IDENTIFICACIÓN EN LINEA DE SISTEMAS UTILIZANDO ALGORITMOS GENÉTICOS

JHON ALEXANDER AMARILES AROCA

UNIVERSIDAD TECNOLÓGICA DE PEREIRA
FACULTAD DE INGENIERÍA ELÉCTRICA
PEREIRA
JULIO DE 2015

IDENTIFICACIÓN EN LINEA DE SISTEMAS UTILIZANDO ALGORITMOS GENÉTICOS

Jhon Alexander Amariles Aroca

Trabajo de grado

Director

MEE. Didier Giraldo Buitrago

UNIVERSIDAD TECNOLÓGICA DE PEREIRA
FACULTAD DE INGENIERÍA ELÉCTRICA
PEREIRA
AGOSTO DE 2015

Nota de aceptación:

El presente trabajo de grado

Recibió una calificación de:

_____

_____

Jurado

_____

MEE. Didier Giraldo

Director de tesis

_____

José Germán López Quintero

Director Ingeniería Eléctrica

Pereira, 12 de agosto de 2015

TABLA DE CONTENIDO

Pág.

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

Pág.

INTRODUCCIÓN

Los algoritmos genéticos son de las denominadas técnicas metaheurísticas de optimización, las cuales son útiles cuando el problema a resolver no es fácilmente desarrollable mediante técnicas de optimización exactas. Las técnicas metaheurísticas poseen mecanismos para escapar de soluciones óptimas locales, ayudando a encontrar de mejor forma un óptimo global, teniendo en cuenta que estas técnicas no lo garantizan. [1]

Los métodos tradicionales de búsqueda utilizan características del problema para determinar el siguiente punto, diferente de los métodos de búsqueda estocásticos, en los cuales el siguiente punto se determina a partir de reglas de muestreo y decisión estocástica. Los algoritmos genéticos manipulan una población de soluciones e implementan una "supervivencia del más adaptado" como estrategia de mejores soluciones. Esto provee un paralelismo implícito y explicito que permite la explotación de varias regiones prometedoras al mismo tiempo.

Los métodos de identificación convencionales utilizan técnicas de optimización para llegar a un objetivo, el cual es lograr que el modelo obtenido con los parámetros estimados reproduzca el valor de la salida con un error mínimo. En el método de identificación de proyección, se utiliza la técnica del multiplicador de Lagrange para obtener el valor óptimo de estos parámetros, por lo que en el caso en que un sistema se pueda expresar de diferentes formas (múltiples soluciones), por este método se llegará al óptimo más cercano, pudiendo no ser el óptimo global, trayendo con esto una identificación no del todo acertada (un modelo que puede ser útil dependiendo de la aplicación).

El uso de una técnica metaheurística de optimización para la identificación en línea de sistemas podría no ser la primera opción, debido a que por lo general estas técnicas son usadas en problemas en los que el tiempo de cálculo del óptimo no es tan importante, lo cual en la identificación si lo es. Para llegar a hacer un buen uso de los algoritmos genéticos en la identificación de sistemas, se debe escoger de forma adecuada los parámetros de control, como lo son la tasa de recombinación y la tasa de mutación. También escoger el número de elementos en la población, ya que para cada problema en particular, este número debe cambiar, dependiendo de la rapidez que se desee en el algoritmo y el área de búsqueda de soluciones a explorar.

En este trabajo se presenta una metodología de solución al problema de la identificación de sistemas, implementada a través de algoritmos genéticos, el cual proporcionará soluciones factibles de buena calidad para el problema en cuestión.

En el capítulo 1 se presenta una descripción de los principales elementos del algoritmo genético y las variaciones que se le puede realizar; en el capítulo 2 se muestran métodos tradicionales de identificación y control; en el capítulo 3 se

presenta el método a usar para la implementación de la identificación con algoritmos genéticos; en el capítulo 4 se enseñan los resultados obtenidos y por último se presentan las conclusiones que se dan a partir de la aplicación del algoritmo implementado.

ALGORITMOS GENÉTICOS

1.1 Introducción a los Algoritmos Genéticos

Los algoritmos genéticos son de las denominadas técnicas metaheurísticas de optimización, las cuales son útiles cuando el problema a resolver no es fácilmente desarrollable mediante técnicas de optimización exactas. Las técnicas metaheurísticas poseen mecanismos para escapar de soluciones óptimas locales, ayudando a encontrar de mejor forma un óptimo global, teniendo en cuenta que estas técnicas no lo garantizan.

Los algoritmos genéticos se desarrollan a partir de lo observado en la naturaleza. En cada especie, los individuos con mejores cualidades y mas adaptados a su ambiente tienen más descendencia que los demás, de esta forma los rasgos hereditarios de aquellos que han tenido más éxito en la reproducción llegan a ser más numerosos en la siguiente generación. De este modo, todos los seres vivos tienen como propósito fundamental preservar y transmitir sus características a otras generaciones mediante moléculas conocidas como genes.

La genética se fundamenta en el concepto de herencia el cual fue formulado por el monje Agustino Gregor Mendel. Mendel realizó sus experimentos cruzando plantas de manera controlada, con los que formuló tres leyes con las cuales sugirió que existían factores (hoy conocidos como genes) responsables de los resultados que obtuvo. Unido al concepto de herencia, se encuentra el concepto de selección natural lo que conlleva a una competencia. Se ve la necesidad de algún tipo de variaciones en los genes, para conducir al organismo a adaptarse a los cambios de su entorno (mutación) [1].

Otro concepto importante es la diversidad de una población. Una población con diversidad genética podría tener una probabilidad de supervivencia mayor que una población homogénea, en un ambiente cambiante. En un estudio se compararon dos poblaciones de moscas *Drosophila serrata* conservadas en botellas separadas por 25 generaciones, con espacio y nutrientes limitados. Aparentemente una población genéticamente diversa progresó mejor que una población homogénea. Hoy en día esa no sería la única interpretación considerada, ya que no pueden excluirse las fluctuaciones aleatorias, pero principalmente estos resultados se dieron por la diversidad que existía [2].

Tener en una población elementos que pueden ser considerados “malos” no es algo que se quiere eliminar por completo, debido a que la coexistencia de individuos “buenos” y “malos” trae beneficios. Es posible que un individuo “malo” contenga alguna característica particular que es mejor que la del individuo “bueno”, por lo que si hay intercambio genético de estos, el resultado podría ser un descendiente mejor adaptado.

Existen algunas hipótesis acerca del comportamiento de una población, que fueron formuladas por Darwin y otras hipótesis relacionadas con la genética. Las hipótesis son las siguientes:

- El número de hijos tiende a ser mayor que el número de padres.
- El número de individuos de una especie permanece aproximadamente constante.
- De las dos anteriores se concluye que existe una lucha por la supervivencia.
- Dentro de una misma especie, los individuos presentan diferencias, y la mayoría de estas también están presentes en los respectivos padres.
- Debe existir algún proceso de variación continuada responsable de la introducción de las nuevas informaciones, contenidas en los genes de los organismos.
- No existe un límite para la sucesión de variaciones que pueden ocurrir.
- La selección natural es un mecanismo utilizado para la preservación de las nuevas informaciones que permite una mejor adaptación al medio ambiente.

El fin de todo el proceso de la selección natural es la evolución de la especie, ésta es la beneficiaria del proceso evolutivo.

El proceso de introducir la genética natural a la optimización matemática inicia con la creación de una población, la cual es un conjunto de soluciones posibles. Estas soluciones se ven calificadas por el valor que corresponde a la función objetivo que representa. Cada individuo en la población es representado por un único cromosoma, el cual se caracteriza por tener una codificación de una posible solución al problema. A la codificación se le da el nombre de genotipo (contenido genético del individuo) y a la solución que da esta codificación se le llama fenotipo (propiedades observables del individuo). Estos cromosomas se implementan por medio de vectores o listas de atributos, donde cada atributo es conocido como gen. Los valores que puede tomar este gen son conocidos como alelos.

La evolución implementada por el algoritmo genético consiste en una búsqueda de un espacio de posibles soluciones para el problema a resolver. En este proceso lo que se quiere es buscar los mejores individuos y realizar una amplia exploración del espacio de soluciones.

Este algoritmo no puede ser llamado un algoritmo completamente aleatorio, ya que contiene acciones que encaminan al algoritmo hacia los mejores individuos, como lo es el operador de selección, que en conjunto con los otros operadores aleatorios, permiten hacer una amplia búsqueda, pero siempre tendiendo a encontrar mejores soluciones. Es importante dentro del proceso, definir una función de adaptabilidad, que desempeña el papel de presión ejercida por el

medio ambiente sobre el individuo. También se debe tener en cuenta que la completa eliminación de los individuos considerados como “malos”, traerá como consecuencia la homogenización de la población. Después de que una población se ha vuelto homogénea es difícil que el método continúe evolucionando y puede conducir a una solución optima local, por lo que se deben implementar acciones para que esto no ocurra.

Después de terminar el proceso de selección se continúa con los otros operadores, recombinación y mutación (se hablara de ellos más adelante). Estos operadores generan una nueva población descendiente de la anterior. La recombinación y la mutación son de naturaleza probabilística.

Un algoritmo genético está completamente definido si ya se ha determinado: el método de generación de la población inicial, el método de selección, los parámetros de control (tamaño de la población, tasa de recombinación, tasa de mutación) y el criterio de parada.

El algoritmo genético resuelve el problema de óptimos locales, ya que al trabajar con una población, permite evaluar ampliamente el espacio de búsqueda, por lo que puede hallar una mayor cantidad de posibles soluciones.

1.2 Algoritmo Genético básico

El algoritmo genético usa una población de individuos que en los problemas combinatoriales representa un conjunto de configuraciones, para resolver un problema de optimización complejo. El algoritmo genético debe entonces hacer lo siguiente:

a. Representar adecuadamente una configuración del problema. La representación más popular es la representación en codificación binaria donde se pueden simular fácilmente los operadores genéticos de recombinación y mutación.
b. Se debe encontrar una forma adecuada para evaluar la función objetivo o su equivalente (fitness). Así, se pueden identificar las configuraciones de mejor calidad como aquellas que contienen funciones objetivo de mejor calidad (fitness value).
c. Debe existir una estrategia de selección de las configuraciones con derecho a participar en la conformación (construcción) de las configuraciones de la nueva población (nueva generación).
d. Debe existir un mecanismo que permita implementar el operador genético de recombinación.
e. Debe existir un mecanismo que permita implementar el operador genético de mutación. Mutación es generalmente considerado como un operador

genético secundario en el AG (Algoritmo Genético) pero las últimas investigaciones le están dando una importancia mucho mayor de la que se le daba inicialmente (inclusive en la genética).

f. Se debe especificar el tamaño de la población, o sea el número de configuraciones en cada generación.
g. Se debe especificar un criterio de parada del algoritmo. [1]

A continuación se explicará la forma de realizar un algoritmo genético mediante un ejemplo. El problema a optimizar será una función cuadrática y se intentará encontrar el máximo de la función, sujeto a una restricción.

Optimizar:

$$f(x) = x^2 \quad x \in \mathbb{Z}\,[0, \infty) \tag{1.1}$$

Sujeto a:

$$f(x) < 130 \tag{1.2}$$

Se debe buscar la forma de codificar el problema. Una de las más comunes y que se utilizará es la codificación binaria, se tomará como un individuo de la población un vector del tamaño que se requiera. Para este problema basta con 4 bits, teniendo como máximo número posible el 15 (1111). Es necesario tener algún conocimiento del problema para poder especificar este tipo de valores.

1.2.1 Población inicial

Es necesario desarrollar un mecanismo que genere una población inicial. En este problema se creará aleatoriamente una población de 6 individuos. Normalmente para los problemas de optimización, las poblaciones tienden a tener poblaciones más grandes, entre 20 y 80 son valores típicos, todo depende de la dificultad del problema que se desea resolver y del área de búsqueda que se quiera explorar.

Tabla 1. Población inicial Aleatoria

Individuo 1	**1**	**0**	**0**	**1**
Individuo 2	**0**	**1**	**0**	**1**

Individuo 3	**1**	**1**	**1**	**0**
Individuo 4	**0**	**1**	**1**	**0**
Individuo 5	**1**	**0**	**1**	**0**
Individuo 6	**1**	**0**	**0**	**0**

Cada fila representa un individuo de la población, con su respectiva codificación. Para conocer el valor de cada configuración, se debe hacer la conversión de binario a entero, siendo el bit de la derecha el menos significativo. Una de las características del algoritmo genético es que se utiliza para problemas en los que se necesita tomar alguna decisión, con respecto a algún peso que tenga la variable, en este problema, los pesos de cada variable es el vector de conversión de binario a entero y la decisión que se toma es si un valor es tomado en cuenta (bit con valor 1) o no es tomado en cuenta (bit con valor 0).

1.2.2 Cálculo de la función Objetivo o algún equivalente

Lo siguiente es el cálculo de la función objetivo que califique cada configuración (individuo). Es común en algunos casos utilizar un equivalente de la función objetivo, ya que de no hacer una buena selección de esto, se podría perder selectividad y el algoritmo podría no diferenciar las configuraciones "buenas" de las "malas". La función objetivo sería la evaluación de los valores de cada individuo en la función $f(x) = x^2$ y un equivalente de la función objetivo sería normalizar estos valores, dividiéndolos por el máximo, en este caso, el máximo $\overline{Z(x)}$ =225 (15^2). De esto depende en gran medida el éxito del algoritmo genético y escoger que forma de seleccionar la función objetivo está en la imaginación de la persona que lo ejecute y de lo que exija cada problema en particular. Por ejemplo otra forma de hacerla es tomar el valor de la función objetivo original y restarle algún valor. En este caso el valor que se le resta a la función objetivo original es 0.9 veces el valor de la peor configuración.

