OPTIMISASI *AUTOMATIC VOLTAGE REGULATOR* (AVR) BERBASIS PID MENGGUNAKAN METODE HYBRID GREY WOLVE OPTIMIZATION – GENETIC ALGORITHM (HGWGA)

TESIS

PROGRAM MAGISTER TEKNIK ELEKTRO

MINAT SISTEM KONTROL DAN ELEKTRONIKA

Ditujukan untuk memenuhi persyaratan

memperoleh gelar Magister Teknik

ANAS SETIAWAN

NIM. 176060300111014

UNIVERSITAS BRAWIJAYA

FAKULTAS TEKNIK

MALANG

2021

TESIS

OPTIMISASI AUTOMATIC VOLTAGE REGULATOR (AVR) BERBASIS PID MENGGUNAKAN METODE HYBRID GREY WOLVE OPTIMIZATION - GENETIC ALGORITHM

ANAS SETIAWAN
NIM. 176060300111014

telah dipertahankan di depan penguji
pada Tanggal **12 Juli 2021**
dinyatakan telah memenuhi syarat
untuk memperoleh gelar **Magister Teknik**

Komisi Pembimbing

Pembimbing I	Pembimbing II
Dr.Eng. Panca Mudjirahardjo, S.T., M.T.	**Ir. Wijono, M.T., Ph.D.**

Malang, 28 JUL 2021
Universitas Brawijaya
Fakultas Teknik, Jurusan Teknik Elektro
Plt. Ketua Jurusan Teknik Elektro

M. Aziz Muslim, S.T., M.T., Ph.D.
NIP. 197412032000121001

PERNYATAAN ORISINALITAS TESIS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan Saya dan berdasarkan hasil penelusuran berbagai karya ilmiah, gagasan dan masalah ilmiah yang diteliti dan diulas di dalam Naskah Tesis ini adalah asli dari pemikiran Saya. Tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu Perguruan Tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata di dalam naskah Tesis ini dapat dibuktikan terdapat unsur-unsur jiplakan, Saya bersedia Tesis dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 tahun 2003, Pasal 25 Ayat 2 dan Pasal 70).

Malang, 26 Juli 2021

Mahasiswa,

Nama : **ANAS SETIAWAN**

NIM : **176060300111014**

PM : **TEKNIK ELEKTRO**

PROGRAM MAGISTER TEKNIK ELEKTRO

JUDUL TESIS :

OPTIMISASI *AUTOMATIC VOLTAGE REGULATOR* (AVR) BERBASIS PID MENGGUNAKAN METODE HYBRID GREY WOLVE OPTIMIZATION – GENETIC ALGORITHM (HGWGA)

Nama Mahasiswa	:	Anas Setiawan
NIM	:	176060300111014
Program Studi	:	Program Magister Teknik Elektro
Minat	:	Sistem Kontrol Elektronika

KOMISI PEMBIMBING :

Ketua	:	Dr. Eng. Panca Mudjirahardjo, S.T., M.T.
Anggota	:	Ir. Wijono, M.T., Ph.D.
SK Pembimbing	:	2591 Tahun 2018

TIM DOSEN PENGUJI :

Dosen Penguji 1	:	M. Aziz Muslim, S.T., M.T., Ph.D.
Dosen Penguji 2	:	Dr. Ir. Dipl.Ing. Mochammad Rusli
Tanggal Ujian	:	12 July 2021
SK Penguji	:	1198 Tahun 2021

RINGKASAN

Anas Setiawan, Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya, July 2021, *Optimisasi Automatic Voltage Regulator (AVR) Berbasis PID Menggunakan Metode Hybrid Grey Wolve Optimization – Genetic Algorithm (HGWGA)*, Dosen Pembimbing : Dr.Eng. Panca Mudjirahardjo, S.T., M.T., dan Ir. Wijono, M.T., Ph.D.

Konsumsi akan listrik sudah menjadi sebuah kebutuhan primer bagi semua lini kehidupan masyarakat. Mulai dari rumah tangga sampai industri sangat bergantung pada ketersediaan energi listrik. Dengan adanya energi listrik, manusia sangat terbantu dan dimudahkan dari berbagai aspek kehidupan. Dimulai dari yang paling sederhana seperti pencahayaan rumah dan jalan, sampai pada percepatan sistem produksi pada sebuah industri. Sedangkan dalam dunia industri, penggunaan energi listrik dapat meningkatkan produktifitas dan efisiensi dalam sistem produksi. Peningkatan konsumsi energi listrik di Indonesia dipengaruhi oleh pertumbuhan ekonomi dan bertambahnya jumlah penduduk

Pemenuhan kebutuhan energi listrik tidak hanya dinilai dari segi kuantitas pembangkitannya, akan tetapi juga secara kualitas atau mutu energi listrik tersebut. Mutu energi listrik dapat dinilai dari kestabilan frekuensi listrik, tegangan listrik, harmonisa dan kontinyuitas pelayanan energi.

Pada *generator set* (genset), sistem kestabilan tegangan dipengaruhi oleh sistem eksitasi yang dikontrol oleh rangkaian kontrol yang disebut AVR (*Automatic Voltage Regulator*). Salah satu komponen penting dalam sistem AVR adalah algoritma dari kontroler. Penggunaan metode kontrol PID telah banyak digunakan dalam perancangan kontroler AVR. Penelitian ini menggunakan metode *hybrid* GWO-GA (*Grey Wolf Optimization – Genetic Algorithm*) dalam penalaan parameter PID. Hasil respon transient *automatic voltage regulator* (AVR) terbaik didapatkan ketika menggunakan metode *hybrid genetic algorithm – grey wolf optimization* (HGAGW) dengan nilai *fitness* sebesar 4.3039, metode *Grey wolf optimization* (GWO) dengan nilai *fitness* sebesar 4.5059, dan metode *genetic algorithm* (GA) dengan nilai *fitness* sebesar 6.0214.

Kata Kunci : Generator set, AVR, PID, hybrid GWO-GA

SUMMARY

Anas Setiawan, *Department of Electrical Engineering, Faculty of Engineering, University of Brawijaya, December 2019, Automatic Voltage Regulator (AVR) Optimization Based on PID Using the Hybrid Grey Wolf Optimization - Genetic Algorithm (HGWGA) Method, Academic Supervisor* : Dr.Eng. Panca Mudjirahardjo, S.T., M.T., and Ir. Wijono, M.T., Ph.D.

Consumption of electrical energy has become a primary need for all lines of community life. Starting from households to industry, it is highly dependent on the availability of electrical energy in the industrial world, the use of electrical energy can increase productivity and efficiency in the production system. The electricity consumption increase in Indonesia is influenced by increasing population and economic growth

Indonesia is one of the electricity deficit countries. It can be seen from the data published by PLN, which states that the level of electrification in Indonesia is 88.3% in 2015 [2]. So that at the end of 2015, PLN made a construction program for electricity generation of 35,000 MW with an electrification target of 97.4% in 2019. Electrification in Indonesia still could not run optimally, one of the causes is Indonesia's geographical location consisting of islands. Due to the level of availability of electrical energy in Indonesia which is still deficit to the level of its needs, the use of generator set (genset) is still used as an alternative as a reserve of electrical energy sources for many consumers.

In the generator set (genset), the voltage stability system is affected by the excitation system controlled by control circuit called AVR (Automatic Voltage Regulator). One of the important components in the AVR system is the algorithm of the controller. The application of the PID control method has been widely used in the design of AVR controllers. This study applies the GWO-GA (Grey Wolf Optimization - Genetic Algorithm) hybrid method on PID parameters setting. The best transient automatic voltage regulator (AVR) response results were obtained when using the hybrid genetic algorithm - grey wolf optimization (HGAGW) method with a fitness score of 4.3039, the Grey wolf optimization (GWO) method with a fitness score of 4.5059, and the genetic algorithm (GA) method with a fitness score of 6.0214.

Keywords **:** *Generator set, AVR, PID, hybrid GWO-GA*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Konsumsi akan listrik sudah menjadi sebuah kebutuhan primer bagi semua lini kehidupan masyarakat. Mulai dari rumah tangga sampai industri sangat bergantung pada ketersediaan energi listrik. Dengan adanya energi listrik, manusia sangat terbantu dan dimudahkan dari berbagai aspek kehidupan. Dimulai dari yang paling sederhana seperti pencahayaan rumah dan jalan, sampai pada percepatan sistem produksi pada sebuah industri. Sedangkan dalam dunia industri, penggunaan energi listrik dapat meningkatkan produktifitas dan efisiensi dalam sistem produksi. Peningkatan konsumsi energi listrik di Indonesia dipengaruhi oleh pertumbuhan ekonomi dan bertambahnya jumlah penduduk (Teuku, 2017)

Indonesia merupakan salah satu negara yang defisit elektrifikasi tenaga listriknya. Hal ini dapat diketahui dari data yang dipublikasi oleh PLN, yang menyatakan bahwa tingkat elektrifikasi di Indonesia sebesar 88,3% pada tahun 2015 (PLN, 2015). Sehingga pada akhir tahun 2015, PLN membuat program pembangunan pembangkit energi listrik sebesar 35.000 MW dengan target elektrifikasi sebesar 97,4% pada tahun 2019. Elektrifikasi di Indonesia yang masih belum maksimal salah satunya dikarenakan geografis indonesia yang memiliki pulau-pulau yang tersebar. Dikarenakan tingkat ketersediaan energi listrik di Indonesia yang masih defisit terhadap tingkat kebutuhannya, penggunaan generator set (genset) masih dijadikan pilihan alternatif utama sebagai cadangan sumber energi listrik bagi banyak konsumen.

Pemenuhan kebutuhan energi listrik tidak hanya dinilai dari segi kuantitas pembangkitannya, akan tetapi juga secara kualitas atau mutu energi listrik tersebut. Mutu energi listrik dapat dinilai dari kestabilan frekuensi listrik, tegangan listrik, harmonisa dan kontinyuitas pelayanan energi. Kestabilan tegangan menjadi perhatian khusus bagi negara-negara maju seperti Perancis, Jepang dan USA. Stabilitas kualitas listrik menyebabkan stabilitas sistem tenaga secara total (Endriyanto,2011). Selain itu, pada sisi konsumen, ketidakstabilan tegangan dapat mengakibatkan kerusakan pada perangkat-perangkat elektronik yang sensitif terhadap fluktuasi tegangan.

Pada generator set (genset), sistem kestabilan tegangan dipengaruhi oleh sistem eksitasi yang dikontrol oleh rangkaian kontrol yang disebut AVR (*Automatic Voltage*

Regulator). AVR memiliki tugas menjaga profil tegangan terminal pada suatu nilai titik operasi tertentu, misalnya 220V, 380V, 13.8kV dan lain-lain. Fungsi AVR yang lain berkaitan dengan aksi kontrol regulasi daya reaktif dan pengaturan osilasi rotor jika terjadi gangguan. Gangguan osilasi rotor salah satunya dikarenakan beban energi listrik yang berubah-ubah, sehingga peran AVR sangat vital dalam menjaga stabilitas tegangan pada terminal keluaran generator set (genset). Berdasarkan prinsip kerjanya, AVR akan mengatur nilai arus eksitasi pada exciter, dimana nilai arus eksitasi berpengaruh pada nilai tegangan pada terminal keluaran generator set. Semakin tinggi nilai arus eksitasi, maka nilai tegangan pada terminal keluaran generator set juga akan semakin tinggi dan begitupun juga sebaliknya. Secara umum, komponen penyusun dari sistem AVR adalah kontroller, amplifier, exciter, generator dan sensor.

Salah satu komponen penting dalam sistem AVR adalah algoritma dari kontroller. Algoritma dari sebuah kontroller mempunyai metode kontrol yang berbasis sistem loop tertutup (*closed loop*). Algoritma kontroller berfungsi untuk memproses sinyal dari sensor yang kemudian diolah menjadi sebuah nilai keluaran untuk mengatur arus eksitasi pada exciter. Penggunaan metode kontrol dalam sistem AVR mempunyai tujuan untuk mengatasi problem stabilitas tegangan pada beban nonlinier dan juga dapat mengoptimasi nilai respon waktu (*time response*) (Ahmed dkk, 2019).

Metode kontrol *Proportional Integral Dervative* (PID) telah banyak diimplementasikan dalam sistem industri seperti chemical proses, biomedical proses, thermal proses dan masih banyak yang lainnya. Kelebihan dari metode kontrol PID yaitu pengaplikasian yang relatif mudah dan memiliki performa yang relatif stabil (Busra, 2019). Berdasarkan hal tersebut, penggunaan metode kontrol PID telah banyak digunakan dalam perancangan kontroller AVR. Optimalisasi metode kontrol PID dilakukan melalui penalaan parameter Kp, Ki, Kd. Metode penalaan parameter PID yang dilakukan secara konvensional menggunakan metode *Ziegler-Nichols* tidak memberikan performa optimum pada plan (Ziegler, 1942). Sehingga telah banyak dilakukan penelitian untuk menemukan metode penalaan yang optimum menggunakan Artificial Intelligence (AI).

Beberapa penelitian telah dilakukan dalam penentuan parameter PID, diantaranya dengan penggunaan metode *Artificial Intelligent* (AI). Penelitian yang pernah dilakukan yakni penggunaan metode *GWO* (GWO) berbasis PID kontroller pada sistem AVR (*Regulator tegangan otomatis*) (Ziegler, 1942). Berdasarkan penelitian dengan menggunakan metode GWO didapatkan respon transien yang masih

memiliki *overshoot* sebesar 1,1301 V dan *settling time* sebesar 0,7739 S. Pada penelitian ini, peneliti akan menggunakan metode *hybrid* GWO-GA (*GWO – GA*) dalam penalaan parameter PID dengan harapan dapat memperbaiki respon trasien dari sistem.

1.2 Rumusan Masalah

Mengacu pada latar belakang penelitian dan pengembangan dari penelitian sebelumnya, maka permasalahan yang akan dibahas meliputi:

1. Bagaimana desain kontroller sistem *Regulator tegangan otomatis* (AVR) berbasis PID dengan menggunakan metode penalaan parameter *HGWGA*?
2. Bagaimana respon transient sistem *regulator tegangan otomatis* (AVR) ketika menggunakan optimasi *GA* (GA), *grey wolve optimization* (GWO), *hybrid GA – grey wolve optimization* (HGAGW)?
3. Bagaimana perbandingan respon transient ketika menggunakan optimasi *GA* (GA), *grey wolve optimization* (GWO) dan *hybrid GA – grey wolve optimization* (HGAGW)?

1.3 Batasan Masalah

Supaya didapatkan hasil penelitian yang optimal dan sesuai dengan harapan, maka peneliti menetapkan batasan masalah penelitian. Adapun batasan masalah penelitian meliputi:

1. Sistem kendali *Regulator tegangan otomatis* (AVR) yang diteliti adalah sistem AVR pada generator sinkron.
2. Metode kontrol loop tertutup (*closed loop*) yang digunakan dalam *Regulator tegangan otomatis* (AVR) menggunakan metode PID.
3. Metode penalaan parameter Kp, Ki, Kd dalam PID menggunakan metode *HGWGA*.
4. Parameter performansi dalam penelitian ini meliputi M_p(*Maksimum Overshoot*), E_{ss} (*Error Steady State*), t_r (*Time Rise*) dan t_s (*Time Settling*).
5. Analisa perbandingan performansi difokuskan pada *regulator tegangan otomatis* (AVR) ketika menggunakan kontroller PID dan tanpa kontroller.

6. Analisa perbandingan performansi pada penalaan parameter PID difokuskan antara metode *GWO* (GWO), *GA* (GA) dan *hybrid GWO – GA* (HGWGA).

1.4 Tujuan Penelitian

Mengacu pada rumusan masalah diatas, tujuan penelitian yang ingin dicapai meliputi:

1. Menemukan desain kontroller sistem *Regulator tegangan otomatis* (AVR) berbasis PID dengan menggunakan metode penalaan parameter *HGWGA.*
2. Mengetahui pengaruh implementasi metode PID terhadap respon transient sistem *Regulator tegangan otomatis* (AVR).
3. Mengetahui pengaruh implementasi penalaan parameter PID menggunakan metode *HGWGA* terhadap respon transient sistem *Regulator tegangan otomatis* (AVR).

1.5 Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan manfaat diantaranya mampu memberikan kontribusi pada aspek teoritis maupun praktis terhadap perkembangan ilmu pengetahuan. Khususnya pada bidang sistem kontrol dan otomasi. Adapun beberapa manfaat penelitian ini meliputi:

1. Menemukan desain kontroller sistem *Regulator tegangan otomatis* (AVR) yang optimum berdasarkan metode kontrol PID dengan penalaan parameter menggunakan metode *Hybrid GWO – GA* (HGWGA).
2. Sebagai sarana referensi dalam pengembangan metode penelitian sistem kontrol *Regulator tegangan otomatis* (AVR) ke depannya agar diperoleh pengembangan yang berkelanjutan.
3. Sebagai sarana referensi dan acuan bagi dunia industri dalam membuat sistem kontrol *Regulator tegangan otomatis* (AVR).

BAB II
LANDASAN TEORI

2.1. Tinjauan Pustaka

Regulator tegangan otomatis (AVR) secara umum terdiri dari empat komponen utama, yakni *amplifier*, *exciter*, *synchronous generator* dan Sensor. Dalam setiap *generator set* (genset) terdapat AVR sebagai salah satu komponen penyusunnya. Tujuan utama dari *regulator tegangan otomatis* (AVR) dalam sistem *generator set* (genset) adalah sebagai manipulator agar tegangan keluaran pada terminal *synchronous generator* terjaga dan stabil pada nilai tertentu. Salah satu faktor yang berpengaruh akan kestabilan tegangan pada terminal *synchronous generator* adalah fluktuasi beban. Ketika nilai beban semakin tinggi, maka akan terjadi penurunan tegangan pada sisi terminal *synchronous generator* dikarekan efek pembebanan. Begitu juga sebaliknya ketika nilai beban semakin kecil, maka akan terjadi kenaikan tegangan pada sisi terminal *synchronous generator*. Oleh sebab itu diperlukan AVR sebagai regulator tegangan supaya nilai tegangan pada terminal *synchronous generator* tetap stabil meskipun terjadi perubahan nilai beban yang fluktuatif.

Berkaitan dengan pentingnya fungsi dan peran *Regulator tegangan otomatis* (AVR) dalam sistem pembangkitan tenaga listrik dalam hal ini *generator set* (genset), maka telah banyak dilakukan penelitian untuk mencari metode kontrol yang efektif dan optimal. Metode kontrol PID banyak digunakan dalam sistem kontrol AVR dikarenakan memiliki struktur yang sederhana dan memliki performa yang stabil terhadap gangguan (*disturbance*) (Busra, 2019). Optimalisasi metode kontrol PID sangat bergantung pada penalaan nilai parameter Kp, Ki dan Kd. Penentuan parameter PID mempunyai beberapa metode mulai dari metode konvensional sampai lanjut (*advanced*). Penggunaan metode lanjut (*advanced*) banyak menggunakan algoritma kecerdasan buatan (*Artifial Intelligence*).

Penelitian sebelumnya yang dilakukan oleh Rvindra Kumar Kuri (2019) dengan judul desain kontroller sistem AVR menggunakan metode *Grey Wolfe Optimization* (GWO) berbasis PID didapatkan hasil *step response* transien dengan nilai *overshoot* sebesar 1,1301V dan *settling time* sebesar 0,7739 *second*.

Penelitian yang dilakukan oleh Faouzi Aboura (2019) dengan judul penalaan PID kontroller menggunakan metode *Hybrid GA Particle Swarm Optimization* pada sistem AVR didapatkan hasil perbandingan antara metode *GA* (GA), *Particle Swarm*

Optimization (PSO) dan *Hybrid GA Particle Swarm Optimization* (HGAPSO). Berdasarkan hasil pengujiannya didapatkan *time settling* terbaik ketika menggunakan metode HGAPSO dengan nilai 0,7385 *second*. Sedangkan ketika menggunakan metode GA dan PSO memiliki time settling sebesar 2,9315 *second* dan 0,9238 *second*.

2.2. Generator Sinkron

Generator sinkron adalah suatu mesin listrik yang digunakan untuk memproduksi energi listrik dari sumber mekanikal dengan menggunakan induksi elektromagnetik. Energi mekanik diperoleh dari putaran rotor yang digerakkan oleh penggerak mula, sedangkan energi listrik diperoleh dari perpotongan medan magnet dengan penghantar, maka pada penghantar akan timbul gaya gerak listrik melalui proses induksi elektromagnetik yang terjadi pada kumparan rotor dan stator. Perubahan energi ini terjadi karena adanya pergerakan relatif antara medan magnet dengan kumparan generator. Pergerakan relatif merupakan terjadinya perubahan medan magnet pada kumparan jangkar (tempat terbangkitnya tegangan pada generator) karena pergerakan medan magnet terhadap kumparan jangkar.

Dikatakan generator sinkron karena kecepatan putaran medan magnet sama dengan kecepatan putaran rotor generator, sehingga kecepatan sinkron dihasilkan dari kecepatan putar rotor dengan kutub-kutub magnet yang berputar dengan kecepatan yang sama dengan medan putar pada stator. Kumparan medan generator sinkron terdapat pada rotor, sedangkan kumparan jangkar terdapat pada stator. Rotor generator sinkron yang terdiri dari belitan medan yang suplai dengan arus searah akan menghasilkan medan magnet yang berputar dengan kecepatan yang sama dengan kecepatan putar rotor. Karena kecepatan putaran medan magnet sama dengan kecepatan putaran rotor generator, maka generator sinkron ini akan menghasilkan energi listrik bolak balik (AC). Hubungan antara kecepatan putar dengan frekuensi ditunjukkan pada persamaan di bawah ini:

$$f = \frac{n.p}{120} \quad (1)$$

Dimana : $f = Frekuensi\ (Hz)$

$n = Kecepatan\ Putar\ (Rpm)$

$p = Jumlah\ Kutub$

Frekuensi adalah banyaknya siklus (gelombang) dalam setiap detik (s). Standar frekuensi listrik di Indonesia adalah 50 Hz. Oleh karena itu apabila generator unit

pembangkit diputar oleh turbin dengan kecepatan 3000 rpm, maka jumlah kutub magnetnya adalah 2 pasang. Jumlah kutub magnet suatu generator ditentukan berdasarkan putaran kerja dan frekuensi generator yang dinginkan. Frekuensi listrik harus dijaga konstan sepanjang waktu, karena perubahan frekuensi akan menyebabkan berubahnya putaran motor. Indikator kualitas listrik yang baik salah satunya ditunjukkan dengan frekuensi yang stabil.

2.2.1. Prinsip Kerja Generator Sinkron

Prinsip kerja alternator menerapkan prinsip pembangkitan listrik berdasarkan induksi. Menurut hukum faraday, apabila kumparan berputar di dalam medan magnet atau sebaliknya medan magnet berputar di dalam kumparan, maka pada ujung-ujung kumparan tersebut akan timbul gaya gerak listrik (GGL) atau timbul tegangan. Besar tegangan yang diinduksikan pada kumparan medan sangat bergantung pada panjang penghantar dalam kumparan medan, kecepatan putaran dan kuat medan magnet.

Suatu mesin listrik akan bekerja apabila memiliki kumparan medan yang berfungsi untuk menghasilkan medan magnet dan kumparan jangkar untuk menghasilkan ggl induksi pada konduktor yang terdapat pada medan jangkar, serta celah udara yang berfungsi untuk memungkinkan berputarnya jangkar dalam medan magnet.

Secara umum prinsip kerja dari generator sinkron adalah apabila kumparan medan yang terdapat pada rotor dihubungkan dengan sumber eksitasi yang akan mensuplai arus searah (DC) terhadap kumparan medan, maka dengan adanya arus searah yang mengalir melalui kumparan medan akan menimbulkan fluks. Penggerak mula (prime mover) yang sudah terkopel dengan rotor generator segera dioperasikan, sehingga rotor akan berputar dengan kecepatan tertentu sesuai dengan jumlah putaran yang diharapkan. Perputaran dari rotor generator tersebut akan sekaligus memutar medan magnet yang dihasilkan oleh kumparan medan rotor. Medan putar yang terdapat pada rotor tersebut, selajutnya akan diinduksikan pada kumparan jangkar, sehingga kumparan jangkar yang terdapat pada stator generator akan menghasilkan fluks magnetik yang berubah-ubah nilainya setiap waktu.

