PENGEMBANGAN PROGRAM PECAHAN LINIER DENGAN TRANSFORMASI ALJABAR

Skripsi

Disusun untuk melengkapi syarat-syarat

guna memperoleh gelar Sarjana Sains

BOBBY REYNALDO

3125121983

Program Studi Matematika

Fakultas Matematika dan Ilmu Pengetahuan Alam

Universitas Negeri Jakarta

2017

LEMBAR PERSETUJUAN HASIL SIDANG SKRIPSI

PENGEMBANGAN PROGRAM PECAHAN LINIER DENGAN TRANSFORMASI ALJABAR

Nama : Bobby Reynaldo

No. Registrasi : 3125121983

	Nama	Tanda Tangan	Tanggal
Penanggung Jawab			
Dekan	: Prof. Dr. Suyono, M.Si. NIP. 19671218 199303 1 005		16-02-2017
Wakil Penanggung Jawab			
Wakil Dekan I	: Dr. Muktiningsih, M.Si. NIP. 19640511 198903 2 001		16-02-2017
Ketua	: Dr. Lukita Ambarwati, S.Pd., M.Si. NIP. 19721026 200112 2 001		14-02-2017
Sekretaris	: Ibnu Hadi, M.Si. NIP. 19810718 200801 1 017		14-02-2017
Penguji	: Dr. Makmuri, M.Si. NIP. 19640715 198903 1 006		14-02-2017
Pembimbing I	: Ratna Widyati, S.Si., M.Kom. NIP. 19750925 200212 2 002		14-02-2017
Pembimbing II	: Med Irzal, M.Kom. NIP. 19770615 200312 1 001		14-02-2017

Dinyatakan lulus ujian skripsi tanggal: 10 Februari 2017

ABSTRACT

BOBBY REYNALDO, 3125121983. Approach Linear Fractional Programming Using Algebraic Transformation. Thesis. Faculty of Mathematics and Natural Science Jakarta State University. 2017.

Solving linear fractional programming in this research using algebraic transformation into a linear programming form which more easily defined. These linear programming solved using revised simplex method which is exceed to identify special case. Optimal results have been obtained is transformed back into linear fractional programming form. Also created an application to help the calculations quickly. From the test results for solving linear fractional program with 10 variables, 10 functions constraint, and 10 times iteration, the application is able to complete the whole process around 0.026 seconds.

Keywords : *Optimization*, *Linear Fractional Programming*, *Ratio*, *Transformation*, *Linear Programming*, *Revised Simplex Method*.

ABSTRAK

BOBBY REYNALDO, 3125121983. Pengembangan Program Pecahan Linier dengan Transformasi Aljabar. Skripsi. Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Jakarta. 2017.

Penyelesaian program pecahan linier pada penelitian ini menggunakan transformasi secara aljabar menjadi bentuk program linier yang mudah didefinisikan. Selanjutnya, program linier diselesaikan menggunakan metode simpleks direvisi. Metode simpleks direvisi memiliki kelebihan dalam mengidentifikasi kasus khusus. Hasil optimal yang telah didapat ditransformasikan kembali dalam bentuk program pecahan linier. Juga dibuat sebuah aplikasi penyelesaiannya menggunakan metode tersebut untuk membantu perhitungan dengan cepat. Dari hasil pengujian untuk menyelesaikan permasalahan program pecahan linier dengan 10 variabel, 10 fungsi kendala, dan 10 kali iterasi, aplikasi mampu menyelesaikan seluruh proses hanya sekitar 0.026 detik.

Kata kunci : Optimasi, Program Pecahan Linier, Rasio, Transformasi, Program Linier, Metode Simpleks Direvisi.

PERSEMBAHANKU...

"*I can do all things through Christ who strengthens me.*"
(*Philippians 4:13*)

"*Prestasi adalah apa yang mampu anda lakukan. Motivasi menentukan apa yang anda lakukan. Sikap menentukan seberapa baik anda melakukannya.*"
- Lois Holtz

Skripsi ini kupersembahkan untuk kedua orangtuaku,
kakak-kakakku, serta keluarga besarku
"Terima kasih atas dukungan, doa, serta kasih sayang kalian".

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa atas pertolongan dan penyertaan-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "Penyelesaian Program Pecahan Linier dengan Transformasi Aljabar" yang merupakan salah satu syarat dalam memperoleh gelar Sarjana Jurusan Matematika Universitas Negeri Jakarta.

Terselesaikannya skripsi ini tidak terlepas dari bantuan berbagai pihak yang telah banyak memberikan arahan, bantuan pemikiran, doa, dan semangat. Oleh karena itu, dalam kesempatan ini penulis ingin menyampaikan terima kasih terutama kepada:

1. Ibu Ratna Widyati, S.Si., M.Kom selaku Dosen Pembimbing I dan selaku Pembimbing Akademik serta Bapak Med Irzal, M.Kom. selaku Dosen Pembimbing II, yang selalu bersedia meluangkan waktunya dalam memberikan bimbingan, bantuan, saran, nasehat serta arahan sehingga skripsi ini dapat terselesaikan.

2. Ibu Dr. Lukita Ambarwati. S,Pd., M.Si., selaku Koordinator Program Studi Matematika FMIPA UNJ. Terima kasih atas segala bimbingan, dukungan, dan bantuan yang telah berikan dengan respon yang cepat kepada penulis.

3. Bapak Drs. Mulyono, M.Kom., Bapak M. Eka Suryana, M.Kom., dan seluruh Bapak/Ibu dosen dijurusan Matematika yang tidak dapat saya sebutkan satu persatu, terimakasih atas segala dan pengajaran yang telah diberikan. Serta karyawan/karyawati FMIPA UNJ atas informasi dan bantuan yang diberikan dalam menyelesaikan skripsi ini.

4. Kedua orang tua terkasih, yang selalu mencurahkan kasih sayang dan cinta serta membiayai penulis selama menempuh pendidikan. Serta Kakak-kakakku yang selalu menghibur dan memberikan motivasi.

5. Teman seperjuangan Chrisna, Steven, Elisabeth, Faralita, Harjadi, Deddy, Miqdad, Leny, serta semua teman-teman matematika 2012 UNJ yang selalu sedia membantu dan memberi semangat kepada penulis.

6. Kakak tingkat ka Monik, ka Albert, ka DP, ka Sandy dan kakak-kakak lainnya serta adik-adik tingkat yang membantu dan memberi masukkan selama perkuliahan.

7. PKK dan TKK yang selalu hadir untuk memotivasi, menemani, membimbing dalam perkulaian dan pertumbuhan rohani. Serta Jessica, Yemmima, dan teman-teman PMK lainnya yang membantu menuliskan revisi penulisan.

8. Semua pihak yang membantu penulis yang tidak dapat disebutkan satu per satu.

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna. Masukan dan kritikan akan sangat berarti. Semoga skripsi ini dapat bermanfaat bagi pembaca sekalian.

Jakarta, Februari 2017

Bobby Reynaldo

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Sekarang ini hampir setiap aspek kehidupan terdapat matematika di dalamnya. Bidang optimasi merupakan salah satu terapan matematika yang sering dijumpai dalam kehidupan sehari-hari. Optimasi adalah suatu proses untuk mencapai hasil yang ideal.

Salah satu penemuan besar di dunia matematika dalam bidang optimasi adalah program linier. Program linier adalah program yang mengangkat suatu permasalahan kemudian dibentuk ke dalam pemodelan matematika. Tujuan program linier adalah menyelesaikan permasalahan dengan cara pengambilan keputusan terbaik yang menyangkut pengelolaan sumber daya yang terbatas dengan beberapa syarat tertentu dan memenuhi tujuan tunggal. Bidang optimasi yang dapat diselesaikan dengan metode program linier, antara lain alokasi sumber daya dalam perencanaan produksi, perhitungan kuantitas material yang digunakan dalam teknik industri, dan lain sebagainya.

Pada kenyataannya tidak semua permasalahan dapat diselesaikan secara optimal dengan menggunakan program linier. Program linier jelas tidak dapat menyelesaikan permasalahan bentuk yang non-linier. Keterbatasan tersebut memunculkan metode optimasi lainnya yang tidak terpaku pada permasalahan linier. Program pecahan linier adalah jenis khusus dari masalah program non-linier di mana fungsi tujuan berbentuk rasio. Rasio yang terdapat pada program pecahan linier dibentuk dari dua fungsi tujuan linier dan kendala

yang masih berupa fungsi linier. Isbell dan Marlow (1956) yang pertama kali mengidentifikasi contoh masalah program pecahan linier dan dipecahkan dengan urutan masalah program linier (Pandian & Jayalakshmi, 2013).

Letak kelebihan program pecahan linier adalah pada fungsi tujuan yang berupa rasio. Rasio adalah perbandingan antara dua besaran atau lebih dimana perbandingan harus menggunakan satuan yang sama. Rasio juga dapat diartikan sebagai suatu angka yang dapat menilai kinerja, menilai keefektifan ataupun menunjukkan hubungan antar suatu unsur dengan unsur lainnya. Optimasi dalam bentuk rasio sering ditemukan dalam permasalahan ekonomi, seperti analisis rasio likuiditas, rasio profitabilitas, rasio kepemilikan dan lainnya.

Sekarang ini sudah banyak metode yang dapat menyelesaikan permasalahan program pecahan linier diantaranya metode *convex*, metode transformasi Chaner dan Cooper, pecahan linier dengan fuzzy dan sebagainya. Penulisan ini membahas pendekatan transformasi baru sehingga program pecahan linier dapat ditransformasikan dan diselesaikan kedalam bentuk program linier. Metode ini dipilih karena secara manual program linier lebih mudah diselesaikan dibandingkan menggunakan metode lainnya.

Penelitian sebelumnya mengenai program pecahan linier telah dilakukan oleh Zuhanda (2013) yang membahas tentang fungsi tujuan berkoefisien interval. Pada penelitiannya, penyelesaian program pecahan linier menggunakan transformasi untuk membantu proses perhitungan. Transformasi yang digunakan ialah transformasi Charnes dan Cooper. Skripsi ini akan membahas transformasi baru program pecahan linier secara aljabar menjadi bentuk program linier yang lebih mudah didefinisikan, sehingga penyelesaian program pecahan linier dapat diselesaikan dalam bentuk program linier.

1.2 Perumusan Masalah

Perumusan masalah yang akan dikaji adalah:

1. Bagaimana transformasi program pecahan linier secara aljabar menjadi bentuk program linier ?
2. Bagaimana transformasi program pecahan linier menjadi bentuk program linier ?
3. Bagaimana langkah pembuatan aplikasi yang akan digunakan untuk membantu perhitungan dengan cepat?

1.3 Pembatasan Masalah

Pembatasan masalah dalam penulisan ini yaitu

1. Penyelesaian program linier menggunakan metode simpleks direvisi yang memiliki kelebihan dalam mengidentifikasi kasus khusus.
2. Kasus khusus program linier seperti tidak ada daerah layak (*infeasible*) dan tidak terbatas (*unbounded*) dianggap tidak memberikan hasil pada program pecahan linier.
3. Hanya menyelesaikan program pecahan linier yang memenuhi asumsi
   - konstanta penyebut yaitu $\beta > 0$
   - kriteria optimasi fungsi tujuan pada penyebut dalam bentuk vektor kolom (berordo $n \times 1$) adalah $\mathbf{d} = [d_{j1}],\ d_{j1} \geq 0$
   - seluruh fungsi kendala dalam bentuk matriks (berordo $m \times n$) adalah $\mathbf{A} = [a_{ij}],\ a_{ij} \geq 0$

untuk $i = 1, 2, 3, \cdots, m$ dan $j = 1, 2, 3, \cdots, n$.

1.4 Tujuan Penulisan

Tujuan yang diharapkan dari penulisan skripsi ini adalah:

1. Mengetahui transformasi program pecahan linier menjadi bentuk program linier.
2. Mengetahui transformasi hasil optimal dari program linier kembali menjadi bentuk program pecahan linier.
3. Membuat aplikasi menggunakan MATLAB untuk membantu perhitungan program pecahan linier dengan cepat.

1.5 Manfaat Penulisan

Dalam penulisan skripsi ini diharapkan dapat meningkatkan pemahaman tentang penyelesaian pada permasalahan program pecahan linier. Selain itu, hasil penulisan ini diharapkan dapat digunakan sebagai salah satu referensi dalam studi mengenai bidang optimasi.

1.6 Metode Penelitian

Skripsi ini merupakan merupakan rekayasa produk dalam bidang matematika optimasi didasarkan pada buku-buku dan jurnal-jurnal yang terkait dengan optimasi menggunakan program pecahan linier dengan transformasi ke dalam bentuk program linier. Referensi utama yang digunakan yaitu Saha, dkk (2015) serta Pandian dan Jayalakshmi (2013).

BAB II

LANDASAN TEORI

Pada bab ini akan dibahas tentang program pecahan linier, program linier, dan penyelesaian program linier untuk setiap jenis kendala dengan metode simpleks direvisi. Sebagai awalan, akan dijelaskan mengenai matriks karena seluruh pembahasan program pecahan linier dan program linier akan dibentuk dalam matriks.

2.1 Matriks

Pendefinisian tentang matriks (Eiselt & Sandblom, 2007:1) yang akan digunakan adalah sebagai berikut.

Definisi 2.1.1. Matriks **A** adalah susunan dua dimensi dari elemen a_{ij} yang berukuran m baris dan n kolom (berordo $m \times n$) sehingga a_{ij} yang merupakan elemen dalam baris i dan kolom j. Jika $m = n$, dikatakan matriks persegi; jika $m = 1$ dikatakan vektor baris, jika $n = 1$ dikatakan vektor kolom, dan jika $m = n = 1$ dikatakan skalar.

Notasi yang biasa digunakan untuk menyatakan suatu matriks adalah tanda kurung "( )" atau kurung siku "[ ]". Ordo dapat dituliskan sebagai indeks pada matriks. Suatu matriks **A** yang memiliki elemen a_{ij} juga dapat dituliskan sebagai $\mathbf{A} = [a_{ij}]$.

Matriks umumnya dinamai dengan menggunakan huruf kapital tebal seperti $\mathbf{A}, \mathbf{B}, \mathbf{C}$, dan seterusnya. Vektor umumnya dinamai dengan menggunakan huruf kecil tebal seperti **a,b,c**, dan seterusnya.

Definisi 2.1.2. Suatu matriks $A = [a_{ij}]$ berordo $n \times n$, dapat dikatakan dengan matriks identitas apabila setiap elemen dengan elemen pada diagonal utamanya bernilai $a_{ij} = 1$ jika $i = j$ dan $a_{ij} = 0$ jika $i \neq j$. Matriks identitas dinotasikan dengan **I**

Definisi 2.1.3. Hasil operasi penjumlahan dari dua matriks berordo $m \times n$ dari matriks **A** dan **B** adalah matriks berordo $m \times n$ yaitu matriks **C** sedemikian sehingga

$$c_{ij} = a_{ij} + b_{ij} \quad \forall\ i = 1, 2, \cdots, m;\ j = 1, 2, \cdots, n$$

Hasil operasi pengurangan didefinisikan sama

Definisi 2.1.4. Hasil operasi perkalian matriks $\mathbf{A} = [a_{ij}]$ berordo $m \times n$ dengan matriks $\mathbf{B} = [b_{jk}]$ berordo $n \times p$ adalah matriks $\mathbf{C} = [c_{ik}]$ berordo $m \times p$ sedemikian sehingga

$$c_{ik} = \sum_{j=1}^{n} a_{ij} b_{jk} \quad \forall\ i = 1, 2, \cdots, m;\ k = 1, 2, \cdots, p$$

Definisi 2.1.5. *Transpose* suatu matriks $\mathbf{A} = [a_{ij}]$ berordo $m \times n$ adalah matriks $\mathbf{A}^T = [a_{ij}^T]$ berordo $n \times m$ sedemikian sehingga $a_{ij} = a_{ji}^T$. Jika $\mathbf{A} = \mathbf{A}^T$ maka **A** dinamakan matriks simetris

Definisi 2.1.6. Jika **A** dan **B** matriks berordo $n \times n$ sedemikian sehingga **AB=BA=I**, dimana **I** matriks identitas maka matriks **B** disebut invers dari matriks **A** dan matriks **A** invers dari matriks **B**. Notasi invers matriks **A** adalah $\mathbf{A}^{-1}$.

Proposisi 2.1.1. Untuk matriks **A**, **B**, **C**, berlaku hasil sebagai berikut untuk suatu perkalian, suatu *trasnspose*, dan suatu invers:

- $\mathbf{AI} = \mathbf{IA} = \mathbf{A}$

- $\mathbf{A}(\mathbf{BC}) = (\mathbf{AB})\mathbf{C}$
- $(\mathbf{A}^T)^T = \mathbf{A}$
- $(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$
- $(\mathbf{A}^{-1})^{-1} = \mathbf{A}$
- $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$
- $(\mathbf{A}^T)^{-1} = (\mathbf{A}^{-1})^T$

2.2 Program Linier (PL)

Program linier yang ditemukan oleh L.W Kantorovich pada tahun 1939 dengan metode yang masih terbatas. Kata program pada program linier bukan berarti program komputer, melainkan sebuah perancangan atau perencanaan. Kata linier menunjukan bahwa semua fungsi matematis yang disajikan haruslah fungsi linier.

