



**T.C.
RECEP TAYYİP ERDOĞAN ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ
MATEMATİK ANABİLİM DALI**

**MAXİMA İLE LİNEER PROGRAMLAMA VE
UYGULAMALARI**

(Yüksek Lisans Tezi)

Fatma ERDEM TOSUN

**Danışman
Dr. Öğr. Üyesi Süleyman ŞENGÜL**

**RİZE
2024**

KABUL VE ONAY

Recep Tayyip Erdoğan Üniversitesi, Lisansüstü Eğitim Enstitüsü, Matematik Anabilim Dalında, Dr. Öğr. Üyesi Süleyman ŞENGÜL danışmanlığında Fatma ERDEM TOSUN tarafından hazırlanan *Maxima ile Lineer Programlama ve Uygulamaları* adlı bu tez çalışması, 22/07/2024 tarihinde yapılan savunma sınavı sonucunda oy birliğiyle/oy çokluğuyla başarılı bulunarak jürimiz tarafından **Yüksek Lisans Tezi** olarak kabul edilmiştir.

Jüri Üyeleri	Ünvanı, Adı SOYADI	İmza
Başkan	: Dr. Öğr. Üyesi İshak CUMHUR	
Üye	: Dr. Öğr. Üyesi Mehmet ÜNLÜ	
Üye	: Dr. Öğr. Üyesi Süleyman ŞENGÜL	

ETİK BEYAN

Matematik Tezli Yüksek Lisans Programından mezun olmak üzere teslim ettiğim “Maxima ile Lineer Programlama ve Uygulamaları” adlı tezim, bilim ve araştırma etiği prensiplerine riayet edilerek tarafımdan yazılmıştır.

Tez çalışmamda, başka kaynaklardan aktarılan bütün bilgi ve alıntılar, Enstitünüz Tez Yazım Kılavuzuna uygun olarak açıkça gösterilmiştir. Kaynağı gösterilenler dışında kalan bütün bilgiler uygun araştırma yöntemi kullanılarak tarafımdan edinilmiş ve esere bu şekilde yansıtılmıştır. Şahsıma ait olmayan hiçbir bilgi, kasıt veya kusurlar, şahsıma aitmiş gibi gösterilmemiştir. İnternet kaynakları dâhil, sahibine/kaynağına atıf yapılmaksızın hiçbir bilgi kullanılmamıştır. Aksinin ortaya çıkması halinde doğacak bütün hukuki, idari, akademik ve etik sorumluluk tarafıma ait olacaktır. Eserin tesliminden sonra herhangi bir zamanda, bilim etiğine aykırılık tespit edilmesi ve / veya eserimle ilgili intihal veya intihal şeklinde anlaşılabilecek bir durumun ortaya çıkması halinde; Üniversiteniz ve eğitim kadronuzun hiçbir şekilde sorumlu tutulmayacağını hür irademle kabul, beyan ve taahhüt ederim.

22/07/2024

Fatma ERDEM TOSUN

ÖN SÖZ

Bu tez çalışmasında doğrusal programlama problemlerin çözüm yöntemleri örneklerle açıklanmış daha sonra bu problemlerin çözümünü yapmak için Maxima programından yararlanılarak kullanıcıların problem çözmenin aşamalarını da görebileceği çözümler üretilmiştir.

Tez çalışmanın başından sonuna kadar bilgi ve desteğini hiçbir zaman eksik etmeyen tez danışmanım değerli hocam Dr. Öğr. Üyesi Süleyman ŞENGÜL'e emeklerinden ötürü sonsuz teşekkürlerimi sunarım

Ayrıca bu süreçte ve hayatımın her döneminde bana desteklerini esirgemeyen sevgili anne babama, değerli eşim İsa TOSUN'a çocuklarım Ertuğrul ve Alperen'e teşekkür ederim.

Fatma ERDEM TOSUN

RİZE/2024

İÇİNDEKİLER

KABUL VE ONAY	I
ETİK BEYAN	II
ÖN SÖZ	III
İÇİNDEKİLER	IV
ÖZET	VI
ABSTRACT.....	VII
TABLolar LİSTESİ	VIII
ŞEKİLLER LİSTESİ	X
GİRİŞ	1
1. LİNEER OPTİMİZASYON YÖNTEMLERİ	7
1.1. Grafik Yöntemi	7
1.2. Simpleks Yöntem	10
1.2.1. Standart Form	11
1.2.2. Temel Çözümün Belirlenmesi	12
1.2.3. Simpleks Algoritması	14
1.3. M yöntemi	27
1.4. İki Aşamalı Simpleks Yöntemi	32
1.5. Dual Problem	35
1.5.1. Dual Problemin Kurulması.....	35
1.5.2. Dual Simpleks Yöntemi	38
1.6. Primal Problemin Matrisel Gösterimi.....	42
1.6.1. Dual Problemin Matris Tanımı	50
1.7. Optimum Sonrası Analiz veya Duyarlılık Analizi	54
1.7.1. Uygunluğu Etkileyen Değişiklikler.....	56
1.7.1.1. b vektörünün değişmesi durumu:	56
1.7.1.2. b 'nin elemanlarının uygunluk aralığı:	58
1.7.1.3. Yeni kısıtın eklenmesi.....	59
1.7.2. Optimumluğu Etkileyen Değişiklikler	62
1.7.2.1. c_j amaç katsayılarındaki değişiklikler	62
1.7.2.2. Yeni bir faaliyetin eklenmesi	64

1.7.2.3. Optimumluğun korunduğu değişim aralığının belirlenmesi.....	66
2. YAPILAN ÇALIŞMALAR	69
2.1. Maxima ortamında grafik yönteminin kodlanması	81
2.2. Simpleks yönteminin maxima’da programlanması.....	87
2.3. <i>M</i> Yönteminin Maxima Ortamında Programlanması.....	92
2.4. İki aşamalı simpleks yönteminin maxima’da programlanması.....	101
2.5. Dual simpleks yönteminin maxima’da programlanması.....	113
2.6. Matrisel Formun Maxima Ortamında Kodlanması	122
2.7. Maxima’da Duyarlılık Analizi	125
3. SONUÇLAR.....	127
4. ÖNERİLER.....	128
KAYNAKÇA.....	129
EKLER.....	130

Recep Tayyip Erdoğan Üniversitesi Lisansüstü Eğitim Enstitüsü

Anabilim Dalı : Matematik Anabilim Dalı

Tez Türü : Yüksek Lisans Tezi

Danışman : Dr. Öğr. Üyesi Süleyman ŞENGÜL

Hazırlayan : Fatma ERDEM TOSUN

Yıl : 2024

Sayfa Sayısı : 164

ÖZET

MAXİMA İLE LİNEER PROGRAMLAMA VE UYGULAMALARI

İktisadi ve ekonomi gibi alanlarda en çok karşılaşılan problemlerden biri doğrusal (lineer) programlama problemleridir. Bu problemlerin çözümü için tarih boyunca değişik yöntemler kullanılmıştır. Bu yöntemlerde en uygun çözüme bir uygun köşe noktasından diğer uygun bir köşe noktasına geçilerek adım adım ulaşılmaktadır. Teknolojik gelişmelerin yaşanmasıyla birlikte lineer programlama problemleri için farklı yazılımlar geliştirilmiştir. Bu yazılımların bir kısmı açık erişimlidir ancak çoğunluğu ücretli yazılımlardan oluşmaktadır. Bu ise herkesin bu yazılımlara erişimini zorlaştırmaktadır. Öte yandan bu yazılımlar lineer optimizasyon problemlerini çözerken genel olarak giriş elemanlarını alarak kullanıcıya sadece en uygun (optimum) sonucu göstermektedir.

Bu çalışmada ücretsiz olarak erişilebilen Maxima programı yardımıyla lineer optimizasyon problemleri için açık erişimli kodlar hazırlanması amaçlanmıştır. Çalışmanın birinci kısmında lineer programlamada ele alınan yöntemler hakkında örnekler üzerinden bilgiler verilmektedir. Seçilen örnekler genel olarak lisans öğrencilerinin anlayabileceği endüstriyel problemlerden seçilmeye çalışılmıştır. İkinci kısmında ise her bir yöntemin Maxima ortamına aktarılması sağlanarak birinci kısımda ele alınan problemlerin çözümleri ilgili program ile elde edilmektedir. Hazırlanan programda diğer programlardan farklı olarak her bir adımda elde edilen veriler ve uygun çözümler ekranda yazdırılarak kullanıcı dostu bir program oluşturulması hedeflenmiştir. Ayrıca oluşturulan Maxima kodları da ek bölümünde verilerek kullanıcılara açık erişimli bir program oluşturulması sağlanmaktadır.

Anahtar Kelimeler: Optimizasyon, Grafik Yöntemi, Simpleks Yöntemi, Dualite, Duyarlılık Analizi

Recep Tayyip Erdogan University Institute of Graduate Studies

Department : Mathematics

Thesis Type : Master's Thesis

Supervisor : Assist. Prof. Dr. Süleyman ŞENGÜL

Author : Fatma ERDEM TOSUN

Year : 2024

Pages : 164

ABSTRACT

LINEAR PROGRAMMING WITH MAXİMA AND ITS APPLICATIONS

One of the most common problems encountered in fields such as economics and business is linear programming problems. Throughout history, various methods have been used to solve these problems. Typically, the optimal solution is reached step by step, moving from one suitable corner point to another. With technological advancements, different software has been developed for solving linear programming problems. While some of this software are open-access, the majority are commercial, making access difficult for everyone. Moreover, this software generally provide the user with only the optimal result based on input elements.

In this study, the aim is to prepare open-access codes for linear optimization problems using the freely available Maxima program. The first part of the study provides information on methods used in linear programming through examples. These examples are generally selected from industrial problems that undergraduate students can understand. In the second part, each method is transferred to the Maxima environment to obtain solutions for the problems discussed in the first part. Unlike other programs, the developed program aims to be user-friendly by displaying data obtained at each step and suitable solutions on the screen. Additionally, the Maxima codes created are provided in an appendix section, ensuring the creation of an open-access program for users.

Keywords: Optimization, Graphical Method, Simplex Method, Duality, Sensitivity Analysis

TABLÖLAR LİSTESİ

Tablo 1. Lastik fabrikasındaki üretim	4
Tablo 2. Özel yemdeki bileşenler	9
Tablo 3. Cam firması için veriler	15
Tablo 4. Başlangıç Simpleks tablosu.....	17
Tablo 5. Pivot sütunu.....	18
Tablo 6. Pivot sütunundaki oranlar	18
Tablo 7. Pivot satırı	19
Tablo 8. Pivot elemanın değerinin 1 yapılması	19
Tablo 9. Gauss Jordan yöntemiyle eliminasyon.....	20
Tablo 10. Pivot sütunundaki oranlar	20
Tablo 11. Pivot elemanın değerinin 1 yapılması	21
Tablo 12. Gauss Jordan yöntemiyle eliminasyon.....	21
Tablo 14. Pivot satırı belirlemek için oranlar	24
Tablo 15. Pivot elemanın değerinin 1 yapılması	24
Tablo 16. 1. Simpleks aşaması sonunda elde edilen tablo	24
Tablo 17. Pivot eleman için oranlar	25
Tablo 18. Pivot elemanın değerinin 1 yapılması	25
Tablo 19. 2. Simpleks aşaması sonunda elde edilen tablo	26
Tablo 20. Başlangıç simpleks tablosu	28
Tablo 21. Düzenlenmiş başlangıç Simpleks tablosu	28
Tablo 22. Pivot satır ve sütun	29
Tablo 23. Pivot sütun.....	29
Tablo 24. Başlangıç Simpleks tablosuna Gauss Jordan yöntemiyle eliminasyon.....	29
Tablo 25. Pivot elemanın bulunması	30
Tablo 27. Yardımcı problem için başlangıç Simpleks tablosu	33
Tablo 28. Yardımcı problem için düzenlenmiş başlangıç Simpleks tablosu	33
Tablo 29. Yardımcı problem için Gauss Jordan yöntemiyle eliminasyon.....	33
Tablo 30. Esas problem için başlangıç Simpleks tablosu.....	34
Tablo 31. Esas problem için düzenlenmiş başlangıç Simpleks tablosu	34
Tablo 32. Primal problem ile dual problem arasındaki ilişki-1 (Taha, 2010)	36

Tablo 33. Primal problem ile dual problem arasındaki ilişki-2 (Taha, 2010)	36
Tablo 34. Diyetisyen problemi	37
Tablo 35. Dual Simpleks yöntemi başlangıç tablosu	40
Tablo 36. Dual Simpleks yöntemi başlangıç tablosundaki oranlar	40
Tablo 37. Dual Simpleks yöntemi birinci yineleme sonuçları	40
Tablo 38. Dual Simpleks yöntemi birinci yineleme sonucundaki oranlar	41
Tablo 39. Dual Simpleks yöntemi ikinci yineleme sonuçları.....	41
Tablo 40. Simpleks tablosunun matris gösterimi	45
Tablo 41. Başlangıç Simpleks tablosu.....	47
Tablo 42. Birinci yineleme sonucunda elde edilen Simpleks tablosu	49
Tablo 43. İkinci yineleme sonucunda elde edilen Simpleks tablosu.....	50
Tablo 44. Örnek 15 için dual ve primal Simpleks tabloları:	52
Tablo 45.1. yineleme sonucundaki tablo	52
Tablo 46.2. yineleme sonucunda oluşan tablo:	52
Tablo 47. Örnek 9 için optimum çözümün bulunduğu Simpleks tablosu	56
Tablo 48. Örnek 14.1 için optimum çözümün bulunduğu Simpleks tablosu	57
Tablo 49. Örnek 14.1 için Dual Simpleks algoritması sonucu.....	57
Tablo 51. Yeni kısıt eklenmesi sonucu temel değişkenler için yapılan işlem tablosu	61
Tablo 52. Birinci dual Simpleks algoritması uygulandığında elde edilen tablo.....	61
Tablo 53. İkinci dual Simpleks algoritması uygulandığında elde edilen tablo	61
Tablo 54. Örnek 15.1 için simpleks tablosu	63
Tablo 55. Örnek 15.1 için birinci yineleme ile elde edilen simpleks tablosu	64
Tablo 56. Örnek 15.2 için Simpleks tablosu	65
Tablo 57. Örnek 15.2 için Gauss-Jordan algoritması uygulanması.....	66
Tablo 58. Örnek 16 başlangıç simpleks tablosu	75
Tablo 59. Alaşımdaki metal oranları	85

ŞEKİLLER LİSTESİ

Şekil 1. Örnek 1'deki kısıtlamalar için uygun bölge	8
Şekil 2. Örnek 4'deki problem için uygun bölge.....	10
Şekil 3. Örnek 8 için uygun çözüm bölgesi.....	16
Şekil 4. Örnek 10'daki problem için uygun çözüm bölgesi	31
Şekil 5. Örnek 12 için uygun çözüm bölgesi.....	42
Şekil 7. Programın elde ettiği başlangıç simpleks tablosu	77
Şekil 8. Temel değişkenler ve temel değişkenlerin değerleri	77
Şekil 9.1. aşamada elde edilen simpleks tablosu	77
Şekil 10.1. aşama sonucunda elde edilen sonuç	78
Şekil 11.2. aşamada programın elde ettiği sonuçları gösteren bölüm	78
Şekil 12. Problemin temel değişkenlerinin değerleri ile optimum çözümü	79
Şekil 13. Örnek 1'deki kısıtlamalar için uygun bölge	83
Şekil 14. Maxima ortamında belirlenmiş olan köşe noktaları	84
Şekil 15. Maxima ortamında belirlenmiş maksimum nokta ve maksimum değer	84
Şekil 16. Örnek 18 kısıtlamaları için uygun bölge	86
Şekil 17. Maksima ortamında belirlenmiş olan köşe noktaları	86
Şekil 18. Maksimum noktalar.....	87
Şekil 19. Örnek 19 için başlangıç simpleks tablosu	89
Şekil 20.1. aşama sonunda programın ekrana yazdırdığı simpleks tablosu	90
Şekil 21.2. aşama sonunda programın ekrana yazdırdığı simpleks tablosu	91
Şekil 22. Örnek 19 için programın elde ettiği sonuçlar.....	91
Şekil 23. Örnek 20 için başlangıç simpleks tablosu	94
Şekil 24. Temel değişkenler ve temel değişkenlerin değerleri	94
Şekil 25. Örnek 20 için uyumsuzluğun giderildiği Simpleks Tablosu	95
Şekil 26. Temel değişkenleri yerine yazıldığı başlangıç simpleks tablosu.....	95
Şekil 27. Birinci aşama sonunda oluşan simpleks algortiması.....	96
Şekil 28. Simpleks algoritması sonucunda oluşan 2. aşaması.....	96
Şekil 29. Temel değişkenler, temel değişken değerleri ile optimum çözüm	97
Şekil 30. Başlangıç simpleks tablosu	98
Şekil 31. Temel değişkenler ve temel değişkenlerin değerleri	98

Şekil 32. Uyumsuzluğun giderildiği simpleks Tablosu	99
Şekil 33. Simpleks algoritmasının 1. aşaması	99
Şekil 34. Simpleks algoritmasının 2. aşaması	100
Şekil 35. Simpleks algoritmasının 3. aşaması	100
Şekil 36. Temel değişkenler, değerleri ve optimum çözüm.....	101
Şekil 37. Örnek 21 için birinci aşama problemin başlangıç simpleks tablosu	104
Şekil 38. Birinci aşamada uyumsuzluğun giderildiği başlangıç simpleks tablosu ..	105
Şekil 39. Temel değişken değerleri ve başlangıç simpleks tablosu	106
Şekil 40. Yardımcı probleme simpleks algoritması uygulandığında oluşan tablo....	106
Şekil 41. İkinci aşama başlangıç simpleks tablosu.....	107
Şekil 42. İkinci aşamada uyumsuzluğun giderildiği başlangıç simpleks tablosu....	107
Şekil 43. İkinci aşama için simpleks algoritması ile oluşan simpleks tablosu	108
Şekil 44. Problemin temel değişkenlerine karşılık gelen çözümü.....	108
Şekil 45. Örnek 22 için oluşan tablolar	109
Şekil 46. Yardımcı problem başlangıç tablosu için temel değişkenler ve değerleri ..110	
Şekil 47. Yardımcı problem için birinci simpleks algoritması ile elde edilen tablo ..110	
Şekil 48. Yardımcı problem için ikinci simpleks algoritması ile elde edilen tablo ..111	
Şekil 49. İkinci aşamada başlangıç simpleks tablosu	111
Şekil 50. İkinci aşama simpleks algoritması sonucunda oluşan tablo	112
Şekil 51. Örnek 22'nin optimum çözümü	112
Şekil 52. Dual simpleks algoritmasının uygulanacağı matris.....	116
Şekil 53. Temel değişkenlerin ile elde edilen başlangıç simpleks matrisi.....	117
Şekil 54. Dual simpleks birinci aşama sonunda elde edilen matris.....	117
Şekil 55. Dual simpleks ikinci aşama sonunda elde edilen matris	118
Şekil 56. Örnek 23 için optimum sonuç	118
Şekil 57. Dual simpleks algoritması için oluşturulan simpleks matrisi.....	119
Şekil 58. Örnek 24 için birinci aşama sonucunda oluşan simpleks tablosu	120
Şekil 59. Standart simpleks algoritmasının birinci aşaması sonucunda oluşan.....	121
Şekil 60. Simpleks algoritmasının ikinci aşaması sonucunda oluşan matris.....	121
Şekil 61. Örnek 24 için optimum çözüm	122
Şekil 62. Örnek 25 için başlangıç simpleks tablosu	123
Şekil 63. Birinci aşama sonucunda oluşan simpleks tablosu.....	124

Şekil 64. İkinci aşama sonucunda oluşan simpleks tablosu	124
Şekil 65. Kısıtlamaların sağ tarafındaki değişimler için uygunluk aralığı	126
Şekil 66. Amaç fonksiyonunun katsayılarındaki değişimler için uygunluk aralığı. 126	



GİRİŞ

Optimizasyon belirli kısıtlamalar altında en iyi çözümü elde etme işlemidir. Bu çözüm bir firmanın minimum maliyetle maksimum kazanç elde etmesi anlamında olabileceği gibi bir bahçenin minimum maliyetle çevrilebilmesi problemi vb. olarak da karşımıza çıkabilmektedir. Bu tip problemler doğrusal (lineer) problemler olabileceği gibi doğrusal (lineer) olmayan problemler de olabilirler. Bu çalışmada doğrusal optimizasyon problemleri ele alınmaktadır.

Optimizasyon problemi, objektif veya hedef fonksiyonu olarak adlandırılan bir maksimizasyon veya minimizasyon probleminin belirli kısıtlamalar altındaki en uygun çözümünü bulmayı amaçlar. Eğer hedef fonksiyonu ile sınırlamalar lineer olarak elde edilmiş ise problem *doğrusal (lineer) programlama problemi* olarak adlandırılır. Doğrusal programlama problemlerinde kısıtlamalar sonlu sayıda eşitlik veya eşitsizlik sisteminden oluşur. Bu kısıtlamalar altında hedef fonksiyonunu minimum veya maksimum yapan en uygun çözüm belirlenir.

Genel olarak bir doğrusal optimizasyon problemi

$$\begin{array}{ll} \max CX & \min CX \\ AX \leq b & \text{veya} \quad AX \geq b \\ X \geq 0 & X \geq 0 \end{array} \quad (1)$$

şeklinde verilebilir. Burada

$$C = [c_1, c_2, \dots, c_n], \quad A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}_{m \times n}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}, \quad X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

dir. (1)'de minimize veya maksimize edilecek olan CX fonksiyonuna *amaç fonksiyonu*, *objektif fonksiyon* veya *hedef fonksiyonu*, $\leq$ veya $\geq$ gibi ifadelerin olduğu $AX \geq b$, $AX \leq b$, $X \geq 0$ gibi ifadelere ise *kısıtlamalar* adı verilir. Bu kısıtlamalar altında hedef fonksiyonunun maksimum veya minimum değerini aldığı X vektörünün değerleri elde edilmeye çalışılır. Problemdeki eşitsizlikleri sağlayan kümeye problemin *uygun çözüm kümesi*, bu uygun çözüm kümesinden verilen problemi maksimum veya minimum yapan değere ise *optimum çözüm* adı verilir (Coşkun, 2022).

Doğrusal programlama problemlerinde model oluşturulurken genelde aşağıdaki süreç işletilir.

a. Karar değişkenlerinin belirlenmesi

(1) problemdeki $x_1, x_2, \dots, x_n$ değişkenlerine karar değişkenleri adı verilir. Problemde belirlenmesi gereken her bir değişken bir karar değişkeni ile temsil edilir. Bu karar değişkenlerinin sağladığı bütün değerler belirlenir. Daha sonra bu değerlerden çözümü optimal yapan değer belirlenir. Genel olarak karar değişkenleri $x_i, i = 1, 2, \dots, n$ notasyonları ile gösterilir.

b. Amaç veya hedef fonksiyonun belirlenmesi

(1) problemdeki CX ile ifade edilen fonksiyona objektif veya hedef fonksiyonu adı verilir. Bu problem minimize veya maksimize edilecek olan problemi ifade eder. Elde edilen karar değişkeni değerlerinden bu fonksiyonu minimize veya maksimize eden değer, problemin aranan çözümüdür.

c. Kısıtlayıcıların belirlenmesi

(1) problemde $AX \leq b$ veya $AX \geq b$ ifadeler kısıtlamalar olarak adlandırılır. Kısıtlamalar, hedef fonksiyonun hangi koşullar altında minimize veya maksimize edileceğini belirtirler. Belirlenecek karar değişkenleri bu kısıtlamaları sağlayacak şekilde seçilmelidir. Bu kısıtlamalar genel olarak hepsi küçük eşit ($\leq$), hepsi büyük eşit ($\geq$) olarak elde edilebileceği gibi bunların kombinasyonu ve eşitlikleri içerek şekilde de verilebilir.

d. İşaret kısıtlamalarının belirlenmesi

(1) problemde $X \geq 0$ kısıtlaması karar değişkenlerinin negatif değerler bulundurmayacağını gösterir. Genelde doğrusal programlama problemleri üretim planlaması, personel planlaması, ulaştırma problemleri vb. gibi günlük hayattan problemleri içerdiğinden bu değerlerin negatif olması çoğu zaman anlamlı değildir. Dolayısı ile karar değişkenleri negatif olmayan değerler olarak belirlenirler.

Doğrusal programlama problemlerinin çözümlerinde kullanılacak bazı temel kavramlar şunlardır:

Uygun bölge: Doğrusal programlamada bütün kısıtlamaları sağlayan noktaların oluşturduğu bölgeye *uygun bölge* adı verilir.

Köşe noktaları: Uygun bölge genellikle bir geometrik şekle karşılık gelir. Bu uygun çözümün köşeleri *köşe noktaları* olarak adlandırılır.

Uygun çözüm: Uygun bölgede içerisinde yer alan herhangi bir noktadan elde edilen çözüme *uygun çözüm* adı verilir.

Optimal uygun çözüm: Uygun çözümler içerisinde amaç fonksiyonunu maksimum veya minimum yapan çözüme *optimal çözüm* adı verilir.

Doğrusal programlama problemlerinde ilk olarak verilen optimizasyon probleminin matematiksel modeli oluşturulur. Bu matematiksel modelin oluşturulmasını bir örnek üzerinden ele alalım:

Örnek 1. (Coşkun, 2022) Bütünleme sınavlarına hazırlanan bir öğrenci A ve B derslerinden sınava girecektir. Bu derslerin sınav hazırlığı için toplam 40 saatlik bir çalışma süresi vardır. Öğrenci önceki deneyimlerine göre, A ve B derslerine bir saatlik çalışmanın getirisinin sırasıyla yaklaşık olarak yüz üzerinden 3 ve 5 puan olduğunu tahmin etmektedir. Ayrıca öğrenci, önceki deneyimlerinden A dersinin zorluğundan dolayı A dersine, B dersine çalışma zamanından 3 kat daha fazla çalışması gerektiğini tahmin etmektedir. Buna göre öğrenci yaklaşık olarak A ve B derslerine en az kaç saat çalışmalıdır?

Matematiksel Model. Karar değişkenlerimiz A ve B dersleri için gerekli çalışma süreleri olup sırasıyla bu değişkenleri x_1 ve x_2 ile gösterelim. Bu durumda amaç fonksiyonu derslerin puan getirisini maksimize edilmesinden elde edileceği için

$$\text{Amaç fonksiyonu: } z = 3x_1 + 5x_2$$

olarak belirlenir. Kısıtlamalar ise problemdeki diğer verilerden elde edilir. Öğrencinin sınav hazırlığı için toplam 40 saatlik bir çalışma süresi mevcut olduğundan

$$x_1 + x_2 \leq 40$$

kısıtlaması elde edilir. Ayrıca A dersine, B dersinden 3 kat daha fazla çalışması gerektiğinden

$$x_1 \geq 3x_2$$

kısıtlaması bulunur. Ayrıca çalışma süreleri negatif olamayacağından dolayı işaret kısıtlamaları

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

olarak belirlenir. Dolayısı ile matematiksel model

$$\max z = 3x_1 + 5x_2$$

$$x_1 + x_2 \leq 40$$

$$x_1 \geq 3x_2$$

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

şeklinde elde edilir.

Örnek 2. (Coşkun, 2022) Bir lastik fabrikası yazlık, kışlık ve mevsimlik olmak üzere üç farklı otomobil lastiği üretimi yapmaktadır. Her bir lastik imalatı, fabrikanın üç farklı bölümden işlem görerek yapılmaktadır. Tablo 1’de bu bölümlerde lastiklerin işlem görme süreleri ve üretilen her bir lastik için elde edilmesi planlanan tahmini kâr verilmiştir. Ayrıca tablonun son kısmında her bir bölümün bu lastikleri üretmesi için kullanılabilecek süreleri belirtilmiştir.

Tablo 1. Lastik fabrikasındaki üretim

	Yazlık	Kışlık	Mevsimlik	Toplam Zaman
1. bölüm	1.5	1	2	90
2. bölüm	1	2	2	70
3. bölüm	2	1	1	80
Kâr	20	16	15	

Buna göre fabrika lastik üretiminden maksimum kâr elde edebilmesi için hangi tip lastikten ne kadar üretmelidir?

Matematiksel Model. Karar değişkenleri yazlık, kışlık ve mevsimlik üretilmesi planlanan lastik sayıları olmak üzere bu değişkenleri sırasıyla x_1, x_2 ve x_3 ile gösterelim. Dolayısı ile maksimize edilecek problem

$$\text{Amaç fonksiyonu: } z = 20x_1 + 16x_2 + 15x_3$$

şeklinde elde edilir. Problemin kısıtlamaları ise

$$1. \text{ bölümde işlem görebileceği iş gücü kısıtlaması: } 1.5x_1 + x_2 + 2x_3 \leq 90$$

$$2. \text{ bölümde işlem görebileceği iş gücü kısıtlaması: } x_1 + 2x_2 + 2x_3 \leq 70$$

$$3. \text{ bölümde işlem görebileceği iş gücü kısıtlaması: } 2x_1 + x_2 + x_3 \leq 80$$

olarak bulunur. Lastik üretimleri negatif olamayacağından dolayı işaret kısıtlamaları

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

dir. Dolayısı ile problemin matematiksel modeli

$$\max z = 20x_1 + 16x_2 + 15x_3$$

$$1.5x_1 + x_2 + 2x_3 \leq 90$$

$$x_1 + 2x_2 + 2x_3 \leq 70$$

$$2x_1 + x_2 + x_3 \leq 80$$

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

olarak belirlenmiş olur.

Örnek 1 ve 2’den görülebileceği üzere doğrusal programlama problemleri hayatın planlanması, üretim planlanması problemleri olarak karşımıza çıkabileceği gibi endüstri, tarım, ulaştırma, ekonomi, sağlık vb. gibi birçok alanda karşımıza çıkabilmektedir. Dolayısı ile bu problemlerin çözülmesi özellikle şirketler için önemli bir yere sahiptir.

Doğrusal programlama problemlerinin matematiksel modeli oluşturulduktan sonra amaç fonksiyonunu optimize eden değerin belirlenmesi için yöntemler geliştirilmiştir. Örnek 1’deki gibi iki karar değişkenine sahip problemleri çözmede en çok kullanılan yöntemlerden bir tanesi grafik yöntemidir. Bu yöntemde kısıtlar yardımıyla uygun çözüm bölgesi belirlenerek bu bölgede problemin optimum çözümü belirlenir. Üç veya daha çok karar değişkenli problemlerin çözümünde ise grafik yöntemi uygulanamaz. Bunun yerine kısıtlamaların türlerine göre Simpleks yöntemi, Dual yöntem, iki aşamalı Simpleks yöntemi, M yöntemi vb. gibi yöntemler geliştirilmiştir. Bu çalışmada bu yöntemlere kısaca değinilerek bu yöntemlerin programlaması üzerine çalışılmıştır.

Doğrusal programlama yöntemlerini çözmede bilgisayar yazılımları önemli yer tutar. Bu programlar ile optimizasyon problemlerinin cevapları kısa sürede elde edilebilmektedir. Bu yazılımlardan bazıları MATLAB (URL-1) programında linprog komutu, Mathematica (URL-2) programında LinearOptimization komutu, Maple (URL-3) LPSolve gibi matematiksel yazılımlar ile doğrusal programlama

problemlerini optimize eden sonuçlar elde edilebilmektedir. Fakat bu programların ücretli olması öğrencilerin bu programlara ulaşılabilirliğini azaltmaktadır. Ücretsiz olarak yüklenebilen Octave (URL-4) programında MATLAB'daki gibi linprog komutu, Maxima (URL-5) programında minimize_lp komutu ile doğrusal optimizasyon problemleri çözülebilmektedir. Fakat bu yazılımlar, sadece optimizasyon problemlerini optimum eden karar değişkenlerinin değerlerini ve optimum çözümü göstermektedirler. Öte yandan doğrusal programlama probleminde kullanılan yöntemlerde en uygun çözüme bir uygun köşe noktasından diğer uygun bir köşe noktasına geçilerek adım adım ulaşılmaktadırlar. Fakat değinilen programlar bu aşamalarda elde edilen çözümleri göstermemektedirler.

Octave ortamında hazırlanan bir diğer optimizasyon kodu ise Coşkun (Coşkun, 2022) tarafından hazırlanıp kendi sitesinde yayınlanan ve problemin aşamalarının program üzerinde verildiği programdır. Burada Simpleks yöntemi, Dual problemler ve iki aşamalı Simpleks yöntemi için ayrıntılı bir Octave kodu hazırlanmıştır. Alan ve Yeşilyurt (Alan ve Yeşilyurt, 2004) excel ortamında örnek lineer programlama probleminin nasıl çözülebileceğini göstermişlerdir. Arık (Arık, 2020) yüksek lisans tezinde doğrusal programlama çözücülerinin performans analizini yapmıştır. Bu çalışmada bazı ticari çözücüler ile açık kaynaklı çözücülerin performanslarını karşılaştırmıştır.

Bu çalışmada ise öğrenciler için kolayca ulaşılabilir olan Maxima programı tercih edilmiştir. Bu program ücretsiz olarak indirilebilir olup kütüphaneler oluşturmaya da elverişlidir. Bu program yardımıyla doğrusal programlama yöntemlerinden olan Grafik yöntemi, Simpleks yöntemi, Dual problemler, İki aşamalı Simpleks yöntemi, M yöntemlerindeki her bir aşamada elde edilen sonuçların kullanıcıya gösterilecek şekilde programlanması amaçlanmıştır. Bu sayede öğrenciler yaptıkları işlemlerin kontrolünü adım adım yapabileceklerdir. Ayrıca seçilen örnekler çoğunlukla endüstriyel alanda karşılaşılabilecek problemlerden seçilerek öğrencilerin matematiğe olan ilgilerinin artırılması hedeflenmektedir.

1. LİNEER OPTİMİZASYON YÖNTEMLERİ

1.1. Grafik Yöntemi

Grafik yöntemi iki karar değişkenine sahip problemler için kullanılan yöntemlerden biridir. Bu problemlerde kısıtlamaları sağlayan çözüm bölgesi grafiksel olarak çizdirilerek bu bölge içerisinde amaç fonksiyonunun optimum çözümü belirlenir. İki değişkenli lineer bir optimizasyon problemi

$$\begin{array}{ll} \max CX & \min CX \\ AX \leq b & \text{veya} \quad AX \geq b \\ X \geq 0 & X \geq 0 \end{array}$$

şeklinde verilebilir. Burada $C = [c_1 \ c_2]$, $X = [x \ y]^T$, $A_{2 \times 2}$ tipinde bir matris $b = [b_1 \ b_2]^T$ dir. Bu problemi grafiksel olarak çözmede Teorem 1'den yararlanılır.

Teorem 1. Bir lineer optimizasyon probleminin çözümü mevcutsa, bu uygun çözüm kümesinin köşe noktalarından birine karşılık gelir. Eğer herhangi iki komşu köşe noktada objektif fonksiyon aynı değere sahip ise, bu iki noktayı birleştiren doğru üzerindeki her nokta da bu problemin bir çözümüdür ve bu durumda problem sonsuz sayıda çözüme sahiptir.

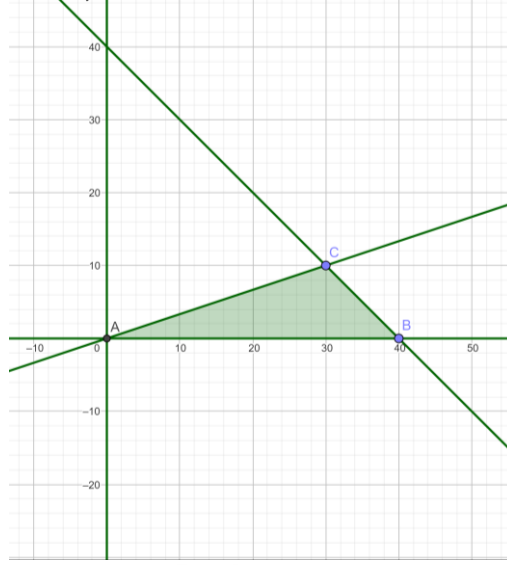
Teorem 1 optimizasyon probleminin uygun çözümünün köşe noktalarında olduğunu söylemektedir. Dolayısı ile grafik yönteminde kısıtlamaları sağlayan uygun bölgede grafiksel olarak çizdirilerek bu uygun bölgenin köşe noktaları belirlenir. Bu köşe noktalarından amaç fonksiyonunu optimum yapan değerler doğrusal programlama probleminin çözümüdür. Şimdi grafik yöntemini Örnek 1'deki probleme uygulayalım.

Örnek 3. Örnek 1'deki ders çalışma planlaması probleminde matematiksel model

$$\begin{array}{ll} \max z = 3x_1 + 5x_2 \\ x_1 + x_2 \leq 40 \\ x_1 \geq 3x_2 \\ x_1 \geq 0, \quad x_2 \geq 0 \end{array}$$

olarak elde edilmişti. Bu problemi grafik yöntemi ile çözelim.

Çözüm. Problemin kısıtlamaları sağlayan uygun çözüm bölgesi grafiksel olarak Şekil 1'de gösterilmiştir.



Şekil 1. Örnek 1'deki kısıtlamalar için uygun bölge

Şekil 1'de problemin kısıtlamalarını sağlayan uygun bölge gösterilmiştir. Teorem 1 ile optimal çözüm bu bölgenin köşe noktalarının birinde olduğu için bu bölgenin köşe noktaları belirlenmelidir. Şekil 1'deki bir köşegen noktası olan A noktası $x_1 = 3x_2$ ve $x_2 = 0$ doğrularının kesişim noktası olan $A(0,0)$ noktası olduğu açıktır. Diğer köşe noktası ise

$$x_1 + x_2 = 40 \text{ ve } x_2 = 0$$

doğrularının kesişim noktası olan $B(40,0)$ noktasıdır. Son köşe noktası ise

$$x_1 + x_2 = 40 \text{ ve } x_1 = 3x_2$$

doğrularının kesişim noktası olan $C(30,10)$ noktasıdır. Dolayısı ile optimal çözüm bu üç noktadan amaç fonksiyonunun maksimum yapan $C(30,10)$ noktasıdır. Dolayısıyla $x_1 = 30, x_2 = 10$ değerlerinden elde edilen çözüm

$$z = 3 \cdot 30 + 5 \cdot 10 = 140$$

problemin optimum uygun çözümdür.

Örnek 4. (Taha, 2010). Bir çiftlik özel bir yemden günde en az 800 kg kullanmaktadır. Bu özel yem mısır ve soya ununun karışımından hazırlanmakta olup karışım Tablo 2'deki bileşime uygun olarak elde edilmektedir:

Tablo 2. Özel yemdeki bileşenler

1 kg yemde kg cinsinden miktarlar			
	Protein	Lif	Maliyet (pb/kg)
Mısır	0.09	0.02	0.3
Soya Unu	0.6	0.06	0.9

Bu özel yemin en az %30 protein, en çok %5 lif içerecek şekilde hazırlanması zorunluluğu vardır. Buna göre hem minimum maliyetli hem de ihtiyaçları karşılayan yem karışımı için hangi üründen ne kadar kullanılmalıdır?

Çözüm. İlk olarak matematiksel modeli oluşturalım. x_1 ile özel karışımda kullanılan mısır miktarını, x_2 ile özel karışımda kullanılan soya unu miktarını gösterelim. Buna göre maliyetin minimum yapılması amaçlandığından Tablo 2'deki verilerden amaç fonksiyonu;

$$\min z = 0.3x_1 + 0.9x_2$$

dir. Problemin kısıtlamaları ise günde en az 800 kg yem kullanılması gerektiğinden

$$x_1 + x_2 \geq 800$$

olmalıdır. Ayrıca özel yemin en az %30 protein içermesi gerektiğinden

$$0.09x_1 + 0.6x_2 \geq 0.3(x_1 + x_2) \Rightarrow 0.21x_1 - 0.3x_2 \leq 0$$

en çok %5 lif içermesi gerektiğinden

$$0.02x_1 + 0.06x_2 \leq 0.05(x_1 + x_2) \Rightarrow 0.03x_1 - 0.01x_2 \geq 0$$

kısıtlamaları elde edilir. İşaret kısıtlamaları ise kullanılan mısır ve soya unu miktarları negatif olamayacağından dolayı

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

dır. Dolayısı ile matematiksel model

$$\min z = 0.3x_1 + 0.9x_2$$

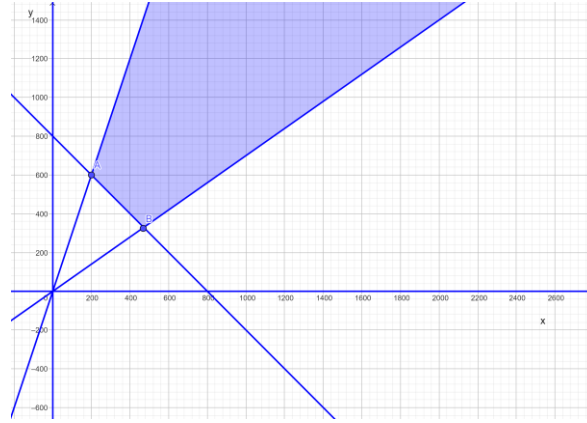
$$x_1 + x_2 \geq 800$$

$$0.21x_1 - 0.3x_2 \leq 0$$

$$0.03x_1 - 0.01x_2 \geq 0$$

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

olarak elde edilir. Bu problem için kısıtlamaları sağlayan uygun çözüm bölgesi Şekil 2'deki gibidir.



Şekil 2. Örnek 4'teki problem için uygun bölge

Şekil 2 ve Teorem 1'den optimizasyon problemini minimum yapan değer A veya B noktasıdır. A noktası

$$x_1 + x_2 = 800 \text{ ile } 0.03x_1 - 0.01x_2 = 0$$

doğrularının kesişim noktası olan $A(200,600)$ noktasıdır. B noktası ise

$$x_1 + x_2 = 800 \text{ ile } 0.21x_1 - 0.3x_2 = 0$$

doğrularının kesişim noktası olan $B(470.588,329.412)$ noktasıdır. Bu noktadan elde edilen çözümler uygun çözümlerdir. Problemin optimum uygun çözümü bunlardan amaç fonksiyonunu minimum yapan $x = 470.588, y = 329.412$ değerlerinin amaç fonksiyonunda yerine yazılması ile elde edilen $z = 437.647$ dir.

1.2. Simpleks Yöntem

Grafik yöntemi iki karar değişkenine sahip problemler için kullanılan bir yöntemdir. Üç veya daha fazla karar değişkenine sahip problemlerde uygulanamamaktadır. Bu durumda kısıtlamalardaki eşitsizliklerin durumuna göre farklı yöntemler uygulanmaktadır. Bunlardan ilki kısıtlamaların hepsinin küçük eşittir ($\leq$) olarak verilmesi durumunda kullanılan Simpleks yöntemidir.

Teorem 1 doğrusal programlamanın optimum sonucunun köşe noktalardan birinde olduğunu söyler. Bu teorem, doğrusal programlama problemlerinin çözümünde kullanılan Simpleks yönteminin anahtar kelimesidir. Bu yöntemde ilk olarak eşitsizlikler ile verilmiş kısıtlamalar bazı yapay değişkenlerin eklenmesi veya çıkartılması ile eşitlikler haline getirilirler. Bu yapıya problemin standart formu adı verilir. Daha sonra standart formdaki kısıtlamalar yardımıyla her biri bir uç noktaya karşılık gelen temel çözüm kümesi eşanlı (simultane) olarak elde edilir. Simpleks

algoritması, bu temel çözümlerden optimum olanını belirlemek için geliştirilmiş bir yöntemdir (Taha, 2010).

1.2.1. Standart Form

Genel olarak doğrusal programlama probleminde problemin çözümüne geçilmeden önce problemin standart formu elde edilir. Bir problemin standart hale gelmesi için aşağıdaki üç özelliği sağlaması gerekir.

i. İşaret kısıtlamaları dışındaki tüm kısıtlamalar, negatif olmayan sağ tarafa sahip eşitlikler olarak ifade edilmelidir.

ii. Hiçbir değişkenin negatif olmaması gerekir.

iii. Amaç fonksiyonu maksimum veya minimum olabilir. (Taha, 2010).

(i) şıkında verilen özellik ile kısıtlamalarda verilen eşitsizlikler ($\leq$ veya $\geq$) eşitlik ($=$) haline getirilmelidir. Bu kısıtlamalarda küçük eşittir ($\leq$) içeren eşitsizliklere yapay bir değişken eklenmesi ile kısıtlama eşitlik haline getirilebilir. Bu şekilde küçük eşittir içeren kısıtlamalara eklenen değişkenlere *dolgu değişkeni* adı verilir. Benzer şekilde büyük eşittir ($\geq$) şeklinde kısıtlama içeren problemlerde ise kısıtlamadan yapay bir değişken eksi ile çarpılarak eklenirse kısıtlama eşitlik formuna dönüşmüş olur. Bu şekilde büyük eşittir kısıtlamalarını eşitlik haline getirmek için kısıtlamaya eksi işareti ile eklenen değişkene *artık değişken* adı verilir. Buradaki dolgu veya artık değişkenler pozitif veya sıfır değerlerine sahip olup bu kısıtlamaların eklenmesi ile işaret kısıtlamalarına bu değişkenlerin büyük eşit sıfır olduğu eklenmelidir.

(iii) şıkında ise amaç fonksiyonunun maksimum veya minimum olacağı belirtilmişti. Bir maksimizasyon problemi minimizasyon problemine dönüştürülebilir. Eğer problem

$$\max f(x_1, x_2, \dots, x_n)$$

şeklinde verilmiş ise

$$\min -f(x_1, x_2, \dots, x_n)$$

problemi bu problem ile eşdeğerdir. Her iki problem için de bulunan değerler problemin optimum değerleridir.

Örnek 5. Örnek 1'deki ders çalışma planlaması matematiksel modeli

$$\max z = 3x_1 + 5x_2$$

$$x_1 + x_2 \leq 40$$

$$x_1 \geq 3x_2$$

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

şeklinde oluşturulmuştu. Bu problemin standart formu için ikinci kısıtlama

$$3x_2 - x_1 \leq 0 \text{ olarak yazılır ve amaç fonksiyonu eksi ile çarpılırsa}$$

$$\min(-z) = -3x_1 - 5x_2$$

$$x_1 + x_2 + s_1 = 40$$

$$-x_1 + 3x_2 + s_2 = 0$$

$$x_1 \geq 0, x_2 \geq 0, s_1 \geq 0, s_2 \geq 0$$

olarak verilebilir. Burada s_1 dolgu değişkeni iken s_2 ise artık değişkendir. Burada maksimizasyon problemi minimizasyon problemine dönüştürülmüştür. Fakat (iii)'ten standart form için böyle bir zorunluluk bulunmadığını da belirtmek gerekir.

Örnek 6. Örnek 2'deki lastik üretim fabrikası için matematiksel model

$$\max z = 20x_1 + 16x_2 + 15x_3$$

$$1.5x_1 + x_2 + 2x_3 \leq 90$$

$$x_1 + 2x_2 + 2x_3 \leq 70$$

$$2x_1 + x_2 + x_3 \leq 80$$

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

olarak elde edilmişti. Bu modelde tüm kısıtlamalar küçük eşittir olarak verildiğinden kısıtlamaları eşitlik haline getirmek için her birine dolgu değişkeni eklenmelidir.

