TUGAS AKHIR – TE141599

PENGEMBANGAN *SOFTWARE ECONOMIC DISPATCH* SKALA BESAR DENGAN ALGORITMA *ENHANCED LAMBDA ITERATION*

Santi Triwijaya
NRP 2213 106 029

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

JURUSAN TEKNIK ELEKTRO
FakultasTeknologiIndustri
Institut Teknologi Sepuluh Nopember
Surabaya 2016

FINAL PROJECT - TE 141599

THE DEVELOPMENT OF LARGE SCALE ECONOMIC DISPATCH SOFTWARE USING ENHANCED LAMBDA ITERATION

Santi Triwijaya
2213 106 029

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

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Industry Technology
Sepuluh Nopember Institute of Technology
Surabaya 2016

PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "**Pengembangkan Software Economic Dispatch Skala Besar dengan Algoritma *Enhanced Lambda Iteration***" adalah benar-benar hasil karya mandiri penulis, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang penulis akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka.Apabila ternyata pernyataan ini tidak benar, penulis bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, 25 Januari 2016

Santi Triwijaya
NRP. 2213106029

PENGEMBANGAN *SOFTWARE ECONOMIC DISPATCH* SKALA BESAR DENGAN ALGORITMA *ENHANCED LAMBDA ITERATION*.

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 : 194907151974121001	NIP: 197411292000121001

SURABAYA
JANUARI. 2016

FINAL PROJECT - TE 141599

THE DEVELOPMENT OF LARGE SCALE ECONOMIC DISPATCH SOFTWARE USING ENHANCED LAMBDA ITERATION

Santi Triwijaya
2213 106 029

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

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Industry Technology
Sepuluh Nopember Institute of Technology
Surabaya 2015

PENGEMBANGAN *SOFTWARE ECONOMIC DISPATCH* SKALA BESAR DENGAN ALGORITMA *ENHANCED LAMBDA ITERATION*.

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 : 194907151974121001	NIP: 197411292000121001

SURABAYA
JANUARI 2016

PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "**Pengembangkan Software Economic Dispatch Skala Besar dengan Algoritma *Enhanced Lambda Iteration***" adalah benar-benar hasil karya mandiri penulis, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang penulis akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka.Apabila ternyata pernyataan ini tidak benar, penulis bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, 25 Januari 2016

Santi Triwijaya
NRP. 2213106029

ABSTRAK

Pengembangkan Software Economic Dispatch Skala Besar dengan Algoritma Enhanced Lambda Iteration
Santi Triwijaya
2213 106 029

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

Abstrak:

Pada tugas akhir ini, algoritma *Enhanced Lambda Iteration* akan ditambahkan sebagai salah satu metode penyelesaian *economic dispatch* dalam *software* powegen pada menu EDC. Dimana *software* powergen adalah software perhitungan economic dispatch berbasis Delphi yang dikembangkan oleh mahasiswa teknik elektro institut Teknologi Sepuluh Nopember. Dengan mempergunakan algoritma *Enhanced Lambda Iteration* (ELI) ini, permasalahan dengan konsep penambahan kriteria *incremental cost* dapat dilakukan untuk menentukan nilai awal lambda dan juga mengatasi permasalahan penentuan batasan *equality* dan *inequality* dengan mudah. Sedangakan penyelesaian *economic dispatch* (ED) dengan *Lambda* dapat mengakibatkan konvergensi lambat karena pemilihan nilai awal *(incremental cost)* yang tidak tepat.

Kemudian akan dilakukan pengetesan dengan algoritma ini dalam tiga tes sistem untuk menunjukkan kelayakan algoritma [2] [7]. Dari hasil pengujian terlihat bahwa hasil perhitungan biaya pembangkitan pada software telah sesuai dengan refrensi [2] dan dapat digunakan sebagai software perhitungan untuk skala kecil hingga skala besar pembangkit.

Kata kunci: *software* powegen, *economic dispatch* , algoritma *Enhanced Lambda Iteration, Lambda Iteration*, *increamental cost*,

--Halaman ini sengaja dikosongkan—

ABSTRACT

High Scale Economic Dispatch Software Developing with Enhanced Lambda Iteration Algorithm

Santi Triwijaya
2213 106 029
Advisor 1 : Prof. Ir. Ontoseno Penangsang M.Sc Ph.D
Advisor 2 : Dr. Rony Seto Wibodo, ST., MT.

Abstract:

In this Final Project, Enhanced Lambda Iteration algoritm will be added as one of the solving methode for economic dispatch study case on Powergen software,sub menu EDC. Powergen is an economic dispatch calculation software based on Delphi programming language that developed by electrical engineering student of Institute Technology Sepuluh Nopember. Enhanced Lambda Iteration (ELI) in economic dispatch with added incremental cost parameter can be solved by setting the initial lambda first and then solving the problem by determining equality and inequality constraint easily. Economic Dispatch problem solving using conventional lambda method will need time to be solved due to inaccurate setting initial value

After the construction of ELI program finished, the program will be tested on 3 system test for ensure the algorithm feasibility [2][7]. From the testing result, it can be determined that the final cost of economic dispatch on software already equal with the reference [2]. By then, the program is valid and can be used as economic dispatch software for low scale until high scale generation systems.

Keyword: software powergen, economic dispatch , Enhanced Lambda Iteration Algorithm, Lambda Iteration, increamental cost,

--Halaman ini sengaja dikosongkan--

KATA PENGANTAR

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

"Pengembangkan Software Economic Dispatch Skala Besar dengan Algoritma Enhanced Lambda Iteration"

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.

Dalam penyusunan Tugas Akhir ini, penulis tidak lepas dari petunjuk, bimbingan, bantuan, dan dukungan dari berbagai pihak. Pada kesempatan ini penulis hendak menyampaikan rasa terima kasih kepada pihak-pihak yang telah memberi bantuan baik itu berupa moral maupun material, langsung maupun tidak langsung :

1. Allah SWT atas limpahan Rahmat dan Petunjuk-Nya serta Nabi Muhammad SAW atas tuntunan jalan-Nya.
2. Aysh dan Ibu saya yang telah membesarkan saya dan menyayangi saya serta membiayai saya hingga kuliah
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. Semua pihak yang telah membantu baik secara langsung maupun tidak langsung, yang tidak mungkin saya sebutkan satu per satu

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

Surabaya, 25 januari 2016

Penulis

--Halaman ini sengaja dikosongkan--

DAFTAR ISI

ABSTRAK i
ABSTRACT iii
KATA PENGANTAR v
DAFTAR ISI vii
DAFTAR GAMBAR ix
DAFTAR TABEL xi
BAB I 1
1.1. Latar Belakang 1
1.2. Tujuan Penelitian 1
1.3. Permasalahan 1
1.4. Batasan Masalah 2
1.5. Metode Penelitian 2
1.6. Sistematika Penulisan 3
1.7. Manfaat dan relefansi 3
BAB II 5
2.1. Unit Pembangkit Termal 5
2.1.1 Karakteristik Unit Pembangkit Termal 6
2.2. Economic *Dispatch* 9
BAB III 15
3.1. Alur Pengerjaan 15
3.2. Perencanaan pengembangan software perhitungan economic dispatch 17
3.3. Algoritma Enhanced Lambda Iteration pada Economic Dispatch 21
3.3.1 Persamaan Economic Dispatch 21
3.4. Uji Kasus 24

3.4.1 Tes Sistem 1 25
3.4.2 Tes Sistem 2 26
3.4.3 Tes Sistem 3 29
BAB IV 31
4.1 Hasil Pengujian Tes Sistem 1 31
4.1.1 Perbandingan Hasil dengan Perhitungan Manual 34
4.1.2 Perbandingan Hasil software dengan Algoritma E*nhanced Lambda Iteration* dan Software dengan *Lambda Iteration* 34
4.1.3 Perbandingan Hasil *running* kasus 1 dengan Algoritma E*nhanced Lambda Iteration* dan Q*uadratic Programing.* 35
4.2 Hasil Pengujian Tes Sistem 2 36
4.2.1 Perbandingan Hasil dengan Perhitungan Manual 39
4.2.2 Perbandingan Hasil software dengan Algoritma E*nhanced Lambda Iteration* dan Software dengan *Lambda Iteration* 40
4.2.3 Perbandingan Hasil *running* kasus 2 dengan Algoritma E*nhanced Lambda Iteration* dan Q*uadratic Programing.* 41
4.3 Hasil Pengujian Tes Sistem 3 41
4.3.1 Perbandingan Hasil software dengan Algoritma E*nhanced Lambda Iteration* dan Software dengan *Lambda Iteration* 44
4.3.2 Perbandingan Hasil *running* kasus 2 dengan Algoritma E*nhanced Lambda Iteration* dan Q*uadratic Programing.* 47
BAB V 49
5.1 Kesimpulan 49
5.2 Saran 49
DAFTAR PUSTAKA 51
LAMPIRAN 53
RIWAYAT HIDUP PENULIS 1

DAFTAR GAMBAR

Gambar 2. 1 Pemodelan boiler-turbin-generator [4] 5
Gambar 2. 2 Karakteristik *input-output* unit pembangkit [4] 7
Gambar 2. 3 Kurva input-output pembangkit thermal[4] 8
Gambar 2. 4 Karakteristik *incremental rate* 9
Gambar 2. 5 Pembangkit yang terhubung dengan satu bus [3] 10
Gambar 3. 1Flowchart Penyelesaian Tugas Akhir 16
Gambar 3. 2 Tampilan utama powergen 17
Gambar 3. 3 Tampilan utama menu EDC 17
Gambar 3. 4 Menu input-an data pembangkit di EDC 18
Gambar 3. 5 Menampilkan data pada menu EDC 18
Gambar 3. 6 Tampilan *Set up Solution* menu EDC 19
Gambar 3. 7 Tampilan hasil perhitungan EDC 19
Gambar 3. 8 Flowchart Penyelesaian ELI 24
Gambar 3.9 Single line diagram 6 unit pembangit tes sistem 1 25
Gambar 3.10 Single line diagram 15 unit pembangit tes sistem 2 27
Gambar 3.11 Single line diagram 40 unit pembangit tes sistem 3 30
Gambar 4.1 Load Profile Harian kasus 1 32
Gambar 4. 2 Tampilan data pada tes sistem1 di menu EDC 32
Gambar 4. 3 Pengujian kasus 1 tampilan *set up solution* 33
Gambar 4. 4 Hasil optimasi tes sistem 1 33
Gambar 4. 5 Hasil *running software* dengan *lambda* 35
Gambar 4.6 *load profile* harian kasus 2 38
Gambar 4. 7 Tampilan data pada tes sistem2 di menu EDC 38
Gambar 4. 8 Hasil optimasi tes sistem 2 39
Gambar 4. 9 *Load Profile* haasrian Kasus 3 43
Gambar 4. 10 Tampilan data pada tes sistem3 di menu EDC 44

--Halaman ini sengaja dikosongkan--

DAFTAR TABEL

Tabel 3. 1 Data kasus 1 25
Tabel 3. 2 Hasil perhitungan kasus 1 dengan beban 1263MW 26
Tabel 3. 3 Hasil akhir perhitungan biaya pembangkitan kasus 1 26
Tabel 3. 4 Data kasus 2 28
Tabel 3. 5 Increamental cost function kasus 2 28
Tabel 3. 6 Data kasus 3 29
Tabel 4. 1 Kapasitas pembangkitan dan fungsi biaya tes sistem 1 31
Tabel 4.2 Load Profile Harian kasus 1 31
Tabel 4. 3 Perbandingan ELI dengan perhitungan manual kasus 1 34
Tabel 4. 4 Perbandingan hasil *software* ELI dengan *lambda.*kasus 1 ... 35
Tabel 4. 5 Perbandingan ELI dengan *Quadratic Programming* kasus 136
Tabel 4. 6 Kapasitas pembangkitan dan fungsi biaya tes sistem 2 36
Tabel 4. 7 Increamental cost function tes sistem 2 37
Tabel 4. 8 Tabel *load profile* harian kasus 2 37
Tabel 4. 9 Variasi nilai lambda dengan iterasi 39
Tabel 4. 10 Hasil perhitungan manual iterasi ke 6 40
Tabel 4. 11 Perbandingan hasil *software* ELI dengan *lambda kasus 2* . 41
Tabel 4. 12 Perbandingan ELI dengan QP kasus 2 41
Tabel 4. 13 Karakteristik 40 unit pembangkit 42
Tabel 4.14 *Load profile* harian kasus 3 43
Tabel 4. 15 Variasi nilai lambda tiap iterasi tes sistem 3 44
Tabel 4. 16 Hasil running software dengan ELI kasus 3 45
Tabel 4. 17 Hasil running *software* dengan *lambda kasus 3*. 46
Tabel 4. 18 Perbandingan ELI dengan QP kausu 3 47

--Halaman ini sengaja dikosongkan--

BAB I
PENDAHULUAN

1.1. Latar Belakang

Analisis aliran daya optimal untuk meminimalkan biaya pembangkitan biasa dikenal dengan istilah *Economic Dispatch* [1]. *Economic dispatch* adalah pembagian pembebanan pada unit-unit pembangkit yang ada dalam sistem secara optimal ekonomi, pada harga beban sistem tertentu. Dengan penerapan *economic dispatch,* maka akan didapatkan biaya pembangkitan yang minimum terhadap produksi daya listrik yang dibangkitkan unit-unit pembangkit pada suatu sistem kelistrikan dengan tetap memperhatikan *constrain* [2].

Untuk dapat menyelesaikan permasalah Economic Dispatch dapat dengan mempergunakan algoritma *Enhanced Lambda Iteration* [2]. Dengan algoritma ini, akan di dapatkan hasil kovergen yang lebih cepat dibanding dengan *Lambda Iteration* biasa. Karena pada kedua metode ini memiliki perbedaan dalam penentuan nilai awal dari lambda.

Aplikasi yang memiliki *user interface* untuk menghitung *Economic Dispatch* masih sangat terbatas. Maka, dibutuhkan aplikasi software yang memiliki *user interface* yang baik. Salah satu software yang memiliki *user interface* yang baik adalah Delphi [6]. Delphi merupakan sebuah bahasa pemprograman yang menawarkan pengembangan perangkat lunak computer berbasis visual dengan cepat. Oleh sebab itu, pada tugas akhir ini akan dirancang aplikasi perhitungan ED dengan metode *Enhanced Lambda Iteration* yang mempergunakan Delphi

1.2. Tujuan Penelitian

Penelitian pada tugas akhir ini memiliki tujuan sebagai berikut:

- Mengembangkan software untuk mendapatkan biaya pembangkitan yang optimal sehingga mempermudah user dalam menyelesaikan perhitungan *economic dispatch* dengan *Enhanced Lambda Iteration*.

1.3. Permasalahan

Permasalahan yang akan dibahas dalam tugas akhir ini adalah :

1. Bagaimana *User Interface* yang sesuai untuk perhitungan ED dengan *Enhanced Lambda Iteration*.
2. Bagaimana kinerja dari software yang dihasilkan.
3. Bagaimana hasil perbandingan perhitungan ED dengan Algoritma *Enhanced Lambda Iteration* terhadap *Lambda Iteration* pada software.

1.4. Batasan Masalah

Agar tugas ini tidak menyimpang dari ketentuan yang digariskan, maka ditetapkan batasan masalah sebagai berikut:

1. Metode optimmasi yang dipakai *Enhanced Lambda Iteration* pada software *Economic Dispatch* yang sudah ada.
2. Tanpa memperhitungkan losses dan perubahan beban.
3. Delphi digunakan untuk aplikasi perhitungan EDC.

1.5. Metode Penelitian

Untuk mendapatkan biaya pembangkitan teroptimum (*Economic Dispatch)* dengan algoritma *Enhanced Lambda Iteration* bedasarkan langkah-langkah sebagi berikut :

1. Study literatur
   Dilakukan dengan mencari referensi dari buku ataupun peper yang terkait dengan topik mengenai *Economic Dispatch.*
2. Menentukan metode
   Penentuan metode ini dipergunakan untuk mendapatkan penyelesaian dari masalah *Economic Dispatch*. Metode yang akan dipergunakan adalah *Enhanced Lambda Iteration.*
3. Menentukan Plan
   Plan dapat diperoleh berdasarkan karakteristik pembangkit yang terdapat pada peper IEEE.
4. Aplikasi
   Dari data-data tersebut dapat dilakukan perhitungan ED dan di aplikasikan dengan menggunakan Delphi.
5. Analisa
   Berdasarkan hasil dari aplikasi tersebut dapat dilakukan analisa dan dibandingkan *Enhanced Lambda Iteration* terhadap *lambda iteration*.

