

515.5
Bali
t

**TINJAUAN NUMERIK METODE FUNGSI PENALTI
INTERIOR MELALUI PERUBAHAN PARAMETER PENALTI
PADA MINIMISASI DENGAN KENDALA**

Diajukan Untuk Melengkapi Persyaratan
Menempuh Ujian Sarjana Matematika



TUGAS AKHIR

Diajukan Oleh :
IWAN ABU BAKAR
0387010246



Universitas Indonesia
Perpustakaan

UNIVERSITAS INDONESIA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
JURUSAN MATEMATIKA
DEPOK
1992

ABSTRAK



IWAN ABU BAKAR (0387010246)

TINJAUAN NUMERIK METODE FUNGSI PENALTI INTERIOR
MELALUI PERUBAHAN PARAMETER PENALTI
PADA MINIMISASI DENGAN KENDALA

iv + 111 halaman (1992)

Bibl. 11 (1967 - 1991)

Dalam tugas akhir ini ditinjau penggunaan beberapa rumusan untuk menentukan nilai parameter penalti awal yang digunakan pada penyelesaian persoalan minimisasi dengan kendala melalui *Metode Fungsi Penalti Interior*. Tiga rumusan yang ditinjau, yaitu rumusan Fiacco-McCormick (1968), rumusan Rao (1984) dan sebuah rumusan percobaan. Kesemua rumusan tersebut dicobakan pada beberapa kasus, termasuk kasus minimisasi fungsi Rosenbrock dalam dimensi dua.

Efisiensi dari metode fungsi penalti interior dalam ditingkatkan dengan memilih nilai parameter penalti awal yang tepat untuk setiap persoalan.

KATA PENGANTAR

Puji dan syukur penulis panjatkan ke hadirat Alloh s.w.t atas karunia dan rahmat-Nya, yang diberikan kepada penulis sehingga tugas akhir ini dapat terselesaikan.

Tugas akhir ini merupakan salah satu syarat untuk dapat menempuh ujian sarjana pada jurusan Matematika FMIPA UI.

Pada kesempatan ini penulis mengucapkan terima kasih dan penghargaan yang setinggi-tingginya kepada :

- Bapak Dr. Djati Kerami, selaku dosen pembimbing dalam penulisan tugas akhir ini, yang telah meluangkan waktu dan tenaganya untuk membimbing penulis.
- Ibu Dr. Belawati Widjaya, selaku Ketua Jurusan Matematika FMIPA UI.
- Ibu Dra. Kiki Aryanti MS., selaku pembimbing akademis.
- Ayah dan Ibu yang telah memberikan nasihat, doa serta dukungannya baik moril maupun materil.
- Rekan-rekan asisten di Lab. Komputasi Matematika jurusan Matematika FMIPA UI atas segala bantuan yang penulis terima.
- Rekan-rekan di jurusan Matematika FMIPA UI, khususnya rekan se-angkatan - *The Unforgettable Math '87*. Semoga persahabatan di antara kita tetap langgeng hingga nanti.

Penulis menyadari sepenuhnya bahwa dalam tugas akhir ini banyak dijumpai kekurangan di sana-sini. Untuk itu semua saran maupun kritik yang membangun akan penulis terima dengan senang hati. Akhirnya, dengan segala kerendahan hati, penulis berharap tugas akhir ini dapat bermanfaat bagi yang membacanya.



Depok, April 1992

Penyusun

DAFTAR ISI

	HAL
ABSTRAK	i
KATA PENGANTAR	ii
DAFTAR ISI	iv
BAB I PENDAHULUAN	1
BAB II OPTIMISASI TANPA KENDALA	5
II.1 METODE DAVIDON-FLETCHER-POWELL	7
II.2 METODE INTERPOLASI KUBIK	9
BAB III METODE FUNGSI PENALTI INTERIOR	16
BAB IV PERCOBAAN NUMERIK DAN ANALISIS KASUS	25
IV.1 STRUKTUR PROGRAM	25
IV.2 ANALISIS KASUS	30
BAB V KESIMPULAN	73
DAFTAR PUSTAKA	76
LAMPIRAN 1. TEOREMA KONVERGENSI METODE FUNGSI PENALTI INTERIOR	78
LAMPIRAN 2. LIST PROGRAM MFPI	81

Akhirnya



BAB I

PENDAHULUAN

Kita mungkin pernah dihadapkan pada persoalan mencari penyelesaian optimum dari sebuah fungsi obyektif dengan beberapa kendala yang harus dipenuhi, atau tanpa kendala. Persoalan ini disebut persoalan optimisasi. Apabila disertai dengan beberapa kendala, persoalannya disebut persoalan optimisasi dengan kendala, jika tidak persoalannya disebut persoalan optimisasi tanpa kendala.

Dengan menggunakan suatu metode yang dikembangkan, penyelesaian persoalan optimisasi yang sederhana, mungkin dapat dikerjakan secara manual. Tetapi untuk persoalan optimisasi yang kompleks, penyelesaian secara manual hampir tidak mungkin dapat dilakukan. Untuk keadaan seperti ini, penggunaan alat bantu mutlak diperlukan.

Penggunaan alat bantu untuk menyelesaikan persoalan optimisasi tidak terlepas dari metode yang digunakan. Terutama menyangkut tingkat efisiensi dari metode tersebut. Tingkat efisiensi di sini didasarkan pada kecepatan konvergensi untuk mencapai penyelesaian yang diinginkan.

Pada tugas akhir ini akan dibahas salah satu metode yang digunakan untuk menyelesaikan persoalan optimisasi dengan kendala, yaitu metode fungsi penalti interior. Di sini pembahasannya lebih ditekankan pada aspek komputasinya

daripada metodenya itu sendiri.

Pada metode fungsi penalti interior, persoalan optimisasi dengan kendala diubah menjadi barisan persoalan optimisasi tanpa kendala. Barisan persoalan optimisasi tanpa kendala ini dibentuk dengan menambahkan suku penalti pada fungsi obyektifnya. Untuk mengendalikan agar penyelesaian dari barisan optimisasi tanpa kendala yang dibentuk menuju ke penyelesaian dari persoalan semula, pada bagian suku penalti diberikan sebuah parameter penalti.

Secara singkat, metode fungsi penalti interior digunakan untuk menyelesaikan persoalan yang berbentuk :

$$\begin{array}{ll} \text{Optimasikan} & : F(X) \quad X \in R^n \\ \text{dengan kendala:} & g_j(X) \geq 0 \quad j = 1, 2, \dots, m \end{array} \quad (1.1)$$

Persoalan optimisasi yang akan diselesaikan di sini adalah persoalan meminimumkan fungsi $F(X)$. Untuk persoalan-persoalan memaksimumkan fungsi $F(X)$ dapat diselesaikan dengan mengubahnya terlebih dahulu menjadi persoalan minimisasi, yaitu dengan mengambil negatif dari $F(X)$. Sehingga persoalan (1.1) menjadi

$$\begin{array}{ll} \text{Minimumkan} & : F(X) \quad X \in R^n \\ \text{dengan kendala} & : g_j(X) \geq 0 \quad j = 1, 2, \dots, m \end{array} \quad (1.2)$$

Persoalan (1.2) kemudian diubah menjadi persoalan optimisasi tanpa kendala berbentuk:

$$\begin{aligned} \text{Minimumkan : } \phi(X, r) &= F(X) + r P(g_j(X)) \\ j &= 1, 2, \dots, m \end{aligned} \quad (1.3)$$

dengan r parameter penalti dan $P(g_j(X))$ fungsi penalti.

Dari keterangan di atas maka ada tiga hal yang mempengaruhi efisiensi metode fungsi penalti interior ini. Yang pertama adalah pemilihan metode untuk menyelesaikan barisan optimisasi tanpa kendala. Yang kedua terletak pada pemilihan parameter penalti r dan terakhir adalah pemilihan bentuk fungsi penalti $P(g_j(X))$.

Pada tugas akhir ini hanya akan ditinjau pengaruh parameter penalti terhadap efisiensi metode fungsi penalti interior. Peninjauan dilakukan berdasarkan pada beberapa perubahan nilai parameter penalti awal pada barisan persoalan optimisasi tanpa kendala. Penulis ingin mengetahui bagaimana pengaruh dari beberapa rumusan parameter penalti yang ada terhadap kecepatan konvergensi dari metode fungsi penalti interior, ditinjau dari waktu proses (*computing time*) yang diperlukan.

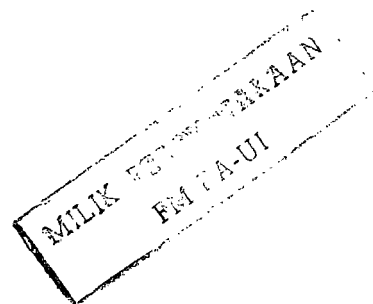
Tujuan dari penulisan tugas akhir ini adalah untuk memperoleh suatu cara mempercepat konvergensi dari metode fungsi penalti interior melalui perubahan nilai parameter

penalti awal yang digunakan. Tentunya tanpa meninggalkan segi keakuratan dari hasil yang diperoleh. Untuk menghilangkan hal-hal yang dapat mengganggu pengaruh parameter penalti terhadap metode fungsi penalti interior, maka untuk setiap kasus diselesaikan dengan metode yang sama.

Penulisan tugas akhir ini penulis bagi menjadi lima bab. Bab pertama berisi pendahuluan, masalah serta tujuan dari penulisan tugas akhir ini. Pada bab ini juga diberikan sistematika pembahasannya.

Pada bab dua diberikan pembahasan mengenai metode optimisasi tanpa kendala yang akan digunakan untuk mencari penyelesaian dari fungsi $\phi(X, r)$. Adapun metode yang dibahas adalah metode Davidon-Fletcher-Powell dan metode interpolasi kubik. Pembahasan mengenai metode fungsi penalti interiornya sendiri diberikan pada bab berikutnya.

Analisis terhadap hasil perhitungan diberikan pada bab selanjutnya. Untuk memudahkan analisis, hasil yang diperoleh untuk masing-masing rumusan parameter penalti diberikan dalam bentuk tabel. Dan di akhir pembahasan, penulis mencoba menarik kesimpulan umum dari analisis yang telah dilakukan.



BAB II

OPTIMISASI TANPA KENDALA

Pada metode fungsi penalti interior, persoalan optimisasi dengan kendala diselesaikan melalui barisan persoalan optimisasi tanpa kendala. Untuk itu pada bab dua ini akan dibahas metode yang akan digunakan untuk mencari penyelesaian dari persoalan optimisasi tanpa kendala tersebut.

Ada beberapa metode iterasi yang dapat digunakan untuk mencari penyelesaian optimum dari masalah optimisasi tanpa kendala. Tetapi pada tugas akhir ini metode yang akan dibahas hanya metode Davidon-Fletcher-Powell.

Pandang persoalan meminimumkan fungsi obyektif

$$\text{Minimumkan: } z = F(X), \quad X \in R^n \quad (2.1)$$

Misalkan $F(X)$ mempunyai titik minimum di sekitar $X = X_0$. Jika $F(X)$ kontinu dan mempunyai turunan parsial tingkat satu dan tingkat dua yang juga kontinu, maka $F(X)$ dapat dihampiri oleh fungsi

$$\begin{aligned} \phi(X) = & F(X_0) + (X - X_0)^T \nabla F(X_0) + \\ & \frac{1}{2} (X - X_0)^T H_f(X_0) (X - X_0) \end{aligned} \quad (2.2)$$

dengan $H_f(X_0)$ matriks Hessian dari $F(X)$ di titik $X = X_0$. $\phi(X)$ akan minimum apabila $\phi'(X) = 0$, atau

$$\nabla F(X_0) + H_f(X_0) (X - X_0) = 0 \quad (2.3)$$

Sehingga titik minimum dari $\phi(x)$ adalah

$$X_m = X_0 - H_f^{-1}(X_0) \nabla F(X_0) \quad (2.4)$$

Dengan memodifikasi persamaan (2.4) diperoleh hampiran berikutnya dari suatu titik $X = X_i$ ke penyelesaian optimum dari persoalan (2.1), yaitu

$$X_{i+1} = X_i - \lambda_i H_f^{-1}(X_i) \nabla F(X_i) \quad (2.5)$$

Di sini λ_i adalah suatu skalar yang ditentukan melalui proses pencarian sepanjang arah $H_f^{-1}(X_i) \nabla F(X_i)$.

Jika λ_i dicari sepanjang arah $-H_f^{-1}(X_i) \nabla F(X_i)$, maka persamaan (2.5) dapat ditulis sebagai

$$X_{i+1} = X_i + \lambda_i H_f^{-1}(X_i) \nabla F(X_i) \quad (2.6)$$

II.1 Metode Davidon-Fletcher-Powell

Metode Davidon-Fletcher-Powell didasarkan pada persamaan (2.5) di atas. Hanya saja metode ini menghindari perhitungan H_f^{-1} . Sebagai gantinya H_f^{-1} didekati dengan matriks simetris definit positif H yang selalu diperbaharui (*update*) pada setiap tahap iterasinya, sehingga

$$X_{i+1} = X_i + \lambda_i H_i \nabla F(X_i) \quad (2.7)$$

Dengan H_i matriks simetris definit positif yang selalu diperbaharui dengan menggunakan rumus rekurensi

$$H_{i+1} = H_i + \lambda_i \frac{(d_i)(d_i)^T}{(d_i)^T(Q_i)} + \frac{(H_i Q_i)(H_i Q_i)^T}{(Q_i)^T(H_i)(Q_i)} \quad (2.8)$$

dengan

$$d_i = -H_i \nabla F(X_i) \quad (2.9)$$

dan

$$Q_i = \nabla F(X_{i+1}) - \nabla F(X_i) \quad (2.10)$$

Proses iterasi metode Davidon-Fletcher-Powell dimulai dengan pengambilan sembarang titik awal $X_1 \in \mathbb{R}^n$ dan matriks

simetris definit positif H_1 . Biasanya H_1 diambil matriks identitas. Kemudian dicari panjang langkah λ sepanjang arah $-(H_f \nabla F)$. Dengan menggunakan persamaan (2.7) kita tentukan titik berikutnya. Lakukan uji konvergensi untuk menentukan apakah titik tersebut sudah mencapai pada penyelesaian sebenarnya dari persoalan (2.1). Proses iterasi kita hentikan apabila titik tersebut sudah memenuhi kriteria konvergensi yang diberikan. Jika tidak, iterasi kita lanjutkan dengan memperbaharui (*update*) matriks H menggunakan persamaan (2.8).

Beberapa kriteria konvergensi yang dapat digunakan untuk menghentikan proses iterasi dari metode Davidon-Fletcher-Powell adalah :

$$1. \quad \|\nabla F(X_{i+1})\| \leq \epsilon_1 \quad (2.11)$$

$$2. \quad \|X_{i+1} - X_i\| \leq \epsilon_2 \quad (2.12)$$

$$3. \quad \left| \frac{F(X_{i+1}) - F(X_i)}{F(X_{i+1})} \right| \leq \epsilon_3 \quad (2.13)$$

untuk nilai ϵ_1 , ϵ_2 dan ϵ_3 yang sudah ditentukan.

Prosedur iteratifnya diberikan pada algoritma di bawah ini :

Algoritme 2.1

1. Mulai dengan pengambilan sembarang titik awal X_1 dan matriks simetris definit positif H_1
2. $i = 1$
3. Hitung $g_i = \nabla F(X_i)$
4. Hitung $d_i = -H_i g_i$
5. Cari λ_i yang meminimumkan $F(X_i + \lambda d_i)$
6. Hitung $X_{i+1} = X_i + \lambda_i d_i$
7. Lakukan uji konvergensi menggunakan persamaan (2.11) sampai (2.13), jika sudah optimum maka hentikan proses iterasi. Jika tidak teruskan ke langkah berikutnya
8. Perbaharui matriks H_i menggunakan persamaan (2.8)
9. $i = i + 1$; kembali ke langkah 3

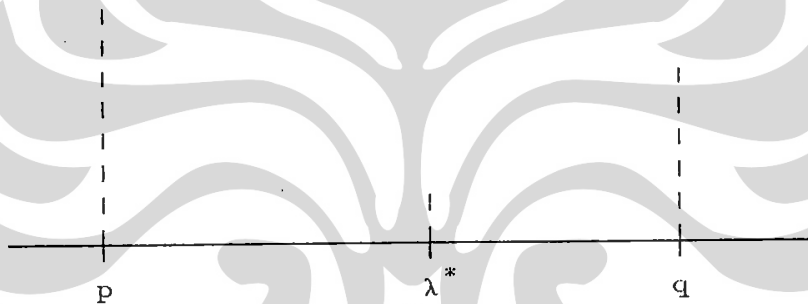
II.2 Metode Interpolasi Kubik

Untuk menentukan hampiran berikutnya dari titik $X = X_i$ pada metode Davidon-Fletcher-Powell, kita perlu menentukan panjang langkah optimum sepanjang arah $d_i = -H_i \nabla F(X_i)$. Pada metode Davidon-Fletcher-Powell, pendekatan yang digunakan untuk mencari panjang langkah optimum itu adalah metode interpolasi kubik.

