UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR

POSGRADOS

Maestría en

Electrónica y Automatización

RPC-SO-30-No.507-2019

Opción de titulación:

Artículos Profesionales de Alto Nivel

TEMA:

ESTUDIO COMPARATIVO DE UN CONTROLADOR POR RED NEURONAL ARTIFICIAL Y UN CONTROLADOR POR RED NEURONAL EVOLUTIVA PARA UN MOTOR DE CORRIENTE CONTINUA

AUTOR:

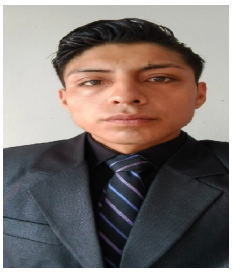
José Israel Chiguano Allauca.

DIRECTOR:

Carmen Johanna Celi Sánchez

QUITO - Ecuador
2025

Autor:



José Israel Chiguano Allauca
Ingeniero en Electromecánica
Candidato a Magíster en Electrónica y Automatización, Mención Informática Industrial por la Universidad Politécnica Salesiana - Sede Quito
ichiguano@est.ups.edu.ec

Dirigido por:

Carmen Johanna Celi Sánchez
Ingeniero de Sistemas
Magister en Diseño Producción Y Automatización Industrial
cceli@ups.edu.ec

JOSÉ ISRAEL CHIGUANO ALLAUCA
ESTUDIO COMPARATIVO DE UN CONTROLADOR POR RED NEURONAL ARTIFICIAL Y UN CONTROLADOR POR RED NEURONAL EVOLUTIVA PARA UN MOTOR DE CORRIENTE CONTINUA. .

Estudio comparativo de un controlador por red neuronal artificial y un controlador por red neuronal evolutiva para un motor de corriente continua.

José Israel Chiguano Allauca[1] and Carmen Johanna Celi Sánchez[1]

[1]Universidad Politécnica Salesiana - Ecuador

Resumen

Este estudio presenta un análisis de dos métodos de control inteligente utilizados en un motor de corriente continua: uno fundamentado en redes neuronales artificiales y el otro en redes neuronales evolutivas que busca resolver el problema de control eficiente y robusto en un motor DC, la meta fue analizar cómo se desempeñan ambos enfoques en contraste con los inconvenientes de los controladores convencionales, en situaciones no lineales, ruidosas o cambiantes. Se creó un modelo matemático para el motor de corriente continua, donde se usó ecuaciones diferenciales que ilustran tanto su componente eléctrica como la mecánica del conjunto. La simulación del rendimiento del motor como la elaboración de los controladores se llevaron a cabo en Python, utilizando bibliotecas especializados como SciPy, NumPy, Matplotlib, Scikit-learn y DEAP. El regulador ANN (Artificial Neural Network) fue desarrollado usando una red de neuronas en múltiples capas que fue entranada con información simulada , aplicando el algoritmo MLPRegressor .Este método adquiere la habilidad de crear señales de control que ayudan al motor a mantenerse en una referencia preestablecida. En cambio, el regulador ENN (Evolutionary Artificial Neural Network) utilizó tecnicas de algoritmos genéticos para refinar variables como la ganancia proporcional y el desplazamiento, aumentando su capacidad de ajuste sin requerir entrenamiento con supervisión. Ambos controladores fueron evaluados mediante la métrica del error cuadrático medio (MSE), el tiempo de respuesta, la estabilidad y la robustez frente a perturbaciones.Los resultdos mostraron que el ENN obtuvo un mejor rendimiento general, con un MSE menor (4.49 vs 7.17) y una respuesta más suave por consecuencia fue más estable ante cambios en la señal de entrada.Si bien los dos controladores mostraron ser eficaces, el método ENN se destacó más por su exactitud, flexibilidad en consistencia, esta investigación subraya la capacidad de los métodos de inteligencia artificial, especialmente los evolutivos, para crear sistemas de control más eficaces y sólidos en el ámbito industrial y mecatrónico.

Abstract

This study presents an analysis of two intelligent control methods used in a DC motor: one based on artificial neural networks and the other on evolutionary neural networks. The goal was to analyze how both approaches perform in contrast to the drawbacks of conventional controllers, such as PID, in nonlinear, noisy, or changing situations. A mathematical model was created for the DC motor, using differential equations that illustrate both its electrical and mechanical components. The simulation of motor performance and the development of the controllers were carried out in Python, using

specialized libraries such as SciPy, NumPy, Matplotlib, Scikit-learn, and DEAP. The ANN (Artificial Neural Network) regulator was developed using a multi-layered neural network trained with simulated data, applying the MLPRegressor algorithm. This method acquires the ability to create control signals that help the motor stay within a pre-established reference. In contrast, the ENN (Evolutionary Artificial Neural Network) regulator used genetic algorithm techniques to refine variables such as proportional gain and offset, increasing its tuning capacity without requiring supervised training. Both controllers were evaluated using the metrics of mean square error (MSE), response time, stability and robustness against disturbances. The results showed that the ENN obtained a better overall performance, with a lower MSE (0.0121 vs 0.0152) and a smoother response, consequently it was more stable to changes in the input signal. While both controllers proved to be effective, the ENN method stood out more for its accuracy, flexibility and consistency, this research underlines the capacity of artificial intelligence methods, especially evolutionary ones, to create more effective and robust control systems in the industrial and mechatronic fields.