La función objetivo original y las equivalentes se muestran en la tabla 2:

Tabla 2. Cálculo de función objetivo o su equivalente

Configuración	**Función objetivo** $\boldsymbol{Z(x)}$	**Equivalente 1** $\boldsymbol{Z_1(x) = Z(x)/\overline{Z(x)}}$	**Equivalente 2** $\boldsymbol{Z_2(x) = Z(x) - K}$
Individuo 1	$\boldsymbol{9^2 = 81}$	**0.36**	**58.5**
Individuo 2	$\boldsymbol{5^2 = 25}$	**0.11**	**2.5**
Individuo 3	$\boldsymbol{14^2 = 196}$	**0.87**	**173.5**
Individuo 4	$\boldsymbol{6^2 = 36}$	**0.16**	**13.5**
Individuo 5	$\boldsymbol{10^2 = 100}$	**0.44**	**77.5**

Individuo 6	$\mathbf{8^2 = 64}$	**0.28**	**41.5**

Se puede observar que existe una configuración que está incumpliendo con la restricción (individuo 3). Para estos casos se debe implementar algún tipo de herramienta la cual opere sobre las configuraciones infactibles, por ejemplo, cuando se calcule el valor de su función objetivo, se le adicione un factor que disminuya ésta, a medida que crece la infactibilidad, esto de la siguiente forma:

$$Z_2(x) = (Z(x) - K) + \big(130 - Z(x)\big) * M \tag{1.3}$$

La constante M es un factor de castigo que es escogida por el ejecutor el algoritmo. Aplicando esta ecuación únicamente a las soluciones infactibles, el valor de la función objetivo del individuo 3 estaría dado por:

$$Z_{23}(x) = (196 - 22.5) + (130 - 196) * 2.5 = 8.5 \tag{1.4}$$

El subíndice 23 significa que es de la función objetivo equivalente 2, aplicado sobre el individuo 3, con la modificación para configuraciones infactibles y se escogió un valor M de 2.5. De esta forma aunque aun se encuentre esta infactibilidad en el problema, su valor de función objetivo es bajo y por lo tanto tendrá pocas posibilidades de tener descendientes. Es importante que sin importar que esta configuración no cumpla con lo requerido del problema, podría tener algún componente genético útil para la población en general, es por esto que no se descarta totalmente.

1.2.3 Selección

Este proceso consiste implementar un operador el cual pueda tomar los valores de la función objetivo de cada individuo y con esto, decidir quiénes son aptos para estar en cada generación.

Existen diferentes tipos selección que se pueden implementar, cada uno de ellos tiene aplicación en diferentes tipos de problema, teniendo beneficios para cada uno de ellos. Un tipo de selección común y que se verá en este ejemplo es la selección proporcional con el método de la ruleta [1]. Estos tipos de selección combinan métodos aleatorios que ayudan a darle variedad a la solución y criterios de elitismo que ayudan a escoger a los mejores individuos.

En la selección proporcional cada individuo tiene derecho a tener igual número de descendientes, dependiendo del valor de su función objetivo. Esto se hace de la siguiente forma:

$$No.\,de\ descendientes = \frac{función\ objetivo}{media\ de\ la\ función\ objetivo}$$

$$Nd_i = \frac{Z_i(x)}{Z_m(x)} \tag{1.5}$$

$$Z_m(x) = \frac{1}{n}\sum_{i=1}^{n} Z_i(x) \tag{1.6}$$

El subíndice *i* tiene como significado cada uno de los individuos de la población, *n* es el número de configuraciones y $Z_m(x)$ es el valor medio de las funciones de adaptación de las *n* configuraciones de la población.

A continuación se mostrará el valor del número de descendientes por configuración con la función objetivo original y la segunda función objetivo equivalente mostradas anteriormente. Se debe tener en cuenta que el valor media de la función objetivo original es $Z_m(x) = 83.66$ y de su equivalente 2 es $Z_{m2}(x) = 33.66$. Con esto se tienen los siguientes valores que se muestran en la tabla 3:

Tabla 3. No de descendientes por configuración

Configuración	Original	Equivalente 2
1	**0.97**	**1.74**
2	**0.30**	**0.08**
3	**2.34**	**0.25**
4	**0.43**	**0.40**
5	**1.19**	**2.30**
6	**0.77**	**1.23**
Total	**6.00**	**6.00**

Se puede observar que para el caso de la función objetivo original, el valor más alto de número de descendientes es el de la configuración infactible, por lo que no se puede ver como una buena forma de calcular la función objetivo. Para la función objetivo equivalente 2 se puede ver que es un poco más selectiva y además que el valor de la función objetivo para la configuración infactible es bajo, que era lo que se quería conseguir.

Del número de descendientes obtenido anteriormente no se puede aun concluir nada, ya que no se puede tener un número no entero de configuraciones. Para solucionar este problema se usa el método de la ruleta la cual es una de las componentes aleatorias que tiene el algoritmo genético. En el esquema de la ruleta se le asigna a cada configuración una porción de una ruleta, dependiendo del valor de número de hijos calculado anteriormente. La ecuación que relaciona la porción de la ruleta y el número de hijos es la siguiente:

$$Porción\ de\ Ruleta = 360°(\frac{1}{n})Nd_i \tag{1.7}$$

Con esta relación se obtiene la región de la ruleta de cada individuo:

Individuo 1 $\rightarrow \frac{1.74}{6}360 = 104.4°$

Individuo 2$\rightarrow \frac{0.08}{6}360 = 4.8°$

Individuo 3$\rightarrow \frac{0.25}{6}360 = 15°$

Individuo 4$\rightarrow \frac{0.40}{6}360 = 24°$

Individuo 5$\rightarrow \frac{2.30}{6}360 = 138°$

Individuo 6$\rightarrow \frac{1.23}{6}360 = 73.8°$

El paso a seguir después de tener la ruleta creada, es generar números aleatorios de 0 a 360, ese sería el equivalente a hacer girar la ruleta y se debe hacer un número de veces igual al número de individuos en la población. Este número corresponde a una región de la ruleta, la cual equivale a una configuración, significando que esta configuración tendrá un descendiente. Se puede observar que de cierto modo, aunque es un proceso aleatorio, las configuraciones con mejor función objetivo tienen más probabilidad de tener descendientes.

Si se quiere tener una mejor forma de representar este proceso para un algoritmo en computador, se dividen los valores anteriores por 360 y así se tendrán valores normalizados. Con esto se puede trabajar ya no con una ruleta, sino con un vector que va de 0 a 1 y se repetiría lo mismo que con la ruleta original, pero generando números aleatorios de 0 a 1. La representación de la ruleta como vector se muestra a continuación:

Ruleta=

0.29
0.30
0.34
0.41
0.795
1

Siendo el rango desde 0 a 0.29 para el individuo 1, el rango desde 0.29 hasta 0.34 para el individuo 2 y así sucesivamente. Haciendo girar la ruleta 6 veces se obtienen los resultados mostrados en la tabla 4:

Tabla 4. Selección aleatoria de individuos

No aleatorio generado [0,1]	Configuración seleccionada
0.17	**1**

0.39	**4**
0.28	**1**
0.93	**6**
0.65	**5**
0.39	**4**

Por lo tanto, el proceso de selección termina, eligiendo a las siguientes configuraciones (Tabla 5):

Tabla 5. No de descendientes por individuo (valores enteros)

Configuración	No. de descendientes
1	**2**
4	**2**
5	**1**
6	**1**

Existen algunos problemas con este tipo de selección, como el que no pueden haber valores de funciones objetivo negativas, ya que la expresión para el número de descendientes sería negativo, teniendo problemas para representar la ruleta. Para solucionar este problema, se le puede sumar a la función objetivo una constante para eliminar los valores negativos. La otra alternativa sería utilizar otro método de selección, que no involucre este tipo de inconvenientes, como por ejemplo la selección por torneo que más adelante se explicará.

1.2.4 Recombinación

En este proceso son sometidas las configuraciones resultantes del proceso de selección. Este procedimiento consiste en tomar dos vectores aleatoriamente, intercambiar entre los vectores una porción de ellos y así generar dos nuevos vectores con información combinada. Esta recombinación se puede hacer de distintas formas, escogiendo uno o más puntos de recombinación. Para el ejemplo que se lleva a cabo, se utilizará recombinación de punto simple, en el cual solo se escoge un punto de cruce de las dos configuraciones seleccionadas, generando un número aleatorio entre 1 y (P-1) siendo P el número de bits que contiene cada configuración. Los bits de la primer configuración que van desde (P-1) hasta el final del vector pasan a ser parte de la otra configuración y viceversa. El procedimiento se muestra a continuación:

Se escogen dos configuraciones aleatoriamente, por ejemplo el individuo 1 y el individuo 5. Para el ejemplo solo se mostrará el procedimiento con una pareja,

pero en el algoritmo se debe hacer para todas o para un porcentaje de todas, normalmente este valor es $p_c = 0.9$ (tasa de recombinación). Para decir si a un par de configuraciones se les realizará recombinación, se genera un número aleatorio de 0 a 1 y si este número es menor que el valor de la tasa de recombinación, la recombinación se efectúa sobre el par de configuraciones. El valor de la constante P elegido aleatoriamente es el número 3.

Punto de recombinación ⇩

					F.O.
Individuo 1	**1**	**0**	**0**	**1**	**81**
Individuo 5	**1**	**0**	**1**	**0**	**100**
Hijo 1	**1**	**0**	**0**	**0**	**64**
Hijo 2	**1**	**0**	**1**	**1**	**121**

Fig. 1 Recombinación de dos configuraciones

Se obtienen dos nuevos hijos, uno de ellos con una función objetivo de mejor valor y sin incumplir la restricción, por lo que con ayuda de este procedimiento se ha mejorado la población.

1.2.5 Mutación

Este es el último operador del algoritmo genético básico, el cual consiste en cambiar aleatoriamente un bit de 0 a 1 o viceversa. Este operador en un principio fue considerado como un operador secundario, pero esto ha cambiado desde el punto de vista de problemas de gran tamaño y cuando el ambiente es dinámico.

La tasa de mutación p_m enseña la probabilidad que tiene un bit de ser cambiado de su valor actual. Se debe ir bit por bit intentando que se realice la mutación, así ésta es independiente de otras mutaciones en otros bits de una configuración. Aquí la forma de realizar esto es generando un número aleatorio y si éste es menor a la tasa de recombinación, el bit es modificado. Se puede observar que se produce otro tipo de decisión aleatoria y con esto se ayuda a mejorar la diversidad de la población. Para este ejemplo se supone una tasa de mutación $p_m = 0.1$. Este valor depende del tamaño de las configuraciones y de que tanto se quiere generar estos cambios. Por probabilidad se podría decir que uno de cada diez bits podría cambiar su valor. Después de aplicar este operador la población quedaría de la siguiente forma, suponiendo que aun no se ha efectuado la recombinación, o sea, trabajando con la población con que se inició el ejemplo. Se señaliza los bits en que se hizo la mutación con una flecha (Fig. 2):

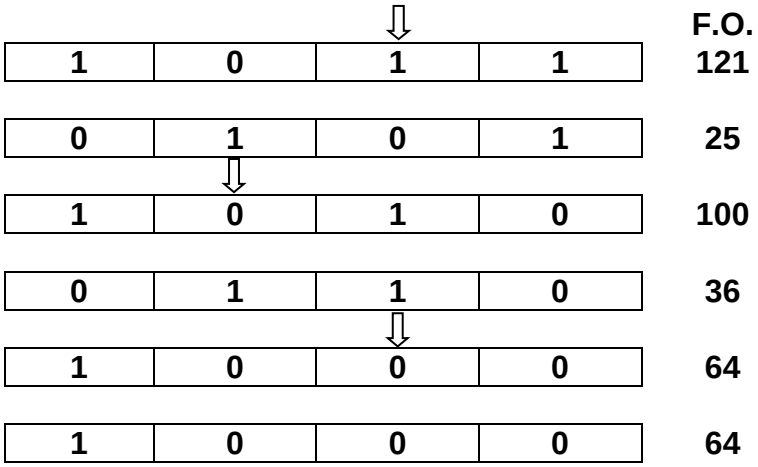

Fig. 2 Mutación aplicada a la población

Siendo el valor de función objetivo F.O. evaluar los valores ya convertidos a enteros en la función $f(x) = x^2$. Se observa que para la primera configuración hubo un cambio que mejoró el valor de la función objetivo y para el caso de la tercera configuración, se eliminó la infactibilidad.

Cada ciclo generacional es el conjunto de los operadores selección, recombinación y mutación, con los cuales se obtienen nuevos individuos para la siguiente generación.

1.2.6 Criterio de parada

Para finalizar el algoritmo, se debe dar un criterio de parada, el cual se puede hacer de diferentes formas. Una forma de hacerlo es especificar el número de generaciones que se quiera realizar. Otra forma es que la incumbente (configuración con mejor función objetivo) alcance un valor de calidad especificado. Por último, cuando la población sea muy homogénea y no exista más evolución se detiene el algoritmo.

1.3 Algoritmo genético de Chu-Beasley

Este algoritmo está basado en el algoritmo genético convencional, pero tiene algunas diferencias que mejora el proceso para problemas de gran tamaño. A continuación se muestran las características más importantes:

-Lleva los valores de infactibilidad y función objetivo separados para realizar diferentes acciones con cada uno de ellos, eso quiere decir que este algoritmo permite trabajar con soluciones infactibles.

- En este algoritmo solo se sustituye un individuo a la vez en cada ciclo generacional, a diferencia del algoritmo genético básico que trabaja simultáneamente con toda la población.

-El nuevo individuo puede ingresar a la población si es de mejor calidad que el padre.

-Todos los individuos de la población deben ser diversos entre ellos, esto evita la prematura convergencia del algoritmo, que es uno de los mayores problemas que tiene. La diversidad se define como que tantos bits de diferencia tienen una configuración con otra. Un nuevo individuo debe ser diverso con respecto a los existentes, si no es así, este individuo no puede hacer parte de la población.

-Existe un criterio de aspiración, el cual consiste en que un individuo que no sea diverso con respecto a los demás puede ingresar a la población únicamente si es de mejor calidad que la incumbente. En este caso todos los individuos no diversos con respecto al nuevo serán eliminados de la población y serán remplazados en las próximas generaciones, esto quiere decir que la población se reduce temporalmente.

Para cada problema que se quiere resolver con los algoritmos genéticos, se debe ajustar todos sus parámetros y hacer modificaciones de los operadores de tal forma que el algoritmo se ajuste al problema. Más adelante se tomará el algoritmo genético básico y se le añadirán características del algoritmo genético de Chu-Beasley, esperando aprovechar con esto todas las ventajas que tiene cada uno de estos métodos.

2. MÉTODOS DE IDENTIFICACIÓN Y CONTROL DE SISTEMAS EN TIEMPO DISCRETO

2.1 Métodos de identificación de sistemas

La identificación de sistemas por medio de algoritmos por computador se convierte en una herramienta muy útil, debido a que existen sistemas que tienen algunas dificultades en cuanto a la forma de obtener la descripción matemática, debido a los elementos que posee o porque no se tiene acceso al interior del sistema.

La identificación se basa en que a partir de datos obtenidos de las respuestas y entradas del sistema, se hace una aproximación de las ecuaciones que rigen dicho sistema, con las cuales se puede describir el funcionamiento de éste.

