

TUGAS AKHIR - EE184801

IMPLEMENTASI KONTROL KECEPATAN MOTOR INDUKSI 3 FASA MENGGUNAKAN VFD DENGAN KONTROLER PI-GA

BAMBANG PERMADI WIDYASMARA

NRP 07111940000222

Dosen Pembimbing

Eka Iskandar, ST., MT.

NIP 198005282008121001

Ir. Ali Fatoni, M.T.

NIP 196206031989031002

Program Studi Sarjana Teknik Elektro

Departemen Teknik Elektro

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2023



TUGAS AKHIR - EE184801

IMPLEMENTASI KONTROL KECEPATAN MOTOR INDUKSI 3 FASA MENGGUNAKAN VFD DENGAN KONTROLER PI-GA

BAMBANG PERMADI WIDYASMARA

NRP 07111940000222

Dosen Pembimbing

Eka Iskandar, ST., MT.

NIP 198005282008121001

Ir. Ali Fatoni, M.T.

NIP 196206031989031002

Program Studi Sarjana Teknik Elektro

Departemen Teknik Elektro

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2023



FINAL PROJECT - EE184801

IMPLEMENTATION OF 3 PHASE MOTOR INDUCTION USING VFD AND PI-GA CONTROLLER

BAMBANG PERMADI WIDYASMARA

NRP 07111940000222

Advisor

Eka Iskandar, ST., MT.

NIP 198005282008121001

Ir. Ali Fatoni, M.T.

NIP 196206031989031002

Undergraduate Program of Electrical Engineering

Department of Electrical Engineering

Faculty of Intelligent Electrical and Informatics Technology

Institut Teknologi Sepuluh Nopember

Surabaya

2023

LEMBAR PENGESAHAN

**Implementasi Kontrol Kecepatan Motor Induksi 3 Fasa menggunakan VFD dengan
Kontroler PI-GA**

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat

memperoleh gelar Sarjana Teknik pada

Program Studi S-1 Teknik Elektro

Departemen Teknik Elektro

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Oleh : **BAMBANG PERMADI WIDYASMARA**

NRP. 07111940000222

Disetujui oleh Tim Penguji Tugas Akhir :

1. Eka Iskandar, ST., MT.

Pembimbing

2. Ir. Ali Fatoni, M.T.

Ko-pembimbing

3. Yurid Eka Nugraha, ST, M.Eng, PhD

Penguji

4. Dr.Ir. Ari Santoso, DEA.

Penguji

5. Dr. Trihastuti Agustinah, ST., MT.

Penguji

SURABAYA

Juli, 2023

Halaman ini sengaja dikosongkan

APPROVAL SHEET

Implementation Of Speed Control 3 Phase Induction Motor Using VFD with PI-GA Controller

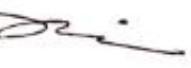
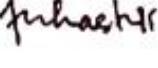
FINAL PROJECT

Submitted to fulfill one of the requirements
for obtaining a Bachelor Engineering degree at
Undergraduate Program of Electrical Engineering
Department of Electrical Engineering
Faculty of Intelligent Electrical and Informatics Technology
Sepuluh Nopember Institute of Technology

By : **BAMBANG PERMADI WIDYASMARA**

NRP. 07111940000222

Approval by Final Project Examiner Team :

- | | | |
|---------------------------------------|------------|---|
| 1. Eka Iskandar, ST., MT. | Advisor |  |
| 2. Ir. Ali Fatoni, M.T. | Co-Advisor |  |
| 3. Yurid Eka Nugraha, ST, M.Eng, PhD | Examiner |  |
| 4. Dr.Ir. Ari Santoso, DEA. | Examiner |  |
| 5. Dr. Trihastuti Agustinah, ST., MT. | Examiner |  |

SURABAYA

July, 2023

Halaman ini sengaja dikosongkan

PERNYATAAN ORISINALITAS

Yang bertanda tangan di bawah ini:

Nama mahasiswa / NRP : Bambang Permadi Widysamara / 07111940000222

Program studi : Teknik Elektro

Dosen Pembimbing / NIP : 1. Eka Iskandar, ST., MT. / 198005282008121001
2. Ir. Ali Fatoni, M.T. / 196206031989031002

dengan ini menyatakan bahwa Tugas Akhir dengan judul "**Implementasi Kontrol Kecepatan Motor Induksi 3 Fasa menggunakan VFD dengan Kontroler PI-GA**" adalah hasil karya sendiri, bersifat orisinal, dan ditulis dengan mengikuti kaidah penulisan ilmiah.

Bilamana di kemudian hari ditemukan ketidaksesuaian dengan pernyataan ini, maka saya bersedia menerima sanksi sesuai dengan ketentuan yang berlaku di Institut Teknologi Sepuluh Nopember.

Surabaya, Juni 2023

Mengetahui
Dosen Pembimbing 1



Eka Iskandar, ST., MT.
198005282008121001

Mengetahui
Dosen Pembimbing 2



Ir. Ali Fatoni, M.T.
196206031989031002

Mahasiswa

Bambang Permadi Widysamara
0711194000222

Halaman ini sengaja dikosongkan

ABSTRAK

IMPLEMENTASI KONTROL KECEPATAN MOTOR INDUKSI 3 FASA MENGGUNAKAN VFD DENGAN KONTROLER PI-GA

Nama Mahasiswa / NRP : **Bambang Permadi Widyasmara / 07111940000222**
Departemen : **Teknik Elektro FTEIC - ITS**
Dosen Pembimbing : **Eka Iskandar, ST., MT.**

Abstrak

Motor listrik merupakan motor yang merubah energi listrik menjadi energi mekanik, salah satu contoh motor listrik adalah motor induksi 3 fasa. Motor ini bekerja dengan menggunakan induksi elektromagnet yang ada di dalam lilitan stator ke lilitan rotornya. Lalu motor ini membuat sebuah gaya elektromotif (emf) atau biasa disebut sebagai tegangan induksi. Motor induksi biasa digunakan di bidang industri karena terkenal dengan kuat dan kendali yang baik. Untuk mengatur kecepatan motor ini digunakan inverter sejenis Variable Frequency Drives. VFD merupakan perangkat elektronik yang digunakan untuk memvariasi frekuensi tegangan AC untuk mengatur kecepatan motor AC. VFD yang digunakan pada penelitian ini adalah Altivar V610. Selain VFD digunakan kontroler PI yang dapat mengendalikan wahtrus steadystate serta sistem tanpa overshoot. Untuk mengatur angka PI digunakan algoritma genetik sebagai optimisasi kontroler. Didapatkan bahwa menggunakan algoritma genetic dengan operasi genetic yang besar pada kontroler PI dapat meredam overshoot pada sistem dengan memberikan output steady state sesuai dengan setpoint pada sistem. Dengan menggunakan PI – GA menghasilkan sistem dengan angka kecepatan RMSE rata – rata ± 65 RPM. Selain implementasi, digunakan simulasi pada Matlab untuk mengatur angka PI – GA ini.

Kata kunci: *Motor Induksi 3 Fasa, Altivar V610, PI controller, Genetic Algorithm, MATLAB*

Halaman ini sengaja dikosongkan

ABSTRACT

IMPLEMENTATION OF SPEED CONTROL 3 PHASE INDUCTION MOTOR USING VFD WITH PI-GA CONTROLLER

Student Name / NRP : **Bambang Permadi Widyasmara / 07111940000222**
Department : **Electrical Engineering FTEIC - ITS**
Advisor : **Eka Iskandar, ST., MT.**

Abstract

An electric motor is a motor that converts electrical energy into mechanical energy, one example of an electric motor is a 3-phase induction motor. This motor works by using electromagnetic induction in the stator winding to the rotor winding. Then this motor creates an electromotive force (emf) or commonly referred to as an induced voltage. Induction motors are commonly used in industry because they are known for their strength and good control. To regulate the speed of this motor, an inverter such as Variable Frequency Drives is used. VFD is an electronic device that is used to vary the frequency of the AC voltage to regulate the speed of the AC motor. The VFD used in this study is Altivar V610. In addition to the VFD, a PI controller is used which can control the steady state current and make a system without overshoot. To set the PI number, a genetic algorithm is used as controller optimization. It was found that using a genetic algorithm with large genetic operations on the PI controller can reduce overshoot in the system and also providing a steady state output according to the set point in the system. Using PI - GA results in a system with an average RMSE speed figure of ± 65 RPM. In addition to the implementation, simulations are used in Matlab to set these PI – GA numbers.

Keywords: *3-Phase Induction Motor, Altivar V610, PI Controller, Genetic Algorithm, MATLAB*

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Allah SWT atas ridhonya saya dapat menyelesaikan penyusunan skripsi ini. Adapun judul Tugas Akhir yang saya ajukan adalah “Implementasi Kontrol Kecepatan Motor Induksi 3 Fasa Menggunakan VFD dengan Kontroler PI-GA”. Skripsi ini diajukan untuk memenuhi syarat kelulusan mata kuliah Tugas Akhir di Fakultas Teknologi Elektro dan Informatika Cerdas Institut Teknologi Sepuluh Nopember. Tidak dapat disangkal bahwa butuh usaha yang keras dalam penyelesaian penggerjaan skripsi ini. Namun, karya ini tidak akan selesai tanpa orang-orang tercinta di sekeliling penulis yang mendukung dan membantu. Terima kasih penulis sampaikan kepada :

1. Keluarga penulis, terutama orang tua dan juga kakak adik saya yang telah memberikan dukungan pada proses penggerjaan TA.
2. Bapak Eka Iskandar, ST., MT. dan Bapak Ir. Ali Fatoni, M.T. selaku Dosen Pembimbing pertama dan kedua pada Tugas Akhir karena telah membimbing dan mendukung kesuksesan dalam menyelesaikan Tugas Akhir.
3. Bapak dan Ibu Dosen bidang studi Sistem Pengaturan yang telah memberikan materi perkuliahan yang berguna untuk penulis Menyusun tugas akhir ini
4. Bapak dan Ibu Dosen Departemen Teknik Elektro yang telah memberikan materi perkuliahan selama kurang lebih 8 semester, serta memberikan saran di dalam ujian Tugas Akhir.
5. Teman-teman dan juga sahabat saya, grup perkumpulan badminton Hidup Sehat, grup teman – teman Solid dari SMA yang memberikan bantuan informasi maupun dukungan untuk menyelesaikan tugas akhir ini.
6. Semua pihak yang telah membantu dan tidak dapat disebutkan satu persatu.

Semoga segala kebaikan dan pertolongan semuanya mendapat berkah dari Allah Swt. Dan akhirnya saya menyadari bahwa skripsi ini masih jauh dari kata sempurna, karena keterbatasan ilmu yang saya miliki. Saya harapkan dengan terbentuknya Tugas Akhir ini dapat memberikan ilmu tambahan kepada seluruh pembaca dan dapat dikembangkan menjadi penelitian yang lebih lanjut.

Surabaya, Juli 2023

Penulis

Halaman ini sengaja dikosongkan

DAFTAR ISI

LEMBAR PENGESAHAN	i
APPROVAL SHEET	iii
PERNYATAAN ORISINALITAS	v
ABSTRAK	vii
ABSTRACT	ix
KATA PENGANTAR	xi
DAFTAR ISI	xiii
DAFTAR GAMBAR	xvii
DAFTAR TABEL	xix
DAFTAR SIMBOL	xxi
BAB 1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	1
1.3 Batasan Masalah	2
1.4 Tujuan	2
1.5 Manfaat	2
BAB 2 TINJAUAN PUSTAKA	3
2.1 Hasil Penelitian Terdahulu	3
2.2 Dasar Teori	3
2.2.1 Motor Induksi	3
2.2.2 Rangkaian Ekuivalen Motor Induksi	4
2.2.3 Struktur Motor Induksi	6
2.2.4 <i>System Identification</i>	7
2.2.5 <i>Variable Frequency Drives</i>	9
2.2.6 Inverter PWM	10
2.2.7 <i>Digital Aquisition System</i>	10
2.2.8 <i>PI controller</i>	11
2.2.9 <i>Genetic Algorithm</i>	12
2.2.10 Operasi Genetik	12
2.2.11 Fitness Function	14
2.2.12 <i>Root Mean Square Error (RMSE)</i>	15
BAB 3 METODOLOGI	17

3.1	Metode yang digunakan	17
3.2	Bahan dan peralatan yang digunakan	17
3.2.1	<i>Hardware</i>	17
3.2.2	<i>Software</i>	22
3.3	Urutan pelaksanaan penelitian	22
3.3.1	Diagram Blok Penelitian	22
3.3.2	Rangkaian Diagram Skema <i>Open – loop</i> Motor Induksi Tiga Fasa dengan VFD	23
3.3.3	Rangkaian Diagram Skema <i>Closed – loop</i> Motor Induksi Tiga Fasa dengan Kontroler PI dan VFD	23
3.3.4	Mendapatkan Data Tegangan dengan Advantech melalui Matlab	24
3.3.5	Mendapatkan nilai RPM dari Motor Induksi dengan menggunakan Laser Tachometer	25
3.3.6	Pengumpulan dan Perhitungan Data Sistem (Tegangan Rata – Rata, RPM, dan nilai konversi)	26
3.3.7	Pendekatan Fungsi Transfer dengan metode Zieger Nichols	27
3.3.8	Perhitungan RMSE Raw data dengan Fungsi transfer	29
3.3.9	Pengambilan data RPM tanpa kontroler pada Sistem Motor Induksi	31
3.3.10	Pembuatan <i>Genetic Algorithm</i> untuk <i>PI Controller</i>	32
3.3.11	Operasi Genetik dalam PI – GA	33
3.3.12	Implementasi penggunaan Angka PI-GA dalam Motor Induksi	37
BAB 4	Hasil dan Pembahasan	39
4.1	Hasil Penelitian	39
4.1.1	Perbandingan PID – ZN dan PID – GA terbaik	39
4.1.2	Perbandingan <i>Crossover</i> dan Mutasi	41
4.1.3	Perbandingan Presentase Mutasi	43
4.1.4	Perbandingan Populasi	45
4.1.5	Perbandingan Seluruh Hasil	47
4.1.6	Peggunaan beban pada simulasi	48
4.2	Analisa Data	48
4.2.1	Perbandingan PID – ZN dan PID – GA	48
4.2.2	Perbandingan Kali Crossover dan Mutation	49
4.2.3	Perbandingan Presentase Mutasi	50
4.2.4	Perbandingan Populasi	51
4.2.5	Perbandingan Hasil Terbaik	51

4.2.6	Perbandingan Hasil Terbaik dengan Menggunakan Beban	52
BAB 5	Kesimpulan dan Saran	53
5.1	Kesimpulan	53
5.2	Saran	53
DAFTAR PUSTAKA		55
LAMPIRAN		57
BIODATA PENULIS		67

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

Gambar 2.1 Rangkaian ekivalen motor induksi (Mansour, 2020)	4
Gambar 2.2 Struktur motor induksi (Alnasir & Almarhoon A H, 2012)	7
Gambar 2.3 Diagram proses identifikasi sistem (Jakoubek, n.d.).....	8
Gambar 2.4 Menentukan titik delay L dan juga time constant T untuk ZN (Patel, 2020)....	8
Gambar 2.5 VFD (Smith, 2022)	9
Gambar 2.6 Rangkaian VFD (Smith, 2022).....	10
Gambar 2.7 PWM inverter motor induksi 3 fasa (Alnasir & Almarhoon A H, 2012)....	10
Gambar 2.8 Block Diagram Data Acquisition System	11
Gambar 2.9 Diagram blok PI control system (Wang, 2020).....	11
Gambar 2.10 Flowchart Genetic Algorithm (Ahmmmed et al., 2020)	12
Gambar 2.11 Roulette wheel selection (Haldurai et al., 2016).....	13
Gambar 2.12 Tiga tipe crossover yang berbeda (Haldurai et al., 2016)	14
Gambar 2.13 Mutation pada algoritma genetik (Liu, 2014).....	14
Gambar 3.1 Sistem Motor AC dengan Motor DC	18
Gambar 3.2 Altivar 610 VFD	20
Gambar 3.3 Advantech USB 4761	20
Gambar 3.4 Laser Tachometer	21
Gambar 3.5 Sistem motor induksi saat terpasang secara lengkap.....	24
Gambar 3.6 Tampak hasil output sistem open – loop di figure Matlab	25
Gambar 3.7 Tampak hasil output sistem open – loop dengan Txt.	25
Gambar 3.8 Pengukuran RPM dengan Laser Tacho pada sistem Motor Induksi	26
Gambar 3.9 Menentukan garis untuk ZN method dengan Tegangan 5 Volt.....	28
Gambar 3.10 Perbandingan sistem ZN method (Biru) dan Real Data (Oranye).....	30
Gambar 3.11 Tanpa kontroler dalam 500 Rpm	31
Gambar 3.12 Tanpa kontroler dalam 750 Rpm	31
Gambar 3.13 Tanpa kontroler dalam 1000 Rpm	32
Gambar 3.14 Diagram Alur Genetic Algorithm.....	32
Gambar 3.15 Hasil 500 generasi dengan 20 populasi	34
Gambar 3.16 Pengaturan PI pada Altivar V610	37
Gambar 4.1 Hasil fitness ZN – PI – GA.....	39
Gambar 4.2 Hasil Simulasi Respons 750 Rpm PI-ZN dan ZN-PI-GA	39
Gambar 4.3 Respon perbandingan ZN – PI dan ZN – PI – GA dalam 500 rpm.....	40
Gambar 4.4 Respon perbandingan ZN – PI dan ZN – PI – GA dalam 750 rpm.....	40
Gambar 4.5 Respon perbandingan ZN – PI dan ZN – PI – GA dalam 1000 rpm.....	41
Gambar 4.6 Hasil fitness Value Crossover/Mutasi 1 dan 2 kali	41
Gambar 4.7 Hasil Simulasi Respons 750 Rpm pada Crossover/Mutasi 1 dan 2 kali.....	41
Gambar 4.8 Respon perbandingan Crossover/Mutasi 1 dan 2 kali dalam 500 rpm	42
Gambar 4.9 Respon perbandingan Crossover/Mutasi 1 dan 2 kali dalam 750 rpm	42
Gambar 4.10 Respon perbandingan Crossover/Mutasi 1 dan 2 kali dalam 1000 rpm	43
Gambar 4.11 Hasil fitness Value Mutrate 0.1 dan 0.2	43
Gambar 4.12 Hasil Simulasi Respons 750 Rpm pada Mutrate 0.1 dan 0.2	43
Gambar 4.13 Respon perbandingan Mutrate 0.1 dan 0.2 dalam 500 rpm.....	44
Gambar 4.14 Respon perbandingan Mutrate 0.1 dan 0.2 dalam 750 rpm.....	44

Gambar 4.15 Respon perbandingan Mutrate 0.1 dan 0.2 dalam 1000 rpm	45
Gambar 4.16 Hasil fitness Value 10 dan 20 Populasi.....	45
Gambar 4.17 Hasil fitness Value 10 dan 20 Populasi.....	45
Gambar 4.18 Respon perbandingan 10 dan 20 populasi dalam 500 rpm	46
Gambar 4.19 Respon perbandingan 10 dan 20 populasi dalam 750 rpm	46
Gambar 4.20 Respon perbandingan populasi 10 dan 20 populasi 1000 rpm.....	47
Gambar 4.21 Respon hasil dengan 500 rpm	47
Gambar 4.22 Respon hasil dengan 750 rpm	47
Gambar 4.23 Respon hasil dengan 1000 rpm	48
Gambar 4.24 Respon hasil dengan beban pada 1000 rpm	48

DAFTAR TABEL

Tabel 3.1 Spesifikasi Motor Induksi	18
Tabel 3.2 Spesifikasi Motor DC tercoupling dengan AC	18
Tabel 3.3 Spesifikasi PC yang digunakan.....	19
Tabel 3.4 Spesifikasi VFD	19
Tabel 3.5 Spesifikasi Advantech USB 4761	20
Tabel 3.6 Spesifikasi Laser Tacho	21
Tabel 3.7 Software yang digunakan.....	22
Tabel 3.8 Jadwal Pelaksanaan Penelitian.....	22
Tabel 3.9 Pengambilan data tegangan dan RPM sistem Open – loop.....	27
Tabel 3.10 Perbandingan data RMSE hasil Transfer Function dan open – loop sistem ..	30
Tabel 3.11 Angka PID menggunakan ZN method.....	33
Tabel 3.12 Angka PID yang digunakan PI-GA.....	33
Tabel 3.13 Populasi 10 dalam genetic algorithm	34
Tabel 3.14 Populasi 20 dalam genetic algorithm	35
Tabel 4.1 Hasil Kp dan Ki pada ZN – PI dan ZN – PI – GA.....	40
Tabel 4.2 Hasil Integral Error pada ZN – PI dan ZN – PI – GA.....	40
Tabel 4.3 Hasil Kp dan Ki pada <i>Crossover/Mutation</i> 1 kali dan 2 kali	42
Tabel 4.4 Hasil Integral Error pada Crossover/Mutation 1 kali dan 2 kali	42
Tabel 4.5 Hasil Kp dan Ki pada Mutrate 0.1 dan 0.2.....	44
Tabel 4.6 Hasil <i>Integral Error</i> pada Mutrate 0.1 dan 0.2	44
Tabel 4.7 Hasil Kp dan Ki pada 10 dan 20 Populasi.....	46
Tabel 4.8 Hasil <i>Integral Error</i> pada 10 dan 20 Populasi	46

Halaman ini sengaja dikosongkan

DAFTAR SIMBOL

n_s = Nilai rotasi Stator

n_r = Nilai rotasi Rotor

f = Frekuensi

P = Kutub Motor

E_{rotor} = Tegangan Ekivalen Rotor

I_{rotor} = Arus Ekivalen Rotor

Z_{Rotor} = Impedansi Rotor

R = Hambatan

X = Reaktansi Bocor pada Rotor

s = Slip

P_g = Daya

τ_d = Torsi motor

ω_s = Kecepatan sudut

$e(t)$ = Error pada t

m = Total data

X_i = Jumlah data Prediksi X

Y_i = Jumlah data Prediksi Y

L = Waktu Delay

T = Waktu Rise Time

K = Gain

e^{-Ls} = Delay sistem

Halaman ini sengaja dikosongkan

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Motor listrik merupakan salah satu komponen yang sering digunakan pada pengaplikasian penggerak roda ataupun penggunaan lainnya. Motor listrik didefinisikan sebagai mesin yang mengubah energi listrik menjadi energi mekanik. Biasanya, motor digerakkan oleh interaksi antara medan magnet dan arus listrik dalam gulungan kumparan. Karena itu, ia menghasilkan gaya dalam bentuk torsi yang diterapkan pada poros motor (Hughes, 2006). Terdapat beberapa tipe motor listrik contohnya ada motor DC, motor induksi, motor PM, motor lainnya (Zarma et al., 2019). Motor induksi merupakan motor yang sinkron yang berarti shaft dari rotor bekerja secara sinkron mengikuti suplai arus dengan frekuensi yang ada pada motor itu sendiri dan rotor ini ditenagai oleh induksi elektromagnetiknya. Motor induksi seringkali digunakan sebagai penggerak kendaraan listrik. Karena motor induksi termasuk jenis motor yang kuat serta mempunyai kemampuan kendali yang baik dan murah (Hughes, 2006).

Terdapat banyak jenis optimisasi pada sistem kontrol. Salah satunya merupakan *Genetic Algorithm*. Ini merupakan salah satu metode optimisasi dengan menggunakan seleksi alam. Metode ini juga dikembangkan dari algoritma evolusioner yang merupakan algoritma berdasarkan populasi generik dan juga disebut sebagai optimisasi metareustik (Savio & Chakraborty, 2019). Seleksi stokastik dari individu yang lebih cocok dilakukan dari populasi saat ini, dan memodifikasi genom setiap individu dan digabungkan untuk membentuk generasi baru. Solusi kandidat yang baru dihasilkan digunakan dalam generasi berikutnya yang merupakan iterasi dari algoritma genetika. Umumnya, algoritma berhenti jika jumlah maksimum (diatur oleh pengguna) dari generasi yang ditentukan sebelumnya telah tercapai, atau tingkat kebugaran yang dianggap optimal atau hampir optimal (memuaskan) telah tercapai untuk populasi (Haldurai et al., 2016).

Dalam sektor industri sendiri, motor induksi sangat sering digunakan maka dari itu diperlukannya optimisasi yang baik pada motor induksi sendiri. Optimisasi yang bisa digunakan adalah dengan menambahkan *Genetic Algorithm* pada *plant* motor induksi. GA telah banyak dibuktikan mempunyai kemampuan yang baik dalam optimisasi berbagai keadaan serta parameter (Samakwong & Assawinchaichote, 2016). Selain penggunaan dari *genetic algorithm* sendiri juga digunakan *Variable Frequency Drives (VFD)* pada kontrol kecepatan di motor induksi ini. Dengan menggunakan ini dapat memberikan kendali secara langsung dengan fluks dan torsinya seperti pada pengaturan kecepatan pada motor DC. Berkembangnya perangkat terhubung di berbagai sektor seperti sektor komersial, industri dan perumahan ditambah dengan meningkatnya permintaan untuk Penggerak Frekuensi Variabel (VFD) karena keunggulannya seperti kecepatan yang dapat disesuaikan, kontrol torsi dinamis, dan penghematan energi. Dengan menggunakan VFD dapat mempermudah pengontrolan dan pengaturan kecepatan pada motor induksi 3 fasa (Smith, 2022).

1.2 Rumusan Masalah

Diharapkan kontrol kecepatan pada motor induksi dengan penggabungan *Variable Frequency Drives* dengan optimisasi *Genetic algorithm* pada kontroler PI dapat membuat *output* kecepatan motor induksi yang konstan dengan angka eror yang minimal.

1.3 Batasan Masalah

Dari penelitian ini terdapat beberapa batasan masalah yang ditemukan sebagai berikut:

1. Digunakan beberapa simulasi untuk penelitian ini. Aplikasi yang digunakan merupakan MATLAB.
2. Penggunaan Motor Induksi 3 Fasa dengan parameter yang kurang lengkap
3. Diperlukan genetik algoritma yang tepat untuk pengaturan motor induksi
4. Akan dijalankan dengan batas 100 Generasi agar tidak memakan waktu lama untuk simulasi.
5. Tuning PI-GA dilakukan secara *offline* untuk mengurangi risiko rusak saat menggunakan implementasi langsung di motor induksi.

