SKRIPSI

ANALISIS PERBANDINGAN PENENTUAN EFISIENSI MOTOR INDUKSI TIGA PHASA DENGAN MENGGUNAKAN METODE ALGORITMA GENETIKA DAN *EVOLUTIONARY PROGRAMMING* DI LABORATORIUM KONVERSI ENERGI ELEKTRIK ITN MALANG

Disusun oleh :

RILYA ARTIKA DEVI

NIM 03.12.051

JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ENERGI LISTRIK
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG

SEPTEMBER 2007

SKRIPSI

ANALISIS PERBANDINGAN PENENTUAN EFISIENSI MOTOR INDUKSI TIGA PHASA DENGAN MENGGUNAKAN METODE ALGORITMA GENETIKA DAN *EVOLUTIONARY PROGRAMMING* DI LABORATORIUM KONVERSI ENERGI ELEKTRIK ITN MALANG

Disusun oleh :

RILYA ARTIKA DEVI
NIM 03.12.051

JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ENERGI LISTRIK
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG

SEPTEMBER 2007

LEMBAR PERSETUJUAN

ANALISIS PERBANDINGAN PENENTUAN EFISIENSI MOTOR INDUKSI TIGA PHASA DENGAN MENGGUNAKAN METODE ALGORITMA GENETIKA DAN *EVOLUTIONARY PROGRAMMING* DI LABORATORIUM KONVERSI ENERGI ELEKTRIK ITN MALANG

SKRIPSI

Disusun Untuk Melengkapi dan Memenuhi Persyaratan
Guna Mencapai Gelar Sarjana Teknik Elektro Strata Satu (S-1)

Disusun Oleh :

RILYA ARTIKA DEVI
NIM: 03.12.051

Diperiksa dan disetujui,

Dosen Pembimbing I	Dosen Pembimbing II
(Ir. Widodo Pudji M, MT)	(Ir. Djojo Priatmono, MT)
NIP. Y. 1028700171	NIP. Y. 1018500107

Mengetahui,
Ketua Jurusan Teknik Elektro S-1

(Ir. F. Yudi Limpraptono, MT)
NIP. Y. 103 950 0274

KONSENTRASI TEKNIK ENERGI LISTRIK
JURUSAN TEKNIK ELEKTRO S-1
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
2007

Analisis Perbandingan Penentuan Efisiensi Motor Induksi Tiga Phasa Dengan Menggunakan Metode Algoritma Genetika Dan *Evolutionary Programming* Di Laboratorium Konversi Energi Elektrik ITN Malang

Rilya Artika Devi
Fakultas Teknik Industri Jurusan Teknik Elektro Institut Teknologi Nasional Malang
Jalan Raya Karanglo Km 2
Ir. Widodo Pudji , MT & Ir. Djojo Priatmono, MT

Abstrak

Motor induksi secara luas digunakan untuk berbagai kepentingan, khususnya untuk kepentingan industri. Diperlukan pengujian – pengujian untuk mengetahui nilai efisiensi suatu motor induksi yang telah terpasang pada beban. Sehingga timbul permintaan dari industri-industri yang sebagian besar mesinnya menggunakan aplikasi motor induksi untuk mengembangkan dan meningkatkan metode penentuan efisiensi motor induksi dengan tingkat ketelitian yang tinggi , tanpa pengujian tak berbeban.

Penulisan ini didasarkan pada studi literatur, dan mengulas data yang diperoleh dari pengujian laboratorium , meliputi daya masukan, tegangan, arus, dan kecepatan. Data-data yang dimaksud di atas digunakan untuk menentukan parameter-parameter motor induksi. Jika parameter-parameter sudah didapatkan, maka efisiensi dapat diperkirakan. Parameter-parameter itu digunakan untuk melakukan perhitungan efisiensi motor induksi menggunakan metode Algoritma Genetika dan *Evolutionary Programming* , yaitu dengan mengevaluasi parameter-parameter motor tersebut dengan pengujian berbeban. Untuk mengilustrasikan seberapa sesuai perkiraan efisiensi di laboratorium dengan hasil program menggunakan metode-metode yang disebutkan , berikut disampaikan pula hasil pengujian di laboratorium.

Hasil uji laboratorium tersebut merupakan acuan nilai yang akan dibandingkan dengan nilai-nilai efisiensi yang didapatkan dari kedua metode , dan hasi yang dapat dilihat adalah bahwa metode Algoritma Genetika memiliki akurasi yang lebih tinggi daripada metode *Evolutionary Programming*.

Kata kunci : Efisiensi, Algoritma Genetika, *Evolutionary Programming*

Abstract

Induction motors are commonly use for many purposes, especially for industrial purposes. It needs some test to find out the EFF value of an induction motor that installed to the leads. It makes some requests from industries that most of their machines uses the application of induction motors to develop and increase some methods to determine the induction motors efficiency with the high accuracy without the no-load test.

This paper is written based on the literature study, and it describes the use of a few sets of data, such as input power , voltage , current, and speed. Those data are use to determine the three phase induction motor's parameters. Once those parameters are known, it is possible to obtain the efficiency of the motor. Those parameters are use to calculate the efficiency of three phase induction motors using Genetic Algorithm and Evolutionary Programming, by evaluate the motor's parameters instead of using the –load test result. To illustrate how well the estimated efficiency obtained from the laboratory test match with the software's result using the mentioned methods , the result of the laboratory test are presented.

The laboratory test result are represented as reference values that will be compared with the values of efficiency that obtained from both methods (Genetic Algorithm and Evolutionary Programming). The result that can be seen is that Genetic Algorithm has a higher accuracy than Evolutionary Programming .

Keywords : *Efficiency, Genetic Algorithm, Evolutionary Programming*

KATA PENGANTAR

Dengan segala puji dan syukur bagi kehadirat Allah SWT atas karunia hikmah yang diberikan-Nya sehingga penulis dapat menyelesaikan skripsi ini dengan judul : "**Analisis Perbandingan Penentuan Efisiensi Motor Induksi Tiga Phasa Dengan Menggunakan Metode Algoritma Genetika Dan *Evolutionary Programming* Di Laboratorium Konversi Energi Elektrik ITN Malang**".

Penyusunan skripsi ini ditujukan untuk memenuhi tugas-tugas dan syarat-syarat akademis guna memperoleh gelar Sarjana Teknik pada Institut Teknologi Nasional Malang.

Pada kesempatan ini penulis mengucapkan rasa terima kasih yang sedalam-dalamnya kepada yang terhormat :

1. Bapak Prof. Dr. Ir. Abraham lomi, MSEE, selaku Rektor Institut Teknologi Nasional Malang.
2. Bapak Ir. Mochtar Anshori, MT selaku Dekan FTI ITN Malang.
3. Bapak Ir. Yudi Limpraptono, MT, selaku Ketua Jurusan Teknik Elektro ITN Malang.
4. Bapak Ir. Widodo Pudji M, MT dan Bapak Ir. Djojo Priatmono, MT selaku dosen pembimbing.
5. Bapak Ir. Abdul Hamid ,MT, selaku dosen keahlian.
6. Seluruh rekan seangkatan atas segala kebersamaannya.
7. Semua pihak yang membantu dalam penyelesaian skripsi ini.

Penulis menyadari sepenuhnya bahwa dalam penyusunan skripsi ini masih terdapat banyak kekurangan, maka dari itu penulis mengharapkan segala saran yang membangun. Semoga skripsi ini bisa bermanfaat bagi semua pihak.

Malang, 25 September 2007

Penulis

(Rilya Artika Devi)

LEMBAR PERSEMBAHAN

Ass.wr.wb,,

Trima kasih dan puji syukur sedalam-dalamnya aku ucapkan kapada Allah SWT, atas berkah dan hidayahNya sehingga skripsi ini akhirnya terselesaikan juga..

Skripsi ini aku persembahkan kepada orang-orang yang sangat berarti dalam hidupku,,especially my Mom and Dad,, thanx 4 all supports (prayer, financial, etc,,walopun aku sering ngecewain kalian,,thx ya ma..pa..^_^)

Buat adek semata wayangku Rendy ,yang gak pernah kehabisan crita (mulai dari dapet beasiswa pe keturutan nabrak orang,busyet ren2,,)makasih dah nemenin ngilangin hp..hehe..

Buat mbah..maturnuwun atas kopinya yang gak pernah telat nemenin aku lembur,,juga makasih dah bersedia jd "weker" bangunin aku tiap pagi sejak aku TK pe segede ini..

Buat orang2 di rumah candi ngrimbi..mamati, papati, mb. Ana+m.roni+fadan, Mb. fi2+m. dimas+ditto, [illegible] , m. Rizki+m. Nophee,,trimakasih buat segenap perhatian dan upayanya untuk menjadikan aku jadi lebih baik,,without you guys,,it ain't me..

Buat Rey... makasih ya,, km dah jadi motivasiku 'n semangatku untuk slesein kuliah tepat waktu,,makasih juga dah temenin aku slama 4 tahun tanpa absen,I love you always...

Tante diah, [redacted], [redacted] tunggu traktirannya aja yah,,Insya Allah..

To : [redacted] + affa, deuhh,,,,asline aku bosen juga liat tampang2 kalian,,coz kita dah sama2 7 taon!! tp gimana2 aku sayang kalian, tengkyu bgt buat kbersamaannya(padahal wktu ak ketik iki kita lg di t4 rateh, dan kalian ngorok kabeh,,huh..),harus kuakui kalian t4 berbagiku yg paling klop deh..Buat oi ,muklis,,whooi..prasaan kumpul2nya kita kok cm pas acara makan doank yo??

Thx juga buat mas ugro wiseno atas validnya programku, maap ngrepotin yaa..Buat mas rudi yang dah nemenin pengujian 'n nyiapin data..[redacted] yg dah ngasi tau alamtnya mas ugro,hehe,, suwon yoo..,mas Ram,yang telah menyadarkanku bahwa pendekatan rumusku di bab 4 salah =)thx bgt..

Buat temen2ku seangkatan yang slalu siap membantu,, [redacted] bangil, [redacted], [redacted], heru, restu, iwan, ditto, farukh, dan masi

byk lagi,,klo ga da kalian aku pasti slalu ketinggalan dpt info,,makasih ya tmen2,,sukses buat kalian..

Buat [redacted],,maap aku banyak tanya,,makasi banyak buat sgala infonya..

Buat manusia2 di rental trowulan, berkat kalian aku gak perlu nglembur tiap mlm..maksih2..

Tak lupa,,buat kucing2ku yang setia,,lolo, [redacted], luna, kiyong, miu beserta 3 anaknya yang belom punya nama, lui, milo, mumu, babi, [redacted], ,keberadaan kalian bener2 bisa menghiburku,,klo aku dah dpt kerja, Insya Allah kalian maem whiskas+friskies tiap hari tanpa diselingi pindang kayak sekarang,,haha..

Fiuuh...lega rasanya udah kelar sgala bebanku di bangku kuliah,,sekian dulu ucapan trimakasihnya yaa,,moga gak ada yg klewatan gak kusebutin,,klo ada, aku sungguh2 minta maaf,,sama sekali gak sengaja..aku cm berharap,semoga apa yang kutulis ada manfaatnya buat kita semua,,amiieeenn,,,

Wass.wr.wb

Rilya Artika Devi

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Motor induksi secara luas digunakan untuk berbagai kepentingan khususnya industri. Sekitar delapan puluh persen untuk motor listrik dan sekitar tiga puluh persen total konsumsi listrik dari motor induksi. Diperlukan pengujian-pengujian untuk mengetahui nilai efisiensi suatu motor induksi yang telah terpasang pada beban. Sehingga timbul permintaan dari industri-industri yang sebagian besar mesinnya menggunakan aplikasi motor induksi untuk mengembangkan dan meningkatkan metode penentuan efisiensi motor induksi dengan tingkat ketelitian yang mendekati nilai sebenarnya tanpa pengujian tak berbeban (*no load test*).

Metode penentuan efisiensi sangat bervariasi dipandang dari segi kerumitan, performa keseluruhan dan kecocokan pada kondisi pabrik. Beberapa diantaranya membutuhkan pengujian tanpa beban, yang sangat sulit diterapkan pada industri-industri besar. Pengujian tanpa beban memerlukan pemutusan motor dengan beban yang tidak dapat dengan mudah dilakukan pada kondisi di dalam pabrik (*plant*). Di lain pihak, evaluasi efisiensi motor berdasarkan pada papan nama (*nameplate*) atau buku arahan (*manual book*) tidak dapat menjamin penilaian yang tepat dari motor induksi yang dipakai di pabrik [1].

1.2 Rumusan Masalah

1. Bagaimanakah aplikasi metode Algoritma Genetika dan *Evolutionary Programming* diterapkan pada penentuan nilai efisiensi motor induksi 3 phasa sebagai suatu pemecahan masalah dan berapakah nilai-nilai yang dihasilkan?
2. Bagaimanakan perbandigan kedua metode tersebut, manakah yang paling baik dalam penerapannya?

1.3 Tujuan

Menganalisis efisiensi motor induksi 3 phasa dengan menggunakan metode Algoritma Genetika dan *Evolutionary Programming* berdasarkan data yang diperoleh dari motor berbeban, kemudian membandingkan kedua metode tersebut manakah yang memiliki ketelitian yang lebih tinggi (error kecil).

1.4 Batasan Masalah

Permasalahan dalam system tenaga listrik sangatlah luas, sehingga dalam menganalisis suatu permasalahan perlu diadakan pembahasan-pembahasan yang sesuai dengan permasalahan tersebut. Di dalam Skripsi ini pembatasan yang dilakukan adalah :

- Analisa hanya dilakukan pada motor induksi 3 phasa rotor sangkar DE LORENZO/DL, 1021, 1,1 KW, 220-380 (Δ/Y)volt, 4.3/2.5 (Δ/Y) Ampere, cos φ 0,83, 50 Hz, 2830 rpm, 2 kutub.
- Pembahasan ditekankan pada analisis penentuan efisiensi motor induksi 3 phasa.
- Metode yang dipakai untuk mencari efisiensi motor induksi adalah Algoritma Genetika dan *Evolutionary Programming*, serta rangkaian ekivalen.

1.5 Metodologi

Metodologi yang digunakan dalam penulisan skripsi ini adalah sebagai berikut:

1. Studi literatur, yaitu dengan analisa yang berdasarkan pada teori-teori yang terkait melalui literatur yang sesuai.
2. Pengumpulan data

   Pengumpulan data melalui percobaan pada motor induksi dengan metode pengujian berbeban.
3. Melakukan perhitungan untuk menentukan daya keluaran sehingga nantinya akan didapat nilai efisiensi untuk hasil sebenarnya dari pengujian.
4. Melakukan analisis penentuan efisiensi motor induksi menggunakan metode Algoritma Genetika dan *Evolutionarry Programming* dengan bantuan program Delphi 7.
5. Menarik kesimpulan.

1.6 Sistematika Penulisan

Secara garis besar sistematika penulisan ini meliputi:

BAB I : PENDAHULUAN

Membahas mengenai latar belakang permasalahan, tujuan, metodologi, sistematika dan relevansi dari penulisan ini.

BAB II : KONSEP DASAR MOTOR INDUKSI

Berisi tentang teori mengenai motor induksi, konstruksi, prinsip kerja motor induksi, rangkaian ekivalen dan pengujian.

BAB III : TEORI DASAR ALGORITMA GENETIKA DAN *EVOLUTIONARRY PROGRAMMING* SERTA APLIKASINYA TERHADAP PENENTUAN EFISIENSI MOTOR INDUKSI TIGA PHASA

Berisi tentang teori dasar Algoritma Genetika, *Evolutionarry Programming* dan pengaplikasikannya pada penentuan efisiensi motor induksi tiga fasa.

BAB IV : SIMULASI DAN ANALISIS HASIL

Membahas tentang analisa data pengujian yang digunakan dalam penentuan efisiensi motor induksi tiga fasa, flowchart program, tampilan program, perbandingan hasil pengujian dengan metode Algoritma Genetika dan dengan metode *Evolutionarry Programming*.

BAB V : PENUTUP

Berisi kesimpulan sesuai analisa yang dilakukan pada BAB IV.

1.5 Relevansi

Dengan adanya analisa ini diharapkan dapat memberikan salah satu alternatif metode dalam penentuan efisiensi motor induksi 3 phasa dengan pengujian berbaban yang tidak memerlukan pemutusan beban pada motor induksi yang dipakai pada kondisi terinstalasi di lapangan.

BAB II
KONSEP DASAR MOTOR INDUKSI

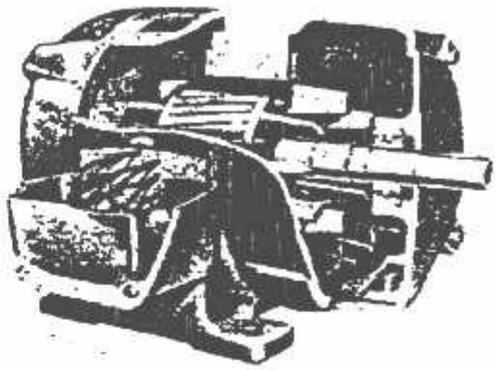
2.1 Motor Induksi

Secara umum, motor listrik berfungsi untuk mengubah energi listrik menjadi energi yang berupa tenaga putar. Di dalam motor DC, energi listrik diambil langsung dari kumparan armature dengan melalui sikat dan komutator, oleh karena itu motor DC disebut motor kondusi. Lain halnya pada motor AC, pada motor AC, kumparan rotor tidak menerima energi listrik langsung, tetapi secara induksi seperti terjadi pada energi kumparan sekunder transformator, sehingga dikenal dengan motor induksi. Sebenarnya motor induksi dapat diidentikkan dengan transformator yang kumparan primernya sebagai kumparan stator atau armature, sedangkan kumparan sekunder sebagai kumparan rotor.

Motor induksi *polyphase* banyak dipakai di kalangan industri. Ini berkaitan dengan beberapa keuntungan dan kerugian.

1. Keuntungan
   a. Sangat sederhana dan daya tahan kuat (konstruksi hampir tak pernah terjadi kerusakan, khususnya tipe *squirrel cage*).
   b. Harga relatif murah dan perawatan mudah.
   c. Efisiensi tinggi. Pada kondisi berputar normal, tidak dibutuhkan sikat dan karenanya rugi daya yang ditimbulkan dapat dikurangi.
2. Kerugian
   a. Kecepatan tidak dapat berubah tanpa pengorbanan efisiensi.
   b. Tidak seperti motor DC atau motor shunt, kecepatannya menurun seiring dengan tambahan beban.

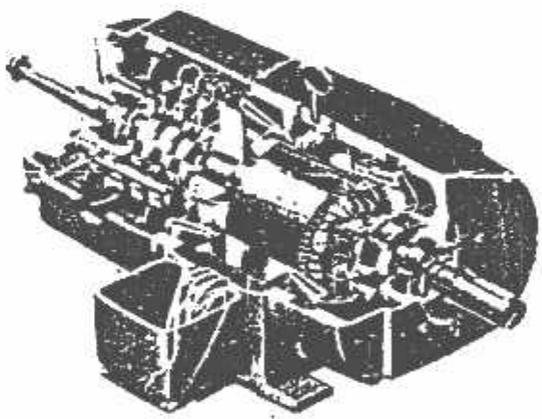
c. Kopel awal mutunya rendah dibanding dengan motor DC shunt.



Gambar 2-1

Penampang Potongan Motor Induksi Sangkar Tupai

Sumber: A.E. Fitzgerald Charles Kingsley, Ir.Stephen D. Umar, Ir. Djoko Achyanto M.Sc.EE, "**Mesin-Mesin Listrik**", Edisi ke empat, Erlangga,1992.



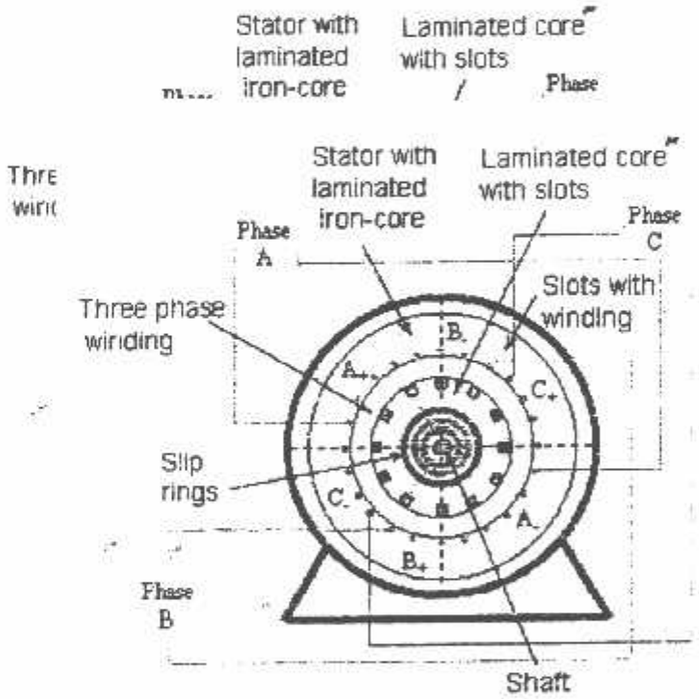
Gambar 2-2

Penampang Potongan Motor Induksi Belitang

Sumber: A.E. Fitzgerald Charles Kingsley, Ir.Stephen D. Umar, Ir. Djoko Achyanto M.Sc.EE, "**Mesin-Mesin Listrik**", Edisi ke empat, Erlangga,1992

2.2 Konstruksi[4]

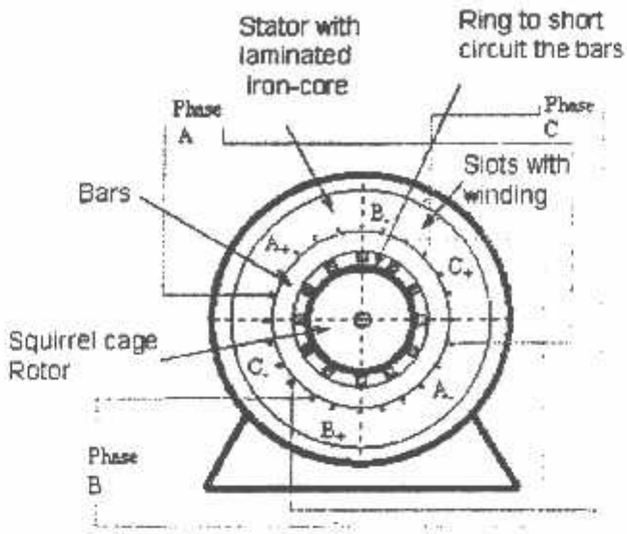
Pada prinsipnya motor induksi dibagi menjadi 2 bagian, yaitu bagian stator dan rotor. Pada bagian stator terdapat beberapa slot yang merupakan tempat kawat (konduktor) dari tiga kumparan tiga phase yang disebut kumparan stator, yang masing-masing kumparan mendapatkan suplai arus tiga phase. Jika kumparan stator mendapatkan suplai arus tiga phase, maka pada kumparan tersebut segera timbul fluks magnet putar. Karena adanya fluks magnet putar pada kumparan stator, mengakibatkan rotor berputar karena adanya induksi magnet dengan kecepatan putar rotor sinkron, ns = 120f/p.



Gambar 2-3

Motor Induksi Jenis *Wound-Rotor*

Sumber : George G. Karady. " **induction Machine**", Topic 5.Lecture note EEE 360

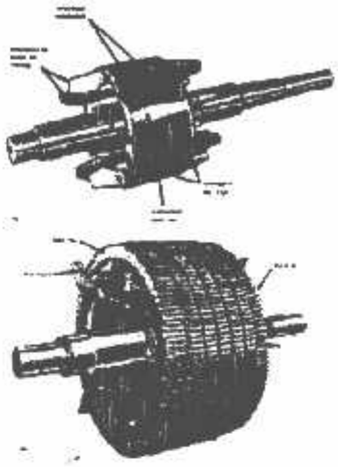


Gambar 2-4

Motor Induksi Jenis *Squirel - Cage*

Sumber : George G. Karady. " **Induction Machine**", Topic 5.Lecture note EEE 360

Bagian rotor yang merupakan tempat kumparan adalah bagian yang bergerak atau berputar. Ada dua jenis kumparan rotor, yaitu *squirrel cage rotor* dan *phase wound rotor*. Hampir 90% kumparan rotor dari motor induksi menggunakan jenis *squirrel cage rotor* (rotor sangkar tupai). Karena bentuk kumparannya sederhana dan tahan terhadap goncangan. Ciri khusus dari *squirrel cage rotor* atau rotor sangkar tupai ialah ujung-ujung kumparan rotor terhubung singkat secara permanen. Lain halnya dengan jenis *phase wound rotor* yang ujung-ujung kumparan rotor akan terhubung langsung bila kecepatan putar normalnya secara otomatis melalui slip ring yang terpasang pada bagian rotor.



Gambar 2-5

Konstruksi Rotor Sangkar Tupai

Sumber : George G. Karady. " **induction Machine**", Topic 5.Lecture note EEE 360

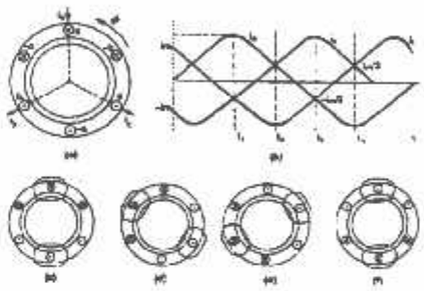
Pada gambar 2-5 diperlihatkan rotor dari motor induksi daya kecil dan besar. Pada rotor berdaya kecil dipasang batang aluminium untuk mengurangi kebisingan dan meningkatkan performa[13]. Sirip (*fins*) ditempatkan diatas cincin (*ring*) yang berfungsi sebagai kipas dan meningkatkan pendinginan (*cooling*).