6. Kesimpulan
   Berkaitan tentang hasil yang dicapai dari software yang dirancang.

1.6. Sistematika Penulisan

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, sistematika penulisan serta manfaat dan relefansi.
2. BAB 2 berisi teori penunjang yang membahas tentang Sistem kelistrikan, *Economic Dispatch*, metode *Enhanced Lambda Iteration* dan 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. Manfaat dan relefansi

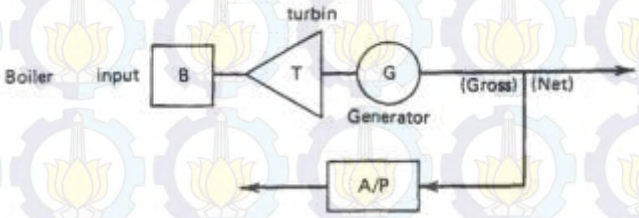
1. Bagi perusahaan listrik
   Tugas akhir ini diharapkan dapat memberikan manfaat bagi perusahaan listrik dalam memutuskan pola pembangkitan yang dilakukan sehingga mendapatkan biaya pembangkitan yang lebih baik.
2. Bagi bidang ilmu pengetahuan dan mahasiswa lain
   Tugas akhir ini diharapkan dapat membantu perkembangan ilmu pengetahuan dengan menjadi alat bantu perhitungan ED yang handal dan mudah digunakan.

--Halaman ini sengaja dikosongkan--

BAB II
ECONOMIC DISPATCH

2.1. Unit Pembangkit Termal

Pembangkit tenaga listrik dapat dibedakan menjadi beberapa jenis sesuai dengan bahan bakar yang digunakan. Salah satu diantaranya adalah pembangkit listrik tenaga panas atau *thermal*. Pengoperasian suatu pembangkit termal sangat tergantung pada bahan bakar, dengan demikian hal tersebut perlu mendapatkan perhatian khusus, karena sebagian besar biaya operasi yang dikeluarkan adalah untuk keperluan bahan bakar[1]. Maka, pembangkitan *thermal* erat kaitannya dengan *economic dispatch* untuk melakukan efisiensi dan operasi optimasi ekonomi serta perencanaan besar daya yang dibangkitkan [4]. Pada Gambar 2. 1 mempresentasikan pembangkit thermal sederhana.



Gambar 2. 1 Pemodelan boiler-turbin-generator [4]

Pembangkit *thermal* dapat didefinisikan dalam beberapa bagian utama meliputi:

- *Boiler*

Dalam pembangkitan sistem tenaga listrik *boiler* berfungsi sebagai *steam supply* [1] yang dihasilkan dari pembakaran sumber energi primer [3].

- *Turbin*

Merupakan alat atau mesin penggerak mula (*prime mover*), yang menghasilkan energi mekanik. Energi mekanik dapat dihasilkan karena adanya tekanan atau aliran yang kontinyu. Seperti pada *prime mover* yang dapat berupa turbin hidrolik pada air terjun [3].

- *Generator*

Generator adalah Salah satu bagian utama dalam sistem pembangkitan tenaga listrik. Generator berfungsi untuk merubah energi mekanik menjadi energi listrik [3].

2.1.1 Karakteristik Unit Pembangkit Termal

Setiap jenis pembangkit memiliki karakteristik yang berbeda-beda. Perbedaan karakteristik unit pembangkit ini menyebabkan prioritas pembangkit dalam mensuplai beban suatu sistem tenaga listrik menjadi berbeda. Secara umum, jenis prioritas pembangkit dalam sistem tenaga listrik terbagi menjadi tiga bagian yaitu

1. *Base load*

Pembangkit beban dasar (*base load*) mempunyai karakteristik yang tidak begitu fleksibel. Pembangkit jenis ini tidak dapat dihidupkan atau dimatikan dalam waktu yang singkat dan lambat dalam menaikkan atau menurunkan daya terbangkitkan Contoh dari pembangkit tipe *base load* adalah PLTU batubara.

2. Pembangkit beban menengah (*load follower*)

Pembangkit tipe *load follower* merupakan pembangkit yang memiliki karakteristik lebih fleksibel namun juga lebih mahal biaya pengoperasiannya apabila dibandingkan dengan pembangkit base load.

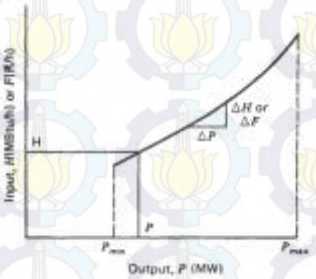
3. Pembangkit beban puncak (*peaker*)

Pembangkit beban puncak merupakan pembangkit yang memiliki karakteristik yang fleksibel sehingga responsif dalam hal kecepatan perubahan pembebanan dan pengaturan kondisi operasional dan non operasional dari pembangkit tersebut. Contoh dari pembangkit jenis ini adalah PLTG (Pembangkit Listrik Tenaga Gas) minyak, PLTD serta PLTA waduk

Pada pembangkit tenaga listrik tenaga panas atau *thermal* merupakan pembangkit listrik yang mayoritas digunakan untuk memenuhi beban harian atau *base load*. Dalam pengoprasiannya, pembangkit termal mempunyai dua karakteristik pembangkitan yaitu Karakteristik *input-output* dan Karakteristik *incremental rate.* Kedua karakteristik tersebut diperesntasikan pada gambar 2.2 dan gambar 2.4

Karakteristik Input Output Pembangkit Thermal

Karakteristik input-output pada pembangkit merupakan dasar penyusunan fungsi biaya. Biaya operasi dari suatu sistem pembangkit tenaga listrik merupakan biaya terbesar dalam pengoprasian suatu perusahaan pembangkit tenaga listrik. Biaya yang dikeluarkan oleh suatu perusahaan listrik untuk menghasilkan energi listrik dalam sistem ditentukan oleh biaya investasi dan biaya operasi pembangkit. Dalam karakteristik *input-output* pembangkit *thermal*, input adalah biaya pembangkitan dari hasil perkalian biaya ($) kalori yang terkandung dalam bahan bakar dengan kebutuhan kalori tiap jam generator (Btu/h). Sedangkan output pembangkit adalah daya yang dibangkitkan (P) dalam Mega Watt.



Gambar 2. 2 Karakteristik *input-output* unit pembangkit [4]

Berdasarkan gambar 2.2 diketahui bahwa Karakteristik input-output merupakan pendekatan atau linearisasi dari *input* berupa biaya bahan bakar yang masuk ke pembangkit (MBtu/h) atau total biaya yang dibutuhkan ($/h) terhadap daya *output* yang dibangkitkan tiap unit pembangkit (MW). Secara umum, karakteristik *input-output* pembangkit didekati dengan fungsi polinomial orde dua yaitu:

$$H(P_i) = c + bP_i + aP_i^2 \tag{2.1}$$

$$\mathrm{F}(P_i) = H(P_i) \times fuel\ cost \tag{2.2}$$

Dalam mendefinisikan karakteristik dari turbin uap digunakan beberapa istilah sebagai berikut :

$$\mathrm{H} = \frac{\mathrm{MBtu}}{\mathrm{jam}} \tag{2.3}$$

$$Fuel\ cost\ = \frac{\$}{\text{MBtu}} \tag{2.4}$$

$$\text{F}(P_i) \quad = \frac{\text{MBtu}}{\text{jam}} \times \frac{\$}{\text{MBtu}} \tag{2.5}$$

$$\text{F}(P_i) \quad = \frac{\$}{\text{jam}} \tag{2.6}$$

Dimana H merupakan energi panas yang dibutuhkan pada setiap jam nya. Sedangkan F adalah biaya yang dibutuhkan tiap jam. Dalam pengoprasiannya, dibutuhkan biaya karyawan dan biaya perbaikan. Untuk biaya karyawan dimasukkan dalam biaya operasional.

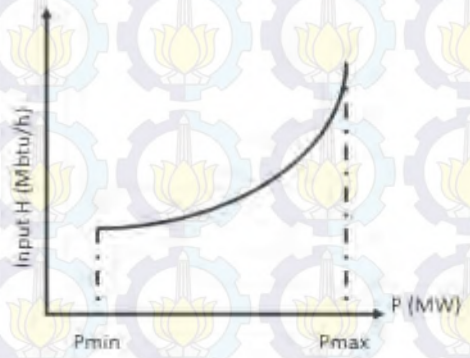
Output setiap unit generator mempunyai batas minimum dan batas maksimum pembangkit yang harus dipenuhi (*inequality contrain*) yaitu:

$$P_i \min \leq P_i \leq P_i \max \tag{2.7}$$

dengan

P_i *min*, P_i *max* adalah daya output minimum dan maksimum generator i.

Kondisi teknis pada boiler dan turbin menyebabkan harus diperhatikannya batas minimum dari daya output generator, kondisi teknis yang dimaksud adalah perbedaan suhu metal motor. Sedangkan batas maksimal dari daya output suatu pembangkit ditentukan dari desain kapasitas boiler dan turbin generator

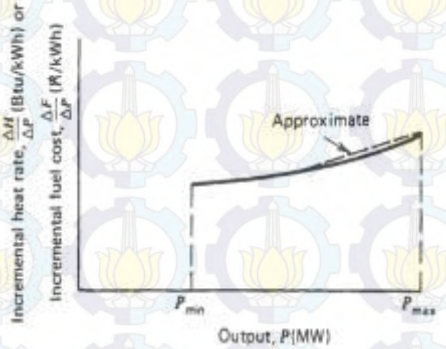


Gambar 2. 3 Kurva input-output pembangkit thermal[4]

Karakteristik *Incremental Heat*

Karakteristik *incremental heat rate* dipresentasikan pada gambar 2.4 yang dinyatakan dengan simbol $\Delta H/\Delta P$ atau $\Delta F/\Delta P$. Pada kurfa karakteristik tersebut data input dinyatakan dalam Btu/kWh or $/kWh, sedangkan pada data output dinyatakan dalam megawatt (MW). Karakteristik ini digunakan untuk semua pembangkitan yang ekonomis dari setiap unit pembangkit yang dikonversikan dari karakteristik increamental fuel cost dengan mengalikan incremental heat rate (Btu/kWh) terhadap besar fuel cost ($/Btu)[4].

Kurva Increamental *fuel-cost* adalah ukuran dari berapa biaya produksi yang dibutuhkan pada kenaikan daya berikutnya. Total biaya pembangkitan adalah biaya bahan bakar, biaya pekerja, cadangan, dan biaya perawatan [3].



Gambar 2. 4 Karakteristik *incremental rate*

2.2. Economic *Dispatch*

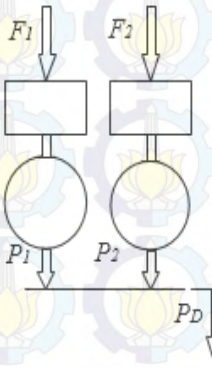
Biaya operasi terekonomis sangatlah penting pada sebuah sistem tenaga listrik untuk mengembalikan modal yang di investasikan. Maka, perlu untuk dilakukan analisis aliran daya optimal untuk meminimalkan biaya pembangkitan (*Economic Dispatch*) dan rugi-rusi minimum pentransmisian dari daya yang dibangkitkan ke beban.[1]. Pada kondisi tertentu, diperlukan penjadwalan yang terekonomis untuk menentukan daya keluaran dari setiap pembangkit. Sehingga dapat meminimalisir biaya pembangkitan yang diperlukan untuk men-*suplay* beban.

Dalam economic dispatch optimasi dilakukan dengan menentukan pembebanan pada unit-unit pembangkit untuk menyuplai kebutuhan beban dengan biaya yang teroptimum. Sehingga didapatkan biaya

pembangkitan yang minimum terhadap produksi daya listrik yang dibangkitkan. Namun tetap memperhatikan batas-batas daya yang dibangkitkan.

Dengan penerapan economic dispatch, maka pengoptimalan dilakukan dengan kalkulasi terhadap parameter-parameter, seperti jenis bahan bakar, jumlah kebutuhan bahan bakar (*fuel)*, ketersediaan bahan bakar dan lain sebagainya. Parameter-parameter inilah yang nantinya akan menentukan perancangan jangka panjang dari sebuah sistem, penentuan porsi biaya bahan bakar dan manajemen operasi pada pembangkit[4].

Pada tugas akhir ini, permasalahan economic dispatch dengan tidak memperhatikan rugi-rugi transmisi dan penjadwalan dari unit-unit pembangkit. Sehingga, permasalahan *economic dispatch* tidak mempertimbangkan konfigurasi sistem dan impedansi jaringan transmisi. Maka diasumsikan bahwa sistem hanya terdiri dari satu bus dengan semua pembangkit dan beban yang terhubung.



Gambar 2. 5 Pembangkit yang terhubung dengan satu bus [3]

Pada gambar 2.5 diperlihatkan sistem yang terdiri dari N unit pembangkit termal yang terhubung pada sebuah bus untuk menyuplai daya pada beban P_D. Input pada setiap unit adalah F_i, yang mempresentasikan fuel cost dari setiap unit [3]. Total biaya pembangkitan pada sistem adalah penjumlahan dari biaya pada setiap unit.

Karena rugi-rugi transmisi diabaikan, maka total biaya pembangkitan pada sistem adalah penjumlahan dari semua pembangkitan dengan fungsi biaya *Fi* pada setiap pembangkit. Sehingga, untuk

menemukan total biaya pembangkitan daya aktif dari setiap plan didefinisikan dengan persamaan berikut.

$$Ft = F_1 + F_2 + \cdots + F_{Ngen} \quad (2.8)$$

$$Ft = \sum_{i=1}^{n} Fi \quad (2.9)$$

$$Ft = \sum_{i=1}^{n} Fi\ (P_i) \quad (2.10)$$

$$= \sum_{i=1}^{n} a_i + b_i P_i + c_i {P_i}^2 \quad (2.11)$$

Persamaan tersebut menunjukkan bahwa input (bahan bakar) adalah fungsi obyektif yang akan dioptimasi. Beban sistem P_D dan karena rugi transmisi diabaikan maka jumlah output dari setiap pembangkit digunakan untuk melayani P_D menjadi [3]:

$$Pt = P_1 + P_2 + \cdots + P_{Ngen} \quad (2.12)$$

Equality constrain

$$\emptyset = 0 = P_D - \sum_{i=1}^{n} P_i \quad (2.13)$$

$$\emptyset = P_D - \sum_{i=1}^{n} P_i \quad (2.14)$$

Dimana,

Ft	: total biaya produksi
Fi	: biaya produksi dari setiap pembangkit
P_i	:daya aktif yang dibangkitkan,
P_D	:daya permintaan beban
n	: total jumlah unit pembangkit.

Permasalahan optimisasi dapat diselesaikan dengan menggunakan fungsi *Lagrange* (*Lagrange multiplier*). Fungsi lagrange diperlihatkan pada persamaan:

$$\mathcal{L} = F_t + \lambda\emptyset \tag{2.15}$$

Ketika minimalisasi tersebut dibatasi dengan $equality\ constrain$ dalam fungsi objektive dengan lagrang multiplier, maka akan menjadi persamaaan 2.16.

$$\mathcal{L} = \sum_{i=1}^{n} F_{t_i} + \lambda\left(P_D - \sum_{i=1}^{n} P_i\right) \tag{2.16}$$

$\lambda =$ faktor pengali Lagrange

Lalu fungsi diatas diturunkan terhadap *Pi* dan λ [3],

$$\frac{\partial \mathcal{L}}{\partial P_i} = 0 \tag{2.17}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = 0 \tag{2.18}$$

Sehingga didapatkan persamaan,

$$\frac{\partial F_t}{\partial P_i} + \lambda(0 - 1) = 0 \tag{2.19}$$

Dengan

$$F_t = F_1 + F_2 + \ldots + F_n \tag{2.20}$$

Maka didapatkan persamaan,

$$\frac{\partial F_t}{\partial P_i} = \frac{dF_i}{dP_i} = \lambda \tag{2.21}$$

$$\frac{dF_i}{dP_i} = \lambda \tag{2.22}$$

Persamaan ini menunjukkan bahwa kondisi optimum dapat dicapai bila *increamental fuel cost* (λ) pada setiap pembangkit adalah sama.

