DOĞRUSAL PROGRAMLAMA VE REVİSED SİMPLEKS ALGORİTMAYA DAYALI BİR BİLGİSAYAR PROGRAMI GELİŞTİRME UYGULAMASI

Sibkat KAÇTIOĞLU(*)
Pakize ERDOĞMUŞ(**)

Özet: Bu çalışmada yöneylem araştırmalarında yaygın kullanım alanına sahip olan Doğrusal Programlama çözüm tekniklerinden Revised Simpleks Algoritması kullanılarak, büyük boyutlu Doğrusal Programlama problemlerine daha hızlı çözümler üreten bir programın geliştirilmesi amaçlanmıştır. ATADP olarak adlandırdığımız program önce QBASIC'te geliştirilmiş, daha sonra Visual Basic'e aktarılmıştır.

Abstract: In this paper, it has been introduced an application programme which has been called ATADP. Our aim has been to developpe an application programme in order to solve large sized Linear Programming Problem quickly. ATADP has been developed by using Revised Simplex Method Algorithm, which is one of the common Lineer Programming solution techniques. Firstly ATADP has been developed with QBASIC, then it has been ported to Visual Basic 6.0 . ATADP has been compared with QSB,which is the common package programme. used for solving LP problems.

I. Giriş

Bir Doğrusal Programlama modeli doğrusal eşitlik veya eşitsizlik sınırlayıcı şartları altında doğrusal bir amaç fonksiyonunu optimize (maksimize veya minimize) etmeye yarar. Genel bir Doğrusal Programlama modeli üç unsurdan oluşur. Bunlar optimize edilecek olan amaç fonksiyonu, kısıtlayıcı ve pozitiflik şartlarıdır. Genel bir Doğrusal Programlama modeli;

(*) Prof.Dr. Atatürk Üniversitesi İİBF İşletme Bölümü Öğretim Üyesi
(**) Dr. Atatürk Üniversitesi Mühendislik Fak. Elektronik Haberleşme Müh.

Optimize edilecek Amaç Fonksiyonu $f. = c_1x_1 + c_2x_2 + ... + c_nx_n$

Kısıtlayıcı Şartlar:

$$\left.\begin{array}{l} a_{11}x_1 + a_{12}x_2 + ... a_{1n}x_n \le b_1 \\ ..... \\ a_{k1}x_1 + a_{k2}x_2 + ... a_{kn}x_n \le b_k \end{array}\right\} k \text{ adet küçük eşit kısıtlayıcı}$$

$$.....$$

$$\left.\begin{array}{l} a_{k+1,1}x_1 + a_{k+1,2}x_2 + ... a_{k+1,n}x_n = b_{k+1} \\ ..... \\ a_{l1}x_1 + a_{l2}x_2 + ... a_{ln}x_n = b_l \end{array}\right\} l - k \text{ adet eşit kısıtlayıcı}$$

$$.....$$

$$\left.\begin{array}{l} a_{l+1,1}x_1 + a_{l+1,2}x_2 + ... a_{l+1,n}x_n \ge b_{l+1} \\ ..... \\ a_{m1}x_1 + a_{m2}x_2 + ... a_{mn}x_n \ge b_m \end{array}\right\} m - l \text{ adet büyük eşit kısıtlayıcı} \qquad (1)$$

Pozitiflik Şartı:

$x_1, x_2, ... x_n \ge 0$

şeklinde tanımlanır.

II. Genel Bir Doğrusal Programlama Modelinin Çözümü

Doğrusal Programlama modelleri için temel çözüm yöntemi 1952 yılında Dantzig tarafından geliştirilen Simpleks Yöntem'dir. Simpleks Yöntem, yapay ve boş değişkenlerin eklenmesi ile başlangıç temel uygun çözümünden hareketle her adımda yeni bir temel uygun çözüme geçerek, belli sayıdaki iterasyonlar sonunda optimal çözüme ulaşır. Simpleks Yöntem'de her bir iterasyonda yeni bir simpleks tablo oluşturulur ve her bir iterasyonda amaç fonksiyonu optimum değerine biraz daha yaklaşır.

III. Revised Simpleks Yöntem

Bir Doğrusal Programlama problemini Simpleks Yöntem'le çözerken her iterasyonda (yani bir tablodan diğerine geçişte) tablonun gövdesi tümü ile işlem görür. Küçük boyutlu problemlerde bu husus fazla bir sakınca doğurmaz. Fakat Doğrusal Programlama problemleri genelde büyük boyutlu olduğundan

ve bilgisayarda çözüm gerektirdiğinden, bu yöntem oldukça büyük bellek gerektirir.

