Universidad Autónoma de Estado de Hidalgo

Instituto de Ciencias Básicas e Ingeniería

Área Académica de Ingeniería y Arquitectura

Identificación de parámetros de un sistema de hidroturbina usando algoritmos evolutivos para mejorar su desempeño dinámico

T E S I S

PARA OBTENER EL TÍTULO DE

Doctor en Ciencias en Ingeniería Industrial

P R E S E N T A:

M. en C. Othon Colorado Arellano

DIRECTOR DE TESIS:

Dr. Norberto Hernández Romero

CODIRECTOR DE TESIS:

Dr. Juan Carlos Seck Tuoh Mora

Julio 2024

Universidad Autónoma del Estado de Hidalgo
Instituto de Ciencias Básicas e Ingeniería
School of Engineering and Basic Sciences
Área Académica de Ingeniería y Arquitectura
Department of Engineering and Architecture

Número de control: ICBI-AAIyA/1892/2024
Asunto: Autorización de impresión

MTRA. OJUKY DEL ROCÍO ISLAS MALDONADO
DIRECTORA DE ADMINISTRACIÓN ESCOLAR DE LA UAEH
PRESENTE.

El Comité Tutorial de **Tesis** del programa educativo de posgrado titulado **"Identificación de parámetros de un sistema de hidroturbina usando algoritmos evolutivos para mejorar su desempeño dinámico"**, realizado por el sustentante **Othón Colorado Arellano** con **número de cuenta 177333** perteneciente al programa de **Doctorado en Ciencias en Ingeniería Industrial**; una vez que ha revisado, analizado y evaluado el documento recepcional de acuerdo a lo estipulado en el Artículo 110 del Reglamento de Estudios de Posgrado, tiene a bien extender la presente:

AUTORIZACIÓN DE IMPRESIÓN

Por lo que el sustentante deberá cumplir los requisitos del Reglamento de Estudios de Posgrado y con lo establecido en el proceso de grado vigente.

Atentamente
"Amor, Orden y Progreso"
Mineral de la Reforma, Hidalgo a 13 de agosto de 2024

El Comité Tutorial

Dr. Norberto Hernández Romero
Director

Dr. Juan Carlos Seck Tuoh Mora
Codirector

Dr. Irving Barragán Vite
Miembro del comité

Dr. Gustavo Erick Anaya Fuentes
Miembro del comité

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO ICBI ÁREA ACADÉMICA INGENIERÍA Y ARQUITECTURA

Ciudad del Conocimiento, Carretera Pachuca-Tulancingo Km. 4.5 Colonia Carboneras, Mineral de la Reforma, Hidalgo, México. C.P. 42184
Teléfono: 52 (771) 71 720 00 Ext. 40031, 40030
aai_icbi@uaeh.edu.mx, mendiola@uaeh.edu.mx

Índice

Índice de figuras

Índice de cuadros

1. Planteamiento del problema

1.1. Resumen

En el presente trabajo se aborda la identificación de parámetros de regulación de un sistema de turbina hidráulica ya que es de gran relevancia para una mejora en el comportamiento dinámico del sistema , de tal modo que no se vea afectada la generación de energía eléctrica; tomando un modelo matemático altamente no lineal que incluye parámetros desconocidos variantes con el tiempo; para la obtención de dicho modelo se consideraron; el gobernador la turbina y el generador lo cual forman el sistema gobernador de hidroturbina; es decir se consideraron los aspectos mecánicos y eléctricos, razón por la cual se hace uso de los algoritmos evolutivos.

Mediante la identificación de parámetros del sistema se puede monitorear el estado operacional de la turbina hidráulica, de modo que se pueda pronosticar el rendimiento del sistema de regulación, en el presente trabajo se implementa la identificación de parámetros con algoritmos genéticos (AG), los cuales son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización, en este caso se tomó como función objetivo el error de mínimos cuadrados ponderado entre los vectores de salida reales y los estimados de tal modo que permita determinar el comportamiento de la amplitud del error.

En el presente trabajo se comparan el desempeño de los algoritmos de optimización; AG binario, enjambre de partículas (PSO), Lobos grises (GWO) y ballenas (WOA); con distintos números de individuos e iteraciones observándose una disminución en la amplitud del error de los valores reales con respecto a los valores estimados en la identificación de los parámetros: constante de tiempo (T_w), Constante de tiempo en la compuerta (T_e), Pérdida de fricción (f), constante de tiempo de inercia del generador ($T'a$) y constante de ajuste del tiempo del generador (eg). Las pruebas se realizaron en dos modos denominados con carga y sin carga para lo cual se modelo el sistema de hidro turbina con generador en Simulink y MatLab, los

algoritmos evolutivos de los métodos empleados se encuentran en el apéndice.

Los resultados obtenidos en cuando a la desviación estándar de los distintos métodos son:

a. Sistema sin carga: (AG) 0.0134, (PSO) 6.2320×10^{-11}, (GWO) 4.7005×10^{-4} y (WOA) 0.0391; cómo podemos observar en donde se tiene una valor mas bajo en la desviación estándar es el método PSO por lo cual es el que tiene un mejor comportamiento ante la identificación de parámetros del sistema.

b. Sistema con carga (AG) 3.5567×10^{-3}, (PSO) 1.8747×10^{-6}, (GWO) 40.0016 y (WOA) 0.0041; cómo podemos observar en donde se tiene una valor más bajo en la desviación estándar es el método PSO por lo cual es el que tiene un mejor comportamiento ante la identificación de parámetros del sistema.

Con los resultados obtenidos podemos considerar que el algoritmo PSO tiene un mejor comportamiento en la identificación de parámetros del sistema.

Por otro lado, se obtuvo la salida con los parámetros reales y con los parámetros sintonizados por los diferentes algoritmos genéticos del sistema en cuando a:

1. Apertura de la compuerta del sistema la cual se regula con el porcentaje de apertura de la válvula.
2. El par que obtiene el generador, en Nm el cual está relacionado con la apertura de la compuerta
3. Las revoluciones por minuto que se generan en la turbina en rad/s.

Obteniendo un mejor comportamiento con el algoritmo PSO, ya que presenta un mejor comportamiento asemejándose al comportamiento con los valores reales del sistema.

1.2. Descripción del problema

El sistema de hidroturbina con gobernador (SHTG) es un conjunto de subsistemas eléctricos, mecánicos y de control que son interconectados con la finalidad de realizar la conversión de energía mecánica a energía eléctrica. El control en este sistema es un tipo proporcional-integral-derivativo (PID) el cuál se sintoniza de acuerdo con los parámetros del sistema en un instante dado. Sin embargo, debido a que se tiene sistemas mecánicos que sufren desgaste y una máquina eléctrica que realiza la conversión de energías y que opera en distintas regiones desde sin carga, carga nominal y sobre carga, ocasiona que los parámetros del SHTG cambien en función del desgaste y los cambios de carga. La consecuencia de un cambio de parámetros hace que la dinámica del sistema cambie y exista la posibilidad de entrar en una región de inestabilidad.

Por lo tanto, es necesario sintonizar nuevamente el controlador para asegura la estabilidad. Así, el problema de identificar estos cambios es un problema de identificación de parámetros con una estructura conocida y se puede convertir en un problema de optimización, de modo tal que, la función objetivo definida debe expresarse con los resultados del sistema original e identificar las salidas del sistema, entonces el vector de parámetros desconocidos puede tomarse como una partícula y la función debe optimizarse en el proceso. Cuando las respuestas del sistema original y de los parámetros estimados están más cerca de los valores reales, se dice que se ha llegado a la optimización de la función objetivo.

1.3. Pregunta de investigación

¿Los algoritmos evolutivos son adecuados para identificar con mayor precisión los parámetros en un SHTG?

1.4. Justificación

La investigación se realiza porque es un problema vigente en la comunidad científica que analiza diferentes enfoques para resolver el problema de identificación de parámetros en el SHTG, debido a que es un sistema de generación de energía eléctrica limpia y renovable.

Los beneficios directos de la propuesta es aportar un análisis con los diferentes algoritmos evolutivos para la identificación de parámetros del SHTG cuyo modelo matemático del sistema es altamente no lineal modelado por ecuaciones diferenciales altamente acopladas en sus variables dinámicas y que no es recomendable linealizar ya que el sistema opera en un amplio rango desde sin carga, carga nominal y con sobrecarga, provocando que las sintonizaciones del control PID afecten el desempeño dinámico del sistema.

La propuesta es factible de concretarse porque se cuenta con modelos matemáticos que son implementados con soluciones numéricas y con el rápido procesamiento de datos de los microprocesadores actuales, la identificación de parámetros se puede lograr en cuestión de menos de 5 horas por cada uno de los algoritmos.
Por otro lado, al plantear el estudio con el uso de la simulación de sistemas permite ahorros en cuanto a la construcción del diseño del modelo físico de la planta.

1.5. Objetivo general

Evaluar el desempeño de distintos algoritmos evolutivos para identificar los parámetros de un sistema de hidroturbina con gobernador que permita predecir con mayor precisión el comportamiento dinámico de las distintas variables del sistema.

1.6. Objetivo específicos

1. Evaluar con base al estado del arte los métodos y técnicas de distintos algoritmos evolutivos (AE) para identificación de parámetros para el SHTG.

2. Definir el modelo del sistema SHTG y simularlo en Matlab-Simulink.

3. Definir la función objetivo de forma ponderada con tres señales de salida del SHTG, así como sus restricciones.

4. Diseñar e implementar los distintos AE para una estructura de un sistema dinámico con tres señales de salida.

5. Evaluar el comportamiento dinámico del sistema para las distintas soluciones proporcionadas por los AE.

1.7. Hipótesis

H_0: Es posible encontrar soluciones factibles y mejores con los algoritmos que los presentados en los diferentes algoritmos genéticos presentados en la literatura, para el problema de identificación para sistemas hidroturbina, logrando una mejoría en el comportamiento dinámico del sistema.

1.8. Sustento: Marco conceptual

La energía hidroeléctrica tiene una contribución importante como fuente en la generación de electricidad en muchos países. En las naciones desarrolladas, los países que tienen características topológicas adecuadas y abundantes lluvias han explotado durante mucho tiempo

su potencial hidroeléctrico, es aquí en donde el sistema de hidroturbina es utilizado para la generación de electricidad por lo que la identificación de parámetros de dicho sistema es de suma importancia ya que al mantener un buen desempeño dinámico del sistema se tiene un sistema de generación de energía con buen desempeño.

Los principios de la energía hidroeléctrica son simples, el calor del sol causa grandes cantidades de vapor de agua para elevarse, aumentando así su energía potencial. Las nubes sirven para concentrar el vapor en gotitas que caen como lluvia en terreno alto.

La extracción de energía del agua consiste en:

- Desviar parte del agua a un canal artificial
- Tener un sistema de almacenaje del agua para lograr un caudal homogéneo.
- Utilizar una máquina para convertir la energía que proviene del agua en energía mecánica.
- Control de la entrada del sistema (compuerta) y de la salida mecánica. (Ardul Munoz Hernandez, Sa'ad Petrous, & Dewi Leuan, 2013)[22]

Durante muchas décadas, las turbinas de agua fueron controladas casi exclusivamente por Gobernadores flyball, como se muestra en la figura 1.1, (Ardul Munoz Hernandez, Sa'ad Petrous, & Dewi Leuan, 2013)[22] la cual consiste en la apertura y cierre de una válvula dependiendo del flujo que circula por el canal, ya que a medida de esto los flyball, giran por la fuerza centrífuga generada, para llevar a cabo el control de la turbina.

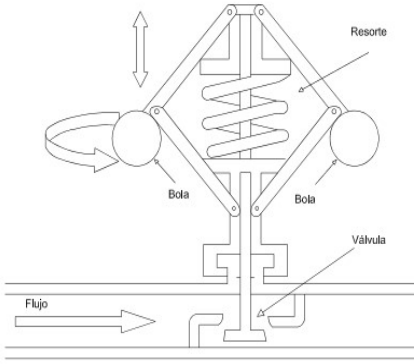

Figura 1.1: Gobernador Flyball, tomado de (Ardul Munoz Hernandez, Sa'ad Petrous, & Dewi Leuan, 2013)[22]

Los gobernadores mecánicos son los descendientes directos de los gobernadores de "flyball" con la adición crucial de un actuador "alimentado por energía" para mover la válvula de control de flujo o de las paletas guía de la turbina, de tal modo que se lograba un control con poca eficiencia, y se veía afectado el suministro de energía eléctrica. (Ardul Munoz Hernandez, Sa'ad Petrous, & Dewi Leuan, 2013)[22]

Las principales ventajas de los gobernadores electrónicos analógicos fueron la mejora de la respuesta transitoria del sistema y la facilidad de combinación de los controles del regulador de la central eléctrica; posterior a ello se implementaron los controladores PID en los sistemas tanto analógicos como digitales como se conocen actualmente.

La más vieja forma de conversión de la energía está por el uso del poder del agua; la turbina convierte la energía potencial del agua en la energía cinética rotacional de la turbina. En el esquema hidroeléctrico tradicional, la energía se obtiene de la caída del agua la cual proviene de un depósito de alto nivel en la turbina en la cual la energía del agua se convierte directamente a energía mecánica. Ya que al girar el eje de la turbina provoca que un generador adquiera velocidad angular, por lo que se genera energía eléctrica, (Ardul Munoz Hernandez, Sa'ad Petrous, & Dewi Leuan, 2013)[22] como se muestra en la figura 1.2.

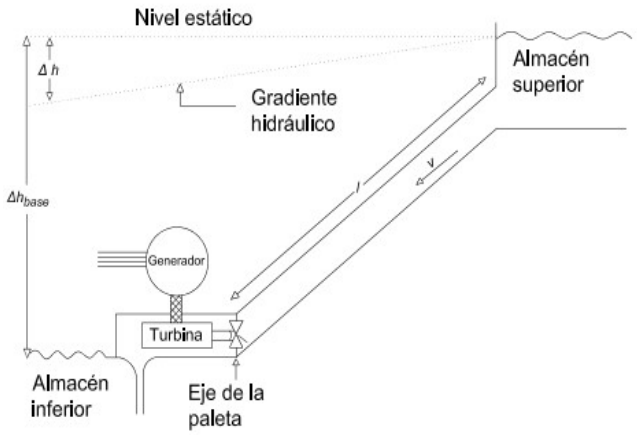

Figura 1.2: Generador turbina, tomada de , (Ardul Munoz Hernandez, Sa'ad Petrous, & Dewi Leuan, 2013)[22]

Un sistema de hidroturbina es una de las partes más importantes de una planta hidroeléctrica ya que juega un papel clave en el mantenimiento de la seguridad, la estabilidad y el funcionamiento económico de la central hidroeléctrica. (Chaoshun & Jianzhong , 2011)[17]

La identificación de parámetros no solo proporciona modelos precisos para sistemas en el diseño de control avanzados, sino que también se pude utilizar en inspección de estado, diagnostico de fallas y predicción del rendimiento. (Chaoshun & Jianzhong , 2011)[17]

Existen muchos modelos matemáticos adecuados, que, aunque sea un trabajo difícil, afortunadamente hay diferentes modelos en términos de las diferentes partes del sistema hidroeléctrico, por ejemplo el análisis dinámico no lineal del sistema de gobierno de la turbina hidráulica con un tanque de compensación. (Diyi , Cong, Xiaoyi, Pu, & Duoduo, 2013)[3], Aplicación de un algoritmo PSO para un ajuste óptimo de las ganancias PID para el regulador de turbina de agua. (Hongquing, Long, & Zuyi, 2011)[8]. Otro trabajo desarrolla tres métodos para la identificación de parámetros, el primero y el segundo métodos están diseñados en base a diferentes observadores de parámetros que utilizan el teorema de estabilidad de sistemas dinámicos, denominados observador de parámetros desconocidos y observador de parámetros basados en sincronización respectivamente, mientras que el tercer método en un algoritmo

de optimización hormiga-león, el cual es un algoritmo genético. (Zhihuan , Xiaohui, Yanbin, & Herbert, 2017)[4]

El algoritmo genético (GA) y el optimización con enjambre de partículas (PSO) han sido utilizados para la identificación de parámetros de un sistema hidroturbina ya que han demostrado su eficacia en el manejo de sistemas no lineales. (Chaoshun & Jianzhong , 2011)[17]

El problema de identificación de parámetros puede definirse como una función de un conjunto de parámetros, el problema consiste en obtener los mejores valores de los parámetros, minimizando una función objetivo, dicha función es definida como la función del error entre las salidas del modelo evaluado y de los valores reales previamente definidos. (Chaoshun & Jianzhong , 2011)[17]

Un sistema de hidroturbina es un complicado sistema no lineal que consiste en un gobernador, una compuerta, una turbina hidráulica y un generador. (Chaoshun, Li, Zhengjun, Yi, & Nan, 2016)[18]

El algoritmo de búsqueda gravitacional mejorado para la identificación de parámetros de un sistema de regulación de una turbina de agua en condiciones bajo carga y sin carga, el cual al combinarse con el PSO acelera la convergencia de búsqueda. (Zhihuan , Xiaohui, Yanbin, & Herbert, 2017)[4]

El algoritmo mejorado de optimización de leones (IALO), para la identificación de parámetros del sistema de gobierno de una turbina hidráulica el algoritmo IALO, tiene buena característica de convergencia y estabilidad. (Tian , Changyu, Qi, Yi, & Qiurong, 2018)[24]

Para la caracterización de un modelo de identificación de parámetros de un sistema de hidroturbina se requiere la utilización de MATLAB, porque es una herramienta computacional

flexible y Simulink porque son herramientas para modelar, simular y analizar estos sistemas dinámicos, ofrecen un conjunto de herramientas que pueden ser usadas para construir complicados sistemas, así como librerías para la construcción de diagramas de bloques. (Hongquing, Long, Nkosinathi, & Zury, 2008)[10]

2. Capitulo 2 modelo matemático del sistema hidroturbina

2.1. Introducción

El sistema de gobierno de una turbina hidráulica SHTG es una de las partes más importantes de una planta hidroeléctrica. El modelo matemático del SHTG es un sistema altamente no lineal y tiene cuatro componentes: gobernador de la turbina hidráulica, compuerta, hidroturbina y sistema de carga (generador) (Chaoshun & Jianzhing, 2011)[16]. La figura 2.1, muestra la estructura de del SHTG.

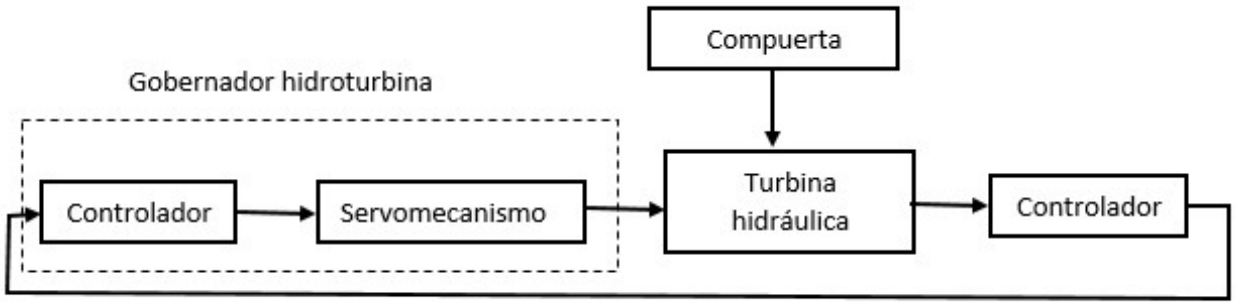

Figura 2.1: Sistema de gobierno de una turbina hidráulica. Elaboración propia

2.2. Gobernador de turbina

El gobernador de turbina hidráulica consta de controlador y servomecanismo. En general, la ley de control de PID paralela es ampliamente utilizada. El controlador PID podría expresarse como:

$$\frac{\sigma(s)}{c(s)-x(s)} = \frac{1}{1+b_p\frac{k_i}{s}}\left(k_p + \frac{k_i}{s} + \frac{k_d s}{1+T_{1v}s}\right) \tag{1}$$

Dónde: s es el operador Laplace, c es la velocidad dada, x es la velocidad establecida a la que se requiere que funcione la hidroturbina, σ la salida del controlador PID, acción de control; k_p, k_i y k_d son la ganancia proporcional, la ganancia integral y la ganancia diferencial respectivamente, T_{1v} es la constante de tiempo diferencial, b_p es el coeficiente de retroalimentación. La figura 2.2, muestra el diagrama a bloques del controlador PID de la planta.

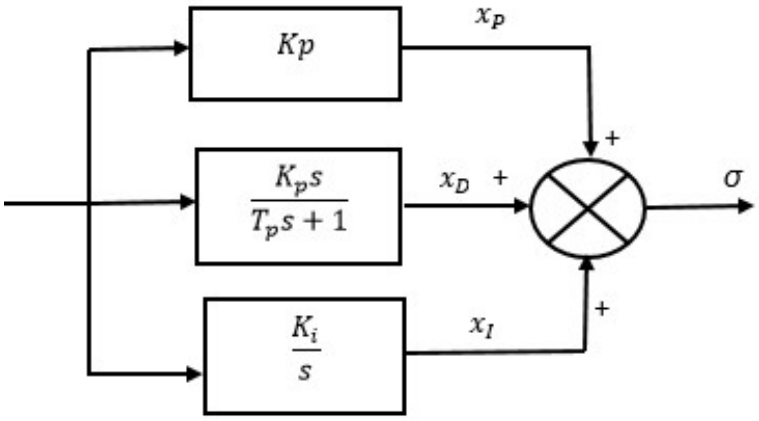

Figura 2.2: Diagrama de bloques del controlador PID

2.3. Servomecanismo

El servomecanismo es el control del gobernador, que se utiliza para operar la turbina hidráulica de acuerdo con la señal de salida del controlador PID, la cual tiene como variable de entrada la apertura de la compuerta (y) y como salida el par (σ) del generador, la función de trans-

ferencia del servomecanismo es:

$$\frac{y(s)}{\sigma(s)} = \frac{1}{T_y s + 1} \tag{2}$$

Dónde: y es la apertura de la compuerta, T_y es la constante de tiempo de inercia del servomecanismo. La figura 2.3 muestra el diagrama de bloques del servomecanismo.

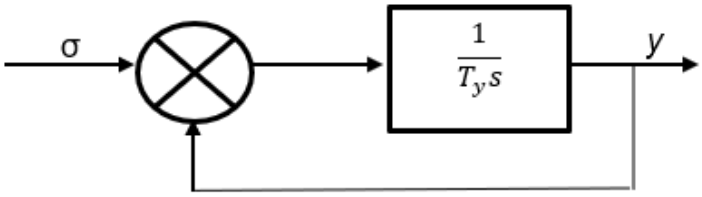

Figura 2.3: Servomecanismo

2.4. Turbina hidráulica y generador

En el SHTG, el sistema hidráulico es uno de los subsistemas importantes que proporciona al rotor primario y contiene principalmente turbina hidráulica, compuerta y tanque de compensación.

En la práctica se ha demostrado que cuando el sistema está en funcionamiento estable y la velocidad de la turbina varía en un rango pequeño, los resultados teóricos obtenidos de las características de estado estable de la turbina hidráulica están de acuerdo con los resultados medidos. Por lo que el modelo matemático del sistema hidráulico se puede considerar en tres partes, (Tian , Changyu, Qi, Yi, & Qiurong, 2018)[24].

1. La turbina hidráulica convierte la energía hidráulica en energía mecánica e impulsa el generador para generar electricidad. En una vecindad de un punto de operación, el par de la turbina m_t y el flujo q se pueden expresar con funciones lineales de apertura de la válvula y, altura de la columna de agua h y la velocidad x. Las características de estado estable de la turbina hidráulica podrían mostrarse como:

$$m_t(s) = e_x x(s) + e_y y(s) + e_h h(s) \tag{3}$$

$$q(s) = e_{qx} x(s) + e_{qy} y(s) + e_{qh} h(s) \tag{4}$$

Dónde: e_x, e_y, e_h, e_{qx}, e_{qy} y e_{qh} son coeficientes de transferencia de turbina hidráulica. La figura 2.4, muestra el diagrama de bloques de la turbina hidráulica.

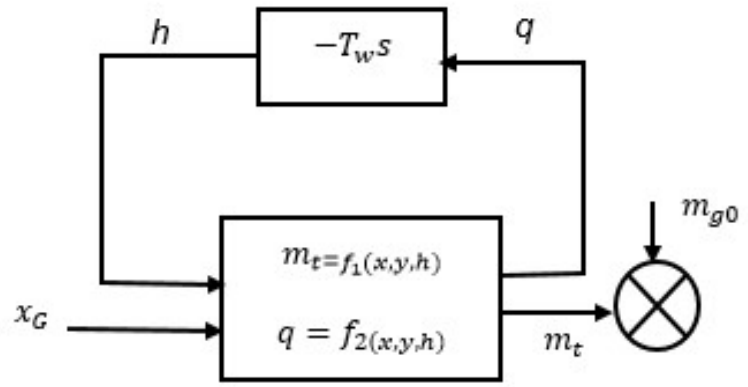

Figura 2.4: Turbina hidráulica

2. Las características del fluido de la compuerta se pueden tomar como parte de la inercia del flujo del sistema de descarga de presión. En la compuerta, los cambios repentinos en el flujo causarán cambios severos en la columna del agua. A eso le llamamos el martillo de agua. El proceso dinámico se puede describir como:

$$F(s) = \frac{h(s)}{q(s)} = -\frac{T_w}{T_e} \tanh(T_e s + f) \tag{5}$$

Dónde: T_w es la constante de tiempo del agua, T_e es el tiempo de viaje del agua, f son las pérdidas por fricción en la compuerta.

3. Modelo generador y carga, en el estudio de SHTG, el generador y el sistema de carga a menudo se simplifican como un sistema de primer orden. La función de transferencia podría expresarse como:

$$\frac{x(s)}{m_t(s) - m_g(s)} = \frac{1}{T_a' s + (e_g - e_x)} \tag{6}$$

donde m_g es el par de carga, $T'_a = T_a + T_b$, T_a es la constante de tiempo del generador, T_b es la constante de tiempo de inercia de la carga, e_g es el coeficiente de ajuste del generador. La figura 2.5, muestra el diagrama de bloques del generador

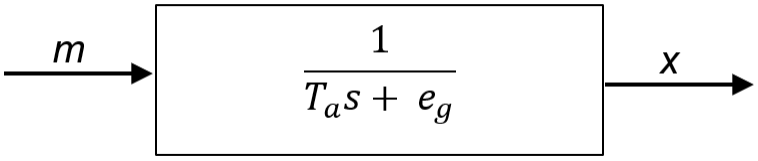

Figura 2.5: Bloque en Laplace del Generador

La interconexión del sistema SHTG, se muestra en la figura 2.6. Se muestra que el sistema es no lineal, iniciando con el sistema de gobernador de la turbina, el servomecanismo, la turbina hidráulica y el generador. (Chaoshun & Jianzhing, 2011)[16]

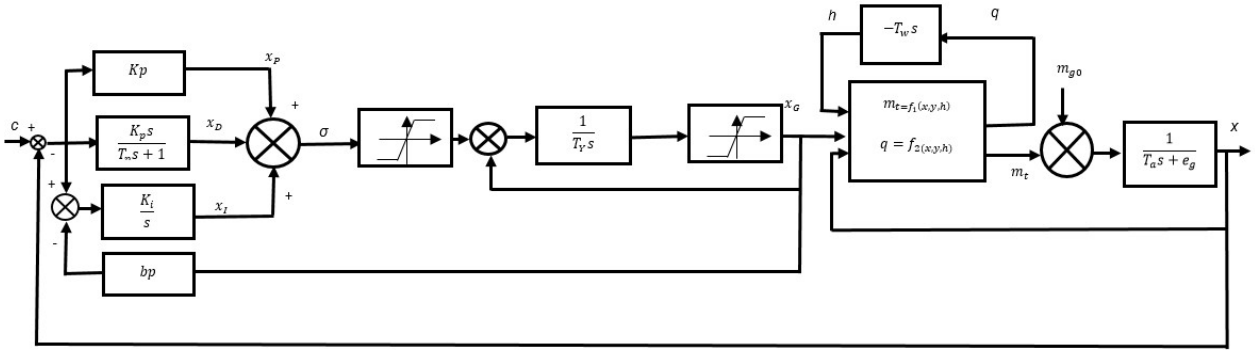

Figura 2.6: Sistema gobernador de la turbina, servomecanismo, turbina hidráulica y generador

2.5. Modelo matemáticos del sistema gobernador de hidroturbina

De acuerdo al diagrama de la Fig, 2.6, se obtiene el sistema de ecuaciones diferenciales del sistema SHTG, (Chaoshun & Jianzhing, 2011)[16] tenemos:

$$\frac{dx}{dt} = e_x x + \frac{1}{t_\alpha} m_t - \frac{1}{t_\alpha} m_{g0} \tag{7}$$

$$\frac{dy}{dt} = -\frac{K_P}{T_Y} x + \frac{1}{T_Y} x_I + \frac{1}{T_Y} x_D - \frac{1}{T_Y} y + \frac{K_P}{T_Y} C \tag{8}$$

$$\frac{dx_I}{dt} = -K_I x + K_I b_P y + K_I C \tag{9}$$

$$\frac{dx_P}{dt} = -\frac{K_D e_g}{T_D T_a} x - \frac{1}{T_D} x_D - \frac{K_D}{T_D T_a} m_t + \frac{K_D}{T_D T_a} m_{g0} + \frac{K_D}{T_D} C \tag{10}$$

$$\frac{dq}{dt} = -\frac{1}{T_w} h \tag{11}$$

$$\tag{12}$$

$s.t. \sigma = x_p + x_I + x_D, \sigma_{min} \leq \sigma \leq \sigma_{max}, y_{min} \leq y \leq y_{max}$

Donde: $[x, y, x_I, x_D, q]$ es el vector de variables de estado; m_t, h, c y m_{g0}, son los valores de entrada. $m_t = f_1(y, x, h), h = f_3(x, y, q)$. Cabe señalar que el cálculo del caudal q y la altura de agua h se calculan de forma iterativa mediante el cálculo de $\frac{dq}{dt} = -\frac{1}{T_w} h$ y $h = f_3(y, x, q)$

Las funciones de excitación del sistema generan valores fijos de 0.04 p.u. y 0.1 p.u., respectivamente.

Los parámetros establecidos para realizar la simulación se dan en la tabla 2.1, (Tian , Changyu, Qi, Yi, & Qiurong, 2018)[24].

