TUGAS AKHIR – TE 141599

DESAIN SISTEM PENGATURAN KECEPATAN MOTOR ARUS SEARAH TANPA SIKAT MENGGUNAKAN PID MULTIOBJEKTIF BERDASARKAN ALGORITMA GENETIKA

Mohammad Safrurriza
NRP 2211100083

Dosen Pembimbing
Ir. Rusdhianto Effendi A.K., MT.
Ir. Ali Fatoni, MT.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2016

FINAL PROJECT – TE 141599

GENETIC ALGORITHM BASED MULTIOBJECTIVE PID SPEED CONTROL DESIGN FOR BRUSHLESS DC MOTOR

Mohammad Safrurriza
NRP 2211100083

Supervisor
Ir. Rusdhianto Effendi A.K., MT.
Ir. Ali Fatoni, MT.

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Industrial Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2016

PENGATURAN KECEPATAN MOTOR ARUS SEARAH TANPA SIKAT MENGGUNAKAN KONTROLER PID MULTIOBJEKTIF BERDASARKAN ALGORITMA GENETIKA

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Teknik Sistem Pengaturan
Jurusan Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui :

Dosen Pembimbing I	Dosen Pembimbing II
Ir. Rusdhianto Effendi AK, MT.	**Ir. Ali Fatoni, MT.**
NIP. 1957 04 24 1985 02 1001	NIP. 1962 06 03 1989 03 1002

SURABAYA
JANAURI, 2016

PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini penulis menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir penulis dengan judul "**Pengaturan Kecepatan Motor Arus Searah Tanpa Sikat menggunakan Kontroler PID Multiobjektif Berdasarkan Algoritma Genetika**" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang penulis akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, penulis bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, Januari 2016

Mohammad Safrurriza
NRP 2211100083

DESAIN SISTEM PENGATURAN KECEPATAN MOTOR ARUS SEARAH TANPA SIKAT MENGGUNAKAN KONTROLER PID MULTIOBJEKTIF BERDASARKAN ALGORITMA GENETIKA

Mohammad Safrurriza – 2211100083

Pembimbing : 1. Ir. Rusdhianto Effendie A.K.,MT.
2. Ir. Ali Fatoni, MT.

ABSTRAK

Pada masyarakat modern, hampir semua sumber energi diubah menjadi energi listrik sebelum digunakan langsung. Konversi energi listrik menjadi energi mekanik dapat menggunakan motor listrik. Terdapat beberapa jenis motor listrik dengan berbagai karakteristik yang berbeda. Salah satu jenis motor listrik yang sedang populer saat ini adalah motor arus searah tanpa sikat (*brushless DC*). Memiliki efisiensi tinggi, tahan lama, ukurannya kecil, noise rendah, kerapatan daya besar, dan memiliki karakteristik kecepatan-torsi yang bagus membuat motor ini banyak dipakai . Sistem pengaturan kecepatan harus diberikan pada motor arus searah tanpa sikat agar dapat mempertahankan performanya. Kontroler PID digunakan untuk mengatur kecepatan motor arus searah tanpa sikat. Algoritma genetika digunakan untuk *tuning* parameter PID yang paling optimal. Berdasarkan hasil simulasi didapatkan parameter PID Kp=0,238, Ki=1,148, Kd=0,013

Kata kunci: Algoritma Genetika, Kontroler PID, Motor Arus Searah Tanpa Sikat (*Brushless* DC, BLDC).

ABSTRACT

In modern society, most energy source converted into electrical form before used. Motor is used to convert electrical energy into mechanical energy. There are many types of motor classification based of usage of the motor. Brushless Direct Current become popular because of its high efficiency, high power density, low noise, and good torque-speed characteristic. Speed control system is needed in order the motor can maintain performance. PID *controller is used to regulate brushless* DC *speed. Genetic Algorithm is used to tune* PID *parameters. Based on simulation, the resulted parameters are* Kp=0.238, Ki=1.148, Kd=0.013

Keywords : *Brushless DC, Genetic Algorithm , PID Controller.*

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Alhamdullilah, puji syukur penulis panjatkan kehadirat Allah SWT karena atas rahmat dan karunia-Nya penulis dapat menyelesaikan perancangan buku tugas akhir dengan judul "**Pengaturan Kecepatan Motor Arus Searah Tanpa Sikat Menggunakan Kontroler PID Multiobjektif Berdasarkan Algoritma Genetika**"

Terima kasih atas dukungan seluruh pihak.Penulis berharap tugas akhir ini dapat bermanfaat bagi penelitian selanjutnya.

Surabaya, 18 Januari 2016

Penulis

Halaman ini sengaja dikosongkan

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Pada masyarakat modern, listrik merupakan sumber energi yang paling populer digunakan [1]. Untuk konversi energi dari listrik ke mekanik digunakan motor listrik. Hampir semua peralatan yang kita gunakan sehari-hari menggunakan motor listrik. Mulai dari peralatan rumah tangga, *air conditioner*, lift, robot hingga alat trans*port*asi saat ini sudah menggunakan motor listrik . Salah satu jenis motor listrik yang paling banyak digunakan adalah motor arus searah tanpa sikat / *brushless direct current (BLDC) motor*. BLDC memiliki efisiensi tinggi, tahan lama, ukurannya kecil, *noise* rendah, dan memiliki karakteristik kecepatan-torsi yang bagus. Motor ini tidak menggunakan sikat sehingga memudahkan perawatan, mengurangi rugi gesekan, serta menghasilkan *electro magnetic interference* (EMI) yang rendah.

Proses pengaturan kecepatan pada BLDC lebih rumit daripada motor DC biasa. Seiring dengan perkembangan saklar elektronik dan elektronika daya, proses komutasi pada BLDC menjadi lebih mudah dan lebih efisien. Untuk mengatur kecepatan motor BLDC dibutuhkan kontroler yang andal agar motor dapat menjaga performansinya saat ada perubahan beban.

Pada tugas akhir ini, penulis menggunakan kontroler PID berdasarkan algoritma genetika. Metode kontrol PID digunakan untuk mengatur kecepatan motor BLDC, sehingga dapat membantu BLDC mempertahankan kecepatan saat terjadi pembebanan . Algoritma genetika digunakan untuk mencari parameter kontroler PID yang paling optimal agar motor dapat menjaga performansinya saat ada pembebanan.

1.2 Perumusan Masalah

Ketika motor BLDC tidak dibebani, motor akan berjalan dengan kecepatan konstan. Namun ketika motor BLDC diberikan suatu beban yang diatur sedemikian rupa, maka kecepatan motor akan berubah. Efek pembebanan ini akan menganggu performa motor yang sedang beroperasi sehingga motor tidak mampu mempertahankan kecepatan pada saat ada perubahan beban. Permasalahan utama dari Tugas Akhir ini ialah bagaimana merancang sistem pengaturan kecepatan motor BLDC pada kondisi berbeban sehingga motor dapat mempertahankan kecepatannya.

1.3 Batasan Masalah

Ruang lingkup pembahasan tugas akhir ini dibatasi sebagai berikut,

a. *Plant* yang digunakan adalah motor arus searah tanpa sikat/ BLDC.
b. Perancangan dan implementasi kontroler untuk mengatur kecepatan motor BLDC menggunakan metode *proportional integral differential* (PID).
c. *Tuning* parameter kontroler PID menggunakan algoritma genetika.
d. *Tuning* parameter Kp, Ki, Kd dilakukan secara *offline*.

1.4 Tujuan Penelitian

Penelitian Tugas Akhir ini bertujuan untuk ,

a. Merancang *plant* pengujian yang terdiri motor BLDC sebagai objek pengujian dan elemen pendukung seperti beban (rem magnetik) dan sensor untuk pembacaan kecepatan motor.
b. Merancang dan mengimplementasikan desain kontroler PID berdasarkan algoritma genetika untuk mengatur kecepatan motor BLDC
c. Menganalisis respon simulasi motor BLDC yang telah diatur kecepatannya

1.5 Sistematika Perancangan

Pembahasan Tugas Akhir ini akan dibagi menjadi lima bab sebagai berikut:

BAB 1 : PENDAHULUAN

Bab ini berisi latar belakang, perumusan masalah, tujuan penelitian, sistematika perancangan dan relevansi dari tugas akhir ini.

BAB 2 **:** TINJAUAN PUSTAKA

Berisi teori yang dijadikan acuan dalam penyusunan tugas akhir.

BAB 3 : PERANCANGAN SISTEM

Bab ini membahas proses perancangan perangkat keras dan *software* pada simulator BLDC.

BAB 4 : PENGUJIAN DAN ANALISIS

Bab ini menjelaskan hasil simulasi dan implementasi kontroler pada simulator BLDC. Data hasil pengujian akan dianalisis lebih lanjut.

BAB 5 : KESIMPULAN DAN SARAN

Bab ini memaparkan kesimpulan dari seluruh proses pengerjaan tugas akhir serta saran untuk pengembangan penelitian selanjutnya

1.6 Relevansi

Hasil dari tugas akhir ini diharapkan dapat memberikan manfaat dalam pengembangan penelitian tentang BLDC khususnya strategi kontrol motor listrik dalam BLDC.

Halaman ini sengaja dikosongkan

BAB 2
TINJAUAN PUSTAKA

2.1 Motor *Brushless* DC [1]

Terdapat dua definisi umum pada motor *brushless DC (*BLDC*).* Beberapa kelompok ilmuan menganggap hanya motor tanpa sikat (*brushless)* dengan bentuk gelombang ggl balik *trapezoid* dan kotak saja yang termasuk BLDC. Sedangkan motor dengan bentuk ggl sinusoidal digolongkan sebagai motor sinkron magnet permanen. Sedangkan kelompok lainnya menganggap seluruh motor *brushless* digolongkan sebagai BLDC. Pada standar ANSI/IEEE 100-1984 mendefinisikan BLDC sebagai "*Brushless Rotary Machinery".* Pada Standar NEMA MG7-1987, BLDC didefinisikan sebagai motor yang berputar secara sinkron dan diatur komutasinya secara elektrik, dimana rotornya merupakan magnet permanen yang dilengkapi sensor posisi rotor. Karena tidak ada kesamaan dalam pendefinisian dan klasifikasi motor BLDC, maka pada buku ini menggunakan definisi pertama diatas.

Motor BLDC merupakan pengembangan dari motor DC. Pada motor DC, proses komutasi dilakukan secara mekanik (pada komutator). Proses komutasi mekanik ini memiliki beberapa kelemahan antara lain, membatasi daya motor, terdapat rugi-rugi gesekan, biaya perawatan motor tinggi, menimbulkan *electromagnetic interference(emi)* yang mengganggu peralatan sekitarnya. Pada BLDC proses komutasi dilakukan secara elektrik. Terdapat sensor *hall effect* yang digunakan untuk mengetahui posisi rotor. Posisi rotor dibutuhkan saat proses komutasi untuk mengatur kecepatan dan arah putaran motor. Bentuk fisik motor BLDC dapat dilihat pada Gambar 2.1.

Gambar 2.1 Bentuk Fisik Motor BLDC **[1]**

2.1.1 Cara Kerja Motor *Brushless* DC [2]

Cara kerja pada motor BLDC cukup sederhana, yaitu magnet permanen yang berada pada rotor akan tertarik dan terdorong oleh gaya elektromagnetik pada belitan stator yang diatur oleh *driver*. Hal ini membedakakn motor BLDC dengan motor DC yang menggunakan sikat mekanis yang berada pada komutator untuk mengatur waktu komutasi dan memberikan medan magnet pada lilitan.

2.1.2 Konstruksi Motor *Brushless* DC

Struktur motor BLDC terdiri atas bagian rotor , stator, sensor posisi rotor, rangkaian *driver*(*inverter*). Berikut ini penjelasan secara detail dari masing- masing komponen diatas,

2.1.2.1 Rotor

Rotor motor BLDC merupakan bagian yang bergerak pada motor. Rotor pada BLDC berupa magnet permanen. Proses magnetisasi menentukan arah fluks magnet yang terbentuk. Bentuk magnet permanen juga mempengaruhi bentuk sinyal dari emf(*electro motive force*) balik. Yaitu gaya gerak listrik yang dihasilkan kumparan stator yang melawan sumber listrik.

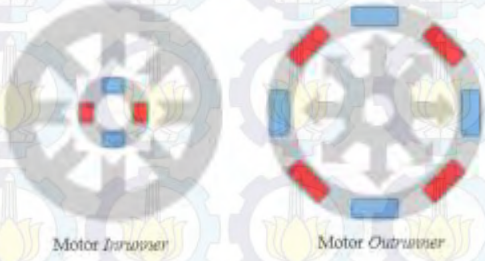
Cara magnetisasi ada tiga macam, yaitu radial, paralel, dan halbach *array*. Konfigurasi halbach-*array* dapat memfokuskan medan magnet ke dalam atau ke luar rotor. Gambar 2.2 dan Gambar 2.3 merupakan cara magnetisasi.

Gambar 2.2 Proses Magnetisasi a.Radial b.Paralel **[2]**

Gambar 2.3 Magnetisasi Halbach-*Array* **[2]**

Berdasarkan bentuk rotor, BLDC dapat digolongkan menjadi dua yaitu *outrunnrer* dan *inrunner*. Motor *Outrunner* merupakan jenis motor BLDC yang memiliki stator terletak di dalam motor dan rotor terletak di

luar sehingga bagian dari motor yang berputar merupakan bagian luar dari motor. Sedangkan *inrunner* merupakan jenis motor yang memiliki stator di bagian luar motor dan rotor berada pada bagian dalam motor sehingga bagian motor yang berputar adalah bagian dalam motor. Gambar 2.4 merupakan konstruksi motor BLDC *outrunner* dan *inrunner*.



Gambar 2.4 Konstruksi Motor BLDC *Outrunner* dan *Inrunner*.

2.1.2.2 Stator

Stator BLDC terdapat kumparan tiga fasa dengan susunan beda fasa 120^0 antar fasa. Sinyal listrik yang diberikan pada kumparan stator dapat berupa sinyal *sinusoid*, *trapezoid*, dan kotak.

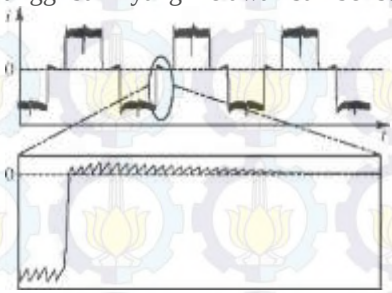
Stator merupakan bagian motor yang memiliki lilitan, sehingga apabila pada lilitan rotor diberi arus akan timbul gaya elektromagnetik. Dan rotor merupakan bagian motor yang diberi magnet permanen. Oleh karena itu, apabila pada rotor diberi arus maka pada pada stator dan rotor akan timbul gaya tarik-menarik dan tolak-menolak yang akan menyebabkan motor akan berputar.

2.1.2.3 Sensor Posisi Rotor

Karena tidak adanya komutator pada motor BLDC, untuk menentukan *timing* komutasi yang tepat, diperlukan 3 buah sensor *hall* atau *encoder*. Pada sensor *hall*, *timing* komutasi ditentukan dengan cara mendeteksi medan magnet rotor dengan menggunakan 3 buah sensor *hall* untuk mendapatkan 6 kombinasi *timing* yang berbeda, sedangkan pada *encoder*, *timing* komutasi ditentukan dengan cara menghitung jumlah pola yang ada pada *encoder*. Pada umumnya *encoder* lebih banyak digunakan pada motor BLDC komersial karena *encoder* cenderung mampu menentukan *timing* komutasi lebih presisi dibandingkan dengan menggunakan sensor *hall*. Hal ini terjadi karena pada *encoder*, kode komutasi telah ditetapkan secara *fixed* berdasarkan banyak kutub dari

motor dan kode inilah yang digunakan untuk menentukan *timing* komutasi. Namun karena kode komutasi *encoder* untuk suatu motor tidak dapat digunakan untuk motor dengan jumlah kutub yang berbeda. Hal ini berbeda dengan sensor *hall*. Apabila terjadi perubahan *pole* rotor pada motor, posisi sensor *hall* dapat diubah dengan mudah. Hanya saja kelemahan dari sensor *hall* adalah apabila posisi sensor *hall* tidak tepat akan terjadi keselahan dalam penentuan *timing* komutasi atau bahkan tidak didapatkan 6 kombinasi *timing* komutasi yang berbeda.

Selain menggunakan sensor *hall* dan *encoder*, posisi rotor dapat didapatkan melalui estimasi. Posisi rotor dapat diestimasi menggunakan fasa pada arus motor. Diasumsikan fasa arus motor sama dengan fasa fluks magnet pada stator. Selain menggunakan arus motor, posisi rotor dapat diestimasi menggunakan ggl balik di tiap fasa. Saat satu fasa dieksitasi akan muncul medan magnet pada kumparan. Saat motor bergerak, akan muncul ggl balik yang melawan sumber suplai motor.