Misalkan diberikan persoalan meminimumkan fungsi $F(X)$, $X \in R^n$, maka panjang langkah optimum sepanjang arah d_i adalah λ yang meminimumkan fungsi $F(X + \lambda d)$. Sehingga pencarian λ dapat dianggap sebagai masalah meminimumkan fungsi obyektif

$$\text{Minimumkan: } \phi(\lambda) = F(X + \lambda d) \quad (2.14)$$

Pada metode interpolasi kubik, fungsi $\phi(\lambda)$ dihampiri oleh suatu fungsi polinomial kubik. Fungsi polinomial kubik ini diperoleh dengan interpolasi yang menggunakan



Gambar 2.1

nilai-nilai fungsi $\phi(\lambda)$ dan gradiennya pada dua titik yang berbeda. Kedua titik tersebut dipilih yang memuat titik minimum dari $\phi(\lambda)$ (gambar 2.1)

Sekarang, misalkan

$$X = [x_1, x_2, \dots, x_n]^T \quad (2.15)$$

dan

$$d = [d_1, d_2, \dots, d_n]^T \quad (2.16)$$

maka

$$\begin{aligned} \phi(\lambda) &= F(X + \lambda d) \\ &= F(x_1 + \lambda d_1, x_2 + \lambda d_2, \dots, x_n + \lambda d_n) \\ \frac{d\phi}{d\lambda} &= \frac{\partial F}{\partial x_1}(X + \lambda d)d_1 + \frac{\partial F}{\partial x_2}(X + \lambda d)d_2 + \\ &\dots + \frac{\partial F}{\partial x_n}(X + \lambda d)d_n \\ &= [\nabla F(X + \lambda d)]^T d \end{aligned} \quad (2.17)$$

Misalkan diketahui

$$\begin{aligned} \phi(p) &= \phi_p = F(X + pd), \quad \phi'(p) = G_p = [\nabla F(X + pd)]^T d \\ \phi(q) &= \phi_q = F(X + qd), \quad \phi'(q) = G_q = [\nabla F(X + qd)]^T d \end{aligned} \quad (2.18)$$

Keempat nilai di atas kemudian digunakan untuk membentuk fungsi polinomial kubik untuk menghampiri fungsi $\phi(\lambda)$, yaitu

$$h(\lambda) = a + b\lambda + c\lambda^2 + d\lambda^3 \quad (2.19)$$

Untuk menentukan nilai-nilai a , b , c dan d pada persamaan di atas, kita substitusikan nilai-nilai pada

persamaan (2.8) ke dalam persamaan (2.19). Diperoleh

$$\left. \begin{aligned} \phi_p &= a + bp + cp^2 + dp^3 \\ \phi_q &= a + bq + cq^2 + dq^3 \\ G_p &= b + 2cp + 3dp^2 \\ G_q &= b + 2cq + 3dq^2 \end{aligned} \right\} \quad (2.20)$$

Penyelesaian dari sistem persamaan di atas memberikan (Rao, S.S., 1984)

$$a = \phi_p - bp - cp^2 - dp^3 \quad (2.21)$$

$$b = \frac{1}{(p - q)^2} (q^2 G_p + p^2 G_q + 2pqz) \quad (2.22)$$

$$c = -\frac{1}{(p - q)^2} \{ (p + q)z + qG_p + pG_q \} \quad (2.23)$$

$$d = \frac{1}{3(p - q)^2} (2z + G_p + G_q) \quad (2.24)$$

dengan

$$z = \frac{3(\phi_p - \phi_q)}{(q - p)} + G_p + G_q \quad (2.25)$$

$h(\lambda)$ akan minimum apabila $h'(\lambda) = 0$, atau

$$b + 2c\lambda + 3d\lambda^2 = 0 \quad (2.26)$$

sehingga nilai λ , yang memberikan $h(\lambda)$ minimum adalah

$$\lambda_{12}^* = \frac{-2c \pm \sqrt{4c^2 - 12bd}}{6d} \quad (2.27)$$

Dengan mensubstitusikan nilai-nilai pada persamaan (2.21) hingga (2.24) diperoleh

$$\lambda_{12}^* = p + \frac{G_p + z \pm w}{G_p + G_q + 2z} (q - p) \quad (2.28)$$

dengan

$$w = \sqrt{z^2 - G_p G_q} \quad (2.29)$$

$w \geq 0$, karena $G_p G_q \leq 0$. Dengan demikian akan kita dapatkan dua nilai λ^* nyata yang berbeda. Salah satunya merupakan titik minimum dari fungsi $h(\lambda)$. Untuk itu perlu diperiksa syarat cukup agar $h(\lambda^*)$ minimum. Syarat cukup agar $h(\lambda^*)$ minimum adalah $h''(\lambda^*) > 0$, atau

$$2c + 6d\lambda^* > 0 \quad (2.29)$$

Substitusikan nilai-nilai pada persamaan (2.23), (2.24) dan persamaan (2.27) ke dalam persamaan (2.29) di atas diperoleh

$$\frac{w}{q-p} > 0$$

(2.30)

Biasanya dalam praktek diambil nilai $p = 0$, kemudian dilakukan pencarian nilai q sedemikian sehingga interval $[p, q]$ memuat titik minimum. Untuk menentukannya dapat digunakan metode pencarian linier (*linear search*).

Mula-mula kita tentukan q awal. Ini memang bukan hal yang mudah, karena tidak ada cara yang terbaik untuk setiap persoalan. Pada tugas akhir ini nilai q awal diambil sama dengan $1/1.05$. Hal yang perlu diperhatikan adalah, jika $G_p < 0$ kita pilih q awal positif, yaitu arah $\phi(\lambda)$ yang turun. Jika tidak demikian, dipilih q awal negatif.

Selanjutnya kita periksa apakah interval $[p, q]$ sudah memuat titik minimum atau belum, yaitu dengan menguji apakah $G_p \times G_q < 0$ atau $\phi_q > \phi_p$. Bila kedua kondisi ini belum terpenuhi, maka q diperbesar dua kali. Proses ini terus dilakukan hingga salah satu dari dua kondisi tersebut terpenuhi.

Prosedur iteratif lengkap dari metode interpolasi kubik ini disajikan dalam algoritme di bawah ini.

Algoritme 2.2

1. Tentukan sembarang titik X_0 dan arah d
2. Ambil $p = 0$, kemudian hitung $\phi_p = F(X_0)$ dan

$$G_p = [\nabla F(X_0)]^T d$$

3. Tentukan q sedemikian sehingga interval $[p,q]$ memuat minimum
4. Hitung: $\phi_q = F(X_0 + qd)$ dan
 $G_q = [\nabla F(X_0 + qd)]^T d$
5. Gunakan persamaan (2.27) untuk menentukan λ yang meminimumkan $F(X + \lambda d)$
6. Jika $|G_\lambda| = |[\nabla F(X_0 + \lambda d)]^T d| \leq \epsilon$, hentikan proses iterasi, jika tidak teruskan ke langkah berikutnya
7. perkecil interval $[p,q]$; kembali ke langkah 4



BAB III

METODE FUNGSI PENALTI INTERIOR

Metode Fungsi Penalti Interior merupakan salah satu metode yang digunakan untuk mencari penyelesaian optimum dari persoalan program non linier berbentuk:

Optimisasikan : $F(X)$, $X \in R^n$

Dengan kendala : $g_j(X) \geq 0$, $j = 1, 2, \dots, m$ (3.1)

Dengan metode ini, persoalan (3.1) diselesaikan melalui pemecahan barisan persoalan optimisasi tanpa kendala. Barisan persoalan optimisasi tanpa kendala tersebut dibentuk dengan menambahkan suatu suku penalti atau hukuman pada fungsi obyektifnya. Suku penalti tersebut dipilih sedemikian sehingga nilainya kecil pada titik-titik di dalam daerah layak dari (3.1) dan menjadi besar menuju tak hingga pada titik-titik batas atau di sekitar batas dari daerah layak persoalan (3.1).

Disebut metode interior karena penyelesaian-penyelesaian dari masalah optimisasi tanpa kendala yang dibentuk terletak di dalam daerah layak dari (3.1). Barisan penyelesaian tadi kemudian akan konvergen ke arah penyelesaian optimum dari persoalan (3.1) juga dari dalam daerah layak dari (3.1).

Berikut diperkenalkan beberapa definisi yang digunakan pada pembahasan dalam penulisan tugas akhir ini

Daerah layak (*feasible region*) dari persoalan (3.1), kita tulis dengan S , didefinisikan sebagai himpunan semua titik yang memenuhi $g_j(X) \geq 0$, atau

$$S = \{ X \mid g_j(X) \geq 0, j = 1, 2, \dots, m \}$$

Semua titik yang memenuhi $g_j(X) \geq 0 \forall j, j = 1, 2, \dots, m$ disebut titik layak dari persoalan (3.1)

Daerah layak dalam (*interior feasible region*) dari persoalan (3.1), kita tulis dengan $\text{int}(S) = S^\circ$, adalah himpunan semua titik yang memenuhi $g_j(X) > 0$, atau

$$\text{int}(S) = S^\circ = \{ X \mid g_j(X) > 0, j = 1, 2, \dots, m \}$$

Semua titik yang memenuhi $g_j(X) > 0, \forall j, j = 1, 2, \dots, m$ disebut titik dalam (*interior*) dari persoalan (3.1).

Fungsi $F(X)$, $X \in R^n$ dikatakan termasuk dalam kelas C^1 apabila $F(X)$ kontinu dan memiliki turunan parsial tingkat satu yang juga kontinu. Jik $F(X)$ termasuk dalam C^1 dan mempunyai turunan parsial tingkat dua yang juga kontinu maka $F(X)$ dikatakan termasuk dalam kelas C^2

Sekarang pandang kembali persoalan (3.1).

Kita asumsikan bahwa :

1. S° tidak kosong
2. Fungsi $F(X)$ dan $g_j(X)$, $j = 1, 2, \dots, m$ termasuk dalam kelas C^2 .

3. Fungsi $F(X)$ terbatas di bawah pada S , yakni
 $F(X) \geq L_M > -\infty$, dengan L_M adalah batas bawah
 terbesar

4. Himpunan $D = \{ X \mid F(X) \leq K, X \in S \}$ terbatas
 untuk setiap nilai real K yang hingga.

Asumsi 1, 2, dan 3 menjamin adanya penyelesaian dari
 persoalan (3.1), sedang asumsi 4 dibuat untuk menjamin bahwa
 setiap minimum lokal dari persoalan (3.1) terletak pada
 titik-titik yang hingga (Fiacco dan McCormick, 1968).

Atas dasar keempat asumsi di atas, persoalan (3.1)
 dapat diubah menjadi persoalan program non linier berbentuk:

$$\begin{aligned} \text{Minimumkan: } \phi(X, r) &= F(X) + r P(g_j(X)) \\ j &= 1, 2, \dots, m \end{aligned} \quad (3.2)$$

dengan $P(g_j(X))$ fungsi penalti dan r suatu skalar positif
 tidak nol.

Fungsi penalti $P(g_j(X))$ dipilih sedemikian sehingga:

1. P terdefinisi dan terbatas di bawah $\forall g_j(X)$,
 $j = 1, 2, \dots, m$
2. $P \longrightarrow \infty$ apabila $g_j(X) \longrightarrow 0$ untuk suatu j ,
 $j = 1, 2, \dots, m$

Pada tugas akhir ini dipilih bentuk fungsi penalti

$$P(g_j(X)) = \sum_{j=1}^m \frac{1}{g_j(X)} \quad (3.3)$$

Sehingga persoalan (3.2) dapat kita tuliskan sebagai:

$$\text{Minimumkan : } \phi(X, r) = F(X) + r \sum_{j=1}^m \frac{1}{g_j(X)} \quad (3.4)$$

Untuk harga r yang turun, barisan penyelesaian dari persoalan (3.4) akan konvergen ke penyelesaian optimum dari persoalan (3.1) (Fiacco dan McCormick, 1968). Hal ini diterangkan oleh teorema berikut:

Teorema 3.1

Misalkan $x_1^*, x_2^*, \dots, x_k^*$ adalah titik-titik optimum dari fungsi

$$\phi(X, r_k) = F(X) + r_k \sum_{j=1}^m \frac{1}{g_j(X)}$$

untuk nilai-nilai $r_1, r_2, \dots, r_k$ yang monoton turun, maka barisan titik-titik tersebut akan konvergen ke penyelesaian optimum dari persoalan (3.1)

Bukti dari teorema di atas diberikan pada lampiran 1.

Proses iterasi dari metode fungsi penalti interior dimulai dengan pengambilan sembarang titik $X \in S^0$. Ini dimungkinkan karena diasumsikan bahwa S^0 tidak kosong. Proses selanjutnya adalah menentukan nilai awal dari parameter penalti r .

Pemilihan nilai awal parameter penalti r hendaknya tidak terlalu kecil. Memang sepintas terlihat bahwa dengan memilih nilai r awal yang cukup kecil, kita dapat menurunkan jumlah iterasi yang diperlukan untuk mencapai ke penyelesaian yang diinginkan. Tetapi jika r awal dipilih cukup kecil maka suku penalti tidak akan memberikan pengaruh apa-apa pada proses minimisasi fungsi $\phi(X, r)$. Sebaliknya jika r awal dipilih cukup besar akan memperlambat penyelesaian, karena membutuhkan jumlah iterasi yang cukup besar. Untuk itu perlu dipilih nilai r awal yang sesuai untuk setiap persoalan.

Ada beberapa rumusan yang dapat kita gunakan untuk menentukan nilai r awal ini. Rumusan pertama diberikan oleh Fiacco-McCormick (1968), yaitu

$$r_1 = - \frac{[\nabla F(X)]^T [\nabla P(g_j(X))]}{[\nabla P(g_j(X))]^T [\nabla P(g_j(X))]} \quad (3.5)$$

dengan

$$P(g_j(X)) = \sum_{j=1}^m \frac{1}{g_j(X)}$$

Rumusan berikutnya diberikan oleh Rao (1984), yaitu

$$r_1 = \frac{F(X)}{P(g_j(X))} = \frac{F(X)}{\sum_{j=1}^m \frac{1}{g_j(X)}} \quad (3.6)$$

Dan rumusan yang terakhir, merupakan rumusan percobaan yang dicoba oleh penulis, yaitu

$$r_1 = \frac{F(X)}{\sum_{j=1}^m g_j(X)} \quad (3.7)$$

Apabila r awal yang dihasilkan oleh masing-masing rumusan di atas negatif atau nol maka nilainya dapat diganti dengan sebuah bilangan positif sembarang. Pada tugas akhir ini diambil nilai pengganti sama dengan 1. Khusus untuk rumusan (3.5), apabila r awal yang dihasilkan negatif, kita dapat bergerak sepanjang vektor $-\nabla F(X)$ sampai diperoleh nilai r awal yang positif atau sampai penyelesaian dari persoalan (3.4) dicapai.

Setelah nilai r awal ditentukan, proses dilanjutkan dengan membentuk fungsi $\phi(X, r)$ yang diberikan dalam persamaan (3.4). Kemudian dicari titik minimum dari fungsi $\phi(X, r)$ tersebut. Pada tugas akhir ini, metode yang

dipergunakan untuk mencari titik minimum dari fungsi $\phi(X, r)$ adalah metode Davidon-Fletcher-Powell.

Langkah selanjutnya adalah menguji apakah titik minimum yang diperoleh itu sudah konvergen ke penyelesaian optimum dari persoalan (3.1) atau belum. Untuk keperluan tersebut dapat digunakan beberapa kriteria konvergensi sebagai berikut

$$1. \quad \| X_{k+1} - X_k \| \leq \epsilon_1 \quad (3.8)$$

$$2. \quad | F(X_{k+1}) - F(X_k) | \leq \epsilon_2 \quad (3.9)$$

$$3. \quad | F(X_{k+1}) - F_{\min} | \leq \epsilon_3 \quad (3.10)$$

dengan

$$F_{\min} = \frac{c^{1/2} F(X_{k+1}) - F(X_k)}{c^{1/2} - 1} \quad (3.11)$$

$$4. \quad \left| \frac{F(X_{k+1}) - F(X_k)}{F(X_{k+1})} \right| \leq \epsilon_4 \quad (3.12)$$

untuk nilai-nilai $\epsilon_1, \epsilon_2, \epsilon_3$ dan ϵ_4 yang sesuai.

Jika salah satu dari keempat kondisi di atas telah terpenuhi maka proses iterasi kita hentikan. Jika tidak proses kita lanjutkan ke tahap berikutnya.

Proses iterasi kita teruskan dengan menurunkan nilai r . Penurunan nilai r dapat menggunakan rumus rekurensi

$$r_{k+1} = r_k / c \quad (3.13)$$

dengan c bilangan real positif yang lebih besar dari satu. Nilai c ini dapat diambil sembarang. Pada tugas akhir ini diambil nilai $c = 10$.

Setelah itu kita bentuk lagi fungsi $\phi(X, r)$ yang baru. Kemudian dicari titik minimum dari fungsi $\phi(X, r)$ tersebut dengan mengambil titik minimum dari fungsi $\phi(X, r)$ pada iterasi sebelumnya sebagai titik awal. Proses ini dilakukan berulang-ulang hingga dicapai penyelesaian optimum dari persoalan (3.1).

Algoritme dari metode fungsi penalti interior ini secara lengkap diberikan di bawah ini.

