TUGAS AKHIR – TE 141599

PENGEMBANGAN *SOFTWARE ECONOMIC DISPATCH* (ED) MENGGUNAKAN *APPROXIMATION METHODS* DENGAN MEMPERHITUNGKAN RUGI – RUGI TRANSMISI

Silsilatil Maghfiroh
NRP 2213106050

Dosen Pembimbing
Prof. Ir. Ontoseno Penangsang, M.Sc, Ph.D
Dr. Rony Seto Wibowo, ST., MT.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2016

FINAL PROJECT – TE 141599

ECONOMIC DISPATCH (ED) SOFTWARE DEVELOPMENT USING APPROXIMATION METHODS WITH TRANSMISSION LOSSES CONSIDERATION

Silsilatil Maghfiroh
NRP 2213106050

Advisor
Prof. Ir. Ontoseno Penangsang, M.Sc, Ph.D
Dr. Rony Seto Wibowo, ST., MT.

ELECTRICAL ENGINEERING DEPARTEMENT
Faculty of Industrial Technology
InstitutTeknologiSepuluhNopember
Surabaya 2016

PENGEMBANGAN *SOFTWARE ECONOMIC DISPATCH* (ED) MENGGUNAKAN *APPROXIMATION METHODS* DENGAN MEMPERHITUNGKAN RUGI – RUGI TRANSMISI

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Teknik Sistem Tenaga
Jurusan Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui:

Dosen Pembimbing I,	Dosen Pembimbing II,
Prof. Ir. Ontoseno Penangsang, M. Sc, Ph. D.	Dr. Rony Seto Wibowo, ST., MT.
NIP. 1949 07 15 1974 12 1001	NIP. 1974 11 29 2000 12 1001

JURUSAN
TEKNIK ELEKTRO

SURABAYA
JANUARI, 2016

ABSTRAK

Pengembangan Software Economic Dispatch (ED) Menggunakan *Approximation Methods* dengan Memperhitungkan Rugi-rugi Transmisi

Silsilatil Maghfiroh
2213106050

Dosen pembimbing 1 : Prof. Ir. Ontoseno Penangsang, M.Sc, Ph.D
Dosen Pembimbing 2 : Dr. Rony Seto Wibowo, ST., MT.

Abstrak

Permasalahan *economic dispatch* pada pembangkit yaitu mengenai perubahan permintaan daya. Perubahan beban yang berubah setiap periode waktu tertentu menyebabkan perubahan perhitungan *economic dispatch* untuk setiap harga tertentu. Selain perubahan permintaan daya, pertimbangan lain yang perlu diperhatikan yaitu mengenai rugi – rugi transmisi. *Approximation Methods* digunakan untuk meminimalkan biaya pembangkitan dengan permintaan total pembebanan yang berbeda di setiap unit pembangkitan. *Approximation Methods* ini merupakan metode perkiraan *(non-convex)* dari fungsi biaya pembangkitan yang dikembangkan untuk menyelesaikan permasalahan dengan rugi – rugi transmisi. Metode ini akan diterapkan pada aplikasi pemrograman, serta referensi plant akan digunakan sebagi uji coba aplikasi software perhitungan ED yang menjadi tujuan dari pembuatan tugas akhir. Hasil dari simulasi tugas akhir diperoleh bahwa untuk biaya pembangkitan bisa lebih murah +/- 370 $/HR.

Kata Kunci :*Economic dispatch, Rugi-rugi Transmisi, Approximation Methods.*

ABSTRACT

Economic Dispatch (ED) Software Development Using Approximation Methods With Transmission Losses Consideration

Silsilatil Maghfiroh
2213106050

Advisor 1 : Prof. Ir. Ontoseno Penangsang, M.Sc, Ph.D
Advisor 2 : Dr. Rony Seto Wibowo, ST., MT.

Abstract

Economic dispatch problems at the reactor which is the power change demand .Change the load that it is changing every a certain period of time causing change economic dispatch calculation for every a certain price .In addition to demand change resources , other consideration that need to be considered which is the compensation compensation transmission .Approximation methods used to minimize the cost stirring up with the demand total imposition of differing in every unit the generation of .Approximation methods this is a method of approximate (non-convex) of the function of the cost of the generation of being developed to solve problems with the transmission of compensation compensation .This method to be applied on the application programming , as well as reference plant will be used for the trial of a software application calculation ed be a goal for the end of the manufacture of duty . The result of the end of the simulation duty obtained that for the cost of the generation of can it be cheaper + / - 370 $/HR.

Index Term: Economic Dispatch, Transmission Losses, Approximation Methods.

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT atas segala rahmat, karunia, dan petunjuk yang telah dilimpahkan-Nya sehingga penulis mampu menyelesaikan tugas akhir dengan judul :

Pengembangan Software Economic Dispatch (ED) Menggunakan *Approximation Methods* dengan Memperhitungkan Rugi-rugi Transmisi

Tugas akhir ini disusun sebagai salah satu persyaratan untuk menyelesaikan jenjang pendidikan S1 pada Bidang Studi Teknik Sistem Tenaga, Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember.

Pada kesempatan ini penulis hendak menyampaikan rasa terima kasih kepada pihak-pihak yang memberikan peranan penting dalam menyelesaikan tugas akhir ini, kepada:

1. Allah SWT atas limpahan Rahmat dan Petunjuk-Nya serta Nabi Muhammad SAW atas tuntunan jalan-Nya.
2. Ayah, ibu dan seluruh keluarga yang selalu mendukung penulis dalam meyelesaikan studi.
3. Prof. Ir. Ontoseno Penangsang, M.Sc, Ph.D dan Dr. Rony Seto Wibowo, ST., MT. sebagai dosen pembimbing yang telah memberikan arahan dan perhatiannya dalam tugas akhir ini.
4. Seluruh dosen yang telah memberikan ilmunya selama kuliah, karyawan, dan keluarga besar Jurusan teknik Elektro ITS.
5. Teman-teman Teknik Elektro Lintas Jalur ITS 2013 Genap dan khususnya kepada teman-teman satu kelompok TA atas bantuan kalian selama masa pengerjaan tugas akhir ini.
6. Semua pihak yang telah membantu baik secara langsung maupun tidak langsung, yang tidak mungkin saya sebutkan satu per satu.

Penulis menyadari bahwa Tugas Akhir ini belum sempurna, Oleh karena itu saran dan masukan sangat diharapkan untuk perbaikan dimasa yang akan datang.

Surabaya, Januari 2016

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

Halaman Ini Sengaja Dikosongkan

BAB I
PENDAHULUAN

1.1 Latar Belakang

Economic dispatch merupakan proses pembagian pembebanan dari pembangkit yang ada dalam sistem yang beroperasi secara optimal ekonomi pada beban sistem tertentu, sehingga diperoleh kombinasi unit pembangkit yang dapat memenuhi kebutuhan beban dengan biaya yang optimum. Perhitungan pembebanan pembangkit sangat dibutuhkan oleh perusahaan untuk mendapatkan pembebanan yang optimal. Permasalahan *economic dispatch* pada pembangkit yaitu mengenai perubahan permintaan daya. Perubahan beban yang berubah setiap periode waktu tertentu menyebabkan perubahan perhitungan *economic dispatch* untuk setiap harga tertentu. Selain perubahan permintaan daya, pertimbangan lain yang perlu diperhatikan yaitu mengenai rugi – rugi transmisi.

Dari permasalahan tersebut, maka digunakan *Approximation Methods* untuk meminimalkan biaya pembangkitan dengan permintaan total pembebanan yang berbeda di setiap unit pembangkitan. *Approximation Methods* ini merupakan metode perkiraan *(non-convex)* dari fungsi biaya pembangkitan yang dikembangkan untuk menyelesaikan permasalahan dengan rugi – rugi transmisi. Perkiraan ini menggunakan *Gradient* dan *Newton Technique* untuk menyelesaikan permasalahan *economic dispatch* dengan *valve point loading effect* dan rugi – rugi transmisi dengan analisa pendekatan.

Penggunaan software yang umum digunakan adalah MATLAB untuk penyelesaian ED, namun tidak semua pengguna bisa menggunakan script yang ada pada MATLAB. Maka pada tugas akhir ini akan diusulkan menggunakan Delphi. Dengan menggunakan Delphi diharapkan bisa mempermudah dalam penggunaan kode program, kompilasi cepat, penggunaan file unit ganda untuk pemrograman modular, pengembangan pada perangkat lunak, pola desain bisa lebih menarik, dan dengan bahasa pemrograman lebih terstruktur dengan bahasa pemrograman Object Pascal. Tujuan dari tugas akhir ini yaitu mendapatkan aplikasi software perhitungan ED yang bisa dan dengan mudah diinteraksikan dengan pengguna.

Teknik Elektro ITS sudah memiliki program aplikasi perhitungan ED karya sendiri berbasis Delphi yang bernama PowerGen. Program aplikasi ini memiliki beberapa fungsi seperti perhitungan ED dengan *Losses*, perhitungan ED dengan linear cost function, dsb. Namun pada aplikasi ini akan ditambah 1 metode yaitu *Approximation Methods*.

1.2 Permasalahan

Penelitian pada tugas akhir ini bertujuan untuk mendapatkan aplikasi perhitungan ED dan pengembangan software pada aplikasi PowerGen. Serta meminimalkan biaya pembangkitan, dan mengalokasikan distribusi daya pada masing – masing unit secara optimal dengan memperhitungkan rugi – rugi transmisi.

1.3 Tujuan

Permasalahan yang akan dibahas dalam tugas akhir ini adalah :

1. Apa metode optimasi *Economic dispatch* yang dapat diterapkan pada Delphi
2. Bagaimana *User Interface* yang baik dan cocok untuk perhitungan *Economic dispatch.*
3. Bagaimana kinerja(*performance*) dari software yang dihasilkan dengan menggunakan metode *Approximation Methods*

1.4 Batasan Masalah

Agar tugas akhir ini tidak menyimpang dari ketentutan yang digariskan maka diambil batasan dan asumsi sebagai berikut :

1. Dari *software* ED yang sudah ada hanya dikembangkan perhitungan dengan metode *Approximation* dengan memperhitungkan *losses* yang sudah ideal.
2. Tipe kurva pembangkit yang dibahas hanya tipe kurva polynomial orde 2.

1.5 Metodologi

Alur metodologi penyelesaian tugas akhir ini adalah sebagai berikut :

1. Studi pustaka
   Studi pustaka dilakukan untuk mengumpulkan buku-buku maupun jurnal yang berkaitan tentang topik tugas akhir yang dibahas. Pustaka-pustaka yang dikumpulkan mencakup *Economic dispatch*, *Approximation Methods*, *Rugi-rugi Transmisi* serta buku pemrograman Delphi.
2. Pengenalan *software* dan ekperimen
   Pengenalan *software* dilakukan dengan mempelajari *software* yang akan dikembangkan disertai melakukan eksperimen-ekperimen untuk mengetahui bagaimana cara kerja *software* tersebut.
3. Penerapan batasan jumlah pembangkit dan *losses*kedalam *software*

Dalam tahap ini jumlah pembangkit dan *losses* diterapkan kedalam *software* hingga dapat menghasilkan *output* dan tidak ada *error*yang terjadi.

4. Pengujian awal dan *troubleshooting* terhadap *bug* yang muncul
   Pengujian awal dilakukan untuk mencari *bug*/kesalahandalam software. Pengujian ini lebih mengutamakan kelancaran penggunaan *software* sehingga diuji dengan data-data sederhana namun beragam.
5. Pengujian akhir dan finalisasi *software*
   Pengujian akhir dilakukan untuk memantapkan kembali kinerja *software* yang telah dikembangkan. Pada pengujian ini lebih mengutamakan kinerja software dalam melakukan perhitungan sehingga diuji dengan data-data yang lebih kompleks.
6. Pembuatan laporan tugas akhir
   Melakukan penulisan laporan yang menunjukkan hasil akhir dari tugas akhir.

1.6 Sistematika Pembahasan

Sistematika penulisan laporan tugas akhir ini dibagi menjadi lima bab dengan masing-masing bab diuraikan sebagai berikut :

1. BAB 1 merupakan pendahuluan yang berisi latar belakang, permasalahan, tujuan, metodologi, batasan masalah dan sistematika penulisan.
2. BAB 2 berisi teori penunjang yang membahas tentang Sistem kelistrikan, *Economic dispatch*, *Approximation Methods, Rugi-rugi Transmisi*, serta dasar-dasar permrograman Delphi
3. BAB 3 berisi tentang uraian perencanaan, pembuatan, dan implementasi kedalam *software* yang dikembangkan.
4. BAB 4 berisi tentang hasil pengujian perangkaat lunak yang telah dirancang.
5. BAB 5 berisi tentang kesimpulan dan saran-saran dari pembuatan sampai pengimplementasian perangkat lunak.

1.7 Relevansi dan Manfaat

Hasil yang diperoleh dari Tugas Akhir ini diharapkan dapat memberikan manfaat sebagai berikut :

1. Bagi perusahaan listrik
   Tugas akhir ini diharapkan dapat memberikan manfaat bagi perusahaan listrik dalam memutuskan pola pembangkitan yang dilakukan sehingga mendapatkan biaya pembangkitan yang lebih baik.

2. Bagi bidang ilmu pengetahuan dan mahasiswa lain
   Tugas akhir ini diharapkan dapat membantu perkembangan ilmu pengetahuan dengan menjadi alat bantu perhitungan ED yang handal dan mudah digunakan. Sehingga bisa lebih mudah memahami penggunaan aplikasi tersebut.

BAB II
ECONOMIC DISPATCH

1.1 Sistem Pembangkitan Tenaga Listrik [6]

Secara umum sistem pembangkitan merupakan kumpulan dari unit pembangkit tenaga listrik yang terdiri dari beberapa komponen utama seperti turbin dan generator. Pembangkit tenaga listrik digunakan untuk membangkitkan daya listrik yang kemudian didistribusikan kepada konsumen. Di dalam sebuah sistem pembangkit, beberapa generator dioperasikan secara paralel dan dihubungkan dengan bus dalam suatu sistem tenaga listrik guna menyediakan total daya yang diperlukan.

Pembangkit tenaga listrik dapat dibedakan menjadi beberapa jenis sesuai dengan bahan bakar yang digunakan. Salah satu diantaranya adalah pembangkit listrik tenaga panas atau *thermal*. Pembangkit tipe ini merupakan pembangkit listrik yang mayoritas digunakan untuk memenuhi beban harian atau *base load*.

Setiap pembangkit memiliki karaktersitik unit pembangkit masing-masing. Karakteristik unit pembangkit meliputi karaktersitik *input-output* pembangkit, dan karakteristik *incremental rate*. Karakteristik tersebut diperoleh dari data-data seperti : desain generator; pabrik pembuat generator; data historis pengoperasian generator; maupun data percobaan. Karakteristik unit pembangkit digunakan dalam perhitungan biaya pembangkitan dari tiap unit pembangkit sehingga dapat dicapai nilai ekonomis atau nilai optimum.

Karakteristik *input-output* dari pembangkit dari pembangkit *thermal* merupakan hubungan antara *input* berupa bahan bakar yang digunakan dengan *output* berupa daya yang dibangkitkan tiap pembangkit. *Input* bahan bakar dinyatakan dalam bentuk MBtu/h atau konsumsi energi sedangkan output daya dinyatakan dalam bentuk MW atau daya yang dibangkitkan.

Karakteristik *incremental rate* pembangkit thermal merupakan hubungan antara perubahan daya pembangkitan yang dihasilkan dengan konsumsi bahan bakar yang dibutuhkan. *Incremental rate* biasanya dinyatakan dengan satuan Btu/kWh.