2.3 Medan Putar [3]

Perputaran pada mesin arus bolak-balik ditimbulkan oleh adanya medan putar (fluks yang berputar) yang dihasilkan dalam kumparan statornya. Medan putar ini terjadi apabila kumparan stator dihubungkan dalam fasa banyak, umumnya fasa tiga. Hubungan dapat berupa hubungan bintang atau delta. Medan putar terjadi apabila kumparan a-a'; b-b'; c-c'; dihubungkan tiga fasa dengan beda fasa masing-masing 120° gambar (2-6 a) dan dialiri arus sinusoid. Distribusi arus i_a, i_b, i_c, sebagai fungsi waktu adalah seperti gambar (2-6 b) pada keadaan t_1,t_2,t_3

dan t_4, fluks resultan yang ditimbulkan oleh kumparan tersebut masing-masing adalah seperti gambar (2-6 c,d,e, dan f).

Pada t_1 fluks resultan mempunyai arah sama dengan arah fluks yang dihasilkan oleh kumparan a-a, sedangkan pada t_2, fluks resultan mempunyai arah sama dengan arah fluks yang dihasilkan oleh kumparan c-c, dan untuk t_3 fluks yang dihasilkan oleh kumparan b-b, untuk t_4, fluks resultannya berlawanan arah dengan fluks resultan yang dihasilkan pada saat t_1.



Gambar 2-6

Medan Putar Pada Motor Induksi

Sumber : Zuhal, " Dasar Teknik Tenaga Listrik ", Penerbit ITB Bandung

2.4 Prinsip Kerja Motor Induksi [9]

Motor induksi tiga phasa dapat dibandingkan dengan transformator karena merupakan piranti yang melibatkan perubahan kebocoran fluks pada kumparan stator. Dalam hubungan ini diasumsikan bahwa rotor terdiri atas tipe lilitan dan hubungan bintang. Dengan lilitan rotor dalam keadaan rangkaian terbuka tidak ada torsi yang dibangkitkan. Dengan demikian pemberian tegangan tiga phasa pada kumparan stator tiga phasa menimbulkan medan magnetik yang berputar dan memotong stator dan rotor pada frekuensi f_1. Nilai rata-rata ggl induksi per fasa dari lilitan rotor dinyatakan dengan:

$E_2 = 4{,}44 f_1 N_2 K_{w2\phi}$ .....(2.1)

Perlu diketahui bahwa frekuensi stator f_1 digunakan disini karena rotor tersebut dalam keadaan diam atau berhenti. Dengan demikian E_2 merupakan ggi (gaya gerak listrik) frekuensi saluran. Fluks (ϕ) tentu merupakan tiap elektroda (pole) dan kumparan stator dan rotor.

Rumus yang serupa menyatakan nilai rata-rata ggl induksi tiap phasa yang terjadi dari kumparan stator, yaitu:

$E_1 = 4{,}44 f_1 N_1 K_{w1\phi}$ .....(2.2)

Berdasarkan persamaan (2.1) dan (2.2) dapat dirumuskan rasio:

$$\frac{E_1}{E_2} = \frac{N_1 K_{w1}}{N_2 K_{w2}} \quad \text{.....(2.3)}$$

Pada dasarnya, motor induksi pada keadaan diam menyerupai karakteristik transformator dengan kumparan stator sebagai sisi primer dan kumparan rotor sebagai sekundernya.

Untuk menghasilkan torsi mula (dan torsi penggerak berturut-turut perlu arus yang mengalir dalam kumparan rotor. Mula-mula ggl (gaya gerak listrik) induksi E_2 mengakibatkan arus rotor perfasa I_2 mengalir melalui rangkaian hubung singkat, menghasilkan distribusi *ampere-conductor* yang bekerja dengan medan fluks untuk menghasilkan torsi mula. Pengaruh torsi ini selalu mengakibatkan rotor berputar dalam arah yang sama sebagai medan putar. Anggaplah bahwa medan fluks berputar searah jarum jam pada kecepatan tertentu yang bergantung pada frekuensi stator dan banyaknya kutub dan kumparan stalor. Kecepatan ini disebut "kecepatan sinkron" dan dinyatakan:

$$n_s = \frac{120 f_1}{p} \quad \text{.....(2.4)}$$

Karena rotor meningkatkan kecepatannya, laju yang mengijinkan medan stator memotong kumparan rotor menurun. Hal ini mengurangi ggl induksi resultan perfasa, pada gilirannya menurunkan magnitude distribusi ampere konduktor dan menghasilkan torsi yang lebih kecil. Pada kenyataannya proses ini berlanjut hingga kecepatan rotor mampu untuk menghasilkan ggl yang cukup agar menghasilkan arus yang diperlukan untuk membangkitkan torsi yang setara dengan torsi lawan.

2.4.1 Slip dan Frekuensi Arus Rotor [8]

Slip diidentifikasikan sebagai bagian dari kecepatan sinkron n_s dan kecepatan aktual rotor n. Slip dirumuskan sebagai:

$$s = \frac{n_s - n}{n_s} \qquad \text{.....(2.5)}$$

Pada keadaan diam medan magnet putar yang dihasilkan oleh stator mempunyai kecepatan relatif yang sama dengan kumparan rotor dan stator. Pada saat ini frekuensi dari arus rotor f_r sama dengan frekuensi stator f_s. Frekuensi rotor f_r adalah nol ketika motor berputar pada kecepatan sinkron. Pada saat tersebut tidak terdapat gerakan (putaran) relatif antara medan putar dan rotor. Pada kecepatan yang lain, frekuensi rotor proposional dengan slip (s). Sebuah mesin putar p kutub pada kecepatan putaran n_s rpm mempunyai frekuensi arus:

$$f_r = \frac{Pn_r}{120} = \frac{P(n_s - n)}{120} \qquad \text{.....(2.6)}$$

Kecepatan relatif motor induksi n_r antara konduktor rotor dan medan putar yang dihasilkan oleh stator diberikan oleh:

$$n_r = n_s - n \qquad \text{.....(2.7)}$$

Dari persamaan (2.5) dan (2.6) kita peroleh:

$$n_s - n = sn_s = \frac{s(120)f_s}{P} \quad \text{.....(2.8)}$$

Dengan mensubstitusikan persamaan (2.8) ke dalam persamaan (2.6), maka frekuensi slip menjadi:

$$f_r = \left(\frac{P}{120}\right)\left[\frac{s(120f_s)}{P}\right] = sf_s \quad \text{.....(2.9)}$$

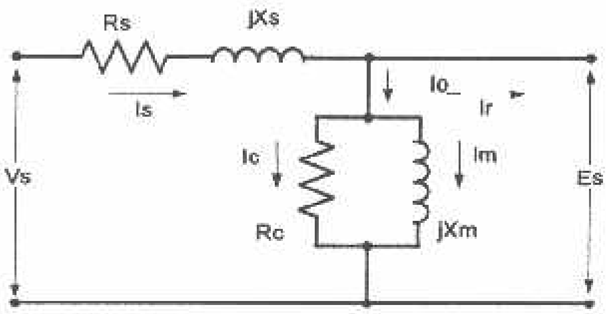
2.5 Rangkaian Ekivalen Motor Induksi

2.5.1 Rangkaian Ekivalen

Suatu rangkaian ekivalen motor induksi tiga phasa diperlukan untuk membantu analisis operasi dan untuk memudahkan penghitungan kinerja. Rangkaian ekivalen tersebut mengasumsi suatu bentuk yang identik dengan rangkaian ekivalen transformator. Proses penurunannya serupa dengan model dengan modifikasi-modifikasi baru seperlunya untuk menghitung kumparan sekunder (rotor) dalam hal ini berputar dan membangkitkan daya mekanik.

2.5.2 Rangkaian Ekivalen Stator

Jika belitan stator mendapatkan tegangan catu dari jala-jala sebesar Vs, maka akan mengalir arus putar tiga phasa pada belitan stator yang akan membangkitkan medan magnet tiga phasa. Arus stator (I_s) dapat dibagi menjadi dua komponen yaitu komponen beban (I_r) dan komponen penguat / eksitasi (I_0).



Gambar 2-7

Rangkaian Ekivalen Stator Motor Induksi

Sumber: A.E. Fitzgerald Charles Kingsley, Ir.Stephen D. Umar, Ir. Djoko Achyanto M.Sc.EE, "**Mesin-Mesin Listrik**", Edisi ke empat, Erlangga,1992.

Dimana:

V_s = tegangan terminal per-fasa

R_s = resistansi kumparan stator per-fasa

X_s = reaktansi bocor kumparan stator per-fasa

E_s = tegangan induksi (ggl) per-fasa didalam kumparan stator

R_c = resistansi rugi-rugi inti stator per-fasa

X_m = reaktansi magnetisari stator per-fasa

2.5.3 Rangkaian Ekivalen Rotor

Pada saat rotor masih diam, medan putar stator akan memotong batang konduktor dengan kecepatan putar sinkron (n_s), sehingga frekuensi arus rotor sama dengan frekuensi arus stator ($f_s = f_r$) dan slip sama dengan satu (s=1) dengan mengetahui bahwa frekuensi arus atau tegangan rotor adalah frekuensi slip, maka reaktansi bocor per fasa adalah:

$X_r' = sXr$ ....(2.10)

Tegangan yang diinduksikan pada rotor:

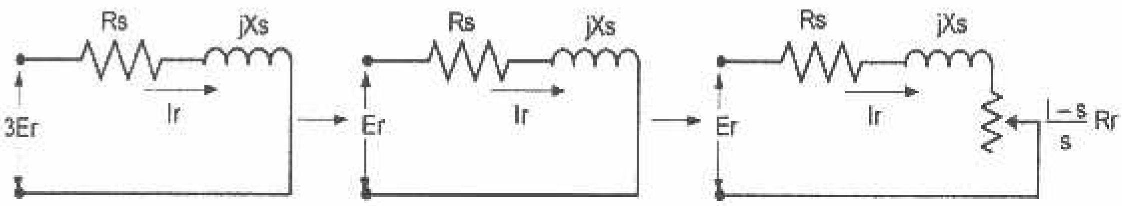
$Er = 4{,}44 \,.\, fr \,.\, Nr \,.\, \phi_m$ ....(2.11)

Pada slip, s, frekuensi rotor menjadi s.fs, maka tegangan yang diinduksikan pada rotor (Er') pada slip, s, adalah:

$$Er' = 4{,}44 \;.\; s.\; fr \;.\; Nr \;.\; \phi_m \qquad (2.12)$$

Dengan memasukkan (2.11) ke (2.12) maka didapat:

$$Er = s \;.\; Er \qquad (2.13)$$



Gambar 2-9

Rangkaian Ekivalen Rotor Motor Induksi

Sumber : S.A. Nasar and L.E. Unnewehr, "**Electromechanics and Electric Machine**", United States of America,1979

Dimana:

S = slip

E_s = tegangan induksi per-fasa di dalam rotor pada kondisi diam

X_r = reaktansi bocor rotor per-fasa

Berdasarkan (2.10) dan (2.13) maka diperoleh rangkaian rotor seperti pada gambar Arus rotor (I_r), yaitu:

$$I_r = \frac{sE_r}{\sqrt{R_r + (X_r)^2}} \qquad (2.14)$$

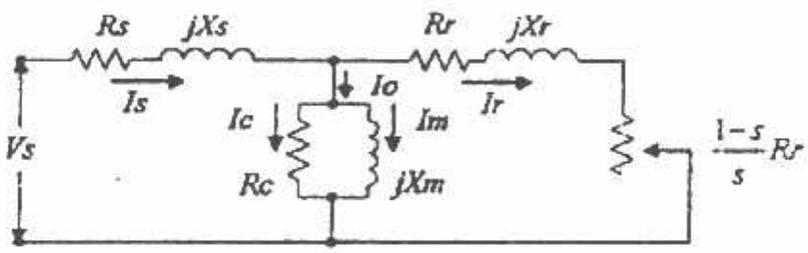
Atau

$$I_r = \frac{E_r}{\sqrt{\left(\frac{R_r}{S}\right)^2 + (X_r)^2}} \qquad (2.15)$$

2.5.4 Rangkaian Ekivalen Motor Induksi

Kerja motor induksi seperti juga kerja pada transformator adalah berdasarkan prinsip induksi elektromagnetik. Oleh karena itu motor induksi dapat dipandang sebagai transformator yang mempunyai ciri-ciri khusus yaitu:

1. Stator sebagai sisi primer
2. Rotor sebagai sekunder yang penghantar-penghantarnya dihubungkan singkat dan berputar.
3. Kopling antara sisi primer dan sisi sekunder dipisahkan oleh celah udara.



Gambar 2-9

Penampang Ekivalen Motor Induksi

Sumber: A.E. Fitzgerald Charles Kingsley, Ir.Stephen D. Umar, Ir. Djoko Achyanto M.Sc.EE, **"Mesin-Mesin Listrik"**, Edisi ke empat, Erlangga,1992.

Rangkaian ekivalen tersebut memperlihatkan bahwa daya keseluruhan yang dialihkan pada celah udara dari stator (masukan daya ke rotor) adalah:

$$P_r = 3I_r^2 \frac{R_r}{s} \quad \text{.....(2.16)}$$

Dan rugi tembaga rotor:

$$P_{cu} = 3\, I_r R_r \quad \text{.....(2.17)}$$

Maka daya mekanis yang dibangkitkan oleh motor induksi adalah:

$$P_m = P_r - P_{cu} = 3I_r^2 \frac{R_r}{s} - 3I_r^2 R_r = 3I_r^2 R_r \frac{(1-s)}{s} \quad \text{.....(2.18)}$$

$$P_m = T\omega_r = T\omega_s (1 - s) \quad \text{.....(2.19)}$$

Dengan :

T = Torsi motor dalam N-m

$T\omega_r$ = Kecepatan rotor dalam rad/detik

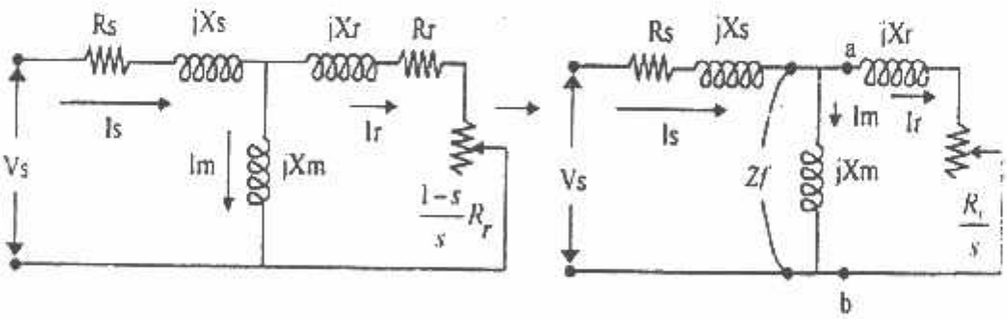
$T\omega_s$ = Kecepatan stator dalam rad/detik

Sehingga diperoleh :

$$T = (3 / \omega_s)\, I_r^2 \frac{R_r}{s} \quad \text{.....(2.20)}$$

$$I_r = \frac{V_s}{\sqrt{\left(R_s + \frac{R_r}{s}\right)^2 + (X_s + X_r)^2}} \quad \text{.....(2.21)}$$

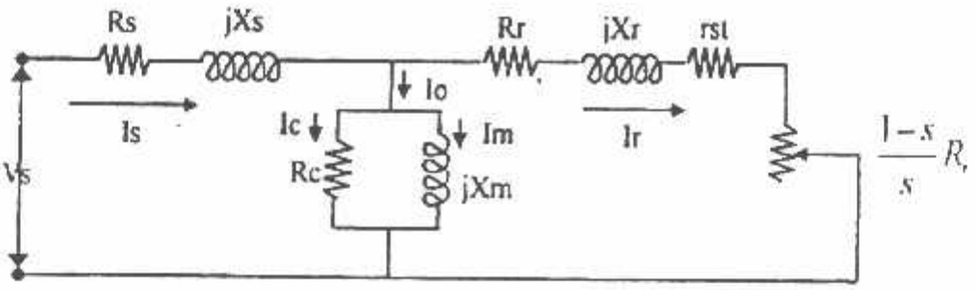
Dalam analisis rangkaian ekivalen, sering disederhanakan dengan menghilangkan resistensi (Rc), sehingga rangkaian ekivalen pada gambar 2-9 berubah menjadi:



Gambar 2-10

Penyederhanaan Rangkaian Ekivalen Motor Induksi

Sumber : Djoko Achyanto Msc, EE, **Mesin-mesin Listrik,** Edisi Keempat, Erlangga , Jakarta



Gambar 2-11

Rangkaian Ekivalen Motor Induksi

Sumber: P.Pillay,V. Levin, P. Otaduy, and J. Kueck "**In-situ Induction Motor Efficiency Determination using The Genetic Algorithm**" IEEE Transaction on Energy Conversion, Vol, 13. pp. 326-333. Dec 1998

Untuk rangkaian ekivalen pada gambar 2-11, mencari tahanan stator (R_1) pada hubungan Y (bintang) motor induksi, digunakan rumus:

$$r_1 = \frac{r_{1lin}}{2} \quad \text{.....(2.22)}$$

Dan rpm pada beban yang berbeda diukur. *Power factor* masukan dapat dicari dengan persamaan:

$$pf = \frac{P_{in}}{\sqrt{3}V_1 I_1} \quad \text{.....(2.23)}$$

Dimana:

r_1 = resistensi phasa stator

r_{1lin} = resistensi phasa stator

V_1 = tegangan line stator

I_1 = arus stator

P_{in} = daya masukan

Data yang digunakan untuk menentukan parameter dan rangkaian ekivalen dan efisiensi motor diukur dan dihitung pada 25%, 50%, 75%, dan 100% dari beban penuh. Menurut prosedur IEEE untuk penentuan efisiensi dengan rugi beban tambahan yang diasumsikan, kerugian ini dibawah beban penuh dapat diperkirakan sekitar 1,8% dari daya keluaran (*output power*) dan kerugian pada beban manapun dapat dihitung dengan persamaan:

$$P_{st} = P_{st\,ft} \frac{I_2^2}{I_{2ft}^2} \quad \text{.....(2.24)}$$

$$r_s = 0{,}018 r_2 (1 - s_{ft}) / s_{ft} \quad \text{.....(2.25)}$$

Dimana:

P_{st} = Rugi beban tambahan (*stray load losses*)

P_{stft} = Rugi beban tambahan pada beban penuh

I_2 = Arus rotor

I_{2ft} = Arus rotor pada beban penuh

R_{st} = Tahanan rugi beban tambahan

S_{st} = Slip pada beban penuh

Temperatur kumparan stator dan rotor diasumsikan sama dan dihitung dengan persamaan berikut:

$$T_t = \frac{I_1 - I_0}{I_{ft} - I_0}(T_r - T_s) + T_s \quad \text{.....(2.26)}$$

Dimana:

I_1 = Arus stator yang diukur

I_{ft} = Arus stator pada nameplate

I_0 = Arus stator pada percobaan DC test

T_r = Temperatur pada isolasi (75°C)

R_{st} = Tahanan rugi beban tambahan

S_{st} = Slip pada beban penuh

S_{st} = Slip pada beban penuh

T_s = Temperatur lingkungan (=25°C)

Nilai resistensi stator dan rotor dikoreksi (*corrected*) dengan data temperatur melalui persamaan:

$$r_{1c} = r_1 \frac{T_t + k_c}{T_s + k_c} \quad \text{.....(2.27)}$$

$$r_{2c} = r_2 \frac{T_t + k_a}{T_s + k_c} \quad \text{.....(2.28)}$$

Dengan:

r_{1c} = nilai resistensi stator DC test yang dikoreksi

r_{2c} = nilai resistensi rotor yang dikoreksi

k_c = faktor koreksi pada tembaga (=234,5)

k_a = faktor koreksi pada aluminium (=225)

Admitansi pada rangkaian ekivalen, dihitung dengan persamaan:

$$\overline{Y}_1 = \frac{1}{r_{1c} + jx_1} \quad \text{.....(2.29)}$$

$$\overline{Y}_2 = \frac{1}{r_{2c} / s + r_{st} + jx_1} \quad \text{.....(2.30)}$$

$$\overline{Y}_m =$$

$$\frac{1}{x_m} + \frac{1}{r_m} \quad \text{.....(2.31)}$$

Arus stator, dihitung dengan persamaan:

$$I_{test} = |(I_1)| = \left| \frac{\overline{V_1}\,\overline{Y_1}\,(\overline{Y_2} + \overline{Y_m})}{\overline{Y_1} + \overline{Y_2} + \overline{Y_m}} \right| \quad \text{.....(2.32)}$$

Dimana, $\overline{Y_1} = V_1 \ / \sqrt{3} + j()$ .....(2.33)

Faktor daya (*power factor*) didapat dengan persamaan :

$$pf_{est} = \frac{R(\bar{I}_1)}{I_{test}} \quad \text{.....(2.34)}$$

Arus rotor didapat dari persamaan:

$$I_2 = \left| \frac{\overline{V_1}\,\overline{Y_1}\,\overline{Y_2}}{\overline{Y_1} + \overline{Y_2} + \overline{Y_m}} \right| \quad \text{.....(2.35)}$$

Arus yang melalui resistor r_m pada rangkaian ekivalen adalah:

$$I_m = \left| \frac{\overline{V_1}\,\overline{Y_1}\,\overline{Y_2}}{\overline{Y_1} + \overline{Y_2} + \overline{Y_m}} \right| \quad \text{.....(2.36)}$$

Daya masukan (*input power*) pada rangkaian ekivalen adalah:

$$Pi_{n\,est} = 3\,((I_1^2\,r_{2c} + I_2^2\,(r_{2c}/s + r_{st}) + I_1^2\,r_m) \quad \text{.....(2.37)}$$

Daya keluaran (*output power*) pada rangkaian ekivalen adalah:

$$P_{out\,est} = 3\,I_1^2\,r_{2c}\,\frac{1-s}{s} \quad \text{.....(2.38)}$$

atau, $P_{out\,est} = 3\,T.\frac{4\mu\int}{p}(1-s)$ .....(2.39)

Efisiensi didapatkan dari persamaan:

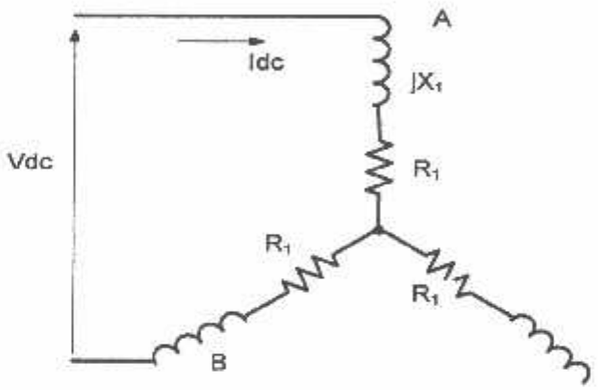
$$Eff = \frac{P_{out\,est}}{P_{in\,est}} \cdot 100\,\% \quad \text{.....(2.40)}$$

2.6 Pengujian Motor Induksi Tiga Phasa

Pengujian motor induksi tiga phasa dimaksudkan untuk mengetahui parameter-parameter dari motor yang akan dianalisis. Pengujian-pengujian tersebut yaitu:

2.6.1 Pengujian Arus Searah (DC)

Selama pengujian arus searah tegangan searah diberikan pada dua terminal stator (A dan B pada gambar). Tegangan searah dan arus searah diukur, sehingga dari pengujian ini didapat resistansi stator (Rs).

