

Comparación de un control difuso y un control por redes neuronales basados en un controlador PI de un circuito RC

Leandro Llontop Herrera,* Juan Contreras Avendaño,† Fidel Castro Suazo,‡ and Franck Condori Sinche§

Escuela Profesional de Ingeniería Mecatrónica, Universidad Nacional de Ingeniería, Perú

(Dated: July 7, 2024)

This research investigates the performance of two advanced control strategies—fuzzy logic control and neural network control—both integrated with a Proportional-Integral (PI) controller, applied to an RC circuit. The primary objective is to evaluate and compare the efficacy of these control methodologies in terms of stability, response time, and robustness. The study begins with the mathematical modeling of the RC circuit, followed by the design and implementation of the fuzzy logic and neural network controllers. Simulations are conducted to analyze the transient and steady-state behaviors of the circuit under each control scheme. Results indicate that while both controllers enhance the performance of the traditional PI controller, the neural network-based control demonstrates superior adaptability and precision in handling non-linearities and uncertainties within the system. Conversely, the fuzzy logic controller excels in simplicity and ease of implementation. This comparative analysis provides valuable insights into the selection of appropriate control techniques for RC circuits in various practical applications.

I. INTRODUCTION

En la ingeniería de control, los circuitos RC (resistor-capacitor) representan el inicio en una amplia gama de aplicaciones de plantas, desde la filtración de señales hasta la regulación de sistemas electrónicos. Este laboratorio se enfoca en el diseño e implementación de dos tipos de controladores avanzados para un circuito RC: uno basado en lógica difusa y otro basado en redes neuronales. La lógica difusa, inspirada en el comportamiento humano para manejar la incertidumbre y la imprecisión, ofrece un enfoque robusto y flexible para el control de sistemas no lineales y complejos. Por otro lado, las redes neuronales, que emulan el funcionamiento del cerebro humano, proporcionan una capacidad adaptativa y de aprendizaje que permite mejorar el rendimiento del sistema a través de la experiencia.

La combinación de estos enfoques permite explorar y comparar distintas estrategias de control, evaluando su eficacia y eficiencia en la regulación de un circuito RC. El objetivo principal de este laboratorio es analizar y demostrar las ventajas y limitaciones de cada método, proporcionando una comprensión más profunda de su aplicabilidad en el diseño de sistemas de control avanzados.

II. MARCO TEÓRICO

A. Lógica Difusa

La lógica difusa, introducida por Lotfi Zadeh en 1965, es una extensión de la lógica clásica que permite trabajar con valores de verdad que se encuentran entre 0 y

1. Esta capacidad la hace muy adecuada para manejar incertidumbres y datos imprecisos.

En la lógica clásica, una proposición es verdadera o falsa. En la lógica difusa, una proposición puede ser parcialmente verdadera o falsa, con un valor de verdad que se representa como un número real en el intervalo [0, 1].

La lógica difusa es una técnica utilizada en sistemas de control para manejar la incertidumbre y la imprecisión presentes en muchos sistemas del mundo real. A diferencia de los métodos de control convencionales, que requieren modelos matemáticos precisos, la lógica difusa permite representar y manejar conocimiento impreciso mediante reglas lingüísticas.

1. Conjuntos Difusos

Un conjunto difuso A en un espacio universal X se define por una función de membresía $\mu_A : X \rightarrow [0, 1]$.

$$\mu_A(x) = \begin{cases} 1 & \text{si } x \text{ pertenece completamente a } A, \\ 0 & \text{si } x \text{ no pertenece a } A, \\ \alpha & \text{si } x \text{ pertenece parcialmente a } A, \alpha \in (0, 1). \end{cases}$$

2. Operaciones en Conjuntos Difusos

Las operaciones básicas en conjuntos difusos incluyen la intersección y la unión, definidas como:

$$\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x)),$$

$$\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x)).$$

* marco.llontop.h@uni.pe

† juan.contreras.a@uni.pe

‡ fidel.castro.s@uni.pe

§ franck.condori.s@uni.pe

3. Sistema de Inferencia Difusa Tipo Mamdani

Un sistema de inferencia difusa de tipo Mamdani consta de cuatro pasos principales: fuzzificación, evaluación de reglas, agregación de resultados y desfuzzificación.

4. Fuzzificación

La fuzzificación es el proceso de transformar valores de entrada nítidos en valores difusos utilizando funciones de membresía.

$$\mu_A(x) = \text{fuzzificación del valor de entrada } x$$

5. Evaluación de Reglas

Las reglas difusas se expresan en la forma de "SI ENTONCES". Una regla típica puede ser:

$$\text{SI } x_1 \text{ ES A1 Y } x_2 \text{ ES A2 ENTONCES y ES B}$$

Cada regla se evalúa utilizando el operador lógico difuso adecuado (generalmente, la conjunción es el mínimo).

$$\text{Salida de la regla} = \min(\mu_{A1}(x_1), \mu_{A2}(x_2))$$

6. Agregación de Resultados

Los resultados de todas las reglas activadas se combinan para formar una salida difusa.

$$\mu_B(y) = \max(\text{Salida de cada regla aplicada a } y)$$

7. Desfuzzificación

La desfuzzificación es el proceso de convertir un conjunto difuso en un valor nítido. Un método común es el centroide:

$$y^* = \frac{\int y \cdot \mu_B(y) dy}{\int \mu_B(y) dy}$$

8. Ejemplo

Consideremos un sistema difuso con dos entradas x_1 y x_2 , y una salida y . Supongamos que tenemos dos reglas difusas:

Regla 1: SI x_1 ES A1 Y x_2 ES A2 ENTONCES y ES B1

Regla 2: SI x_1 ES A3 Y x_2 ES A4 ENTONCES y ES B2

La evaluación y agregación de estas reglas nos da la salida difusa $\mu_B(y)$, y finalmente aplicamos desfuzzificación para obtener el valor nítido y^* .

9. Control por Lógica Difusa

En el control por lógica difusa, se definen reglas lingüísticas que relacionan las entradas del sistema con las acciones de control. Por ejemplo, un controlador difuso puede tener reglas del tipo:

1. Si el error es grande y el cambio en el error es pequeño, entonces aumentar la salida.
2. Si el error es pequeño y el cambio en el error es grande, entonces disminuir la salida.

La salida del controlador difuso se calcula mediante la combinación de estas reglas utilizando operadores de inferencia difusa como el mínimo, el máximo o el promedio ponderado. La lógica difusa se aplica en una variedad de áreas, incluyendo control de velocidad de motores, sistemas de navegación autónoma, sistemas de climatización, entre otros.

B. Control por Redes Neuronales

Para los propósitos de este laboratorio, veremos las redes neuronales como aproximadores de funciones. Como se muestra en la Figura 1, tenemos una función desconocida que deseamos aproximar con la que se entrena la red. Queremos ajustar los parámetros de la red para que produzca la misma respuesta que la función desconocida, si se aplica la misma entrada a ambos sistemas. En el control por Redes Neuronales la función desconocida puede corresponder a un sistema que estamos tratando de controlar, en cuyo caso la red neuronal será el modelo de planta identificado la función desconocida también podría representar el inverso de un sistema que estamos tratando de controlar, en cuyo caso la red neuronal se puede utilizar para implementar el controlador.