Los métodos de identificación se pueden aplicar de dos formas, on-line (en línea) y off-line (fuera de línea) los cuales tienen usos diferentes dependiendo de la aplicación. Los métodos de identificación off-line realizan la identificación con valores obtenidos en un tiempo definido, por lo que se debe tener seguridad de que el sistema no va a cambiar o que trabajará cerca de un punto de operación. Por otro lado, los métodos de identificación on-line operan en todo instante de tiempo o en intervalos definidos por el ejecutor de dicho método, tratando de identificar cualquier cambio que pueda ocurrir en el sistema [3]. Con este último método de identificación, se pueden realizar controles adaptativos, los cuales se ajustan a los cambios que se presenten en el sistema.

Aquí se explicará únicamente un método de identificación on-line, ya que para el problema que se quiere resolver, se requiere de una identificación que pueda captar los cambios que ocurran en el sistema.

2.1.1 Identificación de sistemas con el método de proyección

Sea un sistema cuyo modelo digital es de la forma:

$$A(q^{-1})y[k] = B(q^{-1})u[k] \tag{2.1}$$

$$A(q^{-1}) = 1 + a_1 q^{-1} + a_2 q^{-2} + \cdots + a_n q^{-n}$$

$$B(q^{-1}) = b_o q^{-1} + b_1 q^{-2} + \cdots + b_{n-1}$$

k: Variable de tiempo discreto

y[k]: Salida del sistema

u[k]:Entrada del sistema

q: Operador de desplazamiento

Con función de transferencia:

$$\frac{Y(Z)}{U(Z)} = \frac{b_o Z^{n-1} + b_1 Z^{n-2} + \cdots + b_{n-1}}{Z^n + a_1 Z^{n-1} + \cdots + a_n} \tag{2.2}$$

Z: Variable de la transformada Z

La variable q (operador de desplazamiento) es el operador análogo al operador diferencial, usado para manipular ecuaciones diferenciales lineales con coeficientes constantes. [4]

Cuando se está en el dominio de Z, las variables se usan en letras mayúsculas. Para el dominio del tiempo discreto, las variables se representan con letras minúsculas.

Es posible escribir la salida actual en función de las entradas y salidas anteriores:

$$y[k] = -\sum_{j=1}^{n} a_j y[k-j] + \sum_{j=0}^{n-1} b_j u[k-1-j] \tag{2.3}$$

Factorizando los coeficientes se tienen:

$$y[k] = [a_1\ a_2 \ldots\ a_n\ b_0\ b_1 \ldots\ b_{n-1}] \begin{bmatrix} -y[k-1] \\ -y[k-2] \\ -y[k-3] \\ \ldots \\ -y[k-n] \\ u[k-1] \\ u[k-2] \\ \ldots \\ u[k-n] \end{bmatrix} \tag{2.4}$$

$$\theta^T[k] = [a_1\ a_2 \ldots\ a_n\ b_0\ b_1 \ldots\ b_{n-1}]$$

$$\Phi[k-1] = \begin{bmatrix} -y[k-1] \\ -y[k-2] \\ -y[k-3] \\ \dots \\ -y[k-n] \\ u[k-1] \\ u[k-2] \\ \dots \\ u[k-n] \end{bmatrix}$$

Objetivo:

$$\frac{1}{2}\left|\left|\theta[k] - \theta[k-1]\right|\right|^2 = 0 \tag{2.5}$$

Minimizar sujeto a una restricción:

$$y[k] - \theta^T[k]\Phi[k-1] = 0 \tag{2.6}$$

Se plantea la función de costo:

$$J = \frac{1}{2}\left|\left|\theta[k] - \theta[k-1]\right|\right|^2 + \lambda(y[k] - \theta^T[k]\Phi[k-1]) \tag{2.7}$$

Se define:

$$\left|\left|a - b\right|\right| = a^T a - a^T b - b^T a + b^T b$$

Con a y b vectores columna.

$$J = \frac{1}{2}(\theta^T[k]\theta[k] - 2\theta^T[k]\theta[k-1] + \theta^T[k-1]\theta[k-1]) + \lambda(y[k] - \theta^T[k]\Phi[k-1])$$

$$\frac{dJ}{d\theta[k]} = \frac{1}{2}(2\theta[k] - 2\theta[k-1]) + \lambda(-\Phi[k-1]) = 0 \tag{2.8}$$

$$\theta[k] - \theta[k-1] - \lambda\,\Phi[k-1] = 0$$

$$\frac{dJ}{d\lambda} = \ y[k] - \theta^T[k]\Phi[k-1] = 0 \tag{2.9}$$

Se multiplica por $\Phi^T[k-1]$:

$$\Phi^T[k-1]\theta[k] - \Phi^T[k-1]\theta[k-1] - \lambda\Phi^T[k-1]\,\Phi[k-1] = 0$$

Despejando λ se tiene:

$$\lambda = \frac{\Phi^T[k-1]\theta[k] - \Phi^T[k-1]\theta[k-1]}{\Phi^T[k-1]\,\Phi[k-1]}$$

$$\lambda = \frac{y[k] - \Phi^T[k-1]\theta[k-1]}{\Phi^T[k-1]\,\Phi[k-1]} \tag{2.10}$$

Despejando $\theta[k]$ y reemplazando λ, se tiene:

$$\theta[k] = \theta[k-1] + \frac{(y[k] - \Phi^T[k-1]\theta[k-1])}{\Phi^T[k-1]\,\Phi[k-1]}\Phi[k-1] \tag{2.11}$$

Para obtener la estimación de $\theta[k]$ se tiene:

$$\theta[k] = \theta[k-1] + \frac{(y[k] - \Phi^T[k-1]\theta[k-1])}{c + \Phi^T[k-1]\,\Phi[k-1]}\Phi[k-1] \tag{2.12}$$

Con c = 0 si:

$$\Phi^T[k-1]\,\Phi[k-1] \neq 0$$

Con c = ∞ si:

$$\Phi^T[k-1]\,\Phi[k-1] = 0$$

2.1 Métodos de control

El control de sistemas se puede hacer por medio de controladores de reubicación de polos y por controladores en espacio de estados. A continuación se mostrará un método de control por reubicación de polos.

Se debe tener en cuenta que el controlador que se mostrará a continuación, es un controlador en tiempo discreto y que por medio de un método de identificación como el mostrado anteriormente, se puede construir un controlador adaptativo. El control adaptativo se explicará más adelante.

2.1.1 Control por reubicación de polos

Considere un sistema de la forma que se muestra en la figura 3:

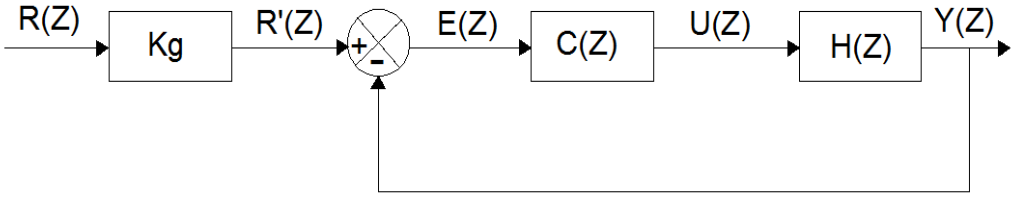

Fig. 3 Diagrama de bloques del controlador por reubicación de polos

Donde $H(Z)$ es la función de transferencia definida como:

$$H(Z) = \frac{b_0 Z^{n-1} + b_1 Z^{n-2} + \cdots + b_{n-1}}{Z^n + a_1 Z^{n-1} + \cdots + a_n} = \frac{B(Z)}{A(Z)} \tag{2.13}$$

Con $K_g = Constante$

Con $C(Z)$ (Controlador) de la forma:

$$C(Z) = \frac{l_0 Z^{n-1} + l_1 Z^{n-2} + \cdots + l_{n-1}}{Z^n + p_0 Z^{n-1} + p_1 Z^{n-2} + \cdots + p_n} = \frac{L(Z)}{P(Z)} \tag{2.14}$$

El objetivo es diseñar el controlador, usando reubicación de polos sobre el sistema en lazo cerrado. Las incógnitas serían entonces K_g, $L(Z)$ y $P(Z)$.

Se mostrará como es el procedimiento para el control de un sistema de orden 1.

Siendo la función de transferencia $H(Z)$ del sistema de la forma:

$$H(Z) = \frac{b_0}{Z + a_1} \tag{2.15}$$

Los valores a_1 y b_0 son conocidos.

El controlador $C(Z)$ debe ser de la forma:

$$C(Z) = \frac{l_0}{Z + p_1} \tag{2.16}$$

Se calcula el sistema en lazo cerrado:

$$\frac{Y(Z)}{R'(Z)} = \frac{C(Z)H(Z)}{1 + C(Z)H(Z)} \tag{2.17}$$

Obteniendo con esto:

$$\frac{Y(Z)}{R'(Z)} = \frac{l_0 b_0}{Z^2 + (p_1 + a_1)Z + l_0 b_0}$$

Lo siguiente es comparar el denominador obtenido con un polinomio deseado, el cual contiene los polos escogidos.

$$P_d = Z^2 + \alpha_1 Z + \alpha_2 \tag{2.18}$$

Donde α_1 y α_2 son valores conocidos. Se observa que el sistema en lazo cerrado resultante, queda con un orden del doble que el original. Para obtener los valores del controlador, se iguala el denominador del sistema en lazo cerrado, con el polinomio deseado, de la siguiente forma:

$$\alpha_1 = p_1 + a_1$$

$$\alpha_2 = l_0 b_0$$

Del sistema de ecuaciones anterior se encuentran los valores de p_1 y l_0. Para sistemas de mayor tamaño es recomendable mostrar el sistema de ecuaciones de forma matricial.

Lo siguiente es encontrar el valor de la constante K_g lo cual se hace de la siguiente forma:

$$R'(Z) = K_g R(Z) \tag{2.19}$$

Entonces:

$$Y(Z) = \frac{l_0 b_0}{Z^2 + (p_1 + a_1)Z + l_0 b_0} K_g R(Z) \tag{2.20}$$

Utilizando el teorema del valor final de la transformada Z asumiendo que el sistema en lazo cerrado es estable:

$$Y_{ss} = \lim_{Z \to 1} (1 - Z^{-1}) Y(Z) \tag{2.21}$$

Y_{ss}: Valor de la salida $Y(Z)$ en estado estacionario.

Con $R(Z)$ escalón unitario:

$$R(Z) = \frac{1}{1 - Z^{-1}} \tag{2.22}$$

$$Y_{ss} = \lim_{Z \to 1} (1 - Z^{-1}) \frac{l_0 b_0}{Z^2 + (p_1 + a_1)Z + l_0 b_0} K_g \frac{1}{1 - Z^{-1}} \tag{2.23}$$

$$Y_{ss} = 1 = \frac{l_0 b_0}{1 + (p_1 + a_1) + l_0 b_0} K_g \tag{2.24}$$

Se le da el valor de 1 a Y_{ss} porque la entrada es un escalón unitario.

Despejando K_g:

$$K_g = \frac{1 + (p_1 + a_1) + l_0 b_0}{l_0 b_0} \tag{2.25}$$

La ecuación de la señal de control queda de la siguiente forma:

$$U(Z) = \frac{l_0}{Z + p_1}(K_g R(Z) - Y(Z)) \tag{2.26}$$

Aplicando el operador de desplazamiento q^{-1} se obtiene lo siguiente:

$$u[k] = \frac{l_0 q^{-1}}{1 + p_1 q^{-1}}(K_g r[k] - y[k]) \tag{2.27}$$

$$(1 + p_1 q^{-1})u[k] = l_0 q^{-1}(K_g r[k] - y[k])$$

De la ecuación anterior se obtiene la expresión para la señal de control $u[k]$:

$$u[k] = -p_1 u[k-1] + l_0 K_g r[k-1] - l_0 y[k-1] \tag{2.28}$$

2.1.2 Control adaptativo

Después de tener todo el modelo del controlador en conjunto con un método de identificación en línea, se puede crear un controlador adaptativo, el cual en cada iteración o después de cierto número de iteraciones del controlador (dependiendo de qué tan dinámico sea el problema), se dé una nueva estimación del sistema y de esta forma, lograr captar los cambios que existan en éste.

La estructura básica de un controlador adaptativo es la siguiente (Fig. 4):

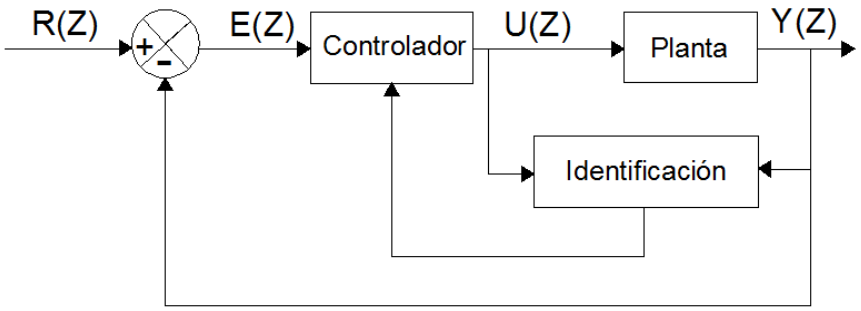

Fig. 4 Diagrama de bloques de un controlador adaptativo básico

Se puede observar en la figura 2.2 que el esquema del controlador sigue siendo el mismo, pero añadiendo un bloque el cual realiza la identificación tomando los valores de entrada y salida de la planta, para así hallarle los parámetros en ese

instante de tiempo y luego llevarlos al controlador, para que éste se adapte a los cambios que se han presentado en la planta.

3. IDENTIFICACIÓN EN LÍNEA DE SISTEMAS UTILIZANDO ALGORITMOS GENÉTICOS

Los métodos de identificación convencionales utilizan técnicas de optimización para llegar a un objetivo, el cual es lograr que el modelo obtenido con los parámetros estimados reproduzca el valor de la salida con un error mínimo. En el método de identificación de proyección, se utiliza la técnica del multiplicador de Lagrange para obtener el valor óptimo de estos parámetros, por lo que en el caso en que un sistema se pueda expresar de diferentes formas (múltiples soluciones), por este método se llegará al óptimo más cercano, pudiendo no ser el óptimo global, trayendo con esto una identificación que puede no ser útil, dependiendo de la aplicación.

El uso de una técnica metaheurística de optimización para la identificación en línea de sistemas podría no ser la primera opción, debido a que por lo general estas técnicas son usadas en problemas en los que el tiempo de cálculo del óptimo no es tan importante, lo cual en la identificación si lo es.

