JITEL (Jurnal Ilmiah Telekomunikasi, Elektronika, dan Listrik Tenaga) p-ISSN: 2774-7972
Vol. 5, No. 1, Maret 2025, pp. 57-66 e-ISSN: 2775-6696
DOI: https://doi.org/10.35313/jitel.v5.i1.2025.57-66

Optimasi kendali PID kecepatan motor DC dengan metode *tuning particle swarm optimization*

Fitria Suryatini[1*], Adhitya Sumardi Sunarya[2], Qisthon Khairul 'Amilin[3]
[1,2,3]Jurusan Teknik Otomasi Manufaktur dan Mekatronika, Politeknik Manufaktur Bandung
Jl. Kanayakan No. 21, Dago, Kota Bandung, 40135 Jawa Barat, Indonesia
[1*]fitria@ae.polman-bandung.ac.id, [2]adhitya@ae.polman-bandung.ac.id, [3]qisthonkha@gmail.com

ABSTRAK

Motor DC merupakan aktuator elekromagnetik yang mengubah energi listrik menjadi energi mekanik dan banyak digunakan dalam industri. Meskipun kecepatan putarnya dapat dikendalikan dengan mudah, performa sistem harus tetap dijaga agar optimal. *Proportional-Integral-Derivative* (PID) merupakan salah satu metode yang sering digunakan dalam pengendalian motor DC. Namun, tantangan utama dalam metode PID adalah *tuning* (penyetelan) parameter, yaitu menentukan nilai Kp, Ki, dan Kd agar menghasilkan performa yang optimal. Dalam penelitian ini, penyetelan parameter PID dilakukan menggunakan algoritma *Particle Swarm Optimization* (PSO) untuk memperoleh kendali kecepatan motor DC yang lebih optimal. Selain itu, penelitian ini juga membandingkan performa *tuning* PSO dengan *tuning* Ziegler-Nichols 2 (ZN 2). Sistem ini menggunakan LabVIEW sebagai perangkat lunak antarmuka serta NI ELVIS II sebagai perangkat keras pengontrol dan akuisisi data. Metode kendali yang digunakan berupa sistem kendali umpan balik dengan keluaran berupa kecepatan motor DC. Hasil penelitian menunjukkan bahwa sistem kendali kecepatan motor DC dengan metode *tuning* PSO menghasilkan *rise time* 5 detik dan *settling time* 7 detik. Metode *tuning* ZN 2 menghasilkan *rise time* 8 detik, dan *settling time* 11 detik. Sementara itu, metode *tuning trial and error* menghasilkan *rise time* 7 detik dan *settling time* 12 detik. Hasil ini menunjukkan bahwa metode *tuning* PSO mampu meningkatkan performa kendali kecepatan motor DC dibandingkan metode ZN 2 dan *trial and error*, dengan respons waktu yang lebih baik.

Kata kunci: motor DC, kendali PID, PSO, Ziegler-Nichols, LabVIEW

ABSTRACT

A DC motor is an electromagnetic actuator that converts electrical energy into mechanical energy and is widely used in the industry. Although its rotational speed can be easily controlled, the system's performance must be maintained for optimal operation. Proportional-Integral-Derivative (PID) is one of the most commonly used methods for DC motor control. However, the main challenge in the PID method is tuning (adjusting) the parameters, which involves determining the values of Kp, Ki, and Kd to achieve optimal performance. In this study, PID parameter tuning was performed using the Particle Swarm Optimization (PSO) algorithm to achieve more optimal DC motor speed control. Additionally, this study compares the performance of PSO tuning with Ziegler-Nichols 2 (ZN 2) tuning. The system uses LabVIEW as the interface software and NI ELVIS II as the hardware controller and data acquisition device. The control method implemented is a feedback control system, with the output being the DC motor speed. The research results show that the PSO tuning method produces a rise time of 5 seconds and a settling time of 7 seconds. The ZN 2 tuning method results in a rise time of 8 seconds and a settling time of 11 seconds, while the trial-and-error tuning method results in a rise time of 7 seconds and a settling time of 12 seconds. These results indicate that the PSO tuning method improves DC motor speed control performance compared to the ZN 2 and trial-and-error methods, providing a better response time.

Keywords: *DC motor, PID control, PSO, Ziegler-Nichols, LabVIEW*

1. PENDAHULUAN

Motor DC merupakan aktuator elektromagnetik yang mengubah energi listrik menjadi energi mekanik dan banyak digunakan dalam berbagai aplikasi industri, seperti otomasi, robotika, dan kendaraan listrik [1]. Motor DC memiliki keunggulan berupa kemudahan dalam pengendalian kecepatan putarnya, namun performanya tetap harus dijaga agar optimal [2]. Meskipun memiliki respons yang cepat dan presisi tinggi, motor DC juga memiliki kelemahan, salah satunya adalah *error steady-state* yang masih dapat terjadi [3]. Selain itu, tantangan lain dalam pengoperasiannya adalah memastikan

Naskah diterima tanggal 6 Maret 2024, disetujui tanggal 3 Maret 2025
*E-mail korespondensi

stabilitas kecepatan putarannya, terutama saat terjadi perubahan beban atau gangguan eksternal. Oleh karena itu, diperlukan sistem kendali yang mampu menjaga kecepatan motor DC tetap optimal dalam berbagai kondisi operasional guna meningkatkan keandalan sistem.

Salah satu metode kendali yang paling umum digunakan dalam pengaturan kecepatan motor DC adalah kendali PID. Kendali PID adalah metode yang menggunakan *feedback* dari keluaran dan membandingkannya dengan referensi yang diinginkan, kemudian menghitung *error* dan mengoreksi kesalahan tersebut melalui aksi kontrol berbasis tiga parameter utama: konstanta proportional (Kp), konstanta integral (Ki), dan konstanta derivatif (Kd) [4]. Namun, tantangan terbesar dalam penerapan PID adalah *tuning* parameter [5], [6], yaitu menentukan nilai Kp, Ki, dan Kd yang optimal untuk mencapai respons sistem yang cepat, stabil, dan memiliki *error* minimum [7].