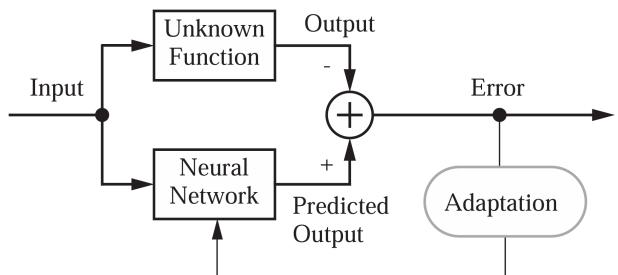


Figure 1. Red Neuronal como aproximador de función

1. Neurona de entrada multiple

Típicamente, una neurona tiene más de una entrada. Una neurona con n entradas se muestra en la Figura 2. Las entradas individuales p_1, p_2, \dots, p_n son ponderadas por los elementos correspondientes $w_{1,1}, w_{1,2}, \dots, w_{1,n}$ de la matriz de pesos W .

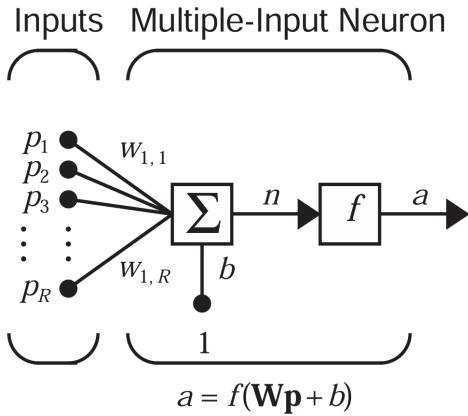


Figure 2. Red Neuronal de entrada multiple

La neurona tiene un sesgo b , que se suma a las entradas ponderadas para formar la entrada neta n :

$$n = w_{1,1}p_1 + w_{1,2}p_2 + \dots + w_{1,R}p_R + b$$

Esta expresión se puede escribir en forma matricial:

$$n = \mathbf{W}\mathbf{p} + b$$

donde la matriz W para el caso de una sola neurona tiene solo una fila. Ahora la salida de la neurona se puede escribir como

$$a = f(\mathbf{W}\mathbf{p} + b)$$

Se ha adoptado una convención particular en la asignación de los índices de los elementos de la matriz de pesos. El primer índice indica el destino particular de la neurona para ese peso. El segundo índice indica la fuente de la señal alimentada a la neurona. Así, los índices en $w_{1,2}$ dicen que este peso representa la conexión a la primera (y única) neurona desde la segunda fuente. Una neurona de entrada múltiple usando esta notación se muestra en la Figura 3.

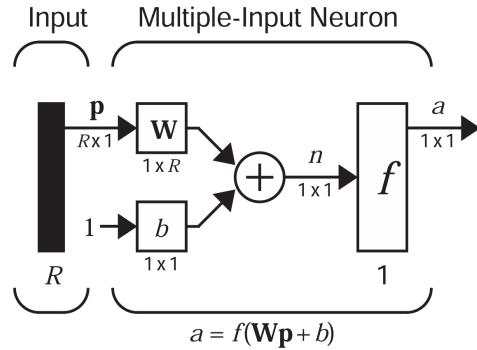


Figure 3. Neurona con R entradas

Como se muestra en la Figura 3, el vector de entrada p está representado por la barra vertical sólida a la izquierda. Las dimensiones de p se muestran debajo de la variable como $R \times 1$, indicando que la entrada es un vector único de R elementos. Estas entradas van a la matriz de pesos W , que tiene R columnas pero solo una fila en este caso de una sola neurona. Una constante 1 entra a la neurona como una entrada y se multiplica por un sesgo escalar b . La entrada neta a la función de transferencia f es $Wp + b$, que es la suma del sesgo b y el producto Wp . La salida de la neurona a es un escalar en este caso. Si tuviéramos más de una neurona, la salida de la red sería un vector.

2. Una Capa de Neuronas

Una red de capa única de S neuronas se muestra en la Figura 4. Nota que cada una de las R entradas está conectada a cada una de las neuronas y que la matriz de pesos ahora tiene S filas.

La capa incluye la matriz de pesos, los sumadores, el vector de sesgos b , las cajas de funciones de transferencia y el vector de salida a . Algunos autores se refieren a las entradas como otra capa, pero nosotros no lo haremos aquí. Cada elemento del vector de entrada p está conectado a cada neurona a través de la matriz de pesos W . Cada neurona tiene un sesgo b_i , un sumador, una función de transferencia f y una salida a_i . Tomados en conjunto, las salidas forman el vector de salida a .

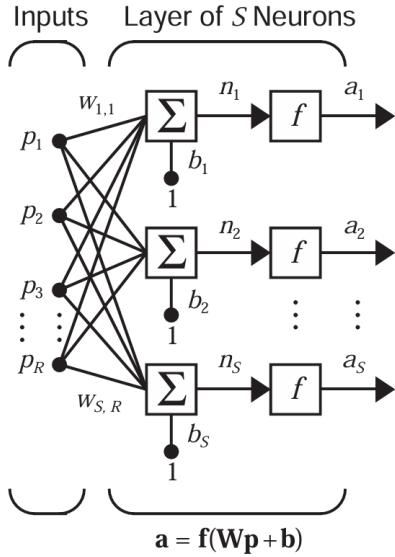


Figure 4. Capa de S neuronas

Los elementos del vector de entrada entran a la red a través de la matriz de pesos W :

$$W = \begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & \ddots & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{pmatrix}$$

3. Entrenamiento de Redes Multicapa

Ahora que se sabe que las redes multicapa son aproximadores universales, el siguiente paso es determinar un procedimiento para seleccionar los parámetros de la red (pesos y sesgos) que mejor aproximen una función dada. El procedimiento para seleccionar los parámetros para un problema dado se llama entrenamiento de la red. En esta sección, se da un procedimiento de entrenamiento llamado retropropagación, que se basa en el descenso de gradiente.

Para redes multicapa, la salida de una capa se convierte en la entrada de la siguiente capa. Las ecuaciones que describen esta operación son

$$a^{(m+1)} = f^{(m+1)}(W^{(m+1)}a^{(m)} + b^{(m+1)})$$

Donde M es el número de capas en la red. Las neuronas en la primera capa reciben entradas externas. Las salidas de las neuronas en la última capa se consideran las salidas de la red:

$$a = a^{(M)}$$

Índice de Desempeño:

El algoritmo de retropropagación para redes multicapa es un procedimiento de optimización de descenso de gradiente en el cual minimizamos un índice de desempeño de

error cuadrático medio. El algoritmo se proporciona con un conjunto de ejemplos de comportamiento adecuado de la red:

$$\{(p_1, t_1), (p_2, t_2), \dots, (p_Q, t_Q)\}$$

donde p_q es una entrada a la red, y t_q es la salida objetivo correspondiente. A medida que cada entrada se aplica a la red, la salida de la red se compara con el objetivo. El algoritmo debe ajustar los parámetros de la red para minimizar el error cuadrático sumado

$$F(x) = \sum_{q=1}^Q (t_q - a_q)^2$$

donde x es un vector que contiene todos los pesos y sesgos de la red. Si la red tiene múltiples salidas, esto se generaliza a

$$F(x) = \sum_{q=1}^Q (t_q - a_q)^T (t_q - a_q)$$

Utilizando una aproximación estocástica, reemplazaremos el error cuadrático sumado por el error en el último objetivo:

$$\hat{F}(x) = (t(k) - a(k))^T (t(k) - a(k)) = e^T(k) e(k)$$

donde la expectativa del error cuadrático ha sido reemplazada por el error cuadrático en la iteración k .