Dolayısı ile s_1, s_2, s_3 dolgu değişkenleri olmak üzere standart formu

$$\max z = 20x_1 + 16x_2 + 15x_3$$

$$1.5x_1 + x_2 + 2x_3 + s_1 = 90$$

$$x_1 + 2x_2 + 2x_3 + s_2 = 70$$

$$2x_1 + x_2 + x_3 + s_3 = 80$$

$$x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, s_1 \geq 0, s_2 \geq 0, s_3 \geq 0$$

olarak elde edilir.

1.2.2. Temel Çözümün Belirlenmesi

Standart forma getirilen doğrusal programlama problemleri n bilinmeyenli (ya da değişkenli) m adet denklemden oluşur. Burada $m < n$ dir. Bu n değişken için iki durum söz konusudur.

- i. $n - m$ tane sıfır değerini alacak değişken
- ii. m adet eşitliğin çözülmesi ile belirlenecek olan m adet değişken.

Eğer m adet değişken tek bir çözümü veremekteyse bu değişkenlere *temel değişkenler* denir. Diğer $n - m$ değişken *temel dışı değişkenler* olarak adlandırılır. Eğer temel değişkenlerin hepsi negatif olmayan değerler alıyorsa temel çözüme *uygun çözüm* denir. Aksi halde çözüm uygun olmayan çözümdür.

Dolayısı ile n değişken sayılı, m eşitlik için maksimum temel uygun çözüm sayısı

$$\binom{n}{m} = \frac{n!}{m!(n-m)!}$$

ile hesaplanır (Taha, 2010).

$$x_1 + 2x_2 + x_3 + 3x_4 + x_5 = 6$$

$$3x_1 + 4x_2 + 2x_3 + 6x_4 + 5x_5 = 12$$

şeklinde $n = 5$ bilinmeyenli $m = 2$ denklemi göz önüne alalım. Bu problem için maksimum temel çözüm kümesi sayısı

$$\binom{5}{2} = \frac{5!}{2!(5-2)!} = 10$$

tanedir. Şimdi problemin uygun çözümünün olduğu ve uygun çözümün olmadığı durumlarını gözlemleyelim.

Durum 1. Temel dışı değişkenler (sıfır değerini alanlar) x_3, x_4, x_5 olsun. Bu durumda denklem sistemi

$$x_1 + 2x_2 = 6$$

$$3x_1 + 4x_2 = 12$$

halini alır. Bu denklemin çözümü $x_1 = 0, x_2 = 3$ olup bunlar $x_1 \geq 0, x_2 \geq 0$ olduğundan temel çözüm uygundur.

Durum 2. Temel dışı değişkenler x_1, x_2, x_4 olsun. Bu durumda

$$2x_3 + x_5 = 6$$

$$x_3 + 5x_5 = 12$$

denkleminin çözümü $x_3 = 2, x_5 = 2$ olup tek çözüm elde edilip iki değer de pozitif olduğundan çözüm uygun çözümdür.

Durum 3. Temel dışı değişkenler x_2, x_3, x_4 olsun. Bu durumda

$$x_1 + x_5 = 6$$

$$3x_1 + 5x_5 = 12$$

denklem sisteminin çözümü $x_1 = 9, x_5 = -3$ dür. Sistemin tek çözümü olmasına rağmen $x_5 < 0$ olduğundan çözüm uygun çözüm değildir.

Durum 4. Temel dışı değişkenler x_1, x_3, x_5 olsun. Bu durumda

$$2x_2 + 3x_4 = 6$$

$$4x_2 + 6x_4 = 12$$

denklem sistemi elde edilir. Bu durumda denklem sisteminin sonsuz çözüme sahip olduğuna dikkat ediniz. Dolayısı ile sistemin tek çözümü yoktur.

1.2.3. Simpleks Algoritması

Simpleks yöntemi 1947 yılında George Dantzig tarafından geliştirilmiştir. Teorem 1’de belirtildiği üzere problemin optimal çözümü daima uygun çözüm bölgesinin köşesindedir. Simpleks yöntemi, bir uç noktaya karşılık gelen temel uygun çözümden başlayarak amaç fonksiyonunu iyileştiren başka bir uç noktaya karşılık gelen uygun çözüme ulaşmayı amaçlar. Amaç fonksiyonunu iyileştiren daha uygun temel çözüm ise, sıfır değeri alan mevcut temel dışı değişkenin değerinde bir artış ile sağlanır. Yani bu işlem, temel dışı değişkenin sonraki adımda temel değişken olması ile mümkündür. Bunun sağlanması için ise, mevcut temel değişkenlerden birinin temel dışı değişken (yani değerinin sıfır) olması gerekmektedir. Bu durumda, temel dışı değişken sonraki adımda temel değişken olduğu için bu değişkene çözüme *giren değişken*, bu değişkenin sisteme girmesi için temel değişkenlerden birinin temel dışı değişken olması gerektiği için bu temel dışı değişkene ise *çıkan değişken* adı verilir (Taha, 2010).

Şimdi, Simpleks yöntemini bir problem üzerinde inceleyelim.

Örnek 8. (Hillier, 2019). Bir firma pencere ve cam kapılar için yüksek kaliteli cam üretimi yapmaktadır. Firmanın üç üretim fabrikası mevcut olup 1. fabrikada alüminyum çerçeve ve aksamaları, 2. fabrikada ahşap çerçeveler ve 3. fabrikada ise cam üretimi ile ürünlerin montajı yapılmaktadır. Son zamanlardaki kazançtaki azalmadan dolayı üst yönetim, şirketin üretim hattında kârsız ürünlerinin üretimi durdurmaya ve açığa çıkan üretim kapasitesiyle yüksek satış potansiyeli olan iki yeni ürün piyasaya sürmeye karar vermiştir. Bu ürünler

1. ürün 8 *ft* alüminyum çerçeveli cam kapı
2. ürün 4 × 6 *ft* ahşap çerçeveli çift cam pencere

şeklindedir. 1. ürünün üretim aşamasında 2. fabrikaya ihtiyaç duyulmamakta iken 2. ürünün üretiminde ise 1. fabrikaya ihtiyaç duyulmaktadır. Pazarlama bölümü, her iki ürünün fabrikalarındaki üretim adetince satış yapabileceği sonucuna varmıştır. Ancak her iki ürün de 3. fabrikada aynı üretim kapasitesi için yarışacaklarından hangi üründen ne kadar üretilmesinin daha kârlı olabileceği bilinmemektedir. Bu soruya cevap bulmak için firma tarafından bir bilirkişi ekibi oluşturulmuştur. B ekibi ile şirketin yönetimi arasındaki iletişim sonunda tablodaki veriler elde edilmiştir.

Tablo 3. Cam firması için veriler

	Parti başına üretim zamanı(saat)		Haftalık mevcut üretim zamanı (saat)
	1.ürün	2.ürün	
1. fabrika	1	0	4
2. fabrika	0	2	12
3. fabrika	3	2	18
Parti başına kâr	3.000	5.000	

Buna göre fabrikalarda Tablo 3'te verilen sınırlı üretim kapasiteleri üzerinden toplam kârı maksimum yapmak için hangi üründen ne kadar üretilmesi gerekir?

Çözüm. Matematiksel modeli oluşturmak için 1. ürünün haftalık üretilen parti adedinin x_1 , 2. ürünün haftalık üretilen parti adedinin x_2 olduğunu kabul edelim. 1. fabrikanın haftalık mevcut üretim zamanı 4 saat, 2. fabrikanın 12 saat ve 3. fabrikanın 18 saat olduğundan

$$x_1 \leq 4$$

$$2 x_2 \leq 12$$

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

kısıtlamaları elde edilir. Her bir ürünün üretiminden elde edilen kâr 1. ürün için 3 000 TL, 2. ürün için 5 000 TL olduğundan amaç fonksiyonu

$$\max z = 3000x_1 + 5000x_2$$

olarak elde edilir. Karar değişkenlerinin işaret kısıtlamaları ise $x_1 \geq 0, x_2 \geq 0$ olduğundan matematiksel model

$$\max z = 3000x_1 + 5000x_2$$

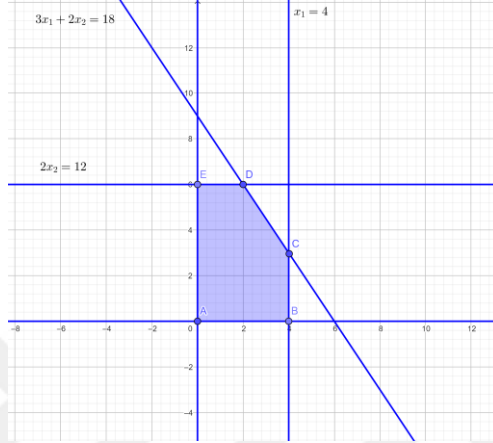
$$x_1 \leq 4$$

$$2 x_2 \leq 12$$

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

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

olarak elde edilir. Bu model iki karar değişkeni içerdiğinden grafik yöntemi ile çözülebilir. Kısıtlamaların sağlandığı uygun çözüm bölgesi Şekil 3'deki gibidir.



Şekil 3. Örnek 8 için uygun çözüm bölgesi

Şekil 3 ve Teorem 1'den problemi optimum yapan çözüm A, B, C, D, E noktalarından birindedir. $A(0,0), B(4,0), E(0,6)$ olduğu açıktır. C noktası ise $x_1 = 4$, $3 x_1 + 2x_2 = 18$ doğrularının kesişim noktası olan $C(4,3)$ noktasıdır. D noktasının koordinatları ise $2x_2 = 12$, $3 x_1 + 2x_2 = 18$ doğrularının kesişim noktasından $D(2,6)$ dir. Bu noktalardan amaç fonksiyonunu maksimum yapan nokta $D(2,6)$ noktasıdır.

Şimdi problemi Simpleks algoritması ile çözelim. İlk olarak problemi standart forma getirelim. Kısıtlamaların hepsi küçük eşittir şeklinde olduğundan bütün kısıtlamalara dolgu değişkenleri eklenmelidir. İşaret kısıtlamalarına da bu dolgu değişkenlerinin işaretlerinin negatif olamayacağı eklenmelidir. Dolayısı ile problemin standart formu

$$\max z - 3000x_1 - 5000x_2$$

$$x_1 + s_1 = 4$$

$$2 x_2 + s_2 = 12$$

$$3 x_1 + 2x_2 + s_3 = 18$$

$$x_1 \geq 0, x_2 \geq 0, s_1 \geq 0, s_2 \geq 0, s_3 \geq 0$$

olarak bulunur. Bu denklemleri bir tabloda ifade edebiliriz.

Tablo 4. Başlangıç Simpleks tablosu

Temel değişkenler	z	x_1	x_2	s_1	s_2	s_3	Çözüm
z	1	-3000	-5000	0	0	0	0
s_1	0	1	0	1	0	0	4
s_2	0	0	2	0	1	0	12
s_3	0	3	2	0	0	1	18

Tablonun ikinci satırı amaç fonksiyonunun

$$z - 3000x_1 - 5000x_2 = 0$$

şeklinde ifade edilmesi ile elde edilmişken diğer satırlar, kısıtlamaların katsayıları ile elde edilmiştir. Denklemlerin sağ tarafları ise çözüm sütununda belirtilmiştir.

Tablo 4'den sütunları birim vektörlere karşılık gelen s_1 ve s_2 dolgu değişkenleri uygun temel çözüm değişkenleridir. Temel dışı değişkenler ise sütunları birim vektörlere karşılık gelmeyen x_1 ve x_2 olarak seçilebilir. Bu durumda $x_1 = 0$, $x_2 = 0$ olup dolgu değişkenleri $s_1 = 4, s_2 = 12, s_3 = 18$ değerlerini alır. Buradaki değerler çözüm sütunundaki değerlerdir. Bu durumda maksimizasyon probleminin çözümü birinci satırın çözüm sütununda yer alan 0 olur. Bu çözüm Şekil 3'deki A uç noktasına karşılık gelmektedir.

Öte yandan bu başlangıç çözümün optimal çözüm olmadığı Şekil 3'den açıktır. Çünkü x_1 in katsayısı 3000 ve x_2 katsayısı 5000 olup x_1 'deki bir birim artma amaç fonksiyonunu 3000 birim arttırırken x_2 'deki bir birim artma amaç fonksiyonu 5000 bir arttıracaktır. Simpleks algoritmasında bir sonraki adımda amaç fonksiyonunun değerinin artması için temel dışı değişkenlerden biri temel değişken olurken temel değişkenlerden biri de temel dışı değişken olacaktır. Temel dışı değişkenlerinden x_1 veya x_2 nin sisteme dahil olması amaç fonksiyonunun değerini arttırırken x_2 'nin bir birimlik değişimi x_1 'e göre daha fazla bir artıma neden olduğundan x_2 'nin temel değişken olarak seçilerek sisteme girmesi artış hızını arttıracaktır. Dolayısı ile x_2 sisteme giren değişken olur. Bunu Tablo 4 üzerinde belirlemek için tabloda amaç fonksiyonu $z - 3000x_1 - 5000x_2$ olarak ifade edildiğinden temel dışı değişkenlerden z satırındaki mutlak değerce en büyük değere sahip olan elemanın sisteme gireceği anlaşılmaktadır. Bu negatif elemanlardan

mutlak değerce en büyük elemanın bulunduğu sütuna *pivot(anahtar) sütunu* adı verilir. Burada pivot sütunu x_2 değişkeninin bulunduğu sütundur.

Tablo 5. Pivot sütunu

Temel değişkenler	z	x_1	x_2	s_1	s_2	s_3	Çözüm
z	1	-3000	-5000	0	0	0	0
s_1	0	1	0	1	0	0	4
s_2	0	0	2	0	1	0	12
s_3	0	3	2	0	0	1	18

x_2 temel değişken olacağından ikinci adımda sıfırın üstüne çıkmalıdır. Şekil 3'den A noktasının komşu uç noktası olan E noktasında x_2 fonksiyonun değeri artarken buna paralel olarak amaç fonksiyonun değeri de artacaktır. Dolayısı ile bu durum çözümün komşu uç nokta olan E noktasına kayacağı ve daha iyi bir çözüm elde edilebileceği anlamına gelmektedir. Şimdi şekilsel olarak elde edilen durumu Simpleks algoritması ile nasıl elde edileceğini belirlemeye çalışalım. Şekil 3'ten E noktası ikinci kısıtın x_2 eksenini ile kesiştiği noktaya karşılık gelir. Kesişim noktası ise ikinci kısıtın sağ tarafının x_2 nin katsayısına bölünmesi ile elde edilir. Aşağıdaki tabloda bu bölümler gösterilmiştir.

Tablo 6. Pivot sütunundaki oranlar

Temel değişkenler	z	x_1	x_2	s_1	s_2	s_3	Çözüm	Oran
z	1	-3000	-5000	0	0	0	0	
s_1	0	1	0	1	0	0	4	$\frac{4}{0} = \infty$
s_2	0	0	2	0	1	0	12	$\frac{12}{2} = 6$
s_3	0	3	2	0	0	1	18	$\frac{18}{2} = 9$

Tablo 6'nın son sütunundaki oranlardan x_2 nin artış yönünü göstermesi sebebiyle negatif oranlar ve sonsuz oranları işleme almayacağız. Öte yandan çözüm sütunundaki katsayılar Simpleks algoritmasının kuralı gereği negatif değerler almamalıdır. Bu yüzden de pivot sütunundaki negatif sayılar oranı negatif yapacağından pivot sütunundaki negatif olan sayılar ile ilgilenmeyeceğiz. Tablo

6'nın oran sütununda negatif oran olmamasına rağmen sonsuz oran olduğundan o satır dikkate alınmayacaktır. Negatif olmayan oranlar arasından ise minimum olan oran x_2 değişkeninin yeni çözümdeki değeri olur ki bu değer 6 dır. Negatif ve sonsuz oranların bulunmadığı minimum orana sahip satıra ise *pivot(anahtar) satırı* adı verilir.

Tablo 7. Pivot satırı

Temel değişkenler	z	x_1	x_2	s_1	s_2	s_3	Çözüm	Oran
z	1	-3000	-5000	0	0	0	0	
s_1	0	1	0	1	0	0	4	$\frac{4}{0} = \infty$
s_2	0	0	2	0	1	0	12	$\frac{12}{2} = 6$
s_3	0	3	2	0	0	1	18	$\frac{18}{2} = 9$

Pivot satır ve pivot sütunun kesişimindeki elemana ise *pivot(anahtar) eleman* adı verilir. Bu problem için pivot eleman tablodaki kesişim noktası olan 2 elemanıdır. x_2 giren değişken olup bu değişkenin temel değişken olması ile temel değişkenler olan s_1, s_2, s_3 den biri çıkan değişken olmalıdır. Yani bu temel değişkenlerden birinin değeri sıfır değeri almaya zorlanarak temel dışı değişken olacaktır. En küçük oranla ilişkili değişken Tablo 7'deki pivot satırından da görüleceği üzere s_2 olup bu değişken çıkan değişken olacaktır. Dolayısı temel değişkenler ikinci aşamada (x_2, s_1, s_3) değişkenleridir. Yeni temel çözümün bulunması Gauss-Jordan yöntemine dayanır (Taha, 2010). Yani, pivot eleman değeri 1 yapılarak pivot sütundaki diğer elemanların değerleri sıfır olacak şekilde satır ve sütun işlemleri yapılır.

İlk olarak pivot elemanı 1 yapmak için pivot sütun 1/2 ile çarpılır. Bu durumda Tablo 8 elde edilir.

Tablo 8. Pivot elemanın değerinin 1 yapılması

Temel değişkenler	z	x_1	x_2	s_1	s_2	s_3	Çözüm
z	1	-3000	-5000	0	0	0	0
s_1	0	1	0	1	0	0	4
x_2	0	0	1	0	1/2	0	6
s_3	0	3	2	0	0	1	18

Daha sonra pivot elemanın bulunduğu sütundaki diğer elemanları sıfırlamak için pivot eleman 5000 ile çarpılarak z ile ilişkili olan satır ile, -2 ile çarpılarak da s_3 ile ilişkili olan son satır ile toplanır. Bu durumda Tablo 9 elde edilir.

Tablo 9. Gauss Jordan yöntemiyle eliminasyon

Temel değişkenler	z	x_1	x_2	s_1	s_2	s_3	Çözüm
z	1	-3000	0	0	2500	0	30000
s_1	0	1	0	1	0	0	4
x_2	0	0	1	0	1/2	0	6
s_3	0	3	0	0	-1	1	6

Tablo 9'dan temel değişkenlerin aldığı değerlerin $(x_2, s_1, s_3) = (6, 4, 6)$ olduğu elde edilir. Bu ise Şekil 3'teki A noktasına komşu olan $E(0, 6)$ noktasına karşılık gelir. Dolayısı ile Simpleks yönteminin ilk aşamasının tamamlanması ile yöntem, sistemin bir uç noktasındaki çözümünden bu noktaya komşu diğer bir uç noktasındaki çözümü elde etmiş olur.

Tablo 9'un z satırında negatif elemanın bulunması, elde edilen sonucun optimum çözüm olmadığını gösterir. Çünkü Tablo 9 ile amaç fonksiyonu

$$z = 3000x_1 - 2500s_2 + 3000$$

olarak elde edilir. Dolayısı ile x_1 'deki artım amaç fonksiyonun değerinin artmasına neden olacaktır. Bu nedenle x_1 sisteme giren değişken olacaktır. Buradan pivot sütunu x_1 ile ilişkili sütundur. Sistemden çıkan değişken ise ilk adımda olduğu gibi oranlara bakılarak elde edilir. Tablo 10'un son satırında bu oranlar verilmiştir.

Tablo 10. Pivot sütunundaki oranlar

Temel değişkenler	z	x_1	x_2	s_1	s_2	s_3	Çözüm	Oran
z	1	-3000	0	0	2500	0	30000	
s_1	0	1	0	1	0	0	4	$\frac{4}{1} = 4$
x_2	0	0	1	0	$\frac{1}{2}$	0	6	$\frac{6}{0} = \infty$
s_3	0	3	0	0	-1	1	6	$\frac{6}{3} = 2$

Tablo 10'daki oranlardan en küçük oran son satırdaki olduğu için o satırla ilişkili olan s_3 temel değişkeni çıkan değişken olacaktır. Dolayısı ile pivot eleman son satırdaki 3 elemanıdır. İlk aşamada yapıldığı gibi pivot eleman 1 yapılırsa Tablo 11 elde edilir.

Tablo 11. Pivot elemanın değerinin 1 yapılması

Temel değişkenler	z	x_1	x_2	s_1	s_2	s_3	Çözüm
z	1	-3000	0	0	2500	0	30000
s_1	0	1	0	1	0	0	4
x_2	0	0	1	0	$\frac{1}{2}$	0	6
x_1	0	1	0	0	$-\frac{1}{3}$	$\frac{1}{3}$	2

Gauss Jordan yöntemi ile pivot elemanın bulunduğu sütundaki diğer elemanları sıfırlamak için pivot eleman 3000 ile çarpılarak z ile ilişkili satır ile, -1 ile çarpılarak s_1 ile ilişkili olan satır ile toplanırsa Tablo 12 elde edilir.

Tablo 12. Gauss Jordan yöntemiyle eliminasyon

Temel değişkenler	z	x_1	x_2	s_1	s_2	s_3	Çözüm
z	1	0	0	0	1500	1000	36000
s_1	0	0	0	1	$\frac{1}{3}$	$-\frac{1}{3}$	2
x_2	0	0	1	0	$\frac{1}{2}$	0	6
x_1	0	1	0	0	$-\frac{1}{3}$	$\frac{1}{3}$	2

Tablo 12'den temel dışı değişkenlerin aldığı değerlerin $(x_1, x_2, s_1) = (2, 6, 2)$ olduğu görülür. Bu durumda amaç fonksiyonunun değeri 36000 olarak elde edilir. Tablo 12'nin z satırında negatif eleman bulunmamasından dolayı bulunan sonuç optimumdur. Dolayısı ile $(x_1, x_2) = (2, 6)$ değerleri problemi optimum yapan değerler olup problemin optimum değeri 36000 dir. Bu noktanın E noktasına komşu olan D uç noktasına karşılık gelen değer olduğuna dikkat ediniz. Dolayısı ile

Simpleks yöntemi her bir aşamada bir uç noktadan komşu uç noktadaki çözüme ulaşarak optimum çözümü aramaktadır.

Bu elde edilen sonucu firma için yorumlamak gerekirse 1. üründen haftada 2 tane, 2. üründen haftada 6 tane üretilmesi şirketin kârını maksimum yapar.

Örnek 8’de bir maksimizasyon problemi ile çalışılmıştır. Problemin bir minimizasyon problemi olması durumunda

$$\max z = \min(-z)$$

olduğundan benzer yaklaşım kullanılabilir. Bu durumda z satırındaki en büyük pozitif değere sahip amaç fonksiyonu katsayısı temel dışı değişken, çözüme giren değişken olur. Minimum değer ise z satırındaki katsayılar pozitif sayı kalmadığında elde edilen değerdir. Giren ve çıkan değişkenler belirlenirken optimumluk ve uygunluk koşullarına başvurulur.

Optimumluk koşulu: Bir maksimizasyon (veya minimizasyon) probleminde giren değişken, z satırındaki mutlak değerce en büyük değere sahip negatif (veya pozitif) katsayısı olan temel dışı değişkendir. İki katsayının değerinin eşit olması durumunda herhangi biri seçilebilir. Optimum sonuca ise, z satırındaki tüm temel dışı değişkenlerinin katsayılarının negatif olmama (veya pozitif olmama) koşulunda ulaşılır.

Uygunluk koşulu: Hem maksimizasyon hem de minimizasyon problemlerinde çıkan değişken, negatif veya sonsuz olmayan oranlar içinden en küçük değere sahip temel değişken olarak belirlenir. Optimumluk koşulunda olduğu gibi eşitlik durumunda oranlardan herhangi biri seçilebilir (Taha, 2010).

Örnek 8’den, optimumluk ve uygunluk koşullarından simpleks algoritması aşağıdaki gibi özetlenebilir (Taha, 2010):

Adım 0. Bir uygun çözüm belirlenir

Adım 1. Optimumluk koşulunu sağlayan giren değişken belirlenir. Böyle bir değişken yoksa algoritma sonlandırılır.

Adım 2. Uygunluk koşulunu sağlayan çıkan değişken belirlenir.

Adım 3. Gauss Jordan yöntemi ile yeni temel çözüm belirlenerek Adım 1 e gidilir. Simpleks algoritmasından görüleceği üzere yöntem iteratif (yinelemeli) bir yöntemdir. Dolayısı ile Simpleks adımlarının her birini iterasyon (yineleme) adı verilir.

Şimdi Simpleks yöntemini üç değişkenli bir probleme uygulayalım. Bilindiği üzere 3 değişken içeren problemlerde grafik yöntemi uygulanamamaktadır. Bu durumda Simpleks yöntemi ile çözüm araştırılır. Bunun için matematiksel modelini Örnek 2’de elde ettiğimiz lastik üretim firması modelini ele alalım.

Örnek 9. Örnek 2’deki lastik üreten firma için matematiksel model

$$\max z = 20x_1 + 16x_2 + 15x_3$$

$$1.5x_1 + x_2 + 2x_3 \leq 90$$

$$x_1 + 2x_2 + 2x_3 \leq 70$$

$$2x_1 + x_2 + x_3 \leq 80$$

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

şeklinde elde edilmişti. Bu problem için standart form ise Örnek 6’dan

$$\max z = 20x_1 + 16x_2 + 15x_3$$

$$1.5x_1 + x_2 + 2x_3 + s_1 = 90$$

$$x_1 + 2x_2 + 2x_3 + s_2 = 70$$

$$2x_1 + x_2 + x_3 + s_3 = 80$$

$$x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, s_1 \geq 0, s_2 \geq 0, s_3 \geq 0$$

olarak bulunmuştu. Bu problemi Simpleks yöntemi ile çözelim.

Çözüm. Bu problem için başlangıç Simpleks tablosu Tablo 13’teki gibi elde edilir.

Tablo 13. Örnek 9 başlangıç simpleks tablosu

Temel değişkenler	z	x_1	x_2	x_3	s_1	s_2	s_3	Çözüm
z	1	-20	-16	-15	0	0	0	0
s_1	0	1.5	1	2	1	0	0	90
s_2	0	1	2	2	0	1	0	70
s_3	0	2	1	1	0	0	1	80

z satırındaki negatif elemanlar içerisinde en büyük terim -20 olduğundan pivot sütun x_1 değişkeninin bulunduğu sütundur. Dolayısı ile x_1 giren değişken olacaktır. Çıkan değişken ise yeni temel değişkenin bulunduğu sütundaki oranlara bakılarak belirlenir. Bu oranlar Tablo 14’te gösterilmiştir.

Tablo 14. Pivot satırı belirlemek için oranlar

Temel değişkenler	z	x_1	x_2	x_3	s_1	s_2	s_3	Çözüm	Oran
z	1	-20	-16	-15	0	0	0	0	
s_1	0	1.5	1	2	1	0	0	90	$\frac{90}{1.5} = 60$
s_2	0	1	2	2	0	1	0	70	$\frac{70}{1} = 70$
s_3	0	2	1	1	0	0	1	80	$\frac{80}{2} = 40$

Tablo 14'teki oranlardan en küçük oran son satırdaki olduğu için o satırla ilişkili olan s_3 temel değişkeni çıkan değişken olacaktır. Dolayısı ile pivot eleman son satırdaki 2 elemanıdır. Pivot eleman 1 yapılırsa Tablo 15 elde edilir.

Tablo 15. Pivot elemanın değerinin 1 yapılması

Temel değişkenler	z	x_1	x_2	x_3	s_1	s_2	s_3	Çözüm
z	1	-20	-16	-15	0	0	0	0
s_1	0	1.5	1	2	1	0	0	90
s_2	0	1	2	2	0	1	0	70
x_1	0	1	$\frac{1}{2}$	$\frac{1}{2}$	0	0	$\frac{1}{2}$	40

Gauss Jordan yöntemi ile pivot elemanın bulunduğu sütundaki diğer elemanları sıfırlamak için pivot eleman 20 ile çarpılarak z ile ilişkili satır ile, -1 ile çarpılarak s_2 ile ilişkili olan satır ile, -1.5 ile çarpılarak s_1 ile ilişkili satır ile toplanırsa Tablo 16 elde edilir.

Tablo 16. 1. Simpleks aşaması sonunda elde edilen tablo

Temel değişkenler	z	x_1	x_2	x_3	s_1	s_2	s_3	Çözüm
z	1	0	-6	-5	0	0	10	800
s_1	0	0	$\frac{1}{4}$	$\frac{5}{4}$	1	0	$-\frac{3}{4}$	30
s_2	0	0	$\frac{3}{2}$	$\frac{3}{2}$	0	1	$-\frac{1}{2}$	30
x_1	0	1	$\frac{1}{2}$	$\frac{1}{2}$	0	0	$\frac{1}{2}$	40

Tablo 16'dan temel değişkenlerin aldığı değerlerin $(x_1, s_1, s_2) = (40, 30, 30)$ olduğu elde edilir. Tablo 16'nın z satırında negatif elemanın bulunması, elde edilen sonucun optimum çözüm olmadığını gösterir. Çünkü Tablo 16 ile amaç fonksiyonu

$$z = -6x_2 - 5x_3 = 800$$

olarak elde edilir. Dolayısı ile x_1 ve x_2 'deki artım amaç fonksiyonun değerinin artmasına neden olacaktır. x_2 değerinin katsayısı mutlak değerce daha büyük olduğundan x_2 sisteme giren değişken olacaktır. Dolayısı ile pivot sütunumuz x_2 ile ilişkili sütundur. Sistemden çıkan değişken ise ilk adımda olduğu gibi oranlara bakılarak elde edilir. Tablo 17'nin son satırında bu oranlar verilmiştir.

Tablo 17. Pivot eleman için oranlar

Temel değişkenler	z	x_1	x_2	x_3	s_1	s_2	s_3	Çözüm	Oran
z	1	0	-6	-5	0	0	10	800	
s_1	0	0	$\frac{1}{4}$	$\frac{5}{4}$	1	0	$-\frac{3}{4}$	30	$\frac{30}{1/4} = 120$
s_2	0	0	$\frac{3}{2}$	$\frac{3}{2}$	0	1	$-\frac{1}{2}$	30	$\frac{30}{3/2} = 20$
x_1	0	1	$\frac{1}{2}$	$\frac{1}{2}$	0	0	$\frac{1}{2}$	40	$\frac{40}{1/2} = 80$

Tablo 17'deki oranlardan en küçük oran s_2 satırındaki olduğu için çıkan değişken s_2 olacaktır. Dolayısı ile pivot eleman s_2 satırındaki $3/2$ elemanıdır. Pivot eleman 1 yapılırsa Tablo 18 elde edilir.

Tablo 18. Pivot elemanın değerinin 1 yapılması

Temel değişkenler	z	x_1	x_2	x_3	s_1	s_2	s_3	Çözüm
z	1	0	-6	-5	0	0	10	800
s_1	0	0	$\frac{1}{4}$	$\frac{5}{4}$	1	0	$-\frac{3}{4}$	30
x_2	0	0	1	1	0	$\frac{2}{3}$	$-\frac{1}{3}$	20
x_1	0	1	$\frac{1}{2}$	$\frac{1}{2}$	0	0	$\frac{1}{2}$	40

Pivot elemanın bulunduğu sütundaki diğer elemanları sıfırlamak için Gauss Jordan yöntemi uygulanırsa Tablo 19 elde edilir.

Tablo 19. 2. Simpleks aşaması sonunda elde edilen tablo

Temel değişkenler	z	x_1	x_2	x_3	s_1	s_2	s_3	Çözüm
z	1	0	0	1	0	4	8	920
s_1	0	0	0	1	1	$-\frac{1}{6}$	$-\frac{2}{3}$	25
x_2	0	0	1	1	0	$\frac{2}{3}$	$-\frac{1}{3}$	20
x_1	0	1	0	0	0	$-\frac{1}{3}$	$\frac{2}{3}$	30

Tablo 19'dan temel değişkenlerin aldığı değerlerin $(x_1, x_2, s_1) = (30, 20, 25)$ olduğu görülür. Bu durumda amaç fonksiyonunun değeri 920 olarak elde edilir. Tablo 19'un z satırında negatif eleman bulunmamasından dolayı bulunan sonuç optimumdur. Dolayısıyla

$$(x_1, x_2, x_3) = (30, 20, 0)$$

değeri problemi optimum yapan değerler olup problemin optimum değeri 920 dir.

Simpleks yöntemi, işaret kısıtlamaları hariç diğer kısıtlamaların hepsi küçük eşit olarak verildiğinde uygulanmaktadır. Bu durumda dolgu değişkenleri yardımıyla kısıtlamalar standart forma çevrilerek başlangıç Simpleks tablosu oluşturulmuştur. Başlangıçta Simpleks tablosundan dolgu değişkenlerinin (s_i ler) bulunduğu sütunlar birim vektörlere karşılık gelirken, karar değişkenlerin (x_i ler) bulunduğu sütunlar genellikle birim vektörden farklıdır. Dolayısı ile genel olarak başlangıç uygun çözüm olarak karar değişkenlerinin değerlerinin sıfır ($x_i = 0$) olduğu, dolgu değişkenlerinin değerlerinin kısıtlamaların sağındaki pozitif değerleri aldığı çözüm ile başlanır. Yani, genelde başlangıç uygun çözüm olarak karar değişkenlerinin temel dışı değişkenler olduğu, dolgu değişkenlerinin temel değişkenler olduğu uygun çözüm kümesi ile Simpleks yöntemine başlanır.

Öte yandan matematiksel modelde $\geq$ veya $=$ gibi kısıtlamaların bulunduğu çözümde eğer kısıtlamaların sağ tarafı sıfır değilse bu durumda karar değişkenleri olan x_i lerin temel dışı değişken olduğu durum kısıtlamaları sağlamamaktadır. Dolayısıyla sıfır çözümü başlangıç çözümü olarak alınamamaktadır. Bu durumda Simpleks yöntemine başlamak için başlangıç uygun temel çözümün nasıl

seçilebileceği sorusu ortaya çıkmaktadır. Bu başlangıç uygun çözümü bulmak için en çok kullanılan yöntem yapay değişkenler kullanmaktır. Bu değişkenler başlangıçtaki yinelemede dolgu değişkeni gibi alınırlar, sonraki yinelemelerde ortadan kaldırılırlar. Bu yöntem kullanılarak geliştirilen yöntemler *M yöntemi ve iki aşamalı yöntemidir* (iki faz yöntemi).

1.3. *M* yöntemi

M yönteminde, standart hale getirilen problemde dolgu değişkenine sahip olmayan ($\leq$ kısıtlaması içermeyen) herhangi bir eşitliğe birer yapay R_i değişkeni eklenir. Bu değişkenler Simpleks algoritmasının sonraki yinelemelerde sıfır değerini alarak temel dışı değişken olmaları sağlanacak olan yapay değişkenlerdirler. Bunun sağlanması için amaç fonksiyonunda bu değişkenlere bir *ceza* katsayısı verilir. Verilen bu ceza katsayısı M ile gösterilir. Bu değer yeterince büyük, pozitif değere sahip bir sayıdır. Ayrıca maksimizasyon problemlerinde R_i yapay değişkeninin katsayısı $-M$, minimizasyon problemlerinde ise $+M$ olacaktır. Yani, amaç fonksiyonuna eğer problem maksimizasyon problemi ise $-MR_i$, minimizasyon problemi ise $+MR_i$ terimleri ilave edilecektir. Bu ceza ile optimizasyon problemlerinde, ileriki Simpleks yinelemelerinde bu yapay değişkenler sıfır almaya zorlanacaktır. (Taha, 2010).

Örnek 10. (Coşkun, 2022)

$$\begin{aligned}\max z &= x_1 + 2x_2 \\ 4x_1 + x_2 &\geq 8 \\ 2x_1 + 3x_2 &\leq 12 \\ x_1 \geq 0, x_2 &\geq 0\end{aligned}$$

problemini *M* yöntemi ile çözünüz.

Çözüm. Problemin standart formu

$$\begin{aligned}\max z &= x_1 + 2x_2 \\ 4x_1 + x_2 - s_1 &= 8 \\ 2x_1 + 3x_2 + s_2 &= 12 \\ x_1 \geq 0, x_2 \geq 0, s_1 \geq 0, s_2 &\geq 0\end{aligned}$$

olup burada s_1 artık değişken, s_2 ise dolgu değişkenidir. *M* yöntemi ile dolgu değişkeni içermeyen birinci kısıtlamaya R_1 yapay değişkeni eklenir. Amaç

fonksiyonuna ise problem maksimizasyon problemi olduğu için $-MR_1$ ifadesi eklenir. Dolayısı ile doğrusal programlama problemi

$$\max z = x_1 + 2x_2 - MR_1$$

$$4x_1 + x_2 - s_1 + R_1 = 8$$

$$2x_1 + 3x_2 + s_2 = 12$$

$$x_1 \geq 0, x_2 \geq 0, s_1 \geq 0, s_2 \geq 0$$

şeklinde elde edilir. Bu, başlangıç Simpleks tablosu ile ifade edilirse

Tablo 20. Başlangıç simpleks tablosu

Temel değişkenler	z	x_1	x_2	s_1	s_2	R_1	Çözüm
z	1	-1	-2	0	0	M	0
R_1	0	4	1	-1	0	1	8
s_2	0	2	3	0	1	0	12

Tablo 20'den dikkat edilirse başlangıçtaki temel değişkenler R_1 ile s_2 olup bu değişkenlerin değerleri $R_1 = 8, s_2 = 12$ dir. Bu durumda temel çözüm

$$z = x_1 + 2x_2 - MR_1 = -8M$$

olarak elde edilir. Fakat Tablo 20'deki başlangıç Simpleks tablosunun z satırındaki çözüm sütunu 0 olarak görünmektedir. Bu uygunsuzluk z satırında R_1 katsayısının sıfır olmamasından ortaya çıkmaktadır. Bu durum R_1 'in yerine uygun kısıt denklemleri kullanılarak giderilebilir. Bu durumda R_1 satırının $-M$ ile çarpılıp z satırı ile toplanması ile yeni bir z satırı oluşturulması ile mümkündür. Böylece yeni Simpleks tablosu Tablo 21'deki gibi elde edilir.

Tablo 21. Düzenlenmiş başlangıç Simpleks tablosu

Temel değişkenler	z	x_1	x_2	s_1	s_2	R_1	Çözüm
z	1	$-1 - 4M$	$-2 - M$	M	0	0	$-8M$
R_1	0	4	1	-1	0	1	8
s_2	0	2	3	0	1	0	12

Tablo 21 ile temel değişkenler olan R_1 ve s_2 değerlerine karşılık gelen çözüm uygun hale getirilmiş olur. Dolayısı ile problem Simpleks algoritması uygulamaya uygun hale gelmiştir. İlk olarak pivot sütunu belirleyelim. Problem maksimizasyon

problemi olduğundan z satırındaki mutlak değerce en büyük negatif elemanın bulunduğu sütun seçilmelidir. M yeterince büyük bir sayı olduğundan pivot sütun x_1 sütununa karşılık gelir. Dolayısı ile sisteme giren temel değişken x_1 olacaktır.

Tablo 22. Pivot satır ve sütun

Temel değ.	z	x_1	x_2	s_1	s_2	R_1	Çözüm	Oran
z	1	$-1 - 4M$	$-2 - M$	M	0	0	$-8M$	
R_1	0	4	1	-1	0	1	8	$8/4 = 2$
s_2	0	2	3	0	1	0	12	$12/2 = 6$

Öte yandan pivot satır ise Tablo 22'deki oran sütunundaki küçük orana karşılık gelen R_1 satırıdır. Dolayısı ile sistemden çıkan değişken R_1 olup pivot eleman Tablo 22'deki iki çizginin kesiştiği yerdir. Bu eleman 1 yapılırsa Tablo 23 elde edilir.

Tablo 23. Pivot sütun

Temel değişkenler	z	x_1	x_2	s_1	s_2	R_1	Çözüm
z	1	$-1 - 4M$	$-2 - M$	M	0	0	$-8M$
x_1	0	1	$1/4$	$-1/4$	0	$1/4$	2
s_2	0	2	3	0	1	0	12

Gauss Jordan eliminasyon yöntemi ile pivot elemanın bulunduğu sütundaki diğer elemanlar sıfırlanırsa Tablo 24 elde edilir.

Tablo 24. Başlangıç Simpleks tablosuna Gauss Jordan yöntemiyle eliminasyon

Temel değişkenler	z	x_1	x_2	s_1	s_2	R_1	Çözüm
z	1	0	$-7/4$	$-1/4$	0	$1/4 + M$	2
x_1	0	1	$1/4$	$-1/4$	0	$1/4$	2
s_2	0	0	$5/2$	$1/2$	1	$-1/2$	8

Tablo 24 ile temel değişkenler x_2 ile R_1 olup temel değişkenlerin değerleri çözüm sütunu ile $x_1 = 2, s_2 = 8$ tür. Bu durumda çözüm $z = x_1 + 2x_2 - MR_1 = 2$ olup z satırındaki çözüm sütunu ile uyumludur. M yeterince büyük bir değer olduğu ve z sütununda negatif eleman bulunduğu için Simpleks yönteminde bir sonraki yinelemeye geçilir. Tablo 24'ten z satırında negatif elemana sahip sütun x_2

olduğundan sisteme giren değişken x_2 olmalıdır. Tablo 25'deki oran sütunundan minimum değere sahip satır s_2 olduğundan s_2 temel dışı değişken olacaktır.

Tablo 25. Pivot elemanın bulunması

Temel değ.	z	x_1	x_2	s_1	s_2	R_1	Çözüm	Oran
z	1	0	$-7/4$	$-1/4$	0	$1/4 + M$	2	
x_1	0	1	$1/4$	$-1/4$	0	$1/4$	2	$\frac{2}{1/4} = 8$
x_2	0	0	$5/2$	$1/2$	1	$-1/2$	8	$\frac{8}{5/2} = \frac{16}{5}$

Pivot eleman 1 yapılır ve Gauss Jordan yöntemi uygulanırsa Tablo 26 elde edilir.

Tablo 26. Gauss Jordan eliminasyon yöntemi ile eliminasyon

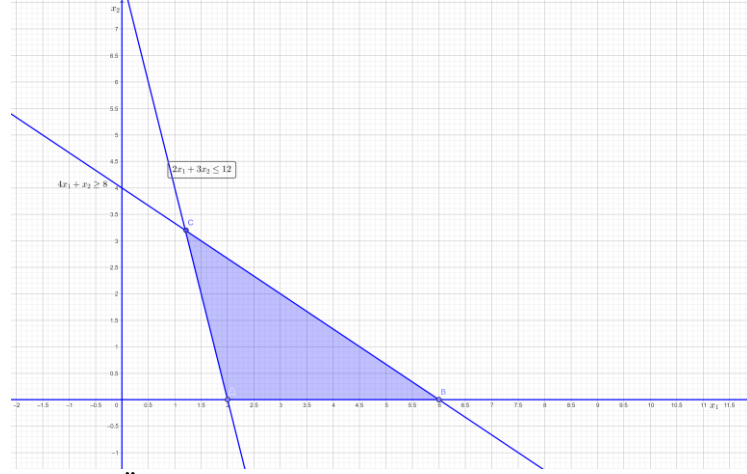
Temel değişkenler	z	x_1	x_2	s_1	s_2	R_1	Çözüm
z	1	0	0	$1/10$	$7/10$	$-1/10 + M$	$38/5$
x_1	0	1	0	$-3/10$	$-1/10$	$3/10$	$6/5$
x_2	0	0	1	$1/5$	$2/5$	$-1/5$	$16/5$

Tablo 26 ile temel değişkenlerin değerleri $x_1 = 6/5, x_2 = 16/5$ olup problemin bu değerler için çözümü $38/5$ olarak elde edilir. Öte yandan M çok büyük bir pozitif tam sayı olduğundan Simpleks algoritması tamamlanmış olup problemin çözümü $x_1 = 6/5, x_2 = 16/5$ için

$$z = x_1 + 2x_2 = \frac{6}{5} + \frac{32}{5} = \frac{38}{5}$$

olarak elde edilmiş olur.

Sonucun doğruluğunu test etmek için grafik yöntemi kullanılırsa uygun çözüm bölgesi Şekil 4'teki gibi elde edilir.



Şekil 4. Örnek 10'daki problem için uygun çözüm bölgesi

Şekil 4'ten problemin optimum çözümü $A(2,0)$, $B(6,0)$ veya iki doğrunun kesişim noktası olan $C(6/5, 16/5)$ noktalarından birinde olduğu görülür. Problemi maksimum yapan değer ise C noktası olup bu M yöntemi ile bulunan sonuç ile aynıdır.

Öte yandan M yönteminde dikkat edilmesi gereken iki önemli durum vardır:

i. M yönteminde, eğer problemin uygun bir çözüm uzayı yoksa, doğrusal programla problemin sonucunu veren son Simpleks tablosundaki yapay değişkenlerden bir tanesi sıfırdan farklı değer aldığı durumlar oluşabilir. Bu durum problemin uygun çözümünün olmadığını belirtir.

ii. M yönteminde dolgu değişkenleri olarak yapay değişkenler probleme ilave edilirken amaç fonksiyonuna da ceza katsayıları eklenmektedir. Buradaki M değerlerinin yeterince büyük pozitif sabitler olduğunu belirtmiştik. Yeterince büyük olması teorik olarak $M \rightarrow \infty$ anlamına gelmektedir. Fakat bilgisayarlar sonlu sayılar ile işlem yaptıklarından bu M değerleri büyük değerler seçilmelidir. Fakat çok büyük değerler seçilmesi ise beraberinde yuvarlama hataları da getireceğinden bu hesaplamaların doğruluğunu bozacağına dikkat edilmelidir. Bundan dolayı bu değerlerin seçilmesinde yeterince büyük ifadesi kullanılmış olup bu değerlerin dikkatli seçilmesi önem kazanmaktadır. Bundan dolayı M yöntemi yerine iki aşamalı Simpleks yöntemi uygulamak problemler için daha uygun olabilir (Taha, 2010).

1.4. İki Aşamalı Simpleks Yöntemi

İki aşamalı Simpleks yöntemi M yöntemindeki M sabitini tamamen ortadan kaldırarak yuvarlama hatalarında oluşan sorunları ortadan kaldırmak için oluşturulmuştur. Bu yöntem iki aşamadan oluşmakta olup ilk aşamada bir uygun başlangıç çözüm bulunmaya çalışılır. İkinci aşamada ise bu başlangıç ile yola çıkarak orijinal problemi çözmeye geçilir. Yani,

I. aşamada, M yönteminde olduğu gibi problem standart hale getirildikten sonra $=$ veya $\geq$ içeren kısıtlamalara yapay değişkenler eklenir. Daha sonra yapay değişkenlerin toplamını minimum kılan temel çözüm elde edilir. Eğer bu optimizasyon probleminde elde edilen değer pozitif ise, esas problemin uygun çözümü yok anlamına gelir. Aksi halde bu değer sıfıra eşit olursa ikinci aşamaya geçilir.

II. aşamada, I. aşamada bulunan temel uygun çözüm, orijinal problemin başlangıç çözümü alınır. Sonrasında yapay değişkenler probleminden çıkartılarak Simpleks algoritması ile orijinal problemin çözümü elde edilir (Taha, 2010).

Şimdi, M yönteminde ele alınan problemi iki aşamalı Simpleks yöntemi ile çözelim.

Örnek 11.

$$\max z = x_1 + 2x_2$$

$$4x_1 + x_2 \geq 8$$

$$2x_1 + 3x_2 \leq 12$$

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

problemini iki aşamalı Simpleks yöntemi ile çözüünüz.