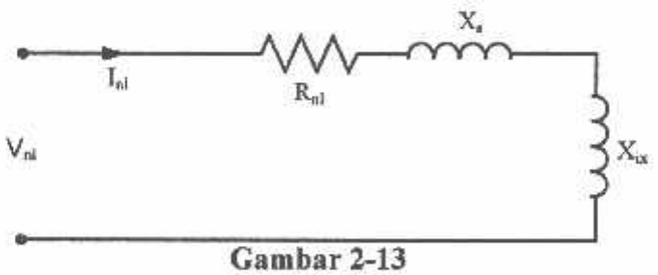


Gambar 2-12
Pengujian Arus Searah
Sumber : George G. Karady, "**Induction Machine**" topic 5 Lecture Notes EEE 360

Dari gambar 2-12 diperoleh persamaan:

$R_s = R_{dc} =$ ..... *(2.41)*

2.6.2 Pengujian Tanpa Beban (*No Load Test*)



Gambar 2-13

Rangkaian Ekivalen Pengujian Tanpa Beban
Sumber : PC. Sen, "**Principles of Electrical Machines and Power Electronic**", Second Edition, By John Wiley & Sins. Inc, 1997

Pengujian tanpa beban adalah sama dengan pengujian rangkaian terbuka. Pada keadaan tanpa beban, $R_{r/s}$ adalah sangat tinggi. Sehingga arus rotor sangat kecil dan hanya diperlukan untuk menghasilkan torsi yang cukup untuk mengatasi gesekan dan pelilitan, dengan demikian rugi-rugi I^2R rotor tanpa beban sangat kecil dan dapat diabaikan [4]. Sebagaimana yang dilukiskan dalam gambar 2-13 diperoleh persamaan-persamaan:

Rugi-rugi tahanan stator:

$$P_I^2{}_R t = 3I_{ml}^2 . R_s \quad \text{.....(2.42)}$$

Besarnya rugi-rugi putaran:

$$P_{rot} = P_{nl} - P_I^2{}_{Rs} \quad \text{.....(2.43)}$$

Dimana:

I_{ml} = arus tanpa beban

P_{nl} = masukan daya ke stator pada keadaan tanpa beban

P_{rot} = rugi-rugi putaran tanpa beban

Impedansi tanpa beban adalah:

$$Z_{nl} = \frac{V_{nl}}{I_{nl}} \quad \text{.....(2.44)}$$

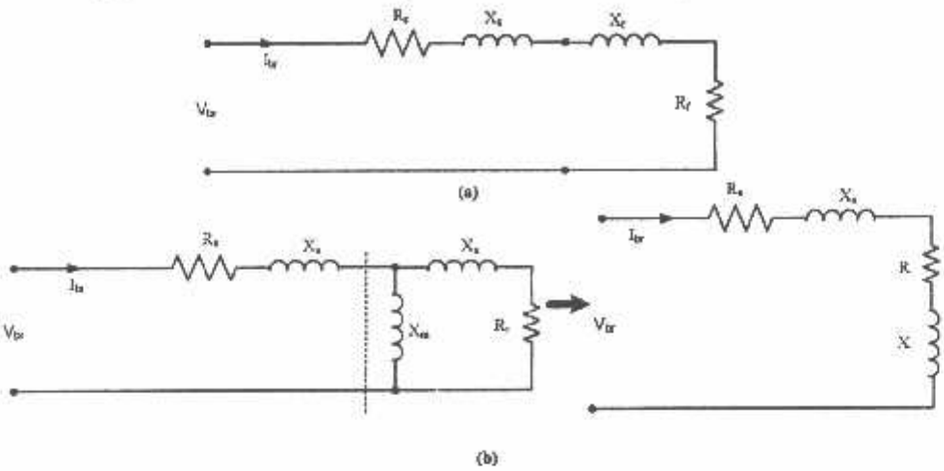
Resistansi tanpa beban adalah:

$$R_{nl} = \frac{P_{nl}}{3.I_{nl}^2} \quad \text{.....(2.45)}$$

Reaktansi tanpa beban adalah:

$$X_{nl} = \sqrt{Z_{nl}^2 - R_{nl}^2} = X_s + X_m \quad \text{.....(2.46)}$$

2.6.3 Pengujian Rotor Tertahan (*Blocked – Rotor Test*)



Gambar 2-14

Rangkaian Ekivalen Pengujian Rotor Tertahan

Sumber : P.C. Sen. " **Principles of Electrical Machine And Power Electronic**". Second Edition. By John Wiley & Sins . Inc. 1997

Pada pengujian rotor tertahan, rotor motor ditahan (s=1). Pengujian ini sama dengan pengujian rangkaian hubung singkat. Rekomendasi dari IEEE bahwa untuk pengujian rotor tertahan pada motor induksi lebih dari 20 hp, suplai frekuensi yang digunakan adalah 25% dari frekuensi normalnya sedangkan untuk motor induksi kurang dari 20 hp maka pengaruh frekuensi dapat diabaikan, dan motor dapat langsung dihubungkan dengan frekuensi normalnya. Karena dalam pembahasan skripsi ini digunakan motor induksi kurang dari 20 hp, maka pengaruh frekuensi dapat diabaikan [4]. Dari pengujian rotor tertahan diperoleh:

Resistansi rotor tertahan adalah:

$$R_{br} = \frac{P_{br}}{3I_{br}} \quad \text{..... (2.47)}$$

Impedansi rotor tertahan adalah:

$$Z_{br} = \frac{V_{nl}}{I_{nl}} \quad \text{..... (2.48)}$$

Reaktansi rotor tertahan adalah:

$$X_{br} = \sqrt{Z_{br}^{\ 2} - R_{br}^{\ 2}} \quad \text{..... (2.49)}$$

Dimana :

I_{br} = arus pada keadaan rotor tertahan

P_{br} = masukan daya ke stator pada keadaan rotor tertahan

V_{br} = tegangan terminal stator pada keadaan rotor tertahan

Motor induksi yang dipakai adalah motor induksi dengan rotor sangkar tunggal. Secara umum X_s dan X_r diasumsikan sama [4]. Sehingga:

$$X_s = X_r = \frac{1}{2} X_{br} \quad \text{.....(2.50)}$$

Besarnya resistansi yang diukur pada terminal stator pada keadaan tanpa beban (X_{nl}) mendekati sama dengan $X_s + X_m$, yang merupakan reaktansi diri stator, sehingga:

$$X_{ss} = X_{nl} = X_s + X_m \quad \text{..... (2.51)}$$

Resistansi stator dapat dipandang sebagai harga DC- nya, maka resistansi rotor dapat ditentukan sebagai berikut:

$$R = R_s - R_{br} \quad \text{.....(2.52)}$$

$$X_{rr} = X_r + X_m \quad \text{.....(2.53)}$$

$$Rr = R\left[\frac{X_{rr}}{X_m}\right]^2 \quad \text{.....(2.54)}$$

BAB III

TEORI DASAR ALGORITMA GENETIKA DAN *EVOLUTIONARRY PROGRAMMING*, SERTA APLIKASI TERHADAP PENENTUAN EFISIENSI MOTOR INDUKSI

3.1 Algoritma Genetika[5]

Algoritma Genetika merupakan metode *adaptive* yang bisa digunakan untuk memecahkan suatu pencarian nilai dalam sebuah masalah. Algoritma ini didasarkan pada proses genetik yang ada dalam makhluk hidup yaitu perkembangan generasi dalam sebuah populasi yang alami, secara lambat laun mengikuti prinsip seleksi alam "siapa yang kuat, dia yang bertahan (*survive)*". Dengan meniru proses ini, algoritma genetika dapat digunakan untuk mencari solusi permasalahan-permasalahan dalam dunia nyata.

Algoritma Genetika ditemukan oleh John Holland pada awal tahun 1970 yang dilandasi oleh sifat-sifat evolusi alam. Holland percaya bahwa ini sangat cocok digabungkan dalam sebuah algoritma komputer, menghasilkan sebuah teknik penyelesaian untuk permasalahan-permasalahan yang sulit dengan langkah alami yaitu melalui evolusi. John Holland mulai bekerja dengan algoritma yang dibentuk dari string-string biner 1 dan 0 yang disebut kromosom. Seperti halnya alam, algoritma ini menyelesaikan permasalahan-permasalahan dengan menemukan kromosom-kromosom yang baik dengan memanipulasi materi dan sifat (*gene*) kromosom-kromosom. Algoritma ini tidak mengetahui tipe permasalahan yang akan diselesaikan. Hanya informasi yang telah diberikan dari

evaluasi berupa nilai *fitness* setiap kromosom dengan nilai *fitness* terbaik yang bertahan hidup dan selalu diproduksi.

Sebelum Algoritma Genetika dijalankan, maka sebuah kode yang sesuai (representasi) untuk persoalan harus dirancang. Titik solusi dalam ruang permasalahan dikodekan dalam bentuk kromosom/ string yang terdiri dari komponen genetik terkecil yaitu gen. pemakaian bilangan seperti integer, *floating point* dan abjad sebagai *allele* (nilai gen) memungkinkan penerapan operator genetika yaitu proses produksi *(reproduction)*, pindah silang *(crossover)*, mutasi *(mutation)* untuk menciptakan himpunan titik-titik solusi. Untuk memeriksa hasil optimasi, kita membutuhkan fungsi fitness yang menandakan gambaran hasil *(solution)* yang sudah dikodekan. Selama proses, induk harus digunakan untuk reproduksi, pindah silang dan mutasi untuk menciptakan keturunan *(offspring)*. Jika Algoritma Genetika didesain dengan biak, populasi akan mengalami konvergensi dan akan mendapatkan sebuah solusi yang optimum. Algoritma Genetika memiliki empat dasar kerja yaitu:

1. Bekerja dengan mengkodekan parameter-parameter permasalahan dan tidak bekerja secara langsung dengan parameter-parameter tersebut.
2. Mencari solusi masalah dari sejumlah populasi kandidat solusi, tidak hanya memproses satu solusi saja.
3. Hanya memperhitungkan fungsi *fitness* setiap kandidat solusi untuk mendapatkan hasil optimum global.
4. Menggunakan aturan transisi secara probabilistik bukan deterministik.

3.1.1 Istilah-istilah Algoritma Genetika[5]

Algoritma Genetika menggunakan mekanisme genetika yang ada pada proses alami dan sistem buatan. Istilah-istilah yang digunakan adalah gabungan dari dua disiplin ilmu, yaitu ilmu Biologi dan ilmu komputer. Mitsuo Gen dan Runwei Cheng (1997) menjelaskan istilah-istilah yang digunakan dalam Algoritma Genetika sebagai berikut:

Tabel 3-1

Istilah yang digunakan dalam Algoritma Genetika

Istilah	Keterangan
Kromosom	Individu berupa segmen string yang sudah ditentukan
Gen	Bagian dari string
Loci	Posisi dari gen
Allele	Nilai yang dimasukkan dalam gen
Phenotype	String yang merupakan solusi terakhir
Genotype	Sejumlah string hasil perkawinan yang berpotensi sebagai solusi

Terdapat beberapa parameter yang digunakan dalam Algoritma Genetika. Parameter tersebut digunakan untuk melihat kompleksitas dari Algoritma Genetika. Parameter yang digunakan tersebut adalah:

1. Jumlah Generasi (*MAXIGEN*)

   Merupakan jumlah perulangan (iterasi) dilakukannya rekombinasi dan seleksi. Jumlah generasi ini mempengaruhi kestabilan *output* dan lama iterasi (waktu proses Algoritma Genetika). Jumlah generasi yang besar dapat mengarahkan ke arah solusi yang optimal, namun akan membutuhkan waktu yang lama. Sedangkan jika jumlah generasinya terlalu sedikit maka solusi akan terjebak pada lokal optimum.

2. Ukuran Populasi (*POPSIZE)*

Ukuran populasi mempengaruhi kinerja dan efektifitas dari Algoritma Genetika. Jika ukuran populasi kecil maka populasi tidak menyediakan cukup materi untuk mencakup ruang permasalahan, sehingga pada umumnya kinerja Algoritma Genetika menjadi buruk. Dalam hal ini dibutuhkan ruang yang lebih besar untuk mempresentasikan keseluruhan ruang permasalahan. Selain itu penggunaan populasi yang besar dapat mencegah terjadinya konvergensi pada wilayah lokal. Zbigniew Michalewics (1996) berpendapat banyak aplikasi Algoritma Genetika mempergunakan populasi pada range 50-100 [6]

Probabilitas Crossover (Pc)

Probabilitas crossover ini digunakan untuk mengendalikan frekuensi operator *crossover*. Dalam hal ini, dalam populasi terdapat *Pc x POPSIZE* struktur (individu) yang melakukan pindah silang. Semakin besar nilai *probabilitas crossover* maka semakin cepat struktur baru yang diperkenalkan dalam populasi. Namun jika *probabilitas crossover* terlalu besar maka struktur dengan nilai fungsi obyektif yang baik dapat hilang dengan lebih cepat dari seleksi. Sebaliknya jika probabilitas terlalu kecil akan menghalangi proses pencarian dalam proses Algoritma Genetika. Zbigniew Michalewics (1996) berpendapat banyak aplikasi Algoritma Genetika mempergunakan angka probabilitas crossover pada range $0.65 - 1$[6].

Probabilitas Mutasi (*Pm*)

Mutasi digunakan untuk meningkatkan variasi populasi digunakan untuk menentukan tingkat mutasi yang terjadi, karena frekuensi terjadinya mutasi

tersebut menjadi *Pm x POPSIZE X N*, dimana N adalah panjang struktur / gen dalam satu individu. Probabilitas mutasi yang rendah akan menyebabkan gen-gen yang berpotensi tidak dicoba. Dan sebaliknya, tingkat mutasi yang tinggi akan menyebabkan keturunan akan semakin mirip dengan induknya. Dalam Algoritma Genetika, mutasi menjalankan aturan penting yaitu:

1. Mengganti gen-gen yang hilang pada proses seleksi
2. Menyediakan gen-gen yang tidak muncul pada saat inisialisasi awal populasi

Zbigniew Michalewics (1996) berpendapat banyak aplikasi Algoritma Genetika mempergunakan angka probabilitas mutasi pada daerah range 0.001 – 0.01.

Panjang Kromosom (*NVAR)*

Panjang kromosom berbeda-beda sesuai dengan model permasalahan. Titik solusi dalam ruang permasalahan dikodekan dalam bentuk kromosom/ string yang terdiri dari komponen genetic terkecil yaitu gen. pengkodean dapat memakai bilangan seperti string biner, integer, *floating point,* dan abjad.

3.1.2 Proses Algoritma Genetika [6]

Sangat perlu untuk mengetahui proses dalam Algoritma Genetika. Dibawah ini akan diuraikan mengenai hal itu, dimana uraian ini merupakan penjabaran dari Algoritma Genetika seperti penjelasan pada bagian berikutnya.

A. Pengkodean atau Representasi

Langkah pertama kali yang dilakukan dalam penggunaan Algoritma Genetika adalah melakukan pengkodean atau representasi terhadap permasalahan yang akan dilakukan.

Secara umum Algoritma Genetika dibentuk oleh serangkaian kromosom yang ditandai dengan *xi (I = 1,2 .....N)*. Setiap elemen dalam kromosom ini adalah variabel string yang disebut gen, berisi nilai-nilai atau *allele*. Variabel-variabel ini dapat dinyatakan dalam bentuk bilangan biner, bilangan real (*floating point)*, integer, abjad.

Selanjutnya beberapa kromosom dibentuk dan berkumpul membentuk populasi. Populasi inilah populasi awal bagi Algoritma Genetika untuk awal melakukan pencarian.

B. Fungsi *Fitness* (Fungsi Evaluasi)

Dalam Algoritma Genetika, sebuah fungsi *fitness f (x)* harus dirancang untuk masing-masing permasalahan yang akan diselesaikan. Dengan menggunakan kromosom tertentu, fungsi obyektif atau fungsi evaluasi akan mengevaluasi status masing-masing kromosom. Setiap gen *xi (I = 1,2, ... N)* dipergunakan untuk menghitung *fk (x) (k=1,2 ....POPSIZE)*.

C. Seleksi

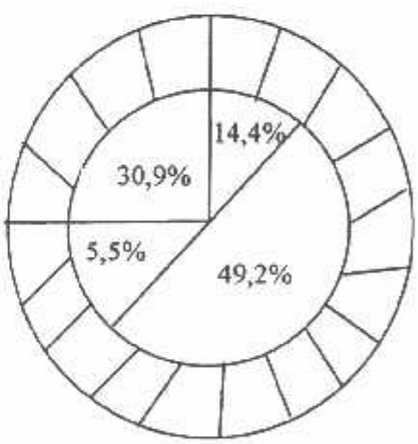
Pada Algoritma Genetika terdapat proses seleksi yaitu proses pemilihan kromosom yang akan di-*crossover*-kan dengan kromosom dari individu lain. Masalah yang paling mendasar pada proses ini adalah bagaimana proses penyeleksiaannya. Menurut teori Darwin proses seleksi individu adalah : "*individu terbaik akan tetap hidup dan menghasilkan keturunan*". Pada proses seleksi ini dapat menggunakan banyak metode seperti *roulette wheel selection, rank selection, elitism* dan lain sebagainya.

- ***Roulette Wheel Selection***

Dimana setiap individu memiliki harga *fitness* sehingga didapatkan probabilitas individual ($f_{(i)} / \sum f_{(i)}$) tersebut di-*copy*-kan pada populasi yang baru. Untuk individu yang memiliki probabilitas 20% untuk jumlah populasi 10 maka kemungkinan individu tersebut dapat terpilih sebanyak dua kali. Ilustrasi kerja operator ini dapat digambarkan seperti pada gambar 3-1.

Adapun logaritma dari *roulette-wheel* adalah sebagai berikut:

a. Menjumlahkan *fitness* dari seluruh anggota populasi
b. Membangkitkan nilai k, suatu nilai *random* antara 0 dan total *fitness*-nya
c. Menjumlahkan *fitness* dari kromosom-kromosom dari populasi mulai 0 hingga total *fitness* lebih besar atau sama dengan nilai k lalu ambil kromosom tersebut.



Gambar 3-1

Roulette – Wheel

Sumber : David E. goldberg, " **Genetic Algorithm in Search, Optimazation, and Machine Learning**". (The University of Alabama. 1989)

- ***Rank Selection***

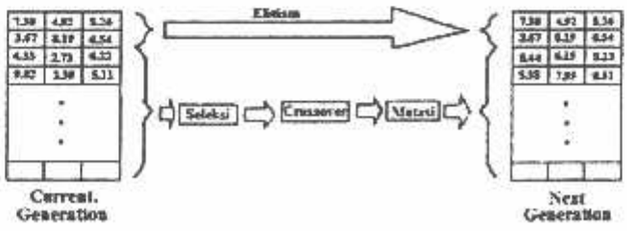
Apabila *fitness* yang dimiliki oleh suatu kromosom dalam populasi berbeda terlalu jauh dari kromosom lainnya maka hal ini dapat menjadi

permasalahan. Misalnya bila kromosom terbaik mempunyai *fitness* yang menyebabkan besarnya tempat yang dimilikinya dalam *roulette wheel* sebesar 90% maka kromosom-kromosom yang lain akan mempunyai peluang yang terlalu kecil untuk diseleksi.

Rank selection pertama kali merangking populasi dan kemudian setiap kromosom diberi nilai *fitness* baru berdasarkan hasil rangking tersebut yang pertama akan mempunyai *fitness 1*, yang kedua akan mempunyai *fitness 2* dan seterusnya sampai yang terakhir akan mempunyai *fitness* N. dengan demikian semua kromosom akan mempunyai peluang untuk diseleksi.

3.1.3 Elitism

Selama membuat populasi baru dengan *crossover* dan *mutasi*, kemungkinan akan terjadi kehilangan kromosom terbaik (*best / few best). Elitism* adalah metode yang pertama kali meng*copy*-kan kromosom terbaik (*best few best)* kedalam populasi baru. Sisanya dikerjakan dengan cara biasa, yaitu melalui *seleksi, crossover, dan mutasi. Elitism* dapat secara cepat meningkatkan performansi dari Algoritma Genetika karena *elitism* menghindarkan hilangnya solusi terbaik (*best / few best)* yang telah ditemukan. Ilustrasi kerja operator ini dapat digambarkan seperti pada gambar 3-2.



Gambar 3-2

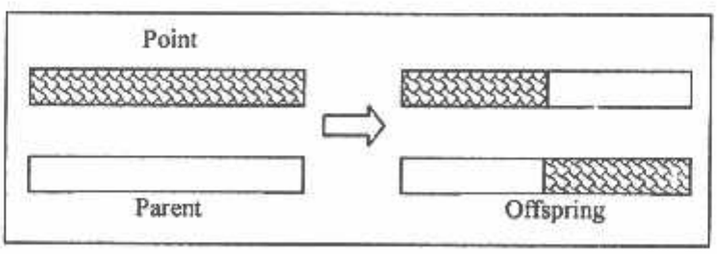
Pembentukan Next Generation dalam Algoritma Genetika

Sumber : Mitsuo Gen , Runwei Cheng " **Genetic Algorithm and Engineering Design**" (John Willey 7 Son, Inc.1994).p.7

3.1.4 *Crossover* (Pindah Silang)

Fungsi dari *crossover* adalah menghasilkan kromosom anak dari kombinasi materi-materi gen dua kromosom induk.c ara kerjanya dengan membangkitkan sebuah nilai random r_k dimana k = 1.2, ..., *POPSIZE. Probabilitas crossover (Pc)* ditentukan dan digunakan untuk mengendalikan frekuensi operator *crossover.* Apabila nilai $r_k < P_c$ maka kromosom ke – k terpilih untuk mengalami *crossover. Crossover* yang paling sederhana adalah *one point crossover.* Posisi titik persilangan (point) ditentukan secara *random* pada *range* satu sampai panjang kromosom. Kemudian nilai *offspring* diambil dari dua *parent* tersebut dengan batas titik persilangan tersebut. Ilustrasi kerja operator ini digambarkan seperti pada gambar 3-3.

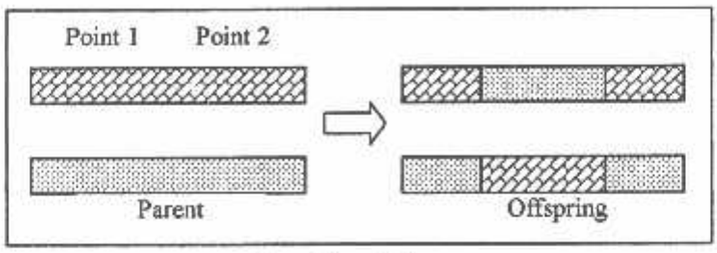
Kemudian ditingkatkan lagi dengan menggunakan *two point crossover.* Penentuan posisi titik persilangan sama seperti *one point crossover* sebelumnya. Pemilihan secara *random* dilakukan 2 kali. Kemudian nilai *offspring* diambil dari dua *parent* tersebut dengan batas dua titik persilangan tersebut. Ilustrasi kerja operator ini digambarkan seperti pada gambar 3-4.



Gambar 3-3

Ilustrasi Operator dengan *One Point Crossover*

Sumber : Mitsuo Gen , Runwei Cheng " **Genetic Algorithm and Engineering Design**" (John Willey 7 Son, Inc.1994).p.7

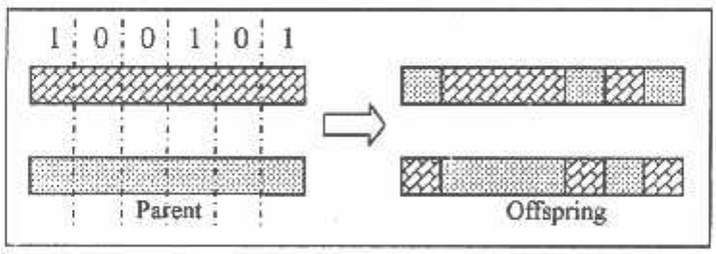


Gambar 3-4

Ilustrasi Operator dengan *Two Point Crossover*

Sumber : Mitsuo Gen , Runwei Cheng " **Genetic Algorithm and Engineering Design**" (John Willey 7 Son, Inc 1994).p.7

Untuk *uniform crossover* dibangkitkan suatu nilai *random* 0 dan 1 sepanjang jumlah kromosom untuk nilai loci. Jika nilai yang dibangkitkan mempunyai nilai 1 maka *allele parent* 2 dan *offspring* 2 untuk loci tersebut diambil dari *allele parent* 1 dan *offspring* 2 untuk loci tersebut diambil dari *allele parent* 2. Ilustrasi kerja operator ini digambarkan seperti pada gambar 3-5.



Gambar 3-5

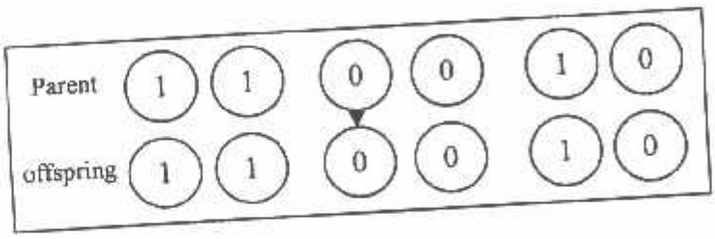
Ilustrasi Operator Crossover dengan *Uniform Crossover*

Sumber : Mitsuo Gen , Runwei Cheng " **Genetic Algorithm and Engineering Design**" (John Willey 7 Son, Inc.1994).p.7

3.1.5 Mutasi (*Mutation)*

Operator mutasi digunakan untuk memodifikasi satu atau lebih nilai gen dalam satu individu. Cara kerjanya dengan membangkitkan sebuah nilai *random* r_k dimana k = 1,2, ..., *NVAR* (panjang kromosom). Probabilitas mutasi (P_m) ditentukan dan digunakan untuk mengendalikan frekuensi operator mutasi. Apabila nilai *random* r_k, P_m maka gen ke-k kromosom tersebut terpilih untuk mengalami mutasi. Mutasi dengan mengganti gen 0 dengan 1 atau sebaliknya gen

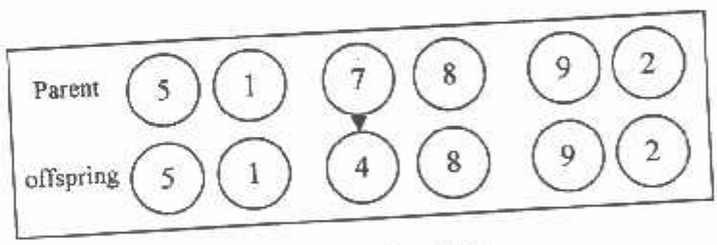
1 dengan 0. Biasanya disebut *flip* yaitu membalik nilai ke 1 atau 1 ke 0. Illustrasi kerja operator untuk represetansi *string biner* digambarkan pada gambar 3-6. Untuk bentuk representasi *integer atau floating point,* atau selain *string biner,* seperti gambar 3-7, proses mutasi terjadi apabila nilai $r_k < P_m$ memenuhi maka gen ke-k digantikan oleh suatu nilai *random* yang dibangkitkan pada *range* tertentu sesuai dengan pembentukan populasi awal.



Gambar 3-6

Ilustrasi Operator Mutasi untuk Representasi String Biner

Sumber : Mitsuo Gen , Runwei Cheng " **Genetic Algorithm and Engineering Design**" (John Willey 7 Son, Inc.1994).p.7



Gambar 3-7

Ilustrasi Operator Mutasi untuk Represetasi Integer

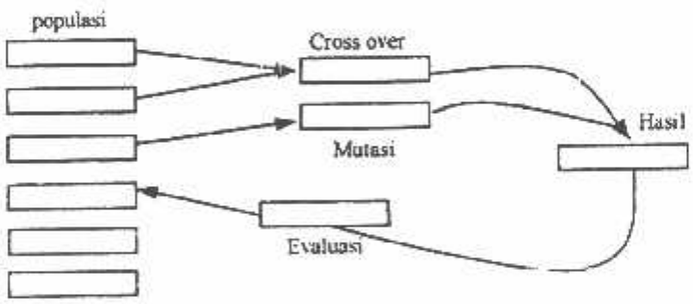
Sumber : Mitsuo Gen , Runwei Cheng " **Genetic Algorithm and Engineering Design**" (John Willey 7 Son, Inc.1994).p.7

Untuk kromosom induk seperti gambar 3-7 diatas yaitu 5-1-7-8-9-2, proses mutasi adalah dibangkitkan sebuah nilai random r_k, [k = 1 ....*NVAR*]. Misalkan pada saat k = 3 nilai $r_3 < P_m$ maka gen ke-n suatu probabilitas mutasi P_m. Posisi elemen pada kromosom yang akan mutasi ditentukan secara *random*. Mutasi dikerjakan dengan cara melakukan perubahan pada elemen tersebut.