1.2 Karakteristik Pembangkit Thermal[6]

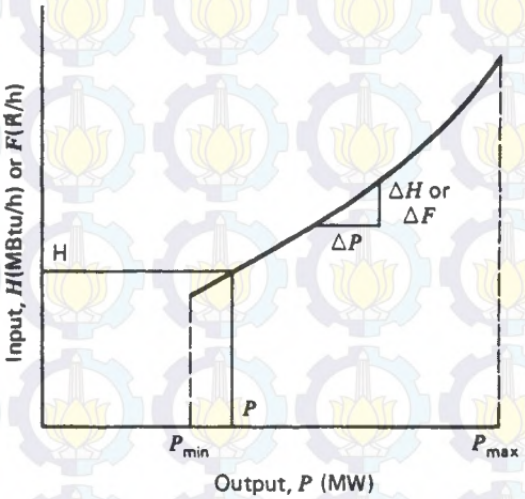
Ada banyak parameter dalam analisis pengaturan operasi sistem tenaga. Hal yang paling mendasar dalam masalah operasi ekonomis adalah karakteristik *input-output* pada pembangkit *thermal*. Untuk menggambarkan karakteristik *input-output*, *input* merepresentasikan sebagai masukan total yang diukur dalam satuan biaya/jam dan *output*

merupakan daya keluaran listrik yang disediakan oleh sistem pembangkit tenaga listrik. Dalam menggambarkan karakteristik unit turbin uap, akan menggunakan *termionologi* (2.1) dan (2.2) sebagai berikut:

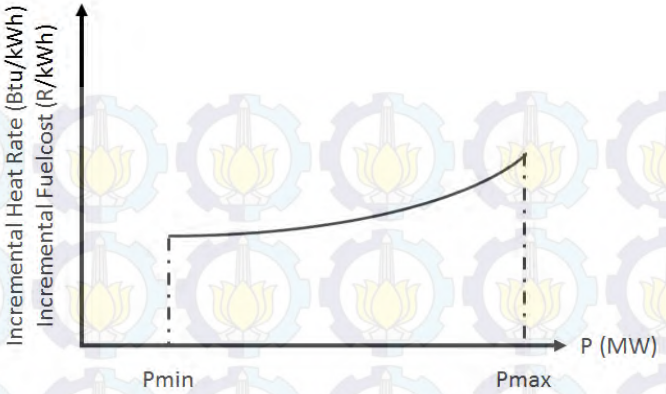
$$H = \frac{Mbtu}{jam} \tag{2.1}$$

$$F = \frac{R}{jam} \tag{2.2}$$

H dapat dinyatakan sebagai energi panas yang dibutuhkan tiap jam dan F dinyatakan sebagai biaya tiap jam. Ada kalanya R/jam biaya operasional suatu unit terdiri dari biaya operasional dan biaya pemeliharaan. Biaya karyawan akan dimasukkan sebagai bagian dari biaya operasi jika biaya ini dapat digambarkan secara langsung sebagai fungsi dari *output* unit. *Output* dari unit pembangkit dinyatakan dengan P dalam Megawatt. Untuk lebih jelasnya dapat dilihat pada Gambar 2.1 dan Gambar 2.2



Gambar 2.1 Kurva input-output pembangkit thermal



Gambar 2.2 Kurva incremental pembangkit thermal

Karakteristik *input-output* dari unit pembangkit *thermal* yang ideal, digambarkan sebagai kurva nonlinear yang kontinyu. Data karakteristik input output diperoleh dari perhitungan desain atau. Pembangkit *thermal* mempunyai batas operasi *minimum* (Pmin) dan *maksimum* (Pmax). Batasan beban minimum biasanya disebabkan oleh kestabilan pembakaran dan masalah desain generator. Pada umumnya unit pembangkit thermal tidak dapat beroperasi dibawah 30% dari kapasitas desain.

Selanjutnya akan dibahas lebih lanjut tentang pemodelan karakteristik *input-output* maupun karakteristik *incremental rate*. Ada dua macam pendekatan dalam memodelkan karakteristik-karakteristik tersebut. Yang paling umum ditemui adalah pemodelan dengan fungsi polinomial. Namun disamping itu tidak jarang juga kita temui bentuk fungsi *piecewise*

1.2.1 Pemodelan Fungsi Polinomial (Continuous)

Bentuk pemodelan fungsi polinomial adalah pendekatan dari kurva *input-output* dengan fungsi polinomial. Fungsi polinomial yang umum digunakan adalah kurva polinomial orde dua. Namun meski begitu tidak menutup kemungkinan bila nantinya ada pendekatan dengan fungsi polinomial dengan orde lebih dari dua.

Sebagai contoh dari gambar 2.3 kita dapat membuat pendekatan fungsi polinomial orde dua seperti pada persamaan 2.3

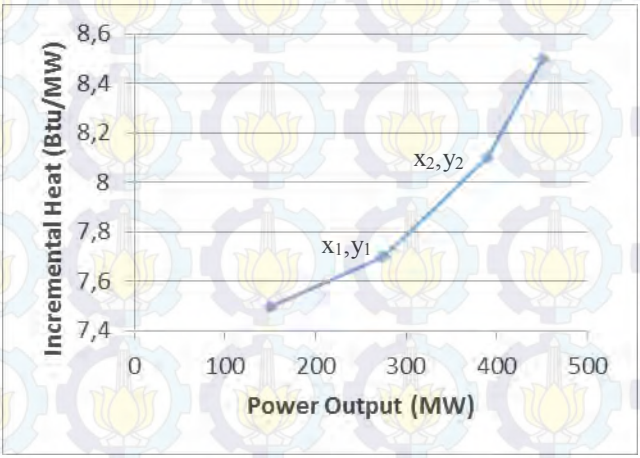
$$H(P) = aP^2 + bP + c \qquad (2.3)$$

Sedangkan fungsi *incremental rate* nya bisa kita dapatkan dari turunan pertama fungsi *input-output*.

$$ihr(P) = \frac{\delta H(P)}{\delta P} = 2aP + b \quad (2.4)$$

1.2.2 Pemodelan Piecewise Incremental Heat

Dalam ilmu matematika, fungsi *piecewise* adalah fungsi yang didefinisikan oleh sub fungsi yang digunakan pada interval/segmen yang berbeda. Pemodelan bentuk ini menyajikan serangkaian set data dari kurva *incremental* heat yang kemudian dapat kita definisikan ke dalam bentuk polinomial untuk setiap interval/segmen. Penjelasannya akan lebih mudah jika kita mengamati Gambar 2.3



Gambar 2.3 Contoh kurva piecewise incremental rate

Untuk segmen antara titik (x_1,y_1) dan (x_2,y_2) dapat kita bentuk persamaan polinomialnya

$$ihr(P) = \propto P + \beta \quad (2.5)$$

Dimana

$$\propto = \frac{y_2 - y_1}{x_2 - x_1} \quad (2.6)$$

$$\beta = \frac{x_2 y_1 - x_1 y_2}{x_2 - x_1} \quad (2.7)$$

Sehingga persamaannya menjadi :

$$ihr(P) = \frac{y_2 - y_1}{x_2 - x_1} P + \frac{x_2 y_1 - x_1 y_2}{x_2 - x_1} \quad (2.8)$$

Dari persamaan 2.6 kita bisa mendapatkan fungsi *input-output* nya dengan mengintegralkan fungsi ihr.

$$H = \int ihr(P)\, dp = \frac{1}{2} \propto P^2 + \beta\, P + C \quad (2.9)$$

Dimana C adalah bahan bakar minimum saat output masih nol megawatt. C disebut juga *no load fuel,* atau pada fungsi biaya C disebut *no load cost*.

1.3 *Economic dispatch*

Tingkat efisiensi dalam operasi optimalisasi ekonomi dan perencanaan daya pembangkitan listrik akan selalu menjadi bagian penting dalam perindustrian listrik. Oleh karena itu diperlukan perhitungan khusus akan pengiriman daya kepada para konsumen tenaga listrik sehinnga perusahaan pemasok listrik tidak mengalami kerugian.

Tujuan utama dari *Economic dispatch* (ED) adalah untuk menentukan kombinasi daya *output* yang minimal dari setiap unit pembangkit, dengan meminimalkan total biaya bahan bakar, sementara dapat memenuhi kebutuhan baban para konsumen. Pengoptimalan permasalahan ED sangat penting untuk melakukan perkiraan jangka panjang dalam sistem tenaga listrik, penentuan porsi biaya, dan pemodelan managemen operasi tenaga listrik pada pembangkit.

Pembangkitan listrik memiliki tiga komponen biaya utama, antara lain biaya pembangunan, biaya kepemilikan, biaya operasional. Biaya operasional merupakan biaya yang berkaitan langsung dengan keuntungan penjalanan produksi. Hal ini dikarenakan biaya operasional berhubungan langsung dengan managemen pembangkitan daya liatrik.

Salah satu bagian yang paling penting dalam biaya operasional adalah biaya bahan bakar (*fuelcost*). Pada setiap unit pembangkitan nilai yang berbeda tergantung dari jenis bahan bakar yang digunakan dalam pembangkitan. Nilai dari *fuelcost* sangat mempengaruhi fungsi biaya yang didapat. Secara umum nilai dari *fuelcost* dapat dinyatakan dalam persamaan (2.10) berikut.

$$fuelcost = \frac{Fuelprice}{RatingThermal} = \frac{\text{R}}{\text{Mbtu}} \quad (2.10)$$

Fuelcost merupakan harga persatuan panas dari bahan bakar, atau dapat dinyatakan sebagai konversi satuan panas ke satuan mata uang.
Pengaruh nilai *fuelcost* terhadap fungsi biaya dalam dilihat dalam persaman objektif ED berikut,

$$Hi(Pi) = aiPi^2 + biPi + ci \quad (2.11)$$

$$i(Pi) = Hi(Pi) \times fuelcost\ i \quad (2.12)$$

$$Ftotal = min \sum_{i=1}^{n} Fi(Pi) \quad (2.13)$$

Dimana
n : jumlah generator

Dengan terhubungnya banyak unit pembangkit dalam sebuah sistem interkoneksi memberikan kemungkinan pengaturan pembangkitan yang lebih kecil untuk setiap unit.
Equality Constrain merupakan batasan yang merepresentasikan keseimbangan daya dalam sistem. Fungsi persamaan pada ED dinyatakan dalam persamaan,

$$\sum_{i=1}^{n} Pi = Pload + Ploss, n = \text{jumlahgenerator} \quad (2.14)$$

Inequality Constrain merupakan batasan yang merepresentasikan kapasitas daya dari pembangkit. Pada ED fungsi pertidaksamaan dinyatakan dalam persamaan (2.15) berikut.

$$Pimin \leq Pi \leq Pimax \quad (2.15)$$

Jika batasan *minimum* memiliki nilai seperti yang didapatkan pada persamaan (2.16) maka akan didapatkan solusi (2.17).

$$Pi \leq Pimin \quad (2.16)$$

$$Pi = Pimin \quad (2.17)$$

Jika batasan *maximum* memiliki nilai seperti yang didapatkan pada persamaan (2.18) maka akan didapatkan solusi (2.19).

$$Pi \geq Pimax \quad (2.18)$$

$$Pi = Pimax \quad (2.19)$$

1.3.1 *Economic dispatch* dengan rugi – rugi transmisi

Dalam sistem tenaga, kerugian transmisi merupakan kehilangan daya yang harus ditanggung oleh unit pembangkitan. Sehingga daya yang hilang pada kerugian transmisi akan menjadi beban tambahan pada sistem tenaga.

Rugi-rugi dalam jaringan transmisi sistem tenaga akan mejadi sebuah fungsi pembangkitan. Berdasarkan formula rugi-rugi yang konstan, fungsi didapatkan dalam persamaan quadratik yang diselesaikan dengan mengetahui *B coefficient*, yang dapat dikenal sebagai *Bloss matrix*.

Bloss matrix formula sudah diperkenalkan sejal 1950 sebagai metode untuk mendapatkan nilai rugi-rugi dan *incremental loss* dalam perhitungan. Dimana persamaan daya loss secara umum dapat dilihat pada persamaan:

$$Ploss = [P1 \quad \dots \quad Pn]\begin{bmatrix} B11 & \cdots & B1j \\ \vdots & \ddots & \vdots \\ Bi1 & \cdots & Bij \end{bmatrix}\begin{bmatrix} P1 \\ \vdots \\ Pn \end{bmatrix} + [Bo1 \quad \dots \quad Boj]\begin{bmatrix} P1 \\ \vdots \\ Pn \end{bmatrix} + Boo \tag{2.20}$$

Sehingga pada persamaan yang didapatkan pada (2.20) dapat disederhanakan menjadi persamaan (2.21) berikut:

$$Ploss = P^T[B]P + Bo^T P + Boo \tag{2.21}$$

Dimana,
P : *vector* semua generator (MW)
$[B]$: *matrix* persegi dari dimensi yang sama dengan P
Bo : vector dengan panjang yang sama dengan P
Boo : koefisien konstan

Nilai pada Bij secara umum merepresentasikan koefisien rugi-rugi dengan persamaan umum:

$$Ploss = \sum_{i}^{n}\sum_{j}^{n} Pi\ Bij\ Pj + \sum_{i}^{n} Bio\ Pi + Boo \tag{2.22}$$

Nilai dari rugi-rugi daya biasanya berkisar antara 20 % hingga 30 % dari jumlah semua total beban.

Ketika nilai rugo-rugi diperhitungkan maka *penalty factor* di setiap unit akan berbeda. Tidak seperti ketika mengitung nilai optimum pembangkitan tanpa menggunakan rugi-rugi, dimana nilai *penalty factor*

dianggap 1 (satu). Persamaan *penalty factor* dapat dilihat pada persaman (2.23) berikut:

$$Pf = \frac{1}{1 - \frac{\partial Ploss}{\partial Pi}} \quad (2.23)$$

Dimana memiliki hubungan dengan *incremental loss* yang dituliskan dalam persamaan (2.24) berikut:

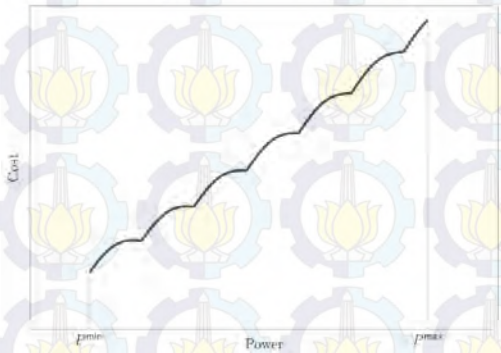
$$incremental\ loss = \frac{\partial Ploss}{\partial Pi} \quad (2.24)$$

1.3.2 *Economic dispatch* dengan *Valve Point Loading Effect*[3]

Economic dispatch dengan *valve point loading effect* yaitu katup yang dioperasikan secara berurutan ada turbin skala besar untuk meningkatkan laju produksi dipengaruhi oleh penggunaan bakar dan menambah riak pada fungsi biaya. Efek ini berupa fungsi sinusoidal, sehingga diperoleh fungsi biaya sebagai berikut :

$$Fi(Pi) = (ai + biPi + ciPi^2) + ei \sin fi\ (Pi_{min} - Pi) \quad (2.25)$$

Berikut gambar kurva *Input Output* dari *Valve Point Loading Effect* :



Gambar 2.4 Kurva *valve point loading effect*

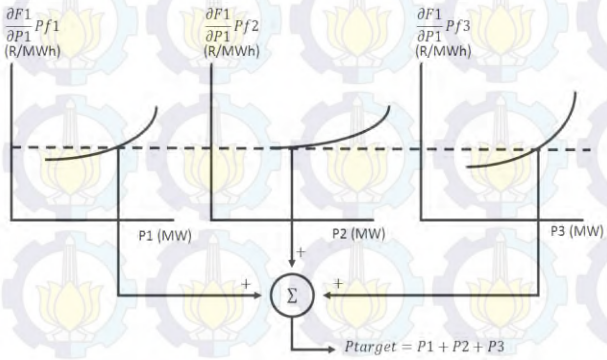
1.3.3 *Economic dispatch Non-convex*[3]

Masalah khusus dari *economic dispatch* adalah meminimalkan biaya operasi pembangkitan dalam sistem tenaga listrik. Sedangkan permasalahan dari *nonconvex economic dispatch* adalah *economic dispatch* dengan quadratic cost function dan *economic dispatch* dengan prohibited operating zones. Biaya operasi dari proses pembangkitan biasanya merupakan fungsi quadratic. *Nonconvex* ini disebabkan adanya

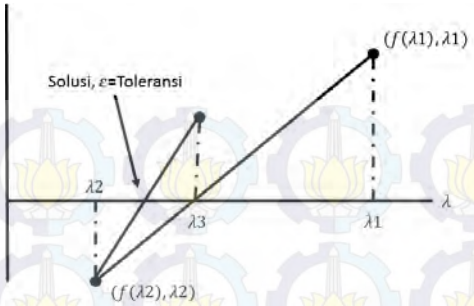
perbedaan harga bahan bakar yang digunakan dan efisiensi tiap bahan bakar berbeda pada level pembangkitan tertentu. *Nonconvex economic dispatch* mempunyai batasan masalah pada pembangkit thermal dengan piecewise quadratic cost function dan dengan prohibited operating zones.