Algoritme 3.1

1. Mulai dengan pengambilan sembarang titik awal $X_1 \in S^0$,
 $r_1 > 0$, dan $c > 1$
2. $k = 1$
3. Minimumkan Fungsi $\phi(X_k, r_k) = F(X_k) + r_k \sum_{j=1}^m \frac{1}{g_j(X)}$
 sebut hasilnya dengan X_k^*
4. Lakukan uji konvergensi. Jika sudah optimum maka hentikan proses iterasi, jika tidak maka teruskan ke langkah

berikutnya

$$5. X_{k+1} = X_k^*$$

$$6. r_{k+1} = r_k / c$$

7. $k = k + 1$; kembali ke langkah 3



BAB IV

PERCOBAAN NUMERIK DAN ANALISIS KASUS

IV.1. Struktur Program

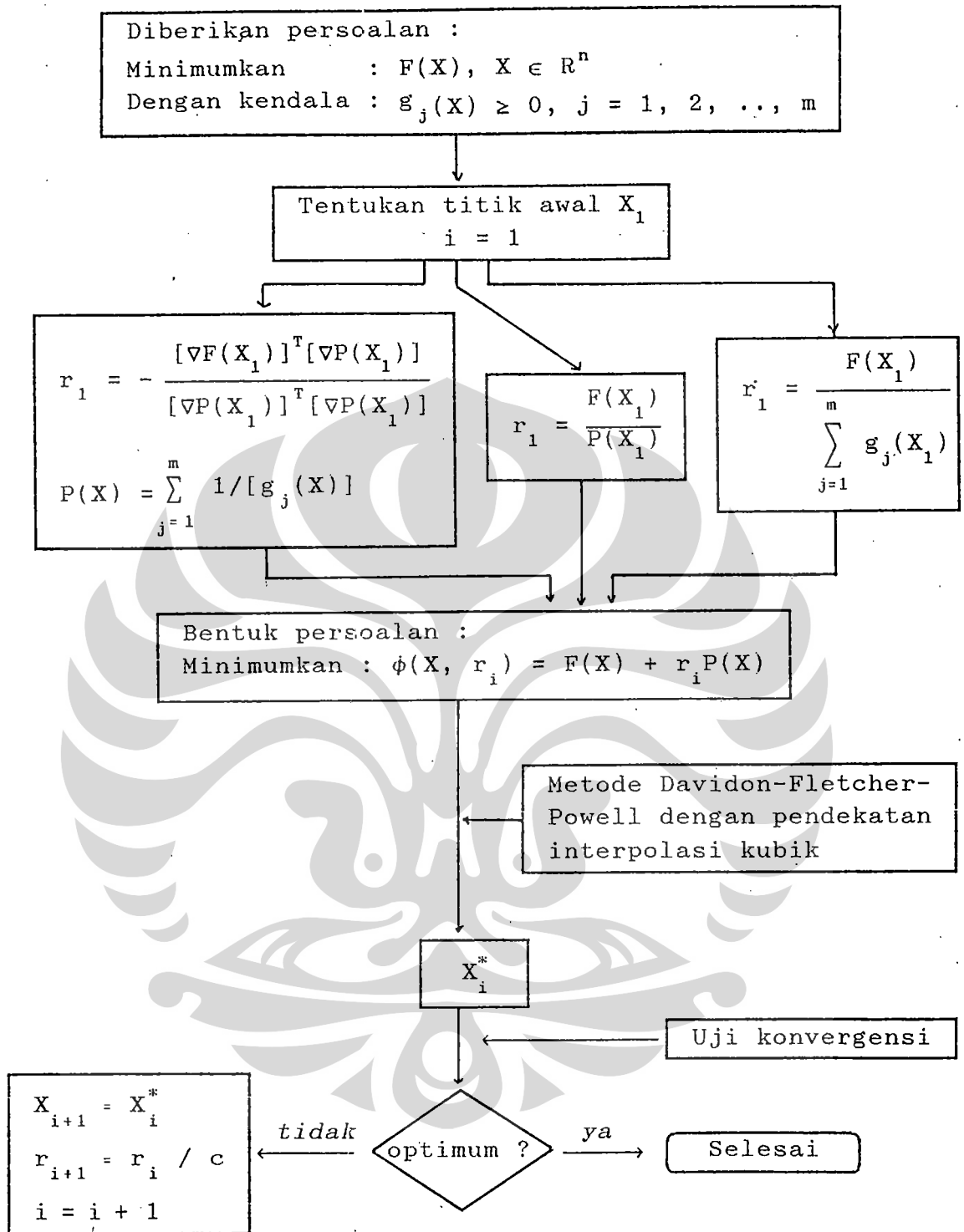
Seperti yang telah disebutkan pada bab tiga, terdapat tiga rumusan yang dapat kita gunakan untuk menentukan nilai parameter penalti awal. Untuk mengetahui bagaimana pengaruh ketiga rumusan tersebut pada kecepatan konvergensi metode fungsi penalti interior, maka dilakukan suatu percobaan numerik. Percobaan dilakukan dengan mencoba ketiga rumusan tersebut pada penyelesaian beberapa kasus. Adapun penyelesaiannya mengikuti diagram yang diberikan pada gambar 4.1.

Dengan menggunakan diagram tersebut, dibuat program sebagai alat bantu untuk menyelesaikan persoalan yang dicobakan. Program dibuat dengan bahasa pemrograman Pascal dan dikompilasi menggunakan kompiler Turbo Pascal versi 5.5. Program lengkapnya diberikan pada bagian lampiran 2.

Subrutin utama yang digunakan dalam program tersebut adalah sebagai berikut :

1. InitAwal

- Fungsi : Membaca jumlah variabel dan jumlah kendala yang digunakan
- Input : Jumlah variabel (n) dan jumlah kendala (m)



Gambar 4.1. Alur penyelesaian program MFPI

2. BacaFungsiObyektif

- Fungsi : Membaca fungsi obyektif
- Input : String fungsi obyektif (fs)

3. BacaFungsiKendala

- Fungsi : Membaca fungsi kendala
- Input : Array string fungsi kendala (cs)

4. BacaTitikAwal

- Fungsi : Membaca titik awal
- Input : Array titik awal (X)

5. Fobj

- Fungsi : Mengevaluasi fungsi obyektif yang sudah dibaca melalui subrutin BacaFungsiObyektif.
- Input : Array titik (X)
- Output : Nilai fungsi obyektif di titik X

6. HitungKendala

- Fungsi : Mengevaluasi fungsi kendala yang sudah dibaca melalui subrutin BacaFungsiKendala
- Input : Array titik (X)
- Output : Nilai fungsi kendala di titik X

7. Dilanggar

- Fungsi : Menguji apakah titik berada di dalam daerah layak dalam dari persoalan yang diberikan atau tidak
- Input : Array titik (X)

- Output : True bila X berada di luar daerah layak,
False bila X berada di dalam daerah layak

8. P

- Fungsi : Menghitung fungsi $P(g_j(X))$
- Input : Array titik (X)
- Output : Nilai fungsi $P(g_j(X))$

9. Phi

- Fungsi : Menghitung fungsi $\phi(X, r)$
- Input : Array titik (X) dan parameter penalti (r)
- Output : Nilai fungsi $\phi(X, r)$

10. HitungGradien

- Fungsi : Menghitung gradien dari fungsi obyektif F
atau fungsi P atau fungsi ϕ di titik X
- Input : String jensi fungsi :
 - 'F' : Fungsi obyektif F
 - 'P' : Fungsi penalti P
 - 'Phi' : Fungsi ϕ
dan array titik (X)
- Output : Array gradien dari fungsi di titik X

11. HitungR_awal

- Fungsi : Menghitung nilai parameter penalti awal
- Input : nomor rumusan (no_r) :
 - 1: Fiacco-McCormick
 - 2: Rao
 - 3: Percobaan

- Output : nilai r awal (r_0)

12. Kubik

- Fungsi : Mencari λ yang meminimumkan fungsi $\phi(X + \lambda D)$ menggunakan metode interpolasi kubik
- Input : Array titik (X) dan array vektor arah (D)
- Output : λ

13. DFP

- Fungsi : Mencari titik minimum dari fungsi $\phi(X, r)$
Subrutin ini merupakan penerapan dari metode Davidon-Fletcher-Powell
- Input : Array titik (X)
- Output : Array titik minimum dari $\phi(X, r) \rightarrow X$

14. PenaltiInterior

- Fungsi : Sebagai subrutin utama untuk menyelesaikan persoalan optimisasi dengan kendala
- Input : -
- Output : Penyelesaian dari persoalan optimisasi dengan kendala

15. UjiKonvergensi

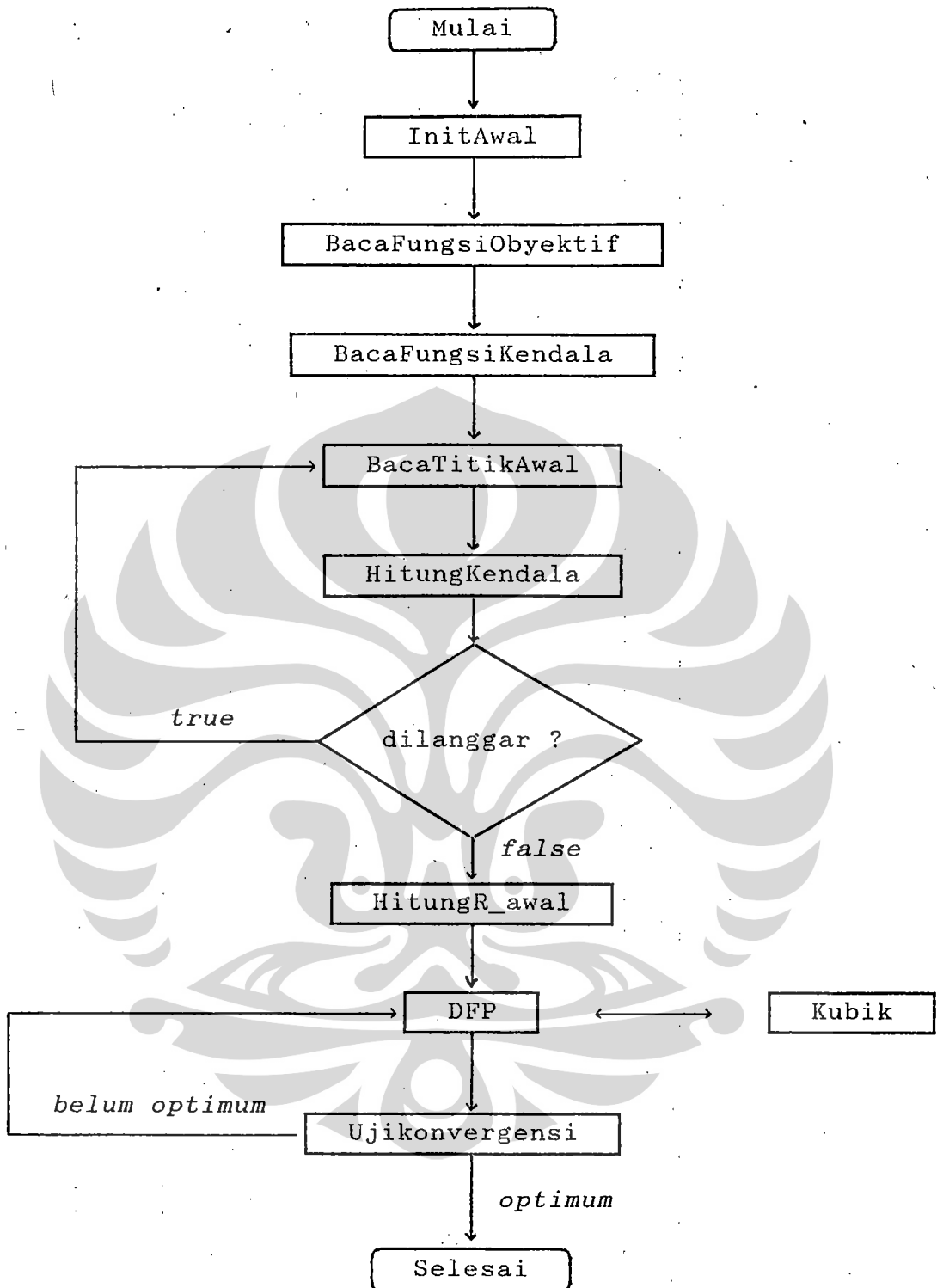
- Fungsi : Menguji konvergensi dari metode fungsi penalti interior
- Input : Array titik (X)
- Output : True bila konvergensi telah tercapai,
False bila konvergensi belum tercapai.

Struktur lengkap dari program tersebut disajikan pada gambar 4.2.

IV.2. Analisis Kasus

Dengan menggunakan program telah dibuat, diselesaikan beberapa persoalan program non linier dengan kendala. Persoalan-persoalan yang diselesaikan dikelompokkan menjadi tiga bagian, yaitu untuk persoalan dengan fungsi obyektif yang berbentuk kuadratik, non kuadratik dan fungsi Rosenbrock. Persoalan dengan fungsi obyektif yang berbentuk kuadratik dibagi lagi menjadi dua bagian, yaitu untuk kendala yang seluruhnya linier dan kendala yang berbentuk umum.

Program dijalankan dengan menggunakan komputer PC AT 286 berkecepatan 21 MHz. Hasil yang diperoleh kemudian dianalisis. Untuk memudahkan penyampaian digunakan istilah "Strategi A" untuk metode fungsi penalti interior yang menggunakan rumusan Fiacco-McCormick, "Strategi B" untuk yang menggunakan rumusan Rao dan "Strategi C" untuk yang menggunakan rumusan percobaan.



Gambar 4.2. Struktur program MFPI

IV.1.1. Fungsi Obyektif Berbentuk Kuadratik

IV.1.1.1. Kendala Yang Seluruhnya Linier

1. Minimumkan : $F(X) = 3x_1^2 + 4x_1x_2 + 5x_2^2$

dengan kendala : $x_1 + x_2 \geq 4$

$$x_1, x_2 \geq 0$$

$$X^* = \begin{pmatrix} 3 \\ 1 \end{pmatrix} \quad F^* = 44$$

a. $X_1 = \begin{pmatrix} 3 \\ 3 \end{pmatrix}$

	Fiacco McCormick	Rao	Percobaan
r_1	99.6918	92.5714	13.50
X_{opt}	$\begin{pmatrix} 3.0054 \\ 0.9946 \end{pmatrix}$	$\begin{pmatrix} 2.9652 \\ 1.0348 \end{pmatrix}$	$\begin{pmatrix} 3.0043 \\ 0.9957 \end{pmatrix}$
F_{opt}	44.0001	44.0048	44.0000
Jumlah iterasi Penalti Interior	6	5	5
Jumlah iterasi DFP	17	13	16
Jumlah refit interp. kubik	63	56	57
Waktu proses (detik)	59	46	55

Tabel 4.1

$$b. X_1 = \begin{bmatrix} 2.5 \\ 2.5 \end{bmatrix}$$

	Fiacco McCormick	Rao	Percobaan
r_1	25.8615	41.6667	12.50
X_{opt}	$\begin{bmatrix} 3.4452 \\ 0.5548 \end{bmatrix}$	$\begin{bmatrix} 2.8870 \\ 1.1130 \end{bmatrix}$	$\begin{bmatrix} 3.0042 \\ 0.9958 \end{bmatrix}$
F_{opt}	44.7927	44.0511	44.0000
Jumlah iterasi Penalti Interior	5	4	5
Jumlah iterasi DFP	13	9	14
Jumlah refit interp. kubik	120	32	57
Waktu proses (detik)	50	35	49

Tabel 4.2

$$c. X_1 = \begin{bmatrix} 4 \\ 4 \end{bmatrix}$$

	Fiacco McCormick	Rao	Percobaan
r_1	383.9999	256	16
X_{opt}	$\begin{bmatrix} 2.8949 \\ 1.1051 \end{bmatrix}$	$\begin{bmatrix} 2.9975 \\ 1.0025 \end{bmatrix}$	$\begin{bmatrix} 3.0000 \\ 0.9990 \end{bmatrix}$
F_{opt}	44.0442	44.0000	44.0000
Jumlah iterasi Penalti Interior	5	6	5
Jumlah iterasi DFP	13	13	15
Jumlah refit interp. kubik	79	53	72
Waktu proses (detik)	47	49	52

Tabel 4.3

Analisis :

Untuk kasus persoalan di atas, terlihat bahwa strategi B memberikan hasil yang cukup baik dengan waktu proses yang lebih kecil bila dibandingkan dengan dua strategi lainnya.

Walaupun untuk titik awal $\begin{bmatrix} 4 \\ 4 \end{bmatrix}$, strategi A membutuhkan waktu yang lebih singkat untuk mencapai ke penyelesaian

dari persoalan yang diberikan, tetapi hasilnya tidak lebih baik dari hasil yang diperoleh dari strategi B.

