Control difuso, neuro artificial y neurodifuso de un reactor químico usando el método Mamdani, Sugeno y ANFIS directo e inverso

Alejandro Naranjo; José Londoño
alejandro.naranjo_z@uao.edu.co
jose.londono_san@uao.edu.co
Ingeniería mecatrónica
Facultad de Ingeniería
Departamento de Electrónica y Automática.
Universidad Autónoma de Occidente
Cali, Colombia

Abstract* — *The following document shows the procedure for designing controllers using neural fuzzy, fuzzy and neural methods. In particular the Mamdani, Sugeno, and ANFIS methods in direct and inverse variation. Finally, the Mamdani method will be implemented in arduino using the corresponding libraries. Communication with arduino is completed.

Keywords – ANFIS, Mamdami, Sugeno, Bias, Synaptic Weights, Arduino.

I. INTRODUCCIÓN

Un reactor químico es un sistema que funciona a base de reacciones químicas y toda la cinetica que ocurre alrededor de ellas. El objetivo principal es el de maximizar ciertos procesos de conversión y selectividad al menor costo posible, haciendo uso de la optimización de estas reacciones antes mencionadas.

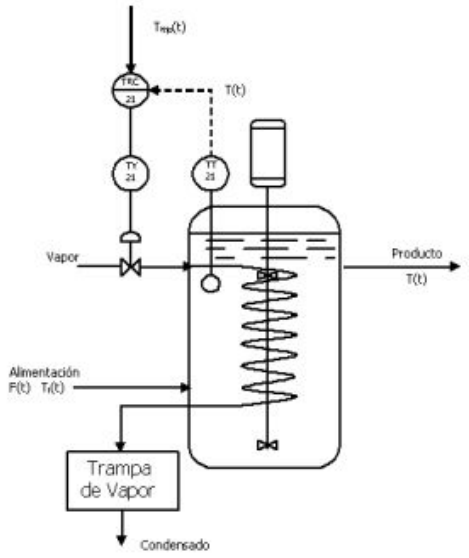

Fig 1. Esquema del biorreactor.

El biorreactor del caso de estudio en particular es un sistema de 3 estados no lineal compuesto por la sinergia entre las variables de interés: la concentración de sustrato (s), la concentración de producto (y) y la concentración de biomasa (x). La dinámica del sistema se describe gracias a las siguientes ecuaciones:

$$S' = \left(S^{(0)} - S\right)D - \frac{X}{\gamma_1}\frac{m_1 S}{a_1 + S} - \frac{Y}{\gamma_2}\frac{m_2 S}{a_2 + S} \quad (1)$$

$$X' = X\left(\frac{m_1 S}{a_1 + s} - D\right) \quad (2)$$

$$Y' = Y\left(\frac{m_2 S}{a_2 + S} - D\right) \quad (3)$$

Fig 2. Ecuaciones que describen la dinámica.

Cabe resaltar que las constantes son a1=0.5 a2=3.5 m1=5.0 m2=6.0 S(0)=0.3 r1=r2=4.0 y que la salida tiene un valor inicial de 0.1, en caso contrario el valor de la salida nunca cambiará de 0 por su comportamiento recurrente entre sí.

II. ANÁLISIS Y RESULTADOS

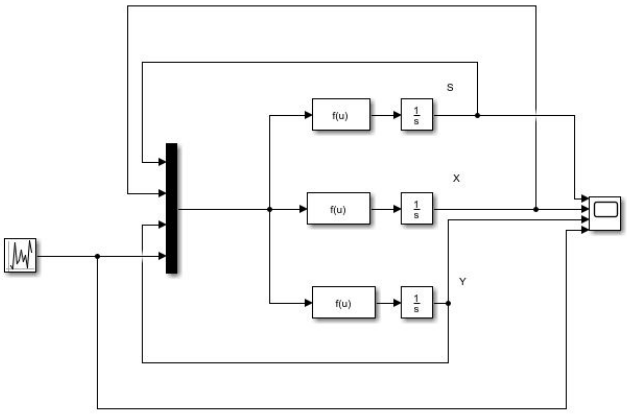

Fig 3. Diagrama de bloques del sistema.

En el diagrama de bloques de la figura 3 se montó el sistema del bioreactor usando los bloques de function, usando una entrada de 0.1 a 1.7 como se indicaba en la guía del sistema. Logrando una salida de 0.2 a 1.2 como se muestra en la siguiente figura. Adicionalmente, se encontró un tiempo de muestreo de 0.6 segundos.

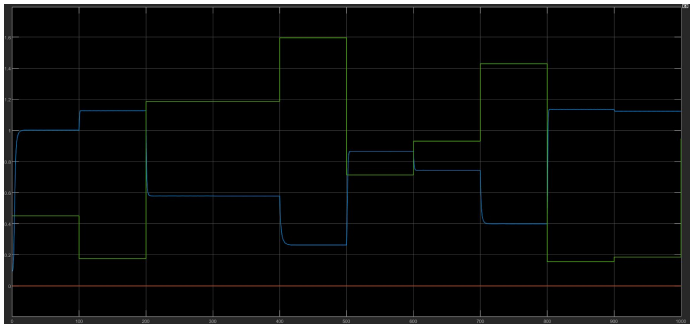

Fig 4. Respuesta del sistema.