Bu sebeple bilgisayarların gelişmesine paralel olarak yeni bir yöntem arayışına gidilmiş ve 1953 yılında G.B. Dantzig ve Orchard-Hays tarafından Revised Simpleks Yöntemi (Düzeltilmiş Simpleks Yöntem) geliştirilmiştir. Bu yöntem aynı bilim adamları tarafından geliştirilen "Product Form of the Inverse" yönteminin geliştirilmiş şeklidir (Tohumcu, 1997:20). Revised Simpleks Yöntem genel olarak Simpleks Çözüm Yöntemini kullanır. Ancak Simpleks Yöntem gibi her iterasyonda yeni bir tablo oluşturmaz. Orjinal tablodan o iterasyonda işlem görecek bazı satır ve sütunları oluşturur. Bu suretle hem hafıza tasarrufu hem de zaman tasarrufu sağlar.

Simpleks Yöntem'de olduğu gibi modele gerekli yapay ve boş değişkenler eklenir. Amaç fonksiyonu f_c ve f_m olarak ikiye ayrılır. f_c, temele girecek değişkenlerin oluşturduğu, f_m ise yapay değişkenlerin oluşturduğu amaç fonksiyonudur. Modelin mümkün çözümünü elde edebilmek için önce f_m optimize edilmeye çalışılır. f_m satırında negatif katsayılı terim kalmadığında f_m optimize edilmiştir. f_m 'e ait satır ve sütunlar silinerek f_c amaç fonksiyonunun optimizasyonuna başlanır. f_c satırında da negatif katsayılı terim kalmayınca, modelin optimum çözümü elde edilmiş olur.

Temele girecek değişken sütunu, B^{-1} matrisi ile ile başlangıç tablosundaki temele girecek değişken sütununlarının çarpımından elde edilir. Aynı şekilde temel değişken değerleri, B^{-1} matrisi ile başlangıç tablosundaki temel değişken değerleri sütununun çarpımından elde edilir. Temel değişken değerlerinin temele girecek değişken değerlerine bölünmesinden elde edilen en küçük pozitif oran değerinden temelden ayrılacak değişken belirlenir.

Temele girecek değişken sütununda sadece anahtar sayı bir, diğer değerler sıfır olacak şekilde B^{-1} matrisi üzerinde işlemler yapılır (Kaçtıoğlu,1987:48). Bu şekilde bir değişken temele girmiş olur. Aynı işlemler bir sonraki iterasyon için tekrar edilir.

Simpleks Yöntem ile Doğrusal Programlama modellerinin çözümünde karşılaşılan dört özel durum mevcuttur (Taha, 2000:97).

A. Dejenerasyon (Yozlaşma) (Degenerecy)

Simpleks Yöntem'in uygunluk koşulları uygulanırken, minimum oran kuralında eşitlik olması halinde görülür. Yozlaşma iki şekilde görülür. En küçük pozitif oran değerini sağlayan sıfırdan büyük değerli birden fazla satırın olması ve pozitif oran değeri sıfır olan birden fazla satırın olması. Bu iki durum için de döngüden çıkmayı sağlayan yöntemler mevcut olduğu halde, pratikte az rastlandığı gerekçesi ile bu yöntemler birçok Doğrusal Programlama yazılımında, programa dahil edilmemişlerdir (Winston, W. L., 1994:162). ATADP'de her iki dejenerasyon durumu için de altprogram mevcuttur. Teorik olarak bakıldığında yozlaşma, problemi sonlu yada sonsuz bir döngüye sokar. Problemin çözümü gereksiz yere uzatılmış olur. Pratik olarak bakıldığında

yozlaşma, Doğrusal Programlama modellerinin gereksiz (redundant) bir kısıtlayıcı bulundurduğu durumda ortaya çıkar.

B. Alternatif Optimum Çözüm (Alternative Optima)

Amaç fonksiyonunun eğiminin, kısıtlayıcılardan herhangi birinin eğimi ile aynı olduğu durumlarda görülür. Optimizasyonu sağlayan birden çok çözüm kümesi mevcuttur.

C. Sınırsız Çözüm (Unbounded Solution)

Kısıtlayıcı şartların sınırlandırdığı bir çözüm uzayı olmadığında görülür.

D. Mümkün Çözümün Olmaması (No Feasable Solution)

Kısıtlayıcılar uygun bir konveks çözüm alanı oluşturmuyorsa modelin mümkün çözümü yoktur.

