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

OPTIMAL POWER FLOW MENGGUNAKAN METODE PARALLEL EVOLUTIONARY PROGRAMMING PADA SUB SISTEM 150 kV PAITON - BALI

SKRIPSI

Disusun oleh:
NUR ASYIK HIDAYATULLAH
00.12.041

MARET 2006

LEMBAR PERSETUJUAN

OPTIMAL POWER FLOW MENGGUNAKAN METODE *PARALLEL EVOLUTIONARY PROGRAMMING* PADA SUB SISTEM 150 kV PAITON - BALI

SKRIPSI

Disusun Guna Melengkapi dan Memenuhi Syarat-Syarat
Untuk Mencapai Gelar Sarjana Teknik

Disusun Oleh :

NUR ASYIK HIDAYATULLAH

NIM. 00.12.041

Mengetahui, Ketua Jurusan Teknik Elektro	Menyetujui, Dosen Pembimbing
Ir. F. Yudio Limpraptono, MT	Ir. H. Choiri
NIP.Y.103 950 0274	NIP.130703042

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

ABSTRAKSI

OPTIMAL POWER FLOW MENGGUNAKAN METODE *PARALLEL EVOLUTIONARY PROGRAMMING* PADA SUB SISTEM 150 kV PAITON DAN BALI

(Nur Asyik Hidayatullah, Nim 00.12.041, Teknik Elektro/T.Energi Listrik)
(Dosen Pembimbing: Ir. H. Choiri.)

Kata Kunci: *Optimal Power Flow, Network Constrained Economic Dispatch, Parallel Evolutionary Programming, Optimasi Biaya Pembangkitan.*

Adanya persoalan dalam menghadapi kebutuhan daya listrik yang tidak tetap dari waktu ke waktu, schingga menimbulkan permasalahan yaitu bagaimana mengoperasikan suatu sistem tenaga listrik yang selalu dapat memenuhi permintaan daya pada setiap saat, dengan kualitas baik dan harga yang murah. Oleh karena itu pada suatu operasi pada beban tertentu, perhitungan ekonomis harus tetap merupakan suatu prioritas atau nilai yang harus diperhitungkan disamping hal-hal lain schingga nantinya diperlukan suatu rencana operasi yang optimum dengan tetap memenuhi beberapa persyaratan pengoperasian sistem tenaga listrik yaitu antara lain : daya yang dibangkitkan cukup untuk memasok beban dan rugi-rugi daya pada saluran transmisi, tegangan bus sesuai dengan ratingnya.

Skripsi ini menganalisis permasalahan *Optimal Power Flow* atau dengan nama lain yaitu *Network Constrained Economic Dispactch* dengan menggunakan metode *Parallel Evolutionary Programming*. Hasil dari analisa tersebut nantinya dapat digunakan sebagai salah satu acuan dalam operasi pembangkit dan penyaluran daya yang ekonomis dan optimal, terutama mengenai optimasi biaya pembangkitan. Input dari program ini adalah biaya bahan bakar*(fuel cost)* dan hasil perhitungan aliran daya. Sedangkan hasil akhir dari program ini yaitu hasil perhitungan aliran daya, tegangan dan sudut fasa tiap-tiap bus serta biaya pembangkit yang paling optimum.

Analisa dilakukan dengan bantuan program komputer dengan menggunakan bahasa pemrograman Delphi versi 7.0 dan telah sukses diuji cobakan pada sub sistem 150 kV Paiton dan Bali yang terdiri dari 25 bus, dimana telah berhasil dilakukan penghematan biaya pembangkitan sebesar Rp. 811.943, 654, - per jam atau terjadi optimasi biaya sebesar 15.90 %.

KATA PENGANTAR

Dengan memanjatkan puji syukur kehadirat Allah SWT, atas segala limpahan Rahmat, Hidayah dan Ma'unah-Nya kepada penulis, sehingga dapat menyelesaikan skripsi yang berjudul "*OPTIMAL POWER FLOW* MENGGUNAKAN METODE *PARALLEL EVOLUTIONARY PROGRAMMING* PADA SUB SISTEM 150 kV PAITON-BALI".

Penulisan Skripsi ini bertujuan untuk memenuhi kurikulum akademik yang harus ditempuh oleh setiap mahasiswa ITN Malang dalam menempuh sekaligus mengakhiri pendidikan pada jenjang strata satu (S-1) pada Jurusan Teknik Elektro Konsentrasi Teknik Energi Listrik.

Selama penulisan skripsi ini, penulis telah banyak mendapatkan bimbingan, pengarahan dan bantuan yang diberikan dari berbagai pihak, sehingga skripsi ini dapat terselesaikan, oleh karena itu, pada kesempatan kali ini penulis menyampaikan terima kasih yang sebesar-besarnya kepada;

1. Bapak Dr. Ir.Abraham Lomi, MSEE, selaku Rektor ITN Malang.
2. Bapak Ir. F Yudi Limpraptono, MT, selaku Kepala Jurusan Teknik Elektro ITN Malang.
3. Bapak Ir. H. Choiri, selaku Dosen Pembimbing.
4. Bapak Ugro W, ST selaku *Programmer.*
5. PT. PLN (Persero) P3B Region Jawa Timur dan Bali yang telah memberikan data yang di butuhkan dalam penulisan skripsi ini.

Penulis menyadari sepenuhnya atas segala kekurangan yang ada dalam penulisan skripsi ini, dengan segala kerendahan hati penulis mengharapkan saran dan kritik demi kesempurnaan skripsi ini. Akhirnya semoga skripsi ini bermanfaat bagi kita semua..

Malang, Maret 2006

Penulis.

DAFTAR ISI

DAFTAR GAMBAR

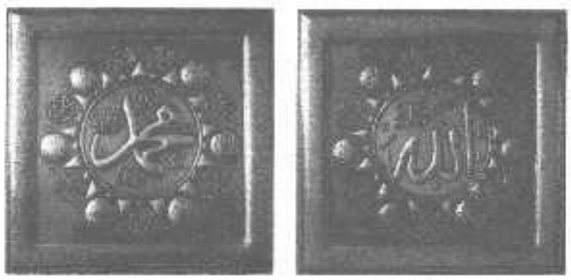
DAFTAR TABEL

DAFTAR GRAFIK

Lembar Persembahan

بِسْمِ اللهِ الرَّحْمٰنِ الرَّحِيْمِ

Segala Puji bagi-Mu Ya Allah atas segala Limpahan Rahmat, Nikmat Hidayah dan Ma'unah-Mu kepadaku.
Hanya Keridhoan-Mu yang hamba Harapkan dalam hidupku ini, agar hamba yang lemah ini tetap istiqomah dalam keimanan dan ketaqwaan.



Sholawat serta salam aku haturkan kepadamu ya Rosulullah, engkau sumber dari semua ilmu dan nur, bibit semua ma'rifat dan sir (rahasia).
Semoga Allah Ridho pada keluarga dan sahabatnya yang senantiasa taat mengikuti jejaknya.Amin.

Karya yang sederhana ini kami Persembahkan untuk;

Ayahanda Makin Al Amin dan Ibunda tercinta Siti Nasikah.

Terimakasih banyak atas Kasih sayang dan Doa restu yang telah diberikan selama ini. Cucuran keringat dan tempaan keras dari abah dan ibu membuat hati dan diri ini semakin kuat dan yakin untuk melangkah menggapai Kemulyaan Hidup di Dunia dan di Akhirat.

Sembah sujud dan Doa dari ananda selalu buat ayahanda dan ibunda sampai akhir zaman.

Terimakasih buat kakanda Umar Hamdani, ST dan Siti Mukhoiyyaroh, Sag. (Kesabaran serta Support dari kalian selama ini besar sekali, sehingga ikut mengantarkanku meyelesaikan semua ini, doakan juga biar aku jadi nglanjutin S-2 ya?).
Adikku tersayang Maria Baroroh, AMd dan Zumaroh Fuadatul Wasi'ah (Utamakan Dzikir dan Fikir dalam menghadapi masalah, tetep semangat ya!!)
Keponakanku yusron dan Aulia, mudah-mudahan jadi anak yang Birul Waalidain.

Seluruh Keluarga Besar Pesantren Mahasiswa Al Hikam Malang.
Al Mukarrom [illegible] Hasyim Muzadi dan Ibu selaku Pengasuh, terimakasih banyak atas Tausiyah dan ilmu yang disampaikan ke Asyik juga syaiun-syaiun nya selama ini.
Seluruh Dewan Asatidz, Ust Nafi, Ust Muzammil, Ust Hadi selaku wali kelas-ku, Ust Cholis (maksih tadz, dah ngasih kelonggaran Asyik tuk membayar syariah pondok) Ust. Kholiq, Ust.Zuhdi, Ust Munjinnasih dan Gus Andik.

Team KBIH Al hikam, Mas Sobri (yang sabar ya Mas?) toha, irfan, umam, tato, haris, hanif, ahya, hendra dll (team yang solid, cepetan lulus !!) serta semua teman-teman santri Al Hikam(jangan malas dirosah!!) tuk m.Abdu, makasih dah mau jadi partner ngomong bahasa Inggris di kamar. (belajarlah untuk dewasa dalam ucap dan sikap)

My Best Friend yang dah lulus duluan, Riza Maulana ST, waktu masih kuliah dulu kita hidup susah dan senang bareng2, terimakasih dukungan yang telah kau berikan, semoga persahabatan ini tak pernah pudar.
My Truly Friend, Dega Latu Baskoro, syukkron katsir deg! Dah dipercaya merawat komputer dan printer kamu selama 8 bulan, mudah-mudahan Allah memberikan kamu kemudahan sebagaimana kau memudahkanku.

Teman-teman seperjuangan sekripsi'06: hendra 99, hendra 00, toni, fahmi, Agung Cuwik Turbo Wind, Iswan dll.(kl menghadapi masalah itu harus sabar, dan optimis beres!!) serta teman2 anak ST angkatan'00 yang lain, terimakasih atas bantuan dan doanya juga persahabatannya.

Mas Ugro dan keluarga, terimakasih banyak atas kerjasamanya dalam pembuatan program ini, mohon maaf jika selama ini dah banyak menggangu dan menyita waktunya.

Keluarga besar Silat Tauhid Indonesia: Mas Ar.Gambang, Mas Agung, Pak Simbolon, Mas Sugeng (terimakasih banyak dah mendidik asyik dengan sabar) tuk Desta, agung mujaer, [illegible]a, aryo (seng rajin Dzikir Ghoib, jo telat dae!!)

Terimakasih untuk Ir. H. Choiri selaku Dosen Pembimbing sekripsi, yang telah setia memberikan arahan dan bimbingan dengan sabar sampai sekripsi ini dapat terselesaikan.

Ir. Yusuf Ismail Nakhoda, MT (dosen yang up to date soal teknologi) Terimakasih atas kepercayaan nya kepada asyik untuk bergabung di LAB.SSTE, juga traktiran-traktirannya. Tuk Aris Y, (sebagai korlab.tolong utamakan kejujuran dan kebersamaan).

Dosen ST yang telah bersedia membantu dan berdisdiskusi dengan saya: Ir.H.Almizan A.M.SEE, Ir.M. Abdul Hamid, MT, Ir. Teguh Herbasuki, MT, Ir.H. Taufik H, MT, dan I Made Wartana , MT. Terimaksih banyak ya pak.....?

"Sesungguhnya Allah tidak akan mengubah nasib suatu kaum, sehingga mereka mengubah nasibnya sendiri. Apabila Allah menghendaki keburukan terhadap suatu kaum, maka tidak ada yang dapat menolaknya. Dan sekali-kali tidak ada perlindungan bagi mereka selain Dia"

(Q. S. Ar-Ra'd [131] : 11)

"Sesungguhnya Allah tidak akan membebani seseorang kecuali sesuai dengan kesanggupannya. Ia mendapat pahala (dari kebaikan) yang dikerjakannya dan ia mendapat siksa (dari kejahatan) yang di kerjakannya

(Q. S. Al-Baqoroh [2] : 286)

"Boleh jadi kamu membenci sesuatu, padahal ia amat baik bagimu dan boleh jadi pula kamu menyukai sesuatu, padahal ia amat buruk bagimu. Allah Maha Mengetahui, sedangkan kamu tidak mengetahui.

(Q. S. Al-Baqoroh [2] : 216)

" Sesungguhnya Allah Selalu Menolong orang yang bertaqwa, dan orang yang benar-benar berbuat baik"

(Q. S. An-Nahl 128)

Katakanlah : "Kalau sekiranya lautan menjadi tinta untuk (menulis) kalimat-kalimat Tuhanku, sungguh habislah lautan itu sebelum habis (ditulis) kalimat-kalimat Tuhanku , meskipun Kami datangkan tambahan sebanyak itu pula"

(Q. S. Al- Kahfi : 109)

ASYIK MOTTO:

"NO DAYS WITHOUT ACT OF DEVOTIONS, LEARNING AND STRIKE A BLOW"

BAB I

PENDAHULUAN

1.1. Latar Belakang

Sistem tenaga listrik secara garis besar dapat dibagi menjadi 3 bagian yaitu: sisi pembangkit tenaga listrik, jaringan transmisi dan beban. Adanya persoalan dalam menghadapi kebutuhan daya listrik yang tidak tetap dari waktu ke waktu, sehingga menimbulkan permasalahan yaitu bagaimana mengoperasikan suatu sistem tenaga listrik yang selalu dapat memenuhi permintaan daya pada setiap saat, dengan kualitas baik dan harga yang murah. Oleh karena itu pada suatu operasi pada beban tertentu, perhitungan ekonomis harus tetap merupakan suatu prioritas atau nilai yang harus diperhitungkan disamping hal-hal lain sehingga nantinya diperlukan suatu rencana operasi yang optimum dengan tetap memenuhi beberapa persyaratan pengoperasian sistem tenaga listrik yaitu antara lain : daya yang dibangkitkan cukup untuk memasok beban dan rugi-rugi daya pada saluran transmisi, tegangan bus sesuai dengan ratingnya serta tidak adanya pembebanan lebih pada unit-unit pembangkit yang beroperasi.

Sehingga koordinasi antara unit-unit pembangkit yang ada pada sistem tenaga listrik sangat diperlukan untuk mencapai biaya operasi yang se-optimum mungkin, dalam hal ini yang dimaksud adalah optimum secara ekonomis, dengan tetap memperhatikan besar beban yang ada dan juga kestabilan tegangan sistem.

Sejak diperkenalkan sebagai *network constrained economic dispatch* oleh Capentier dan didefinisikan sebagai *Optimal Power Flow* (OPF) oleh Dommel dan Tinney [1]. OPF dipakai untuk mengoptimalkan fungsi objektif operasi sistem

tenaga listrik yaitu biaya operasi pembangkit thermal dan sekaligus memberikan seperangkat batasan operasi sistem, yang menyangkut batasan yang ditentukan oleh jaringan listrik.[2]

Dalam skripsi ini akan dibahas metode *Parallel Evolutionary Programming* (PEP), yakni suatu metode optimasi yang berusaha menyerupai operasi – operasi seleksi alamiah. PEP secara simultan dapat mengevaluasi banyak titik yang ada dalam ruang lingkup parameter secara sekaligus dan mengarah pada pencapaian *global optimum solution*.

1.2. Rumusan Masalah

1. Apakah dari hasil perhitungan metode *Parallel Evolutionary Programming* dapat diketahui berapa besarnya tegangan dan sudut fasa tiap bus, besarnya aliran daya dan rugi-rugi daya yang terjadi pada tiap saluran pada sub sistem 150 kV Paiton - Bali?
2. Apakah dengan metode *Parallel Evolutionary Programming* biaya pembangkitan bisa lebih Optimum?
3. Seberapa lama waktu komputasi dari metode *Parallel Evolutionary Programming* dalam pemecahan masalah *Optimal Power Flow* (OPF) pada sub sistem 150 kV Paiton - Bali?

Berdasarkan pada deskripsi latar belakang dan rumusan masalah tersebut diatas, maka skripsi ini disusun dengan judul :

***OPTIMAL POWER FLOW* MENGGUNAKAN METODE**
PARALLEL EVOLUTIONARY PROGRAMMING
PADA SUB-SISTEM 150 kV PAITON DAN BALI

1.3. Tujuan

Tujuan penulisan skripsi ini adalah untuk menganalisa pembagian beban listrik pada unit-unit pembangkit thermal serta untuk memperoleh biaya pembangkitan yang minimal dengan menggunakan metode *Parallel Evolutionary Programming* pada sub sistem 150 kV Paiton- Bali.

1.4. Batasan Masalah

Dalam skripsi ini akan dilakukan analisis terhadap OPF yang menggunakan metode PEP pada sub sistem 150 kV Paiton dan Bali. Agar pembahasan mengarah sesuai tujuan, maka pembahasan dalam skripsi ini dibatasi oleh hal sebagai berikut :

- Analisa dilakukan dengan asumsi bahwa sistem berada dalam operasi normal.
- Proses optimasi hanya dilakukan pada jam beban puncak yaitu pukul 19.30 WIB.
- Biaya *start up*, *shut down* dan *spinning reserve* di abaikan.
- Tidak membahas masalah pengaruh kontrol tegangan terhadap alat proteksi.
- Tidak membahas kabel laut, masalah peralatan kompensasi dan penempatannya.

1.5. Metodologi Penelitian

Metode yang digunakan dalam pembahasan skripsi ini dilakukan dengan langkah – langkah sebagai berikut :

1. Studi Literatur

   Yaitu kajian pustaka yang mempelajari teori – teori yang terkait melalui literatur yang telah ada, yang berhubungan dengan permasalahan.

2. Pengumpulan Data.

   Pengumpulan data lapangan yang dipakai dalam objek penelitian yakni data impedansi saluran transmisi dan data pembebanan sub sistem 150 kV Paiton dan Bali serta data pembangkit thermal.

   Bentuk data yang digunakan:

   a. Data Kuantitatif, yaitu data yang dapat dihitung atau data yang berbentuk angka-angka.

   b. Data Kualitatif, yaitu data yang berbentuk diagram.

3. Analisa Data yaitu; Analisa perhitungan tegangan, sudut phasa tegangan, aliran daya dan rugi daya pada tiap saluran serta biaya pembangkitan dengan menggunakan metode *parallel evolutionary programming* yang disimulasikan dengan komputer.

4. Simulasi dan pembahasan masalah

   Pembahasan masalah pada skripsi ini disimulasikan dengan komputer berbahasa Pemrograman Delphi versi 7.0. Sehingga dapat di ketahui perbandingan metode yang di terapkan lebih efisien atau ekonomis di bandingkan dengan yang di gunakan pada P3B.

1.6. Sistematika Penulisan

Adapun sistimatika pembahasan pada skripsi ini adalah sebagai berikut :

BAB I : PENDAHULUAN.

Menguraikan latar belakang, rumusan masalah, tujuan penelitian, batasan masalah, metodologi penulisan, sistimatika penulisan dan kontribusi penelitian.

BAB II : LANDASAN TEORI

Menguraikan pembahasan sistem tenaga listrik dan operasi ekonomis (*economic dispatch)* dan karakteristik unit pembangkit secara umum.

BAB III : *OPTIMAL POWER FLOW* MENGGUNAKAN METODE *PARALLEL EVOLUTIONARY PROGRAMMING*

Menguraikan teori dasar dari aliran daya, metode aliran daya *Newton Raphson,* teori EP, adaptasi EP ke permasalahan OPF dan prosedur dari PEP.

BAB IV : HASIL DAN ANALISIS HASIL

Menguraikan alur program, hasil validasi, serta hasil perhitungan OPF menggunakan metode PEP.

BAB V : KESIMPULAN DAN SARAN

Memuat intisari dan hasil pembahasan, yang berisikan kesimpulan dan saran yang dapat digunakan sebagai pertimbangan untuk pengembangan selanjutnya.

DAFTAR PUSTAKA

1.7. Kontribusi Penelitian

Penggunaan dari metode PEP dalam menganalisa permasalahan OPF dapat membantu pemerintah khususnya PT. PLN (persero) dalam mengatasi masalah pembangkitan dalam sistem tenaga listrik guna menghasilkan operasi yang andal dan ekonomis khususnya dalam biaya pembangkitan. Maka kami berharap agar metode ini dapat dijadikan sebagai bahan pertimbangan oleh PT. PLN (persero) dalam pemecahan masalah OPF.

BAB II

SISTEM TENAGA LISTRIK, OPERASI EKONOMIS DAN KARAKTERISTIK UNIT PEMBANGKIT

2.1. Sistem Tenaga Listrik[3]

Sistem tenaga listrik ada tiga bagian utama yaitu: pusat pembangkit tenaga listrik, saluran transmisi serta sistem distribusi yang berhubungan langsung dengan konsumen. Saluran transmisi merupakan penghubung antara pusat pembangkit melalui hubungan antar sistem yang menuju sistem pada sistem yang lain.

Saluran transmisi mempunyai empat parameter yang mempengaruhi kemampuannya dalam menyalurkan daya listrik. Keempat parameter tersebut yaitu : resistansi (R), induktansi (L), kapasitansi (C), serta konduktansi (G).

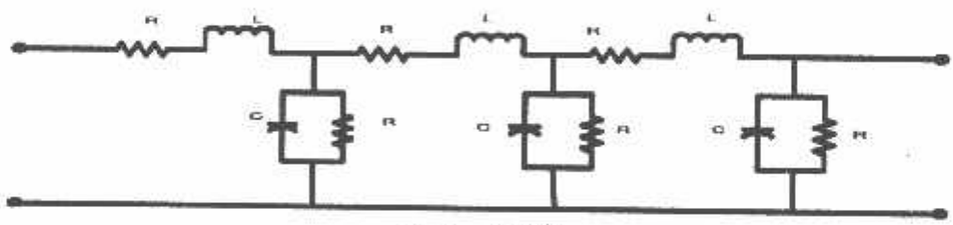
Resistansi umumnya tergantung pada jenis penghantar sedangkan konduktansi menyatakan besarnya arus bocor antar penghantar, antar penghantar dengan tanah, tetapi harganya relatif kecil maka dapat diabaikan. Induktansi adalah parameter rangkaian yang menghubungkan tegangan yang diimbaskan oleh perubahan fluksi akibat perubahan arus, sedangkan kapasitansi suatu saluran transmisi timbul akibat adanya beda potensial antara penghantar dengan tanah, dalam hal ini kapasitansi menyebabkan penghantar bermuatan seperti yang terjadi pada pelat kapasitor.

Impedansi seri terbentuk dari resistansi dan induktansi yang terbagi merata sepanjang saluran. Sedangkan konduktansi dan kapasitansi terdapat diantara

penghantar-penghantar dari saluran fasa tunggal atau diantar penghantar dengan netral dari suatu saluran berfasa tiga membentuk admitansi paralel.

2.2. Saluran Transmisi[3]

Tenaga listrik yang dibangkitkan harus disalurkan melalui saluran transmisi. Saluran – saluran transmisi ini membawa tenaga listrik dari pusat – pusat tenaga listrik ke pusat – pusat beban. Suatu saluran transmisi tenaga listrik mempunyai empat parameter yang mempengaruhi kemampuan untuk berfungsi sebagai bagian dari sistem tenaga, yaitu resistansi, induktansi, kapasitansi, dan konduktansi. Keempat parameter saluran transmisi tersebut merata disepanjang saluran transmisi. Parameter – parameter tersebut sangat berpengaruh terhadap tegangan bus dan aliran daya yang mengalir pada saluran tersebut.



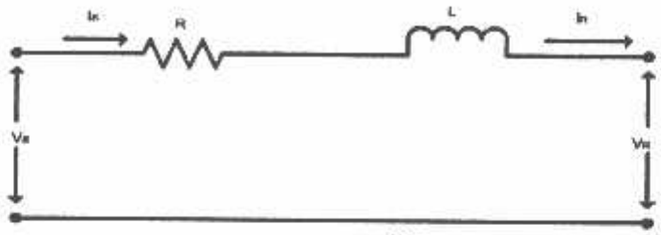
Gambar 2.1[3]
Rangkaian Setara Saluran Transmisi

Menurut panjangnya, saluran transmisi dapat dikasifikasikan menjadi 3 golongan, yaitu :

1. Saluran transmisi pendek, adalah saluran yang panjangnya < 80 km.
2. Saluran transmisi menengah, adalah saluran transmisi yang panjangnya 80 – 240 km.
3. Saluran transmisi panjang, adalah saluran yang panjangnya > 240 km.

2.2.1. Saluran Transmisi Pendek[3]

Rangkaian ekivalen untuk saluran transmisi pendek diperlihatkan pada gambar 2, dimana I_S dan I_R merupakan arus pada ujung pengirim dan ujung penerima, sedangkan V_S dan V_R adalah tegangan saluran terhadap netral pada ujung pengirim dan ujung penerima.



Gambar 2.2.[3]
Rangkaian Setara Saluran Transmisi Pendek

Karena tidak ada cabang pararel (shunt), arus pada ujung – ujung pengirim dan penerima akan sama besar :

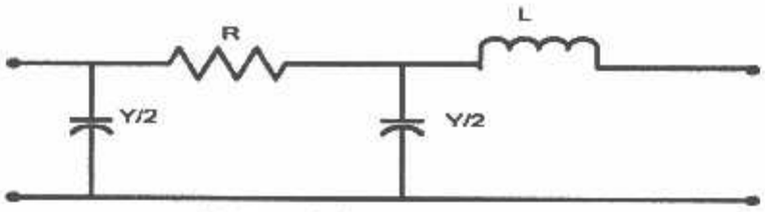
$$I_S = I_R \quad \text{..... (2.1.)}$$

Bila kondisi tegangan pada ujung penerima diketahui, maka tegangan pada ujung sisi kirim adalah :

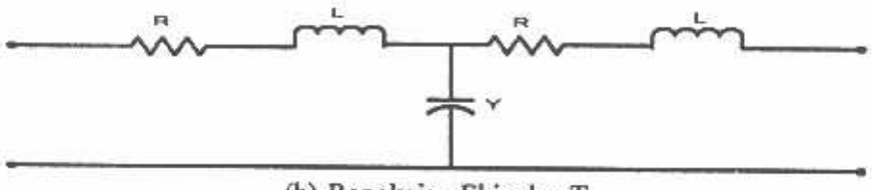
$$V_S = V_R + I_R Z \quad \text{..... (2.2.)}$$

2.2.2. Saluran Transmisi Menengah[3]

Bertambahnya saluran menyebabkan kapasitansi shunt bertambah besar dan tidak dapat diabaikan. Saluran transmisi jarak menengah pada umumnya digambarkan dengan rangkaian π atau rangkaian T. Dari dua versi ini rangkaian π lebih umum dipakai dari pada rangkaian T.



(a) Rangakaian Ekivalen π



(b) Rangkaian Ekivalen T

Gambar 2.3.[3]
Rangkaian Setara Saluran Transmisi Menengah

Untuk rangkaian π berlaku :

$$V_s = \left[\frac{ZY}{2}+1\right]V_R + ZI_R \quad \text{.....(2.3.)}$$

$$I_S = \left[\frac{ZY}{4}+1\right]YV_R + \left[\frac{ZY}{2}+1\right]I_R \quad \text{.....(2.4.)}$$

Untuk rangkaian T berlaku :

$$V_S = \left[\frac{ZY}{2}+1\right]V_R + \left[\frac{ZY}{4}+1\right]ZI_R \quad \text{.....(2.5.)}$$

$$I_S = YV_R + \left[\frac{ZY}{2}+1\right]I_R \quad \text{.....(2.6.)}$$

Dimana :

V_S, I_S = Tegangan, Arus sisi kirim

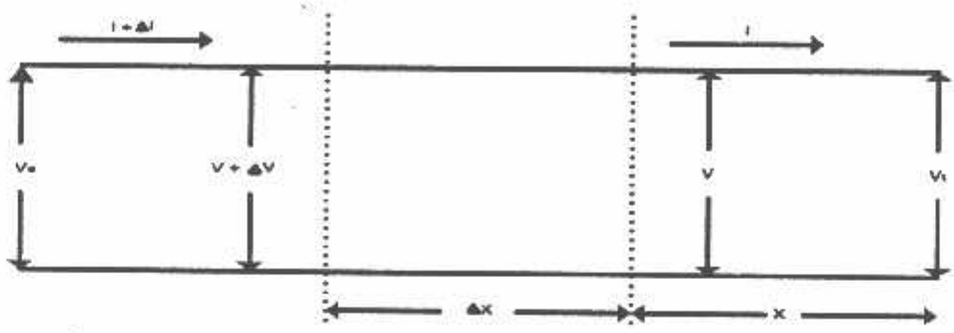
V_R, I_R = Tegangan, Arus sisi terima

Z = Impedansi seri total saluran transmisi

Y = Admitansi shunt total saluran

2.2.3. Saluran Transmisi Panjang[3]

Pada saluran panjang parameter – parameter saluran tidak terpusat menjadi satu, melainkan tersebar merata diseluruh panjang saluran.



Gambar 2.4.[3]
Diagram Skema Saluran Transmisi Panjang

Persamaan tegang dan arus pada setiap titik sepanjang saluran transmisi dengan jarak x dari ujung sisi terima dapat ditulis sebagai berikut :

$$V = \frac{V_R + I_R Z_C}{2} e^{yx} + \frac{V_G - I_G Z_C}{2} e^{-yx} \quad \text{.....(2.7.)}$$

$$I = \frac{\frac{V_R}{Z_C} + I_R}{2} e^{yx} + \frac{\frac{V_R}{Z_C} - I_R}{2} e^{-yx} \quad \text{.....(2.8.)}$$

Persamaan untuk saluran transmisi panjang dapat ditulis dalam bentuk hiperbola sebagai berikut :

$$V_{(x)} = V_R \cosh \tau x + I_R Z_C \sinh \tau x \quad \text{.....(2.9.)}$$

$$I_{(x)} = I_R \cosh \tau x + \frac{V_R}{Z_C} \sinh \tau x \quad \text{.....(2.10.)}$$

Dimana :

τ = Konstanta rambatan pada saluran $= \sqrt{ZY}$

Z_c = Impedansi karakteristik saluran $= \sqrt{\frac{Z}{Y}}$

2.3. Sistem Per-Unit[3]

Untuk memudahkan proses perhitungan, dalam sistem tenaga listrik digunakan sistem Per-Unit (pu).

$$\text{Besaran Per-Unit} = \frac{\text{Besaran sebenarnya}}{\text{Besaran dasar}} \quad \text{.....(2.11.)}$$

Rumus – rumus yang digunakan untuk penentuan arus dasar dan impedansi dasar adalah :

➢ Untuk data 1 phasa

Arus dasar

$$I_d = \frac{\text{kVA dasar 1 fasa}}{\text{kV dasar L - N}} \quad \text{.....(2.12.)}$$

Impedansi dasar

$$Z_d = \frac{(\text{kV dasar L - N})^2 \times 10^3}{\text{kVA dasar 1 fasa}} \quad \text{.....(2.13)}$$

$$Z_d = \frac{(\text{kV dasar L - N})^2}{\text{MVA dasar 1 fasa}} \quad \text{.....(2.14.)}$$

➢ Untuk data 3 fasa

Arus dasar

$$I_d = \frac{\text{kVA dasar 3 fasa}}{\sqrt{3}\ \text{kV dasar L - L}} \quad \text{.....(2.15.)}$$

Impedansi dasar

$$Z_d = \frac{(\text{kV dasar L - L})^2 \times 10^{-3}}{\text{kVA dasar3 fasa}} \quad \text{.....(2.16.)}$$

$$Z_d = \frac{(\text{kV dasar L - L})^2}{\text{MVA dasar 3 fasa}} \quad \text{.....(2.17.)}$$

Dalam persamaan di atas nilai – nilai besaran diberikan untuk rangkaian satu fasa. Jadi tegangannya adalah tegangan antara fasa dengan tanah dan daya setiap fasa.

Setelah besaran – besaran dasar telah ditentukan maka besaran – besaran itu dinormalisasikan terhadap besaran dasar. Dengan demikian impedansi per satuan dari suatu elemen rangkaian didefinisikan sebagai berikut :

$$Z = \frac{\text{Impedansi sebenarnya Z } (\Omega)}{\text{Impedansi dasar } Z_d} \quad \text{.....(2.18.)}$$

2.3.1. Mengubah Dasar Sistem Per-Unit[3]

Kadang-kadang impedansi per-unit untuk suatu komponen dari suatu sistem dinyatakan menurut dasar yang berbeda dengan dasar yang dipilih untuk bagian dari sistem dimana komponen tersebut berada. Karena semua impedansi dalam bagian manapun dari suatu sistem harus dinyatakan dengan dasar impedansi yang sama, maka dalam perhitungannya kita perlu mempunyai cara untuk dapat mengubah impedansi per-unit dari suatu dasar ke dasar yang lain. Dengan mensubstitusikan impedansi dasar yang diberikan dalam persamaan (2.14.) dan (2.17.) ke dalam persamaan (2.18.) maka diperoleh:

$$Z_u = \frac{(\text{Impedansi sebenarnya, } \Omega) \cdot (\text{MVA dasar})}{(\text{Tegangan dasar, kV})^2} \quad \text{.....(2.19.)}$$

Persamaan (2.17.) memperlihatkan bahwa impedansi per-unit berbanding lurus dengan MVA dasar serta berbanding terbalik dengan kuadrat tegangan dasar. Untuk mengubah dari impedansi per-unit menurut suatu dasar yang diberikan menjadi impedansi per-unit menurut suatu dasar yang baru, dapat dipakai persamaan berikut:

$$Z_{baru}\,\text{per-unit} = Z_{diberikan}\,\text{per-unit}\left(\frac{kV_{diberikan}\,\text{dasar}}{kV_{baru}\,\text{dasar}}\right)^2 \times \left(\frac{MVA_{baru}\,\text{dasar}}{MVA_{diberikan}\,\text{dasar}}\right) \quad \text{..... (2.20.)}$$

Persamaan ini tidak ada hubungannya dengan pemindahan nilai impedansi dalam ohm dari salah satu sisi transformator ke sisi yang lain.

2.4. Sistem Operasi Pada Sistem Tenaga Listrik[4]

Seperti telah diketahui bahwa dalam masalah pengaturan beban pada suatu operasi sistem tenaga listrik harus selalu dicapai suatu keadaan operasi yang bisa diandalkan dan cukup ekonomis.

Ada beberapa kerja yang harus dilaksanakan untuk menjamin keandalan sistem operasi antara lain, pengaturan frekuensi dan tegangan sistem untuk berada pada harga normalnya karena adanya perubahan beban pada sistem. Dan seperti yang diketahui dan berulang kali disebutkan bahwa tenaga listrik tidak dapat disimpan sehingga dalam operasinya harus selalu dicapai keseimbangan antara penyediaan dengan pemenuhan kebutuhan daya serta perlu juga diingat bahwa sistem selalu berubah setiap saat. Maka sudah tentu jauh-jauh sebelumnya sudah harus diketahui atau diramalkan keadaan tersebut dengan tepat yaitu keadaan beban pada hari itu dari waktu ke waktu sampai selama 24 jam. Keadaan beban ini

digambarkan sebagai kebutuhan daya sebagai fungsi dari waktu yang biasa disebut dengan lengkung beban harian. Lengkung beban harian ini adalah merupakan sesuatu yang sangat penting disamping karakteristik-karakteristik lainnya sehingga dalam operasi hariannya harus berdasarkan lengkung beban harian yang telah dibuat karena dengan lengkung beban harian ini dapat ditentukan perencanaan operasi pembangkit-pembangkit yang ada, baik itu unit pembangkit thermal maupun hidro. Tentu saja kebutuhan beban dalam suatu harinya tidak merata akan tetapi dari jam ke jam berbeda sesuai dengan kebutuhan konsumen. Berdasarkan lengkung beban yang telah ada maka dapat ditentukan berapa unit pembangkit yang harus bekerja dan siap bekerja pada hari itu.

Sebagai dasar pertimbangan yang sifatnya umum, untuk menentukan biaya produksi tenaga listrik yang dibutuhkan adalah dengan memperhatikan bahwa dalam keadaan beban minimum maka tenaga listrik yang dibutuhkan diberikan oleh unit pembangkit yang bekerja paling efisien pada keadaan tersebut. Pembangkit ini akan terus beroperasi atau dibebani sampai pada batas efesiensi maksimumnya. Dan apabila ternyata beban masih terus bertambah sedangkan unit pembangkit ini telah mencapai maksimumnya maka selanjutnya beban ditanggung oleh unit pembangkit yang lain yang belum mencapai efesiensi maksimumnya. Dengan dasar operasi yang demikian maka dapat dicapai keadaan operasi yang cukup ekonomis.

Akan tetapi dengan semakin berkembangnya sistem itu sendiri maka diperlukan suatu perencanaan pembangkitan yang optimum dengan biaya operasi yang ekonomis dan harus memperhitungkan rugi-rugi yang terjadi pada saluran transmisi. Mengingat bahwa beban sistem adalah selalu berubah-ubah dari waktu

ke waktu maka perlu untuk membuat secara grafis perubahan beban terhadap waktu.

Oleh karena biaya operasi untuk memproduksi daya listrik, suatu pembangkit hidro (PLTA) sangat kecil jika dibandingkan dengan pembangkit thermal (PLTU, PLTG, PLTGU, PLTD) maka pembahasan selanjutnya untuk mendapatkan biaya operasi yang ekonomis sebagian besar ditekan pada unit pembangkit thermal saja karena disini akan membutuhkan biaya operasi yang cukup tinggi sehingga usaha penghematan biaya bahan bakar akan sangat berarti. Dengan kata lain dengan mengkoordinasikan operasi pembangkit-pembangkit yang tersedia dengan tepat dan sesuai dengan beban maka akan didapat suatu keadaan operasi yang ekonomis.

Pembahasan mengenai operasi ekonomis adalah merupakan salah satu cara bagaimana menekan biaya produksi dari sistem tenaga listrik. Dalam hal ini maka metode yang dipakai adalah dengan memanfaatkan karakteristik dari menganalisa operasi dari sistem tersebut. Disamping karakteristik dari unit-unit pembangkit perlu juga diketahui karakteristik beban, karena karakteristik bebanlah maka dapat dianalisa pengaturan yang paling ekonomis dari setiap unit pembangkit. Adapun karakteristik yang perlu diketahui dari setiap unit pembangkit adalah :

1. Karakteristik *input* bahan bakar sebagai fungsi dari *output* daya.
2. Nilai panas sebagi fungsi *output* daya.
3. Kenaikan jumlah bahan bakar yang dibutuhkan jika terdapat perubahan beban.

Ketiga karakteristik tersebut merupakan pedoman menganalisa penjadwalan selanjutnya. Kemudian yang juga perlu diperhitungkan adalah variabel-variabel yang terdapat pada saluran transmisi, karena variabel-variabel

ini juga sangat menentukan ekonomis tidaknya penjadwalan pembangkit yang kita tentukan.

Maka untuk mencapai suatu operasi yang ekonomis pada suatu sistem tenaga listrik adalah dengan melakukan penjadwalan pada sistem pembangkit yang ada pada suatu sistem tenaga listrik yang ditinjau tersebut dengan memanfaatkan karakteristik dari setiap masing-masing unit pembangkit yang ada pada dasarnya bertujuan untuk menekan biaya produksi listrik agar harga dari listrik yang dihasilkan dapat ditekan serendah mungkin sehingga dapat memuaskan pemakai listrik.

2.5. Karakteristik Unit Pembangkit[4]

2.5.1. Karakteristik *Input-Output*[4]

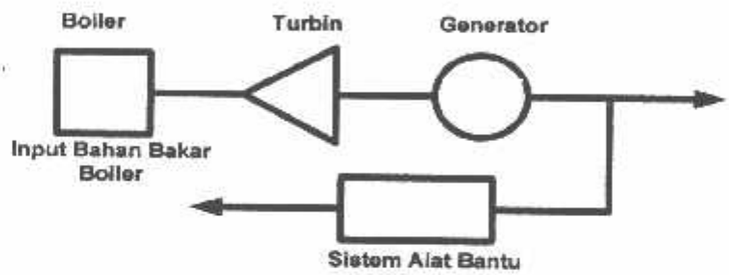
Hal yang paling mendasar dalam operasi pembangkitan yang ekonomis adalah dengan membuat karakteristik *input-output* dari unit pembangkit thermal. Karena ini diperoleh dari desain perencanaan atau melalui test pembangkit. Adapun definisi dari karakteristik *input-output* dari pembangkit itu sendiri adalah formula yang menyatakan hubungan antara *input* pembangkit sebagai fungsi dari *output* pembangkit. Sedangkan ciri dari unit boiler-turbin-generator dapat digambarkan dalam gambar 2.5., dimana unit ini memuat sebuah boiler yang menghasilkan uap untuk menjalankan turbin yang dikopel dengan rotor dari generator.

Pada pembangkit thermal input diberikan dalam satuan panas Btu/jam atau Kalori/jam dari bahan bakar yang diberikan boiler untuk menghasilkan *output* pembangkit. Sedangkan notasi yang digunakan adalah H (*MBtu/h*) atau dalam

satuan yang lain H (*MKal/h*). Adapun dalam skripsi ini, perhitungan dilakukan adalah dalam satuan *MKal/jam*. Selain itu *input* dari pembangkit dapat pula dinyatakan dalam nilai uang yang menyatakan besarnya biaya yang diperlukan untuk bahan bakar. Notasi yang digunakan adalah F (*Rp/h*). Hubungan antara H dan F dapat dinyatakan dalam rumus sebagai berikut ini :

$$F = H \times \frac{Rupiah}{MBtu} \quad \text{.....(2.21.)}$$

Dimana $\frac{Rupiah}{MBtu}$ adalah nilai uang yang diperlukan per satuan panas dari bahan bakar.



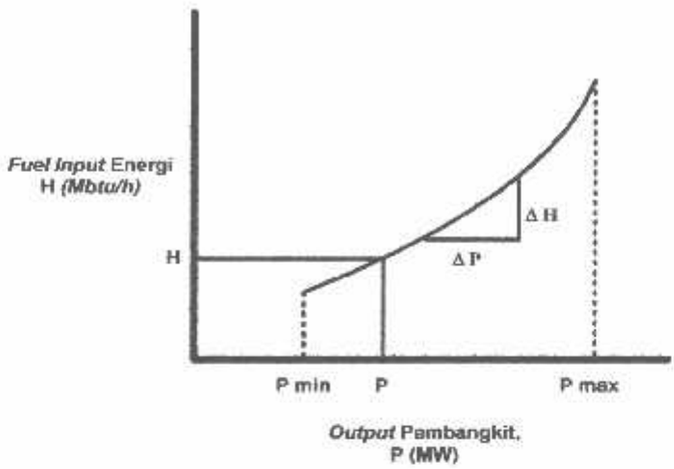
Gambar 2.5.[4]
Unit Boiler-Turbin-Generator

Seperti digambarkan dalam gambar 2.5., maka *output* dari pembangkit tidak hanya dihubungkan dengan sistem saja akan tetapi juga untuk sistem peralatan bantu pembangkit didefinisikan sebagai daya yang dikeluarkan oleh generator karakteristik *input-output* , daya *output* adalah berupa daya netral dari pembangkit, notasi yang digunakan adalah P (MW).

Persamaan karakteristik *input-output* pembangkit dapat dilihat pada persamaan (2.22.) dan (2.23.) dibawah ini, sedangkan kurva dari karakteristik *input-output* pembangkit dapat dilihat pada gambar 2.6.

$H = f(P)$, atau .....(2.22.)