Gambar 2.5 Bentuk GGL Balik pada BLDC **[2]**

2.1.2.4 Driver Motor BLDC

Rangkaian *driver* / *inverter* adalah rangkaian yang mengubah sumber arus searah(DC) menjadi sumber arus bolak-balik (AC) tiga fasa. Sumber AC yang biasanya digunakan berbentuk *sinusoid*, *trapezoid*, atau kotak. Rangkaian *driver* motor BLDC *fullbridge* dapat dilihat pada Gambar 2.6.

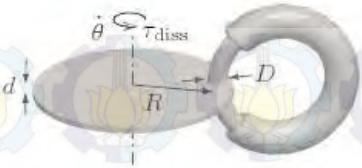
Gambar 2.6 Rangkaian *Driver* Motor BLDC *Fullbridge* **[2]**

2.2 Rem Elektromagnetik

Sistem pengereman ini menggunakan gaya elektromagnetik yang timbul dari suatu magnet permanen tetap atau kumparan yang diberikan arus listrik untuk memperlambat suatu gerakan. Konstruksi dasarnya berupa suatu piringan logam *non*-feromagnetik yang terpasang pada suatu poros yang berputar. Piringan logam tersebut diapit oleh kumparan yang dialiri arus listrik hingga menimbulkan medan magnet yang kutubnya saling berlawanan. Logam piringan tersebut akan memotong medan magnet yang ditimbulkan oleh kumparan tersebut sehingga menimbulkan *eddy current* atau arus *eddy*.

Arus *eddy* merupakan arus listrik yang timbul ketika suatu piringan logam berada di sekitar medan magnet yang garis-garis gayanya sedang berubah-ubah. Arus *eddy* ini mempunyai medan magnet yang arahnya berlawanan dengan arah gerak piringan logam. Akibatnya laju piringan logam akan tertahan akibat dari adanya arus *eddy* ini.

Rem elektromagnetik biasa diletakkan dekat dengan bagian yang bergerak. Rem ini bekerja pada kondisi yang dingin dan memenuhi persyaratan energi pengereman kecepatan tinggi karena tanpa adanya gesekan. Selain pada kendaraan listrik, aplikasi rem elektromagnetik juga dapat ditemukan pada sistem pengereman kereta api. Gambar 2.7 akan memperlihatkan struktur dan konstruksi dari sebuah sistem rem elektromagnetik.



Gambar 2.7 Struktur dari Sebuah Rem Elektromagnetik **[2]**

Nilai torsi pengereman berbanding lurus dengan fluks magnet, diameter cakram, ketebalan cakram, diameter inti magnet, konduktivitas cakram. Pengaturan torsi pengereman dapat dilakukan dengan mengatur tegangan keluaran *driver* rem yang berupa PWM. Hubungan torsi pengereman dinyatakan dalam fungsi $\dot{\theta}$ dan B.

$$\tau\left(B,\dot{\theta}\right) = n\frac{\pi\sigma}{4}D^2dB^2R^2\dot{\theta} \qquad (2.1)$$

τ : Torsi pengereman (Nm)
n : Jumlah magnet
σ : Konduktansi cakram ($\Omega^{-1}m^{-1}$)
D : Diameter inti magnet (m)
d : Ketebalan cakram (m)
B : Kuat medan magnet (T)
R : Radius efektif (m)
$\dot{\theta}$: Kecepatan sudut cakram (rad/s)

2.3 Arduino Uno [3]

Arduino Uno adalah *board* mikrokontroler berbasis Atmega328. Alat ini mempunyai 14 pin *input*/*output* digital dan 6 diantaranya dapat digunakan sebagai *output* PWM, 6 *input* analog, resonator keramik 19MHz, koneksi USB, soket listrik, ICSP *header*, dan tombol *reset*. Dengan kabel USB alat ini mudah dihubungkan ke komputer dan dapat diaktifkan dengan baterai atau adaptor AC ke DC. Spesifikasi Arduino Uno adalah sebagai berikut,

a. Mikrokontroler : Atmega328
b. Tegangan operasi : 5V

c. Tegangan *input* (direkomendasikan) : 7-12V
d. Tegangan *input* (batasan) : 6-20V
e. Pin *input*/*output* digital : 14 (6 diantaranya *output* PWM)
f. Pin *input* analog : 6
g. Arus DC tiap pin I/O : 40mA
h. Arus DC untuk pin 3,3 V : 50mA

14 pin *input* dan *output* Arduino Uno dapat digunakan dengan fungsi *pinMode(), digitalWrite(),* dan *digitalRead*(). Tiap pin memiliki tegangan operasi 5V dan dapat menerima arus maksimum 40mA. Beberapa pin memiliki fungsi khusus.

a. *Serial* : 0 (RX) dan 1 (TX). Digunakan untuk menerima (RX) dan mengirim (TX) data *serial* TTL. Pin ini terhubung dengan pin pada Atmega8U2 USB ke *chip* USB ke TTL *serial*
b. *External Interrupts* : 2 dan 3.pin ini digunakan sebagai *trigger* gangguan pada nilai yang rendah, menaikkan dan menurunkan nilai, atau merubah nilai.
c. PWM : 3,5,6,9,10, dan 11. Menyediakan *output* PWM 8bit dengan fungsi *analogWrite().*
d. SPI (*Serial Peripheral Interface*) : 10 (*Slave Select*), 11 (*Master Out Slave In*), 12 (*Master In Slave Out*), 13 (*Serial Clock*). Pin ini mendukung komunikasi SPI.
e. LED : 13, LED ini terhubung dengan pin 13. Ketika pin memiliki nilai yang tinggi LED akan *on* dan ketika pin memiliki nilai yang rendah, LED akan *off*

Pada Arduino Uno terdapat 6 *input* analog, dengan nama A0 hingga A5. Pin tersebut memiliki resolusi 10-bit (1024 nilai yang berbeda). *Input* analog ini berupa tegangan dari *ground* ke 5V.

2.4 *Software* LabVIEW 2013 [4]

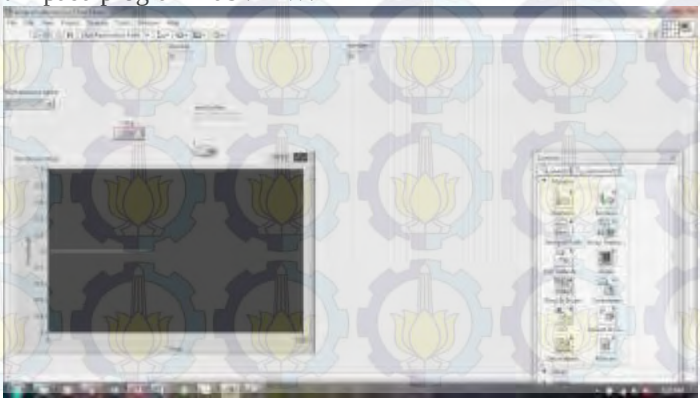
LabVIEW 2013 merupakan salah satu produk dari National Instrument. LabVIEW merupakan singkatan dari *Laboratory Virtual Instrumen Engineering Workbench*. LabVIEW merupakan sebuah *graphical programming environment* di mana program pada LabVIEW dibuat berbasiskan gambar [4]. Gambar 2.8 adalah *screenshot* tampilan awal ketika membuka *software* LabVIEW.

Gambar 2.8 Tampilan Awal LabVIEW 2013

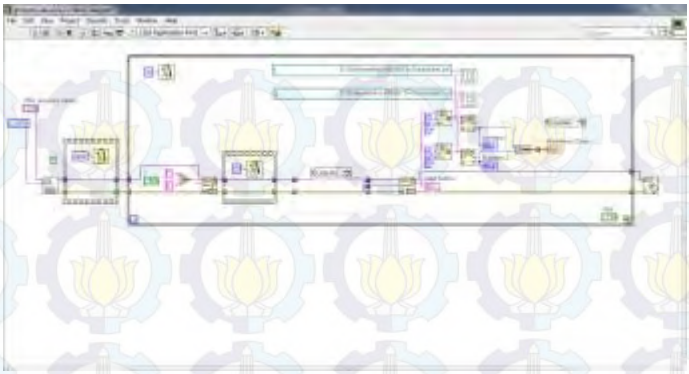
Urutan eksekusinya berdasarkan *data flow* dari blok-blok intruksinya. Dengan pemrograman berbasis gambar, dengan LabVIEW secara relatif membutuhkan waktu yang lebih singkat dalam membangun sebuah program dibandingkan dengan *software* pemrograman lain yang berbasis teks.

2.4.1 Bagian-bagian Dalam Program LabVIEW

Program pada LabVIEW disebut VI. VI artinya *Virtual Instrumen*. LabVIEW terdiri dari 2 bagian, yaitu *front panel* dan *block diagram*. *Front panel* berfungsi sebagai UI (*User Interface*). Sedangkan *block diagram* merupakan tempat di mana program LabVIEW dibuat. Gambar 2.9 dan Gambar 2.10 secara berurutan adalah contoh *front panel* dan *block diagram* pada program LabVIEW.

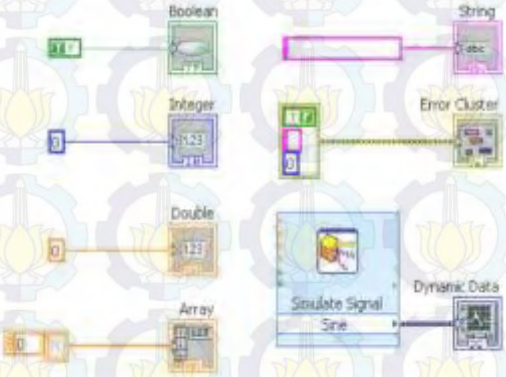


Gambar 2.9 *Front Panel* LabVIEW



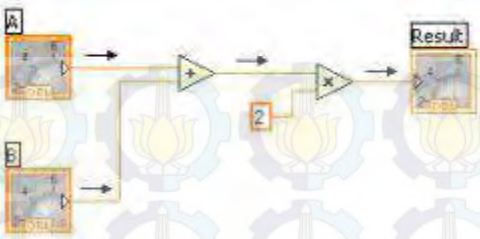
Gambar 2.10 *Block Diagram* LabVIEW

Layaknya *programming environment* lainnya, LabVIEW pun juga memiliki berbagai macam tipa data yang dapat digunakan oleh *user* dalam membangun sebuah program. Gambar 2.11 menunjukkan beberapa macam tipe data yang ada pada LabVIEW.



Gambar 2.11 Tipe Data pada LabVIEW

Boolean berisi status *True/False*. *Integer* merupakan bilangan cacah. Sedangkan *Double* adalah bilangan bulat. *Array* merupakan istilah lain dari *list*. *String* adalah *array* dari karakter. Sedangkan *cluster* adalah *array* yang anggotanya terdiri dari lebih dari satu macam tipe data.



Gambar 2.12 Eksekusi Data LabVIEW

Proses eksekusi data pada LabVIEW dapat dilihat pada Gambar 2.12. Pertama kali, LabVIEW akan mengeksekusi operasi penjumlahan dari variabel A dan B. Kedua, hasil penjumlahan A dan B akan dikalikan dengan konstanta. Ketiga, hasil dari perkalian akan dimasukkan pada variabel *Result*.

LabVIEW juga memiliki kemampuan untuk membuat sub program pada program utama. Tujuannya adalah untuk menyederhanakan tampilan program utama agar *readability* juga dapat ditingkatkan.

2.4.2 Koneksi LabVIEW dengan *Hardware*

Ada berbagai macam cara mengoneksikan LabVIEW dengan *hardware*. Pada Tugas Akhir ini LabVIEW akan dikoneksikan dengan perangkat DAQ menggunakan media USB. Untuk keperluan tersebut, LabVIEW telah menyediakan blok instruksi untuk komunikasi *serial* via USB. Letak blok instruksi ini terletak pada menu *Instrument I/O*. Melalui blok *instrument I/O* ini nantinya dapat menerima dan mengirim paket data pada perangkat DAQ. Gambar merupakan screenshot dari blok instruksi *Insrument I/O*.

Gambar 2.13 LabVIEW *Instrument IO*

2.5 *Software* MATLAB 2013 [5]

MATLAB merupakan paket program dengan bahasa pemrograman yang tinggi untuk mengembangkan algoritma, visualisasi data, dan komputasi numerik. Program MATLAB ini dapat digunakan untuk menyelesaikan masalah komputasi dengan lebih cepat dibandingkan dengan bahasa pemrograman tradisional, seperti C, C++, dan Fortran. MATLAB digunakan untuk banyak aplikasi seperti *signal and image processing*, desain kontrol, pengujian dan pengukuran, permodelan, dan analisis.

Simulink merupakan bagian dari MATLAB untuk memodelkan, mensimulasikan, dan menganalisa sistem dinamik. Simulink dapat membentuk model dari awal atau memodifikasi model yang sudah ada sesuai dengan apa yang diinginkan. Selain itu simulink juga mendukung sistem *linier* dan *non-linier*, permodelan waktu kontinyu atau diskrit, atau gabungan. Simulink ini dapat digunakan sebagai media untuk menyelesaikan masalah dalam industri nyata meliputi kedirgantaraan dan pertahanan, otomotif, komunikasi, elektronik dan pemrosesan sinyal,

Salah satu modul dalam Simulink yang dapat digunakan untuk komunikasi perangkat keras adalah *Instrument Control Toolbox*. Modul ini merupakan kunpulan fungsi *m-file* yang dibangun pada lingkungan komputasi teknis MATLAB. *Toolbox* ini menyediakan kerangka kerja untuk komunikasi instrumen yang mendukung GPIB *interface*, standar VISA, TCP/IP, dan protokol UDP. *Toolbox* ini memperluas fitur dasar *serial port* yang ada dalam MATLAB. Selain itu *toolbox* ini berfugsi untuk komunikasi data antara *workspace* MATLAB dan peralatan lainnya. Data tersebut dapat berbentuk biner atau *text*.

Komunikasi *serial* merupakan protokol dasar tingkat rendah untuk komunikasi antara dua peralatan atau lebih. Pada umumnya satu komputer dengan modem, *printer*, mikrokontroler, atau peralatan lainnya. *Serial port* mengirim dan menerima informasi *bytes* dengan hubungan seri. *bytes* tersebut dikirimkan menggunakan format biner atau karakter ASCII (*American Standard Code for Information Interchange*). Dalam komunikasi *serial* MATLAB, agar data ASCII dapat diproses *real time*, maka digunakan ASCII *encode* dan *decode* yang terdapat pada *xPC Target Library for RS232.* ASCII *encode* merupakan blok dalam simulink yang digunakan untuk mengubah data *bytes* menjadi karakter ASCII. Sedangkan ASCII *decode* merupakan blok Simulink yang digunakan untuk mengubah karakter ASCII menjadi data *bytes* yang kemudian dapat dikonversi sesuai kebutuhan.

2.6 Identifikasi Sistem

Identifikasi sistem dilakukan untuk mengetahui dinamika sistem dengan cara menyatakan dinamika sistem tersebut kedalam persamaan matematik. Persamaan matematik ini merupakan pendekatan yang representatif terhadap dari fenomena fisika dari sebuah sistem dinamik. Seluruh sistem dinamika di alam ini dapat dimodelkan dengan persamaan diferensial. Persamaan diferensial tersebut dapat diperoleh dengan menganalisa hukum-hukum fisis yang berlaku. Misalnya Hukum Newton untuk sistem mekanik, Hukum Kirch*off* untuk sistem elektrik.

Model matematik memiliki bentuk bermacam-macam tergantung sistem yang bersangkutan. Misalnya pada permasalahan analisa transien dari sebuah sistem LTI *single input single output (SISO)* akan lebih mudah jika direpresentasikan dengan *transfer function (TF,*fungsi alih*)* . fungsi alih, merupakan sebuah persamaan matematika yang menggambarkan perbandingan dari fungsi *output* dalam *laplace* dan fungsi *input* dalam bentuk *laplace* .

Gambar 2.14 Diagram Blok TF *Plant*

$$g(\mathrm{t}) = \frac{y(\mathrm{t})}{x(\mathrm{t})} \tag{2.2}$$

$$TF = G(\mathrm{s}) = \frac{\mathrm{Y(s)}}{\mathrm{X(s)}} \tag{2.3}$$

Dimana $g(\mathrm{t})$ merupakan model matematika dari *plant*, TF dapat dicari dengan melakukan transformasi *laplace* pada kedua sisi Persamaan 2.2 menjadi Persamaan 2.3.TF hanya digunakan untuk menyatakan sistem dengan *single input single output(SISO)* .

Teknik identifikasi ada dua macam, yaitu identifikasi *offline* dan *online*. Proses identifikasi *offline* dilakukan dengan cara mematikan kontroler dan memberikan sinyal uji kepada *plant* secara *open loop*, kemudian dinamika sinyal uji dan respon *plant* direkam untuk dianalisis lebih lanjut. Sedangkan proses identifikasi *online* dilakukan secara *close loop*. Artinya identifikasi *online* menggunakan kontroler. Proses

identifikasi *offline* dan *online* dapat dilihat pada Gambar 2.15 dan Gambar 2.16

Gambar 2.15 Proses Identifikasi *Offline*

Gambar 2.16 Proses Identifikasi *Online*

Secara umum ada 2 cara untuk mendapatkan model pendekatan sistem, yaitu melalui pendekatan respon waktu dan pendekatan respon frekuensi.