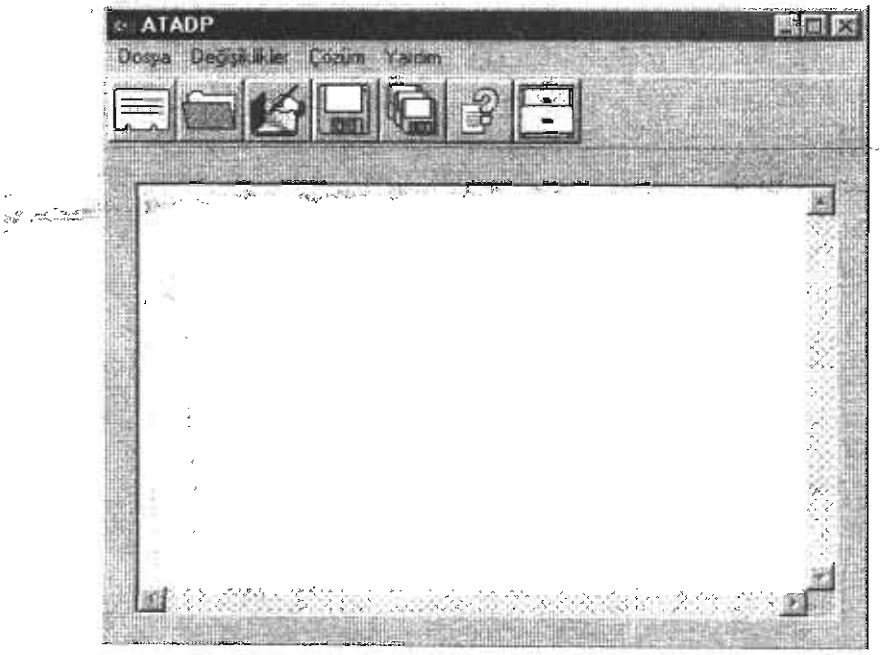
IV. ATADP Programı

ATADP programı Doğrusal Programlama Problemlerini Revised Simpleks Algoritma kullanarak cozen, tarafımızca geliştirilen bir programdır. Program, ilk olarak QBASIC ortamında geliştirilmiş (Şekil 1.), daha sonra grafik arayüzüne (GUI) ve olaya bağımlı bir yapıya sahip olan (Microsoft Press, 1998) Visual Basic'e aktarılmıştır. ATADP Visual Basic Projesi (ATADP.vbp), Atadp.frm, ondeg.frm, gridform.frm, forcoz.frm, frmAbout.frm, frmSplash.frm, frmtip.frm bağımsız formlarından ve degisken.bas modülünden oluşmaktadır.

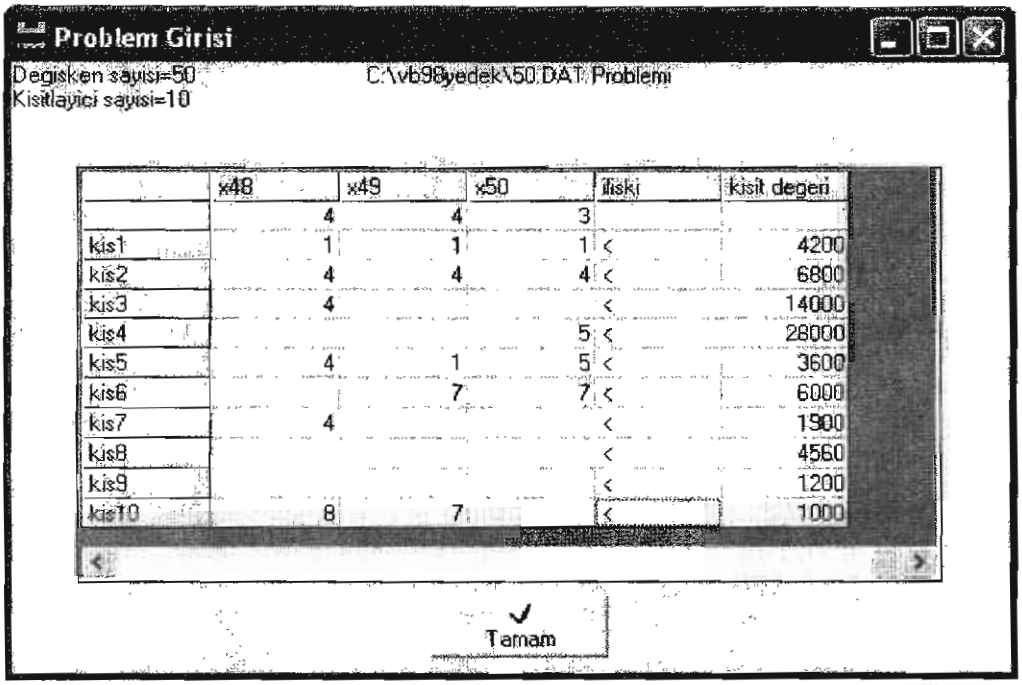
Şekil 1. *ATADP 'nin QBASIC 'deki Açılış Penceresi*