Dengan pendekatan tersebut, maka berlaku *constrain* pada persamaan 2.24 Dimana daya *output* dari pembangkit harus sesuai dengan total daya permintaan beban.

Selain itu daya terbangkit dari unit pembangkit harus memenuhi persamaann 2.23 sebagai *inequalities constrain*. Dimana daya *output* dari setiap pembangkit harus lebih besar dan harus kurang dari daya pembangkitan yang di ijinkan dari unit pembangkit[4].

Dari N buah pembangkit dalam sistem tenaga diatas dan beban sebesar P_D. maka untuk menyelesaikan permasalahan *economic dispatch* adalah:

$$\frac{dF_i}{dP_i} = \lambda \tag{2.22}$$

$$P_i \min \leq P_i \leq P_i \max \tag{2.23}$$

$$\sum_{i=1}^{N} P_i = P_D \tag{2.24}$$

Dimana i = indeks pembangkit ke i

Bilamana hasil P_i yang diperoleh keluar dari batasan $P_i min$ dan $P_i\ max$ nya, maka batasan ketidak samaan tersebut dapat diperluas menjadi:

$\frac{dF_i}{dP_i} = \lambda$ untuk $P_i \min \leq P_i \leq P_i \max$ (2.25)

$\frac{dF_i}{dP_i} \leq \lambda$ untuk $P_i > P_i \max$ set $P_i = P_i \max$ (2.26)

$\frac{dF_i}{dP_i} \geq \lambda$ untuk $P_i < P_i \min$ set $P_i = P_i \min$ (2.27)

-Halaman ini sengaja dikosongkan--

BAB III
IMPLEMENTASI *ALGORITMA ENHANCED LAMBDA ITERATION* PADA *ECONOMIC DISPATCH*

Pada bab ini akan membahas tentang implementasi algoritma *enhanced lambda iteration* yang dipergunakan untuk menyelesaikan permasalahan dalam *economic dispatch* pada sistem kelistrikan. Dimana pengolahan data dan simulasi dengan mengembangan *software* Powergen yang berbasis Delphi 7. Powergen adalah software yang digunakan pada mata kuliah operasi optimum sisitem tenaga listrik dalam melakukan perhitungan *power flow*, *economic dispatch*, *unit comitment*, optimasi sederhana dengan *linear programing*, penjadwalan pembangkit *hydro*, *unit comitment* dan *dynamic economic dispatch*.

3.1. Alur Pengerjaan

Pengerjaan Tugas Akhir ini dimulai dengan menentukan fungsi obyektif dan juga variabel *constraint.* Pada tahap awal dilakukan perhitungan manual dengan 6 unit pembangkit yang menggunakan data pada peper IEEE [2]. Hal ini diperlukan untuk memastikan alur kerja dari metode *algoritma enhanced lambda iteration* dan membandingkan hasil simulasi sebagai validasi apakah program yang dibentuk telah berhasil secara akurat atau tidak.

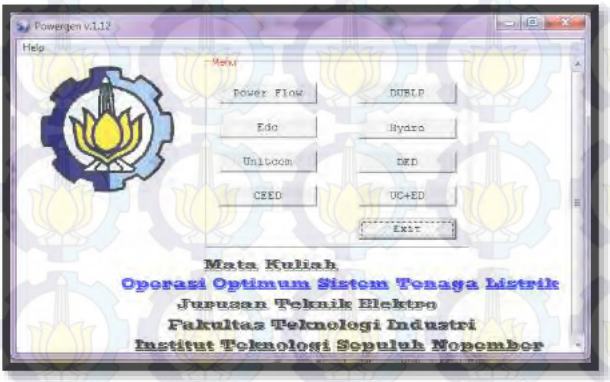
Apabila telah diperoleh hasil yang sesuai antara perhitungan manual dan hasil *running* program dengan 6 unit pembangkit, maka dilanjutkan pada tahap pengembangan program dengan unit yang lebih besar. Dengan melakukan pengujian sesuai dengan kasus pada peper. Pada tahap akhir pengerjaan, semua hasil *running* program *economic dispatch* dengan *algoritma enhanced lambda iteration* dibandingkan dengan *lambda iteration*.

Alur penyelesaian dari tugas akhir dari awal hingga akhir akan dipresentasikan pada gambar 3.1.

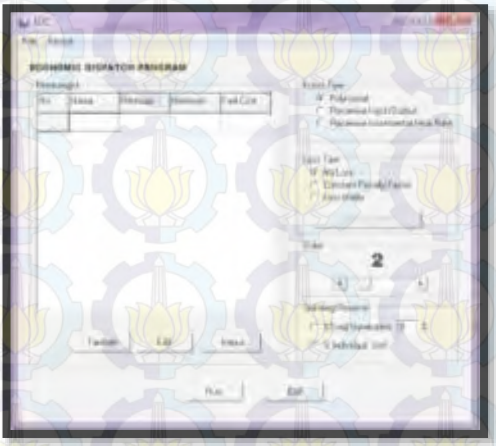
Gambar 3. 1 Flowchart Penyelesaian Tugas Akhir

3.2. Perencanaan pengembangan software perhitungan economic dispatch

Dalam melakukan perhitungan yang berkaitan dengan materi operasi optimasi sistem tenaga listrik dapat dilakukan dengan *software* powergen. Pada gambar 3.2 dapat dilihat tampilan menu utama dari *software* powergen.



Gambar 3. 2 Tampilan utama powergen



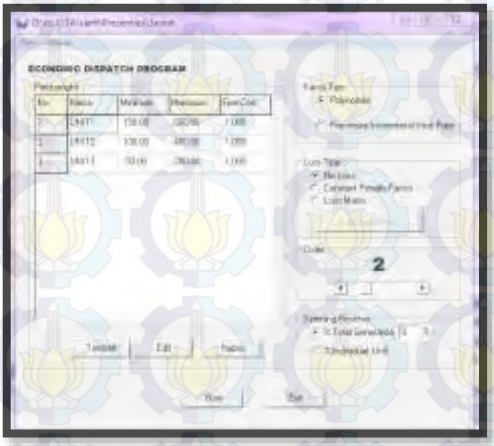
Gambar 3. 3 Tampilan utama menu EDC

Pada menu utama EDC data karakteristik pembangkit yang akan dilakukan perhitungan diinput dengan menekan tombol tambah.

Sedangkan untuk melakukan pengecekan kembali pada data yang telah di simpan dapat dilakukan dengan menekan tombol edit.

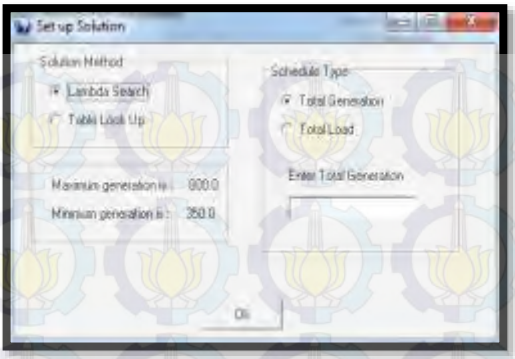
Gambar 3. 4 Menu input-an data pembangkit di EDC

Untuk memanggil data pembangkita yang telah tersimpan dapat dilakukan dengan memilih file-load-open-pilih data yang akan diptimasi. Maka data yang telah tersimpan tersebut akan tampil.



Gambar 3. 5 Menampilkan data pada menu EDC

Setelah data dipanggil, maka langkah selanjutnya adalah dengan memilih tombol *Run*. Ketika tombol *Run* dipilih, maka akan muncul tampilan seperti Gambar 3.7.



Gambar 3. 6 Tampilan *Set up Solution* menu EDC

Pada *set up solution* optimasi sistem dapat dilakukan dengan memilih metode optimasi yang akan dipergunakan. Pada saat ini metode optimasi yang tersedia adalah *lambda serch* dan *table look up*. Pengembangan pada sofware ini dilakukan dengan penambahan algoritma *enhanced lambda iteration* pada bagian *solution metode* sebagai salah satu pilihan metode optimasi sistem.

Selain itu pada tampilan ini dapat dilihat pembangkitan maksimum dan pembangkitan minimum yang dapat di lakukan oleh unit-unit pembangkit yang ada. Selanjutnya mengisikan *Total Generation* atau *Total Load* yang diinginkan sesuai *Schedule Type* yang dipilih. Ketika semua data telah diisikan kemudian memilih *Ok* untuk menampilkan hasil perhitungan optimasi dengan metode lambda search seperti Gambar 3.8.

Gambar 3. 7 Tampilan hasil perhitungan EDC

Selelah total pembangkitan di isi dan di pilih ok, maka akan muncul hasil optimasi dari program pada jendela *result*.

Dalam perancangan program pada economic dispatch dengan algoritma *enhanced lambda iteration* tanpa memperhatikan losses transmisi terdapat argumen dan sintak yang harus dipertimbangkan dalam pengerjaannya, yaitu

1. Argumen input *economic dispatch*

Menentukan argumen input yang difungsikan sebagai masukan awal dalam optimasi dengan menggunakan iterasi enhanced lambda. Dari semua variabel yang terdapat dalam program.. Argumen yang digunakan adalah sebagai berikut:

Coeff [i]	=	Sebagai inputan awal dari nilai koefisien A, B, C dalam persamaan $$Hi\big(Pi(t)\big) = ai + biPi(t) + ciPi(t)^2$$
Unitmax[i]	=	Sebagai inputan awal dari batas *maximum* pembangkitan unit (Pmax)
Unitmin[i]	=	Sebagai inputan awal dari batasan *minimum* pembangititan unit (Pmin)
Fuelcost[i]	=	Sebagai inputan awal nilai dari *fuelcost* yang digunakan untuk persamaan $$Fi(Pi(t)) = Hi(Pi(t)) \times fuelcost\ i$$

2. Sintaksis

Sintaksis program merupakan perintah yang digunakan untuk melakukan pemanggilan program dengan argument input yang telah kita tentukan. Sehingga untuk menjalankan ELI pada program Powergen yang berbasis Delphi ini dibutuhkan sintaksis program sebagai berikut:

Datadump	=	Digunakan sebagai memasukkan semua data awal perhitungan di setiap periode
Enhanced lambda iteration	=	Sebagai prosedur untuk mennjalankan ELI, dengan menentukan nilai awal lambda. Dengan tujuan mendapatkan nilai pembangkitan yang sesui kebutuhan beban. $$\lambda^{[1]} = \frac{P_D + \sum \frac{b_i}{2C_i}}{\sum \frac{1}{2C_i}}$$

Data input	=	Digunakan untuk menerima *input* dari data *file* yang telah tersimpan di awal.
Data_ou3tput	=	Merupakan prosedur untuk memproses dan menampilkan hasil akhir dari seluruh periode pembebanan
Output_Routine	=	Merupakan prosedur untuk memperoleh hasil akhir tiap periode
Output_final	=	Merupakan prosedur untuk memproses dan menampilkan hasil akhir dari seluruh periode pembebanan

3.3. Algoritma Enhanced Lambda Iteration pada Economic Dispatch

Algoritma *enhanced lambda iteration* adalah salah satu metode yang dikembangkan untuk mengatasi permasalahan *economic dispatch* (ED) pada pembangkit termal. Dengan pengaplikasiannya pada *economic dispatch,* maka dapat mempercepat waktu yang diperlukan untuk konvergen .

3.3.1 Persamaan Economic Dispatch

Economic dispatch dengan pembangkit termal mempunyai fungsi obyektif yang direpresentasikan sebagai fungsi kuadrat. Fungsi biaya pembangkitan pada delphi sebagai salah satu argument input dalam bahasa Delphi di presentasikan oleh persamaan 3.1.

$$F_i(P_i) = a_i + b_i P_i + c_i {P_i}^2 \quad (3.1)$$

Pengoprasian ekonomis pembangkit tenaga listrik harus memenuhi batasan-batasan atau *contraints* tertentu. Dua *contraints* yang digunakan dalam tugas akhir ini adalah *equality constrains* dan *inequality constrains*.

a. *Equality constrain*

Equality constrain merupakan batasan keseimbangan daya, yang mengharuskan total daya yang dibangkitkan oleh masing-masing pembangkit harus sama dengan jumlah total kebutuhan beban dan rugi-rugi transmisi yang dapat dinyatakan dengan persamaan berikut:

$$\sum_{i=1}^{N} P_i - (P_D + P_L) = 0 \quad (3.2)$$

Dimana

P_D = Total daya permintaan beban

P_L = Total rugi-rugi transmisi

Namun pada tugas akhir ini tidak memperhatikan *losses* dari sistem ($P_L = 0$). Sehingga menjadi persamaan

$$P_i = P_D \tag{3.3}$$

b. Generator limit

Daya yang dibangkitkan unit pembangkit harus sesuai dengan batasan masing-masing unit pembangkit.

$$P_i^{max} \leq P_i \leq P_i^{min} \tag{3.4}$$

Dimana P_i^{max} dan P_i^{min} adalah minimum dan maksimum daya keluaran dari setia i unit pembangkit.

Dalam memulai program dengan *enhanced lambda,* tahap awal yang harus dilakukan setelah selesai memasukkan fungsi obyektif dan *load demand* adalah *dengan* menentukan nilai awal *increamental cost* (lambda) dihitung berdasarkan konsep dari penjumlahan kriteria increamental cost dan pembangkitan daya aktif. Persamaan untuk mendapatkan lambda awal terdapat pada persamaan berikut,

$$\lambda = \frac{P_D + \sum \frac{b_i}{2C_i}}{\sum \frac{1}{2C_i}} \tag{3.5}$$

Berdasarkan nilai lambda tersebut dapat diketahui nilai pembangkitan untuk setiap unit pembangkit dengan persamaan 3.6 dari hasil penurunan persamaan 3.1.

$$P_i^{[1]} = \frac{\lambda^{[1]} - b_i}{2a_i} \tag{3.6}$$

Untuk beroperasi secara ekonomis, maka seluruh unit pembangkit harus beroperasi pada nilai *incremental cost* yang sama dan memenuhi batasan *inequality constrain*. Jika batasan *minimum* memiliki nilai seperti yang didapatkan pada persamaan (3.7) maka akan didapatkan solusi (3.8).

$$Pi \leq Pi\ min \tag{3.7}$$

$$Pi = Pi\ min \tag{3.8}$$

Jika batasan *maximum* memiliki nilai seperti yang didapatkan pada persamaan (3.9) maka akan didapatkan solusi (3.10).

$$Pi \geq Pi\ max \tag{3.9}$$

$$Pi = Pi\ max \tag{3.10}$$

Nilai total daya pembangkitan ($\sum P_i$) pada awal iterasi tidak selalu sesuai dengan total daya beban (P_D). Untuk menghitung besar ketidak sesuaian tersebut diketahui dengan menggunakan persamaan berikut:

$$\Delta P = P_D - \sum P_i \tag{3.11}$$

Jika nilai ketidak sesuaian $\Delta P^{[1]}$ tidak dalam batas toleransi yang diizikan seperti pada persamaan 3.12. Maka, dilakukan iterasi kedua dengan persamaan 3.13.

$$-0.0002 < \Delta P < 0.0002 \tag{3.12}$$

$$\lambda^{[2]} = \pm 10\ \%\ dari\ \lambda^{[1]} \tag{3.13}$$

Jika nilai ketidak sesuaian $\Delta P^{[1]}$ seperti yang didapatkan pada persamaan (3.14) maka akan didapatkan solusi (3.15).

$$\Delta P \leq 0 \tag{3.14}$$

$$\lambda^{[2]} = \lambda^{[1]} * 90\% \tag{3.15}$$

Jika nilai ketidak sesuaian $\Delta P^{[1]}$ seperti yang didapatkan pada persamaan (3.16) maka akan didapatkan solusi (3.17).

$$\Delta P \geq 0 \tag{3.16}$$

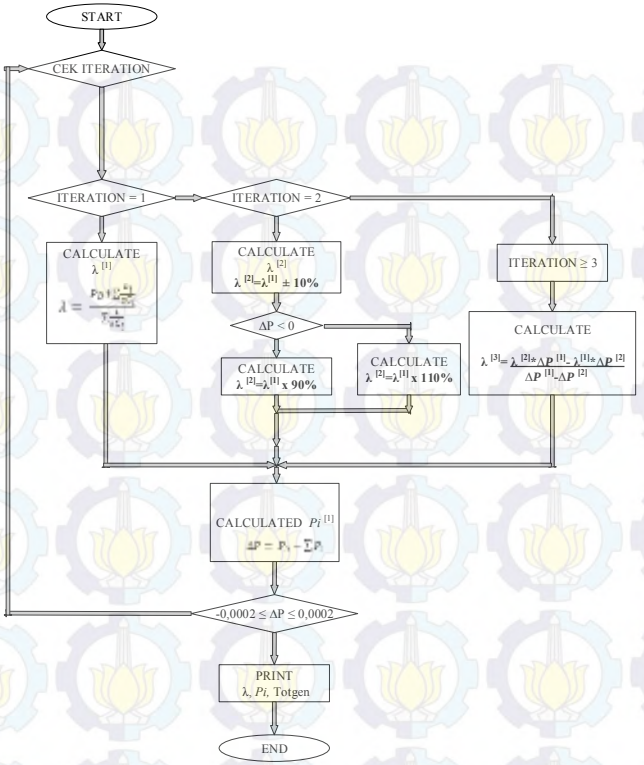
$$\lambda^{[2]} = \lambda^{[1]} * 110\% \tag{3.17}$$

Cek pembangkitan dari setiap unit pembankit hasil perubahan nilai $\lambda^{[2]}$. Jika nilai total daya pembangkitan $(\sum P_i)$ pada iterasi kedua sesuai dengan toleransi persamaan 3.12, maka pencarian nilai lambda telah selesai pada iterasi ke dua.