Çözüm. M yönteminde olduğu gibi yapay dolgu değişkenleri ile problem

$$4x_1 + x_2 - s_1 + R_1 = 8$$

$$2x_1 + 3x_2 + s_2 = 12$$

$$x_1 \geq 0, x_2 \geq 0, s_1 \geq 0, s_2 \geq 0$$

olarak elde edilir. I. aşamada

$$\min r = R_1$$

$$4x_1 + x_2 - s_1 + R_1 = 8$$

$$2x_1 + 3x_2 + s_2 = 12$$

$$x_1 \geq 0, x_2 \geq 0, s_1 \geq 0, s_2 \geq 0$$

problemini ele alalım. Bu durumda Simpleks tablosu Tablo 27'deki gibi oluşur.

Tablo 27. Yardımcı problem için başlangıç Simpleks tablosu

Temel değişkenler	x_1	x_2	s_1	s_2	R_1	Çözüm
r	0	0	0	0	-1	0
R_1	4	1	-1	0	1	8
s_2	2	3	0	1	0	12

M yönteminde olduğu gibi başlangıçtaki temel değişkenler s_2 ile R_1 olup bunların değerleri $s_2 = 12, R_1 = 8$ olup bu durumda I. aşamada oluşturulan problemin çözümü $r = R_1 = 8$ olmalıdır. Fakat tablonun r satırının son satırında bu sonuç 0 olarak görünmektedir. Dolayısı ile M yönteminde olduğu gibi bu uyumsuzluğu düzeltmek için R_1 satırı ile r satırı toplanarak r satırına yazılır. Buradan Tablo 28 elde edilir.

Tablo 28. Yardımcı problem için düzenlenmiş başlangıç Simpleks tablosu

Temel değişkenler	x_1	x_2	s_1	s_2	R_1	Çözüm
r	4	1	-1	0	0	8
R_1	4	1	-1	0	1	8
s_2	2	3	0	1	0	12

Tablo 28'den r satırındaki çözüm sütunu 8 olup, $R_1 = 8$ olduğundan $\min r = 8$ ile uyumlu hale gelmiş olur. Şimdi Simpleks algoritmasını uygulayalım. Yardımcı problem minimizasyon problemi olduğundan r satırındaki en büyük pozitif elemanın bulunduğu x_1 sütunu pivot sütundur. Dolayısı ile x_1 değişkeni sonraki aşamada temel değişken olacaktır. Pivot satır için oranlara bakılırsa pivot satırının R_1 satırı olduğu elde edilir. Dolayısıyla sonraki aşamada R_1 temel dışı değişken olacaktır. Pivot eleman 1 yapılır ve Gauss Jordan yöntemi uygulanırsa Tablo 29 elde edilir.

Tablo 29. Yardımcı problem için Gauss Jordan yöntemiyle eliminasyon

Temel değişkenler	x_1	x_2	s_1	s_2	R_1	Çözüm
r	0	0	0	0	-1	0
x_1	1	1/4	-1/4	0	1/4	2
s_2	0	5/2	1/2	1	-1/2	8

r satırında pozitif eleman kalmadığından yapay problem için Simpleks aşaması sona ermiş olur. Öte yandan çözüm 0 olarak elde edildiğinden ikinci aşamaya geçiş yapılır. Eğer Simpleks aşamasından sonra çözüm 0'dan farklı olsaydı problemin uygun çözümü yok anlamına gelirdi.

II. aşamada ise problemdeki yapay değişken çıkartılarak başlangıçta ele alınan maksimizasyon problemine geçilir. Tablo 29'da oluşturulan Simpleks tablosu ile yeni problem

$$\max z = x_1 + 2x_2$$

$$x_1 + \frac{1}{4}x_2 - \frac{1}{4}s_1 = 2$$

$$\frac{5}{2}x_2 + \frac{1}{2}s_1 + s_2 = 8$$

$$x_1 \geq 0, x_2 \geq 0, s_1 \geq 0, s_2 \geq 0$$

halini almış olur. Dolayısı ile esas problem için başlangıç Simpleks tablosu Tablo 30'daki gibi elde edilir.

Tablo 30. Esas problem için başlangıç Simpleks tablosu

Temel değişkenler	x_1	x_2	s_1	s_2	Çözüm
z	-1	-2	0	0	0
x_1	1	1/4	-1/4	0	2
s_2	0	5/2	1/2	1	8

Tablo 30'dan dikkat edilirse esas değişkenlerin değerleri $x_1 = 2, s_2 = 8$ olup bu durumda çözüm $z = x_1 + 2x_2 = 2 + 2 \cdot 0 = 2$ olmalıdır. Fakat Tablo 30'da z satırındaki çözüm 0 olarak görülmektedir. Dolayısı ile ilk bu uyumsuzluk giderilmelidir. Bunun için x_1 satırı ile z satırı toplanıp z satırına yazılırsa Tablo 31 elde edilir.

Tablo 31. Esas problem için düzenlenmiş başlangıç Simpleks tablosu

Temel değişkenler	x_1	x_2	s_1	s_2	Çözüm
z	0	-7/4	-1/4	0	2
x_1	1	1/4	-1/4	0	2
s_2	0	5/2	1/2	1	8

Tablo 31'e göre başlangıç temel çözüm $x_1 = 2, s_2 = 8$ temel değişkenlerine karşılık gelen 2 çözümdür. Dikkat edilirse bu tablo M yöntemindeki Tablo 24 ile R_1 sütununun bulunmadığı durum ile aynıdır. M yönteminde olduğu gibi bu aşamadan sonra Simpleks algoritması uygulanırsa tablo verileri Tablo 26'daki gibi elde edilir. Sonuç olarak temel değişkenlerin değerleri $x_1 = 6/5, x_2 = 16/5$ olup problem çözümü $38/5$ olarak elde edilir.

1.5. Dual Problem

Yukarıda geliştirilen doğrusal programlama modeli *primal* problem olarak isimlendirilmektedir. Her primal probleme ona dual olan başka bir optimizasyon problemi karşılık gelir. Bu problemlerin incelenerek aralarındaki bağıntının araştırılması düalete teorisinin temel problemini oluşturur. Bu primal problem ile ilişkili olan problemin ikinci şekline *dual (ikiz) problem* adı verilir. Bu elde edilen dual problem bazı şartlar altında primal probleme göre daha kolay veya daha az adımda çözülebilmektedir (Özturan, 2022). Dolayısı ile dual problemlerin incelenmesi önem kazanmaktadır.

Bir diğer önemli nokta ise bir doğrusal programlama problemi model parametrelerinin değerlerindeki küçük değişimlerin optimum çözümde oluşturacağı değişimleri belirleme yeteneğiyle donatılmış olmasıdır. Bu işlemler *duyarlılık analizi* olarak adlandırılmaktadır (Taha, 2010). Dual problem hakkındaki bilgiler primal çözüm üzerindeki duyarlılık analizi yapmamızda önemli rol oynadığından dual problemin ortaya konulması önemlidir.

1.5.1. Dual Problemin Kurulması

Bir primal problemi dual probleme çevirmek işlemi optimizasyonun anlamına (maksimizasyon veya minimizasyon), kısıtların tipine ($\geq, \leq$ veya $=$) ve değişkenlerin işaretine göre tanımlanmıştır. Bu durumun karışıklığa yol açabileceği düşüncesi ile primal problemler standart formları tek bir tanım altında toplanmıştır:

$$\begin{aligned} \min \text{ veya } \max z &= Cx \\ Ax &= b \\ x &\geq 0, b \geq 0, C \geq 0 \end{aligned} \tag{2}$$

Burada,

$$C = [c_1, c_2, \dots, c_n], \quad A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}_{m \times n}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}, \quad X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

olup X vektörünün elemanları $x_j, j = 1, 2, \dots, n$ değişkenleri problemin yapısına göre dolgu veya artık değişkenleri içerebilir. Dual problemin kısıtları ve değişkenleri primal problemden aşağıdaki gibi elde edilir:

- i. m tane primal kısıt denkleminin her biri için bir dual değişken tanımlanır.
- ii. n tane primal değişkenin her biri için bir dual kısıt tanımlanır.
- iii. Dual problemin sol taraf katsayıları ilgili primal değişkenin kısıt katsayılarına (A matrisinin sütunları) eşittir. Sağ taraf sabiti ise ilgili primal değişkenin amaç fonksiyonun katsayısına (C vektörünün elemanlarına) eşittir.
- iv. Dual problemin amaç fonksiyonu katsayıları primal kısıt denklemlerinin sağ tarafına (b vektörünün elemanlarına) eşittir (Taha, 2010).

Eğer dual değişkenler $y_1, y_2, \dots, y_m$ ile gösterilirse bu dört özellik Tablo 32'deki gibi özetlenebilir.

Tablo 32. Primal problem ile dual problem arasındaki ilişki-1 (Taha, 2010)

Primal değişkenler							
	x_1	x_2	$\cdots$	x_j	$\cdots$	x_n	
Dual değişkenler	c_1	c_2	$\cdots$	c_j	$\cdots$	c_n	
y_1	a_{11}	a_{12}	$\cdots$	a_{1j}	$\cdots$	a_{1n}	b_1
y_2	a_{21}	a_{22}	$\cdots$	a_{2j}	$\cdots$	a_{2n}	b_2
$\vdots$	$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$
y_m	a_{m1}	a_{m2}	$\cdots$	a_{mj}	$\cdots$	a_{mn}	b_m
$j.$ Dualin kısıtı							Dual'in amacı

Tablo 33. Primal problem ile dual problem arasındaki ilişki-2 (Taha, 2010)

Primal Problem	Dual Problem		
	Amaç	Kısıt Tipi	Değişken işareti
Maksimizasyon	Minimizasyon	$\geq$	Sınırlandırılmamış
Minimizasyon	Maksimizasyon	$\leq$	Sınırlandırılmamış

Örnek 12. Örnek 10 ve 11'de ele aldığımız problemin dualini bulalım:

Örnek 10'daki problem ve standart formu

Primal problem	Standart formu
$\max z = x_1 + 2x_2$	$\max z = x_1 + 2x_2$
$4x_1 + x_2 \geq 8$	$4x_1 + x_2 - x_3 = 8$
$2x_1 + 3x_2 \leq 12$	$2x_1 + 3x_2 + x_4 = 12$
$x_1 \geq 0, x_2 \geq 0$	$x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0$

Problemdeki iki tane kısıt denklemi olduğundan dual parametreler y_1, y_2 ile gösterilebilir. Bu durumda Tablo 33'ten primal problem maksimizasyon olduğundan dual problem minimizasyon olacaktır ve kısıtlamaların her biri $\geq$ dir. Tablo 32'den ise dual problem aşağıdaki gibi elde edilir:

$$\begin{aligned}
 \min 8y_1 + 12y_2 \\
 4y_1 + 2y_2 &\geq 1 \\
 y_1 + 3y_2 &\geq 2 \\
 -y_1 &\geq 0 \text{ (veya } y_1 \leq 0) \\
 y_2 &\geq 0.
 \end{aligned}$$

Örnek 12. (Coşkun, 2022). Bir diyetisyen Ürün-I ve Ürün-II adını verdiği iki ürünün uygun miktarda karışımı ile bir bitkisel ilaç hazırlamayı düşünmektedir. Ürün-I ve ürün-II nin içerdiği demir, C vitamini ve kolesterol miktarları Tablo 34'te verilmiştir.

Tablo 34. Diyetisyen problemi

	Demir	C Vitamini	Kolesterol
Ürün-I	3 mg	4 mg	2 mg
Ürün-II	6 mg	2 mg	3 mg

Diyetisyen ilacı en az 1500 mg demir ve 800 mg C vitamini içerecek şekilde hazırlamayı istemektedir. Buna göre bitkisel ilacın minimum kolesterol içermesi için hangi tip üründen ne kadar içermelidir?

Çözüm. İlk olarak problemin matematiksel modelini oluşturalım. x_1 ile bitkisel ilacın içerdiği Ürün-I miktarını, x_2 ile bitkisel ilacın içerdiği Ürün-II miktarını gösterelim. Buna göre ilaç en az 1500 mg demir içermesi gerektiğinden

$$3x_1 + 6x_2 \geq 1500$$

ve en az 800 mg C vitamini içermesi gerektiğinden

$$4x_1 + 2x_2 \geq 800$$

kısıtlamaları elde edilir. Bitkisel ilacın minimum miktarda kolesterol içermesi istendiğinden amaç fonksiyonu

$$\min z = 2x_1 + 3x_2$$

olarak elde edilir. İşaret kısıtlamaları ile matematiksel model

$$\min z = 2x_1 + 3x_2$$

$$3x_1 + 6x_2 \geq 1500$$

$$4x_1 + 2x_2 \geq 800$$

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

şeklinde elde edilmiş olur. Problemin standart hali ise artık değişkenlerin probleme eklenmesi ile

$$\min z = 2x_1 + 3x_2$$

$$3x_1 + 6x_2 - x_3 = 1500$$

$$4x_1 + 2x_2 - x_4 = 800$$

$$x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0$$

olarak elde edilir. Bu durumda problemin dual formu y_1 ve y_2 dual değişkenler olmak üzere

$$\max z = 1500y_1 + 800y_2$$

$$3y_1 + 4y_2 \leq 2$$

$$6y_1 + 2y_2 \leq 3$$

$$-y_1 \leq 0 \text{ (veya } y_1 \geq 0)$$

$$-y_2 \leq 0 \text{ (veya } y_2 \geq 0)$$

şeklinde. Bu problemin çözümüne dual problemlerin çözüm yöntemleri kısmında değinilmektedir.

1.5.2. Dual Simpleks Yöntemi

Dual Simpleks algoritmasında doğrusal programlamaya optimum fakat uygun olmayan bir çözümle başlanır. Daha sonraki yinelemelerde optimumluğu bozmadan uygun çözüme ulaşılmaya çalışılır. Uygun çözüm elde edildiğinde algoritma sonlandırılır.

Dual Simpleks yönteminin başlangıç tablosunda en az bir tane uygun olmayan (< 0) temel değişkenin bulunduğu optimum amaç satırı mevcut olmalıdır. Optimumluğu koruyup uygun çözüme ulaşmaya çalışmak için aşağıdaki iki koşuldan yararlanılır:

Dual uygunluk koşulu: çıkan değişken negatif değere sahip olan temel değişkenler içerisinde mutlak değerce en büyük değere sahip olan olarak seçilir. Eşitliğin olması durumunda herhangi biri seçilebilirken eğer negatif değere sahip olan temel değişken yoksa algoritma sonlandırılır.

Dual optimumluk koşulu: Giren değişken ise temel olmayan değişkenler içerisinde

$$\min_{\substack{x_j \text{ temel} \\ \text{olmayan değişken}}} \left\{ \left| \frac{z_j - c_j}{\alpha_{rj}} \right| : \alpha_{rj} < 0 \right\}$$

koşulunu sağlayan seçilir. Burada α_{rj} , tablodaki çıkan değişkenin satırı ile giren değişkenin sütununun çakıştığı kısıt katsayısı olup eşitliğin olması durumunda herhangi biri seçilebilir (Taha, 2010).

Örnek 13. Örnek 12’de elde edilen matematiksel model

$$\min z = 2x_1 + 3x_2$$

$$3x_1 + 6x_2 \geq 1500$$

$$4x_1 + 2x_2 \geq 800$$

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

için dual Simpleks algoritmasını uygulayalım.

Bu problem için problemi standart hale getirdiğimizde artık değişkenler x_3 ve x_4 olmak üzere

$$\min z = 2x_1 + 3x_2$$

$$3x_1 + 6x_2 - x_3 = 1500$$

$$4x_1 + 2x_2 - x_4 = 800$$

$$x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0$$

elde edilir. Eğer artık değişkenlerin bulunduğu denklemler -1 ile çarpılırsa

$$\min z = 2x_1 + 3x_2$$

$$-3x_1 - 6x_2 + x_3 = -1500$$

$$-4x_1 - 2x_2 + x_4 = -800$$

$$x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0$$

bulunur. Dual Simpleks algoritması için başlangıç tablosu Tablo 35'te verilmiştir.

Tablo 35. Dual Simpleks yöntemi başlangıç tablosu

Temel değişkenler	x_1	x_2	x_3	x_4	Çözüm
z	-2	-3	0	0	0
x_3	-3	-6	1	0	-1500
x_4	-4	-2	0	1	-800

Tablo 35'ten temel değişken değerlerinin $x_3 = -1500, x_4 = -800$ elde edilir. Fakat bu durum x_3 ve x_4 değerlerinin pozitif olması ile çeliştiği için uygun bir çözüm değildir. Öte yandan $z_j - c_j \leq 0, j = 1, 2, 3, 4$ olduğundan bu başlangıç temel çözümün optimum olduğu anlamına gelmektedir. Dolayısı ile bu başlangıç tablosu dual Simpleks yönteminin optimum ve uygun olmayan şartlarını sağladığı tablodur (Taha, 2010).

Dual uygunluk koşuluna göre çıkan değişkenin x_3 olduğu anlaşılır. Çünkü x_3 mutlak değerce en büyük değere sahip değişkendir. Giren değişken ise Tablo 36'daki oranlara bakıldığında x_2 olduğu görülür.

Tablo 36. Dual Simpleks yöntemi başlangıç tablosundaki oranlar

Temel değişkenler	x_1	x_2	x_3	x_4	Çözüm
z	-2	-3	0	0	0
x_3	-3	-6	1	0	-1500
x_4	-4	-2	0	1	-800
Oran	2/3	1/2			

Tablo 37'deki pivot eleman 1 yapılır ve Gauss Jordan algoritması uygulanırsa Tablo 38 elde edilir.

Tablo 37. Dual Simpleks yöntemi birinci yineleme sonuçları

Temel değişkenler	x_1	x_2	x_3	x_4	Çözüm
z	-1/2	0	-1/2	0	750
x_2	1/2	1	-1/6	0	250
x_4	-3	0	-1/3	1	-300

Tablo 37’den temel deęiřken deęerleri $x_1 = 250, x_4 = -300$ olup uygun bir çözüml deęildir. Çözüml sütununda negatif eleman olduęu için dual Simpleks yöntemindeki yinelemelere devam edilir. Tek negatif elemanın bulunduęu satır x_4 satır olduęundan ikinci yinelemede çıkan deęiřken x_4 tür. Tablo 38’deki oranlardan ise giren deęiřkenin x_1 olduęu görülür.

Tablo 38. Dual Simpleks yöntemi birinci yineleme sonucundaki oranlar

Temel deęiřkenler	x_1	x_2	x_3	x_4	Çözüml
z	$-1/2$	0	$-1/2$	0	750
x_2	$1/2$	1	$-1/6$	0	250
x_4	-3	0	$-1/3$	1	-300
Oran	$1/6$		$3/2$		

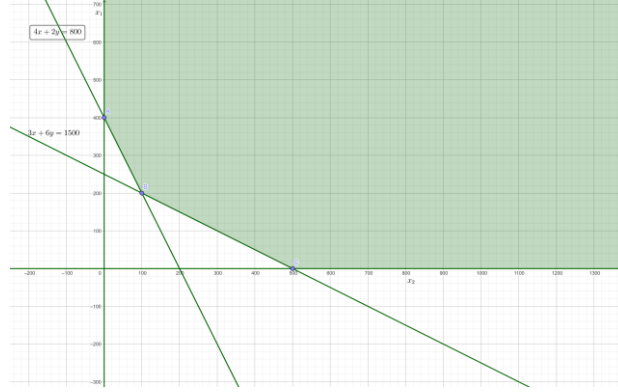
x_4 satırını ile x_1 sütunun kesiřimindeki eleman 1 yapılır ve Gauss-Jordan yöntemi uygulanırsa ikinci yineleme sonucunda Tablo 39 elde edilir.

Tablo 39. Dual Simpleks yöntemi ikinci yineleme sonuçları

Temel deęiřkenler	x_1	x_2	x_3	x_4	Çözüml
z	0	0	$-4/9$	$-1/6$	800
x_2	0	1	$-2/9$	$1/6$	200
x_1	1	0	$1/9$	$-1/3$	100

Tablo 39’dan temel deęiřken deęerleri $x_1 = 100, x_2 = 200$ olduęu görülür. Bu çözüml uygun çözüml olup problemin optimum çözümlünün 800 olduęu elde edilir.

Orijinal problem iki deęiřkene sahip olduęundan grafik yöntemi ile de çözümlülebilir. Bu durumda temel çözüml bölgesi Şekil 5’teki gibidir.



Şekil 5. Örnek 12 için uygun çözüm bölgesi

Şekil 12'den problemin optimum çözümü $A(0,400), B(100,200), C(500,0)$ noktalarından biridir. Bu değerlere karşılık gelen çözümler sırasıyla 1200, 800, 1000 olup minimum değer B noktasına karşılık gelen 800 değeridir. Bu Simpleks çözümü ile uyumludur.

1.6. Primal Problemin Matrisel Gösterimi

Simpleks hesaplamalarını izlemenin en iyi yollarından biri matris kullanmaktır. Dolayısı ile ilk olarak Simpleks yönteminde örnekler üzerinde anlattığımız Simpleks algoritmasını matrisel olarak ifade edelim. Bunun için genel bir doğrusal problemi şu şekilde ele alalım:

$$\begin{aligned} \max \text{ veya } \min z &= CX \\ AX &\leq b \\ X &\geq 0. \end{aligned} \tag{3}$$

Burada ,

$$X = [x_1 \ x_2 \ \dots \ x_{n-m}]^T, C = [c_1 \ c_2 \ \dots \ c_{n-m}], b = [b_1 \ b_2 \ \dots \ b_m]^T \text{ ve}$$

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

şeklindedir. Bu problemin standart formu ise dolu değişkenlerinin eklenmesi ile

$$\begin{aligned} \max \text{ veya } \min z &= CX \\ [A \ I]X &= b \\ X &\geq 0. \end{aligned}$$

olarak ifade edilebilir. Burada $I, m \times m$ 'lik birim vektör, $0, 1 \times m$ 'lik sıfır vektörü olmak üzere $C = [C \ 0] = [c_1 \ c_2 \ \dots \ c_n]$ ve X ise x vektörüne dolgu değişkenlerinin eklenmesi ile elde edilen $X = [x_1 \ x_2 \ \dots \ x_n]^T$ bilinmeyenler vektörüdür. Ayrıca not edelim ki eğer (3) problemindeki kısıtlar büyük eşit ($\geq$) kısıtlamalarını içerirse I vektörünün içerisinde -1 elemanları mevcut olacaktır. Eğer problemdeki kısıtlardan elde edilen denklem sistemi

$$[A \ I]X = b$$

vektörlerle

$$\sum_{j=1}^n P_j x_j = b$$

şeklinde ifade edilirse problem

$$\begin{aligned} \max \text{ veya } \min z &= \sum_{j=1}^n c_j x_j \\ \sum_{j=1}^n P_j x_j &= b \\ x_j &\geq 0, j = 1, 2, \dots, n. \end{aligned} \tag{4}$$

olarak ifade edilebilir. Burada $P_j, [A \ I]$ matrisinin j . sütun vektörüdür. Dikkat edilirse problemin m adet temel değişkeni olabilmesi için P_j vektörlerinin m tanesinin oluşturduğu matris olan B matrisinin tek çözümünün olması ile mümkündür. Bunun için ise bu B matrisini oluşturan P_j vektörlerinin lineer bağımsız olması veya eş değer olarak $\det(B) \neq 0$ olması ile mümkündür. Dolayısı ile X_B ile bu lineer bağımsız P_j 'lere karşılık gelen bilinmeyenlerin oluşturduğu vektör gösterilirse temel çözüm bu X_B vektörünün elemanları olacaktır. X vektörünün diğer elemanları temel dışı değişken olup değerleri sıfıra (0) eşit alınacaktır. Bu durumda temel çözüm

$$X_B = B^{-1}b$$

olarak elde edilir. Öte yandan $\det(B) \neq 0$ olduğundan B matrisi tersinir olup B^{-1} mevcuttur. Eğer $X_B = B^{-1}b \geq 0$ ise elde edilen çözüm uygun çözümdür.

Şimdi (4)'deki problemin Simpleks tablosunun matris formunda ifade edelim. Bunun için problemin bir maksimizasyon problemi olarak verildiğini varsayalım:

$$\begin{aligned}
\max z &= \sum_{j=1}^n c_j x_j \\
\sum_{j=1}^n P_j x_j &= b \\
x_j &\geq 0, j = 1, 2, \dots, n.
\end{aligned} \tag{5}$$

Başlangıç Simpleks tablosunu oluşturmak için $X = [X_I \ X_{II}]$, $C = [C_I \ C_{II}]$ olacak şekilde iki vektöre ayırılım. Dikkat edilirse eğer problem (3) şeklinde verilmiş ise $X_{II} = I$ matrisine karşılık gelecek olup bu vektörün elemanları temel uygun çözümler olacaktır. Diğer bir ifade ile (3) problemi için $B = I$ olup $X_{II} = X_B$ olup başlangıç temel uygun çözümdür. (5)'deki standart form matrisel olarak amaç fonksiyonu

$$z - \sum_{j=1}^n c_j x_j = 0 \Rightarrow z - [C_I \ C_{II}] \begin{bmatrix} X_I \\ X_{II} \end{bmatrix} = 0$$

şeklinde ifade edilebildiğinden

$$\begin{bmatrix} 1 & -C_I & -C_{II} \\ 0 & A & I \end{bmatrix} \begin{bmatrix} z \\ X_I \\ X_{II} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ b \end{bmatrix} \tag{6}$$

olarak yazılabilir. X_B ile herhangi bir Simpleks yinelemesinde temel değişkenleri gösteren vektör ve C_B ise bu temel değişkenlere karşılık gelen amaç fonksiyonunun katsayıları olmak üzere bir Simpleks yinelemesinde temel dışı değişkenlerin değerleri sıfır (0) seçileceğinden standart doğrusal programlama problemi

$$BX_B = b, z = C_B X_B$$

denklem sistemine indirgenir. Buradan temel değişken değerleri ve doğrusal programlama probleminin sonucu

$$\begin{aligned}
\begin{bmatrix} 1 & -C_B \\ 0 & B \end{bmatrix} \begin{bmatrix} z \\ X_B \end{bmatrix} &= \begin{bmatrix} 0 \\ b \end{bmatrix} \Rightarrow \begin{bmatrix} z \\ X_B \end{bmatrix} = \begin{bmatrix} 1 & -C_B \\ 0 & B \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ b \end{bmatrix} \\
\Rightarrow \begin{bmatrix} z \\ X_B \end{bmatrix} &= \begin{bmatrix} 1 & C_B B^{-1} \\ 0 & B^{-1} \end{bmatrix} \begin{bmatrix} 0 \\ b \end{bmatrix} = \begin{bmatrix} C_B B^{-1} b \\ B^{-1} b \end{bmatrix}
\end{aligned} \tag{7}$$

olarak elde edilir. Buradan sonraki yinelemenin başlangıç Simpleks tablosu (6)'daki son matrisin her iki tarafının (7)'deki matris ile çarpılması ile elde edilir:

$$\begin{aligned}
& \begin{bmatrix} 1 & C_B B^{-1} \\ 0 & B^{-1} \end{bmatrix} \begin{bmatrix} 1 & -C_I & -C_{II} \\ 0 & A & I \end{bmatrix} \begin{bmatrix} z \\ X_I \\ X_{II} \end{bmatrix} = \begin{bmatrix} 1 & C_B B^{-1} \\ 0 & B^{-1} \end{bmatrix} \begin{bmatrix} 0 \\ b \end{bmatrix} \\
& \Rightarrow \begin{bmatrix} 1 & C_B B^{-1} A - C_I & C_B B^{-1} - C_{II} \\ 0 & B^{-1} A & B^{-1} \end{bmatrix} \begin{bmatrix} z \\ X_I \\ X_{II} \end{bmatrix} = \begin{bmatrix} C_B B^{-1} b \\ B^{-1} b \end{bmatrix}. \quad (8)
\end{aligned}$$

Dolayısı ile genel Simpleks tablosu Tablo 40'taki gibi elde edilir.

Tablo 40. Simpleks tablosunun matris gösterimi

Temel değişkenler	X_I	X_{II}	Çözüm
z	$C_B B^{-1} A - C_I$	$C_B B^{-1} - C_{II}$	$C_B B^{-1} b$
X_B	$B^{-1} A$	B^{-1}	$B^{-1} b$

Tablodan yineleme adımındaki B matrisi önceden bilinirse Simpleks tablosunun kolaylıkla elde edilebileceği görülür. Çünkü tablodaki diğer matrisler orijinal problemde elde edilebilirler. Bu tablo tüm doğrusal programlama problemleri için temel teşkil eden bir algoritma olması nedeniyle önemlidir ve Simpleks algoritmalarındaki çözüm bir temel B 'den diğer temel çözüm olan B_{yeni} 'ye doğru ilerler. Bu durumda B_{yeni} , B 'deki mevcut çıkan değişken ile yeni giren değişkenin yer değiştirmesi ile elde edilir. Giren ve çıkan değişkenin belirlendiği optimumluk ve uygunluk koşulu matrisel ise aşağıdaki gibidir:

Simpleks optimumluk koşulu: Tablo 40'tan dikkat edilirse x_j değişkeni ile ilişkili olan z denkleminin katsayısı P_j , $[A \ I]$ matrisinin j . sütunu ve c_j, C vektörünün j . elemanı olmak üzere

$$z_j - c_j = C_B B^{-1} P_j - c_j \quad (9)$$

dir. Tüm x_j temel değişkenler için $z_j - c_j$ değerleri sıfırdır (0). Çünkü Gauss-Jordan algoritması ile temel değişkenlerin bulunduğu sütunlardaki diğer elemanlar sıfırlanırlar. Amaç satırı, Gauss-Jordan algoritması ile j temel dışı değişkenlerinin indisini göstermek üzere

$$z + \sum_{j \in TD} (z_j - c_j) x_j = C_B B^{-1} b \quad (10)$$

şeklinde elde edilir. Eğer bu satırda maksimizasyon problemi için negatif değere sahip bir x_j temel dışı değişken mevcut ise bu temel dışı değişkenin değerinin sıfırdan yukarı çıkması z 'nin değerini de $C_B B^{-1}b$ 'den yukarı çıkaracağından çözümü iyileştirecektir (Taha, 2010).

Simpleks uygunluk koşulu: Çıkan değişken, simpleks optimumluk koşulu ile giren değişkenin bulunduğu k . sütundaki oranlara bakılarak belirlenir. Tablo 40'daki simpleks algoritmasında pivot elemanın bulunduğu satır 1 yapıp onun bulunduğu sütundaki diğer tüm elemanlar sıfır yapılacağından bu yineleme sonucunda kısıtların bulunduğu satır elemanları

$$(X_B)_i + \sum_{j \in TD} (B^{-1}P_j)_i x_j = (B^{-1}b)_i$$

olarak elde edilir. Burada $(X_B)_i$ ve $(B^{-1}P_j)_i$ sırasıyla X_B ve $B^{-1}P_j$ sütun vektörlerinin i . elemanlarını göstermektedir. Dolayısı ile x_k ile simpleks optimumluk koşulu ile elde edilen giren değişkeni gösterirsek kısıt denkleminde k . sütundaki elemanlar son denklemden

$$(X_B)_i = (B^{-1}b)_i - (B^{-1}P_k)_i x_k$$

olarak elde edilecektir. Bu denklem, her i için $(B^{-1}P_k)_i > 0$ ve $(X_B)_i \geq 0$ olduğu durumda x_k değerindeki artışın $(X_B)_i$ 'nin negatif olmamasını sağlayacaktır. Buradan, x_k giren değişkeninin maksimum değeri

$$x_k = \min_i \left\{ \frac{(B^{-1}b)_i}{(B^{-1}P_k)_i} \mid (B^{-1}P_k)_i > 0 \right\} \quad (11)$$

olacağından çıkan değişken ifadeyi minimum yapan temel değişkendir (Taha, 2010).

Örnek 14.1 Örnek 9'daki lastik üreten firma için oluşturulan

$$\max z = 20x_1 + 16x_2 + 15x_3$$

$$1.5x_1 + x_2 + 2x_3 \leq 90$$

$$x_1 + 2x_2 + 2x_3 \leq 70$$

$$2x_1 + x_2 + x_3 \leq 80$$

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

problemini göz önüne alalım. Bu problemin standart formu

$$\max z = 20x_1 + 16x_2 + 15x_3$$

$$1.5x_1 + x_2 + 2x_3 + x_4 = 90$$

$$x_1 + 2x_2 + 2x_3 + x_5 = 70$$

$$2x_1 + x_2 + x_3 + x_6 = 80$$

$$x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0, x_5 \geq 0, x_6 \geq 0$$

dir. Burada yukarıdaki matris gösterimlerine uygunluk için dolgu değişkenleri x_4, x_5, x_6 olarak gösterilmiştir. Dolayısı ile standart form matrissel olarak

$$\max z = [20 \quad 16 \quad 15 \quad 0 \quad 0] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = [C_I \quad C_{II}] \begin{bmatrix} X_I \\ X_{II} \end{bmatrix} = CX$$

$$\begin{bmatrix} 1.5 & 1 & 2 & 1 & 0 & 0 \\ 1 & 2 & 2 & 0 & 1 & 0 \\ 2 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 90 \\ 70 \\ 80 \end{bmatrix} \Rightarrow [A \quad I]X = b$$

olarak ifade edilebilir. Burada $C_I = [20 \quad 16 \quad 15], C_{II} = [0 \quad 0 \quad 0]$, $X_I = [x_1 \quad x_2 \quad x_3]^T, X_{II} = [x_4 \quad x_5 \quad x_6]^T$ dir. Başlangıç X_B temel değişkenlere karşılık gelen $X_B = [x_4 \quad x_5 \quad x_6]^T$ olup buna karşılık gelen $B = I$ ve $C_B = [0 \quad 0 \quad 0]$ dir. Bu durumda mevcut çözüm (7) ile

$$\begin{bmatrix} z \\ X_B \end{bmatrix} = \begin{bmatrix} C_B B^{-1} b \\ B^{-1} b \end{bmatrix} = \begin{bmatrix} 0 \\ 90 \\ 70 \\ 80 \end{bmatrix}$$

olarak elde edilir. Bu Simpleks algoritmasındaki başlangıç uygun çözüme karşılık gelir. Bu durumda başlangıç Simpleks tablosu ise Tablo 40 ile Tablo 41 gibi elde edilir.

Tablo 41. Başlangıç Simpleks tablosu

Temel değişkenler	X_I	X_{II}	Çözüm
z	$C_B B^{-1} A - C_I = -C_I$ $= [-20 \quad -16 \quad -15]$	$C_B B^{-1} - C_{II}$ $= [0 \quad 0 \quad 0]$	$C_B B^{-1} b = 0$
$X_B = \begin{bmatrix} x_4 \\ x_5 \\ x_6 \end{bmatrix}$	$B^{-1} A = \begin{bmatrix} 1.5 & 1 & 2 \\ 1 & 2 & 2 \\ 2 & 1 & 1 \end{bmatrix}$	$B^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$B^{-1} b = \begin{bmatrix} 90 \\ 70 \\ 80 \end{bmatrix}$

Tablo 41, Örnek 9'daki başlangıç Simpleks tablosu ile uyumludur. Fakat optimum çözüm değildir. Gerçekten (9)'dan temel dışı değişkenler olan x_1, x_2, x_3 elemanları için $z_j - c_j$ değerleri

$$[z_1 - c_1, z_2 - c_2, z_3 - c_3] = [-20, -16, -15]$$

negatif elemanlar içermektedir. Dolayısı ile elde edilen çözüm optimum çözüm değildir. Bu durumda giren değişken olarak mutlak değerce en büyük değere sahip olan x_1 'in sisteme girmesi çözümün değerini arttıracaktır. Bu durumda çıkan değişken ise (10) ile $k = 1$ olduğundan (giren değişkenin indisi 1)

$$\begin{aligned} x_k &= \min_{i=1,2,3} \left\{ \frac{(B^{-1}b)_i}{(B^{-1}P_1)_i} \mid (B^{-1}P_1)_i > 0 \right\} = \min_{i=1,2,3} \left\{ \left[\frac{(B^{-1}b)_1}{(B^{-1}P_1)_1}, \frac{(B^{-1}b)_2}{(B^{-1}P_1)_2}, \frac{(B^{-1}b)_3}{(B^{-1}P_1)_3} \right] \mid (B^{-1}P_1)_i > 0 \right\} \\ &= \min_{i=1,2,3} \left\{ \left[\frac{90}{1.5}, \frac{70}{1}, \frac{80}{2} \right] \mid (B^{-1}P_1)_i > 0 \right\} = \min_{i=1,2,3} \left\{ \left[\frac{60}{70} \right] \mid (B^{-1}P_1)_i > 0 \right\} \\ &= 40 \end{aligned}$$

olduğundan çıkan değişken x_6 'tür. Bu durumda yeni B matrisi eski B matrisinde P_6 sütunu çıkarılarak yerine

$$P_1 = \begin{bmatrix} 1.5 \\ 1 \\ 2 \end{bmatrix}$$

sütununun eklenmesi ile

$$B_{yeni} = \begin{bmatrix} 1 & 0 & 1.5 \\ 0 & 1 & 1 \\ 0 & 0 & 2 \end{bmatrix}$$

olarak elde edilir. Ayrıca eski temel değişkenler X_B ve C_B vektörlerinde de değiştirilirse $X_B = [x_4 \ x_5 \ x_1]^T$ ve $C_B = [0 \ 0 \ 20]$ olarak bulunur. Bu durumda birinci yineleme sonucunda elde edilen Simpleks tablosu Tablo 42'deki gibidir.

Tablo 42. Birinci yineleme sonucunda elde edilen Simpleks tablosu

Temel değişkenler	X_I	X_{II}	Çözüm
z	$C_B B^{-1} A - C_I$ $= [0 \quad -6 \quad -5]$	$C_B B^{-1} - C_{II}$ $= [0 \quad 0 \quad 10]$	$C_B B^{-1} b$ $= 800$
$X_B = \begin{bmatrix} x_4 \\ x_5 \\ x_1 \end{bmatrix}$	$B^{-1} A = \begin{bmatrix} 0 & 0.25 & 1.25 \\ 0 & 1.5 & 1.5 \\ 1 & 0.5 & 0.5 \end{bmatrix}$	$B^{-1} = \begin{bmatrix} 1 & 0 & -0.75 \\ 0 & 1 & -0.5 \\ 0 & 0 & 0.5 \end{bmatrix}$	$B^{-1} b$ $= \begin{bmatrix} 30 \\ 30 \\ 40 \end{bmatrix}$

Matrisel olarak elde edilen Tablo 42'deki birinci yineleme sonucunda elde edilen tablonun, örnek 9'daki Tablo 16 ile uyumlu olduğu görülür. Bu durumda elde edilen çözümün optimum çözüp olup olmadığına bakılırsa x_2, x_3, x_6 temel dışı değişkenleri için $z_j - c_j$ değerleri

$$\begin{aligned} [z_2 - c_2, z_3 - c_3, z_6 - c_6] &= C_B B^{-1} [P_2 \quad P_3 \quad P_6] - [c_2 \quad c_3 \quad c_6] \\ &= [-6 \quad -5 \quad 10] \end{aligned}$$

olduğundan çözüm optimum değildir. Bu durumda probleme giren değişken en büyük negatif değere sahip olan x_2 'dir. Çıkan değişken ise

$$x_k = \min_{i=1,2,3} \left\{ \frac{(B^{-1}b)_i}{(B^{-1}P_2)_i} \mid (B^{-1}P_2)_i > 0 \right\} = \min_{i=1,2,3} \left\{ \frac{\begin{bmatrix} 120 \\ 20 \\ 80 \end{bmatrix}}{\begin{bmatrix} 120 \\ 20 \\ 80 \end{bmatrix}} \mid (B^{-1}P_1)_i > 0 \right\} = 20$$

olduğundan X_B 'de ikinci elemana karşılık gelen x_5 'tir. Bu durumda B_{yeni} önceki B matrisinde P_5 'nin yerine P_2 'nin getirilmesi ile

$$B_{yeni} = \begin{bmatrix} 1 & 1 & 1.5 \\ 0 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

olarak elde edilir. Benzer şekilde X_B ve C_B 'lerde çıkan değişkenler yerine giren değişkenler eklenirse $X_B = [x_4 \quad x_2 \quad x_1]^T$ ve $C_B = [0 \quad 16 \quad 20]$ olarak bulunur. Bu durumda yineleme sonucunda elde edilen Simpleks tablosu Tablo 43'teki gibidir.

Tablo 43. İkinci yineleme sonucunda elde edilen Simpleks tablosu

Temel değişkenler	X_I	X_{II}	Çözüm
z	$C_B B^{-1} A - C_I$ $= [0 \ 0 \ 1]$	$C_B B^{-1} - C_{II} = [0 \ 4 \ 8]$	$C_B B^{-1} b$ $= 920$
$X_B = \begin{bmatrix} x_4 \\ x_2 \\ x_1 \end{bmatrix}$	$B^{-1} A = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$	$B^{-1} = \begin{bmatrix} 1 & -1/6 & -2/3 \\ 0 & 2/3 & -1/3 \\ 0 & -1/3 & 2/3 \end{bmatrix}$	$B^{-1} b = \begin{bmatrix} 25 \\ 20 \\ 30 \end{bmatrix}$

İkinci yineleme sonucunda matrissel yolla elde edilen Simpleks tablosu Örnek 9'daki Tablo 19 ile uyumludur. Bu durumda elde edilen sonucun optimum çözüm olup olmadığı kontrol edilirse

$$[z_3 - c_3, z_5 - c_5, z_6 - c_6] = C_B B^{-1} [P_3 \ P_4 \ P_6] - [c_3 \ c_4 \ c_6] = [1 \ 4 \ 5]$$

olup negatif eleman bulunmadığından elde edilen çözüm optimumdur. Bu durumda temel değişken değeri $X_B = [x_4 \ x_2 \ x_1]^T = [25 \ 20 \ 30]^T$ olup optimum çözüm 920'dir. Bu sonuç Örnek 9 ile aynıdır.

1.6.1. Dual Problemin Matris Tanımı

Primal bir problem (4)'deki yapıya benzer olarak

$$\begin{aligned} \max z &= [C_I \ C_{II}] X \\ [A \ I] X &= b \\ X &\geq 0 \end{aligned} \tag{12}$$

olarak tanımlansın. Bu durumda problemin dual yapısı Tablo 35 ile $Y = [y_1 \ y_2 \ \dots \ y_m]$ dual değişkenler vektörü olmak üzere

$$\begin{aligned} \max w &= Yb \\ YA &\geq C_I \\ Y &\geq C_{II} \end{aligned} \tag{13}$$

Y : sınırlandırılmamış

şeklinde ifade edilebilir (Taha, 2010). Bu takdirde dual problemin çözümü için aşağıdaki teorem elde edilir.

Teorem. (Taha, 2010) Primal problemin optimum değeri B , bu matris ile ilgili amaç fonksiyonunun katsayılar vektörü ise C_B ise bu takdirde dual problemin optimum çözümü

$$Y = C_B B^{-1}$$

dir.

Optimum Primal ve Dual Çözümler arasındaki İlişki

Primal ie dual problem arasındaki ilişkiyi veren aşağıdaki birinci özelliği verelim:

I. Özellik. (Taha, 2010) Primal veya dual problemlerin herhangi bir Simpleks yinelemesinde

$$\begin{pmatrix} \text{Bir problemdeki} \\ j \text{ değişkeninin} \\ \text{amaç (z) satırı} \end{pmatrix} = \begin{pmatrix} \text{Öteki problemdeki} \\ j \text{ kısıtının} \\ \text{sol tarafı eksi sağ tarafı} \end{pmatrix} \text{dır.}$$

Bu özellik, primal ve dual problemlerin her ikisine göre simetriktir.

I. özellik ile, bir problemin optimum çözümünün diğer problemin optimum simpleks tablosu yardımıyla elde edilebileceğini gösterilir. Dolayısı ile bu özellik, bir problemde optimum çözüm için fazla işlem gereken durumlarda diğer yöntemin seçilmesiyle hesap kolaylığı sağlanabileceği anlamına gelir. Örneğin, kısıt sayısı değişken sayısından çok fazla olan primal problem yerine dual problem alınırsa kısıt sayısı azalacağından işlem sayısının da azalması muhtemeldir. Bu durumda dual problemi çözmek daha avantajlı olabilir.

Şimdi I. özelliği bir örnek ile açıklayalım:

Örnek 15. Örnek 13'teki dual olarak çözülen problemi ele alalım. Fakat dual problemdeki gösterimde dual problemin değişkenleri genel olarak y_i 'ler ile gösterildiğinden problemdeki x_i 'ler yerine y_i 'leri kullanalım. Bu takdirde dual problem

$$\begin{aligned} \min z &= 2y_1 + 3y_2 \\ 3y_1 + 6y_2 &\geq 1500 \\ 4y_1 + 2y_2 &\geq 800 \\ y_1 &\geq 0, y_2 &\geq 0 \end{aligned}$$

için dual Simpleks algoritmasını uygulayalım. Bu probleme karşılık gelen primal problem ise

$$\begin{aligned} \max z &= 1500x_1 + 800x_2 \\ 3x_1 + 4x_2 &\leq 2 \end{aligned}$$

$$6x_1 + 2x_2 \leq 3$$

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

şeklindedir. Bu durumda dual Simpleks tablosu ve primal Simpleks tablosu Tablo 44'teki gibi elde edilir:

Tablo 44. Örnek 15 için dual ve primal Simpleks tabloları:

Temel değiş.	y_1	y_2	y_3	y_4	Çözüm	Temel değiş.	x_1	x_2	x_3	x_4	Çözüm
z	-2	-3	0	0	0	z	-1500	-800	0	0	0
y_3	-3	-6	1	0	-1500	x_3	3	4	1	0	2
y_4	-4	-2	0	1	-800	x_4	6	2	0	1	3

Tablo 45. 1. yineleme sonucundaki tablo

Temel değ.	y_1	y_2	y_3	y_4	Çözüm	Temel değ.	x_1	x_2	x_3	x_4	Çöz.
z	-1/2	0	-1/2	0	750	z	0	-300	0	250	750
y_2	1/2	1	-1/6	0	250	x_3	0	3	1	-1/2	1/2
y_4	-3	0	-1/3	1	-300	x_1	1	1/3	0	1/6	1/2

Tablo 46. 2. yineleme sonucunda oluşan tablo:

Temel değ.	y_1	y_2	y_3	y_4	Çöz.	Temel değ.	x_1	x_2	x_3	x_4	Çöz.
z	0	0	-4/9	-1/6	800	z	0	0	100	200	800
y_2	0	1	-2/9	1/6	200	x_2	0	1	1/3	-1/6	1/6
y_1	1	0	1/9	-1/3	100	x_1	1	0	-1/9	2/9	4/9

Tablo 44'ten dikkat edilirse 1. yineleme sonucunda dual problemin z satırında $y_1 = -1/2, y_3 = -1/2$ 'dir. Primal problem için I. özelliğe baktığımızda ilgili değişkenler x_1 ve x_3 olup $0 \leq x_1, 0 \leq x_3$ ve eşitsizliklerin sağ tarafları ise 1/2 olduğundan

$$\begin{pmatrix} \square \\ j \text{ kısıtının} \\ \text{sol tarafı eksi sağ tarafı} \end{pmatrix} = \begin{pmatrix} 0 - \frac{1}{2} \\ 1 \\ 0 - \frac{1}{2} \end{pmatrix} = \begin{pmatrix} -\frac{1}{2} \\ 1 \\ -\frac{1}{2} \end{pmatrix}$$

dir. Dolayısı ile I. özellik sağlanır. Benzer şekilde 2. yineleme sonucunda primal denklemin z satırında $y_3 = -4/9, y_4 = -1/6$ 'dır. Dual problemde ise $0 \leq x_1, 0 \leq$

x_2 olup eşitsizliklerin sağ tarafı ise çözüm sütunundan x_1 için $4/9$, x_2 için $1/6$ olduğu görülür. Bu durumda

$$\begin{pmatrix} j \text{ kısıtının} \\ \text{sol tarafı eksi sağ tarafı} \end{pmatrix} = \begin{pmatrix} 0 - 4/9 \\ 0 - 1/6 \end{pmatrix} = \begin{pmatrix} -\frac{4}{9} \\ -\frac{1}{6} \end{pmatrix}$$

olup I. özellik sağlanmaktadır. Öte yandan dikkat edilirse son Simpleks tablosunda primal Simpleks algoritmasında x_1 satırının birim matrise karşılık gelen kısım hariç diğer kısmı

	x_3	x_4
x_1	$-1/9$	$2/9$

iken dual Simpleks algoritmasında y_3 'e karşılık gelen sütun

	y_3
y_1	$1/9$
y_2	$-2/9$

dur. Görüldüğü gibi bu değerler birbirinin zıt işaretlisidirler. Benzer sonuç diğer değişkenler olan x_2 ve y_4 için de geçerlidir. Son Simpleks tablosundaki primal problemin x_2 satırının birim matrise karşılık gelen kısım hariç diğer kısmı

	x_3	x_4
x_2	$1/3$	$-1/6$

iken dual Simpleks algoritmasında y_4 'e karşılık gelen sütun

	y_4
y_1	$-1/3$
y_2	$1/6$

dır. Ayrıca son Simpleks algoritmasındaki primal problemin z satırı

	x_3	x_4
z	100	200

dual problemin çözüm sütunu olan

	y_1	y_2
	100	200

karşılık gelirken primal problemin çözüm sütunu ise dual problemin z satırının eksi ile çarpımına karşılık gelmektedir. İki problemin çözümünün de aynı sonucu verdiği son tablodan açıktır.

