# KOMPUTASI METODE SIMPLEKS PADA PENYELESAIAN PROGRAM LINIER

**Akhmad Yusuf dan Dewi Sri Susanti**
Program Studi Matematika
Universitas Lambung Mangkurat
Jl. Jend. A. Yani km. 36 Kampus Unlam Banjarbaru

## ABSTRAK

Salah satu metode yang dapat digunakan untuk menyelesaikan permasalahan program linier adalah metode simpleks. Pada penelitian ini, peneliti akan mencoba membuat algoritma komputasi metode simpleks yang dapat digunakan untuk penyelesaian program linier, sehingga diperoleh rangkaian proses penyelesaian yang efisien dan hasil akhir yang optimal. Penelitian ini menghasilkan kode program Komputasi menggunakan bahasa pemrograman Borland Delphi 6.0 yang dapat digunakan untuk menyelesaikan persoalan program linier dengan menggunakan metode simpleks.

Kata Kunci : *Program Linier, Algoritma Metode Simpleks*.

## 1. PENDAHULUAN

Salah satu metode analisis matematika yang cukup memadai untuk menyelesaikan persoalan alokasi sumber daya adalah metode program linier. Teknik program linier adalah merumuskan masalah dengan jelas dan menggunakan sejumlah informasi yang tersedia untuk mendapatkan penyelesaian akhir yang optimal. Setelah masalah terumuskan dengan baik, maka langkah berikut ialah menerjemahkan masalah ini ke dalam bentuk model matematika, untuk mendapatkan cara penyelesaian eksak yang lebih mudah dan rapi guna menemukan jawaban terhadap masalah tersebut (Siagian, 1987).

Persoalan program linier adalah suatu persoalan untuk menentukan besarnya masing-masing nilai variabel sedemikian sehingga nilai fungsi tujuan atau fungsi obyektif yang linier menjadi maksimum atau minimum dengan memperhatikan batasan yang ada. Salah satu metode yang dapat digunakan untuk menyelesaikan permasalahan tersebut dengan menggunakan metode simpleks (Supranto, 1983). Tulisan ini merupakan hasil penelitian yang bertujuan untuk membuat suatu algoritma komputasi metode simpleks dalam menyelesaikan persoalan program linier yang dinyatakan dalam bahasa pemrograman *Borland Delphi* 6.0.

## 2. TINJAUAN PUSTAKA

### 2.1. Masalah Program Linier

Masalah program linier menurut Supranto (1983) adalah suatu persoalan untuk menentukan besarnya masing-masing nilai variabel sedemikian hingga nilai dari fungsi tujuan yang bersifat linier menjadi optimum (maksimum atau minimum) dengan memperhatikan batasan-batasan dari nilai variabel penentu, dimana pembatasan variabel inputnya dapat dinyatakan sebagai pertidaksamaan linier. Secara umum masalah program linier dapat dinyatakan sebagai berikut:

Terdapat fungsi tujuan : $Z = c_1x_1 + c_2x_2 + \cdots + c_nx_n$, kemudian ditentukan nilai $x_1, x_2, \cdots, x_n$ sehingga fungsi tujuan optimum dengan beberapa fungsi pembatas sebagai berikut :

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &<=> b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &<=> b_2 \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots & \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &<=> b_m \\ x_1 \geq 0, x_2 \geq 0, \ldots, x_n &\geq 0 \end{aligned}$$

dimana :

Z = fungsi tujuan

$x_j$ = banyaknya output ke-*j* yang dihasilkan, dimana j = 1,2, ... , n.

$b_i$ = banyaknya input ke-*i* yang diperlukan, dimana i = 1,2, ... , m.

$a_{ij}$ = banyaknya output ke-*j* yang dihasilkan untuk satu satuan input ke-*i*, dimana i = 1,2, ..., m; j = 1,2, ... , n.

$C_j$ = banyaknya keuntungan yang dihasilkan dari satu satuan output ke-*j*, dimana j = 1,2, ..., n.