Parámetro	Valor
Constante proporcional, K_p	5.5912
Constante integral, K_I	1.0611
Constante derivativa, K_D	3.2800
Tiempo de muestreo, $T_1 v$	0.28
Coeficiente de retroalimentación, b_n	0.04
Constante de tiempo del servomecanismo, T_y	0.1

Tabla 2.1: Parámetros del sistema

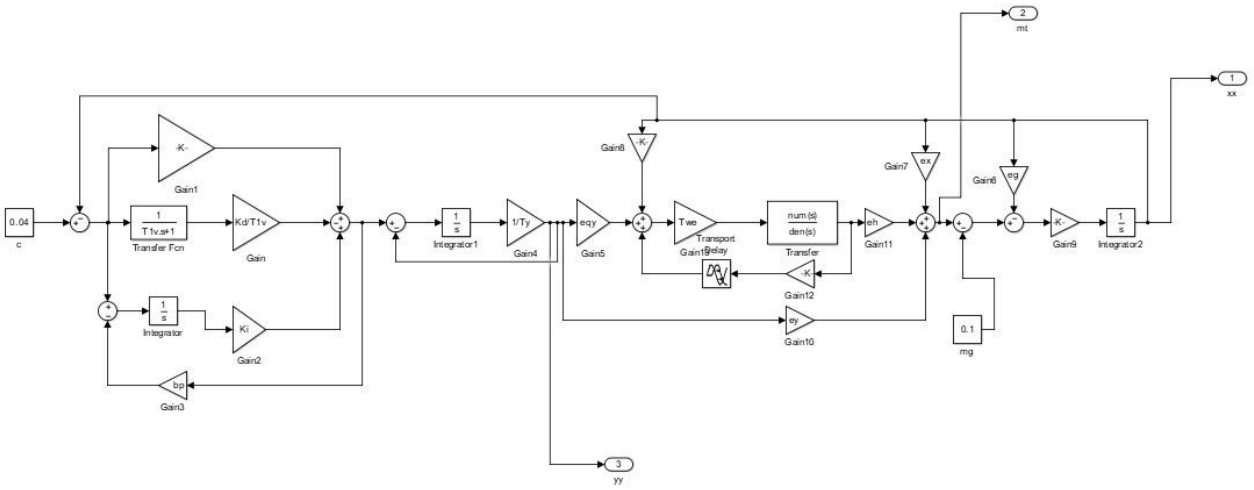

Figura 2.7: Diagrama en Simulink para desarrollo de las simulaciones del sistema. Elaboración propia en Simulink

Los parámetros a identificar, se muestran en la tabla 2.2, los cuales son valores de referencia (Tian , Changyu, Qi, Yi, & Qiurong, 2018)[24]. La importancia de identificar estos parámetros es que permiten analizar la estabilidad del sistema, ya que son cruciales para el buen funcionamiento del sistema.

Parámetro	Valor
Sistema hidráulico, constante de tiempo, T_w	1.5
Constante de tiempo en la compuerta, T_e	0.53
Pérdida de fricción, f	0.01
Constante de tiempo de inercia del generador, T'_a	12.0
Constante de ajuste del generador, e_g	0.4433

Tabla 2.2: Parámetros del sistema

Las salidas en el sistema SHTG, velocidad de la turbina en $[rad/s]$, par del generador en $[Nm]$ y el porcentaje de apertura de la válvula de la compuerta se simularon bajos las condiciones de sin carga y con carga con los parámetros tomados como referencia, para posteriormente comparar las salidas respectivas con los parámetros de referencia y los obtenidos con los algoritmos evolutivos correspondientes.

Salidas del sistema SHGT sin carga, las salidas de velocidad angular de la turbina medido $[rad/s]$, figura 2.8, el par del generador $[Nm]$, figura 2.9 y el porcentaje de apertura de la compuerta, figura 2.10, con respecto al tiempo $[s]$, se observa como dichas salidas tienen un periodo transitorio y posterior a ello tienen al estado estable.

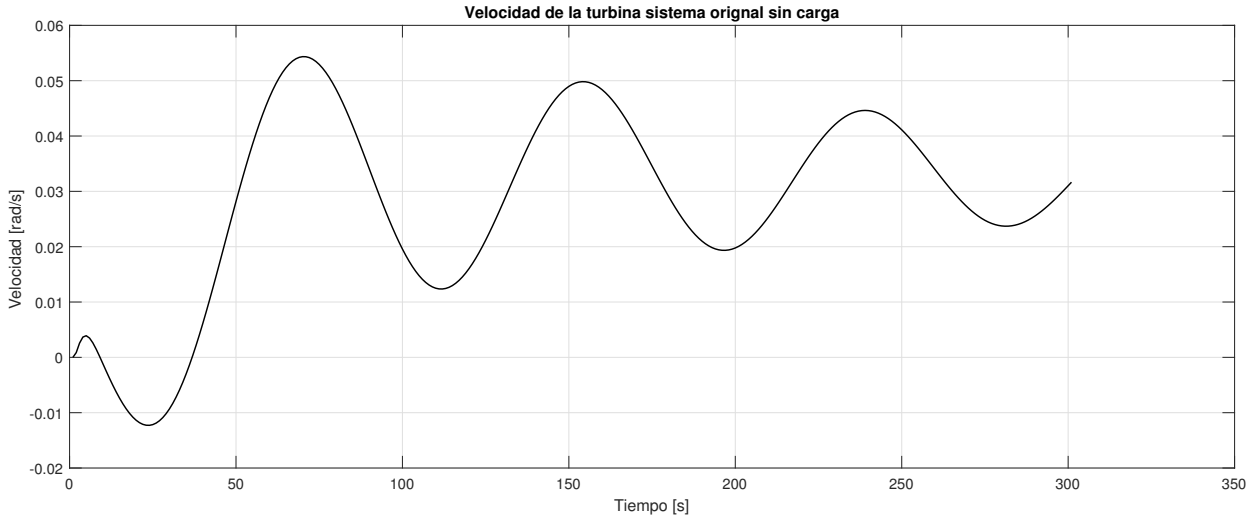

Figura 2.8: Velocidad de la turbina, sin carga.

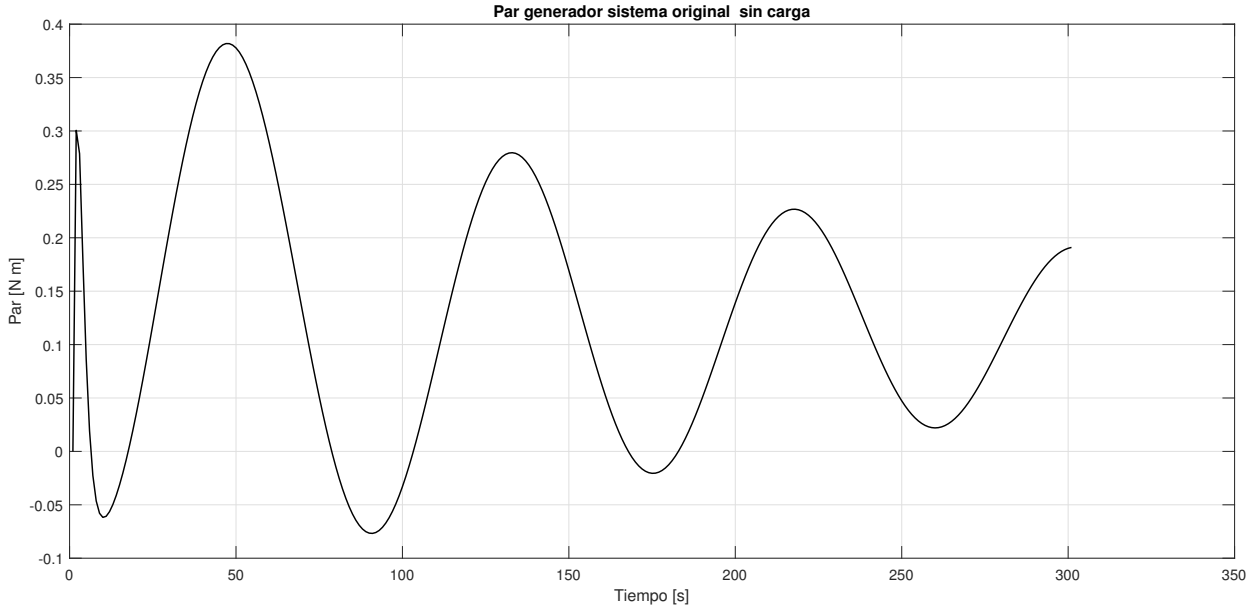

Figura 2.9: Par generador, sin carga

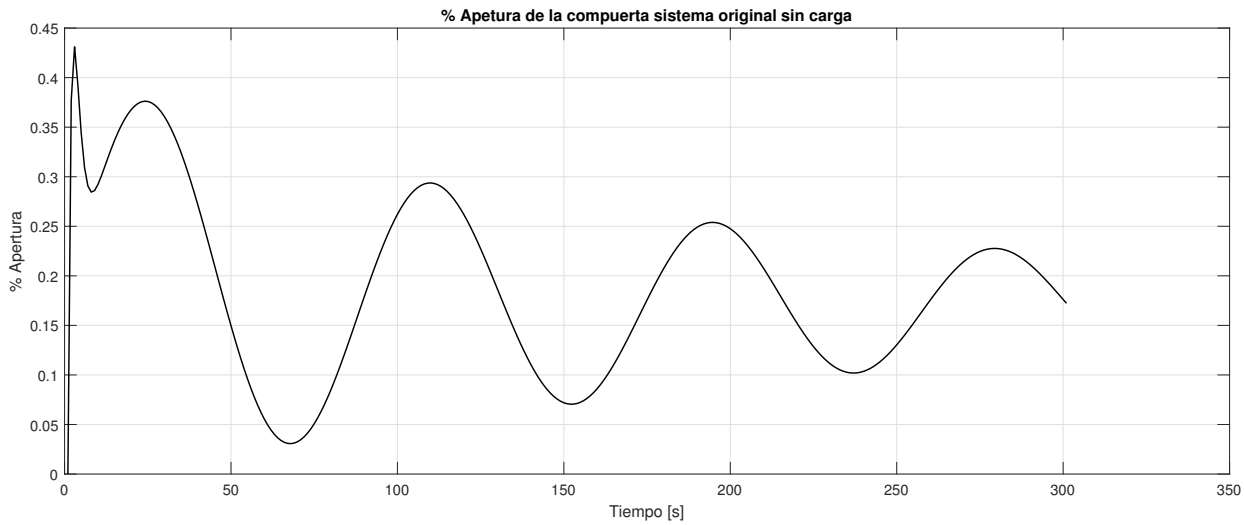

Figura 2.10: Porcentaje de apertura de la compuerta, sin carga

El sistema SHTG, se simulo también en condiciones de carga con los parametros de referencia para observar el comprtamiento de las salidas de velocidad de la turbina, par del generador y porcentaje de apertura de la compuerta, las cuales se muestran en la figuras 2.11, 2.12 y 2.12 respectivamente, en este caso las salidas tienen un comportamiento transitorio de inestabilidad y posterior a ello tienden a un estado estacionario.

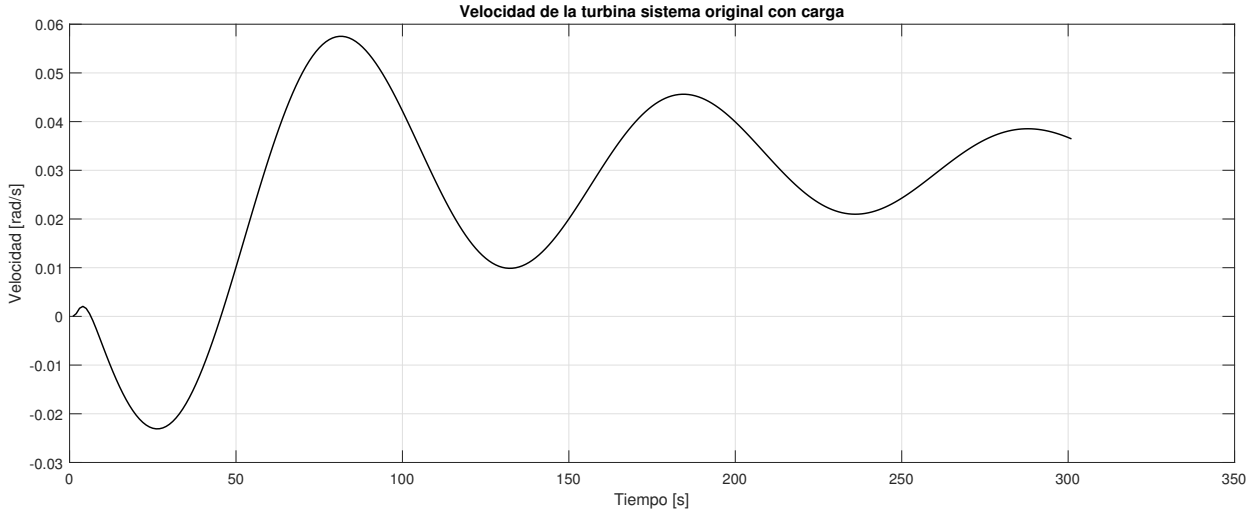

Figura 2.11: Velocidad de la turbina, con carga

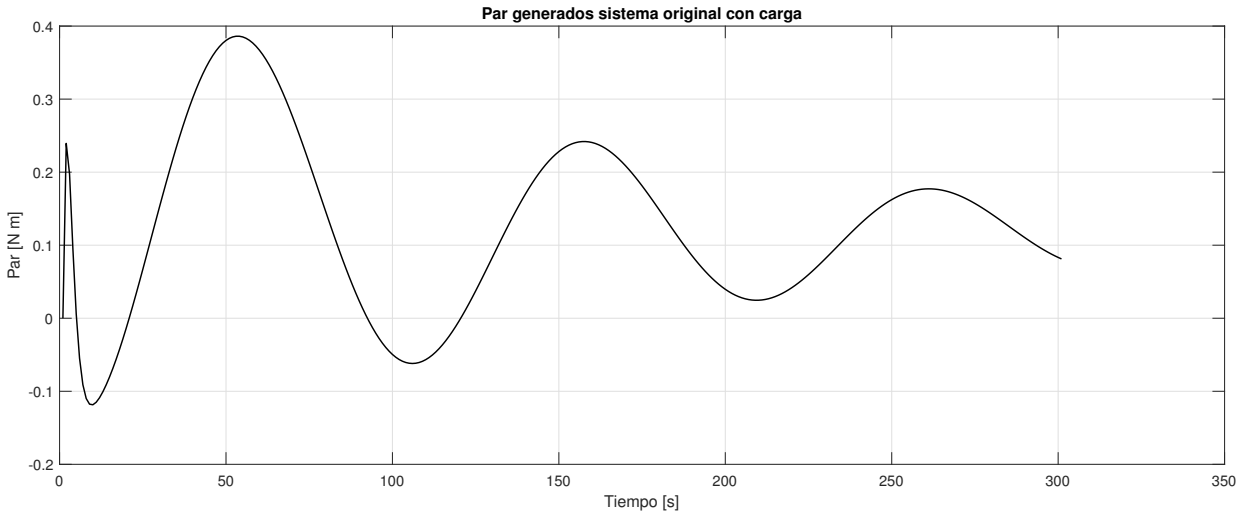

Figura 2.12: Par del generador, con carga

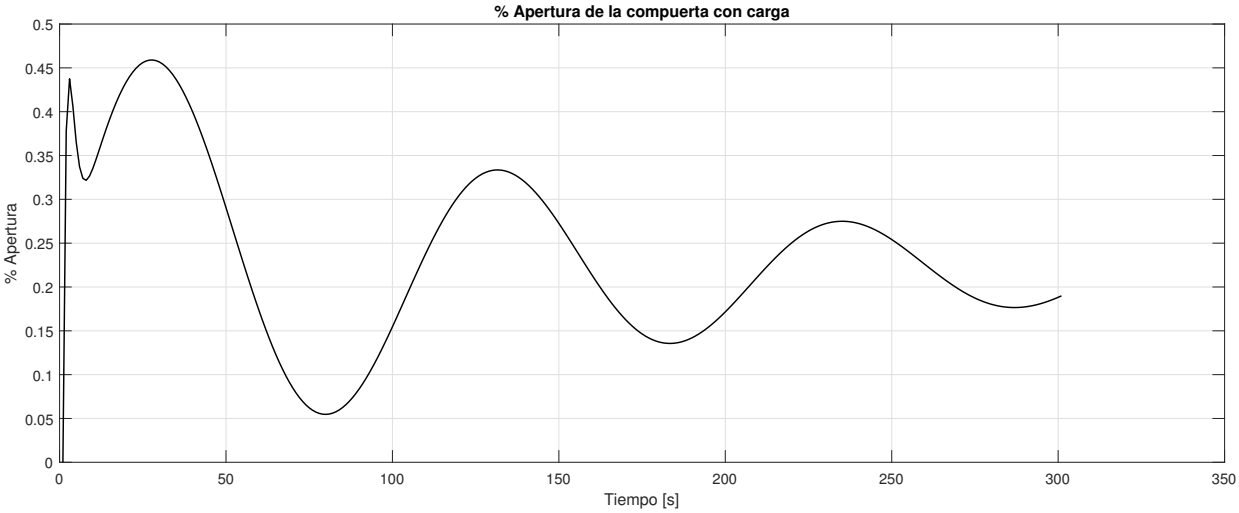

Figura 2.13: Porcentaje de apertura de la compuerta, con carga

En el capítulo siguiente se mostrarán los resultados que se obtienen con los algoritmos evolutivos para determinar si el comportamiento del sistema se conserva bajo las mismas condiciones.

3. Capítulo 3 Algoritmos evolutivos

3.1. Introducción

Los Algoritmos Genéticos (AG) son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización. Están basados en el proceso genético de los organismos vivos. A lo largo de las generaciones, las poblaciones evolucionan en la naturaleza de acuerdo con los principios de la selección natural y la supervivencia de los más fuertes. (Universidad el País Vasco)[13]

El desarrollo de los AG, se debe en gran medida al profesor investigador John Holland de la Universidad de Michigan. A finales de la década de los 60 desarrolló una técnica que imitaba en su funcionamiento a la selección natural, aunque originalmente esta técnica recibió el nombre de "planes reproductivos", a raíz de la publicación en 1975 de su libro "Adaptation in Natural and Artificial Systems" (Holland, 1975)[15] se conoce principalmente con el nombre de AG.

Básicamente, un AG consiste en una población de soluciones codificadas de forma similar a cromosomas; cada uno de estos cromosomas tendrá asociado un ajuste, valor de bondad o fitness, que cuantifica su validez como solución al problema; en función de este valor se le darán más o menos oportunidades de reproducción. (Gestal & Rivero, 2010)[11]

Los AG tienen una fuerte base biológica, en sus orígenes los algoritmos evolutivos consistieron en copiar procesos que tienen lugar en la selección natural, este concepto había sido introducido, por Charles Darwin. A pesar de que aún hoy en día no todos los detalles de la evolución biológica son completamente conocidos existen algunos hechos apoyados sobre una fuerte evidencia experimental. (Gestal & Rivero, 2010)[11]

Por otro lado, los algoritmos de búsqueda estocástica también se han empleado para abordar el problema de identificación de parámetros para el sistema HTRS, (Tian , Changyu, Qi, Yi,

& Qiurong, 2018)[24].

3.2. Algoritmos genéticos (AG) en el proceso de optimización

Los algoritmos genéticos constituyen una técnica de búsqueda fundamentada en el proceso de evolución natural en la cual los individuos más adaptados tienen mayores probabilidades de sobrevivir y de transferir su material genético a las siguientes generaciones. La idea fundamental de los algoritmos genéticos consiste en encontrar una solución aceptable a un problema por medio del mejoramiento de un conjunto de individuos, cuya función de evaluación corresponde a una solución del problema. Esta optimización se realiza mediante procesos selectivos y de intercambio de información genética. Dichos procesos están dados por operadores genéticos, que definen la estructura de un AG (Goldberg, 1989)[12].

3.2.1. Estructura de un algoritmo genético

Un AG básico está constituido por la población y el siguiente conjunto de reglas: Función objetivo: proporciona una medida de desempeño del sistema asociado a cierto individuo en la población. Operador de selección: busca ciertos individuos de la población, quienes darán origen a las futuras generaciones. Por lo general, la selección depende del valor de la función de evaluación de cada individuo. Operador de cruce: consiste en mezclar la información genética de dos individuos, a fin de generar nuevos individuos. Operador de mutación: consiste en alterar las características genéticas de un individuo, con el objeto de aumentar la probabilidad de exploración del espacio de búsqueda y disminuir el riesgo de estancamiento del algoritmo en óptimos locales.

Como los AG se basan en los procesos de evolución de los seres vivos, tienen los siguientes elementos:

- Operadores genéticos, Son los diferentes métodos u operaciones que se pueden ejercer sobre una población que nos permite obtener poblaciones nuevas.
- Población, conjunto de individuos (cromosomas), se debe ir obteniendo de forma suce-

siva distintas poblaciones.

- Individuo, es un ser que caracteriza su propia especie, es el cromosoma y es el código de información sobre el cual opera el algoritmo.

Su aplicación al control se fundamenta en el que un proceso complejo se puede a menudo reducir a una aproximación funcional numérica del problema, que se pueden optimizar mediante los AG (Santos, 2011)[23].

En la Figura 3.1, se muestra como sobre cada población se aplica una función de evaluación para obtener el valor fitness, cada población se transforma, muta a nueva población mediante los operadores de reproducción cruce y mutación

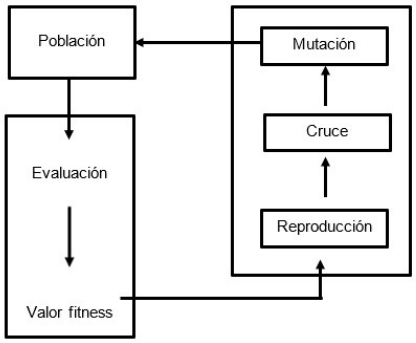

Figura 3.1: Secuencia de un Algoritmo genético, elaboración propia

La secuencia de ejecución de un AG es la siguiente:

- Generar una población aleatoria en el espacio solución del problema.
- Establecer una función, de modo que se pueda evaluar la adaptación de los nuevos individuos a la nueva población.
- Crear una nueva población efectuando operaciones como selección/ reproducción proporcional a la adaptación, cruce y mutaciones en los individuos en la que ésta acaba de ser medida.
- Reemplazar la antigua población.
- Evolucionar utilizando la nueva población hasta cumplir un número de iteraciones o un determinado número de error.

La figura 3.2 muestra el diagrama de flujo de la realización de las diferentes etapas de ejecución del AG, donde básicamente el ciclo es evaluar función-seleccionar individuo-cruza-mutación-prueba de terminación de ciclo.

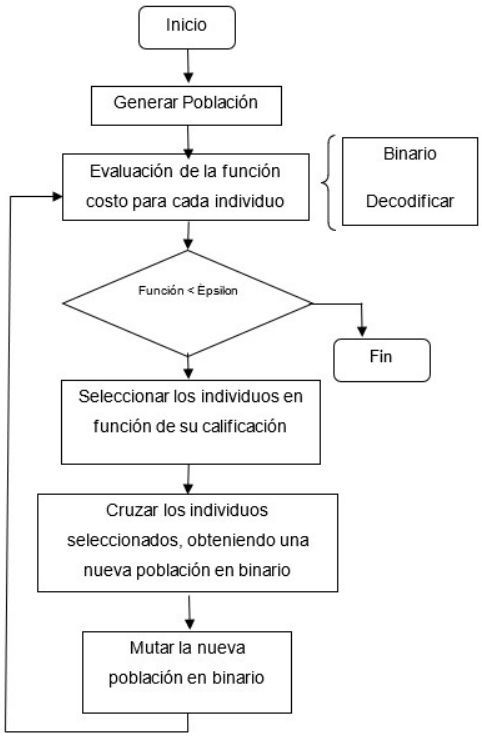

Figura 3.2: Secuencia de un Algoritmo genético (Goldberg, 1989)[12]

3.2.2. Pseudocódigo de un algoritmo genético

```
Inicio
Inicialización de parámetros
Generar la población inicial aleatoria
While terminar = falso
Evaluar la función objetivo
For i = 1 hasta tamaño de la población
Seleccionar 2 individuos para el torneo
El mejor individuo es padre 2
Generar aleatorio para operador cruce
```

```
If aleatorio < ProbCruce
Cruzar padre1 y padre 2
Else
Hijo1 = padre1
Hijo 2 = padre2
Generar aleatorio para operador de mutación Hijo1
If aleatorio < ProbMuta
Mutar Hijo1
Generar aleatorio para operador de mutación Hijo2
If aleatorio < ProbMuta
Mutar Hijo2
Calcular fitness de los dos descendientes
Insertar los dos descendientes en la nueva generación
Actualizar generaciones
Fin for
If generaciones = TotalGeneracines
Terminar = True
Fin while
Fin
```

3.3. Identificación de parámetros en el sistema de hidroturbina basado en AG

En general, un problema de identificación de parámetros en un sistema HTRS puede abordarse como un problema de optimización. Esto implica tener una estructura de modelo conocida y seleccionar un algoritmo para convertirlo en un problema de optimización.

En el trabajo de investigación se tomaron como variables de salida la velocidad de la turbina x, la apertura de la válvula de la compuerta y, así como el par m_t de la turbina. En el SHTG las constantes K_P,K_I y K_D, pueden ser leídas directamente del controlador PID y

T_y del servo mecanismo puede ser medido de manera experimental. Se propone una nueva función objetivo con ponderaciones que representan la importancia de cada parámetro y las ponderaciones se calculan con la desviación de la salida; esta función objetivo es un error de mínimos cuadrados ponderando los vectores de salida real y estimado, la cual puede ser definida como: (Tian , Changyu, Qi, Yi, & Qiurong, 2018)[24].

En el caso del SHTG (Sistema Hidrotérmico de Generación), las constantes K_P, K_I y K_D pueden leerse directamente del controlador PID, mientras que T_y del servomecanismo puede medirse experimentalmente. Se propone una nueva función objetivo con ponderaciones que representan la importancia de cada parámetro. Estas ponderaciones se calculan a partir de la desviación de la salida. La función objetivo es un error de mínimos cuadrados ponderando los vectores de salida real y estimado, y puede definirse como:

$$C_{IOF}(\theta) = \sum_{k=1}^{L} \sum_{j=1}^{n} w_j (z_j(k) - \hat{z}(k))^2 \tag{13}$$

Dónde: $z = [xym_t]$ es el vector de salida del sistema real,$\hat{z} = [\hat{x}\hat{y}\hat{m}_t]$ es el vector de salida del modelo estimado, L es el número de muestras y n es el número de salidas del sistema, $n = 3$. El vector de salida se mide en distintos momentos, al calcular los pesos en la función objetivo. El vector peso $w = [w_1 w_2 w_3]$, es calculado de acuerdo a los siguientes pasos: (Tian , Changyu, Qi, Yi, & Qiurong, 2018)[24].

1. Establecer el valor de cada parámetro del vector θ_i, donde $i = 1, \ldots, m$ (donde m es la dimensión del vector θ, en este caso $m = 5$).
2. *Loop* : $i = 1 : m$
   - Cambia el valor del i-ésimo parámetro: $\theta_{\text{new}} = \theta_i \times (1 + \Delta\,\%)$ y obtener el valor del vector de salida del sistema estimado $\hat{z}_i = [\hat{z}_{i1}(k), \ldots, \hat{z}_{in}(k)]$.

   Endloop

3. Calcular

$$\sum_{j=1}^{m}\sum_{k=1}^{L}(z_j(k)-\hat{z}_{ij}(k))^2, \quad j=1,\ldots,n \tag{14}$$

4. Calcular el $j-$ésimo peso

$$w_j = \frac{g_j}{\sum_{j=1}^{n} g_j} \tag{15}$$

Para realizar la identificación de parámetros con un algoritmo evolutivo primero se elige una señal de entrada adecuada para excitar el sistema original como el estimado, la salida medida obtenida y la salida simulada como entrada en la función fitness se utiliza para calcular el ajuste en dicha función; entonces el AE identifica el vector de parámetros desconocidos minimizando la función fitness $C_{IOF}(\hat{\theta})$, así el ciclo continua hasta que los parámetros identificados se aproximan gradualmente a los valores reales. (Tian , Changyu, Qi, Yi, & Qiurong, 2018)[24], el proceso se muestra en la figura 3.3.

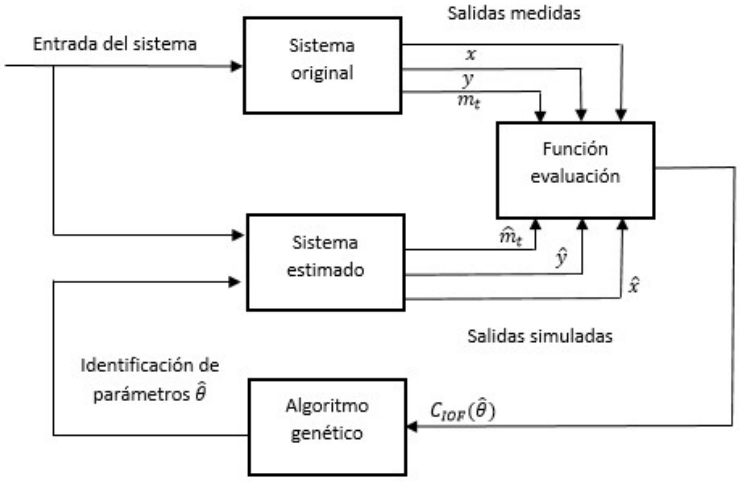

Figura 3.3: Metodología de sintonización de parámetros. Tomada de (Tian , Changyu, Qi, Yi, & Qiurong, 2018)[24].

La precisión de la identificación de parámetros se mide por medio del error:

$$PE = \left|\frac{\theta_i - \hat{\theta}_i}{\theta_i}\right| \times 100\,\%, \quad i=1,2,\ldots,m \tag{16}$$

Y el promedio del error:

$$APE = \frac{1}{m}\sum_{i=1}^{m}\left|\frac{\theta_i - \hat{\theta}_i}{\theta_i}\right| \times 100\,\% \tag{17}$$

Donde θ_i, es el parámetro en el sistema original, $\hat{\theta}_i$,es el parámetro del sistema estimado, m, es la dimensión del vector θ.

3.4. Algoritmo de optimización de ballenas, Whale Optimization Algorithm (WOA)

Las ballenas son elegantes criaturas, consideradas como el mamífero más grande del mundo. Lo más interesante de las ballenas jorobadas es su método especial de caza, este método de alimentación se denomina método de alimentación de burbujas distintivas a lo largo de un círculo o trayectoria en forma de "9". (Mirjalili & Lewis, 2016)[20]

El WOA, se basa como se mencionó en el comportamiento de cacería de las ballenas, el cual se le conoce como estrategia de red de burbujas.

3.4.1. Modelo matemático y optimización del algoritmo

El método consiste en seleccionar una muestra aleatoria de posibles soluciones que circundan al área donde se encuentra el objetivo (presa), repitiendo el proceso actualizando las posiciones en torno al mejor obtenido y así sucesivamente, hasta satisfacer el criterio de búsqueda. (Mirjalili & Lewis, The Whale Optimization Algoritm, 2016)[20]. La figura 3.4 muestra el diagrama de flujo del algoritmo WOA.

El algoritmo WOA asume que la mejor solución candidata actual es la presa objetivo o está cerca de la óptima; una vez definido el mejor agente de búsqueda, los otros agentes de búsqueda intentaran actualizar sus posiciones hacia el mejor agente de búsqueda. Este comportamiento se representa por las ecuaciones:

$$\vec{D} = |\vec{C} \cdot \vec{X}_{\text{best}}(t) - \vec{X}(t)|, \vec{D} = |\vec{X}_{\text{best}}(t) - \vec{X}(t)| \tag{18}$$

Donde t, indica iteración actual, $\vec{A}$ y $\vec{C}$ son los coeficientes del vector, $\vec{X}*$ es la posición del vector de la mejor solución obtenida.

