IEMC-I-06-01

CONTROL ADAPTATIVO INDIRECTO CON APLICACIÓN DE TÉCNICAS DE COMPUTACIÓN FLEXIBLE.

JAVIER ALEXANDER BALLÉN SALAMANCA

UNIVERSIDAD DE LOS ANDES
MAESTRÍA EN INGENIERÍA ELECTRÓNICA Y DE COMPUTADORES
FACULTAD DE INGENIERÍA
BOGOTA
2006

IEMC-I-06-01

CONTROL ADAPTATIVO INDIRECTO CON APLICACIÓN DE TÉCNICAS DE COMPUTACIÓN FLEXIBLE.

JAVIER ALEXANDER BALLÉN SALAMANCA

Proyecto de grado para optar por el titulo de
Magíster en Ingeniería Electrónica y de Computadores

ASESOR
ALAIN GAUTHIER, Ph.D.

UNIVERSIDAD DE LOS ANDES
MAESTRÍA EN INGENIERÍA ELECTRÓNICA Y DE COMPUTADORES
FACULTAD DE INGENIERÍA
BOGOTA
2006

Dedicado a mis padres, Angel y Blanca, y a Consu
por todo el apoyo, la comprensión y los
consejos que me brindaron.

TABLA DE CONTENIDOS

LISTA DE FIGURAS

LISTA DE TABLAS

RESUMEN

En el presente trabajo se estudia la aplicación de distintas técnicas de modelamiento basadas en computación flexible al problema de control adaptativo de sistemas no lineales. Se emplea modelos difusos tipo Mamdani, modelos difusos Takagi Sugeno con aplicación de clustering en línea, y programación genética. Se evalúa la capacidad de cada una de las técnicas propuestas para realizar un modelamiento en línea del proceso de llenado de dos tanques con características no lineales y en lazo cerrado. Se implementan controladores adaptativos RST sintonizados a través de ubicación de polos de mínimo orden. Los resultados son validados tanto en simulación como en la aplicación del proceso real.

OBJETIVO

Realizar un estudio comparativo de controladores adaptativos basados en distintas técnicas de identificación de procesos no lineales, con aplicación de estrategias de computación flexible, que realice una identificación de un proceso y ajuste los parámetros del controlador para compensar los cambios en el sistema en línea, buscando mantener un comportamiento en lazo cerrado definido.

INTRODUCCIÓN

El término adaptativo significa cambiar el comportamiento de acuerdo a nuevas circunstancias. Un controlador adaptativo es un regulador que puede modificar su comportamiento en respuesta a cambios en la dinámica del proceso y a perturbaciones. En cuanto a su funcionamiento, un controlador adaptativo puede considerarse como un tipo especial de control no lineal en el que el estado del proceso puede separarse en dos escalas de tiempo que evolucionan a distinta velocidad, la escala lenta corresponde a cambios en los parámetros del sistema, por lo tanto es la misma velocidad a la que se debe cambiar la configuración del controlador; y la escala rápida, que corresponde a la dinámica del lazo de realimentación convencional [1].

El control adaptativo busca obtener controladores que puedan adaptarse a cambios en la dinámica del proceso y de las perturbaciones, y en general a imprecisiones que se tengan en el modelamiento de la planta. En general, los procesos y sistemas tienen incertidumbres, que en muchos casos dependen de variables que no se miden, pero que afectan visiblemente la dinámica de los procesos. Una de las técnicas de control adaptativas, el control adaptativo indirecto, emplea métodos de identificación, logrando un modelo aproximado del proceso a partir del cual es posible realizar los ajustes del controlador necesarios para mantener una dinámica determinada en lazo cerrado. La estructura de un controlador adaptativo indirecto se compone de un identificador, un bloque de ajuste de parámetros y un controlador ajustable, como se presenta en la figura 1.

El control adaptativo indirecto depende claramente de lo que se pueda lograr a nivel de identificación del proceso. Existe un claro compromiso entre la complejidad del modelo que se emplee y la facilidad para obtener un estimativo válido. Por ejemplo, el mejor modelo para simulación puede no ser el más adecuado para el diseño de un controlador debido a la complejidad del controlador y el grado de dificultad para su diseño [2].

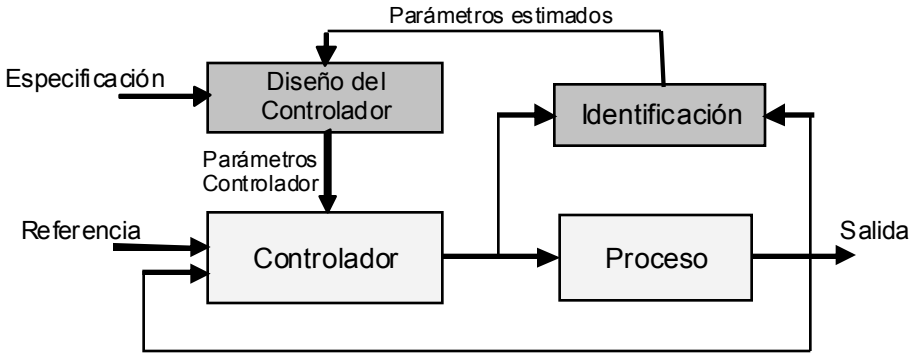

Figura 1. Estructura de un controlador adaptativo indirecto.

En la actualidad existen diversos métodos de identificación de sistemas, incluyendo técnicas lineales y no lineales, involucrando algoritmos recursivos y computación flexible e inteligencia computacional. En el presente trabajo se busca realizar un estudio de la aplicación de distintas técnicas de identificación de procesos al problema del control adaptativo, empleando técnicas de modelamiento basadas en sistemas difusos Takagi Sugeno y tipo Mamdani, y en programación genética.

En el caso del modelo difuso Takagi Sugeno se emplea el procedimiento de clustering en línea para el ajuste del mismo, el modelo difuso tipo Mamdani se deriva del controlador FMRLC (Fuzzy Model Reference Learning Control), mientras que en el caso de la programación genética se emplea un algoritmo genético para evolucionar un modelo no lineal.

1. MODELAMIENTO MEDIANTE SISTEMAS DIFUSOS TSK

El modelo Takagi Sugeno, también denominado TSK (Takagi Sugeno Kang) [3] fue propuesto en un esfuerzo por desarrollar un enfoque sistemático para la generación de reglas difusas partiendo de datos de entrada – salida [4]. Cuando se desea plantear un modelo de un sistema no lineal es posible emplear el modelo difuso TSK en donde la función de cada consecuente toma la forma $f(x,y,t)$, dando un comportamiento dinámico al modelo difuso. En la figura 2 se presenta la estructura de este tipo de sistema.

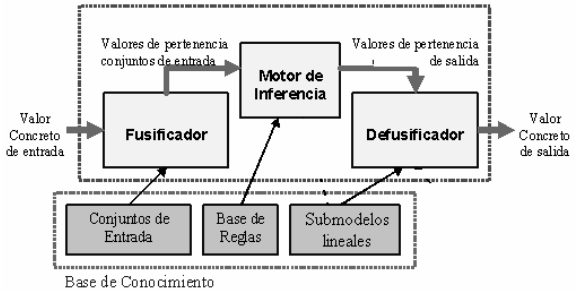

Figura 2. Estructura de un sistema difuso tipo TSK.

En este caso, debido a que en el universo de salida se definen submodelos lineales y no conjuntos difusos, las reglas de inferencia son de la forma:

$$\begin{aligned} &si\ x\ es\ A_1\ \&\ y\ es\ B_1 \Rightarrow z = z_1 = f_1(x,y,t) \\ &si\ x\ es\ A_2\ \&\ y\ es\ B_2 \Rightarrow z = z_2 = f_2(x,y,t) \end{aligned} \quad (1)$$

En la figura 3 se presenta el proceso de inferencia difusa de un sistema TSK de primer orden. La salida del sistema se calcula como la suma ponderada de los submodelos de acuerdo al grado de activación de cada regla difusa.

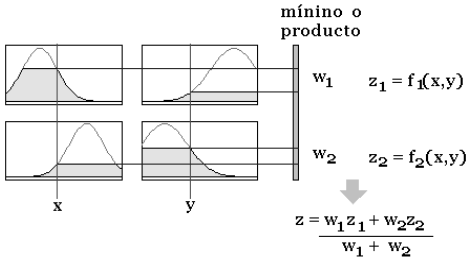

Figura 3. Proceso de inferencia en el sistema difuso TSK.

El modelamiento difuso TSK se realiza tradicionalmente a partir de la recopilación de un conjunto de datos entrada salida sobre los cuales se aplica un proceso de clustering, del cual se derivan los conjuntos difusos del antecedente (universo de entrada) y los submodelos lineales de los consecuentes en el universo de salida. Este proceso de clustering es posible mediante diferentes algoritmos tales como: Algoritmo Gustafson-Kessel, Algoritmo Maximum Likelihood Estimation (MLE), Fuzzy c-Regression Models, Redes neuronales de función de base radial, entre otros. Al permitir el planteamiento de un modelo para sistemas no lineales, basado en submodelos lineales, los modelos difusos tipo Takagi Sugeno han demostrado ser una herramienta de gran utilidad práctica para el modelamiento y control de este tipo de sistemas.

Una de las limitantes que se tenía con el enfoque de modelamiento basado en sistemas difusos TSK es la aplicación de las técnicas tradicionales de clustering, que restringía su aplicación únicamente fuera de línea. Sin embargo, recientemente fue propuesta una nueva técnica de clustering que permite hallar un elemento representativo de un conjunto de forma recursiva, haciendo viable el clustering en línea, como se presenta en [5] [6]. Con este procedimiento es posible plantear un modelo difuso TSK evolutivo (ETS, Evolving Takagi Sugeno), que modifica y actualiza su base de reglas de acuerdo al funcionamiento del proceso.

1.1. CLUSTERING EN LÍNEA

A diferencia de las técnicas tradicionales, en el clustering en línea los datos de entrenamiento son recibidos y analizados continuamente en lugar de realizar todo el procesamiento sobre un conjunto de datos predefinido. Los nuevos datos pueden proveer nueva información, que puede indicar un cambio en las condiciones de operación, una situación de falla o simplemente un cambio significativo en la dinámica del proceso [6]. La evaluación de la importancia de un nuevo dato, o potencial, se realiza de acuerdo a su proximidad espacial respecto a los datos anteriores, lo que no implica realizar un proceso

iterativo sobre todos los datos como se explica posteriormente. La dificultad que presenta esta técnica consiste en que permite encontrar un punto focal o elemento más representativo de un conjunto, pero no arroja información que permita hallar la distribución de los datos del cluster y las funciones de pertenencia involucradas en el antecedente.

El proceso del clustering en línea se inicia asumiendo que el primer dato disponible es un punto focal o centroide del primer cluster, asignándole un potencial igual a 1. A partir del siguiente dato, el potencial se calcula recursivamente empleando una función tipo Cauchy de primer orden [5] [6]:

$$P_k(z_k) = \frac{1}{1 + \frac{1}{(k-1)} \sum_{l=1}^{k-1} \sum_{j=1}^{n+1} \left(z_l^j - z_k^j \right)^2} \quad (2)$$

Donde $P_k(z_k)$ es el potencial del dato z_k calculado en el instante k; $z_l^j - z_k^j$ corresponde a la proyección de la distancia entre los dos datos proyectada en el eje j. Esta función es inversamente proporcional a la distancia de un dato respecto a los demás como se muestra en la figura 4, y hace posible el cálculo recursivo, fundamental en la implementación de un algoritmo que funcione en línea con el proceso. Por otra parte, cuando llega un nuevo dato no se resta una cantidad del mayor potencial, sino que se actualiza el potencial de todos los datos disponibles, teniendo en cuenta que, en esta técnica, en lugar de guardar todos los datos se almacenan únicamente los puntos focales y su valor de potencial.

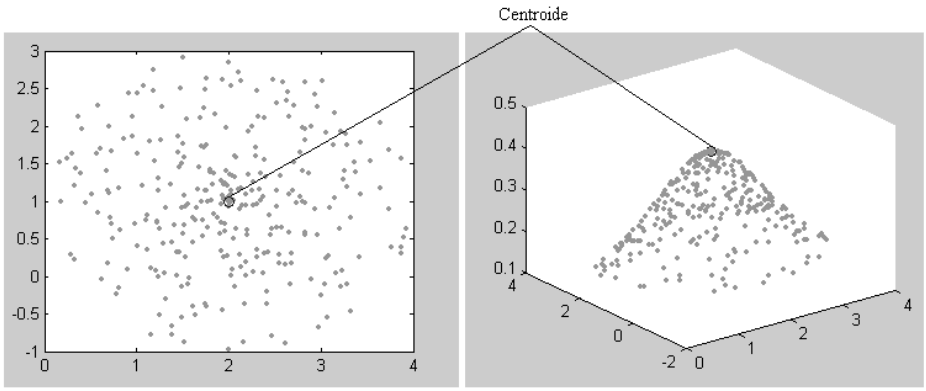

Figura 4. Función de potencial (derecha) calculada sobre un conjunto de datos (izquierda).

El potencial de un nuevo dato es calculado recursivamente de acuerdo a la siguiente expresión [5] [6]:

$$P_k(z_k) = \frac{k-1}{(k-1)(\upsilon_k + 1) + \sigma_k - 2v_k +} \quad (3)$$

Donde,

$$\sigma_k = \sigma_{k-1} + \sum_{j=1}^{n+1} \left(z_{k-1}^j\right)^2 \quad (4)$$

$$v_k = \sum_{j=1}^{n+1} z_k^j \beta_k^j \qquad \beta_k^j = \beta_{k-1}^j + z_{k-1}^j \quad (5)$$

$$\upsilon_k = \sum_{j=1}^{n+1} \left(z_k^j\right)^2 \quad (6)$$

En las ecuaciones (4), (5) y (6) se aprecia que los parámetros υ_k y v_k son calculados a partir del dato actual z_k, mientras que los parámetros β_k^j y σ_k son calculados recursivamente con base en su valor anterior (en el instante $k-1$).

Un aspecto importante consiste en analizar como un dato nuevo afecta el potencial de los datos anteriores (que incluye a los puntos focales hallados previamente), puesto que la definición empleada de potencial toma en cuenta la distancia respecto a todos los puntos. La expresión recursiva que cumple con el objetivo de actualizar el valor del potencial de los datos previamente hallados se presenta a continuación [5] [6].

$$P_k(z_i^*) = \frac{1}{1 + \frac{1}{(k-1)}\left[(k-2)\left(\frac{1}{P_{k-1}(z_i^*)} - 1\right) + \sum_{j=1}^{n+1}\left(z_k^j - z_i^j\right)^2\right]} \quad (7)$$

Donde $P_k(z_i^*)$ corresponde al potencial del punto focal z_i^* en el instante k. En la figura 5 se aprecia la evolución del valor del potencial hallado de forma recursiva en un conjunto de datos que sigue una trayectoria particular. A la izquierda se presenta la trayectoria empleada, en el centro el resultado de la función de potencial recursiva, y a la derecha se tiene el resultado al omitir el procedimiento de ajuste del potencial de los datos previos.

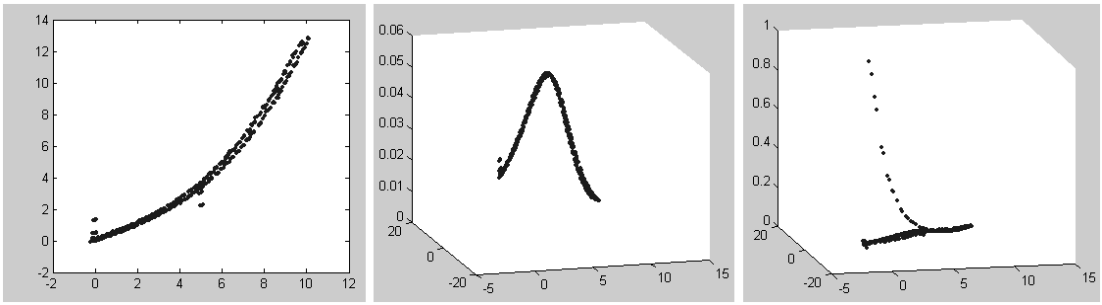

Figura 5. Potencial calculado recursivamente sobre un conjunto de datos que sigue una trayectoria.

En el procedimiento propuesto en [5] y [6] se contempla el cálculo continuo de una única función de potencial sobre todos los datos, teniendo en cuenta las siguientes observaciones:

- El primer punto focal se establece como el punto con el mayor potencial:

$$P_1 = \max_{i=1}^{N} P_i . \quad (8)$$

- El potencial de todos los demás datos se reduce de forma proporcional al potencial del dato e inversamente proporcional a la distancia desde un centro:

$$P_i' = P_i - P_k^* e^{-\beta \left\| z_i - z_k^* \right\|} \quad (9)$$

$$\beta = \frac{4}{r_b^2} \quad (10)$$

  Donde P_k^* denota el potencial del k-ésimo centro, $k = 1,2...N$; $r_b = 1.5r$ es una constante positiva que determina el radio de la vecindad que tendrá reducciones medibles en el valor del potencial debido a la cercanía con un centro. El siguiente centro se encuentra como un dato que presente el más alto potencial luego de aplicar la ecuación (9).

- Adicionalmente, se establece una relación de cercanía que será empleada para determinar si un dato se encuentra en las cercanías de uno de los centros existentes. La siguiente desigualdad establece el doble compromiso entre el valor del potencial y la distancia mínima de un dato a los centros existentes ($\delta_{\min}$).

$$\frac{\delta_{\min}}{r} - \frac{P_k^*}{P_1^*} \leq 1 \quad (11)$$

El procedimiento de clustering en línea [5] [6] es presentado en la figura 6 que describe los siguientes pasos:

1. Se establece el primer dato como centro del primer cluster. Sus coordenadas serán empleadas para definir el antecedente de la primera regla en el modelo TSK y se le asigna un potencial inicial de 1.
2. Comenzando a partir del siguiente dato se calcula el potencial de forma recursiva empleando las ecuaciones (3) a la (6).
3. El potencial de los centros existentes se actualiza con la expresión (7)
4. El potencial del nuevo dato derivados de la expresión (9) es comparado con los potenciales actualizados de los centros existentes:
   a. Si el potencial del dato nuevo es mayor al de los centros existentes el dato se agrega como un nuevo centro y una nueva regla difusa es creada.
   b. Si, además de la condición anterior, el nuevo centro es cercano a un centro existente, se elimina el centro anterior.

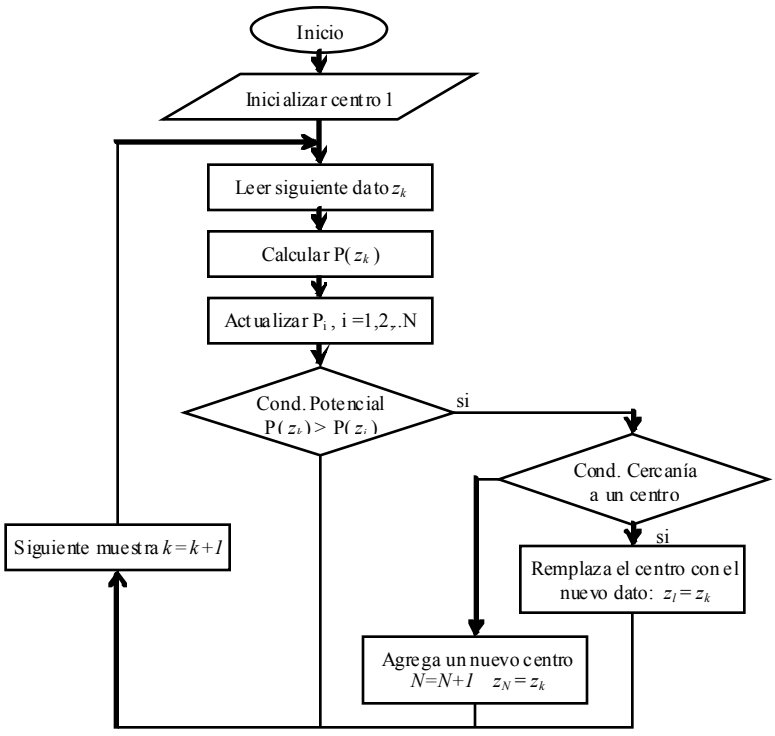

Figura 6. Diagrama de flujo del algoritmo de clustering en línea.

En la figura 7 se presenta el resultado de este algoritmo al buscar los puntos representativos, que servirán como centros de las funciones de pertenencia en el antecedente del modelo TSK, sobre una trayectoria no lineal.

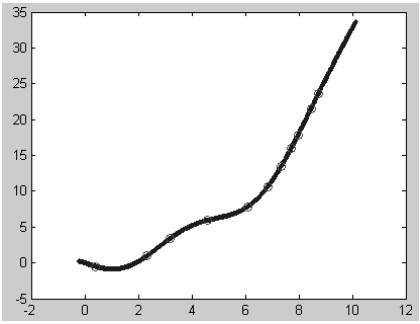

Figura 7. Resultado de clustering en línea sobre una trayectoria no lineal.

En este caso se aprecia como se encuentran puntos distribuidos de acuerdo a la trayectoria en la región comprendida entre 0 y 7 del eje x. Sin embargo a partir de x =7 se tiene un exceso de centroides si se tiene en cuenta que esta es precisamente una región que se podría aproximar por una recta.

Un aspecto importante de esta técnica es el papel del parámetro *r* dentro de los resultados arrojados por el algoritmo. En la figura 8 se ilustra esta situación. En la gráfica de la izquierda se tiene $r = 0.1$, en el centro $r = 5$, mientras que a la izquierda se presenta el resultado con $r = 10$. La concentración de centroides varia en función de *r*. Se aprecia como para $x < 7$ sería más adecuado emplear $r = 1$ (Figura 7), mientras que para $x > 7$ se tiene un mejor comportamiento con $r = 5$.

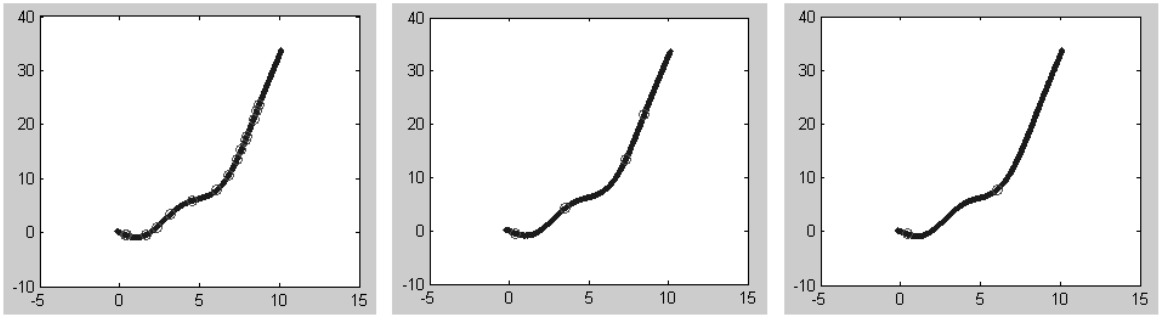

Figura 8. Efecto del parámetro *r* en el resultado de clustering en línea.

Para tratar de dar solución a la influencia del parámetro *r* se derivo una forma de clustering en línea que toma en cuenta valores de pertenencia para definir una función de potencial para cada punto focal. Esta técnica se presenta a continuación.

1.2. CLUSTERING EN LÍNEA MODIFICADO: SOLUCIÓN CON MULTIPLES FUNCIONES DE POTENCIAL.

Esta forma de clustering en línea modificado se basa en la generación de una funcion de potencial independiente para cada centroide. Para tal fin se modifica la función de potencial al incluir valores de pertenencia en su cálculo como se presenta a continuación.

$$P_k(z_k) = \frac{1}{1 + \frac{1}{(k-1)} \sum_{l=1}^{k-1} \sum_{j=1}^{n+1} \mu_l^j \left(z_l^j - z_k^j \right)^2} \quad (12)$$

Donde se incluye un vector de valores de pertenencia evaluada sobre cada eje:

$$\mu_l = \left[\mu_l^1 \quad \mu_l^2 \quad \ldots \quad \mu_l^{n+1} \right] \quad (13)$$

La función de potencial modificada ignora la distancia a los datos cuyo valor de pertenencia es cero, permitiendo que se concentre en las regiones con mayor valor de pertenencia. Para ilustrar esto se aplicó la función a una nube de datos, empleando la función de pertenencia trapezoidal mostrada en la figura 9, con los parámetros de la tabla 1.

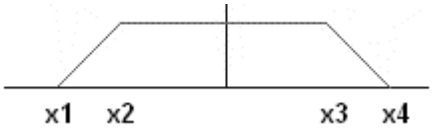

Figura 9. Función de pertenencia trapezoidal empleada en el ejemplo.

Tabla 1. Parámetros de las funciones de pertenencia empleadas para el cálculo de potencial

	μ_{F1}	μ_{F2}
x_1	0	-1
x_2	0.5	-0.8
x_3	3.5	0.8
x_4	4	1
y_1	-1	-1.2
y_2	0.5	-1
y_3	2.5	1
y_4	3	1.2

En la figura 10 se presenta el punto focal (punto rojo) encontrado dentro de la distribución de datos empleada y la gráfica de la función de potencial obtenida con la función de pertenencia μ_{F1}, mientras que en la figura 11 se hace lo propio con la función de pertenencia μ_{F2}.

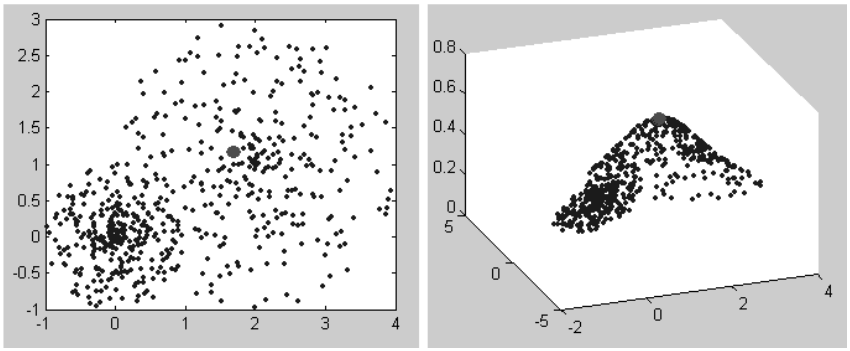

Figura 10. Resultado obtenido con la función de potencial modificada para μ_{F1}.