El algoritmo de descenso más empinado para el error cuadrático medio aproximado es

$$w_{i,j}^{(m)}(k+1) = w_{i,j}^{(m)}(k) - \alpha \frac{\partial \hat{F}}{\partial w_{i,j}^{(m)}}$$

$$b_i^{(m)}(k+1) = b_i^{(m)}(k) - \alpha \frac{\partial \hat{F}}{\partial b_i^{(m)}}$$

donde α es la tasa de aprendizaje.

III. COMPORTAMIENTO DE LA PLANTA

Un circuito RC es un circuito eléctrico que consiste en un resistor R y un capacitor C en serie. Este tipo de circuito es fundamental en muchas aplicaciones de filtrado y temporización debido a su sencilla estructura y comportamiento predecible.

IV. FUNCIÓN DE TRANSFERENCIA

La función de transferencia de un circuito RC se puede derivar a partir de sus ecuaciones diferenciales. La ecuación diferencial que describe el voltaje en el capacitor es:

V. CONTROLADOR PI "EXPERTO"

$$V_{in}(t) = V_R(t) + V_C(t)$$

Donde:

$$V_R(t) = R \cdot I(t) \quad y \quad V_C(t) = \frac{1}{C} \int I(t) dt$$

Aplicando la transformada de Laplace y asumiendo condiciones iniciales cero, obtenemos:

$$V_{in}(s) = R \cdot I(s) + \frac{1}{C} \cdot \frac{I(s)}{s}$$

Factoreando $I(s)$:

$$V_{in}(s) = \left(R + \frac{1}{Cs} \right) I(s)$$

La función de transferencia $H(s)$ del circuito, definida como la relación entre la salida $V_{out}(s)$ (voltaje en el capacitor) y la entrada $V_{in}(s)$, es:

$$H(s) = \frac{V_{out}(s)}{V_{in}(s)} = \frac{\frac{1}{Cs}}{R + \frac{1}{Cs}} = \frac{1}{RCs + 1}$$

Si definimos $\tau = RC$ (la constante de tiempo del circuito), la función de transferencia se simplifica a:

$$H(s) = \frac{1}{\tau s + 1}$$

Consideremos un circuito RC donde $\tau = 1$. En este caso, la función de transferencia se simplifica aún más:

$$H(s) = \frac{1}{s + 1}$$

Esta función de transferencia indica que el circuito actúa como un filtro de primer orden con una constante de tiempo $\tau = 1$ s.

A. Tiempo de Establecimiento

El tiempo de establecimiento (t_s) de un sistema es el tiempo requerido para que la respuesta transitoria de un sistema decaiga y permanezca dentro de un margen especificado (generalmente 2% o 5

$$t_s \approx 5\tau$$

En nuestro caso, con $\tau = 1$ s:

$$t_s = 5 \times \tau = 5 \times 1 = 5 \text{ s}$$

Por lo tanto, el tiempo de establecimiento de este circuito RC es de 5 segundos.

Como se observó en la sección anterior, el tiempo de establecimiento de la planta es de 5 segundos. El sistema de control debe mejorar este tiempo haciendo que la respuesta sea más rápida. Además, se requiere pocas oscilaciones y un sobreceso máximo pequeño en la respuesta transitoria. Obviamente, se desea que el error en estado estacionario sea cero y que el sistema sea robusto frente a perturbaciones.

De esta manera, se propone diseñar un controlador PI que cumpla con los requerimientos antes definidos, los cuales se pueden expresar cuantitativamente de la siguiente forma:

$$t_s = 0.8 \text{ seg} \quad \zeta = 0.9285$$

El valor de ζ se obtuvo con un previo análisis al considerar los requerimientos de pocas oscilaciones y sobreceso máximo pequeño. En seguida, se pueden obtener los polos del lazo cerrado mediante la teoría de control clásica. El tiempo de establecimiento con un criterio del 2% se puede calcular mediante la ecuación 1 y los polos se pueden expresar en función del factor de amortiguamiento y frecuencia natural, tal como lo muestra la ecuación 2.

$$ts = \frac{4}{\zeta \omega_n} \quad (1)$$

$$s_{1,2} = -\zeta \omega_n \pm j\omega_n \sqrt{1 - \zeta^2} \quad (2)$$

Con la ecuación 1 se puede obtener la frecuencia natural, para luego calcular los polos deseados mediante 2.

$$\zeta \omega_n = 4/0.8 = 5$$

$$\omega_n = 5/0.9285 = 5.38503 \text{ rad/s}$$

Entonces, los polos deseados del sistema son:

$$s_{1,2} = -5 \pm j2$$

El sistema de control propuesto se muestra en la Figura 5, el cual cuenta con un bloque de control proporcional-integral que se expresa según la función de transferencia en 3

$$G_c(s) = K_c \left(1 + \frac{1}{T_i s} \right) \quad (3)$$

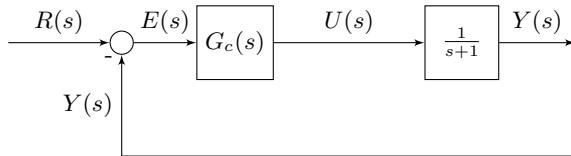


Figure 5. Diagrama de bloques del lazo de control

Para obtener las constantes del controlador PI, es necesario analizar la ecuación característica del sistema de control.

$$G_c(s)G(s) + 1 = 0$$

$$K_c \left(1 + \frac{1}{T_i s}\right) \frac{1}{s+1} + 1 = 0$$

$$s^2 + (K_c + 1)s + \frac{K_c}{T_i} = 0 \quad (4)$$

Como se busca que la raíces de la ecuación 4 sean los polos deseados antes hallados, fácilmente se pueden obtener las constantes del controlador.

$$Kc + 1 = -2(\text{Re}(s)) = 10$$

$$K_i = \frac{K_c}{T_i} = \text{Re}(s)^2 + \text{Im}(s)^2 \rightarrow T_i = 9/29$$

$$Kc = 9 \quad K_i = 29$$

En Simulink se diagrama el sistema de control propuesto y las constantes K_c y K_i halladas se colocan en el bloque PID de Simulink. En la Figura 6, se puede verificar que el controlador diseñado logra que el sistema cumpla con los requerimientos planteados, siendo lo más notable la reducción del tiempo de establecimiento.

