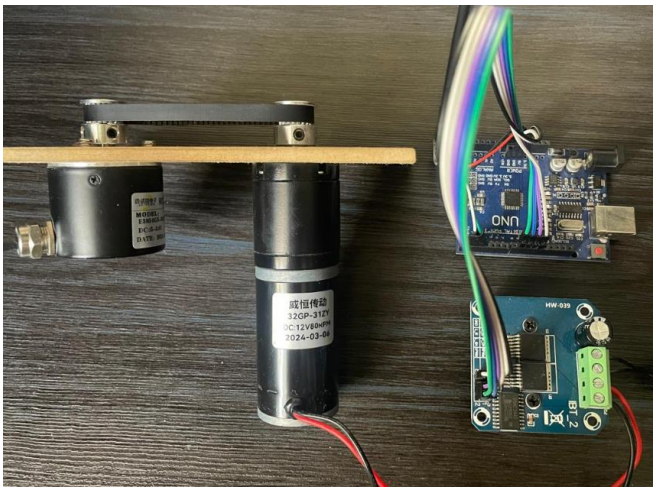

Fig. 1. Planta.

El motor trabaja con un voltaje nominal de 12 volts, sin carga consume 0.45 amperes, con carga nominal 1.4 amperes, llegando a consumir 21.6 watts, alcanzando una velocidad de 65 revoluciones por minuto.

Un puente H energiza al motor con la ayuda de la modulación por ancho de pulso utilizando 2 canales, uno para el sentido horario y otro para el sentido antihorario. Un microcontrolador se encarga de controlar este proceso a la vez que envía y recibe los datos de posición y voltaje que se le aplican al motor respectivamente, dicha comunicación es realizada por puerto serial hacia el software Matlab. La Fig. 2 muestra un diagrama del sistema de prueba.

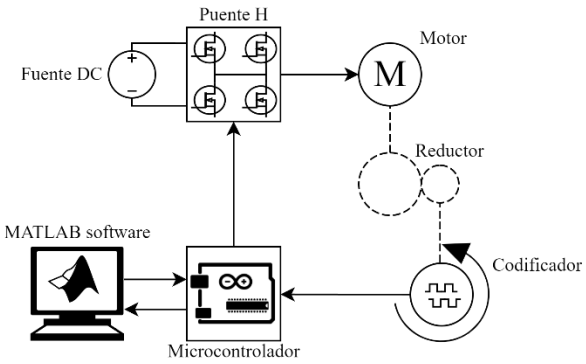

Fig. 2. Diagrama del sistema de prueba.

III. Redes Neuronales

Las redes neuronales artificiales (RNA) son modelos computacionales capaces de aprender relaciones complejas, patrones y clasificar imágenes de manera similar a como lo haría una red de neuronas biológicas en un cerebro humano [7].

La Fig. 3 muestra una forma de relacionar una neurona biológica y una artificial, donde x_i son las entradas de la neurona, w_i conocida como peso, es la importancia que se le da a dicha entrada, este proceso emula las dendritas de una neurona biológica encargadas de recibir la información de otras neuronas, Σ es encargada de recibir y sumar la información de las entradas imitando el cuerpo celular, s es una función de activación qué emula la sinapsis de una neurona biológica encargada de transmitir el potencial de acción en energía eléctrica y química a otra neurona [8].

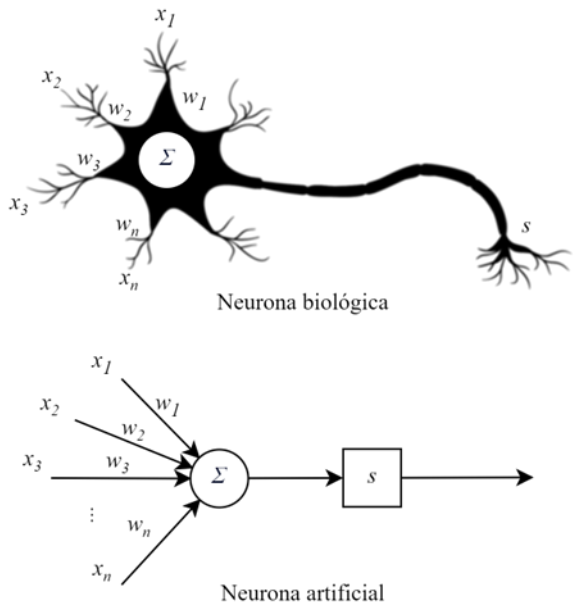

Fig. 3. Neurona biológica y artificial.