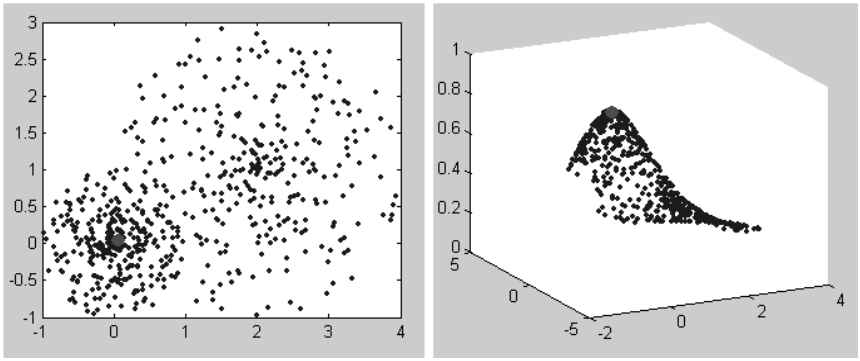

Figura 11. Resultado obtenido con la función de potencial modificada para μ_{F2}.

Además de concentrar el valor de la función de potencial en una región especifica del espacio, la introducción de los términos de pertenencia no evita que se realice el cálculo recursivo de la función. A continuación se presenta la deducción de las expresiones necesarias para llevar a cabo este procedimiento.

Para el caso de un dato nuevo, desarrollando el término al cuadrado de la ecuación (12) se tiene que:

$$P_k(z_k)=\frac{1}{1+\frac{1}{(k-1)}\sum_{l=1}^{k-1}\sum_{j=1}^{n+1}\left(\mu_l^j z_l^{j\,2}-\mu_l^j z_l^j z_k^j+\mu_l^j z_k^{j\,2}\right)} \quad (14)$$

$$P_k(z_k)=\frac{1}{1+\frac{1}{(k-1)}\left[\sum_{l=1}^{k-1}\sum_{j=1}^{n+1}\left(\mu_l^j z_l^{j\,2}\right)-\sum_{l=1}^{k-1}\sum_{j=1}^{n+1}\left(\mu_l^j z_l^j z_k^j\right)+\sum_{l=1}^{k-1}\sum_{j=1}^{n+1}\left(\mu_l^j z_k^{j\,2}\right)\right]} \quad (15)$$

A partir de (15) se definen las siguientes expresiones:

$$\sigma_k=\sum_{l=1}^{k-1}\sum_{j=1}^{n+1}\left(\mu_l^j z_l^{j\,2}\right)=\sum_{l=1}^{k-2}\sum_{j=1}^{n+1}\left(\mu_l^j z_l^{j\,2}\right)+\sum_{j=1}^{n+1}\mu_{k-1}^j\left(z_{k-1}^j\right)^2=\sigma_{k-1}+\sum_{j=1}^{n+1}\mu_{k-1}^j\left(z_{k-1}^j\right)^2 \quad (16)$$

$$v_k=\sum_{l=1}^{k-1}\sum_{j=1}^{n+1}\left(\mu_l^j z_l^j z_k^j\right)=\sum_{j=1}^{n+1}z_k^j\sum_{l=1}^{k-1}\left(\mu_l^j z_l^j\right)=\sum_{j=1}^{n+1}z_k^j\beta_k^j \quad (17)$$

$$\upsilon_k=\sum_{l=1}^{k-1}\sum_{j=1}^{n+1}\left(\mu_l^j z_k^{j\,2}\right)=\sum_{j=1}^{n+1}z_k^{j\,2}\sum_{l=1}^{k-1}\mu_l^j=\sum_{j=1}^{n+1}\left(z_k^j\right)^2\alpha_k^j \quad (18)$$

Donde:

$$\beta_k^j=\sum_{l=1}^{k-1}\left(\mu_l^j z_l^j\right)=\beta_{k-1}^j+z_{k-1}^j\mu_{k-1}^j \quad (19)$$

$$\alpha_k^j=\sum_{l=1}^{k-1}\mu_l^j=\alpha_{k-1}^j+\mu_{k-1}^j \quad (20)$$

De esta forma la expresión para el cálculo recursivo del potencial de un nuevo dato respecto a una región definida mediante los conjuntos difusos es:

$$P_k(z_k)=\frac{k-1}{k-1+\sigma_k-2v_k+\upsilon_k} \quad (21)$$

Por otra parte, es necesario ajustar el valor de potencial de los datos previos (usualmente los centros) teniendo en cuenta la pertenencia a los conjuntos difusos. En este caso, partiendo de la ecuación de potencial en el instante $k-1$.

$$P_{k-1}(z_k)=\frac{1}{1+\frac{1}{(k-1)}\sum_{l=1}^{k-2}\sum_{j=1}^{n+1}\mu_l^j\left(z_l^j-z_k^j\right)^2} \quad (22)$$

De donde se deduce que,

$$\sum_{l=1}^{k-2}\sum_{j=1}^{n+1}\mu_l^j\left(z_l^j-z_k^j\right)^2=(k-2)\left(\frac{1}{P_{k-1}(z_k)}-1\right) \quad (23)$$

Empleando esta expresión en la función de potencial en el instante *k* se obtiene la ecuación para el ajuste del potencial de los datos previos:

$$P_k(z_i^*) = \frac{1}{1+\frac{1}{(k-1)}\left[(k-2)\left(\frac{1}{P_{k-1}(z_i^*)}-1\right)+\sum_{j=1}^{n+1}\mu_k^j\left(z_k^j-z_i^j\right)^2\right]} \quad (24)$$

De esta forma es posible calcular simultáneamente el valor del potencial de un dato con referencia a diferentes centros o clusters.

El procedimiento de clustering en línea con cálculo de múltiples funciones de potencial es presentado en la figura 12. Sus principales pasos son:

1. Se establece el primer dato como centro del primer cluster y se inicializa los parámetros de la función de pertenencia con un radio máximo.
2. Comenzando a partir del siguiente dato se calcula el potencial respecto a cada centro existente de forma recursiva empleando las ecuaciones (16) a la (21).
3. El potencial de los centros existentes se actualiza con la expresión (24)
4. Se establece la pertenencia del dato a un centro de acuerdo a la relación máxima entre el potencial del dato respecto a cada centro y el potencial del punto focal:

$$i_k = \arg\max_i\left(\frac{P_i(z_k)}{P_i(z_i^*)}\right) \quad (25)$$

5. Teniendo la referencia de cual es el centroide más cercano al dato se compara el potencial del dato con el potencial actualizado del centro correspondiente:
   a. Si el potencial del dato nuevo es mayor al del centro correspondiente se reemplaza el centro.
   b. Si el dato tiene un valor de potencial se encuentra por debajo de 0.3 del valor máximo de potencial dado por el punto focal se incluye como un nuevo centro.

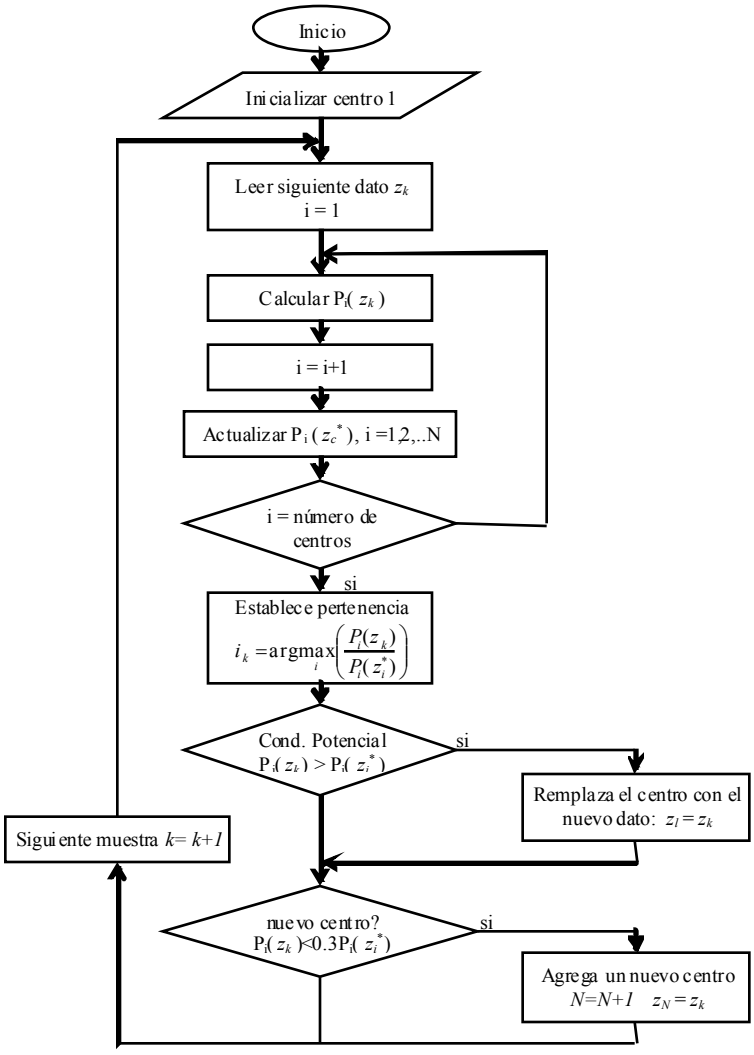

Figura 12. Algoritmo de clustering en línea con cálculo de múltiples funciones de potencial.

Esta modificación al algoritmo de clustering en línea tiene tres propósitos fundamentales: En primer lugar se tiene una función de potencial diferente para cada punto focal. En segundo lugar permitir que la reubicación de un centro se realice enfocándose de forma explicita en su vecindad. Finalmente se busca que la apertura de los conjuntos difusos correspondientes a cada punto focal pueda variar entre cierto rango, sin embargo, encontrar los parámetros más adecuados de las funciones de pertenencia de acuerdo a los datos es aún un problema abierto.

Al aplicar el clustering modificado a la misma trayectoria empleada en los ejemplos anteriores se obtiene los resultados presentados en las figura 13 y 14.

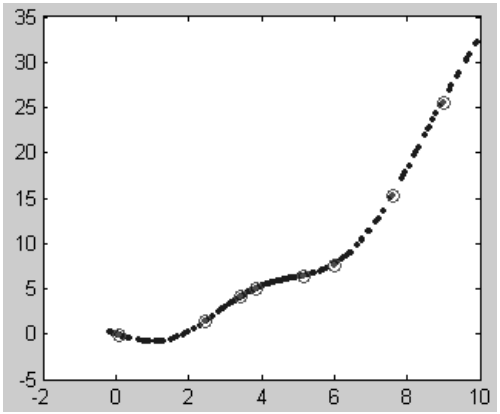

Figura 13. Puntos focales obtenidos con el algoritmo de clustering en línea con cálculo de múltiples funciones de potencial.

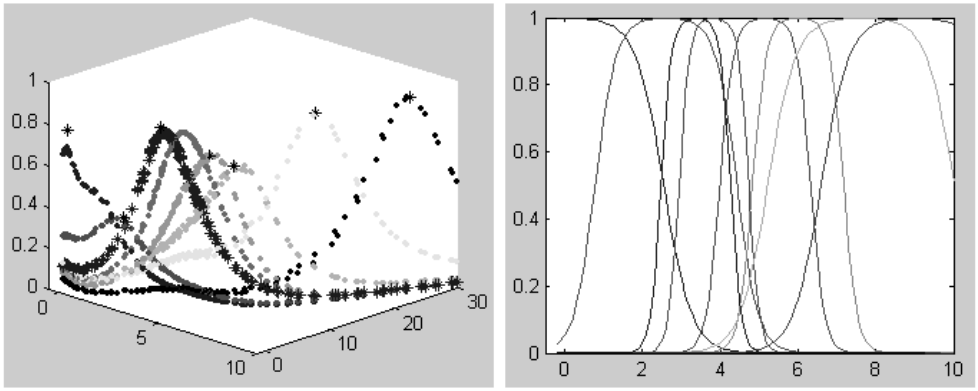

Figura 14. Funciones de potencial (izquierda) y funciones de pertenencia (derecha) obtenidas con el proceso de clustering en línea multipotencial.

Estos resultados muestran la ventaja de calcular una función de potencial correspondiente a cada punto focal encontrado, pues se encuentra una concentración de centros mayor en la zona en la que se tienen mayores comportamientos no lineales, mientras que en la región en donde es posible la aproximación de la trayectoria con una recta se tienen menos centros, como se presenta en la figura 13. En la figura 14 se aprecia el comportamiento de cada función de potencial y las funciones de pertenencia derivadas del proceso.

1.3. ESTIMACIÓN DE PARÁMETROS: MÍNIMOS CUADRADOS PONDERADOS.

Como se mencionó anteriormente, el proceso de clustering en línea es empleado para estimar la parte de antecedentes del modelo difuso TSK. Para completar el modelo es necesario aplicar una técnica que permita encontrar los submodelos lineales de la parte del consecuente. Para tal fin se emplea mínimos cuadrados recursivos ponderados [5][7].

El método de mínimos cuadrados es aplicado a una gran variedad de problemas, siendo particularmente útil para modelos matemáticos que puedan ser expresados de la forma:

$$y(i) = \varphi_1(i)\,\theta_1 + \varphi_2(i)\,\theta_2 + \ldots + \varphi_n(i)\,\theta_n \qquad (26)$$

El modelo se denomina modelo de regresión, donde y es la variable observada, θ_k son los parámetros del modelo que se desea estimar y φ_k corresponden a funciones cuyo valor es conocido, denominadas regresores [8]. En forma de vectores se tiene:

$$y(i) = \varphi^T(i)\,\theta \qquad (27)$$

Donde

$$\begin{aligned} \varphi^T(i) &= \begin{bmatrix} \varphi_1(i) & \varphi_2(i) & \cdots & \varphi_n(i) \end{bmatrix} \\ \theta &= \begin{bmatrix} \theta_1 & \theta_2 & \cdots & \theta_n \end{bmatrix}^T \end{aligned} \qquad (28)$$

Para realizar la estimación de los parámetros se deben obtener pares de observaciones y regresores $\{(y(i) \quad \varphi(i)), i = 1,2,...,t\}$ de un experimento. El objetivo del método de mínimos cuadrados es determinar el valor de los parámetros de tal manera que se obtenga una salida del modelo lo más cercana posible a los datos observados, minimizando la función de error cuadrático [8]:

$$V(\theta,t) = \frac{1}{2}\sum_{i=1}^{t}\left(y(i) - \varphi^T(i)\,\theta\right)^2 \qquad (29)$$

Debido a que el modelo es lineal en los parámetros es posible calcular una solución analítica al problema de minimización. Definiendo la matriz:

$$\Phi(t) = \begin{bmatrix} \varphi^T(1) \\ \vdots \\ \varphi^T(t) \end{bmatrix} \qquad (30)$$

La función objetivo se expresa como:

$$2V(\theta,t) = (Y - \Phi\theta)^T (Y - \Phi\theta) \qquad (31)$$

Factorizando se llega a la siguiente expresión:

$$\begin{aligned} 2V(\theta,t) = {} & Y^T \left(I - \Phi(\Phi^T\Phi)^{-1}\Phi^T\right)Y \\ & + \left(\theta - (\Phi^T\Phi)^{-1}\Phi^T Y\right)^T \Phi^T\Phi \left(\theta - (\Phi^T\Phi)^{-1}\Phi^T Y\right) \end{aligned} \qquad (32)$$

El primer término es independiente de los parámetros. El segundo término es positivo, por lo que el valor mínimo se obtiene con [8]:

$$\hat{\theta} = (\Phi^T\Phi)^{-1}\Phi^T Y \qquad (33)$$

En el caso del cálculo de los parámetros por mínimos cuadrados ponderados es necesario agregar un término asociado al peso o importancia de una muestra u observación que será relacionado con el valor de pertenencia de una muestra a una regla difusa del modelo TSK. En este contexto se tiene un conjunto de muestras $\{(y(i) \quad \varphi(i) \quad \mu(i)), i = 1,2,...,t\}$, donde $\mu(i)$ es el valor asociado de pertenencia. El objetivo consiste en hallar los parámetros que minimicen:

$$V(\theta,t) = \frac{1}{2}\sum_{i=1}^{t} \mu(i)\left(y(i) - \varphi^T(i)\,\theta\right)^2 \qquad (34)$$

$$2V(\theta,t) = (Y - \Phi\theta)^T \mathrm{M} (Y - \Phi\theta) \qquad (35)$$

Donde

$$\mathrm{M} = \begin{bmatrix} \mu(1) & 0 & \cdots & 0 \\ 0 & \mu(2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mu(t) \end{bmatrix} \qquad (36)$$

A partir de la ecuación (35) se obtiene la siguiente expresión,

$$\begin{aligned} 2V(\theta,t) = {} & Y^T \left(\mathrm{M} - \Phi(\Phi^T\mathrm{M}\Phi)^{-1}\Phi^T\mathrm{M}\right)Y \\ & + \left(\theta - (\Phi^T\mathrm{M}\Phi)^{-1}\Phi^T\mathrm{M}Y\right)^T \Phi^T\mathrm{M}\Phi \left(\theta - (\Phi^T\mathrm{M}\Phi)^{-1}\Phi^T\mathrm{M}Y\right) \end{aligned} \qquad (37)$$

En donde se concluye que los parámetros que minimizan la ecuación (34) corresponden a:

$$\hat{\theta} = (\Phi^T \mathrm{M} \Phi)^{-1} \Phi^T \mathrm{M} Y \quad (38)$$

1.3.1. CÁLCULO RECURSIVO: WRLS.

En un controlador adaptativo las observaciones se obtienen secuencialmente en tiempo real, por lo tanto es deseable realizar la estimación de forma recursiva a medida que se reciben los datos para minimizar el tiempo de cómputo. La estimación por mínimos cuadrados puede ser expresada de tal forma que el resultado en *t–1* sea empleado para el cálculo en *t*. Definiendo:

$$P(t)^{-1} = \Phi^T(t)\mathrm{M}(t)\Phi(t) = \sum_{i=1}^{t} \varphi(i)\mu(i)\varphi(i)^T \quad (39)$$

$$P(t)^{-1} = \sum_{i=1}^{t-1} \varphi(i)\,\mu(i)\varphi(i)^T + \varphi(t)\,\mu(t)\,\varphi(t)^T = P(t-1)^{-1} + \varphi(t)\mu(t)\,\varphi(t)^T \quad (40)$$

El estimativo por mínimos cuadrados ponderados se calcula como:

$$\begin{aligned} \hat{\theta}(t) &= P(t)\Phi(t)\mathrm{M}(t)Y(t) = P(t)\left[\sum_{i=1}^{t} \varphi(i)\,\mu(i)y(i)\right] \\ \hat{\theta}(t) &= P(t)\left[\sum_{i=1}^{t-1} \varphi(i)\,\mu(i)y(i) + \varphi(t)\mu(t)y(t)\right] \end{aligned} \quad (41)$$

De donde se obtiene:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)\mu(t)\,\varepsilon(t) \quad (42)$$

Donde:

$$\begin{aligned} K(t) &= P(t)\varphi(t) \\ \varepsilon(t) &= y(t) - \varphi(t)^T\hat{\theta}(t-1) \end{aligned} \quad (43)$$

El residuo ε(t) puede interpretarse como el error en la predicción de la salida *y(t)*.

Aplicando el lema de inversión de matrices [8]:

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1} \quad (44)$$

En la expresión de *P(t)* se observa que:

$$\begin{aligned} P(t) &= (\Phi^T(t)\mathrm{M}(t)\Phi(t))^{-1} = (\Phi^T(t-1)\mathrm{M}(t-1)\Phi(t-1) + \varphi(t)\mu(t)\varphi(t)^T)^{-1} \\ P(t) &= (P(t-1)^{-1} + \varphi(t)\mu(t)\varphi(t)^T)^{-1} \end{aligned} \quad (45)$$

$$P(t) = P(t-1) - P(t-1)\varphi(t)\mu(t)\left(I + \varphi(t)^T P(t-1)\varphi(t)\mu(t)\right)^{-1}\varphi(t)^T P(t-1) \quad (46)$$

Esto implica que:

$$K(t) = P(t)\varphi(t) = P(t-1)\varphi(t)\left(I + \varphi^T(t)P(t-1)\varphi(t)\mu(t)\right)^{-1} \quad (47)$$

Finalmente, el algoritmo de estimación recursiva por mínimos cuadrados ponderados está dado por:

$$\begin{aligned} K(t) &= P(t)\varphi(t) = P(t-1)\varphi(t)\left(I + \mu(t)\varphi^T(t)P(t-1)\varphi(t)\right)^{-1} \\ P(t) &= \left(I - \mu(t)K(t)\varphi(t)^T\right)P(t-1) \\ \hat{\theta}(t) &= \hat{\theta}(t-1) + \mu(t)K(t)\left(y(t) - \varphi(t)^T\hat{\theta}(t-1)\right) \end{aligned} \quad (48)$$

Este algoritmo concentra efectivamente la estimación de parámetros en una región específica del espacio de acuerdo a un valor de pertenencia, logrando una aproximación lineal en dicha región, como se presenta en el ejemplo de la figura 15.

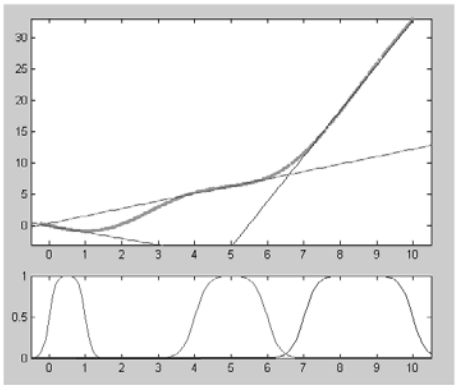

Figura 15. Ejemplo de estimación de submodelos lineales sobre una trayectoria empleando WRLS

1.3.2. ESTIMACIÓN DE PARÁMETROS EN SISTEMAS DINÁMICOS.

La aplicación del método de mínimos cuadrados a sistemas dinámicos depende de las características del modelo y de su parametrización. En el presente trabajo se emplea modelos de función de transferencia, en donde el sistema se describe mediante una ecuación en diferencias correspondiente a la siguiente expresión.

$$\begin{aligned} y(t) + a_1 y(t-1) + \ldots a_n y(t-n) \\ = b_1 u(t-1) + \ldots b_m u(t-m) \end{aligned} \quad (49)$$

Escrito de otra forma se tiene:

$$A(q)y(t) = B(q)u(t) \quad (50)$$

Donde q es el operador de adelanto ($q\ u(t) = u(t+1)$) y $A(q)$ y $B(q)$ son polinomios de la forma:

$$\begin{aligned} A(q) &= q^n + a_1 q^{n-1} + \ldots + a_n \\ B(q) &= b_1 q^m + b_1 q^{m-1} + \ldots + b_m \end{aligned} \quad (51)$$

De esta forma los vectores de parámetros y de regresores corresponden a:

$$\theta^T = \begin{bmatrix} a_1 & \ldots & a_n & b_1 & \ldots & b_m \end{bmatrix} \quad (52)$$

$$\varphi^T(t-1) = \begin{bmatrix} -y(t-1) & \ldots & -y(t-n) \\ & & u(t+n-m+1) & \ldots & u(t-n) \end{bmatrix} \quad (53)$$

Nótese que la señal de salida aparece en el vector de regresión, por lo que el modelo es denominado autoregresivo[8]. Esta representación también es conocida como modelo ARX, donde AR se refiere a la parte autoregresiva $A(q)y(t)$ y X a las variables extras o exógenas $B(q)u(t)$ [7].

1.4. ALGORITMO DE MODELAMIENTO APLICANDO CLUSTERING EN LÍNEA: SISTEMA TAKAGI SUGENO EVOLUTIVO (ETS).

El procedimiento de identificación y modelamiento aplicando clustering en línea se presenta en la figura 16. En términos generales el algoritmo presentado integra el clustering en línea y la estimación de parámetros por mínimos cuadrados ponderados. El clustering en línea se emplea para estimar las funciones de pertenencia que son incluidas en la parte del antecedente del modelo difuso TSK, mientras que mediante la estimación de parámetros por mínimos cuadrados ponderados se hallan los submodelos lineales de la parte del consecuente. Al proceso completo se le denomina modelo takagi sugeno evolutivo (Evolving Takagi Sugeno, ETS) [5][6].

Como se mencionó anteriormente en el caso del algoritmo basado en clustering en línea se emplean funciones de pertenencia tipo campana gaussiana cuya ecuación es de la forma:

$$\mu(x) = e^{-4(x-x_c)^2/r_b^2} \quad (54)$$

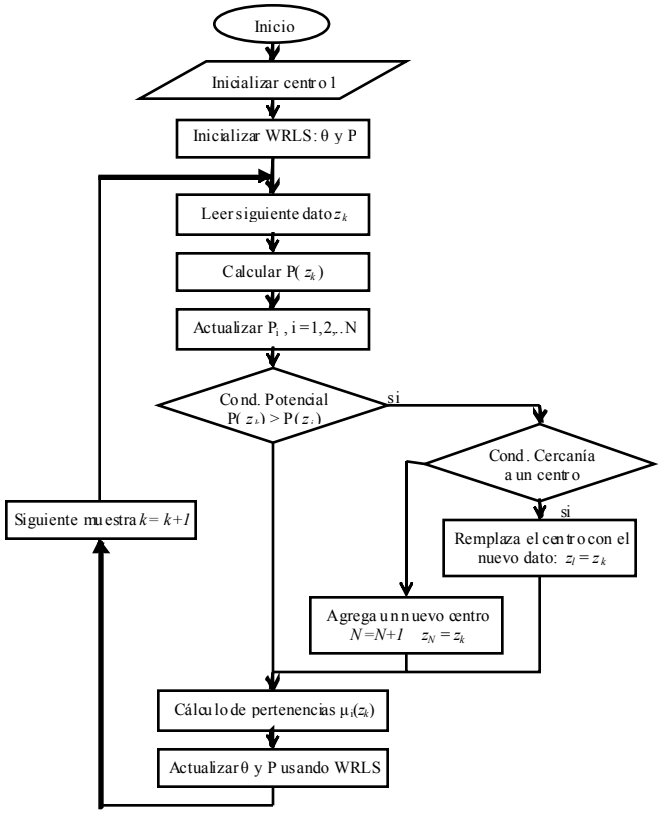

Figura 16. Procedimiento de identificación empleando clustering en línea y WRLS.

