Comparativa de métodos determinísticos para la sintonización óptima fuera de línea del controlador PI de un motor sin escobillas

Alam Gabriel Rojas López,
Miguel Gabriel Villarreal Cervantes
Sección de Mecatrónica
Departamento de Posgrado
CIDETEC-IPN
Ciudad de México, México
arojasl2101@alumno.ipn.mx, mvillarrealc@ipn.mx

Alejandro Rodríguez Molina
División de Investigación y Posgrado
IT de Tlalnepantla
Tecnológico Nacional de México
alejandro.rm@tlalnepantla.tecnm.mx

***Resumen*—En el presente trabajo se desarrollan tres propuestas de algoritmos híbridos que involucran combinaciones entre un método de búsqueda indirecta (basada en el gradiente) con tres métodos de búsqueda directa Hooke-Jeeves, Powell y Simplex aplicados el problema de sintonización fuera de línea del controlador Proporcional Integral para regular la velocidad de un motor sin escobillas. El problema se aborda con un enfoque de optimización en donde el error cuadrático integrado es el criterio a minimizar. Se realiza una comparación sobre el desempeño de los algoritmos híbridos contra su variante original (Hooke-Jeeves, Powell, Simplex y gradiente), los resultados muestran que los algoritmos híbridos reducen el número de iteraciones y dan una mejor convergencia a una mejor solución, siendo el algoritmo de combinación Simplex-Gradiente el de mejor desempeño.**

***Keywords*—** Sintonización de controladores, optimización, motor sin escobillas, algortimos híbridos, método de Powell, método de Hooke-Jeeves, método simplex, método del gradiente

I. Introducción.

El deseo de mejorar el desempeño de equipos y sistemas, ya sea por aumentar precisión o por reducir gastos, ha llevado a la necesidad de optimizar la manera en que se controlan. Diversos métodos han sido desarrollados para poder dar solución a estos problemas de optimización [1, 2, 3, 4]. Una clasificación común de estos métodos son los deterministas, donde ante mismas entradas y condiciones producen los mismos resultados [5], dicha categoría puede dividirse en dos subcategorías: i) métodos de búsqueda directa, los cuales mediante pasos lógicos y cambios medidos generan la dirección de búsqueda y ii) métodos de búsqueda indirecta, los cuales toman como dirección de búsqueda el gradiente del problema de optimización [6].

Diversos trabajos han mostrado la ventaja de los métodos de búsqueda directa como en [7] donde se diseña un lazo de control con una retroalimentación no lineal compuesta para el controlador de velocidad de un motor de corriente continua. En dicho trabajo la función no lineal de la retroalimentación se sintoniza planteando el problema como uno de minimización, el cual se resuelve por el método de Hooke-Jeeves. Dicha implementación se compara con un lazo de control con retroalimentación lineal, en donde se muestra que la propuesta tiene un menor sobre impulso (0.002 % contra 20.78 %) y un tiempo se asentamiento menor (10.36ms contra 39.3ms). Otro ejemplo de aplicación del método de Hooke-Jeeves se muestra en [8] donde también se emplea para sintonizar la retroalimentación no lineal para controlar el par de un sistema de direccionamiento eléctrico. Dicho trabajo muestra una comparativa entre un lazo de control convencional y el propuesto empleando el criterio de desempeño ITAE (error integral absoluto multiplicado por el tiempo) y muestra que la propuesta tiene un desempeño sobresaliente ($5.0095e^{-2}$ contra $9.1600e^{-2}$).

En [9] se emplea el método de Powell para optimizar la programación del punto de ajuste de un controlador difuso para un servomotor. Dicho trabajo muestra que el controlador optimizado por Powell elimina las oscilaciones residuales en la respuesta del sistema. En [2], se emplea el método Simplex para sintonizar un controlador PID para mejorar el desempeño de una máquina CNC, dicho trabajo muestra una comparativa entre la sintonización óptima y una manual comparando los resultados en diferentes criterios como el error cuadrático integrado ponderado en el tiempo (ITSE por sus siglas en inglés), el error absoluto integrado ponderado en el tiempo (ITAE por sus siglas en inglés), el error cuadrático integrado en versión generalizada (GISE por sus siglas en inglés) y el error cuadrático integrado ponderado en el tiempo en versión generalizada GITSE (GITSE por sus siglas en inglés), mostrando en todos los casos que la sintonización abordada por un enfoque de optimización mejora considerablemente los resultados.

Otro ejemplo donde se ha empleado el método Simplex se muestra en [10], donde se sintoniza el controlador PI en una planta integradora (con retraso) generalizada. La función objetivo es minimizar el IAE (error integral absoluto) y se comprobó la convexidad del problema de optimización. De igual manera en [11] se muestra una comparativa entre método Simplex y del Gradiente para la sintonización de un controlador de corriente para controlar la velocidad en un motor de corriente directa. Para las pruebas se realizaron variaciones aleatorias en los parámetros del sistema. Los resultados de dicha comparativa muestran que el método Simplex tiene un menor tiempo de asentamiento (0.308s contra 0.501s).

De igual manera, los métodos deterministas de búsqueda indirecta también se han implementado para la resolución de problemas de optimización para la sintonización de controladores. Un ejemplo de esto se muestra en [12] donde se emplea el método del Gradiente para la sintonización del control PID de un sistema de extracción de aceite esencial de destilación de vapor de hidro-difusión. En el trabajo mencionado se realiza un comparativa con la sintonización mediante el método heurístico de Ziegler Nichols. En dicha comparativa se muestra que el método del gradiente tiene un menor sobre impulso (0.67 % contra 0.98 %), menor tiempo

de asentamiento (1993s contra 2667s) y un menor error cuadrático medio (0.4239 contra 0.6707). Otro ejemplo de los métodos de tipo Gradiente se encuentra en [13] donde fue empleado para optimizar los parámetros de controlador en un sistema de energía tipo microgrid, se mostró la ventaja que presenta al sintonizar una gran cantidad de variables de diseño (10 variables en 6 ganancias, 2 sesgos de frecuencia y 2 caídas potencia-frecuencia).

A pesar de que en años recientes se han explotado el uso de algoritmos estocásticos [14, 15] para encontrar una solución mejorada en el proceso de sintonización de controladores, dejando de lado las implementaciones deterministas, se considera importante analizar los algoritmos que no presentan dicha aleatoriedad en la dirección de búsqueda para poder mejorar, ya sea el tiempo de convergencia o precisión, y así aplicarlos en trabajos futuros. Por tal motivo en este trabajo se presenta el desarrollo de tres algoritmos híbridos para el problema de sintonización de controladores aplicados a un motor sin escobillas. La característica principal de la propuesta es que en sistemas simples es posible prescindir de factores estocásticos para encontrar la solución.

La distribución del presente trabajo está organizada de la siguiente manera: en la sección II se presenta el modelo del sistema de un motor sin escobillas con un controlador PI así como el problema de optimización, la sección III muestra los algoritmos que se aplicarán al sistema considerando algunos métodos básico (Hooke-Jeeves, Powell, Simplex y Gradiente) y 3 algoritmos propuestos, los resultados de las pruebas se presentan en la sección IV para finalmente presentar las conclusiones en la sección V

II. Modelo del sistema

Un motor sin escobillas es en esencia un motor síncrono de imanes permanentes. Este tipo de motores tiene en el rotor los imanes y el estator es un circuito bobinado, que con un flujo de corriente en sus bobinas crea un campo magnético que mueve al rotor. La principal diferencia entre un motor sin escobillas y un motor síncrono es la forma de alimentación de las bobinas del estator. Un motor síncrono se alimenta con una señal sinusoidal, mientras que un motor sin escobillas se alimenta con una señal cuasi sinusoidal generada por un inversor [16].

Tomando en cuenta esto, la Fig. 1 muestra las dos secciones que conforman un motor sin escobillas [17]:

- Estructura física del Motor: Motor síncrono trifásico de imanes permanentes, generalmente el estator tiene una configuración estrella[18].
- Lógica de conmutación: Esta sección genera la señal de alimentación para el motor y tiene dos componentes principales [19]
  - Sensor de posición: Generalmente son 3 sensores de efecto Hall, separados entre ellos 120° que se activan cuando un imán incluido en el rotor pasa frente a ellos.
  - Inversor: Se alimenta con una señal de corriente directa convirtiéndola a una señal de corriente alterna, desfasada en 120° entre cada fase.