Mandani

Seguidamente, se procedió a realizar el controlador difuso mamdani decidiendo los rangos de las entradas y la salida. Como entradas se escogieron el error y la derivada del error como se especifica en el enunciado del ejercicio. El primero en un rango de -1 a 1 debido a que el máximo error se da cuando la entrada es 1.2 y la salida es 0.2, es decir 1, mientras que el mínimo error se da cuando la entrada es 0.2 y la salida 1.2, es decir -1. Mientras que la derivada del error se tomó normalizada en el rango de -1 a 1. Seguidamente, y debido a que la entrada del sistema en la tasa de disolución, un cambio, también se tomó normalizada de -1 a 1. Los conjuntos de los extremos se tomaron rectangulares y no triangulares.

```
% Creacion de la variable que almacena el sistema difuso
fis1 = newfis('BioreactorMandani');%MANDANI

%% Definicion y creacion de los conjuntos difusos del error
fis1 = addvar(fis1,'input','error',[-1 1]);
fis1 = addmf(fis1,'input',1,'muy_neg','trapmf',[-1 -1 -2/3 -1/3]);
fis1 = addmf(fis1,'input',1,'neg','trimf',[-2/3 -1/3 0]);
fis1 = addmf(fis1,'input',1,'cero','trimf',[-1/3 0 1/3]);
fis1 = addmf(fis1,'input',1,'pos','trimf',[0 1/3 2/3]);
fis1 = addmf(fis1,'input',1,'muy_pos','trapmf',[1/3 2/3 1 1]);

%% Definicion y creacion de los conjuntos difusos de  la derivada del error
fis1 = addvar(fis1,'input','derivada_error',[-1 1]);
fis1 = addmf(fis1,'input',2,'muy_neg','trapmf',[-1 -1 -2/3 -1/3]);
fis1 = addmf(fis1,'input',2,'neg','trimf',[-2/3 -1/3 0]);
fis1 = addmf(fis1,'input',2,'cero','trimf',[-1/3 0 1/3]);
fis1 = addmf(fis1,'input',2,'pos','trimf',[0 1/3 2/3]);
fis1 = addmf(fis1,'input',2,'muy_pos','trapmf',[1/3 2/3 1 1])
```

Fig 5. Código en Matlab de la configuración de las entradas del controlador difuso Mamdani.

```
%% MANDANI
fis1 = addvar(fis1,'output','accion',[-1.7 1.7]);
fis1 = addmf(fis1,'output',1,'muy_neg','trapmf',[-1 -1 -2/3 -1/3]);
fis1 = addmf(fis1,'output',1,'neg','trimf',[-2/3 -1/3 0]);
fis1 = addmf(fis1,'output',1,'cero','trimf',[-1/3 0 1/3]);
fis1 = addmf(fis1,'output',1,'pos','trimf',[0 1/3 2/3]);
fis1 = addmf(fis1,'output',1,'muy_pos','trapmf',[1/3 2/3 1 1]);
```

Fig 6. Código de la configuración de la salida del sistema del controlador difuso Mamdani.

En la figura 6 se muestra la conformación de los rangos de la salida donde los conjuntos extremos se toman también rectangulares y no triangulares. Luego, se planearon las reglas a seguir basándonos en el comportamiento del error y del cambio de este, y así aplicar una acción correctiva según dependiera el caso. Las reglas se ven la siguiente tabla:

dE/E	1	2	3	4	5
1	1	1	1	2	3
2	1	2	2	2	2
3	4	3	3	3	2
4	4	4	4	4	5
5	3	4	5	5	5

Tabla 1. Reglas difusas.

Donde, para las entradas: 1, se interpreta como muy negativo; 2, como negativo; 3, como cero; 4, como positivo; y 5, como muy positivo. Mientras que para la salida: 5, significa el aumento rápido de la tasa de cambio; 4, el aumento lento; 3, ningún aumento; 2, la disminución lenta; y 1 la disminución rápida. Finalmente, se utiliza el comando Addrule para introducir las reglas al modelo, el comando Fuzzy para generar el sistema difuso y el Readfis para leer el modelo creado.

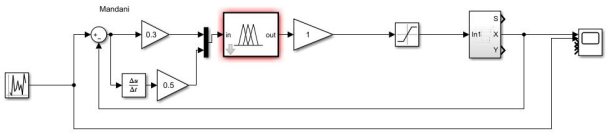

Fig. 7. Diagrama de bloques del sistema difuso.

Se prosiguió a montar el sistema de control difuso con constantes para el error y la derivada de este de 0.3 y 0.5 respectivamente, se cargó el modelo difuso y se conectó con un saturador de -1 a 1 a la planta, ofreciendo la respuesta que se ve en la figura 8, donde se ve un control adecuado del reactor, con ciertos sobre impulsos:

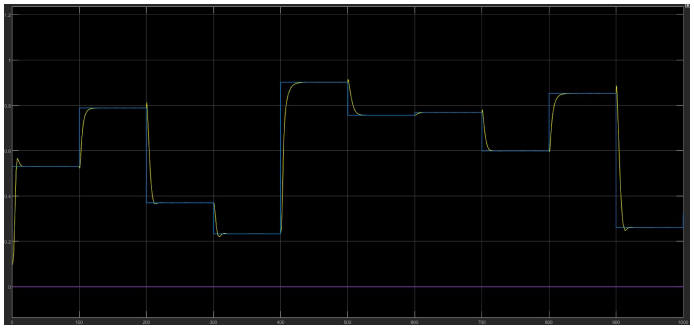

Fig 8. Respuesta del controlador Mamdani.