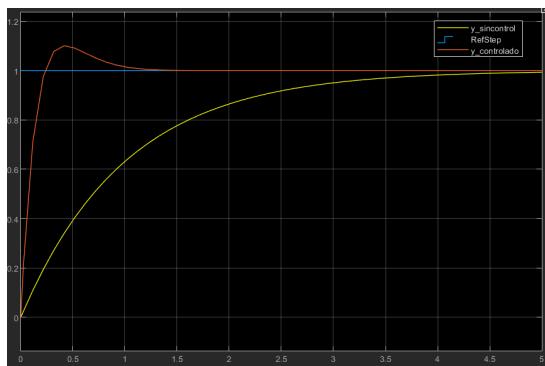


Figure 6. Gráficos de la señal de referencia y las respuestas de la planta sin controlar y con control PID

Como este control actuará como el modelo "experto" para el diseño tanto del controlador difuso como para el de redes neuronales, es necesario extraer la data sobre el error, la derivada del error y la señal de control del sistema, tal como se muestra en la Figura 7

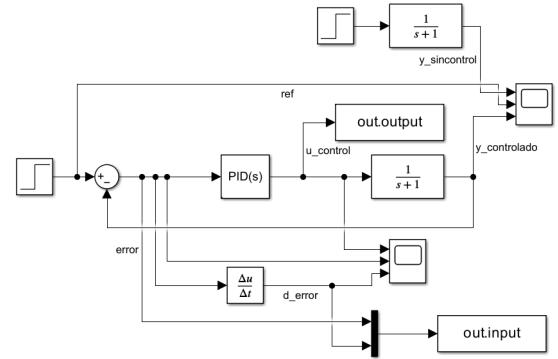


Figure 7. Sistema de control PI en Simulink

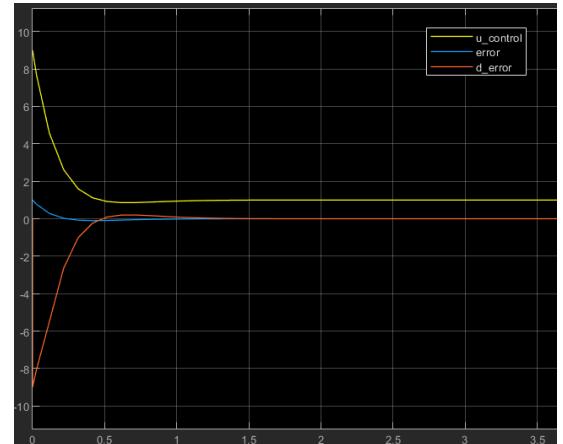


Figure 8. Gráficas del error, su derivada y la señal de control

VI. DISEÑO DE CONTROLADOR DIFUSO

La data extraída del error, su derivada y la señal de control sirve para diseñar un sistema de lógica difusa que controle el circuito RC con los mismos requerimientos antes mencionados.

Una de las ventajas del control difuso es su facilidad de diseño e implementación, siempre y cuando se cuente con un sistema experto confiable. De esta manera, en el diseño solo bastará con definir las funciones de membresía para las dos entradas y la salida, y establecer las reglas de implicación basadas en el comportamiento observado en la Figura 8.

- Primero, se definen las variables lingüísticas para las dos entradas y la salida del sistema de control difuso.

1. Error (e): Negative (N), Zero (Z), Positive Small (PS), Positive Medium (PM), Positive Big (PB), Negative-Zero-Positive (NZP).
 2. Derivada del Error (de): Zero (Z), Negative 1 (N1), Negative 2 (N2), Negative 3 (N3), Negative 4 (N4), Negative 5 (N5), Negative 6 (N6), donde N6 representa valores negativos "grandes".
 3. Salida de control (u): Less than one (Lone), Uno (ONE), Positive 1 (P1), Positive 2 (P2), Positive 3 (P3), Positive 4 (P4), Positive 5 (P5), donde P5 representa valores positivos "grandes" y P1 los positivos mayores y cercanos a uno.
- Para cada variable lingüística, se define una función de membresía que describe cuán bien se ajusta una entrada a cada etiqueta lingüística. El universo de discurso para el error es $[-0.100578 \ 1]$ y las funciones de membresía que permitirán realizar la "fuzzification" se muestran en la Figura 9.

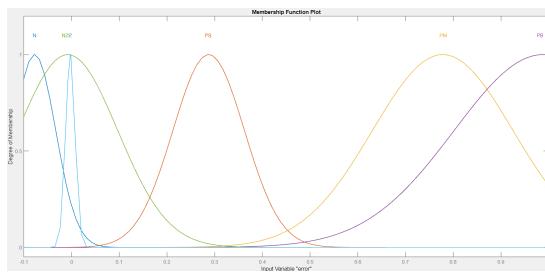


Figure 9. Funciones de membresía para el Error

El universo de discurso para la derivada del error es $[-8.99979 \ 0.200576]$ y las funciones de membresía se muestran en la Figura 10.

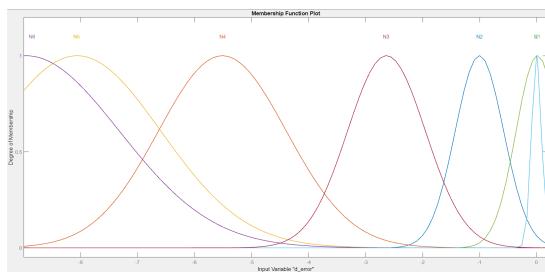


Figure 10. Funciones de membresía para la Derivada del Error

El universo de discurso para la señal de control es $[0.862142 \ 9]$ y las funciones de membresía se muestran en la Figura 10.

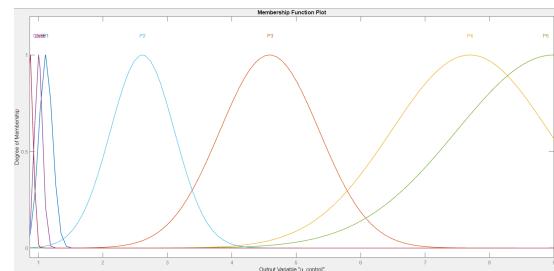


Figure 11. Funciones de membresía para la Señal de Control

- Reglas del Sistema: Se establecen un conjunto de reglas difusas que relacionan las variables de entrada (error y derivada del error) con la salida de control (u).

1. If error is N and derror is N2 then ucontrol is Lone.
2. If error is PS and derror is N4 then ucontrol is P3.
3. If error is PM and derror is N5 then ucontrol is P4.
4. If error is NZP and derror is N1 then ucontrol is ONE.
5. If error is Z and derror is N3 then ucontrol is ONE.
6. If error is Z and derror is Z then ucontrol is ONE.
7. If error is N and derror is N1 then ucontrol is Lone.
8. If error is PB and derror is N6 then ucontrol is P5.
9. If error is N and derror is N2 then ucontrol is ONE.
10. If error is N and derror is N1 then ucontrol is ONE.
11. If error is Z and derror is N1 then ucontrol is P1.

Finalmente, se tiene el sistema de lógica difusa con implicación Mandani mostrado en la Figura 12, el cual ya tiene definidas las reglas difusas y las funciones de membresía.

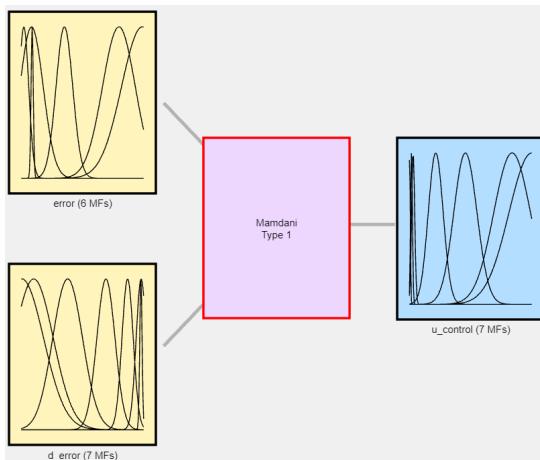


Figure 12. Sistema de lógica difusa para control de circuito RC

El archivo .fis obtenido del diseñador de sistemas difusos de MATLAB se puede incluir en el bloque de *Fuzzy Control* de Simulink. En la Figura 13 se muestra el diagrama de control difuso en Simulink y en la Figura 14 se puede verificar como el control difuso diseñado también cumple con los requerimientos del tiempo de establecimiento y sobreimpulso máximo.