Pada permasalahan program linier dikenal dua macam fungsi, yaitu fungsi tujuan (*objective function*) dan fungsi batasan (*constraint function*). Fungsi tujuan yaitu fungsi yang dapat menggambarkan hasil yang dicapai dengan tujuan yang optimal. Fungsi batasan atau lebih dikenal dengan fungsi kendala adalah fungsi yang menjadi kondisi, syarat, atau batasan yang harus terpenuhi.

Prinsip dasar dalam menyelesaikan proram linier ialah mencari seluruh kemungkinan pemecahan yang layak (*feasible*) kemudian memilih salah satu pemecahan yang memberikan nilai tujuan optimal, yaitu paling besar untuk memaksimalkan atau paling kecil untuk meminimalkan.

Fungsi tujuan program linier dapat dirumuskan sebagai berikut

$$\text{Maksimum } Z = \mathbf{c}^T\mathbf{x} \tag{2.1}$$

dengan kendala

$$\mathbf{Ax}(\leq, =, \geq)\mathbf{b}, \quad \mathbf{x} \geq \mathbf{0} \tag{2.2}$$

dimana $\mathbf{A}$ adalah matriks berordo $m \times n$, $\mathbf{b}$ adalah vektor kolom berordo $m \times 1$, serta $\mathbf{x}$ dan $\mathbf{c}$ adalah vektor kolom berordo $n \times 1$

$$\mathbf{x} = \begin{bmatrix} x_{11} \\ x_{21} \\ \vdots \\ x_{n1} \end{bmatrix}, \qquad \mathbf{c} = \begin{bmatrix} c_{11} \\ c_{21} \\ \cdots \\ c_{n1} \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} b_{11} \\ b_{21} \\ \vdots \\ b_{m1} \end{bmatrix},$$

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

dengan $\mathbf{x,c} \in \mathbb{R}^n$; $\mathbf{b} \in \mathbb{R}^m$; $\mathbf{A} \in \mathbb{R}^{m \times n}$.

Keterangan:

Z = Optimasi nilai pengambilan keputusan dari sautu fungsi tujuan

x_{j1} = Variabel keputusan ke-j (untuk $j = 1, 2, \cdots, n$)

c_{j1} = Kriteria optimasi yang dikenakan pada variabel ke-j

b_{i1} = Batas sumber daya kendala ke-i (untuk $i = 1, 2, \cdots, m$)

a_{ij} = Sumber daya yang digunakan pada variabel ke-j untuk kendala ke-i

Program linier dapat diselesaikan dengan berbagai macam cara seperti menggunakan subtitusu dan eliminasi, metode grafik, metode garis selidik, dan lain sebagainya. Banyaknya metode yang dapat digunakan tidak menjamin hasil selalu bisa didapat, masih terdapat kendala-kendala yang harus dihadapi misalnya tidak ada daerah yang layak (*infeasible*), terdapat daerah layak namun tak terbatas (*unbounded*). Beberapa kasus bahkan menunjukan

mungkin saja terdapat lebih dari satu penyelesaian optimum di daerah layak.

Dalam penyelesaian permasalahan program linear, tanda sisi kanan kendala harus bernilai non negatif, jika tanda pada bagian sisi kanan kendala bernilai negatif maka persamaan tersebut harus dikalikan dengan bilangan -1 agar tanda pada bagian sisi kanan kendala bernilai positif.

2.2.1 Metode Simpleks Direvisi

Penggunaan metode simpleks ternyata bukan merupakan prosedur perhitungan yang paling efektif dalam komputer. Alasannya karena metode simpleks masih mengerjakan beberapa variabel yang tidak diperlukan. Hal tersebut yang mendasari perbaikan metode simpleks menjadi metode simpleks direvisi. Terdapat tiga tahapan dalam metode simpleks direvisi yaitu:

A. Perumusan fungsi

Perumusan fungsi menjadi bentuk yang dapat dikerjakan oleh metode simpleks direvisi yaitu dengan menambahkan variabel *slack* atau variabel buatan pada setiap fungsi untuk mendapatkan bentuk augmentasinya (*augmented form*). Adapun perubahan fungsi kendalanya sebagai berikut:

- Fungsi kendala " $\leq$ "

  Penyelesaian fungsi kendala dengan tanda " $\leq$ " harus diubah menjadi bentuk " = " dengan menambahkan variabel *slack*. Harga variabel *slack* pada fungsi tujuan adalah 0. Kegunaan variabel *slack* menyatakan sumber daya yang tidak terpakai dalam proses.

- Fungsi kendala " = "

  Penyelesaian fungsi kendala dengan tanda " = " harus ditambahkan variabel buatan. Variabel buatan secara fisik tidak mempunyai arti,

karena digunakan hanya untuk membantu perhitungan saja. Penambahan variabel buatan ini akan merusak sistem batasan, tetapi hal ini dapat diatasi dengan membuat suatu bilangan besar M.

Pada fungsi tujuan memaksimalkan harga dibuat menjadi $-M$, dan jika fungsi tujuan meminimalkan harga dibuat menjadi $+M$ sebagai harga dari variabel buatan. Metode ini dikenal dengan nama metode M besar (*Big M method*).

- Fungsi kendala " $\geq$ "

  Penyelesaian fungsi kendala dengan tanda " $\geq$ " harus diubah menjadi bentuk " $=$ " dengan menambahkan variabel *slack*, kemudian tambahkan variabel buatan serta gunakan metode M besar untuk menyelesaikan permalasahan.

Penyelesaian metode simpleks direvisi biasanya digunakan untuk fungsi yang bertujuan memaksimalkan, tetapi terdapat cara yang mudah untuk melakukan metode simpleks direvisi dengan tujuan meminimalkan yaitu dengan mengubah tujuan meminimalkan menjadi memaksimalkan dengan fungsi yang ekuivalen seperti berikut

$$\text{Min} \quad Z = \sum_{i=1}^{n} c_i x_j$$

ekuivalen dengan

$$\text{Max} \quad -Z = \sum_{i=1}^{n} (-c_i) x_j \tag{2.3}$$

Untuk mendapatkan bentuk augmentasi (*augmented form*) dari fungsi-fungsi tersebut maka dibutuhkan vektor kolom untuk variabel *slack* yaitu

$$\mathbf{x_S} = \begin{bmatrix} x_{(n+1)1} \\ \vdots \\ x_{(n+m)1} \end{bmatrix}, \quad \mathbf{x_S} \geq \mathbf{0}$$

setelah itu inisial tabel metode simpleks dapat dituliskan dengan notasi matriks sebagai berikut

$$\begin{bmatrix} 1 & -\mathbf{c}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{A} & \mathbf{I} \end{bmatrix} \begin{bmatrix} Z \\ \mathbf{x} \\ \mathbf{x_S} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix} \tag{2.4}$$

pada matriks

$$\begin{bmatrix} 1 & -\mathbf{c}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{A} & \mathbf{I} \end{bmatrix}$$

dapat dipecah menjadi vektor-vektor kolom yang bersesuaian dengan $Z,\ x_1,\ x_2, \cdots,\ x_{n+m}$ secara berurutan.

B. Inisialisasi Matriks

Pada tahap inisialisai yang terpilih menjadi variabel dasar pada vektor $\mathbf{x_B}$ merupakan variabel *slack* sedemikian sehingga

$$\mathbf{x_B} = \begin{bmatrix} Z \\ x_{(n+1)1} \\ \vdots \\ x_{(n+m)1} \end{bmatrix}, \quad \mathbf{B}^{-1} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} = \mathbf{I}$$

maka

$$\mathbf{x_B} = \mathbf{B}^{-1}\mathbf{b} = \mathbf{b}.$$

Apabila terdapat variabel buatan maka yang terpilih menjadi variabel dasar adalah variabel buatan, dan koefisien baris pertama pada matriks $\mathbf{B}^{-1}$ yang bersesuaian dengan variabel buatan akan diganti dengan $-M$.

C. Perhitungan metode simpleks direvisi

Setelah semua fungsi sudah dalam bentuk augmentasi serta inisialisai variabel sudah didapatkan, proses selanjutnya ialah melakukan perhitungan untuk mendapatkan hasil. Tardapat 5 langkah untuk melakukan perhitungan metode simpleks direvisi dalam **satu kali iterasi** yaitu:

Langkah 1

Menentukan variabel dasar yang masuk, dengan cara mengalikan baris pertama $\mathbf{B}^{-1}$ dengan vektor yang bersesuaian dengan variabel x_i yang bukan bagian dari variabel dasar dinamakan δ_i. Pilihlah δ_i yang paling minimum. Apabila δ_i terpilih bernilai positif maka proses dianggap sudah optimal.

Langkah 2

Menentukan vektor kolom $\mathbf{v} = [v_i]$ yaitu matriks $\mathbf{B}^{-1}$ dikalikan dengan vektor yang bersesuaian dengan x_i yang terpilih dari δ_i.

Langkah 3

Menentukan variabel dasar yang keluar dilihat dari nilai bagi positif terkecil dari elemen $\mathbf{x_B}$ dengan elemen pada vektor $\mathbf{v}$ dengan syarat elemen pada vektor $\mathbf{v}$ harus positif.

Langkah 4

Setelah mendapatkan $\mathbf{x_B}$ baru akan dibentuk matriks $\mathbf{B}^{-1}_{baru}$ dengan rumus

$$\mathbf{B}^{-1}_{baru} = \mathbf{E}\mathbf{B}^{-1}_{lama}$$

dengan matriks $\mathbf{E} = [e_{ij}]$ dibangun dari matriks identitas kecuali pada kolom ke-k dimana k juga merupakan posisi pada variabel dasar yang keluar, diganti dengan

$$e_{ik} = \begin{cases} \dfrac{1}{v_k} & i = k \\ -\dfrac{v_i}{v_k} & i \neq k \end{cases}$$

Langkah 5

Menentukan nilai dari $\mathbf{x_B}$ baru yaitu

$$\mathbf{x_B} = \mathbf{B}_{baru}^{-1}\mathbf{b}$$

setelah itu lanjut ke iterasi berikutnya dengan mengulang kembali proses dari **Langkah 1** sampai nilai dianggap optimal. Selanjutnya akan dibeberikan beberapa contoh permasalahan program linier yang mempresentasikan kasus memaksimalkan, kasus meminimalkan, kendala yang seragam dan kendala yang tidak seragam.

Contoh 2.2.1. Diberikan permasalahan program linier yang mempresentasikan kasus memaksimalkan dan mempresentasikan kendala yang seragam yaitu

$$\text{Maksimum } Z = 2x_1 + x_2 \tag{2.5}$$

dengan kendala

$$\begin{aligned} 3x_1 + 4x_2 &\leq 6 \\ 6x_1 + 1x_2 &\leq 3 \\ x_1 \geq 0, \quad & x_2 \geq 0 \end{aligned} \tag{2.6}$$

akan diselesaikan dengan menggunakan metode simpleks direvisi yaitu sebagai berikut.

A. Perumusan fungsi

Penambahan variabel *slack* pada Persamaan (2.5) dan (2.6) menjadi

$$Z = 2x_1 + x_2 + 0x_3 + 0x_4 \tag{2.7}$$

$$\begin{aligned} 3x_1 + 4x_2 + x_3 &= 6 \\ 6x_1 + 1x_2 + x_4 &= 3 \end{aligned} \tag{2.8}$$

dimana variabel x_3 dan x_4 adalah variabel *slack*. Menggunakan notasi matriks, Persamaan (2.7) dan (2.8) dapat di tuliskan ke dalam bentuk berikut

$$\begin{bmatrix} 1 & -2 & -1 & 0 & 0 \\ 0 & 3 & 4 & 1 & 0 \\ 0 & 6 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} Z \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 6 \\ 3 \end{bmatrix}$$

pada matriks

$$\begin{bmatrix} 1 & -2 & -1 & 0 & 0 \\ 0 & 3 & 4 & 1 & 0 \\ 0 & 6 & 1 & 0 & 1 \end{bmatrix}$$

dapat dibagi menjadi vektor-vektor kolom yang bersesuaian dengan Z , x_1, x_2, x_3, dan x_4 secara berurutan.

B. Inisialisasi Matriks

Pada tahap inisialisai yang terpilih menjadi variabel dasar merupakan variabel *slack* sedemikian sehingga

$$\mathbf{x_B} = \begin{bmatrix} Z \\ x_3 \\ x_4 \end{bmatrix}, \quad \mathbf{B}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \mathbf{I}$$

maka

$$\mathbf{x_B} = \mathbf{B}^{-1}\mathbf{b} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 6 \\ 3 \end{bmatrix} = \begin{bmatrix} 0 \\ 6 \\ 3 \end{bmatrix}.$$

C. Perhitungan metode simpleks direvisi

Setelah semua fungsi sudah dalam bentuk augmentasi serta inisialisai variabel sudah didapatkan, proses selanjutnya ialah melakukan perhitungan untuk mendapatkan hasil.

ITERASI 1

Langkah 1

Menentukan variabel dasar yang masuk dengan cara mengalikan baris pertama $\mathbf{B}^{-1}$ dengan vektor yang bersesuaian. Perkalian tersebut menggunakan variabel-variabel yang bukan bagian dari variabel dasar.

$$\delta_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} -2 \\ 3 \\ 6 \end{bmatrix} = -2\ ,$$

$$\delta_2 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 4 \\ 1 \end{bmatrix} = -1$$

karena δ_1 yang paling negatif maka x_1 akan masuk menjadi variabel dasar.

Langkah 2

menentukan $\mathbf{v}$ yaitu $\mathbf{B}^{-1}$ dikalikan dengan vektor terpilih

$$\mathbf{v} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -2 \\ 3 \\ 6 \end{bmatrix} = \begin{bmatrix} -2 \\ 3 \\ 6 \end{bmatrix}.$$

Langkah 3

Menentukan variabel dasar yang keluar dilihat dari nilai bagi positif terkecil dari elemen $\mathbf{x_B}$ dengan elemen pada vektor $\mathbf{v}$ dengan syarat elemen pada vektor $\mathbf{v}$ harus positif.

Variabel	$\mathbf{B^{-1}}$			$\mathbf{x_B}$	$\mathbf{v}$	$\mathbf{x_B/v}$
Z	1	0	0	0	-2	-
x_3	0	1	0	6	3	2
x_4	0	0	1	3	6	1/2

maka variabel dasar yang baru adalah

$$\mathbf{x_B} = \begin{bmatrix} Z \\ x_3 \\ x_1 \end{bmatrix}.$$

Langkah 4

Membuat matriks $\mathbf{B}^{-1}_{baru}$ dengan rumus

$$\mathbf{B}^{-1}_{baru} = \mathbf{E}\mathbf{B}^{-1}_{lama}$$

dengan matriks $\mathbf{E} = [e_{ij}]$ dibangun dari matriks identitas kecuali pada kolom ke-3 dimana kolom ke-3 juga merupakan posisi pada variabel dasar yang keluar,

diganti dengan

$$e_{ik} = \begin{cases} \dfrac{1}{v_k} & i = k \\ -\dfrac{v_i}{v_k} & i \neq k \end{cases}$$

Sehingga didapat matriks $\mathbf{E}$ sebagai berikut

$$\mathbf{E} = \begin{bmatrix} 1 & 0 & 1/3 \\ 0 & 1 & -1/2 \\ 0 & 0 & 1/6 \end{bmatrix}$$

maka

$$\mathbf{B}_{baru}^{-1} = \mathbf{E}\mathbf{B}_{lama}^{-1} = \begin{bmatrix} 1 & 0 & 1/3 \\ 0 & 1 & -1/2 \\ 0 & 0 & 1/6 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1/3 \\ 0 & 1 & -1/2 \\ 0 & 0 & 1/6 \end{bmatrix}.$$

Langkah 5

Menentukan nilai dari $\mathbf{x_B}$

$$\mathbf{x_B} = \mathbf{B}^{-1}\mathbf{b} = \begin{bmatrix} 1 & 0 & 1/3 \\ 0 & 1 & -1/2 \\ 0 & 0 & 1/6 \end{bmatrix} \begin{bmatrix} 0 \\ 6 \\ 3 \end{bmatrix} = \begin{bmatrix} 1 \\ 9/2 \\ 1/2 \end{bmatrix}.$$

Setelah itu masuk pada iterasi ke-2 dan ulangi langkah dari awal.

ITERASI 2

Langkah 1

Menentukan variabel dasar yang masuk dengan cara mengalikan baris pertama $\mathbf{B}^{-1}$ dengan vektor yang bersesuaian. Perkalian tersebut menggunakan

variable-variabel yang bukan bagian dari variabel dasar.

$$\delta_2 = \begin{bmatrix} 1 & 0 & 1/3 \end{bmatrix} \begin{bmatrix} -1 \\ 4 \\ 1 \end{bmatrix} = -2/3 \,,$$

$$\delta_4 = \begin{bmatrix} 1 & 0 & 1/3 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = 1/3$$

karena δ_2 yang paling negatif maka x_2 akan masuk menjadi variabel dasar.

