Jurnal Integrasi
Vol. 12 No. 1, April 2020, 1-12
e-ISSN: 2548-9828

Article History
Received January 2020
Accepted April 2020

PENALA PARAMETER PID OTOMATIS PADA PENGATUR KECEPATAN MOTOR INDUKSI TIGA FASA

Handri Toar[1], Era Purwanto[2], Hary Oktavianto[3], Ridwan[4], Muhammad Rizani Rusli[5]

[1,4]Politeknik Negeri Batam, Batam, Indonesia
[2,3,5]Politeknik Elektronika Negeri Surabaya, Surabaya, Indonesia
[1]toar@polibatam.ac.id, [2]era@pens.ac.id, [3]hary@pens.ac.id, [4]ridwan@polibatam.ac.id,
[5]ruslirizani@gmail.com

Abstrak

PID (*Proportional-Integral-Derivative*) merupakan salah satu tipe kontroler sederhana, komputasinya ringan, mudah diimplementasi dan dikenal tangguh menghadapi gangguan. Tetapi PID memiliki kelemahan yaitu sulitnya menentukan parameter PID yang optimal, apalagi bila diterapkan pada sistem nonlinear seperti pengaturan kecepatan pada motor induksi berbasis kontrol vektor. Pada paper ini membahas tentang penalaan parameter PID secara otomatis menggunakan algoritma dengan memasukan kriteria parameter performa dinamik serta nilai beban sesuai keinginan pengguna. Luaran hasil penalaan secara otomatis berupa kumpulan data parameter PID terbaik. Pekerjaan ini divalidasi melalui simulasi menggunakan aplikasi LabVIEW dengan pengujian kecepatan dinamik dan beban dinamik dengan pembanding parameter PID hasil *trial&error*. Parameter PID hasil algoritma penala otomatis mampu menghasilkan respon kecepatan yang cepat yang diukur dari masa transiennya kurang dari 20ms, kondisi *overshoot* maksimum 2% ketika kecepatan bertambah dan kondisi *undershoot* maksimum 2% ketika kecepatan berkurang bila dibandingkan dengan parameter PID hasil penggunaan metode *trial&error*.

Kata kunci: PID, penala otomatis, pengatur kecepatan, kontrol vektor, motor induksi, LabVIEW®

Abstract

PID is one of the simple controller, fast computing, easy to implement and reliable to face the disturbance. but PID has a weakness, PID is hard to determine which one is the best parameter especially to implement on the nonlinear system like speed control of the induction motor based on vector control. this paper will discuss tuning PID parameter automatically using an algorithm with input criteria is a collection of the optimal PID parameter After validation using LabVIEW application based on dynamic speed and dynamic load compare to the PID parameter from trial&error. The result are autotuning algrithm give good and faster transient speed respond under 20ms, with overshoot condition maksimum 2% when the speed increases and with undershoot condition maksimum 2% when speed decreases then using PID parameter from trial& error method.

Keyword: *PID, autotuning, speed control, vector control, induction motor, LabVIEW®*

1. Pendahuluan

Motor induksi telah lama menjadi bagian utama dalam penggerak industri, Jika sebelumnya kendaraan listrik menggunakan motor DC sebagai penggerak, kini penggerak tersebut mulai beralih ke motor induksi. Dari sisi emisi, motor ini ramah lingkungan karena tidak menghasilkan gas buang. Keunggulan lainnya adalah motor induksi mudah dirawat, tidak menggunakan sikat penghantar listrik menuju rotor motor sehingga aman dari percikan bunga api, tidak menggunakan magnet permanen yang merupakan logam yang mulai langka [1]. Hal-hal inilah yang membuat motor induksi mulai dilirik oleh para peneliti dan produsen dalam bidang kendaraan listrik [2].

Dalam prakteknya, ketika supir mengemudikan kendaraan listrik terutama perjalanan dalam kota, motor penggerak kendaraan listrik tersebut jarang berputar dengan kecepatan yang sama disetiap waktu, kecepatannya lebih sering berubah. Untuk itu Kendaraan listrik memerlukan motor induksi yang dapat berputar responsif dengan tingkat akurasi yang tinggi. Hal ini berbeda dengan penggunaan motor

induksi terdahulu, seperti kerja kompresor, pompa dan kipas yang selalu berputar dengan kecepatan yang sama. Dan para pengguna motor induksi terdahulu tidak mempermasalahkan jika kecepatan yang dihasilkan oleh motornya tidak presisi.

Pengaturan kecepatan motor induksi dibagi ke dalam dua fokus analisa. Ada yang menganalisa motor induksi sebagai kontrol skalar dan yang lain menganalisanya sebagai kontrol vektor. Jika kontrol skalar mengacu kepada keadaan *steadystate* maka kontrol vektor lebih mengacu pada masa transien [3].

Kontrol skalar lebih menekankan pada mengubah besaran variabel saja. Misalnya mengubah tegangan akan mengatur fluksi, dan mengubah frekuensi atau slip akan mengatur torsi. Kontrol skalar sangat mudah diimplementasi karena kontrol ini memiliki struktur yang sederhana[4]. Namun karena kontrol skalar mengabaikan efek kopling yang terjadi di dalam motor, maka kelemahannya terlihat yakni respon kecepatan yang lamban serta kontrol ini rentan terhadap ketidakstabilan [5]. Kontrol skalar secara umum digunakan sebagai pengatur kecepatan berbiaya rendah yang dengan performa yang cukup.

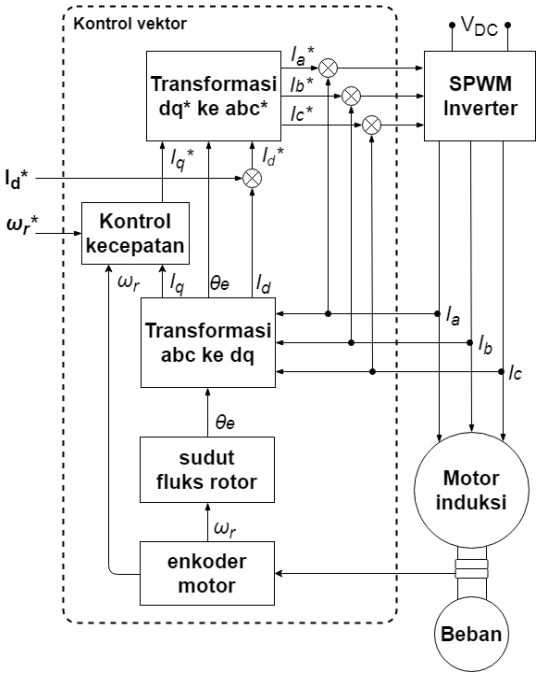
Performa tinggi yang dihasilkan oleh kontrol vektor diawali dari konversi besaran arus yang semula besaran skalar menjadi besaran vektor. Kemudian metode berikutnya adalah kontrol vektor memisahkan kopling motor berdasarkan rangkaian ekivalensi motor induksi [6]. Melalui metode ini, kontrol vektor akan menghasilkan pengaturan fluksi dan pengaturan torsi yang terpisah. Kelemahan kontrol vektor adalah sangat peka terhadap perubahan parameter motor. Selain itu kontrol vektor memerlukan perangkat keras dengan kemampuan komputasi tingkat tinggi dalam menjalankan fungsinya sebagai pengatur kecepatan berperforma tinggi. Performa tinggi ini bermakna bahwa kontroler mampu menghasilkan kecepatan yang presisi, mampu merespon perubahan kecepatan dengan cepat serta stabil terhadap gangguan. Berdasarkan analisa di atas maka penggunaan metode kontrol vektor adalah pilihan yang bijak.

Agar kontrol vektor selalu beroperasi pada performa puncaknya, maka kontrol vektor perlu kontroler tambahan yaitu kontroler optimasi. Salah satu kontroler optimasi yang sederhana adalah kontroler PID. kontroler PID bertugas mengoptimalkan peran kontrol vektor. Salah satu kelemahan dari kontroler PID adalah sulitnya menangani sistem nonlinear, karena performanya akan semakin menurun sementara motor induksi merupakan sebuah sistem nonlinear[7]. Kelemahan tersebut dapat diatasi dengan menala ulang (*tune up)* setiap ada perubahan kondisi sehingga performa pengatur kecepatan motor induksi tiga fasa berbasis kontrol vektor tetap terjaga optimal. Salah satu metode yang digunakan untuk menala ulang parameter PID adalah algoritma penala otomatis.

