Diseño e implementación de una planta de nivel, controlada mediante redes neuronales y lógica difusa, destinada al laboratorio de control industrial de la Universidad de las fuerzas Armadas ESPE

Juan Pablo León Calderón[1] José Carlos Garcés Pico[1]
1. Departamento de Eléctrica y Electrónica, Universidad de las Fuerzas Armadas - ESPE Latacunga
plcj1988@hotmail.com, josegarces4512@gmail.com.

RESUMEN

El presente proyecto describe la aplicación de métodos alternativos de control inteligente, para el monitoreo y control de una planta de nivel. Dichos métodos implementados, se basan en la teoría de la lógica difusa (Fuzzy Logic) y redes neuronales (Neuronal Networks), así como también una breve descripción del software empleado (LabVIEW, MATLAB). Cabe destacar que la única forma de interacción entre la aplicación y el usuario, es mediante su interfaz gráfica, la cual está inspirada en el diseño y funcionalidad de Toolboxs propios del software, tales como: Neuronal Networks (NN) y Fuzzy Logic.

PALABRAS CLAVE: Sistemas de control, Control automático, Software Labview, Matlab

ABSTRACT

This project describes the use of alternative methods of intelligent control to monitor and control plant level. They implemented these methods are based on the theory of fuzzy logic (Fuzzy Logic) and neural networks (Neural Networks) , as well as a brief description of the software used (LabVIEW , MATLAB) . Note that the only form of interaction between the application and the user, through its graphical interface, which is inspired by the design and functionality of toolboxs own software, such as: Neural Networks (NN) and Fuzzy Logic.

KEYWORDS: Control systems, automatic Control, Software Labview, Matlab

I. FUNDAMENTOS TEÓRICOS

Inteligencia Artificial: Muchas veces escuchamos hablar sobre la inteligencia artificial, pero ¿Qué es realmente? A lo largo de la historia son numerosas las definiciones que se han dado sobre este tema; algunas de ellas son:

- "Capacidad que tienen las máquinas para realizar tareas que en el momento son realizadas por seres humanos" [1]

- "Campo de estudio que se enfoca en la explicación y emulación de la conducta inteligente en fusión de procesos computacionales basados en la experiencia y el conocimiento continuo del ambiente"

Técnicas de la inteligencia artificial

Como ya hemos dicho la inteligencia artificial se basa en el conocimiento. Existen tres modelos que los investigadores han utilizado de manera tradicional para la manipulación del mismo:

-***Programación heurística (Lógica difusa)***: Se basa en el modelo de comportamiento humano y su estilo para resolver problemas complejos .Existen varios tipos de programas que incluyen algoritmos heurísticos.

-***Redes neuronales:*** Representación abstraída del modelo neuronal del cerebro humano. Las redes están formadas por un gran número de elementos simples y por sus interconexiones.

-***Evolución artificial (algoritmos genéticos):*** Su modelo está basado en el proceso genético de la evolución natural, propuesto por Charles Darwin. Se utilizan sistemas simulados en computador que evolucionan mediante operaciones de reproducción, mutación y cruce.

En este trabajo nos centramos en la lógica difusa y

redes neuronales, que a continuación explicaremos con más detalle.

Lógica Difusa

Una de las disciplinas matemáticas con mayor aplicación en la actualidad es la Lógica Difusa. [2] Desde su aparición en la década de los 60's hasta nuestros días, las aplicaciones de la Lógica Difusa se han ido consolidando, paulatinamente al comienzo, y con un desbordado crecimiento en los últimos años.

Las principales razones para tal difusión quizás sean la sencillez conceptual de los Sistemas basados en Lógica Difusa, su facilidad para adaptarse a casos particulares con pocas variaciones de parámetros, su habilidad para combinar en forma unificada expresiones lingüísticas con datos numéricos, y el no requerir de algoritmos muy sofisticados para su implementación.

Fundamentos de Lógica Difusa

La lógica difusa se basa en lo relativo de lo observado. Este tipo de lógica toma dos valores aleatorios, pero contextualizados y referidos entre sí. Es una metodología que proporciona una manera simple y elegante de obtener una conclusión a partir de una información de entrada ambigua, imprecisa o incompleta, en general la lógica difusa modela como una persona toma decisiones basada en información con las características mencionadas, en esto se diferencia de la lógica convencional que trabaja con información bien definida y precisa.

La lógica difusa se adapta mejor al mundo real en el que vivimos, e incluso puede comprender y funcionar con nuestras expresiones, del tipo "hace mucho calor", "no es muy alto", "el ritmo del corazón está un poco acelerado", etc. La clave de esta adaptación al lenguaje, se basa en comprender los cuantificadores de nuestro lenguaje ("mucho", "muy" y "un poco") en los ejemplos mencionados.

Esta lógica permite tratar información imprecisa, como estatura alta, media o baja de una persona, tal como se observa en la **Figura 1.** Así, por ejemplo, se puede considerar a una persona que mida 2 metros, claramente como una persona alta, si previamente se ha tomado el valor de una persona de estatura baja y se ha establecido en 1 metro.