Langkah 2

menentukan $\mathbf{v}$ yaitu $\mathbf{B}^{-1}$ dikalikan dengan vektor terpilih

$$\mathbf{v} = \begin{bmatrix} 1 & 0 & 1/3 \\ 0 & 1 & -1/2 \\ 0 & 0 & 1/6 \end{bmatrix} \begin{bmatrix} -1 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} -2/3 \\ 7/2 \\ 1/6 \end{bmatrix}.$$

Langkah 3

Menentukan variabel dasar yang keluar dilihat dari nilai bagi positif terkecil dari elemen $\mathbf{x_B}$ dengan elemen pada vektor $\mathbf{v}$ dengan syarat elemen pada vektor $\mathbf{v}$ harus positif.

Variabel	$\mathbf{B}^{-1}$			$\mathbf{x_B}$	$\mathbf{v}$	$\mathbf{x_B}/\mathbf{v}$
Z	1	0	1/3	1	-2/3	-
x_3	0	1	-1/2	9/2	7/2	9/7
x_1	0	0	1/6	1/2	1/6	3

maka variabel dasar yang baru adalah

$$\mathbf{x_B} = \begin{bmatrix} Z \\ x_2 \\ x_1 \end{bmatrix}.$$

Langkah 4

Membuat matriks $\mathbf{B}^{-1}_{baru}$ dengan rumus

$$\mathbf{B}^{-1}_{baru} = \mathbf{E}\mathbf{B}^{-1}_{lama}$$

dengan matriks $\mathbf{E} = [e_{ij}]$ dibangun dari matriks identitas kecuali pada kolom ke-2 dimana kolom ke-2 juga merupakan posisi pada variabel dasar yang keluar, diganti dengan

$$e_{ik} = \begin{cases} \dfrac{1}{v_k} & i = k \\ -\dfrac{v_i}{v_k} & i \neq k \end{cases}$$

Sehingga didapat matriks $\mathbf{E}$ sebagai berikut

$$\mathbf{E} = \begin{bmatrix} 1 & 4/21 & 0 \\ 0 & 2/7 & 0 \\ 0 & -1/21 & 1 \end{bmatrix}$$

maka

$$\mathbf{B}^{-1}_{baru} = \mathbf{E}\mathbf{B}^{-1}_{lama} = \begin{bmatrix} 1 & 4/21 & 0 \\ 0 & 2/7 & 0 \\ 0 & -1/21 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1/3 \\ 0 & 1 & -1/2 \\ 0 & 0 & 1/6 \end{bmatrix} = \begin{bmatrix} 1 & 4/21 & 5/21 \\ 0 & 2/7 & -1/7 \\ 0 & -1/21 & 4/21 \end{bmatrix}.$$

Langkah 5

Menentukan nilai dari $\mathbf{x_B}$

$$\mathbf{x_B} = \mathbf{B}^{-1}\mathbf{b} = \begin{bmatrix} 1 & 4/21 & 5/21 \\ 0 & 2/7 & -1/7 \\ 0 & -1/21 & 4/21 \end{bmatrix} \begin{bmatrix} 0 \\ 6 \\ 3 \end{bmatrix} = \begin{bmatrix} 13/7 \\ 9/7 \\ 2/7 \end{bmatrix}.$$

Setelah itu masuk pada iterasi ke-3 dan ulangi langkah dari awal.

ITERASI 3

Langkah 1

Menentukan variabel dasar yang masuk dengan cara mengalikan baris pertama $\mathbf{B}^{-1}$ dengan vektor yang bersesuaian. Perkalian tersebut menggunakan variable-variabel yang bukan bagian dari variabel dasar.

$$\delta_3 = \begin{bmatrix} 1 & 4/21 & 5/21 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = 4/21\ ,$$

$$\delta_4 = \begin{bmatrix} 1 & 4/21 & 5/21 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = 5/21$$

karena semua δ_i bernilai positif maka proses berhenti. Solusi didapat dengan nilai $Z = 13/7,\ x_1 = 2/7,\ x_2 = 9/7$.

Contoh 2.2.2. Diberikan permasalahan program linier yang mempresentasikan kasus meminimalkan dan mempresentasikan kendala yang tidak seragam yaitu

$$\text{Minimum } Z = 12x_1 + 10x_2 \tag{2.9}$$

dengan kendala

$$\begin{aligned} 4x_1 + 3x_2 &\leq 66 \\ 6x_1 + 10x_2 &\geq 140 \\ 5x_1 + 4x_2 &= 82 \\ x_1 \geq 0, \quad x_2 &\geq 0 \end{aligned} \tag{2.10}$$

akan diselesaikan dengan menggunakan metode simpleks direvisi yaitu sebagai berikut.

A. Perumusan fungsi

Penambahan variabel *slack* dan variabel buatan pada Persamaan (2.5) dan (2.6) menjadi

$$Z = 12x_1 + 10x_2 + 0x_3 + 0x_4 + Mx_5 + Mx_6$$

ekuivalen dengan memaksimalkan

$$-Z = -12x_1 - 10x_2 - 0x_3 - 0x_4 - Mx_5 - Mx_6 \tag{2.11}$$

dan kendala pada Persamaan (2.10) berubah menjadi

$$\begin{aligned} 4x_1 + 3x_2 + x_3 &= 66 \\ 6x_1 + 10x_2 - x_4 + x_5 &= 140 \\ 5x_1 + 4x_2 + x_6 &= 82 \\ x_1 \geq 0, \quad x_2 &\geq 0 \end{aligned} \tag{2.12}$$

dengan x_3 dan x_4 adalah variabel *slack* serta x_5 dan x_6 adalah variabel buatan. Persamaan (2.11) dan (2.12) dengan menggunakan notasi matriks dapat di tuliskan ke dalam bentuk berikut

$$\begin{bmatrix} 1 & 12 & 10 & 0 & 0 & M & M \\ 0 & 4 & 3 & 1 & 0 & 0 & 0 \\ 0 & 6 & 10 & 0 & -1 & 1 & 0 \\ 0 & 5 & 4 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -Z \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 40 \\ 60 \\ 84 \end{bmatrix}$$

pada matriks

$$\begin{bmatrix} 1 & 12 & 10 & 0 & 0 & M & M \\ 0 & 4 & 3 & 1 & 0 & 0 & 0 \\ 0 & 6 & 10 & 0 & -1 & 1 & 0 \\ 0 & 5 & 4 & 0 & 0 & 0 & 1 \end{bmatrix}$$

dapat dibagi menjadi vektor-vektor kolom yang bersesuaian dengan Z, x_1, x_2, x_3, x_4, x_5 dan x_6 secara berurutan.

B. Inisialisasi Matriks

Pada tahap inisialisasi yang terpilih menjadi variabel dasar merupakan variabel *slack* atau variabel buatan sedemikian sehingga

$$\mathbf{x_B} = \begin{bmatrix} -Z \\ x_3 \\ x_5 \\ x_6 \end{bmatrix}, \quad \mathbf{B}^{-1} = \begin{bmatrix} 1 & 0 & -M & -M \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

maka

$$\mathbf{x_B} = \mathbf{B}^{-1}\mathbf{b} = \begin{bmatrix} 1 & 0 & -M & -M \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 66 \\ 140 \\ 82 \end{bmatrix} = \begin{bmatrix} -222M \\ 66 \\ 140 \\ 82 \end{bmatrix}.$$

C. Perhitungan metode simpleks direvisi

Dengan cara yang sama seperti sebelumnya, jika dijabarkan maka proses terhenti pada iterasi ke-3 dengan nilai $-Z = -200$, $x_1 = 10,\ x_2 = 8$.

2.2.2 Kasus Khusus pada Program Linier

Ada beberapa kasus khusus pada program linier yang antara lain:

1. Optimasi alternatif

Optimasi Alternatif terjadi jika sedikitnya ada satu δ_i pada iterasi terakhir yang memiliki koefisien berinilai 0. Akibatnya Solusi-solusi optimum yang lain dapat diidentifikasi.

2. Degenerasi

Degenerasi terjadi apabila satu atau lebih variabel dasar berharga 0 pada suatu iterasi sehingga hasil dari iterasi berikutnya sama dengan hasil iterasi sebelumnya.

3. Solusi takterbatas

Solusi takterbatas (*unbounded*) dapat langsung terdeteksi jika semua koefisien pada vektor $\mathbf{v}$ tidak ada satupun yang bernilai positif.

4. Tidak ada daerah layak

Tidak ada daerah layak (*infeasible*) terjadi jika semua batasan tidak dapat dipenuhi secara simultan, dapat terdeteksi jika pada iterasi akhir terdapat minimal satu variabel buatan yang masih terpilih sebagai variabel dasar.

2.3 Program Pecahan Linier (PPL)

Program Pecahan Linier (PPL) secara luas dikembangkan oleh seorang matematikawan Hungaria B.Martos dan asosiasinya sekitar tahun 1960. Kelebihan metode ini ada pada fungsi tujuan yang merupakan sebuah pecahan (rasio). Aplikasi program pecahan linier dapat ditemukan pada beberapa bidang, salah satunya bidang ekonomi. Dalam ilmu ekonomi, sering di jumpai permasalahan efisiensi yang dibahas lebih lanjut dengan menggunakan sebuah rasio, berikut beberapa aplikasi yang sering dijumpai seperti:

- Memaksimalkan laba terhadap investasi.

  Masalah alokasi sumber daya yang membahas bagaimana perbandingan antara laba yang didapat dibandingkan dengan modal yang di keluarkan.

- Memaksimalkan laba terhadap resiko.

  Membandingkan laba yang diterima dengan resiko yang harus ditanggung merukan hal yang biasa terjadi dalam ekonomi, keberhasilan penghitungan dapat dicari dengan mencari laba sebesar-besarnya dengan menanggung resiko sekecil mungkin.

dan masih banyak lagi. Fungsi tujuan program pecahan linier dapat dirumuskan sebagai berikut

$$\text{Maksimum } Z = \frac{\mathbf{c}^T\mathbf{x} + \alpha}{\mathbf{d}^T\mathbf{x} + \beta} \tag{2.13}$$

dengan kendala

$$\mathbf{A}\mathbf{x}(\leq, =, \geq)\mathbf{b}, \quad \mathbf{x} \geq \mathbf{0} \tag{2.14}$$

dimana

$$\mathbf{x} = \begin{bmatrix} x_{11} \\ x_{21} \\ \vdots \\ x_{n1} \end{bmatrix}, \qquad \mathbf{c} = \begin{bmatrix} c_{11} \\ c_{21} \\ \vdots \\ c_{n1} \end{bmatrix}, \qquad \mathbf{d} = \begin{bmatrix} d_{11} \\ d_{21} \\ \vdots \\ d_{n1} \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} b_{11} \\ b_{21} \\ \vdots \\ b_{m1} \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

dengan $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x}$ adalah variabel keputusan, $\mathbf{c}, \mathbf{d} \in \mathbb{R}^n$ dan $\mathbf{b} \in \mathbb{R}^m$ adalah koefisien-koefisien yang diketahui, $\mathbf{A} \in \mathbb{R}^{m \times n}$ adalah matriks yang diketahui dan $\alpha, \beta \in \mathbb{R}$ adalah konstanta. Kendala permasalahan dibatasi wilayah layak $\{\mathbf{x} | \mathbf{d}^T\mathbf{x} + \beta \neq 0\}$. (Charnes & Cooper, 1962)

BAB III

REKAYASA MODEL

3.1 Transformasi Program Pecahan Linier

Terdapat beberapa metode dalam memecahkan permasalahan program pecahan linier, salah satunya dengan transformasi. Kegunaan transformasi adalah untuk menyelesaikan permasalahan menjadi bentuk yang lebih mudah untuk dikenali. Program pecahan linier lebih sukar untuk diselesaikan secara langsung, oleh karena itu program pecahan linier akan di transformasoi menjadi bentuk program linier agar proses pencarian optimasi dapat diselesaikan dengan lebih mudah.

Seperti pada batasan masalah, asumsikan selalu terdapat daerah layak bukan himpunan kosong $S = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{d}^T\mathbf{x} + \beta \neq 0\}$ dan terbatas, untuk $\mathbf{c}, \mathbf{d} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{A} \in \mathbb{R}^{m \times n}$ dan $\alpha, \beta \in \mathbb{R}$ yang memenuhi syarat konstanta penyebut yaitu $\beta > 0$, kriteria optimasi penyebut dalam bentuk vektor kolom (berordo $n \times 1$) adalah $\mathbf{d} = [d_{j1}],\ d_{j1} \geq 0$ dan fungsi kendala dalam bentuk matriks (berordo $m \times n$) adalah $\mathbf{A} = [a_{ij}],\ a_{ij} \geq 0$ untuk $i = 1, 2, 3, \cdots, m$ dan $j = 1, 2, 3, \cdots, n$.

$$\text{maksimum } Z = \frac{\mathbf{c}^T\mathbf{x} + \alpha}{\mathbf{d}^T\mathbf{x} + \beta}$$

dengan kendala

$$\mathbf{Ax}(\leq, =, \geq)\mathbf{b}, \quad \mathbf{x} \geq \mathbf{0}$$

akan ditransformasikan menjadi bentuk program linier dengan aljabar, menggunakan konstanta program pecahan linier yaitu α pada pembilang dan β pada penyebut dibentuk sebagai berikut

$$-\frac{\alpha}{\beta}+\frac{\alpha}{\beta}$$

menjadi

$$\begin{aligned}
Z &= \frac{\mathbf{c}^T\mathbf{x}+\alpha}{\mathbf{d}^T\mathbf{x}+\beta} \\
&= \frac{\mathbf{c}^T\mathbf{x}+\alpha}{\mathbf{d}^T\mathbf{x}+\beta}-\frac{\alpha}{\beta}+\frac{\alpha}{\beta} \\
&= \frac{\left(\mathbf{c}^T\mathbf{x}+\alpha\right)\beta}{\left(\mathbf{d}^T\mathbf{x}+\beta\right)\beta}-\frac{\left(\mathbf{d}^T\mathbf{x}+\beta\right)\alpha}{\left(\mathbf{d}^T\mathbf{x}+\beta\right)\beta}+\frac{\alpha}{\beta} \\
&= \left(\frac{\mathbf{c}^T\mathbf{x}\beta+\alpha\beta}{\left(\mathbf{d}^T\mathbf{x}+\beta\right)\beta}-\frac{\mathbf{d}^T\mathbf{x}\alpha+\alpha\beta}{\left(\mathbf{d}^T\mathbf{x}+\beta\right)\beta}\right)+\frac{\alpha}{\beta} \\
&= \frac{(\mathbf{c}^T\beta-\mathbf{d}^T\alpha)\mathbf{x}}{(\mathbf{d}^T\mathbf{x}+\beta)\beta}+\frac{\alpha}{\beta} \\
&= \frac{(\mathbf{c}^T\beta-\mathbf{d}^T\alpha)}{\beta}\frac{\mathbf{x}}{(\mathbf{d}^T\mathbf{x}+\beta)}+\frac{\alpha}{\beta} \qquad (3.1)
\end{aligned}$$

didefinisikan

$$\mathbf{g}^T=\frac{\mathbf{c}^T\beta-\mathbf{d}^T\alpha}{\beta} \qquad (3.2)$$

$$\mathbf{y}=\frac{\mathbf{x}}{\mathbf{d}^T\mathbf{x}+\beta} \qquad (3.3)$$

$$h=\frac{\alpha}{\beta} \qquad (3.4)$$

invers fungsi pada Persamaan (3.3) didapatkan sebagai berikut

$$\mathbf{x} = \frac{\mathbf{y}\beta}{1 - \mathbf{d}^T\mathbf{y}} \tag{3.5}$$

maka kendala misalkan dengan tanda "$\leq$" dapat ditransformasikan menjadi

$$\begin{aligned} \mathbf{A}\mathbf{x} &\leq \mathbf{b} \\ \mathbf{A}\left(\frac{\mathbf{y}\beta}{1 - \mathbf{d}^T\mathbf{y}}\right) &\leq \mathbf{b} \\ \mathbf{A}\mathbf{y}\beta &\leq \mathbf{b}(1 - \mathbf{d}^T\mathbf{y}) \\ \mathbf{A}\mathbf{y}\beta + \mathbf{b}\mathbf{d}^T\mathbf{y} &\leq \mathbf{b} \\ (\mathbf{A}\beta + \mathbf{b}\mathbf{d}^T)\mathbf{y} &\leq \mathbf{b} \end{aligned} \tag{3.6}$$

didefinisikan

$$\mathbf{K} = \mathbf{A}\beta + \mathbf{b}\mathbf{d}^T \tag{3.7}$$

dari Persamaan (3.2), (3.3), dan (3.4) maka program pecahan linier pada Persamaan (3.1) dapat ditransformasikan menjadi bentuk program linier yaitu

$$Z = \mathbf{g}^T\mathbf{y} + h \tag{3.8}$$

Catatan 3.1.1. Konstanta pada program linier akan dimasukkan setelah didapat hasil yang optimal, maka yang akan diproses pada metode simpleks direvisi adalah

$$Z = \mathbf{g}^T\mathbf{y}$$

serta dari Persamaan (3.6) dan (3.7) kendala dibentuk menjadi

$$\mathbf{K}\mathbf{y} \leq \mathbf{b} \tag{3.9}$$

dimana

$$\mathbf{y} = \begin{bmatrix} y_{11} \\ y_{21} \\ \vdots \\ y_{n1} \end{bmatrix}, \qquad \mathbf{K} = \begin{bmatrix} k_{11} & k_{12} & \dots & k_{1n} \\ k_{21} & k_{22} & \dots & k_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ k_{m1} & k_{m2} & \dots & k_{mn} \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} b_{11} \\ b_{21} \\ \vdots \\ b_{m1} \end{bmatrix},$$

$$\mathbf{g} = \begin{bmatrix} g_{11} \\ g_{21} \\ \vdots \\ g_{n1} \end{bmatrix}$$

dengan $\mathbf{y}, \mathbf{g} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{K} \in \mathbb{R}^{m \times n}$ dan $h \in \mathbb{R}$ adalah konstanta.