Adanya perubahan fluks yang terdapat suatu kumparan medan akan menimbulkan ggl induksi pada ujung kumparan medan tersebut sesuai dengan persamaan berikut:

$$E_{ind} = -N\frac{d\theta}{dt} \quad (2)$$

$$E = -N\frac{d\theta_{maks}Sin\omega t}{dt} \quad (3)$$

$$= -N\omega\theta_{maks}Cos\omega t\ ;\ (\omega = 2\pi f)$$

$$= -N(2\pi f)\theta_{maks}Cos\omega t\ ;\ (f = \frac{n.p}{120})$$

$$= -N(2\pi\frac{n.p}{120})\theta_{maks}Cos\omega t$$

$$= -N(2\ x\ 3{,}14\ x\ \frac{n.p}{120})\theta_{maks}Cos\omega t$$

$$E_{maks} = N\ (2\ x\ 3{,}14\ x\ \frac{n.p}{120})\theta_{maks}$$

$$E_{eff} = \frac{e_{maks}}{\sqrt{2}} = \frac{N\ (\ 2\ x\ 3{,}14\ \frac{n.p}{120})\theta_{maks}}{\sqrt{2}}$$

$$E_{maks} = \frac{4{,}44\ .Npn\theta}{120}\ ;\ \left(\frac{4{,}44Np}{120} = C\right) \quad (4)$$

$$E_{eff} = Cn\theta \quad (5)$$

Dimana: $E = ggl\ induksi\ (volt)$

$N = Jumlah\ Lilitan$

$C = Konstanta$

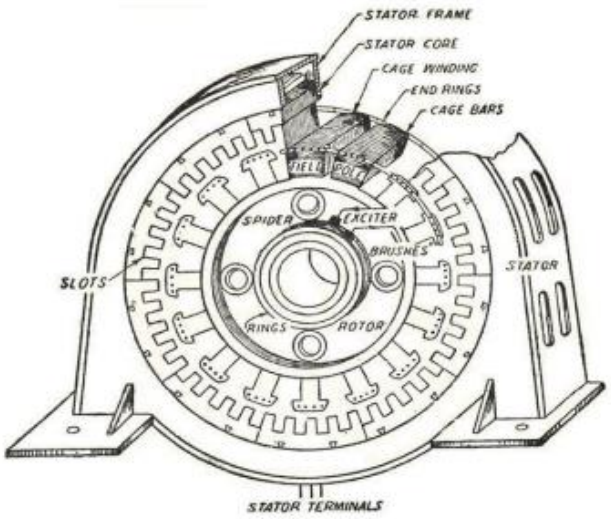
$n = Putaran\ Sinkron\ (rpm)$

$\varphi = Fluks\ Magnetik(Weber)$

2.2.2. Konstruksi Generator Sinkron

Generator sinkron merupakan komponen utama dalam sistem pembangkitan yang berfungsi untuk merubah energi mekanik menjadi energi listrik. Kapasitas generator dari waktu ke waktu berkembang semakin besar dengan teknologi

konstruksi dan rancang bangun yang semakin maju. Kapasitas generator pembangkitan di Indonesia sangat bervariasi, karena pembangunannya disesuaikan dengan kebutuhan energi yang harus dilayani. Konstruksi generator sinkron semuanya menggunakan medan magnet putar stator. Hal ini bertujuan untuk memudahkan penyambungan (*connection*) energi listrik keluar generator, karena titik terminal penyambungannya terdapat pada stator generator. Dapat dilihat pada gambar 2.1 konstruksi sederhana dari sebuah generator sinkron sebagai berikut:



Gambar 2.1 Konstruksi Generator Sinkron

Secara umum konstruksi generator sinkron sama dengan motor sinkron, dimana konstruksi generator sinkron terdiri dari rotor, stator dan celah udara. Rotor yaitu bagian yang berputar dalam suatu generator dimana kumparan medan disuplai arus searah dari eksitasi. Stator yaitu bagian dari generator sinkron yang diam, dimana akan menghasilkan ggl induksi pada konduktor yang terdapat pada medan jangkar. Celah udara adalah ruang antara rotor dan stator yang berfungsi sebagai tempat terjadinya fluks atau induksi energi listrik dari rotor ke stator dan memungkinkan berputarnya jangkar dalam medan magnet.

1. Rotor

Rotor berfungsi sebagai tempat ketika medan magnet dibangkitkan, dimana belitan medan magnet tersebut akan membentuk kemagnetan antara listrik kutub utara dan selatan yang terjadi pada inti rotor. Terdapat beberapa komponen utama dalam rotor generator yaitu:

a. Slip Ring

Slip ring berfungsi untuk mengaliri arus searah menuju medan magnet pada rotor. Slip ring ini terbuat dari bahan yang kuat dan tahan terhadap panas sehingga Slip ring ini mampu mengaliri arus ke rotor generator dengan baik. Kemudian Slip ring dipasangkan pada terminal kumparan rotor dan dihubungkan ke sumber arus searah menggunakan sikat (*brush*).

b. Kumparan Medan

Kumparan medan adalah kumparan medan terbuat dari tembaga berlapiskan perak yang dibuat dengan rapi. Kumparan ini berfungsi untuk menghasilkan medan magnet pada rotor yang mendapat sumber dari eksitasi.

c. Sikat

Pada umumnya generator sinkron memiliki dua jenis yaitu generator sinkron yang menggunakan sikat (*brush*) dan tanpa menggunakan sikat (*brushless*). Sikat pada generator sinkron berguna sebagai saklar putar untuk mengalirkan arus searah ke kumparan medan. Namun daya yang dihasilkan dengan menggunakan sikat pada generator ini sangat terbatas tidak sebesar seperti daya yang dihasilkan oleh generator yang menggunakan sistem *brushless*. Penggunaan sikat pada generator dengan daya yang besar dapat menimbulkan losses yang besar dan lonjatan api yang dapat menimbulkan kebakaran pada sistem pembangkit tersebut.

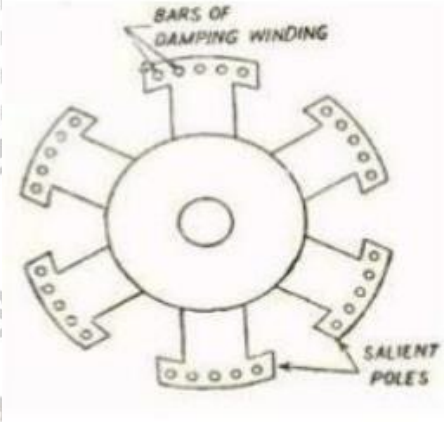
d. Poros Rotor

Poros rotor adalah sebagai tempat untuk meletakkan kumparan medan, dimana pada poros rotor generator tersebut berbentuk slot-slot secara paralel terhadap poros rotor. Pada dasarnya rotor generator sinkron adalah sebuah elektromagnet yang besar, dimana kutub medan magnet pada rotor generator sinkron berupa *salient pole* (kutub menonjol) dan *non-salient pole* (kutub tak menonjol).

- Kutub Menonjol (*Salient pole*)

Pada jenis *salient pole*, kutub magnet menonjol keluar dari permukaan rotor. Belitan-belitan medannya dihubung seri. Ketika belitan medan ini disuplai oleh eksiter, maka kutub yang berdekatan akan membentuk kutub berlawanan. *Salient pole* mempunyai jumlah

kutub yang banyak, hal ini ditandai dimana *salient pole* memiliki diameter yang besar dan panjang serta memiliki sumbu pendek. Kumparan pada *salient pole* ini dibelitkan pada tangkai kutub yang diberi laminasi untuk mengurangi panas berlebihan yang timbul karena arus Eddy. Bentuk dari *salient pole* (kutup menonjol) ditunjukkan pada gambar 2.2.



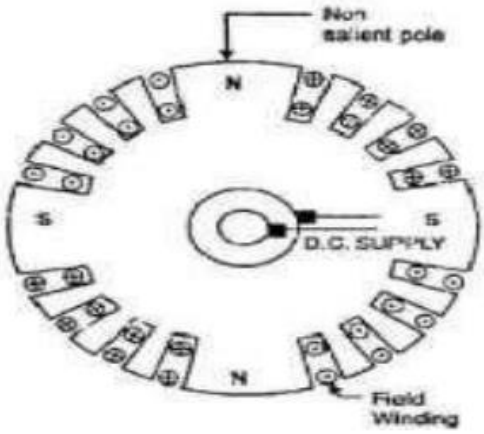
Gambar 2.2 *Rotor Salient pole*

Pada umumnya rotor *salient pole* (kutub menonjol) digunakan pada generator sinkron dengan kecepatan putaran yang rendah dan sedang (120-400 rpm). Generator sinkron menggunakan rotor *salient pole* seperti ini biasanya dikopel oleh mesin diesel atau turbin air pada sistem pembangkit listrik. Rotor *salient pole* sangat bagus untuk digunakan pada putaran rendah dan sedang, hal ini dikarenakan apabila digunakan pada kecepatan yang tinggi rotor ini dapat menimbulkan rugi-rugi yang besar, menimbulkan bersuara bising dan tidak cukup kuat untuk menahan tekanan mekanis apabila diputar dengan kecepatan tinggi.

- Rotor Kutub Silinder (*Non-Salient Pole Rotor*)

Pada jenis non-salient pole, rotor jenis ini terbuat dari plat baja yang berbentuk silinder dimana kontruksi medan magnetnya rata dengan permukaan rotor generator sinkron. Belitan medannya dipasang pada alur-alur sisi luar dan terhubung seri yang mendapatkan pasokan listrik

yang terhubung dengan exciter. Konstruksi dari rotor non-salient pole adalah dengan memberikan keseimbangan mekanis yang lebih baik dibandingkan rotor salient pole karena rugi-ruginya lebih kecil. Bentuk dari rotor non-salient pole generator sinkron ditunjukkan dalam gambar 2.3.

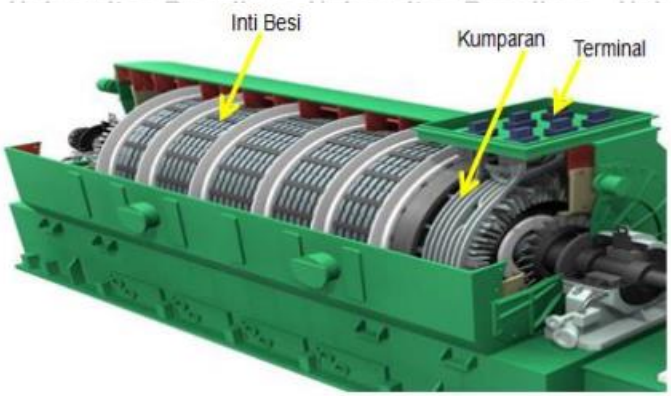


Gambar 2.3 Rotor Kutub *Non-Salient Pole*

Rotor kutub *non-salient pole* ini biasanya digunakan pada generator sinkron dengan kecepatan putar yang tinggi (lebih dari 1500 rpm) dengan diameter kecil dan panjang. Kumparan rotor diatur sedemikian sehingga terdapat fluks maksimum pada satu posisi tertentu. Rotor dengan bentuk ini biasanya lebih balance dan memiliki kebisingan (*noise*) yang rendah. Rotor silinder baik digunakan pada kecepatan putar tinggi karena konstruksinya memiliki kekuatan mekanik yang baik pada kecepatan putar tinggi. Selain itu distribusi di sekeliling rotor mendekati bentuk gelombang sinus sehingga lebih baik dari kutub menonjol.

2. Stator

Stator, bagian ini tersusun dari plat-plat stator yang mempunyai alur-alur sebagai tempat meletakkan lilitan stator. Lilitan ini berfungsi untuk membangkitkan ggl induksi. Konstruksi suatu stator dari generator sinkron ditunjukkan dalam gambar 2.4 berikut:



Gambar 2.4 Konstruksi Stator

a. Rangka Stator (*Stator Frame*)

Rangka stator merupakan sebagai tempat dari kumparan jangkar pada generator. Rangka stator berupa kerangka (rumah pembangkit) yang terbuat dari elemen plat baja yang dibentuk sedemikian rupa hingga diperoleh rangka stator yang sesuai dengan kebutuhan. Pemasangan rangka stator dilakukan dengan cermat agar diperoleh kedudukannya yang tepat dan mampu menahan hal-hal dan kondisi yang tidak menguntungkan baik pada saat gangguan seperti hubung singkat maupun gangguan bencana alam. Gambar dibawah memperlihatkan rangka stator pada generator.

b. Inti Stator

Inti stator merupakan tempat mengalirnya fluks magnet yang memotong kumparan jangkar di stator. Dimana inti stator terbuat dari laminasi-laminasi baja campuran atau besi magnetik khusus yang terpasang ke rangka stator. Tujuan dari laminasi-laminasi tersebut adalah untuk mengurangi besarnya arus pusar (eddy current), karena arus pusar ini dapat menimbulkan panas pada inti stator yang dapat merusak inti stator dan isolasi kumparan penghantar.

c. *Slot* (alur) dan Gigi

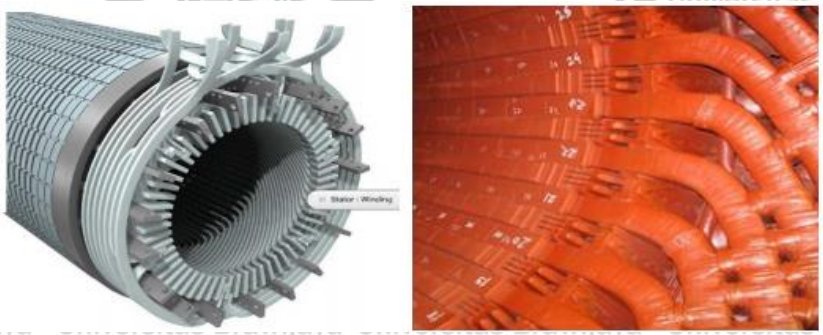
Slot (alur) dan gigi adalah tempat konduktor berada yang letaknya pada bagian dalam sepanjang keliling stator. Pada Slot (alur) dan gigi terdapat tiga bentuk yaitu, slot terbuka, slot setengah terbuka dan slot tertutup. Ketiga bentuk slot (alur) dan gigi tersebut ditunjukkan dalam gambar 2.5.

Gambar 2.5 Bentuk-bentuk Alur

d. Kumparan Stator (Kumparan Jangkar)

Kumparan jangkar adalah tempat terbentuknya GGL induksi yang diakibatkan adanya perpotongan medan magnet putar dari rotor yang memotong kumparan jangkar atau penghantar stator. Kumparan jangkar ini berupa gulungan kawat penghantar yang berisolasi yang disusun sedemikian rupa dan ditempatkan pada alur-alur inti besi.

Pada kumparan jangkar stator akan mengalirkan arus jangkar bolak – balik 3 fasa apabila pada kumparan tersebut terhubung dengan beban. Dimana arus tersebut akan menimbulkan panas pada kumparan yang dapat merusak isolasi kumparan jangkar dan memberi efek pemanasan pada inti besi. Kumparan tersebut dibagi menjadi 3 bagian yang berbeda fasa 1200 listrik. Dimana umumnya dihubungkan dengan sambungan bintang (Y) dan delta (Δ). Adapun gambar konstruksi kumparan jangkar stator ditunjukkan dalam gambar 2.6.



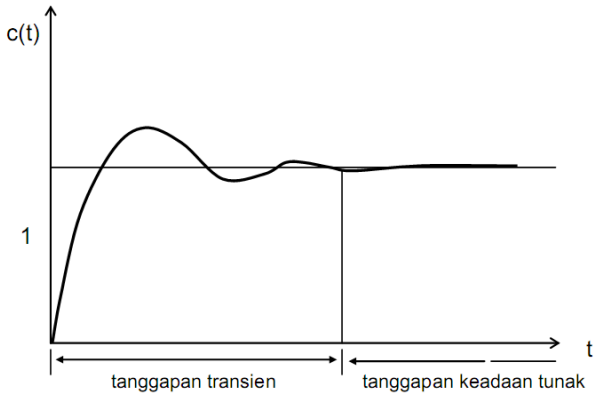
Gambar 2.6 Konstruksi Kumparan Stator

2.3. Tanggapan Transien

Tanggapan waktu dari suatu sistem kontrol dibagi menjadi dua bagian : tanggapan transien (*transient response*) dan tanggapan keadaan tunak (*steady-state response*). Tanggapan transien berlangsung dari saat mulai hingga tanggapan sistem mencapai nilai akhir yang diinginkan (*final state*). Tanggapan keadaan tunak dimulai pada saat tanggapan

mulai pertama kali mendekati nilai akhir hingga waktu yang tak terhingga. Gambar 2.7 mendeskripsikan kedua jenis tanggapan waktu tersebut.

Tanggapan transien digunakan untuk menganalisa sifat naik atau permulaan dari suatu sistem bila diberikan sinyal uji. Sedangkan tanggapan keadaan tunak digunakan untuk menganalisa karakteristik sistem pada saat mencapai harga akhirnya.



Gambar 2.7 Tanggapan Transien dan Tanggapan Keadaan Tunak

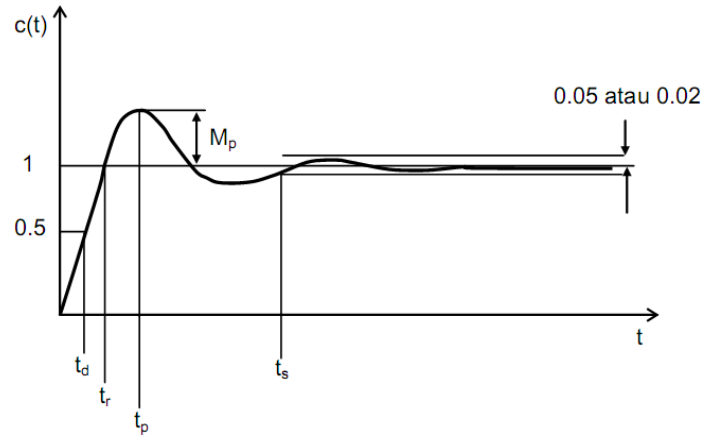
Adapun spesifikasi tanggapan transien dalam domain waktu memiliki beberapa jenis yakni:

a. Waktu Tunda (*delay time*), t_d:
   Adalah waktu yang dibutuhkan tanggapan sistem untuk mencapai setengah dari nilai akhir dari tanggapan untuk pertama kali.
b. Waktu Naik (*rise time*), t_r:
   Adalah waktu yang dibutuhkan untuk naik dari 10% – 90%, 5% – 95%, atau 0% – 100% dari nilai akhir dari tanggapan. Untuk kasus *underdamped*, biasanya digunakan kriteria 0% – 100%. Untuk kasus *overdamped*, biasanya digunakan kriteria 10% – 90%.
c. Waktu Puncak (peak time), t_p:
   Adalah waktu yang dibutuhkan tanggapan untuk mencapai nilai puncak dari *overshoot* pertama kali.
d. *Overshoot* maksimum (*maximum overshoot*), M_p:
   Adalah nilai puncak maksimum dari tanggapan diukur dari nilai akhir dari tanggapan. Biasanya dirumuskan dalam persentase :

$$M_p = \frac{c(t_p) - c(\infty)}{c(\infty)} x100\%$$

e. Waktu *settling* (*settling time*), t_s:
adalah waktu yang dibutuhkan tanggapan untuk mencapai nilai akhir dari tanggapan dan tetap berada pada nilai tersebut dalam range persentase tertentu dari nilai akhir (biasanya 5% atau 2%).

Spesifikasi tanggapan transien untuk masukan fungsi *unit-step* diberikan pada Gambar 2.8.



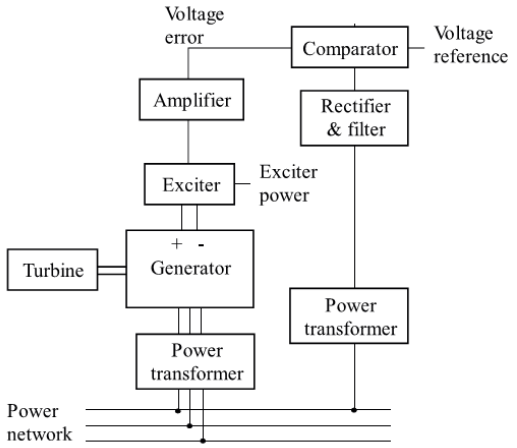
Gambar 2.8 Spesifikasi Tanggapan Transien Fungsi Unit-Step

2.4. *Regulator tegangan otomatis* (AVR)

Eksitasi atau biasa disebut sistem penguatan adalah suatu sistem yang memberikan arus penguat (*If*) kepada kumparan medan generator arus bolak-balik (*alternating current*) yang dijalankan dengan cara membangkitkan medan magnetnya dengan bantuan arus searah. Arus eksitasi adalah pemberian arus listrik pada kutub magnetik. Dengan mengatur besar kecilnya arus listrik tersebut kita dapat mengatur besar tegangan output generator atau dapat juga mengatur besar daya reaktif yang diinginkan pada generator yang dipasang paralel dengan sistem jaringan beban (*infinite bus*). Sistem eksitasi dapat dibagi menjadi dua jenis, yaitu sistem eksitasi dengan menggunakan sikat dan sistem eksitasi tanpa sikat. Eksitasi adalah sebuah dasar yang menjadikan pentingnya penggunaan AVR pada generator.

Prinsip kerja dari AVR adalah mengatur arus penguatan (*If*) pada exciter. Rangkaian AVR (*Regulator tegangan otomatis*) bekerja dengan mendeteksi tegangan keluaran dari generator utama dan menghasilkan suatu sinyal kendali yang sesuai dengan perubahan sinyal keluaran generator yang kemudian diteruskan ke rangkaian penyalaan untuk mengatur sudut penyalaan penyearah maka besar kecilnya arus eksitasi untuk generator

utama dapat diatur. Apabila tegangan output generator di bawah tegangan nominal, maka AVR memperbesar arus penguatan (*If*) pada exciter. Dan juga sebaliknya apabila tegangan pada generator melebihi tegangan nominal generator maka AVR akan mengurangi arus penguatan (*If*) pada exciter. Dengan demikian apabila terjadi perubahan pada tegangan keluaran generator dapat distabilkan oleh AVR secara otomatis. Hal ini dapat dilakukan karena AVR dilengkapi dengan kontroller yang berfungsi untuk menjaga stabilitas tegangan dari terminal keluaran generator. Secara umum sistem *Regulator tegangan otomatis* (AVR) ditunjukkan dalam Gambar 2.9.



Gambar 2.9 Model sistem *regulator tegangan otomatis* (AVR)

Pada sistem *regulator tegangan otomatis* (AVR) seperti yang ditunjukkan diatas dapat diketahui komponen penyusun utama dalam sistem AVR. Adapun penyusun utama dalam sistem *regulator tegangan otomatis* (AVR) adalah amplifier, exsitor, generator dan sensor. Amplifier berfungsi sebagai penguat sinyal yang dihasilkan oleh komponen komparator. Dalam hal ini, komparator berfungsi sebagai penghasil sinyal kontrol yang didapatkan melalui komparasi antara *voltage power network* (tegangan jaringan) dan *voltage reference* (tegangan referensi). Keluaran dari amplifier berupa arus eksitasi digunakan untuk mensupply eksiter (kumparan medan) pada generator. Dengan adanya arus eksitasi yang mengalir pada kumparan medan maka akan menimbulkan fluks elektromagnetik yang akan menginduksi kumparan jangkar. Pada terminal kumparan jangkar akan dihubungkan pada jaringan beban.

2.5. PID Kontroller Pada Sistem AVR

Stabilitas parameter daya (tegangan dan frekuensi) pada generator sinkron merupakan suatu hal yang sangat penting. Dimana hal tersebut bergantung pada performa dari *Regulator tegangan otomatis* (AVR) yang berfungsi meregulasi tegangan generator sinkron agar terjaga pada nilai tertentu. Sampai pada saat ini, kontroller PID masih banyak digunakan dalam sistem *Regulator tegangan otomatis* (AVR) dikarenakan kemudahan penggunaannya dan ekonomis (Ching-Chang, 2019).