Dapat kita lihat pula bahwa hasil yang terbaik diberikan oleh strategi C. Waktu yang dibutuhkannya pun tidak begitu jauh dibandingkan dengan waktu yang dibutuhkan oleh strategi B.

$$2. \text{ Minimumkan: } F(X) = 9 - 3x_1 - 6x_2 - 4x_3 + 2x_1^2 + 2x_2^2 + x_3^2 \\ + 2x_1x_2 + 2x_1x_3$$

$$\text{Dengan kendala: } x_1 + x_2 + 2x_3 \leq 3$$

$$x_1, x_2, x_3 \geq 0$$

$$X^* = \begin{bmatrix} 12/9 \\ 7/9 \\ 4/9 \end{bmatrix} \quad F^* = 1/9$$

$$a. X_1 = \begin{bmatrix} 0.1 \\ 0.1 \\ 0.1 \end{bmatrix}$$

	Fiacco McCormick	Rao	Percobaan
r_1	1.0000	0.2399	2.5138
X_{opt}	$\begin{pmatrix} 1.0985 \\ 0.9050 \\ 0.4983 \end{pmatrix}$	$\begin{pmatrix} 1.2448 \\ 0.6881 \\ 0.5335 \end{pmatrix}$	$\begin{pmatrix} 1.0384 \\ 0.8710 \\ 0.5453 \end{pmatrix}$
F_{opt}	0.1716	0.1509	0.1982
Jumlah iterasi. Penalti Interior	5	4	4
Jumlah iterasi DFP	12	10	9
Jumlah refit interp. kubik	236	152	94
Waktu proses (detik)	70	53	46

Tabel 4.4

$$b. X_1 = \begin{bmatrix} 1.0 \\ 0.5 \\ 0.5 \end{bmatrix}$$

	Fiacco McCormick	Rao	Percobaan
r_1	0.3997	0.1701	0.3000
X_{opt}	$\begin{bmatrix} 1.4834 \\ 0.7511 \\ 0.3827 \end{bmatrix}$	$\begin{bmatrix} 1.1738 \\ 0.6923 \\ 0.5670 \end{bmatrix}$	$\begin{bmatrix} 1.3875 \\ 0.8084 \\ 0.4020 \end{bmatrix}$
F_{opt}	0.1349	0.1798	0.1194
Jumlah iterasi Penalti Interior	7	3	5
Jumlah iterasi DFP	15	7	12
Jumlah refit interp. kubik	199	47	250
Waktu proses (detik)	75	35	86

Tabel 4.5

Analisis :

Untuk titik awal $[0.1 \ 0.1 \ 0.1]^T$ hasil yang cukup baik diberikan oleh strategi B. Sedang untuk titik awal $[1 \ 0.5 \ 0.5]^T$ semua strategi memberikan hasil yang cukup baik. Tetapi tampak bahwa jumlah iterasi serta waktu proses yang dibutuhkan oleh strategi B lebih kecil bila dibandingkan dengan dua strategi lainnya.

3. Minimumkan: $E(X) = -x_1^2 - x_2^2$

Dengan kendala: $x_1 + 2x_2 \leq 3$

$x_1, x_2 \geq 0$

$$X^* = \begin{pmatrix} 3 \\ 0 \end{pmatrix}, \quad F^* = -9$$

a. $X_1 = \begin{pmatrix} 2.5 \\ 0.2 \end{pmatrix}$

	Fiacco McCormick	Rao	Percobaan
r_1	0.0138	1.0000	1.0000
X_{opt}	$\begin{pmatrix} 2.9720 \\ 0.0000 \end{pmatrix}$	$\begin{pmatrix} 2.0446 \\ 0.4777 \end{pmatrix}$	$\begin{pmatrix} 2.0446 \\ 0.4777 \end{pmatrix}$
F_{opt}	-8.8328	-4.4085	-4.4085
Jumlah iterasi Penalti Interior	2	3	3
Jumlah iterasi DFP	8	9	9
Jumlah refit interp. kubik	15	17	17
Waktu proses (detik)	39	42	42

Tabel 4.6

$$b. X_1 = \begin{bmatrix} 0.1 \\ 1.4 \end{bmatrix}$$

	Fiacco McCormick	Rao	Percobaan
r_1	0.0014	1.0000	1.0000
X_{opt}	$\begin{bmatrix} 0.1954 \\ 1.4023 \end{bmatrix}$	$\begin{bmatrix} 1.8098 \\ 0.5951 \end{bmatrix}$	$\begin{bmatrix} 1.8098 \\ 0.5951 \end{bmatrix}$
F_{opt}	-2.0046	-3.6296	-3.6296
Jumlah iterasi Penalti Interior	2	3	3
Jumlah iterasi DFP	7	9	9
Jumlah refit interp. kubik	10	18	18
Waktu proses (detik)	35	41	41

Tabel 4.7

Analisis :

Untuk kasus di atas, dengan titik awal $\begin{bmatrix} 2.5 \\ 0.2 \end{bmatrix}$ penyelesaian terbaik diberikan oleh strategi A. Untuk titik awal $\begin{bmatrix} 0.1 \\ 1.4 \end{bmatrix}$ ketiga strategi memberikan hasil yang kurang baik.

4. Minimumkan: $F(X) = -x_1^2 - x_2^2$

Dengan kendala: $x_1 \leq 1$

$$x_1 + 4x_2 \leq 5$$

$$x_1, x_2 \geq 0$$

$$X^* = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad F^* = -2$$

a. $X_1 = \begin{pmatrix} 0.5 \\ 1.0 \end{pmatrix}$

	Fiacco McCormick	Rao	Percobaan
r_1	0.1408	1.0000	1.0000
X_{opt}	$\begin{pmatrix} 0.9999 \\ 0.8654 \end{pmatrix}$	$\begin{pmatrix} 0.3941 \\ 1.1515 \end{pmatrix}$	$\begin{pmatrix} 0.3941 \\ 1.1515 \end{pmatrix}$
F_{opt}	-1.7489	-1.4812	-1.4812
Jumlah iterasi Penalti Interior	2	4	4
Jumlah iterasi DFP	6	10	10
Jumlah refit interp. kubik	70	416	416
Waktu proses (detik)	27	62	62

Tabel 4.8

Analisis :

Sama seperti pada persoalan sebelumnya, strategi A memberikan hasil yang cukup baik bila dibandingkan dengan dua strategi lainnya.

IV.1.1.2. Fungsi Kendala Umum

1. Minimumkan: $F(X) = x_1^2 + x_2^2 + x_3^2$

Dengan kendala: $x_1 + x_2 + x_3 \geq 3$

$$x_1 x_2 x_3 \geq 3$$

$$x_1, x_2, x_3 \geq 0$$

$$X^* = \begin{bmatrix} 1.4422 \\ 1.4422 \\ 1.4422 \end{bmatrix} \quad F^* = 6.2403$$

a. $X_1 = \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix}$

	Fiacco McCormick	Rao	Percobaan
r_1	0.7611	2.5714	1.1250
X_{opt}	$\begin{bmatrix} 1.4471 \\ 1.4471 \\ 1.4327 \end{bmatrix}$	$\begin{bmatrix} 1.4988 \\ 1.4988 \\ 1.1354 \end{bmatrix}$	$\begin{bmatrix} 1.4391 \\ 1.4391 \\ 1.4486 \end{bmatrix}$
F_{opt}	6.2405	6.2763	6.2404
Jumlah iterasi Penalti Interior	4	4	4
Jumlah iterasi DFP	11	10	10
Jumlah refit interp. kubik	220	34	224
Waktu proses (detik)	60	45	57

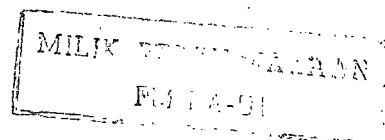
Tabel 4.9

$$b. X_1 = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

	Fiacco McCormick	Rao	Percobaan
r_1	2.3434	5.6000	1.6667
X_{opt}	$\begin{pmatrix} 1.2835 \\ 1.3435 \\ 1.7398 \end{pmatrix}$	$\begin{pmatrix} 1.4489 \\ 1.4507 \\ 1.4272 \end{pmatrix}$	$\begin{pmatrix} 1.1838 \\ 1.6977 \\ 1.4926 \end{pmatrix}$
F_{opt}	6.4791	6.2409	6.5118
Jumlah iterasi Penalti Interior	3	5	3
Jumlah iterasi DFP	7	10	11
Jumlah refit interp. kubik	27	217	48
Waktu proses (detik)	33	58	46

Tabel 4.10

$$c. X_1 = \begin{bmatrix} 4 \\ 4 \\ 4 \end{bmatrix}$$



	Fiacco McCormick	Rao	Percobaan
r_1	101.0794	54.7006	0.5854
X_{opt}	$\begin{bmatrix} 4.0000 \\ 4.0000 \\ 4.0000 \end{bmatrix}$	$\begin{bmatrix} 1.4422 \\ 1.4422 \\ 1.4422 \end{bmatrix}$	$\begin{bmatrix} 1.4422 \\ 1.4422 \\ 1.4422 \end{bmatrix}$
F_{opt}	48.0000	6.2403	6.2403
Jumlah iterasi Penalti Interior	1	5	3
Jumlah iterasi DFP	1	11	9
Jumlah refit interp. kubik	0	47	47
Waktu proses (detik)	30	48	40

Tabel 4.11

Analisis :

Pada kasus di atas, strategi B memberikan hasil yang lebih baik bila dibandingkan dengan dua strategi lainnya. Tetapi jumlah iterasi yang dibutuhkan untuk mencapai ke penyelesaian optimum lebih besar dari waktu yang dibutuhkan oleh strategi C.

Hal yang menarik adalah untuk titik awal yang seluruh komponennya sama, strategi A gagal memberikan penyelesai-

an. Hal ini disebabkan untuk titik-titik tersebut, nilai $\nabla\phi(X, r)$ mendekati nol untuk setiap nilai r awal yang dihasilkan oleh rumusan Fiacco-McCormick.

3. Minimumkan: $F(X) = x_1^2 + x_2^2 + 2x_3^2 + x_4^2 - 5x_1 - 5x_2 - 21x_3 + 7x_4$

Dengan kendala: $2x_1^2 + x_2^2 + x_3^2 + 2x_1 - x_2 - x_4 \leq 5$

$$x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_1 - x_2 + x_3 - x_4 \leq 8$$

$$x_1^2 + 2x_2^2 + x_3^2 + 2x_4^2 - x_1 - x_4 \leq 10$$

$$X^* = \begin{bmatrix} 0 \\ 1 \\ 2 \\ -1 \end{bmatrix} \quad F^* = -44$$

$$a. X_1 = \begin{bmatrix} 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \end{bmatrix}$$

	Fiacco McCormick	Rao	Percobaan
r_1	77.9684	1.0000	1.0000
X_{opt}	$\begin{bmatrix} -0.1356 \\ 0.8863 \\ 2.1081 \\ -0.8914 \end{bmatrix}$	$\begin{bmatrix} 0.1430 \\ 0.7031 \\ 2.1568 \\ -0.2302 \end{bmatrix}$	$\begin{bmatrix} 0.1430 \\ 0.7031 \\ 2.1568 \\ -0.2302 \end{bmatrix}$
F_{opt}	-43.7758	-41.2630	-41.2630
Jumlah iterasi Penalti Interior	6	3	3
Jumlah iterasi DFP	23	13	13
Jumlah refit interp. kubik	292	98	98
Waktu proses (detik)	181	100	100

Tabel 4.12

$$b. X_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

	Fiacco McCormick	Rao	Percobaan
r_1	1.5469	1.0000	1.0000
X_{opt}	$\begin{bmatrix} 0.0976 \\ 1.4268 \\ 1.6267 \\ -1.0736 \end{bmatrix}$	$\begin{bmatrix} 1.0409 \\ 1.2306 \\ 1.0950 \\ -2.3700 \end{bmatrix}$	$\begin{bmatrix} 1.0409 \\ 1.2306 \\ 1.0950 \\ -2.3700 \end{bmatrix}$
F_{opt}	-40.8120	-23.7004	-23.7004
Jumlah iterasi Penalti Interior	2	2	2
Jumlah iterasi DFP	11	8	8
Jumlah refit interp. kubik	39	143	143
Waktu proses (detik)	72	94	94

Tabel 4.13

Analisis :

Untuk setiap titik awal yang diberikan, baik rumusan Rao maupun rumusan percobaan memberikan nilai r_1 sama dengan satu. Sehingga untuk titik awal yang kurang baik, strategi B dan strategi C menghasilkan penyelesaian yang jauh dari penyelesaian yang sebenarnya. Untuk kasus-kasus seperti ini, strategi A memberikan hasil yang cukup dekat dengan penyelesaian yang sebenarnya.

3. Minimumkan: $F(X) = x_1^2 + x_2^2$

Dengan kendala: $x_1 \geq 2$

$$x_1^2 - x_2^2 \leq 1$$

$$X^* = \begin{bmatrix} 2 \\ \sqrt{3} \end{bmatrix} \quad F^* = 7$$

$$a. X_1 = \begin{bmatrix} 2.5 \\ 2.5 \end{bmatrix}$$

	Fiacco McCormick	Rao	Percobaan
r_1	0.7682	4.1667	8.3333
X_{opt}	$\begin{bmatrix} 2.1178 \\ 1.8668 \end{bmatrix}$	$\begin{bmatrix} 2.0000 \\ 1.7811 \end{bmatrix}$	$\begin{bmatrix} 2.0992 \\ 1.8458 \end{bmatrix}$
F_{opt}	7.9701	7.1722	7.8136
Jumlah iterasi Penalti Interior	3	4	4
Jumlah iterasi DFP	17	10	11
Jumlah refit interp. kubik	227	29	34
Waktu proses (detik)	55	37	38

Tabel 4.14

$$b. X_1 = \begin{bmatrix} 4 \\ 4 \end{bmatrix}$$

	Fiacco McCormick	Rao	Percobaan
r_1	0.0160	21.3333	10.6667
X_{opt}	$\begin{bmatrix} 2.1184 \\ 1.8675 \end{bmatrix}$	$\begin{bmatrix} 2.0000 \\ 1.9365 \end{bmatrix}$	$\begin{bmatrix} 2.0000 \\ 1.7355 \end{bmatrix}$
F_{opt}	7.7950	7.7499	7.0120
Jumlah iterasi Penalti Interior	2	5	5
Jumlah iterasi DFP	6	15	16
Jumlah refit interp. kubik	116	134	158
Waktu proses (detik)	24	51	59

Tabel 4.15

Analisis :

Karena kesalahan pembulatan, semua strategi memberikan penyelesaian di sekitar titik minimum yang sebenarnya, Untuk titik awal $\begin{bmatrix} 2.5 & 2.5 \end{bmatrix}^T$ penyelesaian terbaik diberikan oleh strategi B. Sedang untuk titik awal $\begin{bmatrix} 4 & 4 \end{bmatrix}^T$ strategi C memberikan hasil yang mendekati penyelesaian sebenarnya. Untuk titik awal ini strategi A membutuhkan waktu yang lebih singkat bila dibandingkan dengan dua strategi lainnya.

4. Minimumkan: $F(X) = -x_1^2 - x_2^2$

Dengan kendala: $x_1 - x_2 \leq 4$

$$x_2 - \frac{x_1}{3} \leq 4$$

$$10x_1 + 10x_2 - x_1^2 - x_2^2 \leq 41$$

$$X^* = \begin{bmatrix} 12 \\ 8 \end{bmatrix} \quad F^* = -208$$

a. $X_1 = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}$

	Fiacco McCormick	Rao	Percobaan
r_1	1.0000	1.0000	1.0000
X_{opt}	$\begin{bmatrix} 11.0420 \\ 7.6807 \end{bmatrix}$	$\begin{bmatrix} 11.0420 \\ 7.6807 \end{bmatrix}$	$\begin{bmatrix} 11.0420 \\ 7.6807 \end{bmatrix}$
F_{opt}	-180.92	-180.92	-180.92
Jumlah iterasi Penalti Interior	3	3	3
Jumlah iterasi DFP	13	13	13
Jumlah refit interp. kubik	29	29	29
Waktu proses (detik)	91	91	91

Tabel 4.16

$$b. X_1 = \begin{bmatrix} 11.0 \\ 7.5 \end{bmatrix}$$

	Fiacco McCormick	Rao	Percobaan
r_1	0.2789	1.0000	1.0000
X_{opt}	$\begin{bmatrix} 11.9723 \\ 7.9723 \end{bmatrix}$	$\begin{bmatrix} 11.9086 \\ 7.9695 \end{bmatrix}$	$\begin{bmatrix} 11.9086 \\ 7.9695 \end{bmatrix}$
F_{opt}	-206.89	-205.31	-205.31
Jumlah iterasi Penalti Interior	2	2	2
Jumlah iterasi DFP	9	14	14
Jumlah refit interp. kubik	30	22	22
Waktu proses (detik)	60	88	88

Tabel 4.17

Analisis :

Untuk kasus persoalan di atas, ketiga rumusan akan memberikan nilai r_1 sama dengan satu untuk setiap titik awal yang dekat dengan titik awal, sehingga semua strategi akan memberikan hasil yang sama.

Untuk titik awal yang dekat dengan penyelesaian sebenarnya, strategi A memberikan hasil yang lebih baik dan waktu yang lebih singkat dibanding dengan dua strategi lainnya.

IV.1.2. Fungsi Obyektif Non Kuadratik

1. Minimumkan: $F(X) = (x_1 - 1)^4 + (x_2 - 3)^2$

Dengan kendala: $3x_1^2 + 2x_2^2 \leq 21$

$$4x_1 + 5x_2 \leq 20$$

$$x_1, x_2 \geq 0$$

$$X^* = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \quad F^* = 0$$

a. $X_1 = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}$

	Fiacco McCormick	Rao	Percobaan
r_1	1.0000	0.4510	0.2252
X_{opt}	$\begin{bmatrix} 0.8301 \\ 2.9958 \end{bmatrix}$	$\begin{bmatrix} 1.0349 \\ 2.9822 \end{bmatrix}$	$\begin{bmatrix} 0.8640 \\ 2.9989 \end{bmatrix}$
F_{opt}	0.0000	0.0003	0.0003
Jumlah iterasi Penalti Interior	4	5	4
Jumlah iterasi DFP	7	10	5
Jumlah refit interp. kubik	27	48	18
Waktu proses (detik)	25	37	20

Tabel 4.18

$$b. X_1 = \begin{bmatrix} 2.0 \\ 2.0 \end{bmatrix}$$

	Fiacco McCormick	Rao	Percobaan
r_1	1.0000	0.80000	0.2857
X_{opt}	$\begin{bmatrix} 0.8601 \\ 3.0093 \end{bmatrix}$	$\begin{bmatrix} 1.03750 \\ 2.9808 \end{bmatrix}$	$\begin{bmatrix} 0.8367 \\ 3.0039 \end{bmatrix}$
F_{opt}	0.0005	0.0005	0.0007
Jumlah iterasi Penalti Interior	5	5	4
Jumlah iterasi DFP	7	9	5
Jumlah refit interp. kubik	23	50	18
Waktu proses (detik)	27	34	20

Tabel 4.19

Analisis :

Untuk kasus persoalan di atas, semua strategi memberikan hasil yang cukup baik. Dari semuanya, strategi C selesai dalam jumlah iterasi dan waktu proses yang lebih singkat dari dua strategi lainnya.

