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

SKRIPSI

PENENTUAN PARAMETER MOTOR INDUKSI TIGA PHASA DENGAN METODE *EVOLUTIONARY ALGORITMA*

OLEH:
HERMAN YOSEF S. MBORO
01.12.074

SEPTEMBER 2006

PENENTUAN PARAMETER MOTOR INDUKSI TIGA PHASA DENGAN METODE EVOLUTIONARY ALGORITMA

Herman Yosef S. Mboro, 01.12.074
Ir. M. Abdul Hamid, MT

ABSTRAKSI

Penggunaan motor induksi tiga phasa yang sangat luas untuk alat penggerak dalam kepentingan industri. Sekitar 80 % total konsumsi listrik dari motor induksi.

Diperlukan pengujian-pengujian untuk mengetahui nilai parameter suatu moptor induksi, yaitu pengujian arus searah (DC Test), pengujian tanpa beban (No-Load Test), serta pengujian rotor tertahan (Blocked Rotor Test).

Metode Evolutionary Algoritma yang didasarkan oleh mekanisme alam, utnuk mendapatkan nilai parameter yang mendekati nilai sebenarnya (Pengujian) dengan memasukan data name-plate motor sebagai inputan tanpa harus melakukan pengujian.

***Kata Kunci** : Motor induksi, Parameter motor, Torsi, Metode Evolutionary Algoritma*

DETERMINATION PARAMETER OF THREE PHASA INDUCTION MOTOR WITH METHOD *"EVOLUTIONARY ALGORITMHS"*

Herman Yosef S. Mboro, 01.12.074
Ir. M. Abdul Hamid, MT

ABSTRACT

Three phase induction motor used in so many kind of industry. About 80% from total electric consumption from induction motor.

Testing in induction motor needed to find the parameter of machine like Dc Test, No-Load Test and Blocked-rotor Test.

Algorithms evolutionary method base on nature machine to get value of the parameter, closer than real value (f rom testing result). With input the name-plate without doing real testing.

***Key Word** : Induction motor, Motor's parameter, Torque, Algorithms Evolutionary Method.*

DAFTAR ISI

HALAMAN JUDUL i

LEMBAR PERSETUJUAN ii

ABSTRAKSI iii

KATA PENGANTAR iv

DAFTAR ISI v

DAFTAR GAMBAR vi

DAFTAR TABEL vii

DAFTAR GRAFIK viii

DAFTAR FLOWCHART viii

BAB I PENDAHULUAN

1.1. Latar Belakang 1

1.2. Rumusan Masalah 2

1.3. Tujuan 2

1.4. Batasan Masalah 3

1.5. Metodologi Penelitian 4

1.6. Sistemmatika Penulisan 5

1.7. Kontribusi 5

BAB II KONSEP DASAR MOTOR INDUKSI

2.1. Teori Dasar Motor Induksi Tiga Phasa 6

2.2. Konstruksi Motor Induksi Tiga Phasa 7

3.3. Mutation (Mutasi) 32

3.4. Adaptasi Genetika Algoritama Kemasalah Penentuan Parameter Motor Induksi Tiga - Phasa 34
3.4.1. Pengkodean (Representasi) 34

3.4.2. Populasi Awal 37

3.4.3. Reproduksi, Crossover dan mutasi 37

3.5. Objective Function 38

3.5.1. Objective Function untuk permasalahan penentuan parameter motor induksi tiga phasa 38

3.6. Metoda Genetika Programing 40

3.6.1. Parameter Genetika Programing 42

3.6.2. Mekanisme Genetika Programing 43

3.6.3. Objective function 47

3.7. Objective function untuk permasalahan penentuan parameter motor induksi tiga-phasa 47

3.8. Algoritma Program 48

3.8.1. Algoritma program pemecahan masalah secara umum 48

3.8.2. Algoritma program penentuan parameter motor induksi tiga-phasa menggunakan metoda *Genetika Algoritma* 49

3.8.3. Algoritma Penyelesaian penentuan parameter menggunakan metoda *Genetika Programing* 50

3.8.4. Algoritma program fitness 51

BAB IV. PENGUJIAN DAN ANALISIS

4.1. Pengujian Parameter Motot Induksi Tiga - Phasa 52

4.1.1. Alat-Alat Yang Digunakan 52

4.1.2. Pengujian Arus Searah (Dc Test) 53

4.1.3. Pengujian tanpa beban (No-Load Test)..... 53

4.1.4. Pengujian rotor tertahan (Blocked-Rotor Test)..... 54

4.2. Analisa parameter motor induksi tiga-phasa..... 55

4.2.1. Analisa torsi 58

4.2.2. Menghitung nilai torsi 35

BAB V KESIMPULAN DAN SARAN

5.1. Kesimpulan 76

5.2. Saran..... 76

DAFTAR PUSTAKA

LAMPIRAN

DAFTAR GAMBAR

BAB II. KONSEP DASAR MOTOR INDUKSI

2-1.Gamabar. konstruiksi motor induksi 7

2-2. Gambar Stator tiga-phasa motor induksi 7

2-3. Gambar a. Motor induksi rotor belitan 8

b. Rotor sangkar bajing 8

2-4. Gambar. Medan putar pada motor induksi 9

2-5. Gambar Rangkaian ekivalen stator 12

2-6. Gambar Rangkaian ekivalen rotor 13

2-7. Gambar Rangkain ekivalenmotor induksi 14

2-8. Gambar Penyederhanaan rangkaian ekivalen

motor induksi 14

2-9. Gambar Pengujian arus searah (Dc Test)..... 15

2-10.Gambar Diagram pengujian tanpa

beban (No-Load Test) 16

2-11. Gambar Diagram pengujian

rotor tertahan (Blocked-rotor Test) 18

BAB III TEORI DASAR *EVOLUTIONARY ALGORITMA* DAN APLIKASI PADA PENENTUAN PARAMETER MOTOR INDUKSI TIGA - PHASA

3-1. Gambar struktur genetika algoritma 26

3-2. Gambar roulette wheel 29

3-3. Gambar ilustrasi operator crossover dengan one point crossover 31

3-4. Gambar ilustrasi operator crossover dengan two point crossover 31

3-5. Gambar ilustrasi operator uniform crossover 32

3-6. Gambar ilustrasi operator mutasi untuk representasi..... 33

3-7. Gambar algoritma genetika..... 34

3-8. Gambar pengkodean untuk parameter R2 35

3-9. Gambar pengkodean untuk parameter Rm 35

3-10 Gambar pengkodean untuk parameter X1 35

3-11. Gambar pengkodean untuk parameter Xm 36

3-12. Gambar ilustrasi proses mutasi 46

DAFTAR TABEL

BAB. III. TEORI DASAR *EVOLUTIONARY ALGORITMA* DAN APLIKASI PADA PENENTUAN PARAMETER MOTOR INDUKSI TIGA – PHASA

3-1. Tabel istilah yang digunakan dalam Genetika Algoritma .. 23

BAB. IV. PENGUJIAN DAN ANALISA

4-1. Tabel data hasil pengujian arus searah 53

4-2. Tabel data hasil pengujian beban Nol 54

4-3. Tabel data hasil pengujian rotor tertahan 54

4-4. Tabel hasil perhitungan pengujian parameter M.I 57

4-5. Tabel inputan program 65

4-6. Tabel Parameter GA, GP dan Pengujian 65

4-7. Tabel torsi hasil pengujian 66

4-8. Tabel torsi hasil proses genetika algoritma 68

4-9. Tabel torsi hasil proses genetika programing 70

DAFATAR GRAFIK

4-1. Grafik Torsi terhadap slip pengujian, GA dan GP 73

DAFTAR FLOWCHART

4-1. Gambar flowchart algoritma program 61

4-2. Gambar flowchart program genetika algoritma 62

4-3. Gambar flowchart program genetika programming 63

4-4. Gambar flowchart program Fitness 64

BAB I

PENDAHULUAN

1.1. Latar Belakang.

Motor induksi merupakan motor arus bolak-balik (AC) yang paling luas penggunaannya. Motor induksi banyak digunakan dirumah tangga, kantor, pabrik, bengkel, perusahaan-perusahaan maupun industri.

Didalam penggunaannya, motor induksi digunakan sebagai alat penggerak dalam kepentingan industri. Motor merupakan komponen terpenting dalam jaringan sisitem penggerak. Hampir tujuh puluh persen dan bahkan sampai sembilan puluh persen dari semua listrik yang digunakan di industri-industri dikhususkan untuk peralatan yang digerakkan oleh motor. Oleh sebab itu kinerja motor induksi yang optimal merupakan suatu hal yang penting agar kepentingan industri tidak terganggu. Kinerja motor induksi dipengaruhi oleh parameter-parameter yang telah diberikan oleh pabrik pembuatannya. Parameter-parameter tersebut berupa resistansi stator (Rs), resistansi rotor (Rr), reaktansi stator (Xs), reaktansi rotor (Xr), reaktansi maknetisasi (Xm).

Proses pencarian parameter dalam motor induksi yang selama ini digunakan adalah dengan mengadakan beberapa pengujian yaitu ada tiga pengujian atau test, antara lain Pengujian Tanpa beban, Pengujian Rotor Tertahan dan Pengujian Arus Searah. Proses tersebut membutuhkan waktu dan ketelitian alat .

Dalam skripsi ini dibahas sebuah metode analisis yang menggunakan metode *Evolutionary Algoritma* yang akan mencari parameter-parameter motor induksi agar diperoleh parameter yang sesuai dengan data asli motor induksi tersebut.

1.2. Perumusan Masalah

Mengacu pada latar belakang di atas maka permasalahan yang timbul adalah bagaimana menerapkan aplikasi metode *Evolutionary Algoritma* untuk menganalisa penentuan parameter-parameter motor induksi tiga phasa berdasarkan data pengukuran di tempat (pengujian arus searah / *DC test*, pengujian tanpa beban, pengujian rotor tertahan dan data papan-nama / *nameplate*) dari motor induksi sebagai solusi pemecahan masalah. Sehingga skripsi ini mengambil judul :

"PENENTUAAN PARAMETER MOTOR INDUKSI TIGA PHASA DENGAN METODE *EVOLUTIONARY ALGORITMA*"

1.3. Tujuan Penelitian

Tujuan pembahasan ini adalah untuk menganalisis penentuan parameter motor induksi tiga phasa dengan metode *Evolutionary Algoritma.*

1.4. Batasan Masalah

Agar pembahasan dalam skripsi ini lebih terarah sesuai dengan tujuan. maka permasalahan dibatasi oleh hal-hal sebagai berikut :

- Analisis dilakukan pada Motor Induksi Tiga-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, 2820 rpm, 2 kutup.
- Pembahasan hanya ditekankan pada analisa penentuan parameter-parameter motor induksi tiga phasa.
- Motor beroperasi pada keadaan steady state.
- Melakukan perbandingan parameter motor baik secara perhitungan manual dan dengan metode *Evolutionary Algoritma*.
- Tidak membahas sistem proteksi motor induksi.
- Program skripsi ini dijalankan dengan menggunakan bahasa pemrograman Borland Delphi versi 7.0
- Metode yang digunakan untuk mencari parameter adalah Metode *Evolutionary Algoritma*.

1.5. Metodologi Penelitian

Metodologi yang dipakai dalam penyusunan skripsi ini adalah :

1. Kajian Pustaka, yaitu bahan untuk memahami prinsip kerja dari motor induksi tiga-phasa dan metode yang digunakan dalam analisa.
2. Pengumpulan Data

   Pengumpulan data melalui percobaan pada motor induksi dengan pengujian pengujian arus searah (*DC Test*). Pengujian rotor tertahan *(Block Rotor Test)* dan pengujia tanpa beban *(No Load Test)*
3. Melakukan percobaan untuk menentukan parameter motor induksi yang berupa resistansi stator (R_s), resistansi rotor (R_r), reaktansi stator (X_s), reaktansi rotor (X_r), reaktansi magnetisasi (X_m).
4. Melakukan perhitungan untuk menentukan nilai torsi yaitu torsi mula, torsi beban penuh.
5. Melakukan analisis penentuan parameter motor induksi tiga phasa menggunakan metode *Evolutionary Algoritma* dengan bantuan program Delphi.
6. Membuat perbandingan antara parameter hasil pengujian dan parameter dari hasil metode *Evolutionary Algoritma.*
7. Menarik kesimpulan

1.7. Sistematika Penulisan

Skripsi ini dibagi menjadi lima bab, yaitu :

BAB I : PENDAHULUAN

Berisikan tentang latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi, kontribusi dan sistematik penulisan

BAB II : KONSEP DASAR MOTOR INDUKSI

Membahas tentang teori dasar mengenai motor induksi, konstruksi, prinsip kerja motor induksi, rangkaian ekivalen dan pengujian motor induksi tiga phasa.

BAB III : TEORI DASAR EVOLUTIONARY ALGORITMA DAN APLIKASI PADA MOTOR INDUKSI TIGA PHASA

Berisikan tentang teori dasar *Evolutionary Algoritma* dan penerapannya pada penentuan parameter motor induksi tiga phasa.

BAB IV : PENGUJIAN DAN ANALISIS HASIL

Analisis untuk menentukan parameter motor induksi tiga phasa dengan pengujian dan menggunakan metode *Evolutionary Algoritma.*

BAB V : PENUTUP

Berisikan kesimpulan dan saran.

1.6. Kontribusi Penelitian

Dengan adanya analisis ini dapat memberikan suatu alternatif metode dalam membantu pabrik / industri pada penentuan parameter motor induksi tiga phasa menjadi lebih cepat, baik dan tepat sehingga kelangsungan kinerja motor induksi dan proses produksi dalam sebuah industri tidak terganggu.

BAB II
KONSEP DASAR MOTOR INDUKSI TIGA PHASA

2.1. Teori Dasar Motor Induksi.[4]

Motor arus bolak-balik (Motor AC) adalah suatu mesin yang berfungsi untuk mengubah tenaga listrik arus bolak-balik menjadi tenaga mekanik atau tenaga gerak, dimana tenaga gerak ini berupa perputaran pada motor. Salah satu jenis motor AC ini adalah motor induksi atau motor tak serempak.

Disebut motor tak serempak karena putaran motor tidak sama dengan putaran fluks magnet stator. Dengan kata lain, bahwa antara putaran rotor dan putaran fluks magnet terdapat selisih putaran yang disebut slip.

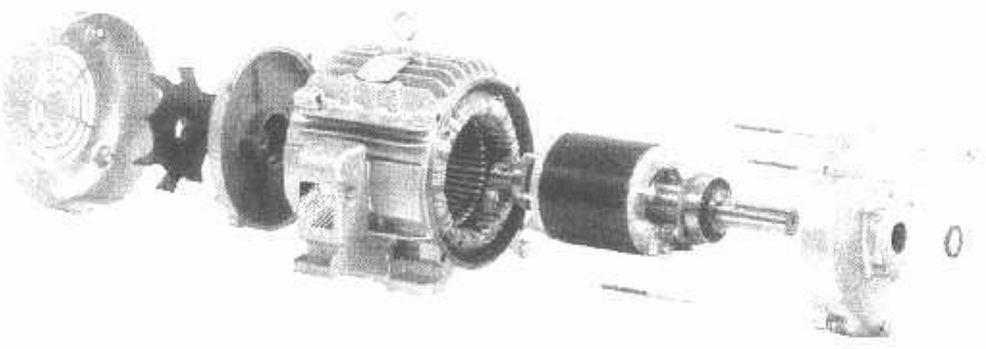
Motor induksi *polyphase* banyak dipakai dikalangan industri. ini berkaitan dengan beberapa keuntungannya.

Keuntungan :

1. Sangat sederhana dan daya tahan kuat (konstruksi hampir tak pernah mengalami kerusakan, khususnya tipe rotor sangkar bajing).
2. Harga relatif murah dan perawatan mudah.
3. Efisiensi tinggi. Pada kondisi berputar normal, tidak dibutuhkan sikat dan karenanya rugi daya yang ditimbulkan dapat dikurangi.

2.2. Konstruksi Motor Induksi [1]

Konstruksi motor induksi terdiri dari dua bagian utama yaitu stator dan rotor. Hal ini dapat dilihat pada gambar 2-1 di bawah ini :

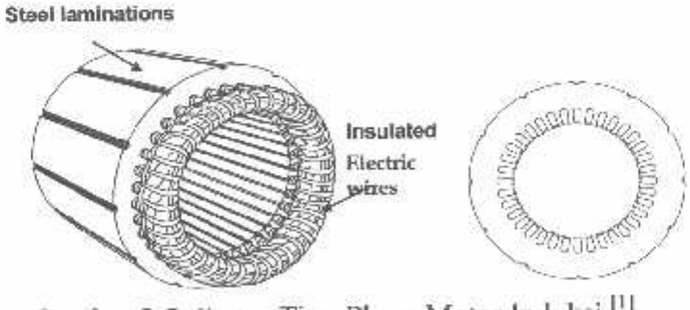


Gambar 2-1 : Konstruksi Motor Induksi [1]

2.2.1. Stator [1]

Pada dasarnya konstruksi stator pada motor induksi mempunyai bentuk fisik yang sama dengan mesin sinkron, yang terdiri dari :

a. Rumah stator terbuat dari besi tuang.
b. Inti stator dari besi atau baja silikon.
c. Alur dan gigi materialnya sama dengan inti, alur tempat meletakan belitan.
d. Belitan stator dari tembaga.



Gambar 2-2: Stator Tiga-Phasa Motor Induksi [1]

2.2.2. Rotor [2]

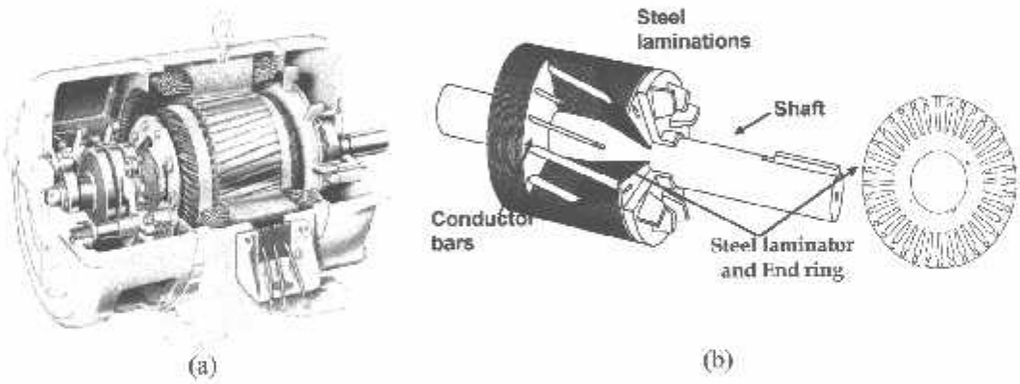
Konstruksi dari rotor motor induksi mempunyai dua bentuk, yaitu :

1. Rotor Belitan (*wound rotor*/ rotor slip ring).

Motor induksi jenis ini mempunyai rotor dengan belitan kumparan tiga-phasa sama seperti kumparan stator. Kumparan stator dan rotor juga mempunyai jumlah kutub yang sama.

2. Rotor Sangkar Bajing (*squirrel cage rotor*).

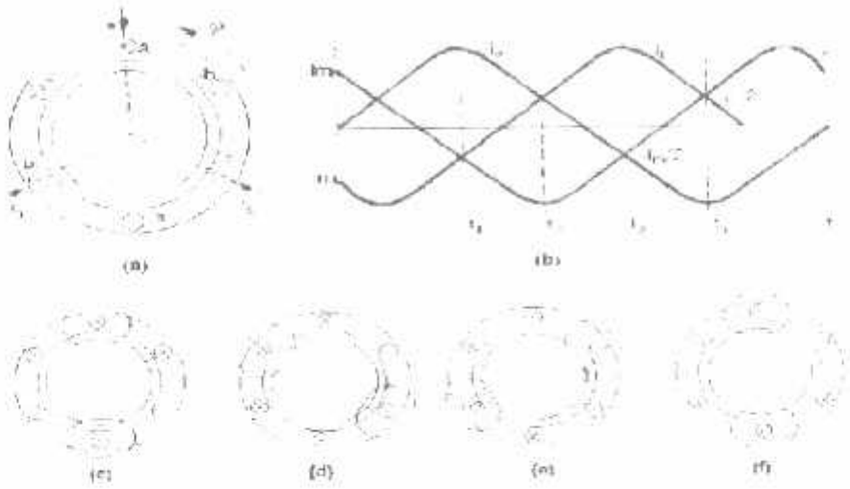
Motor induksi jenis ini mempunyai rotor dengan kumparan yang terdiri atas beberapa batang konduktor yang disusun sedemikian rupa sehingga menyerupai sangkar tupai.



Gambar 2-3 : a) Motor Induksi Rotor Belitan [1]
b) Rotor Sangkar Bajing [1]

2.3. Medan Putar [2]

Perputaran motor 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 phasa banyak, umumnya tiga phasa. hubungan bintang atau delta.



Gambar 2-4 : Medan Putar Pada Motor Induksi [2]

Medan putar terjadi apabila kumparan A-a, B-b, C-c dihungkan tiga phasa dengan beda phasa masing-masing 120^0 (gambar 2-4a) dan dialiri arus sinusoida. Distribusi i_a, i_b, i_c sebagai fungsi waktu adalah seperti gambar 2-4b. Pada keadaan t_1, t_2, t_3 dan t_4 fluks resultan yang ditimbulkan oleh kumparan tersebut masing-masing adalah seperti gambar 2-4c, d, e dan f.

2.4. Prinsip Kerja Motor Induksi [2]

Ada beberapa prinsip kerja motor induksi .

1. Apabila sumber tegangan 3 fasa dipasang pada kumparan stator , timbullah medan putar dengan kecepatan : $n_s = \frac{120}{p} f$
2. Medan putar tersebut akan memotong batang konduktor pada rotor .
3. Akibatnya pada kumparan rotor akan timbul tegangan induksi (GGL) .

