ANALISIS KINERJA PENGENDALIAN KECEPATAN MOTOR DC MENGGUNAKAN PID DENGAN METODE TUNING *TRIAL & ERROR*, ZIEGLER-NICHOLS, DAN ALGORITMA GENETIKA

TUGAS AKHIR

Karya Ilmiah sebagai salah satu syarat untuk menyelesaikan jenjang strata satu (S-1) di Departemen Teknik Elektro, Fakultas Teknik Universitas Andalas

Oleh:

Rahma Azira Ichsan

NIM. 2010953021

Pembimbing:

Mumuh Muharam, S.T., M. T.

NIP. 196711131998031002

PROGRAM STUDI SARJANA

TEKNIK ELEKTRO FAKULTAS TEKNIK

UNIVERSITAS ANDALAS

2025

LEMBAR PENGESAHAN

Analisis Kinerja Pengendalian Kecepatan Motor DC Menggunakan PID dengan Metode *Tuning Trial & Error*, Ziegler-Nichols, dan Algoritma Genetika

Oleh

Rahma Azira Ichsan
2010953021

Departemen Teknik Elektro Fakultas Teknik
Universitas Andalas

Disetujui pada Tanggal : 23 Januari 2025

Pembimbing,

Ir. Mumuh Muharam, S.T., M.T.
NIP. 196711131998031002

Mengetahui,
Ketua Departemen Teknik Elektro

Prof. Syafii, S.T., M.T., Ph.D.
NIP. 197405051998021001

HALAMAN PENGHARGAAN

Alhamdulillah, puji syukur penulis ucapkan kepada Allah SWT atas rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan tugas akhir ini. Penulis mengucapkan terima kasih kepada semua pihak yang telah membantu dalam segala aspek, baik berupa bimbingan, bantuan, dukungan, serta dorongan baik secara moral maupun material. Pada kesempatan ini, penulis ingin menyampaikan rasa terima kasih yang mendalam kepada:

1. Kedua orang tua tercinta, Bapak Marjis (Alm) dan Ibu Darmawati, yang telah memberikan dukungan penuh dan kerja keras selama ini untuk bisa memberikan pendidikan yang layak untuk penulis. Terima kasih atas segala doa, dukungan, kasih sayang, dan pengorbanan yang telah diberikan selama ini. Bahagia disana Ayah, anak-anakmu tidak ada yang putus sekolah dan sehat selalu Ibu tercinta.
2. Kakak dan Abang tercinta, Maryami Rahmadiah, Muh. Rozi Rahmatullah, Rahmaniah Ichsan, dan Muhammad Taufiq Rahmatullah yang telah menemani masa kecil penulis hingga sekarang serta memberikan dukungan dan pengertiannya selama ini. Terima kasih Kakak dan Abang.
3. Bapak Ir. Mumuh Muharam, S.T., M.T. selaku dosen pembimbing tugas akhir yang selalu membimbing, memberikan masukan, arahan, dan nasehat kepada penulis.
4. Bapak Ir. Heru Dibyo Laksono, S.T., M.T dan Bapak Prof. Dr. Eng. Ir. Muhammad Ilhamdi Rusydi, S.T., M.T. selaku dosen penguji yang telah memberikan saran dan masukan dalam tugas akhir ini.
5. Teman – teman “Kos Niyen”, Yesi Fajri, Lara Adrosa Marjuita, Dini Meilinda,, Suci Maretta Salim, Rindina Armysa, Putri Nabila, serta adik – adik Siska Octavia dan Sindy Aprilia yang telah berbagi cerita, canda tawa, kebahagiaan, semangat, serta suka duka selama di kos.
6. Muhammad Fadli, orang yang selalu memberikan semangat dan dukungan kepada penulis dalam menyelesaikan tugas akhir ini.
7. Keluarga besar Laboratorium Kontrol Digital yang telah membantu penulis dalam proses pelaksanaan tugas akhir.
8. Teman – teman seperjuangan dalam mengerjakan tugas akhir, Farras Yufadillah, Rahmi Syafrianda, Raihan Maulana Makhlad, dan Muhammad Farhan Mendra. Terima kasih atas kebersamaan dan cerita yang tercipta agar penulis tetap semangat mengerjakan tugas akhir.
9. Keluarga besar Teknik Elektro Angkatan 2020 “Neutron” dan Himpunan Mahasiswa Teknik Elektro (HMTE) yang memberikan semangat dan motivasi untuk bisa menyelesaikan tugas akhir ini.

LEMBAR PERNYATAAN KEASLIAN

Saya yang bertanda tangan dibawah ini:

Nama : Rahma Azira Ichsan
NIM : 2010953021
Program Studi : S1-Teknik Elektro
Fakultas : Teknik
Perguruan Tinggi : Universitas Andalas

Dengan ini saya menyatakan bahwa tugas akhir yang berjudul "Analisis Kinerja Pengendalian Kecepatan Motor DC Menggunakan PID dengan Metode *Tuning Trial & Error*, Ziegler-Nichols, dan Algoritma Genetika" merupakan hasil karya asli (orisinil) atau tidak plagiat (menjiplak) dan belum pernah diterbitkan/dipublikasikan di halaman manapun dan dalam bentuk apapun, kecuali yang secara tertulis dikutip dalam naskah ini dan tercantum dalam daftar pustaka.

Demikian surat pernyataan ini saya tulis dengan sepenuh hati tanpa ada paksaan dari pihak manapun. Apabila dikemudian hari ternyata saya memberikan keterangan palsu atau ada pihak lain yang mengklaim bahwa tugas akhir yang telah saya buat adalah hasil karya milik seseorang atau badan tertentu, saya bersedia menerima sanksi sesuai dengan ketentuan yang berlaku.

Padang, 22 Januari 2025
Penulis

Rahma Azira Ichsan

RIWAYAT HIDUP

Rahma Azira Ichsan, lahir di Bukittinggi pada tanggal 20 Mei 2002, merupakan anak kelima dari lima bersaudara. Penulis menyelesaikan pendidikan sekolah dasar di SDN 05 Kubang Putiah, menamatkan pendidikan menegah di SMPN 1 Bukittinggi, dan melanjutkan Pendidikan di SMAN 1 Bukittinggi. Penulis melanjutkan Pendidikan di Program Studi S-1 Teknik Elektro, Fakultas Teknik, Universitas Andalas pada tahun 2020. Selama menempuh pendidikan di Departemen Teknik Elektro Universitas Andalas, penulis pernah mengikuti program Kerja Praktek di PT. Pertamina Hulu Rokan (PHR) tim IT OT (*Information Technology Infrastructure Operation*) dan tim IT PCN (*Information Technology Process Control Network*) pada tahun 2024. Penulis juga aktif menjadi asisten di Laboratorium Kontrol Digital dari tahun 2023 hingga tahun 2025. Penulis juga aktif dalam organisasi Himpunan Mahasiswa Teknik Elektro (HMTE) sebagai pengurus selama dua perode kepegurusan, yaitu sebagai anggota Departemen Bursa pada tahun 2022 dan anggota Departemen Kaderisasi pada tahun 2023/2024.

Judul	Analisis Kinerja Pengendalian Kecepatan Motor DC Menggunakan PID dengan Metode *Tuning Trial & Error*, Ziegler-Nichols, dan Algoritma Genetika	Rahma Azira Ichsan
Program Studi	Teknik Elektro	2010953021
Fakultas Teknik Universitas Andalas		

Abstrak

Penelitian ini bertujuan untuk menganalisis kinerja pengendalian kecepatan motor DC menggunakan metode *tuning trial & error*, Ziegler-Nichols, dan algoritma genetika (GA). Motor DC dipilih karena memiliki efisiensi yang tinggi meskipun kecepatannya dapat menurun akibat beban. Untuk memperbaiki respon sistem tersebut digunakan pengendali PID dengan parameter Kp, Ki, dan Kd yang dioptimalkan melalui metode *tuning*. Penelitian dilakukan dengan melakukan simulasi fungsi alih motor DC menggunakan MATLAB. Analisis dilakukan pada domain waktu dengan parameter evaluasi meliputi waku naik (t_r), waktu keadaan mantap (t_s), lewatan maksimum (M_p), dan kesalahan keadaan mantap (ess). Hasil percobaan menunjukkan bahwa *tuning* GA dengan konfigrasi pengendali PD dan PID memberikan kinerja terbaik dengan t_r = 0,001 s, t_s = 0,002 s, M_p = 0,81% dan ess = 0,0126 untuk pengendali PD serta t_r = 0,023 s, t_s = 0,04 s, M_p = 0% dan ess = 0,0103 untuk pengendali PID. Metode *trial & error* menghasilkan t_r dan t_s tercepat sebesar 0,00554 s dan 0,01222 s, tetapi ess tidak konsisten di semua percobaan. *Tuning* Ziegler-Nichols memiliki t_s = 0,347 s dan ess = 0,0354 pada pengendali PD tetapi lewatan maksimum yang tinggi, yaitu 19,375%. Sehingga dapat disimpulkan bahwa *tuning* GA memberikan parameter pengendali yang optimal dan menghasilkan respon yang cepat sesuai dengan kriteria perancangan yang ditetapkan.

Kata kunci: Motor DC, PID, *trial and error*, Ziegler-Nichols, Algoritma Genetika

Title	*"Performance Analysis of DC Motor Speed Control Using PID with Trial & Error Tuning Method, Ziegler-Nichols, and Genetic Algorithm"*	Rahma Azira Ichsan
Mayor	*Electrical Engineering*	2010953021
Engineering Faculty Andalas University		

Abstract

This study aims to analyze the performance of DC motor speed control using trial & error, Ziegler-Nichols, and genetic algorithm (GA) tuning methods. The DC motor was chosen because it has high efficiency although its speed can decrease due to load. To improve the system response, a PID controller is used with Kp, Ki, and Kd parameters optimized through the tuning method. The research was performed by simulating the DC motor transfer function using MATLAB. The analysis was analyzed in the time domain with evaluation parameters including rise time (tr), settling time (ts), maximum overshoot (Mp), and steady state error (ess). Experimental results show that GA tuning with PD and PID controller configuration provides the best performance with tr = 0.001 s, ts = 0.002 s, Mp = 0.81% and ess = 0.0126 for PD controller and tr = 0.023 s, ts = 0.04 s, Mp = 0% and ess = 0.0103 for PID controller. The trial & error method generated the fastest tr and ts equal to 0.00554 s and 0.01222 s, but ess was inconsistent across all trials. Ziegler-Nichols tuning has ts = 0.347 s and ess = 0.0354 for the PD controller but maximum overshoot was high at 19.375%. So it can be summarized that GA tuning provides optimal controller parameters and generates a fast response in accordance with the design criteria set.

Keywords: DC Motor, PID, trial and error, Ziegler-Nichols, genetic algorithm

DAFTAR ISI

PRAKATA

Puji syukur kepada Allah SWT yang telah memberikan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan tugas akhir dengan Judul "Analisis Kinerja Pengendalian Kecepatan Motor DC Menggunakan PID dengan Metode *Tuning Trial & Error*, Ziegler-Nichols, dan Algoritma Genetika". Tugas akhir ini disusun untuk memenuhi salah satu persyaratan dalam menyelesaikan jenjang strata satu (S-1) di Departemen Teknik Elektro, Fakultas Teknik, Universitas Andalas. Dalam menyelesaikan tugas akhir ini, penulis banyak mendapat bantuan dan dorongan dari berbagai pihak.

Tugas akhir ini tersusun tidak hanya karena kerja keras penulis saja, bantuan dari pihak lain juga turut andil dalam tersusunnya tugas akhir ini. Oleh sebab itu, penulis ingin mengucapkan terima kasih kepada:

1. Bapak Prof. Syafii, S.T., M.T., Ph.D. selaku Ketua Departemen Teknik Elektro, Fakultas Teknik Universitas Andalas.
2. Bapak Ir. Primas Emeraldi, S.T., M.T. selaku Kepala Prodi S1 Departemen Teknik Elektro, Fakultas Teknik Universitas.
3. Bapak Ir. Mumuh Muharam, S.T., M.T. selaku dosen pembimbing tugas akhir yang telah membimbing memberikan arahan dan masukan pada pembuatan tugas akhir ini.
4. Bapak Prof. Dr. Eng. Ir. Ilhamdi Rusydi, S.T., M.T. dan Bapak Ir. Heru Dibyo Laksono, S.T., M.T. yang telah memberikan kritikan, saran, dan masukan terhadap penyelesaian tugas akhir ini.
5. Seluruh pihak yang telah membatu penulis dalam menyelesaikan tugas akhir ini, yang tidak dapat disebutkan satu persatu.

Penulis menyadari bahwa dalam penulisan laporan tugas akhir masih belum sempurna, baik dari cara penyajian maupun teknik kepenulisan. Oleh karena itu, penulisa menerima kritikan dan saran yang mendukung demi kesempurnaan tugas akhir ini. Semoga tugas akhir ini bermanfaat bagi kita semua terutama bagi penulis sendiri.

Padang, 23 Januari 2025

Rahma Azira Ichsan
NIM.2010953021

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR SIMBOL

$\omega_{(s)}$	:	Kecepatan Sudut
$V_a(s)$	:	Tegangan Jangkar
R_a	:	Nilai Resistansi
L_a	:	Nilai Induktansi
K_b	:	Konstanta Gaya Gerak Listik
K_t	:	Konstanta Torsi Motor
J_m	:	Momen Inersia
B_m	:	Konstanta Gesekan
L	:	Waktu Tunda
T	:	Waktu Konstan
t_r	:	*Rise Time* (Waktu Naik)
t_p	:	*Peak Time* (Waktu Puncak)
t_s	:	*Settling Time* (Waktu Keadaan Mantap)
Mp	:	*Maximum Overshoot* (Lewatan Maksimum)
ess	:	*Error Steady State* (Kesalahan Keadaan Mantap)
Kp	:	*Proportinal Gain* (Konstanta Proporsional)
Ki	:	*Integral Gain* (Konstanta Integral)
Kd	:	*Derivative Gain* (Konstanta Diferensial)
Ti	:	Konstanta Waktu Integral
Td	:	Konstanta Waktu Diferensial

DAFTAR LAMPIRAN

DAFTAR ISTILAH

Plant	:	Objek yang dikendalikan
Kriteria Perancangan	:	Nilai yang diinginkan dari hasil perancangan yang dilakukan
Fungsi Alih	:	Representasi matematis yang menggambarkan hubungan antara masukan dan keluaran suatu sistem
PID	:	Pengendali yang digunakan untuk memperbaiki performansi sistem melalui perbaikan respon sistem yang dihasilkan.
tuning	:	Proses penyetelan atau penyesuaian parameter PID (Kp, Ki, dan Kd) pada pengendali untuk memperoleh kinerja sistem kendali yang diinginkan

DAFTAR SINGKATAN

P	:	Proporsional
I	:	Integral
D	:	Diferensial
PI	:	Proporsional Integral
PD	:	Proporsional Diferensial
PID	:	Proporsional Integral Diferensial
GA	:	Algoritma Genetika

BAB I PENDAHULUAN

1.1 Latar Belakang

Motor listrik merupakan komponen penting dalam berbagai sistem otomasi industri dan robotika yang digunakan sebagai penggerak (aktuator), terutama motor listrik arus searah (DC). Motor DC merupakan perangkat yang mengubah energi listrik DC menjadi energi mekanik berupa putaran [1]. Motor DC digunakan karena memiliki kecepatan yang tinggi, efisiensi yang tinggi, dan umur pakai yang lama. Namun, terdapat kekurangan seperti penurunan kecepatan yang membuat kecepatan motor DC menjadi tidak konstan akibat beban yang ada dan pengaruh dari besaran nilai masukan tegangan [2].

Untuk meningkatkan kecepatan motor DC, nilai masukan tegangannya dapat dinaikkan. Untuk itu, digunakan pengendali yang dapat mengatur kecepatan motor DC seperti pengendali PID yang mengatur sistem untuk mencapai kestabilan dan memperbaiki respon sistem yang diinginkan. Pengendali PID memiliki tiga parameter kontrol, yaitu *Proportional* (P), *Integral* (I), dan *Derivative* (D) yang masing-masingnya memiliki keunggulan tertentu [3].

Tuning PID merupakan sebuah penyesuaian yang dilakukan untuk mengoptimalkan kinerja pengendali PID itu sendiri. Tujuan utamanya adalah menemukan nilai-nilai parameter yang tepat untuk Kp, Ki, dan Kd agar nilai respon sistem sesuai dengan spesifikasi yang diingikan [4].

Beberapa penelitian sebelumnya yang membahas mengenai kontrol kecepatan motor DC dengan PID di antaranya:

- Iqlimah Khadari, dkk [5] melakukan penelitian yang menganalisa penggunaan pengendali PID dengan metode *self-tuning* Fuzzy PID dan Algoritma Genetika (GA) pada simulasi pengendali kecepatan motor DC. Pada penelitian ini didapatkan bahwa pengendalian yang dilakukan dengan kedua metode *tuning* mampu mengarahkan *output* sesuai dengan yang diinginkan dan memiliki persentase *overshoot* yang masih berada dalam batas ketentuan umum yang diperbolehkan.
- Mila Diah Ika Putri, dkk [1] melakukan penelitian untuk menguji kecepatan motor DC dengan pengendali PID menggunakan metode *tuning trial & error* dan Ziegler-Nichols. Metode pengujian yang digunakan pada penelitian ini menggunakan simulasi dan implementasi *hardware* menggunakan mikrokontroler Arduino. Berdasarkan hasil pengujian, didapatkan bahwa metode *tuning* Ziegler-Nichols menghasilkan keluaran dengan grafik kecepatan motor DC yang stabil dibandingkan dengan metode *trial & error* dikarenakan nilai *overshoot* paling rendah yang didapatkan

menggunakan metode tersebut adalah sebesar 12,11% sehingga sistem belum stabil.

- Muhammad Irhas, dkk [4] melakukan *review* penelitian mengenai penggunaan pengendali PID dengan berbagai metode untuk mengendalikan kecepatan motor DC dari beberapa penelitian yang telah ada sebelumnya. Metode yang digunakan adalah metode *Root-Locus*, metode Algoritma Genetika (GA), metode heuristik, dan metode *Self tuning PID Fuzzy Logic Controller* untuk memperoleh respon keluaran terbaik. Diantara 4 pengujian metode tuning PID, metode heuristik diimplementasikan langsung pada *hardware* motor DC dan didapat kesimpulan bahwa metode tuning *Fuzzy Logic Controller* menghasilkan keluaran *rise time* (tr), *settling time* (ts), *steady state error* (ess), dan *overshoot* yang lebih kecil dibandingkan dengan metode lainnya. [4].

Berdasarkan beberapa penelitian yang telah dilakukan tersebut, dilakukan penelitian yang akan membahas analisis kinerja pengontrolan kecepatan motor DC menggunakan beberapa metode *tuning* PID. Metode *tuning* yang akan digunakan pada penelitian ini adalah metode *trial & error*, metode Ziegler-Nichols, dan metode algoritma genetika (GA). Ketiga metode ini digunakan sebagai representasi dari beberapa klasifikasi metode tuning PID, yaitu metode heuristik yang direpresentasikan oleh metode *trial & error*, metode klasik yang direpresentasikan oleh metode Ziegler-Nichols, dan metode metaheuristik yang direpresentasikan oleh metode algoritma genetika [6]. Analisis kinerja pengontrolan kecepatan motor DC menggunakan ketiga metode *tuning* PID ini dinilai dan dibandingkan untuk mendapatkan respon terbaik sesuai yang diinginkan berdasarkan tanggapan domain waktu dengan parameter yang digunakan adalah waktu naik (tr), waktu tunak (ts), lewatan maksimum (Mp), dan kesalahan keadaan tunak (ess).

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan di atas maka rumusan masalah penelitian ini adalah bagaimana pengaruh pengaruh metode *tuning* PID, yaitu metode *trial & error*, metode Ziegler-Nichols, dan metode Algortima Genetika (GA) terhadap kinerja pengendalian kecepatan motor DC serta bagaimana perbandingan analsis peralihan dan analisis kesalahan yang dihasilkan dari masing-masing metode *tuning* PID yang digunakan?

1.3 Tujuan Penelitian

Tujuan dari tugas akhir ini adalah untuk menganalisis pengaruh metode tuning PID, yaitu metode *trial & error*, metode Ziegler-Nichols, dan metode Algortima Genetika (GA) terhadap kinerja pengendalian kecepatan motor DC serta untuk membandingkan analisis peralihan dan analisis kesalahan yang dihasilkan dari masing-masing metode *tuning* yang digunakan.

1.4 Manfaat Penelitian

Manfaat dari penulisan tugas akhir ini yakni memberikan pemahaman terhadap beberapa metode *tuning* PID dalam pengendalian kecepatan motor DC. Melalui analisis peralihan dan analisis kesalahan yang diperoleh, hasil penelitian ini dapat menjadi referensi dalam menentukan metode *tuning* PID dengan kinerja terbaik untuk aplikasi pengendalian kecepatan motor DC.

1.5 Batasan Masalah

Penelitian dan penulisan tugas akhir ini dibatasi pada hal-hal berikut:

1. Metode *tuning* PID yang akan dibandingkan pada penelitian ini adalah metode *trial and error*, metode Ziegler-Nichols, dan metode algoritma genetika (GA).
2. Metode *tuning trial and error* dilakukan sebanyak lima kali dengan nilai Kp, Ki, dan Kd yang berbeda-beda.
3. Metode *tuning* Ziegler-Nichols yang digunakan adalah metode kurva reaksi.
4. Metode *tuning* algoritma genetika (GA) digunakan untuk meminimalkan *error* antara *setpoint* dan keluaran sistem.
5. Parameter yang digunakan untuk membandingkan beberapa metode *tuning* ini adalah waktu naik (tr), waktu tunak (ts), lewatan maksimum (Mp) dan kesalahan keadaan tunak (ess).
6. Fungsi alih yang digunakan dalam penelitian ini diperoleh dari studi sebelumnya yang telah dipublikasikan oleh Abdulameer, dkk [7].
7. Penelitian ini tidak membahas aspek implementasi teknis atau perangkat keras fisik pada pengendalian motor DC tetapi akan berfokus pada analisis komputer dengan menggunakan perangkat lunak MATLAB.

1.6 Sistematika Penulisan

Laporan akhir ini disusun dalam beberapa bab dengan sistematika tertentu, sistematika laporan ini adalah sebagai berikut:

BAB I PENDAHULUAN

Bab ini membahas tentang latar belakang dari masalah dalam pembuatan tugas akhir ini, tujuan yang ingin dicapai, manfaat, batasan masalah, dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Bab ini membahas teori-teori pendukung yang digunakan dalam penyelesaian masalah dalam tugas akhir ini.

BAB III METODOLOGI PENELITIAN

Bab ini berisikan informasi mengenai metodologi penelitian yang digunakan berupa metode penelitian, *flowchart* (diagram alir) penelitian, peralatan, dan bahan penelitian yang digunakan.

BAB IV HASIL DAN PEMBAHASAN

Bab ini memberi informasi hasil dan pembahasan mengenai hasil penelitian.

BAB V PENUTUP

Bab ini berisi kesimpulan dari hasil dan pembahasan penelitian dan saran untuk penelitian selanjutnya.

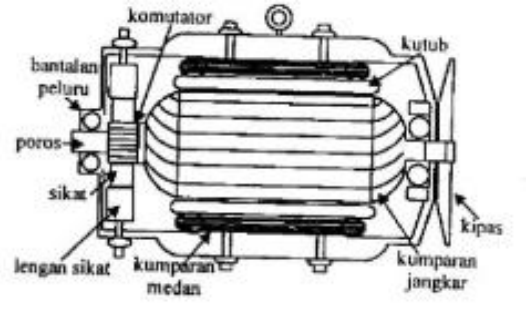
BAB II TINJAUAN PUSTAKA

2.1 Motor DC

Motor DC (*Direct Current*) merupakan alat yang mengubah energi listrik arus searah menjadi energi mekanik. Motor listrik digunakan untuk mengubah daya listrik menjadi daya mekanik. Motor DC dan generator searah sangat mirip pengoperasiannya karena kenyataannya mesin yang bekerja sebagai generator arus searah akan dapat bekerja sebagai motor arus searah [8].

2.1.1 Konstruksi Motor DC

Konstruksi dari motor DC dapat dilihat pada Gambar 2.1 [8].