II-A. Análisis electro mecánico del motor

Para iniciar con el análisis del sistema, se puede representar el motor como en la Fig. 2. Se observa que es un motor trifásico simple, donde se tiene una sección eléctrica, la cual está compuesta por los devanados del motor donde fluye la corriente suministrada por el inversor. También se tiene una sección mecánica que representa la velocidad y posición del rotor. Entre estas dos secciones se tiene una relación de par electromotriz, generado por el campo magnético producido por la corriente que pasa por las bobinas, que permite mover la carga del rotor.

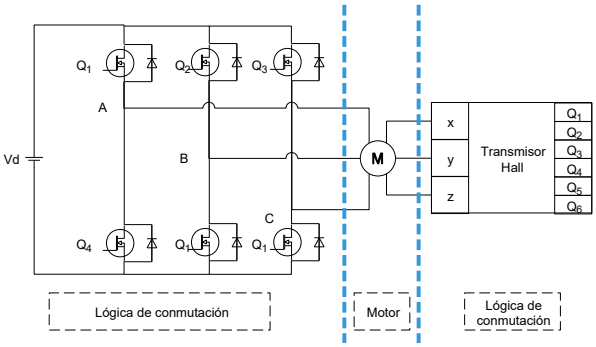

Figura 1: Esquema general de un motor sin escobillas

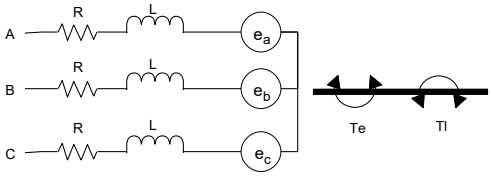

Figura 2: Esquema general de un motor

El sistema de ecuaciones que define la parte eléctrica de cada una de las fases está definida por la ec.(1) [20].

$$V_{\Phi N} = R \cdot i_{\Phi} + (L - M) \cdot \frac{di_{\Phi}}{dt} + e_{\Phi} \tag{1}$$

donde $\Phi \in \{A, B, C\}$ y

$V_{\Phi N}$ ← Voltaje de fase.
i_{Φ} ← Corriente de línea.
R ← Resistencia de línea.
L ← Inductancia de línea.
M ← Inductancia mutua.
e_{Φ} ← Back-emf por fase.

La fuerza contra electromotriz (Back-emf) al igual que el voltaje suministrado por el inversor, está definida en función de la posición del rotor como se puede ver en la ec. (2)[21], donde k_e es la constante de fuerza contra electromotriz, ω es la velocidad angular del rotor, $f_{\Phi}(\theta)$ es la función de la posición para por fase.

$$e_{\Phi} = k_e \cdot \omega \cdot f_{\Phi}(\theta) \tag{2}$$

Las corrientes calculadas mediante la ec. (1) producen un par eléctrico en el motor, el cual está definido en la ec. (3), donde k_t es la contante de par del motor.

$$T_e = k_t \left(i_A \cdot f_A(\theta) + i_B \cdot f_B(\theta) + i_C \cdot f_C(\theta) \right) \tag{3}$$

La ec. (4) [17] representa la relación entre el par eléctrico y la carga que mueve el rotor T_l. Donde J es la inercia del rotor y B es el coeficiente de fricción producida entre el rotor y el estator.

$$T_e - T_l = J \cdot \frac{d\omega}{dt} + B \cdot \omega \tag{4}$$

Finalmente, la velocidad angular del motor se puede representar por la ec.(5), donde P es el número de pares de polos del rotor.

$$\frac{d\theta}{dt} = \frac{P}{2}\omega \tag{5}$$

Un punto importante para terminar de describir el modelo del motor es definir las funciones de $f(\theta)$ que afectan tanto a la ec. (2) como a la ec. (3). Dichas funciones se obtienen mediante la segunda etapa del motor sin escobillas, es decir la lógica de conmutación.

II-B. Análisis de la lógica de conmutación

La lógica de conmutación es la encargada de generar la señal cuasisenoidal de alimentación del motor conforme a la posición del rotor y por ende la back-emf. Para analizar esta etapa, es importante comprender como funciona el sensor de efecto Hall dentro de un motor. En la Fig. 3 un primer grupo de sensores Hall se activa lo cual a su vez activa un par de fases del inversor (Q_1 y Q_5), lo que generara par eléctrico que moverá al rotor hasta una nueva posición lo que activará un nuevo par de fases.

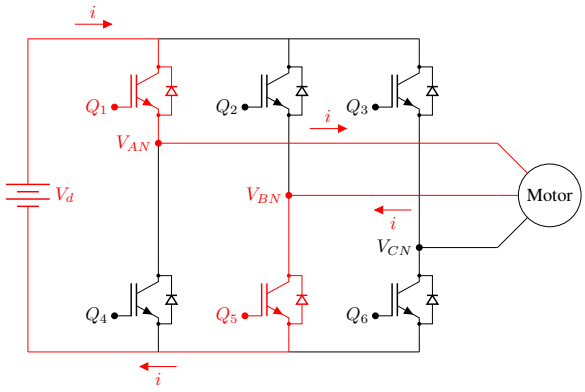

Figura 3: Lógica de conmutación en el estado 1

Finalmente, definiendo los sensores que se activarán en cada posición y las fases que se energizarán en cada una de esas posiciones, se pueden obtener las funciones dependientes de la posición del rotor las cuales se presentan en la Tabla I [22], donde la primer columna representa los rangos de θ que activan diferentes etapas del inversor, las siguientes tres columnas son las funciones de back-emf empleadas en la ec. 2 y las últimas tres columnas son las funciones de activación del inversor para cada Voltaje de fase.

II-C. Ecuaciones del espacio de estados

El sistema representado en espacio de estados estados se presenta en la ec. (6).

$$\frac{d}{dt}\begin{bmatrix} i_A \\ i_B \\ i_C \\ \omega \\ \theta \end{bmatrix} = A\begin{bmatrix} i_A \\ i_B \\ i_C \\ \omega \\ \theta \end{bmatrix} + B\begin{bmatrix} V_{AN} \\ V_{BN} \\ V_{CN} \\ T_l \end{bmatrix} \tag{6}$$

donde:

$$A = \begin{bmatrix} -\frac{R}{L} & 0 & 0 & -\frac{k_e}{L}f_A(\theta) & 0 \\ 0 & -\frac{R}{L} & 0 & -\frac{k_e}{L}f_B(\theta) & 0 \\ 0 & 0 & -\frac{R}{L} & -\frac{k_e}{L}f_C(\theta) & 0 \\ \frac{k_t}{J}f_A(\theta) & \frac{k_t}{J}f_B(\theta) & \frac{k_t}{J}f_C(\theta) & -\frac{B}{J} & 0 \\ 0 & 0 & 0 & \frac{P}{2} & 0 \end{bmatrix} \tag{7}$$

$$B = \begin{bmatrix} \frac{1}{L} & 0 & 0 & 0 \\ 0 & \frac{1}{L} & 0 & 0 \\ 0 & 0 & \frac{1}{L} & 0 \\ 0 & 0 & 0 & -\frac{1}{J} \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{8}$$

Para desarrollar el control PI es necesario encontrar la señal de control u que será el voltaje V_d suministrado al inversor el cual está dado por la ec. (9) [23], donde $e = \omega_d - \omega$ es el error producido entre la velocidad angular deseada ω_d y la medida ω.

$$u = k_p e + ki \int e dt \tag{9}$$

Agregando el error e como un estado $[i_A, i_B, i_C, \omega, \theta, e]$ y transformando vector del espacio de estados por el vector $x = [x_1, x_2, x_3, x_4, x_5, x_6]^T$, se obtiene el modelo en lazo cerrado con el controlador PI presentado en la ec. (10), donde $\bar{x}_4 = \omega_d$ es la velocidad angular deseada .

$$\dot{x} = \widehat{A} \cdot x + \widehat{B} \cdot \begin{bmatrix} \bar{x}_4 \\ T_l \end{bmatrix} \tag{10}$$

donde

$$\widehat{A} = \begin{bmatrix} -\frac{R}{L} & 0 & 0 & -\frac{k_e}{L}f_A(\theta) - \frac{k_p}{2L}fV_A(\theta) & 0 & \frac{k_i}{2L}fV_A(\theta) \\ 0 & -\frac{R}{L} & 0 & -\frac{k_e}{L}f_B(\theta) - \frac{k_p}{2L}fV_B(\theta) & 0 & \frac{k_i}{2L}fV_B(\theta) \\ 0 & 0 & -\frac{R}{L} & -\frac{k_e}{L}f_C(\theta) - \frac{k_p}{2L}fV_C(\theta) & 0 & \frac{k_i}{2L}fV_C(\theta) \\ \frac{k_t}{J}f_A(\theta) & \frac{k_t}{J}f_B(\theta) & \frac{k_t}{J}f_C(\theta) & -\frac{B}{J} & 0 & 0 \\ 0 & 0 & 0 & \frac{P}{2} & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \end{bmatrix} \tag{11}$$

$$\widehat{B} = \begin{bmatrix} \frac{k_p}{2L}fV_A(\theta) & 0 \\ \frac{k_p}{2L}fV_B(\theta) & 0 \\ \frac{k_p}{2L}fV_C(\theta) & 0 \\ 0 & -\frac{1}{J} \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \tag{12}$$