3.1.6 Adaptasi Algoritma Genetika ke Masalah Penentuan Efisiensi Motor Induksi Tiga Phasa[1]

Algoritma Genetika adalah metode alternatif yang bisa menentukan efisiensi motor induksi, sehingga diperoleh hasil yang mendekati nilai sebenarnya (hasil pengujian). Algoritma Genetika pada mekanisme seleksi alam, individu dan sebuah populasi dikodekan secara biner, populasi pertama dibangkitkan secara random. Generasi baru dibuat dengan mengaplikasikan 3 operator berikut terhadap sebuah populasi yaitu: elitism (reproduksinya), crossover dan mutasi dimana reproduksi adalah proses yang tergantung pada fungsi tujuan (*objective function*).

Algoritma genetika merupakan *objective function* yang didasarkan pada suatu kriteria kinerja untuk menentukan error.



Gambar 3-8

Algoritma Genetika

Sumber : Mitsuo Gen , Runwei Cheng " **Genetic Algorithm and Engineering Design**" (John Willey 7 Son, Inc.1994).p.7

3.1.6.1 Pengkodean

Sebuah motor induksi tiga phasa terdiri dari parameter yaitu X_1, R_2, Rm, dan Xm mewakili satu individu, yang terdiri dari 3 string atau kromosan. Setiap kromosom ditentukan panjangnya 30 terdiri atas string yang dibatasi nilai

minimum dan nilai maksimum. Di dalam string terdapat beberapa gen, dimana masing-masingnya adalah kode biner antara 0 atau 1.

Kromoson pertama adalah X_1 dimana dalam X_1 ditentukan bahwa X_1 minimum adalah 2 dan X_1 maksimum adalah 4. Maka arti dari pengkodean 0 atau 1 yaitu:

2									4
1	0	1	0	1	0	1	0	1	0
2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
512	0	128	0	32	0	8	0	2	0

Gambar 3-9

Pengkodean Untuk Parameter X_1

Untuk mencari X_1 dengan melalui persamaan sebagai berikut:

$$X_1 = X_{1min} + \frac{(X_{1max} - X_{1min})\,x}{2^{10} - 1} \quad \text{.......................................(3.1)}$$

Kromoson kedua adalah nilai R_2 dimana R_2 ditentukan bahwa R_2 minimum 3 dan maksimum 5. Maka arti dari pengkodean tersebut adalah:

3									5
0	1	0	1	0	1	0	1	0	1
2^9	2^8	2^7	20^6	2^5	2^4	2^3	2^2	2^1	2^0
0	256	0	64	0	16	0	4	0	1

Gambar 3-10

Pengkodean Untuk Parameter R_2

Untuk mencari R_2 dengan melalui persamaan sebagai berikut:

$$R_2 = R_{2min} + \frac{(R_{2max} - R_{2min})\, x}{2^{10} - 1} \quad \ldots\ldots(3.2)$$

Kromoson ketiga adalah nilai R_m dimana dalam R_m ditentukan bahwa R_m minimum 378 dan maksimum 380. Maka arti dari pengkodean tersebut adalah:

378									380
1	0	0	1	0	0	1	0	0	1
2^9	2^8	2^7	20^6	2^5	2^4	2^3	2^2	2^1	2^0
512	0	0	64	0	0	8	0	0	1

Gambar 3-11

Pengkodean Untuk Parameter R_m

Untuk mencari R_m dengan melalui persamaan sebagai berikut:

$$R_m = R_{m\,min} + \frac{(R_{m\,max} - R_{m\,min})\, x}{2^{10} - 1} \quad \ldots\ldots(3.3)$$

Kromoson ketiga adalah nilai X_m dimana dalam X_m ditentukan bahwa X_m minimum 47 dan maksimum 45. Maka arti dari pengkodean tersebut adalah:

47									45
1	0	0	1	0	1	1	1	0	1
2^9	2^8	2^7	20^6	2^5	2^4	2^3	2^2	2^1	2^0
512	0	0	64	0	16	8	4	0	1

Gambar 3-12

Pengkodean Untuk Parameter X_m

Untuk mencari X_m dengan melalui persamaan sebagai berikut:

$$X_m = X_{m\,min} + \frac{(X_{m\,max} - X_{m\,min})x}{2^{10} - 1} \quad \ldots\ldots(3.4)$$

Setelah nilai X_1, R_2, R_m dan X_m diperoleh maka dimasukkan ke persamaan-persamaan rangkaian ekivalen motor induksi. Parameter input yang digunakan ada pada persamaan berikut :

$$F_{objective} = \sum_{i=1}^{n}\left|\frac{I_{Ii,cal}}{I_{Iimea}} - 1\right|^2 + \sum_{i=1}^{n}\left|\frac{P_{inputi,cal}}{P_{inputi,mea}} - 1\right|^2 \quad \ldots\ldots(3.5)$$

$$Fitness = \frac{100}{100 + F_{objective}} \quad \ldots\ldots(3.6)$$

Perhitungan diatas adalah untuk satu individu sedangkan untuk individu lainnya dilakukan dengan cara yang sama dengan catatan untuk masing-masing kromosom nilainya random. Setelah semua individu dihitung fitnesnya maka dicari prosentase fitness untuk masing-masing individu yaitu:

$$Prosentase = \frac{Fitness_n}{Sum\ Fitness} \times 100\,\% \quad \ldots\ldots(3.7)$$

Dimana:

$Fitness_n$ = Fitness individu ke-n

Sumfitness = Jumlah Fitness seluruh individu

3.1.6.2 Populasi awal

Untuk sebuah motor induksi tiga phasa, populasi awal dibangkitkan secara random (acak). Dalam pembahasan kali ini populasi dibangkitkan sebanyak 50 populasi.

Untuk membuat 1 individu misalnya untuk individu pertama dibuat 4 kromosom yang masing-masing adalah X_1, R_2, X_m, dan R_m. Selanjutnya di dalam

keempat unsur tersebut dibangkitkan string biner secara acak. Selanjutnya proses di atas diulang sampai n kali.

3.1.6.3 Elitism , crossover dan mutasi

Generasi baru dibuat dari hasil yang diperoleh dari generasi sebelumnya. Untuk setiap individu, pertama kita menghitung dan menormalisasi nilai *objective function.* Semakin tinggi *objective function-* nya maka individu akan semakin baik. Reproduksi (kompetisi), crossover & mutasi dilakukan secara bergantian. Kemudian 2 individu diambil secara acak. Probabilitas pengambilan individu ini berhubungan langsung dengan nilai *objective function-* nya. Penarikan ini dilakukan dengan metode *biased roulette wheel* seperti gambar 3-1.

Proses *crossover* dapat dilakukan dengan probabilitas PC. Jika nilai yang dibangkitkan kurang dari nilai Pc maka diadakan *crossover* dan sebaliknya jika nilai yang dibangkitkan melebihi Pc maka tidak perlu diadakan *crossover*. Pada proses ini, posisi untuk memotong keempat string (X=1, R=2, X_m, dan R_m) untuk kedua individu dipilih secara acak (*random*).

3.1.7 Objective Function

Objective function adalah parameter yang penting dalam Algoritma Genetika. Optimasi yang kita inginkan harus dipresentasikan secara matematis. *Objective function* yang buruk tidak dapat menghasilkan invidividu yang baik dan tidak dapat mencapai optimasi yang kita inginkan.

3.1.8 *Objective Function* Untuk Permasalahan Penentuan Efisiensi Motor Induksi Tiga Phasa

Tujuan dari optimasi genetik adalah untuk meminimalisasi kesalahan (*error*) antara pengukuran dan perhitungan parameter. Parameter input yang digunakan dalam perhitungan terdapat dalam persamaan berikut :

$$F_{objective} = \sum_{i=1}^{n} \left| \frac{I_{1i,cal}}{I_{1imea}} - 1 \right|^2 + \sum_{i=1}^{n} \left| \frac{P_{inputi,cal}}{P_{inputi,mea}} - 1 \right|^2 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.8)$$

Dan dicari *fitness* nya adalah

$$Fitness = \frac{100}{100 + F_{objective}} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.9)$$

Proses ini berulang untuk setiap *string* baru (parameter motor yang baru) sampai menghasilkan 150 nilai fungsi *fitness.*

Hasil evaluasi pada proses algoritma genetika digunakan untuk mencari nilai *error* terkecil .Nilai *error* yang diperoleh digunakan untuk menentukan parameter motor induksi tersebut. Parameter motor induksi tersebut didapat dari rangkaian ekivalen motor induksi.

Kemudian parameter motor induksi diacak untuk mendapatkan nilai optimum dengan metode algoritma genetika sehingga didapat nilai *fitness* (kemampuan) yang maksimum.

3.1.9 Algoritma Program

3.1.9.1 Algoritma Program Pemecahan Masalah Penentuan Efisiensi Motor Induksi Tiga Phasa Menggunakan Algoritma Genetika adalah:

1. Memasukkan input-an data parameter motor induksi
2. Menentukan parameter inputan Algoritma Genetika yang meliputi jumlah populasi, maksimum generasi, nilai kemungkinan crossover, nilai kemungkinan mutasi dan panjang kromosom tiap-tiap individu.
3. Generasi = 0, Populasi = 0
4. Membangkitkan populasi
5. Melakukan fitness dan kromosom pada tiap-tiap individu
6. Melakukan proses statistik
7. Melakukan proses seleksi
8. Melakukan proses crossover
9. Melakukan proses mutasi
10. Proses no. 6, 7, 8 diulang sampai offspring / anak sama dengan jumlah populasi
11. Menghitung fitness dari offspring / anak
12. Melakukan proses Elitsm
13. Melakukan proses statistik
14. Apakah generasi yang diinginkan sudah terpenuhi (max. Gen)
15. Jika "tidak" maka generasi = gen + 1, kembali ke langkah 6
16. Jika "ya" maka perhitungan berhenti

3.2 *EVOLUTIONARRY PROGRAMMING*

3.2.1 Teori *Evolutionary Programming* [7]

Evolutionary programming diperkenankan oleh Lawrence J. Fogel pada tahun 1960. *Evolutionarry Programming* merupakan metode stokastik yang biasa dipakai untuk memecahkan pencarian nilai dalam sebuah masalah optimasi. Metode ini berdasarkan teori evolusi, yaitu teori alam yang mengatakan bahwa "yang kuat yang menang", dan yang menang berkesempatan hidup lebih lama.

Evolutionarry Programming metode strategi optimasi yang merupakan cabang dari *Evolutionarry Computation* yang di dalamnya terdiri dari *Genetic Algorithm, Genetic Programming, Evolutionarry Strategies,* dan *Evolotionarry Programming.*

Perbedaan *Evolutionarry Programming* dan *Genetic Algorithm* ada pada proses operasi. Dalam metode *Evolutionarry Programming* tidak ada operasi *crossover* melainkan operasi *competition* (kompetisi).

Parameter –parameter Evolutionarry Programming antara lain :

- Jumlah generasi (MAXGEN)
  Merupakan jumlah iterasi dilakukannyarekombinasi dan seleksi, mempengaruhi kestabilan output dan lama itersi.
- Ukuran Populasi (POSIZE)
  Mempengaruhi kinerja dan efektifitas *Evolutionarry Programming.*
- Probabilitas Mutasi (PM)
  Untuk meningkatkan variasi populasi dan menentukan tingkat mutasi yang terjadi.
- Panjang kromosom.

3.2.2 Skema *Evolutionary Programming*

Evolutionary programming mencari solusi optimal dari proses optimasi dengan menyusun populasi dari kandidat penyelesaian dari jumlah generasi atau proses iterasi. Populasi yang baru dibentuk dari seluruh populasi yang ada menggunakan operator mutasi. Operator ini memberikan gangguan pada masing-

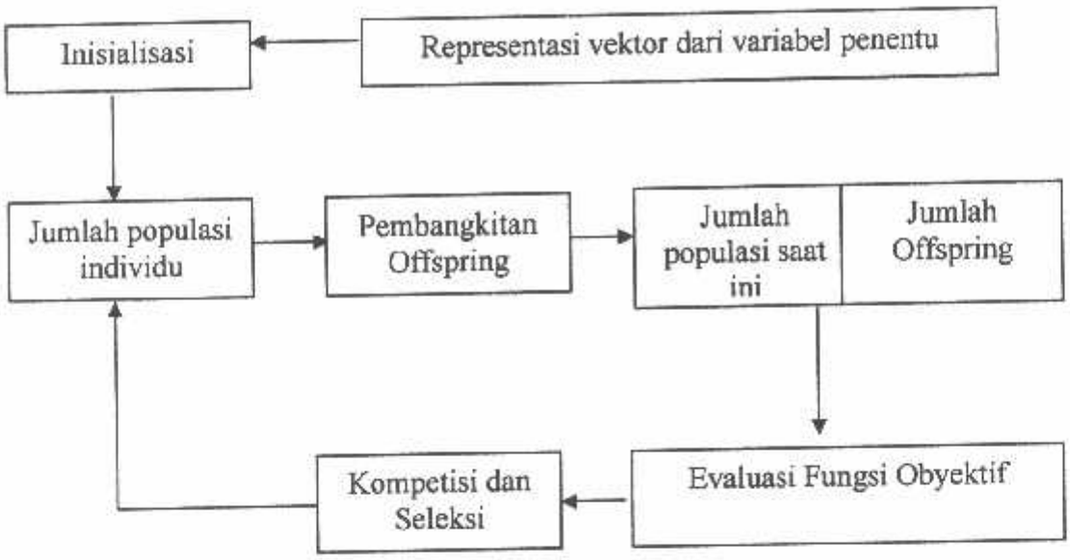
masing komponen dari tiap solusi dalam populasi secara random untuk memperoleh solusi baru.

Tingkat optimal dari masing-masing kandidat solusi yang baru atau individu yang baru dihitung dengan *fitness*, dimana dapat di definisikan sebagai fungsi biaya atau fungsi obyektif dari persoalan yang diselesaikan.

Keseluruhan prosesnya menggunakan skema kompetisi, masing-masing individu dalam populasi saling berkompetisi. Individu yang menang akan dijadikan hasil dan populasi yang akan digunakan untuk generasi selanjutnya. Untuk optimasi berlaku, dalam kompetisi solusi yang lebih kuat menggantikan solusi yang lemah. Dari seluruh populasi ini disusun penyelesaian optimal secara global.

Teknik *Evolutionary Programming* ini seperti proses iterasi dimana proses berhenti setelah mencapai kriteria yang ditentukan sebelumnya. Kriteria ini adalah jika penyelesaian yang spesifik sudah tercapai dalam proses iterasi atau berhenti setelah solusi terbaik tidak berubah dalam beberapa generasi yang telah ditentukan.

Skema dari *Evolutionarry Programming*:



Gambar 3.13 Skema *Evolutionary Programming*

Skema diatas dapat diterangkan sebagai berikut:

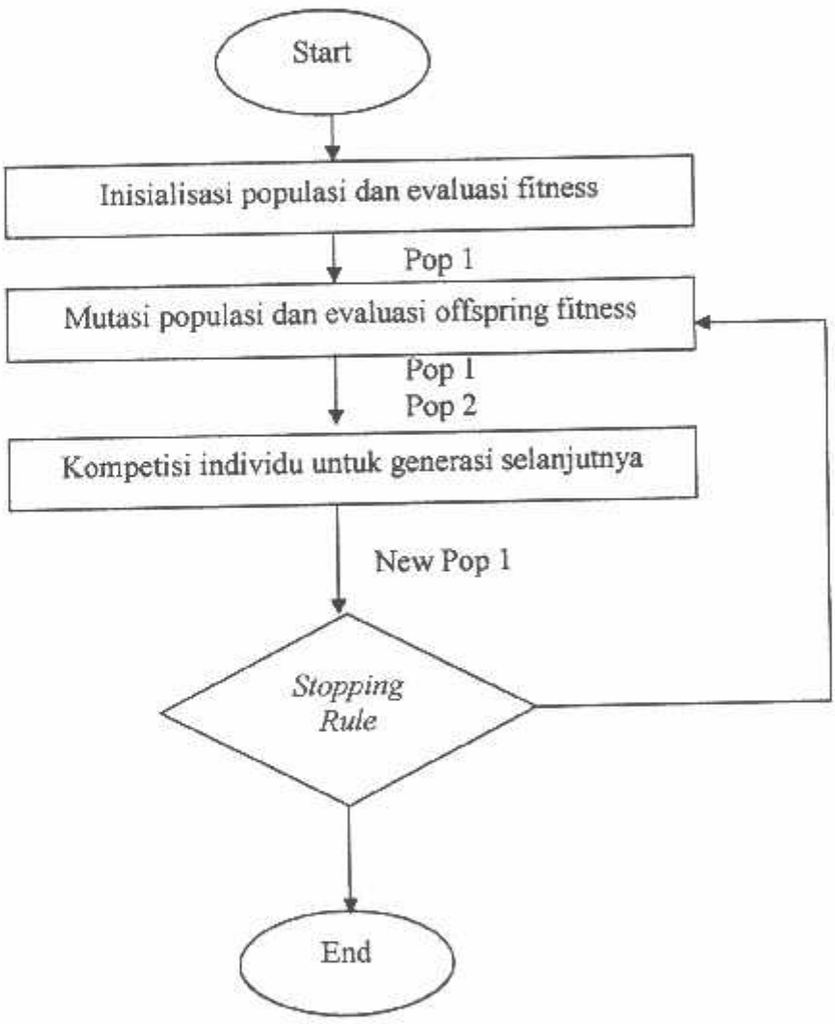
1. Representasi vektor, nilai real dari variabel yang diselesaikan dalam EP, direpresentasikan sebagai vektor p n- demensi yang berhubungan dengan fungsi obyektif Masing-masing vektor p disusun sebagai individu dalam populasi.
2. Inisialisasi, inisial populasi dari individu asal p_i , i = 1,..... , , diseleksi secara random dari range yang mungkin dari masing-masing dimensi, ini terdistribusi uniform.
3. Pembangkitan *offspring*, jumlah dari *offspring* p_i, i = 1, ..., Np, dibangkitkan dengan gaussian random variabel dengan mean nol atau standar deviasi untuk masing-masing komponen p_i, dimana individu termasuk individu asal dan *offspring* berada dalam kelompok seleksi.
4. Kompetisi dan seleksi, masing-masing individu dalam kelompok seleksi harus berdasarkan fungsi $f(p_i)$ dan $f(p_i')$. N_p individu dengan nilai fungsi terbaik

(minimum untuk proses minimasi) diseleksi untuk membentuk survivor set untuk pengambilan keputusan. Individu dalam survivor set menjadi individu asal untuk generasi selanjutnya.

5. *Stopping rules* proses pembangkitan dan seleksi dengan nilai fungsi terbaik terus berlangsung sampai nilai fungsi itu tidak berubah sampai beberapa generasi atau nilai yang ditentukan telah tercapai.

3.2.3 Flow Chart *Evolutionarry Programming*

Flowchart Evolutionary Programming digambarkan sebagai berikut:



Gambar 3.14 Flowchart *Evolutionary Programming*

Lima langkah pemrograman *Evolutionary Programming*

1. Seleksi representasi genetik untuk solusi persoalan. Representasi ini harus dapat menunjukkan semua informasi penting tentang solusi.
2. Pembuatan inisial populasi.
3. Seleksi dari fungsi evolusi.
4. Mendisain operator genetik.
5. Nilai untuk macam-macam parameter, mengacu pada parameter yang bisa adaptasi ke *evolutionary programming.*

3.2.4 Adaptasi *Evolutionarry Programming* ke Masalah Penentuan Efisiensi Motor Induksi Tiga Phasa

Evolutionarry Programming juga merupakan metode alternatif yang bisa menentukan efisiensi motor induksi, sehingga diperoleh hasil yang mendekati nilai sebenarnya (hasil pengujian). *Evolutionarry Programming* pada mekanisme seleksi alam, individu dan sebuah populasi (tanpa pengkodean), populasi pertama dibangkitkan secara random. Generasi baru dibuat dengan mengaplikasikan 2 operator berikut terhadap sebuah populasi yaitu: kompetisi dan mutasi dimana reproduksi adalah proses yang tergantung pada fungsi tujuan (*objective function*).

Evolutionarry Programming merupakan *objective function* yang didasarkan pada suatu kriteria kinerja untuk menentukan error.

3.2.5 Populasi awal

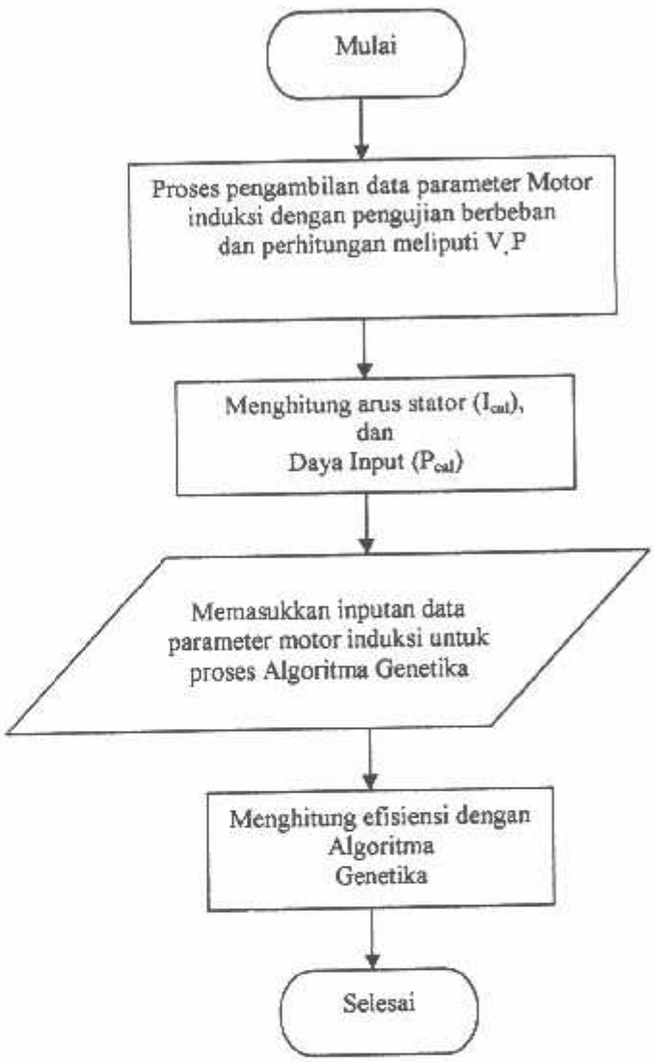
Untuk sebuah motor induksi tiga phasa, populasi awal dibangkitkan secara random (acak). Dalam pembahasan kali ini populasi dibangkitkan sebanyak 50 populasi.

3.2.6 Kompetisi dan mutasi

Generasi baru dibuat dari hasil yang diperoleh dari generasi sebelumnya. Untuk setiap individu, pertama kita menghitung dan menormalisasi nilai *objective function.* Semakin tinggi *objective function-* nya maka individu akan semakin baik. Kompetisi mutasi dilakukan secara bergantian. Kemudian 2 individu diambil secara acak. Probabilitas pengambilan individu ini berhubungan langsung dengan nilai *objective function-* nya. Penarikan ini dilakukan dengan metode *biased roulette wheel* .

