IDENTIFICACIÓN DE SISTEMAS NO LINEALES MEDIANTE EL MÉTODO DE OPTIMIZACIÓN DE LA GOTA DE AGUA VIRTUAL INTELIGENTE

DANIEL EDUARDO DÁVILA LOBO

ALEJANDRO ERNESTO RUTTO MORA

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERIAS FÍSICO-MECÁNICAS

ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES

BUCARAMANGA

2014

IDENTIFICACIÓN DE SISTEMAS NO LINEALES MEDIANTE EL MÉTODO DE OPTIMIZACIÓN DE LA GOTA DE AGUA VIRTUAL INTELIGENTE

DANIEL EDUARDO DÁVILA LOBO

ALEJANDRO ERNESTO RUTTO MORA

TRABAJO DE GRADO PRESENTADO COMO REQUISITO PARCIAL PARA OPTAR AL TÍTULO DE INGENIERO ELECTRÓNICO

DIRECTOR

RODRIGO CORREA CELY,

INGENIERO QUÍMICO, PHD

CODIRECTOR

IVAN AMAYA CONTRERAS

INGENIERO ELECTRÓNICO, PHD (C)

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS

ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES

BUCARAMANGA

2014

CONTENIDO

pág.

LISTADO DE FIGURAS

pág.

LISTADO DE TABLAS

pág.

RESUMEN

Título: IDENTIFICACIÓN DE SISTEMAS NO LINEALES MEDIANTE EL MÉTODO DE OPTIMIZACIÓN DE LA GOTA DE AGUA VIRTUAL INTELIGENTE*

Autores: DANIEL EDUARDO DÁVILA LOBO, ALEJANDRO ERNESTO RUTTO MORA**

Palabras Clave: Motor DC, Runge-Kutta, IWD Continuo, Optimización, Simulación, No Lineal.

Descripción:

Este trabajo de grado aborda el proceso de identificación de sistemas, el cual se basa en la determinación de parámetros de un modelo matemático de sistema concreto, con los datos de entrada y salida observados. También, como ejemplo demostrativo, la selección de un modelo de motor DC que incluye el efecto de torque de fricción no lineal, que aparece cuando arranca o cambia su dirección de positiva a negativa y viceversa. El método numérico de RUNGE-KUTTA 4 se utiliza como herramienta para la simulación de la respuesta del modelo matemático del motor DC, obteniendo así los datos de entrada y salida de referencia. Además, se presenta la verificación del algoritmo de gotas de agua virtuales inteligente continuo con funciones “benchmark”. Por último, la identificación del motor es realizada minimizando la función de error, que relaciona la señal de referencia con la señal de salida del modelo identificado, a través de un script que utiliza el algoritmo de gota de agua virtual inteligente continuo.

Inicialmente se realiza la simulación del modelo de un motor DC previamente identificado para obtener los datos de referencia, entonces la identificación con el algoritmo de las gotas de agua virtuales inteligentes se realiza detallando el proceso, y como última instancia las respuestas son comparadas para confirmar los resultados de la identificación. Donde fue encontrado que es posible determinar los parámetros de un modelo no lineal de un motor DC.

* Trabajo de grado

** Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería Eléctrica, Electrónica y de Telecomunicaciones. Director: Rodrigo Correa Cely, Ingeniero Químico, PhD. Codirector: Ivan Amaya Contreras, Ingeniero Electrónico, PhD (c).

ABSTRACT

Title: IDENTIFICATION OF NON-LINEAR SYSTEMS USING THE INTELLIGENT WATER DROPS OPTIMIZATION METHOD*

Authors: DANIEL EDUARDO DÁVILA LOBO, ALEJANDRO ERNESTO RUTTO MORA**

Keywords: DC Motor, Runge-Kutta, Continuous IWD, Optimization, Simulation, Non-linear.

Description:

This bachelor thesis is aimed the system identification process, which is based on the parameters determination of a concrete mathematical model of a system, having the measured input and output data. Also, for showing purposes, the selection of a DC motor model which includes the non-linear friction torque effect, which appears when it starts or changes its direction from positive to negative and from negative to positive. The RUNGE KUTTA 4 numeric method is used as a DC motor simulation tool for getting the input and output data needed from a reference model. Furthermore, the continuous intelligent water drops algorithm's verification with benchmark functions is performed. At last, the motor's identification is performed by the minimization of an error function between the reference and the simulated data through a script using the non-conventional continuous intelligent water drops algorithm.

Initially a simulation of a DC motor model previously identified is performed to get the reference data, then the identification with the intelligent water drops algorithm is realized detailing the process and as last instance the outputs of the system are compared to confirm the identification results. Where it was found that it is possible to identify the parameters of the non-linear model of a motor DC.

* Bachelor Thesis

** Physico-mechanical Engineering Faculty. School of Electrical Engineering. Supervisor: Rodrigo Correa Cely, Chemical Engineer, PhD. Co-Supervisor: Ivan Amaya Contreras, Electronic Engineer, PhD (c)

INTRODUCCIÓN

El presente trabajo de grado contiene un campo de suma importancia de la ingeniería de control, como es el modelado de sistemas y la determinación de los parámetros del mismo. La identificación de sistemas es vital para el desarrollo y la implementación de controladores porque una correcta identificación permite predecir el comportamiento de los sistemas y aplicar correctamente un controlador y hacer que funcione de la manera deseada.

Se evalúa y propone un método alternativo de identificación, que es el empleo de un algoritmo metaheurístico basado en el comportamiento de enjambres, como la gota de agua virtual inteligente (IWD), en la determinación de los parámetros del modelo matemático de un motor DC, que describa el comportamiento de la fricción al arranque y en el cambio de dirección del motor. Disponiendo de un algoritmo que se puede implementar en una plataforma de programación, con el cual se analizará la eficacia de éste para la identificación de sistemas.

Primero se describe cómo se realiza el proceso de identificación de sistemas utilizando un algoritmo metaheurístico para que después se indique la obtención del modelo no lineal del motor DC y cómo se simulan los datos mediante método de Runge-Kutta 4 y cómo se le agrega ruido débilmente blanco a los datos de referencia. Luego se indica cómo trabaja el algoritmo mediante una descripción detallada de cada uno de los pasos que éste realiza y, antes de realizar las pruebas al algoritmo base se indica el procedimiento general del código para la identificación de sistemas para tener una idea de cómo se adaptó. Después se realiza una descripción detallada del algoritmo y se analizan los resultados mediante una serie de gráficas y tablas, mostrando al final cómo sería una interfaz gráfica para el usuario de este algoritmo.

1. PLANTEAMIENTO DEL PROBLEMA

El presente trabajo de grado en la modalidad de investigación aborda el problema de identificación de sistemas no lineales mediante una estrategia no convencional y más específicamente, empleando el algoritmo metaheurísticos de la IWD en la determinación de los parámetros de un modelo de un motor DC utilizado en aplicaciones de ingeniería electrónica.

2. ANTECEDENTES

En los últimos años han aparecido múltiples algoritmos de optimización inspirados en la naturaleza que han sido aplicados en diversos campos de la ciencia y la tecnología [1], [2]. Uno de ellos es el de “La Gota de Agua Inteligente” (IWD por sus siglas en ingles), aplicado a problemas como el del Vendedor Viajero (TSP) [1], el Enrutamiento de Vehículos (VRP) [3], la Planeación de Rutas Óptimas mediante la solución de laberintos [4], entre otros [5]. Basándose en el comportamiento de una gota de agua en el flujo de un río, y aprovechando las propiedades que ésta posee, se observan los cambios que realiza en su recorrido, moldeando una solución a los problemas, que para el caso de una gota de agua real sería encontrar un camino para llegar al mar.

3. JUSTIFICACIÓN

Por otro lado, la identificación de sistemas es la determinación de los parámetros de un modelo matemático que pueda aproximar la respuesta de éste a la respuesta real del sistema. Además, es el punto de partida para el estudio y solución de problemas prácticos, porque establece las relaciones existentes entre las variables de entrada y salida, para luego darle significado a estas relaciones [6]. El desarrollo de métodos para la identificación de sistemas es todavía un campo amplio de investigación, especialmente en la ingeniería de control, donde se requieren modelos matemáticos que muestren el comportamiento de los sistemas de forma precisa y bastante aproximada a la real, ya sea para diagnósticos, o para predicciones de la respuesta de éstos a distintos tipos de entradas [7]. Es una de las áreas del modelado y simulación que requiere un continuo desarrollo dada su importancia en el control de procesos cada vez más complejos, y al utilizar métodos no convencionales, como un algoritmo metaheurístico que haya arrojado buenos resultados anteriormente [8], [9], [10], se contribuye a la evolución de las técnicas que se utilizan para implementar soluciones de control como el denominado “Control por Modos Deslizantes (SMC)” [11], que para realizar el control correctamente se basa en la identificación precisa y acertada del sistema.

En éste trabajo de grado se abarcan las dos etapas del proceso de modelado de sistemas no-lineales que son consideradas de relativa dificultad: “*la selección de la estructura del modelo con cierto número de parámetros, y la selección del algoritmo para la estimación de los parámetros*” [7].

4. OBJETIVOS

4.1 OBJETIVO GENERAL

- Proponer y evaluar como alternativa la utilización del algoritmo de la gota de agua virtual inteligente (IWD) en la identificacion de sistemas no lineales.

4.2 OBJETIVOS ESPECÍFICOS

- Adaptar e implementar el algoritmo de optimización de la gota de agua virtual inteligente para la identificación de sistemas no lineales en una plataforma de programación comercial (Ver página 35).

- Seleccionar a titulo de ejemplo demostrativo el motor DC, donde se analiza la viabilidad de este algoritmo para la identificación de sistemas no lineales (Ver página 20).

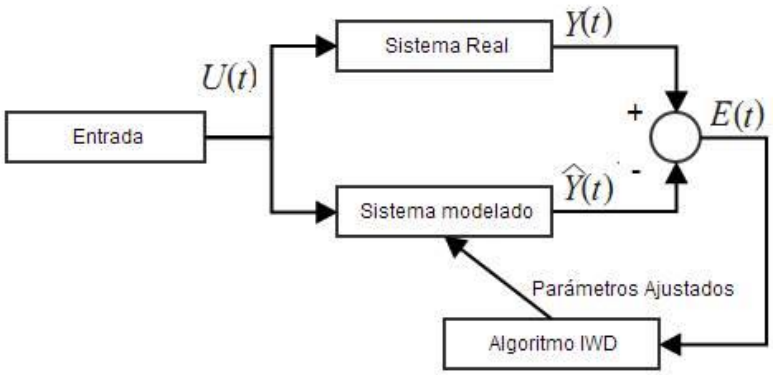
- Contrastar la precisión de las respuestas obtenidas para un sistema seleccionado con las reportadas en la literatura existente (Ver página 39).

5. PROCESO DE IDENTIFICACIÓN

A continuación se describen de forma general los algoritmos y métodos, utilizados como base para el desarrollo del código de identificación. También se muestra el proceso a través del cual se realiza el modelado del motor DC y cómo se obtienen los datos que se toman como referencia.

El proceso de identificación de sistemas se basa en el diagrama de bloques de la Figura 1, el cual consta de un bloque que genera la señal de entrada $U(t)$, que para éste caso es una señal de tensión en $[V]$; un sistema real o de referencia, que es el motor DC o un modelo previamente identificado en la literatura [12]; la señal de salida $Y(t)$ que entrega el sistema real o de referencia, que es la velocidad angular del motor en (rad/s) y los datos de referencia que se utilizarán para la identificación; el modelo de sistema al cual se le determinarán los parámetros, la salida del modelo a identificar $\hat{Y}(t)$, que también es una velocidad angular; una señal de error $E(t)$ que viene dada por la resta de la salida de referencia y la del modelo a identificar; y el bloque de identificación con el algoritmo de la IWD que ajusta los parámetros del modelo a identificar dependiendo de la señal de error.

Figura 1. Diagrama de bloques del proceso de identificación de un sistema.