Para llegar a hacer un buen uso de los algoritmos genéticos en la identificación de sistemas, se debe escoger de forma adecuada los parámetros de control, como lo son la tasa de recombinación y la tasa de mutación. También escoger el número de elementos en la población, ya que para cada problema en particular, este número debe cambiar, dependiendo de la rapidez que se desee en el algoritmo y el área de búsqueda de soluciones a explorar.

Otra posibilidad que los algoritmos genéticos ofrecen, es la de poder ser modificados según las necesidades del problema, por ejemplo, si en un inicio el algoritmo utilizado fue el algoritmo genético básico, después se le pueden agregar características de otro tipo de algoritmo, en este caso, el algoritmo genético de Chu Beasley y así solucionar problemas que tiene el primero.

A continuación se mostrará cómo realizar la identificación, primero usando el algoritmo genético básico y luego el que contiene características del algoritmo genético de Chu Beasley.

3.1 Identificación en línea de sistemas usando el Algoritmo Genético Básico

Sea un sistema en tiempo discreto de la forma:

$$A(q^{-1})y[k] = B(q^{-1})u[k] \tag{3.1}$$

Donde los valores de A,B y las variables de la ecuación ya han sido explicados en el capítulo 2. Es posible escribir la salida actual en función de las entradas y salidas anteriores:

$$y[k] = -\sum_{j=1}^{n} a_j y[k-j] + \sum_{j=0}^{n-1} b_j u[k-1-j] \tag{3.2}$$

Factorizando los coeficientes se tienen:

$$y[k] = [a_1\ a_2 \dots\ a_n\ b_0\ b_1 \dots\ b_{n-1}\ ] \begin{bmatrix} -y[k-1] \\ -y[k-2] \\ -y[k-3] \\ \dots \\ -y[k-n] \\ u[k-1] \\ u[k-2] \\ \dots \\ u[k-n] \end{bmatrix} \tag{3.3}$$

$$\theta^T[k] = [a_1\ a_2 \dots\ a_n\ b_0\ b_1 \dots\ b_{n-1}\ ]$$

$$\Phi[k-1] = \begin{bmatrix} -y[k-1] \\ -y[k-2] \\ -y[k-3] \\ \dots \\ -y[k-n] \\ u[k-1] \\ u[k-2] \\ \dots \\ u[k-n] \end{bmatrix}$$

Se desea encontrar el valor de $\theta[k]$ por medio de los valores de entrada u y salida y de muestras anteriores ($\Phi[k-1]$).

3.1.1 Función objetivo

Para el método de identificación de proyección, se utiliza como función objetivo el cuadrado del error de la estimación de los parámetros $\theta[k]$ en el momento actual con el de una muestra anterior, y como restricción, que la salida real del sistema sea igual a la salida estimada (calculada con los parámetros estimados). Ya que el objetivo general de los algoritmos genéticos es maximizar, para problemas como este en el que se desea llegar a una solución en la que el error entre el valor real y el estimado de la salida sea mínimo, se debe buscar una forma de expresar la función objetivo, de forma que el problema de minimización se convierta en uno de maximización. Teniendo esto en cuenta, la función objetivo que se usará para este problema es la siguiente:

Optimizar:

$$f = -|y[k] - ye[k]| \tag{3.4}$$

Donde:

$$ye = \theta^T[k]\Phi[k-1] \tag{3.5}$$

Se puede observar que el objetivo principal es reducir el error del valor de la salida real y la estimada (ye), pero si se tomara esto como función objetivo, sería un problema de minimización el cual no puede resolver el algoritmo genético, por lo que se hace lo que se ve en la ecuación. Primero se le aplica valor absoluto para que todos los valores que se tomen sean únicamente positivos, de igual forma el problema sigue siendo de minimización, entonces se niega toda la expresión, haciendo que el máximo ahora sea el valor cero y todos los demás valores son negativos.

3.1.2 Codificación

Teniendo claro como es la función objetivo del problema, lo siguiente es encontrar una forma de realizar la codificación para este problema. La codificación en este caso se efectuará por medio de números binarios ya que de esta forma es más sencillo utilizar los operadores de mutación y recombinación, además que presenta un mejor manejo de los individuos de la población, aunque requiere de un mayor uso de memoria debido a las cadenas de bits que se deben usar.

Como se explicó en el primer capítulo, el algoritmo genético no emplea el valor de las posibles soluciones sino una codificación de éstas. Para este problema los

individuos de la población (soluciones) serán posibles valores de θ los cuales se codificaran de la siguiente forma, poniendo como ejemplo un sistema de orden uno con los siguientes parámetros (Fig. 5):

$$\theta[k] = [0.3 \;\; 0.1\,]$$
Valor1|Valor 2

Fig. 5 Parámetros para sistema de orden uno

Dependiendo del sistema sobre el que se desea operar, los valores de θ pueden variar, pero a estos se les puede hacer algún proceso matemático para que estén en un rango requerido, por ejemplo en este caso, que los valores de θ estén entre 0 y 1. Lo siguiente sería la representación de estos números en binarios, lo cual se haría de la siguiente forma:

- Crear un vector del número de bits que se requiera, de esto depende la exactitud de la solución. Por ejemplo, si se quiere tener una solución con dos cifras decimales, entonces se tomaría un vector con 7 bits, el cual al convertirlo a números decimales, el valor máximo que podría tener sería el 127. Después de tener este valor, se divide por cien y ya se tendría un número que va desde 0 hasta 1.27.

$$\theta = [0\;0\;1\;1\;1\;1\;0\;0\;0\;0\;1\;0\;1\;0]$$
Valor 1 | Valor 2

Fig. 6 Individuo para sistema de orden uno codificado

- Al ser dos valores los que van dentro del vector θ, el vector solución debe ser del doble de bits. Para el caso del sistema de orden uno, el vector quedaría de 14 bits, tomando los primeros 7 como el valor 1 del vector θ y los últimos 7 como el valor 2.

Después de tener la codificación del problema, se puede empezar a crear la estructura del algoritmo genético.

3.1.3 Población inicial

Como se dijo en el primer capítulo, la población inicial se genera aleatoriamente, teniendo en cuenta que el número de individuos dependerá del orden del sistema; entre mayor sea el orden del sistema, el número de individuos debe ser mayor. Lo dicho anteriormente podría tener problemas por costo computacional, pero llegado este caso se debe implementar alguna metodología para mejorar este aspecto, por ejemplo, acortar el tamaño de las configuraciones, teniendo menos exactitud, pero más rapidez en el algoritmo.

3.1.4 Selección

Después de tener la población inicial, se debe implementar el operador de selección, el cual para el caso de la identificación de sistemas, es más práctico usar un método diferente al de la ruleta, ya que si la función objetivo tiene valores negativos, se presentarán problemas a la hora de decir cuántos hijos tendrá cada individuo de la población. Entre los otros métodos de selección existentes, uno que puede ser muy útil es el método de selección por torneo, en el cual no existe el problema que tiene el de la ruleta.

El método de selección por torneo consiste en hacer un número de torneos de igual número de la población, en el cual se toman un número de soluciones cualquiera aleatoriamente de la población (normalmente entre 3 y 5) y se les mide el valor de función objetivo a cada uno; el elemento con mayor valor de función objetivo será el que entre a la población.

3.1.5 Mutación y Recombinación

Al tener definido el método de selección, se implementan los otros mecanismos del algoritmo genético, la recombinación y la mutación, de igual forma que se hizo en el capítulo 1. Los parámetros de control (tasa de recombinación y tasa de mutación) se escogen dependiendo del orden del sistema y de que tan dinámico sea.

3.1.6 Criterio de parada

En este caso no existe un criterio de parada definido, ya que el algoritmo genético irá evolucionando a medida que el sistema también lo haga, por lo que el algoritmo se detendrá cuando todo el proceso se detenga.

Como la identificación por sí sola no es muy útil, se adicionará un control, y al tener identificación del sistema permanente, se podrá crear una estructura de control adaptativo. Dependiendo de qué tan rápido sea el equipo en que se ejecutará el algoritmo, se pueden realizar varias iteraciones por cada ciclo de control.

Para resumir, el algoritmo de control adaptativo usando identificación con algoritmos genéticos es el siguiente:

1. Se genera aleatoriamente la población inicial.
2. Se toman los valores de u[k] y de y[k] necesarios dependiendo del orden del sistema.
3. Se aplica el operador de selección para observar que individuos están mejor adaptados y tienen derecho a reproducirse.
4. Se aplican los operadores de recombinación y mutación.
5. Se mide el valor de la función objetivo de todos los individuos resultantes y se escoge como incumbente el que sea mayor.
6. Se realiza la conversión de binario a decimal y se hacen las operaciones necesarias para tener el valor de los parámetros de la identificación.
7. Se introducen los valores obtenidos de la identificación al controlador.
8. Se genera la señal de control.
9. Volver al paso 2.

3.2 Identificación en línea de sistemas usando propiedades del algoritmo genético de Chu-Beasley

Como se dijo anteriormente, una buena característica del algoritmo genético es que puede ser modificado según las necesidades del problema. Algunos problemas que se han visto en la identificación con el algoritmo genético básico son:

- La convergencia prematura a una solución no del todo óptima.
- La dificultad del algoritmo para llegar a otra solución cuando el sistema cambia sus parámetros.
- No responde bien ante perturbaciones.

3.2.1 Características del algoritmo genético usando propiedades del algoritmo genético de Chu-Beasley

Para definir cómo será el algoritmo genético con propiedades del algoritmo genético de Chu Beasley, se debe empezar por la creación de la población inicial. En el algoritmo genético de Chu Beasley se genera la población inicial teniendo un criterio de diversidad, esto quiere decir que se debe asegurar que todos los individuos de la población son diferentes. Se debe tener en cuenta que la diferencia de un individuo a otro se ve reflejada en sus bits, por ejemplo, en la figura 7 se muestran dos individuos con dos bits de diferencia:

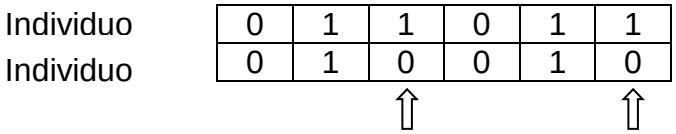

Fig. 7 Individuos con dos bits de diferencia

En la Fig. 7 se pueden observar dos individuos con dos bits de diferencia señalados por las flechas. La diferencia de bits entre los individuos de la población es escogida a criterio del que ejecute el algoritmo, dependiendo del tamaño de los individuos y de la población. Otra característica que tiene el algoritmo genético de Chu Beasley para generar la población inicial, es usar una técnica heurística para llevar a algunas configuraciones cerca a un óptimo (generalmente un óptimo local) y tener estos puntos de partida para el algoritmo. Para el algoritmo a utilizar no se usará lo dicho anteriormente, ya que aumentaría el tiempo del proceso.

En el algoritmo de Chu Beasly después de generar la población inicial, viene un cambio drástico con respecto al algoritmo genético básico, que consiste en que en cada generación solo se crea a un nuevo descendiente y este puede entrar a la población únicamente si es mejor que el peor individuo, mientras que en el básico toda la población es cambiada o la mayoría de ella. Creando un solo individuo por generación, las iteraciones serán mucho más rápidas, pero también se incrementa el tiempo para encontrar una solución.

El operador de selección se implementa por el método de torneo. Anteriormente se hacia un número de torneos igual al número de individuos en la población, pero ahora se realizan dos torneos, en los cuales se toman configuraciones al azar (el número que al ejecutor del programa le parezca adecuado) y de estos dos torneos queda como resultado dos padres, los cuales son las mejores configuraciones de los respectivos torneos.

Para implementar el operador de recombinación, se toma a los dos padres, se da un punto de cruce aleatorio (o más de uno si se quiere) y se cambian las partes del individuo correspondientes. Después de esto, se toma al individuo (hijo) con mejor función objetivo de los dos resultantes y se pasa al siguiente operador.

Para implementar el operador de mutación, se toma el individuo resultante y se le aplica mutación de la misma forma como se ha explicado anteriormente. En el algoritmo genético de Chu Beasley el valor de la tasa de mutación es mayor a la usada en el básico. Es importante que la tasa de mutación sea grande, ya que al hacer identificación en sistemas dinámicos, es posible que la población entera converja a un valor que en un momento pudo ser óptimo, pero al haber un cambio en el sistema ya no lo es y la única manera de salir de ese óptimo es por medio de la mutación, ya que es el único operador encargado de introducir nueva información a los individuos. Es importante aclarar que en el algoritmo genético básico, cuando toda la población converge hacia el mismo valor, es difícil salir de él.

Después de tener el individuo al que se le aplica la mutación y la recombinación, se le mide el valor de función objetivo y se compara con el que tiene el menor valor de toda la población; si el valor de función objetivo del hijo es mayor al que tiene el peor valor de toda la población, este ultimo será sustituido por el hijo, en caso contrario, la población pasa a la siguiente generación sin ser modificada.

Por último se toma al individuo con mejor valor de función objetivo de toda la población y éste es el que se introduce en el algoritmo de control.

A continuación se mostrará un ejemplo de cómo realizar la identificación con los algoritmos genéticos con características del de Chu Beasley.

3.2.2 Ejemplo del algoritmo genético con características del algoritmo genético Chu-Beasley

Lo primero es tener claridad sobre el problema a optimizar, en este ejemplo se quiere obtener la identificación de un sistema de orden uno.

$$y[k] = [a_1\ b_0]\begin{bmatrix} -y[k-1] \\ u[k-1] \end{bmatrix} \tag{3.6}$$

Donde lo que se quiere encontrar son los parámetros del sistema a_1 y b_0 incluidos en un vector llamado Φ.

Después de saber esto, se debe realizar la codificación del problema, la cual se hará como se explicó anteriormente en este capítulo, creando vectores de ceros y unos que representen números en binario. Para cada parámetro del sistema (a_1 y b_0) se tendrán cinco bits, lo cual significa que el máximo número en binario que se tendrá será el 31 (11111) y estos números se dividirán entre 10 y luego se le restará 1.5, para así tener números desde -1.5 hasta 1.6 con una cifra decimal. Teniendo esto claro, se procede a generar la población inicial, asegurando dos bits de diferencia entre los individuos para garantizar la diversidad.