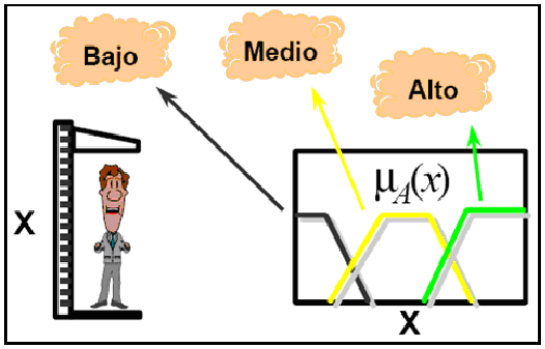

Figura 1. Visión de la lógica difusa

La Fusificación: Es la que nos permite Calcular el grado de pertenencia de cada conjunto difuso (Valor lingüístico).

Defusificación: Es el cambio de cada elemento del conjunto difuso a un solo número o valor.

Redes Neuronales

Definición

Al igual que con la inteligencia artificial, existen multitud de definiciones para las redes neuronales. Algunas de ellas son:

- Una nueva forma de computación, inspirada en modelos biológicos.

- Un modelo matemático compuesto por un gran número de elementos procesales organizados en niveles [3].

Ventajas que ofrecen:

Debido a que presentan un gran número de características similares a las del cerebro humano, las redes neuronales son capaces de aprender de la experiencia, de abstraer características esenciales a partir de entradas que presentan información irrelevante, de generalizar de casos anteriores a nuevos casos…etc. Todo esto permite su aplicación en un gran número de áreas muy

diferenciadas.

Las principales ventajas que representan son:

-Aprendizaje Adaptativo: Capacidad de aprender a realizar tareas basadas en un entrenamiento o en una experiencia inicial.
-Auto-organización: Una red neuronal puede crear su propia organización o representación de la información que recibe mediante una etapa de aprendizaje.

-Tolerancia a fallos: La destrucción parcial de una red conduce a una degradación de su estructura; sin embargo, algunas capacidades de la red se pueden retener, incluso sufriendo un gran daño.

-Operación en tiempo real: Los cómputos neuronales pueden ser realizados en paralelo; para esto se diseñan y fabrican máquinas con hardware especial para obtener esta capacidad.

-Fácil inserción dentro de la tecnología existente: Se pueden obtener chips especializados para redes neuronales que mejoran su capacidad en ciertas tareas. Ello facilitará la integración modular en los sistemas existentes.

Estructura de una Neurona [4]

Para describir la estructura de una red neuronal, primero describimos la estructura de lo que denominamos neurona.

Una neurona es la unidad básica de la red. La podemos describir comparándola con una neurona biológica, ya que el funcionamiento será similar.

En la **Figura 2** vemos una neurona biológica, ésta está formada por sinapsis, axón, dentritas y cuerpo. En la parte inferior tenemos una neurona artificial que es una unidad de procesamiento de la información, es un dispositivo simple de cálculo que ante un vector de entradas proporciona una única salida.

Sabiendo ya que la neurona es la unidad básica de la red, podemos definir una red neuronal como modelos matemáticos inspirados en sistemas biológicos, adaptados y simulados en computadoras convencionales.

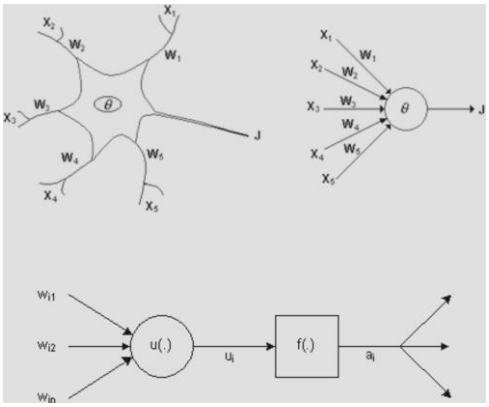

Figura 2. Similitud de Neuronas

El soma*:* es la parte central redonda donde se realizan casi todas las funciones lógicas de la neurona.

El axón: es una fibra nerviosa conectada directamente con el soma y que sirve como canal de salida. El axón usualmente está muy ramificado para permitir su conexión a un gran número de neuronas.

Las dendritas*:* son las entradas de información a la neurona. Son un grupo de fibras muy ramificadas y de forma irregular que se conectan directamente al soma. Se calcula entre 103 y 104 el número de dendritas en una neurona, permitiendo que esta reciba información de un gran grupo de otras neuronas.

Las sinapsis*:* son contactos especializados entre los axónes y las dendritas de diferentes neuronas. Estas sinapsis pueden cambiar la polaridad de los potenciales provenientes de otras neuronas y en estos casos se suele hablar de naturaleza excitadora o inhibidora según sea su función para la excitación o bloques de la neurona. Se considera que el almacenamiento de la información está concentrado en estas conexiones sinápticas.

Entradas y salidas: pueden ser clasificadas en dos grandes grupos, binarias o continuas. Las neuronas binarias (digitales) sólo admiten dos valores posibles. En general en este tipo de

neurona se utilizan los siguientes dos alfabetos {0,1} o {-1,1}. Por su parte, las neuronas continuas (analógicas) admiten valores dentro de un determinado rango, que en general suele definirse como [-1, 1]. La selección del tipo de neurona a utilizar depende de la aplicación y del modelo a construir**.**

Pesos sinápticos*:* El peso sináptico W_{ij} define la fuerza de una conexión sináptica entre dos neuronas, la neurona presináptica i y la neurona postsináptica j. Los pesos sinápticos pueden tomar valores positivos, negativos o cero. En caso de una entrada positiva, un peso positivo actúa como excitador, mientras que un peso negativo actúa como inhibidor. En caso de que el peso sea cero, no existe comunicación entre el par de neuronas.