4. Karena kumparan rotor merupakan rangkaian yang tertutup , ggl (E) akan menghasilkan arus (I) .
5. Adanya arus I di dalam medan magnet menimbulkan gaya F pada rotor .
6. Bila kopel mula yang di hasilkan oleh gaya F pada rotor cukup besar untuk memikul kopel beban , rotor akan berputar searah dengan medan putar stator .
7. Seperti telah di jelaskan pada (3) , tegangan induksi timbul karena terpotongnya batang konduktor / rotor oleh medan putar stator . Artinya agar tegangan terinduksi di perlukan adanya perbedaan relative antara kecepatan medan putar stator (n_s) dengan kecepatan berputar rotor (n_r)
8. Perbedaan kecepatan antara n_r dan n_s di sebut Slip S di nyatakan dengan ;

$$S = \frac{n_s - n_r}{n_s} x100\%$$

9. Bila $n_r = n_s$, tegangan tidak akan terinduksi dan arus tidak mengalir pada kumparan jangkar rotor , dengan demikian tidak di hasilkan kopel . Kopel motor akan ditimbulkan apabila n_r lebih kecil dari n_s

2.4.1. Slip dan Frekuensi Arus Rotor [2]

Slip diidentifikasikan sebagai bagian Dari kecepatan sinkron n_s dan kecepatan aktual rotor n_r. Slip dirumuskan sebagai berikut :

$$s = \frac{n_s - n_r}{n_s} \quad \text{..... (2.1)}$$

Pada keadaan diam medan magnet putar yang dihasilkan oleh stator mempunyai kecepatan relatif yang sama dengan kumparan rotor. Pada saat ini frekuensi dari

arus rotor sama dengan frekuensi stator ($f_r = f_s$). 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 proporsional dengan slip (s). Hunbungan antara slip dan frekuensi dapat dilihat dari persamaan berikut ini :

$$n_s = \frac{120 f_s}{p} \quad atau \quad f_s = \frac{p n_s}{120} \qquad \text{..... (2.2)}$$

dimana : p = jumlah kutub

f_s = frekuensi stator

Pada rotor berlaku hubungan :

$$f_r = \frac{(n_s - n_r)p}{120} = \frac{(n_s - n_r)n_s p}{n_s 120} \qquad \text{..... (2.3)}$$

$$s = \frac{n_s - n_r}{n_s} \quad dan \quad f_s = \frac{p.n_s}{120}$$

Maka : $f_r = s.f_s$ ..... (2.4)

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 mengasumsikan suatu bentuk yang identik rangkaian ekivalen transformator. Oleh karena itu motor induksi dapat dipandang sebagai

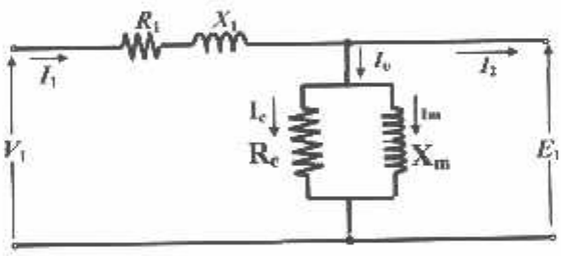
transformator yang mempunyai ciri-ciri khusus yaitu :

1. Stator sebagai sisi primer.
2. Rotor sebagai sisi sekunder yang penghantar-penghantarnya dihubungkan singkat dan berputar.

2.5.2. Rangkaian Ekivalen Stator

Apabila kumparan stator diberikan tegangan catu dari jala-jala sebesar V_1, maka akan mengalir arus putar tiga phasa pada kumparan stator yang membangkitkan medan magnet tiga phasa. Arus stator (I_1) bercabang menjadi dua komponen arus yaitu :

1. Komponen arus beban (I_2)
2. Komponen arus eksitasi (I_0)



Gambar 2-5 : Rangkaian Ekivalen Stator [2][3]

Dimana :

V_1 = tegangan terminal

R_1 = resistansi kumparan

X_1 = reaktansi bocor kumparan

E_1 = tegangan induksi (ggl)

R_c = resistansi tembaga

X_m = reaktansi magnetisasi

2.5.3. Rangkaian Ekivalen Rotor

Pada saat rotor diam, medan putar stator akan memotong batang konduktor rotor 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 / tegangan rotor adalah frekuensi slip, maka reaktansi bocor rotor (*leakage reactance*) adalah :

$$X_2 = sX_2 \quad \text{..... (2.5)}$$

$$X_2 = 2\pi f_s L_2 \quad \text{..... (2.6)}$$

dimana X_2 merupakan reaktansi rotor pada start atau diam.

Tegangan induksi pada rotor :

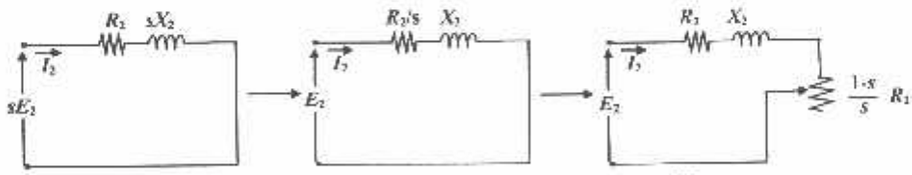
$$E_2 = 4{,}44 f_2 N_2 \Phi_m \quad \text{.....(2.7)}$$

$$f_2 = sf_1$$

$$sE_2 = 4{,}44 sf_1 N_2 \Phi_m \quad \text{.....(2.8)}$$

Dengan memasukkan persamaan (2.7) ke (2.8) maka didapat persamaan :

$$E_2 = sE_2 \quad \text{.....(2.9)}$$



Gambar 2-6: Rangkaian Ekivalen Rotor [2]

Dimana :

S = Slip

E_2 = tegangan induksi pada saat rotor dalam keadaan diam

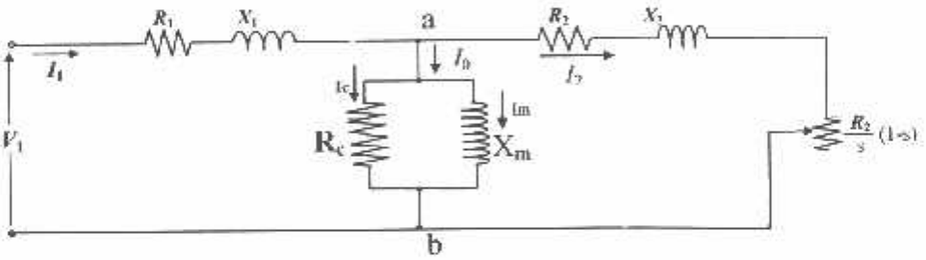
R_2 = resistansi kumparan rotor

X_2 = reaktansi bocor rotor

2.5.4. Rangkaian Ekivalen Motor Induksi

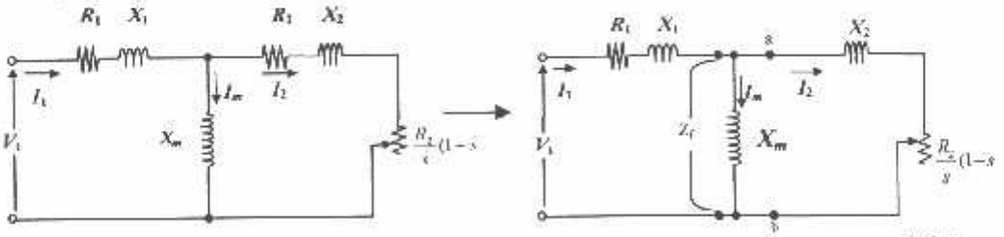
Kerja motor induksi seperti juga kerja pada transformator adalah berdasarkan prinsip induksi elektromagnetik. Oleh karena itu motor induksi dipandang sebagai transformator yang mempunyai ciri-ciri khusus, yaitu :

1. Stator sebagai sisi primer
2. Rotor sebagai sisi sekunder yang penghantar-penghantarnya dihubung-singkat dan berputar
3. Kopling antara sisi primer dan sisi sekunder dipisahkan oleh celah udara (*air gap*).



Gambar 2-7 : Rangkaian Ekivalen Motor Induksi [2][4]

Dalam analisis rangkaian ekivalen, sering disederhanakan dengan menghilangkan konduktansi (R_c), sehingga rangkaian ekivalen pada gambar (2.7) berubah menjadi :



Gambar 2-8 : Penyederhanaan Rangkaian Ekivalen Motor Induksi [3][5]

Tahanan inti adalah :

$$R_C = \frac{V_{th}^{\ 2}}{P_{th}} \Omega \qquad (2.10)$$

Kecepatan singkron dalam radian : $\omega_s = \frac{120.f}{p.60}.2\pi$ ..... (2.11)

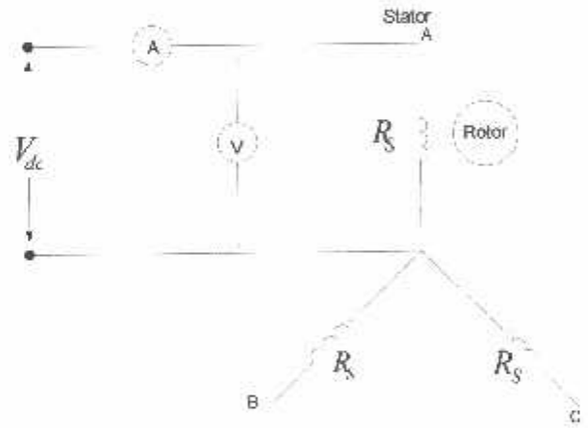
Untuk Torsi diperoleh :

$$T = \frac{3}{\omega_s} \frac{(V_{ab})^2.}{(R_{ab} + R'_r/S)^2 + (X_{ab} + X'_r)^2} \frac{R'_r}{S} = Nm \quad \text{..... (2.12)}$$

2.6. Pengujian Motor Induksi Tiga Phasa

2.6.1. Pengujian Arus Searah (*DC Test*)

Tujuan dari pengujian arus searah (*DC Test*) adalah untuk menentukan nilai resistansi stator. Diagram pengukuran ditunjukkan pada gambar 2-9.



Gambar 2-9 : Pengujian Arus Searah (*DC Test*)

Kumparan stator terhubung bintang (Y) dan bila sumber DC disuplai melalui dua kumparan (kumparan a dan b) , dengan kumparan ke tiga (kumparan c) dalam keadaan terbuka (*open circuit*), maka nilai dari resistansi ekivalen (R_{ek}) :

$R_{ek} = 2R_s\ \Omega$ ..... (2.13)

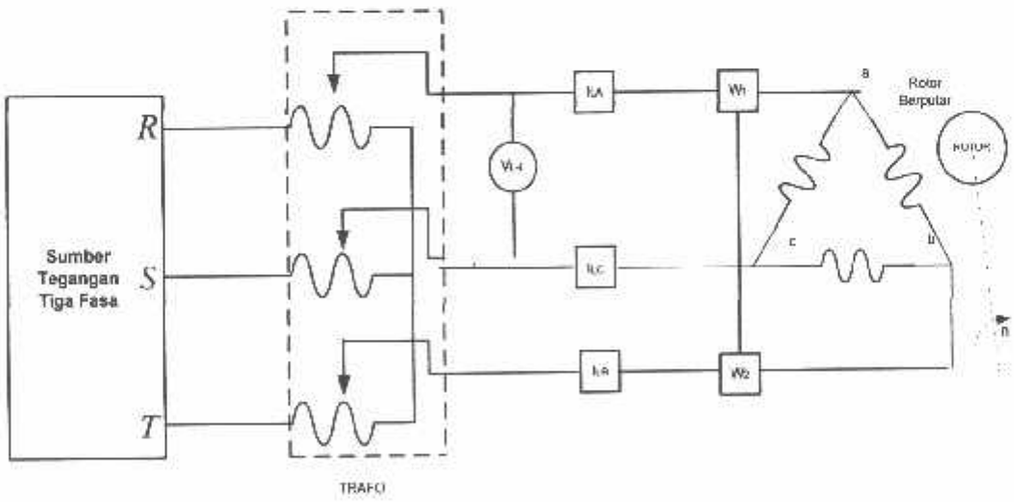
untuk nilai resistansi kumparan stator perphasa (Rs):

$$R_s = R_{dc} = \frac{V_{DC}}{I_{DC}}\ \Omega \quad \text{.....(2.14)}$$

Dalam pengujian arus searah dijaga agar arus DC (I_{DC}) tidak melampaui nilai dari arus nominal motor induksi.

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

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



Gambar 2-10 : Diagram Pengujian Tanpa Beban

$P_{3-\Phi}$,daya total yang terukur dari W_a dan W_b :

$$P_{3-\Phi} = W_a + W_b \quad \text{watt} \qquad (2.15)$$

I_{nl}, arus phasa stator :

$$I_{th} = \frac{I_a + I_b + I_c}{3} \text{ Ampere} \qquad (2.16)$$

Reaktansi diri stator :

$$X_{ss} = X_s + X_m = X_{th}$$

Dimana:

I_{tb} = arus tanpa beban

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

P_{rot} = rugi-rugi putaran tanpa beban

Tahanan inti adalah :

$$R_C = \frac{V_{th}^{\ 2}}{P_{tb}} \Omega$$

Resistansi tanpa beban adalah :

$$R_{th} = \frac{P_{th}}{3.(I_{th})^2} \Omega \qquad (2.17)$$

Impedansi tanpa beban adalah :

$$Z_{tb} = \frac{V_{th}}{\sqrt{3} I_{th}} \Omega \qquad (2.18)$$

$$Z_{th} = R_{th} + j\, X_{th} \qquad (2.19)$$

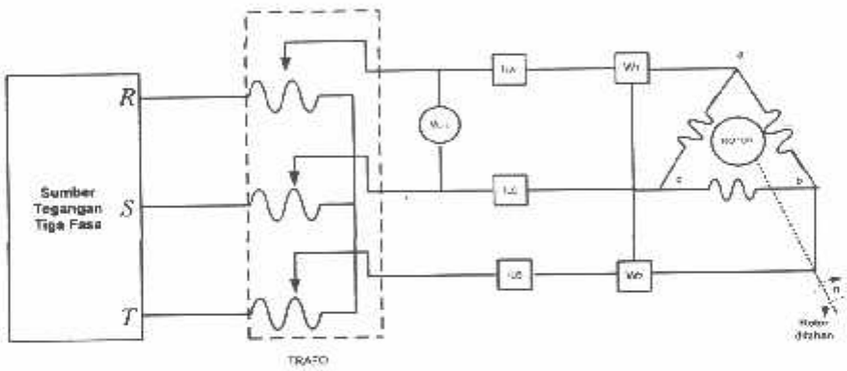
Reaktansi tanpa beban adalah :

$$X_{th} = \sqrt{Z_{th}^{\ 2} - R_1^{\ 2}}\ \Omega \qquad (2.20$$

$$X_m = X_{th} - X_1 \qquad (2.21)$$

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

Tujuan pengujian rotor tertahan adalah untuk menentukan resistansi rotor pada motor induksi. Pada saat pengujian ini perputaran rotor motor induksi dikunci / diblok sehingga slip(s) sama dengan satu. Suplai tegangan tiga phasa motor induksi adalah tegangan yang nilainya di bawah tegangan nominalnya, yakni tegangan yang dapat menghasilkan arus nominalnya. Sebagai pendekatan, diasumsikan bahwa arus pemagnetan (I_m) cukup kecil akibat penurunan suplai tegangan serta motor dalam keadaan tidak berputar (s–1) sehingga rugi-rugi inti dapat diabaikan.



Gambar 2-11 : Diagram Pengujian Rotor Tertahan

$P_{3\text{-}\phi}$,daya total yang terukur dari W_a dan W_b :

$$P_{3\text{-}\phi} = W_a + W_b \quad \text{watt} \qquad (2.22$$

Daya total tiga-phasa merupakan rugi-rugi tembaga stator dan rotor, karena motor tidak berputar maka rugi-rugi inti diabaikan.

I_{rt}, arus phasa stator :

$$I_{rt} = \frac{I_a + I_b + I_c}{3} \text{ Ampere} \qquad (2.23)$$

Resistansi rotor tertahan adalah :

$$R_{rt} = \frac{P_{rt}}{3.(I_{rt})^2}\Omega \quad \text{.....(2.24)}$$

Impedansi rotor tertahan adalah :

$$Z_{rt} = \frac{V_{rt}}{\sqrt{3}.I_{rt}}\ \Omega \quad \text{.....(2.25)}$$

Reaktansi rotor tertahan adalah :

$$X_{rt} = \sqrt{Z_{rt}^{\ 2} - R_{rt}^{\ 2}}\ \Omega \quad \text{.....(2.26)}$$

$$X_{ek} = X_1 + X'_2 \quad \text{.....(2.27}$$

Dimana:

I_{rt} = arus pada keadaan rotor tertahan

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

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

Motor induksi yang dipakai adalah motor induksi dengan rotor sangkar tunggal.

Kelas A maka secara umum :

$$\frac{x_1}{x_1 + x_2} = 0.5 \quad \text{.....(2.28)}$$

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

$$X_{ss} = X_{tb} = X_s + X_m \quad \text{.....(2.29)}$$

Resistansi stator dapat dipandang sebagai harga Denya. Maka resistansi rotor dapat ditentukan sebagai berikut :

$$R = R_{rt} - R_s$$

Dengan $X_{rr} = X_r + X_m$ merupakan reaktansi diri rotor maka :

$$X_{rr} = X_r + X_m \quad \text{.....(2.30)}$$

$$R'_r = R\left(\frac{X_{rr}}{X_m}\right)^2 \quad \text{.....(2.31)}$$

BAB III

TEORI DASAR EVOLUTINARY ALGORITMA DAN APLIKASI PADA PENENTUAN PARAMETER MOTOR INDUKSI TIGA PHASA

3.1. Evolutionary Algoritma[6]

Evolutionary Algoritma (EA) merupakan dua metode, yaitu metode algoritma genetika dan metode genetika programing yang dapat digunakan untuk menghitung parameter motor induksi dengan ketelitian yang tinggi. Teori *Evolutionary Algoritma* merupakan perbandingan hasil proses perhitungan metode genetika algoritma dengan metode genetika programing, sehingga kita dapat melihat hasil perbandingan dari kedua metode tersebut pada suatu kriteria kinerja untuk menentukan error terkecil.

3.1.1. Algoritma Genetika[4]

Algoritma Genetika merupakan metode adaptive yang bisa digunakan untuk memecahkan suatu pencarian nilai dalam sebuah masalah optimasi. Algoritma ini didasarkan pada proses genetik yang ada dalam makluk hidup, yaitu perkembangan generasi dalam sebuah populasi yang alami, secara lambat laun mengikuti prinsip seleksi alam. 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

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 permasalahan-permasalah yang sulit dengan langkah alamia yaitu melalui evolusi. John Holland mulai bekerja dengan algoritma yang dibentuk dengan 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 mengetahui tipe permasalahan yang akan diselesaikan. Hanya informasi yang telah diberikan dari *evaluasi* berupa nilai fitness setiap kromosom yang dihasilkan digunakan untuk seleksi kromosom sehingga kromosom dengan nilai fitness terbaik yang bertahan hidup dan selalu diproduksi.

Sebelum Genetika Algoritma dijalankan, sebuah kode yang sesuai (*representasi*) untuk persoalan harus dirancang. Titik solusi dalam ruang permasalahan dikodekan dalam bentuk kromosom / string yang terdiri dari komponen yang terkecil yaitu gen. Pemakaian bilangan seperti integer, floating point dan abjad sebagai *allele* (nilai gen) memungkinkan penerapan operator genetika, yaitu :

- Reproduksi (*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 Genetiika Algoritma didesain dengan baik, populasi akan mengalami konvergensi dan akan mendapatkan sebuah solusi yang optimum.

Genetika Algoritma 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 optimum global.
4. Menggunakan aturan transisi secara probabilistik bukan deterministik.

3.1.2. Istilah-Istilah Genetika Algoritma

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. Mitsou Gen dan Runwei Cheng (1997) menjelaskan istilah-istilah yang digunakan didalam Genetika algolritma sebagai berikut :

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

Tabel 3-1 : Istilah yang digunakan dalam Algoritma Genetika

Sumber : Mitsuo Gen, Runwei Cheng, " Genetic Algorithm and Egineering Design", (john Wiley & Son, Inc., 1997), p.7.

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

- **Jumlah Generasi (*MAXGEN*)**

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

- **Ukuran Populasi (*POPSIZE*)**

keturunan akan semakin mirip dengan induknya. Dalam Gentika Algoritma, mutasi menjalankan aturan yang penting, yaitu :

1. Menggantikan gen-gen yang hilang selama proses seleksi.
2. Menyediakan gen-gen yang tidak muncul pada saat inisalisasi awal populasi.

➢ **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 genetik terkecil yaitu gen. Pengkodean dapat memakai bilangan seperti string biner, integer, floating point dan abjad.

Mekanisme secara umum dari Genetika Algoritma digambarkan oleh Golberg. Proses kerjanya diawali oleh inisialisasi satu rangkaian nilai random yang disebut *populasi*. Setiap individu didalam populasi disebut *kromosom*. Sebuah kromosom dapat direpresentasikan dalam bentuk simbol-simbol string biner, floating point, integer, abjad. Kromosom-kromosom ini berkembang melalui beberapa iterasi yang disebut *generasi*. Setiap generasi, kromosom-kromosom ini dievaluasi dengan menggunakan ukuran *fitness* melalui fungsi tujuan (*Objective Funtion*) dan batasan-batasan fungsi kendala sehingga individu dengan solusi yang terbaik yang terpilih. Untuk menghasilkan generasi selanjutnya ($t+1$) sebagai kromosom baru yang disebut sebagai *offspring*, dibentuk melalui penggabungan dua kromosom saat ini (t) dengan menggunakan operator crossover dan memodifikasi sebuah kromosom menggunakan operator

mutasi. Satu generasi baru dibentuk melalui proses seleksi sesuai dengan *fitness values* kromosom orang tua dan kromosom yang fit yang akan diturunkan. Kromosom dengan nilai fitness terbesar mempunyai probabilitas tertinggi untuk terpilih. Setelah beberapa generasi, algoritma konvergen pada kromosom terbaik, yang diharapkan sebagai solusi optimum atau solusi suboptimum permasalahan. Prosedur Genetika Algoritma dimodifikasi menurut versi Grenfenstette dan Baker's, digambarkan sebagai berikut :

```
Begin
    t =0;
    Inisialisasi P(t);
    Evaluasi P(t);
    While ( not termination condition ) do
        Recombine P(t) ke C (t);
        Evaluasi C(t);
        Seleksi P(t+1) dari P(t) dan C(t);
    t= t +1;
    End;
```

Gambar 3-1 : Struktur Genetika Algoritma
Sumber : Golberg, "Genetic Algorithms in Machine", NY, 1996.

3.1.3. Proses Genetika Algoritma

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

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

Secara umum Genetika Algoritma dibentuk oleh serangkaian kromosom yang ditandai dengan x_i (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. Pengkodean string biner merupakan pendekatan paling klasik yang digunakan dalam penelitian Genetika Algoritma karena sederhana. Meskipun representasi dengan cara ini menyulitkan untuk beberapa permasalahan optimasi, misalnya permasalahan Graph coloring.