Sugeno

Para el controlador tipo sugeno, se debe incluir la palabra sugeno entre comillas simples durante la llamada del comando newfis de modo que los procedimiento internos del algoritmo de hagan siguiendo el modelo Sugeno y no el Mamdani que está por defecto. La otra diferencia es la forma en como se ponen los rangos de la salida, poniendo solo los valores límites entre conjuntos obviando los otros números que si colocan en el modelo anterior. Los rangos que se usaron fueron los mismos anteriores.

```
%% construccion de un FIS por comandos
% Creacion de la variable que almacena el sistema difuso
fis1 = newfis('BioreactorSugeno','sugeno');%SUGENO


%% SUGENO
fis1 = addvar(fis1,'output','accion',[-1 1]);
fis1 = addmf(fis1,'output',1,'accion_muy_pequeña','constant',[-2/3]);%1
fis1 = addmf(fis1,'output',1,'accion_pequeña','constant',[-1/3]);%2
fis1 = addmf(fis1,'output',1,'accion_constante','constant',[0]);%3
fis1 = addmf(fis1,'output',1,'accion_grande','constant',[1/3]);%4
fis1 = addmf(fis1,'output',1,'accion_muy_grande','constant',[2/3]);%5
```

Fig 9. Código Sugeno en Matlab.

El diagrama del sistema es similar también, con la excepción de que las constantes para el error y la derivada del error cambian a 0.45 y 0.5 respectivamente. Además de que la constante de la entrada de control cambia de 1 a 1.1.

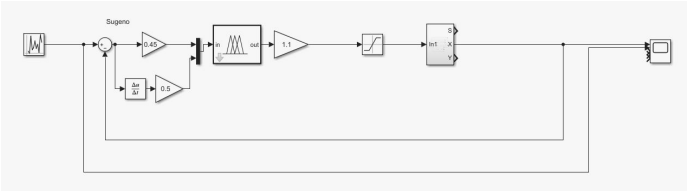

Fig 10. Diagrama de bloques del controlador Sugeno.

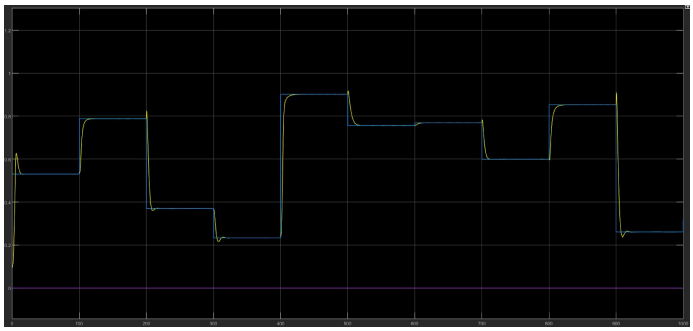

Fig 10. Respuesta del controlador Sugeno.

Como se puede ver en la figura 10, la respuesta obtenida por ambos controladores difusos son muy similares. Con la pequeña diferencia de que existen sobre impulsos más pronunciados en el sugeno.

Modelo directo

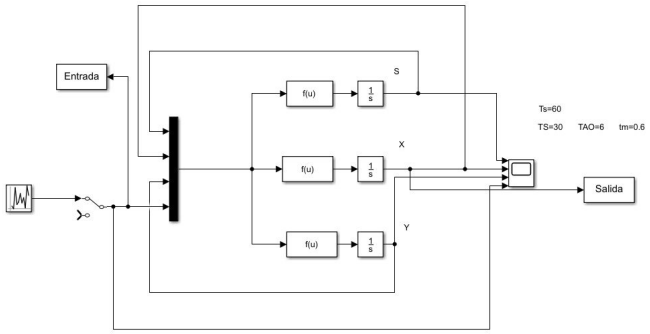

Fig 11. Toma de datos del sistema.

Luego, se prosiguió con la extracción de los datos de entrenamiento de la planta para poder entrenar la red neuronal que simulará la dinámica de la planta. Como se puede ver se encontró un tiempo de muestreo de 0.6 segundos como se dijo anteriormente, y el ancho del escalón idealmente sería de 60 segundos.

```
Tm=0.2
U=Entrada;
Y=Salida;
N=length(U);


X=[Y(3:N-1),Y(2:N-2),Y(1:N-3),U(1:N-3),U(2:N-2),U(3:N-1)]';
Yd=[Y(4:N)]';

Red=newff(X,Yd,[10],{'tansig','purelin'},'trainlm');
Red.trainParam.epochs=500;
Red.dividefcn='';
Red=train(Red,X,Yd);
gensim(Red,Tm)
```

Fig 12. Código del modelo directo.