Este trabajo busca encontrar la ganancia proporcional k_p y derivativa k_i para un controlador PI que minimicen el error cuadrático integrado para la regulación de velocidad de un motor sin escobillas y que está sujeta a las restricciones propias de los límites en las variables de diseño. Por lo tanto, el problema formal de optimización se formula en la ec. (13).

$$\begin{aligned} &\min_{k_p,k_i} J = \tfrac{1}{2}\int_0^{t_f} e(t)^2\, dt \\ &\textit{sujeto a}: \begin{cases} \dot{x} = f(\bar{x}_4, k_p, k_i) \\ 1 \geq k_p \geq 1000 \\ 1 \geq k_i \geq 1000 \end{cases} \end{aligned} \tag{13}$$

III. Algoritmos de optimización

En el presente trabajo se emplearán métodos deterministas, que tienen la característica de que ante la mismas entradas y mismos parámetros se obtendrá la misma respuesta [24]. Los métodos deterministas de búsqueda directa se presentan en los Algoritmos Hooke-Jeeves 1, Powell 2 y Simplex 3 [25, 5]. Por otro lado, un método determinista de búsqueda indirecta se presenta en el Algoritmo de Cauchy

Tabla I: Funciónes dependientes de la posición.

Posición					Back-emf			Voltaje por fase		
					$f_A(\theta)$	$f_B(\theta)$	$f_C(\theta)$	$f_{V_A}(\theta)$	$f_{V_B}(\theta)$	$f_{V_C}(\theta)$
0	$\leq$	θ	$<$	$\frac{\pi}{3}$	1	-1	$1-\frac{6\cdot\theta}{\pi}$	1	-1	0
$\frac{\pi}{3}$	$\leq$	θ	$<$	$\frac{2\pi}{3}$	1	$-3+\frac{6\cdot\theta}{\pi}$	-1	1	0	-1
$\frac{2\pi}{3}$	$\leq$	θ	$<$	π	$5-\frac{6\cdot\theta}{\pi}$	1	-1	0	1	-1
π	$\leq$	θ	$<$	$\frac{4\pi}{3}$	-1	1	$-7+\frac{6\cdot\theta}{\pi}$	-1	1	0
$\frac{4\pi}{3}$	$\leq$	θ	$<$	$\frac{5\pi}{3}$	-1	$9-\frac{6\cdot\theta}{\pi}$	1	-1	0	1
$\frac{5\pi}{3}$	$\leq$	θ	$<$	2π	$-11+\frac{6\cdot\theta}{\pi}$	-1	1	0	-1	1

(Gradiente descendente) 4 [26], el cual emplea como dirección de búsqueda el gradiente del problema a optimizar.

III-A. Algoritmos Originales

El Algoritmo de Hooke-Jeeves 1, emplea como direcciones de búsqueda las componentes ortogonales del espacio creado por las variables de diseño (k_p y k_i). En cada iteración, el punto de búsqueda se desplazará, ya sea aumentando o reduciendo cada una de las componentes ortogonales, evaluando el mejor resultado proporcionado. Al terminar el recorrido de las componentes ortogonales, dará un cambio más, generado por la dirección entre el punto al iniciar la búsqueda y al terminar el recorrido de las componentes [27].

Algoritmo 1 Hook-Jeeves

1: Establecer el punto inicial de búsqueda $\rightarrow x$
2: Establecer incertidumbre $\rightarrow$ e ▷ e= $1e^{-7}$
3: Establecer las direcciones de búsqueda como las componentes$\rightarrow \Delta$
4: Establecer el tamaño del paso $\rightarrow \beta$ ▷ inicial $\beta = 1$
5: Establecer el punto y_{sig} = x
6: Establecer el punto x_{sig} = x
7: **while** Criterio >e **do** ▷ Criterio=β
8: **for** i **do**=1:n ▷ n=componentes
9: Suma=y_{sig}+$\beta\Delta$i
10: Resta=y_{sig}-$\beta\Delta$i
11: **if** f(Suma)<f (y_{sig}) **then**
12: y_{sig}=Suma
13: **else if** f(Resta)<f(y_{sig}) **then**
14: y_{sig}=Resta
15: **else**
16: y_{sig}=y_{sig}
17: **end if**
18: **end for**
19: **if** f(y_{sig}) <f(x_{sig}) **then**
20: Calcular λ^*
21: y_{sig}= y_{sig}+$\lambda^*(y_{sig} - x_{sig})$
22: x_{sig} = y_{sig}
23: **else**
24: $\beta = \beta/$*reducción* ▷ reducción=10
25: $y_{sig} = x_{sig}$
26: **end if**
27: **end while**

De igual manera el Algoritmo de Powell 2 emplea como direcciones iniciales de búsqueda las componentes ortogonales del problema de optimización k_p y k_i. Tiene un comportamiento similar al Algoritmo de Hooke-Jeeves1 con la diferencia que la última dirección de búsqueda se remplaza en cada iteración por la dirección producida por la diferencia entre el punto final de y el punto inicial de cada iteración de búsqueda [28]. Esto quiere decir que solo el inicio del algoritmo utiliza las componentes ortogonales como componentes de búsqueda, en resto de las iteraciones las componentes están formadas por los cambios previos.

El Algoritmo Simplex 3 emplea $n+1$ puntos de búsqueda iniciales donde n es la cantidad de componentes del problema de optimización, en este caso se tiene tres vectores de la forma $[k_p, k_i]^T$. En cada iteración se ordena los puntos de mejor a peor y mediante una secuencia de operaciones, se busca un nuevo punto que pueda remplazar al peor y en caso

Algoritmo 2 Powell

1: Establecer el punto inicial de búsqueda $\rightarrow x$
2: Establecer incertidumbre $\rightarrow$ e ▷ e= $1e^{-7}$
3: Establecer n direcciones S para cada componente del punto x
4: $x_{sig} = x$
5: $y_{sig} = x$
6: **for** i=1:n **do**
7: Calcular λ^*
8: $Cambio_i = y_{sig} + \lambda^* S_i$
9: **end for**
10: $y_{sig} = Cambio_{best}$
11: **while** Criterio>e **do** ▷ Criterio=|$f(x_{sig}) - f(Z)$|
12: Z=x_{sig}
13: **for** i=1:n **do**
14: Calcular λ^*
15: $y_{sig} = y_{sig} + \lambda^* S_i$
16: **end for**
17: $S_{aux} = y_{sig} - Z$
18: Calcular λ^*
19: $y_{sig} = y_{sig} + \lambda^* S_{aux}$
20: $x_{sig} = y_{sig}$
21: **for** i=1:n **do**
22: **if** i==n **then**
23: $S_i = S_{aux}$
24: **else**
25: $S_i = S_{i+1}$
26: **end if**
27: **end for**
28: **end while**

de que no se encuentre, se atraen los puntos hacia el mejor de los puntos ordenados [29]. Para este trabajo, uno de los puntos iniciales es el mismo que en los otros algoritmos y los otros dos puntos son $[6, 26]^T$ y $[10, 30]^T$

El Algoritmo de Cauchy (Gradiente) 4 toma como dirección de búsqueda el gradiente del problema de optimización de la ec.(13) mediante las funciones de sensitividad, que no son más que las componentes de la derivada de la función objetivo en términos de las variables de diseño [30]. Para este algoritmo se reduce el criterio de paro debido a que se notó que la búsqueda converge rápidamente a mínimos locales y tiene poca mejora en los resultados. De cualquier forma, también se optó por aumentar el número de iteraciones máximo a 500 iteraciones mientras que en los demás algoritmos es de 50.

III-B. Algoritmos propuestos

En los Algoritmos Hooke-Gradiente 5, Powell-Gradiente 6 y Simplex-Gradiente 7 se muestran los tres algoritmos propuestos para este trabajo, los cuales son una combinación de cada uno de los métodos de búsqueda directa con el de búsqueda indirecta. Los Algoritmos Hooke-Gradiente 5 y Powell-Gradiente 6 remplazan la dirección de búsqueda inicial por las componentes del gradiente y mantienen los mismos criterios de los algoritmos 1 y 2.