2. Metode

2.1. Kontrol Vektor

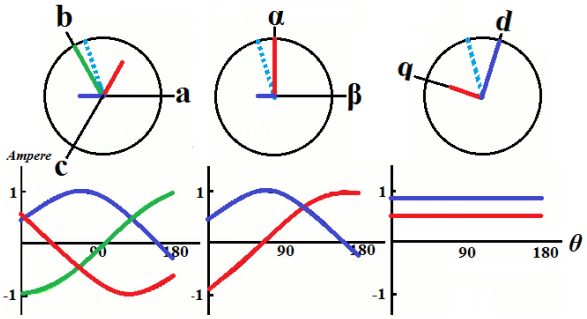
Gambar 1 menunjukkan blok diagram kontrol vektor. Penggunaan kontrol vektor membutuhkan masukan berupa pembacaan konsumsi arus stator, karena arus ini menyimpan informasi penting berupa besaran fluksi dan besaran torsi. Supaya kontrol vektor mendapatkan nilai kedua besaran tersebut, langkah pertama adalah kontroler mentransformasi arus AC tiga fasa yang semula adalah besaran skalar menjadi arus besaran dengan besaran vektor. Kontroler akan memisahkan arus tiga fasa tersebut menjadi arus fluksi dan arus torsi. Dalam mengatur kecepatan motor induksi, kontroler ini hanya perlu berfokus kepada pengaturan arus torsi dengan menjaga arus fluksi bernilai konstan yaitu bernilai satu.



Gambar 1: Blok diagram kontrol vektor pada motor induksi

2.2. Transformasi Clarke dan Park

Gambar 2 menunjukkan proses transformasi dari besaran skalar dengan arus AC tiga fasa (i_{abc}) dan berakhir menjadi besaran vektor dengan dua arus searah (i_{dq}). Pada transformasi Clarke, arus stator i_{abc} ini diubah menjadi vektor arus berbentuk stasionary dua fasa ($i_{\alpha\beta}$) berdasarkan (1) dan (2).



Gambar 2: Transformasi clarke park

$$i_\alpha = \quad i_a \tag{1}$$

$$i_\beta = \frac{1}{\sqrt{3}} i_a + \frac{2}{\sqrt{3}} i_b \tag{2}$$

Transformasi *invers* ditunjukan melalui (3),(4) dan (5).

$$i_a = \quad i_\alpha \tag{3}$$

$$i_b = -\frac{1}{2} i_\alpha + \frac{\sqrt{3}}{2} i_\beta \tag{4}$$

$$i_c = -\frac{1}{2} i_\alpha - \frac{\sqrt{3}}{2} i_\beta \tag{5}$$

Pada tahapan ini, iαβ sudah berbentuk vektor sumbu stasionary α dan β, tetapi vektor ini masih berupa arus AC dan vektor ini berputar pada titik porosnya. Supaya vektor arus ini tidak berputar, sebagai ganti maka sumbunya yang berputar terhadap porosnya. Sehingga jika dipandang dari sumbu yang berputar ini, vektor akan menjadi arus DC (i_{dq}) pada sumbu *direct* dan *quadrature*. Langkah ini menggunakan transformasi Park melalui (6) dan (7).

$$i_d = \quad i_\alpha \cos\theta - i_\beta \sin\theta \tag{6}$$

$$i_q = -i_\alpha \cos\theta + i_\beta \sin\theta \tag{7}$$

Transformasi *invers* ditunjukkan pada (8) dan (9).

$$i_\alpha = \quad i_d \cos\theta - i_q \sin\theta \tag{8}$$

$$i_\beta = -i_d \cos\theta + i_q \sin\theta \tag{9}$$

Transformasi Park membutuhkan parameter sudut fluksi rotor (θ_e). Salah satu metode untuk mendapatkannya adalah melalui metode estimasi fluksi rotor.

2.3. Estimasi sudut fluksi rotor

Permasalahan pertama dalam metode kontrol vektor adalah menentukan keakuratan θ_e, salah satunya didapati dari perhitungan kecepatan putar rotor (ω_r) dan slip (ω_{sl}) dari (10).

$$\theta_e = \int (\omega_r + \omega_{sl}) dt \tag{10}$$

Informasi ω_r didapati dari sensor enkoder yang terhubung pada rotor motor, sementara ω_{sl} dihitung berdasarkan arus stator referensi i_{qs}^*, i_{ds}^* dan flux linkage rotor (λ_r) melalui (11) dan (12).

$$\omega_{sl} = \frac{L_m}{\lambda_r^*} \frac{R_r}{L_r} i_{qs}^* \tag{11}$$

$$\lambda_r = \frac{L_m i_{ds}}{1 + \tau_r s} \tag{12}$$

Yang harus diperhatikan adalah pengaturan λ_r karena didalamnya terdapat *rotor time constant* (τ_r) yang merupakan rasio antara induktansi rotor (Lr) terhadap tahanan rotor (R_r) pada (13). Rasio ini harus diperhatikan supaya nilainya selalu konstan karena rasio tersebut dapat berubah ketika motor bergerak. Selain itu, rasio ini sangat berhubungan dengan efek *ohmic heating* yang akan menyebabkan motor panas.

$$\tau_r = \frac{L_r}{R_r} \tag{13}$$

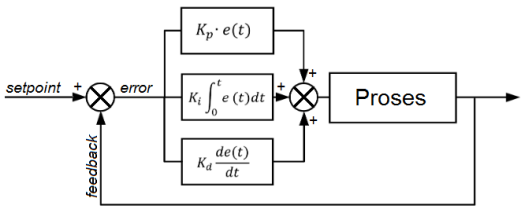
Penentuan nilai arus referensi yang berhubungan fluksi dan torsi dapat dikalkulasikan secara terpisah. Pengaturan torsi diatur melalui regulasi arus i_{qs}^* dan pengaturan fluksi diatur melalui regulasi i_{ds}^* pada (14) dan (15).

$$i_{qs}^* = \frac{2}{3} \cdot \frac{2}{P} \cdot \frac{L_r}{L_m} \cdot \frac{T_e^*}{\lambda_r^*} \tag{14}$$

$$i_{ds}^* = \frac{\lambda^*}{L_m} \tag{15}$$

Setelah pengaturan fluksi dan torsi dilakukan secara terpisah, maka langkah selanjutnya adalah menyiapkan kontroler untuk mengatur kecepatan sesuai yang diinginkan. Salah satu cara sederhananya adalah menggunakan kontroler PID.

2.4. Kontroler PID



Gambar 3: Blok diagram kontroler PID

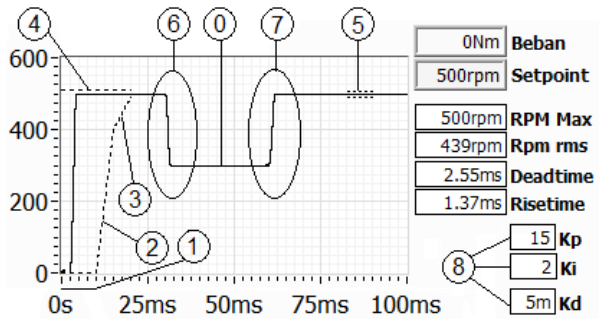
PID merupakan kontroler sederhana yang terdiri dari Proporsional, Integral, Derivatif [8] dan saling memiliki pengaruh satu terhadap yang lain. Gambar 3 menyatakan bahwa jika ketiga parameter tersebut dikombinasikan secara optimal akan menghasilkan respon luaran yang baik. Proporsional sangat

responsif terhadap sinyal error, berperan penting dimasa transien. Integral berperan menjaga kestabilan sinyal pada masa *steadystate*. Sementara Derivatif berperan untuk memperhalus tugas proporsional. PID sangat cocok untuk sistem yang linear dan kualitasnya mulai berkurang ketika menghadapi sistem nonlinear [9].