Apabila kondisi persamaan 3.12 belum tercapai. Maka, dilakukan iterasi ketiga dan seterusnya untuk mencari nilai $\lambda^{[3]}$ dengan persamaan 3.18.

$$\lambda^{[3]} = \frac{\lambda^{[2]} * \Delta P^{[1]} - \lambda^{[1]} * \Delta P^{[2]}}{\Delta P^{[1]} - \Delta P^{[2]}} \tag{3.18}$$

Alur perhitungan ED dengan menggunakan *enhanced lambda iteration* yang akan diimplementasikan pada tugas akhir ini di presentasikan pada gambar 3.8.



Gambar 3. 8 Flowchart Penyelesaian ELI

3.4. Uji Kasus

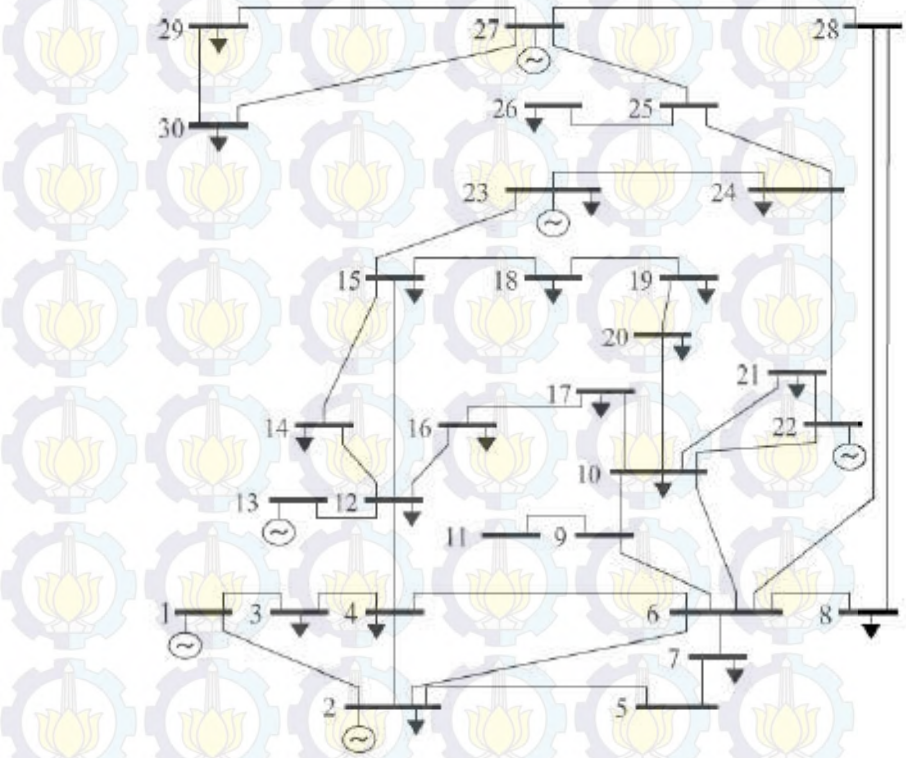
Pada sub bab ini akan dilakukan perhitungan manual pada salah satu kasus uji. Pada kasus 1 akan dilakukan perhitungan nilai lambda (λ), daya pembangkitan tiap unit pembangkit dan total cost dengan menggunakan metode *Enhanced lambda iteration* dengan total 6 unit pembangkit. Kasus 2 akan dilakukan perhitungan untuk 15 unit pembangkit. sedangkan pada unit 3 dilakukan perhitungan dengan skala yang lebih besar yaitu dengan 40 unit pembangkit. Pengujian pada kasus 1 akan dijelaskan pada sub bab 3.4.1

3.4.1 Tes Sistem 1

Data fungsi biaya untuk kasus 1 dapat dilihat pada tabel 3.1.

Tabel 3. 1 Data kasus 1

Generator	P_i^{max} (MW)	P_i^{max} (MW)	*Cost Function* (R/MWh)
1	500	100	$240 + 7\,P_1 + 0.007P_1^2$
2	200	50	$200 + 10\,P_2 + 0.0095P_2^2$
3	300	80	$220 + 8.5\,P_3 + 0.009P_3^2$
4	150	50	$200 + 11\,P_4 + 0.009P_4^2$
5	200	50	$220 + 10.5P_5 + 0.008\,P_5^2$
6	120	50	$190 + 12\,P_6 + 0.0075P_6^2$



Gambar 3.9 Single line diagram 6 unit pembangit tes sistem 1

Berikut ini adalah perhitungan manual dari kasus 1 pada jam 18.00 dengan beban 1263MW.

- Beban 1263 MW

Fungsi biaya pada tabel 3.1 merupakan hasil dari perkalian anatara fungsi IHR dengan *fuel cost*. Berikut perhitungan manual untuk 6 unit pembangkit pada kasus 1 dengan perminataan daya beban 1263 MW.

$$F_1 = 240 + 7P_1 + 0.007P_1^2 \quad \rightarrow \quad \frac{dF_1}{dP_1} = 7{,}0 + 0.014P_1$$

$$F_2 = 200 + 10P_2 + 0.0095P_2^2 \quad \rightarrow \quad \frac{dF_2}{dP_2} = 10 + 0.019P_2$$

$$F_3 = 220 + 8.5P_3 + 0.009P_3^2 \quad \rightarrow \quad \frac{dF_3}{dP_3} = 8.5 + 0.018P_3$$

$$F_4 = 200 + 11P_4 + 0.009P_4^2 \quad \rightarrow \quad \frac{dF_4}{dP_4} = 11 + 0.018P_4$$

$$F_1 = 220 + 10.5P_5 + 0.008P_5^2 \quad \rightarrow \quad \frac{dF_5}{dP_5} = 10.5 + 0.016P_5$$

$$F_1 = 190 + 12P_6 + 0.0075\,P_6^2 \quad \rightarrow \quad \frac{dF_1}{dP_1} = 12 + 0.015P_6$$

$$\lambda = \frac{1263 + \frac{7}{0.014} + \frac{10}{0.019} + \frac{8.5}{0.018} + \frac{11}{0.018} + \frac{10.5}{0.016} + \frac{12}{0.015}}{\frac{1}{0.014} + \frac{1}{0.019} + \frac{1}{0.018} + \frac{1}{0.018} + \frac{1}{0.016} + \frac{1}{0.015}}$$

$$\lambda = 13.2539 \ \frac{\$}{\text{MWH}}$$

$$P_i = \frac{\lambda - b_i}{2C_i}$$

Tabel 3. 2 Hasil perhitungan kasus 1 dengan beban 1263MW

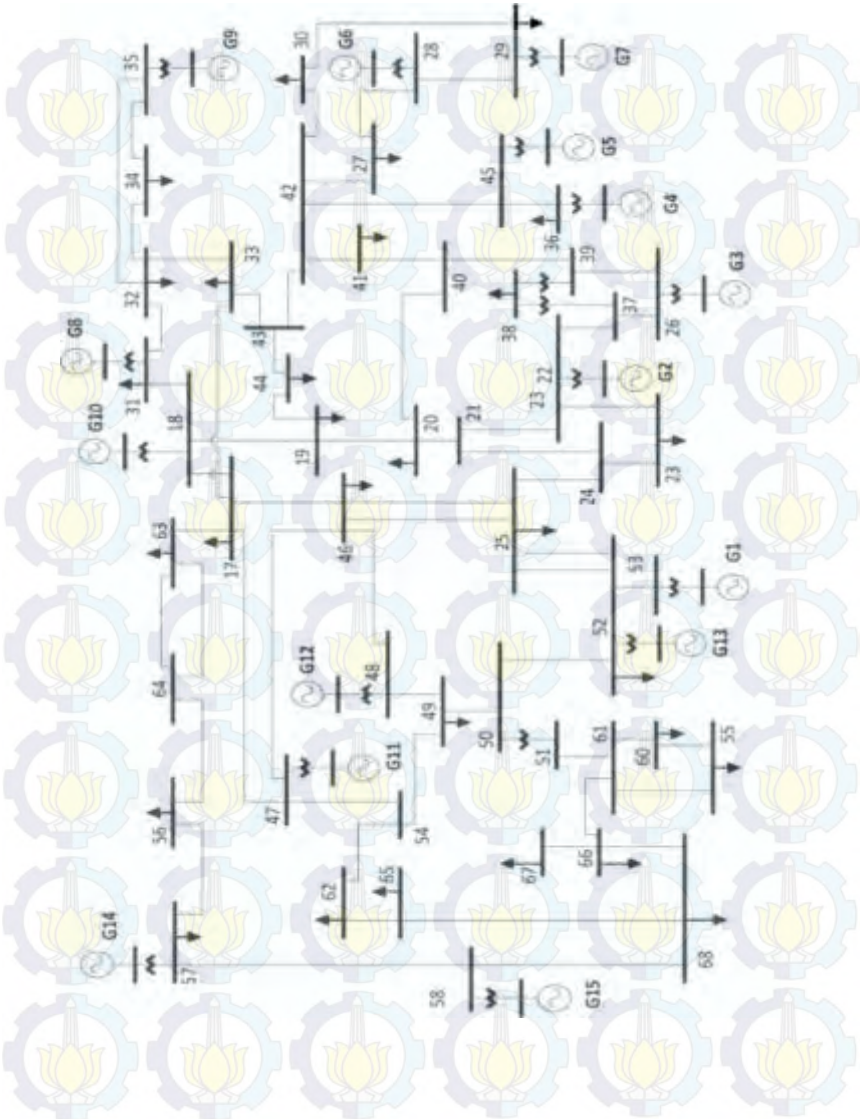
Iterasi	$P_1(MW)$	$P_2(MW)$	$P_3(MW)$	$P_4\ (MW)$	$P_5\ (MW)$	$P_6(MW)$
1	446.707	171.258	264.1057	125.216	172.1189	83.59
Total	1263					

Tabel 3. 3 Hasil akhir perhitungan biaya pembangkitan kasus 1

3.4.2 Tes Sistem 2

Unit	1	2	3	4	5	6	Ptot
P_i (MW)	446.707	171.258	264.1057	125.216	172.1189	83.59	1263
$F_{cost}\ (\$/h)$	4763.78	2191.21	3092.66	1718.50	2264.25	1245.53	15275.93

Pada kasus 2 ini, perhitungan manual economic dispatch dengan ELI pada 15 unit pembangkit dengan permintaan daya beban 2650 MW terlampir.



Gambar 3.10 Single line diagram 15 unit pembangit tes sistem 2

Tabel 3. 4 Data kasus 2

Unit	P_i^{max} (MW)	P_i^{max} (MW)	a_i	b_i	c_i
1	455	150	671.03	10.1	0.000299
2	455	150	574.54	10.22	0.000183
3	130	20	374.59	8.8	0.001126
4	130	20	374.59	8.8	0.001126
5	470	150	461.37	10.4	0.000205
6	460	135	630.14	10.1	0.000301
7	465	135	548.2	9.87	0.000364
8	300	60	227.09	11.5	0.000338
9	162	25	173.72	11.21	0.000807
10	160	20	175.95	10.72	0.001203
11	80	20	186.8	11.21	0.003586
12	80	20	230.27	9.9	0.005513
13	85	25	230.27	13.12	0.000371
14	55	15	309.03	12.12	0.001929
15	55	15	323.79	12.41	0.004447

Tabel 3. 5 Increamental cost function kasus 2

Unit	Fungsi *increamental cost* dF_i/dP_i	$\frac{1}{2C}$	$\frac{b}{2C}$
1	$10.1 + 0{,}000598P_1$	1672,2408	16889,632
2	$10.22 + 0{,}000366P_2$	2732,24044	27923,497
3	$8.8 + 0{,}002252\ P_3$	444,049734	3907,6377
4	$8.8 + 0{,}002252P_4$	444,049734	3907,6377
5	$10.4 + 0{,}00041\ P_5$	1639,34426	25365,854
6	$10.1 + 0{,}000602\ P_6$	1246,88279	16777,409
7	$9.87 + 0{,}000728P_7$	1373,62637	13557,692
8	$11.5 + 0{,}000676\ P_8$	1479,28994	17011,834
9	$11.21 + 0{,}001614\ P_9$	619,578686	6945,4771
10	$10.72 + 0{,}002406\ P_{10}$	415,627598	4455,5278
11	$11.21 + 0{,}007172\ P_{11}$	0,04460303	1563,0229
12	$9.9 + 0{,}011026\ P_{12}$	139,431121	1380,3681
13	$9.9 + 0{,}011026P_{13}$	90,6947216	1189,9147
14	$12.12 + 0{,}003858P_{14}$	259,201659	3141,5241
15	$12.41 + 0{,}008894\ P_{15}$	112,43535	1395,3227
Total		12668,7378	133827,53

3.4.3 Tes Sistem 3

Data pembangkit untuk tes sistem 3 dapat dilihat pada tabel **3.6.**

Tabel 3. 6 Data kasus 3

Unit	Pmin(MW)	Pmax(MW)	A	b	C
1	36	114	0,0069	6,73	94,705
2	36	114	0,0069	6,73	94,705
3	60	120	0,02028	7,07	309,54
4	80	190	0,00942	8,18	369,04
5	47	97	0,0114	5,35	148,89
6	68	140	0,01142	8,05	222,33
7	110	300	0,00357	8,03	287,71
8	135	300	0,00492	6,99	391,98
9	135	300	0,00573	6,6	455,76
10	130	300	0,00605	12,9	722,82
11	94	375	0,00515	12,9	635,2
12	94	375	0,00569	12,8	6544,69
13	125	500	0,00421	12,5	913,4
14	125	500	0,00752	8,84	1760,4
15	125	500	0,00708	9,15	1728,3
16	125	500	0,00708	9,15	1728,3
17	220	500	0,00313	7,97	647,85
18	220	500	0,00313	7,95	649,69
19	242	550	0,00313	7,97	647,83
20	242	550	0,00313	7,97	647,81
21	254	550	0,00298	6,63	785,96
22	254	550	0,00298	6,63	785,96
23	254	550	0,00284	6,66	794,53
24	254	550	0,00284	6,66	794,53
25	254	550	0,00277	7,1	801,32
26	254	550	0,00277	7,1	801,32
27	10	150	0,52124	3,33	1055,1
28	10	150	0,52124	3,33	1055,1
29	10	150	0,52124	3,33	1055,1
30	47	97	0,0114	5,35	148,89
31	60	190	0,0016	6,43	222,92
32	60	190	0,0016	6,43	222,92
33	60	190	0,0016	6,43	222,92
34	90	200	0,0001	8,95	107,87
35	90	200	0,0001	8,62	116,58
36	90	200	0,0001	8,62	116,58
37	25	110	0,0161	5,88	307,45
38	25	110	0,0161	5,88	307,45
39	25	110	0,0161	5,88	307,45
40	242	550	0,00313	7,97	647,83