Mediante el ajuste de los pesos sinápticos la red es capaz de adaptarse a cualquier entorno y realizar una determinada tarea.

Clasificación

Para la clasificación de las redes neuronales vamos a seguir dos tipos de aplicaciones:

-Según su arquitectura
-Según el aprendizaje

Según la arquitectura: La arquitectura de una red **Figura 3** consiste en la disposición y conexionado de las neuronas. Podemos distinguir en una red, el número de capas, el tipo de las capas, que pueden ser ocultas o visibles, de entrada o de salida y la direccionalidad de las conexiones de las neuronas.

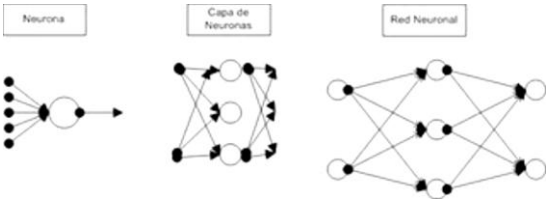

Figura 3. Según su arquitectura

Todo esto nos sirve para clasificarlas en:

-Redes Monocapa: cuentan con una capa de neuronas, que intercambia señales con el exterior y que constituyen a un tiempo la entrada y la salida del sistema. Una de las redes más representativas de este modelo es la red de Hopfield **Figura 4**, que ha tenido una gran influencia en el desarrollo posterior de redes neuronales.

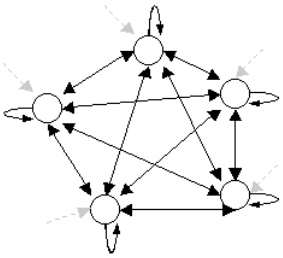

Figura 4. Arquitectura de una red de Hopfield

-*Redes Multicapa*: están formadas por dos o más capas de neuronas conectadas entre ellas ver **Figura 5**.

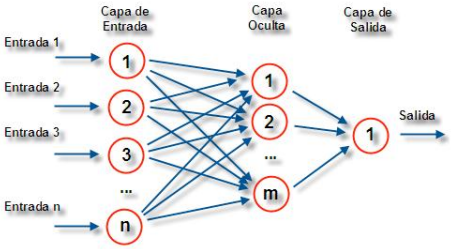

Figura 5. Red multicapa

Dependiendo de cómo sean estas conexiones podemos hacer otra subdivisión:

1. Redes con conexiones hacia delante: Este tipo de redes contienen solo conexiones entre capas hacia delante. Esto implica que una capa no puede tener conexiones a una que reciba la señal antes que ella en la dinámica de la computación.

2. Redes con conexiones hacia atrás: En este tipo de redes pueden existir conexiones de capas hacia atrás y por tanto la información puede regresar a capas anteriores en la dinámica de la red

Según el aprendizaje: El aprendizaje se basa en el entrenamiento de la red con patrones. El proceso de aprendizaje se basa en que la red ejecute los patrones de forma iterativa hasta que se muestren respuestas satisfactorias. Es decir, los pesos sinápticos se ajustan para dar respuestas óptimas para el conjunto de patrones de entrenamiento.

Podemos distinguir 3 tipos de aprendizaje:

-*Aprendizaje Supervisado:* la red dispone de los patrones de entrada y de salida que queremos obtener para esa entrada, y en función de ellos se modifican los pesos de las sinopsis para ajustar la entrada a la salida.

-*Aprendizaje No Supervisado:* consiste en no proporcionar a la red los patrones de salida, sino sólo los de entrada y dejar que la red los clasifique en función de características comunes que encuentre entre ellos.

-*Aprendizaje Híbrido:* No se proporcionan los patrones objetivo, sino que sólo se dice si la respuesta acierta o falla ante un patrón de entrada.

II. DISEÑO DE LA PLANTA

La estructura del sistema de entrenamiento tiene la función de sostener todo el módulo. Sirve de soporte para los tanques (Tanque 1 y Tanque 2), sensor de distancia y al gabinete donde se encuentran las protecciones eléctricas del sistema. En la parte inferior consta de una plancha de madera la cual soportará al tanque de reserva (Tanque 2), y a la bomba como se observa en la **Figura 5**

Figura 5. Planta de nivel

III. PRINCIPIO DE FUNCIONAMIENTO

La tarjeta de National Instruments NI MyDAQ proporcionará una salida analógica que varía desde 0vdc a 10vdc, la misma señal es la que se inyectará en el variador de frecuencia. Para esto hay que configurar la variante análoga de dicho variador (**Figura 6**).El punto más importante en el proceso y que puede derivar en un error en el proyecto es que hay que puentear las tierras. Esto se debe a que son dos señales totalmente diferentes, por lo tanto, están referenciadas en diferente forma, lo que hace que al puentear las tierras o GND de los diferentes sistemas, estén todas referenciadas al mismo sitio y pueda haber una excelente compatibilidad en el desarrollo del proceso.