2.5. Algoritma penala otomatis

Proses mendapatkan parameter PID tidak serta merta berhasil didapatkan dalam sekali percobaan, proses butuh banyak tenaga yang terkuras dan perlu waktu lama, terutama jika menggunakan metode tradisional seperti *trial&error*. Untuk itu diperlukan suatu metode agar perolehan parameter tersebut berjalan otomatis. Ziegler dan Nichols adalah peneliti pertama yang mengusulkan metode penalaan PID berdasarkan karakteristik masa transien [10]. Penggunaan metode ini mensyaratkan sistem harus mengalami masa osilasi kritis terlebih dahulu, sayangnya tidak semua sistem dapat memenuhi syarat tersebut, salah satunya adalah pada kontrol vektor pada motor induksi. Sehingga sejak saat itu pengembangan penalaan PID mulai mendapatkan perhatian lebih. Misalkan pada [11] mengembangkan algoritma genetika untuk memperoleh parameter PID pada pengaturan motor induksi hingga 100 generasi yang diuji untuk tiga beban yang berbeda, didapatkan hasil yang optimal. Pada [12] dengan menggunakan modifikasi *particle swarm optimization* (PSO) yang menggunakan koefisien penyempitan waktu yang beragam dapat meningkatkan hasil yang akurat serta dapat menghemat waktu pencarian parameter PID yang berupa nilai konstanta K_p, K_i dan K_d.

Algoritma penala otomatis akan mencari parameter PID untuk mengatur kecepatan motor induksi berbasis kontrol vektor berdasarkan dua masukan yaitu *setpoint* kecepatan dan respon beban yang dikeluarkan untuk menghasilkan respon kecepatan yang ditentukan pada Tabel 1 dan Gambar 4 menjelaskan bagian-bagian dalam grafik respon kecepatan yang akan digunakan. Dan Flowchart mengenai algoritma penala otomatis bekerja ditunjukkan oleh Gambar 5.

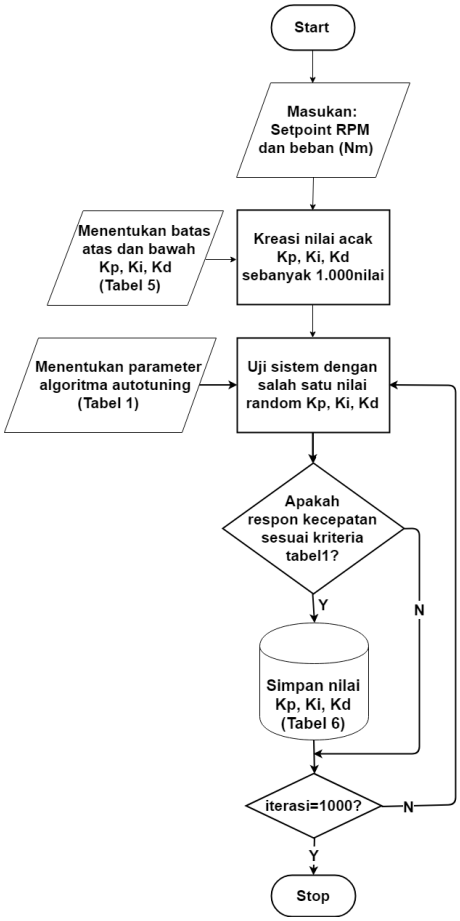


Gambar 4: keterangan pada grafik respon kecepatan

TABEL 1

KARAKTERISTIK RESPON KECEPATAN

No.	Deskripsi	Nilai
	Time simulation	=150ms
	Discrete time (d_t)	=5e-6s
1.	*Dead time* (t_d)	<=10ms
2.	*Rise time*	*Dead time* +10ms
3.	*Middle*	<=15ms, <=*Setpoint*-20%
4.	*Overshoot*	<=*Setpoint*±2%
5.	*Steadystate* (s_t)	<=*Setpoint*±2%
6.	Transisi kecepatan turun	-
7.	Transisi kecepatan naik	-
8.	Parameter PID	-



Gambar 5: Flowchart skema penala otomatis

TABEL 2

SPESIFIKASI PERANGKAT KOMPUTER

Deskripsi	Spesifikasi
Sistem operasi	Windows 7 64bit
Prosesor	Intel Core i5-2304 4CPU@3.0GHz
Memori	8GB RAM
Perangkat lunak	LabVIEW 2016 versi pelajar
Perangkat keras	MyRIO-1900
Waktu diskrit	5E-6s
Waktu simulasi	100ms dan 600ms

3. Hasil dan Pembahasan

Investigasi performa penala otomatis dilakukan dalam simulasi menggunakan spesifikasi komputer yang tertera pada Tabel 2. Penentuan parameter PID menggunakan algoritma penala otomatis ini memerlukan waktu yang cukup lama mengingat sederhananya algoritma yang digunakan tetapi tetap ringan dalam kalkulasinya sehingga proses ini dapat disimulasikan dengan baik menggunakan komputer maupun pada *Digital Signal Processing* (DSP) berupa LabVIEW MyRIO. Keduanya menghasilkan hasil respon yang serupa. Perbedaannya hanya pada waktu komputasi, seperti yang ditampilkan pada Tabel 3.

TABEL 3

WAKTU KOMPUTASI

Perangkat keras	Konsumsi waktu
Komputer	±10menit
LabVIEW MyRIO	±10jam 32menit

Parameter motor induksi yang digunakan dalam simulasi ini ditunjukkan pada Tabel 4.

TABEL 4

PARAMETER MOTOR INDUKSI

Spesifikasi	Nilai
Tahanan stator	0,896 Ω
Tahanan rotor	1,82 Ω
Induktansi stator	1,94 mH
Induktansi rotor	2,45 mH
Induktansi mutual	46,2 mH
Momen inersia	0,000225 $Kg.m^2$
Jumlah kutub	4

Algoritma penala otomatis menggunakan batasan parameter PID yang telah ditentukan pada Tabel 5.

TABEL 5

BATASAN NILAI ALGORITMA PENALA OTOMATIS

Parameter	Batasan nilai
Parameter P	0-100
Parameter I	0-1000
Parameter D	0-0.01

Penelitian ini melakukan beberapa skenario simulasi. Skenario pertama, kedua dan ketiga digunakan untuk mengumpulkan hasil dari algoritma penala parameter PID otomatis lalu disimpan dalam format Tabel 6. skenario keempat dan seterusnya digunakan untuk menvalidasi bahwa algoritma penala otomatis berjalan dengan optimal serta membuktikan algoritma ini lebih baik dibanding metode *trial&error* pada pengaturan kecepatan motor induksi berbasis kontrol vektor.

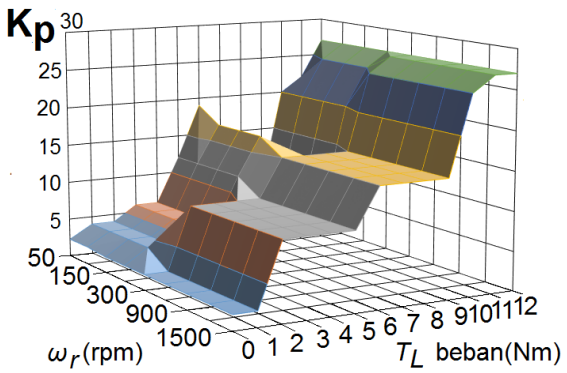
TABEL 6

CONTOH TABEL PENYIMPANAN PARAMETER PID

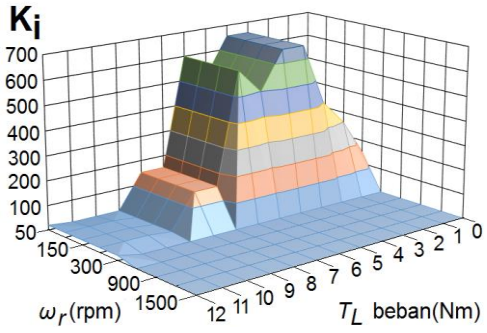
Setpoint (rpm) / (Nm) Beban	50			300			…			1500		
	P	I	D	P	I	D	P	I	D	P	I	D
0	2.3	0	0.3m	15	0	0.3m	…	…	…	3		0.3m
1	4	0	0.5m	20	30	0.50m	…	…	…	3		0.3m
2	4	20	0.7m	23	32	0.65m	…	…	…	15	600	2.3m
…	…	…	…	…	…	…	…	…	…	…	…	…
12	28	20	3m	30	0	3m	…	…	…	28	0	1.5m