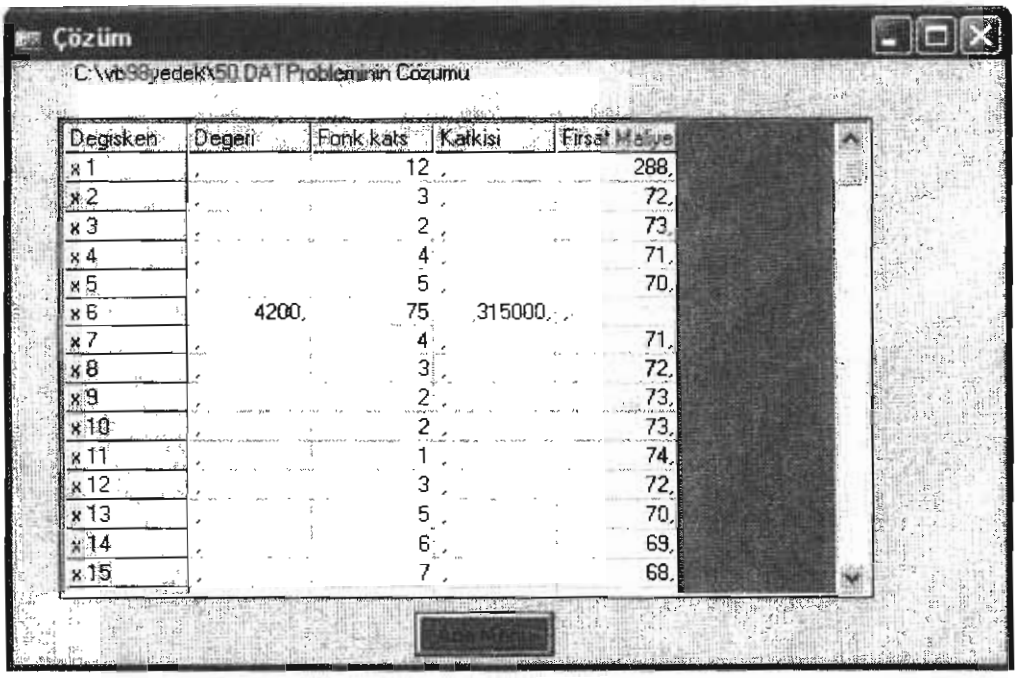
A. Atadp Formu: Bu form aynı zamanda programın açılış ekranıdır. (Şekil 2.) Bu form üzerinde menu çubuğu, ve araç çubuğu bulunur. Programın diğer bütün formları ile ilişkilidir.

B. Ondeg Formu: Yeni problem girişi yapılan formdur. Girilecek problemin amaçı, değişken sayısı ve kısıtlayıcı sayısı bu form üzerinden girilir. Formdeg ve Atadp formu ile bağlantılıdır.
C. Gridform Formu: Yeni problem girişinde, amaç fonksiyonu ve kısıtlayıcı katsayılarının girildiği formdur. Problemin görüntülenmesi, amaç fonksiyonu ve kısıtlayıcı katsayıları üzerinde yapılacak değişiklikler de bu form üzerinden yapılır.
D. Formcoz Formu: Problemin çözümünün yapıldığı formdur.
E. FrmAbout Formu: Program hakkında özet bilgiler bu form üzerinden verilir.
F. FrmSplash Formu: Geliştirilen program, bilgisayara yükleninceye kadar hemen programa girildiği izlenimi vermek, programın ön kapağını ekrana getirmek için kullanılan bir formdur.
G. FrmTip Formu: Programla ilgili ipucu bilgilerini kullanıcıya gösterir.
H. Degisken.bas Modülü: Global değişken ve prosedürlerin tanımlarının yapıldığı ve sub main prosedürünün yer aldığı moduldür.