Masalah Program Linier pada dasarnya terbagi dua, yaitu persoalan memaksimumkan dan persoalan meminimumkan dari suatu fungsi tujuan. Masalah Program Linier berupa persoalan memaksimumkan dinamakan persoalan standar, seperti contoh di bawah ini :

Menentukan $x_1, x_2, \cdots, x_n$ sehingga $Z = c_1x_1 + c_2x_2 + \cdots + c_nx_n$ maksimum dengan fungsi pembatas :

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &\leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &\leq b_2 \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots & \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &\leq b_m \\ x_1 \geq 0, x_2 \geq 0, \ldots, x_n &\geq 0 \end{aligned}$$

Sedangkan persoalan Program Linier berupa persoalan meminimumkan dapat dirumuskan sebagai berikut :

Menentukan $x_1, x_2, \cdots, x_n$ sehingga $Z = c_1x_1 + c_2x_2 + \cdots + c_nx_n$ Minimum dengan fungsi pembatas :

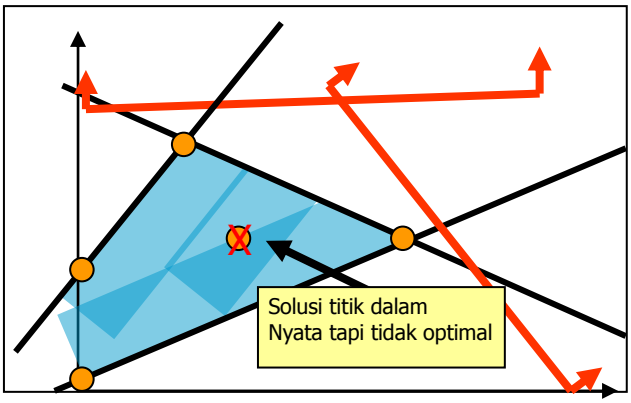
$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &\geq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &\geq b_2 \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots & \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &\geq b_m \\ x_1 \geq 0, x_2 \geq 0, \ldots, x_n &\geq 0 \end{aligned}$$

## 2.2. Teori Metode Simpleks

Lawrence (2008) menyatakan bahwa solusi optimal dari persoalan program linier harus merupakan solusi titik ujung yang tampak (*corner-point feasible*/CPF). Jika terdapat pilihan nilai optimal, maka minimal terdapat 2 titik yang memiliki nilai terdekat terhadap solusi CPF. Secara umum terdapat nilai-

nilai solusi CPF yang berhingga banyaknya. Suatu solusi CPF akan bersifat optimal jika tidak terdapat solusi CPF lain yang lebih baik (mendekati). Gambar 2 menunjukkan beberapa titik yang merupakan solusi CPF dari suatu program linier. Banyaknya solusi CPF yang berhingga dapat ditunjukkan sebagai berikut :

$$\begin{pmatrix} \text{variables} + p\text{embatas} \\ \text{pembatas} \end{pmatrix} = \begin{pmatrix} n+m \\ m \end{pmatrix} = \frac{(m+n)!}{m!n!}$$




Gambar 2. Solusi Titik Ujung

Suatu contoh, jika terdapat $m$ = 50 fungsi pembatas dan $n$ = 100 variabel, maka terdapat solusi CPF sebanyak : $\frac{(m+n)!}{m!n!} = \frac{(50+100)!}{50!100!} \cong 2.01\times 10^{40}$.

Aturan umum dalam penggunaan Metode Simpleks untuk mendapatakan solusi optimal suatu program linier diantaranya adalah :

1. Penentuan variable input dasar (*incoming basic variable*) dari suatu himpunan variabel non-dasar berdasarkan nilai paling negatif dari baris ke-Z
2. Penentuan variabel output dasar (*outgoing basic variable*) sesuai dengan nilai minimum rasio terhadap koefisien variabel input
3. Perlu diingat bahwa pembagian setiap nilai dengan nol akan menghasilkan nilai tak berhingga
4. Jika terdapat koefisien pada variabel input bernilai negatif maka mengindikasikan adanya kesalahan
5. Perlu dihindari penggunaan nilai negatif pada kolom variable input dasar
6. Dilakukan pem-pivot-an dengan cara membagi setiap elemen pada baris pivot dengan elemen pivot
7. Pem-pivotan dilakukan berulang-ulang sampai diperoleh kolom variable input dasar merupakan bagian dari matrik identitas
8. Langkah dihentikan saat diperoleh semua nilai pada baris Z lebih besar atau sama dengan nol.