Pengendali PID (*Proportional*, *Integral*, *Derivative*) merupakan suatu pengendali yang mampu memperbaiki tingkat akurasi dari suatu sistem plant yang memiliki karakteristik umpan balik / *feedback* pada sistem tersebut. Pengendali PID menghitung dan meminimalisasi nilai *error* / selisih antara output dari proses terhadap *input* / *setpoint* yang diberikan ke sistem. Pengendali PID terdiri dari tiga komponen yaitu *proportional*, *integral*, dan *derivative* yang dapat dipakai secara bersamaan maupun sendiri-sendiri tergantung dari respon yang diinginkan pada suatu plant. Penjelasan dari masing-masing komponen adalah sebagai berikut:

A. Pengendali Proportional

Pengendali proportional/gain bertindak sebagai penguat yang mampu mengubah output dari sistem secara proporsional tanpa memberikan efek dinamik pada kinerja pengendali tersebut. Pada pengendali proportional terdapat dua jenis yakni analog dan digital. Persamaan dari pengendali proporsional analog dan digital dinyatakan dalam persamaan:

$$P_{out} = K_c \, . \, E(t) \tag{6}$$

Dimana: $P_{out} = Output\ dari\ pengendali\ proportional$

$K_c = Konstanta\ Gain$

$E(t) = Error\ yang\ dinyatakan\ dalam\ waktu\ kontinu$

$$P_{out} = K_c \, . \, E_N \tag{7}$$

Dimana: $P_{out} = Output\ dari\ pengendali\ proportional$

$K_c = Konstanta\ Gain$

$E(t) = Error\ yang\ dinyatakan\ secara\ diskrit$

Pengaturan dengan pengendali proporsional ini mampu memperbaiki respon transien dari sistem, khususnya *rise time*. Pengendali ini juga mampu memperbaiki *settling time* dari sistem.

B. Pengendali Integral

Pengendali integral merupakan pengendali yang berfungsi untuk memperbaiki respon tunak / *steady state* dari sistem sehingga pengendali ini mampu memperkecil *error* sistem. Pada pengendali integral terdapat dua jenis yakni analog dan digital. Persamaan dari pengendali integral analog dan digital dinyatakan dalam persamaan:

$$I_{out} = K_i \int_0^t E(\tau) d\tau \quad (8)$$

Dimana: $I_{out} = Output\ pengendali\ integral$

$K_i = Konstanta\ Integral$

$E(\tau) = Error\ yang\ dinyatakan\ dalam\ waktu\ kontinu$

$\tau = Variabel\ integrasi$

$$I_{out} = \frac{K_c \Delta t}{T_I} \sum_{i=1}^{N} E_i \quad (9)$$

Dimana: $I_{out} = Output\ pengendali\ integral$

$T_i = Waktu\ Integral$

$\Delta t = Periode\ Sampling$

$E = Error\ yang\ dinyatakan\ secara\ diskrit$

Dengan mengatur nilai dari konstanta integral yang tepat, nilai error steady state dapat diperkecil dalam waktu yang lebih cepat sehingga nilai output akan lebih cepat mengikuti nilai set point.

C. Pengendali Derivative

Pengendali derivative merupakan suatu pengendali yang terutama berfungsi untuk memperbaiki respon transien dari sistem. Pada pengendali derivative terdapat dua jenis yakni analog dan digital. Persamaan pengendali derivative dapat dinyatakan dalam persamaan berikut:

$$D_{out} = K_d \cdot \frac{d}{dt} E(t) \quad (10)$$

Dimana: $D_{out} = Output\ dari\ pengendali\ derivative$

$K_d = Konstanta\ Derivative$

$E(t) = Error\ yang\ dinyatakan\ dalam\ waktu\ kontinu$

$$D_{out} = \frac{K_c T_d}{\Delta t}(E_N - E_{N-1}) \quad (11)$$

Dimana: $P_{out} = Output\ dari\ pengendali\ derivative$

$T_d = Waktu\ derivative$

$\Delta t = Waktu\ Sampling$

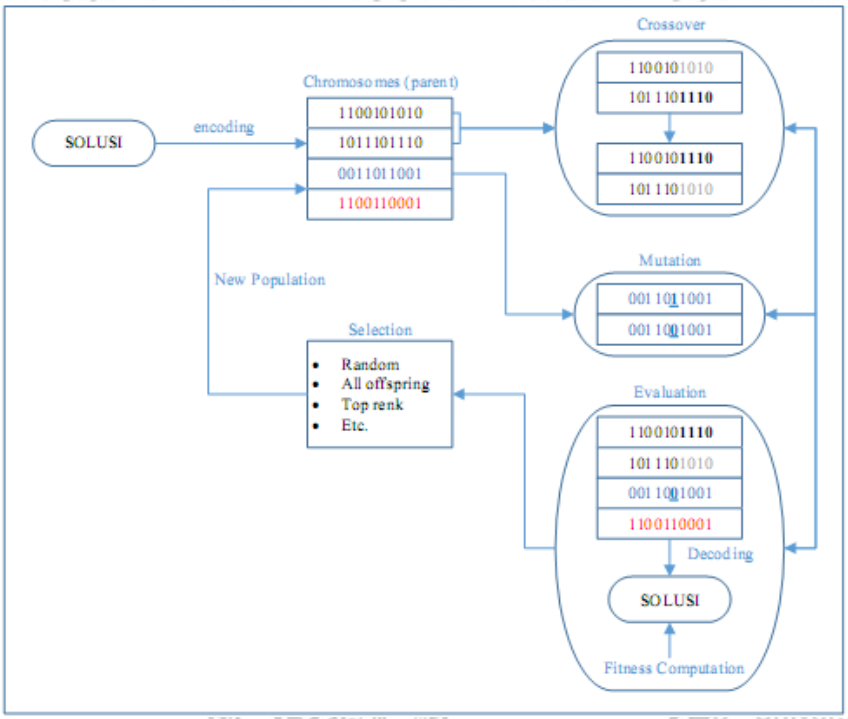
$E_N, E_{N-1} = Error\ yang\ dinyatakan\ dalam\ waktu\ diskrit$

2.6. *GA* (GA)

GA (GA) ditemukan oleh John Holland dan murid-muridnya di Universitas Michigan pada tahun 1960. Pada awalnya John Holland mempelajari fenomena dari mekanisme adaptasi yang terjadi di alam. Kemudian John Holland mereplikasi mekanisme adaptasi (*natural adaptation*) tersebut ke dalam sistem komputer. Dalam bukunya yang berjudul *Adaptation in Natural and Artificial System* pada tahun 1975, John Holland menjelaskan bahwa *GA* (GA) sebagai bagian dari mekanisme evolusi biologi (*biological evolution*) dan menetapkan kerangka teoritis dari algoritma tersebut. Algoritma genetika direpresentasikan oleh urutan langkah-langkah prosedur kromosom buatan yang bergerak dari satu populasi ke populasi baru menggunakan seleksi alami dan teknik yang diambil dari genetika yang dikenal sebagai crossover dan mutasi. Setiap kromosom terdiri dari sejumlah ‘gen’, dan setiap gen diwakili oleh 0 atau 1.

Algoritma genetika berbeda dengan metode konvergensi umum yang bersifat deterministik (Mitsuo, 1997). Algoritma genetika memakai mekanisme seleksi alam dan ilmu genetika sehingga istilah-istilah pada algoritma genetik tidak jauh berbeda dengan istilah-istilah pada seleksi alam dan ilmu genetik. Sebuah solusi yang dibangkitkan dalam algoritma genetika disebut sebagai kromosom, sedangkan kumpulan kromosom-kromosom tersebut disebut sebagai populasi. Sebuah kromosom dibentuk dari komponen-komponen penyusun yang disebut sebagai gen dan nilainya dapat berupa bilangan numerik, biner, simbol ataupun karakter tergantung dari permasalahan yang ingin diselesaikan. Kromosom-kromosom tersebut akan berevolusi secara berkelanjutan yang disebut dengan generasi. Dalam tiap generasi kromosom-kromosom tersebut dievaluasi tingkat keberhasilan nilai solusinya terhadap masalah yang ingin diselesaikan (fungsi_Fitness) menggunakan ukuran yang disebut dengan Fitness.

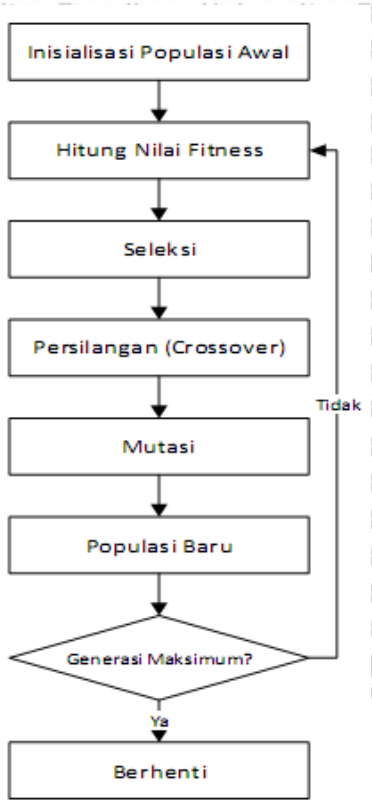
Secara umum tahapan proses dari algoritma genetika diperlihatkan pada Gambar 2.10. Seperti terlihat pada gambar kromosom merupakan representasi dari solusi. Operator genetika yang terdiri dari *crossover* dan mutasi dapat dilakukan bersamaan atau hanya salah satu saja yang selanjutnya operator evolusi dilakukan melalui proses seleksi kromosom dari *parent* (generasi induk) dan dari *offspring* (generasi turunan) untuk membentuk generasi baru (*new population*) yang diharapkan akan lebih baik dalam memperkirakan solusi yang optimum, Kemudian akan dilanjutkan dengan proses iterasi sesuai dengan jumlah generasi yang ditetapkan.



Gambar 2.10 Ilustrasi Tahapan Proses dari Algoritma Genetika

2.6.1. Struktur Umum *GA* (GA)

Algoritma genetika memberikan pilihan untuk menentukan nilai parameter dengan menduplikasi cara reproduksi genetika, pembentukan kromosom baru, proses migrasi gen, serta seleksi alami seperti yang terjadi pada organisme hidup. Secara umum Algoritma Genetika dapat diilustrasikan melalui diagram pada Gambar 2.11.



Gambar 2.11. Diagram Struktur Umum Algoritma Genetika

Inisialisasi populasi awal dilakukan untuk menghasilkan solusi awal dari suatu permasalahan algoritma genetika. Inisialisasi ini dilakukan secara random sesuai dengan jumlah kromosom per populasi yang diinginkan. Selanjutnya dilakukan perhitungan nilai Fitness dan selanjutkan dilakukan seleksi dengan menggunakan metode *roulette wheel*, *tournament* atau *ranking*. Kemudian dilakukan persilangan (*crossover*) dan mutasi. Setelah melalui beberapa generasi maka algoritma ini akan berhenti sebanyak generasi yang diinginkan (Michalewich, 1998). Berikut ini adalah karakteristik-karakteristik yang perlu diketahui sehingga dapat dibedakan dari prosedur pencarian atau optimasi yang lain, yaitu (Goldberg, 1989) :

1. Algoritma Genetika bekerja dengan pengkodean dari himpunan solusi permasalahan.
2. Algoritma Genetika melakukan pencarian pada sebuah populasi dari sejumlah individu-individu yang merupakan solusi permasalahan bukan hanya dari satu individu.

3. Algoritma Genetika merupakan informasi dari fungsi Fitness sebagai cara untuk mengevaluasi individu yang mempunyai solusi terbaik, bukan turunan dari satu fungsi saja.

4. Algoritma Genetika menggunakan aturan-aturan transisi peluang, bukan dari aturan-aturan deterministik.

2.6.2. Membangkitkan Populasi Awal dan Kromosom

Membangkitkan populasi awal merupakan suatu proses pembangkitan sejumlah individu atau kromosom secara acak. Ukuran populasi tergantung pada masalah yang akan diselesaikan dan jenis operator genetika yang akan digunakan. Setelah ukuran populasi ditentukan, dilakukan pembangkitan populasi awal. Apabila ukuran populasi yang dipilih terlalu kecil, maka tingkat eksplorasi atas ruang pencarian global akan terbatas, walaupun arah menuju konvergensi lebih cepat. Apabila ukuran populasi terlalu besar, maka waktu akan banyak terbuang karena berkaitan dengan besarnya jumlah data yang dibutuhkan dan waktu ke arah konvergensi akan lebih lama (Goldberg, 1989).

Teknik dalam pembangkitan populasi awal ini ada beberapa cara, diantaranya adalah sebagai berikut:

1. Random Generator

Inti dari cara ini adalah melibatkan pembangkitkan bilangan random untuk nilai setiap gen sesuai dengan representasi kromosom yang digunakan.

2. Pendekatan Tertentu

Cara ini adalah dengan memasukkan nilai tertentu kedalam gen dari populasi yang dibentuk.

3. Permutasi Gen

Salah satu cara permutasi gen dalam pembangkitan populasi awal adalah penggunaan permutasi Josephus dalam permasalahan kombinatorial seperti TSP.

2.6.3. Operator Genetik

Algoritma genetika merupakan teknik atau metode pencarian untuk permasalahan heuristic dan acak sehingga pemilihan operator yang digunakan sangat dipengaruhi oleh keberhasilan algoritma genetika dalam menemukan solusi optimal permasalahan yang diselesaikan. Adapun operator-operator yang digunakan adalah sebagai berikut:

A. Seleksi

Dalam proses reproduksi setiap individu, populasi pada suatu generasi diseleksi berdasarkan nilai Fitnessnya untuk bereproduksi guna menghasilkan keturunan. Probabilitas terpilihnya suatu individu untuk bereproduksi adalah sebesar nilai Fitness individu tersebut dibagi dengan jumlah nilai Fitness seluruh individu dalam populasi (Davis, 1991).

B. Crossover

Crossover (persilangan) merupakan suatu proses pemilihan lokasi string secara acak dan menukar karakter-karakter stringnya (Goldberg, 1989). Fungsi crossover menghasilkan kromosom anak yang baru dari kombinasi beberapa gen dari dua kromosom induk yang dipindah silangkan. Probabilitas crossover (Pc) ditentukan dan digunakan untuk mengontrol jumlah frekuensi crossover.

C. Mutasi

Mutasi digunakan sebagai cara untuk mengembalikan materi genetik yang hilang. Melalui mutasi, dapat menciptakan individu baru dengan melakukan modifikasi terhadap satu atau lebih nilai gen pada individu yang sama. Mutasi mencegah banyak hilangnya materi genetika setelah reproduksi dan pindah silang. Mutasi berperan untuk menggantikan nilai gen yang hilang dari suatu populasi akibat seleksi yang memungkinkan munculnya kembali gen yang tidak dimunculkan pada saat proses inisialisasi populasi.

D. Parameter – Parameter dalam Algoritma Genetika

Parameter-parameter genetika berperan dalam pengendalian operator-operator genetika yang digunakan dalam optimasi algoritma genetika menggunakan algoritma genetika.(Davis, 1991)

Parameter genetika yang sering digunakan meliputi ukuran populasi (N), probabilitas pindah silang (Pc), dan probabilitas mutasi (Pm). Pemilihan ukuran

populasi yang digunakan tergantung pada masalah yang akan diselesaikan. Untuk masalah yang lebih kompleks biasanya diperlukan ukuran populasi yang lebih besar guna mencegah konvergensi prematur yang menghasilkan lokal optimal.

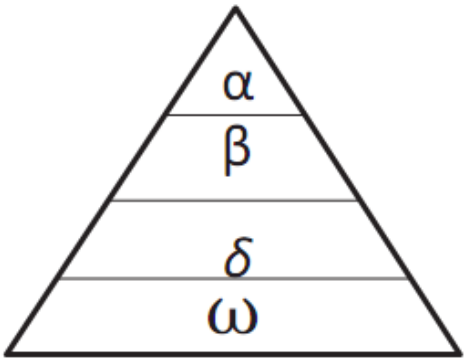
Pada tiap generasi, sebanyak Pc * N individu dalam populasi mengalami pindah silang. Makin besar nilai Pc yang diberikan maka makin cepat struktur individu baru yang diperkenalkan ke dalam populasi. Jika nilai Pc yang diberikan terlalu besar, individu yang merupakan kandidat solusi terbaik dapat hilang lebih cepat dibanding seleksi untuk peningkatan kerja. Sebaliknya nilai Pc yang rendah dapat mengakibatkan stagnasi karena rendahnya angka eksplorasi.

Probabilitas mutasi adalah probabilitas dimana setiap posisi bit pada tiap string dalam populasi baru mengalami perubahan secara acak setelah proses seleksi. Dalam satu generasi dengan L panjang struktur, kemungkinan terjadi mutasi sebanyak Pm*N*L.

2.7. *Grey Wolf Optimizer* (GWO)

Algoritma *Grey Wolf Optimizer* merupakan algoritma yang terinspirasi oleh perilaku berburu serigala di alam. Serigala abu-abu dianggap sebagai predator puncak, yang berarti bahwa serigala abu-abu berada di puncak rantai makanan. Serigala abu-abu juga memiliki hirarki dominan sosial yang tinggi. Para pemimpin yang merupakan tingkatan pertama akan disebut sebagai *alfa*, tingkatan kedua yaitu *beta*, tingkatan ketiga yaitu *delta*, sementara tingkatan terakhir yaitu *omega*. Selain hirarki sosial serigala, berburu secara berkelompok adalah perilaku menarik lain dari serigala abu-abu.

Sosial hirarki dari metode *grey wolves* dikategorikan berdasarkan nilai kebugarannya (*Fitness*). Pembagian hirarki sosial dari *grey wolves* dibagi menjadi empat seperti ditunjukkan dalam Gambar 2.12.



Gambar 2.12 Hirarki Sosial *Grey Wolf*

A. Level 1 – Alpha (α)

Alpha adalah pemimpin dari kelompok serigala abu-abu yang bertanggung jawab dalam pengambilan keputusan kelompoknya. Dalam hal ini, alpha dijadikan acuan utama dari pergerakan gerombolan serigala abu-abu.

B. Level 2 – Beta (β)

Beta merupakan tingkatan kedua dari sosial hirarki serigala abu-abu. Dimana beta berfungsi membantu alpha dalam kepemimpinan dan pengambilan keputusan. Beta merupakan kandidat terkuat apabila alpha telan mati. Beta bertanggung jawab akan hirarki sosial yang ada dibawahnya untuk mengikuti alpha.

C. Level 3 – Delta (δ)

Delta adalah tingkatan ketiga dari sosial hirarki serigala abu-abu. Delta bekerja berdasarkan perintah dari alpha dan beta. Delta memiliki dominasi terhadap omega dan bertugas melaporkan kepada alpha dan beta.

D. Level 4 – Omega (ω)

Omega adalah tingkatan terbawah dari sosial hirarki serigala abu-abu.

Fase utama berburu serigala abu-abu meliputi pelacakan, mengejar, mendekati dan menagacau mangsa sampai berhenti bergerak kemudian menyerang mangsa. Adapun secara rinci perilaku serigala abu-abu dalam berburu dibagi kedalam beberapa tahapan yaitu, melingkari mangsa, berburu, menyerang mangsa (ekploitasi) dan menyerang mangsa (eksplorasi). Seperti disebutkan diatas, serigala abu- abu mengelilingi mangsanya selama berburu. Adapun model matematisnya adalah sebagai berikut:

$$\vec{D} = |\vec{C} . \overrightarrow{X_p}(t) - \vec{X}(t)| \quad (12)$$

$$\vec{X}(t+1) = \overrightarrow{X_p}(t) - \vec{A} . \vec{D} \quad (13)$$

Dimana : t = Iterasi saat ini

$\vec{X}$ = Vektor posisi serigala abu-abu

$\overrightarrow{X_p}$ = Vektor posisi serigala abu-abu dari mangsa

Vektor A dan C dihitung berdasarkan persamaan (14) dan (15).

$$\vec{A} = 2\vec{a} . \vec{r}_1 - \vec{a} \quad (14)$$

$$\vec{C} = 2 . \vec{r}_2 \quad (15)$$

dimana r_1 dan r_2 adalah vektor acak, vektor $\vec{a}$ diset menurun selama waktu iterasi. Tiga penyelesaian terbaik disimpan dalam variabel α, β dan δ. Situasi ini dinyatakan dalam persamaan:

$$\vec{D}_\alpha = \left|\vec{C}_1 . \vec{X}_\alpha - \vec{X}\right| \quad (16)$$

$$\vec{D}_\beta = \left|\vec{C}_2 . \vec{X}_\beta - \vec{X}\right| \quad (17)$$

$$\vec{D}_\delta = \left|\vec{C}_3 . \vec{X}_\delta - \vec{X}\right| \quad (18)$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 . \left(\vec{X}_\alpha\right) \quad (19)$$

$$\vec{X}_2 = \vec{X}_\beta - \vec{A}_2 . \left(\vec{X}_\beta\right) \quad (20)$$

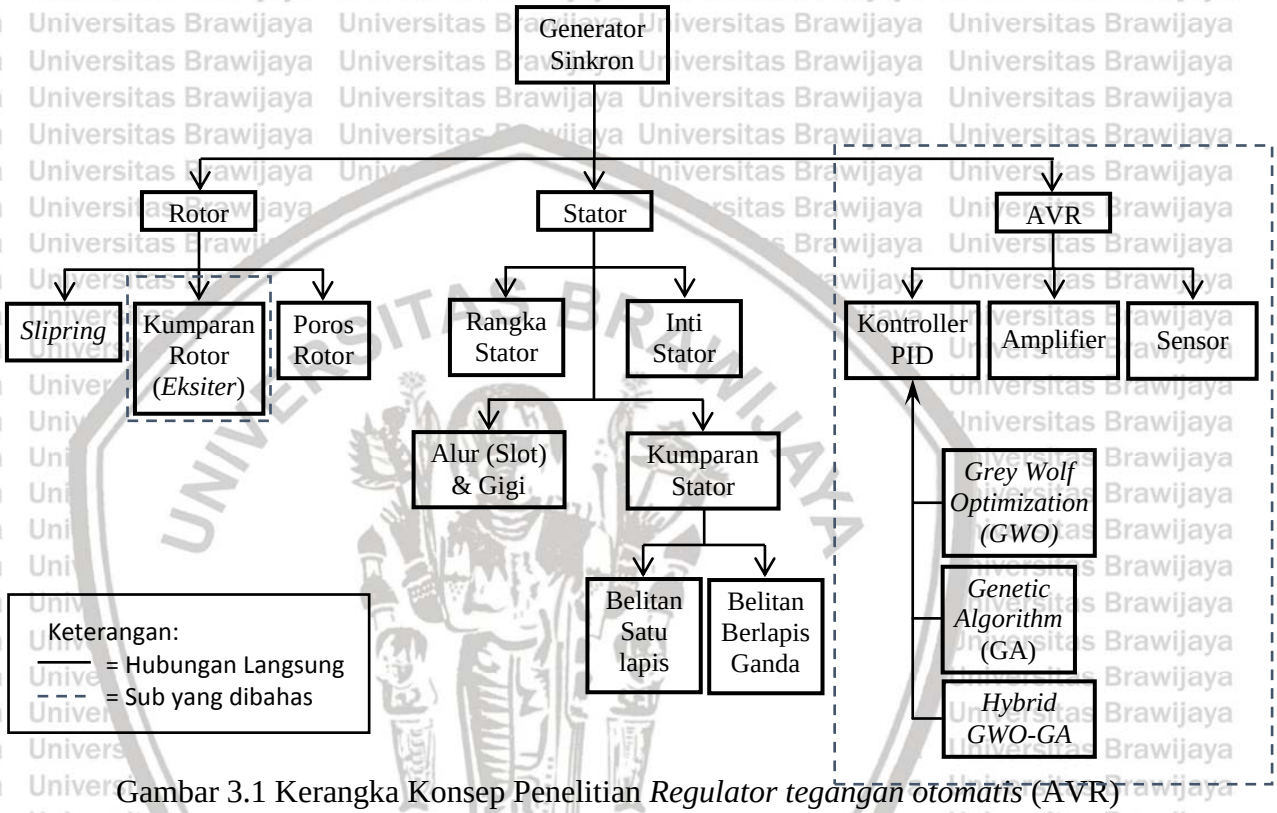
$$\vec{X}_3 = \vec{X}_\delta - \vec{A}_3 . \left(\vec{X}_\delta\right) \quad (21)$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \quad (22)$$

BAB III
KERANGKA KONSEP PENELITIAN

3.1. Kerangka Konsep

Penelitian optimasi *regulator tegangan otomatis* (AVR) berbasis PID menggunakan metode *Hybrid Grey Wolve Optimization – GA* ditunjukkan dalam Gambar 3.1.