Algoritmo 3 Simplex

Establecer x_i puntos, $i \in \{1, \ldots, n+1\}$
Establecer incertidumbre $\rightarrow$ e ▷ e= $1e^{-7}$
while Criterio>e **do** ▷ Criterio |f(x_2)-f(x_1)|
  Ordenar los puntos valuados de menor a mayor desde 1 a n+1
  Calcular el centroide ▷ $x_o = \frac{1}{n}\sum_{i=1}^{1=n} x_i$
  Calcular el punto de reflexión ▷ $x_r = x_o + (x_o - x_n)$
  if f(x_r)<f(x_n) **then**
    if f(x_r)<f(x_1) **then**
      Calcular expansión ▷ $x_e = x_o + 2(x_r - x_o)$
      $x_{n+1} = x_e$
    else
      $x_{n+1} = x_r$
    end if
  else
    if f(x_n)<f(x_r) y f(x_r)<f(x_{n+1}) **then**
      Calcular contracción exterior ▷ $x_{ce} = x_o + 0.5(x_r - x_o)$
      if f(x_{ce})<f(x_r) **then**
        $x_{n+1} = x_{ce}$
      else
        Encoger ▷ $x_i = x_1 + 0.5(x_j - x_1) \quad j = 2, ..., n$
      end if
    end if
    if f(x_r) $\geq$ f(x_{n+1}) **then**
      Calcular contracción interior ▷ $x_{ci} = x_o + 0.5(x_n - x_o)$
      if f(x_{ci})<f(x_n) **then**
        $x_{n+1} = x_{ci}$
      else
        Encoger ▷ $x_i = x_1 + 0.5(x_j - x_1) \quad j = 2, ..., n$
      end if
    end if
  end if
end while

Algoritmo 4 Gradiente

Establecer el punto inicial k
Establecer incertidumbre $\rightarrow$ e ▷ e= $1e^{-7}$
while Criterio>e **do** ▷ Criterio= $\frac{\partial J}{\partial k}\left(\frac{\partial J}{\partial k}\right)^T$
  Resolver las ecuaciones con k
  Resolver funciones de sensitividad ▷ $\dot{S_k} = \frac{\partial f}{\partial x}S_k + \frac{\partial f}{\partial k}$
  Se obtiene el gradiente ▷ $\frac{\partial J}{\partial k} = \int_0^t \left(\frac{\partial L}{\partial x}S_k + \frac{\partial L}{\partial k}\right)dt$
  Se obtiene el siguiente punto k ▷ $k_{i+1} = k_i - \lambda\frac{\partial J}{\partial k_i}$
end while

Algoritmo 5 Hook-Jeeves-Gradiente

Establecer el punto inicial de busqueda $\rightarrow x$
Establecer incertidumbre $\rightarrow$ e ▷ e= $1e^{-7}$
Establecer el tamaño del paso $\rightarrow \beta$
Establecer el punto y_{sig} = x
Establecer el punto x_{sig} = x
Resolver funciones de sensitividad ▷ $\dot{S_k} = \frac{\partial f}{\partial x}S_k + \frac{\partial f}{\partial k}$
Se obtiene el gradiente ▷ $\frac{\partial J}{\partial k} = \int_0^t \left(\frac{\partial L}{\partial x}S_k + \frac{\partial L}{\partial k}\right)dt$
$\Delta = \frac{\partial J}{\partial k}$ ▷ Dirección de búsqueda
while Criterio >e **do** ▷ Criterio=β
  for i **do**=1:n ▷ n=componentes
    Suma=y_{sig}+$\beta\Delta$i
    Resta=y_{sig}-$\beta\Delta$i
    if f(Suma)<f (y_{sig}) **then**
      y_{sig}=Suma
    else if f(Resta)<f(y_{sig}) **then**
      y_{sig}=Resta
    else
      y_{sig}=y_{sig}
    end if
  end for
  if f(y_{sig}) <f(x_{sig}) **then**
    Calcular λ^*
    y_{sig}= y_{sig}+$\lambda^*(y_{sig} - x_{sig})$
    x_{sig} = y_{sig}
  else
    $\beta = \beta$/*reducción* ▷ reducción=10
    $y_{siguiente} = x_{sig}$
  end if
end while

Algoritmo 6 Powell-Gradiente

Establecer el punto inicial de búsqueda $\rightarrow x$
Establecer incertidumbre $\rightarrow$ e ▷ e= $1e^{-7}$
Establecer n direcciones S para cada componente del punto x
$x_{sig} = x$
$y_{sig} = x$
Resolver funciones de sensitividad ▷ $\dot{S_k} = \frac{\partial f}{\partial x}S_k + \frac{\partial f}{\partial k}$
Se obtiene el gradiente ▷ $\frac{\partial J}{\partial k} = \int_0^t \left(\frac{\partial L}{\partial x}S_k + \frac{\partial L}{\partial k}\right)dt$
for i=1:n **do**
  $S_i = \frac{\partial J}{\partial k}$
end for
for i=1:n **do**
  Calcular λ^*
  $Cambio_i = y_{sig} + \lambda^* S_i$
end for
$y_{sig} = Cambio_{best}$
while Criterio>e **do** ▷ Criterio=$|f(x_{sig}) - f(Z)|$
  Z=x_{sig}
  for i=1:n **do**
    Calcular λ^*
    $y_{sig} = y_{sig} + \lambda^* S_i$
  end for
  $S_{aux} = y_{sig} - Z$
  Calcular λ^*
  $y_{sig} = y_{sig} + \lambda^* S_{aux}$
  $x_{sig} = y_{sig}$
  for i=1:n **do**
    if i==n **then**
      $S_i = S_{aux}$
    else
      $S_i = S_{i+1}$
    end if
  end for
end while

El Algoritmo Simplex-Gradiente 7 toma la acción principal de calcular el punto de reflexión, a partir de dicho punto, emplea la dirección del gradiente 4 para aproximar la reflexión hacia el mínimo más cercano, tomando el nuevo punto como el punto de reflexión mediante el cual se realizan el resto de los pasos.

Algoritmo 7 Simplex-Gradiente

Establecer x_i puntos, $i \in \{1, \ldots, n+1\}$
Establecer incertidumbre $\rightarrow$ e ▷ e= $1e^{-7}$
while Criterio>e **do** ▷ Criterio |f(x_2)-f(x_1)|
  Ordenar los puntos de menor a mayor desde 1 a n+1
  Calcular el centroide ▷ $x_o = \frac{1}{n}\sum_{i=1}^{1=n} x_i$
  Calcular el punto de reflexión ▷ $x_r = x_o + (x_o - x_n)$
  Resolver funciones de sensitividad ▷ $\dot{S_k} = \frac{\partial f}{\partial x}S_k + \frac{\partial f}{\partial k}$
  Se obtiene el gradiente ▷ $\frac{\partial J}{\partial k} = \int_0^t \left(\frac{\partial L}{\partial x}S_k + \frac{\partial L}{\partial k}\right)dt$
  Calcular λ^*
  $x_r = x_r - \lambda^*\frac{\partial J}{\partial k}$
  if f(x_r)<f(x_n) **then**
    if f(x_r)<f(x_1) **then**
      Calcular expansión ▷ $x_e = x_o + 2(x_r - x_o)$
      $x_{n+1} = x_e$
    else
      $x_{n+1} = x_r$
    end if
  else
    if f(x_n)<f(x_r) y f(x_r)<f(x_{n+1}) **then**
      Calcular contracción exterior ▷ $x_{ce} = x_o + 0.5(x_r - x_o)$
      if f(x_{ce})<f(x_r) **then**
        $x_{n+1} = x_{ce}$
      else
        Encoger ▷ $x_i = x_1 + 0.5(x_j - x_1) \quad j = 2, ..., n$
      end if
    end if
    if f(x_r) $\geq$ f(x_{n+1}) **then**
      Calcular contracción interior ▷ $x_{ci} = x_o + 0.5(x_n - x_o)$
      if f(x_{ci})<f(x_n) **then**
        $x_{n+1} = x_{ci}$
      else
        Encoger ▷ $x_i = x_1 + 0.5(x_j - x_1) \quad j = 2, ..., n$
      end if
    end if
  end if
end while

III-C. Búsqueda unidireccional

Es importante mencionar que el método de división de intervalos a la mitad (Interval Halving method) [25, 5]

presentado en el Algoritmo 8 se emplea en conjunto con algunos de los algoritmos descritos anteriormente. Debido a su sencillez se emplea para calcular el tamaño de paso óptimo (λ^*).

Algoritmo 8 Método por intervalo a la mitad.