Sin embargo, el tiempo de muestreo se bajó a 0.2 por problemas de oscilaciones en las zonas de estabilización. Se usó una red de una capa de 10 neuronas, con función de activación tansig, y un método de entrenamiento trainlm.

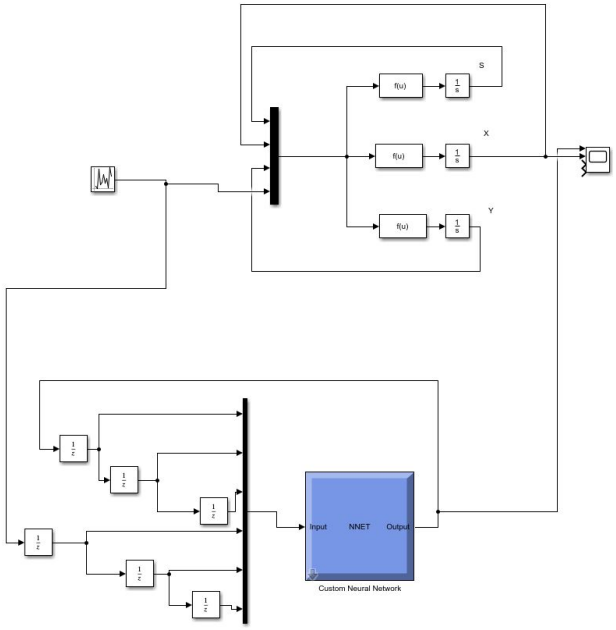

Fig 13. Comparación de la red con la planta real.

Se entrenó el modelo tomando 3 instantes de la salida y de la entrada, y de esta forma realizar una buena aproximación para al tercer orden de la planta. Los resultados se muestran a continuación, donde se puede ver una dinámica correcta, demostrando que aprendió correctamente, sin embargo, presenta problemas paras seguir a la planta en el instante inmediatamente indicial.

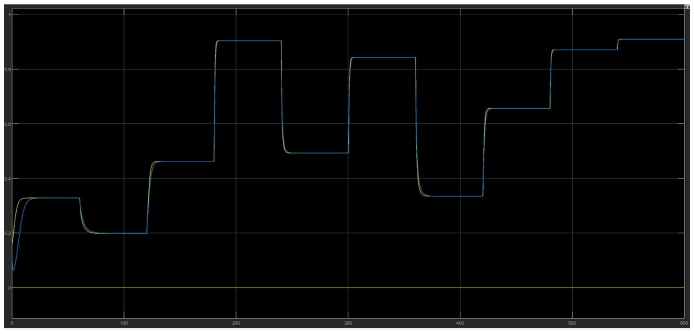

Fig 14. Respuesta del modelo directo.

ANFIS directo

```
Tm=0.1
U=Entrada;
Y=Salida;
N=length(U);


X=[Y(3:N-1),Y(2:N-2),Y(1:N-3),U(1:N-3),U(2:N-2),U(3:N-1)];
Yd=[Y(4:N)];
DatosPlanta=[X Yd];
anfisedit

%fis = readfis('AnfisDirecto.fis')
```

Fig 15. Código usado para el cálculo del ANFIS.

Como se puede ver, se usó un tiempo de muestreo diferente al anterior, esto se debe a que con la respuesta presentaba oscilaciones en las zonas de estabilización. Seguidamente, se usaron los mismos instantes para la salida y la entrada que en el modelo directo y con el comando de anfis edit se abre el configurador que se ve a continuación:

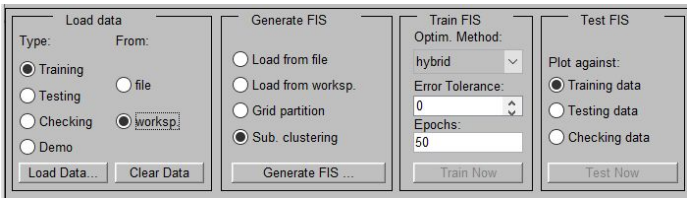

Fig 16. Configuración para el cálculo del ANFIS.

El cuadro de la figura 16 muestra que los datos cargados se pueden tomar como de entrenamiento o como de validación; Luego, el uso de la técnica para generar el FIS, se escogió subcluster debido a que es más preciso con la respuesta. Finalmente, se coloca el método de entrenamiento para las capas entrenables y el número de épocas, se escogió híbrido debido a que no aprendía la dinámica correctamente con el otro método.