3.2.7 Flowchart Algoritma Program

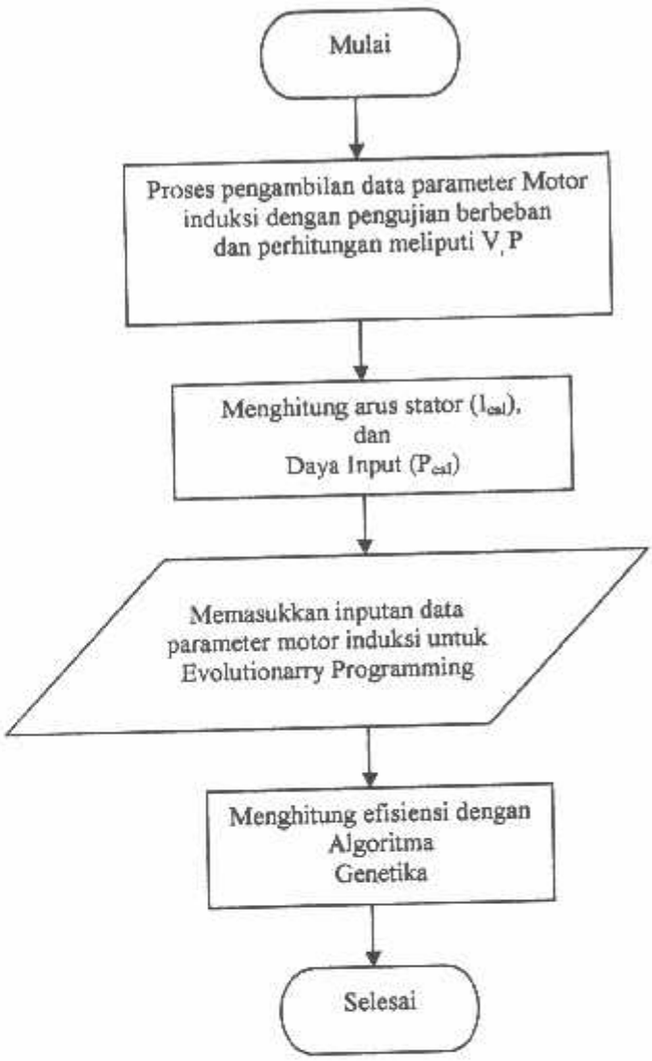
3.2.7.1 Flowchart Pemecahan Masalah untuk Metode Algoritma Genetika



Gambar 3-15

Flowchart Pemecahan Masalah dengan metode Algoritma Genetika

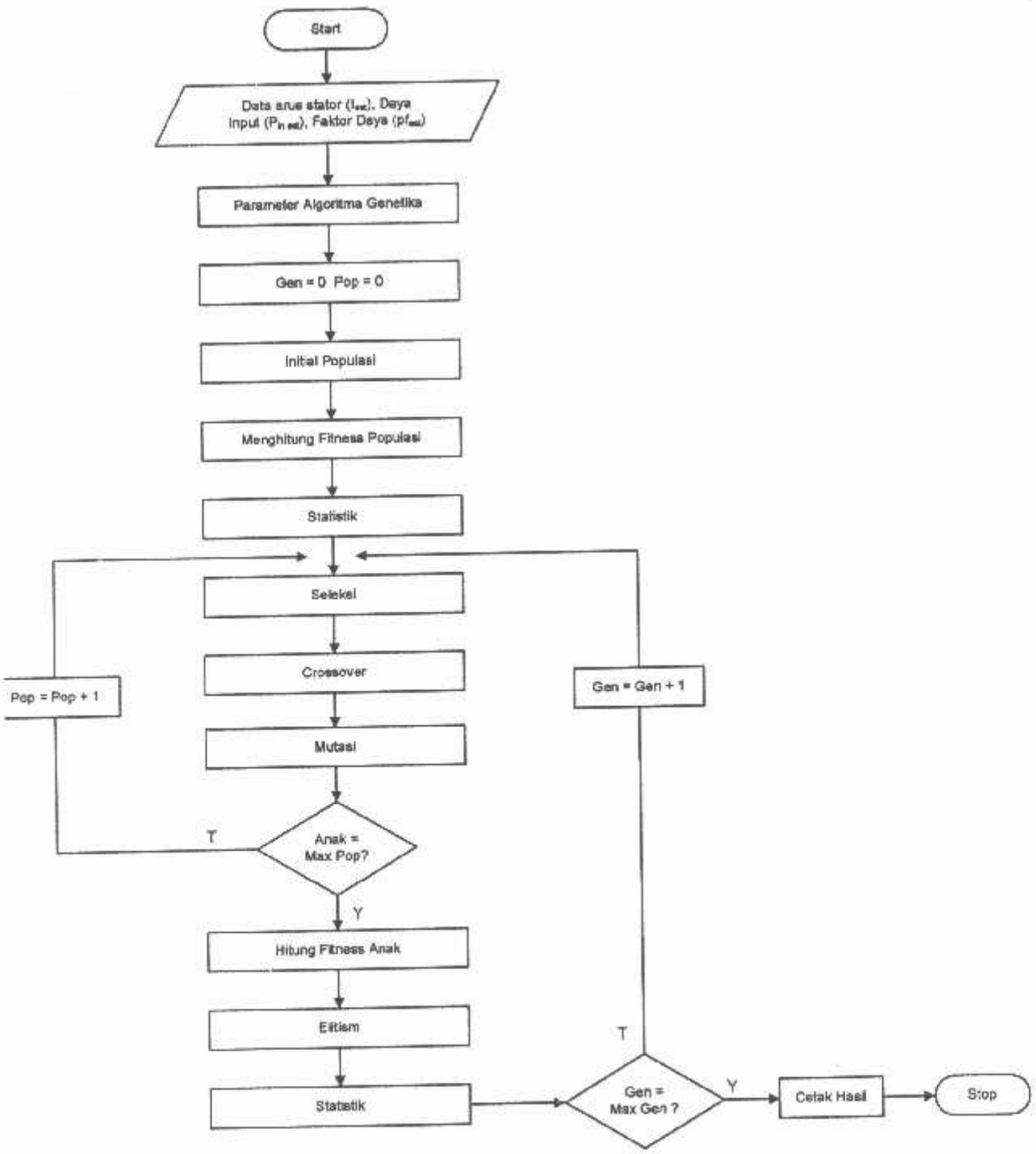
3.2.7.2 Flowchart Pemecahan Masalah untuk Metode *Evolutionarry Programming*



Gambar 3-16

Flowchart Pemecahan Masalah dengan metode Evolutionarry Programming

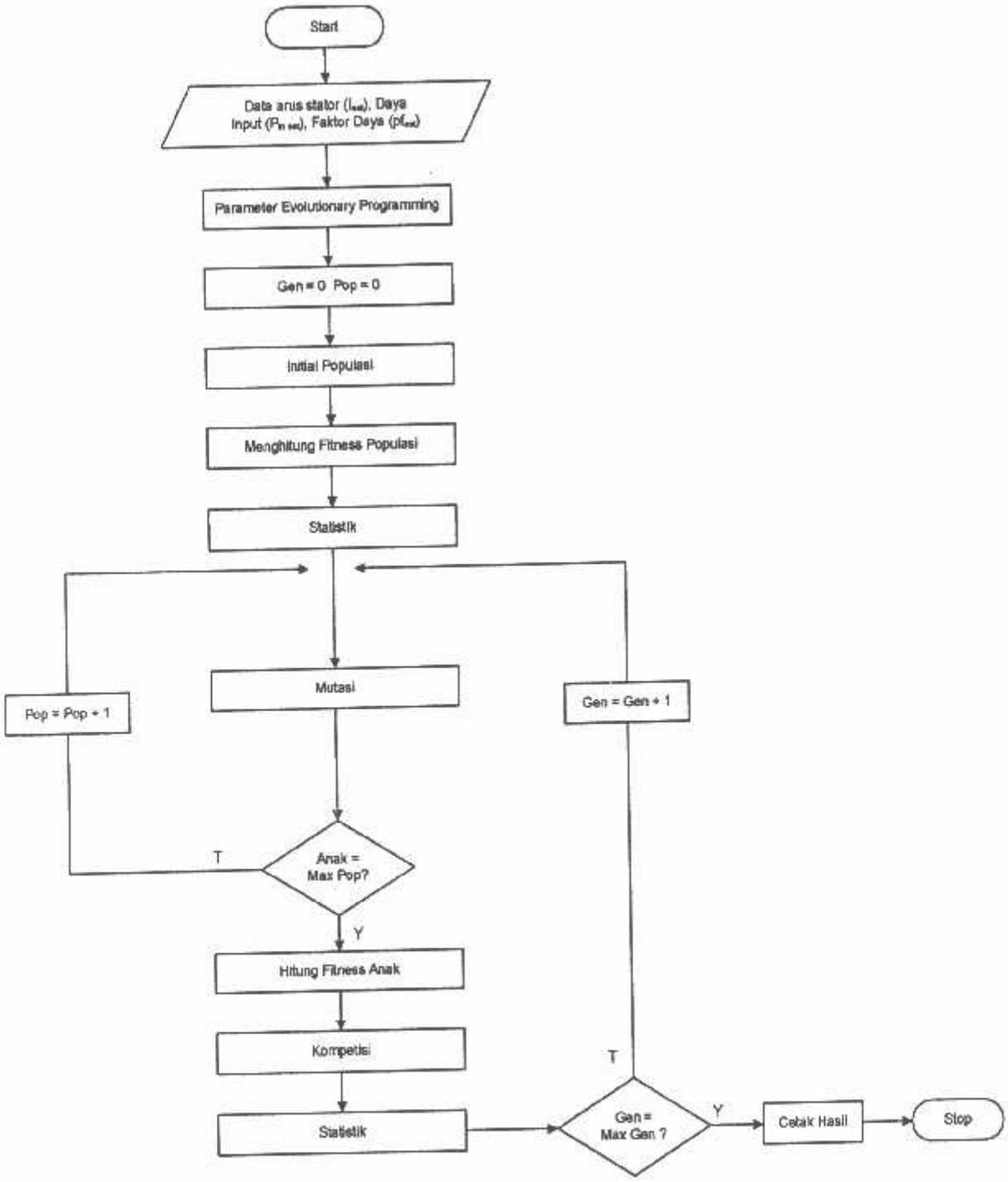
3.2.7.3 Flowchart Program Penentuan Efisiensi Motor Induksi Tiga Phasa dengan Metode Algoritma Genetika



Gambar 3-17

Flowchart Program Algoritma Genetika

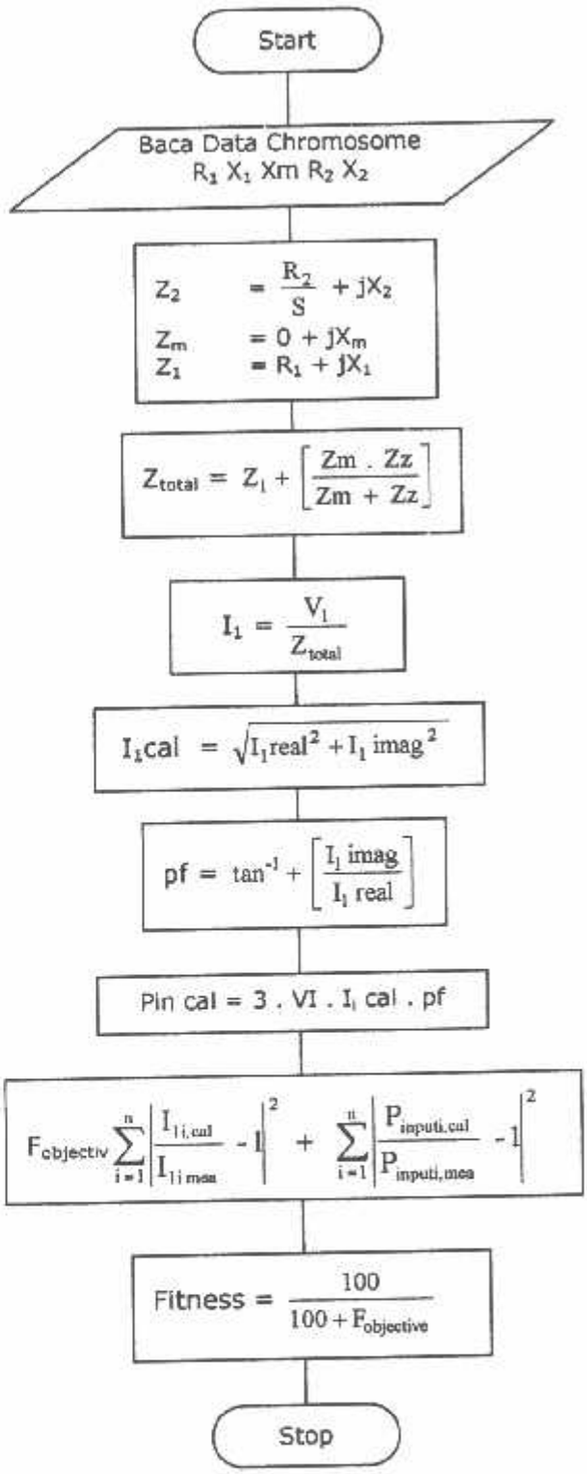
3.2.7.4 Flowchart Program Penentuan Efisiensi Motor Induksi Tiga Phasa dengan Metode *Evolutionarry Programming*



Gambar 3-18

Flowchart Program *Evolutionarry Programming*

3.2.7.5 Flowchart Algoritma Program Fitness



Gambar 3-19

Flowchart Algoritma Fitness

3.2.8 Algoritma Program Pemecahan Masalah Penentuan Efisiensi Motor Induksi Tiga Phasa Menggunakan *Evolutionary Programming* adalah:

1. Memasukkan input-an data parameter motor induksi
2. Menentukan parameter inputan *Evolutionary Programming* yang meliputi jumlah populasi, maksimum generasi, nilai kemungkinan crossover, nilai kemungkinan mutasi dan panjang kromosom tiap-tiap individu.
3. Generasi = 0, Populasi = 0
4. Membangkitkan populasi
5. Melakukan fitness dan kromosom pada tiap-tiap individu
6. Melakukan proses statistic
7. Melakukan proses mutasi
8. Proses no. 7 dan 8 diulang sampai offspring / anak sama dengan jumlah populasi
9. Menghitung fitness dari offspring / anak
10. Melakukan proses kompetisi
11. Melakukan proses statistic
12. Apakah generasi yang diinginkan sudah terpenuhi (max. Gen)
13. Jika "tidak" maka generasi = gen + 1, kembali ke langkah 6
14. Jika "ya" maka perhitungan berhenti

3.2 9 Algoritma Program Fitness

1. Memasukkan input-an data parameter motor induksi
2. Melakukan setingan untuk parameter X_1, R_2, R_m, X_m, minimum dan maksimum

3. Memasukkan data ke persamaan

$$F_{objective} = \sum_{i=1}^{n} \left| \frac{I_{li,cal}}{I_{li\,mea}} - 1 \right|^2 + \sum_{i=1}^{n} \left| \frac{P_{inputi,cal}}{P_{inputi,mea}} - 1 \right|^2$$

4. Menghitung nilai fitness yang dinyatakan oleh :

$$Fitness = \frac{100}{100 + F_{objective}}$$

5. Mencetak hasil

BAB IV
PENGAMBILAN DAN ANALISIS DATA

4.1 Pengujian Parameter Motor Induksi Tiga Phasa

Pengujian yang dilakukan untuk menganalisa efisiensi motor induksi 3 phasa ini adalah pengujian berbeban.

4.1.1 Alat- alat Yang Digunakan

1. Motor Induksi 3ϕ, DE LORENZO / DL 1021 dengan data:

Tegangan	:	220 / 380 (Δ / Y) volt
Arus	:	4,3 / 2,5 (Δ / Y) Amper
Cos φ	:	0,83
Frekwensi	:	50 Hz
Daya	:	1,1 kw
Putaran	:	2830 rpm
Kutup	:	2 kutup

2. Voltmeter (1 buah)
3. Ampermeter (1 buah)
4. Wattmeter 3 ϕ (2 buah)
5. Tachometer (1 buah)
6. AC Voltage Regulator

4.1.2 Pengujian Berbeban

Pada pengujian ini, motor diberikan beban 25%, 50%, 75%, dan 100%. Pengujian ini dimaksudkan untuk memperoleh nilai arus stator (I), daya input (P_{in}), kecepatan rotor (n_r), dan tegangan (V), yang nantinya digunakan untuk menentukan parameter motor induksi 3 phasa, sehingga dapat menentukan Pout, dan efisiensi bisa dihitung.

4.1.2.1 Data Hasil Pengujian Berbeban

Tabel 4-1

Data Hasil Pengukuran Berbeban

Beban	Tegangan	Arus I	Daya input P_{in}	Torsi beban	nr
%	V (volt)	(amp)	(watt)	(Nm)	rpm
25	220	1,709	564,14	1,66	2.808
50	220	2,348	780,66	2,17	2.711
75	220	2,902	953,54	2,52	2.615
100	220	3,363	1081,92	2,84	2.528

Tabel diatas dapat dilihat di lampiran

4.1.2.2 Analisa Data Hasil Pengujian Berbeban

a. Beban 25%

1) $ns = \frac{120F}{P} = \frac{120 \cdot 50}{2} = 3000$ rpm

2) $slip = \frac{ns - nr}{ns} = \frac{3000 - 2808}{3000} = 0,064$

3) $pf = \frac{P_{in}}{\sqrt{3} \cdot V \cdot I}$

$= \frac{564,14}{\sqrt{3} \cdot 220 \cdot 1,709} = 0,86$

4) $\omega = \frac{4\pi f}{P}(1-s)$

$= \frac{4 \cdot 3,14 \cdot 50}{2}(1-0,064) = 293,904$ rad/sec

5) $Pout = T \cdot \omega$

$= 1,66 \cdot 293,904 = 487,88$ Watt

6) Efisiensi = $\frac{\text{Pout}}{\text{Pin}}$ x 100%

$= \frac{487,88}{564,14}$ x 100% = 86,4%

b. Beban 50%

1) ns = $\frac{120F}{P} = \frac{120 \,.\, 50}{2}$ = 3000 rpm

2) slip = $\frac{ns - nr}{ns}$

$= \frac{3000 - 2711}{3000} = 0,096$

3) pf = $\frac{P_{in}}{\sqrt{3} \,.\, V \,.\, I}$

$= \frac{780,66}{\sqrt{3} \,.\, 220 \,.\, 2,348} = 0,872$

4) $\omega = \frac{4\pi f}{P}(1 - s)$

$= \frac{4 \,.3,14 \,.50}{2}(1 - 0,096) = 283,856$ rad/sec

5) Pout = T . ω

$= 2,17 \,.\, 283,856 = 615,96$ Watt

6) Efisiensi = $\frac{\text{Pout}}{\text{Pin}}$ x 100%

$= \frac{615,96}{780,66}$ x 100% = 78,9%

c. Beban 75%

1) $ns = \frac{120F}{P} = \frac{120 \,.\, 50}{2} = 3000\,\text{rpm}$

2) $slip = \frac{ns - nr}{ns}$

$= \frac{3000 - 2615}{3000} = 0{,}128$

3) $pf = \frac{P_{in}}{\sqrt{3} \,.\, V \,.\, I}$

$= \frac{953{,}54}{\sqrt{3} \,.\, 220 \,.\, 2{,}902} = 0{,}862$

4) $\omega = \frac{4\pi f}{P}(1 - s)$

$= \frac{4 \,.\, 3{,}14 \,.\, 50}{2}(1 - 0{,}128) = 273{,}808$ Rad/sec

5) $Pout = T \,.\, \omega$

$= 2{,}52 \,.\, 273{,}808 = 689{,}99$ Watt

6) $\text{Efisiensi} = \frac{Pout}{Pin} \times 100\%$

$= \frac{689{,}99}{953{,}54} \times 100\% = 72{,}3\%$

d. Beban 100 %

1) $ns = \frac{120F}{P} = \frac{120 \,.\, 50}{2} = 3000\,\text{rpm}$

2) $slip = \frac{ns - nr}{ns}$

$$= \frac{3000 - 2582}{3000} = 0{,}157$$

3) $pf = \frac{P_{in}}{\sqrt{3} \cdot V \cdot I}$

$$= \frac{1081{,}92}{\sqrt{3} \cdot 220 \cdot 3{,}363} = 0{,}844$$

4) $\omega = \frac{4\pi f}{P}(1-s)$

$$= \frac{4 \cdot 3{,}14 \cdot 50}{2}(1-0{,}157) = 264{,}702 \text{ Rad/sec}$$

5) $Pout = T \cdot \omega$

$$= 2{,}84 \cdot 264{,}702 = 751{,}75 \text{ Watt}$$

6) $\text{Efisiensi} = \frac{Pout}{Pin} \times 100\%$

$$= \frac{751{,}75}{1081{,}92} \times 100\% = 69{,}5\%$$

Tabel 4-2

Hasil Perhitungan Pengujian Berbeban

Parameter \ % beban	25%	50%	75%	100%
Slip	0,064	0,096	0,128	0,157
Cos Φ	0,866	0,872	0,862	0,844
Efisiensi (%)	86,4	78,9	72,3	69,5
Ω (Rad/sec)	293,904	283,856	273,808	264,702

4.2 Data Masukan Untuk Program Algoritma Genetika dan *Evolutionarry Programming*

Data dibawah ini digunakan sebagai masukan pada program untuk menentukan parameter, yaitu Rs, Xs,Rr,Xr, dan Xm. Untuk mengefektifkan waktu, kedua metode dijalankan dalam waktu yang bersamaan dan dalam satu tampilan program.

Tabel 4-3

Input Data Untuk Program Algoritma Genetika dan *Evolutionarry Programming* (100% beban)

V	P_{in}	I	Slip
220	1081,92	3,363	0,157

Tabel 4-4

Input Data Untuk Program Algoritma Genetika dan *Evolutionarry Programming* (75% beban)

V	P_{in}	I	Slip
220	953,54	2,902	0,128

Tabel 4-5

Input Data Untuk Program Algoritma Genetika dan *Evolutionarry Programming* (50% beban)

V	P_{in}	I	Slip
220	780,66	2,348	0,096

Tabel 4-6

Input Data Untuk Program Algoritma Genetika *Evolutionarry Programming* (25% beban)

V	P_{in}	I_s	Slip
220	564,14	1,709	0,064

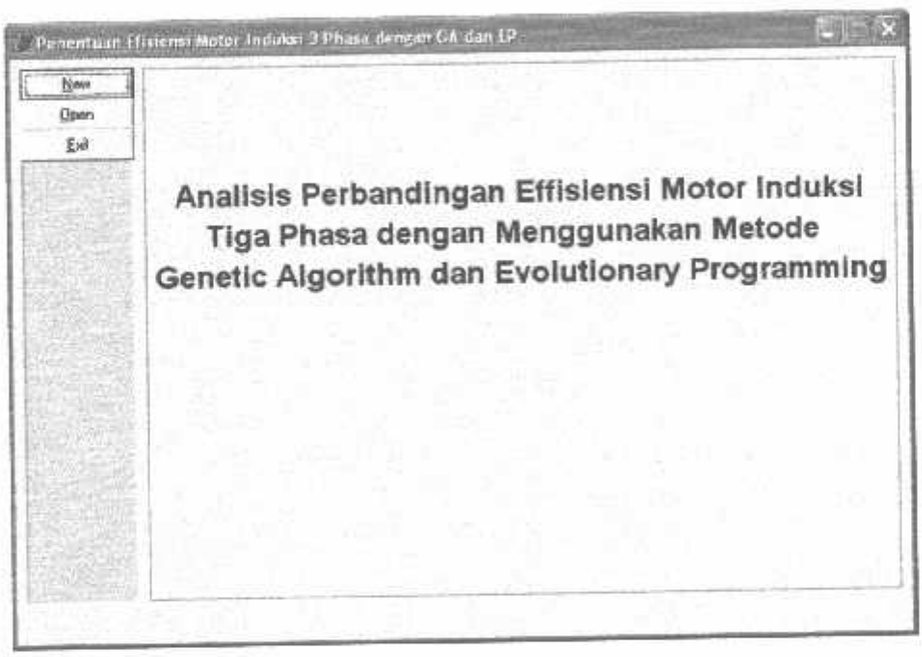
4.3 Program Komputer Metode Algoritma Genetika dan *Evolutionarry Programming*

Untuk pemecahan masalah pada metode Algoritma Genetika dan *Evolutionarry Programming* digunakan bantuan komputer. Program ini sangat berguna untuk mempercepat proses perhitungan yang membutuhkan ketelitian tinggi dan proses dalam pemilihan, seleksi dan statistik pada metode Algoritma Genetika dan *Evolutionarry Programming* serta tidak dimungkinkannya melakukan perhitungan secara manual. Program komputer ini menggunakan bahasa pemrograman Borland Delphi versi 7.