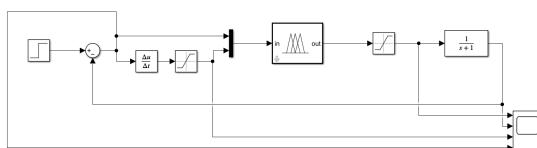


Figure 13. Diagrama de control difuso en Simulink

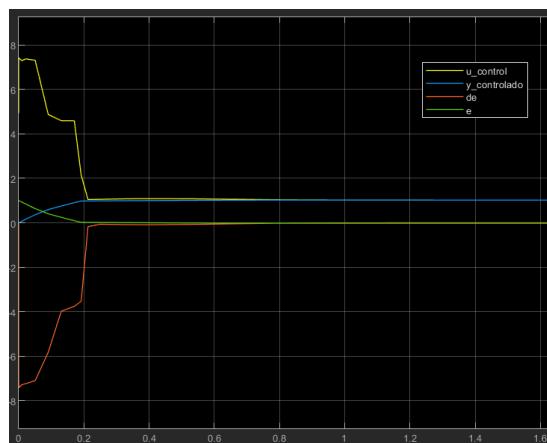


Figure 14. Gráficas del error, derivada del error, señal de control y la respuesta del sistema

VII. DISEÑO DE CONTROLADOR POR REDES NEURONALES

Para el diseño del controlador basado en redes neuronales se usa los datos de la respuesta, con un controlador PID, como datos de entrenamiento de la red de neuronas propuesta.

A. Datos de control PI

Se obtiene los datos de la respuesta del sistema, con control PI, usando la función simout para poder usar el vector de datos en el editor de Matlab como se muestra en la figura 15.

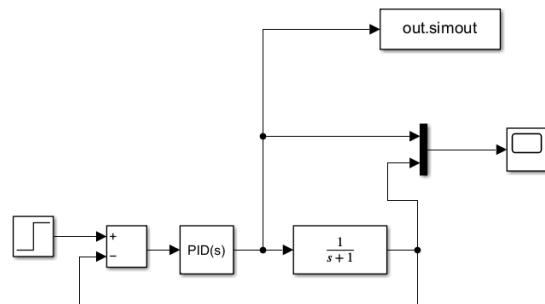


Figure 15. Obteniendo vector de datos de control PI

Con el vector de datos se realiza un código en el editor de matlab y se entrena una red neuronal de tipo 231. Se obtienen los siguientes resultados

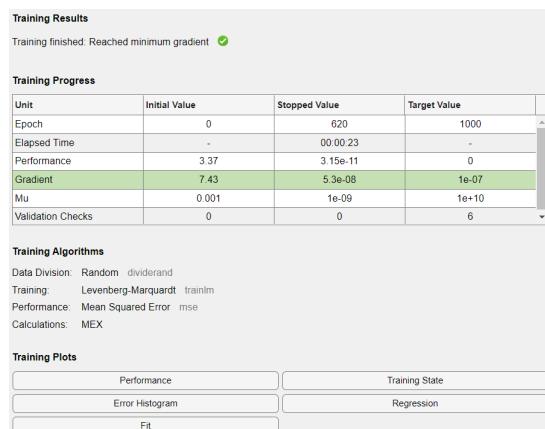


Figure 16. Resultados de entrenamiento

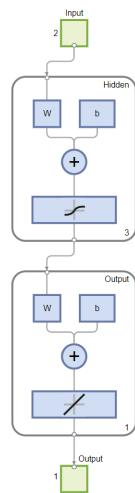


Figure 17. Diagrama de Red Neuronal

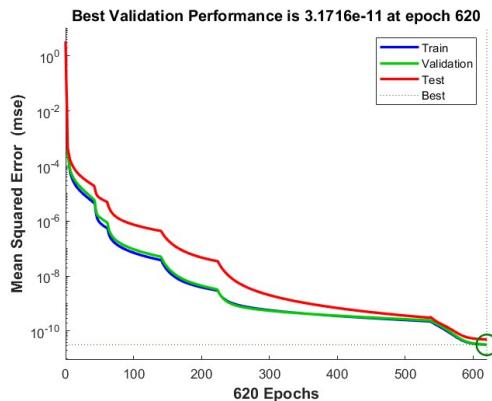


Figure 18. Validación de performance

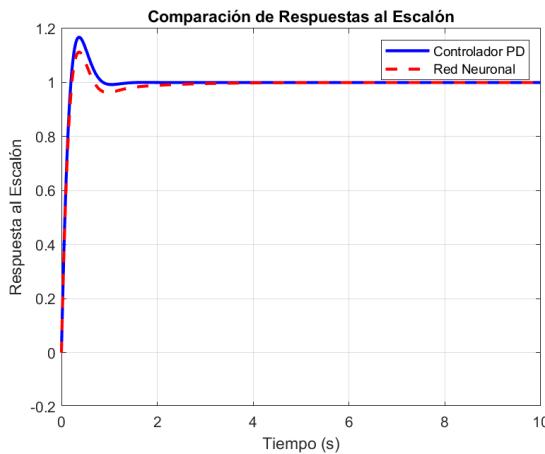


Figure 19. Respuesta de red entrenada

VIII. IMPLEMENTACIÓN DE CONTROL DIFUSO

A. Fuzzy System Designer- Labview

Fuzzy System Designer es una herramienta dentro del entorno de desarrollo LabVIEW de National Instruments, que permite a los ingenieros y científicos diseñar, simular y analizar sistemas de lógica difusa. LabVIEW (Laboratory Virtual Instrument Engineering Workbench) es un entorno de programación gráfica ampliamente utilizado en aplicaciones de adquisición de datos, control y automatización, y análisis de sistemas.

Procedimiento

- Ya se tenían las entradas y las salidas en Matlab (Figura 20), el reto en este paso fue pasar las funciones de pertenencia gaussianas de **MATLAB** a **LABVIEW**, ya que en MATLAB se define la MF gaussiana con dos parámetros c y σ , estas se reemplazan en la siguiente ecuación:

$$f(x; c, \sigma) = e^{-\frac{(x-c)^2}{2\sigma^2}}$$

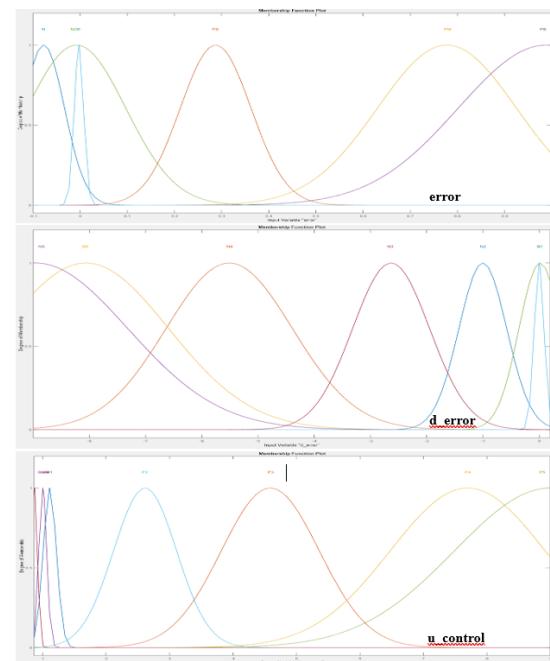


Figure 20. Entradas y salidas desarrolladas en MATLAB

En LABVIEW necesitamos cuatro parámetros, toma dos valores extremos y dos valores extendiendo el pico de la función gaussiana, para definir la MF gaussiana (Figura 21).