3.1. Skenario ke-1: Mendapatkan koleksi parameter penala otomatis

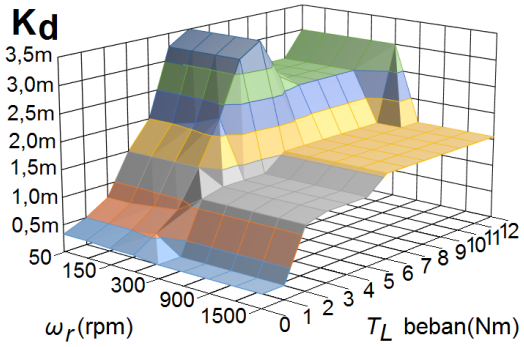
Simulasi algoritma penala otomatis dijalankan mulai dari kecepatan 50rpm hingga 1500rpm dengan kelipatan setiap 100rpm dengan pertambahan beban per 1Nm mulai dari 0Nm hingga mencapai beban puncak yaitu 12Nm, kemudian hasilnya disimpan sesuai dengan format pada Tabel 6. lalu hasil tersebut disajikan dalam bentuk grafik tiga dimensi pada Gambar 6, Gambar 7 dan Gambar 8.



Gambar 6: Pemetaan parameter K_p pada *setpoint* 50-1500rpm dan beban 0-12Nm



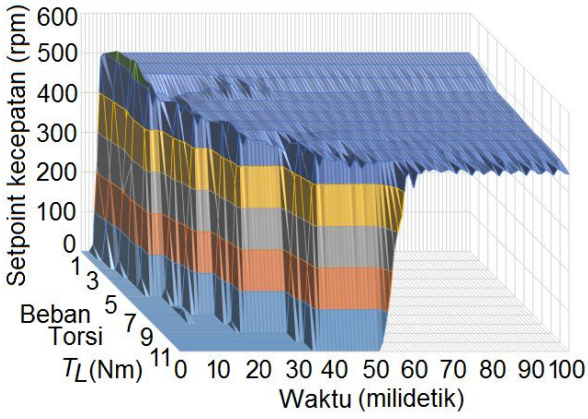
Gambar 7: Pemetaan parameter K_i pada *setpoint* 50-1500rpm dan beban 0-12pNm



Gambar 8: Pemetaan parameter K_d pada *setpoint* 50-1500rpm dan beban 0-12Nm

3.2. Skenario ke-2: Respon penggunaan metode *trial&error* pada masa transien di semua beban

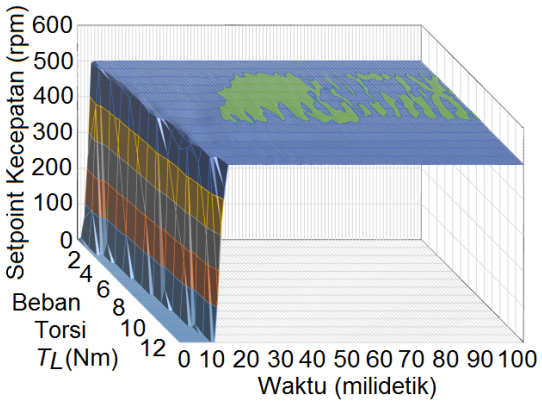
Skenario kedua ini melakukan simulasi untuk mendapatkan respon kecepatan pada *setpoint* 500rpm mulai dari beban 0 hingga 12Nm dengan kelipatan setiap 1Nm dan hasilnya ditampilkan pada Gambar 9. Hasil simulasi memperlihatkan kualitas respon kecepatannya yang semakin menurun seiring meningkatnya beban yang dialami oleh motor, selain itu kondisi *deadtime* dan *error steadystate* juga semakin meningkat.



Gambar 9: Respon kecepatan pada metode *trial&error* *setpoint* 500rpm dan beban torsi 0-12Nm

3.3. Skenario ke-3: Respon penggunaan algoritma penala otomatis pada masa transien di semua beban

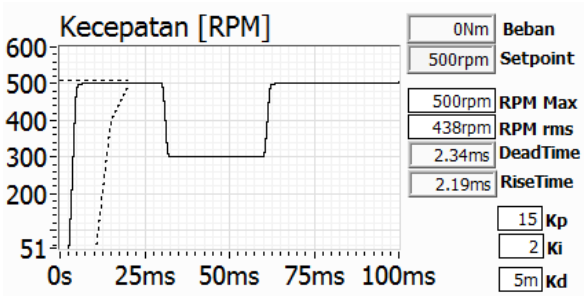
Skenario ketiga ini melakukan simulasi untuk mendapatkan respon kecepatan menggunakan algoritma penala otomatis pada *setpoint* 500rpm mulai dari beban 0 hingga 12Nm dengan kelipatan setiap 1Nm kemudian ditampilkan pada Gambar 10.



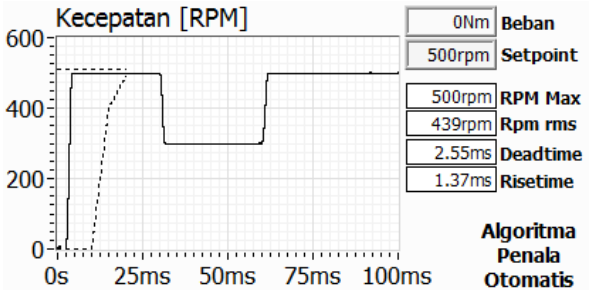
Gambar 10: Respon kecepatan pada algoritma penala otomatis *setpoint* 500rpm dan beban torsi 0-12Nm hasil

Perbandingan antara Gambar 9 dan Gambar 10 membuktikan bahwa algoritma penala parameter PID otomatis dapat menjaga kualitas performa kontroler PID dalam mengatur kecepatan motor induksi tiga fasa berbasis kontrol vektor. Penggunaan algoritma ini membuat respon kecepatan menjadi seragam pada beban berapapun. Demikian juga dengan karakteristik respon lainnya tetap masuk dalam kriteria pada Tabel 1.

3.4. Skenario ke-4: Kecepatan rendah (500rpm) dengan beban nol



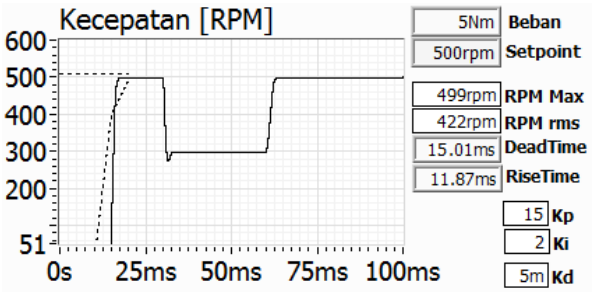
Gambar 11: Respon kecepatan pada metode *trial&error* *setpoint* kecepatan 500rpm dengan beban nol



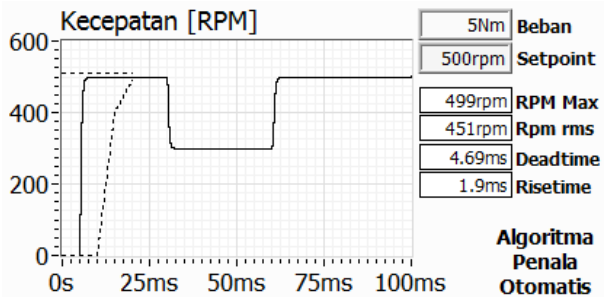
Gambar 12: Respon kecepatan pada algoritma penala otomatis *setpoint* kecepatan 500rpm dengan beban nol

Skenario keempat melakukan simulasi untuk mendapatkan respon pada kecepatan rendah yaitu 500rpm dengan kondisi motor dengan beban nol. Dari Gambar 11 dan Gambar 12, keduanya menunjukkan respon yang serupa antara penggunaan metode *trial&error* maupun algoritma penala otomatis yang menyatakan bahwa kedua respon tersebut masih masuk dalam kriteria yang sesuai pada Tabel 1. Serta respon kecepatan tersebut tidak mengalami kondisi *overshoot* dan *undershoot*.