1: Establecer límite inferior → a
2: Establecer límite superior → b
3: Establecer incertidumbre → e ▷ e= $1e^{-4}$
4: Calcular número de iteraciones → n ▷ $n = \frac{\log\left(\frac{e}{b-a}\right)}{\log(0.5+1)}$
5: Calcular longitud de intervalo → L ▷ L=b-a
6: **for** i=0 : (e>L y i<n) **do**
7: Calcular cuarto inferior → λ ▷ λ=a+(L/4)
8: Calcular punto medio → α ▷ α=(a+b)/2
9: Calcular cuarto superior → μ ▷ μ=b-(L/4)
10: **if** f(λ) <f(α) **then**
11: a = a
12: α = λ
13: b = $alpha$
14: **else if** f(μ) <f(α) **then**
15: a = α
16: α = μ
17: b = b
18: **else**
19: a = a
20: α = α
21: b = b
22: **end if**
23: Calcular longitud de intervalo → L
24: **end for**
25: $\lambda^* = \alpha$

IV. RESULTADOS

Antes de comenzar a describir los resultados, se repasará las condiciones con las que se realizaron las pruebas. Para empezar, la dinámica del motor sin escobillas en lazo cerrado ec. (10) se resolvió mediante el método de Runge Kutta de cuarto orden [31], donde la dinámica se valuó a lo largo de 0.1 segundos, en intervalos $dt = 1e^{-4}$ segundos. Se realizaron 30 corridas para cada algoritmo, cada una constaba de dos criterios de paro, los algoritmos se detendrían si se cumplen 50 iteraciones (500 para el algoritmo de Gradiente 4) o si el criterio de incertidumbre e cumple con $e < 1e^{-7}$. Todos los algoritmos calculan el tamaño de su paso de cambio λ^* mediante el algoritmo 8 (Intervalos a la mitad) con una incertidumbre $e < 1e^{-4}$. Los puntos de partida para cada corrida son los mismos (generados mediante un generador de números enteros aleatorios). En las Tablas III, IV, V, VI, VII, VIII y IX se muestran los resultados obtenidos de cada una de las corridas de todos los algoritmos, dichas tablas incluyen los pares $[k_p, k_i]$ iniciales, los pares finales, el valor $\min J$ al que se llegó, y el número de iteraciones que le tomo a la corrida.

Tabla II: Datos del motor sin escobillas.

R	=	0.6 Ω
L	=	0.8 mH
M	=	0.057 mH
k_e	=	0.035 Vs/rad
k_t	=	0.035 Nm/A
B	=	$24e^{-6}$ kgm^2
J	=	$100e^{-6}$ kgm^2
P	=	8
T_l	=	0 Nm
Vd	=	23 V
ω_d	=	250 rad/s

Tabla III: Resultados del algoritmo Hooke-Jeeves.

Corrida	k_p inicial	k_i inicial	k_p final	k_i final	mín J	Iteraciones
1	471	231	480.4422	240.5154	2.95e-07	10
2	603	712	530.3894	784.6106	5.69e-07	16
3	595	263	530.382	327.618	5.99e-07	15
4	845	195	1019.1	369.1151	9.04e-07	25
5	905	980	1019	1094	9.14e-07	50
6	86	3	87.001	-537.0008	2.07e-06	50
7	82	54	86.5135	216.9184	2.08e-06	50
8	88	56	86.5135	216.9528	2.08e-06	50
9	65	38	75.8782	49.3732	2.50e-06	50
10	63	59	80.3825	76.4925	2.53e-06	14
11	431	185	442	734.9998	2.59e-06	50
12	100	10	94.3847	554.6151	2.94e-06	50
13	439	112	446.7692	119.7735	3.47e-06	10
14	312	924	317.2911	929.2949	4.93e-06	10
15	36	94	44.5851	102.5961	6.67e-06	23
16	50	10	60.1344	20.1404	8.47e-06	10
17	259	409	267.952	582.3454	8.85e-06	50
18	21	31	32	335.2199	1.52e-05	37
19	3	4	32.9508	33.9551	1.82e-05	12
20	10	731	19.5731	740.5813	2.99e-05	10
21	222	118	240.7769	136.8181	3.24e-05	11
22	226	171	226	720.9998	3.51e-05	50
23	228	436	236.1516	444.2627	3.58e-05	50
24	5	10	25.2738	30.2738	8.93e-05	50
25	5	6	25.3156	26.3266	1.13e-04	16
26	10	100	19.9033	109.9033	1.52e-04	10
27	1	10	19.9457	28.9457	2.14e-04	50
28	10	10	19.9505	19.9505	2.22e-04	50
29	1	1	19.9505	19.9505	2.22e-04	50
30	10	1	19.9552	10.9552	2.30e-04	10
					4.86e-05	939

Tabla IV: Resultados del algoritmo Powell.

Corrida	k_p inicial	k_i inicial	k_p final	k_i final	min J	Iteraciones
1	431	185	403.0376	192.145	2.93e-07	2
2	471	231	480.4426	231.0719	2.95e-07	2
3	603	712	530.3898	811.3103	5.67e-07	6
4	595	263	530.3818	317.6694	6.00e-07	5
5	905	980	1018.9	1152.2	8.47e-07	9
6	845	195	1019.1	471.5592	8.96e-07	12
7	88	56	86.5136	217.0685	2.08e-06	7
8	86	3	86.5163	-17	2.09e-06	1
9	100	10	86.5168	-19.9994	2.09e-06	2
10	439	112	434.4009	112.0158	2.22e-06	2
11	82	54	78.2218	54.0363	2.43e-06	2
12	63	59	64.3461	59.001	4.11e-06	2
13	65	38	64.3577	38.0055	4.14e-06	2
14	312	924	317.2922	924.0044	4.93e-06	2
15	10	731	44.3183	731.0112	5.65e-06	3
16	10	100	44.6341	100.0178	6.62e-06	3
17	21	31	44.6686	31.0121	6.75e-06	3
18	1	10	44.6791	10.0304	6.79e-06	3
19	10	10	44.6791	10.0239	6.79e-06	3
20	3	4	44.6821	4.0342	6.80e-06	3
21	1	1	44.6837	1.0315	6.81e-06	3
22	10	1	44.6837	1.0131	6.81e-06	3
23	259	409	277.0364	409.0265	6.92e-06	2
24	36	94	42.3727	94.004	7.12e-06	2
25	5	10	36.9732	42.3927	9.35e-06	3
26	5	6	36.976	38.3933	9.36e-06	3
27	50	10	52.1693	10.0021	1.10e-05	2
28	228	436	235.8406	436.0112	2.76e-05	2
29	226	171	235.9025	171.0051	3.06e-05	2
30	222	118	235.9149	118.0127	3.12e-05	2
					7.13e-06	98

Tabla V: Resultados del algoritmo Simplex.

Corrida	k_p inicial	k_i inicial	k_p final	k_i final	min J	Iteraciones
1	603	712	2245.8	2599.5	5.52e-08	3
2	845	195	1270.5	285.5	9.60e-08	2
3	905	980	2259.5	2417	9.66e-08	3
4	259	409	1279	1949	9.99e-08	6
5	431	185	486.875	207.625	2.29e-07	5
6	226	171	489.125	345.0625	2.29e-07	7
7	228	436	484.0313	906.4688	2.35e-07	6
8	312	924	471	1379	2.83e-07	4
9	439	112	405.7656	13.6875	2.95e-07	5
10	471	231	471	231	2.96e-07	1
11	222	118	548	258	1.12e-06	4
12	10	731	83.5	11560	1.58e-06	7
13	10	100	82	1047	2.16e-06	6
14	36	94	83.5	203.25	2.19e-06	8
15	1	1	86	296	2.21e-06	7
16	1	10	86	210.5	2.22e-06	7
17	82	54	86	58	2.23e-06	1
18	86	3	86	3	2.23e-06	5
19	63	59	84.1418	74.511	2.25e-06	9
20	50	10	86.375	5.125	2.30e-06	6
21	3	4	87.3125	314.2344	2.30e-06	9
22	88	56	80	54.5	2.35e-06	5
23	100	10	79	16.5	2.37e-06	6
24	5	6	76.625	382.8125	2.59e-06	7
25	5	10	76.625	322.5625	2.62e-06	7
26	21	31	77.1875	51.5625	2.62e-06	5
27	595	263	595	263	3.73e-06	1
28	65	38	100.9375	47.5	5.62e-06	4
29	10	10	42	22	8.93e-06	9
30	10	1	40.9688	-0.9395	8.94e-06	12
					2.15e-06	167

Tabla VI: Resultados del algoritmo del Gradiente.