Gambar 3.1 Kerangka Konsep Penelitian *Regulator tegangan otomatis* (AVR)

Sesuai dengan prinsip kerja pembagkitan tegangan pada generator sinkron bergantung pada nilai arus eksitasi pada kumparan medan (eksiter). Semakin tinggi nilai arus eksitasi yang diberikan pada kumparan medan, maka nilai tegangan pada kumparan jangkar juga semakin tinggi. Begitupun juga sebaliknya apabila nilai arus eksitasi pada kumparan medan semakin rendah, maka nilai pembangkitan tegangan pada kumparan jangkar akan semakin rendah. Maka dari itu, peran *regulator tegangan otomatis* (AVR) sangat penting dalam menjaga profil tegangan pada terminal kumparan jangkar agar tetap stabil. Dalam menjaga kestabilan tegangan, *regulator tegangan otomatis* (AVR) mendapatkan *feedback* gaya gerak listrik yang dihasilkan oleh output generator.

Pada penelitian ini digunakan metode *hybrid GWO – GA* dalam penalaan parameter kontroller PID. Dimana kontroller PID berfungsi untuk mempertahankan atau meregulasi

nilai tegangan dimana dalam penelitian ini menggunakan setpoint statis dengan nilai 1 PU. Penggunaan metode *artificial intelligent* (AI) diharapkan mampu menambah performansi dari generator sinkron dalam pembangkitan tegangan listrik. Adapun parameter performansi dari pembangkitan tegangan pada generator sinkron meliputi Mp (*Maximum Overshoot*), Ess (*Error Steady State*), tr (*Time Rise*) dan ts (*Time Settling*). Penggunaan parameter diatas berfungsi untuk memperbaiki kualitas respon transien dari tegangan keluaran generator. Adapun beberapa kondisi yang mengakibatkan respon transien memiliki nilai yang relatif kurang baik, diantaranya yang pertama adalah proses transisi dari generator *off* menuju generator *on*. Kedua, ketika terjadi pembebanan pada keluaran generator, sehingga mengakibatkan jatuh tegangan pada terminal keluaran generator. Kemudian yang ketiga adalah ketika terjadi *noise* / gangguan pada sinyal keluaran *regulator tegangan otomatis* (AVR).

Penelitian ini juga melakukan beberapa perbandingan performansi pembangkitan tegangan generator sinkron. Perbandingan performansi yang pertama adalah ketika *regulator tegangan otomatis* (AVR) tanpa kontroller dan menggunakan kontroller PID. Kemudian yang kedua adalah perbandingan performansi penalaan kontroller PID menggunakan metode *GWO*, *GA* serta *hybrid GWO – GA*. Pengujian dan perbandingan optimasi *regulator tegangan otomatis* (AVR) tersebut dilakukan dengan menggunakan software Matlab.

3.2. Variabel Penelitian

Pada penelitian ini memiliki dua variabel, yakni variabel bebas dan variabel terikat. Adapun pembagian masing-masing variable disebutkan dibawah ini.

- Variabel Bebas:
  a. Parameter Kp, Ki, Kd pada kontroller PID.
  b. Metode penalaan parameter kontroller PID (GWO, GA, HGWGA).
- Variabel Terikat:
  a. Fungsi alih kontroller PID.
  b. Fungsi alih kontroller sensor.
  c. Fungsi alih kontroller amplifier.
  d. Fungsi alih kontroller exciter.
  e. Fungsi alih kontroller generator.

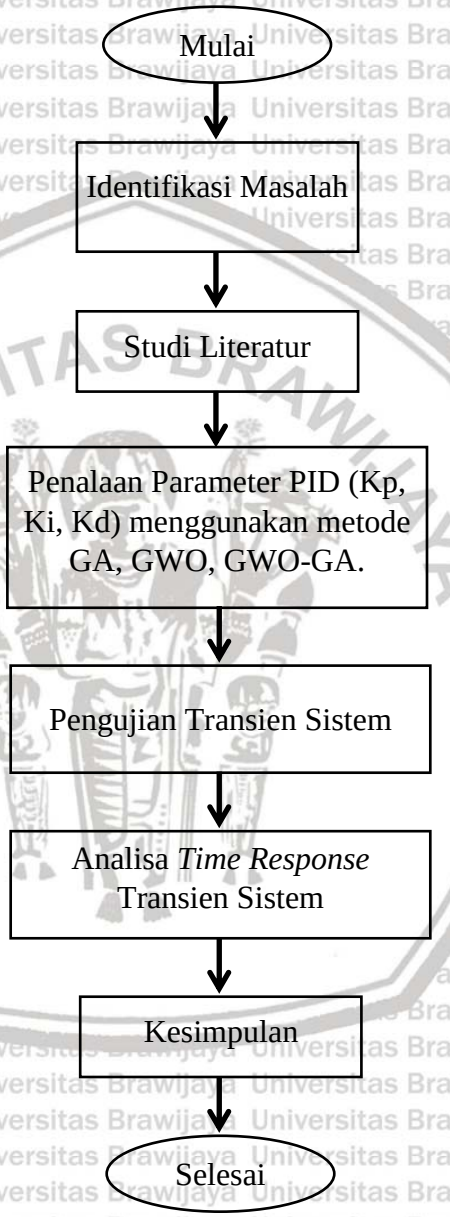
3.3. Hipotesis

Pada penelitian ini, peneliti memiliki hipotesis yakni penggunaan metode penalaan parameter PID dengan menggunakan metode *hybrid GWO – GA* (HGWGA) mempunyai performansi lebih baik daripada metode *GWO* (GWO) dan *GA* (GA). Adapun penilaian performansi yang dimakhsud berdasarkan nilai Mp (*Maksimum Overshoot*), Tr (*Time Rise*), Ts (*Time Settling*), Ess (*Error Steady State*). Adapun kelemahan dari metode HGWGA adalah mempunyai *elapsed time* yang lebih besar daripada metode GA dan GWO.

BAB IV
METODOLOGI PENELITIAN

4.1. Metodologi Penelitian

Tahapan penyusunan penelitian yang dilakukan meliputi langkah-langkah yang ditunjukkan pada diagram alir pada Gambar 4 . 1 .



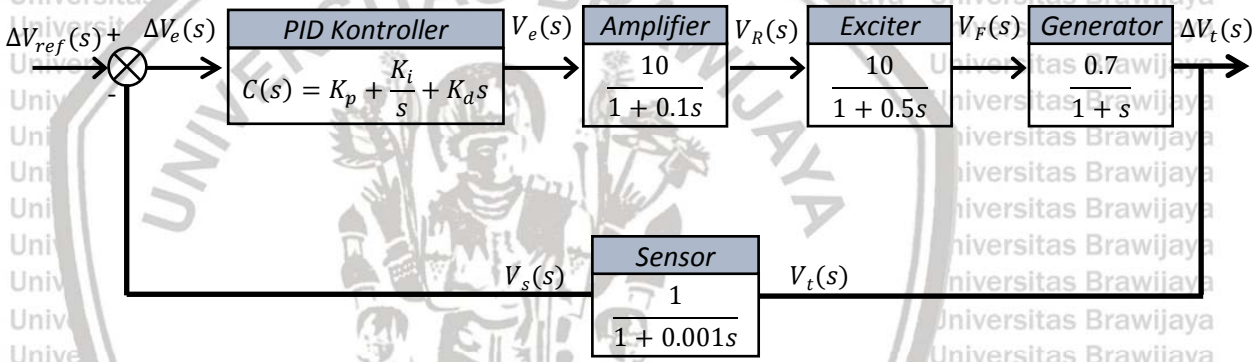
Gambar 4 . 1 Diagram Alir Metodologi Penelitian

Langkah pertama yang dilakukan dalam penelitian ini adalah mengidentifikasi masalah yang menjadi dasar dari latar belakang penelitian. Kemudian studi literatur dilakukan melalui referensi yang terkait baik melalui buku atapun jurnal. Sistem penalaan dari

parameter PID (Kp, Ki, Kd) menggunakan metode GA (*GA*), GWO (*GWO*) dan GWO-GA (*Hybrid GWO - GA*). Tujuan dari penalaan parameter PID menggunakan tiga metode adalah untuk membandingkan dan mencari hasil yang paling optimal dari tiga metode tersebut. Kemudian dilakukan pengujian untuk mengetahui respon plan AVR pada respon transien setelah diberikan penalaan pada parameter Kp, Ki, Kd. Selanjutnya menganalisa hasil *time response* yang meliputi *time rise, time settling, maximum overshoot* dari ketiga metode tersebut. Hasil dari analisa dapat dilakukan penarikan kesimpulan.

4.2. Pemodelan sistem *Regulator tegangan otomatis* (AVR)

Dalam sebuah sistem *Regulator tegangan otomatis* (AVR) tersusun oleh empat komponen penyusun, yakni amplifier, exciter, generator dan sensor. Masing-masing dari komponen penyusun tersebut memiliki fungsi alih yang ditunjukkan dalam Gambar 4.2.



Gambar 4.2 Blok diagram sistem AVR dengan kontroller PID

Fungsi alih dari sistem *regulator tegangan otomatis* (AVR) disusun berdasarkan fungsi alih dari sub-sistem penyusunnya. Adapun masing-masing fungsi alih sub-sistem dijelaskan sebagai berikut:

4.2.1. Fungsi Alih *Amplifier*

Bentuk umum dari persamaan fungsi alih dari amplifier adalah sebagai berikut:

$$\frac{V_R(s)}{V_e(s)} = \frac{K_A}{1+\tau_A s} = \frac{10}{1+0.1s} \quad (23)$$

Dimana nilai K_A merupakan konstanta penguatan (*gain constant*) dari amplifier yang mempunyai range dari 10 sampai 400. Dalam penelitian ini, peneliti menggunakan nilai K_A sebesar 10. Sedangkan konstanta τ_A merupakan konstanta

waktu (*time constant*) yang mempunyai range dari 0.02 sampai 0.1s. Peneliti menggunakan nilai τ_A sebesar 0.1.

4.2.2. Fungsi Alih *Exciter*

Bentuk umum dari persamaan fungsi alih dari exciter adalah sebagai berikut:

$$\frac{V_F(s)}{V_R(s)} = \frac{K_E}{1+\tau_E s} = \frac{10}{1+0.5s} \quad (24)$$

Dimana nilai K_E merupakan konstanta penguatan (*gain constant*) dari exciter yang mempunyai range dari 10 sampai 400. Dalam penelitian ini, peneliti menggunakan nilai K_E sebesar 10. Sedangkan konstanta τ_E merupakan konstanta waktu (*time constant*) yang mempunyai range dari 0.5 sampai 1.0s. Peneliti menggunakan nilai τ_E sebesar 0.5.

4.2.3. Fungsi Alih Generator

Bentuk umum dari persamaan fungsi alih dari generator adalah sebagai berikut:

$$\frac{V_t(s)}{V_F(s)} = \frac{K_G}{1+\tau_G s} = \frac{0.7}{1+s} \quad (25)$$

Dimana nilai K_G merupakan konstanta penguatan (*gain constant*) dari generator yang mempunyai range dari 0.7 sampai 1.0. Dalam penelitian ini, peneliti menggunakan nilai K_G sebesar 0.7. Sedangkan konstanta τ_G merupakan konstanta beban (*load constant*) yang mempunyai range dari 1.0 sampai 2.0s. Peneliti menggunakan nilai τ_G sebesar 1.0.

4.2.4. Fungsi Alih Sensor

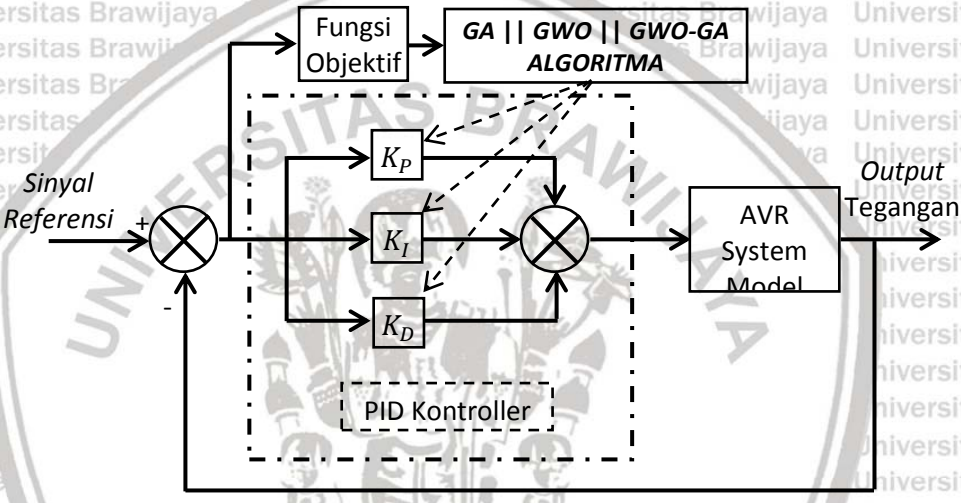
Bentuk umum dari persamaan fungsi alih dari sensor adalah sebagai berikut:

$$\frac{V_s(s)}{V_t(s)} = \frac{K_R}{1+\tau_R s} = \frac{1}{1+0.001s} \quad (26)$$

Pemodelan fungsi alih dari sensor menggunakan fungsi alih orde satu dengan nilai $K_R = 1$ dan $\tau_R = 0.001$.

4.3. Perancangan PID kontroller

Pada penelitian ini, peneliti menggunakan kontroller PID sebagai metode kontrol yang berfungsi untuk memperbaiki kualitas respon *regulator tegangan otomatis* (AVR) dalam menjaga stabilitas tegangan keluaran generator. Adapun parameter dari metode kontrol PID yang berupa nilai Kp, Ki dan Kd ditentukan menggunakan algoritma *GA* (GA), *GWO* (GWO) dan *HGWGA*. Adapun blok diagram sistem *regulator tegangan otomatis* (AVR) berbasis PID dengan penalaan metode *artificial intelligent* (AI) ditunjukkan dalam Gambar 4.3.



Gambar 4.3 Blok diagram sistem AVR berbasis PID dengan menggunakan metode penalaan *Artificial Intelligent* (AI)

4.4. Perancangan Fungsi Fitness

Fungsi Fitness mempunyai peran sangat penting dalam metode *artificial intelligent* (AI). Fungsi Fitness berfungsi untuk mengevaluasi performa dari sebuah solusi yang dihasilkan dari metode *artificial intelligent* (AI). Nilai yang diperoleh dari perhitungan fungsi Fitness adalah representasi dari seberapa baik solusi yang dihasilkan. Dalam penelitian ini, metode *artificial intelligent* (AI) bekerja untuk meminimalisasi nilai fungsi Fitness. Sehingga dalam kasus ini, nilai terkecil fungsi Fitness dari proses optimasi merupakan solusi terbaik yang dihasilkan oleh metode *artifial intelligent* (AI).

Pada penelitian ini, peneliti menggunakan ITAE – *Modified* sebagai fungsi Fitness nya. Fungsi Fitness ITAE - *Modified* merupakan pengembangan dari fungsi Fitness ITAE. Persamaan fungsi Fitness ITAE dan ITAE-*Modified* ditunjukkan dalam persamaan dibawah ini:

$$ITAE = \int_0^{\tau} t|e(t)|dt \quad (27)$$

$$ITAE - Modified = \int_0^{\tau} t|e(t)|dt + w_1.O_v + w_2.e_{ss} + w_3.t_s + w_4.t_r \quad (28)$$

$Keterangan$:

$w_1 = Konstanta\ Faktor\ Pengali\ O_v$
$w_2 = Konstanta\ Faktor\ Pengali\ e_{ss}$
$w_3 = Konstanta\ Faktor\ Pengali\ t_s$
$w_4 = Konstanta\ Faktor\ Pengali\ t_r$
$O_v = Overshoot$
$e_{ss} = Error\ Steady\ State$
$t_s = Time\ Settling$
$t_r = Time\ Rise$

Pada fungsi Fitness ITAE-*Modified* terdapat modifikasi penambahan variabel dari fungsi aslinya ITAE. Penambahan variabel parameter yang digunakan meliputi variabel *overshoot*, *error steady state*, *time settling* dan *time rise*. Pada persamaan ITAE-*Modified* menggunakan konstanta faktor pengali yang dalam hal ini diberikan notasi w_1, w_2, w_3dan w_4. Penentuan nilai konstanta faktor pengali berdasarkan tingkat prioritas variabel yang akan dioptimasi. Pada penelitian ini, menggunakan faktor pengali yang merujuk pada penelitian (M. Zahir, 2020). Adapun nilai konstanta faktor pengali ditentukan sebagai berikut:

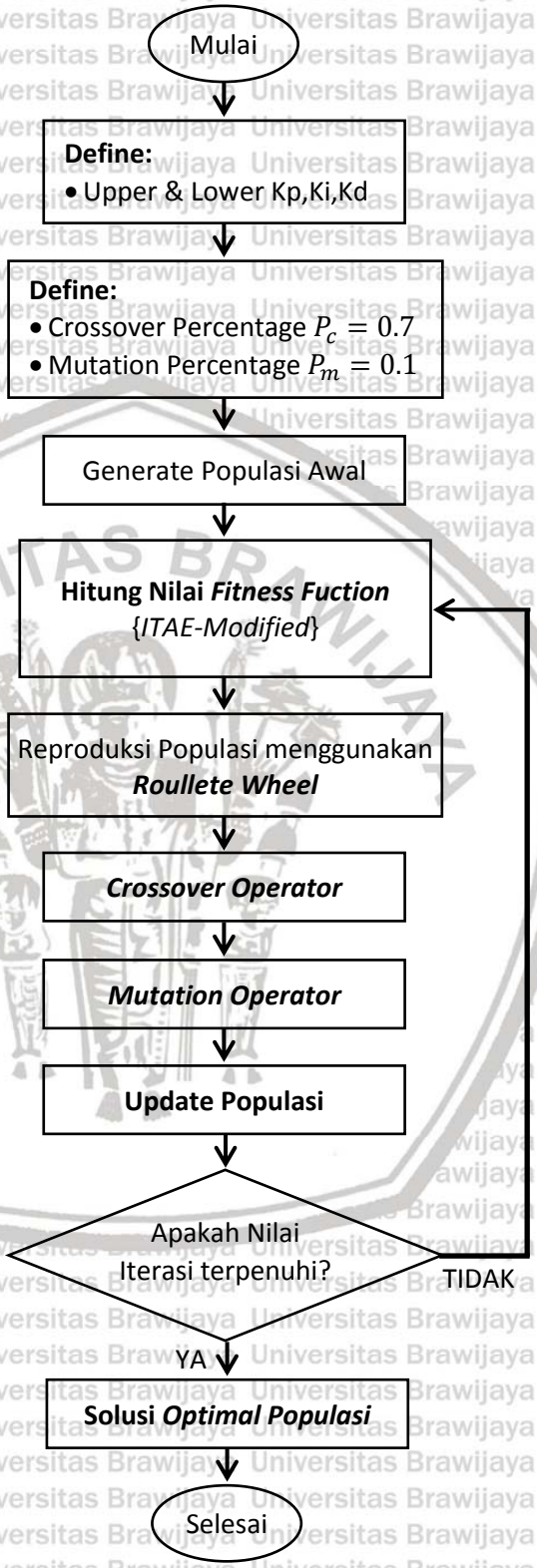
$w_1 = 1.5$
$w_2 = 15$
$w_3 = 7$
$w_4 = 1$

Berdasarkan penentuan nilai konstanta faktor pengali seperti ditunjukkan diatas, maka diperoleh sebuah fungsi Fitness sebagai berikut:

$$ITAE - Modified = \int_0^{\tau} t|e(t)|dt + 1{,}5xO_v + 15xe_{ss} + 7xt_s + 1xt_r \quad (29)$$

4.5. Perancangan *GA* (GA)

Pada penelitian ini, menggunakan salah satu metode *artificial intelligent* (AI) yakni *GA* (GA) dalam proses penalaan parameter PID. Metode *GA* (GA) merupakan algoritma stokastik yang diimplementasi dari proses evolusi biologis. Dalam metode *GA* (GA) mempunyai tiga operator utama, yakni reproduksi, *crossover* dan mutasi. Adapun flowchart dari perancangan metode GA ditunjukkan dalam Gambar 4.4.



Gambar 4.4 Diagram Alir Metode *GA* (GA)

Pada proses perancangan metode *GA* (GA) seperti yang ditunjukkan dalam diagram alir diatas mempunyai beberapa tahapan yang harus dilakukan. Penjelasan dari algoritma diatas sebagai berikut:

Langkah Ke-1 : Mendefinisi batas nilai atas dan bawah dari parameter PID yakni Kp, Ki dan Kd. Lower = [0 0 0]; Upper = [1 1 1].

Langkah Ke-2 : Mendefinisi konstanta operator *GA* (GA) meliputi $P_C = 0.7$ dan $P_m = 0.1$

Langkah Ke-3 : Membangkitkan populasi awal berupa sejumlah individu atau kromosom secara acak.

Langkah Ke-4 : Menghitung nilai *Fitness* (kebugaran) dari populasi sebagai representasi dari sebuah solusi. Dalam hal ini parameter yang digunakan adalah M_p(*Maksimum Overshoot*), E_{ss}(*Error Steady State*), t_r(*Time Rise*), t_s(*Time Settling*).

Langkah Ke-5 : Reproduksi populasi menggunakan metode *roullete wheel*.

Langkah Ke-6 : *Crossover operator* pada populasi dengan parameter *crossover percentage* $P_C = 0.7$

Langkah Ke-7 : *Mutation operator* pada populasi dengan parameter *mutation percentage* $P_m = 0.1$

Langkah Ke-8 : Mengganti populasi awal dengan populasi terakhir (hasil *crossover* dan *mutation* operator)

Langkah Ke-9 : Pengecekan jumlah iterasi apakah sudah terpenuhi. Apabila sudah terpenuhi, maka populasi terakhir adalah solusi paling optimal. Apabila belum terpenuhi kembali pada langkah ke-4.

4.6. Perancangan *GWO* (GWO)

Pada penelitian ini, menggunakan salah satu metode *artificial intelligent* (AI) yakni *GWO* (GWO) dalam proses penalaan parameter PID. Adapun *flowchart* dari perancangan metode GWO (GWO) ditunjukkan dalam Gambar 4.5

Gambar 4.5 Diagram Alir Metode *Grey Wolves Optimization* (GWO)

Pada proses perancangan metode *grey wolves optimization* (GWO) seperti yang ditunjukkan dalam diagram alir diatas mempunyai beberapa tahapan yang harus dilakukan. Penjelasan algoritma diatas sebagai berikut:

Langkah Ke-1 : Mendefinisi jumlah *grey wolves* dan jumlah iterasi yang dilakukan.

Langkah Ke-2 : Membangkitkan populasi awal *grey wolves* dengan tingkatan hirarki sosial $(\alpha, \beta, \delta, \omega)$.

Langkah Ke-3 : Pencarian dan estimasi posisi dari mangsa.