Salah satu metode *tuning* parameter PID yang sering digunakan adalah metode *tuning* konvensional *trial and error* [8], tetapi metode ini memiliki kesulitan dalam penyesuaian parameter, sehingga proses pencarian parameter memakan waktu lama, akurasi kendali kurang optimal, dan parameter yang digunakan belum mencapai hasil terbaik [7]. Metode konvensional lainnya yang banyak digunakan adalah metode Ziegler-Nichols (ZN) [9]. Metode ZN merupakan salah satu metode *tuning* klasik yang banyak digunakan karena prosedurnya yang sederhana, tetapi sering kali menghasilkan sistem dengan *overshoot* besar dan respons transien yang kurang optimal. Dalam beberapa tahun terakhir, para peneliti telah mengembangkan berbagai metode untuk *tuning* parameter PID baik menggunakan metode konvensional dan metode cerdas, di antaranya *Cohen-Coon* [10], *Differential Evolution* [11], *Genetic Algortihm* [12], *Crow Search* [13], *Coronavirus Optimization* [14], *Grey Wolf Optimizer* [15], *Ant Colony* [16], dan *Bat Algorithm* [17].

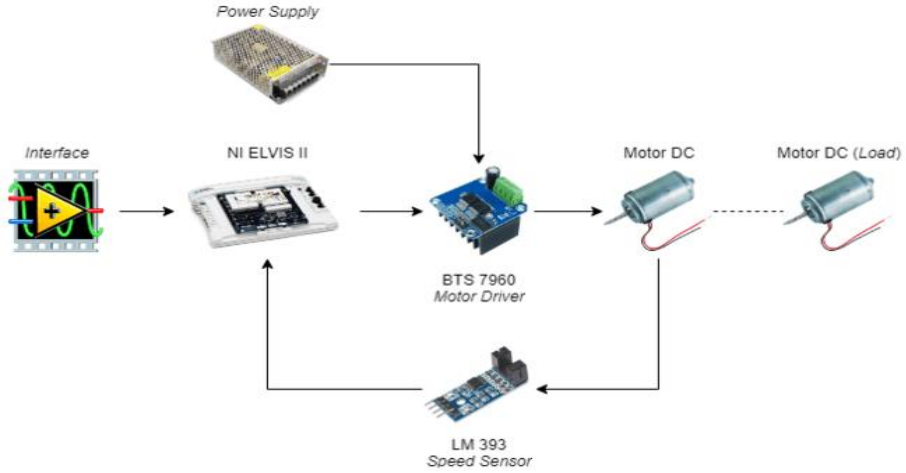
Penelitian ini bertujuan untuk mengimplementasikan dan mengevaluasi *tuning* PID berbasis *Particle Swarm Optimization* (PSO) pada sistem kendali kecepatan motor DC [7], [18]. Beberapa referensi penelitian menyatakan bahwa optimasi parameter menggunakan metode PSO memiliki hasil yang stabil dibandingkan dengan metode lainnya [7], [18], [19], [20]. Beberapa metode kendali untuk pengendalian kecepatan motor DC yang diusulkan dalam penelitian sebelumnya [1], [13], [21] hanya diterapkan pada sistem simulasi, sehingga masih diperlukan validasi lebih lanjut. Penelitian ini menerapkan metode kendali pada perangkat keras. Penelitian [7] juga mengimplementasikan PSO-PID pada perangkat keras. Perbedaannya, dalam penelitian ini, kendali PID diimplementasikan menggunakan NI ELVIS II sebagai kontroler dan LabVIEW sebagai antarmuka. Evaluasi dilakukan dengan mengukur beberapa parameter performa sistem, seperti *rise time*, *settling time*, dan *overshoot* untuk menilai efektivitas metode *tuning* yang digunakan, serta membandingkan performanya dengan metode ZN 2 dan metode *trial and error*. Dengan demikian, penelitian ini berkontribusi pada *tuning* parameter PID melalui perbandingan beberapa metode dan penerapan metode kontrol pada sistem perangkat keras.

2. METODE PENELITIAN

Dalam penelitian ini, kendali PID digunakan untuk mengontrol kecepatan motor DC dengan parameter Kp, Ki, dan Kd yang ditentukan menggunakan metode *tuning* PSO. Metode ini bekerja dengan mencari kombinasi parameter PID yang optimal berdasarkan kriteria performa seperti *error steady-state*, *overshoot*, *rise time*, dan *settling time*.

2.1 Perancangan Perangkat Keras

Penelitian ini mengimplementasikan kendali kecepatan motor DC menggunakan NI ELVIS II. Sistem terdiri dari *driver* motor BTS7960, sensor kecepatan LM393, serta dua motor DC, salah satunya sebagai beban. Diagram blok perangkat keras ditunjukkan pada Gambar 1. NI ELVIS II berfungsi sebagai pengendali utama, menerima *setpoint* dari LabVIEW untuk mengontrol kecepatan motor. *Power supply* menyediakan daya untuk seluruh sistem. *Driver* motor BTS7960 mengontrol motor DC dengan menerima sinyal PWM, yang menentukan kecepatan dan arah putaran motor. Sensor kecepatan LM393 terdiri dari LED inframerah dan foto transistor NPN, dengan IC komparator LM393 untuk menghasilkan sinyal HIGH atau LOW berdasarkan adanya objek. Sinyal ini dihitung sebagai frekuensi perubahan, dikonversi ke RPM, lalu dikirim ke NI ELVIS II sebagai umpan balik. Pengendali PID menganalisis dan mengoreksi sinyal untuk menjaga kestabilan kecepatan motor.