Una de las etapas de la identificación de sistemas, como se mencionó anteriormente, es escoger un modelo adecuado para representar el sistema. Sin embargo, no se puede utilizar un modelo tradicional tal como la función de transferencia, que sería el modelo del motor DC de segundo orden lineal, porque debe tenerse en cuenta el efecto de la fricción no lineal al arrancar el motor o al cambiar de dirección. Con un método numérico como Runge-Kutta 4 se realizan las simulaciones del modelo de motor de la literatura que se ha identificado en trabajos

previos [12], y con el algoritmo objeto de estudio de este trabajo de grado, se hayan los parámetros.

5.1 MODELADO DEL MOTOR DC

Generalmente, el modelo de los motores DC está dado por un sistema de dos ecuaciones lineales, las cuales no tienen en cuenta la fuerza de fricción que se ejerce en el eje del motor cuando está detenido, situándose en una zona muerta cuando la señal de entrada se acerca a cero. El identificar y modelar esta zona es ventajoso, por ejemplo, en el uso de servomotores, donde se puede mejorar su precisión y desempeño.

El modelo clásico de un motor (Figura 2) se rige por las ecuaciones Ec. 1 y Ec. 2, correspondientes a la ecuación de malla y a la ecuación de movimiento respectivamente, donde U es la tensión en los terminales de la armadura, i la corriente de armadura, R_a es la resistencia de armadura, L_a la inductancia de armadura y K_e es el coeficiente de tensión del motor dado en $V \cdot s/rad$, K_t es el coeficiente del torque del motor dado en $N \cdot m/A$, J el momento de inercia del motor y B el coeficiente de fricción $(N \cdot m \cdot s/rad)$.

$$U = R_a \cdot i + L_a \frac{di}{dt} + K_e w \qquad Ec.\ 1$$

$$K_t \cdot i - Bw = J \frac{dw}{dt} \qquad Ec.\ 2$$

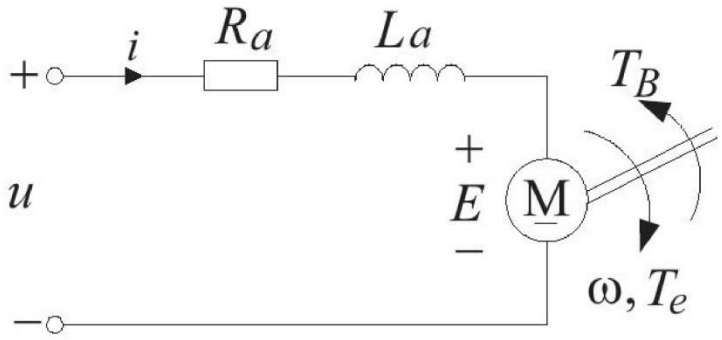
A partir de las ecuaciones Ec. 1 y Ec. 2 se llega a las ecuaciones de estado que definen el sistema (Ec. 3, Ec. 4 y Ec. 5), que constan de dos variables: corriente (i) y velocidad angular (w).

$$J\dot{w} = -Bw + K_t i \qquad Ec.\ 3$$

$$L_a \dot{i} = -K_e w - R_a i + u \qquad Ec.\ 4$$

$$y = w \qquad Ec.\ 5$$

Figura 2. Modelo de motor DC.



Fuente [12].

Al introducir el efecto de fricción no lineal del torque para modelar la zona muerta del motor mediante la Ec. 6, las ecuaciones de estado quedan descritas a través del conjunto de ecuaciones formado por la Ec. 7 y la Ec. 8 [12].

$$T_f = T_c * signo(w) + (T_s - T_c)e^{-\alpha|w|}signo(w) \qquad Ec.\ 6$$

$$J\dot{w} = -Bw + K_t i - T_c signo(w) - (T_s - T_c)e^{-\alpha|w|}signo(w) \qquad Ec.\ 7$$

$$L_a i = -K_e w - R_a i + u \qquad Ec.\ 8$$

El arreglo y reemplazo de variables conduce a las ecuaciones Ec. 9 y Ec. 10. De donde $K_1 = B/J;\ K_2 = K_t/J;\ K_3 = R_a/L_a;\ K_4 = K_e/L_a;\ K_5 = 1/L_a;\ K_6 = T_c/J;\ K_7 = (T_s - T_c)$ y $K_8 = \alpha$.

$$\dot{w} = -K_1 w + K_2 i - K_6 signo(w) - K_7 e^{-K_8|w|}signo(w) \qquad Ec.\ 9$$

$$i = -K_4 w - K_3 i + K_5 u \qquad Ec.\ 10$$

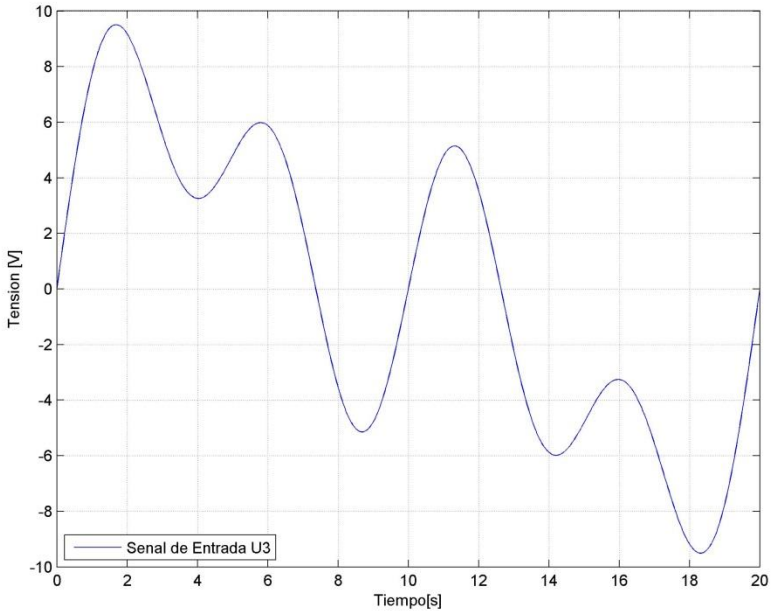
5.1.1 Simulación del modelo (Runge-Kutta 4)

Para obtener la respuesta del sistema, se hace necesario el uso de un método numérico debido a la naturaleza no lineal del modelo. Es posible utilizar el método

de Runge-Kutta 4 en esta aplicación utilizando una adaptación para la solución de sistemas de ecuaciones [13].

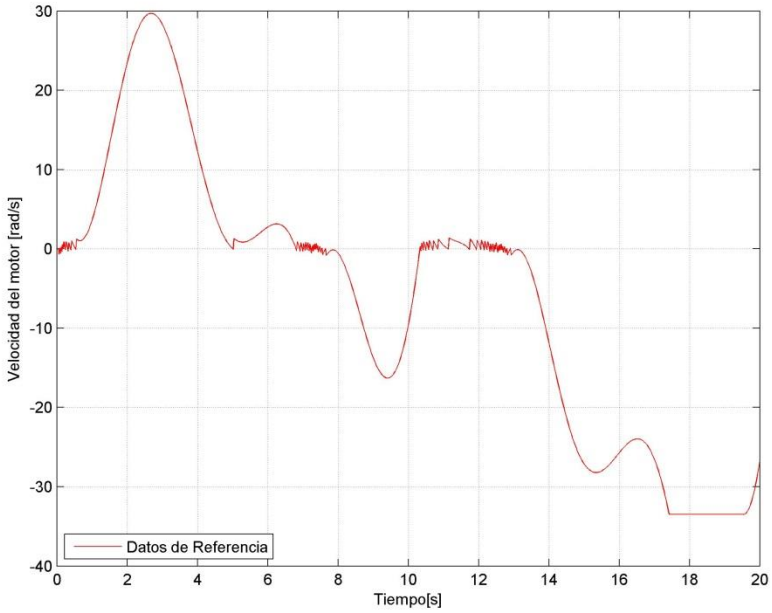
Se copia el sistema de ecuaciones (el modelo) con las constantes K_n que se obtuvieron en trabajos anteriores [12], y se aplica el método numérico con una entrada sinusoidal de distintos valores de amplitud y frecuencia (Figura 3) para obtener la respuesta de éste (Figura 4). Aquí se observan variaciones en la salida cada que ésta se acerca a cero, que no deben confundirse con ruido, sino que son cruces por cero que se tienen, y que si el incremento entre los puntos sucesivos es muy pequeño la simulación puede subir o bajar hasta el infinito y no mostrar la salida del modelo correspondiente. Por tanto, son variaciones inherentes al método de simulación empleado cuando la salida del modelo contiene una gran cantidad de cruces por cero.

Figura 3. Señal de entrada para comparar el resultado del método de simulación utilizado con la salida entregada en [12].



Fuente: [12].

Figura 4. Resultado de la simulación del modelo sin ruido mediante el método de RUNGE-KUTTA 4.



5.1.1.1 Adición del ruido

Con el fin de tener datos de referencia de entrada-salida que se asemejen a la realidad se debe agregar un error que refleje, bien sea ineficiencia de un instrumento de medida, o un factor humano. Para esto se considera una componente aleatoria a manera de ruido débilmente blanco [14], dentro de un rango determinado. Para aplicarlo, basta simplemente con tomar el vector de datos y multiplicarlo (elemento a elemento) por un vector de error, cuyos elementos corresponden a uno más o menos el margen de error deseado (4%, 8%, 15% y 20% para esta investigación). El resultado es una señal de salida con una incertidumbre proporcional a la amplitud de la respuesta (Figura 5), que se denominará “Datos de Referencia”. Son precisamente éstos datos los que se tomarán como datos de velocidad de referencia para determinar si fue o no posible la identificación del modelo del motor DC, que deben ser muy parecidos a los obtenidos por [12] (Figura 6).

En la Figura 5 se observa cómo el ruido afecta de manera principal los picos de la señal y la parte saturada de la respuesta, diferente a los cruces por cero y la parte cercana a cero, donde, por ser proporcional, no se ve tan afectada y responde de manera casi idéntica a la respuesta del modelo sin ruido.

Figura 5. Resultado de la simulación del modelo con ruido débilmente blanco de ±8% mediante el método de RUNGE-KUTTA 4.

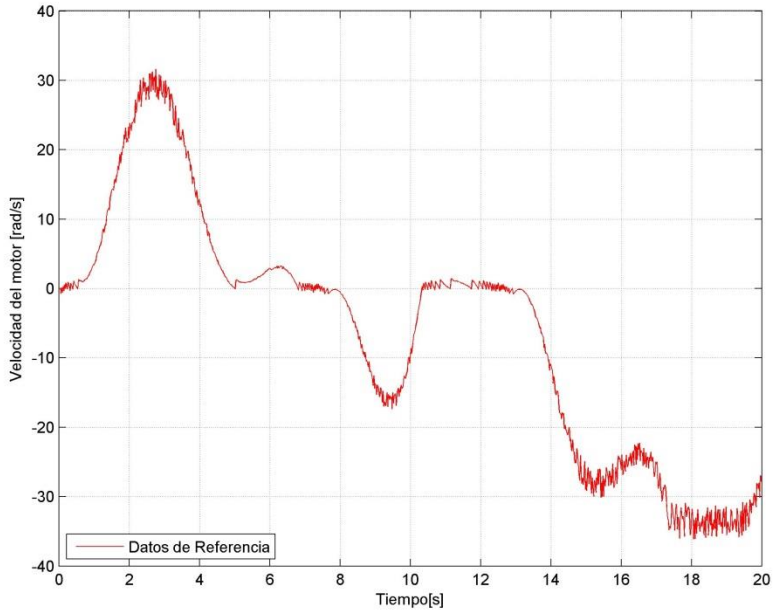
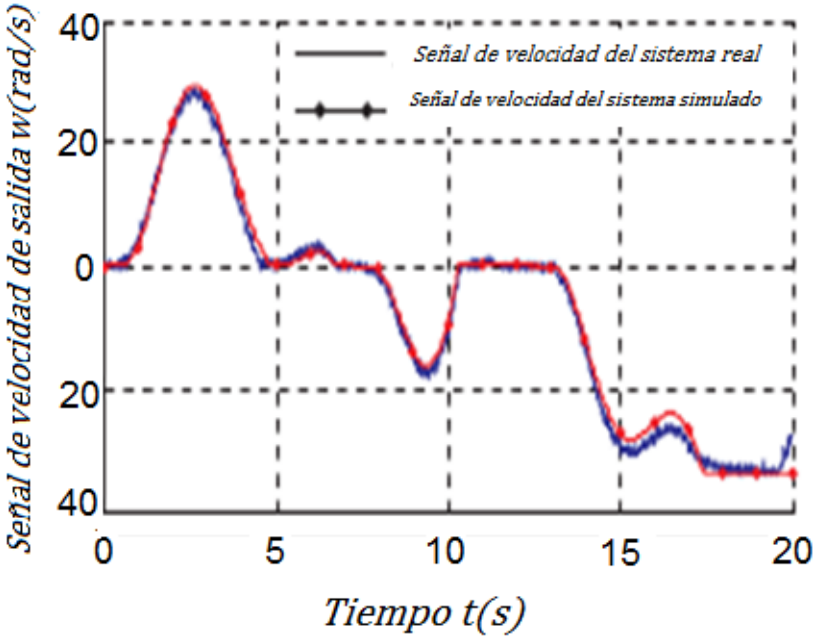


Figura 6. Salida experimental y del modelo previamente identificado en [12].

