UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

Optimización de un Esquema de Control Difuso mediante Algoritmos Evolutivos (Artificial BEE Colony)

Artículo Académico

.

Andrés Vladimir Arosteguí Arias

Ingeniería Electrónica

Trabajo de titulación presentado como requisito
para la obtención del título de
Ingeniero Electrónico

Quito, 10 de mayo de 2019

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

COLEGIO DE CIENCIAS E INGENIERIAS

HOJA DE CALIFICACIÓN
DE TRABAJO DE TITULACIÓN

Optimización de un Esquema de Control Difuso mediante Algoritmos Evolutivos (Artificial BEE Colony)

Andrés Vladimir Arosteguí Arias

Calificación:

Nombre del profesor, Título académico — Diego Benítez, Ph.D.

Firma del profesor _____

Quito, 10 de mayo de 2019

Derechos de Autor

Por medio del presente documento certifico que he leído todas las Políticas y Manuales de la Universidad San Francisco de Quito USFQ, incluyendo la Política de Propiedad Intelectual USFQ, y estoy de acuerdo con su contenido, por lo que los derechos de propiedad intelectual del presente trabajo quedan sujetos a lo dispuesto en esas Políticas.

Asimismo, autorizo a la USFQ para que realice la digitalización y publicación de este trabajo en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Firma del estudiante: _____

Nombres y apellidos: Andrés Vladimir Arostegui Arias

Código: 00117902

Cédula de Identidad: 1717348393

Lugar y fecha: Quito, 10 de mayo de 2019

RESUMEN

Los problemas de optimización son ampliamente utilizados en varios campos de la ciencia, la ingeniería y la industria. En muchas ocasiones, estos problemas de optimización implican un gran número de variables de decisión y objetivos complejos. A menudo, la aplicación de la teoría de lógica difusa a las estrategias de control clásicas o tradicionales requiere el uso de técnicas de optimización para dar mejor forma a las funciones de membresía y para capturar la experiencia del diseñador. Por lo tanto, el objetivo principal de este trabajo es diseñar esquemas de lógica difusa utilizando un algoritmo de optimización basado en el comportamiento inteligente de búsqueda de alimento de una colonia artificial de abejas. Posteriormente, aplicar este algoritmo para controlar la temperatura de salida de un intercambiador de calor en una planta de pasteurización PCT32-MKII y comparar los resultados con un modelo clásico de un controlador PID y un controlador Fuzzy PD+I no optimizado.

Palabras clave: Lógica Difusa, Controlador, Planta, PID, ABC.

ABSTRACT

Optimization problems are widely used in numerous fields of science, engineering, and industry. In many occasions, such optimization problems involve many decision variables and complex structured objectives. Often, applying fuzzy logic theory to the classical or traditional control strategies require the use of optimization techniques to better shape the membership functions and to capture the expertise of the designer. Therefore, the main goal of this work is to design Fuzzy Logic Schemes using an optimization algorithm based on the intelligent foraging behavior of honey bee swarm, which is Artificial Bee Colony. Then, apply this algorithm to control the output temperature of a heat exchanger in a PCT32-MKII pasteurization plant and compare the results to a classical model of a PID controller and an unoptimized Fuzzy PD+I controller.

Key words: Fuzzy Logic, Controller, Plant, PID, ABC

TABLA DE CONTENIDO

Optimización de un Esquema de Control Difuso mediante Algoritmos Evolutivos (Artificial BEE Colony).

Andrés Arostegui
Ingenieria Electrónica
Universidad San Francisco de Quito
Quito, Ecuador
andres.arostegui@estud.usfq.edu.ec

Diego Benitez
Ingenieria Electrónica
Universidad San Francisco de Quito
Quito, Ecuador
dbenitez@usfq.edu.ec

***Abstract*—Optimization problems are widely used in numerous fields of science, engineering, and industry. In many occasions, such optimization problems involve a large number of decision variables and complex structured objectives. Often, applying fuzzy logic theory to the classical or traditional control strategies require the use of optimization techniques to better shape the membership functions and to capture the expertise of the designer. Therefore, the main goal of this work is to design Fuzzy Logic Schemes using an optimization algorithm based on the intelligent foraging behavior of honey bee swarm, which is Artificial Bee Colony. Then, apply this algorithm to control the output temperature of a heat exchanger in a PCT32-MKII pasteurization plant and compare the results to a classical model of a PID controller and an unoptimized Fuzzy PD+I controller.**

***Keywords*—Fuzzy Logic, Controller, Plant, PID, ABC**

I. Introducción

Los sistemas de control clásico forman parte de un cambio fundamental en el que se buscan alternativas con mejores características de robustez, rendimiento y tolerancia a perturbaciones. De lo anterior, es posible establecer que en un modelo clásico de control se requiere obtener un entendimiento intuitivo del modelo cinemático y dinámico de una planta y para esto, es importante obtener su modelo matemático en conjunto con el sistema en lazo cerrado. Posteriormente se generan varias simulaciones para modificar valores importantes hasta encontrar la función de control que cumpla con los objetivos propuestos de la planta.