En la figura 17 se presenta las funciones de pertenencia y los submodelos lineales hallados al aplicar el ETS a una trayectoria no lineal, tomando 25000 muestras. En este caso el algoritmo arroja un conjunto de 10 reglas que se presentan en la tabla 2. En la figura 18 se tiene la aproximación de la trayectoria.

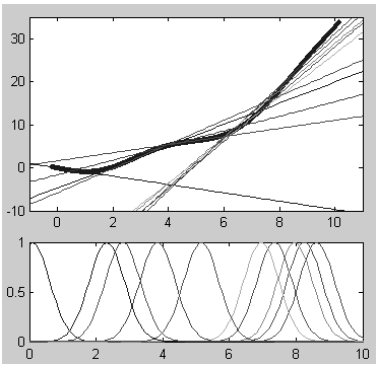

Figura 17. Submodelos lineales hallados mediante ETS sobre una trayectoria no lineal y las funciones de pertenencia correspondientes.

Tabla 2. Modelo Takagi Sugeno hallado mediante ETS.

Regla	Centro Función de Pertenencia	Submodelo lineal
1	0.0823	-0.9690x - 0.0048
2	8.6225	6.0786x - 29.6340
3	8.253	5.7817x - 27.6814
4	7.9641	5.4339x - 25.4384
5	7.3486	5.5024x - 26.8912
6	6.9567	5.0123x - 23.6052
7	3.8452	1.6992x - 1.6298
8	2.3211	2.4649x - 4.7234
9	5.1548	0.9429x + 1.6023
10	2.8005	2.8056x - 5.8074

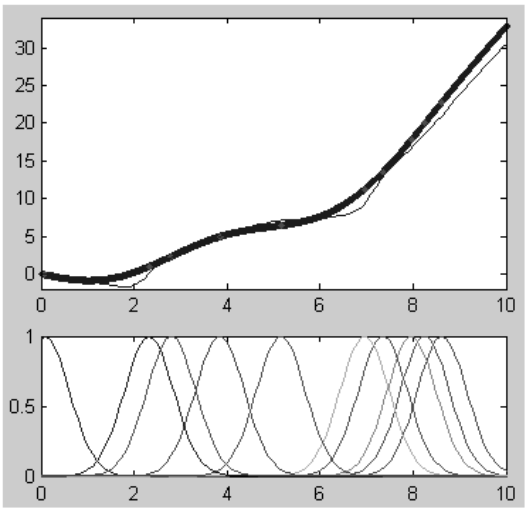

Figura 18. Resultado del Modelo Takagi Sugeno hallado mediante ETS y las funciones de pertenencia correspondientes.

1.5. ALGORITMO DE MODELAMIENTO APLICANDO CLUSTERING EN LÍNEA MULTIPOTENCIAL: SISTEMA ETS MULTIPOTENCIAL.

De forma similar al ETS, el algoritmo de identificación y modelamiento basado en clustering en línea multipotencial incluye dicha técnica para la identificación de la cantidad de reglas y los antecedentes en el modelo TSK, y el procedimiento de mínimos cuadrados ponderados que identifica los parámetros de cada submodelo lineal, como se presenta en la figura 19. En este caso se emplea una función de pertenencia tipo trapezoide sigmoide como la presentada en la figura 20, cuya expresión corresponde a:

$$\mu(x) = 0.5\tanh\left(\frac{5.88(x-x_1)}{d_1}\right) - 0.5\tanh\left(\frac{5.88(x-x_2)}{d_2}\right) \quad (55)$$

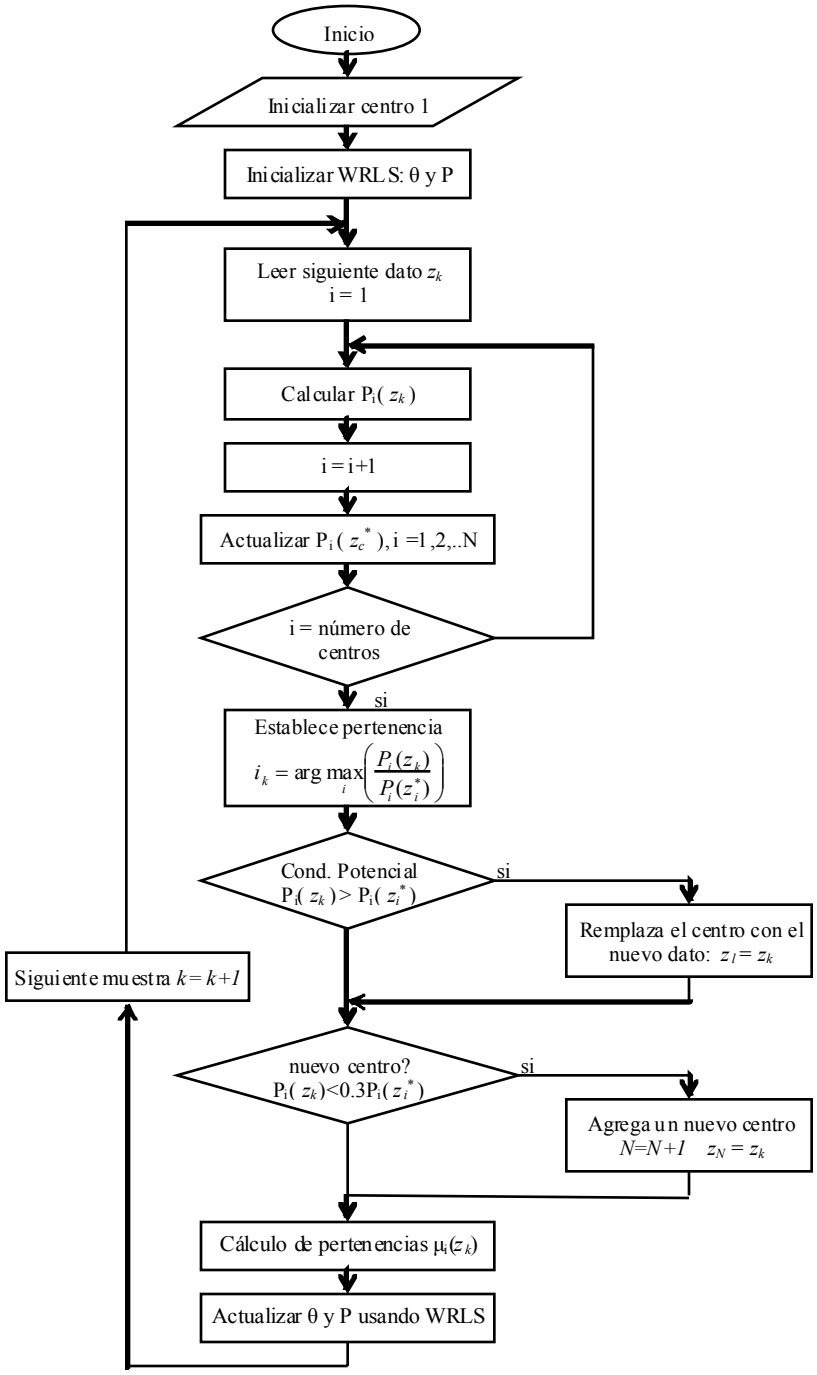

Figura 19. Procedimiento de identificación empleando clustering en línea multipotencial y WRLS

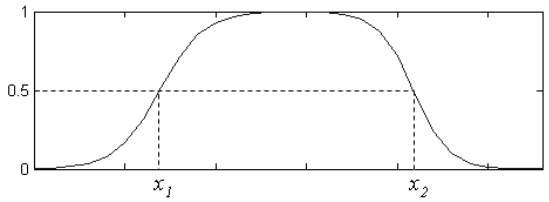

Figura 20. Función de pertenencia tipo trapezoide sigmoide.

Nótese que en este tipo de función de pertenencia no se incluye explícitamente el centro de la función. En el algoritmo ETS multipotencial se emplean dos funciones de pertenencia para cada centro, una para el cálculo de las funciones de potencial y otra para la estimación de parámetros mediante mínimos cuadrados ponderados. En el primer caso los parámetros x_1, x_2, d_1 y d_2 se hallan de acuerdo a la distancia respecto al punto focal más cercano mediante las siguientes expresiones:

$$d_{\min} = \min\left(r_{\max}, \max\left(r_{\min}, \min_i \left\|z_c - z_i\right\|\right)\right) \quad (56)$$

$$\begin{aligned} x_1 &= z_c^j - 0.8 d_{\min} \\ x_2 &= z_c^j + 0.8 d_{\min} \\ d_1 &= d_2 = 1.4 d_{\min} \end{aligned} \quad (57)$$

Para hallar las funciones de pertenencia empleadas en WRLS se tiene en cuenta tanto la distancia al punto focal más cercano como la distancia más cercana sobre el eje *j*-ésimo (dominio de la función de pertenencia), por lo que se hace necesario identificar el centro más cercano sobre el eje tanto a derecha (ecuación 62) como hacia la izquierda (ecuación 59) como se presenta en las siguientes expresiones:

$$d_{\min} = \min_i \left\|z_c - z_i\right\| \quad (58)$$

$$\begin{aligned} i_i &= \left\{i \mid z_c^j < z_i^j\right\} \\ d_{\min\_i} &= \min_{i_i} \left|z_c^j - z_i^j\right| \end{aligned} \quad (59)$$

$$p_1 = 0.15(d_{\min} + d_{\min\_i}) \quad (60)$$

$$\begin{aligned} x_1 &= z_c^j - p_1 = x_c - p_1 \\ d_1 &= 1.8 p_1 \end{aligned} \quad (61)$$

$$\begin{aligned} i_d &= \left\{i \mid z_c^j > z_i^j\right\} \\ d_{\min\_d} &= \min_{i_d} \left|z_c^j - z_i^j\right| \end{aligned} \quad (62)$$

$$p_2 = 0.15(d_{\min} + d_{\min\_d}) \quad (63)$$

$$x_2 = z_c^j - p_2 = x_c - p_2 \qquad (64)$$
$$d_2 = 1.8p_2$$

De esta forma se obtienen los conjuntos difusos presentados en la figura 21, en donde se muestra además los submodelos obtenidos mediante mínimos cuadrados ponderados. En la tabla 3 se presentan los parámetros de las funciones de pertenencia del antecedente de cada regla difusa del modelo TSK y sus correspondientes submodelos lineales. En la figura 22 se compara el modelo obtenido con la trayectoria no lineal.

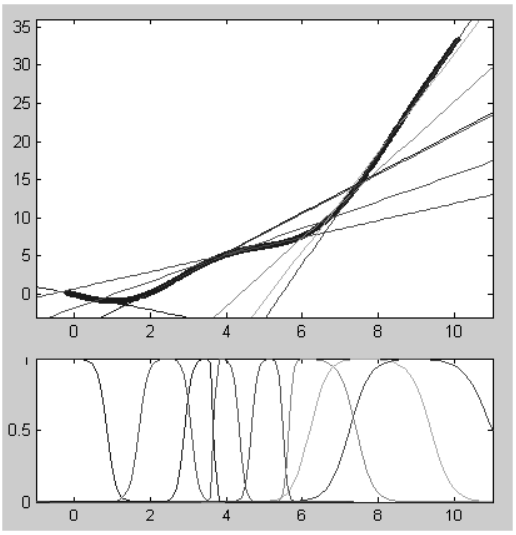

Figura 21. Submodelos lineales hallados mediante ETS sobre una trayectoria no lineal y las funciones de pertenencia correspondientes.

Tabla 3. Modelo Takagi Sugeno hallado mediante ETS multipotencial.

Regla	Centro FP	Parámetros FP		Submodelo lineal
		X_1	X_2	
1	0.10145	-9.898	0.8759	-1.0062x - 0.0065
2	5.1548	4.6551	5.5008	1.1159x + 0.7430
3	2.4664	1.6920	3.0886	2.5738x - 4.8793
4	5.9978	5.6518	7.4201	4.4576x - 19.3445
5	3.8358	3.6819	4.3355	1.7477x - 1.8090
6	7.6181	6.1958	9.3761	6.4831x - 33.2561
7	8.9908	7.2328	10.99	7.2012x - 39.2534
8	3.5291	2.9070	3.6830	2.6020x - 4.8953

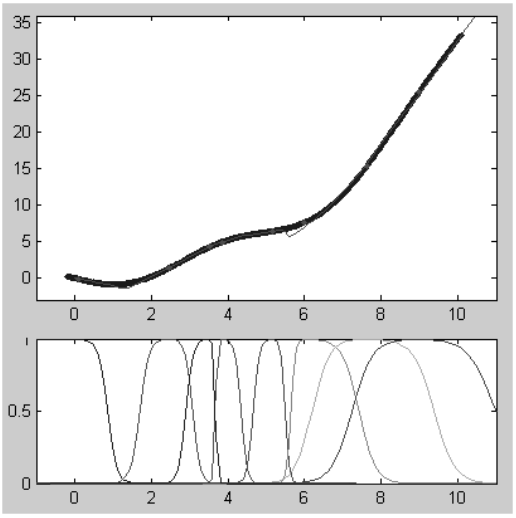

Figura 22. Resultado del Modelo Takagi Sugeno hallado mediante ETS y las funciones de pertenencia correspondientes.

2. MODELAMIENTO MEDIANTE SISTEMAS DIFUSOS MAMDANI

La estructura del sistema difuso tipo Mamdani se presenta en la figura 23. En este caso se tienen conjuntos definidos en el universo de salida. Las reglas de inferencia son de la forma:

$$\begin{aligned} &si\ x\ es\ A_1\ \&\ y\ es\ B_1 \Rightarrow z\ es\ C_1 \\ &si\ x\ es\ A_2\ \&\ y\ es\ B_2 \Rightarrow z\ es\ C_2 \end{aligned} \quad (65)$$

Donde A,B y C son conjuntos difusos correspondientes a dos universos de entrada y uno de salida.

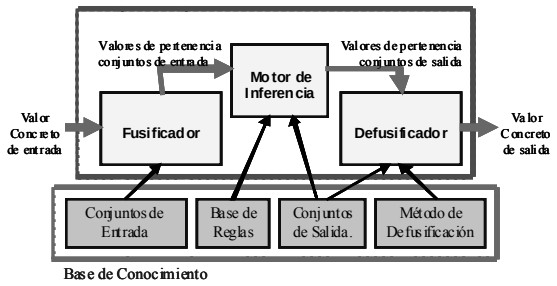

Figura 23. Estructura de un sistema difuso tipo Mamdani.

Por eficiencia computacional es usual emplear conjuntos de tipo singleton como conjuntos de salida, pues éstos solo son diferentes de cero en un punto del universo, reduciendo el procesamiento del motor de inferencia y del defusificador. Aunque a primera vista pareciera que el uso de conjuntos singleton trae consigo cambios abruptos en la salida del sistema, esta situación no se presenta gracias a las propiedades del sistema difuso. En la figura 24 se presenta un ejemplo de proceso de inferencia en un sistema tipo Mamdani.

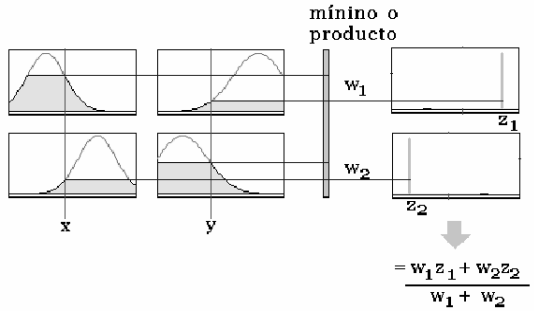

Figura 24. Proceso de inferencia en el sistema difuso Mamdani.

El proceso de identificación del proceso se realiza inspirado en el principio de funcionamiento del controlador FMRLC (Fuzzy Model Reference Learning Control) [9][10][11], en el cual se estima cuanta energía hace falta o sobra a la salida del controlador difuso para ajustar la posición de los conjuntos singleton de salida. En el caso de la identificación mediante sistema difuso tipo Mamdani la base de reglas y los conjuntos en los universos de entrada permanece constante, mientras que se cambia la ubicación de los conjuntos singleton de salida de acuerdo al error entre la salida del proceso y la salida del modelo difuso, como se presenta en la figura 25.

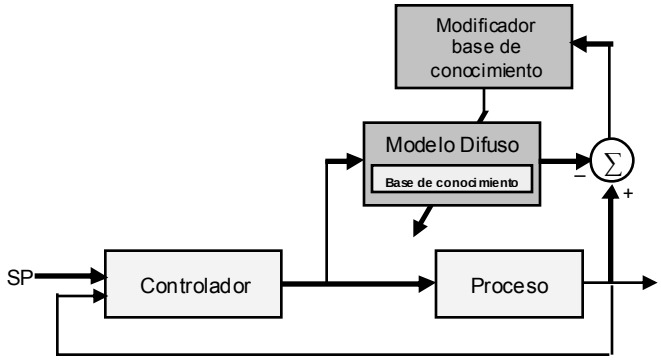

Figura 25. Diagrama de bloques de la identificación de un sistema en lazo cerrado empleando un modelo difuso Mamdani.

El modificador de la base de conocimiento se encarga de evaluar la responsabilidad de cada conjunto singleton en la salida del sistema difuso, para luego realizar la reubicación de los conjuntos de acuerdo al grado de activación de la regla correspondiente, como se muestra en el diagrama de la figura 26 y en el algoritmo de la figura 27.

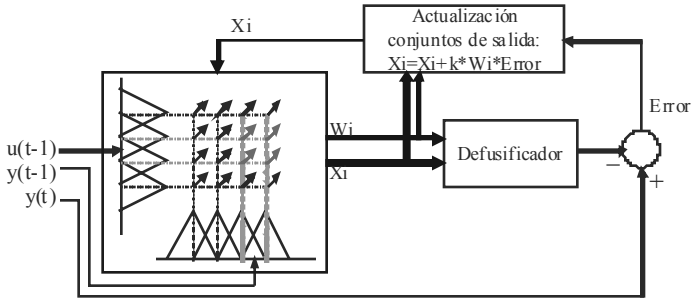

Figura 26. Proceso de actualización de los conjuntos de salida en la estimación empleando sistema difuso tipo Mamdani.

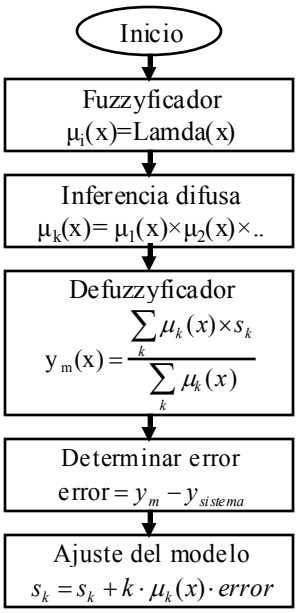

Figura 27. Algoritmo de modelamiento empleando sistema difuso tipo Mamdani, s_k corresponde a la ubicación del k-ésimo conjunto singleton.

Al finalizar el proceso de sintonía de los conjuntos singleton de salida, para lograr una estimación realmente útil para el diseño de un controlador es necesario realizar el procesamiento de los resultados del modelo difuso con el objetivo de calcular submodelos lineales en las diferentes regiones de operación.

Es evidente la necesidad de tener al menos 11 conjuntos definidos en cada conjunto de entrada para realizar la aproximación de los submodelos lineales. Sin embargo tal cantidad de reglas involucradas en el motor de inferencia difusa se incrementara exponencialmente al introducir un nuevo universo, por lo que es complicado lograr un buen estimativo para sistemas de orden mayor a 1.

En la figura 28 se presenta el resultado de aplicar esta estrategia de aproximación a la misma trayectoria no lineal de los ejemplos anteriores, realizando un procesamiento de los datos de forma secuencial, con 5000 muestras. En la figura 29 se presenta la evolución de los conjuntos singleton, mientras que en la figura 30 se presenta el comportamiento de la señal proveniente del sistema no lineal y del modelo durante las primeras 1000 muestras.

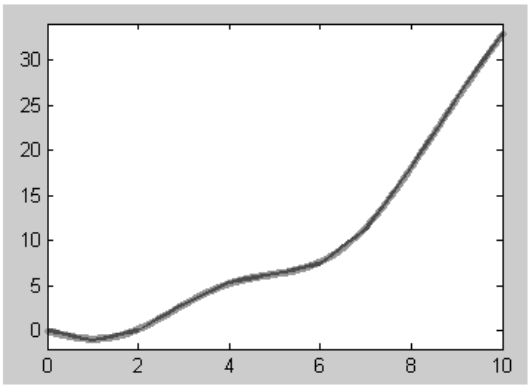

Figura 28. Aproximación de una trayectoria no lineal empleando modelo difuso tipo Mamdani basado en FMRLC.

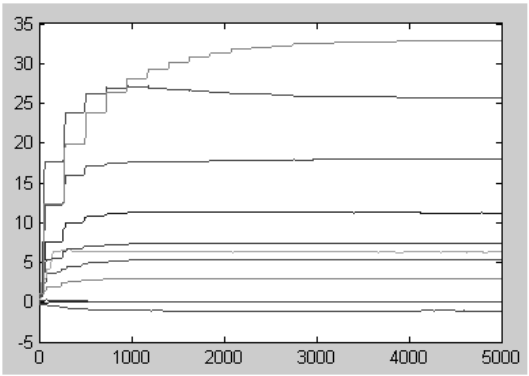

Figura 29. Evolución de la ubicación de cada conjunto difuso tipo singleton vs. muestras.

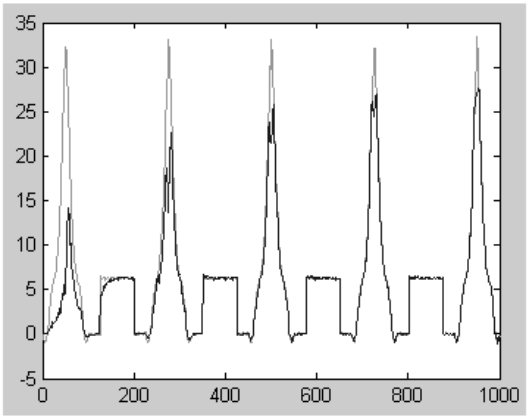

Figura 30. Comparación del comportamiento de la variable *y* en trayectoria original (verde) y del resultado del modelo difuso Mamdani (azul) vs. Muestras.

Es claro que, al tratarse de una función lineal estática, que solo requiere un conjunto de 11 reglas difusas, la evolución del modelo y la aproximación lograda son muy buenas, y se logran con una cantidad de muestras mucho menor que en los casos presentados anteriormente aplicando clustering en línea. En la figura 31 se presentan los conjuntos difusos del universo de entrada y los conjuntos singleton obtenidos en el universo de salida. En la tabla 4 se presenta la base de reglas resultante. Es importante anotar que en este ejemplo se emplean todas las reglas pues se tiene una distribución adecuada de los conjuntos de entrada, sin embargo en un modelo de un sistema dinámico no siempre es así debido a que no se conoce con certeza los valores mínimos y máximos de la señal de entrada.

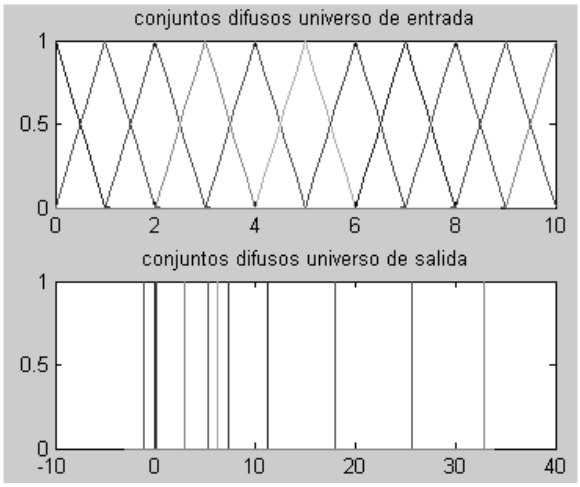

Figura 31. Conjuntos difusos del universo de entrada y del universo de salida.

Tabla 4. Base de reglas obtenida para el modelo de una trayectoria no lineal.

Regla	**Centro FP Entrada**	**Ubicación Singleton salida**
1	0	0.029
2	1	-1.07
3	2	0.043
4	3	2.93
5	4	5.34
6	5	6.28
7	6	7.38
8	7	11.22
9	8	18.01
10	9	25.68
11	10	32.91

Con base en estos resultados es posible hallar submodelos lineales realizando aproximaciones entre reglas adyacentes, como se presenta en la figura 32. Se observa que la región en la cual se ajusta cada submodelo corresponde al intervalo comprendido entre los puntos máximos de dos funciones de pertenencia contiguas, que por lo general se ubican uniformemente en el universo de entrada. Con este procedimiento se obtendría un resultado similar al de dividir el espacio de una variable y aplicar WRLS.

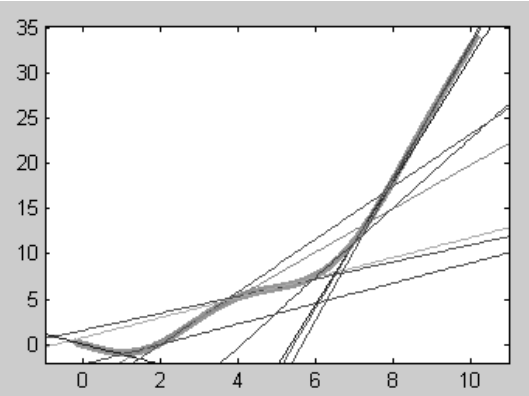

Figura 32. Submodelos lineales derivados del modelo Mamdani.

3. MODELAMIENTO MEDIANTE PROGRAMACIÓN GENÉTICA

El concepto de la programación genética (PG) fue introducido por John Kosa en 1992 como una rama de los algoritmos genéticos, en el cual las estructuras que sufren la adaptación (cromosomas) son en sí mismas programas de computador [12]. Estos programas están compuestos de funciones y terminales apropiadas al dominio del problema y son codificadas como una estructura denominada listas de procesamiento, o LISP (list processing) [13].

