TUGAS AKHIR

OPTIMASI

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK VISUALISASI OPTIMASI TAK LINIER SEBAGAI ALAT BANTU KULIAH OPTIMASI

Disusun Oleh :

R. NANIK HIDAJATININGSIH

NRP. 2690 100 042

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA

1999

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK VISUALISASI OPTIMASI TAK LINIER SEBAGAI ALAT BANTU KULIAH OPTIMASI

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer
Pada
Jurusan Teknik Informatika
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
S u r a b a y a

Mengetahui / Menyetujui
Dosen Pembimbing

Dr. Ir. SUPENO DJANALI
NIP. 130 368 610

S U R A B A Y A
Februari, 1999

ABSTRAK

ABSTRAK

Optimasi merupakan salah satu mata kuliah yang diperoleh mahasiswa teknik komputer. Mata kuliah optimasi secara garis besar membahas cara pencarian nilai maksimum dan minimun dari suatu fungsi. Pada tugas akhir ini penulis hanya membatasi pada fungsi - fungsi tak linier.

Bila pencarian nilai optimal dilakukan secara manual, tentunya akan membutuhkan waktu yang cukup lama. Karena waktu yang tidak mencukupi itu seringkali mahasiswa tidak berhasil memperoleh nilai optimal yang diharapkan.
Pencarian ini kadang-kadang sampai beberapa kali iterasi, sehingga waktu yang diperlukan memang cukup lama.

Untuk mengatasi masalah tersebut diperlukan suatu aplikasi komputer yang dapat memvisualisasikan pencarian nilai optimal sekaligus memperlihatkan bentuk fungsi. Dalam tugas akhir ini penulis hanya menampilkan bentuk fungsi pada fungsi-fungsi satu dimensi.

Pembuatan aplikasi komputer tersebut diharapkan dapat membantu penyampaian materi dalam kuliah optimasi. Semoga mahasiswa dapat memperoleh gambaran yang lebih jelas, sehingga lebih mengerti dan memahami materi kuliah.

KATA PENGANTAR

KATA PENGANTAR

Alhamdulillah. Berkat rahmat Allah SWT penulis dapat menyelesaikan tugas akhir dengan judul :

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK VISUALISASI OPTIMASI TAK LINIER SEBAGAI ALAT BANTU KULIAH OPTIMASI

Dalam menyelesaikan tugas akhir ini, penulis mendapat banyak sekali bimbingan dan bantuan baik moril maupun materiil. Oleh karena itu dalam kesempatan ini penulis mengucapkan banyak terima kasih yang sedalam-dalamnya kepada :

1. Dr. Ir. Arif Djunaidy selaku Ketua Jurusan Teknik Informatika.
2. Dr. Ir. Supeno Djanali selaku dosen pembimbing.
3. Drs.Ec. Ir. Riyanarto Sarno, MSc. PhD. selaku dosen wali.
4. Orang tua, Suami, Anak-anak(Afkar,Husein,Aya), Kakak-kakak dan Adik-adik yang telah memberi dorongan dan dukungan baik moril maupun materiil.
5. Rekan-rekan khususnya C-06 yang telah banyak membantu mulai masa perkuliahan.
6. Juga tak lupa penulis mengucapkan terima kasih kepada seluruh dosen dan karyawan serta pihak-pihak yang tak dapat kami sebutkan satu persatu, yang telah membantu penulis sehingga dapat menyelesaikan tugas akhir ini.

Akhirnya penulis menyadari tentunya tugas akhir ini banyak memiliki kekurangan . Oleh karena itu saran serta kritik yang membangun penulis terima dengan lapang dada. Dan semoga tugas akhir ini akan memberikan manfaat bagi para pembaca.

Surabaya, Pebruari 1999

Penulis

DAFTAR ISI

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR TABEL

BAB I

PENDAHULUAN

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pemanfaatan komputer disemua aspek kehidupan sudah menjadi budaya baru bagi masyarakat Indonesia saat ini.

Lingkungan kampus sebagai sebuah lembaga pendidikan juga tidak terhindar dari pemanfaatan komputer diseluruh aktifitasnya.

Dalam penyampaian kuliah diharapkan kehadiran komputer untuk membantu proses belajar mengajar. Dalam tugas akhir ini penulis memberikan contoh pemanfaatan komputer dalam penyampaian materi mata kuliah optimasi yaitu visualisasi optimasi tak linier dua dimensi.

1.2 Maksud dan Tujuan

Tujuan yang ingin dicapai dari perancangan dan pembuatan perangkat lunak visualisasi optimasi ini adalah:

- Memvisualisasikan pencarian nilai optimal fungsi fungsi tak linier untuk menunjang pemberian materi kuliah optimasi.
- Memanfaatkan waktu yang tersedia secara optimal dengan mengganti perhitungan yang dilakukan secara manual dengan perhitungan yang dilakukan komputer.
- Memberikan gambaran yang lebih jelas, sehingga mahasiswa dapat lebih memahami tentang materi yang diberikan.

1.3 Batasan Masalah

Waktu yang diberikan dalam penyampaian mata kuliah optimasi sangat terbatas. Dalam pencarian nilai optimal seringkali hanya dapat menyelesaikan satu atau dua kali iterasi, sedangkan untuk memperoleh nilai optimal dibutuhkan beberapa kali iterasi, sehingga nilai optimal yang akan dicari tidak diperoleh. Dengan pemanfaatan komputer diharapkan dapat mengatasi keterbatasan waktu karena perhitungan yang dilakukan secara manual diganti dengan perhitungan oleh komputer.

Batasan-batasan masalah sehubungan dengan perancangan dan pembuatan perangkat lunak dalam tugas akhir ini adalah sebagai berikut:

Fungsi-fungsi yang akan dicari nilai optimalnya hanya meliputi fungsi-fungsi tak linier. Untuk fungsi-fungsi linier sudah memiliki cara pencarian khusus.

Selain fungsi tak linier fungsi-fungsi tersebut dibatasi pada fungsi-fungsi yang memiliki dimensi yang lebih kecil atau sama dengan dua. Maksudnya variabel-variabel dalam fungsi tersebut maksimal dua macam. Hal ini disebabkan karena fungsi yang memiliki jumlah variabel lebih dari dua untuk memvisualisasikan baik pencarian nilai optimal maupun bentuk fungsi semakin sulit. Kesulitan ini terletak baik dalam pembuatan maupun cara kita memandang tampilan di layar, mengingat layar monitor berdimensi dua.

1.4 Metodologi

Metodologi yang dipergunakan dalam tugas akhir ini adalah :

a. *Study* literatur, yaitu mencari dan mempelajari literatur - literatur yang berkaitan dengan masalah visualisasi dan optimasi.
b. Perancangan algoritma subrutin -subrutin yang diperlukan.

c. Pembuatan program visualisasi optimasi tak linier.

d. Pengujian dan Analisa program visualisasi optimasi tak linier.

e. Penarikan kesimpulan.

f. Pembuatan laporan tugas akhir.

1.5 Sistematika Penulisan Laporan Tugas Akhir

Sistematika penulisan laporan tugas akhir ini adalah :

Bab I Merupakan bab Pendahuluan.

Pada bab Pendahuluan membahas tentang latar belakang masalah, maksud dan tujuan, batasan masalah, metodologi.

Bab II Dasar Teknik Optimasi

Pada bab Dasar Teknik Optimasi membahas tentang contoh penggunaan optimasi tak linier dalam kehidupan dilengkapi pemakaian grafik serta sedikit penjelasan tentang pengertian yang berhubungan dengan optimasi tak linier.

Bab III Optimasi Tak Linier Dua Dimensi

Pada bab Optimasi Tak Linier Dua Dimensi membahas tentang metode-metode optimasi tak linier baik satu dimensi maupun dua dimensi. Metode-metode tersebut adalah metode Fibonacci, metode Newton, metode Gradien, metode *Conjugate* Gradien dan metode Pinalti dan Penghalang. Dilengkapi juga algoritma dari tiap-tiap metode.

Bab IV Perancangan Visualisasi Optimasi Tak Linier

Pada bab Perancangan Visualisasi Optimasi Tak Linier membahas tentang bagan-alir, perancangan dan pembuatan program, masukan-masukan yang digunakan serta output dari progam.

Bab V Pengujian dan Analisa

Pada Pengujian dan Analisa mencoba kemampuan dari program dengan memberi masukan. Keluaran yang dihasilkan apakah sudah sesuai dengan yang diharapkan. Selain itu juga membandingkan perhitungan yang dilakukan secara manual dengan perhitungan menggunakan program yang telah dibuat.

Bab VI Kesimpulan dan saran.

Pada bab Kesimpulan dan Saran membahas kesimpulan yang diperoleh setelah program dibuat dan dilakukan pengujian. Selanjutnya untuk pengembangan lebih lanjut memberikan langkah-langkah yang perlu ditempuh.

BAB II

DASAR TEORI OPTIMASI

BAB II
DASAR TEKNIK OPTIMASI

Optimasi tak linier adalah pencarian dan penentuan nilai maksimum dan nilai minimum dari suatu persaman tak linier.

Secara umum, *optimasi tak linier* adalah menentukan $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ sehingga

$$\mathbf{f}(\mathbf{x}) \quad \text{maksimum / minimum,}$$

dengan kendala $\quad g_i(\mathbf{x}) \leq b_i$, untuk setiap $i = 1, 2, \ldots, m$.

dan $\quad \mathbf{x} \geq 0$,

di mana fungsi $\mathbf{f}(\mathbf{x})$ dan fungsi $g_i(\mathbf{x})$ merupakan fungsi-fungsi dengan n peubah.

Tidak ada algoritma tertentu dan khusus untuk menyelesaikan bentuk umum di atas, tetapi telah banyak pembahasan yang dibuat mengenai hal ini terhadap kasus per kasus dengan memberikan beberapa asumsi terhadap fungsi-fungsi yang muncul. Oleh karena bidang ini cukup luas dan kita tidak memiliki penelitian yang lengkap, maka dalam hal ini kita hanya membahas beberapa aplikasi yang sederhana, latar belakangnya dan penyelesaian dari beberapa jenis masalah pemrograman tak linier.

2.1 Contoh Penggunaan

Beberapa contoh masalah berikut menggambarkan sedikit dari beraneka ragam masalah yang telah diselesaikan dengan optimasi tak linier.

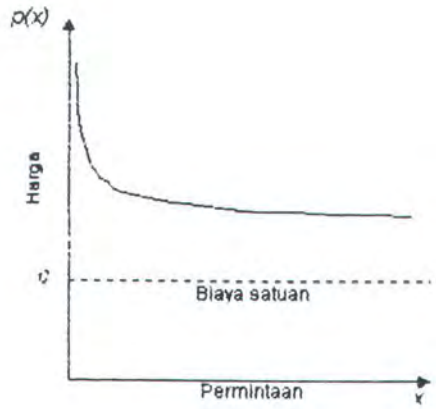
Masalah Produk Campuran dan Elastisitas Harga

Dalam beberapa kasus diketahui bahwa ada keuntungan tetap yang berhubungan dengan setiap jenis produk, sehingga fungsi tujuan yang diperoleh akan berbentuk linier. Namun, dalam banyak masalah produk campuran, ada beberapa faktor yang menyebabkan *ketaklinieran* dalam fungsi tujuannya. Sebagai contoh, dalam suatu perusahaan besar

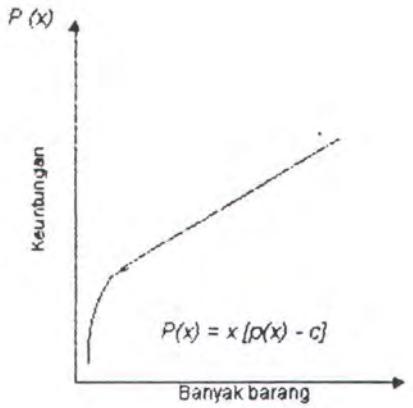
kemungkinan menghadapi *elastisitas harga*, di mana banyaknya barang yang dapat dijual berbanding terbalik dengan harganya. Jadi kurva harga-permintaan akan terlihat seperti kurva dalam gambar 2-1, di mana $p(x)$ adalah harga yang ditetapkan agar terjual x satuan barang. Jika biaya satuan untuk memproduksi barang tersebut adalah konstan yaitu di c (lihat garis terputus-putus pada gambar 2-1), maka keuntungan perusahaan tersebut dalam memproduksi dan menjual x satuan barang akan dinyatakan oleh fungsi tak linier berikut,

$$P(x) = x\,p(x) - cx,$$

seperti yang diperlihatkan dalam gambar 2-2.



Gambar 2-1 Kurva Harga-Permintaan.



Gambar 2-2 Fungsi Keuntungan.

Misalkan bila setiap produk dari n jenis produknya mempunyai fungsi keuntungan yang serupa, sebutlah $P_j(x_j)$ untuk produksi dan penjualan x_j satuan dari produk j ($j = 1, 2, \ldots, n$), maka secara lengkap fungsi tujuannya adalah

$$f(x) = \sum_{j=1}^{n} P_j(x_j)$$

yaitu penjumlahan dari beberapa fungsi tak linier.

Alasan lain yang menyebabkan ketaklinieran muncul pada fungsi tujuan, disebabkan oleh kenyataan bahwa *biaya marjinal* untuk memproduksi satu satuan barang bergantung pada tingkat produksi. Sebagai contoh, biaya marjinal akan turun apabila tingkat produksi

naik, sebagai akibat dari efek dari *kurva belajar* (*learning curve*) (produksi semakin efisien karena lebih banyak pengalaman). Di lain pihak, biaya marjinal dapat saja naik karena ukuran tertentu, seperti fasilitas lembur atau harga barang mahal, mungkin perlu untuk menaikkan produksi.

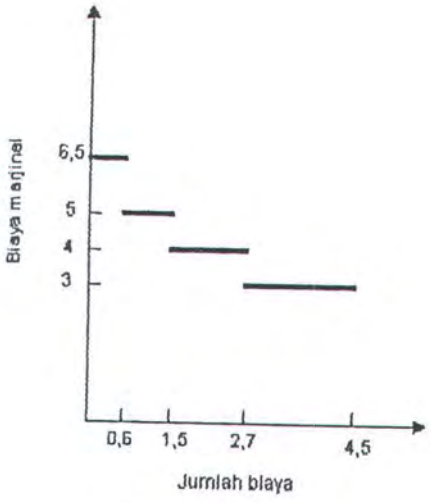
Sifat ketaklinieran dapat juga muncul pada fungsi kendala g_i (**x**) dengan cara yang sama. Sebagai contoh, bila terdapat kendala anggaran dalam biaya produksi total, maka fungsi biaya akan menjadi tak linier bila biaya produksi marjinal berubah seperti yang dijelaskan terdahulu. Kendala g_i (**x**) akan berbentuk tak linier apabila terdapat penggunaan yang tidak sebanding antara sumber daya dengan tingkat produksi dari masing-masing produk.

Masalah Transportasi dan Pengurangan Volume dalam Biaya Pengiriman

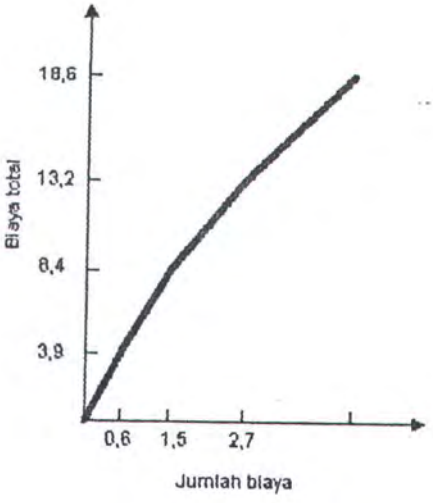
Salah satu contoh jenis pemakaian masalah transportasi misalnya menentukan perencanaan yang optimal untuk pengiriman barang dari tempat awal ke berbagai tempat tujuan, bila diberikan beberapa kendala pada penawaran dan permintaan untuk meminimumkan biaya total pengiriman. Biaya pengiriman tidak tetap, kadang-kadang *pengurangan volume* disediakan untuk pengiriman besar, sehingga biaya marjinal dari pengiriman satu atau lebih barang dapat mengikuti pola seperti Gambar 2-3. Biaya total pengiriman x satuan barang disajikan oleh fungsi *tak linier* C (x) yang berbentuk *fungsi linier bagian demi bagian,* di mana kemiringannya sama dengan besarnya biaya marjinal, seperti yang disajikan dalam Gambar 2-4. Akibatnya setiap kombinasi tempat asal dan tempat tujuan mempunyai fungsi biaya pengiriman yang serupa, maka biaya pengiriman x_{ij} satuan dari tempat asal i (i = 1, 2, ..., m) ke tempat tujuan j (j = 1, 2, ..., n) berbentuk suatu fungsi tak linier $C_{ij}(x_{ij})$ sehingga fungsi tujuan yang *diminimumkan* adalah

$$f(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}(x_{ij})$$

Bahkan dengan fungsi tujuan tak linier tersebut di atas, kendala yang ada umumnya tetap berupa kendala linier.



Gambar 2-3 Biaya marjinal



Gambar 2-4 Fungsi biaya

2.2 Grafik untuk Optimasi Tak Linier

Jika masalah tak linier hanya terdiri dari satu atau dua peubah, maka dapat kita sajikan dalam suatu grafik. Dengan grafik dapat kita perhatikan secara langsung penyelesaian yang optimal dari pemrograman linier maupun tak linier, seperti yang akan kita bahas dalam beberapa contoh berikut. Yang perlu diperhatikan dengan cara ini adalah perbedaan antara yang linier dengan yang tak linier pada beberapa jenis masalah tak liniernya.

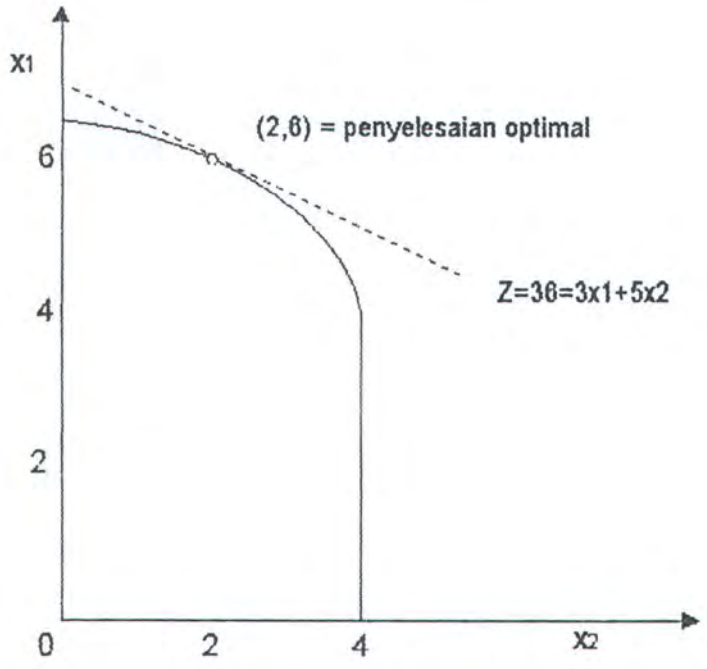
Gambar 2-5 menggambarkan masalah untuk memaksimumkan

$$Z = 3x_1 + x_2$$

dengan kendala $x_1 \leq 4$

$9x_1^2 + 5x_2^2 \leq 216$

dan $x_1 \geq 0, \quad x_2 \geq 0$



Gambar 2-5
Grafik batas tak linier

Penyelesaian optimal yang diperoleh adalah $(x_1, x_2) = (2, 6)$ dan titik ini terletak dalam batas daerah yang mungkin (daerah layak) padahal titik tersebut bukan merupakan titik pojok pada grafiknya (sebagai penyelesaian yang mungkin). Penyelesaian optimal dapat saja merupakan titik pojok pada grafik (sebagai penyelesaian yang mungkin) terhadap fungsi tujuan yang lain (contohnya , $Z = 3x_1 + x_2$). Di sini kita tidak perlu melakukan langkah matematis yang sederhana tapi rumit, dan dengan pendekatan pemrograman linier untuk menyelidiki penyelesaian optimalnya, tetapi cukup menyelidiki titik pojok-titik pojok yang menghasilkan penyelesaian yang mungkin.