2. Minimumkan: $F(X) = 4/x_1 + 9/x_2 + (x_1 + x_2)$

Dengan kendala: $x_1 + x_2 \leq 6$

$$x_1, x_2 \geq 0$$

$$X^* = \begin{bmatrix} 2 \\ 3 \end{bmatrix} \quad F^* = 10$$

a. $X_1 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$

	Fiacco McCormick	Rao	Percobaan
r_1	1.0000	6.4286	4.5000
X_{opt}	$\begin{bmatrix} 1.9971 \\ 2.9975 \end{bmatrix}$	$\begin{bmatrix} 2.0035 \\ 2.9823 \end{bmatrix}$	$\begin{bmatrix} 1.9988 \\ 2.9938 \end{bmatrix}$
F_{opt}	10.0000	10.0001	10.0000
Jumlah iterasi Penalti Interior	4	4	4
Jumlah iterasi DFP	4	5	5
Jumlah refit interp. kubik	12	1	16
Waktu proses (detik)	17	19	19

Tabel 4.20

$$b. X_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

	Fiacco McCormick	Rao	Percobaan
r_1	1.0000	6.6667	2.5000
X_{opt}	$\begin{bmatrix} 1.9993 \\ 1.9986 \end{bmatrix}$	$\begin{bmatrix} 1.9975 \\ 2.9992 \end{bmatrix}$	$\begin{bmatrix} 1.9960 \\ 2.9962 \end{bmatrix}$
F_{opt}	10.0000	10.0000	10.0000
Jumlah iterasi Penalti Interior	4	4	4
Jumlah iterasi DFP	5	6	5
Jumlah refit interp. kubik	20	13	13
Waktu proses (detik)	20	24	20

Tabel 4.21

Analisis :

Pada kasus persoalan di atas, semua strategi memeberikan hasil yang hampir sama baik dari penyelesaian maupun dari waktu yang dibutuhkan. Untuk dua titik awal yang diberikan, baik strategi A maupun strategi C membutuhkan jumlah iterasi yang sama untuk mencapai ke penyelesaian optimum. Tetapi kita lihat untuk titik awal $\begin{bmatrix} 0.5 & 0.5 \end{bmatrix}^T$, waktu proses strategi A lebih kecil dari waktu proses strategi C.

3. Minimumkan: $F(X) = -x_1x_2x_3$

Dengan kendala: $0 \leq x_i \leq 42, i = 1, 2, 3$

$$x_1 + 2x_2 + 2x_3 \leq 72$$

$$X^* = \begin{bmatrix} 24 \\ 12 \\ 12 \end{bmatrix} \quad F^* = -3456$$

$$a. X_1 = \begin{bmatrix} 20 \\ 10 \\ 10 \end{bmatrix}$$

	Fiacco McCormick	Rao	Percobaan
r_1	28944.15	1.0000	1.0000
X_{opt}	$\begin{bmatrix} 23.9641 \\ 12.0086 \\ 12.0093 \end{bmatrix}$	$\begin{bmatrix} 23.8337 \\ 12.0416 \\ 12.0416 \end{bmatrix}$	$\begin{bmatrix} 23.8337 \\ 12.0416 \\ 12.0416 \end{bmatrix}$
F_{opt}	-3455.99	-3455.88	-3455.88
Jumlah iterasi Penalti Interior	6	2	2
Jumlah iterasi DFP	25	10	10
Jumlah refit interp. kubik	246	108	108
Waktu proses (detik)	139	73	73

Tabel 4.22

$$c. X_1 = \begin{pmatrix} 30 \\ 10 \\ 10 \end{pmatrix}$$

	Fiacco McCormick	Rao	Percobaan
r_1	584.6997	1.0000	1.0000
X_{opt}	$\begin{pmatrix} 24.1506 \\ 11.9624 \\ 11.9624 \end{pmatrix}$	$\begin{pmatrix} 30.2776 \\ 10.4247 \\ 10.4247 \end{pmatrix}$	$\begin{pmatrix} 30.2776 \\ 10.4247 \\ 10.4247 \end{pmatrix}$
F_{opt}	-3455.90	-3294.12	-3294.12
Jumlah iterasi Penalti Interior	4	3	3
Jumlah iterasi DFP	13	14	14
Jumlah refit interp. kubik	37	27	27
Waktu proses (detik)	81	74	74

Tabel 4.23

Analisis :

Untuk titik awal yang dekat dengan titik minimum, semua strategi memberikan hasil yang cukup baik. Di sini strategi B dan strategi C selesai dalam 2 iterasi dengan waktu proses 73 detik. Sedang strategi A membutuhkan 56 iterasi dengan waktu proses 139 detik.

Untuk titik awal yang lain yang cukup jauh dari titik minimum, strategi B dan stragei C tidak menghasilkan

penyelesaian yang diinginkan. Untuk kasus ini tampak strategi A merupakan pilihan terbaik.

$$3. \text{ Minimumkan: } f(X) = \sum_{i=2}^5 \left[(x_1 - x_i^2)^2 + (x_i - 1)^2 \right]$$

Dengan kendala: $x_i \geq 1, i = 1, 2, 3, 4, 5$

$$X^* = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \quad F^* = 0$$

$$a. X_1 = \begin{pmatrix} 1.1 \\ 1.1 \\ 1.1 \\ 1.1 \\ 1.1 \end{pmatrix}$$

	Fiacco McCormick	Rao	Percobaan
r_1	0.0037	0.0002	0.1768
X_{opt}	$\begin{bmatrix} 1.0642 \\ 1.0291 \\ 1.0291 \\ 1.0291 \\ 1.0291 \end{bmatrix}$	$\begin{bmatrix} 1.0439 \\ 1.0178 \\ 1.0178 \\ 1.0178 \\ 1.0178 \end{bmatrix}$	$\begin{bmatrix} 1.2148 \\ 1.1036 \\ 1.1036 \\ 1.1036 \\ 1.1036 \end{bmatrix}$
F_{opt}	0.0035	0.0015	0.0430
Jumlah iterasi Penalti Interior	6	4	10
Jumlah iterasi DFP	6	5	11
Jumlah refit interp. kubik	13	13	12
Waktu proses (detik)	32	24	57

Tabel 4.24

$$b. X_1 = \begin{bmatrix} 5 \\ 5 \\ 5 \\ 5 \\ 5 \end{bmatrix}$$

	Fiacco McCormick	Rao	Percobaan
r_1	4710.40	1331.20	83.200
X_{opt}	$\begin{bmatrix} 1.8814 \\ 1.3293 \\ 1.3293 \\ 1.3293 \\ 1.3293 \end{bmatrix}$	$\begin{bmatrix} 1.5598 \\ 1.2674 \\ 1.2674 \\ 1.2674 \\ 1.2674 \end{bmatrix}$	$\begin{bmatrix} 1.0001 \\ 1.0001 \\ 1.0001 \\ 1.0001 \\ 1.0001 \end{bmatrix}$
F_{opt}	0.4865	0.2765	0.0000
Jumlah iterasi Penalti Interior	14	14	10
Jumlah iterasi DFP	18	19	16
Jumlah refit interp. kubik	34	37	79
Waktu proses (detik)	89	97	86

Tabel 4.25

$$c. X_1 = \begin{bmatrix} 10 \\ 10 \\ 10 \\ 10 \\ 10 \end{bmatrix}$$

	Fiacco McCormick	Rao	Percobaan
r_1	222782	58903.2	727.2
X_{opt}	$\begin{bmatrix} 189.96 \\ 13.766 \\ 13.766 \\ 13.766 \\ 13.766 \end{bmatrix}$	$\begin{bmatrix} 14.1555 \\ 3.7782 \\ 3.7782 \\ 3.7782 \\ 3.7782 \end{bmatrix}$	$\begin{bmatrix} 1.0128 \\ 1.0055 \\ 1.0055 \\ 1.0055 \\ 1.0055 \end{bmatrix}$
F_{opt}	652.728	30.9309	0.0001
Jumlah iterasi Penalti Interior	14	14	12
Jumlah iterasi DFP	18	15	16
Jumlah refit interp. kubik	34	28	33
Waktu proses (detik)	92	79	81

Tabel 4.26

Analisis :

Untuk titik awal yang dekat dengan titik minimum, baik strategi A maupun strategi B memberikan penyelesaian yang cukup dekat dengan penyelesaian sebenarnya. Tampak bahwa untuk titik awal $[1.1 \ 1.1 \ 1.1 \ 1.1 \ 1.1]^T$ hasil yang cukup baik diberikan oleh strategi B dengan waktu proses yang

lebih singkat dari dua strategi lainnya.

Untuk titik awal yang jauh dari titik minimum, strategi C memberikan hasil yang cukup mendekati penyelesaian yang sebenarnya dengan waktu proses rata-rata lebih singkat dari dua strategi lainnya. Dapat kita lihat bahwa untuk titik awal $\begin{bmatrix} 10 & 10 & 10 & 10 & 10 \end{bmatrix}^T$ strategi A dan strategi B menghasilkan penyelesaian yang jauh dari penyelesaian optimum.

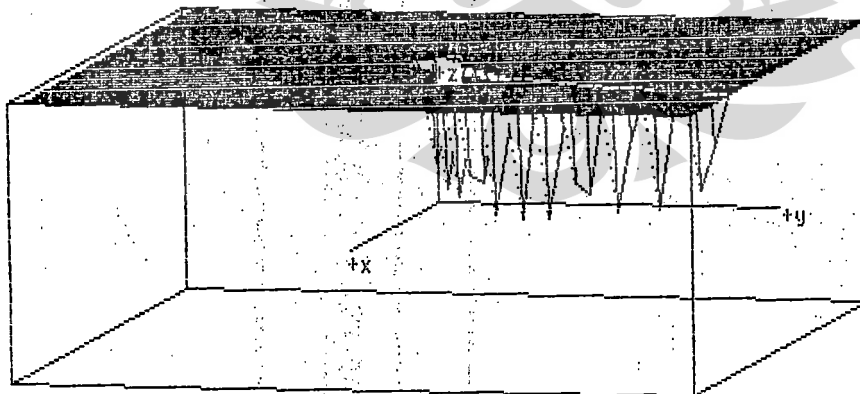
IV.1.3. Fungsi Rosenbrock Dalam Dimensi Dua

Fungsi Rosenbrock dalam dimensi dua didefinisikan sebagai berikut :

$$F(X) = 100(x_2 - x_1^2)^2 + (x_1 - 1)^2$$

Fungsi ini minimum di $X = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ dengan $F^* = 0$. Bentuk grafik

dari fungsi ini dapat dilihat pada gambar 4.3 berikut ini:



Gambar 4.3. Fungsi Rosenbrock dalam dimensi dua

1. Minimisasi fungsi Rosenbrock

Dengan kendala: $x_1, x_2 \geq 0$

a. $X_1 = \begin{bmatrix} 10 \\ 10 \end{bmatrix}$

	Fiacco McCormick	Rao	Percobaan
r_1	1.71E+07	4.05E+06	4.05E+04
X_{opt}	$\begin{bmatrix} 3.85E+01 \\ 1.48E+03 \end{bmatrix}$	$\begin{bmatrix} 4.26E+01 \\ 1.81E+03 \end{bmatrix}$	$\begin{bmatrix} 6.75E+00 \\ 4.55E+01 \end{bmatrix}$
F_{opt}	1.40E+03	1.73E+03	3.30E+01
Jumlah iterasi Penalti Interior	4	3	4
Jumlah iterasi DFP	17	18	8
Jumlah refit interp. kubik	43	47	19
Waktu proses (detik)	50	50	27

Tabel 4.27

$$b. X_1 = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}$$

	Fiacco McCormick	Rao	Percobaan
r_1	6:28E-02	8.10E-02	8.100E+00
X_{opt}	$\begin{bmatrix} 3.70E-05 \\ 1.29E-01 \end{bmatrix}$	$\begin{bmatrix} 1.60E-01 \\ 1.08E-12 \end{bmatrix}$	$\begin{bmatrix} 8.61E+01 \\ 7.41E+03 \end{bmatrix}$
F_{opt}	9.999E-01	7.71E-01	7.24E+03
Jumlah iterasi Penalti Interior	2	2	2
Jumlah iterasi DFP	12	6	43
Jumlah refit interp. kubik	375	254	109
Waktu proses (detik)	46	28	122

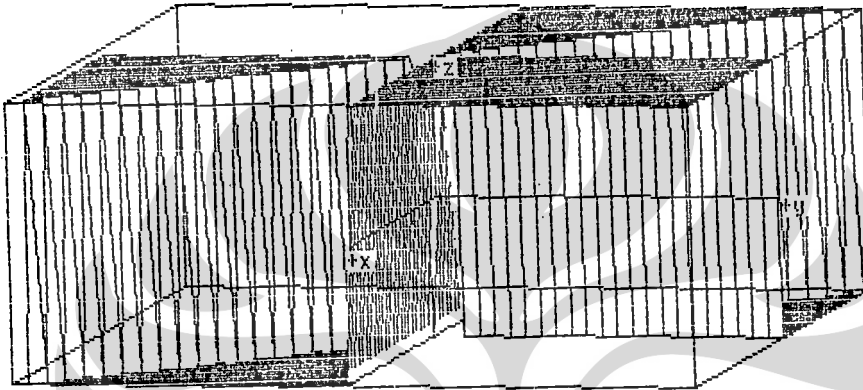
Tabel 4.28

Analisis :

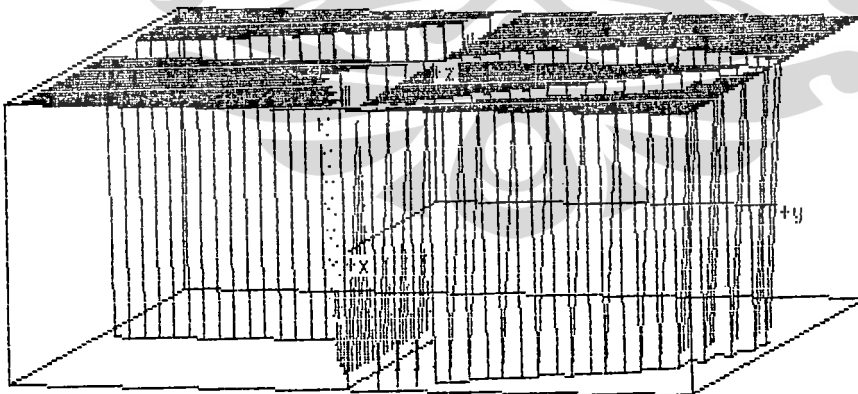
Untuk kendala tersebut di atas, proses minimisasi fungsi Rosenbrock dengan semua strategi gagal memberikan hasil yang diinginkan.

Untuk titik awal $\begin{bmatrix} 10 \\ 10 \end{bmatrix}$, nilai r awal yang dihasilkan oleh semua rumusan cukup besar, sehingga menghasilkan bentuk fungsi $\phi(X, r)$ seperti cerukan lebar yang dalam (gambar 4.4). Akibatnya pada iterasi-iterasi awal, semua

strategi bergerak menjauhi penyelesaian yang diinginkan. Untuk titik awal $\begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}$ strategi A dan strategi B konvergen ke minimum lokal, sedangkan strategi C memberikan penyelesaian yang jauh.



Gambar 4.4. Fungsi ϕ dari persoalan 1 untuk $r = 1E+04$



Gambar 4.5. Fungsi ϕ dari persoalan 3 untuk $r = 1E+03$

2. Minimisasi fungsi Rosenbrock

Dengan kendala: $x_1 \geq 1$

$$x_2 \geq 1$$

a. $X_1 = \begin{bmatrix} 10 \\ 10 \end{bmatrix}$

	Fiacco McCormick	Rao	Percobaan
r_1	1.39E+07	3.65E+06	4.50E+04
X_{opt}	$\begin{bmatrix} 8.61E+01 \\ 7.41E+03 \end{bmatrix}$	$\begin{bmatrix} 1.64E+01 \\ 2.70E+02 \end{bmatrix}$	$\begin{bmatrix} 6.74E+00 \\ 4.54E+01 \end{bmatrix}$
F_{opt}	7.24E+03	2.38E+02	3.29E+01
Jumlah iterasi Penalti Interior	2	5	5
Jumlah iterasi DFP	43	10	10
Jumlah refit interp. kubik	109	25	25
Waktu proses (detik)	112	34	34

Tabel 4.29

$$b. X_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

	Fiacco McCormick	Rao	Percobaan
r_1	6.01E+02	2.01E+02	2.01E+02
X_{opt}	$\begin{bmatrix} 2.33E+00 \\ 5.43E+00 \end{bmatrix}$	$\begin{bmatrix} 2.68E+00 \\ 7.18E+00 \end{bmatrix}$	$\begin{bmatrix} 2.68E+00 \\ 7.18E+00 \end{bmatrix}$
F_{opt}	1.77E+00	2.82E+00	2.82E+00
Jumlah iterasi Penalti Interior	4	4	4
Jumlah iterasi DFP	6	6	6
Jumlah refit interp. kubik	15	15	15
Waktu proses (detik)	22	22	15

Tabel 4.30

Analisis :

Sama halnya dengan kasus pertama, proses minimisasi fungsi Rosenbrock dengan semua strategi gagal memberikan penyelesaian yang diinginkan.