3.1. SECUNECIAS LISP

Las funciones se codificadas en forma de secuencias LISP descomponen la expresión de forma jerárquica, obteniendo un árbol, en donde los operadores se ejecutan de las ramas más bajas hacia arriba hasta obtener un resultado. Las secuencias LISP están compuestas de dos entidades básicas: los átomos, que pueden ser variables, constantes u operadores, y la lista, que consiste en una colección ordenada de elementos dentro de un par de paréntesis [12]. En la figura Figura 33 se presenta un ejemplo de un árbol de funciones jerárquico que representa la siguiente secuencia LISP.

$$(/\,(-\,(\times\, a\; a)\,(\times\, b\; b))\,(\times\, 2\,(+\, a\; c))) \qquad (66)$$

Que equivale a la siguiente expresión:

$$\frac{a^2-b^2}{2(a+c)} \qquad (67)$$

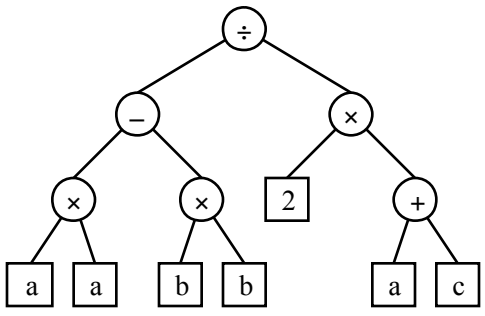

Figura 33. Ejemplo de codificación jerárquica.

3.2. MODELOS DINÁMICOS MEDIANTE LISP

En el caso de un modelo dinámico representado como una secuencia LISP, las variables corresponden a la entrada y salida del proceso en instantes anteriores (*y(t-1), y(t-2), u(t-1)*...) y se emplean los operadores suma, resta y multiplicación, obteniendo modelos de tipo ARX no lineal. Las secuencias LISP estándar emplean siempre dos ramas luego de cada operador, por lo que la representación de un modelo con muchos términos puede ser demasiado extensa. Para compactar el árbol LISP se emplea un número de ramas mayor a dos en el primer nivel. Por otra parte, se agrega un factor que multiplica cada rama en el primer nivel haciendo posible un ajuste fino del modelo mediante la estimación por mínimos cuadrados. En la figura 34 se presenta un ejemplo de las secuencias LISP modificadas.

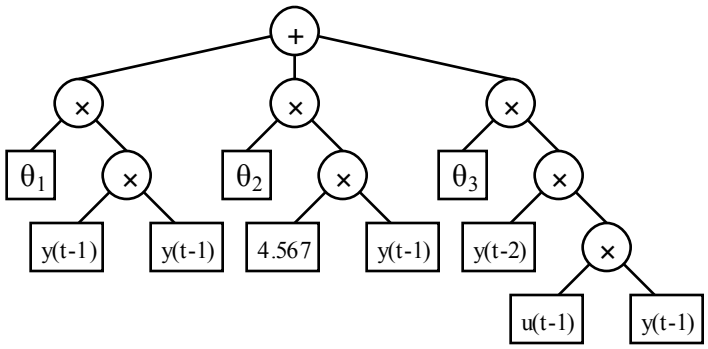

Figura 34. Secuencia LISP modificada. Incluye parámetros que es posible hallar mediante mínimos cuadrados.

3.3. EVOLUCIÓN DE LA SECUENCIA LISP MEDIANTE AG

Un conjunto de secuencias LISP generadas aleatoriamente, denominadas población inicial, serán modificadas mediante un algoritmo genético (AG) en busca de secuencias que se ajusten al comportamiento dinámico del sistema, obteniendo finalmente un modelo del mismo. Para tal fin es necesario plantear los diferentes componentes del AG de acuerdo al problema estudiado.

Para la medida de la aptitud de cada cromosoma, luego de realizar la estimación de los factores incluidos en las ramas del nivel 1 mediante mínimos cuadrados, se emplea el

inverso de la sumatoria del valor absoluto del error durante un tiempo de evaluación, como se escribe a continuación.

$$Aptitud = \frac{1}{\sum_{t=ti}^{tf} |error(t)|} \quad (68)$$

En cuanto al método de selección se emplea la ruleta con elitismo. Es decir, el cromosoma con mayor aptitud pasa directamente a la siguiente población, mientras que el resto son seleccionados aleatoriamente con una probabilidad proporcional a su aptitud [14]. Además, con el fin de contar siempre con un modelo aproximado, se tiene un cromosoma que corresponde a un modelo ARX lineal cuya estructura no se varía.

El cruce de dos cromosomas LISP se realiza de forma aleatoria de tal forma que el intercambio de material entre los progenitores sea lo menos restrictivo posible. Se escoge aleatoriamente un elemento de la secuencia correspondiente al primer progenitor, ya sea operador, constante o variable, y se reemplaza por una secuencia extraída de la misma forma del segundo progenitor. En la Figura 35 se presentan algunos ejemplos.

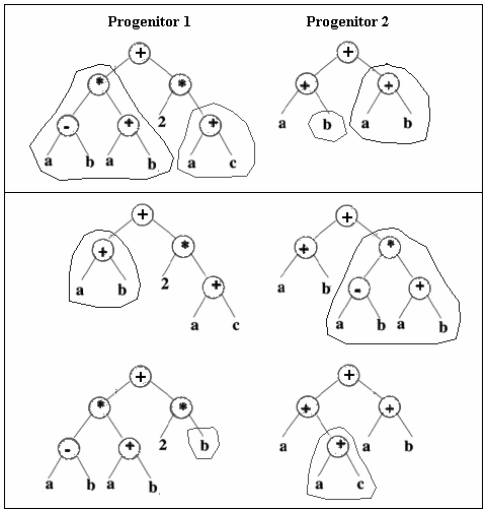

Figura 35. Ejemplo de cruce de dos secuencias LISP.

La mutación de las secuencias LISP debe permitir la exploración en secciones del espacio de búsqueda que no se hayan tenido en cuenta inicialmente. Para tal fin, este operador

puede cambiar cualquier elemento del nivel 1 hacia abajo en el árbol de la secuencia LISP. Dicho en otros términos, la mutación puede cambiar cualquier rama, constante o variable, y reemplazarla por un elemento que puede ser de otra naturaleza, es decir puede extraer una rama e insertar en su lugar una variable, etc, como se aprecia en la figura 36

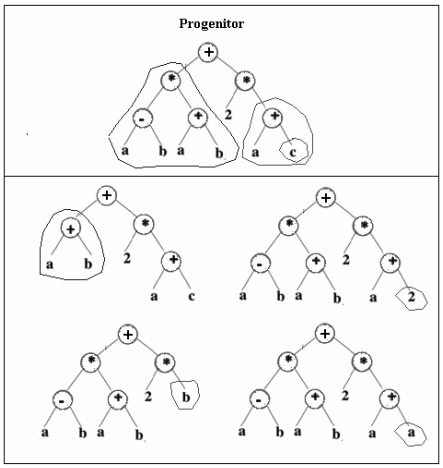

Figura 36. Ejemplo de mutaciones de una secuencia LISP.

De acuerdo a los conceptos introducidos se obtiene el siguiente algoritmo para identificación en línea de un proceso, empleando programación genética.

Paso 1. Generar Población inicial.
Paso 2. Estimar parámetros por mínimos cuadrados (recursivo).
Paso 3. Evaluar aptitud (calcular el error de cada individuo en un intervalo de tiempo – validación cruzada).
Paso 3. Selección de individuos (método de la ruleta - elitismo)
Paso 4. Cruce
Paso 5. Mutación.
Paso 6. Regresar al paso 2 hasta condición final.

Dentro de este procedimiento se distinguen tres fases. En la primera se estiman los parámetros θ empleando mínimos cuadrados recursivos [8], en la segunda se realiza el

proceso de medida de aptitud mediante validación cruzada, es decir se calcula la salida y se acumula el error de cada modelo candidato (individuo). En la última fase se emplean los resultados obtenidos para realizar el procesamiento del algoritmo genético, como se presenta en la figura 37.

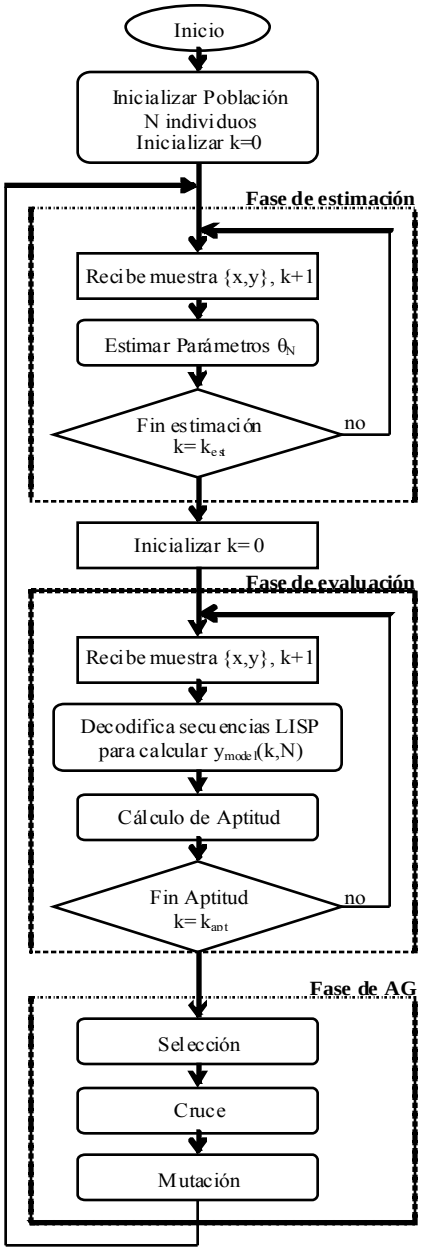

Figura 37. Algoritmo de modelamiento empleando programación genética.

La técnica de identificación mediante programación genética tiene como ventaja más fuerte su capacidad de modelar sistemas no lineales. Sin embargo, al realizar un proceso evolutivo basado en algoritmo genético para lograr un modelo, el tiempo que toma es alto, y mayor aún cuando se procesan los datos en línea con el proceso.

Con el fin de evaluar el desempeño del procedimiento, se aplicó la técnica en la identificación de un proceso discreto descrito por la siguiente ecuación:

$$y(t) = -0.1y(t-1) - 0.2y(t-2) + 0.5u(t-1) + 0.3u(t-2) + 0.02y(t-1)y(t-2) \quad (69)$$

El sistema implementado en simulink se presenta en la figura 38, en donde se incluye la ecuación del proceso dentro de un subsistema y el algoritmo de programación genética se implementa como una función de matlab. Nótese que el sistema se encuentra en lazo cerrado con realimentación unitaria.

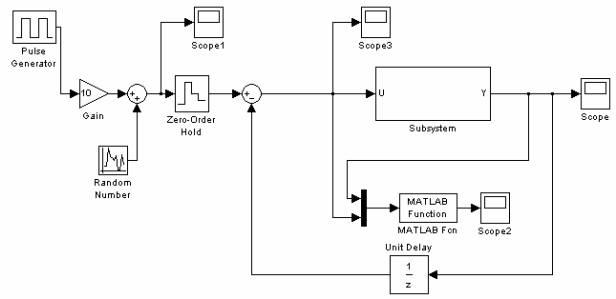

Figura 38. Modelo implementado en simulink para la prueba del método de identificación basado en programación genética.

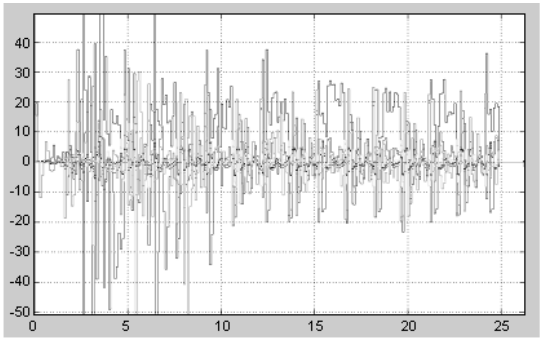

Figura 39. Ciclo de operación del método de identificación por programación genética: fase estimación: 20 seg; fase de evaluación: 5 seg; fase de procesamiento de AG 2 seg.

En la figura 39 se presenta el comportamiento del error de cada modelo candidato durante un ciclo de operación o época del algoritmo de modelamiento por programación genética. Durante los primeros 20 segundos se realiza la estimación de los parámetros θ insertados en el primer nivel de la secuencia LISP, luego se tiene la fase de evaluación durante 5 segundos y, finalmente, se realiza el procesamiento del algoritmo genético en los últimos 3 segundos.

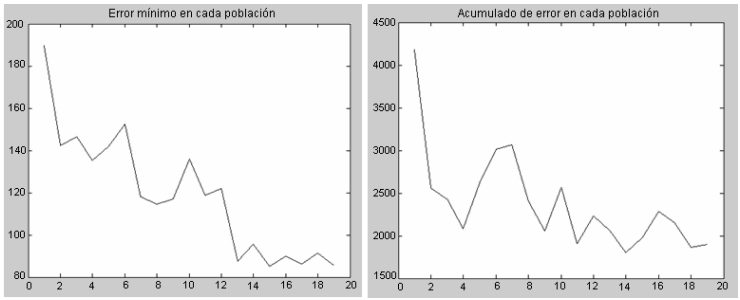

Figura 40. Comportamiento del mínimo (izquierda) y el acumulado (derecha) de error en cada población.

En la figura 40 se presenta el comportamiento del mínimo valor de error encontrado en cada población. Es notoria la evolución del modelo, pues el error presenta tendencia decreciente en la mayoría de las poblaciones. Se tiene un comportamiento similar en la suma de errores de cada población. Finalmente, luego de apenas 14 épocas, que equivale a cerca de 4000 muestras se obtiene un estimativo muy bueno que presenta a continuación.

$$y(t) = -0.092\,y(t-1) - 0.199\,y(t-2) + 0.499\,u(t-1) + 0.294u(t-2) + 0.019\,y(t-1)y(t-2) \quad (70)$$

Sin embargo, debido a que la evolución del modelo se realiza mediante un algoritmo genético, el proceso de obtención de un modelo ajustado puede variar bastante, como se presenta en el siguiente ejemplo, en el cual se empleo la siguiente ecuación para el proceso a identificar.

$$y(t) = -0.1y(t-1) - 0.2y(t-2) + 0.5u(t-1) + 0.3u(t-2) + 0.02y(t-1)y(t-2) + 0.015u(t-1)^2 \quad (71)$$

El comportamiento del error mínimo y de la suma de los errores de cada población se presenta en la figura 41. En el comportamiento del error mínimo se destacan tres regiones en las que el error logrado por el mejor individuo no varía más de 50 unidades, sin embargo en las generaciones 13, 198 y 217 se presenta una caída visible del error mínimo. Esto sugiere que en estas poblaciones se efectúo un proceso de mutación benéfico para el proceso. A pesar de esto, el acumulado del error de cada población no presenta un comportamiento decreciente, e incluso presenta picos en los cuales el error de la población alcanza niveles de más de 5×10^4, lo que puede indicar que no hubo una convergencia apropiada de los términos lineales hallados mediante mínimos cuadrados recursivos o que se presento un sobre ajuste de un modelo candidato en la población anterior.

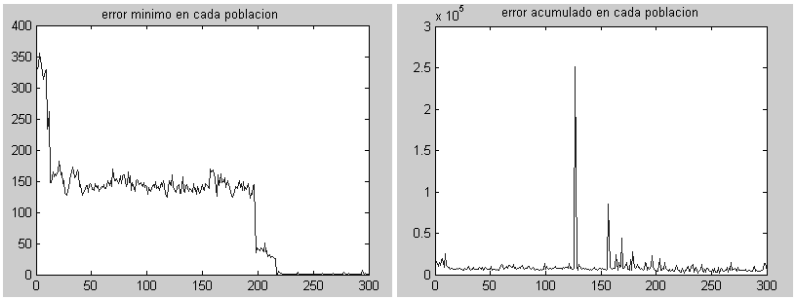

Figura 41. Comportamiento del mínimo (izquierda) y el acumulado (derecha) de error en cada población.

El modelo candidato que más se ajusto al comportamiento del proceso se presenta en la ecuación (72), en donde, a pesar de que la evolución no encuentra la estructura exacta de la ecuación del proceso, se llega a un resultado muy cercano gracias al cálculo de parámetros mediante mínimos cuadrados. El tiempo que toma en este caso llegar a un modelo ajustado es alto, pues fue necesario realizar el ciclo de evolución 220 veces, lo que equivale a 5520 segundos.

$$\begin{aligned} y(t) = &-0.1y(t-1) - 0.2y(t-2) + 0.5u(t-1) + 0.3u(t-2) \\ &+ 0.02y(t-1)y(t-2) + 0.015u(t-1)^2 - 8.73\cdot10^{-6}y(t-1)u(t-1) \end{aligned} \tag{72}$$

4. DISEÑO DEL CONTROLADOR RST POR UBICACIÓN DE POLOS DE MÍNIMO ORDEN.

A continuación se presenta la técnica de diseño del controlador por ubicación de polos de mínimo orden, expuesta en [8], que será empleada dentro del controlador adaptativo indirecto para determinar los parámetros apropiados del regulador con el objeto de lograr una dinámica en lazo cerrado determinada. Se asume que el proceso es un sistema de una entrada y una salida (SISO) descrito por la ecuación en tiempo discreto,

$$A(q)y(t) = B(q)u(t) \quad (73)$$

Donde *y* corresponde a la salida, *u* a la entrada, y *A(q)* y *B(q)* son polinomios definidos con el operador de adelanto *q*. El orden de los polinomios corresponde a,

$$\begin{aligned} \deg A &= n \\ \deg B &= m = n - b_0 \end{aligned} \quad (74)$$

En algunos casos es conveniente tener el modelo del proceso en términos del operador de atraso q^{-1} por medio del polinomio reciproco,

$$A^*(q^{-1}) = q^{-n}A(q) \quad (75)$$

En general, se asume que *A* y *B* no tienen factores comunes y que el coeficiente de mayor orden en *A* es 1. Un controlador lineal puede ser descrito por,

$$Ru(t) = Tu_c(t) - Sy(t) \quad (76)$$

Donde *R*, *S* y *T* son polinomios. Esta ley de control representa una realimentación negativa a través de la función de transferencia *–S/R* y alimentación hacia adelante (feedforward) a través de la función de transferencia *T/R*, teniendo entonces dos grados de libertad [8]. El diagrama de bloques del controlador en lazo cerrado se presenta en la figura 42.

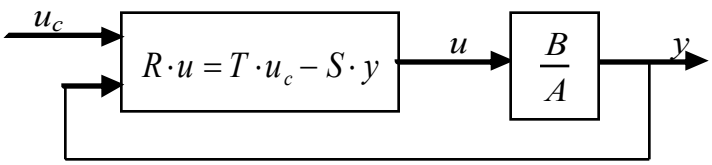

Figura 42. Estructura del sistema en lazo cerrado con el controlador RST.

Se tienen las siguientes funciones de transferencia para el sistema en lazo cerrado:

$$y(t) = \frac{BT}{AR + BS} u_c(t) \quad (77)$$

$$u(t) = \frac{AT}{AR + BS} u_c(t) \quad (78)$$

El polinomio característico en lazo cerrado corresponde a:

$$AR + BS = A_c \quad (79)$$

En el método de diseño se especifica el polinomio característico A_c deseado, de donde es posible obtener los polinomios R y S, empleando la ecuación diofantina (79). Esta ecuación tiene solución si no existen factores comunes entre los polinomios A y B, y puede presentar una solución pobre si dichos polinomios tienen factores muy cercanos [8].

Además de la ecuación diofantina, es necesario incluir otras condiciones para hallar el polinomio T. Teniendo un modelo deseado o de referencia para el comportamiento del sistema de la forma,

$$A_m y(t) = B_m u_c(t) \quad (80)$$

Se debe cumplir la siguiente condición:

$$\frac{BT}{AR + BS} = \frac{BT}{A_c} = \frac{B_m}{A_m} \quad (81)$$

Esta condición, conocida como seguimiento de modelo, busca que la respuesta del sistema en lazo cerrado sea de acuerdo a la ecuación (80). La ecuación 81 implica que puede haber cancelación de factores entre BT y A_c. Así, es necesario factorizar B como:

$$B = B^+ B^- \quad (82)$$

Donde B^+ es un polinomio cuyo coeficiente de orden superior es 1 y sus ceros son estables y, por lo tanto es posible cancelarlos, y B^- que contiene los factores inestables. B^- no puede ser cancelado, por lo tanto debe ser un factor dentro de B_m, como se presenta a continuación,

$$B_m = B^- B'_m \quad (83)$$

De forma similar, debido a que B^+ es cancelado, este polinomio debe ser factor de A_c,

$$A_c = A_o A_m B^+ \quad (84)$$

Puesto que B^+ es factor de B y A_c, de la ecuación diofantina se deduce que también dívide a R, por lo tanto se tiene,

$$R = R' B^+ \quad (85)$$

Entonces la ecuación diofantina se reduce a:

$$AR' + B^- S = A_0 A_m = A'_c \quad (86)$$

Finalmente, introduciendo (82) (83) y (84) en (81) se obtiene que:

$$T = A_o B_m'$$

Por otra parte, para obtener un controlador causal es necesario tener en cuenta las siguientes condiciones acerca del grado de los polinomios R, S y T,

$$\deg S \leq \deg R \quad (87)$$

$$\deg T \leq \deg R \quad (88)$$

Debido a que la ecuación diofantina posee diferentes soluciones, es necesario seleccionar aquella que resulte en un controlador de menor orden, denominada solución de mínimo orden. Puesto que $\deg B < \deg A$, el término de mayor orden del polinomio característico en lazo cerrado es AR, por lo tanto,

$$\deg R = \deg A_c - \deg A \quad (89)$$

Por otra parte, si R^* y S^* son soluciones de la ecuación diofantina, entonces

$$\begin{aligned} R &= R^* + QB \\ S &= S^* - QA \end{aligned} \quad (90)$$

También son soluciones, donde Q es un polinomio arbitrario. Por lo tanto es posible encontrar una solución tal que $\deg S < \deg A$ o $\deg S \leq \deg A - 1$. Además es necesario que el retardo de tiempo del modelo deseado sea igual o mayor que el del proceso. De esta forma se logra el siguiente algoritmo.

Paso 0. Condiciones de compatibilidad.

$$\deg A_m = \deg A \quad (91)$$

$$\deg B_m = \deg B \quad (92)$$

$$\deg A_0 = \deg A - \deg B^+ - 1 \quad (92)$$

$$B_m = B^- B'_m \quad (93)$$

Paso 1. Factorizar $B = B^+ B^-$, donde el coeficiente de mayor orden de B^+ es 1.

Paso 2. Encontrar la solución de R' y S con $\deg S < \deg A$ de la ecuación:

$$AR' + B^- S = A_0 A_m \quad (94)$$

Paso 3. Calcular $R = R'B^+$ y $T = A_0 B'_m$ para hallar el valor de la señal de control.

En el caso de poder cancelar todos los ceros del proceso, el proceso de diseño se simplifica, obteniendo que $\deg A_0 = \deg A - \deg B - 1$. Además, es usual escoger $B_m = A_m(1)q^{d_0}$ para simplificar la factorización del paso 1, concluyendo que $B^- = b_0$ y $B^+ = B/b_0$. De acuerdo a esto se tiene que $T = A_m(1)q^{d_0}/b_0$, y la ecuación característica en lazo cerrado se convierte en $A_c = B^+ A'_c$. La ecuación diofantina es entonces:

$$AR' + b_0 S = A'_c = A_0 A_m \quad (95)$$

Esta ecuación es fácilmente resuelta al tener en cuenta que al dividir $A_0 A_m$ en A se tiene R' como cociente y $b_0 S$ como residuo.

En el caso particular de un sistema de segundo orden, donde:

$$H(q) = \frac{b_0 q + b_1}{q^2 + a_1 q + a_2} \quad (96)$$

Se tiene $\deg A = 2$ y $\deg B = 1$, por lo tanto el controlador es de primer orden y el sistema en lazo cerrado de tercer orden. Teniendo el modelo del comportamiento deseado como:

$$\frac{B_m(q)}{A_m(q)} = \frac{b_{m0} q}{q^2 + a_{m1} q + a_{m2}} \quad (97)$$

Las características dinámicas dependen del denominador de este modelo, mientras que b_{m0} se selecciona para que la ganancia en estado estable sea 1. De acuerdo al procedimiento de diseño descrito se obtiene:

$$B^{+}(q) = q + \frac{b_1}{b_0} \quad (98)$$

$$B^{-}(q) = b_0 \quad (99)$$

$$B'(q) = \frac{b_{m0}}{b_0} q \quad (100)$$

Los polinomios R, S y T son de orden 1, por lo que $R' = 1$. Debido a las condiciones de compatibilidad $\deg A_0 = 0$, con lo cual $A_0(q) = 1$. De esta forma, la ecuación diofantina es:

$$(q^2 + a_1 q + a_2) + b_0(s_0 q + s_1) = q^2 + a_{m1} q + a_{m2} \quad (101)$$

Igualando los coeficientes se obtiene:

$$s_0 = \frac{a_{m1} - a_1}{b_0} \quad (102)$$

$$s_1 = \frac{a_{m2} - a_2}{b_0} \quad (103)$$

Finalmente, el controlador se caracteriza con los siguientes polinomios:

$$R(q) = B^{+} = q + \frac{b_1}{b_0} \quad (104)$$

$$S(q) = s_0 q + s_1 \quad (105)$$

$$T(q) = A_0 B'_m = \frac{b_{m0} q}{b_0} \quad (106)$$

Este resultado será empleado para el ajuste automático del controlador adaptativo.

5. PRUEBAS EN SIMULACIÓN

A continuación se presentan los resultados obtenidos al implementar y probar los algoritmos descritos anteriormente en matlab y simulink. Se emplea como proceso el sistema de tanques acoplados que se presenta en la figura 43, en donde se tiene un tanque con un perfil que posee inclinaciones para hacer que su dinámica sea no lineal [15][16].

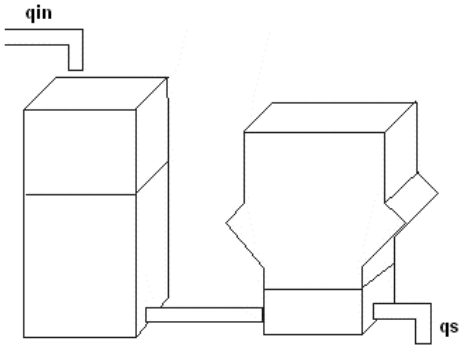

Figura 43. Sistema de tanques acoplados.