Sekarang misalkan kendala kedua (tak linier) kita ganti dengan kendala linier yaitu

$$2x_2 \leq 12 \text{ dan}$$

$$3x_1 + 2x_2 \leq 18$$

dengan fungsi tujuan yang kita buat tidak linier. Sebagai contoh, misalkan

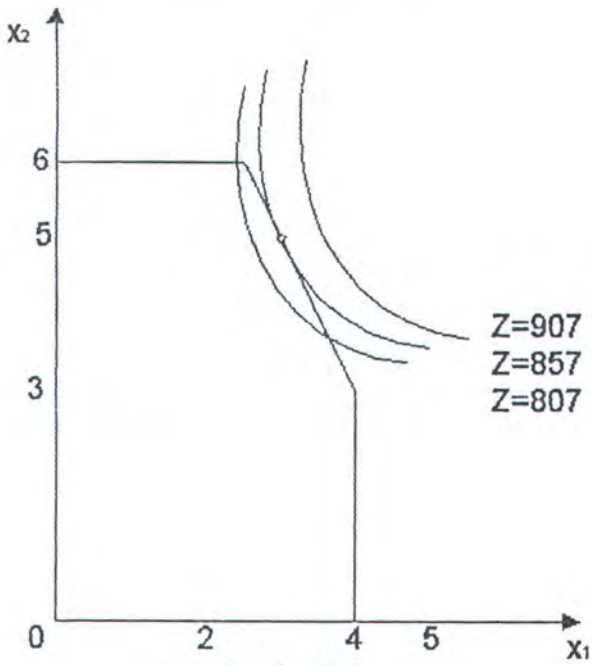
$$Z = 126x_1 - 9x_1^2 + 182x_2 - 13x_2^2,$$

maka kurva yang menggambarkan hal ini dapat dilihat pada Gambar 2-6 di mana penyelesaian optimalnya adalah $x_1 = 8/3$, $x_2 = 5$ dan terletak dalam batas daerah yang mungkin. (Nilai Z untuk penyelesaian optimal ini adalah Z = 857, dan berdasarkan Gambar 2-6 diperoleh bahwa lokasi semua titik yang menghasilkan Z = 857 akan berpotongan dengan daerah layak di satu titik tertentu, sedangkan lokasi semua titik yang menghasilkan Z lebih besar dari 857 tidak akan berpotongan dengan daerah layak.) Pada masalah tak linier yang lain, jika

$$Z = 54x_1 - 9x_1^2 + 78x_2 - 13x_2^2,$$

maka penyelesaian optimalnya adalah $(x_1, x_2) = (3 , 3)$ dan titik ini terletak di dalam daerah layak. Selanjutnya pada masalah yang cukup sederhana sebaiknya diperhatikan pula beberapa titik dalam daerah layak, tidak hanya di batasnya saja.

Masalah yang timbul sekarang dalam pemrograman tak linier yaitu seringkali maksimum lokal tidak diperlukan untuk menentukan maksimum globalnya (sebagai penyelesaian optimal), sebagai contoh kita tinjau fungsi dengan satu peubah yang ditunjukkan dalam Gambar 2-7. Pada selang $0 \leq x \leq 5$, fungsi tersebut mempunyai tiga maksimum lokal yaitu $x = 0$, $x = 2$ dan $x = 4$, tetapi hanya $x = 4$ yang merupakan maksimum global (di lain pihak, minimum lokal tercapai di $x = 1$, 3 dan 5 tetapi hanya $x = 5$ yang merupakan minimum global.)

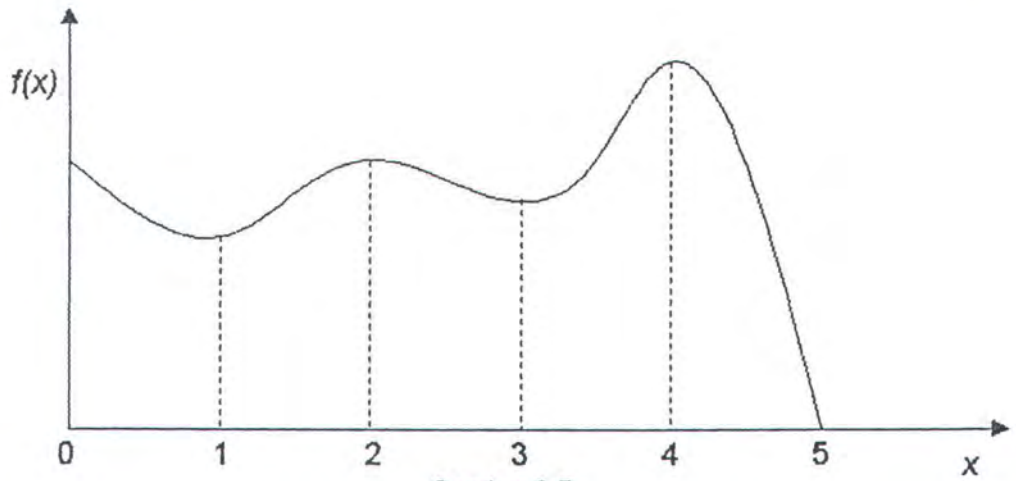


Gambar 2-6
Fungsi Obyektif Tak Linier

Pada umumnya, untuk pemrograman tak linier kita sukar membedakan antara maksimum lokal dan maksimum global (kecuali bila dapat membandingkan titik-titik maksimum lokal yang ditemukan). Selanjutnya, bahwa kondisi lokal akan menjadi jaminan untuk maksimum global dalam daerah kelayakannya. Perhatikan kembali kalkulus, terutama bila kita ingin memaksimumkan fungsi *f(x)* dengan satu peubah tanpa diberikan suatu kendala, maka kita mengingat suatu persamaan diferensial (turunan keduanya), yaitu

$$\frac{d^2}{dx^2} \leq 0 \qquad \text{untuk semua } x,$$

sehingga kurva fungsi f selalu menghadap ke bawah (kurva tidak membelok tajam). Fungsi demikian dikatakan sebagai fungsi cekung (konkav), sebaliknya apabila tanda $\leq$ di atas kita ganti dengan tanda $\geq$ maka kurva fungsi f selalu menghadap ke atas (tidak membelok tajam) dan fungsi seperti ini disebut sebagai fungsi cembung (konveks). (Fungsi linier merupakan fungsi yang cekung dan sekaligus cembung) sebagai salah satu contoh, maka perhatikanlah Gambar 2-8. Gambar 2-7 juga memperlihatkan gabungan antara cekung dan konveks tergantung bagian kurva yang mana menghadap ke atas dan yang menghadap ke bawah.

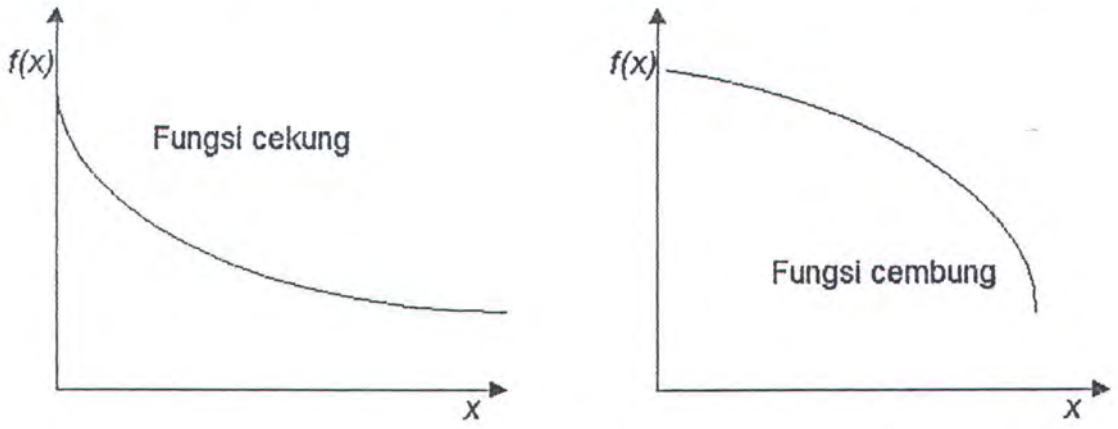


Gambar 2-7
Fungsi dengan beberapa maksimum lokal

Fungsi yang terdiri dari beberapa peubah juga mempunyai sifat cekung ataupun cembung bergantung kepada kurvanya apakah menghadap ke atas atau ke bawah. Sebagai contoh, tinjau fungsi yang memuat bentuk penjumlahan dari peubah-peubahnya. Jika pada setiap peubahnya memberi bentuk cekung (dapat dilihat pada turunan kedua yang bentuknya menjadi hanya satu peubah) maka fungsi tersebut cekung, sebaliknya merupakan fungsi cembung apabila setiap bentuknya juga cembung.

Dalam masalah pemrograman tak linier yang tidak mempunyai kendala, maka fungsi tujuan yang cekung akan menjamin bahwa maksimum lokal merupakan maksimum global (sebaliknya fungsi tujuan yang cembung akan menjamin bahwa minimum lokal adalah minimum global). Sedangkan untuk masalah pemrograman tak linier dengan kendala, maka yang menjamin keadaan di atas adalah sifat daerah layaknya, namakanlah *himpunan cembung*. Himpunan cembung A adalah himpunan titik-titik di mana setiap pasang titik di A, dapat dibuat suatu ruas garis yang menghubungkan pasangan titik tersebut sehingga ruas garis yang dibuat tadi masih di A. Jadi daerah layak untuk masalah dengan kendala linier merupakan himpunan cembung. Sebaliknya daerah layak pada Gambar 2-5 merupakan

himpunan cembung apabila semua fungsi g_i (x) [pada kendala g_i (x) $\leq b_i$] adalah fungsi cembung. Sehingga yang menjamin maksimum lokal merupakan maksimum global pada masalah pemrograman tak linier dengan kendala g_i (x) $\leq b_i$ (i = 1, 2, ... , m) dan $x \geq 0$ adalah bahwa fungsi tujuan dan semua fungsi g_i (x) haruslah merupakan fungsi cembung.



Gambar 2-8
Fungsi cekung dan cembung

2.3 Jenis-jenis Optimasi Tak Linier

Optimasi tak linier dapat dibedakan berdasarkan bentuk dan perkembangannya. Dengan bantuan metoda simpleks pada pemrograman linier, maka dikembangkan suatu algoritma untuk digunakan pada semua jenis masalah yang ada. Dalam hal ini, akan dikembangkan beberapa algoritma untuk menyelesaikan beberapa masalah pemrograman tak linier dari jenis yang sama (dan tertentu). Jenis-jenis masalah akan dibahas sejenak dan pada bagian selanjutnya akan diuraikan penyelesaian dari beberapa jenis masalah pemrograman tak linier tersebut.

Optimasi tanpa Kendala

Masalah optimasi tanpa kendala merupakan masalah optimasi yang tidak memiliki batasan-batasan sehingga untuk semua $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, fungsi tujuan yang sederhana

Memaksimumkan $f(x)$

adalah fungsi $f(x)$ dapat diturunkan, dan Syarat perlu dan cukup agar suatu penyelesaian **x** = **x*** merupakan penyelesaian optimal

$$\frac{\partial f}{\partial x_j} = 0 \qquad \text{di } \mathbf{x} = \mathbf{x}^*, \text{ untuk j} = 1, 2, \ldots, \text{n.}$$

di mana $f(x)$ fungsi cekung. Menentukan penyelesaian **x*** dilakukan dengan mengubah bentuk di atas menjadi suatu sistem n persamaan di mana turunan-turunan parsial-nya sama dengan nol. Hal ini sukar digunakan pada fungsi *tak linier* $f(x)$ karena biasanya beberapa fungsi *tak linier* sulit diselesaikan secara analitis untuk menentukan penyelesaian simultannya.

Lalu bagaimana? Subbab 2.4 akan menerangkan bagaimana langkah-langkah untuk menentukan **x***, namakanlah *algoritma pencarian langkah*, yaitu melakukan yang pertama untuk n = 1 kemudian untuk n > 1. Karena pentingnya hal tersebut, terutama pada beberapa jenis masalah yang memiliki kendala, maka mulailah disusun algoritma mengenai ini. Tetapi karena kurangnya algoritma yang dirancang untuk masalah yang memiliki kendala, maka selama melakukan perhitungan kita fokuskan pembicaraan pada jenis masalah yang tidak memiliki kendala.

Misalkan peubah kendala x_j dibuat tak negatif $x_j > 0$, syarat perlu dan cukup yang dibahas sebelumnya kita ganti menjadi

$$\frac{\partial f}{\partial x_j} \leq 0 \qquad \text{di } x = x^*, \qquad \text{jika } x_j^* = 0$$

$$= 0 \qquad \text{di } x = x^*, \qquad \text{jika } x_j^* > 0$$

untuk setiap j. Maka masalah tersebut memiliki beberapa batasan tak negatif tetapi bukan suatu fungsi kendala atau kasus khusus ($m = 0$) dari salah satu jenis masalah selanjutnya.

Optimasi dengan Kendala Linier

Ciri masalah dengan kendala linier diperlihatkan pada kendalanya yang mirip dengan pemrograman linier di mana semua fungsi kendala $g_i(x)$ linier tetapi fungsi tujuannya *tak linier*. Masalah tak linier yang ditinjau akan kita dekati dengan suatu pemrograman linier pada daerah layaknya. Telah dirancang sejumlah algoritma yang merupakan pengembangan dari metoda simpleks terhadap fungsi tujuan yang tak linier. Salah satu kasus khusus yang dibahas berikut adalah *pemrograman kuadratik*.

Pemrograman Kuadratik

Dalam masalah kuadratik, maka kendalanya berbentuk linier sedangkan fungsi tujuannya berbentuk *kuadrat*. Jadi perbedaannya dengan masalah pemrograman linier terletak pada fungsi tujuannya yang bisa berbentuk akar dari peubahnya atau perkalian dari dua peubah.

Dengan asumsi $f(x)$ cekung, maka beberapa algoritma dikembangkan untuk beberapa masalah. Dalam aplikasinya, penggunaan pemrograman kuadratik agak sulit dalam perumusannya, sebagai contoh dari kasus ini yakni pada perumusan masalah pemilihan pegawai dengan resiko keamanan yang telah diuraikan dalam Subbab 2.1. Dengan pengetahuan yang cukup mengenai masalah optimasi dengan kendala linier, maka dapat ditentukan beberapa pendekatan dari penyelesaian pemrograman kuadratik.

Pemrograman Cembung

Pemrograman cembung meliputi masalah yang merupakan kasus khusus dari semua jenis sebelumnya di mana $f(x)$ adalah cekung. Asumsi yang dipergunakan adalah

1. $f(x)$ cekung,
2. Setiap fungsi $g_i(x)$ cembung.

Seperti yang telah dibicarakan dalam Subbab 2.2 sebelumnya bahwa asumsi di atas akan menjamin maksimum lokal merupakan maksimum global.

Pemrograman Tak Cembung

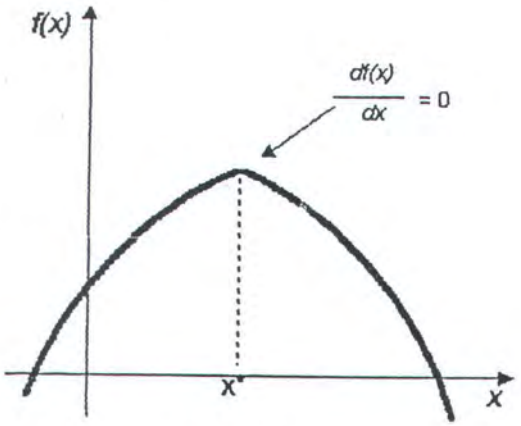
Pemrograman tak cembung meliputi semua masalah pemrograman tak linier yang tidak memenuhi asumsi-asumsi pada pemrograman cembung. Kemudian, maksimum lokal yang diperoleh tidak menjamin sebagai maksimum lokal. Di sini tidak ada algoritma yang menjamin dalam penentuan penyelesaian optimal suatu masalah, tetapi jika bentuk fungsi tak liniernya memenuhi asumsi-asumsi pada pemrograman cembung, maka terdapat beberapa algoritma yang cukup baik untuk menentukan maksimum lokal.

2.4 Optimasi Tanpa Kendala dengan Satu Peubah

Sekarang , kita mulai pembahasan tentang bagaimana menyelesaikan masalah pada suatu jenis yang sama, tetapi sebelumnya akan ditinjau kasus yang sederhana yaitu optimasi tanpa kendala dengan hanya satu peubah x ($n = 1$) di mana fungsi yang akan dimaksimumkan, yaitu $f(x)$, dapat diturunkan dan merupakan fungsi cekung. Jadi syarat perlu dan cukup agar penyelesaian $x = x^*$ menjadi optimal (maksimum global) adalah

$$\frac{df}{dx} = 0 \qquad \text{di } x = x^*,$$

seperti yang diperlihatkan dalam Gambar 2-9 persamaan ini dapat memberikan petunjuk untuk menentukan x^*. Selanjutnya, jika $f(x)$ bukan merupakan fungsi yang sederhana sehingga turunannya tidak linier maupun kuadratik, maka sulit menyelesaikan persamaan di atas secara analitis sehingga perlu digunakan suatu prosedur yang disebut prosedur pencarian dimensi satu yang memuat cara-cara penyelesaian masalah secara numerik.



Gambar 2-9
Masalah pemrograman tanpa kendala satu perubah apabila fungsinya cekung

tercapai apabila turunan di penyelesaian ini praktis sama dengan nol. Saat ini terdapat beberapa petunjuk yang dapat dipergunakan untuk masalah di atas, yang berhubungan dengan perubahah laju proporsional di masing-masing peubahnya. Hasil yang diinginkan tercapai apabila turunan parsialnya di titik tersebut sama dengan nol. Selanjutnya untuk memperluas prosedur pencarian dimensi satu, maka diperlukan nilai turunan parsialnya untuk memilih keadaan yang timbul pada langkah-langkahnya.

BAB III

OPTIMASI TAK LINIER DUA DIMENSI

BAB III

OPTIMASI TAK LINIER

Seperti yang telah dijelaskan pada bab sebelumnya, maka optimasi tak linier dua dimensi adalah pencarian dan penentuan nilai maksimum dan nilai minimum dari suatu persamaan tak linier dimana peubahnya lebih dari satu.

Berikut ini akan diterangkan beberapa cara penyelesaian optimasi tak linier satu dimensi dan dua dimensi.

3.1 Metode Fibonacci

Metode Fibonacci merupakan salah satu jenis optimasi tak linier tanpa kendala untuk satu dimensi .

Pada metode ini terlebih dahulu kita harus menentukan dua bilangan sebagai interval di mana nilai minimum yang akan kita cari berada diantara interval tersebut.

Setiap iterasi memerlukan deret fibonacci, dimana deret fibonacci tersebut ditentukan sebagai berikut :

$$F_{v+1} = F_v + F_{v-1}$$

untuk F_0 dan $F_1 = 1$. Dimana $v = 1,2,...$

Deret tersebut adalah : 1,1,2,3,5,8,13,21,34,55,89,....

Pada iterasi pertama akan dicari dua buah nilai yaitu λ dan μ di mana :

nilai bawah $\lambda_k = a_k + \frac{F_{n-k-1}}{F_{n-k+1}} (b_k - a_k)$ $\quad k = 1,.....,n-1.$

nilai atas $\mu_k = a_k + \frac{F_{n-k}}{F_{n-k+1}} (b_k - a_k)$ $\quad k = 1,.....,n-1.$

Di mana a_k = batas bawah

b_k = batas atas

n = jumlah evaluasi fungsi.

Untuk iterasi selanjutnya kita cari harga dari $F(\lambda_k)$ dan $F(\mu_k)$, hasil keduanya kita bandingkan.