BAB IV
ANALISA DAN SIMULASI

Pada bab ini membahas tentang analisa hasil implementasi perhitungan *economic dispatch* dengan menggunakan *enhanced lambda iteration*. Simulasi dilakukan pada 3 tes sistem dengan data pembangkit IEEE. Dimana masing-masing tes sistem memiliki karakteristik yang berbeda, yaitu:

4.1 Hasil Pengujian Tes Sistem 1

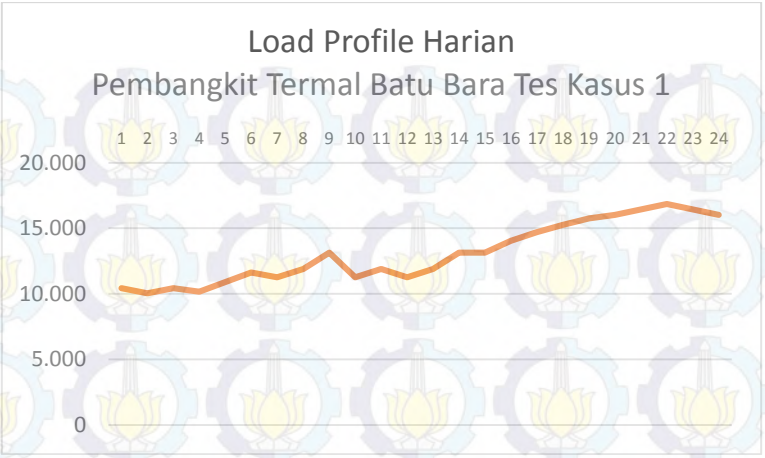
Pada pengujian awal tes sistem pertama, menggunakan 6 unit pembangkit termal dengan beban 1263 MW. Dengan karakteristik sistem diperlihatkan pada tabel 4.1[2]. Pada tes sistem 1 ini tidak ada *contrains* yang dilanggar dan dapat diselesaikan pada iterasi pertama. Data pembangkit yang dibutuhkan pada tes sistem 1 terdapat pada tabel 4.1.

Tabel 4. 1 Kapasitas pembangkitan dan fungsi biaya tes sistem 1

Unit	P_i^{max} (MW)	P_i^{max} (MW)	Fuel cost ($\frac{\$}{Mbtu}$)	*Cost Function* (R/MWh)		
1	500	100	1,0	240 +	7 P_1 +	0.007P_1^2
2	200	50	1,0	200 +	10 P_2 +	0.0095P_2^2
3	300	80	1,0	220 +	8.5 P_3 +	0.009P_3^2
4	150	50	1,0	200 +	11 P_4 +	0.009P_4^2
5	200	50	1,0	220 +	10.5P_5 +	0.008 P_5^2
6	120	50	1,0	190 +	12 P_6 +	0.0075P_6^2

Tabel 4.2 Load Profile Harian kasus 1

Jam	Total beban	Fuelcost	Jam	Total beban	Fuelcost
1	882	10.437,3835	13	1000	11.887,0166
2	850	10.052,3253	14	1100	13.152,0064
3	882	10.437,3835	15	1100	13.152,0064
4	860	10.172,2865	16	1170	14.055,1870
5	920	10.899,1082	17	1220	14.708,5501
6	980	11.638,0238	18	1263	15.275,9304
7	950	11.267,0543	19	1300	15.768,2035
8	1000	11.887,0166	20	1320	16.035,8616
9	1100	13.152,0064	21	1350	16.439,4072
10	950	11.267,0543	22	1380	16.845,4230
11	1000	11.887,0166	23	1350	16.439,4072
12	950	11.267,0543	24	1320	16.035,8616



Gambar 4.1 Load Profile Harian kasus 1

Data pembangkit pada tabel diatas di input-kan pada menu EDC di jendela data pembangkit. seperti pada tabel 3.5. Sehingga tampilan data hasil pengisian data-data pembangkit seperti pada Gambar 4.2.

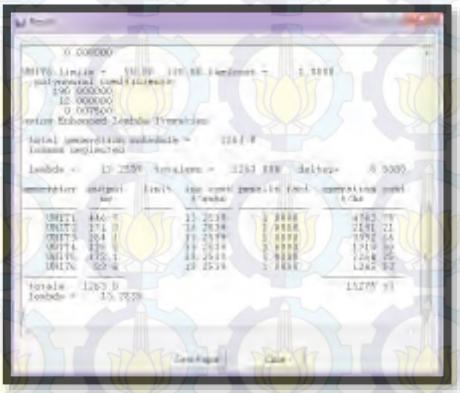
Gambar 4. 2 Tampilan data pada tes sistem1 di menu EDC

Setelah dipilih tombol *run*, maka akan muncul jendela *set up solution* pada gambar 4.2. Pada tampilan *set up solution* terdapat range pembangkitan dari tes sistem 1 yaitu pembangkitan maksimal adalah 1470 MW dan pembangkitan minimal adalah 380 MW. Selain itu juga

terdapat algoritma *enhanced lambda iteration* pada pilihan *solution metode* sebagai hasil dari pengembangan software seperti yang dijelaskan pada bab 3. Setelah dipilih *enhanced lamda iteration* dan telah ditentukan jenis *schedule type*, maka input nilai pembebanan 1263 MW [2].

Gambar 4. 3 Pengujian kasus 1 tampilan *set up solution*

Lalu pilih *ok* untuk memulai optimasi dengan algoritma *enhanced lambda iteration*. Setelah dijalankan maka hasil dari optimasi akan muncul seperti pada gambar 4.4.



Gambar 4. 4 Hasil optimasi tes sistem 1

Untuk menguji validasi dan kehandalan dari *software* yang di kembangkan dengan algoritma *enhanced lambda iteration* pada tes sistem 1 ini, maka hasil optimasi pada gambar 4.4 dibandingkan dengan beberapa pembanding yang lain meliputi:

4.1.1 Perbandingan Hasil dengan Perhitungan Manual

Pada sub bab ini akan menguji validasi dari software yang dikembangkan dengan cara memperbandingkan hasil *running* dengan hasil perhitungan manual. Berdasarkan hasil pebandingan dari keduanya, hasil running pada software sama dengan hasil pada perhitungan manual. Oleh karena itu dapat disimpulkan hasil *running* powergen sudah tepat. Hasil dari *running* powergen dan perhitungan manual terdapat pada tabel 4.3.

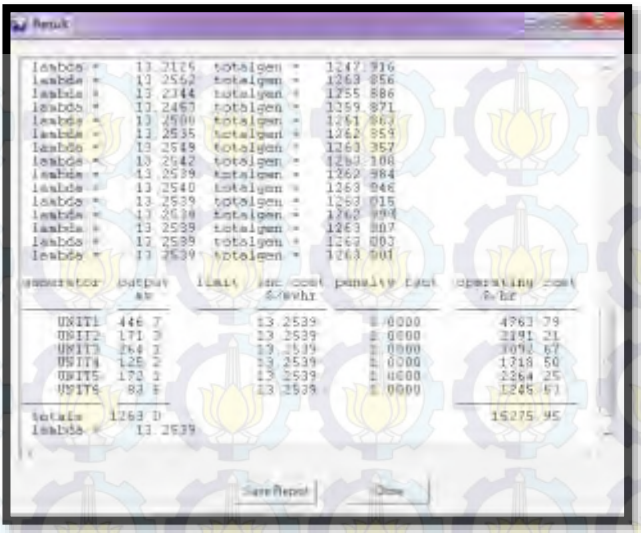
Tabel 4. 3 Perbandingan ELI dengan perhitungan manual kasus 1

Manual			Software		
lambda	Pi(MW)	fcost($/h)	lambda	Pi(MW)	fcost($/h)
13,2539	446,707272	4763,782607	13,2539	446,7	4763,78
	171,25799	2191,208237		171,3	2191,21
	264,105656	3092,66425		264,1	3092,66
	125,216767	1718,497583		125,2	1718,5
	172,118863	2264,247281		172,1	2264,25
	83,5934535	1245,530433		83,6	1245,53
Total	1263	15275,93		1263	15275,9

4.1.2 Perbandingan Hasil software dengan Algoritma E*nhanced Lambda Iteration* dan Software dengan *Lambda Iteration*

Tujuan memperbandingkan Hasil *software* dengan algoritma *enhanced lambda* iteration dan software dengan *lambda iteration* pada sub bab ini adalah untuk melihat kehandalan dari implementasi hasil perancangan software dengan algoritma *enhanced lambda iteration*. Hasil running *software* dengan algoritma *enhanced lambda iteration* dapat dilihat pada Gambar 4.3. Sedangakan hasil running *software* dengan *lambda* dapat dilihat pada gambar 4.4.

Berdasarkan perbandingan hasil runing keduanya terlihat *software* pada tes sistem 1 dengan algoritma *enhanced lambda iteration* cukup sekali iterasi untuk mendapatkan nilai *increamental fuel cost* yang *convergen*. Sedangkan pada lambda iterasi biasa membutuhkan 21 iterasi untuk mendapatkan nilai *increamental fuel cost* yang *convergen.* Perbandingan hasil dari keduanya terdapat dalam tabel 4.3.



Gambar 4. 5 Hasil *running software* dengan *lambda*

Tabel 4. 4 Perbandingan hasil *software* ELI dengan *lambda*.kasus 1

Lambda Iteration				Enhanced Lambda Iteration			
Iterasi	Final lambda	Pi(MW)	fcost($/h)	Iterasi	Final lambda	Pi(MW)	fcost($/h)
21	13,2539	446,7	4763,79	1	13,2539	446,7	4763,78
		171,3	2191,21			171,3	2191,21
		264,1	3092,67			264,1	3092,66
		125,2	1718,5			125,2	1718,5
		172,1	2264,25			172,1	2264,25
		83,6	1245,53			83,6	1245,53
Total		1263,0	15275,95	Total		1263	15275,93

4.1.3 Perbandingan Hasil *running* kasus 1 dengan Algoritma *Enhanced Lambda Iteration* dan *Quadratic Programing.*

Pada sub bab ini memperbandingkan hasil *running* powergen dengan algoritma e*nhanced lambda iteration* terhadap q*uadratic programing*. Q*uadratic programing* adalah program yang telah umum digunakan untuk perhitungan *economic dispatch* berbasis matlab. Pembangdingan keduanya dengan tetap mempergunakan data pada tes

sistem 1 [2]. Yang dibandingkan adalah biaya total pembangkitan dengan keduanya pada permintaan beban yang sama.

Tabel 4. 5 Perbandingan ELI dengan *Quadratic Programming* kasus 1

	Quadratic Programming	Enhanced Lambda
Total cost ($/hr)	15.275,95	15275,93

4.2 Hasil Pengujian Tes Sistem 2

Pada pengujian tes sistem 2, menggunakan 15 unit pembangkit termal dengan beban 2650 MW. Dengan karakteristik sistem diperlihatkan pada tabel 4.5 [2]. Berbeda dengan tes sistem 1, pada tes sistem 2 ini limit pembangkitan pada tiap pembangkit diperhatikan. Pada iterasi ke 1 dengan *increamental cost* 10,7694 sebagian besar daya yang dibangkitkan melanggar *constrains*. Maka akan dilakukan iterasi berikutnya untuk mendapatkan nilai yang sesuai. Hasil perhitungan dari perubahan nilai *increamental cost*, pembangkitan dan biaya pembangkitan terdapat dalam lampiran.

Tabel 4. 6 Kapasitas pembangkitan dan fungsi biaya tes sistem 2

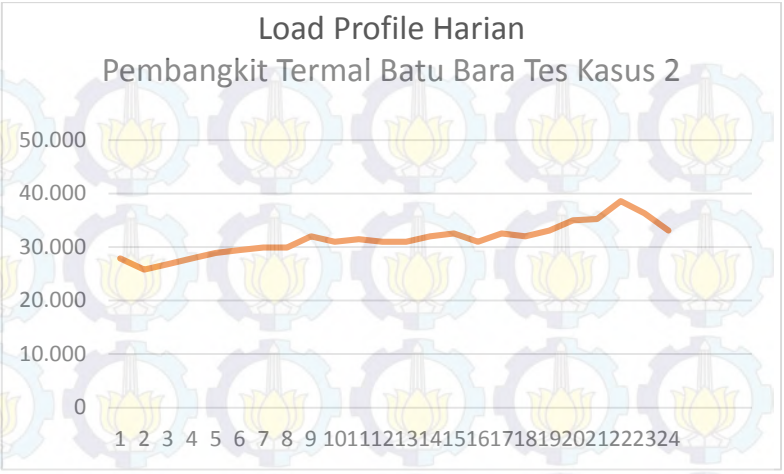
Unit	P_i^{max} (MW)	P_i^{min} (MW)	a_i	b_i	c_i
1	455	150	671.03	10.1	0.000299
2	455	150	574.54	10.22	0.000183
3	130	20	374.59	8.8	0.001126
4	130	20	374.59	8.8	0.001126
5	470	150	461.37	10.4	0.000205
6	460	135	630.14	10.1	0.000301
7	465	135	548.2	9.87	0.000364
8	300	60	227.09	11.5	0.000338
9	162	25	173.72	11.21	0.000807
10	160	20	175.95	10.72	0.001203
11	80	20	186.8	11.21	0.003586
12	80	20	230.27	9.9	0.005513
13	85	25	230.27	13.12	0.000371
14	55	15	309.03	12.12	0.001929
15	55	15	323.79	12.41	0.004447

Tabel 4. 7 Increamental cost function tes sistem 2

Unit	Fungsi *increamental cost* dF_i/dP_i	$\frac{1}{2C}$	$\frac{b}{2C}$
1	10.1 + 0,000598P_1	1672,2408	16889,632
2	10.22 + 0,000366P_2	2732,24044	27923,497
3	8.8 + 0,002252 P_3	444,049734	3907,6377
4	8.8 + 0,002252P_4	444,049734	3907,6377
5	10.4 + 0,00041 P_5	1639,34426	25365,854
6	10.1 + 0,000602 P_6	1246,88279	16777,409
7	9.87 + 0,000728P_7	1373,62637	13557,692
8	11.5 + 0,000676 P_8	1479,28994	17011,834
9	11.21 + 0,001614 P_9	619,578686	6945,4771
10	10.72 + 0,002406 P_{10}	415,627598	4455,5278
11	11.21 + 0,007172 P_{11}	0,04460303	1563,0229
12	9.9 + 0,011026 P_{12}	139,431121	1380,3681
13	9.9 + 0,011026P_{13}	90,6947216	1189,9147
14	12.12 + 0,003858P_{14}	259,201659	3141,5241
15	12.41 + 0,008894 P_{15}	112,43535	1395,3227
Total		12668,7378	133827,53

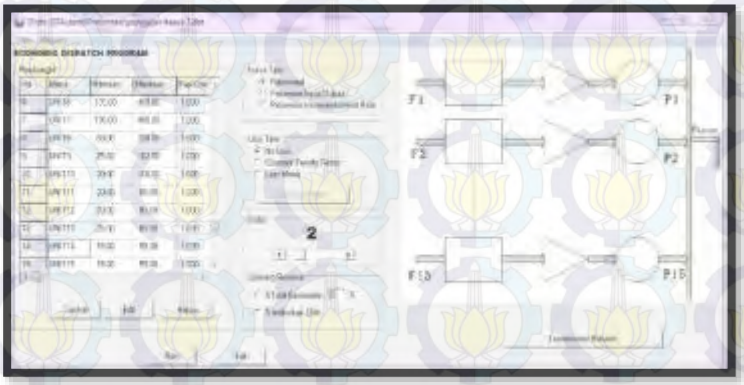
Tabel 4. 8 Tabel *load profile* harian kasus 2

Jam	Total Beban	Fuel cost	Jam	Total Beban	Fuel cost
1	2200	27869,8744	13	2500	30980,5145
2	2000	25806,0382	14	2600	32029,4723
3	2100	26837,1443	15	2650	32555,5476
4	2200	27869,8744	16	2500	30980,4479
5	2300	28904,2286	17	2650	32555,5476
6	2350	29422,2786	18	2600	32029,5261
7	2400	29940,2069	19	2700	33082,5573
8	2400	29940,2069	20	2880	35001,8590
9	2600	32029,4723	21	2900	35220,3492
10	2500	30980,5145	22	3200	38623,6645
11	2550	31504,4929	23	3000	36.332
12	2500	30980,5145	24	2700	33082,5573



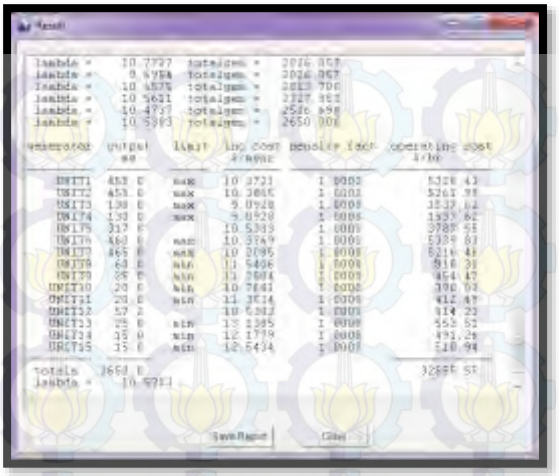
Gambar 4.6 *load profile* harian kasus 2

Data pembangkit pada tabel di input-kan pada menu EDC di jendela data pembangkit. seperti pada tabel 4.5. Sehingga tampilan data hasil pengisian data-data pembangkit seperti pada Gambar 4.7.