### 2.3. Penyelesaian Metode Simpleks Standart

Masalah program linier dapat diselesaikan secara grafik, akan tetapi hampir seluruh problem program linier sesungguhnya tidak dapat diselesaikan dengan cara grafik. Oleh karena itu, George Dantzig pada tahun 1947 mengajukan suatu metode yang paling berhasil untuk menyelesaikan problem program linier yang disebut Metode Simpleks.

Sebagai panduan untuk memahami langkah-langkah penyelesaian metode simpleks standar, berikut ini diberikan suatu contoh soal.

Fungsi tujuan yang akan dimaksimumkan : $Z = 3x_1 + 2x_2$

Fungsi pembatas sebagai berikut :

$$x_1 + x_2 \le 15$$
$$2x_1 + x_2 \le 28$$
$$x_1 + 2x_2 \le 20$$
$$x_1 \ge 0,\ x_2 \ge 0$$

Langkah-langkah penyelesaian :

1. Mengubah fungsi linier ke dalam bentuk baku, dengan cara:
   Fungsi Pembatas (syarat/*constraint*) :
   - Perubahan bentuk pertidaksamaan menjadi persamaan dilakukan dengan memberikan variabel *slack* ($S_n$) yang dapat mengurangi variabel surplus.
   - Bila ruas kanan pada fungsi pembatas bernilai negatif, maka setiap ruas dikalikan dengan –1.
   - Bila fungsi tujuan dalam bentuk pertidaksamaan, maka perkalian dengan –1 akan mengubah tanda (misal tanda awal ≥ menjadi ≤).
   - Variabel *unrestricted*, yaitu variabel yang dapat bernilai positif maupun negatif, dapat diekspresikan dalam dua variabel non negatif, yaitu :

     $$x_j = x'_j - x''$$

     dimana $x_j$ adalah variabel *unrestricted* (tidak dibatasi) dan $x'_j \ge 0, x'' \ge 0$.

     Pada kasus di atas, karena tidak ada variabel yang *unrestricted* maka bentuk bakunya menjadi :

$$Z - 3x_1 - 2x_2 - 0S_1 - 0S_2 - 0S_3 = 0$$
$$x_1 + x_2 + S_1 = 15$$
$$2x_1 + x_2 + S_2 = 28$$
$$x_1 + 2x_2 + S_3 = 20$$

2. Berikutnya dibuat tabel simpleks awal
   Tabel simpleks awal yang tampak pada Tabel 1 merupakan solusi awal masalah. Solusi awal ditentukan dengan menetapkan variabel non basis (non dasar, $x_j$) sama dengan 0. Dengan menetapkan $x_1 = 0$ dan $x_2 = 0$, maka diperoleh $S_1 = 15$, $S_2 = 28$ dan $S_3 = 20$. Nilai–nilai dalam tabel simpleks awal merupakan nilai-nilai elemen (koefisien dari persamaan).

Tabel 1. Tabel simpleks awal

| **Basis** | **$x_1$** | **$x_2$** | **$S_1$** | **$S_2$** | **$S_3$** | **Solusi** |
|---|---|---|---|---|---|---|
| Z | -3 | -2 | 0 | 0 | 0 | 0 |
| $S_1$ | 1 | 1 | 1 | 0 | 0 | 15 |
| $S_2$ | 2 | 1 | 0 | 1 | 0 | 28 |
| $S_3$ | 1 | 2 | 0 | 0 | 1 | 20 |

3. Menentukan *incoming variable*, yaitu variabel non basis yang bila nilainya dinaikkan dari nol dapat memperbaiki nilai fungsi tujuan. Yang diutamakan untuk dipilih sebagai *incoming variable* adalah variabel dengan koefisien positif terbesar karena pengalaman menunjukkan bahwa pemilihan ini

mengakibatkan solusi optimal lebih cepat tercapai. Untuk kasus di atas, *incoming variable*nya adalah $X_1$.