Los vectores $\vec{A}$ y $\vec{C}$, son calculados:

$$\vec{A} = 2\vec{a} \cdot \vec{r} - \vec{a}, \vec{C} = 2 \cdot \vec{r} \tag{19}$$

Donde $\vec{a}$ es un decremento lineal de 2 a 0, $\vec{r}$, es un vector aleatorio en $[0, 1]$ (Mirjalili & Lewis, 2016)[20]

Los movimientos del método de burbuja se describen por [20], como mecanismo de cerco el cual se obtiene reduciendo el valor de a de 2 a 0, o mediando [21] que modela un movimiento en espiral, con b constante y l un número aleatorio.

$$\vec{X}(t+1) = \vec{X}_{\text{best}}(t) - \vec{A} \cdot \vec{D} \tag{20}$$

$$\vec{X}(t+1) = \vec{D'} \cdot e^{bl} \cdot \cos(2\pi l) + \vec{X}_{\text{best}}(t) \tag{21}$$

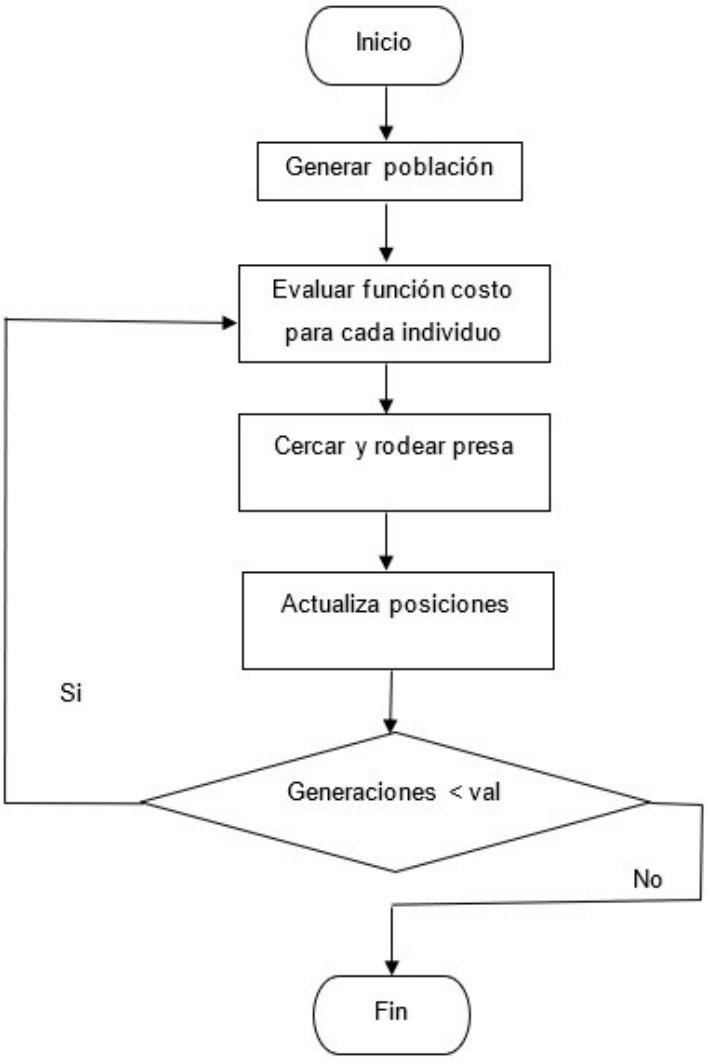

Figura 3.4: Diagrama de flujo de WOA, basado en (Mirjalili & Lewis, The Whale Optimization Algoritm, 2016)[20]

3.4.2. Pseudocódigo del algoritmo WOA

El algoritmo inicia colocando soluciones aleatorias, a cada iteración busca agentes y actualiza su posición con respecto ya sea al azar o la mejor solución, el parámetro a, es disminuido de 2 a 0, un agente es elegido de manera aleatoria cuando $|\vec{A}| > 1$ mientras que la mejor solución es seleccionada cuando $|\vec{A}| < 1$,para cambiar la posición del agente de búsqueda, a continuación, se muestra el algoritmo del WOA. (Mirjalili & Lewis, 2016)[20].

Inicializar población de ballenas $X_i(i = 1, 2, 3, \ldots, n)$

Calcular la fitness de cada agente de búsqueda

$X* =$ el mejor agente buscado

```
while t<maximo numer de iteraciones
for cada agente de búsqueda
bajar a, A, C, I y p
if1 (p<0.5)
if2 (|A|<1)
Actualizar la posición actual
else if2 (|A|<1)
Seleccionar un agente aleatorio (X_ran)
Actualizar la posición actual
end if2
else if1 (p>=0.5)
Actualizar la posición actual
end if1
end for
%Comprobar si algún agente de búsqueda va más allá del espacio de búsqueda y modificarlo
%Calcular la fitness de cada agente de búsqueda, t=t+1
end while
return X*
```

3.5. Algoritmo de Optimización de lobos grises, Grey wolf optimization (GWO)

Los lobos grises son considerados depredadores ápice lo que significa que están en la cima de la cadena alimenticia, la mayoría de los lobos prefieren vivir en manada, tienen una jerarquía

social dominante muy estricta. (Mirjalili, Mohammad Mirjalili, & Lewis, Grey Wolf Optimizer, 2014)[21]

El algoritmo de optimización GWO, se inspira en el comportamiento de las manadas de lobos grises y su organización social para la caza de sus presas. La manada se conforma de dos lideres llamados alfa (α) y el resto de la manada. En esta estructura social los miembros alfa son los responsables de la toma de decisiones y el resto de los miembros deben seguir sus órdenes, dentro del algoritmo el alfa es la mejor solución. El segundo nivel de jerarquía del grupo son los miembros beta (β), los cuales deben colaborar con la toma de decisiones, seguidos por los delta (δ); estos son la segunda y tercera mejor solución respectivamente y existe una última clasificación que se encuentra más abajo en la jerarquía de las manadas de lobos grises denominada (ω), la cual está formada por el resto de posibles soluciones (Mirjalili, Mohammad Mirjalili, & Lewis, Grey Wolf Optimizer, 2014)[21]. La figura 3.5 muestra el diagrama de flujo del algoritmo GWO.

3.5.1. Modelo matemático

Los lobos grises rodean a su presa durante la caza, para modelar el comportamiento se proponen las siguientes ecuaciones:

El proceso de rodear la presa se determina con las siguientes ecuaciones:

$$\vec{D} = |\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)| \tag{22}$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D} \tag{23}$$

Donde t indica la iteración actual, $\vec{A}$ y $\vec{C}$ son los componentes del vector, $\vec{X}_p$ es el vector posición de la presa y $\vec{X}$ indica la posición del vector del lobo gris. (Mirjalili, Mohammad

Mirjalili, & Lewis, Grey Wolf Optimizer, 2014)[21]

$$\vec{D_\alpha} = |\vec{C_1} \cdot \vec{X}_\alpha - \vec{X}|, \vec{D_\beta} = |\vec{C_2} \cdot \vec{X}_\beta - \vec{X}|, \vec{D_\delta} = |\vec{C_3} \cdot \vec{X}_\delta - \vec{X}| \tag{24}$$

$$\vec{X_1} = \vec{X_a} - \vec{A_1} \cdot \vec{D_\alpha}, \vec{X_\beta} - \vec{A_2} \cdot \vec{D_\beta}, \vec{X_\delta} - \vec{A_3} \cdot \vec{D_\delta} \tag{25}$$

Los vectores $\vec{A}$ y $\vec{C}$, son calculados:

$$\vec{C} = 2 \cdot \vec{r}_1 \; \vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a} \tag{26}$$

Donde los componentes de $\vec{a}$, son linealmente decrecientes de 2 a 0 y r_1,r_2 son vectores aleatorios $[0, 1]$.

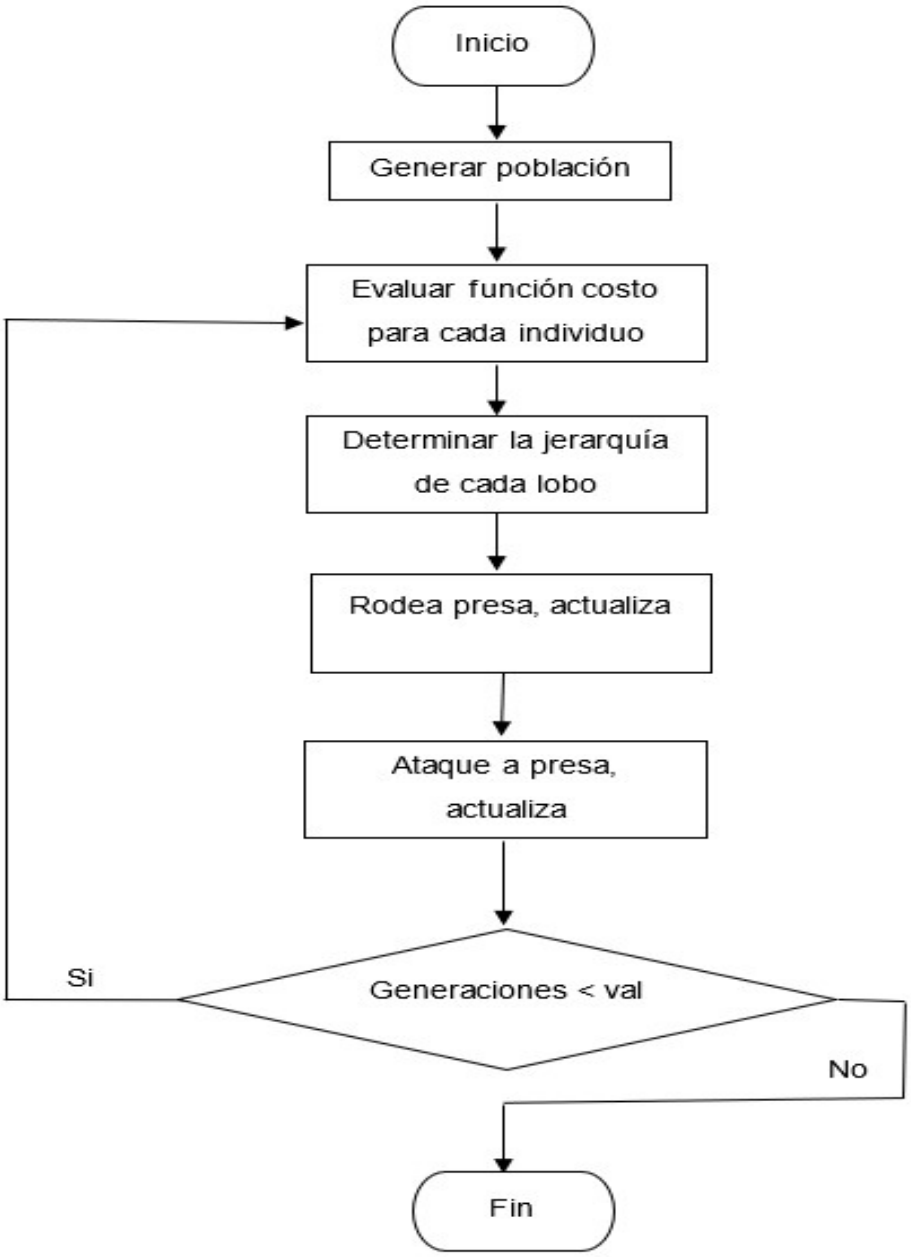

Figura 3.5: Diagrama de flujo del algoritmo GWO, basado en (Mirjalili, Mohammad Mirjalili, & Lewis, Grey Wolf Optimizer, 2014)[21]

3.5.2. Pseudocódigo del GWO

El Pseudocódigo del GWO, se basa principalmente en: La jerarquía social propuesta ayuda al GWO a tener la mejores soluciones en el transcurso de las iteraciones. El mecanismo de cerco propuesto define un círculo en forma de vecindario. Los parámetros A y C ayudan a las soluciones candidatas. El método de búsqueda propuesto permite que las soluciones candidatas localizan la posición de la presa.

Inicializar la población del lobo gris $X_i\,(i = 1, 2, 3, \ldots n)$ Inicializar a A y C calcula los fitness de cada agente de búsqueda X_α=el mejor agente de búsqueda X_β=el segundo agente de búsqueda X_δ=el tercer agente de búsqueda

```
while (t<maximo numero de iteraciones)
for cada búsqueda de agente
Cambiar la posición del agente actual
end for
Cambiar a, A y C
Calcular el fintness de todos los agentes buscados
Cambiar Xa, Xb y Xd
T =t +1
end while
return Xa
```

3.6. Optimización en enjambre de partículas, Particle Swarm Optimization (PSO)

Es un método de búsqueda aleatoria basado en la población desarrollado originalmente a partir de estudios de comportamiento social de las parvadas de aves, desde su creación por Eberhart y Kennedy, se ha convertido en uno de los algoritmos basados en inteligencia de

enjambres más importantes. (K.F., K. s. , & S., 1996)[19], La figura 3.6 muestra el diagrama de flujo del PSO

3.6.1. Modelo matemático del PSO

Supongamos que el enjambre consiste de m partículas alrededor de un espacio. La velocidad y la posición de la partícula en la $k-$ésima dimensión, puede ser calculada:

$$V_{ik}(t+1) = wV_{ik}(t) + C_1 r_1(P_{ik}(t) - X_{ik}(t) + C_2 r_2)(P_{gk}(t) - X_{ik}(t)) \quad (27)$$

$$X_{ik}(t+1) = X_{ik}(t) + V_{ik}(t+1) \quad (28)$$

Donde: w, es el peso inercial, reduce o aumenta la velocidad de la partícula

C_1 y C_2 constantes cognitivas, positivas

r_1 y r_2 vectores de números aleatorios

$V_ik(t+1)$, velocidad de la partícula i en el momento $t+1$, es decir en la nueva velocidad

$V_ik(t)$, la velocidad de la partícula i en el momento t, es decir la velocidad actual

$P_(ik)(t)$, mejor posición en la que ha estado la partícula i hasta el momento

$X_ik(t)$, posición de la partícula en el momento t

$P_gk(t)$, la mejor posición de todo el enjambre hasta el momento t, el mejor valor global

$X_ik(t+1)$, la posición de la partícula i en el momento $t+1$, es decir la nueva posición.

Después de la actualización de posición y velocidad, se evalúa la función fitness y el proceso se repite hasta alcanzar el número de generaciones establecido.

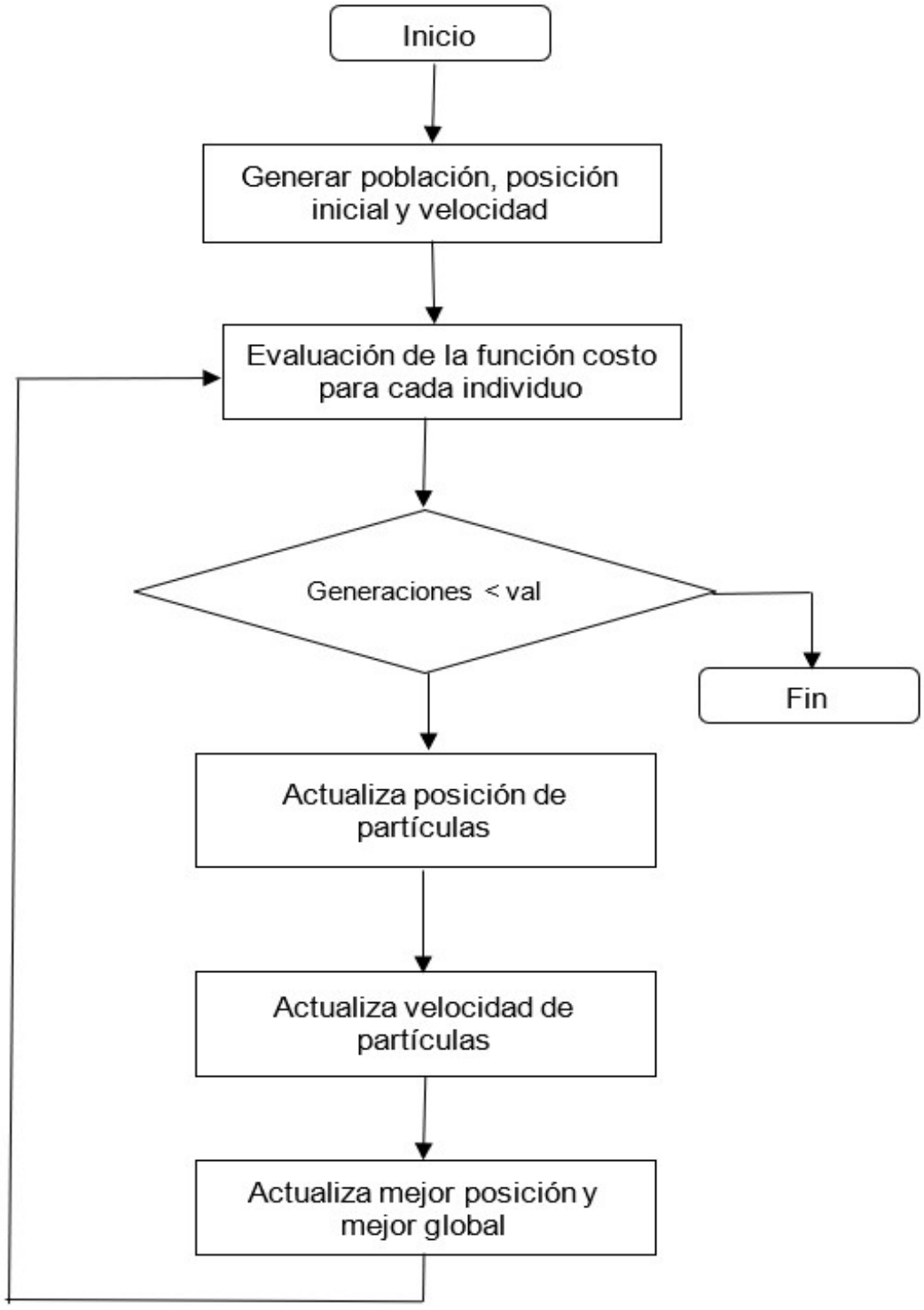

Figura 3.6: Diagrama de flujo del algoritmo PSO, basado en (K.F., K. s. , & S., 1996)[19]

3.6.2. Pseudocódigo del PSO

Un enjambre de partículas actualiza sus posiciones relativas desde iteración a otra, impulsando el algoritmo PSO para debidamente realizar el proceso de búsqueda. Para obtener la solución óptima, cada partícula se mueve hacia su mejor posición personal anterior (P_{best}) y la mejor posición global (g_{best}) en el enjambre. Suponiendo un problema de minimización el pseudocódigo es el siguiente:

```
Inicio
for i = 1 to N, inicia el enjambre al azar
Inicializa la velocidad de las partículas usando una distribución uniforme
Inicializa la posición de las partículas usando una distribución uniforme
Inicializa la mejor posición (Pbest)
end for
Inicializa gbest la posición con el mínimo valor de la fusión fitness
Inicializa la primera iteración
while t < T do
for i = 1 to N do  Itera a través del enjambre
r1, r2 dos vectores independientes son generados aleatoriamente entre 0 y 1
Actualiza la velocidad de la partícula
Actualiza la posición de la partícula
if f(x_iˆt )<f(P_bestˆ(t-1) ) then Si la nueva solución es mejor que la actual
Actualiza la posición de la partícula
end if
end for
Actualiza la mejor posición de la partícula
end while
Fin
```

4. Capítulo 4 Pruebas y resultados

4.1. Introducción

El SHTG es simulado en Matlab y se realiza una identificación de parámetros con los AG, PSO, GWO Y WOA, para cinco parámetros en el sistema, los cuales son: T_w, T_f, f, T_a y e_g. Debido a que el SHTG actúa en un amplio rango de operación en la carga del generador se realizan dos pruebas las cuales son identificación de parámetros con carga y sin carga, en la simulación se consideraron los valores de la amplitud de la perturbación en la velocidad y carga del SHTG respectivamente de 0.04 p.u. y 0.1 p.u. Los coeficientes de la función de transferencia de la turbina para estas dos pruebas (Tian , Changyu, Qi, Yi, & Qiurong, 2018)[24] se muestran en la Tabla 4.1.

Coeficientes de transferencia de la hidroturbina						
Condiciones	ϵ_x	ϵ_y	ϵ_h	ϵ_{qx}	ϵ_{qy}	ϵ_{qh}
Sin carga	-1.0567	0.9080	1.4191	-0.0574	0.7887	0.4571
Con carga	-1.4673	0.7713	1.7179	-0.4901	0.8184	0.7257

Tabla 4.1: Coeficientes de transferencia del sistema hidroturbina bajo dos condiciones de operación

Las constantes usadas en las simulaciones para el controlador PID y el servomecanismo son las que se muestran en la Tabla 4.2

Parámetro	Valor
Constante proporcional, K_p	5.5912
Constante integral, K_I	1.0611
Constante derivativa, K_D	3.2800
Tiempo diferencial, T_{1v}	0.28
Coeficiente de retroalimentación, b_n	0.04
Constante de tiempo del servomecanismo, T_y	0.1

Tabla 4.2: Ganancias del controlador PID y servomecanismo

Los parámetros a identificar del sistema hidráulico y constantes del generador se muestran en la tabla 4.3, los cuales son valores de referencia que se usan para generar las respuesta dinámicas del SHTG y a partir de la respuesta dinámica y la estructura del modelo matemáticos es como se pretende identificar estos valores (Tian , Changyu, Qi, Yi, & Qiurong, 2018)[24].

Parámetro	**Valor**
Sistema hidráulico, constante de tiempo T_w	1.5
Constante de tiempo en la compuerta T_e	0.53
Pérdida de fricción f	0.01
Constante de tiempo de inercia del generador T'_a	12.0
Constante de ajuste del generador e_g	0.4433

Tabla 4.3: Parámetros del sistema hidráulico y generador a identificar en el SHTG

En las simulaciones el vector a identificar es $\theta = [1.5\ 0.53\ 0.01\ 12\ 0.4433]$, el intervalo de búsqueda para los limites inferior y superior se definen por $lb = [1.0\ 0.3\ 0.001\ 10\ 0.2]$ y $ub = [2\ 0.8\ 0.02\ 15\ 0.6]$, respectivamente. El tiempo de simulación es de 30 s con un periodo de muestreo de 0.01 s. El valor de las ganancias, parámetros y constantes garantizan que el sistema se estabilice a partir del proceso transitorio y se puedan tomar los detalles del proceso dinámico del sistema.

4.2. Evaluación del desempeño de los distintos algoritmos evolutivos en condiciones sin carga

En esta sección se reporta la identificación de parámetros para el SHTG sin carga usando cuatro algoritmos evolutivos GA, PSO, GWO y WOA. Se realizan 20 repeticiones y 50 evoluciones y una población de 40 individuos. Los parámetros para cada algoritmo evolutivo se muestran en la Tabla 4.4, estos son los valores típicos con los cuales se reportan para su mejor desempeño.

Algoritmo Evolutivo	**Parámetros del algoritmo evolutivo.**
GA	Cruce por selección, mutación = 3 %
PSO	C1 = 2, C2 = 2, Inercia = [0.9 0.2]
GWO	Inercia = [2 0]
WOA	Posición, p = 0.5

Tabla 4.4: Valores típicos de configuración en los AE

La Tabla 4.5 muestra los parámetros identificados por los AE así como su parámetro de error (PE) para cada uno de los valores identificados

		El mejor valor de los parámetros identificados (20 repeticiones)							
Parámetro	Valor real	GA		PSO		GWO		WOA	
		$\hat{\theta}_i$	PE	$\hat{\theta}_i$	PE	$\hat{\theta}_i$	PE	$\hat{\theta}_i$	Pe
T_w	1.5	2.0000	0.3333	2.0000	0.3333	2.0000	0.3333	2.0000	0.3333
t_e	0.53	0.3000	0.433962	0.3000	0.4339	0.3000	0.4339	0.3000	0.4339
f	0.01	0.0200	1.0000	0.0150	0.5000	0.0200	1.0000	0.0200	1.0000
$T'a$	12.0	13.2038	0.1003	13.6396	0.1366	13.1985	0.0998	13.1887	0.0990
e_g	0.4433	0.3250	0.3640	0.3724	0.1598	0.3112	0.2979	0.3092	0.3025

Tabla 4.5: El mejor valor de los parámetros identificados

La Tabla 4.6 muestra un estadístico del desempeño de los AE en la función costo usando el

MSE, considerando el mejor valor, el peor, el promedio y la desviación estándar para los AE evaluados.

Estadística	GA	PSO	GWO	WOA
Mejor	0.1718	0.234230747299075	0.171603400362333	0.1716
Peor	0.2110	0.234230747580013	0.173160895614182	0.3384
Promedio	0.1888	0.234230747318169	0.171976647376244	0.1946
SD	0.0134	6.232034183224600e-11	4.700575333439131e-04	0.0391

Tabla 4.6: Valores estadísticos del MSE de la función costo para el caso sin carga

Las figura 4.1 muestra el comportamiento de la función objetivo con respecto a las evoluciones en GA.

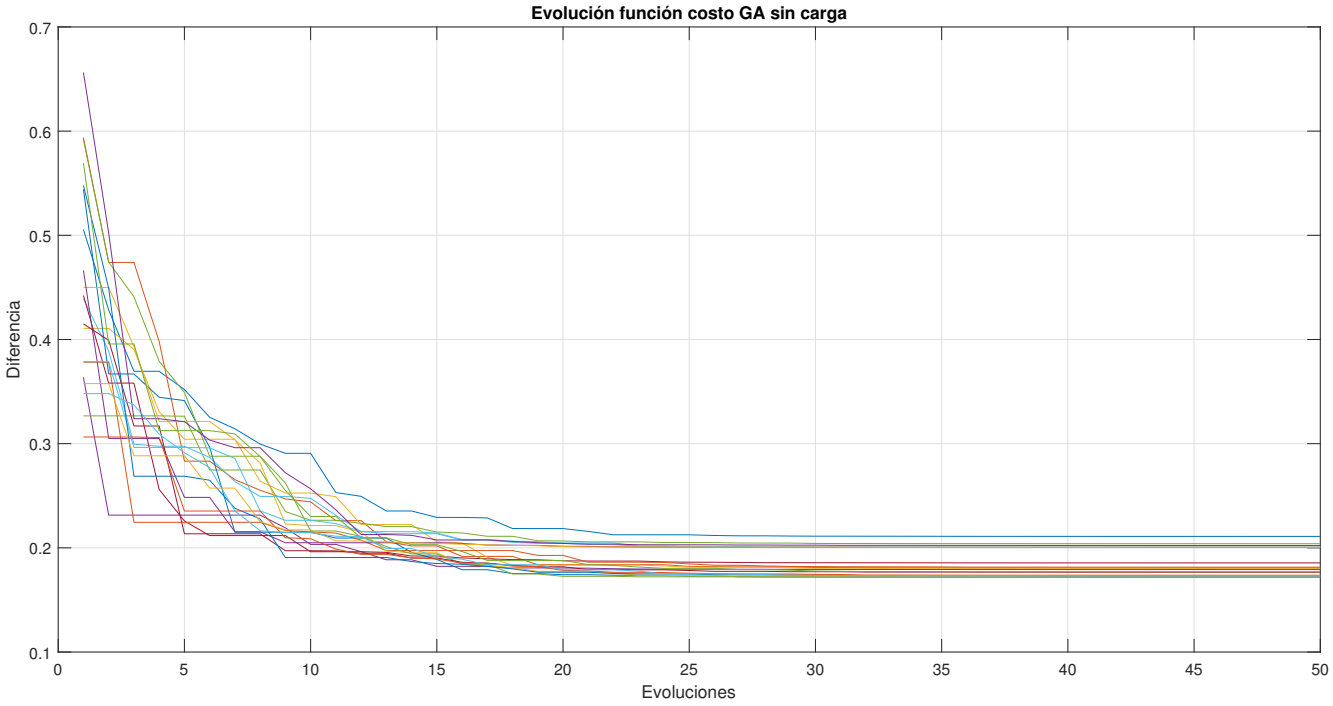

Figura 4.1: Con 20 repeticiones con 50 evoluciones del GA para la identificación de parámetros del SHTG sin carga

De estas la mejor opción en donde se tiene la desviación mínima entre el valor real y el valor estimado se muestra en la figura 4.2

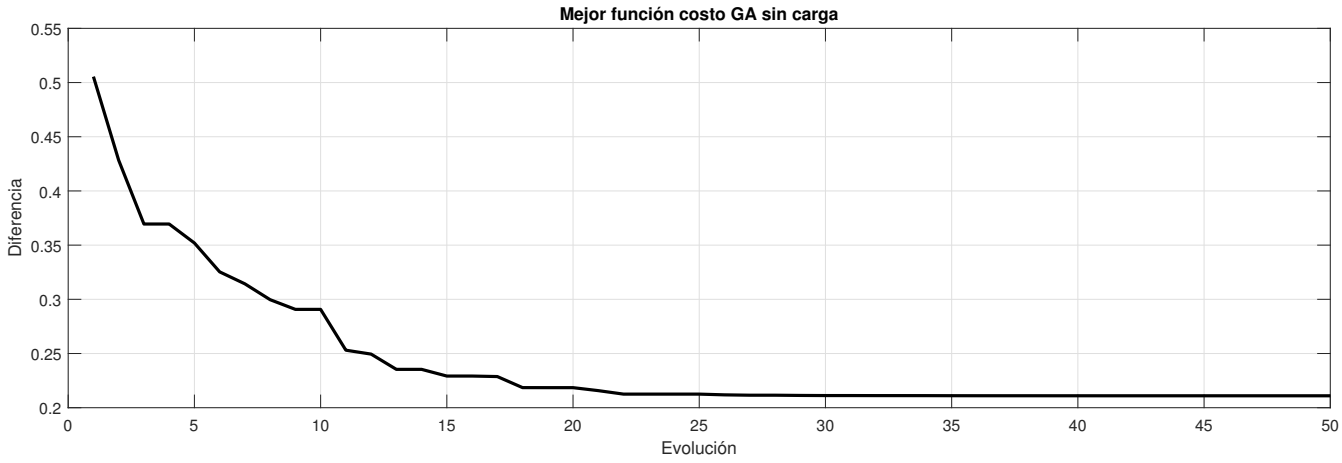

Figura 4.2: Evaluación de la función costo con la menor diferencia entre el valor real y los valores estimados con el algoritmo GA

El comportamiento del sistema sin carga con el algoritmo PSO, se muestra en la figura 4.3

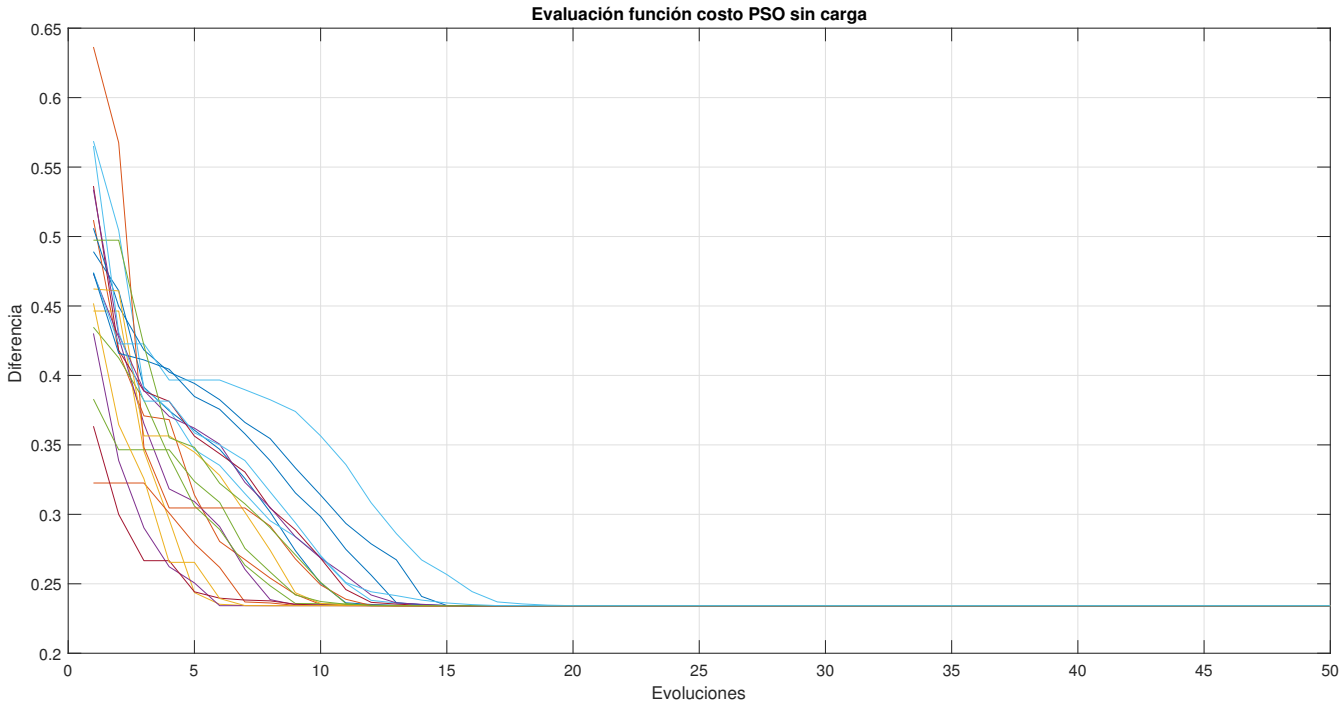

Figura 4.3: Evaluación de la función costo con la menor diferencia entre el valor real y los valores Evaluación de la función costo 20 repeticiones con 50 evoluciones del algoritmo PSO para la identificación de parámetros del SHTG sin carga

De estas la mejor opción en donde se tiene la desviación mínima entre el valor real y el valor

estimado se muestra en la figura 4.4.