1.4 Tujuan

Tujuan dari penelitian tugas akhir ini adalah mengatur kecepatan motor induksi menggunakan *Variable Frequency Drives* dengan kontroler PI – GA dan mencari apakah populasi lebih banyak mengeluarkan *output* lebih bagus dan berapa *rate* mutasi untuk mencapai *setpoint* yang diinginkan tanpa *overshoot*.

1.5 Manfaat

Dari penelitian ini diharapkan dapat memberikan manfaat terhadap pembaca dan juga peneliti lainnya tentang penggunaan *genetic algorithm* untuk kontrol kecepatan serta VFD.

BAB 2 TINJAUAN PUSTAKA

2.1 Hasil Penelitian Terdahulu

Sebelumnya telah ada banyak penelitian terkait kontrol kecepatan pada motor yang digunakan pada kendaraan listrik otonom. Banyak penelitian yang melihat bahwa motor DC merupakan standar motor listrik yang digunakan oleh kendaraan listrik pada jaman dahulu. Ditemukan bahwa saat motor DC dibandingkan dengan motor AC, bahwa hasil dari kegunaan motor AC lebih baik daripada DC (Zarma et al., 2019). Dari penelitian yang telah dilakukan oleh Mounir dan rekannya, didapatkan bahwa motor induksi mempunyai tingkat reliabilitas dan juga kontrol yang sangat tinggi dibandingkan dengan motor DC. Lalu dalam penelitiannya ia melihat bahwa motor induksi memenuhi beberapa kriteria yang cocok untuk dijadikan penggerak kendaraan listrik, salah satunya adalah harganya yang murah dan kecanggihannya yang tinggi (Zeraoulia et al., 2006). Lalu biasanya kontrol kecepatan pada motor induksi menggunakan PID kontroler saja dan tidak banyak yang mengimplementasikannya dengan menggunakan optimisasi algoritma yang lain. Dari penelitian yang telah dilakukan oleh Bousserhane dan rekannya, didapatkan bahwa menggunakan PID dibuktikan bahwa mengeluarkan waktu respons yang lama dan kurang stabil dengan *output* acuannya (Bousserhane et al., 2006). Dalam sebuah penelitian lain yang dilakukan oleh Narayan dan Pratibha (2017), mereka meneliti penggunaan *genetic algorithm* (GA) dan membandingkan dengan menggunakan PID biasa, *input* yang ditawarkan pada motor ini lebih baik hasilnya saat digunakan optimisasi *genetic algorithm* (Pandey & Tiwari, 2017). Maka dari itu diharapkan dengan penggunaan optimisasi GA pada kontrol kecepatan motor induksi ini dapat mengeluarkan *output* yang lebih baik dan stabil agar dapat digunakan di kendaraan listrik otonom.

2.2 Dasar Teori

2.2.1 Motor Induksi

Kontrol motor induksi di bidang penggerak motor listrik telah menarik banyak peneliti dari waktu ke waktu. Untuk memenuhi standar kinerja tinggi yang ditetapkan oleh penggerak dc, penggerak AC yang kecepatan variabel menggunakan motor induksi dan metode kontrol berorientasi lapangan telah dikembangkan dalam beberapa tahun terakhir. Memisahkan istilah antara fluks dan torsi sangat dianjurkan untuk memberikan respons dinamis yang cepat serta kinerja kondisi mapan yang baik. Dinamis tinggi kinerja motor induksi dapat dicapai dengan cara kontrol berorientasi lapangan di mana ia menyediakan yang sesuai deskripsi matematis motor induksi tiga fasa (Deepalis et al., 2013). Induksi motor bekerja dengan menggunakan induksi elektromagnet yang ada di dalam lilitan stator ke lilitan rotornya. Lalu motor ini membuat sebuah gaya elektromotif (emf) atau biasa disebut sebagai tegangan induksi. Cara kerja induksi dalam motor ini adalah adanya elektromagnet dari lilitan stator dan rotornya (Hermanu et al., 2020). Rumus dari kecepatan rotasi dari induksi motor adalah sebagai berikut:

$$n_s = \frac{120 * f}{P} \quad (2.1)$$

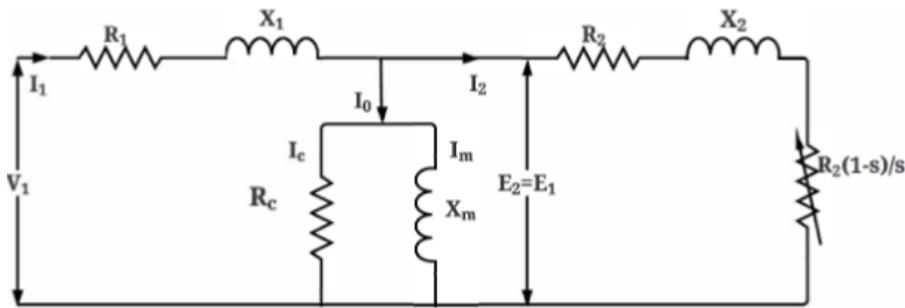
Dimana n_s , merupakan kecepatan sinkron yang ada pada motor induksi. Untuk mengatur ini dapat mengganti nilai dari frekuensi (f) itu sendiri. Lalu juga digunakan nilai dari kutub motor (P) yang ada pada motor induksi ini. Untuk perbedaan dari motor sinkron dan juga

asinkron adalah di motor asinkron mempunyai slip pada motor. Slip ini dapat ditunjukkan dengan adanya perbedaan dari rotasi stator n_s dan juga nilai rotasi pada motor rotor n_r . Berikut adalah rumus dari slip (s) :

$$S = \frac{n_s - n_r}{n_s} \quad (2.2)$$

2.2.2 Rangkaian Ekuivalen Motor Induksi

Dalam motor induksi terdapat model matematis yang digunakan untuk perhitungan dari rangkaian ekuivalennya. Rangkaian ekuivalen terdapat stator, rotor, fluks *linkage*, dan juga torsi dari motor listrik induksi tersebut. Berikut adalah gambar dari rangkaian ekuivalen dari motor induksi:



Gambar 2.1 Rangkaian ekuivalen motor induksi (Mansour, 2020)

Untuk belitan rotor akan sama lilitannya dengan yang ada pada kutub dan phasa stator motor induksi. Jumlah yang efektif untuk lilitan pada stator adalah sebanyak dari jumlah pada lilitan rotor itu sendiri. Kecepatan dan juga fluks pada motor induksi angkanya sama dan hubungannya dengan tegangan Ekuivalen rotor (E_{rotor}) berimbang pada rotor sebenarnya dengan tegangan E_2S sebagai berikut persamaannya:

$$E_2S = a E_{rotor} \quad (2.3)$$

Lalu nantinya jika rotor diganti magnetis, maka lilitan dari masing – masing lilitan arus harus sama dan hubungan arus rotor (Irotor) dengan arus I_{2s} di rotor ekuivalen adalah sebagai berikut:

$$I_{2s} = \frac{I_{Rotor}}{a} \quad (2.4)$$

Dari sini dapat mengakibatkan hubungan dari impedansi (Z) dengan frekuensi Z_{2s} yang merupakan Impedansi bocor rotor frekuensi oleh slip yang diukur dalam Ohm. menjadi bocor dari rotor ekuivalen dan impedansi frekuensi slip Z_{rotor} adalah sebagai berikut:

$$Z_{2s} = \frac{E_2S}{I_{2s}} = a^2 Z_{Rotor} (\text{ohm}) \quad (2.5)$$

Karena rotor yang dihubungkan secara singkat, maka hubungan dari fasor dengan gaya gerak listrik frekuensi slip E_2S dibangkitkan di fasa yang dipotok dari rotor dan juga arus I_{2s} pada fasa ini. Dengan R_2 merupakan tahanan atau hambatan rotor (ohm) dan juga X_{2s} sebagai reaktansi bocor oleh frekuensi slip (ohm). Reaktansi dari persamaan di bawah ini dapat didapatkan apabila frekuensi rotor dan slip sebanding. Lalu X_{2s} didefinisikan sebagai nilai dari

reaktansi bocor pada rotor dengan patokan di frekuensi stator. Persamaan ini dapat ditunjukkan sebagai berikut:

$$Z_2s = R_2 + jsX_2 \quad (2.6)$$

Lalu untuk kecepatan relatif dari gelombang fluks terhadap rotor adalah s kali dari kecepatan stator sebagai berikut:

$$E_2s = sE_2 \quad (2.7)$$

Untuk persamaan arus rotor per fasa didapatkan sebagai berikut:

$$I_2s = \frac{E_2s}{Z_2s} = \frac{sE_2}{R_2 + jsX_2} \quad (2.8)$$

Jika persamaan di atas dibagi dengan slip s maka didapatkan persamaan berikut:

$$I_2s = \frac{E_2}{\frac{R_2}{s} + jX_2} \quad (2.9)$$

Jika kita mengasumsikan bahwa V_1 adalah sama dengan E_1 dan E_2' pada rangkaian di **Gambar 2.1** di atas. Maka komponen yang berhubungan dengan slip adalah resistansinya. Diasumsikan angka lainnya juga konstan, didapatkan persamaan dengan segala slip s sebagai berikut:

Impedansi dalam AA':

$$Z'_{AA} = \left(R_1 + \frac{R'_2}{s} \right) + j(X_1 + X'_2) \quad (2.10)$$

$$I'_2 = \frac{V_1}{Z'_{AA}} = \frac{V_1}{\left(R_1 + \frac{R'_2}{s} \right) + j(X_1 + X'_2)} \quad (2.11)$$

Dari persamaan 10 dan 11 dapat dijadikan persamaan sebagai berikut:

$$I'_2 = |I''_2| = \frac{V_1}{\sqrt{\left(R_1 + \frac{R'_2}{s} \right)^2 + j(X_1 + X'_2)^2}} \quad (2.12)$$

Bisa didapatkan,

$$I'_2 = I'_2 \cos \theta_2 - jI'_2 \sin \theta_2 \quad (2.13)$$

Lalu,

$$\tan \theta_2 = \frac{X_1 + X'_2}{R_1 + \frac{R'_2}{s}} \quad (2.14)$$

Dan,

$$\cos \theta_2 = \frac{R_1 + \frac{R'_2}{s}}{|Z'_{AA}|} \quad (2.15)$$

Pada arus tanpa beban I_0 ,

$$\begin{aligned} I_0 &= I_m + I_c \\ I_0 &= \frac{V_1}{R_c} + \frac{V_1}{jX_m} \\ I_0 &= V_1 \left(\frac{1}{R_c} + j \frac{1}{X_m} \right) \end{aligned} \quad (2.16)$$

Untuk persamaan total dari arus stator adalah sebagai berikut:

$$I_1 = I'_2 + I_0 \quad (2.17)$$

Total core loss (P_{h+e}) dapat dibuat persamaan sebagai berikut:

$$P_{h+e} = 3V_1 I_0 \cos \theta_0 \quad (2.18)$$

$$\text{Input stator} = 3V_1 I_1 \cos \theta_1$$

$$\text{Input stator} = 3V_1 I'_2 \cos \theta_2 + P_{h+e}$$

$$\text{Input stator} = 3I'^2_2 \left(R_1 + \frac{R'_2}{s} \right) + P_{h+e} \quad (2.19)$$

Dari persamaan ini bisa didapatkan perhitungan dari daya air gap per fasa pada motor induksi (P_g) sebagai berikut:

$$P_g = V_1 I'_2 \cos \theta_2 = I'^2_2 \frac{R'_2}{s} = \frac{V_1^2 \left(\frac{R'_2}{s} \right)}{\left(R_1 + \frac{R'_2}{s} \right) + (X_1 + X'_2)^2} \quad (2.20)$$

Pada akhirnya dapat ditemukan persamaan dari torsi motor induksi dengan persamaan sebagai berikut:

$$\tau_d = \frac{P_g}{\omega_s}$$

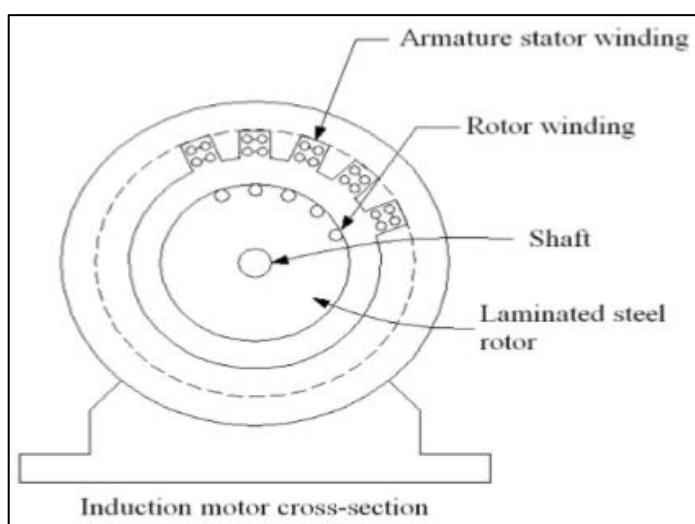
Atau,

$$\tau_d = \frac{V_1^2 \left(\frac{R'_2}{s} \right)}{\omega_s \left[\left(R_1 + \frac{R'_2}{s} \right) + (X_1 + X'_2)^2 \right]} \quad (2.21)$$

2.2.3 Struktur Motor Induksi

Dalam motor induksi terdapat beberapa komponen yang menjadi pembangun dari motor induksi. Yang pertama dan di dalam merupakan *shaft*, ini merupakan bagian tengah dari motor induksi yang terbuat dari besi atau besi karbon yang bentuknya seperti tongkat. *Shaft* berguna

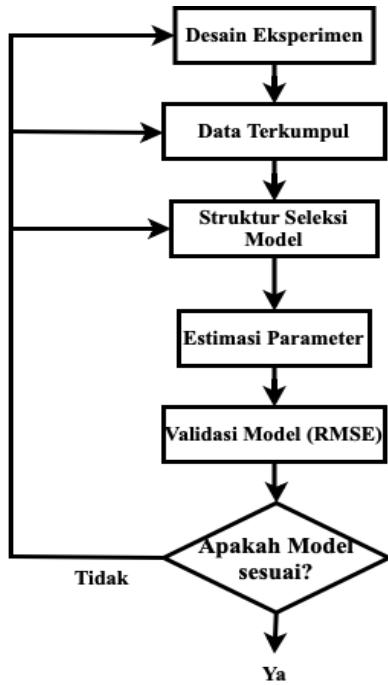
sebagai konversi energi motor untuk dijadikan penggerak dari motor sendiri. Lalu yang kedua adalah laminasi rotor, ini berguna untuk mengurangi energi yang dapat hilang dari penggerakan rotor dari motor induksi. Setelah itu ada juga rotor *winding* (lilitan rotor), ini biasanya ada pada tipe *squirrel-cage* di mana konduktor padat dalam slot *di-short* menjadi satu di setiap ujung besi rotor dengan mengonduski tiap ujung cincin. Rotor *winding* berguna sebagai penggerak dari elektromagnet yang ada pada motor listrik itu dan nantinya berinteraksi dengan lilitan dan menghasilkan medan magnet yang membuat torsi di sekeliling aksis rotor itu sendiri. Yang terakhir merupakan *armature stator winding* (lilitan stator pada armatur), ini adalah lilitan yang dimana menginduksi tegangan listrik pada motor induksi. Nantinya akan menghasilkan sebuah medan fluks saat arus melewati lilitan. Struktur motor induksi dapat dilihat sebagai berikut: (Alnasir & Almarhoon A H, 2012)



Gambar 2.2 Struktur motor induksi (Alnasir & Almarhoon A H, 2012)

2.2.4 System Identification

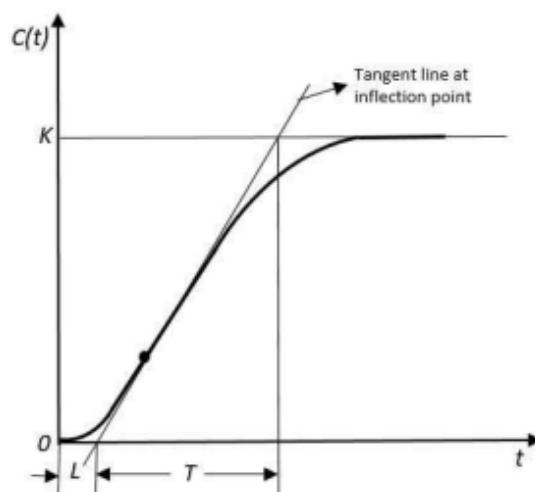
Dalam sebuah sistem, terdapat sebuah perhitungan matematis yang nantinya berguna untuk mendapatkan fungsi transfer yang digunakan untuk pengendalian sistem. Tetapi sistem yang digunakan harus mempunyai parameter yang lengkap, apabila data atau parameter yang digunakan tidak ada atau lengkap dapat digunakan teknik identifikasi sistem (Hjalmarsson, 2009). Identifikasi sistem adalah seni dan ilmu membangun model matematika dari sistem dinamis data masukan-keluaran yang diamati. Itu dapat dilihat sebagai penghubung antara aplikasi sistem dan teori kontrol model matematis dan juga model abstrak (Ljung, 2010). Dalam identifikasi sistem terdapat beberapa metode seperti metode Strejc, Smith's, Zieger-nichols dan lainnya. Berikut merupakan diagram proses identifikasi sistem:



Gambar 2.3 Diagram proses identifikasi sistem (Jakoubek, n.d.)

Dalam penelitian ini digunakan identifikasi sistem dengan metode Zeigler – Nichols. Dari Namanya sendiri metode ini ditemukan pertama kali oleh John G. Ziegler dan Nathaniel B. Nichols. Selain dapat digunakan untuk menentukan fungsi transfer sistem, identifikasi sistem ini juga dapat digunakan untuk mengatur PID dengan teknik ini. Tetapi biasanya mengatur PID dengan menggunakan metode ZN juga dapat memberikan *overshoot* dan juga osilasi yang tidak dinginkan (Åström & Hägglund, 2004). Untuk metode ini nantinya akan dilihat angka L dan juga T seperti di **gambar 2.4** ini merupakan titik dimana pertama data mencapai titik miring dan juga mencapai *steady state*. Lalu nantinya dihitung menggunakan **persamaan 2.22** untuk mendapatkan fungsi transfer yang merupakan orde 1 dengan delay e^{-Ls} . Setelah itu apabila ingin mendapatkan nilai PID – ZN maka digunakan **Tabel 2.1** untuk menghitung nilai PID-nya.

$$G_{tegangan}(s) = \frac{Ke^{-Ls}}{Ts + 1} \quad (2.22)$$



Gambar 2.4 Menentukan titik delay L dan juga time constant T untuk ZN (Patel, 2020)

Tabel 2.1 Perhitungan nilai PID dengan ZN (Patel, 2020)

Tipe Kontroler	Kp	Ti	Td
P	$\frac{T}{L}$	∞	0
PI	$0.9 \frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2 \frac{T}{L}$	$2L$	$0.5L$

Untuk mengganti angka Ti dan Td menjadi Ki dan Kd digunakan persamaan sebagai berikut.

$$Ki = \frac{Kp}{Ti} \quad \text{dan} \quad Kd = Kp * Td \quad (2.23)$$

2.2.5 Variable Frequency Drives

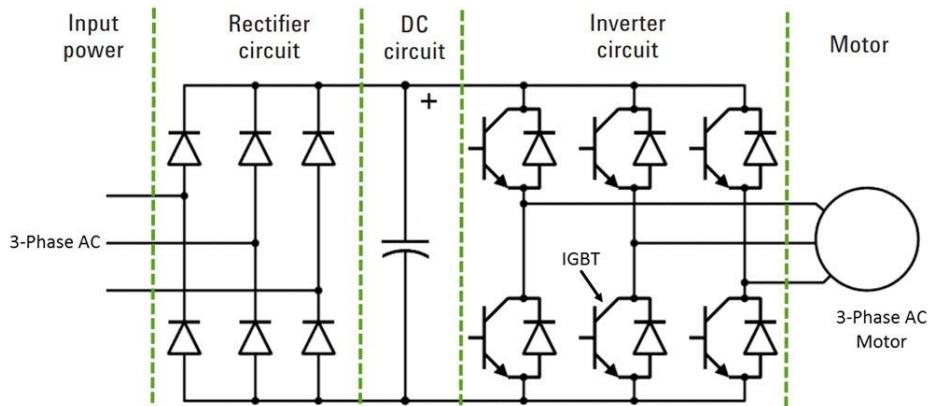
VFD pertama kali ditemukan pada tahun 1980, *Variabel Frequency Drives* (VFD) adalah jenis pengontrol motor yang menggerakkan motor listrik dengan memvariasikan frekuensi dan voltase catu dayanya. VFD juga memiliki kapasitas untuk mengontrol *ramp - up* dan *ramp - down* motor selama start atau stop. Meskipun alat ini mengontrol frekuensi dan voltase daya yang disuplai ke motor, kita sering menyebutnya sebagai kontrol kecepatan, karena hasilnya adalah penyesuaian kecepatan motor (Carrier, 2005).



Gambar 2.5 VFD (Smith, 2022)

Terdapat beberapa kegunaan untuk alat ini yaitu sebagai *inverter*, pengendalian kecepatan pada motor induksi tiga fasa, mengurangi tekanan pada motor induksi dengan menggunakan starting motor yang lebih landai. Alat ini terbagi menjadi empat bagian yaitu *rectifier*, DC Bus, *inverter*, dan juga rangkaian kontrol. Rangkaian *rectifier* berguna sebagai konversi dari tiga fasa menjadi dua fasa. Lalu setelah tergantung menjadi DC akan dimasukkan ke rangkaian DC yang terdiri dari sebuah kapasitor. Kapasitor dalam rangkaian ini berguna sebagai penyimpanan listrik sementara agar potongan arus DC akan menjadi lancar. Setelah itu akan di proses oleh rangkaian *inverter* yang terdiri dari tiga pasang IGBT(*Insulated-gate bipolar transistor*). Di dalam *inverter* ini akan menggantikan sinyal DC menjadi beberapa pulsa dan menjadikan sinyal

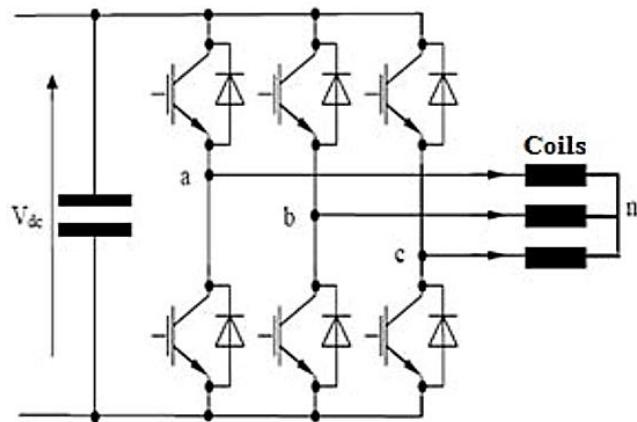
kembali ke tiga fasa (AC) Kembali. Ini ditunjukkan pada gambar rangkaian di bawah ini (Ahmad, 2021).



Gambar 2.6 Rangkaian VFD (Smith, 2022)

2.2.6 Inverter PWM

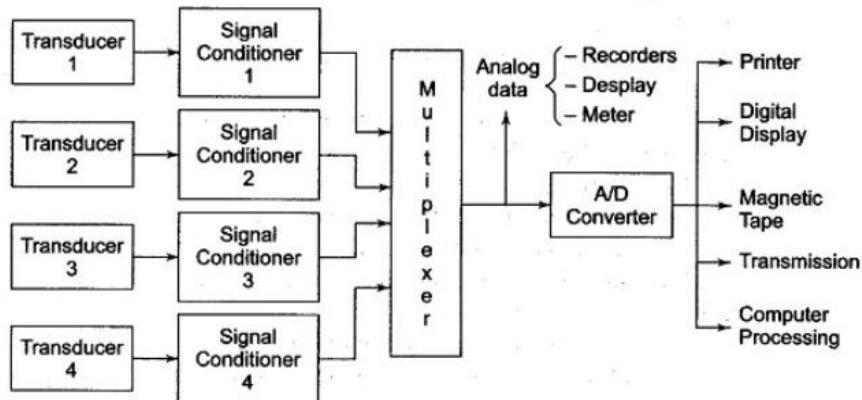
PWM adalah kepanjangan dari *Pulse Width Modulation*, ini menggunakan lebar dari pulsa yang nantinya menjaga magnitudo yang instan dan mengeluarkan *input* yang sama. PWM juga merupakan teknik yang mengganti pengaturan antara beban dan juga suplainya. Lalu untuk *inverter* sendiri merupakan teknik konversi arus DC menjadi AC 3 fasa, karena disini digunakan motor induksi. Berikut adalah rangkaian dari PWM *inverter* yang digunakan untuk motor induksi:



Gambar 2.7 PWM inverter motor induksi 3 fasa (Alnasir & Almarhoon A H, 2012)

2.2.7 Digital Aquisition System

Sistem akuisisi data (DAQ) digunakan untuk memperoleh informasi dari beberapa fisik fenomena. Proses utamanya adalah mengambil sampel sinyal yang mengubah nilai analog yang berupa sinyal listrik dari sensor, ke digital dan dimanipulasi oleh komputer. Dalam sistem data akuisisi terdapat beberapa bagian penting. Yang pertama adalah sensor yang berguna untuk mengonversi dan menerima sinyal listrik. Lalu kedua adalah *Analog to Digital Converter* (ADC), ini berguna untuk mengubah sinyal analog menjadi sinyal digital. Lalu juga ada *multiplexer* dan *amplifier* yang berguna untuk mengganti dan mengamplifikasi *input* sinyal analog dengan ADC dalam bentuk digital. Yang terakhir merupakan komputer yang berguna sebagai *interface* atau alat untuk mengatur sistem (Emilio, 2013).