4. Pilih *outgoing variable*, yaitu variabel basis ($S_1$, $S_2$, $S_3$) yang harus menjadi non basis (nilainya menjadi nol) ketika *incoming variable* menjadi variabel basis. *Outgoing variable* adalah variabel basis yang memiliki rasio terkecil antara sisi kanan persamaan kendala (solusi) dengan koefisien positif *incoming varible*. Secara lengkap ditampilkan pada Tabel 2.

Tabel 2. Rasio Solusi

| **Basis** | **$X_1$** | **$X_2$** | **$S_1$** | **$S_2$** | **$S_3$** | **Solusi** | **Rasio** |
|---|---|---|---|---|---|---|---|
| Z | -3 | -2 | 0 | 0 | 0 | 0 | |
| $S_1$ | 1 | 1 | 1 | 0 | 0 | 15 | 15/1 =15 |
| $S_2$ | (2) | 1 | 0 | 1 | 0 | 28 | 28/2 = 14 |
| $S_3$ | 1 | 2 | 0 | 0 | 1 | 20 | 20/1 = 20 |

Karena nilai rasio $S_2$ paling kecil di antara nilai rasio dua variabel basis lain, maka *outgoing variable* untuk kasus ini adalah $S_2$.

5. Tentukan *incoming column, pivot equation, pivot element*.
   *Incoming column* adalah kolom *incoming variable*. Untuk kasus ini, *incoming column* = $X_1$. Nilai-nilai pada kolom tersebut merupakan *elemen incoming column*. *Pivot equation* yaitu baris di mana terdapat *outgoing variable*. Untuk kasus ini, *pivot equation* adalah $S_2$. *Pivot element* adalah elemen pada perpotongan antara *incoming column* dengan *pivot equation*. Untuk kasus ini, *pivot element* adalah 2 (nilai dalam tanda kurung pada Tabel 2).
6. Untuk menetapkan solusi dasar baru, dilakukan perhitungan menggunakan metode Gauss Jordan:
   Untuk menentukan elemen persaman pivot baru :

$$elemen\ pers.\ pivot\ baru = \frac{elemen\ pers.\ pivot\ lama}{elemen\ pivot}$$

   Contoh :
   Untuk elemen persamaan pivot baru :

   $S_2 - x_1 = 2/2 = 1$
   $S_2 - x_2 = 1/2 = ½$
   $S_2 - S_1 = 0/2 = 0$

   dan seterusnya sehingga seluruh nilai elemen persamaan pivot baru dapat disusun dalam Tabel simpleks iterasi pertama seperti disajikan pada Tabel 3.

Tabel 3. Rasio Simpleks Iterasi Pertama

| **Basis** | **$X_1$** | **$X_2$** | **$S_1$** | **$S_2$** | **$S_3$** | **Solusi** | **Rasio** |
|---|---|---|---|---|---|---|---|
| Z | 0 | -1/2 | 0 | 3/2 | 0 | 42 | |
| $S_1$ | 0 | 1/2 | 1 | -1/2 | 0 | 1 | 2 |
| $X_1$ | 1 | 1/2 | 0 | ½ | 0 | 14 | 28 |
| $S_3$ | 0 | 3/2 | 0 | -1/2 | 1 | 6 | 4 |

Untuk menentukan elemen semua persaman lain termasuk Z, digunakan rumus dari Metode Gauss Jordan :

$$elemen\ pers.\ tabel\ baru = elemen\ pers.\ tabel\ lama - (elemen\ entering\ column\ x\ elemen\ pers.\ pivot\ baru)$$

Contoh soal :

Untuk elemen persamaan Z baru : $Z - x_1 = -3 - (-3 * 1) = 0$

$Z - x_2 = -2 - (-3 * \frac{1}{2}) = -\frac{1}{2}$

$Z - S_1 = 0 - (-3 * 0) = 0$

dan seterusnya.