Pada pendekatan respon frekuensi, Sinyal uji yang digunakan berupa sinyal sinusoidal dengan magnitude konstan dan frekuensi yang diubah-ubah. model pendekatan menggunakan struktur kontinyu

Pada respon waktu kontinyu sinyal uji yang digunakan berupa sinyal step. Keluarannya didekati dengan model berikut,

a. Orde satu

$$\frac{Y(s)}{X(s)} = \frac{K}{\tau s + 1} \tag{2.4}$$

b. Orde dua

$$\frac{Y(s)}{X(s)} = \frac{K}{\frac{1}{\omega_n^2} s^2 + \frac{2\zeta}{\omega_n} s + 1} \tag{2.5}$$

c. Orde tinggi

$$\frac{Y(s)}{X(s)} = \frac{e^{-\tau_1 s}}{(\tau_2 s + 1)^N} \tag{2.6}$$

Pada respon waktu diskrit sinyal uji yang digunakan berupa sinyal random untuk proses *offline* atau sinyal kontrol pada proses *online*. model pendekatan menggunakan struktur diskrit.

2.6.1 Validasi

Proses validasi dilakukan untuk menguji kesesuaian model hasil identifikasi dengan data respon *plant.* Untuk melakukan validasi model dapat digunakan beberapa tolak ukur. Salah satunya adalah *Relative Steady State Error* (RSSE). *RSSE* diperoleh dari menghitung *RMSE* pada Persamaan 2.7 dan membaginya dengan nilai *steady state* kemudian dikalikan 100%.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{k}(y_i - \tilde{y}_i)^2}{k}} \tag{2.7}$$

$$RSSE = \frac{RMSE}{y_{ss}} x100\% \tag{2.8}$$

2.7 Kontroler Proporsional Integral Derivatif (PID)

Kontroler PID merupakan penggabungan antara tiga macam kontroler, yaitu P (*Proportional*), I (*Integral*), dan D (*Differential*). Setiap jenis kontroler memiliki karakteristik yang berdeda. Penggabungan kontroler dimaksudkan agar setiap kekurangan dan kelebihan dari masing – masing kontroler P, I, D dapat saling menutupi. Bentuk standar dari kontroler PID dapat dituliskan seperti Persamaan (2.9), dimana u adalah sinyal kontrol, e adalah sinyal kesalahan ($e=y_{sp}-y$). Sinyal kesalahan adalah hasil penjumlahan dari tiga bagian, bagian P(nilainya proporsional terhadap sinyal kesalahan), bagian I (nilainya proporsional dan integral terhadap sinyal kesalahan), dan bagian D (nilainya proporsional dan derivatif terhadap sinyal kesalahan). Parameter dari kontroler adalah K_p, τ_i, dan τ_d.

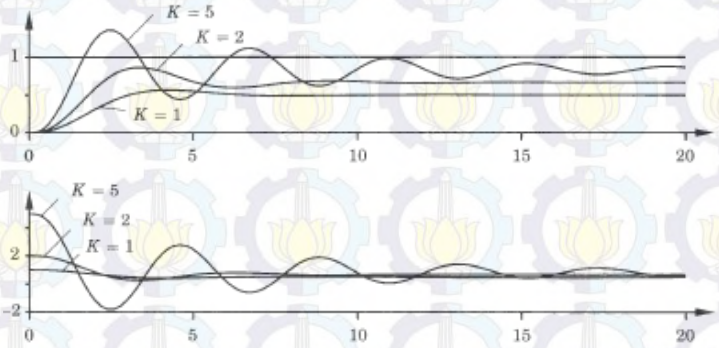
$$u(t) = K_p\left(e(\mathrm{t}) + \frac{1}{\tau_i}\int_0^t e(t)dt + \tau_d \frac{de(\mathrm{t})}{dt}\right) \tag{2.9}$$

Saat sinyal kontrol yang diharapkan hanya aksi proporsional, maka bentuk Persamaan(2.9) direduksi menjadi Persamaan (2.10).

$$u(t) = K_p e(t) + u_b \quad (2.10)$$

Sinyal kontrolnya proporsional terhadap sinyal kesalahan. Dimana u_b merupakan *bias* atau *reset*. Saat sinyal kesalahan bernilai nol, sinyal kontrol bernilai $u(t)=u_b$. *Bias* u_b biasanya diatur nilainya sebesar $u_b=(u_{max}-u_{min})/2$, terkadang nilai bias diatur manual agar sinyal kesalahan bernilai nol saat diberikan *set point* tertentu.

Pada Gambar 2.17 ,fungsi alih *plant* adalah $G(s)=(s+1)^{-3}$. Gambar atas menyatakan respon sistem dengan perubahan nilai K_p. Gambar bawah menggambarkan sinyal kontrolnya.

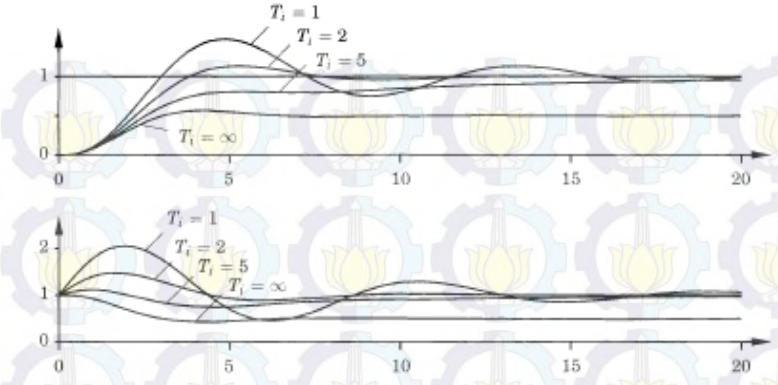


Gambar 2.17 Simulasi Sistem *Closed Loop* Dengan Proporsional Kontrol. **[6]**

Fungsi dari aksi integral adalah agar keluaran sistem sesuai dengan *set point*. Dengan aksi proporsional, akan muncul sinyal kesalahan pada saat *steady state*. Penggunaan aksi integral akan mengurangi sinyal kesalahan saat kondisi *steady state.* Struktur kontroler dengan aksi proporsional dan integral dinyatakan dalam Persamaan (2.11).

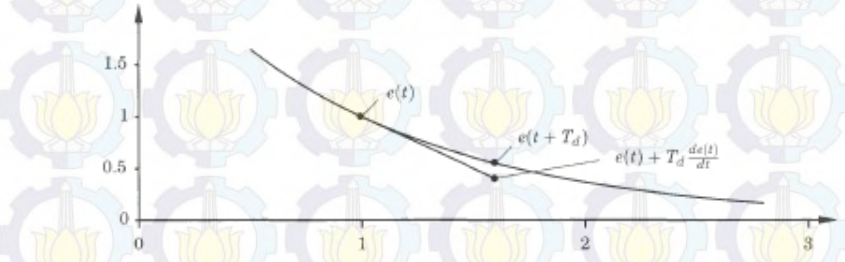
$$u(t) = K_p \left(e(t) + \frac{1}{\tau_i} \int_0^t e(t)dt \right) \quad (2.11)$$

Pada Gambar 2.18, fungsi alih *plant* adalah $G(s) = (s+1)^{-3}$ dengan penguatan proporsional $K_p = 1$ .Gambar atas menyatakan respon sistem dengan perubahan nilai τ_i . Gambar bawah menggambarkan sinyal kontrolnya.



Gambar 2.18 Simulasi Sistem *Closed Loop* Dengan Proporsional Integral Kontrol. **[6]**

Aksi derivatif diperlukan untuk meningkatkan stabilitas *closed loop*. Saat ada perubahan variabel kontrol, dibutuhkan waktu untuk mengoreksi kesalahan sehingga ada keterlambatan pada sinyal kontrol. Aksi kontroler dengan proporsional dan derivatif bernilai proporsional terhadap *output* yang diprediksi. Prediksi dilakukan dengan ekstrapolasi sinyal kesalahan seperti pada Gambar 2.19.



Gambar 2.19 Penggambaran Aksi Derivatif Sebagai Kontrol Prediktif **[6]**

Struktur kontroler PID pada Persamaan 2.7 jika dinyatakan dalam bentuk fungsi alih akan menjadi persamaan berikut,

$$C(s) = K_p\left(1 + \frac{1}{s\tau_i} + s\tau_d\right) \quad (2.12)$$

$$C'(s) = K_p'\left(1 + \frac{1}{s\tau_i}\right)(1 + s\tau_d) \quad (2.13)$$

$$C''(s) = K_p'' + \frac{K_i}{s} + sK_d \quad (2.14)$$

$$K_p'' = K_p''; K_i = \frac{K_p}{\tau_i}; K_d'' = K_p\tau_d$$

Bentuk kontroler PID pada Persamaan 2.12 disebut bentuk standar atau *non-interacting*. Hal ini karena waktu integral τ_i tidak mempengaruhi bagian kontroler derivatif dan waktu derivatif τ_d tidak mempengaruhi bagian kontroler integral. Bentuk kontroler PID pada Persamaan 2.13 disebut bentuk seri atau *interacting* karena kontroler integral dan derivatif saling mempengaruhi. Bentuk ini lebih umum digunakan pada kontroler PID komersial karena dianggap lebih mudah untuk *tuning* secara manual. Struktur kontroler PID pada Persamaan 2.14 disebut bentuk paralel. Struktur ini lebih banyak digunakan untuk perhitungan analitis. Pada bentuk paralel kita bisa mendapatkan sinyal kontrol yang murni proporsional, integral, dan derivatif dengan mengatur parameternya.

Tabel 2.1 Pengaruh K_p, K_i, K_d pada respon sistem

Respon *Closed-Loop*	*Rise Time*	*Overshoot*	*Settling Time*	*SS Errror*
K_p	Turun	Naik	Perubahan Kecil	Turun
K_i	Turun	Naik	Naik	Hilang
K_d	Perubahan Kecil	Turun	Turun	Perubahan Kecil

2.8 Algoritma Genetika [7]

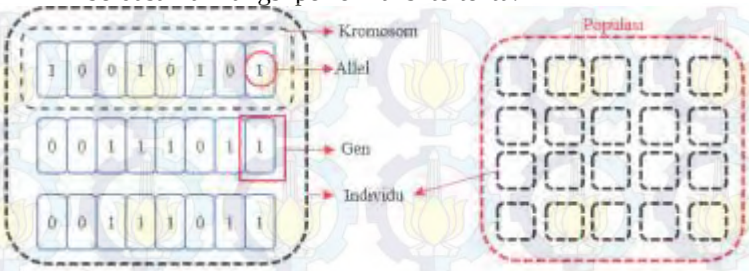
Pada algoritma ini, teknik pencarian dilakukan sekaligus atas sejumlah solusi yang mungkin yang dikenal dengan istilah populasi. Individu yang terdapat dalam satu populasi disebut dengan istiah kromosom. Kromosom ini merupakan suatu solusi yang masih berbentuk simbol. Populasi awal dibangun secara acak, sedangkan populasi berikutnya merupakan hasil evolusi kromosom - kromosom melalui iterasi yang disebut dengan istilah generasi. Pada setiap generasi kromosom akan melalui proses evaluasi dengan menggunakan alat ukur yang disebut fungsi *fitness*. Nilai *fitness* dari suatu kromosom akan menunjukkan kualitas kromosom dalam populasi tersebut. Generasi berikutnya dikenal dengan istilah anak (*offspring*) terbentuk dari gabungan dua kromosom generasi sekarang yang bertindak sebagai sebagai induk (*parent*) dengan menggunakan operator penyilangan (*crossover*). Selain operator penyilangan, suatu kromosom dapat juga dimodifikasi dengan menggunakan operator mutasi. Populasi generasi yang baru dibentuk dengan cara menyeleksi nilai *fitness* dari kromosom induk (*parent*) dan nilai *fitness* dari kromosom anak (*offspring*), serta menolak kromosom-kromosom yang lainnya sehingga ukuran populasi (jumlah kromosom dalam suatu populasi) konstan. Setelah melakukan beberapa generasi, maka algoritma ini akan konvergen ke kromosom terbaik.

2.8.1 Komponen Komponen Utama Algoritma Genetika

Ada enam komponen utam dalam algoritma genetika, yaitu teknik penyandian, Prosedur Inisialisasi, fungsi Evaluasi, Seleksi, Operator Genetika dan Penentuan Parameter. Terdapat juga beberapa definisi penting dalam algoritma genetika, antara lain,

a. *Genotype* (Gen), sebuah nilai yang menyatakan satuan dasar yang membentuk suatu arti tertentu dalam satu kesatuan gen yang dinamakan kromosom
b. Allel, nilai dari gen
c. Kromosom, gabungan gen-gen yang membentuk nilai tertentu.
d. Individu, menyatakan suatu nilai yang menyatakan salah satu solusi yang mungkin untuk suatu permasalahan optimasi.
e. Kumpulan individu yang akan diproses dalam satu siklus algoritma genetika.
f. Generasi, menyatakan jumlah siklus algoritma genetika berjalan.

g. *Fitness*, menyatakan baik-tidaknya suatu individu(solusi) berdasarkan fungsi performansi tertentu.



Gambar 2.20 Istilah dalam Algoritma Genetika.

2.8.2 Teknik Penyandian

Teknik penyandian di sini meliputi penyandian gen dan kromosom. Gen merupakan bagian dari kromosom. Satu gen biasanya akan mewakili satu variabel. Gen dapat direpresentasikan dalam bentuk *string bit*, pohon, *array* bilangan *real*, daftar aturan, elemen permutasi, elemen program, atau representasi lainnya yang dapat diimplementasikan untuk operator genetika

2.8.3 Prosedur Inisialisasi

Ukuran populasi tergantung pada masalah yang akan dipecahkan dan jenis operator genetika yang akan diimplementasikan. Setelah ukuran populasi ditentukan, kemudian harus dilakukan inisialisi terhadap kromosom yang terdapat pada populasi tersebut. Inisialisasi kromosom dilakukan secara acak, namun demikian harus tetap memperhatikan domain solusi dan kendala permasalahan yang ada.

2.8.4 Fungsi Evaluasi

Ada dua hal yang harus dilakukan dalam melakukan evaluasi kromosom, yaitu evaluasi fungsi objektif (fungsi tujuan) dan konversi fungsi objektif kedalam fungsi *fitness*. Secara umum, fungsi *fitness* diturunkan dari fungsi objektif yang tidak negatif.

2.8.5 Seleksi

Seleksi ini bertujuan untuk memberikan kesempatan reproduksi yang lebih besar bagi individu yang *fitness* nya lebih besar. Seleksi akan menentukan individu-individu mana saja yang akan dipilih untuk dilakukan rekombinasi dan bagaimana *offspring* terentuk dari individu-

individu terpilih tersebut. Ada beberapa metode seleksi dari induk, antara lain *Rank-based fitness assigment, Roulette wheel selection, Stochastic universal sampling, Local selection, Truncation selection, Tournament selection* .

2.8.6 Operator Genetika

Secara umum, operasi pada algoritma genetika ada dua, yaitu *crossover* dan mutasi. Jenis *crossover* antara lain,

a. *Crossover* satu titik
b. *Crossover* banyak titik
c. *Crossover uniform*
d. *Crossover* dengan permutasi

Setelah mengalami proses rekombinasi, pada *offspring* dapat dilakukan mutasi. Variabel *offspring* dimutasi dengan menambahkan nilai random yang sangat kecil (ukuran langkah mutasi), dengan probabilitas yang rendah. Mutasi berperan untuk menggantikan gen yang hilang dari populasi akibat proses seleksi yang memungkinkan munculnya kembali gen yang tidak muncul pada inisialisasi populasi. Selain itu mutasi juga berperan untuk memperlebar ruang pencarian. Mutasi dapat dibagi menjadi dua yaitu mutasi bernilai real dan mutasi bernilai biner.

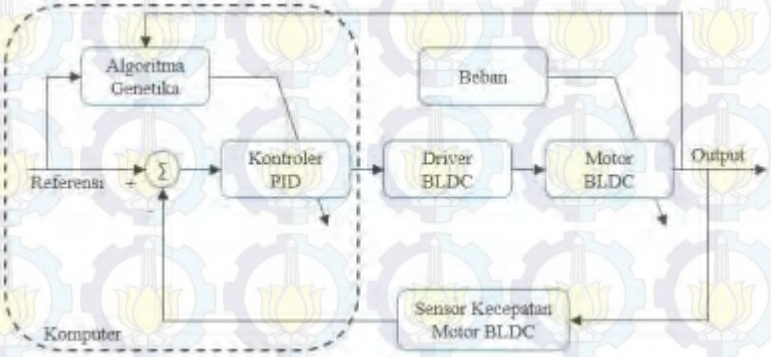
2.8.7 Kondisi *Stopping*

Proses *stopping* bisa menggunakan beberapa kriteria. Antara lain jumlah generasi, lama waktu program berjalan, tercapainya nilai *fitness* tertentu, dan saat *fitness* tidak ada perubahan.