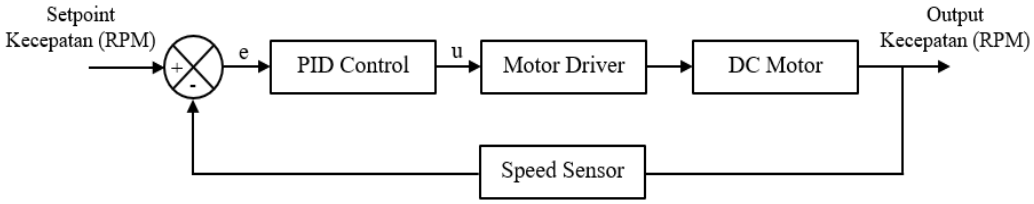
Gambar 1. Arsitektur sistem

2.2 Perancangan Kendali PID pada Kendali Kecepatan Motor DC

Sistem kendali umpan balik (*feedback control*) dengan PID adalah metode kendali yang digunakan dalam pengendalian kecepatan motor DC pada penelitian ini. Pengendali PID banyak diterapkan dalam industri karena menawarkan tingkat presisi yang tinggi, kemudahan dalam penerapan, dan keandalan. Proses pengendalian dilakukan dengan mengurangi sinyal kesalahan saat sistem beroperasi, serta mampu menghasilkan keluaran sinyal dengan respons yang cepat, kesalahan yang minimal, dan mengurangi kemungkinan terjadinya *overshoot* [22]. Keluaran sinyal kendali PID dirumusukan pada persamaan (1).

$$U_t = K_p e(t) + K_i \int_0^t e(t) + K_d \frac{de(t)}{dt} \quad (1)$$

dimana Kp, Ki, dan Kd masing-masing merupakan konstanta proporsional, integral, dan derivatif; dan *e* adalah sinyal *error*. Parameter proporsional berfungsi untuk mempercepat respons sistem dan mengurangi *error steady-state*, sedangkan parameter integral berperan dalam menghilangkan *error steady-state* secara keseluruhan. Parameter diferensial bertujuan untuk mengurangi tingkat *overshoot* [23]. Diagram blok sistem kendali PID pada penelitian ini ditunjukkan pada Gambar 2.



Gambar 2. Diagram blok sistem kendali kecepatan motor DC

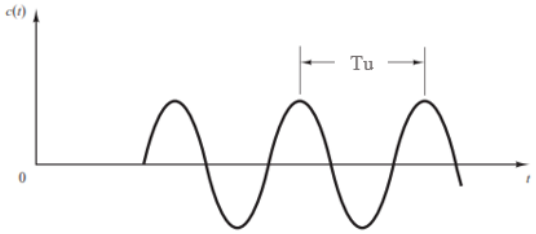
Gambar 2 merupakan diagram blok sistem kendali PID yang dirancang untuk mengatur kecepatan motor DC. Sistem ini bekerja dengan menerima *setpoint* kecepatan dalam satuan RPM sebagai referensi kecepatan yang diinginkan. Nilai *setpoint* kemudian dibandingkan dengan kecepatan aktual yang diukur oleh sensor kecepatan, menghasilkan sinyal *error* (e) yang menunjukkan selisih antara kecepatan yang diinginkan dan kecepatan aktual. Sinyal *error* ini dikirim ke pengendali PID, yang kemudian menghitung sinyal kontrol (u) berdasarkan aksi proporsional (P), integral (I), dan derivatif (D) untuk mengoreksi kecepatan motor. Sinyal kontrol ini diteruskan ke *motor driver*, yang berfungsi mengubah sinyal tersebut untuk menggerakkan motor DC. Motor kemudian berputar sesuai dengan perintah yang diberikan. Kecepatan putaran motor diukur oleh sensor kecepatan, dan hasil pengukuran ini dikirim kembali sebagai umpan balik ke sistem untuk dibandingkan dengan *setpoint*. Proses ini berlangsung

secara terus-menerus dalam sistem *loop* tertutup, memastikan bahwa kecepatan motor selalu sesuai dengan nilai yang diinginkan, dengan respons yang cepat, *error* minimal, dan *overshoot* yang kecil.

2.3 Perancangan *Tuning* Parameter PID dengan metode Ziegler-Nichols 2

Metode Ziegler-Nichols merupakan salah satu teknik *tuning* PID yang banyak digunakan dalam dunia industri untuk mendapatkan parameter Kp, Ki, dan Kd yang optimal. Metode Ziegler-Nichols 2, atau yang sering disebut metode respons frekuensi, digunakan ketika karakteristik sistem lebih mudah diidentifikasi melalui pendekatan osilasi. Pada metode ini sistem disusun pada *loop* tertutup. Berikut langkah-langkah *tuning* PID menggunakan metode ZN 2 [9], [24], [25]:

1. Matikan aksi integral (Ki) dan derivatif (Kd), sehingga hanya kontrol P (proporsional) yang aktif.
2. Tingkatkan nilai Kp secara bertahap hingga sistem mencapai osilasi stabil berkelanjutan dengan periode tertentu (Tu), seperti yang ditunjukkan pada Gambar 3. Titik ini disebut sebagai batas kestabilan kritis.
3. Catat nilai Kp kritis (Ku), yang menyebabkan sistem mencapai osilasi stabil berkelanjutan) dan periode osilasi kritis (Tu).
4. Gunakan tabel parameter Ziegler-Nichols untuk menentukan nilai Kp, Ki, dan Kd sesuai dengan jenis pengendali yang digunakan seperti pada Tabel 1.



