ÖNSÖZ

Optimizasyon Teorisinin mühendislik, üretim, işletme, ekonomi, haberleşme, ulaştırma, sanayi gibi pek çok alanda uygulanması, *YA*'nı vazgeçilmez kılmıştır. Özellikle bilgisayarların yaygın bir kullanım alanına sahip olmasından sonra endüstri kesimi de karar vermede yararlı bir araç olduğunu gördüğü Lineer Programlama (LP) konusuna ilgi duymaya başlamıştır. Petrol endüstrisi, problemlerinin karmaşıklığı sebebiyle, LP ile ciddi bir şekilde ilgilenen ilk endüstri branşı olmuştur [1].

Günümüzde, *YA* aşağıda sadece bir kaçını verebileceğimiz yüzlerce farklı problemlerin çözümünde kullanılmaktadır.Bunlar, Fabrika Organizasyonu, Atelye/Tezgah Optimizasyonu, Proje Yönetimi, kaynakların optimum kullanımı sayılabilir.

Bu ders notunu hazırlama amacımız,lisans seviyesinde eğitim veren fakültelerde Meslek Matematiği,Optimizasyon Teknikleri vb.isimler altında verilen derslere uygun olan ve lisansüstü eğitime önemli derecede katkı sağlayacak bir çalışma yapmak ve bunu daha da geliştirerek Optimizasyon Teknikleri kitabını yazmaktır.Bununla birlikte, güncel hayatın her alanında uygulamalarına rastladığımız optimizasyon kavramını öğrencilerimize uygulamaları ile aktararak bu konudaki bilincin oluşturulması ve en güncel yaklaşım olan yapay zeka tekniklerine zemin hazırlanması amaçlanmıştır.

İçeriğinde temelde Doğrusal Programlama ile Doğrusal Olmayan Programlama tekniklerini barındıran bu çalışmada öğrencilerimize klasik optimizasyon teorisinden ulaştırma problemlerine,gezgin satıcı probleminden en kısa yol problemine ve simpleks yöntemden atama problemine kadar çok sayıda konu örnekler ile desteklenerek ele alınmıştır. Bu çalışmanın gerek dersimizi alan öğrencilere gerekse bu konularla ilgilenen herkese faydalı olması temennisiyle…

Öneri ve eleştirilerini www.tektas@marmara.edu.tr adresine bekliyoruz.

Bu Cennet Vatan için Şehit Düşenlere İthafen…

Önsöz.....I
İçindekiler.....II
1. Giriş.....1
2. Lineer Programlama ve Grafik Çözümü.....2
2.1. Lineer Programlamaya Giriş.....2
2.2. Lineer Programlama Hakkında Genel Bilgi.....2
2.3 Lineer Programlama İşlem Basamakları.....3
2.4. Lineer Programlama Problem Örnekleri 3
3. Lineer Programlama ve Simpleks Metodu.....9
3.1. Simpleks Metoda Giriş.....9
3.2 Aylak Değişkenler ve Simpleks Metodun Örneklerle İncelenmesi 10
3.3 Simpleks Metot Maksimum Problemleri.....19
3.4 Simpleks Metot II (Minimum Problemleri).....30
3.5 Lineer Prramlama Problemlerinin marjinal analizleri ve formülleri:.....38
3.6 Lineer Programlama Problemlerinin Matris Fonksiyonlar.....44
3.7 Duality.....46
3.7.1 Duality ve Simpleks Metot İlişkisi.....46
3.7.2Duality'nin temel teoremi.....51
4. Ulaştırma Problemleri 60
4.1 Ulaştırma Problemlerine Giriş.....60
4.2.Örneklerle Ulaştırma Problemlerinin incelenmesi.....61
4.2.1.- Kuzey-batı köşesi yöntemi.....62
4.2.2. En küçük maliyetli hücreler metodu.....63
4.2.3. VAM(vogel) metodu.....63
5. Atama Problemleri ve Gezgin Satıcı Problemi.....64
5.1 Atama Problemlerine Giriş.....64
5.2 Atama Problemlerinin Çözüm adımları.....64
5.3. Örneklerle Atama Problemlerinin İncelenmesi.....65
6. Gezgin Satıcı Problemi.....67
6.1 Gezgin Satıcı Problemine Giriş.....67
6.2.Gezgin Satıcı Problemi İşlem Adımları.....67
6.3. Gezgin Satıcı Problemlerinin Örneklerle İncelenmesi.....68

7. Dinamik Programlama.....71
7.1.Dinamik Programlaya Giriş.....71
7.2.Dinamik Programlanın Örneklerle İnecelenmesi.....71
8. Uygulama Programları 74
8.1. Simpleks Metot 74
8.2. Atama Problemleri 75
8.3. Uygulamalarda Kullanılan Teknolojiler 76
8.3.1. Java 76
8.3.1.1. Java Hakkında Genel Bilgi.....76
8.3.1.2. Java Program Geliştirme Ortamaları ve Applett Kullanımı.....78
8.3.2. Delphi 79
8.3.2.1. Delphi Hakkında Genel Bilgi.....79
8.3.2.2. Atama Problemi Algoritma Yapısı 79
8.3.3. Active X 80
8.3.4. Html.....81
8.3.4.1.Html hakkında Genel Bilgi.....81
8.3.4.2. Html içerisinde Diğer Dillerin Kullanımı.....81
8.3.4.2.1 Html'de Active X Kullanımı.....81
9. Sonuç ve Öneriler.....82
10.Kaynaklar.....83
11.Özgeçmiş.....84

I.GİRİŞ

1.Optimizasyon

1.1. Tanım: En basit anlamı ile optimizasyon eldeki kısıtlı kaynakları en optimum biçimde kullanmak olarak tanımlanabilir(1).Matematiksel olarak ifade etmek gerekirse optimizasyon kısaca bir fonksiyonun minimize veya maksimize edilmesi olarak tanımlanabilir(2). Diğer bir değişle optimizasyon "en iyi amaç kriterinin en iyi değerini veren kısıtlardaki değişkenlerin değerini bulmaktır " (3).

Başka bir tanımlama ile "belirli amaçları gerçekleştirmek için en iyi kararları verme sanatı" veya "belirli koşullar altında herhangi bir şeyi en iyi yapma" (4) olarak da tanımlanan optimizasyon kısaca "en iyi sonuçları içeren işlemler topluluğudur" (5).Optimizasyonda bir amaç da maksimum kâr veya minimum maliyeti sağlayacak üretim miktarını kısıtlara bağlı olarak tespit etmektir. Günümüzün bilgisayar teknolojisi kadar güncel bir kavram olan optimizasyon kavramı çok çeşitli endüstri kesimlerinde uygulama olanağı bulmuştur.
Değişen teknolojilerin, sınırlı kaynakların, artan rekabetin, karmaşık hale gelen sistemlerin doğurduğu problemlerin klasik yöntemlerle (matematiksel veya matematiksel olmayan, analitik veya sayısal) çözümünün güçleşmesi optimizasyon kavramını güncelleştiren en önemli sebeptir.Bu yönüyle optimizasyonun kullanılmadığı bir bilim dalı hemen hemen yok gibidir (6).

1.2. Tarihçe: Gerçek hayatta karşılaşılan birçok problem için geliştirilen karar modellerinin kısıtları ve amaç fonksiyonlarında her zaman doğrusal bir ilişki kurulamadığından 1950'li yıllardan sonra geliştirilmeye başlayan ve temelleri 18. ve 19. yüzyıllara dayanan yeni analitik ve sayısal yöntemler 1960'lı yıllardan sonra sayısal bilgisayarlarında desteği ile hızla çoğalmıştır.

Özellikle kimyasal işlemlerin süreklilik arz etmesi, planlamacıların, tasarımcıların, mühendislerin, jeologların, ekonomistlerin, iktisatçıların, işletmecilerin v.b. kendi alanlarındaki problemleri çözmek için yaptıkları çalışmalar optimizasyon ve buna bağlı teknikleri hızla ortaya çıkarmıştır. Benzer şekilde bu tekniklerin amaçlandığı alanlara, sistemin özelliklerine, kullanılan matematiksel yöntemlere ve kıstasların tasnifleri aşamalar geçirmiştir(3).

Klasik optimizasyon teorisi Cauchy, Lagrange ve Newton tarafından geliştirilmiştir. Newton ve Leibnitz' in analiz çalışmaları optimizasyonun diferansiyel hesap metodlarının geliştirilmesine katkıda bulunmuştur. Kısıtlı problemler için optimizasyon metodunu adıyla anılan Lagrange geliştirmiştir. Kısıtsız optimizasyon problemlerini çözmek için Steepest Descent (en dik iniş,eğim) metodunun ilk uygulaması da Cauchy tarafından yapılmıştır.

Optimizasyon konusundaki bu çalışmalar 20. yüzyılın ortalarına kadar çok yavaş ilerlemiştir. 1950' lerden sonra sayısal bilgisayarların icadı optimizasyonda çok büyük çalışmaları beraberinde getirerek birçok yeni teori ve metodun ortaya çıkmasını sağlamıştır. Fakat 1960' lı yıllarda kısıtsız optimizasyon konusundaki sayısal metodlar sadece İngiltere' de geliştirilmiştir (5).

Simpleks metodunu 1947' de Dantzing, Dinamik Programlama Tekniğini 1954' de Bellmann geliştirmiştir.Bu çalışmamızın esasını teşkil eden Doğrusal Olmayan Programlama konusundaki ilk önemli çalışmalar 1951 yılında Karush – Kuhn ve Tucker tarafından optimal çözüm için gerek ve yeter şartlar teorisi başlığı adı altında

sunulmuştur(7). 1960' lı yıllarda Zoutendijk ve Rosen' de Doğrusal Olmayan Programlama sahasında önemli çalışmalar yapmışlardır.

Doğrusal Olmayan Programlama alanındaki en büyük gelişme kısıtsız optimizasyonun bilinen tekniklerini kullanarak çok zor problemlerin çözümünü kolaylaştıran ciddi çalışmaların Carroll, Fiacco ve Mc Cormick tarafından ortaya konmasıdır. Geometrik Programlama ise 1960' lı yıllarda Peterson, Zener ve Duffin tarafından geliştirilmiştir(5).Düzlemsel Kesme Algoritması ise 1969'da Zangwill tarafından ortaya konmuştur. İndirgenmiş Gradient Metod ise Wolfe tarafından 1963' de geliştirilmiştir(8).

1.3.Optimizasyon Probleminin Özellikleri ve Çözüm Aşamaları

Bir optimizasyon probleminin temel özelliği üç kategoriye ayrılmasıdır. Bunlar :

En az bir amaç fonksiyonunun optimize edilmesi
Eşitlik kısıtları
Eşitsizlik kısıtlarıdır

Yani genel bir optimizasyon problemi:

maksimum (minimum) f(x)

$g_i(x) \leq 0, (\geq 0)$ i = 1, 2, …….., m

veya

$h_i(x) = 0$ i = m + 1, m + 2, ……, n

şeklindedir.Bu genel tanım altında amaç fonksiyonunun en iyi değerini veren

$$X = (x_1, x_2, ..., x_n)^T$$

n – boyutlu çözüm vektörüne model vektörü de denir(3).