1.4 Iterasi Lambda [5]

Pada metode iterasi lambda, nilai lambda pertama akan ditentukan terlebih dahulu. Tentunya nilai dari lambda petama bukanlah hasil yang benar. Ketika niali total dari $P1 + P2 + P3 + \cdots Pi < Ptarget$ maka nilai λ untuk itersi berikutnya akan bertambah lebih besar dari nilai λ sebelumnya. Dan sebaliknya, jika nilaii total $P1 + P2 + P3 + \cdots Pi > Ptarget$ maka nilai lambda untuk iterasi berikutnya akan lebih kecil daripada nilai dari lambda sebelumnya. Proses ini akan melakukan itersi nilai lambda hingga mendapatkan hasil dimana $P1 + P2 + P3 + \cdots Pi = Ptarget$. Seperti yang dijelaskan dalam Gambar 2.5 dan Gambar 2.6



Gambar 2.5 Grafik penyelesaian iterasi lambda



Gambar 2.6 Proyeksi lambda

Nilai lambda dapat dilihat pada persamaan lagrangian:

$$\frac{\partial Fi}{\partial Pi} = \lambda\left(1 - \frac{\partial Ploss}{\partial Pi}\right) \tag{2.26}$$

$$\lambda = \frac{\partial Fi}{\partial Pi}\left(\frac{1}{1 - \frac{\partial Ploss}{\partial Pi}}\right) \tag{2.27}$$

$$\lambda = \frac{\partial Fi}{\partial Pi} Pfi \tag{2.28}$$

$$\lambda = n\gamma \frac{\partial qi}{\partial Pi} Pfi$$

Menentuka nilai $\lambda start$ dilihat dalam persamaan:

$$\lambda min = min\left(\frac{\partial Fi}{\partial Pi} Pfi, i = 1 \ldots JumlahGenerator\right) \tag{2.29}$$

$$\lambda max = max\left(\frac{\partial Fi}{\partial Pi} Pfi, i = 1 \ldots JumlahGenerator\right) \tag{2.30}$$

$$\lambda start = \frac{\lambda max - \lambda min}{2} \quad (2.31)$$

Untuk melakukan iterasi maka diperlukan $\Delta\lambda$, yang dapat dilihat dalam persamaan:

$$\Delta\lambda = \frac{\lambda max - \lambda min}{2} \quad (2.32)$$

$$Ptarget = Ploss + Pload \quad (2.33)$$

Jika,

$$\sum_{i}^{n} Pi - Ptarget > 0, \qquad n = JumlahGenerator \quad (2.34)$$

$$\lambda = \lambda sebelum - \Delta\lambda \quad (2.35)$$

Jika,

$$\sum_{i}^{n} Pi - Ptarget < 0, \qquad n = JumlahGenerator \quad (2.36)$$

$$\lambda = \lambda sebelum + \Delta\lambda \quad (2.37)$$

Nilai λakan terus berubah hingga mendapatkan nilai,

$$\sum_{i}^{n} Pi - Ptarget = 0, \qquad n = JumlahGenerator \quad (2.38)$$

Nilai λ pada iterasi kedua biasanya bernilai 10 % lebih besar dari nilai λ pertama, atau 10 % kurang dari nilai λ pertama tergantug dari hasil perhitungan error pada iterasi tertentu.
Pada tugas akhir ini nilai *error* (ε) ditentukan sebesar 0.01. Ketika diimplementasikan ke dalam persamaan, maka nilai lambda akan berhenti melakukan iterasi hingga mendapatkan nilai,

$$\sum_{i}^{n} Pi - Ptarget = \varepsilon, \qquad n = JumlahGenerator \tag{2.39}$$

1.5 *Approximation Methods*[1]

Metode ini merupakan penyelesaian *economic dispatch* yang berupa pendekatan dari non-convex untuk fungsi biaya pembangkitan yang dikembangkan untuk memecahkan masalah *economic dispatch* non-convex dengan rugi-rugi transmisi. Sumber non-convex termasuk kondisi fisik seperti : spinning reserve, rugi-rugi transmisi, prohibited operation zones, ramp rate, valve point loading effect, dan dampak pemilihan bakar. Pada tugas akhir ini hanya dibatasi pada rugi-rugi transmisi dan valve point loading effect.

Economic dispatch dengan valve point loading effect yaitu membuat katup turbin secara berurutan menyebabkan ripple pada model biaya pembangkitan. Efek ini adalah model fungsi sinusoidal, sehingga

$$Fi(Pi) = (ai + biPi + ciPi^2) + ei \sin fi\ (Pi_{min} - Pi) \tag{2.40}$$

Dimana ai, bi, ci, ei, dan fi adalah koefisien fuel cost untuk unit i dan Pi_{min} adalah daya pembangkit minimum unit i dengan valve point loading effect.
Sedangkan untuk memperoleh nilai lambda yaitu dengan rumus :

$$\frac{dFi(Pi)}{dPi} = bi + 2ciPi + (-eifi\,(1 - \left(\left(\frac{Yifi(Pi_{min}-Pi))^2}{2!}\right)\right) \tag{2.41}$$

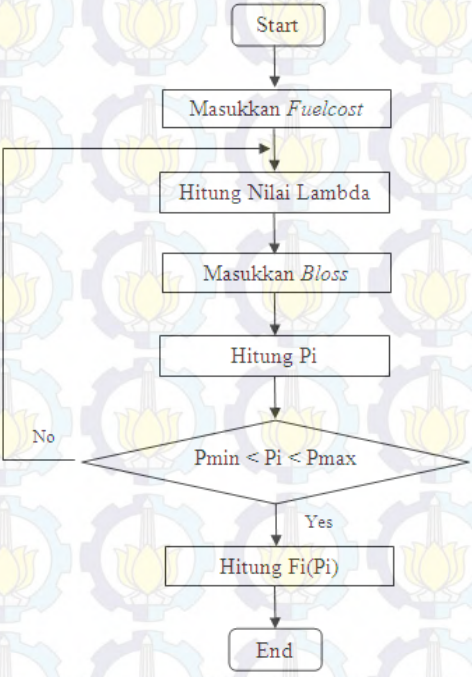
Setelah mendapatkan nilai lambda, maka untuk daya masing-masing generator bisa dicari melalui :

$$Pi = (-lambda \sum_{j=1\neq i}^{N} 2BijPj - 2cifi + eifi^3Yi^2 + Pi_{min}^2)/(Ci + lambda\ Bii \tag{2.42}$$

BAB III
PENGAPLIKASIAN *APPROXIMATION METHODS* PADA *ECONOMIC DISPATCH*

Dalam bab ini dijelaskan mengenai *Approximation Methods* untuk menyelesaikan permasalahan *economic dispatch* pada suatu sistem kelistrikan. Pengolahan data dan simulasi dikerjakan dengan menggunakan hasil pengembangan *software* Powergen yang berbasis Delphi 7. Hasil program metode *Approximation Methods* akan diuji menggunakan contoh sistem pada buku "power generation, operation, and control" karangan Allen J. Wood dan sistem pada paper IET.

1.1 Logaritma ED



Gambar 3.1 Flowcart penerapan ED pada Delphi

Alur dari penggunaan aplikasi perhitungan ED dimulai dengan mengumpulkan semua data yang dibutuhkan. Mulai dari jumlah unit, koefisien tiap orde untuk *Incremental Heat Rate*, *fuelcost*, data batasan

maximum dan *minimum* tiap unit. Setelah itu menentukan berapa beban yang ingin diperhitungkan. Perhitungan akan mengggunakan metode *Approximation* untuk menentukan pembangkitan setiap unit. Hasil pembangkitan akan dimasukan ke dalam persamaan fungsi biaya untuk mendapatkan nilai biaya yang telah ditentukan.

1.2 Inisialisasi Persamaan Objektif dan Constrain ED

Seperti yang telah dijelaskan di dalam bab 2 sebelumnya, ED memiliki persamaan objektif yang dipergunkan untuk mendapatkan nilai pembangkitan dan biaya ketika dikalikan dengan *fuelcost*. Persamaan objektif ED dapat dilihat dalam persaman berikut:

$$Hi(Pi(t)) = aiPi(t)^2 + biPi(t) + ci \quad (3.1)$$

$$Fi(Pi(t)) = Hi(Pi(t)) \times fuelcost\ i \quad (3.2)$$

1.3 *Approximation Methods*

Metode ini merupakan penyelesaian *economic dispatch* yang berupa pendekatan dari non-convex untuk fungsi biaya pembangkitan yang dikembangkan untuk memecahkan masalah *economic dispatch* non-convex dengan rugi-rugi transmisi. Sumber non-convex termasuk kondisi fisik seperti : spinning reserve, rugi-rugi transmisi, prohibited operation zones, ramp rate, valve point loading effect, dan dampak pemilihan bakar. Pada tugas akhir ini hanya dibatasi pada rugi-rugi transmisi dan valve point loading effect.

Untuk mendapatkan solusi *economic dispatch*, valve point loading effect juga dipertimbangkan. Pada metode valve point loading effect yang digunakan Maclaurin dengan fungsi sin dengan fungsi biaya. Maclaurin series digunakan untuk fungsi *Approximation*. Teknik yang dikembangkan untuk menyelesaikan masalah *economic dispatch* dengan biaya kurva non-linier.

Economic dispatch dengan valve point loading effect yaitu membuat katup turbin secara berurutan menyebabkan ripple pada model biaya pembangkitan. Efek ini adalah model fungsi sinusoidal, sehingga

$$Fi(Pi) = (ai + biPi + ciPi^2) + ei \sin fi\ (Pi_{min} - Pi) \quad (3.3)$$

Dimana ai, bi, ci, ei, dan fi adalah koefisien fuel cost untuk unit i dan Pi_{min} adalah daya pembangkit minimum unit i dengan valve point loading effect.

Dari persamaan (3.3) maka dimisalkan :

$$Xi = f_i\ (Pi_{min} - P_i) \quad (3.4)$$

Maclaurin series dan pengembangan untuk fungsi sinus diberikan dalam :

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots. \quad (3.5)$$

Tingkat kompleksitas dari meningkatnya masalah jika urutannya lebih tinggi pada persamaan (3.5) yang digunakan. Maka maclaurin series dianggap sebagai :

$$\sin x = x - \frac{x^3}{3!} \ldots. \quad (3.6)$$

Substitusi Xi di persamaan (3.6) dan *Approximation* sin Xi pada persamaan (3.3) diperoleh :

$$Fi(Pi) = ai + biPi + ciPi^2 + (eifi\left(\frac{fi^2}{6}Pi^3 - \frac{fi^2\,Pi_{min}}{2}Pi^2 + \left(\frac{fi^2\,Pi_{min}^2}{2} - 1\right)Pi + Pi_{min} - \frac{fi^2\,Pi_{min}^3}{6}\right)) \quad (3.7)$$

Turunan dari persamaan (3.7) yang merupakan incremental cost yaitu pada persamaan (3.8) :

$$\frac{dFi(Pi)}{dPi} = bi + 2ciPi + (eifi\left(\frac{fi^2}{2}Pi^2 - fi^2Pi_{min}Pi + \frac{fi^2Pi_{min}^2}{2} - 1\right)) \quad (3.8)$$

Dari persamaan (3.8) bisa disederhanakan menjadi :

$$\frac{dFi(Pi)}{dPi} = bi + 2ciPi + (-eifi\left(1 - \left(\frac{Xi^2}{2!}\right)\right)) \quad (3.9)$$

Saat *Approximation* digunakan akan ada pendekatan yang kurang maksimal. Untuk meminimalisir tingkat kesalahan karena pendekatan, maka ada inisialisasi sebuah faktor yaitu Yi. Dimana yi akan dikalikan ke sisi kanan dalam xi. Persamaan yi adalah sebagai berikut :

$$Yi = real\left(\frac{\cos^{-1}\left(1 - \frac{Xi^2}{2!} + \frac{Xi^4}{4!}\right)}{Xi}\right) \quad (3.10)$$

Dari persamaan (3.9) dan (3.10) maka diperoleh persamaan untuk mencari nilai lambda yaitu :

$$\frac{dFi(Pi)}{dPi} = bi + 2ciPi + (-eifi(1 - \left(\frac{Yifi(Pi_{min} - Pi))^2}{2!}\right)) \quad (3.11)$$

1.3.1 *Approximation Methods* dengan Rugi-rugi Transmisi

Jika menggunakan rugi-rugi transmisi, maka untuk permasalahan lambda bisa dengan cara :

$$L = Fi(Pi) + lambda\ (PD + PL - \sum_{i=1}^{N} Pi \quad (3.12)$$

Sehingga diperoleh :

$$\frac{\partial Fi}{\partial Pi} + lambda\ \frac{\partial P_L}{\partial Pi} = lambda \quad (3.13)$$

Jadi rugi transmisi diperoleh :

$$\frac{\partial P_L}{\partial Pi} = 2\sum_{j=1}^{N} BijPj + Boi \quad (3.14)$$

Dari persamaan (3.14) maka akan diperoleh Pi sebagai berikut :

$$Pi = (-lambda\ \sum_{j=1 \neq i}^{N} 2BijPj - 2cifi + ei\,fi^3 Yi^2 + Pi_{min}^2)/(Ci + lambda\ Bii$$

)

(3.15)

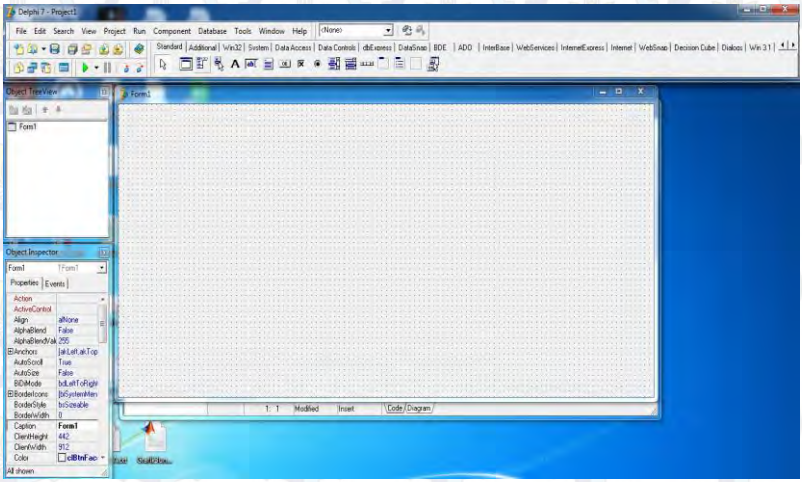
1.4 Delphi

Delphi merupakan bahasa pemrograman berbasis *Windows* yang menyediakan fasilitas pembuatan aplikasi *visual*. Pada tugas akhir ini menggunakan Delphi untuk mengaplikasikan formula DED yang ada. Dengan menghasilkan aplikasi perhitungan DED yang memiliki *interface* yang mudah dipahami, dengan batasan masalah yang telah ditentukan. Delphi sendirinya memiliki beberapa kelebihan diantaranya:

1. Kemudahan penyusunan *User Interface*, Delphi berkomitmen untuk menjadi *Rapid Aplication Development* (RAD). Maksudnya adalah bagaimana mempercepat perkembangan aplikasi.
2. Bahasa *Object Pascal*, merupakan salah satu varian dari bahas pascal dengan sejumlah penambahan, terutama terkait dengan dengan konsep *Object Oriented Programing* (OOP). Dengan salah satu kelebihan bahasa Pascal yang mudah dipahami dan tidak terlalu kompleks.
3. *Native Code*, hasil kompile Delphi adalah kode *native* untuk window 32. Ini berarti *file exe* yang dihasilkan oleh kompiler akan langsung dijalankan oleh mesin tanpa melalui *software* lain seperti *Virtual Machine* (VM). Secara umum *native code* lebih cepat daripada penjalan program dengan VM, hal ini dikarenakan Delphi menggunakan *installer* sederhana. Hasil dari Delphi adalah *file exe* tunggal, tanpa perlu file-file lainnya.

Ketika memulai Delphi, maka akan ditempatkan ke dalam *Integrated Development Environment* (IDE). IDE ini menyediakan semua alat yang dibutuhkan dalam merancang, mengembangkan,

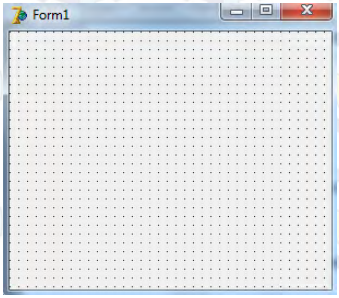
menguji, *debug*, dan penyebaran aplikasi dalam waktu yang singkat. Dalam hal ini IDE mencangkup semua alat yang diperlukan untuk memulai perancangan aplikasi seperti seperti yang terlihat pada Gambar 3.2 berikut.