Gambar 2. 1 Konstruksi Motor DC

- Badan motor: tempat meletakkan sebagian besar komponen mesin dan melindungi bagian mesin.
- Kutub: berfungsi untuk menahan kumparan medan di tempatnya, dan menghasilkan distribusi flus magnet yang lebih merata di seluruh jangkar dengan menggunakan permukaan yang melengkung.
- Inti jangkar: terbuat dari bahan *ferromagnetik* agar komponen-komponen (lilitan jangkar) terletak dalam daerah yang induksi magnetnya besar dan GGL induksi dapat bertambah besar.
- Kumparan: tempat dibangkitkannya GGL induksi
- Kumparan medan: merupakan susunan konduktor yang dibelitkan pada inti kutub. Rangkaian medan yang berfungsi untuk menghasilkan fluks utama dibentuk dari kumparan pada setiap kutub.
- Komutator: terdiri dari sejumlah segmen tembaga yang berbentuk lempengan-lempengan yang dirakit ke dalam silinder yang terpasang pada poros. Tiap-tiap lempengan terisolasi dengan baik satu sama lainnya.

- Sikat: berfungsi sebagai jembatan bagi aliran arus ke kumparan jangkar. Permukaan sikat ditekan ke permukaan segmen komutator untuk menyalurkan arus listrik.
- Celah udara: berfungsi sebagai temapat mengalirnya fluks yang dihasilkan oleh kutub-kutub medan.

2.1.2 Prinsip Kerja Motor DC

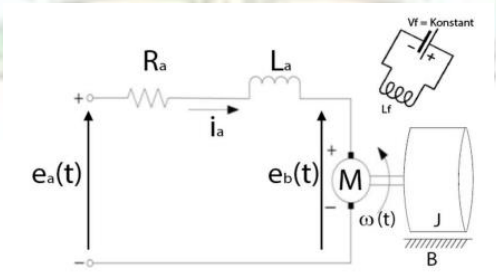
Motor DC memiliki ukuran dan kekuatan yang berbada tergantng kebutuhan pengguna. Meskipun begitu fungsi dasar dari motor DC tetap sama, yaitu mengubah energi listrik menjadi energi mekanik. Energi mekanik tersebut digunakan untuk memutar rotor. Kumparan medan pada motor DC disebut stator (bagian yang idak berputar) dan kumparan jangkar disebut rotor (bagian yang berputar). jika arus lewat pada suatu konduktor maka akan timbul medan magnet di sekitar konduktor. Pada motor DC, daerah kumparan medan yang dialiri arus listrik akan menhasilkan medn magnet yang melingkupi kumparan jangkar dengan arah tertentu. Konversi energi listrik menjasi energi mekanik dan sebaliknya berlangsung melalui medan magnet.

Sederhananya, saat arus mengalir pada kumparan jangkar (rotor) maka akan menghasilkan induksi magnetic. Dengan adanya induksi magnetic, maka akan terjadi interaksi dengan magnet permanen yang ada pada stator. Rotor yang mengalami polaritas akan membuat kutub-kutubnya berubah-ubah (dari N ke S da sebaliknya) sehingga terjadi Gerakan tolak menolak yag membuat rotor berputar.

Agar proses perubahan energi mekanik berlangsung sempurna, maka tegangan smber harus lebih besar daripada tegangan gerak yang disebabkan oleh reaksi lawan [9].

2.1.3 Fungsi Alih Motor DC

Motor DC yang dikendalikan oleh pengendali tertentu perlu diketahui karakteristik-karakteristik dari motor tersebut. Pemodelan sistem dilakukan untuk mengetahui model matematika dari suatu sistem yang akan dikendalikan. Pada motor DC biasanya modelnya ditentukan dalam bentuk fungsi alih (*transfer function*).



Gambar 2. 2 Model Fisik Motor DC

Gambar 2.2 menunjukkan model fisik rangkaian motor. Berdasarkan gambar tersebut, diketahui persamaan untuk semua tegangan yang bekerja pada motor DC ditunjukkan oleh persamaan 2.1

$$V_a(t) = R_a I_a(t) + L_a \frac{dI_a}{dt} + E_b(t) \quad \textbf{(2.1)}$$

Hubungan antara tegangan dan kecepatan sudut yang dihasilkan oleh motor DC ditunjukkan oleh persamaan 2.2.

$$E_b(t) = K_b \omega(t) \quad \textbf{(2.2)}$$

Nilai E_b pada persamaan 2.2 disubstitusikan ke persamaan 2.1 sehingga menghasilkan persamaan 2.3.

$$V_a(t) = R_a I_a(t) + L_a \frac{dI_a}{dt} + K_b \omega(t) \quad \textbf{(2.3)}$$

Nilai torsi (T_m) yang dihasilkan motor DC ditentukan menggunakan persamaan 2.4 berikut.

$$T_m(t) - T_l(t) = J_m \frac{d\omega(t)}{dt} + B_m \omega(t) \quad \textbf{(2.4)}$$

Nilai $T_l(t)$ yang merupakan nilai torsi beban bernilai $T_l(t) = 0$ untuk analisis fungsi alih, sehingga persamaannya menjadi persamaan 2.5 berikut.

$$T_m(t) = J_m \frac{d\omega(t)}{dt} + B_m \omega(t) \quad \textbf{(2.5)}$$

Hubungan antara nilai torsi dengan jumlah arus jangkar yang digunakan dapat dilihat melalui persamaan 2.6 berikut.

$$T_m(t) = K_t I_a(t) \quad \textbf{(2.6)}$$

Substitusikan nilai T_m pada persamaan 2.6 persamaan 2.5 sehingga menghasilkan persamaan 2.7 berikut.

$$K_t I_a(t) = J_m \frac{d\omega(t)}{dt} + B_m \omega(t) \quad \textbf{(2.7)}$$

Persamaan 2.7 jika dinyatakan dalam bentuk $I_a(t)$ sebagai fungsi dari $\omega(t)$ dan turunanya dinyatakan seperti pada persamaan 2.8.

$$I_a(t) = \frac{J_m}{K_t}\frac{d\omega(t)}{dt} + \frac{B_m}{K_t}\omega(t) \quad \textbf{(2.7)}$$

Substitusikan persamaan 2.7 ke persamaan 2.3 sehingga didapatkan nilai pada persamaan 2.8 dan 2.9 berikut.

$$V_a(t) = R_a\left(\frac{J_m}{K_t}\frac{d\omega(t)}{dt} + \frac{B_m}{K_t}\omega(t)\right) + L_a \frac{d}{dt}\left(\frac{J_m}{K_t}\frac{d\omega(t)}{dt} + \frac{B_m}{K_t}\omega(t)\right) + K_b\omega(t) \quad \textbf{(2.8)}$$

$$V_a(t) = \frac{R_a J_m}{K_t}\frac{d\omega(t)}{dt} + \frac{R_a B_m}{K_t}\omega(t) + L_a \frac{d}{dt}\left(\frac{J_m}{K_t}\frac{d\omega(t)}{dt}\right) + L_a \frac{d}{dt}\left(\frac{B_m}{K_t}\omega(t)\right) + K_b\omega(t) \quad \textbf{(2.9)}$$

Nilai turunan $L_a \frac{d}{dt}$ pada persamaan 2.9 ditunjukkan oleh persamaan 2.10 dan 2.11 berikut.

$$L_a \frac{d}{dt}\left(\frac{J_m}{K_t}\frac{d\omega(t)}{dt}\right) = \frac{L_a J_m}{K_t}\frac{d^2\omega(t)}{dt^2} \quad \textbf{(2.10)}$$

$$L_a \frac{d}{dt}\left(\frac{B_m}{K_t}\frac{\omega(t)}{dt}\right) = \frac{L_a B_m}{K_t}\frac{d\omega(t)}{dt} \quad \textbf{(2.11)}$$

Persamaan 2.10 dan 2.11 jika disubstitusikan ke persamaan 2. 9 menghasilkan persamaan 2.12.

$$V_a(t) = \frac{L_a J_m}{K_t}\frac{d^2\omega(t)}{dt^2} + \left(\frac{R_a J_m}{K_t} + \frac{L_a B_m}{K_t}\right)\frac{d\omega(t)}{dt} + \left(\frac{R_a B_m}{K_t} + K_b\right)\omega(t) \quad \textbf{(2.12)}$$

Gunakan transformasi *Laplace* pada persamaan 2.12 sehingga dapat ditulis menjadi persamaan 2.13.

$$V_a(t) = \frac{L_a J_m}{K_t}s^2\omega(s) + \left(\frac{R_a J_m}{K_t} + \frac{L_a B_m}{K_t}\right)s\,\omega(s) + \left(\frac{R_a B_m}{K_t} + K_b\right)\omega(t) \quad \textbf{(2.12)}$$

Sederhanakan persamaan 2.13 sehingga didapatkan fungsi alih motor DC seperti pada persamaan 2.14.

$$\frac{\omega(s)}{V_a(s)} = \frac{K_t}{L_a J_m s^2 + (R_a J_m + L_a B_m)s + (R_a B_m + K_t K_b)} \quad \textbf{(2.14)}$$

Dimana,

R_a = Nilai resistansi
L_a = Nilai induktansi
K_b = Konstanta gaya gerak listrik
K_t = Konstanta torsi pada motor
J_m = Momen inersia rotor
B_m = Konstanta gesekan

2.2 Sistem Kendali

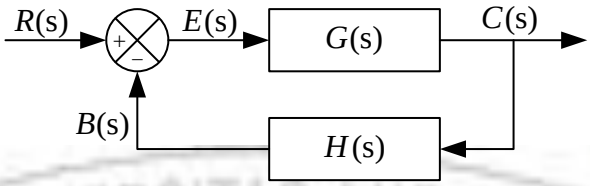
Hubungan antara komponen yang membentuk konfigurasi sistem yang akan memberikan tanggapan sistem sesuai dengan yang diinginkan disebut sistem kendali. Agar dapat menganalisis dan merancang sistem kendali, dibutuhkan model fisis, sistem fisis. Model fisis harus dapat mengambarkan karakteristik dinamis sistem dengan baik sehingga dari model fisis dapat diturunkan model matematisnya. Model matematis merupakan hubungan matematik antara keluaran (*output*) dengan masukan (*input*) [10].

Dalam teori kontrol, fungsi alih digunakan untuk mencirikan hubungan masukan dan keluaran dari komponen atau sistem yang dapat digambarkan dengan persamaan diferensial linear, invarian-waktu. Dapat juga didefenisikan sebagai perbandingan transformasi Laplace keluaran (fungsi tanggapan) terhadap transformasi Laplace masukan (fungsi penentu) dengan anggapan bahwa semua syarat awal nol. Bentuk persamaan fungsi alih dapat dilihat pada persamaan 2.15 berukut [11]:

$$G_{(s)} = \frac{\mathcal{L}[output]}{\mathcal{L}[input]} \quad \textbf{(2.15)}$$

Untuk menyatakan sistem kendali digunakan sebuah diagram yang disebut digram blok. Diagram blok dapat menunjukkan fungsi dari komponen-komponen dalam sistem kendali dengan menggambarkan hubungan antar berbagai elemen yang digunakan. Untuk merepresentasikan sistem pada diagram blok digunkana beberapa blok, sinyal direpresentasikan sebagai panah, dan titik penjumlahan

(*summing point*), dan titik percabangan (*branch point*). Diagram blok lingkar tertutup (*close loop*) dapat dilihat pada Gambar 2.3 berikut:



Gambar 2. 3 Diagram Blok Sistem Kendali Lingkar Tertutup

Berdasarkan diagram blok pada Gambar 2.3, sinyal umpan balik (*feedback)* yang dikirimkan ke titik penjumlahan sebagai perbandingan dapat dinyatakan oleh persamaan 2.16.

$$B_{(S)} = H_{(s)}C_{(s)} \quad \textbf{(2.16)}$$

Perbandingan antara sinyal *feedback* B(s) dengan sinyal kesalahan (*error*) E(s) disebut dengan fungsi alih lingkar terbuka (open-loop) yang dinyatakan persamaan 2.17.

$$\frac{B(s)}{E(s)} = G(s)H(s \quad \textbf{(2.17)}$$

Hasil perbandingan antara *output* C(s) terhadap sinyal *error* E(s) disebut sebagai fungsi alih umpan maju (*feedforward*) yang ditunjukkan oleh persamaan 2.18.

$$\frac{C(s)}{E(s)} = G(s) \quad \textbf{(2.18)}$$

Hubungan antara keluaran C(s) dan masukan R(s) dari sistem pada Gambar 2.3 dinyatakan oleh persamaan 2.19, 2.20, dan 2.21 berikut.

$$C(s) = G(s)E(s) \quad \textbf{(2.19)}$$

$$E(s) = R(s) - B(s) \quad \textbf{(2.20)}$$

$$E(s) = R(s) - H(s)C(s) \quad \textbf{(2.21)}$$

Berdasarkan persamaan 2.18 diperoleh fungsi alih lingkar tertutup yang dinyatakan oleh persamaan 2.22, 2.23, dan 2.24 berikut.

$$C(s) = G(s)[R(s) - H(s)C(s)] \quad \textbf{(2.22)}$$

$$C(s) = G(s)R(s) - H(s)C(s)G(s) \quad \textbf{(2.23)}$$

$$\frac{C(s)}{R(s)} = \frac{G(s)}{1+G(s)R(s)} \quad \textbf{(2.24)}$$

2.3 Pengendali PID (Proportional, Integral, Diferensial)

Sistem pengendali merupakan suatu sistem yang ditujukan untuk meningkatkan kinerja dan efisiensi suatu sistem. Pengendali dasar terdiri dari tiga komponen yang dapat divariasikan, yaitu proporsional (P), integral (I), dan diferensial (D). Setiap komponen pengendali ini memiliki peran khusus dan keunggulan dalam mengendalikan suatu sistem. Tabel 2.1 menunjukkan detail variasi komponen masing-maisng pengendali [12], [13].

Tabel 2.1 Respon perubahan Konstanta Pengendali

Pengendali	Waktu Naik	Lewatan Maksimum	Waktu Keadaan Mantap	Keasalahan Keadaan Mantap
Kp	Menurun	Meningkat	Perubahan Kecil	Menurun
Ki	Menurun	Meningkat	Meningkat	Hilang
Kd	Perubahan Kecil	Menurun	Menurun	Perubahan Kecil

Persamaan nilai *output* dalam sistem kendali PID dapat dilihat pada persamaan 2.25 berikut.

$$U(t) = Kp\, e(t) + Ki \int_0^t e(\tau)d\tau + Kd \frac{d}{dt} e(t) \quad \textbf{(2.25)}$$

Persamaan 2.25 menjelaskan nilai keluaran U(t) merupakan jumlah dari *gain* proporsional (Kp), *gain* integral (Ki), dan *gain* derivatif (Kd) yang masing-masing dipengarui oleh *error* (e) dan waktu (t) tertentu [4].

2.3.1 Pengendali Proporsional (P)

Pengendali proporsional adalah penguat sinyal pada suatu sistem yang disesuaikan. Antara sinyal *error* dan sinyal *output* pada pengendali dirumuskan menggunakan persamaan dalam domain waktu (t). Selanjutnya, persamaan pengendali proporsional dalam domain waktu ditransformaikan menggunakan transformasi Laplace sehingga didapat persamaan dalam domain S. Persamaan pengendali proporsional dalam domain waktu dapat dilihat pada persamaan 2.26 berikut [14].

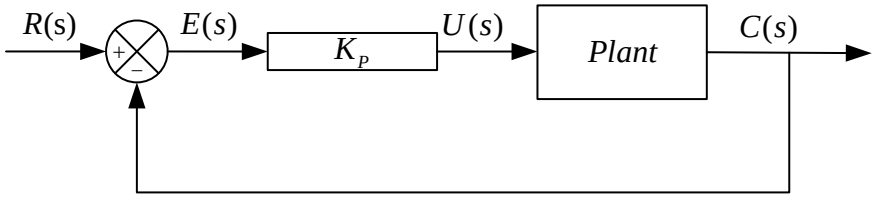
$$\frac{du(t)}{dt} = Ki \cdot e(t) \quad \textbf{(2.26)}$$

Persamaan 2.26 kemudian ditransformasikan ke domain S seperti yang terlihat pada persamaan 2.27, 2.28.

$$U(s) = KpE(s) \quad \textbf{(2.27)}$$

$$\frac{U(s)}{E(s)} = Kp \quad \textbf{(2.28)}$$

Diagram blok pengendali proporsional ditunjukkan oleh Gambar 2.4 berikut.



Gambar 2. 4 Diagram Blok Pengendali Proporsional

2.3.2 Pengendali Integral (I)

Pengendali integral merupakan pengendali yang mempertimbangkan akumulasi *error* dari waktu ke waktu dan memberikan sinyal *output* yang berbanding lurus dengan integral dari *error* tersebut. Pengendali integral dapat memperkecil nilai *overshoot* dan mempercepat *stedy-state*. Persamaan pengendali integral dapat dilihat pada persamaan 2.29 dan 2.30 yang kemudian ditansformasikan ke domain S menjadi persamaan 2.31 dan 2.32 [14].

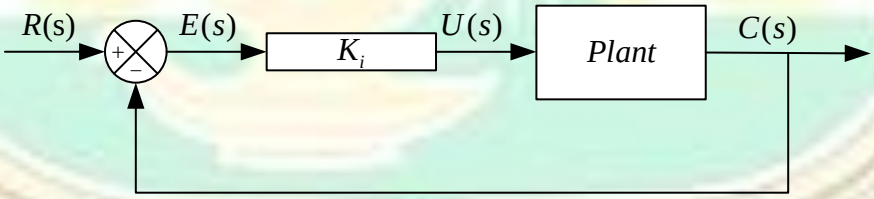
$$\frac{du(t)}{dt} = Ki \cdot e(t) \quad \textbf{(2.29)}$$

$$U(t) = Ki \int_0^t e(t)dt \quad \textbf{(2.30)}$$

$$U(s) = \frac{Ki}{s} \cdot E(s) \quad \textbf{(2.31)}$$

$$\frac{U(s)}{E(s)} = \frac{Ki}{S} \quad \textbf{(2.32)}$$

Diagram blok dari pengendali integral ditunjukkan oleh Gambar 2.5.



Gambar 2. 5 Diagram Blok Pengendali Integral

2.3.3 Pengendali Diferensial (D)

Pengendali diferensial merupakan pengendali yang memiliki sifat seperti diferensial sehingga memberikan sinyal *output* berdasarkan laju perubahan *error* terhadap waktu. Pengendali diferensial membuat sinyal kontrol (*u*) sama dengan laju perubahan sinyal *error*, seperti yang dinyatakan oleh persamaan 2.33 dan 2.34 [14].

$$\frac{u(t)}{e(t)} = Kd\frac{d}{dt} \quad \textbf{(2.33)}$$

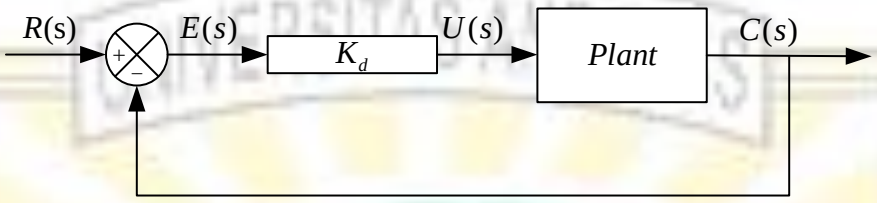
$$u(t) = Kd \int \frac{1}{e(t)} dt \quad \textbf{(2.34)}$$

Persamaan 2.35 kemudian ditransformasikan menggunakan transformasi Laplace menjadi persamaan 2.35 dan 2.36.

$$U(s) = Kd \cdot E(s) \quad \textbf{(2.35)}$$

$$\frac{U(s)}{E(s)} = Kds \quad \textbf{(2.36)}$$

Diagram blok dari pengendali diferensial dapat dilihat pada Gambar 2.6.



Gambar 2. 6 Diagram Blok Pengendali Diferensial

2.4 *Tuning* Pengendali

Terdapat suatu aspek yang penting dalam mendesain pengendali PID, yaitu menentukan parameter pengendali PID agar sistem sesuai dengan *set point* yang diinginkan. Cara untuk menentukan parameter pengendali ini disebut *tuning* pengendali. Metode tuning PID dibagi menjadi dua kategori utama, yaitu metode *closed loop* dan *open loop*. Metode *closed loop* mengangatur pengendali saat sistem beroperasi otomatis sedangkan metode *open loop* mengatur pengendali saat sistem beroperasi manual dan tanpa umpan balik [15]. Beberapa metode *tuning* PID yang digunakan diantaranya:

2.4.1 Metode *Trial & Error*

Metode *trial & error* pada *tuning* PID merupakan salah satu metode yang umum digunakan untuk menentukan parameter pengendali sistem dengan cara pengujian berulang-ulang dari nilai-nilai parameter PID. Hal ini dilakukan untuk melihat bagaimana nilai-nilai tersebut memengarui respon sistem kemudian disesuaikan berdasarkan hasil yang diamati [16]. Metode ini sederhana dan mudah digunakan, akan tetapi tidak efektif karena membutuhkan waktu yang lama dalam menemukan parameter PID yang tepat karena perubahan parameter diujikan secara manual.

Langkah-langkah *tuning* PID dengan *trial & error* adalah sebagai berikut:

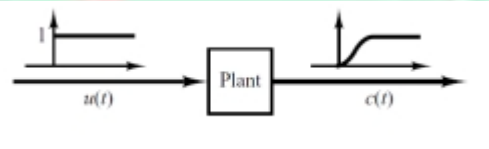
1. *Tuning* Proporsional (Kp)
   - Setel parameter Ki dan Kd ke nol.
   - Tingkatkan Kp secara bertahap hingga sistem mulai berosilasi.
   - Kurangi Kp sedikit demi sedikit hingga osilasi berkurang dan sistem stabil dengan respon sistem yang cukup cepat.
2. *Tuning* Integral (Ki)

- Setelah menemukan Kp yang sesuai, tingkatkan Ki untuk mengurangi *error steady-state.*
- Tingkatkan Ki secara bertahap sambal memantau sistem untuk mencegah terjadinya osilasi berlebihan.

3. *Tuning* Diferensial (Kd)
   - Setelah nilai Kp dan Ki di dapatkan, tambahkan nilai Kd untuk mengurangi *overshoot* dan mempercepat waktu pemulihan.
   - Tingkakan nilai Kd secara bertahap kemudian amati pengaruhnya terhadap *overshoot* dan waktu pemulihan

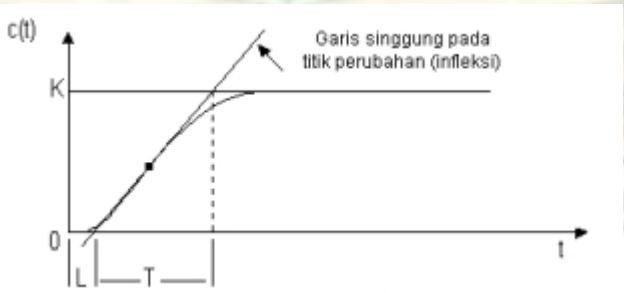
2.4.2 Metode Ziegler-Nichols

Metode *tuning* Ziegler-Nichols memiliki dua pendekatan tergantung sistem yang digunakan, yaitu metode kurva reaksi dan metode osilasi [16]. Pada metode kurva reaksi, digunakan masukan respon step dari sebuah *plant* yang ditunjukkan pada Gambar 2.7.



Gambar 2. 7 Masukan step pada *plant*

Metode kurva reaksi didasarkan terhadap reaksi sistem lingkar terbuka. *Plant* sebagai *open loop* disebut sinyal step. Jika *plant* tidak memiliki integrator atau *pole-pole* kompleks, maka reaksi sistem akan berbentuk S seperti yang ditunjukkan pada Gambar 2.8. Kurva berbentuk S memiliki dua konstanta, yaitu waktu tunda (L) dan waktu konstan (T). waktu tunda dan waktu konstan ini ditentukan dengan menggambar garis tangen pada titik infleksi kurva S dan untuk menentukan titik potong antara garis tangen dengan sumbu t dan garis c(t) = K.