Gambar 4. 7 Tampilan data pada tes sistem2 di menu EDC

Setelah data pada tes sistem 2 diinputkan dan di running maka didapatkan nilai *increamental cost* yang convergen 10.5303 dan total fuel cost untuk membangkitakan daya 2650 adalah 32555.55 $/MWh.



Gambar 4. 8 Hasil optimasi tes sistem 2

Dari hasil *running software* powergen pada gambar 4.6 diketahui jika ingin membangkitakan daya 2650, maka dibutuhkan 6 iterasi untuk *convergen*. variasi nilai lambda dengan iterasinya terdapat pada tabel 4.7.

Tabel 4. 9 Variasi nilai lambda dengan iterasi

Iterasi	Lambda	Totalgen
1	107.727	2.826.057
2	96.954	2.826.057
3	106.575	2.813.700
4	105.611	2.727.853
5	104.737	2.506.698
6	105.303	2.650.000

4.2.1 Perbandingan Hasil dengan Perhitungan Manual

Pada sub bab ini akan menguji validasi dari software yang dikembangkan dengan cara memperbandingkan hasil *running* dengan hasil perhitungan manual. Hasil perhitungan manual yang dilakukan pada tes sistem 2 terlampir. Berdasarkan hasil pebandingan dari keduanya, hasil *running* pada s*oftware* identik dengan hasil pada perhitungan manual. Oleh karena itu dapat disimpulkan hasil pengembangan powergen dengan algoritma e*nhanced lambda iteration* sudah tepat. Hasil perhitungan manual iterasi ke 6 terdapat pada tabel 4.8.

Tabel 4. 10 Hasil perhitungan manual iterasi ke 6

UNIT		P_i^{min} (MW)	P_i (MW)	Status	
1	455	150	719,5852	Max	455
2	455	150	847,8469	Max	455
3	130	20	768,3446	Max	130
4	130	20	768,3446	Max	130
5	470	150	317,834		317,83
6	460	135	714,8039	max	460
7	465	135	907,0219	max	465
8	300	60	-1434,45	min	60
9	162	25	-421,12	min	25
10	160	20	-78,8396	min	20
11	80	20	-94,7697	min	20
12	80	20	57,16597		57,166
13	85	25	-3490,15	min	25
14	55	15	-412,05	min	15
15	55	15	-211,343	min	15
Total					2650

4.2.2 Perbandingan Hasil software dengan Algoritma E*nhanced Lambda Iteration* dan Software dengan *Lambda Iteration*

Berdasarkan perbandingan hasil runing keduanya terlihat *software* pada tes sistem 2 dengan algoritma *enhanced lambda iteration* cukup 6 iterasi untuk mendapatkan nilai *increamental fuel cost* yang *convergen*. Sedangkan pada lambda iterasi biasa membutuhkan 21 iterasi untuk mendapatkan nilai *increamental fuel cost* yang *convergen*. Perbandingan hasil dari keduanya terdapat dalam tabel 4.9. Berdasarkan hasil tersebut diketahui, dengan mempergunakan algoritma *enhanced lambda iteration* dapat mempercepat proses iterasi untuk mendapatkan nilai final lambda (*increamental cost*).

Tabel 4. 11 Perbandingan hasil *software* ELI dengan *lambda kasus 2*

Lambda Iteration				Enhanced Lambda Iteration			
Iterasi	Final lambda	Pi (MW)	fcost ($/h)	Iterasi	Final lambda	Pi (MW)	fcost ($/h)
		455.0	5328.43			455.0	5328.43
		455.0	5261.99			455.0	5261.99
		264,1	1537.62			130.0	1537.62
		125,2	1537.62			130.0	1537.62
		172,1	3787.55			317.8	3787.55
		83,6	5339.83			460.0	5339.83
		465.0	5216.46			465.0	5216.46
21	10.5303	60.0	918.31	6	10.5303	60.0	918.31
		25.0	454.47			25.0	454.47
		20.0	390.83			20.0	390.83
		20.0	412.49			20.0	412.49
		57.2	814.23			57.2	814.23
		25.0	553.51			25.0	553.51
		15	491.26			15	491.26
		15.0	510.94			15.0	510,94
Total		2650.0	32555.54	Total		2650.0	32555.55

4.2.3 Perbandingan Hasil *running* kasus 2 dengan Algoritma *Enhanced Lambda Iteration* dan *Quadratic Programing.*

Pada sub bab ini memperbandingkan hasil *running* powergen dengan algoritma e*nhanced lambda iteration* terhadap hasil *running* matlab dengan q*uadratic programing*. Pembangdingan keduanya dengan tetap mempergunakan data pada tes sistem 2 [2]. Yang dibandingkan pada sub bab ini adalah biaya pembangkitan totalnya.

Tabel 4. 12 Perbandingan ELI dengan QP kasus 2

	Quadratic Programming	Enhanced Lambda
Total cost $/hr	32555.88759	32555.55

4.3 Hasil Pengujian Tes Sistem 3

Pada pengujian tes sistem 3, menggunakan 40 unit pembangkit termal dengan beban 10500 MW. Dengan karakteristik sistem diperlihatkan pada tabel 4.11 [7]. Pada tes sistem 3 ini terdapat variasi nilai lambda (*increamental cost*) pada setiap iterasinya. Variasi nilai

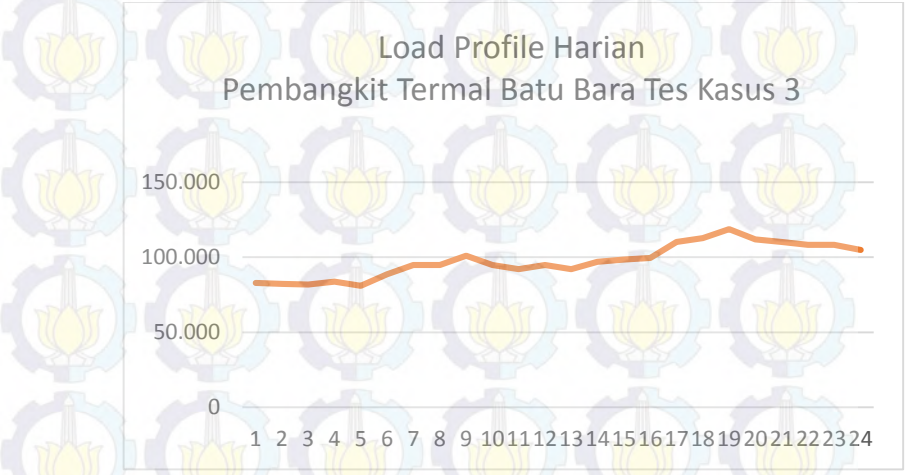
lambda ini terdapat pada tabel 4.12. Pembangkitan pada setiap unit dilambangkan dengan *Pi* dan *Fcost* adalah biaya pembangkitan pada setiap unit. Pada tes sistem 3 tidak dilakukan perhitungan manual. Hal tersebut dikarenakan hasil uji tes sistem 1 dan 2 telah sesuai, maka dapat dipastikan hasil optimasi di tes sistem 3 juga tepat.

Tabel 4. 23 Karakteristik 40 unit pembangkit

Unit	Pmin (MW)	Pmax (MW)	a	b	c	Unit	Pmin (MW)	Pmax (MW)	A	b	c
1	36	114	0,0069	6,73	94,705	20	242	550	0,00313	7,97	647,81
2	36	114	0,0069	6,73	94,705	21	254	550	0,00298	6,63	785,96
3	60	120	0,02028	7,07	309,54	22	254	550	0,00298	6,63	785,96
4	80	190	0,00942	8,18	369,04	23	254	550	0,00284	6,66	794,53
5	47	97	0,0114	5,35	148,89	24	254	550	0,00284	6,66	794,53
6	68	140	0,01142	8,05	222,33	25	254	550	0,00277	7,1	801,32
7	110	300	0,00357	8,03	287,71	26	254	550	0,00277	7,1	801,32
8	135	300	0,00492	6,99	391,98	27	10	150	0,52124	3,33	1055,1
9	135	300	0,00573	6,6	455,76	28	10	150	0,52124	3,33	1055,1
10	130	300	0,00605	12,9	722,82	29	10	150	0,52124	3,33	1055,1
11	94	375	0,00515	12,9	635,2	30	47	97	0,0114	5,35	148,89
12	94	375	0,00569	12,8	6544,69	31	60	190	0,0016	6,43	222,92
13	125	500	0,00421	12,5	913,4	32	60	190	0,0016	6,43	222,92
14	125	500	0,00752	8,84	1760,4	33	60	190	0,0016	6,43	222,92
15	125	500	0,00708	9,15	1728,3	34	90	200	0,0001	8,95	107,87
16	125	500	0,00708	9,15	1728,3	35	90	200	0,0001	8,62	116,58
17	220	500	0,00313	7,97	647,85	36	90	200	0,0001	8,62	116,58
18	220	500	0,00313	7,95	649,69	37	25	110	0,0161	5,88	307,45
19	242	550	0,00313	7,97	647,83	38	25	110	0,0161	5,88	307,45
20	242	550	0,00313	7,97	647,81	39	25	110	0,0161	5,88	307,45
21	254	550	0,00298	6,63	785,96	40	242	550	0,00313	7,97	647,83

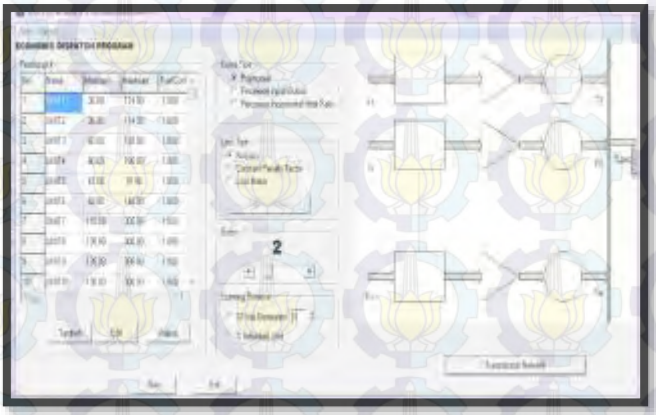
Tabel 4.14 *Load profile* harian kasus 3

Jam	Total Beban	Fuel cost	Jam	Total Beban	Fuel cost
1	7000	82.819,5774	13	8000	92.166,6088
2	6950	82.369,0130	14	8500	97.029,8865
3	6900	81.919,6469	15	8650	98.512,2573
4	7100	83.725,8055	16	8750	99.507,8118
5	6800	81.022,0952	17	9780	110.282,8942
6	7633	88.668,7559	18	10000	112.722,6287
7	8269	94.769,1834	19	10500	118.659,6151
8	8269	94.769,1834	20	9923	111.862,7929
9	8905	101.064,1239	21	9780	110.282,8942
10	8269	94.769,1834	22	9600	108.320,8610
11	8000	92.166,6088	23	9600	108.320,8610
12	8269	94.769,1834	24	9287	104.988,5463



Gambar 4. 9 *Load Profile* haasrian Kasus 3

Data pembangkit pada tabel diatas di input-kan pada menu EDC di jendela data pembangkit. seperti pada tabel 4.11 Sehingga tampilan data hasil pengisian data-data pembangkit seperti pada Gambar 4.7.



Gambar 4. 10 Tampilan data pada tes sistem3 di menu EDC

Apabila semua data telah dilengkapi dan dipilih algoritma Enhanced lambda iteration sebagai metode untuk optimasinya, maka didapatkan *fuel cost* untuk membangkitkan beban 10500 MW adalah sebasar 118659,62 ($/h). Variasi nilai lambda setiap iterasi pada tes sistem 3 untuk mendapatkan *increamental cost* sebesar 12.9308 $/MWh.

Tabel 4. 15 Variasi nilai lambda tiap iterasi tes sistem 3

Iterasi	1	2	3	4
Lambda $/MWh	9,038	9,942	10,930	11,673
P total (MW)	7014,614	7014,614	9718,873	10227,484
Iterasi	5	6	7	
Lambda $/MWh	12,071	12,829	12,931	
P total (MW)	10321,293	10478,769	10500,000	

4.3.1 Perbandingan Hasil software dengan Algoritma E*nhanced Lambda Iteration* dan Software dengan *Lambda Iteration*

Berdasarkan perbandingan hasil runing keduanya terlihat *software* pada tes sistem 3 dengan algoritma *enhanced lambda iteration* cukup 7

iterasi untuk mendapatkan nilai *increamental fuel cost* yang *convergen*. Sedangkan pada lambda iterasi biasa membutuhkan 21 iterasi untuk mendapatkan nilai *increamental fuel cost* yang *convergen.* Perbandingan hasil dari keduanya terdapat dalam tabel 4.13 dan tabel 4.14.

Tabel 4. 36 Hasil running software dengan ELI kasus 3

Iterasi	enhanced lambda					
	Unit	*Pcost*	*Fcost*	Unit	*Pcost*	*Fcost*
7	1	114	951.59	21	550	5333.91
	2	114	951.60	22	551	5333.92
	3	120	1449.97	23	550	5316.63
	4	190	2263.29	24	550	5316.64
	5	97	775.10	25	550	5544.24
	6	140	1573.16	26	550	5544.25
	7	300	3018.01	27	10	1140.52
	8	300	2931.78	28	10	1140.52
	9	300	2951.46	29	10	1140.52
	10	130	2502.06	30	97	775.10
	11	94	1893.30	31	190	1502.38
	12	94	1908.16	32	190	1502.38
	13	124	2528.13	33	190	1502.38
	14	272	4721.14	34	200	1901.87
	15	267	4676.12	35	200	1844.58
	16	267	4676.12	36	200	1844.58
	17	500	5415.35	37	110	1149.06
	18	500	5407.19	38	110	1149.06
	19	550	5978.15	39	110	1149.06
	20	550	5978.13	40	550	5.978.155
	Total				10500	1.186.596.151

Tabel 4. 47 Hasil running *software* dengan *lambda kasus 3*.