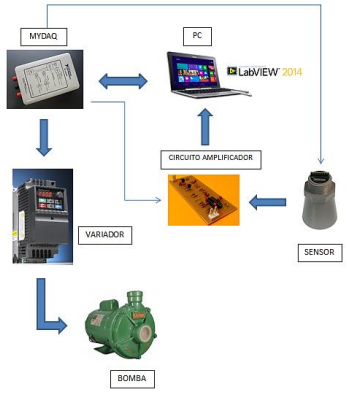

Figura 6. Diagrama Esquemático del Sistema de Control de Nivel

IV. DISEÑO DE LOS CONTROLADORES

Para el diseño del controlador se utilizó el comando fuzzy de Matlab, el que nos despliega la ventana de la figura 6, donde podemos agregar los conjuntos difusos de las variables de entrada y salida así mismo como la regla de control que hemos seleccionado. Para ello debemos seguir los siguientes pasos:

En la tabla 1 se encuentra los valores de entrada y salida para verificar sus respuestas tanto en simulación como en tiempo real.

Tabla 1

Datos de las variables (Entrada y Salida)

ENTRADA: NIVEL(Litros)	SALIDA: DAQ (Voltios)
4	3,163
5	3,369
6	3,31
7	3,568
8	3,643
9	3,7399
10	3,915

Diseño del controlador fuzzy

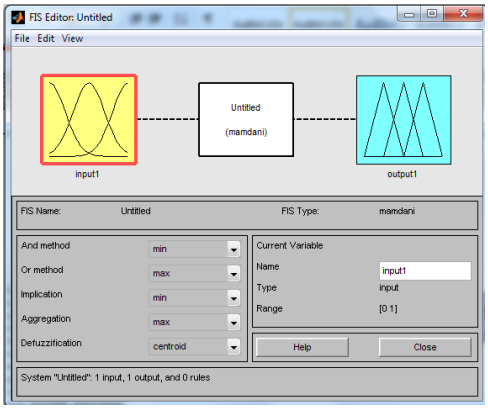

Figura 7. Ventana de comando fuzzy

1. Teclear el comando fuzzy en la ventana principal de Matlab figura 8.

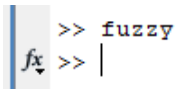

Figura 8. Comando fuzzy

2. Agregamos las entradas y salidas figura 9.

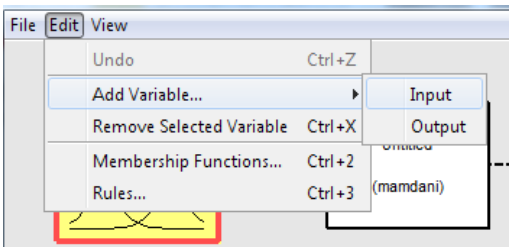

Figura 9. Entradas y Salidas

3. Se procede a cambiar el nombre de cada entrada y salida dando un click sobre cada una y modificando el parámetro que dice Name.

4. Ingresar a una de las variables dando doble click sobre ella y se desplegará la ventada de la **Figura 10.**

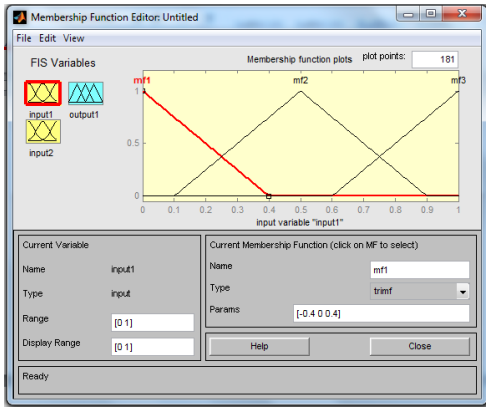

Figura 10. Ventana de Modificación de Variables

5. Se borra el conjunto difuso inicial y se agrega el número de conjuntos propuesto por nuestro diseño. Los pasos para realizar esto se observan en la **Figura 11**.