El modelo matemático del sistema está dado por las siguientes ecuaciones para el tanque 1:

$$\dot{h}_1 = \frac{1}{A_1}(q_{in} - q_2) \quad (107)$$

$$q_2 = 0.00038547(h_1 - h_2) \quad (108)$$

Donde el área de sección transversal del tanque es $A_1 = 0.01$. Para el tanque 2 se tiene:

$$\dot{h}_2 = \frac{1}{A_2}(q_2 - q_s) \quad (109)$$

$$q_s = 0.00004116\sqrt{h_2} \quad (110)$$

$$A_2 = \begin{cases} 0.01 & h_2 < 0.1 \\ 0.2h_2 & 0.1 \le h_2 < 0.2 \\ 0.12 - 0.2h_2 & 0.2 \le h_2 < 0.3 \\ 0.04 & 0.3 \le h_2 \end{cases} \quad (111)$$

En este caso q_s es no lineal pues el flujo de salida del tanque es turbulento [15] [17].

5.1. MÍNIMOS CUADRADOS.

En la figura 44 se presenta el modelo implementado en simulink para realizar la simulación del sistema en lazo cerrado sin ajuste del controlador. En este caso se toma un regulador proporcional con ganancia 0.014. El bloque de mínimos cuadrados recursivo (RLS) se implementa como una función que toma datos de entrada-salida.

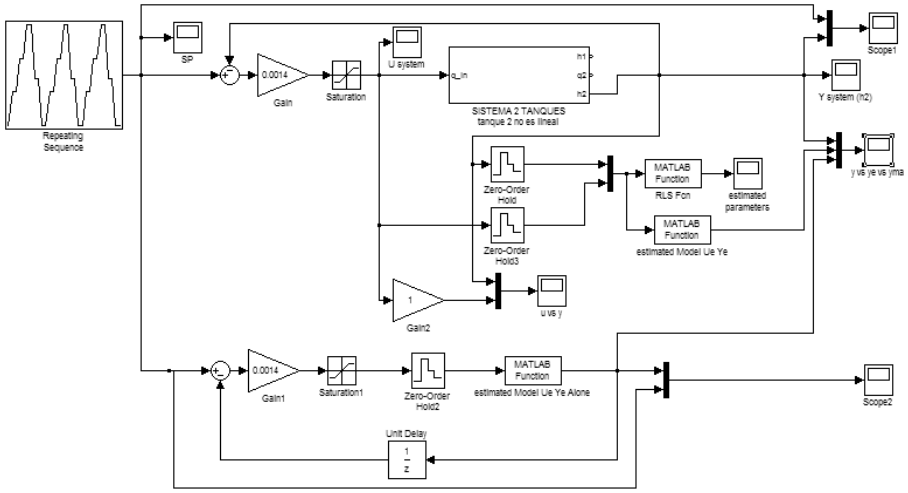

Figura 44. Modelo implementado en simulink para la prueba de mínimos cuadrados.

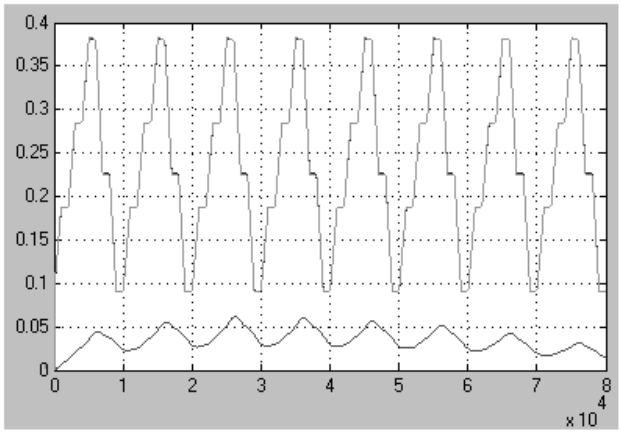

Figura 45. Resultado obtenido antes de agregar el factor multiplicador de la señal de entrada. Se tiene en azul la señal proveniente del proceso, en verde el estimativo dentro del bloque de RLS y en rojo la salida resultante del modelo en lazo cerrado.

En la figura 45 se presenta la señal de salida del proceso, la señal estimada dentro del algoritmo RLS y la señal del modelo excitado de la misma forma que el sistema real, es decir en lazo cerrado y con un controlador proporcional. Nótese que, a pesar de que la diferencia entre las señales del proceso y del estimativo dentro del bloque RLS es pequeña, el desempeño del modelo es muy pobre. De acuerdo a las pruebas realizadas, esto se debe en gran medida a la diferencia en magnitud que presenta la señal de entrada al proceso y la de salida. Por esta razón se agrega un factor en el bloque de estimación que multiplica la señal de entrada de la planta. En la figura 46 se presenta el resultado obtenido al incluir el factor multiplicador.

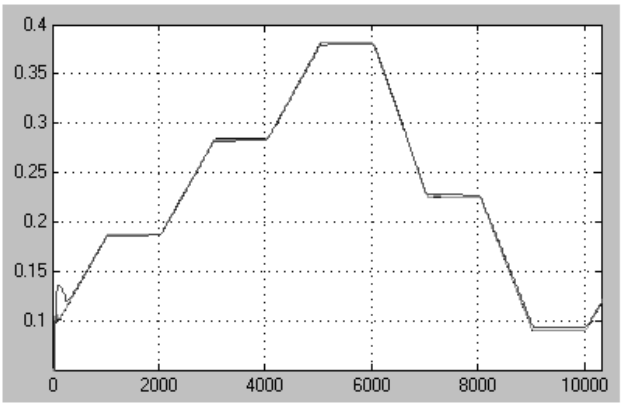

Figura 46. Resultado obtenido al agregar el factor multiplicador de 1000 a la señal de entrada. Se presenta el resultado del modelo en lazo cerrado (rojo), la señal de salida del proceso (azul) y el estimativo en el bloque RLS (verde).

En el caso de mínimos cuadrados ponderados recursivos (WRLS) se presenta una prueba en simulación empleando tres funciones tipo trapezoide sigmoideo, cuyos parámetros se presentan en la tabla 5.

Tabla 5. Parámetros de las funciones de pertenencia tipo trapezoide sigmoideo empleado para la prueba en simulación de WRLS.

Centro	x_1	d_1	x_2	d_2
1	0.5	0.25	0.6	0.25
2	0.25	0.1	0.35	0.1
3	0	0.7	0.15	0.7

El comportamiento del modelo obtenido se presenta en la figura 47, en donde se tiene la señal de salida del proceso en azul, el estimativo dentro del bloque RLS en verde y la señal de salida del modelo en lazo cerrado en rojo. Los parámetros ajustados correspondientes a cada submodelo se presentan en la figura 48.

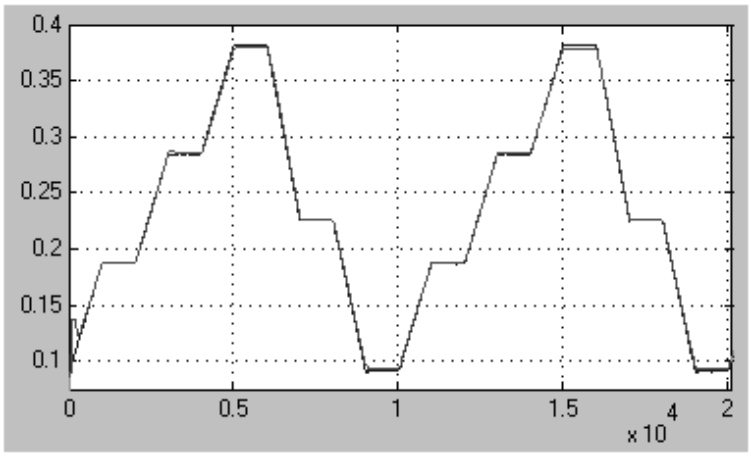

Figura 47. Resultado obtenido en simulación con el algoritmo de mínimos cuadrados ponderados recursivo con un factor multiplicador de 400 en la señal de entrada.

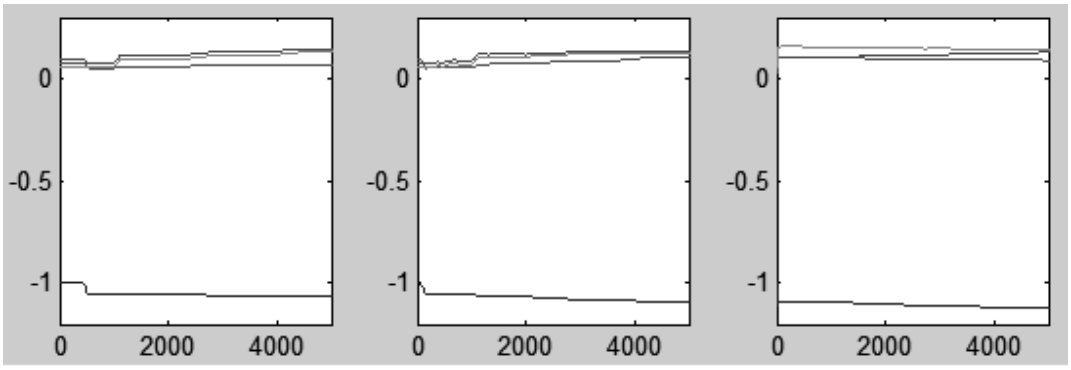

Figura 48. Comportamiento de los parámetros estimados para cada submodelo en la simulación de WRLS.

Un aspecto importante es el comportamiento dinámico del sistema cuando se tiene el controlador RST y se ajusta sus parámetros por medio del procedimiento de diseño por ubicación de polos de mínimo orden (MDPP). En la figura 49 Se contrasta el comportamiento del sistema en lazo cerrado con un controlador proporcional de ganancia

0.014 con el controlador RST, empleando el estimado obtenido con mínimos cuadrados recursivos (RLS). El modelo del comportamiento deseado se presenta a continuación.

$$\frac{B_m(q)}{A_m(q)} = \frac{0.038008q}{q^2 - 0.9708q + 0.009608} \quad (112)$$

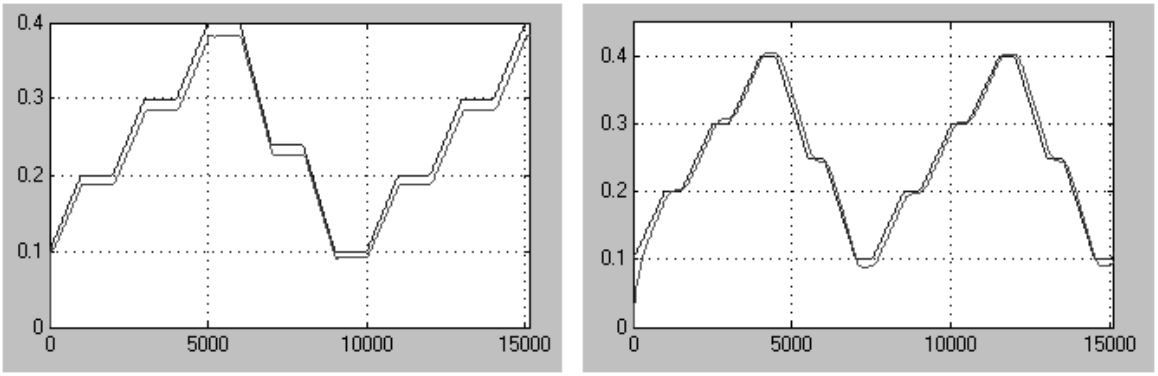

Figura 49. Comparación de la dinámica del sistema simulado con un controlador proporcional (izquierda) y uno RST adaptativo (derecha).

5.2. SISTEMA TAKAGI SUGENO EVOLUTIVO.

En el caso del sistema Takagi Sugeno evolutivo o ETS también se emplea el sistema de tanques acoplados presentado en la figura 42. El modelo implementado en simulink para la prueba de esta técnica se presenta en la figura 50. En este caso se incluyen dos ganancias para adecuar la escala de las señales de entrada y salida del proceso. Al igual que en el caso de mínimos cuadrados se tiene un bloque de función que simula el comportamiento del modelo obtenido en lazo cerrado con el mismo controlador proporcional. En este caso es necesario realimentar el valor de la salida, que corresponde al nivel del tanque 2, pues este valor está involucrado como entrada del sistema TSK del modelo.

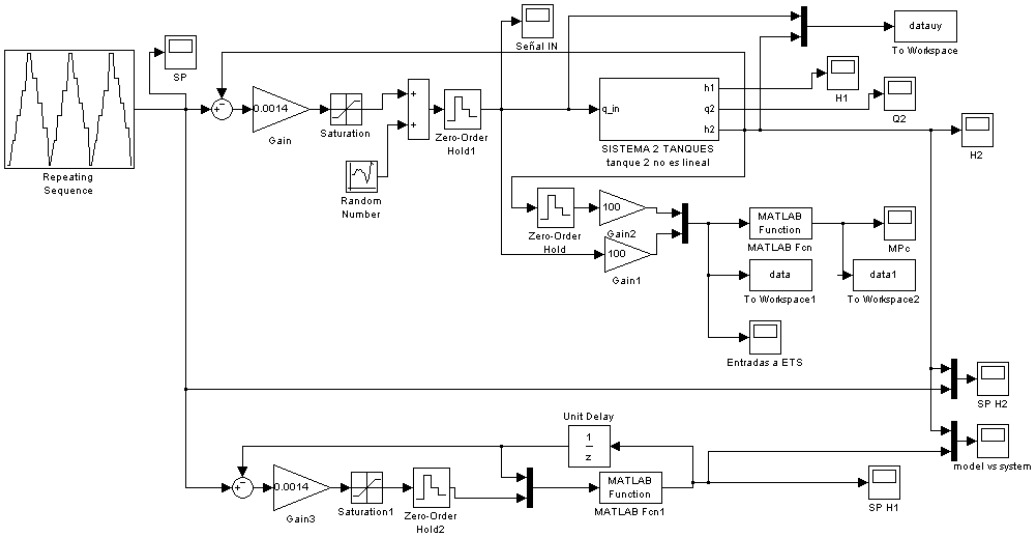

Figura 50. Sistema implementado en simulink para la prueba de modelamiento por ETS.

El resultado de simular el modelo obtenido en lazo cerrado con el controlador proporcional se presenta en la figura 51. Es evidente como en cada periodo de la señal de referencia (nivel) mejora el ajuste el modelo. El espacio de entrada-salida del proceso y los centros hallados mediante ETS se presentan en la figura 52. Además se presenta los conjuntos difusos obtenidos sobre el universo de nivel del tanque 2 en la figura 53 y el comportamiento de los parámetros estimados para cada submodelo en la figura 54. En la tabla 6 se presenta la ubicación de cada punto focal en el universo nivel tanque 2, junto con el ciclo de ejecución del algoritmo en el cual se originó y los parámetros de cada submodelo.

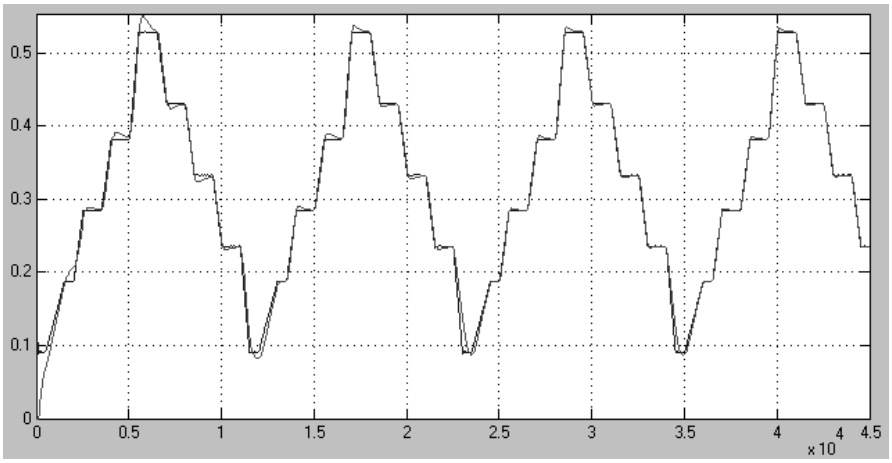

Figura 51. Resultado obtenido mediante modelamiento ETS. En azul se presenta el comportamiento de la planta y en rojo el del modelo obtenido.

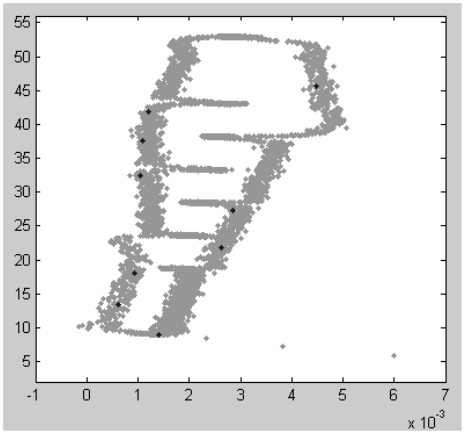

Figura 52. Espacio entrada salida del proceso y centros hallados mediante ETS.

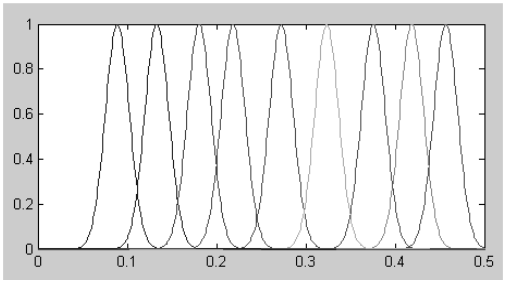

Figura 53. Conjuntos difusos hallados sobre el universo nivel tanque 2 mediante ETS.

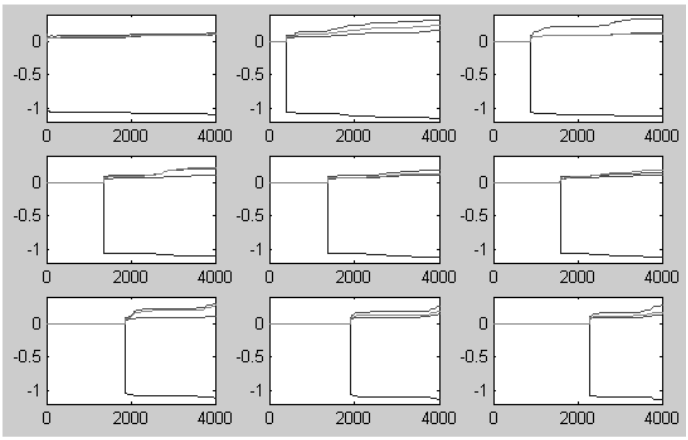

Figura 54. Evolución de los parámetros de cada submodelo lineal.

Tabla 6. Resultados de modelamiento ETS. Se presenta además los coeficientes hallados por WRLS.

Centro	Ciclo aparición	Ubicación (nivel Tk2)	Submodelo lineal			
			y(t-1)	y(t-2)	u(t-1)	u(t-2)
1	25	0.08907	-1.1210	0.1246	0.1588	0.1363
2	414	0.2725	-1.1987	0.2043	0.3683	0.2754
3	885	0.4567	-1.1516	0.1557	0.4143	0.1306
4	1354	0.41868	-1.1313	0.1357	0.2692	0.3021
5	1390	0.3757	-1.1662	0.1701	0.2453	0.1906
6	1601	0.3235	-1.1641	0.1685	0.1883	0.2798
7	1870	0.1807	-1.1386	0.1448	0.3731	0.3425
8	1898	0.1332	-1.1744	0.1795	0.3294	0.2399
9	2283	0.2186	-1.1871	0.1925	0.3878	0.2288

Para el caso del controlador adaptativo indirecto con ETS se agrega el controlador RST diseñado mediante el procedimiento MDPP en lugar del controlador proporcional. En este caso es necesario lograr unos parámetros de modelo lineal únicos para el diseño del controlador. Para tal fin se realiza el proceso de defusificación sobre los parámetros de los submodelos en lugar de realizarlo sobre el resultado de cada submodelo.

Por ejemplo, si se tienen dos submodelos de primer orden dados por:

$$\begin{aligned} y_1(t) &= a_{11}y(t-1) + b_{11}u(t-1) \\ y_2(t) &= a_{12}y(t-1) + b_{12}u(t-1) \end{aligned} \qquad (113)$$

Donde el modelo global se calcula mediante el proceso de defusificación.

$$y(t) = \frac{\mu_1 y_1(t) + \mu_2 y_2(t)}{\mu_1 \mu_2} \quad (114)$$

Desarrollando la expresión (114) y reagrupando los términos se logra un modelo equivalente, cuyos parámetros son empleados para el diseño del controlador

$$y(t) = \frac{\mu_1 a_{11} + \mu_2 a_{12}}{\mu_1 \mu_2} y(t-1) + \frac{\mu_1 b_{11} + \mu_2 b_{12}}{\mu_1 \mu_2} u(t-1) \quad (115)$$

Por otra parte es necesario garantizar que no se van a efectuar cambios bruscos en los parámetros del modelo lineal empleados para el diseño del controlador, por lo que se aplica un filtrado de los mismos, de acuerdo a la expresión:

$$\theta_d(t) = \theta_d(t-1) - 0.01\left(\theta_d(t-1) - \theta_m(t-1)\right) \quad (116)$$

Donde θ_d corresponde a los parámetros para diseño del controlador y θ_m a los parámetros del modelo hallado.

Aplicando estos conceptos se tienen los resultados presentados en la figura 55, en donde se presenta el comportamiento del sistema en lazo cerrado con ajuste del controlador, y en la figura 56, con el comportamiento de los parámetros estimados para el diseño del controlador.

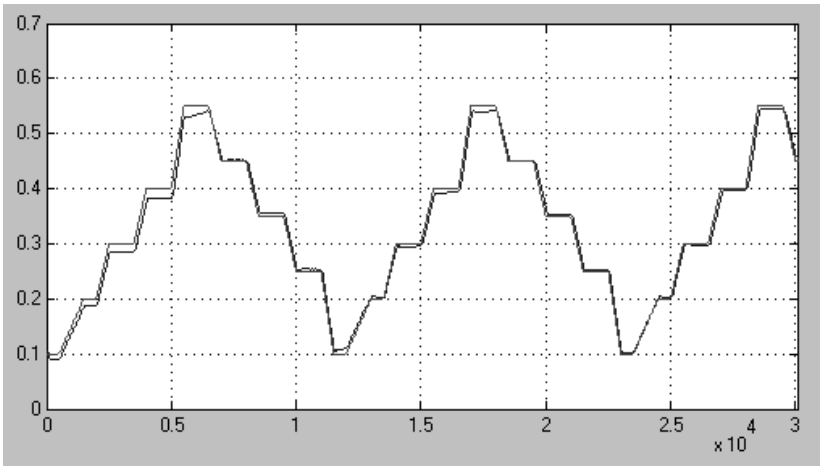

Figura 55. Comportamiento del sistema en lazo cerrado con el controlador adaptativo indirecto basado en modelamiento ETS. El valor de referencia se presenta en rojo y la salida del proceso en azul.

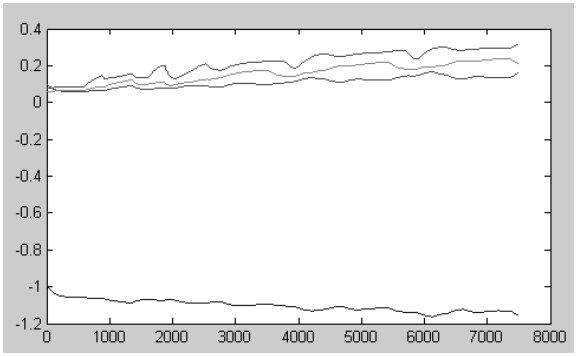

Figura 56. Evolución de los parámetros usados para el diseño del controlador por MDPP.

5.3. SISTEMA TAKAGI SUGENO EVOLUTIVO APLICANDO CLUSTERING MULTIPOTENCIAL.

Para la prueba en simulación del sistema Takagi Sugeno evolutivo multipotencial (ETSmP) se emplea el modelo implementado en simulink presentado en la figura 50, aplicado al proceso de control de nivel de dos tanques acoplados. En la figura 57 se aprecia en ajuste del modelo logrado durante 12000 segundos de simulación. Los puntos focales hallados en el espacio entrada salida se presentan en las figura 58. En el caso del modelamiento del sistema de tanques acoplados, de forma similar al ejemplo presentado con una función no lineal, se observa que los centros correspondientes a un nivel menor de 35cms son más cercanos entre si que aquellos que se encuentran sobre este nivel. Esta situación es más notoria en las graficas de las funciones de pertenencia, mostradas en la figura 59.

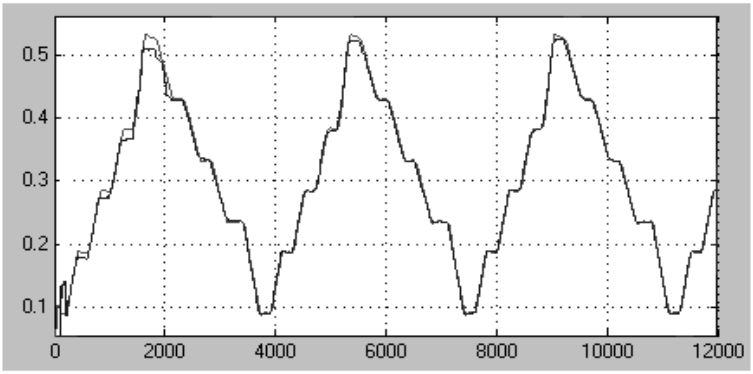

Figura 57. Resultado obtenido mediante modelamiento ETS multipotencial. En azul se presentael comportamiento de la planta y en rojo el del modelo obtenido.