Iterasi	Lambda					
	Unit	*Pcost*	*Fcost*	Unit	*Pcost*	*Fcost*
21	1	114	951.59	21	550	5333.91
	2	114	951.60	22	551	5333.92
	3	120	1449.97	23	550	5316.63
	4	190	2263.29	24	550	5316.64
	5	97	775.10	25	550	5544.24
	6	140	1573.16	26	550	5544.25
	7	300	3018.01	27	10	1140.52
	8	300	2931.78	28	10	1140.52
	9	300	2951.46	29	10	1140.52
	10	130	2502.06	30	97	775.10
	11	94	1893.30	31	190	1502.38
	12	94	1908.16	32	190	1502.38
	13	124	2528.13	33	190	1502.38
	14	272	4721.14	34	200	1901.87
	15	267	4676.12	35	200	1844.58
	16	267	4676.12	36	200	1844.58
	17	500	5415.35	37	110	1149.06
	18	500	5407.19	38	110	1149.06
	19	550	5978.15	39	110	1149.06
	20	550	5978.13	40	550	5.978.155
	Total				10500	118.659,77

4.3.2 Perbandingan Hasil *running* kasus 2 dengan Algoritma E*nhanced Lambda Iteration* dan Q*uadratic Programing.*

Pada sub bab ini memperbandingkan hasil *running* powergen dengan algoritma e*nhanced lambda iteration* terhadap hasil *running* matlab dengan q*uadratic programing*. Pembangdingan keduanya dengan tetap mempergunakan data pada tes sistem 3 [7]. Yang dibandingkan pada sub bab ini biaya pembangkitan total untuk memenuhi permintaan beban.

Tabel 4. 58 Perbandingan ELI dengan QP kausu 3

	Quadratic Programming	Enhanced Lambda
Total cost $/hr	119.812,265	118.659.62

--Halaman ini sengaja dikosongkan--

BAB V
PENUTUP

5.1 Kesimpulan

Dari hasil pengembangan *software* yang telah dilakukan dapat disimpulkan beberapa hal sebagai berikut :

1. Hasil pengembangan s*oftware* dengan menambahkan *Enhanced lambda iteration* (ELI) dapat memecahkan masalah economic dispatch tanpa mempertimbangkan kerugian transmisi pada setiap tes sistem yang dilakukan. Yaitu pada 6 unit, 15 unit, dan 40 unit pembangkit.
2. Dari hasil simulasi terlihat ELI mempunyai kemampuan untuk meminimalisir konvergensi lambat karena pemilihan nilai awal *(incremental cost)* yang tidak tepat.
3. Hasil akhir *software* yang dikembangkan ini dapat digunakan untuk memperbaharui software yang telah ada. Dengan tambahan metode optimasi pada *economi dispatch.*

5.2 Saran

Saran yang diberikan setelah selesainya tugas akhir ini adalah sebagai berikut:

1. Diperlukan pengujian lebih lanjut dengan sistem yang lebih kompleks (contohnya sistem Jawa-Bali).
2. Dibandingkan dengan beberapa metode lain seperti PSO atau GA untuk melihat kemampuan ELI dalam melakukan optimization.

--halaman ini sengaja dikosongkan—

LAMPIRAN

//perhitungan manual tes sistem 1 dan tes sistem 2

1. Tes sistem 1

Dari data tabel 3.1 dapat dihitung nilai lambda dan daya terbangkit seperti berikut:

$$\lambda = \frac{1263 + \frac{7}{0.014} + \frac{10}{0.019} + \frac{8.5}{0.018} + \frac{11}{0.018} + \frac{10.5}{0.016} + \frac{12}{0.015}}{\frac{1}{0.014} + \frac{1}{0.019} + \frac{1}{0.018} + \frac{1}{0.018} + \frac{1}{0.016} + \frac{1}{0.015}}$$

$$\lambda = 13.2539 \ \frac{\$}{\text{MWH}}$$

$$P_i = \frac{\lambda - b_i}{2C_i}$$

Tabel L.1 Hasil manual kasus 1 dengan beban 1263MW

Iterasi	$P_1(MW)$	$P_2(MW)$	$P_3(MW)$	$P_4(MW)$	$P_5(MW)$	$P_6(MW)$
1	446.707	171.258	264.1057	125.216	172.1189	83.59
Total	1263					

Tabel L.2 Hasil perhitungan biaya pembangkitan kasus 1

Unit	1	2	3	4	5	6	Ptot
P_i (MW)	446.707	171.258	264.1057	125.216	172.1189	83.59	1263
F_{cost} ($/h)	4763.78	2191.21	3092.66	1718.50	2264.25	1245.53	15275.93

2. Tes sistem 2

Berdasarkan karakteristik pembangkit tabel 3.4 nilai lambda dapat dihitung dengan menggunakan persamaan 3.5 pada saat beban 2650 MW.

- Iterasi 1

$$\lambda = \frac{P_D + \sum \frac{b_i}{2C_i}}{\sum \frac{1}{2C_i}}$$

$$\lambda = \frac{2650+133827,53}{12668,7378} = 10,7727 \ \frac{\$}{\text{MWH}}$$

Tabel L.3 Hasil perhitungan kasus2 pada iterasi 1

UNIT	P_i^{max} (MW)	P_i^{max} (MW)	P_i (MW)	Status	P_i (MW)
1	455	150	1125	max	455
2	455	150	1510	max	455
3	130	20	397,6	max	130
4	130	20	876	max	130
5	470	150	909	max	470
6	460	135	1117	max	460
7	465	135	1240	max	465
8	300	60	-1075	min	60
9	162	25	-270	min	25
10	160	20	21,93		21,93
11	80	20	-60,96	min	20
12	80	20	79,15		79,15
13	85	25	-3163,4	min	25
14	55	15	-349,2	min	15
15	55	15	-184	min	15
Total					2826,08

Berdasarkan tabel diatas dapat menghitung nilai ΔP dengan persamaan 3.11.

$\Delta P = P_D - P_t$

$\Delta P = 2650 - 2826{,}08$

$\Delta P = -176{,}80$

Ietasi ke-2 pada kasus ke 2

$\Delta P \leq 0$

$\lambda^{[2]} = \lambda^{[1]} * 90\%$

$\lambda^{[2]} = 10{,}7727 * 90\%$

$\lambda^{[2]} = 9{,}69543 \frac{\$}{\text{MWH}}$

Tabel L4 Hasil perhitungan kasus2 pada iterasi 2

UNIT	P_i^{max} (MW)	P_i^{max} (MW)	P_i (MW)	Status	P_i (MW)
1	455	150	-676,5	min	150
2	455	150	-1433	min	150
3	130	20	397,62	min	130
4	130	20	397,62	min	130
5	470	150	-1718	min	150
6	460	135	-672	min	135
7	465	135	-239,8	min	135
8	300	60	-2669	min	60
9	162	25	-938,4	min	25
10	160	20	-425,8	min	20
11	80	20	-211,2	min	20
12	80	20	-18,55	min	20
13	85	25	-4615	min	25
14	55	15	-628,5	min	15
15	55	15	-305,2	min	15
Total					1180

$$\Delta P = P_D - P_t$$
$$\Delta P = 2650 - 1180$$
$$\Delta P = 1470$$

Ietasi ke-3 pada kasus ke 2

$$\lambda^{[3]} = \frac{\lambda^{[2]} * \Delta P^{[1]} - \lambda^{[1]} * \Delta P^{[2]}}{\Delta P^{[1]} - \Delta P^{[2]}}$$
$$\lambda^{[3]} = \frac{\left(9{,}69543 * (-176{,}08)\right) - (10{,}7727 * 1470)}{-176{,}08 - 1470}$$
$$\lambda^{[3]} = 10{,}6575 \frac{\$}{\text{MWH}}$$
$$P_i = \frac{\lambda - b_i}{2C_i}$$

Tabel L.5 Hasil perhitungan kasus2 iterasi 3

UNIT	P_i^{max} (MW)	P_i^{max} (MW)	P_i (MW)	Status	P_i (MW)
1	455	150	932,216	max	455
2	455	150	1195,26	max	455
3	130	20	824,8069266	max	130
4	130	20	824,8069	max	130
5	470	150	627,9639	max	470
6	460	135	926,0219	max	460
7	465	135	1081,683	max	465
8	300	60	-1246,3532	min	60
9	162	25	-342,33878	min	25
10	160	20	-25,99119	min	20
11	80	20	-77,04054	min	20
12	80	20	68,69809		68,698
13	85	25	-3318,780	min	25
14	55	15	-379,0914	min	15
15	55	15	-197,046863	min	15
Total					2813,6981

$\Delta P = P_D - P_t$

$\Delta P = 2650 - 2813{,}6981$

$\Delta P =$-163,698 MW

Ietasi ke-4 pada kasus ke 2

$$\lambda^{[3]} = \frac{\lambda^{[3]} * \Delta P^{[2]} - \lambda^{[2]} * \Delta P^{[3]}}{\Delta P^{[2]} - \Delta P^{[3]}}$$

$$\lambda^{[3]} = \frac{(10{,}6575 * 1470) - (9{,}6954 * (-163.698))}{1470 - (-163.698)}$$

$$\lambda^{[3]} = 10{,}5611 \frac{\$}{\text{MWH}}$$

$$P_i = \frac{\lambda - b_i}{2C_i}$$

Tabel L.6 Hasil perhitungan kasus2 iterasi 4

UNIT	P_i^{max} (MW)	P_i^{max} (MW)	P_i (MW)	Status	P_i (MW)
1	455	150	771,017326	max	455
2	455	150	931,880768	max	455
3	130	20	782,0019367	max	130
4	130	20	782,001936	max	130
5	470	150	392,849662		392,849662
6	460	135	765,894288	max	460
7	465	135	949,269727	max	465
8	300	60	-1388,952128	min	60
9	162	25	-402,0642123	min	25
10	160	20	-66,05637516	min	20
11	80	20	-90,481266	min	20
12	80	20	59,955410		59,95541
13	85	25	-3448,6949	min	25
14	55	15	-404,07767	min	15
15	55	15	-207,8852753	min	15
Total					2727,8051

$\Delta P = P_D - P_t$

$\Delta P = 2650 - 2727{,}8051$

$\Delta P =$ -77,805073MW

Iterasi ke-5 pada kasus ke 2

$$\lambda^{[3]} = \frac{\lambda^{[4]} * \Delta P^{[3]} - \lambda^{[3]} * \Delta P^{[4]}}{\Delta P^{[3]} - \Delta P^{[4]}}$$

$$\lambda^{[3]} = \frac{(10{,}6575 * (-77{,}8050773)) - (10{,}5611 * (-163.698))}{(-163.698) - (-77{,}8050773)}$$

$$\lambda^{[3]} = 10{,}473749 \ \frac{\$}{\text{MWH}}$$

$$P_i = \frac{\lambda - b_i}{2C_i}$$

Tabel L.7 Hasil perhitungan kasus2 iterasi 5

UNIT	P_i^{max} (MW)	P_i^{max} (MW)	P_i (MW)	Status	P_i (MW)
1	455	150	624,9975667	max	455
2	455	150	693,3020352	max	455
3	130	20	743,2275954	max	130
4	130	20	743,2275954	max	130
5	470	150	179,8744997		179,8744997
6	460	135	620,8447589	max	460
7	465	135	829,3249243	max	465
8	300	60	-1518,123454	min	60
9	162	25	-456,1657095	min	25
10	160	20	-102,3489007	min	20
11	80	20	-102,65637	min	20
12	80	20	52,035965		52,035965
13	85	25	-3566,3766	min	25
14	55	15	-426,71111	min	15
15	55	15	-217,7031094	min	15
Total					2506,9105

$\Delta P = P_D - P_t$

$\Delta P = 2650 - 2506{,}9105$

$\Delta P = 143{,}089536$ MW

Iterasi ke-5 pada kasus ke 2

$$\lambda^{[3]} = \frac{\lambda^{[5]} * \Delta P^{[4]} - \lambda^{[4]} * \Delta P^{[5]}}{\Delta P^{[4]} - \Delta P^{[5]}}$$

$$\lambda^{[3]} = \frac{(10{,}4737 * (-77{,}8050773)) - (143{,}08953 * (-163.698))}{(-77{,}8050773) - (143{,}08953)}$$

$$\lambda^{[3]} = 10{,}530312 \ \frac{\$}{\text{MWH}}$$

Tabel L.5 Hasil perhitungan kasus2 iterasi 6

UNIT	P_i^{max} (MW)	P_i^{max} (MW)	P_i (MW)	Status	P_i (MW)
1	455	150	719,5852065	max	455
2	455	150	847,8468674	max	455
3	130	20	768,3445619	max	130
4	130	20	768,3445619	max	130
5	470	150	317,8340329		317,8340329

Tabel L.5 Hasil perhitungan kasus2 iterasi 6 (Lanjutan)

UNIT	P_i^{max} (MW)	P_i^{max} (MW)	P_i (MW)	Status	P_i (MW)
6	460	135	714,8039094	max	460
7	465	135	907,0219141	max	465
8	300	60	-1434,449773	min	60
9	162	25	-421,1202271	min	25
10	160	20	-78,83958708	min	20
11	80	20	-94,769666	min	20
12	80	20	57,165967		57,165967
13	85	25	-3490,1456	min	25
14	55	15	-412,04978	min	15
15	55	15	-211,3433828	min	15
Total					2650

$\Delta P = P_D - P_t$

$\Delta P = 2650 - 2650$

$\Delta P = 0$

Tabel L.6 Hasil akhir perhitungan biaya pembangkitan kasus 2

Unit	1	2	3	4	5	6	7	8
Pi(MW)	455	455	130	130	317,834	460	465	60
Fcost	5328,43	5262,5	1537,6	1537,6	3787,553	5340	5216,46	918,3

Tabel L.7 Lanjutan hasil akhir perhitungan biaya pembangkitan kasus 2

Unit	9	10	11	12	13	14	15	Total
Pi(MW)	25	20	20	57,16	25	15	15	2650
Fcost	454,4744	390,83	412,4	814,2	558,5	491,264	510,941	32561,01

3.4.1 Kasus 3

Data pembangkit untuk kasus 3 dapat dilihat pada tabel 3.19. Perhitungan pada kasus 3 ini memiliki alur sama seperti kasus 1 dan kasus 2. Hanya saja pada kasus 3 ini unit pembangkit yang diuji memiliki jumlah yang jauh berbeda dengan kasus 1 dan kasus 2[7].