BAB 3
PERANCANGAN SISTEM

1.1 Gambaran Umum Sistem

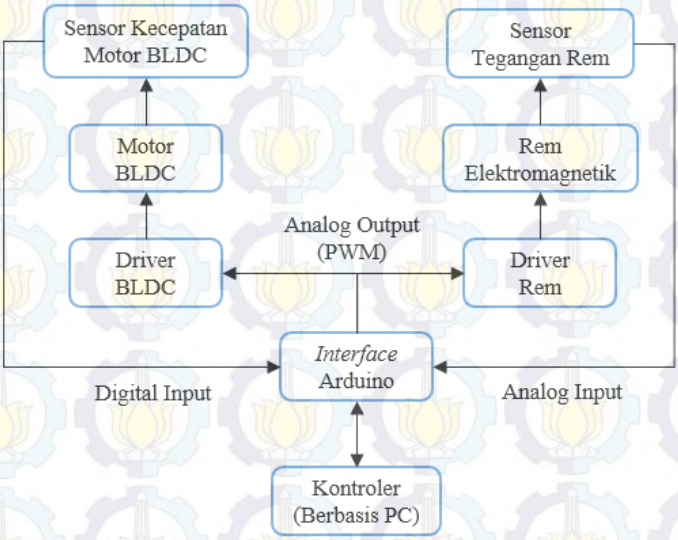
Dalam tugas akhir ini penulis merancang sistem pengaturan kecepatan motor BLDC. Sistem tersebut tersusun atas komponen berikut, motor BLDC sebagai perangkat utama yang akan diatur kecepatannya. Kontroler yang digunakan adalah PID multiobjektif berdasarkan algoritma genetika. Sebutan multiobjektif karena kontroler memiliki lebih dari satu parameter dalam fungsi objektif yang harus dicapai. Parameter yang dijadikan tolok ukur performa kontroler antara lain, *overshoot* dan *relative steady state error* (RSSE). Proses *tuning* parameter kontroler PID akan dilakukan menggunakan algoritma genetika. *Driver* motor BLDC sebagai aktuator yang menjembatani kontroler dengan *plant*. Sinyal kontrol dari kontroler yang berupa sinyal tegangan digunakan sebagai masukan *driver* agar dapat mengontrol kecepatan motor BLDC. Sensor kecepatan digunakan untuk mengukur kecepatan motor secara aktual. Sensor kecepatan pada *plant* BLDC menggunakan sistem pendeteksi ggl balik dari belitan motor atau lebih dikenal dengan sistem sensorless. Untuk memberikan efek pembebanan pada motor digunakan rem elektromagnetik. Dengan mengatur tegangan suplai rem, akan didapatkan torsi pengereman yang bervariasi. Blok diagram dari sistem pengaturan kecepatan BLDC dapat dilihat pada Gambar 1.1



Gambar 1.1 Blok Diagram Pengaturan Kecepatan Motor BLDC

1.2 Perancangan Perangkat Keras

Dalam proses perancangan simulator BLDC, perangkat keras dibagi menjadi 2 golongan, yaitu perangkat mekanik dan perangkat elektronik. Komponen yang termasuk perangkat mekanik antara lain, motor BLDC, kopel karet, rem elektromagnetik. Komponen yang termasuk perangkat elektronik antara lain, *driver* motor BLDC, *driver* rem elektromagnetik, sensor arus motor BLDC, sensor arus rem elektromagnetik, arduino *interface*, dan komputer. Secara umum, susunan perangkat dalam simulator BLDC digambarkan dalam Gambar 3.2.



Gambar 1.2 Konfigurasi Perangkat Keras Simulator BLDC

1.2.1 Perancangan Mekanik

Komponen simulator BLDC yang termasuk dalam perancangan mekanik antara lain,

1.2.1.1 Motor Brushless DC

Motor *Brushless Direct Current* (BLDC) yang penulis gunakan merupakan motor blower dari AC *inverter* Daikin. Spesifikasi motor BLDC dapat dilihat pada Tabel 1.1 .

Tabel 1.1 Spesifikasi Motor BLDC Daikin D43F.

Parameter		Nilai
Tipe		Daikin D43F
Berat Motor		1200 gram
Tegangan Kerja		311 VDC
Kecepatan Motor	Beban Minimal	2115,863 rpm
	Beban Nominal	1116,596 rpm
	Beban Maksimal	1047,136 rpm

1.2.1.2 Rem Elektromagnetik

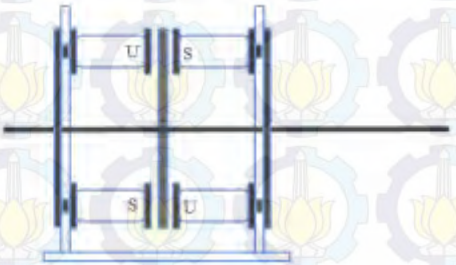
Rem elektromagnetik pada *plant* ini berguna sebagai pembebanan pada motor. Medan elektromagnetik dari rem ini dihasilkan oleh delapan kumparan yang dihubungkan secara seri dan diberikan masukan arus DC. Arus DC yang masuk ke dalam rem elektromagnetik berbentuk PWM yang *duty cycle*-nya diatur oleh *driver* dan Arduino. Adapun sumber tegangannya didapat dari jala-jala PLN yang diberikan trafo *step-down* dari 220V menjadi 30V. lalu *output* pada trafo akan disearahkan melalui rangkaian *rectifier*.

Rem elektromagnetik yang digunakan pada simulator BLDC ini berfungsi sebagai beban yang dipasang pada poros motor BLDC. Rem elektromagnetik yang penulis gunakan tersusun atas cakram alumunium yang diapit empat pasang belitan. Belitan tersebut dialiri arus listrik sehingga timbul medan magnet dan muncul fluks magnet yang memotong cakram alumunium. Ketika cakram berputar, akan muncul ggl induksi pada cakram alumunium. Karena alumunium merupakan konduktor, maka akan muncul arus listrik yang berputar pada cakram alumunium dengan arah berlawanan dengan perubahan fluks. Arus ini disebut arus eddy. Konstruksi dari rem elektromagnetik ditunjukkan pada Gambar 1.3 Spesifikasi rem elektromagnetik ditunjukkan pada Tabel 1.2.

Tabel 1.2 Spesifikasi Rem Elektromagnetik

Parameter	Nilai
Jumlah kumparan	8 (masing –masing 400 kumparan)

Parameter	Nilai
Resistansi kumparan(seri)	12,1 Ω
Diamater kumparan	3 cm
Diameter cakram	15 cm
Diamater kawat	0,6 mm
Tegangan kerja	0-30 V
Arus maksimal	2 A



Gambar 1.3 Konstruksi Rem Elektromagnetik

1.2.1.3 Kopel Fleksibel

Pada konstruksi simulator BLDC, *shaft* motor BLDC dipasang seporos dengan rem elektromagnetik. Oleh karena itu konstruksi penahan *shaft* pada rem elektromagnetik harus benar-benar lurus dengan *shaft* motor BLDC. Jika kedua *shaft* tidak lurus, akan menyebabkan kerusakan pada *bearing*. Bahkan *shaft* bisa bengkok jika kedua *shaft* diputar pada keadaan tidak lurus. Untuk menjaga kedua *shaft* tetap seporos sangat sulit. Hal ini dikarenakan saat motor berputar, akan timbul getaran pada motor sehingga menyebabkan kedua *shaft* tidak lurus. Untuk menghindari hal ini perlu dipasang kopel yang menghubungkan *shaft* motor BLDC dengan *shaft* rem elektromagnetik.

1.2.2 Perancangan Elektronik

Perancangan elektronik terdiri dari beberapa bagian, antara lain arduino sebagai *interface*, *driver* motor BLDC, rangkaian sensor arus motor BLDC, rangkaian sensor kecepatan motor BLDC, rangkaian *driver*

rem elektromagnetik, rangkaian sensor arus rem elektromagnetik. Pembahasan lebih detail akan dijelaskan sebagai berikut,

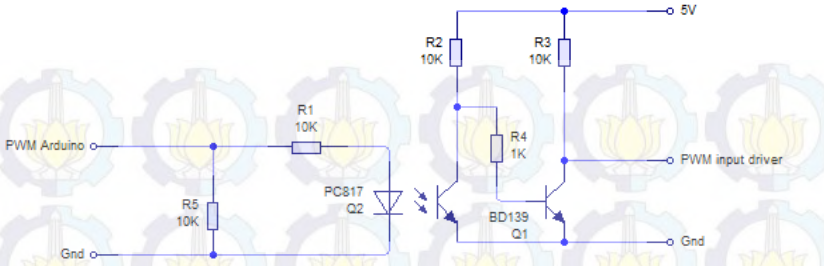
1.2.2.1 Interface Arduino Uno R3

Arduino adalah pengendali mikro *single-board* yang bersifat *open-source*, diturunkan dari *Wiring platform*, dirancang untuk memudahkan penggunaan elektronik dalam berbagai bidang. *Hardware*nya memiliki prosesor Atmel AVR. Pemrograman pada arduino dilakukan menggunakan IDE (menggunakan bahasa C). Pada simulator BLDC, Arduino digunakan sebagai *interface* (penghubung) antara kontroler berbasis komputer dengan *plant*. Arduino digunakan untuk mengakuisisi data kecepatan motor BLDC, arus motor BLDC, dan arus rem elektromagnetik. Tegangan yang masuk ke arduino akan diubah menjadi data digital dengan resolusi 10 bit dan akan diolah oleh program kontroler pada komputer. Kemudian kontroler terebut menghasilkan sinyal kontrol berbentuk PWM yang dikeluarkan melalui pin *output* digital PWM. Transfer data antara komputer dan arduino dilakukan melalui *port* komunikasi *serial*. Berikut ini pemilihan pin *input*/*output* pada arduino,

a. Pin A0 : *input* analog dari sensor tegangan *driver* rem elektromagnetik.
b. Pin 9 : *output* PWM untuk *driver* motor BLDC.
c. Pin 10 : *output* PWM untuk *driver* rem elektromagnetik.
d. Pin 11 : *input digital* sensor kecepatan.
e. Pin +5 V
f. Pin *Ground*

1.2.2.2 Driver Motor BLDC

Pada simulator BLDC, *driver* berfungsi mengubah sinyal kontrol dari kontroler yang tegangan analog (PWM) menjadi urutan komutasi kumparan motor BLDC. Gambar 3.4 merupakan rangkaian isolasi arduino dengan *driver* BLDC.



Gambar 1.4 Rangkaian Isolasi Arduino dengan *Driver* BLDC

Driver BLDC menggunakan ggl balik pada kumparan tiap fasa untuk mendeteksi posisi rotor. Saat motor dalam keadaan start dari keadaan berhenti, *driver* akan memberikan sinyal masukan pada setiap fasa dengan urutan seperti pada Tabel 1.3. Setelah motor berputar, akan muncul ggl balik yang dideteksi dengan rangkaian *zero crossing* pada saat *state* bernilai *off*. Urutan komutasi pada tiap fasa sebagai berikut,

Tabel 1.3 Komutasi *driver* BLDC

State	Fasa U	Fasa V	Fasa W
1	*High*	*Low*	*Off*
2	*High*	*Off*	*Low*
3	*Off*	*High*	*Low*
4	*Low*	*High*	*Off*
5	*Low*	*Off*	*High*
6	*Off*	*Low*	*High*

1.2.2.3 Rangkaian Driver Rem Elektromagnetik

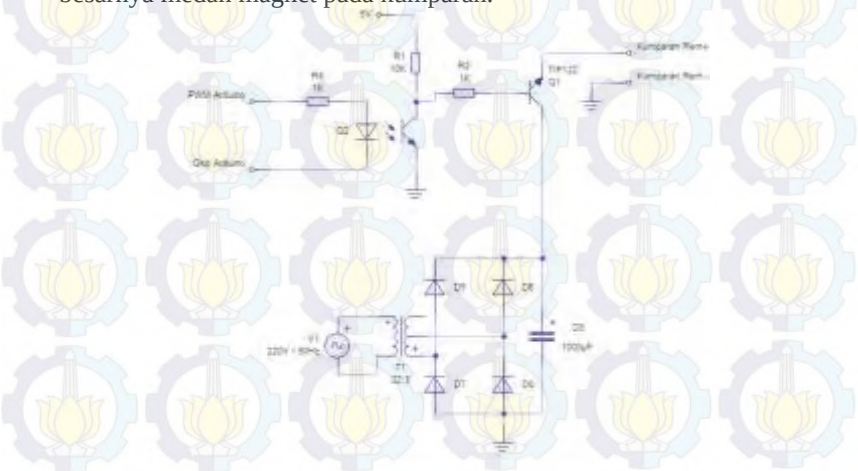
Rangkaian *driver* rem elektromagnetik terdiri atas dua bagian, yaitu penyearah gelombang penuh dan *Pulse Width Modulation* (PWM) generator. Penyearah gelombang penuh bekerja ketika dioda secara bergantian menyearahkan tegangan AC pada saat siklus positif dan negatif. Prinsip kerja dari penyearah ini adalah hanya memperbolehkan arus listrik mengalir dari satu arah saja dengan menggunakan komponen tertentu. Desain rangkaian penyearah gelombang penuh dapat dibuat dengan sistem penyearah *fullbridge* yang disusun dari 4 buah dioda yang dirangkai seperti pada Gambar 1.5 .

Ketika tegangan masukan bernilai negatif, maka dioda D6 dan D7 berada pada posisi *forward bias* sehingga tegangan pada keluaran akan

bernilai positif. Ketika tegangan masukan bernilai positif, maka dioda D5 dan D8 berada pada posisi *forward bias* sehingga tegangan pada keluaran bernilai positif. Tegangan hasil penyearah *fullbridge* masih banyak mengandung *ripple* sehingga perlu ditambahkan kapasitor C6 untuk mengurangi *ripple*. Nilai rasio kumparan primer: sekunder pada trafo 22:3. Ketika kumparan primer diberikan sumber 220 V akan menghasilkan keluaran 30 V di di sisi sekunder.

Rangkaian PWM *generator* digunakan untuk membangkitkan pulsa kotak dengan lebar pulsa yang bisa diatur. Dengan mengatur lebar pulsa, akan didapatkan variasi dari tegangan *output*.

Rem elektromagnetik yang digunakan pada *plant* ini dapat dikendalikan kekuatan medan magnetnya. Hal ini dapat dilakukan dengan mengubah besarnya arus yang akan masuk ke kumparan. Cara termudah untuk merubah arus pada kumparan adalah dengan mengatur tegangan masukan pada kumparan karena dengan berubahnya nilai tegangan masukan pada kumparan maka tentunya nilai arus masukan pada kumparan akan ikut berubah menyesuaikan dengan tegangan. Hal ini berjalan dengan sesuai dengan hukum Ohm yaitu $V = I(R + jX)$, Nilai resistansi dan reaktansi diasumsikan konstan, sehingga impedansi kumparan konstan. Nilai tegangan akan selalu sebanding dengan nilai arus. Hal inilah yang penulis manfaatkan untuk dapat merubah-ubah besarnya medan magnet pada kumparan.



Gambar 1.5 Rangkaian *Driver* Rem Elektromagnetik

1.2.2.4 Sensor Kecepatan BLDC

Rangkaian sensor kecepatan BLDC digunakan untuk melihat kecepatan motor BLDC secara *realtime*. Pada motor BLDC tidak terdapat sensor *hall* maupun *encoder* untuk mengukur kecepatan motor. Pada *driver* motor BLDC, digunakan ggl balik pada setiap belitan fasa untuk mendeteksi posisi rotor. Pada pin *frequency generator* (FG), rangkaian *driver* menghasilkan sinyal kotak dengan duty cycle 50%. Frekuensi sinyal kotak merepresentasikan kecepatan motor BLDC. Sinyal kotak pada pin FG dimasukkan pin digital *input* arduino. Untuk mengisolasi rangkaian arduino dengan *plant* digunakan *optocoupler*.