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

B. Fungsi Fitness (Fungsi Evaluasi)

Dalam Genetika Algoritma, 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 x_i (i =1,2,...N) dipergunakan untuk menghitung fk (x) dimana k = 1,2,.... *POPSIZE*.

Pada permulaan optimasi, biasanya nilai fitness masing-masing individu masih mempunyai rentang yang lebar. Seiring dengan bertambahnya generasi, beberapa kromosom mendominasi populasi dan mengakibatkan rentang nilai

fitness semakin kecil. Hal ini dapat mengakibatkan konvergensi dini (*premature convergence*).

Permasalahan klasik dari Genetika Algoritma adalah beberapa kromosom dengan nilai fitness yang tertinggi (tetapi bukan nilai optimum) mendominasi populasi dan mengakibatkan Genetika Algoritma konvergen pada lokal optimum. Ketika mencapai konvergen, kemampuan Genetika Algoritma untuk mencari solusi yang lebih baik menghilang. Tukar silang antara kromosom induk yang hampir identik menghasilkan keturunan (*offspring*) yang identik. Dalam hal ini hanya operasi mutasi yang mampu menghasilkan kromosom yang relatif baru dan merupakan cara untuk menghindari kromosom tertentu mendominasi populasi.

C. Seleksi

Pada Genetika Algoritma 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 penyeleksiannya. Menurut teori evolusi 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 sebagainya.

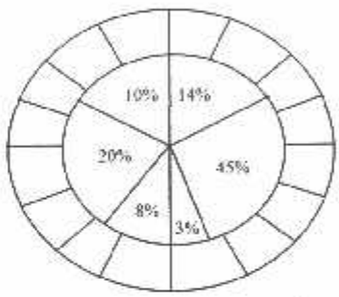
- **Roulette Wheel Selection (Seleksi Roda Roulette)**

Dimana setiap individual memiliki harga fitness sehingga didapatkan probabilitas individual ($f(t) / \Sigma f(t)$)tersebut dicopykan pada populasi yang baru. Untuk individual yang memiliki probabilitas 20% untuk jumlah populasi 10 maka

kemungkinan individual tersebut dapat terpilih sebanyak dua kali. Ilustrasi kerja dari operator ini digambarkan seperti pada gambar 3-2.

Adapun algoritma dari *roulette wheel* adalah sebagai berikut :

1. Menjumlahkan fitness dari seluruh anggota populasi.
2. Membangkitkan nilai *k*, suatu nilai random antara 0 dan total fitnessnya.
3. Menjumlahkan fitness dari kromosom-kromosom dalam populasi mulai dari 0 hingga total fitness lebih besar atau sama dengan nilai *k* lalu ambil kromosom tersebut.



Gambar 3-2 : Roulette Wheel
Sumber : Golberg, "Genetic Algorithms in Machine", NY, 1996.

- **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 meranking populasi dan kemudian tiap kromosom diberi fitness baru berdasarkan hasil ranking tersebut. Yang pertama akan mempunyai fitness 1, yang kedua akan mempunyai fitness 2, dan seterusnya

sampai yang terakhir akan mempunyai fitness. Dengan demikian semua kromosom akan mempunyai peluang yang cukup besar untuk diseleksi.

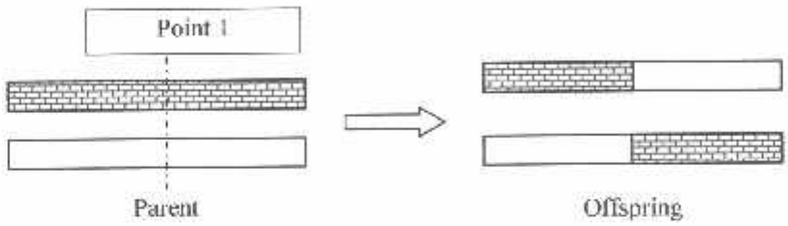
3.1.4. Elitism

Selama membuat populasi baru dengan crossover dan mutasi, kemungkinan akan terjadi kehilangan kromosom terbaik (*best/few best*). Elitism adalah nama metode yang pertama kali meng-Copy-kan kromosom terbaik (*best/few best*) ke dalam populasi baru. Sisanya dikerjakan dengan cara yang biasa, yaitu melalui seleksi, crossover dan mutasi. Elitism dapat secara cepat meningkatkan performasi dari g entika Algoritma karena elitism menghindarkan hilangnya solusi terbaik (*best/few best*) yang telah ditemukan.

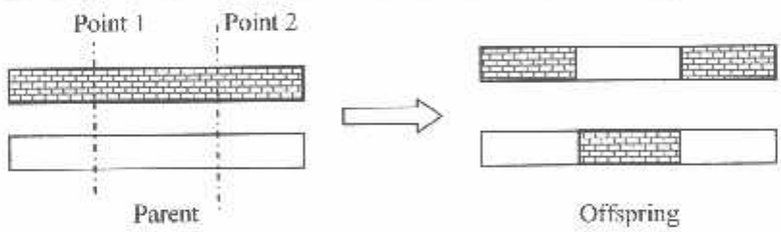
3.2. Crossover (Pindah Silang)

Fungsi dari crossover adalah menghasilkan kromosom anak dari kombinasi materi-materi gen dua kromosom induk. Cara kerjanya dengan membangkitkan sebuah nilai random r_k dimana $k = 1,2,...POPSIZE$. Probabilitas crossover (P_c) ditentukan dan digunakan untuk mengendalikan frekuensi 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 dua (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.



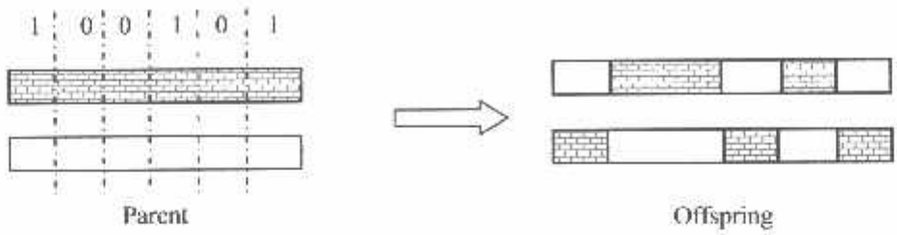
Gambar 3-3 : Ilustrasi Operator Crossover dengan One Point Crossover
Sumber : Mitsuo Gen, Runwei Cheng, " Genetic Algorithm and Egineering Design", (john Wiley & Son, Inc., 1997), p.7.



Gambar 3-4: Ilustrasi Operator Crossover dengan TwoPoint Crossover
Sumber : Mitsuo Gen, Runwei Cheng, " Genetic Algorithm and Egineering Design", (john Wiley & Son, Inc., 1997), p.7.

Untuk crossover uniform dibangkitkan suatu nilai random 0 dan 1 sepanjang jumlah kromosom untuk tiap loci. Jika nilai yang dibangkitkan mempunyai nilai 1 maka allele offspring 1 untuk loci tersebut diambil dari allele parent 2 dan offspring 2 untuk loci tersebut diambil dari allele parent 1. Jika nilai yang dibangkitkan mempunyai nilai 0 maka allele offspring 1 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 Egineering Design", (john Wiley & Son, Inc., 1997), p.7.

3.3. Mutation (Mutasi)

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 mutasi. Apabila nilai random $r_k < P_m$, maka gen ke-k kromosom tersebut terpilih untuk mengalami mutasi. Mutasi dengan menggantikan 0 dengan 1 atau sebaliknya gen 1 dengan 0. Biasanya disebut dengan proses flip yaitu dengan membalik nilai 0 ke 1atau 1 ke 0. Ilustrasi kerja operator ini digambarkan seperti pada gambar 3-6.

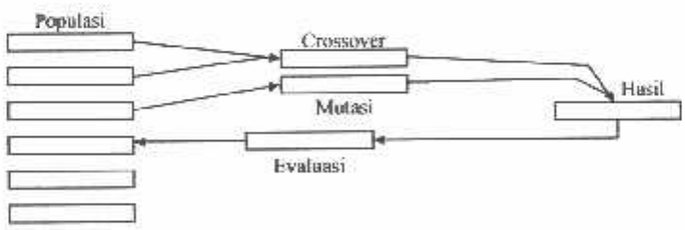
Fungsi dari operator mutasi adalah untuk menghindari agar solusi permasalahan yang diperoleh bukan merupakan solusi optimum lokal. Seperti halnya pada operator crossover, tipe dan implementasi dari operator mutasi bergantung pada jenis pengkodean dan permasalahan yang dihadapi. Seberapa sering mutasi dilakukan dinyatakan dengan suatu probabilitas mutasi, P_m. Posisi

7. Mengulangi proses tersebut sampai diperoleh suatu nilai parameter yang konvergen.

3.4. Adaptasi Genetika Algoritma Ke Masalah Penentuan Parameter Motor Induksi Tiga-Phasa.

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

Genetika Algoritma menggunakan *Objective Function* yang didasarkan pada suatu kriteria kinerja untuk menentukan error.



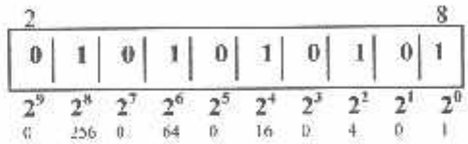
Gambar 3-7 : Algoritma Genetika
Sumber : Mitsuo Gen, Runwei Cheng, " Genetic Algorithm and Egineering Design", (john Wiley & Son, Inc., 1997), p.7.

3.4.1. Pengkodean (Representasi)

Sebuah motor induksi tiga-phasa terdiri dari parameter yaitu R_2, R_m, X_1 dan X_m mewakili satu (1) individu yang terdiri dari empat (4) string atau kromosom. Setiap kromosom ditentukan panjangnya 30, yang dibatasi nilai minimum dan

nilai maksimum. Didalam string terdapat beberapa gen, dimana masing-masingnya adalah kode biner antara 0 dan 1.

Kromosom pertama adalah nilai R_2 dimana dalam R_2 ditentukan bahwa R_2 minimum 2 dan maksimum 8. Maka arti dari pengkodean tersebut adalah[4] :

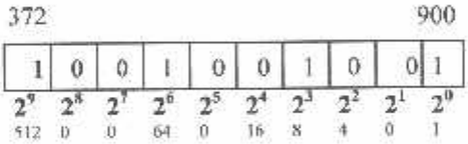


Gambar 3.8 : Pengkodean Untuk Parameter R_2

Untuk mencari R_2 dengan melalui persamaan sebagai berikut :.....

$$R_2 = R_{2min} + \left[\frac{R_{2maks} - R_{2\,min}}{2^{10} - 1} \right] \quad \text{.....(3.1)}$$

Kromosom ke dua adalah nilai R_m dimana dalam R_m ditentukan bahwa R_m minimum 372 dan maksimum 900. Maka arti dari pengkodean tersebut adalah :

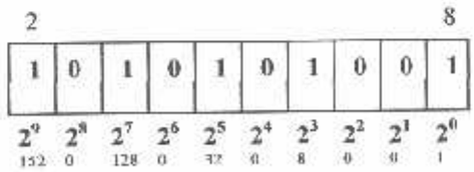


Gambar 3.9 : Pengkodean Untuk Parameter R_m

Untuk mencari R_m dengan melalui persamaan sebagai berikut :

$$R_m = R_{m\,min} + \left[\frac{R_{m\,maks} - R_{m\,min}}{2^{10} - 1} \right] \quad \text{.....(3.2)}$$

Kromosom ke tiga adalah nilai X_1 dimana dalam X_1 ditentukan bahwa X_1 minimum 2 dan X_1 maksimum 8. Maka arti dari pengkodean tersebut adalah :

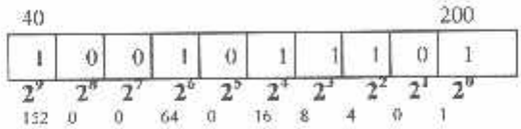


Gambar 3.10 : Pengkodean Untuk Parameter X_1

Untuk mencari X_1 dengan melalui persamaan sebagai berikut :

$$X_1 = X_{1\min} + \left[\frac{X_{1\,maks} - X_{1\,\min}}{2^{10} - 1}\right] \quad \text{.....(3.3)}$$

Kromosom ke empat adalah nilai X_m dimana dalam X_m ditentukan bahwa X_m minimum 40 dan X_m maksimum 200. Maka arti dari pengkodean tersebut adalah :



Gambar 3.11 : Pengkodean Untuk Parameter X_m

Untuk mencari X_m dengan melalui persamaan sebagai berikut :

$$X_m = X_{m\min} + \left[\frac{X_{m\,maks} - X_{m\,\min}}{2^{10} - 1}\right] \quad \text{.....(3.4)}$$

Setelah nilai R_2, R_m, X_1, dan X_m diperoleh maka dimasukkan ke persamaan-persamaan rangkaian ekivalen. Dari hasil perhitungan dengan menggunakan persamaan rangkaian ekivalen maka akan diperoleh nilai arus stator dan daya masukan yang kemudian dimasukkan kedalam fungsi evaluasi (*Objective Function*) sebagai berikut[11] :

$$F_{objective} = \sum_{i=1}^{n}\left|\frac{I_{1i,metode}}{I_{1i,ukur}} - 1\right|^2 + \sum_{i=1}^{n}\left|\frac{P_{input\ 1,metode}}{P_{input\ 1,ukur}} - 1\right| \quad \text{.....(3.5)}$$

Selanjutnya dicari error sebagai berikut :

$$\varepsilon = F_{Objective} \quad \text{.....(3.6)}$$

dan dicari *fitness*nya adalah :

$$Fitness = \frac{100}{100 + F_{objective}} \quad \text{.....(3.7)}$$

Setelah semua individu dihitung *fitness*nya maka dicari prosentase fitness untuk masing-masing individu yaitu :

$$Prosentase = (Fitness_k / sum\ fitness).100\% \quad \ldots\ldots (3.8)$$

Dimana :

$Fitness_k$ = fitness individu ke-k

Sum Fitness = jumlah fitness seluruh individu

$I_{1,metode}$ = Hasil proses metode menggunakan persamaan pada pada metode

$P_{input\ i,\ metode}$ = Hasil proses metode dengan menggunakan persamaan pada metode

$I_{1,ukur}$ = Hasil pengukuran yang diperoleh dari test di lapangan.

$P_{input\ i,ukur}$ = Hasil pengukuran yang diperoleh dari test di lapangan.

Tujuan dari optimasi genetika adalah untuk meminimalisasi kesalahan (Fungsi objektif/$F_{objective}$) antara pengukuran dan proses metode Genetika Algoritma.

3.4.2. Populasi Awal

Untuk sebuah motor induksi tiga phasa, populasi awal yang dibangkitkan secara random (acak). Dalam pembahasan kali ini populasi dibangkitkan sebanyak 100 populasi.

3.4.3. Reproduksi, Crossover dan Mutasi

Generasi baru dibuat dari hasil yang diperoleh dari generasi sebelumnya. Untuk setiap individu, pertama kita menghitung dan menormalisasikan nilai *objective function*. Semakin rendah nilai *objective function*-nya maka individu akan semakin baik. Reproduksi, crossover dan 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 pada gambar 3-2.

Proses crossover dapat dilakukan dengan probilitas crossover (P_c). Jika nilai yang dibangkitkan kurang dari Pc maka tidak perlu diadakan crossover. Pada proses ini, posisi untuk memotong ke empat string (R_2, R_m, X_1 dan X_m) untuk kedua individu dipilih secara acak.

3.5. *Objective Function*

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

3.5.1. *Objective Function* Untuk Permasalahan Penentuan Parameter Motor Induksi Tiga Phasa[1]

Tujuan dari optimasi genetika adalah untuk meminimalisasi kesalahan (*error*) antara pengukuran dan perhitungan parameter. Parameter input tersebut akan dimasukkan ke dalam *Objective function* yang dirumuskan sebagai berikut :

$$F_{objective} = \sum_{i=1}^{n} \left| \frac{I_{1i,metode}}{I_{1i,ukur}} - 1 \right|^2 + \sum_{i=1}^{n} \left| \frac{P_{input\ i,metode}}{P_{input\ i,ukur}} - 1 \right|^2 \quad \text{.....(3.9)}$$

Selanjutnya dicari error sebagai berikut :

$$\varepsilon = F_{Objective} \quad \text{.....(3.10)}$$

dan dicari *fitness*nya adalah :

$$Fitness = \frac{100}{100 + F_{objective}} \quad \text{.....(3.11)}$$

Perhitungan di atas untuk satu individu sedangkan untuk individu lainnya dilakukan dengan cara yang sama dengan catatan masing-masing kromosom nilainya random. 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 atau nilai *fitness* terbesar. Nilai *error* yang diperoleh digunakan untuk menentukan parameter motor induksi tersebut. Parameter motor tersebut didapat dari rangkaian ekivalen motor induksi.

Kemudian parameter motor induksi diacak untuk mendapatkan nilai optimum dengan metode algoritma genetika sehingga didapat nilai fitness yang maksimal.

3.6. Genetika Programming[6]

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

Genetika Programming adalah suatu metode strategi optimasi yang merupakan cabang dari *Evolutionary Computation* yang didalamnya terdiri dari *Algoritma Genetika, Genetic Programming, Evolutionary Strategies,* Perbedaan yang paling mendasar antara *Genetika Programming* dengan *Algorithm Genetika* adalah pada proses operasi. Dalam metode *genetika Programming* tidak menggunakan operasi *crossover* melainkan operasi *competition* (kompetisi).

Genetika Programming ditemukan oleh Lawrence.J. Fogel yang dilandasi oleh sifat-sifat evolusi alam. Fogel 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. Fogel mulai bekerja dengan algoritma yang dibentuk oleh *string-string* bilangan real yang disebut *kromosom*. Seperti halnya alam, metode ini menyelesaikan permasalahan-permasalahan dengan menemukan kromosom-kromoson yang baik dengan memanipulasi materi dan sifat (*gene*) *kromosom-*

kromosom. Algoritma ini tidak mengetahui tipe permasalahan yang akan diselesaikan .

Sebelum *Genetika Programming* dijalankan. maka sebuah kode yang sesuai *(representasi)* untuk persoalan harus dirancang. Titik solusi dalam ruang permasalahan dikodekan dalam bentuk *kromosom/string* yang terdiri dari *genetic* terkecil yaitu gen. Pemakaian bilangan real (*floating point*) sebagai *allele* (nilai gen) memungkinkan penerapan operator *Genentika Programming* yaitu proses produksi (*reproduction*), mutasi (*mutation*), dan kompetisi (*competition*) untuk menciptakan himpunan 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, mutasi dan kompetisi untuk menciptakan keturunan (*offspring*).

Genetika Programming 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, 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*.

3.6.1. Parameter *Genetika Programming*

Terdapat beberapa parameter pada *Genetika Programming* yang digunakan untuk melihat kompleksitas dari *genetika Programming* itu sendiri. Parameter yang digunakan adalah :

- **Jumlah Generasi (MAXGEN)**

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

- **Ukuran Populasi (POPSIZE)**

Ukuran populasi mempengaruhi kinerja dan efektifitas dari GP. Jika ukuran populasi kecil maka populasi tidak menyediakan cukup materi untuk mencakup ruang permasalahan, sehingga pada umumnya kinerja GP menjadi buruk. Dalam hal ini dibutuhkan ruang yang lebih besar untuk mempersentasikan keseluruhan ruang permasalahan. Selain itu penggunaan populasi yang besar dapat mencegah terjadinya *konvergensi* pada wilayah lokal.

- **Probabilitas Mutasi (*Pm*)**

Mutasi digunakan untuk meningkatkan variasi populasi dan 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 suatu 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 GP mutasi menjalankan aturan penting yaitu :

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

❖ **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 memakai *string* bilangan real.

3.6.2. Mekanisme *Genetika Programming*

A. Pengkodean atau *Representasi*

Langkah pertama kali yang dilakukan dalam penggunaan *genetika Programming* adalah melakukan pengkodean atau *representasi* terhadap permasalahan yang akan dilakukan.

Secara umum GP 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 *allele*. Variabel-variabel ini dapat dinyatakan dalam bentiuk bilangan real (*floating point*).

Selanjutnya bebcrapa kromosom dibentuk dan berkumpul membentuk populasi. Populasi inilah populasi awal bagi GP untuk awal melakukan pencarian.

B. Fungsi *Fitness* (Fungsi Evaluasi)

Dalam GP, 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)*

Pada permulaan optimasi, biasanya nilai *fitness* masing-masing individu masih mempunyai rentang yang lebar. Seiring dengan bertambah besar generasi, beberapa kromosom mendominasi populasi dan mengakibatkan rentang nilai *fitness* semakin kecil.

C. Seleksi

Masalah yang paling mendasar pada proses ini adalah bagaimana proses penyeleksiannya. Menurut teori Darwin, proses seleksi individu adalah : "individu terbaik akan tetap hidup dan akan menghasilkan keturunan". Pada proses seleksi ini dapat banyak menggunakan metode seperti *rouletee wheel selection, rank selection, elitesm* dan lain sebagainya.

❖ ***Roulette Wheel Selection***

Dimana setiap individual memiliki harga *fitness* sehingga didapatkan *probabilitas individual* $(f(t)/\sum f(t))$ tersebut dicopykan pada populasi yang baru. Untuk individual yang memiliki probabilitas 20% untuk jumlah populasi 10 maka kemungkinan individual tersebut dapat terpilih sebanyak dua kali.

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

1. Menjumlahkan *fitness* dari seluruh anggota populasi.

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

❖ ***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 mempunyai *fitness* 2 dan seterusnya sampai yang terakhir akan mempunyai *fitness* N. Dengan demikian semua kromosom akan mempunyai peluang untuk diseleksi.

D. *Mutation* (Mutasi)

Operator mutasi digunakan untuk memodifikasi satu atau lebih nilai gen dalam individu. Cara kerjanya dengan membangkitkan sebuah nilai random r_k dimana k = 1,2,...,*NVAR* (panjang kromosom). Probabilitas mutasi (*Pm*) 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. Proses mutasi dalam *Evolutionary Strategies* yaitu menggunakan operator *Gaussian Mutation*, dimana setiap individu akan terpilih secara acak untuk mengalami mutasi berdasarkan nomor acak Gaussian untuk untuk menciptakan individu baru (*offspring*).