Gambar 3.2 Tampilan pengerjaan Delphi

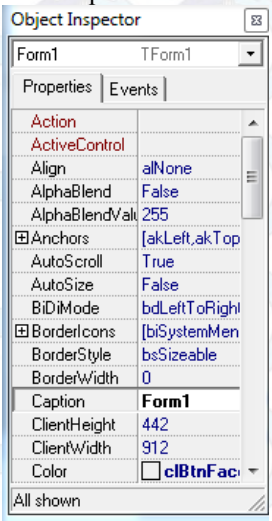
Beberapa bagian penting yang perlu diketahui dalam pemrograman dengan menggunakan Delphi diantaranya adalah:

1. *Form Designer* atau *Form*, merupakan jendela kosong yang digunakan untuk merancang suatu *User Interface* (UI) dalam perancangan aplikasi yang sedang dibuat. Dapat dilihat pada Gambar 3.3 berikut,



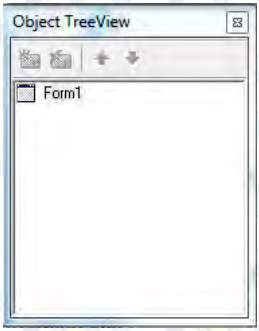
Gambar 3.3 Tampilan *Form Designer*

2. *Object Inspector*, digunakan untuk mengatur dan memeriksa sekumpulan *property* yang berada di dalam *Form* untuk mendapatkan tampilan sesuai denganyang diinginkan. *Object Inspector* dapat dilihat pada Gambar 3.4.



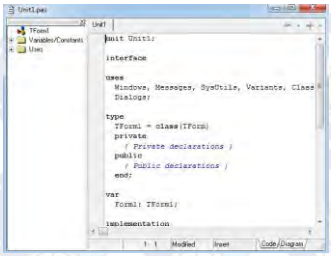
Gambar 3.4 Tampilan *Object Inspector*

3. *Object TreeView*, digunkan untuk menampilkan dan mengubah hubungan antar komponen dalam *Form*. Tampilan dapat dilihat pada Gambar 3.5.



Gambar 3.5 Tampilan *Object TreeView*

4. *Code Editor*, merupakan tempat penulisan dan edititing logika pemrograman yang sedang dibuat. Dapat dilihat pada Gambar 3.6.



Gambar 3.6 Tampilan *Code Editor*

1.4.1 Sintaksis *Approximation Methods* pada Delphi

Sintaksis program adalah perintah yang digunakan untuk melakukan pemanggilan program dengan argumen input yang kita masukkan. Sehingga untuk menjalankan metode *Approximation Methods* pada program Powergen yang berbasis Delphi ini dibutuhkan sintaksis program sebagai berikut :

Tabel 1.1 Daftar sintaksis yang digunakan pada program

Sintaksis	Keterangan
datadump	Digunakan sebagai perintah menampilkan data permasalahan *ED* pada lembar *Output*
Data_input	Digunakan untuk menerima masukan dari data *file* yang telah tersimpan di awal
Data_output	Digunakan untuk menulis masukan pada data *file*, agar dapat tersimpan
ihr_ftn	Sebagai inisiasi turunan pertama persamaan $Hi(Pi(t))$ yang nantinya akan dikalikan dengan *fuelcost*, sehingga didapatkan inisiasi $\frac{\partial Fi}{\partial Pi}$
invers_ihr_ftn	Mencari nilai pembangkitan setiap unit ketika didapatkan nilai lambda
prod_cost	Mendapatkan nilai biaya pembangkitan setiap unit setelah mendapatkan nilai pembangkitan yang optimal dari proses *lambdasearch*
*Approximation*_methods_dispatch	Sebagai prosedur penjalanan metode *Approximation*untuk memperoleh nilai pembangkitan yang optimal
Output_Routine	Merupakan prosedur untuk memperoleh hasil akhir tiap periode

1.4.2 Argumentasi Input Output Pada Delphi

Argumen input output adalah variabel yang dilibatkan sebagai data input dan output dalam program.

Tabel 1.2 Daftar argumentasi input output pada Delphi

Argumen	Keterangan
Coeff[i,j]	Sebagai masukan awal dari nilai koefisien A, B, C dalam persamaan $Hi(Pi(t)) = ai + biPi(t) + ciPi(t)^2$
Fuelcost[i]	Sebagai masukan awal nilai dari *fuelcost* yang digunakan untuk persaman $Fi(Pi(t)) = Hi(Pi(t)) \times fuelcosti$
Fungsif[i]	Sebagai koefisien fuelcost (fi) untuk unit i yang digunakan pada persamaan $Fi(Pi) = (ai + biPi + ciPi^2) + ei \sin fi\ (Pi_{min} - Pi)$
Fungsie[i]	Sebagai koefisien fuelcost (ei) untuk unit i yang digunakan pada persamaan $Fi(Pi) = (ai + biPi + ciPi^2) + ei \sin fi\ (Pi_{min} - Pi)$
Unitmax[i]	Sebagai masukan awal dari batas maximum pembangkitan unit (Pmax)
Unitmin[i]	Sebagaimasukan awal dari batasan minimum pembangititan unit (Pmin)

1.5 *Software* Powergen

Software Powergen adalah sebuah perangkat lunak milik Teknik Elektro Institut Teknologi Sepuluh Nopember yang digunakan untuk melakukan perhitungan-perhitungan yang berkaitan dengan Teknik Sistem Tenaga. *Software* ini sudah digunakan sebagai alat bantu pada proses akademik. Salah satu mata kuliah yang menggunakan *software* ini adalah mata kuliah operasi optimum. Tampilan menu utama *software* Powergen dapat dilihat pada Gambar 3.7.

Software Powergen yang akan dikembangkan memiliki beberapa fitur antara lain :

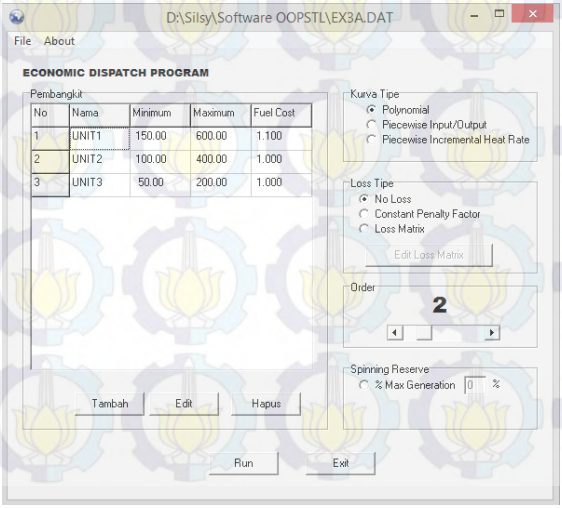
- Power Flow : Melakukan perhitungan aliran daya
- DUBLP : Melakukan perhitungan optimasi sederhana dengan *linear programming*
- EDC : Melakukan perhitungan *economic dispatch*
- Hydro : Menyelesaikan permasalahan penjadwalan pembangkit tenaga air/*hydro*
- Unitcom : Melakukan perhitungan *Unit Commitment*

Gambar 3.7 Tampilan Menu Utama *Software* Powergen

Pada tugas akhir ini menu yang akan dikembangkan adalah menu EDC (*Economic dispatch*). Oleh karena itu akan dijelaskan terlebih dahulu fitur-fitur yang telah ada pada kedua menu tersebut.

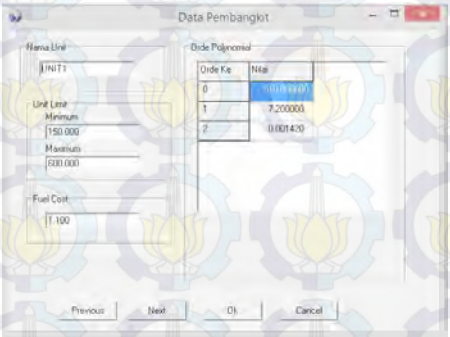
1.5.1 Menu EDC (*Economic dispatch*)

Menu EDC ini adalah menu pada program Powergen yang digunakan untuk melakukan perhitungan *economic dispatch*. Program Powergen ini mampu melakukan perhitungan *economic dispatch* hingga 20 unit pembangkit dengan tiga jenis tipe kurva. Tampilan menu EDC dapat dilihat pada Gambar 3.8



Gambar 3.8 Tampilan Utama Menu EDC

User/pengguna *software* mula mula mengisi data pembangkit dengan cara menekan tombol **Tambah** dan mengisi data-data pembangkit. *User* dapat mengisi kan data-data yang dimiliki oleh unit pembangkit seperti batasan pembangkitan minimum dan maksimum; karakteristik pembangkitan; serta biaya bahan bakar. Tampilan pengisian data-data pembangkit seperti pada Gambar 3.9.



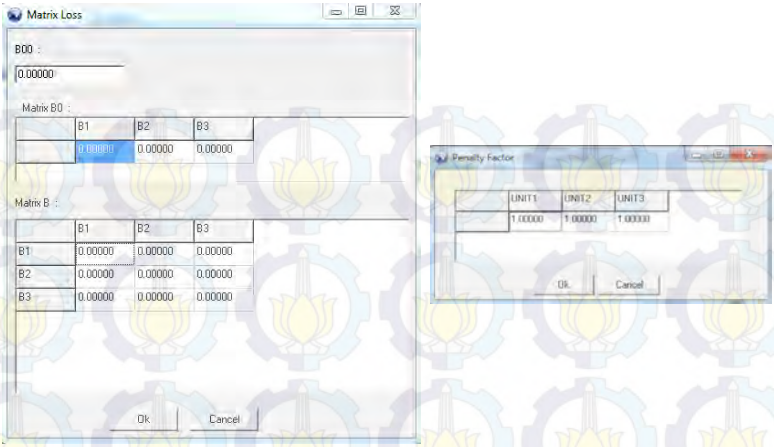
Gambar 3.9 Tampilan Pengisian Data Pembangkit

User pun dapat memilih metode perhitungan *losses* yang akan digunakan. Terdapat dua tipe perhitungan *losses* pada menu EDC ini jika *user* ingin memasukkan faktor *losses* kedalam perhitungan *economic dispatch*-nya. Tipe perhitungan *losses* yang pertama adalah dengan matriks B_{loss}. Tipe perhitungan *losses* yang kedua adalah dengan *constant penalty factor*. Namun dalam tugas akhir ini *losses* akan diperhitungkan menggunakan matriks B_{loss}. Tampilan pengisian *losses*dapat dilihat pada Gambar 3.10.

Pada tugas akhir ini digunakan sistem dengan 6 pembangkit, sehingga untuk matriks B_{loss} akan mengikuti jumlah pembangkit tersebut. Matriks yangdigunakan pada tugas akhir ini adalah sebagai berikut :

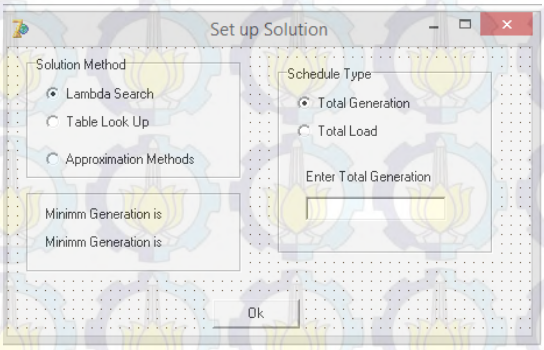
$$B_{ij} = \begin{bmatrix} 0.0017 & 0.0012 & 0.0007 & -0.0001 & -0.0005 & -0.0002 \\ 0.0012 & 0.0014 & 0.0009 & 0.0001 & -0.0006 & -0.0001 \\ 0.0007 & 0.0009 & 0.0031 & 0 & -0.001 & -0.0006 \\ -0.0001 & 0.0001 & 0 & 0.0024 & -0.0006 & -0.0008 \\ -0.0005 & -0.0006 & -0.001 & -0.0006 & 0.0129 & -0.0002 \\ -0.0002 & -0.0001 & -0.0006 & -0.0008 & -0.0002 & 0.015 \end{bmatrix}$$

$$\boldsymbol{Bo} = 0.001*[-0.3908 - 0.12970.70470.05910.2161 - 0.6635], \ \text{Boo} = 0.0056$$



.**Gambar 3.10**Tampilan pilihan pengisian metode perhitungan *losses*

Setelah mengisikan data-data yang dibutuhkan *user* dapat melanjukan ke proses selanjutnya dengan memilih tombol **Run**. Ketika user memilih tombol **Run**, maka akan muncul tampilan seperti Gambar 3.11.

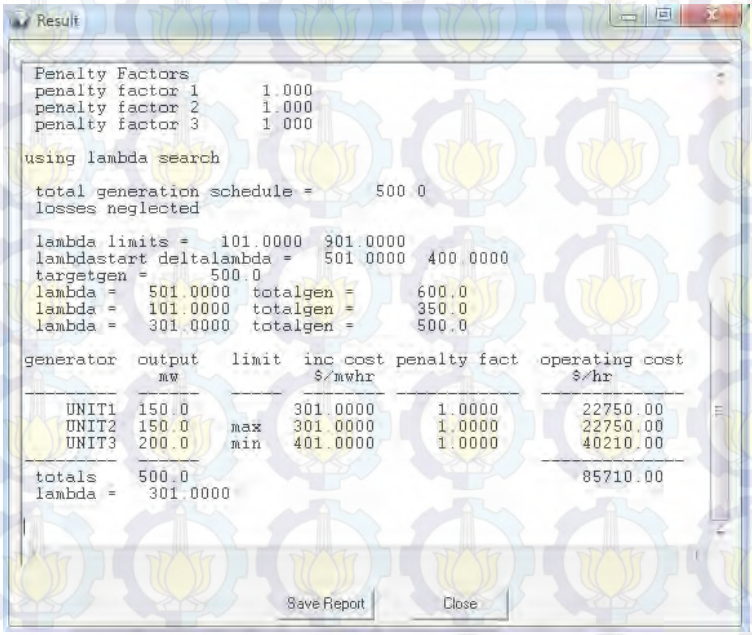


Gambar 3.11 Tampilan *Set up Solution* menu EDC

Pada tampilan ini *user* dapat memilihkan metode optimasi yang akan digunakan. Terdapat pilihan metode optimasi Lambda Search, Table Look Up, dan *Approximation Methods*. Pada tugas akhir ini yang akan dikembangkan hanya pada *Approximation Methods* dengan tipe kurva *polynomial*. Selain itu pada tampilan ini *user* dapatmelihat pembangkitan maksimum dan pembangkitan minimum yang dapat di lakukan oleh unit-unit pembangkit yang ada. Selanjutnya *user*

mengisikan **Total Generation** atau **Total Load** yang diinginkan sesuai **Schedule Type** yang dipilih.

Ketika semua data telah diisikan *user* dapat memilih **Ok** untuk menampilkan hasil perhitungan optimasi seperti Gambar 3.12.

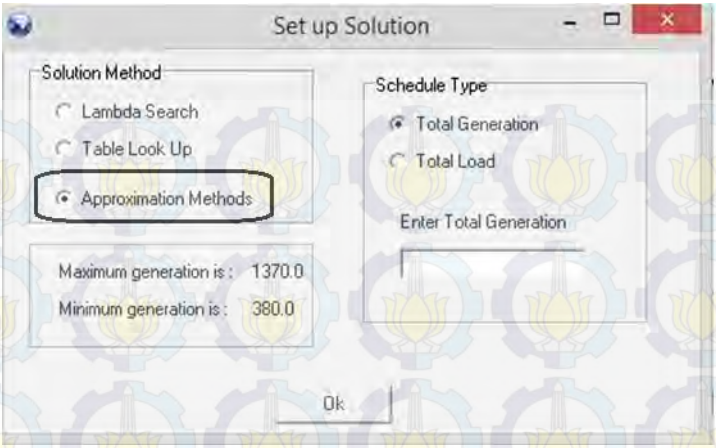


Gambar 3.12 Tampilan hasil perhitungan EDC

Pada tampilan hasil, *user* dapat memilih untuk menyimpan hasil perhitungan dengan menekan tombol **Save Report** atau dapat langsung menutup tampilan hasil dengan menekan tombol **Close**.