Örnek 15'ten de görüldüğü üzere dual Simpleks algoritmasının son tablosu ile primal Simpleks algoritmasının son tablosu birbiri ile yakın ilişkilidir. Yani, birinin son Simpleks tablosundan diğeri rahatlıkla elde edilebilir. Bundan dolayı problemi kısıt sayısına göre hangi yöntem ile daha kolay çözülebilirse o yöntemi kullanmak zaman ve işlem yükü bakımından avantaj sağlar.

Öte yandan I. özellikten aşağıdaki sonucu elde edebiliriz.

Sonuç (Taha, 2010). Herhangi bir primal yinelemede

$$x'_j \text{nin amaç fonksiyonu katsayısı} = \sum_{i=1}^m a_{ij}y_i - c_j = YP_j - c_j$$

dir.

Primal ve dual arasındaki II. Özellik ise aşağıdaki gibi verilir:

II. özellik. (Taha, 2010) Herhangi bir uygun primal ve dual çifti için

$$\left(\begin{array}{c} \text{Maksimizasyon} \\ \text{problemindeki} \\ \text{amaç değeri} \end{array} \right) \leq \left(\begin{array}{c} \text{Minimizasyon} \\ \text{problemindeki} \\ \text{amaç değeri} \end{array} \right)$$

olup optimum çözüm birbirine eşittir.

1.7. Optimum Sonrası Analiz veya Duyarlılık Analizi

Duyarlılık analizindeki en önemli nokta modelin katsayılarındaki değişimlerin mevcut çözümü (optimumu ve uygunluğu) değiştirip değiştirmediğini belirlemektir. Eğer katsayılardaki değişim mevcut çözüm de değişime neden oluyorsa (eğer varsa) yeni optimum çözümün nasıl belirleneceği sorusu ile karşılaşılır. Bunun için problemi yeniden çözmek de bir yoldur ancak kısıt sayısı ve değişkeni fazla olan problemler için bu probleme büyük bir işlem yükü eklenmesine neden olur. Dolayısı ile orjinal modelin katsayılarındaki değişimlerin Simpleks hesaplamalarını nasıl etkileyeceğinin bilinmesi, istenilen sonuçların verimli bir şekilde elde edilmesi için oldukça önemlidir. Özellikle de amaç satırının katsayılarındaki değişim (optimumluk) ve kısıtlamaların sağ tarafındaki değişimlerin

(uygunluk) Simpleks hesaplanmalarını etkisinin bilinmesi büyük önem taşır (Taha, 2010).

Genel olarak modeldeki değişimlerin aşağıdaki sonuçları ortaya çıkarabileceği söylenebilir:

1. Mevcut çözüm değişmeden kalır.
2. Mevcut çözüm uygun olmayan bir hale gelir.
3. Mevcut çözüm optimum olmayan bir hale gelir.
4. Mevcut çözüm hem uygun olmayan hem de optimum olmayan bir hale gelir (Taha, 2010).

İkinci durumda uygunluğu sağlamak için dual simpleks yöntemi kullanılırken üçüncü durumdaki yeni optimum çözüme ulaşmak için ise primal simpleks yöntemi kullanılır. Dördüncü durum ikinci ve üçüncü durumun birleşimi olduğundan hem dual hem de primal simpleks yöntemleri kullanılır (Taha, 2010).

Şimdi herhangi bir katsayıdaki değişimin problem üzerindeki etkisini gözlemlemek için Örnek 13'teki lastik üretimi yapan firma modelini göz önüne alalım.

Örnek 16. Örnek 9'daki lastik üreten firma için oluşturulan

$$\begin{aligned}\max z &= 20x_1 + 16x_2 + 15x_3 \\ 1.5x_1 + x_2 + 2x_3 &\leq 90 \\ x_1 + 2x_2 + 2x_3 &\leq 70 \\ 2x_1 + x_2 + x_3 &\leq 80 \\ x_1 \geq 0, x_2 \geq 0, x_3 &\geq 0\end{aligned}$$

probleminin dual formu

$$\begin{aligned}\min z &= 90y_1 + 70y_2 + 80y_3 \\ 1.5y_1 + y_2 + 2y_3 &\geq 20 \\ y_1 + 2y_2 + y_3 &\geq 16 \\ 2y_1 + 2y_2 + y_3 &\geq 15 \\ y_1 \geq 0, y_2 \geq 0, y_3 &\geq 0\end{aligned}$$

dir. Bu durumda primal problem için optimum Simpleks tablosu aşağıdaki gibi elde edilmişti.

Tablo 47. Örnek 9 için optimum çözümün bulunduğu Simpleks tablosu

Temel değişkenler	x_1	x_2	x_3	x_4	x_5	x_6	Çözüm
z	0	0	1	0	4	8	920
x_4	0	0	1	1	$-1/6$	$-2/3$	25
x_2	0	1	1	0	$2/3$	$-1/3$	20
x_1	1	0	0	0	$-1/3$	$2/3$	30

1.7.1.Uygunluğu Etkileyen Değişiklikler

Optimum çözümün uygunluğu, problemde verilen kısıtlamalardaki b vektörünün değişmesi veya probleme yeni bir kısıt eklenmesi halinde değişebilir. Her iki durum için de eğer $B^{-1}b$ nin elemanlarından en az bir tanesi negatif hale gelirse uygunsuzluk ortaya çıkar. Çünkü bu durumda mevcut temel değişkenlerden birinin değeri negatif hale geleceğinden değişkenlerinin pozitif olması koşulu sağlanmamış olur.

Uygunluğu etkileyen koşulları örnek 14 üzerinden hem b vektörünün değişmesi durumuna göre hem de yeni bir kısıt eklenmesi durumuna göre ayrı ayrı gözlemleyelim:

1.7.1.1. b vektörünün değişmesi durumu:

Örnek 14.1. Eğer lastiklerin imalatında her bir bölümün işlem yapabilme kapasitesi %20 arttırılırsa problem

$$\max z = 20x_1 + 16x_2 + 15x_3$$

$$1.5x_1 + x_2 + 2x_3 \leq 108$$

$$x_1 + 2x_2 + 2x_3 \leq 84$$

$$2x_1 + x_2 + x_3 \leq 96$$

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

olarak bulunur. Bu durumda tabloya yansıyacak tek değişiklik kısıtların sağ tarafında meydana gelecektir. Bu durumda mevcut temel değişken değişmezler fakat değerleri Tablo 40'tan $X_B = B^{-1}b$ olduğundan Tablo 45 ile bu çözüm

$$X_B = \begin{bmatrix} x_4 \\ x_2 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 & -1/6 & -2/3 \\ 0 & 2/3 & -1/3 \\ 0 & -1/3 & 2/3 \end{bmatrix} \begin{bmatrix} 108 \\ 84 \\ 96 \end{bmatrix} = \begin{bmatrix} 30 \\ 24 \\ 36 \end{bmatrix}$$

dır. Yani, $x_1 = 36, x_2 = 24, x_4 = 30$ olup optimum çözüm ise 1104 olarak elde edilir. Bu çözüm yine optimumluğunu korumaktadır.

Öte yandan her bir fabrikanın çalışma kapasitesini arttırmak yerine 2. bölümün işlem kapasitesi 34'e düşürülür diğerleri sabit bırakılırsa problem

$$\max z = 20x_1 + 16x_2 + 15x_3$$

$$1.5x_1 + x_2 + 2x_3 \leq 90$$

$$x_1 + 2x_2 + 2x_3 \leq 34$$

$$2x_1 + x_2 + x_3 \leq 80$$

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

olup sonuç

$$X_B = \begin{bmatrix} 1 & -1/6 & -2/3 \\ 0 & 2/3 & -1/3 \\ 0 & -1/3 & 2/3 \end{bmatrix} \begin{bmatrix} 90 \\ 34 \\ 80 \end{bmatrix} = \begin{bmatrix} 31 \\ -4 \\ 42 \end{bmatrix}$$

olarak elde edilir. Fakat x_2 negatif olduğundan bu çözüm uygun değildir. Bu durumda uygun çözümü elde etmek için probleme dual Simpleks yöntemi uygulanır. Bunun için Tablo 47'in sol tarafı tekrar düzenlenirse çözüm sütunun z satırı $20 \cdot 42 + 16 \cdot (-4) + 15 \cdot 0 = 776$ olup Tablo 48 elde edilir.

Tablo 48. Örnek 14.1 için optimum çözümün bulunduğu Simpleks tablosu

Temel değ.	x_1	x_2	x_3	x_4	x_5	x_6	Çözüm
z	0	0	1	0	4	8	776
x_4	0	0	1	1	$-1/6$	$-2/3$	31
x_2	0	1	1	0	$2/3$	$-1/3$	-4
x_1	1	0	0	0	$-1/3$	$2/3$	42

Bu durumda dual Simpleks uygulanırsa çıkan değişken x_2 olurken giren değişken ise tek negatif ifade x_6 sütununda olduğundan x_6 'dır. Bu durumda pivot eleman 1 yapılır, Gauss-Jordan eliminasyonu uygulanırsa Tablo 49 elde edilir.

Tablo 49. Örnek 14.1 için Dual Simpleks algoritması sonucu

Temel değ.	x_1	x_2	x_3	x_4	x_5	x_6	Çözüm
z	0	24	25	0	20	0	680
x_4	0	-2	-1	1	$-3/2$	0	39
x_6	0	-3	-3	0	-2	1	12
x_1	1	2	2	0	1	0	34

Bu durumda temel deęişken deęerleri $x_1 = 34, x_4 = 39, x_6 = 12$ olup optimum çözüm ise 680 olarak elde edilir. Bu deęer hem kârı azalttığı, hem de sadece bir lastik türü üretildiği için fabrika adına doğru bir yaklaşım olmayacaktır.

1.7.1.2. b 'nin elemanlarının uygunluk aralığı:

Kısıtlamaların sağ taraf deęişimlerinin incelenmesinin bir başka yolu ise mevcut çözümün uygun kalma aralığının belirlenmesidir. Bunu önceki örnek üzerinden inceleyelim.

Örnek 14.2. Eğer lastiklerin imalatlarında sadece 1. fabrika d_1 saat daha fazla çalışabildiğini varsayalım. Bu durumda matematiksel model

$$\max z = 20x_1 + 16x_2 + 15x_3$$

$$1.5x_1 + x_2 + 2x_3 \leq 90 + d_1$$

$$x_1 + 2x_2 + 2x_3 \leq 70$$

$$2x_1 + x_2 + x_3 \leq 80$$

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

olarak elde edilir. Bu durumda Örnek 14.1'den optimum çözüm

$$X_B = \begin{bmatrix} x_4 \\ x_2 \\ x_1 \end{bmatrix} = B^{-1}b = \begin{bmatrix} 1 & -1/6 & -2/3 \\ 0 & 2/3 & -1/3 \\ 0 & -1/3 & 2/3 \end{bmatrix} \begin{bmatrix} 90 + d_1 \\ 70 \\ 80 \end{bmatrix} = \begin{bmatrix} 25 + d_1 \\ 20 \\ 30 \end{bmatrix}$$

olduğunda sağlanır. Bu takdirde deęişkenlerin uygunluğu ile her bir deęişkenin deęeri pozitif olması gerektiğinden $25 + d_1 \geq 0 \Rightarrow d_1 \geq -25$ olduğu sürece uygunluk korunacaktır. Amaç fonksiyonunun deęeri ise $x_4 = 25 + d_1, x_2 = 20, x_1 = 30$ olacağından

$$z = 20 \cdot 30 + 16 \cdot 20 = 920$$

değişmeden kalacaktır.

Benzer düşünce sadece 2. fabrikanın iş yükünün deęiştirilmesi ile gözlemlenebilir. Yani, ikinci fabrikanın iş yükü d_2 kadar arttırılırsa çözümün optimum olduğu temel deęişken deęerleri

$$X_B = B^{-1}b = \begin{bmatrix} 1 & -1/6 & -2/3 \\ 0 & 2/3 & -1/3 \\ 0 & -1/3 & 2/3 \end{bmatrix} \begin{bmatrix} 90 \\ 70 + d_2 \\ 80 \end{bmatrix} = \begin{bmatrix} \frac{150 - d_2}{6} \\ \frac{2d_2 + 60}{3} \\ \frac{90 - d_2}{3} \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

olup uygunluğun bozulmaması için $-30 \leq d_2 \leq 90$ sağlanmalıdır. Bu ise ikinci fabrikanın işlem süresinin

$$\begin{aligned} -30 \leq d_2 \leq 90 &\Rightarrow 40 \leq 70 + d_2 \leq 160 \\ &\Rightarrow 40 \leq 2. \text{fabrikanın işlem süresi} \leq 160 \end{aligned}$$

arasında olması ile mümkündür. Örnek 14.1'den bu değer 34 olduğunda uygunluğun bozulduğu gözlemlenmişti.

Benzer düşünce sadece 3. fabrikanın iş yükünün değiştirilmesi ile gözlemlenebilir. Bu durumda

$$\begin{aligned} X_B = B^{-1}b &= \begin{bmatrix} 1 & -\frac{1}{6} & -\frac{2}{3} \\ 0 & \frac{2}{3} & -\frac{1}{3} \\ 0 & -\frac{1}{3} & \frac{2}{3} \end{bmatrix} \begin{bmatrix} 90 \\ 70 \\ 80 + d_3 \end{bmatrix} = \begin{bmatrix} \frac{75 - 2d_3}{3} \\ \frac{60 - d_3}{3} \\ \frac{2d_3 + 90}{3} \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \\ &\Rightarrow -45 \leq d_3 \leq \frac{75}{2} \end{aligned}$$

olarak elde edilir. Buradan üçüncü fabrikanın işlem süresi 35 ile $235/2$ arasında olduğu sürece çözümün uygunluğunun bozulmayacağı gözlemlenir.

Öte yandan not edelim ki fabrikaların işlem sürelerinin her bir fabrika için ayrı ayrı gözlemleneceği gibi hepsinin birlikte arttırımı veya iki fabrikanın birlikte arttırılması da gözlemlenebilir. Bu durumda arttırım yapılan işlem yüklerine $d_i, i = 1,2,3$ değişkenlerinin eklenmesi gerekir.

1.7.1.3. Yeni kısıtın eklenmesi

Mevcut kısıtlara yeni kısıtların eklenmesi aşağıdaki iki durumundan birinin oluşmasına neden olur.

- Yeni kısıt mevcut optimum çözüm tarafından sağlanır.
- Yeni kısıt uygunluğu bozabilir. Bu durumda yeniden uygun çözüm elde etmek için probleme dual simpleks yöntemi uygulanır.

Bu durumları bir örnek üzerinden inceleyelim.

Örnek 14.3. Lastik üreten fabrika yazlık lastik isteğinin artması ile yazlık lastikler için ikinci bir üretim yapmayı planlamaktadır. Bu lastikler için iş gücü yükü ve elde etmesi düşünülen kâr aşağıdaki gibidir.

1. bölüm	2. bölüm	3. bölüm	Toplam Zaman
1	2	1	75

Bu durumda matematiksel model

$$\begin{aligned}
\max z &= 20x_1 + 16x_2 + 15x_3 \\
1.5x_1 + x_2 + 2x_3 &\leq 90 \\
x_1 + 2x_2 + 2x_3 &\leq 70 \\
2x_1 + x_2 + x_3 &\leq 80 \\
x_1 + 2x_2 + x_3 &\leq 75 \\
x_1 \geq 0, x_2 \geq 0, x_3 &\geq 0
\end{aligned}$$

olup temel değişken değerleri $x_1 = 30, x_2 = 20, x_3 = 25$ için yeni kısıt $30 + 40 \leq 75$ sağlandığından mevcut optimum çözüm değişmeden kalır.

Örnek 14.4. Örnek 14.3'teki iş gücü yükü ve elde etmesi düşünülen kâr Örnek 14.3'deki yerine aşağıdaki gibi belirlenmiş olsun.

1. bölüm	2. bölüm	3. bölüm	Toplam Zaman
2	2	1	75

Bu durumda probleme eklenen kısıt $2x_1 + 2x_2 + x_3 \leq 75$ olup mevcut optimum çözüm bu koşulu sağlamaz:

$$x_1 = 30, x_2 = 20, x_3 = 25 \Rightarrow 60 + 40 \geq 75.$$

Dolayısı ile (b)'de belirtildiği üzere probleme dual Simpleks yöntemi uygulanır. Bunun için mevcut optimum tabloya yeni koşul Tablo 50'deki gibi eklenir.

Tablo 50. Yeni kısıt eklenmesi ile elde edilen tablo

Temel değ.	x_1	x_2	x_3	x_4	x_5	x_6	x_7	Çözüm
z	0	0	1	0	4	8	0	920
x_4	0	0	1	1	$-1/6$	$-2/3$	0	25
x_2	0	1	1	0	$2/3$	$-1/3$	0	20
x_1	1	0	0	0	$-1/3$	$2/3$	0	30
x_7	2	2	1	0	0	0	1	75

x_1 ve x_2 temel değişkenler olduklarından bunların altındaki ve üstündeki sayılar 0 olmalıdır. Yeni x_7 satırı bu durumu bozduğundan ilk işlem olarak bu temel değişkenlere karşılık gelen sütunların altındaki x_7 satırındaki sayılar sıfırlanır:

$$\text{Yeni } x_7 \text{ satırı} = \text{Eksi } x_7 \text{ satırı} - (2 \times x_2 \text{ satırı} + 2 \times x_1 \text{ satırı})$$

Bu durumda Simpleks tablosu Tablo 50'deki gibi elde edilir:

Tablo 51. Yeni kısıt eklenmesi sonucu temel değişkenler için yapılan işlem tablosu

Temel değ.	x_1	x_2	x_3	x_4	x_5	x_6	x_7	Çözüm
z	0	0	1	0	4	8	1	920
x_4	0	0	1	1	$-1/6$	$-2/3$	0	25
x_2	0	1	1	0	$2/3$	$-1/3$	0	20
x_1	1	0	0	0	$-1/3$	$2/3$	0	30
x_7	0	0	-1	0	$-2/3$	$-2/3$	1	-25

Şimdi Tablo 51'e dual Simpleks yöntemi uygulanabilir. Sadece x_7 satırının çözüm sütunu negatif olduğundan çıkan değişkenin x_7 olduğu giren değişkenin ise oranlara bakıldığında x_3 olduğu görülür. Bu durumda pivot eleman 1 yapıp Gauss Jordan yöntemi uygulanırsa Tablo 52 elde edilir.

Tablo 52. Birinci dual Simpleks algoritması uygulandığında elde edilen tablo

Temel değ.	x_1	x_2	x_3	x_4	x_5	x_6	x_7	Çözüm
z	0	0	0	0	$10/3$	$22/3$	0	895
x_4	0	0	0	1	$-5/6$	$-4/3$	1	0
x_2	0	1	0	0	0	-1	0	-5
x_1	1	0	0	0	$-1/3$	$2/3$	0	30
x_3	0	0	1	0	$2/3$	$2/3$	-1	25

Tablo 52'de çözüm sütununda negatif eleman kaldığı için tekrar dual simpleks yöntemi uygulanırsa, çıkan değişken sadece x_2 satırının çözüm sütununda negatif eleman olduğu için x_2 olup giren değişken ise sadece x_6 sütununda negatif eleman olduğu için x_6 'dır. Bu durumda pivot eleman 1 yapıp tekrar Gauss Jordan yöntemi uygulanırsa Tablo 53 elde edilir.

Tablo 53. İkinci dual Simpleks algoritması uygulandığında elde edilen tablo

Temel değ.	x_1	x_2	x_3	x_4	x_5	x_6	x_7	Çözüm
z	0	$22/3$	0	0	$10/3$	0	0	$2575/3$
x_4	0	$-4/3$	0	1	$-5/6$	0	1	$20/3$
x_6	0	1	0	0	0	1	0	5
x_1	1	$2/3$	0	0	$-1/3$	0	0	$80/3$
x_3	0	$2/3$	1	0	$2/3$	0	-1	$65/3$

Tablo 53'te çözüm sütununda negatif eleman bulunmadığından dual simpleks yöntemi sona erdirilir. Bu durumda temel değişken değerleri $x_1 = 80/3$, $x_3 = 65/3$, $x_4 = 20/3$, $x_6 = 5$ olup optimum çözüm ise 2575/3 olarak elde edilir.

1.7.2. Optimumluğu Etkileyen Değişiklikler

Amaç fonksiyonunun katsayıları olan $z_j - c_j$ 'ler optimumluk koşullarından uzaklaşması halinde problemin mevcut çözümünün optimumluğu değişecektir. Sonuçta amaç fonksiyonunun katsayıları

$$z_j - c_j = Y P_j - c_j$$

olduğundan çözümün optimumluğu, ya amaç fonksiyonunun katsayıları olan c_j 'lerin değişmesinden (dolayısı ile de C_B 'nin değişmesinden) ya da P_j vektörünün değişiminden etkilenecektir. Bu iki durumu ayrı ayrı inceleyelim.

1.7.2.1. c_j amaç katsayılarındaki değişiklikler

Temel değişkenlerin c_j 'lerinde değişiklik yapılması durumunda optimum çözümün elde edildiği simpleks tablosunun $z_j - c_j$ satırında bu elemanların değerleri sıfıra (0) eşit olduğundan optimum çözüm üzerinde herhangi bir etkiye neden olmazlar. Fakat temel dışı değişkenlerin katsayılarındaki değişiklikler optimum çözüm üzerinde bir etkiye neden olabileceklerinden temel dışı değişkenlerin $z_j - c_j$ 'lerinin yeniden hesaplanması gerekir. Bu durumda aşağıdaki prosedür uygulanır:

1. Değişim olması halinde dual fiyat vektörü olan $Y = C_B B^{-1}$ yeni C_B vektörü için tekrar hesaplanır.

2. Temel değişken olmayan mevcut x_j 'lerin tümü için $z_j - c_j = Y P_j - c_j$ hesaplanır.

Bu hesaplamalar sonucunda iki durum ortaya çıkar:

a. Optimum çözüm sağlanmışsa mevcut optimum çözüm değişmez fakat amaç fonksiyonu yeni bir optimum değer alır.

b. Optimumluk koşulu sağlanmamışsa, optimumluğu sağlamak için simpleks yöntemi uygulanır. (Taha, 2010)

Bu durumları bir örnek üzerinden ele alalım.

Örnek 15.1. Lastik üreten fabrikada son yıllarda mevsimlik lastiklerin daha popüler olması nedeniyle bu lastiklerden üretiminden elde edilen kâr'ın 15'den 16'ya

çıktığını kabul edelim. Bu durumda X_B 'deki temel değişkenler sırası ile x_4, x_2, x_1 olduğundan bu sıra ile $C_B = [0 \ 16 \ 20]$ olduğundan

$$Y = C_B B^{-1} = [0 \ 16 \ 20] \begin{bmatrix} 1 & -1/6 & -2/3 \\ 0 & 2/3 & -1/3 \\ 0 & -1/3 & 2/3 \end{bmatrix} = [0 \ 4 \ 8]$$

dir. Temelde olmayan değişkenler x_3, x_5, x_6 için $z_j - c_j$ değerleri ise sırasıyla

$$z_3 - c_3 = Y P_3 - c_3 = [0 \ 4 \ 8] \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix} - 16 = 0$$

$$z_5 - c_5 = Y P_5 - c_5 = [0 \ 4 \ 8] \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} - 0 = 4$$

$$z_6 - c_6 = Y P_6 - c_6 = [0 \ 4 \ 8] \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} - 0 = 8$$

olup optimumluk bozulmadığından mevcut çözüm yine optimumdur. x_3 değeri sıfır olduğundan mevcut kâr 920 değeri de değişmez.

Öte yandan mevsimlik lastiklerin kârının 16 yerine 17 olduğu hesaplanırsa bu durumda x_3 temel dışı değişken olduğundan Y değeri değişmez iken temel dışı değişkenler olan x_3 için $z_j - c_j$ değeri

$$z_3 - c_3 = Y P_3 - c_3 = [0 \ 4 \ 8] \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix} - 17 = -1$$

olacaktır. Bu değer negatif olması optimumluğun bozulmasına neden olacaktır. Bu durumda ikinci durumda belirtildiği üzere optimumluğun sağlanması için probleme (primal) Simpleks algoritması uygulanmalıdır. Bu değer negatif olması x_3 'ün giren değişken olduğunu göstermekte olup mevcut simpleks tablosu Tablo 54'teki gibi elde edilir.

Tablo 54. Örnek 15.1 için simpleks tablosu

Temel değ.	x_1	x_2	x_3	x_4	x_5	x_6	Çözüm
z	0	0	-1	0	4	8	920
x_4	0	0	1	1	-1/6	-2/3	25
x_2	0	1	1	0	2/3	-1/3	20
x_1	1	0	0	0	-1/3	2/3	30

Tablo 54 optimum çözümün elde edildiği Tablo 45 ile elde edilmiştir. Tablo 54'ten optimumluğun sağlanması için x_3 'ün giren değişken olması gerekirken

oranlara bakıldığında x_2 de çıkan değişken olmalıdır. Bu durumda x_2 satırı ile x_3 sütunun kesiştiği pivot eleman 1 olduğundan direkt olarak Gauss Jordan algoritması uygulanabilir. Bu durumda Tablo 55 elde edilir.

Tablo 55. Örnek 15.1 için birinci yineleme ile elde edilen simpleks tablosu

Temel değ.	x_1	x_2	x_3	x_4	x_5	x_6	Çözüm
z	0	1	0	0	14/3	23/3	940
x_4	0	-1	0	1	-5/6	-1/3	5
x_3	0	1	1	0	2/3	-1/3	20
x_1	1	0	0	0	-1/3	2/3	30

Tablo 55'ten optimumluğun sağlandığı görülür. Bu durumda temel değişken değerleri $x_1 = 30, x_3 = 20, x_4 = 5$ olup optimum değer 940 olarak elde edilir. Bu durumda kâr'ın arttığı gözlemlenir.

1.7.2.2. Yeni bir faaliyetin eklenmesi

Bir doğrusal programlama problemine yeni bir faaliyetin eklenmesi probleme yeni bir değişken eklenmesi anlamına gelecektir. Bu değişkenin eklenmesi, daha avantajlı (kârlı) bir çözüm elde edildiğinde anlamlı hale gelecektir. Bu sayede optimum çözüm değeri geliştirilmiş olacaktır. Yeni eklenen faaliyetin optimum çözümü arttırıp arttırmadığına ise yeni faaliyet için

$$z_j - c_j = YP_j - c_j$$

değerinin hesaplanması ile karar verilir. Burada Y matrisi mevcut optimum dual değişkenleri, P_j vektörü kaynak kullanımını ve c_j ise eklenen faaliyetin birim başına kârını göstermektedir. Eğer $z_j - c_j$ değeri mevcut optimum çözümü sağlıyorsa yeni eklenen faaliyet avantajlı değildir. Aksi halde yeni eklenen faaliyet kârlı olduğu anlamına gelecek olup temel çözüme dahil edilmelidir.

Örnek 15.2. Lastik üreten fabrika modelinde son zamanlarda elektrikli araçların üretiminin artması ile bu modellere özel yeni lastik modelleri üretilmesi kararlaştırılmıştır. Bu lastiklerin üretimi için tahmini işlem görme süreleri ve tahmini kâr aşağıdaki gibi belirlenmiştir.

1. bölüm	2. bölüm	3. bölüm	Kâr
1	2	2	22

x_7 elektrikli araçlar için özel olarak üretilen lastik sayısını göstermek üzere bu durumda matematiksel model

$$\max z = 20x_1 + 16x_2 + 15x_3 + 22x_7$$

$$1.5x_1 + x_2 + 2x_3 + x_7 \leq 90$$

$$x_1 + 2x_2 + 2x_3 + 2x_7 \leq 70$$

$$2x_1 + x_2 + x_3 + 2x_7 \leq 80$$

$$x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_7 \geq 0$$

olarak elde edilir. Bu durumda C_B değişmediğinden Örnek 15.1'den $Y = [0 \ 4 \ 8]$ olduğundan

$$YP_7 - c_7 = [0 \ 4 \ 8] \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix} - 22 = 2 > 0$$

olduğundan yeni eklenen değişkenin sisteme eklenmesi kârı arttırmayacağı görüldüğünden üretimi avantaj sağlamamaktadır.

Öte yandan eğer elektrikli araçlar için üretilen lastikler 1. bölümde 2, 3. bölümde 1 saat iş gördüğü düşünülür diğer değerler aynı kalırsa bu durumda

$$YP_7 - c_7 = [0 \ 4 \ 8] \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix} - 22 = -6 < 0$$

olduğundan x_7 'nin sisteme dahil edilmesi firma için avantajlıdır. Yeni optimum noktaya ulaşmak için $B^{-1}P_7$ matrisini hesaplayalım:

$$B^{-1}P_7 = \begin{bmatrix} 1 & -1/6 & -2/3 \\ 0 & 2/3 & -1/3 \\ 0 & -1/3 & 2/3 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}.$$

Elde edilen matris son Simpleks algoritmasında yazılırsa Tablo 56 elde edilir.

Tablo 56. Örnek 15.2 için Simpleks tablosu

Temel değişkenler	x_1	x_2	x_3	x_7	x_4	x_5	x_6	Çözüm
z	0	0	1	-6	0	4	8	920
x_4	0	0	1	1	1	-1/6	-2/3	25
x_2	0	1	1	1	0	2/3	-1/3	20
x_1	1	0	0	0	0	-1/3	2/3	30

z satırında negatif sayı bulunduğundan bu tabloya Simpleks algoritması uygulanır. Dolayısı ile x_7 sisteme giren değişken olmalıdır. Çıkan değişkenin ise

oranlara bakıldığında x_2 olduğu görülür. Pivot eleman 1 olduğundan Tablo 55'e Gauss-Jordan algoritması uygulanabilir. Bu durumda Tablo 57 elde edilir.

Tablo 57. Örnek 15.2 için Gauss-Jordan algoritması uygulanması

Temel değişkenler	x_1	x_2	x_3	x_7	x_4	x_5	x_6	Çözüm
z	0	6	7	0	0	8	6	1040
x_4	0	-1	0	0	1	-5/6	-1/3	5
x_7	0	1	1	1	0	2/3	-1/3	20
x_1	1	0	0	0	0	-1/3	2/3	30

Tablo 55'te z satırında negatif eleman kalmadığından optimum çözüm elde edilir. Bu durumda temel değişken değerleri $x_1 = 30, x_7 = 20, x_4 = 5$ olup optimum çözüm ise 1040 olarak elde edilir. Dolayısı ile elde edilen çözümün kâr değerini arttırdığı görülmektedir.

1.7.2.3. Optimumluğun korunduğu değişim aralığının belirlenmesi

b 'nin elemanlarının uygunluk aralığının belirlenmesinde yapıldığı gibi optimumluğun korunduğu aralığın bulunması için de mevcut c_j değeri yerine pozitif veya negatif değişim miktarını temsil eden $c_j + d_j$ alınması ile belirlenebilir. Bu durumda yeni $z_j - c_j$ değerleri belirlenerek problemin maksimizasyon veya minimizasyon olmasına göre optimumluk durumuna karar verilir.

Örnek 15.3. Lastik üretimi yapan firma modelinde $c_1 = 20$ değeri d_1 kadar değiştirildiğinde mevcut optimum çözümün değişmeyeceği aralığı belirleyelim. Bu durumda $C_B = [0 \ 16 \ 20 + d_1]$ olacağından

$$Y = C_B B^{-1} = [0 \ 16 \ 20 + d_1] \begin{bmatrix} 1 & -1/6 & -2/3 \\ 0 & 2/3 & -1/3 \\ 0 & -1/3 & 2/3 \end{bmatrix} = \left[0 \ 4 - \frac{d_1}{3} \ 8 + \frac{2d_1}{3} \right]$$

elde edilir. c_j 'deki değişiklikler kısmında da değinildiği üzere temel dışı değişkenler için $z_j - c_j = Y P_j - c_j$ değerlerine bakılır. Burada temel dışı değişkenler x_3, x_5 ve x_6 olduğundan

$$z_3 - c_3 = Y P_3 - c_3 = \left[0 \ 4 - \frac{d_1}{3} \ 8 + \frac{2d_1}{3} \right] \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix} - 15 = 16 - 15 = 1 > 0$$

$$z_5 - c_5 = YP_5 - c_5 = \begin{bmatrix} 0 & 4 - \frac{d_1}{3} & 8 + \frac{2d_1}{3} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} - 0 = 4 - \frac{d_1}{3} \geq 0$$

$$z_6 - c_6 = YP_6 - c_6 = \begin{bmatrix} 0 & 4 - \frac{d_1}{3} & 8 + \frac{2d_1}{3} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} - 0 = 8 + \frac{2d_1}{3} \geq 0$$

olmalıdır. Buradan problemin optimumluğunun değişmediği aralığın $-12 \leq d_1 \leq 12$ olduğu belirlenir.

Benzer şekilde $c_2 = 16$ değeri d_2 kadar değiştirildiğinde $C_B = [0 \quad 16 + d_2 \quad 20]$ olacağından

$$Y = C_B B^{-1} = [0 \quad 16 + d_2 \quad 20] \begin{bmatrix} 1 & -1/6 & -2/3 \\ 0 & 2/3 & -1/3 \\ 0 & -1/3 & 2/3 \end{bmatrix} = \begin{bmatrix} 0 & 4 + \frac{2d_2}{3} & 8 - \frac{d_2}{3} \end{bmatrix}$$

elde edilir. Bu durumda

$$\begin{aligned} z_3 - c_3 = YP_3 - c_3 &= \begin{bmatrix} 0 & 4 + \frac{2d_2}{3} & 8 - \frac{d_2}{3} \end{bmatrix} \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix} - 15 = 16 + d_2 - 15 \\ &= 1 + d_2 \geq 0 \end{aligned}$$

$$z_5 - c_5 = YP_5 - c_5 = \begin{bmatrix} 0 & 4 + \frac{2d_2}{3} & 8 - \frac{d_2}{3} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} - 0 = 4 + \frac{2d_2}{3} \geq 0$$

$$z_6 - c_6 = YP_6 - c_6 = \begin{bmatrix} 0 & 4 + \frac{2d_2}{3} & 8 - \frac{d_2}{3} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} - 0 = 8 - \frac{d_2}{3} \geq 0$$

olduğundan problemin optimumluğunun değişmediği aralık $-1 \leq d_2 \leq 24$ olarak belirlenir.

Eğer temel dışı değişken olan x_3 'ün değeri d_3 kadar değiştirilirse C_B değişmediğinden Y vektörü de Örnek 15.1'deki gibi $[0 \quad 4 \quad 8]$ olarak belirlenir. Temelde olmayan değişkenler için $z_j - c_j$ değerleri ise

$$z_3 - c_3 = YP_3 - c_3 = [0 \quad 4 \quad 8] \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix} - (15 + d_3) = 1 - d_3 \geq 0$$

$$z_5 - c_5 = YP_5 - c_5 = [0 \quad 4 \quad 8] \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} - 0 = 4 > 0$$

$$z_6 - c_6 = YP_6 - c_6 = [0 \quad 4 \quad 8] \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} - 0 = 8 > 0$$

olduğundan d_3 'ün 1'den küçük eşit olduğu her değer optimumluğu değiştireceği gözlemlenmiş olur.

Son olarak c_1, c_2 ve c_3 değerleri aynı anda sırasıyla d_1, d_2 ve d_3 kadar arttırılırsa bu durumda $C_B = [0 \quad 16 + d_2 \quad 20 + d_1]$ olacağından

$$\begin{aligned} Y = C_B B^{-1} &= [0 \quad 16 + d_2 \quad 20 + d_1] \begin{bmatrix} 1 & -1/6 & -2/3 \\ 0 & 2/3 & -1/3 \\ 0 & -1/3 & 2/3 \end{bmatrix} \\ &= \left[0 \quad 4 + \frac{2d_2}{3} - \frac{d_1}{3} \quad 8 - \frac{d_2}{3} + \frac{2d_1}{3} \right] \end{aligned}$$

bulunur. Temel dışı değişken değerleri ise

$$\begin{aligned} z_3 - c_3 &= YP_3 - c_3 = \left[0 \quad 4 + \frac{2d_2}{3} - \frac{d_1}{3} \quad 8 - \frac{d_2}{3} + \frac{2d_1}{3} \right] \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix} - (15 + d_3) \\ &= 1 + d_2 - d_3 \geq 0 \\ z_5 - c_5 &= YP_5 - c_5 = \left[0 \quad 4 + \frac{2d_2}{3} - \frac{d_1}{3} \quad 8 - \frac{d_2}{3} + \frac{2d_1}{3} \right] \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} - 0 \\ &= 4 + \frac{2d_2}{3} - \frac{d_1}{3} \geq 0 \\ z_6 - c_6 &= YP_6 - c_6 = \left[0 \quad 4 + \frac{2d_2}{3} - \frac{d_1}{3} \quad 8 - \frac{d_2}{3} + \frac{2d_1}{3} \right] \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} - 0 \\ &= 8 - \frac{d_2}{3} + \frac{2d_1}{3} \geq 0 \end{aligned}$$

eşitsizlik sistemi elde edilir.

2. YAPILAN ÇALIŞMALAR

Doğrusal programlama modellerinin çözümü için kullanılabilecek olan birçok bilgisayar destekli program mevcuttur. Fakat bu programların çoğunda programın çözüm aşamaları yerine optimum çözüm gösterilmektedir. Fakat lisans derslerinde verilen lineer programlama problemlerinde öğrenciler çoğunlukla işlem hataları yapmakta ve bilgisayarın bulduğu sonuç ile öğrencinin bulduğu sonuçlar farklılıklar göstermektedir. Bu işlem hatalarının hangi aşamada meydana geldiğini gösteren bir program yazılması öğrencilerin derse olan ilgilerini arttırabileceği düşüncesi bu tez çalışmasının ana konusunu temsil etmektedir. Bu amaçla yazılacak programda her bir aşama sonundaki Simpleks tablolarının ekranda yazılması amaçlanmıştır. Bunun yanında temel değişkenleri ve her bir aşamada giren/çıkan değişkenlerin ekranda yazdırılması ile öğrenci dostu bir program oluşturulması hedeflenmiştir.

Birçok bilgisayar programlama dili Simpleks algoritmalarında tercih edilebilir. Fakat programlama dili seçilirken seçilen problemin türü, programın kullanışlılığı, ulaşılabilirliği vb. faktörler göz önüne alınması öğrencilerin programı kullanabilmesi açısından son derece önemlidir. Bu faktörler göz önüne alındığında doğrusal programlama problemlerinin çözümü için kullanılabilecek programlardan bir tanesi ücretsiz bir program olan Maxima programıdır.

Maxima Programı ilk olarak MIT (Massachusetts Teknoloji Enstitüsü, Boston, ABD) de geliştirilmiş olan Macsyma isimli ticari yazılımdan türetilmiş bir programdır. Program günümüz itibariyle gönüllü bir araştırma grubu tarafından geliştirilerek <http://www.maxima.sourceforge.net/> adresinden ücretsiz olarak kullanılabilmektedir.

Maxima programı hem arayüz olarak hem de program yazımı açısından kullanışlı bir programdır. Öğrencilerin kendi kütüphanelerini oluşturabilme imkânı tanıyan bu program sayesinde kullanıcılar, istenildiği zaman bu kütüphanedeki programları çağırarak kullanabilmektedir. Kütüphane oluşturmada “*blok*” adı verilen yapılar kullanılmaktadır. Bu yapıların genel şekli

blok_ismi(input listesi):=block([varsaya yerel değişkenler], diğer komutlar)

olarak ifade edilebilir (Coşkun, 2018 Maxima). Burada *blok_ismi* programa vereceğimiz ismi temsil ederken, *input listesi* kullanıcıdan alınacak olan girdilerin listesini göstermektedir. Program *block* yapısı ile girdileri alarak programın amacına göre işler. Eğer yerel değişkenler varsa blok yapısının en başında belirtilir. Bu programın sonucunda diğer program için gerekli olan veri *return* komutu kullanıcıya geri gönderilir. Eğer program hazırlanırken geri dön (*return*) fonksiyonunun kullanılmaması durumunda en son hesaplanan değişken değerleri *block* yapısı tarafından geri gönderilir.

Bu çalışmada blok yapısı Simpleks yöntemleri için oluşturulmaktadır. İlk olarak ise klasik Simpleks algoritması için Maxima programı hazırlanmıştır. Çünkü, M-yöntemi, iki aşamalı Simpleks yöntemlerinde problemin belli aşamalarından sonra klasik Simpleks algoritması probleme uygulanması gerekmektedir. Bu yüzden her bir programın içinde klasik Simpleks algoritmasını tekrar tekrar yazmak yerine blok yapısı çağrılarak sonuçların elde edilmesi sağlanmıştır. Bunun için ilgili program hazırlandıktan sonra *.mac* uzantılı bir dosya olarak kaydedilerek maxima kütüphanesine eklenmesi gerekmektedir. Bu kapsamda *simpleks_algoritma.mac* isimli bir blok program oluşturulmuş olup bu programın kodları Algoritma 1’de verilmiştir. Öte yandan bu *mac* dosyasının mutlaka çalışılan dosyanın içerisine kaydedilmiş olması gerektiği unutulmamalıdır.

Hazırlanan program maxima’nın kendi kütüphanesine ait bir program olmadığından çalıştırılmadan önce ilgili kütüphane maxima ortamında yüklenmelidir. Bunun için program çağrılmadan önce *load(program_adi)* komutu ile program çalıştırılmaya hazır hale getirilir. Öte yandan oluşturulan *mac* dosyasının içerisini açmadan programın içerisindeki komutları görmek istenirse maxima ortamında *batch(program_adi)* komutu kullanılabilir.

Algoritma 1. Simpleks algoritması

Bu program Simpleks algortiması ile problemin optimum değerini belirler.

Input: A: Simpleks algortimasının uygulanacağı matris

Output: A: Simpleks algoritmasının uygulanması sonucunda oluşan matris

Not edelim ki giriş matrisi verilirken 1. satır değişken isimlerini, 2. satır z satırını ve 1. sütun ise temel değişkenleri içerecek şekilde sisteme girilmelidir. Eğer sayı yerine değişken ismi kullanılacaksa bunlar " "(çift tırnak) arasında yazılmalıdır.

Eğer başlangıçta temel değişken isimleri bilinmiyorsa program bu değerleri kendisi hesapladığı için bunların yerine " "(boşluk) karakteri girilebilir.