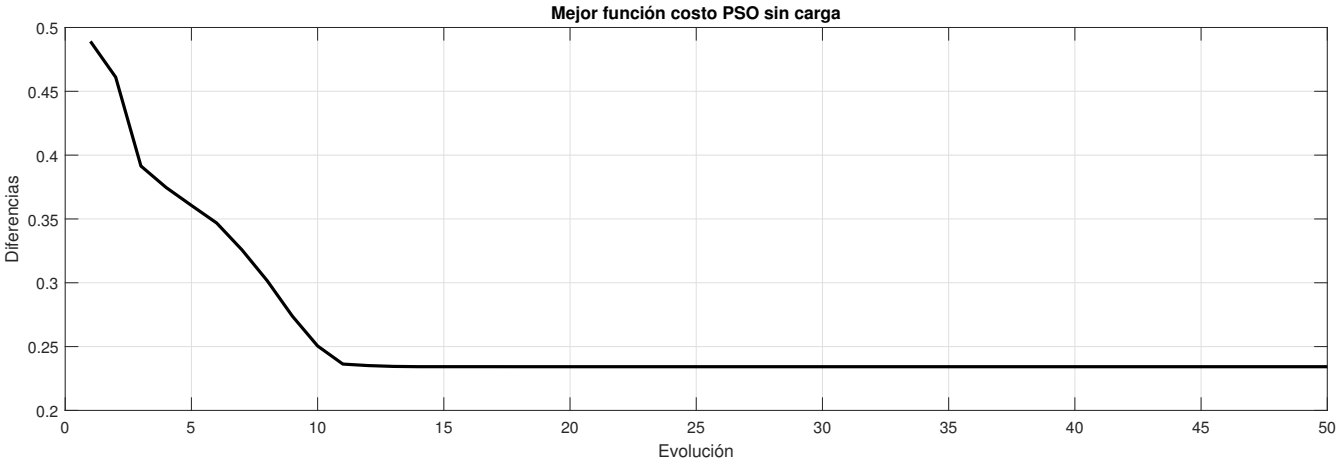

Figura 4.4: Evaluación de la función costo con la menor diferencia entre el valor real y los valores estimados con el método PSO

El comportamiento del sistema con el algoritmo GWO, se muestra en la figura 4.5

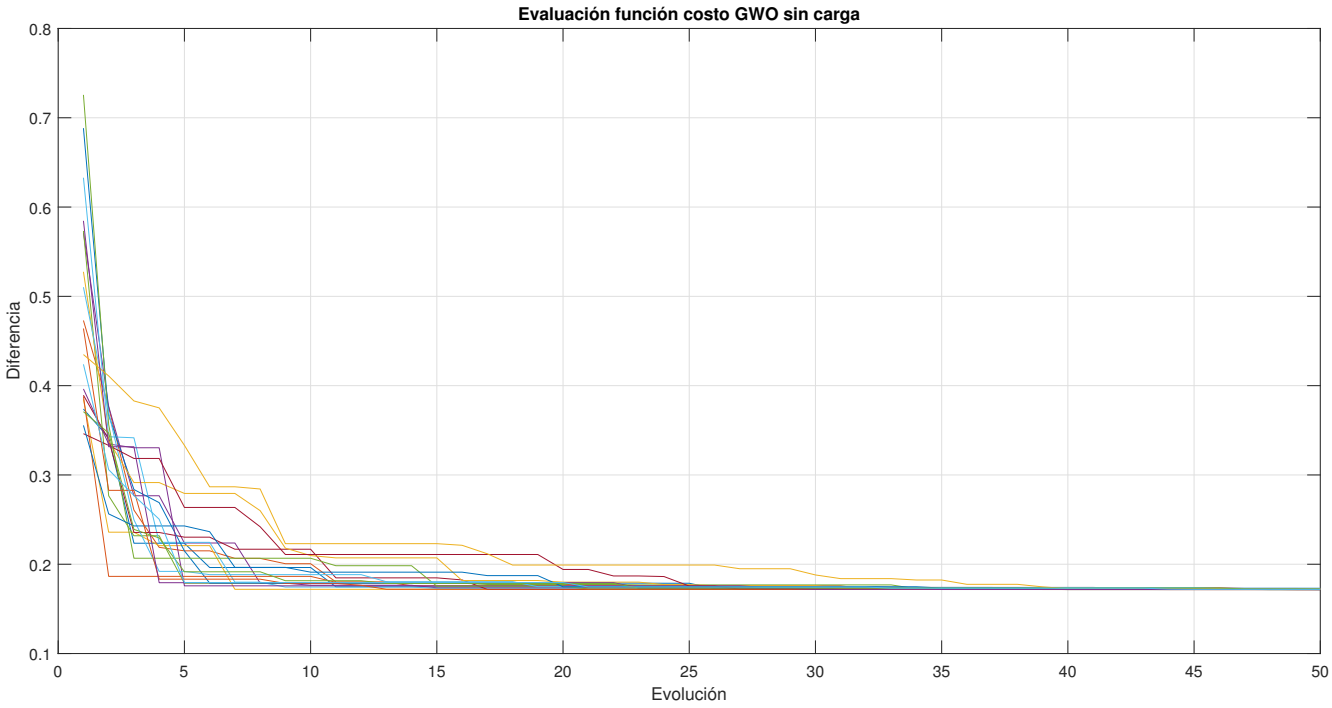

Figura 4.5: Con 20 repeticiones con 50 evoluciones del algoritmo GWO para la identificación de parámetros del SHTG sin carga

De estas la mejor opción en donde se tiene la desviación mínima entre el valor real y el valor estimado se muestra en la figura 4.6

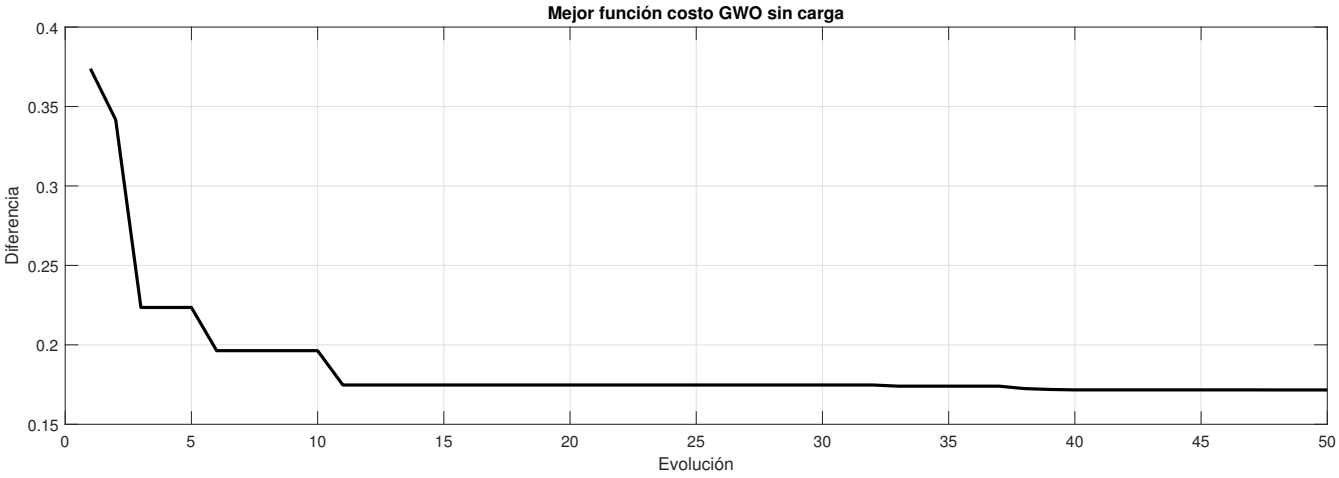

Figura 4.6: Evaluación de la función costo con la menor diferencia entre el valor real y los valores estimados con el método GWO

El comportamiento del sistema sin carga con el algoritmo WOA, se muestra en la figura 4.7

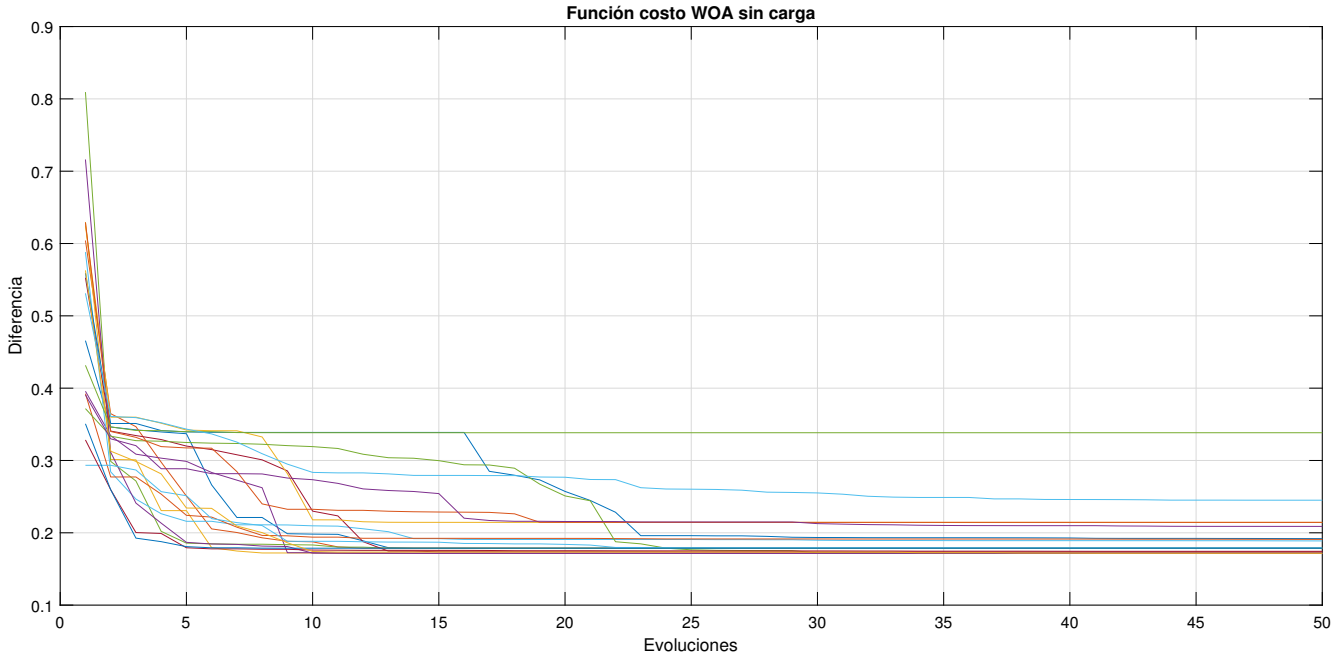

Figura 4.7: Con 20 repeticiones con 50 evoluciones del algoritmo WOA para la identificación de parámetros del SHTG sin carga

De estas la mejor opción en donde se tiene la desviación mínima entre el valor real y el valor estimado se muestra en la figura 4.8

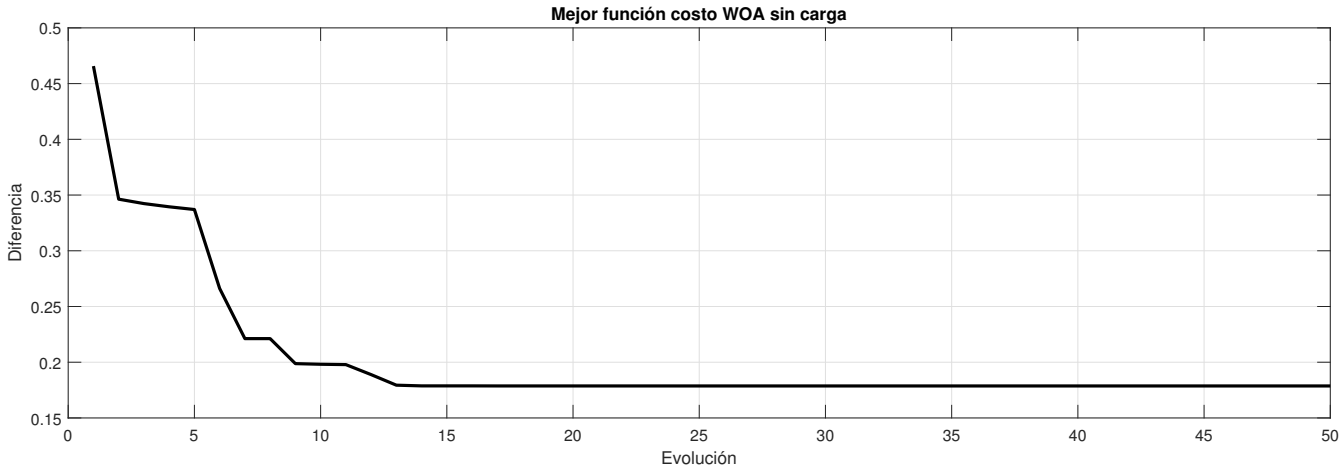

Figura 4.8: Evaluación de la función costo con la menor diferencia entre el valor real y los valores estimados con el método WOA

Dentro de los resultados obtenidos se realizó la simulación del sistema SHTG con los cuatro algoritmos evolutivos utilizados para comparar la respuesta de la salida en las salidas de velocidad de la turbina, en el par del generador y porcentaje de apertura de la compuerta en el sistema primeramente sin carga, los cuales se muestran a continuación.

En el caso del algoritmo genético (GA) sin carga los resultados se muestran en las figuras 4.9, 4.10 y , 4.11 comportamiento de la salida del sistema, velocidad, par y porcentaje respectivamente, como se observa en las figuras el comportamiento de las salidas tiende a estabilizarse al transcurrir un determinado tiempo indicando que cada una de las salidas respectivas se encuentra dentro de los limites de un valor establecido en el estado estacionario, en todos los casos se muestra como las salidas tienen un valor de sobre impulso y un tiempo de establecimiento.

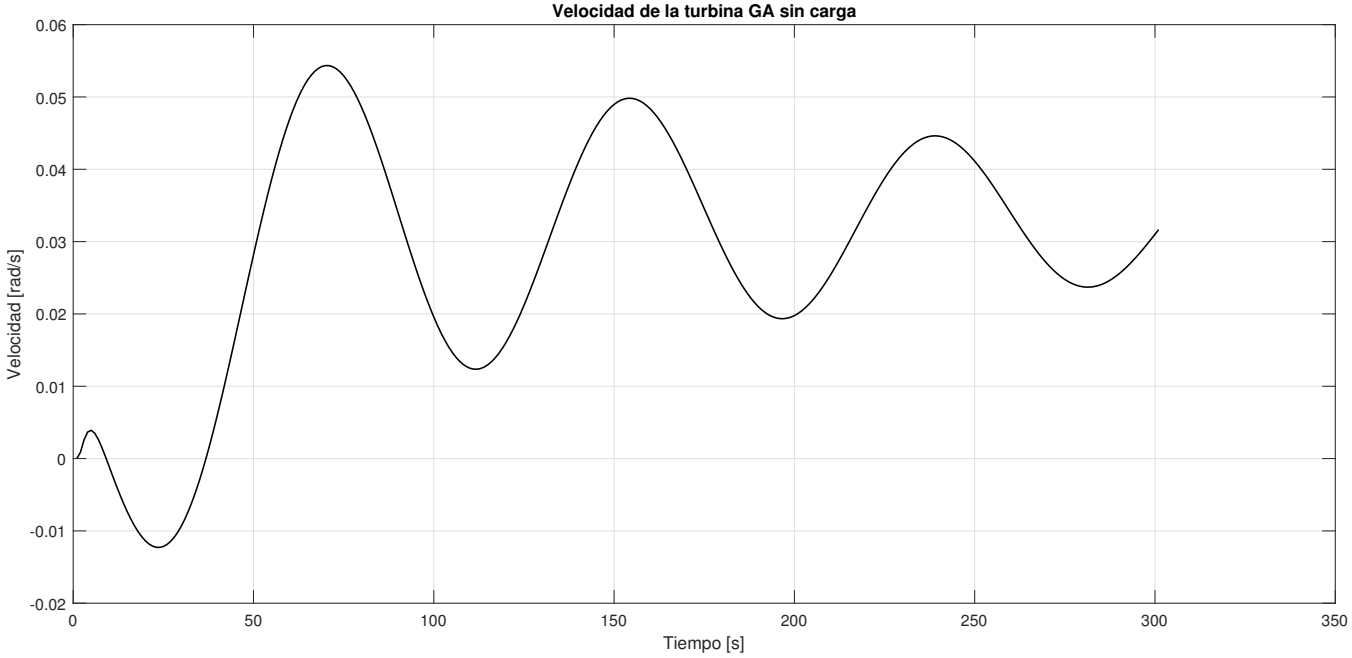

Figura 4.9: Salida velocidad de la turbina

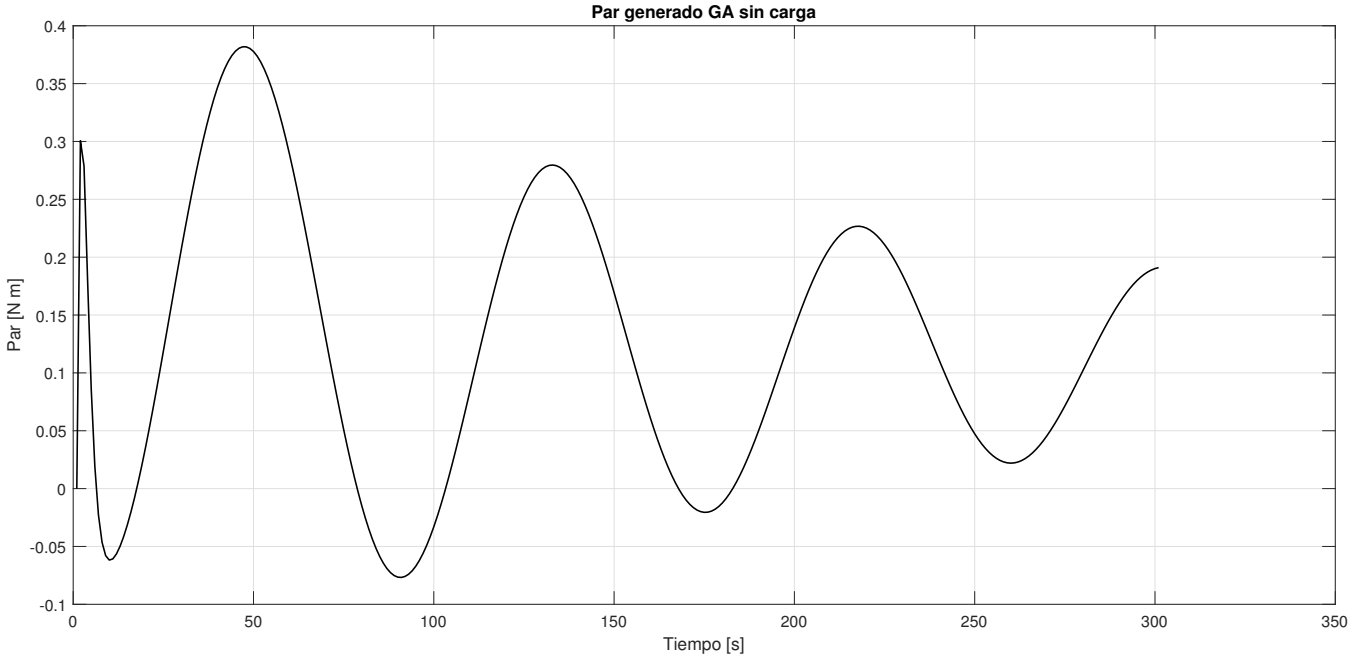

Figura 4.10: Salida Par generador

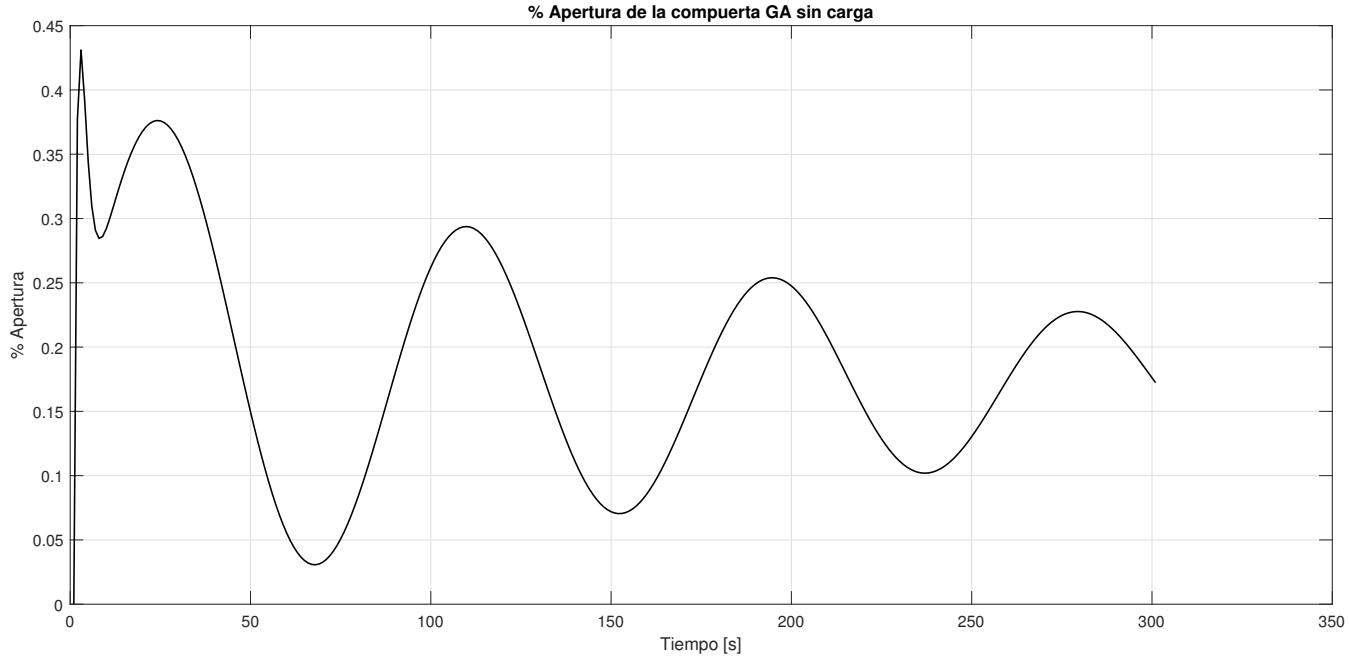

Figura 4.11: Salida porcentaje de apertura de la compuerta

El comportamiento del SHGT con el algoritmo enjambre de abejas (PSO) sin carga los resultados se muestran en las figuras 4.12, 4.13 y , 4.14, como se observa al igual que el genético las salidas de velocidad de la turbina, par del generador y porcentaje de apertura tienen a un estable con respecto del tiempo.

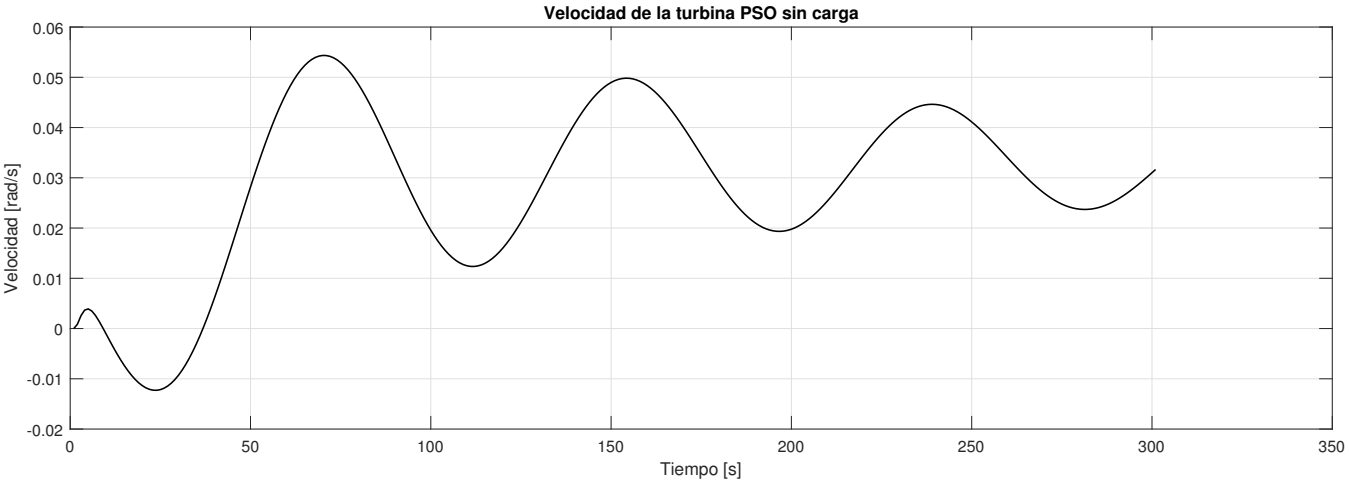

Figura 4.12: Salida velocidad de la turbina

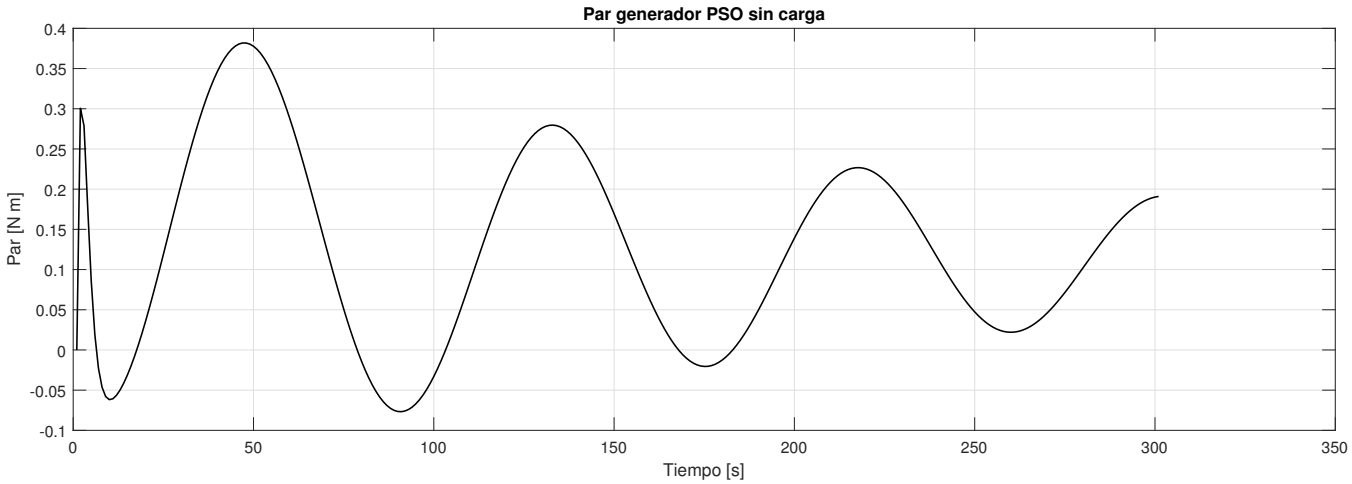

Figura 4.13: Salida Par generador

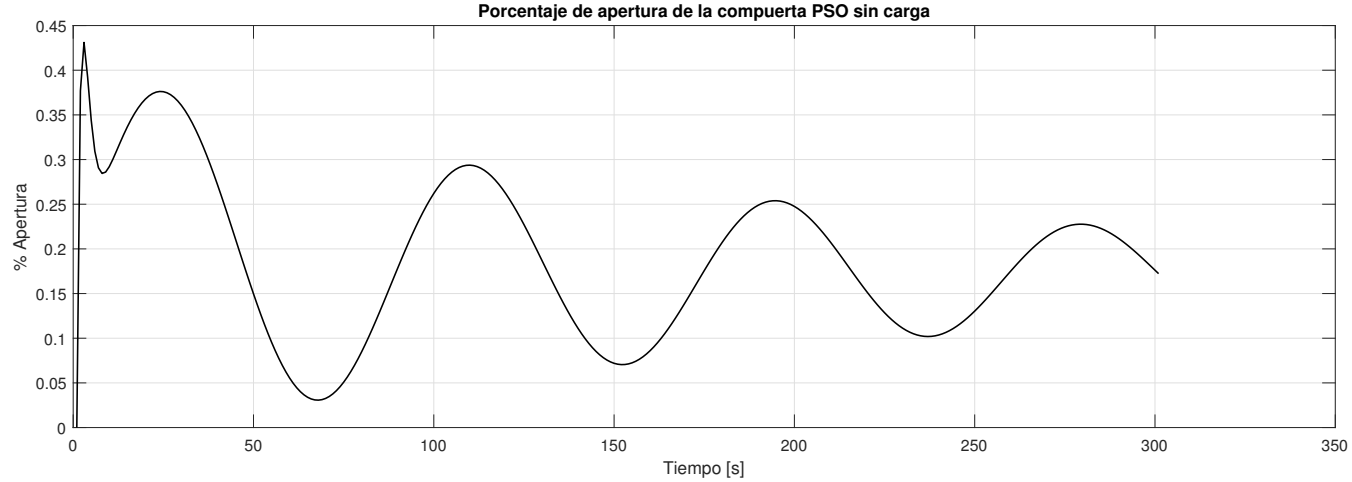

Figura 4.14: Salida porcentaje de apertura de la compuerta

El comportamiento del SHGT con el algoritmo lobos grises (GWO) sin carga los resultados se muestran en las figuras 4.15, 4.16 y , 4.17, como se observa al igual que el genético las salidas de velocidad de la turbina, par del generador y porcentaje de apertura tienen a un estable con respecto del tiempo.