4.4 Tampilan Program

Program dalam skripsi ini dijalankan dengan menggunakan bahasa pemrograman Borland Delphi versi 7.0 dan diaplikasikan pada komputer berprosesor Pentium IV 1,8 MHz dengan memori 256 Mb. Mengenai jalannya program ikuti prosedur program sebagai berikut:

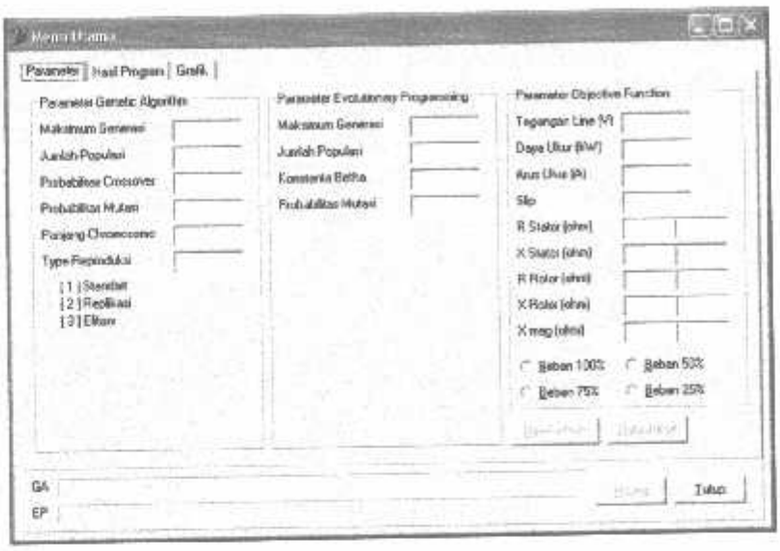
1. Tampilan utama dari program



Gambar 4-1

Tampilan Depan Program

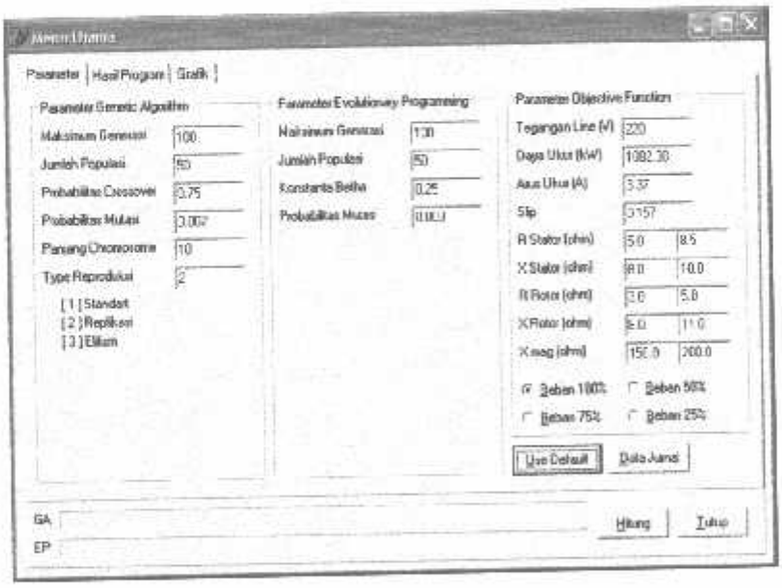
2. Tekan tombol Open File untuk membuka file yang sudah tersimpan



Gambar 4-2

Tampilan Inputan Data

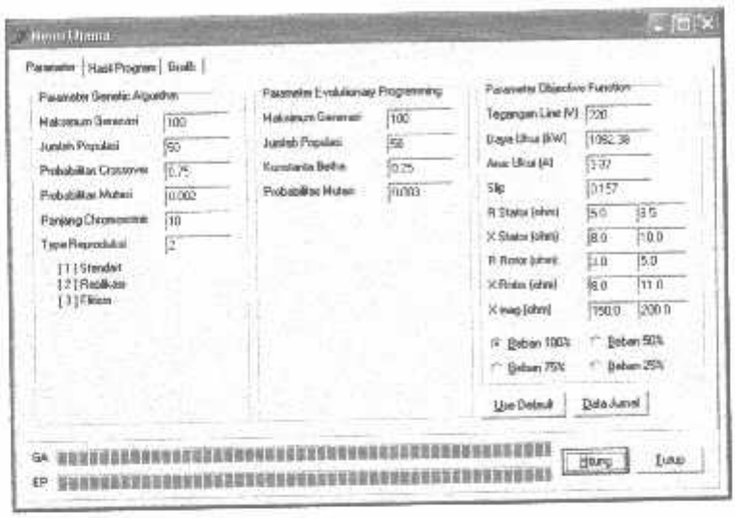
3. Tekan tombol Use Default untuk menampilkan data yang sudah tersimpan



Gambar 4-3

Tampilan Data Yang Sudah Tersimpan (Beban 100%)

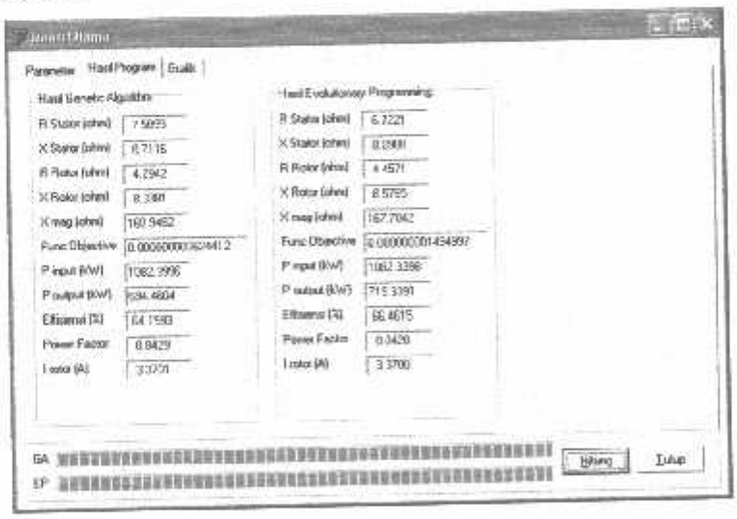
4. Kemudian pilih tombol Use default, dilanjutkan dengan tombol hitung sehingga tampil sebagai berikut



Gambar 4-4

Tampilan Running Program dengan Algoritma Genetika dan *Evolutionarry Programming* (Beban 100%)

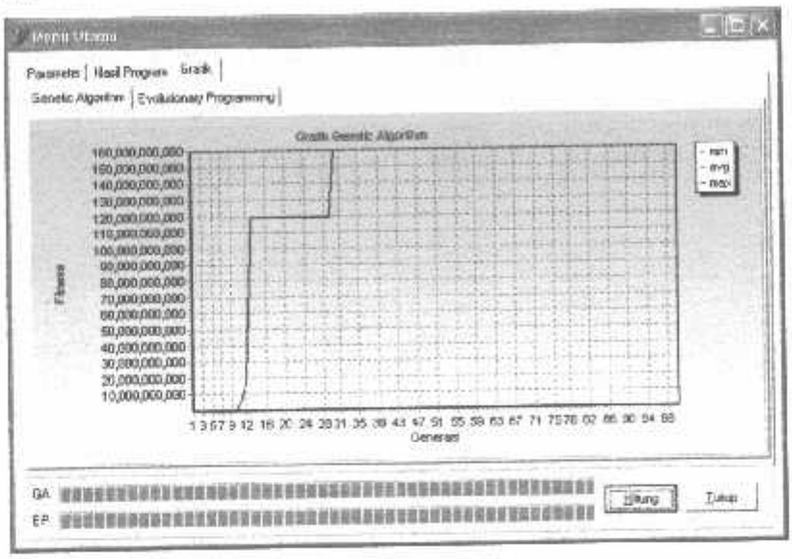
5. Jika ingin melihat hasil program tekan tombol hasil program, sehingga muncul tampilan berikut



Gambar 4-5

Tampilan Hasil Program (Beban 100%)

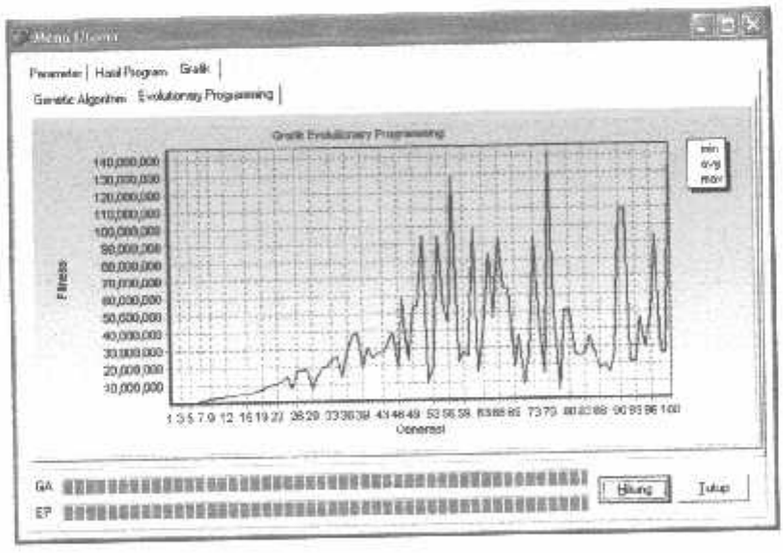
6. Untuk melihat Grafik, tekan tombol Grafik, sehingga muncul tampilan berikut



Gambar 4-6

Grafik Algoritma Genetika (Beban 100%)

7. Grafik pertama yang muncul (grafik pada no 6 di atas) adalah grafik Algoritma Genetika. Untuk melihat Grafik *Evolutionarry Programming* tekan tombol *Evolutionarry Programming*, seehingga tampilan yang muncul adalah sebagai berikut



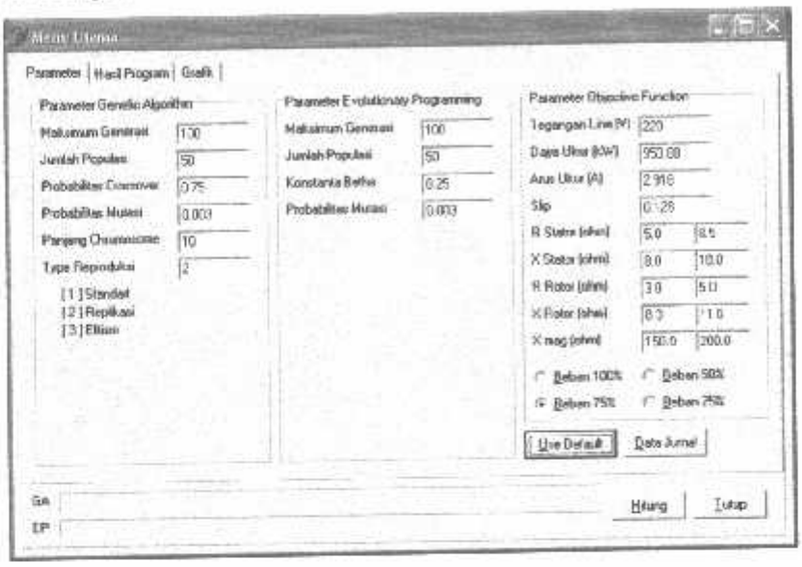
Gambar 4-7

Grafik *Evolutionarry Programming* (Beban 100%)

4.5 Tampilan Program Beban 75%, 50%, dan 25%

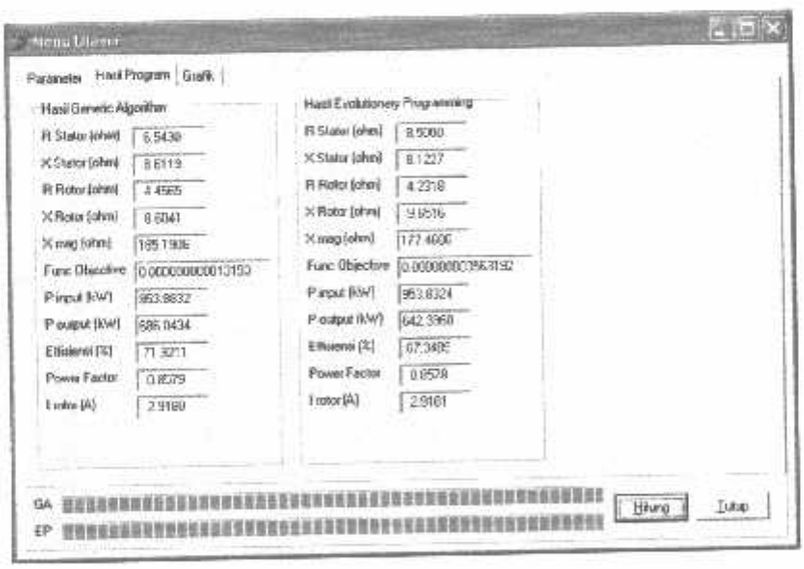
Tampilan program berikut ini bertujuan untuk menampilkan nilai efisiensi pada 25%, 50% dan 75% dari beban penuh.

1. Tampilan Program Beban 75%



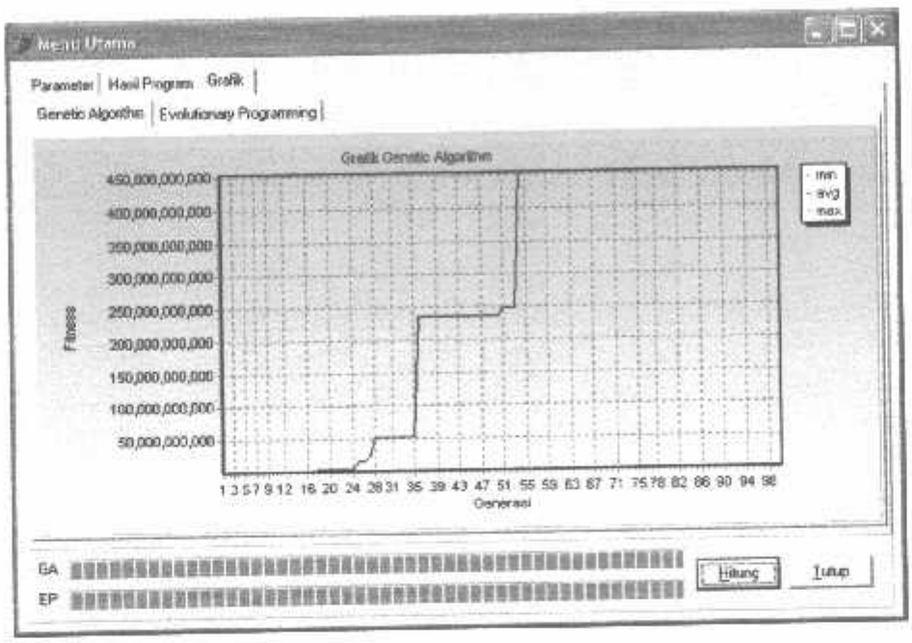
Gambar 4-8

Data Inputan GA dan EP pada 75% Beban



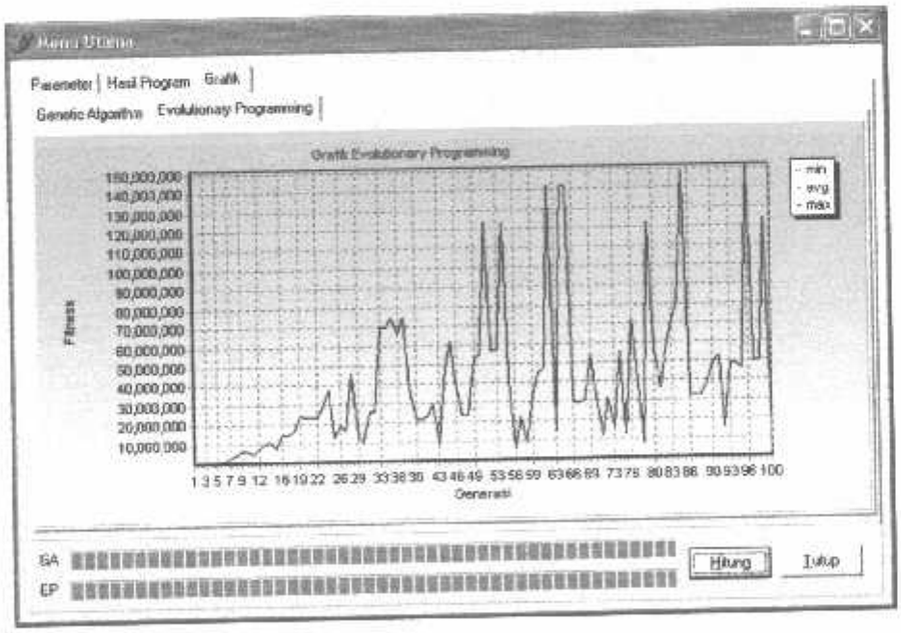
Gambar 4-9

Hasil Program GA dan EP Pada 75% Beban



Gambar 4-10

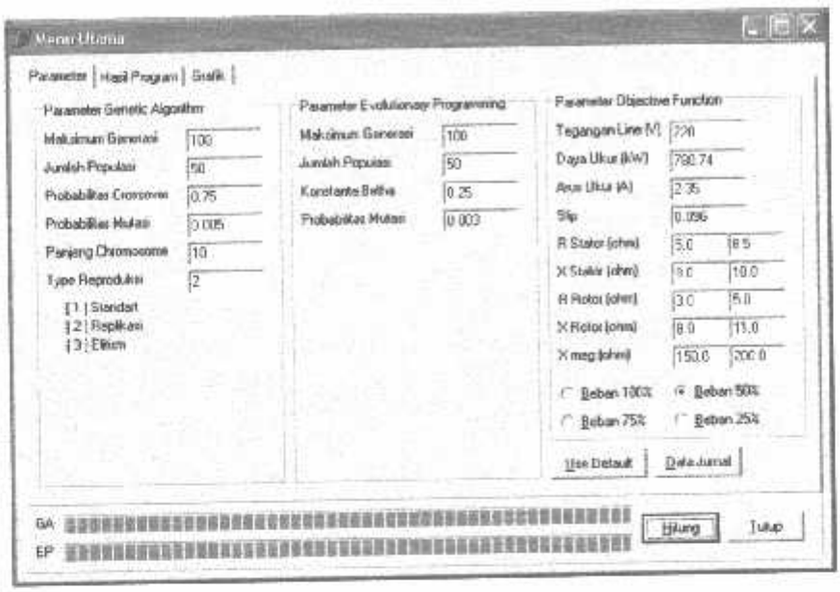
Tampilan Grafik GA pada 75% Beban



Gambar 4-11

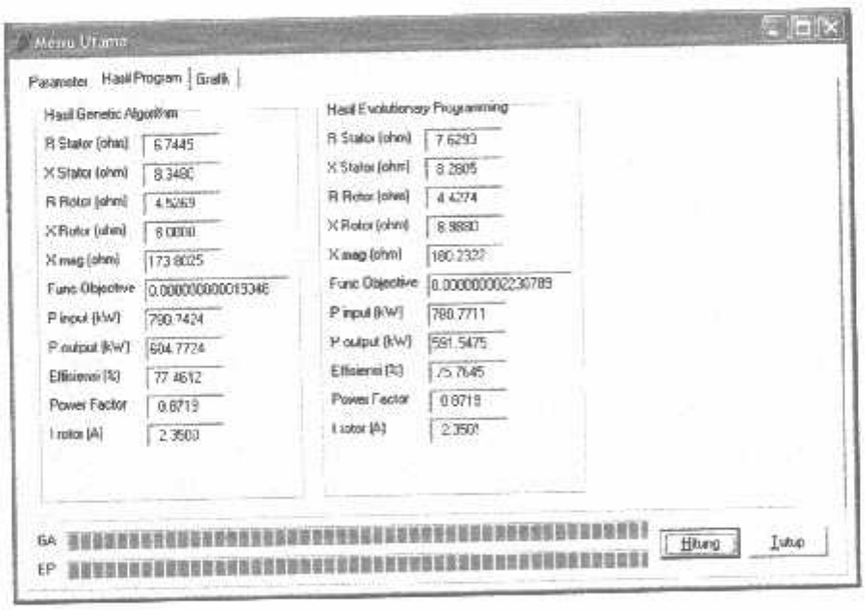
Tampilan Grafik EP pada 75% Beban

2. Tampilan Program Beban 50%



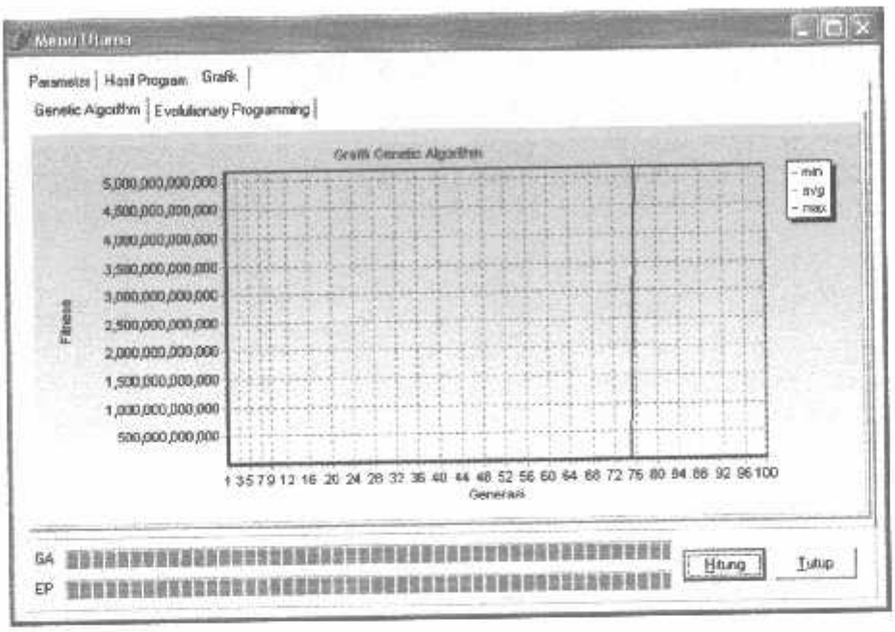
Gambar 4-12

Data Inputan GA dan EP pada 50% Beban



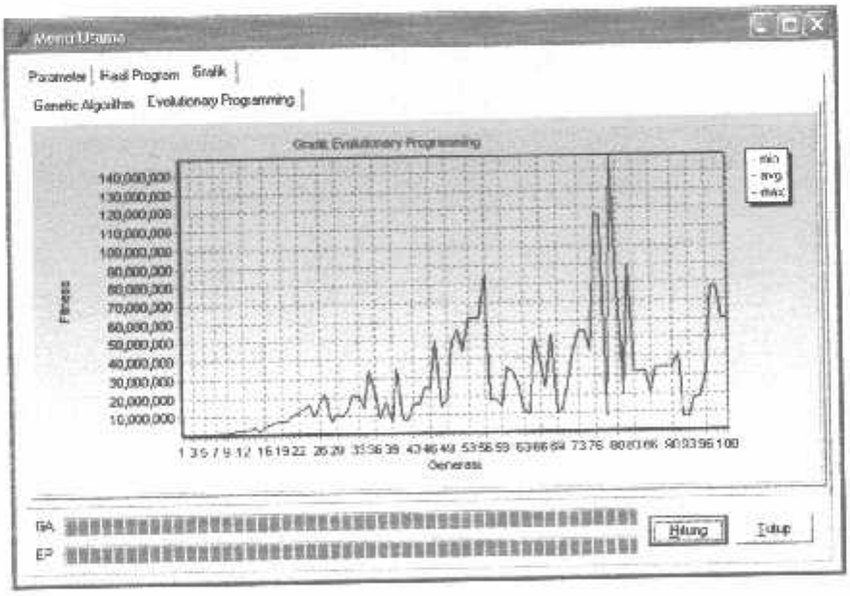
Gambar 4-13

Hasil Program GA dan EP Pada 50% Beban



Gambar 4-14

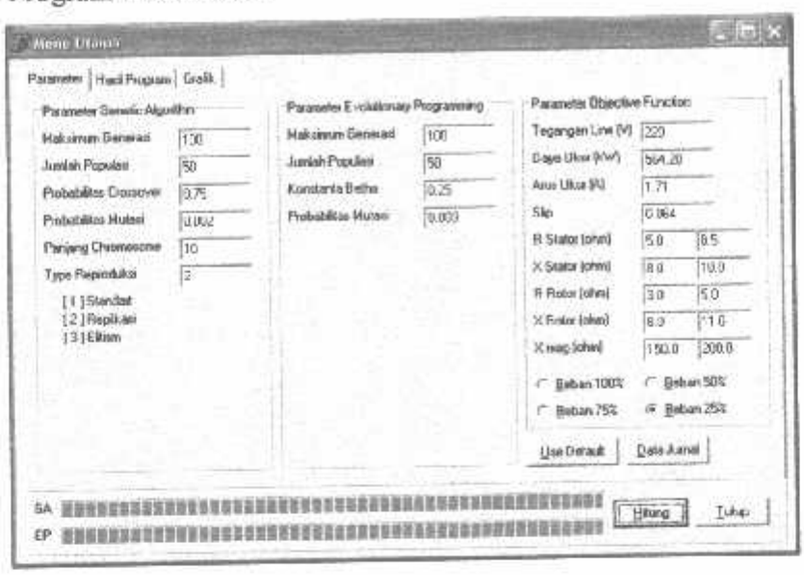
Tampilan Grafik GA pada 50% Beban



Gambar 4-15

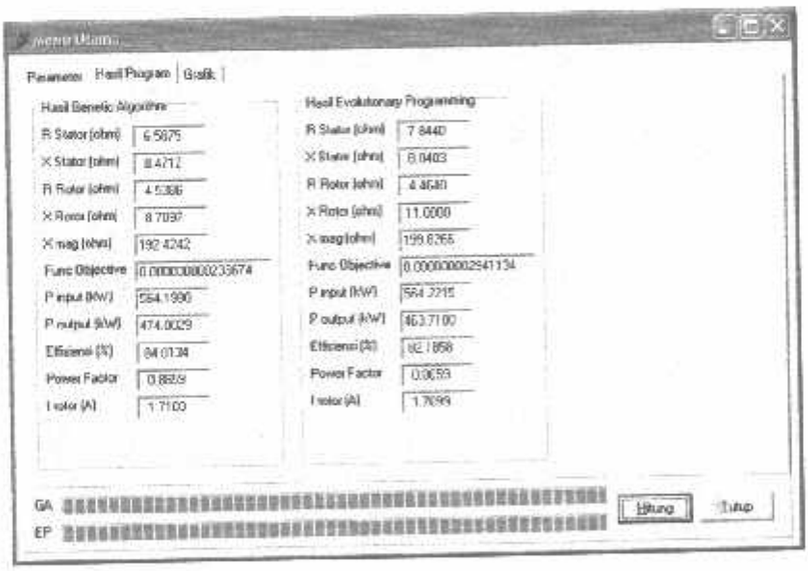
Tampilan Grafik EP pada 50% Beban

3. Tampilan Program Beban 25%



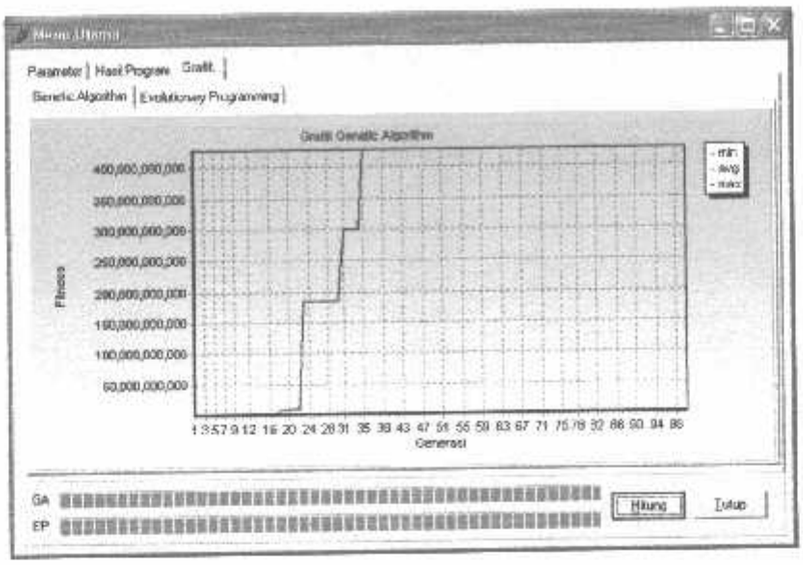
Gambar 4-16

Data Inputan GA dan EP pada 25% Beban



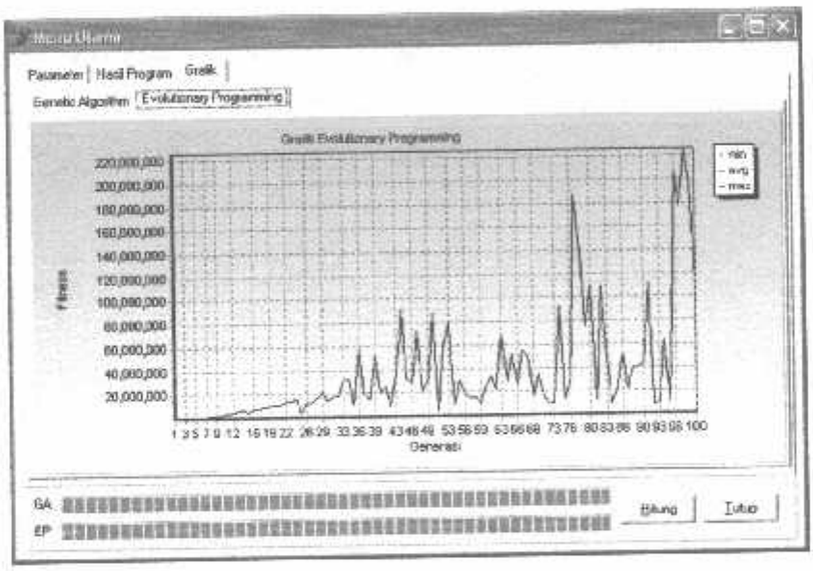
Gambar 4-17

Hasil Program GA dan EP pada 25% Beban



Gambar 4-18

Tampilan Grafik GA pada 25% Beban



Gambar 4-19

Tampilan Grafik EP pada 25% Beban

4.6 Hasil Perhitungan Dengan Metode Algoritma Genetika dan EP

Hasil perhitungan nilai parameter motor induksi dengan bantuan program dapat dilihat pada tabel dibawah ini:

Tabel 4.7

Hasil Perhitungan Dengan Metode Algoritma Genetika Dan *Evolutionarry Programming*

Beban / Param	25%			50%		
	GA	EP	Uji	GA	EP	Uji
Pinput (W)	564,1990	564,2215	564,20	780,7424	780,7711	780,74
I	1,71	1,71	1,71	2,35	2,35	2,35
Efisiensi %	84,0134	82,1858	86,5	77,46	75,76	78,9

Beban / Param	75%			100%		
	GA	EP	Uji	GA	EP	Uji
Pinput (W)	953,8832	953,8324	953,88	1082,3996	1082,3398	1082,38
I	2,918	2,9181	2,918	3,3701	3,37	3,37
Efisiensi %	71,92	67,34	72,3	64,1593	66,4515	64,7

BAB V
PENUTUP

5.1 Kesimpulan

Setelah melakukan pengujian pada motor berbeban yang menghasilkan data sebagaimana diperlihatkan pada tabel 4-7 pada bab 4 di atas, maka dapat disimpulkan :

1. Nilai perbandingan hasil perhitungan efisiensi dengan metode Algoritma Genetika dan *Evolutionary Programming* pada beban 100% adalah 64,759% dan 66,45%, pada beban 75% masing-masing efisiensinya sebesar 71,92% dan 67,34%, pada beban 50% sebesar 77,46% dan 75,76% dan yang terakhir pada beban 25% masing-masing efisiensinya sebesar 84,01% dan 82,18%.
2. Dalam menentukan parameter motor dan menghitung efisiensinya, maka metode Algorita Genetika lebih baik digunakan daripada metode *Evolutionary Programming* karena akurasinya lebih tinggi.