Una RNA está compuesta por múltiples neuronas artificiales ordenadas por capas Fig.4. Durante el entrenamiento de la RNA los pesos de cada neurona se modifican utilizando algoritmos de aprendizaje [9].

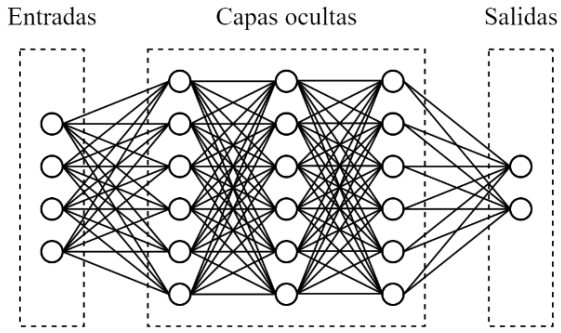

Fig. 4. Estructura general de una red neuronal artificial (RNA).

IV. Control inverso directo

Este método busca encontrar el modelo inverso de una planta con la intención de usarlo como controlador al ser colocado en serie con dicha planta [10], de esta manera se logra cancelar los efectos de la dinámica ante una señal de referencia como se muestra en la Fig. 5.

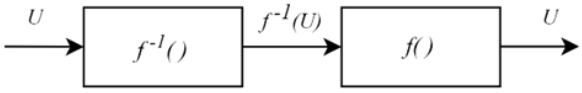

Fig. 5. Idea conceptual del control inverso directo.

Una RNA puede ser entrenada para aprender la dinámica inversa de una planta [11], donde se selecciona y aplica una entrada a la planta para obtener una salida, el propósito del entrenamiento de la RNA consta en obtener las entradas a partir de las salidas. Sin embargo, el correcto funcionamiento de este método depende si la RNA define bien la inversa de la planta.

Obtener la dinámica inversa de manera analítica puede ser demasiado complejo [12]. Por otra parte, la dinámica de una planta puede ser descrita en función a la información entrada-salida utilizando el siguiente modelo en tiempo discreto, donde *y* es la salida del sistema, *u* la entrada aplicada, *k* el instante actual, *n* y *m* son constantes que dependen del orden del sistema y de la dinámica del mismo.

$$y(k+1)=f[y(k),\ldots,y(k-n+1),u(k),\ldots,u(k-m+1)] \quad (1)$$

La salida de la planta $y(k+1)$ depende de n-salidas y m-entradas anteriores. En base al modelo (1) se puede definir la inversa de la planta de la siguiente manera:

$$u(k)=f^{-1}[r(k+1)y(k),\ldots,y(k-n+1),u(k),u(k-m+1)] \quad (2)$$

Donde *y(k+1)* es un valor desconocido el cual puede ser reemplazado por una salida deseada *r(k+1)*. De esta manera se puede entrenar una RNA con datos de entrada-salida de la planta para que aprenda la dinámica inversa de la misma. [13] Plantea que el entrenamiento puede ser realizado al usar solo las salidas del sistema tal que:

$$u(k)=f^{-1}[r(k+1)y(k),\ldots,y(k-n+1)] \quad (3)$$

Este modelo (3) es el que se utiliza en el presente trabajo de investigación, el cual se implementa a través del entrenamiento de una RNA.

V. Entrenamiento e implementación de la RNA

Lo primero a realizar es obtener los datos que se usarán para el entrenamiento de la RNA según la estructura a utilizar Fig. 6, donde $u(k)$ que es la salida de la RNA, se obtiene a partir del voltaje que se le aplica al motor, $y(k)$ y las salidas en diferentes instantes de tiempo, que se obtienen a través de retrasos en tiempo discreto, son las entradas de la RNA, $y(k)$ se obtienen de la posición que alcanza el motor ante $u(k)$. Como el entrenamiento que se realiza es de manera previa a la implementación, los datos para entrenar la red se obtienen de los datos de entrada-salida, haciendo el corrimiento o desfasamiento de la salida necesarios de acuerdo a la estructura elegida.