1.5.1.1 Menu Set Up Solution pada Menu *Economic dispatch* (ED)

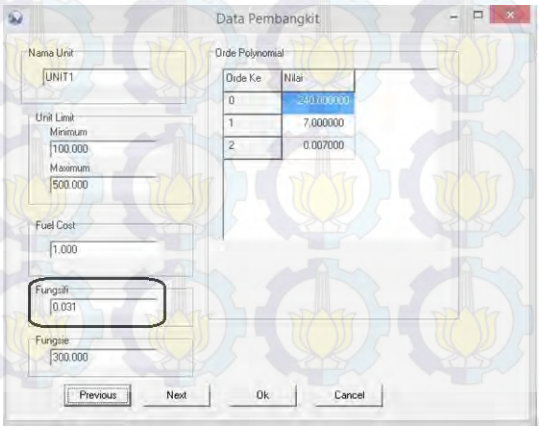
Pada menu set up solution di menu *economic dispatch* telah ditambahkan metode baru yaitu *Approximation Methods*. Metode yang sebelumnya tersedia pada software powergen yaitu metode Lambda Search dan metode Table Look Up. Sehingga untuk tampilan baru pada menu Set Up Solution pada menu economic dispatch seperti pada tampilan gambar 3.13.



Gambar 3.13 Tampilan Set Up Solution EDC

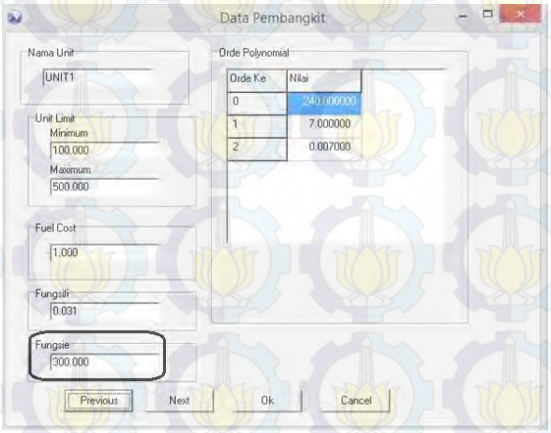
1.5.1.2 Menu Data Pembangkit pada Menu *Economic dispatch* (ED)

Pada menu data pembangkit di *economic dispatch* telah ditambahkan menu untuk penambahan harga bahan bakar. Penambahan harga bahan bakar ini dikarenakan sistem pembangkitan merupakan sistem non convex, yang memperhatikan valve point loading effect. Sehingga biaya bahan bakar berbeda. Penambahan menu data pembangkit adalah sebagai berikut :



Gambar 3.14 Tampilan Data Pembangkit EDC Fungsi f

Pada menu ini terdapat fungsi untuk fuelcost *economic dispatch* dengan valve point loading effect. Hal ini dikarenakan adanya perbedaan harga bahan bakar. Selain fungsi f, ada juga fungsi e yang merupakan fuelcost *economic dispatch* dengan valve point loading effect seperti pada gambar di bawah :



Gambar 3.15 Tampilan Data Pembangkit EDC Fungsi e

1.6 Langkah Perhitungan *Economic dispatch* Menggunakan *Approximation Methods*

1.6.1 Perhitungan *Economic dispatch* tanpa *losses*

Pada perhitungan *economic dispatch* menggunakan *Approximation* method tanpa *losses* ini, langkah yang dilakukan yaitu :

1. Mencari nilai lambda
   Nilai lambda yang dihitung adalah lambda dengan iterasi pertama, sehingga diperoleh :

Tabel 1.3 Hasil perhitungan nilai lambda

Unit	a	b	c	e	f	Pmin	Pmax	lambda
1	240	7	0,007	300	0,031	100	500	13,25349
2	200	10	0,0095	200	0,042	50	200	13,25349
3	220	8,5	0,009	150	0,063	80	200	13,25349
4	200	11	0,009	150	0,063	50	150	13,25349
5	220	10,5	0,008	150	0,063	50	200	13,25349
6	190	12	0,0075	150	0,063	50	120	13,25349

2. Menghitung daya masing-masing unit
   Total daya yang diminta untuk kasus ini yaitu 1263, dengan perbandingan perhitungan menggunakan metode PSO. Dari nilai lambda yang diperoleh, maka dihitung nilai daya pada masing-masing unit.

Tabel 1.4 Hasil perhitungan daya masing-masing unit

Unit	a	b	c	e	f	Pmin	Pmax	Pi
1	240	7	0,007	300	0,031	100	500	446,6779
2	200	10	0,0095	200	0,042	50	200	171,2363
3	220	8,5	0,009	150	0,063	80	200	264,0828
4	200	11	0,009	150	0,063	50	150	125,1939
5	220	10,5	0,008	150	0,063	50	200	172,0931
6	190	12	0,0075	150	0,063	50	120	83,566

   Dari hasil daya yang diminta pada masing-masing unit, masih terdapat beberapa unit yang melebihi batas maksimal. Sehingga perlu di optimasi lagi pada iterasi selanjutnya.
3. Setelah mendapat daya masing-masing unit, maka substitusi nilai daya pada persamaan Fi(Pi)

Tabel 1.5 Hasil perhitungan biaya masing-masing unit

Unit	a	b	c	e	f	Pmin	Pmax	FiPi
1	240	7	0,007	300	0,031	100	500	5054,173007
2	200	10	0,0095	200	0,042	50	200	2376,688286
3	220	8,5	0,009	150	0,063	80	200	3216,022246
4	200	11	0,009	150	0,063	50	150	1868,148144
5	220	10,5	0,008	150	0,063	50	200	2411,939385
6	190	12	0,0075	150	0,063	50	120	1373,524103

4. Akan diperoleh harga pada masing-masing unit
5. Dari harga tersebut, maka akan dijumlah keseluruhan biaya pembangkitan

1.6.2 Perhitungan *Economic dispatch* dengan *losses*

Pada perhitungan *economic dispatch* menggunakan *Approximation* method tanpa *losses* ini, langkah yang dilakukan yaitu :

1. Mencari nilai lambda

Nilai lambda yang dihitung menggunakan rumus :

$$\frac{dFi(Pi)}{dPi} = bi + 2ciPi + (-eifi(1 - \left(\left(\frac{Yifi(Pi_{min} - Pi))^2}{2!}\right)\right)$$

(3.16)

2. Menghitung daya masing-masing unit
   Daya yang dihitung terlebih dahulu yaitu daya masing-masing unit sebelum memperhitungkan valve point loading effect. Setelah mendapatkan nilai daya masing-masing unit tersebut, dihitung kembali untuk mendapatkan daya unit setelah ada valve point loading effect.
   Untuk rumus P[i] yang digunakan yaitu :

$$Pi = (-lambda \sum_{j=1\neq i}^{N} 2BijPj - 2cifi + ei\,fi^3Yi^2 + Pi_{min}^2)/(Ci + lambda\ Bii$$

) (3.17)

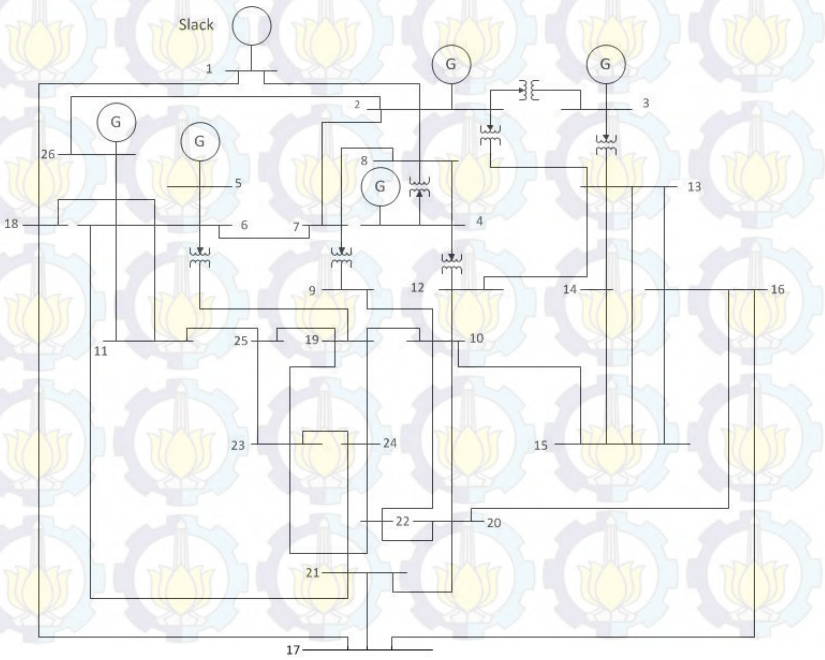
3. Setelah mendapat daya masing-masing unit, maka substitusi nilai daya pada persamaan Fi(Pi)
4. Akan diperoleh harga pada masing-masing unit

Dari harga tersebut, maka akan dijumlah keseluruhan biaya pembangkitan

BAB IV
SIMULASI DAN ANALISA

Pada Bab ini akan menampilkan hasil simulasi perhitungan *Economic dispatch* dengan menggunakan *Approximation Methods dan* PSO. Analisa yang dilakukan adalah membandingkan hasil perhitungan kedua metode tersebut. Diberikan beberapa contoh kasus untuk menganalisa data yang dibutuhkan dalam Tugas Akhir.

Pada kasus ini dilakukan uji coba terhadap kasus plant untuk menguji validasi dari Delphi yang dikembangkan. Pada kasus ini digunakan data-data dari referensi [4] untuk rugi-rugi transmisi yang digunakan dan referensi [1] untuk data pembangkit yang digunakan. Referensi plant yang digunakan mempunyai single-line diagram seperti gambar dibawah ini:



Gambar 1.1 Single-Line Diagram 6 Pembangkit 26 Bus

Untuk data yang dibutuhkan sebagai parameter tambahan yaitu berupa data-data bus, linedata , biaya pembangkitan dan batasan minimal dan maksimal pembangkit sebagai berikut:

Tabel 1.1 Tabel busdata 6 Pembangkit 26 Bus

Dari Bus	Ke Bus	Voltage Mag	Angle Degree	Load		Generation				Injected
				MW	Mvar	MW	Mvar	Qmin	Qmax	Mvar
1	1	1.025	0	51	41	0	0	0	0	4
2	2	1.02	0	22	15	79	0	40	250	0
3	2	1.025	0	64	50	20	0	40	150	0
4	2	1.05	0	25	10	100	0	25	180	2
5	2	1.045	0	50	30	300	0	40	160	5
6	0	1	0	76	29	0	0	0	0	2
7	0	1	0	0	0	0	0	0	0	0
8	0	1	0	0	0	0	0	0	0	0
9	0	1	0	89	50	0	0	0	0	3
10	0	1	0	0	0	0	0	0	0	0
11	0	1	0	25	15	0	0	0	0	1.5
12	0	1	0	89	48	0	0	0	0	2
13	0	1	0	31	15	0	0	0	0	0
14	0	1	0	24	12	0	0	0	0	0
15	0	1	0	70	31	0	0	0	0	0.5
16	0	1	0	55	27	0	0	0	0	0
17	0	1	0	78	38	0	0	0	0	0
18	0	1	0	153	67	0	0	0	0	0
19	0	1	0	75	15	0	0	0	0	5
20	0	1	0	48	27	0	0	0	0	0
21	0	1	0	46	23	0	0	0	0	0
22	0	1	0	45	22	0	0	0	0	0
23	0	1	0	25	12	0	0	0	0	0
24	0	1	0	54	27	0	0	0	0	0
25	0	1	0	28	13	0	0	0	0	0
26	2	1.015	0	40	20	60	0	15	50	0

Untuk parameter transmisi yang lain yaitu berupa tabel line data. Ini digunakan untuk mengetahui impedansi yang digunakan.

Tabel 1.2 Tabel Linedata

Bus nl	Bus nr	R Pu	X pu	½ B pu	Tap setting Value
1	2	0.00055	0.0048	0.03	1
1	18	0.0013	0.0115	0.06	1
2	3	0.00146	0.0513	0.05	0.96
2	7	0.0103	0.0586	0.018	1
2	8	0.0074	0.0321	0.039	1
2	13	0.00357	0.0967	0.025	0.96
2	26	0.0323	0.1967	0	1
3	13	0.0007	0.00548	0.0005	1.017
4	8	0.0008	0.024	0.0001	1.05
4	12	0.0016	0.0207	0.015	1.05
5	6	0.0069	0.03	0.099	1
6	7	0.00535	0.0306	0.00105	1
6	11	0.0097	0.057	0.0001	1
6	18	0.00374	0.0222	0.0012	1
6	19	0.0035	0.066	0.045	0.95
6	21	0.005	0.09	0.0226	1
7	8	0.0012	0.00693	0.0001	1
7	9	0.00095	0.0429	0.025	0.95
8	12	0.002	0.018	0.02	1
9	10	0.00104	0.0493	0.001	1
10	12	0.00247	0.0132	0.01	1
10	19	0.0547	0.236	0	1
10	20	0.0066	0.016	0.001	1
10	22	0.0069	0.0298	0.005	1
11	25	0.096	0.27	0.01	1
11	26	0.0165	0.097	0.004	1
12	14	0.0327	0.0802	0	1
12	15	0.018	0.0598	0	1
13	14	0.0046	0.0271	0.001	1
13	15	0.0116	0.061	0	1
13	16	0.01793	0.0888	0.001	1
14	15	0.0069	0.0382	0	1
15	16	0.0209	0.0512	0	1
16	17	0.099	0.06	0	1

Bus Nl	Bus nr	R pu	X pu	½ B Pu	Tap setting Value
16	20	0.0239	0.0585	0	1
17	18	0.0032	0.06	0.038	1
17	21	0.229	0.445	0	1
19	23	0.03	0.131	0	1
19	24	0.03	0.125	0.002	1
19	25	0.119	0.2249	0.004	1
20	21	0.0657	0.157	0	1
20	22	0.015	0.0366	0	1
21	24	0.0476	0.151	0	1
22	23	0.029	0.099	0	1
22	24	0.031	0.088	0	1
23	25	0.0987	0.1168	0	1

Tabel 1.3 Tabel koefisien *cost* pembangkit

α	β	γ
240	7	0.007
200	10	0.0095
220	8.5	0.009
200	11	0.009
220	10.5	0.008
190	12	0.0075

Tabel 1.4 Tabel batasan daya pembangkit

MW min	MW max
100	500
50	200
80	300
50	150
50	200
50	120

Selain parameter tersebut, ditentukan pula permintaan beban yang dibutuhkan. Untuk tugas akhir ini digunakan pembangkit termal yaitu menggunakan bahan bakar batubara. Selain itu juga hanya dihitung untuk permintaan beban dalam 1 jam.

4.1 Hasil Simulasi *Economic dispatch* tanpa *losses*

Pada kasus ini akan ada 2 pengujian yaitu pengujian sistem tanpa *losses* tanpa batasan Pmax dan Pmin dan dengan batasan Pmax dan Pmin.

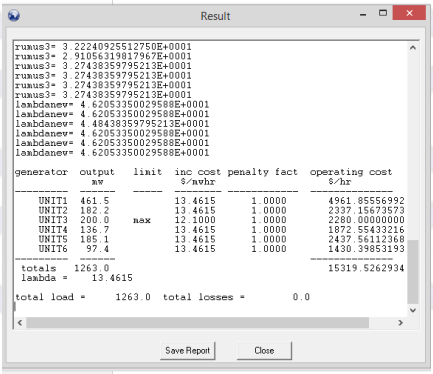
4.1.1 Kasus 1 (Tanpa Batasan Pmax dan Pmin)

Pada simulasi ini digunakan 6 unit pembangkit pada software powergen dengan daya yang diminta adalah 1263 MW.

Tabel 1.5 *Load Profile* Harian Kasus 1

Jam	Total beban	Fuelcost	Jam	Total beban	Fuelcost
1	882	10.437,2807	13	1000	11.892,1446
2	850	10.052,2449	14	1100	13.168,3933
3	882	10.437,2807	15	1100	13.168,3933
4	860	10.172,3137	16	1170	14.081,6818
5	920	10.899,6307	17	1220	14.743,7464
6	980	11.641,4732	18	1300	15.819,8899
7	950	11.268,6936	19	1263	15.319,5262
8	1000	11.892,1446	20	1320	16.092,1643
9	1100	13.168,3933	21	1350	16.503,2009
10	950	11.268,6936	22	1380	16.845,4230
11	1000	11.892,1446	23	1350	16.503,2009
12	950	11.268,6936	24	1320	16.092,1643

Sehingga diperoleh hasil sebagai berikut :



Gambar 4.2 Hasil simulasi *economic dispatch* tanpa *losses*

Dari hasil simulasi pada delphi, maka akan dibandingkan dengan matlab menggunakan metode PSO dengan hasil sebagai berikut :

Tabel 4.6 Perbandingan Hasil Metode *Approximation* dan PSO

Unit	*Approximation Methods*		PSO	
	P (MW)	Biaya ($/hr)	P (MW)	Biaya ($/hr)
1	461.5	4961.8555	404.0251	4210,829712
2	182.2	2337.1567	127.7993	1658,282278
3	200.0	2280.0000	229.6017	2646,086287
4	136.7	1872.5543	199.9046	2761,488841
5	185.1	2437.5611	171.8345	2408,098129
6	97.4	1430.3985	129.8348	2016,962377
Total	1263.0	15319.5262	1263.0	15701,74762

Pada hasil diatas, untuk metode PSO masih ada kekurangan yaitu untuk batasan daya maksimum. Sehingga akan dilakukan uji coba kembali dengan batasan daya maksimum untuk kasus selanjutnya. Pengujian dilakukan kembali bertujuan untuk mendapatkan daya yang sesuai dengan permintaan, yaitu tidak melebihi batas maksimal pembangkitan dan tidak dibawah batas minimal.