Keterangan:

Z = Optimasi nilai pengambilan keputusan dari sautu fungsi tujuan

y_{j1} = Variabel keputusan ke-j (untuk $j = 1, 2, \cdots, n$)

g_{j1} = Kriteria optimasi yang dikenakan pada variabel ke-j

b_{i1} = Batas sumber daya kendala ke-i (untuk $i = 1, 2, \cdots, m$)

k_{ij} = Sumber daya yang digunakan pada variabel ke-j untuk kendala ke-i

Kelebihan metode yang digunakan pada penulisan ini adalah tidak ada tambahan fungsi pada kendala seperti pada transformasi Charnes dan Cooper. Kelebihan lainnya dapat menyelesaikan permasalahan untuk $\mathbf{c}^T\mathbf{x} + \alpha < 0$, dimana menurut Saha, dkk (2015) metode seperti seperti *updated objective function method* akan gagal jika $\mathbf{d}^T\mathbf{x} + \beta > 0$ dan $\mathbf{c}^T\mathbf{x} + \alpha < 0, \ \forall \mathbf{x} \in S$.

Kelemahan metode yang digunakan pada penulisan ini adalah masih terdapat permasalahan PPL yang sebenernya mempunyai solusi tetapi tidak dapat diselesaikan dalam bentuk PL karena teridentifikasi sebagai kasus khusus yaitu *unbounded* dan *infeasible*. Serta untuk beberapa permasalahan diluar

asumsi, dapat diselesaikan bila memenuhi syarat-syarat tertentu atau diselesaikan dengan menggunakan metode lainnya, tetapi tidak dibahas pada penulisan ini.

3.2 Langkah-Langkah Penyelesaian Manual

Untuk menyelesaikan optimasi program pecahan linier dengan transformasi menggunakan aljabar konstanta menjadi bentuk program linier, terdapat tiga tahapan yang harus dikerjakan yaitu:

A. Transformasi program pecahan linier menjadi program linier.

B. Penyelesaian program linier dengan metode simpleks direvisi.

C. Transformasi kembali variabel keputusan. $\mathbf{y}$ menjadi $\mathbf{x}$

Diberikan contoh permasalahan program pecahan linier yang diselesaikan secara manual untuk menunjukan bagaimana cara prosedur dari tahapan diatas, sebagai berikut:

Contoh 3.2.1.

$$\text{Maksimum } Z = \frac{2x_1 + 5x_2 + 1}{0.25x_1 + 0.5x_2 + 0.5} \tag{3.10}$$

dengan kendala

$$\begin{aligned} 3x_1 + 5x_2 &\leq 45 \\ 1x_1 + 1x_2 &= 13 \\ x_1 \geq 0, \quad & x_2 \geq 0 \end{aligned} \tag{3.11}$$

permasalahan tersebut harus dibuat dengan bentuk matriks terlebih dahulu menjadi

$$\mathbf{A} = \begin{bmatrix} 3 & 5 \\ 1 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 45 \\ 13 \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} 2 \\ 5 \end{bmatrix}, \quad \mathbf{d} = \begin{bmatrix} 0.25 \\ 0.5 \end{bmatrix}$$

dan $\alpha = 1$, $\beta = 0.5$.

A. Transformasi program pecahan linier menjadi program linier

dimana

$$\mathbf{g}^T = \frac{0.5 \begin{bmatrix} 2 & 5 \end{bmatrix} - 1 \begin{bmatrix} 0.25 & 0.5 \end{bmatrix}}{0.5} = \begin{bmatrix} 1.5 & 4 \end{bmatrix}$$

dan

$$h = \frac{1}{0.5} = 2$$

serta

$$\mathbf{K} = \begin{bmatrix} 3 & 5 \\ 1 & 1 \end{bmatrix} 0.5 + \begin{bmatrix} 45 \\ 13 \end{bmatrix} \begin{bmatrix} 0.25 & 0.5 \end{bmatrix} = \begin{bmatrix} 12.75 & 25 \\ 3.75 & 7 \end{bmatrix}$$

Maka fungsi tujuan pada Persamaan (3.10) setelah transformasi adalah

$$Z = 1.5y_1 + 4y_2 + 2 \tag{3.12}$$

seperti Catatan (3.1.1) konstanta pada program linier akan dimasukkan setelah didapat hasil yang optimal, maka yang akan diproses pada metode simpleks direvisi hanya

$$Z = 1.5y_1 + 4y_2$$

dengan kendala pada Persamaan (3.11) setelah transformasi adalah

$$\begin{aligned} 12.75y_1 + 25y_2 &\leq 45 \\ 3.75y_1 + 7y_2 &= 13 \\ y_1 \geq 0, \quad & y_2 \geq 0 \end{aligned} \tag{3.13}$$

B. Penyelesaian program linier dengan metode simpleks direvisi

Permasalahan program pecahan linier yang sudah di transfomasikan diselesaikan dengan menggunakan metode simpleks direvisi sebagai berikut.

1. Perumusan fungsi

Penambahan variabel *slack* pada Persamaan (3.12) dan (3.13) menjadi

$$Z = 1.5y_1 + 4y_2 + 0y_3 - My_4 \tag{3.14}$$

$$\begin{aligned} 12.75y_1 + 25y_2 + y_3 &= 45 \\ 3.75y_1 + 7y_2 + y_4 &= 13 \end{aligned} \tag{3.15}$$

dengan y_3 adalah variabel *slack* dan y_4 adalah variabel buatan. Persamaan (3.14) dan (3.15) dengan menggunakan notasi matriks dapat dituliskan ke dalam bentuk berikut

$$\begin{bmatrix} 1 & -1.5 & -4 & 0 & M \\ 0 & 12.75 & 25 & 1 & 0 \\ 0 & 3.75 & 7 & 0 & 1 \end{bmatrix} \begin{bmatrix} Z \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 45 \\ 13 \end{bmatrix}$$

pada matriks

$$\begin{bmatrix} 1 & -1.5 & -4 & 0 & M \\ 0 & 12.75 & 25 & 1 & 0 \\ 0 & 3.75 & 7 & 0 & 1 \end{bmatrix}$$

dapat dibagi menjadi vektor-vektor kolom yang bersesuaian dengan Z , y_1, y_2, y_3, dan y_4 secara berurutan.

2. Inisialisasi Matriks

Pada tahap inisialisai yang terpilih menjadi variabel dasar merupakan variabel *slack* atau variabel buatan sedemikian hingga

$$\mathbf{y_B} = \begin{bmatrix} Z \\ y_3 \\ y_4 \end{bmatrix}, \quad \mathbf{B^{-1}} = \begin{bmatrix} 1 & 0 & -M \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

maka

$$\mathbf{y_B} = \mathbf{B^{-1}b} = \begin{bmatrix} 1 & 0 & -M \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 45 \\ 13 \end{bmatrix} = \begin{bmatrix} -13M \\ 45 \\ 13 \end{bmatrix}$$

3. Perhitungan metode simpleks direvisi

Setelah semua fungsi sudah dalam bentuk augmentasi serta inisialisai variabel sudah didapatkan, proses selanjutnya ialah melakukan perhitungan untuk mendapatkan hasil.

ITERASI 1

Langkah 1

Menentukan variabel dasar yang masuk dengan cara mengalikan baris pertama $\mathbf{B^{-1}}$ dengan vektor yang bersesuaian. Perkalian dilakukan dengan menggunakan variabel-variabel yang bukan merupakan bagian dari variabel dasar.

$$\delta_1 = \begin{bmatrix} 1 & 0 & -M \end{bmatrix} \begin{bmatrix} -1.5 \\ 12.75 \\ 3.75 \end{bmatrix} = -3.75M - 1.5$$

$$\delta_2 = \begin{bmatrix} 1 & 0 & -M \end{bmatrix} \begin{bmatrix} -4 \\ 25 \\ 7 \end{bmatrix} = -7M - 4$$

karena δ_2 yang paling negatif maka y_2 akan masuk menjadi variabel dasar.

Langkah 2

menentukan $\mathbf{v}$ yaitu $\mathbf{B}^{-1}$ dikalikan dengan vektor terpilih

$$\mathbf{v} = \begin{bmatrix} 1 & 0 & -M \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -4 \\ 25 \\ 7 \end{bmatrix} = \begin{bmatrix} -7M - 4 \\ 25 \\ 7 \end{bmatrix}.$$

Langkah 3

Menentukan variabel dasar yang keluar dilihat dari nilai bagi positif terkecil dari elemen $\mathbf{y_B}$ dengan elemen pada vektor $\mathbf{v}$ dengan syarat elemen pada vektor $\mathbf{v}$ harus positif.

Variabel	$\mathbf{B}^{-1}$			$\mathbf{y_B}$	$\mathbf{v}$	$\mathbf{y_B}/\mathbf{v}$
Z	1	0	$-M$	$-13M$	$-7M-4$	-
y_3	0	1	0	45	25	1.8
y_4	0	0	1	13	7	1.86

maka variabel dasar yang baru adalah

$$\mathbf{x_B} = \begin{bmatrix} Z \\ y_2 \\ y_4 \end{bmatrix}.$$

Langkah 4

Membuat matriks B_{baru}^{-1} dengan rumus

$$\mathbf{B}_{baru}^{-1} = \mathbf{E}\mathbf{B}_{lama}^{-1}$$

dengan matriks $\mathbf{E} = [e_{ij}]$ dibangun dari matriks identitas kecuali pada kolom ke-2 dimana kolom ke-2 juga merupakan posisi pada variabel dasar yang keluar, diganti dengan

$$e_{ik} = \begin{cases} \dfrac{1}{v_k} & i = k \\ -\dfrac{v_i}{v_k} & i \neq k \end{cases}$$

Sehingga didapat matriks $\mathbf{E}$ yaitu

$$\mathbf{E} = \begin{bmatrix} 1 & 0.28M + 0.16 & 0 \\ 0 & 0.04 & 0 \\ 0 & -0.28 & 1 \end{bmatrix}$$

maka

$$\mathbf{B}_{baru}^{-1} = \begin{bmatrix} 1 & 0.28M + 0.16 & 0 \\ 0 & 0.04 & 0 \\ 0 & -0.28 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -M \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0.28M + 0.16 & -M \\ 0 & 0.04 & 0 \\ 0 & -0.28 & 1 \end{bmatrix}.$$

Langkah 5

Menentukan nilai dari $\mathbf{y_B}$

$$\mathbf{y_B} = \mathbf{B}^{-1}\mathbf{b} = \begin{bmatrix} 1 & 0.28M + 0.16 & -M \\ 0 & 0.04 & 0 \\ 0 & -0.28 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 45 \\ 13 \end{bmatrix} = \begin{bmatrix} -0.4M + 7.2 \\ 1.8 \\ 0.04 \end{bmatrix}.$$

Setelah itu masuk pada iterasi ke-2 dan ulangi langkah dari awal.

ITERASI 2

Langkah 1

Menentukan variabel dasar yang masuk dengan cara mengalikan baris pertama $\mathbf{B}^{-1}$ dengan vektor yang bersesuaian. Perkalian dilakukan dengan menggunakan variabel-variabel yang bukan merupakan bagian dari variabel dasar.

$$\delta_1 = \begin{bmatrix} 1 & 0.28M + 0.16 & -M \end{bmatrix} \begin{bmatrix} -1.5 \\ 12.75 \\ 3.75 \end{bmatrix} = -0.18M - 0.54 \ ,$$

$$\delta_3 = \begin{bmatrix} 1 & 0.28M + 0.16 & -M \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = 0.28M + 0.16$$

karena δ_1 yang paling negatif maka y_1 akan masuk menjadi variabel dasar.

Langkah 2

menentukan $\mathbf{v}$ yaitu $\mathbf{B}^{-1}$ dikalikan dengan vektor terpilih

$$\mathbf{v} = \begin{bmatrix} 1 & 0.28M + 0.16 & -M \\ 0 & 0.04 & 0 \\ 0 & -0.28 & 1 \end{bmatrix} \begin{bmatrix} -1.5 \\ 12.75 \\ 3.75 \end{bmatrix} = \begin{bmatrix} -0.18M + 0.54 \\ 0.51 \\ 0.18 \end{bmatrix}.$$

Langkah 3

Menentukan variabel dasar yang keluar dilihat dari nilai bagi positif terkecil dari elemen $\mathbf{y_B}$ dengan elemen pada vektor $\mathbf{v}$ dengan syarat elemen pada vektor $\mathbf{v}$ harus positif.

Variabel	$\mathbf{B}^{-1}$			$\mathbf{y_B}$	$\mathbf{v}$	$\mathbf{y_B}/\mathbf{v}$
Z	1	$0.28M + 0.16$	$-M$	$-0.4M + 7.2$	$-0.18M + 0.54$	-
y_2	0	0.04	0	1.8	0.51	3.5
y_4	0	-0.28	1	0.04	0.18	0.2

maka variabel dasar yang baru adalah

$$\mathbf{y_B} = \begin{bmatrix} Z \\ y_2 \\ y_1 \end{bmatrix}.$$

Langkah 4

Membuat matriks B^{-1}_{baru} dengan rumus

$$\mathbf{B}^{-1}_{baru} = \mathbf{E}\mathbf{B}^{-1}_{lama}$$

dengan matriks $\mathbf{E} = [e_{ij}]$ dibangun dari matriks identitas kecuali pada kolom ke-3 dimana kolom ke-3 juga merupakan posisi pada variabel dasar yang keluar, diganti dengan

$$e_{ik} = \begin{cases} \dfrac{1}{v_k} & i = k \\ -\dfrac{v_i}{v_k} & i \neq k \end{cases}$$

Sehingga didapat matriks $\mathbf{E}$ yaitu

$$\mathbf{E} = \begin{bmatrix} 1 & 0 & M-3 \\ 0 & 1 & -2.833 \\ 0 & 0 & 5.556 \end{bmatrix}$$

maka

$$\mathbf{B}_{baru}^{-1} = \begin{bmatrix} 1 & 0 & M-3 \\ 0 & 1 & -2.833 \\ 0 & 0 & 5.556 \end{bmatrix} \begin{bmatrix} 1 & 0.28M+0.16 & -M \\ 0 & 0.04 & 0 \\ 0 & -0.28 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & -3 \\ 0 & 0.833 & -2.833 \\ 0 & -1.555 & 5.556 \end{bmatrix}.$$

Langkah 5

Menentukan nilai dari $\mathbf{y_B}$

$$\mathbf{y_B} = \mathbf{B}^{-1}\mathbf{b} = \begin{bmatrix} 1 & 1 & -3 \\ 0 & 0.83 & -2.833 \\ 0 & -1.556 & 5.556 \end{bmatrix} \begin{bmatrix} 0 \\ 45 \\ 13 \end{bmatrix} = \begin{bmatrix} 6 \\ 0.667 \\ 2.222 \end{bmatrix}.$$

Setelah itu masuk pada iterasi ke-3 dan ulangi langkah dari awal.

ITERASI 3

Langkah 1

Menentukan variabel dasar yang masuk dengan cara mengalikan baris pertama $\mathbf{B}^{-1}$ dengan vektor yang bersesuaian. Perkalian dilakukan dengan mengguna-

kan variabel-variabel yang bukan merupakan bagian dari variabel dasar.

$$\delta_3 = \begin{bmatrix} 1 & 1 & -3 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = 1 \ ,$$

$$\delta_4 = \begin{bmatrix} 1 & 1 & -3 \end{bmatrix} \begin{bmatrix} M \\ 0 \\ 1 \end{bmatrix} = M - 3$$

karena semua δ_i bernilai positif maka proses berhenti. Sosuli didapat dengan nilai sebelum ditambahkan konstanta adalah $Z = 6$ dengan $y_1 = 2.222$ dan $y_2 = 0.667$.