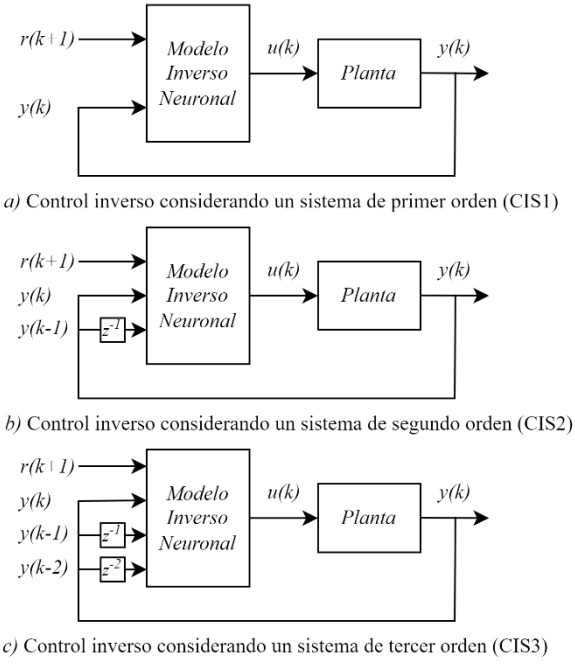

Fig. 6. Estructuras para obtener el modelo inverso.

El parámetro *n* de (3) define cuantas muestras pasadas de la salida *y(k)* se utilizan para predecir la acción de control *u(k)*.

Como el entrenamiento de la RNA es basado en datos [14], *n* puede ser ajustado libremente considerando la planta como un sistema de orden desconocido [15].

Las ecuaciones que definen las diferentes estructuras de control son las siguientes.

CIS1:

$$u(k)=f^{-1}[r(k+1), y(k)] \quad (4)$$

CIS2:

$$u(k)=f^{-1}[r(k+1), y(k), y(k-1)] \quad (5)$$

CIS3:

$$u(k)=f^{-1}[r(k+1), y(k), y(k-1), y(k-2)] \quad (6)$$

La generación y adquisición de datos se realiza a través del software de Simulink, complemento de Matlab. Donde de manera aleatoria se mandan valores de voltaje, se almacenan en un vector y se reciben valores de la posición actual del motor, la cual también es almacenada en un vector. La Fig. 7 muestra el diagrama a bloques que se utiliza para realizar dicha adquisición de datos.

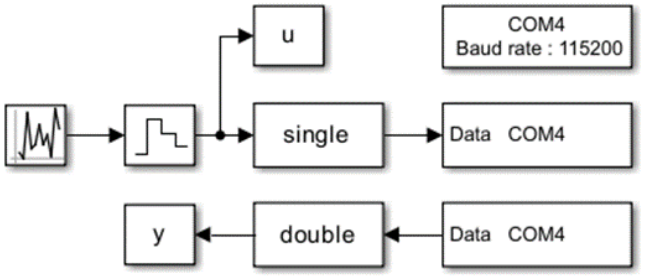

Fig. 7. Adquisición de datos por puerto serial.

La Fig. 8 muestra los datos adquiridos a lazo abierto, donde la salida del sistema no interviene en la entrada.

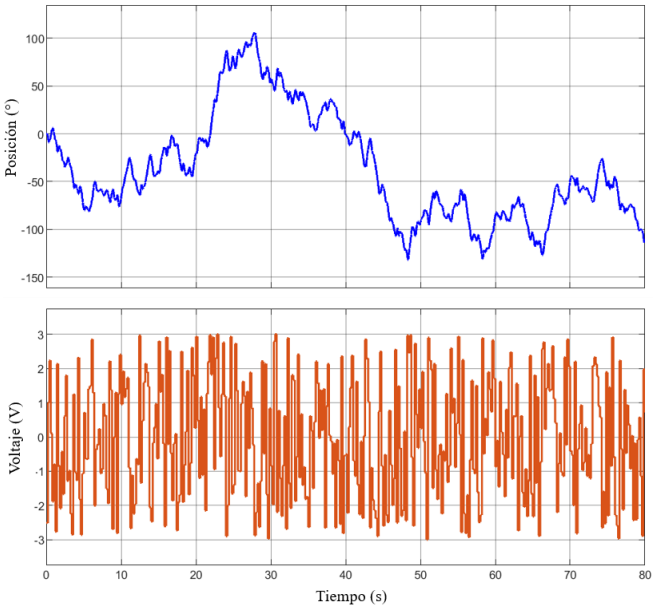

Fig. 8. Datos entrada-salida sin retrasos utilizados para el entrenamiento.