Solusi yang baru memberikan nilai $x_1 = 14$ dan $x_2 = 0$, dan mendapatkan Z = 14. Karena $x_2$ masih bernilai 0, maka masih memungkinkan untuk menaikkan nilai Z dengan jalan menaikkan nilai $x_2$. Berdasarkan Tabel Simpleks Iterasi pertama, akan dilakukan lagi langkah-langkah dari nomor 3 dan seterusnya, sehingga diperoleh tabel simpleks optimum seperti disajikan pada Tabel 4. Solusi baru memberikan $x_1 = 13$, $x_2 = 2$ dan Z = 43. Dikatakan optimum karena tak ada lagi variabel non basis yang memiliki koefisien negatif pada persamaan Z (ditunjukkan pada Tabel 4 untuk Z di mana $x_1 = 0$ dan $x_2 = 0$).

Tabel 4. Tabel Simpleks Optimum

| **Basis** | **$X_1$** | **$X_2$** | **$S_1$** | **$S_2$** | **$S_3$** | **Solusi** |
|---|---|---|---|---|---|---|
| Z | 0 | 0 | 1 | 1 | 0 | 43 |
| $X_2$ | 0 | 1 | 2 | -1 | 0 | 2 |
| $X_1$ | 1 | 0 | -1 | 1 | 0 | 13 |
| $S_3$ | 0 | 0 | -3 | 1 | 1 | 3 |

Salah satu bahasa pemrograman berbasis windows adalah *Borland Delphi*. Menurut Alam (2001), *Delphi* merupakan suatu paket bahasa pemrograman yang bekerja dengan sistem operasi *windows* yang mempunyai cakupan fasilitas yang luas dan sangat canggih. Berbagai perangkat lunak komputer dapat dibuat menggunakan *Delphi*, termasuk angka, teks, database, juga perangkat lunak lainnya. *Delphi* menggunakan dasar bahasa pemrograman Pascal, yaitu bahasa pemrograman yang merupakan perluasan dari bahasa pascal yang berorientasi obyek.

## 3. METODE PENELITIAN

Penelitian ini merupakan penelitian yang bersifat studi literatur dan percobaan laboratorium. Metodologi yang digunakan adalah dengan mengumpulkan referensi mengenai bahasa pemrograman Borland Delphi 6.0 dan metode simpleks untuk menyelesaikan program linier. Selanjutnya menyusun tahapan penyelesaian metode simpleks dan mengaplikasikan ke dalam bahasa pemrograman.

## 4. HASIL DAN PEMBAHASAN

### 4.1. Tahapan Metode Simpleks

Metode simpleks menurut Hieller dan Lieberman (1995) merupakan suatu prosedur aljabar yang merupakan penjabaran suatu prosedur geometris. Prosedur aljabar adalah prosedur yang lebuh mudah dipahami secara intuitif karena dapat digambarkan dalam bentuk grafik, sedangkan prosedur aljabar adalah prosedur yang memudahkan perhitungan dan memudahkan pembuatan algoritma komputer.

Proses iterasi dalam pemrograman terdiri dari tiga langkah iterasi yang masing-masing dinamakan *step* 1, *step* 2 dan *step* 3

**4.2. Kode Program untuk Metode Simpleks**

Sesuai dengan bahasa pemrograman yang digunakan, yaitu Borland Delphi yang merupakan bahasa pemrograman berorientasi obyek, maka struktur data didefinisikan sebagai sebuah obyek. Tipe obyek yang akan menangani penyelesaian dengan metode simpleks didefinisikan dalam sebuah kelas yang dinamakan TSimpleks. Untuk lebih jelasnya tipe kelas TSimpleks didefinisikan sebagai berikut :

```
TSimpleks = class
    private
      a : RealArray;
      jKol,  JBar :  Integer;
    public
      jVar,  jPer    : integer;
      Step    : proses;
      Iterasi         : integer;
      constructor create (elemen: RealArray; jV, jP : integer);
      destructor destroy; override;
      function       optimal: boolean;
      function       NilaiFunqsi : Real;
      procedure    Step1 (var kp word);
      procedure    Step2 (Var bp : word);
      procedure    Step3 (kp, bp   : word);
      procedure    TulisAwal (Var grid : TStringGrid);
      procedure    TulisAkhir(var grid:TStringGrid);
    end;
      Type realArray   :  array of array of real;
```