Tabla 6. Población inicial con diversidad

	Parámetro a_1					Parámetro b_0				
Individuo 1	1	0	0	1	1	0	1	0	1	1
Individuo 2	1	0	1	0	1	1	1	0	0	1
Individuo 3	0	1	1	0	1	1	0	1	1	0
Individuo 4	1	1	1	0	1	0	1	1	1	1
Individuo 5	0	0	0	1	0	1	1	1	0	0
Individuo 6	1	0	1	1	0	1	1	0	0	0

Se puede observar en la tabla 6 que es una población de 6 individuos, en los cuales los primeros 5 bits de izquierda a derecha son representaciones binarias del parámetro a_1 y los otros cinco de b_0. Se toma como una solución el vector completo que contiene los dos parámetros.

Realizando la conversión de binario a decimal y con las operaciones anteriormente dichas realizadas, la población con los valores de los parámetros verdaderos quedaría de la siguiente forma (Tabla 7):

Tabla 7. Población inicial convertida de binario a decimal

	a_1	b_0
Individuo 1	0.3	-0.4
Individuo 2	0.6	1
Individuo 3	-0.2	0.7
Individuo 4	1.4	0
Individuo 5	-1.3	1.3
Individuo 6	0.7	0.9

Teniendo definida la población inicial, se procede a ejecutar el operador de selección, el cual se hará por medio del torneo. Se harán dos torneos, en los cuales se tomarán aleatoriamente tres individuos y se les medirá el valor de función objetivo.

Antes de poder medir el valor de función objetivo de cada uno de los individuos se deben adquirir los valores de la entrada anterior $u[k-1]$ y el valor de la salida $y[k-1]$ y además el valor de la salida en el instante actual $y[k]$. Para este ejemplo se tendrán los siguientes valores, que fueron obtenidos a través de una simulación previamente realizada:

$$u[k-1] = 1$$

$$y[k-1] = 0.4$$

$$y[k] = 0.2$$

Con estos valores se empezará a remplazar las posibles soluciones que se presentan en la población en el modelo de un sistema de orden uno.

$$ye[k] = -a_1 y[k-1] + b_0 u[k-1] \tag{3.7}$$

Se le da el nombre de $ye[k]$ al valor que se encuentra con los parámetros que están en la población, los cuales son posibles soluciones al problema.

Después de tener los valores de $ye[k]$ para toda la población, se procede a calcular el valor de función objetivo de la siguiente forma:

$$f = -|y[k] - ye[k]|$$

Teniendo como resultado lo mostrado en la tabla 8:

Tabla 8. Valores de función objetivo de la población

	Función Objetivo
Individuo 1	-0.72
Individuo 2	-0.56
Individuo 3	-0.58
Individuo 4	-0.76
Individuo 5	-1.66
Individuo 6	-0.42

Con los valores de función objetivo, se procede a realizar los torneos, tomando aleatoriamente 3 individuos por torneo, en este caso, para el primer torneo se toman los individuos 1, 2 y 3 y para el otro torneo se toman a los individuos 4, 5 y 6, teniendo como ganadores a los individuos con mayor valor de función objetivo para cada torneo respectivo, que son el individuo 2 (-0.56) y el individuo 6 (-0.42).

Como se dijo anteriormente, en el algoritmo genético de Chu Beasley solo se genera un hijo por generación, el cual es obtenido a partir de los dos ganadores de los torneos. Los padres con la codificación son los siguientes (Fig. 8):

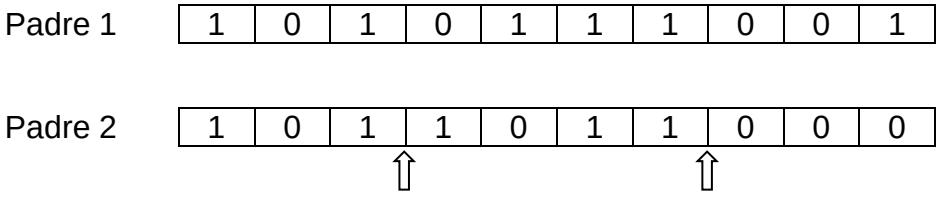

Fig. 8 Puntos escogidos para realizar recombinación en dos puntos

Con estos dos padres se ejecutará el operador de recombinación. En este caso se realizará la recombinación en dos puntos, los cuales son escogidos aleatoriamente con números desde el 1 hasta el 9 (con el 10 no existiría ningún efecto). En este ejemplo se tomarán los puntos de recombinación en 3 y 7 como se muestra en la figura anterior, obteniendo las siguientes configuraciones (Fig. 9):

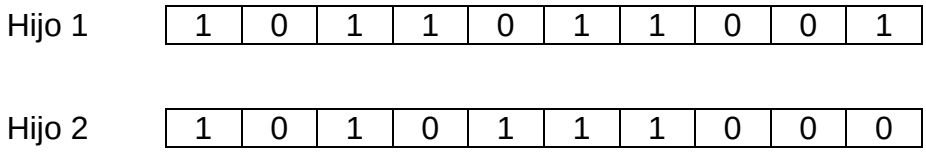

Fig. 9 Hijos resultantes de la recombinación

Quedando entonces con los siguientes valores de función objetivo:

Hijo 1: -0.52

Hijo 2: -0.46

De los cuales se toma solo el mejor de los dos hijos, siguiendo al operador de mutación el hijo 2. Para ejecutar este operador, se debe seleccionar un valor de tasa de mutación, el cual en el algoritmo genético de Chu Beasley es alto, para este caso se tomará un valor de 0.25. La mutación se hace de la misma forma que se explicó en el capítulo 1, dando como resultado lo siguiente (Fig. 10):

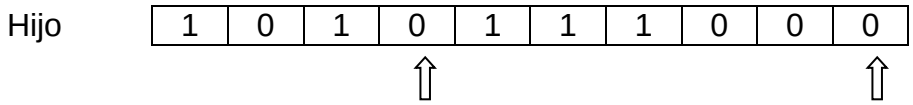

Fig. 10 Puntos escogidos para realizar la mutación

Los bits señalados con las flechas son los que se escogieron aleatoriamente, quedando el hijo así (Fig. 11):

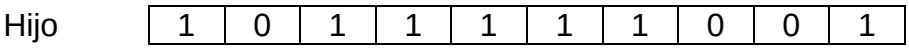

Fig. 11 Hijo resultante de la mutación

Con valor de función objetivo de -0.48.

Lo siguiente en el algoritmo es verificar que el hijo obtenido tiene mejor valor de función objetivo que el peor de la población, si es así, el hijo resultante remplaza al individuo anteriormente mencionado, si no es así, se pasa a la siguiente generación sin modificar la población existente. En este caso el peor valor de función objetivo la tiene el individuo 5 con -1.66, por lo cual se permite al hijo resultante hacer parte de la población remplazándolo. La población queda entonces de la siguiente forma (Tabla 9):

Tabla 9. Población después de una generación

	Parámetro a_1					Parámetro b_0				
Individuo 1	1	0	0	1	1	0	1	0	1	1
Individuo 2	1	0	1	0	1	1	1	0	0	1
Individuo 3	0	1	1	0	1	1	0	1	1	0
Individuo 4	1	1	1	0	1	0	1	1	1	1
Hijo	1	0	1	1	1	1	1	0	0	1

Individuo 6	1	0	1	1	0	1	1	0	0	0

Y la incumbente (mejor configuración) de esta generación es el individuo 6, el cual después de ser convertido de binario a decimal, en un esquema de control adaptativo, entra al controlador. Dependiendo de qué tan rápido sea el equipo en que se ejecute este algoritmo, se pueden realizar un número determinado de iteraciones por ciclo de control, en ese caso, se tomaría el valor de la incumbente después de ese número de generaciones para entrar al algoritmo de control.

4. ANÁLISIS DE RESULTADOS

4.1 Simulaciones

La primera simulación se hace sobre un sistema de orden dos; este modelo simula el comportamiento de la velocidad de un servomotor, en el cual la salida del sistema es la tensión tomada en un taco generador, que es proporcional a la velocidad del motor. Se muestra dicho modelo a continuación, usando un periodo de muestreo de 0.1s. [5]

$$\frac{Y(Z)}{U(Z)} = \frac{0.5965\,Z - 0.2091}{Z^2 - 0.6221\,Z - 0.0857} \tag{4.1}$$

Esta simulación se realiza usando el método de control de variables de estado extendidas, en el cual se toman las entradas y salidas del sistema como variables de estado. Los resultados que se obtuvieron se muestran en la figura 12:

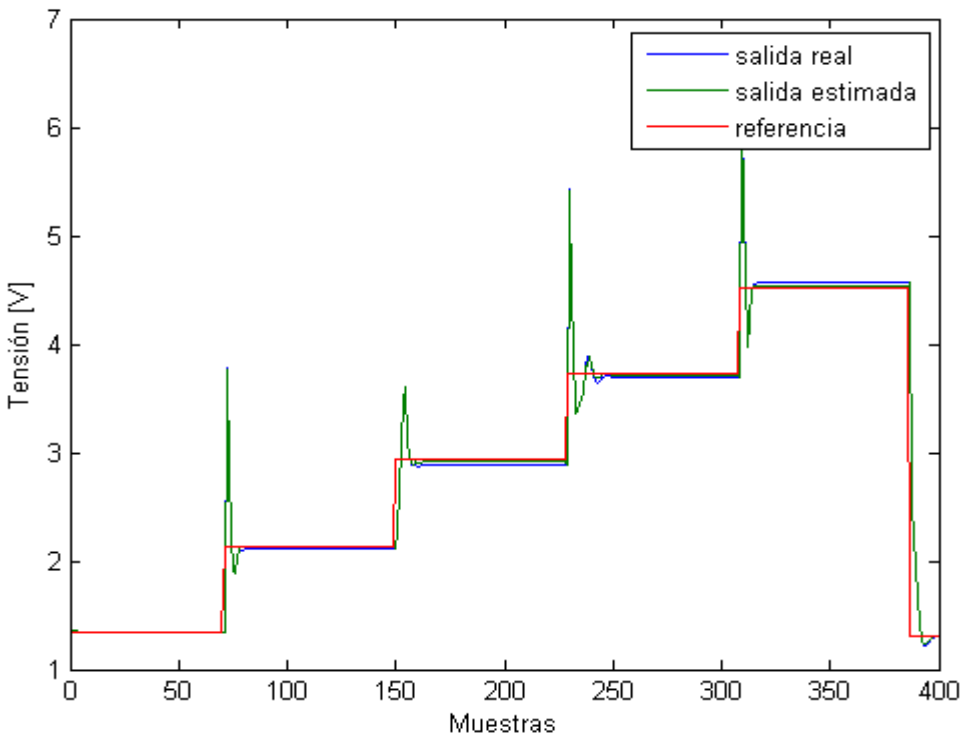

Fig. 12 Identificación con algoritmos genéticos y control en variables de estado extendidas

Los parámetros del algoritmo de identificación y control usados fueron los siguientes:

Número de individuos en la población: 100.

Iteraciones del algoritmo genético por iteración de control: 15.

Periodo de muestreo: 0.1s

Individuos por torneo (selección): 4

Tasa de recombinación: 100%.

Tasa de mutación: 20%.

Polos del controlador: todos en 0.2.

Se puede observar que la identificación que se hace es buena, ya que la mayor parte del tiempo las dos señales coinciden, además que con esta identificación se logra tener un buen control.

La siguiente simulación se hace sobre el mismo sistema, pero lo que se quiere observar es como responde ante perturbaciones. En la figura 13 se muestran los resultados:

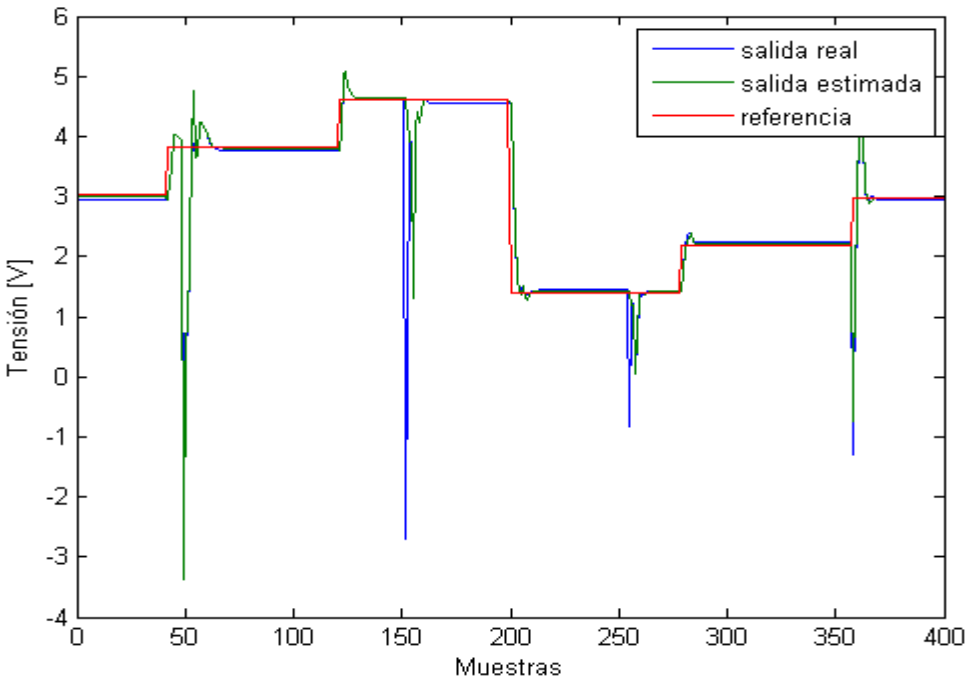

Fig. 13 Identificación con algoritmos genéticos y control en variables de estado extendidas, con perturbaciones

En la figura 13 se puede observar que aproximadamente cada 100 muestras se introduce una perturbación, la cual se realiza cambiando todos los parámetros el sistema durante 3 muestras. Se puede observar que el algoritmo genético responde adecuadamente ante perturbaciones de poca duración, ya que el tiempo de la perturbación no es suficiente para hacer cambiar todas las posibles soluciones de la población, por lo que al volver al estado normal, aún existen soluciones para que el sistema se logre estabilizar. Cuando el cambio del sistema se da por más tiempo, la identificación debe ser capaz de adaptarse a ese nuevo modelo.