La señal azul es la posición alcanzada, y la señal roja es el voltaje aplicado, siendo estos muestreados con un periodo de 50 milisegundos durante 80 segundos.

El propósito de aplicar este tipo de señal de voltaje es para excitar al motor de manera conveniente y lograr capturar la dinámica del sistema.

Una vez obtenidos los datos de entrada y salida de la planta, el entrenamiento puede llevarse a cabo. Sin embargo, [16] presenta el impacto que se obtiene al escalar los datos de entrenamiento, lo que puede afectar el resultado y desempeño de la RNA.

El rendimiento de los entrenamientos se muestra en la Tabla 1, donde se utiliza el error cuadrático medio para comparar entrenamientos sin y con escalamiento.

TABLA 1. DESEMPEÑO DE ENTRENAMIENTO.

Esquema	**Error cuadrático medio**	
	Datos sin escalar	***Datos escalados***
CIS1	1.0131	0.1849
CIS2	0.79174	0.15949
CIS3	0.72335	0.18353

En base a las pruebas realizadas durante la implementación de las RNA utilizando datos sin escalar, se decide utiliza el método de escalado de características. Una de las ventajas de este método es que disminuye el coste computacional y puede aumentar el rendimiento. Aunque este método es sensible a valores atípicos, estos se omiten desde un principio al seleccionar los datos que se utilizarán en el entrenamiento. A continuación, se presenta la ecuación que realiza el escalado de características:

$$\overline{X_{ij}} = 2\left(\frac{X_{ij}-X_{i_{min}}}{(X_{i_{max}}-X_{i_{min}})}\right)+1 \quad (7)$$

Donde $\overline{X_{ij}}$ es el vector de datos escalados, X_{ij} es el vector de datos sin escalar, $X_{i_{max}}$ y $X_{i_{min}}$ representan los valores máximos y mínimos respectivamente del vector sin escalar. Se aplicó (7) a los datos sin retrasos de la Fig. 8. Es importante señalar que la salida debe ser desescalada para regresar a las magnitudes que maneja la planta.

La función que se utiliza para entrenar las RNA divide los datos en tres conjuntos: datos de entrenamiento, datos de validación y datos de prueba. Con porcentajes del 70%, 15%, y 15% respectivamente, cada uno de estos conjuntos cumple con una tarea especifica para asegurar que las RNA aprendan bien el modelo inverso. Los datos de entrenamiento son los que se utilizan directamente para aprender los patrones, los datos de validación ayudan a ajustar las RNA para evitar el sobreajuste y generalicen bien nuevos datos, y los datos de prueba evalúan el rendimiento final con datos no vistos ni en el entrenamiento ni en la validación.

En la Tabla 2 se utiliza el coeficiente de correlación R de Pearson, el cual indica si hay relación lineal entre las predicciones de las RNA y los datos objetivos.

TABLA 2. CORRELACIÓN ENTRE LAS PREDICCIONES Y OBJETIVOS.

Datos	**Esquema**		
	CIS1	***CIS2***	***CIS3***
Entrenamiento	0.63801	0.66803	0.69418
Validacion	0.62651	0.73293	0.65494
Prueba	0.61873	0.59981	0.66405

Según [17], un coeficiente con magnitud entre 0.5 y 1 se puede interpretar como una correlación fuerte.

En las pruebas realizadas durante el entrenamiento se utilizan RNA multicapa no recurrentes, con una estructura de 4 capas, las cuales contienen 15, 20, 5 y 1 neuronas respectivamente. En esta arquitectura la información fluye en una sola dirección: desde la capa de entrada, pasando por las capas ocultas, hasta la capa de salida, en la cual no hay retroalimentaciones [18]. El tipo de función de activación que se utiliza es la sigmoide tangente hiperbólica. La elección depende de la tarea que se realiza ya que no hay una regla general para seleccionar una función de activación [19].

El algoritmo de entrenamiento que se usa es el de Levenberg-Marquardt (LM) debido a su capacidad para combinar rapidez, precisión y estabilidad. LM es generalmente el algoritmo de retropropagación más rápido, diseñado para mejorar la velocidad de convergencia [20], [21]. Sin embargo, [22] y [23] muestran situaciones donde diferentes algoritmos de entrenamiento ofrecen mejores resultados.