Setiap anggota kelas baik yang private maupun public mempunyai peranan masing-masing dalam metode simpleks. Deskripsi tugas masing-masing anggota kelas TSimpleks adalah sebagai berikut:

(1) Anggota *private*

Terdapat tiga variabel yang menjadi anggota private dan kelas TSimpleks, yakni **a, JBar, dan Jkol**. Variabel **a** adalah sebuah matrik real berdimensi 2 guna menangani data persamaan. Oleh karena sifatnya interaktif, dalam artian pemakai bebas menentukan ukuran persamaan yang dimasukkan, maka matrik **a** harus bersifat dinamis. Oleh karena itu matrik **a** harus bertipe RealArray yang didefinisikan :

```
Type RealArray   =  array of array of real;
Var a  :  RealArray;
```

Sedangkan variabel **JBar** dan **Jkol** adalah bertipe integer yang menunjukkan ukuran kolom dan baris sel tempat menempatkan matrik.

(2) Anggota *Public*

Vaniabel *public* **JVar** dan **JPer** adalah bertipe integer yang masing-masing menunjukan jumlah variabel dan jumlah persamaan. Sedangkan variabel **step**

menunjukan proses yang sedang berlangsung. Oleh karena itu dibuat suatu tipe enumerasi proses dengan definisi :

```
Type proses=(Awal, Setup, Inisialisasi, TesOptimal,
        Step1, Step2 , Step3);
Var step : proses;
```

(3) Anggota Public Prosedur dan fungsi

(a) Constructor **create**

Prosedur **create** berfungsi untuk membentuk obyek dan kelas **TSimpleks** dengan masukan berupa matrik utama bertipe RealArray sebagai hasil pembacaan dan masukan pemakai. Sebagai parameter tambahan ada jv dan jp yang masing-masing menyatakan jumlah variabel dan jumlah persamaan. Oleh karena itu prosedur create didefinisikan sebagai :

```
constructor Tsimpleks. create (elemen: RealArray; jV, jP : integer);
var
    kol, bar :  integer;
begin
    Step: =SetUp;
    Iterasi   :=0;
    jVar:=jv;
    jPer :=jp;
    jKol :=jV+jp+2;
    jBar :jp+1;
    setLength(a,  jKol,  jBar);
    for bar:=O to jEar-I. do
    for kol:=0   to   jKol-1 do
    a[kol, bar] :=elemen[kol, bar];
end;
```

(b) Destructor **destroy**

Prosedur destroy berfungsi untuk mendealokasikan memori yang sebelumnya dipakai obyek TSimpleks. Memori terbesar di pakai oleh matrik utama yang maksimal berukuran 116 x 232 elemen real. Oleh karena itu deklarasi prosedur destroy sebagai berikut:

```
destructor TSimpleks.destroy ;
begin
        a : =nil;
end;
```

(c) fungsi **optimal**

Fungsi adalah fungsi yang menunjukkan optimal tidaknya penyelesaian yang diperoleh. Fungsi optimal dideklarasikan sebagai :

```
function TSimpleks.optimal: boolean;
var
    kol : integer;
begin
    optimal :=true;
    for   kol:=1 to jPer+jVar+1 do
        if (a[kol, 0) < 0) then Optimal : false;
end;
```

(d) Fungsi **NilaiFungsi**

**NilaiFungsi** menyatakan nilai fungsi dalam setiap langkah iterasi. Nilai fungsi diambil langsung dan matrik a pada baris 0 kolom terakhir. Oleh karena deklarasinya adalah :

```
function TSimpleks.NilaiFungsi : Real;
begin
    NilaiFungsi : =a [jPer+jVar+l, 0];
end;
```