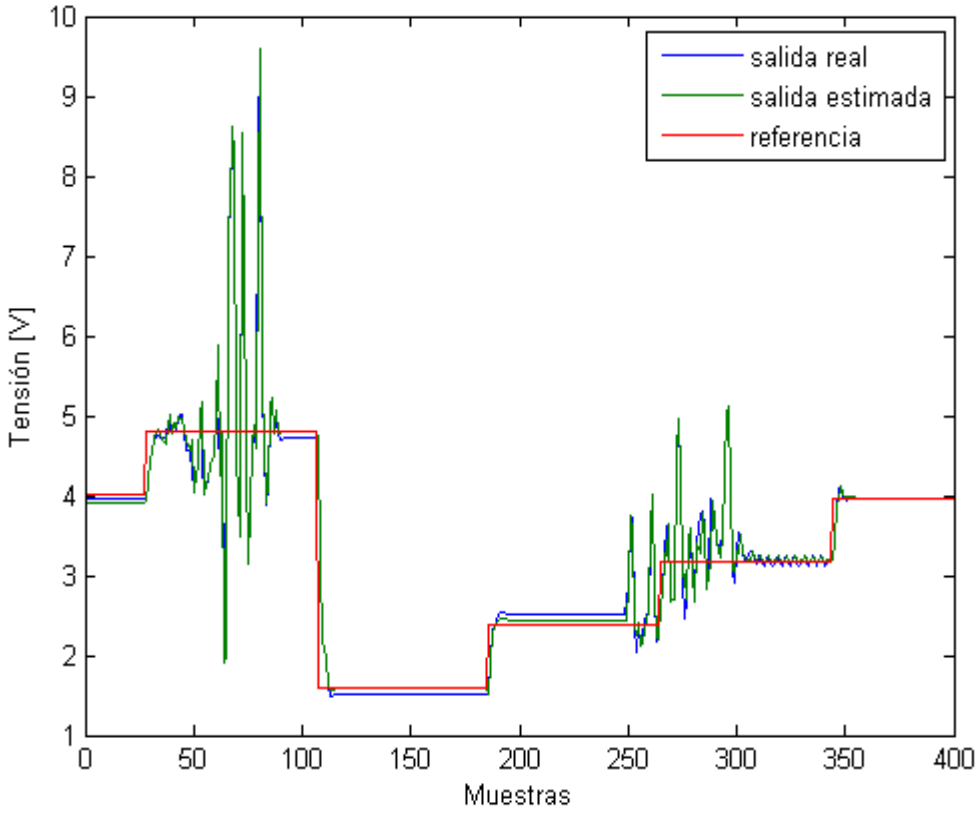

Fig. 14 Identificación con algoritmos genéticos y control en variables de estado extendidas, con cambios temporales en el sistema

En la figura 14 cada 200 muestras se cambia el modelo; el primer cambio se realiza aproximadamente en la muestra 50 y el siguiente por la muestra 250; se puede ver que el algoritmo de identificación logra adaptarse a dicho cambio generando algunas oscilaciones al principio.

Para probar el algoritmo genético con otro controlador, se efectúa la siguiente simulación, con un algoritmo de control en espacio de estados con acción integral. Se obtienen los siguientes resultados:

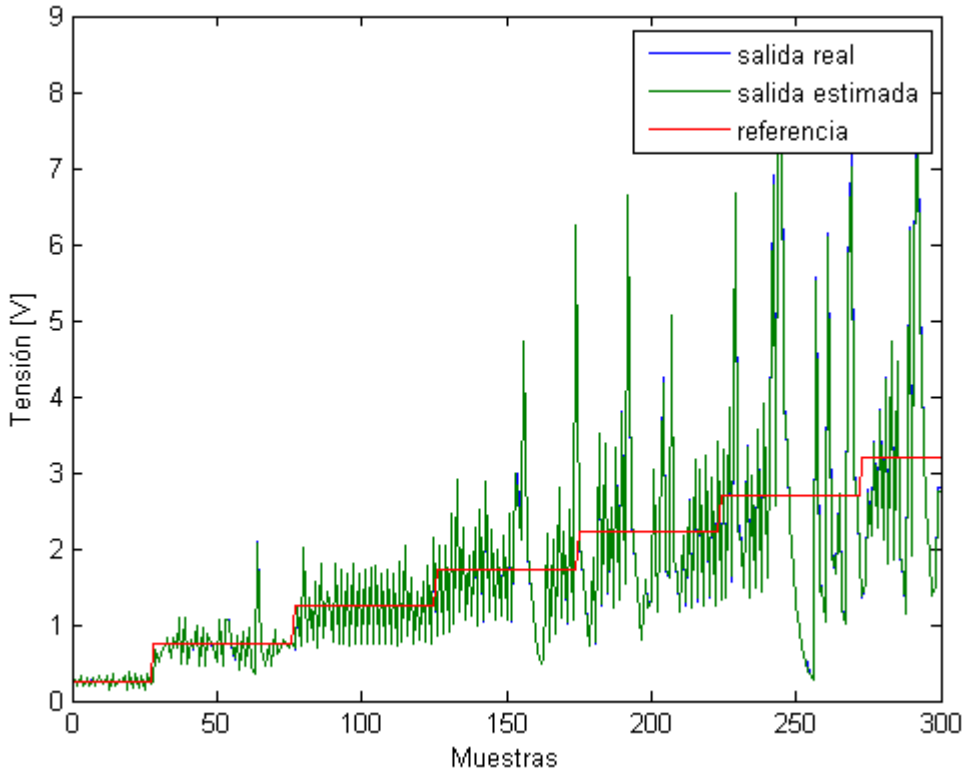

Fig. 15 Inconvenientes del algoritmo genético en conjunto con el observador

En la figura 15 se logra ver que la señal de salida trata de seguir la referencia, pero con una gran cantidad de oscilaciones. Una conclusión importante que se obtiene es que el algoritmo genético en cada iteración del control, genera posibles soluciones y toma la mejor para usarla en el controlador, pero casi siempre se da que de una iteración a otra, la mejor solución cambie, lo que trae grandes problemas con el observador del controlador, que tiene que estimar las variables de estado, pero con este cambio constante de modelo hace que no pueda obtener dicha estimación.

Como el problema es el tiempo que toma el observador en hacer la estimación de las variables de estado, se puede idear una metodología para que la identificación no se ejecute en todo momento. La forma de hacer esto es darle una condición, para que dependiendo de la diferencia entre la señal de salida estimada y la real, el algoritmo de identificación entregue un nuevo valor. En la figura 15 se observa

que las dos señales coinciden, por lo tanto la identificación es correcta. Esto se logra dándole un nuevo valor de identificación en cada iteración, aunque no es completamente necesario. Cuando estas dos señales se mantienen en un margen determinado, se puede lograr que el cambio no sea tan fuerte y el observador pueda hacer la estimación. A esto también se le añade la condición de un error entre la señal de salida real y la referencia, ya que puede darse que se mantenga la diferencia entre las señales de salidas real y estimada, pero el sistema no esté respondiendo de la manera que se requiera con el control. El resultado se muestra en la figura 16:

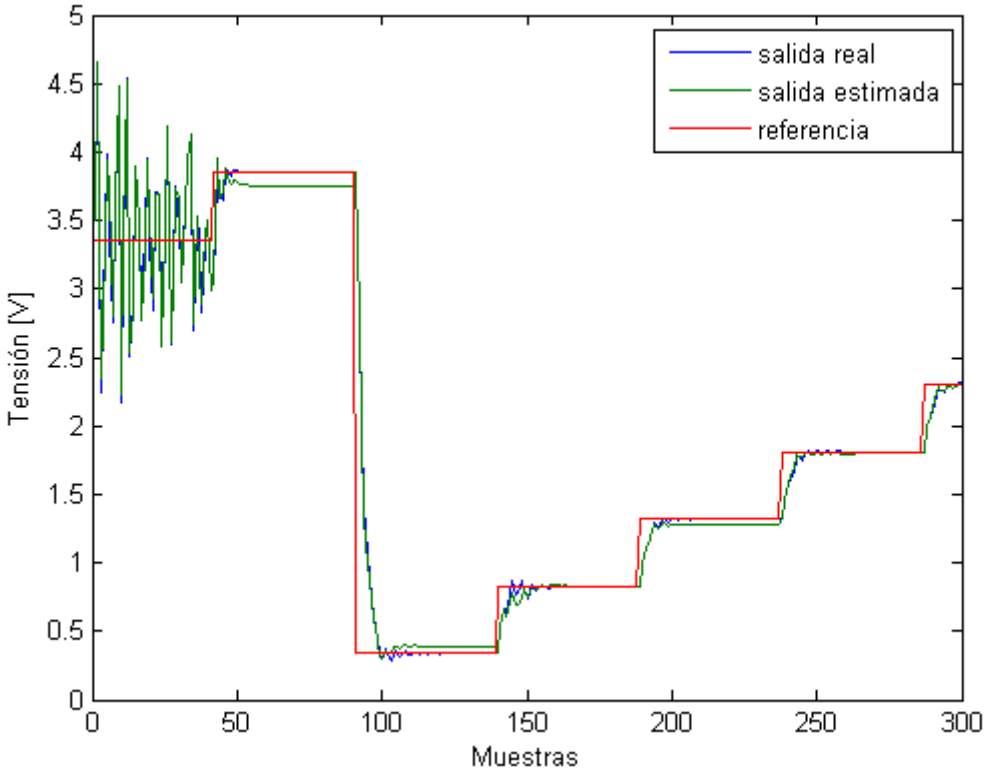

Fig. 16 Alternativa para corregir el problema del algoritmo genético y el observador

En la figura 16 se puede observar que se logra corregir el problema de la estimación de las variables de estado, obteniendo como resultado un control aceptable; aunque se haya podido realizar esto, el numero de iteraciones para lograrlo es alto y en un futuro puede darse que vuelva a aparecer el inconveniente (según las pruebas realizadas), por lo que no es recomendable utilizar este método de control en este problema.

4.2 Resultados sobre sistemas reales

A continuación se enseñan los resultados obtenidos, primero sobre circuitos de diferente orden y después sobre una planta operando en dos modos diferentes. Para la adquisición de datos, se usa una tarjeta NI USB-6008 con la cual también se aplica la señal de entrada que el algoritmo entrega.

4.2.1 Circuito RC de orden 1

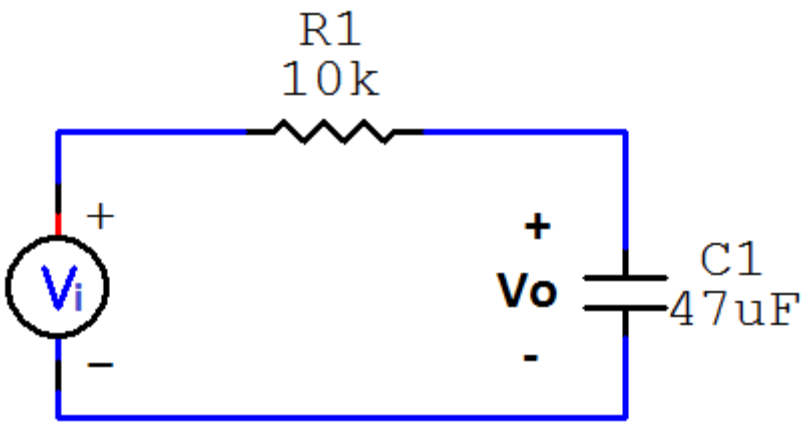

Fig. 17 Circuito RC de orden 1

En figura 17 se puede observar que el circuito está constituido por una resistencia y un capacitor, teniendo como señal de entrada la tensión que entrega la tarjeta de adquisición de datos después de ejecutar el algoritmo y como señal salida la tensión en el capacitor. El objetivo es obtener una identificación acertada, de tal modo que se pueda usar para controlar la tensión del capacitor. El resultado se muestra en la figura 18:

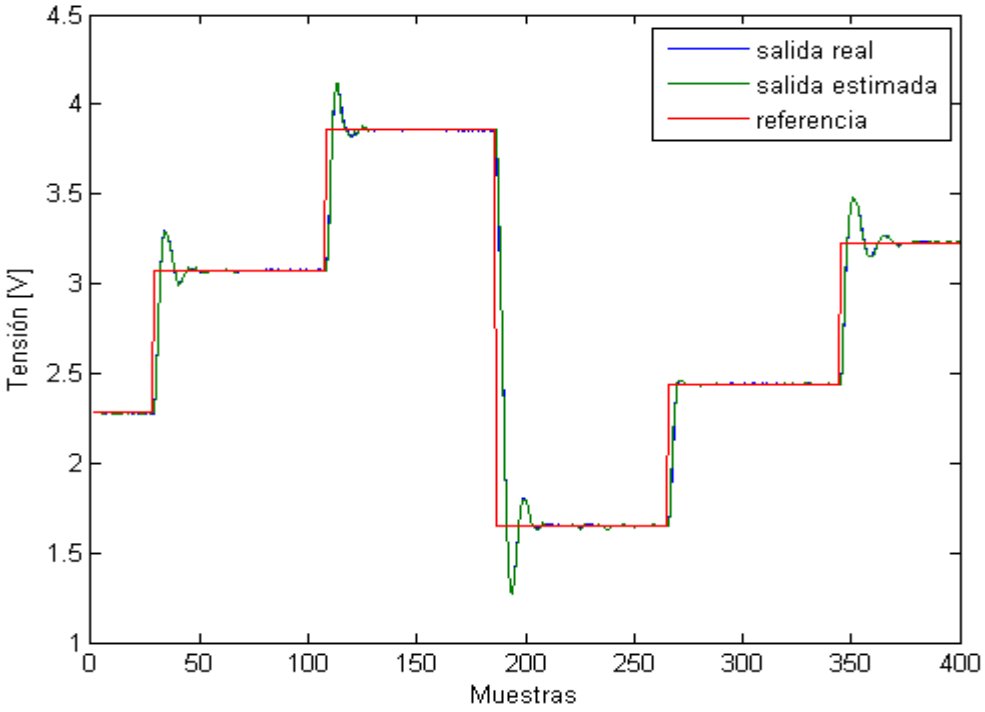

Fig. 18 Identificación y control de un circuito RC de orden 1

Al ser un circuito de orden 1, el algoritmo de identificación debe obtener dos parámetros, por lo que no es necesaria una gran población. Además el tiempo que tarda el algoritmo en obtener una buena estimación es casi instantáneo y se puede utilizar para el control, que hace seguir la señal de salida de manera correcta a la referencia.

Los parámetros del algoritmo de identificación y control fueron los siguientes:

Número de individuos en la población: 40.

Iteraciones del algoritmo genético por iteración de control: 10.

Periodo de muestreo: 0.1s

Individuos por torneo (selección): 4

Tasa de recombinación: 100%.

Tasa de mutación: 10%.

Polos del controlador: todos en 0.2.

Se debe aclarar que el ajuste de estos parámetros se efectúa a ensayo y error, por lo que se debe tener un previo conocimiento de su comportamiento.