Las RNA son entrenadas usando el Deep Learning Toolbox de Matlab e implementada en Simulink. En la Fig. 9 se muestra el diagrama a bloques implementado de las estructuras CIS1, CIS2, y CIS3 presentadas en la Fig. 6.

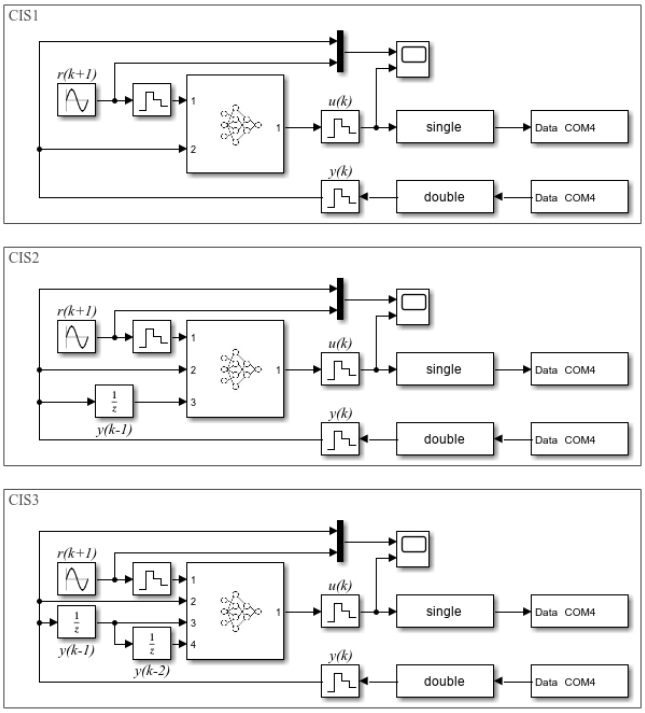

Fig. 9. Implementación de las RNA.

VI. Resultados

A continuación, se presentan los resultados de la implementación, en donde a las RNA se les dan trayectorias de posición a seguir con el objetivo que predigan la señal de voltaje que se le debe aplicar al motor para hacer el seguimiento de trayectoria.

Las trayectorias a seguir son: una señal del tipo escalón con una amplitud de 100 grados, y una señal senoidal a una frecuencia de 0.5 radianes por segundo con una amplitud de 50 grados.

Utilizando CIS1 de la Fig. 9, se muestran los resultados del seguimiento de trayectoria y voltaje predicho por la RNA en la Fig. 10.

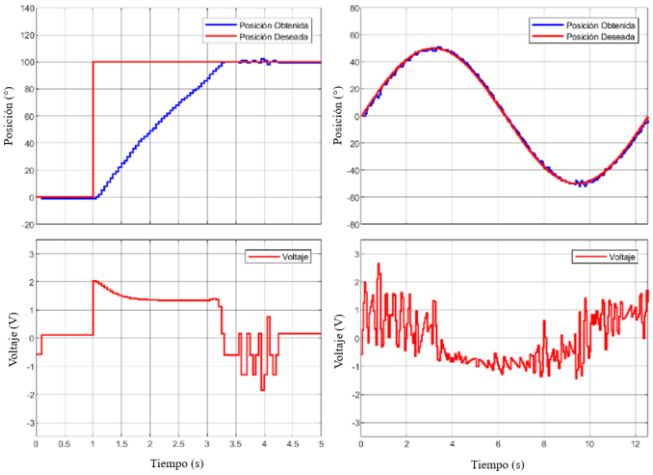

Fig. 10. Resultados de CIS1.

Utilizando CIS2 de la Figura 9, se muestran los resultados del seguimiento de trayectoria y voltaje predicho por la RNA en la Figura 11.

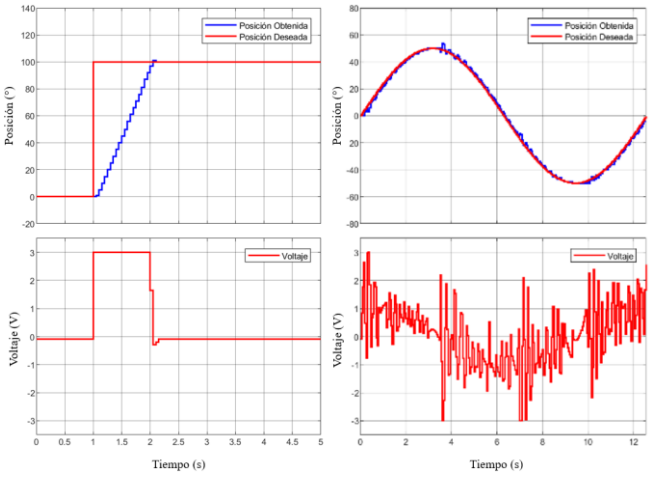

Fig. 11. Resultados de CIS2.