3.5. Skenario ke-5: Kecepatan rendah (500rpm) dengan beban menengah (5Nm)



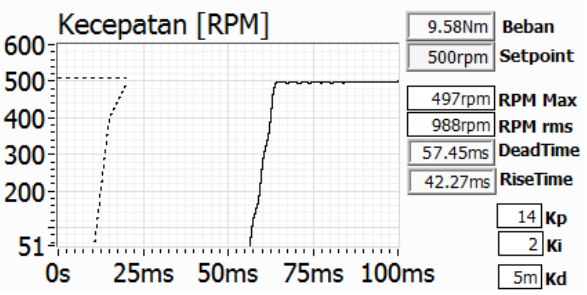
Gambar 13: Respon kecepatan metode *trial&error* *setpoint* 500rpm dengan beban 5Nm



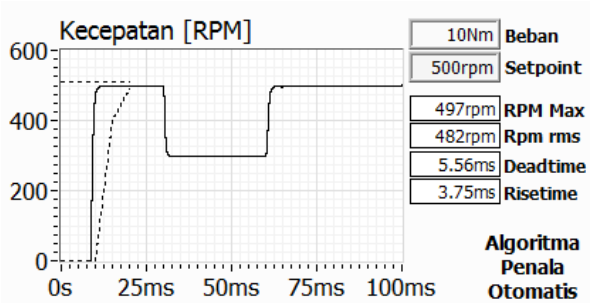
Gambar 14: Respon kecepatan pada algoritma penala otomatis *setpoint* 500rpm dengan beban 5Nm

Gambar 13 memperlihatkan respon kecepatan penggunan metode *trial&error* pada kecepatan rendah yaitu 500rpm dengan beban menengah yaitu 5Nm tidak dapat memenuhi syarat yang telah ditentukan pada Tabel 1. Selain itu terlihat adanya kondisi *undershoot* ketika motor mengalami transisi kecepatan turun dari 500rpm ke 300rpm. Sementara Gambar 14 memperlihatkan penggunaan algoritma penala otomatis masih menunjukkan hasil yang konsisten yaitu masih termasuk dalam kriteria yang ditentukan pada Tabel 1 tanpa mengalami kondisi *overshoot* maupun *undershoot*.

3.6. Skenario ke-6: Kecepatan rendah (500rpm) dengan beban berat (10Nm)



Gambar 15: Respon kecepatan metode *trial&error* *setpoint* 500rpm dengan beban 10Nm

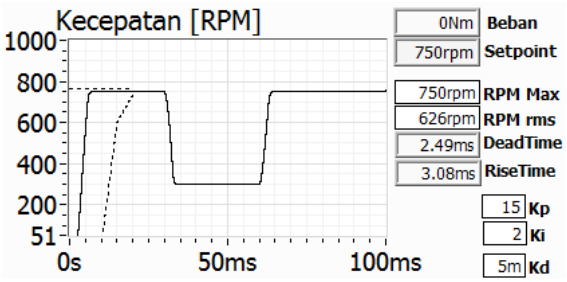


Gambar 16: Respon kecepatan pada algoritma penala otomatis *setpoint* 500rpm dengan beban 10Nm

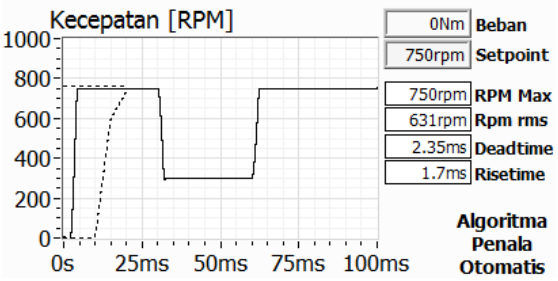
Skenario keenam menjalankan simulasi untuk mengetahui respon pada kecepatan rendah yaitu 500rpm dengan beban berat yaitu 10Nm. Pada Gambar 15 menunjukkan bahwa penggunaan metode *trial&error* tidak mampu menghasilkan respon kecepatan sesuai kriteria yang sesuai pada Tabel 1. Sementara hasil penggunaan algoritma penala otomatis yang terlihat pada Gambar 16 masih konsisten dalam memenuhi kriteria tersebut tanpa mengalami kondisi *overshoot* dan *undershoot*.

3.7. Skenario ke-7: Kecepatan menengah (750rpm) dengan beban nol

Skenario ketujuh menjalankan simulasi untuk mengetahui respon pada kecepatan menengah yaitu 750rpm dengan beban nol. Gambar 17 dan Gambar 18, keduanya menunjukkan respon hasil yang serupa antara penggunaan metode PID maupun algoritma penala otomatis, respon kecepatan masih dalam kriteria yang telah ditentukan pada Tabel 1 dan kedua respon tersebut kecepatan tidak mengalami kondisi *overshoot* dan *undershoot*.

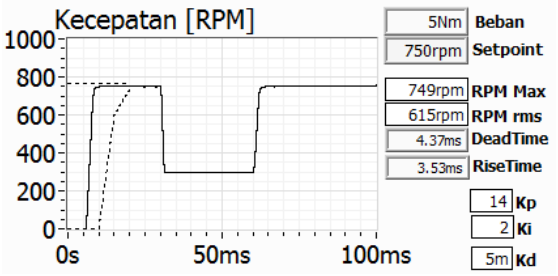


Gambar 17: Respon kecepatan pada metode *trial&error* *setpoint* 750rpm dengan beban nol

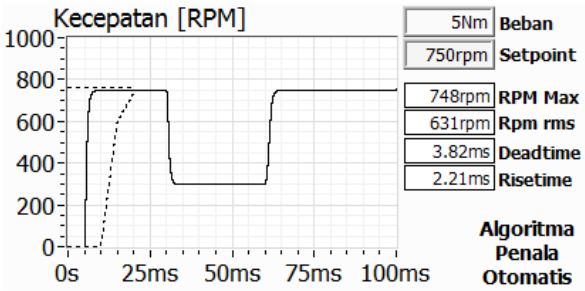


Gambar 18:Respon kecepatan pada algoritma penala otomatis *setpoint* 750rpm dengan beban nol

3.8. Skenario ke-8: Kecepatan menengah (750rpm) dengan beban menengah (5Nm)



Gambar 19: Respon kecepatan metode *trial&error* *setpoint* 750rpm dengan beban 5Nm

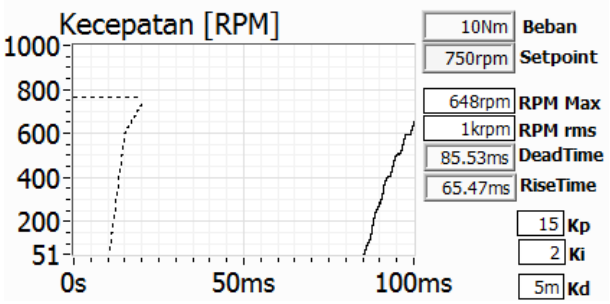


Gambar 20: Respon kecepatan pada algoritma penala otomatis *setpoint* 750rpm dengan beban 5Nm

Skenario kedelapan menjalankan simulasi untuk mendapatkan respon pada kecepatan menengah yaitu 750rpm dengan beban menengah yaitu 5Nm. Gambar 19 dan Gambar 20, keduanya menunjukkan respon hasil yang serupa antara penggunaan metode PID maupun algoritma penala otomatis, respon kecepatan masih dalam kriteria yang telah ditentukan pada Tabel 1 dan hasilnya tidak mengalami kondisi *overshoot* dan *undershoot*.