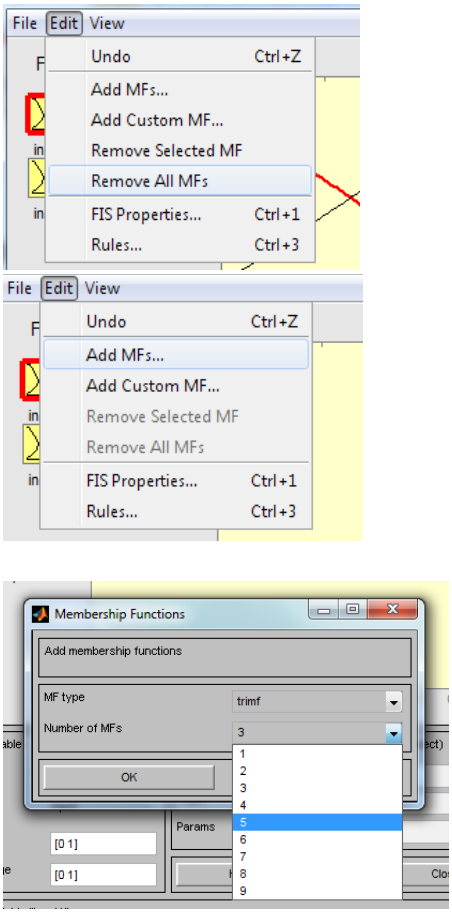

Figura 11. Pasos para generar un conjunto difuso

6. Se cambian los nombres para que coincidan con los que les dimos a un inicio. Para ello se da click sobre el conjunto y se modifica en el campo que dice Name. De igual manera se cambia el rango de acción del grupo de conjuntos en el campo Range. Si todo se realizó correctamente se obtendría lo mismo que se observa en la **Figura 12** y no quedaría más que cerrar la ventana. Para este punto hay que tomar en cuenta los rangos de operación de los equipos, por ejemplo para el rango de la salida de control hay que tomar en cuenta que la bomba trabaja entre el rango de [0.1 a 0.6] e-4, pero al tratarse de valores muy pequeños se utilizará un rango del conjunto fuzzy de [-1 a 1] que posteriormente será compensado mediante un bloque de transformación en la simulación.

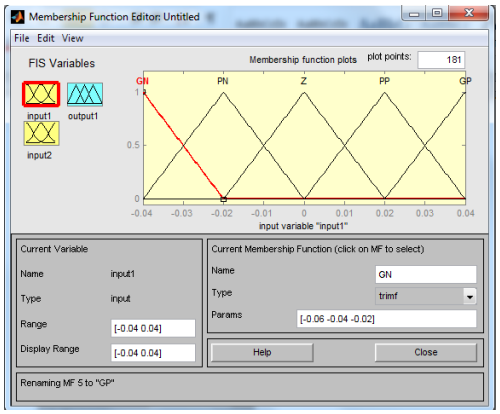

Figura 12. Modificación de los nombres y Rango de los Conjuntos Difusos

7. En la ventana principal de trabajo se da doble click sobre el bloque de color blanco que corresponde a la regla de control que se va a utilizar. El cual nos despliega la ventana que se observa en la **Figura 13**. En ella hay que introducir todas las reglas de control previamente establecidas. Para ello se seleccionar un valor para ERROR, uno PARA CERROR y uno para CONTROL y se da un click en el botón Add rule.

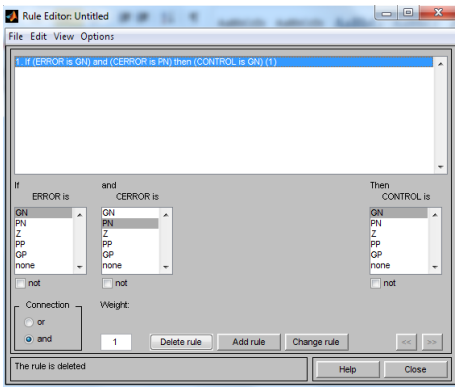

Figura 13. Ventana para agregar las reglas de Control

8. Ahora simplemente hay que exportar el controlador a un archivo, como se muestra en la **Figura 14**, para luego ser cargado al momento de realizar la simulación o al momento de utilizarlo en el control del tanque.

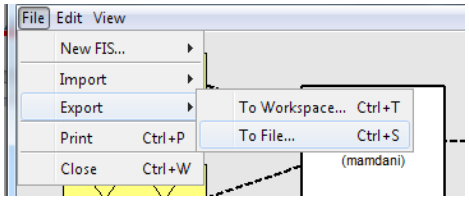

Figura 14. Pasos para exportar el controlador

SIMULACIÓN FUZZY

Antes de realizar la simulación es necesario cargar el archivo generado previamente en el diseño del controlador a una variable con ayuda del comando **readfis** y dicha variable será cargada en el módulo de control fuzzy del Matlab, para ello basta con dar doble click sobre el bloque fuzzy e ingresar el nombre de la variable como se puede ver en la **Figura 15** en este caso tomamos como ejemplo para 5 litros.

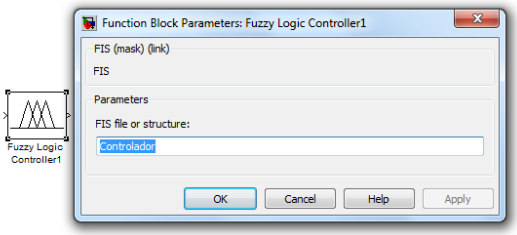

Figura 15. Asignación de Variable a bloque Fuzzy

Para la simulación del controlador se utilizó la herramienta Simulink de Matlab, Ahí se cargó el modelo de la planta y el controlador fuzzy y se generó el lazo de control utilizando al Dicho modelo se puede observar en la **Figura 16.**

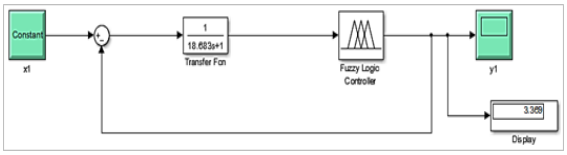

Figura 16. Modelo simulado

Al final en la **Figura 17** observamos que el controlador realiza un correcto control a mas que su respuesta es satisfactoria,

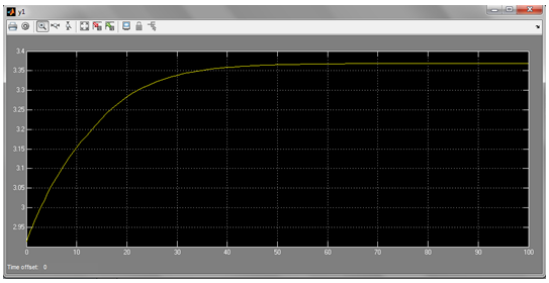

Figura 17. Respuesta del modelo simulado

SIMULACIÓN REDES NEURONALES

Diseño y entrenamiento de la red [5]

Ésta etapa está basada en la selección de una red que posea las características apropiadas para su implementación en el modelo de la planta de proceso virtual, como son: su topología, su tipo de aprendizaje, y la forma en que presenta la salida.