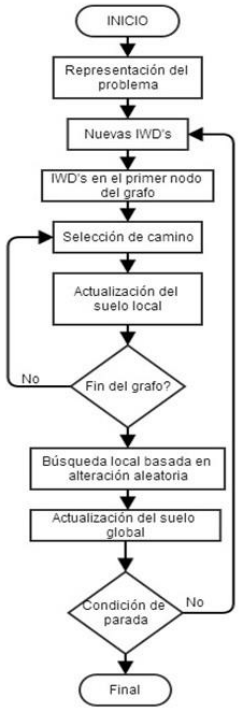


Fuente: [12]

5.2 ALGORITMO IWD CONTINUO [2]

A continuación se muestran los pasos para optimizar una función, y en la Figura 7 se presenta el diagrama de flujo del algoritmo.

Figura 7. Diagrama de flujo del algoritmo IWD Continuo.



Fuente [2]

5.2.1 Representación del problema

Considerando que se quiera maximizar o minimizar una función $f(\boldsymbol{X})$: $R^M \rightarrow R$ que tiene M componentes y $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_M]^T$ como argumentos de entrada. Se construye un "grafo" con $M * (P + 1)$ nodos y $2 * \big(M * (P + 1)\big)$ "caminos directos", que las IWD usan para construir soluciones para la función a optimizar, donde P representa la precisión en número de bits que dividen el rango del espacio de búsqueda de cada una de las componentes de la función objetivo. Entonces, cada $P + 1$ nodos consecutivos en el "grafo" representan una cadena de binarios con P número de bits. Es de notar que inicialmente, todos los "caminos" del "grafo" tienen la misma cantidad de suelo.

Cada IWD empieza su recorrido en el primer nodo y lo termina en el nodo $P + 1$. Un "camino directo" $e_{i,i+1}(k)$, que conecta el nodo i al $i + 1$, está "atado" al bit k que

puede ser uno o cero. Como resultado, habrán dos “caminos dirigidos” conectando el bit k_i al k_{i+1}.

5.2.2 Actualización del suelo local

Cuando una IWD abandona un nodo i usando un camino $e_{i,i+1}(k)$ para llegar al nodo $i+1$, el suelo de la IWD, $soil^{IWD}$, y el suelo del camino usado, $soil\left(e_{i,i+1}(k)\right)$, se actualizan así:

$$soil\left(e_{i,i+1}(k)\right) = 1.1 * soil\left(e_{i,i+1}(k)\right) - 0.01 * \Delta soil\left(e_{i,i+1}(k)\right) \quad \textit{Ec. 14}$$

$$soil^{IWD} = soil^{IWD} + \Delta soil\left(e_{i,i+1}(k)\right) \quad \textit{Ec. 15}$$

$$soil\left(e_{i,i+1}(k)\right) = 0.001 \quad \textit{Ec. 16}$$

Por lo tanto, un camino con menos suelo permite a una IWD ganar más velocidad que uno con más suelo. Una vez que todas las IWD lleguen al último nodo del “grafo” del problema, se aplica un algoritmo de búsqueda local a las soluciones construidas por las IWD.

5.2.3 Búsqueda local basada en el canje

Aquí, las soluciones creadas por las IWD se someten a un canje. Específicamente, se selecciona al azar un camino $e_{i,i+1}(k)$ de una solución y se reemplaza por el otro camino que conecta al nodo i al $i+1$, si el cambio mejora el valor final de la solución. Este proceso se repite un número determinado de veces, aquí 100 para cada solución. Es de notar que esta búsqueda local se aplica a todas las soluciones creadas en la interacción actual.

5.2.4 Actualización global del suelo

Luego de la búsqueda local, se ejecuta la actualización global de suelo en los caminos de la solución de la mejor iteración del algoritmo IWD.

Primero, se encuentra la solución de la mejor iteración T^{IB} entre las soluciones de todas las IWD al final de la iteración actual. T^{IB} es la solución con la mejor calidad entre todas las soluciones de las IWD. Entonces, el suelo de caminos que forman la solución T^{IB} se actualiza por:

$$soil\left(e_{i,i+1}(k)\right) = min\left(max\left(Tempsoil\left(e_{i,i+1}(k)\right), MinSoil\right), Maxsoil\right) \quad \forall\, e_{i,i+1}(k) \in T^{IB} \quad Ec.\ 17$$

Donde

$$Tempsoil\left(e_{i,i+1}(k)\right) = 1.1 * soil\left(e_{i,i+1}(k)\right) - 0.01 * \frac{soil_{IB}^{IWD}}{M*P} \qquad \forall\, e_{i,i+1}(k) \in T^{IB} \qquad Ec.\ 18$$

La actualización global de suelo está limitada por los intervalos $[MinSoil, MaxSoil]$ para prevenir la subutilización de cualquier camino. Por otra parte, $soil_{IB}^{IWD}$ representa la cantidad de suelo que la IWD con la solución de la mejor iteración ha reunido de los caminos del "grafo". En otras palabras, $soil_{IB}^{IWD}$ implícitamente representa la calidad de la solución de la mejor iteración T^{IB}.

Una vez terminado este paso, se ha realizado una iteración del IWD-CO y comienza otra iteración con nuevas IWD pero con los mismos caminos. Este proceso continúa hasta un número máximo de iteraciones u otro criterio de parada.

6. PROCEDIMIENTO GENERAL DEL CÓDIGO PARA LA IDENTIFICACIÓN DE SISTEMAS

Para la correcta y completa identificación del motor DC, se requiere identificar el motor tanto para la dirección positiva como para la negativa, debido a la diferencia de las respuestas en cada uno de los sentidos de giro [12].

Para la dirección positiva se cargan los datos de referencia del sistema, se definen los rangos de los parámetros K_n y también los del algoritmo. Inmediatamente se inicializan las variables dinámicas creando una matriz de ceros de $Niwd\ x\ (Nbits + 1)$ llamada Matriz de Gotas de Agua, donde $Niwd$ es el número de gotas de agua virtuales utilizadas, $Nbits$ el número de bits de precisión del algoritmo, y la columna extra es donde se guarda el valor de la cantidad de suelo recogida por la IWD al final de cada recorrido. Después de tener la matriz completa, los bits se convierten a decimales, se halla el valor de los parámetros y se simula el sistema mediante un método numérico (RUNGE-KUTTA 4), para luego, con la respuesta obtenida y con los datos de referencia, tener los datos que se necesitan para hallar el valor de la función objetivo (Ec. 19). Seguidamente se realiza la búsqueda local mediante el canje y el mismo proceso para obtener datos de la función objetivo, se escoge la mejor solución, se guarda y se sigue con la siguiente iteración, donde se vuelve a inicializar la Matriz de Gotas de Agua. El algoritmo sigue haciendo lo anterior hasta que cumple una de las condiciones de parada y retiene la mejor solución que haya encontrado.

$$F(iwd) = \frac{1}{t_1 - t_0} \int_{t_0}^{t_1} |Y_m(t) - Y(t)| dt \qquad Ec.\ 19$$

7. PRUEBAS DEL ALGORITMO

Para verificar que el código del algoritmo está escrito de manera correcta se realizaron varias pruebas con distintas funciones ya establecidas [15] a las que se le conocen sus valores óptimos. Se utilizaron los mismos parámetros que en [2] para el algoritmo, a excepción del número máximo de iteraciones y el criterio de parada, que para las pruebas con dos dimensiones se escogía dependiendo de cada función [2], y para ocho dimensiones se escogió como $1 * 10^{-3}$ para todas ellas. Las pruebas fueron efectuadas en un PC de escritorio con procesador AMD PHENOM X4 9550 de 2,2GHz, memoria RAM de 6GB DDR2 a 667MHz y un disco duro HITACHI de 500GB SERIALATA II FSB 300MB/s a 7200rpm.

En la Tabla 1 se muestran las funciones que se utilizaron para probar el algoritmo. Se realizaron cinco pruebas para cada una de ellas utilizando dos dimensiones, y otras cinco pruebas utilizando ocho dimensiones, exceptuando f_3 debido a su naturaleza estrictamente bidimensional. Los mejores resultados obtenidos, el valor promedio y el tiempo empleado para encontrar la respuesta, son todos presentados en la Tabla 2 para dos dimensiones, y en la Tabla 3 para ocho dimensiones.

Tabla 1. Funciones prueba utilizadas.

Función	Rango	Dimensión	Mínimo conocido
$\boldsymbol{f_1(X) = \sum_{i=0}^{n-1} x_i^2}$	$\boldsymbol{-5,12 < x_i < 5,12}$	$\boldsymbol{n \geq 1}$	$\boldsymbol{f_1(0) = 0}$
$\boldsymbol{f_2(X) = \sum_{i=0}^{n-1} \lvert x_i \rvert + \prod_{i=0}^{n-1} \lvert x_i \rvert}$	$\boldsymbol{-10 < x_i < 10}$	$\boldsymbol{n \geq 1}$	$\boldsymbol{f_2(0) = 0}$
$\boldsymbol{f_3(x_1, x_2) = -cos(x_1) * cos(x_2)\ e^{(-(x_1-\pi)^2-(x_2-\pi)^2)}}$	$\boldsymbol{-100 \leq x_i \leq 100}$	$\boldsymbol{n = 2}$	$\boldsymbol{f_3(\pi, \pi) = -1}$
$\boldsymbol{f_4(X) = \frac{1}{4000} \sum_{i=0}^{n-1} x_i^2 - \prod_{i=0}^{n-1} cos\left(\frac{x_i}{\sqrt{(i)}}\right) + 1}$	$\boldsymbol{-600 \leq x_i \leq 600}$	$\boldsymbol{n \geq 2}$	$\boldsymbol{f_4(0) = 0}$
$\boldsymbol{f_5(X) = 10n + \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i)\right]}$	$\boldsymbol{-5,12 < x_i < 5,12}$	$\boldsymbol{n \geq 1}$	$\boldsymbol{f_5(0) = 0}$

Tabla 2. Resultados pruebas del algoritmo para dos dimensiones.

Función	Mínimo conocido	Mejor resultado	x_i	Promedio	Tiempo promedio [s]
f_1	$f_1(0) = 0$	$4.349 * 10^{-16}$	$-1.073 * 10^{-8}; 1.788 * 10^{-8}$	$5.576 * 10^{-16}$	83.312
f_2	$f_2(0) = 0$	$5,5879 * 10^{-8}$	$-1.397 * 10^{-7}; 4.191 * 10^{-7}$	$7,9162 * 10^{-8}$	100,3989
f_3	$f_3(\pi, \pi) = -1$	-0.9999	$3.1415; 3.1416$	-0.9999	128.465
f_4	$f_4(0) = 0$	0.1952	$1.397 * 10^{-7}; 1.397 * 10^{-7}$	0.1952	382.752
f_5	$f_5(0) = 0$	0	$-1.192 * 10^{-9}; 1.192 * 10^{-9}$	0	131.649

Tabla 3. Resultados pruebas del algoritmo para ocho dimensiones.