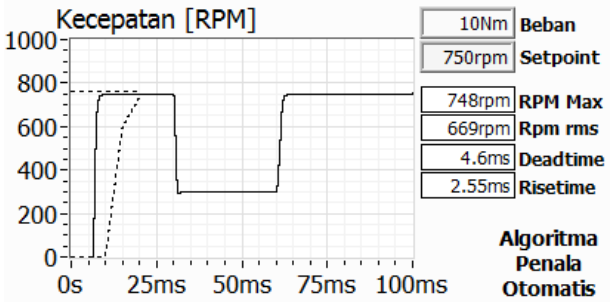
3.9. Skenario ke-9: Kecepatan menengah (750rpm) dengan beban berat (10Nm)

Skenario kesembilan menjalankan simulasi untuk mengetahui respon pada kecepatan menengah yaitu 750rpm dengan beban berat yaitu 10Nm. Penggunaan metode *trial&error* ini membutuhkan *deadtime* sebesar 85.53ms dan *risetime* yang lebih lama sehingga dinyatakan tidak mampu memenuhi kriteria yang ada pada Tabel 1.



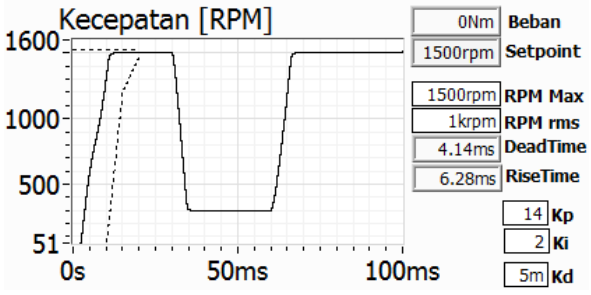
Gambar 21: Respon kecepatan metode *trial&error* *setpoint* 750rpm dengan beban 10Nm

Sementara penggunaan algoritma penala otomatis masih konsisten dalam memenuhi kriteria pada Tabel 1 tanpa mengalami kondisi *overshoot* dan *undershoot*.



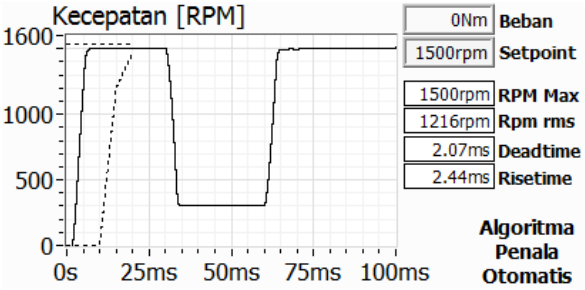
Gambar 22: Respon kecepatan algoritma penala otomatis *setpoint* 750rpm dengan beban 10Nm

3.10.Skenario ke-10: Kecepatan tinggi (1000rpm) dengan beban nol



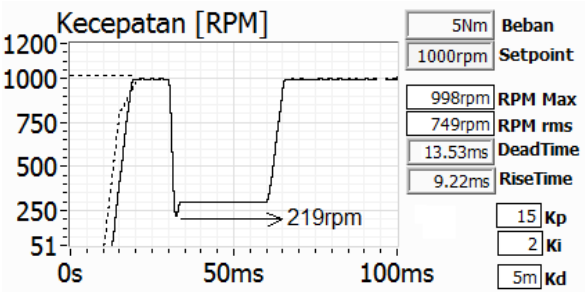
Gambar 23: Respon kecepatan pada metode trial&error *setpoint* 1500rpm dengan dengan beban nol

Simulasi kesepuluh dilakukan untuk mengetahui respon pada kecepatan tinggi yaitu 1000rpm dengan beban nol. Gambar 23 dan Gambar 24 menunjukkan respon kecepatan yang serupa antara penggunaan metode PID maupun algoritma penala otomatis, respon kecepatan masih dalam kriteria yang telah ditentukan pada Tabel 1 dan kedua respon kecepatan tersebut tidak mengalami kondisi *overshoot* dan *undershoot*.



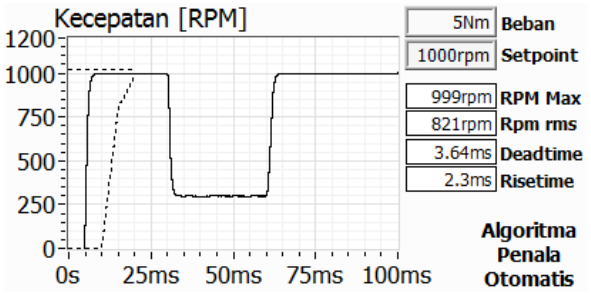
Gambar 24: Respon kecepatan pada algoritma penala otomatis *setpoint* 1500rpm dengan dengan beban nol

3.11.Skenario ke-11: Kecepatan tinggi (1000rpm) dengan beban menengah (5Nm)



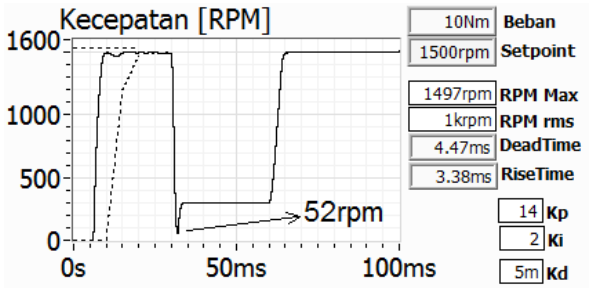
Gambar 25: Respon kecepatan pada metode *trial&error* *setpoint* 1000rpm dengan beban 5Nm

Gambar 25 memperlihatkan respon kecepatan pada penggunaan metode trial&error untuk kecepatan tinggi yaitu 1000rpm dengan beban menengah yaitu 5Nm. Respon tersebut tidak dapat memenuhi persyaratan yang ada pada Tabel 1. serta respon mengalami keadaan *undershoot* ketika terjadi transisi kecepatan turun dari 1000rpm menuju 300rpm hingga menyentuh angka 219rpm. Dan Gambar 26 perlihatkan respon penggunaan algoritma penala otomatis masih konsisten dalam memenuhi persyaratan yang ada pada Tabel 1 dengan tidak mengalami kondisi *overshoot* maupun *undershoot*.



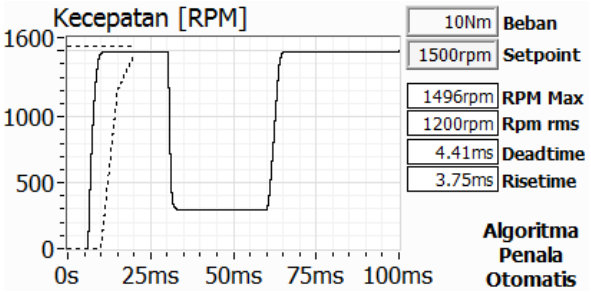
Gambar 26: Respon kecepatan pada algoritma penala otomatis *setpoint* 1000rpm dengan beban 5Nm

3.12.Skenario ke-12: Kecepatan tinggi (1000rpm) dan beban berat (10Nm)



Gambar 27: Respon kecepatan pada metode *trial&error* *setpoint* 1500rpm dengan beban 10Nm

Gambar 27 memperlihatkan respon kecepatan pada penggunaan metode *trial&error* untuk kecepatan tinggi yaitu 1000rpm dengan beban berat yaitu 10Nm tidak dapat memenuhi persyaratan pada Tabel 1 karena mengalami kondisi *undershoot* ketika transisi kecepatan turun dari 1000 rpm menuju 300rpm.



Gambar 28: Respon kecepatan pada algoritma penala otomatis *setpoint* 1500rpm dengan beban 10Nm

Sedangkan pada Gambar 28 yang menunjukkan penggunaan algoritma penala otomatis masih konsisten dalam memenuhi peryaratan tersebut tanpa mengalami kondisi *overshoot* maupun *undershoot*.