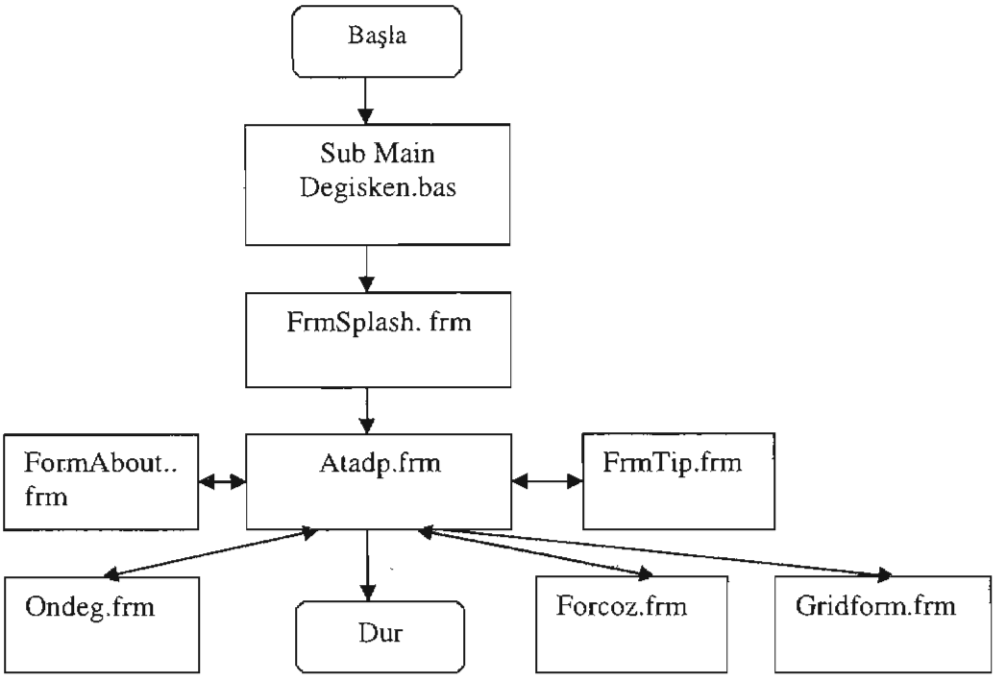
Şekil 2. *ATADP 'nin VISUAL BASIC 'deki Açılış Penceresi*





Şekil 3. *Problem Girişi Formu, Şekil 4. Problem Çözüm Formu*

Aşağıda ATADP projesinin akış şeması verilmiştir.



Şekil 5. *ATADP Projesi Akış Şeması*

ATADP programı geliştirilirken Revised Simpleks Yöntem Algoritması kullanarak, çok büyük boyutlu Doğrusal Programlama problemlerine hızlı çözümler üreten bir program amaçlanmıştır. Bu amaçla ATADP benzer işlevli bir program olan QSB ile kıyaslanmıştır. Tablo 1'de farklı boyutlardaki modellerin ATADP 1.0 ve QSB 2.0'daki çözüm süreleri verilmiştir. ATADP benzer işlevli ve yaygın olarak kullanılan QSB programı ile karşılaştırıldığında aynı problemi ATADP'nin daha kısa sürede çözdüğü görülmüştür. ATADP Revised Simpleks Algoritma, QSB ise Simpleks algoritma kullanmaktadır. Dejenerasyon durumları için de alt yordamlar bulunduran ATADP QSB'den daha az iterasyonla optimal sonuca gitmektedir. Bir Doğrusal Programlama modelinin çözüm süresindeki artışın yaklaşık olarak kısıtlayıcı sayısındaki artışın kübü ile doğru orantılı (Wagner,1974 :118) olduğu göz önüne alınırsa ATADP'nin kısıtlayıcı sayısı arttıkça QSB'ye göre daha da hızlı çözümler üreteceği görülecektir (Tablo 1.).

Tablo1. *ATADP 1.0 ile QSB'nin Çözüm Süreleri*

Problem no:	Boyut: (Değişken Sayısı X Kısıtlayıcı Sayısı)	Çözüm Süresi(sn)		ATADP/ QSB
		QSB	ATADP	
1	15x10	3	2	0.67
2	90x13	13	10	0.77
3	20x30	18	14	0.77
4	30x40	24	17	0.70
5	40x35	25	19	0.76
6	35x40	31	20	0.65
7	100x30	34	25	0.73
8	40x51	56	45	0.80
9	80x70	180	135	0.75

V. Sonuç

Çalışmada Revised Simpleks algoritmasını kullanarak büyük boyutlu Doğrusal Programlama Modellerinin hızlı çözümünü amaçlayan ve tarafımızca geliştirilen ATADP adlı program tanıtılmış ve yaygın olarak kullanılan benzer amaçlı QSB programı ile kıyaslanmıştır. Geliştirilen programda Revised Simpleks algoritmanın tercih edilme sebebi, daha az işlem ve daha az bellek kapasitesi gerektirmesidir. Geliştirilen program (ATADP 1.0) programı kişisel bilgisayarlarda kullanılan ve DOS altında çalışan QSB 2.0 ile, Visual Basic'e aktarılmış olan ATADP ise WinQSB ile kıyaslanmış ve şu sonuçlar elde edilmiştir.