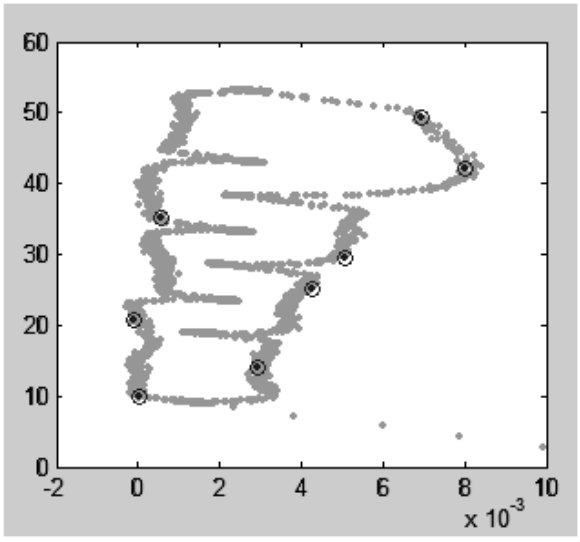

Figura 58. Espacio entrada salida del proceso y centros hallados mediante ETS multipotencial.

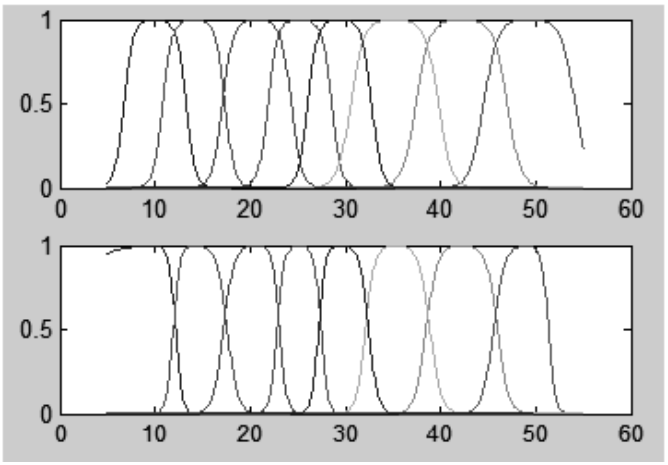

Figura 59. Conjuntos difusos hallados sobre el universo nivel tanque 2 mediante ETS multipotencial. En el recuadro superior se tienen las funciones de pertenencia del proceso ETSmP, mientras que en la parte inferior se muestran los empleados para WRLS.

Por otra parte, en esta técnica se define unas funciones de pertenencia para la parte de clustering en línea, y otras más estrechas para el proceso de mínimos cuadrados ponderados recursivos, pues para el proceso de clustering, de acuerdo a las pruebas realizadas, es deseable un mayor solapamiento entre los conjuntos difusos adyacentes. Aunque a primera vista el sistema TS evolutivo multipotencial parece más rápido de acuerdo a los ciclos de aparición de cada punto focal respecto al sistema ETS, esto depende realmente del periodo

de la señal de referencia y de la opción de generación rápida de centroide que se aplicó al ETSmP, consistente en crear un punto focal con un dato nuevo cuya pertenencia sea cero en todos los conjuntos difusos previos. En realidad la técnica multipotencial es más lenta que el ETS estándar debido a que es necesario esperar a que todas las funciones de potencial converjan. En la figura 60 se presenta el comportamiento de los parámetros de cada submodelo, mientras que el la tabla 7 se recopilan los resultados finales del procedimiento de ETSmP y WRLS.

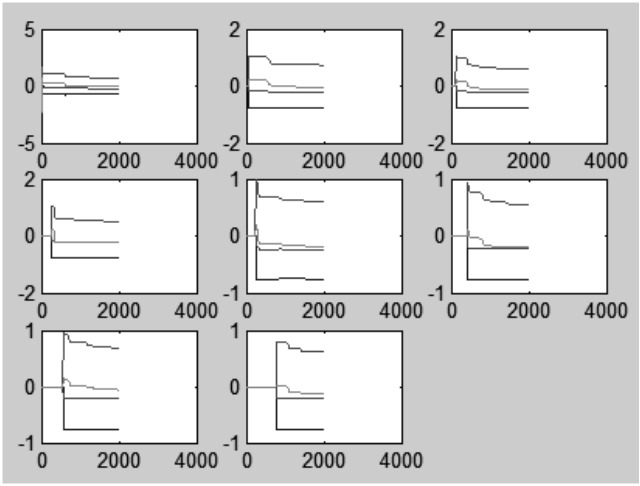

Figura 60. Evolución de los parámetros de cada submodelo lineal durante la simulación de ETSmP.

Tabla 7. Resultados de modelamiento ETSmP. Se presenta además los coeficientes hallados por WRLS.

Centro	Ciclo aparición	Ubicación (nivel Tk2)	Submodelo lineal			
			y(t-1)	y(t-2)	u(t-1)	u(t-2)
1	18	0.0992	-0.7510	-0.2090	0.7217	-0.0441
2	55	0.1408	-0.7460	-0.2259	0.7275	-0.0538
3	128	0.2521	-0.7651	-0.2184	0.6019	-0.1197
4	252	0.4221	-0.7752	-0.2178	0.5198	-0.2316
5	266	0.4933	-0.7539	-0.2370	0.6011	-0.1839
6	432	0.3517	-0.7766	-0.2140	0.5471	-0.1945
7	582	0.2070	-0.7685	-0.2085	0.6670	-0.0571
8	794	0.2941	-0.7704	-0.2169	0.6123	-0.1312

Al igual que en el caso anterior, para completar el controlador indirecto con identificador ETSmP, se incluyó el controlador RST sintonizado a través de la técnica de ubicación de

polos de grado mínimo. Para evitar un funcionamiento caótico, el controlador RST actúa sobre el proceso luego de que el identificador encuentre 4 centros, mientras que en los instantes iniciales se trabaja con el mismo controlador proporcional empleado anteriormente. En la figura 61 se aprecia el comportamiento del sistema en lazo cerrado aplicando el procedimiento descrito y en la figura 62 se tiene la evolución de los parámetros usados para el diseño del controlador. El controlador RST entra en funcionamiento alrededor de los 5000 segundos, tiempo a partir del cual se observa una aproximación progresiva del sistema en lazo cerrado al comportamiento deseado. A los 60000 segundos ya se tiene un comportamiento bastante ajustado al modelo deseado.

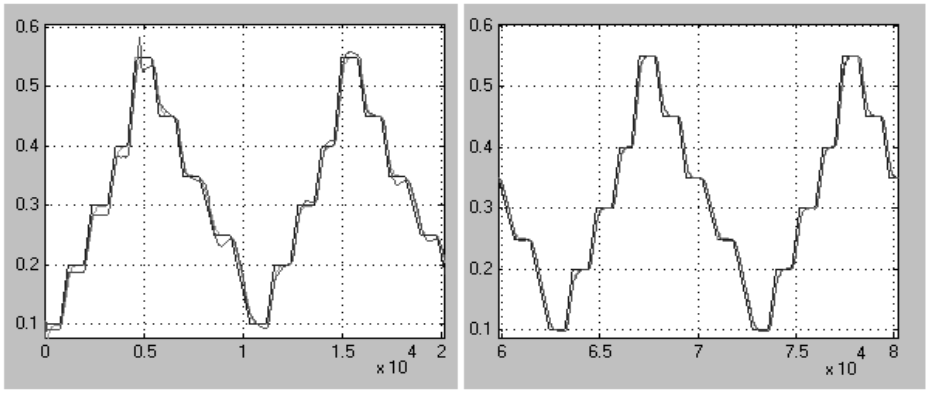

Figura 61. Comportamiento del sistema en lazo cerrado con el controlador adaptativo indirecto basado en modelamiento ETSmP. El valor de referencia se presenta en azul, el comportamiento del modelo deseado en verde y la salida del proceso en azul.

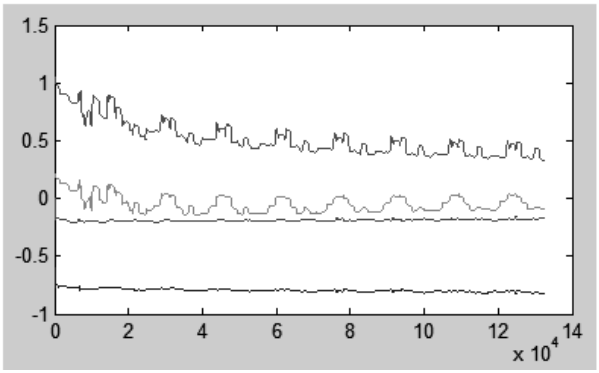

Figura 62. Evolución de los parámetros usados para el diseño del controlador por MDPP en la simulación con modelamiento basado en ETSmP.

5.4. MODELO MAMDANI.

En caso del modelo Mamdani se realizaron pruebas con plantas prototipo de segundo orden en donde se emplea un modelo de tres entradas: *y(t-1)*, *y(t-2)*, *u(t-2)*, y se definieron once conjuntos difusos en cada universo, lo que implica la evolución de 1331 reglas difusas. Como es de esperarse por la cantidad de reglas no es posible lograr que el modelo difuso Mamdani se ajuste al sistema. En las gráficas de la figura 63 se ilustra esta situación.

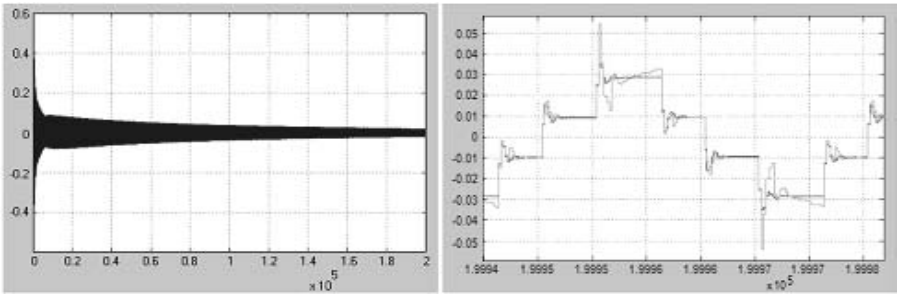

Figura 63. Evolución del error entre el modelo mamdani y el sistema prototipo (izquierda) y comparación del comportamiento del modelo logrado y del prototipo.

En contraste, al realizar la simulación con un sistema prototipo de primer orden se tiene un mejor resultado en cuanto al ajuste del modelo. En este caso se tienen dos universos de entrada *y(t-1)* y *u(t-1)* con 13 conjuntos difusos tipo lambda definidos en cada uno como se muestran en la figura 64. Sobre el universo se salida *y(t)* se ubican los conjuntos singleton. De esta manera se evoluciona un total de 169 reglas. Las pruebas con el sistema prototipo en lazo cerrado se realizaron con una entrada tipo seno y otra tipo escalonada. El resultado obtenido con la entrada tipo escalonada se presenta en la figura 65, mientras que el ajuste del modelo con entrada seno se muestra en la figura 66.

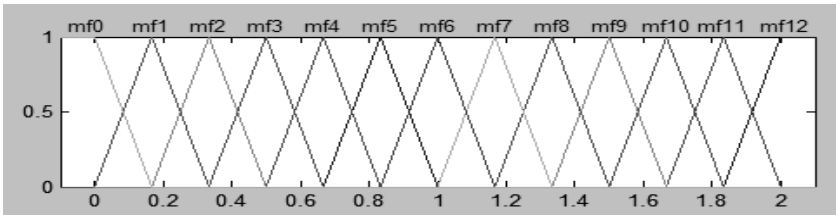

Figura 64. Funciones de pertenencia empleadas en cada universo de entrada del sistema difuso.

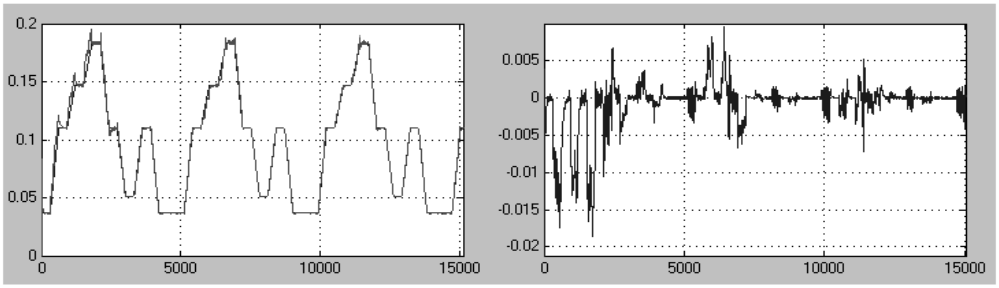

Figura 65. Aproximación de un sistema de primer orden usando modelo Mamdani. Se presenta la señal de salida obtenida y el valor del error con una señal de referencia escalonada.

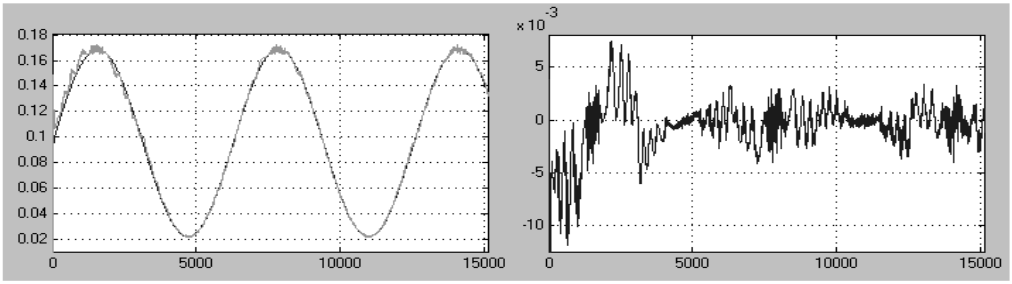

Figura 66. Aproximación de un sistema de primer orden usando modelo Mamdani. Se presenta la señal de salida obtenida y el valor del error con una señal de referencia seno.

A pesar de que la adaptación del modelo es en apariencia bastante rápida, no es posible realizar una prueba con el modelo de forma independiente debido a que no se activan todas las reglas del sistema difuso como se presenta en la tabla de conteo de activación de reglas (tabla 8).

Tabla 8. Tabla de activación de reglas obtenida en una simulación con entrada de referencia tipo seno.

	Ymf0	Ymf1	Ymf2	Ymf3	Ymf4	Ymf5	Ymf6	Ymf7	Ymf8	Ymf9	Ymf10	Ymf11	Ymf12
Umf0	2	2	0	0	0	0	0	0	0	0	0	1	1
Umf1	2	1001	1040	38	0	0	0	0	0	0	0	1	2
Umf2	0	1056	1616	599	39	0	0	0	0	0	0	2	2
Umf3	0	53	631	996	462	42	0	0	0	0	1	3	2
Umf4	0	0	55	487	811	417	38	0	0	1	2	2	1
Umf5	0	0	0	51	440	741	404	53	2	2	1	0	0
Umf6	0	0	0	0	52	417	775	465	55	1	0	0	0
Umf7	0	0	0	0	0	51	461	857	501	53	0	0	0
Umf8	0	0	0	0	0	0	51	499	983	591	56	0	0
Umf9	0	0	0	0	0	0	0	53	589	1473	996	51	0
Umf10	0	0	0	0	0	0	0	0	52	990	1405	470	0
Umf11	0	0	0	0	0	0	0	0	0	45	467	406	0
Umf12	0	0	0	0	0	0	0	0	0	0	0	0	0

Por otra parte, es necesario hallar un modelo lineal derivado del modelo difuso como se mostró en la sección introductoria con una función lineal. Para tal fin se emplean los valores centrales de los conjuntos de entrada y la ubicación de los conjuntos singleton, planteando un conjunto de máximo 8 pares entrada-salida con la regla mayor pertenencia y las reglas adyacentes, siempre que se tenga un número mínimo de activaciones, para luego hallar los valores estimados mediante la expresión:

$$\hat{\theta} = (\Phi^T \Phi)^{-1} \Phi^T Y \quad (117)$$

Donde Φ es la colección de datos de entrada dados por los centros de las funciones de pertenencia y Y incluye los valores correspondientes a la ubicación de los conjuntos singleton en el universo de salida. Aplicando este método se obtiene el estimado de los parámetros del modelo lineal presentado en la figura 67.

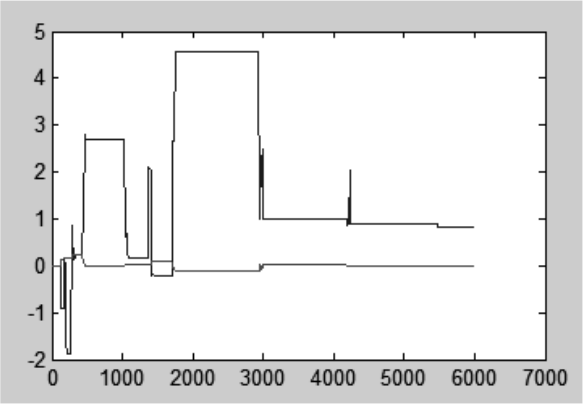

Figura 67. Estimado de los parámetros lineales obtenidos a partir del modelo difuso tipo Mamdani.

En la gráfica se aprecia que al final se encuentra un estimativo aparentemente estable, que no debe su comportamiento a una convergencia del modelo, sino a que no es frecuente que se tenga un conjunto de reglas que cumpla con las características mencionadas anteriormente, por lo que se tiene un modelo correspondiente a un solo punto de operación que no es realmente ajustado a la dinámica de la planta prototipo. De esta forma se concluye que en este caso no es viable emplear el modelo difuso tipo Mamdani como medio de identificación de un proceso para el desarrollo de un controlador adaptativo.

5.5. PROGRAMACIÓN GENÉTICA (PG).

En esta sección se presentan los resultados obtenidos al aplicar el modelamiento por programación genética para la identificación de un proceso de tanques acoplados. Es necesario tener en cuenta que, con el fin de reducir el tiempo de evolución de la solución, se restringe la estructura del árbol LISP para que cada individuo o modelo candidato siempre contenga los términos del modelo lineal en el primer nivel, Además de no repetir ninguno de estos términos lineales. Por otra parte, debido a que es evidente que el repertorio de operadores de las secuencias LISP no está orientado al manejo de funciones a trozos, se cambia el segundo tanque del modelo por uno con forma de pirámide invertida.

A diferencia de las pruebas realizadas con plantas prototipo definidas como un polinomio, presentadas anteriormente, el comportamiento con el proceso de tanques acoplados no es satisfactorio pues no hay una tendencia de reducción del error del mejor modelo candidato de cada población, ni en el acumulado de errores de cada población, evidencia de un funcionamiento pobre del algoritmo genético, como se presenta en la figura 68.

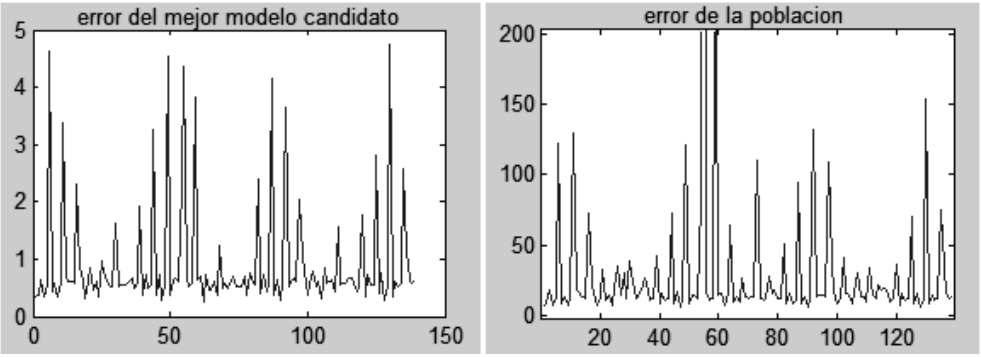

Figura 68. Comportamiento del mínimo error y de la suma de errores en cada población para la simulación de PG con el sistema de dos tanques acoplados.

Por otra parte, se realizó la prueba en simulación del modelamiento basado en programación genética con una planta de primer orden, compuesta por un tanque con forma de pirámide invertida. Aunque éste es un caso simplificado pues se trata de un sistema de primer orden, el resultado es muy similar al anterior, como puede apreciarse en la figura 69.

Esto muestra que el modelamiento por programación genética no tiene un desempeño adecuado para su aplicación en el problema de control adaptativo para el proceso seleccionado.

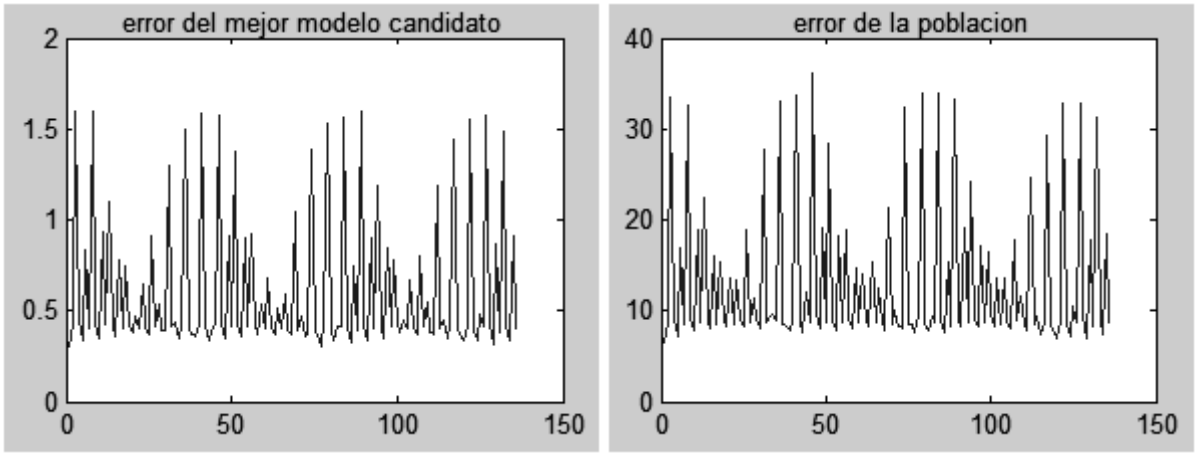

Figura 69. Comportamiento del mínimo error y de la suma de errores en cada población para la simulación de PG con el sistema de llenado de un tanque no lineal.

6. RESULTADOS EXPERIMENTALES.

Con el fin de validar los resultados obtenidos en simulación, las pruebas experimentales se realizaron sobre un montaje de tanques acoplados con el segundo tanque no lineal como se presenta en la figura 70. El caudal de entrada se suministra a través de una bomba eléctrica que toma el líquido del tanque de almacenamiento. La altura del tanque 1 y 2 son de 0.5 y 0.3 metros respectivamente. Los algoritmos del controlador adaptativo se ejecutan en Labview, empleando la interfaz NI-USB6008 como interfaz de entrada y salida de señales análogas.

Figura 70. Fotografía del sistema de tanques acoplados implementado.

A continuación se presentan los resultados experimentales obtenidos con el controlador adaptativo indirecto basado en identificación por mínimos cuadrados recursivos (RLS), por sistema evolutivo Takagi Sugeno (ETS) y sistema evolutivo Takagi Sugeno multipotencial (ETSmP). En todos los casos se emplea el método de diseño de controlador por ubicación de polos de orden mínimo con la siguiente función de transferencia como modelo deseado:

$$\frac{B_m(q)}{A_m(q)} = \frac{0.038008q}{q^2 - 0.9708q + 0.009608} \quad (118)$$

6.1. MÍNIMOS CUADRADOS.

El controlador adaptativo indirecto basado en identificación por mínimos cuadrados recursivos (RLS) se implementa con el fin de comparar sus resultados con los de los controladores basados en modelos difusos TSK. En este caso el controlador RST funciona desde el primer instante por lo que se obtiene un comportamiento que puede ser errático mientras los parámetros estimados convergen a una solución válida. En la figura 71 se presenta el comportamiento del sistema en lazo cerrado durante los dos primeros periodos de la señal de referencia. En la figura 72 se presenta la evolución de los parámetros estimados. Se aprecia que el coeficiente a_0 asociado a *y(t-2)* no converge a un valor estable sino que es decreciente, sin embargo el comportamiento en lazo cerrado del sistema permanece estable, mostrando cierta tolerancia o robustez del controlador respecto a dicho término.

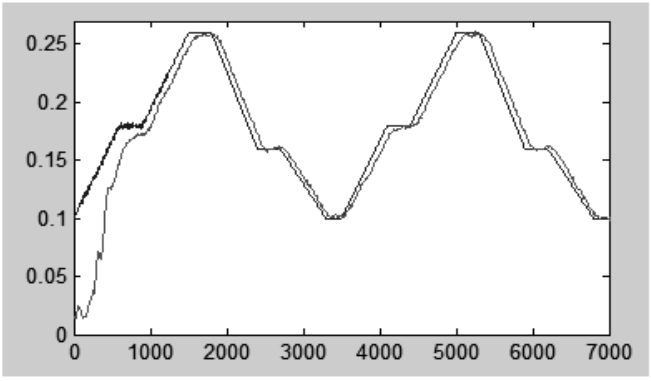

Figura 71. Dinámica resultante con el controlador adaptativo RST (RLS – MDPP).

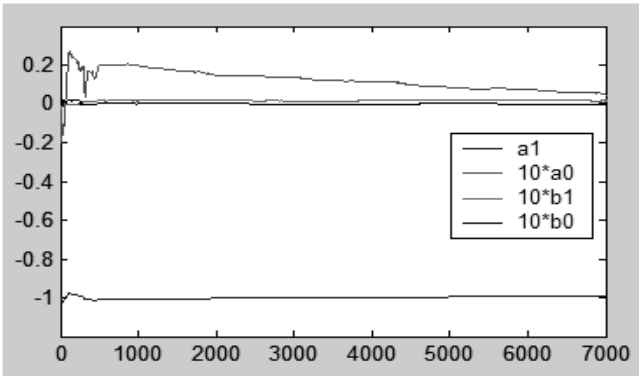

Figura 72. Comportamiento de loa parámetros estimados mediante RLS.

6.2. SISTEMA TAKAGI SUGENO EVOLUTIVO.

En el caso del sistema Takagi Sugeno Evolutivo, a pesar de que el primer punto focal se genera con el primer dato es necesario mantener el funcionamiento del sistema en lazo cerrado a través de un controlador proporcional con una ganancia de 40, debido a que la aplicación de mínimos cuadrados ponderados recursivos hace más lenta la convergencia de los parámetros estimados. El controlador RST sintonizado por MDPP inicia su funcionamiento cuando se encuentra el tercer punto focal en el proceso de clustering en línea. En la figura 73 se presenta el comportamiento del sistema en lazo cerrado con el controlador adaptativo ETS – WRLS – MDPP. Es evidente que al evolucionar distintos submodelos lineales se sacrifica la rapidez de la estimación de parámetros apreciada en RLS.