(1) ile ifade edilen genel problemde f(x) amaç fonksiyonunu, $g_i(x)$ eşitsizlik kısıtları ve $h_i(x)$ eşitlik kısıtları temsil eder. n'in sıfır olması problemin kısıtsız olması, sıfırdan farklı olması da problemin kısıtlı olması anlamına gelir.

Genel bir optimizasyon probleminin çözümü altı adımda gerçekleştirilir.

i. İşlem analiz edilerek işlem değişkenlerinin bütün bir listesi çıkarılır.

ii. Optimizasyon için amaç fonksiyonunu tanımlayacak kriter belirlenir.

iii. Matematiksel ifadelerle kullanılabilir bir işlem gerçekleştirilir.

iv. Problem çok büyükse;

a) Kontrol edilebilir ve modeli basitleştirilir.

b) Amaç fonksiyonu tekniği matematiksel ifadeye uygulanır.

v. Uygun optimizasyon tekniği matematiksel ifadeye uygulanır.

vi. Cevaplar kontrol edilir(3).

Bütün optimizasyon problemlerinin çözümü için etkili tek bir metot olmadığından optimizasyon metotları optimizasyon problemlerinin farklı tiplerinin çözümü için geliştirilmiştir(5).

1.4. Doğrusal Olmayan Programlama

Gerçek hayatta karşılaşılan birçok problem için geliştirilen karar modellerinin kısıtlarından en az biri veya amaç fonksiyonunun doğrusal olmadığı durumlar için geliştirilen tüm kavram ve teknikler "Doğrusal Olmayan Programlama" adı altında incelenmektedir(6).

Doğrusal Olmayan Programlama:

$Z = f(x_i) = f(x_1, ..., x_n)$ (i = 1, 2, ……, n)

şeklinde tanımlanan sürekli ve türevlenebilen bir amaç fonksiyonunun;

$g_j(x_i) \leq 0$ $(x_i \geq 0)$ (i = 1, 2, ……, n)(j = 1, 2, ……, m)

kısıtları altında optimum çözümünü araştırma yöntemidir(9). Doğrusal ve doğrusal olmayan denklemlerden oluşan $g_j(x_i)$ kısıtları eşitlikler veya eşitsizlikler şeklinde verilebilir.

Şöyle ki;

$g_j(x_i) \leq 0$ (≥ 0) (j = 1, 2, ……., l) ve

$g_j(x_i) = 0$ (j = 1 + 1, ……..., m)

şeklinde tanımlanan kısıtlar m tane denklemden oluşan bir denklem sistemidir. Bu denklemlerin 1 tanesi eşitsizlik, (m-1) tanesi eşitlik denkleminden oluşur(10).

1.5. Amaç Fonksiyonunun Yorumlanması

Amaç fonksiyonunun yorumlanması konusunda kısıtlarda ve (veya) kısıtsız problemde yer alan değişkenlerin (karar değişkenleri) "en iyi seçmedeki kriter" programlamada amaç fonksiyonu olarak adlandırılır.Pratikte ise kısıtlarda ve amaç fonksiyonunda yer alacak değişkenlerin (kıt kaynakların) en iyi değerlerini bulmak olarak tanımlanabilir(3).

1.6. Optimizasyon ile İlgili Temel Kavram ve Tanımlar

1.6.1. Fonksiyonlarda Süreklilik Kavramı

A. Tek Değişkenli Fonksiyonlarda Süreklilik

Tek değişkenli bir y = f(x) fonksiyonunun bir x_0 noktasında sürekli olması demek, verilen $\varepsilon > 0$ sayısına karşılık öyle bir $|h| < \sigma, \sigma > 0$ sayısının bulunmasıdır ki;

$$\left| f(x_0 + h) - f(x_0) \right| \leq \varepsilon$$ dır.

B. Çok Değişkenli Fonksiyonlarda Süreklilik

Çok değişkenli bir $z = f(x) = f(x_1, x_2, ..., x_n)$ fonksiyonunun bir x_0 noktasında sürekli olması demek, verilen $\varepsilon > 0$ sayısına karşılık öyle bir $|h| < \sigma, \sigma > 0$ sayısının bulunmasıdır ki;

$$\left| f(x_0 + h) - f(x_0) \right| \leq \varepsilon \quad \text{dır.}$$

Burada;

$$h = (h_1, h_2, ..., h_n) \quad \text{ve} \quad \sigma = (\sigma_1, \sigma_2, ..., \sigma_n) > 0 \quad \text{dır(11).}$$

1.6.2. Unimodal (Tek değer) Fonksiyon

Verilen bir aralıkta bir tek maksimum veya minimuma sahip fonksiyona Unimodal fonksiyon denir(5). Matematiksel olarak ifade etmek gerekirse: [a,b] aralığı üzerinde bir y=f(x) fonksiyonu tanımlanmış olsun.

$p \in [a, b]$ sayısı için;

i) f(x), [a, p] aralığında azalan bir fonksiyon

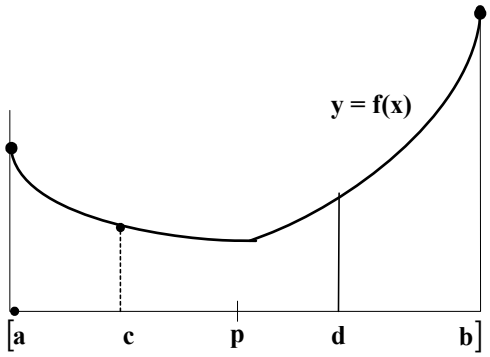
ii) f(x), [p, b] aralığında artan bir fonksiyon

ise y = f(x) fonksiyonuna bu aralıkta Unimodal (tek değerli) fonksiyon denir(12).

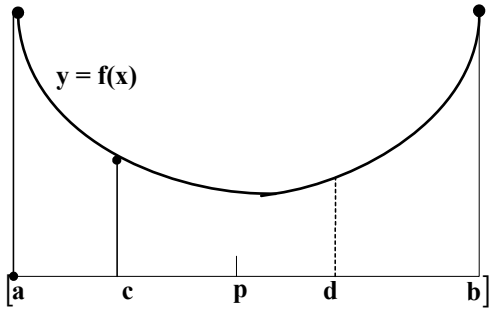
Eğer f(x) fonksiyonu [a, b] aralığında Unimodal fonksiyon ise f(x) minimum değerini $a < c < d < b$ şeklindeki bir [c,d] aralığında alabilir. Bu minimum değer f(c) ve f(d)' nin max[f(a), f(b)]' den daha küçük iken garanti edilir (Şekil 1.a - b).

Eğer $f(c) \leq f(d)$ ise [a, d] aralığının sağından aralık daraltılır (Şekil 1.a).

Eğer $f(d) < f(c)$ ise [c, b] aralığının sağından itibaren aralık daraltılır (Şekil 1.b).



Şekil 1.a



Şekil 1.b

1.7.Konvekslik ile İlgili Tanımlar

1.7.1.Konveks Bileşen

S, E_n, n – boyutlu öklidyen uzayda boş olmayan bir küme olsun.

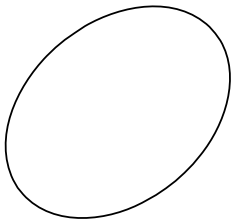
$x_i \in S \text{ ve } \alpha_i \geq 0, \sum \lambda_i = 1$ iken, $x_0 = \alpha_1 x_1 + \alpha_2 x_2 + ... + \alpha_n x_n$ olsun.

Eğer; $x_0 = \sum_{i=1}^{n} \alpha_i x_i$ şeklinde elde edilen x_0 noktasına $x_1, x_2, x_3, ..., x_n$ noktalarının konveks (dışbükey) bileşeni denir(6).

1.7.2. Konveks Küme

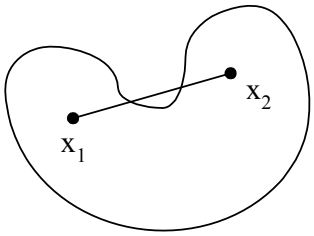
S, E_n, n-boyutlu öklidyen uzayda boş olmayan bir küme olsun. S kümesinin farklı iki noktasının konveks (dışbükey) bileşeni ile bulunan nokta yine S kümesinin bir elemanı ise S kümesine konveks küme denir(8).(Şekil 2.a-b)

Konveks Küme



Şekil 2.a

Konveks değil



Şekil 2.b

Matematiksel olarak ifade etmek gerekirse;

$$x_i, x_j \in S, \ i \neq j, \ 0 \leq \alpha \leq 1 \quad \text{iken} \quad x_0 = \alpha x_i + (1-\lambda) x_j \in S(\forall i \neq j)$$

oluyorsa S kümesine konveks (dışbükey) küme denir.

1.7.3.Konveks Fonksiyon

E_n, n-boyutlu öklidyen uzayda verilen herhangi iki nokta (x_1, x_2) olsun.Eğer aşağıdaki eşitsizlik E_n, n-boyutlu öklidyen uzayındaki her nokta çifti için geçerli ise f fonksiyonuna konvekstir denir(13).

$$(x_1, x_2) \in E_2 ; 0 \leq \alpha \leq 1 \text{ için; } f[(1-\alpha)x_1 + \alpha x_2] \leq [(1-\alpha) f(x_1) + \alpha f(x_2)]$$

1.7.4.Konkav (İçbükey) (Konveks Olmayan) Fonksiyon

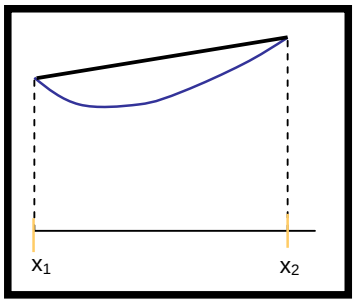
Konveks fonksiyonunun tanımına benzer olarak;

$$(x_1, x_2) \in E_2 ; 0 \le \alpha \le 1 \text{ için; } f[(1-\alpha)x_1 + \alpha x_2] \ge [(1-\alpha) f(x_1) + \alpha f(x_2)]$$

oluyorsa f fonksiyonuna konkav (içbükey) fonksiyondur denir.(1.7.3) ve (1.7.4) ile ifade edilen tanımları geometrik olarak açıklamak gerekirse;

Fonksiyonun yüzeyi üzerinde alınan herhangi iki noktayı birleştiren doğru, fonksiyonun temsil ettiği eğrinin altında kalıyorsa fonksiyona konkav fonksiyon, aksi halde yani yüzey üzerindeki iki noktayı birleştiren doğru fonksiyonun temsil ettiği eğrinin üstünde kalıyorsa fonksiyona konvekstir denir(8) (Şekil 3.a-b-c).

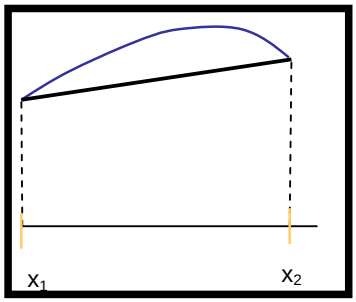
Konveks Fonksiyon



(Şekil 3.a).