3. Minimumkan fungsi Rosenbrock

Dengan kendala: $x_1 \geq 0$

$$x_2 \geq 0$$

$$x_1 + x_2 \leq 3$$

a. $X_1 = \begin{bmatrix} 0.1 \\ 2.5 \end{bmatrix}$

	Fiacco McCormick	Rao	Percobaan
r_1	1.00E+00	4.81E+01	2.07E+02
X_{opt}	$\begin{bmatrix} 8.22E-01 \\ 6.72E-01 \end{bmatrix}$	$\begin{bmatrix} 9.90E-01 \\ 9.80E-01 \end{bmatrix}$	$\begin{bmatrix} 8.21E-01 \\ 6.75E-01 \end{bmatrix}$
F_{opt}	3.26E-02	9.88E-05	3.28E-02
Jumlah iterasi Penalti Interior	3	3	5
Jumlah iterasi DFP	4	6	6
Jumlah refit interp. kubik	10	21	21
Waktu proses (detik)	16	21	24

Tabel 4.31

$$b. X_1 = \begin{bmatrix} 2.5 \\ 0.1 \end{bmatrix}.$$

	Fiacco McCormick	Rao	Percobaan
r_1	1.00E+00	2.93E+02	1.26E+03
X_{opt}	$\begin{bmatrix} 8.37E-01 \\ 7.01E-01 \end{bmatrix}$	$\begin{bmatrix} 9.32E-01 \\ 8.68E-01 \end{bmatrix}$	$\begin{bmatrix} 9.83E-01 \\ 9.63E-01 \end{bmatrix}$
F_{opt}	2.65E-02	4.65E-03	7.69E-04
Jumlah iterasi Penalti Interior	10	4	4
Jumlah iterasi DFP	11	6	6
Jumlah refit interp. kubik	27	20	22
Waktu proses (detik)	46	23	22

Tabel 4.32

Analisis :

Pada kasus di atas, minimisasi fungsi Rosenbrock dibatasi oleh kendala yang membentuk daerah tertutup. Daerah tertutup ini membatasi penyelesaian dari masing-masing strategi, sehingga hasil yang diberikan mendekati penyelesaian yang diinginkan.

Pada kasus ini, strategi C memberikan hasil terbaik dengan waktu proses yang lebih singkat dibanding dua strategi lainnya.

4. Minimisasi fungsi Rosenbrock

Dengan kendala: $x_1 \geq 0$

$$x_2 \geq 0$$

$$x_1^2 + x_2^2 \leq 4$$

a. $X_1 = \begin{bmatrix} 0.1 \\ 1.8 \end{bmatrix}$

	Fiacco McCormick	Rao	Percobaan
r_1	1.00E+00	2.70E+01	1.21E+02
X_{opt}	$\begin{bmatrix} 7.35E-01 \\ 5.36E-01 \end{bmatrix}$	$\begin{bmatrix} 1.05E+00 \\ 1.11E+00 \end{bmatrix}$	$\begin{bmatrix} 1.24E+00 \\ 1.57E+00 \end{bmatrix}$
F_{opt}	7.17E-02	2.82E-03	1.20E-01
Jumlah iterasi Penalti Interior	3	2	2
Jumlah iterasi DFP	4	3	8
Jumlah refit interp. kubik	11	7	111
Waktu proses (detik)	16	11	35

Tabel 4.33

$$b. X_1 = \begin{bmatrix} 1.5 \\ 0.1 \end{bmatrix}.$$

	Fiacco McCormick	Rao	Percobaan
r_1	1.00E+00	4.11E+01	1.38E+02
X_{opt}	$\begin{bmatrix} 8.11E-01 \\ 6.58E-01 \end{bmatrix}$	$\begin{bmatrix} 1.06E+00 \\ 1.13E+00 \end{bmatrix}$	$\begin{bmatrix} 1.08E+00 \\ 1.16E+00 \end{bmatrix}$
F_{opt}	3.56E-02	4.13E-03	6.30E-03
Jumlah iterasi Penalti Interior	10	3	3
Jumlah iterasi DFP	11	5	3
Jumlah refit interp. kubik	28	17	19
Waktu proses (detik)	46	18	13

Tabel 4.34

Analisis :

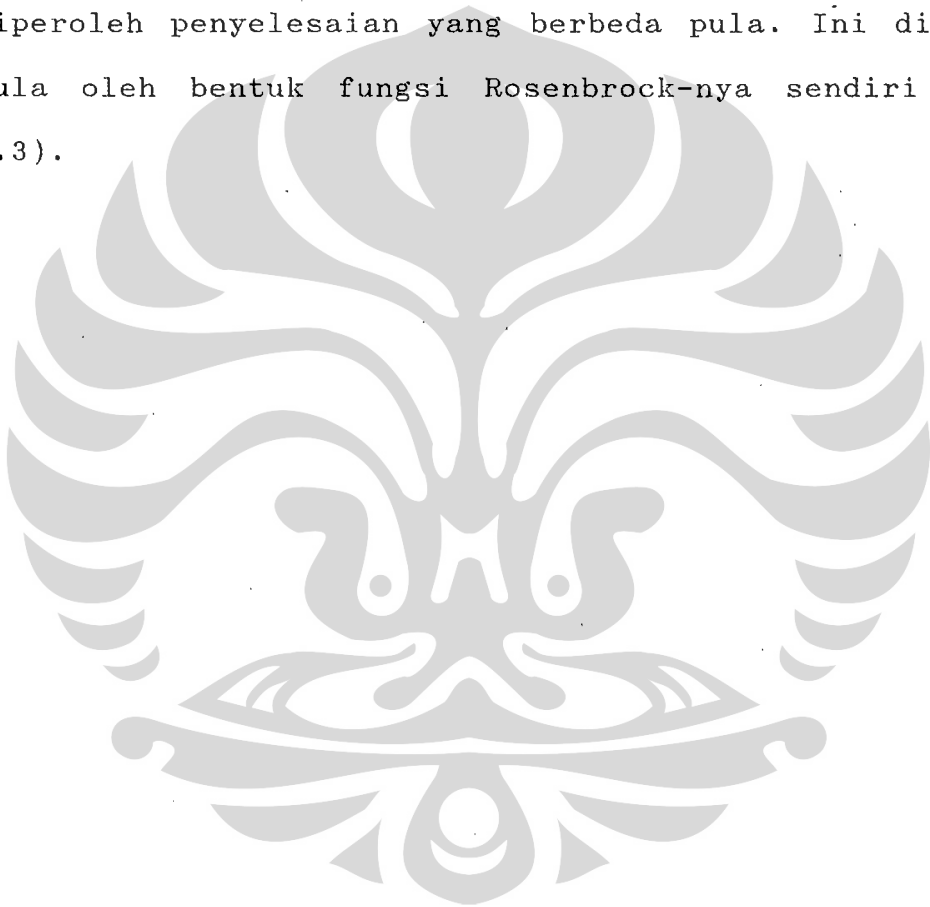
Sama seperti kasus sebelumnya, hasil yang diberikan oleh semua strategi cukup dekat dengan penyelesaian yang sebenarnya.

Untuk dua titik awal yang berbeda, strategi B memberikan hasil yang lebih baik dengan waktu yang lebih singkat dari dua strategi lainnya.

Secara umum, untuk minimisasi fungsi Rosenbrock dengan kendala, penyelesaian yang diberikan akan sangat dipenga-

ruhi oleh fungsi kendalanya.

Untuk fungsi kendala yang membentuk daerah tertutup yang dekat dengan penyelesaian, maka penyelesaian yang dihasilkan cukup dekat pula dengan penyelesaian sebenarnya. Untuk fungsi kendala yang tidak membentuk daerah tertutup, penyelesaian akan konvergen ke minimum lokal. Sehingga untuk setiap titik awal yang diberikan akan diperoleh penyelesaian yang berbeda pula. Ini disebabkan pula oleh bentuk fungsi Rosenbrock-nya sendiri (gambar 4.3).



BAB V

KESIMPULAN

Dari percobaan numerik yang telah dilakukan, hasil-hasil yang diperoleh cenderung menuju ke pola yang sama. Pola tersebut adalah :

1. Untuk kasus-kasus persoalan dimana $F(X_1) \leq 0, \forall X_1 \in S$, seperti pada kasus-kasus IV.1.1.1.3, IV.1.1.1.4, IV.1.1.2.4 dan IV.1.2.3, "Strategi A" memberikan hasil yang lebih baik bila dibandingkan dengan dua strategi lainnya. Ini didasarkan pada waktu proses yang lebih kecil serta penyelesaian yang lebih mendekatati ke penyelesaian sebenarnya dari persoalan yang diberikan.
2. Untuk kasus-kasus lainnya, hasil yang lebih baik diberikan oleh "Strategi B" atau "Strategi C", tergantung dari nilai r awalnya. Hasil-hasil yang diperoleh cenderung menyatakan bahwa penyelesaian yang lebih baik diberikan oleh strategi dengan nilai r awal yang lebih kecil.

Berdasarkan pola tersebut di atas diambil kesimpulan sebagai berikut :

1. Jika $\forall X_1 \in S$ berlaku $F(X_1) \leq 0$ maka sebaiknya rumusan yang digunakan untuk menentukan nilai r awal adalah rumusan Fiacco-McCormick, yaitu :

$$r_0 = - \frac{[\nabla F(X_1)]^T [\nabla P(g_j(X_1))]}{[\nabla P(g_j(X_1))]^T [\nabla P(g_j(X_1))]}$$

dengan

$$P(g_j(X)) = \sum_{j=1}^m \frac{1}{g_j(X)}$$

2. Untuk kasus persoalan yang lain, sebaiknya menggunakan rumusan

$$r_0 = \text{minimum} \left[\frac{F(X_1)}{\sum_{j=1}^m g_j(X_1)}, \frac{F(X_1)}{\sum_{j=1}^m \frac{1}{g_j(X_1)}} \right]$$

untuk menentukan nilai r awal

Kesimpulan tersebut di atas tidak bersifat mutlak adanya. Hal ini disebabkan adanya keterbatasan percobaan seperti terbatasnya jumlah kasus persoalan yang dicobakan serta terbatasnya jumlah titik awal yang dicobakan untuk satu kasus persoalan. Tentunya perlu dilakukan studi lebih lanjut untuk memperbaikinya.



DAFTAR PUSTAKA

1. Avriel, M., *Nonlinear Programming: Analysis and Methods*, Prentice-Hall, Englewood Cliffs, New Jersey, 1976
2. Bazaraa, M.S., Shetty, C.M., *Nonlinear Programming*, John Wiley and Son Inc., New York, 1979
3. Bronson, Richard, *Theory and Problems of Operation Research*, Second Edition, McGraw-Hill Inc., New York, 1982
4. Bunday, Brian D., *Basic Optimization Methods*, Edward Arnold Ltd., London, 1984
5. Davidon, William C., "Variable Metric Method for Optimization", *Siam J. Optimization*, Vol. 1, No. 1, February 1991
6. Fiacco, A.V., McCormick, G.P., *Nonlinear Programming: Sequential Unconstrained Minimization Technique*, John Wiley and Son Inc., New York, N.Y., 1968

7. Goldfarb, Donald., Liu, S., dan Wang, S., "*A Logarithmic Barrier Function Algorithm for Quadratically Constrained Convex Quadratic Programming*", Siam J. Optimization, Vol. 1, No. 2, May 1991
8. Himmelblau, *Aplied Nonlinear Programming*, McGraw-Hill, New York, 1972
9. Jacoby, S.L.S., Kowalik, J.S., Pizzo, J.T., *Iterative Methods for Nonlinear Optimization Problems*, Prentice-Hall, Englewood Cliffs, New Jersey, 1972
10. Rao, S.S., *Optimization: Theory and Applications*, Wiley Eastern Ltd., New Delhi, 1984
11. Zangwill, W.I., "*Nonlinear Programming via Penalty Function*", Man. Sc., Vol. 13, No. 5, 1987

LAMPIRAN 1: TEOREMA KONVERGENSI METODE FUNGSI PENALTI INTERIOR

Pandang kembali persoalan (3.1)

Minimumkan : $F(X)$, $X \in R^n$

dengan kendala : $g_j(X) \geq 0$, $j = 1, 2, \dots, m$

Teorema :

Misalkan X_1^* , X_2^* , $\dots$, X_k^* adalah titik-titik minimum dari fungsi

$$\phi(X, r_k) = F(X) + r_k \sum_{j=1}^m \frac{1}{g_j(X)} \quad (L.1)$$

dan X^* adalah penyelesaian optimum dari persoalan (3.1), untuk nilai-nilai $r_1, r_2, \dots, r_k$ yang monoton turun, maka barisan titik-titik tersebut akan konvergen ke penyelesaian optimum dari persoalan (3.1), atau

$$\lim_{r_k \rightarrow 0} \{ \text{Min } \phi(X, r_k) \} = F(X^*) \quad (L.2)$$

Bukti :

Akan dibuktikan bahwa

$$\lim_{r_k \rightarrow 0} \{ \phi(X_k^*, r_k) \} = F(X^*) \quad (L.3)$$

Sebelumnya akan dibuktikan dulu bahwa untuk $0 < r_k < r_{k-1}$ berlaku

$$\phi(X_{k-1}^*, r_k) < \phi(X_k^*, r_k) \quad (L.4)$$

Dari (L.1) diperoleh :

$$\phi(X_k^*, r_k) = F(X_k^*) + r_k \sum_{j=1}^m \frac{1}{g_j(X_k^*)}$$

$$\phi(X_k^*, r_k) > F(X_k^*) + r_{k-1} \sum_{j=1}^m \frac{1}{g_j(X_k^*)}$$

$$\phi(X_k^*, r_k) > \phi(X_k^*, r_{k-1})$$

Sekarang akan dibuktikan bahwa

$$\lim_{r_k \rightarrow 0} \{ \phi(X_k^*, r_k) \} = F(X^*)$$

Karena $F(X)$ kontinu dan $F(X^*) \leq F(X) \forall X \in S^0$, maka untuk ϵ yang diberikan kita dapat mencari $\tilde{X} \in S^0$ sedemikian sehingga berlaku

$$F(\tilde{X}) < F(X^*) + \frac{\epsilon}{2} \quad (L.5)$$

Karena r_k monoton turun maka kita dapat mencari k sedemikian sehingga untuk $k \geq K$ berlaku

$$r_k \leq \left\{ \frac{\epsilon}{2m} \min_j \left[\frac{1}{g_j(\tilde{X})} \right] \right\} \quad (L.6)$$

Karena $\sum_{j=1}^m g_j(X) \geq 0$, dan dari definisi $\phi(X, r)$, kita peroleh

$$F(X^*) \leq \min \phi(X, r_k) = \phi(X_k^*, r_k) \quad (L.7)$$

untuk $k > K$, $r_k < r_K$ maka berlaku

$$\phi(X_k^*, r_k) \leq \phi(X_K^*, r_K) \quad (L.8)$$

Pertidaksamaan (L.7) berlaku karena X_k^* meminimumkan $\phi(X, r_k)$ dan untuk sebarang nilai X yang lain, khususnya X_k^* , akan

memberikan sebuah nilai yang paling tidak sama dengan $\phi(X_k^*, r_k)$. Selanjutnya dari (L.4) dan (L.8) kita dapatkan

$$F(X^*) \leq \phi(X_k^*, r_k) \leq \phi(X_k^*, r_k) < \phi(X_k^*, r_k) \quad (L.9)$$

$$\phi(X_k^*, r_k) \leq \phi(\tilde{X}, r_k) = F(\tilde{X}) + r_k \sum_{j=1}^m \frac{1}{g_j(\tilde{X})} \quad (L.10)$$

Dari (L.9) dan (L.10) diperoleh

$$F(X^*) \leq \phi(X_k^*, r_k) \leq F(\tilde{X}) + r_k \sum_{j=1}^m \frac{1}{g_j(\tilde{X})} \quad (L.11)$$

Dari (L.6) diperoleh :

$$r_k \sum_{j=1}^m \frac{1}{g_j(\tilde{X})} \leq \frac{\epsilon}{2} \quad (L.12)$$

Substitusikan (L.12) ke dalam (L.11) didapatkan

$$F(X^*) \leq F(\tilde{X}) + \frac{\epsilon}{2} \quad (L.13)$$

Selanjutnya substitusikan (L.5) ke (L.3), diperoleh

$$F(X^*) < F(X^*) + \frac{\epsilon}{2} + \frac{\epsilon}{2} = F(X^*) + \epsilon \quad (L.14)$$

Dengan demikian berlaku

$$F(X^*) \leq \phi(X_k^*, r_k) < F(X^*) + \varepsilon \quad (\text{L.15})$$

sehingga

$$\phi(X_k^*, r_k) - F(X^*) < \varepsilon \quad (\text{L.16})$$

Walaupun ε dapat dipilih sembarang tetapi kita selalu dapat mencari nilai k sedemikian berlaku

$$F(X^*) < \phi(X_k^*, r_k) < F(X^*) + \varepsilon \quad (\text{L.17})$$

dengan demikian untuk $k \longrightarrow \infty$ ($r_k \longrightarrow 0$) diperoleh

$$\lim_{r_k \longrightarrow 0} \{ \phi(X_k^*, r_k) \} = F(X^*) \quad \blacksquare \text{ Terbukti}$$

LAMPIRAN 2. LIST PROGRAM MFPI

A. Cara Menggunakan

Untuk menggunakan program ini, ketik MFPI pada prompt DOS kemudian tekan <Enter>. Sesudah keluar tampilan perkenalan tekan sembarang tombol untuk terus. Untuk seterusnya anda tinggal memilih menu yang tersedia.