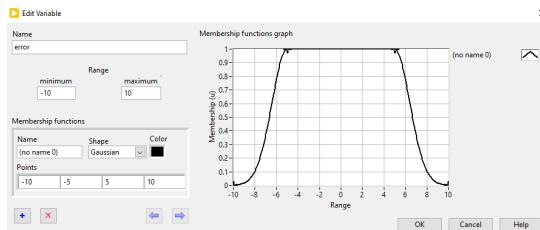


Figure 21. Edición de variables en LABVIEW

Se desarrollo un código en MATLAB donde se obtenían de las funciones gaussianas tres valores, las 2 de los extremos con una consideración que la función gaussiana tendría un valor de 0.004 y en el medio que corresponde con el valor c (media o esperanza matemática), esto se aprecia en la Figura ??.

```

1 clc;clear all
2 % Definición de los parámetros
3 x = -10:0.0001:10;
4 c = 0.287357420859890044 ;
5
6 % Crear un vector de puntos para x
7 x = linspace(-0.8, 0.8, 1000); % Ajusta el rango y el número de puntos si es necesario
8
9 % Calcular la función gaussiana
10 y = exp(-(x - c)^2 / (2 * sigma^2));
11
12 % Encontrar el valor de x cuando la función es igual a 1
13 x_at_y_1 = x(a);
14 % Mostrar los valores de x cuando la función es igual a 0.004
15 x_target = 0.004;
16 x_at_y_004_positive = c + sqrt(2 * sigma^2) * log(y_target);
17 x_at_y_004_negative = c - sqrt(2 * sigma^2) * log(y_target);
18
19 % Mostrar los resultados
20 fprintf('El valor de x cuando la función es igual a 1: %f\n', x_at_y_1);
21 fprintf('Los valores de x cuando la función es igual a 0.004: %f, %f\n', x_at_y_004_positive, x_at_y_004_negative);
22
23
24
```

Figure 22. Código MATLAB

- Obteniendo esos 3 parámetros se procedió a rellanar las 4 casillas, el de los dos entradas y el de la salida (Figura 23 y 24) , en donde hicimos coincidir los dos del medio con el valor de la esperanza matemática (c) y se obtuvieron resultados similares a la Figura 20.

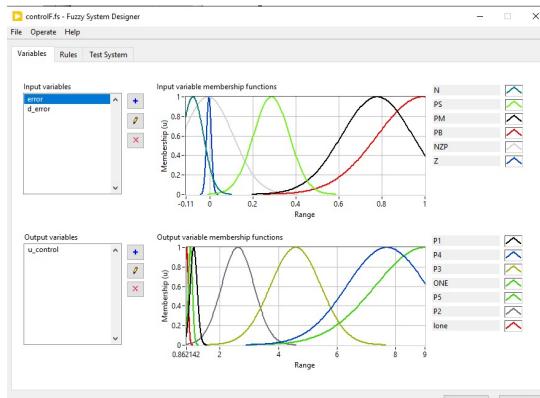


Figure 23. Entrada error y salida u control

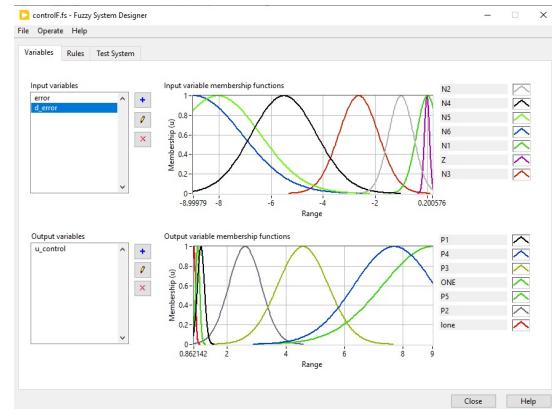


Figure 24. Entrada d_error y salida u control

- luego se procedió a completar las reglas (Figura 25 y 26).

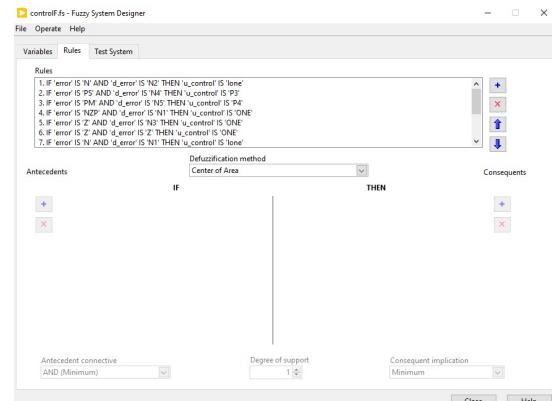


Figure 25. Reglas del Sistema Difuso Parte 1

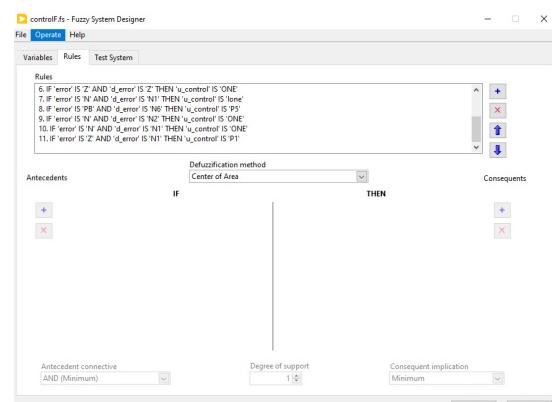


Figure 26. Reglas del Sistema Difuso Parte 2

- Luego en la pestaña de sistema de pruebas se aprecian las superficie de control generada (Figura 27 y 28)

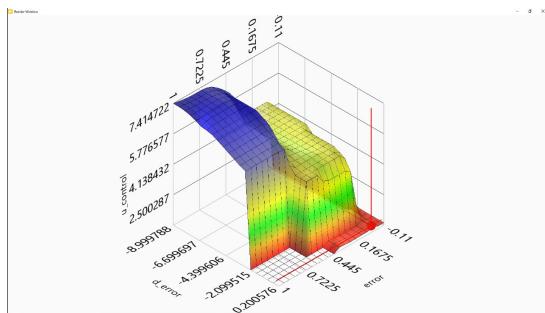


Figure 27. Superficie de Control Primera vista

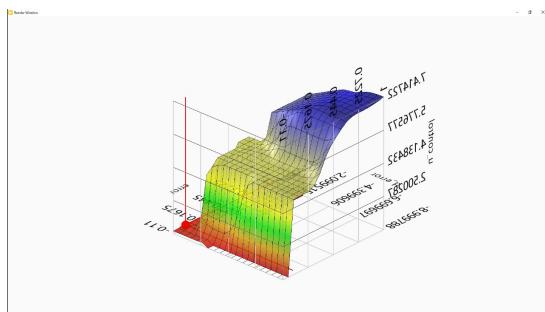


Figure 28. Superficie de Control Segunda vista

B. Panel Frontal - Labview

Se colocaron 4 visualizadores de gráficas (Figura 29), en donde podemos visualizar la **salida**, la **entrada** (señal de control), el **error** y la **variación de error**.