Gambar 1.6 Rangkaian Sensor Kecepatan Motor BLDC

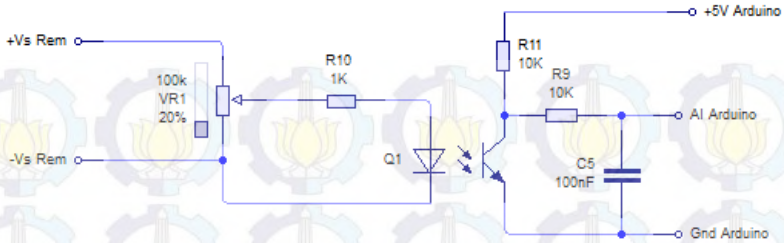
Untuk menghitung kecepatan motor dapat dilakukan dengan menghitung frekuensi gelombang kotak yang dihasilkan pin FG.Untuk mendapatkan kecepatan motor dapat digunakan Persamaan 3.1

$$N = \frac{60FG}{P} \tag{3.1}$$

N : Kecepatan motor (rpm)
FG : *Frequency generator* (Hz)
P : Jumlah pasang kutub

1.2.2.5 Sensor Tegangan Rem Elektromagnetik

Untuk mengetahui besarnya beban pengereman dapat dilakukan dengan cara mengukur tegangan suplai dari rem elektromagnetik. Rem elektromagnetik diberikan suplai 16-24 VDC. Untuk melakukan akuisisi data tegangan rem digunakan rangkaian pembagi tegangan. Karena suplai rem elektromagnetik berbentuk pulsa PWM, maka dibutuhkan *lowpass filter* untuk mengubah tegangan yang berbentuk pulsa menjadi tegangan analog. Bentuk diagram *wiring* sensor dapat dilihat pada Gambar 3.7.



Gambar 1.7 Diagram *Wiring* Sensor Tegangan Rem Elektromagnetik

1.3 Perancangan *Software*

Perancangan *software* dalam simulator BLDC meliputi Matlab, Arduino, dan LabVIEW

1.3.1 *Software* Matlab

Software Matlab digunakan untuk akuisisi data untuk dianalisis lebih lanjut. Untuk melakukan akuisisi data dari *plant* ke komputer digunakan *interface* berupa arduino. Proses transmisi data dilakukan menggunakan *port serial*. Pada simulink, dapat digunakan blok diagram *serial send* untuk mengirim data menuju *serial port*. *Serial receive* digunakan untuk membaca data *serial*.

Desain kontroler PID dan algoritma genetika juga dilakukan menggunakan matlab.

1.3.2 *Software* Arduino

Arduino pada simulator BLDC digunakan sebagai *interface* yang menghubungkan *plant* dengan komputer. Komunikasi data dilakukan secara *half duplex* melalui *port serial*. Pada sensor kecepatan motor, arduino digunakan untuk menghitung waktu on pulsa FG yang dihasilkan *driver* BLDC. Dengan menghitung frekuensi pulsa akan didapatkan kecepatan motor.

Pada rangkaian *driver* BLDC, arduino digunakan sebagai penghasil sinyal kontrol berupa PWM dalam *range* 33% hingga 100%.

Pada rangkaian sensor arus, arduino digunakan untuk mengukur tegangan analog yang dihasilkan sensor arus.

Pada rangkaian *driver* rem elektromagnetik, arduino digunakan untuk menghasilkan pulsa PWM. Pulsa PWM digunakan untuk mengatur tegangan suplai dari rem elektromagnetik. *Range* lebar pulsa PWM yang

digunakan sebesar 20%-80%. Tampilan dari program IDE Arduino dapat dilihat pada Gambar 3.8.

Gambar 1.8 Tampilan IDE Arduino.

1.3.3 *Software* LabVIEW

Software LabVIEW digunakan untuk proses identifikasi *plant*. Proses identifikasi membutuhkan data masukan dan keluaran dari sistem. Proses akuisisi data dilakukan menggunakan arduino sebagai *interface* melalui *port serial*. Pertimbangan penggunaan *software* LabVIEW untuk identifikasi karena proses identifikasi mebutuhkan waktu *sampling* yang cepat. Berdasarkan eksperimen, proses akuisisi data pada LabVIEW dapat dilakukan dengan waktu *sampling* paling cepat 30ms. Sedangkan pada matlab, waktu *sampling* paling cepat 100ms. Jika waktu *sampling* dipercepat maka data hasil akuisisi akan mengalami kerusakan. Proses komunikasi data dilakukan secara *half duplex* dan tersinkronisasi. Tampilan program akuisisi data dapat dilihat pada Gambar 3.9.

Gambar 1.9 Tampilan Program Untuk Akuisisi Data

1.4 Identifikasi dan Pemodelan Sistem

Pada pemodelan sistem di tugas akhir kali ini digunakan identifikasi dinamis. Sinyal uji yang digunakan adalah sunyal *pseudorandom binary sequence (PRBS)*. Bilangan random dibangkitkan dalam *range* 102-255. Nilai random ini menyatakan lebar pulsa PWM pada masukan *driver*. PWM keluaran arduino diatur pada *range* 2-5 V. Nilai PRBS diatur perubahannya setiap 16 kali *sampling*. Data masukan *driver* dan kecepatan motor akan diakuisisi dan diidentifikasi menggunakan *toolbox matlab system identification*. Dari hasil identifikasi akan didapatkan fungsi alih dari BLDC.

1.4.1 Pemodelan Motor BLDC

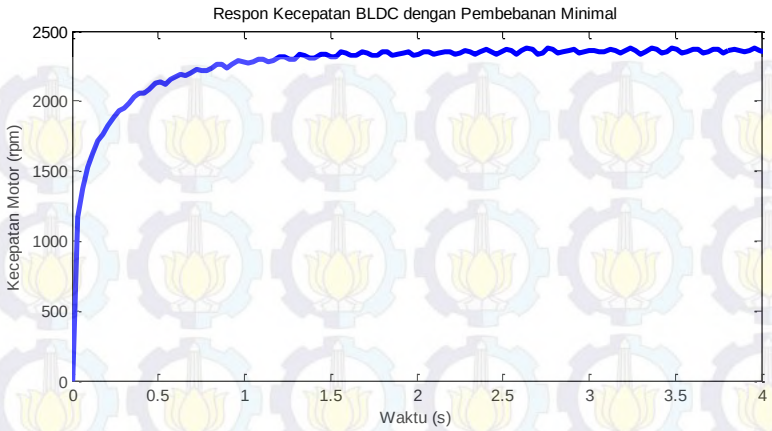
Pemodelan pada motor BLDC dilakukan pada tiga kondisi beban, yaitu beban minimal, beban nominal dan beban maksimal.

1.4.1.1 Pembebanan Minimal

Pada kondisi beban minimal motor dipasang seporos dengan rem elektromagnetik. Rem elektromagnetik diberikan suplai tegangan 16 VDC. Respon *open loop* motor BLDC pada beban minimal ditunjukkan pada Gambar 1.10 .Model yang didapatkan dari hasil identifikasi terdapat pada Tabel 1.4.

Tabel 1.4 Model BLDC pada Kondisi Beban Minimal

No	Model
1	$G_p(s) = \frac{1947s + 43100}{s^2 + 38,22s + 91,36}$
2	$G_p(s) = \frac{436100}{s^2 + 3448s + 3888}$



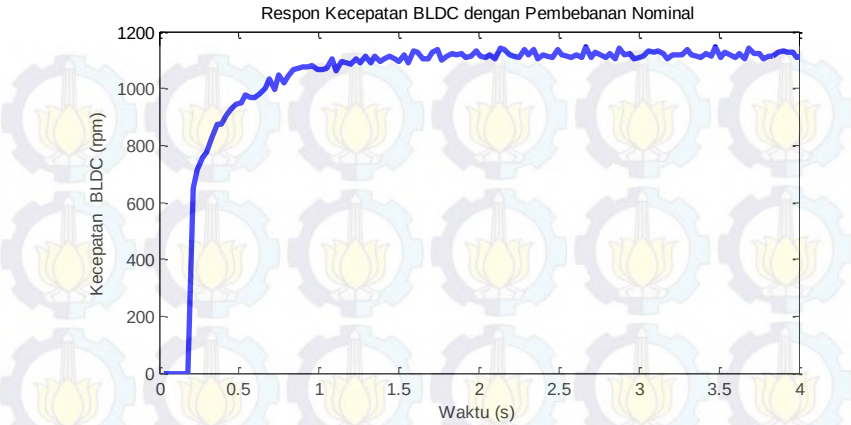
Gambar 1.10 Respon Kecepatan BLDC pada Kondisi Beban Minimal

1.4.1.2 Pembebanan Nominal

Pada kondisi beban nominal, rem elektromagnetik diberikan tegangan suplai 20 V. Respon kecepatan motor BLDC pada kondisi beban nominal ditunjukkan pada Gambar 1.11 . Model hasil identifikasi motor BLDC dengan pembebanan nominal ditunjukkan pada

Tabel 1.5 Model BLDC pada Kondisi Beban Nominal.

No	Model
1	$G_p(s) = \frac{23501{,}17s + 1907123{,}67}{s^2 + 1690{,}81s + 5854{,}55}$
2	$G_p(s) = \frac{126588{,}87}{s^2 + 10061{,}8s + 9609{,}92}$



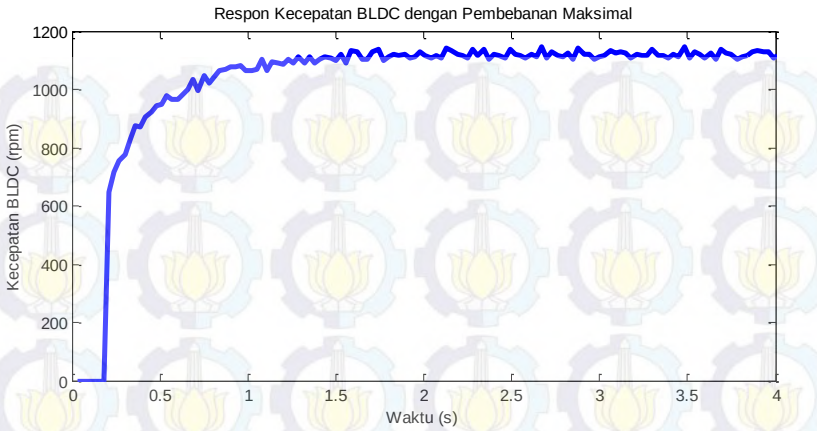
Gambar 1.11 Respon Kecepatan BLDC pada Kondisi Beban Nominal

1.4.1.3 Pembebanan Maksimal

Pada kondisi pembebanan maksimal, rem elektromagnetik diberikan tegangan suplai 24 V. Respon kecepatan motor BLDC pada kondisi beban maksimal ditunjukkan dalam Gambar 1.12. Model yang didapatkan dari hasil identifikasi dalam keadaan pembebanan maksimal ditunjukkan pada Tabel 3.6.

Tabel 1.6 Model BLDC pada Kondisi Beban Maksimal.

No	Model
1	$G_p(s) = \frac{1398.55s - 102.99}{s^2 + 4{,}19s + 0.021}$
2	$G_p(s) = \frac{2831474{,}95}{s^2 + 1972s + 9609{,}92}$



Gambar 1.12 Respon Kecepatan BLDC pada Kondisi Beban Maksimal.

1.4.2 Pengujian dan Validasi Model

Pengujian dan validasi model dilakukan untuk mengetahui kesesuaian model yang kita rancang dibandingkan dengan *plant* yang dimodelkan. Parameter yang digunakan untuk mengetahui keseuaian model adalah *relative steady state error (RSSE)*. Semakin kecil nilai RSSE, maka model hasil pendekatan semakin sesuai dengan *plant* yang dimodelkan. Untuk mendapatkan nilai RSSE dapat digunakan Persamaan 1.1

$$RMSE = \sqrt{\frac{\sum_{i=1}^{k}(y_i - \tilde{y}_i)^2}{k}}$$

$$RSSE = \frac{RMSE}{y_{ss}} x100\% \qquad 1.1$$

Tabel 1.7 Validasi Model BLDC pada Pembebanan Minimal

No	Model	RSSE (%)
1	$G_p(s) = \frac{1947s + 43100}{s^2 + 38,22s + 91,36}$	9,28
2	$G_p(s) = \frac{436100}{s^2 + 3448s + 3888}$	20,51

Hasil validasi model BLDC pada beban minimal ditampilkan pada Tabel 1.7.

Pada pembebanan nominal, hasil validasi model ditampilkan pada Tabel 1.8. Perbandingan respon model pendekatan dengan hasil pengukuran ditampilkan pada Tabel 1.8.

Tabel 1.8 Validasi Model BLDC pada Pembebanan Nominal

No	Model	RSSE (%)
1	$G_p(s) = \frac{23501{,}17s + 1907123{,}67}{s^2 + 1690{,}81s + 5854{,}55}$	9,25
2	$G_p(s) = \frac{126588{,}87}{s^2 + 10061{,}8s + 9609{,}92}$	9,79

1.5 Perancangan Kontroler PID Multiobjektif Berdasarkan Algoritma Genetika

Setelah fungsi alih dari sistem telah didapatkan, langkah selanjutnya adalah merancang kontroler untuk pembebanan nominal. Kontroler digunakan untuk mengembalikan respon ke nilai *set point* yang diinginkan sekalipun motor BLDC diberi beban. Tahapan desain kontroler ini meliputi perancangan kontroler PID dan perancangan algoritma genetika.

1.5.1 Kontroler PID

Struktur kontroler PID yang digunakan pada simulator BLDC menggunakan struktur paralel seperti pada Persamaan 3.2. Parameter K_p, K_i, K_d akan *dituning* menggunakan algoritma genetika.

$$C(s) = K_p + \frac{K_i}{s} + K_d s \qquad (1.2)$$

Objektif dari kontroler PID adalah menghasilkan respon *output* menyerupai sistem orde satu dengan *time constant* =1, *maximum overshoot*= 0% dan *steady state error* =0%

1.5.2 Algoritma Genetika

Algoritma genetika yang digunakan pada penelitian ini digunakan untuk melakukan *tuning* parameter dari kontroler PID. Algoritma genetika didesain menggunakan *software* Matlab 2013.

Individu pada algoritma genetika berisi tiga gen yang menyatakan nilai K_p, K_i, K_d. Proses pencarian nilai parameter dari setiap individu

digunakan fungsi *fitness*. Penilaian optimal atau tidaknya suatu individu berdasarkan pengujiannya menggunakan fungsi *fitness*. Parameter fungsi *fitness* terdiri dari *overshoot* dan *relative steady state error*. Hasil akhir dari algoritma genetika diharapkan didapatkan individu dengan nilai *fitness* paling besar yang mereperesentasikan solusi terbaik dari parameter kontroler PID.

Gambar 1.13 Individu pada Algoritma Genetika

Langkah pertama dari algoritma genetika adalah membangkitkan individu awal secara acak. *Range* nilai bilangan acak yang dibangkitkan berada pada *range* 0-50.

Langkah berikutnya adalah menghitung nilai *fitness* dari setiap individu yang dibangkitkan pada langkah pertama. Fungsi *fitness* dinyatakan dalam Persamaan 3.3,

$$f(i) = \alpha\left(100 - RSSE(i)\right) + \beta(100 - \mathrm{OS}(i))$$
$$\alpha + \beta = 1 \qquad (1.3)$$

α : Bobot *Relative Steady State Error*
RSSE : Relative Steady State Error
β : Bobot *overshoot*
OS : *Overshoot*

$$RMSE = \sqrt{\frac{\sum_{i=1}^{k} (error)^2}{k}}$$

$$RSSE = \frac{RMSE}{y_{ss}} x100\% \qquad (3.5)$$

$$OS = \frac{y_{\max} - y_{ss}}{y_{ss}} x100\% \qquad (3.6)$$

Setelah nilai *fitness* dihitung, individu diurutkan berdasarkan nilai *fitness*nya. Langkah selanjutnya adalah seleksi. Seleksi yang penulis gunakan ada dua, yaitu *truncking* dan *roullete.* Pada seleksi *truncking*, individu dengan nilai *fitness* yang tinggi akan terpilih menjadi induk. Jumlah individu yang terpilih tergantung pada nilai rasio *crossover*(Rc). Individu yang hilang dari proses seleksi *truncking* akan dikembalikan melalui seleksi *roullete. Pada* seleksi *roullete*, individu dipilih secara acak dari induk hingga mencapai jumlah populasi awal sebelum seleksi *truncking*.

Tahap selanjutnya adalah *crossover*. Metode *crossover* yang digunakan adalah metode *crossover* aritmatik karena nilai dari gen berbentuk desimal dengan *range* 0-50. Tahap selanjutnya adalah mutasi, banyaknya gen yang mengalami mutasi bergantung pada rasio mutasi (Rm).

Jika kondisi *stopping* belum terpenuhi, maka program kembali ke langkah penghitungan *fitness*. Pada program yang penulis desain, *stopping condition* yang digunakan adalah jumlah iterasi (generasi). Jika jumlah generasi belum mencapai generasi tertentu, maka program akan terus dijalankan hingga generasi yang ditargetkan terpenuhi.

Halaman ini sengaja dikosongkan

BAB 4
PENGUJIAN DAN ANALISIS

1.1 Pengujian Sistem

Pengujian sistem dilakukan agar komponen sistem yang telah dirancang dapat beroperasi sesuai desain.

Pada tahapan ini akan dilakukan beberapa jenis pengujian yaitu pengujian sensor, pengujian *open loop* dari motor BLDC, lalu pengujian kontroler yang disimulasikan pada hasil identifikasi model, lalu yang terakhir merupakan pengujian implementasi kontroler pada motor BLDC.