Gambar 2. 8 Kurva Respon berbentuk S

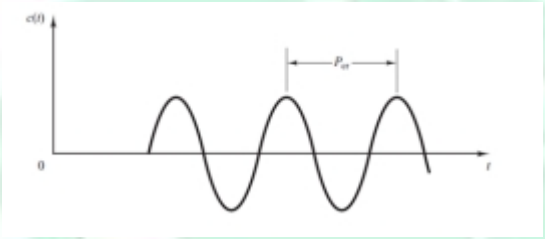
Fungsi alih pengendali PID yang diatur menggunakan metode aturan kurva reaksi Ziegler-Nichols yaitu pada persamaan 2.37 [17], [18].

$$s = \frac{-1}{L} \quad \textbf{(2.37)}$$

Tabel 2.2 Nilai Kp, Ti, dan Td berdasarkan parmeter L dan T [1]

Pengendali	Kp	Ti	Td
P	T/L	∞	0
PI	0.9T/L	L/0.3	0
PID	1.2T/L	2L	0.5L

Pada metode osilasi, osilasi yang stabil diperoleh dengan pengaruh nilai proporsional. Nilai Kp dinaikkan dari 0 ke nilai kritis Ku (*gain ultimate*) yang keluaran pertamanya akan menunjukkan osilasi yang berkelanjutan seperti ditunjukkan pada Gambar 2.9.



Gambar 2. 9 Osilasi dengan periode P_{cr}

Untuk menetapkan nilai parameter Kp, Ti, dan Td pada metode osilasi dapat menggunakan rumus pada Tabel 2.3. Pada metode osilasi, jika sistem diketahu model matemastisnya maka dapat menggunakan metode *root-locus* untuk menentukan penguatan kristis Ku dan frekuensi osilasi yang berkelanjutan ω_{cr}, dimana $\frac{2\pi}{\omega_{cr}} = P_{cr}$. Nilai-nilai tersebut dapat ditentukan dari titik potong *root-locus* dengan sumbu $j\omega$ [16], [18].

Langkah-langkah *tuning* PID dengan metode osilasi Ziegler-Nichols adalah sebagai berikut:

1. Waktu integral (Ti) diatur tak hingga ($Ti = \infty$) dan waktu derivatif diatur nol (Td =0)
2. Nilai Kp dinaikkan bertahap dari nol hingga mencapai harga respon sistem berosilasi dengan *magnitude* tetap (*sustain oscilation*).
3. Nilai penguatan Kp saat berosilasi dinamakan dengan *ultimate gain* (Ku).
4. Periode dari *sustain oscillation* dinamakan *ultimate periode* (Pu).
5. Masukkan nilai Ku dan Pu ke Tabel 2.3.

Tabel 2.3 Nilai Kp, Ti, dan Td berdasarkan parameter Ku dan Pu

Pengendali	Kp	Ti	Td
P	0.5 Ku	∞	0

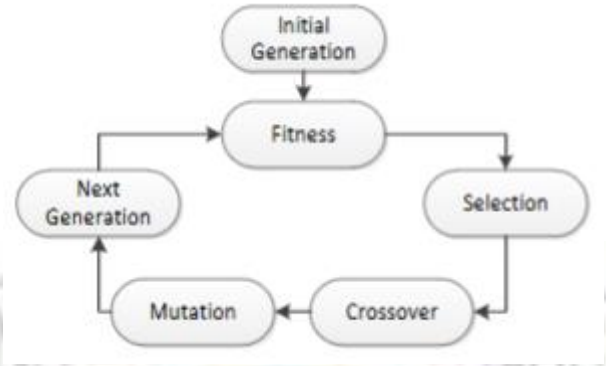
PI	0.45 Ku	1/1.2 Pu	0
PID	0.6 Ku	0.5 Pu	0.125 Pu

Metode *tuning* Ziegler-Nichols merupakan metode yang sederhana dan mudah digunakana sehingga dapat mengurangi waktu *tuning*. Akan tetapi, metode *tuning* Ziegler-Nichols lebih efektif pada sistem yang sederhana dan terbatas pada sistem yang memiliki model non-linier [17].

2.4.3 Metode Algoritma Genetika (GA)

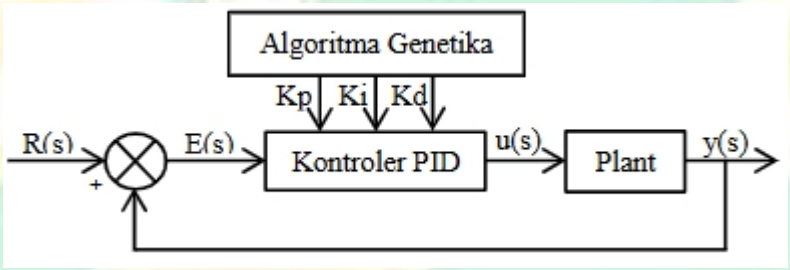
Metode algoritma genetika digunakan untuk mendapatkan nilai optimasi dari suatu permasalahan yang tidak linier. Metode GA pada pengendali PID (GA *based PID*) akan melakukan inisialisasi populasi kemudian populasi akan mengalami proses *crossover* dan mutasi [4].

Cara kerja GA dimulai dari inisiasi nilai *random* yang disebut dengan populasi dan setiap individu didalam populasi disebut kromosom. Kromosom dapat direpresentasikan sebagai calon solusi yang memliki potensi untuk menyelesaikan masalah. Kromosom kemudian akan berkembang melalui beberapa iterasi yang disebut juga generasi. Setiap generasi akan di evaluasi menggunakankan fungsi *fittnes* yang merupakan permasalahan dari setiap individu dengan solusi terbaik yang berpeluang terpilih dan bertahan di generasi selanjutnya. Selanjutnya, seleksi dilakukan untuk memilih kromosom-koromosom yang akan dijadikan orang tua (*parent*) dan membentuk kromosom baru. *Offspring* atau kromosom baru merupakan generasi selanjutnya (t+1) yang dibentuk melalui penggabungan dua kromosom generasi saat ini (t) menggunakan operator *crossover* dan memodifikasi sebuah kromosom menggunakan operator mutasi. Satu generasi dibentuk melalui proses seleksi sesuai dengan nilai *fittnes* kromosom orang tua dan kromosom *fit* yang akan diturunkan. Untuk mencegah terjadiya penurun nlai *fittnes* selama reproduksi dilkukan proses elitisme dengan menyalin kromosom dengan nilai *fittnes* tertinggi sebanyak satu atau dua. Kromosom dengan *fittnes* terbesar memiliki probabilitas tertinggi untuk di pilih. Siklus ini terjadi berulang dan diharapkan paa generasi selanjutnya memiliki kromosom dengn nilai *fittnes* yang semakin baik. Proses iterasi akan berhenti saat syarat atau tujuan yang diingikan terpenuhi [19] [20]. Siklus dari metode algoritma genetika digambarkan oleh Gambar 2.10 berikut.



Gambar 2. 10 Siklus Algoritma Genetika

Pengendali PID yang menggunakan metode GA akan melakukan optimalisasi proses *tuning* parameter PID (Kp, Ki, dan Kd). Metode optimasi GA akan mencari nilai parameter PID terbaik berdasarkan fungsi *fittnes* yaitu meminimalkan *error* antara *set point* dan nilai aktualnya. Algoritma bekerja dengan mencari nilai awal parameter secara acak dalam batasan ruang pencarian yang ditetapkan dan berhenti saat memenuhi kondisi *stopping* dan nilai parameter terbaik akan digunakan pada pengendali. Ilustrasi dari GA *based* PID ditunjukkan pada Gambar 2.11 [20].



Gambar 2. 11 Blok Diagram GA *based* PID

2.5 Analisis Sistem Kendali

2.5.1 Analisis Peralihan

Untuk mengetahui kestabilan dan akurasi suatu sistem digunakan analisa peralihan yang merupakan bagian analisis domain waktu. Analisis ini dilakukan saat kondisi transien hingga keadaan tunak (*steady state*). Ada beberapa parameter yang dibutuhkan untuk menentukan analisis peralihan, yaitu waktu naik (*rise time*), waktu puncak (*peak time*), nilai puncak, nilai lewatan maksimum, dan waktu keadaan mantap (*settling time*) [21].

Fungsi alih sistem orde satu dinyatakanoleh persamaan 2.38 berikut:

$$\frac{C(s)}{R(s)} = \frac{1}{Ts+1} \quad \textbf{(2.38)}$$

Dimana:

C(s) = fungsi *output*
R(s) = fungsi *input*
T = konstan waktu

Kemudian, fungsi alih akan diberi masukan berua undak ($R(s) = \frac{1}{s}$) satuan lalu dilakukan *invers* Laplace sehingga persamaan *ouput* dari sistem orde satu dinyatakan oleh persamaan 2.39.

$$c(t) = 1 - e^{-\frac{1}{T}} \quad \textbf{(2.39)}$$

Untuk persamaan sistem orde dua dinyatakan oleh persamaan 2.40 berikut.

$$\mathrm{G(s)} = \frac{C(s)}{R(s)} = \frac{\omega_n^2}{s^2+2\zeta\omega_n s+\omega_n^2} \quad \textbf{(2.40)}$$

Dimana,

ζ = rasio redaman
ω_n = frekuensi tidak teredam atau frekuensi natural

Semua karakteristik dari sistem orde kedua standar merupakan fungsi dari ζ dan ω_n. tanggapan peralihan sistem kendali praktis sering menunjukkan osilasi teredam sebelum mencapai keadaan mantap jika nilai $\zeta > 1$ untuk masukan undak satuan. Untuk menentukan karakteristik tanggapan peralihan sistem kendali terhadap masukan undak satuan biasanya ditentukan dengan parameter sebagai berikut:

a. Waktu naik (*rise time*)

Rise time (tr) merupakan waktu yang dibutuhkan sistem untuk menapai 10%-90%, 5%-95%, atau 0%-100% dari nilai akhir. Dibutuhkan nilai yang kecil pada tr agar menghasilkan performansi yang baik pada respon sistem kendali. Persamaan untuk menentukan parameter tr dengan satuan detik dapat dilihat pada persamaan 2.41 berikut.

$$\mathrm{t_r} = \frac{1.00+1.10\zeta+1.40\zeta^2}{\omega_n} \quad \textbf{(2.41)}$$

b. Waktu puncak (*peak time*)

Peak time (tp) merupakan waktu yang dibutuhkan sistem kendali untuk mencapai lewatan pertama. Untuk membuat respon sistem dengan performansi yang baik, maka nilai tp haruslah kecil. Persaman 2.42 menunjukkan persamaan dari tp.

$$\mathrm{t_p} = \frac{\pi}{\omega_n\sqrt{1-\zeta^2}} \quad \textbf{(2.42)}$$

c. Nilai puncak

Nilai puncak merupakan nilai respon sistem saat mencapai puncak lewatan pertama. Untuk menghitung nilai puncak dapat menggunakan rumus pada persamaan 2.43 berikut.

$$Y_p = 1.00 - \zeta\pi e\sqrt{1-\zeta^2} \quad \textbf{(2.43)}$$

d. Lewatan Maksimum

Lewatan maksmum adalah nilai puncak maksimal yang dilihat dari grafik respon yag dihitung dari nilai akhir yang ditetapkan. Apabila nilai keadaan mantap

tanggapan sistem berbeda dengan nilai akhir, dapat dilakukan perhitngan menggunakan persamaan 2.44 persamaan lewatan maksimum.

$$Mp = \frac{C(tp)-C(\infty)}{C(\infty)} \times 100\% \quad \textbf{(2.44)}$$

Terdapat juga persamaan lain dari persentase lewatan maksimum yang ditunjukkan ppersamaan 2.45 berikut.

$$Mp = 100e^{\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}} \quad \textbf{(2.45)}$$

e. Waktu keadaan mantap (*settling time*)

Setting time (ts) merupakan waktu respon sistem untuk mencapai dan berada di sekitar nilai akhir. Ukuran waktu keadaan mantap dtetapkan dalam persentase mutlak dari nilai akhir, yaitu 0.5%, 2%, atau 5%. Waktu keadaan mantap saat 2% dapat dihutung dengan persamaan 2.46 berikut.

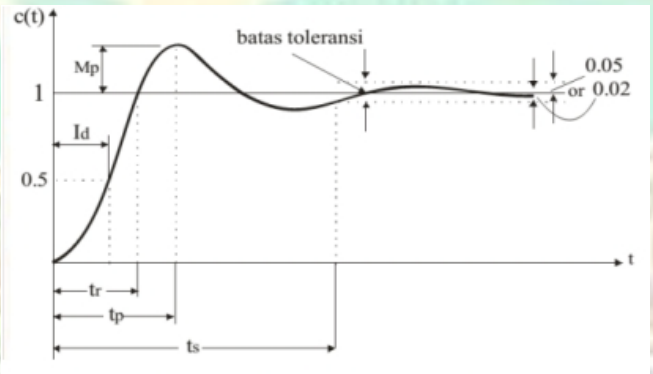
$$ts = \frac{4}{\zeta\omega_n} \quad \textbf{(2.46)}$$

Waktu keadaan mantap saat 5% dapat dihitung dengan persamaan 2.47 dan saat 0.5% dapat dihutung menggunakan persamaan 2.48.

$$ts = \frac{3}{\zeta\omega_n} \quad \textbf{(2.47)}$$

$$ts = \frac{5}{\zeta\omega_n} \quad \textbf{(2.48)}$$

Sehingga untuk mendapatkan performansi respon sistem yang baik dibutuhkan nilai ts yang kecil. Parameter-parameter dalam analsisi peralihan sistem ini dapat dilihat dari grafik tanggapan peralihan pada sistem lingkar tertutup pada Gambar 2.12 berikut.



Gambar 2. 12 Grafik Tanggapan Peralihan

2.5.2 Analisis Kesalahan

Analisis kesalahan didapat dari masukan undak satuan, laju satuan dan parabolik satuan dari sistem kendali lingkar terbuka. Dengan menggunakan analisis kesalahan bisa didapatkan informasi mengenai tipe sistem, konstanta kesalahan posisi, kecepatan, dan percepatan, serta kesalahan keadaan mantap

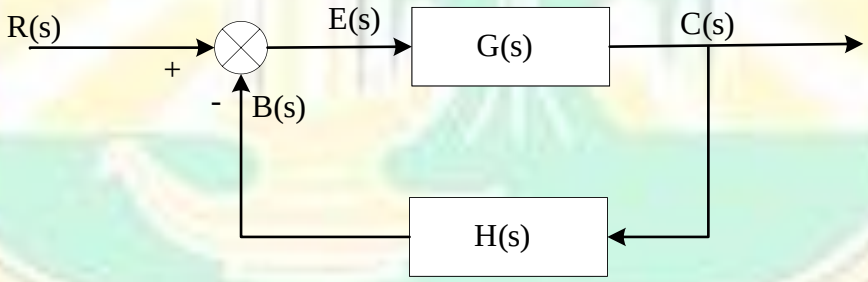
untuk masukan undak satuan, laju dan parabolik satuan. Kesalahan keadaan mantap akan semakin kecil jika konstanta-konstanta tersebut semakin tinggi [22]. Dapat ditinjau dari sistem kendali lingkar terbuka pada persamaan 2.49 berikut:

$$G(s)H(s) = \frac{K(s+z_1)(s+z_2)\cdots(s+z_m)}{s^N(s+p_1)(s+p_2)\cdots(s+p_k)} \quad \textbf{(2.49)}$$

dengan:

- k = n > m
- $z_1, z_2, \ldots, z_m$ yaitu zero pada G(s)H(s)
- $p_1, p_2, \ldots, p_k$ yaitu pole pada G(s)H(s)

Berdasarkan persamaan 2.50 pada bagian pole atau penyebut dari fungsi alih melibatkan bentuk S^N yang mana menyatakan pole rangkap N pada titik asal serta menyatakan tipe sistem. Penyebutan sistem dengan tipe-0, tipe-1, tipe-2 dan seterusnya, apabila nilai N menunjukkan N=0, N=1, N=2, dan seterusnya. Ketika tipe sistem semakin besar maka ketelitian menjadi semakin baik, tetapi hal tersebut menjadikan kestabilan memburuk. Dalam perwujudannya sangat jarang dijumpai sistem dengan tipe ke 3 atau lebih besar, karena kesulitan dalam mendesain sistem yang stabil dengan lebih dari dua integrasi pada lintasan umpan maju. Dilihat dari sistem lingkar tertutup yang ditunjukkan pada Gambar 2.13 dibawah



Gambar 2. 13 Diagram Blok Sistem Lingkar Tertutup

Dari Gambar 2.15 dapat dihasilkan persamaan 2.50 dan persamaan 2.51 yang dituliskan sebagai berikut

$$E(s) = R(s) - H(s)C(s) \quad \textbf{(2.50)}$$

$$C(s) = E(s)G(s) \quad \textbf{(2.51)}$$

Selanjutnya subsitusikan nilai C(s) pada persamaan 2.51 ke persamaan 2.52 yang dituliskan seperti persamaan 2.53, selanjutnya persamaan 2.53 disederhanakan sehingga diperoleh fungsi alih pada persamaan 2.54.

$$E(s) = R(s) - H(s)E(s)G(s) \quad \textbf{(2.52)}$$

$$E(s) + H(s)E(s)G(s) = R(s) \quad \textbf{(2.53)}$$

$$E(s)[1 + G(s)H(s) = R(s) \quad \textbf{(2.54)}$$

Pada fungsi alih diantara sinyal *input* kesalahan [e(t)] dengan sinyal *input* [r(t)] dituliskan pada persamaan 2.55 [11].

$$\frac{E(s)}{R(s)} = \frac{1}{1 + G(s)H(s)} \quad \textbf{(2.55)}$$

Kesalahan keadaan mantap merupakan selisih antara sinyal *input* dengan sinyal *feed-back*. Lalu digunakan teorema nilai akhir sehingga bisa diperoleh performansi keadaan mantap pada sistem stabil, karena nilai E(s) dapat dinyatakan dalam bentuk persamaan 2.56.

$$E(s) = \frac{1}{1 + G(s)H(s)} R(s) \quad \textbf{(2.56)}$$

Jadi, untuk kesalahan keadaan mantap dapat dituliskan pada bentuk persamaan 2.57 berikut

$$e_{ss} = \lim_{t\to\infty} e(t) = \lim_{s\to 0} sE(s) = \lim_{s\to 0} \frac{sR(s)}{1+G(s)H(s)} \quad \textbf{(2.57)}$$

Lalu untuk persamaan 2.58 menyatakan persamaan pada kesalahan keadaan mantap apabila diberi masukan undak

$$e_{ss} = \lim_{s\to 0} \frac{s}{1+G(s)H(s)} \frac{1}{s} = \frac{1}{1+G(0)H(0)} \quad \textbf{(2.58)}$$

Berdasarkan persamaan 2.58 di atas didapat besaran konstanta kesalahan posisi dapat dituliskan pada persamaan 2.59.

$$K_p = \lim_{s\to 0} G(s)H(s) = G(0)H(0) \quad \textbf{(2.59)}$$

Pada sistem tipe-0, nilai konstanta kesalahan posisi dicari menggunakan persamaan 2.60.

$$K_p = \lim_{s\to 0} \frac{K(s+z_1)(s+z_2)\cdots(s+z_m)}{(s+p_1)(s+p_2)\cdots(s+p_k)} = K \quad \textbf{(2.60)}$$

Sistem tipe-1 dan diatasnya, niai konstanta kesalahan dicari menggunakan persamaan 2.61

$$K_p = \lim_{s\to 0} \frac{K(s+z_1)(s+z_2)\cdots(s+z_m)}{s^N(s+p_1)(s+p_2)\cdots(s+p_k)} = \infty \text{ untuk N} \geq 1 \quad \textbf{(2.61)}$$

Berdasarkan persamaan 2.59 sampai persamaan 2.61 diketahui bahwa pada tipe-0, maka nilai konstanta kesalahan posisi nilainya terhingga, tetapi untuk tipe-1 dan dan diatasnya maka nilainya tak terhingga. Maka dari itu, besar kesalahan keadaan mantap yang diberi masukan undak satuan diringkas pada persamaan 2.62 dan persamaan 2.63, dimana nilai kesalahan keadaan mantap akan bernilai 0 untuk masukan undak satuan jika tipe yang digunakan sistem tipe 1 atau diatasnya.

$$e_{ss} = \frac{1}{1+K_p} \quad \textbf{(2.62)}$$

$$e_{ss} = 0 \quad \textbf{(2.63)}$$

Persamaan 2.64 menyatakan persamaan kesalahan keadaan mantap terhadap masukan kecepatan atau laju satuan

$$e_{ss} = \lim_{s\to 0} \frac{s}{1+G(s)H(s)} \frac{1}{s^2} = \lim_{s\to 0} \frac{1}{1+G(s)H(s)} \frac{1}{s} = \lim_{s\to 0} \frac{1}{sG(s)H(s)} \quad \textbf{(2.64)}$$

Berdasarkan persamaan 2.65 di atas, didapat besaran konstanta kesalahan kecepatan yang dinyatakan dalam persamaan 2.66.

$$K_v = \lim_{s\to 0} sG(s)H(s) \quad \textbf{(2.65)}$$

Untuk kesalahan keadaan mantap apabila diberi masukan laju dituliskan dalam persamaan 2.66.

$$e_{ss} = \frac{1}{K_v} \quad \textbf{(2.66)}$$

Kesalahan kecepatan merupakan kesalahan posisi yang timbul dari *input* laju satuan. Pada tipe sistem 0, konstanta kesalahan kecepatan diperoleh menggunakan persamaan 2.67.

$$K_v = \lim_{s\to 0} \frac{sK(s+z_1)(s+z_2)\cdots(s+z_m)}{(s+p_1)(s+p_2)\cdots(s+p_k)} = 0 \quad \textbf{(2.67)}$$

Konstanta kesalahan kecepatan tipe sistem 1 diperoleh menggunakan persamaan 2.68.

$$K_v = \lim_{s\to 0} \frac{sK(s+z_1)(s+z_2)\cdots(s+z_m)}{s(s+p_1)(s+p_2)\cdots(s+p_k)} = K \quad \textbf{(2.68)}$$

Konstanta kesalahan kecepatan tipe sistem 2 atau diatasnya diperoleh menggunakan persamaan 2.69.

$$K_v = \lim_{s\to 0} \frac{sK(s+z_1)(s+z_2)\cdots(s+z_m)}{s^N(s+p_1)(s+p_2)\cdots(s+p_k)} = \infty, N \geq 2 \quad \textbf{(2.69)}$$

Persamaan 2.70 menyatakan persamaan kesalahan keadaan mantap terhadap masukan parabolik satuan

$$e_{ss} = \lim_{s\to 0} \frac{s}{1+G(s)H(s)} \frac{1}{s^3} = \lim_{s\to 0} \frac{1}{1+G(s)H(s)} \frac{1}{s^2} = \lim_{s\to 0} \frac{1}{s^2G(s)H(s)} \quad \textbf{(2.70)}$$

Untuk masukan parabolik dituliskan pada persamaan 2.71 dan persamaan 2.72.

$$r(t) = \frac{t^2}{2}, t \geq 0. \quad \textbf{(2.71)}$$

$$r(t) = 0, t < 0 \quad \textbf{(2.72)}$$

Berdasarkan persamaan 2.70 maka didapat besar konstanta kesalahan apabila diberi masukan parabolik dituliskan pada persamaan 2.73 berikut

$$K_a = \lim_{s\to 0} s^2G(s)H(s) \quad \textbf{(2.73)}$$

Jadi untuk kesalahan keadaan mantap apabila diberi masukan parabolik dituliskan dengan persamaan 2.74

$$e_{ss} = \frac{1}{K_a} \tag{2.74}$$

Kesalahan percepatan merupakan kesalahan posisi yang timbul oleh *input* parabolik satuan. Konstanta kesalahan percepatan untuk tipe sistem 0 diperoleh menggunakan persamaan 2.75.

$$K_a = \lim_{s\to 0} \frac{s^2 K(s+z_1)(s+z_2)\cdots(s+z_m)}{(s+p_1)(s+p_2)\cdots(s+p_k)} = 0 \tag{2.75}$$

Konstanta kesalahan percepatan untuk tipe sistem 1 diperoleh menggunakan persamaan 2.76.

$$K_a = \lim_{s\to 0} \frac{s^2 K(s+z_1)(s+z_2)\cdots(s+z_m)}{s(s+p_1)(s+p_2)\cdots(s+p_k)} = 0 \tag{2.76}$$

Konstanta kesalahan percepatan untuk tipe sistem 2 diperoleh menggunakan persamaan 2.77.

$$K_a = \lim_{s\to 0} \frac{s^2 K(s+z_1)(s+z_2)\cdots(s+z_m)}{s^2(s+p_1)(s+p_2)\cdots(s+p_k)} = K \tag{2.77}$$

Konstanta kesalahan percepatan untuk tipe sistem 3 atau diatasnya diperoleh menggunakan persamaan 2.78.

$$K_a = \lim_{s\to 0} \frac{s^2 K(s+z_1)(s+z_2)\cdots(z_m)}{s^N(s+p_1)(s+p_2)\cdots(s+p_k)} = \infty, N \geq 3 \tag{2.78}$$

Tabel 2.4 yang menyajikan secara ringkas kesalahan keadaan mantap untuk tipe sistem 0, tipe sistem 1 dan tipe sistem 2 dengan masukan undak, laju dan percepatan.