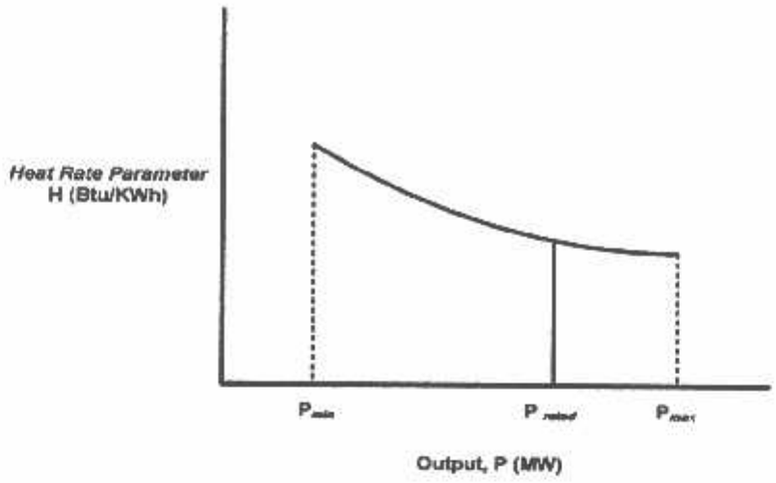
$$F = f(P) \quad \text{.....(2.23.)}$$



Gambar 2.6. [4]
Kurva Karakteristik *Input-Output* Pembangkit Thermal

2.5.2. Karakteristik *Heat-Rate* [4]

Karakteristik *heat-rate* merupakan karakteristik yang menunjukkan efesiensi dari sebuah mesin. Karakteristik *heat-rate* sebuah unit pembangkit menunjukkan *input* kalor yang diberikan untuk menghasilakn energi sebesar 1 kiloWatt jam pada MegaWatt *output* dari suatu unit. Kurva dari karakteristik *heat-rate* ini dapat dilihat pada gambar 2.7. di bawah ini.



Gambar 2.7.[4]
Kurva Karakteristik *Heat-Rate* Unit Pembangkit

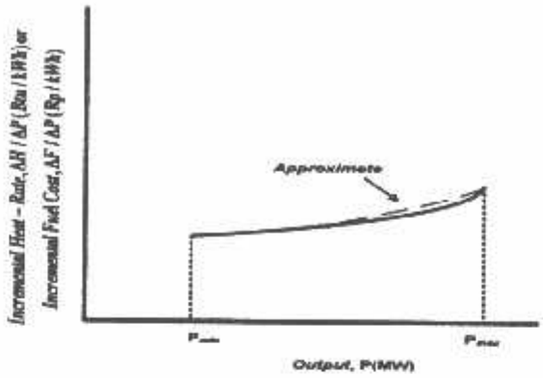
2.5.3. Karakteristik *Incremental Heat-Rate* dan *Incremental Fuel Cost*[4]

Perwujudan yang lain dari karakteristik pembangkit adalah karakteristik *Incremental Heat-Rate* atau perubahan tingkat laju panas dan karakteristik *Incremental Fuel Cost* atau perubahan tingkat laju bahan bakar. Karakteristik ini menyatakan hubungan daya keluaran pembangkit sebagai fungsi *Incremental Heat-Rate* atau *Incremental Fuel Cost*. Karakteristik *Incremental Heat-Rate* ini menunjukkan besarnya perubahan *input* energi bila ada perubahan *output* pada unit pembangkit.

Kurva dari karakteristik *Incremental Heat-Rate* atau *Incremental Fuel Cost* dapat dilihat pada gambar 2.8. Sedangkan persamaan *Incremental Heat-Rate* dan persamaan *Incremental Fuel Cost* dapat dilihat pada persamaan (2.24.) hingga persamaan (2.27.).

$$\textit{Incremental Heat-Rate} = \frac{\Delta H}{\Delta P}\left(\frac{MBtu}{kWh}\right) \quad \text{..... (2.24.)}$$

$$Incremental\ Fuel\ Cost = \frac{\Delta F}{\Delta P}\left(\frac{Rupiah}{kWh}\right) \quad \text{..... (2.25.)}$$



Gambar 2.8.[4]
Kurva Karakteristik *Incremental Heat-Rate/Fuel Cost*

Bila harga Δ sangat kecil maka dapat dinyatakan dengan persamaan berikut ini :

$$Incremental\ Heat\text{-}Rate = \frac{dH}{dP}\left(\frac{MBtu}{kWh}\right) \quad \text{..... (2.26.)}$$

$$Incremental\ Fuel\ Cost = \frac{dF}{dP}\left(\frac{Rupiah}{kWh}\right) \quad \text{..... (2.27.)}$$

2.6. *Economic Dispatch*[4]

Dalam pembahasan tentang OPF dan operasi pada sistem tenaga listrik yang ekonomis, maka kita selalu membicarakan *economic dispatch*. *Economic dispatch* adalah pembagian pembebanan pada pembangkit – pembangkit yang ada dalam suatu sistem tenaga listrik, secara optimum dan ekonomis pada beban tertentu. Dengan dilakukan *economic dispatch* maka akan didapatkan biaya bahan bakar yang paling murah dalam suatu sistem pembangkit. Oleh karena beban yang harus ditanggung oleh sistem pembangkit selalu berubah

setiap periode waktu tertentu, maka perhitungan *economic dispatch* ini dilakukan untuk setiap harga beban tertentu pula.

2.6.1. Fungsi Biaya Bahan Bakar[4]

Biaya bahan bakar merupakan unsur biaya yang penting dalam operasi sistem pembangkit thermal. Fungsi biaya bahan bakar $F_i(P_i)$ untuk setiap pembangkit terhadap daya keluaran diekspresikan dalam bentuk fungsi kuadrat, yang dapat dinyatakan sebagai berikut:

$$F_i(P_i) = a_2 P_i^2 + a_1 P_i + a_0 \quad \text{..... (2.28.)}$$

Dimana:

a, b, c = konstanta persamaan dari unit ke - i

P_i = daya keluaran dari unit ke - i

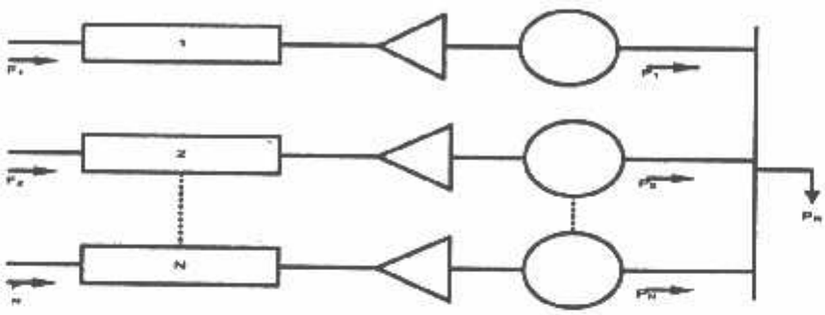
Dalam pengoperasian secara ekonomis adalah penting untuk mengetahui biaya bahan bakar yang digunakan untuk membangkitkan daya yang diperlukan, yaitu :

- Jenis bahan bakar
- Nilai kalori
- Harga bahan bakar.

2.6.2. *Economic Dispatch* Dengan Mengabaikan Rugi-rugi Transmisi[4]

Dalam sistem tenaga listrik, kerugian transmisi merupakan kehilangan daya yang harus ditanggung oleh sisi pembangkit. Jadi dengan adanya kerugian daya tersebut merupakan tambahan beban bagi sistem tenaga listrik.

Sistem dengan mengabaikan rugi-rugi transmisi dapat dilihat pada gambar 2.9. Sistem ini terdiri dari N buah pembangkit thermal yang dihubungkan pada *single bus bar* yang melayani beban P_R. *Input* dari masing-masing pembangkit ditunjukkan oleh F_i yang mewakili biaya dari satu unit pembangkit dan *output* dari masing-masing unit P_i adalah daya yang dihasilkan oleh satu unit pembangkit.



Gambar 2.9.[4]
N Unit Pembangkit Thermal Melayani Beban P_R

Total biaya rata-rata yang harus ditanggung oleh sistem adalah jumlah biaya dari masing-masing unit pembangkit. Dan pembatas yang paling penting adalah jumlah *output* dari masing-masing unit pembangkit sama dengan beban di konsumen. Yang menjadi permasalahan adalah meminimumkan total biaya F_T dengan memperhatikan pembatas ϕ bahwa daya yang dihasilkan oleh pembangkit sama dengan beban yang diterima. Secara matematika pernyataan yang tersebut di atas dapat dinyatakan dengan persamaan sebagai berikut :

$$F_T = F_1 + F_2 + F_3 + ... + F_N \quad \text{..... (2.29.)}$$

$$= \sum_{i=1}^{N} F_i(P_i)$$

$$\phi = 0 = P_R - \sum_{i=1}^{N} P_i \quad \text{..... (2.30.)}$$

Persamaan di atas adalah pembatas yang merupakan problem dari optimasi dan ini dapat dipecahkan dengan menggunakan kalkulus tingkat lanjut yang melibatkan fungsi La Grange. Dimana fungsi ini didapat dengan cara menambahkan pembatas ϕ yang telah dikalikan dengan faktor pengali La Grange λ pada fungsi tujuan F_T. Fungsi La Grange dapat ditunjukkan dengan persamaan di bawah ini :

$$L = F_T + \lambda \cdot \phi \quad \text{..... (2.31.)}$$

Persamaan La Grange di atas merupakan fungsi *output* pembangkit P_i dan faktor pengali La Grange λ. Keadaan dari optimal dari fungsi tujuan F_T dapat diperoleh dengan operasi gradient dari persamaan La Grange sama denga nol.

$$\nabla L = 0 \quad \text{..... (2.32.)}$$

$$\nabla F_T + \lambda \cdot \phi = 0 \quad \text{..... (2.33.)}$$

$$\frac{\partial L}{\partial P} = \frac{\partial F_i}{\partial P_i} + \lambda \cdot \left(\frac{\partial P_R}{\partial P_i} - \frac{\partial P_i}{\partial P_i} \right) = 0 \quad \text{..... (2.34.)}$$

$$\text{atau } \frac{\partial F_i}{\partial P_i} + \lambda \cdot (0 - 1) = 0 \quad \text{..... (2.35.)}$$

$$\frac{\partial F_i}{\partial P_i} = \lambda \quad \text{..... (2.36.)}$$

Persamaan terakhir ini menunjukkan bahwa bila digunakan biaya bahan bakar F_T yang paling minimum maka *incremental cost* setiap unit generator pembangkit harus sama yaitu sebesar λ. Kondisi optimal ini tentunya dengan tetap memperhatikan pembatas yang ada yaitu bahwa daya dari setiap unit generator

pembangkit harus lebih besar atau sama dengan daya *output* minimum dan lebih kecil atau sama dengan daya *output* maksimum yang diijinkan.

Dari N buah pembangkit yang ada dalam sistem tenaga yang telah dibahas dan beban sistem sebesar P_R, maka dapat diambil kesimpulan sebagai berikut :

$$\frac{\partial F_i}{\partial P_i} = \lambda \quad \text{ada N buah persamaan} \qquad (2.37.)$$

$$P_{i\,\min} \le P_i \le P_{i\,\text{maks}} \quad \text{ada 2 N buah pertidaksamaan} \qquad (2.38.)$$

$$\sum_{i=1}^{N} P_i = P_R \qquad (2.39.)$$

Dari batasan pertidaksamaan pembatas di atas dapat diperluas menjadi :

$$\frac{\partial F_i}{\partial P_i} = \lambda \quad \text{untuk} \quad P_{i\,\min} \le P_i \le P_{i\,\text{maks}} \qquad (2.40.)$$

$$\frac{\partial F_i}{\partial P} \le \lambda \quad \text{untuk} \quad P_i = P_{i\,\text{maks}} \qquad 2.41.)$$

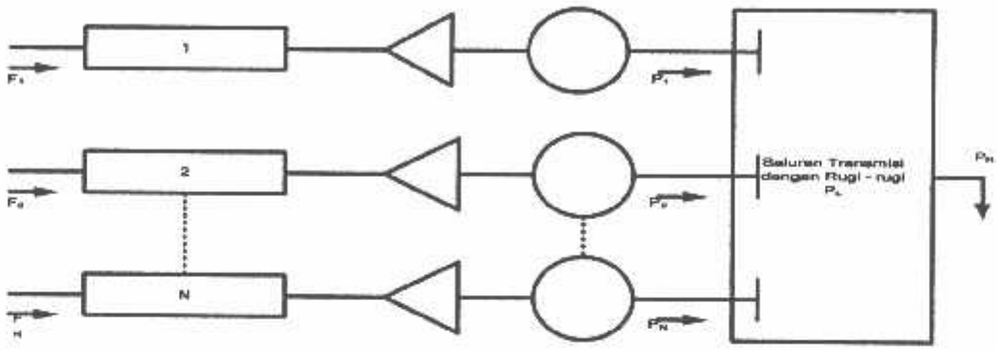
$$\frac{\partial F_i}{\partial P_i} \ge \lambda \quad \text{untuk} \quad P_i = P_{i\,\min} \qquad (2.42.)$$

Karena F_i hanya sebagai fungsi P_i maka $\frac{\partial F_i}{\partial P}$ dapat diganti dengan $\frac{dF_i}{dP_i}$.

2.6.3 *Economic Dispatch* Dengan Memperhitungkan Rugi-Rugi Transmisi[4]

Sistem dengan memperhitungkan rugi – rugi transmisi dapat dilihat pada gambar 2.10. Sistem ini terdiri dari N buah unit pembangkit thermal dihubungkan melalui saluran transmisi yang melayani beban P_R. *Input* dari masing –masing unit ditunjukkan oleh F_i yang mewakili biaya dari satu unit pembangkit dan *output* dari masing - masing unit P_i adalah daya yang dihasilkan olch satu unit pembangkit.

Total biaya rata – rata yang harus ditanggung oleh sistem adalah jumlah dari biaya dari masing – masing unit pembangkit. Dan pembatas yang paling penting adalah bahwa jumlah output dari masing – masing unit pembangkit sama dengan beban di konsumen dan rugi – rugi transmisi.



Gambar 2.10.[4]
N buah Pembangkit Thermal Melayani Beban P_R Melalui Saluran Transmisi

$$P_R + P_L - \sum_{i=1}^{N} P_i = \phi = 0 \quad \text{..... (2.43.)}$$

$$L = F_T + \lambda\phi \quad \text{..... (2.44.)}$$

$$\phi = \sum_{i=1}^{N} P_i - P_R - P_L = 0 \quad \text{..... (2.45.)}$$

Persamaan La Grange nya adalah:

$$L = \sum_{i=1}^{N} F_i - \lambda\left(\sum_{i=1}^{N} P_i - P_R - P_L\right) \quad \text{..... (2.46.)}$$

$$\frac{\partial L}{\partial P_i} = \frac{dF_i}{dP_i} - \lambda\left(1 - \frac{\partial P_L}{\partial P_i}\right) = 0 \quad \text{..... (2.47.)}$$

BAB III

OPTIMAL POWER FLOW MENGGUNAKAN METODE *PARALLEL EVOLUTIONARY PROGRAMMING*

3.1. Analisa Aliran Daya[3][5]

Dalam melayani beban yang dibutuhkan oleh konsumen dan pengoperasian tenaga listrik perlu dilakukan penganalisaan aliran daya, sehingga sistem yang dioperasikan dapat memenuhi persyaratan teknis maupun ekonomisnya. Dalam analisa aliran daya dilakukan perhitungan terhadap tegangan, arus, daya aktif dan reaktif, yang terdapat dalam berbagai titik dalam jala – jala jaringan transmisi tenaga listrik.

Tujuan dari analisa daya adalah:

1. Mencari harga magnitude tegangan $|v|$ dan sudut fasa tegangan δ bus beban.
2. Mencari daya reaktif Q dan sudut fasa tegangan δ dari generator bus.
3. Untuk mendapatkan daya aktif dan daya reaktif pada bus slack.
4. Untuk mengetahui apakah semua peralatan pada sistem memenuhi batas – batas yang telah ditetapkan untuk operasi penyaluran daya.
5. Untuk mengetahui kondisi awal pada perencanaan sistem yang baru.
6. Untuk menentukan daya yang mengalir disetiap saluran jaringan tenaga listrik.

3.1.1. Klasifikasi Bus[3][5]

Pada setiap bus dari jaringan terdapat parameter – parameter yaitu : daya aktif (P), daya reaktif (Q), besar tegangan $|v|$ dan sudut fasa tegangan δ.

Dengan melihat parameter diatas, setiap bus dapat diklasifikasikan menjadi 3 bagian:

1. Bus Beban (Load Bus) (PQ)

Pada bus ini hanya terdapat kebutuhan daya untuk beban dimana P daya aktif dan Q daya reaktif diketahui, sementara $|v|$ dan δ berubah – ubah menurut kebutuhan. Oleh karena itu, $|v|$ dan δ harus ditentukan (dicari).

2. Bus Generator (PV)

Pada bus ini hanya terdapat daya pembangkitan dimana $|v|$ diatur menggunakan regulator tegangan dan P diatur dengan governor. Sehingga untuk bus ini P dan $|v|$ diketahui. Sementara Q (daya reaktif) dan δ (sudut fasa) dicari.

3. Bus Slack

Pada bus ini $|v|$ dan δ sudah ditentukan besarnya sementara P dan Q dihitung. Biasanya nilai $|v|$ adalah 1 pu, sedangkan sudut fasa tegangan δ berharga nol, karena itu fasor tegangan dari bus dipakai sebagai referensi.

Daya total yang mengalir pada setiap bus dituliskan sebagai berikut :

$$S_k = P_k + jQ_k = V_k^* I_k$$

Atau

$$P_k - jQ_k = V_k^* \sum_{n=1}^{N} Y_{kn} V_n \quad \text{.....(3.1.)}$$

Dari persamaan $V_k = v_k + j\delta_k$ dan $Y_{k11} = G_{k11} - jB_{k11}$, maka persamaan 3.1. menjadi :

$$P_k - jQ_k - (v_k + j\delta_k)\sum_{n=1}^{N}(G_{k11} - jB_{k11})(V_n + j\delta_n) \quad \text{.....(3.2.)}$$

Bila dituliskan dalam bentuk real dan imajiner maka persamaan di atas menjadi :

$$P_k = \sum_{n=1}^{N}\{v_k(v_k G_{kn} + \delta_k B_{kn}) + \delta_k(\delta_{kn} G_{kn} - v_k B_{kn})\} \quad \text{.....(3.3.)}$$

$$Q_k = \sum_{n=1}^{N}\{\delta_k(v_n G_{kn} + \delta_n B_{kn}) - v_k(\delta_n G_{kn} - v_n B_{kn})\} \quad \text{.....(3.4.)}$$

3.1.2. Metode Newton Rapshon[6]

Proses yang dilakukan adalah membandingkan antara daya yang ditempatkan berdasarkan data ($P_{k,\,sched}$ dan $Q_{k,\,sched}$) dengan daya hasil perhitungan ($P_{k,\,calc}$ dan $Q_{k,\,calc}$) menggunakan persamaan (3.3.) dan (3.4.) di atas. Selisih daya yang diterapkan dan perhitungan (ΔP_k dan ΔQ_k) dihitung dengan persamaan :

$$\Delta P_k = P_{k,\,sched} - P_{k,\,calc} \quad \text{.....(3.5.)}$$

$$\Delta Q_k = Q_{k,\,sched} - Q_{k,\,calc} \quad \text{.....(3.6.)}$$

Selisih daya dihitung dengan persamaan (3.5.) dan persamaan (3.6.) digunakan untuk menghitung nilai perubahan parameter tegangan bus, yaitu $\Delta|V_k|$ dan $\Delta\delta_k$, yaitu dengan menggunakan elemen Jacobian, sehingga koreksi terhadap nilai parameter tegangan yang telah ditetapkan nilai awal sebelumnya. Elemen Jacobian sendiri merupakan turunan parsial P dan Q terhadap masing-masing

variabel pada persamaan (3.3.) dan (3.4.), yang dalam bentuk matriks dituliskan sebagai:

$$\begin{bmatrix} \Delta P_1 \\ \cdots \\ \Delta P_{n-1} \\ \Delta Q_1 \\ \cdots \\ \Delta Q_{n-1} \end{bmatrix} = \begin{bmatrix} \frac{\partial P_1}{\partial v_1} & \cdots & \frac{\partial P_1}{\partial v_{n-1}} & \frac{\partial P_1}{\partial \delta_1} & \cdots & \frac{\partial P_1}{\partial \delta_{n-1}} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial P_{n-1}}{\partial v_1} & \cdots & \frac{\partial P_{n-1}}{\partial v_{n-1}} & \frac{\partial P_{n-1}}{\partial \delta_1} & \cdots & \frac{\partial P_{n-1}}{\partial \delta_{n-1}} \\ \frac{\partial Q_1}{\partial v_1} & \cdots & \frac{\partial Q_1}{\partial v_{n-1}} & \frac{\partial Q_1}{\partial \delta_1} & \cdots & \frac{\partial Q_1}{\partial \delta_{n-1}} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial Q_{n-1}}{\partial v_1} & \cdots & \frac{\partial Q_{n-1}}{\partial v_{n-1}} & \frac{\partial Q_{n-1}}{\partial \delta_1} & \cdots & \frac{\partial Q_{n-1}}{\partial \delta_{n-1}} \end{bmatrix} \begin{bmatrix} \Delta v_1 \\ \cdots \\ \Delta v_{n-1} \\ \Delta \delta_1 \\ \cdots \\ \Delta \delta_{n-1} \end{bmatrix}$$

$$\begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix} = \begin{bmatrix} J_1 & J_2 \\ J_3 & J_4 \end{bmatrix} \begin{bmatrix} \Delta \delta \\ \Delta |v| \end{bmatrix} \quad \text{.....(3.7.)}$$

Dimana elemen-elemen jacobian dapat dihitung dengan menurunkan persamaan-persamaan (3.3.) dan (3.4.)

Perubahan nilai tegangan bus dijumlahkan dengan nilai tegangan bus sebelumnya, yang kemudian nilai tegangan bus terbaru ini digunakan untuk menghitung kembali daya $P_{k,\,calc}$ dan $Q_{k,\,calc}$ menurut persamaan (3.3.) dan (3.4.) Proses ini terus berulang, yang disebut iterasi hingga mencapai kondisi dimana nilai perubahan daya ΔP dan ΔQ konvergen mencapai suatu nilai minimum yang telah ditentukan (berkisar 0,001 hingga 0, 0001 pu)

3.2. *Evolutionary Programming*[7][8][9]

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

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

EP ditemukan oleh Lawrence.J. Fogel 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 dari string-string bilangan real yang disebut *kromosom*. Seperti halnya alam, metode ini menyelesaikan permasalahan-permasalahan dengan menemukan kromosom-kromosom yang baik dengan memanipulasi materi dan sifat *(gene)* kromosom-kromosom. Algoritma ini tidak mengetahui tipe permasalahan yang akan diselesaikan.

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

EP memiliki empat dasar kerja yaitu :

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

3.2.1. Parameter *Evolutionary Programming*

Terdapat beberapa parameter yang digunakan dalam EP. Parameter tersebut digunakan untuk melihat kompleksitas dari EP. 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 EP). 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 EP. Jika ukuran populasi kecil maka populasi tidak menyediakan cukup materi untuk mencakup ruang permasalah, sehingga pada umumnya kinerja EP menjadi buruk. Dalam hal ini dibutuhkan ruang yang lebih besar untuk mempersentasikan keseluruhan ruang permasalahan. Selain itu penggunaan populasi yang besar dapat mencegah terjasinya konvergensi pada wilayah lokal.

❖ Probabilitas Mutasi *(Pm)*

Mutasi digunakan untuk meningkatkan variasi populasi digunakan untuk menentukan tingkat mutasi yang terjadi, karena frekuensi terjadinya mutasi tersebut menjadi *Pm x POPSIZE x N*, dimana N adalah panjang struktur/gen dalam satu individu. Probabilitas mutasi yang rendah akan menyebabkan gen-gen yang berpotensi tidak dicoba. Dan sebaliknya, tingkat mutasi yang tinggi akan menyebabkan keturunan akan semakin mirip dengan induknya.

Dalam EP mutasi menjalankan aturan penting yaitu :

1. Mengganti gen-gen yang hilang sama 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 genetik terkecil yaitu gen. Pengkodean memakai *string* bilangan *real*.

3.2.2. Mekanisme *Evolutionary Programming*

A. Pengkodean atau Representasi

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

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

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

B. Fungsi *Fitness* (Fungsi Evaluasi).

Dalam EP, 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[11]

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

➢ ***Roulette Wheel Selection***

Dimana setiap individual memiliki harga fitness schingga didapatkan probabilitas individual $\left(f(t)/\sum f(t)\right)$ tersebut dikopikan 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 fitnessnya.
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***

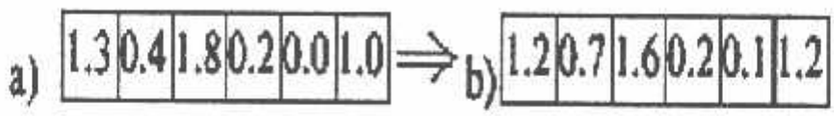
Apabila fitness yang dimiliki oleh suatu kromosom dalam populasi berbeda terlalu jauh dari kromosom lainnya maka hal ini dapat menjadi permasalahan. Misalnya bila kromosom terbaik mempunyai *fitness* yang menyebabkan besarnya tempat yang dimilikinya dalam *roulette wheel* sebesar 90% maka kromosom-kromosom 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 akan mempunyai fitness 2 dan seterusnya sampai yang terakhir akan mempunyai *fitness* N. Dengan demikian semua kromosom akan mempunyai peluang untuk diseleksi..

D. *Mutation* (Mutasi) [11][12]

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 *(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 Programming* sama dengan *Evolutionary Strategies* yaitu menggunakan operator *Gaussian mutation*, dimana setiap individu akan terpilih secara acak untuk mengalami mutasi berdasarkan nomor acak Gaussian untuk menciptakan individu baru (*offspring*)



Gambar 3.1.[13]
Ilustrasi
Mutasi Gaussian Dari Induk (parent) a) Menghasilkan Anak (*offspring*) b)

Fungsi dari operator mutasi adalah untuk menghindari agar solusi masalahan yang diperoleh bukan merupakan solusi optimum lokal. Tipe dan implementasi dari operator mutasi bergantung 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))[7]

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 dikompetisikan atau 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.

3.3. Formulasi Masalah *Optimal Power Flow* [7][8]

Masalah *optimal power flow* berupaya mengoptimalkan performa sistem pembangkitan daya dengan fungsi objektif f dan subyeknya berbagai batasan. Untuk pembagian optimal daya aktif dan reaktif, fungsi obyektif, f , adalah biaya total pembangkit. Tujuan lainnya yaitu mencakup peminimalan kerugian transmisi dan pengoptimalan tingkat tegangan. Secara matematis dapat dirumuskan sebagai berikut :

Min $f(x,u)$ .....(3.8.)

Subjeknya:

$g(x,u)=0$ .....(3.9)

$h(x,u)\le 0$ .....(3.10)

Dimana u adalah vektor kontrol variabel yang terdiri dari daya aktif generator dan tegangan ; x adalah vektor variabel tegangan pada bus beban (PQ), daya reaktif generator; $f(x,u)$ adalah fungsi yang akan dioptimalkan; $g(x,u)$ adalah batasan aliran daya; dan $h(x,u)$ adalah batasan fungsi penalti.

3.4. Adaptasi *Evolutionary Programming* Ke Masalah *Optimal Power Flow*[7][8]

Evolutionary Programming adalah suatu pada mekanisme seleksi alam. Individu dari sebuah populasi dikodekan secara real, Populasi pertama dibangkitkan secara random. Generasi baru dibuat dengan mengaplikasikan 3 operator berikut terhadap sebuah populasi yaitu : inisialisasi, mutasi dan kompetisi dimana tingkat pengoptimalan setiap calon solusi atau individu diukur dengan ketepatannya tergantung pada fungsi tujuan *(objective function)* permasalahan.

3.4.1. Representasi Solusi [7][8]

Individu dalam populasi menunjukkan calon pada solusi OPF. Elemen solusi terdiri dari daya pembangkit yang ditetapkan pada semua bus generator (PV) selain dari bus slack, magnitude tegangan yang ditetapkan pada semua bus generator (PV).

3.4.2. Inisialisasi [7][8]

Setiap variabel dari setiap individu dilakukan inisialisasi secara acak menggunakan distribusi nomor acak. Sebagai contoh, pembangkit daya aktif untuk bus PV i, dengan batasan daya aktif dari P_{min} dan P_{max}, maka diperoleh :

$$P_i = U\left[P_{\min}, P_{\max}\right] \quad \text{.....(3.11.)}$$

$U\left[P_{\min}, P_{\max}\right]$adalah nomor acak antara P_{min}, P_{max}. Sebagai tambahan, satu calon solusi akan memiliki pembangkitan daya aktif yang ditetapkan untuk semua bus PV tidak termasuk bus slack.

3.4.3. *Fitness* Calon Solusi [7][8]

Setiap calon solusi menugaskan fungsi *Fitness* untuk mengukur ke optimalannya yang berkenaan dengan sasaran yang akan dioptimalkan. Dalam permasalahan pengiriman daya aktif dan reaktif, fungsi *fitness* dari setiap individu i adalah:

$$f_i = \frac{M}{C_i + \sum_j VP_j + SQ} \quad \text{.....(3.12.)}$$

$$VP_j = \begin{cases} K_v (V_j - 1.0)^2 \ jika \ V_j > V_j^{max} \ atau \ V_j < V_j^{min} \\ 0 \end{cases} \quad \text{.....(3.13)}$$

$$SQ = \begin{cases} K_q (Q_{slack} - Q_{slack}^{max})^2 \ jika \ Q_{slack} > Q_{slack}^{max} \\ K_q (Q_{slack} - Q_{slack}^{min})^2 \ jika \ Q_{slack} < Q_{slack}^{min} \\ 0 \end{cases} \quad \text{.....(3.14)}$$

Dalam rumus di atas, M adalah kemungkinan biaya maksimum pembangkit sedang C_i adalah biaya pembangkitan dari setiap individu i. VP_j adalah istilah hukuman/penalti pada bus PQ atau bus PV j untuk pelanggaran batasan tegangan yang diset sebelumnya V_j^{min}, V_j^{max}. SQ merepresentasikan hukuman/penalti pada bus slack untuk pelanggaran batas daya reaktif. K_v dan K_q adalah konstanta penalti atau hukuman.

3.4.4. Menghasilkan Solusi Baru Dengan Mutasi [7][8]

Populasi yang baru dari solusi OPF dihasilkan dari populasi yang ada melalui operator mutasi. Individu yang baru p_i' dihasilkan dari setiap individu p_i, dimana variabel j OPF pada individu yang baru p_i' dihitung seperti:

$$x_{ji}' = x_{ji} + N(0,\sigma_{ji}^2) \quad \text{.....(3.15.)}$$

Dimana x_{ji}' menandakan nilai dari variabel j dalam p_i'. x_{ji} adalah dari variabel j dalam induk p_i' dan $N(0,\sigma_{ji}^2)$ adalah nomor acak Gaussian dengan nilai tengah nol dan standart deviasi (simpangan baku) dari σ_{ji}. Cara untuk merancang σ_{ji} adalah:

$$\sigma_{ji}^2 = \left(x_j^{max} - x_j^{min}\right)\left(\left(f_{max} - f_i\right)/ f_{max} + a^r\right) \quad \text{.....(3.16.)}$$

Dimana f_i adalah fitness dari individu i; f_{max} adalah fitness maksimum di dalam populasi; x_j^{max}, x_j^{min} menandakan batasan atas dan bawah dari variabel j, a adalah konstanta angka positif; dan r adalah iterasi yang berulang – ulang. Istilah a^r menandakan suatu tingkat mutasi yang mana tergantung pada nilai a.

3.4.5. Batasan Mutasi [7][8]

Untuk memenuhi batasan daya aktif bus slack, maka semua unit selain dari slack ditugaskan untuk memuat suatu perumusan menurut persamaan (3.15.). Total dari pengiriman kemudian dibandingkan dengan total pembangkitan yang didapat dari aliran daya dari individu sebelumnya. Apabila perbedaan diantara mereka berada di dalam batasan operasi dari unit slack, maka calon akan diterima.

Jika tidak, maka proses diulang untuk lima upaya tersebut. Apabila di dalamnya cara yang mungkin tidak didapat, maka mutasi dibatasi untuk memaksakan pemenuhan dengan membagi kelebihan pembangkitan dari bus slack diantara pembangkit yang tersisa.

Dengan mengasumsikan daya aktif bus slack sebagai individu yang terlewati batasan atasnya dan unit slack sebagai unit 1, maka total kapasitas yang tersedia dari unit 2 sampai N dari individu tersebut di peroleh dengan :

$$C_2 = \sum_{i=2}^{N} \left(P_i^{\max} - P_i \right) \quad \text{.....(3.17.)}$$

Dan kelebihan pembangkitan dari bus slack adalah :

$$E_2 = D_L - \left(\sum_{i=2}^{N} P_i + P_1^{\max} \right) \quad \text{.....(3.18.)}$$

Dimana D_L adalah penjumlahan dari kebutuhan daya aktif dan nilai kerugian transmisi yang mana ditemukan di dalam solusi aliran daya sebelumnya dari individu tersebut. Pembebanan dari unit 2 kemudian dimodifkasi berdasarkan :

$$P_i' = P_2 + E_2 \left(P_2 - P_2^{\max} \right) / C_2 \quad \text{.....(3.19.)}$$

Jika pembebanan yang dimodifikasi melebihi pembebanan maksimum unit 2, maka penetapan dilakukan pada nilai pembatas. Jumlah kelebihan pembangkitan dari bus slack yang tersisa untuk dilakukan pembagian yaitu :

$$E_3 = E_2 - \left(P_2' - P_2 \right) \quad \text{.....(3.20.)}$$

Prosedur di atas dilakukan berulang – ulang untuk memodifikasi pembebanan unit 3 sampai N. Setelah semuanya pembebanan semua unit dimodifikasi, daya aktif bus slack berada di batas atas. Proses sama dipakai ketika batas bawah daya aktif bertemu.

3.4.6. Seleksi Individu Dengan Kompetisi [7][8]

Dalam tahap kompetisi, mekanisme seleksi dipakai untuk menghasilkan populasi baru dari populasi yang ada. Agar optimal, solusi yang lebih pas atau lebih optimal seharusnya memiliki peluang seleksi yang lebih besar. Teknik seleksi dipakai adalah skema *tournament* yang dapat dideskripsikan sebagai berikut :

k sebagai solusi induk (*parent*) p_i, seiring dengan anak (*offspring*) p_i' terbentuk oleh mutasi, i = 1, ... ,k, masing – masing mengalami serangkaian turnamen n_j dengan lawan yang terseleksi secara acak. Setiap individu i diberi suatu skor S_i berdasarkan :

$$S_i = \sum_{j=1}^{N} n_j \quad \text{.....(3.21.)}$$

$$n_j = \begin{cases} 1 \, jika \, f_i > f_r \\ 0 \end{cases} \quad \text{.....(3.22.)}$$

Sedang f_i adalah *fitness* setiap individu i. Lawan r, terpilih acak dari setiap individu 2k berdasarkan $r = [2ku + 1]$. u adalah angka acak yang sama dalam interval $[0,1]$, k adalah solusi calon skor tertinggi diambil dari individu generasi selanjutnya.

3.4.7. *Parallel Evolutionary Programming Algorithm* [8][10]

Ukuran populasi adalah salah satu faktor yang akan mempengaruhi performansi EP-OPF algorithm untuk mencari solusi yang optimal. Jika ukuran populasi yang digunakan besar, kemungkinan untuk menghasilkan solusi yang optimal dengan mutasi dan kompetisi tinggi. Hal ini jelas akan memerlukan perhitungan waktu yang relatif lama. Untuk memperbaiki kecepatan perhitungan ketika mempertahankan kualitas solusi yang sama, maka perlu diimplementasikan *Parallel EP-OPF Algorithm.*

Dasar pemikiran dari *Parallel EP-OPF algorithm* ini adalah untuk membagi *initial population* ke beberapa sub-populasi. Selama ukuran sub populasi ini lebih kecil dari *initial populations*, maka waktu perhitungannya akan di kurangi. Komputer yang digunakan ditata pada struktur *master slave* dan di perintahkan melewati protokol *interface (MPI)* dengan menggunakan bahasa Delphi seri 7.0 untuk di aplikasikan pada program *Parallel EP-OPF.*

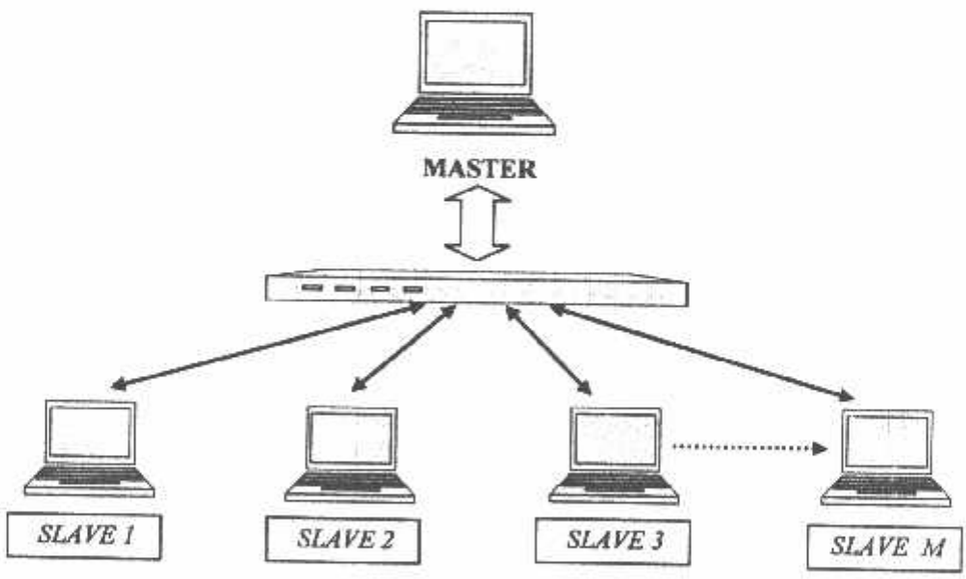
Gambar di bawah ini menggambarkan konfigurasi dari Komputer untuk topologi *master slave Parallel EP-OPF.* Di topologi ini, *single master prosessor* di perintahkan untuk mengkoordinasikan *m slave prosessor. Master prosessor* melakukan perlombaan kompetisi untuk menyeleksi individu yang paling layak dan pantas diantara sub-populasi dari *slave prosessor* yang jelas-jelas membutuhkan mutasi dan *fitness evaluation.*

Prosedur dari Parallel EP-OPF dapat di jelaskan sebagai berikut:

1. *Master processor* secara random / acak menginisialisasikan semua individu populasi.

2. *Master processor* membagi seluruh populasi diantara *m slave* dan kemudian mengirim sub populasi ke setiap *slave processor*.
3. Setelah menerima sub-populasi, *slave processor* bebas menjalankan *mutasi* dan *fitness* calon solusi. Kemudian setiap *slave processor* mengirimkan hasilnya kembali ke *master*.
4. Setelah menggabungkan sub-populasi dari tiap-tiap slave, seleksi individu dengan kompetisi diperintahkan oleh master sampai terpilih individu yang memiliki nilai tertinggi dari induk *(master)* dan populasinya di mutasikan ke *(slave)* untuk membentuk generasi selanjutnya.
5. Aturan ini akan dicek jika tidak memuaskan, dan kembali ke tahap 2.

Gambar 3.2. [8][10]
Konfigurasi dari *master-slave Parallel EP-OPF algorithm*.



BAB IV

HASIL DAN ANALISA HASIL

4.1. Program Komputer Optimal Power Flow Menggunakan Metode *Parallel Evolutionary Programming*

Dalam penyelesaian masalah ini diperlukan bantuan program komputer dalam perhitungan yang membutuhkan ketelitian dan keakuratan. Program komputer dalam skripsi ini di jalankan dengan menggunakan bahasa Pemrograman Bordland Delphi versi 7.0 dan di aplikasikan dengan menggunakan 5 komputer berprosesor AMD Athlon 850 Mhz, dengan memori 256 Mb yang terangkai secara pararel.

4.1.1. Algoritma Program

Algoritma program optimasi dilakukan dengan langkah-langkah sebagai berikut:

1. Memasukkan Inputan data beban yang meliputi tegangan (V), sudut fasa tegangan (δ), daya aktif (P), daya reaktif (Q) dan data impedansi saluran, data pembangkit Pmak, Pmin, konstanta biaya.
2. Lakukan proses *load flow Newton Raphson.*
3. Lakukan proses *Parallel Evolutionary Programming.*
4. Apakah objective function sudah terpenuhi?
5. Jika "tidak" maka melakukan perhitungan *load flow* dan *PEP* lagi.
6. Jika "ya" maka tampilkan hasil optimasi dan perhitungan berhenti.

Sedangkan untuk Algoritma aliran daya dengan metode *Newton Raphson* adalah sebagai berikut:

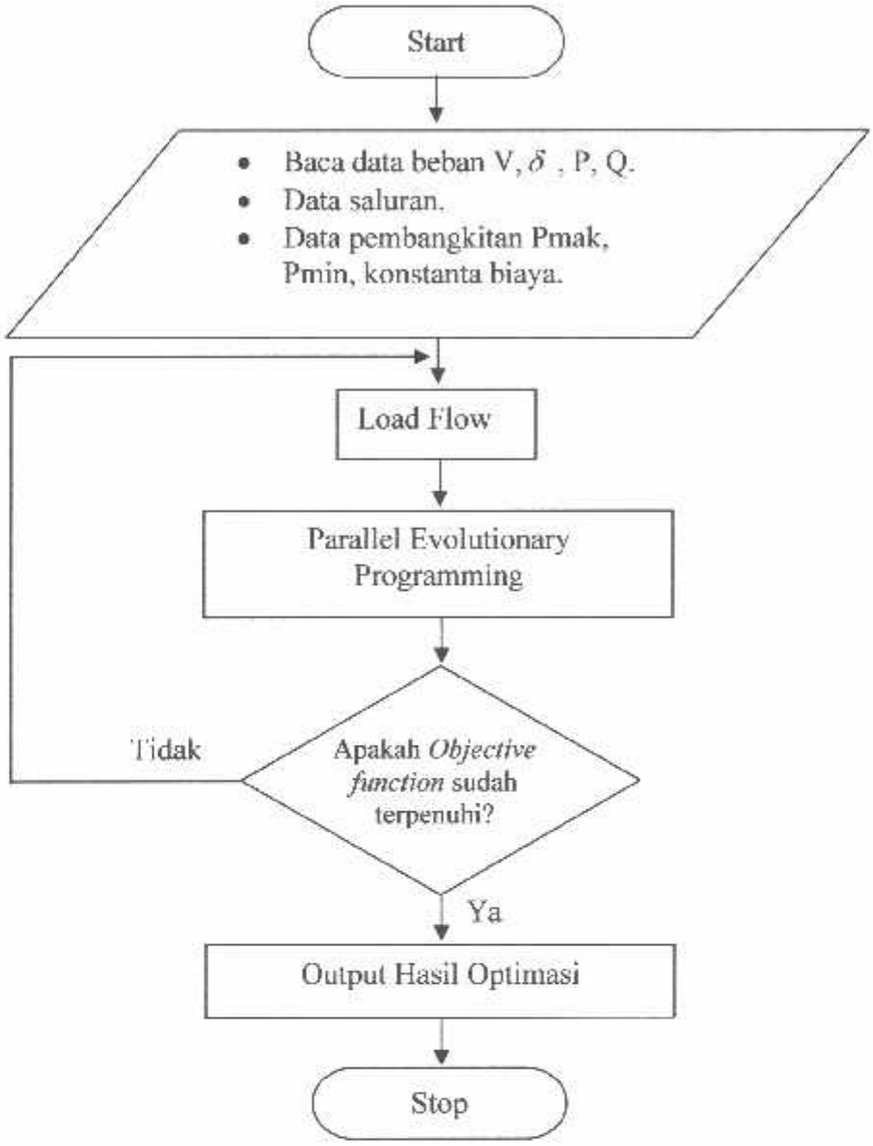
1. Bentuk matriks admitansi bus : [Y_{bus}]
2. Penetapan harga awal tegangan dan sudut fasa untuk semua bus kecuali bus slack [Vi (0), δ (0)].
3. Menentukan nomor iterasi awal, k = 0.
4. Menghitung injeksi daya aktif dan reaktif pada setiap bus dengan persamaan kecuali bus slack.
5. Menghitung selisih daya yang di jadwalkan dengan injeksi daya bus dari perhitungan.
6. Menentukan perubahan maksimum pada daya aktif dan reaktif.
7. Membandingkan apakah selisih daya sudah sama atau lebih kecil dari ε
8. Jika " ya" hitung daya aktif dan reaktif , tegangan dan sudut fasa tegangan pada setiap bus, serta aliran daya pada saluran dan perhitungan selesai, jika "tidak" lanjutkan ke langkah berikutnya.
9. Membentuk element matriks Jacobian.
10. Menghitung faktor koreksi tegangan dan sudut fasa setiap bus kecuali bus slack dan bus generator.
11. Menghitung nilai sudut fasa yang lama dengan sudut fasa yang baru, tegangan yang lama dengan tegangan yang baru.
12. Mengganti nilai sudut fasa yang baru, tegangan yang lama dengan tegangan yang baru.