5.2 Saran

Pada penelitian selanjutnya, dalam menganalisa efisiensi motor induksi 3 phasa sebaiknya dikembbangkan lagi dengan mencoba metode alternatif yang relevan lainnya selain Algoritma Genetika, selain itu penelitian ini bisa juga dikembangkan dengan memasukkan parameter lain selain yang sudah ditentukan dan diujikan di atas, yang ada pada sirkuit ekivalen motor induksi, sehingga hasil yang didapat akan lebih sempurna.

DAFTAR PUSTAKA

[1] P.Pillay,V. Levin, P. Otaduy, and J. Kueck **"In-situ Induction Motor Efficiency Determination using The Genetic Algorithm"** IEEE Transaction on Energy Conversion, Vol, 13. pp. 326-333. Dec 1998

[2] A.E. Fitzgerald Charles Kingsley, Ir.Stephen D. Umar, Ir. Djoko Achyanto M.Sc.EE, **"Mesin-Mesin Listrik"**, Edisi ke empat, Erlangga,1992.

[3] Zuhal, **Mesin- Mesin Listrik**, PT. Gramedia Pustaka Utama Jakarta, 1994

[4] Drs. Yon Rijono , Dasar Teknik Tenaga Listrik, Edisi Pertama, Penerbit Andi, Jogjakarta.

[5] Mitsuo Gen, Runwei Cheng, **Genetic Algorithm and Ebgineering Design** , John Wiley&son, Inc.1994.

[6] David E. Goldberg, **Genetic Algorithms in Search, Optimization, and Machine Learning**, The University of Alabama

[7] Kwang. Y Lee, Senior Member, IEEE " **Optimal Reactive Power Planning Using Evolutionary Algorithm A comparative Study for Evolutionary Programming, Evolutionary Strategy, Genetic Algorithm, and Linear Programming** " IEEE Transactions on Power System, Vol 13, No 1, Feb 1998

[8] H.W. Kabisama, **"Electrical Power Engineering "**, Mc. Graw – Hill

[9] Vincen del Toro, **" Principles of Electrical Engineering"**Second Edition, Prentice- Hall Of India Private Limited New Delhi, 1984.

[10] George G. Karady, " **Induction Machine**", Topic 5, Lecture Notes EEE 360. *(www.eas.asu.-karady 360 stuff lectures)*

[11] P. C. Sen, " **Principles of Electrical Machines and Power Electronic** " Second edition, by John Wiley & Sons, Inc.1997

[12] T. Phumipak and Chat-uthai"***Effective Estimation of Induction Motor Field Efficiency Based on On-site Measurements***"Electrical Power Engineering Departement, Mahanakorn University of Technology, Bangkok 10530 Thailand.

[13] S.A. Nasar and L.E. Unnewehr, **"Electromechanics and Electric Machine "**. United States of America,1979

[14] Djoko Achyanto Msc, EE, **Mesin-mesin Listrik,** Edisi Keempat, Erlangga , Jakarta

LAMPIRAN

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ENERGI LISTRIK

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

1. **Nama** : RILYA ARTIKA DEVI
2. **NIM** : 03.12.051
3. **Jurusan** : Teknik Elektro S-1
4. **Konsentrasi** : Teknik Energi Listrik
5. **Judul Skripsi** : ANALISIS PERBANDINGAN PENENTUAN EFISIENSI MOTOR INDUKSI TIGA PHASA MENGGUNAKAN METODE ALGORITMA GENETIKA DAN *EVOLUTIONARY PROGRAMMING* DI LABORATORIUM KONVERSI ENERGI ELEKTRIK ITN MALANG

Dipertahankan dihadapan Majelis Penguji Skripsi Jenjang Strata Satu (S-1) pada :

Hari : Senin
Tanggal : 03 September 2007
Dengan Nilai : 78 (B+)

Panitia Ujian Skripsi

Ir. Mochtar Asroni, MSME
Ketua

Ir. F. Yudi Limpraptono, MT
Sekretaris

Anggota Penguji

Ir. H. Taufik Hidayat, MT
Penguji Pertama

Bambang Priyo Hartono, ST,MT
Penguji Kedua

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ENERGI LISTRIK

PERSETUJUAN PERBAIKAN SKRIPSI

Dari hasil ujian skripsi Jurusan Teknik Elektro jenjang strata satu (S-1) yang diselenggarakan pada :

Hari : Senin
Tanggal : 03 September 2007

Telah dilakukan perbaikan skripsi oleh :

1. Nama : RILYA ARTIKA DEVI
2. NIM : 03.12.051
3. Jurusan : Teknik Elektro
4. Konsentrasi : Teknik Energi Listrik S-1
5. Judul Skripsi : ANALISIS PERBANDINGAN PENENTUAN EFISIENSI MOTOR INDUKSI TIGA PHASA DENGAN MENGGUNAKAN METODE ALGORITMA GENETIKA DAN *EVOLUTIONARY PROGRAMMING* DI LABORATORIUM KONVERSI ENERGI ELEKTRIK ITN MALANG

Perbaikan meliputi :

No	Materi Perbaikan	Ket
1.	Sempurnakan analisa data BAB IV	

Anggota Penguji

Ir. H. Taufik Hidayat, MT
Penguji Pertama

Dosen Pembimbing

Ir. Widodo Pudji M, MT
Pembimbing Utama

Ir. Djojo Priatmono, MT
Pembimbing Kedua

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ENERGI LISTRIK

PERSETUJUAN PERBAIKAN SKRIPSI

Dari hasil ujian skripsi Jurusan Teknik Elektro jenjang strata satu (S-1) yang diselenggarakan pada :

Hari : Senin
Tanggal : 03 September 2007

Telah dilakukan perbaikan skripsi oleh :

1. Nama : RILYA ARTIKA DEVI
2. NIM : 03.12.051
3. Jurusan : Teknik Elektro
4. Konsentrasi : Teknik Energi Listrik S-1
5. Judul Skripsi : ANALISIS PERBANDINGAN PENENTUAN EFISIENSI MOTOR INDUKSI TIGA PHASA DENGAN MENGGUNAKAN METODE ALGORITMA GENETIKA DAN *EVOLUTIONARY PROGRAMMING* DI LABORATORIUM KONVERSI ENERGI ELEKTRIK ITN MALANG

Perbaikan meliputi :

No	Materi Perbaikan	Ket
1.	Abstraksi jadikan 3 alinea	
2.	Flowchart pindah ke Bab III	
3.	Halaman pindah ke kanan atas	

Anggota Penguji

Bambang Priyo Hartono. ST, MT.
Penguji Kedua

Dosen Pembimbing

Ir. Widodo Pudji M, MT
Pembimbing Utama

Ir. Djojo Priatmono, MT
Pembimbing Kedua

Pengujian Parameter Motor Induksi Tiga Phasa

Alat-alat yang digunakan

a. Motor induksi Tiga Phasa DE LORENZO / DL 1021
   Data papan (*Name-Plate*)

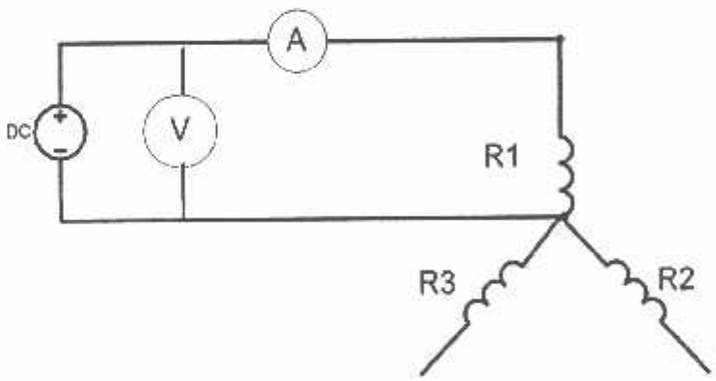
TEGANGAN	**: 220/380 (Δ/Y) VOLT**
ARUS	**: 4.3/2.5 (Δ/Y) AMPERE**
COS φ	**: 0.83**
FREKUENSI	**: 50 HZ**
DAYA	**: 1.1 KW**
PUTARAN	**: 2820 RPM**
KUTUP	**: 2 KUTUP**
KELAS ISOLASI	**: F**

- Generator DELORENZO/DL 1025
- Satu set alat ukur (watt meter, ampere meter, volt meter) DELORENZO/DL 1031
- AC/DC voltage regulator DELORENZO/DL 1013 M2
- Tachometer
- Current Brake
- Multimeter XIOLE DT9205A

Adapun model pengujian yang akan dilakukan adalah sebagai berikut:

1. Pengujian Arus Searah (*DC Test*)

Tujuan dari pengujian arus searah (*DC Test*) adalah untuk menentukan nilai resistansi stator. Diagram pengukuran ditunjukkan pada gambar 1.



Gambar 1 : Pengujian Arus Searah (*DC Test*)

Tabel 1
Data hasil pengujian arus searah

Vdc (volt)	I (Ampere)
8	1,47
7	1,28
6	1,11
5,1	0,96
4	0,77

Analisa pengujian arus searah:

$$R_s = R_{dc} = \frac{V_{dc}}{I_{dc}}$$

$$R_{dc1} = \frac{8}{1,47} = 5,44\ \Omega$$

$$R_{dc2} = \frac{7}{1,28} = 5,46\ \Omega$$

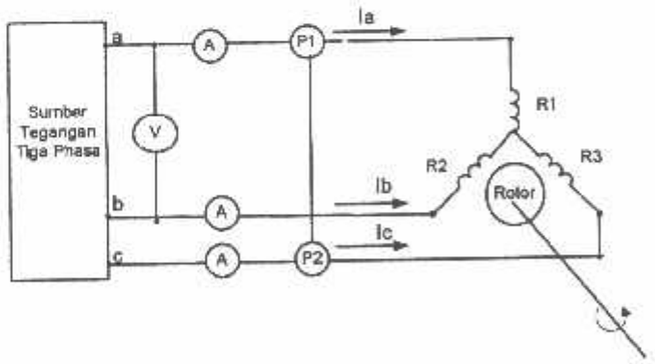
$$R_{dc3} = \frac{6}{1,11} = 5,40\ \Omega$$

$$R_{dc4} = \frac{5,1}{0,96} = 5,31\ \Omega$$

$$R_{dc5} = \frac{4}{0,77} = 5,19\ \Omega$$

$$Rs = \frac{5,44 + 5,46 + 5,40 + 5,31 + 5,19}{5} = 5,36\Omega$$

2. Pengujian Tanpa Beban (*No-Load Test*)



Gambar 2 Diagram Pengujian Tanpa Beban (*No-Load Test*)

Tabel 2
Data hasil pengujian beban nol

I (Ampere)			W 3Φ	V	F
A_1	A_2	A_3	(watt)	(volt)	(IIz)
0,53	0,61	0,53	90	220	50

Analisa pengujian beban nol:

$I_{rb} = \frac{I_{Ia} + I_{Ib} + I_{Ic}}{3} = \ldots A$

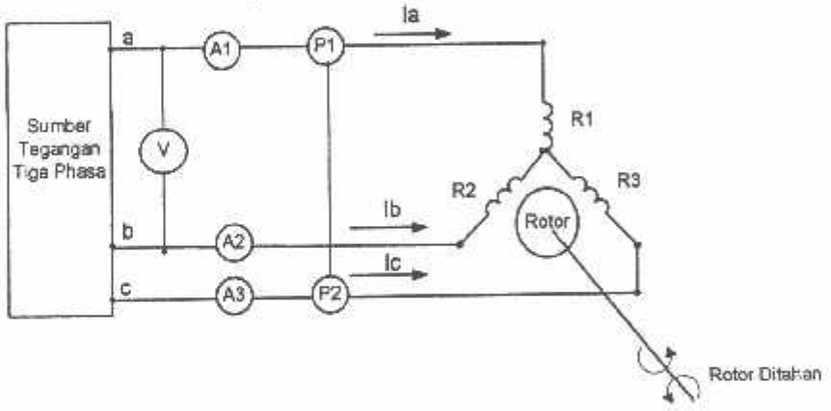
$I_{rb} = \frac{0{,}53 + 0{,}61 + 0{,}53}{3} = 0{,}55A$

$Z_{rb} = \frac{V_{rb}}{\sqrt{3} I_{rb}} = \frac{220}{\sqrt{3} x 0{,}56} = 231\Omega$

$R_{rb} = \frac{P3\Phi}{3 I_{rb}^2} = \frac{90}{3.(0{,}55^2)} = 99\Omega.$

$X_{rb} = \sqrt{Z_{rb}^2 - R_{rb}^2} = \sqrt{231^2 - 99^2} = 208{,}71\ \Omega$

3. Pengujian Rotor Tertahan (*Blocked Rotor Test*)



Tabel 3
Data hasil pengujian Rotor Tertahan

I (Ampere)			W 3Φ	V_{L-L}
A_1	A_2	A_3	(watt)	(volt)
1,88	1,93	1,.85	110	65

Analisa data pengujian Rotor Tertahan:

$I_{rt} = \frac{I_{Ia} + I_{Ib} + I_{Ic}}{3} = \frac{1{,}88 + 1{,}93 + 1{,}85}{3} = 1{,}9\ A$

$$Z_{n}=\frac{V}{\sqrt{3}I_{n}}=\frac{65}{\sqrt{3}x1,9}=19,75\ \Omega$$

$$R_{n}=\frac{P3\Phi}{3(I_{n})^{2}}=\frac{110}{3(1,9^{2})}=10,15\ \Omega$$

$$X_{n}=\sqrt{Z_{n}^{2}-R_{n}^{2}}=\sqrt{19,96^{2}-10,15^{2}}=17,18\ \Omega$$

$$X_{n}=X_{s}+X'_{r}$$

Motor induksi yang dipakai adalah Motor induksi dengan rotor sangkar tunggal kelas A, maka secara umum X_s dan X_r diasumsikan sama, sehingga :

$$X_{s}=X'_{r}=\frac{1}{2}X_{n}=\frac{1}{2}\,17,18=8,59\ \Omega$$

Besarnya reaktansi yang diukur pada terminal stator pada keadaan tanpa beban (X_{tb}) mendekati sama dengan X_s+X_m yang meruapakan reaktansi diri stator sehingga:

$$X_{ss}=X_{tb}=X_{s}+X_{m}$$

$$X_{m}=X_{tb}-X_{s}=.\ 208,71-8,59=200,12\ \Omega$$

Resistansi stator dapat dipandang sebagai harga DC-nya maka resistansi rotor dapat ditentukan sebagai berikut :

$$R=R_{n}-R_{s}$$
$$=10,15-5,36=4,79\ \Omega$$

$$X_{rr}=X'_{r}+Xm$$
$$=8,59+200,12=208,71$$

$$R'_{r}=R\left(\frac{X_{rr}}{X_{m}}\right)^{2}=4,79\left(\frac{208,71}{200,12}\right)^{2}=4,59\ \Omega$$

Tabel 4
Hasil Perhitungan Pengujian Parameter Motor Induksi Tiga Phasa

rs	rr	xls	xlr	xm
5,36	4,59	8,59	8,59	200,12

Jenis : DE LORENZO/DL 1022
Daya : 1,1 kW
Teg : 220/380 V
Arus : 4,3/2,5 A delta wye
Cos pi : 0.83;
Freq : 50 Hz
Putaran: 2820 rpm

Daya : 0.75 kW
Teg : 220/380 V
Arus : 2 A
Cos pi : 0.85
Freq : 50 Hz
Putaran: 1390 rpm

Data Hasil Pengukuran Berbeban

Beban	**Tegangan**	**Arus I**	**Daya input P_{in}**	**Torsi beban**	**nr**
%	**V (volt)**	**(amp)**	**(watt)**	**(Nm)**	**rpm**
25	220	1.709	564,14	1,66	2.808
50	220	2,348	780.66	2,17	2.711
75	220	2,902	953.54	2,52	2.615
100	220	3.363	1081,92	2,84	2.528