Utilizando CIS3 de la Figura 9, se muestran los resultados del seguimiento de trayectoria y voltaje predicho por la RNA en la Figura 12.

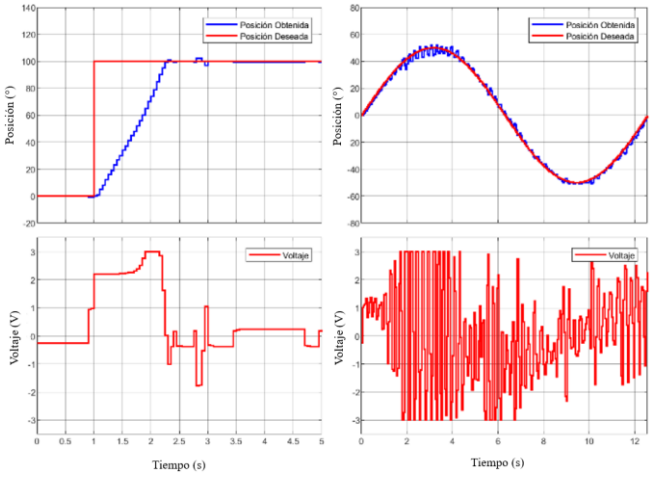

Fig. 12. Resultados de CIS3.

Aunque cada RNA fue entrenada bajo las mismas condiciones, cada esquema muestra comportamientos diferentes en el seguimiento de trayectoria y predicción de voltaje, donde destaca la velocidad y oscilaciones dependiendo del tipo de trayectoria a seguir. CIS1 tiene una respuesta lenta y con pocas oscilaciones. CIS2 tiene la respuesta más rápida, aunque presenta cambios bruscos en la predicción de voltaje. CIS3 tiene una respuesta un poco más lenta que CIS2, sin embargo, es el que presenta más oscilaciones.

No se incluye un análisis de estabilidad ya que no se maneja un modelo analítico del sistema.

CONCLUSIONES

El uso de redes neuronales como controladores es una estrategia eficiente que ofrece varias ventajas, como aprender la no linealidad de sistemas complejos con los datos de entrada-salida. Lo cual es útil en sistemas difíciles de modelar.

En este artículo se utiliza la tendencia general del control inverso directo como método de entrenamiento, él cual muestra buenos resultados durante la implementación.

El escalamiento de datos fue de gran ayuda durante el entrenamiento, ya que en los entrenamientos que se llevaron acabo con datos sin escalar, la predicción de voltaje excedía el rango de operación del motor. Estos excesos siguen estando presente en las pruebas con datos escalados aunque con menos frecuencia.

Deep Learning Toolbox de Matlab tiene la capacidad de diseñar e implementar redes neuronales artificiales de manera eficiente, en donde se puede modificar el número de neuronas, capas ocultas, funciones de activación, algoritmos de entrenamiento entre otras cosas de manera sencilla mediante funciones o apps.

Como trabajos futuros se propone utilizar métodos de entrenamiento on-line, es decir, que la red neuronal artificial se ajuste mientras el sistema está operando, probar diferentes algoritmos de entrenamiento, incluso probar controlar diferentes tipos de plantas como convertidores de voltaje DC-DC. Además, comparar el rendimiento de las redes neuronales artificiales presentadas con otro tipo de controladores como PID o basados en modelos lineales.

REFERENCIAS

[1] S. K. Sahdev, Electrical Machines. Cambridge Univ. Press, 2019.

[2] Werbos, P. (1991). An overview of neural networks for control. IEEE Control Systems Magazine, 11(1), 40–41.

[3] Rajesh, R., et al, "Artificial neural network based inverse model control of a nonlinear process," in 2015 International Conference on Computer, Communication and Control (IC4), 2015, pp. 1–6.