13. Perhitungan di lanjutkan ke langkah 4 dengan nilai iterasi yang baru sampai hasil yang di dapatkan konvergen.

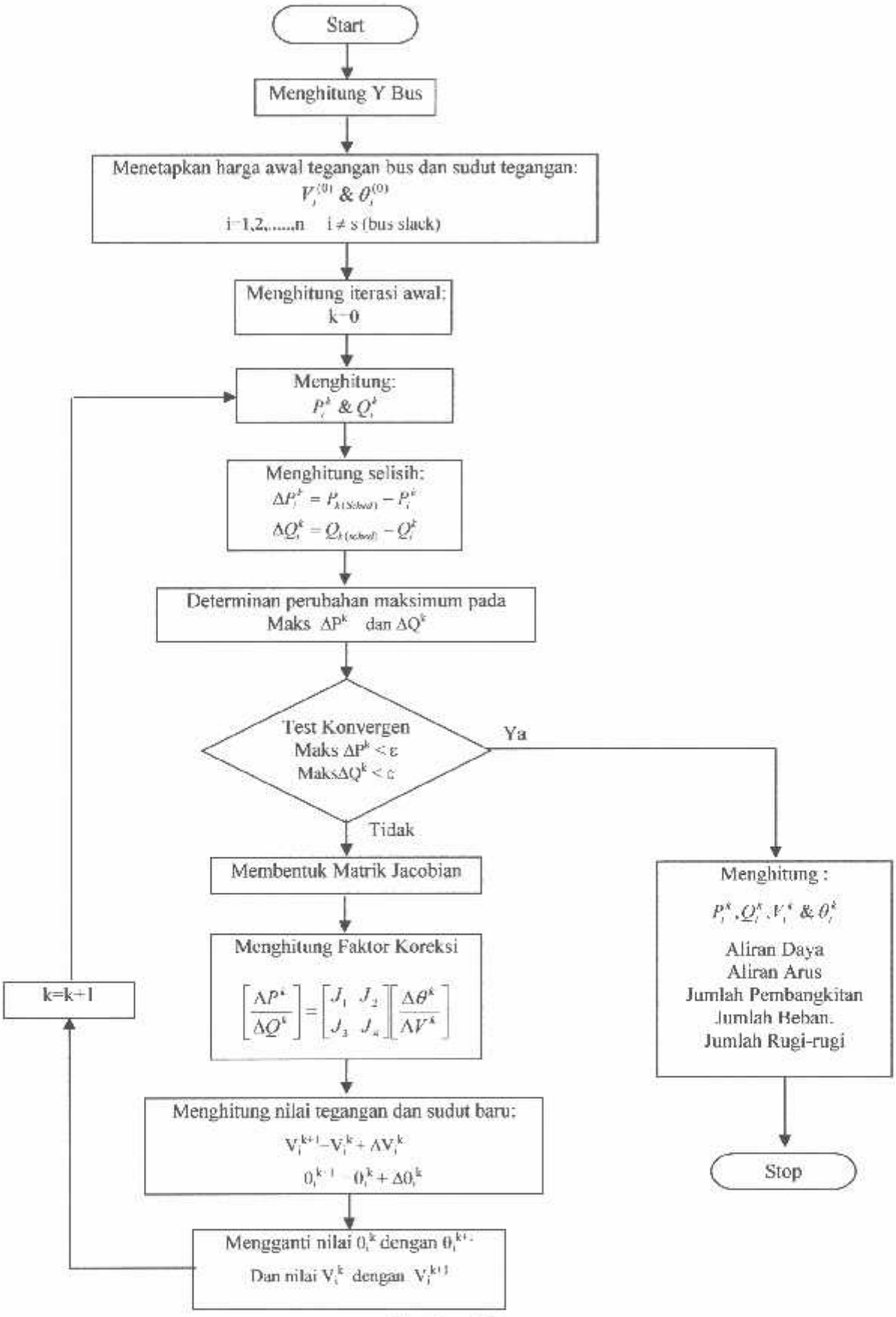
Selanjutnya untuk Algoritma program OPF menggunakan metode PEP dilakukan dengan langkah-langkah sebagai berikut:

1. Memasukkan inputan data beban yang meliputi tegangan (V), sudut fasa tegangan (δ), daya aktif (P), daya reaktif (Q) dan data impedansi saluran, data pembangkitan Pmak, Pmin, konstanta biaya.
2. Menentukan parameter inputan EP yang meliputi jumlah populasi, maksimum generasi, nilai kemungkinan mutasi, dan panjang kromosom tiap-tiap individu.
3. Generasi = 0, Populasi = 0.
4. Melakukan inisialisasi calon solusi.
5. Menghitung *fitness* dari kromosom tiap-tiap individu.
6. Melakukan proses statistik.
7. Melakukan proses mutasi di *slave 1, slave2, slave3 dan slave4.*
8. Melakukan proses kompetisi di *slave 1, slave2, slave3 dan slave4.*
9. Proses no. 7,8 di ulang sampai *offspring slave 1, slave2, slave3 dan slave4* sama dengan maksimum populasi.
10. Menghitung fitness dari *offspring.*
11. Apakah generasi yang di inginkan sudah terpenuhi (max Gen).
12. Jika "tidak" maka genersai = gen + 1, kembali ke langkah 7.
13. Jika "Ya" maka perhitungan berhenti.

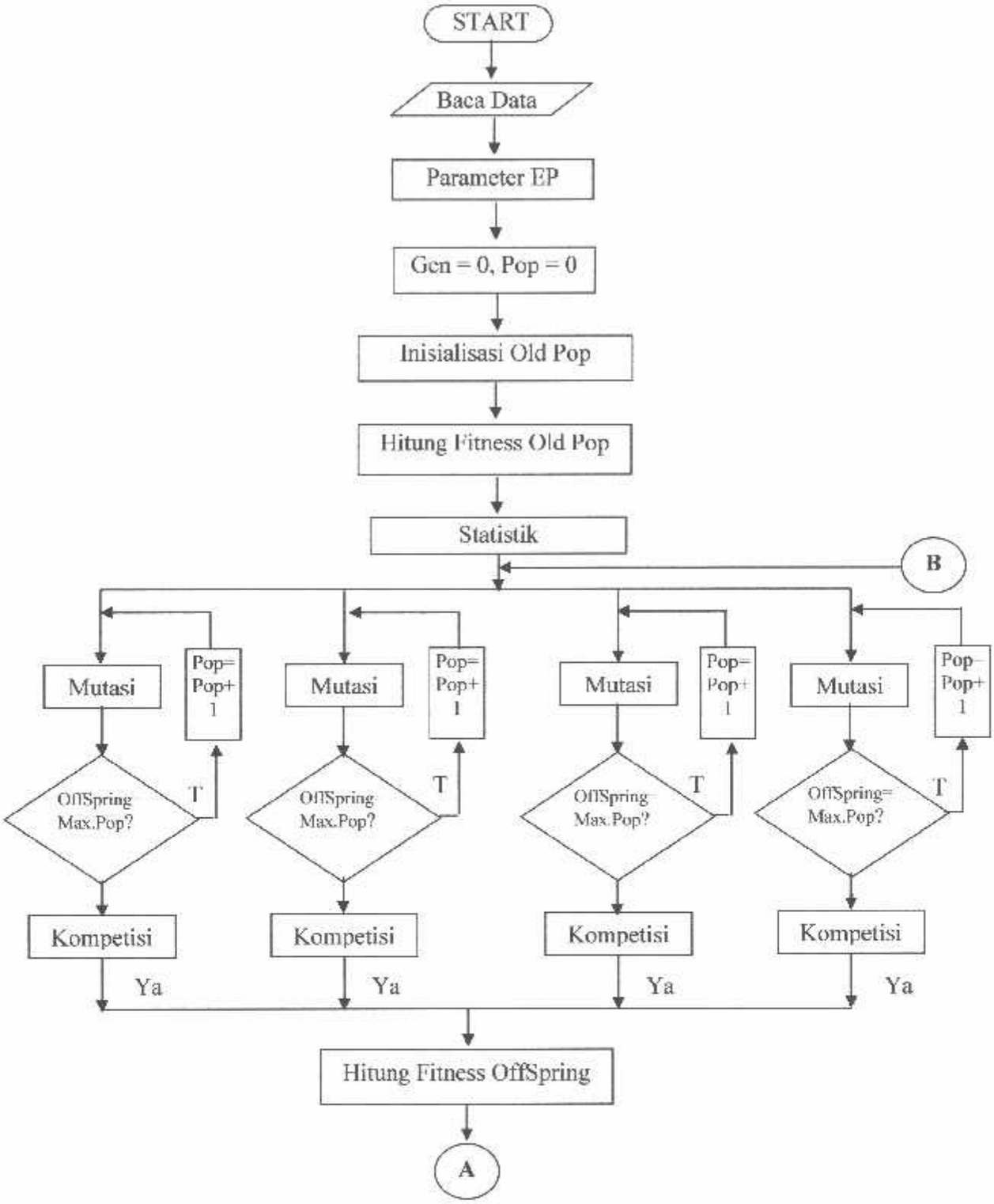
4.1.2. Flowchart Program.

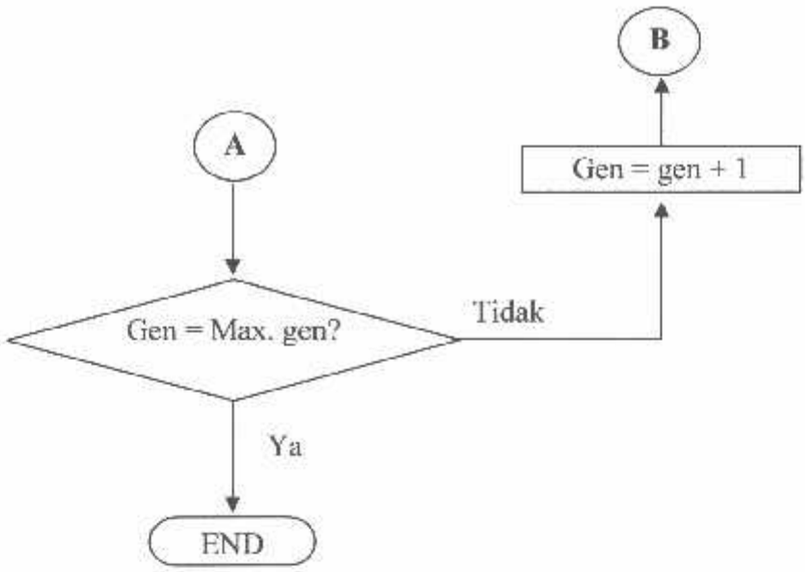


Gambar 4.1
Flowchart Perhitungan Program Optimasi



Gambar 4.2.
Flowchart Perhitungan Aliran Daya Metode *Newton Rapshon*


START
Baca Data
Parameter EP
Gen = 0, Pop = 0
Inisialisasi Old Pop
Hitung Fitness Old Pop
Statistik
B
Mutasi
Pop= Pop+ 1
OffSpring Max.Pop?
T
Kompetisi
Ya
Hitung Fitness OffSpring
A



Gambar 4.3.
Flowchart Program *Optimal Power Flow* Menggunakan *Parallel Evolutionary Programming*

4.2. Validasi Data IEEE 30 Untuk Menyelesaikan Permasalahn *Optimal Power Flow* Menggunakan Metode *Parallel Evolutionary Programming*.

Tabel 4.1 [7]
Data Generator Dan Koefisien Biaya IEEE 30.

No Bus	Qmin (MVAR)	Qmax (MVAR)	a2	a1	a0	Pmin (MW)	Pmax (MW)
1	-20	150	0.00375	2.00	0.00	50	200
2	-20	60	0.01750	1.75	0.00	20	80
5	-15	62.5	0.06250	1.00	0.00	15	50
8	-15	50	0.00834	3.25	0.00	10	35
11	-10	40	0.02500	3.00	0.00	10	30
13	-15	45	0.02500	3.00	0.00	12	40

Tabel 4.2[7] [10]
Data Bus IEEE 30

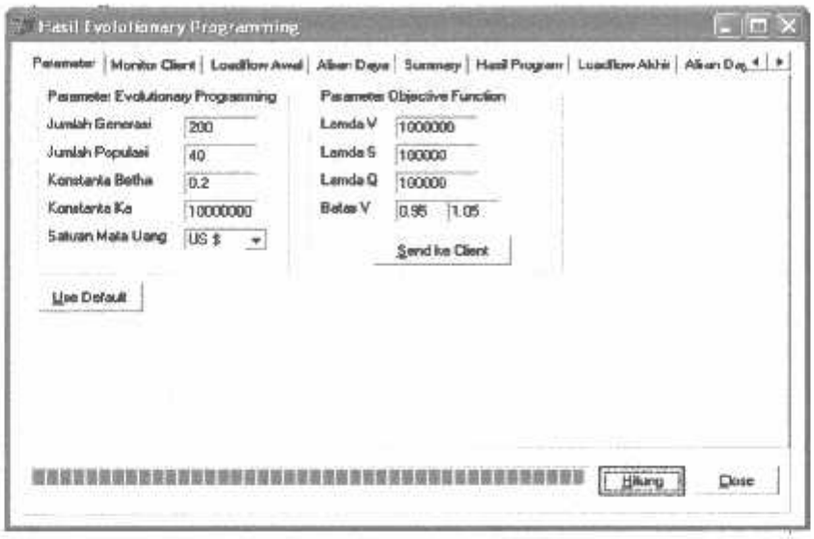
Bus No.	Tegangan		Pembangkit		Pembebanan	
	Magnitude (pu)	Sudut (deg)	Aktif (MW)	Reaktif (MVAR)	Aktif (MW)	Reaktif (MVAR)
1	1.05	0.00	0.00	0.00	0.00	0.00
2	1.0338	0.00	57.56	0.00	21.70	12.70
3	1.00	0.00	0.00	0.00	2.400	1.20
4	1.00	0.00	0.00	0.00	7.600	1.60
5	1.0058	0.00	24.56	0.00	94.20	19.00
6	1.00	0.00	0.00	0.00	0.00	0.00
7	1.00	0.00	0.00	0.00	22.80	10.90
8	1.0230	0.00	35.00	0.00	30.00	30.00
9	1.00	0.00	0.00	0.00	0.00	0.00
10	1.00	0.00	0.00	0.00	5.80	2.00
11	1.0913	0.00	17.93	0.00	0.00	0.00
12	1.00	0.00	0.00	0.00	11.20	7.50
13	1.0883	0.00	16.91	0.00	0.00	0.00
14	1.00	0.00	0.00	0.00	6.20	1.60
15	1.00	0.00	0.00	0.00	8.20	2.50
16	1.00	0.00	0.00	0.00	3.50	1.80
17	1.00	0.00	0.00	0.00	9.00	5.80
18	1.00	0.00	0.00	0.00	3.20	0.90
19	1.00	0.00	0.00	0.00	9.50	3.40
20	1.00	0.00	0.00	0.00	2.20	0.70
21	1.00	0.00	0.00	0.00	17.50	11.20
22	1.00	0.00	0.00	0.00	0.00	0.00
23	1.00	0.00	0.00	0.00	3.200	1.60
24	1.00	0.00	0.00	0.00	8.700	6.70
25	1.00	0.00	0.00	0.00	0.00	0.00
26	1.00	0.00	0.00	0.00	2.30	2.30
27	1.00	0.00	0.00	0.00	0.000	0.00
28	1.00	0.00	0.00	0.00	0.000	0.00
29	1.00	0.00	0.00	0.00	0.400	0.90
30	1.00	0.00	0.00	0.00	10.60	1.90

Tabel 4.3[14]

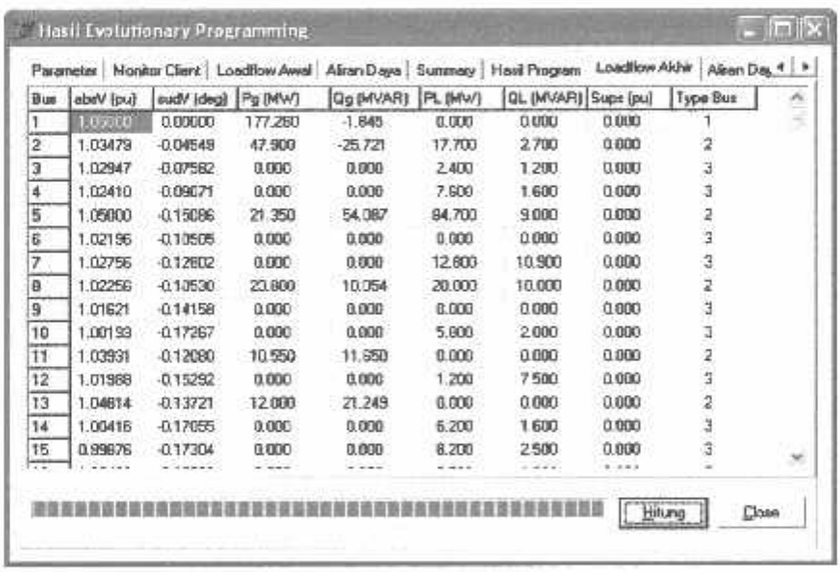
Data saluran Impedansi IEEE 30

No Saluran	Hubungan Bus	R (pu)	X (pu)	B (pu)
1	1-2	0.0192	0.0575	0.0264
2	1-3	0.0452	0.1852	0.0204
3	2-4	0.0570	0.1737	0.0184
4	3-4	0.0132	0.0379	0.0042
5	2-5	0.0472	0.1983	0.0209
6	2-6	0.0581	0.1763	0.0187
7	4-6	0.0119	0.0414	0.0045
8	5-7	0.0460	0.0102	0.0102
9	6-7	0.0267	0.0820	0.0085
10	6-8	0.0120	0.0420	0.0045
11	6-9	0.0000	0.2080	0.0000
12	6-10	0.0000	0.5560	0.0000
13	9-11	0.0000	0.2080	0.0000
14	9-10	0.0000	0.1100	0.0000
15	4-12	0.0000	0.2560	0.0000
16	12-13	0.0000	0.1400	0.0000
17	12-14	0.01231	0.2559	0.0000
18	12-15	0.0602	0.1304	0.0000
19	12-16	0.0945	0.1987	0.0000
20	14-15	0.2210	0.1997	0.0000
21	16-17	0.0824	0.1932	0.0000
22	15-18	0.1070	0.2185	0.0000
23	18-19	0.0639	0.1292	0.0000
24	19-20	0.0340	0.0680	0.0000
25	10-20	0.0936	0.2090	0.0000
26	10-17	0.0324	0.0845	0.0000
27	10-21	0.0348	0.0749	0.0000
28	10-22	0.0727	0.1499	0.0000
29	21-22	0.0116	0.0236	0.0000
30	15-23	0.1000	0.2020	0.0000
31	22-24	0.1150	0.1790	0.0000
32	23-24	0.1320	0.2700	0.0000
33	24-25	0.1885	0.3292	0.0000
34	25-26	0.2544	0.3800	0.0000
35	25-27	0.1093	0.2087	0.0000
36	28-27	0.0000	0.3960	0.0000
37	27-29	0.2198	0.4153	0.0000
38	27-30	0.3202	0.6027	0.0000
39	29-30	0.2399	0.4533	0.0000
40	8-28	0.0636	0.2000	0.0214
41	6-28	0.0169	0.0599	0.0065

4.2.1 Hasil Validasi IEEE 30 Dengan Menggunakan Metode *Parallel Evolutionary Programming*

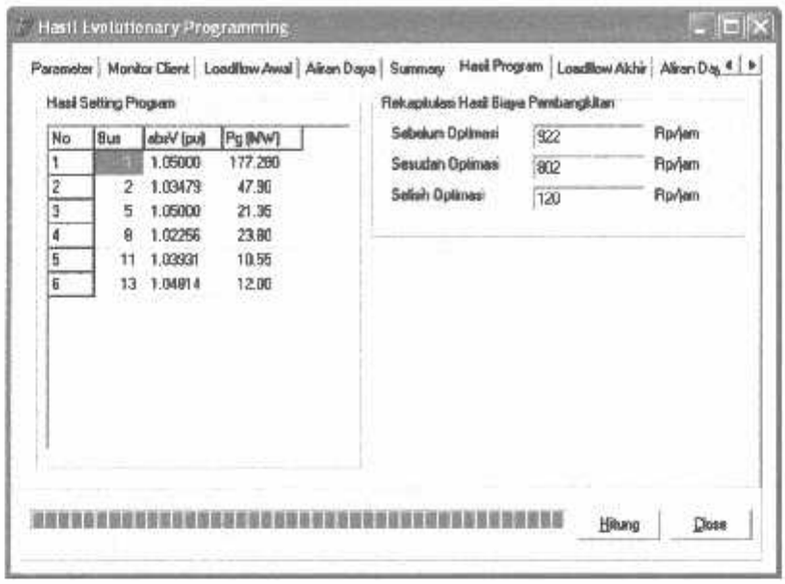


Gambar 4.4
Tampilan Parameter Data Validasi IEEE 30



Bus	absV (pu)	sudV (deg)	Pg (MW)	Qg (MVAR)	PL (MW)	QL (MVAR)	Susc (pu)	Type Bus
1	1.06000	0.00000	177.260	-1.845	0.000	0.000	0.000	1
2	1.03479	-0.04549	47.900	-25.721	17.700	2.700	0.000	2
3	1.02947	-0.07582	0.000	0.000	2.400	1.200	0.000	3
4	1.02410	-0.09671	0.000	0.000	7.600	1.600	0.000	3
5	1.05000	-0.15086	21.350	54.087	94.700	9.000	0.000	2
6	1.02196	-0.10505	0.000	0.000	0.000	0.000	0.000	3
7	1.02756	-0.12602	0.000	0.000	12.600	10.900	0.000	3
8	1.02256	-0.10530	20.800	10.054	28.000	10.000	0.000	2
9	1.01621	-0.14158	0.000	0.000	0.000	0.000	0.000	3
10	1.00193	-0.17267	0.000	0.000	5.800	2.000	0.000	3
11	1.03931	-0.12080	10.550	11.950	0.000	0.000	0.000	2
12	1.01968	-0.15292	0.000	0.000	1.200	7.500	0.000	3
13	1.04814	-0.13721	12.000	21.249	0.000	0.000	0.000	2
14	1.00416	-0.17055	0.000	0.000	6.200	1.600	0.000	3
15	0.99876	-0.17304	0.000	0.000	8.200	2.500	0.000	3

(a) Hasil *Load Flow*



(b) Hasil Perhitungan Biaya Pembangkitan

Gambar 4.5
Tampilan Hasil Validasi IEEE 30

Tabel 4.4
Perbandingan Hasil Data Referensi Jurnal dengan Data Optimasi

Unit No	Bus No	**Data Referensi**			**Data Optimasi**		
		Tegangan (pu)	P Gen MW	Biaya $/h	Tegangan (pu)	P Gen MW	Biaya $/h
1	1	1.0500	177.260	472.379	1.05000	177.280	472.415
2	2	1.0360	47.92	124.045	1.03479	47.90	123.977
3	5	1.0500	21.36	49.875	1.05000	21.36	49.875
4	8	1.0100	23.81	82.110	1.02256	23.80	82.074
5	11	1.0990	10.56	34.467	1.03931	10.55	34.436
6	13	1.0600	12.00	39.600	1.04814	12.00	39.600
Total Biaya			**292.91**	**802.476**	**Total Biaya**	**292.90**	**802.377**

Setelah Dilakukan Proses Optimasi maka di peroleh biaya pembangkitan dari data IEEE 30 yaitu sebesar 802.476 $/h sedangkan biaya total hasil program sebesar 802.377 $/h.

Sehingga dari proses validasi diatas didapatkan error / kesalahan perhitungan untuk keluaran daya aktif sebesar 0,00034 %, sedangkan untuk biaya pembangkitan sebesar 0,000123 %.

4.3 Data Pembangkitan Thermal Pada Sub Sistem Paiton dan Bali

Pada skripsi ini akan membahas pembangkit thermal yang berada pada sub sistem Paiton dan Bali. Pembangkit thermal yang dibahas dalam skripsi ini adalah PLTU Paiton 1-2, PLTG Gilimanuk, PLTD Pesanggaran, dan PLTG Pesanggaran.

Untuk data dari bentuk karakteristik semua unit pembangkit yang dibahas dalam skripsi ini beserta kapasitasnya dan *fuel cost* (biaya bahan bakar) yang digunakan dalam perhitungan adalah berdasarkan data dari PT. Indonesia Power dan PT. PJB.

Tabel 4.5 [15]
Parameter Unit Pembangkit Thermal

Nama Pembangkit	a2	a1	a0	Pmin (MW)	Pmax (MW)
PLTU Paiton 1&2	6.180000	1306.150	388144.168	150	700
PLTG Gilimanuk	1.406900	1599.000	87435.000	50	133,8
PLTD Pesanggaran	14.293200	1670.000	88960.000	21	75
PLTG Pesanggaran	113.900000	5297.000	139560.000	15	125,5

Dengan memasukkan data-data pada tabel 4.5. ke persamaan fungsi biaya bahan bakar, maka untuk unit pembangkit thermal PLTU Paiton 1&2 diperoleh sebagai berikut :
Biaya bahan bakar pembangkit:

$$F_i(G_i) = a_2 P_{Gi}^{\ 2} + a_1 P_{Gi} + a_0$$

$$F_1(G_1) = 6.180000\,P^2 + 1306.150\,P + 388144.168$$

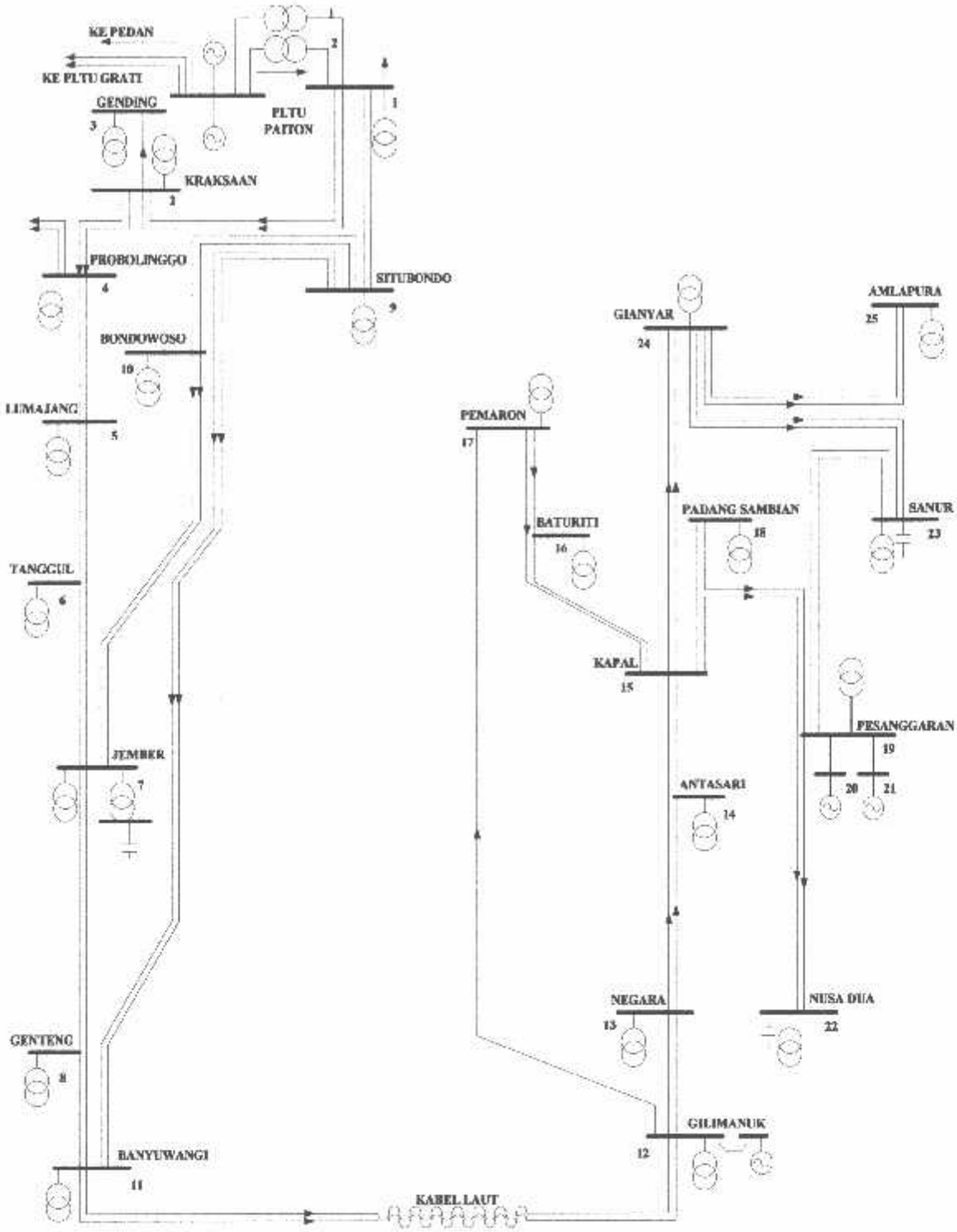
Untuk persamaan biaya pembangkitan dari masing-masing unit pembangkit yang berdasarkan data-data diatas adalah sebagai berikut:

Tabel 4.6
Persamaan Biaya Pembangkitan
Unit Pembangkit Thermal Paiton dan Bali

No.	Nama Pembangkit	Persamaan Biaya Bahan Bakar (Rp/Jam)
1	PLTU Paiton 1&2	$F = 6.180000\ P^2 + 1306.150\ P + 388144.168$
2	PLTG Gilimanuk	$F = 1.406900\ P^2 + 1599.000\ P + 87435.000$
3	PLTDPesanggaran	$F = 14.293200\ P^2 + 1670.000\ P + 88960.000$
4	PLTGPesanggaran	$F = 113.900000\ P^2 + 5297.000\ P + 139560.000$

Catatan:

Harga Batu Bara	253 Rp/Kg
Harga MFO	1595,5 Rp/Liter
Harga HSD	15945,5 Rp/Liter
Harga Gas	2.53 US$/MMBTU
Nilai tukar	9000 Rp/US$



Sumber : PT. PLN (Persero) P3B Jawa-Bali

Gambar 4.6
Diagram Segaris Jaringan Sistem Tenaga Listrik Sub Sistem
150 kV Paiton-Bali

Tabel 4.7
Penomoran Bus 150 kV Sub Sistem Paiton dan Bali

No. Bus	Nama Bus
1	PAITON
2	KRAKSAAN
3	GENDING
4	PROBOLINGGO
5	LUMAJANG
6	TANGGUL
7	JEMBER
8	GENTENG
9	SITUBONDO
10	BONDOWOSO
11	BANYUWANGI
12	GILIMANUK
13	NEGARA
14	ANTASARI
15	KAPAL
16	BATURITI
17	PEMARON
18	PADANG SAMBIAN
19	PESANGGARAN
20	PLTD PESANGGARAN
21	PLTG PESANGGARAN
22	NUSA DUA
23	SANUR
24	GIANYAR
25	AMPLAPURA

4.4 Data Pembangkitan dan Pembebanan 150 kV Sub Sistem Paiton dan Bali

Berdasarkan data pada referensi [17] maka dilakukan pengolahan data untuk mengubah dasar tegangan ke dalam pu. Serta perhitungan P_{load} dan Q_{load}.

Pada bus nomor 1, maka :

Tegangan Dasar = 150 kV

Tegangan Sebenarnya = 155 kV

$$\text{Tegangan (pu)} = \frac{\text{Tegangan sebenarnya}}{\text{Tegangan dasar}}$$

$$\text{Tegangan (pu)} = \frac{155}{150} = 1.033 \text{ pu}$$

P_{beban} = 11.7 MW

Q_{beban} = 5.7 MVAR

Untuk selanjutnya dilakukan perhitungan seperti langkah di atas, sehingga didapatkan hasil sebagai berikut :

Tabel 4.8.[17]
Data Pembangkitan dan Pembebanan
150 kV Sub Sistem Paiton dan Bali
Hari Rabu, Tanggal 30 Maret 2005, Pukul : 19:30 WIB

No	Nama Bus	Tegangan (pu)	Pgenerator (MW)	Qgenerator (MVAR)	Pload (MW)	Qload (MVAR)	Tipe
1	PAITON	1,03333	0.0000	0.0000	11,700	5,700	1
2	KRAKSAAN	1.00000	0.0000	0.0000	16,500	6,500	3
3	GENDING	1.00000	0.0000	0.0000	17,500	6,700	3
4	PROBOLINGGO	1.00000	0.0000	0.0000	39,300	19,800	3
5	LUMAJANG	1.00000	0.0000	0.0000	39,800	17,500	3
6	TANGGUL	1.00000	0.0000	0.0000	23,400	9,300	3
7	JEMBER	1.00000	0.0000	47,000	58,900	30,300	3

8	GENTENG	1.00000	0.0000	0.0000	39,800	21,800	3
9	SITUBONDO	1.00000	0.0000	0.0000	24,400	5,800	3
10	BONDOWOSO	1.00000	0.0000	0.0000	17,800	7,400	3
11	BANYUWANGI	1.00000	0.0000	0.0000	37,800	14,700	3
12	GILIMANUK	0,99260	99,900	0.0000	5,700	2,000	2
13	NEGARA	1.00000	0.0000	0.0000	11,700	4,600	3
14	ANTASARI	1.00000	0.0000	0.0000	6,000	2,800	3
15	KAPAL	1.00000	0.0000	50,000	69,300	23,500	3
16	BATURITI	1.00000	0.0000	0.0000	4,000	0,500	3
17	PEMARON	1.00000	0.0000	0.0000	24,800	8,800	3
18	PDG SAMBIAN	1.00000	0.0000	0.0000	32,800	13,300	3
19	PESANGGARAN	1.00000	0.0000	0.0000	76,400	28,900	3
20	PLTD PSGRAN	0,93600	31,400	0.0000	0,000	0,000	2
21	PLTG PSGRAN	0,93600	84,600	0.0000	0,000	0,000	2
22	NUSADUA	1.00000	0.0000	25,000	51,400	19,300	3
23	SANUR	1.00000	0.0000	25,000	60,300	17,700	3
24	GIANYAR	1.00000	0.0000	0.0000	33,700	9,600	3
25	AMLAPURA	1.00000	0.0000	0.0000	14,400	7,000	3

Ket: 1 : bus slack
2 : bus generator
3 : bus beban

4.5 Data Saluran Transmisi 150 kV Sub Sistem Paiton dan Bali

Pada sub sistem Paiton dan Bali terdiri dari 25 bus dan 34 saluran transmisi, dalam hal ini saluran transmisi yang dibahas adalah saluran transmisi 150 kV.

Berdasarkan data pada referensi [18], maka dilakukan pengolahan data untuk saluran dengan jumlah sirkuit 2. Sebagai contoh pada hubungan saluran 1-2 dilakukan perhitungan sebagai berikut :

$$R = \frac{R_1 \times R_2}{R_1 + R_2} = \frac{0.0105 \times 0.0105}{0.0105 + 0.0105} = \frac{1.1025 \times 10^{-4}}{0.021} = 0.0053 \text{ pu}$$

$$X = \frac{X_1 \times X_2}{X_1 + X_2} = \frac{0.0359 \times 0.0359}{0.0359 + 0.0359} = \frac{1.2888 \times 10^{-3}}{0.0718} = 0.0179 \text{ pu}$$

$$B = \frac{B_1 \times B_2}{B_1 + B_2} = \frac{0.0130 \times 0.0130}{0.0130 + 0.0130} = \frac{1.69 \times 10^{-4}}{0.026} = 0.0065 \text{ pu}$$

Untuk selanjutnya dilakukan perhitungan seperti langkah di atas, sehingga didapatkan hasil sebagai berikut :

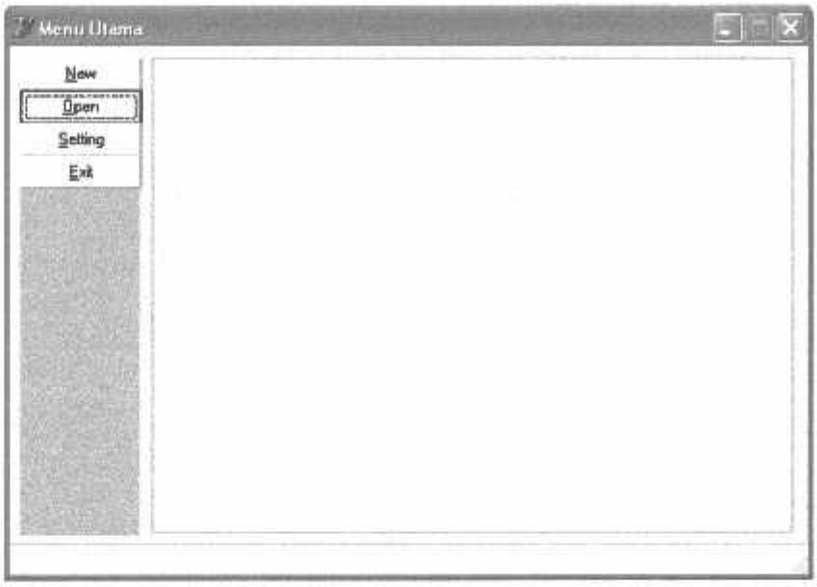
Tabel 4.9 [18]
Data saluran transmisi sub sistem 150 kV Paiton- Bali.

No Saluran	Hubungan Bus	Jumlah sirkuit	R (pu)	X (pu)	B (pu)
1	1-2	2	0.0053	0.0179	0.0065
2	1-9	2	0.0048	0.0346	0.0251
3	2-3	1	0.0100	0.0340	0.0123
4	2-4	2	0.0079	0.0269	0.0097
5	4-5	2	0.0135	0.0460	0.0166
6	5-6	1	0.0167	0.0571	0.0206
7	5-7	1	0.0315	0.1077	0.0389
8	6-7	1	0.0148	0.0506	0.0183
9	7-8	1	0.0337	0.1151	0.0415
10	7-10	2	0.0104	0.0353	0.0128
11	7-11	1	0.0430	0.1469	0.0530
12	8-11	1	0.0177	0.0604	0.0218
13	9-10	2	0.0091	0.0310	0.0112
14	9-11	2	0.0966	0.0457	0.0340
15	11-12	2	0.0016	0.0035	0.0000
16	12-13	1	0.0116	0.0336	0.0124
17	12-17	2	0.0399	0.1314	0.0502
18	13-14	1	0.0270	0.0783	0.0288
19	13-15	1	0.0412	0.1194	0.0439
20	14-15	1	0.0142	0.0411	0.0151
21	15-16	1	0.0362	0.0693	0.0240
22	15-17	1	0.0556	0.1054	0.0369
23	15-18	1	0.0105	0.0304	0.0112
24	15-19	1	0.0105	0.0304	0.0112
25	15-22	1	0.0293	0.0561	0.0195
26	15-24	2	0.0059	0.0170	0.0063
27	16-17	1	0.0194	0.0371	0.0129
28	18-19	1	0.0045	0.0130	0.0048
29	19-20	1	0.0020	0.0044	0.0000
30	19-21	1	0.0020	0.0056	0.0000
31	19-22	1	0.0127	0.0243	0.0084
32	19-23	2	0.0037	0.0070	0.0025
33	23-24	2	0.0078	0.0149	0.0052
34	24-25	2	0.0103	0.0298	0.0110

4.6 Prosedur pelaksanaan program perhitungan.

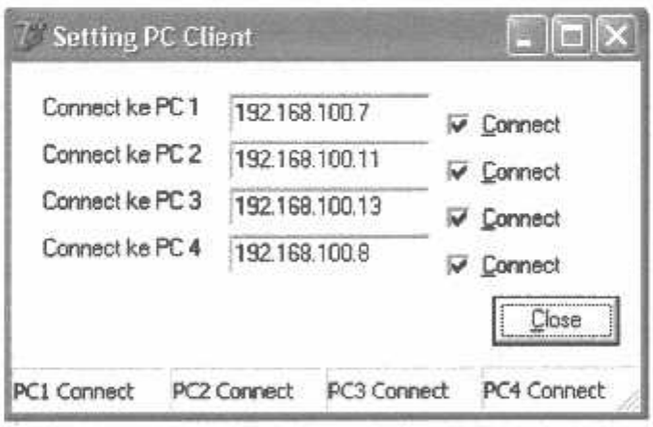
Prosedur menjalankan program perhitungan dilakukan setelah memasukkan seluruh data ke dalam program dengan menggunakan bahasa Pemrograman Borland Delphi versi 7.0. Mengenai prosedur jalannya program dapat dilakukan sebagai berikut:

1. Tampilan utama program.



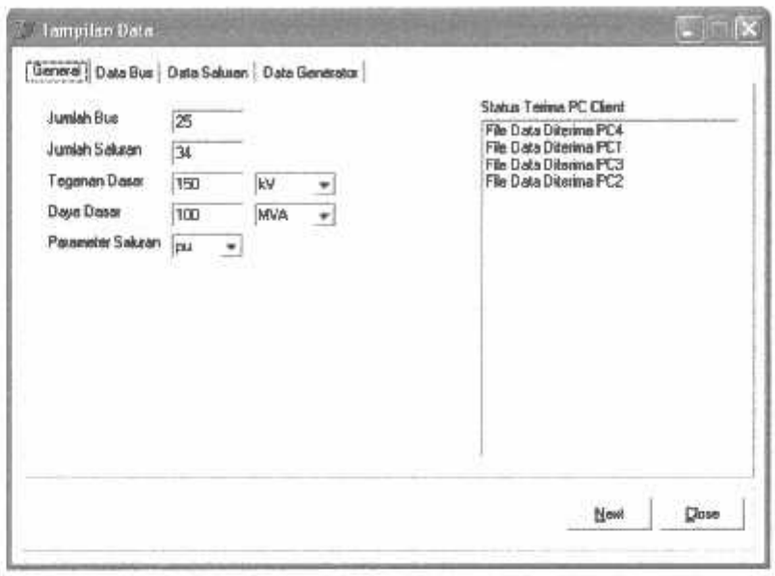
Gambar 4.7
Tampilan Utama Program.

2. Tekan tombol setting untuk memasukkan *IP address* untuk memulai program *Perarel EP* ke *PC Client*, jika sudah tekan *Connect* dan *Close*



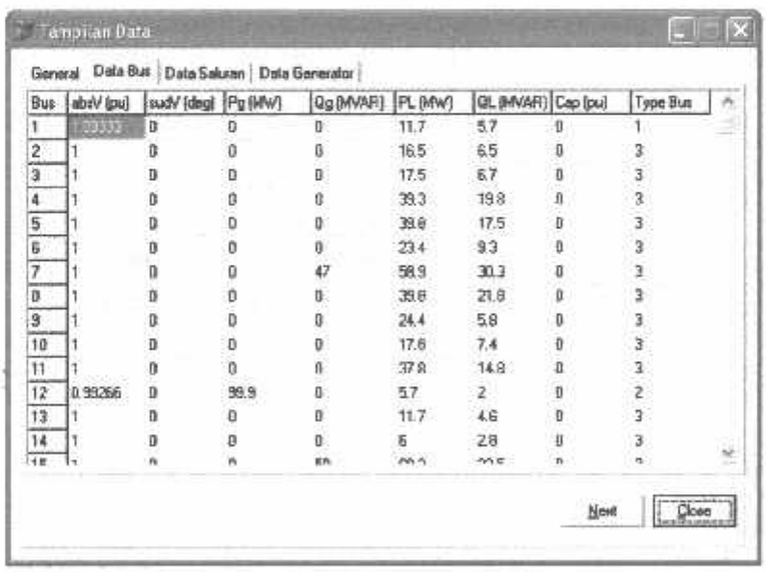
Gambar 4.8
Tampilan Setting PC Client

3. Selanjutnya, tekan open file untuk membuka data yang sudah tersimpan sekaligus mengecek status File di *PC Client* apakah sudah diterima atau belum.