$[(1-\alpha)x_1 + \alpha x_2]$

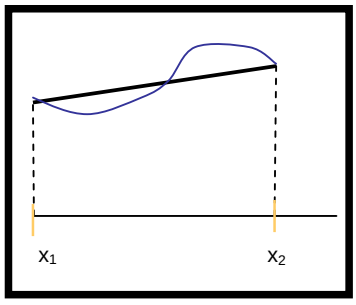
Konkav Fonksiyon



(Şekil 3.b).

$[(1-\alpha)x_1 + \alpha x_2]$

Ne Konveks – Ne Konkav



(Şekil 3.c).

$[(1-\alpha)x_1 + \alpha x_2]$

1.8. Optimum Aramada Konveksliğin ve Konkavlığın Etkileri

1.8.1. Kısıtsız Maksimum (Minimum)

Eğer bir doğrusal olmayan programlama problemi bir f(x) amaç fonksiyonunu içerirse ve ayrıca f(x) konveks (konkav) ise uygun bölge içindeki bir noktada bir tek optimum çözüm vardır ve bu noktada 1. mertebeden türevlerin hepsi sıfırdır. Aynı zamanda bu nokta bir sınır noktada olabilir. Aynı özellik bu sınır nokta içinde geçerlidir(13).

1.8.2.Kısıtlı Maksimum

Eğer bir doğrusal olmayan programlama problemi aynı anda hem bir amaç fonksiyonu hem de kısıtların bir kümesinden oluşuyorsa optimum çözümün tekliği amaç fonksiyonu ve kısıtlara bağlıdır.Eğer amaç fonksiyonu konkav ve kısıtların kümesi konveks ise problemin bir tek maksimum çözümü vardır. Bu nedenle herhangi bir sabit nokta mutlak maksimum çözüm olmak zorundadır.

1.8.3.Kısıtlı Minimum

Eğer bir doğrusal olmayan programlama problemi aynı anda bir amaç fonksiyonu ve kısıtların bir kümesini içeriyorsa optimum çözümün tekliği amaç fonksiyonu ve kısıtlara bağlıdır.
Eğer amaç fonksiyonu konveks ve keza kısıtların kümesi de konveks bölge formunda ise problemin bir tek minimum çözümü olacaktır. Bu nedenle herhangi bir sabit nokta mutlak minimum çözüm olmak zorundadır.

1.8.4.Konkav (Konveks) Fonksiyonun Minimizasyonu (Maksimizasyonu)

Eğer bir konveks fonksiyon maksimize (konkav fonksiyon minimize) edilirse optimal çözüm kısıtlar kümesinin ekstremum noktalarının yalnız birisinde bulunacaktır.

1.9. Bir Fonksiyonun Gradienti

$f(x) = f(x_1, x_2, ..., x_n)$ n-değişkenli fonksiyonunu göz önüne alalım.

Burada, $(x_1, x_2, ..., x_n)$ koordinatları n-boyutlu öklidyen uzayda X-sütun vektörü ile temsil edilirler(1).

$f(x) = f(x_1, x_2, ..., x_n)$ fonksiyonunun gradienti ise $\nabla f(x)$ veya grad f(x) sembolleri ile gösterilir ve;

$$\text{grad } f(x) = \nabla f(x) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_n} \right)$$

veya kısaca;

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_k} \right)$$ dır. (k =1, 2 , … , n) şeklinde tanımlanır.

1.10. Hessian Matrisi

f(x) fonksiyonunun ikinci mertebeden sürekli türevlere sahip olması durumunda bütün

i ve j' ler için;

$$\frac{\partial^2 f}{\partial x_i . \partial x_j} = \frac{\partial^2 f}{\partial x_i . \partial x_j} \text{ eşitliği geçerlidir(14).}$$

Bu nedenle f(x) = $f(x_1, x_2, ..., x_n)$ n-değişkenli fonksiyonunun ikinci mertebeden kısmi türevleri;

$$H_f = \left[\frac{\partial^2 f}{\partial x_i . \partial x_j} \right]_{n \times n}$$

şeklinde bir matris ile gösterilebilir. İşte bu n x n' lik $H_f(x)$ matrisine f(x) fonksiyonunun Hessian matrisi denir. Bu matris aynı zamanda simetriktir(15). Açıkça yazmak gerekirse;

$$H_f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}_{n \times n}$$

II. KLASİK OPTİMİZASYON TEORİSİ

2. Klasik Optimizasyon Teorisi

Klasik Optimizasyon Teorisi kısıtlı ve kısıtsız fonksiyonlar için ekstremum noktaların belirlenmesinde diferansiyel hesabın kullanılmasını geliştirmiştir.Geliştirilen bu metodlar sayısal hesaplamalar için uygun olmayabilir.

Bu temel başlık altında kısıtsız ekstremumların belirlenmesi için gerek ve yeter şartları, eşitlik kısıtlara sahip problemler için Karush – Kuhn – Tucker gerek ve yeter şartlarını inceleyip örnekler vererek açıklayacağız(11).

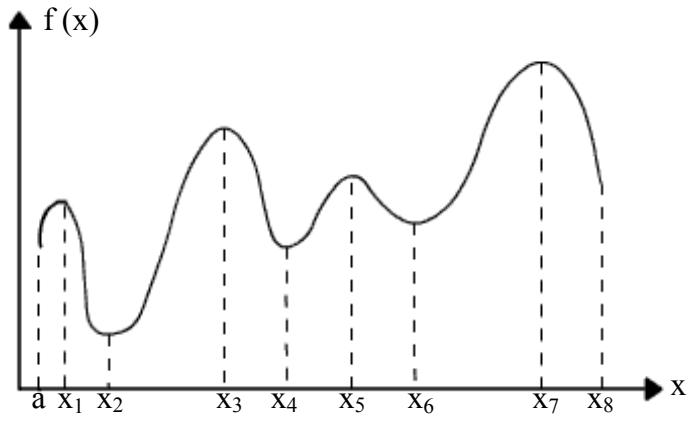
2.1 Kısıtsız Ekstremum Problemleri

f(x) fonksiyonunun bir ekstremum noktası maksimum veya minimum nokta olarak tanımlanır. Matematiksel olarak tanımlamak gerekirse;

$h = (h_1, h_2, ..., h_j, ..., h_n)$ öyleki $|h_j|$ bütün j' ler için yeterince küçük olmak üzere;

$|f(x_0 + h) - f(x_0)| \leq \varepsilon$ şartı sağlanıyorsa x_0 noktası bir maksimum noktadır(13).

Bir başka deyişle; x_0'ın komşuluğundaki her noktada f fonksiyonunun değeri $f(x_0)$'dan küçük ya da eşit kalırsa x_0'a f fonksiyonunun maksimum noktası denir. Benzer şekilde; $h = (h_1, h_2, ..., h_j, ..., h_n)$ öyleki $|h_j|$ bütün j'ler için yeterince küçük olmak üzere; $f(x_0 + h) \geq f(x_0)$ şartı sağlanıyorsa x_0 noktası bir minimum noktadır. Yani x_0'ın komşuluğundaki her noktada f 'nin aldığı değer $f(x_0)$ değerinden büyük yada eşit kalırsa x_0 noktasına f fonksiyonunun minimum noktası denir. Aşağıdaki şekil [a, b] aralığında tek değişkenli bir f(x) fonksiyonunun maksimum ve minimumlarını tanımlar (Şekil 4).



Şekil.4. f(x) fonksiyonunun maksimum ve minimumları

Şeklimize göre x_1, x_2, x_3, x_4, x_6 noktaları f(x) fonksiyonunun ekstremum noktalarıdır. Bu noktalardan x_1, x_3 ve x_6 noktaları maksimum noktalar iken x_2 ve x_4 noktaları da minimum noktalarıdır. $f(x_6) = \max\{f(x_1), f(x_3), f(x_6)\}$ olduğundan $f(x_6)$ global maksimum veya mutlak maksimum olarak isimlendirilir. $f(x_6)$' ya göre $f(x_1)$ ve $f(x_3)$ noktaları da yerel maksimum olarak adlandırılır.

Benzer olarak;

$f(x_2) = \min \{f(x_2), f(x_4)\}$ olduğundan $f(x_2)$ noktası mutlak minimum nokta olarak isimlendirilirken, $f(x_2)$'ye göre $f(x_4)$ noktası yerel minimum nokta olarak isimlendirilir.

x_1 ile x_3 noktaları karşılaştırıldığında x_1 zayıf maksimum iken x_3 güçlü maksimumdur. x_2 ile x_4 noktaları karşılaştırıldığında x_2 noktası güçlü minimum nokta iken x_4 noktası x_2 noktasına göre zayıf minimum noktadır(11).

Genelleştirecek olursak;

$f(x_0 + h) \leq f(x_0)$ ise x_0 bir zayıf maksimum

$f(x_0 + h) < f(x_0)$ ise x_0 bir güçlü maksimum

$f(x_0 + h) \geq f(x_0)$ ise x_0 bir zayıf minimum

$f(x_0 + h) > f(x_0)$ ise x_0 bir güçlü minimum noktadır.

Burada h daha önce tanımlandığı gibidir. Şekil 4'den de görüldüğü gibi bütün ekstremum noktalarda f(x) fonksiyonunun eğiminin (1. türevi) sıfıra eşit olduğu sonucuna varabiliriz. Buna karşılık bu özellik tek değildir. Yani tersi doğru olmayabilir. f(x) fonksiyonunun eğimi herhangi bir noktada sıfır olduğu halde bu nokta ekstremum nokta olmayabilir. Şekil 4'deki x_5 noktası böyle bir noktadır. Yani bu noktada f(x) fonksiyonunun eğimi sıfır olduğu halde x_5 noktası bir ekstremum nokta değildir.

İşte böyle noktalara, gradienti (eğim) sıfır oldugğu halde ekstremum olmayan noktalara büküm noktaları denir.

2.2. Ekstremum İçin Gerek ve Yeter Şartlar

n-değişkenli bir f(x) fonksiyonunu gözönüne alalım. f(x) fonksiyonunun her x noktasında birinci ve ikinci mertebeden sürekli türevlere sahip olduğunu varsayalım.

Teorem-1: Herhangi bir x_0 noktasının f(x) fonksiyonunun ekstremum noktası olması için gerek şart; $\nabla f(x_0) = 0$ olmasıdır.

Ispat: $0 < \theta < 1$ için Taylor teoreminden;

…… $f(x_0 + h) - f(x_0) = \nabla f(x_0)h + \left(\frac{1}{2}\right) h^T H h \Big|_{x_0 + \theta h}$

Yeterince küçük $|h_j|$ 'ler için kalan terim $\left(\frac{1}{2}\right) h^T H h$, $h_j{}^2$' nin mertebesindedir.