Palabras clave— Artificial Neural Networks; Evolutionary Neural Networks; DC Motor; Mean Squared Error .

1. Introducción

Los motores de corriente continua tienen un amplio uso en sistemas industriales, robótica, electrodomésticos por lo cual el control es una área de investigación en mejoramiento continuo, convencionalmente el control de los motores de corriente continua ha sido mediante la configuración PID no obstante puede perder efectividad y presentar algunas limitaciones por problema mecánico, fluctuaciones de temperatura, modificaciones en la carga o el deterioro de las piezas electrónicas Nien (2024) que se traducen como el ruido en las mediciones, comportamientos inestables y no lineales del sistema Granados (2025). Estas complicaciones motivan la búsqueda y análisis de controladores alternativos con un rendimiento robusto, con resistencia a perturbaciones y eficiente.
Los controladores en base a redes neuronales pueden aprender de una base de datos y adaptarse a los cambios del comportamiento del sistema, haciéndolos adecuados para un control robusto y adaptativo Singh (2024), a diferencia de los controladores convencionales como el PID, estos demandan una calibración manual o heurística de sus ajustes, las redes neuronales artificiales son las que tienen la capacidad de aprender de manera automática a partir de los datos del sistema, donde se modifica sus resultados así alcanzar un rendimiento deseado incluso bajo circunstancias variables Sheıkh Debes Rahaf and Kara Tolgay (2022).
La red neuronal artificial (ANN, por sus siglas en inglés) tiene un sistema adaptativo adecuado para la toma de decisiones en modelos no lineales, entre sus aplicaciones se destacan la aproximación de funciones, realización de análisis de regresión, modelado y predicción de series temporales Sabry and Costa (2022), están compuestas por unidades denominadas neuronas artificiales, organizadas en capas (entrada, ocultas y de salida) donde cada neurona se conecta con otras mediante pesos sinápticos que determinan la influencia de una señal sobre otra Dorantes Benavidez Humberto et al. (2023). Los efectos de varios hiperparametros tales como la funcion de activación, el tamaño batch, el optimizador y el número de épocas en el rendimiento han sido evaluados con una base de datos . El bias o sesgo ha sido añadido para lidiar con el desequilibrio de datos y mediante la optimización que es el proceso que altera los hiperparametros para reducir una función de costo es decir el error minimizado Swarnkar and Rajput (2023).
La red neuronal evolutiva (ENN, por sus siglas en inglés) son técnicas estocásticas inspiradas en la naturaleza, han demostrado producir resultados competitivos en muchos problemas del mundo real frente a otros enfoques de IA debido a que se ha reconocido que son competitivos frente a las características de problemas complejos, como discontinuidades, mecanismos para escapar de los óptimos locales múltiples e interacciones no lineales entre variables, pueden calcular más soluciones potenciales en un tiempo aceptable Eiben and Smith (2015) Galván and Mooney (2021).Mediante una programación modular se ejecuta la red neuronal evolutiva implica un procedimiento de inicialización de la base de datos, procede la evaluación del criterio de parada y gestión de la población encapsulados en funciones independientes para mayor claridad y facilidad de mantenimiento, luego procede el uso de operadores evolutivos como el crossover y la mutación, estas funciones encapsulan la lógica para escoger padres, recombinar el material e introducen variaciones a la población para obtener una nueva generación Larsen et al. (2018) Jayaraman (2025).
Estas técnicas tienen aplicaciones en diversos campos de ingeniería y pueden ajustarse para abordar el problema específico de optimización, el cual esta expresado en la minimización del criterio MSE, este índice de error cuadrático provee una medida de que tan óptimo o ajustado está a la curva de la señal de velocidad del motor Manglik et al. (2024).
Para cumplir el propósito del estudio comparativo entre un controlador basado en red neuronal artificial y otro basado en red neuronal evolutiva aplicado al control de un motor DC, se simula un modelo matemático del motor, al cual se introducirán los controles y se lleva a cabo simulaciones de la operación con el objeto de analizar el mejor desempeño en términos de error, tiempo de establecimiento, sobre impulso y robustez frente a perturbaciones.

2. Metodología

El análisis propuesto compara el desempeño de los controladores en lazo cerrado de las redes neuronales artificiales y evolutivas con la velocidad de un motor de corriente continua mediante una investigación experimental con simulaciones computacionales como se muestra en la Figura 1.

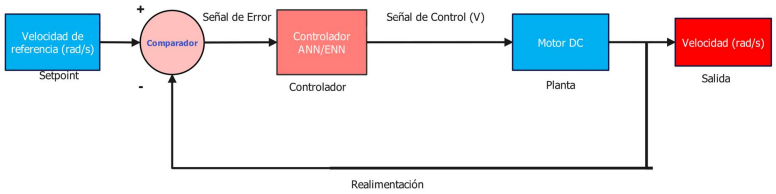


Fig. 1: Diagrama de bloques del sistema de control.