(e) Prosedur **Step 1**

Step 1 dalam iterasi adalah menentukan kolom pivot. Sehingga prosedur untuk menentukan kolom pivot sebagai berikut :

```
Procedure Tsimpleks.Step1;
    var
      kol : word;
      mm : real;
    begin
      kolPivot:=0;
      mm : =0;
      if a[1,0]<a [2,0] then
    begin
      kolPivot:=1;
      mm :=a[l,O];
    end else
    begin
      kol.Pivot:=2;
      min:=a[2,O];
    end;
    for kol:=3 to jKol-l do
    begin
      if (a[kol,0]<min) then
      begin
          min:=a[kol,0);
          kolPivot : =kol;
      end;
    end; end;
```

(f) Prosedur **Step2**

Step 2 dan iterasi adalah mencari baris pivot. Oleh karena itu deklarasi prosedur Step 2 dinyatakan sebagai:

```
Procedure TSimpleks.Step2;
var
    kp, bar : word;
    aKol : array of real;
    min . real;
begin
    barPivot : =0;
    setLength (akol, jBar);
    kp:=Simpleks .kolPivot;
```

```
    for bar:=1 to JPer do
        if (a[jPer+jVar+l,bar]>O)and (a[kp,bar]>O)
then
        aKol [bar-1] :=a[jPer+jVar+1 ,bar] /a[kp,bar]
        else aKol[bar—1]:=1000000;
    min:=0;
    if akol[0]<akol[1] then
    begin
        min:=akol[0];
        barPivot : =1;
    end else
  begin
    barPivot:=2;
    min:=akol[1);
end;
    for bar:=2 to JPer do
    if akol[bar-l]<min then
    begin
        min:=akol [bar-1);
        barPivot : =bar;
    end;
    akol:=nil;
end;
```

(g) Prosedur **Step3**

Setelah ditemukan baris dan kolom pivot, selanjutnya dilakukan operasi baris elementer untuk memperoleh penyelesaian dari persoalan program linear. Deklarasi dan prosedur Step 3 adalah :

```
procedure Tsimpleks.Step3 (kp, bp : word);
var
    kol, bar : word;
    pv : real;
    am, bm real;
    det, br : array of real;
begin
    pv :a[kp,bp);
    setLength (det, jPer+1);
    setLength (br,jPer+jVar+2);
    for bar:=0 to jPer do // ambil kolom pivot
        det[bar]:=a[kp, bar);
    for kol:=0 to Per+jVar+l do 1// bagi baris
pivot dgan pivot
        a(kol, bP] :=a[kol,bPj/pv;
            for bar:=0 to Per do
    if bar<> bp then
    begin
        for kol:=0 to jper+jVanl+1 do
            br[kol] :=-(det[barj*a[kol,bp]);
        for kol:=0 to Per+jVar+1 do
```

```
                    a[kol,bar] :a[kol,barJ+br[kol);
                end;
            end;
```

(h) Prosedur **TulisAwal**

Setelah dilakukan perhitungan untuk mencari penyelesaian persoalan program linear, maka selanjutnya matrik a dituliskan ke dalam sekelompok sel. Prosedur TulisAwal adalah prosedur untuk menuliskan matrik a ke dalam format sel sebelum adanya penambahan slack variabel. Tempat menuliskan basil perhitungan dipilih komponen visual String grid yang dideklarasikan sebagai grid. Selanjutnya deklarasi prosedur tulisAwal sebagai berikut :

```
procedure TSimpleks.tulisAwal(vargrid;TStringGrid);
var  kol, bar : word;
begin
    grid.ColCount:=jVaR.+4; // set ukuran
    grid Rowcount : =j Per+3;
    for kol:=0 to jVar+3 do
    for bar:=0 to JPer+2 do
    grid.Cefls[kol, bar] :=‘  ‘;
    grid.CellS [0,0] :=‘          No’; // nomor persamaan
    grid.Cells [0,1) :=‘ Pers.’;
    grid.Cells[1,0):=’     Variabel’; // variabel dasar
    grid.Cells[1,1] :=‘ dasar’;
    grid.Cells[2,1]:=’       Z’; 1/ Z di bans judul
    for kol:=1 to jVar do // Xl, X2, ... Xxn
    grid.Cefls[2+kol,1) :=‘       X’+inttostr(kol);
    for bar:=0 to jPer do /1 (0), (1), ... (n)
    grid.Cells[0,21-bar] :=‘b(‘+inttostr (bar) +‘)‘;
    grid. Cells [1,2] :‘       Z’;       //  Z pada variabel dasar
    grid.Cells[jVar+3,0):=’  Ruas’; // ruas kanan
    grid.Cells(jVar+3,1] :=‘ kanan’;
    grid.Cells[jVar+3,2] :=‘0’;
    for bar:1 to jPer do // mengisi ruas kanan
    grid.Cefls [jVar+3 ,2+bar] :=FloatToStr (a[jVar+jPer +1,bar]);
for kol:0 to jVar do
for bar:0 to jPer do
    grid.Cells[2+kol,      2+bar] :=FloatToStr(a[kol,bar]);
    grid.Visible :=true;
end;
```