(a) (b)

Gambar 3.1. Ilustrasi Proses Mutasi

a). Mutasi Gaussian dari Induk (*parent*) b). Menghasilkan Anak (*offspring*)

Fungsi dari operator mutasi adalah untuk menghindari agar solusi permasalahan yang diperoleh bukan merupakan solusi optimum lokal. Tipe dan implementasi dari operator mutasi bergantung dari pada jenis pengkodean dan permasalahan yang dihadapi. Seberapa sering mutasi dilakukan dinyatakan dengan suatu probabilitas mutasi, Pm. Posisi elemen pada kromosom yang akan mutasi ditentukan secara random. Mutasi dikerjakan dengan cara melakukan perubahan pada elemen tersebut.

E. *Competition* (kompetisi)

Dalam tahap kompetisi, mekanisme seleksi dipakai untuk menghasilkan populasi baru dari populasi yang ada. Melalui penggunaan skema kompetisi setiap individu dalam populasi baik orang tua (*parent*) maupun anak (*offspring*) akan dikompetisi/bersaing satu dengan yang lainnya. Kompetisi setiap individu dengan lawannya didasarkan pada nilai *fitness* dari setiap individu tersebut. Agar optimal, solusi yang lebih pas atau lebih optimal seharusnya memiliki peluang seleksi yang lebih besar. Individu yang memenangkan dari kompetisi akan digunakan sebagai individu yang baru bagi pembangkitan selanjutnya.

3.6.3. *Objective Function*

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

3.7. *Objective Function* Untuk Permasalahan Penentuan Parameter Motor Induksi Tiga Phasa

Tujuan adalah untuk meminimalisasi kesalahan (*error*) antara pengukuran dan perhitungan parameter. Parameter input tersebut akan dimasukkan ke dalam *Objective function* yang dirumuskan sebagai berikut :

$$F_{objective} = \sum_{i=1}^{n} \left| \frac{I_{1i,Metode}}{I_{1i,ukur}} - 1 \right|^2 + \sum_{i=1}^{n} \left| \frac{P_{input\ i,Metode}}{P_{input\ i,ukur}} - 1 \right|^2 \quad \text{.....(3.9)}$$

Selanjutnya dicari error sebagai berikut :

$$\varepsilon = F_{Objective} \quad \text{.....(3.10)}$$

dan dicari *fitness*nya adalah :

$$Fitness = \frac{100}{100 + F_{objective}} \quad \text{.....(3.11)}$$

3.8. Algoritma

3.8.1. Algoritma Program Pemecahan Masalah Secara Umum.

1. Memasukkan input data berupa informasi Papan-nama (*name-plate*) dari motor induksi yang meliputi :
   - Tegangan nominal (V_{in}) dalam Volt
   - Arus nominal *(I)* dalam Ampere
   - Kecepatan putaran rotor nominal (rpm)
   - Jumlah kutub
2. Proses evaluasi parameter
   - Metode Genetika Algoritma
   - Metode Genetika programming
3. Menghitung nilai parameter yang dicari berupa R_s, R_r, X_m, X_s dan X_r.
4. Dengan data parameter hasil evaluasi digunakan untuk menentukan nilai torsi.
5. Cetak hasil
   - Table torsi
   - Grafik torsi

3.8.2. Algoritma Program Penentuan Parameter Motor Induksi Tiga-Phasa Menggunakan Metoda Genetika Algoritma

1. Memasukkan input data motor induksi
2. Menentukan parameter inputan Algoritma Genetika yang meliputi jumlah populasi, maksimum generasi, nilai-nilai kemungkinan
3. crossover, nilai kemungkinan mutasi dan panjang kromosom tiap-tiap individu
4. Generasi = 0; Populasi = 0
5. Initialisasi populasi
6. Evaluasi fitness populasi
7. Melakukan proses statistik
8. Melakukan proses seleksi
9. Melakukan proses crossover
10. Melakukan proses mutasi
11. Apakah offspring sudah mencapai nilai maksimum populasi (100)?
    - Jika Tidak Pop = Pop+1 "proses 7, 8, 9, 10 diulangi sampai offspring mencapai nilai maksimum (100).
    - Jika Ya " maka Proses dilanjutkan".
12. Menghitung fitness dari offspring
13. melakukan Proses statistik
14. melakukan proses elistim
15. Apakah generasi yang diinginkan sudah mencapai nilai Maks Gen (1000)
    - Jika "tidak" maka Gen = Gen + 1, kembali ke langkah 7
    - Jika "ya" maka perhitungan berhenti

3.8.3. Algoritma Penyelesaian Penentuaan Parameter menggunakan Metode *Genetika Programming*

Langkah-langkah Proses program komputer:

1. Memasukkan input data motor induksi
2. memasukan parameter inputan Genetika Programing
3. Generasi = 0; Populasi = 0
4. Initialisasi populasi
5. Evaluasi fitness populasi
6. Melakukan proses statistik
7. Melakukan proses seleksi
8. Melakukan proses mutasi
10. Apakah offspring sudah mencapai nilai maksimum populasi (100)?
    - Jika Tidak Pop = Pop+1 "proses 6, 7, 8, diulangi sampai offspring mencapai nilai maksimum (100).
    - Jika Ya " maka Proses dilanjutkan".
11. Terjadi proses kompetisi
12. Menghitung fitness dari offspring
13. melakukan Proses statistik
14. melakukan proses elistim
15. Apakah generasi yang diinginkan sudah mencapai nilai Maks Gen (1000)
    - Jika "tidak" maka Gen = Gen + 1, kembali ke langkah 6.
    - Jika "ya" maka perhitungan berhenti

3.8.4. Algoritma Program Fitness.

1. Memasukkan input data parameter
2. Memasukkan datanya ke persamaan Rangkaian Ekivalen
3. Memasukkan datanya ke persamaan Fungsi Objektif /$F_{objective}$
4. Memasukkan nilai fungsi objektif ke persamaan *fitness*
5. Cetak hasil

BAB IV
PENGUJIAN DAN ANALISIS

4.1. Pengujian Parameter Motor Induksi Tiga Phasa.

4.1.1. Alat-alat Yang Digunakan

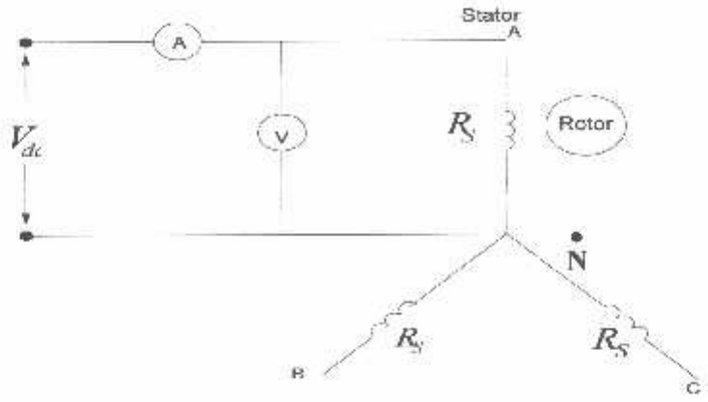
1. Motor Induksi Tiga-Phasa DE LORENZO / DL 1021

   Data papan-nama (*nameplate*) :

TEGANGAN	**: 220/380 (Δ / Y) VOLT**
ARUS	**: 4.3/2.5 (Δ / Y) AMPERE**
COS φ	**: 0,83**
FREKUENSI	**: 50 Hertz**
DAYA	**: 1.1 kW**
PUTARAN	**: 2820 rpm**
KUTUP	**: 2 KUTUP**
KELAS ISOLASI	**: F**

2. Voltmeter : DELORENZO DL 1031
3. Ampermeter : DELORENZO DL 1031
4. Wattmeter 3Φ : DELORENZO DL 1031
5. Tachometer : DELORENZO DL 2026
6. AC Voltage Regulator & DC Supply : DELORENZO DL 1013 M2

4.1.2. Pengujian Arus Searah *(DC – Test)*



4.1.3. Prosedur Pengujian Arus Searah (Dc Test)

Tujuan dari pengujian arus searah (*DC Test*) adalah untuk menentukan nilai resistansi stator (Rs).

Langkah Langkah Pengujian Arus Searah (Dc Test)

- Kumparan Stator dihubung bintang .
- Sumber tegangan dc dihubungkan dengan alat ukur (Volt meter dan Ampere meter)
- Sumber tegangan dc dari alat ukur dihubungkan pada kumparan stator yaitu pada titik **A** dan titik **N**
- Atur sumber tegangan dari tegangan yang paling kecil sampai arus yang terbaca pada alat ukur amper meter mendekati arus nominal dari data name-plate motor.
- Catat hasil pengukuran dari arus dan tegangan dari alat ukur untuk dianalisa.

Tabel 4 – 1

Data Hasil Pengujian Arus - Searah

Vdc (Volt)	I (Ampere)
1	0,19
2	0,37
3	0,55
4	0,72
5	0,91
6	1,09
7	1,27
8	1,45
9	1,63
10	1,80
11	1,98
12	2,12

4.1.4. Analisa dari pengujian arus searah, besarnya resistansi stator adalah :

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

$$R_{dc(1)} = \frac{1}{0,19} = 5,26\ \Omega$$

$$R_{dc(2)} = \frac{2}{0,37} = 5,40\ \Omega$$

$$R_{dc(3)} = \frac{3}{0,55} = 5,45\ \Omega$$

$$R_{dc(4)} = \frac{4}{0,72} = 5,55\ \Omega$$

$$R_{dc(5)} = \frac{5}{0,91} = 5,49\ \Omega$$

$$R_{dc(6)} = \frac{6}{1,09} = 5,50\Omega$$

$$R_{dc(7)} = \frac{7}{1,27} = 5,51\Omega$$

$$R_{dc(8)} = \frac{8}{1,45} = 5,51\Omega$$

$$R_{dc(9)} = \frac{9}{1,63} = 5,52\Omega$$

$$R_{dc(10)} = \frac{10}{1,80} = 5,55\Omega$$

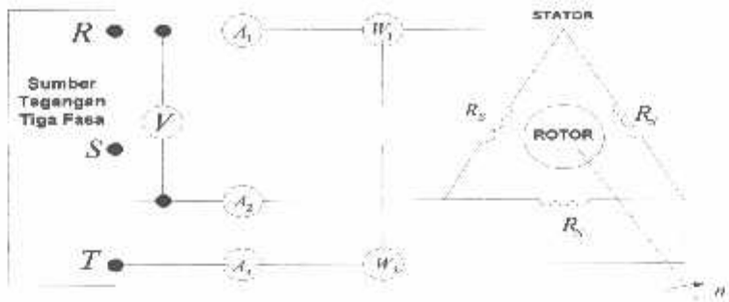
$$R_{dc(11)} = \frac{11}{1,98} = 5,55\ \Omega$$

$$R_{dc(12)} = \frac{12}{2,12} = 5,63\Omega$$

$$R_{.} = \frac{5,26 + 5,40 + 5,45 + 5,55 + 5,49 + 5,50 + 5,51 + 5,51 + 5,52 + 5,55 + 5,63}{12}$$

$$R_S = 5,49 Ohm\ /\ Phasa$$

4.2.1. Pengujian Tanpa Beban *(No – Load Test)*



4.2.2. P rosedur Pengujian Tanpa Beban (No-load Test)

- Sumber tegangan tiga phasa dihubungkan pada alat ukur (Volt meter, Amper meter dan watt meter)
- Sumber tegangan tiga phasa dari alat ukur dihubungkan pada kumparan stator.
- Kumparan stator motor dirangkai hubungan bintang .
- Atur tegangan tiga phasa jala-jala sampai tegangan nominal motor.
- Ukur kecepatan (rpm) dengan tachometer pada kecepatan nominal motor.
- Catat hasil pengukuran dari alat ukur (Arus, Tegangan dan Daya) untuk dianalisa.

Tabel 4 - 2
Data Hasil Pengujian Beban Nol

I (Ampere)			W 3Ø (Watt)	V (Volt)	F (Hz)
A_1	A_2	A_3			
0.59	0.68	0.60	90	220	50

4.2.3. Analisa Pengujian Beban Nol

$$I_{th} = \frac{I_{ta} + I_{tb} + I_{tc}}{3} = A \qquad : I_{tb} = \frac{0.59 + 0.68 + 0.59}{3} = 0.62A$$

$$Z_{tb} = \frac{V_0}{\sqrt{3} I_{th}} = \frac{220}{\sqrt{3X}\ 0.62} = 205\ \Omega$$

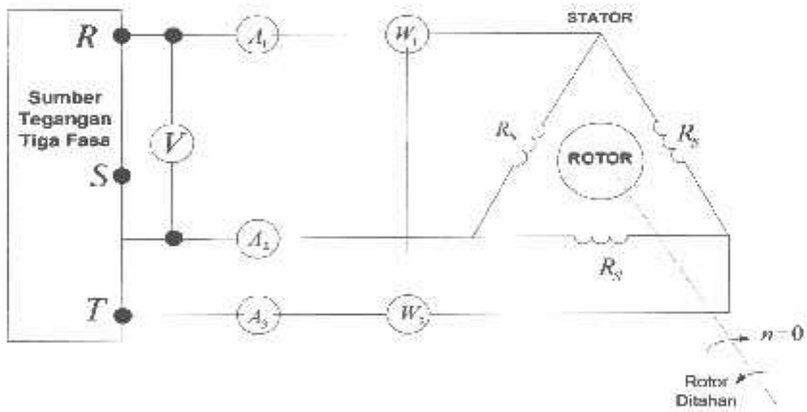
$$R_{tb} = \frac{P(3\Phi)}{3I_0} = \frac{90}{3.(0.62)^2} = 78\Omega$$

$$X_{tb} = \sqrt{Z_{th}^{\ 2} - R_{th}^{\ 2}}$$

$$= \sqrt{205^2 - 78^2}$$

$$= 189.58\ \Omega \text{ /Phasa}$$

4.3.1. Pengujian Rotor Tertahan *(Blocked-Rotor Test)*



4.3.2. P rosedur Pengujian Rotor Tertahan (Block – rotor Test)

- Sumber tegangan tiga phasa diatur dari tegangan kecil dan dihubungkan pada alat ukur (Volt meter, Amper meter dan watt meter)
- Sumber tegangan tiga phasa dari alat ukur dihubungkan pada kumparan stator motor induksi tiga phasa
- Kumparan stator motor dirangkai hubungan bintang .
- Motor induksi dikopel dengan Current – break
- Motor dalam keadaan berputar direm dengan injeksi tegangan Dc dari Curent break hingga motor berhenti dalam scsaat.
- Motor direm hingga tidak berputar penunjukan arus pada alat ukur usahakan mendekati arus nominal dari Data name-plate motor.
- Catat hasil pengukuran dari alat ukur (Arus, Tegangan dan Daya) untuk dianalisa.

Tabel 4 - 3
Data Hasil Rotor Tertahan

I (Amperc)			W 3Ø (Watt)	V (Volt)
A_1	A_2	A_3		
2.01	2.03	1.95	140	65

4.3.3. Analisa Pengujian Rotor Tertahan

$$I_{rt} = \frac{I_{la} + I_{lb} + I_{lc}}{3} = A \quad ; \qquad I_{rt} = \frac{2.01 + 2.03 + 1.97}{3} = 2.0A$$

$$Z_{rt} = \frac{V_{l-l}}{\sqrt{3}.I_{rt}} = \frac{65}{\sqrt{3}x2} = 18.76\Omega$$

$$R_{rt} = \frac{P(3\Phi)}{3.(I_{rt})^2} = \frac{140}{3.(2.0)^2} = 11,67\,\Omega$$

$$X_{rt} = \sqrt{Z_{rt}^{\ 2} - R_{rt}^{\ 2}}$$

$$= \sqrt{18.76^2 - 11,67^2}$$

$$= 14,6\ \Omega$$

$$X_{rt} = X_s + X'_r$$

Motor innduksi yang dipakai adalah motor induksi dengan rotor sangkar tunggal kelas A, maka secara umum Xs dan X'_r diasumsikan sama, sehingga :

$$X_s = X'_r = \frac{1}{2}X_{rt} = \frac{1}{2}(14,6) = 7,3\Omega$$

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

$$X_{ss} = X_{tb} = X_s + X_m$$

$$X_m = X_{tb} - X_s$$

$$= 189,58\text{-}7,3$$

$$= 182,28\ \Omega$$

Resistansi stator dapat dipandang sebagai harga Denya maka resistansi rotor dapat ditentukan sebagai berikut :

$$R = R_{rt} - R_s$$

$$= 11.67 - 5.49$$

$$- 6.18\ \Omega$$

$$X_{rr} = X'r + X_m$$

$$x_{rr} = 7.3 + 182.28$$

$$x_{rr} = 189{,}58\Omega$$

$$R'_r = R\left(\frac{X_{rr}}{X'_m}\right)^2$$

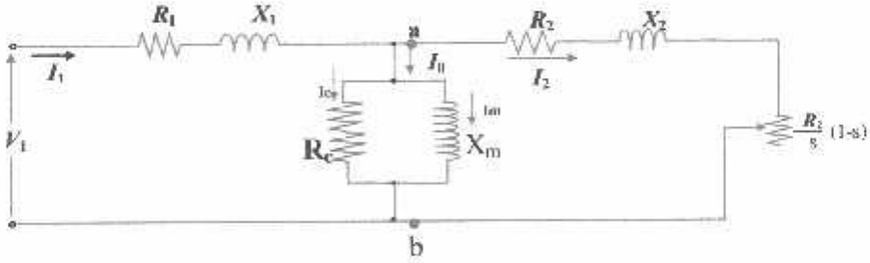
$$= 6.18\left(\frac{189{,}58}{182.28}\right)^2 = 6{,}68\ \Omega$$

Tabel 4 - 4

Hasil Perhitungan Pengujian Parameter Motor induksi Tiga Phasa

R_s	R'_r	X_s	X'_r	X_m
5.49	6.68	7.3	7.3	182.28

- **Rangakaian Ekivalen Motor Induksi**



Ket : *Rc diabaikasn*

4.4.1. Menghitung Torsi Mula, Torsi Beban Penuh.

$V_s = 220$ Volt/phasa

$$n_s = \frac{120.f}{p} = \frac{120.50}{2} = 3000\,rpm$$

$$S = \frac{n_s - n_r}{n_s} = \frac{3000 - 2820}{3000} = \frac{180}{3000} = 0{,}06$$

$$\omega_s = \frac{120.f}{p.60}.2\pi = \frac{120.50}{2.60}.6{,}28 = 314 \text{ rad / s}$$

$$V_1 = \frac{220}{\sqrt{3}} = 127 \text{ volt/phasa}$$

$$V_{ab} = \frac{JX_m}{R_s + JX_s + JX_m} V_s$$

$$= \frac{J182.28}{5.49 + J7.3 + j182.28} 127\angle 0^0$$

$$= \frac{182.28\angle 0^0}{189\angle 88.34^0} 127$$

$$= 122{,}0\ \angle -88.34^0$$

$$Z_{ab} = \frac{JX_m (R_s + JX_s)}{JX_m + R_s + JX_s} = \frac{j182.28(5.49 + J7.3)}{J182.28 + 5.49 + J7.3}$$

$$= \frac{1670\angle 36.726^0}{189.659\angle 88.344^0}$$

$$= 8.8\angle -51.618^0$$

$$= 5{,}46 + J6{,}8 \text{ ohm}$$

jadi $Z_{ab} = R_{ab} + X_{ab}$ $= 5{,}46 + 6{,}8$

1. Menghitung Torsi Mula

n_r = 0 rpm , S = 1

$$T = \frac{3}{\omega_s} \frac{(V_{ab})^2.}{(R_{ab} + R'_r/S)^2 + (X_{ab} + X'_r)^2} \frac{R'_r}{S}$$

$$= \frac{3}{314} \frac{(122)^2.}{(5,46 + 6,68/1)^2 + (6,8 + 7.3)^2} \frac{6.68}{1}$$

$$= 0,00955 \cdot \frac{14884}{(5,46 + 6,68)^2 + (6,8 + 7.3)^2} 6,68$$

$$= 0,00955 \cdot \frac{14884}{346,1896} 6.68$$

$$= 0,00955 \text{ X } 42.99378 \text{ X } 6,68$$

$$= 2,743 \text{ N.m}$$

2. Menghitung Torsi Keadaan Beban Penuh

n_r = 2820 rpm, S = 0,06

$$T = \frac{3}{\omega_s} \frac{(V_{ab})^2.}{(R_{ab} + R'_r/S)^2 + (X_{ab} + X'_r)^2} \frac{R'_r}{S}$$

$$= \frac{3}{314} \frac{(122)^2.}{\left(5,46 + \frac{6,68}{0,06}\right)^2 + (6.8 + 7.3)^2} \frac{6,68}{0,06}$$

$$= 0,00955 \cdot \frac{14884}{\left(5,46 + \frac{6,68}{0,06}\right)^2 + (6,8 + 7.3)^2} 111,33$$