1.1.1 Pengujian Sensor Kecepatan Motor BLDC

Pada pengujian ini dilakukan pengecekan hasil pembacaan sensor kecepatan. Hasil keluaran sensor dibandingkan dengan pengukuran menggunakan *laser tachometer*. Hasil pembacaan sensor kecepatan memiliki kesalahan pengukuran paling besar 1,93%.

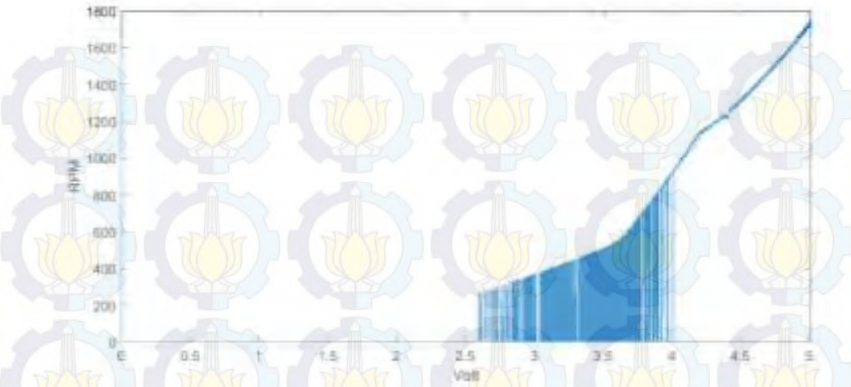
Tabel 1.1 Hasil Pengujian Sensor Kecepatan Motor.

Tachometer (rpm)	Sensor (rpm)	*Error* (%)
972,36	984,6	1,26
972,12	984,6	1,28
971,82	984,6	1,32
1022,1	1035,41	1,30
1020,6	1035,41	1,45
1021,8	1035,41	1,33
1059,1	1070,83	1,11
1057,2	1070,83	1,29
1058,5	1070,83	1,16
1201,8	1218,04	1,35
1202,2	1218,04	1,32
1203,4	1218,04	1,22
1292,8	1310,42	1,36

Tachometer (rpm)	Sensor (rpm)	*Error* (%)
1293,4	1310,42	1,32
1294,1	1310,42	1,26
1363,4	1382,67	1,41
1364,6	1382,67	1,32
1365,1	1382,67	1,29
1429,5	1448,82	1,35
1430,1	1448,82	1,31
1432,2	1448,82	1,16
1472,5	1491,69	1,30
1471,2	1491,69	1,39
1473,1	1491,69	1,26
1703,6	1726	1,31
1707,4	1726	1,09
1705,2	1726	1,22
2253	2290,45	1,66
2255	2290,45	1,57
2256	2290,45	1,53
2397	2441,21	1,84
2395	2441,21	1,93
2399	2441,21	1,76

1.1.2 Pengujian *Open Loop* Kecepatan Motor

Pengujian ini dilakukan untuk melihat hubungan antara *input* dan *output* dari motor BLDC. *Driver* BLDC diberikan masukan PWM dengan *range* 0 - 5 V. Hasil pengujian *open loop* kecepatan motor ditunjukkan dalam Gambar 1.1.



Gambar 1.1 Hubungan *Input Output* Kecepatan Motor.

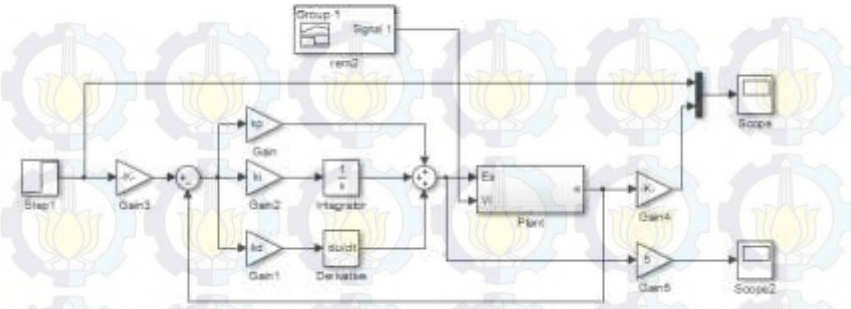
Pada gambar diatas, terdapat kesalahan pembacaan sensor pada saat tegangan *input driver* dibawah 1,65 V. Hal ini karena *range* tegangan *input driver* berada pada 1,65-5 V. Pada kondisi awal, terjadi lonjakan kecepatan hingga sekitar 2216 rpm. Hal ini karena mekanisme *starting driver* motor. Saat motor mulai berjalan dari keadaan berhenti, *driver* secara otomatis memberikan masukan tegangan PWM 5 V sampai ggl balik di setiap fasa terdeteksi. Setelah itu proses komutasi baru berjalan normal.

1.2 Simulasi Sistem

Simulasi merupakan salah satu hal yang harus dilakukan sebelum mengimplementasikan kontroler pada *plant*. Dengan hasil simulasi yang sesuai akan mempermudah dalam penerapan kontroler agar sistem mencapai kriteria yang diinginkan oleh penulis.

1.2.1 Diagram Simulink Simulasi Sistem

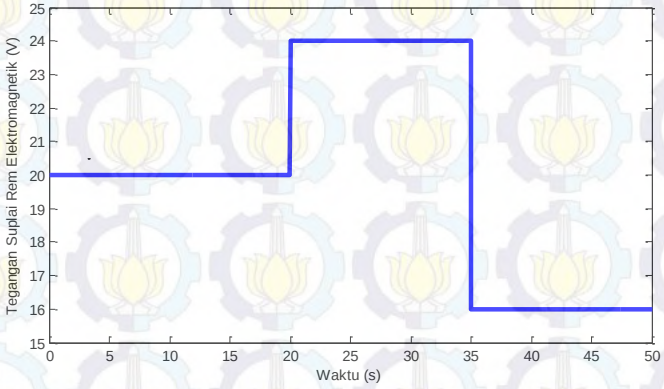
Berikut ini merupakan blok untuk simulasi dengan menggunakan kontroler PID.



Gambar 1.2 Blok Simulink Simulasi Sistem.

1.2.2 Program Algoritma Genetika

Pada tahapan ini, kontroler yang telah didesain akan disimulasikan pada *plant* BLDC. Proses *tuning* parameter kontroler PID dilakukan pada kondisi beban nominal menggunakan algoritma genetika. Pada pengujian kontroler, beban akan diubah-ubah untuk menguji performa kontroler saat ada perubahan beban. Gambar 4.3 merupakan pembebanan pada motor BLDC.



Gambar 1.3 Pembebanan pada Motor BLDC.

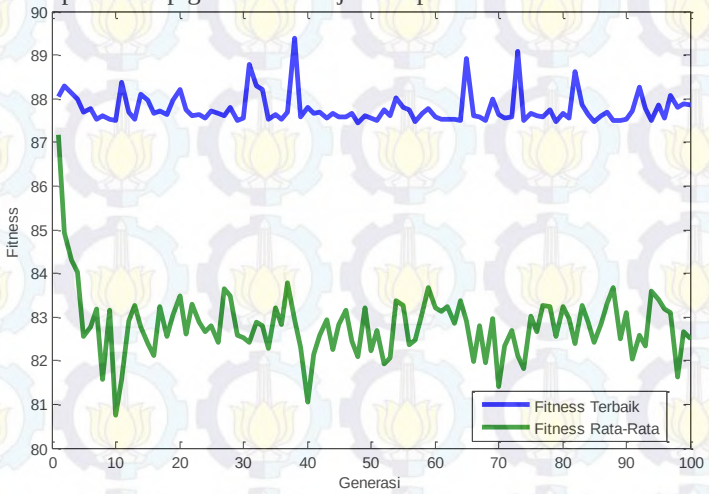
Selama proses *tuning*, parameter algoritma genetika yang diubah-ubah adalah populasi (P), generasi (G), rasio seleksi (Rs), rasio *crossover* (Rc), dan rasio mutasi (Rm).

Pada pengujian pertama penulis menggunakan parameter algoritma genetika yang terdapat pada Tabel 1.2.

Tabel 1.2 Parameter 1 Algoritma Genetika.

No	Parameter 1	Nilai
1	Populasi (P)	100
2	Generasi (G)	100
3	Rasio Seleksi (Rs)	0,5
4	Rasio *Crossover* (Rc)	0,5
5	Rasio Mutasi (Rm)	0,5

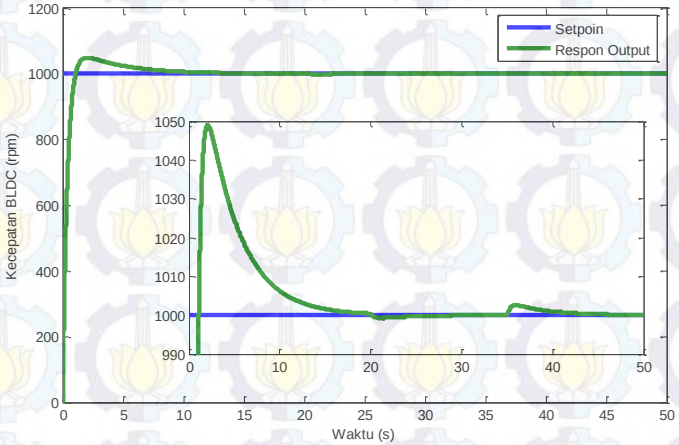
Setelah algoritma genetika dijalankan, perubahan nilai *fitness* individu pada setiap generasi ditunjukkan pada Gambar 1.4.



Gambar 1.4 Perubahan *Fitness* Individu Menggunakan Parameter 1.

Pada saat menggunakan parameter 1, algoritma genetika gagal mencapai konvergensi, hal ini dapat dilihat dari perubahan *fitness* terbaik di setiap generasi yang cenderung tetap di sekitar nilai 88. Setelah

dilakukan *tuning* parameter PID menggunakan parameter 1, didapatkan parameter Kp=53,226 Ki=13,367 Kd=19,253 dengan nilai *fitness* sebesar 87,67. Parameter hasil *tuning* akan disimulasikan pada *plant* BLDC dengan variasi beban seperti pada Gambar 1.3. Respon *output* sistem hasil simulasi digambarkan pada Gambar 1.5.



Gambar 1.5 Respon *Output* Sistem dengan Parameter 1.

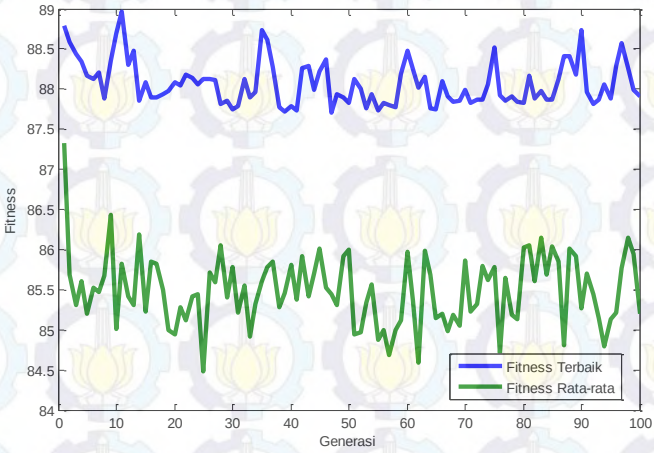
Saat menggunakan parameter 1, masih terdapat *overshoot* pada saat transien sebesar 4,9 %. *Rise time* sebesar 0,72 detik dan *settling time* sebesar 0,98 detik.

Pada percobaan kedua penulis menggunakan parameter 2 yang ditunjukkan pada Tabel 1.3. Nilai rasio seleksi diatur lebih besar dibandingkan rasio *crossover* dan rasio mutasi.

Tabel 1.3 Parameter 2 Algoritma Genetika.

No	Parameter 2	Nilai
1	Populasi (P)	100
2	Generasi (G)	100
3	Rasio Seleksi (Rs)	0,5
4	Rasio *Crossover* (Rc)	0,3
5	Rasio Mutasi (Rm)	0,3

Setelah algoritma genetika dijalankan, perubahan nilai *fitness* individu pada setiap generasi ditunjukkan pada Gambar 1.6.

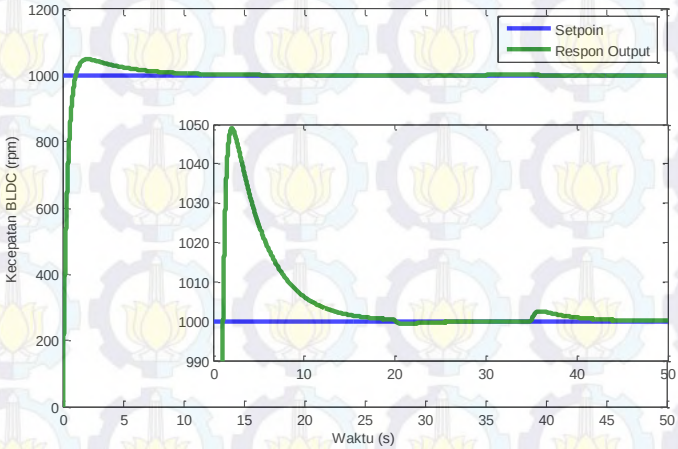


Gambar 1.6 Perubahan *Fitness* Individu Menggunakan Parameter 2.

Pada saat menggunakan parameter 2, algoritma genetika gagal mencapai konvergensi, hal ini dapat dilihat dari perubahan *fitness* terbaik di setiap generasi yang cenderung tetap di sekitar nilai 85. Pada parameter dua rasio seleksi dibuat lebih besar dibandingkan rasio mutasi dan rasio *crossover*. Hal ini menyebabkan kemungkinan terpilihnya individu dengan *fitness* kecil saat proses seleksi semakin besar. Sehingga nilai *fitness* gagal mencapai konvergensi menuju nilai terbaik. Tingginya nilai rasio mutasi juga menyebabkan perubahan yang besar pada individu setiap generasi.

Rasio seleksi yang tinggi menyebabkan turunnya peluang individu terbaik untuk terpilih kembali. Tapi hal ini menyebabkan ruang pencarian solusi algoritma genetika semakin lebar. Setelah dilakukan *tuning* parameter PID menggunakan parameter 2, didapatkan parameter Kp=9,09 Ki=20,35 dan Kd=0,72 dengan nilai *fitness* sebesar 88,23. Parameter hasil *tuning* akan disimulasikan pada *plant* BLDC dengan variasi beban seperti

pada Gambar 1.3. Respon *output* sistem hasil simulasi dinyatakan pada Gambar 1.7.



Gambar 1.7 Respon *Output* Sistem dengan Parameter 2.

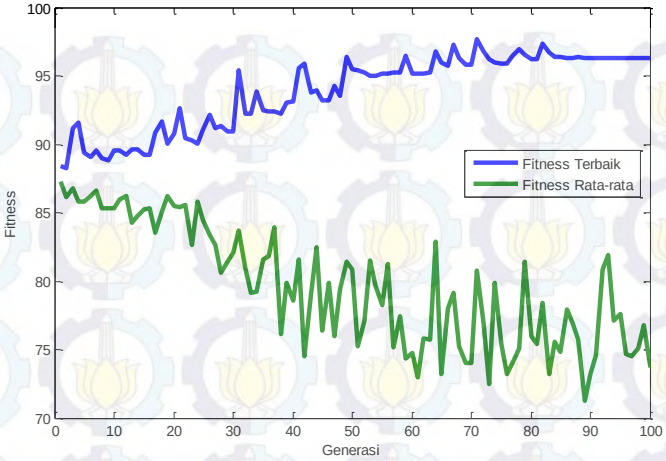
Saat menggunakan parameter 2, masih terdapat *overshoot* pada saat transien sebesar 5,5 %. *Rise time* sebesar 0,21 detik dan *settling time* sebesar 0,26 detik.

Pada percobaan ketiga penulis menggunakan parameter pada Tabel 1.4. Parameter rasio *crossover* diatur lebih tinggi daripada parameter rasio seleksi dan rasio mutasi.

Tabel 1.4 Parameter 3 Algoritma Genetika.

No	Parameter 3	Nilai
1	Populasi (P)	100
2	Generasi (G)	100
3	Rasio Seleksi (Rs)	0,3
4	Rasio *Crossover* (Rc)	0,5
5	Rasio Mutasi (Rm)	0,3

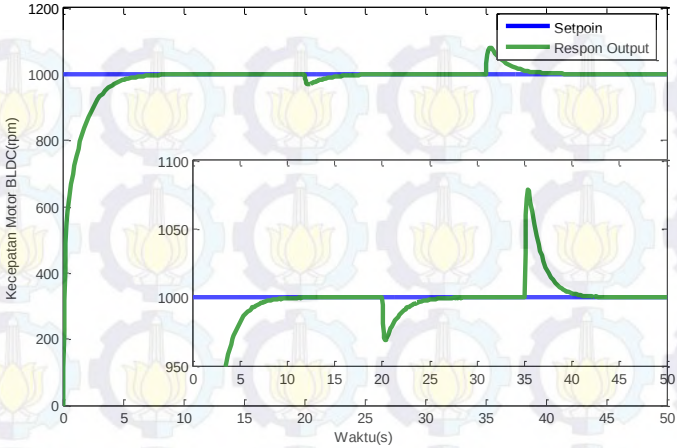
Setelah algoritma genetika dijalankan, perubahan nilai *fitness* individu pada setiap generasi ditunjukkan pada Gambar 1.8.



