Controlling parameters proportional integral derivative of DC motor using a gradient-based optimizer

Widi Aribowo¹, Reza Rahmadian¹, Mahendra Widyartono¹, Ayusta Lukita Wardani¹, Aditya Prapanca², Laith Abualigah³,⁴,⁵,⁶,⁷

¹Department of Electrical Engineering, Faculty of Vocational, Universitas Negeri Surabaya, Surabaya, Indonesia
²Department of Computer Engineering, Faculty of Engineering, Universitas Negeri Surabaya, Surabaya, Indonesia
³Artificial Intelligence and Sensing Technologies (AIST) Research Center, University of Tabuk, Tabuk, Saudi Arabia
⁴Hourani Center for Applied Scientific Research, Al-Ahliyya Amman University, Amman, Jordan
⁵MEU Research Unit, Middle East University, Amman, Jordan
⁶Department of Electrical and Computer Engineering, Lebanese American University, Byblos, Lebanon
⁷Applied Science Research Center, Applied Science Private University, Amman, Jordan

Article Info

ABSTRACT

In this paper, a gradient-based optimizer (GBO) algorithm is presented to optimize the parameters of a proportional integral derivative (PID) controller in DC motor control. The GBO algorithm which mathematically models and mimics is inspired by the gradient-based Newton method. It was developed to address various optimization issues. To determine the performance of the proposed method, a comparison method with the ant colony optimization (ACO) method. It was compared using the integral of time multiplied absolute error (ITAE). They are most popularly used in the literature. From the test results, the proposed method is promising and has better effectiveness. The proposed method, namely GBO-PID, shows the best performance.

Keywords:
Artificial intelligence
DC motor
Gradient-based optimizer
Metaheuristic
Proportional integral derivative

This is an open access article under the CC BY-SA license.

1. INTRODUCTION

There are various sorts of control actions in a control system, including proportional, integral, and derivative control actions [1]–[4]. There are benefits to each of these control measures. Fast research is a benefit of proportional control action, minimizing errors is a benefit of integral control action, and lowering errors or overshoot/undershoot is a benefit of derivative control action [5].

The industry uses proportional integral derivative (PID) control extensively, which improves the system's transient and steady-state behavior [6]–[8]. To accomplish the conditions as per the anticipated setpoint, this control system processes computations based on the control variables Kp, Ki, and Kd. The DC motor rotational speed can be controlled by this control system to generate a satisfactory output response. However, in practice, when the setpoint changes, this PID control system has not been able to deliver a good output response in accordance with the intended circumstances [9]–[13].

Only linear conditions will allow a PID control system to function. DC motor convert electrical energy into mechanical energy [14]–[16]. A DC motor, however, exhibits a non-linearity effect. A single PID control system cannot generate an output response with the same characteristics under multiple setpoint values due to the variance in properties. A technique that can remove this non-linearity effect must be used in order to provide...
an output response with the same properties from various setpoints. A DC motor’s rotational speed can be managed using an adaptive PID control, which is one method of removing this non-linearity impact.

In recent years, several improving PID control methods using artificial intelligence have been presented, such as the neural network [17]–[20], henry gas solubility optimization algorithm [21], [22], transit search optimization algorithm [23], gray wolf optimization [24], salp swarm algorithm [25], slime mould algorithm [26], and particle swarm optimization [27]. This paper will present DC motor control using PID which is optimized using the gradient-based optimizer (GBO) algorithm. The GBO was introduced by Ahmadianfar et al. in 2020 [28]. The method was inspired by Newton’s gradient-based search method. To test the performance of the proposed method, this paper will make a comparison with the ant colony optimizer (ACO) method. The contributions of this research are: i) Application of the gradient-based optimizer (GBO) algorithm method to tune parameter PID as DC motor control and ii) Comparison of the GBO method with the ACO method applied to PID as DC motor control.

This paper is divided into some sections: i) Section 2, which is about the concept of DC motor and the gradient-based optimizer (GBO) method; ii) The section 3 is the results and discussion; and iii) The last section is to draw conclusions from the research.

2. METHODOLOGY

2.1. DC motor

DC motor is controlling by armature and field [29]. Stator and rotor are important parts of a DC motor. The non-rotating part of the DC motor is called the stator. While the rotating part is the rotor. DC motor with anchor control uses armature current as the controlling variable. Current coils or permanent magnets can generate a stator field. When a fixed field current pours in the field coil, the motor torque ($\tau_m$) shown as (1).

$$\tau_m(s) = (K_iK_f I_f)I_a(s) = K_m I_a(s)$$

(1)

If it is using permanent magnets, then shown as (2).

$$T_m(s) = K_m I_a(s)$$

(2)

Where, $K_m$ is the permeability function of the magnetic material. The relationship between the armature current ($I_a$) and the input voltage ($V_a$) in the armature circuit can be formulated as (3) and (4).