Una alternativa al modelo clásico de controladores se conoce como Lógica Difusa y es una técnica que no basa su funcionamiento en el modelo de la planta, sino que modifica su parametrización basado en la experiencia y conocimiento que posee su diseñador respecto del sistema [2]. De esta manera, y en contraste con un diseño convencional de controladores PID, adelanto-atraso o control de estados, mismos que se enfocan en construir un modelo mediante ecuaciones diferenciales, el sistema de Lógica Difusa se enfoca en adquirir un conocimiento intuitivo de cómo controlar el proceso de manera óptima .

De lo anterior, se puede establecer a las ecuaciones diferenciales como el lenguaje de control convencional, y a métodos heurísticos y reglas de control como el lenguaje de control en Lógica Difusa [3]. Adicional a esto, es importante mencionar que los métodos basados en Lógica Difusa se describen como prácticos, robustos, económicos y una alternativa inteligente para el modelamiento y control de sistemas complejos; sin embargo, solo se pueden desarrollar si existe un conocimiento por experiencia disponible para el diseñador. Por lo tanto, en el diseño de controladores difusos no existe una forma sistemática de adquirir este conocimiento para todas sus aplicaciones.

De esta manera, en aplicaciones en las que el ser humano no posee un conocimiento total del sistema resulta complejo que el diseñador pueda exponer su experticia de forma adecuada en una serie de reglas y funciones miembro, además de garantizar que dichas funciones sean óptimas para su entorno de aplicación [11]. Entonces, se debe complementar la lógica difusa con varias estrategias de optimización como Neural Networks (NN), Swarm Intelligence (SI), y Evolutionary Computation (EC). Cada una de estas técnicas tiene como objetivo principal encontrar la mejor opción de un conjunto de posibles soluciones, y aquellas que se basan en nuevas técnicas de optimización basadas en observaciones de la naturaleza constituyen un nuevo camino de inteligencia artificial capaces de optimizar parámetros con un costo de procesamiento razonable [1].

En un intento por explorar a fondo estos casos de optimización, el presente trabajo investiga sobre los beneficios de usar un algoritmo de optimización basado en enjambre de abejas llamado “Artificial Bee Colony” (ABC). Este incluye abejas obreras, exploradoras y espectadoras a diferencia del algoritmo Bee colony optimización BCO utilizado en [1] que solo cuenta con las dos primeras. El algoritmo ABC se utiliza en conjunto con un sistema difuso para controlar

la temperatura final de un intercambiador de calor en una planta de pasteurización. Después se compara los resultados obtenidos en una simulación de MATLAB con los resultados de un controlador PID clásico y con los resultados de un sistema de control difuso sin optimizar.

La estructura de este trabajo se describe a continuación. La sección II describe el proceso de modelado de la planta para obtener su función de transferencia en tiempo discreto, la sección III especifica el diseño de un controlador difuso PD+I ,la sección IV explica cómo se implementa el algoritmo de optimización ABC al esquema de control difuso, la sección V exhibe los resultados obtenidos, y por último, en la sección VI se argumentan las conclusiones del trabajo.

II. MODELO DE LA PLANTA

El presente trabajo utiliza la planta PCT32-MKII en la que se encuentran distribuidos cuatro subsistemas, que son: Bombas peristálticas, depósito de agua caliente, tubo de mantenimiento e intercambiador de calor. Estos subsistemas forman parte del proceso de pasteurización realizado por la planta de entrenamiento que se muestra a continuación [9].

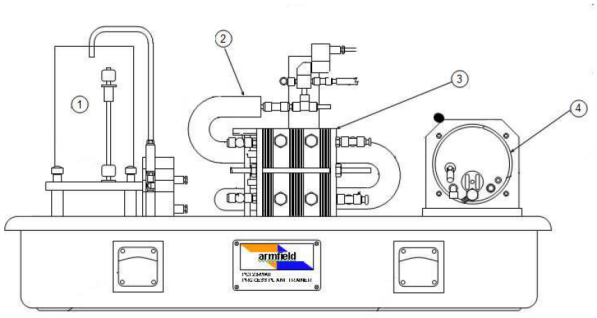

Fig. 1. Planta PCT23-MKII

Se puede observar la unidad de procesos de la planta PCT23-MKII en la Figura 1. Esta cuenta con dos recipientes de alimentación (1), un tubo de mantenimiento (2), un intercambiador de calor (3) y un depósito de agua caliente (4). Sin embargo, con el fin de aplicar la lógica difusa en una de sus aplicaciones más comunes (control de temperatura) solo se toma en cuenta el intercambiador de calor para el diseño de los controladores que siguen en las secciones posteriores.

El intercambiador de calor realiza la función de transmitir energía en forma de calor entre dos líquidos sin la necesidad de mezclarlos y como se explica en el trabajo [9], existen tres fases que componen este proceso. Primero se realiza el calentamiento del producto para el proceso de pasteurización, después se recicla la energía del producto en la fase de regeneración y finalmente se enfría el producto mediante un liquido de menor temperatura. Es importante mencionar que se modelan únicamente las fases de regeneración y calentamiento porque la fase restante no influye en el control de la temperatura de salida del producto.