Gambar 1.8 Perubahan *Fitness* Individu Menggunakan Parameter 3.

Pada saat menggunakan parameter 3, *fitness* individu pada algoritma genetika mampu mencapai konvergensi. Rasio *crossover* yang tinggi menyebabkan perubahan yang sangat bervariasi setiap generasi. Hal ini dapat dilihat dari perubahan rata-rata *fitness* pada setiap generasi yang tinggi. Penurunan rasio seleksi menyebabkan nilai *fitness* individu mampu mencapai nilai konvergen.

Penurunan rasio mutasi juga berpengaruh dalam perubahan individu. Penurunan nilai rasio mutasi menyebabkan perubahan nilai *fitness* tidak terlalu besar seperti pada parameter 1 dan parameter 2. Setelah dilakukan *tuning* parameter PID menggunakan parameter 3, didapatkan parameter Kp=0,972 Ki=1,277 Kd=0,029 dengan nilai *fitness* sebesar 96,32. Parameter hasil *tuning* akan disimulasikan pada *plant* BLDC dengan variasi beban seperti pada Gambar 1.3. Respon *output* sistem hasil simulasi digambarkan pada Gambar 1.9.



Gambar 1.9 Respon *Output* Sistem dengan Parameter 3.

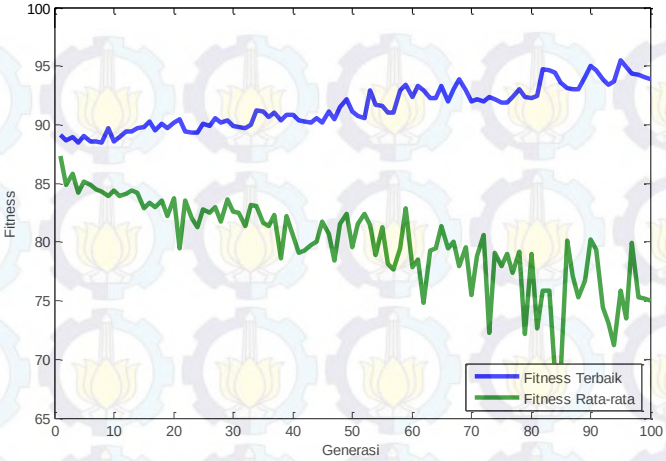
Pada saat simulasi sistem menggunakan parameter 3, respon *output* sistem menyerupai orde satu, tanpa *overshoot. Rise time* sebesar 2,65 detik dan *settling time* sebesar 3,52 detik.

Pada percobaan keempat penulis menggunakan parameter algoritma genetika pada Tabel 1.5. Rasio mutasi diatur lebih besar daripada rasio *crossover* dan rasio seleksi.

Tabel 1.5 Parameter 4 Algoritma Genetika.

No	Parameter 4	Nilai
1	Populasi (P)	100
2	Generasi (G)	100
3	Rasio Seleksi (Rs)	0,3
4	Rasio *Crossover* (Rc)	0,3
5	Rasio Mutasi (Rm)	0,5

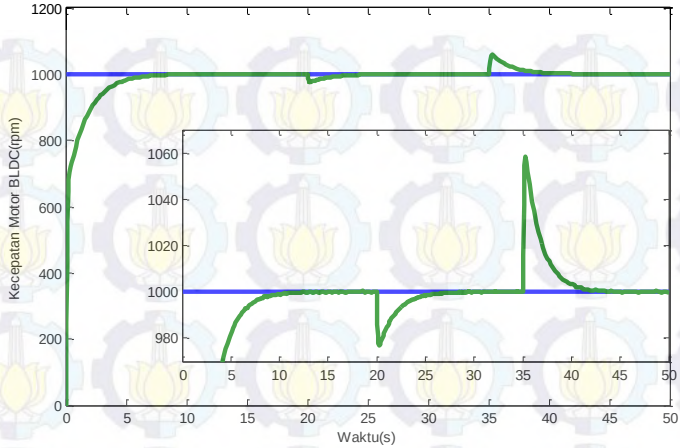
Setelah algoritma genetika dijalankan, perubahan nilai *fitness* individu pada setiap generasi ditunjukkan pada Gambar 1.10.



Gambar 1.10 Perubahan *Fitness* Individu Menggunakan Parameter 4.

Pada saat menggunakan parameter 4, *fitness* individu pada algoritma genetika belum mampu mencapai konvergensi meskipun ada kecenderungan nilai *fitness*nya naik. Rasio mutasi yang tinggi menyebabkan perubahan yang sangat bervariasi setiap generasi. Hal ini dapat dilihat dari perubahan rata-rata *fitness* pada setiap generasi yang tinggi. Selain itu nilai *fitness* terbaik juga terus mengalami perubahan pada setiap generasi meskipun memiliki kecenderungan naik.

Hal ini disebabkan rasio seleksi yang diturunkan menjadi 0,3. Ketika rasio seleksi besar, kemungkinan terpilihnya individu dengan *fitness* yang kecil saat seleksi roulete semakin besar. Setelah dilakukan *tuning* parameter PID menggunakan parameter 4, didapatkan parameter Kp=1,833 Ki=1,656 dan Kd=0,038 dengan nilai *fitness* sebesar 93,91. Parameter hasil *tuning* akan disimulasikan pada *plant* BLDC dengan variasi beban seperti pada Gambar 1.3. Respon *output* sistem hasil simulasi digambarkan pada Gambar 1.11.



Gambar 1.11 Respon *Output* Sistem dengan Parameter 4

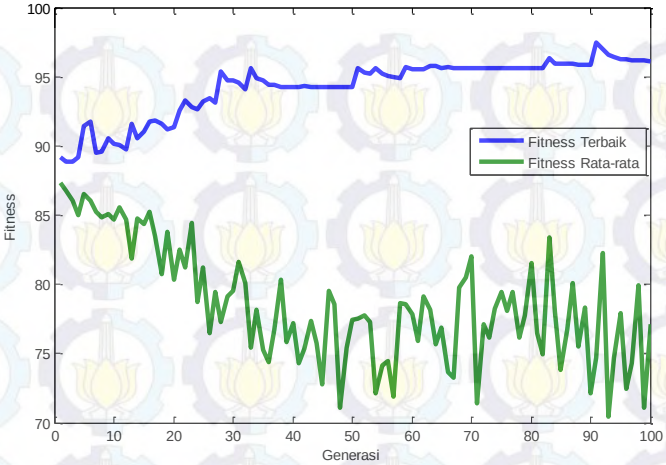
Pada saat simulasi sistem menggunakan parameter 4, respon *output* sistem menyerupai orde satu, tanpa *overshoot. Rise time* sebesar 2,13 detik dan *settling time* sebesar 3,21 detik.

Pada percobaan kelima penulis menggunakan parameter pada Tabel 1.6. Nilai rasio *crossover*, rasio mutasi dan rasio seleksi diturunkan menjadi 0,3. Pemilihan parameter ini digunakan untuk membandingkan dengan parameter 1 algoritma genetika.

Tabel 1.6 Parameter 5 Algoritma Genetika.

No	Parameter 5	Nilai
1	Populasi (P)	100
2	Generasi (G)	100
3	Rasio Seleksi (Rs)	0,3
4	Rasio *Crossover* (Rc)	0,3
5	Rasio Mutasi (Rm)	0,3

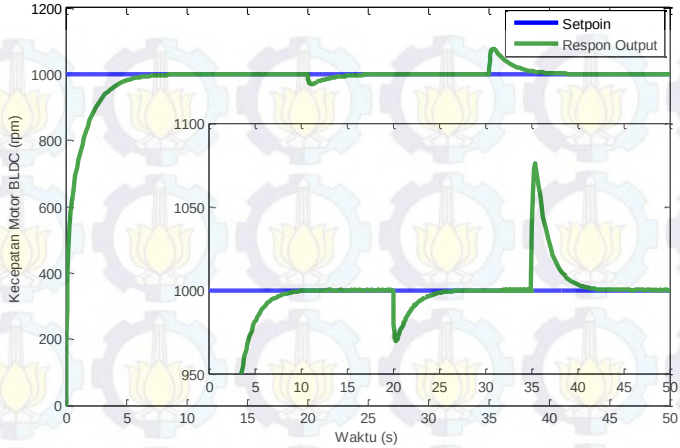
Setelah algoritma genetika dijalankan, perubahan nilai *fitness* individu pada setiap generasi ditunjukkan pada Gambar 1.12.



Gambar 1.12 Perubahan *Fitness* Individu Menggunakan Parameter 5.

Pada saat menggunakan parameter 5, *fitness* individu pada algoritma genetika mampu mencapai konvergensi. Dibandingkan dengan parameter 1, hasil *tuning* menggunakan parameter 5 memberikan hasil yang lebih baik. Nilai *fitness* individu dapat mencapai konvergensi dikarenakan turunnya rasio seleksi menyebabkan individu dengan *fitness* baik memiliki peluang terpilih lebih besar untuk terpilih.

Selain itu turunnya rasio mutasi juga berpengaruh dalam penurunan perubahan nilai *fitness* setiap pergantian generasi. Setelah dilakukan *tuning* parameter PID menggunakan parameter 5, didapatkan parameter Kp=1,032 Ki=1,278 Kd=0,047 dengan nilai *fitness* sebesar 96,12. Parameter hasil *tuning* akan disimulasikan pada *plant* BLDC dengan variasi beban seperti pada Gambar 1.3. Respon *output* sistem hasil simulasi digambarkan pada Gambar 1.13.



Gambar 1.13 Respon *Output* Sistem dengan Parameter 5

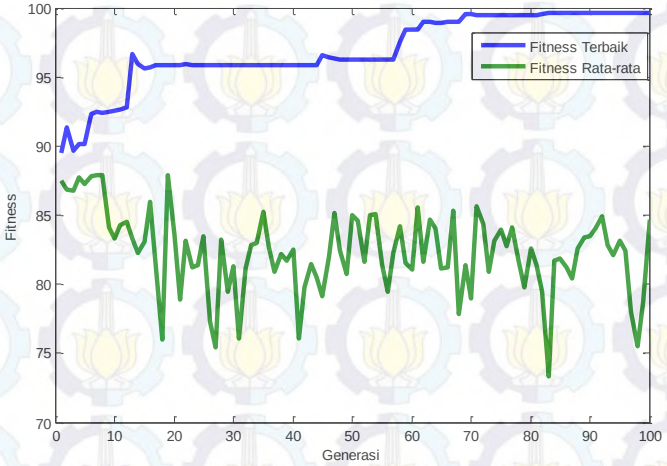
Pada saat simulasi sistem menggunakan parameter 5, respon *output* sistem menyerupai orde satu, tanpa *overshoot. Rise time* sebesar 2,49 detik dan *settling time* sebesar 3,46 detik.

Pada percobaan keenam penulis menggunakan parameter algoritma genetika pada Tabel 1.7. Nilai rasio *crossover*, rasio mutasi, rasio seleksi diturunkan menjadi 0,2. Pemilihan nilai parameter 6 digunakan untuk membandingkan dengan nilai parameter 1 dan parameter 5.

Tabel 1.7 Parameter 6 Algoritma Genetika.

No	Parameter 6	Nilai
1	Populasi (P)	100
2	Generasi (G)	100
3	Rasio Seleksi (Rs)	0,2
4	Rasio *Crossover* (Rc)	0,2
5	Rasio Mutasi (Rm)	0,2

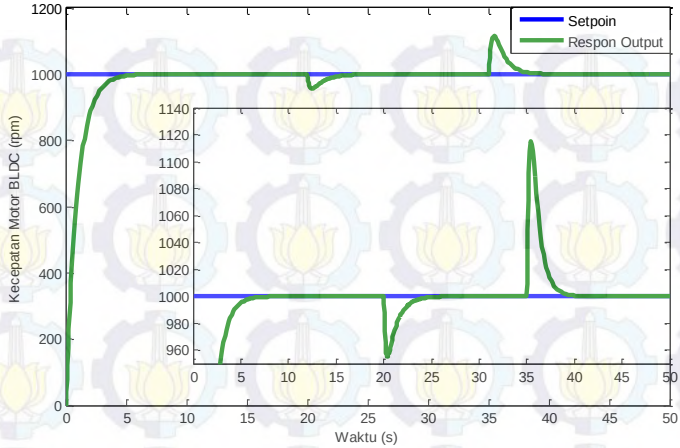
Setelah algoritma genetika dijalankan, perubahan nilai *fitness* individu pada setiap generasi ditunjukkan pada Gambar 1.14.



Gambar 1.14 Perubahan *Fitness* Individu Menggunakan Parameter 6.

Pada saat menggunakan parameter 6, *fitness* individu pada algoritma genetika mampu mencapai konvergensi. Dibandingkan dengan parameter 1 dan parameter 5, hasil *tuning* menggunakan parameter 6 memberikan hasil yang lebih baik. Turunnya rasio mutasi menurunkan variasi perubahan nilai *fitness* sehingga tidak terlalu banyak perubahan *fitness* setiap perubahan generasi. Hal ini dapat dilihat pada nilai *fitness* terbaik yang memiliki sedikit variasi di setiap perubahan generasi.

Turunnya rasio seleksi menyebabkan individu dengan *fitness* baik memiliki peluang terpilih lebih besar. Penurunan rasio *crossover* menyebabkan perubahan pada individu akibat *crossover* berkurang. Hal ini ditandai dengan perubahan *fitness* rata-rata pada parameter 6 lebih kecil dibandingkan pada parameter 5 dan parameter 1 algoritma genetika. Setelah dilakukan *tuning* parameter PID menggunakan parameter 6, didapatkan parameter Kp=0,238 Ki=1,148 Kd=0,013 dengan nilai *fitness* sebesar 99,64. Parameter hasil *tuning* akan disimulasikan pada *plant* BLDC dengan variasi beban seperti pada Gambar 1.3. Respon *output* sistem hasil simulasi digambarkan pada Gambar 1.15.



Gambar 1.15 Respon *Output* Sistem dengan Parameter 6

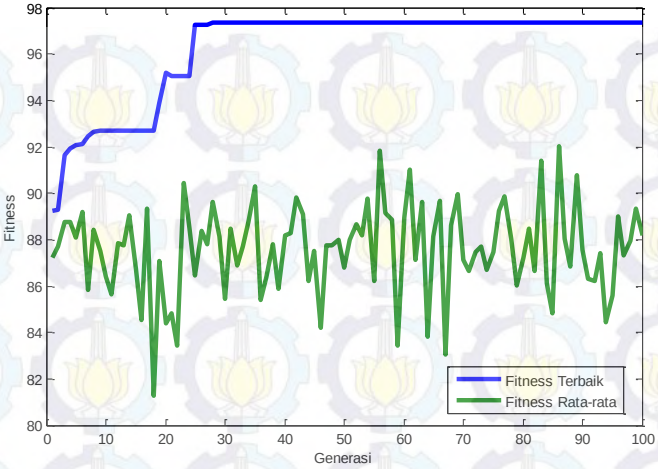
Pada saat simulasi sistem menggunakan parameter 6, respon *output* sistem menyerupai orde satu, tanpa *overshoot. Rise time* sebesar 2,20 detik dan *settling time* sebesar 2,81 detik.

Pada percobaan ketujuh penulis menggunakan parameter algoritma genetika pada Tabel 1.8. Pertimbangan pemilihan nilai parameter 7 algoritma genetika untuk dilakukan perbandingan dengan parameter 6.

Tabel 1.8 Parameter 7 Algoritma Genetika.

No	Parameter 7	Nilai
1	Populasi (P)	100
2	Generasi (G)	100
3	Rasio Seleksi (Rs)	0,1
4	Rasio *Crossover* (Rc)	0,1
5	Rasio Mutasi (Rm)	0,1

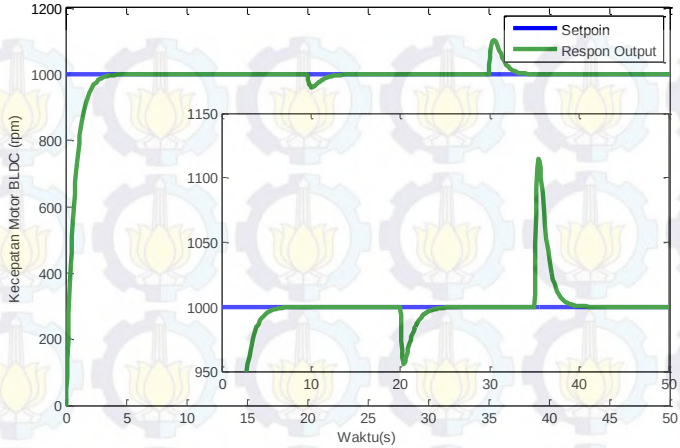
Setelah algoritma genetika dijalankan, perubahan nilai *fitness* individu pada setiap generasi ditunjukkan pada Gambar 1.16.