C. Trasnformasi kembali variabel keputusan y menjadi x

Setelah mendapatkan hasil dari program pecahan linier, maka solusi optimum yaitu **y** harus dikembalikan menjadi bentuk **x** dengan

$$\mathbf{x} = \frac{\mathbf{y}\beta}{1 - \mathbf{d}^T\mathbf{y}}$$

dengan pembulatan menjadi

$$\mathbf{x} = \frac{\begin{bmatrix} 2.222 \\ 0.667 \end{bmatrix} 0.5}{1 - \begin{bmatrix} 0.25 & 0.5 \end{bmatrix} \begin{bmatrix} 2.222 \\ 0.667 \end{bmatrix}} = \begin{bmatrix} 10 \\ 3 \end{bmatrix}$$

maka didapatkan $x_1 = 10$ dan $x_2 = 3$, masukan angka tersebut pada Persamaan (3.10) didapatkan hasil optimal yaitu

$$Z = \frac{2(10) + 5(3) + 1}{0.25(10) + 0.5(3) + 0.5} = \frac{36}{4.5} = 8$$

Hasil tersebut sama dengan hasil yang diperoleh pada program linier dengan ditambahan konstanta yaitu $Z = 6+2 = 8$. Setelah mengerti proses penyelesaian secara manual, selanjutnya akan dibuat sebuah aplikasi untuk membantu melakukan perhitungan.

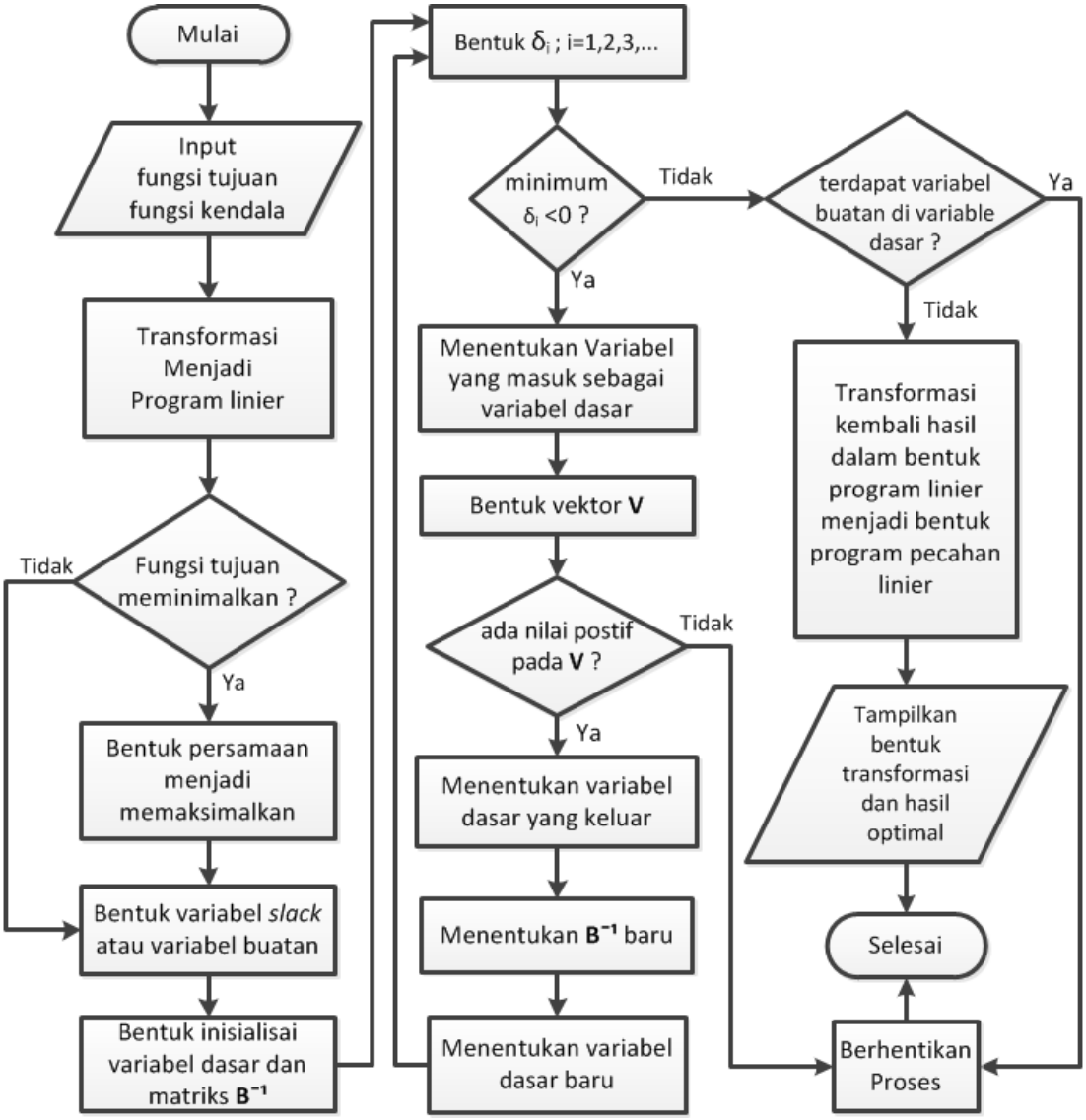
3.3 Proses Pembuatan Aplikasi Penyelesaian Program Pecahan Linier

Salah satu hal penting sebelum merancang sebuah aplikasi adalah menentukan perangkat keras (*hardware*) yang akan digunakan. Spesikasi *hardware* yang digunakan dalam pembuatan aplikasi adalah sebagai berikut:

Processor	:	Intel Core i3
Memory	:	3072
Monitor	:	14 Inch
Mouse dan Keyboard		

dengan menggunakan *operating system* Windows 7 Ultimate 32-bit dan menggunakan MATLAB R2012B sebagai perangkat lunaknya. Hal penting lainnya sebelum merancang sebuah aplikasi adalah menentukan algoritma dari aplikasi yang akan dibuat.

Algoritma atau langkah-langkah penyelesaian program pecahan linier pada aplikasi sama seperti penyelesaian secara manual pada subbab 3.2 dengan tambahan pendefinisian kasus-kasus khususnya seperti pada subsubbab 2.2.2. Algoritma penyelesaian program pecahan linier pada aplikasi dapat dijelaskan menggunakan diagram alir sebagai berikut:



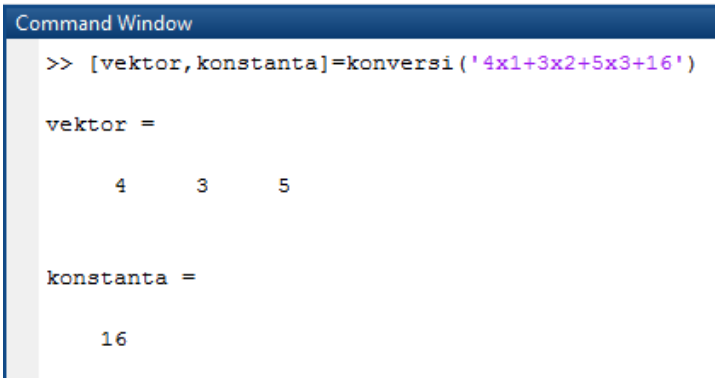
Gambar 3.1: Diagram Alir Penyelesaian Program Pecahan Linier pada Aplikasi.

Fungsi dalam pemrograman adalah pembentukan satu blok kode yang melakukan instruksi atau tugas tertentu yang bekerja ketika dipanggil dalam suatu program. Terdapat 4 fungsi pemrograman yang dibuat untuk menjalankan aplikasi yaitu:

1. Fungsi "konversi"

   Input dari fungsi pemrograman ini hanya 1 parameter yaitu sebuah string dimana *output*nya adalah vektor dan konstanta. Tujuan fungsi

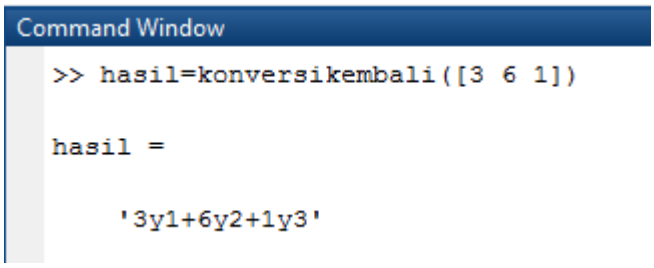
pemrograman ini adalah mengkonversi seluruh *input* dalam bentuk string pada aplikasi menjadi sebuah vektor dan konstanta agar dapat diolah menggunakan operasi matematika.



Gambar 3.2: Contoh Kerja Fungsi "konversi"

2. Fungsi "konversikembali"

   Input dari fungsi ini hanya 1 parameter yaitu sebuah vektor dimana *output*nya adalah sebuah string. Tujuan fungsi pemrograman ini adalah mengkonversi kembali hasil yang telah didapat dalam vektor menjadi sebuah string dangan menggunakan variabel y untuk ditampilkan pada aplikasi.

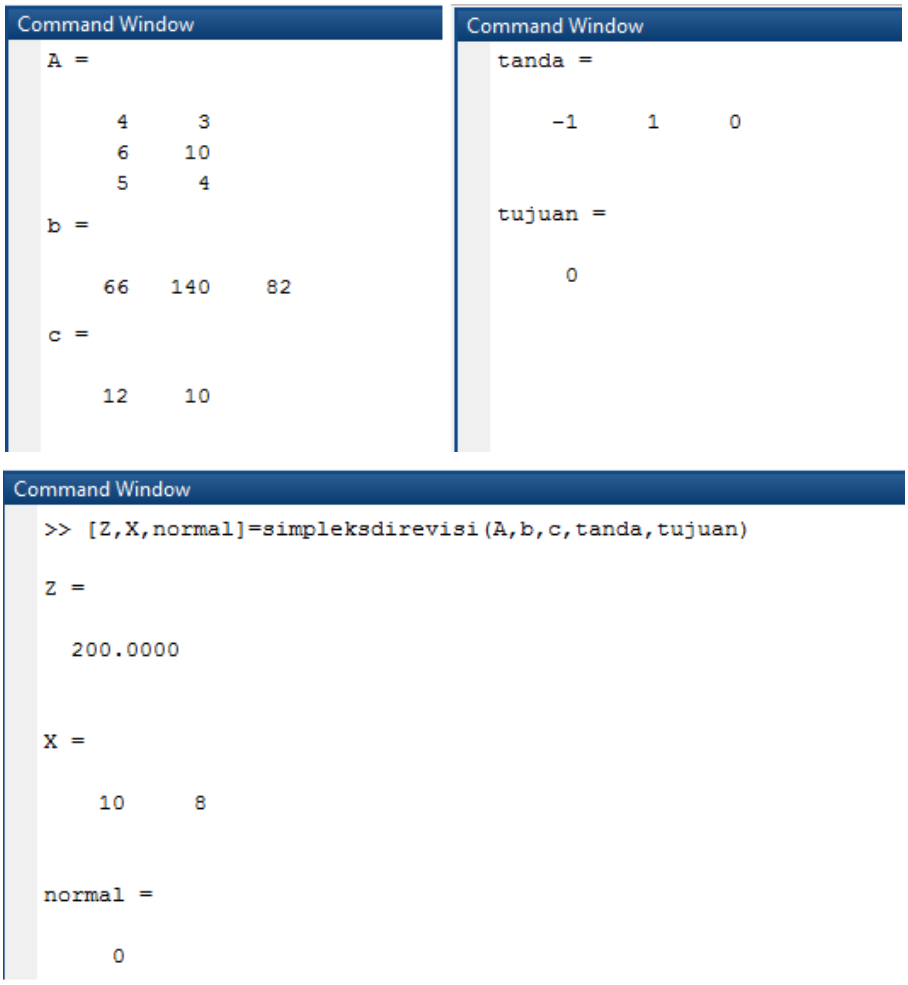


Gambar 3.3: Contoh Kerja Fungsi "konversikembali"

3. Fungsi "simpleksdirevisi"

   Input dari fungsi ini adalah 5 paramater dari permasalahan program linier yaitu fungsi kendala dalam bentuk matriks, nilai-nilai batas dalam bentuk vektor, fungsi tujuan dalam bentuk vektor, inisialisasi

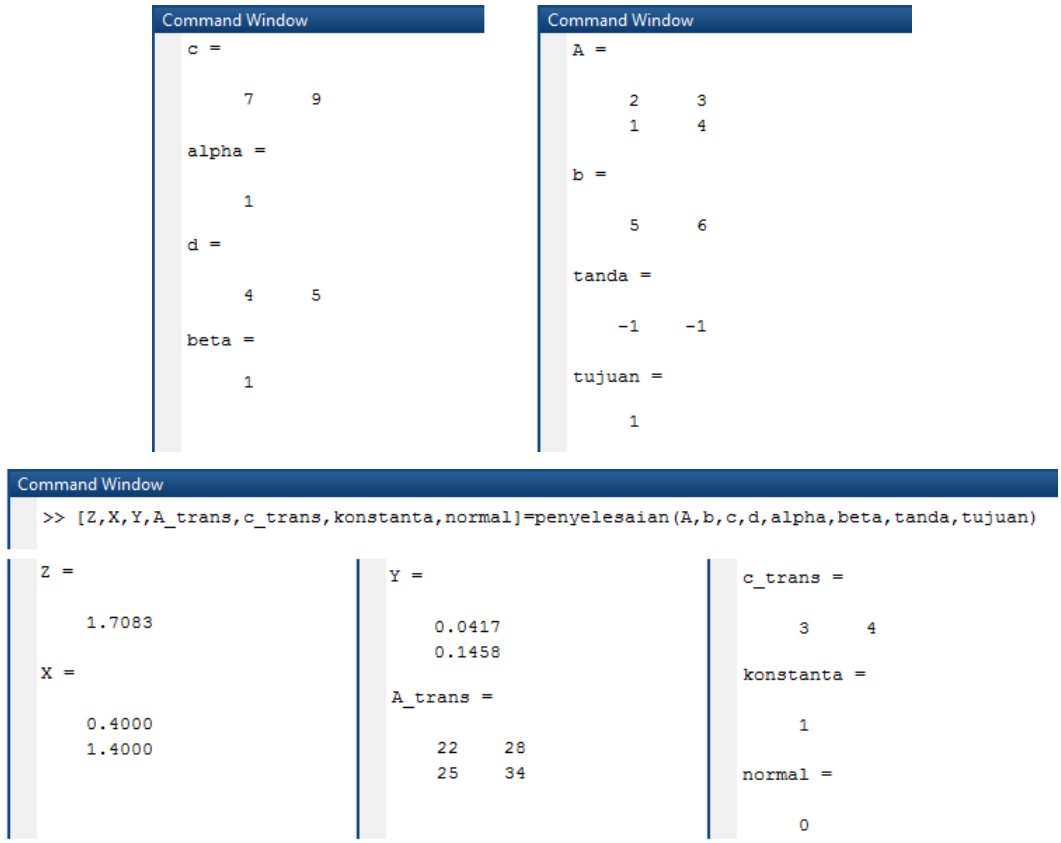
kendala dalam bentuk vektor (-1 untuk "≤", 0 untuk "=", dan 1 untuk "≥"), dan tujunnya memaksimalkan (didefinisikan dengan angka 1) atau meminimalkan (didefinisikan dengan angka 0). *Output*nya adalah hasil dari penyelesaian permasalahan program linier yaitu variabel keputusan dalam bentuk vektor, hasil optimal dalam bentuk konstanta serta apakah proses berjalan normal (0 untuk normal, 1 untuk optimasi alternatif, 2 untuk Degenari, 3 untuk *unbounded*, 4 untuk *infeasible*). Tujuan fungsi pemrograman ini adalah menyelesaikan permasalahn program linier menggunakan metode simpleks direvisi.



Gambar 3.4: Contoh Kerja Fungsi "simpleksdirevisi"

4. Fungsi ”penyelesaian”

Fungsi ”penyelesaian” merupakan inti dari aplikasi yang dibuat. *Input* fungsinya adalah 8 paramater dari permasalahan program pecahan linier yaitu fungsi kendala dalam bentuk matriks, nilai batas dalam bentuk vektor, pembilang fungsi tujuan dalam bentuk vektor, penyebut fungsi tujuan dalam bentuk vektor, konstanta pada pembilang fungsi tujuan, konstanta pada penyebut fungsi tujuan, inisialisasi kendala dalam bentuk vektor, dan tujunnya memaksimalkan atau meminimalkan. *Output*nya adalah hasil penyelesain permasalahan program pecahan linier yaitu variabel kebutusan dalam bentuk vektor hasil optimal dalam bentuk konstanta serta hasil dari bentuk transformasi program pecahan linier menjadi bentuk program linier serta kenormalan.