Bundan dolayı (1) deki açılım;

$$f(x_0 + h) - f(x_0) \cong \nabla f(x_0)h + 0.(h_j{}^2) \cong \nabla f(x_0) . h$$

Şimdi x_0 noktasının bir minimum nokta olduğunu varsayalım. Olmayana ergi yöntemiyle gösterilebilir ki $\nabla f(x_o)$ sıfıra eşit olmak zorundadır. x_0 bir minimum nokta değil iken özel bir j için;

$$\frac{\partial f(x_0)}{\partial x_j} < 0 \text{ veya } \frac{\partial f(x_0)}{\partial x_j} > 0 \text{ olabilir.}$$

h_j 'nin işareti uygun seçilerek $h_j . \frac{\partial f(x_0)}{\partial x_j} < 0$ her zaman elde edilebilir.

Diğer h_j 'lerin kümesi sıfıra eşitlenerek Taylor açılımındaki kabulden $f(x_0 + h) - f(x_0) < 0$ veya $f(x_0 + h) < f(x_0)$ bulunur. Bu ise x_0 noktasının bir minimum nokta olması ile çelişir. O halde $\nabla f(x_o) = 0$ olmak zorundadır.

Bu sonuç gerektir fakat yeter değildir. Yani $\nabla f(x_o) = 0$ iken x_0 bir ekstremum nokta olmayabilir.

Teorem-2: Sabit bir x_0 noktasının bir ekstremum nokta olması için yeter şart Hessian Matrisinin $(H_f) x_0$' daki değeri ile belirlenir.

$H(f)|_{x_0} > 0 \Rightarrow x_0$ bir minimum noktadır.

$H(f)|_{x_0} < 0 \Rightarrow x_0$ bir maksimum noktadır.

İspat: $0 < \theta < 1$ için Taylor teoreminden;

$$f(x_0 + h) - f(x_0) = \nabla f(x_0)h + \left(\frac{1}{2}\right) h^T H h \Big|_{x_0 + \theta h} \text{ idi.}$$

x_0 bir sabit nokta iken Teorem-1 gereğince $\nabla f(x_0) = 0$'dır.Buna göre;

$$f(x_0 + h) - f(x_0) = \left(\frac{1}{2}\right) h^T H h \Big|_{x_0 + \theta h} \text{ olur.}$$

x_0 noktasını minimum nokta olarak alalım.

Tanımdan; $f(x_0 + h) > f(x_0)$ dır. Bunun anlamı şudur;

x_0 noktasının bir minimum nokta olması için; $\left(\frac{1}{2}\right) h^T H h\Big|_{x_0 + \theta h} > 0$ olmalıdır.

İkinci kısmi türevlerin sürekli olmasını kabulü ile $\left(\frac{1}{2}\right) h^T H$ x_0 ve $x_0 + \theta h$ 'ın her ikisinde de değerlendirildiğinde aynı işarete sahip olmak zorundadır. $h^T H h\Big|_{x_0}$ bir karesel form olarak tanımlanır ve x_0 noktasında değerlendirilirse x_0 'ın minimum nokta olması için $H\Big|_{x_0}$ pozitif tanımlı olmalıdır.

Bu son ifadenin anlamı şudur:Sabit bir x_0 noktasının minimum nokta olması için yeter şart Hessian Matrisinin bu noktada **pozitif tanımlı olmasıdır.** Aynı yeter şart x_0 'ın maksimum nokta olması için yapıldığında Hessian matrisinin x_0 noktasında **negatif tanımlı** olması gerektiği söylenebilir.

Sonuç 1: Eğer $H\Big|_{x_0}$ tanımsız ise x_0 bir büküm noktası olmak zorundadır.

Sonuç 2: Teorem-1 ve Teorem-2 ile sunulan ifadeler tek değişkenli y = f(x) fonksiyonu için şu şekilde özetlenebilir.Herhangi bir x_0 noktasının y = f(x) fonksiyonunun bir ekstremum noktası olması için gerek şart **f '(x_0) = 0** olmasıdır. Yeter şart;

f ''(x_0) < 0 ise x_0 bir maksimum noktadır.

f ''(x_0) > 0 ise x_0 bir minimum noktadır(16).

Eğer, **f ''(x_0) = 0** ise x_0 'ın ekstremum nokta olması için yüksek mertebeden türevler gözönüne alınmak zorundadır. Bunu aşağıdaki sonuç teorem ile sunabiliriz.

Teorem-3: y=f(x) fonksiyonu verilsin. f(x)'in sabit bir x_0 noktasında;

$$f^{(n-1)}(x_0) = 0 \text{ ve } f^{(n)}(x_0) \neq 0 \text{ oluyorsa } x = x_0 \text{ noktasında f(x);}$$

n tek ise bir büküm noktasına sahiptir.**n çift ise;**

a) $f^{(n)}(x_0) < 0$ ise maksimuma sahiptir.

b) $f^{(n)}(x_0) > 0$ ise minimuma sahiptir(11).

2.3. Çok değişkenli Fonksiyonlarının Optimizasyonu (Kısıtsız Optimizasyon)

Daha önce tanımlandığı gibi,$y=f(x_1,x_2,...x_n)$ n- değişkenli fonksiyonu 2. mertebeden sürekli kısmi türevlere sahipken bu fonksiyonun **hessian matrisi** simetrik bir matris (simetrik matris: $\forall i \neq j$ olmak üzere aij=aji olan matristir) olup aşağıdaki gibidir ;

$$Hf(x)= \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} = \begin{bmatrix} f_{11} & f_{12} & f_{1n} \\ f_{21} & f_{22} & f_{2n} \\ . & . & . \\ . & . & . \\ . & . & . \\ f_{n1} & f_{n2} & f_{nn} \end{bmatrix}$$

nxn nxn

$y=f(x_1,x_2,...x_n)$ fonksiyonların ekstremumlara sahip olması için;

i) Gerek şart:

$\forall(x_1,x_2,..,x_n) = gradf(x_1,x_2,..,x_n) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ... \frac{\partial f}{\partial x_n}\right) = 0$ olmasıdır. Bu denklemin çözümü olan noktalara **sabit noktalar** denir,

ii) Yeter şart: x_0 noktası $\nabla f(x) = 0$ şartını sağlayan nokta(**sabit nokta)** olsun.Buna göre;

1.Test :

i. **$Hf(x_0) > 0$ (Pozitif tanımlı) ise minimum noktasıdır**
ii. **$Hf(x_0) < 0$ (Negatif tanımlı)ise maximum noktasıdır**
iii. **$Hf(x_0)$ tanımsız ise x_0 büküm noktasıdır**

2.Test:

$det(A-\lambda I)=0$ n.dereceden bir polinom denklem olup buna f fonksiyonunun **karakteristik polinomu** ,bu denklemin köklerine de **karakteristikler** veya **özdeğerler** denir.Buna göre ;

i. $\forall$i için $\lambda_i > 0$ **ise A pozitif tanımlıdır ($A=(a_{ij})_{nxn}$)**
ii. $\forall$i için $\lambda_i < 0$ **ise A negatif tanımlıdır ($A=(a_{ij})_{nxn}$)**
iii. **Diğer durumlarda tanımsızdır.**

2.3.1. A matrisinin tanımlılığı:nxn'lik bir A matrisinin tanımlılığını belirlemek için aşağıdaki test uygulanır.

A= $\begin{bmatrix} a_{11} & a_{12} & a_{1n} \\ a_{21} & a_{22} & a_{2n} \\ . & . & . \\ . & . & . \\ a_{n1} & a_{n2} & a_{nn} \end{bmatrix}$ olsun. A'nın uzanan alt matrisleri ;

nxn

$A_1=[a_{11}]$; $A_2=\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$,..., $Aj=\begin{bmatrix} a_{11} & a_{12} & a_{1j} \\ a_{21} & a_{22} & a_{2j} \\ a_{j1} & a_{j2} & a_{jj} \end{bmatrix}$,...$An\begin{bmatrix} a_{11} & . & a_{1n} \\ a_{21} & . & a_{2n} \\ a_{n1} & . & a_{nn} \end{bmatrix}$ olarak tanımlanır.

2x2 jxj nxn

Bu matrislerin determinantlarının hepsi pozitif ise A matrisi pozitif tanımlıdır .Yani;

i. $\forall i$ **için det A_i>0 ise A pozitif tanımlıdır.**

ii. $\forall i$ **için $(-1)^i$det A_i>0 ise A negatif tanımlıdır.**

iii. **Diğer durumlarda tanımsızdır .Nokta büküm noktasıdır.**

2.3.2. Konveks,Konkav Fonksiyonunun Hessian Matrisi ile Tayini (D'escartes kuralı)

det $(A-\lambda I)=0$ ifadesi $P(\lambda)=0$ şeklinde λ'ya bağlı bir polinom olup;

p(λ)'daki işaret değişikliği sayısı δ (**pozitif kök sayısı**)

p(-λ)'daki işaret değişikliği sayısı δ' (**negatif kök sayısı**)

ile tanımlanır ve sabit noktanın kimliği ilefonksiyonun konkav yada konveksliği kolayca belirlenir.

Örnek : $f(x_1,x_2,x_3)=x_1+2x_3+x_2x_3-x_1^2-x_2^2-x_3^2$ fonksiyonun ekstremumlarını ve konveks yada konkavlığını inceleyiniz

1)Gerek şart: $\nabla f(x)=0$ olmalıdır

$\frac{\partial f}{\partial x_1}=1-2x_1=0$, $\frac{\partial f}{\partial x_2}=x_3-2x_2=0$, $\frac{\partial f}{\partial x_3}=2+x_2-2x_3=0$

Buradan; $x_1=1/2$, $x_2=2/3$, $x_3=4/3$ olduğu görülür . $x_0=(1/2,2/3,4/3)$ sabit noktadır.

2)Yeter şart:Bu fonksiyona ait Hessian matrisini oluşturalım

$Hf(x_0)=\begin{bmatrix} f_{x1\,fx1} & f_{x1x2} & f_{x1x3} \\ f_{x2x1} & f_{x2x2} & f_{x2x3} \\ f_{x3x1} & f_{x3x2} & f_{x3x3} \end{bmatrix}$ ise $Hf(x_0)=\begin{bmatrix} -2 & 0 & 0 \\ 0 & -2 & 1 \\ 0 & 1 & -2 \end{bmatrix}$

H_1=[-2] olduğundan det H_1=-2

$H_2=\begin{bmatrix} -2 & 0 \\ 0 & -2 \end{bmatrix}$ olduğundan det H_2=4

$H_3\begin{bmatrix} -2 & 0 & 0 \\ 0 & -2 & 1 \\ 0 & 1 & -2 \end{bmatrix}$ olduğundan det H_3=-6

$H_f(x)$ negatif tanımlıdır x_0=(1/2,2/3,4/3) maksimum noktadır

Not:yeter şart için II. Metot şu şekildedir.

Det $(H\text{-}I\lambda)=\begin{vmatrix} -2 & 0 & 0 \\ 0 & -2 & 1 \\ 0 & 1 & -2 \end{vmatrix} - \lambda \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} = \begin{vmatrix} -2-\lambda & 0 & 0 \\ 0 & -2-\lambda & 1 \\ 0 & 1 & -2-\lambda \end{vmatrix} = 0$ ve buradan;

$p(\lambda)=(-2-\lambda)[(-2-\lambda)^2-1]=0$ ise $p(\lambda) = -\lambda^3 - 6\lambda^2 - 11\lambda - 6 = 0$ olur.