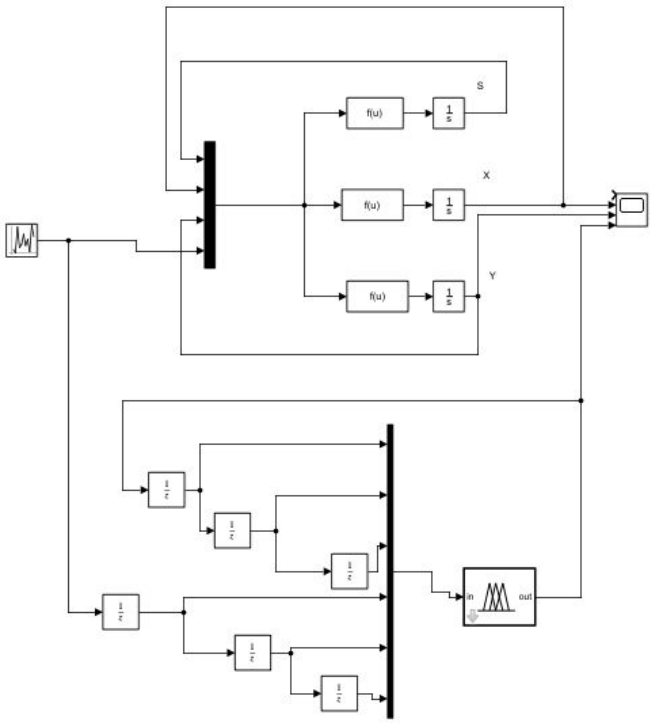

Fig 17. Diagrama de bloques del ANFIS directo.

Se comparó, al igual que con el método neuronal anterior, con la planta original para ver su parecido. La figura 18 ilustra que no se logra una similitud igual a de su contraparte calculada únicamente por inteligencia artificial, presenta cierta error de estado estacionario y existen sobre impulsos que con el otro método no aparecían. Finalmente, también tiene problemas para seguir a la planta en los instantes iniciales.

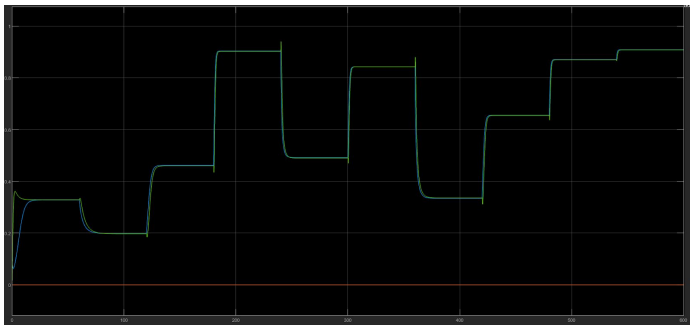

Fig 18. Respuesta del modelo ANFIS directo..

Modelo inverso

El modelo inverso si pudo llevarse a cabo con el tiempo de muestreo de 0.6, y cabe resaltar que el ancho de los escalones se dejó en 100 segundos con el objetivo de evitar problemas de estabilización. Se tomaron 5 instantes de la salida y 3 de la entrada para simular el comportamiento inverso de la planta. Se usó una capa de 15 neuronas con función de activación tansig y método de entrenamiento trainlm.

```
Tm=0.6
U=Entrada;
Y=Salida;
N=length(U);


X=[Y(5:N),Y(4:N-1),Y(3:N-2),Y(2:N-3),U(1:N-4),U(2:N-3),U(3:N-2)]';
Yd=[U(4:N-1)]';

Red=newff(X,Yd,[15],{'tansig','purelin'},'trainbr');
Red.trainParam.epochs=500;
Red.dividefcn='';
Red=train(Red,X,Yd);
gensim(Red,Tm)
```

Fig 19. Código del modelo inverso.

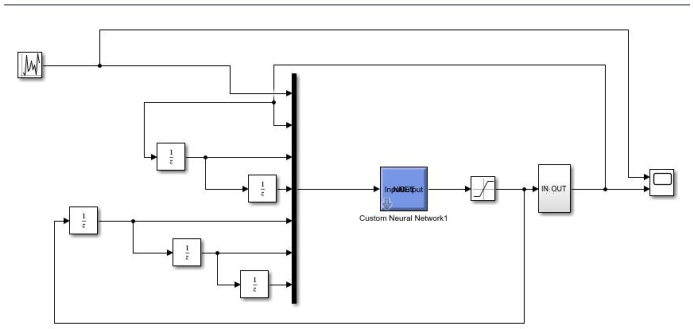

Fig 20. Diagrama de bloques del modelo inverso.

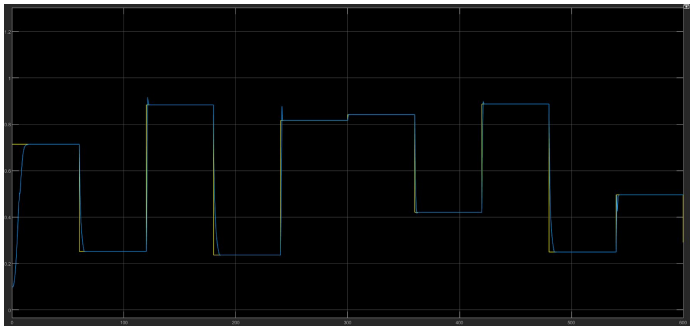

Fig 21. Respuesta del modelo inverso.

La figura 21 muestra el control inverso por modelo únicamente neuro artificiales, donde se ve un seguimiento de la entrada con un leve problema de estabilización en el primer escalón, además de algunos sobre impulsos.

ANFIS inverso

El Anfis Inverso contiene las mismas características que su contraparte en el sentido de entradas a la red para el manejo de datos de entrenamiento. Pero se calcula con 0.6 segundos de entrenamiento a diferencia del anterior. Adicionalmente, su configuración usando anfis edit es igual que su contraparte directa.