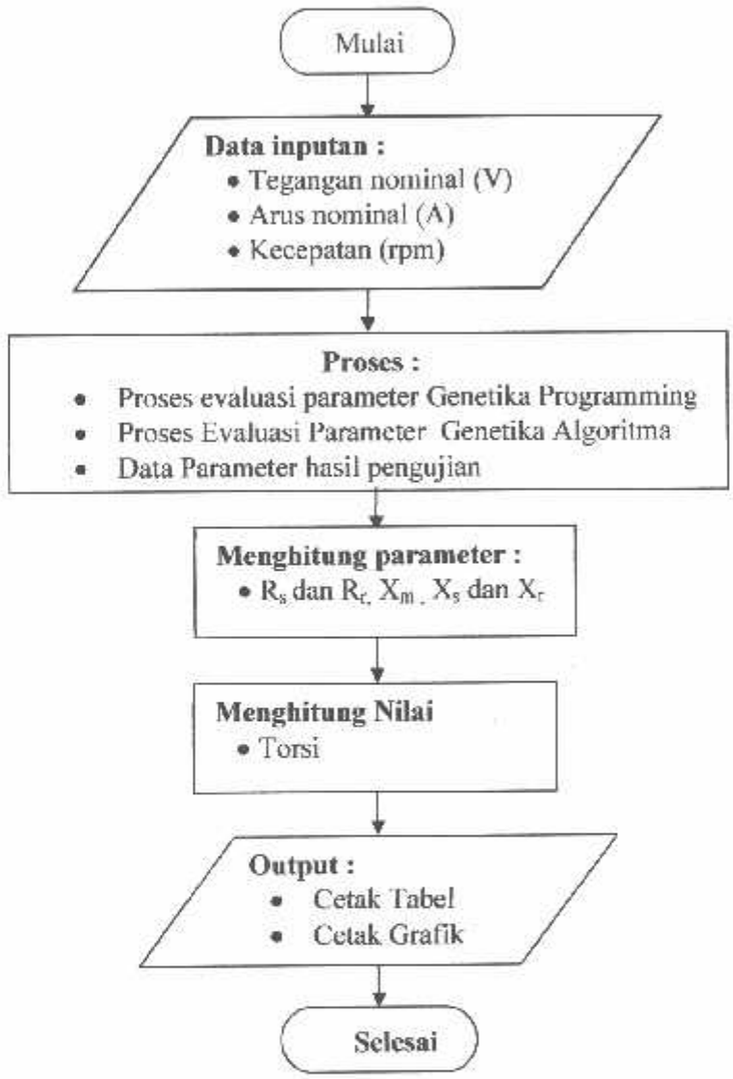
$$= 0,00955 \cdot \frac{14884}{13839,49271} 111,33$$

$$= 0,00955 \text{ X } 1,075472946 \text{ X } 111.33$$

$$= 1,144 \text{ N.m}$$

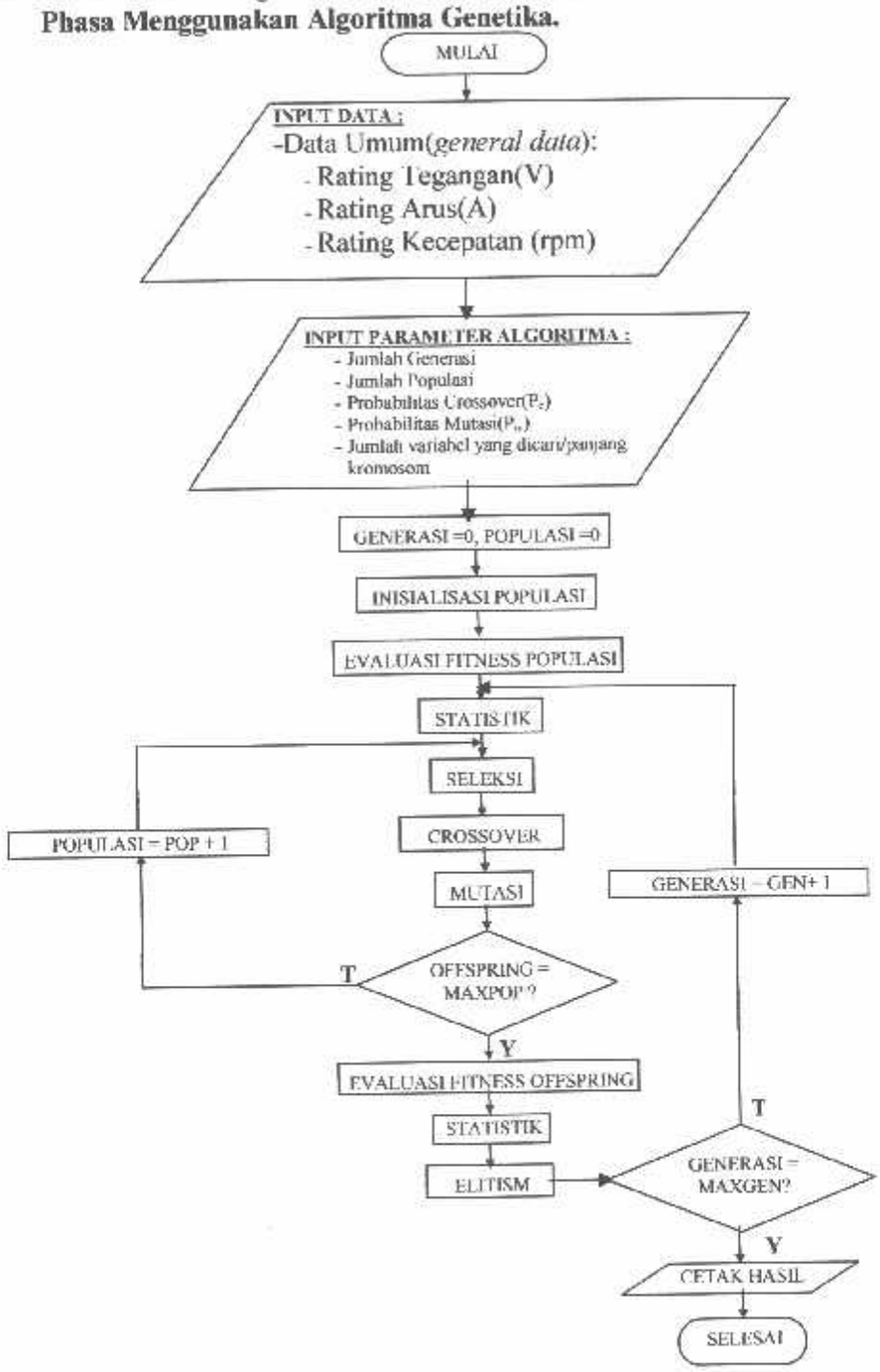
4.5. Flowchart Algoritma

4.5.1. Flowchart Pemecahan Masalah Secara Umum



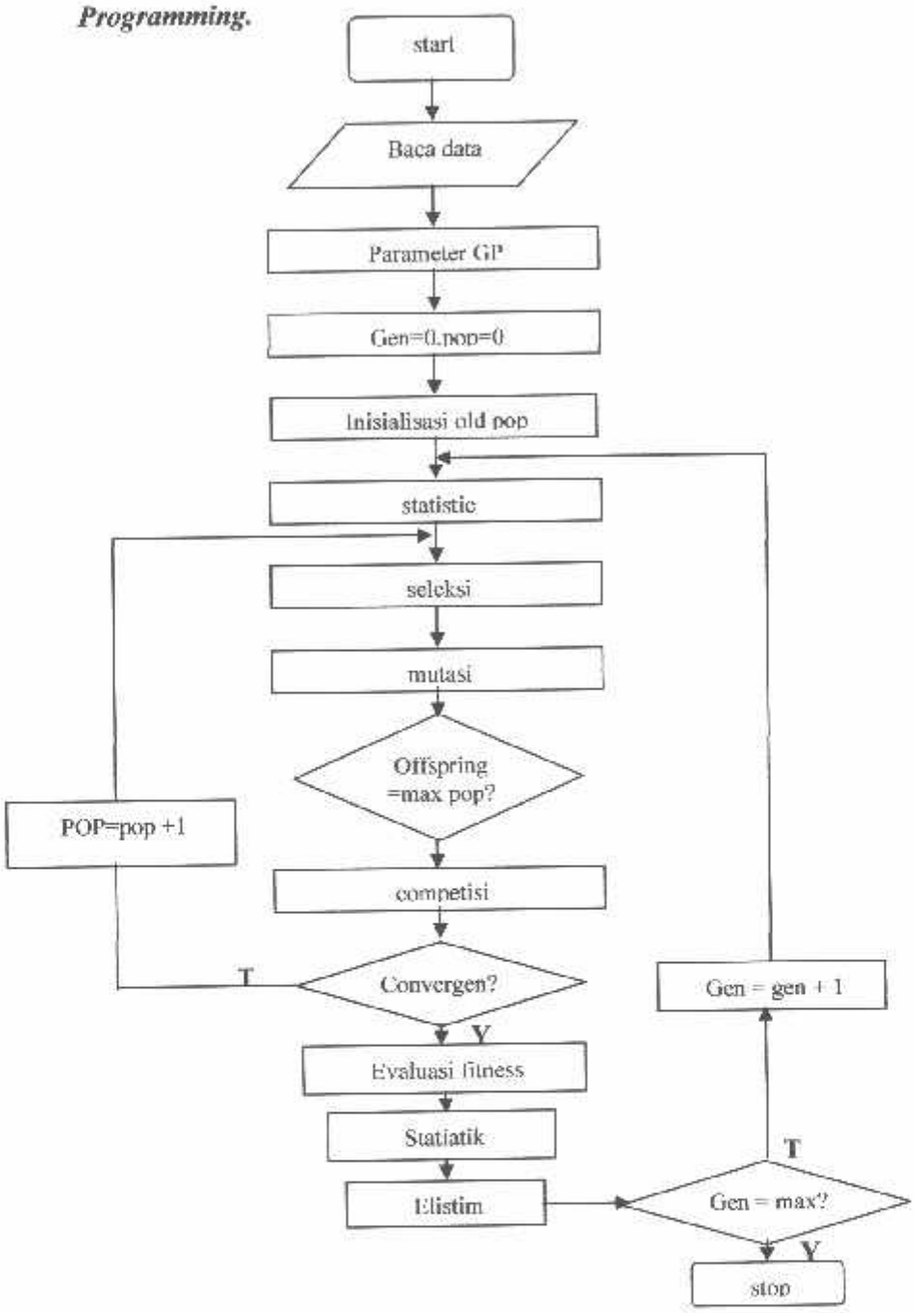
Gambar 4-1: Flowchart Program Algoritma Genetika

4.5.2. Flow Chart Program Penentuan parameter Motor Induksi Tiga - Phasa Menggunakan Algoritma Genetika.



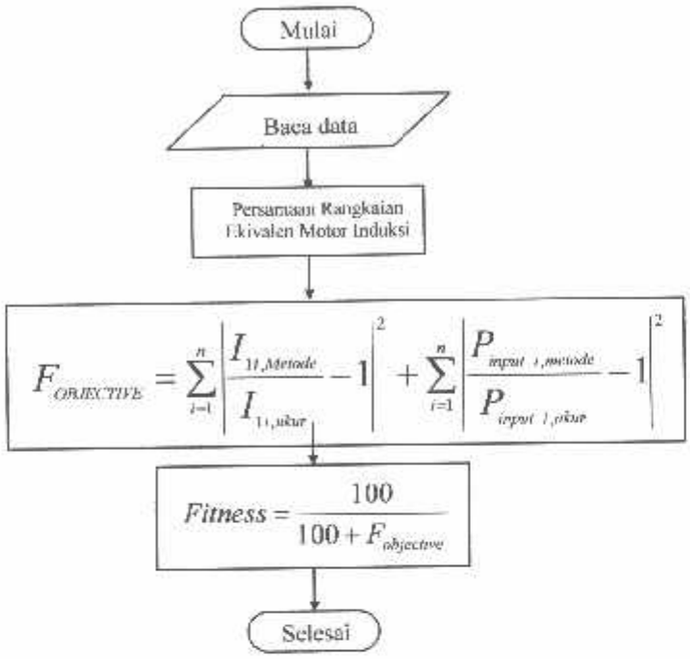
Gambar 4-2 : Flowchart Program genetika Algoritma.

4.5.3. Flowchart Penentuan Parameter menggunakan metode *Algoritma Programming.*



Gambar 4-3 : Flowchart Program Genetika Programing

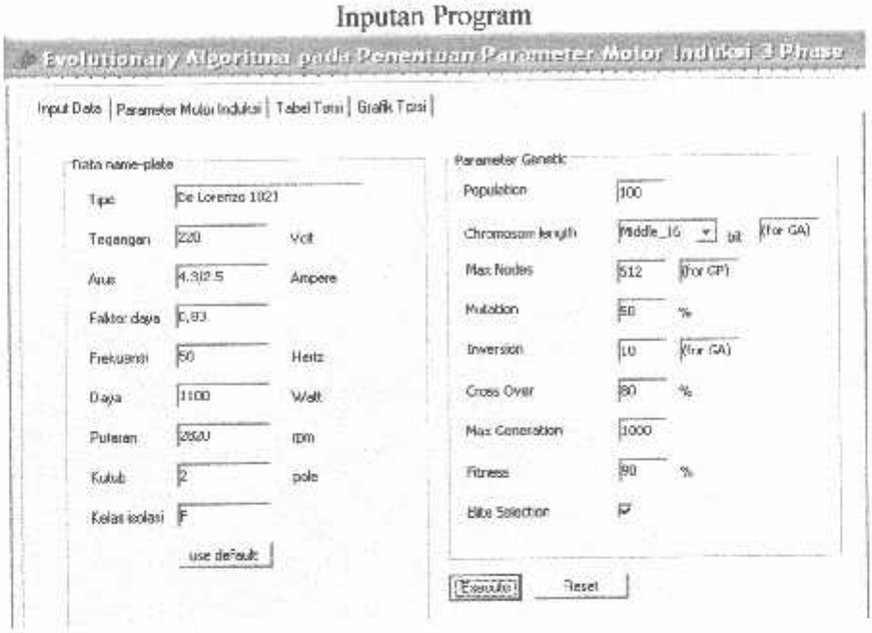
4.5.4. Flowchart Program Fitness.



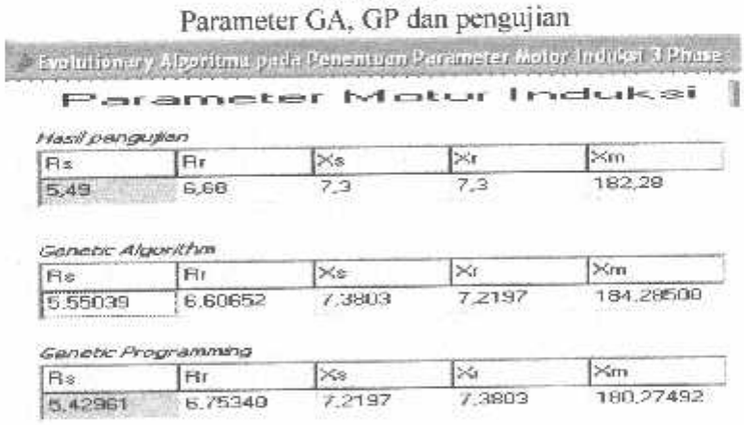
Gambar 4-4: Flowchart Program Fitness

4.5.5. TAMPILAN PROGRAM

Tabel 4-5
Inputan Program



Tabel 4-6
Parameter GA, GP dan pengujian



Tabel 4-7
Torsi Hasil Pengujian

Tabel Torsi

Evolutionary Algoritma pada Penentuan Parameter Motor Induksi 3 Phase

No	Slip	Kec(rpm)	Torsi(Nm)
1	1	0	2,738158080072
2	0,99	30	2,753210417610
3	0,98	60	2,7693550001676
4	0,97	90	2,7836899354252
5	0,96	120	2,7989128884537
6	0,95	150	2,8143214584025
7	0,94	180	2,8298129687143
8	0,93	210	2,8453844905047
9	0,92	240	2,8610328259476
10	0,91	270	2,8767544986029
11	0,9	300	2,8925456946180
12	0,89	330	2,9084023227307

No	Slip	Kec(rpm)	Torsi(Nm)
13	0,88	360	2,9243199129970
14	0,87	390	2,9402936941620
15	0,86	420	2,9563182615860
16	0,85	450	2,9723881516348
17	0,84	480	2,9884972144353
18	0,83	510	3,0046380040916
19	0,82	540	3,0209060918525
20	0,81	570	3,0369912253116
21	0,8	600	3,0531861015167
22	0,79	630	3,0693819258563
23	0,78	660	3,0855692530043
24	0,77	690	3,1017379460348

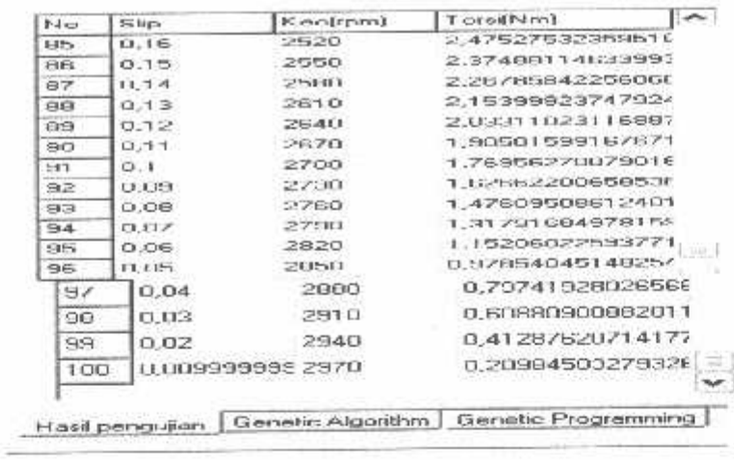
No	Slip	Kec(rpm)	Torsi(Nm)
25	0,76	720	3,1178771319717
26	0,75	750	3,1339751501096
27	0,74	780	3,1500195076309
28	0,73	810	3,1659968223188
29	0,72	840	3,1818927653131
30	0,71	870	3,1976920054881
31	0,7	900	3,2133781350429
32	0,69	930	3,2289336110818
33	0,68	960	3,2443396779575
34	0,67	990	3,2595762911362
35	0,66	1020	3,2746220353386
36	0,65	1050	3,2894540377002

No	Slip	Kec(rpm)	Torsi(Nm)
37	0,64	1080	3,3040478756871
38	0,63	1110	3,31037747949846
39	0,62	1140	3,3324150286802
40	0,61	1170	3,3461004267211
41	0,6	1200	3,35949326500761
42	0,59	1230	3,37246854000995
43	0,58	1260	3,38502068706926
44	0,57	1290	3,39711135935515
45	0,56	1320	3,40869970071324
46	0,55	1350	3,41974219770003
47	0,54	1380	3,43019251709094
48	0,53	1410	3,44000133013019

No	Slip	Kec(rpm)	Torsi(Nm)
49	0,52	1440	3,44911618200218
50	0,51	1470	3,45748121786025
51	0,5	1500	3,46503708058454
52	0,49	1530	3,47172066585411
53	0,48	1560	3,47746492165396
54	0,47	1590	3,48219863295200
55	0,46	1620	3,48584619744376
56	0,45	1650	3,48832739400762
57	0,44	1680	3,48955715084921
58	0,43	1710	3,48944520056444
59	0,42	1740	3,48709630897036
60	0,41	1770	3,48400909152143

No	Slip	Kec(rpm)	Torsi(Nm)
61	0,4	1800	3,48007669488883
62	0,39	1830	3,47350608401135
63	0,38	1860	3,46521789100746
64	0,37	1890	3,45484617932016
65	0,36	1920	3,44233821247376
66	0,35	1950	3,42755425266525
67	0,34	1980	3,41034735925784
68	0,33	2010	3,39056327758372
69	0,32	2040	3,3680400536139
70	0,31	2070	3,34260843076396
71	0,3	2100	3,31409134209866
72	0,29	2130	3,28230408962090

No	Slip	Kec(rpm)	Torsi(Nm)
73	0,28	2160	3,24705444904004
74	0,27	2190	3,20814278920096
75	0,26	2220	3,16536230153207
76	0,25	2250	3,11849940127703
77	0,24	2280	3,06733414223878
78	0,23	2310	3,01164084984563
79	0,22	2340	2,95110007149627
80	0,21	2370	2,88574050188336
81	0,2	2400	2,81500709507056
82	0,19	2430	2,73892038106 5
83	0,18	2460	2,65706400421930
84	0,17	2490	2,56826029982063



No	Slip	Kec(rpm)	Torsi(Nm)
85	0,16	2520	2,475275323859615
86	0,15	2550	2,374881141633993
87	0,14	2580	2,267858422560065
88	0,13	2610	2,153999237479245
89	0,12	2640	2,033110231168875
90	0,11	2670	1,905015991674671
91	0,1	2700	1,769562700790165
92	0,09	2730	1,626862200660539
93	0,08	2760	1,476095086124015
94	0,07	2790	1,317916849781655
95	0,06	2820	1,152060279393771
96	0,05	2850	0,978540451482574
97	0,04	2880	0,797419280285668
98	0,03	2910	0,608880900882011
99	0,02	2940	0,412876207141775
100	0,009999999	2970	0,209045002793285

Hasil pengujian | Genetic Algorithm | Genetic Programming

Tabel 4-8
Torsi Hasil Proses Genetika Algoritma

Tabel Torsi

Evolutionary Algoritma pada Penentuan Parameter Motor Induksi 3 Phase

No	Slip	Kec(rpm)	Torsi(Nm)
1	1	0	2,711816517254365
2	0,99	30	2,726662279941915
3	0,98	60	2,741804032410335
4	0,97	90	2,757033938050355
5	0,96	120	2,772367951046015
6	0,95	150	2,787786022836685
7	0,94	180	2,803291067603585
8	0,93	210	2,818880957405455
9	0,92	240	2,834552502914235
10	0,91	270	2,850302436971455
11	0,9	300	2,866127719562325
12	0,89	330	2,882022898523385

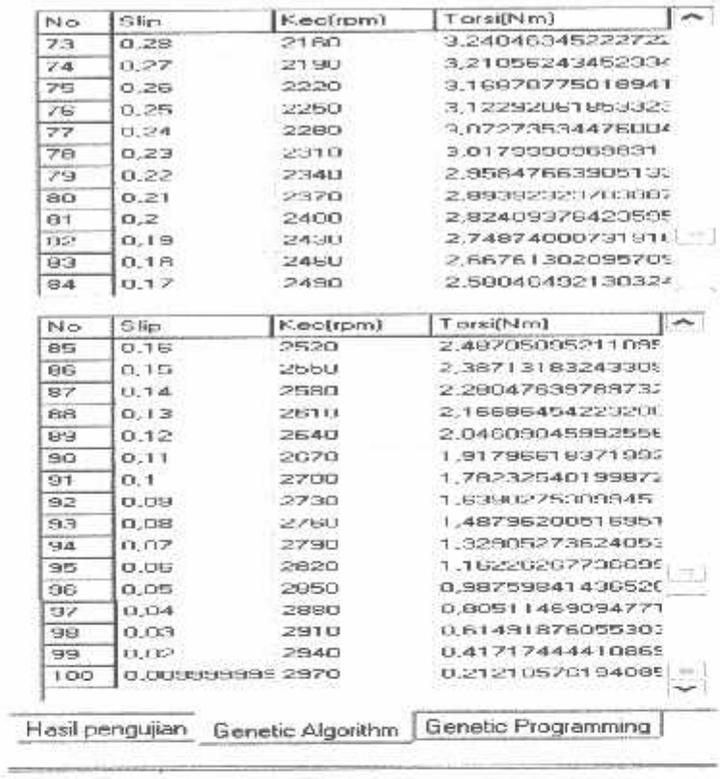
No	Slip	Kec(rpm)	Torsi(Nm)
13	0,88	360	2,897985327970895
14	0,87	390	2,914009906596255
15	0,86	420	2,930091673605785
16	0,85	450	2,946225259491275
17	0,84	480	2,962404859106365
18	0,83	510	2,978624203800225
19	0,82	540	2,994876526912145
20	0,81	570	3,011154539206365
21	0,8	600	3,027450386583025
22	0,79	630	3,043755615726985
23	0,78	660	3,060061139589555
24	0,77	690	3,076357186402025