Gambar 2.8 Block Diagram Data Acquisition System

2.2.8 PI controller

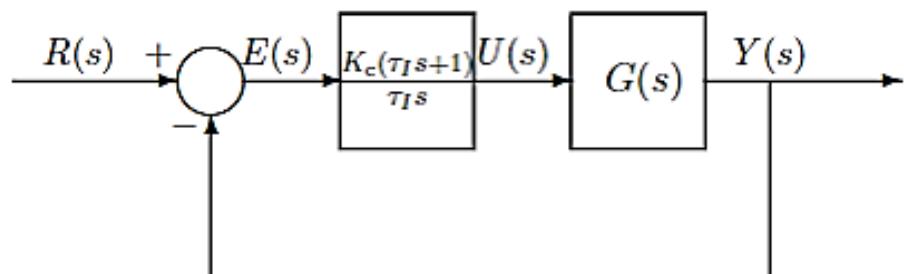
Proporsional Integral Controller juga dikenal sebagai pengontrol proporsional plus integral (PI). Ini adalah jenis pengontrol yang dibentuk dengan menggabungkan aksi kontrol proporsional dan integral. Oleh karena itu disebut sebagai pengontrol PI. Dalam pengontrol proporsional-integral, aksi kontrol dari kedua proporsional, serta pengontrol integral, digunakan(Dwyer, 2009). Kombinasi dari dua pengontrol yang berbeda ini menghasilkan pengontrol yang lebih efisien yang menghilangkan kerugian yang terkait dengan masing-masing pengontrol. Dengan adanya integral menjadikan *steady-state error* yang biasanya ada di dalam kontroler proporsional menjadi hilang (Wang, 2020). Dapat ditemukan *output* dari kontroler dalam integral sebagai berikut:

$$u(t) = K_c e(t) + \frac{K_c}{\tau I} \int_0^t e(\tau) d\tau \quad (2.24)$$

Dengan $e(t) = r(t) - y(t)$ dijadikan sinyal eror antara sinyal $r(t)$ dan *output* dari $y(t)$, K_c merupakan gain proporsional, dan τI waktu integral konstan yang bentuknya selalu positif dan hasilnya menjadi inversi proporsional dengan pengaruh dari *PI controller* itu sendiri. Semakin kecil angka τI maka akan menjadikan efek integral yang lebih kuat (Wang, 2020). Transformasi Laplace dari *PI controller* adalah berikut:

$$U(s) = K_c E(s) + \frac{K_c}{\tau_{Is}} E(s) \quad (2.25)$$

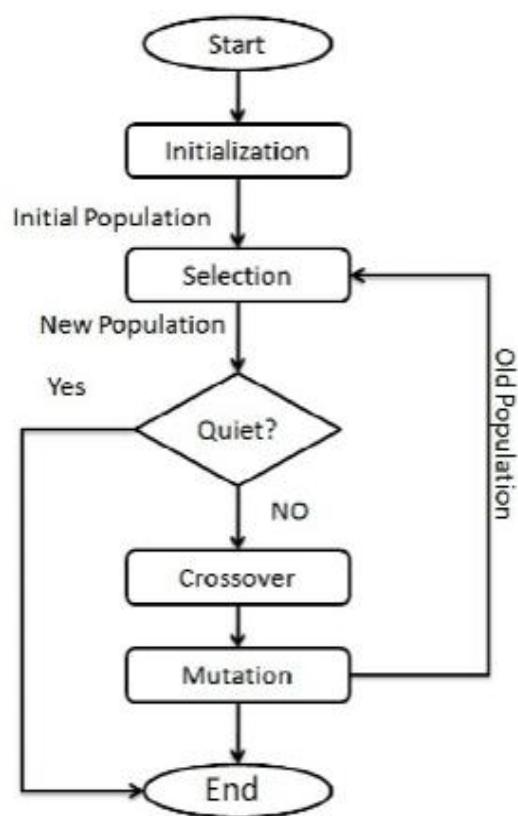
$E(s)$ Merupakan transformasi laplace dari signal error $e(t)$, dan didapatkan untuk sistem kontrol dari *PI controller* adalah sebagai berikut:



Gambar 2.9 Diagram blok *PI control system* (Wang, 2020)

2.2.9 Genetic Algorithm

Genetic algorithm (GA) merupakan metode adaptif yang biasanya digunakan untuk memecahkan pencarian dan juga masalah optimisasi dari sebuah sistem. Asal dari istilah *genetic algorithm* sendiri diambil dari proses genetik dari organisme biologis. dengan meniru proses ini, algoritme genetik dapat menemukan solusi untuk masalah dunia nyata, jika telah dikodekan dengan sesuai. Terdapat beberapa keuntungan dalam menggunakan algoritma genetik untuk optimisasi seperti dapat mengendalikan *noise* dengan mudah, bekerja sangat baik pada masalah optimisasi yang berskala besar, dan juga dapat digunakan dengan banyak jenis masalah optimisasi sistem (Sangwan & Dahiya, 2018). Di dalam GA terdapat lima komponen yang digunakan yaitu *Initial Population*, *Fitness function*, *Selection*, *Crossover*, dan *Mutation*. *Initial population* adalah banyaknya populasi pertama saat sistem dijalankan. Lalu untuk *fitness function* adalah yang bertugas mengevaluasi seberapa dekat solusi yang didapatkan sesuai dengan yang telah diinginkan secara optimum. *Selection* merupakan pemilihan seleksi dari genetika itu sendiri. *Crossover* adalah suatu operator rekombinasi yang bertujuan untuk memperoleh individu yang lebih baik. *Mutation* atau mutasi sendiri adalah operator genetik yang digunakan untuk mempertahankan perbedaan gen dari satu generasi populasi terhadap algoritma kromosom berikutnya (Ahmmmed et al., 2020).



Gambar 2.10 Flowchart Genetic Algorithm (Ahmmmed et al., 2020)

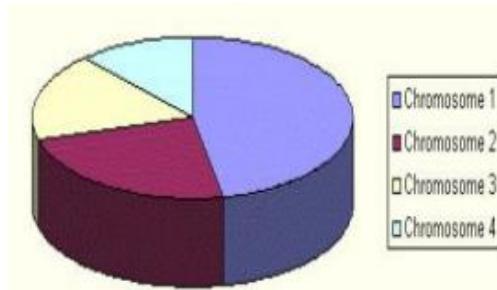
2.2.10 Operasi Genetik

Di dalam metode algoritma genetik terdapat beberapa operasi genetik yang digunakan seperti pada **Gambar 2.10** diatas. Terdapat tiga jenis operasi genetik yaitu operasi *selection*,

crossover, dan juga *mutation* (Haldurai et al., 2016). Berikut adalah penjelasan terkait ketiga operasi genetik:

A. Selection

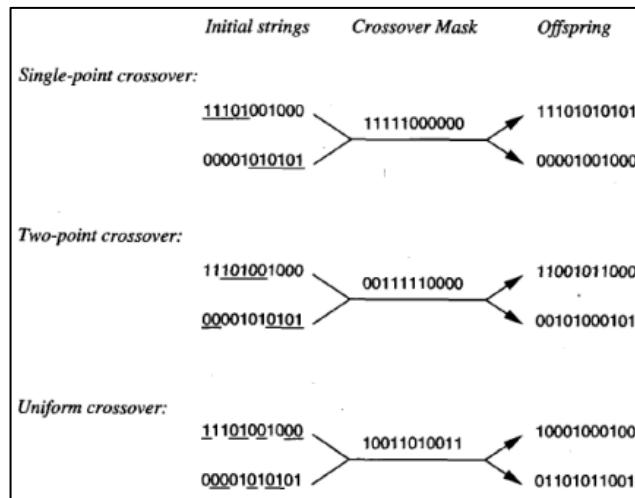
Dalam jenis operasi ini terdapat pemilihan untuk berbagai individual yang akan dijadikan sebagai orang tua pada setiap populasi. Nantinya kedua individual yang menjadi orang tua ini akan menghasilkan sebuah turunan. Lalu terdapat *fitness value* yang menjadikan kriteria seleksi mana individual yang dihasilkan adalah yang terbaik. Dalam pemilihan hasil kromosom terbaik terdapat beberapa seleksi yaitu *Roulette wheel selection*. Di dalam tipe seleksi ini orang tua dalam data akan dipilih sesuai dengan *fitness*-nya, semakin baik kromosom yang ada maka akan besar persentasenya untuk dipilih. Selanjutnya adalah *rank selection*, dari kata – katanya sendiri ini berdasarkan ranking. Jadi seluruh populasi akan disortir oleh *fitness value* dan pada akhirnya akan mendapatkan *fitness* untuk tiap kromosom sesuai dengan rankingnya. Tetapi metode ini dapat menyebabkan konvergensi yang lebih lambat, karena kromosom terbaik tidak jauh berbeda dari yang lain. Lalu juga ada tipe seleksi Elitis, di seleksi ini akan disalin kromosom terbaik dan bisa juga beberapa kromosom ke populasi baru lalu lainnya dilakukan dengan cara klasik. Elitis dapat meningkatkan kinerja GA dengan sangat cepat, karena mencegah hilangnya solusi terbaik yang ditemukan. Tidak hanya tiga metode seleksi itu saja tetapi terdapat beberapa lainnya yang bisa digunakan (Haldurai et al., 2016).



Gambar 2.11 *Roulette wheel selection* (Haldurai et al., 2016)

B. Crossover

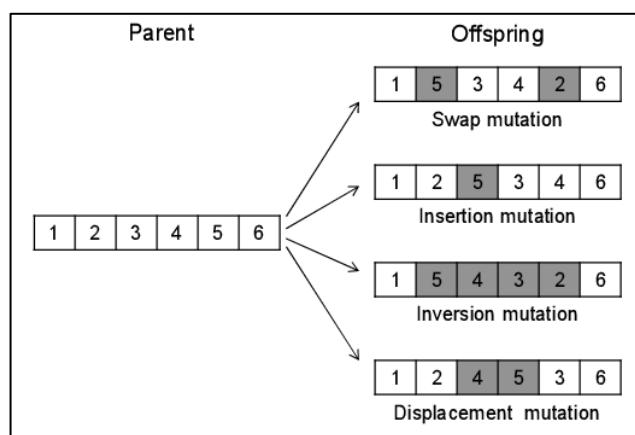
Operasi *crossover* merupakan penggabungan dua kromosom (orang tua dan pasangan) untuk menghasilkan keturunan kromosom yang baru. Ide di balik *crossover* adalah kromosom yang baru bisa jadi lebih baik daripada kedua orang tua yang mempunyai karakteristik terbaik dari masing – masing individu. Selain itu *crossover* juga terjadi pada evolusi dan tergantung pada probabilitas *crossover*. (Kaya et al., 2011) Terdapat tiga tipe *crossover* yaitu *single-point crossover*, *two-point crossover*, dan juga *uniform crossover*. Dalam tipe *single-point*, akan diambil titik *crossover* dari organisme *string* orang tua. Lalu data yang ada pada titik itu *string* organismenya akan ditukar dengan kedua orang tua. *String* sendiri merupakan sebuah bias posisi. Setelah itu untuk *two-point crossover* sendiri menggunakan kasus spesifik dengan titik-N pada teknik *crossover*. Terdapat dua titik yang dipilih secara acak dengan *string* lainnya dengan bahan genetik yang ditukarkan di tiap titik. Yang terakhir yaitu *uniform crossover*, disini setiap gen atau bit dari tiap kromosom orang tua akan diseleksi secara acak dengan gen yang lainnya (Haldurai et al., 2016). Berikut adalah gambaran dari tiap *crossover*:



Gambar 2.12 Tiga tipe *crossover* yang berbeda (Haldurai et al., 2016)

C. Mutation

Setelah dilakukan seleksi dan juga *crossover* dari gen kromosom di algoritma genetik, maka akan dilakukan mutasi gen. Mutasi mencegah algoritma terjebak dalam minimum lokal. Mutasi memainkan peran pemulihian materi genetik yang hilang serta untuk informasi genetik yang mengganggu secara acak. Ini adalah polis asuransi terhadap hilangnya materi genetik yang tidak dapat diubah. Mutasi secara tradisional dianggap sebagai operator pencarian sederhana. Jika *crossover* seharusnya mengeksplorasi solusi saat ini untuk menemukan solusi yang lebih baik, mutasi seharusnya membantu eksplorasi seluruh ruang pencarian. Mutasi dipandang sebagai operator latar belakang untuk menjaga keragaman genetik dalam populasi. Ini memperkenalkan struktur genetik baru dalam populasi dengan memodifikasi beberapa blok penyusunnya secara acak. Mutasi membantu melepaskan diri dari jebakan minimal lokal dan mempertahankan keragaman dalam populasi. Itu juga membuat kumpulan gen terisi dengan baik, dan dengan demikian memastikan ergodisitas (Abdoun et al., 2012).



Gambar 2.13 *Mutation* pada algoritma genetik (Liu, 2014)

2.2.11 Fitness Function

Di dalam *genetic algorithm* untuk menghitung dan mengukur apakah angka yang didapatkan oleh algoritma ini dapat digunakan pada sistem, pastinya diperlukan sebuah

perhitungan eror antara data yang didapatkan dan referensi. Untuk data pengukuran eror ini kriterianya juga disebut sebagai *fitness function*. Terdapat beberapa perhitungan integral eror yang dapat dijadikan *fitness function* untuk seleksi data yang berguna pada sistem seperti ITAE, ISE, dan IAE. Berikut adalah penjelasan tiap fungsi:

A. ITAE

Untuk integral ini merupakan *Integral of Time multiplied by Absolute Error*. Ini menghitung eror yang ada pada waktu ‘t’, karena biasanya untuk eror pertama pada saat sistem dijalankan akan nilainya besar. Sehingga eror ini dihitung untuk menurunkan eror pada awal sistem mulai (Khuriati, 2013). Berikut adalah rumus dari ITAE.

$$ITAE = \int_0^{\infty} t|e(t)|dt \quad (2.26)$$

B. ISE

Untuk integral ini merupakan *Integral of Squared Error*. Ini merupakan ukuran kinerja sistem yang dibentuk dengan mengintegrasikan kuadrat dari kesalahan sistem selama interval waktu yang tetap; ukuran kinerja ini dan generalisasinya sering digunakan dalam teori kontrol dan estimasi optimal linier. Dengan adanya ISE ini berguna agar respons menjadi tercepat dengan amplitudo yang rendah. Tetapi apabila hanya menggunakan ISE saja maka *output* yang ada biasanya sering berosilasi (Khuriati, 2013). Berikut adalah rumus lengkap dari ISE.

$$ISE = \int_0^{\infty} \{e(t)\}^2 dt \quad (2.27)$$

C. IAE

Yang terakhir merupakan *Integral Absolute Error*. Dalam perhitungan integral ini mirip dengan ITAE karena menghitung eror absolut, tetapi pembedanya adalah disini tidak dihitung dengan mengalikan waktu ‘t’. Perhitungan ini menggunakan eror yang mutlak per data dari tiap waktu. Ini membuat respons yang lambat daripada ISE tetapi mengurangi osilasi yang biasa dihasilkan dari ISE. Sehingga apabila digunakan maka osilasi akan berkurang untuk sistem (Khuriati, 2013). Berikut adalah rumus dari IAE.

$$IAE = \int_0^{\infty} |e(t)|dt \quad (2.28)$$

2.2.12 Root Mean Square Error (RMSE)

Ini merupakan salah satu metode pengukuran untuk mengukur berapa perbedaan hasil nilai prediksi dan observasi. Metode ini menggunakan standar deviasi dari hasil dan menghitung eror dari prediksi penelitian. Nantinya angka dari jumlah prediksi (X_i) akan dikurangi oleh nilai hasil (Y_i) dan dikuadratkan. Setelah itu angka kuadrat selisih ini akan dibagi dengan jumlah data (m) yang nantinya seluruh angka ini hasilnya di-akar (Chicco et al., 2021). Berikut adalah rumus dari RMSE:

$$RMSE = \sqrt{\frac{1}{m} \sum_{i=1}^m (X_i - Y_i)^2} \quad (2.29)$$

Halaman ini sengaja dikosongkan

BAB 3 METODOLOGI

3.1 Metode yang digunakan

1. Studi Literatur

Di bagian ini akan menjelaskan tentang teori – teori yang digunakan sebagai penunjang dalam penelitian tugas akhir ini. Disini dapat ditemukan juga data – data dari penelitian tentang Motor induksi, *Variable Frequency Drives*, *PI controller*, *Genetic Algorithm*, operasi genetik.

2. Perancangan Sistem

Di bagian ini akan dilakukan rancangan dari sistem yang digunakan untuk penelitian ini. Penelitian ini menggunakan implementasi pada motor induksi bertipe SGA63264. Selain itu juga dilakukan simulasi dan pengambilan data serta *Genetic Algorithm* akan digunakan *software* MATLAB. Untuk tahap pertama akan merangkai sistem dari *Variabel Frequency Drives* sampai dengan sistem motor induksi.

3. Pengujian Sistem dan Evaluasi

Di bagian ini akan dilakukan pengambilan data awal sebagai referensi. Setelah pengambilan data akan diolah menggunakan *System Identification* agar ditemukan *transfer function* yang berguna untuk kontroler penelitian ini. Lalu juga dilakukan uji sistem dengan *transfer function* yang telah ditemukan dan dibandingkan dengan data *output* awal melihat RMSE nya. Setelah semua itu *transfer function* dari identifikasi sistem dapat divalidasi dan dianggap untuk seluruh sistem. Dari sini dilakukan pengontrolan dengan kontroler PI-GA untuk sistem motor induksi, dengan menyertakan *input* dan melihat *output* respons dari sistem.

4. Pembuatan Tugas Akhir

Di bagian akhir ini akan dilakukan evaluasi ulang hasil dari penelitian dan membuat laporan yang lengkap untuk disusun menjadi laporan Tugas Akhir. Di dalam laporan akhir ini akan ada pendahuluan, dasar teori, metodologi, hasil dan evaluasi, kesimpulan daftar Pustaka, dan juga lampiran.

3.2 Bahan dan peralatan yang digunakan

Untuk penelitian Tugas Akhir ini terdapat beberapa bahan dan juga peralatan yang digunakan untuk memenuhi kebutuhan simulasi dan data penelitian ini. Dalam peralatan yang digunakan pada penelitian ini terdapat dua macam peralatan yaitu *Hardware* (perangkat keras) dan juga *software* (perangkat lunak). Kedua macam ini dapat dijelaskan lebih detail sebagai berikut:

3.2.1 *Hardware*

Peralatan yang digunakan pada *hardware* adalah berikut ini:

a) Motor Induksi dan Motor DC sebagai Tacho

Dalam Tugas Akhir ini digunakan motor induksi 3 fasa dengan seri SGA63264 yang diproduksi oleh Shanghai Machinery Import and Export Corp. Model dari motor induksi ini adalah Squirrel – Cage. Berikut merupakan spesifikasi dari motor induksi ini.

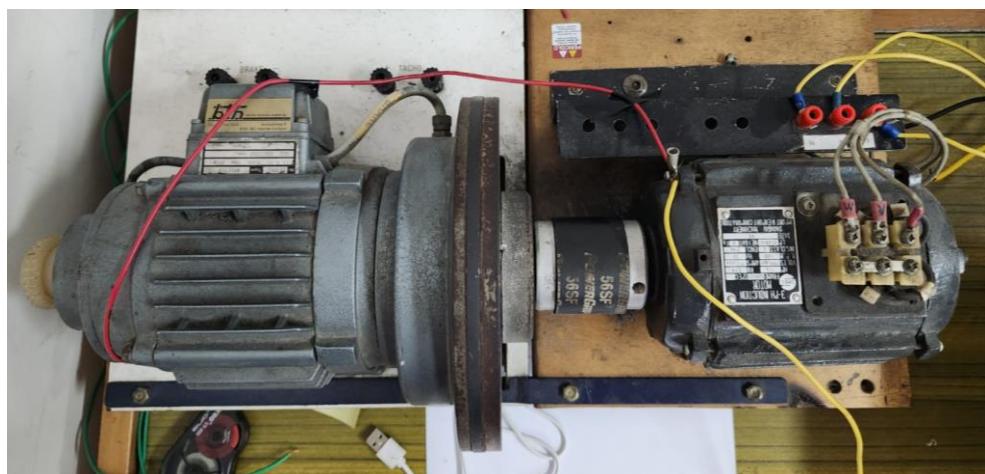
Tabel 3.1 Spesifikasi Motor Induksi

No.	Parameter	Nilai
1	Manufaktur	Shanghai Machinery Import and Export Corp
2	Seri	SGA63264
3	Output	0.25 Hp
4		0.18 kW
5	RPM	1400
6	Tegangan (Star)	380 V
7	Arus (Star)	0.64 A
8	Frekuensi	50 Hz
9	Efisiensi	60%
10	Power Factor	0.73
11	Poles	4
12	Start Torque	2.1
13	Start Current	4.4
14	Max Torques	2.2
15	Min Torque	1.7

Selain itu terdapat Motor DC yang telah tersambung dengan Motor Induksi. Motor DC ini dipakai disambung agar berguna sebagai pembaca kecepatan rotasi (RPM) yang dikeluarkan oleh motor induksi. Motor yang digunakan mempunyai spesifikasi sebagai berikut:

Tabel 3.2 Spesifikasi Motor DC tercoupling dengan AC

No.	Spesifikasi	
1	Manufaktur	Brink Techniek Heerde
2	Tipe	NKU/71R
3	Output	5Nm
4		265W
5	RPM	1000Rpm
6	Tegangan	24Vdc



Gambar 3.1 Sistem Motor AC dengan Motor DC

b) PC

Untuk mengerjakan penelitian Tugas Akhir ini juga digunakan komputer yang memadai guna menjalankan *software* dan juga pembacaan hasil dari penelitian ini. Komputer yang digunakan adalah dengan spesifikasi sebagai berikut.

Tabel 3.3 Spesifikasi PC yang digunakan

Komponen	Spesifikasi
Motherboard	TUF GAMING B660-PLUS WIFI D4
CPU	Processor 12th Gen Intel(R) Core(TM) i5-12600
GPU	Nvidia RTX 3050
RAM	16 GB
Memori internal (SSD)	Samsung SSD 980 500GB
Memori internal (HDD)	WD HDD 2TB
Display	Xiaomi 24" UHD
Sistem Operasi	Microsoft Windows 11 Pro

c) Variable Frequency Drive

Dalam penelitian ini digunakan alat *Variable Frequency Drive* yang diproduksi oleh Schneider, dengan tipe Altivar 610. Nantinya VFD berguna sebagai *inverter* sinyal untuk mengontrol kecepatan motor induksi. Pengendalian dari alat ini digunakan gabungan MATLAB dan juga Advantech USB6714. Berikut adalah spesifikasi serta gambar dari alat VFD yang digunakan.

Tabel 3.4 Spesifikasi VFD

Nama Produk	Altivar ATV610U75N4
Protokol Komunikasi	MODBUS SL
Suplai Tegangan	380 – 460V
Rating Arus	Max Input: 14.7 A & Max Output: 16.5 A
Rating Daya	7.5 kW
Tipe – Jaringan Listrik	3 Fasa
Output Frekuensi	0 – 500Hz



Gambar 3.2 Altivar 610 VFD

d) Advantech USB 4761

Dalam penelitian ini diperlukan alat untuk mengambil data yang biasa disebut sebagai I/O module atau prosesnya yang disebut sebagai Data Acquisition System. Alat yang digunakan merupakan DAQ USB Module yang dibuat oleh Advantech. Berikut adalah spesifikasi lengkap dari alat ini:

Tabel 3.5 Spesifikasi Advantech USB 4761

Nama Produk	Advantech USB 4761
Protokol Komunikasi	USB 3.0
Jumlah Analog Input/Output	16/2
Sampling Rate Maksimal	200kS/s
Resolution	16 Bits
Digital Input/Output	8/8
I/O Connector	Screw Terminal



Gambar 3.3 Advantech USB 4761

e) *Laser Tachometer*

Dalam penelitian ini juga diperlukan laser *tacho*, yang berguna sebagai alat penghitung kecepatan rotasi dari motor induksi yang satuannya adalah RPM(*Rotation Per Minute*). Alat ini digunakan karena *output* kecepatan dari motor DC (takometer) hanya berupa tegangan, jadi untuk mendapatkan RPM real dari motor induksi diperlukan laser takometer. Berikut merupakan spesifikasi alat takometer yang digunakan.

Tabel 3.6 Spesifikasi Laser Tacho

Tipe	DT-1236L	
Range	Photo Tachometer: 10 – 99,999 RPM	
	Contact Tachometer: 0.5 – 19,999 RPM	
	Surface Speed (m/min.): 0.05 – 1,999.9 m/min	
Display	5 Digits, 10 mm (0.4") LCD	
Accuracy	$\pm(0.05\% + 1 \text{ Digit})$	
Sampling Time	Photo Tacho – 1 Detik ($\geq 60 \text{ RPM}$)	
	Contact Tacho – 1 Detik ($\geq 60 \text{ RPM}$)	
Resolution	0.1 RPM	<1,000 RPM
	1 RPM	$\geq 1,000 \text{ RPM}$
Time Base	Quartz Crystal	
Detecting Distance	50 – 2,000mm	
Laser Light Source	Class 2 Laser Diode (645 nm Wavelength)	



Gambar 3.4 Laser Tachometer

3.2.2 Software

Peralatan yang digunakan untuk *software* adalah MATLAB (aplikasi simulasi), *coding genetic algorithm* untuk merancang *controller PI* dengan VFD, dan juga Simulink untuk pembuatan rangkaian blok diagram desain sistem yang akan digunakan. Selain MATLAB, juga digunakan beberapa perangkat lunak untuk penyusunan tugas akhir sebagai berikut.

Tabel 3.7 Software yang digunakan

Software	Versi
MATLAB	R2023a
Microsoft Word	Office 365
Microsoft Powerpoint	Office 365

3.3 Urutan pelaksanaan penelitian

Dalam pelaksanaan penelitian ini akan digunakan urutan jadwal pelaksanaan yang dirancang menjadi tabel sebagai berikut:

Tabel 3.8 Jadwal Pelaksanaan Penelitian

Kegiatan	Minggu Ke -															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Studi Literatur																
Perancangan Sistem																
Pengujian Sistem dan Evaluasi																
Penyusunan Buku Tugas Akhir																

3.3.1 Diagram Blok Penelitian

Di dalam penelitian ini digunakan pelaksanaan dengan blok diagram yang tertera pada **Diagram 3.1** dibawah ini. Dimana sistem pertama kali akan diberikan referensi input kecepatan dalam bentuk RPM. Lalu setelah dimasukkan kecepatan akan dikonversi dalam bentuk tegangan menggunakan nilai konversi yang nantinya ditentukan. Setelah itu tegangan akan masuk kedalam kontroler PI yang telah diatur secara offline angka PI nya oleh algoritma genetik. Nantinya setelah masuk akan diproses ke inverter untuk menggantikan tegangan dari dua fasa menjadi tiga fasa oleh inverter. Lalu, akan menggerakan motor induksi dan didapatkan kecepatan motornya melalui tachometer yang outputnya berbentuk tegangan. Untuk mendapatkan nilai akhir dalam bentuk kecepatan yang diingkan (Speed ref.) maka akan dikonversi Kembali angka tegangan yang sudah diproses dan dijadikan kecepatan berbentuk RPM.