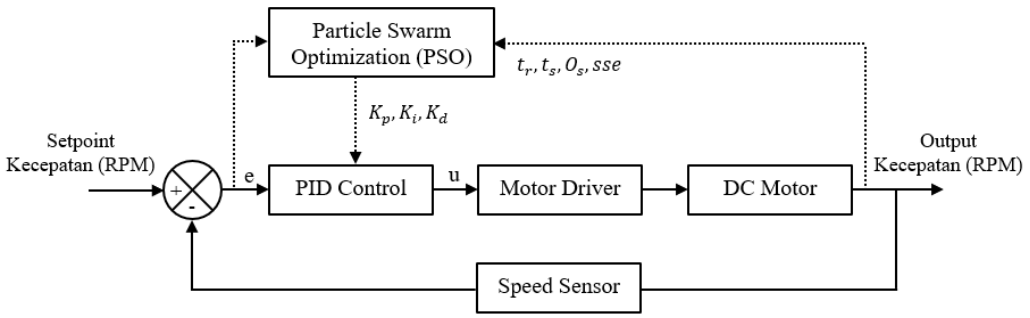
Gambar 3. Osilasi stabil berkelanjutan dengan periode Tu [24]

Tabel 1. Tabel perhitungan ZN 2 [9]

Pengendali	Kp	Ki	Kd
P	Ku/2	-	-
PI	2Ku/5	4Tu/5	-
PID	3Ku/5	Tu/2	3Tu/25

2.4 Perancangan *Tuning* Parameter PID dengan PSO

PSO adalah salah satu algoritma optimasi berbasis populasi yang terinspirasi dari perilaku kawanan burung atau sekumpulan ikan dalam mencari makanan. Metode ini dikembangkan oleh Kennedy dan Eberhart pada tahun 1995 dan sering digunakan dalam *tuning* parameter PID untuk mendapatkan performa sistem kendali yang lebih optimal dibandingkan metode konvensional seperti Ziegler-Nichols.



Gambar 4. Kendali PID menggunakan metode PSO

PSO bekerja dengan membuat sekelompok solusi kandidat (partikel) yang bergerak dalam ruang pencarian untuk menemukan solusi optimal. Dalam konteks *tuning* PID, Kp, Ki, dan Kd dianggap sebagai parameter yang harus dioptimalkan agar sistem memiliki performa terbaik. Perancangan kendali PSO-PID ditunjukkan pada Gambar 4. PSO digunakan untuk mencari nilai parameter PID terbaik dengan mempertimbangkan beberapa kriteria, seperti t_r(*rise time*), t_s(*settling time*), O_s(*overshoot*), dan sse (*steady state error*). Langkah-langkah PSO untuk *tuning* PID [26], [27], [28]:

1. Inisialisasi Populasi
   a. Tentukan jumlah partikel (solusi awal).
   b. Set nilai acak awal untuk parameter Kp, Ki, dan Kd dalam batas tertentu.
2. Evaluasi Fungsi Objektif
   a. Setiap partikel diuji menggunakan fungsi objektif (misalnya *Integral of Time-weighted Absolute Error* (ITAE), *Integral of Square Error* (ISE), atau *Integral of Absolute Error* (IAE)).
   b. Fungsi objektif ini mengukur performa sistem kendali berdasarkan respons waktu transien, *error steady-state*, dan *overshoot*.
3. Perbarui Kecepatan dan Posisi Partikel
   a. Setiap partikel diperbarui berdasarkan:
      1) *Best personal position* (*pbest*): Posisi terbaik yang pernah dicapai oleh partikel itu sendiri.
      2) *Best global position* (*gbest*): Posisi terbaik dari seluruh partikel dalam populasi.
   b. Rumus pembaruan kecepatan partikel:

$$v_{i,j}^{t+1} = w.v_{i,j}^{t} + c_1.r_1\left(pBest_{i,j}^{t} - x_{i,j}^{t}\right) + c_2.r_2(gBest_{i,j}^{t} - x_{i,j}^{t}) \quad (2)$$

   di mana:
   w = faktor inersia (mengontrol keseimbangan eksplorasi dan eksploitasi).
   c_1, c_2 = koefisien percepatan untuk *pbest* dan *gbest*.
   r_1, r_2 = bilangan acak antara 0 dan 1.
   c. Posisi partikel diperbarui dengan:

$$x_{i,j}^{t+1} = x_{i,j}^{t} + v_{i,j}^{t} \quad (3)$$

4. Iterasi Sampai Konvergensi
   Proses ini diulang hingga mencapai kriteria penghentian, seperti jumlah iterasi maksimum atau *error* yang sudah minimal.
5. Mendapatkan Parameter PID Optimal
   Setelah iterasi selesai, nilai Kp, Ki, dan Kd terbaik diambil dari gbest dan digunakan dalam sistem kendali PID.

2.5 Perancangan Antarmuka

Perancangan antarmuka pengguna (*user interface*) menggunakan LabVIEW dalam penelitian ini bertujuan untuk memudahkan interaksi antara pengguna dan sistem kendali motor DC serta untuk melihat hasil respons sistem. Antarmuka ini akan menyediakan dua menu utama: Menu manual/PID dan Menu PID-PSO. Berikut adalah penjelasan mengenai masing-masing menu dan fungsionalitasnya.

2.5.1 Menu Manual/PID

Menu ini dirancang untuk memungkinkan pengguna melakukan pengaturan kecepatan motor DC dengan menginputkan nilai parameter Kp, Ki, dan Kd secara manual dari hasil perhitungan *tuning* menggunakan metode Ziegler-Nichols 2 dan hasil *tuning Particle Swarm Optimization*, maupun berdasarkan *trial and error*. Tampilan antarmuka menu manual/PID terdapat pada Gambar 6a.