Jika $F(\lambda_k) > F(\mu_k)$ maka interval yang baru $[\lambda_k, b_k]$

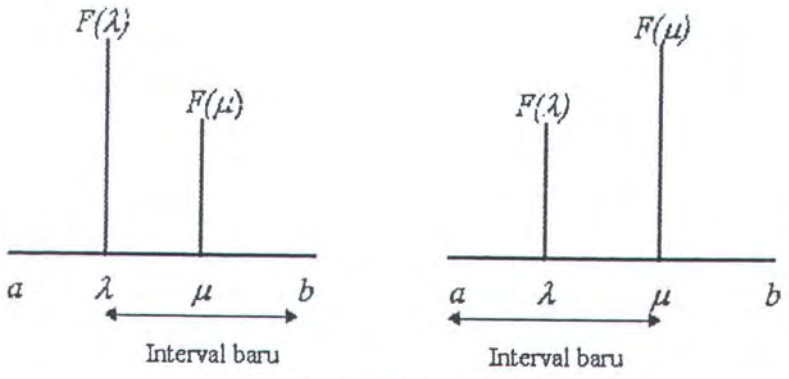
Jika $F(\lambda_k) \leq F(\mu_k)$ maka interval yang baru $[a_k, \mu_k]$,

Hal ini berdasarkan suatu teorema yang berbunyi :

- Jika interval yang membatasi adalah a dan b, sedangkan $\lambda, \mu \in [a, b]$ dan $\lambda < \mu$, maka :

  jika $F(\lambda) > F(\mu)$ maka $F(z) \geq F(\mu)$ untuk semua $z \in [a, \lambda]$

  jika $F(\lambda) \leq F(\mu)$ maka $F(z) \geq F(\mu)$ untuk semua $z \in [\mu, b]$



Gambar 3-1
Reduksi Interval

Interval yang baru $[a_{k+1}, b_{k+1}]$ adalah $[\lambda_k, b_k]$, jika $F(\lambda_k) > F(\mu_k)$ dan $[a_k, \mu_k]$, jika $F(\lambda_k) \leq F(\mu_k)$.

$$b_{k+1} - a_{k+1} = b_k - \lambda_k = b_k - [a_k + \frac{F_{n-k-1}}{F_{n-k+1}} (b_k - a_k)] = \frac{F_{n-k}}{F_{n-k+1}} (b_k - a_k)$$

atau

$$b_{k+1} - a_{k+1} = \mu_k - a_k = \frac{F_{n-k}}{F_{n-k+1}} (b_k - a_k)$$

Pada iterasi $k+1$, $\lambda_{k+1} = \mu_k$ atau $\mu_{k+1} = \lambda_k$, hal ini bisa kita turunkan

jika $F(\lambda_k) > F(\mu_k)$ maka

$$\lambda_{k+1} = a_{k+1} + (F_{n-k-2} / F_{n-k})(b_{k+1} - a_{k+1})$$

$$= \lambda_k + (F_{n-k-2} / F_{n-k})(b_k - \lambda_k).$$

dengan mensubtitusi λ_k, maka diperoleh :

$$\lambda_{k+1} = [a_k + \frac{F_{n-k-1}}{F_{n-k+1}}(b_k - a_k)] + \frac{F_{n-k-2}}{F_{n-k}}\left(\frac{1 - F_{n-k-1}}{F_{n-k+1}}\right)(b_k - a_k)$$

dengan mensubtitusi $[1 - F_{n-k-1} / F_{n-k+1}] = F_{n-k} / F_{n-k+1}$

yaitu dari $[1 - F_{n-k-1} / F_{n-k+1}] = [F_{n-k+1} - F_{n-k-1}] / F_{n-k+1} = [F_{n-k} / F_{n-k+1}]$

dengan contoh : 1, 1, 2, 3, 5

$\frac{5-2}{\delta} = \frac{3}{\delta}$ di mana 3 di antara kedua bilangan tersebut

persamaan baru yang kita peroleh sekarang adalah

$$\lambda_{k+1} = a_k + (F_{n-k-1} / F_{n-k+1})(b_k - a_k)$$

$$+ [(F_{n-k-2} / F_{n-k}) . (F_{n-k} / F_{n-k+1})](b_k - a_k)$$

$$= a_k + [(F_{n-k-1} + F_{n-k-2}) / F_{n-k+1}](b_k - a_k)$$

dengan mensubstitusi $[(F_{n-k-1} + F_{n-k-2}) / F_{n-k+1}] = (F_{n-k} / F_{n-k+1})$

maka akan diperoleh

$$\lambda_{k+1} = a_k + (F_{n-k} / F_{n-k+1}) (b_k - a_k).$$

$$\lambda_{k+1} = \mu_k .$$

Demikian juga bila $F(\lambda_k) \leq F(\mu_k)$, maka akan diperoleh

$$\mu_{k+1} = \lambda_k ..$$

Jadi pada iterasi pertama kita perlu mencari nilai μ *dan* λ. Sedangkan untuk iterasi berikutnya cukup mencari μ *atau* λ saja.

Untuk iterasi k = n-1

$$\mu_{n-1} = \lambda_{n-1} = 1/2 (a_{n-1} + b_{n-1})$$

yang bisa diturunkan dari

$$\lambda_{k+1} = a_k + (F_{n-k-1} / F_{n-k+1}) (b_k - a_k).$$

$$\lambda_{n-1} = a_{n-1} + (F_{n-(n-1)-1} / F_{n-(n-1)+1}) (b_{n-1} - a_{n-1}).$$

$$= a_{n-1} + (F_0 / F_2) (b_{n-1} - a_{n-1}).$$

$$= a_{n-1} + 1/2 (b_{n-1} - a_{n-1}).$$

$$\lambda_{n-1} = 1/2 (b_{n-1} + a_{n-1})$$ terbukti.

Menentukan jumlah iterasi.

Dalam metode ini harus menentukan terlebih dahulu nilai "n", dimana nilai tersebut merupakan deret Fibonacci yang ke-n serta menentukan jumlah iterasi yang diperlukan untuk memperoleh nilai minimum.

Algoritma Fibonacci.

Langkah pendahuluan :

Menentukan interval awal yaitu a_1 dan b_1.

Memilih jumlah iterasi 'n'.

Menentukan λ_1 dan μ_1, dan k =1,ε = 0.01.

Kemudian hitung F (λ_1) dan F (μ_1).

Lanjutkan ke langkah iterasi.

Langkah iterasi adalah sebagai berikut :

1. Jika $F(\lambda_k) > F(\mu_k)$, lanjutkan ke langkah 2, jika $F(\lambda_k) \leq F(\mu_k)$ lanjutkan ke langkah 3.
2. $a_{k+1} = \lambda_k$ dan $b_{k+1} = b_k$.

   Selanjutnya, $\lambda_{k+1} = \mu_k$ dan $\mu_{k+1} = a_{k+1} + (F_{n-k-1} / F_{n-k})(b_{k+1} - a_{k+1})$. Jika $k = n-2$, lanjutkan ke langkah 5, jika tidak hitung $F(\mu_{k+1})$ dan lanjutkan ke langkah 4.
3. $a_{k+1} = a_k$ dan $b_{k+1} = \mu_k$. Kemudian $\mu_{k+1} = \lambda_k$, dan $\lambda_{k+1} = a_{k+1} + (F_{n-k-2} / F_{n-k})(b_{k+1} - a_{k+1})$ Jika $k = n - 2$, lanjutkan ke langkah 5, jika tidak tentukan $F(\lambda_{k+1})$ dan lanjutkan ke langkah 4.
4. Tambahkan $k = k+1$ dan kembali ke langkah 1.
5. $\lambda_n = \lambda_{n-1}$ dan $\mu_n = \lambda_{n-1} + \varepsilon$. Jika $F(\lambda_n) > F(\mu_n)$, maka $a_n = \lambda_n$ dan $b_n = b_{n-1}$. Jika tidak $F(\lambda_n) \leq F(\mu_n)$, maka $a_n = a_{n-1}$ dan $b_n = \lambda_n$. Interval terakhir diperoleh yaitu $[a_n, b_n]$.

3.2 Metode Newton

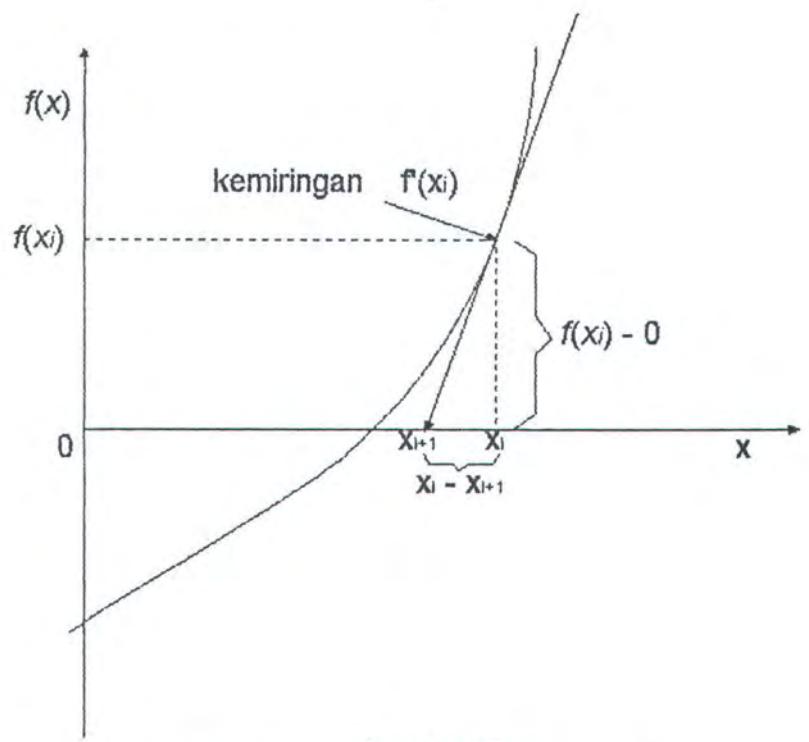
Seperti metode Fibonacci, metode Newton juga merupakan salah satu jenis optimasi tak linier tanpa kendala untuk satu dimensi.

Pada metode ini pertama kita menentukan nilai awal (x_0), selanjutnya dari nilai awal ini, sebuah garis singgung dapat diperluas dari titik $(x_0, f(x_0))$.

Titik dimana garis singgung memotong garis sumbu x biasanya menunjukkan sebuah taksiran perbaikan dari harga x.

Metode Newton dapat diturunkan berdasarkan interpretasi geometri k atau sebuah metode alternatif yang didasarkan pada deret Taylor.

Pada gambar dibawah ini



Gambar 3-2
Garis singgung terhadap fungsi pada x_1

turunan pertama pada x_i adalah ekivalen terhadap kemiringan (slope)

$$f'(x_i) = \frac{f(x_i) - 0}{x_i - x_{i+1}}$$

yang dapat diatur kembali menjadi:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$ yang disebut formula Newton

Deret Taylor sendiri dinyatakan sebagai berikut :

$$f(x_{i+1}) = f(x_i) + f'(x_i)(x_{i+1} - x_i) + f''(\xi)\frac{1}{2}(x_{i+1} - x_i)^2$$

dimana ξ terletak disembarang tempat dalam interval x_i hingga x_{i+1}.

Salah satu cara pendekatan diperoleh dengan memotong deret Taylor setelah suku turunan pertama, $f(x_{i+1}) \approx f(x_i) + f'(x_i)(x_{i+1} - x_i)$

pada perpotongan dengan sumbu x , $f(x_{i+1})$ akan sama dengan nol atau

$$0 \approx f(x_i) + f'(x_i)(x_{i+1} - x_i)$$

dapat diselesaikan untuk

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

$$f'(x_i)(x_{i+1} - x_i) = f(x_i)$$

jadi serupa dengan persamaan diatas.

Algoritma Newton

Langkah pendahuluan :

Ambil titik sembarang x_0 , $\varepsilon > 0$, $i = 0$.

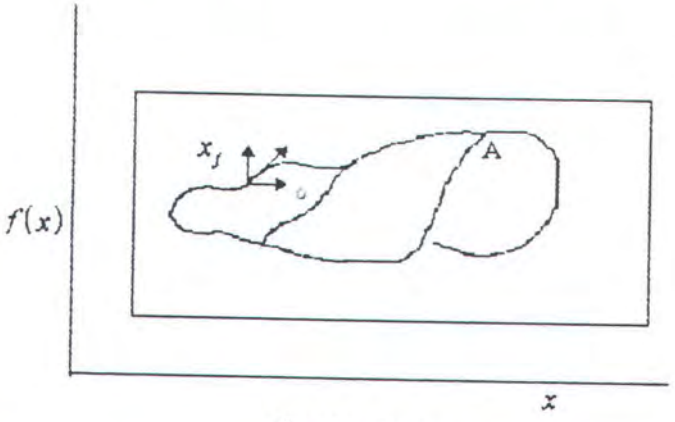
Langkah iterasi :

1. hitung $f'(x_i)$ dan $f(x_i)$.
2. hitung x_{i+1}.
3. bila $|f(x_i)| \le \varepsilon$ atau $|(x_{i+1} - x_i)| \le \varepsilon$ stop, jika tidak $i = i + 1$ kembali ke 1.

3.3. Metode Gradien .

Metode gradien merupakan salah satu jenis optimasi tak linier tanpa kendala untuk dua dimensi.

Metode ini paling sering dipakai dibandingkan metode tak linier yang lain. Dalam penggunaan metode gradien kita perhatikan suatu titik x_j pada gambar dibawah ini :



Gambar 3-3.
Fungsi tak linier berdimensi banyak.

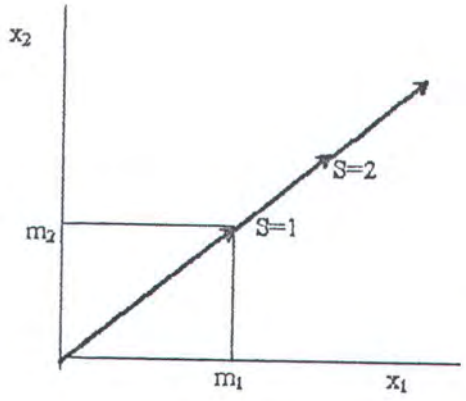
Misalkan kita berada pada titik x_j dan kita mencoba untuk mendapatkan nilai maksimum $f(x)$ pada titik A.

Kemudian kita hendak meneruskan dari titik x_j ke titik x_{j+1} sedemikian rupa sehingga mendekati titik optimum dengan kemungkinan yang tercepat, x_{j+1} akan diperoleh dengan berpindah sejauh satuan s menuju solusi optimum.

Dalam bentuk komponen ruang atau vektor dimensi sebagai berikut :

$$x_{j+1}^{(i)} = x_j^{(i)} + s.m_i$$

dimana m_i adalah arah perpindahan untuk komponen ke-i (lihat gambar berikut) :



Gambar 3-4
Perpindahan dua dimensi

Misalkan kita hendak mengambil langkah yang kecil sedemikian rupa sehingga fungsi yang dituju $y = f(x)$ bertambah atau berkurang sebanyak mungkin.

Jarak perpindahan diberikan sebagai berikut :

$$ds = \sqrt{dx_1^{\,2} + dx_2^{\,2} + \ldots\ldots + dx_n^{\,2}}$$

dengan asumsi bahwa y dapat diturunkan, maka perubahan y sehubungan dengan serangkaian pergeseran dx_i ditentukan oleh :

$$dy = \sum_{i=1}^{n} \left(\frac{\partial y}{\partial x_i} \right) dx_i$$

atau

$$\frac{dy}{ds} = \sum_{i=1}^{n} \left(\frac{\partial y}{\partial x_i} \right) \cdot \frac{dx_i}{ds}$$

sekarang serangkaian pergeseran tertentu akan membuat $\frac{dy}{ds}$ sebesar atau sekecil mungkin, hal ini adalah arah naik atau turun yang paling curam.

Dengan menganggapnya sebagai persoalan optimasi, maka kita akan memaksimumkan atau meminimumkan persamaan diatas menurut ds

maksimumkan / minimumkan : $$\frac{dy}{ds} = \sum_{i=1}^{n} \left(\frac{\partial y}{\partial x_i} \right) \frac{dx_i}{ds}.$$

menurut : $$ds = \sqrt{\sum_{i=1}^{n} dx_i^{\,2}}.$$

Sehingga membentuk fungsi lagrangian

maksimumkan / minimumkan : $$\sum_{i=1}^{n} \left(\frac{\partial y}{\partial x_i} \right) \frac{dx_i}{ds} - \lambda \left[1 - \sum_{i=1}^{n} \left(\frac{dx_i}{ds} \right)^2 \right].$$

diturunkan terhadap $\frac{dx_i}{ds}$

$$\sum_{i=1}^{n} \left(\frac{\partial y}{\partial x_i} \right) \frac{dx_i}{ds} - \lambda + \lambda \sum_{i=1}^{n} \left(\frac{dx_i}{ds} \right)^2$$

$$\frac{\partial y}{\partial x_i} - 2\lambda . \left(\frac{dx_i}{ds} \right) = 0 \qquad i = 1,2,\ldots\ldots,n$$

$$\frac{\partial y}{\partial x_i} = 2\lambda . \left(\frac{dx_i}{ds} \right).$$

$$\frac{1}{2\lambda} \frac{\partial y}{\partial x_i} = \frac{dx_i}{ds}.$$

dan diturunkan terhadap pengali lagrangian λ

$$\sum_{i=1}^{n} \left(\frac{dx_i}{ds} \right)^2 = 1.$$

maka

$$\frac{1}{4\lambda^2}\sum_{i=1}^{n}\left(\frac{\partial y}{\partial x_i}\right)^2 = 1$$

$$\left(\frac{\partial y}{\partial x_i}\right)^2 = 4\lambda^2$$

$$2\lambda = \pm\sqrt{\sum_{i=1}^{n}\left(\frac{\partial y}{\partial x_i}\right)^2}$$

Sebagaimana telah dibahas sebelumnya, kita berusaha untuk mengubah titik x_j ketitik x_{j+1} secara optimal.

Perubahan pada komponen ke - i ditentukan oleh :

$$\frac{\partial x_i}{ds} = \frac{\partial y}{\partial x_i}\cdot\frac{1}{2\lambda} \qquad i = 1,2,\ldots\ldots,n$$

dalam bentuk parameter untuk komponen ke - i

$$x_{j+1}^{(i)} = x_j^{(i)} + \left[\frac{\partial y}{\partial x_i}\cdot\frac{1}{2\lambda}\right]s = x_j^{(i)} + m_i s$$

maka dengan menggunakan persamaan diatas perpindahan dimana perubahan y adalah terbesar diberikan oleh :

$$m_i = \frac{\dfrac{\partial y}{\partial x_i}}{\sqrt{\sum_{i=1}^{n}\left(\dfrac{\partial y}{\partial x_i}\right)^2}} \qquad i = 1,2,\ldots\ldots\ldots,n$$

dan penurunan yang terbesar diberikan oleh :

$$m_i = \frac{-\dfrac{\partial y}{\partial x_i}}{\sqrt{\sum_{i=1}^{n}\left(\dfrac{\partial y}{\partial x_i}\right)^2}} \qquad i = 1,2,\ldots\ldots,n$$

dengan catatan bahwa pembilangnya adalah gradien, sedangkan penyebutnya adalah faktor penormal.

Algoritma Gradien

Diketahui suatu nilai x, algoritma gradien bergerak sepanjang garis :

$$\frac{-\nabla f(x)}{\|\nabla f(x)\|}$$

atau dengan arah $-\nabla f(x)$.

Langkah pendahuluan : tentukan $\varepsilon > 0$ dan nilai awal x_1 , $k = 1$.

Langkah iterasi :

1. jika $\|\nabla f(x_k)\| < \varepsilon$ berhenti, jika tidak $d_k = -\dfrac{\nabla f(x_k)}{\|\nabla f(x_k)\|}$ dan λ_k adalah solusi optimal untuk meminimumkan harga dari $f(x_k + \lambda \quad d_k)$ dengan batasan $\lambda \geq 0$, selanjutnya $x_{k+1} = x_k + \lambda_k d_k$, $k = k+1$ dan ulangi langkah iterasi.

3.4. Metode *Conjugate*

Dalam subbab ini akan membahas metode *conjugate* yang berdasar pada penggunaan turunan dan evaluasi fungsi. Metode *conjugate* banyak dipakai pada optimasi tak linier tanpa kendala terutama pada fungsi-fungsi kwadrat. Hal ini disebabkan pencarian titik minimum sepanjang arah *conjugate* dapat diperoleh dalam n tahapan. Sebagaimana metode gradien metode ini digunakan pada fungsi-fungsi dua dimensi

Definisi

Anggap H sebagai matrik simetri n x n. Vektor $d_1,\ldots,d_k$ disebut sebagai H *conjugate* atau *conjugate* saja jika bebas linier dan jika $d_i H d_j = 0$ untuk $i \neq j$.

Contoh berikut memberikan gambaran hal-hal penting dari optimasi sepanjang arah *conjugate* untuk fungsi kwadrat.

Contoh.