Función	Mínimo conocido	Mejor resultado	x_i	Promedio	Tiempo promedio [s]
f_1	$f_1(0) = 0$	$4.074 * 10^{-4}$	$0.006;\ 0.0062;\ -0.0109;\ -0.0069;\ -0.0029;\ -0.0052;\ 0.0046;\ -0.0105;$	$6.761 * 10^{-4}$	625.037
f_2	$f_2(0) = 0$	$3.442 * 10^{-4}$	$1 * 10^{-4} * (0.424; 0.028;\ 1.236;\ 0.611;\ -0.146;\ -0.473;\ -0.421;\ -0.102)$	$7.092 * 10^{-4}$	2107.921
f_4	$f_4(0) = 0$	0.5471	$1 * 10^{-5} * (1.355; -0.210;\ -0.265;\ 1.802;\ 0.042;\ 0.014;\ -1.802;\ -0.461)$	0.5471	6898.691

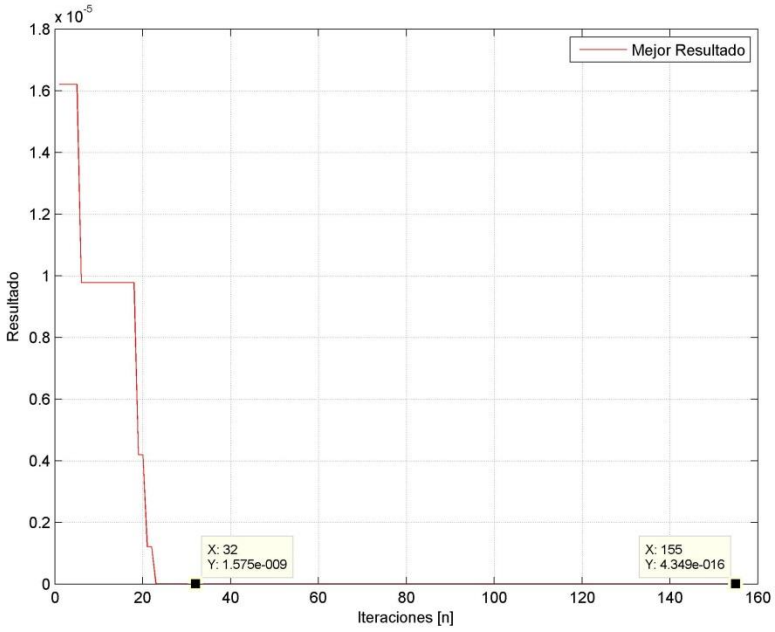
Tabla 3. (Continuación).

Función	Mínimo conocido	Mejor resultado	x_i	Promedio	Tiempo promedio [s]
f_5	$f_5(0) = 0$	$4.717 * 10^{-5}$	$1 * 10^{-4} * (-2.734; 3.325; 0.562; 0.423; -0.475; -0.097; 1.719; -1.251)$	$3.963 * 10^{-4}$	2823.314

7.1 PRUEBA FUNCIÓN 1

La llamada "Primera función de De Jong" tiene como área de búsqueda el hipercubo comprendido por $-5{,}12 \leq x_i \leq 5{,}12$ y su valor mínimo es $f_1(x_i) = 0$ para todo $x_i = 0, i = 1, \ldots, n$ [15]. Se observa como converge el algoritmo para dos dimensiones en la Figura 8.

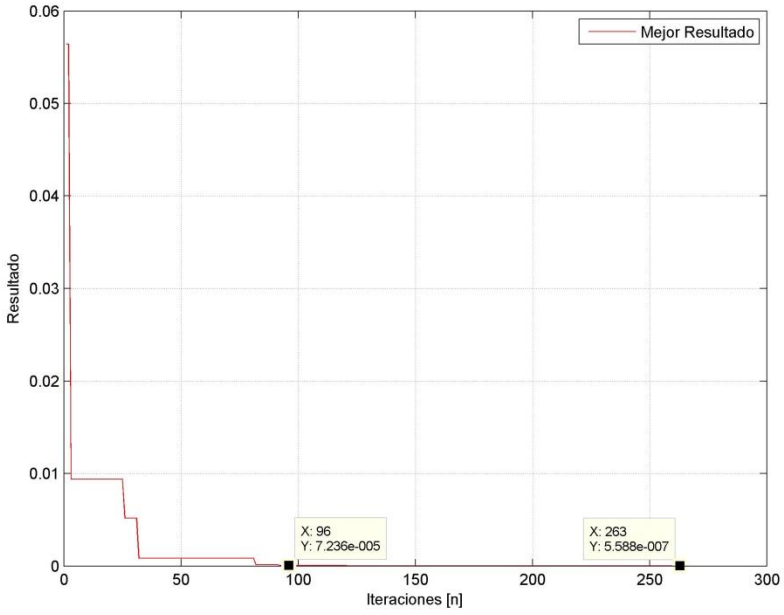
Figura 8. Convergencia del algoritmo de la IWD Continua a $f_1(x_1, x_2)$.



7.2 PRUEBA FUNCIÓN 2

La función 2 está definida por una sumatoria de valores absolutos más una productoria de los valores absolutos. Tiene como área de búsqueda el hipercubo comprendido por $-10 \leq x_i \leq 10$ y su mínimo es $f_2(x_i) = 0$ para todo $x_i = 0, i = 1, \ldots, n$ [15]. Se observa como converge el algoritmo para dos dimensiones en la Figura 9.

Figura 9. Convergencia del algoritmo de la IWD Continua a $f_2(x_1, x_2)$.



7.3 PRUEBA FUNCIÓN 3

La función de "Easom", es una función de prueba unimodal definida únicamente por dos variables y tiene un solo mínimo global. Tiene como área de búsqueda el área del cuadrado comprendido por $-100 \leq x_i \leq 100$ y su mínimo es $f_3(x_1, x_2) = -1$ para $x_i = \pi, i = 1,2$ [15]. Se observa como converge el algoritmo para dos dimensiones en la Figura 10.

7.4 PRUEBA FUNCIÓN 4

La función de “Griewank” es una función con muchos mínimos locales uniformemente distribuidos. Tiene como área de búsqueda el hipercubo comprendido por $-600 \leq x_i \leq 600$ y su mínimo es $f_4(x_i) = 0$ para todo $x_i = 0, i = 1, \ldots, n$ [15]. Se observa como converge el algoritmo para dos dimensiones en la Figura 11.

Figura 10. Convergencia del algoritmo de la IWD Continua a $f_3(x_1, x_2)$.

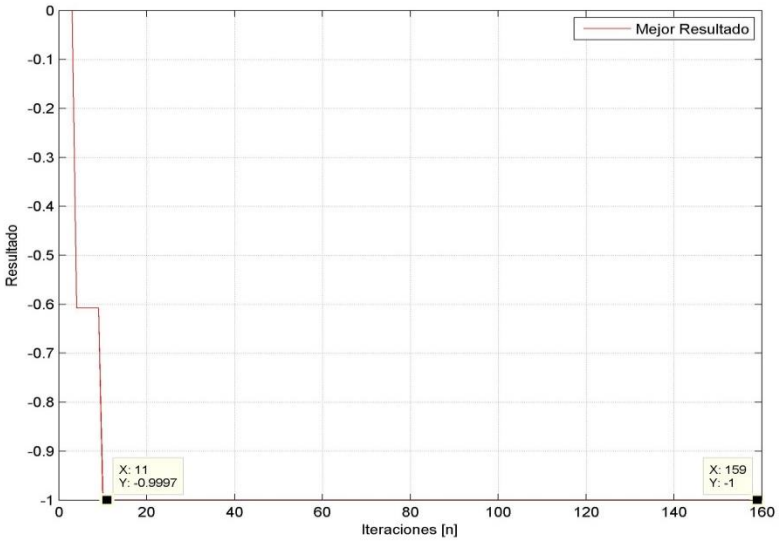
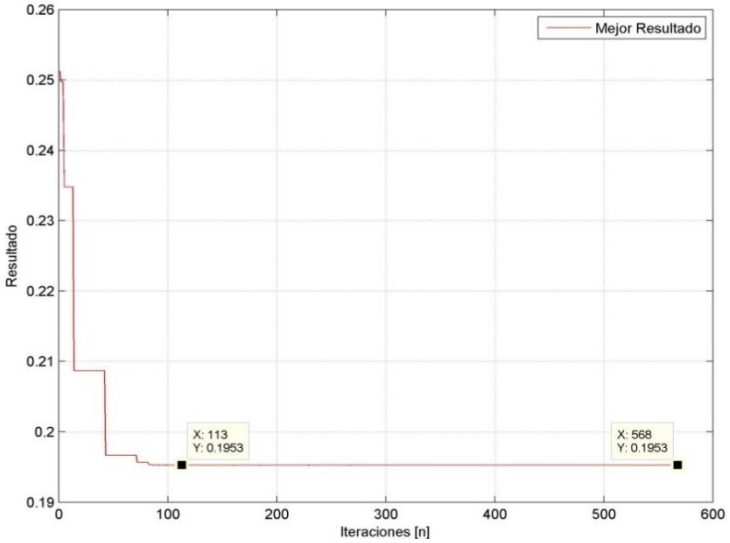


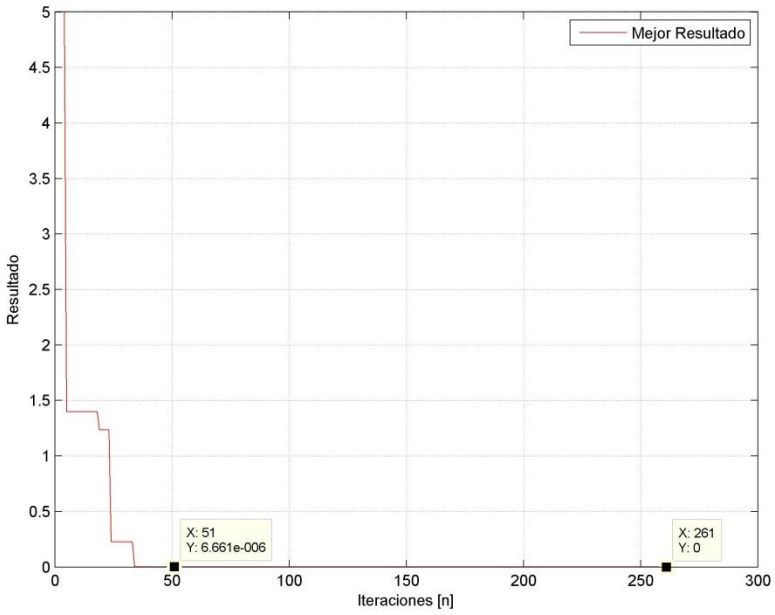
Figura 11. Convergencia del algoritmo de la IWD Continua a $f_4(x_1, x_2)$.



7.5 PRUEBA FUNCIÓN 5

La función de prueba de “Rastrigin” es multimodal ya que tiene una modulación coseno que produce mínimos locales frecuentes, aunque distribuidos de manera regular. El área de búsqueda es el hipercubo comprendido por $-5.12 \leq x_i \leq 5.12$ y su valor mínimo es $f_5(x_i) = 0$ para todo $x_i = 0, i = 1, \ldots, n$ [15]. La convergencia del algoritmo con la función de “Rastrigin” para dos dimensiones se observa en la Figura 12

Figura 12. Convergencia del algoritmo de la IWD Continua a $f_5(x_1, x_2)$.



De la Tabla 1 se observa que los rangos de búsqueda de las funciones de prueba unimodales f_1 a f_3 van aumentando. Asimismo, en la Tabla 2 se nota que el tiempo promedio que se demora el algoritmo en encontrar una solución óptima aumenta y que la precisión disminuye. Cabe resaltar que este efecto es natural, debido a que los agentes de búsqueda están explorando un espacio mayor, y por tanto existen más posibilidades que deben ser analizadas. Éste comportamiento también se puede observar con las funciones multimodales f_4 y f_5, y con laTabla 3, donde además se ve que al incrementar el número de dimensiones el tiempo también aumenta, aunque en este caso se debe también al aumento de argumentos de entrada, que se traduce en más nodos y “caminos directos” (Ver página 25), lo que aumenta el esfuerzo computacional al tener que utilizar una matriz más grande.

8. DESCRIPCIÓN DEL ALGORITMO DE IDENTIFICACIÓN

En la Tabla 4 se muestran los parámetros que se utilizaron para desarrollar el código del algoritmo con una breve descripción de su función y el tipo de variable que utiliza cada parámetro:

Tabla 4. Lista de parámetros utilizados en el código.