$p(-\lambda) = \lambda^3 - 6\lambda^2 + 11\lambda - 6 = 0$'dır (3 tane negatif kök vardır , fonksiyon konkavdır yani x_0 noktasımaksimum noktasıdır)

Örnek : $f(x,y)=x^2-4xy+y^2$ fonksiyonunun ekstremumlarını ve konveks yada konkavlığını inceleyin.

Çözüm:

1)Gerek şart: $\nabla f(x) = 0$ olmalıdır

$\frac{\partial f}{\partial x}$=2x-4y=0 , $\frac{\partial f}{\partial y}$=-4x+2y=0

Buradan; x=0, y=0 olduğu görülür. x_0=(0,0) sabit noktadır.

2)Yeter şart:$Hf(x_0)=\begin{bmatrix} fxx & fxy \\ fyx & fyy \end{bmatrix}=\begin{bmatrix} 2 & -4 \\ -4 & 2 \end{bmatrix}$

H_1=[2] ise det H_1=2>0

$H_2=\begin{bmatrix} 2 & -4 \\ -4 & 2 \end{bmatrix}$ ise det H_2 = -12<0

H f tanımsız olduğundan f fonksiyonu ne konveks nede konkavdır, $x_0$büküm noktasıdır.

2. yol : det (H- λI) = 0

$$\begin{vmatrix} -2 & -4 \\ -4 & 2 \end{vmatrix} - \lambda \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix} = \begin{vmatrix} 2-\lambda & -4 \\ -4 & 2-\lambda \end{vmatrix} = 0$$

p(λ) =(2- λ)^2-16= $\lambda^2 - 4\lambda - 12 = 0$

p(- λ) = $\lambda^2 + 4\lambda - 12 = 0$

λ 'ların bir kısmı pozitif bir kısmı negatif olduğundan büküm noktasıdır.

Örnek: f(x,y,z)=4x^2-6y^2-2xy+3xz-2y-4yz+1 fonksiyonunun ekstremumlarını bulunuz

Çözüm:

Gerek Şart: $\nabla f(x) = 0$ olmalıdır.Buna göre;

f_x=8x-2y+3z=0

f_y=-12y-2x-4z-2=0

f_z=3x-4y=0 olup buradan x_0(-6/7,-9/14,13/7) bulunur.

Yeter şart : f_{xx}=-2 ,f_{xy}=3 ,f_{yy}=-12 ,f_{yz}=-4 ,f_{zz}=0 olup Hessian matrisi;

$$H = \begin{bmatrix} 8 & -2 & 3 \\ -2 & -12 & -4 \\ 3 & -4 & 0 \end{bmatrix}$$ elde edilir.

1.test : det [8] = 8 >0 ve det $\begin{bmatrix} 8 & -2 \\ -2 & -12 \end{bmatrix} = -100 < 0$ (H_f(x) tanımsız , x_0 büküm noktası , fonksiyon ne konveks ne de konkavdır)

2.test : det (A- λI) = 0

$$\begin{bmatrix} 8 & -2 & 3 \\ -2 & -12 & -4 \\ 3 & 4 & 0 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = 0 \begin{bmatrix} 8-\lambda & -2 & 3 \\ -2 & -12-\lambda & -4 \\ 3 & -4 & -\lambda \end{bmatrix} = 0$$ ise buradan şu bulunur;

$$p(\lambda) = -\lambda^3 - 4\lambda^2 + 67\lambda + 28 = 0 \Rightarrow p(-\lambda) = \lambda^3 - 4\lambda^2 - 67\lambda + 28 = 0$$

üç negatif kök varsa negatif tanımlı , 3 pozitif kök varsa pozitif tanımlı bunun dışında tanımsızdır.

2.3. Kısıtlı Ekstremum Problemleri:

Bu bölümde sınır şartları ve kısıtlarıyla sürekli fonksiyonların optimizasyonu ele alacağız. Bu sınır şartları veya kısıtlar denklem formunda olabillir veya olmayabilir(2).

2.3.1. Eşitlik Kısıtlar

Eşitlik kısıtlarına sahip amaç foksiyonunun optimizasyonu için iki metod geliştirilmiştir. Bunlardan ilki Jacobian (Kısıtlı Türevler) metodu, ikincisi ise Lagrange metodudur(11).
Jacobian metodu Doğrusal Programlama için simpleks metodunun bir genellemesi olarak ele alınabilir. Gerçekten de simpleks metodu şartları Jacobian metodundan türetilebilir. İkinci bir metod olan Lagrange metoduda yine benzer olarak Jacobian metoduna benzer bir mantıkla geliştirilmiştir. Bu ilişki Lagrange metodunun ilginç bir ekonomik yorumunu kabul eder.

A) Lagrange Metodu

$$\frac{\partial f}{\partial g} = \nabla_{Y_0} \; J^{-1}$$

duyarlılık katsayıları f'nin optimum değeri üzerinde kısıtlardaki küçük değişikliklerin etkisini belirlemede kullanılır. Keza bu katsayılar sabittir. Bu özellikler eşitlik kısıtlarına sahip kısıtlı problemleri çözmek için kullanılır.

$$\lambda = \nabla_{Y_0} \; J^{-1} = \frac{\partial f}{\partial g}$$

Buradan; $\partial f - \lambda \partial g = 0$ bulunur. Bu denklem sabit noktalar için gerek şartlarda yeterlidir. Yani $\frac{\partial f}{\partial g}, \nabla_C f$ 'ye benzer olarak hesaplanır. Bu denklemleri sunmak için daha elverişli bir ifade de bütün x_j'lerin kısmi türevleri alınmak suretiyle elde edilir.

Böylece;

$$\frac{\partial}{\partial x_j}(f - \lambda g) = 0 \qquad (j = 1, 2, \ldots\ldots\ldots\ldots, n)$$

g = 0 kısıt denklemleri ile bu son denklem x ve λ'nın uygun değerlerini kabul eder ki sabit noktalar için gerek şartlar kâfidir.Eşitlik kısıtlarına sahip optimizasyon problemlerinin sabit noktalarının belirlenmesi işlemi Lagrange işlemi olarak adlandırılır. Bu metodu formulize eden ifade;

$L(X, \lambda) = f(x) - \lambda \; g(x)$ ile verilir.

Burada L fonksiyonu Lagrange fonksiyonu ve λ parametreleri de Lagrange çarpanları olarak bilinirler.

$$\frac{\partial L}{\partial \lambda} = 0 \quad \text{ve} \quad \frac{\partial L}{\partial X} = 0$$

denklemleri Lagrange fonksiyonu için gerek şartların oluşturulmasında direkt olarak kullanılır. Bir başka deyişle, g(x) = 0 kısıtları ile f(x) fonksiyonunun optimizasyonu
L (X , λ) Lagrange fonksiyonunun optimizasyonuna eşittir. Şimdide Lagrange metodu için yeter şartları ispatsız olarak tanımlayalım.

$$H^B = \begin{pmatrix} 0 & P \\ P^T & Q \end{pmatrix}_{(m+n)x(m+n)}$$

Burada;

$$P = \begin{pmatrix} \\ \nabla g_1(x) \\ \\ \nabla g_2(x) \\ . \\ . \\ . \\ \nabla g_m(x) \end{pmatrix}_{(m\times n)} \qquad \text{ve} \qquad Q = \left|\left| \frac{\partial^2 L(X,\lambda)}{\partial x_i . \partial x_j} \right|\right|_{(n\times n)} \qquad (\forall \text{ i,j için})$$

İşte bu şekilde tanımlanan H^B matrisine sınırlandırılmış Hessian Matrisi denir(11).

Verilen bir (X_0, λ_0) noktasında H^B sınırlandırılmış Hessian Matrisi ve $L(X, \lambda)$ Lagrange fonksiyonu değerlendirilirse;

1) Eğer H^B, (2m + 1)'inci mertebeden temel minör determinantı ile başlayan ve (n-m)'inci mertebeden temel minör ile son bulan determinantların işareti $(-1)^{m+1}$ ile değişiyorsa (X_0, λ_0) bir maksimum noktadır.

2) Eğer H^B, (2m + 1)'inci mertebeden temel minör determinantı ile başlayan ve (n-m)'inci mertebeden temel minör ile son bulan determinantların işareti $(-1)^m$ ile aynı işarete sahipse (X_0, λ_0) bir minimum noktayı belirtir(18).

Bu şartlar bir ekstremum noktayı tanımlamak için yeterlidir fakat gerek değildir. Bir başka deyişle bir sabit nokta yukarıdaki şartları sağlamaksızın ekstremum nokta olabilir.

Bu metodun dezavantajı işlem akışının hesaplama olarak pratik kararlar için uygun olmayışıdır. Bunun için;

$$\Delta = \begin{pmatrix} 0 & P \\ P^T & Q - \mu I \end{pmatrix}$$

matrisini bu şekilde tanımlayıp (X_0, λ_0) noktasında değerlendirelim. Burada P ve Q daha önce tanımladığımız gibi, μ ise bilinmeyen bir parametredir.

$|\Delta| = 0$ determinantını gözönüne alırsak;

$|\Delta| = 0$ polinom denkleminin (n -m) tane u_i reel kökünün herbiri için;

a) $|\Delta| < 0$ oluyorsa (X_0, λ_0) bir maksimum noktadır.

b) $|\Delta| > 0$ oluyorsa (X_0, λ_0) bir minimum noktadır.

2.3.2. Eşitsizlik Kısıtlar

Bu bölümde ilk olarak Lagrange metodunun genişlemesini ele alacağız. Yani sınırlı bir anlamda eşitsizlik kısıtlarını gözönüne alarak Lagrange metodunu genişleteceğiz. İkinci olarak ise eşitsizlik kısıtlarına sahip problemlerin analitik çözümü için Karush-Kuhn-Tucker gerek ve yeter şartları sunulmaktadır.

A. Lagrange Metodunun Genişletilmesi

$$\text{Max } z = f(x)$$

Kısıtlar $\quad g_i(x) \le 0 \qquad (i = 1,2, \ldots\ldots\ldots, m)$

$$x_i \ge 0$$

problemini gözönüne alalım.Lagrange metodunun genişletilmesinin esası şudur:

Eğer f(x)'in kısıtsız optimimu bütün kısıtları sağlamazsa, kısıtlı optimum çözüm uzayının bir sınır noktasında olmak zorundadır. Yani denklem formunda m kısıtta yeterli olmak zorundadır.Buna göre işlem adımları şu şekilde özetlenebilir.

Adım 1: Maksimum z = f(x) kısıtlı probleminin çözümünde eğer sonuç optimum bütün kısıtlarda yeterli ise k = 1 alınıp adım 2'ye geçilir.

Adım 2: Herhangi k kısıt işleme sokulur ve f(x), k aktif kısıt için optimize edilir. Eğer sonuç, kalan kısıtlar itibariyla uygunsa dururuz. Bu bir yerel minimumdur. Bir başka deyişle diğer aktif k kısıt işler hale getirilip adım tekrarlanır. Eğer alınan k aktif kısıtının bütün kümeleri uygun bir çözüm karşı gelmeksizin aynı anda gözönüne alınırsa adım 3'e geçilir.