Perhatikan contoh berikut :

Minimumkan $-12y + 4x^2 + 4y^2 - 4yx$.

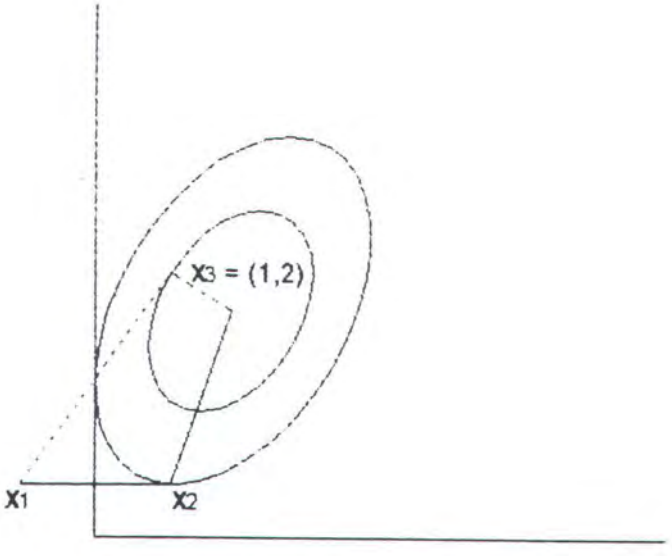
Catat bahwa matrik Hessian adalah sebagai berikut :

$$H = \begin{bmatrix} 8 & -4 \\ -4 & 8 \end{bmatrix}$$

Kita buat dua arah *conjugate* $d_1 \& d_2$. Seandainya kita memilih $d_1^{'} = (1;0)$ dan $d_2^{'} = (a;b)$, maka akan menjadi $0 = d_1^{'} H d_2^{'} = 8a - 4b$. Kemudian kita menetapkan $a = 1$ dan $b = 2$, sehingga $d_2^{'} = (1;2)$. Hal ini bisa kita catat bahwa arah *conjugate* tidak *unique*.

Jika kita minimumkan fungsi F mulai dari $x_1^{'} = \left(-\frac{1}{2};1\right)$ sepanjang arah $d_1^{'}$, maka kita akan memperoleh titik $x_2^{'} = \left(\frac{1}{2};1\right)$. Selanjutnya mulai dari $x_2^{'}$ dan meminimumkan sepanjang $d_2^{'}$, akan diperoleh $x_3^{'} = (1;2)$. x_3 adalah titik minimum.

Bentuk dari fungsi yang dituju dan lintasan untuk memperoleh titik optimal ditunjukkan gambar dibawah ini. Pada gambar dapat dilihat bahwa dimulai dari sembarang titik kemudian diminimumkan sepanjang $d_1 \& d_2$ titik optimal akan diperoleh paling tidak dalam dua tahapan.



Gambar 3-5
Ilustrasi arah *Conjugate*

Metode *Fletcher* dan *Reeves*

Metode *Conjugate* gradien yang kami bahas disini adalah metode *conjugate* yang dipercayakan kepada *Fletcher* dan *Reeves* (1994). Dalam metode ini membelokkan arah turun yang paling curam dengan menambahkan suatu pengali arah positif yang digunakan pada tahap terakhir. Untuk kasus kwadrat seperti yang akan kita pelajari, pembelokan arah turun yang paling curam akan menghasilkan sekumpulan arah-arah *conjugate*.

Algoritma *conjugate*

Tahap Pendahuluan: memilih suatu skalar penghentian $\varepsilon > 0$ dan titik awal x_1. Misal

$$y_1 = x_1,\ d_1 = -\nabla f(y_1),\ k = j = 1$$ dilanjutkan ke tahap iterasi.

Tahap iterasi :.

1. Jika $\|\nabla f(y_j)\| < \varepsilon$, berhenti. Jika tidak misalnya λ_j adalah solusi optimal terhadap masalah peminimalan $f(y_j + \lambda d_j)$ menurut $\lambda \geq 0$ dan misal $y_{j+1} = y_j + \lambda_j d_j$. Jika $j < n$, lanjutkan ke tahap 2, jika tidak lanjutkan ke tahap 3.

2. Misal $d_{j+1} = -\nabla f(y_{j+1}) + \alpha_j d_j$, dimana $\alpha_j = \dfrac{\|\nabla f(y_{j+1})\|^2}{\|\nabla f(y_j)\|^2}$. Ganti j dengan j+1 dan lanjutkan ke tahap 1.

3. Misal $y_1 = x_{k+1} = y_{n+1}$ dan misal $d_1 = -\nabla f(y_1)$. Misal j=1, ganti k dengan k+1 dan

lanjutkan ke tahap 1.

Perhatikan bahwa $d_{j+1} = \left({}^{1}/_{\mu_1} \right)\left(\mu_1 \left[-\nabla f\left(y_{j+1} \right) \right] + \mu_2 d_j \right)$, dimana

$$\mu_1 = \frac{\left\| \nabla f\left(y_j \right) \right\|^2}{\left\| \nabla f\left(y_j \right) \right\|^2 + \left\| \nabla f\left(y_{j+1} \right) \right\|^2} \quad \text{dan}$$

$$\mu_2 = \frac{\left\| \nabla f\left(y_{j+1} \right) \right\|^2}{\left\| \nabla f\left(y_j \right) \right\|^2 + \left\| \nabla f\left(y_{j+1} \right) \right\|^2}.$$

sehingga d_{j+1} pada dasarnya merupakan kombinasi arah turun paling curam yang sekarang dengan arah yang digunakan pada iterasi terakhir.

3.5 Metode Penalti dan Penghalang (*barrier*)

Fungsi-fungsi penalti digunakan untuk mentransformasikan suatu masalah yang terkendala kesuatu masalah yang tak terkendala. Kendala diberikan terhadap suatu fungsi melalui parameter penalti sehingga meniadakan penyimpangan dari kendala - kendala tersebut.

Perhatikan masalah berikut dengan kendala tunggal $h(x) = 0$

Minimumkan $f(x)$

Dengan kendala $h(x) = 0$

Jika masalah ini diganti dengan masalah tanpa kendala berikut ini, dimana $h > 0$ merupakan bilangan yang besar, maka :

Minimumkan $f(x) + \mu h^2(x)$

dimana $x \in En$

solusi optimal masalah tersebut diatas harus memenuhi $h^2(x) \to 0$.

Sekarang jika kendalanya berupa pertidaksamaan tunggal $g(x) \leq 0$,

Minimumkan $f(x)$.

Dengan kendala $g(x) \leq 0$.

Jelas bahwa bentuk $f(x) + \mu\, g^2(x)$ tidak sesuai, karena penalti akan diberikan untuk $g(x) > 0$ maupun $g(x) < 0$. Jadi penalti diberikan hanya jika titik x tidak mungkin diselesaikan, yaitu bila $g(x) > 0$. Dengan demikian kendala yang tepat adalah:

Minimumkan $f(x) + \mu$ maksimum $\{0, g(x)\}$.

Dimana $x \in E_n$.

Jika $g(x) \leq 0$, maka maksimum $\{0, g(x)\} = 0$ tidak dipenalti, tetapi jika $g(x) > 0$, maka maksimum $\{0, g(x)\} > 0$ akan diperoleh suku penalti sebesar $\mu\, g(x)$.

Secara umum fungsi penalti yang tepat harus memberi penalti yang positif untuk titik-titik yang tidak mungkin dan tidak diberi penalti untuk titik-titik yang mungkin. Jika

kendala tersebut berbentuk $g_i(x) \le 0$ untuk i = 1,...,m dan $h_i(x) = 0$ untuk $i = 1,...,l$, maka fungsi penalti yang tepat didefinisian oleh :

$$\alpha(x) = \sum_{i=1}^{m} \phi\left[g_i(x)\right] + \sum_{i=1}^{l} \varphi\left[h_i(x)\right]$$

dimana ϕ dan φ adalah fungsi - fungsi kontinyu yang memenuhi :

$$\phi(y) = 0 \text{ jika } y \le 0 \quad \text{dan } \phi(y) > 0 \text{ jika } y > 0.$$

$$\varphi(y) = 0 \text{ jika } y = 0 \quad \text{dan } \varphi(y) > 0 \text{ jika } y \neq 0.$$

ϕ dan φ berbentuk :

$$\phi(y) = \left[maksimum\{0, y\}\right]^p.$$

$$\varphi(y) = |y|^p.$$

dimana p adalah integer positif sehingga fungsi penalti berbentuk

$$\alpha(x) = \sum_{i=1}^{m}\left[maksimum\{0, g_i(x)\}\right]^p + \sum_{i=1}^{l}\left|h_i(x)\right|^p$$

fungsi $f(x) + \mu\,\alpha(x)$ disebut fungsi pembantu .

Algoritma Penalti

Berikut ini akan terlihat metode fungsi penalti untuk menyelesaikan masalah peminimuman $f(x)$ menurut $g(x) \le 0$, $h(x) = 0$ dan $x \in X$.

Metode yang ada hanya membatasi bahwa f, g dan h adalah fungsi yang kontinyu. Tetapi metode tersebut hanya dapat digunakan secara efektif pada kasus-kasus dimana terdapat suatu prosedur solusi yang efisien untuk menyelesaikan masalah yang diberikan pada tahapan dibawah ini.

Tahap pendahuluan :

misalkan $\varepsilon > 0$ adalah skalar penghentian. Pilihlah suatu titik awal x_1, suatu parameter penalti $\mu_1 > 0$ dan suatu skalar $\beta > 1$. Misalkan $k = 1$ dan teruskan ke tahap itersai.

Tahap iterasi:

1. mulai dengan x_k, selesaikan masalah berikut :

   minimumkan $f(x) + \mu_k \alpha(x)$.

   menurut $x \in X$.

   misalkan x_{k+1} suatu solusi optimal, dan teruskan ke 2.

2. jika $\mu_k \alpha(x_{k+1}) < \varepsilon$ berhenti; jika tidak dimisalkan $\mu_{k+1} = \beta\mu_k$, ganti k dengan $k+1$ dan lanjutkan ke 1.

Metode Penghalang (*barrier*)

Sama halnya dengan fungsi penalti, fungsi-fungsi penghalang juga digunakan untuk mentransformasikan masalah yang terbatas menjadi masalah tanpa batas . Fungsi ini

mengatur penghalang di sekeliling daerah yang mungkin. Jika solusi optimal berada pada daerah yang mungkin, maka prosedurnya berpindah dari dalam ke batas.

Masalah utama dan penghalang dirumuskan sebagai berikut:

Masalah utama:

Minimumkan $f(x)$

dengan penghalang $g(x) \leq 0$

dan $x \in X$

dimana g suatu fungsi vektor yang komponennya g1,g2,...,gm.

f,g1,...,gm adalah fungsi-fungsi yang kontinyu pada En, dan x adalah himpunan pada En yang tidak kosong.

Perhatikan bahwa batasan persamaan tidak ada. Jika diberikan batasan h(x) =0, maka metode-metode penggunaan fungsi penghalang menghendaki interior dari himpunan $\{x: g(x) \leq 0, h(x) = 0\}$ tidak kosong, dan ini jelas tidak mungkin.

Masalah penghalang .

Minimumkan $\theta(\mu)$.

Menurut $\mu \geq 0$

dimana $\theta(\mu) = \inf\{f(x) + \mu B(x): g(x) < 0, x \in X\}$.

disini B merupakan fungsi penghalang yang tidak negatif dan kontinyu pada daerah $\{x:g(x)<0\}$ bila daerah $\{x:g(x)\le 0\}$ didekati dari interior maka B mendekati ∞.

Fungsi penghalang didefinisikan sebagai berikut :

$$B(x)=\sum_{i=1}^{m}\phi\left[g_i(x)\right]$$

dimana ϕ adalah suatu fungsi satu variabel yang kontinyu pada $\{y:y<0\}$.

dan memenuhi :

$$\phi(y)\ge 0 \text{ jika } y<0 \text{ dan } \lim_{y\to 0}\phi(y)=\infty$$

jadi fungsi penghalang yang khas berbentuk :

$$B(x)=\sum_{i=1}^{m}\frac{-1}{g_i(x)}$$

Fungsi $f(x)+\mu B(x)$ disebut fungsi pembantu. Idealnya fungsi B bernilai nol pada daerah $\{x:g(x)<0\}$ dan ∞ pada batas-batasnya. Hal ini untuk menjamin bahwa kita tidak akan meninggalkan daerah $\{x:g(x)\le 0\}$, dan menetapkan bahwa masalah peminimalan dimulai dari suatu titik dalam. Tetapi ketidakkontinyuan ini merupakan kesulitan yang serius dalam semua prosedur komputasi. Oleh karena itu konstruksi B yang ideal ini diganti dengan persyaratan yang lebih realistis, yaitu B tak negatif dan kontinyu pada daerah

$\{x:g(x)<0\}$ dan mendekati ∞ bila batasnya didekati dari interior. Perlu diingat bahwa μB mendekati fungsi penghalang ideal bila μ mendekati nol.

Untuk $\mu > 0$ pada $\theta(\mu) = \inf\{f(x) + \mu B(x): g(x) < 0, x \in X\}$ tidak lebih sederhana daripada menyelesaikan masalah yang awal karena adanya batasan $g(x) < 0$. Tetapi jika kita memulai optimasi dari titik di daerah $S = \{x:g(x)<0\} \cap X$, dan mengabaikan batasan $g(x) < 0$, maka titik optimal berada dalam S. Hal ini disebabkan struktur B.

Hasil ini diperoleh dari fakta jika kita mendekati batas $\{x:g(x) \le 0\}$ dari interior, maka B mendekati ∞ yang akan mencegah kita untuk meninggalkan set S.

Algoritma.

Tahap pendahuluan: misalkan $\varepsilon > 0$ merupakan skalar penghentian, pilih suatu titik $x \in X$ dengan $g(x) < 0$. Misalkan $\mu_1 > 0, \beta \in 0.1$, k=1 dan lanjutkan ke tahap iterasi.

Tahap iterasi :

1. Mulai dengan x_k selesaikan masalah berikut :

   Minimumkan $f(x) + \mu_k B(x)$

   menurut $g(x) < 0, x \in X$

   Misalkan x_{k+1} adalah solusi optimal dan lanjutkan ke tahap 2.

2. Jika $\mu_k B(x_{k+1}) < \varepsilon$, berhenti. Jika tidak misalkan $\mu_{k+1} = \beta\mu_k$, ganti k dengan k+1 dan ulangi tahap 1.

BAB IV

PERANCANGAN PROGRAM

BAB IV

PERANCANGAN PROGRAM

Program visualisasi optimasi ini menggunakan bahasa pemrograman Pascal dengan sistim operasi DOS. Selanjutnya untuk mengetahui bagaimana program visualisasi ini dibuat, perlu untuk melihat hal berikut.

4.1. Bagan-alir

Sebelum kita mulai membuat program, yang pertama kali dibuat adalah bagan-alir. Bagan-alir ini dirancang berdasarkan algoritma yang terdapat dalam masing-masing metode yang telah dibahas dalam bab tiga. Dalam subbab ini bagan-alir masing-masing metode tersedia dalam lampiran A.

4.2. Masukan Program

Masukan pada program visualisasi ini meliputi :

Metode Fibonacci

Dalam metode fibonacci terdapat empat masukan yang terdiri atas :

1. Masukan fungsi persamaan

   Masukan ini bertipe string yang dibatasi sebanyak 200 karakter.

2. Masukan nilai atas .

   Nilai atas ini maksudnya batas dari daerah pencarian. Disebut nilai atas karena nilainya lebih besar dibandingkan nilai yang lain. Tipe dari masukan ini adalah real karena bisa bernilai negatif maupun positif dan bisa bernilai desimal.

3. Masukan nilai bawah .

   Sama seperti nilai atas tetapi nilainya lebih kecil dibandingkan nilai atas dan bertipe real.

4. Masukan nilai n.

   Merupakan deret fibonacci yang ke n. Memiliki tipe word karena nilainya dimulai dari 0.

Metode Newton

Dalam metode newton terdapat dua masukan yang terdiri atas :

1. Masukan fungsi persamaan.

   Masukan ini bertipe string yang dibatasi sebanyak 200 karakter.

2. Masukan nilai x.

   Masukan ini merupakan nilai x awal. Tipe dari masukan ini adalah real.

Metode Gradien

Dalam metode gradien terdapat tiga masukan yang terdiri atas :

1. Masukan fungsi persamaan .

   Masukan ini memiliki tipe string yang dibatasi sebanyak 200 karakter.

2. Masukan nilai x. Masukan ini menunjukkan nilai x awal. Tipe dari masukan x ini adalah real karena bisa bernilai negatif maupun positif dan berupa pecahan desimal.

3. Masukan nilai y.

   Seperti nilai x masukan ini bertipe real juga.

Metode *Conjugate* Gradien

Dalam metode *conjugete* gradien ada tiga masukan yang terdiri atas:

1. Masukan fungsi persamaan.

   Masukan ini bertipe string yang dibatasi sebanyak 200 karakter.

2. Masukan nilai x.

2. Masukan nilai x.

   Masukan ini menunjukkan nilai x awal. Tipe dari nilai x ini adalah real karena bisa bernilai positif maupun negatif dan berupa pecahan desimal.

3. Masukan nilai y.

   Sama seperti masukan nilai x bertipe real.

Metode Penalti dan Penghalang (***barrier*****)**

Dalam metode ini terlebih dahulu ditanyakan dimensi dari fungsi yang akan kita masukkan. Baik dimensi satu atau dua akan diperlukan masukan yang terdiri atas:

1. Masukan fungsi persamaan.

   Pada metode penalti dan penghalang diperlukan dua buah fungsi persamaan. Yang pertama sebagai fungsi utama, sedangkan yang kedua sebagai fungsi kendala. Masukan ini bertipe string yang dibatasi sebanyak 200 karakter .

2. Masukan batas.

   Masukan ini berupa tanda baik persamaan atau pertidaksamaan, dalam hal ini penulis membatasi pertidaksamaan hanya berupa ≤ (lebih kecil atau sama dengan). Tipe dari batas tersebut adalah string.

3. Masukan nilai x.

   Masukan ini menunjukkan nilai x awal dan bertipe real karena bisa bernilai positif maupun negatif dan berupa pecahan desimal.

4. Masukan nilai y.

   Sama seperti nilai x dan bertipe real.

5. Masukan pilihan metode.

   Masukan ini merupakan pilihan metode yang akan kita gunakan apakah penalti

atau penghalang.

Pada metode-metode diatas setiap kita selesai menuliskan masukan harus diakhiri dengan menekan 'enter'. Penekanan 'enter' ini sebagai tanda bahwa penulisan sebuah masukan sudah selesai(dalam program prosedur ini terdapat dalam file read).

Bila kita selesai memberikan masukan-masukan, maka kita harus meng-klik blok 'selesai' sebagai tanda pemberian masukan-masukan sudah selesai. Dalam memilih menu masukan kita tinggal meng-klik blok-blok yang ada, karena program ini dilengkapi pemakaian mouse yang terdapat dalam file mouse.

Prosedur yang dilakukan terhadap masukan-masukan adalah sebagai berikut:

Setelah selesai penulisan masukan-masukan, kemudian fungsi persamaan akan dibaca dalam notasi infix. Selanjutnya fungsi ini akan dikonversikan menjadi notasi postfix. Prosedurnya sebagai berikut, fungsi persamaan ini akan dibaca perkarakter sebagai berikut:

```
for i := 1 to length(infix) do
begin
   .....;
end;
```

Jika karakter yang dibaca saat itu berupa x,y dan operand, maka karakter tersebut akan disimpan dalam *array*. Terlebih dahulu di-cek jika karakter sebelumnya berupa 'y' atau operand maka membandingkan dan memasukkan operator '*' dulu pada *record* kemudian memasukkan karakter[i] ke *array* .Jika karakter sebelumnya adalah operator, maka terlebih dahulu memasukkan nilai '1' ke *array* kemudian membandingkan dan memasukkan operator '*' ke *record* baru yang terakhir memasukkan karakter[i] ke *array*.