Gambar 3.5: Contoh Kerja Fungsi ”penyelesaian”

BAB IV

APLIKASI HASIL MODEL

4.1 Contoh Kasus Program Pecahan Linier

Seseorang ingin menginvestasikan asetnya dalam bidang *furniture*. Bahan dasar yang sering digunakan yaitu kayu jati, kayu sungkai, kayu akasia, dan kayu mahoni. *Furniture* yang dibuat dari bahan tersebut adalah 1 set lengkap (meja lemari kursi pintu jendela dll) untuk rumah bertipe besar. Diperlukan biaya awal untuk membeli peralatan-peralatan dan mesin sekitar 700 juta. Gudang sudah tersedia dan dapat menampung bahan mentah untuk dijadikan sekitar 20 set lengkap. Untuk memaksimalkan gudang yang sudah tersedia maka gudang harus terisi minimal untuk 16 set lengkap. Dengan menggambil contoh data beberapa perusahaan di bidang *furniture*, didapatkan rata-rata biaya yang habiskan untuk membuat *furniture* 1 set lengkap dengan berbagai bahan kayu terdapat dalam table berikut (dalam Juta)

Tabel 4.1: Rata-rata Daftar Harga *Furniture*

Jenis Kayu	Bahan Mentah (Log Kayu)	Pegawai	Bahan Kimia	Peralatan Sekali Pakai	Biaya Operasional dan tahap finisihing	Rata-rata Harga Jual 1 Set Lengkap
Jati	148	30	8	11	6	347
Sungkai	126	27	7	10	5	246
Akasia	110	24	7.5	8.5	4	233
Mahoni	96	21	7	9	4	228

Maksimal investasi yang akan di berikan pada setiap bidang adalah sebagai berikut:

Investasi biaya maksimal untuk bahan mentah (log kayu)	:	2 Miliar
Investasi biaya maksimal untuk pegawai	:	420 Juta
Investasi biaya maksimal untuk bahan kimia	:	125 Juta
Investasi biaya maksimal untuk peralatan sekali pakai	:	160 Juta
Investasi biaya maksimal untuk operasional dan finisihing	:	100 Juta

Dengan kebutuhan dana sebesar itu terlalu beresiko jika hanya memikirkan keuntungan yang maksimal. Untuk itu fungsi tujuan utama adalah memaksimalkan rasio pengembalian dana investasi, atau dengan kata lain memaksimalkan seluruh hasil penjualan dengan meminimalkan dana investasi yang digunakan.

4.2 Hasil Penyelesaian Contoh Kasus

Pada contoh kasus yang sudah diberikan terdapat data yang dapat di bangun menjadi suatu fungsi. Untuk membangun fungsi di didefinisikan variabelnya yaitu

x_1 : satu set lengkap *furniture* dengan bahan dasar kayu jati.

x_2 : satu set lengkap *furniture* dengan bahan dasar kayu sungkai.

x_3 : satu set lengkap *furniture* dengan bahan dasar kayu akasia.

x_4 : satu set lengkap *furniture* dengan bahan dasar kayu mahoni.

Tahap selanjutnya adalah mengidentifikasikan fungsi tujuannya. Karena tujuan utamanya memaksimalkan seluruh hasil penjualan dengan meminimalkan dana investasi yang digunakan maka akan dibuat fungsi tujuan dengan membandingkan harga jual dengan modal yang dibutuhkan. Fungsi tujuan dapat dibentuk sebagai berikut

$$\text{Memaksimalkan } Z = \frac{347x_1 + 246x_2 + 233x_3 + 228x_4}{203x_1 + 175x_2 + 154x_3 + 137x_4 + 700}$$

Setalah menentukan fungsi tujuan, langkah selanjutnya adalah mengidentifikasi adalah fungsi kendalanya dari contoh data yang dimiliki sebagai berikut:

1. Karena keterbatasan investasi pada bahan mentah, maka didapatkan

$$148x_1 + 126x_2 + 110x_3 + 96x_4 \leq 2000$$

2. Karena keterbatasan investasi untuk pegawai, maka didapatkan

$$30x_1 + 27x_2 + 24x_3 + 21x_4 \leq 420$$

3. Karena keterbatasan investasi pada bahan kimia, maka didapatkan

$$8x_1 + 7x_2 + 7.5x_3 + 7x_4 \leq 125$$

4. Karena keterbatasan investasi pada peralatan yang sekali pakai, maka didapatkan

$$11x_1 + 10x_2 + 8.5x_3 + 9x_4 \leq 160$$

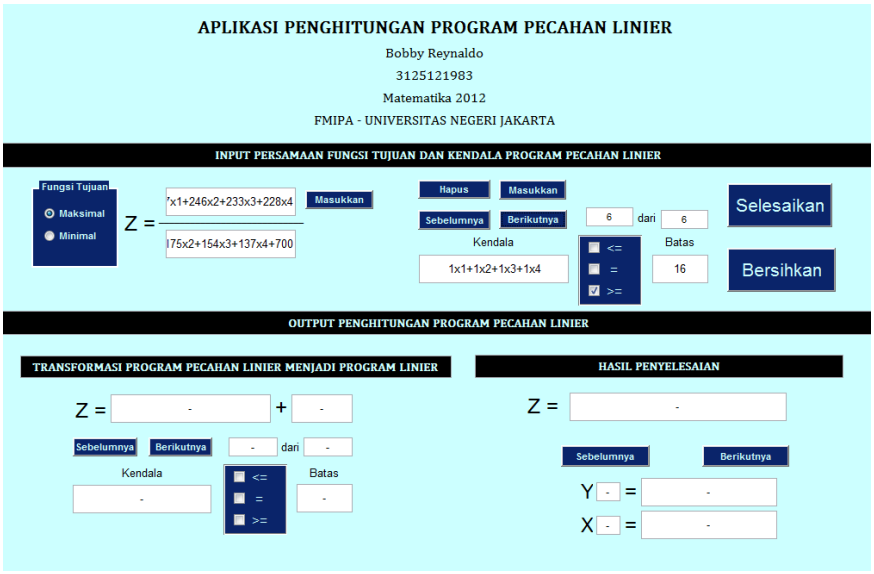
5. Karena keterbatasan investasi biaya operasional dan tahap finisihing, maka didapatkan

$$6x_1 + 5x_2 + 4x_3 + 4x_4 \leq 100$$

6. Untuk memaksimalkan gudang maka didapatkan

$$x_1 + x_2 + x_3 + x_4 \geq 16$$

Fungsi tujuan dan fungsi kendala kini telah lengkap. Selanjutnya penyelesaian contoh kasus di atas menggunakan aplikasi penghitungan program pecahan liner adalah sebagai berikut



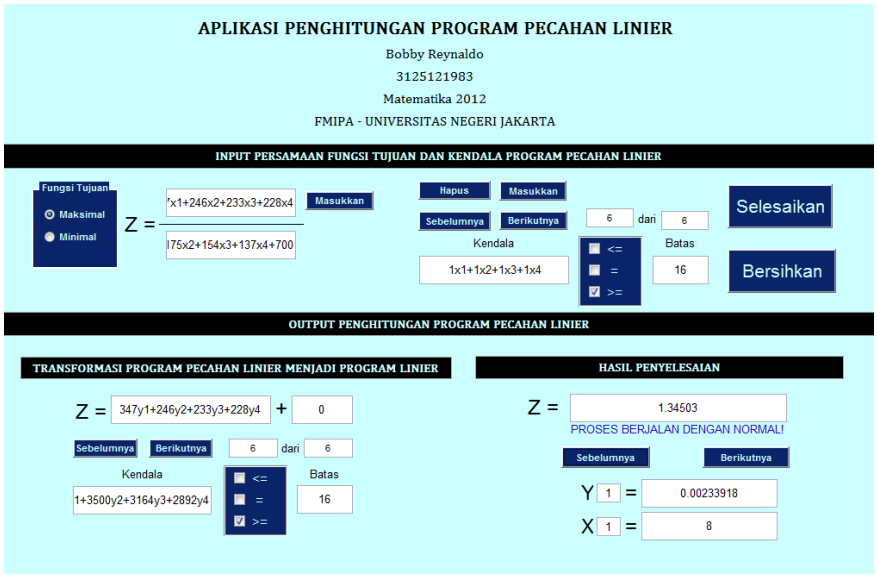
Gambar 4.1: *Input* Fungsi Tujuan dan Fungsi Kendala

Hasil transformasi fungsi tujuan program pecahan linier menjadi program linier sebagai berikut

$$Z = 347y_1 + 246y_2 + 233y_3 + 228y_4 + 0$$

Hasil transformasi fungsi kendala sebagai berikut

$$\begin{aligned}
509600y_1 &+ 438200y_2 + 385000y_3 + 341200y_4 \leq 2000 \\
106260y_1 &+ 92400y_2 + 81480y_3 + 72240y_4 \leq 420 \\
30975y_1 &+ 26775y_2 + 24500y_3 + 22025y_4 \leq 125 \\
40180y_1 &+ 35000y_2 + 30590y_3 + 28220y_4 \leq 160 \\
24500y_1 &+ 21000y_2 + 18200y_3 + 16500y_4 \leq 100 \\
3948y_1 &+ 3500y_2 + 3164y_3 + 2892y_4 \geq 16
\end{aligned}$$



Gambar 4.2: *Output* Transformasi dan Hasil Penyelesaian

Hasil dari variabel keputusan adalah

$$\begin{aligned} y_1 = 0.00233918 \quad &; \quad x_1 = 8 \\ y_2 = 0 \quad &; \quad x_2 = 0 \\ y_3 = 0 \quad &; \quad x_3 = 0 \\ y_4 = 0.00233918 \quad &; \quad x_4 = 8 \end{aligned}$$

Dengan hasil optimal yaitu

$$Z = 1.34503$$

artinya rasio terbesar dari hasil penjualan adalah sebesar 1.34503 kali dibandingkan dengan modal investasi yang digunakan.

4.3 Analisis Dan Kinerja Aplikasi

Aplikasi penghitungan program pecahan linier dapat berjalan baik bila mengikuti strandar dalam menginput data. Strandar dalam menginput data meliputi:

1. Variabel *input* didefinisikan hanya menggunakan variabel x.
2. Setiap angka pada variabel harus dituliskan walau bernilai 0 sekalipun untuk konsistensi dalam pembacaan data.
3. Dalam menginput fungsi, variabel harus dituliskan secara berurutan dari variabel ke-1 sampai ke-n secara konsisten.
4. Konstanta harus dituliskan terakhir.

Hal di atas perlu diperhatikan agar kesalahan dapat dihindari. Jika hal tersebut tidak terpenuhi maka aplikasi tidak mungkin memberikan hasil yang tepat atau bahkan tidak dapat dijalankan.

Tujuan utama pembuatan aplikasi adalah menyelesaikan permasalahan dengan cepat. Seberapa cepatnya waktu kinerja aplikasi selain dari perangkat keras juga dipengaruhi oleh tiga faktor lainnya yaitu jumlah variabel, jumlah fungsi, dan jumlah iterasi (pengulangan). Diberikan tabel untuk memperlihatkan seberapa cepat rata-rata waktu yang di butuhkan untuk menyelesaikan seluruh proses (dalam detik).

Tabel 4.2: Rata-rata Waktu Penyelesaian Aplikasi.

	3 Variabel		**4 Variabel**	
	3 Fungsi Kendala	**4 Fungsi Kendala**	**3 Fungsi Kendala**	**4 Fungsi Kendala**
3 Iterasi	0.006124	0.007062	0.007127	0.007508
4 Iterasi	0.006228	0.007215	0.007416	0.007812

dari tabel di atas, terlihat bahwa faktor variabel serta faktor fungsi kendala memberikan penambahan waktu yang cukup signifikan dibandingan dengan penambahan waktu pada faktor iterasi. Jika dianalogikan ke dalam bentuk

matriks maka penambahan variabel seperti penambahan pada kolom dan penambahan fungsi kendala seperti penambahan pada baris, artinya memperluas ruang matriks yang diproses. Walaupun demikian, aplikasi masih dapat dikatakan berjalan dengan cepat. Dari hasil pengujian untuk menyelesaikan permasalahan program pecahan linier dengan 10 variabel, 10 fungsi kendala, dan 10 kali iterasi, aplikasi mampu menyelesaikan seluruh proses hanya sekitar 0.026 detik.

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil dan pembahasan di atas, maka dapat diambil kesimpulan sebagai berikut:

1. Transformasi program pecahan linier menjadi bentuk program linier adalah sebagai berikut

$$Z = \mathbf{g}^T\mathbf{y} + h$$

dengan

$$\mathbf{g}^T = \frac{\mathbf{c}^T\beta - \mathbf{d}^T\alpha}{\beta} \; ; \quad \mathbf{y} = \frac{\mathbf{x}}{\mathbf{d}^T\mathbf{x} + \beta} \; ; \quad h = \frac{\alpha}{\beta}$$

dan transformasi kendalanya misalkan dengan tanda "$\leq$" menjadi

$$\mathbf{Ky} \leq \mathbf{b}$$

dengan

$$\mathbf{K} = \mathbf{A}\beta + \mathbf{bd}^T$$

2. Transformasi hasil optimal dari program linier dengan variabel keputusan yaitu $\mathbf{y}$ ditransformasi kembali menjadi bentuk program pecahan linier dengan variabel keputusan yaitu $\mathbf{x}$ adalah sebagai berikut

$$\mathbf{x} = \frac{\mathbf{y}\beta}{1 - \mathbf{d}^T\mathbf{y}}$$

3. Langkah dalam membuat aplikasi yaitu:

   - Menentukan perangkat yang akan digunakan.
   - Membangun Algoritmanya.
   - Membuat fungsi pemrogramannya.

   Sebuah Aplikasi telah dibuat menggunakan MATLAB R2012b untuk membantu penghitungan dengan cepat. Dari hasil pengujian untuk menyelesaikan permasalahan program pecahan linier dengan 10 variabel, 10 fungsi kendala, dan 10 kali iterasi, aplikasi mampu menyelesaikan seluruh proses hanya sekitar 0.026 detik.

5.2 Saran

Berikut beberapa saran untuk pengembangan selanjutnya adalah sebagai berikut:

- Aplikasi yang sudah dibuat masih memiliki banyak kelemahan, salah satunya tidak bisa membaca konstanta bila ditulis paling awal sebelum variabel, untuk itu aplikasi masih harus dilakukan pemuktahiran.
- Penyelesaian program pecahan linier masih menggunakan asumsi, untuk penelitian selanjutnya dapat mengembangkan penyelesaian program pecahan linier dengan lebih luas dan tanpa menambahkan asumsi.
- Penyelesaian program linier menggunakan metode simpleks direvisi dapat dikembangakan dengan metode lainnya seperti metode primal dual *path following* titik interior, *convex set* atau dengan metode lain tentu dengan menekankan kelebihan metode masing-masing.

DAFTAR PUSTAKA

Bitran, G.R. and A.G. Novaes, 1973. "Linear Programming with a Fractional Objective Function". *Operations Research*, 21(1), 22-29. [ONLINE] Tersedia: *http://pubsonline.informs.org/doi/abs/10.1287/opre.21.1.22* Diakses Sabtu, 12 Maret 2016 Pukul 16.07 WIB.

Chandra, S. Jayadeva. and Aparna M. 2009. *Numerical Optimization Applications*. Alpha Science International.

Charnes, A. and W.W. Cooper, 1962. "Programming with linear fractional". *Naval Research Logistics Quaterly*, Vol. 9: 181-186. [ONLINE] Tersedia: *http://onlinelibrary.wiley.com/doi/10.1002/nav.v9:3-4/issuetoc* Diakses Minggu, 6 Desember 2015 Pukul 18.31 WIB.

Dantzig, G.B, Thapa, M.N. 1997. *Linear Programming 1 : Introduction*. New York : Springer Berlin Heidelberg.

Dimyati, Tjutju T., Dimyati, Ahmad. 2003. *Operation Research, Model-model Pengambilan Keputusan*. Bandung: Sinar Baru Algesindo.

Eiselt, H.A., Sandblom, C.-L. 2007. *Linear Programming and its Application*. New York : Springer Berlin Heidelberg.

Hillier and Lieberman. Terjemahan : Dewa, P.M, The Jin Ai, Wigati, Selamat Setio, Hardjono, D. 2008. *Penelitian Operasional*. Yogyakarta : ANDI.

Pandian, P. and M. Jayalakshmi, 2013. "On Solving Linear Fractional Programming Problems", *Modern Applied Science*, Vol. 7, No. 6; 2013. [ONLINE] Tersedia: *http://dx.doi.org/10.5539/mas.v7n6p90* Diakses Minggu, 6 Desember 2015 Pukul 15.22 WIB.

Saha, S.K., dkk. 2015. "A New Approach of Solving Linear Fractional Programming Problem (LFP) by Using Computer Algorithm".

Open Journal of Optimization, 4, 74-86. [ONLINE] Tersedia: *http://dx.doi.org/10.4236/ojop.2015.43010* Diakses Senin, 7 Desember 2015 Pukul 20.42 WIB.

Zuhanda, M. Khahfi. 2013. "Optimasi Prgoram Linier Pecahan Dengan Fungsi Tujuan Berkoefisien Interval". *Skripsi*. FMIPA, Matematika, Universitas Sumatra Utara.