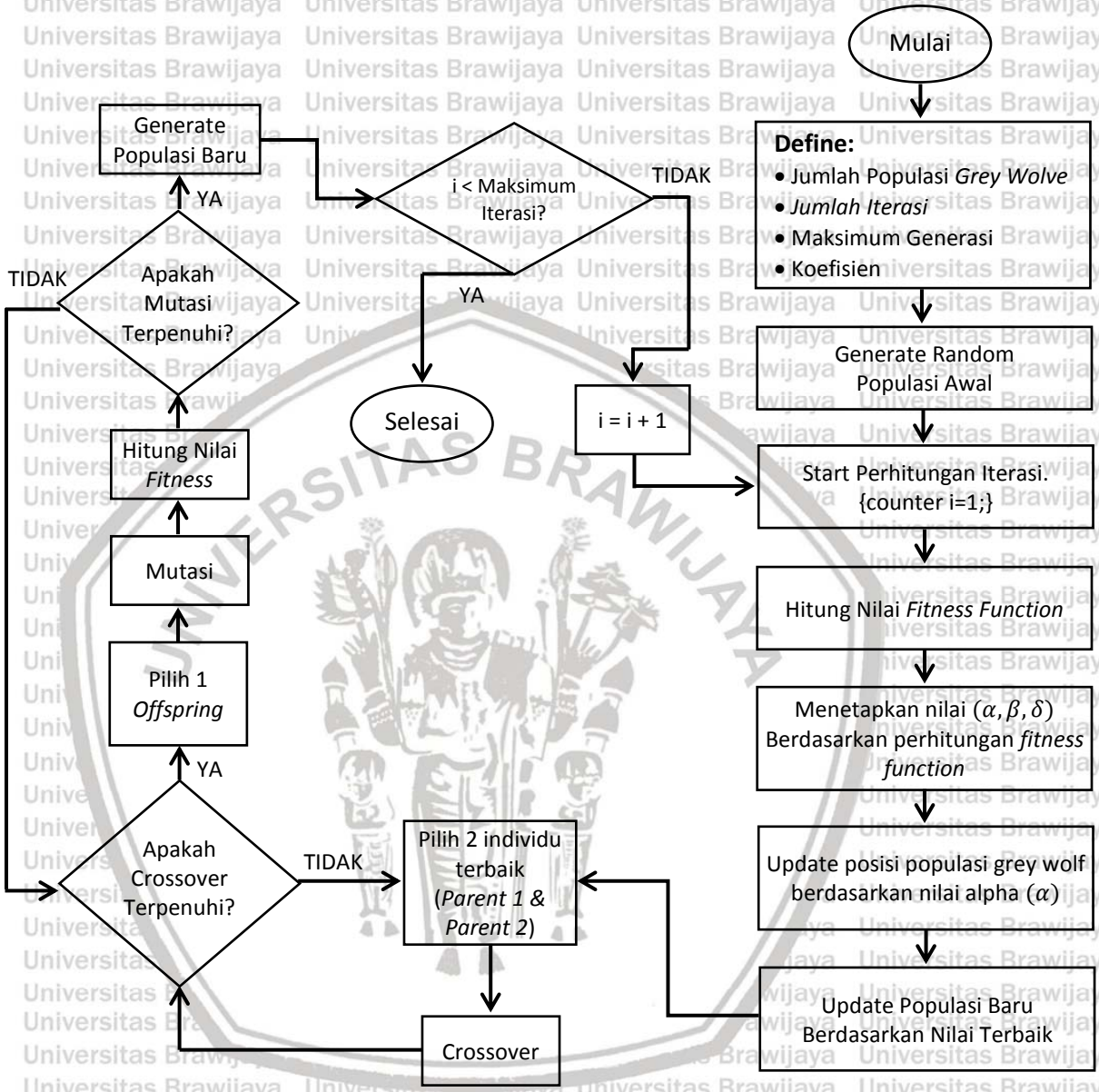
Langkah Ke-4 : Mengevaluasi posisi *grey wolves* terhadap mangsa kemudian melakukan reposisi *grey wolves* ke arah mangsa.

Langkah Ke-5 : Mengklasifikasi solusi berdasarkan tingkatan sosial hirarki (α, β, δ).

Langkah Ke-6 : Jika nilai iterasi telah terpenuhi, maka didapatkan solusi optimal grey wolves. Apabila nilai iterasi belum terpenuhi, maka proses kembali ke langkah 3.

4.7. Perancangan *Hybrid Grey Wolf - GA* (HGWGA)

Pada penelitian ini, menggunakan metode *artificial intelligent* (AI) yakni *hybrid GWO – GA* (HGWGA) dalam proses penalaan parameter PID. Metode HGWGA menggabungkan antara metode GWO dan GA. Proses *hybrid* dari kedua metode tersebut dilakukan secara serial. Dalam hal ini, proses yang pertama dilakukan adalah dengan menggunakan metode *GWO* (GWO) terlebih dahulu, kemudian dilanjutkan dengan menggunakan metode *GA* (GA). Adapun *flowchart* metode hybrid GWO – GA (HGWGA) ditunjukkan dalam Gambar 4.6.



Gambar 4.6 Diagram Alir Metode *Hybrid Grey Wolves Optimization – GA* (HGWGA)

Pada proses perancangan metode *Hybrid grey wolves optimization – GA* (HGWGA) seperti yang ditunjukkan dalam diagram alir diatas mempunyai beberapa tahapan yang harus dilakukan. Penjelasan algoritma diatas sebagai berikut:

Langkah Ke-1 : Mendefinisi jumlah populasi *grey wolves*, jumlah iterasi, maksimum generasi dan koefisien.

Langkah Ke-2 : Membangkitkan populasi awal.

Langkah Ke-3 : Memulai *increment counter iterasi* (i=1)

Langkah Ke-4 : Menghitung nilai *Fitness function* dari setiap populasi yang dibangkitkan.

Langkah Ke-5 : Berdasarkan hasil perhitungan *Fitness function* dari langkah empat, dapat ditentukan nilai α, β, δ.

Langkah Ke-6 : Reposisi dari populasi *grey wolves* berdasarkan nilai α.

Langkah Ke-7 : Memperbarui populasi berdasarkan nilai-nilai terbaik dari populasi awal.

Langkah Ke-8 : Memilih dua individu terbaik (*Parent 1* & *Parent 2*).

Langkah Ke-9 : Proses *crossover* dari dua individu terbaik.

Langkah Ke-10 : Mengecek apakah proses crossover sudah terpenuhi. Apabila sudah terpenuhi maka dilanjutkan pada langkah 11. Jika belum terpenuhi maka akan kembali ke langkah 9.

Langkah Ke-11 : Memilih satu *offspring* dari hasil *crossover*.

Langkah Ke-12 : Melakukan proses mutasi.

Langkah Ke-13 : Menghitung nilai *Fitness*.

Langkah Ke-14 : Mengecek apakah proses mutasi sudah terpenuhi. Apabila sudah terpenuhi dilanjutkan ke poin 15. Jika belum terpenuhi kembali ke poin 10.

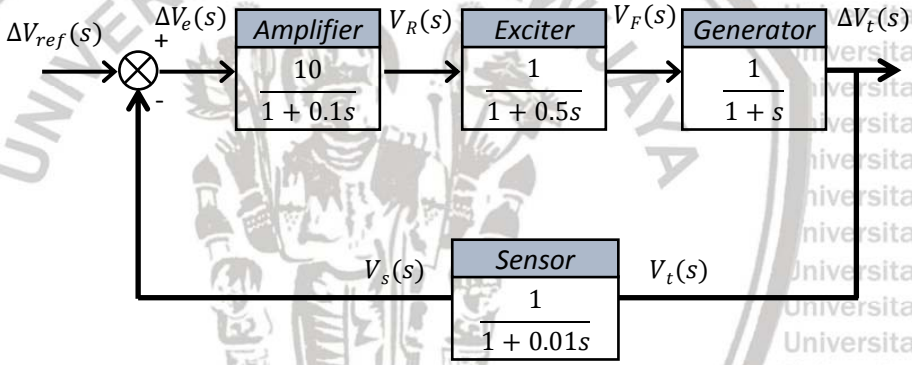
Langkah Ke-15 : Membangkitkan populasi baru dari hasil mutasi.

Langkah Ke-16 : Mengecek apakah nilai iterasi 'i' sudah terpenuhi. Apabila sudah terpenuhi, maka proses selesai. Apabila belum terpenuhi maka nilai *counter* iterasi ditambahkan (*i=i+1*).

BAB V
HASIL PENELITIAN DAN PEMBAHASAN

5.1 Pengujian *Regulator tegangan otomatis* (AVR) Tanpa Kontroller

Pada pengujian sistem *regulator tegangan otomatis* (AVR) tanpa kontroller menggunakan empat sub-sistem sebagai penyusunnya. Adapun sub-sistem yang digunakan adalah amplifier, exciter, generator dan sensor. Pengujian *regulator tegangan otomatis* (AVR) tanpa kontroller bertujuan untuk mengetahui respon transien sistem ketika tidak terdapat blok kontroller PID yang berfungsi sebagai regulator tegangan keluaran. Blok diagram loop tertutup pada *regulator tegangan otomatis* (AVR) tanpa kontroller ditunjukkan dalam Gambar 5.1. Pengujian *regulator tegangan otomatis* (AVR) dilakukan dengan cara memberikan masukan unit step pada sistem.



Gambar 5.1 Blok Diagram Loop Tertutup Sistem *Regulator tegangan otomatis* (AVR) Tanpa Kontroller

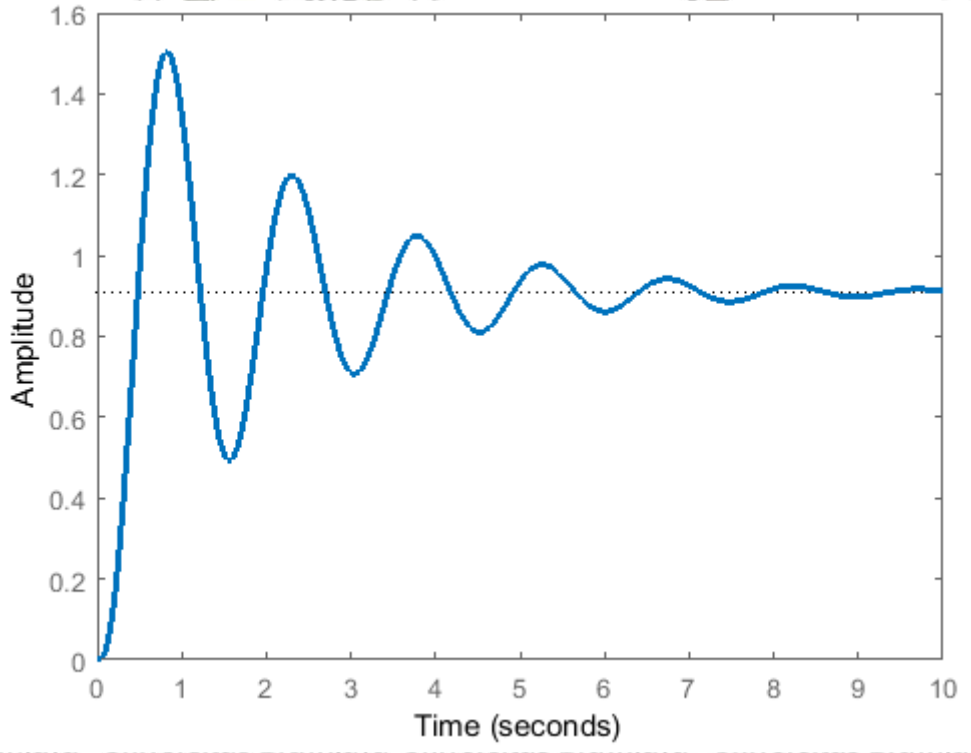
Parameter sub-sistem menggunakan fungsi alih dari masing-masing sub-sistem. Pada fungsi alih penyusun sistem AVR terdapat nilai konstanta yang mempunyai range nilai tertentu. Berdasarkan nilai fungsi alih sub-sistem dapat ditentukan nilai fungsi alih sistem secara keseluruhan. Adapun fungsi alih dari sub-sistem dan konstanta penyusunnya ditunjukkan Tabel 5 . 1.

Tabel 5.1 Fungsi Alih Sub-Sistem Regulator tegangan otomatis (AVR) Tanpa Kontroller

No.	Sub-Sistem	Fungsi Alih	Range Konstanta	Nilai Konstanta
1	Amplifier	$\frac{K_A}{1+\tau_A s}$	$K_A = 10 - 400$ $\tau_A = 0.02 - 0.1$	$\boldsymbol{K_A = 10}$ $\boldsymbol{\tau_A = 0.1}$
2	Exciter	$\frac{K_E}{1+\tau_E s}$	$K_E = 10 - 400$ $\tau_E = 0.5 - 1.0$	$\boldsymbol{K_E = 10}$ $\boldsymbol{\tau_E = 0.5}$
3	Generator	$\frac{K_G}{1+\tau_G s}$	$K_G = 0.7 - 1.0$ $\tau_G = 1.0 - 2.0$	$\boldsymbol{K_G = 1.0}$ $\boldsymbol{\tau_G = 1.0}$
4	Sensor	$\frac{K_R}{1+\tau_R s}$	$K_R = 0.1 - 1.0$ $\tau_R = 0.001 - 0.01$	$\boldsymbol{K_R = 1.0}$ $\boldsymbol{\tau_R = 0.01}$

5.1.1 Hasil Pengujian *Regulator tegangan otomatis* (AVR) Tanpa Kontroller

Hasil pengujian *regulator tegangan otomatis* (AVR) tanpa kontroller dengan memberikan masukan unit step pada sistem ditunjukkan Gambar 5 . 2 .



Gambar 5.2 Hasil pengujian unit step *regulator tegangan otomatis* (AVR) tanpa kontroller

Berdasarkan hasil pengujian *regulator tegangan otomatis* (AVR) tanpa kontroller didapatkan nilai parameter respon transien yang ditunjukkan Tabel 5 . 2 .

Tabel 5 . 2 Parameter respon transien sistem *regulator tegangan otomatis* (AVR) tanpa kontroller

Parameter	Nilai
Rise Time	0.2850 s
Settling Time	7.6433 s
Maksimum Overshoot	1.5046 V
Peak Time	0.8256 s

Fungsi alih sistem *regulator tegangan otomatis* (AVR) secara keseluruhan ditunjukkan dalam persamaan dibawah ini:

$$G(s) = \frac{Vt(s)}{Vref(s)} = \frac{0.1s + 10}{0.0005\,s^4 + 0.0565\,s^3 + 0.666\,s^2 + 1.61\,s + 11}$$

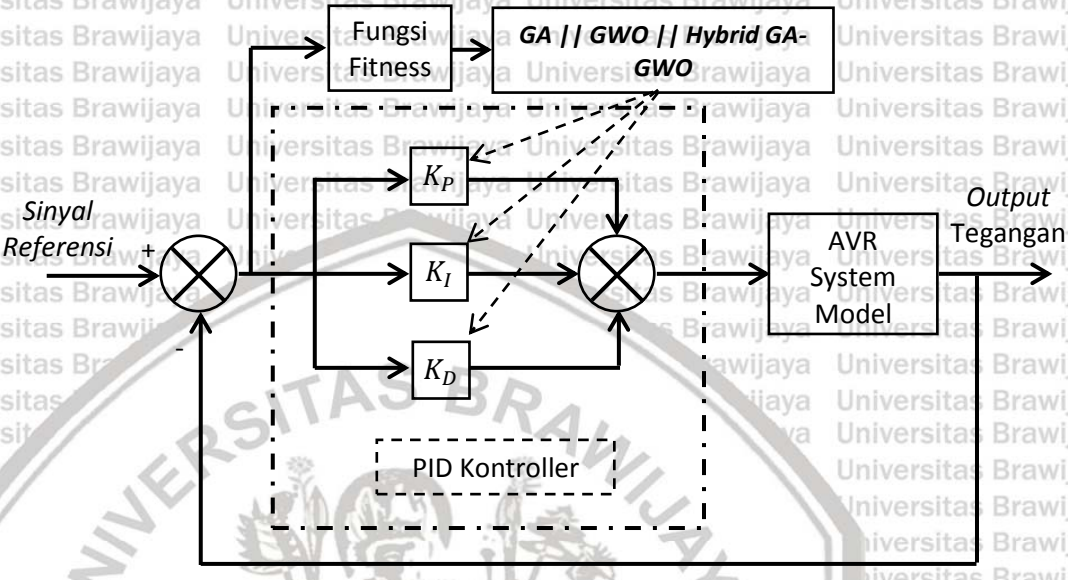
5.1.2 Pembahasan hasil pengujian *regulator tegangan otomatis* (AVR) Tanpa Kontroller

Berdasarkan hasil pengujian *regulator tegangan otomatis* (AVR) tanpa kontroller seperti yang ditunjukkan Tabel 5 . 2 dapat diketahui bahwa parameter respon transien *rise time, settling time, maksimum overshoot* dan *peak time* memiliki nilai yang relatif besar dan masih jauh dari nilai yang diharapkan. Berdasarkan grafik respon sistem yang ditunjukkan dalam Gambar 5.2 dapat diketahui bahwa sistem masih memiliki osilasi dengan nilai *overshoot* yang relatif tinggi. Hal ini dikarenakan sistem tidak memiliki kontroller yang bertugas sebagai regulator tegangan output.

5.2 Pengujian *Regulator tegangan otomatis* (AVR) Berbasis PID Menggunakan Metode *Artificial Intelligent* (AI)

Pada pengujian sistem *regulator tegangan otomatis* (AVR) berbasis PID dengan penalaan parameter Kp, Ki dan Kd menggunakan metode *Artificial Intelligent* (AI) menggunakan lima sub-sistem penyusunnya. Adapun sub-sistem yang digunakan adalah kontroller (PID), amplifier, exciter, generator dan sensor. Pengujian *regulator tegangan otomatis* (AVR) dengan menggunakan PID kontroller bertujuan untuk mengetahui respon

transien sistem ketika penalaan parameter Kp, Ki dan Kd menggunakan metode *GA* (GA), *grey wolve optimization* (GWO), *hybrid GA – grey wolve optimization*. Blok diagram sistem AVR berbasis PID dengan menggunakan metode penalaan *artificial intelligent* (AI) ditunjukkan dalam Gambar 5 . 3. Pengujian sistem *regulator tegangan otomatis* (AVR) dilakukan dengan cara memberikan masukan unit step pada sistem.



Gambar 5.3 Blok diagram sistem AVR berbasis PID dengan menggunakan metode penalaan *GA* (GA)

5.2.1 Pengujian *GA* (GA)

Algoritma *GA* (GA) adalah metode pertama dalam optimasi sistem *regulator tegangan otomatis* (AVR) berbasis PID. Langkah pertama dalam proses penalaan parameter PID adalah dengan cara membangkitkan nilai populasi secara acak. Dimana dalam sebuah populasi terdapat individu atau kromosom yang merepresentasikan sebuah solusi yang dalam hal ini adalah parameter Kp, Ki dan Kd. Langkah selanjutnya adalah menghitung nilai Fitness dari tiap individu dengan menggunakan persamaan fungsi Fitness. Berdasarkan hasil dari proses penghitungan nilai Fitness, dilakukan proses seleksi, mutasi dan *crossover* yang kemudian dijadikan acuan dalam membangkitkan populasi baru. Proses akan berulang terus menerus sampai didapatkan nilai Fitness konvergen atau nilai maksimum generasi terpenuhi. Adapun parameter dalam *GA* (GA) ditunjukkan Tabel 5 . 3 .

Tabel 5 . 3 Parameter *GA* (GA)

Parameter	Keterangan	Nilai
MaxGenerations	Maksimum Generasi	*10*
PopulationSize	Jumlah Populasi	*25*
PopulationType	Tipe Data Populasi	*doubleVector*
CreationFcn	Fungsi Pembangkitan Populasi	*Uniform*
SelectionFcn	Fungsi Pemilihan '*parents*'	*Uniform*
MutationFcn	Fungsi Mutasi Individu	*Uniform*
CrossoverFcn	Fungsi *Crossover* Individu	*Single Point Crossover*
CrossoverFraction	Nilai Crossover Fraction	*0.8*

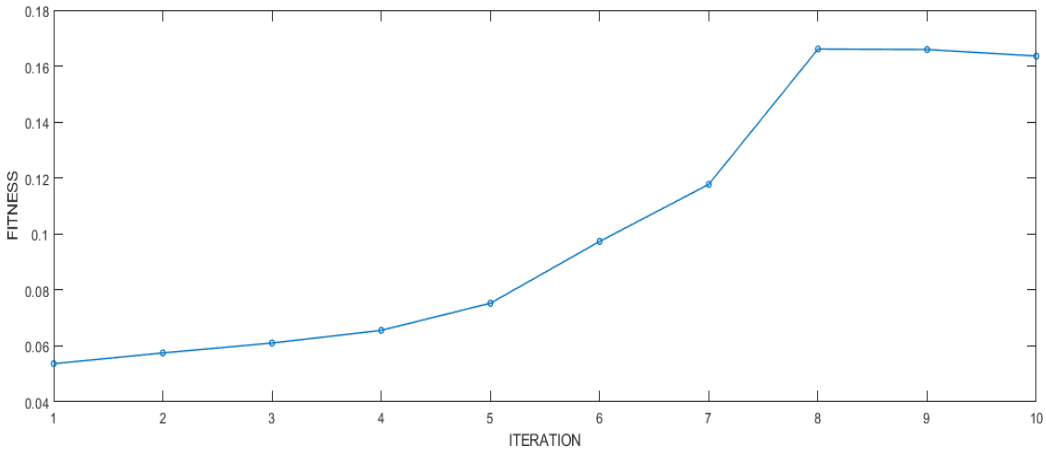
5.2.2 Hasil Pengujian *GA* (GA)

Hasil pengujian *regulator tegangan otomatis* (AVR) berbasis PID dengan menggunakan metode penalaan *GA* ditunjukkan Tabel 5 . 4 .

Tabel 5.4 Hasil Pengujian Regulator tegangan otomatis (AVR) Berbasis PID dengan Penalaan Menggunakan Metode *GA* (GA)

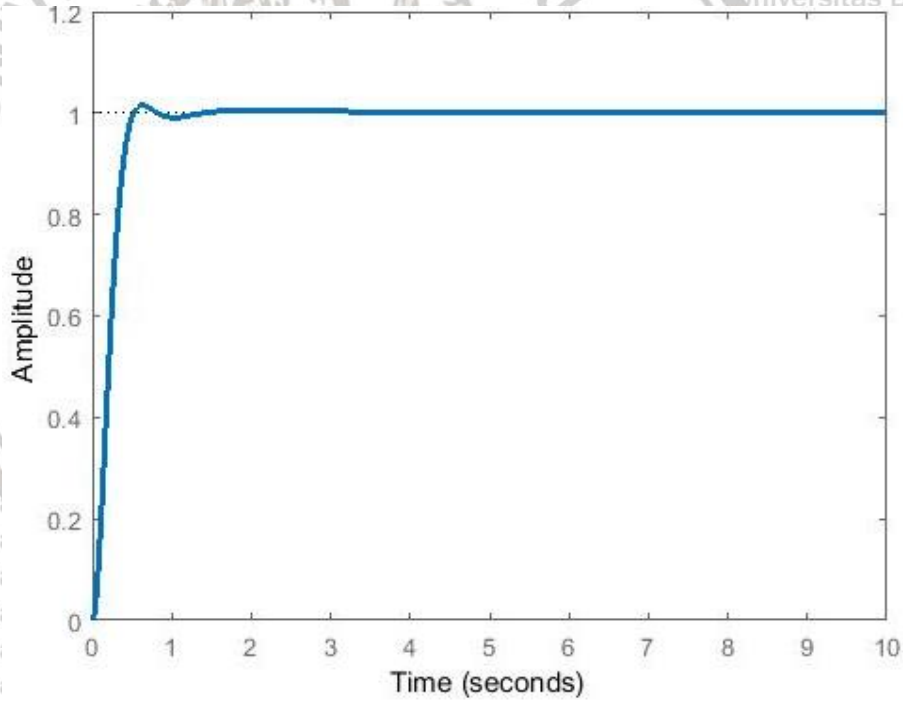
Percobaan	Parameter PID			Rise Time (s)	Settling Time (s)	Maksimum Overshoot (%)	Peak (PU)	Peak Time (s)	Fitness	Elapsed Time (s)
	Kp	Ki	Kd							
1	0.7037	0.5204	0.3368	0.2509	1.1248	1.1254	1.0113	0.4655	0.1017	35.0820
2	0.6432	0.5304	0.4387	0.2097	1.4671	1.6907	1.0169	3.0039	0.0765	32.1153
3	0.6603	0.3880	0.3044	0.2797	1.6160	0.0000	0.9982	3.0359	0.0862	30.7722
4	0.6284	0.4641	0.2728	0.3049	1.0777	0.9119	1.0091	2.3515	0.1084	31.1939
5	0.5932	0.4332	0.3061	0.2940	1.3494	1.0059	1.0101	2.8518	0.0886	30.6142
6	**0.6477**	**0.4592**	**0.2476**	**0.3149**	**0.4756**	**1.5813**	**1.0158**	**0.6283**	**0.1661**	**30.7106**
7	0.8076	0.5506	0.3155	0.2461	0.9519	5.2686	1.0527	0.4880	0.0675	30.4010
8	0.5547	0.3497	0.2762	0.3352	1.7121	0.2263	1.0023	3.1340	0.0788	30.2577
9	0.4432	0.3361	0.3000	0.9303	1.8976	1.7118	1.0171	3.5258	0.0591	30.6451
10	0.7515	0.6064	0.3948	0.2176	1.0578	2.9935	1.0299	0.4105	0.0825	30.8452
								Rata-Rata		**31.2637**

Berdasarkan hasil pengujian yang ditunjukkan Tabel 5.4, dapat diketahui bahwa penalaan parameter PID menggunakan metode *GA* (GA) didapatkan nilai terbaik pada percobaan ke -6. Hal ini mengacu pada nilai Fitness yang terbaik yakni sebesar 0,1661. Adapun nilai parameter Kp = 0,6477, Ki = 0,4592 dan Kd = 0,2476. Proses optimasi nilai Fitness pada percobaan ke -1 ditunjukkan Gambar 5 . 4 .