Terdapat 5 buah menu, yaitu Utama, Pilihan, Keluaran, Jalankan dan Selesai. Menu Utama digunakan untuk mendefinisikan persoalan baru, mengganti fungsi obyektif dan mengganti fungsi kendala. Menu Pilihan digunakan untuk menentukan rumusan yang digunakan untuk mencari nilai parameter awal r_1 , mengganti nilai galat (kesalahan). Media keluaran yang akan digunakan dapat ditentukan melalui menu Keluaran dan untuk keluar dari program pilihlah menu Selesai.

Ada beberapa cara untuk mengaktifkan menu. Pertama dengan menekan tombol fungsi F1 - F5. Cara yang kedua dengan menekan Alt + Huruf awal dari masing-masing menu. Sebagai contoh, untuk mengaktifkan menu Utama, anda dapat menekan tombol F1 atau dengan menekan tombol Alt+U. Kemudian gunakan tombol panah ke atas dan ke bawah untuk menentukan pilihan yang diinginkan. Setiap pemilihan suatu pilihan selalu diakhiri dengan penekanan tombol <Enter>.

Ada beberapa hal yang perlu diperhatikan dalam penggunaan program ini, yaitu :

1. Jumlah variabel maksimum yang diperbolehkan adalah 10.
2. Jumlah fungsi kendala maksimum yang diperbolehkan adalah 15.
3. Variabel dituliskan dalam bentuk x_i atau X_i , $i = 1, 2, \dots, 10$. Contoh: x_1 atau X_2
4. Fungsi kendala dituliskan dalam bentuk $g_j(X) \geq 0$, sebagai contoh untuk kendala yang berbentuk
$$x_1 + x_2 \leq 0$$
$$x_1, x_2 \geq 0$$
dituliskan dalam bentuk :
$$x_1 + x_2 \quad : \text{kendala ke-1}$$
$$x_1 \quad \quad \quad : \text{kendala ke-2}$$
$$x_2 \quad \quad \quad : \text{kendala ke-3}$$
5. Operasi pangkat dituliskan dengan tanda $^$
Contoh : x_2^2 ditulis x_2^2 atau X_2^2
6. Fungsi matematika yang dapat digunakan adalah :
 $\sin()$, $\cos()$, $\tan()$, $\cot()$, $\sec()$, $\csc()$, $\text{abs}()$,
 $\text{sqrt}()$, $\text{sqr}()$, $\log()$, $\ln()$, $\exp()$, $\sinh()$, $\cosh()$,
 $\tanh()$, $\text{sech}()$, $\text{cosech}()$, $\text{int}()$, $\text{frac}()$
7. Konstanta terdefinisi yang dapat digunakan adalah π dan e

8. Penulisan ekspresi matematika sama seperti penulisan ekspresi matematika pada bahasa pemrograman BASIC ataupun FORTRAN. Sebagai contoh ekspresi

$$\frac{\sin x_1 + \cos x_2}{x_2^2} \text{ dituliskan sebagai}$$

`(sin(x1) + cos(x2))/sqr(x2)` atau

`(sin(x1) + cos(x2))/x2^2`

dan ekspresi $e^{(x_1^2 + x_2^2)}$ dapat dituliskan sebagai

`e^(x1^2 + x2^2)` atau `exp(x1^2 + x2^2)`

B. Contoh Aplikasi

Minimumkan : $F(X) = -x_1 x_2 x_3$

dengan kendala : $0 \leq x_i \leq 42, i = 1, 2, 3$

$$x_1 + 2x_2 + 2x_3 \leq 72$$

Pada kasus ini,

Jumlah variabel = 3

Jumlah kendala = 7

Fungsi obyektif : $-(x_1 * x_2 * x_3)$

Fungsi kendala : 1. x_1

2. x_2

3. x_3

4. $42 - x_1$

5. $42 - x_2$

6. $42 - x_3$

7. $72 - x_1 - 2 * x_2 - 2 * x_3$

C. List Program MFPI

```
{*****}
*
* nama program : mfpas
*
* pembuat      : iwan abu bakar (0387010246)
*
* kompiler     : turbo pascal versi 5.5
*
*****}

uses crt, ifds;

const
    maxelemen = 10;
    maxconstraints = 15;
    eps : array[1..3] of real = (1e-5, 1e-5, 1e-8);

type
    array_string = array[1..maxconstraints] of string;
    array_real = array[1..maxconstraints] of real;

var
    x : vector;           {didefinisikan pada unit ifds}
    m : integer;          {jumlah kendala}
    n : integer;          {jumlah variabel}
    r : real;             {parameter penalti}
    fs : string;          {fungsi obyektif}
    cs : array_string;    {fungsi kendala}

procedure initawal;
begin
    write(' jumlah variabel = '); readln(n);
    write(' jumlah kendala = '); readln(m);
    writeln;
end;

procedure bacafungsiobyektif;
begin
    write(' f(x) = '); readln(fs);
    writeln;
end;
```



```

procedure bacafungsikendala;
var
  i : integer;
begin
  for i := 1 to m do
  begin
    write(' kendala ke - ', i, ' : '); readln(cs[i]);
  end;
  writeln;
end;

function fobj(x : vector) : real ;
begin
  define(fs);
  fobj := f(x, n);
end;

procedure hitungkendala(x : vector; var cv : array_real);
var
  i : integer;
begin
  for i := 1 to m do
  begin
    define(cs[i]);
    cv[i] := f(x, n);
  end;
end;

function dilanggar(x : vector) : boolean;
var
  i, j : integer;
  cv : array_real;
begin
  hitungkendala(x, cv);
  j := 0; i := 0;
  repeat
    i := i + 1;
    if cv[i] <= 0 then j := j + 1;
  until (j > 0) or (i = m);
  if j > 0 then dilanggar := true
  else dilanggar := false;
end;

```

```

procedure bacatitikawal;
var
  i : integer;
  violated : boolean;
begin
  repeat
    for i := 1 to n do
      begin
        write(' x',i,' = '); readln(x[i]);
      end;
      writeln;
      violated := dilanggar(x);
      if violated then
        begin
          writeln(' ada kendala yang dilanggar !!');
          writeln;
        end;
    until not violated;
  end;

function p(x : vector) : real;
var
  cv : array_real;
  ns : real;
  i : integer;
begin
  hitungkendala(x, cv);
  ns := 0;
  for i := 1 to m do
    ns := ns + 1/cv[i];
  p := ns;
end;

function phi(x : vector) : real;
begin
  phi := fobj(x) + r * p(x);
end;

procedure hitunggradien(fs : string; var g, x : vector);
var
  i, id : integer;
  xt, dx, sum : real;

  function fgrad : real;
  begin
    case id of
      1: fgrad := fobj(x);
      2: fgrad := p(x);
    end;
  end;

```

```

        3: fgrad := phi(x);
      end;
    end;

begin
  if fs = 'fobj' then id := 1
  else if fs = 'p' then id := 2
  else id := 3;
  dx := 0.005;
  for i := 1 to n do
    begin
      sum := 0;
      xt := x[i];
      x[i] := xt + dx;
      sum := sum + fgrad;
      x[i] := xt - dx;
      sum := sum - fgrad;
      g[i] := sum / (2 * dx);
      x[i] := xt;
    end;
  end;

function dotproduct(x, y : vector) : real;
var
  i : integer;
  t : real;
begin
  t := 0;
  for i := 1 to n do
    t := t + x[i] * y[i];
  dotproduct := t;
end;

function norm(x : vector) : real;
begin
  norm := sqrt(dotproduct(x, x));
end;

procedure cetakvektor(x : vector; xs : string);
var
  i : integer;
begin
  for i := 1 to n do
    writeln(' ', xs, '(', i, ') = ', x[i]);
  writeln;
end;

```

```

procedure sumvektor(var z, x, y : vector; c : real);
var
    i : integer;
begin
    for i := 1 to n do
        z[i] := x[i] + c * y[i];
    end;

procedure salinvektor(var y, x : vector);
var
    i : integer;
begin
    for i := 1 to n do
        y[i] := x[i];
    end;

procedure tunggu;
begin
    delay(500);
end;

procedure hitungr_awal(no_r : integer);
var
    gf, gp : vector;

    function q(x : vector) : real;
    var
        i : integer;
        t : real;
        v : array_real;
    begin
        t := 0;
        hitungkendala(x, v);
        for i := 1 to m do
            t := t + v[i];
        q := t;
    end;

begin
    case no_r of
        1: begin
            hitunggradien('fobj', gf, x);
            hitunggradien('p', gp, x);
            r := - dotproduct(gf, gp) / dotproduct(gp, gp);
        end;
        2: r := fobj(x) / p(x);
        3: r := fobj(x) / q(x);
    end;
end;

```

```

    if  $\hat{r} \leq 0$  then  $r := 1$ ;
end;

```

```

procedure kubik(var x, g, d : vector; var lambda : real);
var

```

```

    xt                : vector;
    p, q, l, s, z, w, lt : real;
    fp, fq, fl, gp, gg, gl : real;
    search, refit       : integer;
    optimum             : boolean;

```

```

    procedure tukar(x, y : real);

```

```

    var
        t : real;
    begin
        t := x;
        x := y;
        y := t;
    end;

```

```

begin

```

```

    clrscr;
    writeln(' pemakaian interpolasi kubik untuk menentukan lambda ')
    writeln(' ----- ')
    writeln;

```

```

    p := 0;
    fp := phi(x);
    gp := dotproduct(g, d);
    s := abs(2 * fp / gp);
    if s > 1 then s := 1/1.05;
    if gp > 0 then s := -s;

```

```

    writeln(' p = ', p);
    writeln(' gp = ', gp);
    writeln(' s = ', s);
    writeln;
    tunggu;

```

```

    q := s/2;
    l := 0;
    search := 0;
    optimum := false;
    repeat

```

```

        search := search + 1;
        q := 2 * q;

```

```

        writeln(' pencarian ke : ', search);
    until optimum;

```

```

sumvektor(xt, x, d, q);
hitunggradien('phi', g, xt);
fq := phi(xt);
gq := dotproduct(g, d);

writeln(' q = ',q);
writeln(' gq = ',gq);
writeln;

if abs(gq) < 0.001 then
begin
    optimum := true;
    l := q;
end;

if (s * gq > 0) or (fq > fp) then optimum := true
else begin
    p := q;
    fp := fq;
    gp := gq;
end;
until optimum;

if q < p then
begin
    tukar(p, q);
    tukar(fp, fq);
    tukar(gp, gq);
end;

writeln(' interval yang memuat titik minimum : [' ,p ,',',q ,'] ');
writeln;
tunggu;

refit := 0;
if i = 0 then
begin
    lt := 1000;
    optimum := false;
    repeat
        refit := refit + 1;
        z := 3 * (fp - fq) / (q - p) + gp + gq;
        w := sqrt(z * z - gp * gq);
        if w/(q - p) < 0 then w := -w;
        l := p + ((gp + z + w) * (q - p)) / (gp + gq + 2 * z);
        sumvektor(xt, x, d, l);
        hitunggradien('phi', g, xt);
        fl := phi(xt);
        gl := dotproduct(g, d);
    until optimum;
end;

```

```

writeln(' refit ke : ', refit);
writeln;
writeln(' lambda = ', l);
writeln;

if (abs(gl) < 0.001) or (abs(lt - 1) < 1e-3)
then optimum := true
else
begin
  lt := 1;
  if gl < 0 then
  begin
    p := 1;
    fp := fl;
    gp := gl;
  end else
  begin
    q := 1;
    fq := fl;
    gq := gl;
  end;
end;
until optimum;
end;

lambda := 1;

write(' pemakaian interpolasi kubik selesai dalam ', refit);
writeln(' refit ');
writeln;
writeln(' lambda = ', lambda);
writeln;
tunggu;
end;

procedure dfp(var x : vector);
type
  matriks = array[1..maxelemen, 1..maxelemen] of real;
var
  xt, g, d, u, v      : vector;
  h, mm, nn           : matriks;
  i, j, iterasi        : integer;
  optimum, violated    : boolean;
  lambda, vv, vg, ft, eps, nphi, t : real;

procedure sethessianunity;
var
  i, j : integer;

```

```

begin
  for i := 1 to n do
    begin
      for j := 1 to n do
        h[i,j] := 0;
      h[i,i] := 1;
    end;
  end;

  procedure matrikskalivektor(var u, v : vector; a : matriks;
                               c : real);
  var
    i, j : integer;
  begin
    for i := 1 to n do
      begin
        u[i] := 0;
        for j := 1 to n do
          u[i] := u[i] + a[i,j] * v[j];
        u[i] := c * v[i];
      end;
    end;

  procedure vektorkalimatriks(var a : matriks; u,v : vektor;
                               c : real);
  var
    i, j : integer;
  begin
    for i := 1 to n do
      for j := 1 to n do
        a[i,j] := c * u[i] * v[j];
      end;
    end;

  procedure cetakmatriks(a : matriks; as : string);
  var
    i, j : integer;
  begin
    writeln(' matriks ',as);
    writeln;
    for i := 1 to n do
      begin
        for j := 1 to n-1 do
          write(' ',a[i,j]);
        writeln(' ',a[i,n]);
      end;
    writeln;
  end;
end;

```



```

procedure normalkan(var d : vektor);
var
    i : integer;
    x : real;
begin
    x := abs(d[1]);
    for i := 2 to n do
        if abs(d[i]) > x then x := abs(d[i]);
    for i := 1 to n do
        d[i] := d[i] / x;
end;

procedure skalarkalivektor(skalar : real; var u, v : vektor);
var
    i : integer;
begin
    for i := 1 to n do
        u[i] := skalar * v[i];
end;

begin
    clrscr;
    write(' minimisasi fungsi phi dengan metode davidon-');
    writeln('fletcher-powell ');
    write(' -----');
    writeln('----- ');
    writeln;

    optimum := 0;
    eps := 0.0005;
    iterasi := 0;

    sethessian;
    repeat
        iterasi := iterasi + 1;
        nphi := phi(x);
        hitunggradien('phi', g, x);

        writeln(' iterasi ke : ', iterasi);
        writeln;
        cetakvektor(x, 'x');
        tunggu;

        vg := norm(g);
        if abs(vg) < 0.005 then optimum := true else
        begin
            ft := nphi;
            salinvektor(u, g);
            matrikskalivektor(d, g, h, -1);
            normalkan(d);
            cetakvektor(d, 'd');
        end;
    until optimum;
end;

```

```

kubik(x, g, d, lambda);

repeat
    sumvektor(xt, x, d, lambda);
    violated := dilanggar(xt);
    if violated then lambda := lambda / 1.05;
until not violated;

skalarkalivektor(v, d, lambda);
sumvektor(x, x, v, 1);
hitunggradien('phi', g, x);

writeln(' lambda = ', lambda);
writeln;
cetakvektor(x, 'x~');

vv := norm(v);
if abs(vv) < 1e-4 then optimum := true else
begin
    sumvektor(u, g, u, -1);
    t := lambda / dotproduct(d, u);
    vektorkalimatriks(mm, d, d, t);
    matrikskalivektor(xt, u, h, 1);
    t := dotproduct(u, xt);
    vektorkalimatriks(nn, xt, xt, -1/t);

    cetakmatriks(mm, 'm');
    cetakmatriks(nn, 'n');

    for i := 1 to n do
        for j := 1 to n do
            h[i,j] := h[i,j] + mm[i,j] + nn[i,j];
        end;
    end;
    ft := ft - nphi;
end;
until optimum;

writeln(' minimisasi selesai dalam ', iterasi, ' iterasi ');
writeln;
writeln(' dengan hasil sbb : ');
writeln;
cetakvektor(x, 'x');
tunggu;
end;

```

```

procedure penaltiinterior(k : integer);
var
  xt          : vector;
  i           : integer;
  c, f1, f2   : real;
  optimum     : boolean;

  procedure ujikonvergensi(var optimum : boolean;
                           x1, x2 : vector);
  var
    xt : vector;
    px, fmin, fc, nx, ps : real;
  begin
    sumvektor(xt, x2, x1, -1);
    nx := norm(xt);
    px := p(x2);
    if abs(f1) > 0.001 then ps := f1 else ps := 1;
    fc := abs((f2 - f1) / ps);
    if (nx <= eps[1]) or (abs(fc) <= eps[2] or
        (abs(r * px) < eps[3])) then optimum := true
    else optimum := false;
  end;

begin
  clrscr;
  writeln(' pemakaian metode fungsi penalti interior ');
  writeln(' ----- ');
  writeln;

  f1 := fobj(x);
  c := 10;
  hitungr_awal(k);

  i := 0;
  optimum := false;
  repeat
    i := i + 1;
    salinvektor(xt, x);

    writeln(' iterasi ke : ', i);
    writeln;
    writeln(' r = ', r);
    writeln;
    cetakvektor(x, 'x');
    writeln(' f(x) = ', f1);
    writeln;
    tunggu;

    dfp(x);
    f2 := fobj(x);
    ujikonvergensi(optimum, xt, x);
  until optimum;
end;

```

```

    if not optimu then
    begin
        f1 := f2;
        r := r / c;
    end;
until optimum;

clrscr;
write(' proses optimisasi dengan kendala selesai dalam ');
writeln(i, ' iterasi. ');
writeln;
writeln(' dengan hasil sebagai berikut : ');
writeln;
cetakvektor(x, 'x');
writeln(' f(x) = ', fobj(x));
end;

{ ----- bagian program utama ----- }

var
    i      : integer;
    x1, x  : vector;

begin
    clrscr;
    write(' optimisasi dengan kendala melalui metode fungsi');
    writeln(' penalti interior ');
    write(' ----- ');
    writeln(' ----- ');
    writeln;

    initawal;
    bacafungsiobyektif;
    bacafungsikendala;
    bacatitikawal;
    salinvektor(x1, x);
    for i := 1 to 3 do
    begin
        salinvektor(x, x1);
        penaltiinterior(i);
        readln;
    end;
end;