Los conjuntos difusos derivados del proceso de clustering en línea tienen la misma apertura como se presento anteriormente, aunque es notoria la mayor concentración de puntos focales a partir de 0.18 metros, como se muestra en la figura 74, situación que no es el comportamiento ideal, pues la sección no lineal den tanque 2 se encuentra de 0.1 a 0.2 metros.

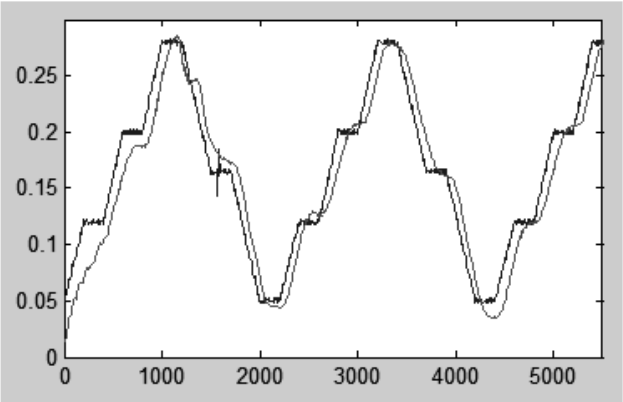

Figura 73. Dinámica resultante con el controlador adaptativo RST (WRLS – ETS – MDPP).

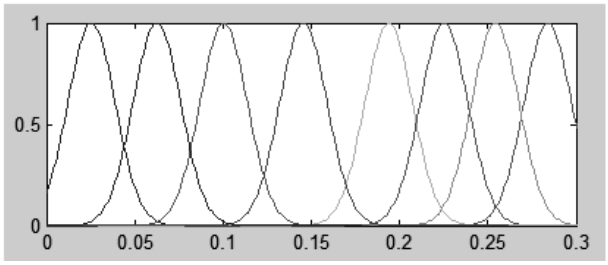

Figura 74. Funciones de pertenencia derivadas del sistema Takagi Sugeno Evolutivo.

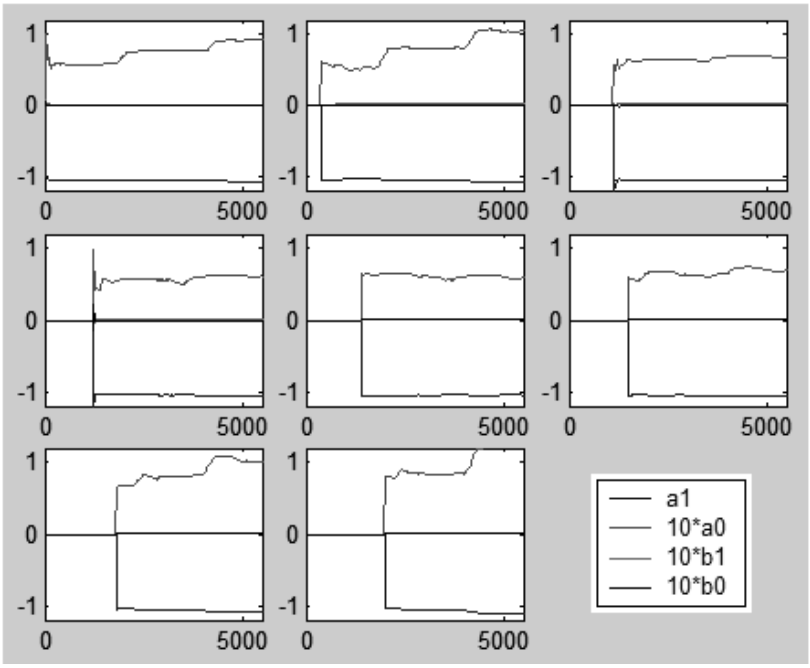

Figura 75. Comportamiento de los parámetros estimados mediante WRLS para cada regla del sistema TSK hallada por ETS.

Tabla 9. Modelo Takagi Sugeno hallado mediante ETS.

Regla	Centro FP	Submodelo lineal			
		y(t-1)	y(t-2)	u(t-1)	u(t-2)
1	0,025577	-1,0731	0,09164	0,0011164	0,0006038
2	0,10046	-1,0894	0,10472	0,0018838	5,32E-05
3	0,28404	-1,0593	0,069041	0,0026313	-0,000892
4	0,25424	-1,0513	0,061489	0,0024664	-0,000685
5	0,22526	-1,0484	0,059208	0,0014526	0,0003966
6	0,19386	-1,0566	0,068668	0,0013917	0,0005886
7	0,14555	-1,0879	0,10112	0,0016618	0,0003298
8	0,062615	-1,1102	0,12581	0,0013465	0,0004074

La evolución de los parámetros estimados para cada modelo se muestra en la figura RE6 y los puntos focales y sus submodelos correspondientes se presentan en la tabla 9. Se aprecia como la mayor variación en los parámetros ocurre en los intervalos de tiempo correspondientes a la activación de cada regla del modelo TSK. Finalmente, el comportamiento de los parámetros empleados para la sintonía del controlador RST se presenta en la figura 76.

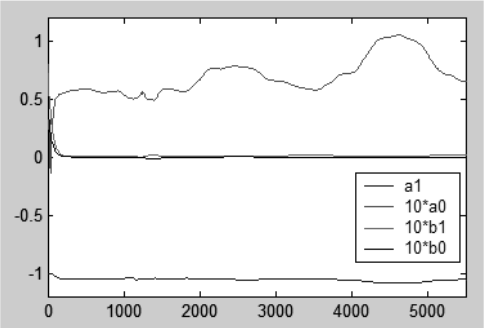

Figura 76. Comportamiento de los parámetros estimados para diseño del controlador RST.

6.3. SISTEMA TAKAGI SUGENO EVOLUTIVO APLICANDO CLUSTERING MULTIPOTENCIAL.

En el caso del sistema Takagi Sugeno Evolutivo con múltiples funciones de potencial el primer punto focal se genera cuando el cálculo de la primera función de potencial converge, por lo que es necesario incrementar el tiempo en el cual el controlador RST permanece inactivo. El controlador RST sintonizado por MDPP inicia su funcionamiento cuando se encuentra el cuarto punto focal en el proceso de clustering en línea multipotencial. En la figura 77 se presenta el comportamiento del sistema en lazo cerrado con el controlador adaptativo ETSmP – WRLS – MDPP. Es notoria la lentitud en la sintonía del controlador al comportamiento deseado. Las funciones de pertenencia derivadas del proceso de clustering en línea multipotencial presentan aperturas muy similares en los conjuntos ubicados entre 0.07 y 0.22 metros, auque en el caso del primer y el último conjunto es notoria la diferencia, como se muestra en la figura 78.

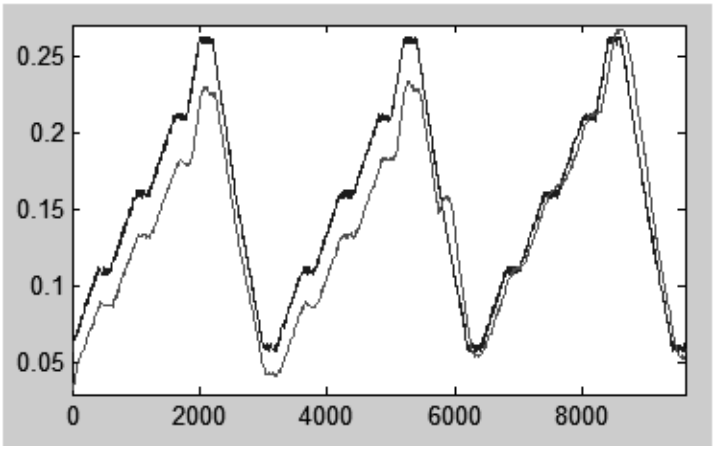

Figura 77. Dinámica resultante con el controlador adaptativo RST (WRLS – ETSmP – MDPP).

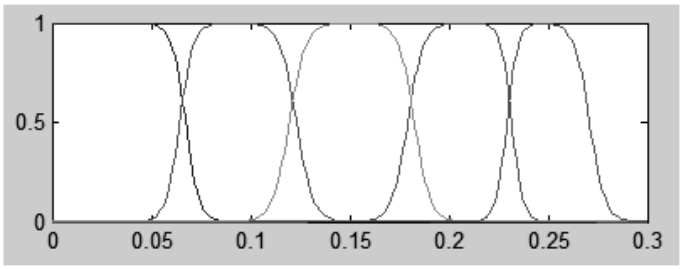

Figura 78. Funciones de pertenencia derivadas del sistema Takagi Sugeno Evolutivo.

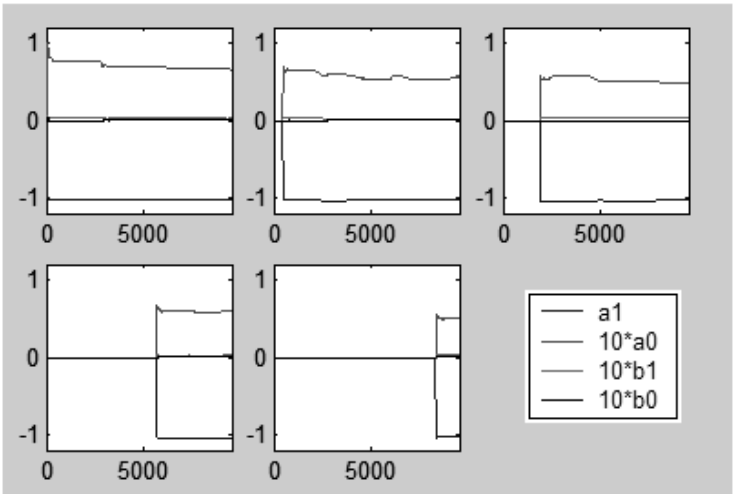

Figura 79. Comportamiento de los parámetros estimados mediante WRLS para cada regla del sistema TSK hallada por ETS.

Tabla 10. Modelo Takagi Sugeno hallado mediante ETSmP.

Regla	Centro FP	Ciclo	Submodelo lineal			
			y(t-1)	y(t-2)	u(t-1)	u(t-2)
1	0,040875	20	-1,0213	0,0651	0,0025	0,0003
2	0,090796	96	-1,0305	0,0564	0,0021	0,0002
3	0,20996	396	-1,0326	0,0485	0,003	-0,0008
4	0,15199	1147	-1,0378	0,0604	0,0024	0,0002
5	0,25102	1690	-1,0322	0,0497	0,0026	0,0003

La evolución de los parámetros estimados para cada modelo se muestra en la figura 79 y los puntos focales y sus submodelos correspondientes se presentan en la tabla 10. De igual forma al controlador basado en ETS, se aprecia que mayor variación en los parámetros estimados ocurre en los intervalos de tiempo correspondientes a la activación de cada regla del modelo TSK. El comportamiento de los parámetros empleados para la sintonía del controlador RST se presenta en la figura 80.

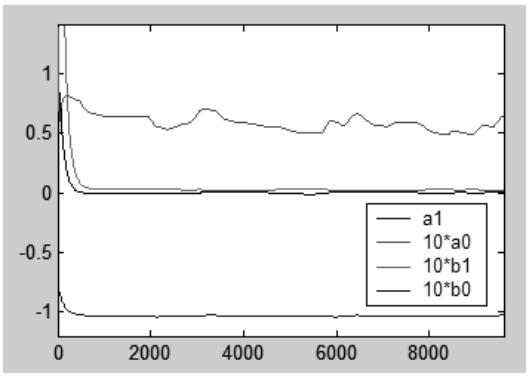

Figura 80. Comportamiento de los parámetros estimados para diseño del controlador RST.

CONCLUSIONES Y PERSPECTIVAS

En el presente trabajo se presentan algunas estrategias de modelamiento basadas en técnicas de computación flexible y control inteligente, con el objetivo de evaluar si es viable su aplicación en un controlador adaptativo indirecto. Se estudian técnicas que emplean sistemas difusos, tanto tipo Takagi Sugeno, como tipo Mamdani, y modelamiento mediante programación genética. Se busca sintonizar un controlador tipo RST para el control de un sistema de tanques acoplados con características no lineales.

Se muestra como los sistemas difusos TSK, evolucionados mediante dos formas de clustering en línea, son una opción viable para el modelamiento de un proceso con fines de control adaptativo. En el caso del modelo TSK se tiene una buena evolución de los modelos para sistemas con no linealidades a trozos, sin embargo aún no es tan acertado en los sectores donde se tienen no linealidades continuas como ocurre en las regiones con perfil inclinado del tanque no lineal. Además, en el caso de ETS se tiene un comportamiento que evoluciona de forma constructiva, pues a medida que avanza por las distintas regiones del espacio de entrada salida se generan las nuevas reglas difusas del modelo. Sin embargo, presenta una desventaja respecto al controlador implementado con RLS, ocasionada por la lentitud que se tiene para encontrar los parámetros de los submodelos lineales, causada por la reducción del tiempo que se excita cada regla.

Se presenta una variación de la técnica ETS, denominada multipotencial (ETSmP), en la cual se logra la identificación de puntos focales a través de la asignación de una función de potencial para cada centro. Esta técnica presenta ventajas al permitir conjuntos difusos con distintas aperturas pues la selección de un nuevo centro depende de varias funciones de potencial, pero hace mas lento el proceso debido a que requiere reinicializar el proceso de clustering cada vez que entra un nuevo centro, y espera a que las funciones de potencial

converjan antes de generar un nuevo punto focal. Por otra parte, encontrar las funciones de pertenencia más ajustadas a los datos es un problema abierto.

En cuanto a sistemas Mamdani evolucionados por técnicas de aprendizaje inspiradas en el controlador FMRLC (Fuzzy Model Reference Learning Control) se mostró su inviabilidad para aplicación en control adaptativo del proceso de llenado de tanques no lineales debido a la gran cantidad de parámetros que se generan, a que su ajuste es pobre y a que el modelo obtenido resulta cercano solo a los comportamientos dinámicos que se tienen en entrenamiento, así que si se migra a otro punto de operación sería como si se reiniciara el proceso sin información previa

El modelamiento mediante programación genética presento un gran poder para aproximar estructuras tipo polinómicas, pero con el sistema de tanques acoplados no resulta tan ajustado. Al respecto queda abierto un campo de estudio en el mejoramiento de esta técnica en tres aspectos: la eliminación de redundancias en las soluciones candidatas, la expansión del campo de acción al integrar otras funciones en el repertorio de operadores LISP, como raíz, exponenciales, etc, el estudio de la inclusión de parámetros dentro de las ramas de nivel mayor a 1, que constituirían parámetros no lineales y la evolución del árbol LISP por medios más constructivos y menos aleatorios que el algoritmo genético, incluyendo metaheurísticas como listas tabú. El modelamiento con programación genética se muestra más viable para aplicaciones fuera de línea o tipo supervisor en donde un experto podría evaluar la utilidad del mejor modelo candidato obtenido.

BIBLIOGRAFIA

[1] RUBIO, Francisco y LOPEZ, Manuel. Control adaptativo y robusto. España. Universidad de Sevilla. 1996. 365 p.

[2] VAN OVERSCHEE, Peter y DE MOOR, Bart. Subespace Identification for Linear Systems. Kluwer Academic publishers. Boston. 1998

[3] WANG, Li-Xin. A Course in Fuzzy Systems and Control. Prentice Hall. 1997.

[4] JANG, Jyh-Shing; SUN, Chuen; MITZUTANI, Eiji. Neuro Fuzzy and Soft Computing. Prentice Hall. 1997.

[5] ANGELOV, Plamen; FILEV, Dimitar. An Approach to Online Identification of Takagi-Sugeno Fuzzy Models. IEEE Transactions on Systems, Man and Cybernetic. Vol 34, No 1, Febrero de 2004.

[6] ANGELOV, Plamen. An Approach for Fuzzy Rule-Base Adaptation Using On-line Clustering. International Journal of Approximate Reasoning. 2004.

[7] LJUNG, Lennart. System Identification, Theory for the User. Prentice Hall PTR. 1999.

[8] ASTROM, Karl y WITTENMARK, B. Adaptive Control. Addison Wesley. 1995.

[9] PASSINO, Kevin y LAYNE, Jeffery. Fuzzy Model Reference Learning Control for Cargo Ship Steering. IEEE Control Systems Magazine. Diciembre de 1993.

[10] PASSINO, Kevin; et al. Fuzzy Learning Control for Antiskid Braking Systems. IEEE Transactions on Control System Technology. Vol 1, No 2. Junio de 1993.

[11] PASSINO, Kevin y YURKOVICH, Stephen. Fuzzy Control. Addison Wesley. 1998.

[12] ROJAS, Sergio y MARTÍNEZ, José. Introducción a al informática evolutiva. Universidad Nacional de Colombia. 1999.

[13] FLEMING, P. J; FONSECA, C. M; RODRIGUEZ, K. Identifying the Structure of Nonlinear Dynamic Systems Using Multiobjetive Genetic Programming. IEEE Transactions on Systems, Man and Cybernetics. Vol 34, No 4, Julio de 2004.

[14] GEN, Mitsuo y CHENG, Runwei. Genetic Algorithms and Engineering Desing. John Wiley & Sons. 1997.

[15] SOTO, J; GAUTHIER, A; GRISALES, V. H. Una Propuesta de Identificación por Clustering de Modelos Difusos Takagi Sugeno para Sistemas Variantes con el Tiempo. Publicado en: II Congreso Internacional del Área Andina ANDESCON, 2004. Bogotá.

[16] SOTO, J; CASTILLO, I; GAUTHIER, A; GRISALES, V. H. Identificación y Control de Sistemas no Lineales Mediante Clustering y GPC Usando Modelos Takagi-Sugeno. Revista Ingeniería, Universidad de los Andes. No 19, 2004.

[17] PALM, W. Modeling Analisys and Control of Dinamic Systems. Ed. John Wiley & Sons. 2000.

ANEXO 1. FUNCIÓN RECURSIVA CLUSTERING EN LÍNEA

```
function Vout=c0003_ETS_ANG_function(Vin);
global Vz_ka betak sigmak ka % para calculo de funcion potencial
global centros Mcentros M_Pk_centros maxcentros rac % VARIABLES DE ETS
global Mtetag MPe Ye Ue Ygr Ugr onum oden mu centros nit_rls Vdccm2 fmu % para
proceso WRLS
global HistMtetag   % para graficar despues

k_ey=100;
rbc=4*rac;       % El factor multiplicador fue alterado
betac=4/rbc^2;
betar=5.5*betac*k_ey;

% Vz_ka    % zka=z_ka(1,:) zka_ant=z_ka(2,:)
ka=ka+1;
Vz_ka(2,:)=Vz_ka(1,:);
Vz_ka(1,:)=Vin';
z_ka=Vz_ka(1,:);
z_ka_ant=Vz_ka(2,:);
% PROCESAMIENTO DE ETS ARTICULO ANGELOV
   % CALCULO RECURSIVO FUNCION POTENCIAL
   % betak= betak_anterior + zka
   betak=betak+z_ka_ant;
   % sigmak= sigmak_anterior + sum(j=1:n+1,z_ka_anterior)
   sigmak=sigmak+sum(z_ka_ant.^2);
   % vevk "ve cursiva en el paper" =sum(j=1:n+1,z_ka)
   vevk=sum(z_ka.^2);
   % vk = sum(j=1:n+1,z_ka.*betak)
   vk = sum(z_ka.*betak);
   % calculo del potencial de un dato nuevo
   Pk_ka=(ka-1)/(((ka-1)*(vevk+1))+sigmak-2*vk);

% Actualizacion del potencial de los centros previos.
   for ele=1:centros
      z_ka_ce=Mcentros(ele,:);
      Pk_ce=M_Pk_centros(ele);
      [Pk_c]=a0004_fun_cauchy_odat_v1(z_ka,z_ka_ce,ka,Pk_ce);
      M_Pk_centros(ele)=Pk_c;
   end

   % a continuacion se realiza el proceso de seleccion de centroide
   % si se esta buscando el primer centroide se emplea la tecnica del
   % "maximo inmediato"
   if (centros==0)&(ka>25)
```

```
    centros=1;
    Mcentros(centros,:)=z_ka;
    M_i_centros(centros)=ka;
    M_Pk_centros(centros)=Pk_ka;

 elseif centros>0
    % para los nuevos centros es necesario evaluar si estan cerca a los
    % centros existente (para reemplazo) o si es un nuevo centroide y
    % evaluar la relacion con el potencial del centroide mas cercano

    rbc=3.5*rac;       % El factor multiplicador fue alterado
    betac=4/rbc^2;
    % se halla la distancia de los centros al dato
    x1=z_ka(1)/k_ey;
    for b=1:centros
       z_ce=Mcentros(b,:);
       d_d_c(b)=sqrt(sum((z_ce-z_ka).^2));
       MPka_pr(b)=Pk_ka-M_Pk_centros(b)*exp(-betac*d_d_c(b));
       % valoracion de lejania por mfs
       x_ce=Mcentros(b,1)/k_ey;
       d_dc_x=((x_ce-x1)^2);
       Vmfdc(b)=exp(-betar*d_dc_x);
    end
    far=sum(Vmfdc)<0.01;
    [d_min indc]=min(d_d_c);
    % evaluacion del potencial
    if MPka_pr(indc)>M_Pk_centros(indc)|far
       % evaluacion de cercania
       near=((-M_Pk_centros(indc)/Pk_ka)+(d_min/rac))<=1;
       if near==1
          M_Pk_centros(centros)=1.0*Pk_ka;
          Mcentros(centros,:)=z_ka;
          %'near'
       elseif centros<maxcentros
          centros=centros+1;
          M_Pk_centros(centros)=1.0*Pk_ka;
          Mcentros(centros,:)=z_ka;
          M_i_centros(centros)=ka;
          ka
          %'new center'
          % inicializacion de WRLS
          tetag0(:,:)=Mtetag(:,:,1);
          Pe0=MPe(:,:,1);
          for i=1:(onum+oden-1)
             Pe0(i,i)=(1+5*(rand))*Pe0(i,i);
          end
          MPe(:,:,centros)=Pe0;
          Mtetag(:,:,centros)=tetag0;
       end
```

```
    end
  end
  Vout=Pk_ka;


% PROCESAMIENTO DE WRLS

  ye=Vin(1)/k_ey;    % tener cuidado con la escalizacion usada en simulink
  ue=fmu*(Vin(2)/k_ey);    % el factor multiplicador se emplea para acelerar convergencia
  % calculo del valor de pertenencia
  rbc=4*rac;       % El factor multiplicador fue alterado
  betac=4/rbc^2;
  x1=ye;
  betar=5.5*betac*k_ey;
  for cent=1:centros,
    x_ce=Mcentros(cent,1)/k_ey;
    d_dc_x=((x_ce-x1)^2);
    Vmfd(cent)=exp(-betar*d_dc_x);
  end

  lamda=0.98;    %forgetting factor
  phi(:,1)=[-Ye'; Ue'];
  for b=1:centros
    Pe=MPe(:,:,b);
    tetag(:,:)=Mtetag(:,:,b);
    mfd=Vmfd(b);

    K(:,1)=Pe*phi(:,1)*(lamda + mfd*phi(:,1)'*Pe*phi(:,1))^-1;
    Pe=Pe-mfd*K(:,1)*phi(:,1)'*Pe/lamda;
    tetag(:,1)=tetag(:,2)+mfd*mu*K(:,1).*(ye-phi(:,1)'*tetag(:,2));
    % corrimiento de los datos
    tetag(:,2)=tetag(:,1);

    MPe(:,:,b)=Pe;
    Mtetag(:,:,b)=tetag(:,:);
  end
  Ye(2:oden-1)=Ye(1:oden-2);     % [y(t-1) y(t-2)]
  Ye(1)=ye;
  Ue(2:onum)=Ue(1:onum-1);       % [u(t-1) u(t-2)]
  Ue(1)=ue;

  % reset periodico de matriz P
  nit_rls=nit_rls+1;
  if nit_rls>30
    for i=1:centros
      for j=1:(onum+oden-1)
        MPe(j,j,i)=(1+0.0002*(rand))*MPe(j,j,i);
      end
    end
```

```
    nit_rls=0;
end
% Registro historico
sh4=size(HistMtetag,4);
[sMt1,sMt2,sMt3]=size(Mtetag);
HistMtetag(1:4,1:2,1:sMt3,sh4+1)=Mtetag(1:4,1:2,1:sMt3);
```