Örneğin, tezdeki örnek 9'daki problem için A matrisi tablo 13'de olduğu gibi
A:matrix(["", "x_1", "x_2", "x_3", "s_1", "s_2", "s_3", "çözüm"], ["z", -20, -16, -15, 0, 0, 0, 0], ["", 1.5, 1, 2, 1, 0, 0, 90], ["", 1, 2, 2, 0, 1, 0, 70], ["", 2, 1, 1, 0, 0, 1, 80]) şeklinde girilmelidir.*/ simpleks_algoritma(A):=block(

/*Girilen matrisin satır ve sütun sayıları hesaplanarak simpleks algoritmasının **hangi** satır ve sütunlara göre uygulanacağı belirleniyor*/

A_satir_sayisi:length(A),

A_sutun_sayisi:length(transpose(A)),

/*-----*/

asama:1,/* Ekrana hangi simpleks aşamasından gidildiği yazdıracak olan değişken*/ /*Başlangıçtaki temel değişkenlerin belirlemek için kullanılacak matris oluşturuluyor:*/ S_satir_sayisi:A_satir_sayisi-2,

S_sutun_sayisi:A_sutun_sayisi-2,

S:zeromatrix(S_satir_sayisi,S_sutun_sayisi),

for i:1 thru S_satir_sayisi do (for j:1 thru S_sutun_sayisi do S[i][j]:A[i+2][j+1]),

St:transpose(S),

/*-----*/

/*İlk matristeki temel değişkenler belirlenerek A matrisinde ilgili yere yazdırılıyor:*/

I:ident(S_satir_sayisi),

k:0,/*Temel değişkenlerin sayısını gösteriyor. Başlangıçta 0 alınıyor*/

kill(yer),/*Temel değişkenin yerini gösteren vektör*/

for i:1 thru S_satir_sayisi do (for j:1 thru S_sutun_sayisi do (if St[j]=I[i] then (k:k+1,yer[k]:j))),

temel_degisken:zeromatrix(1,k),/*Temel değişkenleri tutan matris*/

temel_degisken_deger:zeromatrix(1,k),/*Temel değişkenlerin değerlerini tutan matris*/

for i:1 thru k do (temel_degisken[1][i]:A[1][yer[i]+1],

```

temel_degisken_deger[1][i]:A[i+2][A_sutun_sayisi],A[i+2][1]:A[1][yer[i]+
1]),

disp("Temel deęişkenler:",temel_degisken),
disp("Temel deęişkenlerin deęerleri:",temel_degisken_deger),
disp("Başlangıç simpleks tablosu:"),
disp(A),
/*-----*/
/*Deęişken isimlerinin yazılı olduęu satırlar çıkarılarak simpleks
algoritmasının uygulancaęı ana matris oluşturuluyor:*/
Simp_satir_sayisi:A_satir_sayisi-1,
Simp_sutun_sayisi:A_sutun_sayisi-1,
Simp_matris:zeromatrix(Simp_satir_sayisi,Simp_sutun_sayisi),
for i:1 thru Simp_satir_sayisi do
(for j:1 thru Simp_sutun_sayisi do Simp_matris[i][j]:A[i+1][j+1]),
/*-----*/
devam:1;/*Eđer devam 1 deęerine eşitse döngü devam edeceęi 0 ise
döngünün biteceęini gösteriyor. Döngü durdurma deęişkeni*/
while devam=1 do (
/*Simpleks algoritmasında giren deęişken belirleniyor ve A matrisinde yerine
yazdırılıyor:*/
min_deger:0,
min_sutun:0,
for i:1 thru Simp_sutun_sayisi-1 do
(if min_deger>Simp_matris[1][i]
then(min_deger:Simp_matris[1][i],min_sutun:i)),
disp("Giren deęişken:",A[1][min_sutun+1]),
/*-----*/
/*Simpleks algoritmasında çıkan deęişken belirleniyor ve A matrisinde
yerine yazdırılıyor:*/
bolme:zeromatrix(Simp_satir_sayisi-1,1)-1,
for i:1 thru Simp_satir_sayisi-1 do

```

```

        (if                               Simp_matris[i+1][min_sutun]#0                               then
bolme[i][1]:Simp_matris[i+1][Simp_sutun_sayisi]/Simp_matris[i+1][min_sutun]),
        min_satdeger:10^10,
        min_satir:0,
        for i:1 thru Simp_satir_sayisi-1 do (if bolme[i][1]>0 then
        (if                               min_satdeger>bolme[i][1]
then(min_satdeger:bolme[i][1],min_satir:i+1))),
        disp("Çıkan                               değişken",A[min_satir+1][1]),
A[min_satir+1][1]:A[1][min_sutun+1],
        /*-----*/
        /*Pivot elemanı 1 yapıp matris Gauss-Jordan işlemine hazır hale
        getiriliyor:*/
        Simp_matris[min_satir]:Simp_matris[min_satir]/Simp_matris[min_satir][mi
n_sutun],
        /*Gauss-Jordan işlemi Simpleks matrisine uygulanıyor:*/
        for i:1 thru Simp_satir_sayisi do
        (if i#min_satir then
        Simp_matris[i]:Simp_matris[i]-
Simp_matris[min_satir]*Simp_matris[i][min_sutun]) ,
        disp([asama, ". aşamanın sonucunda elde edilen simpleks tablosu:"]),
        /*-----*/
        /*A matrisinde yeni temel değişkenler ve değerleri ile sonuç ilgili yere
        yazılıyor:*/
        for i:1 thru Simp_satir_sayisi do (for j:1 thru Simp_sutun_sayisi do
A[i+1][j+1]:Simp_matris[i][j]),
        disp(A), temel_degisken:zeromatrix(1,k),
        temel_degisken_deger:zeromatrix(1,k),
        for i:1 thru A_satir_sayisi-2 do (temel_degisken[1][i]:A[i+2][1],
        temel_degisken_deger[1][i]:A[i+2][A_sutun_sayisi]),
        disp("Temel değişkenler:",temel_degisken),
        disp("Temel değişkenlerin değerleri:",temel_degisken_deger),
        disp("Temel değişkenlere karşılık gelen çözüm:",A[2][A_sutun_sayisi]),

```

```

/*-----*/
/*Simpleks algoritmasına devam edilip edilmeyeceği kontrol ediliyor:*/
dongu:0,
for i:1 thru Simp_sutun_sayisi-1 do (if Simp_matris[1][i]<0 then
dongu:dongu+1),
if dongu>0 then devam:1 else devam:0,
/*-----*/
asama:asama+1), return(A))$

```

Algoritma 1'de maxima ortamında oluşturulan simpleks algoritması verilmiştir. Algoritmanın üst kısmında algoritmanın çalıştırılması için gerekli giriş elemanı olan A matrisi hakkında kullanıcıya kısa bir bilgi verilmiştir. Algoritmada da değinildiği üzere giriş elemanı olan A matrisi, başlangıç Simpleks tablosunu belirtmektedir. Fakat kullanıcı ile aynı değişken isimlerinin kullanılması tercih edildiğinden kullanıcı A matrisinin ilk satırına kullandığı değişken isimlerini kendisi girmelidir. Bu sayede elde edilecek çözümler ve değişken isimleri kullanıcının bulacağı sonuçlar ile uyumlu olması amaçlanmıştır. Verilen başlangıç simpleks matrisi ile program ilk olarak başlangıçtaki temel değişkenleri belirleyerek ilgili matriste 1. Sütundaki ilgili yere atamaktadır.

Programda dikkat edilmesi gereken önemli bir nokta ise program bir maksimizasyon problemine göre hazırlanmış olmasıdır. Çünkü programı durdurma kriteri z satırında negatif eleman bulunmamasıdır. Dolayısı eğer problem bir minimizasyon problemi ise ilk olarak problem maksimizasyon problemine dönüştürülmeli ve A matrisindeki z satırı ona göre girilmesi gerekmektedir.

Program simpleks algoritmalarını yürütürken *while* döngüsü yardımıyla z satırında negatif eleman bulunmayana kadar sırasıyla giren ve çıkan değişkenleri belirleyerek ekrana yazdırmakta, sonrasında ise pivot elemanı 1 yaparak Gauss Jordan algoritmasını uygulamaktadır. Programın sonunda *return(A)* komutu ile en son elde edilen simpleks tablosunu geri gönderilmektedir. Bu sayede M yöntemi, iki aşamalı simpleks yöntemlerinde gerekli olan simpleks algoritması, ilgili program içerisinde tekrar yazdırılmasına gerek olmadan efektif bir çözüm elde edilmesine olanak sağlayacaktır. Fakat eğer hazırlanan program diğer program tarafından

kullanılacaksa kaydedilirken uzantısının *.mac* olmasına dikkat edilmelidir. Çünkü hazırlanan dosya çağrılmadan önce yüklenilerek kullanılmaya hazır hale getirilmelidir. Bu amaçla dosyalar için maxima ortamında *.mac* uzantılı dosyalar hazırlanır. Bunun için program yazıldıktan sonra *dosya->çıkart* ile ilgili dosya *.mac* uzantılı dosya olarak kaydedilir. Böylece ilgili dosya kütüphaneye eklenerek diğer programlarda kullanılmaya hazır hale getirilir. Öte yandan hazırlanan dosya maxima'nın kendi hazır dosyalarından biri olmadığı için diğer programlarda kullanılmadan önce maxima ortamına yüklenmesi gerekir. Bunun için bu programın çağrılacağı programda ilgili dosya kullanılmadan önce “*load(program_ismi.mac)*” yazılarak program hazır hale getirilmelidir. Fakat sonrasında ilgili dosyada değişiklik yapmak istendiğinde *.mac* uzantılı dosya açılmadığından hazırlanan dosya *.wxmx* uzantısı ile de kaydedilmelidir. Bu dosyada gerekli değişiklikler yapıldıktan sonra tekrar *.mac* uzantısı ile kaydedilmesi unutulmamalıdır.

Dikkat edilirse program içerisinde kullanılan *disp* komutları her bir aşamadaki temel değişkenler, temel değişken değerleri ve her bir adımdaki oluşan simpleks tabloları ekranda yazdırılarak kullanıcının her bir aşamayı ayrıntılı olarak görmesi sağlanmış olmaktadır. Böylece kullanıcı, bulduğu sonuçlar ile programın bulduğu sonuçları karşılaştırabildiğinden elde ettiği sonuçların doğruluğunu gözlemleyebilmektedir.

Bu programdaki *A* matrisinin nasıl girilmesi gerektiğini ve programın nasıl çalıştırıldığını bir örnek üzerinden ele alalım.

Örnek 16. Giriş kısmında incelenen lastik üreten firma modelini tekrar ele alalım. Bu durumda başlangıç simpleks tablosu Tablo 58 gibi elde edilmiştir.

Tablo 58. Örnek 16 başlangıç simpleks tablosu

Temel değişkenler	z	x_1	x_2	x_3	s_1	s_2	s_3	Çözüm
z	1	-20	-16	-15	0	0	0	0
s_1	0	1.5	1	2	1	0	0	90
s_2	0	1	2	2	0	1	0	70
s_3	0	2	1	1	0	0	1	80

Tablo 58'deki başlangıç simpleks tablosu ilgili programa bir matris olarak girilmesi gerekmektedir. Bu *A* matrisinin birinci satırı Tablo 58'in 1. satırındaki

değişkenlerin isimlerinin aynı sırada yazılmış hali olmalıdır. Fakat 1. satırdaki değişkenler bir sayı değil bir karakterdirler. Bu karakterler matrisin ilgili yerine yazılırken iki tırnak arasında (" ") yazılmalıdır. Tablo 58'in 1. satır 1. sütun elemanı boş olduğundan bu karakter maxima'da " " olarak girilebilir. Diğer karakterlerin her biri ise tırnak arasında yazılabilirler. Fakat alt indis kullanımı için bu çalışmada alt çizgi sembolü kullanılmaktadır. Yani, tablonun 1. satırı A matrisinin 1. satırına yazılırken [" ", "x_1", "x_2", "x_3", "s_1", "s_2", "s_3", "çözüm"] şeklinde girilebilir. Öte yandan maxima'da matris yazılırken matrisin her bir satırları [] arasına aralarında virgül konularak yazılmalıdır. Ayrıca her bir sütun arasına da virgül konulması unutulmamalıdır. Dolayısı ile Tablo 58'deki başlangıç simpleks tablosu yardımıyla ilgili algoritma çalıştırılması Şekil 6'daki gibi verilir.

```
A:matrix([" ", "x_1", "x_2", "x_3", "s_1", "s_2", "s_3", "çözüm"], ["z", -20, -16, -15, 0, 0, 0],
[" ", 1.5, 1.2, 1.0, 0, 90], [" ", 1.2, 2.0, 1.0, 70], [" ", 2.1, 1.0, 0, 1, 80]);
load("simpleks_algoritma.mac");
simpleks_algoritma(A);
```

Şekil 6. Örnek 16 için Maxima'da girilmesi gereken komutlar

Şekil 6'dan görüleceği üzere ilk olarak A matrisi sisteme girilmiştir. Not edelim ki bu programda z satırı oluşturulurken her zaman problemin bir maksimizasyon problemi olarak girilmesi gerekmektedir. Daha sonra çağrılacak program *load* komutu yardımıyla maxima'ya yüklenerek kullanıma hazır hale geldikten sonra ilgili program A girdisi ile birlikte çalıştırılmıştır. Öte yandan Şekil 6'dan dikkat edilirse başlangıçtaki temel değişkenleri de program kendisi belirlediği için bu temel değişkenler yerine boşluklar konulmuştur. Program *shift+enter* komutuyla çalıştırıldığında temel değişkenlerin de yerleştirildiği başlangıç simpleks tablosu Şekil 7'deki gibi ekrana yazdırılır.

Başlangıç simpleks tablosu:

	x ₁	x ₂	x ₃	s ₁	s ₂	s ₃	çözüm
z	-20	-16	-15	0	0	0	0
s ₁	1.5	1	2	1	0	0	90
s ₂	1	2	2	0	1	0	70
s ₃	2	1	1	0	0	1	80

Şekil 7. Programın elde ettiği başlangıç simpleks tablosu

Şekil 7’de programın elde ettiği başlangıç Simpleks tablosu ile Tablo 58’deki başlangıç simpleks tabloları birbiri ile uyumludur. Bu tablonun hemen üstünde ise problemin başlangıçtaki temel değişkenleri ve temel değişkenlerin değerleri Şekil 8 de görüldüğü gibi ekranda yazdırılmıştır.

Temel değişkenler:

$(s_1 \ s_2 \ s_3)$

Temel değişkenlerin değerleri:

$(90 \ 70 \ 80)$

Şekil 8. Temel değişkenler ve temel değişkenlerin değerleri

Bu aşamadan sonra program A matrisinin z satırında negatif eleman kalmayıncaya kadar, simpleks algoritmalarını kendi içerisinde yürütmekte ve her aşamanın sonunda kullanıcıya her bir aşamada giren ve çıkan değişkenler ile elde edilen simpleks tabloaları ekrana yazdırmaktadır. Örneğin, ilk simpleks algoritması sonucunda elde edilen verilen Şekil 9’daki gibi ekrana yazdırılmıştır.

Giren değişken:

x₁

Çıkan değişken

s₃

[1., aşamanın sonucunda elde edilen simpleks tablosu:]

	x ₁	x ₂	x ₃	s ₁	s ₂	s ₃	çözüm
z	0	-6	-5	0	0	10	800
s ₁	0.0	0.25	1.25	1	0	-0.75	30.0
s ₂	0	$\frac{3}{2}$	$\frac{3}{2}$	0	1	$-\left(\frac{1}{2}\right)$	30
x ₁	1	$\frac{1}{2}$	$\frac{1}{2}$	0	0	$\frac{1}{2}$	40

Şekil 9.1. aşamada elde edilen simpleks tablosu

Öte yandan program her bir aşama sonunda elde edilen sonuçları da ilgili tablonun hemen altında kullanıcıya sunmaktadır. Örneğin, 1. aşama sonucunda elde edilen temel değişkenler ve temel değişkenlerin değerleri ile elde edilen çözüm şekil 10'daki gibi ekrana yazdırılmıştır.

Temel değişkenler:
 $(s_1 \ s_2 \ x_1)$
 Temel değişkenlerin değerleri:
 $(30.0 \ 30 \ 40)$
 Temel değişkenlere karşılık gelen çözüm:
 800

Şekil 10. 1. aşama sonucunda elde edilen sonuç

Öte yandan Şekil 10'da elde edilen 1. aşama sonundaki simpleks tablosu Tablo 16'da elde edilen sonuçlar ile uyumlu olduğu görülür. 1. aşama sonucunda z satırında negatif eleman bulunduğundan program simpleks algoritmasını yürütmeye devam edecektir. Bu durumda giren ve çıkan değişkenler ile 2. aşama sonucunda elde edilen simpleks tablosu Şekil 11'deki gibi ekrana yazdırılmıştır.

Giren değişken:

x_2

Çıkan değişken

s_2

[2., aşamanın sonucunda elde edilen simpleks tablosu:]

	x_1	x_2	x_3	s_1	s_2	s_3	çözüm
z	0	0	1	0	4	8	920
s_1	0.0	0.0	1.0	1	-0.16666666666666666	-0.6666666666666666	25.0
x_2	0	1	1	0	$\frac{2}{3}$	$-\left(\frac{1}{3}\right)$	20
x_1	1	0	0	0	$-\left(\frac{1}{3}\right)$	$\frac{2}{3}$	30

Şekil 11. 2. aşamada programın elde ettiği sonuçları gösteren bölüm

Şekil 11'de verilen giren ve çıkan değişkenler ile oluşan simpleks tablosu Tablo 19 ile uyumlu olduğu görülür. Öte yandan Şekil 11'den görüleceği üzere z

satırında 2. aşamanın sonunda negatif eleman kalmadığından program simpleks algoritmasını sonlandırmaktadır. 2. aşama sonucunda elde edilen çözüm temel değişken değerleri ve optimum çözüm ise Şekil 12’deki gibi ekrana yazdırılmıştır.

Temel değişkenler:
 $(s_1 \ x_2 \ x_1)$
Temel değişkenlerin değerleri:
 $(25.0 \ 20 \ 30)$
Temel değişkenlere karşılık gelen çözüm:
920

Şekil 12. Problemin temel değişkenlerinin değerleri ile optimum çözümü

Öte yandan program en son aşamada A matrisini geri gönderdiğinden optimum çözümden sonra ekranda en son elde edilen simpleks matrisi de yazdırılacaktır.

Örnek 16’dan görüldüğü üzere Maxima ortamında oluşturulan program tamamen kullanıcı dostu bir programdır. Bu program ile kullanıcı kendi yaptığı işlemlerin doğruluğunu her aşamada rahatlıkla kontrol edebilmektedir. Ayrıca, kullanıcı giren/çıkan değişkenlerin doğruluğunu veya elde ettiği simpleks aşamalarındaki simpleks tablolarının doğruluğunu bu program sayesinde karşılaştırma şansı bulmuş olur. Maxima programının ücretsiz olması ile her öğrencinin rahatlıkla ulaşabileceği bir program olduğundan lisans veya yüksek lisans öğrencileri için güzel bir yardımcı program olacağı ön görülmektedir. Bazı programlarda genellikle sadece elde edilen sonuçlar verilirken bazı programlarda optimal sonuç ile her bir aşamadaki simpleks tabloları verilmektedir. Ayrıca mevcut programların maliyetlerin çok olması öğrenciler için bir dezavantaj oluşturmaktadır. Bu sebeplerden dolayı ücretsiz erişim sağlanabilen Maxima programı ile oluşturulan yukarıdaki program hem simpleks tablolarını hem giren/çıkan değişkenleri hem de her aşama sonunda temel değişkenlerin değerlerinin gösterdiği için öğrenciler için daha kullanışlı olduğu düşünülmektedir.

Şimdi grafik yöntemi, M yöntemi, iki aşamalı simpleks ve dual simpleks algoritmalarının maxima ortamına kodlanmasını verelim. Bu programların bir çoğunda simpleks algoritması kullanılacağından yukarıdaki hazırlanmış program

.mac uzantısı ile kütüphanede tutularak diğer programlarda kullanılmasına olanak sağlanmıştır.

Bundan sonraki programlarda giriş elemanları verilirken genel olarak aşağıdaki kurallara dikkat edilmiştir.

1. Dual yöntem hariç optimizasyon probleminde optimum edilecek amaç fonksiyonunun katsayıları bir maksimizasyon problemi olacak şekilde P matrisine girilmelidir. Eğer problem minimizasyon problemi ise amaç fonksiyonu eksi ile çarpıldıktan sonra P matrisine girilmelidir.

2. Kısıtlamalarda küçük eşittir ($\leq$) içeren kısıtlamaları $A_{ke}x = b_{ke}$ olacak şekilde tekrar yazdırılarak A_{ke} matrisini programda A_{ke} matrisi yerine, b_{ke} vektörü ise bir sütun matrisi olacak şekilde b_{ke} matrisine girilmelidir. Eğer programda küçük eşittir ($\leq$) kısıtlaması mevcut değilse $A_{ke}:matrix([\text{ }]), b_{ke}:matrix([\text{ }])$ şeklinde girilmelidir.

3. Kısıtlamalarda büyük eşittir ($\geq$) içeren kısıtlamaları $A_{be}x = b_{be}$ olacak şekilde tekrar yazdırılarak A_{be} matrisini programda A_{be} matrisi yerine, b_{be} vektörü ise bir sütun matrisi olacak şekilde b_{be} matrisine girilmelidir. Eğer programda büyük eşittir ($\geq$) kısıtlaması mevcut değilse $A_{be}:matrix([\text{ }]), b_{be}:matrix([\text{ }])$ şeklinde girilmelidir.

4. Kısıtlamalarda eşitlik ($=$) içeren kısıtlamaları $A_{eq}x = b_{eq}$ olacak şekilde tekrar yazdırılarak A_{eq} matrisini programda A_{eq} matrisi yerine, b_{eq} vektörü ise bir sütun matrisi olacak şekilde b_{eq} matrisine girilmelidir. Eğer programda eşitlik ($=$) kısıtlaması mevcut değilse $A_{eq}:matrix([\text{ }]), b_{eq}:matrix([\text{ }])$ şeklinde girilmelidir.

5. Grafik yöntemi hariç uyumluluk açısından ana problemdeki değişken isimleri ile problem standart forma getirildiğinde kullanılan değişken isimleri de programa verilmelidir. Böylece kullanıcının bulduğu sonuçlar ile programın bulduğu sonuçlar arasındaki sonuçların karşılaştırılması daha kolay olacaktır. Bu amaçla yöntemde değişken isimleri diye bir giriş elemanı içeriyorsa bu giriş elemanı bir satır matrisi olacak şekilde ve her bir elemanı ana problemdeki değişkenleri içerecek şekilde sisteme girilmelidir. Örneğin, problemimiz Örnek 16'da olduğu gibi x_1, x_2, x_3 değişkenlerini içeriyorsa programa değişken isimleri $degisken:matrix(["x_1", "x_2", "x_3"])$ şeklinde girilmelidir.

6. Problem standart forma getirildiğinde probleme eklenen yeni değişkenler *yapay_degiskenler* isimli bir satır matrisi olarak sisteme girilmelidir. Örneğin, Örnek 16'da olduğu gibi sisteme s_1, s_2, s_3 değişkenleri ilave ediliyorsa programa bu değişkenler *yapay_degisken:matrix(["s_1", "s_2", "s_3"])* olarak girilmelidir.

7. Dual problemde problemin maksimizasyon veya minimizasyon olmasına göre küçük değişiklik olduğunda bu yöntemde yukarıdaki giriş elemanlarına ek olarak *problem* değişkeni girilmesi istenmiştir. Bu değişken, problem minimizasyon ise "*min*", problem maksimizasyon ise "*max*" olarak tanımlanmalıdır.

Öte yandan 5, 6 ve 7'de sisteme bir değer değil de bir değişken ismi girildiğinden çift tırnak (" ") işareti unutulmamalıdır.

Şimdi bu yöntemler için oluşturulan maxima kodlarını verelim.

2.1. Maxima ortamında grafik yönteminin kodlanması

Grafik yöntemi iki karar değişkenine sahip problemler için kullanılan yöntemlerden biridir. Bu problemlerde kısıtlamaları sağlayan çözüm bölgesi grafiksel olarak çizdirilerek bu bölge içerisinde amaç fonksiyonunun optimum çözümü belirlenir. Giriş bölümünde belirtildiği üzere optimum çözüm uygun çözüm bölgesinin köşe noktalarından birisi olacaktır.

Bu çalışmada grafik yöntemi için hazırlanan maxima algoritması aşağıda verilmiştir.

Algoritma 2. Grafik yönteminin maxima ortamına programlanması

Adım 1. Sınırlamaların satır ve sütun sayılarına bakılarak kesişim noktası belirlenerek denklem sayısı belirlenir.

Adım 2. Problemdeki kısıtlamalar lineer denklemlere dönüştürülür. Yani kesişim noktalarının belirlenmesi için küçük eşittir veya büyük eşittir gibi eşitlikler başlangıçta eşitlik olarak göz önüne alınarak kesişim noktaları belirlenmesine hazır hale getirilir.

Adım 3. Genel olarak algoritmalarda $x \geq 0, y \geq 0$ gibi kısıtlamalar her zaman mevcut olduğundan giriş elemanlarında bunların girilmesi istenmemektedir. Fakat uygun çözüm bölgesi belirlenirken eksenleri de kestiği noktalarda gerekli olduğundan bunlar ayrı denklemler olarak sisteme ilave edilir.

Adım 4. Adım 3'te elde edilen denklemlerin kesişim noktaları belirlenir.

Adım 5. Uygun çözüm bölgesinin köşe noktaları belirlenir. Bunun için kısıtlamalardaki eşitsizlikler orijinal haline dönüştürülerek Adım 4’de elde edilen kesişim noktalarından hangilerinin bütün sınırlamaları sağladığı belirlenir.

Adım 6. Adım 5’te elde edilen köşe noktalarından optimum olanı belirlenir.

Algoritma 2’de P yerine amaç fonksiyonun katsayıları, A_{ke} , A_{be} ve A_{eq} yerine sırasıyla kısıtlamalardaki küçük eşitliklerin, büyük eşitliklerin ve eşitliklerin katsayıları matris olarak girilecektir.

Algoritmadan görüleceği üzere amaç fonksiyonunun ve kısıtlamalarının katsayıları programa giriş elemanları olarak önceden girilmesi gerekmektedir. Program grafik yöntemi ile doğruların kesişim noktalarını belirleyerek optimum bölgeye karar vermekte ve bu bölgenin köşe noktalarını *mak_deger* değişkenine atayarak kullanıcının görmesi için ekrana yazdırmaktadır. Bu noktalar içerisinde optimum olanını belirleyerek bu değeri *mak_sonuc* değişkeni ile problemin sonucu olarak geri göndermektedir. En sonda ise problemin kısıtlamalarının grafikleri de kullanıcının sonuçları kontrol etmesi için çizdirilmektedir. Fakat şunu not edelim ki bir önceki bölümde de değinildiği üzere amaç fonksiyonunun katsayıları sisteme girilirken problemin maksimizasyon problemi olması gerekmektedir.

Şimdi Örnek 1 de verilmiş olan problemi maxima programını kullanarak grafik yöntemiyle çözelim.

Örnek 17. Örnek 1’deki problem

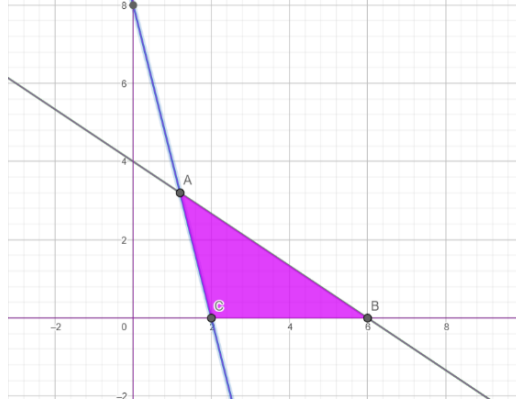
$$\max z = x_1 + 2x_2$$

$$2x_1 + 3x_2 \leq 12$$

$$4x_1 + x_2 \geq 8$$

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

olarak verilmişti. Verilen problem iki değişkenli ve iki kısıttan oluşan bir maksimizasyon problemidir. Dolayısıyla grafik yöntemle çözmeye uygundur.



Şekil 13. Örnek 1’deki kısıtlamalar için uygun bölge

Ele alınan problem matrissel olarak

$$\max z = [1 \quad 2] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = Px$$

$$[2 \quad 3] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq 12 \Rightarrow A_{ke}x \leq b_{ke}$$

$$[4 \quad 1] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \geq 8 \Rightarrow A_{kbe}x \geq b_{be}$$

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

şeklinde ifade edilebilir. Problemden eşitlik kısıtlamaları olmadığı için hazırlanan maxima koduna girdiler

P: matrix([1], [2]);

Ake: matrix([2,3]);

Abe: matrix([4,1]);

Aeq: matrix([[]]);

bke: matrix([12]);

bbe: matrix([8]);

beq: matrix([[]]);

olarak tanımlanır. Not edelim ki $x_1 \geq 0, x_2 \geq 0$ kısıtlamaları program içerisinde otomatik olarak tanımlandığı için bu kısıtlamaları programa girdi olarak verilmesi gerekmemektedir.

İlgili alt dosya çalıştırıldıktan sonra alt kısmına girdiler girildikten sonra

grafik_yontemi(P, Ake, Abe, Aeq, bke, bbe, beq);

yazılıp problem için ilgili program çalıştırılırsa çözüm bölgesinin köşe noktalarını gösteren Şekil 14 elde edilir.

Köşe Noktaları

$$\begin{pmatrix} 6 & 16 \\ 5 & 5 \\ 6 & 0 \\ 2 & 0 \end{pmatrix}$$

Şekil 14. Maxima ortamında belirlenmiş olan köşe noktaları

Şekil 14'ten görüleceği üzere uygun çözüm bölgesinin köşe noktaları, Şekil 13'deki gibi $A\left(\frac{6}{5}, \frac{16}{5}\right)$, $B(6,0)$, $C(2,0)$ olarak elde edilir. Dolayısı ile bulunan sonuçlar uyumludur.

Bulunan bu köşe noktalarından hangisinin amaç fonksiyonunu maksimum yaptığını belirlemek için noktalar amaç fonksiyonunda yerine yazılırsa

$$\text{A noktası için; } 1\left(\frac{6}{5}\right) + 2\left(\frac{16}{5}\right) = \frac{38}{5}$$

$$\text{B noktası için; } 1 \cdot (6) + 2(0) = 6$$

$$\text{C noktası için; } 1 \cdot (2) + 2(0) = 2$$

elde edilir. Bulunan bu değerlerden yola çıkarak problemi maksimum yapan noktanın A noktası ve problemin maksimum değerinin de $\frac{38}{5}$ olduğu görülmektedir. Bulunan bu sonuç maxima programında bulunan sonuçların gösterildiği Şekil 15 ile uyumludur.

Optimum noktalar:

$$\left[\frac{6}{5}, \frac{16}{5} \right]$$

Optimum değer

$$\frac{38}{5}$$

Şekil 15. Maxima ortamında belirlenmiş maksimum nokta ve maksimum değer

Örnek 18. M_1 ve M_2 metallerinin farklı karışımlarından oluşan A ve B alaşımlarının üretimindeki kullanılacak veriler aşağıdaki Tablo 59 da verilmiştir.

Verilen tabloya göre M_1 ve M_2 metallerinin A alaşımı içerisinde bulundukları miktarlar 0.5 er ton, B alaşımı içerisinde bulundukları miktarlar ise M_1 metali için 0.25 ton M_2 metali için 0.75 ton dur. M_1 metalinin günlük tedariki 10 ton, M_2

metalinin günlük tedarigi 15 ton dur. A alařımından elde edilen net kazanç 60 lira, B alařımından elde edilen net kazanç 50 liradır.

Tablo 59.Alařımdaki metal oranları

	A Alařımı	B Alařımı	Günlük tedarik
M ₁ metali	0.5	0.25	10
M ₂ metali	0.5	0.75	15
Net kazanç	60	50	

A alařımı yapmak için M₁ ve M₂ metallerinden x_1 'er ton alındığında elde edilen net kazanç $120x_1$ lira, B alařımı için M₁ metalinden x_2 ton M₂ metalinden $3x_2$ ton alındığında elde edilen net kazanç $200x_2$ olmaktadır. Böylelikle M₁ metalinden kullanılan miktar $x_1 + x_2$ ve M₂ metalinden kullanılan miktar $x_1 + 3x_2$ olmaktadır.

Bu verilere göre matematiksel model

$$\text{maks } z = 120x_1 + 200x_2$$

$$x_1 + x_2 \leq 10$$

$$x_1 + 3x_2 \leq 15$$

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

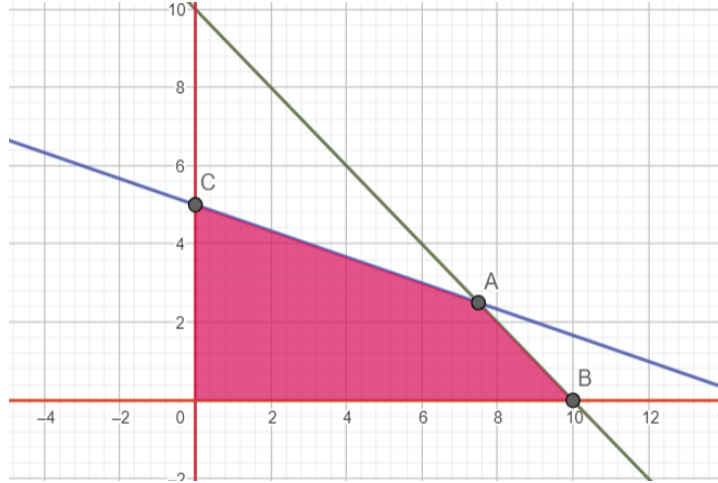
veya matrisel olarak

$$\text{maks } z = [120 \quad 200] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = Px$$

$$\begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 10 \\ 15 \end{bmatrix} \Rightarrow A_{ke}x \leq b_{ke}$$

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

řeklinde modellenabilir. Problemin grafik yöntemiyle řözümü yapıldığında řekil 16'daki grafik elde edilir.



Şekil 16. Örnek 18 kısıtlamaları için uygun bölge

Probleme ait veriler maxima ortamındaki grafik çözüm yöntemi için hazırlanan programa problemde büyük eşittir ve eşittir kısıtlamaları mevcut olmadığından aşağıdaki gibi girilir:

$P: matrix([120], [200]);$

$Ake: matrix([1,1], [1,3]);$

$Abe: matrix([[]]);$

$Aeq: matrix([[]]);$

$bke: matrix([10], [15]);$

$bbe: matrix([[]]);$

$beq: matrix([[]]);$

Bu girdiler ile hazırlanan program çalıştırılırsa köşe noktaları Şekil 17 deki gibi belirlenir.

Köşe Noktaları

15	5
2	2
10	0
0	5
0	0

Şekil 17. Maxima ortamında belirlenmiş olan köşe noktaları

Şekil 17'deki köşe noktalarının Şekil 16'daki uygun çözüm bölgesinin köşe noktaları olan A, B, C, O noktalarına karşılık geldiği kolaylıkla görülür. Bu köşe noktaları amaç fonksiyonunda yerine yazılarak hangisinin problemi maksimum yaptığı bulunur. Program tarafından problemi maksimum nokta ve problemin maksimum değeri Şekil 18'de gösterildiği gibi bulunur.

Optimum noktalar:

$$\left[\frac{15}{2}, \frac{5}{2} \right]$$

Optimum değer

1400

Şekil 18. Maksimum noktalar

Bu sonucun doğruluğu köşe noktalarının maksimizasyon probleminde yerine yazılması ile rahatlıkla görülebilir.

2.2. Simpleks yönteminin maxima'da programlanması

Bir optimizasyon probleminde tüm kısıtlamaları küçük eşit olduğu durumda klasik simpleks algoritması uygulanabilir. Bilindiği üzere giriş bölümünde klasik simpleks algoritması oluşturulmuştu. Bu probleme başlangıç simpleks tablosunun tüm elemanları girilerek çalıştırılabileceği belirtilmişti. Burada ise diğer yöntemlerle uyumlu olması açısından tüm kısıtlamaları küçük eşittir olan problem için yeni bir algoritma yazdırılmaktadır. Burada sadece P , Ake , bke , $degisken$ ve $yapay_degisken$ matrisleri girilerek algoritmanın başlangıç simpleks tablosunu kendisinin oluşturulması amaçlanmıştır. Bu sayede tüm bölümlerde girdiler birbirine benzetilerek oluşabilecek karmaşıklığın önüne geçilmiş olacaktır. Sonrasında oluşturulan başlangıç matrisi simpleks algoritmasına gönderilerek optimum sonuçların bulunması hedeflenmiştir. Bu yöntem için algoritma aşağıdaki gibi verilebilir. Ayrıca bu algoritma Ek 2'de ayrıntılı olarak gösterilmiştir

Algoritma 3. Simpleks yöntemi maxima programı

Adım 1. Küçük eşittir sınırlamasının satır ve sütun sayılarına bakılarak probleme eklenecek olan dolgu değişkenleri sayısı belirlenir.

Adım 2. Problem için başlangıç simpleks tablosu oluşturulur:

Adım 2.1. Girilen kısıtlama matrisleri başlangıç simpleks matrisinde ilgili yere atanır.

Adım 2.2. Girilen değişken isimleri ile problemin birinci satırı oluşturulur.

Adım 2.3. Belirlenen dolgu değişkenleri başlangıç simpleks tablosunda ilgili yere atanır. Not etmek gerekirse, kısıtlamalardaki dolgu değişkenlerinin katsayıları +1 olacak şekilde programdaki yapay değişkenlerin olduğu sütunlarda ilgili yerlere atanır.

Adım 3. Oluşturulan başlangıç simpleks tablosu önceden hazırlanan *simpleks_algoritma* programı kullanılarak çözdürülür.

Algoritma 3'ten görüleceği üzere girilen amaç fonksiyonu katsayıları, küçük eşittir ($\leq$) kısıtlamaları ve problemdeki değişken isimleri için sadece başlangıç simpleks tablosu oluşturulmuş olup çözüm ise bir önceki bölümdeki simpleks algoritması ile elde edilmiştir. Çünkü kısıtlamaların hepsi küçük eşittir olduğu durumda simpleks algoritması uygulanabilmektedir. Fakat burada direkt başlangıç simpleks tablosu girilerek simpleks algoritması uygulamak yerine programın başlangıç simpleks tablosunu kendisinin oluşturmasını sağlamamızdaki ana etken diğer programlar ile uyumlu olmasını amaçlanmıştır. Diğer programlardaki girdiler genel olarak bu şekilde alındığından bütünlüğün bozulmaması için bu program oluşturulmuştur. Şimdi bu programı bir örnek üzerinden inceleyelim.

Örnek 19. Giriş bölümünde Örnek 8'de ele alınan problemi algoritma yardımıyla çözelim. Örnek 8'de matematiksel model

$$\max z = 3000x_1 + 5000x_2$$

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

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

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

olarak elde edilmişti. Bu problemi, algoritmaya göre uyarlamak için matrisel formda ifade edilirse

$$\max z + [-3000 \quad -5000] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = z + Px$$

$$\begin{bmatrix} 1 & 0 \\ 0 & 2 \\ 3 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix} \Rightarrow A_{ke}x \leq b_{ke}$$

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

elde edilir. Dolayısı ile problemdeki girdiler

P: matrix([-3000, -5000]);

Ake: matrix([1,0], [0,2], [3,2]);

bke: matrix([4], [12], [18]);

degisken: matrix(["x_1", "x_2"]);

şeklinde girilmelidir. Örnek 8 ile uyumlu olması açısından yapay değişkenler örnek 8'deki standart formda olduğu için

yapay_degisken: matrix(["s_1", "s_2", "s_3"]);

olarak girilebilir. Bu girdiler ile yazılan program ilgili algoritmanın altına

simpleks_standart(P, Ake, bke, degisken, yapay_degisken);

yazılarak çalıştırılırsa aranan çözüm elde edilir.

Bu durumda başlangıç simpleks tablosunu programın kendisi oluşturarak problemin çözümü için simpleks algoritmasını kullanmaktadır. Simpleks algoritması da ilk olarak temel değişkenleri ve değerlerini belirleyerek ilgili yere yazdırmakta ve oluşan başlangıç simpleks tablosunu ekranda Şekil 19'daki gibi yazdırmaktadır.

Temel değişkenler:

(s_1 s_2 s_3)

Temel değişkenlerin değerleri:

(4 12 18)

Başlangıç simpleks tablosu:

	x_1	x_2	s_1	s_2	s_3	cözüm
z	-3000	-5000	0	0	0	0
s_1	1	0	1	0	0	4
s_2	0	2	0	1	0	12
s_3	3	2	0	0	1	18

Şekil 19. Örnek 19 için başlangıç simpleks tablosu

Şekil 19'da programın bulduğu başlangıç simpleks tablosu ile Örnek 8'deki Tablo 4'ün uyumlu olduğu görülür. Öte yandan simpleks algoritması her bir aşama

sonundaki simpleks tablolarını ve giren/çıkan değişkenleri ekranda yazdırmaktadır. İlk aşamada programa giren/çıkan değişkenler ve birinci aşama sonunda oluşan simpleks tablosu Şekil 20'deki gibi elde edilmiştir.

Giren değişken:

x_2

Çıkan değişken

s_2

[1.,. aşamanın sonucunda elde edilen simpleks tablosu:]

	x_1	x_2	s_1	s_2	s_3	cözüm
z	-3000	0	0	2500	0	30000
s_1	1	0	1	0	0	4
x_2	0	1	0	$\frac{1}{2}$	0	6
s_3	3	0	0	-1	1	6

Şekil 20. 1. aşama sonunda programın ekrana yazdırdığı simpleks tablosu

Şekil 20'den görüleceği üzere programın 1. aşama sonunda bulduğu simpleks tablosu ile örnek 8'deki Tablo 9 birbiri ile uyumludur. Fakat z satırında negatif eleman mevcut olduğundan program işlemler yapmaya devam edecektir. Bu durumda programa giren/çıkan değişkenler ile 2. aşama sonunda oluşan simpleks tablosu program tarafından Şekil 21'deki gibi elde edilmektedir.

Giren değişken:

x_1

Çıkan değişken

s_3

[2., aşamanın sonucunda elde edilen simpleks tablosu:]

	x_1	x_2	s_1	s_2	s_3	çözüm
z	0	0	0	1500	1000	36000
s_1	0	0	1	$\frac{1}{3}$	$-\left(\frac{1}{3}\right)$	2
x_2	0	1	0	$\frac{1}{2}$	0	6
x_1	1	0	0	$-\left(\frac{1}{3}\right)$	$\frac{1}{3}$	2

Şekil 21. 2. aşama sonunda programın ekrana yazdırdığı simpleks tablosu

1. aşamaya benzer şekilde 2. aşamada da programın bulduğu tablo ile örnek 8'deki Tablo 12 uyumludur. Bu durumda z satırında negatif eleman bulunmadığından program sonlandırılmakta ve temel değişkenler ve temel değişken değerleri ile optimum çözüm ekranda Şekil 22'deki gibi yazdırılmaktadır.

Temel değişkenler:

$(s_1 \ x_2 \ x_1)$

Temel değişkenlerin değerleri:

$(2 \ 6 \ 2)$

Temel değişkenlere karşılık gelen çözüm:

36000

Şekil 22. Örnek 19 için programın elde ettiği sonuçlar

Şekil 22'den dikkat edilirse program temel değişkenleri x_1, x_2, s_1 olarak belirlemiş ve bu temel değişken değerlerini ise sırasıyla 2, 6, 2 olarak bulmuştur. Optimum çözümü ise 36000 olarak belirlemiştir. Bu sonuçlar örnek 8'deki sonuçlar ile uyumludur.

2.3. *M* Yönteminin Maxima Ortamında Programlanması

Giriş bölümünde belirtildiği üzere grafik yöntemi iki karar değişkeni içeren problemlerin çözümünde kullanılabilirken klasik simpleks algoritması bütün kısıtlamaların küçük eşittir ($\leq$) olduğu problemlerde kullanılmaktadır. İki den fazla değişken içeren problemlerde problemin duali alınarak grafik yöntemi uygulayabilmek söz konusu olsa da ikiden fazla karar değişkeni içeren problemlerin çözümünde grafik yöntem kullanmak kullanışlı değildir. Öte yandan küçük eşittir ($\leq$) kısıtlaması yanında büyük eşittir ($\geq$) veya eşittir ($=$) içeren denklemlerde kullanılan yöntemlerden bir tanesi *M* yöntemidir. *M* yönteminin uygulaması giriş bölümünde detaylı bir şekilde anlatılmıştır. Bu kısımda *M* yönteminin maxima ortamına uygulaması anlatılacaktır.

Bu çalışmada *M* yöntemi için maxima algoritması aşağıda verilmiştir.

Algoritma 4. M yönteminin maxima algoritması:

Adım 1. Sınırlamaların satır ve sütun sayılarına bakılarak probleme eklenecek olan dolgu değişkenleri, artık değişkenler ve *M* değişkenleri sayısı belirlenir.

Adım 2. Problem için başlangıç simpleks tablosu oluşturulur:

Adım 2.1. Girilen kısıtlama matrisleri başlangıç simpleks matrisinde ilgili yere atanır.

Adım 2.2. Girilen değişken isimleri ile simpleks tablosunun satırı oluşturulur.

Adım 2.3. Belirlenen dolgu değişkenleri, artık değişkenler ve *M* değişkenleri başlangıç simpleks tablosunda ilgili yere atanır. Not etmek gerekirse, kısıtlamalardaki dolgu değişkenlerinin katsayıları +1, artık değişkenlerin katsayıları -1 ve *M* değişkenlerinin katsayıları olarak girilirken matrisin 2. satırındaki *M* değişkenleri katsayıları problem maksimizasyon problemi olacağı için +*M* olarak girileceği giriş bölümünden açıktır.

Adım 2.4. Yardımcı problemde başlangıçta temel değişken değerleri bilinmediğinden bunların yerine " " (çift tırnak) yazılır.

Adım 3. Oluşturulan başlangıç simpleks tablosunun çözüm satırı ile temel değişkenlere karşılık gelen çözümde uyumsuzluk olduğundan dolayı bu uyumsuzluk giderilir.

Adım 4. Simpleks algoritması uygulamaya uygun hale getirilen simpleks tablosu önceden hazırlanan *simpleks_algoritma* programı kullanılarak çözdürülür.

Yukarıda algoritması verilen yöntemin ayrıntılı ile verilmiş hali Ek 3'tedir. *M* yöntemi algoritmasından da görüleceği üzere bu programda giriş elemanları yardımıyla başlangıç simpleks tablosu oluşturulmuştur. Fakat Örnek 10'dan da görüldüğü üzere oluşturulan başlangıç simpleks tablosunda temel değişken değerleri ile çözüm değerinde uyumsuzluk oluşmaktadır. Programda ilk olarak bu uyumsuzluk giderilmekte ve bunun sonucunda oluşan tablo önceden hazırlanan simpleks algoritması ile çözdürülerek sonuçlar elde edilmektedir. Not etmek gerekirse *M* yönteminde problemdeki $\geq$ veya $=$ içeren ifadeler R_i değişkenleri eklenmektedir. Simpleks algoritmasının aşamalarında bu değişkenler sıfır değerini alarak temel dışı değişken olmaktadır. Bunun sağlanması için amaç fonksiyonunda bu değişkenlere bir *ceza katsayısı* verilir. Verilen bu *ceza katsayısı* *M* ile gösterilir. Bu değer yeterince büyük, pozitif değere sahip bir sayıdır. *M* değerinin yeterince büyük pozitif bir değer olmasını sağlamak için maxima yönteminde bu değer 10^5 olarak alınmıştır. Şimdi ilgili programı bir örnek üzerinde inceleyelim.

Örnek 20. Örnek 10'da (Coşkun, 2022).

$$\max z = x_1 + 2x_2$$

$$4x_1 + x_2 \geq 8$$

$$2x_1 + 3x_2 \leq 12$$

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

problemi giriş bölümünde *M* yöntemi ile çözülmüştü. Şimdi bu örneği program yardımıyla çözelim. Bu problem için hazırlanan maxima kodunda giriş matrisleri problemde eşitlik kısıtlamaları mevcut olmadığından

P: matrix([-1], [-2]);

Abe: matrix([4,1]);

Ake: matrix([2,3]);

Aeq: matrix([[]]);

bbe: matrix([8]);

bke: matrix([12]);

beq: matrix([[]]);

degisken: matrix(["x_1, x_2"]

$yapay_degisken: matrix(["s_1, s_2, R_1"])$

şeklinde girilmelidir. Yapay değişkenler ise örnek 10 ile uyumlu olması açısından s_1, s_2, R_1 olarak sisteme girilmiştir. Oluşturulan algoritma çalıştırılarak hazır hale getirildikten sonra program alt kısmında yukarıdaki giriş elemanları tanımlanıp $simpleks_M_yontemi(P, Ake, Abe, Aeq, bke, bbe, beq, degisken, yapay_degisken);$ yazılarak program çalıştırıldığında başlangıç simpleks tablosu ekranda Şekil 23'deki gibi yazdırılır.

başlangıç simpleks tablosu

	x_1	x_2	s_1	s_2	R_1	çözüm
z	-1	-2	0	0	100000	0
	4	1	-1	0	1	8
	2	3	0	1	0	12

Şekil 23. Örnek 20 için başlangıç simpleks tablosu

Başlangıçtaki simpleks tablosuna göre temel değerler ve temel değişken değerleri ise Şekil 24'teki gibi elde edilir.

Temel değişkenler:

$$(R_1 \ s_2)$$

Temel değişkenlerin değerleri:

$$(8 \ 12)$$

Şekil 24. Temel değişkenler ve temel değişkenlerin değerleri

Dikkat edilirse $R_1 = 8, s_2 = 12$ dir. Bu durumda temel çözüm $z = x_1 + 2x_2 - MR_1 = -8M$ şeklinde oluşur. Ancak Şekil 23'de yer alan başlangıç simpleks tablosu incelendiğinde ise çözüm 0 olarak görünmektedir. Bu uyumsuzluk z satırında R_1 katsayısının sıfır olmamasından ortaya çıkmaktadır. Bu durum R_1 'in yerine uygun kısıt denklemleri kullanılarak giderilebilir. Bu durumda giriş bölümünde yapıldığı gibi program kendi içerisinde R_1 satırını $-M$ ile çarpıp z satırı ile toplayarak

uyumsuzluğu gidermektedir. Bu durumda uyumsuzluğun giderildiği simpleks tablosu Şekil 25’deki gibi oluşturulur.

uyumsuzluk giderildiği simpleks tablosu

$$\begin{array}{c|cccccc} & x_1 & x_2 & s_1 & s_2 & R_1 & \text{çözüm} \\ \hline z & -400001 & -100002 & 100000 & 0 & 0 & -800000 \\ \hline & 4 & 1 & -1 & 0 & 1 & 8 \\ \hline & 2 & 3 & 0 & 1 & 0 & 12 \end{array}$$

Şekil 25. Örnek 20 için uyumsuzluğun giderildiği Simpleks Tablosu

Şekil 25’ten dikkat edilirse Örnek 10’daki tablolarda M değişkeni mevcut iken oluşturulan programda bu değişken yerine mutlak değerce çok büyük sayılar mevcuttur. Bunun nedeni programda M değeri yerine 10^5 olarak seçilmesidir. Eğer örnek 10 Tablo 22’de M yerine seçilen sayı yazıldığında iki tablonun birbiri ile uyumlu olduğu görülür.