[4] W. Ariza-Zambrano, A. Serpa. "Direct inverse control for active vibration suppression using artificial neural networks," in Journal of vibration and control, vol. 27, no. 1-2, pp. 31–42, 2021.

[5] H. Suprijono, B. Kusumoputro. "Direct inverse control based on neural network for unmanned small helicopter attitude and altitude control," in Journal of Telecommunication, Electronic and Computer Engineering (JTEC), vol. 9, no. 2-2, pp. 99–102, 2017.

[6] A. Widaryanto, B. Kusumopotro, "Modeling and Designing Direct Inverse Control Using Back-propagation Neural Network for Skid Steering Boat Model," in 2019 IEEE International Conference on Innovative Research and Development (ICIRD), 2019, pp. 1–5.

[7] S. Ghosh-Dastidar, H. Adeli. "Spiking neural networks," in International journal of neural systems, vol. 19, no. 04, pp. 295–308, 2009.

[8] J. Lin. "Artificial neural network related to biological neuron network: a review," in Advanced Studies in Medical Sciences, vol. 5, no. 1, pp. 55–62, 2017.

[9] Dike, H., et al, "Unsupervised learning based on artificial neural network: A review," in 2018 IEEE International Conference on Cyborg and Bionic Systems (CBS), 2018, pp. 322–327.

[10] [B. Widrow, G. Plett, "Nonlinear adaptive inverse control," in Proceedings of the 36th IEEE Conference on Decision and Control, 1997, pp. 1032–1037.

[11] D. Psaltis, A. Sideris, A. Yamamura. "A multilayered neural network controller," in IEEE control systems magazine, vol. 8, no. 2, pp. 17–21, 1988.

[12] J. Žilkova, J. Timko, P. Girovsky. "Nonlinear system control using neural networks," in Acta Polytechnica Hungarica, vol. 3, no. 4, pp. 85–94, 2006.

[13] E. Perez, J. Ascencio. "Identification and control of systems with and without zeros via approximation of the state evolution function," in IEEE Latin America Transactions, vol. 12, no. 4, pp. 564–573, 2014.

[14] Hassija, V., et al. "Interpreting black-box models: a review on explainable artificial intelligence," in Cognitive Computation, vol. 16, no. 1, pp. 45–74, 2024.

[15] O. Nelles, "Nonlinear System Identification: From Classical Approaches to Neural Network and Fuzzy Models Springer-Verlag," 2001.

[16] J. Rodríguez González, E. Ugalde Saborio. "Impacto de la estandarización y escalado: factor para predicción de costos en proyectos a través de una red neuronal artificial," in Ingeniare. Revista chilena de ingeniería, vol. 29, no. 2, pp. 265–275.

[17] Lalinde, J., et al. "Sobre el uso adecuado del coeficiente de correlación de Pearson: definición, propiedades y suposiciones," in Archivos venezolanos de Farmacología y Terapéutica, vol. 37, no. 5, pp. 587–595, 2018.

[18] Sharma, P., et al. "feedforward neural network: A Review," in International Journal of Advanced Research in Engineering and Applied Sciences (IJAREAS), vol. 2, no. 10, pp. 25–34, 2013.

[19] S. Sharma, S. Sharma, A. Athaiya. "Activation functions in neural networks," in Towards Data Sci, vol. 6, no. 12, pp. 310–316, 2017.

[20] J. More, "The Levenberg-Marquardt algorithm: implementation and theory," in Numerical analysis: proceedings of the biennial Conference held at Dundee, June 28–July 1, 1977, 2006, pp. 105–116.

[21] A. Ranganathan. "The levenberg-marquardt algorithm," in Tutoral on LM algorithm, vol. 11, no. 1, pp. 101–110, 2004.

[22] M. Ibrahim, K. Jihad, L. Kamal. "Determining optimum structure for artificial neural network and comparison between back-propagation and levenberg-marquardt training algorithms," in International Journal of Engineering Science, vol. 14887, 2017.

[23] K. Jazayeri, M. Jazayeri, S. Uysal, "Comparative analysis of Levenberg-Marquardt and Bayesian regularization backpropagation algorithms in photovoltaic power estimation using artificial neural network," in Advances in Data Mining. Applications and Theoretical Aspects: 16th Industrial Conference, ICDM 2016, New York, NY, USA, July 13-17, 2016. Proceedings 16, 2016, pp. 80–95.