No	Slip	Kec(rpm)	Torsi(Nm)
25	0,76	720	3,09263326068113
26	0,75	750	3,10887810017333
27	0,74	780	3,12507961736033
28	0,73	810	3,14122485058249
29	0,72	840	3,15729990610587
30	0,71	870	3,17328989764155
31	0,7	900	3,18917888189743
32	0,69	930	3,20494979010124
33	0,68	960	3,22058435519953
34	0,67	990	3,23608630344888
35	0,66	1020	3,25136492741885
36	0,65	1050	3,26640176882976

No	Slip	Kec(rpm)	Tnrsi(Nm)
37	0,64	1080	3,28134749361764
38	0,63	1110	3,29597864371681
39	0,62	1140	3,31033405647798
40	0,61	1170	3,32439455676752
41	0,6	1200	3,33809808930633
42	0,59	1230	3,35144445467448
43	0,58	1260	3,36438526813
44	0,57	1290	3,37686377298887
45	0,56	1320	3,38889970416597
46	0,55	1350	3,40039013378014
47	0,54	1380	3,41130930830406
48	0,53	1410	3,42160847777030

No	Slip	Kec(rpm)	Torsi(Nm)
49	0,52	1440	3,43123571492929
50	0,51	1470	3,44013572730067
51	0,5	1500	3,44824965918095
52	0,49	1530	3,45551486476511
53	0,48	1560	3,46186470203984
54	0,47	1590	3,46722856296508
55	0,46	1620	3,47153093192206
56	0,45	1650	3,47469195438825
57	0,44	1680	3,47662675467027
58	0,43	1710	3,47724527280306
59	0,42	1740	3,47645200520109
60	0,41	1770	3,47414574141237

No	Slip	Kec(rpm)	Torsi(Nm)
61	0,4	1800	3,47021929070084
62	0,39	1830	3,46455925652667
63	0,38	1860	3,45704509346171
64	0,37	1890	3,44755192954905
65	0,36	1920	3,43594427767477
66	0,35	1950	3,42208180814065
67	0,34	1980	3,40581613132406
68	0,33	2010	3,38699120407911
69	0,32	2040	3,36544316038774
70	0,31	2070	3,34100021569727
71	0,3	2100	3,31346252738935
72	0,29	2130	3,28270223090062



No	Slip	Kec(rpm)	Torsi(Nm)
73	0,28	2160	3,240463452227
74	0,27	2190	3,2105624345230
75	0,26	2220	3,16970775018941
76	0,25	2250	3,122992061853323
77	0,24	2280	3,07273534476004
78	0,23	2310	3,0179990069831
79	0,22	2340	2,9584766390513
80	0,21	2370	2,89389282176088
81	0,2	2400	2,82409376423595
82	0,19	2430	2,74874000731911
83	0,18	2460	2,66761302095705
84	0,17	2490	2,58004092130324

No	Slip	Kec(rpm)	Torsi(Nm)
85	0,16	2520	2,487050052110
86	0,15	2550	2,387131832493
87	0,14	2580	2,28047638788732
88	0,13	2610	2,166864542302
89	0,12	2640	2,04609045992556
90	0,11	2670	1,91796618371992
91	0,1	2700	1,78232540199872
92	0,09	2730	1,63902753099845
93	0,08	2760	1,48796200516951
94	0,07	2790	1,32905273624052
95	0,06	2820	1,16220207730699
96	0,05	2950	0,98759841436520
97	0,04	2880	0,80511469094771
98	0,03	2910	0,61491876055303
99	0,02	2940	0,41717444410865
100	0,009999999	2970	0,21210570194085

Hasil pengujian | Genetic Algorithm | Genetic Programming

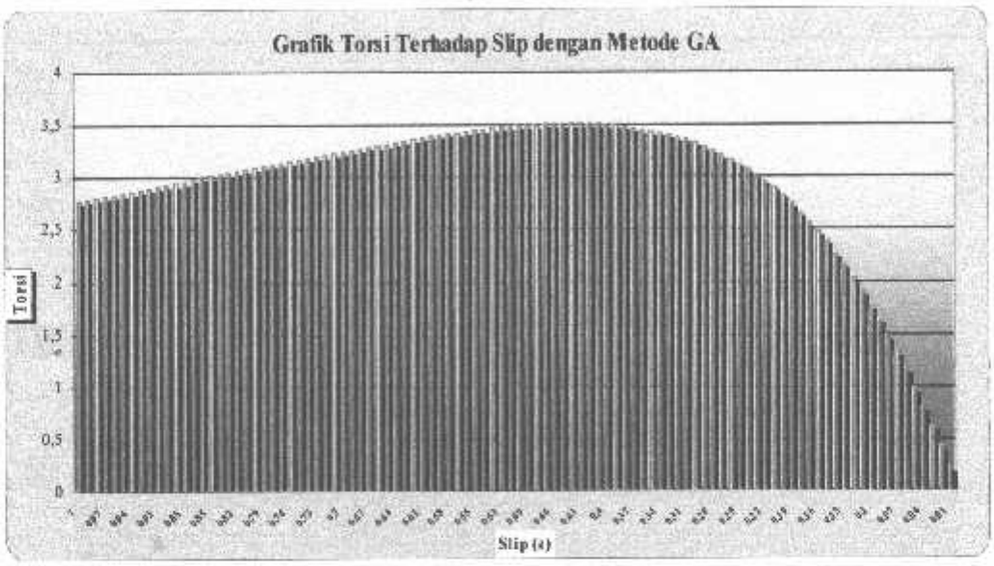
Tabel 4-9
Torsi Hasil Proses Genetika Programing

Tabel Torsi

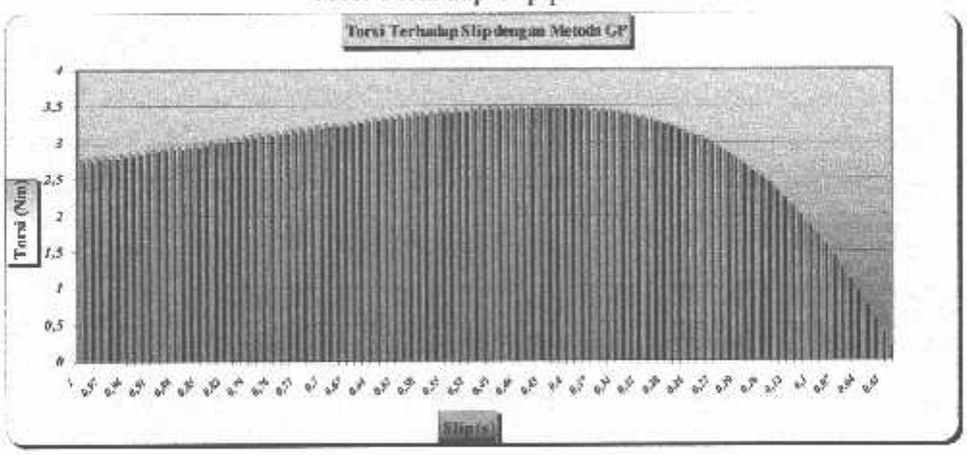
Evolutionary Algorithm pada Penentuan Parameter Motor Induksi 3 Phase

No	Slip	Kec(rpm)	Torsi(Nm)
1	1	0	2,7646248740412
2	0,99	30	2,779680435734
3	0,98	60	2,7948246997414
4	0,97	90	2,8100552041503
5	0,96	120	2,8253697918615
6	0,95	150	2,8407655965222
7	0,94	180	2,8562398275721
8	0,93	210	2,8717900543511
9	0,92	240	2,8874107892050
10	0,91	270	2,9031003695286
11	0,9	300	2,9188541386770
12	0,89	330	2,9346677256744

Grafik 4-2
Torsi Terhadap Slip Pada GA



Grafik 4-3
Torsi Terhadap Slip pada GP



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

LEMBAR BIMBINGAN SKRIPSI

Nama Mahasiswa : **Herman Yosef S. M**

N.I.M. : **01. 12. 074**

Jurusan : **Teknik Elektro S-1**

Kosentrasi : **Teknik Energi Listrik**

Judul Skripsi :

PENENTUAN PARAMETER MOTOR INDUKSI TIGA PHASA DENGAN METODE *EVOLUTIONARY ALGORITMA*

Tanggal Mengajukan Skripsi : **16 Juni 2006**

Tanggal Menyelesaikan Skripsi : **22 September 2006**

Dosen Pembimbing : **Ir. M. Abdul Hamid, MT**

Telah di Evaluasi Dengan Nilai : **85 (Delapan Puluh Lima)**

Malang 07 Oktober 2006

Mengetahui,
Ketua Jurusan Teknik Elektro

Ir. F. Yudi Limpraptono, MT
NIP. Y. 103 9500 274

Diperiksa dan disetujui,
Dosen Pembimbing

Ir. M. Abdul Hamid, MT
NIP. Y. 101 8800 188

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

BERITA ACARA UJIAN SKRIPSI

Nama : **HERMAN YOSEF S. MBORO**
N.I.M : 01.12.074
Jurusan : TEKNIK ELEKTRO
Konsentrasi : ENERGI LISTRIK
Judul Skripsi : **PENENTUAN PARAMETER MOTOR INDUKSI TIGA PHASA DENGAN METODE *EVOLUTIONARY ALGORITMHS***

Dipertahankan dihadapan majelis penguji jenjang strata satu (S-1),

Hari : Jum'at
Tanggal: 22 September 2006
Nilai : 79 (B+)

Panitia Ujian Skripsi

Ketua	**Sekertaris**
Ir. Mochtar Asroni, MSME	**Ir. F. Yudi Limpraptono, MT**

Anggota Penguji

Irrine Budi S, ST, MT	**Ir. H. Taufik Hidayat, MT**
Penguji Pertama	**Penguji kedua**

Institut Teknologi Nasional Malang
Fakultas Teknologi Industri
Jurusan Teknik Elektro S-1

LEMBAR PENGAJUAN JUDUL SKRIPSI
JURUSAN TEKNIK ELEKTRO S-1

Konsentrasi : Teknik Energi Listrik/Teknik Elektronika *)

1	Nama Mahasiswa : HERMAN YOSEF S. M		Nim : 01.12.074.	
2	Waktu pengajuan	Tanggal :	Bulan :	Tahun :
3	Spesifikasi judul (berilah tanda silang)			
	a. Sistem Tenaga Elektrik b. Energi & Konversi Energi c. Tegangan Tinggi & Pengukuran d. Sistem Kendali Industri		e. Elektronika & Komponen f. Elektronika Digital & Komputer g. Elektronika Komunikasi h. lainya	
4	Konsultasikan judul sesuai materi bidang ilmu kepada Dosen *) : Ir. Hj. Abdul Hamid, MT		Mengetahui, Ketua Jurusan. Ir. F. Yudi Limpraptono, MT Nip. Y. 1039500274	
5	Judul yang diajukan mahasiswa :	PENENTUAN PARAMETER MOTOR INDUKSI TIGA PHASA DENGAN MENGGUNAKAN METODE EVOLUTIONARY ALGORITMA		
6	Perubahan Judul yang disetujui Dosen sesuai materi bidang ilmu	.....		
7	Catatan :			
	Persetujuan Judul Skripsi yang dikonsultasikan kepada Dosen materi bidang ilmu		Disetujui, 23-5-, 200.6. Dosen Ir. ABDUL HAMID, MT	

Perhatian :
1. Formulir Pengajuan ini harap dikembalikan kepada jurusan paling lambat satu minggu setelah disetujui kelompok dosen keahlian dengan dilampirkan proposal skripsi beserta persyaratan skripsi sesuai form S-1
2. Keterangan : *) coret yang tidak perlu
   **) dilingkari a, b, c, ... atau g.sesuai bidang keahlian

Form.S-2

INSTITUT TEKNOLOGI NASIONAL
JL. Bendungan sigura-gura No.2
MALANG

Lampiran : 1 (satu) berkas
Pembimbing skripsi

Kepada : Yth. Ir. M. Abdul Hamid, MT
Dosen Institut teknologi Nasional
MALANG

Yang bertanda tangan dibawah ini :

Nama	: Herman Yosef S.M
Nim	: 01.12.074
Jurusan	: Teknik Elektro S-1
Konsentrasi	: Teknik Energi Listrik

Dengan ini mengajukan permohonan kiranya bapak / ibu bersedia menjadi dosen pembimbing utama / pendamping *), untuk penyusunan skripsi dengan judul (proposal terlampir) :

PENENTUAN PARAMETER MOTOR INDUKSI 3 PHASA DENGAN METODE EVOLUTIONARY ALGORITMA

Adapun tugas tersebut sebagai salah satu syarat untuk menempu ujian Akhir Serjana Teknik.

Demikian permohonan kami dan atas kesediaan Bapak / ibu kami ucapkan terima kasih.

Malang, Juni 2006

Ketua
Jurusan Teknik Elektro S-1

Ir, F. Yudi Limpraptono, MT
NIP. 1039500274

Hormat kami,

Herman Yosef S.M
Nim: 01.12.074

*) Coret yang tidak perlu

Form S-3a

Institut Teknologi Nasional Malang
Fakultas Teknologi Industri
Jurusan Teknik Elektro S-1

BERITA ACARA SEMINAR SKRIPSI
JURUSAN TEKNIK ELEKTRO S-1

Konsentrasi : Teknik Energi Listrik/~~Teknik Elektronika~~*)

1.	Nama Mahasiswa: HERMAN YOSEP, S.M			Nim: 01.12.074
2.	Keterangan Pelaksanaan	Tanggal: 13/9 '06	Waktu:	Tempat: Ruang:
3.	Spesifikasi Judul (berilah tanda silang)**) a. Sistem Tenaga Elektrik b. Energi & Konversi Energi c. Tegangan Tinggi & Pengukuran d. Sistem Kendali Industri e. Elektronika & Komponen f. Elektronika Digital & Komputer g. Elektronika Komunikasi h. lainnya			
4.	Judul Proposal yang diseminarkan Mahasiswa	PENENTUAN PARAMETER MOTOR INDUKSI 3Ø DENGAN METODE EVOLUTIONARY ALGORITMA		
5.	Perubahan Judul yang diusulkan oleh Kelompok Dosen Keahlian/Pengamat	.....		
6.	Keputusan: Dari hasil penilaian sejumlah .....orang dosen keahlian dan .....orang dosen pengamat sesuai format penilaian terlampir, peserta seminar tersebut diatas (1) dengan judul skripsi (4) dinyatakan LULUS/TIDAK LULUS *) dengan nilai Komulatif: .....(angka) atau .....(huruf)			
7.	Persetujuan Judul Skripsi			
	Disetujui, Dosen Keahlian I		Disetujui, Dosen Keahlian II	
	Disetujui, Dosen Pengamat I		Disetujui, Dosen Pengamat II	
	Mengetahui, Ketua Jurusan. Ir. F. Yudi Limpraptono,MT NIP. P. 1039500274		Disetujui, Dosen Pembimbing (Ir. M ABDUL HAMID, MT)	

Perhatian:
1. Keterangan: *) Coret yang tidak perlu
   **) dilingkari a, b, c, ...atau g sesuai bidang keahlian

Form S-4c

```
unit uMain;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, TeeProcs, TeEngine, Chart, ComCtrls, Gauges, StdCtrls,
  Grids, Series, JvComponentBase,GPUnit, genetic;
type
  TTargetFunction = function(Rs,Rr,Xs,Xr,Xm: double):double;// of object;
  Tform_main = class(TForm)
    page_main: TPageControl;
    tab_math: TTabSheet;
    tab_torsi: TTabSheet;
    bvl_main: TBevel;
    tab_plate: TTabSheet;
    pnl_plate: TPanel;
    Label12: TLabel;
    grid_ga: TStringGrid;
    Label17: TLabel;
    grid_gp: TStringGrid;
    grid_uji: TStringGrid;
    Label18: TLabel;
    TabSheet1: TTabSheet;
    PageControl1: TPageControl;
    TabSheet2: TTabSheet;
    TabSheet3: TTabSheet;
    TabSheet4: TTabSheet;
    grid_torsi_uji: TStringGrid;
    GroupBox1: TGroupBox;
    Label19: TLabel;
    Label20: TLabel;
    Label21: TLabel;
    GroupBox3: TGroupBox;
    ed_tipe: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    ed_volt: TEdit;
    Label10: TLabel;
    Label4: TLabel;
    ed_ampere: TEdit;
    Label11: TLabel;
    Label3: TLabel;
    ed_pf: TEdit;
    Label6: TLabel;
    ed_hertz: TEdit;
    Label13: TLabel;
    Label5: TLabel;
```

ed_watt: TEdit;
Label14: TLabel;
Label7: TLabel;
ed_rpm: TEdit;
Label15: TLabel;
Label8: TLabel;
ed_pole: TEdit;
Label16: TLabel;
Label9: TLabel;
ed_isolasi: TEdit;
gbx_genetic: TGroupBox;
ed_populate: TEdit;
ed_mutate: TEdit;
ed_cross: TEdit;
ed_maxgen: TEdit;
Label25: TLabel;
Label26: TLabel;
Label27: TLabel;
cbx_elite: TCheckBox;
Label28: TLabel;
Label29: TLabel;
btn_reset: TButton;
btn_ok: TButton;
Label30: TLabel;