Corrida	k_p inicial	k_i inicial	k_p final	k_i final	min J	Iteraciones
1	471	231	438.0031	230.7947	9.95e-07	33
2	431	185	437.9997	184.9377	1.00e-06	7
3	439	112	438.0028	111.9254	1.02e-06	1
4	905	980	736.0003	979.8988	2.57e-06	169
5	603	712	736.0001	711.7748	2.66e-06	125
6	595	263	736	262.7542	2.80e-06	143
7	845	195	736.0003	194.9298	2.83e-06	109
8	10	731	25.0103	694.8881	2.48e-05	357
9	312	924	184.9987	922.8937	5.40e-05	500
10	259	409	184.9987	406.372	6.80e-05	500
11	226	171	184.9941	167.6643	7.93e-05	500
12	222	118	184.9999	115.908	8.18e-05	500
13	63	59	184.9976	56.3621	8.43e-05	500
14	88	56	184.9991	53.6209	8.43e-05	500
15	82	54	185.9989	51.5666	8.44e-05	500
16	65	38	184.9975	36.3215	8.51e-05	500
17	50	10	184.9975	8.3129	8.62e-05	500
18	100	10	184.9994	8.6766	8.63e-05	500
19	86	3	184.9988	1.4879	8.66e-05	500
20	228	436	185.9941	432.372	9.33e-05	500
21	10	100	24.8448	65.1918	1.07e-04	247
22	36	94	185.9951	91.3446	1.10e-04	500
23	21	31	24.888	29.2434	1.19e-04	12
24	1	10	24.9207	8.883	1.27e-04	28
25	5	10	24.9208	8.8626	1.27e-04	24
26	10	10	24.9208	8.8479	1.27e-04	19
27	5	6	24.9373	5.2034	1.28e-04	22
28	3	4	24.9378	3.2335	1.29e-04	25
29	1	1	24.9384	0.2711	1.31e-04	26
30	10	1	24.9388	0.2013	1.31e-04	17
					7.50e-05	7864

Tabla VII: Resultados del algoritmo Hooke-Jeeves-Gradiente.

Corrida	k_p inicial	k_i inicial	k_p final	k_i final	min J	Iteraciones
1	471	231	480.4426	231.0043	2.95e-07	33
2	603	712	530.3882	712.0161	5.73e-07	50
3	595	263	530.381	263.0146	6.03e-07	50
4	905	980	1018.9	980.0109	8.59e-07	50
5	845	195	1019.2	195.0162	9.18e-07	50
6	82	54	86.5138	54.0287	2.09e-06	50
7	88	56	86.5136	56.0229	2.09e-06	50
8	86	3	87.002	2.9781	2.10e-06	50
9	63	59	80.1886	59.055	2.37e-06	50
10	65	38	75.8667	38.0467	2.50e-06	50
11	431	185	442	185.003	2.89e-06	50
12	439	112	446.7709	112.0017	3.48e-06	50
13	100	10	94.386	10.8331	3.50e-06	50
14	312	924	317.2922	924.0248	4.93e-06	50
15	36	94	44.4513	94.0683	6.69e-06	50
16	50	10	60.1395	10.0191	8.55e-06	50
17	259	409	267.9514	409.0834	1.46e-05	50
18	21	31	31.9994	33.4322	2.58e-05	50
19	228	436	235.8406	436.0087	2.76e-05	50
20	10	731	19.5781	731.103	3.03e-05	50
21	222	118	240.778468	118.026507	3.24e-05	50
22	3	4	32.9656	4.5607	3.31e-05	50
23	226	171	226	171.0104	4.68e-05	50
24	5	6	25.2859	6.4674	9.70e-05	50
25	5	10	25.2839	10.5327	1.20e-04	50
26	10	100	19.9085	100.0973	1.57e-04	50
27	1	10	19.9557	10.04	2.31e-04	50
28	10	10	19.9557	10.0012	2.31e-04	50
29	1	1	19.9603	1.2519	2.39e-04	50
30	10	1	19.9604	1.0201	2.39e-04	50
					5.23e-05	1483

Tabla VIII: Resultados del algoritmo Powell-Gradiente.

Corrida	k_p inicial	k_i inicial	k_p final	k_i final	min J	Iteraciones
1	431	185	403.0988	184.9924	2.93e-07	2
2	471	231	480.4426	231.002	2.95e-07	2
3	603	712	530.3882	711.9859	5.73e-07	4
4	595	263	530.381	262.9874	6.03e-07	3
5	905	980	1018.9	980.01	8.59e-07	5
6	845	195	1019.2	195.0156	9.18e-07	7
7	88	56	86.5136	55.9973	2.09e-06	1
8	86	3	86.5163	3.001	2.09e-06	1
9	100	10	86.5161	9.9793	2.09e-06	2
10	439	112	434.4009	111.999	2.22e-06	1
11	82	54	78.2218	53.9929	2.43e-06	1
12	63	59	64.3461	59.0043	4.11e-06	1
13	65	38	64.3577	37.9985	4.14e-06	1
14	312	924	317.2922	924.0017	4.93e-06	1
15	10	731	44.3185	730.6311	5.65e-06	2
16	21	31	44.6685	31.243	6.75e-06	2
17	5	10	44.6791	10.1712	6.79e-06	3
18	1	10	44.6791	10.2108	6.79e-06	3
19	5	6	44.6811	6.2077	6.80e-06	3
20	3	4	44.682	4.325	6.80e-06	3
21	1	1	44.6833	1.7665	6.81e-06	3
22	259	409	277.0364	409.0056	6.92e-06	2
23	10	100	42.3685	99.6822	7.11e-06	2
24	36	94	42.3726	94.0515	7.12e-06	1
25	10	10	42.4345	10.0039	7.25e-06	2
26	10	1	42.4411	1.0656	7.26e-06	2
27	50	10	52.1693	9.9959	1.10e-05	1
28	228	436	235.8406	436.0024	2.76e-05	1
29	226	171	235.9025	171.0031	3.06e-05	1
30	222	118	235.9149	118.0042	3.12e-05	2
					7.00e-06	65

Tabla IX: Resultados del algoritmo Simplex-Gradiente.

Corrida	k_p inicial	k_i inicial	k_p final	k_i final	min J	Iteraciones
1	905	980	2252.8	2413	4.92e-08	3
2	603	712	2246	2599.5	5.49e-08	3
3	259	409	1086.1	1660.7	1.02e-07	5
4	845	195	1269.4	285.4995	1.26e-07	2
5	312	924	734.3903	2137.4	1.49e-07	10
6	439	112	488.9846	127.4407	2.29e-07	5
7	431	185	488.9845	209.599	2.29e-07	5
8	226	171	488.0484	345.0617	2.30e-07	7
9	228	436	483.8964	906.4687	2.59e-07	6
10	471	231	474.2489	234.9975	2.86e-07	1
11	88	56	181.9734	81.9695	7.69e-07	8
12	222	118	548	258	1.12e-06	4
13	10	731	85.908	3374.9	1.80e-06	8
14	36	94	85.7741	165.8376	2.19e-06	4
15	5	10	84.2178	105.618	2.19e-06	7
16	21	31	84.2316	54.1739	2.19e-06	8
17	10	1	84.2392	16.4495	2.19e-06	5
18	5	6	85.7845	121.5855	2.19e-06	5
19	63	59	83.4912	61.3806	2.19e-06	6
20	86	3	83.596	4.0182	2.19e-06	5
21	65	38	85.8108	43.349	2.20e-06	6
22	10	10	85.8234	25.3973	2.20e-06	5
23	3	4	84.458	129.438	2.21e-06	5
24	1	1	83.8659	141.4232	2.21e-06	7
25	1	10	83.8968	105.4477	2.21e-06	7
26	10	100	88.0882	325.1197	2.22e-06	4
27	82	54	82	54	2.23e-06	1
28	100	10	86.4066	14.4551	2.30e-06	6
29	50	10	75.4702	15.9999	2.87e-06	4
30	595	263	897.054	387.496	3.64e-06	1
					1.50e-06	153

IV-A. Discusión

En la Tabla X se muestra una comparación entre los resultados de todos los algoritmos. En ella se incluye los mejores (Min J Best) y peores (Min J worst) resultados de las funciones objetivo, así como el promedio (Min J average) de las 30 corridas, adicionalmente se incluye el número total de iteraciones de las 30 corridas que necesitó cada algoritmo para llegar al punto que satisface el criterio de paro.