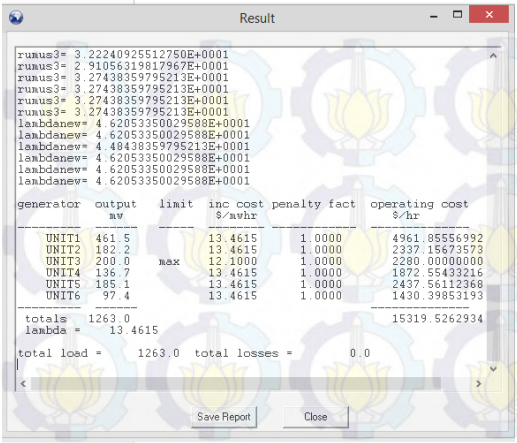
4.1.2 Kasus 2 (Dengan Batasan Pmax dan Pmin)

Pada simulasi ini digunakan 6 unit pembangkit pada software powergen dengan daya yang diminta adalah 1263 MW.

Tabel 1.7 *Load Profile* Harian Kasus 2

Jam	Total beban	Fuelcost	Jam	Total beban	Fuelcost
1	882	10.437,2807	13	1000	11.892,1446
2	850	10.052,2449	14	1100	13.168,3933
3	882	10.437,2807	15	1100	13.168,3933
4	860	10.172,3137	16	1170	14.081,6818
5	920	10.899,6307	17	1220	14.743,7464
6	980	11.641,4732	18	1300	15.819,8899
7	950	11.268,6936	19	1263	15.319,5262
8	1000	11.892,1446	20	1320	16.092,1643
9	1100	13.168,3933	21	1350	16.503,2009
10	950	11.268,6936	22	1380	16.845,4230
11	1000	11.892,1446	23	1350	16.503,2009
12	950	11.268,6936	24	1320	16.092,1643

. Sehingga diperoleh hasil sebagai berikut :



Gambar 4.3 Hasil simulasi *economic dispatch* tanpa *losses*

Dari hasil simulasi pada delphi, maka akan dibandingkan dengan matlab menggunakan metode PSO, dimana pada metode PSO telah dilakukan perbaikan dengan batasan daya yang diminta dengan hasil sebagai berikut :

Tabel 4.8 Perbandingan Hasil Metode *Approximation* dan PSO

Unit	*Approximation Methods*		PSO	
	P (MW)	Biaya ($/hr)	P (MW)	Biaya ($/hr)
1	461.5	4961.8555	400	4197,336327
2	182.2	2337.1567	200	2583,36278
3	200.0	2280.0000	195	2342,759647
4	136.7	1872.5543	148	2041,481274
5	185.1	2437.5611	200	2643,782905
6	97.4	1430.3985	120	1881,194166
Total	1263.0	15319.5262	1263.0	15689,9171

Dari hasil perhitungan diatas, diperoleh daya masing – masing unit sesuai dengan permintaan yaitu tidak melebihi batas maksimum dan minimum pembangkitan. Selain itu diperoleh bahwa untuk biaya pembangkitan menggunakan *Approximation Methods* lebih murah dibandingkan dengan metode PSO. Selisih biaya pembangkitan antara metode *Approximation* dengan metode PSO yaitu +/- 370 $/hr.

4.2 Hasil Simulasi *Economic dispatch* dengan *Losses*

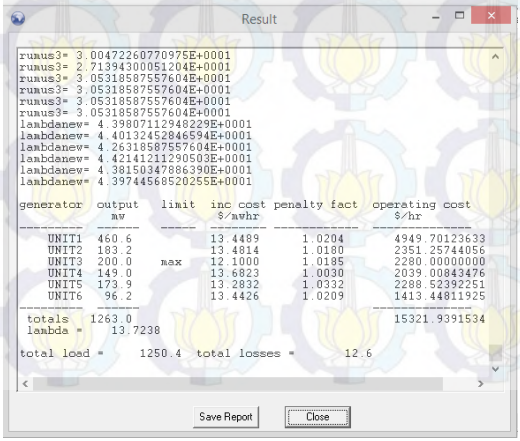
Pada simulasi ini digunakan 6 unit pembangkit dengan losses pada software powergen dengan daya yang diminta adalah 1263 MW.

Tabel 1.9 *Load Profile* Harian Kasus 3

Jam	Total beban	Fuelcost	Jam	Total beban	Fuelcost
1	882	10.438,1513	13	1000	11.893,0051
2	850	10.053,1334	14	1100	13.169,7977
3	882	10.438,1513	15	1100	13.169,7977
4	860	10.173,0037	16	1170	14.083,5036
5	920	10.900,2278	17	1220	14.745,7702
6	980	11.642,3722	18	1300	15.821,2074
7	950	11.269,5185	19	1263	15.321,9391
8	1000	11.893,0051	20	1320	16.093,1500
9	1100	13.169,7977	21	1350	16.504,0142
10	950	11.269,5185	22	1380	16.845,4230
11	1000	11.893,0051	23	1350	16.504,0142
12	950	11.269,5185	24	1320	16.093,1500

.

Sehingga diperoleh hasil sebagai berikut :



Gambar 4.4 Hasil simulasi *economic dispatch* dengan *losses*

Tabel 4.10 Perbandingan Hasil Metode *Approximation* dan paper

Unit	*Approximation Methods*		Paper *Approximation*	
	P (MW)	Biaya ($/hr)	P (MW)	Biaya ($/hr)
1	460.6	4949.7012	467.5062	5319,131581
2	183.2	2351.2574	168.2134	2344,598093
3	200.0	2280.0000	268.1626	3244,599499
4	149.0	2039.0084	117.8149	1756,596593
5	173.9	2288.5239	167.3637	2335,807025
6	96.2	1413.4481	87.3330	1401,71109
Total	1263.0	15321.9391	1263	15357

Dari hasil tersebut diperoleh bahwa harga pembangkitan antara paper dan perhitungan delphi nilainya mendekati. Namun pada paper ada 1 pembangkit yang daya nya melebihi pmax. Sehingga masih diperlukan iterasi selanjutnya. *Losses* yang digunakan pada tugas akhir ini yaitu *losses* iterasi ke 3 pada referensi [4] dengan plan pada IEEE 26 bus dengan 6 generator.

Halaman ini sengaja dikosongkan

BAB V
PENUTUP

5.1 Kesimpulan

Dari semua proses yang meliputi studi literatur, serta simulasi dan analisis, maka terdapat beberapa hal yang dapat disimpullkan terkait Tugas Akhir ini, yaitu:

1. Aplikasi perhitungan ED pada powergen untuk metode *Approximation Methods* bisa berjalan dengan baik dengan memperhatikan batasan daya maksimum yang diminta pada setiap pembangkit.
2. Valve point loading effect pada *economic dispatch* memiliki pengaruh pada fungsi biaya pembangkitan.
3. Dalam simulasi pengoperasian perhitungan ED menggunakan *Approximation Methods*, biaya pembangkitan lebih murah jika dibandingkan dengan metode PSO. Dalam simulasi diperoleh bahwa untuk biaya pembangkitan bisa lebih murah +/- 370 $/HR.
4. Hasil akhir dari aplikasi perhitungan yang dikembangkan dapat digunakan untuk meng-*upgrade* aplikasi perhitungan yang selama ini digunakan pada mata kuliah Operasi Optimum Sistem Tenaga Listrik.

5.2 Saran

Adapun saran untuk penelitian selanjutnya yang berkaitan dengan Tugas Akhir ini, yaitu:

1. Aplikasi perhitungan ED dapat dikembangkan dengan memberikan perhitungan rugi-rugi dengan menggunakan *Optimal Power Flow (OPF),* agar hasil perhitungan lebih optimal.
2. Aplikasi software powergen bisa disempurnakan lagi dan dikembangkan dengan metode yang lain.
3. Aplikasi ED dapat dikembangkan dengan menggunakan metode perhitungan yang berbeda sehingga dapat dibandingkan metode *Approximation Methods*.

DAFTAR PUSTAKA

[1] Abouheaf, Mohammed I., Wei-Jen Lee, Frank L.Lewis.,"Dynamic formulation and *Approximation Methods* to solve *economic dispatch* problems",IET Generation, Transmission and Distribution, 2013

[2] Penangsang, O., "Analisis Aliran Daya", ITS Press Surabaya, 2012

[3] Asyhari, Nurcholis., "*Nonconvex Economic dispatch* dengan Menggunakan Metode Integrated Artificial Intelligence", ITS Surabaya,2002

[4] Hadi S., "Power System Analysis 2nd Edition",McGrowHill, Ch1,1999.

[5] Wood, A. J. dan Wollenberg, B. F., "Power Generation, Operation and Control", Wiley., New York, 3rd ed., 2013

[6] Rizkyanto, Hardi.,"Dynamic *Economic dispatch* (DED) dengan Memperhatikan Ramp-Rate Menggunakan Metode Iterasi Lambda Berbasis Delphi", ITS Surabaya, 2015