4.2.2 Circuito RC de orden 2

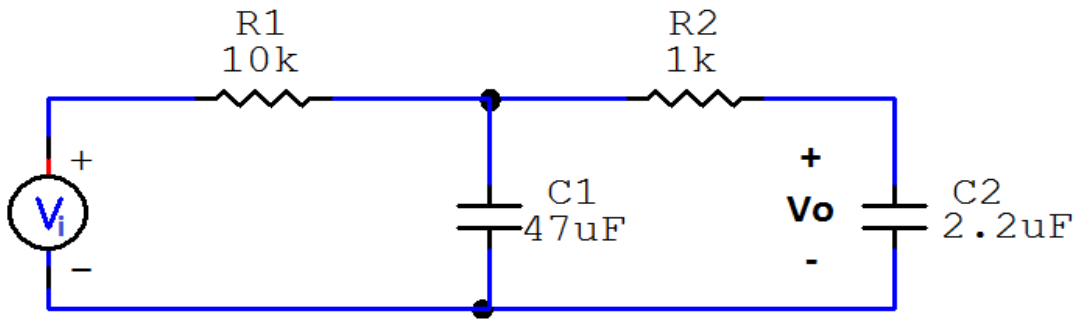

Fig. 19 Circuito RC de orden 2

Al aplicar el algoritmo de identificación y de control sobre el circuito de la figura 19, se obtuvo el resultado que se muestra en la figura 20:

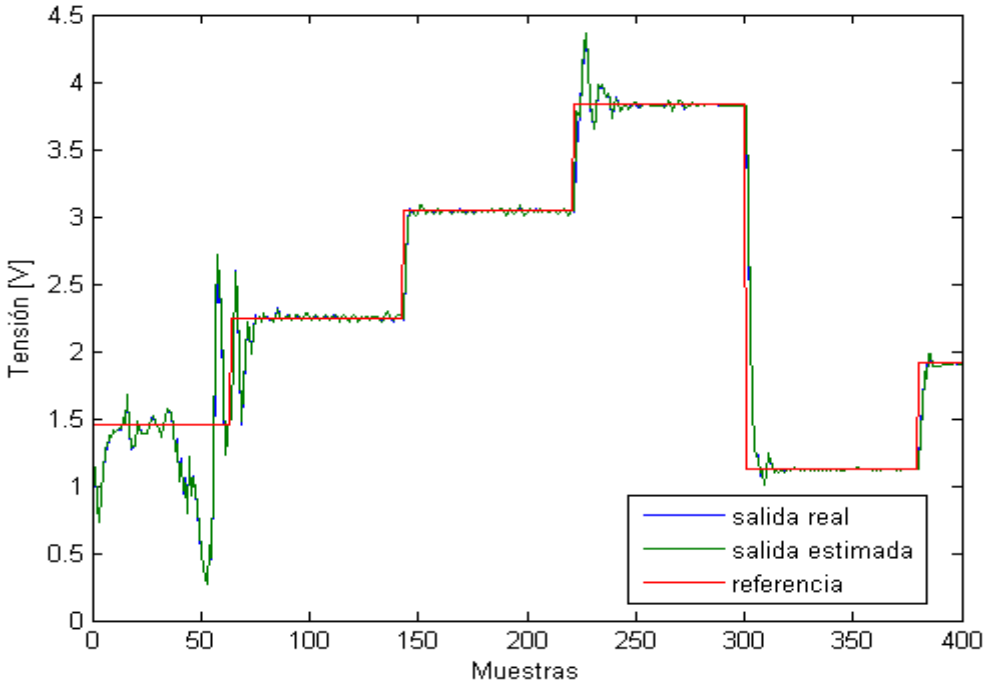

Fig. 20 Identificación y control de un circuito RC de orden 2

La figura 20 muestra el progreso desde que se inicia el algoritmo; además se observa que toma un corto tiempo realizar una identificación que pueda ser usada

por el algoritmo de control, ya que desde el comienzo la señal de salida estimada coincide con la señal de salida real, pero el control no es efectivo. Al ser un sistema de orden 2, el algoritmo de identificación debe estimar 4 parámetros, por lo que se necesita de una población más grande y ajustar los parámetros de forma correcta.

Los parámetros del algoritmo de identificación y control fueron los siguientes:

Número de individuos en la población: 80.

Iteraciones del algoritmo genético por iteración de control: 20.

Periodo de muestreo: 0.1s

Individuos por torneo (selección): 5

Tasa de recombinación: 100%.

Tasa de mutación: 10%.

Polos del controlador: todos en 0.2.

4.2.3 Planta de control marca “Feedback”

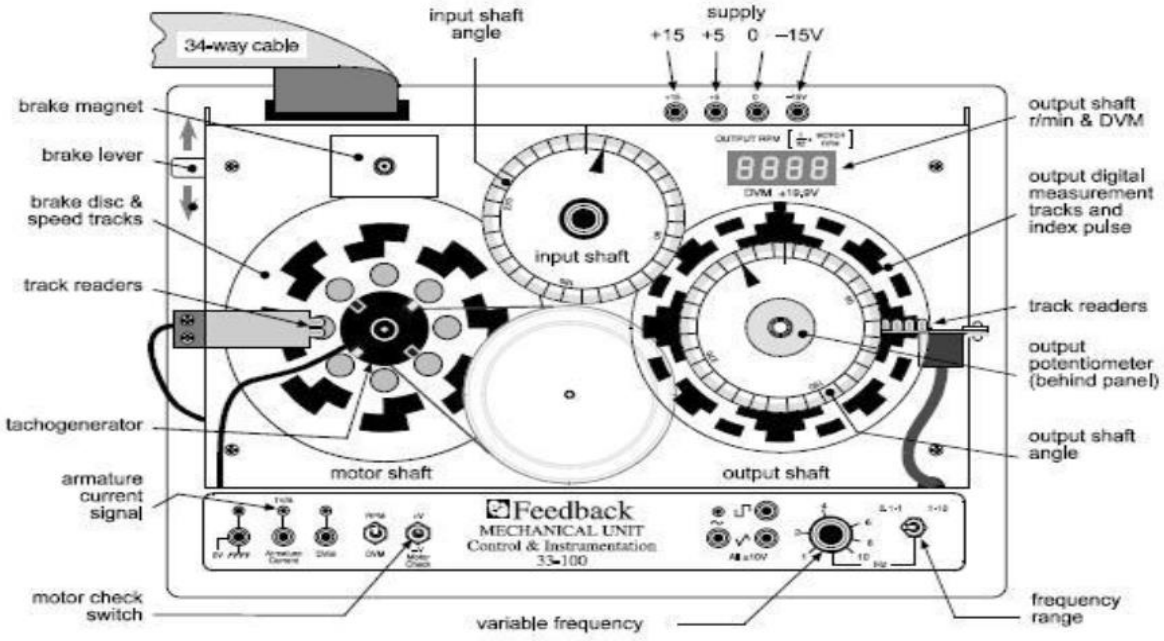

Fig. 21 Planta de control marca “Feedback”

Este sistema consta de un servomotor, el cual por medio de cintas, hace girar la rueda del lado derecho. Este sistema se puede controlar de dos formas, una es

hacer un control de velocidad y la otra es un control de posición angular. El servomotor está conectado a un taco generador, el cual da un valor de tensión, dependiendo de la velocidad con que gira el servomotor; la tensión obtenida por el taco generador es la señal de salida que se toma para el control de velocidad. Por otro lado, el sistema cuenta con un puerto de salida, para tomar el valor de la posición angular señalada por la flecha de la rueda de la derecha; cuando la flecha esta en el rango de 0° a 180°, genera una tensión entre 0V y 10V respectivamente; cuando la flecha esta en el rango de 0° a -180°, se genera una tensión entre 0V y -10V respectivamente. La señal de entrada para los dos modos de operación, es la tensión aplicada al servomotor.

4.2.3.1 Control de velocidad

Como se mencionó anteriormente, para realizar este control se toma el valor de tensión que entrega el taco generador, por medio de la tarjeta de adquisición de datos, que es proporcional a la velocidad del servomotor. La señal de entrada es la que se obtiene después de ejecutar el algoritmo de control (señal de control) la cual se le inyecta al servomotor. Usando el algoritmo de identificación con algoritmos genéticos y el algoritmo de control en espacio de estados extendidos, los resultados se muestran en la figura 22:

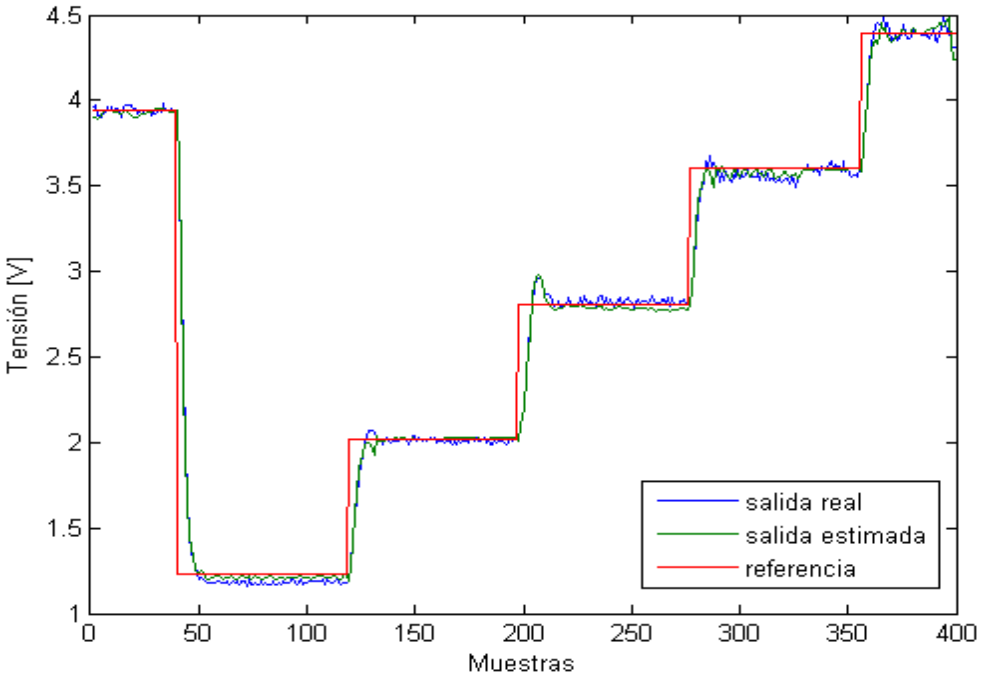

Fig. 22 Identificación y control de velocidad

Se aprecia que la señal de salida real oscila alrededor del valor deseado y solo en algunos casos existe un error poco significativo entre la salida real y la estimada.

Dándole continuidad al proceso, se observa cómo responde el algoritmo ante perturbaciones (figura 23).

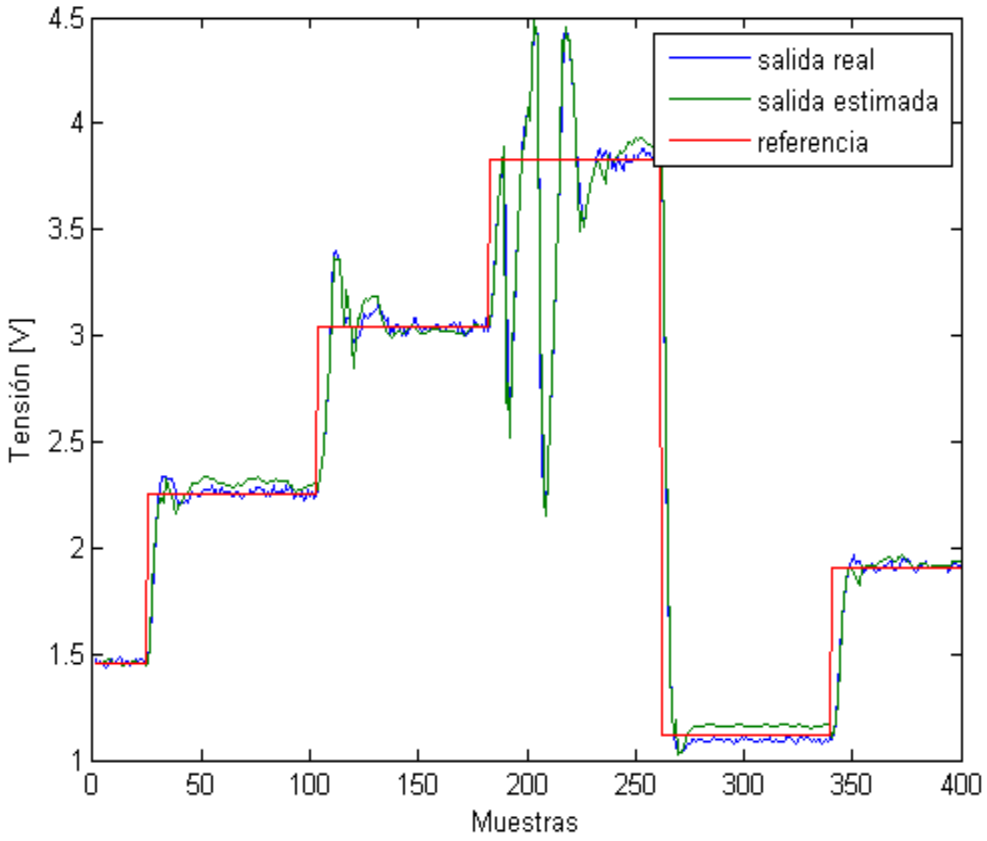

Fig. 23 Identificación y control de velocidad con perturbación

En la figura 23, se observa como alrededor de la muestra 180 se perturba al sistema, lo cual genera oscilaciones, pero después de un tiempo vuelve a su estado normal, con pequeñas oscilaciones.