ed_fitness: TEdit;
Label31: TLabel;
Label32: TLabel;
Label33: TLabel;
btn_default: TButton;
Label34: TLabel;
ed_maxnodes: TEdit;
Label35: TLabel;
grid_torsi_ga: TStringGrid;
grid_torsi_gp: TStringGrid;
combo_bit: TComboBox;
gen_alg: TGeneticAlgorithm;
gen_prog: TGPopulation;
Label36: TLabel;
ed_inversion: TEdit;
Chart1: TChart;
Series1: TFastLineSeries;
Series3: TFastLineSeries;
Series4: TFastLineSeries;
Shape1: TShape;
Shape2: TShape;

```
    Shape3: TShape;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    procedure btn_okClick(Sender: TObject);
    procedure btn_resetClick(Sender: TObject);
    procedure btn_defaultClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);

    function Torque(Rs,Rr,Xs,Xr,Xm,S:double):double;
    function ga_GetSuitability(
      Chromosome: TChromosome): Double;
    function  Pmek(Ir,Rr,S:Double):Double;
    function Ir(Vab,Rab,Rr,S,Xab,Xr:Double):Double;
    function Pin(Ir,Rr,S:Double):Double;
    function Pout(Pin,Ir,Rr,S:Double):Double;
  private
    { Private declarations }
    fTarget : TTargetFunction;
  public
    { Public declarations }
    StopFlag : boolean;
    property Target : TTargetFunction read fTarget write fTarget;
    procedure OneEpoch;

  end;

var
  form_main: Tform_main;
  Rab,Xab,Vab,Ws,S : Double;
  V1,Vs,ns,nr,p,f,pf: Double;
  Rs,Rr,Xs,Xr,Xm:double ;
  Rs_uji,Rr_uji,Xs_uji,Xr_uji,Xm_uji :double ;
  Rs_ga,Rr_ga,Xs_ga,Xr_ga,Xm_ga :double ;
  Rs_gp,Rr_gp,Xs_gp,Xr_gp,Xm_gp :double ;
implementation

{$R *.dfm}
procedure Tform_main.OneEpoch;
begin
  gen_alg.OneEpoch;
end;

function Tform_main.ga_GetSuitability(Chromosome: TChromosome): Double;
var
  Rs,Rr,Xs,Xr,Xm : double;
```

```
begin
 Rs := Chromosome.GeneAsFloat[0];
 Rr := Chromosome.GeneAsFloat[1];
 Xs := Chromosome.GeneAsFloat[2];
 Xr := Chromosome.GeneAsFloat[3];
 Xm := Chromosome.GeneAsFloat[4];
 Result := Target(Rs,Rr,Xs,Xr,Xm);
end;

function Tform_main.Torque(Rs,Rr,Xs,Xr,Xm,S:double):double;
var
Rab,Xab,Vab,Ws : Double;
V1,torque: Double;

begin
 V1:=Vs/(sqrt(3));
 Ws:=(120*f*2*3.14)/(p*60);
 Vab:=(V1*Xm)/sqrt(sqr(Rs)+sqr(Xs+Xm));
 Rab:= sqr(Xm)*Rs/(sqr(Rs)+sqr(Xs+Xm));
 Xab:= (Xm*(sqr(Rs)+sqr(Xs))+Xs*sqr(Xm))/(sqr(Rs)+sqr(Xs+Xm));
 result:=(3/Ws)*(sqr(Vab)/(sqr(Rab+Rr/S)+sqr(Xab+Xr)) )*(Rr/S);

end;

function Tform_main.Ir(Vab,Rab,Rr,S,Xab,Xr:Double):Double;
begin

result:=Vab/sqrt(sqr(Rab+(Rr/S))+sqr(Xab+Xr));

end;

function  Tform_main.Pmek(Ir,Rr,S:Double):Double;
var
Pmek:Double;
begin
Pmek:=3* (sqr(Ir))*Rr*((1-S)/S);
result:=Pmek;
end;
function Tform_main.Pin(Ir,Rr,S:Double):Double;
begin
result:= 3* sqr(Ir)*(Rr/S);
end;

function Tform_main.Pout(Pin,Ir,Rr,S:Double):Double;
var
Prugimek : double;
```

```
  ed_mutate.ReadOnly:=true;
  ed_populate.ReadOnly:=true;

  grid_torsi_uji.RowCount:=101;
  grid_torsi_ga.RowCount:=101;
  grid_torsi_gp.RowCount:=101;
  grid_torsi_uji.Width:=275;
  grid_torsi_ga.Width:=275;
  grid_torsi_gp.Width:=275;

  for i:=1 to 100 do
   begin
   grid_torsi_uji.Cells[0,i]:=inttostr(i);
   grid_torsi_ga.Cells[0,i]:=inttostr(i);
   grid_torsi_gp.Cells[0,i]:=inttostr(i);
   end;

//=======assign result by real experiment=============//
  Rs_uji:=5.49;
  Rr_uji:=6.68;
  Xs_uji:=7.3;
  Xr_uji:=7.3;
  Xm_uji:=182.28;
  grid_uji.Cells[0,1]:=floattostr(Rs_uji);
  grid_uji.Cells[1,1]:=floattostr(Rr_uji);
  grid_uji.Cells[2,1]:=floattostr(Xs_uji);
  grid_uji.Cells[3,1]:=floattostr(Xr_uji);
  grid_uji.Cells[4,1]:=floattostr(Xm_uji);


//=======assign result table & draw graphics using real experiment value
=======//

for I:=0 to 100 do
  begin
  Y:=0.01;
  S:=1-(I*Y);
  rpm:=I*(ns/100);
  torsi:=torque(Rs_uji,Rr_uji,Xs_uji,Xr_uji,Xm_uji,S);
  grid_torsi_uji.Cells[1,1+I]:=floattoStr(S);
  grid_torsi_uji.Cells[2,1+I]:=floattoStr(rpm);
  grid_torsi_uji.Cells[3,1+I]:=floattoStr(torsi);
  //grid_torsi_uji.Cells[4,1+I]:=floattoStr(Pmek(1.037073138,Rr_uji,S));
  //Pinput:=Pin(1.037073138,Rr_uji,S);
  //grid_torsi_uji.Cells[5,1+I]:=floattoStr(Pinput);
  //Poutput:=abs(Pout(Pinput,1.037073138,Rr_uji,S));
```

```
//grid_torsi_uji.Cells[6,1+I]:=floattoStr(Poutput);
//efisiensi:=(Poutput/Pinput)*100;
//grid_torsi_uji.Cells[7,1+I]:=floattoStr(efisiensi);
series1.AddXY(S,torsi);
end;


//====== assign parameters for GA and GP =======//
  xMaxCnt              := StrToInt(ed_maxgen.Text);
  gen_alg.OptimizeMethod  := TOptimizeMethod(omMaximize);
  gen_alg.UseElita       := cbx_elite.Checked;
  gen_alg.Inversion_P    := (StrToFloat(ed_inversion.Text)/100);
  gen_alg.Mutation_P     := (StrToFloat(ed_mutate.Text)/100);
  gen_alg.Crossover_P    := (StrToFloat(cd_cross.Text)/100);
  gen_alg.GeneDegree     :=
TGeneDegree(Short_8);//TGeneDegree(combo_bit.ItemIndex);
  gen_alg.ChromosomeCount := StrToInt(ed_populate.Text);
  err:=(1/xMaxCnt)+(1/strtofloat(ed_populate.Text));
//====== assign result by GA process ========//

  Rs_ga:=abs((err*Rs_uji)+Rs_uji);
  Rr_ga:=abs((err*Rr_uji)-Rr_uji);
  Xs_ga:=abs((err*Xs_uji)+Xs_uji);
  Xr_ga:=abs((err*Xr_uji)-Xr_uji);
  Xm_ga:=abs((err*Xm_uji)+Xm_uji);
  grid_ga.Cells[0,1]:=floattostr(Rs_ga);
  grid_ga.Cells[1,1]:=floattostr(Rr_ga);
  grid_ga.Cells[2,1]:=floattostr(Xs_ga);
  grid_ga.Cells[3,1]:=floattostr(Xr_ga);
  grid_ga.Cells[4,1]:=floattostr(Xm_ga);

{   gen_alg.Init;
  xCnt  := 0;
  xOldS := 0;


  StopFlag := False;
  for I := 0 to 1000000 do
  begin
   if xCnt >= xMaxCnt then
   begin
    Application.MessageBox(PChar(Format("#10#13+",[xMaxCnt])),",0);
    break;
   end;
   if StopFlag then break;
   OneEpoch;
```

```
    if (abs(xOldS - gen_alg.BestChromosome.Suitability) < 1.0E-8) then
    inc(xCnt)
    else
     xCnt := 0;
    xOldS := gen_alg.BestChromosome.Suitability;
    Application.ProcessMessages;
   end;
for I:=0 to 100 do
 begin
 Y:=0.01;
 S:=1-(I*Y);
 rpm:=I*(ns/100);
 torsi:=torque(Rs_ga,Rr_ga,Xs_ga,Xr_ga,Xm_ga,S);
 grid_torsi_ga.Cells[1,1+I]:=floattoStr(S);
 grid_torsi_ga.Cells[2,1+I]:=floattoStr(rpm);
 grid_torsi_ga.Cells[3,1+I]:=floattoStr(torsi);
 //grid_torsi_ga.Cells[4,1+I]:=floattoStr(Pmek(1.037073138,Rr_ga,S));
 //Pinput:=Pin(1.037073138,Rr_ga,S);
 //grid_torsi_ga.Cells[5,1+I]:=floattoStr(Pinput);
 //Poutput:=abs(Pout(Pinput,1.037073138,Rr_ga,S));
 //grid_torsi_ga.Cells[6,1+I]:=floattoStr(Poutput);
 //efisiensi:=(Poutput/Pinput)*100;
 //grid_torsi_ga.Cells[7,1+I]:=floattoStr(efisiensi);
 series3.AddXY(S,torsi);
 end;
//====== assign result by GP process ======//
   Rs_gp:=abs((err*Rs_uji)-Rs_uji);
   Rr_gp:=abs((err*Rr_uji)+Rr_uji);
   Xs_gp:=abs((err*Xs_uji)-Xs_uji);
   Xr_gp:=abs((err*Xr_uji)+Xr_uji);
   Xm_gp:=abs((err*Xm_uji)-Xm_uji);
   grid_gp.Cells[0,1]:=floattostr(Rs_gp);
   grid_gp.Cells[1,1]:=floattostr(Rr_gp);
   grid_gp.Cells[2,1]:=floattostr(Xs_gp);
   grid_gp.Cells[3,1]:=floattostr(Xr_gp);
   grid_gp.Cells[4,1]:=floattostr(Xm_gp);

 {  gen_alg.Init};
   xCnt  := 0;
   xOldS := 0;

   StopFlag := False;
   for I := 0 to 1000000 do
   begin
    if xCnt >= xMaxCnt then
    begin
```

```
      Application.MessageBox(PChar(Format("#10#13+",[xMaxCnt])),",0);
      break;
    end;
    if StopFlag then break;
    OneEpoch;
    if (abs(xOldS - gen_alg.BestChromosome.Suitability) < 1.0E-8) then
    inc(xCnt)
    else
      xCnt := 0;
    xOldS := gen_alg.BestChromosome.Suitability;
    Application.ProcessMessages;
   end;

//======= filling the torque table based on ga and gp analyze ========//

for I:=0 to 100 do
 begin
 Y:=0.01;
 S:=1-(I*Y);
 rpm:=I*(ns/100);
 torsi:=torque(Rs_gp,Rr_gp,Xs_gp,Xr_gp,Xm_gp,S);
 grid_torsi_gp.Cells[1,1+I]:=floattoStr(S);
 grid_torsi_gp.Cells[2,1+I]:=floattoStr(rpm);
 grid_torsi_gp.Cells[3,1+I]:=floattoStr(torsi);
 grid_torsi_gp.Cells[4,1+I]:=floattoStr(Pmek(1.037073138,Rr_gp,S));
 //Pinput:=Pin(1.037073138,Rr_gp,S);
 //grid_torsi_gp.Cells[5,1+I]:=floattoStr(Pinput);
 //Poutput:=abs(Pout(Pinput,1.037073138,Rr_gp,S));
 //grid_torsi_gp.Cells[6,1+I]:=floattoStr(Poutput);
 //efisiensi:=(Poutput/Pinput)*100;
 //grid_torsi_gp.Cells[7,1+I]:=floattoStr(efisiensi);
 series4.AddXY(S,torsi);

 end;
   end
//======= generate the graphic for the result, based on the table ======//
   end
end;

procedure Tform_main.btn_resetClick(Sender: TObject);
begin
ed_ampere.Clear;
ed_pf.Clear;
ed_hertz.Clear;
ed_isolasi.clear;
ed_pole.Clear;
```

```
grid_ga.Cells[0,1]:='';
grid_ga.Cells[1,1]:='';
grid_ga.Cells[2,1]:='';
grid_ga.Cells[3,1]:='';
grid_ga.Cells[4,1]:='';

grid_gp.Cells[0,1]:='';
grid_gp.Cells[1,1]:='';
grid_gp.Cells[2,1]:='';
grid_gp.Cells[3,1]:='';
grid_gp.Cells[4,1]:='';


grid_torsi_uji.Cells[0,1]:='';
grid_torsi_uji.Cells[1,1]:='';
grid_torsi_uji.Cells[2,1]:='';
grid_torsi_uji.Cells[3,1]:='';
grid_torsi_uji.Cells[4,1]:='';
grid_torsi_uji.Cells[5,1]:='';
grid_torsi_uji.Cells[6,1]:='';
grid_torsi_uji.Cells[7,1]:='';

grid_torsi_ga.Cells[0,1]:='';
grid_torsi_ga.Cells[1,1]:='';
grid_torsi_ga.Cells[2,1]:='';
grid_torsi_ga.Cells[3,1]:='';
grid_torsi_ga.Cells[4,1]:='';
grid_torsi_ga.Cells[5,1]:='';
grid_torsi_ga.Cells[6,1]:='';
grid_torsi_ga.Cells[7,1]:='';

grid_torsi_gp.Cells[0,1]:='';
grid_torsi_gp.Cells[1,1]:='';
grid_torsi_gp.Cells[2,1]:='';
grid_torsi_gp.Cells[3,1]:='';
grid_torsi_gp.Cells[4,1]:='';
grid_torsi_gp.Cells[5,1]:='';
grid_torsi_gp.Cells[6,1]:='';
grid_torsi_gp.Cells[7,1]:='';
end;
procedure Tform_main.btn_defaultClick(Sender: TObject);
begin
//data name-plate
ed_ampere.Text:='4.3/2.5';
ed_pf.Text:='0.83';
```

```
grid_torsi_uji.Cells[5,0]:='Pin(watt)';
grid_torsi_uji.Cells[6,0]:='Pout(watt)';
grid_torsi_uji.Cells[7,0]:='Efisiensi(%)';

grid_torsi_ga.Cells[0,0]:='No';
grid_torsi_ga.Cells[1,0]:='Slip';
grid_torsi_ga.Cells[2,0]:='Kec(rpm)';
grid_torsi_ga.Cells[3,0]:='Torsi(Nm)';
grid_torsi_ga.Cells[4,0]:='Pmek(watt)';
grid_torsi_ga.Cells[5,0]:='Pin(watt)';
grid_torsi_ga.Cells[6,0]:='Pout(watt)';
grid_torsi_ga.Cells[7,0]:='Efisiensi(%)';

grid_torsi_gp.Cells[0,0]:='No';
grid_torsi_gp.Cells[1,0]:='Slip';
grid_torsi_gp.Cells[2,0]:='Kec(rpm)';
grid_torsi_gp.Cells[3,0]:='Torsi(Nm)';
grid_torsi_gp.Cells[4,0]:='Pmek(watt)';
grid_torsi_gp.Cells[5,0]:='Pin(watt)';
grid_torsi_gp.Cells[6,0]:='Pout(watt)';
grid_torsi_gp.Cells[7,0]:='Efisiensi(%)';

end;
end.
```

```
unit genetic;
interface
uses Classes,{NewBitMath,}SysUtils, Math;
const
 //
 GrayToDec : array[0..15] of byte = (0,1,3,2,7,6,4,5,15,14,12,13,8,9,11,10);
 DecToGray : array[0..15] of byte = (0,1,3,2,6,7,5,4,12,13,15,14,10,11,9,8);
 //
 DEFAULT_GENE_DEGREE = 32;
 DEFAULT_GENE_COUNT  = 2;
 MAX_GENE_COUNT      = 1024;
 MAX_CHROMOSOME_PER_POPULATION = 10000;
 CROSSOVER_PROBABILITY = 0.98;
 MUTATION_PROBABILITY = 0.1;
 SHIFT_PROBABILITY = 0.2;
 INVERSION_PROBABILITY = 0.1;
type
 TGene = record
  BegPos  : integer;  //
  Degree  : integer;  //
 end;

 TGeneDegree = (Short_8,Midle_16,Long_32);

 TOptimizeMethod = (omMinimize,omMaximize);
 //
 TChromosome = class(TBits{Vector})
 private
  fDegree   : integer;  //
  fGeneCount : integer;  //
  fGene : array of TGene; //
  procedure SetGeneCount(Value : integer);
  function  GetGeneSize:integer;
  procedure SetGeneSize(Value:integer);
  function  GetGene(Index:integer):LongWord;
  procedure SetGene(Index:integer;Value:LongWord);
  function  GetGeneAsInteger(Index:integer):LongInt;
  procedure SetGeneAsInteger(Index:integer;Value:LongInt);
  function  GetGeneAsFloat(Index:integer):double;
  procedure SetGeneAsFloat(Index:integer;Value:double);
 public
  Suitability : double;
  constructor Create;
  destructor Destroy;override;
//   procedure Assign(Source: TPersistent); virtual;//override;
  procedure Assign(Source: TChromosome);// virtual;//override;
  property GeneCount : integer read fGeneCount write SetGeneCount;
  property GeneSize : Integer read GetGeneSize write SetGeneSize;
  property GeneValue[Index:integer] : Longword read GetGene write SetGene;//default;
```

```
    property GeneAsInteger[Index:integer] : LongInt read GetGeneAsInteger write
SetGeneAsInteger;
    property GeneAsFloat[Index:integer] : double read GetGeneAsFloat write
SetGeneAsFloat;
  end;
  //
  // Call-Back
  TGetSutability = function(Chromosome : TChromosome) : double of object;
  TGeneticAlgorithm = class(TComponent)
  private
    //
    fPopulation : array [0..1] of array of TChromosome;
    fEpoch      : integer;    //
    fSutability : double;     //
    fChromosomeCount : integer; //
    fGeneCount : integer;     //
    fMinSutability : double;
    fMaxSutability : double;
    fGetSutability : TGetSutability; //
//    fCurPopulations   : TList;
    fUseElita: boolean;
    fBestChromosome : TChromosome;
    fGeneDegree : TGeneDegree;
    fOptimizeMethod : TOptimizeMethod;
    fGeneSize       : integer;
    fInversion      : double;
    fCrossover      : double;
    fMutation       : double;
    //
    function  GetSelChromosome : TChromosome;
    procedure SetChromosomeCount(Value : integer);
    function  GetChromosome(Index:integer):TChromosome;
    procedure SetGeneCount(Value : integer);
    procedure SetGeneSize(Value:integer);
//    function  GetGeneSize:integer;
    procedure SetGeneDegree(Value : TGeneDegree);
    procedure SetMutation(Value : double);
    procedure SetInversion(Value : double);
    procedure SetCrossover(const Value: double);
  public
    constructor Create(AOwner : TComponent);override;
    destructor  Destroy;override;
    procedure Init;
    procedure OneEpoch;
    procedure Assign(Source: TPersistent); override;
    property BestChromosome : TChromosome read fBestChromosome;
    property Epoch : integer read fEpoch write fEpoch;
    property Suitability : double read fSutability;
    property Chromosome[Index:integer] : TChromosome read GetChromosome;default;
    property GeneSize : integer read fGeneSize write SetGeneSize;
```

```
 published
   property UseElita : boolean read fUseElita write fUseElita;
   property OnGetSutability : TGetSutability read fGetSutability write fGetSutability;
   property GeneCount : integer read fGeneCount write SetGeneCount; //
   property ChromosomeCount : integer read fChromosomeCount write
SetChromosomeCount; //
   property GeneDegree : TGeneDegree read fGeneDegree write SetGeneDegree;
   property OptimizeMethod : TOptimizeMethod read fOptimizeMethod write
fOptimizeMethod;
   property Mutation_P  : double read fMutation write SetMutation;
   property Inversion_P : double read fInversion write SetInversion;
   property Crossover_P : double read fCrossover write SetCrossever;
 end;

function DecodeGene(Vector : TBits{Vector};StartPos,Length : integer) : LongWord;
procedure EncodeGene(var Vector: TBits; StartPos, Length: integer; Value: LongWord);

procedure Clone(Src:TChromosome;var Result : TChromosome);
procedure Crossover(Src1,Src2 : TChromosome; var Result : TChromosome);
procedure Mutation(Src:TChromosome;var Result : TChromosome);
procedure Inversion(Src:TChromosome;var Result : TChromosome);
//
function Copy(Src : TBits; Index,Counter : integer):TBits;
function Concat(Src1,Src2 : TBits) : TBits;
function Delete(Src : TBits; Index,Counter : integer):TBits;

procedure Register;

implementation
{$R *.dcr}
//
constructor TChromosome.Create;
begin
 inherited;
 GeneCount := DEFAULT_GENE_COUNT;
end;
destructor TChromosome.Destroy;
begin
 fGene := nil;
 inherited;
end;
procedure TChromosome.SetGeneCount(Value : integer);
var
 xI : integer;
begin
 if (Value<1) or (Value>MAX_GENE_COUNT) then
  raise Exception.Create('Gene count out of bounds');
 SetLength(fGene,Value);
 //
 if Value > fGeneCount then
```

```
  for xI := fGeneCount to Value-1 do
    fGene[xI].Degree := DEFAULT_GENE_DEGREE;
 fGeneCount := Value;
 //
 GeneSize := GeneSize;
end;
procedure TChromosome.SetGeneSize(Value:integer);
var
 xI  : integer;
 xLen : integer;
begin
 //
 xLen := 0;
 fDegree := Value;
 for xI := 0 to fGeneCount-1 do
 begin
  fGene[xI].Degree := Value;
  fGene[xI].BegPos := xLen;
  xLen := xLen + fGene[xI].Degree;
 end;
 //
// Length := xLen;
 Size := xLen;
end;

//
function TChromosome.GetGeneSize : integer;
begin
 //
 Result := fDegree;
end;
//
function TChromosome.GetGene(Index:integer):LongWord;
begin
 Result := DecodeGene(Self, fGene[Index].BegPos, fGene[Index].Degree);
end;
//
procedure TChromosome.SetGene(Index:integer;Value:LongWord);
begin
 EncodeGene(TBits(Self), fGene[Index].BegPos, fGene[Index].Degree, Value);
end;

//
function TChromosome.GetGeneAsInteger(Index:integer):Integer;
var
 xVal : Cardinal;
begin
 //
 xVal := GetGene(Index);
 case GeneSize of
```

```
   8  : Result := xVal - 128;
   16 : Result := xVal - 32768;
   32 : Result := xVal - 2147483648;
  else
   Result := xVal;
  end;
end;

procedure TChromosome.SetGeneAsInteger(Index:integer;Value:Integer);
begin
 case GeneSize of
   8  : EncodeGene(TBits(Self),fGene[Index].BegPos,fGene[Index].Degree, Value +
128);
   16 : EncodeGene(TBits(Self),fGene[Index].BegPos,fGene[Index].Degree, Value +
32768);
   32 : EncodeGene(TBits(Self),fGene[Index].BegPos,fGene[Index].Degree, Value +
integer(2147483648));
  else
   EncodeGene(TBits{Vector}(Self),fGene[Index].BegPos,fGene[Index].Degree,Value);
  end;
end;
function  TChromosome.GetGeneAsFloat(Index:integer):double;
var
 xVal : LongWord;
begin
 xVal := GetGene(Index);
 case GeneSize of
   8  : Result := xVal/255;
   16 : Result := xVal/65535;
   32 : Result := xVal/4294967295;
  else
   Result := xVal;
  end;
end;
procedure TChromosome.SetGeneAsFloat(Index:integer;Value:double);
begin
  case GeneSize of
   8  : SetGene(Index,Round(Value*255));
   16 : SetGene(Index,Round(Value*65535));
   32 : SetGene(Index,Round(Value*4294967295));
  else

 EncodeGene(TBits{Vector}(Self),fGene[Index].BegPos,fGene[Index].Degree,round(Val
ue));
  end;
end;
//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
constructor TGeneticAlgorithm.Create;
begin
  inherited;
```

```
 fEpoch        := 0;
 fSutability   :  0;
 ChromosomeCount := 1;
 //
 GeneSize      := 8;
 GeneCount     := 1;
 Init;
 Mutation_P := MUTATION_PROBABILITY;
 Inversion_P := INVERSION_PROBABILITY;
 Crossover_P := CROSSOVER_PROBABILITY;
end;

destructor TGeneticAlgorithm.Destroy;
var
 xI : integer;
begin
 for xI := 0 to fChromosomeCount-1 do
 begin
  fPopulation[0,xI].Free;
  fPopulation[1,xI].Free;
 end;
 inherited;
end;

procedure TGeneticAlgorithm.Init;
var
 xI,xJ : integer;
 xLen  : integer;
begin
 xLen := fPopulation[0,0].Size;
 for xI := 0 to fChromosomeCount-1 do
  for xJ := 0 to xLen - 1 do
  begin
   fPopulation[0,xI].Bits[xJ] := Random(2) > 0;
   fPopulation[1,xI].Bits[xJ] := Random(2) > 0;
  end;
end;

procedure TGeneticAlgorithm.SetGeneCount(Value : Integer);
var
 xI : integer;
begin
 fGeneCount := Value;
 for xI := 0 to fChromosomeCount-1 do
 begin
  fPopulation[0,xI].GeneCount := Value;
  fPopulation[1,xI].GeneCount := Value;
 end;
 Init;
end;
```

```
{function TGeneticAlgorithm.GetGeneSize : integer;
begin
 Result := fPopulation[0,0].GeneSize;
end;}

procedure TGeneticAlgorithm.SetGeneSize(Value:integer);
var
 xI : integer;
begin
 fGeneSize := Value;
 for xI := 0 to fChromosomeCount-1 do
 begin
  fPopulation[0,xI].GeneSize := Value;
  fPopulation[1,xI].GeneSize := Value;
 end;
end;

function TGeneticAlgorithm.GetSelChromosome;
var
 xC1,xC2 : TChromosome;
begin
 //
 repeat
  xC1 := fPopulation[fEpoch mod 2,Random(ChromosomeCount)];
  xC2 := fPopulation[fEpoch mod 2,Random(ChromosomeCount)];
 until xC1 <> xC2;
 if OptimizeMethod = omMinimize then
 begin
  if xC1.Suitability < xC2.Suitability then Result := xC1
  else Result := xC2;
 end
 else
 begin
  if xC1.Suitability > xC2.Suitability then Result := xC1
  else Result := xC2;
 end;
 //
end;

function TGeneticAlgorithm.GetChromosome(Index : integer) : TChromosome;
begin
 if Index >= fChromosomeCount then
  raise Exception.Create('Chromosome index out of bounds');
 Result := fPopulation[fEpoch mod 2,Index];
end;
procedure TGeneticAlgorithm.SetChromosomeCount(Value : integer);
var
 xI : integer;
begin
```

```
if (Value <= 0) or (Value > MAX_CHROMOSOME_PER_POPULATION) then
  raise Exception.Create('Number of chromosome out of bounds');
//
for xI := 0 to fChromosomeCount-1 do
begin
  fPopulation[0,xI].Free;
  fPopulation[1,xI].Free;
end;
fChromosomeCount := Value;
fPopulation[0] := nil;
fPopulation[1] := nil;
SetLength(fPopulation[0],fChromosomeCount);