Tabel 2.4 Kesalahan Keadaan Mantap dalam Bentuk Penguatan K [23]

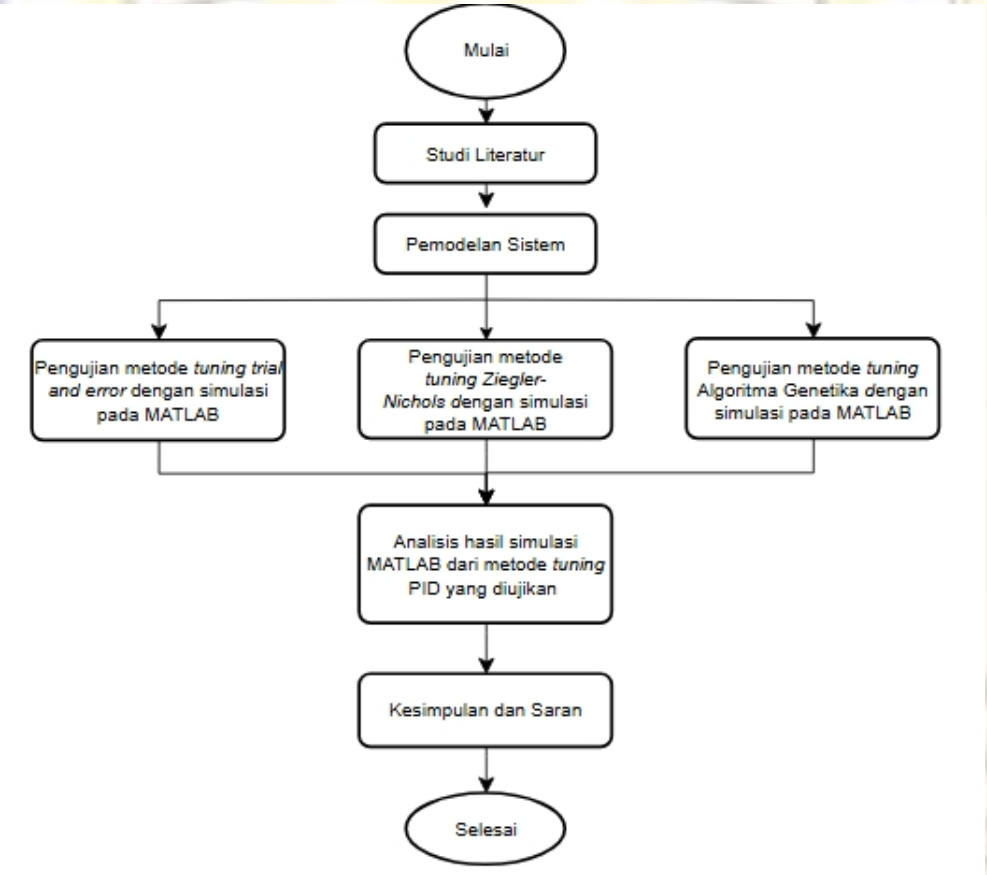
Tipe Sistem	**Masukan**		
	Undak Satuan	**Laju Satuan**	**Percepatan**
0	$\frac{1}{1+K_p}$	∞	∞
1	0	$\frac{1}{K_v}$	∞
2	0 .	0	$\frac{1}{K_a}$

BAB III METODOLOGI PENELITIAN

Bab ini membahas tentang diagram alir penelitian dan tahapan penelitian yang akan dilakukan.

3.1 Diagram Alir Penelitian

Diagram alir dari penelitian ini apa dilihat pada Gambar 3.1.



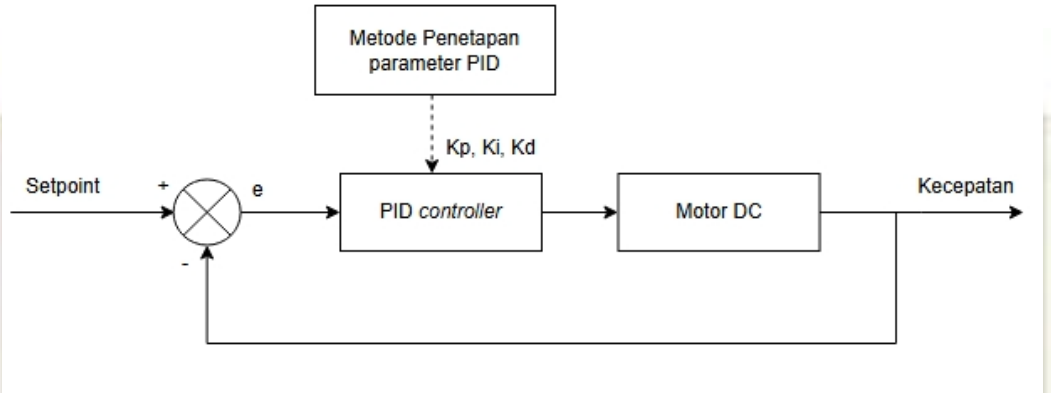
Gambar 3. 1 Diagram Alir Penelitian

3.1.1 Studi Literatur

Pada penelitian ini, studi literatur dilakukan dengan pengumpulan materi tentang pengendalian motor DC mengggunakan pengendali PID dan menentukan topik pembahasan. Pengumpulan materi meliputi pengendalian motor DC menggunakan PID, metode *tuning* PID, dan pengaruh metode *tuning* terhadap kinerja pengendalian kecepatan motor DC.

3.1.2 Pemodelan Sistem

Untuk memudahkan pembuatan pengendalian motor DC menggunakan PID dengan beberapa metode *tuning* PID dibutuhkan pemodelan sistem. Diagram blok pengendali PID ditunjukkan oleh Gambar 3.2.



Gambar 3. 2 Diagram blok *tuning* pengendali PID

Pada diagram blok tersebut dapat dilihat bahwa *input* sistem berupa *set point* dengan *tuning* parameter PID menggunakan *trial and error*, Ziegler-Nichols, dan GA. *Plant* yang digunakan adalah sistem umpan balik yang keluarannya berupa kecepatan motor DC [24].

Keluaran dari sistem ini berupa kecepatan motor DC sehingga diperlukan fungsi alih dari motor DC untuk memberikan gambaran jelas pada pengendaliannya. Fungsi alih motor DC yang digunakan ditunjukkan pada persamaan 3.1 berikut.

$$\frac{\omega(s)}{V_a(s)} = \frac{K_t}{J_m L_a s^2 + (R_a J_m + B_m L_a)s + (K_t K_b + R_a B_m)} \quad \textbf{(3.1)}$$

Nilai-nilai di persamaan tersebut didapatkan melalui parameter motor DC dan parameter motor DC yang ada pada Tabel 3.1 berikut ini [7].

Tabel 3. 1 Parameter Motor DC

Parameter	Nilai
R_a	0,45 ohm
La	0,1 H
K_b	0,067 Vs/rad
K_t	0,067 Nm/A
J_m	0,0113 Kgm^2
B_m	0,028 Nms/rad

Berdasarkan Tabel 3.1 di atas, maka fungsi alih motor DC dapat dihitung menggunakan persamaan 3.1 dan menghasilkan fungsi alih seperti pada persamaan 3.2 berikut

$$\frac{\omega(s)}{V_a(s)} = \frac{0{,}067}{0{,}00113s^2 + 0{,}0078854s + 0{,}0171} \quad \textbf{(3.2)}$$

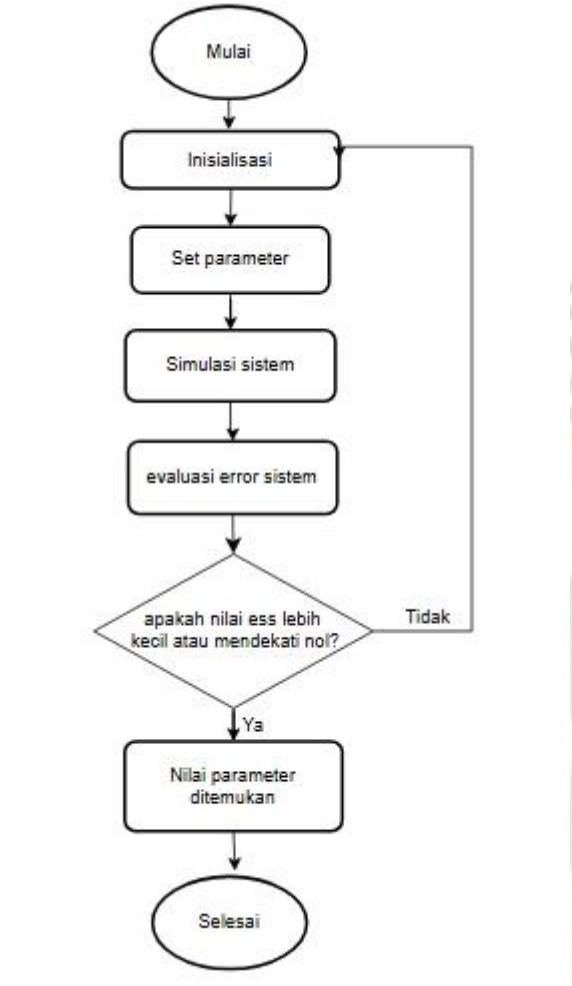
3.1.3 Pengujian Metode *Tuning*

3.1.3.1 *Tuning Trial and Error*

Diagram alir untuk proses *tuning trial & error* dapat dilihat pada Gambar 3.3 dimana proses dimulai saat melakukan inisialisasi dengan memasukkan nilai secara acak. *Tuning trial & error* akan berhenti saat kriteria peghentian terpenuhi, yaitu jika parameter Kp, Ki, dan Kd nilai *error*-nya lebih kecil atau mendekati nol [25].

Tuning trial & error akan dibatasi sebanyak lima kali dengan nilai parameter pengendali PID yang akan diujikan pada penelitian ini adalah sebagai berikut:

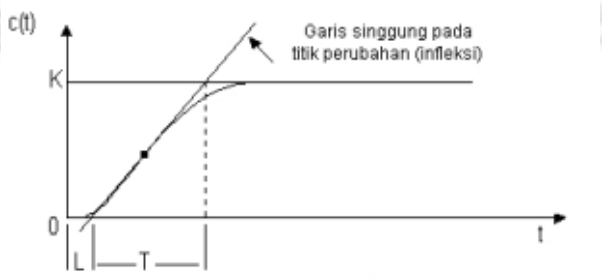
- nilai P = nilai I = nilai D
- nilai P > nilai I > nilai D
- nilai I > nilai P > nilai D
- nilai D > nilai P > nilai I
- nilai P > nilai D > nilai I



Gambar 3. 3 Diagram Alir *Tuning Ttrial & Error*

3.1.3.2 *Tuning* Ziegler-Nichols

Tuning PID Ziegler-Nichols yang digunakan pada penelitian ini merupakan metode kurva reaksi yang digunakan pada kondisi *open loop*. Metode ini menggunakan persamaan pada Tabel 3.2 untuk menghitung Kp, Ti, dan Td dimana L adalah waktu tunda dan T adalah waktu konstan. Nilai L dan T dapat dihitung dengan menambahkan garis tangen pada kurva seperti yang terlihat pada Gambar 3. 4 berikut



Gambar 3. 4 Kurva Reaksi S

Tabel 3.2 Nilai Kp, Ti, dan Td berdasarkan parameter L dan T

Pengendali	Kp	Ti	Td
P	T/L	∞	0
PI	0,9T/L	L/0,3	0
PID	1,2T/L	2L	0,5L

Nilai Ki dan Kd akan didapatkan dengan menggunakan perhitungan pada persamaan 3.3 dan persamaan 3.4 berikut

$$K_i = \frac{K_p}{T_i} \quad \textbf{(3.3)}$$

$$K_d = K_p \times T_d \quad \textbf{(3.4)}$$

3.1.3.3 *Tuning* Algoritma Genetika (GA)

Algoritma genetika merupakan teknik optimasi yang terinspirasi dari teori evolusi Darwin yang menyatakan bahwa kelangsungan hidup suatu organisme ditentukan oleh aturan "yang terkuatlah yang bertahan". Algoritma genetika pada pengendali PID digunakan untuk mencari parameter PID yang optimal. Pada proses pencarian solusi optimal tersebut digunakan proses reproduksi, *crossover,* dan mutasi [26].

Nilai parameter PID direpresentasikan sebagai kromosom yang membentuk satu individu. Proses reproduksi dilakukan untuk memilih individu dari populasi saat ini tanpa perubahan untuk menjadi bagian dari populasi baru. Hal ini bertujuan untuk memberikan kesempatan bagi solusi yang sudah baik untuk tetap bertahan. Proses *crossover* menciptakan individu baru (*offspring*) dari individu saat ini (*parents*) sesuai dengan probabilitas *crossover.* Proses ini dilakukakn dengan memilih satu atau lebih titik *crossover* di dalam kromosom setiap *parent* pada posisi yang sama kemudian dipertukarkan antar *parents.* Setelah itu, proses mutasi membuat perubahan pada setiap individu yang dipilih berdasarkan probabilitas mutasi dengan memodifikasi satu atau lebih nilai dalam kromosom. Kemudian, populasi akan di evaluasi agar mendapatkan solusi yang optimal dengan menghitung nilai *fitness* dari setiap individu melalui proses meminimalkan *error* menggunakan fungsi *fitness* pada persamaan 3.5 berikut [27]

$$Fitness = \sum_{i=1}^{n} (desire\ output_i - actual\ output_i)^2 \quad \textbf{(3.5)}$$

Dimana:

Desire output = keluaran yang diinginkan
Actual output = keluaran aktual
n = individu

Satu generasi pada algoritma genetika merupakan satu siklus lengkap yang meliputi pembangkitan populasi, penerapan operator genetik (reproduksi,

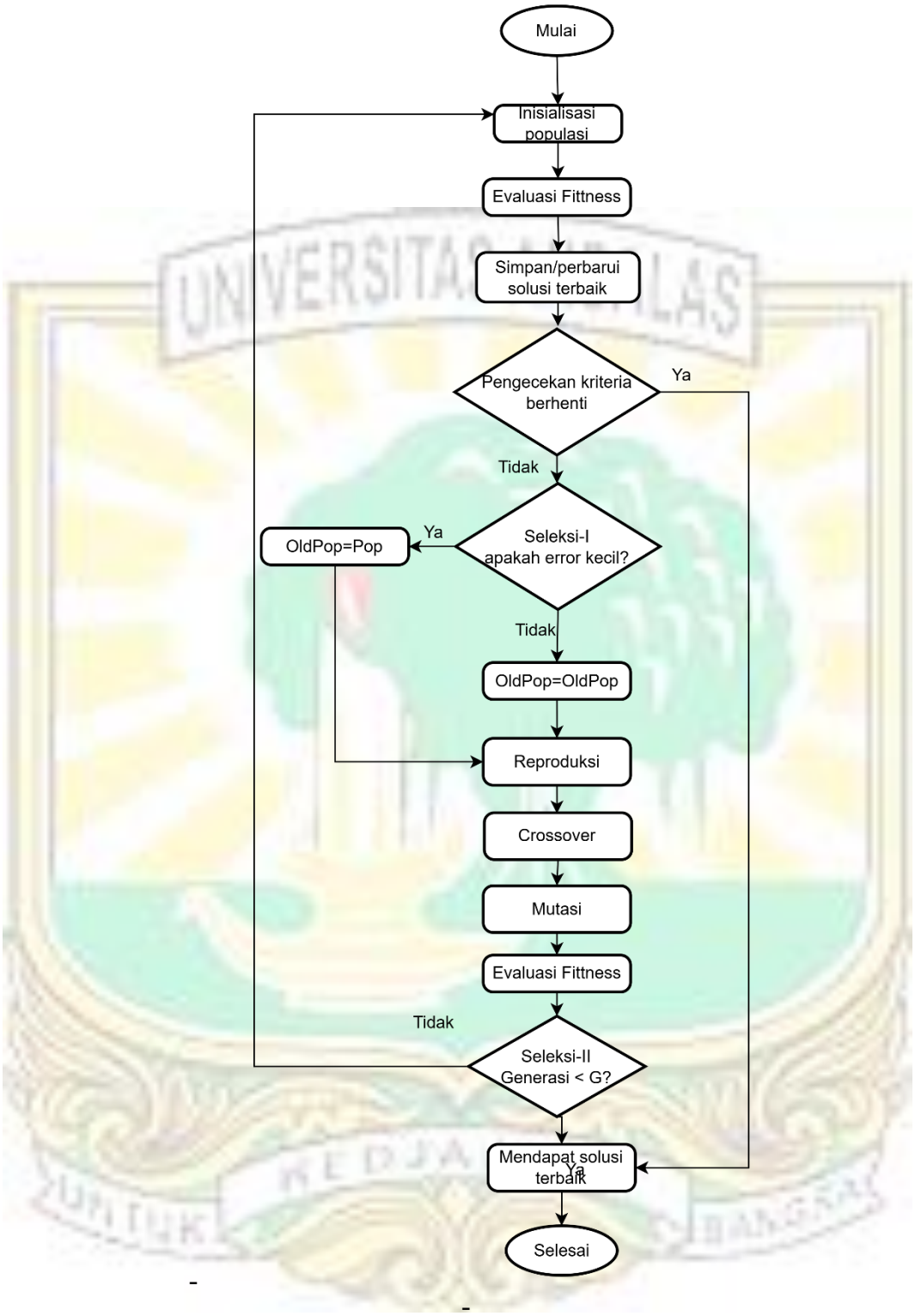
crossover, dan mutasi), dan evaluasi anggota populasi. Siklus ini akan terus berulang hingga kriteria penghentian terpenuhi.

Parameter algoritma genetika yang digunakan pada peneltitian ini untuk mendapatkan parameter PID yang optimal dapat dilihat pada Tabel 3.3 berikut

Tabel 3. 3 Parameter untuk *Tuning* PID menggunakan Algoritma Genetika

Parameter GA	Nilai
Populasi	50
Generasi	100
Crossover	0,8
Mutasi	0,06

Secara keseluruhan, proses algoritma genetika digambarkan oleh diagram alir pada Gambar 3.5 berikut



Gambar 3. 5 Diagram Alir *Tuning* Parameter PID menggunakan Algoritma Genetika

3.1.4 Metode Analisis Hasil *Tuning* PID

Analisis sistem kendali dilakukan untuk menganalisis sistem yang sesuai dengan kriteria perancangan. Kriteria perancangan digunakan sebagai acuan untuk mendapatkan hasil yang sesuai dengan kebutuhan pengguna. Analisis yang digunakan pada penelitian ini meliputi analisis peralihan dan analisis kesalahan. Hasil analisis sistem pengendali PID dengan metode *tuning trial & error,* Ziegler-Nichols, dan algoritma genetika akan dibandingkan dengan kriteria perancangan untuk melihat metode *tuning* mana yang mempunyai kecepatan respon sistem yang lebih baik. Respon sistem yang dijadikan pembanding ialah kesalahan keadaan tunak (ess), waktu naik (tr), waktu tunak (ts) dan lewatan maksimum (Mp).

3.1.5 Penyusunan Laporan

Penyusunan laporan dilakukan setelah memperoleh hasil pada tahapan sebelumnya.

3.2 Kriteria Perancangan

Pada penelitian ini dilakukan analisis domain waktu, yaitu analisis peralihan dan analisis kesalahan dengan masukan undak satuan untuk melihat performansi kecepatan motor DC. Analisis peralihan dilakukan dengan memerhatikan nilai dari parameter-parameter peralihan, yaitu waktu naik (t_p), waktu keadaan mantap (t_s), dan lewatan maksimum (M_P). Analisis kesalahan yang diharapkan dari sistem yang dirancang adalah nilai kesalahan keadaan mantap (ess) kurang dari 0,5. Analisis peralihan dan analisis kesahalan dari sistem dapat dilihat pada Tabel 3.4 berikut

Tabel 3. 4 Kriteria Perancangan Sistem Pengendalian Kecepatan Motor DC

Parameter	Nilai Perancangan
Waktu Naik (t_r)	< 0,15 s
Waktu Keadaan Mantap (t_s)	< 0,6 s
Lewatan Maksimum (M_P)	< 20%
Kesalahan Keadaan Mantap (ess)	< 0,1

BAB IV HASIL DAN ANALISIS

Analisis peralihan dan analisis kesalahan digunakan untuk menjelaskan data hasil percobaan pengendalian kecepatan motor DC. Analisis kinerja kecepatan motor DC dilakukan dengan pengendali PID yang menggunakan metode *tuning trial and error*, *tuning* Ziegler-Nichols, dan *tuning* Algoritma Genetika (GA). Hasil data akan dianalisis dengan membandingkan ketiga metode *tuning* berdasarkan krteria perancangan

4.1 Analisis Sistem Pengendalian Kecepatan Motor DC Menggunakan Pengendali PID dengan Metode *Tuning Trial & Error*

Pada penelitian analisis kinerja pengendalian kecepatan motor DC menggunakan pengendali PID ini, *tuning* metode *trial & error* dilakukan sebanyak lima kali dengan nilai-nilai parameter PID yang berbeda.

4.1.1 Analisis Peralihan

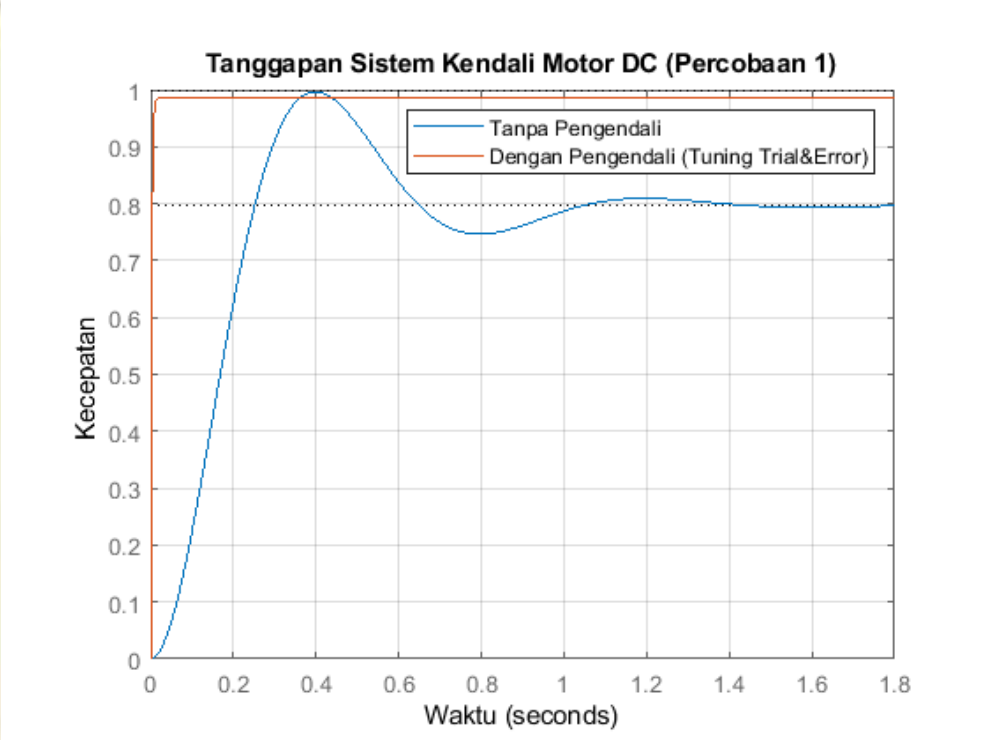
Hasil dan analisis peralihan sistem pengendalian kecepatan motor DC menggunakan pengendali PID dengan *tuning trial & error* dapat dilihat pada Tabel 4.1 berikut. Percobaan 1 diatur dengan nilai Kp = 7, Ki = 7, Kd = 7, percobaan 2 diatur dengan nilai Kp = 25, Ki = 14, Kd = 1, percobaan 3 diatur dengan nilai Kp = 8, Ki = 30, Kd = 1, percobaan 4 diatur dengan nilai Kp = 2, Ki = 1, Kd = 6, serta percobaan 5 diatur dengan nilai Kp = 10, Ki = 2, Kd = 5.