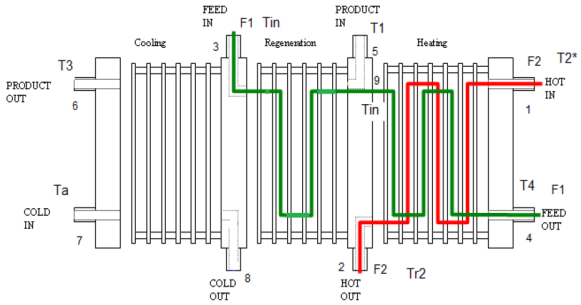

Fig. 2. Fase de Calentamiento

La Figura 2 muestra la fase de calentamiento, en la que se considera T_{in}, N_1 y N_2 (Estos últimos reemplazan a los valores de F_1 y F_2) como variables de entrada, mientras que las variables de salida son T_{r2} y T_4. De igual forma la Figura 3 muestra la fase de regeneración con variables de entrada T_1, N_1 y N_2; y variables de salida T_{in} y T_3.

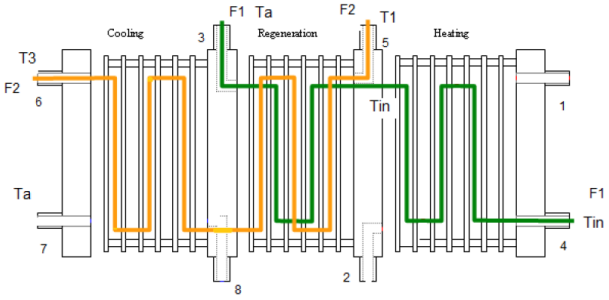

Fig. 3. Fase de Regeneración

El modelado de estos subsistemas se explica en el trabajo [9] con variaciones en las bombas peristálticas N1 y N2 alrededor del punto de equilibrio, 50% y 65% de velocidad respectivamente. Se aclara que se emplea el método por identificación y estimación de parámetros por lo que se aplica $\pm 5\%$ del punto de equilibrio en las entradas (Bombas N1 y N2) y se toman los valores de salida. Con estos datos se recurre a MATLAB y se utiliza el $toolbox$ de identificación, $ident$.

Como no existe los sensores suficientes, se modela cada fase por separado. De aquí que para cada fase del intercambiador de calor se obtiene lo siguientes valores L_f (Loss Function) y FPE (Final Prediction Error) para sistemas multivariables ARX y en espacio estados:

	ARX211	s4s3	n4s4
L_f	0.002364	0.002075	0.001935
FPE	0.002491	0.002282	0.002139

TABLE I
TABLA DE VALORES L_f Y FPE PARA LA FASE DE CALENTAMIENTO

	ARX211	s4s3	n4s4
L_f	$1.7396x10^{-5}$	$1.3413x10^{-5}$	$9.8459x10^{-6}$
FPE	$1.7900x10^{-5}$	$1.4079x10^{-5}$	$1.3990x10^{-5}$

TABLE II
TABLA DE VALORES L_f Y FPE PARA LA FASE DE REGENERACIÓN

Se observa que en ambos casos, se obtiene un mejor resultado con el sistema $n4s4$ que se representa en espacio de estados de la siguiente manera:

$$x(k+1) = Ax(k) + Bu(k) \quad (1)$$
$$y(k) = Cx(k) + Du(k) \quad (2)$$

$u(k)$: Entradas en el punto de equilibrio.

$y(k)$: Salidas en el punto de equilibrio.

En el trabajo realizado por [9] se considera a la salida T_{in} de la fase de Regeneración como una entrada en la fase de calentamiento. Por consiguiente, se obtiene el espacio de estados con la siguiente forma:

$$x(k+1) = A_i x(k) + B_i u(k) \quad (3)$$
$$y(k) = C_i x(k) + D_i u(k) \quad (4)$$

Aquí $u(k)$ representa las entradas en el punto de equilibrio del sistema conjunto:

$$u(k) = \begin{bmatrix} N_{1\delta}(k) \\ N_{2\delta}(k) \\ T_{1\delta}(k) \end{bmatrix}$$

y $y(k)$ representa las salidas del sistema conjunto:

$$y(k) = \begin{bmatrix} T_{4\delta}(k) \\ T_{r2\delta}(k) \\ T_{3\delta}(k) \end{bmatrix}$$

A partir de lo anterior se utiliza la función tf de MATLAB y se obtiene la siguiente función de transferencia:

$$G_i(z) = \frac{0.005306z^2 - 0.006926z + 0.001822}{z^3 - 2.504z^2 + 2052z - 0.5473} \quad (5)$$

La entrada a este sistema es $N_{2\delta}$ y su salida es $T_{4\delta}$. El término δ determina que son variables aplicadas en el punto de equilibrio. Su respuesta al escalón unitario es de la siguiente manera:

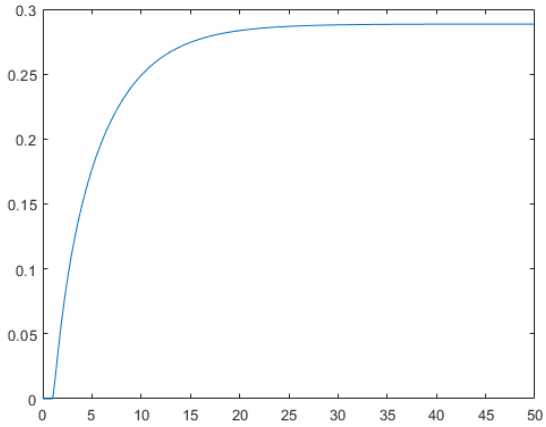

Fig. 4. Respuesta de la Planta al escalón unitario de la Planta

III. DISEÑO DE CONTROLADOR DIFUSO PD+I

Un esquema de control difuso está conformado por: un conjunto de reglas que cuantifican la descripción lingüística de diseño, un mecanismo de inferencia que simula la toma de decisiones para un control óptimo, un sistema de fuzzificación que convierte las entradas del controlador en información útil para el mecanismo de inferencia, y finalmente, un sistema de defuzzificación para transformar la salida del mecanismo de inferencia en la señal de control de la planta [2]. Estos elementos se muestran en la Figura 5

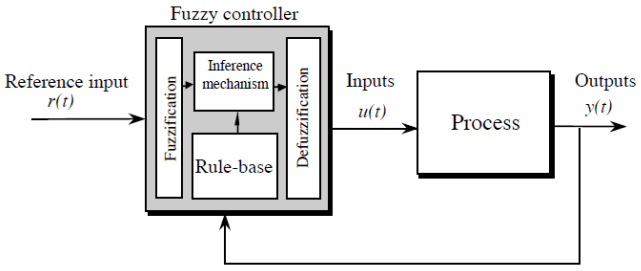

Fig. 5. Estructura de un Controlador Difuso [2]

Los elementos de la Figura 5 se construyen a partir del conocimiento obtenido previamente sobre la planta de estudio. Por está razón se escoge el método de fuzzificación definido por la siguiente ecuación:

$$\mu_{A^*}(x) = \begin{cases} \text{a} & \text{x=}\mu_i \\ 0 & \text{x}\neq \mu_i \end{cases} \quad (6)$$

μ_i es el dominio de la función miembro A, x es el valor de entrada al sistema difuso, a es el nivel de pertenencia de x en la función miembro A, $\mu_{A^*}(x)$ representa la función miembro.

Adicional a esto, se elige el sistema de inferencia de Mamdani porque, a pesar de ser el más utilizado, cumple con los objetivos de diseño de este controlador. Esté método presenta conjuntos difusos en en la entrada y en la salida del

sistema difuso, lo que simplifica el diseño del controlador ya que se ajusta mejor a la experiencia humana. Esto significa que existe un conjunto difuso para cada variable lingüística de salida después del proceso de inferencia.

Finalmente, se utiliza el centroide como el sistema de defuzzificación. Este se define por la ecuación 7.

$$z^* = \frac{\int \mu_B(Z) \cdot z dz}{\int \mu_B(Z) dz} \tag{7}$$

En donde, z^* es el valor de salida del sistema de control difuso, $\mu_B(Z)$ es la función miembro B para un conjunto difuso de salida y z es la salida del sistema de inferencia.

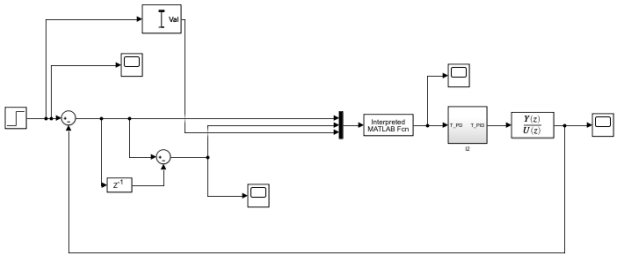

Fig. 6. Controlador PD diseñado en Simulink

El controlador difuso mostrado en la Figura 6 es Proporcional-Derivativo y por lo tanto toma las entradas e_T y Δe_T. Las mismas que se consiguen de la siguiente forma:

$$e_T = Y_{ref} - Y(S) \tag{8}$$

$$\Delta e_T = e_T(k) - e_T(k-1) \tag{9}$$

También se añade la parte integral definida por la Ecuación10 a la salida del controlador.

$$u_{PD+I} = u_{PD}(k) + \Delta u_{PD}(k) \tag{10}$$

Aquí se considera a u_{PD} como la salida del sistema difuso y a u_{PD+I} como la salida del sistema difuso incluido el bloque integral.

A. Funciones Miembro

Las funciones miembro de entrada se construyen sobre un dominio de discurso normalizado [-1,1], que se modifica con factores de escalamiento determinados por el valor de la consigna. Esto se realiza con el propósito de poder utilizar varias señales de prueba, de otro modo, el sistema estaría sujeto a ser evaluado únicamente por la respuesta al escalón unitario. Por otro lado, las funciones miembro de salida se definen en el dominio [-5,5] porque el modelado de la planta se realiza en el rango $\pm 5\%$ del punto de equilibrio.