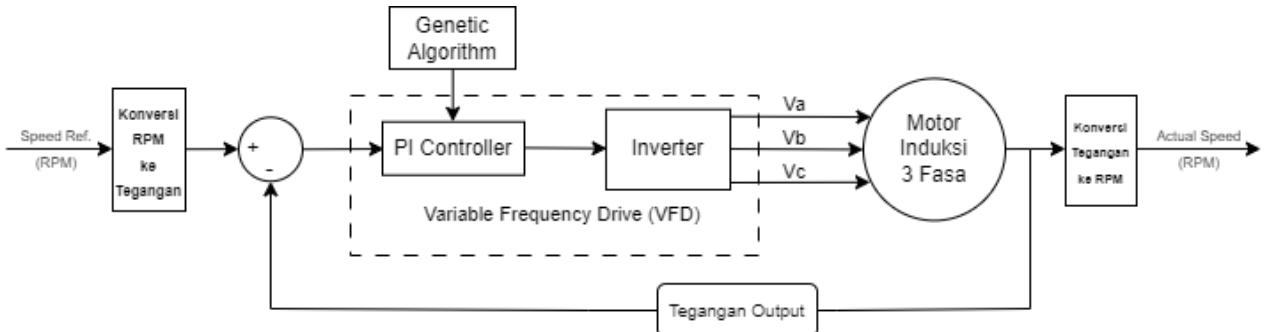


Diagram 3.1 Diagram Blok Sistem Penelitian

3.3.2 Rangkaian Diagram Skema Open – loop Motor Induksi Tiga Fasa dengan VFD

Dalam rangkaian *open – loop* ini berguna untuk mengitung *input* dan *output* tegangan referensi yang digunakan. Lalu saat kita memasukkan tegangan referensi akan diukur dengan takometer berbasis laser dan melihat berapa RPM yang dihasilkan motor induksi pada tegangan tertentu. Selain itu tegangan yang dikeluarkan oleh Motor DC sebagai *tacho* akan diukur tegangannya. Nantinya kedua RPM dari laser dan tegangan *output* motor DC berguna untuk mengetahui perhitungan konversi dari tegangan ke RPM dengan membagi RPM motor AC dan *output* tegangan *tacho* motor DC. Setelah ditemukan perbandingannya maka akan digunakan untuk kontroler PI – GA nantinya pada sistem *closed – loop* penelitian ini. Berikut adalah gambaran blok diagram untuk sistem *open – loop*:

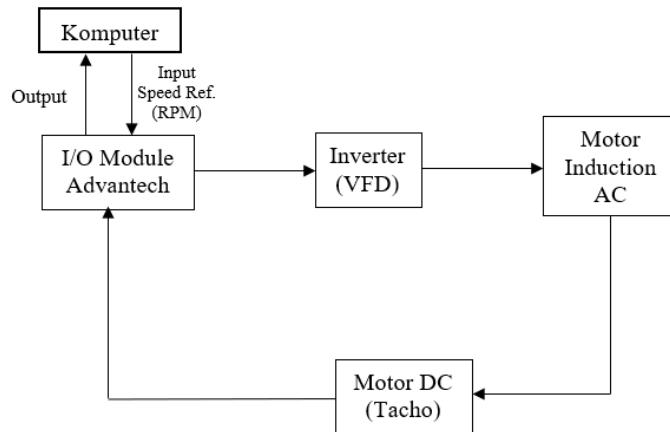


Diagram 3.2 Diagram blok Motor Induksi dengan VFD

3.3.3 Rangkaian Diagram Skema Closed – loop Motor Induksi Tiga Fasa dengan Kontroler PI dan VFD

Di dalam penelitian ini digunakan motor induksi tiga fasa dengan tipe *Squirrel Cage*. Tipe motor induksi ini merupakan salah satu jenis motor induksi yang menghasilkan gerakan, dan menguatkan gaya elektromagnetiknya. Poros luar akan terhubung ke komponen dalam rotor yang terlihat seperti sangkar (*cage*), maka dari itu disebut sebagai *Squirrel Cage*. Sistem ini pertamanya akan dihubungkan secara *closed - loop* dengan step yang merupakan kecepatan referensi dan digabungkan dengan kontroler PI untuk pengontrol dari tegangan yang merupakan konversi dari RPM ke tegangan agar dapat dikontrol dengan kontroler PI-GA, kontroler ini akan . Lalu setelah nilai PI didapatkan melalui GA, maka akan keluar referensi tegangan yang

nantinya memasuki alat *inverter* (VFD) mengubah dari dua fasa kembali ke tiga fasa yang nantinya melewati ke motor induksi. Setelah melewati motor induksi maka akan terhubung dengan motor DC yang merupakan tacho yang menghitung putaran RPM *output* dari motor AC melalui *shaft*. Kecepatan terbaca dan akan masuk ke *I/O module* Advantech USB 4761 dan ter-*input* sebagai bentuk tegangan yang dikonversi oleh *module* menjadi angka bit agar dapat dibaca melalui komputer dan terus melakukan perhitungan dengan PI-GA secara terus menerus, sampai mendapatkan *output* RPM yang diinginkan.

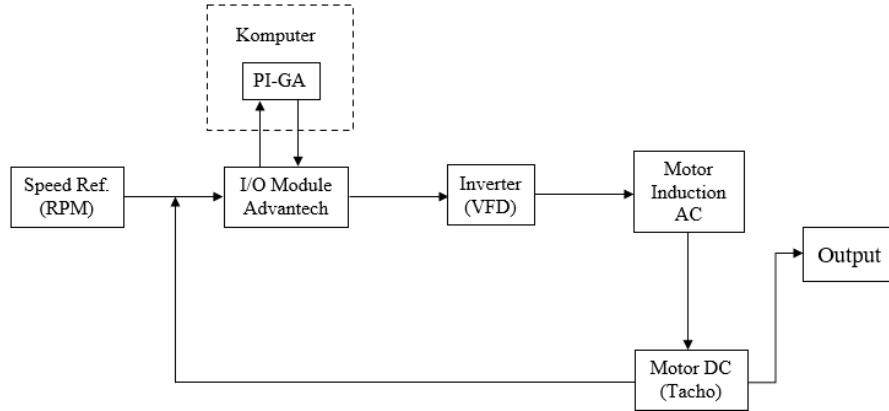


Diagram 3.3 Diagram skema Motor Induksi dengan VFD dengan kontroler PI.

Berikut adalah gambar sistem dalam bentuk rangkaian lengkap:

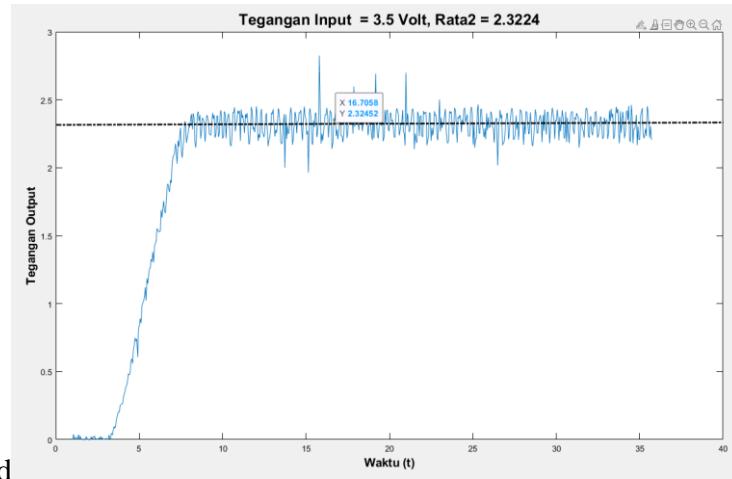


Gambar 3.5 Sistem motor induksi saat terpasang secara lengkap

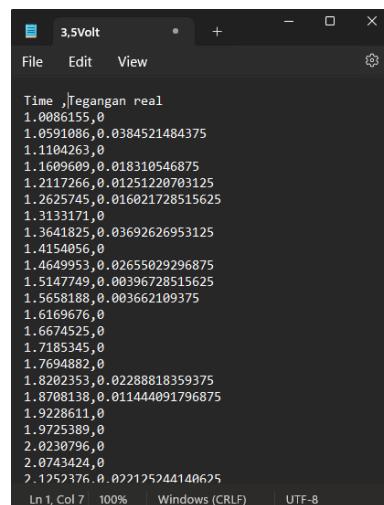
3.3.4 Mendapatkan Data Tegangan dengan Advantech melalui Matlab

Setelah dipasang seluruh sistem yang digunakan untuk penelitian, maka data yang pertama diperlukan adalah mendapatkan *raw* data tegangan *output* yang ada dari sistem motor induksi. *Input* tegangan yang digunakan bervariasi dari 3.5 Volt sampai dengan 7 Volt (dalam selang 0.5 Volt). Pengaturan dari input tegangan ke dalam Altivar diperlukan alat Advantech agar dapat mengatur tegangan melalui komputer. Nantinya Advantech akan berguna sebagai *I/O Module* atau penghubung komunikasi antara sistem dengan computer. Untuk mengatur Advantech sendiri digunakan aplikasi dan coding yang dimasukkan dalam Matlab. Koding lengkap akan ditampilkan pada halaman lampiran. Setelah menginput tegangan dan dijalankan ke sistem, maka akan

dibaca tegangan yang didapatkan dengan time sampling sebesar 0.05 dan dijalankan sampai mencapai *steady – state* pada sistem. Untuk menghitung rata – rata dari *steadystate* sistem digunakan perhitungan dengan koding matlab yang ada pada lampiran. Setelah mendapatkan *output* tiap sistem yang ada maka data akan dikumpulkan dengan bentuk grafik dan juga data *file txt*. Berikut merupakan salah satu contoh hasil dari pengambilan data:



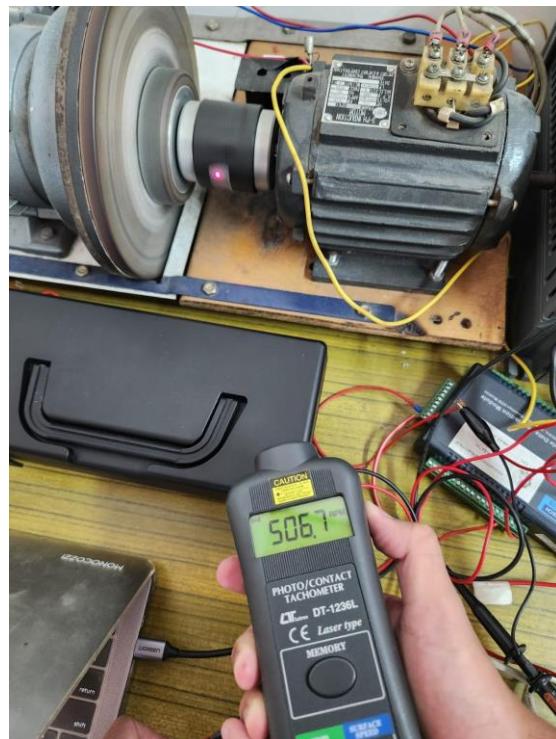
Gambar 3.6 Tampak hasil *output* sistem open – loop di figure Matlab



Gambar 3.7 Tampak hasil *output* sistem open – loop dengan *Txt*.

3.3.5 Mendapatkan nilai RPM dari Motor Induksi dengan menggunakan Laser Tachometer

Setelah data yang didapatkan menggunakan modul I/O, diperlukan juga menghitung rata – rata RPM saat nilai *steadystate* agar bisa dihitung berapa nilai untuk mengganti dari tegangan menjadi RPM pada sistem. Alat yang digunakan adalah laser takometer yang nantinya ditunjuk ke *shaft* motor induksi yang berputar dan dilihat nilai RPM yang terlah dibaca saat *steadystate*.



Gambar 3.8 Pengukuran RPM dengan Laser Tacho pada sistem Motor Induksi

3.3.6 Pengumpulan dan Perhitungan Data Sistem (Tegangan Rata – Rata, RPM, dan nilai konversi)

Setelah melakukan Langkah pada 3.3.3 dan juga 3.3.4 akan didapatkan data seperti tegangan *output* sistem, tegangan rata – rata, RPM rata – rata yang didapatkan melalui laser *tacho* dan yang terakhir adalah nilai konversi dari tegangan menjadi RPM. Lalu juga ada nilai frekuensi yang didapatkan pada sistem *inverter* Altivar yang merupakan VFD yang nilainya adalah Hz.

$$V_{\text{output}} (\text{Steadystate}) = \frac{\text{Total nilai } V \text{ steadystate}}{\text{Total Nilai Data saat Steadystate}} \quad (3.1)$$

Persamaan 3.1 Persamaan Tegangan Steadystate

$$\text{Nilai konversi} = \frac{\text{RPM Rata – rata}}{V_{\text{output}} (\text{Steadystate})} \quad (3.2)$$

Persamaan 3.2 Persamaan Nilai konversi Tegangan ke RPM

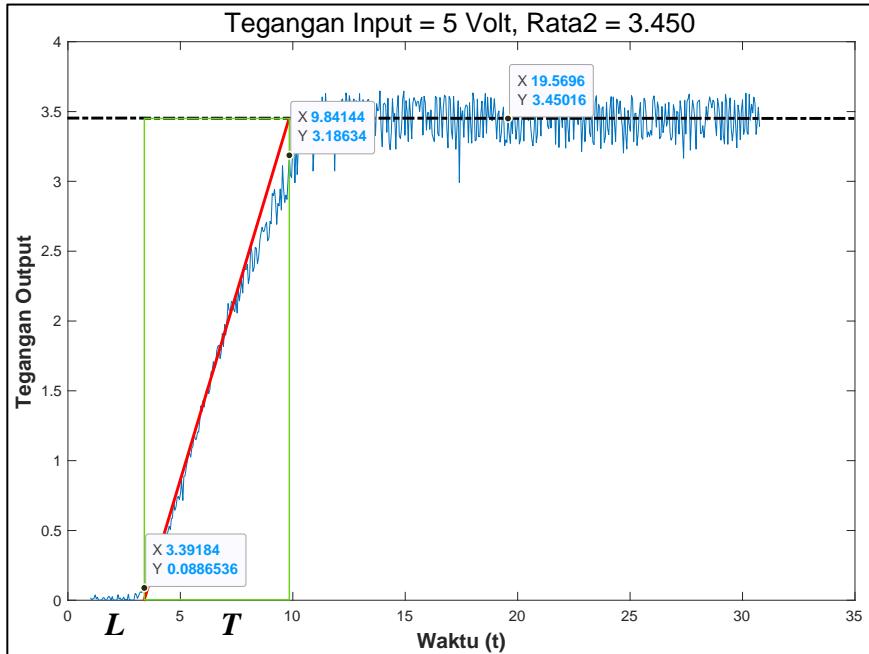
Seluruh data didapatkan pengumpulan data sistem awal dan dijadikan satu tabel sebagai berikut.

Tabel 3.9 Pengambilan data tegangan dan RPM sistem Open – loop

V Input (Volt)	V output Steadystate (Volt)	RPM Rata – rata (RPM)	Frekuensi (Hz)	Nilai Konversi
3.5	2.3224	504.1	17.5	217.0595
4	2.7066	581.4	20	214.8083
4.5	3.0767	652.8	22.5	212.1733
5	3.45	728.3	25	211.1011
5.5	3.8257	802.1	27.5	209.6609
6	4.1921	878	30	209.4415
6.5	4.5833	952.5	32.5	207.7102
7	4.9211	1028	35	208.8954

3.3.7 Pendekatan Fungsi Transfer dengan metode Ziegler Nichols

Setelah seluruh data terkumpul, maka data yang didapatkan dari sistem *open – loop* digunakan untuk menghitung dan mencari fungsi transfer yang berguna untuk kontroler PID. Dalam menghitung fungsi transfer sistem ini dikarenakan parameter yang kurang lengkap, maka digunakan pendekatan identifikasi sistem. Dalam penelitian ini digunakan pendekatan identifikasi sistem dengan metode Ziegler Nichols. Untuk menggunakan metode ini maka diperlukan untuk menggambar garis miring (merah) yang dianggap sebagai garis gradien potong dari titik yang menyentuh dengan kurva berbentuk S pada sistem seperti berikut dengan menggunakan data *input* sebesar 5 volt. Perhitungan Zeigler Nichols sendiri menghasilkan sebuah fungsi transfer yang berbentuk orde 1 dan ditambahkan dengan *delay*, sehingga fungsi transfer menyerupai dengan bentuk orde 2. Garis potong gradien ini ditentukan dengan cara melihat titik naik dari sistem. Ini ditunjukkan melalui grafik sebagai berikut.



Gambar 3.9 Menentukan garis untuk ZN method dengan Tegangan 5 Volt

Di dalam grafik diatas terdapat juga garis hijau yang merupakan batas dari awal garis miring menyentuh dan sampai ke *steady state*. Dari sini dapat ditunjukkan beberapa parameter untuk perhitungan metode Ziegler Nichols. L sebagai titik awal sumbu x pertama kali garis miring menyentuh sumbu y pada nol. Lalu T merupakan selisih titik akhir garis miring ($T + L$) dengan titik awal (L). Dari sini dapat ditemukan hasil masing – masing titik L dan T sebagai berikut

$$\begin{aligned} L &= 1.2 \\ L + T &= 4.65 \\ T &= 4.65 - L \\ T &= 3.45 \end{aligned}$$

Dalam metode Ziegler Nichols terdapat persamaan untuk fungsi transfer sebagai berikut.

$$G_{tegangan}(s) = \frac{Ke^{-Ls}}{Ts + 1} \quad (3.3)$$

Lalu untuk e^{-Ls} merupakan delay dari sistem, jadi pemodelan sistem menggunakan metode ini akan langsung dihitung dengan delay yang ada pada sistem saat data diambil. Ini memudahkan agar tidak perlu menambahkan delay saat mencoba fungsi transfer yang telah didapatkan. Delay e^{-Ls} juga dapat diganti atau dijadikan persamaan dengan L sebagai berikut.

$$\begin{aligned} e^{-Ls} &= \frac{1}{1 + Ls} \\ &= \frac{1}{1 + 1.2s} \end{aligned} \quad (3.4)$$

Setelah seluruh parameter untuk membentuk fungsi transfer menggunakan pendekatan Ziegler – Nichols, diperlukan untuk mencari gain K. Ini didapatkan dengan membagi *output steady state* sistem dengan *input* tegangan yaitu 5 Volt. Berikut adalah persamaan dan perhitungan lengkapnya.

$$\text{Output Steady State} = 3.450 \text{ Volt}$$

$$\text{Input Tegangan} = 5 \text{ Volt}$$

$$K = \frac{\text{Output Steady State}}{\text{Input Tegangan}} = \frac{3.45}{5} \quad (3.5)$$

$$\mathbf{K = 0.69}$$

Setelah gain dan juga parameter lainnya didapatkan maka kita bisa substitusikan seluruh nilai pada fungsi transfer $G_{\text{tegangan}}(s)$ dan dari sini dapat dijadikan fungsi transfer sistem. Berikut adalah perhitungan lengkap untuk $G_{\text{tegangan}}(s)$ untuk sistem *open-loop* ini.

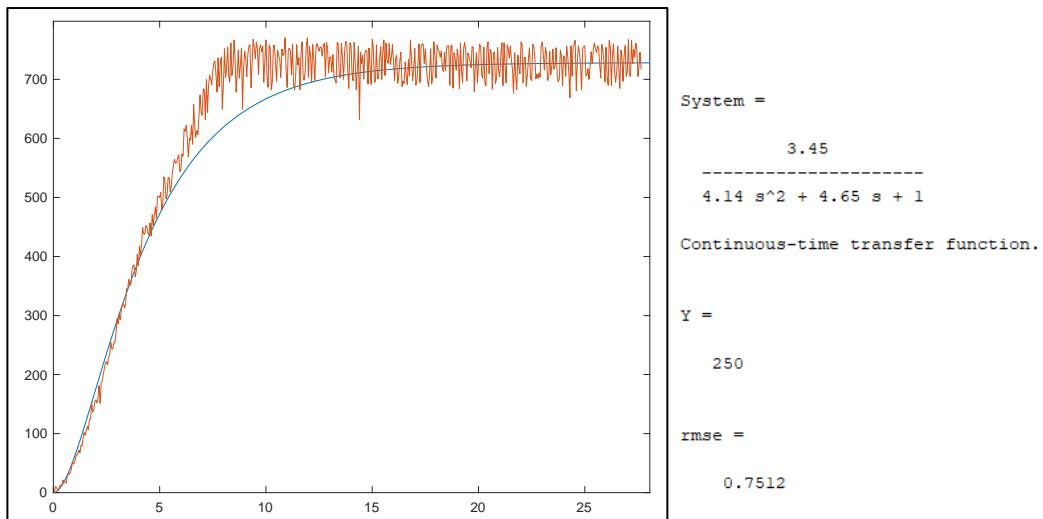
$$\begin{aligned} G_{\text{tegangan}}(s) &= \frac{K}{Ts + 1} * \frac{1}{1 + Ls} \\ &= \frac{0.69}{3.45s + 1} * \frac{1}{1 + 1.2s} \\ G_{\text{Tegangan}}(s) &= \frac{0.69}{4.14s^2 + 4.65s + 1} \end{aligned} \quad (3.6)$$

3.3.8 Perhitungan RMSE Raw data dengan Fungsi transfer

Setelah didapatkan fungsi transfernya, maka perlu dibandingkan untuk sistem dan juga *raw data* yang didapatkan dari sistem. Perbandingan *output* sistem ini digunakan perhitungan RMSE untuk melihat seberapa kecocokan *raw data* dengan fungsi transfer yang didapatkan menggunakan identifikasi sistem metode Zeigler Nichols. Perhitungan RMSE berguna sebagai pembanding antara data yang didapatkan dan juga hasil perhitungan fungsi transfer. Data fungsi transfer akan dikalikan dengan *input* 5 Volt dan dilihat RMSE nya sebagai berikut perhitungan dan juga grafik perbandingannya.

$$\begin{aligned} \mathbf{RMSE} &= \sqrt{\frac{1}{m} \sum_{i=1}^m (X_i - Y_i)^2} \\ \mathbf{RMSE pada 5 Volt} &= \sqrt{\frac{1}{250} \sum_{i=1}^{250} (X_i - Y_i)^2} \\ &= \mathbf{0.7512} \end{aligned} \quad (3.7)$$

Berikut merupakan grafik perbandingannya, dimana warna merah adalah data yang didapatkan, sedangkan garis biru merupakan *output* dari fungsi transfer yang telah didapatkan melalui perhitungan Zeigler Nichols.



Gambar 3.10 Perbandingan sistem ZN method (Biru) dan Real Data (Oranye)

Dikarenakan fungsi transfer yang didapatkan dengan menggunakan tegangan 5 Volt dianggap merepresentasikan seluruh tegangan, maka dari itu digunakan fungsi transfer 5 Volt dan dibandingkan dengan melihat RMSE dari tiap tegangan yang sebelumnya telah ditemukan melalui *open – loop* sistem **Tabel 3.9** diatas. Berikut merupakan tabel rmse dengan fungsi transfer yang telah didapatkan pada 5 Volt.

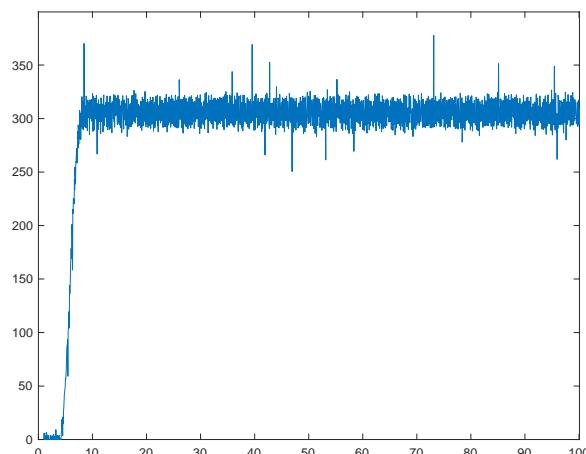
Tabel 3.10 Perbandingan data RMSE hasil Transfer Function dan open – loop sistem.