PARÁMETROS	DESCRIPCIÓN	TIPO
Rinf	Rango inferior de los parámetros K	Vector
Rsup	Rango superior de los parámetros K	Vector
delt	Diferencia entre Rsup y Rinf para ajustar la resolución de búsqueda	Vector
minsoil	Mínima cantidad de suelo en el recorrido	Double
maxsoil	Máxima cantidad de suelo en el recorrido	Double
Npar	Número de parámetros a calcular en el modelo	Int
Nbits	Número de bits utilizados para establecer la precisión	Int
Niwd	Número de gotas virtuales de agua utilizados (Mínimo el mismo número de parámetros)	Int
param	Contador cuando se utilizan más de una gota virtual por parámetro	Int
es, ev, ro	Parámetros de actualización del suelo	Double
Deltsoil	Cantidad de suelo recogida por cada gota virtual	Double
initsoil	Cantidad de suelo con que comienzan los caminos	Double
soil	Matriz que contiene la cantidad de suelo de cada camino	Matriz
Gsoil	Matriz para la actualización de la cantidad de suelo en cada camino	Matriz
Fsoil	Matriz para la actualización de la cantidad de suelo en cada camino	Matriz
itermax	Número máximo de iteraciones a realizar	Int
Nmut	Número de mutaciones a realizar	Int
Z	Vector de Guardado de la mejor solución	Double
res	Variable intermedia para convertir datos binario a decimal dentro del rango	Double
X	Solución de la iteración	Double
Zo	Vector de guardado del error en la iteración	Double
Zom	Vector de guardado error mutada	Double
bitrand	Posición de los bits que se mutarán	Vector

Xm	Mejor solución	Double

Tabla 4. (Continuación).

PARÁMETROS	DESCRIPCIÓN	TIPO
Zc	Vector de registro de los errores en las iteraciones	Double
Y	Resultado de la simulación del modelo	Double
IWDs	Matriz de gotas de agua	Double
iter	Contador de iteraciones	Int
Vn	Peso de los bits en IWDs	Int
iwd	Contador para las gotas de agua virtuales	Int
edge	Contador del valor del bit	Int
den	Denominador para el parámetro Fsoil	Double
P	Vector de probabilidad	Double
parbin	Variable intermedia para convertir de bits de IWDs a decimal	String
dirpos	Función de simulación del modelo	Función
Biwd	Matriz con los vectores en binario de la mejor solución	Matriz
tempsoil	Parámetro para la actualización global del suelo	Double
count	Contador para detención del algoritmo	Int

El código tiene una estructura definida: se cargan los datos medidos de entrada, salida y tiempos de muestreo, se inicializan las variables y los parámetros del algoritmo, se simula con los resultados obtenidos, se actualizan las variables del algoritmo, y se repite el procedimiento hasta que se halle la solución óptima.

Los datos medidos deben estar contenidos en un archivo .mat, el cual contenga la entrada aplicada al sistema guardada en una variable “u”, el tiempo de cada una de las muestras en la variable “t” y la salida del sistema en la variable “w”.

Cuando se inicializan las variables, primero se plasman los valores mínimos del rango de búsqueda de cada parámetro K_n en el vector “Rinf”, luego los máximos en el vector “Rsup”, de allí en adelante el programa calcula los deltas de cada uno de los rangos de los parámetros del modelo. Luego se establecen los parámetros y se crean las variables estáticas propias del algoritmo para asignar los bloques de memoria en el PC y aumentar la velocidad de procesamiento de éste.

A continuación se abre un primer bucle con una sentencia “for”, que funcionará como el contador de iteraciones y también como criterio de parada cuando este alcance el número máximo de iteraciones establecido anteriormente. Dentro de este bucle se inicializa la matriz “IWDs” en ceros, para que al iniciar cada iteración se realice el procedimiento con gotas de agua virtuales nuevas [2]. El número de filas que contiene esta matriz representa el número de gotas virtuales de agua, que como mínimo es una por parámetro, las columnas son el peso de cada bit, por lo tanto, una fila completa de la matriz “IWDs” está llena con los bits que representan un número decimal con el cual se obtendrá un parámetro, y la columna final guarda la cantidad de suelo que recoge cada IWD al final de su recorrido.

Después, para llenar la matriz se abren dos bucles consecutivos, uno que cuente el número de bits, es decir, indique en qué columna va, y otro que cuente la gota de agua virtual, o lo que es lo mismo, en qué fila de la matriz. Dentro del último bucle se realiza el proceso de la selección del camino, o selección del bit correspondiente, hallando primero las probabilidades de que sean cero o uno y guardándolas en el vector de probabilidades “P”. Luego, utilizando el vector “P” se le asigna aleatoriamente el valor del bit a la gota de agua, al terminar de asignar los bits de cada gota de agua, se cierra el segundo bucle y se abre nuevamente un nuevo bucle con el que se actualiza de manera local la matriz “soil”, dependiendo de cuales fueron los bits seleccionados, y también, la columna final de la matriz “IWDs”. Al finalizar, el primer bucle sigue con la siguiente columna de la matriz “IWDs”, es decir, con los bits para el siguiente peso hasta llenar toda la matriz y terminar el bucle.

Al tener la matriz “IWDs” llena se procede a convertir en decimales los bit que representan cada fila. Mediante un bucle se convierte fila por fila los números (ceros y unos) en “string”, y luego en decimal, se guarda en un vector “res”, se multiplica por el ancho de búsqueda de ese parámetro, se divide entre la resolución y el resultado se suma con el valor del rango inferior para tener el valor del parámetro que se guardará en el vector “X”, realizando este procedimiento para cada una de las gotas de agua (filas de la matriz “IWDs”).

Con todos los valores de los parámetros del modelo hallados, se pasa a la evaluación de la función. Se halla la respuesta del modelo con los parámetros del vector “X” a través de las funciones “dirpos” o “dirneg”, que utilizan el método de RUNGE-KUTTA 4 para simular la respuesta del modelo, y la salida del modelo se guarda en un vector “Y”. Luego, junto con los datos medidos se halla el error (la función a optimizar) en el modelo guardando éste en el vector “Zo”.

Ahora empieza el proceso de búsqueda local o canje, para lo cual se abren dos bucles consecutivos, el primero con un contador para detener la búsqueda en un número de canjes indicados, y el siguiente con el contador de gotas de agua. Para

cada fila se escoge un número de columna al azar, estos se guardan en el vector “bitrand”, luego, se realiza el cambio de bit a través de una sentencia “if” y se repite el procedimiento para convertir todos los bit de ésta nueva matriz “IWDs” en decimales y hallar los parámetros canjeados, simulando otra vez el modelo con los nuevos datos y hallando el error del modelo guardándolo en “Zom”. Entonces, se comparan los errores obtenidos y si el error obtenido en el canje es mayor al obtenido sin éste, se vuelven a cambiar los bits en la matriz “IWDs”. De lo contrario, la matriz se deja igual pero se hace el valor de “Zo” igual al de “Zom”, hasta que termine de hacer el número de canjes establecido.

Al terminar las iteraciones correspondientes al canje, se verifica si la solución obtenida es la mejor de todas las soluciones encontradas hasta ese momento, que están guardadas en el vector “Z”. De ser el caso, se hace el valor de “Z” igual a la solución hallada y se guarda la matriz “IWDs” en una matriz idéntica con nombre “Biwd”, se le realiza la debida conversión a decimal y luego los parámetros del modelo se guardan en el vector “Xm”. En el caso de que la solución no resulte mejor que una de las anteriores, el valor del “Z” queda igual al de la última vez.

Después se procede con una actualización global de la matriz “soil” de manera parecida a como se realizó de manera local, solo que esta vez utilizando la matriz “Biwd”, una variable llamada “tempsoil” y la última columna de la matriz “IWDs”, que debe corresponder con la última columna de la matriz “Biwd”.

Después de lo anterior, se ubican las condiciones de parada del algoritmo, donde se tomaron tres criterios más de parada aparte del número máximo de iteraciones. Un valor mínimo de error, que para este trabajo y a criterio propio se estimó, una variación mínima de la mejor solución y que la mejor solución no cambie en un determinado número de iteraciones. Luego de todo lo anterior se cierra el bucle de iteraciones y se repite el procedimiento para una iteración nueva.

9. RESULTADOS DE LA IDENTIFICACIÓN DEL MOTOR

Se realizaron cinco pruebas para la identificación de cada dirección de giro del motor, y para cada prueba se registraron los valores obtenidos del error, la cantidad de iteraciones requerida para llegar a un resultado y el tiempo que tarda el algoritmo. Estos datos se observan en la Tabla 5, Tabla 6, Tabla 7, Tabla 8 y Tabla 9 además de los valores promedio de cada una de las características antes mencionadas. Las pruebas fueron realizadas para los datos de entrada y salida antes de agregar el ruido débilmente blanco (para observar el error cuando los datos son uniformes), y después de haber agregado el ruido. Se observa en todas las tablas que los errores que entrega el algoritmo siempre son menores cuando se identifica el sentido de giro negativo. Un factor podría ser la proporcionalidad del ruido débilmente blanco que no afecta a éstos tanto como a los datos para el sentido de giro positivo, donde estos últimos alcanzan mayores valores absolutos en la respuesta.

Tabla 5. Comportamiento del algoritmo en la identificación del motor DC utilizando datos de entrada-salida sin considerar la presencia de ruido.

Dirección Negativa						
Prueba	1	2	3	4	5	Promedio
Error [rad/seg]	0.0379	0.0301	0.0103	0.0461	0.0288	0.0306
No. De iteraciones	898	1015	894	852	1099	951.6
Tiempo [s]	2162.159	2489.222	2204.329	2092.425	2711.595	2331.946
Dirección Positiva						
Prueba	1	2	3	4	5	Promedio
Error [rad/seg]	0.1206	0.0742	0.0877	0.1702	0.0829	0.1071
No. De iteraciones	1509	947	734	1059	760	1001.8
Tiempo [s]	3722.623	2379.434	1735.142	2610.314	1822.801	2454.063

Tabla 6. Comportamiento del algoritmo en la identificación del motor DC utilizando datos de entrada-salida considerando la presencia de ruido débilmente blanco del ±4%.

Dirección Negativa						
Prueba	1	2	3	4	5	Promedio
Error [rad/seg]	0.2848	0.3123	0.2846	0.2928	0.2898	0.29286
No. De iteraciones	693	774	843	832	514	731.2
Tiempo [s]	1657.964	1881.819	2039.200	1986.356	1262.153	1765.498
Dirección Positiva						
Prueba	1	2	3	4	5	Promedio
Error [rad/seg]	0.3297	0.3189	0.3106	0.3094	0.3462	0.3229
No. De iteraciones	800	990	714	965	814	856.6
Tiempo [s]	1954.859	2453.620	1751.336	2440.934	2009.863	2122.122

Tabla 7. Comportamiento del algoritmo en la identificación del motor DC utilizando datos de entrada-salida considerando la presencia de ruido débilmente blanco del ±8%.

Dirección Negativa						
Prueba	1	2	3	4	5	Promedio
Error [rad/seg]	0.5378	0.5130	0.5295	0.5318	0.5147	0.52536
No. De iteraciones	724	1205	653	860	711	830
Tiempo [s]	1817.792	2974.757	1544.655	2089.817	1735.5106	2032.506
Dirección Positiva						
Prueba	1	2	3	4	5	Promedio
Error [rad/seg]	0.6099	0.6124	0.6311	0.6333	0.6049	0.61832
No. De iteraciones	636	691	763	919	768	755.4
Tiempo [s]	1526.308	1679.264	1860.189	2242.539	1925.561	1846.772

Tabla 8. Comportamiento del algoritmo en la identificación del motor DC utilizando datos de entrada-salida considerando la presencia de ruido débilmente blanco del ±15%.

Dirección Negativa						
Prueba	1	2	3	4	5	Promedio
Error [rad/seg]	0.9787	0.9777	0.9776	0.9863	0.9748	0.97902
No. De iteraciones	1019	566	726	554	1065	786
Tiempo [s]	1803.227	1587.345	1325.120	1022.554	1910.975	1529.844
Dirección Positiva						
Prueba	1	2	3	4	5	Promedio
Error [rad/seg]	1.0592	1.0663	1.0880	1.0851	1.0871	1.07714
No. De iteraciones	821	676	411	710	915	706,6
Tiempo [s]	982.239	1116.711	875.013	1489.261	1234.769	1139.598

Tabla 9. Comportamiento del algoritmo en la identificación del motor DC utilizando datos de entrada-salida considerando la presencia de ruido débilmente blanco del ±20%.