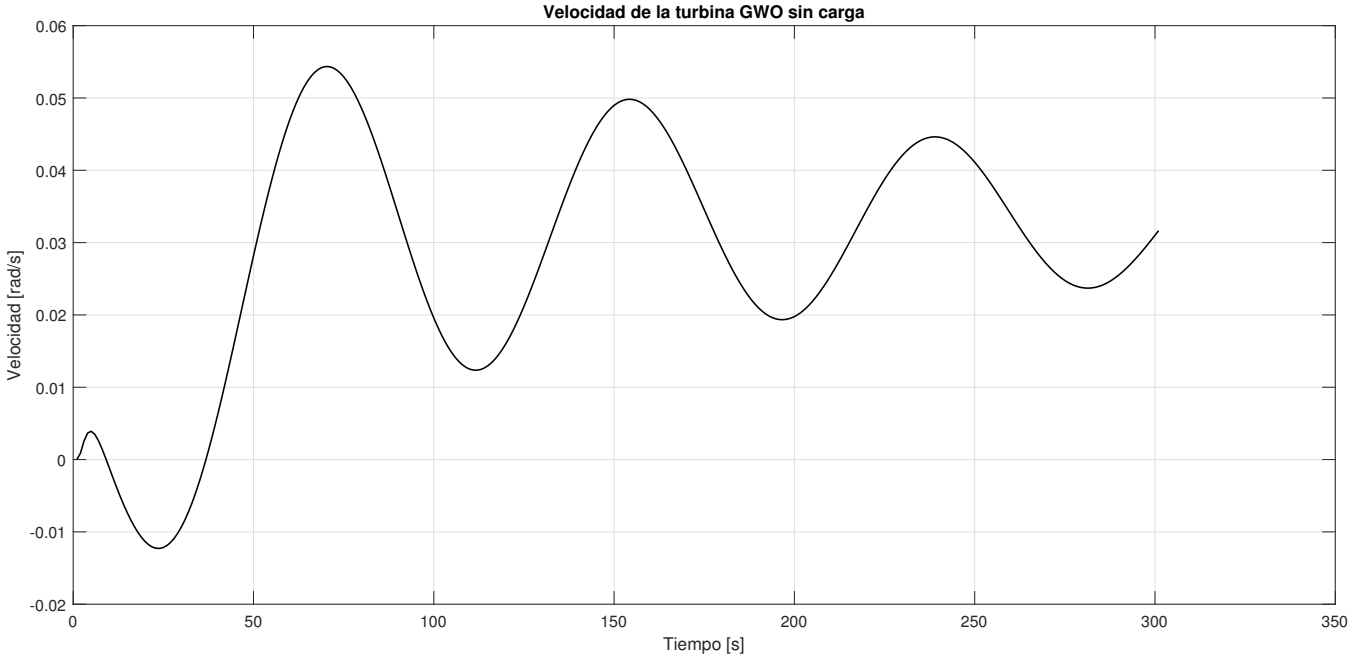

Figura 4.15: Salida velocidad de la turbina

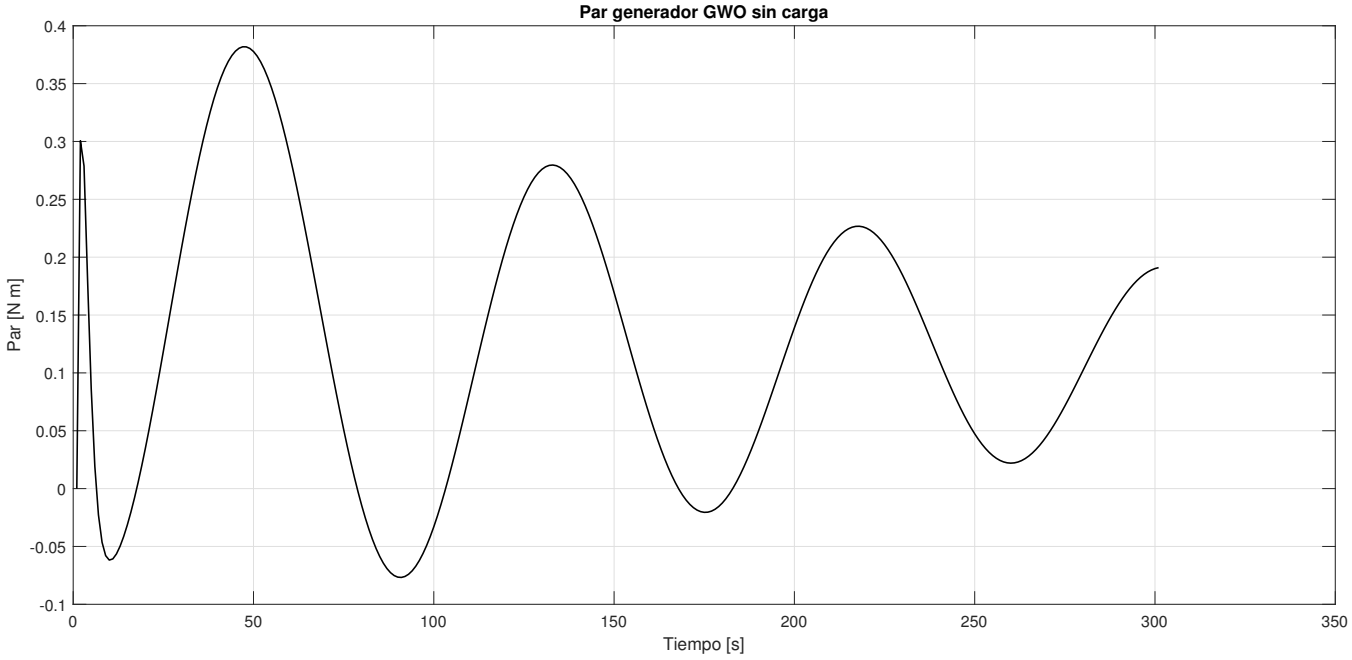

Figura 4.16: Salida Par generador

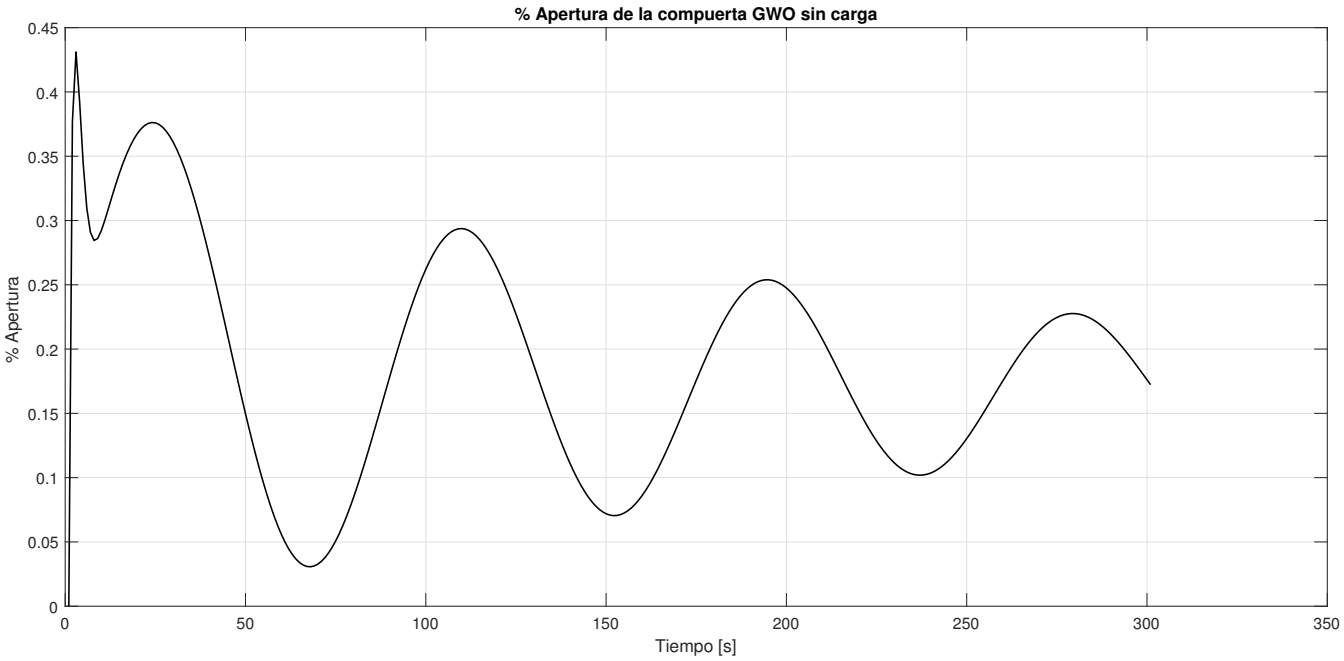

Figura 4.17: Salida porcentaje de apertura de la compuerta

Comportamiento del SHGT con el algoritmo ballenas (WOA) sin carga los resultados se muestran en las figuras 4.18, 4.19 y , 4.20, como se observa al igual que el genético las salidas de velocidad de la turbina, par del generador y porcentaje de apertura tienen a un estable con respecto del tiempo.

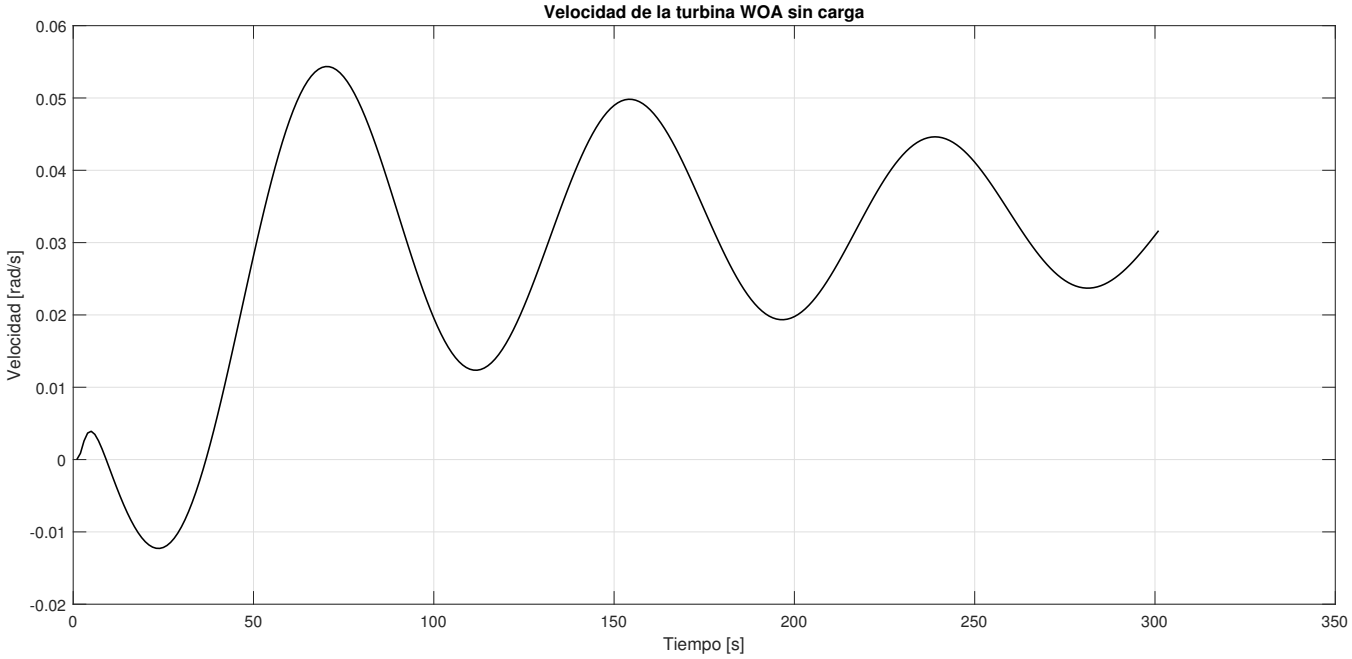

Figura 4.18: Salida velocidad de la turbina

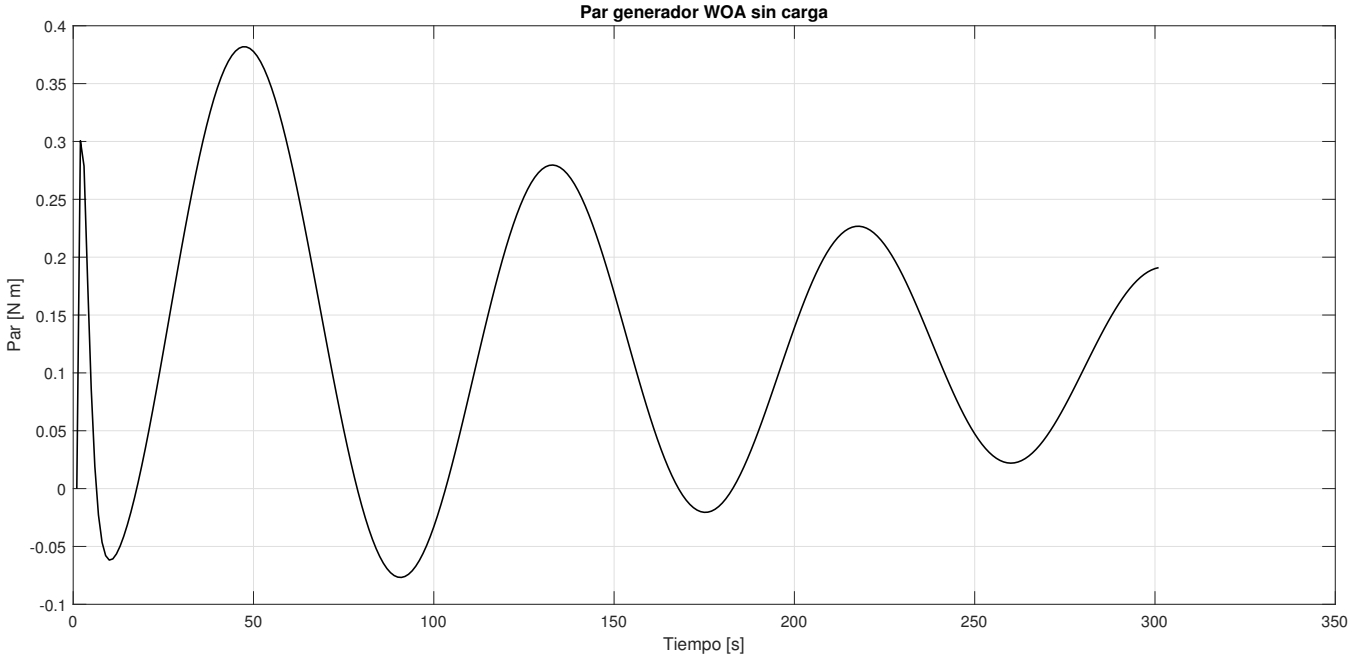

Figura 4.19: Salida Par generador

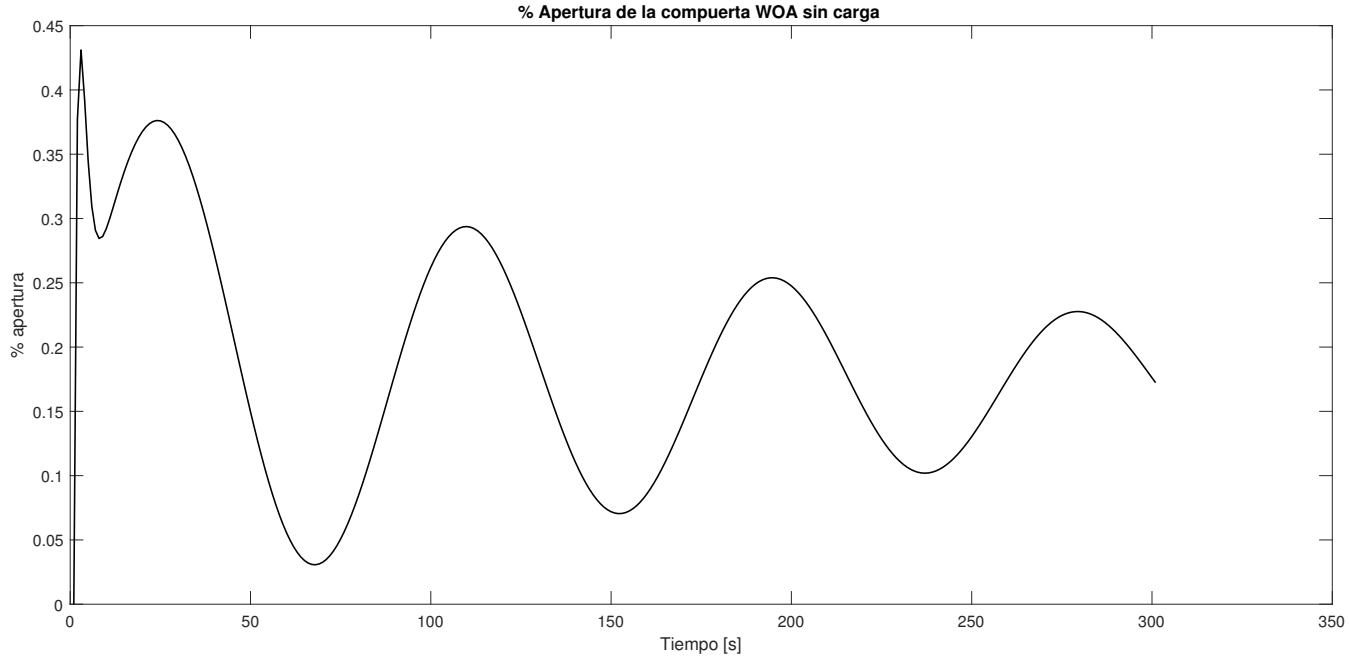

Figura 4.20: Salida porcentaje de apertura de la compuerta

4.3. Evaluación del desempeño de los distintos algoritmos evolutivos en condiciones con carga

La Tabla 4.7 muestra los parámetros identificados por los AE así como su parámetro de error (PE) para cada uno de los valores identificados

		El mejor valor de los parámetros identificados (20 repeticiones)							
Parámetro	Valor real	GA		PSO		GWO		EHO	
		$\hat{\theta}_i$	PE	$\hat{\theta}_i$	PE	$\hat{\theta}_i$	PE	$\hat{\theta}_i$	Pe
T_w	1.5	1.4997	2e-04	1.4999	1.066e-05	1.5013	0.0089	1.5138	0.0092
T_e	0.53	0.5491	0.0360	0.5304	0.0009	0.5070	0.0433	0.7728	0.4581
f	0.01	0.0104	0.04	0.0100	0.0019	0.0086	0.1359	0.0088	0.1200
$T'a$	12.0	11.9945	4.583e-04	11.9998	1.565e-05	12.0392	0.0032	12.1634	0.0136
e_g	0.4433	0.4422	2.481e-03	0.4434	0.0004	0.4391	0.0094	0.5113	0.1533

Tabla 4.7: El mejor valor de los parámetros identificados

La Tabla 4.8 muestra un estadístico del desempeño de los AE en la función costo usando el MSE, considerando el mejor valor, el peor, el promedio y la desviación estándar para los AE evaluados.

Estadística	GA	PSO	GWO	WOA
Mejor	0.1718	0.234730724790775	0.174160348623323	0.1716
Peor	0.2110	0.2347307247580133	0.17316953641482	0.3384
Promedio	0.1888	0.234730724731869	0.174196647376244	0.1946
SD	0.0134	6.22830418232406e-11	4.700575333493134e-04	0.0391

Tabla 4.8: Valores estadísticos del MSE de la función costo para el caso con carga

Figura 4.21 Con 20 repeticiones con 50 evoluciones del GA para la identificación de parámetros del SHTG con carga.

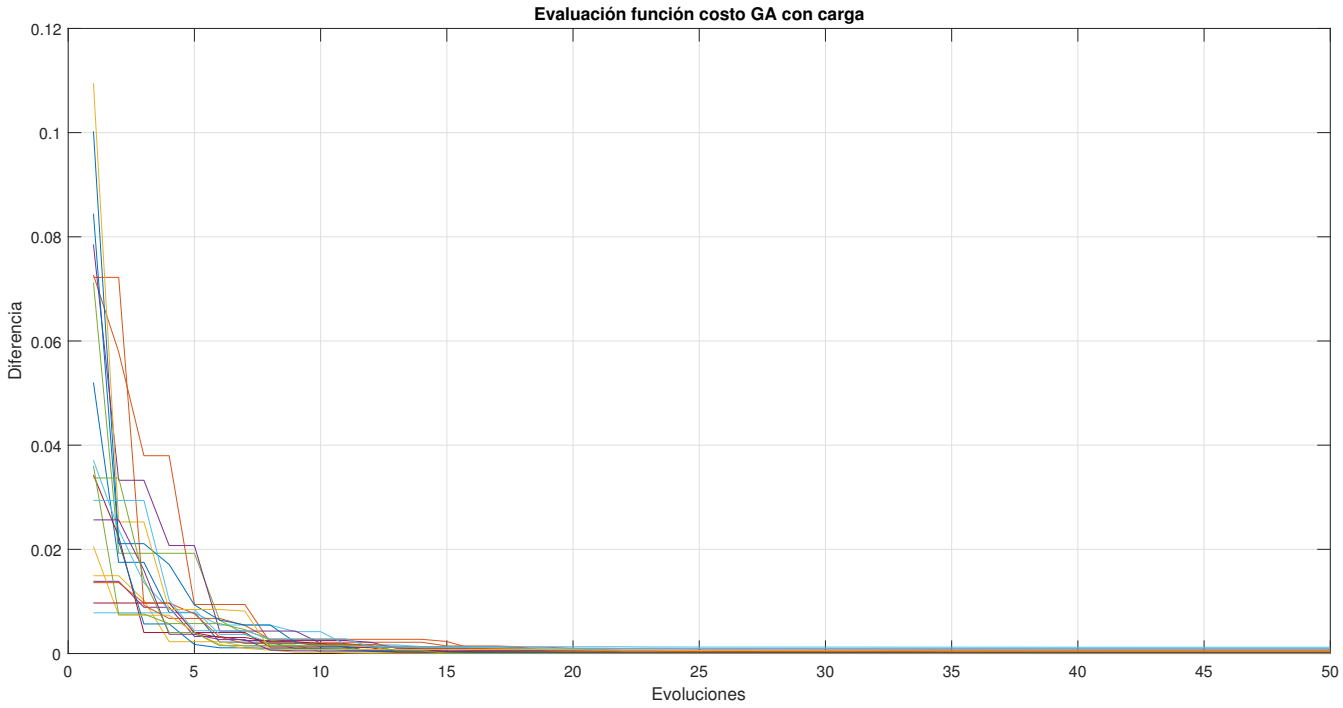

Figura 4.21: Evaluación de la función costo 20 repeticiones con 50 evoluciones del algoritmo GA para la identificación de parámetros del SHTG con carga

De estas la mejor opción en donde se tiene la desviación mínima entre el valor real y el valor

estimado se muestra en la figura 4.28

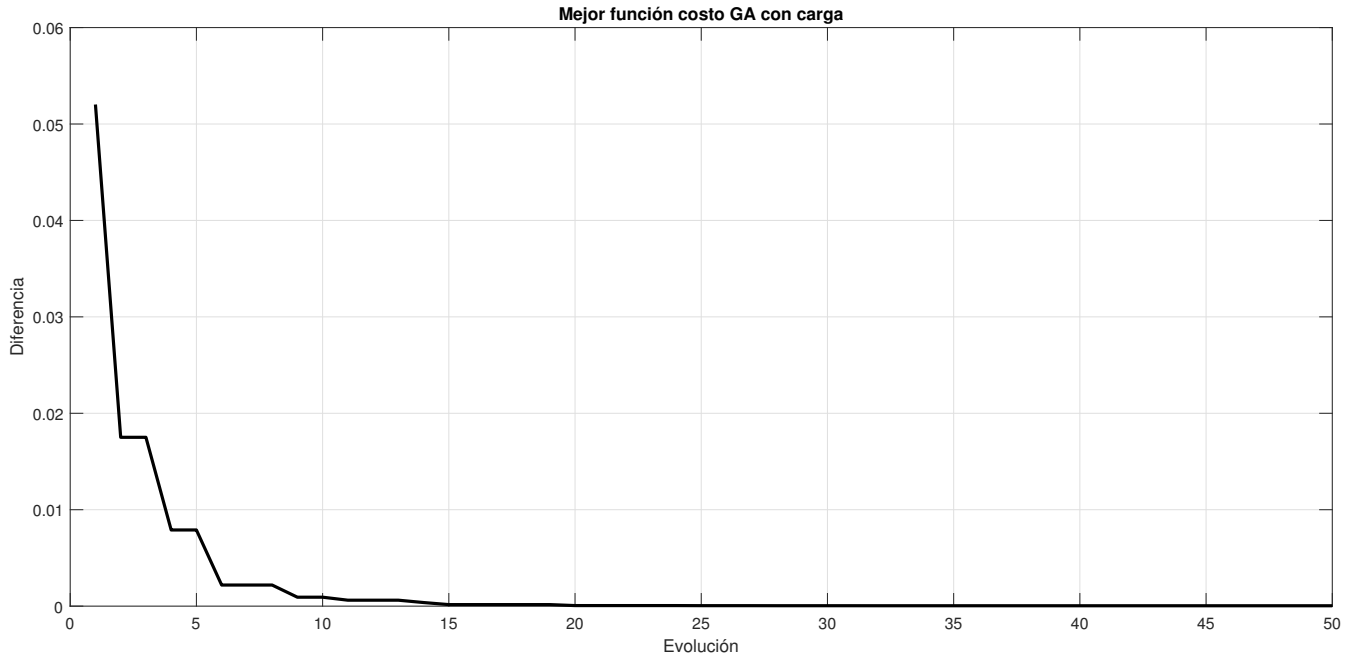

Figura 4.22: Evaluación de la función costo con la menor diferencia entre el valor real y los valores estimados con el método GA

El comportamiento del sistema con carga con el algoritmo PSO, se muestra en la figura 4.23

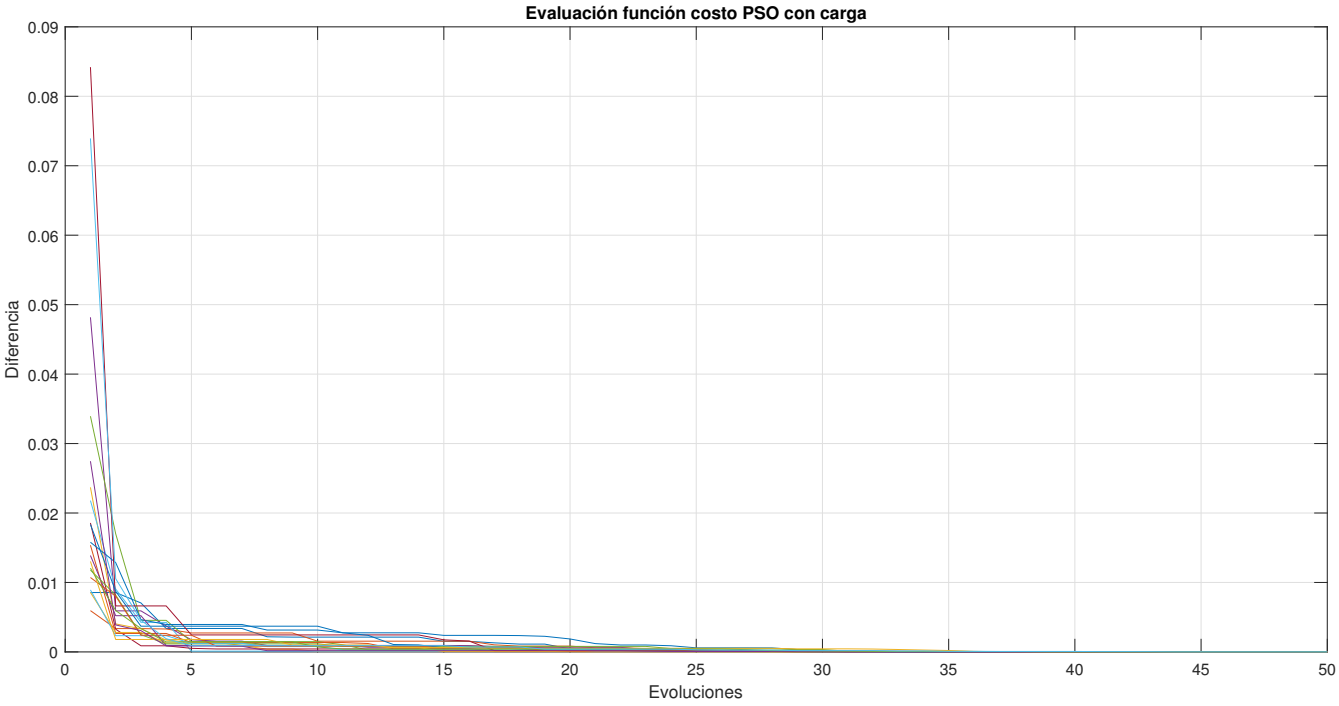

Figura 4.23: Evaluación de la función costo 20 repeticiones con 50 evoluciones del algoritmo PSO para la identificación de parámetros del SHTG con carga

De estas la mejor opción en donde se tiene la desviación mínima entre el valor real y el valor estimado se muestra en la figura 4.24

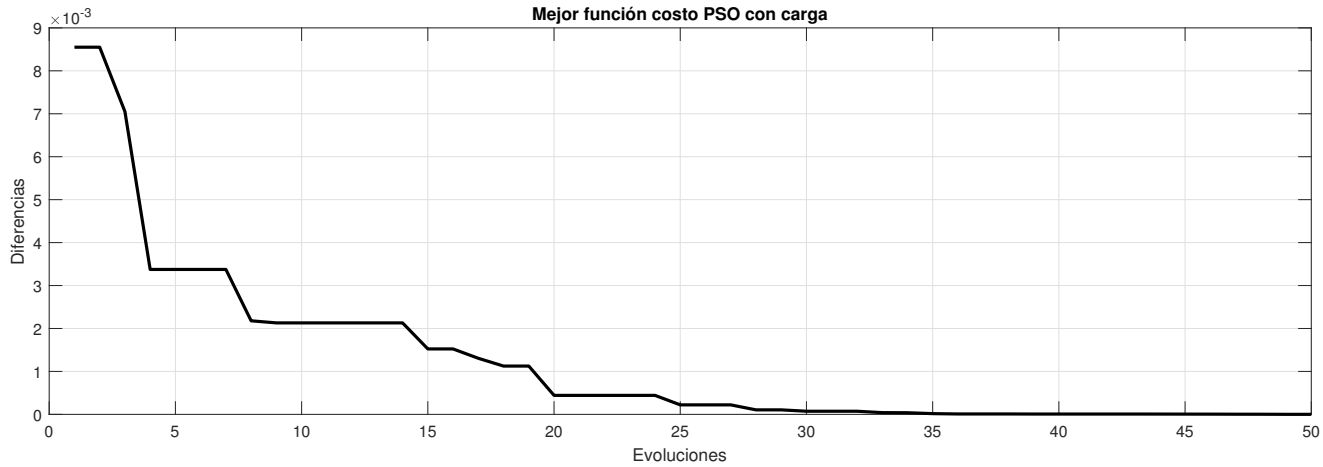

Figura 4.24: Evaluación de la función costo con la menor diferencia entre el valor real y los valores estimados con el método PSO

El comportamiento del sistema con el algoritmo GWO con carga, se muestra en la figura 4.25