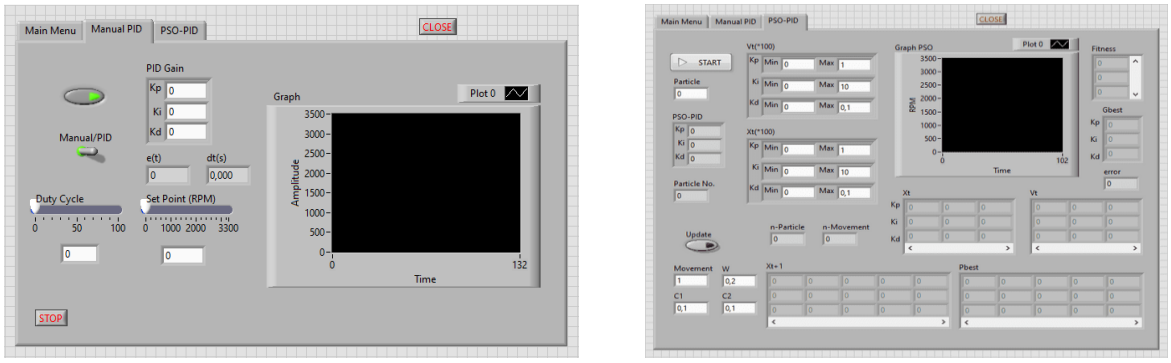
Pada antarmuka terdapat *switch* untuk memilih manual/PID. Manual artinya motor dijalankan secara manual dengan mengatur *duty cycle* tanpa pengendali PID. Menu manual ini bertujuan untuk menguji apakah motor berfungsi atau tidak. Pada menu PID, pengguna dapat memasukkan nilai *setpoint* kecepatan yang diinginkan untuk motor DC. Nilai ini akan menjadi target yang ingin dicapai oleh sistem kendali. Pada menu PID *gain* terdapat kontrol untuk mengatur parameter PID (Kp, Ki, Kd) secara manual. Pengguna dapat mengubah nilai-nilai ini berdasarkan hasil *tuning* Ziegler-Nichols, PSO, maupun berdasarkan *trial and error*. Antarmuka dilengkapi dengan grafik yang menunjukkan kecepatan aktual motor DC dibandingkan dengan *setpoint*. Hal ini memungkinkan pengguna untuk memantau kinerja sistem secara *real-time*.

2.5.2 Menu PID-PSO

Menu ini dirancang untuk mengoptimalkan kendali PID menggunakan metode PSO. Pengguna dapat memulai proses *tuning* PSO dengan menekan tombol yang sesuai. Proses ini akan mengoptimalkan parameter PID (Kp, Ki, Kd) secara otomatis berdasarkan algoritma PSO. Pada antarmuka terdapat fitur untuk mengatur parameter PSO, seperti jumlah partikel, iterasi maksimum, dan

parameter kecepatan partikel. Ini memberikan fleksibilitas kepada pengguna untuk menyesuaikan proses optimasi. Tampilan antarmuka menu PID-PSO terdapat pada Gambar 5.

Antarmuka pada penelitian ini dilengkapi juga dengan visualisasi proses PSO berupa grafik yang menunjukkan evolusi nilai parameter PID selama proses *tuning* PSO. Sehingga dapat memberikan wawasan kepada pengguna tentang bagaimana algoritma PSO bekerja dalam mencari solusi optimal. Setelah proses *tuning* selesai, nilai parameter PID yang dioptimalkan akan ditampilkan di antarmuka, yang dapat langsung diterapkan untuk kendali kendali kecepatan motor DC pada menu manual/PID.



(a) **(b)**

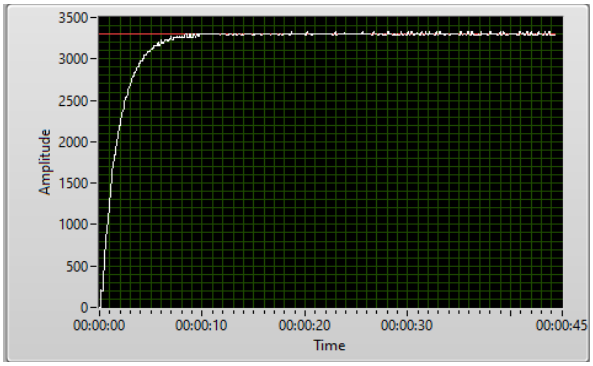
Gambar 5. Tampilan antarmuka (a) Manual/PID, dan (b) PID-PSO

3. HASIL DAN PEMBAHASAN

Pengujian pada sistem ini dilakukan untuk mengevaluasi hasil perancangan *tuning* parameter PID pada kendali kecepatan motor DC yang diimplementasikan menggunakan NI ELVIS II dan LabVIEW. Proses pengujian mencakup pengujian kendali PID menggunakan metode *trial and error*, ZN 2, dan PSO, serta membandingkan hasilnya untuk mencari parameter kendali yang optimal.

3.1 Hasil Kendali PID Menggunakan Metode *Trial and Error*

Metode *trial and error* dilakukan dengan menyesuaikan parameter Kp, Ki, dan Kd secara manual hingga diperoleh respons sistem yang dianggap optimal. Pada metode *trial and error* didapatkan nilai Kp = 0,01, Ki = 0,01, dan Kd 0,001. Nilai parameter PID yang telah ditentukan, dimasukkan ke dalam antarmuka LabVIEW dan terhubung ke NI ELVIS II dan motor DC. Setpoint ditetapkan sebesar 3300 RPM. Grafik respons sistem terhadap setpoint ditunjukkan pada Gambar 6.



Gambar 6. Respons PID *trial and error*

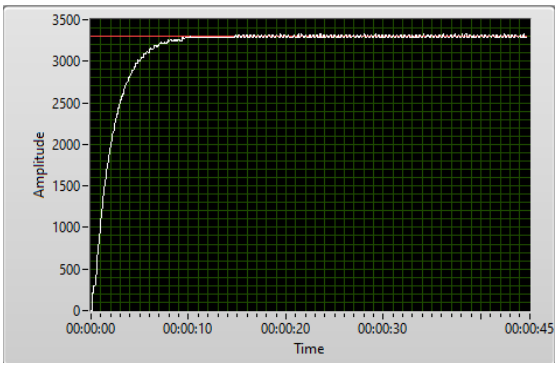
Respons sistem PID menggunakan metode *trial and error* menghasilkan *rise time* sebesar 7 detik, *settling time* 12 detik, dan tidak ada *overshoot*. Nilai parameter hasil respons PID ditunjukkan pada Tabel 2.