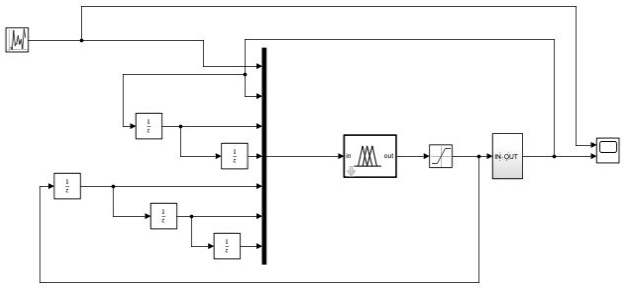

Fig 22. Diagrama de bloques del ANFIS inverso.

La respuesta del modelo se muestra en la figura 23, donde se puede apreciar el mismo problema de estabilización para el primer escalón e igual que con al ANFIS directo con el modelo directo, este presenta más tendencia a sobre impulsos que el modelo inverso neuronal.

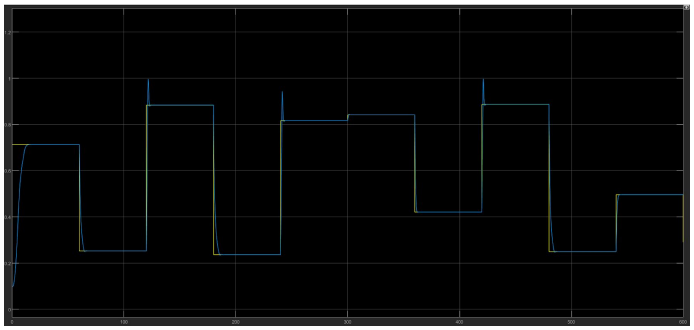

Fig 23. Respuesta del controlador por ANFIS inverso.

Arduino

Después, se prosiguió a implementar el controlador Mamdani en arduino, conectándolo con Matlab, donde estaría la planta simulada.

```
  // Se inicializa la comunicación serial
  Serial.begin(9600);
  randomSeed(analogRead(0));
}

void loop() {
// Se crea la variable tipo fuszzy_system
  FUZ_SYS fuzzy_system;

  if (Serial.available()>0){
   Y_p=Serial.readStringUntil('\n').toFloat();
  }

// Contador de muestras para saber cuando se va a cambiar la entrada al siste
  Samples=Samples+1;


// Si el contador de muestras es 1 (uno) se modifica la entrada al sistema ger
if (Samples==1)
{
  //Se genera un número randómico entre 0 y 400
  randNumber = random(10);
}

// Si el contador de muestras supera un valor se resetea el contador para que
if (Samples>250)
{
  Samples=0;
}

// Se escala el numero randómico para que quede en el rango entre 0.1 y 0.5
Ref=(randNumber/10)+0.2;

R_K=Ref;
```

Fig 24. Código arduino del controlador Fuzzy.

Utilizando una comunicación serial de 9600 baudios se utilizaron escalones de 250 muestras, un número mayor a las 166 necesarias teniendo en cuenta el tiempo de muestreo de 0.6 segundos y el tamaño del escalón de 100 segundos. Dichos escalones van de 0.2 a 1.2.

```
fuzzy_init(&fuzzy_system);

U_Fuzzy = fuzzy_control(0.45*E,0.5*Edot,&fuzzy_system);
U_K=1.1*U_Fuzzy ;

fuzzy_free(&fuzzy_system);

//Se manda vía serial la salida del sistema y la entrada
Serial.print( Y_p,7);
Serial.print(";");
Serial.print( R_K,7);
Serial.print(";");
Serial.println( U_K,7);
//Retardo o tiempo en el que el ciclo se vuelve a repetir
delay(100);
}
```

Fig 25. Cálculo del controlador Fuzzy.

En la figura 25 se puede observar como haciendo uso de la librería de Arduino para sistemas difusos se calcula el controlador usando las constantes antes mencionadas durante la fase de implementación en Matlab. Finalmente, se imprimen los datos necesarios para comunicarse con Simulink.

```
fuzzy_system->emem->width[0] =0.666*Merror;
fuzzy_system->emem->width[1] =0.666*Merror;
fuzzy_system->emem->width[2] =0.666*Merror;
fuzzy_system->emem->width[3] =0.666*Merror;
fuzzy_system->emem->width[4] =0.666*Merror;

fuzzy_system->emem->center[0] =-0.666*Merror;
fuzzy_system->emem->center[1] =-0.333*Merror;
fuzzy_system->emem->center[2] =0.0*Merror;
fuzzy_system->emem->center[3] =0.333*Merror;
fuzzy_system->emem->center[4] =0.666*Merror;


fuzzy_system->edotmem->width[0] = 0.666*Mder;
fuzzy_system->edotmem->width[1] = 0.666*Mder;
fuzzy_system->edotmem->width[2] = 0.666*Mder;
fuzzy_system->edotmem->width[3] = 0.666*Mder;
fuzzy_system->edotmem->width[4] = 0.666*Mder;

fuzzy_system->edotmem->center[0] = -0.666*Mder;
fuzzy_system->edotmem->center[1] = -0.333*Mder;
fuzzy_system->edotmem->center[2] = 0.0*Mder;
fuzzy_system->edotmem->center[3] = 0.333*Mder;
fuzzy_system->edotmem->center[4] = 0.666*Mder;


fuzzy_system->outmem->center[0] = -0.666*Mu;
fuzzy_system->outmem->center[1] = -0.333*Mu;
fuzzy_system->outmem->center[2] = 0.0*Mu;
fuzzy_system->outmem->center[3] = 0.333*Mu;
fuzzy_system->outmem->center[4] = 0.666*Mu;
```

Fig 26. Conjuntos difusos.