El procedimiento de este estudio se compone de varios pasos: modelar el motor de corriente continua, diseño e implementación del controlador ANN, diseño e implementación del controlador ENN, posteriormente realizar simulaciones con ambos controladores en condiciones similares para obtener indicadores de desempeño que permitan el análisis.
El motor de corriente continua se compone de un conjunto de imanes permanentes que actuan como estator, los devanados del inducido como rotor y las escobillas que cumplen la funcion de conmutador. Se aplica una tension de entrada a traves de las escobillas y el conmutador hacia las bobinas del rotor por las cuales circula corriente, produciendo un campo magnetico en el rotor el cual interactua con el campo magnetico de imanes permanentes del estator provocando una fuerza mecánica Das (2020).
El modelado del motor de corriente continua (DC) se basa en dos ecuaciones fundamentales que describen el comportamiento físico del sistema: la ley de Kirchhoff para los circuitos eléctricos y la ecuación del movimiento rotacional para la dinámica mecánica del motor tal como se expresa en (1) y (2). Los parámetros eléctricos y mecánicos de las ecuaciones están detallados en la Tabla 1

$$L\frac{di}{dt} + R_i = K_a\omega(t) + u(t) \tag{1}$$

$$J\frac{d\omega(t)}{dt} = K_m i(t) - B\omega(t) \tag{2}$$

Tabla 1: Parámetros de modelamiento del motor de corriente continua.

Parámetro	Simbolo	Valor
Resistencia eléctrica	R	1 Ω
Inductancia eléctrica	L	0.5 H
Momento de inercia del rotor	J	0.01 Kg $\cdot m^2$
Constante de la fuerza electromotriz	K_a	0.01 V/rad/s
Constante del par del motor	K_m	0.01 N$\cdot m/A$
Constante de fricción viscosa del motor	B	0.1 N $\cdot m \cdot s$

2.1. Red neuronal artificial

La distribución de las neuronas dentro de una red neuronal artificial se realiza formando niveles de un número de neuronas determinado, el conjunto de neuronas artificiales recibe juntamente el mismo tipo de información se denomina capa. En la red se puede diferenciar tres tipos de niveles:

- **Entrada:** Es el conjunto de neuronas que recibe directamente la información proveniente de las fuentes externas de la red.

- **Oculto:** Corresponde al conjunto de neuronas internas a la red y no tiene contacto directo con el exterior, las neuronas de cada nivel oculto comparten el mismo tipo de información.

- **Salida:** es el conjunto de neuronas que transfiere la información que la red ha procesado hacia el exterior.

Los pasos para desarrollar un modelo de ANN aplicando un perceptrón multicapa para ejecutar análisis de regresión consisten en la compilación y preparación de datos para su entrenamiento y validación creando dos datasets separados y correspondientes a los atributos dependientes e independientes, la posterior selección de parámetros estructurales, el entrenamiento del modelo y, por último, la predicción del modelo para el consecuente análisis de resultados Jafari (2022).
El criterio clave que determina el éxito de un modelo de ANN es la elección de sus parámetros de entrenamiento y arquitectura. Estadísticamente, los modelos de ANN desarrollados se prueban para comprobar su precisión de predicción y, finalmente, se selecciona el modelo óptimo.
El paso inicial para desarrollar el modelo ANN es obtener la base de datos de entradas y salidas, se simula un control proporcional con diversos valores, de los cuales se obtiene un dataset de errores con sus correspondientes valores de control, estos valores se usaran para entrenar la red neuronal. El método de prueba y error se utiliza principalmente para la selección de parámetros. Los parámetros esenciales que se consideran son la eficiencia del aprendizaje, el momentum y el número de neuronas presentes en las capas ocultas, así como el número exacto de capas ocultas Ren et al. (2021).

El proceso para generar el dataset de entrenamiento, se ejecuta 50 diferentes valores de la constante proporcional que a de multiplicar el error para obtener una señal de control, en cada una de estos 50 experimientos se registran 100 datos de la señal de control y la señal de error, al final de este proceso se obtiene un dataset de 5000 datos de entrada que son los correspondientes a los valores de la señal de error, y 5000 datos de salida correspondientes a los valores de la señal de control.

Similar al caso de la neurona biológica, la neurona artificial recibe entradas de estímulo provenientes del sistema sensorial externo o de otras neuronas con las que esta interconectada. Para el caso del modelo que se propone en la Figura 2, recibe unas entradas de estímulo que puedan provenir del sistema sensorial externo o de otras neuronas con las cuales posee conexión.
La información recibida por la neurona es modificada por un vector de pesos sinápticos ω cuyo papel es emular la sinapsis existente entre las neuronas biológicas. Estos valores se pueden asimilar a ganancias que pueden atenuar o amplificar los valores que se desean propagar hacia la neurona.El parámetro θ_j se conoce como el bias o umbral de una neurona.

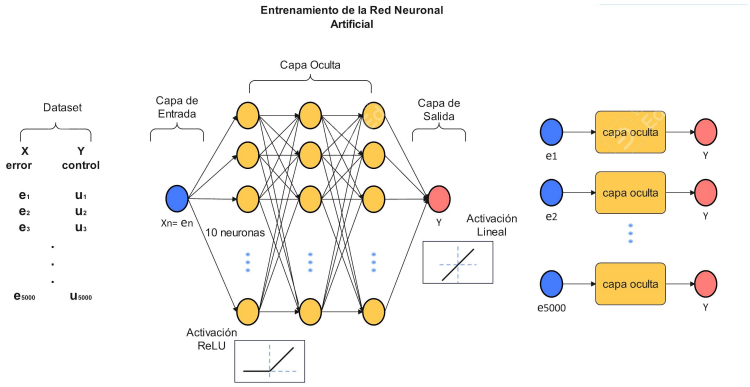


Fig. 2: Modelo de la Red neuronal artificial.