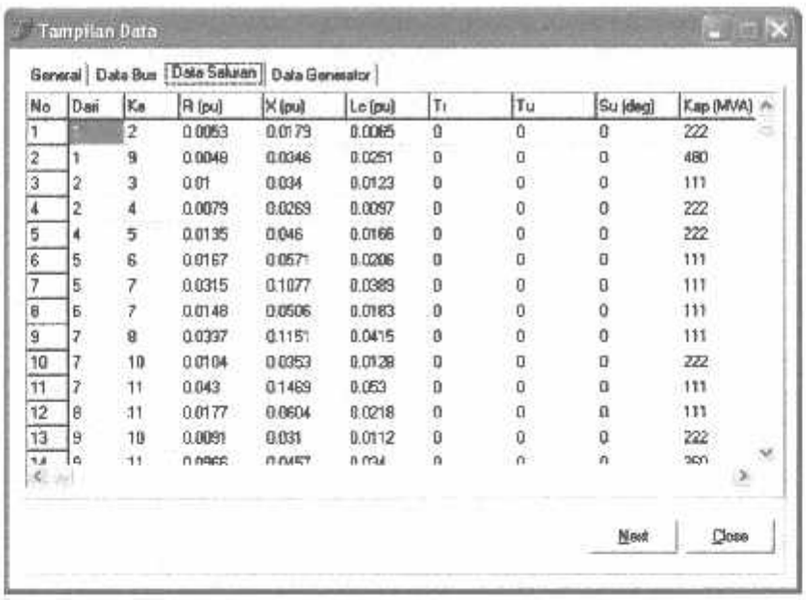
Gambar 4.9
Tampilan Masukan Data.

4. Kemudian tekan tombol Data Bus



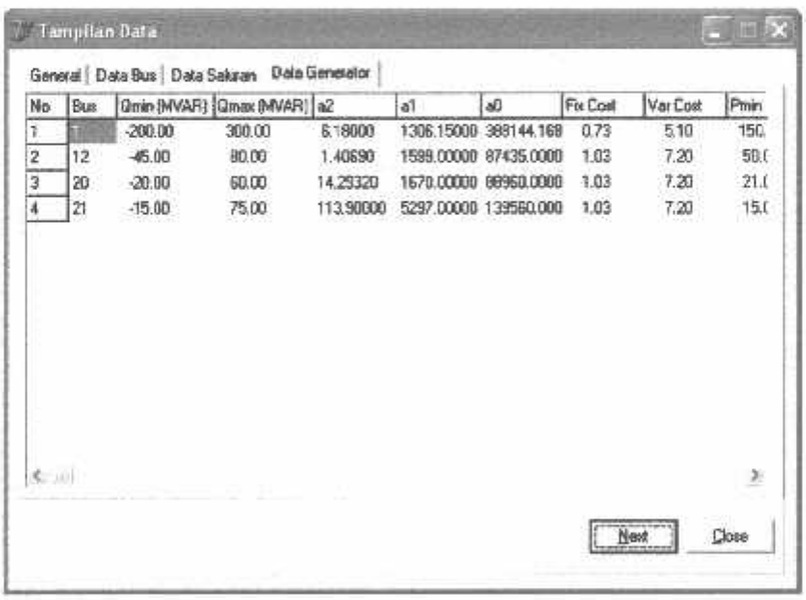
Gambar 4.10
Tampilan Data Bus

5. Kemudian tekan data saluran.



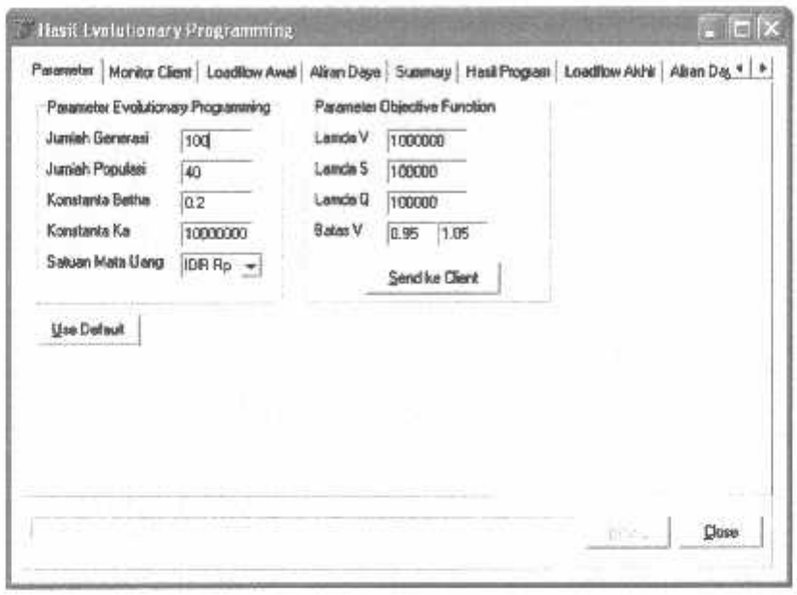
Gambar 4.11
Tampilan Data Saluran

6. Tekan tombol data generator



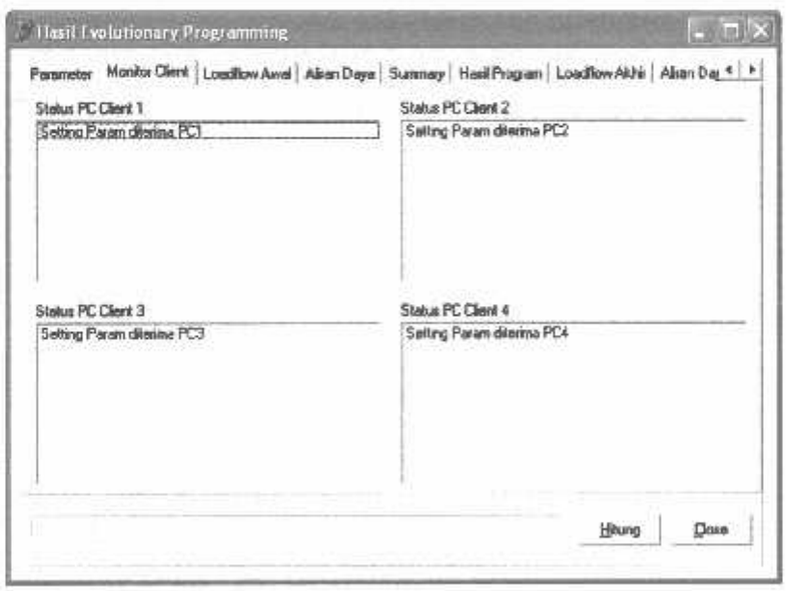
Gambar 4.12
Tampilan Data Generator

7. Tekan tombol next, kemudian tekan tombol *Use Default* untuk memasukkan parameter EP.



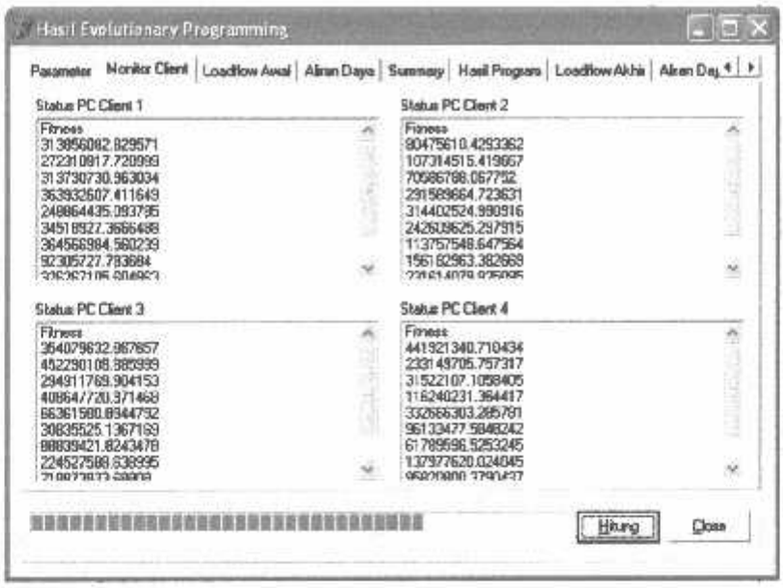
Gambar 4.13
Tampilan Parameter EP.

8. Kemudian tekan tombol *Send* ke *Client* dan tombol *Monitor client* untuk melihat setting parameter yang dikirim apakah sudah di terima oleh *PC client.*



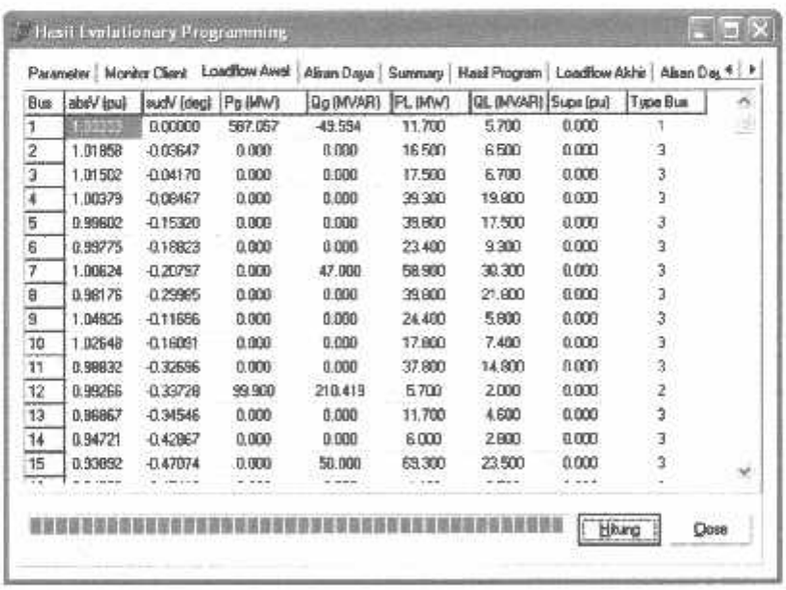
Gambar 4.14
Tampilan Monitor Status PC Client

9. Selanjutnya tekan tombol Hitung untuk menjalankan program.



Gambar 4.15
Tampilan dari *Monitor Client*

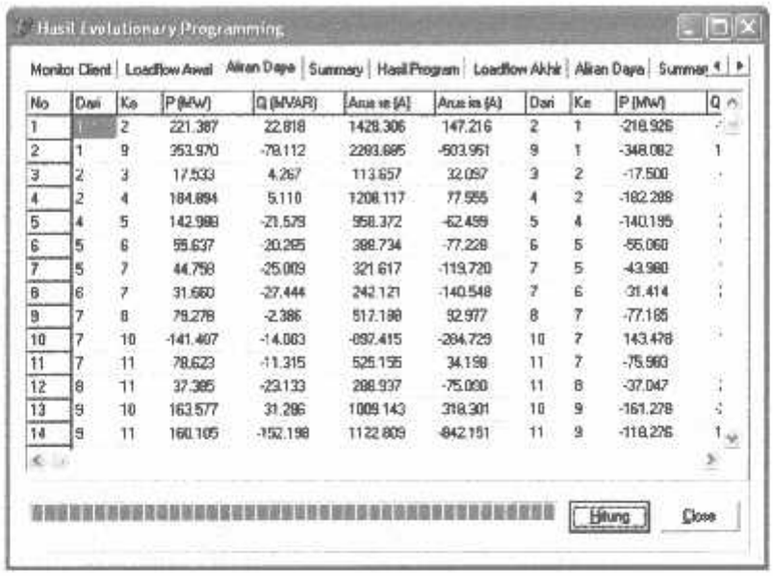
10. Tekan tombol *Load flow* awal untuk melihat hasil perhitungan pada kondisi awal (sebelum optimasi).



Bus	absV (pu)	sudV (deg)	Pg (MW)	Qg (MVAR)	PL (MW)	QL (MVAR)	Supa (pu)	Type Bus
1	[illegible]	0.00000	567.057	-49.554	11.700	5.700	0.000	1
2	1.01858	-0.03647	0.000	0.000	16.500	6.500	0.000	3
3	1.01502	-0.04170	0.000	0.000	17.500	6.700	0.000	3
4	1.00379	-0.08467	0.000	0.000	39.300	19.800	0.000	3
5	0.99802	-0.15320	0.000	0.000	39.800	17.500	0.000	3
6	0.99775	-0.18823	0.000	0.000	23.400	9.300	0.000	3
7	1.00624	-0.20757	0.000	47.000	58.900	30.300	0.000	3
8	0.98176	-0.25965	0.000	0.000	39.600	21.800	0.000	3
9	1.04825	-0.11656	0.000	0.000	24.400	5.800	0.000	3
10	1.02648	-0.16091	0.000	0.000	17.800	7.400	0.000	3
11	0.98832	-0.32696	0.000	0.000	37.800	14.800	0.000	3
12	0.99266	-0.33728	99.900	210.419	5.700	2.000	0.000	2
13	0.96867	-0.34546	0.000	0.000	11.700	4.600	0.000	3
14	0.94721	-0.42967	0.000	0.000	6.000	2.800	0.000	3
15	0.93692	-0.47074	0.000	50.000	69.300	23.500	0.000	3

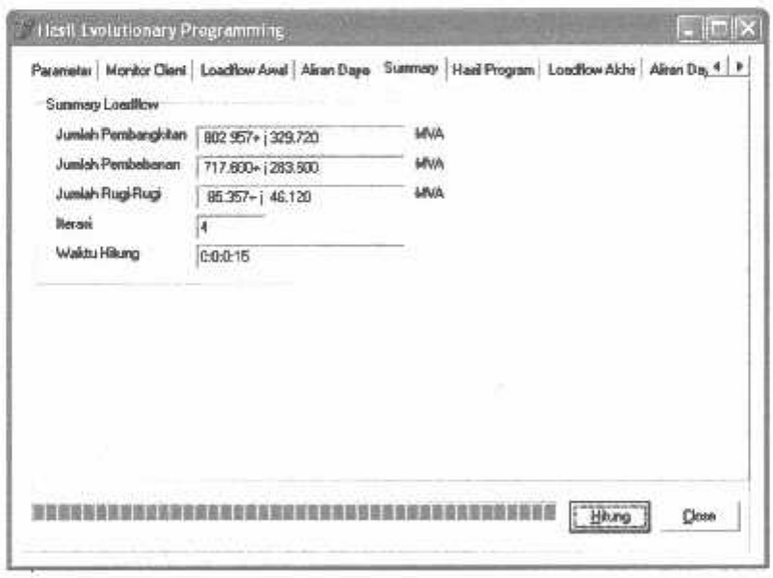
Gambar 4.16
Tampilan hasil *Load Flow* pada Kondisi Awal (Sebelum Optimasi)

11. Tekan tombol Aliran Daya untuk melihat kondisi aliran daya pada kondisi awal (sebelum optimasi).



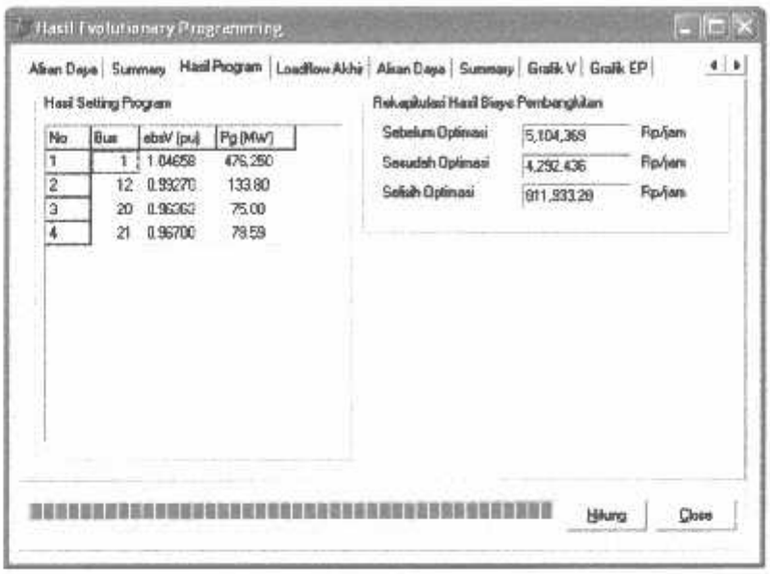
Gambar 4.17
Tampilan Hasil Aliran Daya Pada Kondisi Awal (Sebelum Optimasi)

12. Tekan tombol *Summary* untuk melihat *summary load flow* (sebelum optimasi)



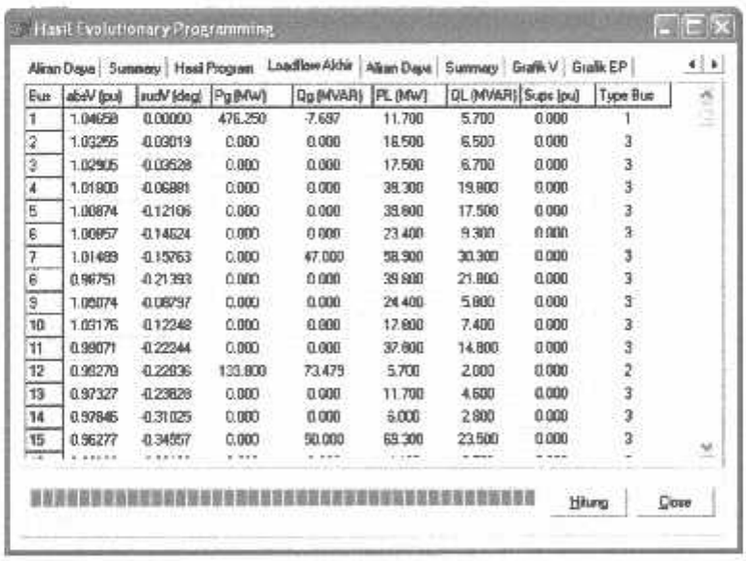
Gambar 4.18
Tampilan *Summary Load Flow* Sebelum Optimasi

13. Tekan tombol Hasil Program untuk melihat Rekapitulasi Biaya Pembangkitan.



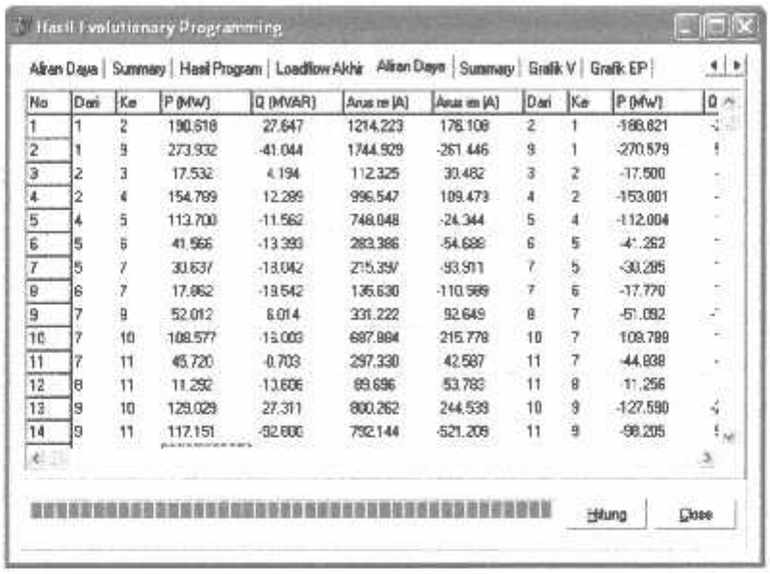
Gambar 4.19
Tampilan Hasil Program pada Kondisi Akhir (Setelah Optimasi)

14. Tekan tombol *Load Flow* akhir untuk melihat hasil perhitungan pada kondisi akhir (setelah optimasi).



Gambar 4.20
Tampilan Hasil *Load Flow* pada Kondisi Akhir (Setelah Optimasi)

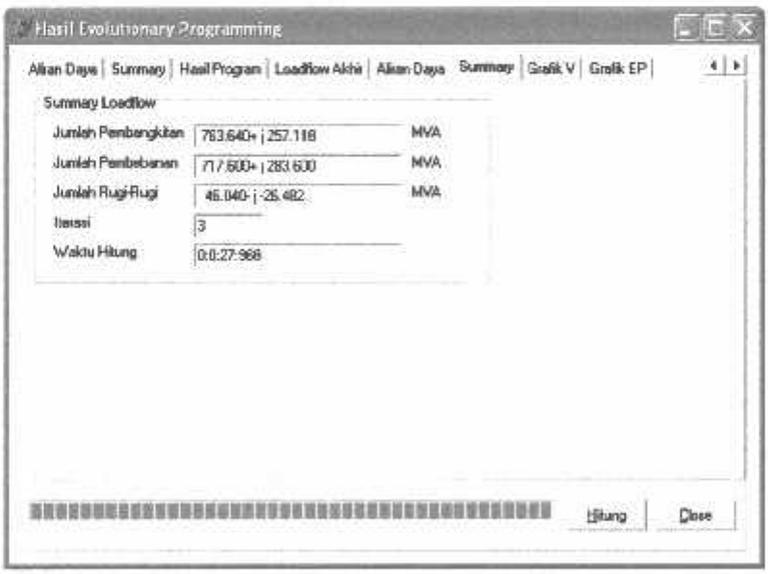
15. Tekan tombol Aliran Daya untuk melihat hasil perhitungan pada kondisi akhir (setelah optimasi).



No	Dari	Ke	P (MW)	Q (MVAR)	Arus re (A)	Arus im (A)	Dari	Ke	P (MW)	Q
1	1	2	190.518	27.647	1214.223	176.108	2	1	-188.821	
2	1	9	273.932	-41.044	1744.929	-261.445	9	1	-270.579	
3	2	3	17.532	4.194	112.325	30.482	3	2	-17.500	
4	2	4	154.799	12.289	996.547	109.473	4	2	-153.001	
5	4	5	113.700	-11.562	748.048	-24.344	5	4	-112.004	
6	5	6	41.566	-13.393	283.386	-54.688	6	5	-41.262	
7	5	7	30.637	-13.042	215.359	-93.911	7	5	-30.285	
8	6	7	17.862	-19.542	135.630	-110.589	7	6	-17.770	
9	7	8	52.012	6.014	331.222	92.649	8	7	-51.092	
10	7	10	108.577	-15.009	687.864	215.778	10	7	109.789	
11	7	11	45.720	-0.703	297.330	42.587	11	7	-44.838	
12	8	11	11.292	-13.606	89.696	53.783	11	8	-11.256	
13	9	10	129.029	27.311	800.262	244.539	10	9	-127.590	
14	9	11	117.151	-92.600	792.144	-521.209	11	9	-98.205	

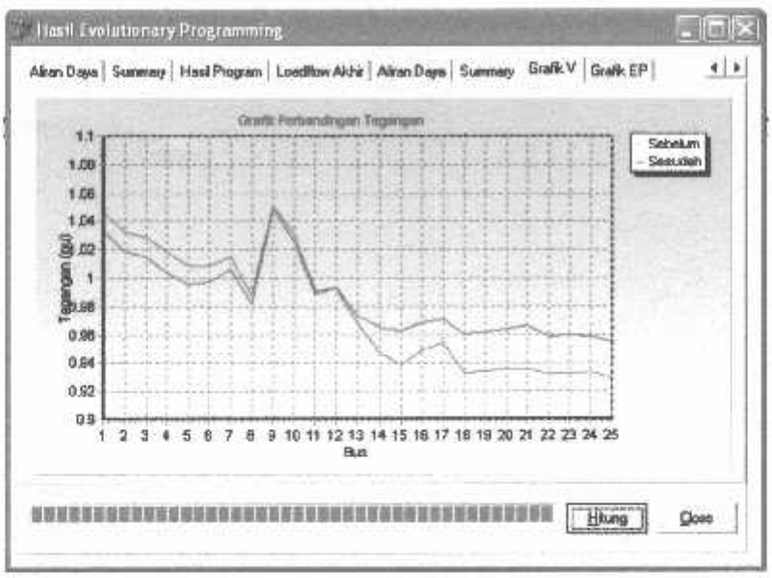
Gambar 4.21
Tampilan Aliran Daya pada Kondisi Akhir (Setelah Optimasi)

16. Kemudian tekan tombol *summary* untuk melihat *summary load flow* setelah optimasi.



Gambar 4.22
Tampilan *Summary Load Flow* Setelah Optimasi

17. Tekan tombol Grafik untuk melihat grafik tegangan.



Gambar 4.23
Tampilan Grafik Nilai Tegangan Sebelum dan Sesudah Optimasi

4.7. Hasil dan Analisis Hasil Perhitungan *Optimal Power Flow Menggunakan Metode Parallel Evolutionary Programming* Pada Saluran Transmisi 150 kV Sub Sistem Paiton dan Bali

4.7.1. Hasil Perhitungan Sebelum Optimasi

Tabel 4.10
Hasil Perhitungan Tegangan, Sudut Tegangan, Pembangkitan dan Pembebanan Sebelum Optimasi

Bus No.	Tegangan		Pembangkit		Pembebanan	
	Magnitude (pu)	Sudut (deg)	Aktif (MW)	Reaktif (MVAR)	Aktif (MW)	Reaktif (MVAR)
1	1.03333	0.00000	587.057	-49.594	11.700	5.700
2	1.01858	-0.03647	0.000	0.000	16.500	6.500
3	1.01502	-0.04170	0.000	0.000	17.500	6.700
4	1.00379	-0.08467	0.000	0.000	39.300	19.800
5	0.99602	-0.15320	0.000	0.000	39.800	17.500
6	0.99775	-0.18823	0.000	0.000	23.400	9.300

7	1.00624	-0.20797	0.000	47.000	58.900	30.300
8	0.98176	-0.29985	0.000	0.000	39.800	21.800
9	1.04926	-0.11656	0.000	0.000	24.400	5.800
10	1.02648	-0.16091	0.000	0.000	17.800	7.400
11	0.98832	-0.32696	0.000	0.000	37.800	14.800
12	0.99266	-0.33728	99.900	210.419	5.700	2.000
13	0.96867	-0.34546	0.000	0.000	11.700	4.600
14	0.94721	-0.42867	0.000	0.000	6.000	2.800
15	0.93892	-0.47074	0.000	50.000	69.300	23.500
16	0.94896	-0.45410	0.000	0.000	4.400	0.500
17	0.95427	-0.44293	0.000	0.000	24.800	8.800
18	0.93337	-0.48116	0.000	0.000	32.800	13.300
19	0.93426	-0.48136	0.000	0.000	76.400	28.900
20	0.93600	-0.48031	31.400	22.777	0.000	0.000
21	0.93600	-0.47593	84.600	-0.882	0.000	0.000
22	0.93219	-0.48896	0.000	25.000	51.400	19.300
23	0.93270	-0.48518	0.000	25.000	60.300	17.700
24	0.93321	-0.48226	0.000	0.000	33.700	9.600
25	0.92968	-0.48642	0.000	0.000	14.200	7.000

Tabel 4.11
Hasil Perhitungan Aliran Daya Antar Saluran Sebelum Optimasi

No.	Saluran		Daya	
	Dari	Ke	Aktif (MW)	Reaktif (MVAR)
1	1	2	221.387	22.818
2	1	9	353.970	-78.112
3	2	3	17.970	4.267
4	2	4	184.894	5.110
5	4	5	142.988	-21.579
6	5	6	55.637	-20.265
7	5	7	44.758	-25.009
8	6	7	31.660	-27.444
9	7	8	79.278	-2.386
10	7	10	-141.407	-14.083
11	7	11	78.623	-11.315
12	8	11	37.385	-23.133
13	9	10	163.577	31.286
14	9	11	160.105	-152.198
15	11	12	193.505	-209.570
16	12	13	207.586	-8.220
17	12	17	78.786	4.154
18	13	14	96.473	-5.365
19	13	15	94.335	-6.559
20	14	15	87.793	-10.651
21	15	16	-22.316	-3.889

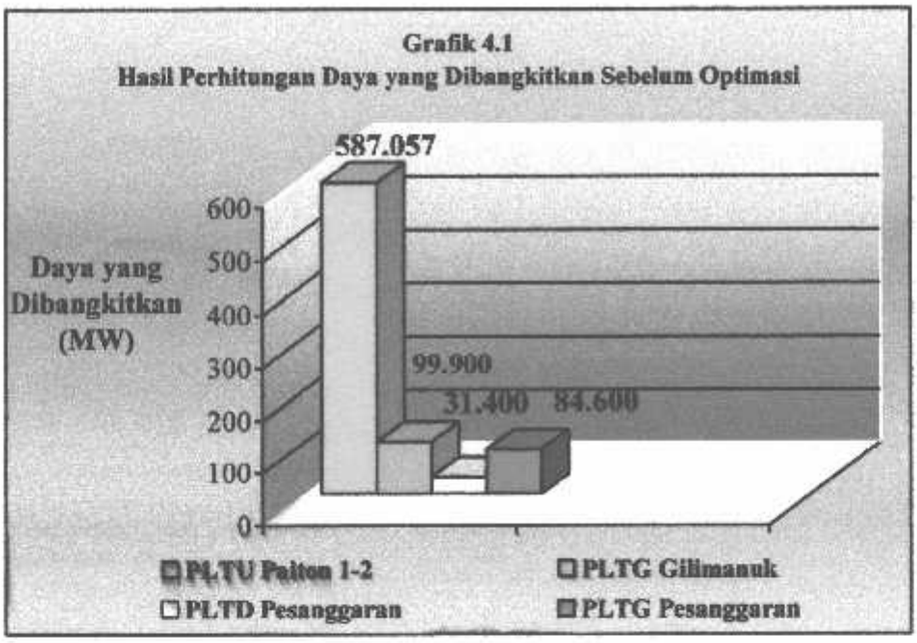
22	15	17	-23.998	-3.940
23	15	18	32.177	5.180
24	15	19	31.883	2.548
25	15	22	27.057	-4.341
26	15	24	62.881	9.506
27	16	17	-26.922	-0.506
28	18	19	-0.751	-6.527
29	19	20	-31.366	-22.701
30	19	21	-84.437	1.339
31	19	22	24.681	-5.590
32	19	23	45.729	-3.486
33	23	24	-14.660	4.081
34	24	25	14.228	5.173

Tabel 4.12
Total Pembangkitan, Pembebanan dan Rugi-rugi Saluran Sebelum Optimasi

Pembangkitan		**Pembebanan**		**Rugi-rugi**		**Waktu komputasi**
Aktif (MW)	**Reaktif (MVAR)**	**Aktif (MW)**	**Reaktif (MVAR)**	**Aktif (MW)**	**Reaktif (MVAR)**	
802.957	329.720	717.600	283.600	85.357	46.120	0 :0:0:16

Tabel 4.13
Hasil Perhitungan Daya Yang Dibangkitkan dan Biaya Operasi Sebelum Optimasi

No	**Nama Pembangkit**	**Daya Yang Dibangkitkan (MW)**	**Biaya Operasi (Rupiah / Jam)**
1	PLTU Paiton 1 & 2	587.057	Rp 3.284.778,662
2	PLTG Gilimanuk	99.900	Rp 261.215,9761
3	PLTD Pesanggaran	31.400	Rp 155.490,5235
4	PLTG Pesanggaran	84.600	Rp 1.402.886,724
Total		**802.957**	**Rp 5.104.371,886**



4.7.2. Hasil Perhitungan Setelah Optimasi

Tabel 4.14
Hasil Perhitungan Tegangan, Sudut Tegangan, Pembangkitan dan Pembebanan Setelah Optimasi.

Bus No.	Tegangan		Pembangkit		Pembebanan	
	Magnitude (pu)	Sudut (deg)	Aktif (MW)	Reaktif (MVAR)	Aktif (MW)	Reaktif (MVAR)
1	1.04658	0.00000	476.250	-7.697	11.700	5.700
2	1.03255	-0.03019	0.000	0.000	16.500	6.500
3	1.02905	-0.03528	0.000	0.000	17.500	6.700
4	1.01800	-0.06881	0.000	0.000	39.300	19.800
5	1.00874	-0.12106	0.000	0.000	39.800	17.500
6	1.00857	-0.14624	0.000	0.000	23.400	9.300
7	1.01489	-0.15763	0.000	47.000	58.900	30.300
8	0.98751	-0.21393	0.000	0.000	39.800	21.800
9	1.05074	-0.08797	0.000	0.000	24.400	5.800
10	1.03176	-0.12248	0.000	0.000	17.800	7.400
11	0.99071	-0.22244	0.000	0.000	37.800	14.800
12	0.99270	-0.22836	133.800	73.479	5.700	2.000
13	0.97327	-0.23828	0.000	0.000	11.700	4.600
14	0.96492	-0.31025	0.000	0.000	6.000	2.800
15	0.96277	-0.34557	0.000	50.000	69.300	23.500
16	0.96822	-0.33182	0.000	0.000	4.400	0.500
17	0.97098	-0.32217	0.000	0.000	24.800	8.800
18	0.96031	-0.35315	0.000	0.000	32.800	13.300
19	0.96240	-0.35235	0.000	0.000	76.400	28.900

20	0.96363	-0.34864	75.000	-7.054	0.000	0.000
21	0.96700	-0.34873	78.590	51.391	0.000	0.000
22	0.95917	-0.36054	0.000	25.000	51.400	19.300
23	0.96009	-0.35651	0.000	25.000	60.300	17.700
24	0.95888	-0.35496	0.000	0.000	33.700	9.600
25	0.95547	-0.35891	0.000	0.000	14.200	7.000

Tabel 4.15
Hasil Perhitungan Aliran Daya Antar Saluran Setelah Optimasi

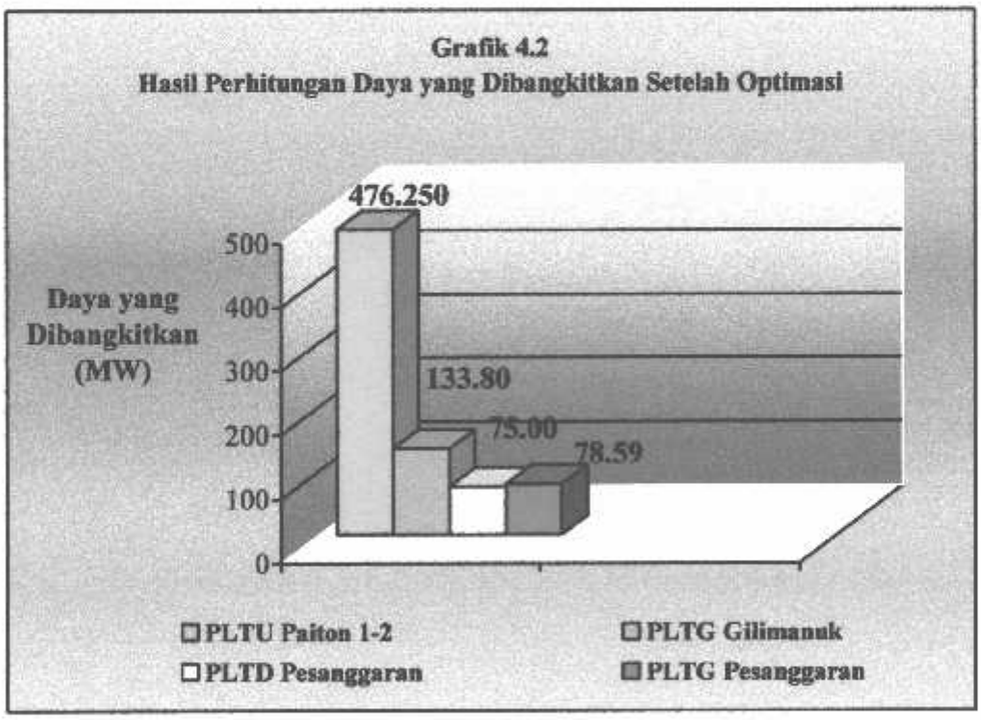
No.	Saluran		Daya	
	Dari	Ke	Aktif (MW)	Reaktif (MVAR)
1	1	2	190.618	27.642
2	1	9	273.932	-41.044
3	2	3	17.532	40194
4	2	4	154.789	12.289
5	4	5	113.700	-11.562
6	5	6	41.566	-13.393
7	5	7	30.637	-18.042
8	6	7	17.863	-19.542
9	7	8	52.012	6.014
10	7	10	-108.577	-16.003
11	7	11	45.720	-0.703
12	8	11	11.290	-10.606
13	9	10	129.029	27.311
14	9	11	117.151	-92.800
15	11	12	116.498	-108.942
16	12	13	175.814	-32.347
17	12	17	68.369	-6.047
18	13	14	81.236	-17.257
19	13	15	79.125	-18.392
20	14	15	73.295	-20.277
21	15	16	-17.590	-0.476
22	15	17	-19.232	-0.530
23	15	18	23.023	-1.096
24	15	19	18.848	-6.301
25	15	22	21.963	-6.905
26	15	24	52.423	3.507
27	16	17	-22.112	3.507
28	18	19	-9.837	-12.499
29	19	20	-74.878	7.329
30	19	21	-78.402	-50.863
31	19	22	29.720	-3.396
32	19	23	56.116	2.038
33	23	24	-4.310	9.561
34	24	25	14.227	5.062

Tabel 4.16
Total Pembangkitan, Pembebanan dan Rugi-rugi Saluran Setelah Optimasi

Pembangkitan		Pembebanan		Rugi-rugi		Waktu komputasi
Aktif (MW)	Reaktif (MVAR)	Aktif (MW)	Reaktif (MVAR)	Aktif (MW)	Reaktif (MVAR)	
763.640	257.118	717.600	283.600	46.040	26.482	0:0:27:968

Tabel 4.17
Hasil Perhitungan Daya Yang Dibangkitkan dan Biaya Operasi Setelah Optimasi

No	Nama Pembangkit	Daya Yang Dibangkitkan (MW)	Biaya Operasi (Rupiah / Jam)
1	PLTU Paiton 1 & 2	476.250	Rp 2.411.909,012
2	PLTG Gilimanuk	133.80	Rp 326.568,142
3	PLTD Pesanggaran	75.00	Rp 294.609,25
4	PLTG Pesanggaran	78.59	Rp 1.259.341,83
Total		**763.640**	**Rp 4.292.428,232**



4.8. Perbandingan Hasil Perhitungan Sebelum dan Setelah Optimasi *Optimal Power Flow* Menggunakan Metode *Parallel Evolutionary Programming.*

Dari semua hasil perhitungan diatas maka bisa dibuat tabel-tabel perbandingan sesuai dengan tujuan yang ingin dicapai.

4.8.1. Perbandingan Tingkat Tegangan dan Sudut Tegangan Pada Tiap Bus

Berikut ini adalah tabel perbandingan tingkat tegangan dan sudut tegangan pada tiap bus.

Tabel 4.18
Perbandingan Tingkat Tegangan dan Sudut Tegangan Pada Tiap Bus

Bus No	Sebelum Optimasi		Sesudah Optimasi	
	Tegangan (pu)	Sudut Tegangan (deg)	Tegangan (pu)	Sudut Tegangan (deg)
1	1.03333	0.00000	1.04658	0.00000
2	1.01858	-0.03647	1.03255	-0.03019
3	1.01502	-0.04170	1.02905	-0.03528
4	1.00379	-0.08467	1.01800	-0.06881
5	0.99602	-0.15320	1.00874	-0.12106
6	0.99775	-0.18823	1.00857	-0.14624
7	1.00624	-0.20797	1.01489	-0.15763
8	0.98176	-0.29985	0.98751	-0.21393
9	1.04926	-0.11656	1.05074	-0.08797
10	1.02648	-0.16091	1.03176	-0.12248
11	0.98832	-0.32696	0.99071	-0.22244
12	0.99266	-0.33728	0.99270	-0.22836
13	0.96867	-0.34546	0.97327	-0.23828
14	0.94721	-0.42867	0.97846	-0.31025
15	0.93892	-0.47074	0.96277	-0.34557
16	0.94896	-0.45410	0.96822	-0.33182
17	0.95427	-0.44293	0.97098	-0.32217
18	0.93337	-0.48116	0.96031	-0.35315
19	0.93426	-0.48136	0.96240	-0.35235
20	0.93600	-0.48031	0.96363	-0.34864
21	0.93600	-0.47593	0.96700	-0.34873
22	0.93219	-0.48896	0.95917	-0.36054

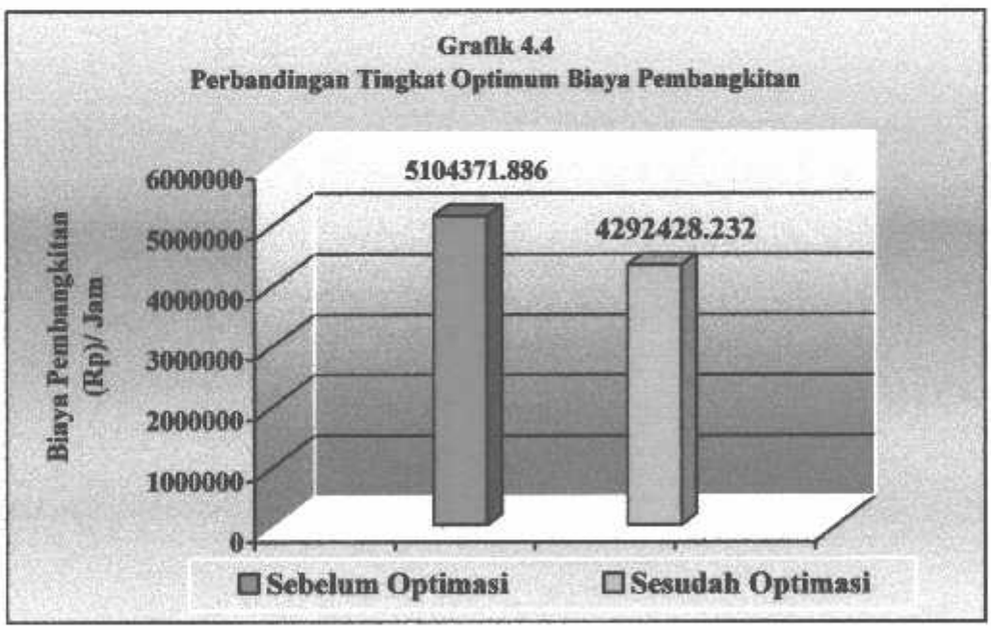
4.8.3. Tingkat Optimum Biaya Pembangkitan

Berikut ini adalah tabel yang berisi perbandingan biaya optimum pembangkitan antara hasil perhitungan sebelum dengan sesudah optimasi.

Tabel 4.19
Perbandingan Tingkat Optimum Biaya Pembangkitan

No	Nama Pembangkit	Sebelum Optimasi	Setelah Optimasi
1	PLTU Paiton 1 & 2	Rp 3.284.778,662	Rp 2.411.909,012
2	PLTG Gilimanuk	Rp 261.215,9761	Rp 326.568,142
3	PLTD Pesanggaran	Rp 155.490,5235	Rp 294.609,25
4	PLTG Pesanggaran	Rp 1.402.886,724	Rp 1.259.341,83
Total		**Rp 5.104.371,886**	**Rp 4.292.428,232**

Dari tabel tingkat optimum diatas dapat dilihat bahwa besarnya biaya produksi pembangkitan sebelum optimasi adalah Rp 5.104.371, 886,- per jam , sedangkan besarnya biaya produksi pembangkitan setelah optimasi adalah Rp 4.292.428,232,- per jam. Jadi selisih antara biaya produksi pembangkitan sebelum dengan sesudah optimasi adalah Rp 811.943, 654, - per jam atau terjadi optimasi biaya sebesar 15.90 %.