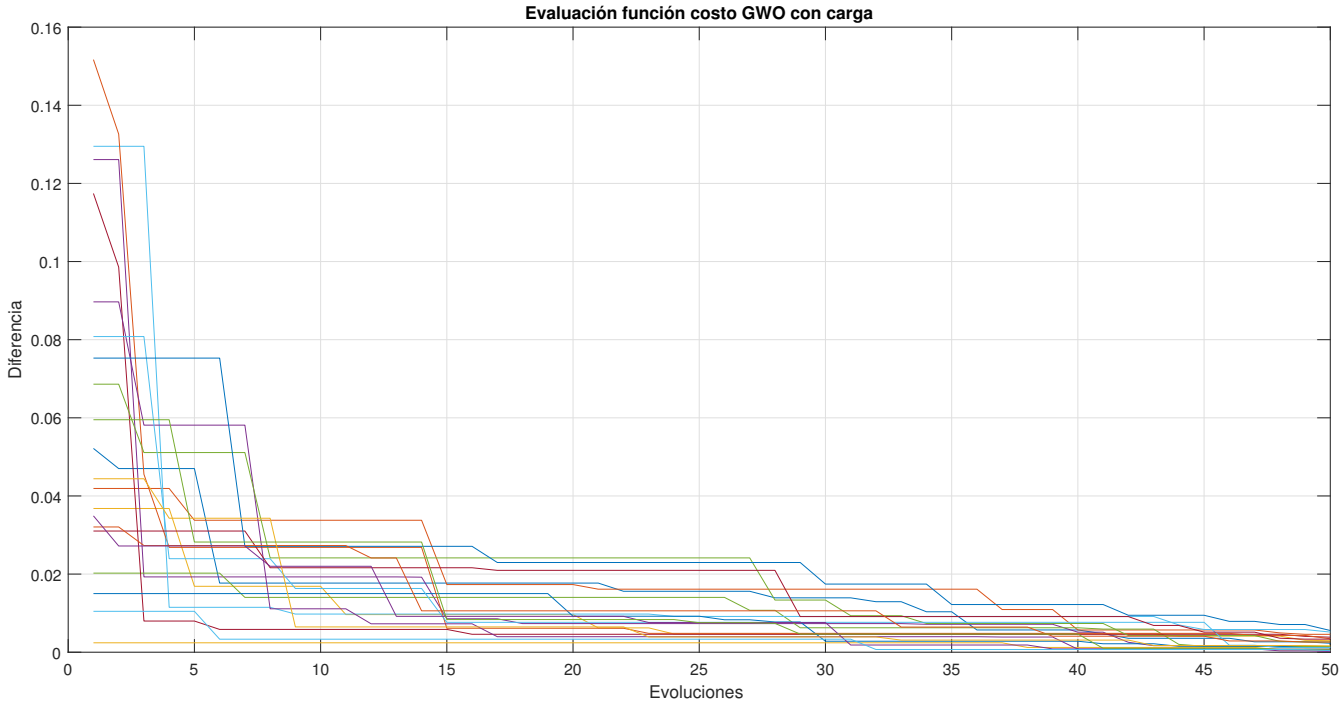

Figura 4.25: Con 20 repeticiones con 50 evoluciones del algoritmo GWO para la identificación de parámetros del SHTG sin carga

De estas la mejor opción en donde se tiene la desviación mínima entre el valor real y el valor estimado se muestra en la figura 4.26

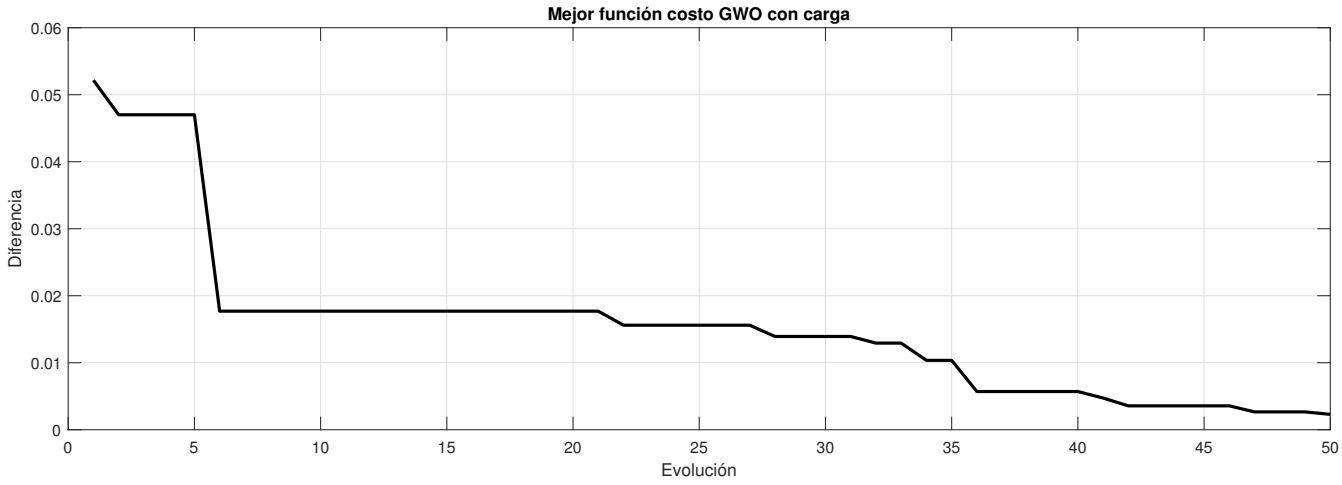

Figura 4.26: Evaluación de la función costo con la menor diferencia entre el valor real y los valores estimados con el método GWO con carga

El comportamiento del sistema sin carga con el algoritmo WOA, se muestra en la figura 4.27

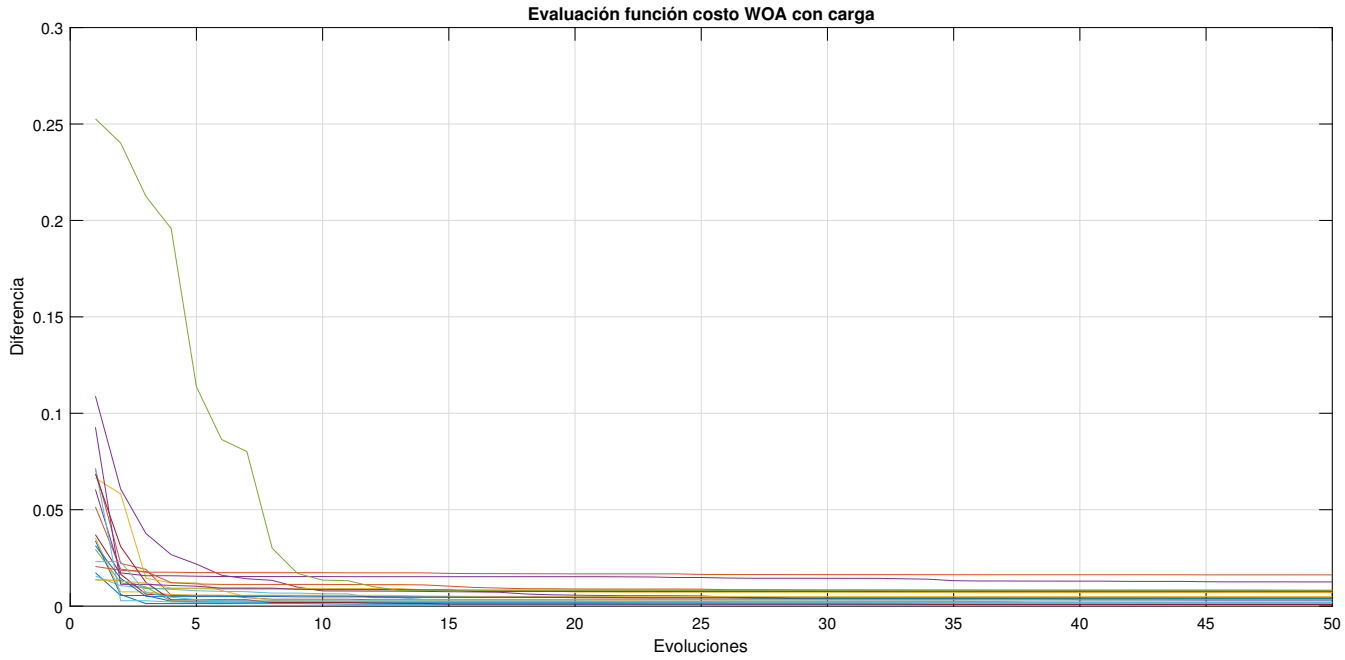

Figura 4.27: Con 20 repeticiones con 50 evoluciones del algoritmo WOA para la identificación de parámetros del SHTG con carga

De estas la mejor opción en donde se tiene la desviación mínima entre el valor real y el valor estimado se muestra en la figura 4.28

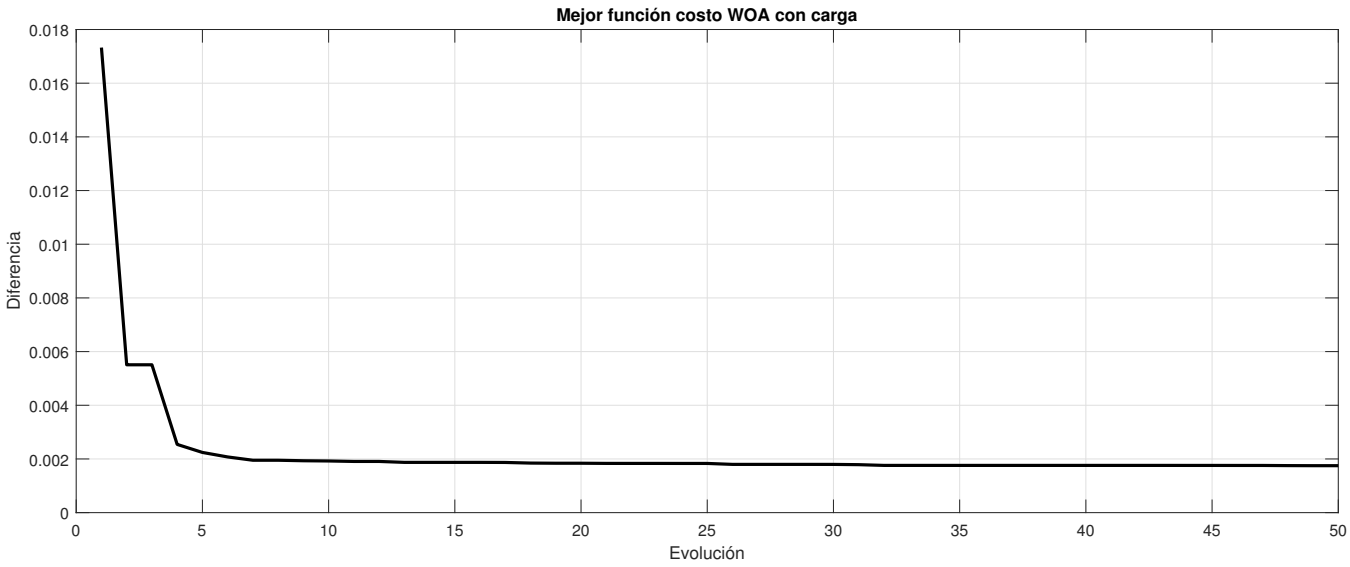

Figura 4.28: Evaluación de la función costo con la menor diferencia entre el valor real y los valores estimados con el método WOA con carga

Se realizó la simulación del sistema SHTG con los cuatro algoritmos evolutivos utilizados para comparar la respuesta de la salida en las salidas de velocidad de la turbina, en el par del generador y porcentaje de apertura de la compuerta en el sistema primeramente con carga, los cuales se muestran a continuación.

Comportamiento del sistema SHTG con el algoritmo genético (GA) con carga, se muestra en la figura 4.29 salida de la velocidad de la turbina, en la figura 4.30 salida del par del generador y finalmente en la figura 4.31 el porcentaje de apertura de la compuerta, observándose como las respectivas salidas tienden a un valor estable una vez transcurrido el estado transitorio del sistema.

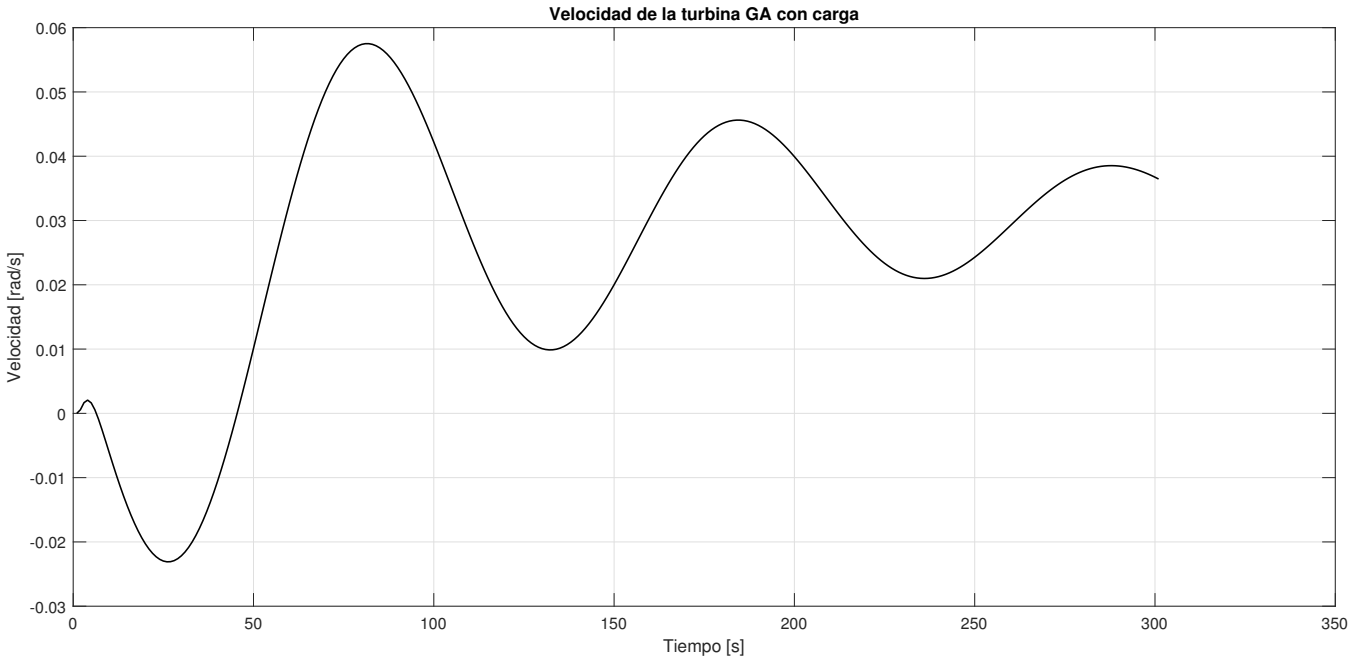

Figura 4.29: Salida velocidad de la turbina

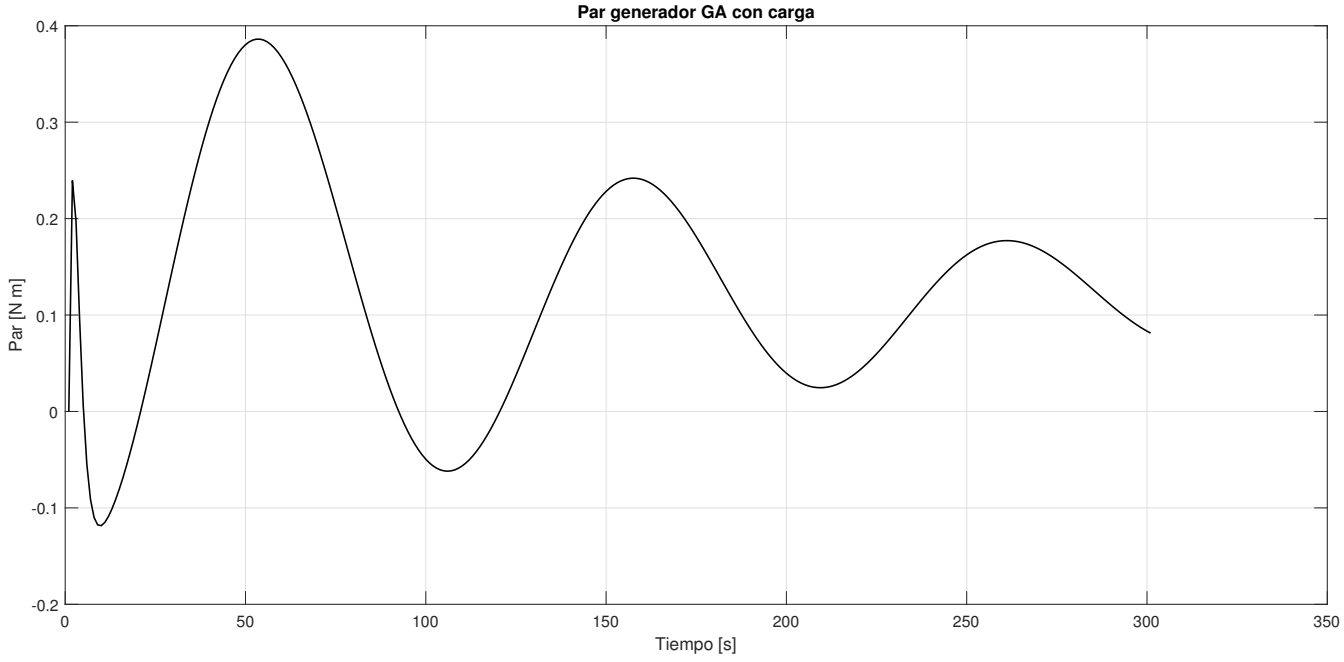

Figura 4.30: Salida Par generador

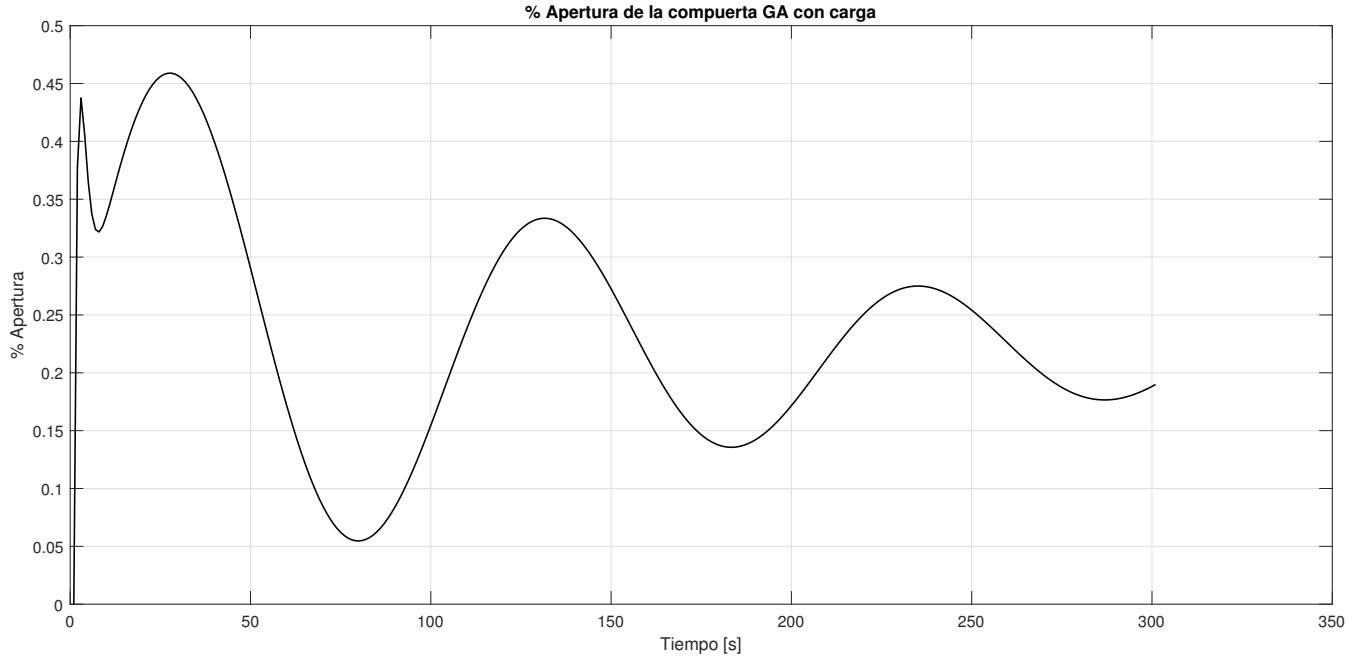

Figura 4.31: Salida porcentaje de apertura de la compuerta

Comportamiento del sistema SHTG con el algoritmo enjambre de abejas (PSO) con carga, se muestra en la figura 4.32 salida de la velocidad de la turbina, en la figura 4.33 salida del par del generador y finalmente en la figura 4.34 el porcentaje de apertura de la compuerta, observándose como las respectivas salidas tienden a un valor estable una vez transcurrido el estado transitorio del sistema.

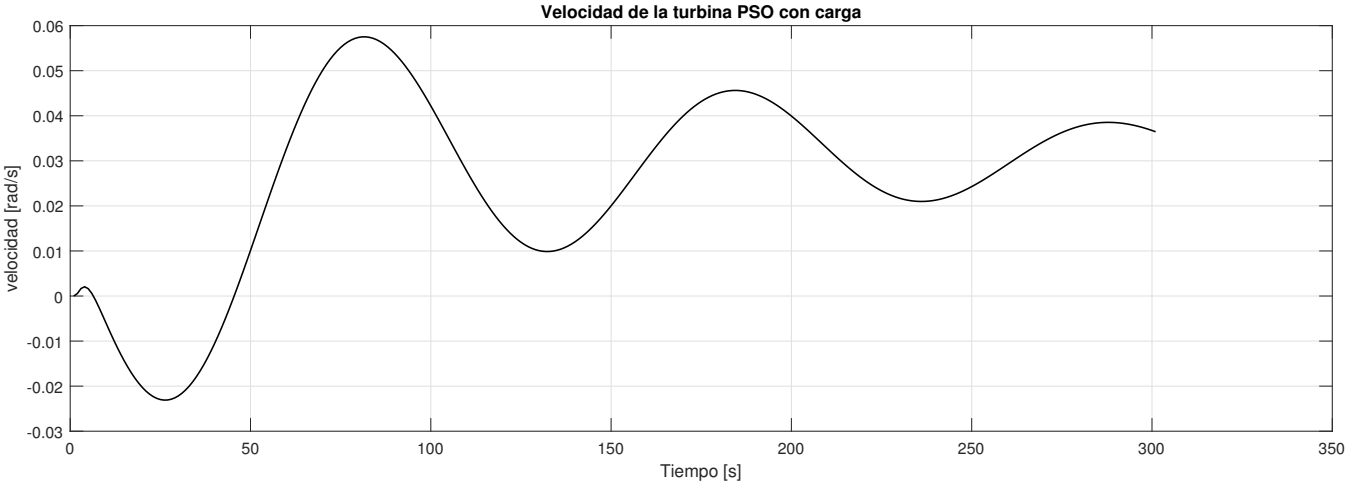

Figura 4.32: Salida velocidad de la turbina

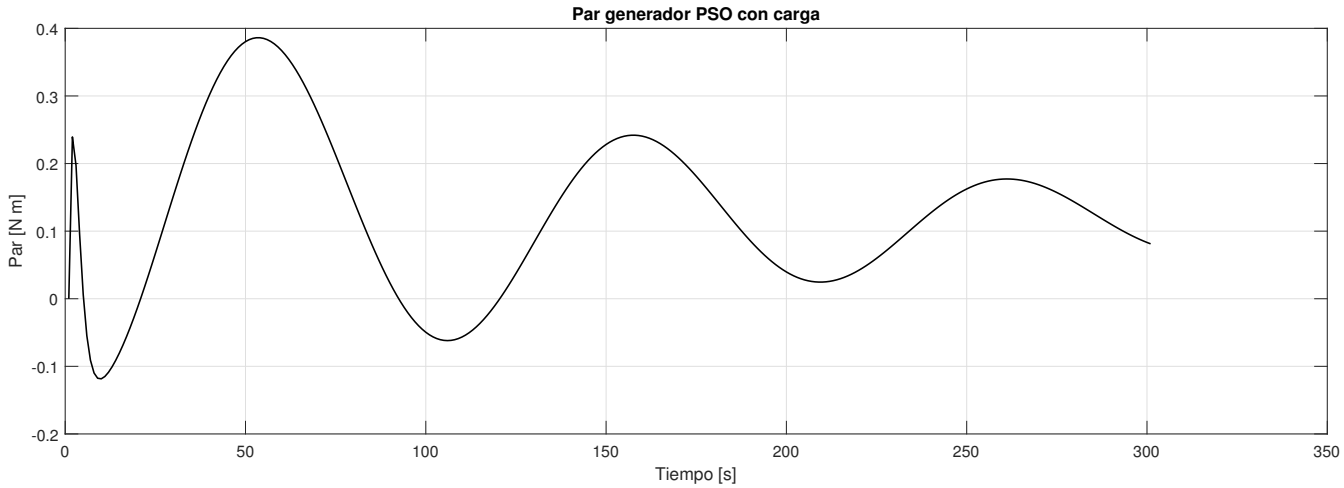

Figura 4.33: Salida Par generador

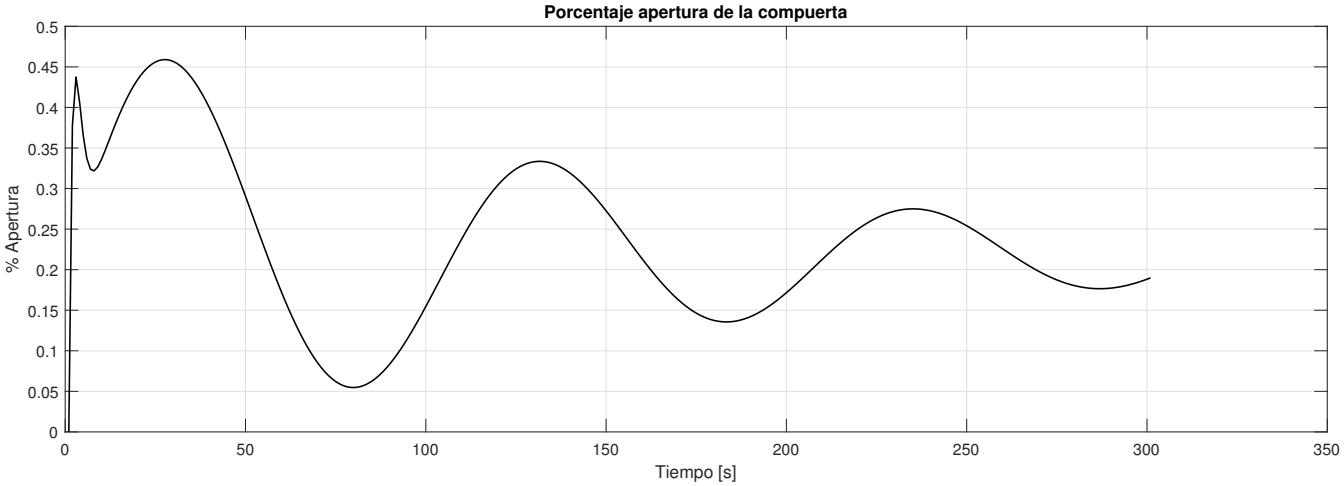

Figura 4.34: Salida porcentaje de apertura de la compuerta

Comportamiento del sistema SHTG con el algoritmo lobos grises (GWO) con carga, se muestra en la figura 4.35 salida de la velocidad de la turbina, en la figura 4.36 salida del par del generador y finalmente en la figura 4.37 el porcentaje de apertura de la compuerta, observándose como las respectivas salidas tienden a un valor estable una vez transcurrido el estado transitorio del sistema.