Los diferentes valores que recibe la neurona, modificados por los pesos sinápticos, se suman para producir algo denominado entrada neta que es la que va a determinar si la neurona se activa o no.
La activación o no de la neurona, depende la Función de Activación que evalua la entrada neta y se obtiene la salida de la red. Se propagan la salida de las neuronas hacia otras neuronas o puede ser la salida de la red.
La función de activación ReLU (Unidad Lineal Rectificada) se puede implementar solo con una simple comparación elemento por elemento de la matriz con umbral cero, tiene la ventaja de que acelera el

proceso de entrenamiento mediante una rapida convergencia del gradiente estocástico, además involucra operaciones menos costosas computacionalmente en comparación con otras funciones de activación como la exponencial y la función de tangente hiperbólica tanh Mirtaheri and Shahbazian (2022).
La entrada neta se puede representar con la ecuación (3)

$$Net_j = \sum_{i=1}^{n}(x_i\omega_{ji} + b_j) \tag{3}$$

La salida de la neurona artificial está determinada por una función de activación ($Fact$) descrita en (4), en las capas ocultas se usa una función RelU mientras en la función de salida es del tipo lineal.Los parámetros y estructura de la red neuronal artificial de este estudio se muestran en la Tabla 2

$$y_j = (Fact)_j(Net)_j \tag{4}$$

Se realiza un ajuste de los pesos mediante un descenso por la gradiente, el gradiente es un vector que indica en que direccion moverse en el espacio de pesos en cada iteración, los pesos se actualizan siguiendo la regla(5), donde η es la tasa de aprendizaje que es un hiperparámetro que controla que tan grande son los pasos que da el algoritmo al actualizar los pesos y sesgos, el valor asignado a este hiperparámetro es 0.001. La función de pérdida con respecto a la unidad de sesgo sigue el mismo concepto lo que da como resultado (6) Raschka (2023).

$$W \leftarrow W - \eta\nabla W \tag{5}$$

$$b \leftarrow b - \eta\nabla b \tag{6}$$

Tabla 2: Parámetros y estructura de la red neuronal.

Capa	Tipo	Neuronas	Funcion de Activación	Entradas (forma / variables)	Salidas (forma / variables)	Parametros entrenables
Entrada	-	1	-	(batch size,1) error, $u_t - 1$	Alimenta a capa oculta	-
Capa oculta	Densa	10	ReLU	(batch size,1)	(batch size,10)	30 pesos + 10 sesgos = 40
Salida	Densa	1	Lineal	(batch size,10)	(batch size,1)u_t (control)	10 pesos + 1 sesgos = 11

Algoritmo 1 Red Neuronal Artificial

Inicializar Conjunto de datos de entrenamiento $(X_{\text{train}}, y_{\text{train}})$
Inicializar pesos y sesgos de la red neuronal Definir arquitectura:
Entrada: error e Capa oculta: 10 neuronas con función de activación ReLU Capa de salida: una neurona lineal que estima u
for *cada muestra* (e, u) *en* (x_{train}, y_{train}) **do**

Propagación hacia adelante (Forward Pass):
Calcular la activación de la capa oculta:

$$z^{(1)} = W^{(1)}e + b^{(1)}, \quad a^{(1)} = \text{ReLU}(z^{(1)})$$

Calcular la salida de la red:

$$z^{(2)} = W^{(2)}a^{(1)} + b^{(2)}, \quad \hat{u} = z^{(2)}$$

Cálculo del error:

$$J = (u - \hat{u})^2$$

donde u es la acción de control real (proveniente del control proporcional) y $\hat{u}$ es la salida estimada por la ANN.
Retropropagación (Backward Pass):
Calcular gradientes de J respecto a los pesos y sesgos.
Propagar el error hacia atrás desde la salida hasta la entrada.
Actualización de parámetros:

$$W \leftarrow W - \eta \nabla W, \quad b \leftarrow b - \eta \nabla b$$

end
Salida: Red neuronal entrenada que aproxima la ley de control

El Algoritmo 1 explica en detalle que la red recibe como entrada el error que es la diferencia entre la referencia de velocidad y la velocidad actual del motor, en la capa oculta (10 neuronas) aplica una transformación no lineal con ReLU, La salida es una neurona lineal que genera la señal de control u.

Durante el entrenamiento, se compara la salida de la ANN con la acción de control real (obtenida de un controlador proporcional).Usando retropropagación y descenso de gradiente, se ajustan los pesos y sesgos para minimizar el error cuadrático medio como se presenta en (7):

$$MSE = \frac{1}{N}\sum_{i=1}^{n}(y_i - \bar{y_i})^2 \tag{7}$$

2.2. Red Neuronal Evolutiva

Las técnicas de los algoritmos evolutivos se basan en diversos procesos biológicos observados en la naturaleza. El proceso de comportamiento óptimo observado en la naturaleza es la evolución de las especies a lo largo de sus generaciones. Cada nueva generación de cualquier especie cambiará constantemente con respecto a la anterior e intentará adaptarse lo mejor posible al entorno cambiante. Las características que aparezcan en la nueva generación la harán más apta para sobrevivir.
El algoritmo genético (AG) intenta implementar el mismo comportamiento/proceso evolutivo de la naturaleza para encontrar la solución óptima.