Gambar 5.4 Nilai Fitness Terhadap Generasi

Berdasarkan proses optimasi yang ditunjukkan dalam Gambar 5.4 dapat diketahui bahwa nilai Fitness terbaik pada saat optimasi sebesar 0.1661, dan waktu pemrosesan yang dibutuhkan (*elapsed time*) sebesar 30,7106 detik. Adapun hasil pengujian respon transien sistem *regulator tegangan otomatis* (AVR) berbasis PID dengan memberikan masukan unit step ditunjukkan Gambar 5 . 5 .



Gambar 5 . 5 Hasil Pengujian unit step *Regulator tegangan otomatis* (AVR) berbasis PID Dengan Penalaan *GA* (GA)

5.2.3 Pengujian *Grey Wolve Optimization* (GWO)

Pada pengujian sistem *regulator tegangan otomatis* (AVR) berbasis PID dengan penalaan parameter Kp, Ki dan Kd menggunakan metode *GWO* (GWO)

menggunakan lima sub-sistem penyusunnya. Adapun sub-sistem yang digunakan adalah kontroller (PID), amplifier, exciter, generator dan sensor. Pengujian *regulator tegangan otomatis* (AVR) dengan menggunakan PID kontroller bertujuan untuk mengetahui respon transien sistem ketika penalaan parameter Kp, Ki dan Kd menggunakan metode *GWO* (GWO). Blok diagram sistem AVR berbasis PID dengan menggunakan metode penalaan *GWO* (GWO) ditunjukkan dalam Gambar 5 . 3 . Pengujian sistem *regulator tegangan otomatis* (AVR) dilakukan dengan cara memberikan masukan unit step pada sistem.

Algoritma *GWO* (GWO) adalah metode kedua dalam optimasi sistem *regulator tegangan otomatis* (AVR) berbasis PID. Langkah pertama dalam proses penalaan parameter PID adalah dengan cara membangkitkan nilai jumlah serigala abu-abu (*search agents*) secara acak. Dimana dalam sebuah *search agents* merepresentasikan sebuah solusi yang dalam hal ini adalah parameter Kp, Ki dan Kd. Langkah selanjutnya adalah menghitung nilai Fitness dari tiap individu dengan menggunakan persamaan fungsi Fitness. Berdasarkan hasil dari proses penghitungan nilai Fitness, dapat ditentukan nilai dari *alpha*, *beta* dan *delta*. Dimana penentuan ketiga parameter tersebut berdasarkan nilai tiga *Fitness* terbaik. Langkah selanjutnya adalah melakukan proses pengepungan dan penyerangan mangsa. Proses akan berulang terus menerus sampai didapatkan nilai *Fitness* yang konvergen atau nilai maksimum iterasi terpenuhi. Adapun parameter dalam algoritma *GWO* (GWO) ditunjukkan Tabel 5 . 5 .

Tabel 5 . 5 Parameter *GWO* (GWO)

Parameter	Keterangan	Nilai
Search Agents	Jumlah Serigala Abu-Abu	*25*
Max Iteration	Maksimum Iterasi	*10*
lb	Batas Bawah Parameter	*[0 0 0]*
ub	Batas Atas Parameter	*[1 1 1]*

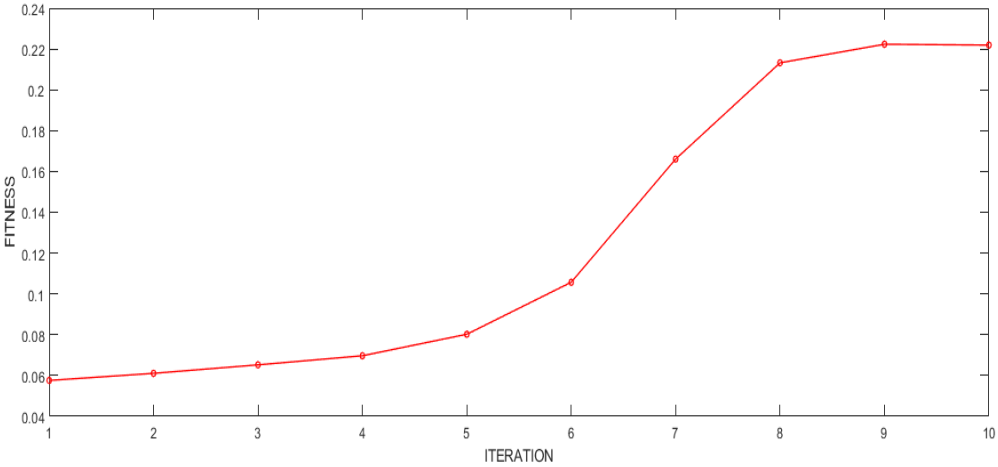
5.2.4 Hasil Pengujian *Grey Wolve Optimization* (GWO)

Hasil pengujian *regulator tegangan otomatis* (AVR) berbasis PID dengan menggunakan metode penalaan *grey wolve optimization* (GWO) ditunjukkan Tabel 5 . 6 .

Tabel 5 . 6 Hasil Pengujian Regulator tegangan otomatis (AVR) Berbasis PID dengan Penalaan Menggunakan Metode *Grey Wolve Optimization* (GWO)

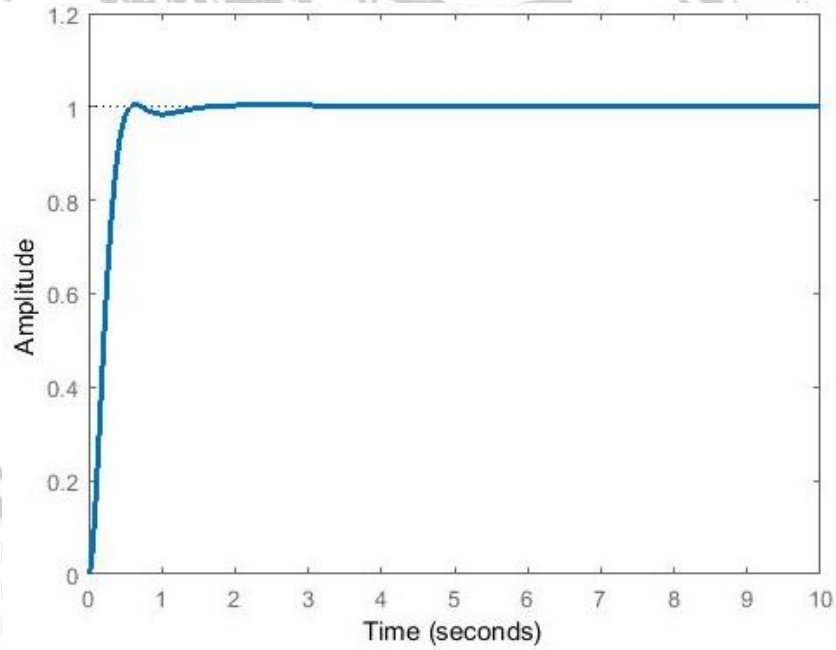
Percobaan	Parameter PID			Rise Time (s)	Settling Time (s)	Maksimum Overshoot (%)	Peak (PU)	Peak Time (s)	Fitness	Elapsed Time (s)
	Kp	Ki	Kd							
1	0.6450	0.5070	0.2710	0.3000	0.4582	1.3289	1.0133	2.0039	0.18149	38.9603
2	0.6264	0.4471	0.2507	0.3204	0.4947	0.6403	1.0064	2.2609	0.21026	38.5080
3	0.6827	0.5276	0.2921	0.2781	0.4187	1.4958	1.0150	0.5431	0.18310	38.2868
4	0.5669	0.3683	0.2118	0.3760	0.6049	0.0353	1.0004	2.7527	0.21421	38.1238
5	0.6078	0.4155	0.2426	0.3338	0.5271	0.3657	1.0037	2.3512	0.21809	38.2569
6	0.5792	0.3845	0.2209	0.3623	0.5797	0.1654	1.0017	2.5547	0.21383	38.8785
7	0.7112	0.5212	0.2931	0.2725	0.5863	2.4345	1.0243	0.5311	0.12435	38.6790
8	0.6239	0.4668	0.2558	0.3170	0.4909	0.9880	1.0099	2.1188	0.19060	38.5827
9	0.6614	0.4700	0.2696	0.2978	0.4524	0.9793	1.0098	0.5841	0.20231	38.4851
10	**0.6278**	**0.4316**	**0.2458**	**0.3238**	**0.4986**	**0.4516**	**1.0045**	**0.6239**	**0.22193**	**39.5309**
								Rata-Rata		**38.6292**

Berdasarkan pengujian yang ditunjukkan Tabel 5.6, dapat diketahui bahwa penalaan parameter PID menggunakan metode *grey wolve optimization* (GWO) didapatkan nilai terbaik pada percobaan ke -10. Hal ini mengacu pada nilai Fitness yang terbaik yakni sebesar 0,2219. Adapun nilai parameter Kp = 0,6278, Ki = 0,4316 dan Kd = 0,2458. Proses optimasi nilai Fitness pada percobaan ke -10 ditunjukkan Gambar 5 . 6 .



Gambar 5 . 6 Nilai Fitness Terhadap Iterasi

Berdasarkan proses optimasi yang ditunjukkan dalam Gambar 5.6 dapat diketahui bahwa nilai Fitness terbaik pada saat optimasi sebesar 0,2219 dan waktu pemrosesan yang dibutuhkan (*elapsed time*) sebesar 39,5309 detik. Adapun hasil pengujian respon transien sistem *regulator tegangan otomatis* (AVR) berbasis PID dengan memberikan masukan unit step ditunjukkan Gambar 5 . 7 .



Gambar 5 . 7 Hasil Pengujian unit step *Regulator tegangan otomatis* (AVR) berbasis PID Dengan Penalaan *Grey Wolve Optimization* (GWO)

5.2.5 Pengujian *Hybrid GA - Grey Wolve Optimization*

Algoritma *hybrid GA - GWO* adalah metode ketiga dalam optimasi sistem *regulator tegangan otomatis* (AVR) berbasis PID. Langkah pertama dalam proses penalaan parameter PID adalah dengan cara membangkitkan nilai jumlah serigala

abu-abu (*search agents*) secara acak. Dimana dalam sebuah *search agents* merepresentasikan sebuah solusi yang dalam hal ini adalah parameter Kp, Ki dan Kd. Langkah selanjutnya adalah menghitung nilai Fitness dari tiap individu dengan menggunakan persamaan fungsi Fitness. Berdasarkan hasil dari proses penghitungan nilai Fitness, dapat ditentukan nilai dari *alpha*, *beta*. Dimana penentuan ketiga parameter tersebut berdasarkan nilai *Fitness* terbaik. Langkah selanjutnya adalah melakukan perkawinan silang (*crossover*) antara *alpha* dan *beta*. Kemudian hasil dari perkawinan silang menghasilkan satu individu baru yang kemudian dijadikan nilai delta. Langkah selanjutnya adalah melakukan proses pengepungan dan penyerangan mangsa. Proses akan berulang terus menerus sampai didapatkan nilai *Fitness* yang konvergen atau nilai maksimum iterasi terpenuhi. Adapun parameter dalam algoritma *hybrid GA - GWO* ditunjukkan Tabel 5 . 7 .

Tabel 5 . 7 Parameter *Hybrid GA – Grey Wolve Optimization*

Parameter	Keterangan	Nilai
Search Agents	Jumlah Serigala Abu-Abu	*25*
Max Iteration	Maksimum Iterasi	*10*
Lb	Batas Bawah Parameter	*[0 0 0]*
Ub	Batas Atas Parameter	*[1 1 1]*
CrossoverFcn	Fungsi *Crossover* Individu	*Single Point Crossover*
CrossoverFraction	Nilai Crossover Fraction	*0.5*
MutationFcn	Fungsi Mutasi Individu	*Uniform*

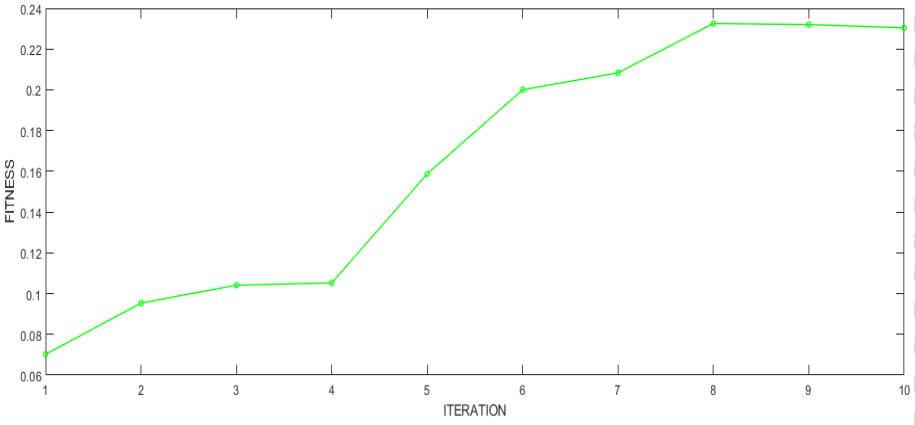
5.2.6 Hasil Pengujian *Hybrid GA - Grey Wolve Optimization*

Hasil pengujian *regulator tegangan otomatis* (AVR) berbasis PID dengan menggunakan metode penalaan *grey wolve optimization* (GWO) ditunjukkan Tabel 5 . 8 .

Tabel 5 . 8 Hasil Pengujian Regulator tegangan otomatis (AVR) Berbasis PID dengan Penalaan Menggunakan Metode *Hybrid GA - Grey Wolve Optimization*

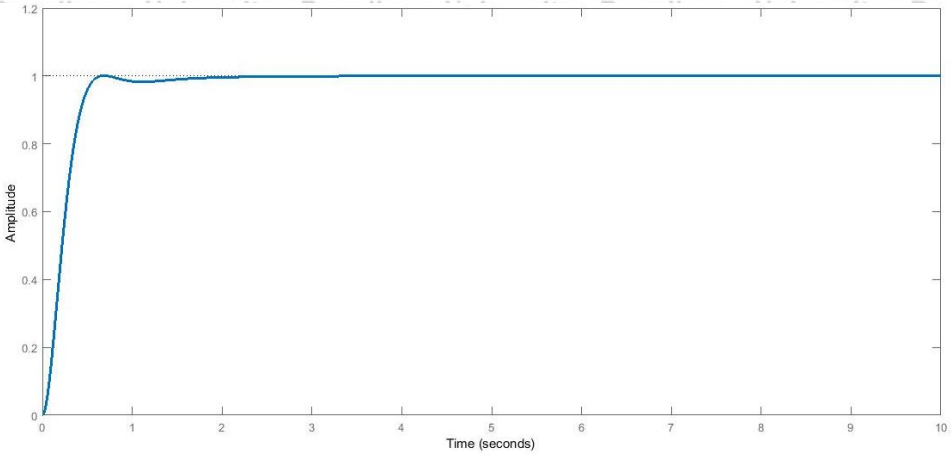
Percobaan	Parameter PID			Rise Time (s)	Settling Time (s)	Maksimum Overshoot (%)	Peak (PU)	Peak Time (s)	Fitness	Elapsed Time (s)
	Kp	Ki	Kd							
1	0.6980	0.5885	0.3720	0.2334	1.0728	1.6556	1.0166	2.3511	0.0977	41.2618
2	0.6284	0.4151	0.2422	0.3267	0.5031	0.4436	1.0044	0.6272	0.2215	38.4502
3	0.6532	0.4880	0.2676	0.3006	0.4566	1.0087	1.0101	0.5819	0.1994	38.3162
4	0.6500	0.4890	0.2713	0.2991	0.4569	0.9952	1.0100	2.1372	0.2000	38.6997
5	**0.6030**	**0.3808**	**0.2242**	**0.3497**	**0.5447**	**0.0909**	**1.0009**	**0.6884**	**0.2323**	**42.7498**
6	0.5835	0.3887	0.2254	0.3571	0.5728	0.1996	1.0020	2.5473	0.2139	38.5451
7	0.6420	0.4397	0.2492	0.3167	0.4827	0.9373	1.0094	0.6264	0.1957	39.3628
8	0.5645	0.3412	0.2050	0.3841	0.6200	0.0000	0.9953	2.6091	0.2111	38.8125
9	0.6867	0.5588	0.2826	0.2819	0.6193	2.4080	1.0241	0.5642	0.1215	38.9736
10	0.6326	0.4618	0.2546	0.3153	0.4845	0.7931	1.0079	2.1491	0.2038	38.5967
								Rata-Rata		**39.3768**

Berdasarkan pengujian yang ditunjukkan Tabel 5 . 8, dapat diketahui bahwa penalaan parameter PID menggunakan metode *hybrid GA - grey wolve optimization* (GWO) didapatkan nilai terbaik pada percobaan ke -5. Hal ini mengacu pada nilai fitness yang terbaik yakni sebesar 0,2323. Adapun nilai parameter Kp = 0,6030, Ki = 0,3808 dan Kd = 0,2242. Proses optimasi nilai Fitness pada percobaan ke -5 ditunjukkan dalam Gambar 5 . 8 .



Gambar 5 . 8 Nilai Fitness Terhadap Iterasi

Berdasarkan proses optimasi yang ditunjukkan dalam Gambar 5 . 8 dapat diketahui bahwa nilai fitness terbaik pada saat optimasi sebesar 0,2323 dan waktu pemrosesan yang dibutuhkan (*elapsed time*) sebesar 42,7498 detik. Adapun hasil pengujian respon transien sistem *regulator tegangan otomatis* (AVR) berbasis PID dengan memberikan masukan unit step ditunjukkan Gambar 5 . 9.



Gambar 5 . 9 Hasil Pengujian unit step *Regulator tegangan otomatis* (AVR) berbasis PID Dengan Penalaan *Hybrid GA - Grey Wolve Optimization*

5.3 Pembahasan Hasil Pengujian *Regulator tegangan otomatis* (AVR) Berbasis PID Menggunakan Metode *Artificial Intelligent* (AI)

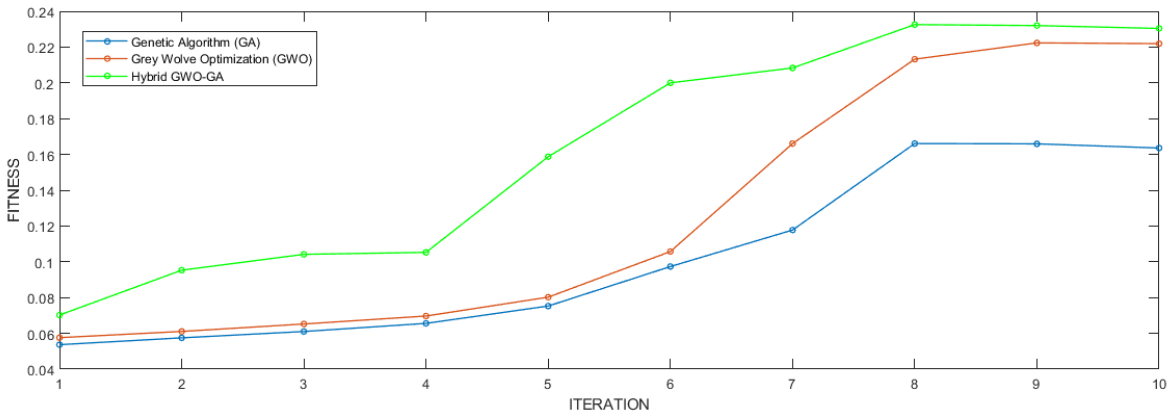
Berdasarkan hasil pengujian *regulator tegangan otomatis* (AVR) berbasis kontroller PID dengan menggunakan metode *GA, grey wolve optimization* dan *hybrid GA – grey wolve optimization* yang telah dilakukan mempunyai karakteristik yang berbeda-beda. Adapun ringkasan hasil pengujian ketiga metode tersebut ditunjukkan Tabel 5 . 9 .

Tabel 5 . 9 Ringkasan Hasil Pengujian GA, GWO dan GA-GWO Pada Sistem *Regulator tegangan otomatis* (AVR) Berbasis PID

	Parameter PID			Rise Time (s)	Settling Time (s)	Maksimum Overshoot (%)	Peak (PU)	Peak Time (s)	Fitness	Elapsed Time (s)
	Kp	Ki	Kd							
GA (GA)	0.6477	0.4592	0.2476	0.3149	0.4756	1.5813	1.0158	0.6283	0.1661	30.7106
Grey Wolve Optimization (GWO)	0.6278	0.4316	0.2458	0.3238	0.4986	0.4516	1.0045	0.6239	0.2219	39.5309
Hybrid GA-GWO	0.6030	0.3808	0.2242	0.3497	0.5447	0.0909	1.0009	0.6884	0.2323	42.7498

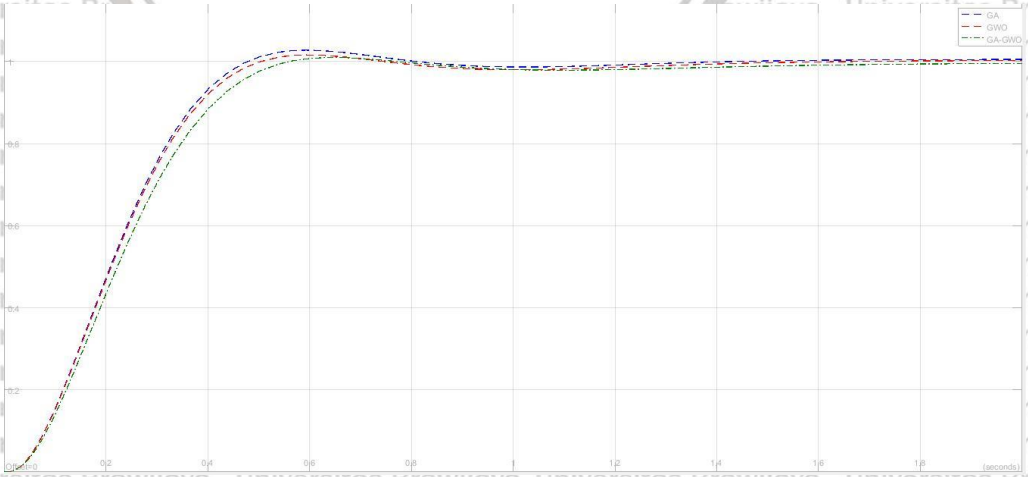
Berdasarkan pengujian ketiga metode yang ditunjukkan tabel 5.9 dapat disimpulkan bahwa nilai fitness terbaik diperoleh ketika menggunakan metode *hybrid GA – grey wolve optimization* dengan nilai 0.2323. Sehingga dapat disimpulkan bahwa dengan menggunakan metode hybrid GA-GWO diperoleh respon transien terbaik dibandingkan dengan metode GA dan GWO. Hal ini mengacu pada nilai fitness tertinggi yang dicapai oleh masing-masing metode. Adapun kelemahan dari metode hybrid GA-GWO adalah

mempunyai waktu eksekusi proses (*elapsed time*) yang lebih tinggi dibandingkan dengan metode GA dan GWO. Hal ini dikarenakan pada proses *hybrid* GA-GWO terdapat penambahan fungsi *crossover* dan mutasi yang mengakibatkan penambahan waktu eksekusi. Adapun karakteristik proses pencarian nilai Fitness pada setiap iterasi ditunjukkan Gambar 5 . 10.