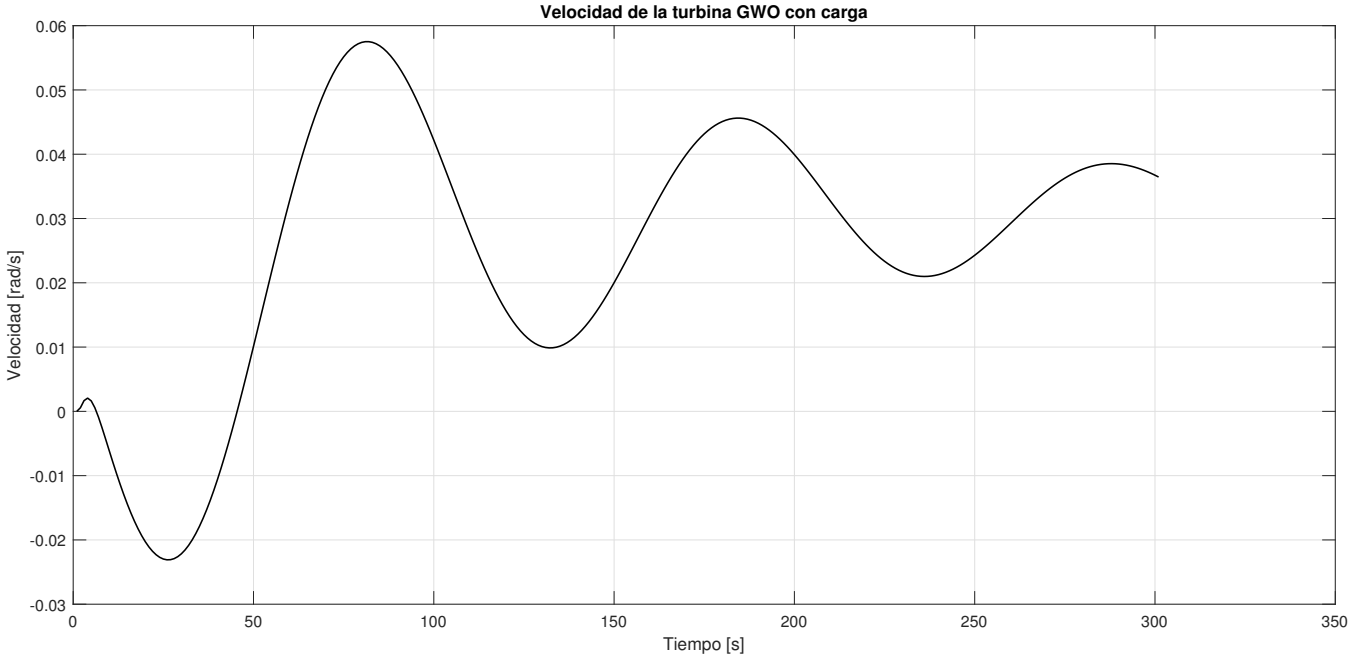

Figura 4.35: Salida velocidad de la turbina

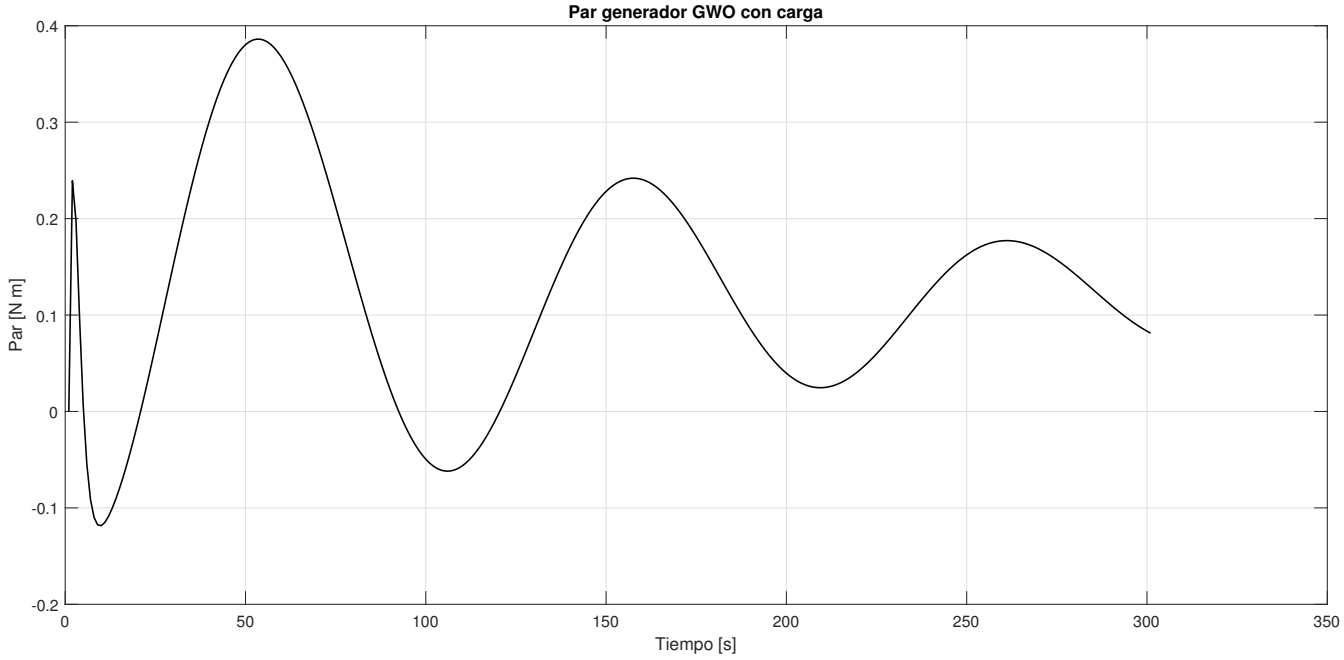

Figura 4.36: Salida Par generador

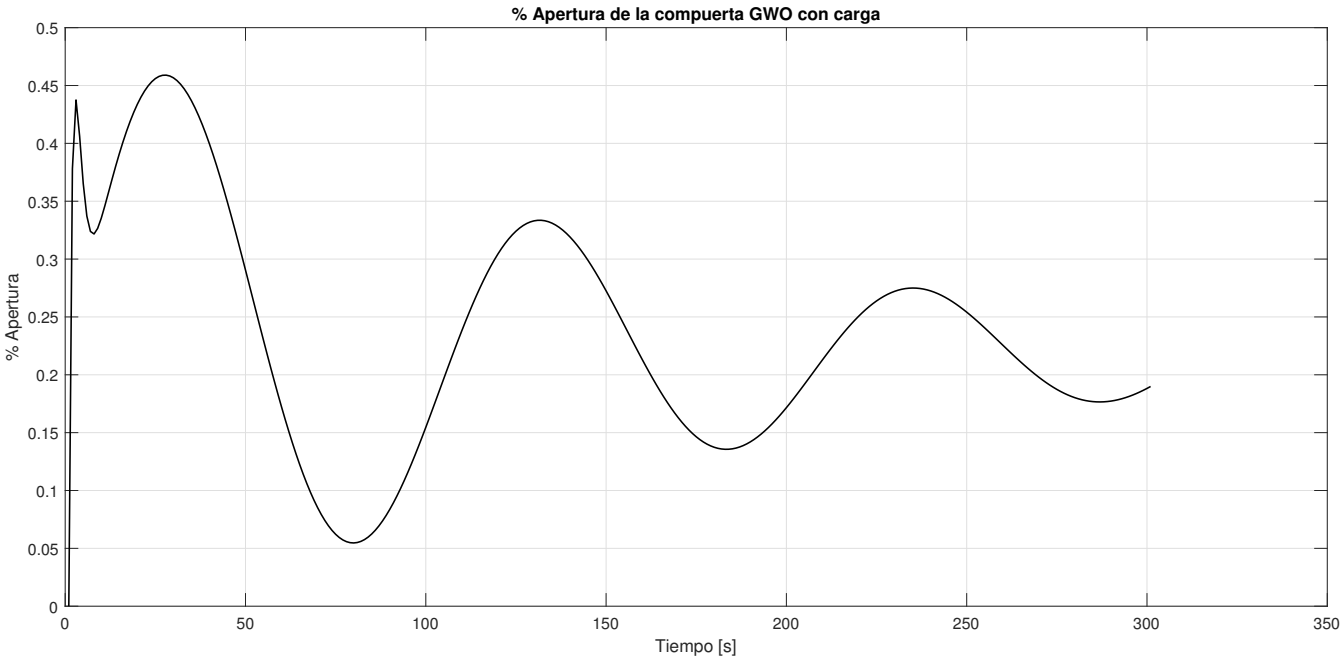

Figura 4.37: Salida porcentaje de apertura de la compuerta

Comportamiento del sistema SHTG con el algoritmo ballenas (WOA) con carga, se muestra en la figura 4.38 salida de la velocidad de la turbina, en la figura 4.39 salida del par del generador y finalmente en la figura 4.40 el porcentaje de apertura de la compuerta, observándose como las respectivas salidas tienden a un valor estable una vez transcurrido el estado transitorio del sistema.

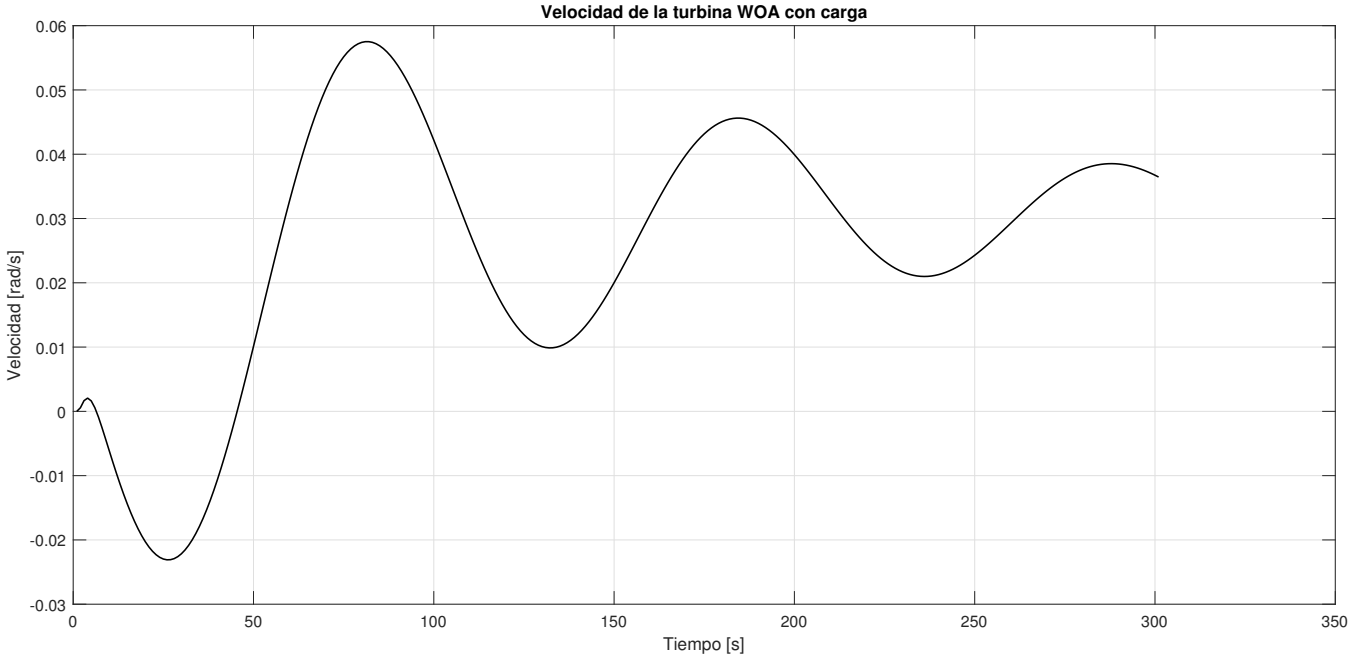

Figura 4.38: Salida velocidad de la turbina

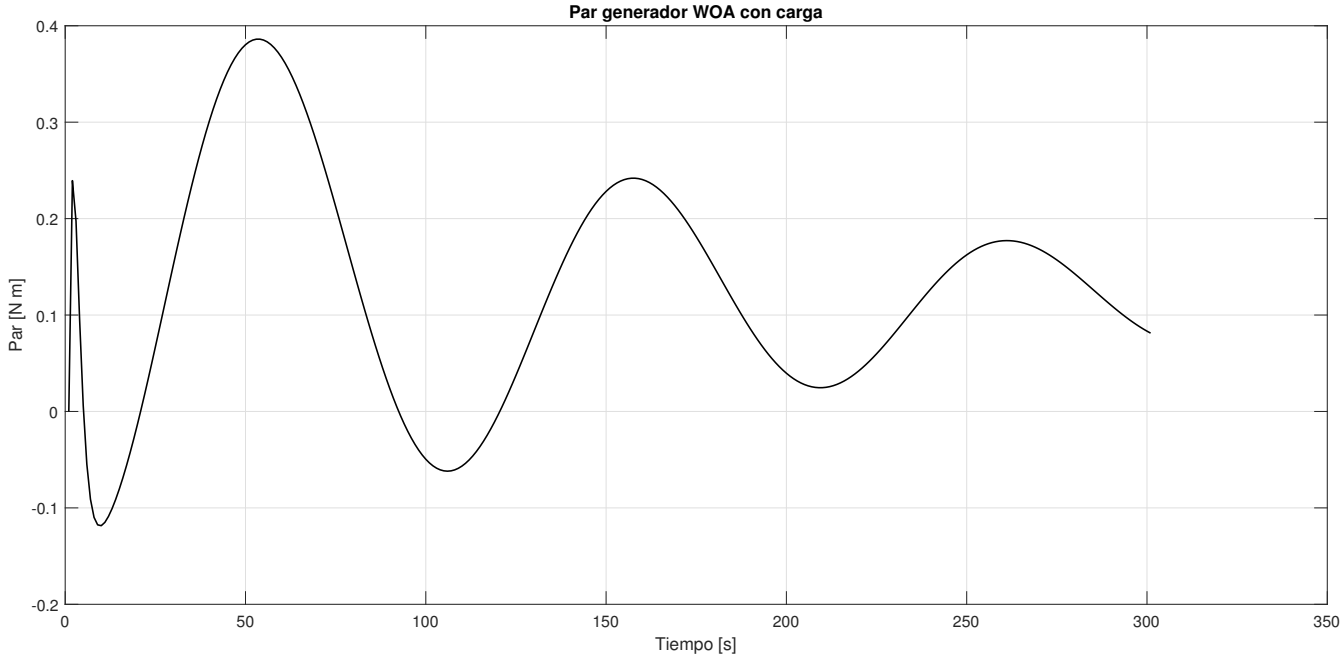

Figura 4.39: Salida Par generador

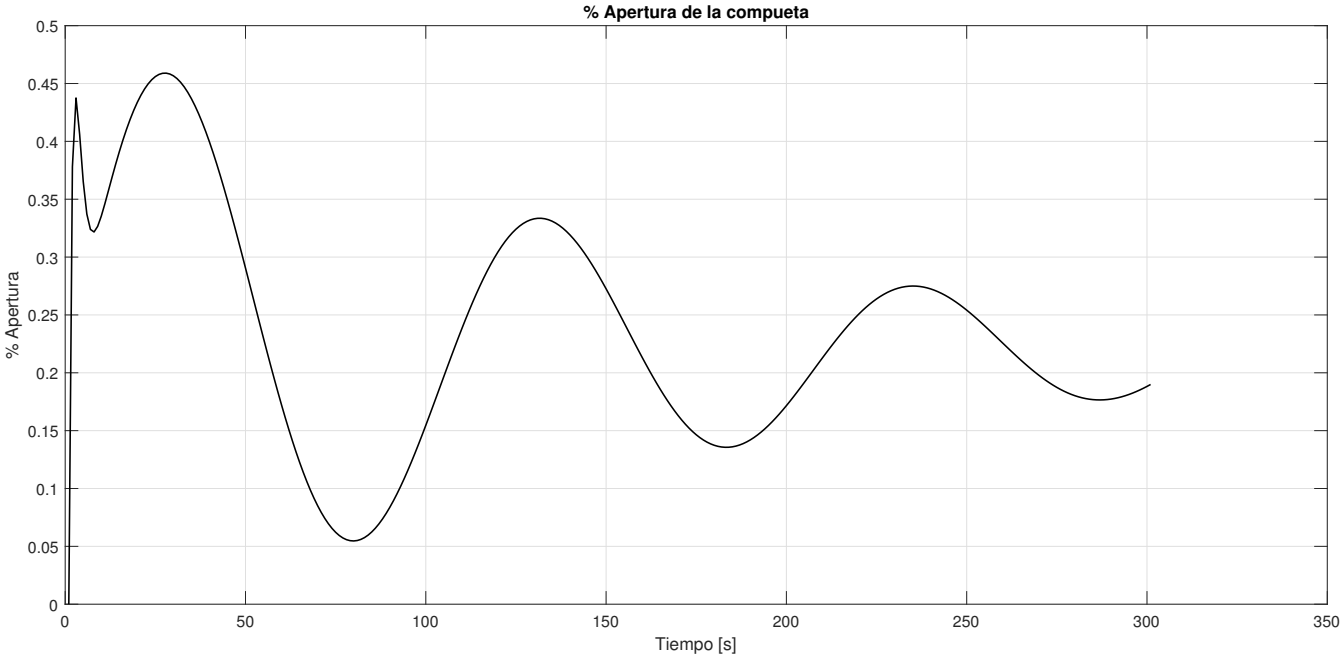

Figura 4.40: Salida porcentaje de apertura de la compuerta

5. Capítulo 5

5.1. Conclusiones

En el presente trabajo de investigación se realizó la identificación de los parámetros de un sistema de hidroturbina con generador (SHTG): constante de tiempo del sistema hidráulico (T_w), constante de tiempo de la compuerta (T_e), perdida de fricción (f), constante de tiempo del generador ($\acute{T}a$) y constante de ajuste del generador (e_g); mediante una simulación con Matlab y Simulink, ya que son de suma importancia para el buen funcionamiento de un sistema de generación de energía eléctrica con la aplicación de algoritmos evolutivos tales como el GA, PSO, GWO y WOA en condiciones de sin carga y con carga del SHTG.

Al término del trabajo podemos concluir que con los algoritmos evolutivos se cumple el objetivo establecido ya que fue posible evaluar el desempeño dinámico del sistema y obtener

valores estimados de los parámetros y al obtener el error con respecto a los valores reales se tiene un margen de error que tiende a cero, corroborando la hipótesis establecida de que con los algoritmos genéticos se obtienen soluciones factibles en la identificación de parámetros del SHTG en su comportamiento dinámico.

Como primer resultado se realizó una simulación del sistema con los valores de referencia: constante de tiempo del sistema hidráulico, constante de tiempo en la compuerta, pérdida de fricción, constante de tiempo de inercia del generador y finalmente la constante de ajuste del generador en una primera condición de sin carga del sistema y con carga en el SHTG, en las salidas velocidad del eje de giro de la turbina en rad/s, par del eje del generador en N-m y porcentaje de apertura de la válvula de la compuerta en cada salida observamos con base a la simulación que dichas salidas tienen un estado transitorio al pasar dicho estado tienden al establecimiento de modo que el sistema permanezca dentro de un set point especificado, es decir en estable.

Posterior a ello se realizó la identificación de los parámetros con los algoritmos evolutivos, obteniendo los siguientes resultados bajo condición sin carga y con carga.

Una vez realizadas las simulaciones observamos que hay poca variabilidad entre los diferentes algoritmos genéticos al realizar la identificación de los parámetros en comparación con los valores reales, sin embargo, el algoritmo que mejor comportamiento tiene es el PSO ya que es el que tiene menor desviación con respecto a los valores reales con un promedio del error $6.232034183224600e-11$ en condición sin carga y un valor de $1.874743155314987e-06$, con carga.
De los resultados se observa que los algoritmos evolutivos utilizados tienen a estabilizar las salidas de velocidad, par y porcentaje de apertura de válvula.

Con base a los resultados se cumple la hipótesis ya que con los algoritmos evolutivos se obtienen soluciones factibles en la identificación de parámetros de un sistema de hidroturbina

5.2. Aportaciones

Dentro de las principales aportaciones que se obtienen al desarrollar el trabajo de investigación podemos decir que los Algoritmos evolutivos se pueden utilizar en la identificación de parámetros de un sistema donde su función de transferencia es altamente no lineal.

Las identificaciones de parámetros en los sistemas mecánicos sufren desgaste y operan de con carga o sin carga ocasionando que los parámetros del SHTG cambien en función del desgaste y la carga a que son sometidos, esto indudablemente provoca que los parámetros del SHTG cambien en su comportamiento dinámico, de ahí la relevancia del trabajo de investigación.

Otra de las aportaciones es la implementación de los algoritmos evolutivos para evaluar su desempeño: GA, PSO, GWO y WOA en la mejora en el diseño dinámico del sistema.

En cuanto al uso de un modelo con Simulink utilizado como medio de simulación evitando utilizar modelos matemáticos que resultan complicados y altamente no lineales para la identificación de parámetros en este caso de un sistema en su comportamiento dinámico.

5.3. Trabajos futuros

Como trabajos futuros podemos implementar otros algoritmos evolutivos o bien hacer alguna hibridación para identificar los parámetros de un SHTG de modo que se mejore el comportamiento dinámico del sistema. Otro aspecto en la simulación es incrementar el número de repeticiones en la implementación de los algoritmos evolutivos.

6. Apéndice

Para la determinación de los resultados del sistema de hidroturbina con gobernador (SHTG) se realizó una simulación en Matlab y Simulink, en el cual se probo con los algoritmos: GA, PSO GWO y WOA, en una primera instancia se realiza la simulación sin carga del sistema y posteriormente con carga. Algoritmo SHTG y función consto para la simulación sin carga en Matlab, el cual es utilizado con Simulink para la obtención de resultados.

```
% Algoritmo HTGS

clear all
clc
Tfinal= 0:0.1:30;
paramNameValStruct.SimulationMode = 'normal';
paramNameValStruct.AbsTol = '1e-5';
paramNameValStruct.SaveState = 'on';
paramNameValStruct.StateSaveName = 'xout';
paramNameValStruct.SaveOutput = 'on';
paramNameValStruct.OutputSaveName = 'yout';
paramNameValStruct.SaveFormat = 'Dataset';

open_system('simulink_HTGS');
eg=0.4433;
Ta=12;
Taa=1/Ta;
Te=0.53;
Tw=1.5;
Twe=-Tw/Te;

set_param('simulink_HTGS/Gain6', 'Gain','eg');
```

```
set_param('simulink_HTGS/Gain9', 'Gain','Taa');
set_param('simulink_HTGS/Gain13', 'Gain','Twe');

% No load
ex  = -1.0567;
ey  =  0.9080;
eh  =  1.4191;
eqx =  -0.0574;
eqy =  0.7887;
eqh =  0.4571;

%Governor
Kp = 5.5912;
Ki = 1.061;
Kd = 3.2800;
T1v = 0.28;
bp = 0.04;

%Servo
Ty = 0.1;

%Hydraulic system
f = 0.01;
b2 = 2*f*Te^2/pi^2;
b1 = Te;
b0 = f;
a1 = 4*f*Te/(pi*sqrt(4+2*f^2));

num = [b2 b1 b0];
set_param('simulink_HTGS/Transfer', 'Numerator','num');
```

```
numerador = get_param('simulink_HTGS/Transfer','Numerator');



den = [1 a1 1];
set_param('simulink_HTGS/Transfer', 'Denominator','den');
denominador = get_param('simulink_HTGS/Transfer','Denominator');

SimOut = sim('simulink_HTGS',paramNameValStruct);
outputs = SimOut.get('yout');

x1 = outputs.getElement(1);
x2 = outputs.getElement(2);
x3 = outputs.getElement(3);

xr = x1.Values.Data
mtr = x2.Values.Data
yr = x3.Values.Data

plot(xr)
grid on
title("Velocidad de la turbina")
figure
plot(mtr)
title ("Par Generador")
grid on
figure
plot(yr)
title ("Apertura de la compuerta")
grid  on
```

```
Datos.xr=xr
Datos.mtr=mtr
Datos.yr=yr

save  ('Datos')
```

Función costo para evaluar cada uno de los algoritmos utilizados en la simulación.

```
function[ff]=funcioncosto(num,den)
global Datos;
Tfinal= 0:0.1:30;

paramNameValStruct.SimulationMode = 'normal';
paramNameValStruct.AbsTol = '1e-5';
paramNameValStruct.SaveState = 'on';
paramNameValStruct.StateSaveName = 'xout';
paramNameValStruct.SaveOutput = 'on';
paramNameValStruct.OutputSaveName = 'yout';
paramNameValStruct.SaveFormat = 'Dataset';

open_system('simulink_HTGS');

set_param('simulink_HTGS/Gain6', 'Gain','eg');
set_param('simulink_HTGS/Gain9', 'Gain','Taa');
set_param('simulink_HTGS/Gain13', 'Gain','Twe');

set_param('simulink_HTGS/Transfer', 'Numerator','num');
%numerador = get_param('simulink_HTGS/Transfer','Numerator');

set_param('simulink_HTGS/Transfer', 'Denominator','den');
```

```
%denominador = get_param('simulink_HTGS/Transfer','Denominator');

SimOut = sim('simulink_HTGS',paramNameValStruct);
outputs = SimOut.get('yout');

x1 = outputs.getElement(1);
x2 = outputs.getElement(2);
x3 = outputs.getElement(3);

x = x1.Values.Data;
mt = x2.Values.Data;
y = x3.Values.Data;

aux1=Datos.xr-x;
aux2=Datos.mtr-mt;
aux3=Datos.yr-y;

s1=sum(aux1.^2);
s2=sum(aux2.^2);
s3=sum(aux3.^2);
w1=s1/(s1+s2+s3);
w2=s2/(s2+s2+s3);
w3=s3/(s3+s2+s3);
ff= w1*s1+w2*s2+w3*s3;

end


% ALGORITMO GENETICO BINARIO
```

```
clear all
clc
global Datos;
Datos = load('Datos');
% N = Numero de individuos, L= Longitud del cromosoma.
N = 40; L = 18;
V = 5;          % Numero de variables independientes.
Epsilon = 1e-12; % Maximo error de la optimizacion
prob = 3;      % Porcentaje de mutacion
ITER_MAX =50;
Repeticiones = 1;
%Intervalos de busqueda
for z = 1:Repeticiones
I = [1.0 2.0;0.3  0.8; 0.001 0.02; 10 15; 0.2 0.6];

% No load
ex  = -1.0567;
ey  =  0.9080;
eh  =  1.4191;
eqx =  -0.0574;
eqy =  0.7887;
eqh =  0.4571;

%Governor
Kp = 5.5912;
Ki = 1.061;
Kd = 3.2800;
T1v = 0.28;
bp = 0.04;
```

```
%Servo
Ty = 0.1;

%Generemos la poblacion de individuos
Pob_2 = Poblacion(N,L,V);
%Poblacion en base decimal
Pob_10= PobDec(Pob_2,L,V);
%Poblaciones en R
Pob_real = Escalamiento(Pob_10,I,V,L);
cont = 1;
ff_aux = 1000;
while ((cont <= ITER_MAX) && (ff_aux > Epsilon))
%Determinamos la funcion costo
for k = 1: N
Tw = Pob_real(k,1);
Te = Pob_real(k,2);
f = Pob_real(k,3);
Ta = Pob_real(k,4);
eg = Pob_real(k,5);
Taa=1/Ta;
Twe=-Tw/Te;
b2 = 2*f*Te^2/pi^2;
b1 = Te;
b0 = f;
a1 = 4*f*Te/(pi*sqrt(4+2*f^2));
num = [b2 b1 b0];
den = [1 a1 1];
ff(k) = funcioncosto(num,den);
end
```

```
error(cont)=min(ff);
%Seleccionamos la poblacion
PobSel_2 = Seleccion(Pob_2,ff);
%Cruzamos la Pob seleccionada
Pob_C = Cruce(PobSel_2,L,V);
%Concatenamos PobSel + Pob_C
Pob_2 = [PobSel_2;Pob_C];
%Mutamos la nueva poblacion
Pob_2 = Mutacion(Pob_2,L,prob);
%Poblacion en base decimal
Pob_10= PobDec(Pob_2,L,V);
%Poblacion en R
Pob_real = Escalamiento(Pob_10,I,V,L);
[ff_aux, idy] = min(ff);
cont = cont+1
end

%Determinamos la funcion costo
for k = 1: N
Tw = Pob_real(k,1);
Te = Pob_real(k,2);
f = Pob_real(k,3);
Ta = Pob_real(k,4);
eg = Pob_real(k,5);
Taa=1/Ta;
Twe=-Tw/Te;
b2 = 2*f*Te^2/pi^2;
b1 = Te;
b0 = f;
a1 = 4*f*Te/(pi*sqrt(4+2*f^2));
```

```
num = [b2 b1 b0];
den = [1 a1 1];
Curves(z,k) = funcioncosto(num,den);
end
[ff_aux, idy] = min(ff);

Best_score(z) = ff_aux;
Best_position(z,:) = Pob_real(idy,:)

end

Res.ag_best_pos = Best_position;
Res.ag_best_score = Best_score;
Res.ag_curves = Curves;
Res.ag_individuos = N;
Res.ag_iteraciones = ITER_MAX ;
Res.ag_repeticiones = Repeticiones;
%
save Res;

% ALGORITMO PSO

clear all
clc
Repeticiones = 20;
global Datos;
Datos = load('Datos');
% N = Numero de individuos
N = 40;
D = 5;       % Numero de variables independientes.
```

```
ITER_MAX = 50;
Evo = ITER_MAX;
W = 0.95-((0.9:Evo).*(0.85 ./ Evo));%Peso inercial, para hacer exploracion.
C1 = 2.0;      %Ganancia de factor cognitivo, memoria de mejor posicion
C2 = 2.0;      %Ganancia de factor social, influencia del lider
lb = [1.0  0.3  0.001 10 0.2];
ub = [2   0.8   0.015  15 0.6];

minx(1:D) =lb;
maxx(1:D) =ub;

MinX = repmat(minx,N,1);%se genera de manera matricial
MaxX = repmat(maxx,N,1);
Vmax=(MaxX-MinX)*0.1;
Vmin= -Vmax;

for j = 1: Repeticiones

% No load
ex  = -1.0567;
ey  =  0.9080;
eh  =  1.4191;
eqx =  -0.0574;
eqy =  0.7887;
eqh =  0.4571

%Governor
Kp = 5.5912;
Ki = 1.061;
Kd = 3.2800;
```

```
T1v = 0.28;
bp = 0.04;

%Servo
Ty = 0.1;

%Inicializar vector posicion
X = rand(N,D).*(MaxX-MinX)+MinX;
%%dX = rand(N,D).*(MaxX-MinX)+MinX;
dX=zeros(N,D);

%Calcular la funcion costo
for k = 1: N
Tw = X(k,1);
Te = X(k,2);
f = X(k,3);
Ta = X(k,4);
eg = X(k,5);
Taa=1/Ta;
Twe=-Tw/Te;
b2 = 2*f*Te^2/pi^2;
b1 = Te;
b0 = f;
a1 = 4*f*Te/(pi*sqrt(4+2*f^2));
num = [b2 b1 b0];
den = [1 a1 1];
ff(k) = funcioncosto(num,den);
end
%Al inicio Pbest = X,la mejor posicion, la primera corrida es la posicion inicial
Pbest = X;
```

```
%Al inicio Pbestval = ff
Pbestval = ff;

%Calcular el mejor global
[gbestval, idx] = min(Pbestval);%se determina el lider
Gbest = repmat(X(idx,:),N,1);% se replica para formar una matriz

for i = 1:ITER_MAX
%Determinamos la aceleracion
dX = W(i).*dX+C1*rand(N,D).*(Pbest-X) + C2*rand(N,D).*(Gbest-X);
%Cuidar el intervalo de dX
%Limite superior
dX=min(dX, Vmax);
dX=max(dX, Vmin);
%Actualizar el vector posicion
X = X+dX;
%Verificar que X se encuentre en el intervalo
X=min(X, MaxX);
X=max(X, MinX);
%Determinamos la funcion costo
for k = 1: N
Tw = X(k,1);
Te = X(k,2);
f = X(k,3);
Ta = X(k,4);
eg = X(k,5);
Taa=1/Ta;
Twe=-Tw/Te;
b2 = 2*f*Te^2/pi^2;
```

```
b1 = Te;
b0 = f;
a1 = 4*f*Te/(pi*sqrt(4+2*f^2));
num = [b2 b1 b0];
den = [1 a1 1];
ff(k) = funcioncosto(num,den);
end

%Si se mueve a una mejor posicion entonces es considerada

tmp = Pbestval < ff;
tmp1 = repmat(tmp',1,D);
Pbest = Pbest.*tmp1+(1-tmp1).*X;

%Se actualizan los mejores desempenos
Pbestval = tmp.*Pbestval + (1-tmp).*ff;

%Se determina el nuevo lider
[gbestval, idx] = min(Pbestval);
Gbest = repmat(X(idx,:),N,1);

%error(cont)=min(ff);
[ff_aux, idy] = min(ff);
%cont = cont+1;
Curves(j,i)=min(Pbestval);
i
end
[val idx]      = min(Pbestval);
Best_pos(j,:)  = Pbest(idx,:);
Best_score(j)  = val;
```

```
j

end

Res.pso_best_pos = Best_pos;
Res.pso_best_score = Best_score;
Res.pso_curves = Curves;
Res.pso_individuos = N;
Res.pso_iteraciones = ITER_MAX;
Res.pso_repeticiones = Repeticiones;

save Res;



% ALGORITMO GWO, tomado de https://seyedalimirjalili.com/gwo

clear all
clc
Repeticiones = 20;
global Datos;
Datos = load('Datos');

% N = Numero de individuos
N = 40;
SearchAgents_no = N;
D = 5;          % Numero de variables independientes.
dim = D;
Max_iter = 50;
Evo = Max_iter;
```

```
%Sin Carga
ex  = -1.0567;
ey  = 0.9080;
eh  = 1.4191;
eqx = -0.0574;
eqy = 0.7887;
eqh = 0.4571;

%Governor
Kp = 5.5912;
Ki = 1.061;
Kd = 3.2800;
T1v = 0.28;
bp = 0.04;

%Servo
Ty = 0.1;

for k = 1: Repeticiones
% initialize alpha, beta, and delta_pos
Alpha_pos=zeros(1,dim);
Alpha_score=inf; %change this to -inf for maximization problems

Beta_pos=zeros(1,dim);
Beta_score=inf; %change this to -inf for maximization problems

Delta_pos=zeros(1,dim);
Delta_score=inf; %change this to -inf for maximization problems

lb = [1.0  0.3  0.001 10 0.2];
```

```
ub = [2   0.8   0.02  15 0.6];
%Initialize the positions of search agents
Positions=generar_poblacion(SearchAgents_no,lb,ub,dim);

Convergence_curve=zeros(1,Max_iter);

l=0;% Loop counter

% Main loo
while l<Max_iter
for i=1:size(Positions,1)
% Return back the search agents that go beyond the boundaries of the search space
Flag4ub=Positions(i,:)>ub;
Flag4lb=Positions(i,:)<lb;
Positions(i,:)=(Positions(i,:).*(~(Flag4ub+Flag4lb)))+ub.*Flag4ub+lb.*Flag4lb;
% Calculate objective function for each search agent
%fitness=fobj(Positions(i,:));
Tw = Positions(i,1);
Te = Positions(i,2);
f = Positions(i,3);
Ta = Positions(i,4);
eg = Positions(i,5);
Taa=1/Ta;
Twe=-Tw/Te;
b2 = 2*f*Te^2/pi^2;
b1 = Te;
b0 = f;
a1 = 4*f*Te/(pi*sqrt(4+2*f^2));
num = [b2 b1 b0];
den = [1 a1 1];
```

```
fitness = funcioncosto(num,den);

% Update Alpha, Beta, and Delta
if fitness<Alpha_score
Alpha_score=fitness; % Update alpha
Alpha_pos=Positions(i,:);
end

if fitness>Alpha_score && fitness<Beta_score
Beta_score=fitness; % Update beta
Beta_pos=Positions(i,:);
end

if fitness>Alpha_score && fitness>Beta_score && fitness<Delta_score
Delta_score=fitness; % Update delta
Delta_pos=Positions(i,:);
end
end
a=2-l*((2)/Max_iter); % a decreases linearly fron 2 to 0
% Update the Position of search agents including omegas
for i=1:size(Positions,1)
for j=1:size(Positions,2)

r1=rand(); % r1 is a random number in [0,1]
r2=rand(); % r2 is a random number in [0,1]

A1=2*a*r1-a;
C1=2*r2;

D_alpha=abs(C1*Alpha_pos(j)-Positions(i,j));
```

```
X1=Alpha_pos(j)-A1*D_alpha;

r1=rand();
r2=rand();

A2=2*a*r1-a;
C2=2*r2;

D_beta=abs(C2*Beta_pos(j)-Positions(i,j));
X2=Beta_pos(j)-A2*D_beta;

r1=rand();
r2=rand();

A3=2*a*r1-a;
C3=2*r2;

D_delta=abs(C3*Delta_pos(j)-Positions(i,j));
X3=Delta_pos(j)-A3*D_delta;
Positions(i,j)=(X1+X2+X3)/3;
end
end

l=l+1;
Curves(k, l)=Alpha_score;
end
Best_pos(k,:)  = Alpha_pos;
Best_score(k)  = Alpha_score;
k
end
```

```
Res.gwo_best_pos = Best_pos;
Res.gwo_best_score = Best_score;
Res.gwo_curves = Curves;
Res.gwo_individuos = N;
Res.gwo_iteraciones = Max_iter ;
Res.gwo_repeticiones = Repeticiones;

save Res;


% Algoritmo WOA, tomado de https://seyedalimirjalili.com/woa

global Datos;
Datos = load('Datos');

SearchAgents_no = 40;
Max_iter = 50;
lb = [1.0  0.3  0.001 10 0.2];
ub = [2   0.8   0.02  15 0.6];
dim = 5
Repetition = 20;

%Sin Carga
ex  = -1.0567;
ey  = 0.9080;
eh  = 1.4191;
eqx = -0.0574;
eqy = 0.7887;
eqh = 0.4571;
```

```
%Governor
Kp = 5.5912;
Ki = 1.061;
Kd = 3.2800;
T1v = 0.28;
bp = 0.04;

%Servo
Ty = 0.1;


for cont=1:Repetition
% initialize position vector and score for the leader
Leader_pos=zeros(1,dim);
Leader_score=inf; %change this to -inf for maximization problems

%Initialize the positions of search agents
Positions=generar_poblacion(SearchAgents_no,lb,ub,dim);

Convergence_curve=zeros(1,Max_iter);

t=0;% Loop counter

% Main loop
while t<Max_iter
for i=1:size(Positions,1)
% Return back the search agents that go beyond the boundaries of the search space
Flag4ub=Positions(i,:)>ub;
Flag4lb=Positions(i,:)<lb;
```

```
Positions(i,:)=(Positions(i,:).*(~(Flag4ub+Flag4lb)))+ub.*Flag4ub+lb.*Flag4lb;

% Calculate objective function for each search agent
% fitness=fobj(Positions(i,:));

% Calculate objective function for each search agent
%fitness=fobj(Positions(i,:));
Tw = Positions(i,1);
Te = Positions(i,2);
f = Positions(i,3);
Ta = Positions(i,4);
eg = Positions(i,5);
Taa=1/Ta;
Twe=-Tw/Te;
b2 = 2*f*Te^2/pi^2;
b1 = Te;
b0 = f;
a1 = 4*f*Te/(pi*sqrt(4+2*f^2));
num = [b2 b1 b0];
den = [1 a1 1];
fitness = funcioncosto(num,den);

% Update the leader
if fitness<Leader_score % Change this to > for maximization problem
Leader_score=fitness; % Update alpha
Leader_pos=Positions(i,:);
end

end
```

```
a=2-t*((2)/Max_iter);
a2=-1+t*((-1)/Max_iter);

% Update the Position of search agents
for i=1:size(Positions,1)
r1=rand(); % r1 is a random number in [0,1]
r2=rand(); % r2 is a random number in [0,1]

A=2*a*r1-a;
C=2*r2;
b=1;
l=(a2-1)*rand+1;
p = rand();

for j=1:size(Positions,2)
if p<0.5
if abs(A)>=1
rand_leader_index = floor(SearchAgents_no*rand()+1);
X_rand = Positions(rand_leader_index, :);
D_X_rand=abs(C*X_rand(j)-Positions(i,j));
Positions(i,j)=X_rand(j)-A*D_X_rand;

elseif abs(A)<1
D_Leader=abs(C*Leader_pos(j)-Positions(i,j));
Positions(i,j)=Leader_pos(j)-A*D_Leader;
end

elseif p>=0.5

distance2Leader=abs(Leader_pos(j)-Positions(i,j));
```

```
Positions(i,j)=distance2Leader*exp(b.*l).*cos(l.*2*pi)+Leader_pos(j);
end
end
end
t=t+1
Convergence_curve(t)=Leader_score;
end
Best_score(cont) = Leader_score;
Best_pos(cont,:)= Leader_pos;
Curves(cont,:) = Convergence_curve;
cont
end

Res.gwo_best_pos = Best_pos;
Res.gwo_best_score = Best_score;
Res.gwo_curves = Curves;
Res.gwo_individuos = N;
Res.gwo_iteraciones = Max_iter ;
Res.gwo_repeticiones = Repeticiones;

save Res;
```