$$V_a(s) = (R_a + L_a s)I_a(s) + e_b(s)$$

(3)

$$e_b(s) = K_b \omega(s)$$

(4)

Where $R_a$ and $L_a$ are armature resistance and armature inductance. $e_b$ is back electromotive force. The torque in the motor is the same as the torque delivered to the load.

$$\tau_m(s) = \tau_L(s) + \tau_d(s)$$

(5)

The load torque for a rotating object is written as (6).

$$\tau_L(s) = J \omega(s) + B \omega(s)$$

(6)

Where $\tau_L$ is the torque connected to the load. $\tau_d$ is fault torque. $J$ and $B$ is inertia of the DC motor and damping friction ratio. Schematically of the DC motor are shown in Figure 1.

![Figure 1. DC motor block diagram](image)
2.2. A gradient-based optimizer (GBO)

The GBO method uses two main algorithms namely gradient tracing rules (GSR) and local escape operators (LEO) with a set of vectors to explore the search space. To increase exploration and convergence speed in finding the best position in the search space, GSR uses a gradient-based method. Meanwhile, according to Ahmadianfar [28] LEO is used to achieve local optimal.

In GBO, the amount of iterance and the population dimensions (\(\alpha\)) are based on the difficulty of the problem. Each member of the population is represented as a vector. Thus, the method adds a vector \(N\) in the D-dimension. The GBO method can be formulated as (7).

\[
X_{n,d} = [X_{n,1}, X_{n,2}, \ldots, X_{n,D}], n = 1, 2, \ldots, N, \ d = 1, 2, \ldots, D
\]  

In first stage, the vector was randomly selected in the prospecting zone. This could be formulated as (8).

\[
X_n = X_{\text{min}} + \text{rand} \ (0,1) \times (X_{\text{max}} - X_{\text{min}})
\]  

Where the limit of the decision variable is represented by \(X_{\text{min}}\) and \(X_{\text{max}}\).

2.2.1. Gradient search rule (GSR)

Vector displacement is controlled in an effort to find better searches in viable domains. Besides, to achieve a better position. This is done using the GSR method. The proposed method is applying the gradient based (GB) method in an effort to increase exploration and accelerate the convergence of GBO. The GB method initiates the initially estimated completion and shifts towards the next location along the direction detailed by the gradient. To derive the GSR, the first-order derivative is calculated using the Taylor series. The GSR method can be formulated as (9).

\[
GSR = \text{randn} \times \frac{2\Delta x \times x_n}{(x_{\text{worst}} - x_{\text{best}} + \varepsilon)}
\]  

Where random numbers that are normally distributed are represented as \(\text{randn}\). The small number in the range [0, 0.1] is \(\varepsilon\). The best solution is \(x_{\text{best}}\). \(x_{\text{worst}}\) is the worst solution.

The optimization method must maintain a balance motion to probe a hopeful area in the prospecting zone that leads to a globally best completion. In the GSR, the adaptive coefficient is used to equilibrium processes. This could be formulated as (10)-(20).

\[
\rho_1 = 2 \times \text{rand} \times \alpha \times -\alpha
\]

\[
\alpha = \left| \beta \times \sin\left(\frac{3\pi}{2} + \sin\left(\beta \times \frac{3\pi}{2}\right)\right) \right|
\]

\[
\beta = \beta_{\text{min}} + (\beta_{\text{max}} - \beta_{\text{min}}) \times \left(1 - \left(\frac{m}{M}\right)^2\right)^2
\]

The (9) changes to:

\[
GSR = \text{randn} \times \rho_1 \times \frac{2\Delta x \times x_n}{(x_{\text{worst}} - x_{\text{best}} + \varepsilon)}
\]

\[
\Delta x = \text{rand}(1:N) \times |\text{step}|
\]

\[
\text{step} = \frac{(x_{\text{best}} - x_{\text{min}}) + \delta}{2}
\]

\[
\delta = 2 \times \text{rand} \times \left(\frac{x_{\text{min}}^m + x_{\text{max}}^m + x_{\text{min}} + x_{\text{max}}}{4} - x_{\text{best}}^m\right)
\]

\[
x_{n+1} = x_n - GSR
\]

To make better use of the nearby area, a direction of movement (DM) parameter was added as (18) and (19).

\[
DM = \text{rand} \times \rho_2 \times (x_{\text{best}} - x_n)
\]

\[
\rho_2 = 2 \times \text{rand} \times \alpha \times -\alpha
\]
Where the random number in [0, 1] is denoted rand. The random variable helps each vector have a diverse pace measure is represented by ρ2. The current vector position in (20) can be updated based on GSR and DM.

\[ X_{1n}^m = x_{1n}^m - GSR + DM \]  

(20)

2.2.2. Local escaping operator (LEO)

LEO is enabled to boost the performance of method in breaking complicated issues. The (21) can find a significant solution position.