1. ATADP QSB'den yaklaşık %30 daha hızlıdır. Program teorik olarak beklenen vermiştir. Kısıtlayıcı sayısı arttıkça ATADP programının %30'dan daha hızlı olacağı söylenebilir. Küçük boyutlu modellerde Revised Simpleks Yöntemi işlem sayısı Simpleks Yöntem işlem sayısına yakın olmasına rağmen boyut büyüdükçe (kısıtlayıcı sayısı ve değişken sayısı arttıkça) Revised Simpleks Yöntem daha iyi sonuçlar verecektir.
2. QSB'de dejenerasyon durumları göz önüne alınmamıştır. Dejenerasyon oluşacak bir problem girildiğinde ise iterasyon sayısı artmaktadır. ATADP'de ise dejenerasyonu önleyen iki alt yordam bulunmaktadır. Böylece dejenere bir problemin sebep olacağı fazla iterasyonlar önlenmiştir.

Kaynaklar

Kaçıoğu, S. (1987), Doğrusal Programlama ve Ulaştırma Modeli. Atatürk Üniversitesi, İktisadi ve İdari Bilimler Araştırma Merkezi, Erzurum.

Microsoft . (1998), Microsoft Visual Basic 6.0 Programmer's Guide, Microsoft Press, Washington, USA.

Taha, A. H. (2000), Yöneylem Araştırması, 6. Basımdan Çeviri, Literatür Kitabevi, İstanbul. (Çeviren: Baray, Ş., Alp., Esnaf, Ş.)

Tohumcu, P. (1997), Doğrusal Programlama ve Revised Simpleks Yönteme Dayalı Bir Bilgisayar Programı Geliştirme Uygulaması, Fen Bilimleri Enstitüsü, Bilgisayar Bilimleri Anabilim Dalı, Erzurum.

Wagner, H. M. (1974), Principles of Operations Research, Printice-Hall of Indiana Private Limited, Indiana.

Winston,W. L. (1994), Operations Research, Third Edition, International Tomson Publishing (ITP), Belmont, California.

Ekler 1.