Como se puede observar el Algoritmo Simplex y la variante propuesta Simplex-Gradiente, son los que presentan los mejores resultados en cuanto al desempeño de la función objetivo, siendo la variante Simplex-Gradiente el que presenta los resultados más sobresalientes en cuanto a encontrar el mejor valor, tener el mejor promedio de todas las corridas, esto puede ser atribuido al amplio espacio de búsqueda con el que cuentan los algoritmos Simplex. Por otra parte, es importante mencionar que Powell y la variante Powell-Gradiente tienen un gran desempeño en cuanto al número de iteraciones necesarias, pues necesita menos de la mitad de iteraciones que las versiones Simplex, que son las segundas con el menor número de iteraciones.

Además, en las Fig. 4, 5, 6, 7, 8 y 9, se muestra una comparativa de la regulación de velocidad del motor sin escobillas. Todas comparan el mejor resultado obtenido de cada algoritmo y lo comparan contra el mejor de la variante Simplex-Gradiente. Se puede observar que en realidad no existe una gran diferencia en el desempeño obtenido entre cada resultado, esto es de esperarse debido a que el criterio de paro fue el llegar a un rango de valor aceptable en lugar del número de evaluaciones de la función objetivo. Esto se consideró para probar que todos métodos los determinísticos tienen la capacidad de llegar a un valor óptimo, si bien es cierto que en algunos les toma muchas más iteraciones.

Por otra parte, en cuanto al desempeño del control de velocidad se observa que los resultados generan un gran sobre impulso con muchas oscilaciones en el estado transitorio. Dichas oscilaciones además de ser generadas por la rápida respuesta del sistema (debido a que el intervalo de tiempo es pequeño) también son generadas por la forma en que se planteó el problema de optimización, ya que no se incluyeron restricciones para reducir el sobre-impulso, ni límite en cuanto a la potencia requerida.

Tabla X: Comparativa de resultados

Algoritmo	min J best	min J worst	min J average	Iteraciones Totales
Hooke-Jeeves	2.95e-07	2.30e-04	4.86e-05	939
Powell	2.93e-07	3.12e-05	7.13e-06	98
Simplex	5.52e-08	8.94e-06	2.15e-06	167
Gradiente	9.95e-07	1.31e-04	7.50e-05	7864
Hooke-Jeeves-Gradiente	2.95e-07	2.39e-04	5.23e-05	1483
Powell-Gradiente	2.93e-07	3.12e-05	7.00e-06	**65**
Simplex-Gradiente	**4.92e-08**	**3.64e-06**	**1.50e-06**	153

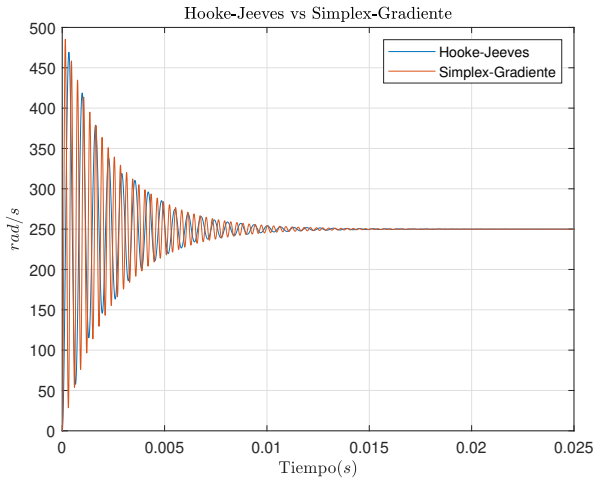

Figura 4: Hooke-Jeeves vs Simplex-Gradiente

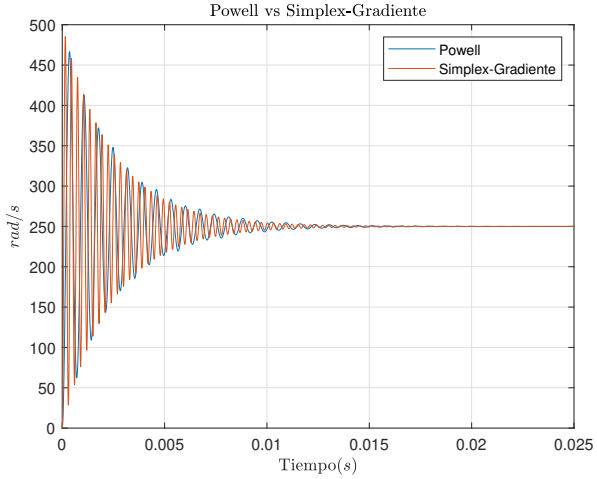

Figura 5: Powell vs Simplex-Gradiente

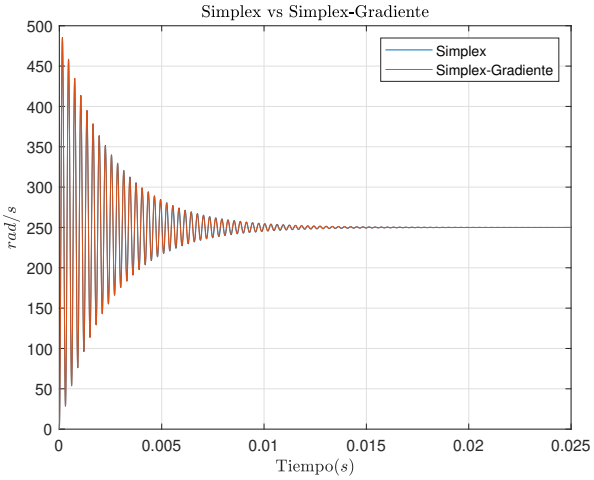

Figura 6: Simplex vs Simplex-Gradiente

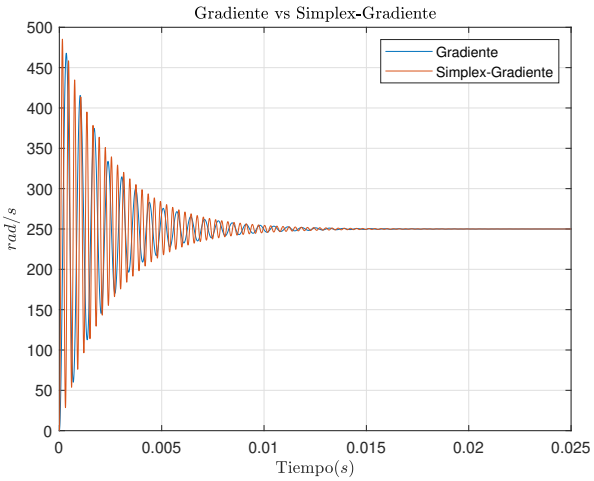

Figura 7: Gradiente vs Simplex-Gradiente

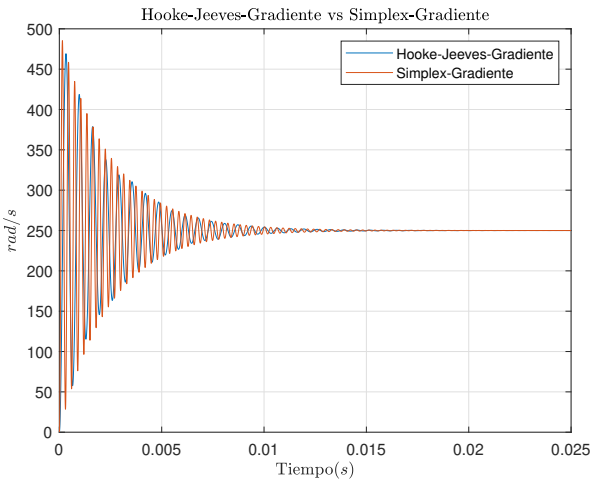

Figura 8: Hooke-Jeeves-Gradiente vs Simplex-Gradiente

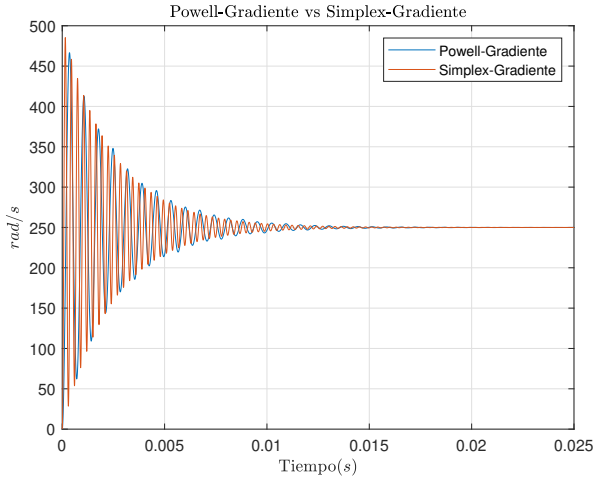

Figura 9: Powell-Gradiente vs Simplex-Gradiente

V. Conclusiones

En el presente trabajo se desarrolla la comparativa entre diferentes métodos deterministas. Cabe mencionar que para este particular problema se observa que la variable k_p tiene un gran impacto en los resultados ya que los resultados en mismos valores de k_p son similares ante diferentes valores de k_i. Se puede observar que los métodos de búsqueda directa (Tablas III, IV y V) dan mejores resultados que los de búsqueda indirecta (Tabla VI), aunque por otro lado el de búsqueda indirecta mantiene una similitud entre las corridas aproximándose a los mismos puntos k_p, esto puede estar asociado más a las características del problema de optimización que a los algoritmos, ya que se observa que el problema es multimodal, y diferentes soluciones cumplen con el criterio de paro de los algoritmos de búsqueda, esto se puede observar en las Fig. 4, 5, 6, 7, 8 y 9.