3.13.Skenario ke13: Variasi kecepatan dengan beban nol

Skenario ketiga belas ini berbeda dengan skenario sebelumnya, dimana pada skenario ini diuji menggunakan kecepatan dinamis beban nol yang menghasilkan beberapa poin analisa yaitu:

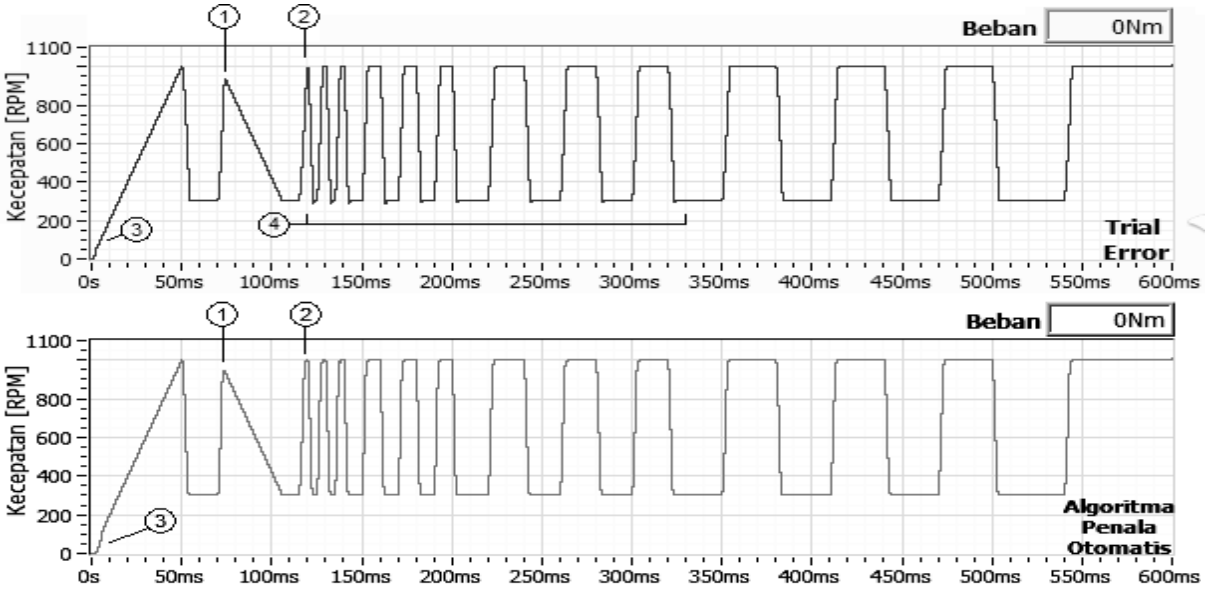
1. Penggunaan metode *trial&error* maupun algoritma penala otomatis tidak dapat mengikuti alur kecepatan yang diinginkan, hal ini terlihat pada poin 1 yang dapat dilihat pada Gambar 29, bentuk segitiga pada poin tersebut tidak dapat

2. mencapai angka 1000rpm. Pada metode t*rial&error* hanya sampai pada titik 916rpm sedangkan algoritma penala otomatis mencapai titik yang lebih tinggi yaitu 921rpm.

3. Pada poin ini, Baik pada metode *trial&error* maupun algoritma penala otomatis masih mampu mencapai puncak 1000rpm meski hanya diberi jarak sebesar 5ms.

4. Meski pada skenario sebelumnya, algoritma penala otomatis menunjukkan keunggulannya namun pada skenario ini, bentuk respon risetime-nya tidak sebaik yang dihasilkan oleh metode *trial&error.*

5. Penggunaan metode *trial&error* pada pengujian dengan beban nol sudah mengalami kondisi *undershoot* meskipun nilainya sangat kecil.



Gambar 29: Respon kecepatan pada metode *trial&error* dan algoritma penala otomatis pada *setpoint* dinamis dengan beban nol

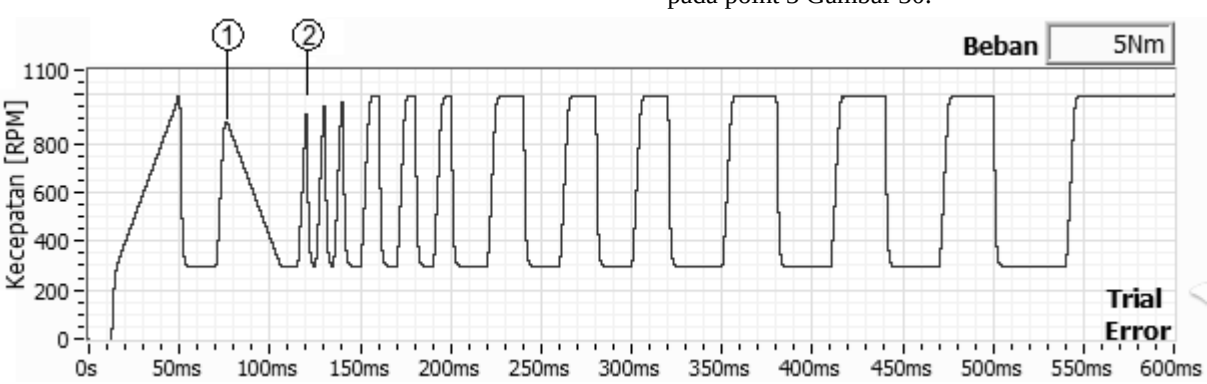
3.14.Skenario ke14: Variasi kecepatan pada dengan beban menengah (5Nm)

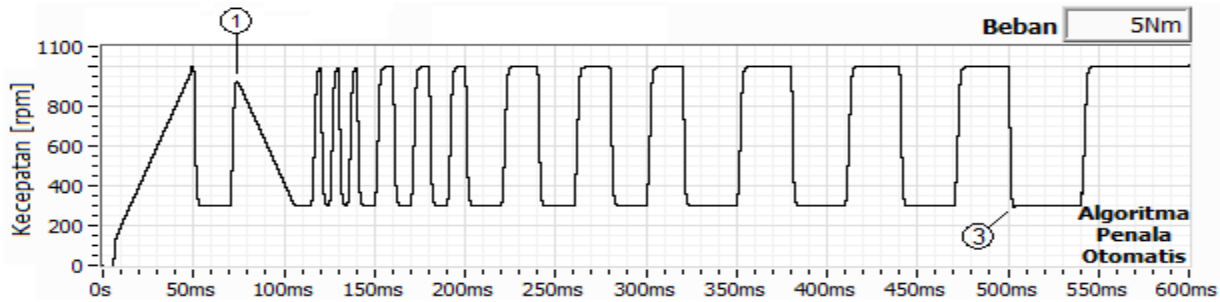
Pengujian pada skenario keempat belas ini adalah menganalisa metode *trial&error* dan algoritma penala otomatis pada kecepatan dinamis dengan beban menengah yaitu 5Nm. Berikut point yang dianalisa:

1. Serupa pada skenario ke13 poin pertama pada pengujian kecepatan dinamis dengan beban nol sebelumnya, pada skenario ini kedua metode tersebut tidak dapat mencapai angka 1000rpm pada poin 1 dalam Gambar 30.

2. Tidak seperti pada skenario ke 13 poin kedua, pada beban menengah, metode *trial&error* pada titik yang ditunjukan oleh poin 2 pada Gambar 30 tidak mampu mencapai angka yang seharusnya yaitu 1000rpm. Karena pada pengujian ini telah menggunakan beban, maka respon mengalami kondisi *deadtime*, 10ms pada penggunaan metode *trial&error* dan 7ms pada penggunaan algoritma tuning otomatis.

3. Penggunaan metode algoritma penala otomatis secara random akan mengalami kondisi *undershoot* meski bernilai kecil yang ditunjukan pada point 3 Gambar 30.