Elaboración del modelo de proceso.

El modelo es desarrollado por medio de Matlab, utilizando una de sus herramientas (Simulink). Es construido a partir de la visualización de los datos obtenidos en la primera etapa e implementando la red neuronal. Para realizar este tipo de programación lo primero que debemos obtener son las variables de entrada y salida, en nuestro caso como variable de entrada es el Nivel (litros) y como salida es el Voltaje de la DAQ en (Voltios) como se puede apreciar en la tabla 1.

Implementación de la Red Neuronal

La red neuronal a implementar es generada desde M-file de Matlab, en donde se usa el código mostrado en la siguiente **Figura 18.**

```
x=[0 1 2 3 4 5 6 7 8 9 10]; % Valores del nivel en Litros
y=[0 3.055 3.19 3.31 3.163 3.369 3.31 3.568 3.643 3.7399 3.915]; % Valores de la DAQ en Voltios .
plot(x,y) % Comando de Matlab que genera la figura
% 3.2 con los valores de x & y
% Instrucciones para la generación y entrenamiento de la Red Neuronal
p=[0 1 2 3 4 5 6 7 8 9 10]; % Vector entrada, el cual esta constituido
% por los valores del nivel.
t=[0 3.055 3.19 3.31 3.163 3.369 3.31 3.568 3.643 3.7399 3.915]; % Vector de salida, el cual contiene los
% valores de la DAQ en voltios de salida.
% El comando newff genera e indica las capas y el tipo de entrenamiento
% con el cual se forma la Red Neuronal. Donde:
% Los valores que toma la Red Neuronal son los valores minimos y máximos
% (0-10 mm) de la matriz p, mientras que la función de transferencia
% en la capa oculta es la tan-sigmoidal (tansig) y en la capa de salida
% la función de transferencia es lineal (purelin), por ultimo el algoritmo
% de entrenamiento es Levenberg-Marquardt (trainlm).
net = newff([0 10],[10 1],{'tansig' 'purelin'},'trainlm');

y1 = sim(net,p); % Este comando guarda todos los datos generados, para
% el uso futuro de ellos en Simulink.
net.trainParam.epochs = 200; % Indica el numero de capas con la que se
% genera la red.
net.trainParam.goal = 0.001; % Indica el valor de corrección de error
% que tendra la red.
net = train(net,p,t); % El comando train, entrena a la Red Neuronal
% con los datos proporcionados anteriormente.
y2 = sim(net,p); % Guarda los nuevos valores de la Red Neuronal después
% del entrenamiento en Simulink.
plot(p,t,p,y2) % Genera la grafica de la Figura 3.5
gensim(net,1) % Comando que genera la Red Neuronal en Simulink.
```

Figura 18. Código de Entrenamiento

Entrenamiento de la red

Una vez generada la red, comienza a entregar valores aleatorios a los pesos, para lograr que la red responda a los objetivos planteados es necesario entrenarla, en la figura 17 se observa una ventana donde se van a generar las gráficas. El entrenamiento genera una gráfica, mostrada en la Figura 19, la cual es la comparación de la respuesta experimental con la simulada.

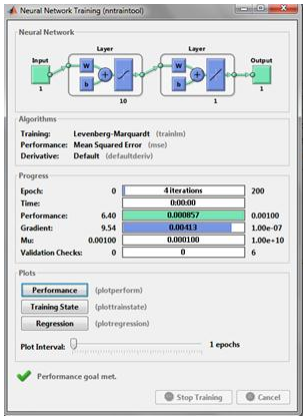

Figura 19. Neuronal Network Training

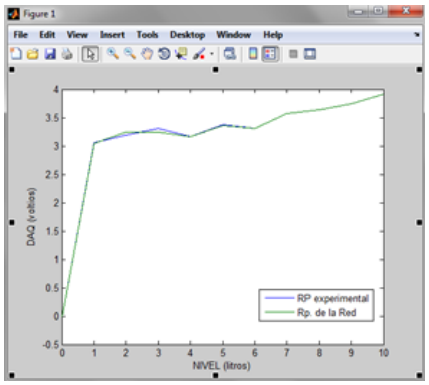

Figura 20. Comparación de curvas experimental y simulada

Generación de la red neuronal en Simulink

Una de las ventajas que ofrece Matlab es generar la red obtenida en Simulink esto es mediante el código de **la Figura 18.**

La red neuronal formada por dicho código y con el modelo de la planta es mostrada en la **Figura 21**, y la respuesta para 5 litros en la **Figura 22.**

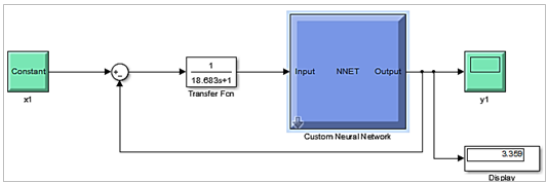

Figura 21. Diagrama del proceso de la planta de nivel.