LAMPIRAN-LAMPIRAN

Script Aplikasi Penyelesaian Pogram Pecahan Linier

```
function varargout = solvepenyelesaian(varargin)
% SOLVEPPL MATLAB code for solvePPL.fig
%      SOLVEPPL, by itself, creates a new SOLVEPPL or raises the
%      existing
%      singleton*.
%
%      H = SOLVEPPL returns the handle to a new SOLVEPPL or the
%      handle to
%      the existing singleton*.
%
%      SOLVEpenyelesaian('CALLBACK',hObject,eventData,handles,...)
%      calls
%      the local
%      function named CALLBACK in SOLVEPPL.M with the given input
%      arguments.
%
%      SOLVEpenyelesaian('Property','Value',...) creates a new
%      SOLVEPPL or raises the
%      existing singleton*.  Starting from the left, property
%      value pairs are
%      applied to the GUI before solvePPL_OpeningFcn gets called.
%      An
%      unrecognized property name or invalid value makes property
%      application
%      stop.  All inputs are passed to solvePPL_OpeningFcn via
%      varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
%      only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help solvePPL
% Last Modified by GUIDE v2.5 20-Dec-2016 22:07:57

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @solvePPL_OpeningFcn, ...
                   'gui_OutputFcn',  @solvePPL_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before solvePPL is made visible.
function solvePPL_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to solvePPL (see VARARGIN)
handles.simpankendala={};
handles.A=[];
handles.B=[];
handles.C=[];
handles.D=[];
handles.beta=0;
handles.tanda=[];
set(handles.cb1,'Value',1);
% Choose default command line output for solvePPL
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes solvePPL wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = solvePPL_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
% Hints: get(hObject,'String') returns contents of penyebut as
text
%        str2double(get(hObject,'String')) returns contents of
penyebut as a double
[handles.D,handles.beta]=konversi(get(hObject,'String'));
guidata(hObject, handles);


% --- Executes during object creation, after setting all
properties.
function penyebut_CreateFcn(hObject, eventdata, handles)
% hObject    handle to penyebut (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in btnintujuan.
function btnintujuan_Callback(hObject, eventdata, handles)
% hObject    handle to btnintujuan (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
L=num2str(length(handles.simpankendala));
L1=length(handles.C);
L2=length(handles.D);
if L1==L2 && handles.beta>0 && ~any(handles.D<0)
    set(handles.indeks1,'String',L);
    set(handles.total1,'String',L);
    set(handles.ket1,'String','Berhasil!');
    pause(2);
    set(handles.ket1,'String',' ');
else
    set(handles.ket1,'String','Gagal!');
    pause(2);
    set(handles.ket1,'String',' ');
end
% Choose default command line output for solvePPL
handles.output = hObject;

% --- Executes on button press in btndelkendala.
```

```
l=length(handles.simpankendala);
indeks=str2num(get(handles.indeks1,'String'));
total=str2num(get(handles.total1,'String'));
if l>1 && indeks==0
    handles.simpankendala{l}=[];
    handles.simpankendala=potong(handles.simpankendala);
    handles.A(l,:)=[];
    handles.B(l)=[];
    handles.tanda(l)=[];
    set(handles.total1,'String',l-1);
    set(handles.indeks1,'String',0);
    set(handles.kendala,'String','');
    set(handles.batas,'String','');
    set(handles.cb1,'Value',1);
    set(handles.cb2,'Value',0);
    set(handles.cb3,'Value',0);
    set(handles.ket2,'String','Berhasil!');
    pause(2);
    set(handles.ket2,'String',' ');
elseif l>1 && indeks<=total
    handles.simpankendala{indeks}=[];
    handles.simpankendala=potong(handles.simpankendala);
    handles.A(indeks,:)=[];
    handles.B(indeks)=[];
    handles.tanda(indeks)=[];
    set(handles.total1,'String',l-1);
    set(handles.indeks1,'String',0);
    set(handles.kendala,'String','');
    set(handles.batas,'String','');
    set(handles.cb1,'Value',1);
    set(handles.cb2,'Value',0);
    set(handles.cb3,'Value',0);
    set(handles.ket2,'String','Berhasil!');
    pause(2);
    set(handles.ket2,'String',' ');
elseif l==1
    handles.simpankendala={};
    handles.A=[];
    handles.B=[];
    handles.tanda=[];
    set(handles.total1,'String',0);
    set(handles.indeks1,'String',0);
    set(handles.kendala,'String',' ');
    set(handles.cb1,'Value',1);
    set(handles.cb2,'Value',0);
    set(handles.cb3,'Value',0);
    set(handles.batas,'String',' ');
    set(handles.ket2,'String','Berhasil!');
    pause(2);
    set(handles.ket2,'String',' ');
end
guidata(hObject, handles);

% --- Executes on button press in btninkendala.
function btninkendala_Callback(hObject, eventdata, handles)
% hObject    handle to btninkendala (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
l=length(handles.simpankendala);
string=get(handles.kendala,'String');
kendala=konversi(string);
batas=str2num(get(handles.batas,'String'));
cek1=get(handles.cb1,'Value');
cek2=get(handles.cb2,'Value');
cek3=get(handles.cb3,'Value');
L1=length(handles.C);
L2=length(kendala);
if L1==L2 && cek1+cek2+cek3==1
    handles.simpankendala{l+1}=string;
    handles.A(l+1,:)=kendala;
    handles.B(l+1)=batas;
    if cek1==1
        handles.tanda(l+1)=-1;
        set(handles.cb1,'Value',1);
    elseif cek2==1
        handles.tanda(l+1)=0;
        set(handles.cb2,'Value',0);
        set(handles.cb1,'Value',1);
    elseif cek3==1
        handles.tanda(l+1)=1;
        set(handles.cb3,'Value',0);
        set(handles.cb1,'Value',1);
    end
    set(handles.total1,'String',l+1);
    set(handles.kendala,'String',' ');
    set(handles.batas,'String',' ');
    set(handles.ket2,'String','Berhasil!');
    pause(2);
    set(handles.ket2,'String',' ');

else
    set(handles.cb1,'Value',1);
    set(handles.cb2,'Value',0);
    set(handles.cb3,'Value',0);
    set(handles.ket2,'String','Gagal!');
    pause(2);
    set(handles.ket2,'String',' ');
end
guidata(hObject, handles);

function kendala_Callback(hObject, eventdata, handles)
% hObject    handle to kendala (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of kendala as text
%        str2double(get(hObject,'String')) returns contents of
kendala as a double
```

```
% --- Executes during object creation, after setting all
properties.
function kendala_CreateFcn(hObject, eventdata, handles)
% hObject    handle to kendala (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function batas_Callback(hObject, eventdata, handles)
% hObject    handle to batas (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of batas as text
%        str2double(get(hObject,'String')) returns contents of
batas as a double

% --- Executes during object creation, after setting all
properties.
function batas_CreateFcn(hObject, eventdata, handles)
% hObject    handle to batas (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function indeks1_Callback(hObject, eventdata, handles)
% hObject    handle to indeks1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of indeks1 as text
%        str2double(get(hObject,'String')) returns contents of
indeks1 as a double
```

```
% --- Executes during object creation, after setting all
properties.
function indeks1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to indeks1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in btnsebelum.
function btnsebelum_Callback(hObject, eventdata, handles)
% hObject    handle to btnsebelum (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
cek1=get(handles.cb1,'Value');
cek2=get(handles.cb2,'Value');
cek3=get(handles.cb3,'Value');
indeks=str2num(get(handles.indeks1,'String'));
total=str2num(get(handles.total1,'String'));
if indeks-1<1 || mod(indeks,1)~=0 || total<1
    set(handles.indeks1,'String',0);
    set(handles.kendala,'String',' ');
    set(handles.cb1,'Value',1);
    set(handles.cb2,'Value',0);
    set(handles.cb3,'Value',0);
    set(handles.batas,'String',' ');
else
    set(handles.indeks1,'String',indeks-1);
    set(handles.kendala,'String',handles.simpankendala{indeks-1});
    if handles.tanda(indeks-1)==-1
        set(handles.cb1,'Value',1);
        set(handles.cb2,'Value',0);
        set(handles.cb3,'Value',0);
    elseif handles.tanda(indeks-1)==0
        set(handles.cb1,'Value',0);
        set(handles.cb2,'Value',1);
        set(handles.cb3,'Value',0);
    elseif handles.tanda(indeks-1)==1
        set(handles.cb1,'Value',0);
        set(handles.cb2,'Value',0);
        set(handles.cb3,'Value',1);
    end
    set(handles.batas,'String',handles.B(indeks-1));
end
```

```
guidata(hObject, handles);

% --- Executes on button press in btnsesudah.
function btnsesudah_Callback(hObject, eventdata, handles)
% hObject    handle to btnsesudah (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
cek1=get(handles.cb1,'Value');
cek2=get(handles.cb2,'Value');
cek3=get(handles.cb3,'Value');
indeks=str2num(get(handles.indeks1,'String'));
total=str2num(get(handles.total1,'String'));
if mod(indeks,1)~=0 || total<1
    set(handles.indeks1,'String',0);
    set(handles.kendala,'String',' ');
    set(handles.cb1,'Value',1);
    set(handles.cb2,'Value',0);
    set(handles.cb3,'Value',0);
    set(handles.batas,'String',' ');
elseif indeks+1>total
    set(handles.indeks1,'String',total);
    set(handles.kendala,'String',handles.simpankendala{total});
    if handles.tanda(total)==-1
        set(handles.cb1,'Value',1);
        set(handles.cb2,'Value',0);
        set(handles.cb3,'Value',0);
    elseif handles.tanda(total)==0
        set(handles.cb1,'Value',0);
        set(handles.cb2,'Value',1);
        set(handles.cb3,'Value',0);
    elseif handles.tanda(total)==1
        set(handles.cb1,'Value',0);
        set(handles.cb2,'Value',0);
        set(handles.cb3,'Value',1);
    end
    set(handles.batas,'String',handles.B(total));
else
    set(handles.indeks1,'String',indeks+1);
    set(handles.kendala,'String',handles.simpankendala{indeks+1});
    if handles.tanda(indeks+1)==-1
        set(handles.cb1,'Value',1);
        set(handles.cb2,'Value',0);
        set(handles.cb3,'Value',0);
    elseif handles.tanda(indeks+1)==0
        set(handles.cb1,'Value',0);
        set(handles.cb2,'Value',1);
        set(handles.cb3,'Value',0);
    elseif handles.tanda(indeks+1)==1
        set(handles.cb1,'Value',0);
        set(handles.cb2,'Value',0);
        set(handles.cb3,'Value',1);
    end
    set(handles.batas,'String',handles.B(indeks+1));
end
guidata(hObject, handles);
```

```
function tfkendala_Callback(hObject, eventdata, handles)
% hObject    handle to tfkendala (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of tfkendala as
text
%        str2double(get(hObject,'String')) returns contents of
tfkendala as a double


% --- Executes during object creation, after setting all
properties.
function tfkendala_CreateFcn(hObject, eventdata, handles)
% hObject    handle to tfkendala (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function tfbatas_Callback(hObject, eventdata, handles)
% hObject    handle to tfbatas (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of tfbatas as text
%        str2double(get(hObject,'String')) returns contents of
tfbatas as a double

% --- Executes during object creation, after setting all
properties.
function tfbatas_CreateFcn(hObject, eventdata, handles)
% hObject    handle to tfbatas (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function indeks2_Callback(hObject, eventdata, handles)
% hObject    handle to indeks2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of indeks2 as text
%        str2double(get(hObject,'String')) returns contents of
indeks2 as a double


% --- Executes during object creation, after setting all
properties.
function indeks2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to indeks2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in btnclear.
function btnclear_Callback(hObject, eventdata, handles)
% hObject    handle to btnclear (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.pembilang,'String','');
set(handles.penyebut,'String','');
handles.simpankendala={};
handles.A=[];
handles.B=[];
handles.tanda=[];
set(handles.total1,'String','');
set(handles.indeks1,'String','');
set(handles.kendala,'String','');
set(handles.cb1,'Value',1);
set(handles.cb2,'Value',0);
set(handles.cb3,'Value',0);
set(handles.batas,'String','');
set(handles.total2,'String','-');
set(handles.indeks2,'String','-');
set(handles.tftujuan,'String','-');
set(handles.konstanta,'String','-');
set(handles.tfkendala,'String','-');
set(handles.cbtf1,'Value',0);
set(handles.cbtf2,'Value',0);
```

```
set(handles.cbtf3,'Value',0);
set(handles.tfbatas,'String','-');
set(handles.ket3,'String','');
set(handles.hasil,'String','-');
set(handles.indeksx,'String','-');
set(handles.hasilx,'String','-');
set(handles.indeksy,'String','-');
set(handles.hasily,'String','-');

set(handles.ket2,'String','Berhasil!');
pause(2);
set(handles.ket2,'String',' ');
guidata(hObject, handles);

% --- Executes on button press in btnsolve.
function btnsolve_Callback(hObject, eventdata, handles)
% hObject    handle to btnsolve (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

l=length(handles.simpankendala);
pilih=get(handles.tujuan,'SelectedObject');
tujuan=get(pilih,'String');
if strcmp(tujuan,'Maksimal')
    handles.nilaitujuan=1;
elseif strcmp(tujuan,'Minimal')
    handles.nilaitujuan=0;
end
if l>=1
    [handles.Z, handles.X, handles.Y, handles.tfA,
handles.tfC,konstanta,normal]=penyelesaian(handles.A,handles.B,han
dles.C,handles.D,handles.alpha,handles.beta,handles.tanda,handles.
nilaitujuan);
    handles.simpantfkendala=konversikembali(handles.tfA);
    total=str2num(get(handles.total1,'String'));
    set(handles.tftujuan,'String',konversikembali(handles.tfC));
    set(handles.konstanta,'String',konstanta);
    set(handles.total2,'String',total);
    set(handles.indeks2,'String',1);
    set(handles.tfkendala,'String',handles.simpantfkendala{1});
    if handles.tanda(1)==-1
        set(handles.cbtf1,'Value',1);
        set(handles.cbtf2,'Value',0);
        set(handles.cbtf3,'Value',0);
    elseif handles.tanda(1)==0
        set(handles.cbtf1,'Value',0);
        set(handles.cbtf2,'Value',1);
        set(handles.cbtf3,'Value',0);
    elseif handles.tanda(1)==1
        set(handles.cbtf1,'Value',0);
        set(handles.cbtf2,'Value',0);
        set(handles.cbtf3,'Value',1);
    end
    set(handles.tfbatas,'String',handles.B(1));
```

```
    set(handles.indeksx,'String',1);
    set(handles.hasilx,'String',handles.X(1));
    set(handles.indeksy,'String',1);
    set(handles.hasily,'String',handles.Y(1));
    set(handles.hasil,'String',handles.Z);
    if normal==0
        set(handles.ket3,'String','PROSES BERJALAN DENGAN
NORMAL!');
    elseif normal==1
        set(handles.ket3,'String','PERSAMAAN MEMILIKI SOLUSI
ALTERNATIF!');
    elseif normal==2
        set(handles.ket3,'String','PERSAMAAN MENGALAMI
DEGENERASI!');
    elseif normal==3
        set(handles.ket3,'String','PROSES BERHENTI, SOLUSI
TAKTERBATAS!');
    elseif normal==4
        set(handles.ket3,'String','PROSES BERHENTI, TIDAK ADA
DAERAH LAYAK!');
    else
        set(handles.ket3,'String','ERROR!!!');
    end
    save datainput handles
end
guidata(hObject, handles);

% --- Executes on button press in btntfsebelum.
function btntfsebelum_Callback(hObject, eventdata, handles)
% hObject    handle to btntfsebelum (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
indeks=str2num(get(handles.indeks2,'String'));
total=str2num(get(handles.total2,'String'));
if indeks<=1 || mod(indeks,1)~=0
    set(handles.indeks2,'String',1);
    set(handles.tfkendala,'String',handles.simpantfkendala{1});
    if handles.tanda(1)==-1
        set(handles.cbtf1,'Value',1);
        set(handles.cbtf2,'Value',0);
        set(handles.cbtf3,'Value',0);
    elseif handles.tanda(1)==0
        set(handles.cbtf1,'Value',0);
        set(handles.cbtf2,'Value',1);
        set(handles.cbtf3,'Value',0);
    elseif handles.tanda(1)==1
        set(handles.cbtf1,'Value',0);
        set(handles.cbtf2,'Value',0);
        set(handles.cbtf3,'Value',1);
    end
    set(handles.tfbatas,'String',handles.B(1));
elseif indeks<=total
    set(handles.indeks2,'String',indeks-1);
```

```
    set(handles.tfkendala,'String',handles.simpantfkendala{indeks-
1});
    if handles.tanda(indeks-1)==-1
        set(handles.cbtf1,'Value',1);
        set(handles.cbtf2,'Value',0);
        set(handles.cbtf3,'Value',0);
    elseif handles.tanda(indeks-1)==0
        set(handles.cbtf1,'Value',0);
        set(handles.cbtf2,'Value',1);
        set(handles.cbtf3,'Value',0);
    elseif handles.tanda(indeks-1)==1
        set(handles.cbtf1,'Value',0);
        set(handles.cbtf2,'Value',0);
        set(handles.cbtf3,'Value',1);
    end
    set(handles.tfbatas,'String',handles.B(indeks-1));
end
guidata(hObject, handles);


% --- Executes on button press in btntfsesudah.
function btntfsesudah_Callback(hObject, eventdata, handles)
% hObject    handle to btntfsesudah (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
indeks=str2num(get(handles.indeks2,'String'));
total=str2num(get(handles.total2,'String'));
if mod(indeks,1)~=0 || total<1
    set(handles.indeks2,'String',0);
    set(handles.tfkendala,'String',' ');
    set(handles.tftanda,'String',' ');
    set(handles.tfbatas,'String',' ');
elseif indeks+1>total
    set(handles.indeks2,'String',total);
    set(handles.tfkendala,'String',handles.simpantfkendala{total});
    if handles.tanda(total)==-1
        set(handles.cbtf1,'Value',1);
        set(handles.cbtf2,'Value',0);
        set(handles.cbtf3,'Value',0);
    elseif handles.tanda(total)==0
        set(handles.cbtf1,'Value',0);
        set(handles.cbtf2,'Value',1);
        set(handles.cbtf3,'Value',0);
    elseif handles.tanda(total)==1
        set(handles.cbtf1,'Value',0);
        set(handles.cbtf2,'Value',0);
        set(handles.cbtf3,'Value',1);
    end
    set(handles.tfbatas,'String',handles.B(total));
else
    set(handles.indeks2,'String',indeks+1);

set(handles.tfkendala,'String',handles.simpantfkendala{indeks+1});
    if handles.tanda(indeks+1)==-1
```

```
        set(handles.cbtf1,'Value',1);
        set(handles.cbtf2,'Value',0);
        set(handles.cbtf3,'Value',0);
    elseif handles.tanda(indeks+1)==0
        set(handles.cbtf1,'Value',0);
        set(handles.cbtf2,'Value',1);
        set(handles.cbtf3,'Value',0);
    elseif handles.tanda(indeks+1)==1
        set(handles.cbtf1,'Value',0);
        set(handles.cbtf2,'Value',0);
        set(handles.cbtf3,'Value',1);
    end
    set(handles.tfbatas,'String',handles.B(indeks+1));
end
guidata(hObject, handles);


% --- Executes on button press in btnhasilsebelum.
function btnhasilsebelum_Callback(hObject, eventdata, handles)
% hObject    handle to btnhasilsebelum (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
indeks=str2num(get(handles.indeksx,'String'));
total=length(handles.C);
if indeks<=1 || mod(indeks,1)~=0
    set(handles.indeksx,'String',1);
    set(handles.hasilx,'String',handles.X(1));
    set(handles.indeksy,'String',1);
    set(handles.hasily,'String',handles.Y(1));
else
    set(handles.indeksx,'String',indeks-1);
    set(handles.hasilx,'String',handles.X(indeks-1));
    set(handles.indeksy,'String',indeks-1);
    set(handles.hasily,'String',handles.Y(indeks-1));
end
guidata(hObject, handles);

% --- Executes on button press in btnhasilsesudah.
function btnhasilsesudah_Callback(hObject, eventdata, handles)
% hObject    handle to btnhasilsesudah (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
indeks=str2num(get(handles.indeksx,'String'));
total=length(handles.C);
if mod(indeks,1)~=0
    set(handles.indeksx,'String',1);
    set(handles.hasilx,'String',handles.X(1));
    set(handles.indeksy,'String',1);
    set(handles.hasily,'String',handles.Y(1));
elseif indeks+1>total
    set(handles.indeksx,'String',total);
    set(handles.hasilx,'String',handles.X(total));
    set(handles.indeksy,'String',total);
```

```
    set(handles.hasily,'String',handles.Y(total));
else
    set(handles.indeksx,'String',indeks+1);
    set(handles.hasilx,'String',handles.X(indeks+1));
    set(handles.indeksy,'String',indeks+1);
    set(handles.hasily,'String',handles.Y(indeks+1));
end
guidata(hObject, handles);


function konstanta_Callback(hObject, eventdata, handles)
% hObject    handle to konstanta (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of konstanta as
text
%        str2double(get(hObject,'String')) returns contents of
konstanta as a double


% --- Executes during object creation, after setting all
properties.
function konstanta_CreateFcn(hObject, eventdata, handles)
% hObject    handle to konstanta (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function pembilang_Callback(hObject, eventdata, handles)
% hObject    handle to pembilang (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of pembilang as
text
%        str2double(get(hObject,'String')) returns contents of
pembilang as a double
[handles.C,handles.alpha]=konversi(get(hObject,'String'));
guidata(hObject, handles);

% --- Executes during object creation, after setting all
properties.
```

```
function pembilang_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pembilang (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function hasil_Callback(hObject, eventdata, handles)
% hObject    handle to hasil (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of hasil as text
%        str2double(get(hObject,'String')) returns contents of
hasil as a double


% --- Executes during object creation, after setting all
properties.
function hasil_CreateFcn(hObject, eventdata, handles)
% hObject    handle to hasil (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function tftujuan_Callback(hObject, eventdata, handles)
% hObject    handle to tftujuan (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of tftujuan as
text
%        str2double(get(hObject,'String')) returns contents of
tftujuan as a double
```

```
% --- Executes during object creation, after setting all
properties.
function tftujuan_CreateFcn(hObject, eventdata, handles)
% hObject    handle to tftujuan (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function totall_Callback(hObject, eventdata, handles)
% hObject    handle to totall (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of totall as text
%        str2double(get(hObject,'String')) returns contents of
totall as a double


% --- Executes during object creation, after setting all
properties.
function totall_CreateFcn(hObject, eventdata, handles)
% hObject    handle to totall (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function total2_Callback(hObject, eventdata, handles)
% hObject    handle to total2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of total2 as text
```

```
%        str2double(get(hObject,'String')) returns contents of
total2 as a double


% --- Executes during object creation, after setting all
properties.
function total2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to total2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function hasily_Callback(hObject, eventdata, handles)
% hObject    handle to hasily (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of hasily as text
%        str2double(get(hObject,'String')) returns contents of
hasily as a double


% --- Executes during object creation, after setting all
properties.
function hasily_CreateFcn(hObject, eventdata, handles)
% hObject    handle to hasily (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function indeksy_Callback(hObject, eventdata, handles)
% hObject    handle to indeksy (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of indeksy as text
%        str2double(get(hObject,'String')) returns contents of
indeksy as a double

% --- Executes during object creation, after setting all
properties.
function indeksy_CreateFcn(hObject, eventdata, handles)
% hObject    handle to indeksy (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function hasilx_Callback(hObject, eventdata, handles)
% hObject    handle to hasilx (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of hasilx as text
%        str2double(get(hObject,'String')) returns contents of
hasilx as a double

% --- Executes during object creation, after setting all
properties.
function hasilx_CreateFcn(hObject, eventdata, handles)
% hObject    handle to hasilx (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function indeksx_Callback(hObject, eventdata, handles)
% hObject    handle to indeksx (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of indeksx as text
%        str2double(get(hObject,'String')) returns contents of
indeksx as a double


% --- Executes during object creation, after setting all
properties.
function indeksx_CreateFcn(hObject, eventdata, handles)
% hObject    handle to indeksx (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit25_Callback(hObject, eventdata, handles)
% hObject    handle to edit25 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit25 as text
%        str2double(get(hObject,'String')) returns contents of
edit25 as a double

% --- Executes during object creation, after setting all
properties.
function edit25_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit25 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit26_Callback(hObject, eventdata, handles)
% hObject    handle to edit26 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit26 as text
%        str2double(get(hObject,'String')) returns contents of
edit26 as a double


% --- Executes during object creation, after setting all
properties.
function edit26_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit26 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit27_Callback(hObject, eventdata, handles)
% hObject    handle to edit27 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit27 as text
%        str2double(get(hObject,'String')) returns contents of
edit27 as a double

% --- Executes during object creation, after setting all
properties.
function edit27_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit27 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit28_Callback(hObject, eventdata, handles)
% hObject    handle to edit28 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit28 as text
%        str2double(get(hObject,'String')) returns contents of
edit28 as a double


% --- Executes during object creation, after setting all
properties.
function edit28_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit28 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in cb1.
function cb1_Callback(hObject, eventdata, handles)
% hObject    handle to cb1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(hObject,'Value',1);
set(handles.cb2,'Value',0);
set(handles.cb3,'Value',0);

% --- Executes on button press in cb2.
function cb2_Callback(hObject, eventdata, handles)
% hObject    handle to cb2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of cb2
set(hObject,'Value',1);
set(handles.cb1,'Value',0);
set(handles.cb3,'Value',0);

% --- Executes on button press in cb3.
```

```
function cb3_Callback(hObject, eventdata, handles)
% hObject    handle to cb3 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(hObject,'Value',1);
set(handles.cb1,'Value',0);
set(handles.cb2,'Value',0);
% Hint: get(hObject,'Value') returns toggle state of cb3


% --- Executes on button press in cbtf1.
function cbtf1_Callback(hObject, eventdata, handles)
% hObject    handle to cbtf1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of cbtf1


% --- Executes on button press in cbtf2.
function cbtf2_Callback(hObject, eventdata, handles)
% hObject    handle to cbtf2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of cbtf2


% --- Executes on button press in cbtf3.
function cbtf3_Callback(hObject, eventdata, handles)
% hObject    handle to cbtf3 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of cbtf3
function
[hasil,X,Y,A,C,konstanta,normal]=penyelesaian(a,b,c,d,alpha,beta,t
anda,tujuan)
B=b;
b=b';
konstanta=alpha/beta;
if all(c==d*konstanta)
    C=(c*beta-d*alpha)/beta;
    A=a*beta+b*d;
    hasil=konstanta;Y=zeros(n,1);X=Y;normal=1;
elseif beta>0
    C=(c*beta-d*alpha)/beta;
    A=a*beta+b*d;
```

```
    [z,Y,normal]=simpleksdirevisi(A,B,C,tanda,tujuan);
    if normal>=3
        hasil=0;X=Y;
    elseif normal<3
        Y=Y';
        X=(Y*beta)/(1-d*Y);
        hasil=z+konstanta;
    end
end
function [Z,X,normal]=simpleksdirevisi(a,b,c,tanda,tujuan)
n=length(c);
m=length(b);
M=max(abs(c))*10;
while max(max(a))> M
    M=M*10;
end
if tujuan==0
    c=-c;
end
hitung=n;
VD=zeros(1,m);
BVD=1:n;
VB=zeros(1,m);
for i=1:m
    if b(i)<0
        a(i,:)=-a(i,:);
        b(i)=-b(i);
    end
    if tanda(i)<0
        hitung=hitung+1;
        c(hitung)=0;
        a(i,hitung)=1;
        VD(i)=hitung;
    elseif tanda(i)==0
        hitung=hitung+1;
        c(hitung)=-M;
        a(i,hitung)=1;
        VD(i)=hitung;
        VB(i)=hitung;
    else
        hitung=hitung+1;
        c(hitung)=0;
        a(i,hitung)=-1;
        BVD=[BVD hitung];
        hitung=hitung+1;
        c(hitung)=-M;
        a(i,hitung)=1;
        VB(i)=hitung;
        VD(i)=hitung;
    end
end
A=[-c;a];
B_inv=eye(m+1,m+1);
B_inv(1,2:m+1)=c(VD);
```

```
x_b=B_inv*[0; b'];
berhenti=0;
hitung=0;
simpan=0;
while(berhenti~=1)
    [s,t]=min(B_inv(1,:)*A(:,BVD));
    v=B_inv*A(:,BVD(t));
    hitung=hitung+1;
    if(any(v(2:m+1)>0))
        if hitung>1 && simpan==x_b(1)
            berhenti=1;
            if tujuan==0
                x_b(1)=-x_b(1);
            end
            normal=2;
            Z=x_b(1);
            for i=1:n
                ada=0;
                for j=1:m
                    if VD(j)==i
                        X(i)=x_b(1+j);
                        ada=1;
                    end
                end
                if ada==0
                    X(i)=0;
                end
            end
        else
            simpan=x_b(1);
            if(s>=0)
                berhenti=1;
                for i=1:length(VB)
                    for j=1:m
                        if VB(i)==VD(j)
                            Z=0;
                            X=0;
                            normal=4;
                            return
                        end
                    end
                end
                if tujuan==0
                    x_b(1)=-x_b(1);
                end
                normal=0;
                Z=x_b(1);
                for i=1:n
                    ada=0;
                    for j=1:m
                        if VD(j)==i
                            X(i)=x_b(1+j);
                            ada=1;
                        end
                    end
                    if ada==0
```

```
                    X(i)=0;
                end
            end
            if (s==0 && any(v(2:m+1)>0))
                normal=1;
            end
        else
            u=M;
            for i=2:m+1
                if v(i)>0
                    if (x_b(i)/v(i))<u
                        u=(x_b(i)/v(i));
                        k=i-1;
                    end
                end
            end
            ganti=VD(k);
            VD(k)=BVD(t);
            BVD(t)=ganti;
            E=eye(m+1,m+1);
            E(:,1+k)=-v/v(1+k);
            E(1+k,1+k)=1/v(1+k);
            B_inv=E*B_inv;
            x_b=B_inv*[0; b'];
        end
    end
else
    Z=0;
    X=0;
    normal=3;
    return
end
end

function hasil=potong(K)
l=length(K);
j=1;
for i=1:l
    if ~isempty(K{i})
        hasil{j}=K{i};
        j=j+1;
    end
end


function [hasil,konstanta]=konversi(s)
variabel=[];
operasi=[];
if ~strcmp(s(1),'-')
    s=['+',s];
end
L=length(s);
for i=1:L
```

```
    if strcmp(s(i),'x')
        variabel=[variabel,i];
    end
    if strcmp(s(i),'+') || strcmp(s(i),'-')
        operasi=[operasi,i];
    end
end
lv=length(variabel);
lo=length(operasi);
for i=1:lv
        hasil(i)=str2num(s(abs(operasi(i)):variabel(i)-1));
end
if lo>lv
    konstanta=str2num(s(abs(operasi(end)):end));
else
    konstanta=0;
end


function hasil=konversikembali(A)
[b,k]=size(A);
hasil={};
for i=1:b
    ruang=[];
    for j=1:k
        if A(i,j)<0
            ruang=[ruang,num2str(A(i,j)),'y',num2str(j)];
        else
            ruang=[ruang,'+',num2str(A(i,j)),'y',num2str(j)];
        end
    end
    if strcmp(ruang(1),'+')
        ruang(1)=[];
    end
    hasil{i}=ruang;
end
```