Input Tegangan	Orde	Fungsi Transfer	Nilai RMSE
3.5	2	$G_{Tegangan}(s) = \frac{0.69}{4.14s^2 + 4.65s + 1}$	0.2009
4	2	$G_{Tegangan}(s) = \frac{0.69}{4.14s^2 + 4.65s + 1}$	0.3808
4.5	2	$G_{Tegangan}(s) = \frac{0.69}{4.14s^2 + 4.65s + 1}$	0.4280
5	2	$G_{Tegangan}(s) = \frac{0.69}{4.14s^2 + 4.65s + 1}$	0.7512
5.5	2	$G_{Tegangan}(s) = \frac{0.69}{4.14s^2 + 4.65s + 1}$	0.7416

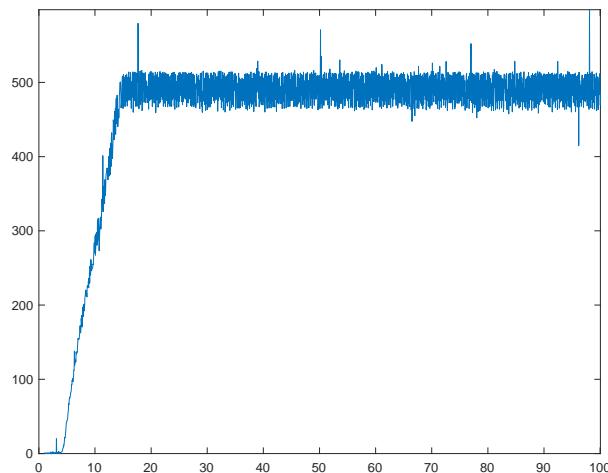
6	2	$G_{Tegangan}(s) = \frac{0.69}{4.14s^2 + 4.65s + 1}$	1.0769
6.5	2	$G_{Tegangan}(s) = \frac{0.69}{4.14s^2 + 4.65s + 1}$	0.7899
7	2	$G_{Tegangan}(s) = \frac{0.69}{4.14s^2 + 4.65s + 1}$	1.6221

3.3.9 Pengambilan data RPM tanpa kontroler pada Sistem Motor Induksi

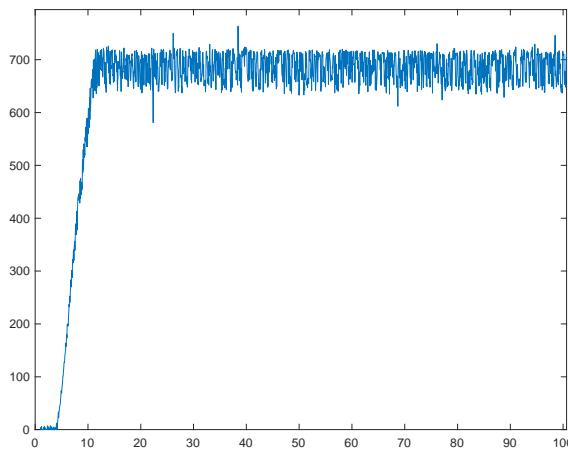
Sebelum dilakukan *tuning* dengan *Genetic Algorithm*, maka diperlukan untuk mengambil data dengan bentuk RPM atau kecepatan rotasi motor. Ini digunakan nantinya sebagai *setpoint* serta sebagai pembanding antar data yang nantinya digunakan untuk membandingkan data sebelum dan sesudah menggunakan kontroler PI – GA. Data didapatkan dengan memberikan *input* sebesar 500rpm, 750rpm, dan juga 1000rpm. Berikut adalah hasil dari pengambilan data RPM tanpa menggunakan kontroler.



Gambar 3.11 Tanpa kontroler dalam 500 Rpm



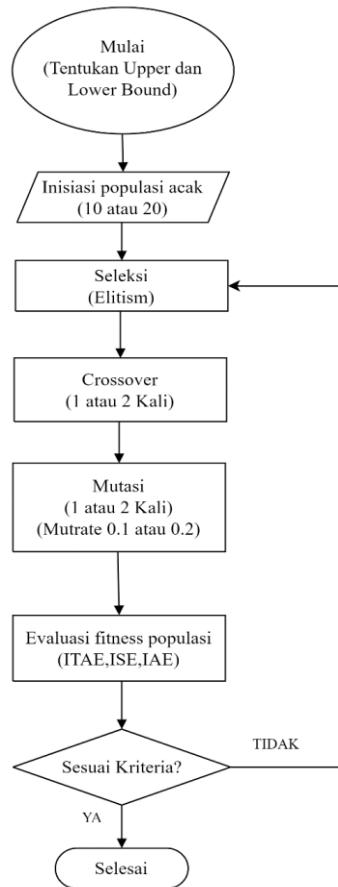
Gambar 3.12 Tanpa kontroler dalam 750 Rpm



Gambar 3.13 Tanpa kontroler dalam 1000 Rpm

3.3.10 Pembuatan *Genetic Algorithm* untuk *PI Controller*

Dalam penelitian ini digunakan kontroler PI dengan optimisasi *Genetic Algorithm*. Disini akan dibagi menjadi beberapa bagian proses untuk pembuatan *genetic algorithm*. Untuk penelitian ini juga dilakukan proses tuning PI – GA digunakan simulasi menggunakan Matlab, karena apabila langsung implementasi dengan motor induksi dapat merusak motor apabila *output* yang dikeluarkan atau saat proses mencari GA melewati batas dan memberikan *output* tegangan yang tidak diinginkan dan dapat membakar motor. Berikut adalah skema proses dari *genetic algorithm* dalam penelitian ini.



Gambar 3.14 Diagram Alur Genetic Algorithm

3.3.11 Operasi Genetik dalam PI – GA

Dalam penelitian ini seluruh simulasi *tuning* PI – GA digunakan angka RPM sebesar 750rpm karena mempresentasikan angka tengah dari 500 – 1000 rpm yang menjadi batasan kecepatan dari penelitian ini. Lalu, untuk yang pertama adalah perlu dilakukan batasan (*upper bound*) untuk angka PI yang akan di atur. Memberikan Batasan (*bound*) untuk sistem berguna agar nilai PID tidak melebihi atau kurang dari angka yang sudah ditentukan, sehingga sistem GA dapat melakukan perhitungan dengan tepat dan juga efisien. Untuk menghitung *upper bound* digunakan angka Kp, Ki, dan Kd yang digunakan melalui perhitungan ZN di **Persamaan 2.22** Dengan menggunakan referensi perhitungan tabel yang ada pada **Tabel 2.1** diatas. Setelah dihitung akan didapatkan batasannya sebagai berikut adalah batasannya:

Tabel 3.11 Angka PID menggunakan ZN method

Kp	Ki	Kd
2.5875	0.646875	0

Angka Kd diberikan nol karena untuk ini digunakan kontroler PI saja, jadi untuk gain derivatifnya tidak dipakai. Setelah ditentukan seluruh angka batasan atasnya (*upper bound*) maka untuk batas bawah (*lower bound*) sendiri diberikan nilai nol untuk semuanya. Ini dikarenakan apabila nilai PID negatif, nantinya hasil dari output sistem tidak bisa stabil. Setelah itu dilakukan simulasi dengan GA menggunakan coding yang ada pada halaman lampiran. Setelah dilakukan simulasi dengan angka PID dengan ZN ditemukan bahwa hasil Xmin lebih sering Kp dibawah 1 dan Ki dibawah 0.5, jadinya untuk batasan angka PID ditentukan sebagai berikut.

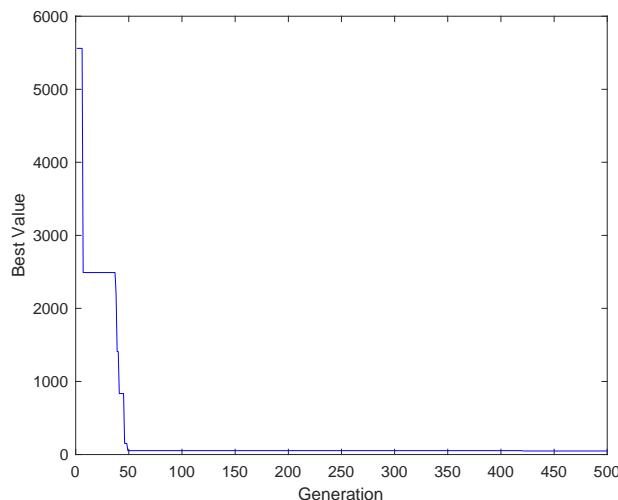
Tabel 3.12 Angka PID yang digunakan PI-GA

Kp	Ki	Kd
1	0.5	0

Setelah melakukan penentuan untuk angka PID sebagai Batasan maka dimasukkan Terdapat beberapa bagian dalam sistem GA ini yaitu:

A. Penentuan populasi dan generasi

Dalam penelitian ini pertama digunakan banyaknya generasi sebanyak 100 dan 500 dan dilihat apakah generasi banyak diperlukan atau tidak dalam penelitian ini. Saat sudah dilakukan tes didapatkan bahwa generasi 100 sudah cukup. Karena saat dijalankan di 500 generasi ditunjukkan pada Gambar 3.11 dibawah. Terlihat bahwa tidak sampai generasi 100 *fitness value* sudah mencapai nol yang berarti sistem sudah bisa dibilang tidak perlu generasi diatas 100 gen.



Gambar 3.15 Hasil 500 generasi dengan 20 populasi

Lalu untuk populasi pertama kali diberikan 10 dan dibandingkan dengan 20 populasi dan dilihat apakah populasi lebih banyak dapat mengeluarkan *output* lebih baik. Untuk populasi pastinya akan berpengaruh juga dengan berapa kromosom atau data yang nantinya akan diseleksi dan mutasi untuk mendapatkan angka PI yang terbaik melalui *genetic algorithm*. Penggunaan populasi berpengaruh dengan hasil serta banyaknya kromosom yang dijadikan *parent* untuk persilangan. Berikut adalah contoh populasi yang didapatkan angka acak.

Tabel 3.13 Populasi 10 dalam genetic algorithm

Populasi (kromosom)	Kp	Ki	Kd
1	0.568767475	0.025673864	0
2	0.036945447	0.13358536	0
3	0.662304542	0.312878737	0
4	0.275834585	0.498042407	0
5	0.501265365	0.066161182	0
6	0.237941501	0.033545053	0
7	0.989005564	0.106012277	0
8	0.504589314	0.315284167	0
9	0.679960522	0.065016764	0
10	0.045383669	0.276963979	0

Tabel 3.14 Populasi 20 dalam genetic algorithm

Populasi (kromosom)	Kp	Ki	Kd
1	0.77045814	0.49343415	0
2	0.36563796	0.07536975	0
3	0.70626364	0.38329283	0
4	0.24051833	0.45224416	0
5	0.87467431	0.13386738	0
6	0.82833836	0.43977477	0
7	0.25103041	0.42688492	0
8	0.1315882	0.15043942	0
9	0.54644161	0.21834758	0
10	0.20463645	0.04507496	0
11	0.18712597	0.32567504	0
12	0.94228801	0.32010143	0
13	0.3768231	0.00439504	0
14	0.44319232	0.16968412	0
15	0.08841115	0.00186556	0
16	0.29366302	0.35567618	0
17	0.29608903	0.13103111	0
18	0.07370131	0.39609612	0
19	0.05033658	0.16804578	0
20	0.15186122	0.15477289	0

B. *Elitism* dan Seleksi

Setelah itu populasi atau kromosom yang telah didapatkan akan melewati sebuah proses *elitism* dan juga seleksi. Untuk *elitism* sendiri merupakan fungsi yang memastikan agar populasi atau kromosom yang sudah didapatkan terbaik tidak akan terbuang dan tergantikan dengan yang baru. Jadi *elitism* sendiri bertujuan untuk memilih sejumlah (biasanya kecil) individu terbaik, sehingga mereka tidak dikeluarkan dari populasi, dan dengan demikian dapat menyebarkan sebagian genom mereka ke generasi berikutnya. Jadi, dari generasi ke generasi, keturunan individu elit akan lebih umum daripada keturunan individu non-elit, yang mengarah ke keragaman yang lebih sedikit. Namun, *elitism* dapat membantu percepat peningkatan *fitness* terbaik dari generasi ke generasi, dan meningkatkan kecepatan perubahan. Sedangkan seleksi sendiri merupakan fungsi sistem yang berguna untuk memilah kromosom terbaik dari populasi yang ada. Biasanya seleksi sendiri dilakukan sejajar dengan berapa angka *crossover* (*crossover times*, lihat lampiran kode).

C. *Arithmetic Crossover*

Setelah dilakukan seleksi dari tiap kromosom yang digunakan maka, akan melewati sebuah proses pengalian antar kromosom yang ada yaitu disebut sebagai *Arithmetic Crossover*. Ini berbeda dengan *crossover* biasanya karena *crossover* jenis ini menggunakan *encoding* angka real, berbeda dengan *crossover* biasa pada GA yang menggunakan angka biner. Ini menggabungkan dua kromosom induk secara linier. Dalam crossover aritmatika, dua kromosom dipilih secara acak untuk

crossover, dengan kombinasi linear dari kromosom ini, maka akan menghasilkan *offspring (children)* yang baru dari kedua kromosom (*parent*). Dimana “*a*” merupakan berapa kali akan dilakukan crossover. Untuk perhitungan dari *crossover* aritmetik ini sendiri adalah sebagai berikut.

$$\begin{aligned} \text{Children1} &= a * \text{Parent1} + (1 - a) * \text{Parent2} \\ \text{Children2} &= a * \text{Parent2} + (1 - a) * \text{Parent1} \end{aligned} \quad (3.8)$$

Dari sini dapat ditunjukkan bahwa untuk *crossover* tipe ini tidak menukar antar kromosom tetapi menambahkan dari kedua *parent*. Digunakan tipe ini karena seperti yang sudah diketahui angka dari Kp, Ki, dan juga Kd tidak bisa ditukar nilainya satu sama lain. Proses ini juga menggunakan nilai yang acak dengan menggunakan fungsi “rand” dalam matlab. Untuk pemilihan dari kromosom (*parent*) sendiri digunakan variabel “*crossovertimes*”, yang merupakan jumlah berapa kali akan dilakukan crossover. Dalam penelitian ini dilakukan *crossovertimes* (*a*) yaitu 1 dan juga 2 kali dan dilihat perbandingan dari *output* sistem GA nya.

D. Mutasi

Lalu setelah melakukan *crossover* maka kromosom atau *children* yang didapatkan melalui pengalian kedua *parent* akan di mutase datanya dan menghasilkan *offspring* yang lebih baik. Berbeda dengan crossover child yang didapatkan tidak melalui persilangan tetapi dengan memilih kromosom secara acak dengan menggunakan “*randomsample*”. Nantinya setelah melewati pemilihan maka kromosom akan diacak dengan interval dari 0 ke 1 dan akan dikurangi nilai awalnya. Untuk mutase digunakan fungsi dari matlab yang merupakan “*geneMutation()*”. Di dalam penelitian digunakan *mutation times* 1 dan juga 2 untuk melihat perbandingan apakah menambah mutase dapat mengeluarkan *output* GA yang lebih bagus atau tidak. Lalu selain perkalian mutase dengan hasil *crossover* juga digunakan *rating* sebesar 0.1 dan juga 0.2 untuk penelitian ini.

E. Replikasi

Untuk yang terakhir merupakan replikasi dimana kromosom individu atau *parent* yang tidak mengalami persilangan dan juga mutase. Maka untuk kromosom ini akan diambil secara acak dan diturunkan kepada generasi yang selanjutnya. Jadi replikasi berguna agar seluruh data tidak tertinggal dan mengalami pengolahan agar mendapatkan nilai PI yang paling bagus dan memenuhi kriteria.

F. Fitness Function

Setelah semua *children*, *parent*, populasi/kromosom, melewati seluruh proses diatas tidak lupa untuk digunakan *fitness function* untuk hasil dari data. *Fitness function* beraksisi sebagai kriteria untuk data yang didapatkan apakah memenuhi kriteria atau tidak. Semisal tidak memenuhi kriteria maka akan dibuang datanya atau diolah lagi agar mendapatkan nilai yang memenuhi. *Fitness function* ini dilakukan dengan menggunakan ketiga perhitungan eror selisih yaitu ITAE, ISE, dan juga IAE (sesuai dengan dasar teori di bab 2). Lalu dilakukan pembobotan (*weight/w1/w2/w3*) dengan angka 0.3. Beban dari *fitness function* sendiri berguna untuk menunjukkan berapa persen respons atau hasil dari sistem dikalikan dengan

perhitungan eror ITAE, ISE, dan juga IAE (lihat bab 2.2.11). Berikut merupakan fungsi perhitungan dari *fitness* yang pada matlab.

$$j=w1*itae(length(itae))+w2*iae(length(iae))+w3*ise(length(ise))$$

3.3.12 Implementasi penggunaan Angka PI-GA dalam Motor Induksi

Setelah dilakukan *tuning* angka PI dengan GA, maka angka yang didapatkan dengan proses mutase, persilangan, dan seleksi akan dikumpulkan datanya dan dimasukkan ke dalam sistem motor induksi. Untuk memasukkan angka PI – GA digunakan kontroler PID yang ada di dalam altivar. Berikut adalah tampak dari *setting* PID dalam VFD Altivar.



Gambar 3.16 Pengaturan PI pada Altivar V610

Nantinya dalam penelitian ini tiap angka PI yang telang di *tuning* oleh GA akan di *input* ke dalam VFD. Untuk *feedback* PID dalam altivar sendiri digunakan tegangan *output* yang dikeluarkan oleh *tacho generator* (motor DC) yang telah ter-coupling Bersama dengan motor induksi. Setelah itu proses pengambilan data sendiri hampir sama dengan proses *open – loop* yang ada pada bagian 3.3.1 diatas. Jadi proses lebih mudah dan praktis karena dalam Altivar digunakan PID kontrolernya.

Halaman ini sengaja dikosongkan

BAB 4 Hasil dan Pembahasan

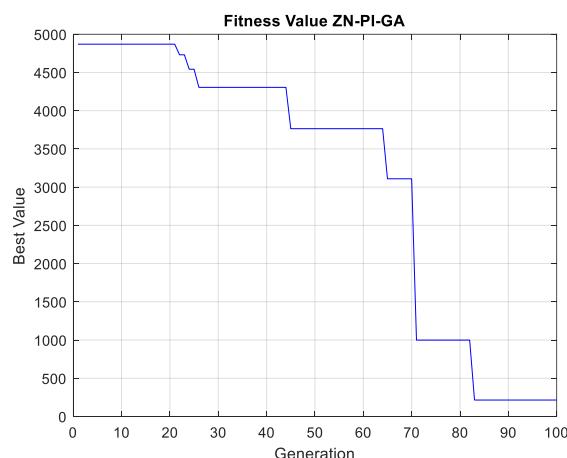
4.1 Hasil Penelitian

Setelah dilakukan pengambilan data maka akan dikumpulkan dan dibagi menjadi beberapa kriteria data. Seluruh data yang didapatkan dari Genetic Algorithm digunakan 100 generasi untuk menjalankan *tuning* angka PI. Untuk data dari simulasi respons 750 Rpm sendiri dibandingkan dengan hasil tanpa kontroler (garis merah). Seluruh data simulasi telah dijalankan dengan *stop time* sebesar 1000 detik, tetapi hanya diambil data sampai *steady state*. Berikut adalah kriteria dan hasil data yang telah didapatkan dalam penelitian ini.

4.1.1 Perbandingan PID – ZN dan PID – GA terbaik

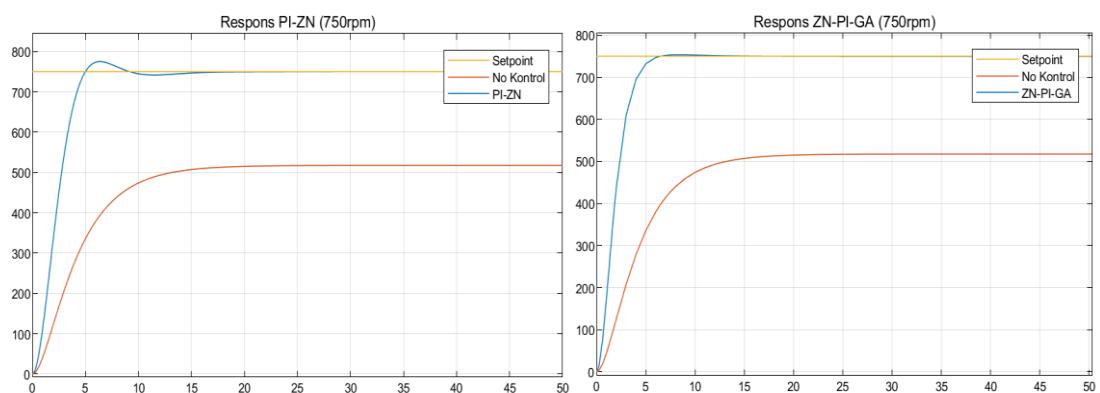
Untuk kriteria penelitian ini akan dibandingkan kedua hasil dari penggunaan ZN saja dan juga parameter PID dengan PID – GA yang menggunakan batasan *upper bound* dengan angka ZN seperti pada tabel 3.11 diatas. Perbandingan ini digunakan untuk menunjukkan apakah menggunakan GA lebih baik daripada *tuning* PID menggunakan ZN saja atau tidak. Parameter GA ini digunakan sebesar 100 generasi, 20 populasi, *mutation rate* 0.1, *crossover* dan *mutation times* sebesar 1 kali. Berikut adalah hasil dari kriteria ini.

a. Hasil Fitness Value



Gambar 4.1 Hasil *fitness* ZN – PI – GA

b. Hasil Simulasi Repons 750 RPM



Gambar 4.2 Hasil Simulasi Respons 750 Rpm PI-ZN dan ZN-PI-GA

c. Hasil Nilai PI

Tabel 4.1 Hasil K_p dan K_i pada ZN – PI dan ZN – PI – GA

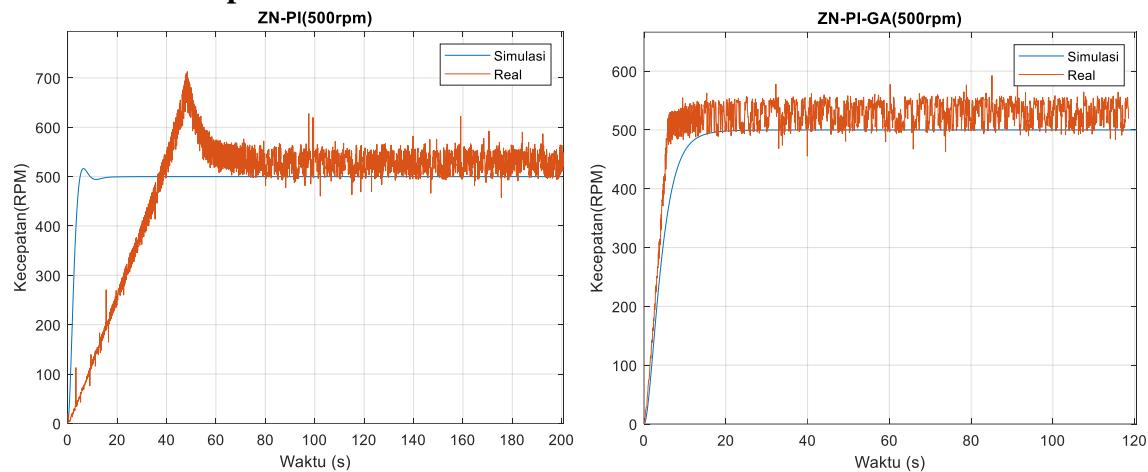
Kontroler	K _p	K _i
ZN – PI	2.5875	0.646875
ZN – PI – GA	1.1264	0.3132

d. Hasil *Integral Error*

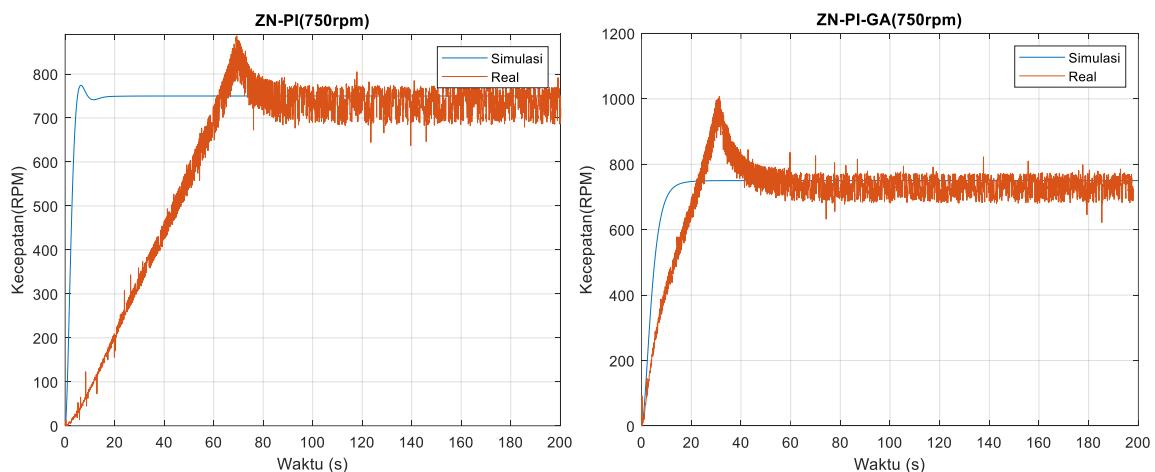
Tabel 4.2 Hasil Integral Error pada ZN – PI dan ZN – PI – GA

Kontroler	ISE	IAE	ITAE	Total
ZN – PI	9.46e+05	3.064e+04	1.547e+07	1.6448e+07
ZN – PI – GA	16.44	7.518	220.5	244.458