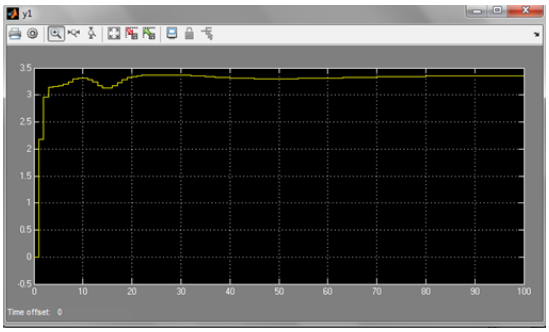

Figura 22. Respuesta para 5 litros.

PARTE EXPERIMENTAL FUZZY Y REDES NEURONALES UTILIZANDO LABVIEW

CONTROL FUZZY

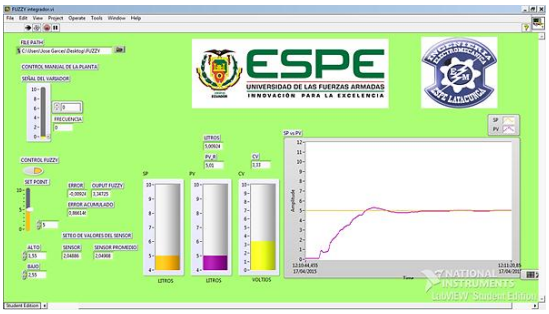

Figura 23. Respuesta para 5 litros control fuzzy.

CONTROL NEURONAL

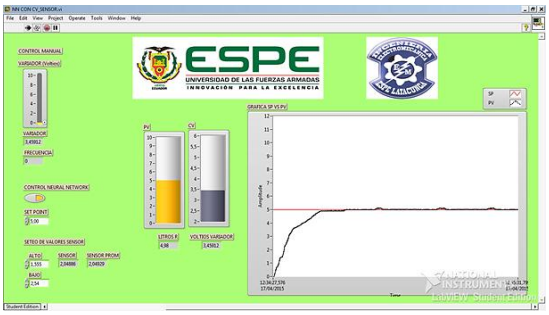

Figura 24. Respuesta para 5 litros control neuronal.

ANÁLISIS DE RESULTADOS

Análisis de resultados simulación vs parte experimental

Las respuestas de los controladores inteligentes (fuzzy y Neuronal) para el proceso de la planta de nivel simulada mediante la plataforma de MATLAB, son satisfactorias dado que las respuestas de la misma son cercanas en comparación con las respuestas obtenidas experimentalmente con la planta física, otro punto favorable de la planta simulada, es que tanto para el control fuzzy y neuronal se puede modificar la arquitectura de la programación para conseguir la respuesta deseada, ya que nos ayudaría de una manera lógica para implementar en la parte experimental. Este último punto es bueno ya que en la realidad a veces no se logra mantener estable la señal del sensor, debido al excesivo ruido que hay en el exterior y por ende no obtendríamos un

control satisfactorio. Mientras en la parte experimental se obtuvo un óptimo desempeño de los dos controladores a través de la plataforma de LABVIEW.

En las figuras 15 (MATLAB) y 21 (LABVIEW) para un set point de 5 litros se justifica que ambos controles presentan un desempeño satisfactorio, es decir que sus respuestas son cercanas con las de la tabla 1 Por otro lado en sus tiempos de estabilización ambos controles se estabilizan en un tiempo t>21 segundos. Mientras que las curvas de respuestas de las figuras antes mencionadas se observa que hay un pequeño sobre-impulso al iniciar su trayectoria, esto implica el tiempo de reacción de la variable de proceso (PV) cuando la bomba arranca, pero el mismo puede mejorar entrenándose con más puntos de referencia, tomados al arrancar la bomba, provocando una reacción más óptima de la variable de proceso, esto obviamente implica más tiempo de entrenamiento y la realización de pruebas de funcionamiento de la red.

CONCLUSIONES Y RECOMENDACIONES

a) Conclusiones:

- Para el desarrollo exitoso del proyecto, fue primordial recopilar y analizar fundamentos teóricos relacionados al control inteligente, con la finalidad de adquirir una perspectiva clara y objetiva de los temas, dándole un enfoque correcto al diseño y construcción de un controlador inteligente.
- Con la información recopilada de fuentes bibliográficas y digitales se verificó que los controles empleados para este proyecto no requieren de un modelo matemático como su principal característica para la construcción del mismo, por lo que se emplea un lenguaje heurístico, que no es más que la utilización de un lenguaje propio del ser humano, para la toma de decisiones, en el comportamiento y desempeño del control de la planta, es importante notar que este lenguaje debe ser de fácil comprensión y entendimiento para que el programador pueda realizar un buen diseño del controlador.
- Como se ha podido apreciar durante el desarrollo del presente trabajo, se implementaron dos controles inteligentes (fuzzy y neuronal), los mismo que se realizaron en dos etapas, la primera consiste en simular el proceso mediante la plataforma de MATLAB y la otra parte se realizó de manera práctica en LABVIEW, en ambos casos sus respuestas son satisfactorias, ya que mostraron cercanía con las respuestas esperadas tomadas experimentalmente en la planta física.
- Al momento de probar el diseño del controlador difuso en la planta real, ésta se encontró inestable, a tal inconveniente, hubo la necesidad de implementar un control integrador al diseño, que ayude a eliminar el error entre la variable de proceso y el Set-Point para que el sistema se estabilice.
- En base a la parte experimental, el control Difuso presentó mejor respuesta de la variable de proceso al accionar el actuador, mientras que el control neuronal se mostró lento al arranque de la bomba.
- En la parte real, el control neuronal denotó mayor rapidez que el Difuso, estabilizando su variable de proceso con mayor eficiencia; esto quiere decir, que los puntos designados al entrenamiento de la red fueron los adecuados.
- Si bien el control neuronal presentó mejor tiempo de estabilización que el Difuso, este último brinda menor error entre la variable de proceso y el Set-Point cuando las dos señales se encuentran en estado estable, esto se produce, gracias a la acción integral que tiende a cero a medida que el tiempo transcurre.
- La planta presentó inconvenientes al tomar la señal de voltaje analógico del sensor, a causa de pérdidas de potencial en el conexionado, por lo tanto aquella señal no llegaba a la interface; con aquel problema, se montó un acondicionamiento de señal, donde primero elevó el voltaje original y luego transformó dicha tensión a corriente, evitando pérdidas en el cable.

b) Recomendaciones:

- Las soluciones que se presentó en el control simulado dentro del proceso, son favorables, ya que en la parte práctica a veces se perdía la

señal y por ende no existía un control. Por esta razón fue que el control simulado responde de mejor manera a las respuestas esperadas.

- Se recomienda mejorar el diseño de los conjuntos difusos y utilizar un control integrador para eliminar el error en estado estable de la planta y evitar su inestabilidad.
- Se puede mejorar el desempeño de la planta en el arranque del actuador, dotando a la red Neuronal de más puntos de referencia para su entrenamiento y así pueda responder de inmediato cuando la bomba de marcha.
- En base a la experiencia ganada, se podría mejorar el tiempo de estabilización de la planta, al editar el conjunto difuso de salida o la vez su rango de acción, así como también se podría jugar con el integrador y conseguir mejores resultados.
- Se puede minimizar el error de estabilización en la red Neuronal con respecto al tiempo, entrenando a la misma con mayor cantidad de datos.
- Al trabajar con un sensor ultrasónico se dan varios tips tales como: utilizar un acondicionamiento de señal o trabajar con señales digitales incluso señales con ancho de pulsos para evitar pérdidas de potencial en el cable y garantizar una buena señal de control.
- Se recomienda para la toma de medidas trabajar con un solo tipo de líquido para evitar errores de medida y la constante calibración del sensor ultrasónico.

Bibliografía

[1] P. Ponce Cruz, Inteligencia Artificial con Aplicaciones a la Ingeniería.

[2] R. Ilera, Redes Neuronales y Lógica difusa teroría y aplicaciones, Alfaomega, 1998.

[3] I. A. Olier y G. Guerrero, Redes neuronales artificiales - Fundamentos modelos y aplicaciones.

[4] Wintermute, Redes de neuronas artificiales y pensamiento.

[5] *MatLab H. Desing*, NARMA-L2, 2013.

Biografía

Juan Pablo León Calderón, nació en Ambato -Ecuador, el 20 de Abril de 1988; cursó sus estudios secundarios en el Colegio Nacional Jorge Álvarez, donde obtuvo el título de Bachiller en Ciencias, especialización Físico Matemático. Sus estudios superiores los realizó en la Universidad de las Fuerzas Armadas ESPE Extensión Latacunga, en donde obtuvo el Título de Ingeniero en Electromecánica en el 2015.

José Carlos Garcés Pico, nació en Ambato -Ecuador, el 10 de Abril de 1987; cursó sus estudios secundarios en el Instituto técnico Superior "Bolívar", donde obtuvo el título de Bachiller en Ciencias, especialización Físico Matemático. Sus estudios superiores los realizó en la Universidad de las Fuerzas Armadas ESPE Extensión Latacunga, en donde obtuvo el Título de Ingeniero en Electromecánica en el 2015.

Milton Fabricio Pérez Gutiérrez, nació en Ambato - Ecuador, el 19 de Diciembre de 1972; cursó sus estudios secundarios en el Colegio "San Alfonso", donde obtuvo el título de Bachiller en Ciencias, especialización Físico Matemático. Sus estudios superiores los realizó en la Escuela Politécnica del Ejército Sede Latacunga, en donde obtuvo el Título de Ingeniero en Electrónica e Instrumentación y Máster en Control de Procesos Industriales en la Universidad de Córdoba (España).

Ing. Freddy William Salazar Paredes, nació en Latacunga - Ecuador, el 21 de Noviembre del 1981; cursó sus estudios secundarios en el Colegio "Vicente León", donde obtuvo el título de Bachiller

en Ciencias, especialización Físico Matemático. Sus estudios superiores los realizó en la Escuela Politécnica del Ejército Sede Latacunga, en donde obtuvo el Título de Ingeniero en Electromecánico y Máster en Gestión de Energía en la Universidad Técnica de Cotopaxi.