Halaman ini sengaja dikosongkan

LAMPIRAN

```
unit Unit3;

interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls, ExtCtrls,strutils, jpeg, Math, Types;

const

        max_units = 20;
        max_order = 10;
        max_curve_points = 10;
        max_total_segments = 200;
        total_gen_tolerance = 0.01 ;
        ihr_tolerance = 0.000001 ;


        alpha : real = 0.5; {See note in loss matrix procedure}

type

        unit_array_real = array[1..max_units] of real;
        unit_name_array = array[1..max_units] of string;
        coefficients = array[0..max_order] of real;
        unit_poly_array = array[1..max_units] of coefficients;
        curve_points = array[0..max_curve_points] of real;
        unit_curve_array = array[1..max_units] of curve_points;
        system_ihr_array_real = array[1..max_total_segments] of real;
        system_ihr_array_integer = array[1..max_total_segments] of
integer;
        B_matrix = array[1..max_units] of unit_array_real;
        filename_array = string;
        curvetype_list = (poly,pinc,pio);
        losstype_list = (noloss,constpf,lossform);
        solution_type_list =
(lamsearch,tbllookup,Approximation Methods);
        schedtype_list = (totgen,totload);
```

```
var
       ioerr : boolean;
        ioval : integer;
    genname:unit_name_array;          {Generator name identifier}
    p:unit_array_real;                {Present value of P}
    pmin:unit_array_real;             {Minimum MW}
    pmax:unit_array_real;             {Maximum MW}
fungsifi:unit_array_real;       {Fungsi F}
    fungsie:unit_array_real;        {Fungsi E}
    fungsiy:unit_array_real;
  minihr:unit_array_real;       {Minimum unit incremental heat rate}
    maxihr:unit_array_real;       {Maximum unit incremental heat rate}
    fuelcost:unit_array_real;           {Fuel cost ( $/fuel unit )}
    coeff : unit_poly_array;            {Unit polynomial coefficients}
    ihr_mwpoint:unit_curve_array;     {MW points on ihr cost curve}
    ihr_cost:unit_curve_array;          {Cost points for ihr curve}
    io_mwpoint : unit_curve_array;     {MW Points on unit io curve}
    io_cost : unit_curve_array;        {Cost points on io curve}
    minput:unit_array_real;         {Minimum input for PINC curves }
    penfac:unit_array_real;           {Loss penalty factor}
    penfacD:unit_array_real;          {Loss penalty factor}

    b00:real;                     {Loss matrix constant}
    b0:unit_array_real;               {Loss matrix linear terms}
    b:B_matrix;                     {Loss matrix quadratic terms}
    seginccost:system_ihr_array_real;  {Segment inc cost for table look
up }
    segunit:system_ihr_array_integer;  {Unit associated with segment}
    segmw:system_ihr_array_real;       {MW contributed by segment}
    order:system_ihr_array_integer;    {Order routine output}
    ordvalue:system_ihr_array_real;    {Numbers to be ordered}
    inputfile:text;
    filename:filename_array;
    title1, title2 : string[80];
    print_output, diagflag, read_data : boolean;
    inputchar,quitflag : char;
    linenumber, ngen : integer;
    mw losses, schedmw, lambda : real;
    curvetype_input, losstype_input : string[8];
    number_string : string[20];
    curveorder : integer;
```

```
    curvetype : curvetype_list;
    losstype : losstype_list;
    solution_type : solution_type_list;
    schedtype : schedtype_list;
    pgenmax, pgenmin : real;
    math : real;
    FF:Text;
    NoGenerator:integer ;
    NomerOrder:integer;
    ModeDataPembangkit:integer;
     ProsesRun :boolean;
procedure datainput(namafile :string);
procedure ihr_ftn(i : integer; unitmw : real;  var unitihr : real );
procedure GetFileName(s :string;var d:string;var f:string;var e :string);
procedure datadump( var outfile:text );
procedure lambda_search_dispatch( var lambda : real );
procedure Approximation_Methods_dispatch( var lambda : real );
procedure loss_matrix_ftn;
procedure inverse_ihr_ftn(i : integer; unitihr : real; var unitmw : real );
procedure table_lookup_dispatch( var lambda : real );
procedure order_routine(        numorder : integer;
                          ordertable : system_ihr_array_real;
                    var   orderindex : system_ihr_array_integer );
procedure output_routine( var outfile : text;
                          lambda : real );
procedure prod_cost(       i : integer;
                  unitmw : real;
            var   unitcost : real );
procedure dataOutput(namafile :string);
function cekIoPoint(var unitG,Order:integer):boolean;
function cekIhrPoint(var unitG,Order:integer):boolean;
function CekEdcFile(filename:string): boolean;

implementation
function CekEdcFile(filename:string): boolean;
label keluar;
var s,d,f,e:string;
ff:text;
bc : boolean;
begin
bc:=false;
```

```
getfilename(filename,d,f,e);
 s:=trim(uppercase(f))+'.'+uppercase(e);
if s = 'EDC1.DAT' then bc:= true;
if s = 'EDCTEST.DAT' then bc:= true;

if s = 'EDC2.DAT' then bc:= true;
if s = 'EDC3.DAT'  then bc:= true;
if s = 'EDC4.DAT'  then bc:= true;
if s = 'EDC5.DAT'  then bc:= true;

if s = 'EX3A.DAT'  then bc:= true;
if s = 'EX3B.DAT'  then bc:= true;
if s = 'EX3C.DAT'  then bc:= true;
if s = 'EX3D.DAT'  then bc:= true;

if s = 'PR32.DAT'  then bc:= true;
if s = 'PR33.DAT'  then bc:= true;
if s = 'PR38.DAT'   then bc:= true;
if s = 'PR43A.DAT'  then bc:= true;
if s = 'PR43B.DAT'   then bc:= true;
if s = 'EX4D.DAT'   then bc:= true;

if bc = true then goto Keluar;

assign(ff,filename);
reset(ff);
readln(ff,s);
if pos('EDC FILE >>',s)>0 then bc:=true;
close(ff);

keluar:
cekedcfile:=bc;
end;
function cekIoPoint(var unitG,Order:integer):boolean;
label keluar;
var i,j:integer;
begin
  cekIopoint:=true;
  for i:=1 to ngen do
  begin
    for j:= 0 to curveorder-1 do begin
```

```
    if (io_mwpoint[i,j+1] - io_mwpoint[i,j])<=0 then
    begin
      cekIoPoint:=false;
      UnitG:=i;
      Order:=j;
      goto keluar;
    end;
   end;
  end;
keluar:
end;

function cekIhrPoint(var unitG,Order:integer):boolean;
label keluar;
var i,j:integer;
begin
  cekIhrpoint:=true;
  for i:=1 to ngen do
  begin
    for j:= 0 to curveorder-1 do begin
     if (ihr_mwpoint[i,j+1] - ihr_mwpoint[i,j])<=0 then
     begin
      cekIhrPoint:=false;
      UnitG:=i;
      Order:=j;
      goto keluar;
     end;
    end;
  end;
keluar:
end;

procedure prod_cost(       i : integer;
                    unitmw : real;
          var   unitcost : real );

    { Routine to return unit production cost given unit output in mw}
    { input : unit index = i}
    {         unit MW =  unitmw}
    { output: unit production cost = unitcost}
```

```
var
      j : integer;
      partmw, unitihr, segmentcost : real;


label return;

begin

case curvetype of
  poly :                    {Polynomial I/O curve}
     begin
     unitcost := 0;
     for j := curveorder downto 1 do

     unitcost := ( unitcost + coeff[ i,j ] ) * unitmw;
     unitcost := unitcost + coeff[ i,0 ];
     unitcost := unitcost * fuelcost[ i ];
     goto return
     end;

  pinc :                    {Piecewize incremental curve}
     begin
     unitcost := minput[ i ] * fuelcost[ i ];
     for j := 1 to curveorder do
       begin
       if (unitmw > ihr_mwpoint[ i,j ]) and ( j < curveorder ) then
                  {Calculate area under complete segment}
         begin
         segmentcost :=  ( (ihr_cost[i,j] + ihr_cost[i,j-1] )/2.0 ) *
                    ( ihr_mwpoint[i,j] - ihr_mwpoint[i,j-1] ) *
                          fuelcost[ i ];
         unitcost := unitcost + segmentcost;
         end
         else
                  {Calculate area under partial segment}
         begin
         partmw := (unitmw - ihr_mwpoint[i,j-1] )/
                     (ihr_mwpoint[i,j] - ihr_mwpoint[i,j-1] );
         unitihr := ihr_cost[i,j-1] +
                    ( ihr_cost[i,j] - ihr_cost[i,j-1] ) * partmw;
```

```
          segmentcost :=  ( (unitihr + ihr_cost[i,j-1])/ 2.0 ) *
                          ( unitmw - ihr_mwpoint[i,j-1] )  *
                          fuelcost[ i ];
          unitcost := unitcost + segmentcost;

          goto return;
          end;
        end;

      end;

    pio :                    {Piecewize I/O curve}
      begin
      for j := 1 to curveorder do
        begin
        if io_mwpoint[i,j] > unitmw then
          begin
          partmw := (unitmw-io_mwpoint[i,j-1] ) /
                          (io_mwpoint[i,j]-io_mwpoint[i,j-1] );
          unitcost := io_cost[i,j-1] +
                          ( io_cost[i,j]-io_cost[i,j-1] ) * partmw;
          unitcost := unitcost * fuelcost[ i ];
          goto return
          end;
        if j = curveorder then     {Unit is at or above pmax}
          begin
          unitcost := io_cost[ i, j ] * fuelcost[ i ];
          goto return
          end;
        end;
      end;
   end;  { End of case statement}

return:

end; { End procedure }

procedure output_routine( var outfile : text;
                          lambda : real );

var
```

```
        limittxt : string[5];
        totalgen, totalcost, totnewload ,newload: real;
        unitihr, unitinccost, unitcost, totalload : real;
        i : integer;

label return;

begin

writeln(outfile);
writeln(outfile,
    'generator  output   limit  inc cost penalty fact  operating cost');
writeln(outfile,
    '            mw            $/mwhr                $/hr       ');
writeln(outfile,
    '---------  ------   -----  --------  ------------  --------------');

totalgen := 0.0;
totalcost := 0.0;

for i := 1 to ngen do
  begin
  write(outfile,genname[i]:9);
  write(outfile, ' ',p[i]:6:1, '    ');
  limittxt := '     ';
  if abs( p[i] - pmin[i] ) < total_gen_tolerance then limittxt := 'min  ';
  if abs( p[i] - pmax[i] ) < total_gen_tolerance then limittxt := 'max  ' ;
  write(outfile, limittxt );

  ihr_ftn( i, p[ i ], unitihr) ;    {Get unit incremental heat rate}

  unitinccost := unitihr * fuelcost[i];
  write(outfile, unitinccost:9:4);
  write(outfile, '   ',penfac[i]:9:4, '    ');

  prod_cost( i, p[ i ], unitcost );          {Calculate unit operating cost}

  writeln(outfile, '   ',unitcost:9:12);
  totalgen := totalgen + p[i];
  totalcost := totalcost + unitcost;
  end;
```

```
writeln(outfile,
        '---------  ------                              ----------');
write(outfile, ' totals');
write(outfile, totalgen:9:1,
'                          ', totalcost:9:12);
writeln(outfile);
writeln(outfile, ' lambda = ', lambda:10:4 );
writeln(outfile);

if (schedtype = totgen ) and ( losstype <> lossform ) then goto return;

if schedtype=totload then
begin
  totalload := schedmw;
  totnewload := newload;
end ;

if schedtype=totgen then  totalload := totalgen - mwlosses;

if (solution_type = lamsearch) or (solution_type =
Approximationmethods) then
begin
  writeln(outfile, 'total load = ',totalload:10:1,
            '  total losses = ',mwlosses:10:1);
end
else
begin
  writeln(outfile, 'total load = ',totnewload:10:1,
            '  total losses = ',mwlosses:10:1);
end;
return:

end; { End procedure }

procedure order_routine(       numorder : integer;
                        ordertable : system_ihr_array_real;
                    var   orderindex : system_ihr_array_integer );

{ subroutine to order a list, least first                          }
{                                                  }
{ input numorder = the number of items to be ordered               }
```

```
{ input ordertable = the items to be ordered                    }
{ output orderindex = pointer to order value table                 }
{                                                                  }
{ nxt = Table used in order subroutine                             }
{                                                                  }
 var
    stop:boolean;
    i,j,top,last,indx:integer;
    nxt : system_ihr_array_integer;

 begin
   for i := 1 to numorder do begin
    if (i <= 1) then begin
      top := 1;
      nxt[ 1 ] := 0;
      end
    else begin
      j := top;
      last := 0;
      repeat
        stop := true;
        if (ordertable[ i ] > ordertable[ j ]) then begin
          last := j;
          j := nxt[ j ];
          stop  := (j = 0);
          if (stop) then begin
            nxt[ last ] := i;
            nxt[ i ] := 0;
          end
          end
        else begin
          if (j <> top) then begin
            nxt[ last ] := i;          { j not = top }
            nxt[ i ] := j;
            end
          else begin
            top := i;                  { j = top }
            nxt[ i ] := j;
          end;
        end;
      until stop;
```

```
    end;
  end;
  indx := 1;
  j := top;
  repeat
    orderindex[ indx ] := j;
    j := nxt[ j ];
    indx := indx + 1;
  until (j = 0);
 end;

procedure table_lookup_dispatch( var lambda : real );

    {  Routine to perform economic dispatch by table look up}

var
      targetgen, ptotal, segihr, unitihr : real;
      i, j, k, kseg, kunit, numsegments : integer;
      done : boolean;

label return;

begin
writeln(ff,'Table Look Up');
writeln(ff);
if curvetype <> pio then
    begin
      { tmsg:=' ERROR -- must have piecewize i/o curves to use table
lookup';
       }
      {form5.showmodal;}
    goto return
    end;

if losstype = lossform then loss_matrix_ftn;  { Calc losses and pen
factors}

if schedtype = totgen then targetgen := schedmw;
if schedtype = totload then targetgen := schedmw + mwlosses;

kseg := 0;                    {Build segment tables}
```

```
for i := 1 to ngen do
  begin
  for j := 1 to curveorder do
    begin
    kseg := kseg + 1;
    segihr := (io_cost[i,j]-io_cost[i,j-1]) /
                (io_mwpoint[i,j]-io_mwpoint[i,j-1]);
    seginccost[ kseg ] := segihr * fuelcost[ i ] * penfac[ i ];
    segunit[ kseg ] := i;
    segmw[ kseg ] := io_mwpoint[i,j] - io_mwpoint[i,j-1];
    end;
  end;
numsegments := kseg;

            {Set up for ordering routine}

for k := 1 to numsegments do
     ordvalue[k] := seginccost[k];
order_routine( numsegments, ordvalue, order );    {Call ordering
routine}
                                  {Result in order table}

        {Print segments table in incremental cost order}


  writeln(ff);
  writeln(ff,' segment number   inc cost      MW       unit ');
  writeln(ff,' --------------   ----------   ------    ----');

for k := 1 to numsegments do
begin
kseg := order[ k ];

   writeln(ff,'       ',kseg:3,'       ',seginccost[kseg]:10:4
            ,segmw[kseg]:10:1,'   ',segunit[kseg]:4)

end;


    ptotal := 0.0;
    for i := 1 to ngen do
```

```
        begin
        p[ i ] := pmin[ i ];
        ptotal := ptotal + p[ i ]
        end;

      done := false;
      k := 0;
      repeat
        k := k + 1;
        kseg := order[ k ];
        kunit := segunit[ kseg ];
        if ( ptotal + segmw[ kseg ] ) < targetgen then
          begin
          p[ kunit ] := p[ kunit ] + segmw[ kseg ];
          ptotal := ptotal + segmw[ kseg ]
          end
          else
          begin
          p[ kunit ] := p[ kunit ] + ( targetgen - ptotal );
          done := true
          end;
      until done;

      lambda := seginccost[ kseg ];

return:

end; { End procedure }

procedure inverse_ihr_ftn(       i : integer;
                        unitihr : real;
                      var unitmw : real );

      { Routine to return unit MW given unit incremental heat rate}
      { input : unit number = i}
      {         unit inc heat rate = unitihr}
      { output: unit MW stored in unitmw}

label  return;

var
```

```
        unitihr1, delihr, partihr, dihrdp : real;
        segmentihr : real;
        j, step : integer;

begin

if unitihr >= maxihr[ i ] then
   begin
   unitmw := pmax[ i ];
   goto return
   end;
if unitihr <= minihr[ i ] then
   begin
   unitmw := pmin[ i ];
   goto return
   end;

case curvetype of
  poly :                    {Polynomial curve}
     begin
     if curveorder <= 1 then
      begin
      if unitihr > coeff[ i,1 ] then unitmw:=pmax[i] else unitmw:=pmin[i];
      goto return
      end;

     if curveorder = 2 then
      begin
      unitmw := ( unitihr - coeff[ i,1 ] ) / ( 2.0 * coeff[ i,2 ] );
      goto return
      end;

          { for curves of order >= 3 search for unitmw using Newtons
method }

     unitmw := ( pmin[ i ] + pmax[ i ] )/ 2.0;
     step := 0;
     repeat
     step := step + 1;

     unitihr1 := 0;               {Calc unitihr at unitmw as unitihr1}
```

```
    for j := curveorder downto 2 do
      unitihr1 := ( unitihr1 + j * coeff[i,j] ) * unitmw;
    unitihr1 := unitihr1 + coeff[i,1];
    delihr := unitihr - unitihr1;
    if abs( delihr ) < ihr_tolerance then goto return;

    dihrdp := 0;                    {Calc curve second derivative}
    for j := curveorder downto 3 do
      dihrdp := ( dihrdp + j*(j-1) * coeff[i,j] ) * unitmw;
    dihrdp := dihrdp + 2.0 * coeff[ i,2 ];
    unitmw := unitmw + delihr/dihrdp;

    until step > 20;

    goto return
    end;

  pinc :                    {Piecewize incremental curve}
    begin
    j := 0 ;
    repeat
    j := j + 1;
    until (ihr_cost[i,j] > unitihr) or
        (j = curveorder);

    partihr := ( unitihr - ihr_cost[i,j-1] )/
                  ( ihr_cost[i,j] - ihr_cost[i,j-1] );
    unitmw := ihr_mwpoint[i,j-1] +
              ( ihr_mwpoint[i,j] - ihr_mwpoint[i,j-1] ) * partihr;
    goto return
    end;

  pio :                     {Piecewise I/O curve}
    begin
    j := 0;
    repeat
    j := j + 1;
    if j = curveorder then
      begin
      unitmw := io_mwpoint[i,j];
      goto return
```

```
      end;
    segmentihr :=(io_cost[i,j] - io_cost[i,j-1]) /
                  (io_mwpoint[i,j] - io_mwpoint[i,j-1]);
    until segmentihr >= unitihr;

    unitmw := io_mwpoint[ i,j-1 ];
    goto return
    end;

 end;  { End of case statement}

return:

end; { End procedure }

procedure loss_matrix_ftn;

    { Routine to calculate losses and penalty factors from loss formula}
    {  Input:   Table of unit generation p[ i ]}
    {          Loss formula b( i,j ), b0[ i ], b00}
    {  Output:  losses mwlosses and penalty factors penfac[ i ]}
    {  Note: loss formula expects p(i)'s to be in per unit so divide by
100.}

var
      i, j : integer;
      incloss, penfac_old, penfac_new  : real;
      mwlossesnew, rumus4,  lambdanew, rumus3, ptot : real;

label return;

begin

    mwlosses := b00;      {matrix Bloss}
    for i := 1 to ngen do
      begin
      mwlosses := mwlosses + b0[i] *
                  ( p[i]/100.0) + b[i,i] * sqr( p[i]/100.0 );
      for j := i+1 to ngen do
        mwlosses := mwlosses + 2.0 * ( p[i]/100.0) * ( p[j]/100.0 ) * b[i,j]
      end;
```

```