Por otro lado, en cuanto al desempeño los algoritmos propuestos por los autores de este trabajo (Tablas VII, VIII y IX) muestran que existe una reducción tanto en el número de iteraciones como en el promedio de los valores obtenidos en los resultados. Finalmente se observó que los mejores resultados fueron obtenidos por los algoritmos del tipo Simplex, esto es debido al espacio de búsqueda que plantea el algoritmo, lo que evita que caiga prematuramente en un mínimo local. Aunque cabe mencionar que los algoritmos del tipo Powell son los que menos iteraciones necesitan, necesitando aproximadamente una tercera parte de las iteraciones que necesitan los del tipo Simplex.

Referencias

[1] Lei Yang, Xiangyang Liu, Ningfei Wang, and Feng Wang. The structure analysis and design of a new self-optimizing fuzzy controller based on nelder-mead simplex method. In *2011 IEEE Power Engineering and Automation Conference*, volume 3, pages 136–139. IEEE, 2011.

[2] Lanhui Fu, Lei Zhou, Jianan Liang, and Mingyong Lin. Pid parameters self-tuning based on simplex method. In *2018 Chinese Control And Decision Conference (CCDC)*, pages 4769–4774. IEEE, 2018.

[3] Hee Do Kim, Byeong Seon Yu, Chae Rin Lee, and Seok-Kyoon Kim. Auto-tuning altitude controller with steepest gradient descent algorithm for quadrotors. In *2018 18th International Conference on Control, Automation and Systems (ICCAS)*, pages 867–871. IEEE, 2018.

[4] AH Habbi and M Zelmat. Fuzzy logic based gradient descent method with application to a pi-type fuzzy controller tuning: New results. In *2007 International*

Symposium on Computational Intelligence and Intelligent Informatics, pages 93–97. IEEE, 2007.

[5] Singiresu S Rao. *Engineering optimization: theory and practice*. John Wiley & Sons, 2009.

[6] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[7] Weiyao Lan and Qi Zhou. Speed control of dc motor using composite nonlinear feedback control. In *2009 IEEE International Conference on Control and Automation*, pages 2160–2164. IEEE, 2009.

[8] Nai Ho Ling and Yahaya Md Sam. Active torque control of electric power steering system using composite nonlinear feedback control. In *2015 IEEE Student Conference on Research and Development (SCOReD)*, pages 150–155. IEEE, 2015.

[9] Tung-Yung Huang and Huu Khoa Tran. Optimization of setpoint-based fuzzy gain scheduling and its application to a linear stage. In *2012 7th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 2106–2111. IEEE, 2012.

[10] Jonas Esch, Tim Könings, and Steven X Ding. Optimal performance tuning of a pi-controller for an integrator plant with uncertain parameters as a convex optimisation problem. In *52nd IEEE Conference on Decision and Control*, pages 1977–1982. IEEE, 2013.

[11] Deepti Singh, Nitin Singh, Brijesh Singh, and Surya Prakash. Optimal gain tuning of pi current controller with parameter uncertainty in dc motor drive for speed control. In *2013 Students Conference on Engineering and Systems (SCES)*, pages 1–6. IEEE, 2013.

[12] Zakiah Mohd Yusoff, Zuraida Muhammad, Mohd Noor Nasriq Nordin, Mohd Hezri Fazalul Rahiman, and Mohd Nasir Taib. Real time pid control for hydro-diffusion steam distillation essential oil extraction system using gradient descent tuning method. In *2012 IEEE Control and System Graduate Research Colloquium*, pages 288–293. IEEE, 2012.

[13] G Mallesham, S Mishra, and AN Jha. Optimization of control parameters in agc of microgrid using gradient descent method. In *16th national power systems conference*, pages 37–42, 2010.

[14] Alejandro Rodríguez-Molina, Efrén Mezura-Montes, Miguel G. Villarreal-Cervantes, and Mario Aldape-Pérez. Multi-objective meta-heuristic optimization in intelligent control: A survey on the controller tuning problem. *Applied Soft Computing*, 93:106342, 2020.

[15] Miguel G. Villarreal-Cervantes and Jaime Alvarez-Gallegos. Off-line pid control tuning for a planar parallel robot using de variants. *Expert Systems with Applications*, 64:444–454, 2016.

[16] Ramu Krishnan. *Electric motor drives: modeling, analysis, and control*, volume 626. Prentice Hall New Jersey, 2001.

[17] Ming-Fa Tsai, Tran Phu Quy, Bo-Feng Wu, and Chung-Shi Tseng. Model construction and verification of a bldc motor using matlab/simulink and fpga control. In *2011 6th IEEE Conference on Industrial Electronics and Applications*, pages 1797–1802. IEEE, 2011.

[18] Ashish Jethwani, Dhanraj Aseri, Thakur Sumeet Singh, and Amit Kumar Jain. A simpler approach to the modelling of permanent magnet brushless dc machine in matlab environment. In *2016 IEEE 6th International Conference on Power Systems (ICPS)*, pages 1–6. IEEE, 2016.

[19] Chang-liang Xia. *Permanent magnet brushless DC motor drives and controls*. John Wiley & Sons, 2012.

[20] Fangbo Xiao, Zhangyong Chen, Yong Chen, and Haifeng Liu. A finite control set model predictive direct speed controller for pmsm application with improved parameter robustness. *International Journal of Electrical Power & Energy Systems*, 143:108509, 2022.

[21] Ossama Ammari, Khalid El Majdoub, and Fouad Giri. Modeling and control of a half electric vehicle including an inverter, an in-wheel bldc motor and pacejka's tire model. *IFAC-PapersOnLine*, 55(12):604–609, 2022.

[22] Alam Gabriel Rojas-López, Miguel Gabriel Villarreal-Cervantes, Alejandro Rodríguez-Molina, and Consuelo Varinia García-Mendoza. Offline optimum tuning of the proportional integral controller for speed regulation of a bldc motor through bio-inspired algorithms. In *International Congress of Telematics and Computing*, pages 169–184. Springer, 2020.

[23] Maher GM Abdolrasol, MA Hannan, SM Suhail Hussain, and Taha Selim Ustun. Optimal pi controller based pso optimization for pv inverter using spwm techniques. *Energy Reports*, 8:1003–1011, 2022.

[24] Thomas Weise. Global optimization algorithms-theory and application. *Self-Published Thomas Weise*, 2009.

[25] Rajesh Kumar Arora. *Optimization: algorithms and applications*. Chapman and Hall/CRC, 2015.

[26] Arthur E. Bryson. *Dynamic Optimization*. Addison Wesley, jun 1999.

[27] Fangjin Xiong, Bowen Wei, and Fugang Xu. Identification of arch dam mechanical parameters based on sensitivity analysis and hooke–jeeves algorithm optimization. In *Structures*, volume 46, pages 88–98. Elsevier, 2022.

[28] Pu Zhang, Qi Sun, and Wei-Lin Xiao. Parameter identification in mixed brownian–fractional brownian motions using powell's optimization algorithm. *Economic Modelling*, 40:314–319, 2014.

[29] Song Pei Ye, Yi Hua Liu, Shun Chung Wang, and Hung Yu Pai. A novel global maximum power point tracking algorithm based on nelder-mead simplex technique for complex partial shading conditions. *Applied Energy*, 321:119380, 2022.

[30] Kalyanmoy Deb. *Optimization for engineering design: Algorithms and examples*. PHI Learning Pvt. Ltd., 2012.

[31] Kasim Hussain, Fudziah Ismail, and Norazak Senu. Solving directly special fourth-order ordinary differential equations using runge–kutta type method. *Journal of Computational and Applied Mathematics*, 306:179–199, 2016.