Tabel 4. 1 Informasi Analisis Peralihan Sistem Pengendalian Kecepatan Motor DC menggunakan Pengendali PID dengan *Tuning Trial & Error*

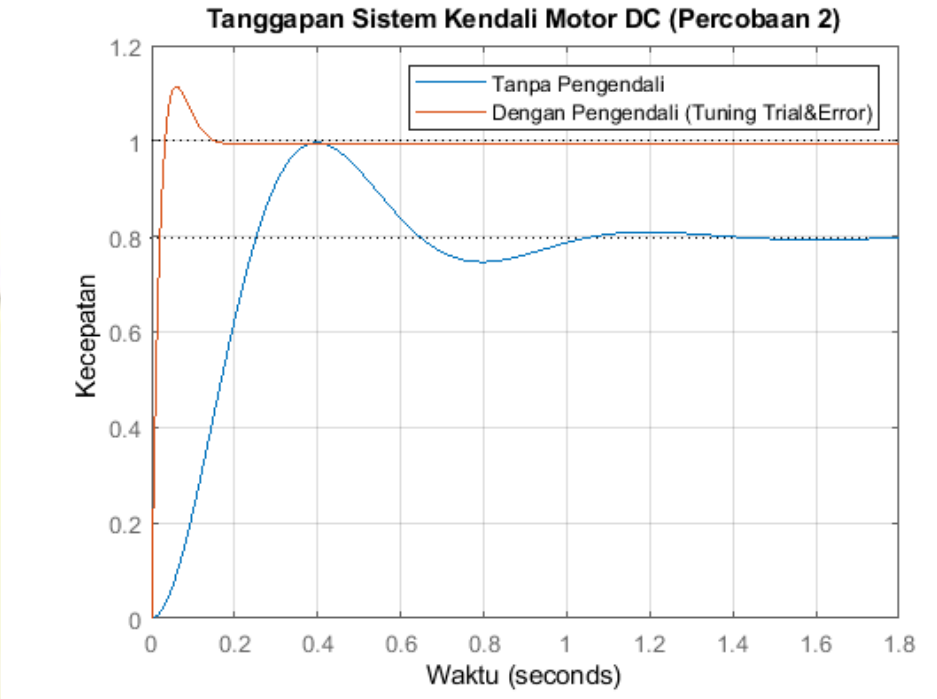
Para meter	Nilai					
	Tanpa Pengendali	P 1	P 2	P 3	P 4	P 5
t_r (s)	0,17032	0,00554	0,02388	0,03506	0,00654	0,00782
t_s (s)	0,97482	0,01222	0,12453	0,25344	6,4269	0,01886
M_P (%)	25,109	0	11,424	2,3008	1,1864	0

Berdasarkan Tabel 4.1, nilai parameter analisis peralihan yang dilakukan dengan *tuning trial & error* sebanyak lima percobaan menghasilkan empat percobaan yang memenuhi kriteria perancangan, yaitu Percobaan 1, Percobaan 2, Percobaan 3, dan Percobaan 5. Berdasarkan data tersebut, setelah sistem diberi pengendali PID, didapatkan bahwa waktu naik di setiap percobaan memenuhi kriteria perancangan yang telah ditetapkan karena adanya pengendali P yang bisa

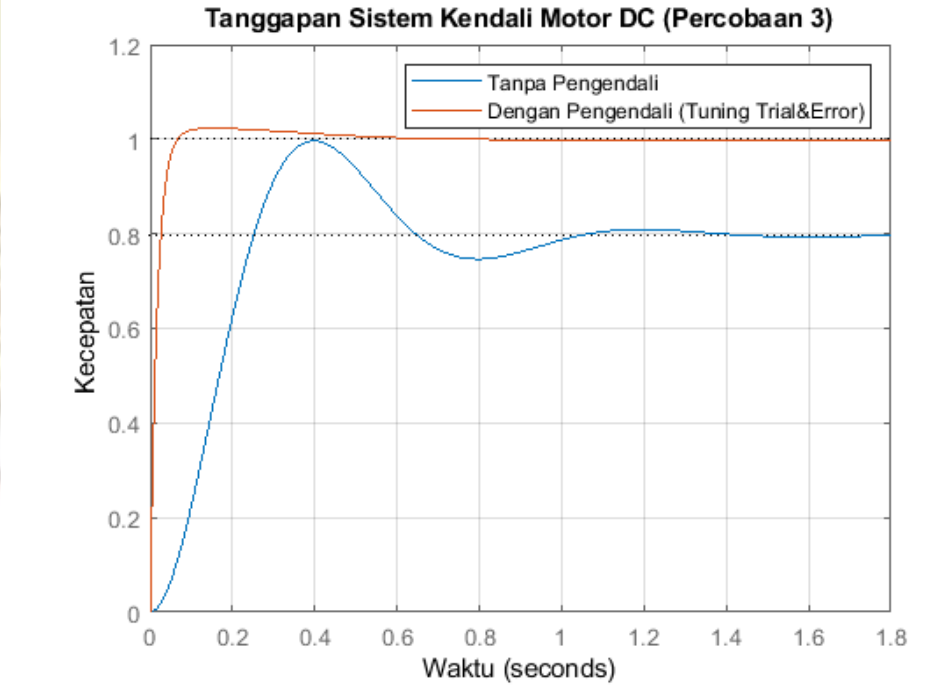
menurunkan waktu naik sistem. Waktu keadaan mantap yang dihasilkan memenuhi kriteria perancangan kecuali percobaan 4. Lewatan maksimum yang diperoleh setelah sistem diberi pengendali PID pada percobaan 1 sampai percobaan 5 semuanya memenuhi kriteria perancangan. Hal ini disebabkan pada percobaan diberikan pengendali D yang mambuat lewatan maksimum menurun sesuai dengan Tabel 2.1 [12] [13]. Grafik analisis peralihan sistem kendali kecepatan motor DC untuk *tuning trial & error* percobaan 1 sampai 5 dapat dilihat pada Gambar 4.1 sampai Gambar 4.5



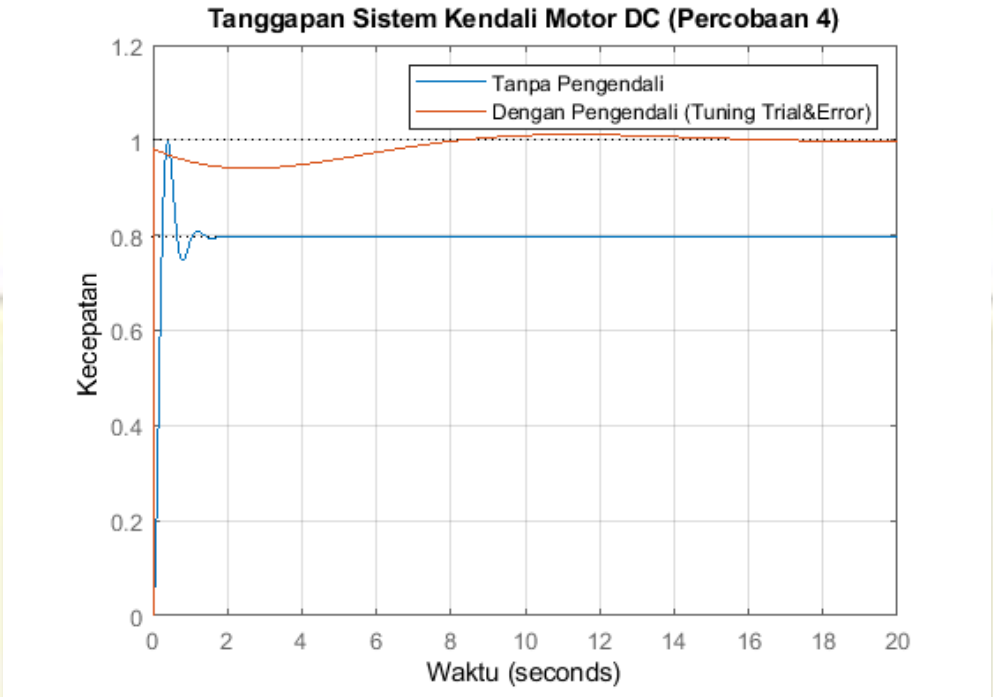
Gambar 4. 1 Grafik Analisis Peralihan Sistem Pengendalian Kecepatan Motor DC menggunakan Pengendali PID dengan *Tuning Trial & Error* Percobaan 1



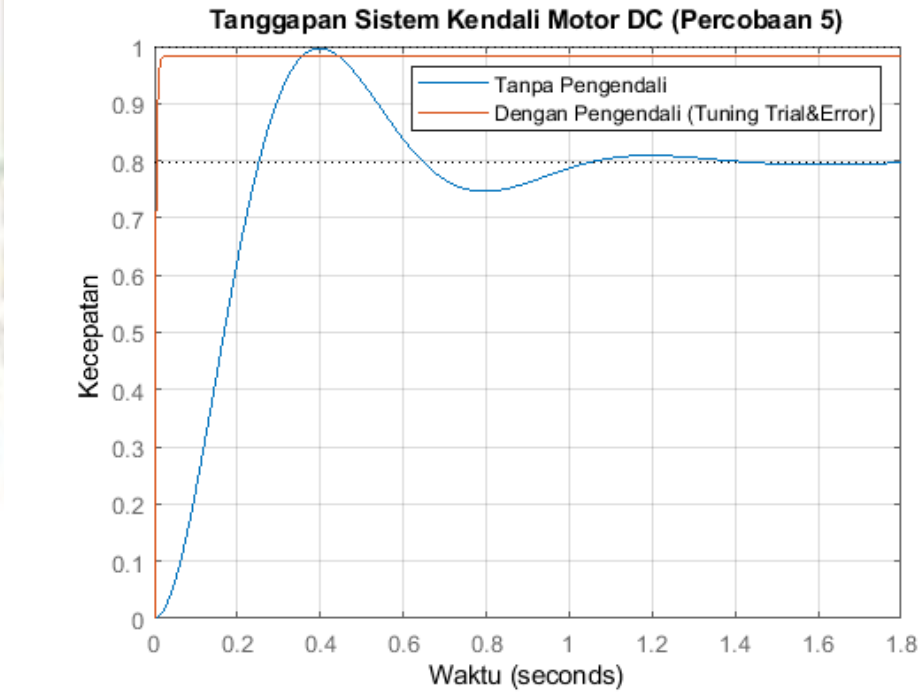
Gambar 4. 2 Grafik Analisis Peralihan Sistem Pengendalian Kecepatan Motor DC menggunakan Pengendali PID dengan *Tuning Trial & Error* Percobaan 2



Gambar 4. 3 Grafik Analisis Peralihan Sistem Pengendalian Kecepatan Motor DC menggunakan Pengendali PID dengan *Tuning Trial & Error* Percobaan 3



Gambar 4. 4 Grafik Analisis Peralihan Sistem Pengendalian Kecepatan Motor DC menggunakan Pengendali PID dengan *Tuning Trial & Error* Percobaan 4



Gambar 4. 5 Grafik Analisis Peralihan Sistem Pengendalian Kecepatan Motor DC menggunakan Pengendali PID dengan *Tuning Trial & Error* Percobaan 5

4.1.2 Analisis Kesalahan

Hasil dan analisis kesalahan sistem pengendalian kecepatan motor DC menggunakan pengendali PID dengan *tuning trial & error* dapat dilihat pada Tabel 4.2 berikut

Tabel 4. 2 Informasi Analisis Kesalahan Sistem Pengendalian Kecepatan Motor DC menggunakan Pengendali PID dengan *Tuning Trial & Error*

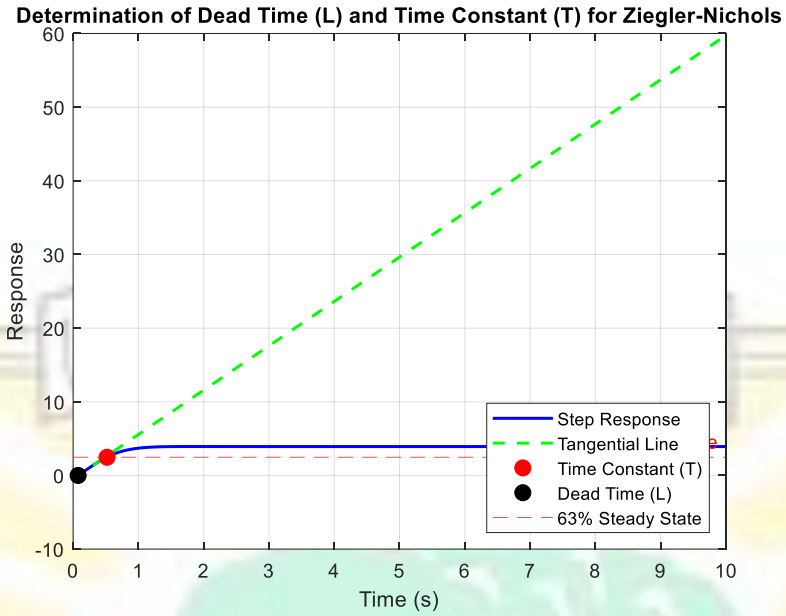
Parameter	**Nilai**				
	Percobaan 1 (Kp = 7, Ki = 7, Kd = 7)	**Percobaan 2** (Kp = 25, Ki = 14, Kd = 1)	**Percobaan 3** (Kp = 8, Ki = 30, Kd = 1)	**Percobaan 4** (Kp = 2, Ki = 1, Kd = 6)	**Percobaan 5** (Kp = 10, Ki = 2, Kd = 5)
ess	0,0364	0,0182	0,0085	0,2551	0,1275

Berdasarkan Tabel 4.2 di atas nilai kesalahan keadaan mantap yang dihasilkan dari lima kali percobaan *tuning trial & error* tiga diantara lima percobaan memenuhi kriteria perancangan, yaitu percobaan 1, percobaan 2, dan percoobaan 3. Hal ini disebabkan oleh besarnya nilai pengendali I yang diberikan pada percobaan 1, percobaan 2, dan percobaan 3.

4.2 Analisis Sistem Pengendalian Kecepatan Motor DC Menggunakan Pengendali P, PI, PD, dan PID dengan Metode *Tuning* Ziegler-Nichols

Pada penelitian ini, metode *tuning* Ziegler-Nichols yang digunakan adalah metode kurva reaksi. Metode ini dilakukan dengan menentukan waktu tunda (L) dan waktu konstan (T) pada kurva S menggunakan bantuan garis tangen.

Pada penelitian ini, fungsi alih menghasilkan kurva S dengan nilai L = 0,0765 dan T = 0,4435 seperti yang terlihat pada Gambar 4.6.



Gambar 4. 6 Kurva Reaksi Ziegler-Nichols

Nilai parameter pengendali P, PI, PD, dan PID yang didapatkan berdasarkan nilai L dan T kemudian dihitung menggunakan Tabel 3.2 yang kemudian menghasilkan nilai seperti pada Tabel 4.3 sampai Tabel 4.6 berikut.

Tabel 4. 3 Hasil Nilai Parameter Pengendali P menggunakan *Tuning* Ziegler-Nichols

Parameter Pengendali	Nilai
Kp	5,797

Tabel 4. 4 Hasil Nilai Parameter Pengendali PI menggunakan *Tuning* Ziegler-Nichols

Parameter Pengendali	Nilai
Kp	5,218
Ki	20,461

Tabel 4. 5 Hasil Nilai Parameter Pengendali PD menggunakan *Tuning* Ziegler-Nichols

Parameter Pengendali	Nilai
Kp	6,957
Kd	0,266

Tabel 4. 6 Hasil Nilai Parameter Pengendali PID menggunakan *Tuning* Ziegler-Nichols

Parameter Pengendali	Nilai
Kp	6,957
Ki	45,47
Kd	0,266

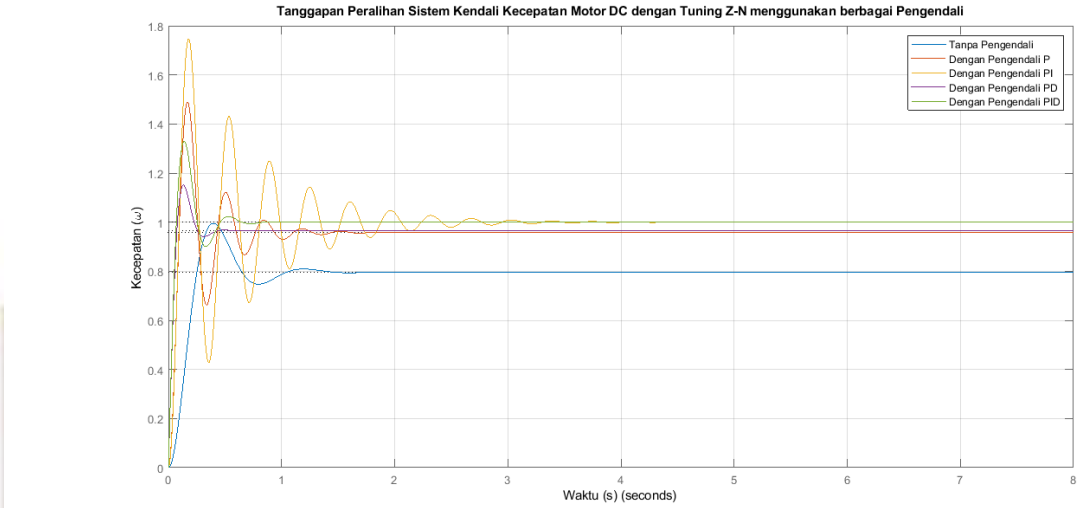
4.2.1 Analisis Peralihan

Hasil dan analisis peralihan sistem pengendalian kecepatan motor DC menggunakan pengendali PID dengan *tuning* Ziegler-Nichols ditunjukkan pada Tabel 4.7 berikut

Tabel 4. 7 Informasi Analisis Peralihan Sistem Pengendalian Kecepatan Motor DC Menggunakan pengendali dengan *Tuning* Ziegler-Nichols

Parameter	Nilai				
	Tanpa Pengendali	Pengendali P	Pengendali PI	Pengendali PD	Pengendali PID
t_r (s)	0,17023	0,063	0,064	0,057	0,055
t_s (s)	0,97482	1,058	2,511	0,347	0,56
M_P (%)	25,109	55,425	74,625	19,357	32,973

Berdasarkan Tabel 4.7 dapat dilihat bahwa waktu naik untuk semua pengendali memenuhi kriteria perancangan karena pada setiap pengendali diberi pengendali P yang berguna untuk menurunkan waktu naik pada sistem kendali berdasarkan Tabel 2.1 [12] [13]. Waktu keadaan mantap yang dihasilkan menggunakan metode *tuning* Ziegler-Nichols yang memenuhi kriteria perancangan adalah pengendali PD dan PID. Lewatan maksimum yang didapatkan dengan berbagai konfigurasi pengendali memiliki nilai yang cukup tinggi dari nilai kriteria perancangan yang ditetapkan, dimana lewatan maksimum yang memenuhi kriteria perancangan hanya pada pengendali PD, yaitu 19,357%. Namun, nilai tersebut hampir mendekati batas kriteria perancangan untuk lewatan maksimum, yaitu <20%. Hal ini disebabkan karena nilai pengendali D pada *tuning* Ziegler-Nichols yang kecil. Grafik analisis peralihan untuk sistem pengendalian kecepatan motor DC dengan *tuning* Ziegler-Nichols dapat dilihat pada Gambar 4.7 berikut.



Gambar 4. 7 Grafik Analisis Peralihan Sistem Pengendalian Kecepatan Motor DC menggunakan Pengendali P, PI, PD, dan PID dengan *Tuning* Ziegler-Nichols

4.2.2 Analisis Kesalahan

Hasil dan analisis kesalahan sistem pengendalian kecepatan motor DC menggunakan pengendali PID dengan *tuning* Ziegler-Nichols dapat dilihat pada Tabel 4.8 berikut

Tabel 4. 8 Informasi Analisis Kesalahan Sistem Pengendalian Kecepatan Motor DC Menggunakan pengendali PID dengan *Tuning* Ziegler-Nichols

Parameter	Nilai				
	Tanpa Pengendali	Pengendali P	Pengendali PI	Pengendali PD	Pengendali PID
ess	0,2032	0,0421	0,0125	0,0354	0,0056

Berdasarkan Tabel 4.8 nilai kesalahan keadaan mantap untuk sistem yang menggunakan pengendali P, PI, PD, dan PID dengan *tuning* Ziegler-Nichols memenuhi kriteria perancangan yang ditetapkan. Hal ini disebabkan karena semua pengendali mampu menurunkan kesalahan keadaan mantap walaupun tidak sebesar pengendali I. Bedasarkan tabel tersebut, dapat dilihat juga bahwa kesalahan keadaan mantap untuk pengendali PI dan PID yang kecil.

4.3 Analisis Sistem Pengendalian Kecepatan Motor DC Menggunakan Pengendali P, PI, PD, dan PID dengan Metode *Tuning* Algoritma Genetika (GA)

Pada penelitian ini, nilai parameter PID yang didapatkan oleh *tuning* algoritma genetika didapatkan menggunakan fungsi *fitness* yang telah ditetapkan

pada persamaan 3.5, dimana fungsi *fitness* didefenisikan untuk meminimalkan kesalahan antara keluaran yang diiinginkan dan keluaran aktual. Parameter untuk *tuning* PID algoritma genetika yang digunakan berdasarkan pada Tabel 3.3.

Berdasarkan hasil *tuning* menggunakan algoritma genetika, nilai parameter pengendali P, PI, PD, dan PID yang didapatkan dapat dilihat pada Tabel 4.9 sampai Tabel 4. 12 berikut.

Tabel 4. 9 Hasil Nilai Parameter Pengendali P menggunakan *Tuning* GA

Parameter Pengendali	Nilai
Kp	15.0065

Tabel 4. 10 Hasil Nilai Parameter Pengendali PI menggunakan *Tuning* GA

Parameter Pengendali	Nilai
Kp	9.4871
Ki	4.0454e-05

Tabel 4. 11 Hasil Nilai Parameter Pengendali PD menggunakan *Tuning* GA

Parameter Pengendali	Nilai
Kp	20
Kd	30

Tabel 4. 12 Hasil Nilai Parameter Pengendali PID menggunakan *Tuning* GA

Parameter Pengendali	Nilai
Kp	11.3678
Ki	24.7629
Kd	1.6365

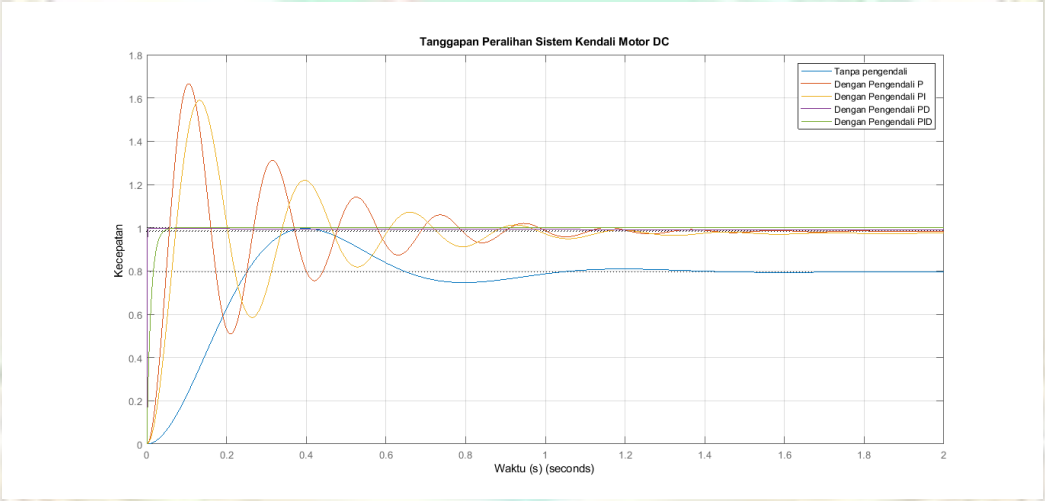
4.3.1 Analisis Peralihan

Hasil dan analisis peralihan sistem pengendalian kecepatan motor DC menggunakan pengendali PID dengan *tuning* GA ditunjukkan oleh Tabel 4.13 berikut

Tabel 4. 13 Informasi Analisis Peralihan Sistem Pengendalian Kecepatan Motor DC Menggunakan pengendali PID dengan *Tuning* GA

Parameter	Nilai				
	Tanpa Pengendali	Pengendali P	Pengendali PI	Pengendali PD	Pengendali PID
t_r (s)	0,17023	0,038	0,050	0,001	0,023
t_s (s)	0,97482	1,075	NaN	0.0	0,040
M_P (%)	25,109	69,39	58,89	0,81	0
ess	0,2032	0.0167	6304.9316	0.0126	0,0103

Berdasarkan Tabel 4.13 dapat dilihat bahwa waktu naik untuk pemua konfigurasi pengendali memenuhi kriteria perancangan karena terdapat pengendali P yang bisa menurunkan waktu naik sesuai dengan Tabel 2.1 [12] [13]. Waktu keadaan mantap yang memenuhi kriteria perancangan adalah pada pengendali PI, PD, dan PID. Lewatan maksimum yang memenuhi kriteria perancangan adalah pada konfigurasi pengendali PD dan PID karena terdapat pengendali D yang dapat menurunkan lewatan maksimum. Grafik analisis peralihan untuk sistem pengendalian kecepatan motor DC dengan *tuning* GA dapat dilihat pada Gambar 4.8 berikut.