Rutin programnya sebagai berikut :

```
if infix[i] = 'x' then
```

```
begin
 if (infix[i-1] in operand) or (infix[i-1] = 'y') then
  begin
  while (T<> nil) and (valensi('*') <= valensi(T^.huruf) do
   begin
              oper(j);j:=j+1;
    end;
     push(T,'*',0);
      d:= d+kar;
       postfix[j] := d;
  end
   else
       begin
          j := j+1;
          postfix[j] := '1';
         while (T<> nil) and (valensi('*') <= valensi(T^.huruf) do
           begin
                oper(j);j:=j+1;
            end;
            push(T,'*',0);
           d := d+kar;
            j:= j+1;
           postfix[j] := d
       end;
end;
```

Demikian juga jika karakter yang dibaca saat itu berupa 'y', sedangkan jika karakter yang dibaca saat itu berupa operand , maka operand tersebut langsung dimasukkan ke *array.*

Jika karakter yang dibaca saat itu berupa operator, kita akan membandingkan apakah *record* saat itu kosong atau tidak. Bila *record* kosong, maka karakter tersebut akan dimasukkan(di-*push*) ke dalam *record.* Bila *record* tidak kosong terlebih dahulu akan

dibandingkan apakah karakter yang dibaca saat itu memiliki valensi lebih kecil atau sama dengan karakter terakhir pada *record*. Jika karakter yang dibaca saat itu memiliki valensi yang lebih kecil atau sama dengan karakter terakhir dalam *record*, maka karakter yang ada dalam *record* dikeluarkan(di-*pop*) dan dimasukkan ke *array*. Demikian seterusnya sampai karakter terakhir yang ada dalam *record* memiliki valensi lebih kecil daripada karakter yang dibaca saat itu. Kemudian karakter yang dibaca saat itu akan dimasukkan(di-push) ke dalam *record*.

Rutin program nya sebagai berikut :

```
if infix[i] in operator then
begin
 while (T<> nil) and (valensi(kar) <= valensi(T^.huruf) do
   begin
            oper(j);j:=j+1;
    end;
     push(T,'*',0);
end;
```

Bila pembacaan fungsi persamaan sampai pada karakter terakhir, maka karakter yang ada dalam *record* dikeluarkan satu persatu dan dimasukkan ke *array* sampai isi *record* kosong.

Rutin programnya sebagai berikut :

```
if t <> nil then
repeat
  oper(j);j := j+1;
until t= nil;
```

Kemudian tahap selanjutnya membaca isi *array* dari isi pertama sampai isi terakhir. Jika isi *array* berupa x,y dan operand, maka isi *array* akan dimasukkan ke *record* dan memasukkan nilai x dan y.

Rutin programnya sebagai berikut :

```
for i := 1 to j do
begin
  kar1 := postfix[i];
   if kar1 = 'x' then
   push(awal,'',s);
  if kar1 = 'y' then
   push(awal,'',b);
  if kar1 in operand then
  val(kar1,kode,kode11);
   push(awal,'',kode);
end;
```

Jika isi *array* berupa operator, maka nilai dari *record* yang terakhir dan kedua dari yang terakhir dikeluarkan (di-*pop*) dan dilakukan operasi matematika sesuai dengan operator yang dibaca saat itu. Hasilnya akan dimasukkan(di-*push*) ke dalam *record* kembali. Demikian seterusnya sampai isi *record* tinggal satu. Nilai terakhir tersebut merupakan nilai dari fungsi persamaan.

Untuk operasi turunan baik terhadap x maupun y, prosedurnya sama seperti diatas, tetapi jika karakter yang dibaca saat itu berupa x, untuk turunan x maka harus ditinjau terlebih dahulu karakter selanjutnya apakah ada pangkat atau tidak. Jika ada pangkat karakter sesudah pangkat dikalikan ke x, jika tidak karakter sebelumnya dikalikan satu. Selanjutnya jika karakter yang dibaca saat itu berupa *operand*, harus dilihat terlebih dahulu karakter sebelumnya apakah berupa pangkat atau tidak. Jika berupa pangkat, maka karakter tersebut harus dikurangi satu. Jika tidak akan dimasukkan(di-*push*) ke *array*. Demikian juga untuk turunan y.

Rutin programnya sebagai berikut :

```
if ((infix[i] = 'x') or (infix[i] = 'X'))  then
  begin
   if ( i = 1) then
   begin
    j := j+1;
    d := '1';
    postfix[j] := d;
    while(T<>nil) and (valensi('*')<= valensi(T^.huruf)) do
    begin
     oper;j := j +1;end;
    push(T,'*',0);
  end
  else if  (infix[i-1] in operator)then
  begin
   j := j+1;
   d := '1';
   postfix[j] :=d;
   while(T<>nil) and (valensi('*')<= valensi(T^.huruf)) do
   begin
    oper;j := j +1;end;
   push(T,'*',0);
  end
```

Pada metode conjugate gradien diperlukan juga turunan ke-dua. Kita biarkan bertipe string dan dilakukan penurunan pertama. Jadi turunan yang pertama masih dalam tipe string. Selanjutnya kita turunkan lagi sebagaimana mencari turunan ke-satu seperti diatas.

Rutin programnya sebagai berikut :

```
if (kar = 'x') or (kar = 'X') then
    begin
     if (i = 1) then begin
       z := z+'1'; p := p +1; end
```

```
else if infix[i-1] in operator then
begin
  z := z+'1';p := p+1;
end;
if infix[i+1] = '^' then
begin
  z := z+'*'; p := p +1;
  z := z+infix[i+2];  p := p +1;
  z := z+kar; p :=p +1;
end
else {if infix[i-1] in operand then}
z := z+'';
end
```

4.3. Keluaran Program

Sebelum kita mengisi masukan, pada monitor akan tampak 2 pilihan yaitu tabel dan grafik . Bila kita memilih grafik , maka akan tampak sumbu koordinat dan blok-blok menu masukan. Bila salah satu menu masukan dipilih, maka pada sebelah kanan layar akan tampak tempat untuk menuliskan masukan tersebut. Setelah masukan-masukan selesai ditulis kemudian akan diproses sesuai dengan metode yang digunakan. Bila proses pencarian nilai minimum selesai, maka pada sumbu koordinat akan digambar grafik dari fungsi persamaan dilengkapi nilai x(satu dimensi) serta nilai x dan y (dua dimensi). Nilai-nilai ini diperoleh pada tiap-tiap iterasi. Pada kanan bawah akan dituliskan nilai x dan y minimun serta nilai fungsi minimum. Bila tabel yang kita pilih maka setelah masukan-masukan kita isi hasil yang diperoleh tiap iterasi akan ditampilkan pada tabel . Pada baris dua kolom dua bawah juga akan dituliskan nilai terakhir yang diperoleh.

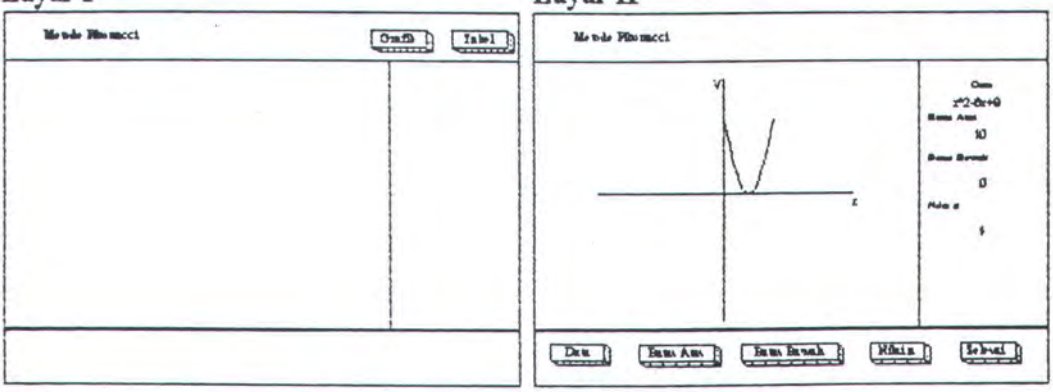
Selanjutnya pada kiri atas akan tampak blok-blok menu pilihan untuk mengulangi kembali atau tidak.

Pada gambar ini akan diberikan contoh metode Fibonacci dengan persamaan yang dimasukkan adalah :

x^2-6x+9 dengan interval 0 - 10 dan n = 4.

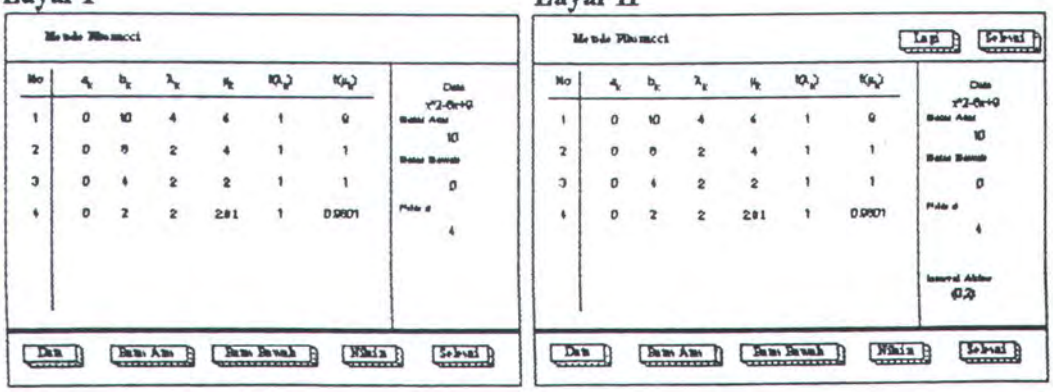
Untuk layar pertama menunjukkan kondisi saat memilih grafik atau tabel. Bila grafik yang kita pilih dan masukan-masukannya kita isi maka tampilan selanjutnya tampak pada layar II

Layar I **Layar II**



Bila tabel yang kita pilih maka tampilan yang kita peroleh setelah masukan-masukan diisi akan tampak pada layar I dibawah ini. Selanjutnya pada baris satu kanan akan tampak pilihan untuk mencoba lagi atau selesai seperti pada layar II di bawah ini.

Layar I **Layar II**



Gambar 4 - 1 Keluaran Program

BAB V

PENGUJIAN DAN ANALISA

BAB V

PENGUJIAN DAN ANALISA

Dalam bab ini akan diberikan beberapa contoh penyelesaian dengan menggunakan program yang telah dibuat dan hasilnya akan dibandingkan dengan hasil perhitungan secara manual. Dari hal tersebut akan terlihat apakah program yang telah dibuat cukup mampu mewakili metode yang digunakan.

5.1 Metode Fibonacci

Pada metode ini akan dicoba contoh sebagai berikut :

Carilah daerah minimum untuk fungsi : F(x) = x^2-6x+9 dengan n = 4 daerah mula-mula antara 0-10.

Dengan menggunakan rumus-rumus yang terdapat dalam metode fibonacci dan merunut tahapan pada algoritma, maka nilai-nilai yang diperoleh secara manual akan diberikan pada tabel berikut ini.

Tabel 5 - 1 Metode Fibonacci dengan perhitungan manual

Iterasi k	a_k	b_k	λ_k	μ_k	$F(\lambda_k)$	$F(\mu_k)$
1	0	10	4	6	1	9
2	0	6	2	4	1	1
3	0	4	2	2	1	1
4	0	2.01	2	2	1	0.98

Interval terakhir yang diperoleh adalah [0,2].

Selanjutnya kita bandingkan dengan hasil yang diperoleh melalui komputasi.

Tabel 5 - 2 Metode Fibonacci dengan perhitungan komputer

Iterasi k	a_k	b_k	λ_k	μ_k	$F(\lambda_k)$	$F(\mu_k)$
1	0	10	4	6	1	9
2	0	6	2	4	1	1
3	0	4	2	2	1	1
4	0	2.01	2	2	1	0.98

Interval terakhir yang diperoleh adalah [0,2].

Pada setiap iterasi menghasilkan nilai-nilai yang sama. Hal ini disebabkan karena soal yang diberikan cukup sederhana.

5.2 Metode Newton

Fungsi yang akan dicoba adalah sebagai berikut :

F(x) = 4x^3-3x^4, x awal = 0.4.

dengan x' = x-f(x)/f'(x)

Nilai yang diperoleh tiap iterasi dengan perhitungan manual terdapat dalam tabel berikut ini.

Tabel 5 - 3 Metode Newton dengan perhitungan manual

Iterasi k	x_k	$f(x)$	$f'(x)$	x_{k+1}
1	0.4	0.1792	1.152	0.244444
2	0.244444	0.0477139	0.5417611	0.1563725
3	0.1563725	0.0135	0.2475441	0.1018367
4	0.1018367	0.0039018	0.1117751	0.0669291
5	0.0669291	0.001139	0.0501563	0.0442201
6	0.0442201	0.0003344	0.0224273	0.0293097

karena nilai dari $f(x6) < \varepsilon$ dimana $\varepsilon = 0.001$, maka perhitungan dihentikan.

Selanjutnya kita bandingkan dengan hasil yang diperoleh melalui komputasi.

Tabel 5 - 4 Metode Newton dengan perhitungan komputer

Iter. k	x_k	$f(x)$	$f'(x)$	x_{k+1}
1	0.4	0.1792	1.152	0.244444
2	0.244444	0.04746045	0.541761317	0.156372490
3	0.156372490	0.0135009644	0.24754432826	0.1018329659
4	0.1018329659	0.0039014044	0.11176739855	0.06692650358
5	0.06692650358	0.00113389086	0.05015259087	0.04421763296
6	0.04421763296	0.00033434868	0.0224249374	0.0293079508

Bila kita bandingkan kedua tabel tersebut secara garis besar sama, tetapi perhitungan dengan komputer akan menghasilkan nilai yang lebih teliti sampai 12 digit.

5.3 Metode Gradien

Fungsi yang akan dicoba sebagai berikut:

$f(x,y) = 4x^2+4y^2+4yx$ dimana (x, y) awal = (5,2)

Nilai yang diperoleh dengan perhitungan manual pada tiap iterasi sebagai berikut:

Tabel 5 - 5 Metode Gradien dengan perhitungan manual

No. Iter.	P_1	P_2	λ Optim.	X'	y'
0	-	-	-	5	2
1	-0.8	-0.6	5.07	0.944	-1.042
2	-0.5959	0.8030	1.36	0.1336	0.0501
3	-0.8051	-0.5932	0.13	0.0289	-0.027

Selanjutnya kita bandingkan dengan hasil yang diperoleh melalui komputasi.

Tabel 5 - 6 Metode Gradien dengan perhitungan komputer

No. Iter.	P_1	P_2	λ Optimal	x'	y'
1	-	-	-	5	2
2	-0.8	-0.6	5.067566	0.945947	-1.040540
3	-0.6	0.8	1.364441	0.127279	0.051010
4	-0.8	-0.6	0.128967	0.024125	-0.026397
5	-0.606	0.795	0.034973	0.002923	0.001417
6	-0.784	-0.6211	0.003189	0.000424	-0.000564
7	-0.373	0.9276	0.000001	0.000423	-0.000563

Dari kedua tabel bisa dilihat perbedaan nilai tidak begitu besar. Komputasi bisa dilakukan dalam beberapa iterasi sampai nilai x dan y minimum.

5.4 Metode Conjugate

Fungsi yang akan dicoba sebagai berikut:

$f(x,y)= 1.5x^2+0.5y^2-yx-2x$ dimana (x , y) awal = (-2,4).

Nilai yang diperoleh dengan perhitungan manual pada tiap iterasi sebagai berikut:

Tabel 5 - 7 Metode Conjugate dengan perhitungan manual

No. Iter.	x_k	y_k	∇f	α	z
1	-2	4	[-12;6]	-	-
2	1.5294118	2.1111111	[0.35;0.706]	0.294	[12;-6]
3	1	1	[0;0]	1.7	[-0.31;-0.73]

Selanjutnya kita bandingkan dengan hasil yang diperoleh melalui komputasi.

Tabel 5 - 8 Metode Conjugate dengan perhitungan komputer

No. Iter.	x_k	y_k	∇f	α	z
1	-2	4	[-12;6]	-	-
2	1.5294118	2.1111111	[0.35;0.706]	0.294	[12;-6]
3	1	1	[0;0]	1.7	[-0.31;-0.73]

Pada metode ini tiap iterasi menghasilkan nilai yang sama.

5.5 Metode Penalti

Fungsi yang akan dicoba sebagai berikut:

F(x) = x^2+4y^2 dengan kendala x+2y-1=0 dan (x, y) awal = (1,1).

Nilai yang diperoleh dengan perhitungan manual pada tiap iterasi sebagai berikut:

Tabel 5 - 9 Metode Penalti dengan perhitungan manual

No Iterasi	r	x'	y'
1	0 .1	0.0833333	0.041666
2	1	0.33333	0.166666
3	10	0.47619	0.238

Selanjutnya kita bandingkan dengan hasil yang diperoleh melalui komputasi.

Tabel 5 - 10 Metode Penalti dengan perhitungan komputer

No Iterasi	r	x'	y'
1	0 .1	0.084	0.042
2	1	0.328	0.172
3	10	0.431	0.262
4	100	0.497	0.249
5	1000	0.523	0.238

Dari kedua tabel tersebut dapat dibandingkan. Pada tiap-tiap iterasi hanya ada perbedaan yang kecil .

5.6 Metode Penghalang

Fungsi yang akan dicoba sebagai berikut:

$F(x) = \frac{1}{2} x$ dengan kendala $1-x \leq 0$ dimulai dari $x_0 = 4$

Hasil dari perhitungan manual terdapat dalam tabel berikut :

Tabel 5 - 11 Metode Penghalang dengan perhitungan manual

No Iterasi	r	x'
1	10	5.472136
2	1	2.4142136
3	0.1	1.4472136

Hasil perhitungan komputer sebagai berikut :

Tabel 5 - 12 Metode Penghalang dengan perhitungan komputer

No Iterasi	r	x'
1	10	5.472136
2	1	2.4142136
3	0.1	1.4472136

Hasil yang diperoleh dari tiap iterasi sama.

Dari semua metode yang dipakai nilai nilai yang diperoleh tiap iterasi mendekati sama, sehingga program visualisasinya bisa digunakan untuk mempermudah penyampaian mata kuliah optimasi.

BAB VI

KESIMPULAN DAN SARAN

BAB VI

KESIMPULAN DAN SARAN

6.1 Kesimpulan

Dengan menggunakan komputer dapat dilakukan pencarian nilai - nilai optimum dari suatu fungsi satu dimensi, dua dimensi dan dengan atau tanpa penghalang. Ada beberapa keuntungan yang dapat penulis catat disisni :

1. Waktu yang dibutuhkan untuk mencari nilai optimum dari fungsi - fungsi yang ada jauh lebih sedikit bila dibandingkan dengan perhitungan secara manual. Diharapkan pengurangan waktu ini dapat dimanfaatkan untuk lebih mendalami ilmu optimasinya itu sendiri.

2. Ketelitian yang dihasilkan jauh lebih baik, karena perhitungan dengan komputer mempunyai angka signifikan yang lebih banyak.

6.2 Saran

1. Penulis berharap visualisasi optimasi ini bisa dikembangkan lebih baik lagi dengan menggunakan bahasa pemrograman yang bisa menghasilkan tampilan lebih baik, misalnya visual C++, Visual Basic dengan operating sistem window.

2. .Dalam program yang dibuat dalam tugas akhir ini *array* digunakan untuk menyimpan karakter, yang ternyata cukup banyak menghabiskan memori. Dalan program ini setiap persamaan yang kita masukkan akan disimpan ke dalam array. Array yang disediakan dari 1 - 50 tempat. Meskipun tempat yang

dipakai hanya 25 secara otomatis tempat yang diblok tetap 50. Bila kita menggunakan pointer tempat yang dipakai tergantung persamaan yang dimasukkan. Misalkan hanya butuh 25 tempat, maka tempat yang dipakai/diblok hanya 25 juga. Oleh karena itu untuk pengembangan lebih lanjut sebaiknya menggunakan *pointer* .