Algoritmo SHTG y función consto para la simulación con carga en Matlab, el cual es utilizado con Simulink para la obtención de resultados.

```
% Algoritmo HTGS

clear all
clc
Tfinal= 0:0.1:30;
```

```
paramNameValStruct.SimulationMode = 'normal';
paramNameValStruct.AbsTol = '1e-5';
paramNameValStruct.SaveState = 'on';
paramNameValStruct.StateSaveName = 'xout';
paramNameValStruct.SaveOutput = 'on';
paramNameValStruct.OutputSaveName = 'yout';
paramNameValStruct.SaveFormat = 'Dataset';

open_system('simulink_HTGS');
eg=0.4433;
Ta=12;
Taa=1/Ta;
Te=0.53;
Tw=1.5;
Twe=-Tw/Te;

set_param('simulink_HTGS/Gain6', 'Gain','eg');
set_param('simulink_HTGS/Gain9', 'Gain','Taa');
set_param('simulink_HTGS/Gain13', 'Gain','Twe');

% Load
ex  = -1.4673;
ey  = 0.7713;
eh  = 1.7179;
eqx = -0.4901;
eqy = 0.8184;
eqh = 0.7257;


%Governor
```

```
Kp = 5.5912;
Ki = 1.061;
Kd = 3.2800;
T1v = 0.28;
bp = 0.04;

%Servo
Ty = 0.1;

%Hydraulic system
f = 0.01;
b2 = 2*f*Te^2/pi^2;
b1 = Te;
b0 = f;
a1 = 4*f*Te/(pi*sqrt(4+2*f^2));

num = [b2 b1 b0];
set_param('simulink_HTGS/Transfer', 'Numerator','num');
numerador = get_param('simulink_HTGS/Transfer','Numerator');


den = [1 a1 1];
set_param('simulink_HTGS/Transfer', 'Denominator','den');
denominador = get_param('simulink_HTGS/Transfer','Denominator');

SimOut = sim('simulink_HTGS',paramNameValStruct);
outputs = SimOut.get('yout');

x1 = outputs.getElement(1);
x2 = outputs.getElement(2);
```

```
x3 = outputs.getElement(3);


xr = x1.Values.Data
mtr = x2.Values.Data
yr = x3.Values.Data


plot(xr)
grid on
title("Velocidad de la turbina")
figure
plot(mtr)
title ("Par Generador")
grid on
figure
plot(yr)
title ("Apertura de la compuerta")
grid  on


Datos.xr=xr
Datos.mtr=mtr
Datos.yr=yr
```

Función costo para evaluar cada uno de los algoritmos utilizados en la simulación:

```
function[ff]=funcioncosto(num,den)
global Datos;
Tfinal= 0:0.1:30;


paramNameValStruct.SimulationMode = 'normal';
```

```
paramNameValStruct.AbsTol = '1e-5';
paramNameValStruct.SaveState = 'on';
paramNameValStruct.StateSaveName = 'xout';
paramNameValStruct.SaveOutput = 'on';
paramNameValStruct.OutputSaveName = 'yout';
paramNameValStruct.SaveFormat = 'Dataset';

open_system('simulink_HTGS');

set_param('simulink_HTGS/Gain6', 'Gain','eg');
set_param('simulink_HTGS/Gain9', 'Gain','Taa');
set_param('simulink_HTGS/Gain13', 'Gain','Twe');

set_param('simulink_HTGS/Transfer', 'Numerator','num');
%numerador = get_param('simulink_HTGS/Transfer','Numerator');

set_param('simulink_HTGS/Transfer', 'Denominator','den');
%denominador = get_param('simulink_HTGS/Transfer','Denominator');

SimOut = sim('simulink_HTGS',paramNameValStruct);
outputs = SimOut.get('yout');

x1 = outputs.getElement(1);
x2 = outputs.getElement(2);
x3 = outputs.getElement(3);

x = x1.Values.Data;
mt = x2.Values.Data;
y = x3.Values.Data;
```

```
aux1=Datos.xr-x;
aux2=Datos.mtr-mt;
aux3=Datos.yr-y;

s1=sum(aux1.^2);
s2=sum(aux2.^2);
s3=sum(aux3.^2);
w1=s1/(s1+s2+s3);
w2=s2/(s2+s2+s3);
w3=s3/(s3+s2+s3);
ff= w1*s1+w2*s2+w3*s3;

end


% ALGORITMO GENETICO BINARIO

clear all
clc
global Datos;
Datos = load('Datos');
% N = Numero de individuos, L= Longitud del cromosoma.
N = 40; L = 18;
V = 5;          % Numero de variables independientes.
Epsilon = 1e-12; % Maximo error de la optimizacion
prob = 3;     % Porcentaje de mutacion
ITER_MAX =50;
Repeticiones = 1;
%Intervalos de busqueda
for z = 1:Repeticiones
```

```
I = [1.0 2.0;0.3  0.8; 0.001 0.02; 10 15; 0.2 0.6];

% Load
ex  = -1.4673;
ey  = 0.7713;
eh  = 1.7179;
eqx = -0.4901;
eqy = 0.8184;
eqh = 0.7257;


%Governor
Kp = 5.5912;
Ki = 1.061;
Kd = 3.2800;
T1v = 0.28;
bp = 0.04;

%Servo
Ty = 0.1;

%Generemos la poblacion de individuos
Pob_2 = Poblacion(N,L,V);
%Poblacion en base decimal
Pob_10= PobDec(Pob_2,L,V);
%Poblaciones en R
Pob_real = Escalamiento(Pob_10,I,V,L);
cont = 1;
ff_aux = 1000;
while ((cont <= ITER_MAX) && (ff_aux > Epsilon))
```

```
%Determinamos la funcion costo
for k = 1: N
Tw = Pob_real(k,1);
Te = Pob_real(k,2);
f = Pob_real(k,3);
Ta = Pob_real(k,4);
eg = Pob_real(k,5);
Taa=1/Ta;
Twe=-Tw/Te;
b2 = 2*f*Te^2/pi^2;
b1 = Te;
b0 = f;
a1 = 4*f*Te/(pi*sqrt(4+2*f^2));
num = [b2 b1 b0];
den = [1 a1 1];
ff(k) = funcioncosto(num,den);
end

error(cont)=min(ff);
%Seleccionamos la poblacion
PobSel_2 = Seleccion(Pob_2,ff);
%Cruzamos la Pob seleccionada
Pob_C = Cruce(PobSel_2,L,V);
%Concatenamos PobSel + Pob_C
Pob_2 = [PobSel_2;Pob_C];
%Mutamos la nueva poblacion
Pob_2 = Mutacion(Pob_2,L,prob);
%Poblacion en base decimal
Pob_10= PobDec(Pob_2,L,V);
%Poblacion en R
```

```
Pob_real = Escalamiento(Pob_10,I,V,L);
[ff_aux, idy] = min(ff);
cont = cont+1
end

%Determinamos la funcion costo
for k = 1: N
Tw = Pob_real(k,1);
Te = Pob_real(k,2);
f = Pob_real(k,3);
Ta = Pob_real(k,4);
eg = Pob_real(k,5);
Taa=1/Ta;
Twe=-Tw/Te;
b2 = 2*f*Te^2/pi^2;
b1 = Te;
b0 = f;
a1 = 4*f*Te/(pi*sqrt(4+2*f^2));
num = [b2 b1 b0];
den = [1 a1 1];
Curves(z,k) = funcioncosto(num,den);
end
[ff_aux, idy] = min(ff);

Best_score(z) = ff_aux;
Best_position(z,:) = Pob_real(idy,:)

end

Res.ag_best_pos = Best_position;
```

```
Res.ag_best_score = Best_score;
Res.ag_curves = Curves;
Res.ag_individuos = N;
Res.ag_iteraciones = ITER_MAX ;
Res.ag_repeticiones = Repeticiones;
%
save Res;
% ALGORITMO PSO

clear all
clc
Repeticiones = 20;
global Datos;
Datos = load('Datos');
% N = Numero de individuos
N = 40;
D = 5;       % Numero de variables independientes.
ITER_MAX = 50;
Evo = ITER_MAX;
W = 0.95-((0.9:Evo).*(0.85 ./ Evo));%Peso inercial, para hacer exploracion.
C1 = 2.0;      %Ganancia de factor cognitivo, memoria de mejor posicion
C2 = 2.0;      %Ganancia de factor social, influencia del lider
lb = [1.0  0.3  0.001 10 0.2];
ub = [2   0.8   0.015  15 0.6];

minx(1:D) =lb;
maxx(1:D) =ub;

MinX = repmat(minx,N,1);%se genera de manera matricial
MaxX = repmat(maxx,N,1);
```

```
Vmax=(MaxX-MinX)*0.1;
Vmin= -Vmax;

for j = 1: Repeticiones

% Load
ex  = -1.4673;
ey  = 0.7713;
eh  = 1.7179;
eqx = -0.4901;
eqy = 0.8184;
eqh = 0.7257;



%Governor
Kp = 5.5912;
Ki = 1.061;
Kd = 3.2800;
T1v = 0.28;
bp = 0.04;

%Servo
Ty = 0.1;

%Inicializar vector posicion
X = rand(N,D).*(MaxX-MinX)+MinX;
%%dX = rand(N,D).*(MaxX-MinX)+MinX;
dX=zeros(N,D);

%Calcular la funcion costo
```

```
for k = 1: N
Tw = X(k,1);
Te = X(k,2);
f = X(k,3);
Ta = X(k,4);
eg = X(k,5);
Taa=1/Ta;
Twe=-Tw/Te;
b2 = 2*f*Te^2/pi^2;
b1 = Te;
b0 = f;
a1 = 4*f*Te/(pi*sqrt(4+2*f^2));
num = [b2 b1 b0];
den = [1 a1 1];
ff(k) = funcioncosto(num,den);
end
%Al inicio Pbest = X,la mejor posicion, la primera corrida es la posicion inicial
Pbest = X;

%Al inicio Pbestval = ff
Pbestval = ff;

%Calcular el mejor global
[gbestval, idx] = min(Pbestval);%se determina el lider
Gbest = repmat(X(idx,:),N,1);% se replica para formar una matriz

for i = 1:ITER_MAX
%Determinamos la aceleracion
dX = W(i).*dX+C1*rand(N,D).*(Pbest-X) + C2*rand(N,D).*(Gbest-X);
%Cuidar el intervalo de dX
```

```
%Limite superior
dX=min(dX, Vmax);
dX=max(dX, Vmin);
%Actualizar el vector posicion
X = X+dX;
%Verificar que X se encuentre en el intervalo
X=min(X, MaxX);
X=max(X, MinX);
%Determinamos la funcion costo
for k = 1: N
Tw = X(k,1);
Te = X(k,2);
f = X(k,3);
Ta = X(k,4);
eg = X(k,5);
Taa=1/Ta;
Twe=-Tw/Te;
b2 = 2*f*Te^2/pi^2;
b1 = Te;
b0 = f;
a1 = 4*f*Te/(pi*sqrt(4+2*f^2));
num = [b2 b1 b0];
den = [1 a1 1];
ff(k) = funcioncosto(num,den);
end

%Si se mueve a una mejor posicion entonces es considerada
tmp = Pbestval < ff;
tmp1 = repmat(tmp',1,D);
Pbest = Pbest.*tmp1+(1-tmp1).*X;
```

```
%Se actualizan los mejores desempenos
Pbestval = tmp.*Pbestval + (1-tmp).*ff;

%Se determina el nuevo lider
[gbestval, idx] = min(Pbestval);
Gbest = repmat(X(idx,:),N,1);

%error(cont)=min(ff);
[ff_aux, idy] = min(ff);
%cont = cont+1;
Curves(j,i)=min(Pbestval);
i
end
[val idx]      = min(Pbestval);
Best_pos(j,:)  = Pbest(idx,:);
Best_score(j)  = val;
j

end

Res.pso_best_pos = Best_pos;
Res.pso_best_score = Best_score;
Res.pso_curves = Curves;
Res.pso_individuos = N;
Res.pso_iteraciones = ITER_MAX;
Res.pso_repeticiones = Repeticiones;

save Res;
```

```
% ALGORITMO GWO tomado de https://seyedalimirjalili.com/gwo

clear all
clc
Repeticiones = 20;
global Datos;
Datos = load('Datos');

% N = Numero de individuos
N = 40;
SearchAgents_no = N;
D = 5;          % Numero de variables independientes.
dim = D;
Max_iter = 50;
Evo = Max_iter;



% Load
ex  = -1.4673;
ey  = 0.7713;
eh  = 1.7179;
eqx = -0.4901;
eqy = 0.8184;
eqh = 0.7257;

%Governor
Kp = 5.5912;
Ki = 1.061;
```

```
Kd = 3.2800;
T1v = 0.28;
bp = 0.04;

%Servo
Ty = 0.1;

for k = 1: Repeticiones
% initialize alpha, beta, and delta_pos
Alpha_pos=zeros(1,dim);
Alpha_score=inf; %change this to -inf for maximization problems

Beta_pos=zeros(1,dim);
Beta_score=inf; %change this to -inf for maximization problems

Delta_pos=zeros(1,dim);
Delta_score=inf; %change this to -inf for maximization problems

lb = [1.0  0.3  0.001 10 0.2];
ub = [2   0.8   0.02  15 0.6];
%Initialize the positions of search agents
Positions=generar_poblacion(SearchAgents_no,lb,ub,dim);

Convergence_curve=zeros(1,Max_iter);

l=0;% Loop counter

% Main loo
while l<Max_iter
for i=1:size(Positions,1)
```

```
% Return back the search agents that go beyond the boundaries of the search space
Flag4ub=Positions(i,:)>ub;
Flag4lb=Positions(i,:)<lb;
Positions(i,:)=(Positions(i,:).*(~(Flag4ub+Flag4lb)))+ub.*Flag4ub+lb.*Flag4lb;
% Calculate objective function for each search agent
%fitness=fobj(Positions(i,:));
Tw = Positions(i,1);
Te = Positions(i,2);
f = Positions(i,3);
Ta = Positions(i,4);
eg = Positions(i,5);
Taa=1/Ta;
Twe=-Tw/Te;
b2 = 2*f*Te^2/pi^2;
b1 = Te;
b0 = f;
a1 = 4*f*Te/(pi*sqrt(4+2*f^2));
num = [b2 b1 b0];
den = [1 a1 1];
fitness = funcioncosto(num,den);

% Update Alpha, Beta, and Delta
if fitness<Alpha_score
Alpha_score=fitness; % Update alpha
Alpha_pos=Positions(i,:);
end

if fitness>Alpha_score && fitness<Beta_score
Beta_score=fitness; % Update beta
Beta_pos=Positions(i,:);
```

```
end

if fitness>Alpha_score && fitness>Beta_score && fitness<Delta_score
Delta_score=fitness; % Update delta
Delta_pos=Positions(i,:);
end
end
a=2-l*((2)/Max_iter); % a decreases linearly fron 2 to 0
% Update the Position of search agents including omegas
for i=1:size(Positions,1)
for j=1:size(Positions,2)

r1=rand(); % r1 is a random number in [0,1]
r2=rand(); % r2 is a random number in [0,1]

A1=2*a*r1-a;
C1=2*r2;

D_alpha=abs(C1*Alpha_pos(j)-Positions(i,j));
X1=Alpha_pos(j)-A1*D_alpha;

r1=rand();
r2=rand();

A2=2*a*r1-a;
C2=2*r2;

D_beta=abs(C2*Beta_pos(j)-Positions(i,j));
X2=Beta_pos(j)-A2*D_beta;
```

```
r1=rand();
r2=rand();

A3=2*a*r1-a;
C3=2*r2;

D_delta=abs(C3*Delta_pos(j)-Positions(i,j));
X3=Delta_pos(j)-A3*D_delta;
Positions(i,j)=(X1+X2+X3)/3;
end
end

l=l+1;
Curves(k, l)=Alpha_score;
end
Best_pos(k,:)  = Alpha_pos;
Best_score(k)  = Alpha_score;
k
end

Res.gwo_best_pos = Best_pos;
Res.gwo_best_score = Best_score;
Res.gwo_curves = Curves;
Res.gwo_individuos = N;
Res.gwo_iteraciones = Max_iter ;
Res.gwo_repeticiones = Repeticiones;

save Res;
```

```
% Algoritmo WOA tomado de https://seyedalimirjalili.com/woa

global Datos;
Datos = load('Datos');

SearchAgents_no = 40;
Max_iter = 50;
lb = [1.0  0.3  0.001 10 0.2];
ub = [2   0.8   0.02  15 0.6];
dim = 5
Repetition = 20;

% Load
ex  = -1.4673;
ey  = 0.7713;
eh  = 1.7179;
eqx = -0.4901;
eqy = 0.8184;
eqh = 0.7257;

%Governor
Kp = 5.5912;
Ki = 1.061;
Kd = 3.2800;
T1v = 0.28;
bp = 0.04;

%Servo
Ty = 0.1;
```

```
for cont=1:Repetition
% initialize position vector and score for the leader
Leader_pos=zeros(1,dim);
Leader_score=inf; %change this to -inf for maximization problems

%Initialize the positions of search agents
Positions=generar_poblacion(SearchAgents_no,lb,ub,dim);

Convergence_curve=zeros(1,Max_iter);

t=0;% Loop counter

% Main loop
while t<Max_iter
for i=1:size(Positions,1)
% Return back the search agents that go beyond the boundaries of the search space
Flag4ub=Positions(i,:)>ub;
Flag4lb=Positions(i,:)<lb;
Positions(i,:)=(Positions(i,:).*(~(Flag4ub+Flag4lb)))+ub.*Flag4ub+lb.*Flag4lb;

% Calculate objective function for each search agent
% fitness=fobj(Positions(i,:));

% Calculate objective function for each search agent
%fitness=fobj(Positions(i,:));
Tw = Positions(i,1);
Te = Positions(i,2);
f = Positions(i,3);
Ta = Positions(i,4);
eg = Positions(i,5);
```

```
Taa=1/Ta;
Twe=-Tw/Te;
b2 = 2*f*Te^2/pi^2;
b1 = Te;
b0 = f;
a1 = 4*f*Te/(pi*sqrt(4+2*f^2));
num = [b2 b1 b0];
den = [1 a1 1];
fitness = funcioncosto(num,den);

% Update the leader
if fitness<Leader_score % Change this to > for maximization problem
Leader_score=fitness; % Update alpha
Leader_pos=Positions(i,:);
end

end

a=2-t*((2)/Max_iter);
a2=-1+t*((-1)/Max_iter);

% Update the Position of search agents
for i=1:size(Positions,1)
r1=rand(); % r1 is a random number in [0,1]
r2=rand(); % r2 is a random number in [0,1]

A=2*a*r1-a;
C=2*r2;
b=1;
l=(a2-1)*rand+1;
```

```
p = rand();

for j=1:size(Positions,2)
if p<0.5
if abs(A)>=1
rand_leader_index = floor(SearchAgents_no*rand()+1);
X_rand = Positions(rand_leader_index, :);
D_X_rand=abs(C*X_rand(j)-Positions(i,j));
Positions(i,j)=X_rand(j)-A*D_X_rand;

elseif abs(A)<1
D_Leader=abs(C*Leader_pos(j)-Positions(i,j));
Positions(i,j)=Leader_pos(j)-A*D_Leader;
end

elseif p>=0.5

distance2Leader=abs(Leader_pos(j)-Positions(i,j));
Positions(i,j)=distance2Leader*exp(b.*l).*cos(l.*2*pi)+Leader_pos(j);
end
end
end
t=t+1
Convergence_curve(t)=Leader_score;
end
Best_score(cont) = Leader_score;
Best_pos(cont,:)= Leader_pos;
Curves(cont,:) = Convergence_curve;
cont
end
```

```
Res.gwo_best_pos = Best_pos;
Res.gwo_best_score = Best_score;
Res.gwo_curves = Curves;
Res.gwo_individuos = N;
Res.gwo_iteraciones = Max_iter ;
Res.gwo_repeticiones = Repeticiones;

save Res;
```

7. Referencias

Referencias

[1] G. Ahmed. “Particle Swarm Optimization Algorithm and Its Applications”. En: *Archives of Computational Methods in Engineering* (2022), págs. 2531-2561.

[2] Diyi Chen et al. “Nonlinear Dynamical Analysis of Hydro-Turbine Governing System with a Surge Tank”. En: *Applied Mathematical Modelling* (2013), págs. 7611-7623.

[3] Diyi Chen et al. “Nonlinear Dynamical Analysis of Hydro-Turbine Governing Systems with a Surge Tank”. En: *Applied Mathematical Modelling* (2013), págs. 7611-7623.

[4] Zhihuan Chen et al. “Parameter Identification of Integrated Model of Hydraulic Turbine Regulating System With Uncertainties Using Three Different Approaches”. En: *IEEE Transactions on Power Systems* (2017), págs. 3482-3491.

[5] O. Colorado Arellano et al. “Algoritmo Genético Aplicado a la Sintonización de un Controlador PID para un Sistema Acoplado de Tanques”. En: *Pädi Boletín Científico De Ciencias Básicas E Ingenierías Del ICBI* 5.10 (2018).

[6] K. Dalibor y S. Gorazd. “Differential Evolution Based Identification of the Nonlinear Kaplan Turbine Model”. En: *IEEE Transactions on Energy Conversion* (2014), págs. 178-187.

[7] Hongqing Fan, Long Chen y Zuyi Shi. “Application of an Improved PSO Algorithm to Optimal Tuning of PID Gains for Water Turbine Governor”. En: *Energy Conversion and Management* (2011), págs. 1763-1770.

[8] Hongqing Fan et al. “Basic Modeling and Simulation Tool for Analysis of Hydraulic Transients in Hydroelectric Power Plants”. En: *IEEE Transactions on Energy Conversion* (2008), págs. 834-841.

[9] Hongquing Fan, Long Chen y Zuyi Shi. “Application of an Improved PSO Algorithm to Optimal Tuning of PID Gains for Water Turbine Governor”. En: *Energy Conversion and Management* (2011), págs. 1763-1770.

[10] Hongquing Fan et al. “Basic Modeling and Simulation Tool for Analysis of Hydraulic Transients in Hydroelectric Power Plants”. En: *IEEE Transactions on Energy Conversion* (2008), págs. 834-841.

[11] M. Gestal y D. Rivero. *Introducción a los Algoritmos Genéticos y la Programación Genética*. España: Universidad de Coruña, 2010.

[12] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. xiii, 412 p. Reading, Mass.: Addison-Wesley Pub. Co., 1989.

[13] Vicente Gómez Garay. *Algoritmos Genéticos*. Bilbao, España: Publicaciones Escuela Técnica Superior de Ingeniería de Bilbao, 2008. ISBN: 978-84-95809-43-8.

[14] Z. Hao et al. “Nonlinear Modeling and Dynamic Analysis of Hydro-Turbine Governing System in the Process of Load Rejection Transient”. En: *Energy Conversion and Management* (2015), págs. 128-137.

[15] John H. Holland. *Genetic Algorithms and Adaptation*. Springer, 1968.

[16] Chaoshun Li y Jianzhing Zhou. “Parameters Identification of Hydraulic Turbine Governing System Using Improved Gravitational Search Algorithm”. En: *Energy Conversion and Management* (2011), págs. 334-381.

[17] Chaoshun Li y Jianzhong Zhou. “Parameters Identification of Hydraulic Turbine Governing System Using Improved Gravitational Search Algorithm”. En: *Energy Conversion and Management* (2011), págs. 374-381.

[18] Chaoshun Li et al. “Parameter Identification of a Nonlinear Model of Hydraulic Turbine Governing System with an Elastic Water Hammer Based on a Modified Gravitational Search Algorithm”. En: *Emerging Applications of Artificial Intelligence* (2016), págs. 177-191.

[19] K.F. Man, K.S. Tang y S. Kwong. “Genetic Algorithms: Concepts and Applications”. En: *IEEE Transactions on Industrial Electronics* (1996), págs. 519-534.

[20] S. Mirjalili y A. Lewis. “The Whale Optimization Algorithm”. En: *Advances in Engineering Software* (2016), págs. 51-67.

[21] S. Mirjalili, S.M. Mirjalili y A. Lewis. “Grey Wolf Optimizer”. En: *Advances in Engineering Software* (2014), págs. 46-61.

[22] Ardul G. Munoz Hernandez, Sa’ad Petrous y Dewi Leuan. *Modelling and Controlling Hydropower Plants*. New York: Springer, 2013.

[23] M. Santos. “Un Enfoque Aplicado del Control Inteligente”. En: *Revista Iberoamericana de Automatización e Informática Industrial* 8.4 (2011), págs. 283-296. DOI: 10.1016/j.riai.2011.09.016.

[24] T. Tian et al. “An Improved Ant Lion Optimization Algorithm and Its Application in Hydraulic Turbine Governing System Parameter Identification”. En: *Energies MDPI* (2018), págs. 2-15.

[25] Dongshu Wang, Dapei Tang y Lei Li. “Particle Swarm Optimization Algorithm”. En: *Soft Computing* (2018), págs. 1433-7479.

[26] G.-G. Wang, S. Deb y L. Coelho. “Elephant Herding Optimization”. En: *3rd International Symposium on Computational and Business Intelligence (ISCBI)*. 2015, págs. 1-5.