Las funciones miembro se exponen en la Figura 7. Es importante destacar que se escogen funciones con forma triangular por su flexibilidad para cambiar sus vértices cuando se utiliza el sistema de optimización. De aquí que, las variables

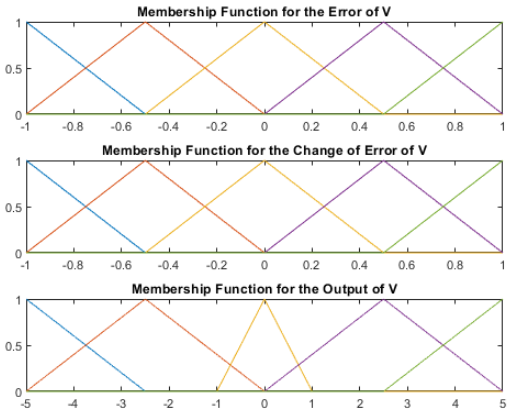

Fig. 7. Funciones Miembro del Controlador Difuso

lingüísticas del sistema difuso son: Error proporcional (e_T), variación del error (Δe_T) y velocidad de la bomba $N_{2\delta}(z)$; los valores lingüísticos de los conjuntos difusos se presentan en la Tabla III.

Definición lingüística
Más Negativo
Negativo Cero
Positivo
Más Positivo

TABLE III
VALORES LINGÜÍSTICOS UTILIZADOS EN LOS CONJUNTOS DIFUSOS DE LAS ENTRADAS e_T, Δe_T Y LA SALIDA z

B. Conjunto de reglas

El conjunto de reglas especifica el funcionamiento del controlador y esta basado completamente en la experiencia del ser humano sobre el comportamiento del sistema [2]. Para esto se necesitan las variables lingüísticas definidas previamente como los antecedentes y consecuentes de la toma de decisiones.

El número de reglas se consigue mediante la ecuación $N = x^y$ en la que x es el numero de valores lingüísticos (en este caso son 5), como se observa en la Tabla III, y y es el numero de entradas al sistema difuso, es decir, 2 (e_T y Δe_T). Por consiguiente es posible generar un conjunto de 25 reglas.

"Velocidad Bomba" $N_{2\delta}$		"Variación Error" Δe_T				
		MN	N	Z	P	MP
"Error" Δe_T	MN	MN	MN	MN	N	Z
	N	MN	MN	N	Z	P
	Z	MN	N	Z	P	MP
	P	N	Z	P	MP	MP
	MP	Z	P	MP	MP	MP

TABLE IV
VALORES LINGÜÍSTICOS UTILIZADOS EN LOS CONJUNTOS DIFUSOS DE LAS ENTRADAS e_T, Δe_T Y LA SALIDA z

Las reglas del sistema difuso se organizan en la Tabla IV y se representan de manera más general por la expresión:

R_n = IF e_T is A_i and Δe_T is A_j THEN $N_{2\delta}$ is $B_{i,j}$

En donde A_i, A_j y $B_{i,j}$ son el conjunto de valores lingüísticos relacionados con las variables e_T, Δe_T y $B_{i,j}$ respectivamente. Para esto se tiene $n = 1, 2, 3...25$ $\forall$ $(i,j) \in$ [0,5].

IV. Optimización del Controlador Difuso mediante el algoritmo ABC

El sistema de optimización ABC funciona como se muestra en el diagrama de flujo de la Figura 8. Este algoritmo simula el comportamiento de las abejas para encontrar una fuente de alimento abundante. En el caso de un sistema de optimización, el objetivo es encontrar soluciones óptimas a los parámetros del sistema difuso. Las abejas que forman parte de este proceso biológico se dividen en tres grupos: Abeja obrera, abeja exploradora y abeja espectadora.

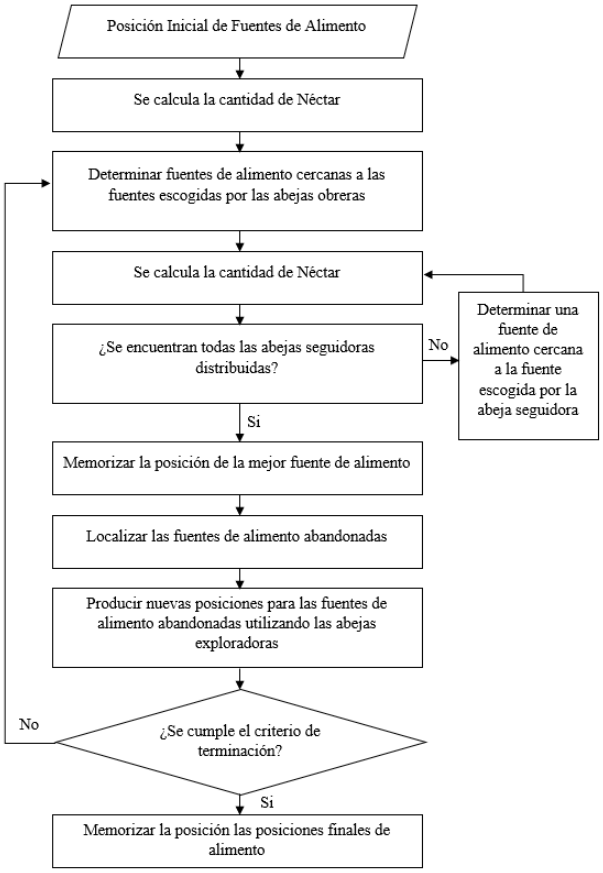

Fig. 8. Diagrama de Flujo del Algoritmo Artificial Bee Colony

Es importante añadir que las fuentes de alimento iniciales se generan aleatoriamente dentro de un rango [l_i,u_i] utilizando la Ecuación 11. Para esto es necesario modificar los valores como se muestra a continuación:

$D = 10$

u_i =[0.4 0.6 0.6 0.8 1 0.4 0.6 0.6 0.8 1]

l_i =[0.1 0.2 0.4 0.6 0.6 0.1 0.2 0.4 0.6 0.6]

$$x_{mi} = l_i + rand(0,1) * (u_i - l_i) \tag{11}$$

x_{mi} representa una fuente de alimento, es decir un vector que contiene los parámetros iniciales. l_i representa el limite inferior de un parámetro y u_i, su limite superior.

En la fase de las abejas trabajadoras se cambian los valores de parámetros aleatorios de acuerdo a la ecuación:

$$v_{mi} = x_{mi} + \theta_{mi}(x_{mi} - x_{ki}) \tag{12}$$

en la que x_k es una fuente de alimento aleatoria, i es un parámetro a cambiar de manera aleatoria y θ_{mi} es un numero aleatorio en el rango [-1,1]. De aquí se utiliza una función para calcular el ajuste de la nueva respuesta, se comparan los errores y se elige el que tenga un valor menor. Por esta razón se implementa la siguiente sección en el código:

```
        if (FitnessSol>Fitness(i)) %/*If the mutant solution is better
            %than the current solution i, replace the solution with the
            %mutant and reset the trial counter of solution i*/
            Foods(i,:)=sol;
            Fitness(i)=FitnessSol;
            ObjVal(i,:)=ObjValSol;
            trial(i)=0;
        else
            trial(i)=trial(i)+1; %/*if the solution i can not be improved,
            %increase its trial counter*/
        end
```

y se crea la siguiente función de ajuste:

```
function fFitness=calculateFitness(fObjV)
global av bv cv dv e_v avp bvp cvp dvp e_vp

[m n]=size(fObjV);

for i=1:m
  av=fObjV(i,1);
  bv=fObjV(i,2);
  cv=fObjV(i,3);
  dv=fObjV(i,4);
  e_v=fObjV(i,5);

  avp=fObjV(i,6);
  bvp=fObjV(i,7);
  cvp=fObjV(i,8);
  dvp=fObjV(i,9);
  e_vp=fObjV(i,10);

  sim('FuzzyPrueba',15);
  fFitness(i)=(max(Sys_Out1.Data))^2;
end

% sim('FuzzyPrueba',15);
% Step = stepinfo(Sys_Out.Data,Sys_Out.Time);
% fitness=(Step.Peak-1)+Step.Overshoot;
```

```
% fitness=max(Sys_Out1.Data);
% fitness=0.5*max((Sys_Out1.Data(end)))+0.5*max((
    Sys_Out2.Data(end))); %ITAE + IAU
fFitness=1./fFitness;
end
```

Esta información se comparte con las abejas espectadoras que calculan la probabilidad de que x_m sea escogida con la siguiente ecuación:

$$p_m = \frac{fit_m(x_m)}{\sum_{m=1}^{SN} fit_m(x_m)} \tag{13}$$

Con la fuente de alimento escogida por la abeja espectadora, se utiliza la Ecuación 12 para cambiar un parámetro aleatorio y comparar la nueva fuente de alimento de acuerdo a la función de ajuste.

Las soluciones que no pudieron ser mejoradas hasta cumplirse el criterio del limite máximo de intentos por fuente de comida, se abandonan y las abejas relacionadas con estas soluciones se convierten en abejas exploradoras para buscar nuevas fuentes de alimento, es decir, se genera una solución aleatoria con la ecuación para dicha fuente de alimento.

El algoritmo cuenta con parámetros de paro, se puede utilizar un numero máximo de repeticiones o hasta que se obtenga un error deseado. En este caso se deja correr el algoritmo por un numero limitado de repeticiones y se compara los resultados.

El factor que determina la eficiencia de una fuente de alimento o solución al problema, se define por la suma del error cuadrático como se muestra en la Ecuación 14.

$$ISE = \int (Y(S) - Y_{ref}(S))^2 dt \tag{14}$$

V. Resultado de la Simulación

El diagrama de control realizado en Simulink para realizar el proceso de optimización se expone en la Figura 9.