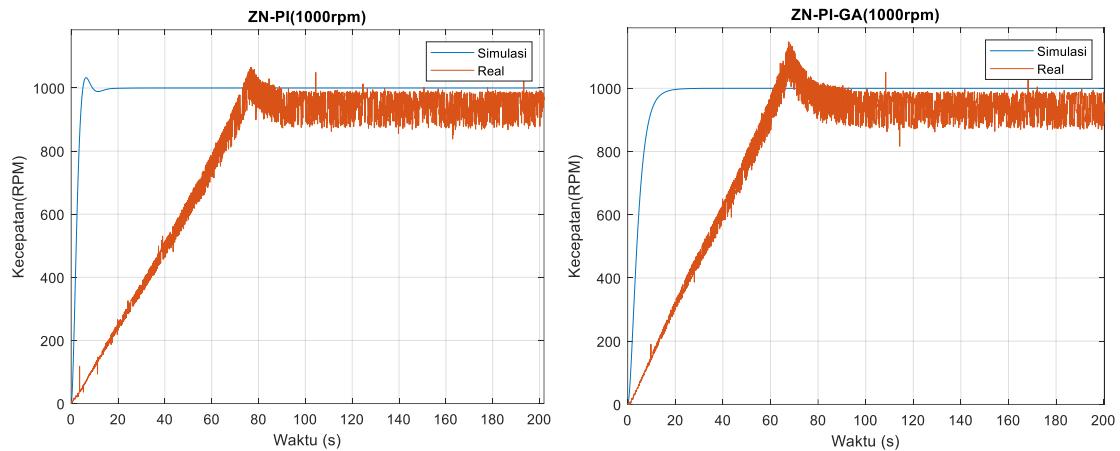
e. Hasil Implementasi



Gambar 4.3 Respon perbandingan ZN – PI dan ZN – PI – GA dalam 500 rpm



Gambar 4.4 Respon perbandingan ZN – PI dan ZN – PI – GA dalam 750 rpm

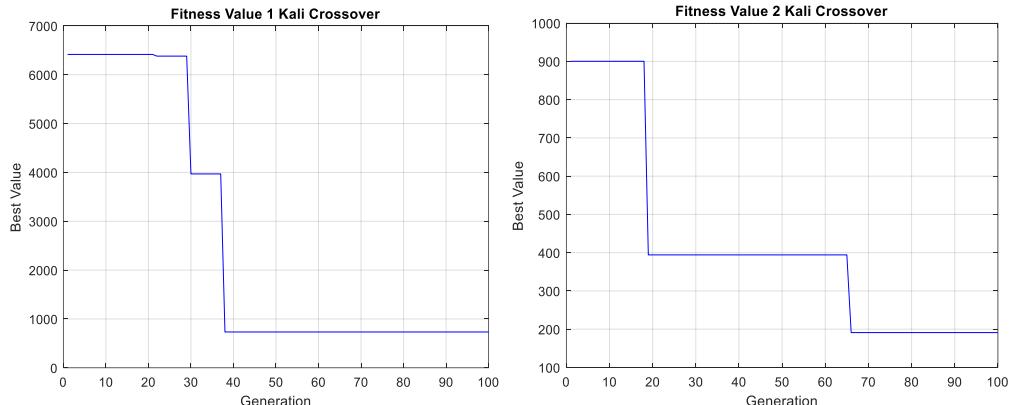


Gambar 4.5 Respon perbandingan ZN – PI dan ZN – PI – GA dalam 1000 rpm

4.1.2 Perbandingan *Crossover* dan *Mutasi*

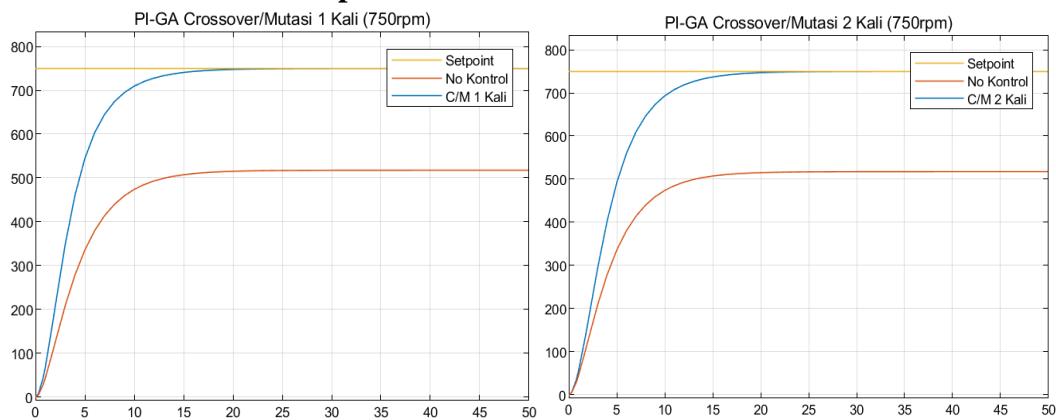
Di dalam data kriteria ini digunakan populasi sebesar 20 dan juga *mutation rate* sebesar 0.1. Lalu digunakan dua bentuk variasi kali ini adalah perkalian *crossover* dan juga mutasinya. Untuk yang pertama (kiri) digunakan *crossover times* dan *mutation times* sebesar 1 kali. Lalu yang kedua (kanan) digunakan *crossover times* dan *mutation times* sebanyak 2 kali. Berikut adalah hasil dari data yang didapatkan.

a. Hasil *Fitness Value*



Gambar 4.6 Hasil *fitness Value Crossover/Mutasi 1 dan 2 kali*

b. Hasil Simulasi Repons 750 RPM



Gambar 4.7 Hasil Simulasi Respons 750 Rpm pada *Crossover/Mutasi 1 dan 2 kali*

c. Hasil Nilai PI

Tabel 4.3 Hasil K_p dan K_i pada Crossover/Mutation 1 kali dan 2 kali

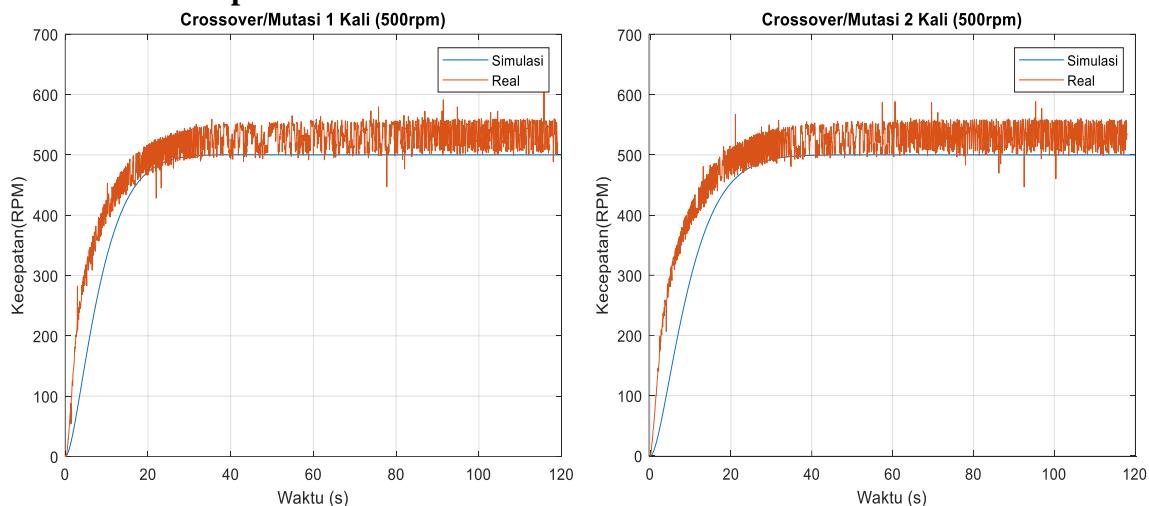
Crossover/Mutation Times	K _p	K _i
1	0.3999	0.1671
2	0.3040	0.1452

d. Hasil Integral Error

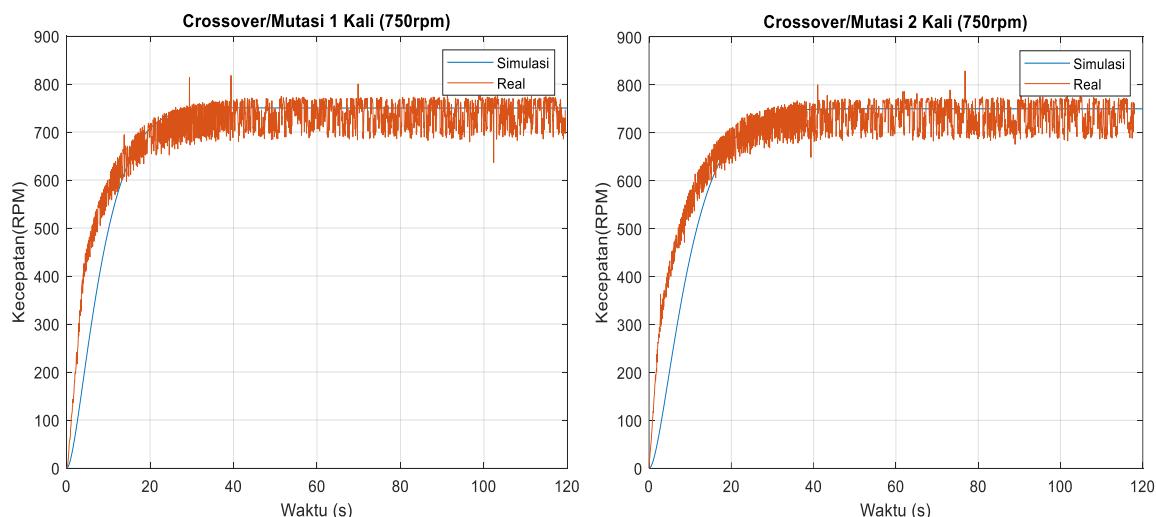
Tabel 4.4 Hasil Integral Error pada Crossover/Mutation 1 kali dan 2 kali

Crossover/Mutation Times	ISE	IAE	ITAE	Total
1	30.8	15.51	605.1	651.41
2	35.46	16.5	139.2	191.16

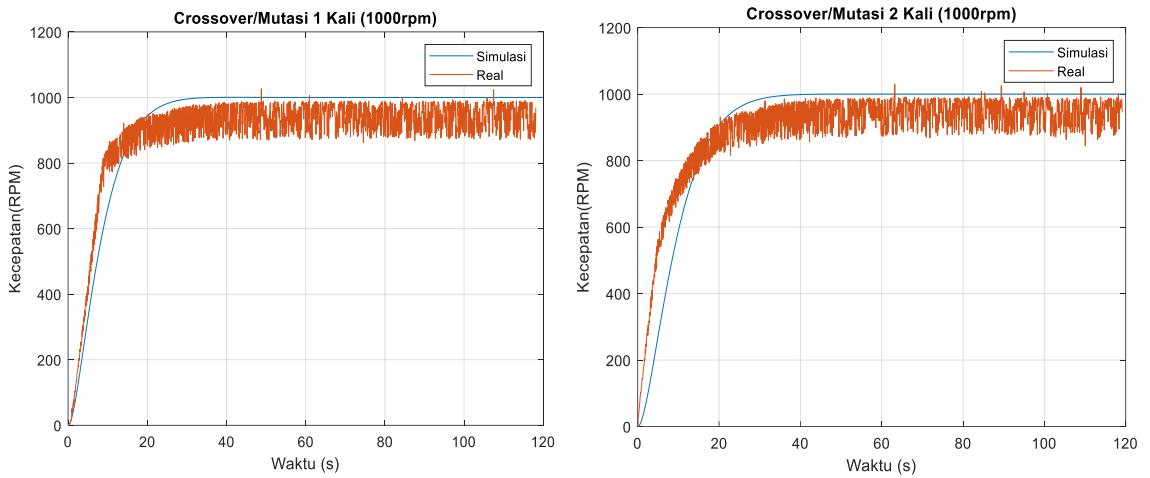
e. Hasil Implementasi



Gambar 4.8 Respon perbandingan Crossover/Mutasi 1 dan 2 kali dalam 500 rpm



Gambar 4.9 Respon perbandingan Crossover/Mutasi 1 dan 2 kali dalam 750 rpm

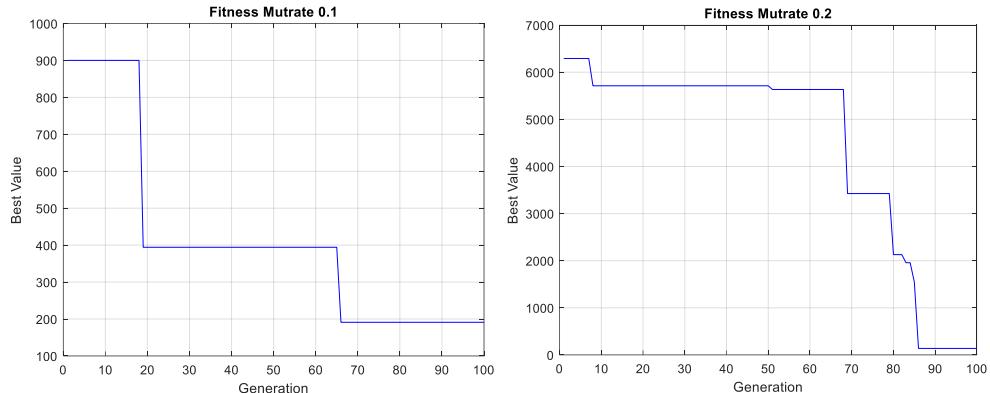


Gambar 4.10 Respon perbandingan *Crossover/Mutasi* 1 dan 2 kali dalam 1000 rpm

4.1.3 Perbandingan Presentase Mutasi

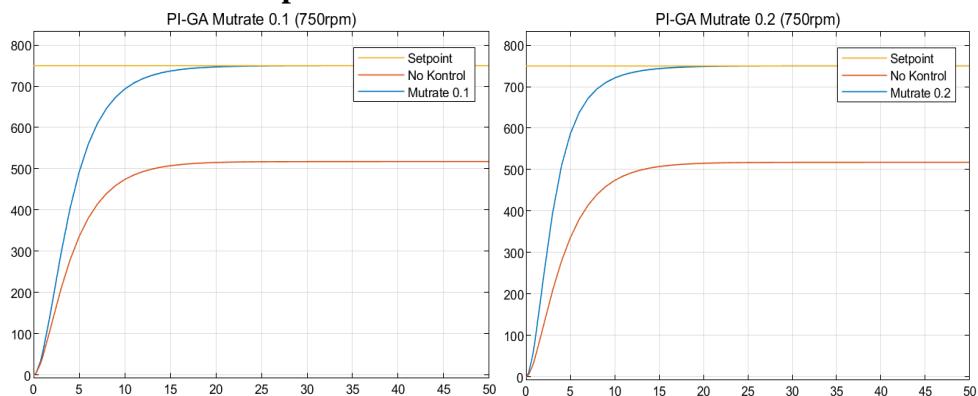
Di dalam data kriteria ini digunakan populasi sebesar 20 dan juga *mutation times* serta *crossover timesnya* sebesar 2 kali. Lalu digunakan dua bentuk variasi persentase mutase yang berbeda yang pertama (kiri) merupakan 0.1 atau sebesar 10 persen. Selanjutnya yang kedua (kanan) merupakan *mutation rate* sebesar 0.2 atau 20 persen dengan hasil berikut.

a. Hasil *Fitness Value*



Gambar 4.11 Hasil *fitness Value* Mutrate 0.1 dan 0.2

b. Hasil Simulasi Repons 750 RPM



Gambar 4.12 Hasil Simulasi Respons 750 Rpm pada Mutrate 0.1 dan 0.2

c. Hasil Nilai PI

Tabel 4.5 Hasil K_p dan K_i pada Mutrate 0.1 dan 0.2

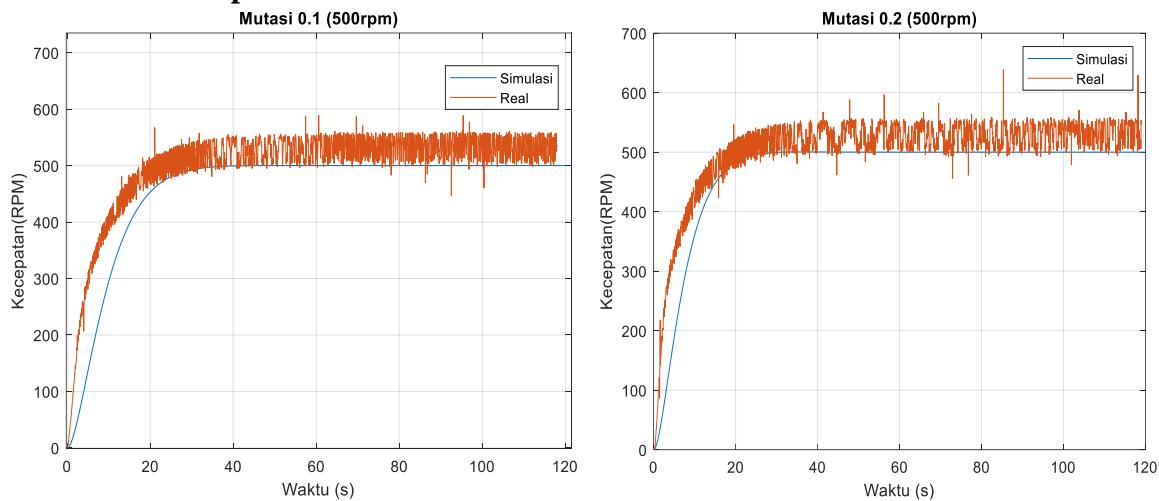
Mutation Rate	K _p	K _i
0.1	0.3040	0.1452
0.2	0.4942	0.1883

d. Hasil Integral Error

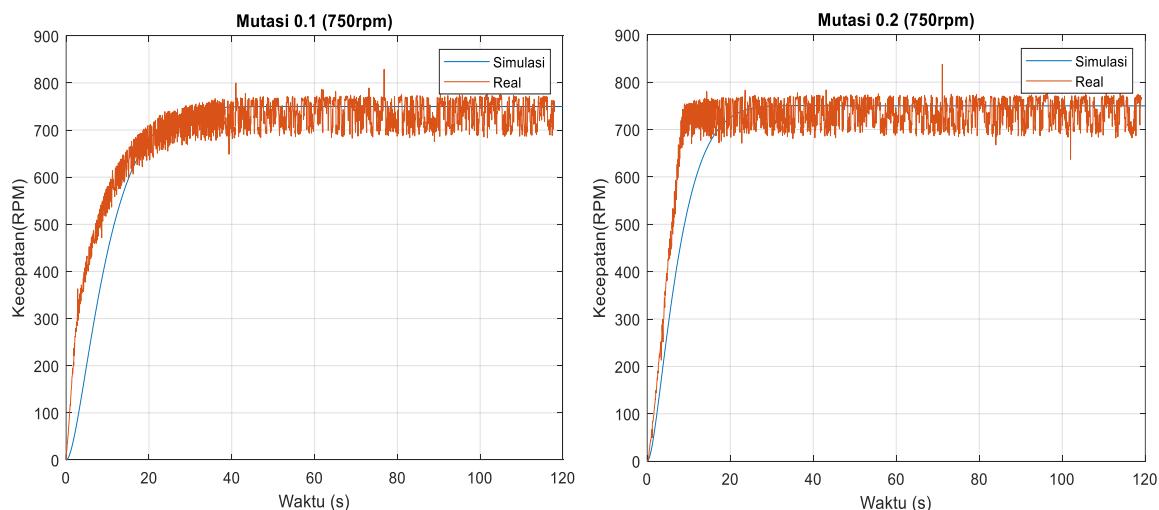
Tabel 4.6 Hasil Integral Error pada Mutrate 0.1 dan 0.2

Mutation Rate	ISE	IAE	ITAE	Total
0.1	35.46	16.5	139.2	191.16
0.2	27.35	13.23	257.3	298.03

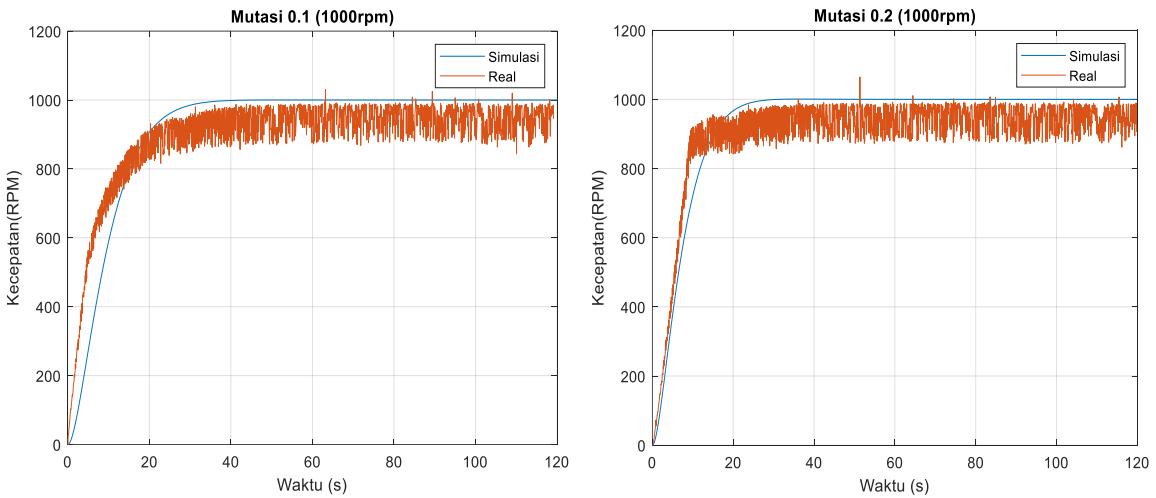
e. Hasil Implementasi



Gambar 4.13 Respon perbandingan Mutrate 0.1 dan 0.2 dalam 500 rpm



Gambar 4.14 Respon perbandingan Mutrate 0.1 dan 0.2 dalam 750 rpm

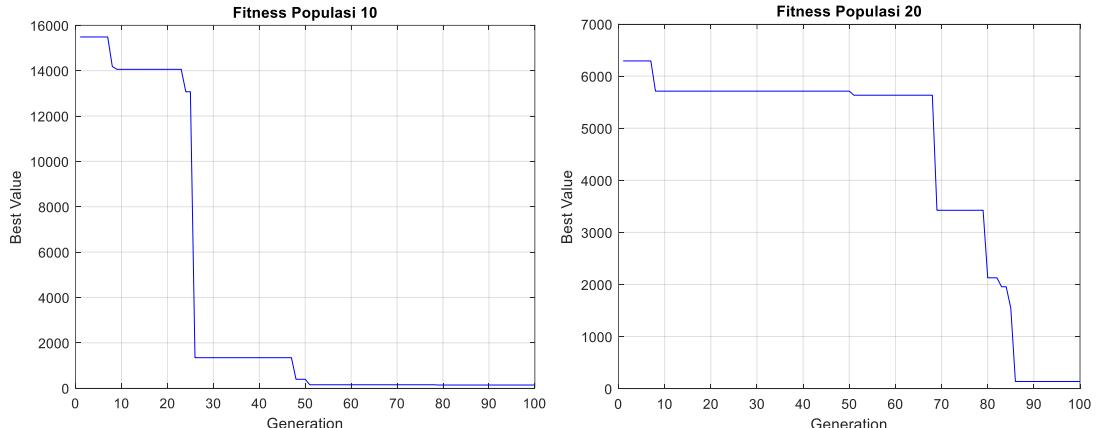


Gambar 4.15 Respon perbandingan Mutrate 0.1 dan 0.2 dalam 1000 rpm

4.1.4 Perbandingan Populasi

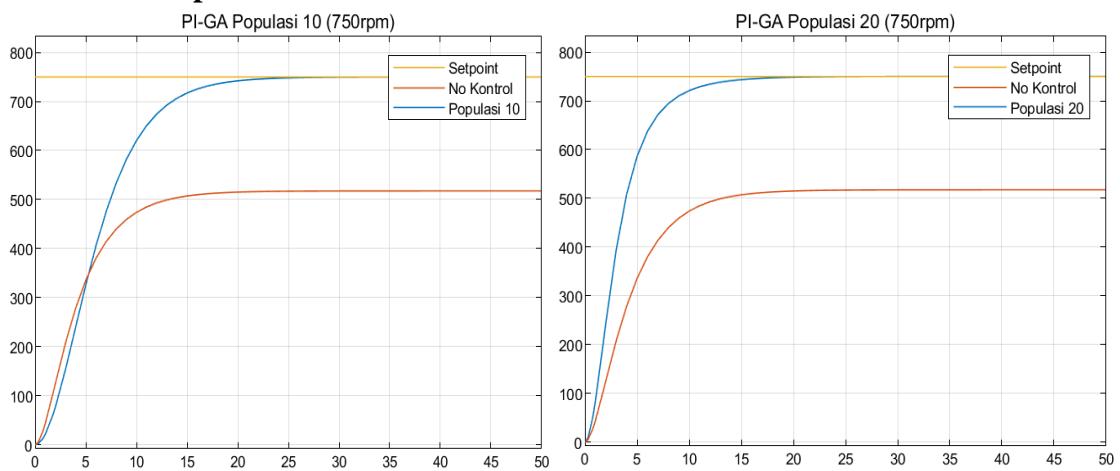
Didalam data kriteria ini digunakan *mutation rate* sebesar 0.2 yang berarti 20 persen. Dan juga *crossover times* serta *mutation times* sebesar dua kali. Digunakan populasi berbeda sebesar 10 populasi (kiri) dan juga populasi sebesar 20 (kanan) berikut hasilnya.

a. Hasil *Fitness Value*



Gambar 4.16 Hasil *fitness Value* 10 dan 20 Populasi

b. Hasil Repons 750 RPM



Gambar 4.17 Hasil *fitness Value* 10 dan 20 Populasi

c. Hasil Nilai PI

Tabel 4.7 Hasil K_p dan K_i pada 10 dan 20 Populasi

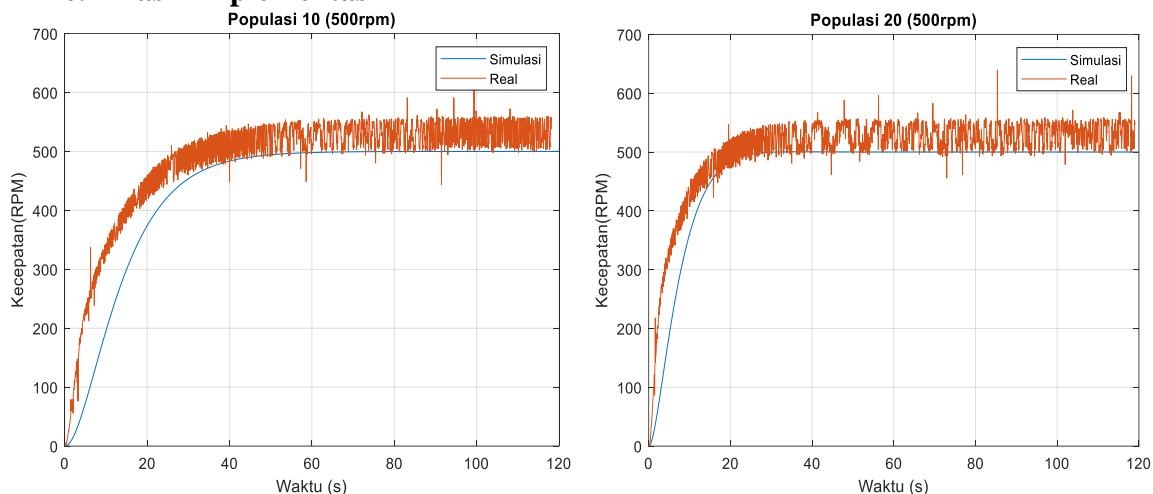
Jumlah Populasi	K_p	K_i
10	0.1030	0.0968
20	0.4942	0.1883

d. Hasil *Integral Error*

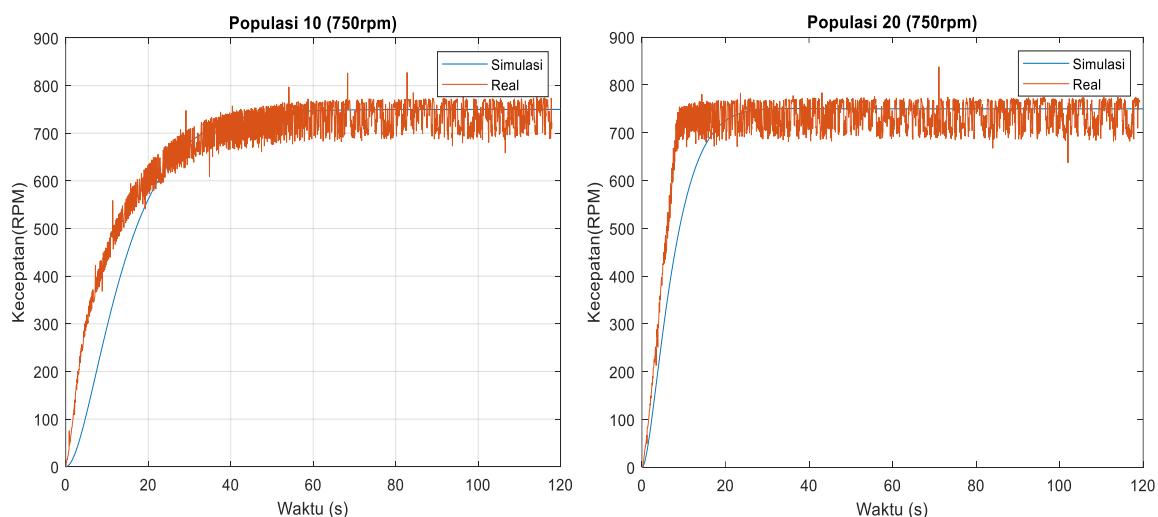
Tabel 4.8 Hasil *Integral Error* pada 10 dan 20 Populasi