Dirección Negativa						
Prueba	1	2	3	4	5	Promedio
Error [rad/seg]	1.2546	1.2505	1.2511	1.2471	1.2524	1.2511
No. De iteraciones	579	643	626	958	569	675
Tiempo [s]	950.422	1253.790	1044.521	2154.277	1152.265	1311.055

Tabla 9. (Continuación).

Dirección Positiva						
Prueba	**1**	**2**	**3**	**4**	**5**	**Promedio**
Error [rad/seg]	**1.4621**	**1.4522**	**1.4475**	**1.4477**	**1.4353**	**1.4490**
No. De iteraciones	**930**	**880**	**996**	**1157**	**755**	**943.6**
Tiempo [s]	**2036.263**	**1780.312**	**2095.078**	**2039.361**	**1991.552**	**1988.513**

Se muestran las gráficas donde se comparan los datos de referencia con ruido débilmente blanco de ±4% con los datos del modelo al realizar la identificación en la dirección positiva (Figura 14) para la entrada $U_1(t)$ (Figura 13), y en la negativa (Figura 16), para la entrada $U_2(t)$ (Figura 15). Aquí se puede observar que la salida del modelo identificado se encuentra dentro de la variación de los datos de referencia, efecto que se resalta en los picos de las respuestas, por lo que se puede decir que el modelo entrega unos datos de salida bastante precisos. Para la verificación del modelo completo se utilizaron las entradas U_3 (Figura 17), que es una señal de voltaje con diferentes frecuencias y amplitudes que cambian el sentido de giro del motor utilizada en el Capítulo 2.1.2 (Ver página 21), y U_4 (Figura 19), que alcanza las zonas de saturación del motor. En la Figura 18 y Figura 20 se observan las respuestas para las entradas U_3 y U_4, respectivamente, de los modelos identificados para cada uno de los sentidos de giro comparadas con los datos referencia, considerando un ruido del ±4%, y se ve que la velocidad de respuesta es un poco más rápida en el sentido positivo y más lenta en el sentido negativo que la respuesta de los datos de referencia, que es uno de los motivos por los cuales el modelo presenta error.

Figura 13. Entrada utilizada para identificar el modelo del motor DC en el sentido de giro positivo.

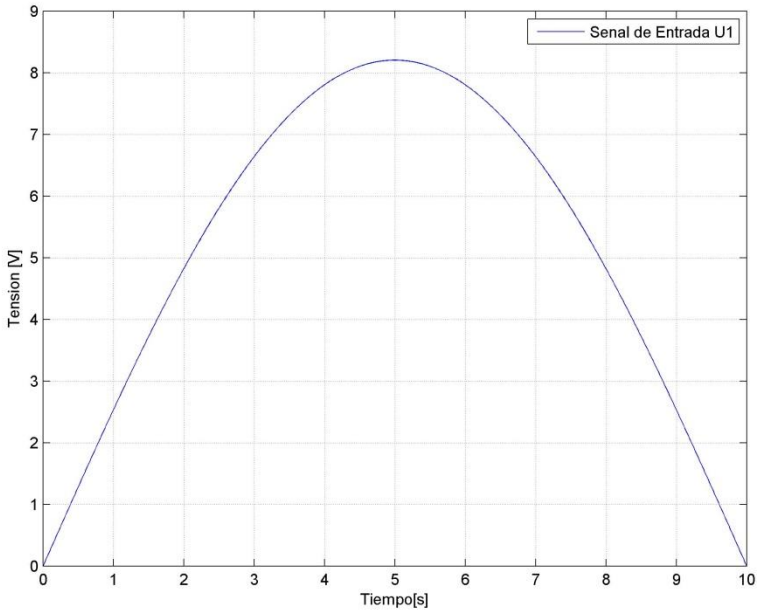


Figura 14. Resultado de la identificación de los parámetros de la dirección positiva del motor, considerando la presencia de ruido débilmente blanco del ±4%.

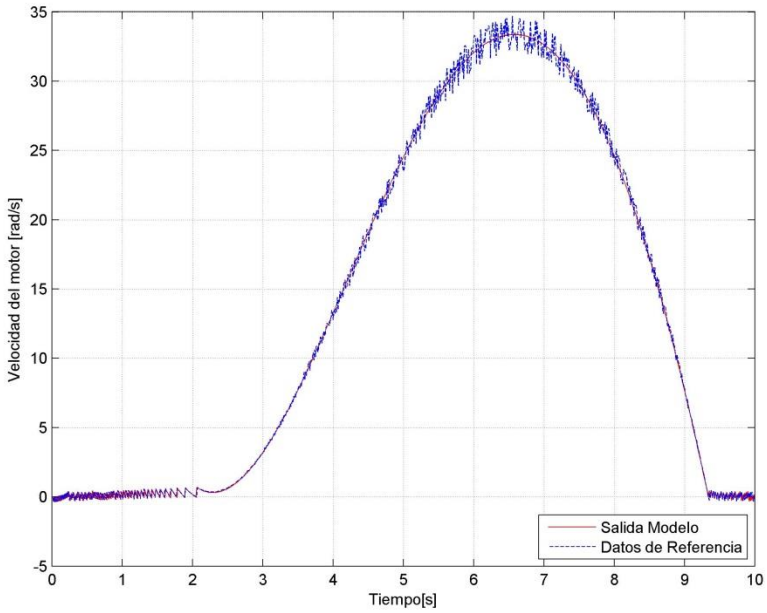


Figura 15. Entrada utilizada para identificar el modelo del motor DC en el sentido de giro negativo.

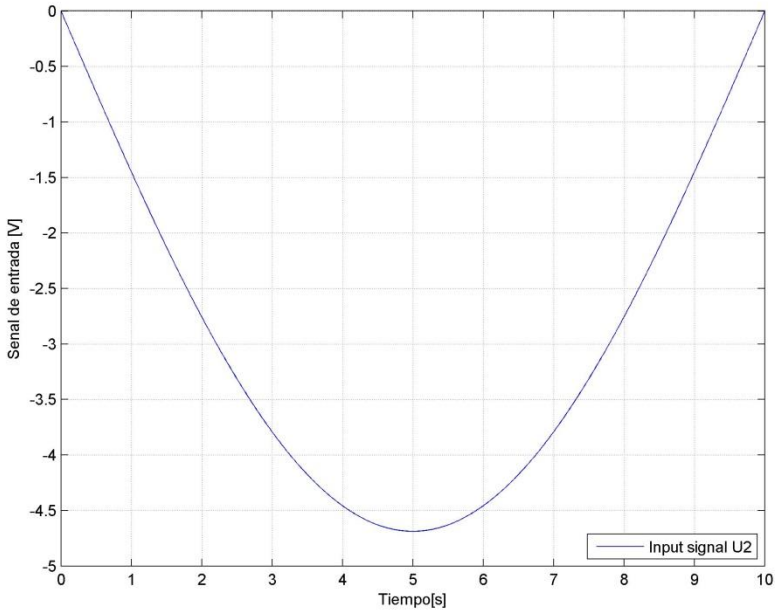


Figura 16. Resultado de la identificación de los parámetros de la dirección negativa del motor, considerando la presencia de ruido débilmente blanco del ±4%.

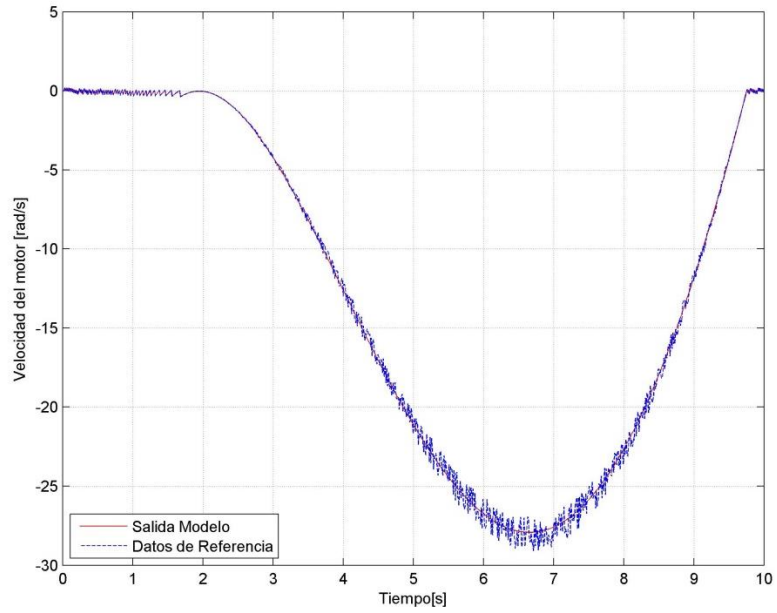
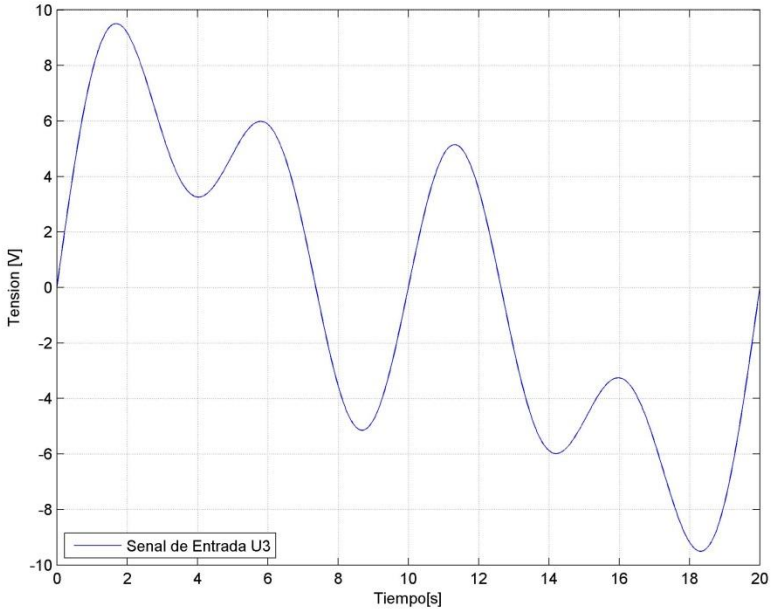


Figura 17. Señal de entrada U3 para prueba del modelo completo.



Fuente [12].

Figura 18. Datos de referencia con ruido débilmente blanco del ±4% y salida del modelo identificado a la señal de entrada U3.