Program uyumsuzluğu giderdikten sonra ilgili tablo simpleks algoritması uygulamaya hazır hale geldiğinden bundan sonraki kısım *simpleks_algoritma* programı tarafından yürütülmektedir. Bu durumda ilgili algortima ilk olarak başlangıçtaki temel değişkenleri belirleyerek Şekil 25’teki boş kısımlara yazmaktadır. Bu durumda Şekil 26 elde edilir.

Başlangıç simpleks tablosu:

$$\begin{array}{c|cccccc} & x_1 & x_2 & s_1 & s_2 & R_1 & \text{çözüm} \\ \hline z & -400001 & -100002 & 100000 & 0 & 0 & -800000 \\ \hline R_1 & 4 & 1 & -1 & 0 & 1 & 8 \\ \hline s_2 & 2 & 3 & 0 & 1 & 0 & 12 \end{array}$$

Şekil 26. Temel değişkenleri yerine yazıldığı başlangıç simpleks tablosu

İkinci aşamada programa giren ve çıkan değişkenler belirlenerek Gauss Jordan algoritması ile birinci aşama sonucunda oluşan simpleks tablosu Şekil 27’deki gibi elde edilir.

Giren değişken:

x_1

Çıkan değişken

R_1

[1., aşamanın sonucunda elde edilen simpleks tablosu:]

$$\begin{array}{c|cccccc} & x_1 & x_2 & s_1 & s_2 & R_1 & \text{çözüm} \\ \hline z & 0 & -\left(\frac{7}{4}\right) & -\left(\frac{1}{4}\right) & 0 & \frac{400001}{4} & 2 \\ x_1 & 1 & \frac{1}{4} & -\left(\frac{1}{4}\right) & 0 & \frac{1}{4} & 2 \\ s_2 & 0 & \frac{5}{2} & \frac{1}{2} & 1 & -\left(\frac{1}{2}\right) & 8 \end{array}$$

Şekil 27. Birinci aşama sonunda oluşan simpleks algortiması

Şekil 27’de elde edilen tablo ile Örnek 10’daki Tablo 24’te M yerine 10^5 yazılması elde edilen tablonun uyumlu olduğu görülür. Öte yandan z satırında negatif eleman bulunduğundan algoritma tekrarlanmalıdır. Örnek 10’da ikinci simpleks algoritmasında giren ve çıkan değişkenler sırası ile x_2, s_2 olarak elde edilmişti. Oluşturulan program da benzer sonuçları elde etmiş olup ikinci aşama sonunda elde edilen simpleks tablosu Şekil 28’de gösterilmiştir.

Giren değişken:

x_2

Çıkan değişken

s_2

[2., aşamanın sonucunda elde edilen simpleks tablosu:]

$$\begin{array}{c|cccccc} & x_1 & x_2 & s_1 & s_2 & R_1 & \text{çözüm} \\ \hline z & 0 & 0 & \frac{1}{10} & \frac{7}{10} & \frac{999999}{10} & \frac{38}{5} \\ x_1 & 1 & 0 & -\left(\frac{3}{10}\right) & -\left(\frac{1}{10}\right) & \frac{3}{10} & \frac{6}{5} \\ x_2 & 0 & 1 & \frac{1}{5} & \frac{2}{5} & -\left(\frac{1}{5}\right) & \frac{16}{5} \end{array}$$

Şekil 28. Simpleks algoritmasının 2.aşaması

Şekil 28’de elde edilen tablo ile Tablo 26’da elde edilen tablonun M yerine 10^5 yazıldığında uyumlu olduğu gözlemlenir. Öte yandan Şekil 28 incelendiğinde z

satırında negatif eleman kalmadığı görülmektedir. Dolayısıyla elde edilen tablo nihai tablodur. Bu tablodan yola çıkarak temel değişkenler, temel değişken değerleri ve optimum çözüm program tarafından Şekil 29'daki gibi elde edilir.

Temel değişkenler:

$$\begin{pmatrix} x_1 & x_2 \end{pmatrix}$$

Temel değişkenlerin değerleri:

$$\begin{pmatrix} 6 & 16 \\ 5 & 5 \end{pmatrix}$$

Temel değişkenlere karşılık gelen çözüm:

$$\frac{38}{5}$$

Şekil 29. Temel değişkenler, temel değişken değerleri ile optimum çözüm

Bulunan çözüm ile temel değişkenler ve değerlerinin giriş bölümündeki Örnek 10 çözümüyle uyumlu olarak gözlemlenmektedir.

Örnek 21.(Özturan, 2022)

$$\min z = 4x_1 + x_2$$

$$3x_1 + x_2 = 3$$

$$4x_1 + 3x_2 \geq 6$$

$$x_1 + 2x_2 \leq 4$$

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

görüldüğü üzere verilen problem bir minimizasyon problemidir. Minimizasyon problemlerini program yardımıyla çözebilmek için öncelikle problem maksimizasyon problemine çevrilir. Bu durumda

$$\max z = -4x_1 - x_2$$

$$3x_1 + x_2 = 3$$

$$4x_1 + 3x_2 \geq 6$$

$$x_1 + 2x_2 \leq 4$$

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

problemi elde edilir. Bu problem için hazırlanan maxima kodu ile giriş elemanları

$$P: \text{matrix}([4], [1]);$$

$$Abe: \text{matrix}([4,3]);$$

```

Ake: matrix([1,2]);
Aeq: matrix([3,1]);
bbe: matrix([6]);
bke: matrix([4]);
beq: matrix([3]);

degisken: matrix(["x_1, x_2"]);
yapay_degisken: matrix(["s_1, s_2, R_1, R_2"]);

```

olarak tanımlanabilir. Bu durumda

`simpleks_M(P, Ake, Abe, Aeq, bke, bbe, beq, degisken, yapay_degisken);`
yazılarak çalıştırıldığında ekranda başlangıç simpleks tablosu Şekil 30' daki gibi elde edilir.

başlangıç simpleks tablosu

	x_1	x_2	s_1	s_2	R_1	R_2	çözüm
z	4	1	0	0	100000	100000	0
	3	1	0	0	1	0	3
	4	3	-1	0	0	1	6
	1	2	0	1	0	0	4

Şekil 30. Başlangıç simpleks tablosu

Şekil 30 için problemdeki temel değerler ve temel değişken değerleri ise Şekil 31'deki gibi elde edilir.

Temel değişkenler:
 $(R_1 \ R_2 \ s_2)$
Temel değişkenlerin değerleri:
 $(3 \ 6 \ 4)$

Şekil 31. Temel değişkenler ve temel değişkenlerin değerleri

Öte yandan bu temel çözümler için problemin çözümü z satırına yapay değişkenlerin de eklenmesiyle

$$z = -4x_1 - x_2 - MR_1 - MR_2 = -3M - 6M = -9M$$

olmalıydı. Fakat Şekil 30'da bu değer sıfır (0) olarak görülmektedir. Program içerisinde bu uyumsuzluk giderilerek Şekil 32'deki tablo elde edilir.

uyumsuzluk giderildiği simpleks tablosu

	x_1	x_2	s_1	s_2	R_1	R_2	çözüm
z	-699996	-399999	100000	0	0	0	-900000
	3	1	0	0	1	0	3
	4	3	-1	0	0	1	6
	1	2	0	1	0	0	4

Şekil 32.Uyumsuzluğun giderildiği simpleks Tablosu

Bu aşamadan sonra problem simpleks algoritmasına uygun hale geldiğinden simpleks_algoritma programı ile problem çözülür. Bu durumda giren/çıkan değişkenler ile birinci aşama sonunda elde edilen simpleks tablosu Şekil 33'deki gibidir.

Giren değişken:

x_1

Çıkan değişken

R_1

[1., aşamanın sonucunda elde edilen simpleks tablosu:]

	x_1	x_2	s_1	s_2	R_1	R_2	çözüm
z	0	-166667	100000	0	233332	0	-200004
x_1	1	$\frac{1}{3}$	0	0	$\frac{1}{3}$	0	1
R_2	0	$\frac{5}{3}$	-1	0	$-\left(\frac{4}{3}\right)$	1	2
s_2	0	$\frac{5}{3}$	0	1	$-\left(\frac{1}{3}\right)$	0	3

Şekil 33. Simpleks algoritmasının 1.aşaması

Bu durumda program temel değişken ve değerlerini ekranda yazdırarak bunlara karşılık gelen optimum sonucu da kullanıcıya geri göndermektedir. Fakat Şekil 33'ten de görüldüğü üzere z satırında negatif eleman bulunduğu için algoritma

program tarafından tekrarlanmış olup giren/çıkan değişkenler ile ikinci aşama sonunda oluşan simpleks tablosu Şekil 34'deki gibi ekranda yazdırılmıştır.

Giren değişken:

x_2

Çıkan değişken

R_2

[2,. aşamanın sonucunda elde edilen simpleks tablosu:]

	x_1	x_2	s_1	s_2	R_1	R_2	çözüm
z	0	0	$-\left(\frac{1}{5}\right)$	0	$\frac{499992}{5}$	$\frac{500001}{5}$	$-\left(\frac{18}{5}\right)$
x_1	1	0	$\frac{1}{5}$	0	$\frac{3}{5}$	$-\left(\frac{1}{5}\right)$	$\frac{3}{5}$
x_2	0	1	$-\left(\frac{3}{5}\right)$	0	$-\left(\frac{4}{5}\right)$	$\frac{3}{5}$	$\frac{6}{5}$
s_2	0	0	1	1	1	-1	1

Şekil 34. Simpleks algoritmasının 2.aşaması

Şekil 34'ten dikkat edilirse 2.aşamanın sonunda da z satırında negatif eleman bulunduğundan algoritma bir kez daha tekrarlanmalıdır. Bu durumda program tarafından elde edilen üçüncü simpleks tablosu Şekil 35'teki gibidir.

Giren değişken:

s_1

Çıkan değişken

s_2

[3,. aşamanın sonucunda elde edilen simpleks tablosu:]

	x_1	x_2	s_1	s_2	R_1	R_2	çözüm
z	0	0	0	$\frac{1}{5}$	$\frac{499993}{5}$	100000	$-\left(\frac{17}{5}\right)$
x_1	1	0	0	$-\left(\frac{1}{5}\right)$	$\frac{2}{5}$	0	$\frac{2}{5}$
x_2	0	1	0	$\frac{3}{5}$	$-\left(\frac{1}{5}\right)$	0	$\frac{9}{5}$
s_1	0	0	1	1	1	-1	1

Şekil 35. Simpleks algoritmasının 3.aşaması

Şekil 35'ten görüldüğü üzere z satırında negatif eleman kalmamıştır. Dolayısıyla elde edilen tablo nihai tablodur. Bu tablodan yola çıkarak temel değişkenler, temel değişken değerleri ve optimum çözüm program tarafından Şekil 36'daki gibi elde edilir.

Temel değişkenler:

$$\begin{pmatrix} x_1 & x_2 & s_1 \end{pmatrix}$$

Temel değişkenlerin değerleri:

$$\begin{pmatrix} \frac{2}{5} & \frac{9}{5} & 1 \end{pmatrix}$$

Temel değişkenlere karşılık gelen çözüm:

$$-\left(\frac{17}{5}\right)$$

Şekil 36. Temel değişkenler, değerleri ve optimum çözüm

Görüleceği üzere optimum çözüm $\frac{-17}{5}$ olarak görülmektedir. Problemin başlangıçta bir minimizasyon problemi olup maksimizasyon problemine çevirdiğimiz için minimizasyon probleminin çözümü $\frac{17}{5}$ olmalıdır.

2.4. İki aşamalı simpleks yönteminin maxima'da programlanması

Küçük eşittir ($\leq$) kısıtlamasının yanında büyük eşittir ($\geq$) veya eşittir ($=$) içeren kısıtlama denklemlerinde kullanılan yöntemlerden biri M yöntemidir. Öte yandan giriş bölümünde de belirtildiği üzere M yönteminin dışında iki aşamalı simpleks yöntemi de bu problemlerde kullanılmaktadır. İki aşamalı simpleks yöntemi M yöntemindeki M sabitini tamamen ortadan kaldırarak yuvarlama hatalarında oluşan sorunları ortadan kaldırmak için oluşturulmuştur. Bu yöntemde ilk olarak yardımcı bir problem oluşturularak bu probleme simpleks yöntemi uygulanır. Uygulanan simpleks aşamalarının sonunda çözüm kısmı sıfıra eşit olursa ikinci aşamaya geçilerek problem çözülür.

Giriş bölümünde iki aşamalı simpleks yöntemine değinilerek örnek üzerinden uygulaması yapılmıştı. Bu kısımda iki aşamalı simpleks yönteminin maxima

algoritması oluşturularak bulunan sonuçlar örnekler üzerinden test edilmektedir. Maxima ortamında hazırlanan iki aşamalı simpleks algoritması aşağıda verilmiştir.

Algoritma 4. İki aşamalı simpleks yönteminin maxima algoritması:

Adım 1. Sınırlamaların satır ve sütun sayılarına bakılarak probleme eklenecek olan dolgu değişkenleri, artık değişkenler ve R değişkenleri sayısı belirlenir.

Adım 2. Problemin yardımcı problemi oluşturulur:

Adım 2.1. Girilen kısıtlama matrisleri yardımcı problem matrisinde ilgili yere atanır.

Adım 2.2. Girilen değişken isimleri ile simpleks tablosunun ilgili satırı oluşturulur.

Adım 2.3. Belirlenen dolgu değişkenleri, artık değişkenler ve R değişkenleri yardımcı problem matrisinde ilgili yere atanır. Not etmek gerekirse, kısıtlamalardaki dolgu değişkenlerinin katsayıları $+1$, artık değişkenlerin katsayıları -1 ve R değişkenlerinin katsayıları olarak girilirken matrisin 2. satırındaki (z satırındaki) R değişkenleri katsayıları problem minimizasyon problemi olacağı için $+1$ olarak girileceği giriş bölümünden açıktır. Ayrıca problemin kısıtlamalarında, R değişkeninin eklendiği kısımların katsayıları $+1$ olacağından bu değerler R sütununda ilgili yere atanır.

Adım 2.4. Yardımcı problemde başlangıçta temel değişken değerleri bilinmediğinden bunların yerine " " (çift tırnak) yazılır.

Adım 3. Oluşturulan yardımcı problemin simpleks tablosunun çözüm satırı ile temel değişkenlere karşılık gelen çözümde uyumsuzluk olduğundan dolayı bu uyumsuzluk giderilir.

Adım 4. Simpleks algoritması uygulamaya uygun hale getirilen yardımcı problem simpleks tablosu önceden hazırlanan *simpleks_algoritma* programı kullanılarak çözdürülür.

Adım 5. Eğer uygulanan simpleks algoritması sonucunda çözüm değeri sıfıra (0) eşitse ikinci aşamaya geçilir:

Adım 5.1. Esas problemin amaç fonksiyonun katsayıları son simpleks tablosunun z satırında ilgili yerlere atanır.

Adım 5.2. Oluşturulan simpleks tablosunda uyumsuzluğun olup olmadığı kontrol edilir.

Adım 5.2.1. Eğer temel değişken değerleri ile çözüm arasında bir uyumsuzluk varsa giderilir. Ardından ikinci aşama başlangıç matrisi ekranda yazdırılır.

Adım 5.2.2. Simpleks algoritması uygulamaya uygun hale getirilen ikinci aşama simpleks tablosu önceden hazırlanan *simpleks_algoritma* programı kullanılarak çözdürülür.

Yukarıdaki algorithmandan da görüleceği üzere ilk olarak yardımcı problem için simpleks tablosu oluşturulmuş ve simpleks algoritması ile çözdürülmüştür. Simpleks algoritması sonucunda oluşan simpleks tablosundaki çözüm değeri sıfıra eşit ise algoritma ikinci aşamaya geçmektedir. İkinci aşamanın ilk kısmında ise temel değişken değerleri ile çözüm değeri arasında bir uyumsuzluk varsa ilk olarak o uyumsuzluk giderilerek problem simpleks algoritması uygulamaya uygun hale getirilir. Son aşamada ise önceden hazırlanan simpleks algoritması ile aranan çözüm belirlenmektedir. Algoritması yukarıda verilen iki aşamalı simpleks yöntemin maxima ortamında aktarılması Ek 4'te verilmiştir.

Not etmek gerekirse hazırlanan simpleks algoritması maksimizasyon problemlerini çözdürülecek şekilde hazırlanmıştı. Fakat giriş bölümünden bilindiği üzere iki aşamalı simpleks yönteminde yardımcı problem bir minimizasyon problemi olup Örnek 11'deki birinci aşama işlemleri minimizasyon problemine göre yürütülmüştü. Fakat simpleks algoritması maksimizasyon problemine göre hazırlandığı için birinci aşamada oluşturulan yardımcı problem maksimizasyon problemine çevrilerek sisteme girilmelidir. Adım 2.3'ten görüleceği üzere z satırında R değişkeninin katsayıları olarak $+1$ olarak problem tarafından oluşturulmuştur. Öte yandan bu değer giriş bölümünde Örnek 11'de Tablo 27'de -1 olarak girilmiştir. Bu durumdan dolayı giriş bölümündeki örneklerdeki simpleks tabloları ile programın birinci aşamada elde edeceği simpleks tablolarında küçük farklılıkların oluşmasına neden olacağına dikkat ediniz.

Şimdi hazırlanan maxima ortamında hazırlanan programı örnekler üzerinden test edelim.

Örnek 21. Örnek 11 (Coşkun, 2022)

$$\begin{aligned}\max z &= x_1 + 2x_2 \\ 4x_1 + x_2 &\geq 8 \\ 2x_1 + 3x_2 &\leq 12 \\ x_1 &\geq 0, x_2 \geq 0\end{aligned}$$

problemi iki aşamalı yöntem ile çözülmüştü. Şimdi bu örneği program yardımıyla çözelim:

Hazırlanan maxima algoritmasında giriş elemanları problemin sınırlamalarında eşitlik sınırlaması mevcut olmadığından

```
P: matrix([1], [2]);
Abe: matrix([4,1]);
Ake: matrix([2,3]);
Aeq: matrix([[]]);
bbe: matrix([8]);
bke: matrix([12]);
beq: matrix([[]]);
degisken: matrix(["x_1, x_2"]
yapay_degisken: matrix(["s_1, s_2, R_1"]
```

şeklinde girilmelidir. Yapay değişkenler kısmının ise giriş bölümüyle uyumlu olacak şekilde girildiğine dikkat edelim. Bu giriş elemanları ile ilgili algoritma

simpleks_iki_asama(P, Ake, Abe, Aeq, bke, bbe, beq, degisken, yapay_degisken);

yazılarak çalıştırıldığında ekranda birinci aşamadaki problemin başlangıç simpleks tablosu Şekil 37'deki gibi yazdırılır.

başlangıç simpleks tablosu

	x_1	x_2	s_1	s_2	R_1	çözüm
z	0	0	0	0	1	0
	4	1	-1	0	1	8
	2	3	0	1	0	12

Şekil 37. Örnek 21 için birinci aşama problemin başlangıç simpleks tablosu

Şekil 37’de programın elde ettiği birinci aşama başlangıç simpleks tablosu ile giriş bölümünde Örnek 11’deki Tablo 27 karşılaştırıldığında z satırı, R_1 sütunu programda $+1$ iken ilgili örnekte -1 olarak alınmıştı. Bunun sebebi yukarıda da belirtildiği üzere örnekteki aşamalar minimizasyon problemine göre yürütülmekte iken hazırlanan programdaki algoritmalar maksimizasyon problemine göre yürütülmektedir. Öte yandan Şekil 37’de temel değişkenleri programın kendisi belirlediği için ilk tabloda bu değişkenlerin yeri boş bırakılmıştır. Sonrasında program temel değişken değerleri ile çözüm değerinde oluşan uyumsuzluğu gidererek yeni birinci aşama başlangıç tablosunu Şekil 38’deki gibi belirlemektedir.

uyumsuzluk giderildiği simpleks tablosu

	x_1	x_2	s_1	s_2	R_1	çözüm
z	-4	-1	1	0	0	-8
	4	1	-1	0	1	8
	2	3	0	1	0	12

Şekil 38. Birinci aşamada uyumsuzluğun giderildiği başlangıç simpleks tablosu

Şekil 38’den görüleceği üzere temel değişkenleri simpleks algoritmasının kendisi belirlediği için bu alanlar boş bırakılmıştır. Uyumsuzluğun giderilmesi ile ilk aşama simpleks tablosu simpleks algoritmasını uygulamaya hazır hale getirilmiştir. Bu durumda program *simpleks_algoritma* programı yükleyerek çalıştırmakta ve temel değişkenleri ve değerlerini belirleyerek tabloda boş bırakılan kısma yazdırmaktadır. Bu durumda simpleks algoritmasının uygulanacağı başlangıç simpleks tablosu Şekil 39’daki gibi elde edilir.

Temel değişkenler:

$$(R_1 \ s_2)$$

Temel değişkenlerin değerleri:

$$(8 \ 12)$$

Başlangıç simpleks tablosu:

$$\begin{array}{c|cccccc} & x_1 & x_2 & s_1 & s_2 & R_1 & \text{çözüm} \\ \hline z & -4 & -1 & 1 & 0 & 0 & -8 \\ R_1 & 4 & 1 & -1 & 0 & 1 & 8 \\ s_2 & 2 & 3 & 0 & 1 & 0 & 12 \end{array}$$

Şekil 39. Temel değişken değerleri ve başlangıç simpleks tablosu

Şekil 39 ve giriş bölümündeki Tablo 28 karşılaştırıldığında tek farkın z satırındaki değişkenlerin katsayılarının işareti ve çözümün işareti olduğu görülür. Bunun nedeni de yukarıda bahsedildiği üzere program maksimizasyon problemini çözerken giriş bölümünde bahsedilen örnekte minimizasyon probleminin çözdürülmesidir. Daha sonra simpleks algoritması ile giren ve çıkan değişkenler belirlenerek birinci aşama sonunda elde edilen simpleks tablosu Şekil 40'taki gibidir.

Giren değişken:

$$x_1$$

Çıkan değişken

$$R_1$$

[1.,. aşamanın sonucunda elde edilen simpleks tablosu:]

$$\begin{array}{c|cccccc} & x_1 & x_2 & s_1 & s_2 & R_1 & \text{çözüm} \\ \hline z & 0 & 0 & 0 & 0 & 1 & 0 \\ x_1 & 1 & \frac{1}{4} & -\left(\frac{1}{4}\right) & 0 & \frac{1}{4} & 2 \\ s_2 & 0 & \frac{5}{2} & \frac{1}{2} & 1 & -\left(\frac{1}{2}\right) & 8 \end{array}$$

Şekil 40. Yardımcı probleme simpleks algoritması uygulandığında oluşan tablo

Şekil 40'taki tablo ile Örnek 11'deki Tablo 29 benzer sonuçlar verdiği görülür. Öte yandan z satırında negatif eleman bulunmadığından simpleks algoritması program tarafından sonlandırılır. Ayrıca çözüm değeri sıfır (0) olarak elde edildiğinden dolayı birinci aşamanın sona erdiği anlamına gelir. Bu durumda program ikinci aşamaya geçtiğini ekranda “ikinci aşamaya geçiliyor” yazarak

kullanıcıya bildirmektedir. Bundan sonra program z satırına başlangıçtaki esas problemin katsayılarını yerleştirerek ikinci aşamanın başlangıç simpleks tablosunu Şekil 41’deki gibi ekranda yazdırmaktadır.

İkinci aşama başlangıç matrisi:

$$\begin{pmatrix} & x_1 & x_2 & s_1 & s_2 & \text{çözüm} \\ z & -1 & -2 & 0 & 0 & 0 \\ x_1 & 1 & \frac{1}{4} & -\left(\frac{1}{4}\right) & 0 & 2 \\ s_2 & 0 & \frac{5}{2} & \frac{1}{2} & 1 & 8 \end{pmatrix}$$

Şekil 41. İkinci aşama başlangıç simpleks tablosu

Şekil 41’den görüleceği üzere temel değişken değerleri ile çözüm arasında bir uyumsuzluk söz konusudur. Program ilk olarak bu uyumsuzluğu gidererek simpleks algoritması uygulamaya hazır hale gelen Şekil 42’deki tabloyu oluşturur.

uyumsuzluğun giderildiği matris

$$\begin{pmatrix} & x_1 & x_2 & s_1 & s_2 & \text{çözüm} \\ z & 0 & -\left(\frac{7}{4}\right) & -\left(\frac{1}{4}\right) & 0 & 2 \\ x_1 & 1 & \frac{1}{4} & -\left(\frac{1}{4}\right) & 0 & 2 \\ s_2 & 0 & \frac{5}{2} & \frac{1}{2} & 1 & 8 \end{pmatrix}$$

Şekil 42. İkinci aşamada uyumsuzluğun giderildiği başlangıç simpleks tablosu

Şekil 42’deki uyumsuzluğun giderildiği tablo ile Örnek 11’deki Tablo 31 birbiri ile aynıdır.

Problem simpleks algoritması uygulamaya uygun hale getirilmiş olup ilgili algoritmanın yüklenerek çalıştırılması ile giren ve çıkan değişkenler ile birinci aşama sonucunda oluşan simpleks tablosu Şekil 43’teki gibi elde edilir.

Giren değişken:

x_2

Çıkan değişken

s_2

[1., aşamanın sonucunda elde edilen simpleks tablosu:]

$$\begin{array}{c|ccccc} & x_1 & x_2 & s_1 & s_2 & \text{çözüm} \\ \hline z & 0 & 0 & \frac{1}{10} & \frac{7}{10} & \frac{38}{5} \\ x_1 & 1 & 0 & -\left(\frac{3}{10}\right) & -\left(\frac{1}{10}\right) & \frac{6}{5} \\ x_2 & 0 & 1 & \frac{1}{5} & \frac{2}{5} & \frac{16}{5} \end{array}$$

Şekil 43. İkinci aşama için simpleks algoritması ile oluşan simpleks tablosu

Şekil 43'ten görüleceği üzere z satırında negatif eleman mevcut değildir. Dolayısı ile simpleks algoritması sonlandırılacaktır. Bu durumda oluşan temel değişkenler ve değerleri ile çözüm ekranda Şekil 44'teki gibi yazdırılmaktadır.

Temel değişkenler:

$$\begin{pmatrix} x_1 & x_2 \end{pmatrix}$$

Temel değişkenlerin değerleri:

$$\begin{pmatrix} \frac{6}{5} & \frac{16}{5} \end{pmatrix}$$

Temel değişkenlere karşılık gelen çözüm:

$$\frac{38}{5}$$

Şekil 44. Problemin temel değişkenlerine karşılık gelen çözümü

Şekil 44'ten elde edilen simpleks tablosu ile aranan çözüm Örnek 11 ile uyumludur. Dolayısı ile oluşturulan algoritma ilgili problemi çözmede doğru sonuç vermektedir.

Örnek 22. (Ozturan, 2022) M yönteminde ele alınan

$$\min z = 4x_1 + x_2$$

$$3x_1 + x_2 = 3$$

$$4x_1 + 3x_2 \geq 6$$

$$x_1 + 2x_2 \leq 4$$

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

problemini göz önüne alalım. Bu problem için yardımcı problem

$$\min z = R_1 + R_2$$

$$3x_1 + x_2 + R_1 = 3$$

$$4x_1 + 3x_2 - s_1 + R_2 = 26$$

$$x_1 + 2x_2 + s_2 = 4$$

$$x_1 \geq 0, x_2 \geq 0, s_1 \geq 0, s_2 \geq 0, R_1 \geq 0, R_2 \geq 0$$

şeklindedir. Bu durumda iki aşamalı simpleks algoritmasında da problemin giriş elemanları M yöntemindeki gibi girilecektir. Daha sonrasında programı çalıştırmak için

simpleks_iki_asama(P, Ake, Abe, Aeq, bke, bbe, beq, degisken, yapay_degisken); yazılırsa ekrana ilk olarak başlangıç simpleks tablosu ve uyumsuzluğun giderildiği simpleks tabloları Şekil 45'deki gibi ekranda yazdırılır.

başlangıç simpleks tablosu

	x_1	x_2	s_1	s_2	R_1	R_2	çözüm
z	0	0	0	0	1	1	0
	3	1	0	0	1	0	3
	4	3	-1	0	0	1	6
	1	2	0	1	0	0	4

uyumsuzluk giderildiği simpleks tablosu

	x_1	x_2	s_1	s_2	R_1	R_2	çözüm
z	-7	-4	1	0	0	0	-9
	3	1	0	0	1	0	3
	4	3	-1	0	0	1	6
	1	2	0	1	0	0	4

Şekil 45. Örnek 22 için oluşan tablolar

Şekil 45'ten görüldüğü üzere başlangıç temel değişkenler ve değerleri sırasıyla $s_2 = 4, R_1 = 3, R_2 = 6$ olup bu değerlere karşılık gelen çözüm $z = -R_1 - R_2 = -3 - 6 = -9$ olması gerekirken Şekil 45'deki birinci tabloda sıfır (0) olarak

görülmektedir. Bu uyumsuzluğun Şekil 45'teki ikinci tabloda giderildiği görülmektedir. Öte yandan burada elde edilen temel değişkenler ve değerleri programda da Şekil 46'daki gibi ekranda yazdırılmaktadır.

Temel değişkenler:

$$(R_1 \ R_2 \ s_2)$$

Temel değişkenlerin değerleri:

$$(3 \ 6 \ 4)$$

Şekil 46. Yardımcı problem başlangıç tablosu için temel değişkenler ve değerleri

Simpleks algoritması uygulamaya hazır hale getirilen tabloya simpleks algoritması uygulandığında giren ve çıkan değişkenler ile birinci aşama sonunda oluşan simpleks tablosu Şekil 47'deki gibidir.

Giren değişken:

x_1

Çıkan değişken

R_1

[1.,. aşamanın sonu:]

$$\begin{array}{c|ccccccc} & x_1 & x_2 & s_1 & s_2 & R_1 & R_2 & \text{çözüm} \\ \hline z & 0 & -\left(\frac{5}{3}\right) & 1 & 0 & \frac{7}{3} & 0 & -2 \\ x_1 & 1 & \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 & 1 \\ R_2 & 0 & \frac{5}{3} & -1 & 0 & -\left(\frac{4}{3}\right) & 1 & 2 \\ s_2 & 0 & \frac{5}{3} & 0 & 1 & -\left(\frac{1}{3}\right) & 0 & 3 \end{array}$$

Şekil 47. Yardımcı problem için birinci simpleks algoritması ile elde edilen tablo

Şekil 47'den görüleceği üzere z satırında negatif eleman mevcuttur. Dolayısı ile yardımcı probleme ikinci bir simpleks algoritması uygulanması gerekmektedir. Bu durumda giren/çıkan değişkenler ile ikinci aşama sonucunda oluşan simpleks tablosu Şekil 48'deki gibidir.

Giren değişken:

x_2

Çıkan değişken

R_2

[2., aşamanın sonu:]

$$\begin{array}{c|ccccccc} & x_1 & x_2 & s_1 & s_2 & R_1 & R_2 & \text{çözüm} \\ \hline z & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ x_1 & 1 & 0 & \frac{1}{5} & 0 & \frac{3}{5} & -\left(\frac{1}{5}\right) & \frac{3}{5} \\ x_2 & 0 & 1 & -\left(\frac{3}{5}\right) & 0 & -\left(\frac{4}{5}\right) & \frac{3}{5} & \frac{6}{5} \\ s_2 & 0 & 0 & 1 & 1 & 1 & -1 & 1 \end{array}$$

Şekil 48. Yardımcı problem için ikinci simpleks algoritması ile elde edilen tablo

Şekil 48'den görüleceği üzere ikinci aşamada sonunda elde edilen simpleks tablosunun z satırında negatif eleman bulunmamaktadır. Ayrıca çözüm değeri sıfır (0) olduğundan problemde ikinci aşamaya geçilecektir. Bu durum ekranda *ikinci aşamaya geçiliyor* yazdırılmaktadır.

İkinci aşamada z satırına ana problemin katsayıları yazdırılarak başlangıçtaki uyumsuzluk giderilirse ekranda Şekil 49'daki sonuçlar yazdırılır.

İkinci aşama başlangıç matrisi:

$$\begin{array}{c|cccccc} & x_1 & x_2 & s_1 & s_2 & \text{çözüm} \\ \hline z & 4 & 1 & 0 & 0 & 0 \\ x_1 & 1 & 0 & \frac{1}{5} & 0 & \frac{3}{5} \\ x_2 & 0 & 1 & -\left(\frac{3}{5}\right) & 0 & \frac{6}{5} \\ s_2 & 0 & 0 & 1 & 1 & 1 \end{array}$$

uyumsuzluğun giderildiği matris

$$\begin{array}{c|cccccc} & x_1 & x_2 & s_1 & s_2 & \text{çözüm} \\ \hline z & 0 & 0 & -\left(\frac{1}{5}\right) & 0 & -\left(\frac{18}{5}\right) \\ x_1 & 1 & 0 & \frac{1}{5} & 0 & \frac{3}{5} \\ x_2 & 0 & 1 & -\left(\frac{3}{5}\right) & 0 & \frac{6}{5} \\ s_2 & 0 & 0 & 1 & 1 & 1 \end{array}$$

Şekil 49. İkinci aşamada başlangıç simpleks tablosu

Şekil 49'daki uyumsuzluğun giderildiği tablonun z satırında negatif eleman bulunduğundan simpleks algoritması uygulanması gerekmektedir. Bu durumda program simpleks algoritmasını çalıştıracığından programa giren ve çıkan değişken değerleri ile birinci aşama sonunda oluşan simpleks tablosu Şekil 50'deki gibi elde edilir.

Giren değişken:

s_1

Çıkan değişken

s_2

[1., aşamanın sonu:]

	x_1	x_2	s_1	s_2	çözüm
z	0	0	0	$\frac{1}{5}$	$-\left(\frac{17}{5}\right)$
x_1	1	0	0	$-\left(\frac{1}{5}\right)$	$\frac{2}{5}$
x_2	0	1	0	$\frac{3}{5}$	$\frac{9}{5}$
s_1	0	0	1	1	1

Şekil 50. İkinci aşama simpleks algoritması sonucunda oluşan tablo

Şekil 50'den de görüleceği üzere ilk simpleks algoritması sonucunda z satırında negatif eleman kalmadığından simpleks algoritması sonlandırılacaktır. Bu durumda temel değişken değerleri ile optimum sonuç Şekil 51'deki gibi elde edilir.

Temel değişkenler:

$(x_1 \ x_2 \ s_1)$

Temel değişkenlerin değerleri:

$\left(\frac{2}{5} \ \frac{9}{5} \ 1\right)$

Temel değişkenlere karşılık gelen çözüm:

$-\left(\frac{17}{5}\right)$

Şekil 51. Örnek 22'nin optimum çözümü

Şekil 51'den dikkat edilirse elde edilen optimum değer Örnek 20'deki sonuç ile uyumludur.

2.5. Dual simpleks yönteminin maxima'da programlanması

Dual simpleks yöntemi kısıtlardan en az biri $\geq$ veya $=$ olduğu ve sağ taraf değerlerinin pozitif olduğu durumlarda kullanılır. Kısıt denklemlerinden $\geq$ şeklinde olanlar $\leq$ kısıtına çevrilir. Böylelikle sağ taraf değerleri negatif olur. $\leq$ kısıtlarının sağ taraf değeri negatif olanı varsa yine dual simpleks yöntemi uygulanır.

Dual Simpleks algoritmasında doğrusal programlamaya optimum fakat uygun olmayan bir çözümle başlanır. Daha sonraki yinelemelerde optimumluğu bozmadan uygun çözüme ulaşılmaya çalışılır. Uygun çözüm elde edildiği yinelemede algoritma sonlandırılır. Dual simpleks yöntemi ve dual simpleks yönteminin alt yapısını oluşturan dualite teorisi giriş bölümünde detaylıca ele alınmıştır. Bu kısımda dual simpleks yönteminin maxima ortamına uygulanışına değinilmektedir.

Dual simpleks algoritmasını simpleks algoritmasında olduğu gibi iki kısma ayırabiliriz. Birinci kısım dual simpleks yönteminin genel algoritmasının oluşturulması, ikinci kısım ise bu algoritma kullanılarak örnek problemlerin çözülmesi şeklindedir. İkinci kısımda örnek problemin katsayıları alınarak dual algoritma uygulanarak genel matris elde edilecektir. Elde edilen bu matris birinci kısımda oluşturulan dual simpleks algoritmasına gönderilerek problemin çözülmesi sağlanacaktır.

Birinci kısımda oluşturulan dual simpleks algoritması aşağıdaki şekildedir.

Algoritma 5. Dual Simpleks Algoritması

Adım 1. Dual simpleks algoritması uygulanacak genel matris alınarak bu matrisin satır ve sütun sayıları belirlenir.

Adım 2. Başlangıçtaki matriste temel değişkenler ve temel değişkenlerin yerleri belirlenerek matriste ilgili yere atanır.

Adım 3. z satırında negatif eleman kalmayıncaya kadar aşağıdaki işlemler yürütülür:

Adım 3.1. Çözüm sütunundaki negatif elemanlara bakılarak *çıkan değişken* ve yeri belirlenir.

Adım 3.2. İlgili oranlara bakılarak pozitif olanlar arasından *giren değişken* ve yeri belirlenir.

Adım 3.3. Pivot elemanın değeri 1 yapılır.

Adım 3.4. Gauss Jordan iterasyon yöntemi uygulanarak pivot elemanın altındaki ve üstündeki elemanlar sıfır (0) yapılır.

Adım 3.5. Elde edilen yeni matris ekranda yazdırılır.

Yukarıda oluşturulan dual simpleks algoritması simpleks yönteminde olduğu gibi program çağrılarak kullanılabilir. Bunun için giriş elemanları genel olarak diğer simpleks algoritmalarında olduğu gibi alınabilir. Fakat burada küçük bir fark oluşmaktadır. Dual simpleks algoritmasının uygulanacağı problemlerde problemin minimizasyon problemi veya maksimizasyon problemi olup olmadığı giriş elemanlarında belirtilmelidir. Bunun nedeni karşılaşılan bazı problemlerde dual simpleks algoritmasının sonunda tekrar bir simpleks algoritması gerekebiliyor olmasıdır. Örneğin, bir maksimizasyon probleminde dual simpleks algoritmasının sonunda z satırında negatif eleman kalabilir. Bu durumda optimum çözüm elde edilmediği anlamına gelmektedir. Bu durumda probleme dual simpleks algoritmasından sonra simpleks algoritması uygulanarak optimum çözümün belirlenmesi gerekebilir. Bundan dolayı probleme diğer algoritmalarından farklı olarak problemin minimizasyon problemi veya maksimizasyon problemi olup olmadığı girilmelidir. Bunu da algoritmada giriş elemanlarına *problem* diye bir değişken girilerek sağlanmaktadır. Eğer problem minimizasyon problemi ise *problem: "min"* eğer problem maksimizasyon problemi ise *problem: "max"* yazılarak giriş elemanlarında belirtilmesi gerekmektedir.

İkinci bir husus yukarıda da belirtildiği üzere dual simpleks yöntemi kısıtlardan en az biri $\geq$ veya $=$ olduğu ve sağ taraf değerlerinin pozitif olduğu durumlarda kullanılmasıdır. Fakat kısıtlamaların hepsi küçük eşit ($\leq$) olduğu durumda dual simpleks algoritması ile problem çözülemeyeceği gibi bir izlenim oluşmaktadır. Bu durumda dual simpleks algoritmasının uygulanması için problem primal problemden dual probleme dönüştürülmektedir. Giriş bölümde anlatıldığı üzere her bir primal problemin Tablo 33'teki gibi bir dual karşılığı mevcuttur. Tüm kısıtlamaları küçük eşittir ($\leq$) olan problem için dual formda tüm kısıtlamalar büyük eşittire ($\geq$) döndürür. Dolayısı ile hazırlanan dual simpleks algoritmasında eğer tüm kısıtlamalar küçük eşittir şeklinde ise hazırlanan algoritma kendi içerisinde bu problemi dual probleme dönüştürdükten sonra ilgili algoritmayı uygulamaktadır.

Şimdi problemlerin dual simpleks algoritması kullanılarak çözdürüldüğü algoritmayı verelim. Burada ana tema dual simpleks algoritmasını uygulayacağımız ana matrisi oluşturmaktır. Dual simpleks algoritması uygulamak için uygun hale getirilen matris ilgili algoritmaya gönderilerek aranılan çözümler elde edilmektedir.

Algoritma 6.: Dual Problemlerin Çözüm algoritması

Adım 1. Sınırlamaların satır ve sütun sayılarına bakılır.

Adım 1.1. Eğer problemin tüm kısıtlamaları küçük eşittir ise problem dual probleme dönüştürülür.

Adım 1.1.1. Dual problemin kısıtlamaların katsayıları ilgili problemin kısıtlamaların transpozu alınarak elde edilir.

Adım 1.1.2. Dual problemin kısıtlamalarının z satırı ilgili problemin kısıtlamaların karşı tarafındaki değerlerden elde edilir.

Adım 1.1.3. Dual problemde kısıtlamaların karşı tarafı ilgili problemin amaç fonksiyonun katsayılarından elde edilir.

Adım 1.2. Eğer en az bir kısıtlama büyük eşittir kısıtlaması içeriyorsa, büyük eşittir kısıtlaması içeren satırlar eksi ile çarpılır.

Adım 2. Dual simpleks algoritması uygulanacak matris elde edilir.

Adım 2.1. İlgili matrisin ilk satırı değişken isimleri ve çözüm'den, ikinci satırı z satırından, diğer satırları sırasıyla eşittir, büyük eşittir, küçük eşittir kısıtlamalarının katsayılarında ve karşı tarafından oluşturulur.

Adım 3. Dual simpleks uygulamaya hazır hale getirilen matris dual simpleks algoritması ile çözdürülür.

Adım 4. Problem eğer maksimizasyon problemi ise ve z satırında negatif eleman mevcut ise standart simpleks algoritması ile problemi çözerek optimum çözümü belirler.

Dual simpleks algoritmasının ve problemlere uygulama algoritmasının maxima'da yazılan kodları Ek 5'te ayrıntılı olarak verilmiştir. Yukarıda dikkat edilirse Adım 4'te ekstra bir standart algoritması gerekebilen durumlar ile karşılaşılabilir. Örneğin, bir maksimizasyon probleminde eğer dual simpleks algoritması sonucunda z satırında negatif eleman yer alıyorsa problemin optimum sonucunun elde edilmediği görülür. Bu durumda optimum çözümü elde etmek için

dual simpleks algoritmasından sonra standart simpleks algoritması uygulanarak aranan çözüm elde edilir. Şimdi dual simpleks algoritmasını bir örnek üzerinden inceleyelim.

Örnek 23. Giriş bölümünde Örnek 12'deki diyetisyen problemini göz önüne alalım.

Örnek 12'de problem

$$\min z = 2x_1 + 3x_2$$

$$3x_1 + 6x_2 \geq 1500$$

$$4x_1 + 2x_2 \geq 800$$

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

şeklinde elde edilmişti. Bu problem için hazırlanan maxima kodunda giriş elemanları

```
P: matrix([-2, -3]);
Abe: matrix([3,6], [4,2]);
Ake: matrix([[]]);
Aeq: matrix([[]]);
bbe: matrix([1500], [800]);
bke: matrix([[]]);
beq: matrix([[]]);
degisken: matrix(["x_1", "x_2"]);
yapay_degisken: matrix(["s_1", "s_2"]);
problem: min;
```

giriş elemanları tanımlanarak

```
dual_simpleks(P, Abe, bbe, degisken, yapay_degisken, problem);
```

şeklinde çalıştırıldığında ilk olarak ekranda dual simpleks algoritmasının uygulanacağı Şekil 52'deki matris yazdırılır.

$$\begin{pmatrix} & x_1 & x_2 & s_1 & s_2 & \text{çözüm} \\ z & -2 & -3 & 0 & 0 & 0 \\ & -3 & -6 & 1 & 0 & -1500 \\ & -4 & -2 & 0 & 1 & -800 \end{pmatrix}$$

Şekil 52. Dual simpleks algoritmasının uygulanacağı matris

Oluşturulan bu matris $dual_simpleks_algoritma(A)$ komutu yardımıyla dual simpleks algoritmasına gönderilerek çözdürülür. Bu durumda ilk olarak temel değişkenler ve değerleri belirlenerek ilgili matriste ilgili yere atanır. Bu durumda Şekil 53'teki matris elde edilir.

Temel değişkenler:

$$(s_1 \ s_2)$$

Temel değişkenlerin değerleri:

$$(-1500 \ -800)$$

$$\begin{pmatrix} & x_1 & x_2 & s_1 & s_2 & \text{çözüm} \\ z & -2 & -3 & 0 & 0 & 0 \\ s_1 & -3 & -6 & 1 & 0 & -1500 \\ s_2 & -4 & -2 & 0 & 1 & -800 \end{pmatrix}$$

Şekil 53. Temel değişkenlerin ile elde edilen başlangıç simpleks matrisi

Şekil 53'te elde edilen matris ile Tablo 35'te verilen dual simpleks başlangıç tablosu birbiri ile uyumludur. Daha sonra algoritma giren çıkan değişkenleri belirleyerek Gauss Jordan eliminasyon yöntemini uygulayarak birinci aşama sonunda Şekil 54'teki simpleks tablosu elde edilir.

Çıkan değişken:

s_1

Giren değişken

x_2

[1.,. aşamanın sonu:]

$$\begin{pmatrix} & x_1 & x_2 & s_1 & s_2 & \text{çözüm} \\ z & -\left(\frac{1}{2}\right) & 0 & -\left(\frac{1}{2}\right) & 0 & 750 \\ x_2 & \frac{1}{2} & 1 & -\left(\frac{1}{6}\right) & 0 & 250 \\ s_2 & -3 & 0 & -\left(\frac{1}{3}\right) & 1 & -300 \end{pmatrix}$$

Şekil 54. Dual simpleks birinci aşama sonunda elde edilen matris

Şekil 54'te algoritmanın birinci aşama sonunda elde ettiği matris ile Tablo 37'deki sonuç matrisi birbiri ile uyumludur. Çözüm sütununda negatif eleman olduğunda algoritma ikinci aşamaya geçecektir. Bu durumda dual simpleks algoritması giren ve çıkan değişkenler ile ikinci aşama sonunda elde edilen simpleks tablosunu Şekil 55'teki gibi ekranda yazdırır.