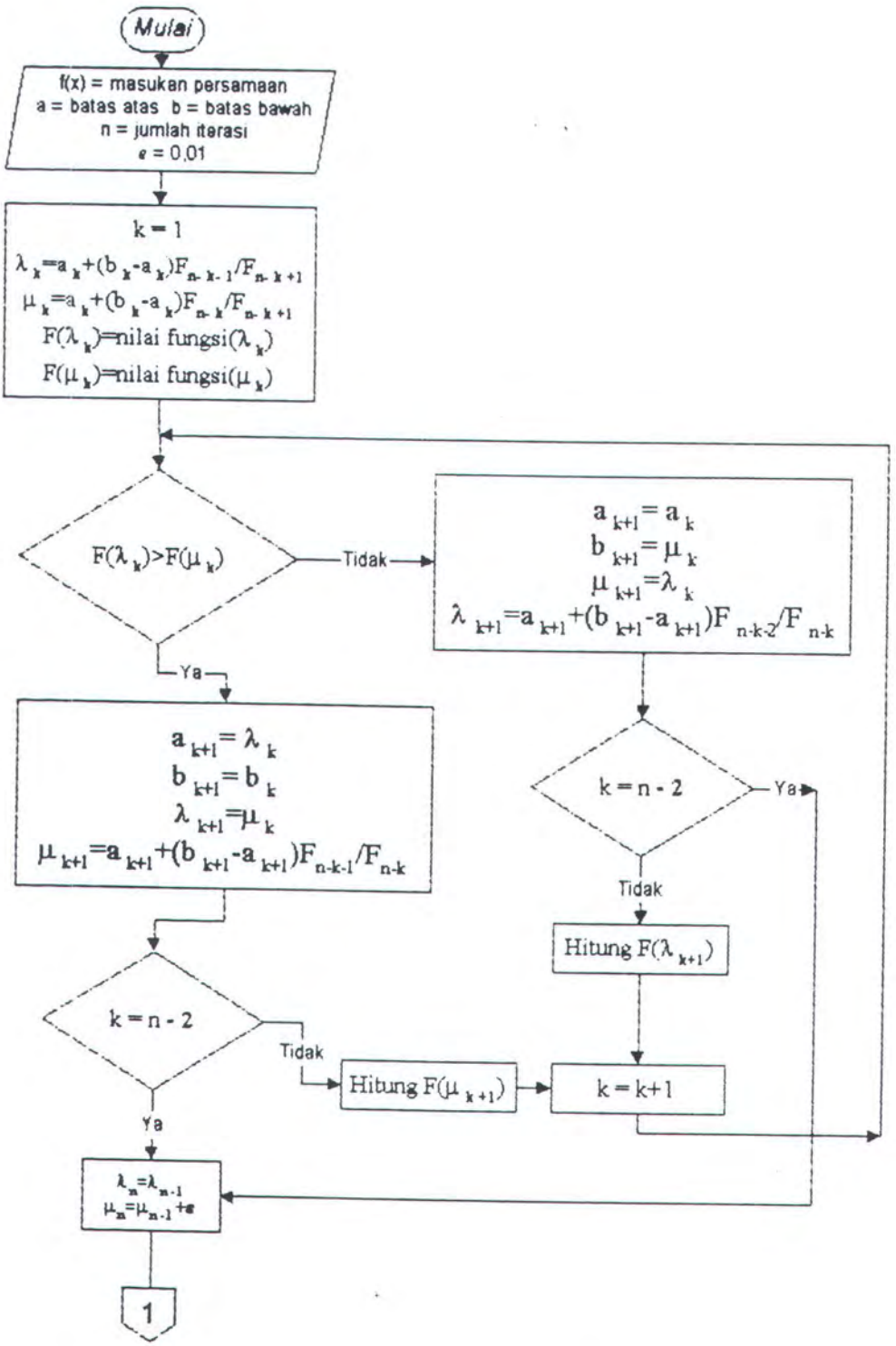
DAFTAR PUSTAKA

Daftar Pustaka

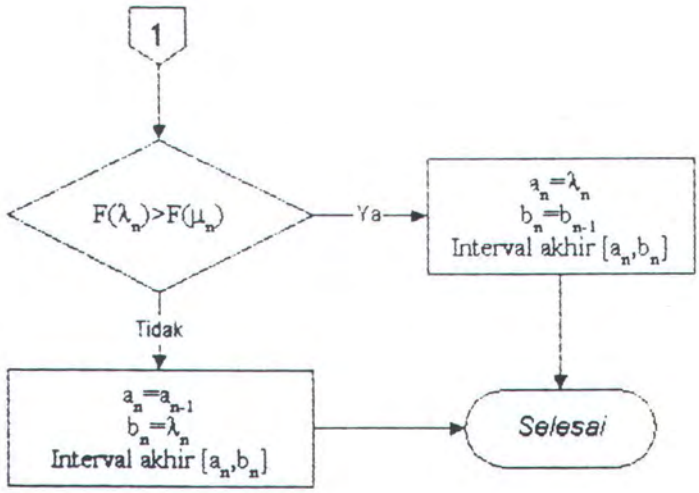
1. Bazara, Mokhtar S. / CM Shetty,1990, *Non Linear Programing*, John Wiley & Son, Singapore
2. Scales, L. E., 1985, *Introduction to Non Linear Optimization*, Mac Millan London
3. Beveridge, Gordon S. G. / Schechter, Robert S. ,1970, *Optimization : Theory and Practice*, Mc Graw - Hill, London
4. Jogiyanto H. M., 1995, *Turbo Pascal*, Andi Offset, Yogyakarta
5. P. Insap Santosa, Ir, M sc, 1994, *Grafika Komputer dan Antarmuka Grafis*, Andi Offset, Yogyakarta.
6. Chapra, Steven C., 1985, *Numerical Methods for Engineers*, Mc Graw - Hill, London
7. Hiller, Frederick S., 1990, *Introduction to Operations Research*, Mc Graw - Hill, Singapore

LAMPIRAN

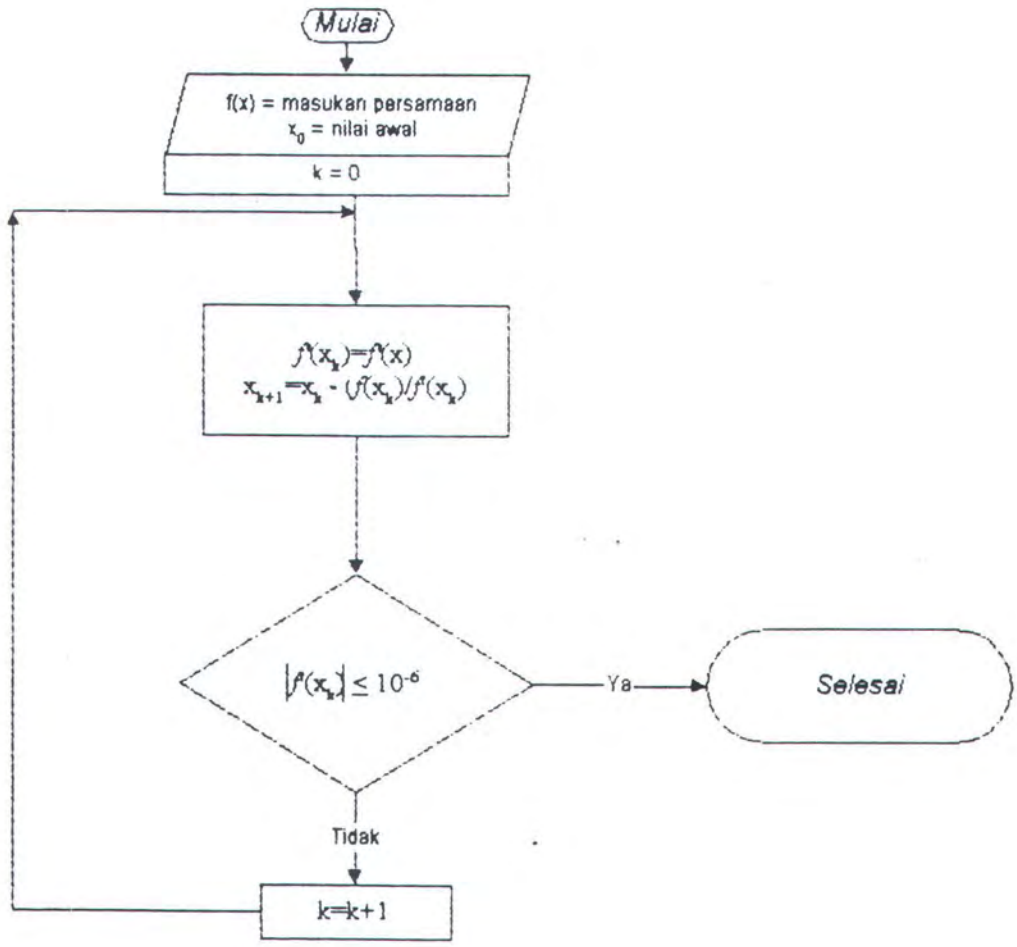
LAMPIRAN A-1
BAGANALIR METODE FIBONACCI



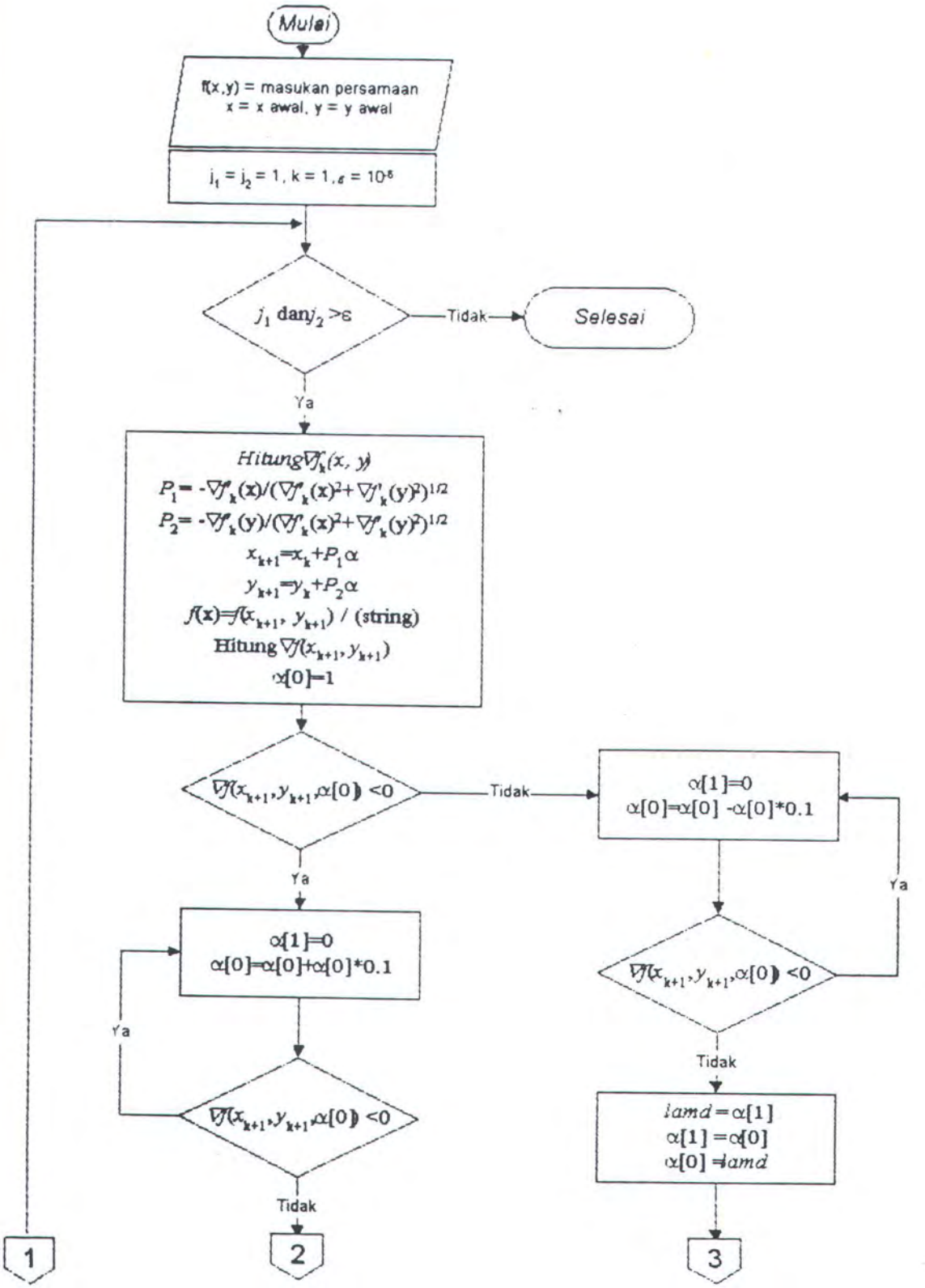
LAMPIRAN A-1.1
BAGANALIR METODE FIBONACCI