    mwlosses := mwlosses * 100.0;
   ptot := 0;
   rumus4 := 0;
   for i := 1 to ngen do
     end;

{ Note, in the formula above the penalty factor is "filtered" by the }
{ alpha filtering constant. If alpha is set to 1.0 no filtering action }
{ takes place, if alpha is 0.0 penfac is constant at 1.0 , suggested }
{ value for alpha is 0.5 to 0.9 }

return:

end; { End procedure }

procedure Approximation_Methods_dispatch( var lambda : real );
var
      i,j,n,k,lossiter : integer;
      lambdanew,lambdamax,lambdamin : real;
      lambdastart, deltalambda : real;
      targetgen, rumus3, rumus1 : real;
      unitihr, unitmw, totalgen : real;
      rumus2, fungsiy : real;
      unitcost, unitcostnew, unitcostnew1 : real;
      arccos : real;
      endloop : boolean;

begin
writeln (ff,'Approximation Methods');
writeln (ff);
for i := 1 to ngen do                { Mencari nilai P[i] awal}
 begin
 p[ i ] := ( pmin[ i ] + pmax[ i ] ) / 2.0
 end;

lossiter := 0;
endloop := false;

repeat                               {Top of iterative loop with losses}
```

```
  lambdamin := 10000.0;
  lambdamax := 0.0;
  mwlosses := 0 ;
  if losstype = lossform then   { Calc losses and pen factors}
  begin
    loss_matrix_ftn;
    writeln(ff);
    writeln(ff,' mw losses = ',mwlosses:10:1);
  end;

  for i := 1 to ngen do              {Calculate max and min lambdas}
    begin
    lambda := maxihr[ i ] * penfac[ i ] * fuelcost[ i ];
    if lambda > lambdamax then lambdamax := lambda;
    lambda := minihr[ i ] * penfac[ i ] * fuelcost[ i ];
    if lambda < lambdamin then lambdamin := lambda
    end;

    writeln(ff,' lambda limits = ',lambdamin:10:4,lambdamax:10:4);

  lambdastart := ( lambdamax + lambdamin ) / 2.0;
  deltalambda := ( lambdamax - lambdamin ) / 2.0;

   writeln(ff,' lambdastart deltalambda =
',lambdastart:10:4,deltalambda:10:4);
  {Set up total generation target}
  if schedtype = totgen then targetgen := schedmw;
  if schedtype = totload then targetgen := schedmw + mwlosses;
  {Lambda search}
  lambda := lambdastart;
  writeln(ff, ' targetgen = ',targetgen:10:1);

  n := 0;
  repeat                        {Top of lambda search loop}
     n := n + 1;
     totalgen := 0;
     for i := 1 to ngen do
       begin
       unitihr := lambda / ( penfac[ i ] * fuelcost[ i ] )  ;
       inverse_ihr_ftn( i, unitihr, unitmw );  {For given unitihr get
unitmw}
```

```
        p[ i ] := unitmw;
        totalgen := totalgen + p[ i ]
        end;
        writeln(ff,' lambda = ',lambda:10:4,'  totalgen = ',totalgen:10:1);
      if abs( totalgen - targetgen ) >= total_gen_tolerance then
        begin
      if totalgen > targetgen then lambda := lambda - deltalambda;
      if totalgen < targetgen then lambda := lambda + deltalambda;
      deltalambda := deltalambda / 2.0
      end;

    until ( abs( totalgen - targetgen ) < total_gen_tolerance ) or
        ( n > 20 ) ;

{See if another loss iteration is needed}

    if losstype <> lossform then endloop := true;
    lossiter := lossiter + 1;
    if lossiter > 10 then endloop := true

until endloop;

    {Start Approximation Methods}
for i := 1 to ngen do
      begin
      rumus1 := fungsifi[i]*(Pmin[i]-P[i]);
      rumus2 := abs (fungsie[i]*sin((rumus1/3.14)*180));
      unitcostnew := unitcost + rumus2 ;
      unitcostnew1 := unitcostnew1 + unitcostnew ;
     end;
     writeln(ff,'unitcostnew1=',unitcostnew1);
   end;
for i := 1 to ngen do
   begin
   lambdanew :=  coeff[i,1] + (2* coeff[i,2]*P[i]) + rumus3;
   writeln(ff,'lambdanew=',lambdanew);
   end;


end; {end Approximation Methods}
```

```
procedure GetFileName(s :string;var d:string;var f:string;var e :string);
var ss ,sd:string;
  n:integer;
begin
  ss := s;
  d := '';
  n :=  pos('\',ss);
  while  n>0 do
  begin
    sd := copy(ss,1,n);
    d := d+sd;
    delete(ss,1,n);
  n :=  pos('\',ss);
  end;
  n := pos('.',ss);
  if n = 0 then
  begin
     f := ss;
     e:= '';
  end else
  begin
     f:=copy(ss,1,n-1);
     delete(ss,1,n);
     e := ss;
  end;
end;
procedure IOCheck( linenumber : integer );
begin
  IOVal := IOresult;
  IOErr := (IOVal <> 0);
end; { of proc IOCheck }


procedure datainput(namafile :string);

label quit;

var i,j,k,jj : integer;
    a : char;

begin
```

```
print_output := True;

linenumber := 0;

assign(inputfile, namafile);
    iocheck( linenumber );
    if ioerr then goto quit;
{               }
{ Open data file    }
{               }
reset(inputfile);
    iocheck( linenumber );
    if ioerr then goto quit;
{               }
{ Read file header }
{               }
linenumber := linenumber + 1;
readln( inputfile, title1 );
    iocheck( linenumber );
    if ioerr then goto quit;
linenumber := linenumber + 1;
readln( inputfile, title2 );
    iocheck( linenumber );
    if ioerr then goto quit;

    {                                       }
    {  Read Number of generators, curve type, loss type  }
    {                                       }
linenumber := linenumber + 1;
read( inputfile, ngen );
    iocheck( linenumber );
    if ioerr then goto quit;

repeat
read( inputfile, inputchar );
      iocheck( linenumber );
      if ioerr then goto quit;
until inputchar <> ' ';
curvetype_input := inputchar;
repeat
read( inputfile, inputchar );
```

```
      iocheck( linenumber );
      if ioerr then goto quit;
if inputchar <>' ' then curvetype_input := curvetype_input + inputchar;
until inputchar = ' ';

read( inputfile, curveorder );
      iocheck( linenumber );
      if ioerr then goto quit;

repeat
read(inputfile, inputchar );
     iocheck( linenumber );
     if ioerr then goto quit;
until inputchar <> ' ';
losstype_input := inputchar;
readln(inputfile);

 {                                    }
 {  Set up internal variables for curvetype and losstype  }
 {                                    }

if (curvetype_input = 'poly') or
   (curvetype_input = 'POLY') then curvetype := poly;

if (curvetype_input = 'pinc') or
   (curvetype_input = 'PINC') then curvetype := pinc;

if (curvetype_input = 'pio' ) or
   (curvetype_input = 'PIO' ) then curvetype := pio;

if (losstype_input = 'n'  ) or
   (losstype_input = 'N'  ) then losstype := noloss;

if (losstype_input = 'c'  ) or
   (losstype_input = 'C'  ) then losstype := constpf;

if (losstype_input = 'l'  ) or
   (losstype_input = 'L'  ) then losstype := lossform;


        {                }
```

```
        { Read generator data }
        {                   }

for i := 1 to ngen do
  begin  { Read generator name }
    linenumber := linenumber + 1;
    repeat
    read( inputfile, inputchar );
         iocheck( linenumber );
         if ioerr then goto quit;
    until inputchar <> ' ';
    genname[i] := inputchar;
    repeat
    read( inputfile, inputchar );
         iocheck( linenumber );
         if ioerr then goto quit;
    if inputchar <>' ' then genname[i] := genname[i] + inputchar;
    until inputchar = ' ';
      {                         }
      { Read generator min, max, fuelcost }
      {                         }
    readln( inputfile, pmin[i], pmax[i], fuelcost[i] );
         iocheck( linenumber );
         if ioerr then goto quit;
      {                         }
      { Read generator cost curve data    }
      {                         }
    case curvetype of
     poly :
          begin  { read polynomial curve data}
          for j := 0 to curveorder do
            begin
              linenumber := linenumber + 1;
              readln( inputfile, coeff[i,j] );
               iocheck( linenumber );
               if ioerr then goto quit;
            end;
          end;

     pinc :
          begin  { read piecewise incremental cost curve data}
```

```
            linenumber := linenumber + 1;
            readln( inputfile, minput[i] );
              iocheck( linenumber );
              if ioerr then goto quit;
            for j := 0 to curveorder  do
              begin
                linenumber := linenumber + 1;
                readln( inputfile, ihr_mwpoint[i,j], ihr_cost[i,j] );
                  iocheck( linenumber );
                  if ioerr then goto quit;
              end;
            end;

      pio  :
            begin  { read piecewise I/O curve data}
            for j := 0 to curveorder do
              begin
                linenumber := linenumber + 1;
                readln( inputfile, io_mwpoint[i,j], io_cost[i,j] );
                  iocheck( linenumber );
                  if ioerr then goto quit;
              end;
            end;

        end;  { End of case statement}
    end;
        {           }
        { Read loss data }
        {           }

case losstype of

  noloss :
        begin
        for i := 1 to ngen do      { Init penalty factors}
            penfac[ i ] := 1.0
        end;

  constpf  :
      begin   { read constant penalty factor data}
        linenumber := linenumber + 1;
```

```
    for i := 1 to  ngen do
     begin
     if i < ngen then
     read( inputfile, penfac[i] )
     else
     readln( inputfile, penfac[ngen] );
        iocheck( linenumber );
        if ioerr then goto quit
     end;
   end;

lossform :

   begin  { read loss formula data}

   linenumber := linenumber + 1;
   readln( inputfile, b00 );
        iocheck( linenumber );
        if ioerr then goto quit;

   linenumber := linenumber + 1;
   for j := 1 to  ngen  do
    begin
    if j < ngen then
      read( inputfile, b0[ j ] )
      else
      readln( inputfile, b0[ngen] );
       iocheck( linenumber );
       if ioerr then goto quit;
    end;

   for i := 1 to ngen do
    begin
    linenumber := linenumber + 1;
    for j := 1 to ngen  do
     begin
     if j < ngen then
        read( inputfile, b[ i, j ] )
        else
        readln( inputfile, b[ i, ngen ] );
        iocheck( linenumber );
```

```
          if ioerr then goto quit
        end;
       end;
       for i := 1 to ngen do      { Init penalty factors}
           penfac[ i ] := 1.0
    end;
  end;  { End of case statement}
{                          }
{  End of data input, close file   }
{                          }

close ( inputfile );

{A very useful table is the incremental cost at the max and min of each
unit. }
{These are calculated and stored in tables maxihr and minihr.}
{Also calculate the max and min generation}



quit:

end;   { end of data input }

procedure dataOutput(namafile :string);
var i,j,k,jj : integer;
   a : char;
   d,e,f:string;
  var inputfile:text;
begin
getfilename(namafile,d,f,e);
assign(inputfile, namafile);
rewrite(inputfile);
writeLn( inputfile, 'EDC FILE >> '+f+'.'+e );
writeln( inputfile,
'=======================================');
write( inputfile, ngen );
write(inputfile,' ');

if  curvetype = poly then write(inputfile,'POLY');
if  curvetype = pinc then write(inputfile,'PINC');
```

```
if  curvetype = PIO then write(inputfile,'PIO');
write(inputfile,' ');
write(inputfile,curveorder);
write(inputfile,' ');

if losstype = noloss  then write(inputfile,'NOLOSS');
if losstype = constpf  then write(inputfile,'CONSTPF');
if losstype = lossform  then write(inputfile,'LOSSFORM');
writeln(inputfile);
for i := 1 to ngen do
  begin
    write(inputfile,genname[i]);
    write(inputfile,' ');
    writeln( inputfile, pmin[i]:15:6, pmax[i]:15:6,
fuelcost[i]:15:6,fungsifi[i]:15:6, fungsie[i]:15:6 );
   case curvetype of
     poly :
          begin
          for j := 0 to curveorder do
            begin
              writeln( inputfile, coeff[i,j]:15:6 );
            end;
          end;

     pinc :
          begin
          writeln( inputfile, minput[i]:15:6 );
          for j := 0 to curveorder  do
            begin
              writeln( inputfile, ihr_mwpoint[i,j]:15:6, ihr_cost[i,j]:15:6 );
            end;
          end;

     pio  :
          begin
          for j := 0 to curveorder do
            begin
              writeln( inputfile, io_mwpoint[i,j]:15:6, io_cost[i,j]:15:6);
            end;
          end;
```

```
      end;
   end;

case losstype of

 noloss :
      begin
      for i := 1 to ngen do      { Init penalty factors}
           penfac[ i ] := 1.0
      end;

 constpf  :
    begin
      for i := 1 to  ngen do
      begin
      if i < ngen then
      begin
      write( inputfile, penfac[i]);
      write(inputfile,' ');

      end
      else writeln( inputfile, penfac[ngen]);
      end;
    end;

 lossform :

    begin

    linenumber := linenumber + 1;
    writeln( inputfile, b00);
     for j := 1 to  ngen  do
      begin
      if j < ngen then
        begin
        write( inputfile, b0[ j ]);
        write(inputfile,' ');
        end
        else
        writeln( inputfile, b0[ngen]);
      end;
```

```
    for i := 1 to ngen do
      begin
      for j := 1 to ngen  do
       begin
       if j < ngen then
          begin
          write( inputfile, b[i, j]);
          write(inputfile,' ');

          end
          else
          writeln( inputfile, b[i,ngen]);
       end;
      end;

      for i := 1 to ngen do      { Init penalty factors}
           penfac[ i ] := 1.0

    end;

 end;


close ( inputfile );



end;

procedure ihr_ftn(          i : integer;
                  unitmw : real;
             var unitihr : real );

    { Routine to return unit incremental heat rate given unit output in
MW }
    { input : unit index = i}
    {         unit mw = unitmw}
    { output: unit incremental heat rate = unitihr}

var
```

```
        partmw : real;
        j : integer;

begin

case curvetype of
  poly :                {Polynomial I/O curve}
     begin
     unitihr := 0.0;
     for j := curveorder downto 2 do
       unitihr := ( unitihr + j * coeff[ i,j ] ) * unitmw;
     unitihr := unitihr + coeff[ i,1 ]
     end;

  pinc :                {Piecewize incremental curve}
     begin
     j := 0;
     repeat
     j := j + 1;
     until (ihr_mwpoint[ i,j ] > unitmw) or (j = curveorder);

     partmw := (unitmw - ihr_mwpoint[i,j-1] )/
                  (ihr_mwpoint[i,j] - ihr_mwpoint[i,j-1] );
     unitihr := ihr_cost[i,j-1] + ( ihr_cost[i,j] - ihr_cost[i,j-1] ) * partmw
     end;

  pio  :                {Piecewize I/O curve}
     begin
     j := 0;
     repeat
     j := j + 1;
     until (io_mwpoint[ i,j ] > unitmw) or (j = curveorder);

     unitihr := (io_cost[i,j] - io_cost[i,j-1] ) /
                    ( io_mwpoint[i,j] - io_mwpoint[i,j-1] )
     end;
  end;  { End of case statement}


end; { End ihr_ftn procedure }
```

```
procedure datadump( var outfile:text );


var
  i,j : integer;

begin

writeln(outfile);
writeln(outfile, title1);
writeln(outfile, title2);
writeln(outfile);
writeln(outfile ,' number of generator units = ',ngen );

case curvetype of
 poly : writeln(outfile ,' unit curve type = poly ');
 pinc : writeln(outfile ,' unit curve type = pinc ');
 pio  : writeln(outfile ,' unit curve type = pio');
 end;  { End of case statement}

writeln(outfile ,' curve order = ',curveorder);

case losstype  of
 noloss :   writeln(outfile ,' network loss representation = noloss ');
 constpf :  writeln(outfile ,' network loss representation = constpf ');
 lossform : writeln(outfile ,' network loss representation = lossform ');
 end;  { End of case statement}

for i := 1 to ngen do
 begin
  writeln(outfile);
  write(outfile, genname[i],' limits = ',pmin[i]:7:2, ' ',pmax[i]:7:2 );
  writeln(outfile, ' fuelcost = ',fuelcost[i]:10:4 );
   case curvetype of
      poly :
         begin
         writeln(outfile,'  polynomial coefficients' );
         for j := 0 to curveorder do
           begin
           writeln(outfile, coeff[i,j]:15:6);
           end;
```

```
          end;
        pinc :
          begin
          writeln(outfile,' incremental cost curve points');
          writeln(outfile,'input at pmin = ',minput[i]:10:2);
          for j := 0 to curveorder do
           begin
           writeln(outfile,ihr_mwpoint[i,j]:9:2,ihr_cost[i,j]:9:3 )
           end;
          writeln(outfile);
          end;
        pio :
          begin
          writeln(outfile,' cost curve points');
          for j := 0 to curveorder do
           begin
           writeln(outfile,io_mwpoint[i,j]:9:2,'  ',io_cost[i,j]:9:3 )
           end;
          writeln(outfile);
          end;
        end; {end of case statement }
  end;
case losstype  of

  constpf :
            begin
            writeln(outfile);
            writeln(outfile,' Penalty Factors');
            for i := 1 to ngen do
              begin
              writeln(outfile,' penalty factor ',i,' ',penfac[i]:10:3)
              end;
            writeln(outfile);
            end;

  lossform :
            begin
            writeln(outfile);
            writeln(outfile,' Loss Formula');
            writeln(outfile,'B00 = ',b00:10:4);
            writeln(outfile);
```

```
            writeln(outfile,'B0 = ');
            for i := 1 to ngen do
              if i < ngen then
              write(outfile,b0[i]:10:4,' ')
              else
              writeln(outfile,b0[i]:10:4);
            writeln(outfile);
            writeln(outfile,' B = ');
            for i := 1 to ngen do
              begin
                 for j := 1 to ngen do
                   if j < ngen then
                   write(outfile,b[i,j]:10:4,' ')
                   else
                   writeln(outfile,b[i,ngen]:10:4);
              end;
            writeln(outfile)
            end;
  end;  { End of case statement}
   //if solution_type = lamsearch then writeln(outfile,'using lambda
search');
   //if solution_type = tbllookup then writeln(outfile,'using tablelookup');

    writeln(outfile);
    if schedtype = totgen then
      writeln(outfile, ' total generation schedule = ',schedmw:10:1)
    else
      writeln(outfile, ' total load schedule = ', schedmw:10:1);
    if losstype = lossform then writeln(outfile, ' using loss formula ')
                    else writeln(outfile,' losses neglected');
    writeln(outfile);
end; { End procedure }

end.
```

RIWAYAT HIDUP PENULIS

SILSILATIL MAGHFIROH, lahir di Sampang, 18 April 1990. Penulis tamat dari bangku sekolah dasar di SDN Dalpenang I Sampang pada tahun 2002 dan melanjutkan di sekolah menengah pertama di SMPN 1 Sampang, lulus tahun 2005. Setelah lulus SMP, penulis melanjutkan sekolah ke SMAN 1 Sampang. Setelah lulus pada tahun 2008, penulis melanjutkan studi D3 di Teknik Elektro ITS Surabaya dengan program studi Komputer Kontrol dan lulus pada tahun 2011, kemudian melanjutkan S1 di Institut Teknologi Sepuluh Nopember (ITS) Surabaya jurusan Teknik elektro dan mengambil konsentrasi dalam Bidang Studi Teknik Sistem Tenaga. Penulis dapat dihubungi melalui alamat email :silsymaghfiroh@gmail.com