Tabel L.8 Karakteristik pembangkit kasus 3

Unit	Pmin(MW)	Pmax(MW)	a	b	c
1	36	114	0,0069	6,73	94,705
2	36	114	0,0069	6,73	94,705
3	60	120	0,02028	7,07	309,54
4	80	190	0,00942	8,18	369,04
5	47	97	0,0114	5,35	148,89
6	68	140	0,01142	8,05	222,33
7	110	300	0,00357	8,03	287,71
8	135	300	0,00492	6,99	391,98
9	135	300	0,00573	6,6	455,76
10	130	300	0,00605	12,9	722,82
11	94	375	0,00515	12,9	635,2
12	94	375	0,00569	12,8	6544,69
13	125	500	0,00421	12,5	913,4
14	125	500	0,00752	8,84	1760,4
15	125	500	0,00708	9,15	1728,3
16	125	500	0,00708	9,15	1728,3
17	220	500	0,00313	7,97	647,85
18	220	500	0,00313	7,95	649,69
19	242	550	0,00313	7,97	647,83
20	242	550	0,00313	7,97	647,81
21	254	550	0,00298	6,63	785,96
22	254	550	0,00298	6,63	785,96
23	254	550	0,00284	6,66	794,53
24	254	550	0,00284	6,66	794,53
25	254	550	0,00277	7,1	801,32
26	254	550	0,00277	7,1	801,32
27	10	150	0,52124	3,33	1055,1
28	10	150	0,52124	3,33	1055,1
29	10	150	0,52124	3,33	1055,1
30	47	97	0,0114	5,35	148,89
31	60	190	0,0016	6,43	222,92
32	60	190	0,0016	6,43	222,92
33	60	190	0,0016	6,43	222,92
34	90	200	0,0001	8,95	107,87
35	90	200	0,0001	8,62	116,58
36	90	200	0,0001	8,62	116,58
37	25	110	0,0161	5,88	307,45
38	25	110	0,0161	5,88	307,45
39	25	110	0,0161	5,88	307,45
40	242	550	0,00313	7,97	647,83

Kode logika (*Logic code)*

```
unit Unit3;

interface
uses  sysutils, Messages, Dialogs;

const

      max_units = 200;
      max_order = 10;
      max_curve_points = 10;
      max_total_segments = 200;
      total_gen_tolerance = 0.0001 ;
      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,lamenhanced);
      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}
    UR:unit_array_real;
    DR:unit_array_real;
    unitsebelum:unit_array_real;
    Ramprate:boolean;
    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}
    unitmax:unit_array_real;          {maksimum unit generation awal}
    unitmin:unit_array_real;          {minimum unit generation awal}

    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;
    mwlosses, 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, SRGenTotal : real;
    FF:Text;
    NoGenerator:integer ;
    NomerOrder:integer;
    ModeDataPembangkit:integer;
    ProsesRun :boolean;
    SR : real;
    IndividuSR:unit_array_real;

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 lambda_enhanced_iterasion( 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);
//procedure TotalGeneration (pmin,pmax,
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, ihr_a,ihr_b : 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
    j := 0;
    repeat
    j := j + 1;
    until (ihr_mwpoint[ i,j ] > unitmw) or (j = curveorder);
```

```
    ihr_a:=0.5*(ihr_cost[i,j]-ihr_cost[i,j-1])/(ihr_mwpoint[i,j]-
ihr_mwpoint[i,j-1]);
    ihr_b:=((ihr_mwpoint[i,j]*ihr_cost[i,j-1])-(ihr_mwpoint[i,j-
1]*ihr_cost[i,j]))/(ihr_mwpoint[i,j]-ihr_mwpoint[i,j-1]);

    unitcost:= ihr_a*unitmw*unitmw + ihr_b*unitmw + minput[i];
    unitcost:= unitcost*fuelcost[i];
    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, totalload : real;
        unitihr, unitinccost, unitcost : 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:9:4);
```

```
  totalgen := totalgen + p[i];
  totalcost := totalcost + unitcost;
  end;
writeln(outfile,
      '---------  ------                          --------------');
write(outfile, ' totals');
write(outfile, totalgen:9:1,
'                              ', totalcost:9:4);
writeln(outfile);
writeln(outfile, ' lambda = ', lambda:10:4 );
writeln(outfile);

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

if schedtype=totload then totalload := schedmw;

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

writeln(outfile, 'total load = ',totalload:10:1,
           '  total losses = ',mwlosses:10:1);
if totalload+mwlosses>pgenmax then showmessage('WARNING!! Load
cant be fully served, Load Shedding Needed');

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

if curvetype <> pio then
     begin
       ShowMessage(' 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;

label return;

begin

    mwlosses := b00;
```

```
    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;


    for i := 1 to ngen do
      begin
      penfac_old := penfac[ i ];
      incloss := b0[i];
      for j := 1 to ngen do
        incloss := incloss + 2.0 * ( p[j]/100.0 ) * b[i,j];
      penfac_new := 1.0 / ( 1.0 - incloss );
      penfac[ i ] := penfac_old + alpha *
                           ( penfac_new  -  penfac_old)
      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 lambda_enhanced_iterasion(var lambda : real );
var
      i, n, lossiter,iterasi : integer;
      lambdamin, lambdamax : real;
      lambdastart, deltalambda, targetgen: real;
      unitihr,a,b, unitmw, totalgen, eli1, eli2, eli3, eli4 : real;
      lambda1, lambda2, lambda3, ptot, ptot1, ptot2: real;
      deltap,deltap1,deltap2,deltap3 : real;
```

```
        endloop : boolean;

begin

 //---------------- rumus baru ------------------
 if schedtype = totgen then targetgen := schedmw;
 if schedtype = totload then targetgen := schedmw + mwlosses;

 iterasi := 1;
 repeat
  if iterasi = 1 then
  begin
   eli1:=0;
   eli2:=0;
   ptot:=0;
   for i:=1 to ngen do
   begin
    eli1:= eli1 +  1 / (2* coeff[i,2]);
    eli2:= eli2 + coeff[i,1] / (2* coeff[i,2]);
   end;
   lambda1 := (targetgen + eli2) / eli1;

    for i:=1 to ngen do
    begin
     p[i]:= (lambda1 - coeff[i,1]) / (2 * coeff[i,2]);       {hitung p}
     if p[i] <= pmin[i] then p[i] := pmin[i];       {check limit nilai pmin}
     if p[i] >= pmax[i] then p[i] := pmax[i];       {check limit nilai
pmax}
      Ptot:= ptot + p[i];                          {hitung ptot}
      Writeln( ff, 'p= ', p[i]:3:2);
    end;
   deltap1 := targetgen - ptot;                  {hitung delta p}
   Writeln( ff, 'iteration= ', iterasi);
   Writeln( ff, 'lambda= ', lambda1:10:4);
   Writeln( ff, 'target generator= ', targetgen:10:4);
   deltap := deltap1;
   Writeln( ff, 'deltap1=', deltap:10:4);
   end;
```

```
  if iterasi = 2 then
  begin
   ptot1:=0;
   lambdamax := lambda1 * 1.1;
   if deltap1 > 0.0001 then lambda2:=lambdamax;
   lambdamin := lambda1 * 0.9;
   if deltap1 < 0.0001 then lambda2:=lambdamin;

    for i:=1 to ngen do
    begin
     p[i]:= (lambda2 - coeff[i,1]) / (2 * coeff[i,2]);       {hitung p}
     if p[i] <= pmin[i] then p[i] := pmin[i];       {check limit nilai pmin}
     if p[i] >= pmax[i] then p[i] := pmax[i];       {check limit nilai
pmax}
     Ptot1:= ptot1 + p[i];                          {hitung ptot}
     Writeln( ff, 'p= ', p[i]:3:2);
    end;
   deltap2 := targetgen - ptot1;                    {hitung delta p}
   Writeln( ff, 'iteration= ', iterasi);
   Writeln( ff, 'lambda= ', lambda2:10:4);
   Writeln( ff, 'target generator= ', targetgen:10:4);
   deltap := deltap2;
   Writeln( ff, 'deltap2=', deltap:10:4);
  end;

   //Writeln(ff,'lambda1=', lambda1:10:4 );
   //Writeln( ff, 'lambda=', lambda:10:4);
   //Writeln( ff, 'deltap=', deltap:10:4);

  if iterasi >=3 then
  begin
  ptot2 := 0;
  repeat
   eli3 := 0;
   eli4 := 0;
   a := lambda2 * deltap1;
   b := lambda1 * deltap2;
   eli3 := a - b;
   eli4 := deltap1 - deltap2;
```

```
    if eli4=0 then eli4 := deltap1+deltap2;
    lambda3 := eli3 / eli4;
    for i:=1 to ngen do
    begin
     p[i]:= (lambda3 - coeff[i,1])/(2 * coeff[i,2]);       {hitung p}
     if p[i] <= pmin[i] then p[i] := pmin[i];       {check limit nilai pmin}
     if p[i] >= pmax[i] then p[i] := pmax[i];       {check limit nilai pmax}
     ptot2:= ptot2 + p[i];                          {hitung ptot}
    end;
   deltap3 := targetgen - ptot2;                    {hitung delta p}
   Writeln( ff, 'iteration= ', iterasi);
   Writeln( ff, 'Ptot= ', ptot2:10:4);
   Writeln( ff, 'deltap= ', deltap3:10:4);
   Writeln( ff, 'lambda1=', lambda1:10:4);
   Writeln( ff, 'lambda2=', lambda2:10:4);
   Writeln( ff, 'lambda3=', lambda3:10:4);
   lambda1:=lambda2;
   lambda2:=lambda3;
   deltap := deltap3;
   deltap1:=deltap2;
   deltap2:=deltap3;
   ptot2 :=0;
   iterasi := iterasi+1;

   until (deltap >= -0.0002) and (deltap < 0.0002);// or (iterasi >= 5);
  end;
  iterasi := iterasi + 1;
 until (deltap >= -0.0002) and (deltap < 0.0002);// or (iterasi >= 5);

  end; { End Enhanced Lambda Iteration procedure }


//-----------------------------------------end rumus-------------------


procedure lambda_search_dispatch( var lambda : real );

var
       i, n, lossiter : integer;
```

```
        lambdamin, lambdamax : real;
        lambdastart, deltalambda, targetgen : real;
        unitihr, unitmw, totalgen : real;
        endloop : boolean;

begin

for i := 1 to ngen do                { Set unit output to midrange}
  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
    ihr_ftn (i,pmax[i],maxihr[i]);
    lambda := maxihr[ i ] * penfac[ i ] * fuelcost[ i ];
    if lambda > lambdamax then lambdamax := lambda;
    ihr_ftn (i,pmin[i],minihr[i]);
    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:10:3);


     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;

end; { End Lambda search procedure }

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], DR[i], UR[i],
unitsebelum[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,
DR[i]:15:6, UR[i]:15:6, unitsebelum[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
      begin
      unitihr := ( unitihr + j * coeff[ i,j ] ) * unitmw;
      end;
    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
for i := 1 to ngen do
begin
pmax[i]:=unitmax[i]-(individuSR[i]/100)*unitmax[i];
pmin[i]:=unitmin[i];
if unitsebelum[i] <> 0 then if ramprate then
  begin
  if (unitsebelum[i]+UR[i])<(unitmax[i]-
(IndividuSR[i]/100)*unitmax[i]) then pmax[i] := unitsebelum[i]+UR[i];
  if (unitsebelum[i]-DR[i])>(unitmin[i]) then pmin[i] := unitsebelum[i]-
DR[i];
  end;
end;

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')
                        else 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.


unit Unit4;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls,unit3,unit1;//, unit1;

type
  TFrmEDCSetUpSolution = class(TForm)
    GroupBox1: TGroupBox;
    RbLamSearch: TRadioButton;
    Rbtbllookup: TRadioButton;
    GroupBox2: TGroupBox;
    GroupBox3: TGroupBox;
    RbTotGen: TRadioButton;
    RbTotLoad: TRadioButton;
    Ed: TEdit;
    Lb: TLabel;
    LbMaxGe: TLabel;
    LbMinG: TLabel;
    Button1: TButton;
    LbSpinningReserve: TLabel;
    GroupBox4: TGroupBox;
    RbRampYes: TRadioButton;
    RbRampNo: TRadioButton;
```

```
    RbELI: TRadioButton;

    procedure RbTotGenClick(Sender: TObject);
    procedure RbTotLoadClick(Sender: TObject);
    procedure RbLamSearchClick(Sender: TObject);
    procedure RbtbllookupClick(Sender: TObject);
//    procedure TFrmEDCSetUpSolution(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
//    procedure RbELIClick(Sender: TObject);
    procedure RbELIClick(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FrmEDCSetUpSolution: TFrmEDCSetUpSolution;

implementation

uses Unit5;//, Unit6, Unit1;

{$R *.dfm}

procedure TFrmEDCSetUpSolution.RbTotGenClick(Sender: TObject);
begin
schedtype := totgen;
lb.caption :='Enter Total Generation';
end;

procedure TFrmEDCSetUpSolution.RbTotLoadClick(Sender: TObject);
begin
```

```
schedtype := totload;
lb.caption :='Enter Total Load';
end;

procedure TFrmEDCSetUpSolution.RbLamSearchClick(Sender:
TObject);
begin
solution_type  := lamsearch;
end;

procedure TFrmEDCSetUpSolution.RbtbllookupClick(Sender:
TObject);
begin
solution_type  := tbllookup;
end;

procedure TFrmEDCSetUpSolution.RbELIClick(Sender: TObject);
begin
solution_type  := lamenhanced;
end;

procedure TFrmEDCSetUpSolution.FormActivate(Sender: TObject);
var s:string;
i:integer;
begin
    pgenmax := 0.0;
pgenmin := 0.0;
SRGenTotal :=0.0;

for i := 1 to ngen do
  begin
    ihr_ftn( i, pmax[ i ], maxihr[ i ] );
    ihr_ftn( i, pmin[ i ], minihr[ i ] );

{ calculate maximum and minimum generation available from
generators }
```

```
    SRGenTotal := SRgenTotal + ((IndividuSR[i]/100)*pmax[i]);
    pmax[i] := pmax[i] - ((IndividuSR[i]/100)*pmax[i]);
    pgenmax := pgenmax + pmax[ i ];
    pgenmin := pgenmin + pmin[ i ];
   end;

if FrmEdcMain.rbSR2.Checked then
  begin
    SRGenTotal := pgenmax*SR/100;
    pgenmax := pgenmax-SRGenTotal;
  end;

prosesrun:=false;
str(pgenmax:10:1,s);
lbMaxGe.caption:=' Maximum generation is :'+s;
str(pgenmin:10:1,s);
lbming.caption:= ' Minimum generation is :'+s;
str (SRGenTotal:10:1,s);
LbSpinningReserve.caption := ' Spinning Reserve is : '+s;
 LbSpinningReserve.Visible:=true;
if SRGenTotal =0 then LbSpinningReserve.Visible:=false;
end;

 procedure PRoses;
var s,e,f,d:string;
 k:integer;
begin
    if FrmEDCSetUpSolution.RbRampYes.Checked then Ramprate:=true
else RampRate:=false;
    s:=frmEdcSetupSolution.ed.text;
    val(s, schedmw,k);
    if (schedmw<pgenmin) or (schedmw>pgenmax) then
    begin
      showmessage('EDC not possible with that scheduled generation');
      prosesrun:=false;
    end
```

```
    else
    begin
      prosesrun:=true;
      getfilename(filename,d,f,e);
      title1 := 'File Name : '+f+'.'+e;
      assign(ff,'data.dum');
      rewrite(Ff);
      datadump(Ff) ;
      if solution_type = lamsearch  then lambda_search_dispatch( lambda
);
      if solution_type = tbllookup  then table_lookup_dispatch( lambda );
      if solution_type = lamenhanced  then lambda_enhanced_iterasion(
lambda );
      output_routine(ff, lambda ) ;
      close(fF);

      frmEdcSetupSolution.Close ;
    end;


 end;
procedure TFrmEDCSetUpSolution.Button1Click(Sender: TObject);
begin
 title1:='EDC FILE ' +fileName;


 proses;

end;


procedure TFrmEDCSetUpSolution.FormClose(Sender: TObject;
  var Action: TCloseAction);
var i:integer;
begin
for i :=1 to ngen do
```

```
begin
pmax[i]:=unitmax[i];
pmin[i]:=unitmin[i];
end;
end;

end.
```

DAFTAR PUSTAKA

[1] Willam D. Stevenson, "Power System Analisi", McGrw-Hill, 1994

[2] Prateek K. Singhal, "Enhanced Lambda Iteration Algorithm for the solution of large Scale Economic Dispatch Problem", IEEE press, 2014.

[3] Saadat, Hadi. "Power System Analysis 2nd Edition", McGrowHill, Ch.1, 1999

[4] Wood Allen J, Wollenberg Bruce F, (1996), "Power Generation, Operational, and Control", Second Edition, Jhon Wiley & Sons, Inc.

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

[6] Ida Bagus Krisna P. "Economic Dispatch Menggunakan Ant Colony Optimization pada Sistem Transmisi 500 KV Jawa Bali", Jurusan Teknik Elektro ITS FTI-ITS, Surabaya, 2009

[6] MADCOMS, 2006. "Seri Panduan Pemprograman: Pemprograman Borland Delphi 7", Yogyakarta: ANDI

[8] Ryzkyanto.H, "Dynamic Economic Dispatch dengan Memperhatikan Ramp-Rate Menggunakan Metode Iterasi Lambda Berbasis Delphi", ITS, 2015.

[7] Nidul Sinha, "Evolutionary Programming Techniques for Economic Load Dispatch", IEEE press, 2003.

RIWAYAT HIDUP PENULIS

Santi Triwijaya, terlahir di Malang, 11 September 1991. Penulis menyelesaikan pendidikan Sekolah Menengah Kejuruan di SMKN 1 Tuban pada tahun 2010. Praktikan telah menyelesaikan pendidikan Diploma dan memperoleh gelar Ahli madya (A.Md) dari kampus Diploma Teknik Elektro Universitas Gadjah Mada pada tahun 2013 dengan bidang keahlian Teknik Sistem Tenaga Listrik. Pada tahun 2013, penulis melanjutkan studi S1 di Institut Teknologi Sepuluh November (ITS) Surabaya dengan bidang keahlian yang sama Teknik Sistem Tenaga. Penulis dapat dihubungi melalui alamat Email: santi.triwijaya@gmail.com