Jumlah Populasi	ISE	IAE	ITAE	Total
10	53.18	23.58	419.3	496.06
20	27.35	13.23	257.3	297.88

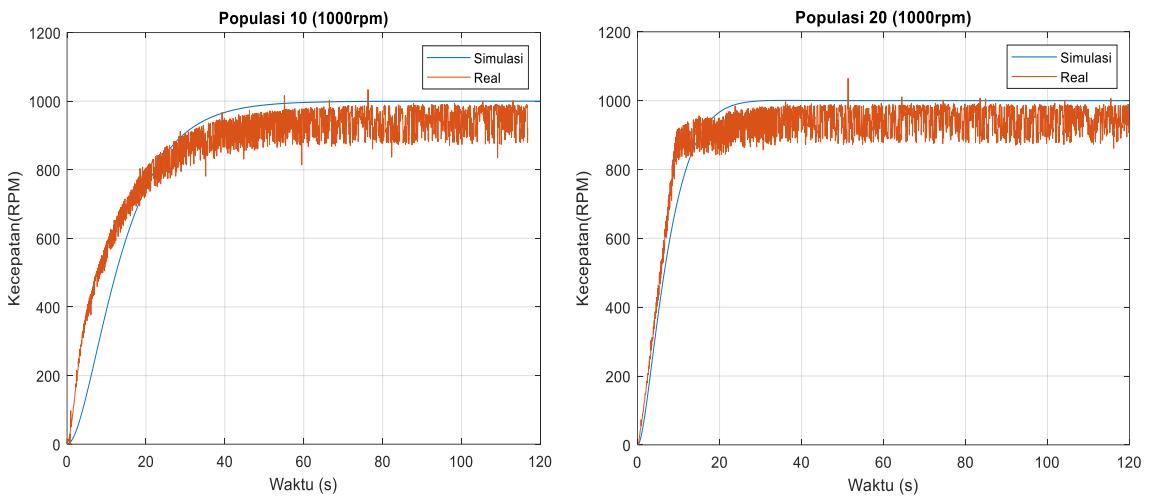
e. Hasil Implementasi



Gambar 4.18 Respon perbandingan 10 dan 20 populasi dalam 500 rpm



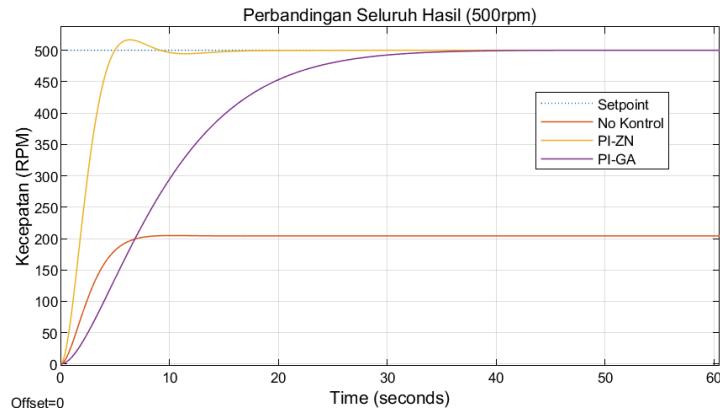
Gambar 4.19 Respon perbandingan 10 dan 20 populasi dalam 750 rpm



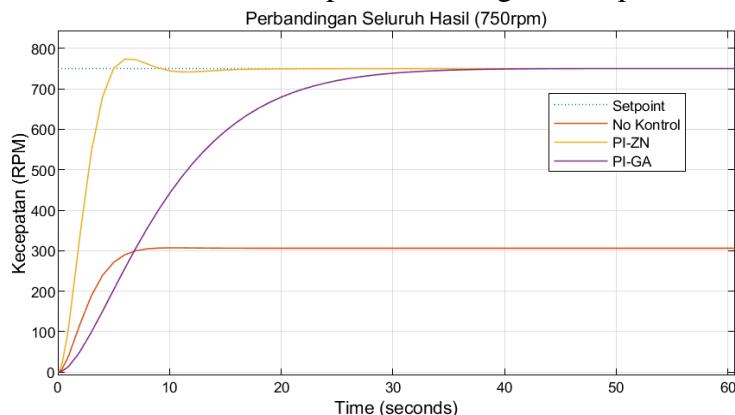
Gambar 4.20 Respon perbandingan populasi 10 dan 20 populasi 1000 rpm

4.1.5 Perbandingan Seluruh Hasil

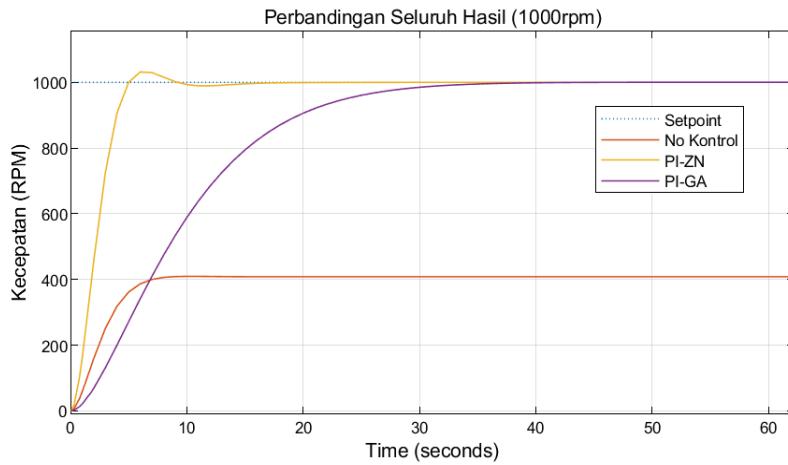
Dalam kriteria ini digunakan data terbaik dari seluruh percobaan dan dibandingkan dengan tanpa PI kontroler, PI – ZN, dan PI – GA. Untuk PI – GA yang digunakan adalah dengan generasi sebanyak 100, populasi sebanyak 20, mutrate sebesar 0.2, serta *crossover* dan *mutation times* sebesar 2 kali. Digunakan PI – GA ini karena setelah didapatkan seluruh data dan diambil parameter yang mempunyai nilai *fitness value* (ITAE, ISE, dan IAE) terkecil tetapi yang *risetime* responsnya cepat. Berikut adalah simulasi perbandingan seluruh hasil yang didapatkan.



Gambar 4.21 Respon hasil dengan 500 rpm



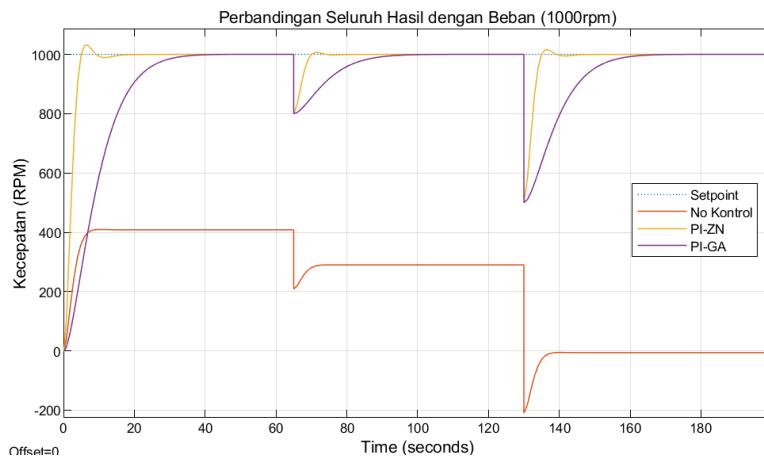
Gambar 4.22 Respon hasil dengan 750 rpm



Gambar 4.23 Respon hasil dengan 1000 rpm

4.1.6 Peggunaan beban pada simulasi

Dalam penelitian kali ini, digunakan penggunaan beban untuk melihat respons dan juga *output* dari sistem apakah bisa mengikuti beban yang diberikan atau tidak. Untuk ini di gunakan *plant* tanpa kontroler, plant dengan kontroler PI – ZN, dan plant dengan kontroler PI – GA yang telah di-*tuning* dengan 0.2 *mutrate*, *crossover/mutasi* 2 kali, 20 populasi. Untuk beban yang digunakan adalah 20 persen dari *setpoint* (1000rpm) yaitu 200rpm, dan selanjutnya dengan beban 50 persen *setpoint* yaitu 500rpm. Untuk beban sendiri dapat diandaikan sebagai pengereman atau beban *load* pada sistem eskalator yang apabila dibebankan dengan manusia saat menyala. Digunakan *setpoint* 1000rpm agar mensimulasikan pada saat *load* maksimum sistem.



Gambar 4.24 Respon hasil dengan beban pada 1000 rpm

4.2 Analisa Data

4.2.1 Perbandingan PID – ZN dan PID – GA

Dari data yang telah didapatkan diatas, untuk perbandingan antara kontroler PI dengan ZN saja dengan PI dengan GA terdapat perbedaan yang sangat jauh. Yang pertama adalah nilai dari K_p dan K_i nya menunjukkan bahwa tanpa GA angkanya lebih besar dibandingkan dengan yang sudah menggunakan GA. Ini disebabkan karena GA akan terus merubah angka K_p dan K_i sesuai dengan batasannya dan parameter lainnya, sampai mendapatkan eror yang

lebih minimal. Lalu data diatas menunjukkan bahwa untuk PI – ZN terdapat nilai eror yang sangat tinggi sebesar **1.6448e+07**. Sedangkan untuk nilai eror PI – GA hanya sebesar **244.458**, yaitu sangat berkali lipat lebih kecil dibandingkan PI – ZN. Ini juga menunjukkan bahwa dengan adanya *tuning* PI dengan GA menghasilkan nilai eror yang jauh lebih kecil.

Di grafik simulasi dengan *setpoint* 750rpm di PI – ZN juga ditunjukkan bahwa terdapat *overshoot* yang lebih besar dibandingkan yang sudah di-*tuning* dengan menggunakan GA. Sebenarnya untuk *overshoot* sendiri sebuah karakteristik yang ada pada kontroler yang di *tune* menggunakan Zeigler – Nichols. Untuk hasil pada implementasi real sistem, ditemukan bahwa pada setpoint kecepatan 500 rpm, menunjukkan respons dengan kontroler PI – ZN terdapat *overshoot* yang sangat besar sama seperti pada di simulasi. Tetapi, di dalam percobaan pada kontroler ZN – PI – GA, didapatkan responsnya sangat cepat mencapai pada setpoint tetapi hasilnya tidak ada *overshoot*, berbeda dengan hasil simulasinya. Ini bisa dianggap karena terkadang di simulasi menunjukkan respons yang lebih bagus atau sama dengan yang ada saat diimplementasi, semua ini terjadi karena dalam real sistem akan ada *noise* atau *disturbance* lainnya dibandingkan dengan simulasi yang tidak ada *noise* atau *disturbance*.

Untuk hasil dari *setpoint* 750 didapatkan bahwa, hasil dari ZN – PI – GA menunjukkan rise time yang sangat cepat kepada *setpoint* dibandingkan tanpa *tuning* GA. Di *setpoint* 1000 rpm menunjukkan berbeda dengan kedua 500 dan 750 rpm, dimana respons kedua kontroler menunjukkan kemiripan, dengan tanpa tuning mempunyai overshoot lebih tinggi *tetapi rise time* yang lebih lambat dibandingkan yang sudah menggunakan *genetic algorithm*.

Kecepatan (RPM)	RMSE	
	ZN – PI	ZN – PI – GA
500	177.8252	44.9869
750	358.8961	112.0757
1000	516.4822	417.6429
Rata – rata	351.0678	191.5685

4.2.2 Perbandingan Kali Crossover dan Mutation

Setelah itu dilakukan pengujian dengan membandingkan angka berapa kali *crossover* dan juga mutase pada sistem GA. Didapatkan *fitness value* yang berbeda antar keduanya, untuk yang pertama saat dijalankan PI – GA dengan *crossover* dan *mutation* 1 kali menghasilkan angka sebesar **651.41** dan untuk hasil dengan 2 kali didapatkan nilai sebesar **191.16** nilai eror dari *fitness valuenya*. Lalu, kondisi ini berbeda dengan hasil dari angka Kp dan Ki nya yang menunjukkan angka pada 1 kali lebih besar dibandingkan dengan yang ada pada 2 kali. Tetapi perbedaannya tidak terlalu jauh dibandingkan nilai *fitness valuenya*. Dari grafik *fitness value* sendiri dapat dilihat juga dengan persilangan yang lebih besar menunjukkan respons *fitness value* yang lebih cepat mencapai nilai minimal yang lebih baik.

Untuk tes sistem menggunakan kecepatan 750 rpm, ditunjukkan perbedaan yang minimal tetapi, pada *crossover* dan mutase satu kali *rise time* lebih cepat dibandingkan

dengan yang dua kali. Untuk *steadystate* sendiri didapatkan juga tren yang sama dengan *rise time* yang ada. Saat digunakan pada real sistem *setpoint* kecepatan pada 500 rpm dan 750 rpm, menunjukkan juga bahwa *risetime* lebih cepat pada *crossover/mutation* 1 kali. Lalu untuk kecepatan 1000 rpm, didapatkan *risetime* yang sangat jauh lebih cepat. Di sisi lain semua hal saat *risetime* lebih cepat pada 1 kali, untuk data dua kali menunjukkan respons yang lebih landau dan menuju *steadystate* lebih baik dibandingkan dengan satu kali. Ini membuktikan bahwa *rise time* cepat tidak menjamin sistem akan berjalan dengan baik, mengingat juga untuk *fitness value* pada dua kali lebih baik juga dibandingkan *crossover/mutation times* satu kali. Untuk *rise time* yang cepat sendiri ditunjukkan karena angka Kp yang lebih besar pada percobaan *crossover/mutation* satu kali, dimana angka Kp pada kontroler PI berguna untuk mengatur *risetime* pada sistem.

Kecepatan (RPM)	RMSE	
	Crossover/Mutasi 1 Kali	Crossover/Mutasi Kali
500	54.3637	61.3445
750	63.5324	71.6443
1000	79.3651	96.5674
Rata – rata	65.7537	76.5187

4.2.3 Perbandingan Presentase Mutasi

Dalam pengujian *mutation rate* didapatkan bahwa dengan angka mutase 0.1 menunjukkan angka *fitness value* sebesar **191.16**, dengan mutase yang 0.2 mempunyai angka *fitness value* lebih besar yaitu **298.03** seperti yang ada pada data yang didapatkan diatas. Ini juga menunjukkan tren yang sama pada angka Kp dan Ki yang dihasilkan pada 0.1 lebih kecil dibandingkan dengan 0.2 ratenya. Lalu pada simulasi percobaan respon kecepatan 750 rpm, didapatkan bahwa *risetime* lebih cepat dengan 0.2 rate dibandingkan dengan 0.1 rate. Dari hasil diatas menunjukkan angka *risetime* sebesar ± 15 detik dibandingkan dengan 0.1 yang *risetimenya* mencapai ± 20 detik. Ini ditunjukkan karena nilai Kp yang lebih tinggi menjadikan *risetime* yang cepat. Lalu pada saat implementasi sistem menunjukkan tren yang sama dimana saat 750 dan 1000 rpm *risetime* untuk rate 0.2 didapatkan ± 20 detik sudah mencapai *rise time*. Berbeda dengan rate 0.1 yang hampir 40 detik baru mulai mencapai *steadystate* dengan *risetime* ± 30 detik. Tetapi seluruh data yang didapatkan menunjukkan tidak ada *overshoot* sama sekali.

Kecepatan (RPM)	RMSE	
	Mutrate 0.1	Mutrate 0.2
500	61.3445	53.2448
750	71.6443	67.6118
1000	96.5674	74.7061
Rata – rata	76.5187	65.1876

4.2.4 Perbandingan Populasi

Dalam pengujian dengan dua variasi populasi yaitu 10 dan 20 populasi, didapatkan bahwa hasil grafik *fitness value* menunjukkan angka *best value* pada populasi 20 yang lebih kecil dibandingkan yang ada pada populasi 10. Ini sebenarnya sejalan dengan konsep banyaknya populasi dapat menghasilkan *best value* yang lebih baik. Karena saat populasi lebih banyak dapat memberikan variasi atau menghasilkan sampel Kp dan Ki terbaik lebih banyak, besar proses maka akan lebih baik hasilnya. Untuk angka *fitness value* dalam populasi 10 sebesar **496.06** dan pada populasi 20 sebesar **297.88** nilainya. Tetapi untuk angka Kp dan Ki lebih besar pada populasi 20, yang saat dilakukan pengetesan dengan respons kecepatan 750 rpm didapatkan bahwa *rise time* di populasi 20 lebih besar dibandingkan pada populasi 10. Angka *rise time* populasi 10 ± 25 detik, dengan hasil respons tanpa *overshoot* dan langsung melandai ke *steady state* sesuai setpoint. Untuk populasi 20 didapatkan *rise time* menuju *steady state* sebesar ± 20 detik dengan hasil respons yang sama seperti 10 populasi tanpa overshoot dan menuju steady state. Lalu pada pengujian dengan real sistem atau implementasi, yang pertama didapatkan bahwa saat respons 500 rpm mempunyai respons yang hampir sama seperti simulasi, dengan angka *risetime* lebih cepat pada populasi 20. Nilai *risetime* populasi 10 sendiri ± 60 detik, tetapi pada populasi 20 menuju *steady state* hanya dengan *risetime* ± 30 detik. Selanjutnya pada respons 750rpm dan 1000 rpm, didapatkan respon yang mirip pada populasi 20 lebih cepat dibandingkan populasi 10. Untuk populasi 10 sendiri didapatkan *risetime* hampir sama dengan respons 500rpm. Tetapi pada populasi 20 ditunjukkan angka *risetime* yang jauh lebih bagus dan cepat dibandingkan *respon* pada 500rpm, dengan angka *risetime* menuju *steady state* ± 15 detik.

Kecepatan (RPM)	RMSE	
	Populasi 10	Populasi 20
500	66.8867	53.2448
750	71.8094	67.6118
1000	95.8845	74.7061
Rata – rata	78.1935	65.1876

4.2.5 Perbandingan Hasil Terbaik

Dalam percobaan yang terakhir ini membandingkan hasil terbaik dari PI – GA dengan PI – ZN dan juga sistem tanpa kontroler. Seluruh percobaan digunakan *closed – loop*. Dari hasil yang telah ada ditunjukkan bahwa respon dengan PI – ZN menunjukkan adanya overshoot dibandingkan dengan PI – GA. Untuk *rise time* pada PI – ZN dan juga tanpa kontroler, didapatkan *rise time* yang cepat dengan ± 5 detik. Dibandingkan dengan respons yang menggunakan kontroler didapatkan *risetime* lebih lambat ± 30 detik. Lalu untuk sistem tanpa kontroler sendiri didapatkan respons tidak mencapai *setpoint*, berbeda dengan menggunakan kontroler yang responsnya mencapai *setpoint*. Perbedaan tanpa kontroler dan juga PI – ZN juga ditunjukkan dengan adanya *rise time* yang cepat dibandingkan dengan respons dengan PI – GA. Tetapi untuk PI – GA sendiri didapatkan hasil yang lebih baik daripada kedua sistem (tanpa kontroler dan PI – ZN). Ini karena untuk PI – GA responsnya

menunjukkan *rise time* yang tidak terlalu cepat, tetapi mencapai *steady state* dengan tanpa *overshoot*.

4.2.6 Perbandingan Hasil Terbaik dengan Menggunakan Beban

Dari data yang telah didapatkan dengan menggunakan beban, menunjukkan bahwa sistem tanpa kontroler bisa mencoba untuk tetap menggerakkan motor, tetapi sistem tidak bisa mengikuti beban dan pada saat *load* 50 persen sistem tanpa kontroler memberhentikan motor. Ini menunjukkan bahwa tanpa kontroler tidak bisa mengikuti beban yang ada. Lalu untuk sistem dengan kontroler PI -ZN maupun PI – GA, didapatkan bisa mengikuti beban dan mengembalikan motor mencapai ke *setpoint*. Tetapi perbedaan antara kedua kontroler adalah seperti yang sudah diuji pada 4.2.5, dimana sistem PI – ZN cenderung mempunyai *overshoot* dan sedikit *undershoot* dan mencapai *setpoint* lebih cepat. Berbeda dengan PI – GA, sistem dengan kontroler ini cenderung memberikan respons yang bagus tanpa *overshoot* dan *undershoot*, cuma *rise time*nya lebih lambat untuk mencapai *setpoint steady state*. Untuk waktu *rise time* dari PI – ZN adalah sekitar 20 detik menuju *steady state* dan PI – GA sekitar 40 detik.

BAB 5 Kesimpulan dan Saran

5.1 Kesimpulan

Dari penelitian motor induksi 3 fasa menggunakan VFD dan juga kontroler dari PI – GA, dapat disimpulkan beberapa kesimpulan yaitu:

1. Sistem PI – GA dengan *offline tuning* dapat memberikan respons yang baik, meskipun *rise time* lebih lambat.
2. Menggunakan PI – GA sebagai kontroler Motor Induksi 3 Fasa, dapat memberikan respons *output* yang baik tanpa *overshoot* tetapi *rise time* yang lambat menuju *steady state*.
3. Apabila menggunakan *crossover/mutation times* yang lebih besar dapat memberikan *respon* yang lebih cepat dan lebih baik dibandingkan perkalian kecil, meskipun *fitness value* yang dihasilkan lebih besar pada 1 kali.
4. Angka populasi dan juga mutrate, dapat mempengaruhi kinerja PI – GA, dimana lebih tinggi angka ini dapat memberikan respons sistem yang lebih baik.
5. Sistem real *plant* atau implementasi dapat memberikan *output* yang berbeda daripada simulasi, dikarenakan adanya *disturbance* atau *noise* natural pada sistem, seperti slip, efisiensi, dan juga *power loss* dikarenakan panas yang diciptakan oleh sistem.

5.2 Saran

Dari penelitian ini didapatkan beberapa saran untuk penelitian selanjutnya dengan topik seperti ini, sebagai berikut:

1. Motor yang digunakan dapat diganti menjadi motor yang mempunyai spesifikasi lebih baik daripada yang digunakan sekarang. Karena spesifikasi motor seperti sekarang ini dapat memberikan *loss* atau *disturbance* yang banyak pada sistem, sehingga dapat memberatkan kontroler atau motor saat bekerja.
2. Dapat digunakan fungsi transfer yang lebih spesifik, seperti setiap tegangan atau rpm dicari fungsi transfernya agar saat simulasi dapat mengeluarkan hasil yang lebih real dengan sistem.
3. Penelitian ini dapat dilanjutkan dengan memberikan perbedaan pada beban seperti *tracking* kecepatan dan juga membuat *tuning* PI – GA menjadi *online tuning*.
4. Dapat dilakukan *tuning* PI – GA dengan populasi, *mutrate*, *crossover/mutation times* yang lebih besar agar memberikan respons sistem lebih cepat dan lebih baik.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

- Abdoun, O., Abouchabaka, J., & Tajani, C. (2012). Analyzing the Performance of Mutation Operators to Solve the Travelling Salesman Problem. *Int. J. Emerg. Sci*, 2(1), 61–77.
- Ahmad, I. (2021). *Working Principle of Variable Frequency Drive (VFD)*.
- Ahmmed, T., Akhter, I., Rezaul Karim, S. M., & Sabbir Ahamed, F. A. (2020). Genetic Algorithm Based PID Parameter Optimization. *American Journal of Intelligent Systems*, 10(1), 8–13. <https://doi.org/10.5923/j.ajis.20201001.02>
- Alnasir, Z. A., & Almarhoon A H. (2012). Design of Direct Torque Controller of Induction Motor (DTC). *International Journal of Engineering and Technology (IJET)*, 4(2), 54–70.
- Åström, K. J., & Hägglund, T. (2004). Revisiting the Ziegler-Nichols step response method for PID control. *Journal of Process Control*, 14(6), 635–650. <https://doi.org/10.1016/j.jprocont.2004.01.002>
- Bousserhane, I. K., Hazzab, A., Rahli, M., Kamli, M., & Mazari, B. (2006). Adaptive PI controller using fuzzy system optimized by genetic algorithm for induction motor control. *International Power Electronics Congress - CIEP*, 133–140. <https://doi.org/10.1109/CIEP.2006.312162>
- Carrier. (2005). *VARIABLE FREQUENCY DRIVE OPERATION AND APPLICATION OF VARIABLE FREQUENCY DRIVE (VFD) TECHNOLOGY*.
- Chicco, D., Warrens, M. J., & Jurman, G. (2021). The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation. *PeerJ Computer Science*, 7, 1–24. <https://doi.org/10.7717/PEERJ-CS.623>
- Deepalis, M., Shirke, Haripriya, H., & Kulkarni. (2013). Microcontroller based speed control of three phase induction motor using v/f method. *International Journal of Scientific and Research Publications*, 3(2). www.ijrsp.org
- Dwyer, A. O. (2009). *HANDBOOK OF PI AND PID CONTROLLER TUNING RULES* (3rd ed.). Imperial College Press.
- Emilio, M. (2013). *Data Acquisition Systems*.
- Haldurai, L., Madhubala, T., & Rajalakshmi, R. (2016). A Study on Genetic Algorithm and its Applications. *International Journal of Computer Sciences and Engineering International Journal of Computer Sciences and Engineering*. www.ijcseonline.org
- Hermanu, C., Apribowo, B., Adhiguna, M. H., & Maghfiroh, H. (2020). Performance Analysis of Three Phase Induction Motor based on ATV12HU15M2 Inverter for Control System Practicum Module. *Journal of Electrical, Electronic, Information, and Communication Technology*, 2(1), 9–13.
- Hjalmarsson, H. (2009). System identification of complex and structured systems. *European Journal of Control*, 15(3–4), 275–310. <https://doi.org/10.3166/EJC.15.275-310>
- Hughes, A. (2006). *Electric Motors and Drives* (3rd ed., Vol. 3).