Tabel 2. Respons PID *trial and error*

Parameter PID			*Rise Time* (s)	*Settling Time* (s)	*Overshoot*
Kp	Ki	Kd			
0,01	0,01	0,001	7	12	0

3.2 Hasil Kendali PID Menggunakan Metode *ZN 2*

Langkah pertama yang dilakukan dalam metode ZN 2 adalah mencari nilai Kp hingga respons berosilasi stabil berkelanjutan, yang disebut dengan Ku. Selanjutnya menentukan nilai Tu yaitu nilai periode osilasi tersebut. Kemudian masukkan nilai Tu dan Ku ke dalam tabel 1. Dari hasil percobaan didapatkan parameter Kp = 0,022, Ki = 0,0075, dan Kd = 0,0018. Nilai parameter PID yang telah didapatkan menggunakan metode ZN 2, kemudian dimasukkan ke dalam antarmuka LabVIEW yang terhubung ke NI ELVIS II dan Motor DC. Setpoint ditetapkan sebesar 3300 RPM. Grafik respons sistem terhadap setpoint ditunjukkan pada Gambar 7.



Gambar 7. Respons PID ZN 2

Respons sistem PID menggunakan metode ZN 2 menunjukan *rise time* sebesar 8 detik, *settling time* 11 detik, dan tidak ada *overshoot*. Nilai parameter hasil respons PID ditunjukkan pada Tabel 3.

Tabel 3. Respons PID ZN 2

Parameter PID			*Rise Time* (s)	*Settling Time* (s)	*Overshoot*
Kp	Ki	Kd			
0,0022	0,0075	0,0018	8	11	0

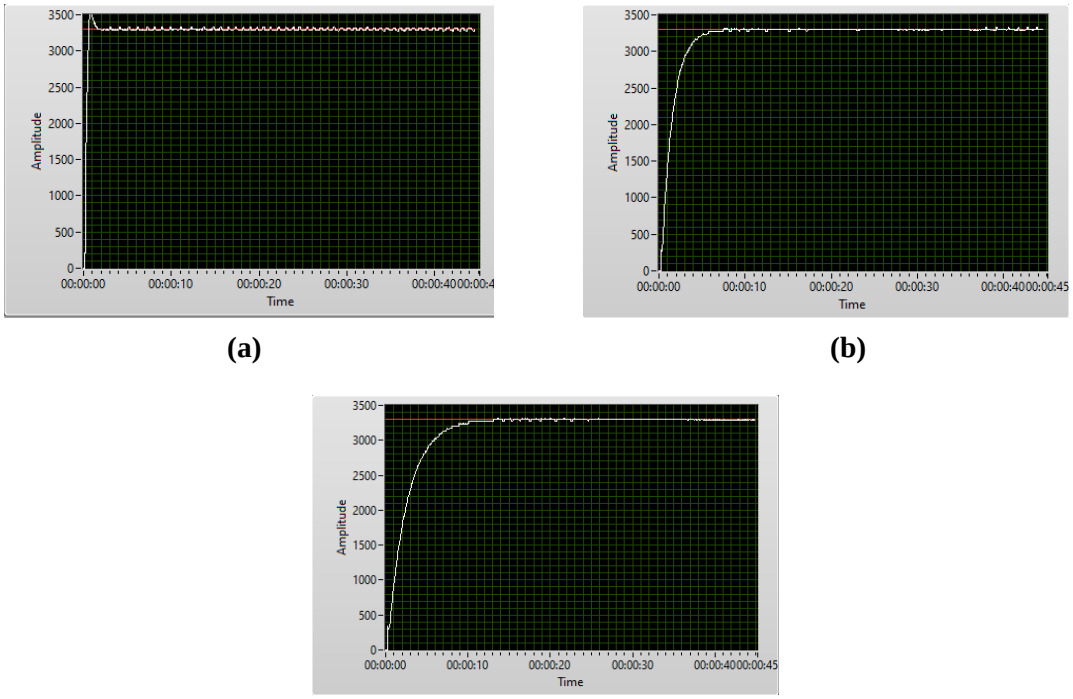
3.3 Hasil Kendali PID Menggunakan Metode PSO

Pada metode PSO, partikel diinisialisasi dengan menentukan batas maksimum dan minimum tiap partikel, jumlah partikel, nilai w (momen inersia), nilai c (koefisien), dan jumlah pergerakan tiap partikel. Setelah tahap inisialisasi, posisi tiap partikel dievaluasi menggunakan PID dengan mengambil nilai error sebagai acuan. Partikel dengan nilai error terkecil akan menjadi partikel dengan posisi terbaik (Gbest). Selanjutnya, posisi tiap partikel diperbarui menggunakan nilai Gbest yang telah diperoleh, sesuai dengan jumlah pergerakan yang telah ditentukan pada tahap awal, untuk mencari posisi terbaik masing-masing partikel (Pbest). Dari seluruh nilai Pbest, partikel dengan nilai error terkecil akan dipilih sebagai Gbest yang baru. Setelah posisi terbaik keseluruhan (Gbest) diperoleh dari posisi terbaik tiap partikel (Pbest), nilai Gbest yang baru ini digunakan sebagai input untuk parameter PID.

Pada penelitian ini, *tuning* parameter PID menggunakan metode PSO diuji dengan mengubah jumlah partikel dan pergerakannya secara acak pada motor DC dengan kecepatan 3300 RPM. Respons sistem dengan berbagai nilai parameter PID hasil *tuning* metode PSO disajikan pada Tabel 4, sedangkan respons sistem secara grafis ditunjukkan pada Gambar 8.

Tabel 4. Respons PSO-PID ZN 2

No	Inisialisasi		Parameter PID			*Rise Time* (s)	*Settling Time* (s)	*Overshoot*
	n-partikel	n-*movement*	Kp	Ki	Kd			
1	5	10	0,001005	0,000332	0,001130	0,7	2	200
2	10	10	0,004643	0,006617	0,005841	5	7	0
3	20	10	0,000630	0,001138	0,001144	8	10	0



(a) **(b)**

(c)

Gambar 8. Respons PSO-PID (a) menggunakan 5 partikel, (b) menggunakan 10 partikel, dan (c) menggunakan 20 partikel

Respons sistem dengan metode *tuning* PSO yang diuji pada beberapa variasi jumlah partikel menunjukkan hasil yang beragam. *Rise time* dan *settling time* tercepat, masing-masing sebesar 0,2 detik dan 2 detik, diperoleh saat jumlah partikel sebanyak 5. Namun, respons ini memiliki *overshoot*. Respons tercepat tanpa *overshoot* diperoleh saat jumlah partikel 10, dengan parameter PID Kp = 0,004643, Ki = 0,006617, dan Kd = 0,005841. Pada konfigurasi ini, *rise time* mencapai 5 detik, dan *settling time* sebesar 7 detik.