Listing program

```
unit uGenetic;

interface

uses uUtils,uRandom,uObjFunc,uMenuUtama;

type
  TIndividu=record
    chrom:bArr2;
    fitness:double;
  end;

  TPopulasi=array of TIndividu;

  TBGA=class
  private
    FMaxGen,FPopSize,FLength,FParam,FNewParent:integer;

FPcross,FPmutasi,FPflip,FMin1,FAvg1,FMax1,FSumFitness,FKa:double;
    FMin,FAvg,FMax:dArr1;
    FParent,FChild:TPopulasi;
    FBestIndi:TIndividu;
    FRandom:TRandomu;
    function getIndividu(const rIndi:TIndividu):TIndividu;
    function getBestParent:TIndividu;
    procedure SwapIndi(var rIndi1,rIndi2:TIndividu);
    procedure InitParent;
    procedure Statistik;
    function Seleksi:integer;
    function Mutasi(const rAllele:boolean):boolean;
    procedure doCrossover(const rParent1,rParent2:bArr2;
              var rChild1,rChild2:bArr2);
    procedure Generasi;
    procedure FindNewParent;
    procedure doHitung;
    function getBestChrom:bArr2;
    function getMin:dArr1;
    function getAvg:dArr1;
    function getMax:dArr1;
  public
    constructor Create(const rMaxGen,rPopSize,rLength,rParam,
                rNewParent:integer;
                const rPcross,rPmutasi,rPflip,rKa:double);
    destructor Destroy;override;
    property MaxGen:integer read FMaxGen write FMaxGen;
    property PopSize:integer read FPopSize write FPopSize;
    property Length:integer read FLength write FLength;
    property Param:integer read FParam write FParam;
    property Pcross:double read FPcross write FPcross;
    property Pmutasi:double read FPmutasi write FPmutasi;
    property Ka:double read FKa write FKa;
    property BestChrom:bArr2 read getBestChrom;
    property Min:dArr1 read getMin;
    property Avg:dArr1 read getAvg;
```

```
    property Max:dArr1 read getMax;
  end;

implementation

//constructor
constructor TBGA.Create(const rMaxGen,rPopSize,rLength,rParam,
            rNewParent:integer;
            const rPcross,rPmutasi,rPflip,rKa:double);
begin
  inherited Create;
  FMaxGen:=rMaxGen;
  FPopSize:=rPopSize;
  FLength:=rLength;
  FParam:=rParam;
  FNewParent:=rNewParent;
  FPcross:=rPcross;
  FPmutasi:=rPmutasi;
  FPflip:=rPflip;
  FKa:=rKa;
  FRandom:=TRandomu.Create;
end;

//destructor
destructor TBGA.Destroy;
begin
  try
    FRandom.Free;
  finally
    inherited Destroy;
  end;
end;

//data proccessing
function TBGA.getIndividu(const rIndi:TIndividu):TIndividu;
var i,j:integer;
begin
  SetLength(result.chrom,FParam,FLength);
  for i:=0 to FParam-1 do
  begin
    for j:=0 to FLength-1 do
    begin
      result.chrom[i,j]:=rIndi.chrom[i,j];
    end;
  end;
  result.fitness:=rIndi.fitness;
end;

function TBGA.getBestParent:TIndividu;
var i:integer;
begin
  result:=getIndividu(FParent[0]);
  for i:=1 to FPopSize-1 do
  begin
    if result.fitness<FParent[i].fitness then
    begin
```

```
        result:=getIndividu(FParent[i]);
      end;
    end;
  end;

  procedure TBGA.SwapIndi(var rIndi1,rIndi2:TIndividu);
  var tmp:TIndividu;
  begin
    tmp:=getIndividu(rIndi1);
    rIndi1:=getIndividu(rIndi2);
    rIndi2:=getIndividu(tmp);
  end;

  procedure TBGA.InitParent;
  var i,j,k:integer;
  begin
    SetLength(FMin,FMaxGen);
    SetLength(FAvg,FMaxGen);
    SetLength(FMax,FMaxGen);
    SetLength(FParent,FPopSize);
    SetLength(FChild,FPopSize);
    for i:=0 to FPopSize-1 do
    begin
      SetLength(FParent[i].chrom,FParam,FLength);
      SetLength(FChild[i].chrom,FParam,FLength);
      for j:=0 to FParam-1 do
      begin
        for k:=0 to FLength-1 do
        begin
          FParent[i].chrom[j,k]:=FRandom.NextBoolean(FPflip);
        end;
      end;
      FParent[i].fitness:=gObjFunc.getObjFunc(FParent[i].chrom);
    end;
    SetLength(FBestIndi.chrom,FParam,FLength);
  end;

  procedure TBGA.Statistik;
  var i:integer;
  begin
    FMin1:=FParent[0].fitness;
    FMax1:=FParent[0].fitness;
    FSumFitness:=FParent[0].fitness;
    for i:=1 to FPopSize-1 do
    begin
      if FMin1>FParent[i].fitness then
      begin
        FMin1:=FParent[i].fitness;
      end;
      if FMax1<FParent[i].fitness then
      begin
        FMax1:=FParent[i].fitness;
      end;
      FSumFitness:=FSumFitness+FParent[i].fitness;
    end;
    FAvg1:=FSumFitness/FPopSize;
```

```
end;

function TBGA.Seleksi:integer;
var rand,partsum:double;
    i:integer;
begin
  partsum:=0;
  i:=0;
  rand:=FRandom.NextDouble*FSumFitness;
  repeat
    i:=i+1;
    partsum:=partsum+FParent[i-1].fitness;
  until (partsum>rand) or (i=PopSize);
  Result:=i-1;
end;

function TBGA.Mutasi(const rAllele:boolean):boolean;
begin
  if FRandom.NextBoolean(FPMutasi)=true then
  begin
    result:=not rAllele;
  end
  else
  begin
    result:=rAllele;
  end;
end;

procedure TBGA.doCrossover(const rParent1,rParent2:bArr2;
          var rChild1,rChild2:bArr2);
var i,j,posX1,posX2,posY1,posY2:integer;
begin
  if FRandom.NextBoolean(FPCross)=true then
  begin
    posX1:=FRandom.NextInt(1,(FParam-1));
    repeat
      posX2:=FRandom.NextInt(1,(FParam-1));
    until PosX2<>posX1;
    if posX1>posX2 then
    begin
      Swap(posX1,posX2);
    end;
    posY1:=FRandom.NextInt(1,(FLength-1));
    repeat
      posY2:=FRandom.NextInt(1,(FLength-1));
    until PosY2<>posY1;
    if posY1>posY2 then
    begin
      Swap(posY1,posY2);
    end;
    //window1
    for i:=1 to PosX1 do
    begin
      for j:=1 to PosY1 do
      begin
        rChild1[i-1,j-1]:=Mutasi(rParent1[i-1,j-1]);
```

```
      rChild2[i-1,j-1]:=Mutasi(rParent2[i-1,j-1]);
    end;
end;
//window2
for i:=1 to PosX1 do
begin
    for j:=PosY1+1 to PosY2 do
    begin
      rChild1[i-1,j-1]:=Mutasi(rParent2[i-1,j-1]);
      rChild2[i-1,j-1]:=Mutasi(rParent1[i-1,j-1]);
    end;
end;
//window 3
for i:=1 to PosX1 do
begin
    for j:=PosY2+1 to FLength do
    begin
      rChild1[i-1,j-1]:=Mutasi(rParent1[i-1,j-1]);
      rChild2[i-1,j-1]:=Mutasi(rParent2[i-1,j-1]);
    end;
end;
//window4
for i:=PosX1+1 to PosX2 do
begin
    for j:=1 to PosY1 do
    begin
      rChild1[i-1,j-1]:=Mutasi(rParent2[i-1,j-1]);
      rChild2[i-1,j-1]:=Mutasi(rParent1[i-1,j-1]);
    end;
end;
//window5
for i:=PosX1+1 to PosX2 do
begin
    for j:=PosY1+1 to PosY2 do
    begin
      rChild1[i-1,j-1]:=Mutasi(rParent1[i-1,j-1]);
      rChild2[i-1,j-1]:=Mutasi(rParent2[i-1,j-1]);
    end;
end;
//window 6
for i:=PosX1+1 to PosX2 do
begin
    for j:=PosY2+1 to FLength do
    begin
      rChild1[i-1,j-1]:=Mutasi(rParent2[i-1,j-1]);
      rChild2[i-1,j-1]:=Mutasi(rParent1[i-1,j-1]);
    end;
end;
//window7
for i:=PosX2+1 to FParam do
begin
    for j:=1 to PosY1 do
    begin
      rChild1[i-1,j-1]:=Mutasi(rParent1[i-1,j-1]);
      rChild2[i-1,j-1]:=Mutasi(rParent2[i-1,j-1]);
    end;
```

```
      end;
      //window8
      for i:=PosX2+1 to FParam do
      begin
        for j:=PosY1+1 to PosY2 do
        begin
          rChild1[i-1,j-1]:=Mutasi(rParent2[i-1,j-1]);
          rChild2[i-1,j-1]:=Mutasi(rParent1[i-1,j-1]);
        end;
      end;
      //window 9
      for i:=PosX2+1 to FParan do
      begin
        for j:=PosY2+1 to FLength do
        begin
          rChild1[i-1,j-1]:=Mutasi(rParent1[i-1,j-1]);
          rChild2[i-1,j-1]:=Mutasi(rParent2[i-1,j-1]);
        end;
      end;
    end
    else
    begin
      for i:=1 to FParam do
      begin
        for j:=1 to FLength do
        begin
          rChild1[i-1,j-1]:=Mutasi(rParent2[i-1,j-1]);
          rChild2[i-1,j-1]:=Mutasi(rParent1[i-1,j-1]);
        end;
      end;
    end;
  end;

procedure TBGA.Generasi;
var i,mate1,mate2:integer;
begin
  i:=0;
  repeat
    mate1:=Seleksi;
    mate2:=Seleksi;
    doCrossover(FParent[mate1].chrom,FParent[mate2].chrom,
                FChild[i].chrom,FChild[i+1].chrom);
    FChild[i].fitness:=gObjFunc.getObjFunc(FChild[i].chrom);
    FChild[i+1].fitness:=gObjFunc.getObjFunc(FChild[i+1].chrom);
    i:=i+2;
  until i>=PopSize;
end;

procedure TBGA.FindNewParent;
var i,j,mate:integer;
    tmpPop:TPopulasi;
begin
  if FNewParent=1 then
  begin
    for i:=0 to PopSize-1 do
    begin
```

```
        FParent[i]:=getIndividu(FChild[i]);
      end;
    end
    else if FNewParent=2 then
    begin
      SetLength(tmpPop,PopSize);
      for i:=0 to PopSize-1 do
      begin
        repeat
          mate:=FRandom.NextInt(0,PopSize-1);
        until mate<>i;
        if FChild[i].fitness>FParent[mate].fitness then
        begin
          tmpPop[i]:=getIndividu(FChild[i]);
        end
        else
        begin
          tmpPop[i]:=getIndividu(FParent[mate]);
        end;
      end;
      for i:=0 to PopSize-1 do
      begin
        FParent[i]:=getIndividu(tmpPop[i]);
      end;
    end
    else
    begin
      SetLength(tmpPop,2*PopSize);
      for i:=0 to PopSize-1 do
      begin
        tmpPop[i]:=getIndividu(FParent[i]);
        tmpPop[i+PopSize]:=getIndividu(FChild[i]);
      end;
      for i:=0 to 2*PopSize-2 do
      begin
        for j:=i to 2*PopSize-1 do
        begin
          if tmpPop[i].fitness<tmpPop[j].fitness then
          begin
            SwapIndi(tmpPop[i],tmpPop[j]);
          end;
        end;
      end;
      for i:=0 to PopSize-1 do
      begin
        FParent[i]:=getIndividu(tmpPop[i]);
      end;
    end;
  end;

procedure TBGA.doHitung;
var i:integer;
    tmpIndi:TIndividu;
begin
  InitParent;
  Statistik;
```

```
    FBestIndi:=getBestParent;
    i:=0;
    repeat
      Generasi;
      FindNewParent;
      Statistik;
      tmpIndi:=getBestParent;
      if FBestIndi.fitness<tmpIndi.fitness then
      begin
        FBestIndi:=getIndividu(tmpIndi);
      end;
      FMin[i]:=FMin1;
      FAvg[i]:=FAvg1;
      FMax[i]:=FMax1;
      frmMenuUtama.pbGA.StepBy(1);
      inc(i);
    until i>=MaxGen;
  end;

  function TBGA.getBestChrom:bArr2;
  var i,j:integer;
  begin
    doHitung;
    SetLength(result,FParam,Length);
    for i:=0 to FParam-1 do
    begin
      for j:=0 to Length-1 do
      begin
        result[i,j]:=FBestIndi.chrom[i,j];
      end;
    end;
  end;

  function TBGA.getMin:dArr1;
  var i:integer;
  begin
    SetLength(result,FMaxGen);
    for i:=0 to FMaxGen-1 do
    begin
      result[i]:=FMin[i];
    end;
  end;

  function TBGA.getAvg:dArr1;
  var i:integer;
  begin
    SetLength(result,FMaxGen);
    for i:=0 to FMaxGen-1 do
    begin
      result[i]:=FAvg[i];
    end;
  end;

  function TBGA.getMax:dArr1;
  var i:integer;
  begin
```

```
  SetLength(result,FMaxGen);
  for i:=0 to FMaxGen-1 do
  begin
    result[i]:=FMax[i];
  end;
end;

end.


unit uEvoPro;

interface

uses uUtils,uRandom,uObjFunc,uMenuUtama;

type
  TIndividu1=record
    chrom:dArr1;
    fitness:double;
  end;

  TPopulasi1=array of TIndividu1;

  TIndividu2=record
    chrom:dArr2;
    fitness:double;
  end;

  TPopulasi2=array of TIndividu2;

  TEvoPro=class
  private
    FMaxGen,FPopSize,FLength:integer;
    FBetha:double;
    FRandom:TRandomu;
    function getMin:dArr1;
    function getAvg:dArr1;
    function getMax:dArr1;
  protected
    FMin,FMax,FAvg:dArr1;
  public
    constructor Create(const rMaxGen,rPopSize,rLength:integer;
                const rBetha:double);
    destructor Destroy;override;
    property MaxGen:integer read FMaxGen write FMaxGen;
    property PopSize:integer read FPopSize write FPopSize;
    property Length:integer read FLength write FLength;
    property Betha:double read FBetha write FBetha;
    property Min:dArr1 read getMin;
    property Avg:dArr1 read getAvg;
    property Max:dArr1 read getMax;
  end;

  TEvoPro1=class(TEvoPro)
```

```
  private
    FBatas:TBatasArr1;
    FBestIndi:TIndividu1;
    FParent,FChild:TPopulasi1;
    FBestChrom:dArr1;
    FMin1,FAvg1,FMax1,FKa,FPmutasi:double;
    function getBatas:TBatasArr1;
    procedure setBatas(const rBatas:TBatasArr1);
    function getIndividu(const rIndi:TIndividu1):TIndividu1;
    procedure SwapIndi(var rIndi1,rIndi2:TIndividu1);
    function FindFitnessMax:double;
    function FindIndiMax:TIndividu1;
    procedure InitParent;
    procedure Statistik;
    procedure Generasi;
    procedure Kompetisi;
    procedure doHitung;
    function getBestChrom:dArr1;
  public
    constructor Create(const rMaxGen,rPopSize,rLength:integer;
                const rBetha,rKa,rPmutasi:double;
                const rBatas:TBatasArr1;
                const rBestChrom:dArr1);
    property Batas:TBatasArr1 read getBatas write setBatas;
    property Ka:double read FKa write FKa;
    property BestChrom:dArr1 read getBestChrom;
  end;

implementation

{ TEvoPro }
//constructor
constructor TEvoPro.Create(const rMaxGen,rPopSize,rLength:integer;
            const rBetha:double);
begin
  inherited Create;
  FMaxGen:=rMaxGen;
  FPopSize:=rPopSize;
  FLength:=rLength;
  FBetha:=rBetha;
  FRandom:=TRandomu.Create;
end;

//destructor
destructor TEvoPro.Destroy;
begin
  try
    FRandom.Free;
  finally
    inherited Destroy;
  end;
end;

//data accessing
function TEvoPro.getMin:dArr1;
var i:integer;
```

```
begin
  SetLength(result,FMaxGen);
  for i:=0 to FMaxGen-1 do
  begin
    result[i]:=FMin[i];
  end;
end;

function TEvoPro.getAvg:dArr1;
var i:integer;
begin
  SetLength(result,FMaxGen);
  for i:=0 to FMaxGen-1 do
  begin
    result[i]:=FAvg[i];
  end;
end;

function TEvoPro.getMax:dArr1;
var i:integer;
begin
  SetLength(result,FMaxGen);
  for i:=0 to FMaxGen-1 do
  begin
    result[i]:=FMax[i];
  end;
end;

{ TEvoPro1 }

//constructor
constructor TEvoPro1.Create(const
rMaxGen,rPopSize,rLength:integer;
            const rBetha,rKa,rPmutasi:double;
            const rBatas:TBatasArr1;
            const rBestChrom:dArr1);
var i:integer;
begin
  inherited Create(rMaxGen,rPopSize,rLength,rBetha);
  FKa:=rKa;
  FPmutasi:=rPmutasi;
  SetLength(FBatas,Length);
  SetLength(FBestChrom,Length);
  for i:=0 to Length-1 do
  begin
    FBatas[i].min:=rBatas[i].min;
    FBatas[i].max:=rBatas[i].max;
    FBestChrom[i]:=rBestChrom[i];
  end;
end;

function TEvoPro1.getBatas:TBatasArr1;
var i:integer;
begin
  SetLength(result,Length);
  for i:=0 to Length-1 do
```

```
  begin
    result[i].min:=FBatas[i].min;
    result[i].max:=FBatas[i].max;
  end;
end;

procedure TEvoPro1.setBatas(const rBatas:TBatasArr1);
var i:integer;
begin
  SetLength(FBatas,Length);
  for i:=0 to Length-1 do
  begin
    FBatas[i].min:=rBatas[i].min;
    FBatas[i].max:=rBatas[i].max;
  end;
end;

//data proccessing
function TEvoPro1.getIndividu(const rIndi:TIndividu1):TIndividu1;
var i:integer;
begin
  SetLength(result.chrom,Length);
  for i:=0 to Length-1 do
  begin
    result.chrom[i]:=rIndi.chrom[i];
  end;
  result.fitness:=rIndi.fitness;
end;

procedure TEvoPro1.SwapIndi(var rIndi1,rIndi2:TIndividu1);
var tmp:TIndividu1;
begin
  tmp:=getIndividu(rIndi1);
  rIndi1:=getIndividu(rIndi2);
  rIndi2:=getIndividu(tmp);
end;

function TEvoPro1.FindFitnessMax:double;
var i:integer;
begin
  result:=FParent[0].fitness;
  for i:=0 to PopSize-1 do
  begin
    if result<FParent[i].fitness then
    begin
      result:=FParent[i].fitness;
    end;
  end;
end;

function TEvoPro1.FindIndiMax:TIndividu1;
var i:integer;
begin
  result:=getIndividu(FParent[0]);
  for i:=1 to PopSize-1 do
  begin
```

```
    if result.fitness<FParent[i].fitness then
    begin
      result:=getIndividu(FParent[i]);
    end;
  end;
end;

procedure TEvoPro1.InitParent;
var i,j:integer;
begin
  SetLength(FParent,FPopSize);
  SetLength(FChild,FPopSize);
  SetLength(FMin,FMaxGen);
  SetLength(FAvg,FMaxGen);
  SetLength(FMax,FMaxGen);
  for i:=0 to PopSize-1 do
  begin
    SetLength(FParent[i].chrom,Length);
    SetLength(FChild[i].chrom,Length);
    {if i=0 then
    begin
      for j:=0 to Length-1 do
      begin
        FParent[i].chrom[j]:=FBestChrom[j];
      end;
    end
    else
    begin)
      for j:=0 to Length-1 do
      begin

FParent[i].chrom[j]:=FRandom.NextDouble(FBatas[j].min,FBatas[j].ma
x);
      end;
    //end;
    FParent[i].fitness:=gObjFunc.getObjFunc(FParent[i].chrom);
  end;
end;

procedure TEvoPro1.Statistik;
var i:integer;
    sumFitness:double;
begin
  sumFitness:=FParent[0].fitness;
  FMin1:=FParent[0].fitness;
  FMax1:=FParent[0].fitness;
  for i:=1 to PopSize-1 do
  begin
    sumFitness:=sumFitness+FParent[i].fitness;
    if FMin1>FParent[i].fitness then
    begin
      FMin1:=FParent[i].fitness;
    end;
    if FMax1<FParent[i].fitness then
    begin
      FMax1:=FParent[i].fitness;
```

```
    end;
  end;
  FAvg1:=sumFitness/FPopSize;
end;

procedure TEvoProl.Generasi;
var i,j:integer;
    Fmax,fitEP,fitES:double;
    tho:double;
    chromEP,chromES:dArr1;
begin
  Fmax:=FindFitnessMax;
  SetLength(chromEP,Length);
  SetLength(chromES,Length);
  for i:=0 to PopSize-1 do
  begin
    for j:=0 to Length-1 do
    begin
      tho:=(FBatas[j].max-FBatas[j].min)*((Fmax-
FParent[i].fitness)/
          Fmax+Betha);

chromEP[j]:=FParent[i].chrom[j]+FRandom.NextGaussian(0,sqr(tho));
      if chromEP[j]>FBatas[j].max then
      begin
        chromEP[j]:=FBatas[j].max;
      end;
      if chromEP[j]<FBatas[j].min then
      begin
        chromEP[j]:=FBatas[j].min;
      end;
    end;
    fitEP:=gObjFunc.getObjFunc(chromEP);
    for j:=0 to Length-1 do
    begin

chromES[j]:=FParent[i].chrom[j]+FRandom.NextGaussian(0,FPmutasi);
      if chromES[j]>FBatas[j].max then
      begin
        chromES[j]:=FBatas[j].max;
      end;
      if chromES[j]<FBatas[j].min then
      begin
        chromES[j]:=FBatas[j].min;
      end;
    end;
    fitES:=gObjFunc.getObjFunc(chromES);
    if fitEP>fitES then
    begin
      for j:=0 to Length-1 do
      begin
        FChild[i].chrom[j]:=chromEP[j];
      end;
      FChild[i].fitness:=fitEP;
    end
    else
```

```
    begin
      for j:=0 to Length-1 do
      begin
        FChild[i].chrom[j]:=chromES[j];
      end;
      FChild[i].fitness:=fitES;
    end;
  end;
end;

procedure TEvoProl.Kompetisi;
var i,j,Ntmp,sa:integer;
    tmp:TPopulasil;
    sort:iArrl;
begin
  Ntmp:=2*PopSize;
  SetLength(tmp,Ntmp);
  for i:=0 to PopSize-1 do
  begin
    tmp[i]:=getIndividu(FParent[i]);
    tmp[PopSize+i]:=getIndividu(FChild[i]);
  end;
  SetLength(sort,Ntmp);
  for i:=0 to Ntmp-1 do
  begin
    sort[i]:=0;
  end;
  for i:=0 to Ntmp-1 do
  begin
    for j:=0 to Ntmp-2 do
    begin
      repeat
        sa:=FRandom.NextInt(0,(Ntmp-1));
      until sa<>i;
      if tmp[i].fitness>tmp[sa].fitness then
      begin
        inc(sort[i]);
      end;
    end;
  end;
  for i:=0 to Ntmp-2 do
  begin
    for j:=i to Ntmp-1 do
    begin
      if sort[i]<sort[j] then
      begin
        Swap(sort[i],sort[j]);
        SwapIndi(tmp[i],tmp[j]);
      end;
    end;
  end;
  for i:=0 to PopSize-1 do
  begin
    FParent[i]:=getIndividu(tmp[i]);
  end;
end;
```

```
procedure TEvoPro1.doHitung;
var gen:integer;
    TempIndi:TIndividu1;
begin
  InitParent;
  Statistik;
  FBestIndi:=FindIndiMax;
  gen:=1;
  repeat
    Generasi;
    Kompetisi;
    Statistik;
    TempIndi:=FindIndiMax;
    if FBestIndi.fitness<TempIndi.fitness then
    begin
      FBestIndi:=GetIndividu(TempIndi);
    end;
    FMin[gen-1]:=FMin1;
    FAvg[gen-1]:=FAvg1;
    FMax[gen-1]:=FMax1;
    frmMenuUtama.pbEP.StepBy(1);
    inc(gen);
  until (gen>MaxGen);
end;

//data output
function TEvoPro1.getBestChrom:dArr1;
var i:integer;
begin
  doHitung;
  SetLength(result,Length);
  for i:=0 to Length-1 do
  begin
    result[i]:=FBestIndi.chrom[i];
  end;
end;

end.


unit uObjFunc;

interface

uses uUtils,uComplex;

type
  TObjFunc=class
  private
    FRs,FXs,FRr,FXr,FXm:TBatas;
    Fs,FPmea,FImea,FV:double;
  public
    constructor Create(const rRs,rXs,rRr,rXr,rXm:TBatas;
                       const rs,rPmea,rImea,rV:double);
    function getObjFunc(const rChrom:bArr2):double;overload;
```

```
    function getObjFunc(const rChrom:dArr1):double;overload;
    procedure getObjFuncAkhir(const rChrom:bArr2;
              var rRs,rXs,rRr,rXr,rXm,rPinp,rFitness,rPout,rEff,
              rpf,rAbsI1:double);overload;
    procedure getObjFuncAkhir(const rChrom:dArr1;
              var rRs,rXs,rRr,rXr,rXm,rPinp,rFitness,rPout,rEff,
              rpf,rAbsI1:double);overload;
  end;

var gObjFunc:TObjFunc;

implementation

//constructor
constructor TObjFunc.Create(const rRs,rXs,rRr,rXr,rXm:TBatas;
                            const rs,rPmea,rImea,rV:double);
begin
  inherited Create;
  FRs.min:=rRs.min;FRs.max:=rRs.max;
  FXs.min:=rXs.min;FXs.max:=rXs.max;
  FRr.min:=rRr.min;FRr.max:=rRr.max;
  FXr.min:=rXr.min;FXr.max:=rXr.max;
  FXm.min:=rXm.min;FXm.max:=rXm.max;
  Fs:=rs;
  FPmea:=rPmea;
  FImea:=rImea;
  FV:=rV/sqrt(3);
end;

//data proccessing

//data output
function TObjFunc.getObjFunc(const rChrom:bArr2):double;
var data:dArr1;
    Rs,Xs,Rr,Xr,Xm,AbsI1,pf,P1,FObj:double;
    Zs,Zr,Zm,Ze,Zp,Zt,I1,Vt:TCmplx;
begin
  data:=DecodeBinToFloat2Base0(rChrom);
  Rs:=GetBatasToReal(data[0],FRs.min,FRs.max);
  Xs:=GetBatasToReal(data[1],FXs.min,FXs.max);
  Rr:=GetBatasToReal(data[2],FRr.min,FRr.max);
  Xr:=GetBatasToReal(data[3],FXr.min,FXr.max);
  Xm:=GetBatasToReal(data[4],FXm.min,FXm.max);
  Vt:=Cmplx(FV,0);
  Zs:=Cmplx(Rs,Xs);
  Zr:=Cmplx(Rr/Fs,Xr);
  Zm:=Cmplx(0,Xm);
  Ze:=Multiply(Zr,Zm);
  Zt:=Add(Zr,Zm);
  Zp:=Divide(Ze,Zt);
  Zt:=Add(Zs,Zp);
  I1:=Divide(Vt,Zt);
  AbsI1:=getAbs(I1);
  pf:=cos(getAngleRad(I1));
  P1:=3*FV*absI1*pf;
  FObj:=sqr(abs(absI1/FImea-1))+sqr(abs(P1/FPmea-1));
```

```
  result:=100/FObj;
end;

function TObjFunc.getObjFunc(const rChrom:dArr1):double;
var Rs,Xs,Rr,Xr,Xm,AbsI1,pf,P1,FObj:double;
    Zs,Zr,Zm,Ze,Zp,Zt,I1,Vt:TCmplx;
begin
  Rs:=rChrom[0];
  Xs:=rChrom[1];
  Rr:=rChrom[2];
  Xr:=rChrom[3];
  Xm:=rChrom[4];
  Vt:=Cmplx(FV,0);
  Zs:=Cmplx(Rs,Xs);
  Zr:=Cmplx(Rr/Fs,Xr);
  Zm:=Cmplx(0,Xm);
  Ze:=Multiply(Zr,Zm);
  Zt:=Add(Zr,Zm);
  Zp:=Divide(Ze,Zt);
  Zt:=Add(Zs,Zp);
  I1:=Divide(Vt,Zt);
  AbsI1:=getAbs(I1);
  pf:=cos(getAngleRad(I1));
  P1:=3*FV*absI1*pf;
  FObj:=sqr(abs(absI1/FImea-1))+sqr(abs(P1/FPmea-1));
  result:=100/FObj;
end;

procedure TObjFunc.getObjFuncAkhir(const rChrom:bArr2;
          var rRs,rXs,rRr,rXr,rXm,rPinp,rFitness,rPout,rEff,
          rpf,rAbsI1:double);
var data:dArr1;
    PlossStator:double;
    Zs,Zr,Zm,Ze,Zp,Zt,I1,Vt:TCmplx;
begin
  data:=DecodeBinToFloat2Base0(rChrom);
  rRs:=GetBatasToReal(data[0],FRs.min,FRs.max);
  rXs:=GetBatasToReal(data[1],FXs.min,FXs.max);
  rRr:=GetBatasToReal(data[2],FRr.min,FRr.max);
  rXr:=GetBatasToReal(data[3],FXr.min,FXr.max);
  rXm:=GetBatasToReal(data[4],FXm.min,FXm.max);
  Vt:=Cmplx(FV,0);
  Zs:=Cmplx(rRs,rXs);
  Zr:=Cmplx(rRr/Fs,rXr);
  Zm:=Cmplx(0,rXm);
  Ze:=Multiply(Zr,Zm);
  Zt:=Add(Zr,Zm);
  Zp:=Divide(Ze,Zt);
  Zt:=Add(Zs,Zp);
  I1:=Divide(Vt,Zt);
  rAbsI1:=getAbs(I1);
  rpf:=cos(getAngleRad(I1));
  rPinp:=3*FV*rabsI1*rpf;
  rFitness:=sqr(abs(rabsI1/FImea-1))+sqr(abs(rPinp/FPmea-1));
  PLossStator:=3*sqr(rabsI1)*rRs;
  rPout:=(1-Fs)*(rPinp-PlossStator);
```

```
  rEff:=rPout/rPinp*100;
end;

procedure TObjFunc.getObjFuncAkhir(const rChrom:dArr1;
          var rRs,rXs,rRr,rXr,rXm,rPinp,rFitness,rPout,rEff,
          rpf,rAbsI1:double);
var PlossStator:double;
    Zs,Zr,Zm,Ze,Zp,Zt,I1,Vt:TCmplx;
begin
  rRs:=rChrom[0];
  rXs:=rChrom[1];
  rRr:=rChrom[2];
  rXr:=rChrom[3];
  rXm:=rChrom[4];
  Vt:=Cmplx(FV,0);
  Zs:=Cmplx(rRs,rXs);
  Zr:=Cmplx(rRr/Fs,rXr);
  Zm:=Cmplx(0,rXm);
  Ze:=Multiply(Zr,Zm);
  Zt:=Add(Zr,Zm);
  Zp:=Divide(Ze,Zt);
  Zt:=Add(Zs,Zp);
  I1:=Divide(Vt,Zt);
  rAbsI1:=getAbs(I1);
  rpf:=cos(getAngleRad(I1));
  rPinp:=3*FV*rabsI1*rpf;
  rFitness:=sqr(abs(rabsI1/FImea-1))+sqr(abs(rPinp/FPmea-1));
  PLossStator:=3*sqr(rabsI1)*rRs;
  rPout:=(1-Fs)*(rPinp-PlossStator);
  rEff:=rPout/rPinp*100;
end;

end.
```