Gambar 4. 8 Grafik Analisis Peralihan Sistem Pengendalian Kecepatan Motor DC menggunakan Pengendali P, PI, PD, dan PID dengan *Tuning* GA

4.3.2 Analisis Kesalahan

Hasil dan analisis kesalahan sistem pengendalian kecepatan motor DC menggunakan pengendali PID dengan *tuning* GA ditunjukkan oleh Tabel 4.14 berikut

Tabel 4. 14 Informasi Analisis Kesalahan Sistem Pengendalian Kecepatan Motor DC menggunakan Pengendali PID dengan *Tuning* GA

Parameter	Nilai				
	Tanpa	Pengendali	Pengendali	Pengendali	Pengendali

	Pengendali	P	PI	PD	PID
ess	0,2032	0.0167	6304.9316	0.0126	0,0103

Berdasarkan Tabel 4.14 nilai kesalahan keadaan mantap yang dihasilkan dari hasil *tuning* parameter pengendali menggunakan GA yang memenuhi kriteria perancangan yang sudah diteteapkan adalah pada pengendali P, PD, PID. Pengendali PI tidak memenuhi kriteria perancangan disebebkan oleh nilai pengendali I yang sangat kecil sehingga tidak mampu menghilangkan kesalahan keadaan mantap.

4.4 Perbandingan Hasil Metode *Tuning*

4.4.1 Perbandingan Hasil Analisis Peralihan Sistem Pengendalian Kecepatan Motor DC

Hasil analisis peralihan untuk sistem pengendalian kecepatan motor DC menggunakan metode *tuning* PID *trial & error*, Ziegler-Nichols, dan Algoritma Genetika (GA) ditunjukkan pada Tabel 4.15 berikut ini.

Tabel 4. 15 Perbandingan Hasil Analisis Peralihan Sistem Pengendalian Kecepatan Motor DC Menggunakan Beberapa Metode *Tuning* PID

Metode *Tuning* Pengendali PID		**Parameter Analisis Peralihan**		
		Waktu Naik	**Waktu Keadaan Mantap**	**Lewatan Maksimum**
Tuning Trial & Error	Percobaan 1	✓	✓	✓
	Percobaan 2	✓	✓	✓
	Percobaan 3	✓	✓	✓
	Percobaan 4	✓	X	✓
	Percobaan 5	✓	✓	✓
Tuning Ziegler-Nichols	Pengendali P	✓	X	X
	Pengendali PI	✓	X	X
	Pengendali PD	✓	✓	✓
	Pengendali PID	✓	✓	X
Tuning GA	Pengendali P	✓	X	X
	Pengendali PI	✓	✓	X
	Pengendali PD	✓	✓	✓
	Pengendali PID	✓	✓	✓

Berdasarkan data pada Tabel 4.15, dapat dilihat bahwa parameter PID yang memenuhi kriteria perancangan untuk metode *tuning trial & error* ialah percobaan 1, percobaan 2, percobaan 3, dan percobaan 5. Metode *tuning* Ziegler-Nichols dengan konfigurasi pengendali yang memenuhi kriteria perancangan ialah pengendali PD dan untuk metode *tuning* Algoritma Genetika yang memenuhi kriteria perancangan adalah konfigurasi pengendali PD dan PID.

4.4.2 Perbandingan Hasil Analisis Kesalahan Sistem Pengendalian Kecepatan Motor DC

Hasil analisis kesalahan untuk sistem pengendalian kecepatan motor DC menggunakan metode *tuning* PID *trial & error*, Ziegler-Nichols, dan algoritma genetika ditunjukkan pada Tabel 4.16 berikut ini.

Tabel 4. 16 Perbandingan Hasil Analisis Kesalahan Sistem Pengendalian Kecepatan Motor DC Menggunakan Beberapa Metode *Tuning* PID

Metode *Tuning* Pengendali PID		**Kesalahan Keadaan Mantap**
Tuning Trial & Error	Percobaan 1	✓
	Percobaan 2	✓
	Percobaan 3	✓
	Percobaan 4	X
	Percobaan 5	X
Tuning Ziegler-Nichols	Pengendali P	✓
	Pengendali PI	✓
	Pengendali PD	✓
	Pengendali PID	✓
Tuning GA	Pengendali P	✓
	Pengendali PI	X
	Pengendali PD	✓
	Pengendali PID	✓

Berdasarkan data pada Tabel 4.16 di atas dapat dilihat bahwa parameter PID yang memenuhi kriteria perancangan untuk metode *tuning trial & error* ialah percobaan 1, percobaan 2, dan percobaan 3. Metode *tuning* Ziegler-Nichols dengan semua konfigurasi pengendali yang dilakukan, yaitu pengendali P, PI, PD, dan PID memehuni kriteria perancangan serta untuk metode *tuning* algoritma genetika yang memenuhi kriteria perancangan adalahan pengendali dengan konfigurasi P, PD, dan PID.

BAB V PENUTUP

5.1 Kesimpulan

Pada penelitian ini telah dihasilkan analisis peralihan dan analisis kesalahan untuk sistem pegendalian kecepatan motor DC menggunakan pengendali PID dengan metode *tuning trial & error*, Ziegler-Nichols, dan Algoritma Genetika (GA). Berdasarkan pengujian dan penelitian yang dilakukan, dapat disimpulkan bahwa:

1. Metode *tuning trial & error* yang dilakukan sebanyak lima kali percobaan menghasilkan tiga percobaan yang sesuai kriteria perancangan yaitu percobaan 1 dengan nilai Kp = 7, Ki = 7, Kd = 7, percobaan 2 dengan nilai Kp = 25, Ki = 14, Kd = 1, dan percobaan 3 dangan nilai Kp = 8, Ki = 30, Kd = 1. Percobaan lainnya tidak memenuhi kriteria karena adanya nilai lewatan maksimum, waktu keadaaan mantap, dan kesalahan keadaan mantap yang melebihi batas toleransi yang ditetapkan.
2. Metode *tuning* Ziegler-Nichols yang dilakukan menggunakan berbagai konfigurasi pengendali menghasilkan bahwa hanya pengendali PD memenuhi kriteria perancangan. Meskipun begitu, lewatan maksimumnya hampir mencapai batas ketentuan yang diperbolehkan, yaitu 19.357%. Hal ini disebabkan oleh nilai pengendali D yang sangat kecil sehingga kurang efektif untuk meredam osilasi sistem. Sedangkan analisis kesalahan menunjukkan bahwa di antara keempat konfigurasi pengendali, hanya pengendali PI yang memiliki nilai kesalahan keadaan mantap sangat besar, yaitu 6304.9316. Kesalahan ini terjadi akibat nilai dari pengendali I yang terlalu kecil sehingga tidak mampu menghilangkan kesalahan keadaan mantap secara efektif.
3. Pengendali PD dan PID yang diperoleh melalui metode *tuning* Algoritma Genetika (GA) memiliki respon sistem sesuai dengan kriteria perancangan. Metode ini mampu memberikan parameter pengendali yang optimal dengan respon sistem yang lebih baik dibandingkan dengan metode lainnya.

5.2 Saran

Saran yang dapat penulis berikan untuk penelitian selanjutnya ialah melakukan pengembangan metode *tuning* lainnya agar mampu memberikan respon sistem yang lebih baik lagi dalam pengendalian kecepatan motor DC serta melakukan implementasi pada perangkat keras.

DAFTAR PUSTAKA

[1] M. putri, A. Ma'arif, and R. Dp, "Pengendali Kecepatan Sudut Motor DC Menggunakan Kontrol PID dan Tuning Ziegler Nichols," *Techno (Jurnal Fak. Tek. Univ. Muhammadiyah Purwokerto)*, vol. 23, p. 2022, Apr. 2022, doi: 10.30595/techno.v23i1.10773.

[2] G. A. Siregar, "Analisis Performansi Pengendali PID pada Motor DC Menggunakan Metode Tuning Cohen-Coon," *Pros. Sains Nas. dan Teknol.*, vol. 12, no. 1, p. 633, Dec. 2022, doi: 10.36499/psnst.v12i1.7291.

[3] M. Reza, A. N. Putera, R. Hidayat, and S. Karawang, "Kendali Kecepatan Motor DC Menggunakan Pengendali PID dengan Encoder sebagai Feedback."

[4] M. Irhas, I. Iftitah, and S. A. Azizah Ilham, "Penggunaan Kontrol PID dengan berbagai Metode Untuk Analisis Pengaturan Kecepatan Motor DC," *JFT J. Fis. dan Ter.*, vol. 7, no. 1, pp. 78–86, Nov. 2020, doi: 10.24252/jft.v7i1.13846.

[5] I. Khadari, "Simulasi Kontroler Pid Tuning Menggunakan Logika Fuzzy Dan Algoritma Genetika Sebagai Pengendali Kecepatan Motor Dc," *Setrum Sist. Kendali-Tenaga-elektronika-telekomunikasi-komputer; Vol 8, No 2 Ed. Desember 2019*, Dec. 2019, [Online]. Available: https://jurnal.untirta.ac.id/index.php/jis/article/view/6457

[6] S. B. Joseph, E. G. Dada, A. Abidemi, D. O. Oyewola, and B. M. Khammas, "Metaheuristic algorithms for PID controller parameters tuning: review, approaches and open problems," *Heliyon*, vol. 8, no. 5, p. e09399, 2022, doi: https://doi.org/10.1016/j.heliyon.2022.e09399.

[7] A. Abdulameer, M. Sulaiman, M. S. Mohd Aras, and D. Saleem, "Tuning Methods of PID Controller for DC Motor Speed Control," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 3, pp. 343–349, Aug. 2016, doi: 10.11591/ijeecs.v3.i2.pp343-349.

[8] R. Rosalina, I. Qosim, and M. Mujirudin, "Analisis Pengaturan Kecepatan Motor DC Menggunakan Kontrol PID (Proportional Integral Derivative)," *Pros. Semin. Nas. Teknoka*, vol. 2, no. SE-Teknik Elektro, pp. E89–E94, Nov. 2017, [Online]. Available: https://journal.uhamka.ac.id/index.php/teknoka/article/view/782

[9] Z. Arifin, "Pengaruh Pembebanan terhadap Arus Motor EG-530AD-2F," *J. Disprotek*, vol. 10, pp. 43–49, Jan. 2019, doi: 10.34001/jdpt.v10i1.930.

[10] S. Bachri, "Sistem kendali," *Jte*, vol. 8, no. 2, pp. 25–34, 2004.

[11] K. Ogata, *Modern Control Engineering*, Fifth. PEARSON, 2017.

[12] K. H. Ang, G. Chong, and Y. Li, "PID control system analysis, design, and

technology," *IEEE Trans. Control Syst. Technol.*, vol. 13, no. 4, pp. 559–576, 2005, doi: 10.1109/TCST.2005.847331.
[13] K. J. Åström and T. Hägglund, "The future of PID control," *Control Eng. Pract.*, vol. 9, no. 11, pp. 1163–1175, 2001, doi: https://doi.org/10.1016/S0967-0661(01)00062-4.
[14] L. Wang, "Basics of PID Control," 2020.
[15] S. Anusha, G. Karpagam, and E. Bhuvaneswarri, "Comparison of Tuning Methods of Pid Controller," *BEST Int. J. Manag. Inf. Technol. Eng. (BEST IJMITE)*, vol. 2, no. 8, pp. 1–8, 2014, [Online]. Available: http://www.bestjournals.in/view_archives.php?year=2014&id=14&jtype=2&page=5
[16] N. Kuyvenhoven, "PID Tuning Methods An Automatic PID Tuning Study with MathCad," *Neil Kuyvenhoven Calvin Coll. ENGR. 315*, pp. 1–8, 2002.
[17] Handy Wicaksono, "Analisa Performansi dan Robustness Beberapa Metode Tuning Kontroler PID pada Motor DC," *J. Tek. Elektro*, vol. 4, no. 2, pp. 70–78, 2004, [Online]. Available: http://puslit2.petra.ac.id/ejournal/index.php/elk/article/view/16191
[18] R. F. Nugroho, "Sistem Pengendalian Motor DC Menggunakan PID dengan Metode Ziegler – Nichols (Implementasi Palang Pintu Parkir)," 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:214317159
[19] S. Prayoga and R. P. Hudhajanto, "Auto Tuning Pid Pada Labview Menggunakan Metode Algoritma Genetika," *J. Appl. Sci. Electr. Eng. Comput. Technol.*, vol. 3, no. 01, pp. 24–33, 2022, doi: 10.30871/aseect.v3i01.4386.
[20] A. Aliansyah, N. Nurhayati, S. Jaya, L. Pagiling, W. O. Siti Nur Alam, and M. Nur, "Analisis Tuning Parameter PID Menggunakan Algoritma Genetika pada Pengontrolan Kecepatan Motor DC," *Maj. Ilm. Teknol. Elektro*, vol. 21, p. 287, Dec. 2022, doi: 10.24843/MITE.2022.v21i02.P17.
[21] H. D. Laksono, *Perancangan dan Analisa Sistem Kendali Dengan Berbagai Pengendali*. Padang: Andalas University Press, 2015.
[22] F. Golnaraghi and B. c. Kuo, *Automatic Control System*, 9th Editio. Unitid State: Wiley, 2009.
[23] H. D. Laksono, *Pengantar Teknik Kendali dengan Matlab*. Yogyakarta: Andi Offset, 2013.
[24] E. W. Suseno, A. Ma, and R. D. Puriyanto, "Tuning Parameter Pengendali PID dengan Metode Algoritma Genetik pada Motor DC Tuning of PID Controller Parameters with Genetic Algorithm Method on DC Motor," vol. 8, no. 1.
[25] K. Sekarsari and T. Tata, "Performance analysis of PID control in DC Brushless motor using trial and error method," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1098, p. 42027, Mar. 2021, doi: 10.1088/1757-899X/1098/4/042027.
[26] T. A. Faris Ku Yusoff, M. F. Atan, N. Abdul Rahman, S. F. Salleh, and N. A. Wahab, "Optimization of PID Tuning Using Genetic Algorithm," *J. Appl. Sci. & Process Eng.*, vol. 2, no. 2 SE-Articles, pp. 97–106, Sep. 2015, doi: 10.33736/jaspe.168.2015.
[27] A. Y. Jaen-Cuellar, R. de J. Romero-Troncoso, L. Morales-Velazquez, and

R. A. Osornio-Rios, “PID-Controller Tuning Optimization with Genetic Algorithms in Servo Systems,” *Int. J. Adv. Robot. Syst.*, vol. 10, no. 9, p. 324, Sep. 2013, doi: 10.5772/56697.

LAMPIRAN

Lampiran A

1. Program dan hasil untuk analisis peralihan kecepatan motor DC tanpa pengendali

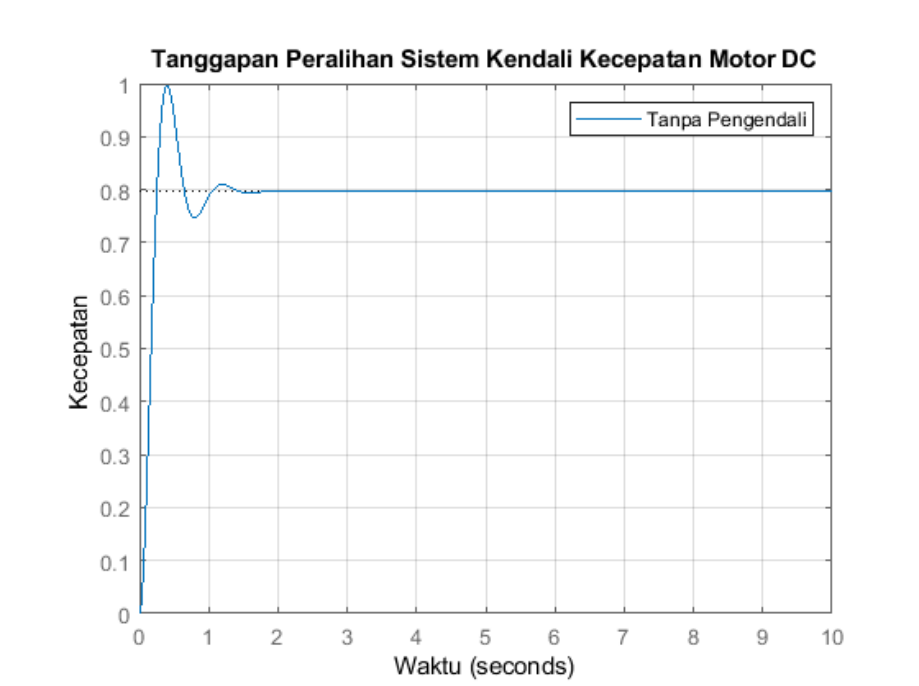
Program:

```
clc
clear all
close all
close all hidden
%
% Fungsi Alih Lingkar Terbuka Motor DC
num_ol = [0.067];
den_ol = [0.00113 0.00785 0.017089];
%
%Tanpa Pengendali
%Fungsi alih lingkar terbuka
disp('Fungsi Alih Lingkar Terbuka')
sys_ol = tf(num_ol,den_ol)

% Fungsi Alih Lingkar Tertutup
disp('Fungsi Alih Lingkar Tertutup')
[num_cl,den_cl] = cloop(num_ol,den_ol,-1);
sys_cl = tf(num_cl,den_cl)

% Informasi Analisa Peralihan Sistem Tanpa Pengendali
fprintf('\n')
disp('Informasi Analisa Peralihan Tanpa Pengendali')
P_cl = stepinfo(sys_cl);
Tr_cl = P_cl.RiseTime;
Tp_cl = P_cl.PeakTime;
Ts_cl = P_cl.SettlingTime;
M_p_cl = P_cl.Peak;
Mp_cl = P_cl.Overshoot;
%
fprintf('Nilai Waktu Naik                = %10.5g detik\n',Tr_cl)
fprintf('Nilai Waktu Puncak              = %10.5g detik\n',Tp_cl)
fprintf('Nilai Waktu Keadaan Mantap      = %10.5g detik\n',Ts_cl)
fprintf('Nilai Puncak                    = %10.5g \n',M_p_cl)
fprintf('Nilai Lewatan Maksimum          = %10.5g Persen\n',Mp_cl)

% Tanggapan Peralihan Sistem Kendali Motor DC Terhadap Masukan
 Undak Satuan
t = 0.000 : 0.001 : 20;
figure
stepplot(sys_cl,t );
```

```
ylabel('Kecepatan (\omega)')
xlabel('Waktu')
grid on
title('Tanggapan Peralihan Sistem Kendali Kecepatan Motor DC')
hleg = legend('Tanpa Pengendali');
%
```

Hasil:

Informasi Analisa Peralihan Tanpa Pengendali
Nilai Waktu Naik = 0.17032 detik
Nilai Waktu Puncak = 0.39775 detik
Nilai Waktu Keadaan Mantap = 0.97482 detik
Nilai Puncak = 0.99684
Nilai Lewatan Maksimum = 25.109 Persen



Gambar A. 1 Grafik Tanggapan Peralihan Sistem Pengendalian Motor DC Tanpa Pengendali

2. Program dan hasil untuk analisis kesalahan kecepatan motor DC tanpa pengendali

Program:

```
clc
clear all
close all
close all hidden
%
% Fungsi Alih Lingkar Terbuka Motor DC
num_ol = [0.067];
```

```
den_ol = [0.00113 0.00785 0.017089];

%Konfigurasi Tanpa Pengendali
%Fungsi Alih lingkar Terbuka
disp('Fungsi Alih Lingkar Terbuka')
sys_ol = tf(num_ol,den_ol)
%
% Fungsi Alih Lingkar Tertutup
disp('Fungsi Alih Lingkar Tertutup')
[num_cl,den_cl] = cloop(num_ol,den_ol,-1);
sys_cl = tf(num_cl,den_cl)
%

% Informasi Analisa Kesalahan
fprintf('\n')
disp('Informasi Analisa Kesalahan Tanpa Pengendali ')
[num_ol,den_ol] = tfdata(sys_ol,'v');
[num_cl,den_cl] = tfdata(sys_cl,'v');
sys_e = tf(1,den_cl);
Errortf(num_ol,den_ol);
%

% %Tanggapan Kesalahan Sistem Terhadap Masukan Undak Satuan
% t = 0.000 : 0.001 : 20;
% figure
% stepplot(sys_cl,t);
% ylabel('Kecepatan')
% xlabel('Waktu')
% grid on
% title('Tanggapan Kesalahan Sistem Kendali Kecepatan Motor DC')
% hleg = legend('Tanpa Pengendali');
%
```

Hasil:

Informasi Analisa Kesalahan Tanpa Pengendali
Tipe Sistem adalah 0
Konstanta Kesalahan Posisi (Kp) adalah 3.9207
Konstanta Kesalahan Kecepatan (Kv) adalah 0.0000
Konstanta Kesalahan Percepatan (Ka) adalah 0.0000
Kesalahan Keadaan Mantap Untuk Masukan Undak adalah 0.2032
Kesalahan Keadaan Mantap Untuk Masukan Laju adalah Inf
Kesalahan Keadaan Mantap Untuk Masukan Parabolik adalah Inf

Lampiran B

1. Program dan hasil untuk analisis peralihan kecepatan motor DC dengan pengendali PID metode *tuning trial & error*

Program:

```
clc;
clear all;
close all;
close all hidden;
%
% Masukan unit step (setpoint)
unit_step = 1;

% Jumlah percobaan tuning
num_trials = input('Masukkan jumlah percobaan tuning PID: ');

% Data Fungsi Alih Lingkar Terbuka Motor DC
num_ol = [0.067];
den_ol = [0.00113 0.00785 0.017089];

% Fungsi Alih Lingkar Terbuka Motor DC
sys_ol = tf(num_ol, den_ol);
% Fungsi Alih Lingkar Tertutup Motor DC
[num_cl,den_cl] = cloop(num_ol,den_ol,-1);
sys_cl = tf(num_cl,den_cl);

% Loop untuk melakukan tuning PID secara trial and error
for i = 1:num_trials
    fprintf('Percobaan %d: \n', i);

    % Input nilai Kp, Ki, Kd dari pengguna
    Kp = input('Masukkan nilai Kp: ');
    Ki = input('Masukkan nilai Ki: ');
    Kd = input('Masukkan nilai Kd: ');

    % Pengendali Proporsional Integral Diferensial (PID)
    C_pid = pid(Kp, Ki, Kd);

    % Fungsi Alih Lingkar Terbuka Dengan PID
    sys_ol_pid = series(C_pid, sys_ol);

    % Fungsi Alih Lingkar Tertutup Dengan PID
    sys_cl_pid = feedback(sys_ol_pid, 1);

    % Tanggapan Terhadap Masukan Undak Satuan (Unit Step)
    disp('Analisis Kinerja Sistem Kendali Motor DC Dengan PID');
    p_pid_step = stepinfo(sys_cl_pid);
    Tr_step = p_pid_step.RiseTime;
    Tp_step = p_pid_step.PeakTime;
    Ts_step = p_pid_step.SettlingTime;
    N_p_step = p_pid_step.Peak;
    Mp_step = p_pid_step.Overshoot;

    fprintf('Waktu Naik = %10.5g detik\n', Tr_step);
    fprintf('Waktu Puncak = %10.5g detik\n', Tp_step);
```

```
    fprintf('Waktu Keadaan Mantap = %10.5g detik\n', Ts_step);
    fprintf('Nilai Puncak = %10.5g\n', N_p_step);
    fprintf('Lewatan Maksimum = %10.5g\n\n', Mp_step);

    % Plot Tanggapan Sistem
    figure('Name', ['Tanggapan Sistem Kendali Motor DC -
 Percobaan ', num2str(i)]);

    % Plot tanggapan terhadap masukan undak satuan
    t = 0.000 : 0.001 : 20;
    stepplot(sys_cl, sys_cl_pid * unit_step, t);
    title(['Tanggapan Sistem Kendali Motor DC (Percobaan ',
 num2str(i), ')']);
    ylabel('Kecepatan (\omega)');
    xlabel('Waktu');
    grid on;
    hleg = legend ('Tanpa Pengendali', 'Dengan Pengendali (Tuning
 Trial&Error)');
end
```