3.4 Perbandingan Respons Sistem

Perbandingan respons sistem kendali kecepatan motor DC menggunakan pengendali PID yang diimplementasikan pada NI ELVIS II dan LabVIEW, dengan penyetelan parameter PID menggunakan metode *trial and error*, ZN 2, dan PSO-PID, ditunjukkan pada Tabel 5.

Tabel 5. Perbandingan respons sistem

Metode	Parameter PID			*Rise Time* (s)	*Settling Time* (s)	*Overshoot*
	Kp	Ki	Kd			
Trial and Error	0,01	0,01	0,001	7	12	0
ZN 2	0,0022	0,0075	0,0018	8	11	0
PSO	0,004643	0,006617	0,005841	5	7	0

Tabel 5 menunjukkan perbandingan hasil pengujian metode *tuning* parameter PID menggunakan metode *trial and error*, ZN 2, dan PSO-PID yang menunjukkan perbedaan dalam *rise time*, *settling time*,

dan *overshoot*. Pada pengujian dengan metode *trial and error* dengan parameter Kp = 0,01, Ki = 0,01, dan Kd = 0,001, menghasilkan *rise time* sebesar 7 detik, *settling time* sebesar 12 detik, dan tidak ada *overshoot*. Pada pengujian dengan metode ZN 2 dengan parameter Kp = 0,0022, Ki = 0,0075, dan Kd = 0,0018, menghasilkan *rise time* sebesar 8 detik, *settling time* sebesar 11 detik, dan tidak ada *overshoot*. Sedangkan pada pengujian dengan metode PSO dengan parameter Kp = 0,004643, Ki = 0,006617, dan Kd = 0,005841, menghasilkan *rise time* sebesar 5 detik, *settling time* sebesar 7 detik, dan tidak ada *overshoot*.

4. KESIMPULAN

Setelah melakukan penelitian dan pengujian sistem kendali PID pada pengendalian kecepatan motor DC menggunakan beberapa metode *tuning* PID yang diimplementasikan pada NI ELVIS II dan LabVIEW, dapat disimpulkan hal-hal berikut. *Tuning* parameter PID menggunakan metode PSO, yang diuji dengan variasi jumlah partikel, menghasilkan nilai optimal pada jumlah partikel sebanyak 10, dengan parameter Kp = 0,004643, Ki = 0,006617, dan Kd = 0,005841. Pada parameter ini diperoleh *rise time* sebesar 5 detik, *settling time* sebesar 7 detik, dan tanpa *overshoot*. Perbandingan antara metode *tuning* PSO, ZN 2, dan *trial and error* menunjukkan bahwa ketiganya menghasilkan respons tanpa *overshoot*. Namun, metode PSO menghasilkan respons yang lebih cepat dibandingkan dengan metode *trial and error* dan ZN 2. Penelitian selanjutnya dapat diterapkan algoritma optimisasi yang dapat mengatasi adanya gangguan pada sistem motor DC.

UCAPAN TERIMA KASIH

Ucapan terima kasih diberikan kepada Laboratorium Sistem Kendali, Politeknik Manufaktur Bandung yang telah berkontribusi dalam memberikan fasilitas penelitian.

REFERENSI

[1] M. Ya. Alwardat and P. V. Balabanov, "Speed Control of DC Motor Using PID Controller Based on Matlab," *Vestnik Tambovskogo*, vol. 27, no. 2, pp. 195–202, Jul. 2021, doi: 10.17277/vestnik.2021.02.pp.195-202.

[2] S. Triyani and S. K. Risandriya, "Kendali Kecepatan Motor DC Berbasis Fuzzy Setting Point Pada LabView," *Journal of Applied Electrical Engineering*, vol. 2, no. 1, pp. 6–11, 2018.

[3] N. Roni Wibowo, Aminuddin, and M. Niel Authar Syaputra, "Rancang Bangun Sistem Kendali Kecepatan Motor DC Sebagai Media Pembelajaran Praktikum Sistem Kendali Menggunakan Labview," *Jurnal Sains Terapan*, vol. 6, no. 2, pp. 60–67, 2020.

[4] F. Firdaus, E. Priatna, N. Hiron, and N. Busaeri, "Prototype Sistem Kendali Kecepatan Motor DC dengan Proportional Integral Derivative (PID) Controller," *Journal of Energy and Electrical Engineering (JEEE)*, vol. 4, no. 1, pp. 32–40, 2022.

[5] J. Sun, H. Zhou, X. Ma, and Z. Ju, "Study on PID *tuning* strategy based on dynamic stiffness for radial active magnetic bearing," *ISA Trans*, vol. 80, pp. 458–474, Sep. 2018, doi: 10.1016/j.isatra.2018.07.036.

[6] Y. Fan, J. Shao, G. Sun, and X. Shao, "Improved Beetle Antennae Search Algorithm-Based Lévy Flight for *Tuning* of PID Controller in Force Control System," *Math Probl Eng*, vol. 2020, 2020, doi: 10.1155/2020/4287315.

[7] E. S. Rahayu, A. Ma'arif, and A. Cakan, "Particle Swarm Optimization (PSO) *Tuning* of PID Control on DC Motor," *International Journal of Robotics and Control Systems*, vol. 2, no. 2, pp. 435–447, 2022, doi: 10.31763/ijrcs.v2i2.476.

[8] W. Farag, "Complex Trajectory Tracking Using PID Control for Autonomous Driving," *International Journal of Intelligent Transportation Systems Research*, vol. 18, no. 2, pp. 356–366, May 2020, doi: 10.1007/s13177-019-00204-2.