Adım 3: Eğer k = m ise dur. Uygun çözüm yoktur. Yani k = k + 1 teşkil edilerek adım 2'ye geri dönülür. Bu işlemin önemli bir noktası sık sık ihmal edilmektedir. Bu nokta problemin uygun davrandığında bile mutlak optimum garanti edilememesidir. Diğer bir önemli nokta ise p < q için f(x)'in optimumunun p eşitlik kısıt için her zaman q eşitlik kısıttan daha kolay sağlanması gibi yanlış bir kanıya varmaktır.

B.Karush – Kahn – Tucker Şartları

Karush, Kuhn ve Tucker tarafından geliştirilen bu şartlar eşitsizlik kısıtlarına sahip doğrusal olmayan kısıtlı bir problemin sabit noktalarını tanımlamak için gerek ve yeter şartları sunar(19).Metoddaki gelişme temelde Lagrange metodu üzerindedir(11). Aşağıdaki eşitsizlik kısıtlı problemi gözönüne alalım.

i)Gerek Şartlar

maksimum $z = f(x_i)$

kısıtlar $g(x_i) \leq 0$

$x_i \geq 0$

Eşitsizlik kısıtlar negatif olmayan aylak değişkenlerin yaklaşık toplamı olarak denklemler içinde dönüştürülebilir.

$S = (S_1, S_2, ..., S_m)^T$ ve $S^2 = ({S_1}^2, {S_2}^2, ..., {S_m}^2)^T$ tanımlayalım.

Burada, **m** eşitsizlik kısıtların toplam sayısıdır. Buna göre Lagrange fonksiyonu;

$L(X, S, \lambda) = f(X) - \lambda\left[g(x) + S^2\right]$ olarak tanımlanır. Verilen kısıtlar ($g(x) \leq 0$) optimallik için gerek şarttır. Yani;

λ'nın negatif olmama (pozitif olmama) durumu maksimizasyon (minimizasyon) problemleri için verilen $g(x) \leq 0$ kısıtlarında optimallik için gerek şarttır. Burada sadece maksimizasyon durumunu ele alalım.

f'nin g'ye göre değişim oranı λ ile ölçüldüğünden; $\lambda = \dfrac{\partial f}{\partial g}$ 'dır.

Bu son ifadenin sağ tarafı $g \leq 0$ olduğundan artar ve çözüm uzayı daha az sınırlanmış olur.

Buna göre, f azalmaz, bu $\lambda \geq 0$ demektir. Benzer olarak, minimizasyon için f artmaz ki bu $\lambda \leq 0$ demektir. Eğer kısıtlar eşitlik halinde ise $[g(x = 0)]$ ise λ işarette sınırsız olur.

Bu λ üzerindeki kısıtlamalar Kuhn – Tucker şartlarını kısmen kapsamaktadır. Şimdi Lagrange fonksiyonu L (X, S, λ)'nın sırasıyla X, S ve λ 'ya göre kısmi türevlerini alalım.

$$\frac{\partial L}{\partial X} = \nabla f(x) - \lambda \nabla g(x) = 0 \quad \ldots\ldots\ldots\ldots\ldots\ldots (1)$$

$$\frac{\partial L}{\partial S_i} = -2\lambda_i . S_i = 0 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2) \quad (i = 1,2, \ldots . , m)$$

$$\frac{\partial L}{\partial \lambda} = -\left[g(x) + S^2 \right] = 0 \quad \ldots\ldots\ldots\ldots\ldots\ldots (3)$$

(2) ile ifade eden denklemden takip eden sonuçlar açığa çıkar.

a) $\lambda_i \geq 0$ **ise** $S_i^{\,2} = 0$ **'dır. Bu demektir ki eşitlik kısıt yoktur.**

b) Eğer $S_i^{\,2} > 0$ ise $\lambda_i = 0$ 'dır.

Yani; $\lambda_i = \dfrac{\partial f}{\partial g_i} = 0$ 'dır.

(2) ve (3) ile ifade edilen denklemlerin kümesinden;

$$\lambda_i \, g_i(x) = 0 \qquad (i = 1, 2, \ldots\ldots\ldots , m)$$

Bu yeni şart esas itibariyla $\lambda_i > 0$ 'a göre tekrarlanacak olursa $g_i(x) = 0$ veya $S_i^{\,2} = 0$ 'dır. Benzer olarak; eğer $g_i(x) < 0 \Rightarrow S_i^{\,2} > 0$ ve $\lambda_i = 0$ 'dır.

Aşağıda özetleneceği gibi X ve λ için Karush – Kuhn – Tucker şartları yukarıdaki maksimum probleminin bir sabit noktası olması için gerektir. Buna göre;

Maksimum nokta için;

$$\lambda \geq 0$$

$$\nabla f(x) - \lambda \nabla g(x) = 0$$

$$\lambda_i \, g_i(x) = 0 \qquad (i = 1, 2, \ldots\ldots\ldots , m)$$

$$g(x) \leq 0$$

Aynı şey minimum durum için uygulanırsa $\lambda \leq 0$ olmak zorundadır. Hem maksimum hemde minimum durumda, eşitlik kısıtlarına göre Lagrange çarpanları işarette sınırsız olmak zorundadır.

ii) Karush – Kuhn – Tucker Yeter Şartları

Eğer amaç fonksiyonu ve çözüm uzayı konvekslik ve konkavlık ile ilgili kesin şartlara sahip ise Karush – Kuhn – Tucker gerek şartları aynı zamanda yeterdir.

Bu şartları aşağıdaki gibi özetleyelim (Tablo 1).

Tablo.1

Optimizasyon Çeşidi	Gerek Şartlar	
	Amaç Fonksiyonu	Çözüm Uzayı
Maksimizasyon	Konkav	Konveks Küme
Minimizasyon	Konveks	Konveks Küme

Tablo 1'den de anlaşılacağı gibi amaç fonksiyonu ister konveks isterse konkav, çözüm uzayı konvekstir.Bu şartları sağlamak için aşağıdaki genel doğrusal olmayan programlama problemini gözönüne alalım.

(maksimum / minimum)

kısıtlar; $g_i(x) \le 0$ $(i = 1, 2, \ldots\ldots, r)$

$g_i(x) \ge 0$ $(i = r + 1, r + 2, \ldots\ldots, p)$

$g_i(x) = 0$ $(i = p + 1, p + 2, \ldots\ldots, m)$

$$L(X, S, \lambda) = f(x) - \sum_{i=1}^{r} \lambda_i \left[g_i(x) + S_i^2 \right] - \sum_{i=r+1}^{p} \lambda_i \left[g_i(x) + S_i^2 \right] - \sum_{i=p+1}^{m} \lambda_i g_i(x)$$

Burada λ_i, i kısıttaki Lagrangeçarpanları gösterir. Böylece Karush – Kuhn – Tucker yeter şartlarını belirlemek için gereken bilgiler bir tablo ile gösterilecek olursa;

Tablo.2

Optimizasyon Çeşidi	Gerek Şartlar			
	f(x)	$g_i(x)$	λ_i	
Maksimizasyon	Konkav	Konveks	≥ 0	$1 \le i \le r$
		Konkav	≤ 0	$r+1 \le i \le p$
		Lineer	sınırsız	$p+1 \le i \le m$
Minimizasyon	Konveks	Konveks	≥ 0	$1 \le i \le r$
		Konkav	≤ 0	$r+1 \le i \le p$
		Lineer	sınırsız	$p+1 \le i \le m$

Tablo 2'deki şartlar Tablo 1'deki şartların genellemesini tarif eder. Minimizasyon durumda $L(X, S, \lambda)$ konveks, maksimizasyon durumda $L(X, S, \lambda)$ konkav fonksiyondur. Bu genişletilecek olursa;

i. Eğer $g_i(x)$ konveks ve $\lambda \geq 0$ ise $\lambda_i g_i(x)$ konvekstir.

ii. Eğer $g_i(x)$ konveks ve $\lambda \leq 0$ ise $\lambda_i g_i(x)$ konkavdır.

Unutulmaması gereken birşey de doğrusal fonksiyonun hem konveks hemde konkav bir fonksiyon olmasıdır.

Sonuç:

Kısıtlı doğrusal olmayan problemlerin ekstremumlarını yerleştirmek için geliştirilen Klasik Optimizasyon Teorisi sayısal hesaplamalar için uygun değildir. Fakat bu teoriler hesap algoritmalarını geliştirmede temel teşkil ederler. Öyle ki Kuadratik Programlama Karush – Kuhn – Tucker gerek ve yeter şartlarını kullanan mükemmel bir örnektir.

III. DOĞRUSAL OLMAYAN PROGRAMLAMA ALGORİTMALARI

3.1. Optimizasyon Metodlarının Genel Tasnifi

TASNİF

Klasik Optimizasyon Metodları

Lagrange Çarpanlar Metodu

Jacobian (Kısıtlı Türevler) Metodu

Karush – Kuhn – Tucker Metodu (5, 11, 20)

Statik Optimizasyon Metodları

Bir Boyutlu Kısıtsız Optimizasyon Metodları

Arama Metodları

Fibonacchi Arama Metodu

Altın Oran Metodu

Ayrıntılı Arama Metodu

Çatallaşma Arama Metodu

Sınırsız Arama Metodu

Karesel İnterpolasyon Metodu

Rastgele Arama Metodu (1, 5, 12, 21,28)

Gradient Metodlar

Newton Raphson Metodu

Secant Metodu

Kübik İnterpolasyon Metodu

Direkt Kök Metodları

4.1- İkiye Bölme Metodu

4.2- Kirişler Metodu

4.3- Deneme Metodu

4.4- İterasyon Metodu (2, 12, 22)

Çok Boyutlu Kısıtsız Optimizasyon Metodları

Arama Metodları

Hooke ve Jeeves Metodu

Nelder ve Mead Metodu

Spendly, Hext ve Himswort'un Simpleks Metodu

Kompleks Metodu

Izgara - Ağ Metodu

Döngüsel Koordinat Metodu

Rosenbrock Metodu (1, 5, 8, 21, 23, 24, 28)

Gradient Metodlar

Davidon - Fletcher - Powell Metodu

En Dik İniş (Çıkış) (Eğim) Metodu

Fletcher - Reeves Metodu

Smith Metodu

Newton Metodu

Birleşik Doğrultuda Hareket Metodu (1, 3, 5, 8, 24, 25, 27, 30)

Dinamik Optimizasyon Metodları

Minimum Yol Problemi

2- Dinamik Programlama, Genel Matematiksel Optimizasyon, Optimallik İlkesi

3- Kesikli Karar Modelleri, Doğrusal Olmayan Sürekli Modeller

4- Dinamik Programlama ve Fonksiyonların Optimizasyonu

5- Kesikli Maksimum İlkesi (3, 21, 26)

Çok Boyutlu Kısıtlı Optimizasyon Metodları

Yarı İlmikleme

Gradient Yansıtma Metodu

Ceza ve Engel Fonksiyon Metodları

Ardışık Kısıtsız Optimizasyon Tekniği (SUMT)

Frank Wolfe Algoritması (1, 3, 5, 8, 21, 26, 29)

Özel Programlama Tipleri

Karesel Programlama

Geometrik Programlama

Ayrılabilir Programlama

Tamsayılı Programlama

Tahmini Programlama

Amaç Programlama

Lineer Çarpanlar Metodu (5, 8, 11, 24)

3.2. Seçilen Doğrusal Olmayan Programlama Tekniklerinin Seçim Sebepleri

Bu çalışmada ele alınan Newton – Raphson, Nelder – Mead, Gradient, Frank – Wolfe ve SUMT metodlarının diğer optimizasyon metodları içindeki yerleri ve önemleri açısından ele alındığında denilebilir ki, bunlar genel sınıflandırma içinde kısıtsız, kısıtlı, direkt (doğrudan arama = türevsiz) ve indirekt (türevli = gradient) yöntemleri şu noktalarda temsil ederler.