Debido a que todos los rangos de los conjuntos van de -1 a 1 las constantes que acompañan los números son iguales a 1. Mientras que estas limitaciones son entonces los valores puros donde acaba y comienza otro conjunto. Además, se transcribieron las reglas necesarias para el funcionamiento del controlador.

```
fuzzy_system->Rules[0][0]=0;
fuzzy_system->Rules[0][1]=0;
fuzzy_system->Rules[0][2]=0;
fuzzy_system->Rules[0][3]=1;
fuzzy_system->Rules[0][4]=2;

fuzzy_system->Rules[1][0]=0;
fuzzy_system->Rules[1][1]=1;
fuzzy_system->Rules[1][2]=1;
fuzzy_system->Rules[1][3]=1;
fuzzy_system->Rules[1][4]=1;

fuzzy_system->Rules[2][0]=3;
fuzzy_system->Rules[2][1]=2;
fuzzy_system->Rules[2][2]=2;
fuzzy_system->Rules[2][3]=2;
fuzzy_system->Rules[2][4]=1;

fuzzy_system->Rules[3][0]=3;
fuzzy_system->Rules[3][1]=3;
fuzzy_system->Rules[3][2]=3;
fuzzy_system->Rules[3][3]=3;
fuzzy_system->Rules[3][4]=4;

fuzzy_system->Rules[4][0]=2;
fuzzy_system->Rules[4][1]=3;
fuzzy_system->Rules[4][2]=4;
fuzzy_system->Rules[4][3]=4;
fuzzy_system->Rules[4][4]=4;
```

Fig 27. Reglas del controlador difuso.

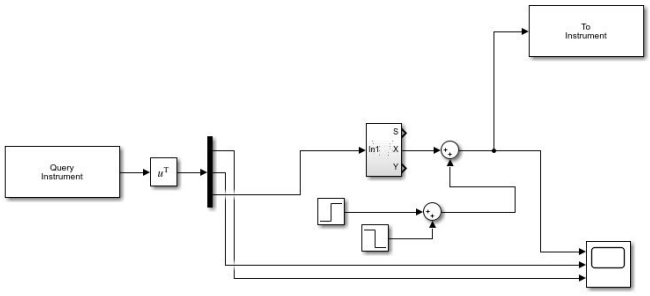

Fig 28. Respuesta del controlador por ANFIS inverso.

Se usaron los bloques query y to instrument con las configuraciones necesarias de comunicación de datos ya especificadas en el código de arduino, como los baudios y el orden de datos. Se introdujo una perturbación en el instante 50, para ser retirada en el instante 100, los escalones se muestran de 150 para poder apreciar este efecto. Como se ve en la figura 29, el sistema es a prueba de perturbaciones, inclusive si dicha perturbación es retirada después.

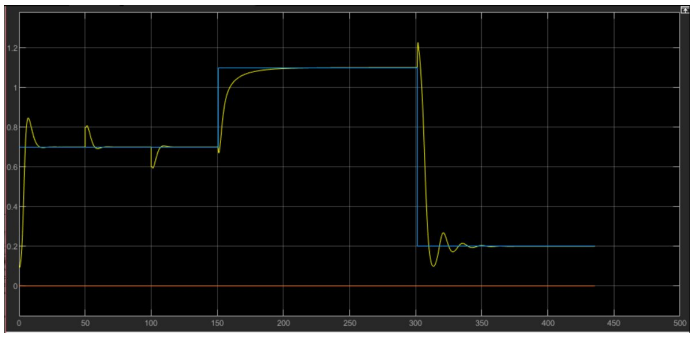

Fig 29. Respuesta del controlador por ANFIS inverso.

Labview-arduino

La última sección de la práctica consiste en la conexión de arduino y labview, donde se simulará la planta en el primero, y el controlador difuso en el segundo. Los conjuntos, rangos y reglas del controlador fueron implementadas usando una librería de sistemas difusos en labview, mientras que la planta fue recreada usando un método número en arduino. Las ecuaciones del sistema se muestran a continuación, donde no se cambio ningun otro parámetro, pues el código de ejemplo ya se encontraba implementado para orden 3.

```
// Funciones que representan las ecuaciones diferenciales del sistema
float sistemader1(float u,float x1,float x2,float x3){
float derX1;
derX1=(0.3-x1)*u-(((x2)/4)*(5*x1)/(0.5+x1))-((x3/4)*(6*x1))/(3.5+x1);
return derX1;
}

float sistemader2(float u,float x1,float x2,float x3){
float derX2;
derX2=x2*(((5*x1)/(0.5+x1))-u);
return derX2;
}

float sistemader3(float u,float x1,float x2,float x3){
float derX3;
derX3=x3*(((6*x1)/(3.5+x1))-u);
return derX3;
}
```

Fig 30. Método numérico de simulación de la planta.