La información genética de cualquier organismo vivo natural se almacena en un cromosoma y los genes son responsables de determinar las características de un individuo. Un gen representa una única

característica o variable de entrada independiente, el equivalente a este cromosoma individuo es el compuesto por un par de constante proporcional Kp y un offset, los cuales en conjunto producen un determinado valor de MSE entre la diferencia de la señal de salida y la referencia.
En cada iteración de un AG, los cromosomas o soluciones deben pasar por los siguientes procesos (términos relacionados específicamente con el AG):

- **Selección:** Se utiliza para elegir un número determinado de cromosomas, como progenitores, del conjunto de soluciones disponibles para el apareamiento/reproducción.
  Uno de los métodos más populares para seleccionar cromosomas progenitores es el método de selección por torneo. En este método, se escogen aleatoriamente algunos cromosomas del grupo de apareamiento y se les hace competir entre sí. La competencia se basa en los valores de la función objetivo de los cromosomas progenitores determinados, el ganador del torneo se reproduce Badar (2021).

- **Cruce:** El cruce es un proceso para generar descendencia mediante el apareamiento de los cromosomas progenitores.
  El metodo de cruce blend escoge genes al azar en el rango definido por los genes de los padres $x1, x2$, en el caso del estudio se trata de los parámetros Kp y offset. El rango de los genes de los hijos se define por (8), donde α es el parámetro que expande el rango de los genes de los padres, usualmente su valor es 0.5 Gridin (2021).

$$[x1 - \alpha(x2 - x1), x2 + \alpha(x2 - x1)] \tag{8}$$

- **Mutación:** La mutación se utiliza para introducir variaciones en un cromosoma existente.
  En una mutación gaussiana, todos los elementos del cromosoma mutan según la función de distribución gaussiana. Esta función tiene una media de cero y una varianza adaptativa. El efecto de la mutación disminuye a medida que aumentan las iteraciones. La distribución normal se expresa mediante (9), donde $\mu = 0$ y σ se evalúa como (10) donde k es la posición del elemento, T es el número total de iteraciones, t es el número de iteración actual, v_k^{max} y v_k^{min} son los límites de la variable de entrada Reina et al. (2020).

$$N(\mu, \sigma) \tag{9}$$

$$\sigma_k = \frac{T - t}{T} \cdot \frac{v_k^{max} - v_k^{min}}{3} \tag{10}$$

Tras completar la selección, el cruce y la mutación, se evalúan los valores de la función objetivo de la nueva población. El proceso se repite varias veces hasta que se cumple el criterio de parada. En la Figura 3 se puede ver el proceso general y en el Algoritmo 2 el proceso a mas detalle.

Algoritmo 2 Red Neuronal Evolutiva

Evaluación de fitness:

Dado un individuo $ind = (k_p, \text{offset})$:

Inicializar $\omega = 0, i = 0, mse = 0$

for $t = 1$ **to** 100 **do**

  Calcular error: $e = \omega_{ref} - \omega$

  Calcular control: $u = k_p \cdot e + \text{offset}$

  Actualizar motor: $(\omega, i) = MotorModel(u, \omega, i)$

  Acumular error: $mse = mse + (\omega_{ref} - \omega)^2$

end

return $mse/100$ `// Error cuadrático medio`

Operadores evolutivos:

Inicializar población P con $N = 50$ individuos aleatorios

for *generación* $= 1$ **to** 20 **do**

  for *individuo ind en P* **do**

    $ind.fitness = (ind)$; `// Evaluación del individuo`

  end

  Seleccionar padres mediante **torneo** (tamaño 3)

  Aplicar **cruza blend** con $\alpha = 0{,}5$

  Aplicar **mutación gaussiana** con $\mu = 0, \sigma = 0{,}1$ y probabilidad 0,1

  Generar nueva población P

end

Seleccionar mejor individuo por minimizacion de MSE

$best = (k_p^*, \textbf{offset}^*)$ como solución óptima

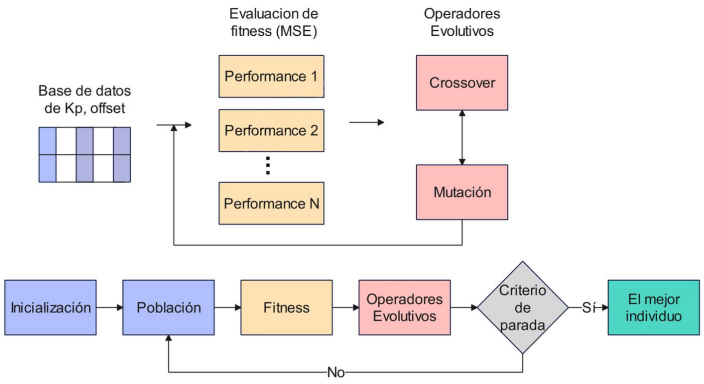


Fig. 3: Modelo de la Red neuronal evolutiva.