Gambar 1.16 Perubahan *Fitness* Individu Menggunakan Parameter 7.

Pada saat menggunakan parameter 7, *fitness* individu pada algoritma genetika mampu mencapai konvergensi. Dibandingkan dengan parameter 1 dan parameter 5, hasil *tuning* menggunakan parameter 7 memberikan hasil yang lebih baik. Tetapi tidak lebih baik daripada parameter 6. Hal ini disebabkan rasio mutasi dan rasio *crossover* yang terlalu kecil, sehingga ruang pencarian algoritma genetika terlalu sempit dan tidak mampu mencari solusi terbaik.

Rasio seleksi yang kecil memungkinkan generasi terbaik memiliki peluang terpilih lebih besar. Tetapi juga menyebabkan turunnya variasi individu. Hal ini mempersempit ruang pencarian algoritma genetika. Hal ini membuat algoritma genetika sudah mencapai konvergensi pada generasi ke 40. Setelah dilakukan *tuning* parameter PID menggunakan parameter 7, didapatkan parameter Kp=0,335 Ki=1,536 Kd=0.038 dengan nilai *fitness* sebesar 97,33. Parameter hasil *tuning* akan disimulasikan pada *plant* BLDC dengan variasi beban seperti pada Gambar 1.3. Respon *output* sistem hasil simulasi digambarkan pada Gambar 1.17



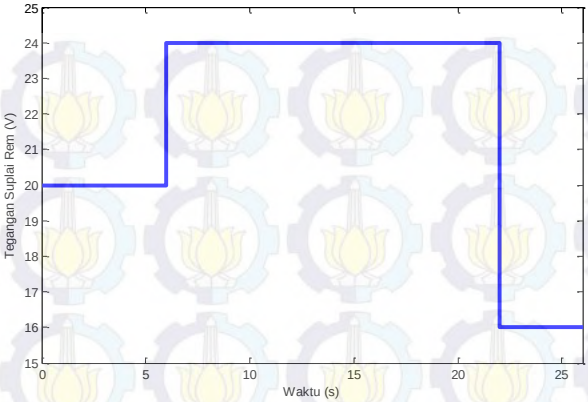
Gambar 1.17 Respon *Output* Sistem dengan Parameter 7

Pada saat simulasi sistem menggunakan parameter 6, respon *output* sistem menyerupai orde satu, tanpa *overshoot*. *Rise time* sebesar 1,585 detik dan *settling time* sebesar 2,05 detik.

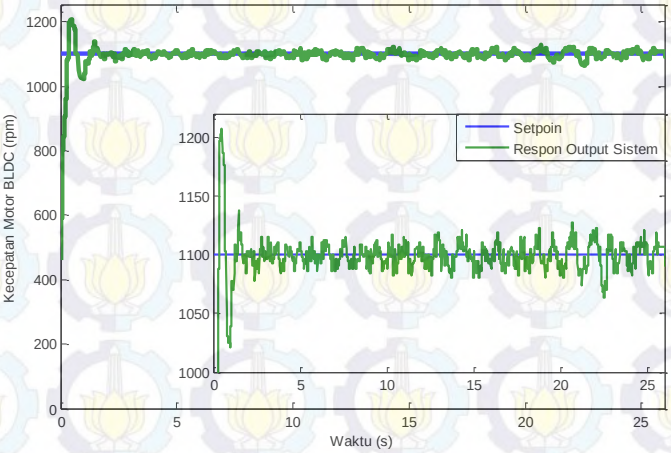
Dari percobaan pertama hingga ketujuh, nilai *fitness* terbesar didapatkan saat percobaan keenam dengan nilai *fitness* 99,64 dengan nilai parameter kontroler PID Kp=0,238 Ki=1,148 Kd=0,013. Nilai parameter PID yang didapatkan pada percobaan keenam akan diimplementasikan pada *plant*.

1.3 Implementasi sistem

Pada implementasi sistem digunakan parameter Kp=0,238 Ki=1,148 Kd=0,013. Kecepatan motor BLDC diatur pada 1100 rpm. Beban rem elektromagnetik diubah ubah untuk memberikan efek pembanan. Perubahan torsi pengereman dilakukan dengan cara mengubah tegangan suplai rem. Nilai tegangan suplai rem ditunjukkan pada Gambar 1.18 . Respon *output* sistem ditunjukkan pada Gambar 1.19.



Gambar 1.18 Pembebanan Motor BLDC



Gambar 1.19 Respon *Output* Hasil Implementasi

Pada kondisi awal, motor diberikan beban nominal(20VDC). Pada saat t=6 beban diganti menjadi beban maksimal (24VDC). Pada saat t=22 beban diturunkan menjadi beban minimal(16V). Terdapat *overshoot* sebesar 8% dan *error* steady *state* sebesar 2,1 %.

Halaman ini sengaja dikosongkan

BAB 5
PENUTUP

1.1 Kesimpulan

Setelah dilakukan pengujian dan analisis dapat ditarik beberapa kesimpulan sebagai berikut,

1. Algoritma genetika memberikan hasil optimal saat diberikan parameter P=100, G=100, Rc=0.2, Rs=0,2, Rm=0,2.
2. Dari hasil *tuning* kontroler PID menggunakan algoritma genetika didapatkan parameter optimal sebagai berikut, Kp=0,238, Ki=1,148, Kd=0,013
3. Kontroler PID menggunakan parameter pada poin 2 mampu mengatur kecepatan motor BLDC dan mampu mengatasi efek pembebanan.

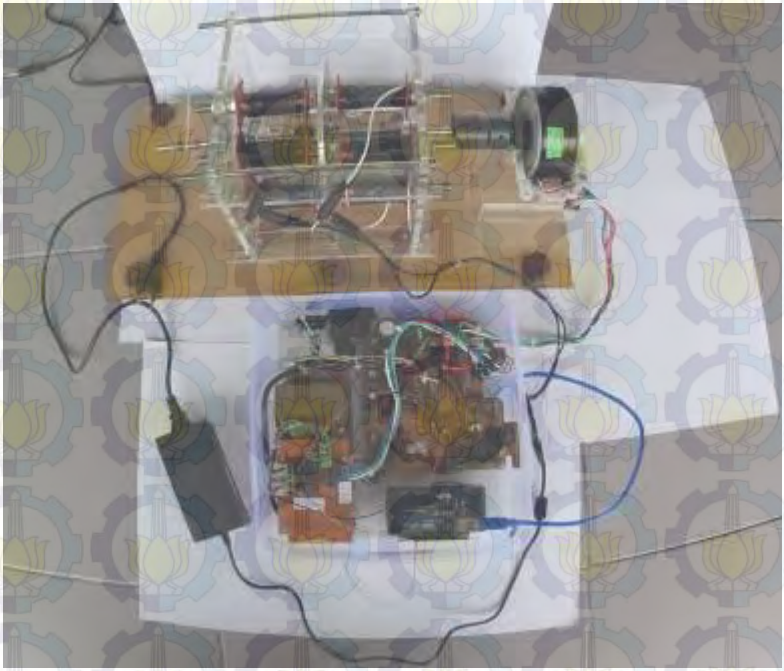
1.2 Saran

Untuk penelitian selanjutnya diharapkan melakukan identifikasi berdasarkan parameter motor BLDC.

Halaman ini sengaja dikosongkan

LAMPIRAN

Bentuk Fisik *Plant* Simulator BLDC



Program Matlab Algoritma Genetika
Program Utama

```
clear all;
hold off;
%Parameter program algoritma genetika
Npop=input('Masukkan Jumlah Populasi=');
jumlah_iterasi=input('Masukkan Jumlah
Generasi=');
rasio_seleksi=input('Masukkan Rasio Seleksi=');
p_crossover=input('Masukkan Rasio Crossover=');
p_mutasi=input('Masukkan Probabilitas Mutasi=');
Nbits=3;
iterasi=1;
%Pembangkitan populasi awal
POP=generate(Npop,Nbits);
POP7=[];
POP8=[];
POP9=[];
while iterasi<=jumlah_iterasi
    %Penghitungan nilai fitness
    [POP1]=fitnesscoba(POP,Npop);
    [POP8]=[POP8;POP1(1,4)];
    [POP7]=[POP7;POP1];
    [POP9]=[POP9;mean(POP1(:,4))]
    iterasi=iterasi+1;
    %plot hasil bagian 1
    dscatter(iterasi-1,1)=iterasi-1;
    dscatter(iterasi-1,2)=POP1(1,4);
    %Seleksi Truncking dan Mesin Roullet
    [POP3]=seleksi(POP1,rasio_seleksi);
    %Crossover
    [POP4]=crossover(POP3,p_crossover);
    %Mutasi
    [POP]=mutasi(POP4,p_mutasi,Npop);
end;
%Penghitungan nilai fitness bagian II
[POP6]=fitnesscoba(POP,Npop);
%plot hasil bagian 2
dscatter=[1:(iterasi-1)*Npop]';
dscatter=[dscatter POP7];
```

```
scatter(dscatter(:,1),dscatter(:,5))
%plot best individual
figure(2)
tt=1:100;
ttt=tt';
plot(ttt,POP8,ttt,POP9);
%menampilkan nilai Q dan R paling optimal
kp=POP6(1,1);
ki=POP6(1,2);
kd=POP6(1,3);
bestfitness=POP6(1,4);
    u=20;
    k=2.71375624999999*u^2-
91.7800749999996*u+872.337599999996;
    a=0.540574999999998*u^2-
18.5418499999999*u+183.594899999999;
    b=3.22795312499999*u^2-
107.205937500000*u+987.516999999996;
    gp=tf([k],[1 a b]);
    gr=tf(1,[1 1]);
    gc=pid(kp,ki,kd);
    gfb=feedback(gp*gc,1);
    [yp,tp]=step(gfb,0.01:0.01:50);
    [yr,tr]=step(gr,0.01:0.01:50);
    figure(3)
    plot(tr,yr,tp,yp)
```

Program Fungsi *Fitness*

```
%Fungsi Fitness
function [POP1]=fitnesscoba(POP,Npop)
fitness=[];
for i=1:Npop;
    kp=POP(i,1);
    ki=POP(i,2);
    kd=POP(i,3);
    sp=1200;
    kos=0.3;
    krsse=0.7;
    kst=0;
```

```
    u=20;
    k=2.71375624999999*u^2-
91.7800749999996*u+872.337599999996;
    a=0.540574999999998*u^2-
18.5418499999999*u+183.594899999999;
    b=3.22795312499999*u^2-
107.205937500000*u+987.516999999996;
    gp=tf([k],[1 a b]);
    gr=tf(1,[1 1]);
    gc=pid(kp,ki,kd);
    gfb=feedback(gp*gc,1);
    [yp,tp]=step(gfb,0.01:0.01:15);
    [yr,tr]=step(gr,0.01:0.01:15);
    tt=length(tp);
    yss=mean(yp(tt-(0.2*tt):tt));
    os=(max(yp)-1)*100/1;
    if os>100;
        os=100;
    end
    rsse=(rms(yp-yr))*100/1;
    if rsse>100;
        rsse=100;
    end
    fitness(i,1)=kos*(100-os)+krsse*(100-rsse);
    %fitness(i,2)=os;
    %fitness(i,3)=rsse;
end;
fitness;
POP1=[POP fitness];
v=[];
%mengurutkan populasi berdasarkan nilai fitness
for w=1:Npop
    for x=1:Npop
        if POP1(w,4)>POP1(x,4)
            v=POP1(w,:);
            POP1(w,:)=POP1(x,:);
            POP1(x,:)=v;
        end;
```

```
    end;
end;
```

Program *Crossover*

```
%Perkawinan Silang
function [POP4]=crossover(POP3,p_crossover)
i=0;
[a, b]=size(POP3);
while i<(a-1)
    i=i+1;
    cut_point=randi([1,3],[1,1]);
    if cut_point==1

anak(i,:)=[(POP3(i,1)*p_crossover)+(POP3(i+1,1)*
(1-p_crossover)) POP3(i,2:3)];
    elseif cut_point==2

anak(i,:)=[(POP3(i,1)*p_crossover)+(POP3(i+1,1)*
(1-p_crossover))
(POP3(i,2)*p_crossover)+(POP3(i+1,2)*(1-
p_crossover)) POP3(i,3)];
    else

anak(i,:)=[(POP3(i,1)*p_crossover)+(POP3(i+1,1)*
(1-p_crossover))
(POP3(i,2)*p_crossover)+(POP3(i+1,2)*(1-
p_crossover))
(POP3(i,3)*p_crossover)+(POP3(i+1,3)*(1-
p_crossover))];
    end;
end;
for i=a
    cut_point=randi([1,3],[1,1]);
    if cut_point==1

anak(i,:)=[(POP3(i,1)*p_crossover)+(POP3(1,1)*(1
-p_crossover)) POP3(i,2:3)];
    elseif cut_point==2

anak(i,:)=[(POP3(i,1)*p_crossover)+(POP3(1,1)*(1
```

```
-p_crossover))
(POP3(i,2)*p_crossover)+(POP3(1,2)*(1-
p_crossover)) POP3(i,3)];
    else

anak(i,:)=[(POP3(i,1)*p_crossover)+(POP3(1,1)*(1
-p_crossover))
(POP3(i,2)*p_crossover)+(POP3(1,2)*(1-
p_crossover))
(POP3(i,3)*p_crossover)+(POP3(1,3)*(1-
p_crossover))];
    end;
end;
POP4=anak;
```

Program Mutasi

```
%Mutasi Random
function [POP5]=mutasi(POP4,p_mutasi,Npop)
POP5=POP4;
jumlah=floor(Npop*3*p_mutasi);
posisi=randi([1,Npop*3],[1,jumlah]);
for i=1:jumlah;
    if mod(posisi(1,i),3)==0;
        POP5(posisi(1,i)/3,3)=rand(1,1);
    elseif mod(posisi(1,i),3)==2;

POP5(ceil(posisi(1,i)/3),2)=rand(1,1)*randi([1,1
000],[1,1]);
    else

POP5(ceil(posisi(1,i)/3),1)=rand(1,1)*randi([1,1
000],[1,1]);
    end;
end;
```

Program untuk membangkitkan populasi awal

```
function POP=generate(Npop,Nbits)
```

```
xx1=rand(Npop,1);
xx2=rand(Npop,1);
xx3=rand(Npop,1);
i=1:Npop;
yy1(i)=0.001+ ( (xx1(i)-min(xx1))/(max(xx1)-
min(xx1)))*(50-0.0001);
yy2(i)=0.001+ ( (xx2(i)-min(xx2))/(max(xx2)-
min(xx2)))*(50-0.0001);
yy3(i)=0.001+ ( (xx3(i)-min(xx3))/(max(xx3)-
min(xx3)))*(50-0.0001);
POP(:,1)=yy1';
POP(:,2)=yy2';
POP(:,3)=yy3';
```

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] R. Krishnan, *Permanent Magnet Synchronous and Brushless DC Motor Drives*, Blacksburg: CRC Press, 2010.

[2] C. l. Xia, *Permanent magnet brushless DC motor drives and controls*, Tianjin: Science Press, 2012.

[3] D. Wheat, *Arduino Internals*, New York, Appress, 2011.

[4] R. W. Larsen, *LabVIEW for Engineers*, Prentice *Hall*, 2011.

[5] I. Elrosa, "Traction Control pada Parallel Hybrid Electric Vehicle dengan Metode Generalized Predictive Control,"*Tugas Akhir*, Surabaya: Institut Teknologi Sepuluh Nopember, 2014.

[6] K. J. Astrom and T. Hdgglund, *Advanced PID Control*, ISA - Instrumentation, Systems, and Automation Society, 2005.

[7] M. Gen and R. Cheng, *Genetic Algorithm and Engineering Design*, New York: John Wiley & Sons Inc., 1997.

[8] K. Ogata, *Modern Control Engineering*, 4th ed., Prentice *Hall*, 2002.

[9] A. S. Nugroho, "Optimalisasi Linear Quadratic Integral Tracking Dengan Algoritma Genetika Untuk Pengaturan Akselerasi Simulator Parallel Hybrid Electric Vehicle," ,*Tugas Akhir*, Institut Teknologi Sepuluh Nopember, 2015.

Halaman ini sengaja dikosongkan

RIWAYAT HIDUP

M. Safrurriza, dilahirkan di Jombang, Jawa Timur, Indonesia pada tahun 1993. Penulis sedang menempuh pendidikan S1 di Teknik Sistem Pengaturan, Jurusan Teknik Elektro, Institut Teknologi Sepuluh Nopember Surabaya. Penulis tertarik pada topik penelitian seputar pengaturan motor listrik, *artificial intelligence*, dan *power electronics*.