Çıkan değişken:

s_2

Giren değişken

x_1

[2,. aşamanın sonu:]

$$\begin{array}{c|ccccc} & x_1 & x_2 & s_1 & s_2 & \text{çözüm} \\ \hline z & 0 & 0 & -\left(\frac{4}{9}\right) & -\left(\frac{1}{6}\right) & 800 \\ x_2 & 0 & 1 & -\left(\frac{2}{9}\right) & \frac{1}{6} & 200 \\ x_1 & 1 & 0 & \frac{1}{9} & -\left(\frac{1}{3}\right) & 100 \end{array}$$

Şekil 55. Dual simpleks ikinci aşama sonunda elde edilen matris

Şekil 55'te algoritmanın birinci aşama sonunda elde ettiği matris ile giriş bölümünde Tablo 39'daki sonuç matrisi birbiri ile uyumludur. Çözüm sütununda negatif eleman kalmadığından algoritma sonlandırılacaktır. Bu durumda ekranda optimum değerler ve optimum çözüm Şekil 56'daki gibi yazdırılır.

Temel değişkenler:

$(x_2 \ x_1)$

Temel değişkenlerin değerleri:

$(200 \ 100)$

Temel değişkenlere karşılık gelen çözüm:

800

Şekil 56. Örnek 23 için optimum sonuç

Şekil 56'da elde edilen sonuç ile Örnek 12'deki optimum sonuç birbiri ile uyumlu olduğu görülür.

Örnek 24. (Özturan, 2022)

$$\max z = 2x_1 + x_2$$

$$x_1 - 2x_2 \leq 2$$

$$2x_1 + 3x_2 \geq 6$$

$$3x_1 + 5x_2 \leq 15$$

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

problemını dual simpleks algoritması ile çözelim. Bu durumda giriş elemanları eşitlik kısıtlaması mevcut olmadığı için

```
P: matrix([-2, -1]);
Abe: matrix([2,3]);
Ake: matrix([1, -2], [3,5]);
Aeq: matrix([]);
bbe: matrix([15]);
bke: matrix([2], [15]);
beq: matrix([]);
degisken: matrix(["x_1", "x_2"]);
yapay_degisken: matrix(["s_1", "s_2", "s_3"]);
problem: max;
```

olarak girilerek hemen ardından

```
simpleks_dual(P, Aeq, beq, Ake, bke, Abe, bbq, degisken, problem);
```

yazılarak program çalıştırılır. Bu durumda dual simpleks algoritması için oluşturulan simpleks matrisi Şekil 57'deki gibi ekrana yazdırılır.

$$\begin{pmatrix} & x_1 & x_2 & s_1 & s_2 & s_3 & \text{çözüm} \\ z & -2 & -1 & 0 & 0 & 0 & 0 \\ & -2 & -3 & 1 & 0 & 0 & -6 \\ & 1 & -2 & 0 & 1 & 0 & 2 \\ & 3 & 5 & 0 & 0 & 1 & 15 \end{pmatrix}$$

Şekil 57. Dual simpleks algoritması için oluşturulan simpleks matrisi

Şekil 57’de oluşturulan simpleks matrisi dual simpleks algoritması tarafından çözdürüldüğünde ilk olarak temel değişkenler ilgili matriste yerlerine yazdırılır ve giren/çıkan değişkenler belirlenir. Elde edilen değerler ve değişkenler ekranda yazdırıldıktan sonra ikinci aşama sonunda oluşan simpleks tablosu Şekil 58’deki elde edilir.

$$[1., \text{ aşamanın sonu:}]$$

	x_1	x_2	s_1	s_2	s_3	çözüm
z	$-\left(\frac{4}{3}\right)$	0	$-\left(\frac{1}{3}\right)$	0	0	2
x_2	$\frac{2}{3}$	1	$-\left(\frac{1}{3}\right)$	0	0	2
s_2	$\frac{7}{3}$	0	$-\left(\frac{2}{3}\right)$	1	0	6
s_3	$-\left(\frac{1}{3}\right)$	0	$\frac{5}{3}$	0	1	5

Şekil 58. Örnek 24 için birinci aşama sonucunda oluşan simpleks tablosu

Şekil 58’den görüleceği üzere çözüm sütununda birinci aşama sonunda elde edilen matriste negatif eleman bulunmamaktadır. Bu yüzden dual simpleks algoritması programı sonlandıracaktır. Fakat z satırında negatif eleman bulunduğu ve problem bir maksimizasyon problemi olduğundan optimum çözüm elde edilmediği görülmektedir. Bu gibi durumların yaşanabilmesi durumuna göre yukarıda da belirtildiği üzere problem diye bir değişken tanımlanarak oluşan bu problem çözülmeye çalışılmıştır.

Problem bir maksimizasyon problemi olduğundan ve z satırında negatif eleman bulunduğundan elde edilen son simpleks matrisindeki çözüm optimum çözüm değildir. Dolayısı ile algoritmanın 4. adımında da belirtildiği üzere hazırlanan program probleme standart simpleks algoritması uygulayacaktır. Bu durumda ekranda başlangıçtaki temel değişkenler ve başlangıç simpleks matrisi yazdırılır. Ardından giren ve çıkan değişkenler belirlenerek standart simpleks algoritması sonucunda oluşna birinci simpleks tablosu ekranda Şekil 59’daki gibi yazdırılır.

Giren değişken:

x_1

Çıkan değişken

s_2

[1,. aşamanın sonucunda elde edilen simpleks tablosu:]

$$\begin{array}{c|cccccc} & x_1 & x_2 & s_1 & s_2 & s_3 & \text{çözüm} \\ \hline z & 0 & 0 & -\left(\frac{5}{7}\right) & \frac{4}{7} & 0 & \frac{38}{7} \\ x_2 & 0 & 1 & -\left(\frac{1}{7}\right) & -\left(\frac{2}{7}\right) & 0 & \frac{2}{7} \\ x_1 & 1 & 0 & -\left(\frac{2}{7}\right) & \frac{3}{7} & 0 & \frac{18}{7} \\ s_3 & 0 & 0 & \frac{11}{7} & \frac{1}{7} & 1 & \frac{41}{7} \end{array}$$

Şekil 59. Standart simpleks algoritmasının 1aşaması sonucunda oluşan tablo

Şekil 59'dan dikkat edilirse z satırında negatif eleman mevcuttur. Dolayısı ile optimum değer için ikinci bir simpleks algoritmasına gerek vardır. Bu durumda algoritma problemi çözmeye devam edecek olup ikinci aşama sonunda ekranda Şekil 60'taki matris yazdırılacaktır.

[2,. aşamanın sonucunda elde edilen simpleks tablosu:]

$$\begin{array}{c|cccccc} & x_1 & x_2 & s_1 & s_2 & s_3 & \text{çözüm} \\ \hline z & 0 & 0 & 0 & \frac{7}{11} & \frac{5}{11} & \frac{89}{11} \\ x_2 & 0 & 1 & 0 & -\left(\frac{3}{11}\right) & \frac{1}{11} & \frac{9}{11} \\ x_1 & 1 & 0 & 0 & \frac{5}{11} & \frac{2}{11} & \frac{40}{11} \\ s_1 & 0 & 0 & 1 & \frac{1}{11} & \frac{7}{11} & \frac{41}{11} \end{array}$$

Şekil 60. Simpleks algoritmasının ikinci aşaması sonucunda oluşan matris

Şekil 60'tan z satırında negatif eleman mevcut olmadığına problemnin optimum değeri elde ettiği anlaşılır. Dolayısı ile program algoritmayı sonlandırarak ekranda temel değişkenlerin değerleri ve optimum çözüm Şekil 61'deki gibi yazdırılır.

Temel deęişkenler:

$$(x_2 \ x_1 \ s_1)$$

Temel deęişkenlerin deęerleri:

$$\begin{pmatrix} 9 & 40 & 41 \\ 11 & 11 & 11 \end{pmatrix}$$

Temel deęişkenlere karřılık gelen çözüm:

$$\frac{89}{11}$$

Şekil 61. Örnek 24 için optimum çözüm

2.6 Matrisel Formun Maxima Ortamında Kodlanması

Giriş bölümünde primal problemler için simpleks yöntemi matrisel formda ifade edilmişti. Bu bölümde ise matrisel form için maxima kodu oluşturulmaktadır. Bu kod oluşturulurken ana tema giriş bölümü Tablo 40'ta verilen şema referans alınmıştır. Verilen giriş elemanları yardımıyla matrisel formda deęinilen B ve C_B matrisleri ile Tablo 40'taki matris oluşturulmaktadır. Sonrasında ilgili bölümde belirtildięi şekilde giren ve çıkan deęişkenler belirlenerek dięer aşamalarda oluşan simpleks tabloları yine matrisel form yöntemiyle oluşturularak optimizasyon problemlerinin çözülmesi sağlanmaktadır. Öte yandan matrisel form oluşturulurken giriş elemanları olarak giriş bölümünde verilen C_I, C_{II}, b, P matrisleri probleme girilmelidir. Burada P matrisi $[A \ I]$ matrisi olarak girilmelidir. Matrisel form için oluşturulan maxima algoritması ařaęıdaki şekildedir.

Algoritma 7. Matrisel form için maxima algoritması

Adım 1. Verilen giriş elemanları ile P matrisindeki birim vektörler yardımıyla temel deęişkenler ve yerleri belirlenir.

Adım 2. Temel deęişkenlerin bulunduęu sütunlar yardımıyla B matrisi ve C_B matrisi belirlenir.

Adım 3. Tablo 40 ile başlangıç simpleks tablosu belirlenir.

Adım 4. Optimumluk saęlanana kadar ařaęıdaki işlemleri tekrarlanır:

Adım 4.1. (9)'daki optimumluk koşulu ile giren deęişkeni belirlenir.

Adım 4.2. (10)'daki simpleks uygunluk koşulu ile çıkan deęişkeni belirlenir.

Adım 4.3. Tablo 40 ile matrissel gösterim ile diğer simpleks tabloları elde edilir.

Yukarıda verilen algoritma için hazırlanan maxima kodu Ek 7’de verilmiştir. Şimdi hazırlanan algoritmayı bir örnek üzerinden test edelim.

Örnek 25. Giriş bölümünde ele alınan Örnek 14 için maxima ortamında hazırlanan kodun giriş elemanları

$X: \text{matrix}(["x_1", "x_2", "x_3", "s_1", "s_2", "s_3"]);$

$CI: \text{matrix}([20,16,15]);$

$CII: \text{matrix}([0,0,0]);$

$C: \text{addcol}(CI, CII);$

$b: \text{matrix}([90], [70], [80]);$

$P: \text{matrix}([1.5,1,2,1,0,0], [1,2,2,0,1,0], [2,1,1,0,0,1]);$

şeklinde girilmelidir. Sonrasında hazırlanan kod

$\text{simpleks_matrissel}(X, CI, CII, b, P);$

yazılarak çalıştırılırsa ilk olarak ekranda başlangıç simpleks tablosu Şekil 62’deki gibi yazdırılır.

başlangıç matrisi:

	x_1	x_2	x_3	s_1	s_2	s_3	çözüm
z	-20.0	-16	-15	0	0	0	0
s_1	1.5	1	2	1	0	0	90
s_2	1.0	2	2	0	1	0	70
s_3	2.0	1	1	0	0	1	80

Şekil 62. Örnek 25 için başlangıç simpleks tablosu

Sonrasında program optimumluk koşulu ve uygunluk koşulları yardımıyla giren ve çıkan elemanları belirleyerek birinci aşama sonunda oluşan simpleks tablosunu Şekil 63’teki gibi ekranda yazdırır.

Giren değişken:

x_1

Çıkan değişken

s_3

[1,.aşamada oluşan Simpleks matrisi:]

	x_1	x_2	x_3	s_1	s_2	s_3	çözüm
z	0.0	-6.0	-5.0	0.0	0.0	10.0	800.0
s_1	0.0	0.25	1.25	1.0	0.0	-0.75	30.0
s_2	0.0	1.5	1.5	0.0	1.0	-0.5	30.0
x_1	1.0	0.5	0.5	0.0	0.0	0.5	40.0

Şekil 63. Birinci aşama sonucunda oluşan simpleks tablosu

Şekil 63'te elde edilen tablo ile Tablo 42'de elde edilen tablo birbiri ile uyumludur. z satırında negatif eleman bulunduğundan algoritma ikinci aşamaya geçecektir. Bu durumda giren ve çıkan değişkenler ve ikinci aşama sonunda elde edilen simpleks tablosu Şekil 64'teki gibidir.

Giren değişken:

x_2

Çıkan değişken

s_2

[2,.aşamada oluşan Simpleks matrisi:]

	x_1	x_2	x_3	s_1	s_2	s_3	çözüm
z	0.0	0.0	1.0	0.0	4.0	7.999999999999999	919.9999999999999
s_1	0.0	1.1102230246251565 10^{-16}	1.0	1.0	-0.16666666666666666	-0.6666666666666666	25.0
x_2	0.0	1.0	1.0	0.0	0.6666666666666666	-0.3333333333333333	20.0
x_1	1.0	0.0	0.0	0.0	-0.3333333333333333	0.6666666666666666	29.999999999999996

Şekil 64. İkinci aşama sonucunda oluşan simpleks tablosu

Şekil 64'teki elde edilen tabloda rasyonel sayılar yerine ondalıklı sayılar kullanıldığı için giriş bölümündeki Tablo 43'ten küçük bir farklılık görülmektedir. Fakat ilgili sayıları yuvarladığımızda aynı sonuçlar ortaya çıktığı açıktır.

2.7. Maxima’da Duyarlılık Analizi

Giriş bölümünde yapılan duyarlılık analizini Maxima programına uyarlayalım. Bu bölümde duyarlılık analizi yapılırken iki durum üzerinde durulacaktır:

1. Kısıtlamaların sağ taraflarındaki değişimler için bir aralık belirleme
2. Amaç fonksiyonu katsayılarındaki değişimler için bir aralık belirleme.

Giriş bölümü b’nin elemanlarının uygunluk aralığı kısmında yapıldığı gibi her bir kısıt değişkeninin ayrı ayrı uygunluk bölgesi bulunacak şekilde matrissel form için hazırlanan Maxima kodu geliştirilecektir. Öte yandan bütün kısıtlamaların sağ tarafı aynı anda değiştiği durum için uygunluk bölgesinin belirlenmesinin zor olduğunu giriş bölümünden biliyoruz. Bütün kısıtlamaların sağ tarafındaki değişimler için hazırlanacak programı yazmak mümkün iken bu şekilde hazırlanan programda eşitsizliklerin çözümleri hem uzun süreceğinden hem de karmaşık sonuçlar ekranda yazdıracağından bu kısım ile bu çalışmada ilgilenilmeyecektir. Benzer durum amaç fonksiyonundaki değişimler için de söz konusudur. Bu çalışmada her bir amaç katsayısındaki değişim için ayrı ayrı uygunluk bölgesi belirlenmektedir.

Maxima’da eşitsizlik sistemi çözdürülürken kullanılabilecek komutlardan biri *fourier_elim*’dir. Bu komutun çalıştırılması için ilk önce bu komutun Maxima ortamında yüklenerek kullanıma hazır hale getirilmelidir. Bunun için bu programı kullanmadan önce *load('fourier_elim)* ile ilgili komut Maxima ortamına yüklenmelidir. Daha sonra bu komut ile elde edilen eşitsizlik sistemi çözdürülerek problemin uygunluk aralığı belirlenebilir.

Kısıtlamalardaki değişim ve amaç katsayılarındaki değişim için matrissel forma eklenmesi gereken kodlar Ek 8’de verilmiştir. Şimdi bu programı giriş bölümünde verilen örnek için test edelim.

Örnek 26. Giriş bölümünde Örnek 14.2’da kısıtlamaların sağ tarafındaki değişimler için uygunluk bölgeleri sırasıyla $d_1 \geq -25$, $-30 \leq d_2 \leq 90$, $-45 \leq d_3 \leq 75/2$ olarak belirlenmişti. Aynı problem Maxima ortamında yazılan kod ile hesaplanırsa şekil 65’teki sonuç elde edilir.

		[3,.değişken için]
	[2,.değişken için]	$-45 < d$
[1,.değişken için]	$-30 < d$	$d < \frac{75}{2}$
$-25 < d$	$d < 90$	

Şekil 65. Kısıtlamaların sağ tarafındaki değişimler için uygunluk aralığı

Şekil 65'te elde edilen sonuçlar ile Örnek 14.2'deki sonuçlar birbiri ile uyumludur. Programda her bir değişkenin ayrı ayrı olmak üzere d kadar arttırıldığını not edelim.

Öte yandan duyarlılık analizi ikinci kısımda optimumluğu etkileyen değişiklikler incelenmişti. İlgili kısımda (iii)'de optimumluğun korunduğu değişim aralığının nasıl belirlendiğine değinilmişti. Örnek 15.3'te ise lastik üretim yapan firma modeli için amaç fonksiyonunun katsayılarındaki değişime göre bir uygunluk aralığı belirlenmişti. Şimdi oluşturduğumuz matrissel form algoritmasını amaç fonksiyonunun katsayılarındaki değişime göre bir uygunluk aralığı bulacak şekilde geliştirelim. Matrissel formda hazırlanan koda eklenmesi gereken kısım Ek 8 (ii)'de verilmiştir. Bu kod çalıştırılarak ele alınan örneklerde optimumluk için bir uygunluk aralığı belirlenebilir.

Örnek 15.3 için hazırlanan kod çalıştırılırsa ekranda aşağıdaki sonuçlar yazdırılır.

[1,.değişken için]	[2,.değişken için]	[3,.değişken için]
$-12 < d$	$-1 < d$	
$d < 12$	$d < 24$	$d < 1$

Şekil 66. Amaç fonksiyonunun katsayılarındaki değişimler için uygunluk aralığı

Şekil 65'ten görüleceği üzere ilgili program x_1 'in $(-12,12)$ aralığında değiştiğinde, x_2 'inin $(-1,24)$ aralığında değiştiğinde ve x_3 'ün 1'den küçük değerler aldığı anda optimumluğun bozulmayacağını göstermektedir. Bu sonuçlar Örnek 15.3'teki sonuçlar ile uyumludur.

3. SONUÇLAR

Bu çalışmanın ilk bölümünde doğrusal programlama problemlerinin çözümünde kullanılan grafik yöntemi, simpleks yöntemi, M yöntemi, iki aşamalı simpleks yöntemi, dual simpleks yöntemi, duyarlılık analizi gibi yöntemler örneklerle açıklanmıştır. Bu kısımda seçilen örnekler lisans öğrencilerinin anlayabileceği düzeyde seçilmeye çalışılmıştır. Seçilen örnekler günlük hayatta karşılaşılabilecek endüstriyel problemlerden seçilmiştir. Böylece lisans öğrencileri için de güzel bir kaynak oluşturulmaya çalışılmıştır.

Bu çalışmanın ikinci kısmında ise ilk bölümde ele alınan yöntemler için geliştirilmiş olan maxima algoritmaları verilmiştir. Bu programın seçilmesindeki ana etken ücretsiz olması ve programlamaya elverişli olmasıdır. Böylece herkes tarafından ulaşılabilir olup kullanıcıların kendi programlarını sisteme entegre edebilmesine olanak sağlamaktadır. Hazırlanan programlar birinci bölümde ele alınan örnekler için test edilmiştir. İlgili programlar her bir aşamada elde edilen çözümleri ekranda yazdıracak şekilde hazırlanmıştır. Elde edilen sonuçlar şekil olarak teze eklenmiştir. Maxima programının elde ettiği sonuçlar ile birinci bölümde elde edilen sonuçların uyumlu olduğu gözlemlenmiştir. Böylece ilgili programların güvenilirliği gösterilmeye çalışılmıştır. İlgili programlar ekte açık erişimli kod olarak verilmiş olup herkes tarafından kullanılabilir bir formatta teze eklenmiştir. Dolayısı ile lineer programlama çalışan öğrencilerin kendi problemlerini hazırlanan program üzerinden test edebilmelerine imkân sağlanmıştır. Böylece kullanıcılar ele aldıkları problemde elde ettikleri sonuçlar ile programın elde ettiği sonuçları karşılaştırabileceklerdir. Sonuçların uyumlu olmaması durumunda hangi adımda hata yaptıklarını kolaylıkla görüp o aşamadan sonraki çözümlere yoğunlaşabileceklerdir. Bu sayede problemi tekrar çözmek yerine sadece hata olan kısımdan sonrasını çözmeleri yeterli olacaktır. Bu ise önemli bir işlem yükünden kullanıcıyı kurtarmasına neden olacağı düşünülmektedir.

4. ÖNERİLER

Bu çalışmada lineer programlama yöntemlerinden grafik yöntemi, simpleks yöntemi, M yöntemi, iki aşamalı simpleks yöntemi, dual simpleks yöntemi gibi yöntemler incelenerek programlanmıştır. Bu yöntemlerin haricinde tam sayılı programlama, iç bölge yöntemleri gibi yöntemlerde mevcuttur. Bu yöntemler de maxima programına entegre edilerek açık erişimli kodlar hazırlanabilir. Ayrıca yeni yöntemlerin eklendiği maxima programları günlük hayattaki endüstriyel problemler üzerinden test edilebilir. Literatürdeki daha karmaşık problemlerde hazırlanan maxima kodları ile çözdürülebilir.

Öte yandan bu çalışmadaki problemlerin hepsinde karar değişkenleri üzerinde $x_i \geq 0$ kısıtlamasının mevcut olduğu problemler ele alınarak algoritmalar buna göre oluşturulmuştur. Bazı durumlarda bu değişken değerleri sınırlı olmadığı problemlerde söz konusu olabilir. Bu durumlar için ekte verilen algoritmalar güncellenebilir.

Bu çalışmada ele alınan yöntemlerden grafik yöntemi, ortaöğretim düzeyindeki öğrencilerin anlayabileceği seviyede bir yöntem olarak göze çarpmaktadır. Bu yöntemin kısa bir özeti dersin öğretmeni tarafından öğrencilere anlatılarak örnekler üzerinden problemlerin çözülmesi sağlanabilir. Bu sayede öğrencilerin matematiğin günlük hayattaki karşılığına dair bir bilgi sahibi olabilmeleri sağlanabilir. Böylece öğrencilerin matematiğe olan ilgilerinin artacağı düşünülmektedir. Bu düşüncenin doğruluğu eğitim fakültesindeki akademisyenler ile koordineli olarak okullarda test edilebilir.

Hazırlanan maxima programları ekte verilmiştir. Diğer programlama dillerinde olduğu gibi giriş elemanlarını alarak yöntemi programın kendisinin belirlediği bir program yazılabilir. Bunun için problemin hangi yöntem ile çözülebileceğini belirleyen küçük bir maxima kodu yazılabilir. Daha sonra bu çalışmada ekte verilen kodlara giriş elemanları yönlendirilerek ilgili problem rahatlıkla çözdürülebilir.

KAYNAKÇA

- Alan, M. A. ve Yeşilyurt C. (2004). Doğrusal Programlama Problemlerinin Excel ile Çözümü. *C.Ü İktisadi ve İdari Bilimler Dergisi* 5(1). 151-162
- Arık, M. C., (2020). *Doğrusal Programlama Çözücülerinin Performans Analizleri* [Yüksek Lisans Tezi]. Bursa Uludağ Üniversitesi.
- Coşkun, A, (2018). *Maxima ile Sembolik Hesaplama ve Kodlama*
- Coşkun, A, (2022). *Endüstriyel ve Uygulamalı Matematiğe Giriş: Üretim Organizasyonu: Optimizasyon Modelleri, Bölüm3.*
- Hiller, F. S. ve Lieberman, G.J. (2019) *Yöneylem Araştırmasına Giriş*. (Çeviri Editörü F. Çebi, İ. D. Kocabaş, Nobel Akademi Yayıncılık. (1992).
- Özturan, A, (2022). *Optimizasyon ve Matlab Uygulamaları*.
- Taha, H. A, (2010). *Yöneylem Araştırması*. (Çeviri Editörü Ş.A. Baray, Ş. Esnaf), Literatür Yayıncılık. (2007).
- URL-1, <https://www.mathworks.com/products/matlab/student.html>, linprog komutu, 22 Aralık 2023.
- URL-2, <https://www.wolfram.com/mathematica>, LinearOptimization komutu, 22 Aralık 2023.
- URL-3, <https://www.maplesoft.com/products/Maple>, LPSolve Komutu, 22 Aralık 2023.
- URL-4, <https://octave.org>, Lingprog Komutu, 22 Aralık 2023.
- URL-5, <https://maxima.sourceforge.io>, minimize_lp komutu, 22 Aralık 2023.

EKLER

Ek-1. Grafik yöntemi için maxima kodları

```
grafik_yontemi(P,Ake,bke,Abe,bbe,Aeq,beq):=block(  
  /*Sınırlamaların sayısı tüm sınırlama matrislerinin satır sayılarına bakılarak  
kontrol ediliyor. Ayrıca boş küme olan elemanları sistemden çıkarmak için sütun  
sayılarına da bakılıyor*/  
  Aeq_sutun_say:length(transpose(Aeq)),  
  Aeq_satir_say:length(Aeq),  
  Ake_sutun_say:length(transpose(Ake)),  
  Ake_satir_say:length(Ake),  
  Abe_sutun_say:length(transpose(Abe)),  
  Abe_satir_say:length(Abe),  
  /*-----*/  
  /*Kesişim noktaları belirlenecek lineer denklemler elde ediliyor.*/  
  say:0,/*Çözülmesi gereken denklem sayısını tutan değişken*/  
  denklemler:makelist(concat(x,i),i,1,1),  
  if Aeq_sutun_say#0 then (for i:1 thru Aeq_satir_say do  
(say:say+1,denk[say]:Aeq[i].matrix([x,y])=beq[i][1],graf:solve(denk[say],y),  
denklemler:cons(rhs(graf[1]),denklemler))),  
  if Ake_sutun_say#0 then (for i:1 thru Ake_satir_say do  
(say:say+1,denk[say]:Ake[i].  
matrix([x,y])=bke[i][1],graf:solve(denk[say],y),denklemler:cons(rhs(graf[1]  
,denklemler))),  
  if Abe_sutun_say#0 then (for i:1 thru Abe_satir_say do  
(say:say+1,denk[say]:Abe[i].  
matrix([x,y])=bbe[i][1],graf:solve(denk[say],y),denklemler:cons(rhs(graf[1]  
,denklemler))),  
  denklemler:delete(x1,denklemler),  
  /*-----*/  
  /*Değişkenlerin sıfırdan büyük eşit olması kısıtlamaları ekleniyor*/  
  say:say+1,
```

Ek-1 (Devam). Grafik yöntemi için maxima kodları

```
denk[say]:x=0,
say:say+1,
denk[say]:y=0,
/*-----*/
/*Doğruların kesişim noktaları belirleniyor*/
say2:0/*Doğruların kesişim noktası sayısını tutan değişken*/
for i:1 thru say-1 do (for j:i+1 thru say do
(bb:linsolve([denk[i],denk[j]], [x,y]),
(if bb#[[] then (say2:say2+1,b[say2]:bb))))),
/*-----*/
/*Kesişim noktalarından hangileri başlangıçtaki eşitlitsizlik sistemini
sağladığını belirlemek için orjinal denklem sistemleri oluşturuluyor*/
say3:0,
if Aeq_sutun_say#0 then (for i:1 thru Aeq_satir_say do
(say3:say3+1,indenk[say3]:Aeq[i].
matrix([x,y])=beq[i][1])),
if Ake_sutun_say#0 then (for i:1 thru Ake_satir_say do
(say3:say3+1,indenk[say3]:Ake[i].
matrix([x,y])<=bke[i][1])),
if Abe_sutun_say#0 then (for i:1 thru Abe_satir_say do
(say3:say3+1,indenk[say3]:Abe[i].
matrix([x,y])>=bbe[i][1])),
say3:say3+1,
indenk[say3]:x>=0,
say3:say3+1,
indenk[say3]:y>=0,
/*-----*/
/*Elde edilen kesişim noktalarından hangilerinin orjinal eşitsizlik sistemini
sağladığı kontrol ediliyor*/
esasm:matrix([0,0]),
for i:1 thru say2 do (sayac:0,(for j:1 thru say3 do
```

Ek-1 (Devam). Grafik yöntemi için maxima kodları

```
(if subst([x=rhs(b[i][1]),y=rhs(b[i][2])],indenk[j]) then (sayac:sayac+1)),
(if
                                sayac=say3                                then
(esasm:addrow(esasm,matrix([rhs(b[i][1]),rhs(b[i][2])])))),
/*-----*/
/*Elde edilen matriste birkaç nokta birden fazla yazıldığı görülüyor. Bu fazla
noktalar sistemden çıkarılıyor*/
esasm:submatrix(1,esasm),
satir_say_esasm:length(esasm),
son_matris:matrix([esasm[1][1],esasm[1][2]]),
for i:2 thru satir_say_esasm do (sayac:0,(for j:1 thru i-1 do (if
esasm[i]#esasm[j]      then      sayac:sayac+1)),(if      sayac=i-1      then
son_matris:addrow(son_matris,esasm[i])),
satir_say_sonm:length(son_matris),
disp("Köşe Noktaları"),
disp(son_matris),
/*-----*/
/*Köşe noktaları arasından maksimum olanı belirlenerek maksimum değer ve
maksimum yapan noktalar ekrana yazdırılıyor*/
sonuc:son_matris.P,
mak:-10^10,
for i:1 thru satir_say_sonm do (if sonuc[i][1]>mak then
(mak:sonuc[i][1],yer:i)),
mak_deger:son_matris[yer],
mak_sonuc:sonuc[yer][1],
disp("Optimum noktalar:"),
disp(mak_deger),
disp("Optimum değer"),
/*-----*/
plot2d(denklemler,[x,0,mak_deger[1]+5]),
return(mak_sonuc));
```

Ek-2. Sadece küçük eşittir kısıtlamaları içeren problem için maxima programı
*/*Bu program her biri küçük eşit sınırlamalarından oluşan lineer
programlama problemleri için Simpleks algoritmasını oluşturur.*

Input: amaç fonksiyonunun katsayıları:P

küçük eşittir sınırlamalarının katsayıları: Ake

küçük eşittir sınırlamalarının karşı tarafları:bke

değişken isimleri: degisken

yapay değişken isimler:yapay_degisken

Output:

optimum değer matrisi:A

şeklindedir./*

simpleks_standart(P,Ake,bke,degisken,yapay_degisken):=block(

*/*Ake nin satır sayısı kaç yapay değişken eklenmesi gerektiğini verdiğinden
hesaplanır.*/ A_satir_sayisi:length(Ake),*

*/*1. satırdaki değişkenler satırı oluşturulurken problem değişkenleri ile
yapay değişkenler yan yan ilk satırda yazılır.*/*

degiskenler:addcol(degisken,yapay_degisken),

*birim_matris:ident(A_satir_sayisi),/*Yapay değişkenler ile eklenecek olan
birim matris*/*

*/*Ake matrisinin yanına birim matris eklenerek sınırlamaların orta kısmı elde
edilir.*/*

*A_alt:addcol(Ake,birim_matris),/*Ake nin yanına birim matris ekleniyor*/*

*/*Ana matrisin 2. satırında amaç fonksiyonunda yapay değişkenler
olmadığından bunların katsayıları olarak sıfırlar alınıyor*/*

*sifirlar:zeromatrix(1,A_satir_sayisi),/*Yapay değişkenlerin amaç fonksiyonda
olmamasından dolayı katsayıları sıfırlar alınıyor*/*

*P_y:addcol(P,sifirlar),/*Amaç fonksiyonuna yapay değişkenlerin katsayıları
ekleniyor*/*

*/*Ana matrisin ilk sütunu temel değişkenler ve z yi içermekte olup başlangıçta
temel değişkenler bilinmediğinden bunlar yerine boşluk yazılıyor.*/*

*A_sol:zeromatrix(A_satir_sayisi+2,1),/*Ana matrisin ilk sütunu sıfırlanıyor*/*

Ek-2 (Devam). Sadece küçük eşittir kısıtlamaları içeren problem için maxima programı

```
for i:1 thru A_satir_sayisi+2 do (if i#2 then A_sol[i][1]:" " else A_sol[i][1]:z),/*Ana matrisin ilk sütunu dolduruluyor*/
```

```
/*Ana matrisin son sütunu [çözüm,0,bke] içerdiğinden bunlar birbirine ekleniyor.*/
```

```
A_son_sutun:addrow(matrix(["çözüm"],[0]),bke),
```

```
/*Ana matrisin ust orta kısmı ilk satırı değişkenleri ikinci satırı ile P_y yi içerdiğinden bunlar birbirine ekleniyor.*/
```

```
A_ust:addrow(degiskenler,P_y),
```

```
/*Ana matrisin orta kısmı değişkenler, amaç fonksiyonu ve A_alt ı içerdiğinden bunlar birbirine ekleniyor.*
```

```
/ A_orta:addrow(A_ust,A_alt),
```

```
/*Ana matrisin sol kısmı A_sol ile A_orta nın birleşimi olduğundan bunlar birbirine ekleniyor.*/
```

```
A_sol_taraf:addcol(A_sol,A_orta),
```

```
/*Ana matrisin sol tarafına ana matrisin son sütunu eklenmesiyle ana matris oluşturulmuş olur.*/
```

```
A:addcol(A_sol_taraf,A_son_sutun),
```

```
/*Simpleks algoritması çağrılarak sonuçlar hesaplanıyor.*/
```

```
load("simpleks_algoritma.mac"),
```

```
simpleks_algoritma(A );
```

Ek-3. M yöntemi için maxima kodları

*/*Bu program kısıtlamaları küçük eşittir, büyük eşittir ve eşittir'den en az iki tanesinden*

oluşan lineer programlama problemleri için

M Yöntemi algoritmasını oluşturur.

Input:

amaç fonksiyonunun katsayıları:P

küçük eşittir sınırlamalarının katsayıları: Ake

küçük eşittir sınırlamalarının karşı tarafları:bke

büyük eşittir sınırlamalarının katsayıları: Abe

büyük eşittir sınırlamalarının karşı tarafları:bbe

eşittir sınırlamalarının katsayıları: Aeq

eşittir sınırlamalarının karşı tarafları:beq

değişken isimleri: degisken

yapay değişken isimler:yapay_degisken

Output:

optimum değer matrisi:A

şeklindedir./*

simpleks_M_yontemi(P,Aeq,beq,Ake,bke,Abe,bbq,degisken,yapay_degisken):
=block(

*/*Problemdeki sınırlamaların satır ve sütun sayılarına bakılarak probleme kaç tane*

dolgu değişken, kaç tane artık değişken ve kaç tane M değişkeni katsayısı ekleneceğine karar veriliyor./*

Aeq_sutun_say:length(transpose(Aeq)),

if Aeq_sutun_say#0 then Aeq_satir_say:length(Aeq) else Aeq_satir_say:0,

Ake_sutun_say:length(transpose(Ake)),

if Ake_sutun_say#0 then Ake_satir_say:length(Ake) else Ake_satir_say:0,

Abe_sutun_say:length(transpose(Abe)),

if Abe_sutun_say#0 then Abe_satir_say:length(Abe) else Abe_satir_say:0,

R_sayisi:Aeq_satir_say+Abe_satir_say,

dolgu_deg_say:Ake_satir_say,

Ek-3 (Devam). M yöntemi için maxima kodları

```
if Abe_sutun_say#0 then artik_deg_say:Abe_satir_say else artik_deg_say:0,
toplaml_satir_say:R_sayisi+dolgu_deg_say,
if Ake_sutun_say#0 then
toplaml_sutun_say:Ake_sutun_say+dolgu_deg_say+artik_deg_say+R_sayisi
else
(if Abe_sutun_say#0 then
toplaml_sutun_say:Abe_sutun_say+dolgu_deg_say+artik_deg_say+R_sayisi
else
toplaml_sutun_say:Aeq_sutun_say+dolgu_deg_say+artik_deg_say+R_sayisi),
/*Problemdeki 1. ve son sütun hariç sütun sayısı*/
yapay_kisim_sutun_say:dolgu_deg_say+artik_deg_say+R_sayisi,
/*-----
-----*/
/*Problemin başlangıç simpleks tablosu oluşturuluyor.*/
M:10^5,/*M değeri büyük bir değer olarak seçiliyor*/
sifirlar_mat: zeromatrix(toplaml_satir_say,yapay_kisim_sutun_say),
/*Yapay değişkenlerin bölgeleri sıfırlanıyor*/
/*Yardımcı problemdeki sınırlamalar için eşitlikler, büyük eşittirler, küçük
eşittirler sıralamasındadır.
Sınırlamaların katsayıları sıfırlanan bölgeye ekleniyor.*/
if (Abe_sutun_say=0) then yordimci_problem1:addrow(Aeq,Ake)
elseif (Aeq_sutun_say=0) then yordimci_problem1:addrow(Abe,Ake)
elseif (Ake_sutun_say=0) then yordimci_problem1:addrow(Aeq,Abe)
else yordimci_problem1:addrow(Aeq,Abe,Ake),/*Sınırlamaların katsayıları*/
/*Problemin son sütunu 2. satır çözüm satırı olacağından ilk değeri 0'dır.
Diğer değerleri ise sınırlamaların karşı tarafına karşılık gelen değerlerdir.
Bunlar aynı sınırlamalar sırası ile girilmelidir.*/
if (Abe_sutun_say=0) then b:addrow(matrix([0]),beq,bke)
elseif (Aeq_sutun_say=0) then b:addrow(matrix([0]),bbe,bke)
elseif (Ake_sutun_say=0) then b:addrow(matrix([0]),beq,bbe)
```

Ek-3 (Devam). M yöntemi için maxima kodları

```
else b:addrow(matrix([0]),beq,bbe,bke),/*Sınırlamaların karşı tarafının katsayıları*/  
  
/*Problemin son sütunu, değişken satırı hariç.*/  
  
/*Simpleks tablosunun ilk 2 satırı ve 1. ve son sütunları hariç sınırlamaları içeren kısmı  
problemin sınırlamaları ile yapay değişkenlerin sıfırlarından oluşan kısımdır.*/  
  
yardimci_problem:addcol(yardimci_problem1,sifirlar_mat),/*Sınırlamalar katsayılar matrisi*/  
  
/*Yardımcı problemin 2. satırında sadece yapay R değişkenlerine içereceğinden  
ilk olarak bunların hepsi sıfır alınıyor. Bunların eleman sayısı 1. ve son sütun hariç  
toplam değişken sayısı kadardır. Toplam değişken sayısı toplam_sutun_say ile başlangıçtaki değişkenlerin toplamı kadardır.*/  
  
yardimci_problem_ikinci_satirr:zeromatrix(1,yapay_kısım_sutun_say),  
/*Problemdeki ikinci satır sıfırlanıyor*/  
  
yardimci_problem_ikinci_satir:addcol(P,yardimci_problem_ikinci_satirr),  
/*Problemdeki ikinci satır sıfırlanıyor*/  
  
/*Yardımcı problemdeki sıfırlanan yerler dolduruluyor. Bunun için ilk artık değişkenlerin katsayıları -1 yapılıyor. Artık değişkenler eşitlik içeren satırlardan sonra eklenmelidir.*/  
  
for i:1 thru artik_deg_say do  
if Ake_sutun_say#0 then  
yardimci_problem[Aeq_satir_say+i][Ake_sutun_say+i]:-1  
else yardimci_problem[Aeq_satir_say+i][Ake_sutun_say+i]:-1,  
/*Artık değişkenlerin katsayıları -1 yapılıyor*/  
/*Yardımcı problemdeki dolgu değişkenlerin katsayıları +1 olmalıdır.  
Problemdeki dolgu değişkenleri eşitlik ve küçük eşitliklerden sonra verilmiş idi.*/  
  
for i:1 thru dolgu_deg_say do
```

Ek-3 (Devam). M yöntemi için maxima kodları

```
yardimci_problem[Aeq_satir_say+Abe_satir_say+i][Ake_sutun_say+artik_deg_say+i]:1,  
/*Dolgu değişkenlerinin katsayıları +1 yapılıyor*/  
/*Yardımcı problemdeki R'lerin katsayıları +1 olmalıdır. Yardımcı problemin  
ilk satırındaki R lerin katsayıları ise 1 olarak girilmelidir*/  
for i:1 thru R_sayisi do  
if Ake_sutun_say#0 then  
(yardimci_problem[i][Ake_sutun_say+artik_deg_say+dolgu_deg_say+i]:1,  
yardimci_problem_ikinci_satir[1][Ake_sutun_say+artik_deg_say+dolgu_deg_say+i]:M)  
else  
(yardimci_problem[i][Aeq_sutun_say+artik_deg_say+dolgu_deg_say+i]:1,  
yardimci_problem_ikinci_satir[1][Aeq_sutun_say+artik_deg_say+dolgu_deg_say+i]:M),  
/*R lerin katsayıları giriliyor*/  
/*Yardımcı problemin ikinci satırı ile alt satırları birleştiriliyor. Bunlar b  
sütununu ve temel değişkenler sütununu içermemektedir.*/  
yardimci_prob_son1:addrow(yardimci_problem_ikinci_satir,yardimci_problem),  
/*Yardımcı problemin sınırlamalar matrisi*/  
/*Yardımcı problem 1. satır hariç son sütunu çözümü ve sınırlamaların karşı  
tarafını içerecek şekilde oluşturuluyor.*/  
yardimci_prob_son:addcol(yardimci_prob_son1,b),/*Yardımcı problemin  
son sütunu*/  
/*Yardımcı problem değişkenler satırı hariç oluşturuldu*/  
/*-----*/  
yardimci_prob_ilk_satir1:addcol(degisken,yapay_degisken),  
yardimci_prob_ilk_satir:addcol(yardimci_prob_ilk_satir1,matrix(["çözüm"])  
,  
yardimci_prob_ilk_sutun:zeromatrix(toplam_satir_say+2,1),  
yardimci_prob_ilk_sutun[1][1]:" ",
```

Ek-3 (Devam). M yöntemi için maxima kodları

```
yardimci_prob_ilk_sutun[2][1]:"z",
for i:1 thru toplam_satir_say do yardimci_prob_ilk_sutun[i+2][1]:" ",
yardimci_prob_degisken1:addrow(yardimci_prob_ilk_satir,yardimci_prob_s
on),
yardimci_prob_degisken:addcol(yardimci_prob_ilk_sutun,yardimci_prob_de
gisken1),
disp("başlangıç simpleks tablosu"),
disp(yardimci_prob_degisken),
/* Problemdaki M lerin değerleri sıfırlanıyor*/
for i:1 thru R_sayisi do
yardimci_prob_son[1]:-M*yardimci_prob_son[i+1]+yardimci_prob_son[1],
/*Yeni z satırı ilgili yerlerde yerine yazılıyor*/
for i:1 thru toplam_sutun_say+1 do
yardimci_prob_degisken[2][i+1]:yardimci_prob_son[1][i],
/*Yardımcı probleme değişkenler satırı ve temel değişkenler satırı
oluşturulup ekleniyor*/
disp("uyumsuzluk giderildiği simpleks tablosu"),
disp(yardimci_prob_degisken),
load("simpleks_algoritma.mac"),
yardimci_cozum:simpleks_algoritma(yardimci_prob_degisken),
disp(yardimci_prob_degisken),
return(yardimci_cozum));
```

Ek-4.İki aşamalı Simpleks yönteminin maxima algoritması

*/*Bu program kısıtlamaları küçük eşittir, büyük eşittir ve eşittir'den en az iki tanesinden*

oluşan lineer programlama problemleri için

İki Aşamalı Simpleks algoritmasını oluşturur. Girilmesi gereken değişkenler

Input:

amaç fonksiyonunun katsayıları:P

küçük eşittir sınırlamalarının katsayıları: Ake

küçük eşittir sınırlamalarının karşı tarafları:bke

büyük eşittir sınırlamalarının katsayıları: Abe

büyük eşittir sınırlamalarının karşı tarafları:bbe

eşittir sınırlamalarının katsayıları: Aeq

eşittir sınırlamalarının karşı tarafları:beq

değişken isimleri: degisken

yapay değişken isimler:yapay_degisken

Output:

optimum değer matrisi:A

şeklindedir./*

simpleks_iki_asama(P,Aeq,beq,Ake,bke,Abe,bbq,degisken,yapay_degisken):=

block(

*/*Problemdeki sınırlamaların satır ve sütun sayılarına bakılarak probleme kaç tane dolgu değişken, kaç tane artık değişken ve yardımcı problemdeki değişken ekleneceğine karar veriliyor.*/*

P_sutun_say:length(transpose(P)),

*Aeq_sutun_say:length(transpose(Aeq)),/*Sınırlamalardaki eşitliklerin sayısı*/*

*if Aeq_sutun_say#0 then Aeq_satir_say:length(Aeq) else Aeq_satir_say:0,/*Dolgu değişkeni sayısı*/*

*/*Sınırlamalardaki eşitliklerin sütun sayısı. Sınırlamalardaki sütun sayısı değişken sayısı ile ilişkili olduğundan sadece bir tanesinin sütun sayısının belirlenmesi yeterlidir.*/*

Ek-4 (Devam).İki aşamalı Simpleks yönteminin maxima algoritması

```
Ake_sutun_say:length(transpose(Ake)),/*Sınırlamalardaki eşitliklerin sayısı*/
if Ake_sutun_say#0 then Ake_satir_say:length(Ake) else
Ake_satir_say:0,/*Dolgu değişkeni sayısı*/
Abe_sutun_say:length(transpose(Abe)),/*Sınırlamalardaki eşitliklerin
sayısı*/
if Abe_sutun_say#0 then Abe_satir_say:length(Abe) else
Abe_satir_say:0,/*Dolgu değişkeni sayısı*/
/*Problemdeki yapay değişkenlerin sayısı sınırlamalardaki büyük eşittirlerin
sayısı ve eşittirlerin sayısı toplamı kadardır.*/
R_sayisi:Aeq_satir_say+Abe_satir_say,/*Probleme eklenecek R
katsayılarının sayısı = ve >= içeren sınırlamaların sayısı kadardır*/
/*Problemdeki dolgu değişkeni sayısı küçük eşittir sınırlaması sayısı
kadardır.*/
dolgu_deg_say:Ake_satir_say,/*Dolgu değişkeni sayısı*/
/*Problemdeki artık değişken sayısı sınırlamalardaki büyük eşitlik sayısı
kadardır.*/
if Abe_sutun_say#0 then artik_deg_say:Abe_satir_say else
artik_deg_say:0,/*Dolgu değişkeni sayısı*/
/*Problemdeki sınırlamalar ile oluşan toplam satır sayısı toplam R_sayısı
büyük eşit ve eşitliklerin sayısı ve dolgu değişkenleri sayısı küçük eşittirlerin sayısını
göstermek üzere R_sayısı+dolgu_deg_say'dir*/
toplam_satir_say:R_sayisi+dolgu_deg_say,/*Sınırlamalar için toplam satır
sayısı*/
/*Problemdeki 1. sütundaki başlangıçtaki x_i değişkenleri, temel değişkenleri
ve son sütundaki çözümü gösteren sütunlar hariç
dolgu_degiskenleri+R_degiskenleri+artık_degiskenler mevcut olduğundan
problemdeki değişken isimleri gösteren toplam sütun sayısı değişken sayısı ile
ilişkilidir.*/
if Ake_sutun_say#0 then
toplam_sutun_say:Ake_sutun_say+dolgu_deg_say+artik_deg_say+R_sayisi
else (if Abe_sutun_say#0 then
```