\[ \text{if } \text{rand} < 0.5 \]
\[ X_{leo} = X_{n+1} + f_1(u_1 \times X_{best} - u_2 \times X_k^m) + f_2 \times p_1 \times \left( u_3 \times (X_{2n}^m - X_{1n}^m) + u_2 \times (x_{1n}^m - x_{2n}^m) \right)/2 \]  

(21)

Else, as (22)-(25) show.

\[ X_{leo} = X_{best} + f_1(u \times X_{best} - u_2 \times X_k^m) + f_2 \times p_1 \times \left( u_3 \times (X_{2n}^m - X_{1n}^m) + u_2 \times (x_{1n}^m - x_{2n}^m) \right)/2 \]  

(22)

\[ u_1 = \begin{cases} 2 \times \text{rand}, & \text{if } \mu_1 < 0.5 \\ 1, & \text{otherwise} \end{cases} \]

(23)

\[ u_2 = \begin{cases} \text{rand}, & \text{if } \mu_1 < 0.5 \\ 1, & \text{otherwise} \end{cases} \]

(24)

\[ u_3 = \begin{cases} \text{rand}, & \text{if } \mu_1 < 0.5 \\ 1, & \text{otherwise} \end{cases} \]

(25)

2.3. Proposed GBO for controlling DC motor speed

To increase the reaction of the DC motor in the detailed point as overshoot, rise-time, and settling time, the PID controller parameter values are searched using the proposed method, namely the GBO algorithm. Figure 2 is a block diagram illustration of the proposed method with the GBO-PID for the DC motor. GBO gets input from ITAE calculations which are always updated during the iteration process. The output obtained is the PID parameter

Figure 2. Proposed method diagram

3. RESULTS AND DISCUSSION

The programming code required for the GBO algorithm and simulations is performed using the MATLAB/Simulink. The laptop is used with an AMD A9 (3.10 GHz) and ram 4 GB. The variable of the GBO and the values can be seen in Table 1.

To see the effectiveness and advantages of the proposed GBO-PID approach, the GBO-PID controller was compared with ACO-PID. The convergence curve can be seen in Figure 3. DC motor controlled by PID optimized using GBO has the lowest integral of time multiplied absolute error (ITAE) value. In addition, the GBO-PID control has the least number of iterations, which is under five iterations.
The DC motor speed step response for the GBO-PID and ACO-PID controllers is shown in Figure 4. Details regarding the step respond of GBO-PID and ACO-PID can be seen in Table 2. The proposed GBO-PID has the best reaction step because it has the fastest constancy. The performance index used as a comparison is ITAE. ITAE has been widely used in several studies. The mathematical formula of the ITAE index is as (26).

\[ ITAE = \int_0^t t \cdot e(t) \, dt \]  

(26)

Table 1. Parameter of GBO

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of populations</td>
<td>50</td>
</tr>
<tr>
<td>Maximum number of iterations</td>
<td>50</td>
</tr>
<tr>
<td>Probability parameter</td>
<td>0.5</td>
</tr>
<tr>
<td>Lower bound</td>
<td>0</td>
</tr>
<tr>
<td>Upper bound</td>
<td>10</td>
</tr>
<tr>
<td>Dim</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 2. Comparison of transient result

<table>
<thead>
<tr>
<th>Controller</th>
<th>Overshoot</th>
<th>Rise time</th>
<th>Setting time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACO-PID</td>
<td>1.03245</td>
<td>1.917</td>
<td>3.012</td>
</tr>
<tr>
<td>GBO-PID</td>
<td>1.03201</td>
<td>1.777</td>
<td>2.829</td>
</tr>
</tbody>
</table>

Table 3. Comparison of ITAE result

<table>
<thead>
<tr>
<th>Controller</th>
<th>ITAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACO-PID</td>
<td>0.0329</td>
</tr>
<tr>
<td>GBO-PID</td>
<td>0.0292</td>
</tr>
</tbody>
</table>

Table 4. Detail of test condition

<table>
<thead>
<tr>
<th>Test number</th>
<th>Ra</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.03</td>
<td>0.005</td>
</tr>
<tr>
<td>2</td>
<td>0.012</td>
<td>0.005</td>
</tr>
<tr>
<td>3</td>
<td>0.03</td>
<td>0.009</td>
</tr>
</tbody>
</table>

Table 5. Comparison of results for test 1

<table>
<thead>
<tr>
<th>Controller</th>
<th>Rise time</th>
<th>Setting time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACO-PID</td>
<td>3.303</td>
<td>4.6229</td>
</tr>
<tr>
<td>GBO-PID</td>
<td>3.202</td>
<td>4.594</td>
</tr>
</tbody>
</table>
Controlling parameters proportional integral derivative of DC motor using a ... (Widi Aribowo)


Controlling parameters proportional integral derivative of DC motor using a … (Widi Aribowo)