Hasil:

Masukkan jumlah percobaan tuning PID: 5
Percobaan 1:
Masukkan nilai Kp: 7
Masukkan nilai Ki: 7
Masukkan nilai Kd: 7
Analisis Kinerja Sistem Kendali Motor DC Dengan PID
Waktu Naik = 0.0055438 detik
Waktu Puncak = 0.023848 detik
Waktu Keadaan Mantap = 0.012228 detik
Nilai Puncak = 0.98547
Lewatan Maksimum = 0

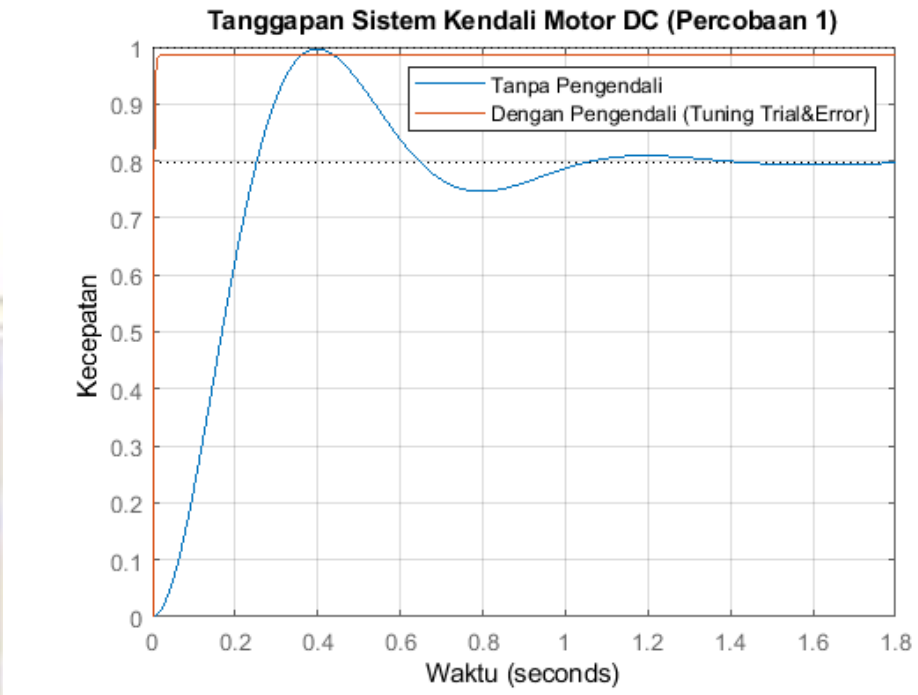
Percobaan 2:
Masukkan nilai Kp: 25
Masukkan nilai Ki: 14
Masukkan nilai Kd: 1
Analisis Kinerja Sistem Kendali Motor DC Dengan PID
Waktu Naik = 0.023886 detik
Waktu Puncak = 0.060308 detik
Waktu Keadaan Mantap = 0.12453 detik
Nilai Puncak = 1.1142
Lewatan Maksimum = 11.424

Percobaan 3:
Masukkan nilai Kp: 8
Masukkan nilai Ki: 30

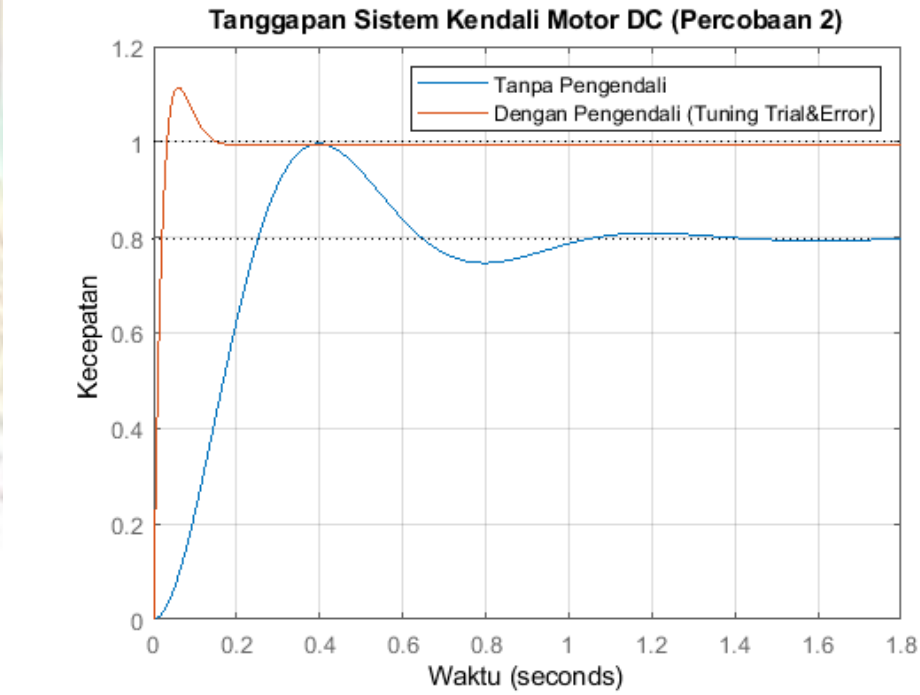
Masukkan nilai Kd: 1
Analisis Kinerja Sistem Kendali Motor DC Dengan PID
Waktu Naik = 0.035063 detik
Waktu Puncak = 0.15383 detik
Waktu Keadaan Mantap = 0.25344 detik
Nilai Puncak = 1.023
Lewatan Maksimum = 2.3008

Percobaan 4:
Masukkan nilai Kp: 2
Masukkan nilai Ki: 1
Masukkan nilai Kd: 6
Analisis Kinerja Sistem Kendali Motor DC Dengan PID
Waktu Naik = 0.0065475 detik
Waktu Puncak = 11.301 detik
Waktu Keadaan Mantap = 6.4269 detik
Nilai Puncak = 1.0119
Lewatan Maksimum = 1.1864

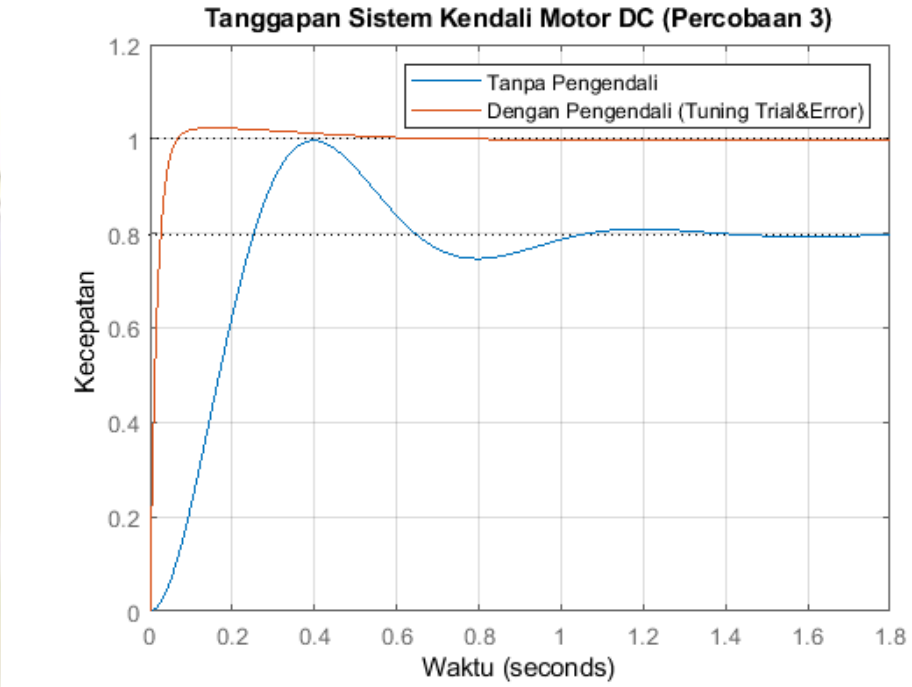
Percobaan 5:
Masukkan nilai Kp: 10
Masukkan nilai Ki: 2
Masukkan nilai Kd: 5
Analisis Kinerja Sistem Kendali Motor DC Dengan PID
Waktu Naik = 0.0078255 detik
Waktu Puncak = 0.032851 detik
Waktu Keadaan Mantap = 0.018806 detik
Nilai Puncak = 0.98314
Lewatan Maksimum = 0



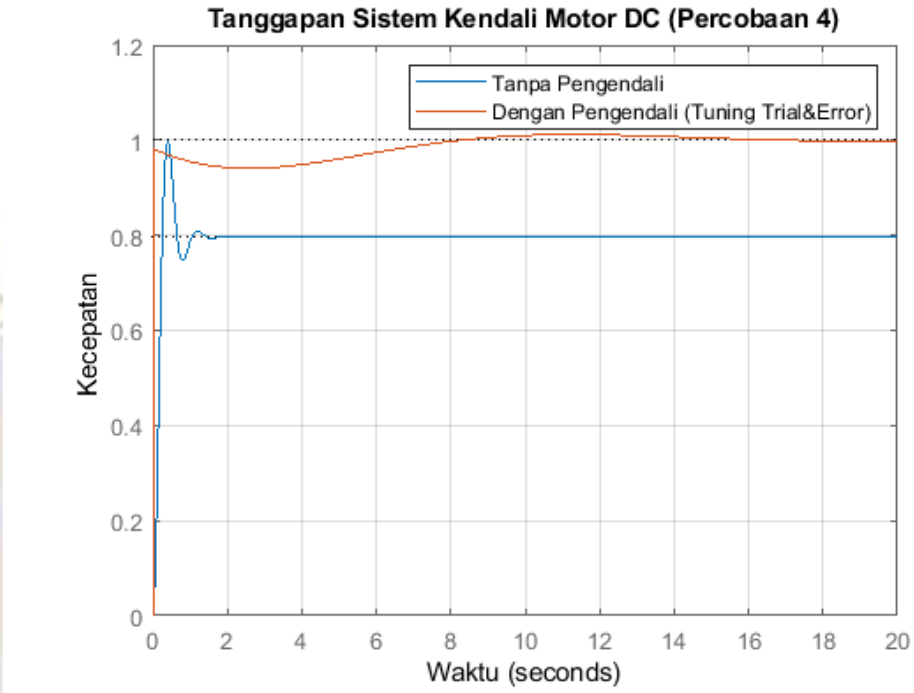
Gambar B. 1 Grafik Tanggapan Peralihan Sistem Pengendalian Motor DC dengan PID Metode T*uning Trial & Error* – Percobaan 1



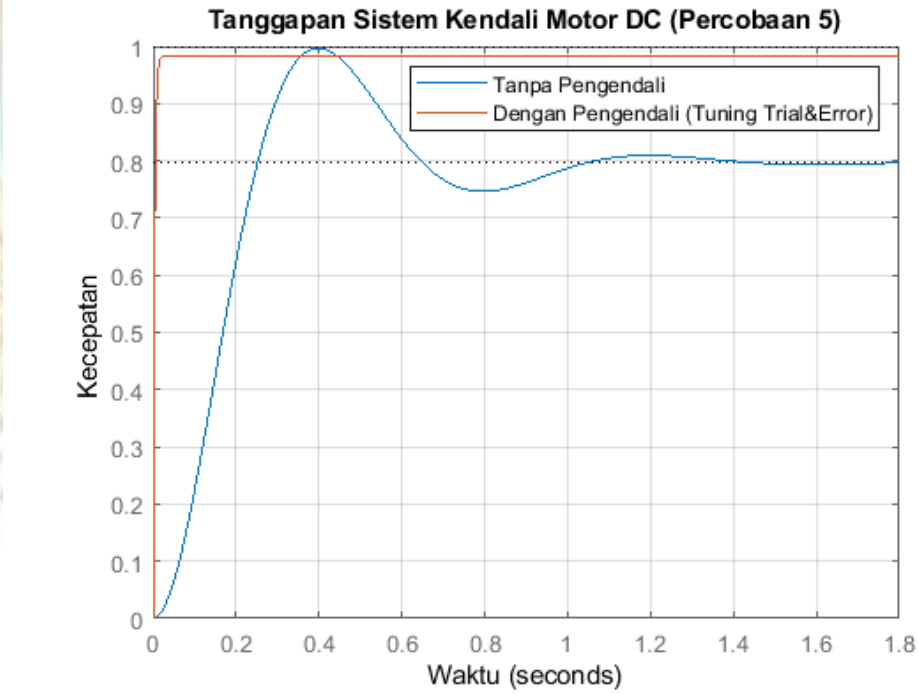
Gambar B. 2 Grafik Tanggapan Peralihan Sistem Pengendalian Motor DC dengan PID Metode *Tuning Trial & Error* – Percobaan 2



Gambar B. 3 Grafik Tanggapan Peralihan Sistem Pengendalian Motor DC dengan PID Metode *Tuning Trial & Error* – Percobaan 3



Gambar B. 4 Grafik Tanggapan Peralihan Sistem Pengendalian Motor DC dengan PID Metode *Tuning Trial & Error* – Percobaan 4



Gambar B. 5 Grafik Tanggapan Peralihan Sistem Pengendalian Motor DC dengan PID Metode *Tuning Trial & Error* – Percobaan 5

2. Program dan hasil untuk analisis kesalahan kecepatan motor DC dengan pengendali PID metode *tuning trial & error*

Program:

```
clc;
clear all;
close all;
close all hidden;
%
% Masukan unit step (setpoint)
unit_step = 1;

% Jumlah percobaan tuning
num_trials = input('Masukkan jumlah percobaan tuning PID: ');

% Data Fungsi Alih Lingkar Terbuka Motor DC
num_ol = [0.067];
den_ol = [0.00113 0.00785 0.017089];

% Fungsi Alih Lingkar Terbuka Motor DC
sys_ol = tf(num_ol, den_ol);
% Fungsi Alih Lingkar Tertutup Motor DC
[num_cl,den_cl] = cloop(num_ol,den_ol,-1);
sys_cl = tf(num_cl,den_cl);

% Loop untuk melakukan tuning PID secara trial and error
for i = 1:num_trials
    fprintf('Percobaan %d: \n', i);

    % Input nilai Kp, Ki, Kd dari pengguna
    Kp = input('Masukkan nilai Kp: ');
    Ki = input('Masukkan nilai Ki: ');
    Kd = input('Masukkan nilai Kd: ');

    % Pengendali Proporsional Integral Diferensial (PID)
    C_pid = pid(Kp, Ki, Kd);

    % Fungsi Alih Lingkar Terbuka Dengan PID
    sys_ol_pid = series(C_pid, sys_ol);

    % Fungsi Alih Lingkar Tertutup Dengan PID
    sys_cl_pid = feedback(sys_ol_pid, 1);

    % Analisis Kesalahan Sitem Kendali Motor DC
    fprintf('\n')
    disp('Analisis Kesalahan Kinerja Sistem Kendali Motor DC
Pengendali PID Tuning Trial&Error')
    [num_ol_pid, den_ol_pid]= tfdata (sys_ol_pid, 'v');
    [num_cl_pid, den_cl_pid]= tfdata (sys_cl_pid, 'v');
    sys_e_pid = tf(1, den_cl_pid);
```

```
    Errortf(num_ol_pid, den_ol_pid);

    % Plot Tanggapan Sistem
    figure('Name', ['Tanggapan Sistem Kendali Motor DC -
Percobaan ', num2str(i)]);

%     % Plot tanggapan terhadap masukan undak satuan
%     t = 0.000 : 0.001 : 20;
%     stepplot(sys_cl, sys_cl_pid * unit_step, t);
%      title(['Tanggapan Sistem Kendali Motor DC (Percobaan ',
num2str(i), ')']);
%     ylabel('Kecepatan');
%     xlabel('Waktu');
%     grid on;
%       hleg = legend ('Tanpa Pengendali', 'Dengan Pengendali
(Tuning Trial&Error)');
end
```

Hasil:

Masukkan jumlah percobaan tuning PID: 5

Percobaan 1:
Masukkan nilai Kp: 7
Masukkan nilai Ki: 7
Masukkan nilai Kd: 7
Analisis Kesalahan Kinerja Sistem Kendali Motor DC Pengendali PID Tuning Trial&Error
Tipe Sistem adalah 1
Konstanta Kesalahan Posisi (Kp) adalah Inf
Konstanta Kesalahan Kecepatan (Kv) adalah 27.4446
Konstanta Kesalahan Percepatan (Ka) adalah 0.0000
Kesalahan Keadaan Mantap Untuk Masukan Undak adalah 0.0000
Kesalahan Keadaan Mantap Untuk Masukan Laju adalah 0.0364
Kesalahan Keadaan Mantap Untuk Masukan Parabolik adalah Inf

Percobaan 2:
Masukkan nilai Kp: 25
Masukkan nilai Ki: 14
Masukkan nilai Kd: 1
Analisis Kesalahan Kinerja Sistem Kendali Motor DC Pengendali PID Tuning Trial&Error
Tipe Sistem adalah 1
Konstanta Kesalahan Posisi (Kp) adalah Inf
Konstanta Kesalahan Kecepatan (Kv) adalah 54.8891
Konstanta Kesalahan Percepatan (Ka) adalah 0.0000
Kesalahan Keadaan Mantap Untuk Masukan Undak adalah 0.0000

Kesalahan Keadaan Mantap Untuk Masukan Laju adalah 0.0182
Kesalahan Keadaan Mantap Untuk Masukan Parabolik adalah Inf

Percobaan 3:
Masukkan nilai Kp: 8
Masukkan nilai Ki: 30
Masukkan nilai Kd: 1
Analisis Kesalahan Kinerja Sistem Kendali Motor DC Pengendali PID Tuning Trial&Error
Tipe Sistem adalah 1
Konstanta Kesalahan Posisi (Kp) adalah Inf
Konstanta Kesalahan Kecepatan (Kv) adalah 117.6195
Konstanta Kesalahan Percepatan (Ka) adalah 0.0000
Kesalahan Keadaan Mantap Untuk Masukan Undak adalah 0.0000
Kesalahan Keadaan Mantap Untuk Masukan Laju adalah 0.0085
Kesalahan Keadaan Mantap Untuk Masukan Parabolik adalah Inf

Percobaan 4:
Masukkan nilai Kp: 2
Masukkan nilai Ki: 1
Masukkan nilai Kd: 6
Analisis Kesalahan Kinerja Sistem Kendali Motor DC Pengendali PID Tuning Trial&Error
Tipe Sistem adalah 1
Konstanta Kesalahan Posisi (Kp) adalah Inf
Konstanta Kesalahan Kecepatan (Kv) adalah 3.9207
Konstanta Kesalahan Percepatan (Ka) adalah 0.0000
Kesalahan Keadaan Mantap Untuk Masukan Undak adalah 0.0000
Kesalahan Keadaan Mantap Untuk Masukan Laju adalah 0.2551
Kesalahan Keadaan Mantap Untuk Masukan Parabolik adalah Inf

Percobaan 5:
Masukkan nilai Kp: 10
Masukkan nilai Ki: 2
Masukkan nilai Kd: 5
Analisis Kesalahan Kinerja Sistem Kendali Motor DC Pengendali PID Tuning Trial&Error
Tipe Sistem adalah 1
Konstanta Kesalahan Posisi (Kp) adalah Inf
Konstanta Kesalahan Kecepatan (Kv) adalah 7.8413
Konstanta Kesalahan Percepatan (Ka) adalah 0.0000
Kesalahan Keadaan Mantap Untuk Masukan Undak adalah 0.0000

Kesalahan Keadaan Mantap Untuk Masukan Laju adalah 0.1275
Kesalahan Keadaan Mantap Untuk Masukan Parabolik adalah Inf

Lampiran C

1. Program dan hasil untuk analisis peralihan kecepatan motor DC dengan pengendali PID metode Ziegler-Nichols

Program:

```
clc;
clear all;
close all;
close all hidden;

% Parameter Motor DC
Ra = 0.45;        % Resistansi armature (Ohm)
La = 0.1;       % Induktansi armature (H)
J  = 0.0113;       % Momen inersia rotor (kg·m^2)
B  = 0.028;      % Damping mekanis (N·m·s/rad)
Kb = 0.067;         % Konstanta back-EMF (V·s/rad)
Kt = 0.067;

% Fungsi alih motor DC (kecepatan)
s = tf('s');
num_motor = Kt; % Pembilang (konstanta torsi)
den_motor = [La*J, (Ra*J + La*B), (Ra*B + Kb*Kt)]; % Penyebut
sys_ol = tf(num_motor, den_motor);
[num_cl, den_cl]= cloop (num_motor, den_motor, -1);
sys_cl = tf (num_cl, den_cl);

% Simulasi Step untuk Identifikasi Kurva Reaksi
t = 0:0.001:8; % Waktu simulasi
[y, t] = step(sys_ol, t); % Respons step

% Plot Respons Step
figure;
plot(t, y, 'LineWidth', 2);
title('Respons Step untuk Identifikasi Kurva Reaksi');
xlabel('Waktu (s)');
ylabel('Kecepatan (\omega)');
grid on;

% Identifikasi Parameter Kurva Reaksi (L dan T) Secara Manual
disp('Identifikasi parameter L (dead time) dan T (time constant)
dari grafik.');

% === Masukkan L dan T Berdasarkan Kurva Reaksi ===
L = input('Masukkan nilai L (dead time): '); % Dead time dari
grafik
T = input('Masukkan nilai T (time constant): '); % Time constant
dari grafik

% Perhitungan Parameter P Menggunakan Metode Ziegler-Nichols
```

```
Kp_P = T/L;

% Perhitungan Parameter PI Menggunakan Metode Ziegler-Nichols
Kp_PI = 0.9 * T/L;
Ti_PI = L/0.3;
Ki_PI = Kp_PI/Ti_PI;

% Perhitungan Parameter PD Menggunakan Modifikasi Metode Ziegler-
Nichols (dari PID)
Kp_PD = 1.2 * (T / L);
Td_PD = 0.5 * L;
Kd_PD = Kp_PD * Td_PD;

% Perhitungan Parameter PID Menggunakan Metode Ziegler-Nichols
Kp_PID = 1.2 * (T / L);
Ti_PID = 2 * L;
Td_PID = 0.5 * L;
Ki_PID = Kp_PID / Ti_PID;
Kd_PID = Kp_PID * Td_PID;

% Menampilkan Parameter Pengendali
fprintf('\nParameter PID berdasarkan Ziegler-Nichols Metode Kurva
Reaksi:\n');
fprintf('P: Kp = %.3f\n', Kp_P);
fprintf('PI: Kp = %.3f, Ki = %.3f\n', Kp_PI, Ki_PI);
fprintf('PD: Kp = %.3f, Kd = %.3f\n', Kp_PD, Kd_PD);
fprintf('PID: Kp = %.3f, Ki = %.3f, Kd = %.3f\n', Kp_PID, Ki_PID,
Kd_PID);

% Membuat Pengendali
p_controller     = pid(Kp_P, 0, 0);
pi_controller    = pid(Kp_PI, Ki_PI, 0);
pd_controller    = pid(Kp_PD, 0, Kd_PD);
pid_controller   = pid(Kp_PID, Ki_PID, Kd_PID);

% Fungsi Alih Closed-Loop dengan pengendali
sys_cl_p     = feedback(p_controller * sys_ol, 1);
sys_cl_pi    = feedback(pi_controller * sys_ol, 1);
sys_cl_pd    = feedback(pd_controller  * sys_ol, 1);
sys_cl_pid   = feedback(pid_controller * sys_ol, 1);

% Simulasi dan Plot Respons Step Closed-Loop
figure;
step(sys_cl, sys_cl_p, sys_cl_pi, sys_cl_pd, sys_cl_pid, t);
title('Tanggapan Peralihan Sistem Kendali Kecepatan Motor DC
dengan Tuning Z-N menggunakan berbagai Pengendali');
xlabel('Waktu (s)');
ylabel('Kecepatan (\omega)');
grid on;
```

```
hleg = legend ('Tanpa Pengendali', 'Dengan Pengendali P', 'Dengan
Pengendali PI', 'Dengan Pengendali PD', 'Dengan Pengendali PID');

% Analisis Kinerja Sistem
fprintf('\nAnalisis Kinerja Sistem :\n');

% Analisis sistem tanpa pengendali
info_ol = stepinfo(sys_cl);
fprintf('\nSistem Tanpa Pengendali:\n');
fprintf('Rise Time: %.3f s\n', info_ol.RiseTime);
fprintf('Settling Time: %.3f s\n', info_ol.SettlingTime);
fprintf('Overshoot: %.3f%%\n', info_ol.Overshoot);

% Analisis sistem dengan pengendali P
info_P = stepinfo(sys_cl_p);
fprintf('\nSistem Dengan Pengendali P:\n');
fprintf('Rise Time: %.3f s\n', info_P.RiseTime);
fprintf('Settling Time: %.3f s\n', info_P.SettlingTime);
fprintf('Overshoot: %.3f%%\n', info_P.Overshoot);

% Analisis sistem dengan pengendali PI
info_PI = stepinfo(sys_cl_pi);
fprintf('\nSistem Dengan Pengendali PI:\n');
fprintf('Rise Time: %.3f s\n', info_PI.RiseTime);
fprintf('Settling Time: %.3f s\n', info_PI.SettlingTime);
fprintf('Overshoot: %.3f%%\n', info_PI.Overshoot);

% Analisis sistem dengan pengendali PD
info_PD = stepinfo(sys_cl_pd);
fprintf('\nSistem Dengan Pengendali PD:\n');
fprintf('Rise Time: %.3f s\n', info_PD.RiseTime);
fprintf('Settling Time: %.3f s\n', info_PD.SettlingTime);
fprintf('Overshoot: %.3f%%\n', info_PD.Overshoot);

% Analisis sistem dengan pengendali PID
info_PID = stepinfo(sys_cl_pid);
fprintf('\nSistem Dengan Pengendali PID:\n');
fprintf('Rise Time: %.3f s\n', info_PID.RiseTime);
fprintf('Settling Time: %.3f s\n', info_PID.SettlingTime);
fprintf('Overshoot: %.3f%%\n', info_PID.Overshoot);
```