Por otro lado, en labview, se usaron bloques de la librería serial que permiten tanto escribir como leer datos enviados por este medio, debido a que se envían caracteres, hizo falta la conversión a números flotantes, y adicionalmente, la implementación de un script de Matlab que permitiera la derivada del error en este entorno de simulación. Finalmente, en la figura 32 se muestra la salida del sistema a entradas de 0.2, 0.8, 1 y 0.2 nuevamente que fueron cambiando en el tiempo por medio del tanque que se ve en el lado izquierdo de la imagen. Presenta un sobreimpulso al comienzo, para luego estabilizar correctamente cada una de estas entradas.

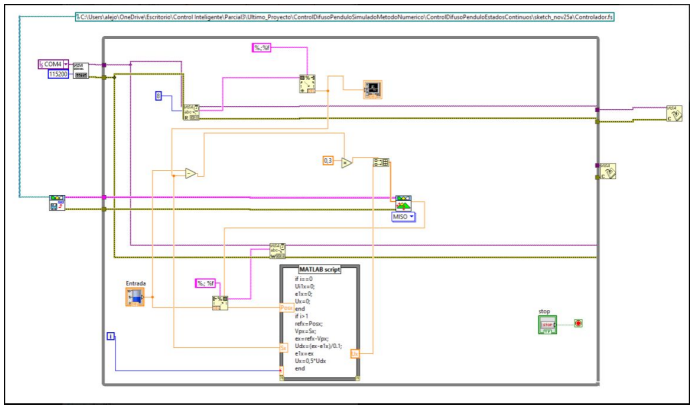

Fig 31. Diagrama de comunicación serial con arduino.

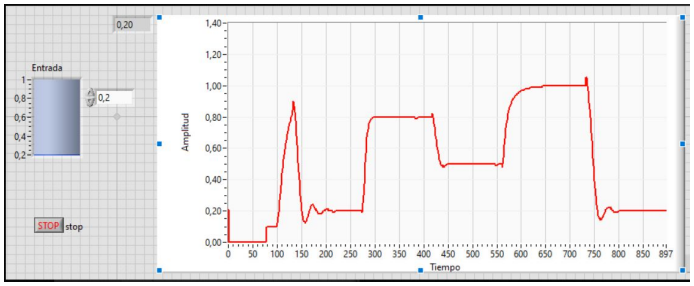

Fig 32. Respuesta del sistema.

III. CONCLUSIONES

- Es importante estudiar correctamente la dinámica del sistema pues más allá de conocer los rangos de entradas y de salida que es capaz de soportar, es importante conocer si la entrada de control será incremental o no, pues esto marca una diferencia sustancial a la hora de realizar el modelo difuso. En caso de que sea incremental no se podrían usar valores negativos de control para una magnitud real como la concentración de una sustancia, sin embargo, al hablar de una tasa de cambio si es posible, pues denota la rapidez con la que se incrementa o disminuye un valor.

- Es necesario calibrar las constantes de las entradas y salidas del sistema difuso ya que presentan alta sensibilidad a cambios, lo que podría significar el funcionamiento adecuado o incorrecto del controlador.

- A pesar de que la técnica de la frecuencia de nyquist y del tiempo de estabilización mínimo del sistema son importantes, no representan un valor incambiante y absoluto para el tiempo de muestreo o el ancho de escalón, sino indicativos d e valores máximos o mínimos de estos datos de diseño y simulación.

- Los modelos inversos y directos de los ANFIS presentan mayores sobre impulsos que sus contrapartes neuro artificiales debido presumiblemente al esfuerzo de control que se debe hacer, debido a que implementan mayor número de técnicas.

- Los problemas que presentan las redes para asimilar correctamente el seguimiento del primer escalón en los modelos directos se debe a la condición inicial del sistema, que estos diseños no poseen al estar compuestos enteramente de la dinámica y con solo ese valor inicial de referencia, impidiendo que aprenda detalles tan poco repetidos.

- Las irregularidades como el gran sobreimpulso que existe en la comunicación de labview con arduino se debe principalmente a problemas de comunicación serial persistentes en todas las actividades realizadas en el curso. Mientras que luego de un tiempo, el comportamiento es el esperado y es satisfactorio.

- Finalmente, la implementación de estos sistemas en interfaces como Labview y microcontroladores como Arduino acercan más el ejercicio teórico y práctico de la clase a aplicaciones industriales reales, fomentando un mejor aprendizaje y preparación para la vida laboral.

Referencias

[1] Marulanda, Barco, López, J., 2020. *Control Inteligente De Un Reactor Químico*. [online] Research Gate. Available at: <https://www.researchgate.net/profile/Juan_Marulanda/publication/264195775_Intelligent_Control_of_a_Chemical_Reactor_Spanish/links/53d17a0a0cf220632f3a46cb/Intelligent-Control-of-a-Chemical-Reactor-Spanish.pdf> [Accessed 22 November 2020].