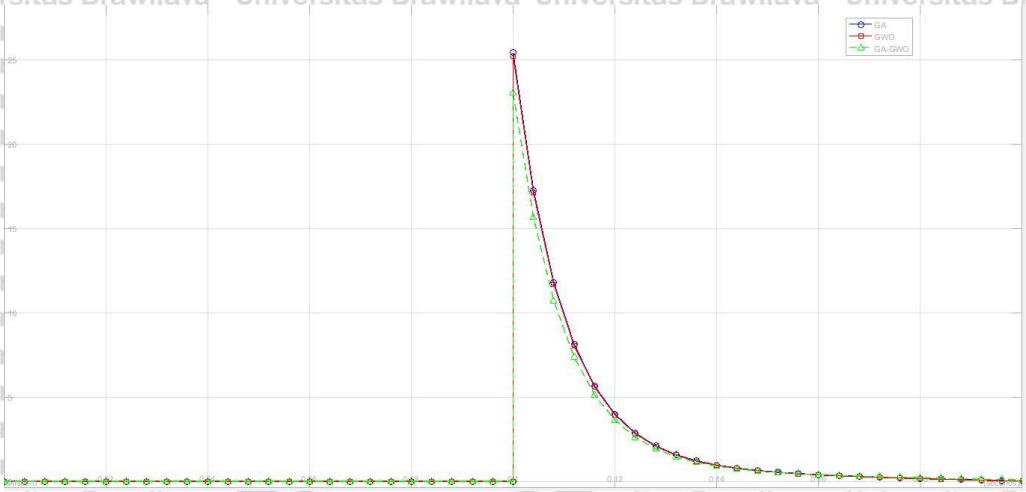


Gambar 5 . 10 Perbandingan Nilai Fitness Terhadap Iterasi Pada Metode GA, GWO dan *Hybrid* GA-GWO

Berdasarkan perbandingan karakteristik proses pencarian nilai Fitness pada setiap metode, dapat diketahui bahwa metode hybrid GA-GWO memiliki nilai Fitness yang relatif tinggi pada setiap iterasi dibandingkan dengan metode GA dan GWO. Hal ini dikarenakan pada metode *hybrid* GA-GWO mengkombinasikan proses pencarian secara deterministik yang dimiliki metode GWO dan probabilistik yang dimiliki metode GA. Sehingga proses pencarian nilai fitness pada metode *hybrid* GA-GWO lebih optimal dibandingkan dengan metode GA dan GWO. Adapun perbandingan respon transien dan sinyal kontrol PID pada ketiga metode ditunjukkan Gambar 5.11. dan 5.12.



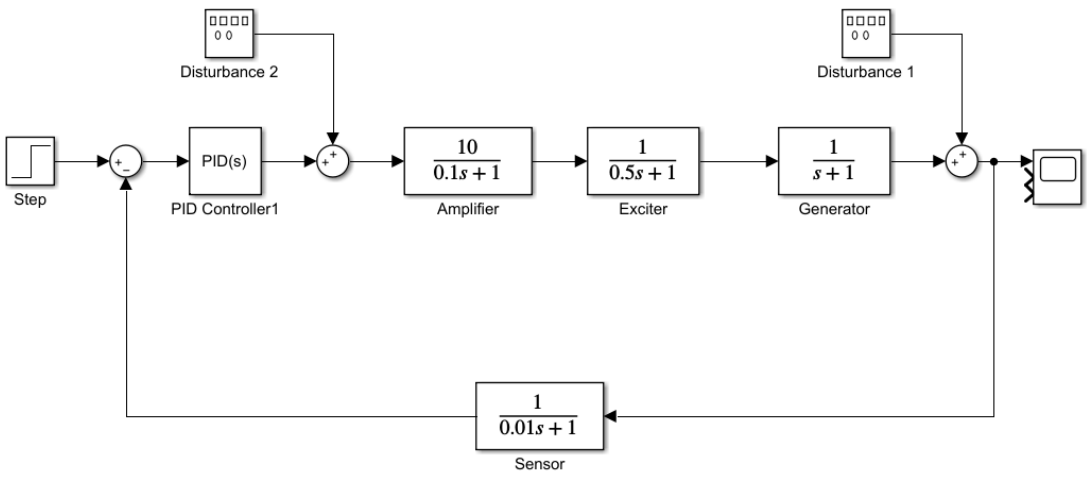
Gambar 5 . 11 Perbandingan Respon Transien Sistem *Regulator tegangan otomatis* (AVR) Pada Metode GA, GWO dan *Hybrid* GA-GWO



Gambar 5 . 12 Perbandingan Sinyal Kontrol PID Pada Metode GA, GWO dan *Hybrid* GA-GWO

5.4 Pengujian Pembebanan *Regulator tegangan otomatis* (AVR)

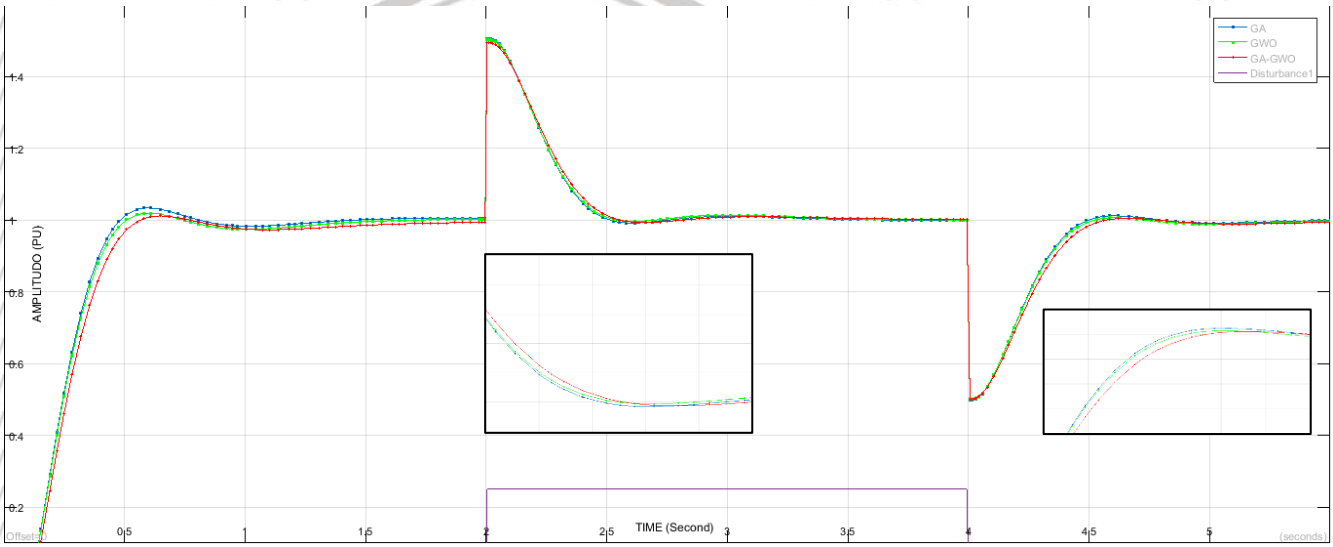
Pengujian pemberian beban pada *regulator tegangan otomatis* (AVR) diperlukan untuk mengetahui respon transien sistem dalam mencapai nilai setpoint ketika terdapat *disturbance* (gangguan) dikarenakan efek pembebanan. Selain itu juga untuk mengetahui perbandingan respon kontrol metode *artificial intelligent* (AI) yang diimplementasi pada sistem kontrol AVR. Pada pengujian pembebanan *regulator tegangan otomatis* (AVR) dilakukan dengan cara memberikan sinyal gangguan (*disturbance*) pada sistem. Adapun pemberian sinyal *disturbance* diletakkan pada titik keluaran sistem dan pada titik keluaran kontroller. Blok diagram pengujian pembebanan *regulator tegangan otomatis* (AVR) ditunjukkan Gambar 5 . 13.



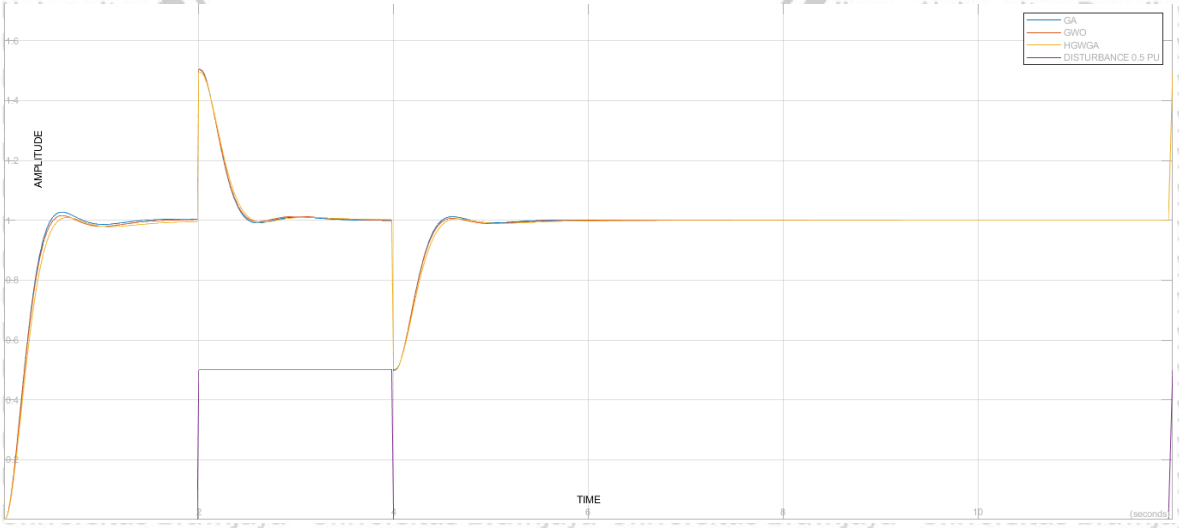
Gambar 5 . 13 Blok Diagram Pengujian Pembebanan *Regulator tegangan otomatis* (AVR)

5.4.1 Hasil Pengujian Pemberian *Disturbance 1*

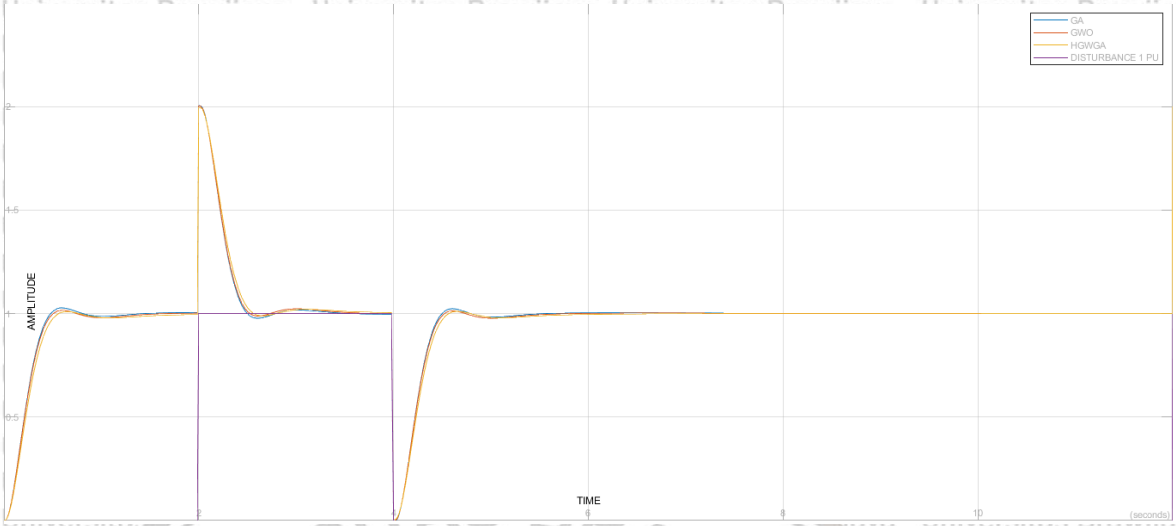
Hasil pengujian pemberian beban / gangguan (*disturbance 1*) dilakukan dengan cara memberikan sinyal gangguan pada posisi keluaran sistem AVR (*Sum1*). Pada hasil pengujian ini, penulis membandingkan respon transien yang dihasilkan oleh ketiga metode *artificial intelligent* (AI) yakni *GA* (GA), *grey wolve optimization* (GWO) dan *Hybrid* GA-GWO ketika diberikan gangguan (*disturbance*). Nilai sinyal gangguan pada pengujian ini sebesar 0.25 PU, 0.5 PU dan 1 PU.



Gambar 5 . 14 Hasil Pengujian Pemberian *Disturbance 1* Dengan Nilai 0.25 PU



Gambar 5 . 15 Hasil Pengujian Pemberian *Disturbance 1* Dengan Nilai 0.5 PU

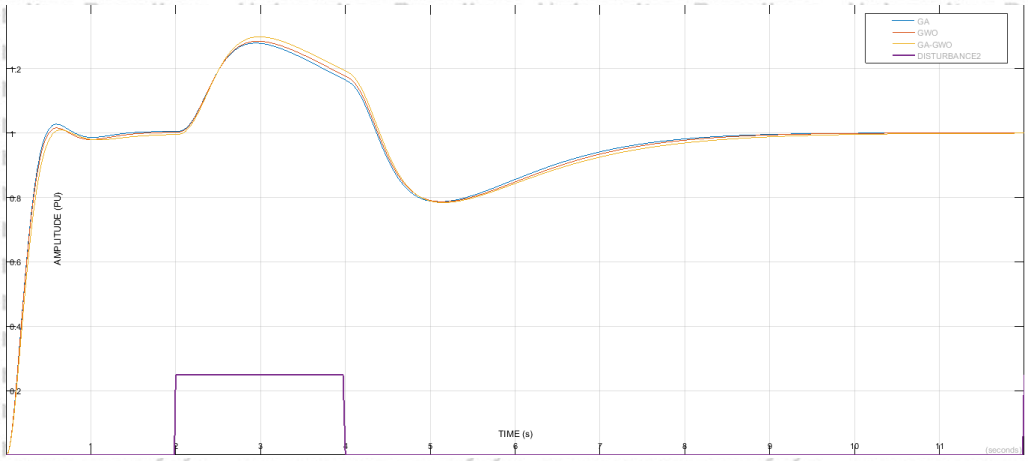


Gambar 5 . 16 Hasil Pengujian Pemberian *Disturbance 1* Dengan Nilai 1 PU

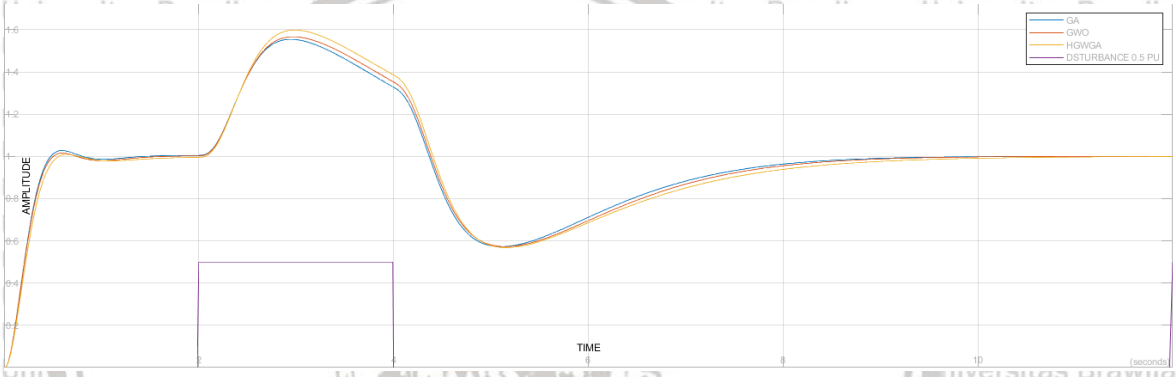
Pada hasil pengujian pemberian *disturbance 1* yang ditunjukkan dalam Gambar 5 . 14 sampai 5 . 16 dapat diketahui bahwa *regulator tegangan otomatis* (AVR) dapat menstabilkan respon sistem ketika diberikan gangguan. Sinyal *disturbance* terlihat mempengaruhi respon sistem pada saat *time rise* dan *time fall*. Pengujian pemberian *disturbance 1* merepresentasikan pemberian beban pada sistem *regulator tegangan otomatis* (AVR).

5.4.2 Hasil Pengujian Pemberian *Disturbance 2*

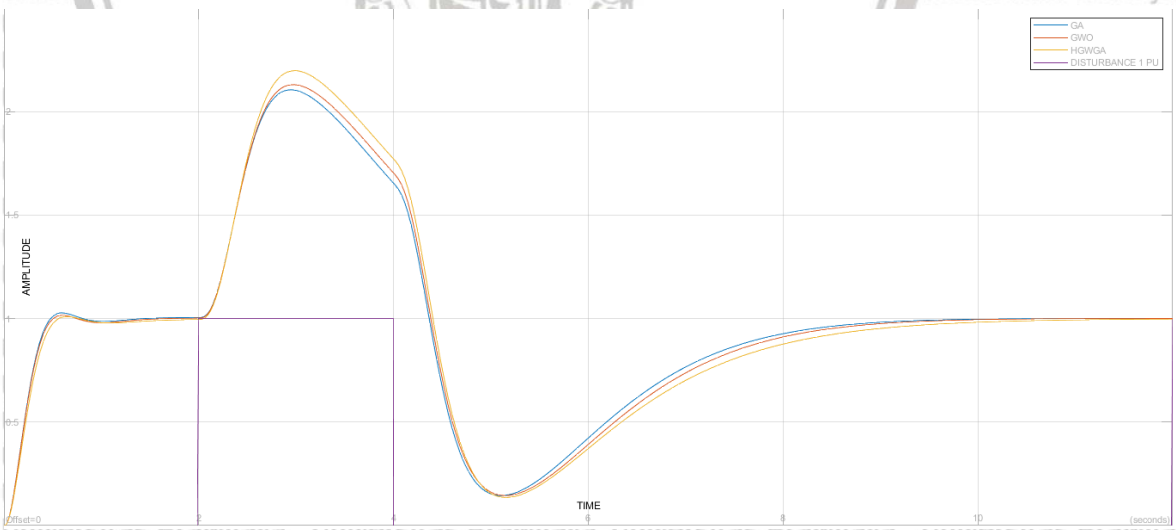
Hasil pengujian pemberian beban / gangguan (*disturbance 2*) dilakukan dengan cara memberikan sinyal gangguan pada posisi keluaran PID kontroller (*Sum2*). Pada hasil pengujian ini, penulis membandingkan respon transien yang dihasilkan oleh ketiga metode *artificial intelligent* (AI) yakni *GA* (GA), *grey wolve optimization* (GWO) dan *Hybrid* GA-GWO ketika diberikan gangguan (*disturbance*). Nilai sinyal gangguan pada pengujian ini sebesar 0.25 PU, 0.5 PU dan 1 PU.



Gambar 5 . 17 Hasil Pengujian Pemberian *Disturbance 2* Dengan Nilai 0.25 PU



Gambar 5 . 18 Hasil Pengujian Pemberian *Disturbance 2* Dengan Nilai 0.5 PU



Gambar 5 . 19 Hasil Pengujian Pemberian *Disturbance 2* Dengan Nilai 1 PU

Pada hasil pengujian pemberian *disturbance 2* yang ditunjukkan dalam Gambar 5 . 17 sampai 5 . 19 dapat diketahui bahwa *regulator tegangan otomatis*

(AVR) dapat menstabilkan respon sistem ketika diberikan gangguan. Sinyal *disturbance* terlihat mempengaruhi respon sistem pada saat *time rise* dan *time fall*.

Pengujian pemberian *disturbance 2* merepresentasikan noise yang terjadi pada sinyal kontrol PID.

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Optimisasi *regulator tegangan otomatis* (AVR) berbasis kontroller PID dengan menggunakan metode *Hybrid GA – Grey Wolve Optimization*, maka dapat disimpulkan sebagai berikut:

1. Sistem kontrol *regulator tegangan otomatis* (AVR) menggunakan metode PID sebagai kontroller utama dalam menjaga kestabilan tegangan keluaran generator. Dengan penalaan parameter Kp, Ki dan Kd menggunakan metode *artificial intelligent* (GA-GWO-HGWGA).
2. Respon transien sistem *regulator tegangan otomatis* (AVR) ketika menggunakan metode kontrol PID dengan menggunakan metode penalaan (GA-GW-HGWGA) memeiliki respon yang relatif baik dengan parameter penilaian *Mp* (*Maximum Overshoot*), *tr* (*Time Rise*), *ts* (*Time Settling*) dan *Ess* (*Error Steady State*).
3. Respon transien sistem *regulator tegangan otomatis* (AVR) dengan metode kontrol PID memiliki respon terbaik ketika menggunakan penalaan HGWGA. Kemudian yang kedua dan ketiga, ketika menggunakan penalaan GWO dan GA.

6.2 Saran

Proses optimasi *regulator tegangan otomatis* (AVR) menggunakan metode *hybrid GA – grey wolve optimization* (HGAGW) menghasilkan perbaikan respon transient dibandingkan dengan metode *GA* (GA) dan *grey wolve optimization* (GWO). Karena itu, pada penelitian selanjutnya diharapkan dapat menerapkan modifikasi pada sistem HGAGW agar didapatkan hasil yang lebih optimal. Yang kedua diharapkan dapat mencari kombinasi konstanta faktor pengali pada fungsi Fitness, sehingga didapatkan nilai respon transient yang lebih baik.

DAFTAR PUSTAKA

Ahmed M. Mosaad, Mahmoud A.A., Almoataz Y.A. 2019. *Whale optimization algorithm to tune PID and PIDA controllers on AVR system*. Ain Shams Engineering Journal 10 (2019) 755-767.

Bahran ,Teuku B. (2017). *Konsumsi Energi Listrik, Pertumbuhan Ekonomi dan penduduk terhadap Emisi Gas Rumah Kaca Pembangkit Listrik di Indonesia*. Skripsi. Universitas Syiah Kuala

Busra O. 2019. *Optimally Tuned PID Controller Design for an AVR System : A Comparison Study*. Internationa Journal of Multidisciplinary Studies and Innovative Technologies 3 (2019) 157-161.

Ching-Chang W. 2009. *Optimal PID controller design for AVR system*". Tamkang Journal of Science and Engineering,Vol.12,No.3,pp.259270.

Endriyanto NW. 2011. *Perencanaan Optimal Sistem Kontrol AVR (Automatic Voltage Regulator) Untuk Memperbaiki Kestabilan Tegangan Dengan Menggunakan Algoritma Genetik*. Jurusan Teknik Elektro, Fakultas Teknik Universitas Diponegoro Semarang

Mitsuo Gen. 1997. *Genetic Algorithm for Solving Shortest Path Problems*". Department of Industrial & System Enggineering Ashikaga Institute of Technology, Ashikaga 326, Japan.

PT. PLN (Persero). (2015). *35000MW Untuk Indonesia Laporan Tahunan 2015*, (Online), (http://www.pln.co.id, diakses 23 Januari 2021)

Rvindra Kumar K. 2019. *Grey Wolf Optimization Algorithm based PID controller design for AVR Power system.* 2nd International Conference on Power Energy, Environment and Intelligent Control (PEEIC).

Ziegler J. G. and N. B. Nichols. 1942. *Optimum settings for automatic controllers*. Trans. ASME, vol/issue: 64(8), pp.759/768–759/768.