Ek-4 (Devam).İki aşamalı Simpleks yönteminin maxima algoritması

```
toplam_sutun_say:Abe_sutun_say+dolgu_deg_say+artik_deg_say+R_sayisi
else
    toplam_sutun_say:Aeq_sutun_say+dolgu_deg_say+artik_deg_say+R_sayisi)
,/*Problemdeki 1. ve son sütun hariç sütun sayısı*/
/*-----*/
-----*/
/*İlk aşamada yardımcı problem oluşturulur. Bu problem çözülürülerek ana
probleme geçilir. Bu aşamada yardımcı problemi oluşturalım*/
/*Poblemdaki yardımcı denklemde ana değişkenler hariç diğer değişkenlerin
oluşturduğu satır ve sütunlar başlangıçta sıfırlanıyor. Bu elemanların satır ve
sütun sayısı toplam_satir_say ile toplam_sutun_say değişkenlerine atanmıştı.*/
yapay_kisim_sutun_say:dolgu_deg_say+artik_deg_say+R_sayisi,
/*-----*/
-----*/
/*Yardımcı problemin Simpleks tablosu oluşturuluyor.*/
sifirlar_mat:zeromatrix(toplam_satir_say,yapay_kisim_sutun_say),/*Yapay
değişkenlerin bölgeleri sıfırlanıyor*/
/*Yardımcı problemdeki sınırlamalar için eşitlikler, büyük eşittirler, küçük
eşittirler sıralamasındadır. Sınırlamaların katsayıları ekleniyor.*/
if (Abe_sutun_say=0) then yordimci_problem1:addrow(Aeq,Ake)
elseif (Aeq_sutun_say=0) then yordimci_problem1:addrow(Abe,Ake)
elseif (Ake_sutun_say=0) then yordimci_problem1:addrow(Aeq,Abe)
else yordimci_problem1:addrow(Aeq,Abe,Ake),/*Sınırlamaların katsayıları*/
/*Problemin son sütunu 2. satır çözüm satırı olacağından ilk değeri 0'dır.
Diğer değerleri ise sınırlamaların karşı tarafına karşılık gelen değerlerdir. Bunlar
aynı sınırlamlar sırası ile girilmelidir.*/
if (Abe_sutun_say=0) then b:addrow(matrix([0]),beq,bke)
elseif (Aeq_sutun_say=0) then b:addrow(matrix([0]),bbe,bke)
elseif (Ake_sutun_say=0) then b:addrow(matrix([0]),beq,bbe)
else b:addrow(matrix([0]),beq,bbe,bke),/*Sınırlamaların katsayıları*/
/*Problemin son sütunu, değişken satırı hariç.*/
```

Ek-4 (Devam).İki aşamalı Simpleks yönteminin maxima algoritması

*/*Simpleks tablosunun ilk 2 satırı ve 1. ve son sütunları hariç sınırlamaları içeren kısmı problemin sınırlamaları ile yapay değişkenlerin sıfırlarından oluşan kısımdır.*/*

*yardimci_problem:addcol(yardimci_problem1,sifirlar_mat),/*Sınırlamalar katsayılar matrisi*/*

*/*Yardımcı problemin 2. satırında sadece yapay R değişkenlerine içereceğinden ilk olarak bunların hepsi sıfır alınıyor. Bunların eleman sayısı 1. ve son sütun hariç toplam değişken sayısı kadardır. Toplam değişken sayısı toplam_sutun_say ile başlangıçtaki değişkenlerin toplamı kadardır.*/*

*yardimci_problem_ikinci_satir:zeromatrix(1,P_sutun_say+yapay_kisim_sutun_say),/*Problemdeki ikinci satır sıfırlanıyor*/*

*/*Yardımcı problemdeki sıfırlanan yerler dolduruluyor. Bunun için ilk artık değişkenlerin katsayıları -1 yapılıyor. Artık değişkenler eşitlik içeren satırlardan sonra eklenmelidir.*/*

for i:1 thru artik_deg_say do
if *Ake_sutun_say#0* *then*
yardimci_problem[Aeq_satir_say+i][Ake_sutun_say+i]:-1
else *yardimci_problem[Aeq_satir_say+i][Ake_sutun_say+i]:-1,/*Artık değişkenlerin katsayıları -1 yapılıyor*/*

*/*Yardımcı problemdeki dolgu değişkenlerin katsayıları +1 olmalıdır. Problemdeki dolgu değişkenleri eşitlik ve küçük eşitliklerden sonra verilmiş idi.*/*

for i:1 thru dolgu_deg_say do
yardimci_problem[Aeq_satir_say+Abe_satir_say+i][Ake_sutun_say+artik_deg_say+i]:1,

*/*Dolgu değişkenlerinin katsayıları +1 yapılıyor*/*

*/*Yardımcı problemdeki R'lerin katsayıları +1 olmalıdır. Yardımcı problemin ilk satırındaki R lerin katsayıları ise 1 olarak girilmelidir*/*

for i:1 thru R_sayisi do

(yardimci_problem[i][P_sutun_say+artik_deg_say+dolgu_deg_say+i]:1,

Ek-4 (Devam).İki aşamalı Simpleks yönteminin maxima algoritması

```

yardimci_problem_ikinci_satir[1][P_sutun_say+artik_deg_say+dolgu_deg_
say+i]:1),
/*R lerin katsayıları giriliyor*/
/*Yardımcı problemin ikinci satırı ile alt satırları birleştiriliyor. Bunlar b
sütununu ve temel değişkenler sütununu içermemektedir.*/
yardimci_prob_son1:addrow(yardimci_problem_ikinci_satir,yardimci_prob_l
em),
/*Yardımcı problemin sınırlamalar matrisi*/
/*Yardımcı problem 1. satır hariç son sütunu çözümü ve sınırlamaların karşı
tarafını içerecek şekilde oluşturuluyor.*/
yardimci_prob_son:addcol(yardimci_prob_son1,b),/*Yardımcı problemin
son sütunu*/
/*Yardımcı problem değişkenler satırı hariç oluşturuldu*/
/*-----*/
yardimci_prob_ilk_satir1:addcol(degisken,yapay_degisken),
yardimci_prob_ilk_satir:addcol(yardimci_prob_ilk_satir1,matrix(["çözüm"])
),
yardimci_prob_ilk_sutun:zeromatrix(toplam_satir_say+2,1),
yardimci_prob_ilk_sutun[1][1]:" ",
yardimci_prob_ilk_sutun[2][1]:"z",
for i:1 thru toplam_satir_say do yardimci_prob_ilk_sutun[i+2][1]:" ",
yardimci_prob_degisken1:addrow(yardimci_prob_ilk_satir,yardimci_prob_s
on),
yardimci_prob_degisken:addcol(yardimci_prob_ilk_sutun,yardimci_prob_de
gisken1),
disp("başlangıç simpleks tablosu"),
disp(yardimci_prob_degisken),
/*-----*/
/*Yardımcı problemdeki başlangıçta oluşan uyumsuzluk gideriliyor.*/
/*Yardımcı problemdeki R lerin değerleri sıfırlanıyor*/
for i:1 thru R_sayisi do

```

Ek-4 (Devam).İki aşamalı Simpleks yönteminin maxima algoritması

```

yardimci_prob_son[1]:-yardimci_prob_son[i+1]+yardimci_prob_son[1],
/*Yardımcı probleme değişkenler satırı ve temel değişkenler satırı
oluşturulup ekleniyor*/

/*İlk aşamada yardımcı problem oluşturuldu. Şimdi bu yardımcı problem için
Simpleks algoritması çalıştırılarak yardımcı problem çözdürülecek. Eğer son
aşamada gelen matrisin çözüm değeri sıfıra eşit ise ikinci aşamaya geçilecek.*/

/*Yeni z satırı ilgili yerlerde yerine yazılıyor*/
for i:1 thru toplam_sutun_say+1 do
yardimci_prob_degisken[2][i+1]:yardimci_prob_son[1][i],
/*Yardımcı probleme değişkenler satırı ve temel değişkenler satırı
oluşturulup ekleniyor*/

/*-----*/
disp("uyumsuzluk giderildiği simpleks tablosu"),
disp(yardimci_prob_degisken),
/*-----*/

/*Yardımcı problem Simpleks algoritması ile çözdürülüyor*/
load("simpleks_algoritma.mac"),
yardimci_cozum:simpleks_algoritma(yardimci_prob_degisken),
/*-----*/

/*İkinci aşamaya geçilip geçilmeyeceğini belirlemek için yardımcı çözümün
2. satır son sütun elemanına yani toplam_sutun_say+Aeq_sutun_say+2 sütun
elemanına bakılıyor*/

if yardimci_cozum[2][toplam_sutun_say+2]=0 then (disp("ikinci aşamaya
geçiliyor"),
/*-----*/

/*Esas problemin amaç fonksiyonun katsayıları son simpleks tablosunun z
satırında ilgili yerlere atanır.*/

yardimci_problem_satir_say:length(yardimci_cozum),/*yardımcı problemin
satır sayısı*/

```

Ek-4 (Devam).İki aşamalı Simpleks yönteminin maxima algoritması

```

yardimci_problem_sutun_say:length(transpose(yardimci_cozum)),/*yardimci
problemin sütun sayısı*/

yardimci_cozum_transpoz:transpose(yardimci_cozum),/*Yardimci
problemin transpozu. Bu transpoz ile birim sütun var mı diye kontrol sağlanacak*/

c:zeromatrix(yardimci_problem_satir_say,yardimci_problem_sutun_say-
R_sayisi),

/*yardimci problemdeki R sütunu hariç birimlere bakılacak sütunları c
matrisine atıyor */

for i:1 thru yardimci_problem_satir_say do
(for j:1 thru yardimci_problem_sutun_say-R_sayisi-1 do
c[i][j]:yardimci_cozum[i][j]),
for i:1 thru yardimci_problem_satir_say do
c[i][yardimci_problem_sutun_say-
R_sayisi]:yardimci_cozum[i][yardimci_problem_sutun_say],
c_sutun_say: yardimci_problem_sutun_say-R_sayisi,/*c matrisinin sütun
sayısı esas matrisin R sütunu hariç diğer elemanları olduğundan esas matrisin sütun
sayısında 1 eksiktir*/

for i:1 thru length(transpose(P)) do c[2][i+1]:P[1][i],/* ikinci aşamada
esas problemin z satırı P vektörünün elemanları olarak alınacağından bu elemanlar
c nin 2. satırında ilgili yere atanıyor*/

/*-----*/
disp("İkinci aşama başlangıç matrisi:"),
disp(c),
/*-----*/

/*İkinci aşamada ilk olarak uyumsuzluk var mı yok mu diye kontrol ediliyor.
Yani eğer ikinci aşama başlangıç Simpleks tablosunda birime karşılık gelen bir esas
değişken varsa çözüm satırı sıfır olduğundan uyumsuzluk meydana
gelmiş olur. İlk olarak bu düzeltilmeli sonra Simpleks algoritması
uygulanmalıdır.*/

d:zeromatrix(toplam_satir_say,P_sutun_say),/*c matrisinde birim vektör
var mı yok mu diye uyumluluğu kontrol etmek için c matrisinin başlangıçtaki x_i

```

Ek-4 (Devam).İki aşamalı Simpleks yönteminin maxima algoritması
değişkenlerine karşılık gelen sütunları *d* matrisine atanarak karşılaştırma
yapılacaktır*/

```
for i:1 thru toplam_satir_say do  
  (for j:1 thru P_sutun_say do d[i][j]:c[i+2][j+1]),/* c nin ilgili sütunları d  
matrisine atanıyor*/
```

```
e:submatrix(1,c,1),/*c matrisinin 1. satırı ile 1. sütunu olan değişkenlerin  
isimlerinin yer aldığı yerler çıkartılması ile elde edilen matris e ye atanıyor. Bu  
sayede eğer sıfırlama işlemi yapılacaksa bunun üzerinden yapılacak*/
```

```
I:ident(toplam_satir_say),/*Birim vektörleri tutan matris. Karşılaştırma  
için gerekli olan matris.*/
```

```
St:transpose(d),/*d matrisinin sütunları ile karşılaştırılacağından d nin  
transpozu alınıyor*/
```

```
k:0,/*Temel değişkenlerin sayısını gösteriyor. Başlangıçta 0 alınıyor*/
```

```
kill(yer),/*sıfırlanacak yerler başlangıçta yok ediliyor*/
```

```
for i:1 thru toplam_satir_say do (for j:1 thru P_sutun_say do  
  (if St[j]=I[i] then (k:k+1,yer[k]:j))),/*sıfırlanacak yerler yer değişkeninin  
içerisinde saklanıyor*/
```

```
/*eğer sıfırlama işlemi varsa gerçekleştiriliyor*/
```

```
if k#0 then (for i:1 thru k do (e[1]:e[1]-e[yer[i]+1]*e[1][i]),
```

```
for i:1 thru yordimci_problem_satir_say-1 do
```

```
  (for j:1 thru c_sutun_say-1 do c[i+1][j+1]:e[i][j]),
```

```
/*-----*/
```

```
disp("uyumsuzluğun giderildiği matris"),disp(c),
```

```
/*-----*/
```

```
/*ikinci aşama simpleks tablosu önceden hazırlanan simpleks_algoritma  
programı kullanılarak çözülür.*/
```

```
load("simpleks_algoritma.mac"),
```

```
sonuc:simpleks_algoritma(c));
```

Ek-5. Dual Simpleks Algoritmasının maxima ortamında hazırlanan kodu

*/*Bu program Dual simpleks algortiması ile problemin optimum değerini belirler.*

Input: A: Dual simpleks algortimasının uygulanacağı matris

Output: A: Dual simpleks algoritmasının uygulanması sonucunda oluşan matris

Dual simpleks algoritmasında önceden verilen simpleks algoritmasına benzer şekilde giriş elamanı verileceğini not edelim./*

*dual_simpleks_algoritma(A):=block(/*A matrisinin 1. satırı değişkenleri göstermekte, 1. sütunu temel değişkenleri göstermektedir*/*

*/*Başlangıçta temel değişkenler bilinmediği için onların kısmı boş bırakılmıştır*/*

*/*A matrisinin 2. satırı maksimize edilecek olan amaç fonksiyonunun katsayıları olarak girilmelidir*/*

*/*Diğer satırları ise her bir sınırlamanın katsayıları olarak girilmelidir*/*

*A_satir_sayisi:length(A),/*Ana matrisin satır sayısı*/*

*A_sutun_sayisi:length(transpose(A)),/*Ana matrisin sütun sayısı*/*

/-----*/*

*/*S matrisi temel değişkenlerin bulunacağı matristir.*/*

*/*Temel değişkenler, A matrisinin 3. sütunlarından son sütunlarına kadar olan vektörlerden birim elemanlara karşılık gelen vektörlere karşılık olanlardan seçilir.*/*

*/*S matrisi için A matrisinin 3. satır ile son satır arası 2. sütun ile sondan bir önceki sütun arası elemanlardan oluşan matristir.*/*

*S_satir_sayisi:A_satir_sayisi-2,/*A matrisinin 3. satır ile son satır arası alındığı için*/*

*S_sutun_sayisi:A_sutun_sayisi-2,/*A matrisinin 2. sütun ile sondan bir önceki sütun arası alındığı için*/*

*devam:1,/*Eğer devam 1 değerine eşitse döngü devam edeceği 0 ise döngünün biteceğini gösteriyor. Döngü durdurma değişkeni*/*

asama:1,/ Ekrana yazdırırken j. aşamanın sonu olduğunu yazdıracak*/*

Ek-5 (Devam).Dual Simpleks Algoritmasının maxima ortamında hazırlanan kodu

*I:ident(S_satir_sayisi),/*Birim vektörleri çekmek için kullanılacak birim matris. Bu sayede temel değişkenler belirlenmeye çalışılacak*/*

*S:zeromatrix(S_satir_sayisi,S_sutun_sayisi),/*S matrisi başlangıçta sıfırlardan oluşan matris olarak tanımlanıyor*/*

*/*S matrisi A'nın elemanları alınarak oluşturuluyor:*/*

*for i:1 thru S_satir_sayisi do (for j:1 thru S_sutun_sayisi do S[i][j]:A[i+2][j+1]),/*S matrisi oluşturuluyor.*/*

/-----*/*

*/*S matrisinin birim vektörlere karşılık gelen elemanları temel değişkenler olacağından*

ilk olarak birim vektörlere karşılık gelen elemanlar belirlenmelidir/*

*/*Bunun için aynı büyüklükte bir birim matrisinin her bir satırı ile S sütunun her bir sütunu eşleştirilmek şartıyla eşit olduğu durumda temel değişken ortaya çıkarılmış olur.*/*

*St:transpose(S),/*S matrisinin transpozu. Maxima'da S matrisinin sütunlarını çekmek zor iken S'nin satırlarını S[i] komutu ile çekebiliriz. Bu yüzden S'nin transpozunu alıyoruz.*/*

*/*S'nin birim vektörlere karşılık gelen sütunları bulunuyor:*/*

*k:0,/*Temel değişkenlerin sayısını gösteriyor. Başlangıçta 0 alınıyor*/*

kill(yer),

for i:1 thru S_satir_sayisi do (for j:1 thru S_sutun_sayisi do (if St[j]=I[i] then (k:k+1,yer[k]:j))),

*/*S'nin sütunları ile birim vektörünün bütün sütunları karşılaştırılıp eşitlik olup olmadığına bakılıyor. Eğer eşitlik varsa temel değişken bulunmuş olup temel değişken sayısı 1 artırılıyor ve bu sütunun yeri 'yer' değişkenine atılıyor. Çünkü temel değişkenin hangi değişkene karşılık geldiği bu sütun ile belirlenecek.*/*

*/*Temel değişkenler St'nin satırlarında birim vektörlere karşılık gelen vektörlerdi. Bu vektörlerin yer değişkeninde tutuldu. Şimdi bunların hangi elemanlara karşılık geldiği A matrisinin 1. satırından belirlenmektedir.Ayrıca bu*

Ek-5 (Devam).Dual Simpleks Algoritmasının maxima ortamında hazırlanan kodu vektörlere karşılık gelen çözüm de verilmektedir. Son olarak ise temel değişkenlerin isimleri boş bırakılmıştı. Onların yerine isimleri yazılıyor.*/

```
temel_degisken:zeromatrix(1,k),/*Temel değişkenleri tutan matris*/
temel_degisken_deger:zeromatrix(1,k),/*Temel değişkenlerin değerlerini
tutan matris*/
for i:1 thru k do (temel_degisken[1][i]:A[1][yer[i]+1],
temel_degisken_deger[1][i]:A[i+2][A_sutun_sayisi],A[i+2][1]:A[1][yer[i]+
1]),/*Temel değişkenlerin hangi değişkenler olduğu belirlendi*/
disp("Temel değişkenler:",temel_degisken),/*Temel değişkenler yazdırıldı*/
disp("Temel değişkenlerin değerleri:",temel_degisken_deger),
disp(A),
/*-----*/
/*Simpleks algoritmasını uygulayacağımız matrisi oluşturuyoruz.
Bu matris A matrisinin 3. satırından son satırına kadar olan matristir:*/
Simp_satir_sayisi:A_satir_sayisi-1,/*Simpleks matrisin satır sayısı A
matrisinden ilk satırının çıkarılması ile elde edilir*/
Simp_sutun_sayisi:A_sutun_sayisi-1,/*Simpleks matrisin sütun sayısı A
matrisinden ilk sütununun çıkarılması ile elde edilir*/
Simp_matris:zeromatrix(Simp_satir_sayisi,Simp_sutun_sayisi),
for i:1 thru Simp_satir_sayisi do (for j:1 thru Simp_sutun_sayisi do
Simp_matris[i][j]:A[i+1][j+1]),/*Simpleks matrisi oluşturuluyor.*/
/*İlk olarak pivot sütun belirleniyor. Bu sütun Simp_matris in 1. satırındaki
negatif elemanlardan mutlak değerce en büyük olanına karşılık gelir.*/
while devam=1 do(
/*(-----*/
min_deger:0,/*Başlangıçta min değerimizi sıfır yapıyoruz*/
min_satir:0,/*Başlangıçta pivot satırımızı sıfır yapıyoruz*/
for i:1 thru Simp_satir_sayisi-1 do
(if min_deger>Simp_matris[i+1][Simp_sutun_sayisi]
```

Ek-5 (Devam).Dual Simpleks Algoritmasının maxima ortamında hazırlanan kodu

```

then(min_deger:Simp_matris[i+1][Simp_sutun_sayisi],min_satir:i+1)),/*son
satırdaki minimum eleman ve minimum değer belirlenmesi aşaması*/

/*Pivot satıra karşılık gelen A matrisindeki değişken sonraki aşamada temel
değişken olacaktır.*/

disp("Çıkan değişken:",A[min_satir+1][1]),/*Simpleks algoritmasında giren
değişken*/

/*Pivot satır min_satir olarak belirlendi*/

/*-----*/

/*İkinci işlem olarak pivot sütün belirlenir. Pivot sütün Simp_matris in ilk
satırındaki negatif elemanlar ile Pivot satırın negatif elemanlarının oranlanması ile
belirlenir. Oranı en küçük olan pivot satırdır.*/

bolme:zeromatrix(Simp_sutun_sayisi-1,1),
for i:1 thru Simp_sutun_sayisi-1 do (if Simp_matris[min_satir][i]#0 then
bolme[i][1]:Simp_matris[1][i]/Simp_matris[min_satir][i]),
min_satdeger:10^10,/*Başlangıçta min değerimizi sıfır yapıyoruz*/
min_sutun:0,/*Başlangıçta pivot sütunumuzu sıfır yapıyoruz*/
for i:1 thru Simp_sutun_sayisi-1 do (if bolme[i][1]>0 then (if
min_satdeger>bolme[i][1] then(min_satdeger:bolme[i][1],min_sutun:i))),/*son
satırdaki minimum eleman ve minimum değer belirlenmesi aşaması*/

/*Pivot sütuna karşılık gelen A matrisindeki değişken sonraki aşamada temel
dışı değişken olacaktır.*/

disp("Giren değişken",A[1][min_sutun+1]),
A[min_satir+1][1]:A[1][min_sutun+1],

/*-----*/

/*Üçüncü işlem olarak pivot elemanı 1 yapıp Simpleks matris Gauss-Jordan
işlemine hazır hale getirilecektir.*/

Simp_matris[min_satir]:Simp_matris[min_satir]/Simp_matris[min_satir][mi
n_sutun],/*Pivot satır olan S[min_satir] pivot eleman olan
S[min_satir][min_sutun]bölünerek pivot eleman 1 yapılıyor*/

/*-----*/

```

Ek-5 (Devam).Dual Simpleks Algoritmasının maxima ortamında hazırlanan kodu

*/*Dördüncü işlem olarak Gauss-Jordan işlemi Simpleks matrisine uygulanır.*

Bu işlem ile bir Simpleks aşaması sona ermiş olur./*

```
for i:1 thru Simp_satir_sayisi do (if i#min_satir then
Simp_matris[i]:Simp_matris[i]-
Simp_matris[min_satir]*Simp_matris[i][min_sutun]),
disp([asama, ". aşamanın sonu:"]),
/*-----*/
/*A matrisi yeni temel değişkenler ve elemanlara göre yeniden yazılır.*/
for i:1 thru Simp_satir_sayisi do (for j:1 thru Simp_sutun_sayisi do
A[i+1][j+1]:Simp_matris[i][j]),
disp(A),
temel_degisken:zeromatrix(1,k),/*Temel değişkenleri tutan matris*/
temel_degisken_deger:zeromatrix(1,k),/*Temel değişkenlerin değerlerini
tutan matris*/
for i:1 thru A_satir_sayisi-2 do (temel_degisken[1][i]:A[i+2][1],
temel_degisken_deger[1][i]:A[i+2][A_sutun_sayisi]),
disp("Temel değişkenler:",temel_degisken),/*Temel değişkenler yazdırıldı*/
disp("Temel değişkenlerin değerleri:",temel_degisken_deger),
disp("Temel değişkenlere karşılık gelen çözüm:",A[2][A_sutun_sayisi]),
/*-----*/
/*Son satırdaki negatif eleman olup olmadığı kontrol ediliyor*/
dongu:0,
for i:1 thru Simp_satir_sayisi-1 do (if
Simp_matris[i+1][Simp_sutun_sayisi]<0 then dongu:dongu+1),
if dongu>0 then devam:1 else devam:0,
/*Son satırdaki negatif eleman olup olmadığı kontrol edildi*/
asama:asama+1),
return(A));
```

Ek-6.Dual Simpleks Algoritmasının uygulanacağı problem algoritması

*/*Bu program dual simpleks algoritması ile doğrusal programlama problemini çözer.*

Eğer problem tamamen küçük eşittir kısıtlamalarından oluşuyorsa problem dual probleme dönüştürülerek çözülür. Diğer çözümler için dual simpleks algoritması direkt olarak uygulanır.Girilmesi gereken değişkenler

Input:

amaç fonksiyonunun katsayıları:P

küçük eşittir sınırlamalarının katsayıları: Ake

küçük eşittir sınırlamalarının karşı tarafları:bke

büyük eşittir sınırlamalarının katsayıları: Abe

büyük eşittir sınırlamalarının karşı tarafları:bbe

eşittir sınırlamalarının katsayıları: Aeq

eşittir sınırlamalarının karşı tarafları:beq

değişken isimleri: degisken

yapay değişken isimler:yapay_degisken

prob: "max" veya "min" giriniz

Output:

optimum değer matrisi: A

şeklindedir./*

*simpleks_dual(P,Aeq,beq,Ake,bke,Abe,bbq,degisken,problem):=block(Aeq_sutun_say:length(transpose(Aeq)),/*Sınırlamalardaki eşitliklerin sayısı*/*

*if Aeq_sutun_say#0 then Aeq_satir_say:length(Aeq) else Aeq_satir_say:0,/*Dolgu değişkeni sayısı*/*

*/*Sınırlamalardaki eşitliklerin sütun sayısı. Sınırlamalardaki sütun sayısı değişken sayısı ile ilişkili olduğundan sadece bir tanesinin sütun sayısının belirlenmesi yeterlidir.*/*

*Ake_sutun_say:length(transpose(Ake)),/*Sınırlamalardaki eşitliklerin sayısı*/*
if Ake_sutun_say#0 then Ake_satir_say:length(Ake) else Ake_satir_say:0,
*Abe_sutun_say:length(transpose(Abe)),/*Sınırlamalardaki eşitliklerin sayısı*/*

if Abe_sutun_say#0 then Abe_satir_say:length(Abe) else Abe_satir_say:0,

Ek-6 (Devam).Dual Simpleks Algoritmasının uygulanacağı problem algoritması

*/*Eğer problem sadece küçük eşittir kısıtlamalarını içeriyorsa probleme dual simpleks algoritmasını uygulamak için problem dual yapıya çevriliyor*/*

```

    if (Abe_sutun_say=0 and Aeq_sutun_say=0) then (A_yeni:-transpose(Ake),
    b_yeni:transpose(P),P_yeni:-transpose(bke),I:ident(Ake_sutun_say),
    A_alt:addcol(A_yeni,I,b_yeni),A_ilk:addcol(P_yeni,zeromatrix(1,Ake_sutun_
say+1)),
    A_isim:addcol(degisken,yapay_degisken,matrix(["çözüm"])),
    A_sol_yan:zeromatrix(Ake_sutun_say,1),
    for i:1 thru Ake_sutun_say do A_sol_yan[i][1]:" ",
    A_sol:addrow(matrix([" "],["z"]),A_sol_yan),
    A:addrow(A_isim,A_ilk,A_alt),A:addcol(A_sol,A)) else (for i:1 thru
Aeq_satir_say do ((if Ake_sutun_say#0 then
(Ake:addrow(Ake,Aeq),bke:addrow(bke,beq)) else if Aeq_sutun_say#0 then
(Ake:Aeq,bke:beq)),
    (if Abe_sutun_say#0 then (Abe:addrow(Abe,Aeq),bbe:addrow(bbe,beq)) else
if Aeq_sutun_say#0 then (Abe:Aeq,bbe:beq))),disp(Ake),
    Ake_yeni_sutun_say:length(transpose(Ake)),
disp(transpose(Ake)),Abe_yeni_sutun_say:length(transpose(Abe)),
    (if Ake_yeni_sutun_say#0 then Ake_yeni_satir_say:length(Ake) else
Ake_yeni_satir_say:0),
    (if Abe_yeni_sutun_say#0 then Abe_yeni_satir_say:length(Abe) else
Abe_yeni_satir_say:0),disp(Ake_yeni_satir_say),birim_top_satir_say:Ake_yeni_satir_
_say+Abe_yeni_satir_say,II:ident(birim_top_satir_say),
    Abe:-Abe,bbe:-bbe,disp(II),
    (if (Ake_yeni_sutun_say#0 and Abe_yeni_sutun_say#0) then
(A_alt:addrow(Abe,Ake), A_alt:addcol(A_alt,II),b_alt:addrow(bbe,bke)) else if
Ake_yeni_sutun_say=0 then (A_alt:addcol(Abe,II),b_alt:bbe)
    else (A_alt:addcol(Ake,II),b_alt:bke)),A_isim:addcol(matrix([" "]),
degisken,yapay_degisken,matrix(["çözüm"])),
    A_sol_alt:zeromatrix(birim_top_satir_say,1),disp(A_sol_alt),for i:1

```

Ek-6 (Devam).Dual Simpleks Algoritmasının uygulanacağı problem algoritması

```
thru birim_top_satir_say do A_sol_alt[i][1]:" ",A_alt:addcol(A_sol_alt,A_alt,b_alt),
    A_ilk:addcol(matrix(["z"]),P,zeromatrix(1,birim_top_satir_say),matrix([0])),
    A:addrow(A_isim,A_ilk,A_alt)),
disp(A),
load("dual_simpleks_algoritma.mac"),
dual_simpleks_algoritma(A),
if problem="max" then
    (A_sutun_say:length(transpose(A)),
    dongu:0,
    for i:1 thru A_sutun_say-2 do (if A[2][i+1]<0 then dongu:dongu+1),
    if dongu>0 then devam:1 else devam:0,
    if                                devam=1                                then
    (load("simpleks_algoritma.mac"),simpleks_algoritma(A))));/*Bu program dual
simpleks algoritması ile doğrusal programlama problemini çözer.*/
```

Ek-7. Simpleks yönteminin matrisel gösterimi

/ X vektörü: Değişkenlerin isimlerinin tutulduğu satır vektörü*

CI vektörü: Problemdaki değişkenlerin katsayılarından oluşan satır vektörü

CII vektörü: Probleme eklenen yapay değişkenlerin katsayılarından oluşan satır vektörü

b vektörü: Sınırlamaların karşı tarafındaki değerlerinden oluşan sütun vektörü

P matrisi: Yapay değişkenlerinde sisteme eklenmesiyle oluşan katsayılar matrisi/*

*simpleks_matrisel(X,CI,CII,b,P):=block(/*P matrisinin transpozu belirleniyor*/*

A_trp:transpose(P),

*/*P matrisinin satır sayısı belirleniyor*/*

A_satir_sayisi:length(P),

*/*P matrisinin sütun sayısı belirleniyor*/*

A_sutun_sayisi:length(A_trp),

*/*Birim matris oluşturuluyor. Bu matrisin oluşturulma nedeni temel değişkenlerin belirlenmesidir. Eğer P matrisinin sütunlarında (veya P matrisinin transpozunun satırlarında) birim matrisinin satırlarından herhangi biri mevcut ise o sütundaki değişken temel değişkendir.*/*

birimler:ident(A_satir_sayisi),

*/*P'nin birim vektörlere karşılık gelen sütunları bulunuyor:*/*

*k:0,/*Temel değişkenlerin sayısını gösteriyor. Başlangıçta 0 alınıyor*/*

kill(yer),

for i:1 thru A_satir_sayisi do (for j:1 thru A_sutun_sayisi do (if A_trp[j]=birimler[i] then (k:k+1,yer[k]:j))),

*/*k: Temel değişken sayısı*

yer: k. temel değişkeni P matrisinin kaçınıcı sütununa karşılık geldiğini gösteren matris

temel_degiskenler_matrisi: Temel değişkenlerin isimlerini tutan sütun vektörü

Ek-7 (Devam). Simpleks yönteminin matrisel gösterimi

temel_disi_degiskenler_matrisi: Temel dışı değişken isimlerini tutan sütun vektörü
temel_disi_degisken_yer: i. temel dışı değişkenin P matrisinin kaçınıcı
sütununa karşılık geldiğini gösteren satır vektörü/*
temel_degiskenler_matrisi:zeromatrix(k,1),
*temel_disi_degiskenler_matrisi:zeromatrix(A_sutun_sayisi-k,1),/*Temel dışı*
değişken sayısı tüm değişken sayısından temel değişken sayısının çıkarılması ile elde
edilir./*
temel_disi_degisken_yer:zeromatrix(A_sutun_sayisi-k,1),
*/*C_B: Yöntemdeki C_B vektörüdür. Yani temel değişkenlerin C vektöründeki*
değerlerinden oluşan vektördür.
B: Yöntemdeki B matrisidir. Yani temel değişkenlerin P matrisindeki
sütunlarından oluşan matristir.
B_trp : B matrisinin transpozudur. Bu genel olarak temel değişken değiştiği
durumda B matrisindeki yeni temel dışı değişken yerine yeni temel değişkene karşılık
gelen vektörün eklenmesinde kullanılmaktadır.
invB: B matrisinin tersidir. Simpleks matrisinin oluşturulmasında
kullanılır./*
C_B:zeromatrix(1,k),
B_trp:zeromatrix(k,A_satir_sayisi),
*/*Temel değişkenler ilgili matriste yerine yazılıyor. Temel değişkenlerin*
sayısı (k) ve yerleri(yer) önceden belirlenmişti. Bunlara karşılık gelen temel
değişkenler belirlenerek ilgili matrise atanıyor. Ayrıca bu temel değişkenlere karşılık
gelen B matrisinin transpozu ve C_B vektörü de aynı anda belirleniyor/*
for i:1 thru k do
(temel_degiskenler_matrisi[i][1]:X[1][yer[i]],B_trp[i]:A_trp[yer[i]],
C_B[1][i]:C[1][yer[i]]),
*jj:0,/*Temel dışı değişken sayısını tutan değişken.*/*
*/*Başlangıçtaki temel dışı değişkenler belirlenerek ilgili matrise atanıyor.*
Bunun için X vektörünün elemanlarına bakılarak temel elemanlara karşılık gelen bir

Ek-7 (Devam). Simpleks yönteminin matrisel gösterimi

eleman varsa sayac değişkeni sıfırlanıyor. Eğer yoksa sayac değeri 1 arttırılarak o değişkenin temel değişken matrisinde olup olmadığı kontrol ediliyor./*

```
for i:1 thru A_sutun_sayisi do (sayac:0, (for j:1 thru k do
  (if X[1][i]=temel_degiskenler_matrisi[j][1] then sayac:1)), if sayac=0 then
  (jj:jj+1,temel_disi_degiskenler_matrisi[jj][1]:X[1][i],temel_disi_degisken_yer[jj][1
  ]:i ),
```

*/*B_trp_temel_disi : P matrisinin temel dışı elemanlarına karşılık gelen B_temel_disi matrisinin transpozudur. Bu genel olarak temel değişken değiştiği durumda B_temel_disi matrisindeki yeni temel dışı değişken yerine yeni temel değişkene karşılık gelen vektörün eklenmesinde kullanılmaktadır.*

C_B_temel_disi: Yöntemdeki temel dışı değişkenlerin C vektöründeki değerlerinden oluşan vektördür.

Not etmek gerekirse bu temel dışı matrisler döngünün sonlanıp sonlanmadığını ve giren değişkenin hangi değişken olacağını belirlemede kullanılmaktadır./*

```
B_trp_temel_disi:zeromatrix(jj,A_satir_sayisi),
C_B_temel_disi:zeromatrix(1,jj),
for i:1 thru jj do (B_trp_temel_disi[i]:A_trp[temel_disi_degisken_yer[i][1]],
C_B_temel_disi[1][i]:C[1][temel_disi_degisken_yer[i][1]]),
B:transpose(B_trp),
invB:invert(B),
B_temel_disi:transpose(B_trp_temel_disi),
```

*/*z_satir_degisken: Simpleks matrisinin XI lara yani z satırında başlangıçtaki değişkenlere (x_i lere) karşılık gelen değerleri tutan satır vektörüdür.*

z_satir_yapay: Simpleks matrisinin XII lara yani z satırında yapay değişkenlere (s_i lere) karşılık gelen değerleri tutan satır vektörüdür.

cozum: z satırındaki çözüm değeridir./*

```
z_satir:C_B.invB.P-C,
```

```
cozum:C_B.invB.b,
```

*/*Başlangıç Simpleks tablosu oluşturuluyor.*/*

Ek-7 (Devam). Simpleks yönteminin matrisel gösterimi

*alt_kisim:invB.P,/*Simpleks tablosundaki başlangıçtaki değişkenlere (x_i lere) karşılık gelen kısmın katsayılarını gösteren matristir.*/*

*b_kisim:invB.b,/*Simpleks tablosunda b lerin (yani kısıtlamaların karşı taraflarının) oluşturduğu sütun vektörüdür*/*

disp("başlangıç matrisi:"),

*ilk_satir:addcol(matrix([" "]),X,matrix(["çözüm"])),/*Simpleks tablosundaki değişken isimlerinin verildiği satır*/*

*ikinci_satir:addcol(matrix(["z"]),z_satir,matrix([cozum])),/*Simpleks tablosundaki z satırı*/*

*alt_satirlar:addcol(temel_degiskenler_matrisi,alt_kisim,b_kisim),/*Simpleks tablosundaki kısıtlamaların oluşturduğu satırlar*/*

*simp_matris_tum:addrow(ilk_satir,ikinci_satir,alt_satirlar),/*Tüm tablo bir araya getiriliyor*/*

*disp(simp_matris_tum),/*Tüm tablo ekrana yazdırılıyor*/*

*optimumluk:C_B.invB.B_temel_disi-C_B_temel_disi,/*Problemdeki optimumluk koşulu bu vektörün içerisinde negatif eleman olmamasıdır. Ona bakılıyor*/*

*dongu:1,/*Simpleks yönteminin kaç döngü yaptığını tutan değişken*/*

*sayac:0,/*optimumluğun sağlanıp sağlanmadığını sayan değişken. Eğer optimum vektöründeki tüm elemanlar (temel dışı değişkenlerin sayısı kadar) eğer pozitif ise optimumluk sağlanmış demektir. Aksi halde negatif eleman varsa Simpleks yöntemi uygulanmaya devam etmelidir.*/*

for i:1 thru jj do (if optimumluk[1][i]>=0 then sayac:sayac+1),

*/*Eğer sayac değeri temel dışı değişken sayısına eşit ise optimumluk sağlanmış olur. Kontrol sağlanıyor.*/*

if sayac=jj then dongu:0,

*asama:1,/*Kaç Simpleks aşamasının probleme uygulandığını tutan değişken.*/*

while dongu=1 do (

*min_deger:0,/*Başlangıçta min değerimizi sıfır yapıyoruz*/*

*min_sutun:0,/*Başlangıçta pivot sütunumuzu sıfır yapıyoruz*/*

Ek-7 (Devam). Simpleks yönteminin matrissel gösterimi

*/*Optimumluk koşulunda optimumluk vektöründeki negatif elemanlardan en küçüğü Simpleks algoritmasına giren değişken olacaktır. min_sutun ise optimum vektöründe bu elemanın hangi satıra denk geldiğini tutan değişken*/*

*for i:1 thru jj do (if min_deger>optimumluk[1][i]
then(min_deger:optimumluk[1][i],min_sutun:i)),*

*/*Optimumluk vektöründe temel dışı değişkenler sıra ile tutulduğundan bu minimum sütuna karşılık gelen temel_disi_degiskenler_matrisi ndeki eleman probleme giren eleman olacaktır.*/*

*disp("Giren
değişken:",temel_disi_degiskenler_matrisi[min_sutun][1]),/*Simpleks
algoritmasında giren değişken*/*

*/*Pivot sütun min_sutun olarak belirlendi*/*

/-----*/*

*/*Yöntemde çıkan değişken $B^{(-1)} \cdot b / (B^{(-1)} \cdot P_i)$ vektöründeki minimum eleman olduğundan*

cikan_ust: bölmedeki üst kısım olan $B^{(-1)} \cdot b$

cikan_alt: bölmedeki alt kısım olan $(B^{(-1)} \cdot P_i)$ olup burada i giren değişkenin P matrisindeki sütunudur.

*temel_disi_degisken_yer temel dışı değişkenlerin P matrisindeki yerlerini
tuttuğundan bunun minimum sütuna karşılık gelen sütunu alınmıştır.*/*

cikan_ust:invB.b,

cikan_alt:invB.A_trp[temel_disi_degisken_yer[min_sutun][1]],

*min_satdeger:10^10,/*Başlangıçta min değerimizi sıfır yapıyoruz*/*

*min_satir:0,/*Başlangıçta pivot sütunumuzu sıfır yapıyoruz*/*

*/*Yöntemde çıkan değişken cikan_belirleme vektöründeki en küçük pozitif elemandır. min_satir ise bu eleman bu vektörde kaçınıcı elemene denk geldiğidir. Buelemanlar temel değişkenlerden oluştuğu için temel dışı değişken temel değişkenler içerisinde min_satir a karşılık gelen değişken olacaktır.*/*

*for i:1 thru k do (if cikan_alt[i][1]#0 then
bolme:cikan_ust[i][1]/cikan_alt[i][1],(if (bolme>0 and min_satdeger>bolme) then
(min_satdeger:bolme,min_satir:i))),*

Ek-7 (Devam). Simpleks yönteminin matrissel gösterimi

*/*Pivot satıra karşılık gelen temel_degiskenler_matrisi ndeki değişken sonraki aşamada temel dışı değişken olacaktır.*/*

disp("Çıkan değişken",temel_degiskenler_matrisi[min_satir][1]),

*/*B matrisindeki temel dışı kalan değişken yerine P matrisinde yeni temel değişkenin bulunduğu sütun yerleştirilmelidir.*/*

B_trp[min_satir]:A_trp[temel_disi_degisken_yer[min_sutun]][1],

*/*B deki benzer değişme C_B vektöründe yapılmalıdır.*/*

C_B[1][min_satir]:C[1][temel_disi_degisken_yer[min_sutun]][1],

*/*Yukarıda yapıldığı gibi Simpleks tablosunun yeni hali oluşturuluyor.*/*

C_B_temel_disi[1][min_sutun]:C[1][yer[min_satir]],

B_trp_temel_disi[min_sutun]:A_trp[yer[min_satir]],

B_temel_disi:transpose(B_trp_temel_disi),

B:transpose(B_trp),

invB:invert(B),

z_satir:C_B.invB.P-C,

cozum:C_B.invB.b,

alt_kisim:invB.P,

b_kisim:invB.b,

*/*Yeni temel değişken ve yeni temel dışı değişkenler temel_degiskenler_matrisi ve temel_disi_degiskenler_matrisi nde yerlerine yerleştirilmelidir. Benzer şekilde bunların P matrisindeki hangi sütunlara karşılık geldikleri temel_disi_degisken_yer ve temel değişkenlerin yerlerini tutan yer vektörlerinde yerlerine yazılmalıdır.*/*

yeni_temel_degisken:temel_disi_degiskenler_matrisi[min_sutun][1],

yeni_temel_disi_degisken:temel_degiskenler_matrisi[min_satir][1],

yeni_temel_degisken_yeri:temel_disi_degisken_yer[min_sutun][1],

yeni_temel_disi_degisken_yeri:yer[min_satir],

temel_degiskenler_matrisi[min_satir][1]:yeni_temel_degisken,

temel_disi_degiskenler_matrisi[min_sutun][1]:yeni_temel_disi_degisken,

temel_disi_degisken_yer[min_sutun][1]:yeni_temel_disi_degisken_yeri,

yer[min_satir]:yeni_temel_degisken_yeri,

Ek-7 (Devam). Simpleks yönteminin matrisel gösterimi

```
/*Yeni oluşan Simpleks matrisi ekrana yazdırılıyor.*/  
disp([asama, ".aşamada oluşan Simpleks matrisi:"]),  
ilk_satir:addcol(matrix([" "]),X,matrix(["çözüm"])),  
ikinci_satir:addcol(matrix(["z"]),z_satir,matrix([cozum])),  
alt_satirlar:addcol(temel_degiskenler_matrisi,alt_kisim,b_kisim),  
simp_matris_tum:addrow(ilk_satir,ikinci_satir,alt_satirlar),  
disp(simp_matris_tum),  
asama:asama+1,  
/*Yukarıda yapıldığı gibi problemin optimumluk koşulunu sağlayıp  
sağlamadığı kontrol ediliyor.*/  
optimumluk:C_B.invB.B_temel_disi-C_B_temel_disi,/*Problemdeki  
optimumluk koşulu bu vektörün içerisinde negatif eleman olmamasıdır. Ona  
bakılıyor*/  
sayac:0,  
for i:1 thru jj do (if optimumluk[1][i]>=0 then sayac:sayac+1),  
if sayac=jj then dongu:0  
));
```

EK-8. Uygunluk Aralığı belirlenmesi

i.Kısıtlamalardaki değişimler için matrissel forma eklenmesi gereken kısım

Bu bölümde matrissel algorithmada optimum çözüm bulunduktan sonra maxima ortamına aşağıdaki komutlar yazılır:

```
satir_say:length(b),
esitsizlik:makelist (random(6), satir_say),
load('fourier_elim),
Y:zeromatrix(satir_say,1),
for j:1 thru satir_say do ((for s:1 thru satir_say do (if j=s then
Y[s][1]:b[s][1]+d else Y[s][1]:b[s][1])),
carpim:factor(invB.Y),
for i:1 thru satir_say do esitsizlik[i]:carpim[i][1]>0,
sol:fourier_elim(esitsizlik,[d]),
disp([j, ".değişken için"]),
map(disp,sol),
Y:zeromatrix(satir_say,1))
```

ii.Amaç fonksiyonu katsayılarındaki değişimler için matrissel forma eklenmesi gereken kısım

```
esas_deg_satir_say:length(transpose(CI)),
C_B_satir_say:length(transpose(C_B)),
C_B_temel_disi_satir_say:length(transpose(C_B_temel_disi)),
Y:zeromatrix(1,C_B_satir_say),
YY:zeromatrix(1,C_B_temel_disi_satir_say),
for iii:1 thru esas_deg_satir_say do (if yer[iii]<=esas_deg_satir_say then (for
ik:1 thru C_B_satir_say do (if ik=iii then Y[1][ik]:C_B[1][ik]+d else
Y[1][ik]:C_B[1][ik]),
carpim:factor(Y.invB),
opti:factor(carpim.B_temel_disi-C_B_temel_disi),
esitsizlik:makelist (random(6), C_B_satir_say),
load('fourier_elim),
for il:1 thru C_B_satir_say do esitsizlik[il]:opti[1][il]>0,
sol:fourier_elim(esitsizlik,[d]),
```

```

disp(yer[iii]),
disp([yer[iii], ".değişken için"]),
map(dispsol),
Y:zeromatrix(1,C_B_satir_say))
else (Y:C_B,(for ikk:1 thru C_B_temel_disi_satir_say do (if
temel_disi_degisken_yer[ikk][1]<=esas_deg_satir_say then
(YY[1][ikk]:C_B_temel_disi[1][ikk]+d,disp([temel_disi_degisken_yer[ikk][1], ".deği
şken için"]))) else YY[1][ikk]:C_B_temel_disi[1][ikk])),
carpim:factor(Y.invB),
opti:factor(carpim.B_temel_disi-YY),
esitsizlik:makelist (random(6), C_B_satir_say),
load('fourier_elim),
for il:1 thru C_B_satir_say do esitsizlik[il]:opti[1][il]>0,
sol:fourier_elim(esitsizlik,[d]),
map(dispsol),
Y:zeromatrix(1,C_B_satir_say),YY:zeromatrix(1,C_B_temel_disi_satir_say)))

```