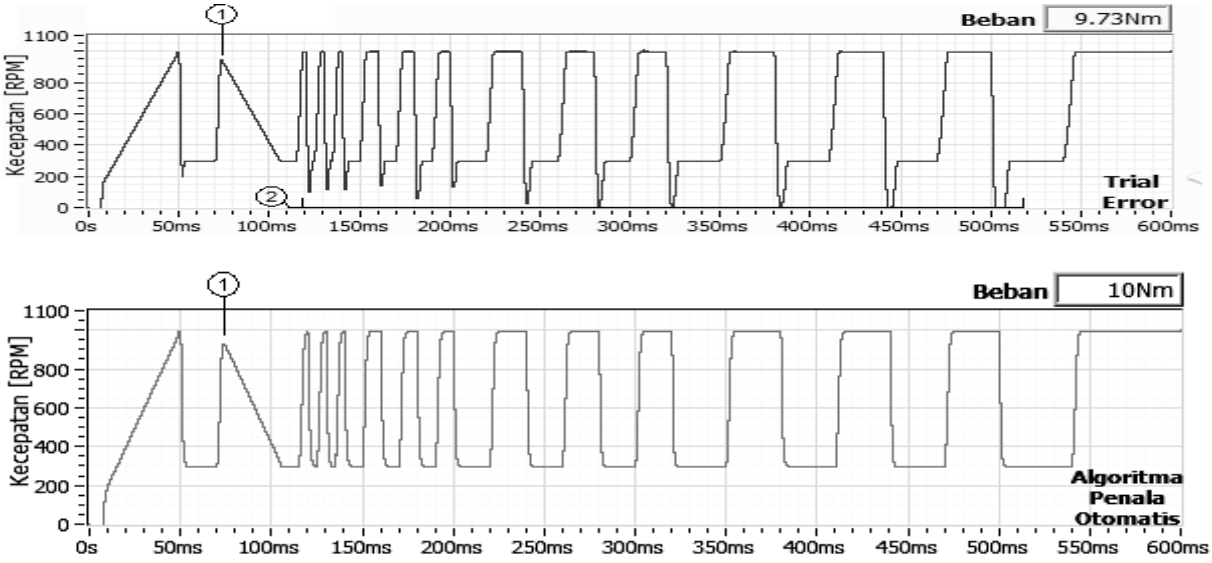


Gambar 30: Respon kecepatan pada metode *trial&error* dan algoritma penala otomatis pada *setpoint* dinamis dengan beban 5Nm

3.15.Skenario ke15: Variasi kecepatan dengan beban menengah (5Nm)

Pengujian metode *trial&error* dan algoritma penala otomatis pada pengujian dengan kecepatan dinamis dengan beban berat yaitu 10Nm memiliki beberapa poin analisa yaitu:

1. Serupa dengan pengujian pada skenario ketiga belas dan skenario keempat belas, pada skenario ini, baik pada penggunaan metode *trial&error* maupun pengunaan algoritma penala otomatis tetap tidak dapat mencapai angka 1000rpm pada titik poin nomor satu pada Gambar 31.
2. Penggunaan metode *trial&error* telah mengalami keadaan *undershoot*. Hal ini berbeda dengan penggunaan algoritma penala otomatis masih konsisten memenuhi persyaratan pada Tabel 1.



Gambar 31: Respon kecepatan pada metode *trial&error* dan algoritma penala otomatis pada *setpoint* dinamis dengan beban 10Nm

TABEL 7
ANALISA HASIL PENGGUNAAN METODE *TRIAL&ERROR*

Kecepatan / beban	Rendah (500rpm)	Menengah (750rpm)	Tinggi (1000rpm)
Nol (0Nm)	Memenuhi syarat	Memenuhi syarat	Memenuhi syarat
Menengah (5Nm)	Hampir memenuhi syarat	Memenuhi syarat	Hampir memenuhi syarat
Berat (10Nm)	Tidak memenuhi syarat	Tidak memenuhi syarat	Hampir memenuhi syarat

TABEL 8
ANALISA HASIL PENGGUNAAN ALGORITMA PENALA OTOMATIS

Kecepatan / beban	Rendah (500rpm)	Menengah (750rpm)	Tinggi (1000rpm)
Nol (0Nm)	Memenuhi syarat	Memenuhi syarat	Memenuhi syarat
Menengah (5Nm)	Memenuhi syarat	Memenuhi syarat	Memenuhi syarat
Berat (10Nm)	Memenuhi syarat	Memenuhi syarat	Memenuhi syarat

Analisa mulai dari skenario keempat hingga skenario kedua belas dirangkum kedalam tabel yang disajikan pada Tabel 7 dan Tabel 8. Persyaratan yang dimaksud adalah peryaratan yang telah ditentukan sebelumnya pada Tabel 1. Kontroler PID dengan menggunakan metode *trial&error* tidak mampu memenuhi persyaratan tersebut disegala kondisi yang diuji, sementara persyaratan tersebut dapat dipenuhi oleh penggunaan algoritma penala otomatis.

4. Kesimpulan

Berdasarkan hasil simulasi dan analisa yang telah dilakukan dapat ditarik kesimpulan bahwa desain Penala Parameter PID Otomatis Pada Pengatur Kecepatan Motor Induksi Tiga Fasa mampu menghasilkan karakteristik performa kecepatan dinamik yang lebih baik dari penggunaan metode *trial&error* dengan tolak ukur dari performa masa transien berupa *risetime* kurang dari 20ms, *error steadystate*, *overshoot* dan *undershoot* masing-masing maksimum bernilai ±2%.

Terima kasih

Penulis mengucapkan terima kasih kepada grup riset laboratorium mesin listrik dan kontrol Politeknik Elektronika Negeri Surabaya yang telah menyediakan segala bantuan dan panduan selama melaksanakan kegiatan penelitian ini.

Daftar Pustaka

[1] B. Praharsena, E. Purwanto, Jaya. Arman, et al., “Evaluation of Hysteresis Loss Curve on 3 Phase Induction Motor by Using Cascade Feed Forward Neural Network,” in 2018 International Electronics Symposium on Engineering Technology and Applications, IES-ETA 2018 - Proceedings, 2019.

[2] I. Boldea, L. N. Tutelea, L. Parsa, and D. Dorrell, “Automotive electric propulsion systems with reduced or no permanent magnets: An overview,” IEEE Trans. Ind. Electron., 2014.

[3] M. A. Hernandez, J. M. G. Villalobos, S. M. Soldara, F. M. Mondragon, and J. R. Resendiz, “A speed performance comparative of field oriented control and scalar control for induction motors,” in 2016 IEEE Conference on Mechatronics, Adaptive and Intelligent Systems, MAIS 2016, 2016.

[4] A. W. Aaditya, D. C. Happyanto, and B. Sumantri, “Application of Sliding Mode Control in Indirect Field Oriented Control (IFOC) for Model Based Controller,” Emitter International Journal of Enginerring. Technology, 2018.

[5] J. Yu, T. Zhang, and J. Qian, Electrical motor products: International energy-efficiency standards and testing methods. 2011.

[6] A. T. De Almeida, F. J. T. E. Ferreira, and G. Baoming, “Beyond induction motors - Technology trends to move up efficiency,” IEEE Trans. Ind. Appl., 2014.

[7] I. Ferdiansyah, E. Purwanto, and N. A. Windarko, “Fuzzy Gain Scheduling of PID (FGS-PID) for Speed Control Three Phase Induction Motor Based on Indirect Field Oriented Control (IFOC),” Emitter International Journal Engineering Technology, 2017.

[8] Nurfaizah, D. Istardi, and H. Toar, “Rancang Bangun Modul Praktikum Motor AC dengan Aplikasi Pengaturan Posisi dengan Menggunakan PID,” Jurnal Integrasi, vol. 7, no. 1, pp. 50–56, 2015.

[9] I. Ferdiansyah, M. R. Rusli, B. Praharsena, H. Toar, Ridwan, and E. Purwanto, “Speed control of three phase induction motor using indirect field oriented control based on real-time control system,” in Proceedings of 2018 10th International Conference on Information Technology and Electrical Engineering: Smart Technology for Better Society, ICITEE 2018, 2018.

[10] J. G. Ziegler, N. B. Nichols, and N. Y. Rocheiester, “Optimum Sttings for Automatic Controllers,” Transacction ASME, 1942.

[11] E. Purwanto, A. M. wibowo, Soebagio, and M. H. Purnomo, “Pengembangan metoda self tuning parameter pid controller dengan menggunakan genetic algorithm pada pengaturan motor induksi sebagai penggerak mobil listrik,” (SNATI 2009), vol. 2009, p. E-120-E-127, 2009.

[12] Alrijadjis, Shenglin Mu, Shota Nakashima, and K. Tanaka, “PID Controller Design of Nonlinear System using a New Modified Particle Swarm Optimization with Time-Varying Constriction Coefficient,” Emit. Int. J. Eng. Technol., vol. 2, no. 2, pp. 80–90, 2014.