Hasil:

Identifikasi parameter L (dead time) dan T (time constant) dari grafik.
Masukkan nilai L (dead time): 0.0765
Masukkan nilai T (time constant): 0.4435

Parameter PID berdasarkan Ziegler-Nichols Metode Kurva Reaksi:
P: Kp = 5.797

PI: Kp = 5.218, Ki = 20.461
PD: Kp = 6.957, Kd = 0.266
PID: Kp = 6.957, Ki = 45.470, Kd = 0.266

Analisis Kinerja Sistem :

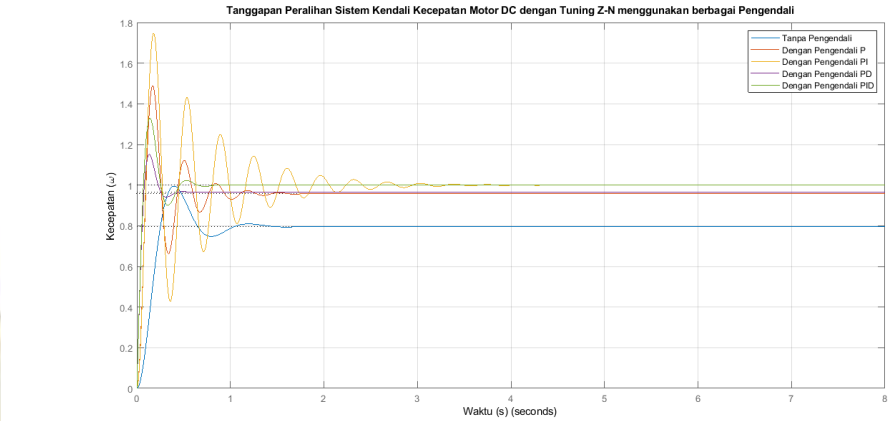
Sistem Tanpa Pengendali:
Rise Time: 0.171 s
Settling Time: 0.975 s
Overshoot: 24.920%

Sistem Dengan Pengendali P:
Rise Time: 0.063 s
Settling Time: 1.058 s
Overshoot: 55.425%

Sistem Dengan Pengendali PI:
Rise Time: 0.064 s
Settling Time: 2.511 s
Overshoot: 74.625%

Sistem Dengan Pengendali PD:
Rise Time: 0.057 s
Settling Time: 0.347 s
Overshoot: 19.357%

Sistem Dengan Pengendali PID:
Rise Time: 0.055 s
Settling Time: 0.560 s
Overshoot: 32.973%



Gambar C. 1 Grafik Tanggapan Peralihan Sistem Pengendalian Motor DC dengan PID Metode *Tuning* Ziegler-Nichols

2. Program dan hasil untuk analisis kesalahan kecepatan motor DC dengan pengendali PID metode Ziegler-Nichols

Program:

```
clc;
clear all;
close all;
close all hidden;

% Parameter Motor DC
Ra = 0.45;        % Resistansi armature (Ohm)
La = 0.1;       % Induktansi armature (H)
J  = 0.0113;       % Momen inersia rotor (kg·m^2)
B  = 0.028;      % Damping mekanis (N·m·s/rad)
Kb = 0.067;         % Konstanta back-EMF (V·s/rad)
Kt = 0.067;

% Fungsi alih motor DC
s = tf('s');
num_motor = Kt; % Pembilang (konstanta torsi)
den_motor = [La*J, (Ra*J + La*B), (Ra*B + Kb*Kt)]; % Penyebut
sys_ol = tf(num_motor, den_motor);
[num_cl, den_cl]= cloop (num_motor, den_motor, -1);
sys_cl = tf (num_cl, den_cl);

% Simulasi Step untuk Identifikasi Kurva Reaksi
t = 0:0.001:8; % Waktu simulasi
[y, t] = step(sys_ol, t); % Respons step

% === Masukkan L dan T Berdasarkan Kurva Reaksi ===
L = input('Masukkan nilai L (dead time): '); % Dead time dari
grafik
```

```
T = input('Masukkan nilai T (time constant): '); % Time constant
dari grafik

% Perhitungan Parameter Pengendali Menggunakan Metode Ziegler-
Nichols
Kp_P = T/L;

Kp_PI = 0.9 * T/L;
Ti_PI = L/0.3;
Ki_PI = Kp_PI/Ti_PI;

Kp_PD = 1.2 * (T / L);
Td_PD = 0.5 * L;
Kd_PD = Kp_PD * Td_PD;

Kp_PID = 1.2 * (T / L);
Ti_PID = 2 * L;
Td_PID = 0.5 * L;
Ki_PID = Kp_PID / Ti_PID;
Kd_PID = Kp_PID * Td_PID;

% Menampilkan Parameter Pengendali
fprintf('\nParameter   Pengendali   berdasarkan   Ziegler-Nichols
Metode Kurva Reaksi:\n');
fprintf('P: Kp = %.3f\n', Kp_P);
fprintf('PI: Kp = %.3f, Ki = %.3f\n', Kp_PI, Ki_PI);
fprintf('PD: Kp = %.3f, Kd = %.3f\n', Kp_PD, Kd_PD);
fprintf('PID: Kp = %.3f, Ki = %.3f, Kd = %.3f\n', Kp_PID, Ki_PID,
Kd_PID);

% Membuat Pengendali
controller_P  = pid(Kp_P, 0, 0);
controller_PI = pid(Kp_PI, Ki_PI, 0);
controller_PD = pid(Kp_PD, 0, Kd_PD);
controller_PID = pid(Kp_PID, Ki_PID, Kd_PID);

% Fungsi alih dengan PID
sys_ol_P = controller_P * sys_ol;
sys_cl_P = feedback(controller_P * sys_ol, 1);
sys_ol_PI = controller_PI * sys_ol;
sys_cl_PI = feedback(controller_PI * sys_ol, 1);
sys_ol_PD = controller_PD * sys_ol;
sys_cl_PD = feedback(controller_PD * sys_ol, 1);
sys_ol_PID = controller_PID * sys_ol;
sys_cl_PID = feedback(controller_PID * sys_ol, 1);

% Analisis Kesalahan Sitem Kendali Motor DC
fprintf('\n')
```

```
disp('Analisis Kesalahan Kinerja Sistem Kendali Motor DC
Pengendali PID Tuning Ziegler-Nichols Menggunakan berbagai
Pengendali')
[num_ol_P, den_ol_P]= tfdata (sys_ol_P, 'v');
[num_cl_P, den_cl_P]= tfdata (sys_cl_P, 'v');
[num_ol_PI, den_ol_PI]= tfdata (sys_ol_PI, 'v');
[num_cl_PI, den_cl_PI]= tfdata (sys_cl_PI, 'v');
[num_ol_PD, den_ol_PD]= tfdata (sys_ol_PD, 'v');
[num_cl_PD, den_cl_PD]= tfdata (sys_cl_PD, 'v');
[num_ol_PID, den_ol_PID]= tfdata (sys_ol_PID, 'v');
[num_cl_PID, den_cl_PID]= tfdata (sys_cl_PID, 'v');
sys_e_P = tf(1, den_cl_P);
sys_e_PI = tf(1, den_cl_PI);
sys_e_PD = tf(1, den_cl_PD);
sys_e_PID = tf(1, den_cl_PID);
Errortf(num_ol_P, den_ol_P);
Errortf(num_ol_PI, den_ol_PI);
Errortf(num_ol_PD, den_ol_PD);
Errortf(num_ol_PID, den_ol_PID);

% %Simulasi dan plot respons step
% figure;
% step(sys_cl, sys_e_PID, t);
% title('Tanggapan Kesalahan Sistem Kendali Kecepatan Motor DC');
% xlabel('Waktu');
% ylabel('Kecepatan (\omega)');
% grid on;
% hleg = legend('Tanpa Pengendali', 'Dengan Pengendali (Tuning
Ziegler-Nichols)');
```

Hasil:

Masukkan nilai L (dead time): 0.0765
Masukkan nilai T (time constant): 0.4435

Parameter Pengendali berdasarkan Ziegler-Nichols Metode Kurva Reaksi:
P: Kp = 5.797
PI: Kp = 5.218, Ki = 20.461
PD: Kp = 6.957, Kd = 0.266
PID: Kp = 6.957, Ki = 45.470, Kd = 0.266

Analisis Kesalahan Kinerja Sistem Kendali Motor DC Pengendali PID Tuning Ziegler-Nichols Menggunakan berbagai Pengendali
Tipe Sistem adalah 0
Konstanta Kesalahan Posisi (Kp) adalah 22.7295
Konstanta Kesalahan Kecepatan (Kv) adalah 0.0000
Konstanta Kesalahan Percepatan (Ka) adalah 0.0000
Kesalahan Keadaan Mantap Untuk Masukan Undak adalah 0.0421
Kesalahan Keadaan Mantap Untuk Masukan Laju adalah Inf

Kesalahan Keadaan Mantap Untuk Masukan Parabolik adalah Inf

Tipe Sistem adalah 1
Konstanta Kesalahan Posisi (Kp) adalah Inf
Konstanta Kesalahan Kecepatan (Kv) adalah 80.2218
Konstanta Kesalahan Percepatan (Ka) adalah 0.0000
Kesalahan Keadaan Mantap Untuk Masukan Undak adalah 0.0000
Kesalahan Keadaan Mantap Untuk Masukan Laju adalah 0.0125
Kesalahan Keadaan Mantap Untuk Masukan Parabolik adalah Inf

Tipe Sistem adalah 0
Konstanta Kesalahan Posisi (Kp) adalah 27.2754
Konstanta Kesalahan Kecepatan (Kv) adalah 0.0000
Konstanta Kesalahan Percepatan (Ka) adalah 0.0000
Kesalahan Keadaan Mantap Untuk Masukan Undak adalah 0.0354
Kesalahan Keadaan Mantap Untuk Masukan Laju adalah Inf
Kesalahan Keadaan Mantap Untuk Masukan Parabolik adalah Inf

Tipe Sistem adalah 1
Konstanta Kesalahan Posisi (Kp) adalah Inf
Konstanta Kesalahan Kecepatan (Kv) adalah 178.2708
Konstanta Kesalahan Percepatan (Ka) adalah 0.0000
Kesalahan Keadaan Mantap Untuk Masukan Undak adalah 0.0000
Kesalahan Keadaan Mantap Untuk Masukan Laju adalah 0.0056
Kesalahan Keadaan Mantap Untuk Masukan Parabolik adalah Inf

Lampiran D

1. Program dan hasil untuk analisis peralihan kecepatan motor DC dengan pengendali PID metode Algoritma Genetika

Program:

```
clc;
clear all;
close all;
close all hidden;

% Defenisi variabel laplace
s = tf('s');

% Fungsi alih plant
num_ol = [0.067];
den_ol = [0.00113 0.00785 0.0171];

sys_ol = tf(num_ol, den_ol);
[num_cl,den_cl] = cloop(num_ol,den_ol,-1);
sys_cl = tf(num_cl,den_cl);

% Parameter Pengendali (sesuaikan dengan hasil optimasi atau
nilai yang diinginkan)

% Parameter Pengendali P
Kp_P = 15.0065;

% Parameter Pengendali PI
Kp_PI = 9.4871;
Ki_PI = 4.0454e-05;

% Parameter Pengendali PD
Kp_PD = 20;
Kd_PD = 30;

% Parameter Pengendali PID
Kp_PID = 11.3678;
Ki_PID = 24.7629;
Kd_PID = 1.6365;

% Fungsi alih pengendali
cont_P = Kp_P;
cont_PI = Kp_PI + Ki_PI/s;
cont_PD = Kp_PD + Kd_PD*s;
cont_PID = Kp_PID + Ki_PID/s + Kd_PID*s;


% Fungsi alih lingkar tertutup
```

```
closed_loop_P = feedback(sys_ol * cont_P, 1);
closed_loop_PI = feedback(sys_ol * cont_PI, 1);
closed_loop_PD = feedback(sys_ol * cont_PD, 1);
closed_loop_PID = feedback(sys_ol * cont_PID, 1);


% Respons langkah (step response)
t = 0.000 : 0.001 : 2;
figure;
step(sys_cl,  closed_loop_P,  closed_loop_PI,  closed_loop_PD,
closed_loop_PID, t);
title('Tanggapan Peralihan Sistem Kendali Motor DC');
xlabel('Waktu (s)');
ylabel('Kecepatan');
grid on;
hleg = legend('Tanpa pengendali','Dengan Pengendali P', 'Dengan
Pengendali PI', 'Dengan Pengendali PD','Dengan Pengendali PID');

% Analisis kinerja respons transien
info_P = stepinfo(closed_loop_P);
info_PI = stepinfo(closed_loop_PI);
info_PD = stepinfo(closed_loop_PD);
info_PID = stepinfo(closed_loop_PID);


% Menampilkan hasil analisis transien di Command Window
fprintf('\nAnalisis Respons Transien:\n');

fprintf('\n--- Pengendali P ---\n');
fprintf('Rise Time: %.3f s\n', info_P.RiseTime);
fprintf('Settling Time: %.3f s\n', info_P.SettlingTime);
fprintf('Overshoot: %.2f%%\n', info_P.Overshoot);

fprintf('\n--- Pengendali PI ---\n');
fprintf('Rise Time: %.3f s\n', info_PI.RiseTime);
fprintf('Settling Time: %.3f s\n', info_PI.SettlingTime);
fprintf('Overshoot: %.2f%%\n', info_PI.Overshoot);

fprintf('\n--- Pengendali PD ---\n');
fprintf('Rise Time: %.3f s\n', info_PD.RiseTime);
fprintf('Settling Time: %.3f s\n', info_PD.SettlingTime);
fprintf('Overshoot: %.2f%%\n', info_PD.Overshoot);

fprintf('\n--- Pengendali PID ---\n');
fprintf('Rise Time: %.3f s\n', info_PID.RiseTime);
fprintf('Settling Time: %.3f s\n', info_PID.SettlingTime);
fprintf('Overshoot: %.2f%%\n', info_PID.Overshoot);
```

Hasil:

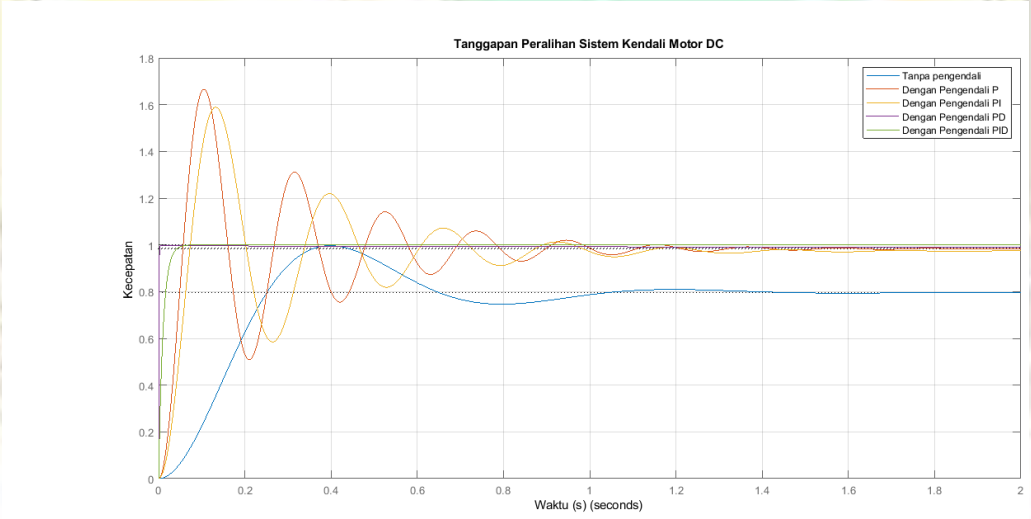
Analisis Respons Transien:

--- Pengendali P ---
Rise Time: 0.038 s
Settling Time: 1.075 s
Overshoot: 69.39%

--- Pengendali PI ---
Rise Time: 0.050 s
Settling Time: NaN s
Overshoot: 58.89%

--- Pengendali PD ---
Rise Time: 0.001 s
Settling Time: 0.002 s
Overshoot: 0.81%

--- Pengendali PID ---
Rise Time: 0.023 s
Settling Time: 0.040 s
Overshoot: 0.00%



Gambar D. 1 Grafik Tanggapan Peralihan Sistem Pengendalian Motor DC dengan PID Metode *Tuning* Algoritma Genetika

2. Program dan hasil untuk analisis kesalahan kecepatan motor DC dengan pengendali PID metode Algoritma Genetika

Program:

```
clc;
```

```
clear all;
close all;
close all hidden;

% Definisi variabel Laplace
s = tf('s');

% Fungsi alih plant
num_ol = [0.067];
den_ol = [0.00113 0.00785 0.017089];
sys_ol = tf(num_ol, den_ol);
[num_cl, den_cl] = cloop(num_ol, den_ol, -1);
sys_cl = tf(num_cl, den_cl)

% Parameter pengendali
% Parameter Pengendali P
Kp_P = 15.0065;
% Parameter Pengendali PI
Kp_PI = 9.4871;
Ki_PI = 4.0454e-05;
% Parameter Pengendali PD
Kp_PD = 20;
Kd_PD = 30;
% Parameter Pengendali PID
Kp_PID = 11.3678;
Ki_PID = 24.7629;
Kd_PID = 1.6365;

% Fungsi alih pengendali PID
cont_P = Kp_P;
cont_PI = Kp_PI + Ki_PI/s;
cont_PD = Kp_PD + Kd_PD*s;
cont_PID = Kp_PID + Ki_PID/s + Kd_PID*s;

% Fungsi alih lingkar tertutup
open_loop_P = (cont_P * sys_ol);
closed_loop_P = feedback(sys_ol * cont_P, 1);
open_loop_PI = (cont_PI * sys_ol);
closed_loop_PI = feedback(sys_ol * cont_PI, 1);
open_loop_PD = (cont_PD * sys_ol);
closed_loop_PD = feedback(sys_ol * cont_PD, 1);
open_loop_PID = (cont_PID * sys_ol);
closed_loop_PID = feedback(sys_ol * cont_PID, 1);

% Analisis Kesalahan Sistem Kendali Motor DC
fprintf('\n')
disp('Analisis Kesalahan Kinerja Sistem Kendali Motor DC');
[num_ol_P, den_ol_P]= tfdata (open_loop_P, 'v');
[num_cl_P, den_cl_P]= tfdata (closed_loop_P, 'v');
[num_ol_PI, den_ol_PI]= tfdata (open_loop_PI, 'v');
```

```
[num_cl_PI, den_cl_PI]= tfdata (closed_loop_PI, 'v');
[num_ol_PD, den_ol_PD]= tfdata (open_loop_PD, 'v');
[num_cl_PD, den_cl_PD]= tfdata (closed_loop_PD, 'v');
[num_ol_PID, den_ol_PID]= tfdata (open_loop_PID, 'v');
[num_cl_PID, den_cl_PID]= tfdata (closed_loop_PID, 'v');
sys_e_P = tf(1, den_cl_P);
sys_e_PI = tf(1, den_cl_PI);
sys_e_PD = tf(1, den_cl_PD);
sys_e_PID = tf(1, den_cl_PID);
Errortf(num_ol_P, den_ol_P);
Errortf(num_ol_PI, den_ol_PI);
Errortf(num_ol_PD, den_ol_PD);
Errortf(num_ol_PID, den_ol_PID);
```

Hasil:

Analisis Kesalahan Kinerja Sistem Kendali Motor DC
Tipe Sistem adalah 0
Konstanta Kesalahan Posisi (Kp) adalah 58.8352
Konstanta Kesalahan Kecepatan (Kv) adalah 0.0000
Konstanta Kesalahan Percepatan (Ka) adalah 0.0000
Kesalahan Keadaan Mantap Untuk Masukan Undak adalah 0.0167
Kesalahan Keadaan Mantap Untuk Masukan Laju adalah Inf
Kesalahan Keadaan Mantap Untuk Masukan Parabolik adalah Inf

Tipe Sistem adalah 1
Konstanta Kesalahan Posisi (Kp) adalah Inf
Konstanta Kesalahan Kecepatan (Kv) adalah 0.0002
Konstanta Kesalahan Percepatan (Ka) adalah 0.0000
Kesalahan Keadaan Mantap Untuk Masukan Undak adalah 0.0000
Kesalahan Keadaan Mantap Untuk Masukan Laju adalah 6304.9316
Kesalahan Keadaan Mantap Untuk Masukan Parabolik adalah Inf

Tipe Sistem adalah 0
Konstanta Kesalahan Posisi (Kp) adalah 78.4130
Konstanta Kesalahan Kecepatan (Kv) adalah 0.0000
Konstanta Kesalahan Percepatan (Ka) adalah 0.0000
Kesalahan Keadaan Mantap Untuk Masukan Undak adalah 0.0126
Kesalahan Keadaan Mantap Untuk Masukan Laju adalah Inf
Kesalahan Keadaan Mantap Untuk Masukan Parabolik adalah Inf

Tipe Sistem adalah 1
Konstanta Kesalahan Posisi (Kp) adalah Inf
Konstanta Kesalahan Kecepatan (Kv) adalah 97.0867
Konstanta Kesalahan Percepatan (Ka) adalah 0.0000

Kesalahan Keadaan Mantap Untuk Masukan Undak adalah 0.0000
Kesalahan Keadaan Mantap Untuk Masukan Laju adalah 0.0103
Kesalahan Keadaan Mantap Untuk Masukan Parabolik adalah Inf