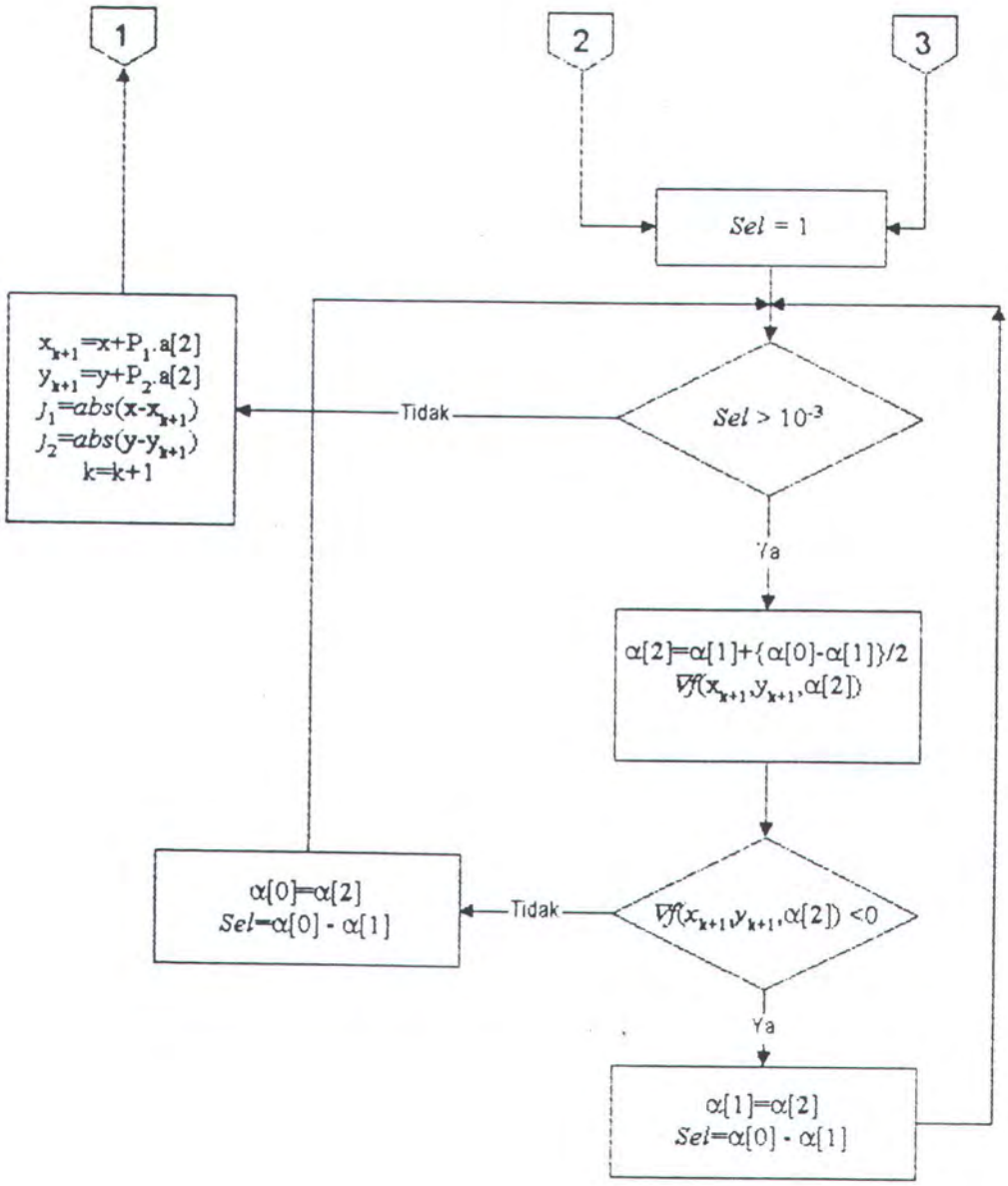
LAMPIRAN A-2
BAGANALIR METODE NEWTON



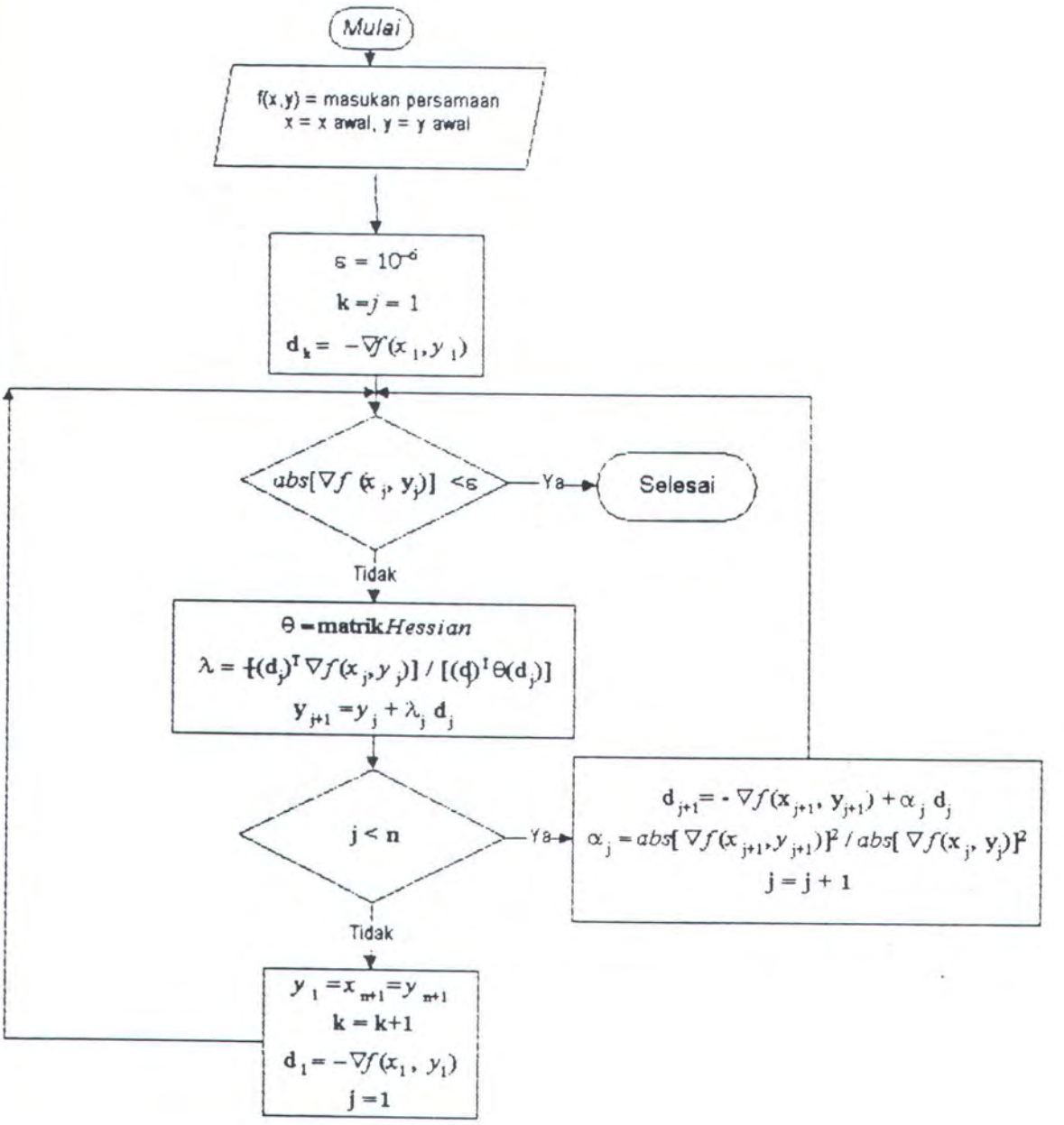
LAMPIRAN A-3
BAGANALIR METODE GRADIEN



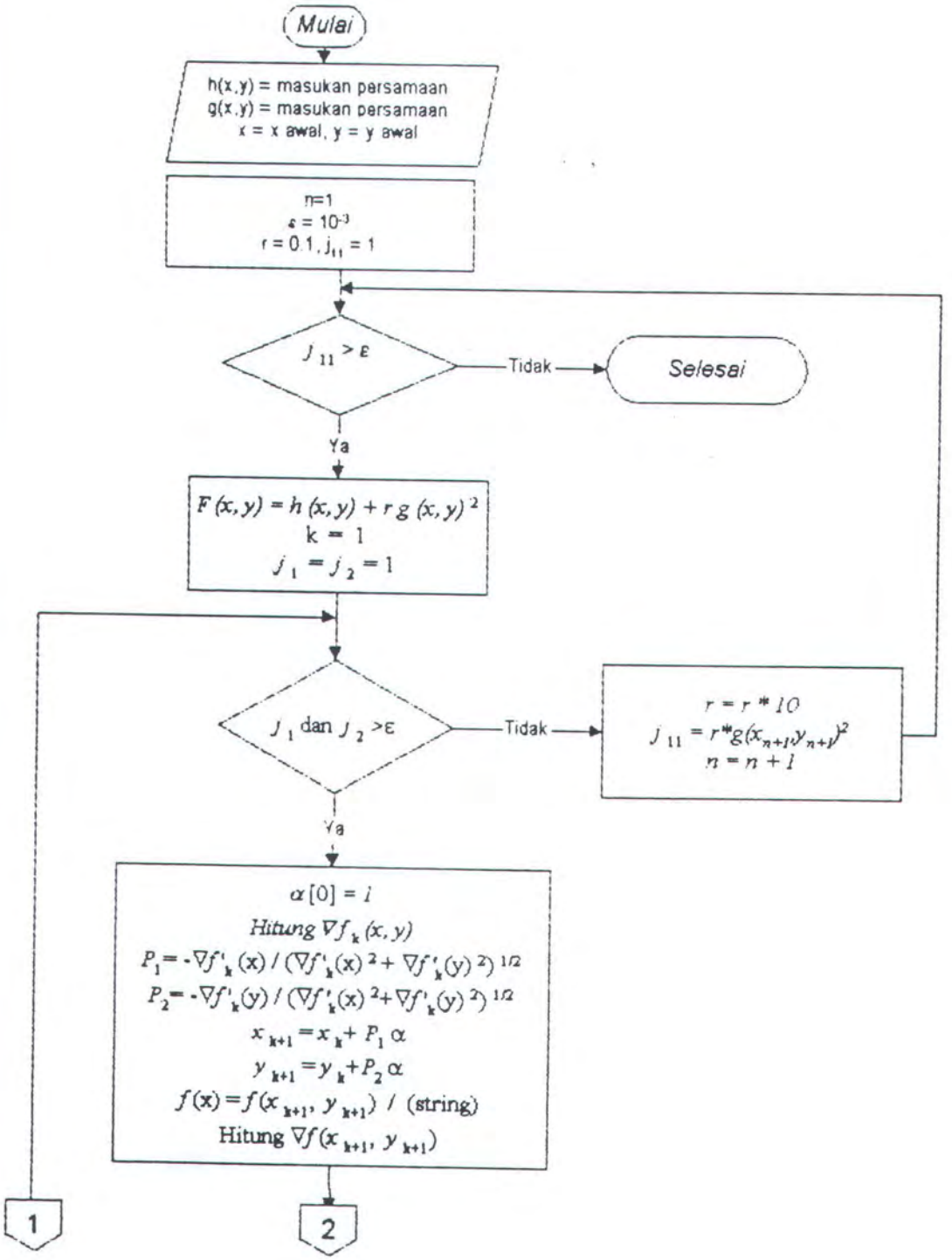
LAMPIRAN A-3.1
BAGANALIR METODE GRADIEN



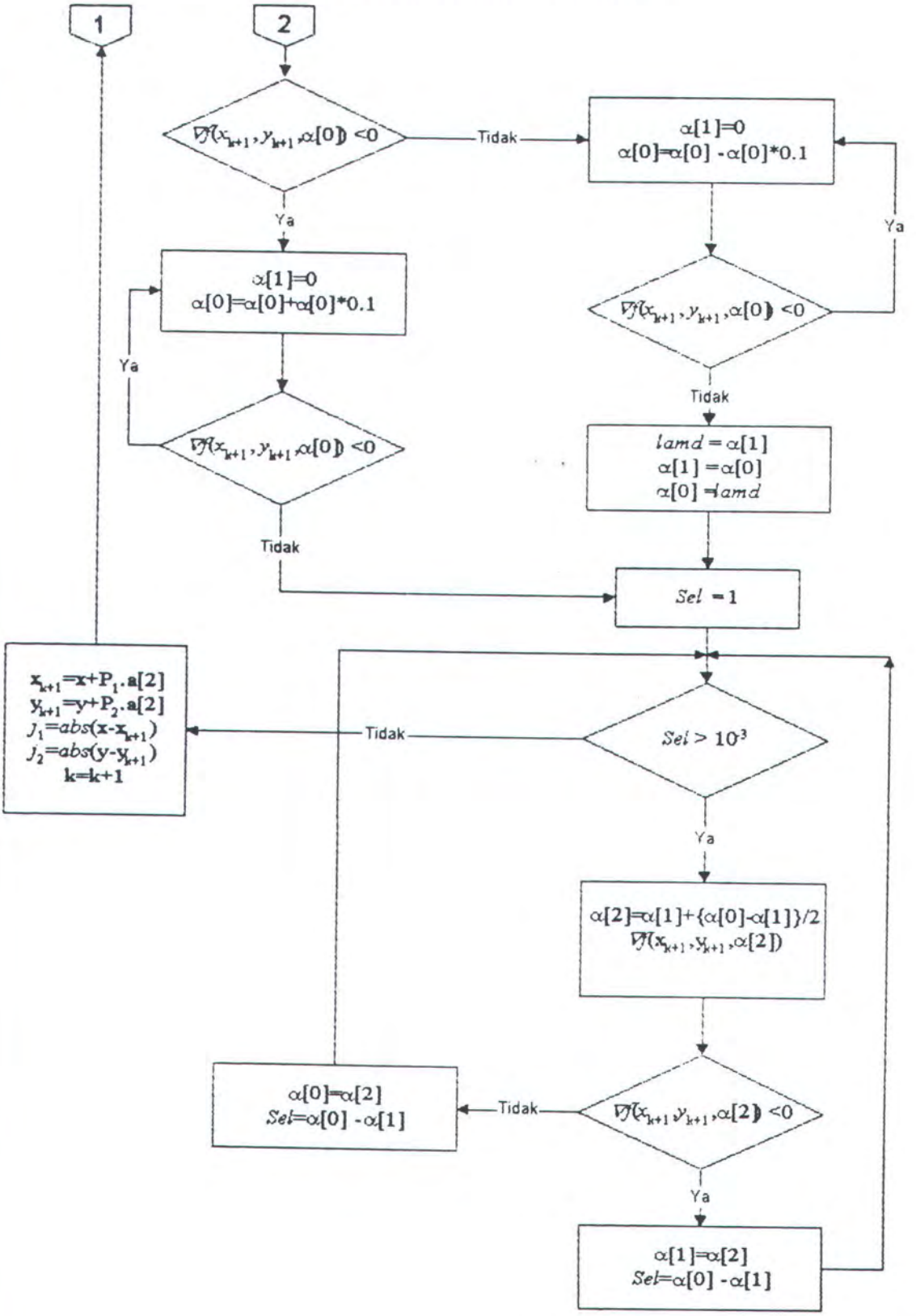
LAMPIRAN A-4
BAGANALIR METODE CONJUGATE



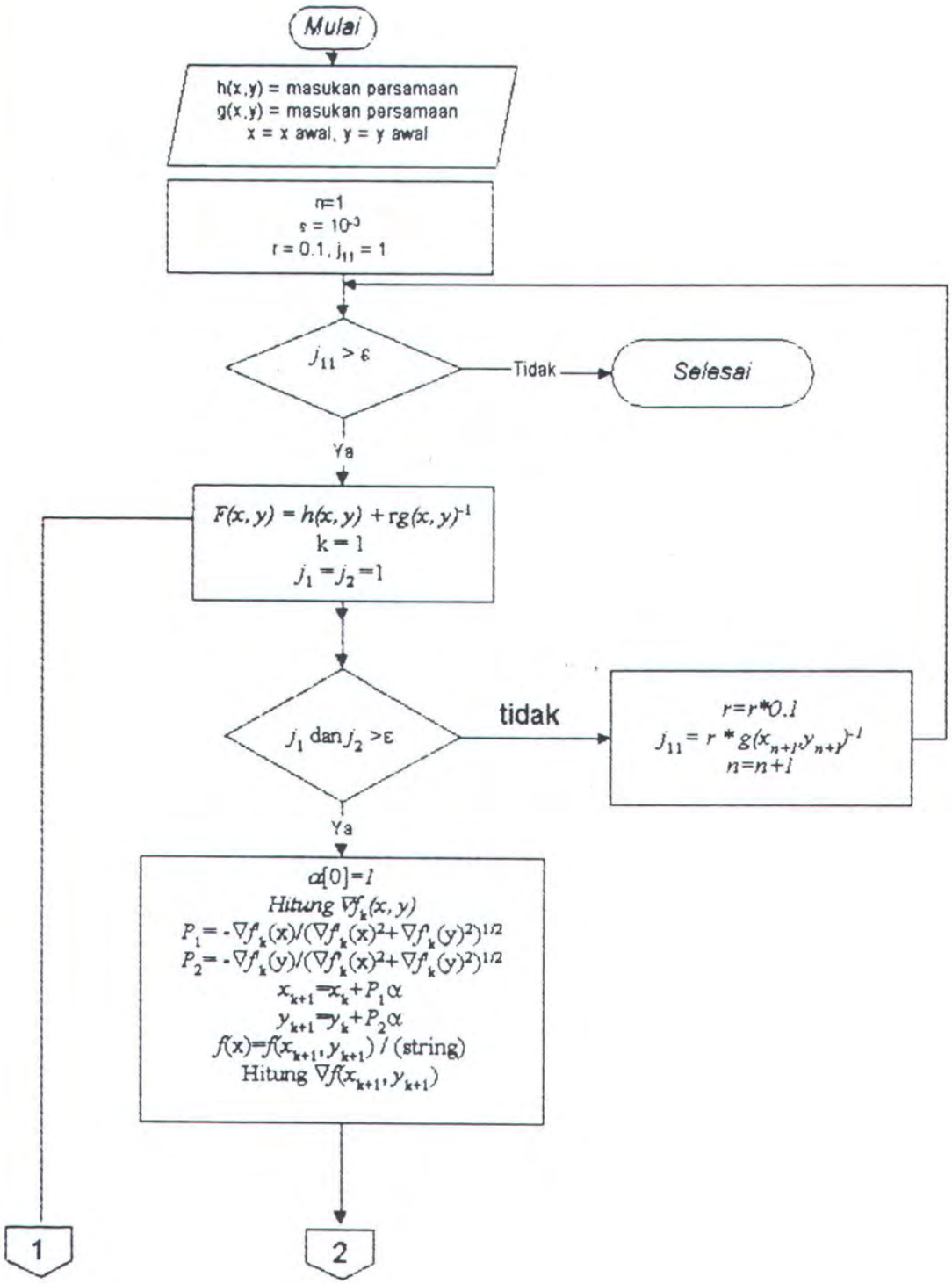
LAMPIRAN A-5
BAGANALIR METODE PENALTI



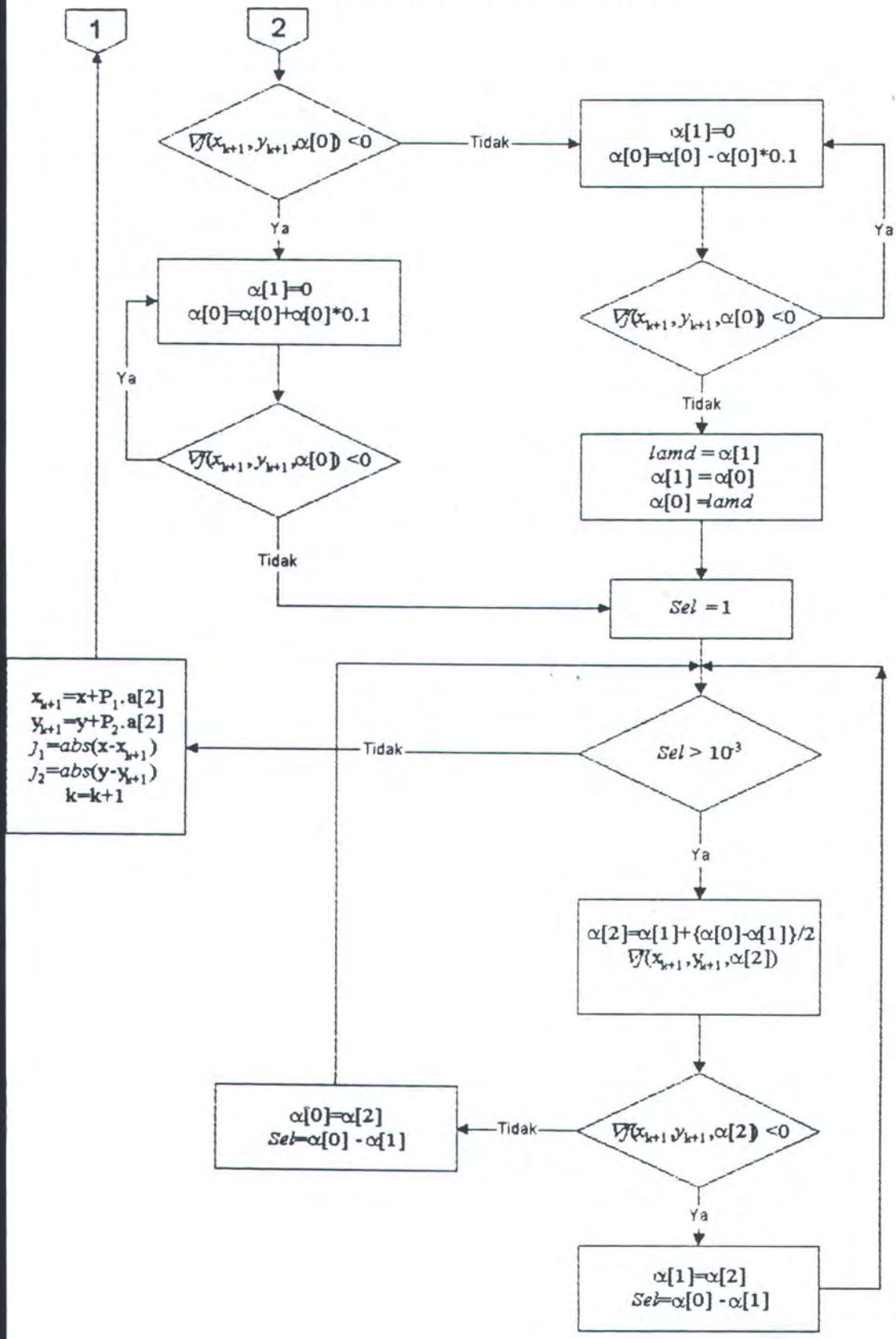
LAMPIRAN A-5.1
BAGANALIR METODE PENALTI



LAMPIRAN A-6
BAGANALIR METODE PENGHALANG



LAMPIRAN A-6.1
BAGANALIR METODE PENGHALANG



LAMPIRAN B

PETUNJUK PENGGUNAAN

Visualisai Optimasi

Oleh : R. Nanik H.

Daftar isi :

---

1). Petunjuk Penting:

Persyaratan perangkat keras /lunak :

1. PC 386 atau yang lebih tinggi dengan sistim operasi DOS.
2. VGA-monitor.
3. *Mouse.*

---

2). Daftar file - file yang dipakai :

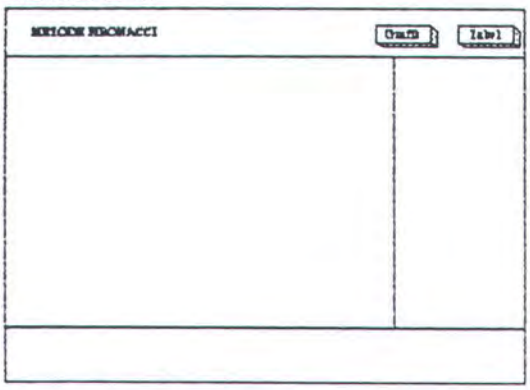
1. conj.exe.
2. fibonac.exe.
3. grad1.exe.
4. inpenalt.exe.
5. newt.exe.
6. att.bgi.
7. egavga.bgi.
8. graph.int.
9. init.
10. initgrp.tpu.
11. kotak.tpu.
12. *mouse*.tpu.
13. persam.tpu.
14. persam5.tpu.
15. read.tpu.
16. tur3.tpu.
17. turunan2.tpu.
18. turunan3.tpu.
19. turunan4.tpu.
20. turunan5.tpu.

3). Cara menggunakan :

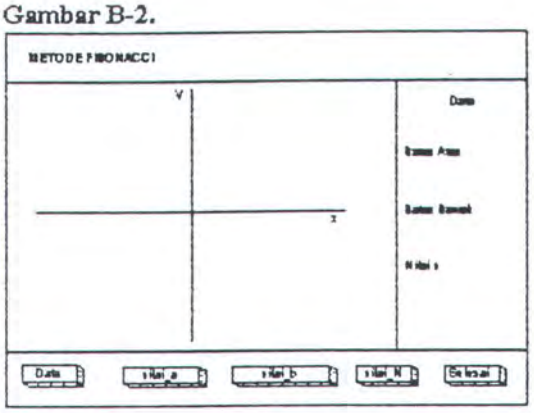
METODE FIBONACCI

Pilih fibonac.exe dan tekan tombol *enter*, selanjutnya pada layar monitor akan tampak tampilan seperti gambar B-1. Dari gambar tersebut tampak dua pilihan Grafik dan Tabel yang dapat dipilih dengan *mouse*. Bila dipilih Grafik maka pada layar akan tampak seperti gambar B-2 dan apabila dipilih Tabel maka pada layar akan tampak seperti gambar B-3. Proses selanjutnya adalah memasukkan data dengan cara memilih pilihan yang ada dibaris bawah. Dengan memilih ikon Data maka pada baris dua kolom dua akan muncul tulisan data yang selanjutnya harus kita isi dengan persamaan yang diakhiri dengan menekan tombol *enter*.

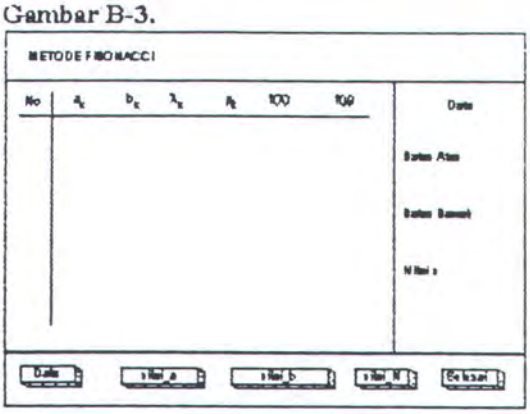
Gambar B-1.



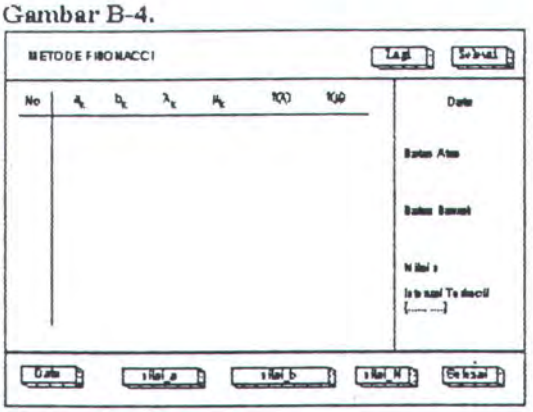
Gambar B-2.



Gambar B-3.



Gambar B-4.