3. Resultados y Discusión

Para evaluar la efectividad de los controladores ANN y ENN se analizan las gráficas de las curvas de salida , control y fase además se utilizaron métricas fundamentales del análisis de sistemas dinámicos: error cuadrático medio MSE, tiempo de establecimiento, sobreimpulso y tiempo de subida. Estas medidas permiten cuantificar la diferencia entre la salida del motor y la señal de referencia, además de describir el comportamiento transitorio del sistema, evaluando su rapidez, precisión y estabilidad en la respuesta. Las curvas de respuesta de la Figura 4 muestra la señal de salida como velocidad angular en

radianes/segundo en función del paso del tiempo, ambas curvas presentan las características de un sistema sobreamortiguado con oscilasciones antes de alcanzar su valor estable. El tiempo de subida fue más rápido en el control ENN, mientras el de establecimiento fue mas rapido al aplicar el control ANN, el sobreimpulso resulta significativamente mayor en el control ENN. La velocidad angular al controlador ENN resulta ser mas preciso y cercano al valor de referencia que cuando se usa el control ANN.

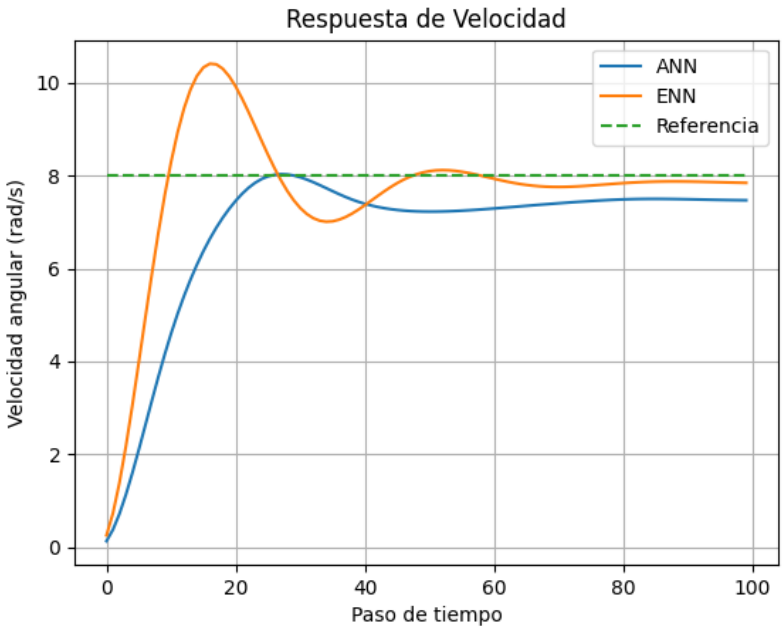


Fig. 4: Señales de Salida con control ANN y ENN.

La Figura 5 muestra el comportamiento de las señales de control al ingreso del motor y sus valores son del voltaje a lo largo del tiempo, se puede ver que el control ENN exige un valor más alto inicialmente y presenta mas oscilaciones en comparación a la señal de control al aplicarse un control de red neuronal artificial que posee una curva menos pronunciada y con menor valor.

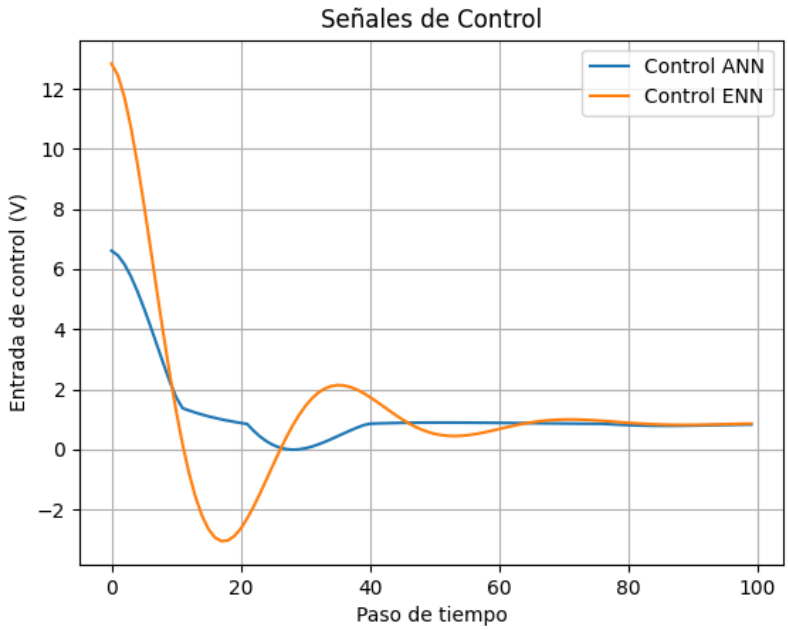


Fig. 5: Curvas de la ley de Control.

La Figura 6 indica el plano de fase donde se muestra el recorrido de la corriente respecto a la velocidad angular mientras el motor de corriente directa se estabiliza. Al usar el control ENN se requiere de una corriente alta y su curva converge suavemente mientrás que al probarse con el control ANN no se necesita de un corriente tan alta más presenta una ligera oscilación antes de terminar en el punto de equilibrio.