LAMPIRAN A

Listing Program

Lampiran A.1 (Genetic Algorithm)

Script gapid.m

```
%{
This code is adapted from the lectures of Dr. Steven Brunton.
The video lectures of Dr. Steven is available on
https://www.youtube.com/watch?v=S5C_z1nVaSg
%}
clear all, close all, clc
tic

dt = 0.01;
PopSize = 25;
MaxGenerations = 10;
s = tf('s');
Amp = 10 / (0.1*s + 1);
Exc = 1 / (0.5*s + 1);
Gen = 1 / (s + 1);
%Sen = 1 / (0.01*s + 1);
Loop1 = series(Amp,Exc);
G = series(Loop1,Gen);

%G = (0.1*s+10) / (0.0004*s^4 + 0.0454*s^3 + 0.555*s^2 + 1.51*s + 11)

% Transfer function of a third order system with poles at -1, -2 , -3
% G = 1/((s+1)*(s+2)*(s+3))


% Transfer function of the DC motor
%G = tf(0.9, [0.00105 0.2104 0.8913 0]);


options =
optimoptions('ga','PlotFcn',{@gaplotbestf},'PopulationSize',PopSize,'MaxG
enerations',MaxGenerations,'SelectionFcn',{@selectionuniform},'MutationFc
n',{@mutationuniform},'OutputFcn',@myfun,'CrossoverFcn',{@crossoversingle
point});%optimoptions(@ga,'PopulationSize',PopSize,'MaxGenerations',MaxGe
nerations,'OutputFcn',@myfun);  'ga','PlotFcn',{@gaplotbestf}
[x,fval,exitflag] = ga(@(K)pidtest(G,dt,K),3,[],[],[],[],[0 0 0],[1 1
1],[],options);%ga(@(K)pidtest(G,dt,K),3,-
eye(3),zeros(3,1),[],[],[],[],[],options);
toc
```

Lampiran A.2 (Genetic Algorithm)

Script myfun.m

```
function [state, options,optchanged] = myfun(options,state,flag)
persistent history
persistent cost
optchanged = false;

switch flag
 case 'init'
        history(:,:,1) = state.Population;
        cost(:,1) = state.Score;
    case {'iter','interrupt'}
        ss = size(history,3);
        history(:,:,ss+1) = state.Population;
        cost(:,ss+1) = state.Score;
    case 'done'
        ss = size(history,3)
        history(:,:,ss+1) = state.Population;
        cost(:,ss+1) = state.Score;
        cd ('E:\ANAS\THESIS\JUDUL\MATLAB CODE\GA-PID-master\GA-PID-
master')
        save hasil.mat history cost
        %save ('history')
end
```

Lampiran A.3 (Genetic Algorithm)

Script pidtest.m

```
function J = pidtest(G,dt,parms)
s = tf('s');
K = parms(1) + parms(2)/s + parms(3)*s/(1+.001*s);
Sen = 1 / (0.01*s + 1);
Loop4 = series(K,G);
ClosedLoop = feedback(Loop4,Sen);
t = 0:dt:5;
[y,t] = step(ClosedLoop,t);
Hasil = stepinfo(ClosedLoop);
e = (1 - step(ClosedLoop,t))
TimeRise = Hasil.RiseTime
Overshoot = Hasil.Overshoot
TimeSettling = Hasil.SettlingTime
ess = abs(1-y(end))
% Cost function based on ISE, IAE and ITAE
 w1 = 1; w2 = 1; w3 = 1; w4 = 1.5; w5 = 15; w6 = 7; w7 = 1;
ITAE_Modif = (w3*trapz(t, dt.*abs(e)))+ w4*Overshoot + w5*ess +
w6*TimeSettling + w7*TimeRise
J = ITAE_Modif
```

Lampiran B.1 (Grey Wolve Optimization)

Script main.m

```
%_____________________________________________________________%
%  Grey Wolf Optimizer (GWO) source codes version 1.0         %
%                                                             %
%  Developed in MATLAB R2011b(7.13)                           %
%  Author and programmer: Seyedali Mirjalili                  %
%                                                             %
%         e-Mail: ali.mirjalili@gmail.com                     %
%                 seyedali.mirjalili@griffithuni.edu.au       %
%                                                             %
%       Homepage: http://www.alimirjalili.com                 %
%                                                             %
%    Main paper: S. Mirjalili, S. M. Mirjalili, A. Lewis      %
%                Grey Wolf Optimizer, Advances in Engineering %
%                Software , in press,                         %
%                DOI: 10.1016/j.advengsoft.2013.12.007        %
%                                                             %
%_____________________________________________________________%

% You can simply define your cost in a seperate file and load its handle
to fobj
% The initial parameters that you need are:
%_____________________________________
% fobj = @YourCostFunction
% dim = number of your variables
% Max_iteration = maximum number of generations
% SearchAgents_no = number of search agents
% lb=[lb1,lb2,...,lbn] where lbn is the lower bound of variable n
% ub=[ub1,ub2,...,ubn] where ubn is the upper bound of variable n
% If all the variables have equal lower bound you can just
% define lb and ub as two single number numbers

% To run GWO:
[Best_score,Best_pos,GWO_cg_curve]=GWO(SearchAgents_no,Max_iteration,lb,u
b,dim,fobj)
%_____________________________________

clear all
clc

SearchAgents_no=30; % Number of search agents

Function_name='F10'; % Name of the test function that can be from F1 to
F23 (Table 1,2,3 in the paper)

Max_iteration=500; % Maximum numbef of iterations

% Load details of the selected benchmark function
[lb,ub,dim,fobj]=Get_Functions_details(Function_name);

[Best_score,Best_pos,GWO_cg_curve]=GWO(SearchAgents_no,Max_iteration,lb,u
b,dim,fobj);

figure('Position',[500 500 660 290])
%Draw search space
```

```
subplot(1,2,1);
func_plot(Function_name);
title('Parameter space')
xlabel('x_1');
ylabel('x_2');
zlabel([Function_name,'( x_1 , x_2 )'])

%Draw objective space
subplot(1,2,2);
semilogy(GWO_cg_curve,'Color','r')
title('Objective space')
xlabel('Iteration');
ylabel('Best score obtained so far');

axis tight
grid on
box on
legend('GWO')

display(['The best solution obtained by GWO is : ', num2str(Best_pos)]);
display(['The best optimal value of the objective funciton found by GWO
is : ', num2str(Best_score)]);
```

Lampiran B.2 (Grey Wolve Optimization)

Script func_plot.m

```
%_____________________________________________________________%
%  Grey Wolf Optimizer (GWO) source codes version 1.0         %
%                                                             %
%  Developed in MATLAB R2011b(7.13)                           %
%                                                             %
%  Author and programmer: Seyedali Mirjalili                  %
%                                                             %
%         e-Mail: ali.mirjalili@gmail.com                     %
%                 seyedali.mirjalili@griffithuni.edu.au       %
%                                                             %
%       Homepage: http://www.alimirjalili.com                 %
%                                                             %
%    Main paper: S. Mirjalili, S. M. Mirjalili, A. Lewis      %
%                Grey Wolf Optimizer, Advances in Engineering %
%                Software , in press,                         %
%                DOI: 10.1016/j.advengsoft.2013.12.007        %
%                                                             %
%_____________________________________________________________%

% This function draw the benchmark functions

function func_plot(func_name)

[lb,ub,dim,fobj]=Get_Functions_details(func_name);

switch func_name
    case 'F1'
        x=-100:2:100; y=x; %[-100,100]

    case 'F2'
        x=-100:2:100; y=x; %[-10,10]

    case 'F3'
        x=-100:2:100; y=x; %[-100,100]

    case 'F4'
        x=-100:2:100; y=x; %[-100,100]
    case 'F5'
        x=-200:2:200; y=x; %[-5,5]
    case 'F6'
        x=-100:2:100; y=x; %[-100,100]
    case 'F7'
        x=-1:0.03:1;   y=x   %[-1,1]
    case 'F8'
        x=-500:10:500;y=x; %[-500,500]
    case 'F9'
        x=-5:0.1:5;    y=x; %[-5,5]
    case 'F10'
        x=-20:0.5:20; y=x;%[-500,500]
    case 'F11'
        x=-500:10:500; y=x;%[-0.5,0.5]
    case 'F12'
        x=-10:0.1:10; y=x;%[-pi,pi]
    case 'F13'
        x=-5:0.08:5; y=x;%[-3,1]
```

```
    case 'F14'
        x=-100:2:100; y=x;%[-100,100]
    case 'F15'
        x=-5:0.1:5; y=x;%[-5,5]
    case 'F16'
        x=-1:0.01:1; y=x;%[-5,5]
    case 'F17'
        x=-5:0.1:5; y=x;%[-5,5]
    case 'F18'
        x=-5:0.06:5; y=x;%[-5,5]
    case 'F19'
        x=-5:0.1:5; y=x;%[-5,5]
    case 'F20'
        x=-5:0.1:5; y=x;%[-5,5]
    case 'F21'
        x=-5:0.1:5; y=x;%[-5,5]
    case 'F22'
        x=-5:0.1:5; y=x;%[-5,5]
    case 'F23'
        x=-5:0.1:5; y=x;%[-5,5]
    case 'F24'
        x=0:0.01:1; y=0:0.01:1; z=0:0.01:1;
end
L=length(x);
f=[];

for i=1:L
    for j=1:L
         for k=1:L
        if strcmp(func_name,'F15')==1
            f(i,j)=fobj([x(i),y(j),0,0]);
        end
        if strcmp(func_name,'F19')==1
            f(i,j)=fobj([x(i),y(j),0]);
        end
        if strcmp(func_name,'F20')==1
            f(i,j)=fobj([x(i),y(j),0,0,0,0]);
        end
        if strcmp(func_name,'F21')==1 || strcmp(func_name,'F22')==1
||strcmp(func_name,'F23')==1
            f(i,j)=fobj([x(i),y(j),0,0]);
        end
        if strcmp(func_name,'F24')==1
            f(i,j,k)=fobj([x(i),y(j),z(k)]);
        end
        end
    end
end
End
```

Lampiran B.3 (Grey Wolve Optimization)

Script Get_Function_details.m

```
%_____________________________________________________________________%
%  Grey Wolf Optimizer (GWO) source codes version 1.0               %
%                                                                     %
%  Developed in MATLAB R2011b(7.13)                                 %
%                                                                     %
%  Author and programmer: Seyedali Mirjalili                        %
%                                                                     %
%         e-Mail: ali.mirjalili@gmail.com                           %
%                 seyedali.mirjalili@griffithuni.edu.au             %
%                                                                     %
%       Homepage: http://www.alimirjalili.com                       %
%                                                                     %
%    Main paper: S. Mirjalili, S. M. Mirjalili, A. Lewis            %
%                Grey Wolf Optimizer, Advances in Engineering       %
%                Software , in press,                               %
%                DOI: 10.1016/j.advengsoft.2013.12.007              %
%                                                                     %
%_____________________________________________________________________%

% This function containts full information and implementations of the
benchmark
% functions in Table 1, Table 2, and Table 3 in the paper

% lb is the lower bound: lb=[lb_1,lb_2,...,lb_d]
% up is the uppper bound: ub=[ub_1,ub_2,...,ub_d]
% dim is the number of variables (dimension of the problem)

function [lb,ub,dim,fobj] = Get_Functions_details(F)


switch F
    case 'F1'
        fobj = @F1;
        lb=-100;
        ub=100;
        dim=30;

    case 'F2'
        fobj = @F2;
        lb=-10;
        ub=10;
        dim=30;

    case 'F3'
        fobj = @F3;
        lb=-100;
        ub=100;
        dim=30;

    case 'F4'
        fobj = @F4;
        lb=-100;
        ub=100;
        dim=30;
```

```
    case 'F5'
        fobj = @F5;
        lb=-30;
        ub=30;
        dim=30;
    case 'F6'
        fobj = @F6;
        lb=-100;
        ub=100;
        dim=30;
    case 'F7'
        fobj = @F7;
        lb=-1.28;
        ub=1.28;
        dim=30;
    case 'F8'
        fobj = @F8;
        lb=-500;
        ub=500;
        dim=30;
    case 'F9'
        fobj = @F9;
        lb=-5.12;
        ub=5.12;
        dim=30;
    case 'F10'
        fobj = @F10;
        lb=-32;
        ub=32;
        dim=30;
    case 'F11'
        fobj = @F11;
        lb=-600;
        ub=600;
        dim=30;
    case 'F12'
        fobj = @F12;
        lb=-50;
        ub=50;
        dim=30;
    case 'F13'
        fobj = @F13;
        lb=-50;
        ub=50;
        dim=30;
    case 'F14'
        fobj = @F14;
        lb=-65.536;
        ub=65.536;
```

```
        dim=2;
    case 'F15'
        fobj = @F15;
        lb=-5;
        ub=5;
        dim=4;
    case 'F16'
        fobj = @F16;
        lb=-5;
        ub=5;
        dim=2;
    case 'F17'
        fobj = @F17;
        lb=[-5,0];
        ub=[10,15];
        dim=2;
    case 'F18'
        fobj = @F18;
        lb=-2;
        ub=2;
        dim=2;

    case 'F19'
        fobj = @F19;
        lb=0;
        ub=1;
        dim=3;

    case 'F20'
        fobj = @F20;
        lb=0;
        ub=1;
        dim=6;

    case 'F21'
        fobj = @F21;
        lb=0;
        ub=10;
        dim=4;

    case 'F22'
        fobj = @F22;
        lb=0;
        ub=10;
        dim=4;
    case 'F23'
        fobj = @F23;
        lb=0;
        ub=10;
        dim=4;
    case 'F24'
        fobj = @F24;
        lb = 0,0,0;
```

```
        ub = 1,1,1;
        dim = 3;
end
end
% F1
function o = F1(x)
o=sum(x.^2);
end
% F2
function o = F2(x)
o=sum(abs(x))+prod(abs(x));
end
% F3
function o = F3(x)
dim=size(x,2);
o=0;
for i=1:dim
    o=o+sum(x(1:i))^2;
end
end
% F4
function o = F4(x)
o=max(abs(x));
end
% F5
function o = F5(x)
dim=size(x,2);
o=sum(100*(x(2:dim)-(x(1:dim-1).^2)).^2+(x(1:dim-1)-1).^2);
end
% F6
function o = F6(x)
o=sum(abs((x+.5)).^2);
end
% F7
function o = F7(x)
dim=size(x,2);
o=sum([1:dim].*(x.^4))+rand;
end
% F8
function o = F8(x)
o=sum(-x.*sin(sqrt(abs(x))));
```

```
end

% F9

function o = F9(x)
dim=size(x,2);
o=sum(x.^2-10*cos(2*pi.*x))+10*dim;
end

% F10

function o = F10(x)
dim=size(x,2);
o=-20*exp(-.2*sqrt(sum(x.^2)/dim))-exp(sum(cos(2*pi.*x))/dim)+20+exp(1);
end

% F11

function o = F11(x)
dim=size(x,2);
o=sum(x.^2)/4000-prod(cos(x./sqrt([1:dim])))+1;
end

% F12

function o = F12(x)
dim=size(x,2);
o=(pi/dim)*(10*((sin(pi*(1+(x(1)+1)/4)))^2)+sum((((x(1:dim-
1)+1)./4).^2).*...
(1+10.*((sin(pi.*(1+(x(2:dim)+1)./4)))).^2))+((x(dim)+1)/4)^2)+sum(Ufun(x
,10,100,4));
end

% F13

function o = F13(x)
dim=size(x,2);
o=.1*((sin(3*pi*x(1)))^2+sum((x(1:dim-1)-
1).^2.*(1+(sin(3.*pi.*x(2:dim))).^2))+...
((x(dim)-1)^2)*(1+(sin(2*pi*x(dim)))^2))+sum(Ufun(x,5,100,4));
end

% F14

function o = F14(x)
aS=[-32 -16 0 16 32 -32 -16 0 16 32 -32 -16 0 16 32 -32 -16 0 16 32 -32 -
16 0 16 32;,...
-32 -32 -32 -32 -32 -16 -16 -16 -16 -16 0 0 0 0 0 16 16 16 16 16 32 32 32
32 32];
for j=1:25
    bS(j)=sum((x'-aS(:,j)).^6);
end
o=(1/500+sum(1./([1:25]+bS))).^(-1);
end

% F15
```

```
function o = F15(x)
aK=[.1957 .1947 .1735 .16 .0844 .0627 .0456 .0342 .0323 .0235 .0246];
bK=[.25 .5 1 2 4 6 8 10 12 14 16];bK=1./bK;
o=sum((aK-((x(1).*(bK.^2+x(2).*bK))./(bK.^2+x(3).*bK+x(4)))).^2);
end

% F16

function o = F16(x)
o=4*(x(1)^2)-2.1*(x(1)^4)+(x(1)^6)/3+x(1)*x(2)-4*(x(2)^2)+4*(x(2)^4);
end

% F17

function o = F17(x)
o=(x(2)-(x(1)^2)*5.1/(4*(pi^2))+5/pi*x(1)-6)^2+10*(1-
1/(8*pi))*cos(x(1))+10;
end

% F18

function o = F18(x)
o=(1+(x(1)+x(2)+1)^2*(19-14*x(1)+3*(x(1)^2)-
14*x(2)+6*x(1)*x(2)+3*x(2)^2))*...
    (30+(2*x(1)-3*x(2))^2*(18-32*x(1)+12*(x(1)^2)+48*x(2)-
36*x(1)*x(2)+27*(x(2)^2)));
end

% F19

function o = F19(x)
aH=[3 10 30;.1 10 35;3 10 30;.1 10 35];cH=[1 1.2 3 3.2];
pH=[.3689 .117 .2673;.4699 .4387 .747;.1091 .8732 .5547;.03815 .5743
.8828];
o=0;
for i=1:4
    o=o-cH(i)*exp(-(sum(aH(i,:).*((x-pH(i,:)).^2))));
end
end

% F20

function o = F20(x)
aH=[10 3 17 3.5 1.7 8;.05 10 17 .1 8 14;3 3.5 1.7 10 17 8;17 8 .05 10 .1
14];
cH=[1 1.2 3 3.2];
pH=[.1312 .1696 .5569 .0124 .8283 .5886;.2329 .4135 .8307 .3736 .1004
.9991;...
.2348 .1415 .3522 .2883 .3047 .6650;.4047 .8828 .8732 .5743 .1091 .0381];
o=0;
for i=1:4
    o=o-cH(i)*exp(-(sum(aH(i,:).*((x-pH(i,:)).^2))));
end
end

% F21

function o = F21(x)
```

```
aSH=[4 4 4 4;1 1 1 1;8 8 8 8;6 6 6 6;3 7 3 7;2 9 2 9;5 5 3 3;8 1 8 1;6 2
6 2;7 3.6 7 3.6];
cSH=[.1 .2 .2 .4 .4 .6 .3 .7 .5 .5];

o=0;
for i=1:5
    o=o-((x-aSH(i,:))*(x-aSH(i,:))'+cSH(i))^(-1);
end
end

% F22

function o = F22(x)
aSH=[4 4 4 4;1 1 1 1;8 8 8 8;6 6 6 6;3 7 3 7;2 9 2 9;5 5 3 3;8 1 8 1;6 2
6 2;7 3.6 7 3.6];
cSH=[.1 .2 .2 .4 .4 .6 .3 .7 .5 .5];

o=0;
for i=1:7
    o=o-((x-aSH(i,:))*(x-aSH(i,:))'+cSH(i))^(-1);
end
end

% F23

function o = F23(x)
aSH=[4 4 4 4;1 1 1 1;8 8 8 8;6 6 6 6;3 7 3 7;2 9 2 9;5 5 3 3;8 1 8 1;6 2
6 2;7 3.6 7 3.6];
cSH=[.1 .2 .2 .4 .4 .6 .3 .7 .5 .5];

o=0;
for i=1:10
    o=o-((x-aSH(i,:))*(x-aSH(i,:))'+cSH(i))^(-1);
end
end


function o = F24(x)
dim=size(x,3);
dt = 0.01;

s = tf('s');
K = x(1) + x(2)/s + x(3)*s/(1+.001*s);
Amp = 10 / (0.1*s + 1);
Exc = 1 / (0.5*s + 1);
Gen = 1 / (s + 1);
Sen = 1 / (0.01*s + 1);
Loop1 = series(K,Amp);
Loop2 = series(Exc,Gen);
Loop3 = series(Loop1,Loop2);
G = feedback(Loop3,Sen);

t = 0:dt:5;
[y,t] = step(G,t);
Hasil = stepinfo(G);

e = (1 - step(G,t));
TimeRise = Hasil.RiseTime;
Overshoot = Hasil.Overshoot;
TimeSettling = Hasil.SettlingTime;
```

```
ess = abs(1-y(end));

w1 = 1; w2 = 1; w3 = 1; w4 = 1.5; w5 = 15; w6 = 7; w7 = 1;
ITAE_Modif = (w3*trapz(t, dt.*abs(e)))+ w4*Overshoot + w5*ess +
w6*TimeSettling + w7*TimeRise;
o = ITAE_Modif;
end

function o=Ufun(x,a,k,m)
o=k.*((x-a).^m).*(x>a)+k.*((-x-a).^m).*(x<(-a));
end
```

Lampiran B.4 (Grey Wolve Optimization)

Script GWO.m

```
%_____________________________________________________________%
%  Grey Wolf Optimizer (GWO) source codes version 1.0         %
%                                                             %
%  Developed in MATLAB R2011b(7.13)                           %
%                                                             %
%  Author and programmer: Seyedali Mirjalili                  %
%                                                             %
%         e-Mail: ali.mirjalili@gmail.com                     %
%                 seyedali.mirjalili@griffithuni.edu.au       %
%                                                             %
%       Homepage: http://www.alimirjalili.com                 %
%                                                             %
%    Main paper: S. Mirjalili, S. M. Mirjalili, A. Lewis      %
%                Grey Wolf Optimizer, Advances in Engineering %
%                Software , in press,                         %
%                DOI: 10.1016/j.advengsoft.2013.12.007        %
%                                                             %
%_____________________________________________________________%

% Grey Wolf Optimizer
function
[Alpha_score,Alpha_pos,Convergence_curve]=GWO(SearchAgents_no,Max_iter,lb
,ub,dim,fobj)

% initialize alpha, beta, and delta_pos
Alpha_pos=zeros(1,dim);
Alpha_score=inf; %change this to -inf for maximization problems

Beta_pos=zeros(1,dim);
Beta_score=inf; %change this to -inf for maximization problems

Delta_pos=zeros(1,dim);
Delta_score=inf; %change this to -inf for maximization problems

%Initialize the positions of search agents
Positions=initialization(SearchAgents_no,dim,ub,lb);

Convergence_curve=zeros(1,Max_iter);

l=0;% Loop counter

% Main loop
while l<Max_iter
    for i=1:size(Positions,1)

        % Return back the search agents that go beyond the boundaries of
the search space
            Flag4ub=Positions(i,:)>ub;
            Flag4lb=Positions(i,:)<lb;
Positions(i,:)=(Positions(i,:).*(~(Flag4ub+Flag4lb)))+ub.*Flag4ub+lb.*Fla
g4lb;

        % Calculate objective function for each search agent
            Fitness=fobj(Positions(i,:));
```

```
        % Update Alpha, Beta, and Delta
        if Fitness<Alpha_score
            Alpha_score=Fitness; % Update alpha
            Alpha_pos=Positions(i,:);
        end

        if Fitness>Alpha_score && Fitness<Beta_score
            Beta_score=Fitness; % Update beta
            Beta_pos=Positions(i,:);
        end

        offspring = Alpha_pos + (0.5*(Beta_pos-Alpha_pos))
        Delta_score=offspring;
        d=0.01*unifrnd(-0.1,0.1,size(offspring)).*(ub-lb)
        Delta_score = Delta_score + d
        Delta_pos=Positions(i,:);
    end
    a=2-l*((2)/Max_iter); % a decreases linearly fron 2 to 0
    % Update the Position of search agents including omegas
    for i=1:size(Positions,1)
        for j=1:size(Positions,2)
            r1=rand(); % r1 is a random number in [0,1]
            r2=rand(); % r2 is a random number in [0,1]
            A1=2*a*r1-a; % Equation (3.3)
            C1=2*r2; % Equation (3.4)
            D_alpha=abs(C1*Alpha_pos(j)-Positions(i,j)); % Equation
(3.5)-part 1
            X1=Alpha_pos(j)-A1*D_alpha; % Equation (3.6)-part 1

            r1=rand();
            r2=rand();

            A2=2*a*r1-a; % Equation (3.3)
            C2=2*r2; % Equation (3.4)

            D_beta=abs(C2*Beta_pos(j)-Positions(i,j)); % Equation (3.5)-
part 2
            X2=Beta_pos(j)-A2*D_beta; % Equation (3.6)-part 2

            r1=rand();
            r2=rand();

            A3=2*a*r1-a; % Equation (3.3)
            C3=2*r2; % Equation (3.4)

            D_delta=abs(C3*Delta_pos(j)-Positions(i,j)); % Equation
(3.5)-part 3
            X3=Delta_pos(j)-A3*D_delta; % Equation (3.5)-part 3
            Positions(i,j)=(X1+X2+X3)/3;% Equation (3.7)
        end
    end
    l=l+1;
    Convergence_curve(l)=Alpha_score;
end
```

Lampiran B.5 (Grey Wolve Optimization)

Script initialization.m

```
%_____________________________________________________________________%
%  Grey Wolf Optimizer (GWO) source codes version 1.0               %
%                                                                     %
%  Developed in MATLAB R2011b(7.13)                                 %
%                                                                     %
%  Author and programmer: Seyedali Mirjalili                        %
%                                                                     %
%         e-Mail: ali.mirjalili@gmail.com                           %
%                 seyedali.mirjalili@griffithuni.edu.au             %
%                                                                     %
%       Homepage: http://www.alimirjalili.com                       %
%                                                                     %
%    Main paper: S. Mirjalili, S. M. Mirjalili, A. Lewis            %
%                Grey Wolf Optimizer, Advances in Engineering       %
%                Software , in press,                               %
%                DOI: 10.1016/j.advengsoft.2013.12.007              %
%                                                                     %
%_____________________________________________________________________%

% This function initialize the first population of search agents
function Positions=initialization(SearchAgents_no,dim,ub,lb)

Boundary_no= size(ub,2); % numnber of boundaries

% If the boundaries of all variables are equal and user enter a signle
% number for both ub and lb
if Boundary_no==1
    Positions=rand(SearchAgents_no,dim).*(ub-lb)+lb;
end

% If each variable has a different lb and ub
if Boundary_no>1
    for i=1:dim
        ub_i=ub(i);
        lb_i=lb(i);
        Positions(:,i)=rand(SearchAgents_no,1).*(ub_i-lb_i)+lb_i;
    end
end
```