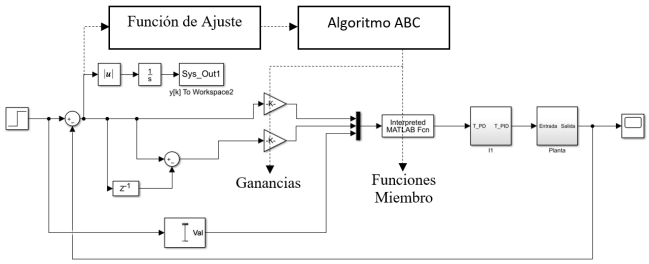

Fig. 9. Diagrama de Control para Optimización

Se necesita fijar los parámetros de control del algoritmo ABC previo al proceso de optimización. Esto se presenta a continuación:

```
%/* Control Parameters of ABC algorithm*/
NP=26; %/* The number of colony size (employed bees+
    onlooker bees)*/
FoodNumber=NP/2; %/* The number of food sources
    equals the half of the colony size */
limit=6; %/*A food source which could not be
    improved through "limit" trials is abandoned by
    its employed bee*/
maxCycle=12; %/*The number of cycles for foraging {a
    stopping criteria}*/
%/* Problem specific variables*/

%cost function to be optimized
D=12; %/*The number of parameters of the problem to
    be optimized*/
ub=[0.4 0.6 0.6 0.8 1 0.4 0.6 0.6 0.8 1 -10 -10]; %
    /*lower bounds of the parameters. */
lb=[0.1 0.2 0.4 0.6 0.6 0.1 0.2 0.4 0.6 0.6 10 10];%
    /*upper bound of the parameters.*/
```

Con estas variables, la optimización de las ganancias y los vértices de las funciones miembro para las entradas se demoró aproximadamente 1 hora. Para esto se utilizó un computador con Windows 10, 16 GB de RAM y un procesador de 8 núcleos a 4 GHz.

Las nuevas funciones miembro del sistema difuso se presentan en la Figura 10, en la que se observa claramente la diferencia con las funciones de la Figura 7.

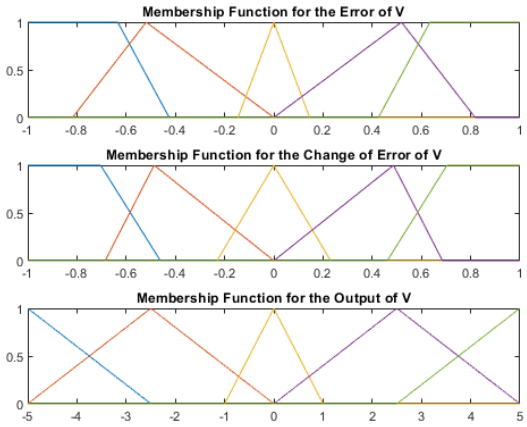

Fig. 10. Funciones Miembro Optimizadas

Adicional a esto, los valores para las ganancias de las entradas del sistema difuso obtenidos después del proceso de optimización se modifican a lo siguiente:

$$k_P* = 0.2738$$
$$k_D* = 7.8032$$

Al aumentar el parámetro NP a 50 en el algoritmo ABC, se consiguen las funciones miembro de la Figura 11 y las siguientes ganancias:

$$k_P* = 0.7823$$
$$k_D* = 5.6346$$

El controlador PID se diseña con la herramienta $PID\ Tuner$ de MATLAB. Para lo que obtienen los valores: $K_p = 4.9534$, $K_d = -2.5704$ y $K_i = 2.2019$

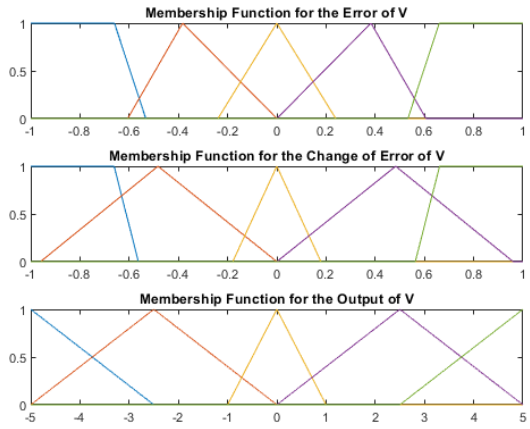

Fig. 11. Funciones Miembro Optimizadas con NP=50

De lo anterior que se logran las respuestas transitorias mostradas en las Figuras 12 y 13. Es posible observar que la respuesta al escalón unitario del controlador difuso optimizado cambia cuando se utiliza un tamaño de colonia diferente.

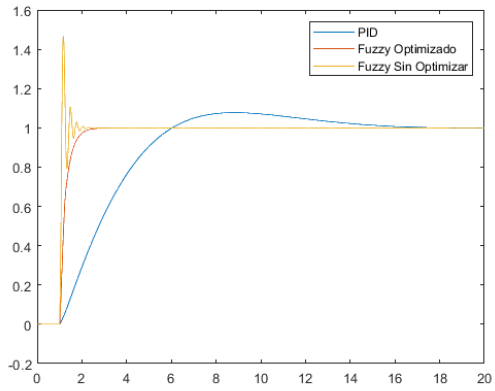

Fig. 12. Comparación de respuestas transitorias, Sistema difuso obtenido con NP=26

Finalmente se aplican varias acciones de control a los tres controladores con lo que se llega a la figura 14.