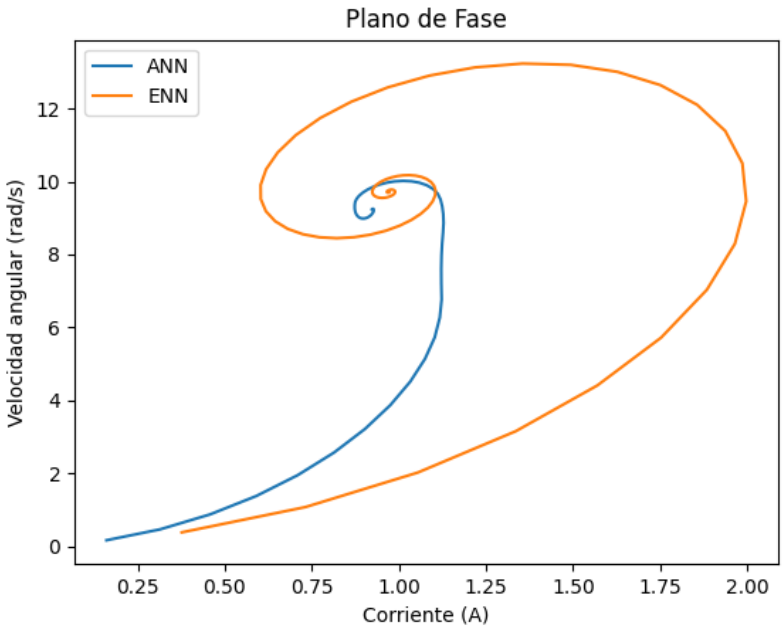


Fig. 6: Plano de Fase.

La Figura 7 presenta las métricas de cada uno de los controladores, se hace notar que el índice de error medio cuadrático, tiempo de subida y tiempo de establecimiento son menores al usar el controlador ENN en contraposición el porcentaje de sobreimpulso o overshoot es mucho manor al usarse el controlador ENN.

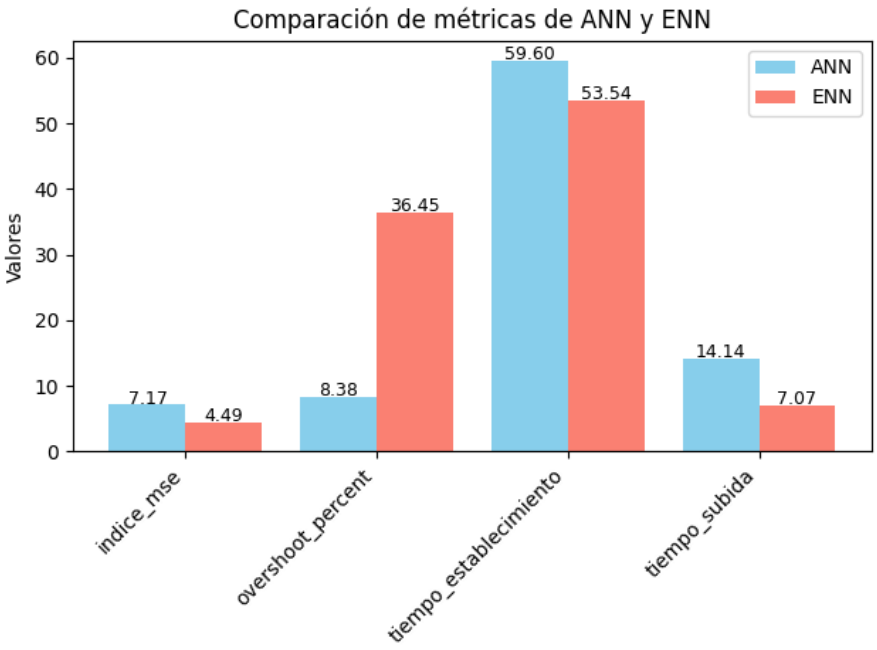


Fig. 7: Comparación de métricas.

4. Conclusiones

Se comparan dos maneras de diseñar el control para el motor de corriente directa mientras el ANN aprende de ejemplos de un control proporcional, el control ENN encuentra kp y offset óptimos usando algoritmos evolutivos por las cuales resulta fundamental aplicar correctamente la arquitectura y parámetros característicos para obtener un buen desempeño como controladores del motor de corriente continua

En la gráfica de plano de fase se puede observar que la tryectoria del controlador ANN no llega del todo al punto de equilibrio posee una ligera oscilación y una menor corriente cercana a 1.25 A, mientras que la trayectoria del controlador ENN tiene una oscilación mas amplia con una corriente alta que llega casi a los 2A, se concluye que la rapidez de convergencia del controlador ENN es mas rápida y cercana al valor de referencia aunque en desventaja tiene menos estabilidad y un mayor consumo energético antes de estabilizarse.

Los resultados obtenidos evidencian un desempeño considerablemente inferior del controlador ANN en comparación con el ENN. El error cuadrático medio (MSE) del modelo ANN con valor de 7.17 es mayor que el del ENN con 4.49, lo que refleja una menor capacidad de la red neuronal artificial para acercar la salida del motor a la señal de referencia. La respuesta del sistema bajo control ENN permaneció cercana a la referencia despues de su estabilización, con un sobreimpulso aproximado del 36.45 %, dando un mejor resultado el sobreimpulso del control ANN que alcanzó un 8.38 % del valor deseado. Adicionalmente, el sistema con el controlador ENN logró tanto el tiempo de establecimiento (53,54 segundos) como el tiempo de subida (7.07 segundos) con mejores resultados que el controlador ANN con el tiempo de establecimiento (59,6 segundos) como el tiempo de subida (14.014 segundos).