BAB V

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Setelah dilakukan analisis perhitungan *Optimal Power Flow (OPF)* dengan menggunakan metode *Parallel Evolutionary Programming (PEP)* maka dapat diambil kesimpulan sebagai berikut:

1. Dari out put program *Optimal Power Flow* dengan menggunakan metode *Parallel Evolutionary Programming* didapatkan hasil bahwa, untuk tingkat tegangan masih dalam range yang di ijinkan yaitu antara -10% sampai 5% dari tegangan nominal 150 kV, dan untuk rugi – rugi daya pada saluran berhasil dilakukan minimalisasi sebesar 39.317 MW dan 19.638 MVAR. Dimana sebelum optimasi dihasilkan rugi-rugi daya sebesar 85.357 MW dan 46.120 MVAR, sedangkan besarnya rugi-rugi daya setelah optimasi adalah 46.040 MW dan 26.482 MVAR.

2. Penyelesaian masalah OPF dengan menggunakan metode *Parallel Evolutionary Programming* pada sub sistem 150 kV Paiton dan Bali untuk melayani permintaan beban puncak sebesar 717.600 MW menghasilkan biaya operasi yang lebih Optimum. Biaya produksi pembangkitan sebelum optimasi adalah Rp **5.104.371, 886** per jam, sedangkan besarnya biaya produksi pembangkitan setelah optimasi adalah **Rp 4.292.428, 232, -** per jam. Jadi selisih antara biaya produksi pembangkitan sebelum dengan sesudah optimasi adalah Rp **811.943, 654, -** per jam atau terjadi optimasi biaya sebesar **15.90 %.**

3. Dalam pemecahan masalah *Optimal Power Flow*, waktu komputasi dari metode *Parallel Evolutionary Programming* ternyata lebih cepat dari pada waktu komputasi dari metode *Evolutionary Programming* [16], yaitu 0:0:27:968.

5.2 Saran

Penggunaan metode *Parallel Evolutionary Programming* untuk menyelesaikan permasalahan OPF masih perlu dikembangkan lagi terhadap sistem jaringan yang lebih luas dengan menggunakan komputer yang mempunyai kapasitas memorynya besar dan kecepatan processornya tinggi, sehingga waktu komputasinya lebih cepat yang nantinya dapat digunakan secara lebih efektif dan efisien dalam pemecahan permasalahan optimasi biaya pembangkitan.

DAFTAR PUSTAKA

[1] Dommel, H. W. and W. F. Tinney, "Optimal Power Flow Solutions", IEEE Trans. Power Apparat. Syst., Vol. PAS – 87, pp. 1866-1879, Oct 1968.

[2] Bakirtzis, Anastasios G., Pandel N. Biskas, Christoforos E. Zoumas, Vasilos Petridis, *"Optimal Power Flow by Enhanced Genetic Algorithm"*, IEEE Transactions On Power Systems, vol. 17. No. 2. May 2002.

[3] Stevenson, William D., Jr, *"Analisa Sistem Tenaga Listrik"*, Penerbit Erlangga edisi ke-empat, 1996.

[4] Wood, Allan J. and B. F. Wollwnberg.*"Power Generation, Operation, and Control"*, John Wiley & Sons, Inc., 1996.

[5] Zuhal, *"Dasar Teknik Tenaga Listrik dan Elektronika Daya"*, PT Gramedia Utama, Jakarta, 1995.

[6] Stagg, Glenn W, and Ahmed H. El-Abiad *"Computer Methodes in Power System Analysis"*.

[7] Yuryevich, J. and K.P. Wong, *"Evolutionary Programming Based Optimal Power Flow Algorithm"*, IEEE Trans.on Power System, Vol. 14, No. 4, pp.1245-1250, Nov. 1999.

[8] C. H. Lo., C. Y. Chung., D. H. M. Nguyen and K. P. Wong.., *" Parallel Evolutionary Programming for Optimal Power Flow"*, IEEE International Conference on Electric Utility Deregulation, Restructuring and Power Technologies (DRPT), April 2004

[9] Luke, Brian T.,*"Overview of Evolutionary Programing Methods"*, Learning From the Web.net.

[10] W. Gropp, E. Lusk, and A. Skjellum, *"Using MPI: Portable Parallel Programming with the message Pasing Interface"*, London: MIT Press, 1994

[11] Kusumadewi, Sri, "Artificial Intelligence (Teknik dan Aplikasinya)", Graha Ilmu, Yogyakarta, 2003

[12] Spears, William M., *"An Overview of Evolutionary Computations"*.

[13] Hussain, Talib S., *"An Introduction to Evolutionary Computation"*, Department of Computing and Information Science Queen's University, Kingston.

[14] Pai, M. A., *"Computer Techniques in Power System Analysis"*, Tata Mc Graw-Hill Publissing Company Limited.

[15] Syafena, Jatri, Skripsi, *"Economic Dispact Dengan Optimasi Daya Aktif dan Reaktif Pada Pembangkit Thermal di Area IV"*, 2002.

[16] Iswandi, Yudi, Skripsi, *"Optimal Power Flow Menggunakan Metode Evolutionary Programming Pada Sub Sistem 150kV Paiton Dan Bali"*, 2004

[17] Data Load Flow Region Jawa Timur dan Bali, Sub Sistem 150kV Paiton dan Bali, Rabu 30 Maret 2005, Pukul 19:30 WIB, PT. PLN (Persero) P3B Region Jawa Timur dan Bali, Waru.

[18] Data Saluran Transmisi Sub Sistem 150 kV Paiton dan Bali.

LAMPIRAN

1. Berita Acara Ujian Skripsi.
2. Formulir Perbaikan Skripsi.
3. Lembar Bimbingan Skripsi.
4. Formulir Bimbingan Skripsi.
5. Surat Survey di PT. PLN (persero) Jawa-Bali, Region Jawa Timur dan Bali.
6. Data ***Load Flow*** Region Jawa Timur dan Bali, Sub Sistem 150 kV Paiton dan Bali, Rabu 30 Maret 2005 pukul: 19:30 WIB PT. PLN (persero) P3B Region Jawa Timur dan Bali.
7. Data Saluran Transmisi Sub Sistem 150 kV Paiton dan Bali.
8. Listing Prigram Borland Delphi 7.0 ***Optimal Power Flow*** **Menggunakan Metode *Parallel Evolutionary Programming*.**

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

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

Nama Mahasiswa	:	**NUR ASYIK HIDAYATULLAH**
N.I.M	:	**00.12.041**
Jurusan	:	**Teknik Elektro S-1**
Konsentrasi	:	**Teknik Energi Listrik**
Judul Skripsi	:	***OPTIMAL POWER FLOW*** **MENGGUNAKAN METODE** ***PARALLEL EVOLUTIONARY PROGRAMMING*** **PADA SUB SISTEM 150 kV PAITON - BALI.**

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

Hari	:	**Sabtu**
Tanggal	:	**18 Maret 2006**
Dengan Nilai	:	**80,95 (A)**

Panitia Ujian

(Ir. Machtar Asroni, MSME)	**(Ir. F. Yudi Limpraptono, MT)**
Ketua	**Sekretaris**

Anggota Penguji

(Ir. H. Almizan Abdullah, MSEE)	**(Ir. I Made Wartana, MT)**
Penguji Pertama	**Penguji Kedua**

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

FORMULIR PERBAIKAN SKRIPSI

Nama	:	**NUR ASYIK HIDAYATULLAH**
NIM	:	**00.12.041**
Masa Bimbingan	:	**21 Maret 2005 – 21 Maret 2006**
Judul Skripsi	:	***OPTIMAL POWER FLOW* MENGGUNAKAN METODE *PARALLEL EVOLUTIONARY PROGRAMMING* PADA SUB SISTEM 150 kV PAITON - BALI.**

No	Tanggal	Uraian	Paraf Pembimbing
1	23 Maret 2006	Perbaikan batasan masalah.	
2	23 Maret 2006	Penambahan acuan biaya bahan bakar.	
3	23 Maret 2006	Perbaikan *flow chart load flow*.	

Disetujui,

(Ir. H. Almizan Abdullah, MSEE.)
Penguji Pertama

(Ir. I Made Wartana, MT)
Penguji Kedua

Mengetahui,
Dosen Pembimbing

(Ir. H. CHOIRI)

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

LEMBAR BIMBINGAN SKRIPSI

1.	Nama	: NUR ASYIK HIDAYATULLAH
2.	NIM	: 00.12.041
3.	Jurusan	: Teknik Elektro S-1
4.	Konsentrasi	: Teknik Energi Listrik
5.	Judul Skripsi	: *OPTIMAL POWER FLOW* MENGGUNAKAN METODE *PARALLEL EVOLUTIONARY PROGRAMMING* PADA SUB SISTEM 150 kV PAITON-BALI.
6.	Tanggal Mengajukan Skripsi	: 21 Maret 2005
7.	Tanggal Menyelesaikan Skripsi	: 21 Desember 2005
8.	Dosen Pembimbing	: Ir. H. Choiri
9.	Telah dievaluasi dengan nilai	: 85,00 (Delapan Puluh Lima Koma Nol)

Malang, Maret 2006

Mengetahui,
Ketua Jurusan Teknik Elektro S-1

Ir. F. Yudi Limpraptono, MT
NIP. Y. 1039500274

Disetujui,
Dosen Pembimbing

Ir. H. Choiri
NIP. 130703042

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

FORMULIR BIMBINGAN SKRIPSI

Nama : NUR ASYIK HIDAYATULLAH
Nim : 00.12.041
Masa Bimbingan : 21 Maret 2005 s/d 21 Maret 2006
Judul Skripsi : OPTIMAL POWER FLOW MENGGUNAKAN METODE PARALLEL EVOLUTIONARY PROGRAMMING PADA SUS SISTEM 150 kV PAITON – BALI

No	Tanggal	Uraian	Paraf Pembimbing
1	28-5-2005	Konsultasi setelah seminar proposal dan sekaligus membuat program komputer.	
2	14 -5-2005	Konsultasi BAB I (Tujuan dan Rumusan masalah harus sinkron).	
3	17-5-2005	Konsultasi BAB II (Acc).	
4	28-10-2005	Konsultasi BAB III (Acc).	
5	28-11-2005	Konsultasi BAB IV (Acc).	
6	15-12-2005	Konsultasi BAB V (Revisi).	
7	21-12-2005	Konsultasi akhir keseluruhan BAB (Acc).	
		Acc di seminarkan	

Malang, Desember 2005
Dosen Pembimbing,

(Ir. H.Choiri)

Form.S-4b

PT PLN (PERSERO)
PENYALURAN DAN PUSAT PENGATUR BEBAN JAWA BALI
REGION JAWA TIMUR & BALI

Jl. Suningrat No. 45 Taman Sidoarjo 61257

Telepon : (031) 7882113, 7882114
Facsimile : (031) 7882578, 7881024

Kotak Pos : 4119 SBS
Bank : Bank Mandiri

Nomor : 092 /330/RJTB/2005.
Surat Sdr. No. : ITN-997/III.TA/2/2005.
Lampiran : 1 (satu) lampiran
Perihal : Ijin Survey/ Pengambilan Data.

13 APR 2005

Kepada

Yth. Dekan Fakultas Teknik.
Institut Teknologi Nasional Malang
Di
M A L A N G

Menunjuk surat Saudara nomor : ITN-997/III.TA/2/2005 tanggal 16 Maret 2005 perihal : Survey/ Permintaan Data, dengan ini diberitahukan bahwa kami tidak keberatan untuk memberikan ijin kepada Mahasiswa Saudara, bernama :

- NUR ASYIK H Nim : 00.12.041

Untuk melakukan Pengambilan Data pada PT. PLN (Persero) P3B Region Jawa Timur dan Bali Bidang OPHAR, dengan persyaratan sebagai berikut :

1. Mahasiswa tersebut diatas supaya mengisi dan menanda tangani Surat Pernyataan 1 (satu) lembar bermeterai Rp. 6.000,-
2. Mahasiswa yang bersangkutan agar mematuhi peraturan/ketentuan yang berlaku di PT. PLN (PERSERO) sehingga faktor-faktor kerahasiaan harus benar-benar diutamakan.
3. Semua biaya perjalanan, penginapan, makan dan lain sebagainya tidak menjadi tanggungan PT. PLN (Persero) P3B Region Jawa Timur dan Bali.
4. Buku Laporan Kerja Praktek Mahasiswa tersebut agar dikirimkan kepada PT. PLN (Persero) P3B Region Jawa Timur dan Bali 1 (satu) buah.
5. Untuk informasi lebih lanjut dapat menghubungi PT. PLN (Persero) P3B Region Jawa Timur dan Bali Cq. Bidang Enjiniring.

Demikian harap maklum dan terima kasih atas perhatian saudara.

a.n. MANAGER
MANAGER BIDANG ENJINIRING,

TRI AGUS CAHYONO

Tembusan Yth. :

1. M.SDM PLN P3B.
2. Sdr. Nur Asyik H.

Model 1001

4

PT.PLN (Persero)
Penyaluran Dan Pusat Pengatur Beban Jawa Bali
Region Jawa Timur dan Bali

SURAT PERNYATAAN

Yang bertanda tangan dibawah ini saya :

Nama : NUR ASYIK HIDAYATULLAH

Pria/ Wanita : PRIA

Tempat / Tanggal lahir : MADIUN / 12 - JANUARI - 1982

Alamat / no telepon : JL. CENGGER AYAM NO 25 MALANG
(0341) 495 375 EXT 82

Pekerjaan : MAHASISWA

Dengan ini saya menerangkan bahwa :

1. Saya bersedia dan setuju menanggung semua akibat yang ditimbulkan karena kesalahan maupun kelalaian saya dan semua akibat lainnya yang terjadi pada instalasi peralatan milik PLN selama melakukan Training/ Praktek Kerja/ Riset pada PT PLN (Persero) P3B Region Jawa Timur dan Bali, yang telah mendapat ijin dari PT PLN (Persero) P3B Region Jawa Timur dan Bali ;

2. Saya atas peringatan pertama akan membayar sepenuhnya , semua biaya yang langsung menimbulkan kerugian atau kecelakaan , karena kelalaian saya ;

3. Saya akan segera mematuhi semua petunjuk –petunjuk yang diberikan oleh Petugas PT PT PLN (Persero) P3B Region Jawa Timur dan Bali ;

4. Saya sanggup tidak membocorkan hal – hal yang bersifat rahasia perusahaan PT PLN (Persero) P3B Region Jawa Timur dan Bali dan bahan yang saya peroleh dalam Training/ Praktek Kerja/ Riset, dan tidak saya pergunakan untuk hal – hal yang dapat merugikan PT PLN (Persero) P3B Region Jawa Timur dan Bali

5. Saya sanggup menanggung sendiri segala sesuatu untuk keperluan Training/ Praktek Kerja/ Riset termasuk biaya perjalanan , penginapan makan dan sebagainya ;

6. Saya sanggup menyerahkan 1 (satu) buah buku laporan Training/ Praktek Kerja/ Riset kepada PT PLN (Persero) P3B Region Jawa Timur dan Bali, setelah saya presentasikan kepada Manager Bidang Enjiniring PT PLN (Persero) P3B Region Jawa Timur dan Bali mengenai tugas Training/ Praktek Kerja/ Riset.

7. Saya tunduk dan akan mentaati semua peraturan yang berlaku di PT PLN (Persero) P3B Region Jawa Timur dan Bali, dan saya sanggup tidak meninggalkan tugas kedinasan selama Training/ Praktek Kerja/ Riset.

Mengetahui
Manager Bidang SDM & AD

Ir. ZAENAL ARIFIN

Surabaya, 6/09 05
Yang membuat pernyataan

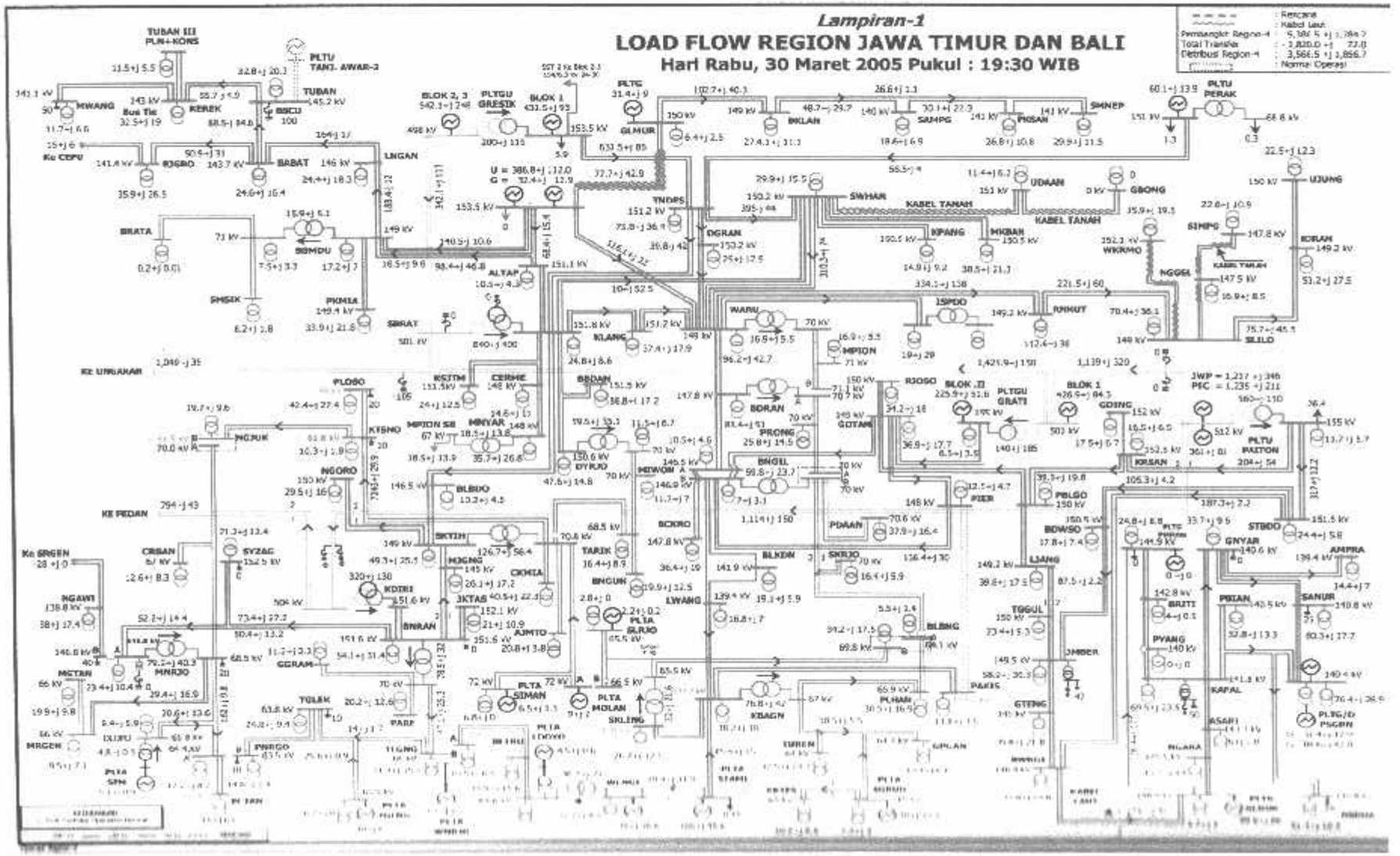
METERAI TEMPEL
6000

NUR ASYIK H

Lampiran-1

LOAD FLOW REGION JAWA TIMUR DAN BALI

Hari Rabu, 30 Maret 2005 Pukul : 19:30 WIB



DATA KARAKTERISTIK TRANSMISI UPT & SRB

Jml Sirkit	Dari	Ke	No. Sirkit	Data_Penghantar Teg (kV)	Jarak (Km)	Jenis		I Nom (Amp)	Tap CT Dari	Tap CT Ke	Tap WT Dari	Tap WT Ke	R (Ohm/km)	X (Ohm/km)	B (MyS/km)	R (Ohm)	X (Ohm)	B (Ohm)	Z act (Ohm)	Z base (Ohm)	Z pu (Ohm)	R pu (Ohm)	X pu (Ohm)	B pu (Ohm)	Daya (MVA)	TAHUN OPERASI	Keterangan
				(1)	(2)								(3)	(4)	(5)	(6)=(2)x(3)	(7)=(2)x(4)	(8)=(2)x(5)	(9)=akar (6)²+(7)²	(10)=(1)²/100	(11)=(9)/(10)	(12)=(6)/(10)	(13)=(7)/(10)	(14)=(8)(kV)²/0.1975	(15)=(1)x(A)x1.732/1000		MVA base = 100 150 kV = 0,1975 70 kV = 4,153

6. UPT Probolinggo

Jml Sirkit	Dari	Ke	No. Sirkit	Teg (kV)	Jarak (Km)	Jenis		I Nom (Amp)	Tap CT Dari	Tap CT Ke	Tap WT Dari	Tap WT Ke	R (Ohm/km)	X (Ohm/km)	B (MyS/km)	R (Ohm)	X (Ohm)	B (Ohm)	Z act (Ohm)	Z base (Ohm)	Z pu (Ohm)	R pu (Ohm)	X pu (Ohm)	B pu (Ohm)	Daya (MVA)	TAHUN OPERASI	Keterangan
1	PITONT	GRATI7	1	500	88.363	GANNET	4x282.6	2600	2000	2000			0.0251	0.2808	4.0490	2.2179	24.8123	357.7818	24.9113	2 500.000	0.0100	0.0009	0.0099	0.0073	2 424.8000		
1	PITONT	GRATI7	2	500	88.363	GANNET	4x282.6	2600	2000	2000			0.0251	0.2808	4.0490	2.2179	24.8123	357.7818	24.9113	2 500.000	0.0100	0.0009	0.0099	0.0072	2 424.8000		
1	BNGIL6	GDTAN6	1	150	15.805	ACSR AW	330 mm^2	740	1000	1000	1250	1500	0.1172	0.4003	2.8630	1.9896	6.7270	47.9447	7.0094	225.0000	0.0312	0.0088	0.0299	0.0108	192.2520		
1	BNGIL5	GDTAN6	2	150	15.805	ACSR AW	330 mm^2	740	1000	1000	1250	1500	0.1172	0.4003	2.8630	1.9896	6.7270	47.9447	7.0094	225.0000	0.0312	0.0088	0.0299	0.0108	192.2520		
1	BNGIL5	BDRAN6	1	150	9.800	ACSR AW	330 mm^2	740	1000	1000	1800	1500	0.1172	0.4003	2.8630	1.1486	3.9229	27.9594	4.0876	225.0000	0.0182	0.0051	0.0174	0.0063	192.2520		
1	BNGIL5	WARU6	1	150	9.800	ACSR AW	330 mm^2	740	1000	1000	1800	1800	0.1172	0.4003	2.8630	1.1486	3.9229	27.9594	4.0876	225.0000	0.0182	0.0051	0.0174	0.0063	192.2520		
1	BNGIL5	LWANG6	1	150	34.680	ACSR AW	330 mm^2	740	1000	1000	1800	1800	0.1172	0.4003	2.8630	4.0645	13.8824	98.9420	14.4652	225.0000	0.0643	0.0181	0.0617	0.0223	192.2520		
1	BNGIL6	BLKDN6	1	150	27.770	ACSR AW	330 mm^2	740	1000	1000	1500	1500	0.1172	0.4003	2.8630	3.2546	11.1163	79.2278	11.5830	225.0000	0.0515	0.0145	0.0494	0.0178	192.2520		
1	BNGIL5	PIER6	1	150	5.150	ACSR ZEBRA	2 x 435 mm^2	1620	2000	1000	1250	1250	0.3870	0.2807	4.0203	1.9931	1.4456	20.7045	2.4621	225.0000	0.0109	0.0089	0.0064	0.0047	420.8760		
1	BNGIL5	PIER6	2	150	5.150	ACSR ZEBRA	2 x 435 mm^2	1620	2000	1000	1250	1250	0.3870	0.2807	4.0203	1.9931	1.4456	20.7045	2.4621	225.0000	0.0109	0.0089	0.0064	0.0047	420.8760		
1	BNGIL5	BCKR06	1	150	6.200	ACSR ZEBRA	2 x 435 mm^2	1620	1000	1000	1250	1250	0.3870	0.2807	4.0203	2.3994	1.7403	24.9259	2.9641	225.0000	0.0132	0.0107	0.0077	0.0056	420.8760		
1	BNGIL5	BCKR06	2	150	6.200	ACSR ZEBRA	2 x 435 mm^2	1620	1000	1000	1250	1250	0.3870	0.2807	4.0203	2.3994	1.7403	24.9259	2.9641	225.0000	0.0132	0.0107	0.0077	0.0056	420.8760		
1	GDTAN6	RJOS06	1	150	10.487	ACSR AW	330 mm^2	740	1000	800	1250	1250	0.1172	0.4003	2.8630	1.2291	4.1979	29.9194	4.3742	225.0000	0.0194	0.0055	0.0187	0.0067	192.2520		
1	GDTAN6	RJOS06	2	150	10.487	ACSR AW	330 mm^2	740	1000	800	1250	1250	0.1172	0.4003	2.8630	1.2291	4.1979	29.9194	4.3742	225.0000	0.0194	0.0055	0.0187	0.0067	192.2520		
1	PIER6	GRATI5	1	150	30.000	TACSR	410 mm^2	1620	1000	1000	1250	2000	0.3870	0.2807	4.0203	11.6100	8.4210	120.6090	14.3424	225.0000	0.0637	0.0516	0.0374	0.0271	420.8760		
1	PBLGO6	GDTAN6	1	150	33.827	ACSR AW	330 mm^2	740	800	1000	1250	1250	0.1172	0.4003	2.8630	3.9645	13.5409	96.5084	14.1094	225.0000	0.0627	0.0176	0.0602	0.0217	192.2520		
1	PBLGO6	GDTAN6	2	150	33.827	ACSR AW	330 mm^2	740	800	1000	1250	1250	0.1172	0.4003	2.8630	3.9645	13.5409	96.5084	14.1094	225.0000	0.0627	0.0176	0.0602	0.0217	192.2520		
1	PBLGO6	LJANG6	1	150	51.692	ACSR AW	330 mm^2	740	800	800	800	800	0.1172	0.4003	2.8630	6.0583	20.6923	147.4773	21.5610	225.0000	0.0958	0.0269	0.0920	0.0332	192.2520		
1	PBLGO6	LJANG6	2	150	51.692	ACSR AW	330 mm^2	740	800	800	800	800	0.1172	0.4003	2.8630	6.0583	20.6923	147.4773	21.5610	225.0000	0.0958	0.0269	0.0920	0.0332	192.2520		
1	KRSAN6	PBLGO6	1	150	30.239	ACSR AW	330 mm^2	740	1000	1000	1250	1250	0.1172	0.4003	2.8630	3.5440	12.1047	86.2719	12.6128	225.0000	0.0561	0.0158	0.0538	0.0194	192.2520		
1	KRSAN6	PBLGO6	2	150	30.239	ACSR AW	330 mm^2	740	1000	1000	1250	1250	0.1172	0.4003	2.8630	3.5440	12.1047	86.2719	12.6128	225.0000	0.0561	0.0158	0.0538	0.0194	192.2520		
1	KRSAN6	GDING6	1	150	19.127	ACSR AW	330 mm^2	740	1000	400	1250	1250	0.1172	0.4003	2.8630	2.2429	7.6600	54.6979	7.9821	225.0000	0.0355	0.0100	0.0340	0.0123	192.2520		
1	PITON6	KRSAN6	1	150	20.204	ACSR AW	330 mm^2	740	1000	1000	1250	1250	0.1172	0.4003	2.8630	2.3679	8.0877	57.6420	8.4272	225.0000	0.0375	0.0105	0.0359	0.0130	192.2520		
1	PITON6	KRSAN6	2	150	20.204	ACSR AW	330 mm^2	740	1000	1000	1250	1250	0.1172	0.4003	2.8630	2.3679	8.0877	57.6420	8.4272	225.0000	0.0375	0.0105	0.0359	0.0130	192.2520		
1	PITON6	STBDO6	1	150	55.433	ACSR ZEBRA	2 x 435 mm^2	1600	2000	2000	1250	1250	0.3870	0.2807	4.0203	21.4526	15.5600	222.8573	26.5016	225.0000	0.1178	0.0953	0.0692	0.0502	415.6800		
1	PITON6	STBDO6	2	150	55.433	ACSR ZEBRA	2 x 435 mm^2	1600	2000	2000	1250	1250	0.3870	0.2807	4.0203	21.4526	15.5600	222.8573	26.5016	225.0000	0.1178	0.0953	0.0692	0.0502	415.6800		
1	BNGIL4	PDAAN4	1	70	9.700	ACSR OSTRICH	300 MCM	440	250	400	800	800	0.2175	0.3871	2.9560	2.1098	3.7549	28.6732	4.3070	49.0000	0.0879	0.0431	0.0766	0.0014	53.3456		
1	BNGIL4	PDAAN4	2	70	9.700	ACSR OSTRICH	300 MCM	440	250	400	800	800	0.2175	0.3871	2.9560	2.1098	3.7549	28.6732	4.3070	49.0000	0.0879	0.0431	0.0766	0.0014	53.3456		
1	BNGIL4	SDRAN4	1	70	11.700	ACSR PIPER	300 MCM	440	400	400	400	400	0.2180	0.3859	2.9650	2.5506	4.5150	34.6905	5.1857	49.0000	0.1058	0.0521	0.0921	0.0017	53.3456		
1	BNGIL4	PRONG4	1	70	11.700	ACSR PIPER	300 MCM	440	400	400	400	400	0.2180	0.3859	2.9650	2.5506	4.5150	34.6905	5.1857	49.0000	0.1058	0.0521	0.0921	0.0017	53.3456		
1	BNGIL4	BLBNG4	1	70	40.000	ACSR PIGEON	3/0 AWG	300	200	200	800	800	0.3866	0.3584	3.2020	15.4640	14.3360	128.0800	21.0869	49.0000	0.4303	0.3156	0.2926	0.0063	36.3720		
1	BNGIL4	BLBNG4	2	70	40.000	ACSR PIGEON	3/0 AWG	300	200	200	400	800	0.3866	0.3584	3.2020	15.4640	14.3360	128.0800	21.0869	49.0000	0.4303	0.3156	0.2926	0.0063	36.3720		
1	GRATI5	GDTAN5	1	150	18.500	TACSR	410 mm^2	1620	1000	1000	1250	2000	0.3870	0.2807	4.0203	7.1595	5.1930	74.3756	8.8446	225.0000	0.0393	0.0318	0.0231	0.0167	420.8760		
1	GRATI5	GDTAN5	2	150	10.743	TACSR	410 mm^2	1620	1000	1000	1250	2000	0.3870	0.2807	4.0203	4.1575	3.0156	43.1901	5.1360	225.0000	0.0228	0.0185	0.0134	0.0097	420.8760		
1	GDTAN5	PIER5	1	150	11.500	TACSR	410 mm^2	1620	1000	1000	1250	2000	0.3870	0.2807	4.0203	4.4505	3.2281	46.2335	5.4979	225.0000	0.0244	0.0198	0.0143	0.0104	420.8760		
1	GDTAN5	PIER5	2	150	11.500	TACSR	410 mm^2	1620	1000	1000	1250	2000	0.3870	0.2807	4.0203	4.4505	3.2281	46.2335	5.4979	225.0000	0.0244	0.0198	0.0143	0.0104	420.8760		

DATA KARAKTERISTIK TRANSMISI UPT & SRB

Jml Sirkit	Dari	Ke	No. Sirkit	Data_Penghantar Teg (kV)	Jarak (Km)	Jenis		I Nom (Amp)	Tap CT Dari	Tap CT Ke	Tap WT Dari	Tap WT Ke	R (Ohm/km)	X (Ohm/km)	B (MyS/km)	R (Ohm)	X (Ohm)	B (Ohm)	Z act (Ohm)	Z base (Ohm)	Z pu (Ohm)	R pu (Ohm)	X pu (Ohm)	B pu (Ohm)	Daya (MVA)	TAHUN OPERASI	Keterangan
				(1)	(2)								(3)	(4)	(5)	(6)=(2)x(3)	(7)=(2)x(4)	(8)=(2)x(5)	(9)=sqr (6)²+(7)²	(10)=(1)²/100	(11)=(9)/(10)	(12)=(6)/(10)	(13)=(7)/(10)	(14)=(8)/kV/0,1975	(15)=(1)x(kA)x1,732/1000		MVA base = 100 150 kV = 0,1975 70 kV = 4,153

(approx.)

7. UPT Jember

Jml Sirkit	Dari	Ke	No. Sirkit	Teg (kV)	Jarak (Km)	Jenis		I Nom (Amp)	Tap CT Dari	Tap CT Ke	Tap WT Dari	Tap WT Ke	R (Ohm/km)	X (Ohm/km)	B (MyS/km)	R (Ohm)	X (Ohm)	B (Ohm)	Z act (Ohm)	Z base (Ohm)	Z pu (Ohm)	R pu (Ohm)	X pu (Ohm)	B pu (Ohm)	Daya (MVA)	TAHUN OPERASI	Keterangan
1	PBLGO5	LJANG5	1	150	51.592	ACSR DOVE	330 mm^2	740	800	800	800	800	0.1172	0.4003	2.8630	6.0683	20.6523	147.4773	21.5610	225.0000	0.0958	0.0269	0.0920	0.0332	192.2520		
1	PBLGO5	LJANG5	2	150	51.592	ACSR DOVE	330 mm^2	740	800	800	800	800	0.1172	0.4003	2.8630	6.0683	20.6523	147.4773	21.5610	225.0000	0.0958	0.0269	0.0920	0.0332	192.2520		
1	LJANG5	TGGUL5	1	150	32.060	ACSR AW	330 mm^2	740	800	800	1250	1250	0.1172	0.4003	2.8630	3.7596	12.8416	91.5242	13.3807	225.0000	0.0595	0.0167	0.0571	0.0206	192.2520		
1	LJANG5	JMBER5	1	150	60.530	ACSR AW	330 mm^2	740	800	800	1250	800	0.1172	0.4003	2.8630	7.0941	24.2302	172.692	25.2473	225.0000	0.1122	0.0315	0.1077	0.0389	192.2520		
1	TNGUL5	JMBER5	1	150	28.450	ACSR AW	330 mm^2	740	1000	800	1250	800	0.1172	0.4003	2.8630	3.3343	11.3888	81.1579	11.8066	225.0000	0.0527	0.0148	0.0506	0.0183	192.2520		
1	JMBER5	BWNGI5	1	150	82.595	ACSR AW	330 mm^2	740	800	800	800	800	0.1172	0.4003	2.8630	9.5801	32.0628	235.5436	34.4507	225.0000	0.1531	0.0430	0.1469	0.0530	192.2520		
1	JMBER5	GTENG5	1	150	64.700	ACSR AW	330 mm^2	740	800	1000	1250	1250	0.1172	0.4003	2.8630	7.5828	25.9026	184.8119	26.9900	225.0000	0.1200	0.0337	0.1151	0.0416	192.2520		
1	BDWSO5	JMBER5	1	150	38.662	ACSR AW	330 mm^2	740	800	800	800	800	0.1172	0.4003	2.8630	4.5484	15.8787	113.1567	16.5432	225.0000	0.0735	0.0207	0.0706	0.0255	192.2520		
1	BDWSO5	JMBER5	2	150	38.662	ACSR AW	330 mm^2	740	800	800	1600	1600	0.1172	0.4003	2.8630	4.5484	15.8787	113.1567	16.5432	225.0000	0.0735	0.0207	0.0706	0.0255	192.2520		
1	STBDO5	PITON5	1	150	55.433	ACSR ZEBRA	2 x 435 mm^2	1670	2000	2000	1250	1250	0.3870	0.2807	4.0200	21.4526	15.5600	222.8573	26.5015	225.0000	0.1178	0.0953	0.0692	0.0502	420.6780		
1	STBDO5	PITON5	2	150	55.433	ACSR ZEBRA	2 x 435 mm^2	1620	2000	2000	1250	1250	0.3870	0.2807	4.0200	21.4526	15.5600	222.8573	26.5015	225.0000	0.1178	0.0953	0.0692	0.0502	420.6780		
1	STBDO5	BDWSO5	1	150	34.786	ACSR AW	330 mm^2	740	800	800	800	800	0.1172	0.4003	2.8530	4.0769	13.9248	99.2445	14.5094	225.0000	0.0645	0.0181	0.0619	0.0223	192.2520		
1	STBDO5	BDWSO5	2	150	34.786	ACSR AW	330 mm^2	740	800	800	1600	1600	0.1172	0.4003	2.8530	4.0769	13.9248	99.2445	14.5094	225.0000	0.0645	0.0181	0.0619	0.0223	192.2520		
1	STBDO5	BWNGI5	1	150	74.200	ACSR DOVE	2 x 340 mm^2	1200	800	800	1600	1600	0.0586	0.2815	4.0740	4.3481	20.8873	302.2908	21.3351	225.0000	0.0948	0.0193	0.0928	0.0680	311.7600		
1	STBDO5	BWNGI5	2	150	74.200	ACSR DOVE	2 x 340 mm^2	1200	800	800	1600	1600	0.0586	0.2815	4.0740	4.3481	20.8873	302.2908	21.3351	225.0000	0.0948	0.0193	0.0928	0.0680	311.7600		
1	GTENG5	BWNGI5	1	150	33.960	ACSR DOVE	330 mm^2	740	1000	800	1250	800	0.1172	0.4003	2.8530	3.9799	13.5902	96.8594	14.1807	225.0000	0.0629	0.0177	0.0604	0.0213	192.2520		
1	BWNGI5	KTPNG5	1	150	7.990	ACSR HAWK	477 MM	566	600	800			0.1370	0.3966	2.8800	1.0946	3.1688	23.0112	3.3526	225.0000	0.0149	0.0049	0.0141	0.0052	170.1880		
	BWNGI5	KTP-GLM	I (1.1)	150	4.471	SUBMARINE CABLE	300 mm^2	500	600	800			0.0870	0.1310	73.0000	0.3890	0.5857	326.3830	0.7031	225.0000	0.0031	0.0017	0.0026	0.0734	129.9000		Kabel laut
	BWNGI5	KTP-GLM	I (lama)	150	4.829	OSS	300 mm^2	465	600	800						-	-					#DIV/0!	#DIV/0!	-	120.8070		Kabel laut rusak
1	BWNGI5	KTPNG5	2	150	7.990	ACSR HAWK	477 MM	566	600	800			0.1370	0.3966	2.8800	1.0946	3.1688	23.0112	3.3526	225.0000	0.0149	0.0049	0.0141	0.0052	170.1880		
	BWNGI5	KTP-GLM	II (1.0)	150	4.389	SUBMARINE CABLE	300 mm^2	500	600	800			0.0870	0.1310	73.0000	0.3818	0.5750	320.3970	0.6902	225.0000	0.0031	0.0017	0.0026	0.0721	129.9000		Op.tgl 03-03-2020
	BWNGI5	KTP-GLM	II (lama)	150	4.260	OSS	300 mm^2	465	600	800						-	-		-			#DIV/0!	#DIV/0!	-	120.8070		Kabel laut rusak

DATA KARAKTERISTIK TRANSMISI UPT & SRB

Jml Sirkit	Dari	Ke	No. Sirkit	Data Penghantar			I Nom	Tap CT		Tap WT		R	X	B	R	X	B	Z set	Z base	Z pu	R pu	X pu	B pu	Daya	TAHUN OPERASI	Keterangan
				Teg (kV)	Jarak (Km)	Jenis	(Amp)	Dari	Ke	Dari	Ke	(Ohm/km)	(Ohm/km)	(MyS/km)	(Ohm)	(Ohm)	(Ohm)	(Ohm)	(Ohm)	(Ohm)	(Ohm)	(Ohm)	(Ohm)	(Ohm)	(MVA)		
				(1)	(2)							(3)	(4)	(5)	(6)=(3)x(2)	(7)=(4)x(4)	(8)=(5)/(2)	(9)=sqrt(6^2+7^2)	(10)=(11)^2/(12)	(11)=(6)/(10)	(12)=(6)/(10)	(13)=(7)/(10)	(14)=(8)/(10)/1000	(15)=(11)x(12)^2/1000		MVA base = 100 150 kV = 0,1875 70 kV = 0,185
Sub Region Bali																										
1	GLNUK5	CHGLNUK	1	150	1.720	ACSR HAWK	477 MCM	645	800	800			0.1370	0.3968	2.5800	0.2356	0.6822	4.4536	0.7217	225.0000	0.0032	0.0010	0.0030	0.0011	167.5713		
1	GLNUK5	CHGLNUK	2	150	1.720	ACSR HAWK	477 MCM	645	800	800			0.1370	0.3968	2.5800	0.2356	0.6822	4.4536	0.7217	225.0000	0.0032	0.0010	0.0030	0.0011	167.5713		
1	CHGLNUK	CHKTPNG	1	150	4.200	OPC	3 x 300 mm2	462	800	800			0.0870	0.1310	73.0000	0.3654	0.5502	306.6000	0.6606	225.0000	0.0029	0.0016	0.0024	0.0490	120.0278		
1	CHGLNUK	CHKTPNG	2	150	4.200	OPC	3 x 300 mm2	462	800	800			0.0870	0.1310	73.0000	0.3654	0.5502	306.6000	0.6606	225.0000	0.0029	0.0016	0.0024	0.0490	120.0278		
1	GLNUK5	NSARA5	1	150	35.040	ACSR HAWK	477 MCM	645	800	800	630	630	0.1370	0.3968	2.5800	5.2115	15.0867	109.5552	16.0614	225.0000	0.0700	0.0232	0.0671	0.0247	167.5713		
1	GLNUK5	NSARA5	2	150	35.040	ACSR HAWK	477 MCM	645	800	800	630	630	0.1370	0.3968	2.5800	5.2115	15.0867	109.5552	16.0614	225.0000	0.0700	0.0232	0.0671	0.0247	167.5713		
1	GLNUK5	PMRON5	1	150	75.600	ACSR HAWK	2 x 477 MCM	1290	800	800	1250	1250	0.0686	0.2045	2.7000	5.1886	16.5219	211.8000	16.3892	225.0000	0.0728	0.0231	0.0690	0.0478	333.1429		
1	NSARA5	NSARA5	1	150	44.400	ACSR HAWK	477 MCM	645	800	800	630	630	0.1370	0.3968	2.5800	6.0826	17.6090	127.8720	18.6300	225.0000	0.0828	0.0270	0.0783	0.0288	167.5713		
1	NSARA5	NSARA5	2	150	44.400	ACSR HAWK	477 MCM	645	800	800	630	1250	0.1370	0.3968	2.5800	6.0826	17.6090	127.8720	18.6300	225.0000	0.0828	0.0270	0.0783	0.0288	167.5713		
1	KAPAL5	KNGRE	1	150	23.310	ACSR HAWK	477 MCM	645	800	800	630	1250	0.1370	0.3968	2.5800	3.1935	9.2447	67.1328	9.7806	225.0000	0.0435	0.0142	0.0411	0.0151	167.5713		
1	KAPAL5	KNGRE	2	150	23.310	ACSR HAWK	477 MCM	645	800	800	630	1250	0.1370	0.3968	2.5800	3.1935	9.2447	67.1328	9.7806	225.0000	0.0435	0.0142	0.0411	0.0151	167.5713		
1	KAPAL5	GNYAR5	1	150	19.210	ACSR HAWK	477 MCM	645	800	800	630	800	0.1370	0.3968	2.5800	2.6318	7.6187	55.3048	8.0604	225.0000	0.0358	0.0117	0.0339	0.0125	167.5713		
1	KAPAL5	GNYAR5	2	150	19.210	ACSR HAWK	477 MCM	645	800	800	630	800	0.1370	0.3968	2.5800	2.6318	7.6187	55.3048	8.0604	225.0000	0.0358	0.0117	0.0339	0.0125	167.5713		
1	KAPAL5	NSDUA5	1	150	20.910	ACSR PARTRIDGE	135 mm2	400	400	400	630	630	0.2139	0.4060	2.7960	4.4684	8.6313	57.8571	9.6297	225.0000	0.0428	0.0190	0.0368	0.0130	103.9259		
1	KAPAL5	NSDUA5	1	150	0.960	CLED XLPE	3 x 240 mm2	400				1250	0.0870	0.1320	73.0000	0.0835	0.1267	70.0800	0.1516	0.0120	15.1766	0.0323	12.5720	384.8354	0.6923		
1	KAPAL5	SRTT5	1	150	36.170	TACSR	160 mm2	400	800	800	1250	1250	0.1370	0.3970	2.8800	5.2298	16.1626	109.9256	16.0304	225.0000	0.0712	0.0232	0.0673	0.0247	103.9259		
1	KAPAL5	PNGAN5	1	150	27.700	TACSR	160 mm2	400	800	800	1250	1250	0.1370	0.3970	2.8800	3.8729	8.8149	82.4960	9.1134	225.0000	0.0405	0.0138	0.0393	0.0141	103.9259		
1	STRT5	PNGAN5	1	150	28.010	TACSR	160 mm2	400	800	800	1250	1250	0.1370	0.3970	2.8800	3.8374	11.1200	80.6688	11.7536	225.0000	0.0523	0.0171	0.0494	0.0182	103.9259		
1	GNYAR5	SANUR5	1	150	16.500	ACSR PARTRIDGE	135 mm2	400	200	1000	800	630	0.2139	0.4060	2.7960	3.5294	6.7320	46.0360	7.5988	225.0000	0.0338	0.0132	0.0298	0.0103	103.9259		
1	GNYAR5	BABLI5	2	150	16.380	ACSR HAWK	477 MCM	645	800	1000	630	630	0.1370	0.3970	2.8800	2.2441	6.6029	47.1744	6.8782	225.0000	0.0306	0.0100	0.0289	0.0106	167.5713		
1	GNYAR5	AMPRA5	1	150	33.780	ACSR HAWK	477 MCM	645	400	1000	630	630	0.1370	0.3968	2.8800	4.6251	13.3892	97.2288	14.1656	225.0000	0.0630	0.0206	0.0595	0.0219	167.5713		
1	GNYAR5	AMPRA5	2	150	33.780	ACSR HAWK	477 MCM	645	1000	1000			0.1370	0.3968	2.8800	4.6251	13.3892	97.2288	14.1656	225.0000	0.0630	0.0206	0.0595	0.0219	167.5713		
1	BANJR5	PGRAN5	1	150	7.740	ACSR PARTRIDGE	135 mm2	400	400	400	630	1250	0.2139	0.4060	2.7960	1.6563	3.1579	21.4085	3.5845	225.0000	0.0159	0.0073	0.0140	0.0049	103.9259		
1	BANJR5	PGRAN5	2	150	7.730	ACSR HAWK	477 MCM	400	1000	1000	630	1250	0.1370	0.3970	2.8800	1.0918	3.0768	22.3200	3.2549	225.0000	0.0145	0.0047	0.0137	0.0050	103.9259		
1	NSDUA5	PGRAN5	1	150	13.410	ACSR PARTRIDGE	135 mm2	400	200	200	630	630	0.2139	0.4060	2.7960	2.8544	5.4713	37.9921	5.7257	225.0000	0.0274	0.0127	0.0243	0.0085	103.9259		
1	NSDUA5	PGRAN5	1	150	1.200	XLPE	3 x 240 mm2	400				1250	0.0870	0.1320	73.0000	0.1044	0.1584	87.6000	0.1897	0.0120	15.8716	10.4400	15.8400	443.5443	0.6923		
1	BTRT5	PMRON5	1	150	19.280	ACSR PARTRIDGE	135 mm2	400	200	200	630	630	0.2139	0.4060	2.7960	4.1262	7.8282	53.9285	8.8590	225.0000	0.0394	0.0183	0.0350	0.0120	103.9259		
1	BTRT5	PMRON5	2	150	19.280	ACSR PARTRIDGE	135 mm2	400	200	200	630	630	0.2139	0.4060	2.7960	4.1262	7.8282	53.9285	8.8590	225.0000	0.0394	0.0183	0.0350	0.0120	103.9259		
1	KAPAL5	PRANG	1	150	5.120	ACSR HAWK	477 MCM	645	400	1200	630	630	0.1370	0.3968	2.8800	1.2494	3.6170	26.2656	3.8267	225.0000	0.0170	0.0054	0.0161	0.0059	167.5713		
1	PGAN5	PGRAN5	1	150	7.600	ACSR HAWK	477 MCM	645	400	1000	630	630	0.1370	0.3968	2.8800	1.0412	3.0142	21.8880	3.1889	225.0000	0.0142	0.0046	0.0134	0.0049	167.5713		