- Jakoubek, I. P. (n.d.). *Experimental Identification of Stable Nonoscillatory Systems from Step-Responses by Selected Methods*.
- Kaya, Y., Uyar, M., & Tekđn, R. (2011). *A Novel Crossover Operator for Genetic Algorithms: Ring Crossover*.
- Khuriati, A. (2013). Identifikasi dan Perancangan Pengendali PID Menggunakan Penduga ARX Pada Sistem Pemanas Udara. *Jurnal Fisika Indonesia*, XVII(51), 1–5.
- Liu, C. (2014). *Multi-robot task allocation for inspection problems with cooperative tasks using hybrid genetic algorithms*.
- Ljung, L. (2010). Perspectives on system identification. *Annual Reviews in Control*, 34(1), 1–12. <https://doi.org/10.1016/j.arcontrol.2009.12.001>
- Mansour, F. (2020). *Induction Motors: Construction, Principle of Operation, Power and Torque Calculations, Characteristics and Speed Control*. <https://doi.org/10.13140/RG.2.2.15490.71360>
- Pandey, N. D., & Tiwari, P. (2017). COMPARISON BETWEEN SPEED CONTROL DC MOTOR USING GENETIC ALGORITHM AND PSO-PID ALGORITHM. *International Journal of Electrical Engineering & Technology (IJEET)*, 8(1), 17–25. <http://www.iaeme.com/IJEET/index.asp?JType=IJEET&VType=8&IType=1>
- Patel, V. V. (2020). Ziegler-Nichols Tuning Method: Understanding the PID Controller. *Resonance*, 25(10), 1385–1397. <https://doi.org/10.1007/s12045-020-1058-z>
- Samakwong, T., & Assawinchaichote, W. (2016). PID Controller Design for Electro-hydraulic Servo Valve System with Genetic Algorithm. *Procedia Computer Science*, 86, 91–94. <https://doi.org/10.1016/j.procs.2016.05.023>
- Sangwan, S., & Dahiya, A. (2018). *Literature Review on Genetic Algorithm*. <https://edupediapublications.org/journals>
- Savio, I., & Chakraborty, U. (2019). *Genetic Algorithm : An Approach on Optimization*.
- Smith, R. (2022). Variable Frequency Drive Market 2022: Comprehensive Study by Top Key Playes ABB, Siemens, Schneider Electric. *Digital Journal*. <https://www.digitaljournal.com/pr/variable-frequency-drive-market-2022-comprehensive-study-by-top-key-players-abb-siemens-schneider-electric>
- Wang, L. (2020). *Basics of PID Control*. <https://www.researchgate.net/publication/346653954>
- Zarma, T. A., Galadima, A. A., & Aminu, M. A. (2019). *A Review of Motors for Electric Vehicles*.
- Zeraoulia, M., Benbouzid, M., Diallo, D., El, M., & Benbouzid, H. (2006). Electric motor drive selection issues for HEV propulsion systems: A comparative study. *IEEE Transactions on Vehicular Technology*, 55(6), 1756–1764. <https://doi.org/10.1109/TVT.2006.878719>

LAMPIRAN

1. Coding untuk Penggunaan I/O Module Sistem

```
function [arr_vrpm, arr_time] = ReadWrite(VoltAO0,VoltAO1)

% int main, looping once
% clear; clf; clc

BDaq = NET.addAssembly('Automation.BDaq4'); % Make Automation.BDaq assembly visible
to MATLAB.

deviceDescription = 'USB-4716,BID#0';
startChannelAi = int32(0);
channelCountAi = int32(2);

channelStartAo = int32(0);
channelCountAo = int32(1);

Ts=0.05;
global arr_vrpm arr_time;
arr_vrpm=[];
arr_time=[];

instantAiCtrl = Automation.BDaq.InstantAiCtrl();
instantAoCtrl = Automation.BDaq.InstantAoCtrl();

try
    instantAiCtrl.SelectedDevice = Automation.BDaq.DeviceInformation(deviceDescription);
    instantAoCtrl.SelectedDevice = Automation.BDaq.DeviceInformation(deviceDescription);
    % Generate waveform data.
    data = NET.createArray('System.Double', channelCountAi);

    errorCode2 = Automation.BDaq.ErrorCode.Success;
    if BioFailed(errorCode2)
        throw Exception();
    end
    start_time= tic;

    errorCode = Automation.BDaq.ErrorCode();

    t = timer('TimerFcn', {@TimerCallback, instantAiCtrl, startChannelAi, ...
        channelCountAi,instantAoCtrl,data,start_time,VoltAO0,VoltAO1}, 'period', Ts,
    'executionmode', 'fixedrate', ...
    'StartDelay', 1); % Read samples and do post-process, we show data here.

    start(t);
    input('InstantAI-AO is in progress...Press Enter key to quit!');
    stop(t);
    delete(t);
catch e
    if BioFailed(errorCode)
        errStr = 'Some error occurred. And the last error code is ' ...

```

```

        + errorCode.ToString(); % if Something is wrong.
    else
        errStr = e.message;
    end
    disp(errStr);
end

instantAiCtrl.Dispose(); % Close device and release any allocated resource.
instantAoCtrl.Dispose();
plot(arr_time,arr_vrpm);
end

%% Jangan ubah kode dibawah
function result = BioFailed(errorCode)
    result = errorCode < Automation.BDaq.ErrorCode.Success && ...
        errorCode >= Automation.BDaq.ErrorCode.ErrorHandleNotValid;
end

function [vrpm, level] = TimerCallback(obj, event, instantAiCtrl, startChannelAi,
channelCountAi, ...)
    instantAoCtrl,data, start_time,VoltAO0,VoltAO1)
    global arr_vrpm arr_time;
    gain_rpm = 211.101;
    errorCode = instantAiCtrl.Read(startChannelAi, channelCountAi, data);
% kontroller();
    errorCode2 = instantAoCtrl.Write(0,1,VoltAO0);
    errorCode3 = instantAoCtrl.Write(1,1,VoltAO1);

    if BioFailed(errorCode)
        throw Exception();
    end
    if BioFailed(errorCode2)
        e = MException('DAQWarning:Notcompleted', ...
            'StaticAO is completed compulsorily');
        throw (e);
    end
    fprintf('\n');
    current_time = toc(start_time);
    fprintf('timestamp = %2.5f | ', current_time);

    for j=0:(channelCountAi - 1)
        if j == 0
            vrpm = data.Get(0)*gain_rpm;
        elseif j == 1
            level = data.Get(1);
            fprintf('rpm = %2.5f ', vrpm);
            arr_vrpm = [arr_vrpm vrpm];
        end
    end
    arr_time = [arr_time current_time];
end

```

```
% % Input RPM kepada USB
rpm = 750
% % Konversi RPM ke tegangan (x)
x = rpm/211.101
[arr_vrpm, arr_time] =ReadWrite(x,0);
arr_vrpm = arr_vrpm';
arr_time = arr_time';

% % Save data menjadi Txt. Kedalam laptop
writetable(table([arr_time,arr_vrpm]),"750rpm.txt");
```

2. Coding Perhitungan Rata – rata Tegangan dan Hitung RMSE Transfer Function

```
T=readmatrix("Alamat data di PC");
```

```
%%Baca Data Tegangan menjadi RPM
```

```
volt_tacho_data=T(:,2);
time_samplingv5=T(:,1);
k=1:length(volt_tacho_data);
x=k*0.05;
gainrpm = 212.1733
h=gainrpm*volt_tacho_data
```

```
plot(x,h)
```

```
%%Perhitungan RPM/Tegangan menggunakan Transfer Function metode ZN
```

```
gainrpm5 = 211.1011
```

```
input = 4.5;
```

```
%Fungsi transfer
```

```
System = tf(0.69*input*gainrpm5,[4.14 4.65 1], 'InputDelay',0)
[y,t] = step(System,33);
```

```
plot(t,y);
hold on
```

```
plot(x,h);
hold off
```

```
Y= 300 %datatotal-Y
```

```
e = h(1:Y) - y(1:Y);
```

```
%%Perhitungan RMSE
```

```
rmse = (sqrt(sum(e.^2)/Y))/gainrpm
```

```
%%Perhitungan Rata - Rata
```

```
mean_rpmData_sebenarnya=mean(h(230:length(h)))
```

```
mean_rpmData_tf=mean(y(230:length(y)))
```

3. Coding Genetik Algorithm

```
% % Masukkan batas bawah (Lower Bound) dan atas (Upper Bound)
LB = [0 0 0];
UB = [1 0.5 0];
nVars = 3;
% Maksimum generasi GA
generation = 10;
populationSize = 10;
mutRate = 0.2;
crossoverTimes = 2;
mutationTimes = 2;

% Memulai perhitungan populasi
population = zeros(populationSize, nVars);
tempPopulation = zeros(populationSize, nVars);
sPopulation = zeros(populationSize, nVars+1);
F = zeros(populationSize, 1);
for i=1:1:populationSize
    for n=1:1:nVars
        % Masukkan angka random untuk populasi
        population(i, n) = unifrnd(LB(1,n), UB(1,n));
    end
    F(i) = fitness(population(1,:));
end

% Penyortiran Populasi Sesuai dengan PArams
sPopulation(:, 1:nVars) = population(:,:);
sPopulation(:, nVars+1) = F;
sPopulation = sortrows(sPopulation, nVars+1);

%% Selection and Crossover
% Generation
for ii=1:1:generation
    k=1;
    % Elitism
    tempPopulation(k, 1:nVars) = sPopulation(1,1:nVars);
    k=k+1;

    % Seleksi Parent
    for j=1:1:crossoverTimes
        y1(j) = geornd(0.1)+1;
        while y1(j) > populationSize
            y1(j) = geornd(0.1)+1;
        end
        y2(j) = geornd(0.1)+1;
        while y2(j) > populationSize
            y2(j) = geornd(0.1)+1;
        end
    end
end
```

```

% Arithmetic Crossover
for u=1:1:crossoverTimes
    parent1 = sPopulation(y1(u), 1:nVars);
    parent2 = sPopulation(y2(u), 1:nVars);

    [children] = arithmeticCrossover(parent1, parent2);
    tempPopulation(k, 1:nVars) = children(1,:);

    tempPopulation(k, 1:nVars) = max(tempPopulation(k, 1:nVars), LB);
    tempPopulation(k, 1:nVars) = min(tempPopulation(k, 1:nVars), UB);

    k=k+1;
    tempPopulation(k, 1:nVars) = children(2,:);

    tempPopulation(k, 1:nVars) = max(tempPopulation(k, 1:nVars), LB);
    tempPopulation(k, 1:nVars) = min(tempPopulation(k, 1:nVars), UB);

    k=k+1;
end

% Mutasi
for e=1:1:mutationTimes
    parent = sPopulation(unidrnd(populationSize), 1:nVars);
    [child] = geneMutation(parent, mutRate, LB, UB);

    tempPopulation(k, 1:nVars) = child;
    tempPopulation(k, 1:nVars) = max(tempPopulation(k, 1:nVars), LB);
    tempPopulation(k, 1:nVars) = min(tempPopulation(k, 1:nVars), UB);
    k = k+1;
end

% Replikasi
for k=k:1:populationSize
    replicatedChild = sPopulation(randi([1 populationSize]), 1:nVars);
    tempPopulation(k, 1:nVars) = replicatedChild;
    k = k+1;
end

% Hitung Fitness Function untuk pemilihan terbaik
for iii=1:1:populationSize
    F(iii) = fitness(tempPopulation(iii,:));
end

% Penyortiran per Generasi
sPopulation(:, 1:nVars) = tempPopulation;
sPopulation(:, nVars+1) = F(:, :);
sPopulation = sortrows(sPopulation, nVars+1);

bestF(ii) = sPopulation(1, nVars+1);

%% Simulasi Kromosom terbaik
Xmin = sPopulation(1,1:nVars)

```

```

end

%Simulasi dari Kromosom terbaik setiap generasi
Fval = bestF(1, generation);
figure(3);
plot(bestF, 'b');
xlabel('Generation');
ylabel('Best Value');
grid on

```

4. Coding Fitness Value

```

function cost = fitness(x)
assignin('base','x',x);
sim('simulinkGA.slx');
w1=0.3;
w2=0.3;
w3=0.3;
j=w1*itae(length(itae))+w2*iae(length(iae))+w3*ise(length(is));
cost=itae(length(itae));
end

```

5. Coding Arithmetic Crossover

```

function [children] = arithmeticCrossover(parent1, parent2)
alpha = rand;
child1 = alpha*parent1 + (1-alpha)*parent2;
child2 = (1-alpha)*parent1 + alpha*parent2;
children = [child1; child2];
end

```

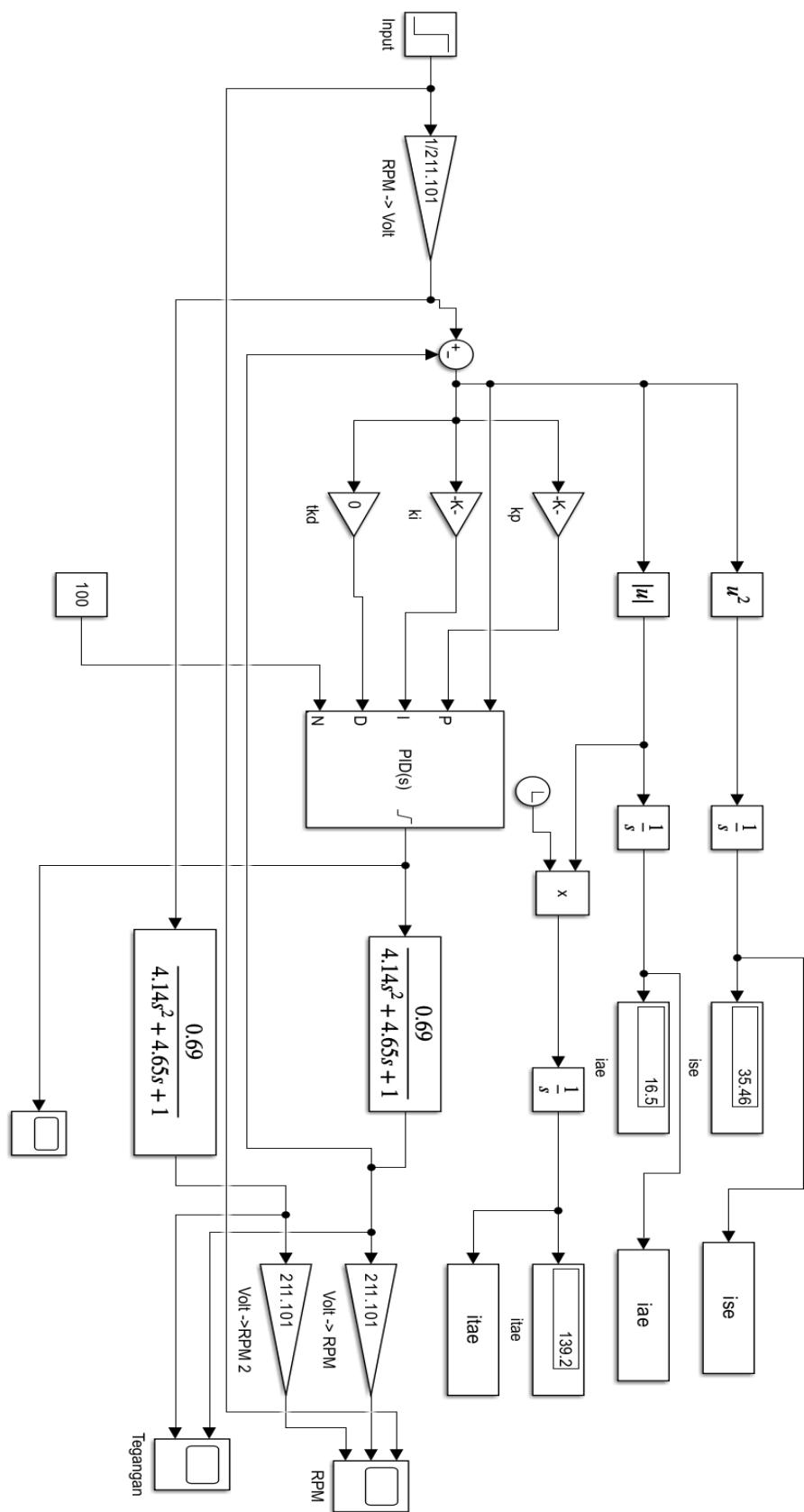
6. Coding Mutasi Gen

```

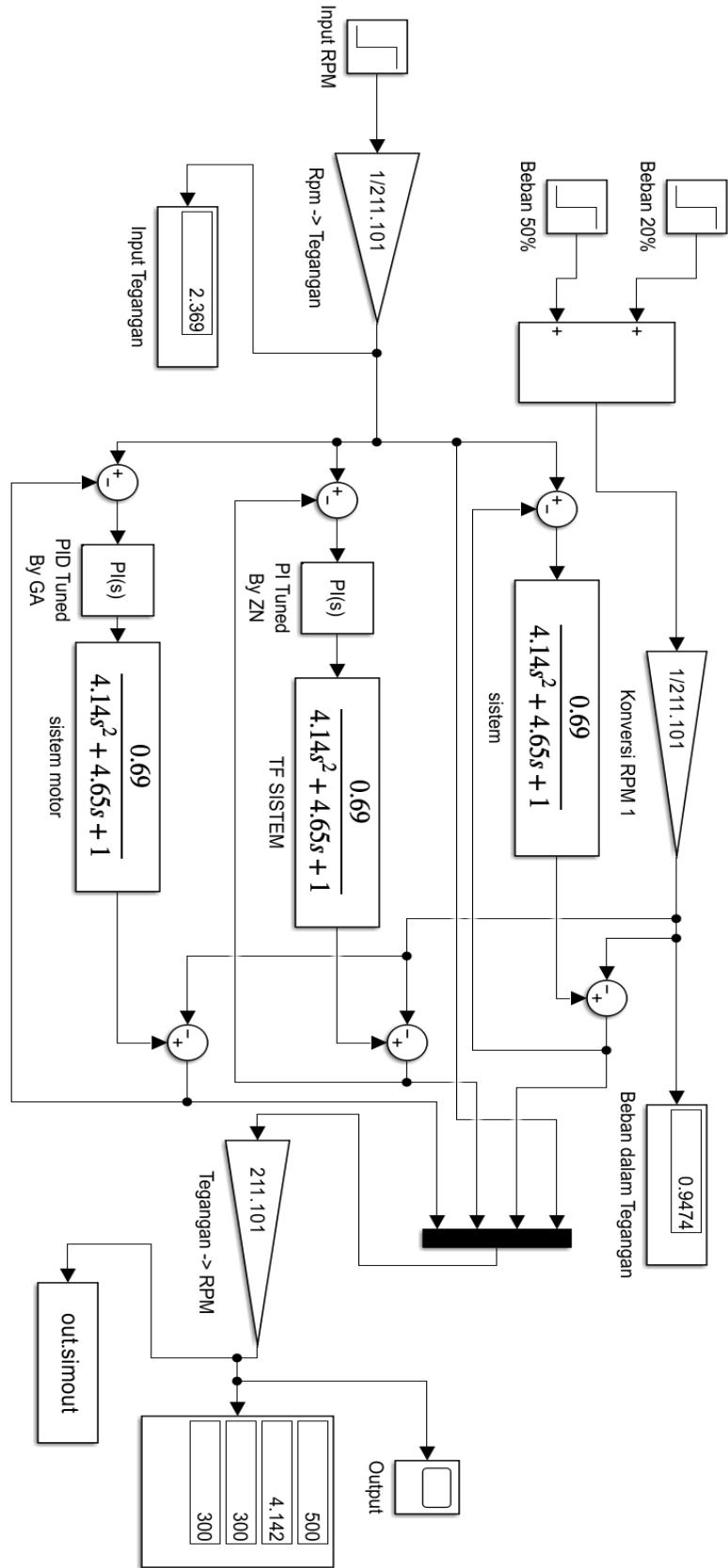
function [child] = geneMutation(parent, mutRate, LB, UB)
nVars = numel(parent);
nmu = ceil(mutRate * nVars);
j = randsample(nVars, 1);
child = parent;
child(j) = parent(j) - rand*parent(j);
end

```

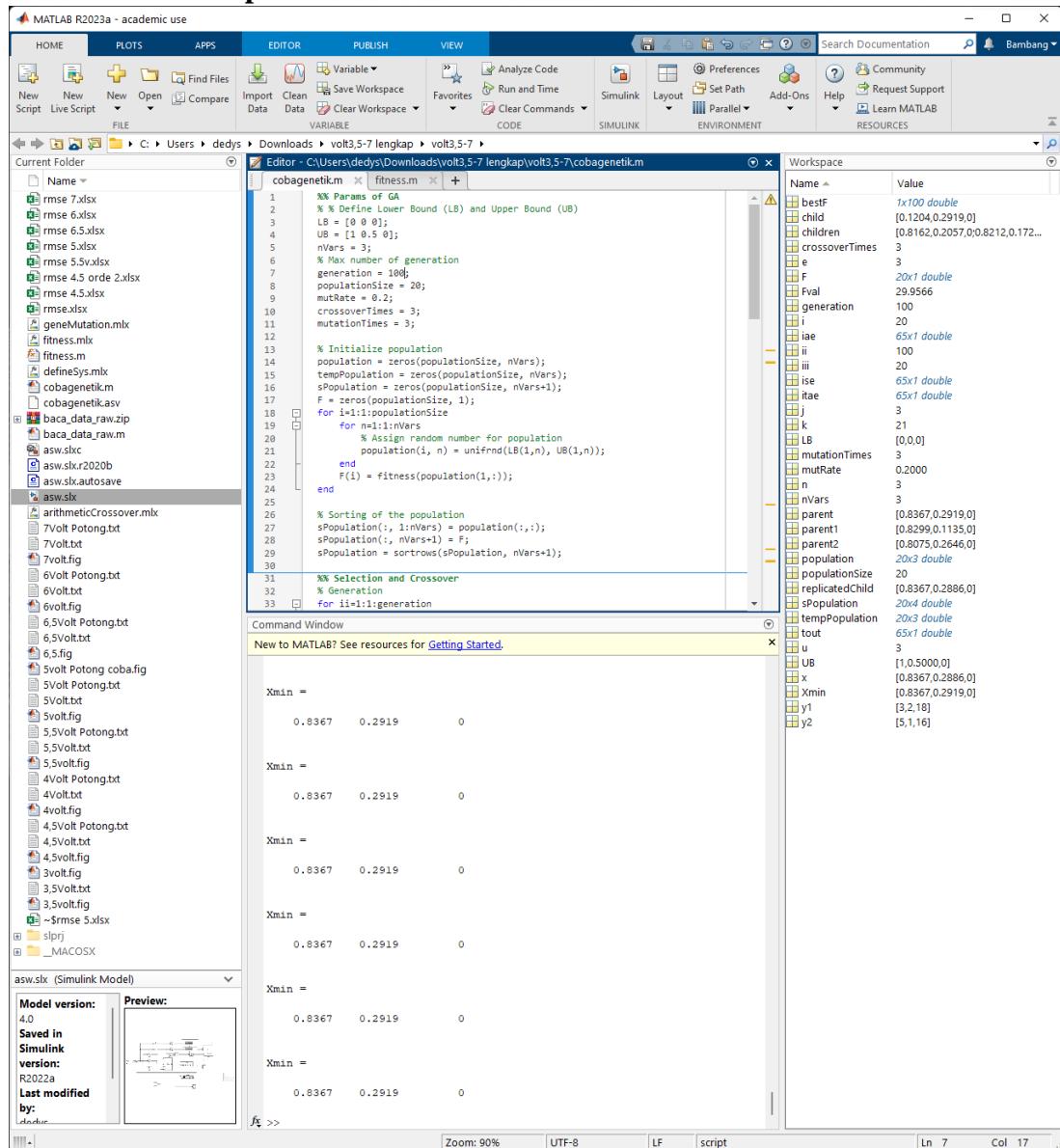
7. Simulink PI – GA



8. Simulink Tes Hasil Kontroler dan Tanpa Kontroler



9. Dokumentasi Workspace Simulasi PI – GA Pada Matlab



Halaman ini sengaja dikosongkan

BIODATA PENULIS



Bambang Permadi Widyasmara dilahirkan di Surabaya, 28 Juli 2001, merupakan anak kedua dari tiga bersaudara. Saya telah menempuh pendidikan formal yaitu di SD Godwins School, SMP dan SMA di sekolah tiga Bahasa Xin Zhong School Surabaya. Setelah saya lulus SMA di tahun 2019, saya mengikuti program penerimaan mahasiswa di ITS dengan jurusan Teknik Elektro FTEIC dan diterima di tahun yang sama. Saya terdaftar sebagai mahasiswa ITS dengan NRP 07111940000222.

Di Departemen Teknik Elektro penulis aktif di beberapa kegiatan kampus seperti mengikuti Ukafo, Click ITS, dan jug Astronomi ITS. Dalam kegiatan departemen Teknik Elektro, penulis mengikuti panitia Akotrans pada acara Electra 2020.

Kontak email: dedysby48@gmail.com