3.2.1. Newton Yöntemi:

Kısıtsız tek değişkenli y = f(x) fonksiyonunun optimum noktasını bulmada yerel olarak (lokal) karesel bir yakınsaklık gösterir. Karesel bir fonksiyonda minimuma bir iteresyonda ulaşır. Çok değişkenli fonksiyonlar için bir indirekt metod olarak ikinci kısmi türevlerden elde edilen bilgi ile ikinci (karesel) yaklaşımı kullanarak minimizasyon (veya maksimizasyon) için hızlı bir yöntem olarak kendini gösterir.

3.2.2. Nelder - Mead Metodu:

Türevleri kullanmaksızın bir arama doğrulltusunda değerlendirilecek f(x) fonksiyonunun düzenli geometrik şekil (simpleks) kullanarak köşe noktalarının seçimini formüle eder. Bu yöntemle sürekli olarak arama ile daha etkili biçimde daha karmaşık şekille (n + 1) köşe noktalarını değerlendirerek fonksiyonun minimumunu bulur.

3.2.3. Gradient Metodu:

Kısıtsız ve kısıtlı fonksiyonlarda türevleri kullanarak optimum doğrultuda ve adım büyüklüğünde ilerleyerek minimuma veya maksimuma en hızlı biçimde ulaşır (steepest descent - ascent). En önemli özelliği f(x)'in ölçeğine göre çok duyarlı olmasıdır.

3.2.4. Frank - Wolfe Metodu:

Kısıtlı fonksiyonları gradient yöntemi ile doğrusal forma sokup doğrusal programlama ile çözümüne hazırlık yapar. Doğrusal olmayan amaç fonksiyonu ve doğrusal kısıtlı problemi genelleştirilmiş İndirgenmiş gradient yöntemi ile doğrusallaştırma işlemini yürütür. En sık kullanılan karesel programlama için ideal bir yöntemdir.

3.2.5. SUMT (Ardışık Kısıtsız Optimizasyon Tekniği):

Çok değişkenli sistemlere ceza – fonksiyonu yöntemini uygulama Fiacco ve McCormick tarafından ilk defa ele alınmış olup Zangwill'in katkılarıyla şu şartlar geliştirilmiştir.

i. m adet $g_i(x) \geq 0$ formunda kısıt
ii. En az bir olurlu çözüm
iii. Amaç ve kısıt fonksiyonları sürekli
iv. Kısıtlı problemin kısıtsız hale dönüştürülmesi ile oluşan yeni problemin içerdiği K gibi pozitif ceza sabiti E gibi bir yakınsaklık ölçütü.

Bu yöntemlerin genel değerlendirilmesi sonucu şöyle özetlenebilir. Ceza maliyet temelli (SUMT gibi) yöntemler en az bir yerel optimuma yakınsama özelliğinden dolayı muhtemelen en kuvvetli yöntemlerdir. Bu durum örneklerde görülecektir.

3.3.Seçilen Tek - Çok Değişkenli Arama - Gradient Metodları

3.3.1. Newton Metodu:

Bir boyutlu problemler için geliştirilmiş Newton - Raphson metodunun n-boyutlu problemlerdeki karşılığı bir gradient arama metodu formundadır. Bu arama karesel yakınsaklığın arzu edilen niteliğe sahip olmasına rağmen ikinci kısmi türevlerinin ve ters matrisinin belirlenmesini gerektirir.

Varsayalım ki, $x = x^k$ geçerli arama noktasının civarında f(x), $f_s(x)$ kesilmiş Taylor serisi açılımı ile verilsin. Buna göre;

$$f_s(x) \cong f(x^k) + \sum_{i=1}^{n} \frac{\partial f(x^k)}{\partial x_i}(x_i - x_i^k) + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}(x_i - x_i^k).(x_j - x_j^k)$$

$$f_s(x) \cong f(x) \qquad (1)$$

Burada; $a_{ij} = a_{ji} = \left.\frac{\partial^2 f}{\partial x_i . \partial x_j}\right|_{x = x^k}$ (2)

$f_s(x)$ karesel fonksiyonunun sabit noktalarıaşağıdaki klasik yaklaşım ile;

$$\frac{\partial f_s(x)}{\partial x_m} = \frac{\partial f(x^k)}{\partial x_m} + \frac{1}{2}\sum_{n=1}^{n} a_{mi}(x_i - x_i^k) + \frac{1}{2}\sum_{j=1}^{n} a_{mj}(x_j - x_j^k)$$

$$\frac{\partial f_s(x)}{\partial x_m} = \frac{\partial f(x^k)}{\partial x_m} + \sum_{j=1}^{n} a_{mj}(x_j - x_j^k) \ldots \qquad (3)$$

$$\frac{\partial f_s(x)}{\partial x_m} = 0 \qquad (m = 1,\ 2,\ \ldots\ldots,\ n)$$

şeklinde elde edilir.(3) ifadesi matris formunda sunulduğunda;

$$A^k(x - x^k) = -\nabla f(x^k) = -g^k \ldots\ldots \qquad (4)$$

(4) ifadesinde $A^k, (n \times n)$ tipinde bir kare matris olup;

$$A^k = \begin{bmatrix} a_{11}\ a_{12}\ \ldots\ldots\ a_{1n} \\ a_{21}\ a_{22}\ \ldots\ldots\ a_{2n} \\ .\ \ .\ \ \ .\ \ \ . \\ .\ \ .\ \ \ .\ \ \ . \\ a_{n1}\ a_{n2}\ \ldots\ldots\ a_{nn} \end{bmatrix}_{(n \times n)}$$

$(A^k)^{-1}$ ters matrisinin var olduğunu farz edelim. Buna göre (4) ifadesinin her iki yanını $(A^k)^{-1}$ ile çarpar ve x yerine x^{k+1} yazılırsa;

$$(A^k)^{-1}.(A^k)(x^{k+1} - x^k) = -(A^k)^{-1}.g^k \qquad (5)$$

elde edilir. Bu son ifadeyi basitleştirirsek;

$$x^{k+1} = x^k - (A^k)^{-1}.g^k \qquad (6)$$

şeklini alır. Bu son ifade genelde bir gradient arama işlemini belirtir. Şimdi bu genellemeden sonra Newton arama metodunun bir iterasyonunda yapılması gereken işlem adımlarını özetleyelim.

i) İlk olarak n elemanlı gradient $x = x^k$'da hesaplanır.

ii) İkinci olarak ise a_{ij}'lerin sayısı $\frac{n.(n+1)}{2}$ olup bunlar f(x)'in kısmi türevlerine göre $x = x^k$'da belirlenir.

iii) Son olarak A^k kare matrisinin tersi $(A^k)^{-1}$ hesaplanır.

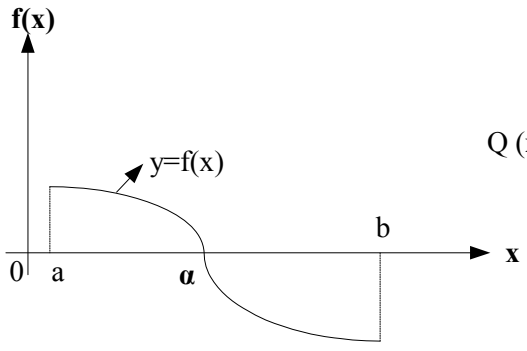
3.3.1.1. Bir Boyutta Newton – Raphson Arama:

Tek değişkenli y =f(x) fonksiyonu verilmiş olsun ve bu fonksiyonun ikinci mertebeden sürekli türevlere sahip olduğunu varsayalım. f(x) fonksiyonunun sabit noktaları;

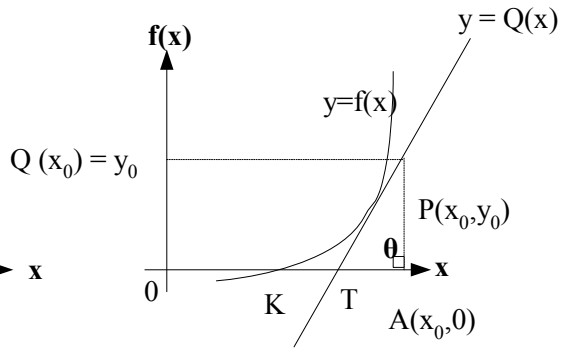
$$\frac{df}{dx} \equiv f'(x) = g(x) = 0$$

denklemi x'e göre çözülerek bulunur. Bir boyutlu problemlerde Newton - Raphson bir ardışık işlem tekniğidir. Öyleki g(x) = 0 denkleminin sıfırlarını bulmakta kullanılır.

Tek değişkenli fonksiyonlar için klasik yaklaşım, f(x)'in dönüm noktalarındaki x değerlerini $f'(x) = 0$ eşitliğinin çözümleri gibi bulur. Yaklaşık $y = f'(x)$ eğrisi ortalama çözümünün bulunmasını sağlayabilir.Eğer $f'(a)$ ve $f'(b)$ 'nin ters işaretlere sahip olduğu a ve b gibi iki bulunabilirse, bu tahmin kesin sürekliliği mecbur kılar. Bu aralıkta $a < \alpha < b$ şartına uyan bir α kökü olacaktır. (Şekil 6)



Şekil 6



Şekil 7

Newton metodu kabaca Q(x) denkleminin tahmini kökünü bulmayı sağlar. $\left[Q(x) = f'(x)\right]$

Şekil 7'deki P noktasının apsisi olan x_0 noktası Q(x)'in tahmini bir kökü olsun. PT, $y = Q(x)$ eğrisinin P noktasındaki tanjantı ve T noktası bu tanjant doğrunun x - eksenini kestiği nokta olsun. Buna göre OT genellikle K gerçek kökü için x_0 noktasından daha iyi bir tahmin olacaktır.