❖ **Listing Program Borland Delphi 7.0**

❖ **Listing Program *Optimal Power Flow* Menggunakan Metode Parallel *Evolutionary Programming*.**

```
unit uMenu;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ComCtrls, StdCtrls, ExtCtrls, Grids;

type
  TfrmMenu = class(TForm)
    Panel1: TPanel;
    btnNew: TButton;
    btnOpen: TButton;
    btnSetting: TButton;
    StatusBar1: TStatusBar;
    Panel2: TPanel;
    OpenDialog1: TOpenDialog;
    btnExit: TButton;
    procedure btnNewClick(Sender: TObject);
    procedure btnOpenClick(Sender: TObject);
    procedure btnExitClick(Sender: TObject);
    procedure btnSettingClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmMenu: TfrmMenu;

implementation

uses uInputLF, uLoadflow, uComplex, uUtils, uSetting;

{$R *.dfm}

procedure TfrmMenu.btnNewClick(Sender: TObject);
begin
  frmInput.Caption:='Input Data';
  frmInput.btnNext.Caption:='&Save';
  frmInput.Show;
end;

procedure TfrmMenu.btnOpenClick(Sender: TObject);
var NamaFile,Nama:string;
    output:TextFile;
    i,j,Typ,dari,ke,Nbus,Nsal,Param,Ngen,NCable:integer;
    Cap,absV,sudV,Pg,Qg,PL,QL,CapSal,Pmin,Pmax,Harga,Length:double;
    R,X,Lc,Tr,Tu,Su,VKonst,PKonst,Pbasc,Vbase:double;
    FileStream:TFileStream;
    str:TStringList;
begin
  try
    if OpenDialog1.Execute then
    begin
      NamaFile:=OpenDialog1.FileName;
      FileStream:=TFileStream.Create(NamaFile,
      fmOpenRead or fmShareDenyWrite);
      str:=TStringList.Create;
      str.LoadFromStream(FileStream);
      frmSetting.ClientSocket1.Socket.SendText(str.Text);
```

```
frmSetting.ClientSocket2.Socket.SendText(str.Text);
frmSetting.ClientSocket3.Socket.SendText(str.Text);
frmSetting.ClientSocket4.Socket.SendText(str.Text);
str.Free;
FileStream.Free;
AssignFile(output,NamaFile);
Reset(output);
Readln(output,Nbus);
Readln(output,Nsal);
Readln(output,Vbase);
Readln(output,VKonst);
Readln(output,Pbase);
Readln(output,PKonst);
Readln(output,param);
gGeneral.Vbase:=Vbase;
gGeneral.VKonst:=VKonst;
gGeneral.Pbase:=Pbase;
gGeneral.PKonst:=PKonst;
if Param=1 then
begin
  gGeneral.Param:=psPu;
end
else if Param=2 then
begin
  gGeneral.Param:=psOhm;
end;
gGeneral.MaxIterasi:=15;
gGeneral.Toleransi:=0.0001;
frmInput.edtNbus.Text:=IntToStr(Nbus);
frmInput.edtNsal.Text:=IntToStr(Nsal);
frmInput.edtVbase.Text:=FloatToStr(Vbase);
if VKonst=1 then
begin
  frmInput.cmbVKonst.Text:='V';
end
else if VKonst=1000 then
begin
  frmInput.cmbVKonst.Text:='kV';
end
else if VKonst=1000000 then
begin
  frmInput.cmbVKonst.Text:='MV';
end;
frmInput.edtPbase.Text:=FloatToStr(Pbase);
if PKonst=1 then
begin
  frmInput.cmbPKonst.Text:='VA';
  frmInput.fgBus.Cells[3,0]:='Pg (W)';
  frmInput.fgBus.Cells[4,0]:='Qg (VAR)';
  frmInput.fgBus.Cells[5,0]:='PL (W)';
  frmInput.fgBus.Cells[6,0]:='QL (VAR)';
  frmInput.fgBranch.Cells[9,0] :='Kap (VA)';
  frmInput.fgBranch.Cells[10,0]:='P (W)';
  frmInput.fgBranch.Cells[11,0]:='Q (VAR)';
  frmInput.fgBranch.Cells[16,0]:='P (W)';
  frmInput.fgBranch.Cells[17,0]:='Q (VAR)';
end
else if PKonst=1000 then
begin
  frmInput.cmbPKonst.Text:='kVA';
  frmInput.fgBus.Cells[3,0]:='Pg (kW)';
  frmInput.fgBus.Cells[4,0]:='Qg (kVAR)';
  frmInput.fgBus.Cells[5,0]:='PL (kW)';
  frmInput.fgBus.Cells[6,0]:='QL (kVAR)';
  frmInput.fgBranch.Cells[9,0] :='Kap (kVA)';
  frmInput.fgBranch.Cells[10,0]:='P (kW)';
  frmInput.fgBranch.Cells[11,0]:='Q (kVAR)';
  frmInput.fgBranch.Cells[16,0]:='P (kW)';
```

```
  frmInput.fgBranch.Cells[17,0]:='Q (kVAR)';
end
else if PKonst=1000000 then
begin
  frmInput.cmbPKonst.Text:='MVA';
  frmInput.fgBus.Cells[3,0]:='Pg (MW)';
  frmInput.fgBus.Cells[4,0]:='Qg (MVAR)';
  frmInput.fgBus.Cells[5,0]:='PL (MW)';
  frmInput.fgBus.Cells[6,0]:='QL (MVAR)';
  frmInput.fgBranch.Cells[9,0] :='Kap (MVA)';
  frmInput.fgBranch.Cells[10,0]:='P (MW)';
  frmInput.fgBranch.Cells[11,0]:='Q (MVAR)';
  frmInput.fgBranch.Cells[16,0]:='P (MW)';
  frmInput.fgBranch.Cells[17,0]:='Q (MVAR)';
end;
if param=1 then
begin
  frmInput.cmbParam.Text:='pu';
  frmInput.fgBranch.Cells[3,0]:='R (pu)';
  frmInput.fgBranch.Cells[4,0]:='X (pu)';
  frmInput.fgBranch.Cells[5,0]:='Lc (pu)';
  frmInput.fgBus.Cells[7,0]:='Cap (pu)';
end
else if param=2 then
begin
  frmInput.cmbParam.Text:='ohm';
  frmInput.fgBranch.Cells[3,0]:='R (ohm)';
  frmInput.fgBranch.Cells[4,0]:='X (ohm)';
  frmInput.fgBranch.Cells[5,0]:='Lc (ohm)';
  frmInput.fgBus.Cells[7,0]:='Cap (ohm)';
end;
SetLength(gBus,Nbus,8);
for i:=0 to Nbus-1 do
begin
  Readln(output,absV,sudV,Pg,Qg,PL,QL,Cap,Typ);
  gBus[i,0]:=absV;
  gBus[i,1]:=sudV;
  gBus[i,2]:=Pg;
  gBus[i,3]:=Qg;
  gBus[i,4]:=PL;
  gBus[i,5]:=QL;
  gBus[i,6]:=Cap;
  gBus[i,7]:=Typ;
  frmInput.fgBus.Cells[0,i+1]:=IntToStr(i+1);
  frmInput.fgBus.Cells[1,i+1]:=FloatToStr(absV);
  frmInput.fgBus.Cells[2,i+1]:=FloatToStr(sudV);
  frmInput.fgBus.Cells[3,i+1]:=FloatToStr(Pg);
  frmInput.fgBus.Cells[4,i+1]:=FloatToStr(Qg);
  frmInput.fgBus.Cells[5,i+1]:=FloatToStr(PL);
  frmInput.fgBus.Cells[6,i+1]:=FloatToStr(QL);
  frmInput.fgBus.Cells[7,i+1]:=FloatToStr(Cap);
  frmInput.fgBus.Cells[8,i+1]:=IntToStr(typ);
end;
SetLength(gBranch,Nsal,17);
SetLength(gLengthBranch,Nsal);
for i:=0 to Nsal-1 do
begin
  Readln(output,dari,ke,R,X,Lc,Tr,Tu,Su,CapSal);
  gBranch[i,0]:=dari;
  gBranch[i,1]:=ke;
  gBranch[i,2]:=R;
  gBranch[i,3]:=X;
  gBranch[i,4]:=Lc;
  gBranch[i,5]:=Tr;
  gBranch[i,6]:=Tu;
  gBranch[i,7]:=Su;
  gBranch[i,8]:=CapSal;
  frmInput.fgBranch.Cells[0,i+1]:=IntToStr(i+1);
```

```
        frmInput.fgBranch.Cells[1,i+1]:=IntToStr(dari);
        frmInput.fgBranch.Cells[2,i+1]:=IntToStr(ke);
        frmInput.fgBranch.Cells[3,i+1]:=FloatToStr(R);
        frmInput.fgBranch.Cells[4,i+1]:=FloatToStr(X);
        frmInput.fgBranch.Cells[5,i+1]:=FloatToStr(Lc);
        frmInput.fgBranch.Cells[6,i+1]:=FloatToStr(Tr);
        frmInput.fgBranch.Cells[7,i+1]:=FloatToStr(Tu);
        frmInput.fgBranch.Cells[8,i+1]:=FloatToStr(Su);
        frmInput.fgBranch.Cells[9,i+1]:=FloatToStr(CapSal);
      end;
      Readln(output,Ngen);
      if Ngen<>0 then
      begin
        frmInput.fgGen.RowCount:=Ngen+1;
        SetLength(gGen,Ngen,10);
        for i:=0 to Ngen-1 do
        begin
          Readln(output,dari,R,X,Lc,Tr,Tu,Su,CapSal,Pmin,Pmax);
          gGen[i,0]:=dari;
          gGen[i,1]:=R;
          gGen[i,2]:=X;
          gGen[i,3]:=Lc;
          gGen[i,4]:=Tr;
          gGen[i,5]:=Tu;
          gGen[i,6]:=Su;
          gGen[i,7]:=CapSal;
          gGen[i,8]:=Pmin;
          gGen[i,9]:=Pmax;
          frmInput.fgGen.Cells[0,i+1]:=IntToStr(i+1);
          frmInput.fgGen.Cells[1,i+1]:=RealToStr(gGen[i,0],0);
          frmInput.fgGen.Cells[2,i+1]:=RealToStr(gGen[i,1],2);
          frmInput.fgGen.Cells[3,i+1]:=RealToStr(gGen[i,2],2);
          frmInput.fgGen.Cells[4,i+1]:=RealToStr(gGen[i,3],5);
          frmInput.fgGen.Cells[5,i+1]:=RealToStr(gGen[i,4],5);
          frmInput.fgGen.Cells[6,i+1]:=RealToStr(gGen[i,5],5);
          frmInput.fgGen.Cells[7,i+1]:=RealToStr(gGen[i,6],2);
          frmInput.fgGen.Cells[8,i+1]:=RealToStr(gGen[i,7],2);
          frmInput.fgGen.Cells[9,i+1]:=RealToStr(gGen[i,8],2);
          frmInput.fgGen.Cells[10,i+1]:=RealToStr(gGen[i,9],2);
        end;
      end
      else
      begin
        frmInput.fgGen.RowCount:=2;
      end;
      CloseFile(output);
      frmInput.Caption:='Tampilan Data';
      frmInput.btnNext.Caption:='&Next';
      frmInput.Show;
    end;
  except
    MessageDlg('File Corrupt atau Error Program!',mtWarning,[mbOK],0);
  end;
end;

procedure TfrmMenu.btnExitClick(Sender: TObject);
begin
  Application.Terminate;
end;

procedure TfrmMenu.btnSettingClick(Sender: TObject);
begin
  frmSetting.Show;
  btnOpen.Enabled:=true;
end;end.
```

```
unit uNewtonPolar;

interface

uses uUtils,uComplex,uLoadflow,uMatrix;

procedure NewtonPolar(var rBus,rBranch:dArr2;
          var rGeneral:TGeneral);

implementation

function MismatchDaya(const rNbus,rNgen:integer;
         const rE,rF,rPg,rQg,rPL,rQL:dArr1;
         const rTyp:iArr1;
         const rG,rB:dArr2):dArr1;
var i,j,Ns,Np,Nq:integer;
    sumP,sumQ:double;
begin
  Ns:=rNbus-1+rNbus-rNgen-1;
  SetLength(result,Ns);
  Np:=-1;
  Nq:=rNbus-2;
  for i:=0 to rNbus-1 do
  begin
    if rTyp[i]<>1 then
    begin
      inc(Np);
      sumP:=0.0;
      for j:=0 to rNbus-1 do
      begin
        //sumP:=sumP+Ui*Uj*(Gij*cos(di-dj)+Bij*sin(di-dj));
        sumP:=sumP+rE[i]*rE[j]*(rG[i,j]*cos(rF[i]-rF[j])+
                                rB[i,j]*sin(rF[i]-rF[j]));
      end;
      result[Np]:=rPg[i]-rPL[i]-sumP;
    end;
    if rTyp[i]=3 then
    begin
      inc(Nq);
      sumQ:=0.0;
      for j:=0 to rNbus-1 do
      begin
        //sumQ:=sumQ+Ui*Uj*(Gij*sin(di-dj)-Bij*cos(di-dj));
        sumQ:=sumQ+rE[i]*rE[j]*(rG[i,j]*sin(rF[i]-rF[j])-
                                rB[i,j]*cos(rF[i]-rF[j]));
      end;
      result[Nq]:=rQg[i]-rQL[i]-sumQ;
    end;
  end;
end;

{function MismatchDayaUpfc(const aNbus,aNgen:integer;
         const aV,aSg,aSL:CArr1;
         const aTyp:iArr1;
         const aY:CArr2;
         const aUpfc:TUpfcArr2):dArr1;
var i,j,Np,Nq:integer;
    Ui,Uj,dij,Tij,O,bij:double;
    Ps,Qs:dArr1;
begin
  result:=MismatchDaya(aNbus,aNgen,aV,aSg,aSL,aTyp,aY);
  SetLength(Ps,aNbus);
  SetLength(Qs,aNbus);
  for i:=0 to aNbus-1 do
  begin
    Ps[i]:=0;
    Qs[i]:=0;
  end;
```

```
  for i:=0 to aNbus-1 do
  begin
    Ui:=aV[i].Real;
    for j:=0 to aNbus-1 do
    begin
      if aUpfc[i,j].tap<>0 then
      begin
        Uj:=aV[j].Real;
        dij:=aV[i].Imag-aV[j].Imag;
        Tij:=aUpfc[i,j].tap;
        O:=aUpfc[i,j].sudut;
        bij:=-aY[i,j].Imag;
        Ps[i]:=Ps[i]+(bij*Ui*Uj*sin(dij+O)-sin(dij));
        Qs[i]:=Qs[i]+(bij*(sqr(Ui)*(sqr(Tij)-1)-Ui*Uj*(Tij*cos(dij+O)-
               cos(dij))));
        Ps[j]:=Ps[j]+(-bij*Ui*Uj*(Tij*sin(dij+O)-sin(dij)));
        Qs[j]:=Qs[j]+(-bij*Ui*Uj*(Tij*cos(dij+O)-cos(dij)));
      end;
    end;
  end;
  Np:=-1;
  Nq:=aNbus-2;
  for i:=0 to aNbus-1 do
  begin
    if aTyp[i]<>1 then
    begin
      inc(Np);
      result[Np]:=result[Np]+Ps[i];
    end;
    if aTyp[i]=3 then
    begin
      inc(Nq);
      result[Nq]:=result[Nq]+Qs[i];
    end;
  end;
end;}

function Jaqobian(const rNbus,rNgen:integer;
         const rE,rF:dArr1;
         const rTyp:iArr1;
         const rG,rB:dArr2):dArr2;
var i,j,k,row,col:integer;
    sum,Pj,Qj:double;
begin
  row:=rNbus-1+rNbus-rNgen-1;
  SetLength(result,row,row);
  //Pembentukan Jaqobian H dP/d0
  row:=-1;
  for i:=0 to rNbus-1 do
  begin
    if rTyp[i]<>1 then
    begin
      inc(row);
      col:=-1;
      for j:=0 to rNbus-1 do
      begin
        if rTyp[j]<>1 then
        begin
          inc(col);
          if j=i then
          begin
            sum:=0.0;
            for k:=0 to rNbus-1 do
            begin
              //sum:=sum+((Gjk*sin(dj-dk)-Bjk*cos(dj-dk))*Uk);
              sum:=sum+((rG[j,k]*sin(rF[j]-rF[k])-
                         rB[j,k]*cos(rF[j]-rF[k]))*rE[k]);
            end;
```

```
          //Qj:=sum*Uj;
          Qj:=sum*rE[j];
          //result[row,col]:=-Qj-Bij*sqr(Ui);
          result[row,col]:=-Qj-rB[i,j]*sqr(rE[i]);
        end
        else
        begin
          //result[row,col]:=Ui*Uj*(Gij*sin(di-dj)-Bij*cos(di-dj));
          result[row,col]:=rE[i]*rE[j]*(rG[i,j]*sin(rF[i]-rF[j])-
                                        rB[i,j]*cos(rF[i]-rF[j]));
        end;
      end;
    end;
  end;
end;
//Pembentukan Matrik N dP/dV
row:=-1;
for i:=0 to rNbus-1 do
begin
  if rTyp[i]<>1 then
  begin
    inc(row);
    col:=rNbus-2;
    for j:=0 to rNbus-1 do
    begin
      if rTyp[j]=3 then
      begin
        inc(col);
        if j=i then
        begin
          sum:=0.0;
          for k:=0 to rNbus-1 do
          begin
            //sum:=sum+((Gjk*cos(dj-dk)+Bjk*sin(dj-dk))*Uk);
            sum:=sum+((rG[j,k]*cos(rF[j]-rF[k])+
                       rB[j,k]*sin(rF[j]-rF[k]))*rE[k]);
          end;
          //Pj:=sum*Uj;
          Pj:=sum*rE[j];
          //result[row,col]:=Pj+Gij*Ui;
          result[row,col]:=Pj+rG[i,j]*rE[i];
        end
        else
        begin
          //result[row,col]:=Uj*(Gij*cos(di-dj)+Bij*sin(di-dj));
          result[row,col]:=rE[j]*(rG[i,j]*cos(rF[i]-rF[j])+
                                  rB[i,j]*sin(rF[i]-rF[j]));
        end;
      end;
    end;
  end;
end;
//Pembentukan Jaqobian M dQ/d0
row:=rNbus-2;
for i:=0 to rNbus-1 do
begin
  if rTyp[i]-3 then
  begin
    inc(row);
    col:=-1;
    for j:=0 to rNbus-1 do
    begin
      if rTyp[j]<>1 then
      begin
        inc(col);
        if j=i then
        begin
          sum:=0;
```

```
            for k:=0 to rNbus-1 do
            begin
              //sum:=sum+((Gjk*cos(dj-dk)+Bjk*sin(dj-dk))*Uk);
              sum:=sum+((rG[j,k]*cos(rF[j]-rF[k])+
                        rB[j,k]*sin(rF[j]-rF[k]))*rE[k]);
            end;
            //Pj:=sum*Ui;
            Pj:=sum*rE[i];
            //result[row,col]:=Pj-Gij*sqr(Ui);
            result[row,col]:=Pj-rG[i,j]*sqr(rE[i]);
          end
          else
          begin
            //result[row,col]:=-Ui*Uj*(Gij*cos(di-dj)+Bij*sin(di-dj));
            result[row,col]:=-rE[i]*rE[j]*(rG[i,j]*cos(rF[i]-rF[j])+
                                           rB[i,j]*sin(rF[i]-rF[j]));
          end;
        end;
      end;
    end;
  end;
  //Pembentukan Jaqobian L dQ/dV
  row:=rNbus-2;
  for i:=0 to rNbus-1 do
  begin
    if rTyp[i]=3 then
    begin
      inc(row);
      col:=rNbus-2;
      for j:=0 to rNbus-1 do
      begin
        if rTyp[j]=3 then
        begin
          inc(col);
          if j=i then
          begin
            sum:=0.0;
            for k:=0 to rNbus-1 do
            begin
              //sum:=sum+((Gjk*sin(dj-dk)-Bjk*cos(dj-dk))*Uk);
              sum:=sum+((rG[j,k]*sin(rF[j]-rF[k])-
                        rB[j,k]*cos(rF[j]-rF[k]))*rE[k]);
            end;
            //Qj:=sum*Ui;
            Qj:=sum*rE[i];
            //result[row,col]:=Qj-Bij*Ui;
            result[row,col]:=Qj-rB[i,j]*rE[i];
          end
          else
          begin
            //result[row,col]:=Uj*(Gij*sin(di-dj)-Bij*cos(di-dj));
            result[row,col]:=rE[j]*(rG[i,j]*sin(rF[i]-rF[j])-
                                    rB[i,j]*cos(rF[i]-rF[j]));
          end;
        end;
      end;
    end;
  end;
end;

{function JaqobianUpfc(const aNbus,aNgen:integer;
         const aV:CArr1;
         const aTyp:iArr1;
         const aY:CArr2;
         const aUpfc:TUpfcArr2):dArr2;
var i,j,col,row:integer;
    bij,Ui,Uj,dij,Tij,O,uJaq:double;
begin
```

```
result:=Jaqobian(aNbus,aNgen,aV,aTyp,aY);
//update jaqobian H (dP/dsudV)
row:=-1;
for i:=0 to aNbus-1 do
begin
  col:=-1;
  if aTyp[i]<>1 then
  begin
    inc(row);
    Ui:=aV[i].Real;
    for j:=0 to aNbus-1 do
    begin
      if aTyp[j]<>1 then
      begin
        inc(col);
        if aUpfc[i,j].tap<>0 then
        begin
          bij:=-aY[i,j].Imag;
          Tij:=aUpfc[i,j].tap;
          O:=aUpfc[i,j].sudut;
          Uj:=aV[j].Real;
          dij:=aV[i].Imag-aV[j].Imag;
          //dPis/di
          uJaq:=-bij*Ui*Uj*(Tij*cos(dij+O)-cos(dij));
          result[row,row]:=result[row,row]+uJaq;
          //dPis/dj
          uJaq:=-bij*Ui*Uj*(-Tij*cos(dij+O)+cos(dij));
          result[row,col]:=result[row,col]+uJaq;
          //dPjs/di
          uJaq:=bij*Ui*Uj*(Tij*cos(dij+O)-cos(dij));
          result[col,row]:=result[col,row]+uJaq;
          //dPjs/dj
          uJaq:=bij*Ui*Uj*(-Tij*cos(dij+O)+cos(dij));
          result[col,col]:=result[col,col]+uJaq;
        end;
      end;
    end;
  end;
end;
//update jaqobian L (dP/dV)
row:=-1;
for i:=0 to aNbus-1 do
begin
  col:=aNbus-2;
  if aTyp[i]<>1 then
  begin
    inc(row);
    Ui:=aV[i].Real;
    for j:=0 to aNbus-1 do
    begin
      if aTyp[j]=3 then
      begin
        inc(col);
        if aUpfc[i,j].tap<>0 then
        begin
          bij:=-aY[i,j].Imag;
          Tij:=aUpfc[i,j].tap;
          O:=aUpfc[i,j].sudut;
          Uj:=aV[j].Real;
          dij:=aV[i].Imag-aV[j].Imag;
          //dPis/dVi
          uJaq:=-bij*Uj*(Tij*sin(dij+O)-sin(dij));
          result[row,row]:=result[row,row]+uJaq;
          //dPis/dVj
          uJaq:=-bij*Ui*(Tij*sin(dij+O)-sin(dij));
          result[row,col]:=result[row,col]+uJaq;
          //dPjs/dVi
          uJaq:=bij*Uj*(Tij*sin(dij+O)-sin(dij));
```

```
            result[col,row]:=result[col,row]+uJaq;
            //dPjs/dVj
            uJaq:=bij*Ui*(Tij*sin(dij+O)-sin(dij));
            result[col,col]:=result[col,col]+uJaq;
          end;
        end;
      end;
    end;
  end;
//update jaqobian M (dQ/dsudV)
row:=aNbus-2;
for i:=0 to aNbus-1 do
begin
  col:=-1;
  if aTyp[i]=3 then
  begin
    inc(row);
    Ui:=aV[i].Real;
    for j:=0 to aNbus-1 do
    begin
      if aTyp[j]<>1 then
      begin
        inc(col);
        if aUpfc[i,j].tap<>0 then
        begin
          bij:=-aY[i,j].Imag;
          Tij:=aUpfc[i,j].tap;
          O:=aUpfc[i,j].sudut;
          Uj:=aV[j].Real;
          dij:=aV[i].Imag-aV[j].Imag;
          //dQis/ddi
          uJaq:=bij*Ui*Uj*(-Tij*sin(dij+O)+sin(dij));
          result[row,row]:=result[row,row]+uJaq;
          //dQis/ddj
          uJaq:=bij*Ui*Uj*(Tij*sin(dij+O)-sin(dij));
          result[row,col]:=result[row,col]+uJaq;
          //dQjs/ddi
          uJaq:=bij*Ui*Uj*Tij*(-sin(dij+O)+sin(dij));
          result[col,row]:=result[col,row]+uJaq;
          //dQjs/ddj
          uJaq:=bij*Ui*Uj*Tij*(sin(dij+O)-sin(dij));
          result[col,col]:=result[col,col]+uJaq;
        end;
      end;
    end;
  end;
end;
//update jaqobian N (dQ/dV)
row:=aNbus-2;
for i:=0 to aNbus-1 do
begin
  col:=aNbus-2;
  if aTyp[i]=3 then
  begin
    inc(row);
    Ui:=aV[i].Real;
    for j:=0 to aNbus-1 do
    begin
      if aTyp[j]=3 then
      begin
        inc(col);
        if aUpfc[i,j].tap<>0 then
        begin
          bij:=-aY[i,j].Imag;
          Tij:=aUpfc[i,j].tap;
          O:=aUpfc[i,j].sudut;
          Oj:=aV[j].Real;
          dij:=aV[i].Imag-aV[j].Imag;
```

```
            //dQis/dVi
            uJaq:=-bij*(2*Ui*(sqr(Tij)-1)-Uj*(Tij*cos(dij+O)-cos(dij)));
            result[row,row]:=result[row,row]+uJaq;
            //dQis/dVj
            uJaq:=bij*Ui*(Tij*cos(dij+O)-cos(dij));
            result[row,col]:=result[row,col]+uJaq;
            //dQjs/dVi
            uJaq:=bij*Uj*Tij*(cos(dij+O)-cos(dij));
            result[col,row]:=result[col,row]+uJaq;
            //dQjs/dVj
            uJaq:=bij*Ui*Tij*(cos(dij+O)-cos(dij));
            result[col,col]:=result[col,col]+uJaq;
          end;
        end;
      end;
    end;
  end;
  //
end;}

procedure UpdateTegangan(const rNbus:integer;
          const rdS:dArr1;
          const rJaq:dArr2;
          const rTyp:iArr1;
          var rE,rF:dArr1);
var i,Np,Nq:integer;
    YE:dArr1;
begin
  YE:=EllGauss(rJaq,rdS);
  Np:=-1;
  Nq:=rNbus-2;
  for i:=0 to rNbus-1 do
  begin
    if rTyp[i]<>1 then
    begin
      inc(Np);
      rF[i]:=rF[i]+YE[Np];
    end;
    if rTyp[i]=3 then
    begin
      inc(Nq);
      rE[i]:=rE[i]+YE[Nq];
    end;
  end;
end;

//main program
procedure NewtonPolar(var rBus,rBranch:dArr2;
          var rGeneral:TGeneral);
var Nbus,Nsal,Ngen,i:integer;
    max:double;
    Typ:iArr1;
    Cap,dS,E,F,Ep,Fp,Pg,Qg,PL,QL:dArr1;
    Lc,Tr,mJaq,R,X,G,B,AlirRe,AlirIm,ArusRe,ArusIm,Tu,Su:dArr2;
begin
  DecodeData(rBus,rBranch,rGeneral,Nbus,Nsal,E,F,Pg,Qg,PL,QL,
  Cap,Typ,R,X,Lc,Tr,Tu,Su);
  Ngen:=FindSumGen(Nbus,Typ);
  Admitansi(Nbus,R,X,Lc,Tr,Tu,Su,Cap,G,B);
  rGeneral.Iterasi:=0;
  for i:=1 to rGeneral.MaxIterasi do
  begin
    dS:=MismatchDaya(Nbus,Ngen,E,F,Pg,Qg,PL,QL,Typ,G,B);
    max:=MaxArray(dS);
    if max<=rGeneral.Toleransi then break;
    inc(rGeneral.Iterasi);
    mJaq:=Jaqobian(Nbus,Ngen,E,F,Typ,G,B);
    UpdateTegangan(Nbus,dS,mJaq,Typ,E,F);
```

```
    end;
    PolarToRec(E,F,Ep,Fp);
    AliranDaya(Nbus,Ep,Fp,G,B,Lc,AlirRe,AlirIm);
    ArusBranch(Nbus,Ep,Fp,Lc,G,B,ArusRe,ArusIm);
    DayaGen(Nbus,Ep,Fp,PL,QL,G,B,Typ,Qg);
    DayaSlack(Nbus,AlirRe,AlirIm,Typ,PL,QL,Pg,Qg);
    Summary(Nbus,Pg,Qg,PL,QL,rGeneral.sumGen,rGeneral.sumLoad,
    rGeneral.sumLoss);
    rGeneral.sumGen:=rGeneral.sumGen.Multiply(rGeneral.Pbase);
    rGeneral.sumLoad:=rGeneral.sumLoad.Multiply(rGeneral.Pbase);
    rGeneral.sumLoss:=rGeneral.sumLoss.Multiply(rGeneral.Pbase);
    UpdateBusGen(Nbus,E,F,Pg,Qg,rGeneral,rBus);
    UpdateBranch(Nsal,AlirRe,AlirIm,ArusRe,ArusIm,rGeneral,rBranch);
  end;

  end.
```

```
unit uObjFunc;

interface

uses uUtils,uNewtonPolar,uLoadflow;

type
  TObjFunc=class
  private
    FLamdaV,FLamdaQ,FLamdaS:double;
    FBatasV:TBatas;
    function getBatasV:TBatas;
    procedure setBatasV(const rBatasV:TBatas);
    function DecodeChrom(const rChrom:dArr1):dArr2;
    function FindPinV(const rLBus:dArr2):integer;
    function FindPinKapBranch(const rLBranch:dArr2):double;
    function FindPinKapQgen(const rLBus:dArr2):double;
  public
    constructor Create(const rLamdaV,rLamdaQ,rLamdaS:double;
                const rBatasV:TBatas);
    function getPgen(const rLbus:dArr2):dArr1;
    function getCostPgen(const rPgen:dArr1):double;
    function FindLength:integer;
    function FindBatasChrom(var rBatasV:TBatas):TBatasArr1;
    function doHitung(const rChrom:dArr1):double;
    function DecodeChromAkhir(const rChrom:dArr1):dArr2;
    procedure doHitungAkhir(const rChrom:dArr1;
              var rLBus,rLBranch:dArr2;
              var rGeneral:TGeneral;
              var rCost:double);
    property LamdaV:double read FLamdaV write FLamdaV;
    property LamdaQ:double read FLamdaQ write FLamdaQ;
    property LamdaS:double read FLamdaS write FLamdaS;
    property BatasV:TBatas read getBatasV write setBatasV;
  end;

var gObjFunc:TObjFunc;

implementation

//constructor
constructor TObjFunc.Create(const rLamdaV,rLamdaQ,rLamdaS:double;
            const rBatasV:TBatas);
begin
  inherited Create;
  FLamdaV:=rLamdaV;
  FLamdaQ:=rLamdaQ;
  FLamdaS:=rLamdaS;
  FBatasV.min:=rBatasV.min;
  FBatasV.max:=rBatasV.max;
end;

//data accessing
function TObjFunc.getBatasV:TBatas;
begin
  result.min:=FBatasV.min;
  result.max:=FBatasV.max;
end;

procedure TObjFunc.setBatasV(const rBatasV:TBatas);
begin
  FBatasV.min:=rBatasV.min;
  FBatasV.max:=rBatasV.max;
end;

//data proccessing
function TObjFunc.FindLength:integer;
var i:integer;
```

```
begin
  result:=0;
  for i:=0 to high(gBus) do
  begin
    if gBus[i,7]=1 then
    begin
      inc(result);
    end
    else if gBus[i,7]=2 then
    begin
      inc(result);
      inc(result);
    end;
  end;
end;

function TObjFunc.FindBatasChrom(var rBatasV:TBatas):TBatasArr1;
var i,sa,Ngen:integer;
begin
  sa:=FindLength;
  SetLength(result,sa);
  sa:=0;
  Ngen:=0;
  for i:=0 to high(gBus) do
  begin
    if gBus[i,7]=1 then
    begin
      result[sa].min:=rBatasV.min;
      result[sa].max:=rBatasV.max;
      inc(sa);
      inc(Ngen);
    end
    else if gBus[i,7]=2 then
    begin
      result[sa].min:=rBatasV.min;
      result[sa].max:=rBatasV.max;
      inc(sa);
      result[sa].min:=gGen[Ngen,8];
      result[sa].max:=gGen[Ngen,9];
      inc(sa);
      inc(Ngen);
    end;
  end;
end;

function TObjFunc.DecodeChrom(const rChrom:dArr1):dArr2;
var i,j,sa:integer;
begin
  SetLength(result,high(gBus)+1,high(gBus[0])+1);
  for i:=0 to high(gBus) do
  begin
    for j:=0 to high(gBus[0]) do
    begin
      result[i,j]:=gBus[i,j];
    end;
  end;
  sa:=0;
  for i:=0 to high(gBus) do
  begin
    if gBus[i,7]=1 then
    begin
      result[i,0]:=rChrom[sa];
      inc(sa);
    end
    else if gBus[i,7]=2 then
    begin
      result[i,0]:=rChrom[sa];
      inc(sa);
```

```
      result[i,2]:=rChrom[sa];
      inc(sa);
    end;
  end;
end;

function TObjFunc.FindPinV(const rLBus:dArr2):integer;
var i:integer;
begin
  result:=0;
  for i:=0 to high(rLBus) do
  begin
    if rLBus[i,0]>FBatasV.max then
    begin
      inc(result);
    end;
    if rLBus[i,0]<FBatasV.min then
    begin
      inc(result);
    end;
  end;
end;

function TObjFunc.FindPinKapBranch(const rLBranch:dArr2):double;
var i:integer;
    absAlir:double;
begin
  result:=0;
  for i:=0 to high(rLBranch) do
  begin
    if rLBranch[i,9]>0 then
    begin
      absAlir:=sqrt(sqr(rLBranch[i,9])+sqr(rLBranch[i,10]));
      if absAlir>rLBranch[i,8] then
      begin
        result:=result+abs(absAlir-rLBranch[i,8]);
      end;
    end
    else
    begin
      absAlir:=sqrt(sqr(rLBranch[i,11])+sqr(rLBranch[i,12]));
      if absAlir>rLBranch[i,8] then
      begin
        result:=result+abs(absAlir-rLBranch[i,8]);
      end;
    end;
  end;
end;

function TObjFunc.FindPinKapQgen(const rLBus:dArr2):double;
var i,sa:integer;
begin
  result:=0;
  sa:=0;
  for i:=0 to high(rLBus) do
  begin
    if rLBus[i,7]<>3 then
    begin
      if rLBus[i,3]>qGen[sa,2] then
      begin
        result:=result+rLBus[i,3]-gGen[sa,2];
      end;
      if rLBus[i,3]<gGen[sa,1] then
      begin
        result:=result+abs(gGen[sa,1])-abs(rLBus[sa,3]);
      end;
      inc(sa);
    end;
```

```
  end;
end;

function TObjFunc.getPgen(const rLbus:dArr2):dArr1;
var i,Ngen,Nbus:integer;
begin
  Nbus:=high(rLbus)+1;
  Ngen:=0;
  for i:=0 to Nbus-1 do
  begin
    if rLbus[i,7]<>3 then
    begin
      inc(Ngen);
    end;
  end;
  SetLength(result,Ngen);
  Ngen:=0;
  for i:=0 to Nbus-1 do
  begin
    if rLbus[i,7]<>3 then
    begin
      result[Ngen]:=rLbus[i,2];
      inc(Ngen);
    end;
  end;
end;

function TObjfunc.getCostPgen(const rPgen:dArr1):double;
var i,Ngen:integer;
begin
  Ngen:=high(rPgen)+1;
  result:=0;
  for i:=0 to Ngen-1 do
  begin
    result:=result+getCostGen(i,gGen,rPgen[i]);
  end;
end;

function TObjFunc.doHitung(const rChrom:dArr1):double;
var LBus:dArr2;
    Pgen:dArr1;
    pinV:integer;
    pinS,Cost,PinQ:double;
begin
  LBus:=DecodeChrom(rChrom);
  NewtonPolar(LBus,gBranch,gGeneral);
  Pgen:=getPgen(LBus);
  Cost:=getCostPgen(Pgen);
  pinV:=FindPinV(LBus);
  pinS:=FindPinKapBranch(gBranch);
  pinQ:=FindPinKapQgen(LBus);
  result:=Cost+FLamdaV*pinV+FLamdaS*pinS+FLamdaS*pinQ;
end;

function TObjFunc.DecodeChromAkhir(const rChrom:dArr1):dArr2;
var i,sa,ne:integer;
begin
  ne:=0;
  for i:=0 to high(gBus) do
  begin
    if gBus[i,7]<>3 then
    begin
      inc(ne);
    end;
  end;
  SetLength(result,ne,3);
  sa:=0;
  ne:=0;
```

```
  for i:=0 to high(gBus) do
  begin
    if gBus[i,7]=1 then
    begin
      result[ne,0]:=i;
      result[ne,1]:=rChrom[sa];
      inc(sa);
      inc(ne);
    end
    else if gBus[i,7]=2 then
    begin
      result[ne,0]:=i;
      result[ne,1]:=rChrom[sa];
      inc(sa);
      result[ne,2]:=rChrom[sa];
      inc(sa);
      inc(ne);
    end;
  end;
end;

procedure TObjFunc.doHitungAkhir(const rChrom:dArr1;
          var rLBus,rLBranch:dArr2;
          var rGeneral:TGeneral;
          var rCost:double);
var i,j:integer;
    Pgen:dArr1;
begin
  rLBus:=DecodeChrom(rChrom);
  SetLength(rLBranch,high(gBranch)+1,high(gBranch[0])+1);
  for i:=0 to high(gBranch) do
  begin
    for j:=0 to high(gBranch[0]) do
    begin
      rLBranch[i,j]:=gBranch[i,j];
    end;
  end;
  rGeneral.MaxIterasi:=gGeneral.MaxIterasi;
  rGeneral.Vbase:=gGeneral.Vbase;
  rGeneral.VKonst:=gGeneral.VKonst;
  rGeneral.Pbase:=gGeneral.Pbase;
  rGeneral.PKonst:=gGeneral.PKonst;
  rGeneral.Toleransi:=gGeneral.Toleransi;
  rGeneral.Param:=gGeneral.Param;
  NewtonPolar(rLBus,rLBranch,rGeneral);
  Pgen:=getPgen(rLBus);
  rCost:=getCostPgen(Pgen);
end;

end.
```

```
    fgHasil: TStringGrid;
    GroupBox5: TGroupBox;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    lblLoss1: TLabel;
    lblLoss2: TLabel;
    lblLoss3: TLabel;
    edtSebelumOpt: TEdit;
    edtSesudahOpt: TEdit;
    edtSelisihOpt: TEdit;
    GroupBox6: TGroupBox;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    lblGenA: TLabel;
    lblLoadA: TLabel;
    lblLossP: TLabel;
    edtSumGenA: TEdit;
    edtSumLoadA: TEdit;
    edtSumLossA: TEdit;
    edtIterasiA: TEdit;
    edtTimeA: TEdit;
    fgBranchA: TStringGrid;
    fgBusA: TStringGrid;
    Label7: TLabel;
    lbxPesan1: TListBox;
    Label13: TLabel;
    lbxPesan2: TListBox;
    Label14: TLabel;
    lbxPesan3: TListBox;
    Label22: TLabel;
    lbxPesan4: TListBox;
    Panel1: TPanel;
    pbIterasi: TProgressBar;
    btnHitung: TButton;
    btnClose: TButton;
    cmbMataUang: TComboBox;
    Label23: TLabel;
    procedure btnCloseClick(Sender: TObject);
    procedure btnUseDefaultClick(Sender: TObject);
    procedure btnHitungClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure btnClientClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmParamTS: TfrmParamTS;

implementation

uses uUtils, uTabuSearch, uFitness, uLoadflow, uNewtonPolar, uEvoPro,
  uObjFunc, uSetting;

{$R *.dfm}

procedure TfrmParamTS.btnCloseClick(Sender: TObject);
begin
  Close;
end;

procedure TfrmParamTS.btnUseDefaultClick(Sender: TObject);
```

```
begin
  edtMaxGen.Text:='200';
  edtPopSize.Text:='40';
  edtKa.Text:='10000000';
  edtBetha.Text:='0.2';
  edtLamdaV.Text:='1000000';
  edtLamdaS.Text:='100000';
  edtLamdaQ.Text:='100000';
  edtVmin.Text:='0.95';
  edtVmax.Text:='1.05';
end;

procedure TfrmParamTS.btnHitungClick(Sender: TObject);
var sa,i,dari,ker,MaxGen,PopSize,Length:integer;
    Pgen:dArr1;
    LamdaV,LamdaS,LamdaQ,Betha,Ka,CostPLN,CostOpt:double;
    BatasV:TBatas;
    Batas:TBatasArr1;
    ep:TEvoPro1;
    BestChrom,Min,Avg,Max:dArr1;
    LBus,LBranch,LSetting:dArr2;
    LGeneral:TGeneral;
    mulai,selesai,selang:TDateTime;
    jam,menit,detik,mdetik:word;
begin
  //
  lbxPesan1.Items.Clear;
  fgBusA.Cells[0,0]:='Bus';
  fgBusA.Cells[1,0]:='absV (pu)';
  fgBusA.Cells[2,0]:='sudV (deg)';
  fgBusA.Cells[7,0]:='Sups (pu)';
  fgBusA.Cells[8,0]:='Type Bus';
  fgBranchA.Cells[0,0]:='No';
  fgBranchA.Cells[1,0]:='Dari';
  fgBranchA.Cells[2,0]:='Ke';
  fgBranchA.Cells[5,0]:='Arus re (A)';
  fgBranchA.Cells[6,0]:='Arus im (A)';
  fgBranchA.Cells[7,0]:='Dari';
  fgBranchA.Cells[8,0]:='Ke';
  fgBranchA.Cells[11,0]:='Arus re (A)';
  fgBranchA.Cells[12,0]:='Arus im (A)';
  fgBus.Cells[0,0]:='Bus';
  fgBus.Cells[1,0]:='absV (pu)';
  fgBus.Cells[2,0]:='sudV (deg)';
  fgBus.Cells[7,0]:='Sups (pu)';
  fgBus.Cells[8,0]:='Type Bus';
  fgBranch.Cells[0,0]:='No';
  fgBranch.Cells[1,0]:='Dari';
  fgBranch.Cells[2,0]:='Ke';
  fgBranch.Cells[5,0]:='Arus re (A)';
  fgBranch.Cells[6,0]:='Arus im (A)';
  fgBranch.Cells[7,0]:='Dari';
  fgBranch.Cells[8,0]:='Ke';
  fgBranch.Cells[11,0]:='Arus re (A)';
  fgBranch.Cells[12,0]:='Arus im (A)';
  if gGeneral.PKonst=1 then
  begin
    fgBusA.Cells[3,0]:='Pg (W)';
    fgBusA.Cells[4,0]:='Qg (VAR)';