ATADP'de problem çözümünü yapan program kesimi. (Forcoz Formu)

```
' ********************  A T A D P  ************************
'Problem çözümünde kullanilan degiskenlerin resetlenmesi
For i = 1 To cons + 1
   tgd(i) = 0: tdd(i) = 0
Next i
t = 0: cu = 0: va = 0: it = 0: k = 0: l = 0: sv = 0: ata = 0
For i = 1 To 100
          bb$(i) = "": aa$(i) = "": oran(i) = 0: c(i) = 0: ek(i) = 0: esay(i) = 0
Next i
For i = 0 To 100
          For j = 0 To 100
                    mat(i, j) = 0: b(i, j) = 0
          Next j:  Next i
'Yapay ve boş değişkenlerin eklenmesi
For i = 1 To cons
   If a(i, var + 1) = 0 Then
      t = t + 1:  bb$(i + 1) = "y" + Str$(t)
   Else
      cu = cu + 1:  bb$(i + 1) = "z" + Str$(cu)
   End If
   If a(i, var + 1) = 2 Then
      va = va + 1:  aa$(var + va) = "t" + Str$(va)
    End If
Next i
For i = 1 To var
   aa$(i) = "x" + Str$(i)
Next i
'Temele girecek degiskenler aa$(i),temelden ayrilacaklar bb$(i) dizilerinde saklanmaktadir.
' Temele girecek olan degiskenler, modelin degiskenleri ile  buyuk esit kisitlayicilardan cikarilan
' ti bos degiskenleridir. Temelden ayrilacak degiskenler ise yi bos degiskenleri ile zi yapay
' degiskenleridir.
cu = 0: t = 0: b(0, 0) = 0: b(1, 0) = 0
' KURULUS TABLOSU OLUSTURMA
'Simpleks tablo B matrisinde olusturulur. B matrisinin ilk sutununda(0. sutun) kisitlayicilarin kisit
degerleri yani temel degisken degerleri saklanir. B matrisinin ilk satirinda (0. satir) fc fonksiyonu,
ikinci satirinda fm fonksiyonu saklanmaktadir. Ucuncu satirdan itibaren ise kisitlayicilar
yerlestirilmistir.
For i = 2 To cons + 1
   b(i, 0) = a(i - 1, var + 2)
Next i
For i = 0 To var + va
   b(1, i) = 0
Next i
For i = 1 To var
   b(0, i) = -a(0, i)
Next i
For i = 2 To cons + 1
   For j = 1 To var
      b(i, j) = a(i - 1, j)
   Next j
```

```
    If a(i - 1, var + 1) = 2 Then
      sv = sv + 1: b(i, var + sv) = -1
    End If
  Next i
' KURULUS TABLOSUNDAN BASLANGIC TABLOSUNA GECIS
' ">" veya "=" kisitlayicilarda Mat inverse matrisini kanonik hale
'. getirmek icin fm satirindaki fm sutunu disindaki diger katsayilari
' sifirlamak iciu islemler yapilarak Mat matrisi kanonik hale getirilir.
  For i = 2 To cons + 1
    If a(i - 1, var + 1) <> 0 Then
      For j = 0 To var + va
        b(1, j) = b(1, j) - b(i, j)
      Next j
    End If
  Next i
  For i = 0 To cons + 1
    For j = 0 To cons + 1
      If i = j Then mat(i, j) = 1 Else mat(i, j) = 0
    Next j: Next i
'************** COZUM ************
5500
' BIRINCI EVRE
  For i = 1 To var + va
    km(i) = 0
  Next i
' fm katsayilarinin hesabi . Km(i) dizisi fm amac fonksiyonu katsayilarini icerir.
  For i = 1 To var + va
    For j = 0 To cons + 1
      km(i) = mat(1, j) * b(j, i) + km(i)
    Next j: Next i
' Temele girecek degiskenin(anahtar sutunun belirlenmesi)
' En buyuk negatif katsayili degisken temele giren degisken olarak secilir.
  enk = 999999
  For i = 1 To var + va
    If km(i) < -0.00001 Then
      If km(i) < enk Theu enk = km(i): c(it) = i
    End If
  Next i
  For i = 0 To cons + 1
    tdd(i) = 0: tgd(i) = 0
  Next i
  For i = 0 To cons + 1
    For j = 0 To cons + 1
      tgd(i) = tgd(i) + mat(i, j) * b(j, c(it))
      tdd(i) = tdd(i) + mat(i, j) * b(j, 0)
    Next j: Next i
' Temelden ayrilacak degiskenin(anahtar satirin belirlenmesi) En kucuk
' pozitif katsayili degisken temelden ayrilacak degisken olarak secilir.
  If enk <> 999999 Then
    For i = 2 To cons + 1
      If tgd(i) > 0 Then
        oran(i) = tdd(i) / tgd(i)
```

```
        If oran(i) < 0 Then oran(i) = -1
      Else
      oran(i) = -1
      End If
    Next i
' Burada dejenerasyona bakilir(Birden fazla en kucuk oran degeri)
    enk = 999999
    For i = 2 To cons + 1
      If oran(i) <> -1 Then
        If oran(i) < enk Then enk = oran(i): ek(it) = i
      End If
    Next i
' Oran degerlerinin hepsi 0 dan kucukse temelden ayrilacak degisken yoktur.
' Yani sinirsiz cozum vardir.
    If enk = 999999 Then
      MsgBox ("SINIRSIZ ÇÖZÜM(UNBOUNDED SOLUTION)")
      atdp.Show
    End If
    t = 1
' Dejenerasyon arastirmasi
    For i = 2 To cons + 1
      If i <> ek(it) Then
        If oran(i) = enk Then t = t + 1: esay(t) = i
      End If
    Next i
    If t <> 1 Then