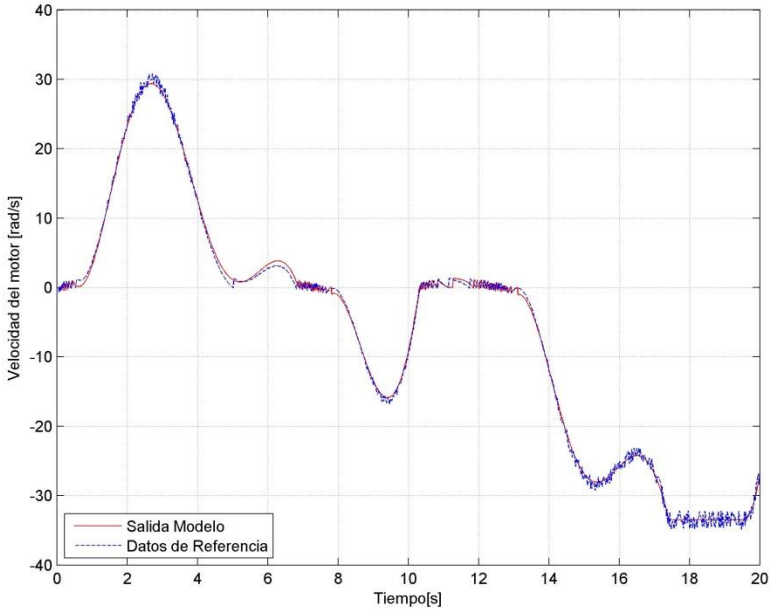
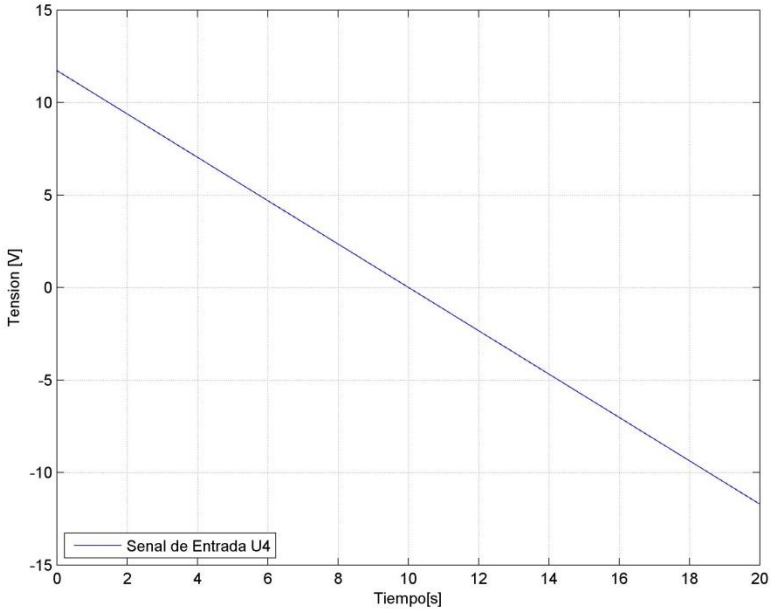
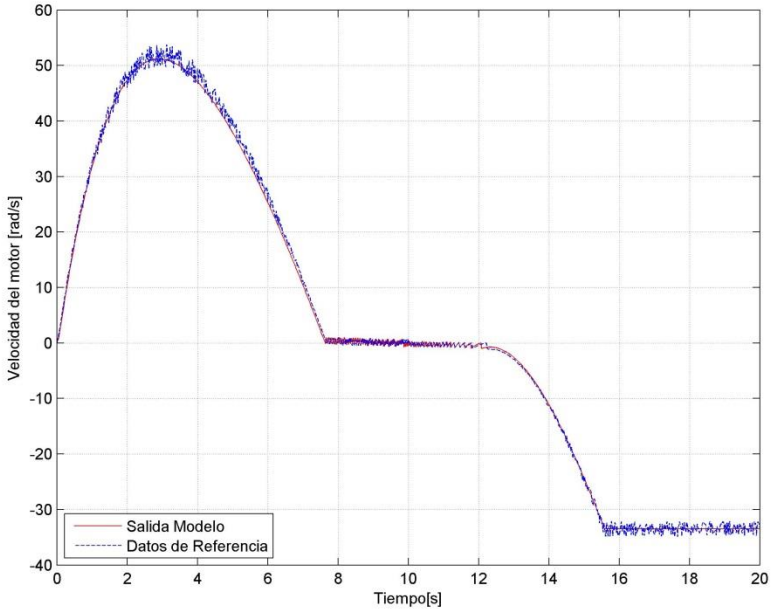


Figura 19. Señal de entrada U4 para prueba del modelo completo.



Fuente [12].

Figura 20. Datos de referencia con ruido débilmente blanco del ±4% y salida del modelo identificado a la señal de entrada U4.



Al comparar los errores promedio de la Tabla 5 y la Tabla 6 (sin ruido y con ruido débilmente blanco del ±4%, respectivamente) con el error obtenido por [16], encontramos que los primeros (sin ruido) son hasta 4 veces menores que los obtenidos con PSO y los segundos muestran un error de casi tres cuartos del entregado por PSO. Aun comparando el error de este último (PSO) con los obtenidos para un nivel de ruido del ±8% (Tabla 7), se encuentra que son parecidos, aunque mayores los obtenidos con el algoritmo de IWD. Por tanto, podemos decir que IWD permite identificar sistemas de manera bastante precisa, aun utilizando instrumentos que podrían ser bastante rústicos o poco precisos. Como muestra, se presentan también datos para niveles de ruido de ±15% (Tabla 8) y ±20% (Tabla 9), para los que fue posible identificar el sistema (Figura 21), pero con errores de más de un rad/s, por lo tanto, podría utilizarse también con instrumentos de medida poco precisos.

En la Tabla 10 se observan cada uno de los parámetros que el algoritmo arrojó para las cinco pruebas realizadas en la dirección negativa, y en la Tabla 11 para la dirección positiva, en la cual se evidencian los valores promedio y la desviación estándar de cada uno de ellos. Se tiene que los parámetros obtenidos difieren entre sí cada vez que se realiza una prueba, y esto puede deberse a que la función objetivo tenga varios mínimos locales en el rango de búsqueda, y a la naturaleza estocástica del algoritmo IWD.

Figura 21. Datos de referencia con ruido débilmente blanco del ±20% y del modelo identificado a la señal de entrada U3.

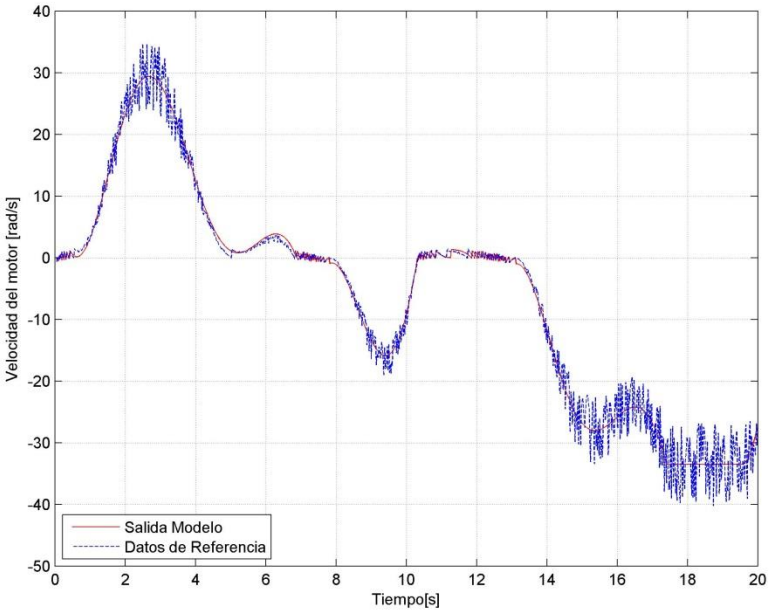


Tabla 10. Parámetros hallados en la identificación de la dirección negativa, considerando la presencia de ruido débilmente blanco del ±4%.

Dirección Negativa								
K	Prueba					Media	Desviación Estándar	Rango
	1	2	3	4	5			
1	0.3137	0.1796	0.0559	0.3150	0.1298	0.1988	0.1022	$[0-5]$
2	31.0164	24.3011	15.1053	30.3939	28.7391	25.9112	5.8909	$[0-50]$
3	55.2244	18.8731	55.0814	60.1012	59.8538	49.8268	15.6268	$[0-80]$
4	0.0588	0.2174	1.0181	0.1957	0.4598	0.3899	0.3396	$[0-30]$
5	13.5310	5.9788	27.4803	14.8264	15.5674	15.4768	6.9067	$[0-30]$
6	9.7631	2.0605	8.2458	1.7364	20.1172	8.3846	6.6898	$[0-30]$
7	11.3457	19.4313	12.6860	18.8165	0.2901	12.5139	6.9049	$[0-20]$
8	0.0008	0.0065	$1*10^{-3}$	0.0038	0.0046	$3*10^{-3}$	$2.4*10^{-3}$	$[0-0.01]$

Tabla 11. Parámetros hallados en la identificación de la dirección positiva, considerando la presencia de ruido débilmente blanco del ±4%.

Dirección Positiva								
K	Prueba					Media	Desviación Estándar	Rango
	1	2	3	4	5			
1	0.1920	0.1236	0.3050	0.1121	0.0176	0.1500	0.0953	$[0-5]$
2	30.2922	23.4376	21.4108	31.7035	15.0415	24.3771	6.0909	$[0-50]$
3	70.1325	77.4794	63.7126	69.9908	16.3429	59.5316	22.0301	$[0-80]$
4	0.5514	1.0577	0.3105	0.6517	0.5070	0.6156	0.2473	$[0-30]$
5	15.1817	22.0633	19.8735	14.6498	7.2119	15.7960	5.1241	$[0-30]$
6	14.3575	28.8314	20.7452	29.0654	14.8826	21.5764	6.4238	$[0-30]$
7	19.5906	6.2538	14.2213	5.7213	19.2817	13.0137	6.0477	$[0-20]$
8	0.0013	0.0092	0.0006	0.0022	0.0029	$3*10^{-3}$	$3.1*10^{-3}$	$[0-0.01]$

En la Tabla 12 se muestra entonces los resultados de la identificación para una de las pruebas.

Tabla 12. Resultados de la identificación de parámetros para las direcciones positiva y negativa.

K	Parámetros dirección positiva	Parámetros dirección negativa
1	0.1920	0.3137
2	30.2922	31.0164
3	70.1325	55.2244
4	0.5514	0.0588
5	15.1817	13.5310
6	14.3575	9.7631
7	19.5906	11.3457
8	0.0013	0.0008

10. INTERFAZ GRÁFICA PARA EL USUARIO

La interfaz para el usuario consta de un panel de parámetros en la parte izquierda (Figura 22) donde se muestran: los parámetros K_n del motor (para ambas direcciones); el respectivo error en la identificación; un botón llamado “Cargar Datos Dir”, ubicado arriba del panel de parámetros y el cual se utiliza para cargar el archivo .mat con los datos de referencia de entrada, salida y tiempo en el que se realiza la muestra; tres botones debajo del panel que corresponden con el botón “Id. Positivos”, para empezar la identificación de los parámetros positivos, el botón “Id. Negativos”, para realizar la identificación de los parámetros negativos, y el botón “Testing sys”, con el cual se prueba que los parámetros hallados sean fieles a la realidad. Para esto, se carga un archivo .mat con los datos de referencia de una entrada que cambie la dirección de giro del motor, de la respuesta real del motor a esa entrada, y de los tiempos de muestreo. Debajo de todo lo anterior, se muestra un panel con el modelo de motor utilizado (Figura 23), y ocupando gran parte de la interfaz se muestran dos gráficas (Figura 24), la gráfica de velocidad, que muestra los datos medidos y los datos de salida del modelo identificado, y la gráfica de función objetivo, que muestra cómo progresó el algoritmo.

Figura 22. Panel de parámetros y botones de la interfaz gráfica.

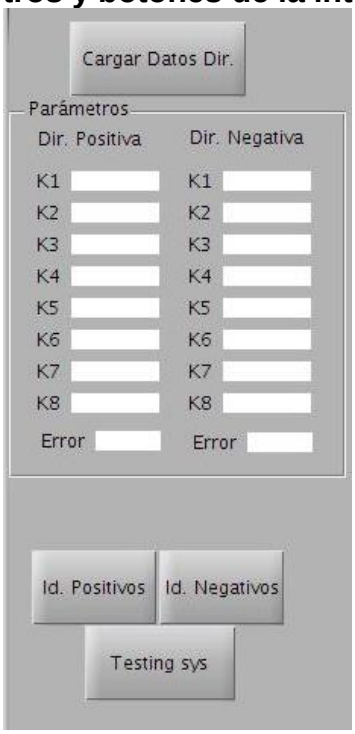


Figura 23. Panel con el modelo del motor DC.