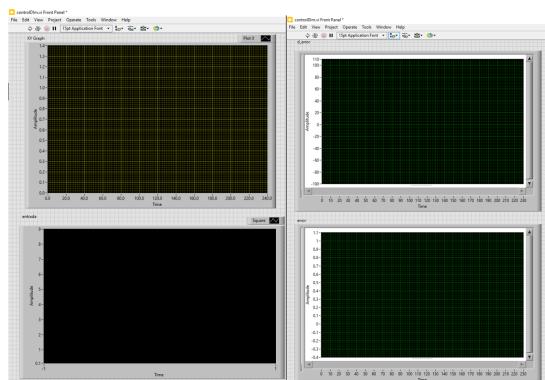


Figure 29. Panel Frontal - LABVIEW

C. Diagrama de Bloques - Labview

Lo que podríamos resaltar de este diagrama de bloques (Figura 30) son dos bloques nuevos utilizados, el FL Fuzzy Load System nos permite que carguemos el archivo que generamos con nuestras reglas, MF entradas y MF

salidas , y este se conecta con el bloque FL Fuzzy Controller, este último interactúa con la planta, entregandole una señal de control y leyendo el error y 1 variación de error.

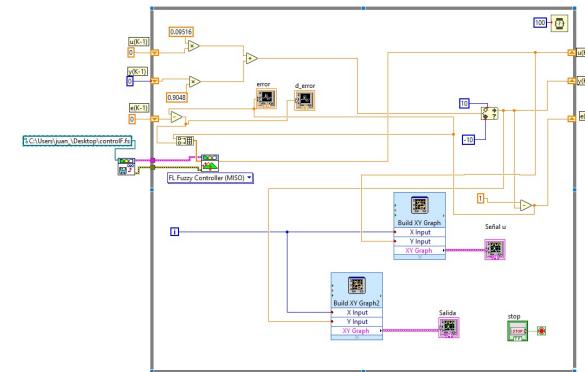


Figure 30. Diagrama de bloques - LABVIEW

D. Resultados

LabVIEW Prueba 1

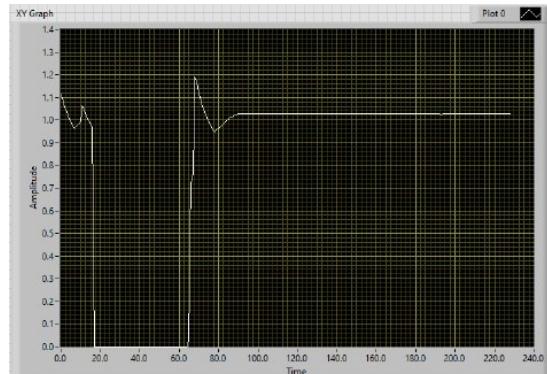


Figure 31. Gráfico de salida de la planta

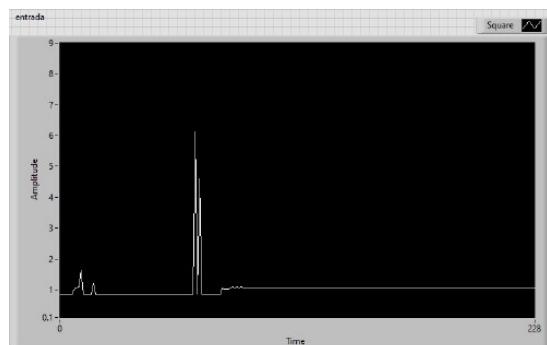


Figure 32. Gráfico de la Señal de Control



Figure 33. Gráfico de la variación del error

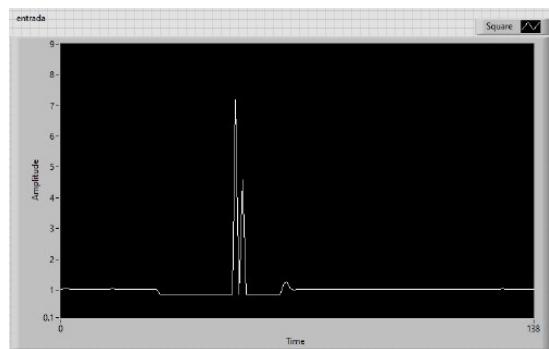


Figure 36. Gráfico de la Señal de Control



Figure 34. Gráfico del error



Figure 37. Gráfico de la variación del error



Figure 38. Gráfico del error

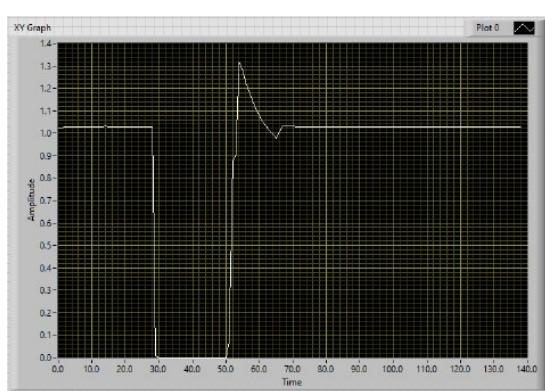


Figure 35. Gráfico de salida de la planta

Pruebas con el osciloscopio

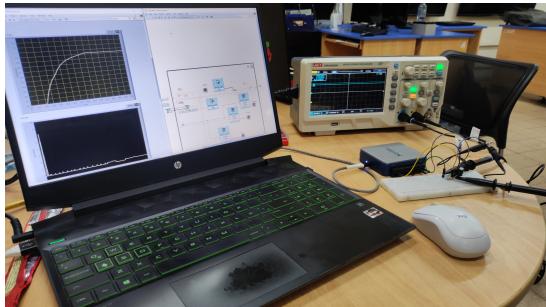


Figure 39. Conexión de Computadora, DAQ y Osciloscopio a un Circuito RC



Figure 40. Prueba física 1 CH1: 1 V/div CH2: 200mV/div



Figure 41. Prueba física 1 CH1: 1 V/div CH2: 500mV/div

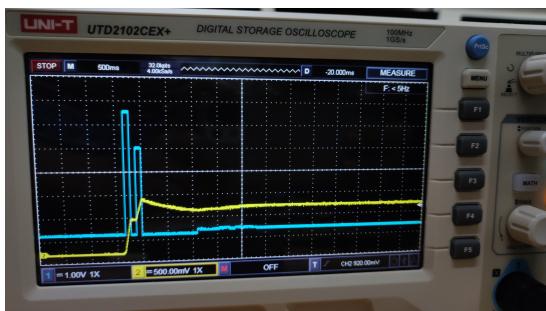


Figure 42. Prueba física 2 CH1: 1 V/div CH2: 500mV/div

IX. CONSLUSIONES

- El control basado en redes neuronales demostró una gran capacidad de replicación de su salida con los datos de entrenamiento proporcionados. Esto los hace particularmente útiles en aplicaciones donde las dinámicas del sistema pueden cambiar con el tiempo o donde se necesita una alta precisión en el control.
- El uso de redes neuronales permitió una optimización del control del circuito RC, ajustando los pesos y sesgos durante el proceso de entrenamiento para minimizar el error de seguimiento.
- En aplicaciones prácticas, la elección entre un controlador difuso y uno basado en redes neuronales dependerá de las necesidades específicas del sistema, los recursos disponibles y las condiciones operativas.
- La implementación de un controlador difuso suele ser más sencilla en términos de diseño y configuración inicial, ya que se basa en reglas predefinidas y conocimiento experto. Sin embargo, la implementación de redes neuronales, aunque más compleja y demandante en términos de recursos computacionales, puede ofrecer un mejor desempeño una vez que está bien entrenada.