VI. Conclusión

El presente trabajo expone la implementación de un algoritmo basado en el comportamiento biológico de las abejas a un controlador difuso. Para esto se cuenta con una planta de control de temperatura final en un proceso de pasteurización. Se definen las funciones miembro del sistema difuso para las entradas: error (e_T) y variación del error (Δe_T) y para la salida: velocidad de la bomba ($N_2\delta$). Se comprueba el funcionamiento del algoritmo ABC al optimizar los vértices $abcd$ de las funciones miembro y las ganancias

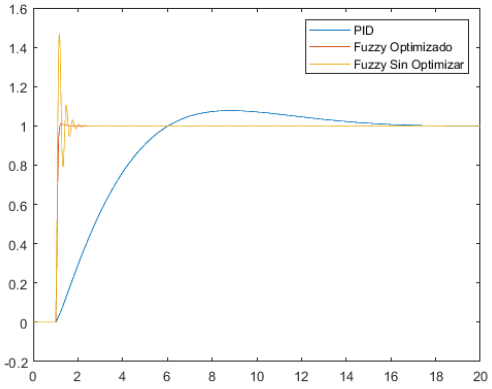

Fig. 13. Comparación de respuestas transitorias, Sistema difuso obtenido con NP=52

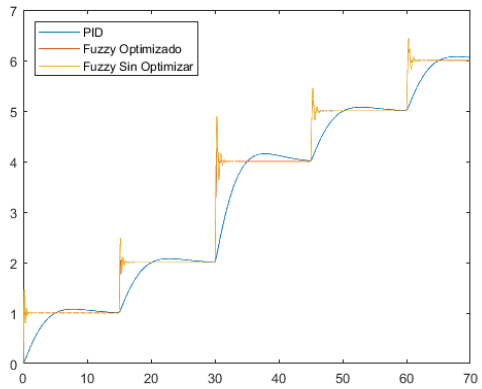

Fig. 14. Respuesta de los sistemas a varias acciones de control, Sistema difuso obtenido con NP=52

aplicadas a la entrada del controlador difuso. Esto ayudo a comparar las respuestas de un controlador PID convencional y un controlador difuso sin optimizar. En las figuras 12 y 13 se observa que el sistema difuso sin optimizar presenta un overshoot mucho mayor pero un tiempo de establecimiento menor a los otros dos sistemas; sin embargo, el sistema PID convencional presenta un overshoot menor al sistema difuso sin optimizar y un tiempo de establecimiento mayor.

Es posible notar que el sistema difuso optimizado con el algoritmo ABC presenta una mejor respuesta transitoria que los otros dos sistemas, aunque en la figura 14 se aprecia que aumenta el valor de overshoot a medida que se incrementa la consigna. Finalmente se toma a consideración el valor de tamano de colonia ya que al utilizar $NP = 26$ se consigue una respuesta transitoria optima pero con un tiempo de establecimiento menor que cuando se fija $NP = 52$; no obstante, es importante considerar que no es necesario aumentar la cantidad del tamaño de la colonia porque se obtendría resultados similares pero tomaría mas tiempo en finalizar el proceso de optimización

REFERENCES

[1] C. Caraveo, O. Castillo, "Optimization of Fuzzy Controllers Design Using the Bee Colony Algorithm," Studies in Computational Intelligence, vol. 547, pp. 163–175, Switzerland: Springer, 2014.

[2] K. Passino and S. Yurkovich, Fuzzy Control, Addison Wesley Longman, Inc, 1998.

[3] J. Chand, P. Kumar, and R. Nikhil, Evolutionary and Swarm Intelligence Algorithms. Switzerland: Springer, 2019.

[4] B. Akay and K. Demir, "Artificial Bee Colony Algorithm Variants and Its Application to Colormap Quantization," Studies in Computational Intelligence, vol. 779, pp. 26–41, Switzerland: Springer, 2019.

[5] K. Das Sharma et al., "Intelligent Adaptive Fuzzy Control," Chapter 1, Switzerland: Springer.

[6] E. Turanoglu, E, Ozceylan, and M. Servet, Particle Swarm Optimization and Artificial Bee Colony approaches to optimize of single input-output fuzzy membership functions, Turkey.

[7] S. Sivanandam, S. Deepa, and S. Sumathi, Introduction to Fuzzy Logic using MATLAB, Berlin: Springer, 2007.

[8] P. Melin, O. Castillo, J. Kacprzyk, M. Reformat and W. Melek, Fuzzy Logic in Intelligent System Design, Switzerland: Springer, 2018.

[9] L. Caiza, Diseño e implementación de estrategias de control predictivo para una planta de pasteurización a escala, Barcelona, 2012.

[10] Artificial Bee Colony (ABC) Algorithm. Intelligent Systems Research Group, Department of Computer Engineering, Erciyes University, Turkiye. Recuperado el 15 de septiembre de 2018 desde https://abc.erciyes.edu.tr/.

[11] Mohammad, R., Akbarzadek, T., Meghadi, A. Evolutionary Fuzzy Systems, Chapter 19.