SURAT PERNYATAAN KEASLIAN SKRIPSI

Dengan ini saya yang bertanda tangan di bawah ini, mahasiswa Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Jakarta:

Nama	:	Bobby Reynaldo
No. Registrasi	:	3125121983
Jurusan	:	Matematika
Program Studi	:	Matematika

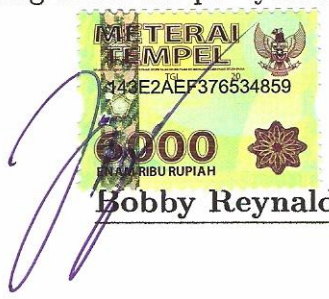
Menyatakan bahwa skripsi ini yang saya buat dengan judul "**Pengembangan Program Pecahan Linier dengan Transformasi Aljabar**" adalah :

1. Dibuat dan diselesaikan oleh saya sendiri.
2. Bukan merupakan duplikat skripsi yang pernah dibuat oleh orang lain atau jiplakan karya tulis orang lain.

Pernyataan ini dibuat dengan sesungguhnya dan saya bersedia menanggung segala akibat yang timbul jika pernyataan saya tidak benar.

Jakarta, Februari 2017

Yang membuat pernyataan



Bobby Reynaldo

DAFTAR RIWAYAT HIDUP

BOBBY REYNALDO. Lahir di Jakarta, 16 April 1994. Anak ke empat dari pasangan Bapak Mangantar Pangaribuan dan Ibu Rotua Huta Julu. Saat ini bertempat tinggal di Jalan Pesona I RT 01/ RW 05 No 38, Jakarta Timur 13790.

No. Ponsel : 085778011108

Email : bobbyreynaldo@gmail.com

Riwayat Pendidikan : Awal Pendidikan penulis di TK Kartika selama 1 tahun, kemudian melanjutkan pendidikan di SD Negri Srengseng Sawah 07 Pagi pada tahun 2000 - 2006. Setelah itu, penulis melanjutkan ke SMP Negri 179 Jakarta pada tahun 2006 - 2009. Kemudian kembali melanjutkan ke SMA Negeri 98 Jakarta dan lulus pada tahun 2012. Di Tahun yang sama penulis melanjutkan ke Universitas Negeri Jakarta (UNJ), jurusan Matematika, melalui jalur SNMPTN Tulis. Ditahun 2017 penulis telah memperoleh gelar Sarjana Sains untuk Jurusan Matematika, Program Studi Matematika, FMIPA, UNJ.
Riwayat Organisasi : Selama di bangku perkuliahan, penulis aktif di beberapa organisasi kemahasiswaaan. Dalam tahun pertama, penulis menjadi staff Divisi Rental Komputer di KOPMA UNJ. Tahun kedua penulis dipercaya untuk menjadi staff Advokasi dan Olahraga BEMJ Matematika.