- El controlador difuso depende del conocimiento y la experiencia del experto que define las reglas y las funciones de membresía. Un experto puede desarrollar un conjunto de reglas robusto, siempre habrá un margen de error inherente debido a la subjetividad en la definición de estas reglas.

X. RECOMENDACIONES

- La combinación de lógica difusa y redes neuronales puede proporcionar un enfoque híbrido, aprovechando las fortalezas de ambos métodos. Esto podría mejorar aún más el desempeño del sistema de control, permitiendo una mayor adaptabilidad y robustez.
- La efectividad del control difuso en un circuito RC puede verse afectada por la escala de la señal de control utilizada. Es crucial diseñar y ajustar adecuadamente las funciones de pertenencia y las reglas difusas para asegurar que la señal de control, esté dentro del rango operativo óptimo del circuito y no provoque sobrecargas, y tambien verificar si el DAQ te puede dar esa magnitud de señal.

REFERENCIAS

- K. J. Hunt, D. Sbarbaro, R. Zbikowski, and P. J. Gawthrop, “Neural Networks for Control Systems Survey,” *Survey Paper*, Pergamon Press Ltd, 1992.
- Faiber Ignacio Robayo B., Ana María Barrera F., y Laura Camila Polanco C. “Desarrollo de un controlador basado en redes neuronales para un sistema multivariante de nivel y caudal,” *Revista Ingeniería y Región*, vol. 14, no. 2, pp. 43-54, 2015. (Development of a controller based on neural networks for a multivariable level and flow system).
- Raghad Ali Mejeed, Ahmed K. Jameil, y Husham Idan Hussein, “Harmonic Amplification Damping Using a DSTATCOM-based Artificial Intelligence Controller,” *International Journal of Sensors, Wireless Communications and Control*, vol. 9, pp. 1-10, 2019. Send Orders for Reprints to reprints@benthamscience.net.
- Óscar Eduardo López-Manchola, Juan David Gómez-Buitrago, Andrés Eduardo Gaona-Barrera, y Nelson Leonardo Díaz-Aldana, “Diseño y simulación de un control neuronal aplicado a un convertidor flyback para la regulación de tensión,” *Revista UIS Ingenierías*, vol. 20, no. 4, pp. 111-126, 2021. [Enlace](<https://revistas.uis.edu.co/index.php/revistauisinsin>)

Appendix A: Primer Anexo

1. Código de entrenamiento de red neuronal

```

close all; clc;

% Definir los parámetros del controlador PI
Kp = 8.1;
Ki = 42.25;
Kd = 0;

% Definir la planta
numG = 1;
denG = [1 1];
G = tf(numG, denG);

% Definir el controlador PD
numC = [Kd Kp Ki];
denC = [1 0];
C = tf(numC, denC);

% Sistema en lazo cerrado
sys_open = series(C, G);
sys_closed = feedback(sys_open, 1);

% Simulación de la respuesta al escalón
t = 0:0.01:9.99; % tiempo de simulación
[y, t] = step(sys_closed, t);

% Generar el error (entrada de la red neuronal)

```

```

setpoint = ones(size(t)); % referencia
error = setpoint - y; % error de seguimiento

% Calcular la integral del error
dt = t(2) - t(1); % suponer que el paso de
                   % tiempo es constante
error_int = cumsum(error) * dt; % integral del
                                 % error

u=out.simout;

% Normalización de datos
inputs = [error'; error_int']; % error en la
                                % primera fila, integral en la segunda
targets = u';

% Normalizar entradas y salidas
[input_norm, input_settings] = mapminmax(inputs)
;
[target_norm, target_settings] = mapminmax(
targets);

% Definir la red neuronal
hiddenLayerSize = 3;
net = fitnet(hiddenLayerSize);

% Cambiar las funciones de activación
net.layers{1}.transferFcn = 'tansig'; % Capa
                                       % oculta
net.layers{2}.transferFcn = 'purelin'; % Capa de
                                         % salida

% Dividir los datos en conjuntos de
      % entrenamiento, validación y prueba
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Entrenar la red neuronal
[net, tr] = train(net, input_norm, target_norm);

% Verificar el desempeño de la red
outputs_norm = net(input_norm);
outputs = mapminmax('reverse', outputs_norm,
target_settings);
errors = gsubtract(targets, outputs);
performance = perform(net, targets, outputs);

% Mostrar las curvas de entrenamiento
figure;
plotperform(tr);

% Generar nuevas entradas para la red neuronal
inputs_nn = [error'; error_int'];
inputs_nn_norm = mapminmax('apply', inputs_nn,
input_settings);

% Obtener la señal de control generada por la
      % red neuronal
u_nn_norm = net(inputs_nn_norm);
u_nn = mapminmax('reverse', u_nn_norm,
target_settings);

% Simulación de la planta con la señal de
      % control de la red neuronal
[y_nn, t_nn] = lsim(G, u_nn, t);

% Comparar las respuestas del controlador PD y
      % la red neuronal
figure;

```

```

plot(t, y, 'b', 'LineWidth', 2); hold on;
plot(t_nn, y_nn, 'r--', 'LineWidth', 2);
xlabel('Tiempo (s)');
ylabel('Respuesta al Escal n');
legend('Controlador PD', 'Red Neuronal');
title('Comparaci n de Respuestas al Escal n');
grid on;

% Pesos y sesgos de la capa oculta
hiddenLayerWeights = net.IW{1,1};
hiddenLayerBiases = net.b{1};

% Pesos y sesgos de la capa de salida
outputLayerWeights = net.LW{2,1};
outputLayerBiases = net.b{2};

% Mostrar los valores
disp('Pesos de la capa oculta:');
disp(hiddenLayerWeights);
disp('Sesgos de la capa oculta:');
disp(hiddenLayerBiases);

disp('Pesos de la capa de salida:');
disp(outputLayerWeights);
disp('Sesgos de la capa de salida:');
disp(outputLayerBiases);

1. Obtenci n de parmetros de las MF gaussianas

lc,clear all

```

```

% Definici n de los par metros
sigma = 0.0749883998040164 ;
c = 0.287357426085706;

% Crear un vector de puntos para x
x = linspace(0.8, 9, 1000); % Ajusta el rango y el n mero de puntos si es necesario

% Calcular la funci n gaussiana
y = exp(-((x - c).^2) / (2 * sigma^2));

% Encontrar el valor de x cuando la funci n es igual a 1
x_at_y_1 = c; % La funci n gaussiana es igual a 1 solo en su centro

% Encontrar los valores de x cuando la funci n es igual a 0.004
y_target = 0.004;
x_at_y_004_positive = c + sqrt(-2 * sigma^2 * log(y_target));
x_at_y_004_negative = c - sqrt(-2 * sigma^2 * log(y_target));

% Mostrar los resultados
fprintf('El valor de x cuando la funci n es igual a 1: %f\n', x_at_y_1);
fprintf('Los valores de x cuando la funci n es igual a 0.004: %f, %f\n',
       x_at_y_004_negative, x_at_y_004_positive);

```

ENLACE AL VIDEO DE EXPLICACI N:
Link del video del laboratorio N°4.