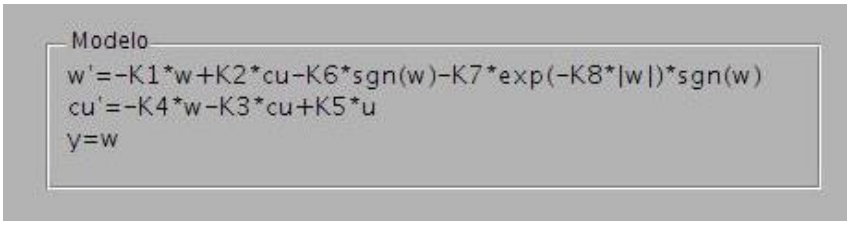
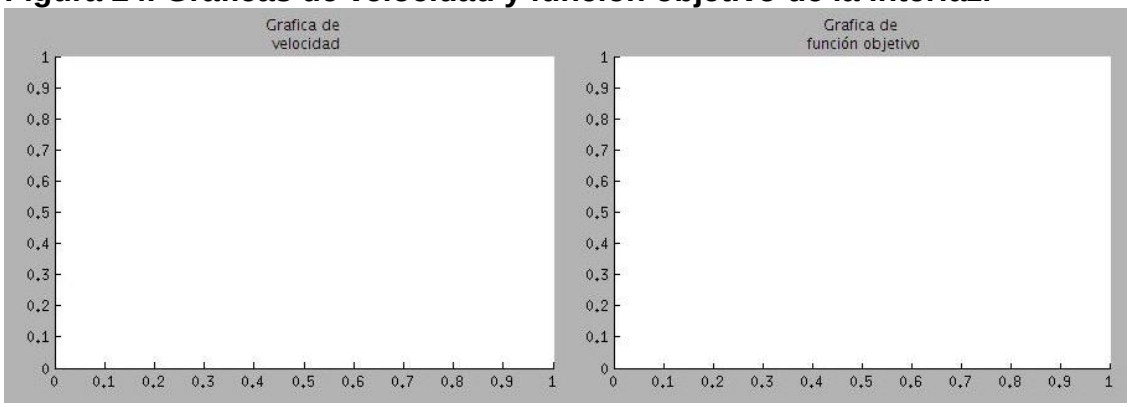


Figura 24. Gráficas de velocidad y función objetivo de la interfaz.



11. CONCLUSIONES

Se cumplieron satisfactoriamente todos los objetivos propuestos en el plan de trabajo de grado.

Es posible utilizar el algoritmo de IWD continuo para la identificación del modelo no lineal de un motor DC, por lo menos dentro de los parámetros considerados en esta investigación.

El tiempo requerido para encontrar una solución óptima para las funciones de prueba aumenta al tener un espacio de búsqueda más amplio, como se observa en la Tabla 2 y en la Tabla 3.

La precisión de la solución óptima para las funciones de prueba baja cuando hay un aumento en espacio de búsqueda debido a la disminución de la resolución que puede tomar la solución, como se observa en la Tabla 2 y en la Tabla 3.

El aumento en el número de dimensiones en las funciones de prueba produjo un aumento en el tiempo empleado para encontrar una solución óptima porque para cada dimensión se utiliza una gota de agua virtual, lo que se traduce en un aumento en la cantidad de filas de la matriz “IWDs”. Por lo tanto, se necesita más tiempo para salir de algunos de los bucles. Adicionalmente, las dimensiones extra aumentan el esfuerzo computacional requerido para evaluar la función objetivo.

Los errores fueron mayores cuando se realizó la identificación de la dirección positiva, dado que los datos de referencia de la dirección opuesta (negativa) eran menores y por tanto eran menos afectados por el ruido proporcional.

Para la planta y las señales de excitación analizadas, es posible obtener una respuesta adecuada del modelo aun con niveles de ruido del 20%, aunque se obtienen errores mayores a un rad/seg.

De la Tabla 10 y la Tabla 11 se observa que los parámetros más influyentes para la identificación del motor con este modelo son K_1 y K_4, pues tienen una desviación estándar baja respecto al espacio de búsqueda de cada parámetro.

Al utilizar el método de RUNGE-KUTTA 4 como herramienta para la simulación de modelos de sistemas que contengan cruces por cero, es posible que se presenten

variaciones cerca de cero que se pueden llegar a confundir con ruido, pero son un efecto inherente al método de simulación al utilizar un tamaño de paso relativamente grande.

12. RECOMENDACIONES

Modificar el código para que cada una de las gotas de agua opere de manera paralela a las otras y se mejore el tiempo en el que el algoritmo entrega la respuesta. Así también, la implementación en lenguajes de bajo nivel.

Realizar una comparación entre el algoritmo de la IWD continuo y otros del mismo tipo (Swarm-Based), para observar el consumo de recursos que conlleva el uso de este algoritmo y determinar según los resultados cual sería más económico de implementar, ya sea mediante software o hardware.

Se debería observar la influencia de utilizar más gotas de agua por cada parámetro, ¿Aumentaría la precisión?, ¿Qué tanto aumentaría el tiempo de proceso?, ¿Convergería más rápido el algoritmo?, ¿Cuánto cuesta en recursos computacionales el aumento de estas?.

Aunque la identificación arroja errores bastante bajos, los parámetros después de cada identificación no son los mismos, por lo que se recomienda un estudio sobre cuál es la implicación o el propósito de cada uno de los parámetros del modelo sobre la respuesta del sistema.

CITAS BIBLIOGRAFICAS

[1] H. Shah-Hosseini, “Problem solving by intelligent water drops,” *Evol. Comput. 2007. CEC 2007. IEEE Congr.*, pp. 3226–3231.

[2] H. Shah-Hosseini, “An approach to continuous optimization by the Intelligent Water Drops algorithm,” *Procedia - Soc. Behav. Sci.*, vol. 32, no. 2010, pp. 224–229, Jan. 2012.

[3] I. Kamkar, M.-R. Akbarzadeh-T, and M. Yaghoobi, “Intelligent water drops a new optimization algorithm for solving the Vehicle Routing Problem,” *Syst. Man Cybern. (SMC), 2010 IEEE Int. Conf.*, pp. 4142–4146.

[4] J. Arias and M. Mogollon, “Algoritmo de optimización gotas de agua virtuales inteligentes aplicado a la planeación de ruta óptima de un robot móvil,” Universidad Industrial de Santander (UIS), 2013.

[5] H. Shah-Hosseini, “Optimization with the nature-inspired intelligent water drops algorithm,” *Evol. Comput.*, no. October, 2009.

[6] X. Deng, “System Identification Based on Particle Swarm Optimization Algorithm,” *Comput. Intell. Secur. 2009. CIS ’09. Int. Conf.*, vol. 1, pp. 259–263.

[7] S. Garrido, L. Moreno, and C. Balaguer, “Identificación, estimación y control de sistemas no-lineales mediante RGO,” *Univ. Carlos III*, 1999.

[8] G. Liu, X. Xu, and F. Wang, “Identification of a kind of nonlinear system,” *Nat. Comput. (ICNC), 2011 Seventh Int. Conf.*, vol. 3, pp. 1730–1733.

[9] X. Xu, F. Wang, and F. Qian, “Study on Method of Nonlinear System Identification,” *Intell. Comput. Technol. Autom. (ICICTA), 2011 Int. Conf.*, vol. 1, pp. 944–947, Mar. 2011.

[10] T. Kumon and M. Iwasaki, “Nonlinear system identification using genetic algorithm,” *Ind. Electron. Soc. 2000. IECON 2000. 26th Annu. Conf. IEEE*, vol. 4, pp. 2485–2491, 2000.

[11] D. Centeno and F. Moreno, “Analisis comparativo del desempeño de tecnicas de control conmutado implementadas en dispositivos proogramables,” Universidad Industrial de Santander, 2013.

[12] S. Cong, G. Li, and X. Feng, “Parameters identification of nonlinear DC motor model using compound evolution algorithms,” *Proc. World Congr. Eng.*, vol. I, 2010.

[13] A. Collante, “Métodos numéricos para ecuaciones diferenciales ordinarias con MATLAB,” Universidad Nacional del Callao, 2011.

[14] J. A. Fessler and A. Arbor, “On Transformations of Random Vectors: Communications & Signal Processing Laboratory,” 1998.

[15] M. Molga and C. Smutnicki, “Test functions for optimization needs,” *Test Funct. Optim. needs*, 2005.

[16] J. Quiroga and C. Duarte, “Algoritmo PSO para identificación de parámetros en un motor DC,” *Rev. Fac. Ing.*, vol. 55, pp. 116–124, 2010.

BIBLIOGRAFIA

ARIAS, Juan y MOGOLLÓN, Moisés. Algoritmo de Optimización Gotas de Agua Virtuales Inteligentes Aplicado a La Planeación de Ruta Óptima de Un Robot Móvil. Trabajo de grado Ingeniero Electrónico. Bucarmanga: Universidad Industrial de Santander. Facultad de Ingeniería Físico-Mecánicas. 2013. 63p.

CENTENO, Diego y MORENO, Fabio. Análisis Comparativo Del Desempeño de Técnicas de Control Conmutado Implementadas En Dispositivos Programables. Trabajo de grado Ingeniero Electrónico. Bucarmanga: Universidad Industrial de Santander. Facultad de Ingeniería Físico-Mecánicas. 2013. 92 p.

COLLANTE, Andres. Métodos Numéricos Para Ecuaciones Diferenciales Ordinarias Con MATLAB. Universidad Nacional del Callao. Trabajo de grado Licenciado en Matemáticas. Bellavista: Universidad Nacional del Callao. Facultad de Ingeniería Mecánica Energía. 2011. 70 p.

CONG, Shuang, LI, Guodong y FENG, Xianyong. Parameters Identification of Nonlinear DC Motor Model Using Compound Evolution Algorithms. En: World Congress on Engineering (1: junio 30 – julio 2: Londres, Reino Unido). Memorias. Londres: WCE 2010.

DENG, Xiuqin. System Identification Based on Particle Swarm Optimization Algorithm. En: Computational Intelligence and Security (11-14, diciembre. Beijing, China). Memorias. p. 259–263.

FESSLER, Jeffrey y ARBOR, Ann. On Transformations of Random Vectors: Communications & Signal Processing Laboratory. University of Michigan. 1998. Reporte 314.

GARRIDO, Santiago. 1999. Identificación, Estimación Y Control de Sistemas No-Lineales Mediante RGO. Tesis doctoral. Leganés: Universidad Carlos III de Madrid. Departamento de Ingeniería Eléctrica, Electrónica y Automática. 1999. 161 p.

KAMKAR, Iman; AKBARZADEH-T, Mohammad-R y YAGHOOBI, Mahdi. Intelligent Water Drops a New Optimization Algorithm for Solving the Vehicle Routing Problem. En: IEEE International Conference on Systems Man and Cybernetics (SMC). (2010: 10-13, octubre: Istambul, Turquía). Memorias. p. 4142-4146.

KUMON, Toshiro; IWASAKI, Makoto; SUZUKI, Tatsuya; HASHIYAMA, Tomonori; MATSUI, Nobuyuki y OKUMA, Shigeru. Nonlinear System Identification Using Genetic Algorithm. En: 26th Annual Conference of the IEEE Industrial Electronics Society. (26: 22-28, octubre: Nagoya, Japón). Memorias. p. 2485–2491.

LIU, Guangjun; XIAOPING, Xu, y FENG, Wang. Identification of a Kind of Nonlinear System. En: Seventh International Conference on Natural Computation (7: 26-28, julio: Shanghái, China). Memorias. 2011. p. 1730–1733.

MOLGA, Marcin y SMUTNICKI, Czesław. Test Functions for Optimization Needs. Test Functions for Optimization Needs. The Bioinformatics Laboratory. 2005. 43 p.

DUARTE, César y QUIROGA, Jabid. Algoritmo PSO para identificación de parámetros en un motor DC. Rev.fac.ing.univ. Antioquia [online]. 2010, n.55 [citado 2014-08-25], pp. 116-124. Disponible en: <http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0120-62302010000500012&lng=es&nrm=iso>. ISSN 0120-6230

SHAH-HOSSEINI, Hamed. Optimization with the Nature-Inspired Intelligent Water Drops Algorithm. En: PINHEIRO DOS SANTOS W. Evolutionary Computation. 2009. p. 297-320.

. Problem Solving by Intelligent Water Drops. En: IEEE Congress on Evolutionary Computation (2007: 25-28, septiembre: Singapore). Memoria. p. 3226–3231.

. An Approach to Continuous Optimization by the Intelligent Water Drops Algorithm. En: Procedia - Social and Behavioral Sciences, 2010. vol. 32. p. 224–229.

XU, Xiaoping; WANG, Feng y QIAN, Fucai. Study on Method of Nonlinear System Identification. En: International Conference on Intelligent Computation Technology and Automation (4: 28-29, marzo: Shenzhen, China). Memorias. 2011. p. 944 – 947.