      If enk = 0 Then GoSub 5800 Else GoSub 5700
    End If
' aa$(c(it))  temele giriyor.
' bb$(ek(it)) temelden cikiyor
' Temele girecek degisken sutunu kanonik hale getirilir.
    bb$(ek(it)) = aa$(c(it))
    For i = 0 To cons + 1
      mat(ek(it), i) = mat(ek(it), i) / tgd(ek(it))
    Next i
    For i = 0 To cons + 1
      If i <> ek(it) Then
        For j = 0 To cons + 1
          mat(i, j) = mat(i, j) - mat(ek(it), j) * tgd(i)
        Next j
      End If
    Next i
    it = it + 1
    GoTo 5500
' Fm optimize edildikten sonra temelde 0 dan farkli bir katsayi ile yapay degisken varsa modelin '
mumkun cozumu yoktur. Eger bir yapay degisken mevcut fakat katsayisi 0 ise model optimize
' edilmistir. Artik ikinci evreye gecilmez.
  Else
    ata = 0
      For i = 2 To cons + 1
        If Left$(bb$(i), 1) = "z" Then
          If tdd(i) <> 0 Then
```

```
            MsgBox ("MUMKUN COZUM YOK(NO FEASABLE SOLUTION)")
              atdp.Show
                    Else
              ata = 2
            End If
          End If
        Next i
      If ata = 2 Then
        GoTo son
      End If
  End If
' IKINCI EVRE
' fc amac fonksiyonunun optimize edilmesi
  For i = 2 To cons + 1
    bb$(i - 1) = bb$(i)
  Next i
  k = 0: l = 0
  For i = 0 To cons + 1
    If i = 1 Then i = i + 1
' fm'e ait satir ve sutunlar silinir.
    For j = 0 To cons + 1
      If j = 1 Then j = j + 1
      mat(k, l) = mat(i, j)
      l = l + 1
    Next j
    k = k + 1: l = 0
  Next i
  k = 0
' Yeni B matrisi
  For i = 0 To cons + 1
    If i = 1 Then i = i + 1
    For j = 0 To var + va
      b(k, j) = b(i, j)
    Next j
    k = k + 1
  Next i
5600
  For i = 1 To var + va
    km(i) = 0
  Next i
  For i = 0 To cons
    tdd(i) = 0: tgd(i) = 0: oran(i) = 0
  Next i
  For i = 1 To var + va
    For j = 0 To cons
      km(i) = mat(0, j) * b(j, i) + km(i)
    Next j
   Next i
  enk = 999999
  For i = 1 To var + va
    If km(i) < -0.000001 Then
      If km(i) < enk Then enk = km(i): c(it) = i
```

```
    End If
  Next i
  If enk <> 999999 Then
    For i = 0 To cons
      For j = 0 To cons
        tgd(i) = tgd(i) + mat(i, j) * b(j, c(it))
        tdd(i) = tdd(i) + mat(i, j) * b(j, 0)
      Next j
      If tgd(i) > 0 Then
        oran(i) = tdd(i) / tgd(i)
        If oran(i) < 0 Then oran(i) = -1
      Else
        oran(i) = -1
      End If
    Next i
' PRINT aa$(c(it)) temele giriyor.
' Burada dejenerasyona bakilir.
    enk = 999999

            For i = 1 To cons
      If oran(i) <> -1 Then
        If oran(i) < enk Then enk = oran(i): ek(it) = i
      End If
    Next i
    If enk = 999999 Then
      MsgBox ("SINIRSIZ ÇÖZÜM(UNBOUNDED SOLUTION)")
      atdp.Show
    End If
    t = 1
' Dejenerasyon arastirmasi
    For i = 1 To cons
      If i <> ek(it) Then
        If oran(i) = enk Then t = t + 1: esay(t) = i
      End If
    Next i
    If t <> 1 Then
      If enk = 0 Then GoSub 5800 Else GoSub 5700
      End If
' Temele girecek degisken sutununu kanonik hale getirmek
    bb$(ek(it)) = aa$(c(it))
    For i = 0 To cons
      mat(ek(it), i) = mat(ek(it), i) / tgd(ek(it))
    Next i
    For i = 0 To cons
      If i <> ek(it) Then
        For j = 0 To cons
          mat(i, j) = mat(i, j) - mat(ek(it), j) * tgd(i)
        Next j
      End If
    Next i
    it = it + 1
    GoTo 5600
```

```
  End If
  For i = 0 To cons
     For j = 0 To cons
        tdd(i) = tdd(i) + mat(i, j) * b(j, 0)
     Next j:  Next i
  GoTo son
5700
' Anahtar satir seciminde 0 dan farkli en kucuk oran degeri birden fazla  olursa dejenerasyon ile
' karsilasilir.
   esay(1) = ek(it)
   For j = 1 To cons + 1
      enk = 34986
      For i = 1 To t
         oran(i) = mat(esay(i), j) / tgd(esay(i))
         If oran(i) < enk Then enk = oran(i): ek(it) = esay(i)
      Next i
      For i = 1 To t
         If i <> ek(it) Then
            If oran(i) = enk Then esp = 1
         End If
      Next i
      If esp = 0 Then j = cons
   Next j:  Return
5800
' Perturbation metod
car = 1: esay(1) = ek(it):  enk = 67667
   For i = 1 To t
      car = car * 0.1175:  tdd(esay(i)) = car:  oran(i) = tdd(esay(i)) / tgd(esay(i))
      If oran(i) < enk Then
      enk = oran(i):  ek(it) = esay(i)
      End If
   Next i:  Return:
son:
```