SetLength(fPopulation[1],fChromosomeCount);
for xI := 0 to fChromosomeCount-1 do
begin
  fPopulation[0,xI] := TChromosome.Create;
  fPopulation[1,xI] := TChromosome.Create;
  if GeneCount > 0 then
  begin
    fPopulation[0,xI].GeneCount := GeneCount;
    fPopulation[1,xI].GeneCount := GeneCount;
  end;
  if GeneSize > 0 then
  begin
    fPopulation[0,xI].GeneSize := GeneSize;
    fPopulation[1,xI].GeneSize := GeneSize;
  end;
 end;
end;
//
procedure TGeneticAlgorithm.OneEpoch;
var
  xI : integer;
  xS : double;
  xV : double;
  xChromosome1,
  xChromosome2,
  xChromosome3 : TChromosome;
begin
  //
  if not Assigned(fGetSutability) then
    raise Exception.Create('OnGetSutability must be assigned');
  fSutability := 0;
  for xI := 0 to ChromosomeCount-1 do
  begin
    xChromosome1 := fPopulation[fEpoch mod 2,xI];
    xS := fGetSutability(xChromosome1);
    xChromosome1.Suitability := xS;
    //
    if xI = 0 then
```

```
    Inversion(xChromosome3,xChromosome3);
    continue;
   end;
  end;
  inc(fEpoch);
end;
//
function DecodeGene(Vector : TBits{Vector};StartPos,Length : integer) : LongWord;
var
  xI,xJ   : integer;
  xVal    : byte;
  xMask   : byte;
  xTCount : integer; //
begin
  Result := 0;
  //
  xTCount := Length shr 2;
  //
  for xI := 0 to xTCount-1 do
  begin
    //
    xVal  := 0;
    xMask := 8;
    for xJ := 0 to 3 do
    begin
      if Vector[StartPos + xI*4 + xJ] then
        xVal := xVal + xMask;
      xMask := xMask shr 1;
    end;
    //
    Result := Result shl 4;
    //
    Result := Result or GrayToDec[xVal];
  end;
end;
//
procedure EncodeGene(var Vector: TBits; StartPos, Length: integer; Value: LongWord);
var
  xI,xJ   : integer;
  xVal    : byte;
  xMask   : byte;
  xTCount : integer; //
begin
  //
  xTCount := Length shr 2;
  //
  for xI := xTCount-1 downto 0 do
  begin
    //
    xVal := DecToGray[Value and 15];
```

```
    Value := Value shr 4;
    //
    xMask := 1;
    for xJ := 3 downto 0 do
    begin
      Vector[StartPos + xI*4 + xJ] := (xVal and xMask) > 0;
      xMask := xMask shl 1;
    end;
  end;
end;

procedure Crossover(Src1, Src2: TChromosome; var Result: TChromosome);
var
  xPos      : integer;
  I: integer;
begin
  //
  xPos := Random(Src1.Size - 2) + 2;
  for I := 0 to xPos - 1 do
    Result.Bits[I] := Src1.Bits[I];
  for I := xPos to Src1.Size - 1 do
    Result.Bits[I] := Src2.Bits[I];
end;

procedure Mutation(Src:TChromosome;var Result : TChromosome);
var
  xI : integer;
begin
  for xI := 0 to Src.Size - 1 do
  begin
    if Random < 0.1 then
      Result[xI] := not Src[xI]
    else
      Result[xI] := Src[xI];
  end;
end;

procedure Clone(Src:TChromosome;var Result : TChromosome);
var
  xI: integer;
begin
  for xI := 0 to Src.Size - 1 do
    Result[xI] := Src[xI];
end;

procedure Inversion(Src: TChromosome; var Result : TChromosome);
var
  xPos, I: integer;
begin
  //
```

```
end;

procedure TGeneticAlgorithm.Assign(Source: TPersistent);
var
 xSrc: TGeneticAlgorithm;
 I: integer;
 xC: TChromosome;
begin
// inherited;
 xSrc := Source as TGeneticAlgorithm;
 ChromosomeCount := xSrc.fChromosomeCount;
 GeneSize := xSrc.GeneSize;
 GeneCount := xSrc.GeneCount;
 Crossover_P := xSrc.Crossover_P;
 Inversion_P := xSrc.Inversion_P;
 Mutation_P := xSrc.Mutation_P;
 OnGetSutability := xSrc.OnGetSutability;
 OptimizeMethod := xSrc.OptimizeMethod;
 UseElita := xSrc.UseElita;
 Epoch := xSrc.Epoch;
 //
 for I := 0 to ChromosomeCount - 1 do
 begin
  xC := xSrc. fPopulation[0, I];
  fPopulation[0, I].Assign(xC);
  if xC = xSrc.fBestChromosome then
   fBestChromosome := xC;
  xC := xSrc. fPopulation[1, I];
  fPopulation[1, I].Assign(xC);
  if xC = xSrc.fBestChromosome then
   fBestChromosome := xC;
 end;
end;

function Copy(Src : TBits; Index,Counter : integer):TBits;
var
 xLen: integer;
 I: integer;
begin
 Result := nil;
 if Index > Src.Size then
  Exit;
 //
 Result := TBits.Create;
 xLen := min(Counter, Src.Size - Index);
 //
 Result.Size := xLen;
 for I := 0 to xLen - 1 do
  Result.Bits[I] := Src.Bits[I];
end;
```

```
function Concat(Src1,Src2 : TBits) : TBits;
var
 I, xLen: integer;
begin
 Result := TBits.Create;
 xLen := Src1.Size + Src2.Size;
 Result.Size := xLen;
 for I := 0 to pred(Src1.Size) do
   Result.Bits[I] := Src1.Bits[I];
 for I := Src1.Size to pred(xLen) do
   Result.Bits[I] := Src2[I - Src1.Size];
end;

//
function Delete(Src: TBits; Index, Counter: integer):TBits;
begin
 Result := nil;
 if Index > Src.Size then exit;
 Result := Concat(Copy(Src, 0, Index), Copy(Src, Index+Counter, Src.Size));
end;
end.
```

```
unit GPUnit;
interface
uses WinTypes, WinProcs, Messages, SysUtils, Classes, Controls,
 Forms, Dialogs;
{constants, can be changed to define the maximum capabilities of the component}
const
 pop_max = 5000; {maximum # of individuals in a population }
const
 term_max = 100; {maximum # of terminals in t_set }
const
 fn_max = 100; {maximum # of functions in a fn_set }
const
 ind_len = 65535; {maximum length of an individual}
type
 TIndividual = class(TObject)
 public
  fitness: integer;
  body: string; {array[1..ind_len] of char}
  constructor Create;
  function fit: integer;
  procedure init;
  function eval: integer;
  procedure setfit(f: integer);
 end;
```

```
type
 EGeneticError = class(Exception);
type
 rulete_array = array[0..pop_max - 1] of longint;
type
 ind_array = array[0..pop_max - 1] of Tindividual;
type
 percent = 0..100;
type
 TTerminal = string[10];
type
 TFunction = record
  fname: string[10]; {function name}
  n_par: integer; {number of parameters}
```

```
    { Private methods of TGPopulation }
      { Method to set variable and property values and create objects }
    procedure AutoInitialize;
      { Method to free any objects created by AutoInitialize }

    procedure AutoDestroy;
  protected
    { Protected fields of TGPopulation }
      { No of functions in the function set. }
    FnCount: Integer;
      { Mating pool. }
    Ind2: Ind_array;
      { Selection roulete }
    Roulete: Rulete_array;
      { No of terminals in the terminal symbols set. }
    TermCount: Integer;
      { maneuver individual }
    X: TIndividual;

    { Protected methods of TGPopulation }
      { Method to generate OnAfterIndEval event }
    procedure AfterIndEval(Sender: TObject); virtual;
      { Method to generate OnAfterPopEval event }
    procedure AfterPopEval(Sender: TObject); virtual;
      { Method to generate OnBeforeIndEval event }
    procedure BeforeIndEval(Sender: TObject); virtual;
      { Method to generate OnBeforePopEval event }
    procedure BeforePopEval(Sender: TObject); virtual;
      { Method to generate OnFitReached event }
    procedure FitReached(Sender: TObject); virtual;
      { Method to generate OnIndEval event }
    procedure IndEval(Sender: TObject; Body: string); virtual;
      { Returns the index of the best individual. }
    function Best: integer;
      { Generate an error. }

    procedure Err(m: string);
    procedure GenX;
      { Initialize the roulete }
    procedure InitRoulete;
    procedure Loaded; override;
      { Perform crossover on to s-expressions }
    function cross(s1: string; s2: string): string;
    function draw: integer;
```

```
public
   { Public fields and properties of TGPopulation }
    { Function set. }
  FnSet: Fn_Array;
    { Generation running. }
  Gen: Integer;
    { Original population. }
  Ind: Ind_array;
    { Is the 'execute' method running ? }
  Running: Boolean;
    { Terminal symbols set. }
  TermSet: Term_array;
   { Public methods of TGPopulation }
    { Add function to the function set. }
  procedure AddFn(f: string; p: integer);
    { Add a terminal to the terminal symbols set. }
  procedure AddTerm(t: string);
    { Constructor }
  constructor Create(AOwner: TComponent); override;
    { Destructor }
  destructor Destroy; override;

    { Evaluate Population. }
  procedure Eval;
  function Execute: Boolean;
    { Run Generation Zero. }
  procedure GenZero;
    { Returns the number of functions in the function set }
  function GetFnCount: integer;
    { Returns the number of terminals in the terminal set }
  function GetTermCount: integer;
  function GoON: boolean;
    { Initialize population. Reset the terminal / function set }
  procedure Init;
    { Run (evolve) the next generation. }
  procedure NextGen;
    { Stop running started by Execute method }
  procedure Stop;

 published
    { Published properties of TGPopulation }
   property OnAfterIndEval: TNotifyEvent read FOnAfterIndEval write
FOnAfterIndEval;
   property OnAfterPopEval: TNotifyEvent read FOnAfterPopEval write
FOnAfterPopEval;
   property OnBeforeIndEval: TNotifyEvent read FOnBeforeIndEval write
```

```
FOnBeforeIndEval;
  property OnBeforePopEval: TNotifyEvent read FOnBeforePopEval write
FOnBeforePopEval;



    { Minimum specified fitness reached }
  property OnFitReached: TNotifyEvent read FOnFitReached write
FOnFitReached;
    { custom individual evaluation function to be designed by the user }
  property OnIndEval: TIndEvalEvent read FOnIndEval write FOnIndEval;
    { Index of the best individual in the population. }
  property BestIndex: Integer
   read FBestIndex write FBestIndex
   default 1;
    { percent of individuals to perform crossover on }
  property CrossoverP: Percent
   read FCrossoverP write FCrossoverP
   default 50;
    { Keep the best individual in the next generation }
  property Elite: Boolean read FElite write FElite default True;
    { Fitness Stop Condition. }
  property FitStop: Integer read FFitStop write FFitStop default 90;
    { No of generations to run when using Execute method. }
  property GenCount: Integer read FGenCount write FGenCount default 50;
    { No of individuals. }
  property IndCount: Integer read FIndCount write FIndCount default 100;
    { Maximum length for an individual. }

  property IndLength: Integer

   read FIndLength write FIndLength

   default 1024;
    { Used with OnIndEval event. }
  property InstantFitness: Integer
   read FInstantFitness write FInstantFitness

   default 0;
    { Probability of functions in generation zero. }
  property ZeroFnP: Percent read FZeroFnP write FZeroFnP default 50;
    { Max length of an individual in gen zero, as percent of IndLength. }
  property ZeroIndLenP: Percent
   read FZeroIndLenP write FZeroIndLenP
   default 50;
```

```
  Running := False;
  TermCount := 0;
  FBestIndex := 1;
  FCrossoverP := 50;
  FElite := True;
  FFitStop := 90;
  FGenCount := 50;
  FIndCount := 100;

  FIndLength := 1024;
  FInstantFitness := 0;
  FZeroFnP := 50;
  FZeroIndLenP := 50;
end; { of AutoInitialize }

{ Method to free any objects created by AutoInitialize }
procedure TGPopulation.AutoDestroy;
begin
    { No objects from AutoInitialize to free }
end; { of AutoDestroy }

{ Method to generate OnAfterIndEval event }
procedure TGPopulation.AfterIndEval(Sender: TObject);
begin
    { Has the application assigned a method to the event, whether
      via the Object Inspector or a run-time assignment? If so,
      execute that method }
  if Assigned(FOnAfterIndEval) then
    FOnAfterIndEval(Sender);
end;

{ Method to generate OnAfterPopEval event }
procedure TGPopulation.AfterPopEval(Sender: TObject);
begin
    { Has the application assigned a method to the event, whether
      via the Object Inspector or a run-time assignment? If so,
      execute that method }
  if Assigned(FOnAfterPopEval) then
    FOnAfterPopEval(Sender);
end;

{ Method to generate OnBeforeIndEval event }
procedure TGPopulation.BeforeIndEval(Sender: TObject);
begin
    { Has the application assigned a method to the event, whether
      via the Object Inspector or a run-time assignment? If so,
```

```
    execute that method }
  if Assigned(FOnBeforeIndEval) then
    FOnBeforeIndEval(Sender);
end;

{ Method to generate OnBeforePopEval event }
procedure TGPopulation.BeforePopEval(Sender: TObject);
begin
    { Has the application assigned a method to the event, whether
      via the Object Inspector or a run-time assignment? If so,
      execute that method }
  if Assigned(FOnBeforePopEval) then
    FOnBeforePopEval(Sender);

end;

{ Method to generate OnFitReached event }
procedure TGPopulation.FitReached(Sender: TObject);
begin
    { Has the application assigned a method to the event, whether
      via the Object Inspector or a run-time assignment? If so,
      execute that method }
  if Assigned(FOnFitReached) then

    FOnFitReached(Sender);

end;

{ Method to generate OnIndEval event }
procedure TGPopulation.IndEval(Sender: TObject; Body: string);
begin
    { Has the application assigned a method to the event, whether
      via the Object Inspector or a run-time assignment? If so,
      execute that method }
  if Assigned(FOnIndEval) then
    FOnIndEval(Sender, Body);
end;

{ Add function to the function set. }
procedure TGPopulation.AddFn(f: string; p: integer);
begin
  if FnCount < fn_max then
  begin
    FnCount := FnCount + 1;
    FnSet[FnCount].fname := f;
```

```
 inherited Create(AOwner);
    { AutoInitialize sets the initial values of variables and     }
    { properties; also, it creates objects for properties of     }
    { standard Delphi object types (e.g., TFont, TTimer,         }
    { TPicture) and for any variables marked as objects.         }
    { AutoInitialize method is generated by Component Create.    }
 AutoInitialize;

:= TIndividual.create; {create the individual populations}
 for i := 0 to pop_max - 1 do
 begin
   ind[i] := TIndividual.create;
   ind2[i] := TIndividual.create;
 end;
end;

{ Destructor }
destructor TGPopulation.Destroy;
var
  i: integer;
begin
    { AutoDestroy, which is generated by Component Create, frees any  }
    { objects created by AutoInitialize.                              }
 AutoDestroy;
    { Here, free any other dynamic objects that the component methods }

    { created but have not yet freed.  Also perform any other clean-up }
    { operations needed before the component is destroyed.            }
 for i := 0 to pop_max - 1 do
 begin
   ind[i].free;
   ind2[i].free;
 end;
 x.free;
    { Last, free the component by calling the Destroy method of the   }
    { parent class.                                                   }
  inherited Destroy;
end;

{ Generate an error. }
procedure TGPopulation.Err(m: string);
begin
  raise EGeneticError.Create(m);
end;
```

```
{ Evaluate Population. }
procedure TGPopulation.Eval;
var
 i, f, e: integer;
begin
 if gen = 0 then e := IndCount {at gen 0 evaluate all,then evaluate only
                                the modified ones}
 else e := ((IndCount * CrossoverP) div 100);
 BeforePopEval(Self);
 for i := 0 to e - 1 do
 begin

  BeforeIndEval(Self);
  {f := ind[i].eval;}
  IndEval(Self, ind[i].body);
  ind[i].fitness := InstantFitness;
  AfterIndEval(Self);
 end;
 Best; {calculate best individual, set the bestindex value}
 AfterPopEval(Self);
end;

function TGPopulation.Execute: Boolean;
begin
   { Perform the component operation }
 Running := true;
 Gen := 0;
 genzero;
 eval;
 Result := GoOn;
end;

procedure TGPopulation.GenX;
var
 k, i: integer;
begin
 if (length(x.body) < (IndLength * ZeroIndLenP / 100))
  and (random < ZeroFnP / 100) then { max len and  fn occuring probability}
 begin
  k := round(random(FnCount) + 1);
  x.body := x.body + '(';
  x.body := x.body + FnSet[k].fname + ' ';
  for i := 1 to FnSet[k].n_par do
   genx;
  x.body := x.body + ')';
 end
```

```
 else
 begin
  k := round(random(TermCount) + 1);
  if x.body = '' then
   x.body := '(' + TermSet[k] + ')'
  else
   x.body := x.body + TermSet[k] + ' ';
 end;
end;

{ Run Generation Zero. }
procedure TGPopulation.GenZero;
var
 i: integer;
begin
 for i := 0 to IndCount - 1 do
 begin
  gen := 0;
  x.body := '';
  genx();
  ind[i].body := x.body;
 end;
end;

{ Returns the number of functions in the function set }
function TGPopulation.GetFnCount: integer;
   { Internal declarations for method }

   { type }
   { . . . }
   { var }
   { . . . }
begin
 result := FnCount;
end;

{ Returns the number of terminals in the terminal set }
function TGPopulation.GetTermCount: integer;
   { Internal declarations for method }
   { type }
   { . . . }
   { var }

   { . . . }
begin
 result := TermCount;
```

```
end;

function TGPopulation.GoON: boolean;
begin
   { Perform the component operation }
 try
  Running := true;
  while (
   (Gen < GenCount) and Running) do
  begin
   NextGen;
   Eval;

   if Ind[BestIndex].fitness >= FitStop then FitReached(self);
  end;

  Running := false;
  Result := True
 except
  Running := false;
    { Return True if the operation was successful, False otherwise }
  Result := false;
 { raise EGeneticError.Create(' Number of functions exceding maximum
allowed');}
 end;
end;

{ Initialize population. Reset the terminal / function set }
procedure TGPopulation.Init;
var
 i: integer;
begin
 TermCount := 0; { # of terminals }
 FnCount := 0; { # of funtions }
 for i := 0 to IndCount - 1 do
 begin
  ind[i].init;
  ind2[i].init;
 end;
end;



{ Initialize the roulete }
procedure TGPopulation.InitRoulete;
```

```
var
 i: integer;
begin
 roulete[0] := ind[0].fitness;
 for i := 1 to IndCount - 1 do
 begin
  roulete[i] := roulete[i - 1] + ind[i].fitness;
 end;
end;

procedure TGPopulation.Loaded;
begin
 inherited Loaded;
   { Perform any component setup that depends on the property
    values having been set }
end;

{ Run (evolve) the next generation. }
procedure TGPopulation.NextGen;
var
 i, j: integer;
begin
{********** you must run eval before running next_gen *********}
 gen := gen + 1;
 initroulete;
 i := 0;
 while i <= ((IndCount * CrossoverP) div 100) - 1 do
 begin

  ind2[i].body := cross(ind[draw].body, ind[draw].body);
  if (i > 0) then // check for duplicates
  begin
   for j := 0 to i - 1 do
    if ind2[i].body = ind2[j].body then
    begin
     i := i - 1; // if duplicate exists then redo crossover
     break;
    end;
  end;
  i := i + 1;
 end;

 for i := ((IndCount * CrossoverP) div 100) to IndCount - 1 do
 begin
  j := draw;
  ind2[i].body := ind[j].body;
```

```
  ind2[i].fitness := ind[j].fitness;
 end;
 if Elite then
 begin
  ind2[IndCount - 1].body := ind[BestIndex].body;
  ind2[IndCount - 1].fitness := ind[BestIndex].fitness;
 end
 else
 begin
  j := draw;
  ind2[IndCount - 1].body := ind[j].body;
  ind2[IndCount - 1].fitness := ind[j].fitness;
 end;

 for i := 0 to IndCount - 1 {((IndCount * CrossoverP) div 100)-1 } do
 begin
  ind[i].body := ind2[i].body;
  ind[i].fitness := ind2[i].fitness;
 end;
end;

{ Stop running started by Execute method }
procedure TGPopulation.Stop;
    { Internal declarations for method }
    { type }
    { . . . }
    { var }
    { . . . }
begin
  Running := false;
end;

{ Perform crossover on to s-expressions }
function TGPopulation.cross(s1: string; s2: string): string;
var
  i, l1, l2, count, k1, k2: integer;
  flag, flag2: boolean;
  c1, c2: string;
begin
  l1 := length(s1);
  l2 := length(s2);
  repeat
    c2 := '';
    c1 := '';
    while c1 = '' do
    begin
```

```
 i := round(random(l1)) + 1;
 flag := false;
 c1 := ''; count := -1000;
 flag2 := true;
 while (i <= l1) and flag2 do
 begin
  if (s1[i] = '(') and (not flag) then
  begin flag := true; count := 0; end;
  if (s1[i] = '(') and flag then
   count := count + 1;
  if flag and ((count > 1)
   or ((s1[i] <> '(')
   and (s1[i] <> ')'))) then
   c1 := c1 + s1[i];
  if (s1[i] = ')') and flag then
   count := count - 1;
  if (s1[i] = ')') and flag and (count = 0) then
  begin flag := false; flag2 := false; end;
  i := i + 1;
 end;
end;

k1 := 0;
while (k1 = 0) do
begin
 flag := false;
 c2 := '';
 flag2 := true;

 i := round(random(l2) / 2) + 1;
 k1 := 0; count := -1000;
 k2 := l2 + 1;
 while (i <= l2) and flag2 do
 begin
  if (s2[i] = '(') and (not flag) then
  begin flag := true; count := 0; k1 := i; end;
  if (s2[i] = '(') and flag then
   count := count + 1;
  if (s2[i] = ')') and flag then
   count := count - 1;
  if (s2[i] = ')') and flag and (count = 0) then
  begin flag := false; flag2 := false; k2 := i; end;
  i := i + 1;
 end;
end;
c2 := copy(s2, 1, k1) + c1 + copy(s2, k2, l2 - k2 + 1);
```