    fgBusA.Cells[5,0]:='PL (W)';
    fgBusA.Cells[6,0]:='QL (VAR)';
    fgBranchA.Cells[3,0]:='P (W)';
    fgBranchA.Cells[4,0]:='Q (VAR)';
    fgBranchA.Cells[9,0]:='P (W)';
    fgBranchA.Cells[10,0]:='Q (VAR)';
    lblGenA.Caption:='VA';
    lblLoadA.Caption:='VA';
    lblLossA.Caption:='VA';
```

```
  fgBus.Cells[3,0]:='Pg (W)';
  fgBus.Cells[4,0]:='Qg (VAR)';
  fgBus.Cells[5,0]:='PL (W)';
  fgBus.Cells[6,0]:='QL (VAR)';
  fgBranch.Cells[3,0]:='P (W)';
  fgBranch.Cells[4,0]:='Q (VAR)';
  fgBranch.Cells[9,0]:='P (W)';
  fgBranch.Cells[10,0]:='Q (VAR)';
  lblGen.Caption:='VA';
  lblLoad.Caption:='VA';
  lblLoss.Caption:='VA';
end
else if gGeneral.PKonst=1000 then
begin
  fgBusA.Cells[3,0]:='Pg (kW)';
  fgBusA.Cells[4,0]:='Qg (kVAR)';
  fgBusA.Cells[5,0]:='PL (kW)';
  fgBusA.Cells[6,0]:='QL (kVAR)';
  fgBranchA.Cells[3,0]:='P (kW)';
  fgBranchA.Cells[4,0]:='Q (kVAR)';
  fgBranchA.Cells[9,0]:='P (kW)';
  fgBranchA.Cells[10,0]:='Q (kVAR)';
  lblGenA.Caption:='kVA';
  lblLoadA.Caption:='kVA';
  lblLossA.Caption:='kVA';
  fgBus.Cells[3,0]:='Pg (kW)';
  fgBus.Cells[4,0]:='Qg (kVAR)';
  fgBus.Cells[5,0]:='PL (kW)';
  fgBus.Cells[6,0]:='QL (kVAR)';
  fgBranch.Cells[3,0]:='P (kW)';
  fgBranch.Cells[4,0]:='Q (kVAR)';
  fgBranch.Cells[9,0]:='P (kW)';
  fgBranch.Cells[10,0]:='Q (kVAR)';
  lblGen.Caption:='kVA';
  lblLoad.Caption:='kVA';
  lblLoss.Caption:='kVA';
end
else if gGeneral.PKonst=1000000 then
begin
  fgBusA.Cells[3,0]:='Pg (MW)';
  fgBusA.Cells[4,0]:='Qg (MVAR)';
  fgBusA.Cells[5,0]:='PL (MW)';
  fgBusA.Cells[6,0]:='QL (MVAR)';
  fgBranchA.Cells[3,0]:='P (MW)';
  fgBranchA.Cells[4,0]:='Q (MVAR)';
  fgBranchA.Cells[9,0]:='P (MW)';
  fgBranchA.Cells[10,0]:='Q (MVAR)';
  lblGenA.Caption:='MVA';
  lblLoadA.Caption:='MVA';
  lblLossA.Caption:='MVA';
  fgBus.Cells[3,0]:='Pg (MW)';
  fgBus.Cells[4,0]:='Qg (MVAR)';
  fgBus.Cells[5,0]:='PL (MW)';
  fgBus.Cells[6,0]:='QL (MVAR)';
  fgBranch.Cells[3,0]:='P (MW)';
  fgBranch.Cells[4,0]:='Q (MVAR)';
  fgBranch.Cells[9,0]:='P (MW)';
  fgBranch.Cells[10,0]:='Q (MVAR)';
  lblGen.Caption:='MVA';
  lblLoad.Caption:='MVA';
  lblLoss.Caption:='MVA';
end;
//
mulai:=time;
NewtonPolar(gBus,gBranch,gGeneral);
selesai:=time;
fgBusA.RowCount:=high(gBus)+2;
for i:=0 to high(gBus) do
```

```
begin
  fgBusA.Cells[0,i+1]:=IntToStr(i+1);
  fgBusA.Cells[1,i+1]:=RealToStr(gBus[i,0],5);
  fgBusA.Cells[2,i+1]:=RealToStr(gBus[i,1],5);
  fgBusA.Cells[3,i+1]:=RealToStr(gBus[i,2],3);
  fgBusA.Cells[4,i+1]:=RealToStr(gBus[i,3],3);
  fgBusA.Cells[5,i+1]:=RealToStr(gBus[i,4],3);
  fgBusA.Cells[6,i+1]:=RealToStr(gBus[i,5],3);
  fgBusA.Cells[7,i+1]:=RealToStr(gBus[i,6],3);
  fgBusA.Cells[8,i+1]:=RealToStr(gBus[i,7],0);
end;
fgBranchA.RowCount:=high(gBranch)+2;
for i:=0 to high(gBranch) do
begin
  dari:=round(gBranch[i,0]);
  ker:=round(gBranch[i,1]);
  fgBranchA.Cells[0,i+1]:=IntToStr(i+1);
  fgBranchA.Cells[1,i+1]:=IntToStr(dari);
  fgBranchA.Cells[2,i+1]:=IntToStr(ker);
  fgBranchA.Cells[3,i+1]:=RealToStr(gBranch[i,9],3);
  fgBranchA.Cells[4,i+1]:=RealToStr(gBranch[i,10],3);
  fgBranchA.Cells[5,i+1]:=RealToStr(gBranch[i,13],3);
  fgBranchA.Cells[6,i+1]:=RealToStr(gBranch[i,14],3);
  fgBranchA.Cells[7,i+1]:=IntToStr(ker);
  fgBranchA.Cells[8,i+1]:=IntToStr(dari);
  fgBranchA.Cells[9,i+1]:=RealToStr(gBranch[i,11],3);
  fgBranchA.Cells[10,i+1]:=RealToStr(gBranch[i,12],3);
  fgBranchA.Cells[11,i+1]:=RealToStr(gBranch[i,15],3);
  fgBranchA.Cells[12,i+1]:=RealToStr(gBranch[i,16],3);
end;
edtSumGenA.Text:=gGeneral.sumGen.toStringJ(3);
edtSumLoadA.Text:=gGeneral.sumLoad.toStringJ(3);
edtSumLossA.Text:=gGeneral.sumLoss.toStringJ(3);
edtIterasiA.Text:=IntToStr(gGeneral.Iterasi);
selang:=selesai-mulai;
DecodeTime(selang,jam,menit,detik,mdetik);
edtTimeA.Text:=IntToStr(jam)+':'+IntToStr(menit)+':'+
              IntToStr(detik)+':'+IntToStr(mdetik);
//
LamdaV:=StrToFloat(edtLamdaV.Text);
LamdaS:=StrToFloat(edtLamdaS.Text);
LamdaQ:=StrToFloat(edtLamdaQ.Text);
BatasV.min:=StrToFloat(edtVmin.Text);
BatasV.max:=StrToFloat(edtVmax.Text);
gObjFunc:=TObjFunc.Create(LamdaV,LamdaQ,LamdaS,BatasV);
Pgen:=gObjFunc.getPgen(gBus);
CostPLN:=gObjFunc.getCostPgen(Pgen);
edtSebelumOpt.Text:=FormatFloat('#,##0',CostPLN);
Batas:=gObjFunc.FindBatasChrom(BatasV);
//
MaxGen:=StrToInt(edtMaxGen.Text);
pbIterasi.Max:=MaxGen;
PopSize:=StrToInt(edtPopSize.Text);
Length:=gObjFunc.FindLength;
Betha:=StrToFloat(edtBetha.Text);
Ka:=StrToFloat(edtKa.Text);
ep:=TEvoProi.Create(MaxGen,PopSize,Length,Betha,Ka,Batas);
mulai:=time;
BestChrom:=ep.BestChrom;
Min:=ep.Min;
Avg:=ep.Avg;
Max:=ep.Max;
selesai:=time;
ep.Free;
selang:=selesai-mulai;
DecodeTime(selang,jam,menit,detik,mdetik);
edtTime.Text:=IntToStr(jam)+':'+IntToStr(menit)+':'+
              IntToStr(detik)+':'+IntToStr(mdetik);
```

```
  LSetting:=gObjFunc.DecodeChromAkhir(BestChrom);
  fgHasil.RowCount:=high(LSetting)+2;
  for i:=0 to high(LSetting) do
  begin
    fgHasil.Cells[0,i+1]:=IntToStr(i+1);
    fgHasil.Cells[1,i+1]:=RealToStr(LSetting[i,0]+1,0);
    fgHasil.Cells[2,i+1]:=RealToStr(LSetting[i,1],5);
    fgHasil.Cells[3,i+1]:=RealToStr(LSetting[i,2],2);
  end;
  gObjFunc.doHitungAkhir(BestChrom,LBus,LBranch,LGeneral,CostOpt);
  Pgen:=gObjFunc.getPgen(LBus);
  CostOpt:=gObjFunc.getCostPgen(Pgen);
  edtSesudahOpt.Text:=FormatFloat('#,##0',CostOpt);
  edtSelisihOpt.Text:=FormatFloat('#,##0.00',(CostPLN-CostOpt));
  fgBus.RowCount:=high(LBus)+2;
  Series1.Clear;
  Series2.Clear;
  for i:=0 to high(LBus) do
  begin
    fgBus.Cells[0,i+1]:=IntToStr(i+1);
    fgBus.Cells[1,i+1]:=RealToStr(LBus[i,0],5);
    fgBus.Cells[2,i+1]:=RealToStr(LBus[i,1],5);
    fgBus.Cells[3,i+1]:=RealToStr(LBus[i,2],3);
    fgBus.Cells[4,i+1]:=RealToStr(LBus[i,3],3);
    fgBus.Cells[5,i+1]:=RealToStr(LBus[i,4],3);
    fgBus.Cells[6,i+1]:=RealToStr(LBus[i,5],3);
    fgBus.Cells[7,i+1]:=RealToStr(LBus[i,6],3);
    fgBus.Cells[8,i+1]:=RealToStr(LBus[i,7],0);
    Series1.Add(gBus[i,0],IntToStr(i+1));
    Series2.Add(LBus[i,0],IntToStr(i+1));
  end;
  fgHasil.Cells[3,1]:=RealToStr(LBus[0,2],3);
  fgBranch.RowCount:=high(LBranch)+2;
  for i:=0 to high(LBranch) do
  begin
    dari:=round(LBranch[i,0]);
    ker:=round(LBranch[i,1]);
    fgBranch.Cells[0,i+1]:=IntToStr(i+1);
    fgBranch.Cells[1,i+1]:=IntToStr(dari);
    fgBranch.Cells[2,i+1]:=IntToStr(ker);
    fgBranch.Cells[3,i+1]:=RealToStr(LBranch[i,9],3);
    fgBranch.Cells[4,i+1]:=RealToStr(LBranch[i,10],3);
    fgBranch.Cells[5,i+1]:=RealToStr(LBranch[i,13],3);
    fgBranch.Cells[6,i+1]:=RealToStr(LBranch[i,14],3);
    fgBranch.Cells[7,i+1]:=IntToStr(ker);
    fgBranch.Cells[8,i+1]:=IntToStr(dari);
    fgBranch.Cells[9,i+1]:=RealToStr(LBranch[i,11],3);
    fgBranch.Cells[10,i+1]:=RealToStr(LBranch[i,12],3);
    fgBranch.Cells[11,i+1]:=RealToStr(LBranch[i,15],3);
    fgBranch.Cells[12,i+1]:=RealToStr(LBranch[i,16],3);
  end;
  edtSumGen.Text:=LGeneral.sumGen.toStringJ(3);
  edtSumLoad.Text:=LGeneral.sumLoad.toStringJ(3);
  edtSumLoss.Text:=LGeneral.sumLoss.toStringJ(3);
  edtIterasi.Text:=IntToStr(LGeneral.Iterasi);
  gObjFunc.Free;
  Series3.Clear;
  Series4.Clear;
  Series5.Clear;
  for i:=0 to high(Min) do
  begin
    Series3.Add(Min[i],IntToStr(i+1));
    Series4.Add(Avg[i],IntToStr(i+1));
    Series5.Add(Max[i],IntToStr(i+1));
  end;
end;

procedure TfrmParamTS.FormCreate(Sender: TObject);
```

```
begin
  fgHasil.Cells[0,0]:='No';
  fgHasil.Cells[1,0]:='Bus';
  fgHasil.Cells[2,0]:='absV (pu)';
  fgHasil.Cells[3,0]:='Pgen';
  cmbMataUang.Text:='IDR Rp';
  lblLoss1.Caption:='Rp/jam';
  lblLoss2.Caption:='Rp/jam';
  lblLoss3.Caption:='Rp/jam';
end;

procedure TfrmParamTS.btnClientClick(Sender: TObject);
var paramFitness:TStringList;
begin
  paramFitness:=TStringList.Create;
  paramFitness.Add('Param');
  paramFitness.Add(edtLamdaV.Text);
  paramFitness.Add(edtLamdaS.Text);
  paramFitness.Add(edtLamdaQ.Text);
  paramFitness.Add(edtVmin.Text);
  paramFitness.Add(edtVmax.Text);
  frmSetting.ClientSocket1.Socket.SendText(paramFitness.Text);
  frmSetting.ClientSocket2.Socket.SendText(paramFitness.Text);
  frmSetting.ClientSocket3.Socket.SendText(paramFitness.Text);
  frmSetting.ClientSocket4.Socket.SendText(paramFitness.Text);
  paramFitness.Free;
  if cmbMataUang.Text='IDR Rp' then
  begin
    lblLoss1.Caption:='Rp/jam';
    lblLoss2.Caption:='Rp/jam';
    lblLoss3.Caption:='Rp/jam';
  end
  else
  begin
    lblLoss1.Caption:='$/jam';
    lblLoss2.Caption:='$/jam';
    lblLoss3.Caption:='$/jam';
  end;
  btnHitung.Enabled:=true;
end;

end.
```

```
unit uSetting;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ScktComp, ComCtrls;

type
  TfrmSetting = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    edtServer1: TEdit;
    edtServer2: TEdit;
    edtServer4: TEdit;
    edtServer3: TEdit;
    CheckBox1: TCheckBox;
    CheckBox2: TCheckBox;
    CheckBox3: TCheckBox;
    CheckBox4: TCheckBox;
    btnClose: TButton;
    ClientSocket1: TClientSocket;
    ClientSocket2: TClientSocket;
    ClientSocket3: TClientSocket;
    ClientSocket4: TClientSocket;
    StatusBar1: TStatusBar;
    procedure btnCloseClick(Sender: TObject);
    procedure CheckBox1Click(Sender: TObject);
    procedure CheckBox2Click(Sender: TObject);
    procedure CheckBox3Click(Sender: TObject);
    procedure CheckBox4Click(Sender: TObject);
    procedure ClientSocket1Read(Sender: TObject; Socket: TCustomWinSocket);
    procedure ClientSocket1Connecting(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure FormCreate(Sender: TObject);
    procedure ClientSocket1Disconnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ClientSocket2Connecting(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ClientSocket2Disconnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ClientSocket2Read(Sender: TObject; Socket: TCustomWinSocket);
    procedure ClientSocket3Connecting(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ClientSocket3Disconnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ClientSocket3Read(Sender: TObject; Socket: TCustomWinSocket);
    procedure ClientSocket4Connecting(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ClientSocket4Disconnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ClientSocket4Read(Sender: TObject; Socket: TCustomWinSocket);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmSetting: TfrmSetting;

implementation

uses uVarGlobal, uParamTS, uInputLF;

{$R *.dfm}
```

```
procedure TfrmSetting.btnCloseClick(Sender: TObject);
begin
  Close;
end;

procedure TfrmSetting.CheckBox1Click(Sender: TObject);
begin
  if not ClientSocket1.Active then
  begin
    ClientSocket1.Address:=edtServer1.Text;
  end;
  ClientSocket1.Active:=CheckBox1.Checked;
end;

procedure TfrmSetting.CheckBox2Click(Sender: TObject);
begin
  if not ClientSocket2.Active then
  begin
    ClientSocket2.Address:=edtServer2.Text;
  end;
  ClientSocket2.Active:=CheckBox2.Checked;
end;

procedure TfrmSetting.CheckBox3Click(Sender: TObject);
begin
  if not ClientSocket3.Active then
  begin
    ClientSocket3.Address:=edtServer3.Text;
  end;
  ClientSocket3.Active:=CheckBox3.Checked;
end;

procedure TfrmSetting.CheckBox4Click(Sender: TObject);
begin
  if not ClientSocket4.Active then
  begin
    ClientSocket4.Address:=edtServer4.Text;
  end;
  ClientSocket4.Active:=CheckBox4.Checked;
end;

procedure TfrmSetting.ClientSocket1Read(Sender: TObject;
  Socket: TCustomWinSocket);
var str:TStringList;
begin
  strPC1:=Socket.ReceiveText;
  if strPC1='Setting Param Diterima PC1' then
  begin
    statPC1:=false;
    frmParamT3.lbxPesan1.Items.Add('Setting Param diterima PC1');
  end
  else if strPC1='File Data Diterima PC1' then
  begin
    frmInput.lbxStatFile.Items.Add(strPC1);
  end
  else
  begin
    str:=TStringList.Create;
    str.Text:=strPC1;
    if str.Strings[0]='Fitness' then
    begin
      statPC1:=true;
    end;
    str.Free;
  end;
end;
```

```
procedure TfrmSetting.ClientSocket1Connecting(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  StatusBar1.Panels[0].Text:='PC1 Connect';
end;

procedure TfrmSetting.FormCreate(Sender: TObject);
begin
  StatusBar1.Panels[0].Text:='PC1 Disconnect';
  StatusBar1.Panels[1].Text:='PC2 Disconnect';
  StatusBar1.Panels[2].Text:='PC3 Disconnect';
  StatusBar1.Panels[3].Text:='PC4 Disconnect';
end;

procedure TfrmSetting.ClientSocket1Disconnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  StatusBar1.Panels[0].Text:='PC1 Disconnect';
end;

procedure TfrmSetting.ClientSocket2Connecting(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  StatusBar1.Panels[1].Text:='PC2 Connect';
end;

procedure TfrmSetting.ClientSocket2Disconnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  StatusBar1.Panels[1].Text:='PC2 Disconnect';
end;

procedure TfrmSetting.ClientSocket2Read(Sender: TObject;
  Socket: TCustomWinSocket);
var str:TStringList;
begin
  strPC2:=Socket.ReceiveText;
  if strPC2='Setting Param Diterima PC2' then
  begin
    statPC2:=false;
    frmParamTS.lbxPesan2.Items.Add('Setting Param diterima PC2');
  end
  else if strPC2='File Data Diterima PC2' then
  begin
    frmInput.lbxStatFile.Items.Add(strPC2);
  end
  else
  begin
    str:=TStringList.Create;
    str.Text:=strPC2;
    if str.Strings[0]='Fitness' then
    begin
      statPC2:=true;
    end;
    str.Free;
  end;
end;

procedure TfrmSetting.ClientSocket3Connecting(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  StatusBar1.Panels[2].Text:='PC3 Connect';
end;

procedure TfrmSetting.ClientSocket3Disconnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  StatusBar1.Panels[2].Text:='PC3 Disconnect';
```

```
end;

procedure TfrmSetting.ClientSocket3Read(Sender: TObject;
  Socket: TCustomWinSocket);
var str:TStringList;
begin
  strPC3:=Socket.ReceiveText;
  if strPC3='Setting Param Diterima PC3' then
  begin
    statPC3:=false;
    frmParamTS.lbxPesan3.Items.Add('Setting Param diterima PC3');
  end
  else if strPC3='File Data Diterima PC3' then
  begin
    frmInput.lbxStatFile.Items.Add(strPC3);
  end
  else
  begin
    str:=TStringList.Create;
    str.Text:=strPC3;
    if str.Strings[0]='Fitness' then
    begin
      statPC3:=true;
    end;
    str.Free;
  end;
end;

procedure TfrmSetting.ClientSocket4Connecting(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  StatusBar1.Panels[3].Text:='PC4 Connect';
end;

procedure TfrmSetting.ClientSocket4Disconnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  StatusBar1.Panels[3].Text:='PC4 Disconnect';
end;

procedure TfrmSetting.ClientSocket4Read(Sender: TObject;
  Socket: TCustomWinSocket);
var str:TStringList;
begin
  strPC4:=Socket.ReceiveText;
  if strPC4='Setting Param Diterima PC4' then
  begin
    statPC4:=false;
    frmParamTS.lbxPesan4.Items.Add('Setting Param diterima PC4');
  end
  else if strPC4='File Data Diterima PC4' then
  begin
    frmInput.lbxStatFile.Items.Add(strPC4);
  end
  else
  begin
    str:=TStringList.Create;
    str.Text:=strPC4;
    if str.Strings[0]='Fitness' then
    begin
      statPC4:=true;
    end;
    str.Free;
  end;
end;

end.
```

```
unit uTabuSearch;

interface

uses uUtils,uFitness,uParamTS;

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

  TPopulasi=array of TIndividu;

  TTabuSearch=class
  private
    FIterasiMax,FImproveMax,FNTabu,FLength,FIterasi,FImprove:integer;
    FLamdaV,FLamdaS:double;
    FBatasV:TBatas;
    FBatas:TBatasArr1;
    FIndiTabu:TPopulasi;
    FIndiBest:TIndividu;
    function BuatTetangga(const rIndi:TIndividu):TIndividu;
    function CheckChrom(const rChrom:dArr1):boolean;
    function getIndividu(const rIndi:TIndividu):TIndividu;
    procedure InsertTabu(const rIndi:TIndividu);
    function getBestChrom:dArr1;
    procedure doHitung;
  public
    constructor Create(const rIterasiMax,rImproveMax,rNtabu,
                rLength:integer;
                const rLamdaV,rLamdaS:double;
                const rBatasV:TBatas;
                const rBatas:TBatasArr1);
    property BestChrom:dArr1 read getBestChrom;
  end;

implementation

//constructor
constructor TTabuSearch.Create(const rIterasiMax,rImproveMax,
            rNtabu,rLength:integer;
            const rLamdaV,rLamdaS:double;
            const rBatasV:TBatas;
            const rBatas:TBatasArr1);
var i,j:integer;
begin
  inherited Create;
  FIterasiMax:=rIterasiMax;
  FImproveMax:=rImproveMax;
  FNtabu:=rNtabu;
  FLength:=rLength;
  FLamdaV:=rLamdaV;
  FLamdaS:=rLamdaS;
  FBatasV.min:=rBatasV.min;
  FBatasV.max:=rBatasV.max;
  SetLength(FBatas,FLength);
  for i:=0 to FLength-1 do
  begin
    FBatas[i].min:=rBatas[i].min;
    FBatas[i].max:=rBatas[i].max;
  end;
  SetLength(FIndiTabu,FNtabu);
  for i:=0 to FNtabu-1 do
  begin
    SetLength(FIndiTabu[i].chrom,FLength);
    for j:=0 to FLength-1 do
    begin
```

```
      FIndiTabu[i].chrom[j]:=0;
    end;
    FIndiTabu[i].fitness:=0;
  end;
  SetLength(FIndiBest.chrom,FLength);
  for i:=0 to FLength-1 do
  begin
    FIndiBest.chrom[i]:=0;
  end;
  FIndiBest.fitness:=0;
end;

function TTabuSearch.BuatTetangga(const rIndi:TIndividu):TIndividu;
var i:integer;
    rand:double;
begin
  SetLength(result.chrom,FLength);
  for i:=0 to FLength-1 do
  begin
    rand:=random;
    if rand<=0.5 then
    begin
      result.chrom[i]:=rIndi.chrom[i]+random*(FBatas[i].max-FBatas[i].min);
    end
    else
    begin
      result.chrom[i]:=rIndi.chrom[i]-random*(FBatas[i].max-FBatas[i].min);
    end;
    if result.chrom[i]>FBatas[i].max then
    begin
      result.chrom[i]:=FBatas[i].max;
    end
    else if result.chrom[i]<FBatas[i].min then
    begin
      result.chrom[i]:=FBatas[i].min;
    end;
  end;
  result.fitness:=doHitungFitness(result.chrom,FBatasV,FLamdaV,FLamdaS);
end;

function TTabuSearch.CheckChrom(const rChrom:dArr1):boolean;
var i,j:integer;
begin
  result:=true;
  for i:=0 to FNtabu-1 do
  begin
    if FIndiTabu[i].fitness<>0 then
    begin
      for j:=0 to FLength-1 do
      begin
        if rChrom[j]<>FIndiTabu[i].chrom[j] then
        begin
          result:=false;
          break;
        end;
      end;
    end;
  end;
end;

function TTabuSearch.getIndividu(const rIndi:TIndividu):TIndividu;
var i:integer;
begin
  SetLength(result.chrom,FLength);
  for i:=0 to FLength-1 do
  begin
    result.chrom[i]:=rIndi.chrom[i];
  end;
```

```
  result.fitness:=rIndi.fitness;
end;

procedure TTabuSearch.InsertTabu(const rIndi:TIndividu);
var i,pos,posMax:integer;
    Max,Min:double;
    check:boolean;
begin
  pos:=0;
  for i:=0 to FNtabu-1 do
  begin
    if FIndiTabu[i].fitness<>0 then
    begin
      inc(pos);
    end;
  end;
  if pos=0 then
  begin
    FIndiTabu[pos]:=getIndividu(rIndi);
  end
  else
  begin
    check:=CheckChrom(rIndi.chrom);
    if check=false then
    begin
      if pos<FNtabu-1 then
      begin
        FIndiTabu[pos+1]:=getIndividu(rIndi);
      end
      else
      begin
        posMax:=0;
        max:=FIndiTabu[0].fitness;
        min:=FIndiTabu[0].fitness;
        for i:=0 to FNtabu-1 do
        begin
          if max<FIndiTabu[i].fitness then
          begin
            posMax:=i;
            max:=FIndiTabu[i].fitness;
          end;
          if (min>FIndiTabu[i].fitness) and (FIndiTabu[i].fitness<>0) then
          begin
            min:=FIndiTabu[i].fitness;
          end;
        end;
        if rIndi.fitness<FIndiTabu[posMax].fitness then
        begin
          FIndiTabu[posMax]:=getIndividu(rIndi);
        end;
        if rIndi.fitness<min then
        begin
          FImprove:=0;
        end
        else
        begin
          inc(FImprove);
        end;
        inc(FIterasi);
        frmParamTS.pbIterasi.StepBy(1);
      end;
    end;
  end;
end;

procedure TTabuSearch.doHitung;
var Tetangga:TIndividu;
begin
```

```
  //initial counter
  FIterasi:=0;
  FImprove:=0;
  //initial FIndiBest
  FIndiBest.chrom:=InitChrom;
  FIndiBest.fitness:=doHitungFitness(FIndiBest.chrom,FBatasV,
                     FLamdaV,FLamdaS);
  //proses Tabu Search ada disini lho!!
  repeat
    Tetangga:=BuatTetangga(FIndiBest);
    InsertTabu(Tetangga);
    if FIndiBest.fitness>Tetangga.fitness then
    begin
      FIndiBest:=getIndividu(Tetangga);
    end;
  until (FIterasi>=FIterasiMax) or (FImprove>=FImproveMax);
end;

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

end.
```

```
program OpfEPParallel;

uses
  Forms,
  uAbout in 'uAbout.pas' {frmAbout},
  uComplex in 'uComplex.pas',
  uMatrix in 'uMatrix.pas',
  uMenu in 'uMenu.pas' {frmMenu},
  uUtils in 'uUtils.pas',
  uInputLF in 'Loadflow\uInputLF.pas' {frmInput},
  uLoadflow in 'Loadflow\uLoadflow.pas',
  uNewtonPolar in 'Loadflow\NewtonPolar\uNewtonPolar.pas',
  uFitness in 'TabuSearch\uFitness.pas',
  uParamTS in 'TabuSearch\uParamTS.pas' {frmParamTS},
  uTabuSearch in 'TabuSearch\uTabuSearch.pas',
  uRandom in 'EvoPro\uRandom.pas',
  uEvoPro in 'EvoPro\uEvoPro.pas',
  uObjFunc in 'EvoPro\uObjFunc.pas',
  uSetting in 'uSetting.pas' {frmSetting},
  uVarGlobal in 'uVarGlobal.pas';

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TfrmMenu, frmMenu);
  Application.CreateForm(TfrmInput, frmInput);
  Application.CreateForm(TfrmAbout, frmAbout);
  Application.CreateForm(TfrmParamTS, frmParamTS);
  Application.CreateForm(TfrmSetting, frmSetting);
  Application.Run;
end.
```

```
unit uEvoPro;

interface

uses uUtils,uRandom,uObjFunc,uParamTS,uSetting,Classes,SysUtils,Forms;

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

  TPopulasi1=array of TIndividu1;

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

  TPopulasi2=array of TIndividu2;

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

  TEvoPro1=class(TEvoPro)
  private
    FBatas:TBatasArr1;
    FBestIndi:TIndividu1;
    FParent,FChild:TPopulasi1;
    FMin1,FAvg1,FMax1,FKa:double;
    function getBatas:TBatasArr1;
    procedure setBatas(const rBatas:TBatasArr1);
    function getIndividu(const rIndi:TIndividu1):TIndividu1;
    procedure SwapIndi(var rIndi1,rIndi2:TIndividu1);
    function FindFitnessMax:double;
    function FindIndiMax:TIndividu1;
    function DecodeStrToFitness(const rStr:string):dArr1;
    procedure InitParent;
    procedure Statistik;
    procedure Generasi;
    procedure Kompetisi;
    procedure doHitung;
    function getBestChrom:dArr1;
  public
    constructor Create(const rMaxGen,rPopSize,rLength:integer;
                const rBetha,rKa:double;
                const rBatas:TBatasArr1);
    property Batas:TBatasArr1 read getBatas write setBatas;
```

```
    property Ka:double read FKa write FKa;
    property BestChrom:dArr1 read getBestChrom;
  end;

implementation

uses uVarGlobal;

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

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

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

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

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

{ TEvoPro1 }

//constructor
constructor TEvoPro1.Create(const rMaxGen,rPopSize,rLength:integer;
            const rBetha,rKa:double;
            const rBatas:TBatasArr1);
```

```
var i:integer;
begin
  inherited Create(rMaxGen,rPopSize,rLength,rBetha);
  FKa:=rKa;
  SetLength(FBatas,Length);
  for i:=0 to Length-1 do
  begin
    FBatas[i].min:=rBatas[i].min;
    FBatas[i].max:=rBatas[i].max;
  end;
end;

function TEvoPro1.getBatas:TBatasArr1;
var i:integer;
begin
  SetLength(result,Length);
  for i:=0 to Length-1 do
  begin
    result[i].min:=FBatas[i].min;
    result[i].max:=FBatas[i].max;
  end;
end;

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

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

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

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

function TEvoPro1.FindIndiMax:TIndividu1;
```

```
var i:integer;
begin
  result:=getIndividu(FParent[0]);
  for i:=1 to PopSize-1 do
  begin
    if result.fitness<FParent[i].fitness then
    begin
      result:=getIndividu(FParent[i]);
    end;
  end;
end;

function TEvoProl.DecodeStrToFitness(const rStr:string):dArr1;
var i,sa:integer;
    str:TStringList;
begin
  str:=TStringList.Create;
  str.Text:=rStr;
  sa:=0;
  for i:=1 to str.Count-1 do
  begin
    if str.Strings[i]<>'' then
    begin
      inc(sa);
    end;
  end;
  SetLength(result,sa);
  sa:=0;
  for i:=1 to str.Count-1 do
  begin
    if str.Strings[i]<>'' then
    begin
      result[sa]:=StrToFloat(str.Strings[i]);
      inc(sa);
    end;
  end;
  str.Free;
end;

procedure TEvoProl.InitParent;
var i,j:integer;
begin
  SetLength(FParent,FPopSize);
  SetLength(FChild,FPopSize);
  SetLength(FMin,FMaxGen);
  SetLength(FAvg,FMaxGen);
  SetLength(FMax,FMaxGen);
  for i:=0 to PopSize-1 do
  begin
    SetLength(FParent[i].chrom,Length);
    SetLength(FChild[i].chrom,Length);
    for j:=0 to Length-1 do
    begin
      FParent[i].chrom[j]:=FRandom.NextDouble(FBatas[j].min,FBatas[j].max);
    end;
    FParent[i].fitness:=FKa/gObjFunc.doHitung(FParent[i].chrom);
  end;
end;

procedure TEvoProl.Statistik;
var i:integer;
    sumFitness:double;
begin
  sumFitness:=0;
  FMin1:=FParent[0].fitness;
  FMax1:=FParent[0].fitness;
  for i:=1 to PopSize-1 do
  begin
```

```
    sumFitness:=sumFitness+FParent[i].fitness;
    if FMin1>FParent[i].fitness then
    begin
      FMin1:=FParent[i].fitness;
    end;
    if FMax1<FParent[i].fitness then
    begin
      FMax1:=FParent[i].fitness;
    end;
  end;
  FAvg1:=sumFitness/FPopSize;
end;

procedure TEvoPro1.Generasi;
var i,j,sa:integer;
    Fmax:double;
    tho:double;
    fitness:dArr1;
    strIndi:TStringList;
    strPop1,strPop2,strPop3,strPop4:TStringList;
begin
  Fmax:=FindFitnessMax;
  strPop1:=TStringList.Create;
  strPop1.Add('DataChrom');
  strPop2:=TStringList.Create;
  strPop2.Add('DataChrom');
  strPop3:=TStringList.Create;
  strPop3.Add('DataChrom');
  strPop4:=TStringList.Create;
  strPop4.Add('DataChrom');
  for i:=0 to PopSize-1 do
  begin
    strIndi:=TStringList.Create;
    strIndi.Clear;
    for j:=0 to Length-1 do
    begin
      tho:=(FBatas[j].max-FBatas[j].min)*((Fmax-FParent[i].fitness)/
           Fmax+Betha);

FChild[i].chrom[j]:=FParent[i].chrom[j]+FRandom.NextGaussian(0,sqr(tho));
      if FChild[i].chrom[j]>FBatas[j].max then
      begin
        FChild[i].chrom[j]:=FBatas[j].max;
      end;
      if FChild[i].chrom[j]<FBatas[j].min then
      begin
        FChild[i].chrom[j]:=FBatas[j].min;
      end;
      strIndi.Add(FloatToStr(FChild[i].chrom[j]));
    end;
    if (i>=0) and (i<=9) then
    begin
      strPop1.Add(strIndi.Text);
    end
    else if (i>=10) and (i<=19) then
    begin
      strPop2.Add(strIndi.Text);
    end
    else if (i>=20) and (i<=29) then
    begin
      strPop3.Add(strIndi.Text);
    end
    else if (i>=30) and (i<=39) then
    begin
      strPop4.Add(strIndi.Text);
    end;
    strIndi.Free;
  end;
```

```
  frmSetting.ClientSocket1.Socket.SendText(strPop1.Text);
  frmSetting.ClientSocket2.Socket.SendText(strPop2.Text);
  frmSetting.ClientSocket3.Socket.SendText(strPop3.Text);
  frmSetting.ClientSocket4.Socket.SendText(strPop4.Text);
  statPC1:=false;
  statPC2:=false;
  statPC3:=false;
  statPC4:=false;
  repeat
    Application.ProcessMessages;
  until (statPC1=true) and (statPC2=true) and (statPC3=true) and
  (statPC4=true);
  fitness:=DecodeStrToFitness(strPC1);
  sa:=0;
  for i:=0 to 9 do
  begin
    FChild[i].fitness:=FKa/fitness[sa];
    inc(sa);
  end;
  frmParamTS.lbxPesan1.Items.Text:=strPC1;
  strPop1.Free;
  fitness:=DecodeStrToFitness(strPC2);
  sa:=0;
  for i:=10 to 19 do
  begin
    FChild[i].fitness:=FKa/fitness[sa];
    inc(sa);
  end;
  frmParamTS.lbxPesan2.Items.Text:=strPC2;
  strPop2.Free;
  fitness:=DecodeStrToFitness(strPC3);
  sa:=0;
  for i:=20 to 29 do
  begin
    FChild[i].fitness:=FKa/fitness[sa];
    inc(sa);
  end;
  frmParamTS.lbxPesan3.Items.Text:=strPC3;
  strPop3.Free;
  fitness:=DecodeStrToFitness(strPC4);
  sa:=0;
  for i:=30 to 39 do
  begin
    FChild[i].fitness:=FKa/fitness[sa];
    inc(sa);
  end;
  frmParamTS.lbxPesan4.Items.Text:=strPC4;
  strPop4.Free;
end;

procedure TEvoPro1.Kompetisi;
var i,j,Ntmp,sa:integer;
    tmp:TPopulasi1;
    sort:iArr1;
begin
  Ntmp:=2*PopSize;
  SetLength(tmp,Ntmp);
  for i:=0 to PopSize-1 do
  begin
    tmp[i]:=getIndividu(FParent[i]);
    tmp[PopSize+i]:=getIndividu(FChild[i]);
  end;
  SetLength(sort,Ntmp);
  for i:=0 to Ntmp-1 do
  begin
    sort[i]:=0;
  end;
  for i:=0 to Ntmp-1 do
```

```
  begin
    for j:=0 to Ntmp-2 do
    begin
      repeat
        sa:=FRandom.NextInt(0,(Ntmp-1));
      until sa<>i;
      if tmp[i].fitness>tmp[sa].fitness then
      begin
        inc(sort[i]);
      end;
    end;
  end;
  for i:=0 to Ntmp-2 do
  begin
    for j:=i to Ntmp-1 do
    begin
      if sort[i]<sort[j] then
      begin
        Swap(sort[i],sort[j]);
        SwapIndi(tmp[i],tmp[j]);
      end;
    end;
  end;
  for i:=0 to PopSize-1 do
  begin
    FParent[i]:=getIndividu(tmp[i]);
  end;
end;

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

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

end.
```

```
unit uFitness;

interface

uses uUtils,uNewtonPolar,uLoadflow;

function FindLength:integer;
function FindBatasChrom(var rBatasV:TBatas):TBatasArr1;
function InitChrom:dArr1;
function doHitungFitness(const rChrom:dArr1;
         const rBatasV:TBatas;
         const rLamdaV,rLamdaS:double):double;
procedure doHitungFitnessAkhir(const rChrom:dArr1;
          var rLBus,rLBranch:dArr2;
          var rGeneral:TGeneral);
function DecodeChromAkhir(const rChrom:dArr1):dArr2;

implementation

function FindLength:integer;
var i:integer;
begin
  result:=0;
  for i:=0 to high(gBus) do
  begin
    if gBus[i,7]=1 then
    begin
      inc(result);
    end
    else if gBus[i,7]=2 then
    begin
      inc(result);
      inc(result);
    end;
  end;
end;

function FindBatasChrom(var rBatasV:TBatas):TBatasArr1;
var i,sa,Ngen:integer;
begin
  sa:=FindLength;
  SetLength(result,sa);
  sa:=0;
  Ngen:=0;
  for i:=0 to high(gBus) do
  begin
    if gBus[i,7]=1 then
    begin
      result[sa].min:=rBatasV.min;
      result[sa].max:=rBatasV.max;
      inc(sa);
      inc(Ngen);
    end
    else if gBus[i,7]=2 then
    begin
      result[sa].min:=rBatasV.min;
      result[sa].max:=rBatasV.max;
      inc(sa);
      result[sa].min:=gGen[Ngen,8];
      result[sa].max:=gGen[Ngen,9];
      inc(sa);
      inc(Ngen);
    end;
  end;
end;

function InitChrom:dArr1;
var i,sa:integer;
```

```
begin
  sa:=FindLength;
  SetLength(result,sa);
  sa:=0;
  for i:=0 to high(gBus) do
  begin
    if gBus[i,7]=1 then
    begin
      result[sa]:=gBus[i,0];
      inc(sa);
    end
    else if gBus[i,7]=2 then
    begin