Pilihan selanjutnya adalah nilai_a yang akan memunculkan tulisan batas atas pada baris dua kolom dua dan harus kita isi dengan nilai batas atas yang kita tentukan dan diakhiri dengan menekan tombol *enter*. Pilihan selanjutnya adalah nilai_b yang akan memunculkan tulisan batas bawah pada baris dua kolom dua dan harus kita isi dengan nilai batas bawah yang kita tentukan dan diakhiri dengan menekan tombol *enter*. Pilihan selanjutnya adalah nilai_n

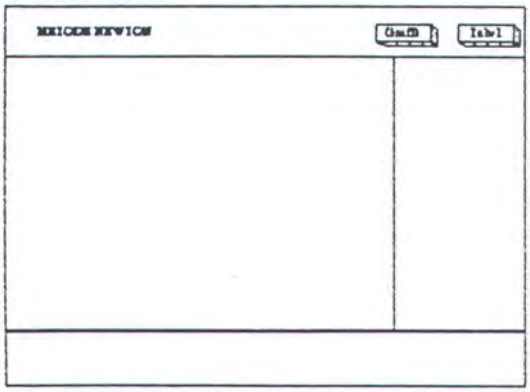
yang akan memunculkan tulisan nilai n pada baris dua kolom dua dan harus kita isi dengan n deret Fibonacci yang kita tentukan dan diakhiri dengan menekan tombol *enter*. Pilihan terakhir adalah selesai yang harus kita pilih dengan menggunakan *mouse*, dan pilihan ini akan memerintahkan komputer untuk mulai menghitung menggunakan metode Fibonacci dengan bentuk tampilan seperti yang kita tentukan.

Setelah melewati beberapa tahapan iterasi maka akan diperoleh interval terkecil yang akan terlihat pada baris dua kolom dua bawah, selanjutnya pada baris satu sebelah kanan akan muncul dua pilihan 'lagi dan selesai' yang akan memerintahkan komputer untuk mengakhiri program atau mengulangi lagi seperti tampak pada gambar B-4.

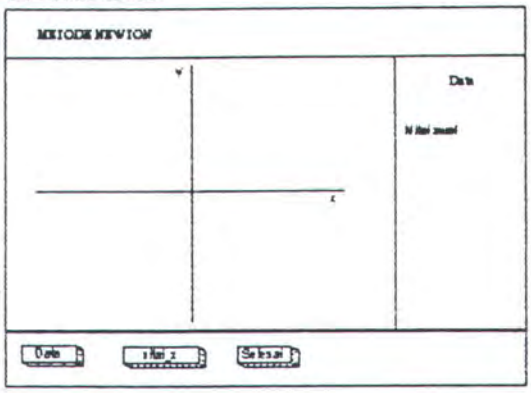
METODE NEWTON

Pilih newt.exe dan tekan tombol *enter*, selanjutnya pada layar monitor akan tampak tampilan seperti gambar B-5.

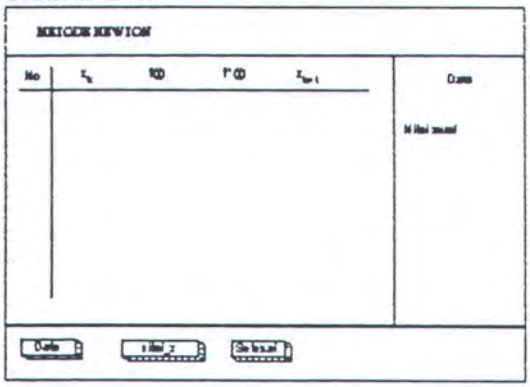
Gambar B-5.



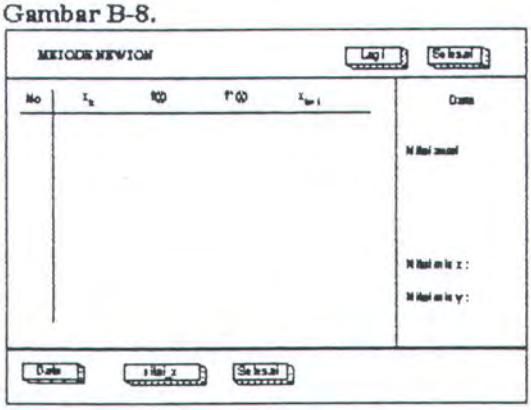
Gambar B-6.



Gambar B-7.



Gambar B-8.



Dari gambar tersebut tampak dua pilihan Grafik dan Tabel yang dapat dipilih dengan *mouse*. Bila dipilih Grafik maka pada layar akan tampak seperti gambar B-6 dan apabila dipilih Tabel maka pada layar akan tampak seperti gambar B-7. Proses selanjutnya adalah

memasukkan data dengan cara memilih pilihan yang ada dibaris bawah. Dengan memilih ikon Data maka pada baris dua kolom dua akan muncul tulisan data yang selanjutnya harus kita isi dengan persamaan yang akan kita masukkan dan diakhiri dengan menekan tombol *enter*. Pilihan selanjutnya adalah nilai_x yang akan memunculkan tulisan Nilai x pada baris dua kolom dua dan harus kita isi dengan nilai x awal yang kita tentukan dan diakhiri dengan menekan tombol *enter*. Pilihan terakhir adalah selesai yang harus kita pilih dengan menggunakan *mouse*, dan pilihan ini akan memerintahkan komputer untuk mulai menghitung dengan menggunakan metode Newton dengan bentuk tampilan seperti yang kita tentukan.

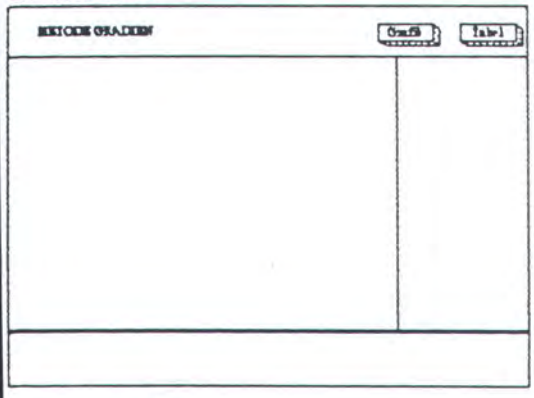
Setelah proses pencarian nilai x minimum selesai maka hasil yang diperoleh serta nilai fungsi (persamaan) dengan x minimum akan ditampilkan pada baris dua kolom dua bawah. Selanjutnya pada baris satu sebelah kanan akan muncul dua pilihan 'lagi dan selesai' yang akan memerintahkan komputer untuk mengakhiri program atau mengulangi lagi seperti tampak pada gambar B-8.

METODE GRADIEN

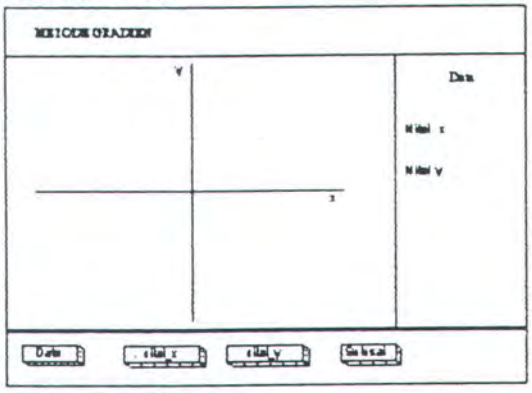
Pilih grad1.exe dan tekan tombol *enter*, selanjutnya pada layar monitor akan tampak tampilan seperti gambar B-9. Dari gambar tersebut tampak dua pilihan Grafik dan Tabel yang dapat dipilih dengan *mouse*. Bila dipilih Grafik maka pada layar akan tampak seperti gambar B-10 dan apabila dipilih Tabel maka pada layar akan tampak seperti gambar B-11. Proses selanjutnya adalah memasukkan data dengan cara memilih pilihan yang ada dibaris bawah. Dengan memilih ikon Data maka pada baris dua kolom dua akan muncul tulisan data yang selanjutnya harus kita isi dengan persamaan yang akan kita cari nilai minimumnya dan diakhiri dengan menekan tombol *enter*. Pilihan selanjutnya adalah nilai_x yang akan memunculkan tulisan Nilai x pada baris dua kolom dua dan harus kita isi dengan nilai x awal yang kita tentukan dan diakhiri dengan menekan tombol *enter*. Pilihan selanjutnya adalah nilai_y yang akan memunculkan tulisan Nilai y pada baris dua kolom dua dan harus kita isi dengan nilai y awal yang kita tentukan dan diakhiri dengan menekan tombol *enter*. Pilihan terakhir adalah selesai yang harus kita pilih dengan menggunakan *mouse*, dan pilihan ini akan memerintahkan komputer untuk memulai menghitung nilai minimum dengan metode Gradien dengan bentuk tampilan seperti yang kita tentukan. Setelah nilai minimum dihasilkan oleh komputer yang di tandai dengan munculnya nilai (x,y) serta nilai minimum fungsi dengan x,y yang diperoleh pada baris dua kolom dua bawah, maka pada baris satu

sebelah kanan akan muncul dua pilihan 'lagi dan selesai' yang akan memerintahkan komputer untuk mengakhiri program atau mengulangi lagi seperti pada gambar B-12.

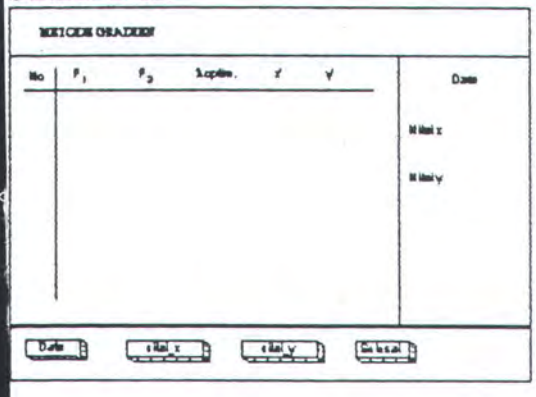
Gambar B-9.



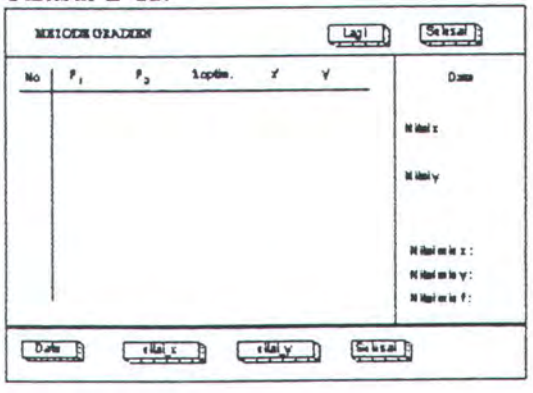
Gambar B-10.



Gambar B-11.



Gambar B-12.



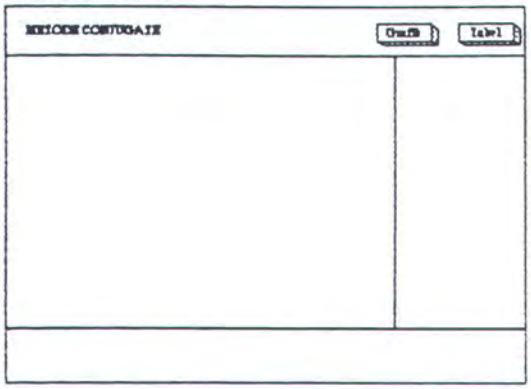
METODE *CONJUGATE*

Pilih conj.exe dan tekan tombol *enter*, selanjutnya pada layar monitor akan tampak tampilan seperti gambar B-13. Dari gambar tersebut tampak dua pilihan Grafik dan Tabel yang dapat dipilih dengan *mouse*. Bila dipilih Grafik maka pada layar akan tampak seperti gambar B-14 dan apabila dipilih Tabel maka pada layar akan tampak seperti gambar B-15. Proses selanjutnya adalah memasukkan data dengan cara memilih pilihan yang ada dibaris bawah. Dengan memilih ikon Data maka pada baris dua kolom dua akan muncul tulisan data yang selanjutnya harus kita isi dengan persamaan yang akan kita masukkan dan diakhiri dengan menekan tombol *enter*. Pilihan selanjutnya adalah nilai_x yang akan memunculkan tulisan Nilai x pada baris dua kolom dua dan harus kita isi dengan nilai x awal yang kita tentukan dan diakhiri dengan menekan tombol *enter*. Pilihan selanjutnya adalah nilai_y yang akan memunculkan tulisan Nilai y pada baris dua kolom dua dan harus kita isi dengan nilai y awal yang kita tentukan dan diakhiri dengan menekan tombol *enter*. Pilihan terakhir

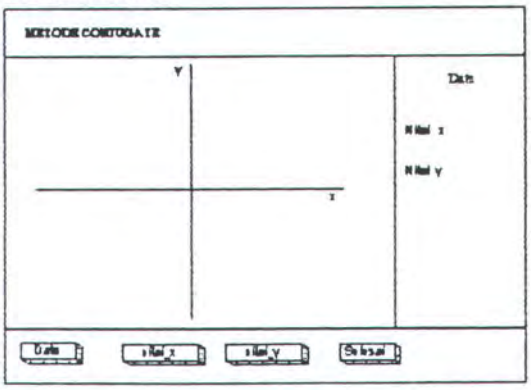
adalah selesai yang harus kita pilih dengan menggunakan *mouse*, dan pilihan ini akan memerintahkan komputer untuk memulai menghitung nilai minimum dengan menggunakan metode *Conjugate* dengan bentuk tampilan seperti yang kita tentukan.

Setelah diperoleh nilai minimum pada baris dua kolom dua bawah akan ditampilkan nilai x, nilai y serta nilai minimum fungsi. Selanjutnya pada baris satu sebelah kanan akan muncul dua pilihan 'lagi dan selesai' yang akan memerintahkan komputer untuk mengakhiri program atau mengulangi lagi seperti tampak pada gambar B-16.

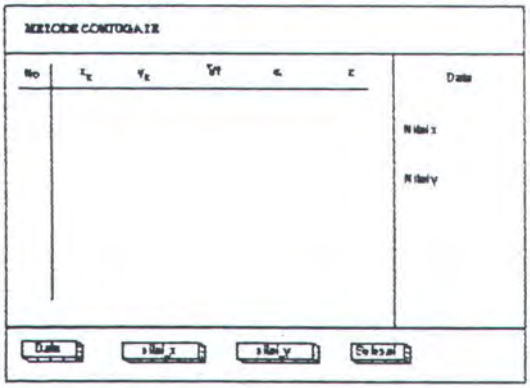
Gambar B-13.



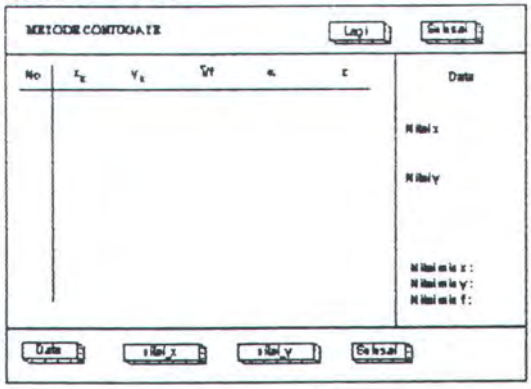
Gambar B-14.



Gambar B-15.



Gambar B-16.

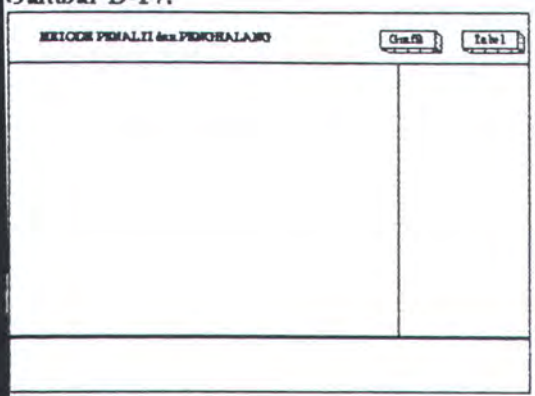


METODE PENALTI DAN PENGHALANG

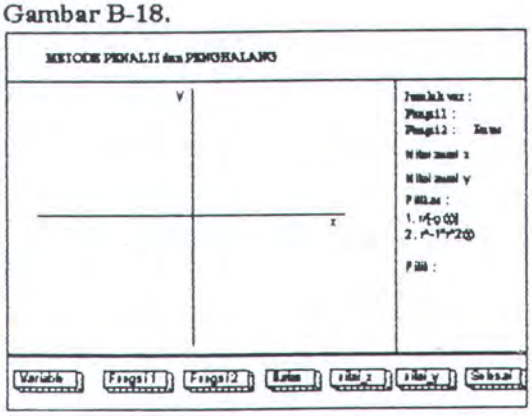
Pilih inpenalt.exe dan tekan tombol *enter*, selanjutnya pada layar monitor akan tampak tampilan seperti gambar B-17. Dari gambar tersebut tampak dua pilihan Grafik dan Tabel yang dapat dipilih dengan *mouse*. Bila dipilih Grafik maka pada layar akan tampak seperti gambar B-18 dan apabila dipilih Tabel maka pada layar akan tampak seperti gambar B-19. Proses selanjutnya adalah memasukkan data dengan cara memilih pilihan yang ada dibaris bawah. Dengan memilih ikon Variabel maka pada baris dua kolom dua akan muncul tulisan jumlah variabel yang menunjukkan dimensi dari fungsi utama dan harus kita isi dengan

angka 1 (satu) untuk satu dimensi atau 2(dua) untuk dua dimensi dan diakhiri dengan menekan tombol *enter*. Pilihan selanjutnya adalah fungsi1 yang akan memunculkan tulisan fungsi1 pada baris dua kolom dua dan harus kita isi dengan fungsi utama yang diakhiri dengan menekan tombol *enter*. Pilihan selanjutnya adalah fungsi2 yang akan memunculkan tulisan fungsi2 pada baris dua kolom dua dan harus kita isi dengan fungsi kendala yang diakhiri dengan menekan tombol *enter*. Pilihan selanjutnya adalah batas yang akan memunculkan tulisan batas pada baris dua kolom dua dan harus kita isi dengan batas yang kita tentukan dan diakhiri dengan menekan tombol *enter*.

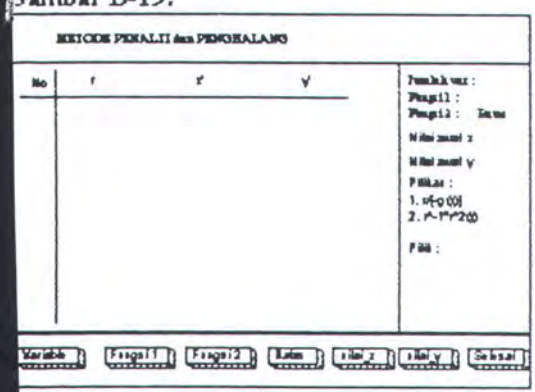
Gambar B-17.



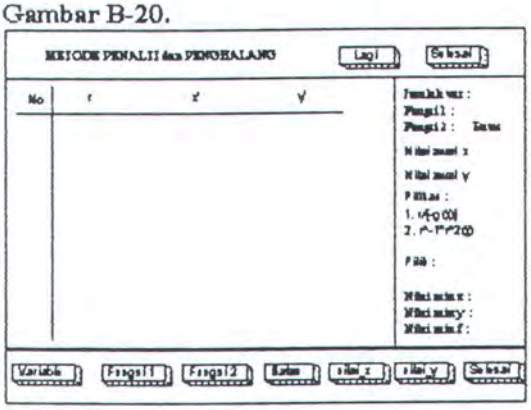
Gambar B-18.



Gambar B-19.



Gambar B-20.



ilihan selanjutnya adalah nilai_x yang akan memunculkan tulisan Nilai x pada baris dua olom dua dan harus kita isi dengan nilai x awal diakhiri dengan menekan tombol *enter*. ilihan selanjutnya adalah nilai_y yang akan memunculkan tulisan Nilai y pada baris dua olom dua dan harus kita isi dengan nilai y awal diakhiri dengan menekan tombol *enter*. elanjutnya akan muncul pilihan metode yang akan kita pakai apakah Penalti atau enghalang dengan memasukkan angka 1(satu) untuk penalti dan angka 2(dua) untuk enghalang sesuai dengan rumus yang digunakan tiap-tiap metode. Setelah proses pencarian lai minimum selesai pada baris dua kolom dua bawah akan ditampilkan nilai x dan y

serta nilai minimum fungsi dengan (x,y) yang telah diperoleh. Kemudian pada baris satu sebelah kanan akan muncul dua pilihan 'lagi dan selesai' yang akan memerintahkan komputer untuk mengakhiri program atau mengulangi lagi seperti pada gambar B-20.

4). Catatan Penting:

- Setiap memasukkan persamaan tidak boleh diawali dengan tanda negatif '-'.
- Setiap memasukkan persamaan, variabel y harus didepan x, misalnya
  - y^2x+5yx.
  - yx^3-yx.

  bukan dengan penulisan
  - xy^2+5xy.
  - x^3y-xy.