Referencias

A. Badar. *Evolutionary Optimization Algorithms*. CRC Press, 2021. ISBN 9781000462166. URL https://books.google.com.ec/books?id=f3g-EAAAQBAJ.

S. Das. Modeling and simulation of mechatronic systems using simscape. *Synthesis Lectures on Mechanical Engineering 2020-mar 09 vol. 5 iss. 1*, 5, mar 2020. doi: 10.2200/S00972ED1V01Y201912MEC024. URL libgen.li/file.php?md5=eb699709eb66d46a014cb0fc8a77ecc0.

Dorantes Benavidez Humberto, Zarate Santiago Aldo, Trejo Martínez Alfredo, Acosta Mendizábal Marco, and Dorantes Benavidez F. de J. El uso de las TIC´S como herramienta de simulación de una red neuronal en motores de corriente continúa como sistema de control. *Ciencia Latina Revista Científica Multidisciplinar*, 7(2):2565–2580., 2023. doi: 10.37811/cl_rcm.v7i2.5512.

A. Eiben and J. Smith. From evolutionary computation to the evolution of things. *Nature*, 521:476–82, 05 2015. doi: 10.1038/nature14544.

E. Galván and P. Mooney. Neuroevolution in deep neural networks: Current trends and future challenges. *IEEE Transactions on Artificial Intelligence*, 2(6):476–493, Dec. 2021. ISSN 2691-4581. doi: 10.1109/tai.2021.3067574. URL http://dx.doi.org/10.1109/TAI.2021.3067574.

P. Granados. *Robótica*. Pablo Arturo Pacheco Granados, 2025. URL https://books.google.com.ec/books?id=tSJzEQAAQBAJ.

I. Gridin. *Learning Genetic Algorithms with Python: Empower the performance of Machine Learning and AI models with the capabilities of a powerful search algorithm (English Edition)*. bpb, 2021. ISBN 9788194837756. URL https://books.google.com.ec/books?id=8OEdEAAAQBAJ.

R. Jafari. *Hands-On Data Preprocessing in Python: Learn how to effectively prepare data for successful data analytics*. Packt Publishing, 2022. ISBN 9781801079952. URL https://books.google.com.ec/books?id=H6lVEAAAQBAJ.

P. Jayaraman. *Python-Based Evolutionary Algorithms for Engineers*. Educohack Press, 2025. ISBN 9789361525711. URL https://books.google.com.ec/books?id=75lIEQAAQBAJ.

L. Larsen, A. Schuster, J. Kim, and M. Kupke. Path planning of cooperating industrial robots using evolutionary algorithms. *Procedia Manufacturing*, 17:286–293, 2018. ISSN 2351-9789. doi: https://doi.org/10.1016/j.promfg.2018.10.048. URL https://www.sciencedirect.com/science/article/pii/S2351978918311648. 28th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2018), June 11-14, 2018, Columbus, OH, USAGlobal Integration of Intelligent Manufacturing and Smart Industry for Good of Humanity.

R. Manglik, E. Experts, and E. Community. *Electric Vehicles*. EduGorilla Publication, 2024. ISBN 9789367842508. URL https://books.google.com.ec/books?id=inQ7EQAAQBAJ.

S. Mirtaheri and R. Shahbazian. *Machine Learning: Theory to Applications*. CRC Press, 2022. ISBN 9781000737691. URL https://books.google.com.ec/books?id=Nud4EAAAQBAJ.

L. T. Nien. Ti u hiu sut tc đ đng c servo dc s dng b điu khin pid kt hp mng n-ron. 2024.

S. Raschka. *Machine Learning con PyTorch y Scikit-Learn*. Marcombo, 2023. ISBN 9788426736253. URL https://books.google.com.ec/books?id=NumwEAAAQBAJ.

D. Reina, A. Córdoba, and A. Nozal. *Algoritmos Genéticos con Python: Un enfoque práctico para resolver problemas de ingeniería*. Marcombo, 2020. ISBN 9788426730688. URL https://books.google.com.ec/books?id=yExOEAAAQBAJ.

J. Ren, W. Shen, Y. Man, and L. Dong. *Applications of Artificial Intelligence in Process Systems Engineering*. Elsevier, 2021. ISBN 9780128217436. URL https://books.google.com.ec/books?id=GID-DwAAQBAJ.

F. Sabry and G. Costa. *Cerebro Artificial: Dar a los robots la inteligencia para tareas complejas*. Tecnologías Emergentes En Tecnologías De La Información Y Las Comunicaciones [Spanish]. Mil Millones De Conocimientos [Spanish], 2022. URL https://books.google.com.ec/books?id=Rv6TEAAAQBAJ.

Sheıkh Debes Rahaf and Kara Tolgay. Design and simulation of a pid neural network controller for pmdc motor speed and position control. *Avrupa Bilim ve Teknoloji Dergisi*, (44):46–50, 2022. doi: 10.31590/ejosat.1222247.

I. Singh. *BLDC (Brushless Direct Current) Motors*. Pencil, 2024. ISBN 9789362639073. URL https://books.google.com.ec/books?id=xQsOEQAAQBAJ.

M. Swarnkar and S. Rajput. *Artificial Intelligence for Intrusion Detection Systems*. CRC Press, 2023. ISBN 9781000967555. URL https://books.google.com.ec/books?id=hKMIEQAAQBAJ.