[9] M. Diah Ika Putri, A. Ma'arif, and R. Dwi Puriyanto, "Pengendali Kecepatan Sudut Motor DC Menggunakan Kontrol PID dan *Tuning* Ziegler Nichols," *Techno*, vol. 23, no. 1, pp. 9–18, 2022.

[10] G. Anwari Siregar and S. Amalia, "Analisis Performansi Pengendali PID Pada Motor DC dengan Menggunakan Metode *Tuning* Cohen-Coon," in *Prosiding Seminar Nasional Sains dan Teknologi*, 2022, pp. 633–638.

[11] S. Fatimah Anggraini, A. Ma, and R. Dwi Puriyanto, "Pengendali PID pada Motor DC dan *Tuning* Menggunakan Metode Differential Evolution PID Controllers on DC Motors and *Tuning* Using the Differential Evolution Method," *TELKA*, vol. 6, no. 2, pp. 147–159, 2020.

[12] E. W. Suseno and A. Ma'Arif, "*Tuning* of PID Controller Parameters with Genetic Algorithm Method on DC Motor," *International Journal of Robotics and Control Systems*, vol. 1, no. 1, pp. 41–53, 2021, doi: 10.31763/ijrcs.v1i1.249.

[13] A. Alkrwy, A. A. Hussein, T. H. Atyia, and M. Khamees, "Adaptive *Tuning* of PID Controller using Crow Search Algorithm for DC motor," *IOP Conf Ser Mater Sci Eng*, vol. 1076, no. 1, p. 012001, Feb. 2021, doi: 10.1088/1757-899x/1076/1/012001.
[14] M. A. Shamseldin, "Optimal coronavirus optimization algorithm based pid controller for high performance brushless dc motor," *Algorithms*, vol. 14, no. 7, Jul. 2021, doi: 10.3390/a14070193.
[15] J. Agarwal, G. Parmar, R. Gupta, and A. Sikander, "Analysis of grey wolf optimizer based fractional order PID controller in speed control of DC motor," *Microsystem Technologies*, vol. 24, no. 12, pp. 4997–5006, Dec. 2018, doi: 10.1007/s00542-018-3920-4.
[16] M. Nur Masrukhan, M. Piono Mulyo, D. Ajiatmo, and M. Ali, "Optimasi Kecepatan Motor DC Menggunakan PID dengan *Tuning* Ant Colony Optimization (ACO) Controller," in *Prosiding SENTIA*, 2016, pp. 49–52.
[17] Y. Gala Hartlambang, H. Nurohmah, and M. Ali, "Optimasi Kecepatan Motor DC Menggunakan Algoritma Kelelawar (Bat Algorithm)," 2017.
[18] A. K. Kashyap and D. R. Parhi, "Particle Swarm Optimization aided PID gait controller design for a humanoid robot," *ISA Trans*, vol. 114, pp. 306–330, Aug. 2021, doi: 10.1016/j.isatra.2020.12.033.
[19] H. Feng, W. Ma, C. Yin, and D. Cao, "Trajectory control of electro-hydraulic position servo system using improved PSO-PID controller," *Autom Constr*, vol. 127, Jul. 2021, doi: 10.1016/j.autcon.2021.103722.
[20] S. M. H. Mousakazemi and N. Ayoobian, "Robust tuned PID controller with PSO based on two-point kinetic model and adaptive disturbance rejection for a PWR-type reactor," *Progress in Nuclear Energy*, vol. 111, pp. 183–194, Mar. 2019, doi: 10.1016/j.pnucene.2018.11.003.
[21] M. E.El-Telbany, "*Tuning* PID Controller for DC Motor: An Artificial Bees Optimization Approach," *Int J Comput Appl*, vol. 77, no. 15, pp. 18–21, Sep. 2013, doi: 10.5120/13559-1341.
[22] H. Supriyanto, F. Suryatini, A. Rohman, H. Martawireja, and H. Rudiansyah, "Implementasi Kontroler PID dengan Metode *Tuning* Ziegler-Nichols dan Cohen-Coon pada Sistem SCADA Kendali Level Air," *Jurnal Teknologi Terapan)* |, vol. 8, no. 2, pp. 149–157, 2022.
[23] M. Ya. Alwardat and P. V. Balabanov, "Speed Control of DC Motor using PID Controller based on Matlab," *Vestnik Tambovskogo gosudarstvennogo tehnicheskogo universiteta*, vol. 27, no. 2, pp. 195–202, Jul. 2021, doi: 10.17277/vestnik.2021.02.pp.195-202.
[24] Katsuhiko. Ogata, *Modern control engineering*. Prentice Hall, 2010.
[25] Y. A. K. Utama and T. Tamaji, "Perbandingan Metode *Tuning* PID pada Pengaturan Kecepatan Parallel Hybrid Electric Vehicle," *Telekontran : Jurnal Ilmiah Telekomunikasi, Kendali dan Elektronika Terapan*, vol. 10, no. 1, pp. 9–17, Aug. 2022, doi: 10.34010/telekontran.v10i1.7411.
[26] Z. Xiang, D. Ji, H. Zhang, H. Wu, and Y. Li, "A simple PID-based strategy for particle swarm optimization algorithm," *Inf Sci (N Y)*, vol. 502, pp. 558–574, Oct. 2019, doi: 10.1016/j.ins.2019.06.042.
[27] A. K. Kashyap and D. R. Parhi, "Particle Swarm Optimization aided PID gait controller design for a humanoid robot," *ISA Trans*, vol. 114, pp. 306–330, Aug. 2021, doi: 10.1016/j.isatra.2020.12.033.
[28] A. M. Rizki and A. L. Nurlaili, "Algoritme Particle Swarm Optimization (PSO) untuk Optimasi Perencanaan Produksi Agregat Multi-Site pada Industri Tekstil Rumahan," *Journal of Computer, Electronic, and Telecommunication*, vol. 1, no. 2, pp. 1–9, Jan. 2021, doi: 10.52435/complete.v1i2.73.