```

```

{ ----- }
{
{   nama program : ifds.pas -> ifds.tpu
{
{   implementasi pendefinisian fungsi secara interaktif pada
{   pascal, dimodifikasi dari ifds.pas (c) 1990 hindra joshua,
{   bandung, mikrodata volume 10 seri 5
{
{   user function : define(fs : string)
{                       f(x : vector; n : integer) : real
{
{ ----- }
}

unit ifds;

{ ----- }
interface
{ ----- }

uses crt, power, maths;

const
    maxelemen = 10 ;

type
    vector = array[1..maxelemen] of real ;

var
    errorfound : boolean ;

    procedure define(buf : string);
    function f(x : vector; n : integer): real ;

{ ----- }
implementation
{ ----- }

const
    idlen      = 10 ; {panjang identifier/pengenal}
    numlen     = 25 ; {panjang konstanta numerik}
    exprlen    = 255 ; {panjang ekspresi fungsi}
    e          = 2.71828182845904523536 ;
    pi         = 3.14159265358979322846 ;

var
    exprbuf, buf : string[exprlen];
    ch           : char ;
    i, cc        : integer ;

```

```

    paramx      : array[1..maxelemen] of real ;
    a           : real ;
    exitsave    : pointer ;

{$f+}
procedure exitunit;
begin
    exitproc := exitsave ;
end;
{$f-}

{ ----- }
{          rutin untuk menangani kesalahan          }
{ ----- }

procedure salah(s : string);
type
    pixel = record
        karakter : char ;
        atribut  : byte ;
    end;
    bufferlayar = array[1..25,1..80] of pixel ;
var
    layar : bufferlayar absolute $b800 : $0000 ;
    layarsimpan : bufferlayar ;
    posX, posY : integer ;
begin
    errorfound := true ;

    posX := wherex ;
    posY := wherey ;

    move(layar,layarsimpan,4000);

    textcolor(yellow);
    textbackground(red);

    gotoxy(1,1); clreol;
    write(' error : ',s);

    repeat until keypressed;

    move(layarsimpan,layar,4000);

    textcolor(white);
    textbackground(black);
    gotoxy(posX,posY);

    exitproc := @exitunit;

```

end;

```
{ ----- }  
{      rutin untuk mengubah ekspresi ke dalam huruf kecil      }  
{ ----- }
```

```
function locase(c : char): char ;  
begin  
    if ch in ['a'..'z'] then locase := chr(ord(ch) + ord(' '))  
    else locase := ch ;  
end;
```

```
{ ----- }  
{      rutin untuk mengambil token berikutnya dalam ekspresi      }  
{ ----- }
```

```
procedure next;  
begin  
    repeat  
        inc(cc);  
        ch := exprbuf[cc];  
    until (ch <> ' ') or (cc = exprlen);  
  
    if cc = exprlen then salah('expression too long ! ');  
    ch := locase(ch);  
end;
```

```
{ ----- }  
{      rutin untuk mengambil bilangan, notasi ilmiah bisa digunakan      }  
{ ----- }
```

```
function evalnumber: real ;  
var  
    num : string[numlen]; x : real; dummy : integer ;  
begin  
    num := '';  
    repeat  
        num := num + ch; next;  
    until not(ch in ['0'..'9']);  
  
    if ch = '.' then begin  
        num := num + ch; next;  
        while ch in ['0'..'9'] do begin  
            num := num + ch; next  
        end;  
    end;  
end;  
  
if ch = 'e' then begin
```

```

    num := num + ch; next;
    if ch in ['+', '-'] then begin
        num := num + ch; next;
    end;
    while ch in ['0'..'9'] do begin
        num := num + ch; next;
    end;
end;
val(num, x, dummy);
evalnumber := x;

if (not (ch in ['^', '*', '/', '+', '-'])) and (not (ch in
[')', ';']))
    then salah('operator expected ');
end;

```

```

{ ----- }
{          rutin untuk menentukan tipe dari identifier          }
{ ----- }

```

```

function lookup(id : string): integer ;
var
    num : string[idlen] ; m, dummy : integer;
begin
    if (id = 'sin') or (id = 'cos') or (id = 'arctan') or
        (id = 'ln') or (id = 'abs') or (id = 'int') or
        (id = 'frac') or (id = 'tan') or (id = 'sec') or
        (id = 'csc') or (id = 'cot') or (id = 'log') or
        (id = 'alog') or (id = 'arcsin') or (id = 'arccos') or
        (id = 'arcsec') or (id = 'arccsc') or (id = 'arccot') or
        (id = 'sinh') or (id = 'cosh') or (id = 'tanh') or
        (id = 'coth') or (id = 'sech') or (id = 'csch') or
        (id = 'exp') or (id = 'sqr') or (id = 'sqrt')
    then lookup := 1

    else if (id = 'e') or (id = 'pi') then lookup := 2
    else if (copy(id,1,1) = 'x') then
        begin
            num := copy(id,2,idlen);
            val(num,m,dummy);
            str(maxelemen, num);
            if m > maxelemen then salah(num + ' elements only ');
            if (not (ch in ['^', '*', '/', '+', '-'])) and (not (ch in
                [')', ';']))
                then salah('operator expected ');
            lookup := 3
        end
    else lookup := 4 ;
end;

```



```
function expression : real ; forward ;
```

```
{ ----- }
{   rutin untuk mengevaluasi pengenalan sesuai dengan tipenya   }
{ ----- }
```

```
function evalident : real ;
```

```
var
```

```
    id : string[idlen]; x : real ;
```

```
    numstr : string[idlen] ; num, dummy : integer ;
```

```
begin
```

```
    id := '';
```

```
    repeat
```

```
        if length(id) < idlen then id := id + ch ;
```

```
    next;
```

```
    if (id = 'x') then
```

```
    begin
```

```
        while ch in ['0'..'9'] do
```

```
        begin
```

```
            id := id + ch ;
```

```
        next ;
```

```
        end;
```

```
    end;
```

```
until not (ch in ['a'..'z']);
```

```
case lookup(id) of
```

```
1 : begin
```

```
    if ch <> '(' then salah('"' expected ');
```

```
    next;
```

```
    x := expression;
```

```
    if ch <> ')' then salah('"' expected ');
```

```
    next;
```

```
    if id = 'sin' then x := sin(x)
```

```
    else if id = 'cos' then x := cos(x)
```

```
    else if id = 'arctan' then x := arctan(x)
```

```
    else if id = 'ln' then x := ln(x)
```

```
    else if id = 'abs' then x := abs(x)
```

```
    else if id = 'frac' then x := frac(x)
```

```
    else if id = 'int' then x := int(x)
```

```
    else if id = 'tan' then x := tan(x)
```

```
    else if id = 'sec' then x := sec(x)
```

```
    else if id = 'csc' then x := csc(x)
```

```
    else if id = 'cot' then x := cot(x)
```

```
    else if id = 'alog' then x := alog(a,x)
```

```
    else if id = 'log' then x := log(x)
```

```
    else if id = 'arcsin' then x := arcsin(x)
```

```
    else if id = 'arccos' then x := arccos(x)
```

```
    else if id = 'arcsec' then x := arcsec(x)
```

```
    else if id = 'arccsc' then x := arccsc(x)
```

```
    else if id = 'arccot' then x := arccot(x)
```

```

        else if id = 'sinh' then x := sinh(x)
        else if id = 'cosh' then x := cosh(x)
        else if id = 'tanh' then x := tanh(x)
        else if id = 'sech' then x := sech(x)
        else if id = 'csch' then x := csch(x)
        else if id = 'coth' then x := coth(x)
        else if id = 'exp' then x := exp(x)
        else if id = 'sqr' then x := sqr(x)
        else if id = 'sqrt' then x := sqrt(x)
    end;
2 : begin
    if id = 'e' then x := e
    else if id = 'pi' then x := pi;
    end;
3 : begin
    numstr := copy(id,2,idlen) ;
    val(numstr, num, dummy);
    x := paramx[num];
    end;
    else salah('unknown identifier');
end;

evalident := x ;
end;

{ ----- }
{      rutin untuk memecah token berdasarkan jenisnya      }
{ ----- }

function element: real ;
begin
    if ch in ['0'..'9'] then element := evalnumber
    else if ch in ['a'..'z'] then element := evalident
    else salah('identifier or number expected ');
end;

{ ----- }
{ bagian dari parser untuk menangani operator + dan - tunggal, }
{      parentisasi, dan token      }
{ ----- }

function primary : real ;
var x : real; i : integer ;
begin
    i := 0;
    while ch in ['+', '-'] do begin
        if ch = '-' then inc(i);
        next;
    end;

```

```

    if ch = '(' then begin
        next;
        x := expression;
        if ch <> ')' then salah('"' expected');
        next;
    end
    else x := element;
    if odd(i) then x := -x;
    primary := x ;
end;

```

```

{ ----- }
{      bagian dari parser untuk menangani operator pangkat      }
{ ----- }

```

```

function factor : real ;
var x,y : real;
begin
    x := primary;
    y := 1;
    if ch = '^' then begin
        next;
        y := factor ;
    end;
    factor := pow(x,y);
end;

```

```

{ ----- }
{      bagian dari parser untuk menandai operator kali dan bagi      }
{ ----- }

```

```

function term : real ;
var
    x,y : real ;
    z : char ;
begin
    x := factor ;
    while ch in ['*', '/'] do begin
        z := ch;
        next;
        y := factor ;
        if z = '*' then
            x := x*y
        else
            x := x/y
    end;
    term := x
end;

```

```

{ ----- }
{ bagian dari parser yang menangani operator jumlah dan kurang }
{ (asosiatif kiri) }
{ ----- }

function expression : real ;
var x,y : real ;
    z : char ;
begin
    x := term;
    while ch in ['+', '-'] do begin
        z := ch; next;
        y := term;
        if z = '+' then
            x := x + y
        else
            x := x - y ;
        end;
        expression := x ;
    end;

{ ----- }
{ rutin utama mengevaluasi fungsi yang sebelumnya sudah didefi- }
{ sikan melalalui rutin define(fs : string) }
{ ----- }

function f(x : vector; n : integer) : real;
begin
    if exprbuf = '' then salah('unable to calculate empty
function');
    for i := 1 to n do
        paramx[i] := x[i];
    cc := 0;
    next;
    f := expression ;
end;

{ ----- }
{ rutin utama untuk mendefinisikan sebuah fungsi secara }
{ interaktif }
{ ----- }

procedure define(buf : string) ;
begin
    if buf = '' then salah('unable to calculate empty function');
    exprbuf := buf;
    exprbuf := exprbuf + ';' ;
end;

```

```

{ ----- }
{                                     }
{                                     }
{ ----- }
begin
    exitsave := exitproc ;
    errorfound := false ;
    exprbuf := '';
end.

{ ----- }
{                                     }
{ nama program : power.pas -> power.tpu }
{                                     }
{ implementasi pemangkatan untuk pascal, diambil dari }
{ "computer language", september 1986, oleh john snader }
{ dengan sedikit modifikasi }
{                                     }
{ user function : pow(x, n : real) }
{                                     }
{ ----- }

unit power;

{ ----- }
interface
{ ----- }

uses crt, dos ;

function pow(x,n : real) : real ;

{ ----- }
implementation
{ ----- }

type
    pixel = record
        karakter : char ;
        atribut   : byte ;
    end;
    bufferlayar = array[1..25,1..80] of pixel ;

var
    layar : bufferlayar absolute $b800 : $0000 ;

```

```

    layarsimpan : bufferlayar ;

procedure change_color_to(fore,back : byte);
begin
    textcolor(fore);
    textbackground(back);
end;

procedure normal_color;
begin
    change_color_to(white,black);
end;

{ ----- }
{          rutin untuk menangani kesalahan          }
{ ----- }

procedure error(s : string);
var
    posx, posy : integer ;

    procedure space(n : byte);
    var
        i : byte ;
    begin
        for i := 1 to n do write(' ');
    end;

begin
    posx := wherex ;
    posy := wherey ;

    move(layar,layarsimpan,4000);

    change_color_to(yellow,red);
    gotoxy(1,1);
    space(80);
    gotoxy(1,1);
    write(' error : ',s);
    repeat until keypressed;

    move(layarsimpan,layar,4000);

    normal_color ;
    normal_color ;

    gotoxy(posx, posy);

```

```

    halt;
end;

```

```

{ -----
{      rutin utama untuk menghitung pangkat dalam pascal
{ -----

```

```

function pow(x,n : real) : real ;

    function pwr(n : integer) : real ;
    begin
        if n = 1 then pwr := x
        else if odd(n) then pwr := sqr(pwr(n div 2)) * x
        else pwr := sqr(pwr(n div 2));
    end;

begin
    if x = 0 then pow := 0
    else if n = 0 then pow := 1
    else if frac(n) <> 0 then begin
        if x <= 0 then error('unable to calculate imaginary
        number ! ')
        else pow := exp(ln(x) * n)
    end
    else if n < 0 then pow := 1/pwr(-trunc(n))
    else pow := pwr(trunc(n));
end;

```

```

{ -----
{
{ nama program : maths.pas  ->  maths.tpu
{
{ implementasi fungsi-fungsi matematika pada pascal
{
{ user function : lihat daftar interface
{
{ -----

```

```

unit maths ;

```

```

{ -----
interface
{ -----

```

```
uses crt;
```

```
function tan(x : real) : real ;
function sec(x : real) : real ;
function csc(x : real) : real ;
function cot(x : real) : real ;
function log(x : real) : real ;
function alog(a,x : real) : real ;
function arcsin(x : real) : real ;
function arccos(x : real) : real ;
function arcsec(x : real) : real ;
function arccsc(x : real) : real ;
function arccot(x : real) : real ;
function sinh(x : real) : real ;
function cosh(x : real) : real ;
function tanh(x : real) : real ;
function sech(x : real) : real ;
function csch(x : real) : real ;
function coth(x : real) : real ;
function arcsinh(x : real) : real ;
function arccosh(x : real) : real ;
function arctanh(x : real) : real ;
function arcsech(x : real) : real ;
function arccsch(x : real) : real ;
function arccoth(x : real) : real ;
```

MILIK PERSERIKSAAN
FMIPA UI

```
{ ----- }
implementation
{ ----- }
```

```
{ ----- }
{      procedure untuk menangani kesalahan      }
{ ----- }
```

```
procedure error;
begin
  writeln; writeln;
  writeln('r+++++++n error v+++++++y ');
  writeln('1                               1 ');
  writeln('1                wrong domain      1 ');
  writeln('1                               1 ');
  writeln('v+n press any key to continue v+n ');
  repeat until keypressed;
  halt;
end;
```



```

{ ----- }
{           user function           }
{ ----- }

```

```

function tan ;
begin
    tan := sin(x)/cos(x);
end;

```

```

function sec ;
begin
    sec := 1/cos(x);
end;

```

```

function csc ;
begin
    csc := 1/sin(x);
end;

```

```

function cot ;
begin
    cot := 1/tan(x);
end;

```

```

function log ;
begin
    if x <= 0 then error else log := ln(x)/ln(10);
end;

```

```

function alog ;
begin
    if (x <= 0) or (a <= 0) or (a = 1) then error else
        alog := log(x)/log(a);
end;

```

```

function arcsin ;
begin
    if (x <= -1) or (x >= 1) then error else
        arcsin := arctan(x/sqrt(- x*x + 1));
end;

```

```

function arccos ;
begin
    if (x <= -1) or (x >= 1) then error else
        arccos := pi/2 - arcsin(x);
end;

```

```

function arccot ;
begin
    arccot := pi/2 - arctan(x);
end;

```

```

function arccsc ;
begin
    if (x >= -1) and (x <= 1) then error else
        arccsc := arcsin(1/x);
end;

function arcsec ;
begin
    if (x >= -1) and (x <= 1) then error else
        arcsec := arccos(1/x);
end;

function sinh ;
begin
    sinh := (exp(x) - exp(-x))/2;
end;

function cosh ;
begin
    cosh := (exp(x) + exp(-x))/2;
end;

function tanh ;
begin
    tanh := (exp(x) - exp(-x))/(exp(x) + exp(-x));
end;

function sech ;
begin
    sech := 2/(exp(x) + exp(-x));
end;

function csch ;
begin
    csch := 2/(exp(x) - exp(-x));
end;

function coth;
begin
    coth := (exp(x) + exp(-x))/(exp(x) - exp(-x));
end;

function arcsinh;
begin
    arcsinh := ln(x + sqrt(x * x + 1));
end;

function arccosh;
begin
    if x < 1 then error
    else arccosh := ln(x + sqrt(x * x - 1));
end;

```

```

end;

function arctanh;
begin
    if (x <= -1) or (x >= 1) then error else
        arctanh := 0.5 * ln((1 + x)/(1 - x));
end;

function arcsech;
begin
    if (x <= 0) or (x > 1) then error else
        arcsech := ln(1/x + sqrt(1/(x * x) - 1));
end;

function arccsch;
begin
    if x = 0 then error
    else arccsch := ln(1/x + sqrt(1/(x * x) + 1));
end;

function arccoth;
begin
    if (x < 1) and (x > -1) then error else
        arccoth := 0.5 * ln((x + 1)/(x - 1));
end;

```