ANEXO 2. FUNCIÓN RECURSIVA CLUSTERING EN LÍNEA MULTIPOTENCIAL

```
% Funcion Sistema evolutivo Takagi Sugeno Multipotencial ETSmP

function Vout=c0001_prueba_ETS_fnc(Vin)
%definicion de variables globales
global ka betak sigmak alfak vevk vk Vz_ka MPc Mcentros cent MPmin
global dim centros mxparam ka_up
global Mtetag MPe Ye Ue Ygr Ugr onum oden mu centros nit_rls Vdccm2 fmu
mxparam_rls% para proceso WRLS
%variables de prueba
global Mctest Nctest
global HistMtetag   % para graficar despues

% entrada: dato ka
% Vin[y u]
% salida:  Potenciales
Vout=zeros(1,10);

%%%%%   FASE 1. CALCULO DE POTENCIAL  %%%%%%%%%%%%%%%
% si es la primera vez que se evalua se usa expresion de nuevo dato
% se usa la expresion (k-1)/((k-1)(Vk+1)+sigk-2vk)   ---> el dato
% "nuevo" es el dato ka
Vz_ka(2,:)=Vz_ka(1,:);
Vz_ka(1,:)=Vin';
z_ka=Vz_ka(1,:);
z_ka_ant=Vz_ka(2,:);
ka=ka+1;

% inicializacion centro 0
if (centros==0)&(ka>18)
   Mcentros(1,:)=z_ka;
   MPc(1,1)=1;
   MPmin(1)=1;
   centros=centros+1;
   Pk_ka=1;
   a0008_initWRLSfornew centro(centros);
   mxparam(1:2,1:4,1)=[z_ka(1)-6,4,z_ka(1)+8,4; z_ka(2)-6,4,z_ka(2)+8,4];
   [centros ka]
   ka=2;
end
for cent=1:centros
   % calculo de vector de pertenencia
   for d=1:dim
      x1=z_ka_ant(d);
```

```
mfd(d)=fun_per_trapsig_1d(x1,mxparam(1,1,cent),mxparam(1,2,cent),mxparam(1,3,cent),
mxparam(1,4,cent));
   end
   mu_ka_ant=0.95*mfd+0.05;
   for d=1:dim
      x1=z_ka(d);

mfd(d)=fun_per_trapsig_1d(x1,mxparam(1,1,cent),mxparam(1,2,cent),mxparam(1,3,cent),
mxparam(1,4,cent));
   end
   mu_ka=0.95*mfd+0.05;
   Pk_ka(cent)=0;

      %'inicio calculo potencial'
      % betak= betak_anterior + zka
      betak(cent,:)=betak(cent,:)+(mu_ka_ant.*z_ka_ant);
      % sigmak= sigmak_anterior + sum(j=1:n+1,z_ka_anterior)
      sigmak(cent)=sigmak(cent)+sum(mu_ka_ant.*(z_ka_ant.^2));
      % alfa termino asociado a la petenencia en la expresion de ve_cursiva
      alfak(cent,1:dim)=alfak(cent,1:dim)+mu_ka_ant;
      % vevk "ve cursiva en el paper" =sum(j=1:n+1,z_ka)
      vevk(cent)=sum(alfak(cent,1:dim).*(z_ka.^2));
      % vk = sum(j=1:n+1,z_ka.*betak)
      vk(cent)=sum(z_ka.*betak(cent,:));
      % calculo del potencial de un dato nuevo
      Pk_ka(cent)=(ka-1)/((ka-1)+vevk(cent)+sigmak(cent)-2*vk(cent));
      %'calculo dato nuevo ok'

   % la llegada de un nuevo dato debe afectar el potencial de los datos
   % anteriores:
   for ele=1:centros
      z_ka_ele=Mcentros(ele,:);
      d_k_kant=z_ka-z_ka_ele;
      Pk_ele=MPc(ele,cent);
      Pk_new =(ka-1)*Pk_ele/(ka-2+Pk_ele+Pk_ele*sum((mu_ka.*(d_k_kant.^2))));
      MPc(ele,cent)=Pk_new ;
   end
   % actualiza el minimo encontrado en cada funcion de potencial
   if Pk_ka(cent)<MPmin(cent)
      MPmin(cent)=(MPmin(cent)+Pk_ka(cent))/2;
   else
      %MPmin(cent)=MPmin(cent)+0.001;
   end
   %'ajuste centros ok'
end%for cent=1:centros
if centros>0
   Vout(1:centros)=Pk_ka(1:centros);
   Vout(centros:10)=0;
```

```
else
   Vout(1:10)=0;
end
%'inicio fase 2'
%%%%%%%%   FASE 2:  EVALUACION DE POTENCIALES   %%%%%%%%%%%%%
% criterios:
% - lejania: crea nuevo centro
% - cercania & Potencial mayor: reemplaza centro existente
for cent=1:centros
   Vrpot(cent)=Pk_ka(cent)/MPc(cent,cent);  %medida "normalizada" de Potencial
end
% condicion de cercania para comparar el potencial del dato nuevom con el
% del centro mas cercano
if centros>0
   [value,cent]=max(Vrpot);
   for d=1:dim
      x1=z_ka(d);
mfd(d)=fun_per_trapsig_1d(x1,mxparam(1,1,cent),mxparam(1,2,cent),mxparam(1,3,cent),
mxparam(1,4,cent));
   end
   mu_ka=mfd;
   for i=1:centros
      z_ce=Mcentros(i,:);
      d_k_c(i)=sqrt(sum((z_ka-z_ce).^2));
   end
   d_kcmin=min(d_k_c);

   % si el datom tiene mayor potencial en un conjunto reemplaza el centro
   Pk_max=Pk_ka(cent);
   if
(Pk_max>1.05*MPc(cent,cent))&(d_k_c(cent)<0.3)&(sum(mu_ka)>0.98*dim)&(sum(MPmi
n)/centros>0.3*MPmin(cent))
      Mcentros(cent,:)=z_ka;
      MPc(cent,cent)=1.05*Pk_ka(cent);
      % ajuste de los limites de los conjuntos
      %mxparam=c0003_mxparam_sigm_creat(Mcentros,centros,dim);
      % inicializacion de la variable de ciclos sin actualizacion
      ka_up=0;
      % only for test the next tw o lines
      Nctest(cent)=Nctest(cent)+1;
      Mctest(Nctest(cent),:,cent)=z_ka;
   else
      ka_up=ka_up+1;    % Variable necesaria para saber si ya se encontraron centroides
'correctos'
   end
   % si un dato es lejano a todos los centros se crea un nuevo conjunto
   %'evaluacion lejania para agregar un conjunto'
   Vrpot2=Vrpot;
   Vrpot2(cent)=0.02*min(Vrpot2);
```

```
   [value2,cent2]=max(Vrpot2);
   if                                                                    (abs(value-
value2)<0.1)&(d_kcmin>0.5)&(max(value,value2)<0.6)&(ka_up>5)&centros<8
      'cerca a una frontera con bajos potenciales';
      new2=1;
   else
      new2=0;
   end
   % cuando un dato tiene pertenencia max a los conjuntos difusos menor a
   % 0.001 se crea centro
   x1d=z_ka(1);
   mfd1=1;
   for c01=1:centros

mfd1(c01)=fun_per_trapsig_1d(x1d,mxparam(1,1,c01),mxparam(1,2,c01),mxparam(1,3,c0
1),mxparam(1,4,c01));
      VPcent(c01)=MPc(c01,c01);
   end
   newc_fst=(max(mfd1)<0.05)&(centros>3);
   if ((((max(Vrpot)<0.28)|new2)&(ka_up>5))|newc_fst)&(d_kcmin>4)&centros<8
      % llamado a la funcion de eliminacion de centros redundantes

%[Mcentros_crr,centros_crr]=c0003_fun_per_trap1dc_delcenters(Mcentros,MPc,centros);
      %Mcentros=Mcentros_crr;
      %centros=centros_crr;
      centros=centros+1;
      %'reinicializa ets'
      betak(1:centros,1:dim)=zeros(centros,dim);
      sigmak(1:centros)=0;
      alfak(1:centros,1:dim)=zeros(centros,dim);
      vevk(1:centros)=0;
      [centros ka]
      ka=1;
      % se agrega el nuevo centro
      MPc(1:centros,1:centros)=0.5;
      for g=1:centros
         MPc(g,g)=1.2;
      end
      MPmin(1:centros)=0.2;
      Mcentros(centros,:)=z_ka;
      %'agregar parametros'
      mxparam=c0003_mxparam_sigm_creat(Mcentros,centros,dim);
      a0008_initWRLSfornewcentro(centros);
   end

end
%' fin de ciclo ETS'
centros;
rac=1;
```

```
% PROCESAMIENTO DEL ALGORITMO RLS
nit_rls=nit_rls+1;
lamda=0.90;    %forgetting factor
k_eu=100;
k_ey=100;
ye=Vin(1)/k_ey;
ue=fmu*Vin(2)/k_eu;
% calculo de valores de pertenencia
x1=ye;
neje=1;
if centros>0
   mxparam_rls=c0003_mxparam_sigm_creatRLS(Mcentros,centros,dim);
end
for cent=1:centros,
   mxparam2=[mxparam_rls(neje,1,cent)                    mxparam_rls(neje,2,cent)
mxparam_rls(neje,3,cent) mxparam_rls(neje,4,cent)]/k_ey;

Vmfd(cent)=fun_per_trapsig_1d(x1,mxparam2(1),mxparam2(2),mxparam2(3),mxparam2(4
));
end
centros;

   phi(:,1)=[-Ye'; Ue'];
   for b=1:centros
      Pe=MPe(:,:,b);
      tetag(:,:)=Mtetag(:,:,b);
      mfd=Vmfd(b);

      K(:,1)=Pe*phi(:,1)*(lamda + mfd*phi(:,1)'*Pe*phi(:,1))^-1;
      Pe=Pe-mfd*K(:,1)*phi(:,1)'*Pe/lamda;
      tetag(:,1)=tetag(:,2)+mfd*mu*K(:,1).*(ye-phi(:,1)'*tetag(:,2));
      % corrimiento de los datos
      tetag(:,2)=tetag(:,1);

      MPe(:,:,b)=Pe;
      Mtetag(:,:,b)=tetag(:,:);
   end
   Ye(2:oden-1)=Ye(1:oden-2);     % [y(t-1) y(t-2)]
   Ye(1)=ye;
   Ue(2:onum)=Ue(1:onum-1);       % [u(t-1) u(t-2)]
   Ue(1)=ue;

   % reset periodico de matriz P
   nit_rls=nit_rls+1;
   if nit_rls>30
      for i=1:centros
         for j=1:(onum+oden-1)
            MPe(j,j,i)=(1+0.0002*(rand))*MPe(j,j,i);
```

```
        end
    end
    nit_rls=0;
end
% Registro historico
sh4=size(HistMtetag,3);
[sMt1,sMt2,sMt3]=size(Mtetag);
HistMtetag(1:4,1:centros,sh4+1)=Mtetag(1:4,1,1:centros);
```

ANEXO 3. FUNCIONES MODELAMIENTO POR PROGRAMACIÓN GENÉTICA

FUNCIÓN PRINCIPAL.

```
% funcion modelamiento mediante programacion genetica
% LISP y cuantificar el error en datos de entrenamiento.
function Merror=a0002_lms_lisp_sim(Vin)
y=Vin(1);
u=Vin(2);
% Es necesario evolucionar las funciones lisp antes de evaluar su
% desempeño. Se emplea algoritmo recursivo LMS
global ARG POBLA MTETA_b MPE tc Mcteta n_indiv state indiv NewPOBLA AcuError
Ruleta HistMin HistSum HistTeta1 AcumInd EQBest
%calculo de los regresores para cada muestra
% los regresores corresponden a las subfunciones del arbol lisp
%ARG=[y(t-1) y(t-2) y(t-3) y(t-4) u(t-1) u(t-2) u(t-3) u(t-4)]
% state define en que etapa del proceso se encuentra:
%    1: evolucion de la poblacion (LMS)
%    2: evaluacion por validacion cruzada
%    3: creacion de la ruleta
%    4: seleccion y cruce
%    5: mutacion
tc=tc+1;
if tc < 120
   state=1;       % estimacion de parametros nivel uno via LMS
elseif tc < 550
   state=2;       % evaluacion de desempeño via validacion cruzada
elseif tc==551
   state=3;       % Algoritmo genetico
end

if state==1        % estimacion de parametros nivel uno via LMS
   for i=1:n_indiv
      EQ=(POBLA(i,:));
      cteta=Mcteta(i);
      Pe=MPE(1:cteta,1:cteta,i);
      tetag_b=MTETA_b(1:cteta,i);

      [out,FUNC]=a0002_eval_lisp(EQ,ARG);
      phi=FUNC';
      ye=y;
      [tetag,Pe]=a0002functRLS(ye,Pe,phi,tetag_b);
      tetag_b=tetag;
```

```
      ygorro=FUNC*tetag;
      error=ye-ygorro;
      Merror(i)=error;
      MPE(1:cteta,1:cteta,i)=Pe;
      MTETA_b(1:cteta,i)=tetag_b;
   end
   ARG(2)=ARG(1);
   ARG(1)=y;
   ARG(4)=ARG(3);
   ARG(3)=u;
   histl=size(HistTeta1,2);
   HistTeta1(:,histl+1)=MTETA_b(1:4,1);

elseif state==2    % evaluacion de desempeño via validacion cruzada
   for i=1:n_indiv
      EQ=(POBLA(i,:));
      cteta=Mcteta(i);
      tetag=MTETA_b(1:cteta,i);
      [out,FUNC]=a0002_eval_lisp(EQ,ARG);
      phi=FUNC';
      ygorro=FUNC*tetag;
      error=y-ygorro;
      Merror(i)=error;
   end
   AcuError=AcuError+abs(Merror');
   ARG(2)=ARG(1);
   ARG(1)=y;
   ARG(4)=ARG(3);
   ARG(3)=u;
elseif state==3    % creacion de la ruleta
   [Acuminval AcuminInd]=min(AcuError);
   EQBest=POBLA(AcuminInd,:)
   Merror=zeros(n_indiv,1);
   Fitness=1./AcuError;
   sumerror=sum(AcuError);
   h=length(HistSum);      % HISTORICO SUM ERROR
   HistSum(h+1)=sumerror;
   Ruleta(1)=Fitness(1)/sum(Fitness);
   for i=2:n_indiv
      Ruleta(i)=Ruleta(i-1)+Fitness(i)/sum(Fitness);
   end
   state=4;
elseif state==4    % seleccion y cruce
   Merror=zeros(n_indiv,1);
   %  step:  Seleccion
   if indiv==1        % el primer individuo de New POBLA siempre modelo lineal
      indsel=1;
      EQ1=POBLA(indsel,:);
      EQlon=length(EQ1);
```

```
      New POBLA(indiv,1:EQlon)=EQ1;
   elseif indiv==2     % el segundo individuo de New POBLA se escoge por elitismo
      [minerror,indsel] = min(AcuError);
      h=length(HistMin);
      HistMin(h+1)=minerror;  %HISTORICO MIN ERROR
      EQ1=POBLA(indsel,:);
      EQlon=length(EQ1);
      New POBLA(indiv,1:EQlon)=EQ1;
   else
      numrand=rand;
      Vect=Ruleta < numrand;
      indsel=sum(Vect)+1;
      EQ1=POBLA(indsel,:);
      numrand=rand;
      Vect=Ruleta < numrand;
      indsel=sum(Vect)+1;
      EQ2=POBLA(indsel,:);
      EQout=a0002_cruce_lisp(EQ1,EQ2);
      EQlon=length(EQout);
      New POBLA(indiv,1:EQlon)=EQout;
   end
   indiv=indiv+1;
   if indiv > n_indiv
      indiv=1;
      state=5;
   end
elseif state==5    % lanzar la nueva poblacion
   POBLA=New POBLA;
   MTETA_b=0;
   for i=1:n_indiv
      EQ=char(POBLA(i,:));
      [out,FUNC]=a0002_eval_lisp(EQ,ARG);
      cteta=length(FUNC);
      Mcteta(i)=cteta;
      MTETA_b(1:cteta,i)=zeros(cteta,1);
      Pe=20*rand(cteta);
      for v=1:cteta
         Pe(v,v)=400;
      end
      MPE(1:cteta,1:cteta,i)=Pe;
   end
   Merror=zeros(n_indiv,1);
   AcuError=zeros(n_indiv,1);
   state=1
   tc=0;
end
```

FUNCIÓN GENERACIÓN DE INDIVIDUO LISP

```
function EQ=a0002_gen_indiv_lisp(argmax,maxram)
% algoritmo para generar una funcion aleatoria en lisp modificado con suma
% en nivel 0 y permitir mas de dos ramas en nivel 1.
% funciones de la forma:
% EQ='(+(*a03a02)(*a1(S#2.000a05))(*#2.000a01)(C#4.000a03))'
% maxram: maximo numero de ramas abiertas en nivel 2 diferentes a los
% terminos lineales.
% argmax: numero de argumentos en vector ARG. Numero de variables de enrada
% involucradas en el modelo
% maxcte: valor maximo contante
%clear all
%maxram=4;
%argmax=6;
p_var=0.5;   % probabilidad de que aparezca una variable y no un operador en rama
maxcte=10;
maxniv=3;
nivel=1;     % nivel procesando. nivel 1 siempre aditivo
k=1;         %apuntador de caracteres
clear EQ
EQ(k)='('; EQ(k+1)='+';
k=k+2;
ramniv1=min(maxram+argmax,argmax+floor(maxram*rand));
nivel=1;
actRAM(nivel)=0;
for ramlin=1:argmax,
   EQ(k)='a';
   EQ(k+1)='0';
   EQ(k+2)=num2str(ramlin);
   EQ(k+3)=',';
   k=k+4;
end
actRAM(nivel)=argmax;
while actRAM(1)<=ramniv1
   actRAM(nivel)=actRAM(nivel)+1;
   rannum=rand;
   EQR=a0002_genrama_lispm(argmax,p_var,maxniv,maxcte);
   for m=1:length(EQR)
      EQ(k)=EQR(m);
      k=k+1;
   end
   if actRAM(1)<=ramniv1
      EQ(k)=',';
      k=k+1;
   end
end
```

```
EQ(k)=')';
k=k+1;
EQ(k)='f';
```

FUNCIÓN GENERACIÓN DE RAMA DENTRO DE UNA SECUECIA LISP

```
function EQR=a0002_genrama_lispm(argmax,p_var,maxniv,maxcte)
%argmax=5;
%p_var=0.5;
%maxniv=3;
%maxcte=10;
OPER='********';   %operaciones disponibles, S=seno, C=coseno
nOPER=length(OPER);
k=1;  nivel=1;
actRAM(1)=0;
while nivel>0|k==1
   if actRAM(nivel)<2
      actRAM(nivel)=actRAM(nivel)+1;
      rnum=rand;
      if ((rand<p_var)|(nivel>=maxniv))&(k>1)
         EQR(k)='a';
         k=k+1;
         num=min(argmax,1+floor(argmax*rand));
         EQAA=num2str(num);
         if length(EQAA)==1
            EQR(k)='0';
            k=k+1;
            EQR(k)=EQAA;
            k=k+1;
         else
            EQR(k)=EQAA(1);
            k=k+1;
            EQR(k)=EQAA(2);
            k=k+1;
         end
         if actRAM(nivel)<2
            EQR(k)=',';
            k=k+1;
         end
      else  %((rand<p_var)|(nivel>=maxniv))&(k>1)
         EQR(k)='(';
         k=k+1;
         if k>2
            nivel=nivel+1;
         end
         actRAM(nivel)=0;
         numOPER=min(nOPER,1+floor(nOPER*rand));
         EQo=OPER(numOPER);
```

```
        if (EQo=='C')|(EQo=='S')
            EQR(k)=EQo;
            k=k+1;
            actRAM(nivel)=actRAM(nivel)+2;
            EQR(k)='#';
            k=k+1;
            numale=maxcte*rand;
            NUMstr=num2str(numale);
            for m=1:length(NUMstr)
                EQR(k)=NUMstr(m);
                k=k+1;
            end
            EQR(k)=',';
            k=k+1;
            EQR(k)='a';
            k=k+1;
            num=min(argmax,1+floor(argmax*rand));
            EQAA=num2str(num);
            if length(EQAA)==1
                EQR(k)='0';
                k=k+1;
                EQR(k)=EQAA;
                k=k+1;
            else
                EQR(k)=EQAA(1);
                k=k+1;
                EQR(k)=EQAA(2);
                k=k+1;
            end
        else
            EQR(k)=EQo;
            k=k+1;
        end
    end
  else    %actRAM(nivel)>2
    EQR(k)=')';
    k=k+1;
    actRAM(nivel)=0;
    nivel=nivel-1;
    if nivel>0
        if actRAM(nivel)<2
            EQR(k)=',';
            k=k+1;
        end
    end
  end

end
```

FUNCIÓN DE CRUCE DE SECUENCIAS LISP

```
function EQout=a0002_cruce_lisp(EQ1,EQ2)
% funcion de cruce de dos cadenas en lenguage lisp
% se toman dos padres y se generan un hijo
%clear all
%clc
%load equat2.mat
m=1;
while EQ1(m)~='f'
   m=m+1;
end
kmax1=m;
m=1;
while EQ2(m)~='f'
   m=m+1;
end
kmax2=m;
% evaluacion si la cadena corresponde al individuo lineal
p1linflag=kmax1<=19;  % padre 1 es lineal -> no importa, verificacion saca terminos redundantes.
p2linflag=kmax2<=19;  % padre 2 es lineal -> se le agrega la rama a padre 2 (no se remplaza termino lineal)
p1=min(kmax1,max(19,4+floor((kmax1-4)*rand))); %punto aproximados de extraccion de progenitor1
p2=min(kmax2,max(19,4+floor((kmax2-4)*rand))); %punto aproximado de insercion en progenitor2
% ahora se extrae la parte correspondiente de progenitor 1
% esto puede ser constante(3), argumento(2) o rama(1);
tipo1=0;
k=p1;
k1=1;
tipoget=min(3,1+floor(3*rand));
flag_done=0;
%if p1linflag==0
   while flag_done==0
      k=k-1;
      if k==2
         k=kmax1;
      end
      k1=k1+1;
      if k1>=kmax1
         flag_done=0;
         k1=1;
         if tipoget==1
            tipoget=3;
         else
```

```
            tipoget=tipoget-1;   %si no se encontro el tipo buscado en una vuelta se cambia.
        end
        tipo1=0;
    end
    EQtest=EQ1(k);
    if EQtest=='('
        tipo1=1;
    elseif EQtest=='a'
        tipo1=2;
    elseif EQtest=='#'
        tipo1=3;
    end
    if tipo1==tipoget
        flag_done=1;
    end
end
% extraccion del objeto.
if tipoget==1       % se extrae una rama
    nivel=0;
    m=1;
    EQget(m)=EQ1(k);
    nivel=1;
    while nivel>0
        m=m+1;
        k=k+1;
        if EQ1(k)=='('
            nivel=nivel+1;
        elseif EQ1(k)==')'
            nivel=nivel-1;
        end
        EQget(m)=EQ1(k);
    end
elseif tipoget==2   % se extrae un argumento
    EQget=EQ1(k:k+2);
elseif tipoget==3   % se extrae una constante
    m=1;
    EQget(m)=EQ1(k);
    m=m+1;
    k=k+1;
    while (EQ1(k)~=')')&(EQ1(k)~=',')
        EQget(m)=EQ1(k);
        m=m+1;
        k=k+1;
    end
end
tipoget1=tipoget;
%else %if p1linflag==0
% si el padre 1 es lineal, para evitar redundancia de parametros en
% nivel 1, se inserta una rama creada:
```

```
   %p_varn=0.5;   % probabilidad de que aparezca una variable y no un operador en rama
   %maxcten=10;
   %maxnivn=2;
   %argmax=4;
   %EQget=a0002_genrama_lispm(argmax,p_varn,maxnivn,maxcten);
   %tipoget1=1;
   %end %if p1linflag==0
% se busca intervalo de insercion en progenitor 2
% progenitor 2 en EQ2
% esto puede ser constante(3), argumento(2) o rama(1);
% pero si es rama no puede ser la rama ppal
if p2linflag==0
   tipo1=0; k=p2; k1=1;
   if tipoget1==3
      tipoget=min(3,1+floor(3*rand));
   else
      tipoget=min(2,1+floor(2*rand));
   end
   flag_done=0;
   w hile flag_done==0
      k=k-1;
      if k<3
         k=kmax2;
      end
      k1=k1+1;
      if k1>=kmax2
         flag_done=0;
         k1=1;
         if tipoget==1
            tipoget=3;
         else
            tipoget=tipoget-1;   %si no se encontro el tipo buscado en una vuelta se cambia.
         end
         tipo1=0;
      end
      EQtest=EQ2(k);
      if EQtest=='('
         tipo1=1;
      elseif EQtest=='a'
         tipo1=2;
      elseif EQtest=='#'
         tipo1=3;
      end
      if tipo1==tipoget
         flag_done=1;
      end
   end
   kainit=k;
   flagavoid=0;  % bandera para evitar cambiar un termino lineal en nivel 1.
```

```
% determinacion long objeto
if tipoget==1       % se extrae una rama
   m=1;
   nivel=1;
   while nivel>0
      m=m+1;
      k=k+1;
      if EQ2(k)=='('
         nivel=nivel+1;
      elseif EQ2(k)==')'
         nivel=nivel-1;
      end
   end
   kaend=k;
elseif tipoget==2   % se extrae un argumento
   kaend=k+2;
   if kainit<18
      flagavoid=1;
   end
elseif tipoget==3   % se extrae una constante
   k=k+1;
   while (EQ2(k)~=')')&(EQ2(k)~=',')
      k=k+1;
   end
   kaend=k-1;
else
   %pause
end
% INSERCION DEL MATERIAL DE PROGENITOR 1 EN EL INTERVALO DE
PROGENITOR 2
if flagavoid==0,
   keqget=length(EQget);
   kout=1;
   k=1;
   m=1;
   while EQ2(k)~='f'
      if k<kainit
         EQout(kout)=EQ2(k);
         k=k+1;
         kout=kout+1;
      elseif (k>=kainit)&(k<=kaend)
         EQout(kout)=EQget(m);
         kout=kout+1;
         m=m+1;
         if m>keqget
            k=kaend+1;
         end
      else
         EQout(kout)=EQ2(k);
```

```
            k=k+1;
            kout=kout+1;
          end
        end
        EQout(kout)='f';
    else
        p2linflag=1;
    end
end %p2linflag==0
if p2linflag==1,
    % si el padre 2 es lineal, para evitar que se eliminen parametros
    % lineales en nivel 1, se agrega el material de padre 1 como rama nivel 1:
    keqget=length(EQget);
    if keqget<10 % <10 inidica que EQget no es una rama sini un argumento
        p_varn=0.5;     % probabilidad de que aparezca una variable y no un operador en
rama
        maxcten=10;
        maxnivn=2;
        argmax=4;
        EQget=a0002_genrama_lispm(argmax,p_varn,maxnivn,maxcten);
        keqget=length(EQget);
    end
    EQout=EQ2;
    k=kmax2-1;
    EQout(k)=',';
    for m=1:keqget
        EQout(k+m)=EQget(m);
    end
    EQout(k+m+1)=')';
    EQout(k+m+2)='f';
end %p2linflag==0
% Verificacion de terminos lineales redundantes al nivel 1
k=0; m=1; nivel=0;
while EQout(k+1)~='f'
    k=k+1;
    if EQout(k)=='('
        nivel=nivel+1;
    end
    if EQout(k)==')'
        nivel=nivel-1;
    end
    if (nivel==1)&(k>17)
        if EQout(k:k+1)==',a'
            k=k+3;
            act=0;
        else
            act=1;
        end
    else
```

```
        act=1;
    end
    if act==1
        EQcorrect(m)=EQout(k);
        m=m+1;
        act=0;
    end
end
EQcorrect(m)='f';
clear EQout;
EQout=EQcorrect;
```