      result[sa]:=gBus[i,0];
      inc(sa);
      result[sa]:=gBus[i,2];
      inc(sa);
    end;
  end;
end;

function DecodeChrom(const rChrom:dArr1):dArr2;
var i,j,sa:integer;
begin
  SetLength(result,high(gBus)+1,high(gBus[0])+1);
  for i:=0 to high(gBus) do
  begin
    for j:=0 to high(gBus[0]) do
    begin
      result[i,j]:=gBus[i,j];
    end;
  end;
  sa:=0;
  for i:=0 to high(gBus) do
  begin
    if gBus[i,7]=1 then
    begin
      result[i,0]:=rChrom[sa];
      inc(sa);
    end
    else if gBus[i,7]=2 then
    begin
      result[i,0]:=rChrom[sa];
      inc(sa);
      result[i,2]:=rChrom[sa];
      inc(sa);
    end;
  end;
end;

function DecodeChromAkhir(const rChrom:dArr1):dArr2;
var i,sa,ne:integer;
begin
  ne:=0;
  for i:=0 to high(gBus) do
  begin
    if gBus[i,7]=1 then
    begin
      inc(ne);
    end
    else if gBus[i,7]=2 then
    begin
      inc(ne);
    end;
  end;
  SetLength(result,ne,4);
  sa:=0;
  ne:=0;
```

```
  for i:=0 to high(gBus) do
  begin
    if gBus[i,7]=1 then
    begin
      result[ne,0]:=i;
      result[ne,1]:=rChrom[sa];
      inc(sa);
      inc(ne);
    end
    else if gBus[i,7]=2 then
    begin
      result[ne,0]:=i;
      result[ne,1]:=rChrom[sa];
      inc(sa);
      result[ne,2]:=rChrom[sa];
      inc(sa);
      inc(ne);
    end;
  end;
end;

function getPgen(const rLbus:dArr2):dArr1;
var i,Nbus,length:integer;
begin
  Nbus:=high(rLbus)+1;
  length:=0;
  for i:=0 to Nbus-1 do
  begin
    if rLbus[i,7]<>3 then
    begin
      inc(length);
    end;
  end;
  SetLength(result,length);
  length:=0;
  for i:=0 to Nbus-1 do
  begin
    if rLbus[i,7]<>3 then
    begin
      result[length]:=rLbus[i,2];
      inc(length);
    end;
  end;
end;

function getCostPgen(const rPgen:dArr1):double;
var i,Ngen:integer;
begin
  Ngen:=high(gGen)+1;
  result:=0;
  for i:=0 to Ngen-1 do
  begin
    result:=result+getCostGen(i,gGen,rPgen[i]);
  end;
end;

function FindPinV(const rLBus:dArr2;
         const rBatasV:TBatas):integer;
var i:integer;
begin
  result:=0;
  for i:=0 to high(rLBus) do
  begin
    if rLBus[i,0]>rBatasV.max then
    begin
      inc(result);
    end;
    if rLBus[i,0]<rBatasV.min then
```

```
    begin
      inc(result);
    end;
  end;
end;

function FindPinKapBranch(const rLBranch:dArr2):double;
var i:integer;
    absAlir:double;
begin
  result:=0;
  for i:=0 to high(rLBranch) do
  begin
    if rLBranch[i,9]>0 then
    begin
      absAlir:=sqrt(sqr(rLBranch[i,9])+sqr(rLBranch[i,10]));
      if absAlir>rLBranch[i,8] then
      begin
        result:=result+abs(absAlir-rLBranch[i,8]);
      end;
    end
    else
    begin
      absAlir:=sqrt(sqr(rLBranch[i,11])+sqr(rLBranch[i,12]));
      if absAlir>rLBranch[i,8] then
      begin
        result:=result+abs(absAlir-rLBranch[i,8]);
      end;
    end;
  end;
end;

function FindPinKapQgen(const rLBus:dArr2):double;
var i,sa:integer;
begin
  result:=0;
  sa:=0;
  for i:=0 to high(rLBus) do
  begin
    if rLBus[i,7]<>3 then
    begin
      if rLBus[i,3]>gGen[sa,2] then
      begin
        result:=result+rLBus[i,3]-gGen[sa,2];
      end;
      if rLBus[i,3]<gGen[sa,1] then
      begin
        result:=result+abs(gGen[sa,1])-abs(rLBus[sa,3]);
      end;
      inc(sa);
    end;
  end;
end;

function doHitungFitness(const rChrom:dArr1;
         const rBatasV:TBatas;
         const rLamdaV,rLamdaS:double):double;
var LBus:dArr2;
    Pgen:dArr1;
    pinV:integer;
    pinS,PinQ,Cost:double;
begin
  LBus:=DecodeChrom(rChrom);
  NewtonPolar(LBus,gBranch,gGeneral);
  Pgen:=getPgen(Lbus);
  Cost:=getCostPgen(Pgen);
  pinV:=FindPinV(LBus,rBatasV);
  pinS:=FindPinKapBranch(gBranch);
```

```
  pinQ:=FindPinKapQgen(LBus);
  result:=Cost+rLamdaV*pinV+rLamdaS*pinS+rLamdaS*pinQ;
end;

procedure doHitungFitnessAkhir(const rChrom:dArr1;
          var rLBus,rLBranch:dArr2;
          var rGeneral:TGeneral);
var i,j:integer;
begin
  rLBus:=DecodeChrom(rChrom);
  SetLength(rLBranch,high(gBranch)+1,high(gBranch[0])+1);
  for i:=0 to high(gBranch) do
  begin
    for j:=0 to high(gBranch[0]) do
    begin
      rLBranch[i,j]:=gBranch[i,j];
    end;
  end;
  rGeneral.MaxIterasi:=gGeneral.MaxIterasi;
  rGeneral.Vbase:=gGeneral.Vbase;
  rGeneral.VKonst:=gGeneral.VKonst;
  rGeneral.Pbase:=gGeneral.Pbase;
  rGeneral.PKonst:=gGeneral.PKonst;
  rGeneral.Toleransi:=gGeneral.Toleransi;
  rGeneral.Param:=gGeneral.Param;
  NewtonPolar(rLBus,rLBranch,rGeneral);
end;

end.
```

```
unit uInputLF;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ComCtrls, ExtCtrls, StdCtrls, Grids;

type
  TfrmInput = class(TForm)
    PageControl1: TPageControl;
    Panel1: TPanel;
    TabSheet1: TTabSheet;
    TabSheet2: TTabSheet;
    TabSheet3: TTabSheet;
    btnClose: TButton;
    btnNext: TButton;
    SaveDialog1: TSaveDialog;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    edtNbus: TEdit;
    edtNsal: TEdit;
    edtVbase: TEdit;
    edtPbase: TEdit;
    cmbParam: TComboBox;
    cmbVKonst: TComboBox;
    cmbPkonst: TComboBox;
    fgBus: TStringGrid;
    fgBranch: TStringGrid;
    TabSheet4: TTabSheet;
    fgGen: TStringGrid;
    lbxStatFile: TListBox;
    Label6: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure btnCloseClick(Sender: TObject);
    procedure edtNbusChange(Sender: TObject);
    procedure edtNsalChange(Sender: TObject);
    procedure cmbPkonstChange(Sender: TObject);
    procedure cmbParamChange(Sender: TObject);
    procedure btnNextClick(Sender: TObject);
    procedure TabSheet4Show(Sender: TObject);
    procedure FormActivate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmInput: TfrmInput;

implementation

uses uLoadflow, uNewtonPolar, uUtils, uParamTS;

{$R *.dfm}

procedure TfrmInput.FormCreate(Sender: TObject);
begin
  edtNbus.Text:='1';
  edtNsal.Text:='1';
  edtVbase.Text:='150';
  edtPbase.Text:='100';
  cmbVkonst.Text:='kV';
  cmbPkonst.Text:='MVA';
```

```
  cmbParam.Text:='pu';
  fgBus.Cells[0,0]:='Bus';
  fgBus.Cells[1,0]:='absV (pu)';
  fgBus.Cells[2,0]:='sudV (deg)';
  fgBus.Cells[3,0]:='Pg (MW)';
  fgBus.Cells[4,0]:='Qg (MVAR)';
  fgBus.Cells[5,0]:='PL (MW)';
  fgBus.Cells[6,0]:='QL (MVAR)';
  fgBus.Cells[7,0]:='Cap (pu)';
  fgBus.Cells[8,0]:='Type Bus';
  fgBranch.Cells[0,0] :='No';
  fgBranch.Cells[1,0] :='Dari';
  fgBranch.Cells[2,0] :='Ke';
  fgBranch.Cells[3,0] :='R (pu)';
  fgBranch.Cells[4,0] :='X (pu)';
  fgBranch.Cells[5,0] :='Lc (pu)';
  fgBranch.Cells[6,0] :='Tr';
  fgBranch.Cells[7,0] :='Tu';
  fgBranch.Cells[8,0] :='Su (deg)';
  fgBranch.Cells[9,0] :='Kap (MVA)';
  fgBranch.Cells[10,0]:='P (MW)';
  fgBranch.Cells[11,0]:='Q (MVAR)';
  fgBranch.Cells[12,0]:='Arus re (A)';
  fgBranch.Cells[13,0]:='Arus im (A)';
  fgBranch.Cells[14,0]:='Dari';
  fgBranch.Cells[15,0]:='Ke';
  fgBranch.Cells[16,0]:='P (MW)';
  fgBranch.Cells[17,0]:='Q (MVAR)';
  fgBranch.Cells[18,0]:='Arus re (A)';
  fgBranch.Cells[19,0]:='Arus im (A)';
  fgBranch.Cells[20,0]:='Kap (kVA)';
  fgBranch.Cells[21,0]:='Length (m)';
  fgGen.Cells[0,0]:='No';
  fgGen.Cells[1,0]:='Bus';
  fgGen.Cells[2,0]:='Qmin (MVAR)';
  fgGen.Cells[3,0]:='Qmax (MVAR)';
  fgGen.Cells[4,0]:='a2';
  fgGen.Cells[5,0]:='a1';
  fgGen.Cells[6,0]:='a0';
  fgGen.Cells[7,0]:='Fix Cost';
  fgGen.Cells[8,0]:='Var Cost';
  fgGen.Cells[9,0]:='Pmin (MW)';
  fgGen.Cells[10,0]:='Pmax (MW)';
  lbxStatFile.Items.Clear;
end;

procedure TfrmInput.btnCloseClick(Sender: TObject);
begin
  Close;
end;

procedure TfrmInput.edtNbusChange(Sender: TObject);
var i:integer;
begin
  if edtNbus.Text='' then
  begin
    fgBus.RowCount:=2;
  end
  else
  begin
    try
      fgBus.RowCount:=StrToInt(edtNbus.Text)+1;
      for i:=1 to StrToInt(edtNbus.Text) do
      begin
        fgBus.Cells[0,i]:=IntToStr(i);
        fgBus.Cells[1,i]:='1';
        fgBus.Cells[2,i]:='0';
        fgBus.Cells[3,i]:='0';
```

```
        fgBus.Cells[4,i]:='0';
        fgBus.Cells[5,i]:='0';
        fgBus.Cells[6,i]:='0';
        fgBus.Cells[7,i]:='0';
        fgBus.Cells[8,i]:='3';
      end;
      fgBus.Cells[8,1]:='1';
    except
      raise Exception.Create('Tolong Masukkan angka!');
    end;
  end;
end;

procedure TfrmInput.edtNsalChange(Sender: TObject);
var i:integer;
begin
  if edtNsal.Text='' then
  begin
    fgBranch.RowCount:=2;
  end
  else
  begin
    try
      fgBranch.RowCount:=StrToInt(edtNsal.Text)+1;
      for i:=1 to StrToInt(edtNsal.Text) do
      begin
        fgBranch.Cells[0,i]:=IntToStr(i);
        fgBranch.Cells[3,i]:='0';
        fgBranch.Cells[4,i]:='0';
        fgBranch.Cells[5,i]:='0';
        fgBranch.Cells[6,i]:='0';
        fgBranch.Cells[7,i]:='0';
        fgBranch.Cells[8,i]:='0';
        fgBranch.Cells[9,i]:='10000';
      end;
    except
      raise Exception.Create('Tolong Masukkan angka!');
    end;
  end;
end;

procedure TfrmInput.cmbPkonstChange(Sender: TObject);
begin
  if cmbPKonst.Text='VA' then
  begin
    gGeneral.PKonst:=1;
    fgBus.Cells[3,0]:='Pg (W)';
    fgBus.Cells[4,0]:='Qg (VAR)';
    fgBus.Cells[5,0]:='PL (W)';
    fgBus.Cells[6,0]:='QL (VAR)';
    fgBranch.Cells[9,0] :='Kap (VA)';
    fgBranch.Cells[10,0]:='P (W)';
    fgBranch.Cells[11,0]:='Q (VAR)';
    fgBranch.Cells[16,0]:='P (W)';
    fgBranch.Cells[17,0]:='Q (VAR)';
    fgGen.Cells[9,0]:='Pmin (W)';
    fgGen.Cells[10,0]:='Pmax (W)';
  end
  else if cmbPKonst.Text='kVA' then
  begin
    gGeneral.PKonst:=1000;
    fgBus.Cells[3,0]:='Pg (kW)';
    fgBus.Cells[4,0]:='Qg (kVAR)';
    fgBus.Cells[5,0]:='PL (kW)';
    fgBus.Cells[6,0]:='QL (kVAR)';
    fgBranch.Cells[9,0] :='Kap (VA)';
    fgBranch.Cells[10,0]:='P (kW)';
    fgBranch.Cells[11,0]:='Q (kVAR)';
```

```
    fgBranch.Cells[16,0]:='P (kW)';
    fgBranch.Cells[17,0]:='Q (kVAR)';
    fgGen.Cells[9,0]:='Pmin (kW)';
    fgGen.Cells[10,0]:='Pmax (kW)';
  end
  else if cmbPKonst.Text='MVA' then
  begin
    gGeneral.PKonst:=1000000;
    fgBus.Cells[3,0]:='Pg (MW)';
    fgBus.Cells[4,0]:='Qg (MVAR)';
    fgBus.Cells[5,0]:='PL (MW)';
    fgBus.Cells[6,0]:='QL (MVAR)';
    fgBranch.Cells[9,0] :='Kap (VA)';
    fgBranch.Cells[10,0]:='P (MW)';
    fgBranch.Cells[11,0]:='Q (MVAR)';
    fgBranch.Cells[16,0]:='P (MW)';
    fgBranch.Cells[17,0]:='Q (MVAR)';
    fgGen.Cells[9,0]:='Pmin (MW)';
    fgGen.Cells[10,0]:='Pmax (MW)';
  end;
end;

procedure TfrmInput.cmbParamChange(Sender: TObject);
begin
  if cmbParam.Text='pu' then
  begin
    gGeneral.Param:=psPu;
    fgBranch.Cells[3,0]:='R (pu)';
    fgBranch.Cells[4,0]:='X (pu)';
    fgBranch.Cells[5,0]:='Lc (pu)';
    fgBus.Cells[7,0]:='Cap (pu)';
  end
  else if cmbParam.Text='ohm' then
  begin
    gGeneral.Param:=psOhm;
    fgBranch.Cells[3,0]:='R (ohm)';
    fgBranch.Cells[4,0]:='X (ohm)';
    fgBranch.Cells[5,0]:='Lc (ohm)';
    fgBus.Cells[7,0]:='Cap (ohm)';
  end;
end;

procedure TfrmInput.btnNextClick(Sender: TObject);
var NamaFile,Nama:string;
    input:TextFile;
    NCable,Nbus,Nsal,Ngen,i,typ,dari,ke,param:integer;
    R,X,Lc,Tr,Tu,Su,Cap,CapSal,phi,Pmin,Pmax,Harga,length:double;
    VKonst,PKonst,Vbase,Pbase,absV,sudV,Pg,Qg,PL,QL:double;
    jam,menit,detik,mdetik:word;
    mulai,selesai,selang:TDateTime;
    load:string;
begin
  if btnNext.Caption='&Save' then
  begin
    Nbus:=StrToInt(edtNbus.Text);
    Nsal:=StrToInt(edtNsal.Text);
    Vbase:=StrToFloat(edtVbase.Text);
    VKonst:=1;
    if cmbVKonst.Text='V' then
    begin
      VKonst:=1;
    end
    else if cmbVKonst.Text='kV' then
    begin
      VKonst:=1000;
    end
    else if cmbVKonst.Text='MV' then
    begin
```

```
  VKonst:=1000000;
end;
Pbase:=StrToFloat(edtPbase.Text);
PKonst:=1;
if cmbPKonst.Text='VA' then
begin
  PKonst:=1;
end
else if cmbPKonst.Text='kVA' then
begin
  PKonst:=1000;
end
else if cmbPKonst.Text='MVA' then
begin
  PKonst:=1000000;
end;
Param:=1;
if cmbParam.Text='pu' then
begin
  Param:=1;
end
else if cmbParam.Text='ohm' then
begin
  Param:=2;
end;
try
  if SaveDialog1.Execute then
  begin
    NamaFile:=SaveDialog1.FileName;
    AssignFile(input,NamaFile+'.txt');
    Rewrite(input);
    Writeln(input,Nbus);
    Writeln(input,Nsal);
    Writeln(input,Vbase:6:2);
    Writeln(input,VKonst:7:0);
    Writeln(input,Pbase:6:2);
    Writeln(input,PKonst:7:0);
    Writeln(input,param);
    phi:=4*arctan(1);
    for i:=1 to Nbus do
    begin
      absV:=StrToFloat(fgBus.Cells[1,i]);
      sudV:=StrToFloat(fgBus.Cells[2,i])*180/phi;
      Pg:=StrToFloat(fgBus.Cells[3,i]);
      Qg:=StrToFloat(fgBus.Cells[4,i]);
      PL:=StrToFloat(fgBus.Cells[5,i]);
      QL:=StrToFloat(fgBus.Cells[6,i]);
      Cap:=StrToFloat(fgBus.Cells[7,i]);
      Typ:=StrToInt(fgBus.Cells[8,i]);
      Writeln(input,absV:7:5,' ',sudV:7:5,' ',Pg:9:3,' ',Qg:9:3,
      ' ',PL:9:3,' ',QL:9:3,' ',Cap:7:5,' ',Typ);
    end;
    for i:=1 to Nsal do
    begin
      dari:=StrToInt(fgBranch.Cells[1,i]);
      ke:=StrToInt(fgBranch.Cells[2,i]);
      R:=StrToFloat(fgBranch.Cells[3,i]);
      X:=StrToFloat(fgBranch.Cells[4,i]);
      Lc:=StrToFloat(fgBranch.Cells[5,i]);
      Tr:=StrToFloat(fgBranch.Cells[6,i]);
      Tu:=StrToFloat(fgBranch.Cells[7,i]);
      Su:=StrToFloat(fgBranch.Cells[8,i]);
      CapSal:=StrToFloat(fgBranch.Cells[9,i]);
      Length:=StrToFloat(fgBranch.Cells[21,i]);
      Writeln(input,dari,' ',ke,' ',R:7:5,' ',X:7:5,' ',
      Lc:7:5,' ',Tr:7:5,' ',Tu:7:5,' ',' ',Su:7:5,' ',
      CapSal:7:2,' ',Length:7:2);
    end;
```

```
        Ngen:=0;
        load:='3';
        for i:=1 to Nbus do
        begin
          if fgBus.Cells[8,i]<>load then
          begin
            inc(Ngen);
          end;
        end;
        Writeln(input,Ngen);
        for i:=1 to Ngen do
        begin
          dari:=StrToInt(fgGen.Cells[1,i]);
          R:=StrToFloat(fgGen.Cells[2,i]);
          X:=StrToFloat(fgGen.Cells[3,i]);
          Lc:=StrToFloat(fgGen.Cells[4,i]);
          Tr:=StrToFloat(fgGen.Cells[5,i]);
          Tu:=StrToFloat(fgGen.Cells[6,i]);
          Su:=StrToFloat(fgGen.Cells[7,i]);
          CapSal:=StrToFloat(fgGen.Cells[8,i]);
          Pmin:=StrToFloat(fgGen.Cells[9,i]);
          Pmax:=StrToFloat(fgGen.Cells[10,i]);
          Writeln(input,dari,' ',R:7:2,' ',X:7:2,' ',Lc:7:5,' ',
          Tr:7:5,' ',Tu:7:5,' ',Su:7:2,' ',CapSal:7:2,' ',
          Pmin:7:5,' ',Pmax:7:2);
        end;
      end;
      CloseFile(input);
      MessageDlg('File berhasil disimpan!',mtInformation,[mbOK],0);
    except
      MessageDlg('Tolong dicek angka-angkanya kembali!',
      mtWarning,[mbOK],0);
    end;
  end
  else if btnNext.Caption='&Next' then
  begin
    if gGeneral.PKonst=1 then
    begin
      frmParamTS.fgHasil.Cells[3,0]:='Pg (W)';
      frmParamTS.fgHasil.Cells[4,0]:='Qg (VAR)';
    end
    else if gGeneral.PKonst=1000 then
    begin
      frmParamTS.fgHasil.Cells[3,0]:='Pg (kW)';
      frmParamTS.fgHasil.Cells[4,0]:='Qg (kVAR)';
    end
    else if gGeneral.PKonst=1000000 then
    begin
      frmParamTS.fgHasil.Cells[3,0]:='Pg (MW)';
      frmParamTS.fgHasil.Cells[4,0]:='Qg (MVAR)';
    end;
    frmParamTS.Show;
  end;
end;

procedure TfrmInput.TabSheet4Show(Sender: TObject);
var i,Ngen:integer;
    load:string;
begin
  Ngen:=0;
  load:='3';
  for i:=1 to fgBus.RowCount-1 do
  begin
    if fgBus.Cells[8,i]<>load then
    begin
      inc(Ngen);
    end;
  end;
```

```
    fgGen.RowCount:=Ngen+1;
    Ngen:=0;
    for i:=1 to fgBus.RowCount-1 do
    begin
      if fgBus.Cells[8,i]<>load then
      begin
        inc(Ngen);
        fgGen.Cells[0,Ngen]:=IntToStr(Ngen);
        fgGen.Cells[1,Ngen]:=IntToStr(i);
        if fgGen.Cells[2,Ngen]='' then
        begin
          fgGen.Cells[2,Ngen]:='-1000';
        end;
        if fgGen.Cells[3,Ngen]='' then
        begin
          fgGen.Cells[3,Ngen]:='1000';
        end;
        if fgGen.Cells[4,Ngen]='' then
        begin
          fgGen.Cells[4,Ngen]:='0';
        end;
        if fgGen.Cells[5,Ngen]='' then
        begin
          fgGen.Cells[5,Ngen]:='0';
        end;
        if fgGen.Cells[6,Ngen]='' then
        begin
          fgGen.Cells[6,Ngen]:='0';
        end;
        if fgGen.Cells[7,Ngen]='' then
        begin
          fgGen.Cells[7,Ngen]:='0';
        end;
        if fgGen.Cells[8,Ngen]='' then
        begin
          fgGen.Cells[8,Ngen]:='0';
        end;
        if fgGen.Cells[9,Ngen]='' then
        begin
          fgGen.Cells[9,Ngen]:='0';
        end;
        if fgGen.Cells[10,Ngen]='' then
        begin
          fgGen.Cells[10,Ngen]:='1000';
        end;
      end;
    end;
  end;

procedure TfrmInput.FormActivate(Sender: TObject);
begin
  lbxStatFile.Items.Clear;
end;

end.
```

```
unit uLoadflow;

interface

uses uUtils,uComplex,SysUtils;

type
  TParam=(psPu,psOhm);

  TGeneral=record
    MaxIterasi,Iterasi:integer;
    Vbase,VKonst,Pbase,PKonst,Zbase,Ibase,Toleransi:double;
    Param:TParam;
    sumGen,sumLoad,sumLoss:TComplex;
  end;

var gGeneral:TGeneral;
    gLengthBranch:dArr1;
    gBus,gBranch,gGen:dArr2;

procedure DecodeData(const rBus,rBranch:dArr2;
          var rGeneral:TGeneral;
          var rNbus,rNsal:integer;
          var rE,rF,rPg,rQg,rPL,rQL,rCap:dArr1;
          var rTyp:iArr1;
          var rR,rX,rLc,rTr,rTu,rSu:dArr2);

function FindSumGen(const rNbus:integer;const rTyp:iArr1):integer;
procedure Admitansi(const rNbus:integer;
          const rR,rX,rLc,rTr,rTu,rSu:dArr2;
          const rCap:dArr1;
          var rG,rB:dArr2);
procedure RecToPolar(const rRecRe,rRecIm:dArr1;
          var rPolRe,rPolIm:dArr1);
procedure PolarToRec(const rPolRe,rPolIm:dArr1;
          var rRecRe,rRecIm:dArr1);
procedure AliranDaya(const rNbus:integer;const rE,rF:dArr1;
          const rG,rB,rLc:dArr2;
          var rAlirP,rAlirQ:dArr2);
procedure DayaGen(const rNbus:integer;
          const rE,rF,rPL,rQL:dArr1;
          const rG,rB:dArr2;const rTyp:iArr1;
          var rQg:dArr1);
procedure DayaSlack(const rNbus:integer;
          const rAlirP,rAlirQ:dArr2;
          const rTyp:iArr1;
          const rPL,rQL:dArr1;
          var rPg,rQg:dArr1);
procedure ArusBranch(const rNbus:integer;
          const rE,rF:dArr1;
          const rLc,rG,rB:dArr2;
          var rArusRe,rArusIm:dArr2);
procedure Summary(const rNbus:integer;
          const rPg,rQg,rPL,rQL:dArr1;
          var rSumGen,rSumLoad,rSumLoss:TComplex);
function MaxArray(const rData:dArr1):double;

procedure UpdateBusGen(const rNbus:integer;
          const rE,rF,rPg,rQg:dArr1;
          const rGeneral:TGeneral;
          var rBus:dArr2);
procedure UpdateBranch(const rNsal:integer;
          const rAlirRe,rAlirIm,rArusRe,rArusIm:dArr2;
          const rGeneral:TGeneral;
          var rBranch:dArr2);
function getCostGen(const rNo:integer;
         const rGen:dArr2;
         const rDaya:double):double;
```

```
implementation

procedure DecodeData(const rBus,rBranch:dArr2;
          var rGeneral:TGeneral;
          var rNbus,rNsal:integer;
          var rE,rF,rPg,rQg,rPL,rQL,rCap:dArr1;
          var rTyp:iArr1;
          var rR,rX,rLc,rTr,rTu,rSu:dArr2);
var i,j,dari,ke:integer;
begin
  rGeneral.Zbase:=sqr(rGeneral.Vbase*rGeneral.VKonst)/
                  (rGeneral.Pbase*rGeneral.PKonst);
  rGeneral.Ibase:=rGeneral.Vbase*rGeneral.VKonst/rGeneral.Zbase;
  rNbus:=high(rBus)+1;
  rNsal:=high(rBranch)+1;
  SetLength(rE,rNbus);
  SetLength(rF,rNbus);
  SetLength(rPg,rNbus);
  SetLength(rQg,rNbus);
  SetLength(rPL,rNbus);
  SetLength(rQL,rNbus);
  SetLength(rCap,rNbus);
  SetLength(rTyp,rNbus);
  SetLength(rR,rNbus,rNbus);
  SetLength(rX,rNbus,rNbus);
  SetLength(rLc,rNbus,rNbus);
  SetLength(rTr,rNbus,rNbus);
  SetLength(rTu,rNbus,rNbus);
  SetLength(rSu,rNbus,rNbus);
  for i:=0 to rNbus-1 do
  begin
    rE[i]:=rBus[i,0];
    rF[i]:=rBus[i,1];
    rPg[i]:=rBus[i,2]/rGeneral.Pbase;
    rQg[i]:=rBus[i,3]/rGeneral.Pbase;
    rPL[i]:=rBus[i,4]/rGeneral.Pbase;
    rQL[i]:=rBus[i,5]/rGeneral.Pbase;
    if rGeneral.Param=psPu then
    begin
      rCap[i]:=rBus[i,6];
    end
    else
    begin
      rCap[i]:=rBus[i,6]/rGeneral.Zbase;
    end;
    rTyp[i]:=round(rBus[i,7]);
    for j:=0 to rNbus-1 do
    begin
      rR[i,j]:=0.0;
      rX[i,j]:=0.0;
      rLc[i,j]:=0.0;
      rTr[i,j]:=0.0;
      rTu[i,j]:=0.0;
      rSu[i,j]:=0.0;
    end;
  end;
  for i:=0 to rNsal-1 do
  begin
    dari:=round(rBranch[i,0])-1;
    ke:=round(rBranch[i,1])-1;
    if rGeneral.Param=psPu then
    begin
      rR[dari,ke]:=rBranch[i,2];
      rX[dari,ke]:=rBranch[i,3];
      rLc[dari,ke]:=rBranch[i,4];
      rR[ke,dari]:=rBranch[i,2];
      rX[ke,dari]:=rBranch[i,3];
```

```
      rLc[ke,dari]:=rBranch[i,4];
    end
    else if rGeneral.Param=psOhm then
    begin
      rR[dari,ke]:=rBranch[i,2]/rGeneral.Zbase;
      rX[dari,ke]:=rBranch[i,3]/rGeneral.Zbase;
      rLc[dari,ke]:=rBranch[i,4]/rGeneral.Zbase;
      rR[ke,dari]:=rBranch[i,2]/rGeneral.Zbase;
      rX[ke,dari]:=rBranch[i,3]/rGeneral.Zbase;
      rLc[ke,dari]:=rBranch[i,4]/rGeneral.Zbase;
    end;
    rTr[dari,ke]:=rBranch[i,5];
    rTu[dari,ke]:=rBranch[i,6];
    rSu[dari,ke]:=rBranch[i,7];
  end;
end;

function FindSumGen(const rNbus:integer;
         const rTyp:iArr1):integer;
var i:integer;
begin
  result:=0;
  for i:=0 to rNbus-1 do
  begin
    if rTyp[i]=2 then
    begin
      inc(result);
    end;
  end;
end;

procedure Admitansi(const rNbus:integer;
          const rR,rX,rLc,rTr,rTu,rSu:dArr2;
          const rCap:dArr1;
          var rG,rB:dArr2);
var i,j,k:integer;
    a,b:double;
    Cr,Ci:dArr2;
    sum,Ca,Za:TComplex;
begin
  SetLength(Cr,rNbus,rNbus);
  SetLength(Ci,rNbus,rNbus);
  sum:=TComplex.Create(1.0,0.0);
  for i:=0 to rNbus-1 do
  begin
    for j:=0 to rNbus-1 do
    begin
      if rX[i,j]<>0 then
      begin
        Za:=TComplex.Create(rR[i,j],rX[i,j]);
        Ca:=sum.Divide(Za);
        Cr[i,j]:=Ca.Real;
        Ci[i,j]:=Ca.Imag;
        Ca.Free;
        Za.Free;
      end
      else
      begin
        Cr[i,j]:=0.0;
        Ci[i,j]:=0.0;
      end;
    end;
  end;
  sum.Free;
  SetLength(rG,rNbus,rNbus);
  SetLength(rB,rNbus,rNbus);
  for i:=0 to rNbus-1 do
  begin
```

```
  for j:=0 to rNbus-1 do
  begin
    if j=i then
    begin
      rG[i,j]:=0.0;
      rB[i,j]:=0.0;
      for k:=0 to rNbus-1 do
      begin
        rG[i,j]:=rG[i,j]+Cr[i,k];
        rB[i,j]:=rB[i,j]+Ci[i,k]+rLc[i,k];
      end;
    end
    else
    begin
      rG[i,j]:=-Cr[i,j];
      rB[i,j]:=-Ci[i,j];
    end;
  end;
end;
for i:=0 to rNbus-1 do
begin
  for j:=0 to rNbus-1 do
  begin
    if rTr[i,j]<>0 then
    begin
      rG[i,i]:=rG[i,i]-Cr[i,j];
      rB[i,i]:=rB[i,i]-Ci[i,j]-rLc[i,j];
      rG[i,i]:=rG[i,i]+Cr[i,j]/sqr(rTr[i,j]);
      rB[i,i]:=rB[i,i]+Ci[i,j]/sqr(rTr[i,j]);
      rG[j,j]:=rG[j,j]-Cr[i,j];
      rB[j,j]:=rB[j,j]-Ci[i,j]-rLc[i,j];
      rG[j,j]:=rG[j,j]+Cr[i,j];
      rB[j,j]:=rB[j,j]+Ci[i,j];
      rG[i,j]:=-1*Cr[i,j]/rTr[i,j];
      rB[i,j]:=-1*Ci[i,j]/rTr[i,j];
      rG[j,i]:=rG[i,j];
      rB[j,i]:=rB[i,j];
      {CLc:=TComplex.Create(0.0,aLc[i,j]);
      result[i,i]:=result[i,i]-Cx[i,j]-CLc;
      result[i,i]:=result[i,i]+Cx[i,j]/sqr(at);
      result[j,j]:=result[j,j]-Cx[i,j]-CLc;
      result[j,j]:=result[j,j]+Cx[i,j];
      result[i,j]:=-Cx[i,j]/at;
      result[j,i]:=result[i,j];
      CLc.Free;}
    end;
  end;
end;
for i:=0 to rNbus-1 do
begin
  for j:=0 to rNbus-1 do
  begin
    if rTu[i,j]<>0 then
    begin
      a:=rTu[i,j]*cos(rSu[i,j]);
      b:=rTu[i,j]*sin(rSu[i,j]);
      rG[i,i]:=rG[i,i]-Cr[i,j];
      rB[i,i]:=rB[i,i]-Ci[i,j]-rLc[i,j];
      rG[i,i]:=rG[i,i]+Cr[i,j]/(sqr(a)+sqr(b));
      rB[i,i]:=rB[i,i]+Ci[i,j]/(sqr(a)+sqr(b))+rLc[i,j];
      Za:=TComplex.Create(Cr[i,j],Ci[i,j]);
      Za:=Za.Negative;
      Ca:=TComplex.Create(a,b);
      sum:=Za.Divide(Ca);
      rG[j,i]:=sum.Real;
      rB[j,i]:=sum.Imag;
      sum.Free;
      Ca:=Ca.Conj;
```

```
        sum:=Za.Divide(Ca);
        rG[i,j]:=sum.Real;
        rB[i,j]:=sum.Imag;
        sum.Free;
        Ca.Free;
        Za.Free;
        {sUpfc:=TComplex.Create(aUpfc[i,j].tap*cos(aUpfc[i,j].sudut),
              aUpfc[i,j].tap*sin(aUpfc[i,j].sudut));
        CLc:=TComplex.Create(0.0,aLc[i,j]);
        result[i,i]:=result[i,i]-Cx[i,j]-CLc;
        result[i,i]:=result[i,i]+Cx[i,j]/sqr(sUpfc.Abs)+CLc;
        result[i,j]:=-Cx[i,j]/conj(sUpfc);
        result[j,i]:=-Cx[i,j]/sUpfc;
        CLc.Free;
        sUpfc.Free;}
      end;
    end;
  end;
  for i:=0 to rNbus-1 do
  begin
    if rCap[i]<>0 then
    begin
      rB[i,i]:=rB[i,i]+rCap[i];
    end;
  end;
end;

procedure RecToPolar(const rRecRe,rRecIm:dArr1;
          var rPolRe,rPolIm:dArr1);
var i:integer;
begin
  if high(rRecRe)<>high(rRecIm) then
  begin
    raise Exception.Create('Dimensi kedua Vector tidak sama!');
  end;
  SetLength(rPolRe,high(rRecRe)+1);
  SetLength(rPolIm,high(rRecRe)+1);
  for i:=0 to high(rRecRe) do
  begin
    rPolRe[i]:=sqrt(sqr(rRecRe[i])+sqr(rRecIm[i]));
    rPolIm[i]:=arctan(rRecIm[i]/rRecRe[i]);
  end;
end;

procedure PolarToRec(const rPolRe,rPolIm:dArr1;
          var rRecRe,rRecIm:dArr1);
var i:integer;
begin
  if high(rRecRe)<>high(rRecIm) then
  begin
    raise Exception.Create('Dimensi kedua Vector tidak sama!');
  end;
  SetLength(rRecRe,high(rPolRe)+1);
  SetLength(rRecIm,high(rPolRe)+1);
  for i:=0 to high(rPolRe) do
  begin
    rRecRe[i]:=rPolRe[i]*cos(rPolIm[i]);
    rRecIm[i]:=rPolRe[i]*sin(rPolIm[i]);
  end;
end;

procedure AliranDaya(const rNbus:integer;const rE,rF:dArr1;
          const rG,rB,rLc:dArr2;
          var rAlirP,rAlirQ:dArr2);
var i,j:integer;
    Vi,Vj,Ya,Lca,Sa,tmp1,tmp2:TComplex;
begin
  SetLength(rAlirP,rNbus,rNbus);
```

```
  SetLength(rAlirQ,rNbus,rNbus);
  for i:=0 to rNbus-1 do
  begin
    Vi:=TComplex.Create(rE[i],rF[i]);
    for j:=0 to rNbus-1 do
    begin
      rAlirP[i,j]:=0.0;
      rAlirQ[i,j]:=0.0;
      if j<>i then
      begin
        if rB[i,j]<>0 then
        begin
          Vj:=TComplex.Create(rE[j],rF[j]);
          Ya:=TComplex.Create(rG[i,j],rB[i,j]);
          Lca:=TComplex.Create(0.0,rLc[i,j]);
          tmp1:=Vi.Conj.Multiply(Vi.Subtract(Vj)).Multiply(Ya.Negative);
          tmp2:=Vi.Conj.Multiply(Vi).Multiply(Lca);
          Sa:=tmp1.Add(tmp2);
          rAlirP[i,j]:=Sa.Real;
          rAlirQ[i,j]:=-Sa.Imag;
          //result[i,j]:=conj(aV[i])*(aV[i]-aV[j])*(-aY[i,j])+
          //             conj(aV[i])*aV[i]*dLc;
          //result[i,j]:=conj(result[i,j]);
          Sa.Free;
          tmp2.Free;
          tmp1.Free;
          Lca.Free;
          Ya.Free;
          Vj.Free;
        end;
      end;
    end;
    Vi.Free;
  end;
end;

procedure DayaGen(const rNbus:integer;
          const rE,rF,rPL,rQL:dArr1;
          const rG,rB:dArr2;const rTyp:iArr1;
          var rQg:dArr1);
var i,j:integer;
    sum:double;
begin
  for i:=0 to rNbus-1 do
  begin
    sum:=0.0;
    if rTyp[i]=2 then
    begin
      for j:=0 to rNbus-1 do
      begin
        //sum:=sum+(Fi*(Ej*Gij+Fj*-Bij)-Ei*(Fj*Gij-Ej*-Bij));
        sum:=sum+(rF[i]*(rE[j]*rG[i,j]+rF[j]*-rB[i,j])-
                  rE[i]*(rF[j]*rG[i,j]-rE[j]*-rB[i,j]));
      end;
      rQg[i]:=sum-rQL[i];
    end;
  end;
end;

procedure DayaSlack(const rNbus:integer;
          const rAlirP,rAlirQ:dArr2;
          const rTyp:iArr1;
          const rPL,rQL:dArr1;
          var rPg,rQg:dArr1);
var i,j:integer;
    sumP,sumQ:double;
begin
  for i:=0 to rNbus-1 do
```

```
  begin
    if rTyp[i]=1 then
    begin
      sumP:=0.0;
      sumQ:=0.0;
      for j:=0 to rNbus-1 do
      begin
        if rAlirQ[i,j]<>0 then
        begin
          sumP:=sumP+rAlirP[i,j];
          sumQ:=sumQ+rAlirQ[i,j];
        end;
      end;
      rPg[i]:=sumP+rPL[i];
      rQg[i]:=sumQ+rQL[i];
    end;
  end;
end;

procedure ArusBranch(const rNbus:integer;
          const rE,rF:dArr1;
          const rLc,rG,rB:dArr2;
          var rArusRe,rArusIm:dArr2);
var i,j:integer;
    Vi,Vj,Ya,xLc,Arus,tmp1,tmp2:TComplex;
begin
  SetLength(rArusRe,rNbus,rNbus);
  SetLength(rArusIm,rNbus,rNbus);
  for i:=0 to rNbus-1 do
  begin
    Vi:=TComplex.Create(rE[i],rF[i]);
    for j:=0 to rNbus-1 do
    begin
      if rB[i,j]<>0 then
      begin
        Vj:=TComplex.Create(rE[j],rF[j]);
        Ya:=TComplex.Create(rG[i,j],rB[i,j]);
        xLc:=TComplex.Create(0,rLc[i,j]);
        tmp1:=Ya.Negative.Multiply(Vi.Subtract(Vj));
        tmp2:=Vi.Multiply(xLc);
        Arus:=tmp1.Add(tmp2);
        rArusRe[i,j]:=Arus.Real;
        rArusIm[i,j]:=-Arus.Imag;
        Arus.Free;
        tmp2.Free;
        tmp1.Free;
        xLc.Free;
        Ya.Free;
        Vj.Free;
        //result[i,j]:=(aV[i]-aV[j])*(-aY[i,j])+aV[i]*xLc;
        //result[i,j]:=Conj(result[i,j]);
      end
      else
      begin
        rArusRe[i,j]:=0.0;
        rArusIm[i,j]:=0.0;
      end;
    end;
    Vi.Free;
  end;
end;

procedure Summary(const rNbus:integer;
          const rPg,rQg,rPL,rQL:dArr1;
          var rSumGen,rSumLoad,rSumLoss:TComplex);
var i:integer;
    sumPg,sumPL,sumQg,sumQL:double;
begin
```

```
  sumPg:=0.0;
  sumQg:=0.0;
  sumPL:=0.0;
  sumQL:=0.0;
  for i:=0 to rNbus-1 do
  begin
    sumPg:=sumPg+rPg[i];
    sumQg:=sumQg+rQg[i];
    sumPL:=sumPL+rPL[i];
    sumQL:=sumQL+rQL[i];
  end;
  rSumGen:=TComplex.Create(sumPg,sumQg);
  rSumLoad:=TComplex.Create(sumPL,sumQL);
  rSumLoss:=rSumGen.Subtract(rSumLoad);
end;

function MaxArray(const rData:dArr1):double;
var i:integer;
begin
  result:=abs(rData[0]);
  for i:=1 to high(rData) do
  begin
    if result<abs(rData[i]) then
    begin
      result:=abs(rData[i]);
    end;
  end;
end;

procedure UpdateBusGen(const rNbus:integer;
          const rE,rF,rPg,rQg:dArr1;
          const rGeneral:TGeneral;
          var rBus:dArr2);
var i:integer;
begin
  for i:=0 to rNbus-1 do
  begin
    rBus[i,0]:=rE[i];
    rBus[i,1]:=rF[i];
    rBus[i,2]:=rPg[i]*rGeneral.Pbase;
    rBus[i,3]:=rQg[i]*rGeneral.Pbase;
  end;
end;

procedure UpdateBranch(const rNsal:integer;
          const rAlirRe,rAlirIm,rArusRe,rArusIm:dArr2;
          const rGeneral:TGeneral;
          var rBranch:dArr2);
var i,dari,ke:integer;
begin
  for i:=0 to rNsal-1 do
  begin
    dari:=round(rBranch[i,0])-1;
    ke:=round(rBranch[i,1])-1;
    rBranch[i,9] :=rAlirRe[dari,ke]*rGeneral.Pbase;
    rBranch[i,10]:=rAlirIm[dari,ke]*rGeneral.Pbase;
    rBranch[i,11]:=rAlirRe[ke,dari]*rGeneral.Pbase;
    rBranch[i,12]:=rAlirIm[ke,dari]*rGeneral.Pbase;
    rBranch[i,13]:=rArusRe[dari,ke]*rGeneral.Ibase;
    rBranch[i,14]:=rArusIm[dari,ke]*rGeneral.Ibase;
    rBranch[i,15]:=rArusRe[ke,dari]*rGeneral.Ibase;
    rBranch[i,16]:=rArusIm[ke,dari]*rGeneral.Ibase;
  end;
end;

function getCostGen(const rNo:integer;
         const rGen:dArr2;
         const rDaya:double):double;
```

```
begin
  result:=0;
  if rDaya>0 then
  begin
    result:=rGen[rNo,3]*sqr(rDaya)+rGen[rNo,4]*rDaya+rGen[rNo,5];
  end;
end;

end.
```