(i) Prosedur **TulisAkhir**

Path proses selanjutnya diperlukan penambahan slack variable sehingga program harus dapat menampilkan matrik a ke dalam string grid yang dideklarasikan sebagai grid dengan menambah alolasi sel untuk slack variabel. Deklarasi dari prosedur tulis akhir dapat dinyatakan sebagai berikut:

```
procedure TSimpleks.tulis(var grid:TStringGrid);
  var
```

```
kol, bar : word;
begin
    grid.ColCount:=jVar+Per+4; // set ukuran
    grid .RowCount :jPer+3;
    for kol:=0 to jPer+jVar+3 do
    for bar:=0 to jPer+2 do grid.Cells [kol bar] :=‘  ‘;
        grid.Cells[0,0]:=’    No’; // nomor persamaan
        grid.Cells[0,1] :=‘ Pers.’;
        grid.Cells[1, 0] :‘ Variabel’; // variabel dasar
        grid.Cells[1,1] :=‘ dasar’;
        grid.Cells[2,1]:=’    Z’; // Z di baris judul
    for kol:=1 to jVar+jper do // Xl, X2, …Xn
        grid.Cells[2+kol,l] :=‘       X’+inttostr(kol);
    for bar:=0 to jPer do // (0), (1), ... (n)
        grid.Cells(0, 2+bar] :=‘(‘+inttostr(bar)+’)’;
        grid.Cefls[l,2]:=’       Z’; // Z pada vaiabel dasar
        grid.Cells [jVar+JPer+3 , 0] := Ruas’; //1 ruas kanan
        grid.Cells[jVar+jPer+3,0) :=‘ kanan’;
        grid.Cells[jVar+jPer+3,2]:=FloatTostr(a[jPer+jVar+1,0);
    for bar:=l to JPer do // mengisi ruas kanan
    grid. Cells [jVar+JPer+3, 2+bar):FloatToStr
    (a[jVar+jPer+1,bar]);
    for kol:O to jVar+jPer do // mengisi entry a[i,j]
    for bar:=O to jPer do
        grid.Cells[2+kol, 2+bar] :=FloatToStr(a[kol,bar]);
        grid.Visible :=true;
end;
```

## 5. PENUTUP

Penelitian pengembangan yang dilakukan menghasilkan suatu kode program komputasi menggunakan bahasa pemrograman borland delphi 6.0 untuk dapat digunakan menyelesaikan program linier dengan metode simpleks. Disarankan kepada peneliti selanjutnya agar melakukan penelitian sampai dengan terciptanya suatu software atau perangkat lunak baru yang dapat digunakan langsung untuk menyelesaikan persoalan program linier dengan metode simpleks.

## DAFTAR PUSTAKA

[1]. Alam, M.A.J. 2001. *Belajar Sendiri Borland Delphi 6.0*. PT. Elex Media Komputindo, Jakarta.
[2]. Hieller & Lieberman. 1995. *Operation Research*. McGraw Hill, New Jersey.
[3]. Lawrence, Stephen. 2008. *Survey of Operation Research*. Dictate of Engineering Management Program. University of Colorado.
[3]. Siagian, P. 1987. *Penelitian Operasional*. Universitas Indonesia, Jakarta.
[4]. Supranto, J. 1983. *Program Linier*. Lembaga Penerbit Fakultas Ekonomi UI, Jakarta.