Por último, se pone a prueba el algoritmo para un cambio de parámetros en un intervalo de tiempo prolongado. El resultado se muestra en la figura 24:

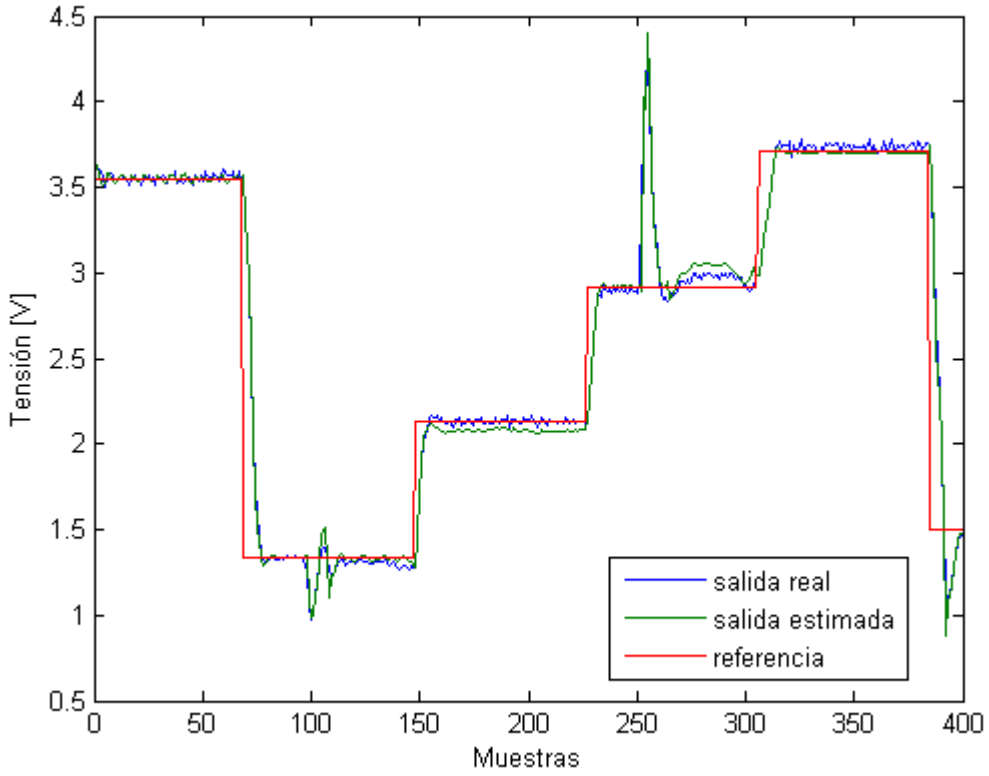

Fig. 24 Identificación y control de velocidad con cambio en los parámetros

Aproximadamente en la muestra 100, se le aumenta la fricción que tiene la rueda, lo que equivale a un cambio de parámetros, pero se observa que el algoritmo logra una rápida adaptación. En la muestra 250 se devuelve el sistema al estado inicial, desestabilizándolo por un momento, pero adecuándose poco después.

Los parámetros del algoritmo de identificación y control fueron los siguientes:

Número de individuos en la población: 100.

Iteraciones del algoritmo genético por iteración de control: 20.

Periodo de muestreo: 0.1s

Individuos por torneo (selección): 5

Tasa de recombinación: 100%.

Tasa de mutación: 10%.

Polos del controlador: todos en 0.2.

4.2.3.2 Control de posición angular

El sistema sobre el cual se va a aplicar el algoritmo, puede ser modelado de orden tres, por lo que los vectores para realizar la identificación crecen de tamaño y se debe ajustar de manera correcta los parámetros, de tal modo que el algoritmo siga teniendo la velocidad suficiente para no afectar el periodo de muestreo. Teniendo esto claro, los resultados se muestran en la figura 25:

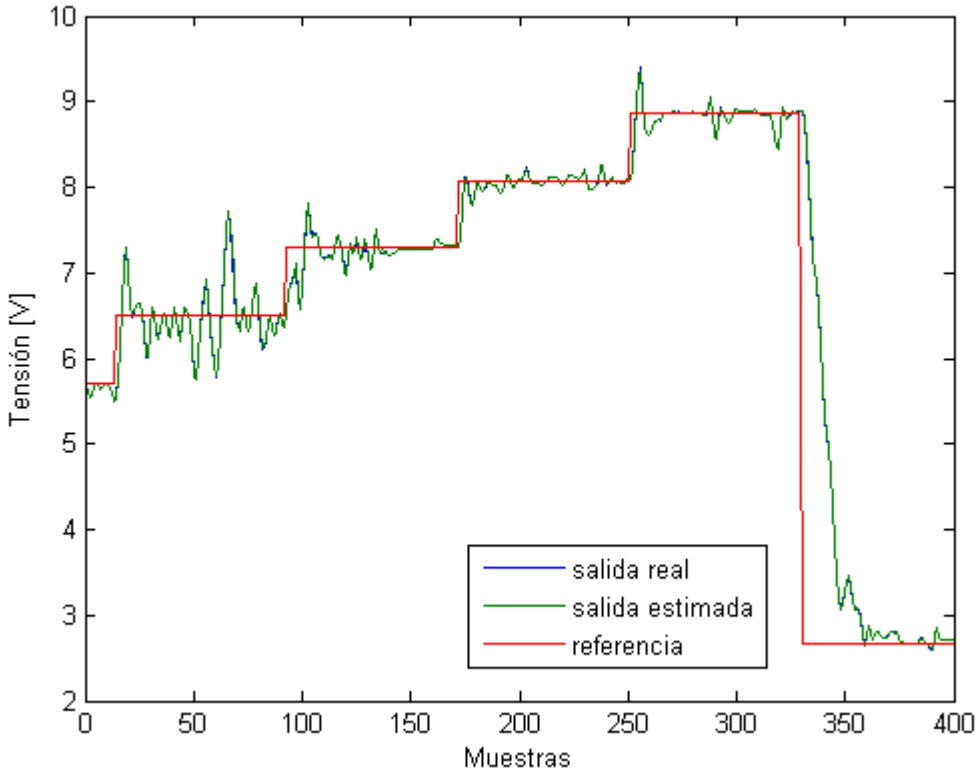

Fig. 25 Identificación y control de posición angular en las primeras iteraciones

En la figura 25 se puede observar como desde el inicio el algoritmo intenta realizar una identificación que se pueda usar en el controlador, pero se generan muchas oscilaciones, ya que toma tiempo obtener buenas soluciones. Este tiempo depende de la forma en que se hayan elegido los parámetros; además al existir ruido que se da en el momento de la adquisición de datos, pueden existir individuos en la población que si bien hacen que la salida real sea igual a la salida estimada, no servirán en futuras iteraciones. Aquí es donde es importante el proceso de selección y de mejora constante que ofrece el algoritmo genético de

Chu Beasley; con este proceso se irán mejorando todas las soluciones de la población, hasta que se llegue a un momento en que aunque el ruido haga que se cambie de una solución a otra, todas estas son de buena calidad y hacen que el sistema se comporte de manera adecuada; esto puede ser llamado un proceso de aprendizaje el cual toma un tiempo, pero después de tener una población con buenos individuos, el algoritmo responderá de mejor manera y para cuando se requiera volver a usar el algoritmo para el mismo sistema, se puede tomar dicha población como punto de inicio.

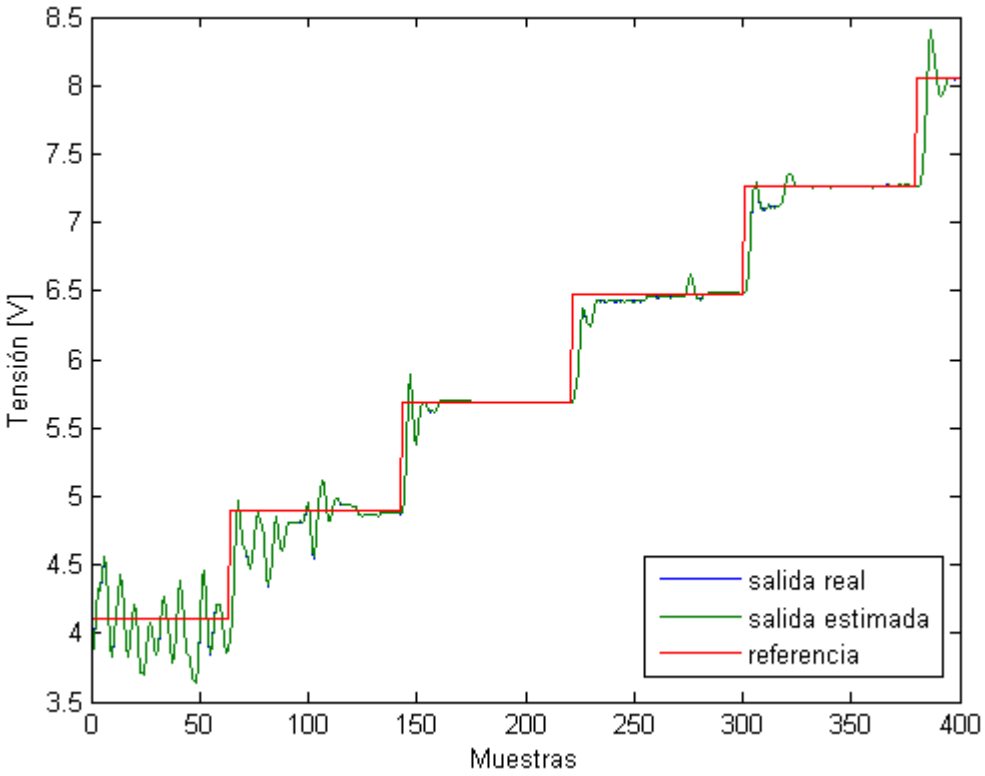

Fig. 26 Identificación y control de posición angular

Como se observa en la figura 26, después de que la población esté compuesta por soluciones de buena calidad y aunque el algoritmo de identificación cambie la solución casi que en cada iteración, la mayoría de ellas son buenas por lo que se logra un buen control.

Los parámetros del algoritmo de identificación y control fueron los siguientes:

Número de individuos en la población: 50.

Iteraciones del algoritmo genético por iteración de control: 15.

Periodo de muestreo: 0.1s

Individuos por torneo (selección): 5

Tasa de recombinación: 100%.

Tasa de mutación: 10%.

Polos del controlador: todos en 0.2.

Como se dijo anteriormente, la población se tuvo que reducir debido a que el algoritmo no alcanzaba a ejecutarse en el tiempo que debía, aun así, el algoritmo presenta un buen comportamiento.

Por último, se muestra en la figura 27 como responde este algoritmo ante una perturbación:

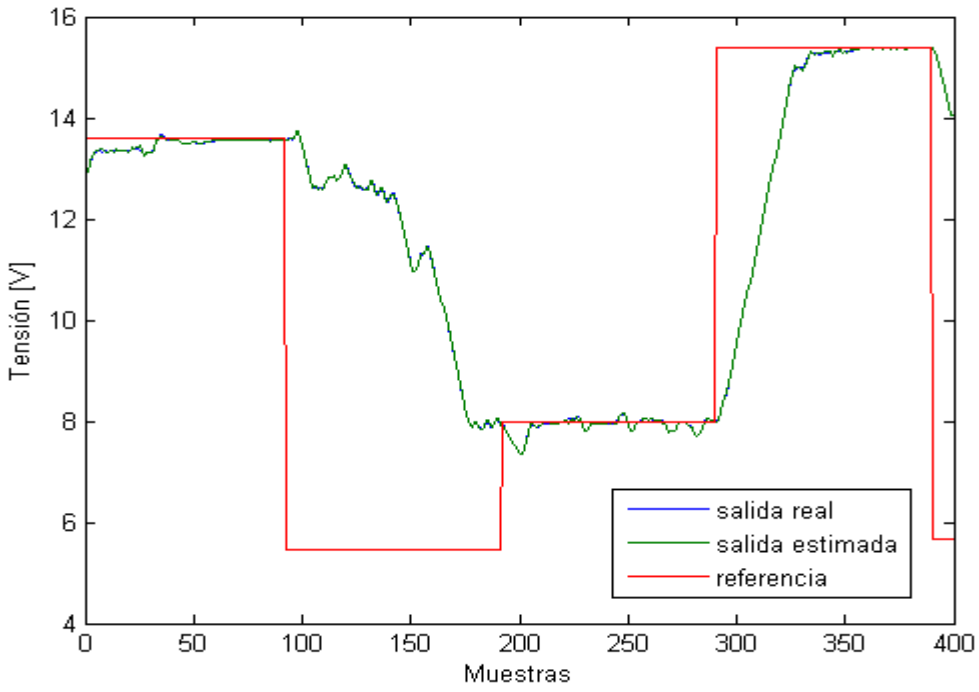

Fig. 27 Perturbación en control de posición angular

Alrededor de la muestra 100 se genera una perturbación, la cual el algoritmo logra superar. Después de tener una población con soluciones de buena calidad, se puede tomar como punto de inicio para la siguiente ejecución del algoritmo y de esta forma llegar de forma más rápida a un estado en que los resultados sean estables.

5. CONCLUSIONES

- Aunque se cree que los algoritmos genéticos se utilizan principalmente de manera off-line, se ha demostrado que con ajustes, pueden ser lo suficientemente rápidos para diversos tipos de problemas on-line.

- La identificación con algoritmos genéticos presenta buenos resultados ante perturbaciones de poca duración, ya que si la población está compuesta por soluciones de buena calidad, al terminar la perturbación, el algoritmo podrá llevar de forma rápida el sistema al estado deseado.

- Al tener un amplio conocimiento de la planta a identificar y controlar, el algoritmo genético se puede ajustar de una forma más adecuada y obtener mejores resultados.

- La unión entre la identificación con algoritmos genéticos y un controlador en espacio de estados con observador puede traer problemas, debido a la variación de los parámetros que entrega el algoritmo genético iteración tras iteración; como el observador funciona de forma iterativa, al haber un cambio de modelo del sistema en cada iteración, no permite que se pueda realizar la estimación de las variables de estado.

- Ya que el algoritmo genético se presta para ser modificado de muchas maneras, se presenta como trabajo futuro pensar en otros tipos de codificación y de ajuste de parámetros para lograr obtener mejores resultados.

- Como trabajo futuro se puede pensar en utilizar otra técnica metaheurística para implementar un algoritmo de identificación, como por ejemplo enjambre de partículas.

6. BIBLIOGRAFÍA

[1] R. Gallego Rendon, A. Escobar Zuluaga y E. Toro Ocampo, TÉCNICAS METAHEURÍSTICAS DE OPTIMIZACIÓN, PEREIRA, 2008.

[2] E. Passarge, Genética Texto y Atlas, Panamericana, 2010.

[3] G. C. Goodwin y K. S. Sin, Adaptative Filtering Prediction and Control, Dover Publications, 2009.

[4] D. Giraldo y E. Giraldo, TEORÍA DE CONTROL DIGITAL, Pereira: Produmedios, 2012.

[5] D. J. Arredondo Arteaga, «USO DE LA TÉCNICA DE CONTROL POR PLANOS DESLIZANTES APLICADA A SISTEMAS,» PEREIRA, 2014.

[6] G. Feng y L. Rogelio, Adaptative Control Systems, Newnes, 1999.

[7] V. Bovál, J. Böhm, J. Fessl y J. Machácek, Digital self-tuning Controllers, Springer, 2005.