Böylece; $OT = OA - TA = x_0 - TA$

$$\frac{PA}{TA} = \text{Tan}\,\theta = Q'(x_0) \Rightarrow TA = PA \Big/ Q'(x_0) = \frac{Q(x_0)}{Q'(x_0)}$$

$$\Rightarrow TA = \frac{Q(x_0)}{Q'(x_0)} \text{ iken } TA = x_0 - x_1$$

$$\Rightarrow x_0 - x_1 = \frac{Q(x_0)}{Q'(x_0)}$$

$$\Rightarrow x_1 = x_0 - \frac{Q(x_0)}{Q'(x_0)} \text{ elde edilir.}$$

Bu son ifadeyi genişletecek olursak;

$$x_{k+1} = x_k - \frac{Q(x_0)}{Q'(x_0)} \quad k = 0, 1, \ldots\ldots, n$$

veya tek değişkenli y =f(x) fonksiyonu için Newton – Raphson formülü;

$$x_{k+1} = x_k - \frac{f(x_0)}{f'(x_0)} \quad (k = 0, 1, \ldots\ldots, n) \text{ olur.}$$

Böylece Newton metodu bu son formülasyon ile bir ardışık işlem haline gelmiş olur ve bu işlem x_{k+1} ile x_k arasındaki fark (hata payı) istenilen hassasiyete ulaşınca son bulur.

3.3.2. Nelder ve Mead Metodu

Nelder ve Mead Metodu Spendly, Hext ve Himsworth'un simpleks metodunun genişletilmiş bir şeklidir(1). Bu metod adından da anlaşılacağı gibi çok değişkenli bir fonksiyonun optimizasyonunu bulmak için Nelder ve Mead tarafından tasarlanmış bir simpleks metoddur.

Bu metotda (n + 1) adet nokta, bir düzenli simpleks (basit ve düzenli) olarak bilinen n-boyutlu Öklidyen uzayda karşılıklı olarak konulur. Bu nedenle iki boyutta simpleks bir dörtyüzlüdür(12).

İki boyutta bu metod üçgenin köşelerinde fonksiyon değerlerini karşılaştıran bir model arama metodudur. İki değişkenli bir $z = f(x, y)$ fonksiyonunun minimizasyon durumunu gözönüne aldığımızda, üçgenin en kötü köşesinde (w = worst vertex) $z = f(x, y)$ fonksiyonunun değeri en büyüktür.

Bu nedenle, bu en kötü köşe yeni bir nokta ile yer değiştirir. Böylece yeni bir üçgen elde etmiş oluruz. Artık arama işlemi bu yeni üçgen içinde devam ettirilir ve böylece bu işlem köşe nokta değerinde fonksiyon değerinin gittikçe küçüldüğü bir üçgenler dizisine dönüşür.

İşlem üçgenlerin küçülerek bir noktaya yaklaşması ile son bulur ki bu nokta istenen minimum noktadır. İki değişkenli $z = f(x, y)$ fonksiyonu için geliştirilen bu metod n-değişkenli $f(x) = f(x_1, x_2, ..., x_n)$ fonksiyonuna genelleştirilebilir. Nelder ve Mead'ın geliştirilmiş olduğu bu metod etkili ve hesap açısından kısadır.

Bu metodda simpleksin hareketi üç temel işlem (**Reflection**- Yansıtma, **Expansion**-Büyüme-Genişleme) ve (**Contraction**-Büzülme-Kısalma) ile sağlanır. Şimdi bu işlemleri safhaları ile açıklayalım.

i) Başlangıç Üçgeni BGW: (Initial Best Good Worst Triangle)

Min $z = f(x, y)$ fonksiyonunu gözönüne alalım. Bir üçgenin verilen üç köşesi ile işleme başlayalım.

$V_k = (x_k, y_k)..... k = 1, 2, 3$

Sonra bu üç köşe noktanın herbirinde fonksiyon değerlerini bulalım ve buna $z_k = f(x_k, y_k)$ (k = 1, 2, 3) diyelim. Bu değerleri de büyüklük sırasına göre $z_1 \le z_2 \le z_3$ olacak şekilde sıralayalım. Üçgenin köşe noktalarını;

$B = (x_1, y_1), G = (x_2, y_2), W = (x_3, y_3)$ ile gösterelim.Burada;

B = Best Vertex = En iyi köşe nokta

G = Good Vertex = Sonraki en iyi köşe nokta

W = Worst Vertex = En kötü köşe nokta olarak tanımlanabilir.

B noktası maksimum durumunda foksiyonda yerine konduğunda en büyük fonksiyon değerine sahip iken aynı nokta minimum durumunda ise en küçük fonksiyon değerine sahiptir.

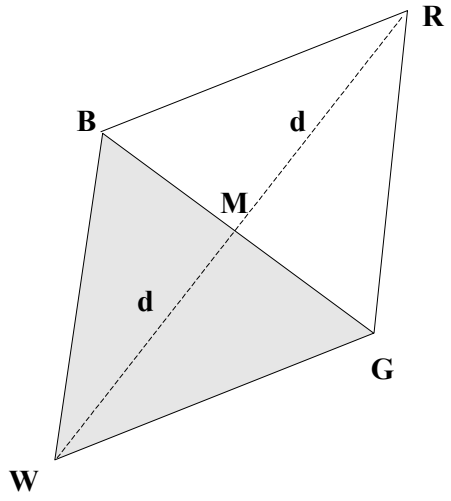
İyi Kenarın Orta Noktası:İyi kenardan maksat üçgenin $B = (x_1, y_1)$ ve $G = (x_2, y_2)$ noktalarını birleştiren doğru parçasıdır. Bu doğru parçasının orta nokta koordinatlarıda $B = (x_1, y_1)$ ve $G = (x_2, y_2)$ noktalarının koordinatlarının ortalamaları alınarak aşağıdaki şekilde bulunur.

$$M = \frac{B+G}{2} = \left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2} \right)$$

R Noktasını Kullanarak Yansıtma İşlemi:

W noktası ile B noktası arasındaki BGW üçgeninin kenarı boyunca hareket ettiğimizde f(x , y) fonksiyonu gittikçe azalan bir grafik çizer. Benzer olarak W noktasından G noktasına doğru üçgenin bu kenarı boyunca hareket ettiğimizde f(x , y) fonksiyonu yine azalan bir grafik çizer. Bu nedenle B ve G noktalarını birleştiren doğrunun karşı kenarı üzerinde W'ya uzak noktalarda f(x , y) fonksiyonu çok küçük değerler alır. Bunu test etmek için R noktasını BG kenarından geçecek şekilde yansıtarak elde edebiliriz.

R noktasını belirleyebilmek için ilk olarak BG kenarının M noktasını daha önce tanımladığımız gibi tespit ederiz. İkinci olarak W ile M noktalarını birleştirerek bir doğru elde ederiz. Bu doğrunun uzunluğuna d dersek, son olarak M noktasından d uzunluğunda bir doğru çizeriz ve bu doğrunun bittiği noktaya R deriz. Böylece yansıma işlemini kullanarak R noktasını elde etmiş oluruz(Şekil 8).



Şekil 8

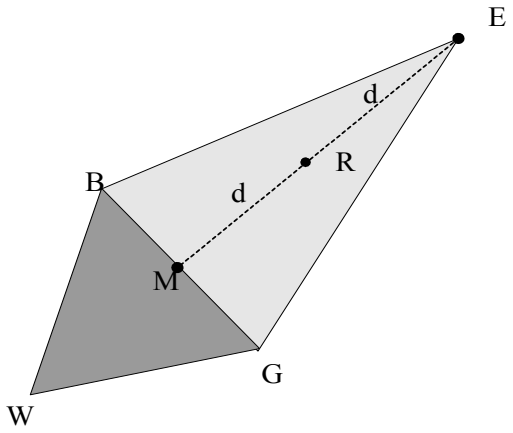
Şekilden de görüleceği gibi R noktasının vektörel notasyonu;

R = M + (M - W) = 2M - W şeklindedir.

Genişleme İşlemini Kullanarak E noktasını Elde Etmek:

Eğer R'deki fonksiyon değeri W'deki fonksiyon değerinden daha küçükse yani $f(R) \leq f(W)$ ise minimuma doğru bir yönde hareket ederiz. R noktası minimum noktaya uzak olabilir. Bu nedenle M noktasından R noktasına olan doğruyu R noktasından aynı d uzunluğunda genişlettiğimizde bitim noktası E'yi elde ederiz.

Böylece biz BGW üçgeninden genişleme işlemi ile BGE üçgenini elde ederiz.(Şekil 9)



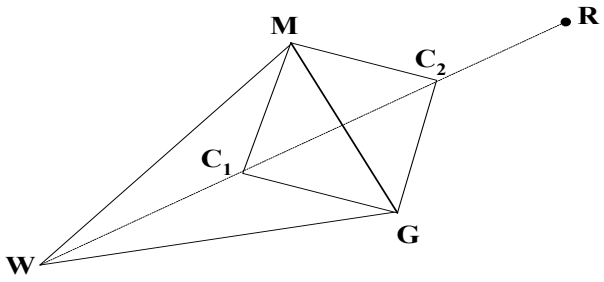
Şekil 9

Eğer E noktasındaki fonksiyon değeri R noktasındaki fonksiyon değerinden daha küçük ise yani f(E) < f(R) ise, R'den daha iyi bir köşe bulmuş oluruz.Genişleme işlemi ile elde ettiğimiz (Şekil 9)'daki E noktasına ait vektör formülasyonu;

$$E = R + (R - M) = 2R - M\text{'dir.}$$

Daraltma (Büzülme) İşlemini Kullanarak C noktasının Elde Edilmesi:

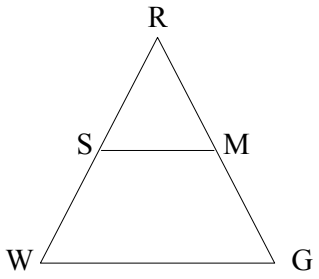
Eğer R noktasındaki fonksiyon değeri W noktasındaki fonksiyon değeri ile aynı ise yani, $f(R) = f(W)$ ise, başka bir nokta test edilmek zorundadır. Fonksiyon M noktasında küçük olabilir fakat biz M ile W'yi yeniden yazamayız çünkü yeni bir üçgen elde etmek zorundayız. WM ve MR doğrularının orta noktalarına sırasıyla C1 ve C2 dersek yeni üçgenimiz BGC olur. Burada BGC1 veya BGC2 olması iki boyutlu durumunda farketmez. Fakat çok boyutlu durumda farkader (Şekil 10).



Şekil 10

Üçgenin B noktasına Doğru Daraltılması:

Eğer C noktasındaki fonksiyon değeri W noktasındaki fonksiyon değerinden daha küçük değilse yani $f(C) \le f(W)$ ise G ve W noktaları B noktasına doğru daraltılmak zorundadır. (Şekil 11) Bu daraltma işlemini ilk adımda G noktası yerine BG doğrusunun orta noktası olan M noktası ve W noktası yerine BW doğrusunun orta noktası olan S noktası alınır. Sonraki ardışık işlem adımları B noktasına doğru hep orta noktalar alınarak devam ettirilir. İşlem B noktası elde edildiğinde son bulur. (Şekil 11)



Şekil. 11

Her bir adım için mantıksal kararlar: Yukarıda açıklanan her bir adımda yeni bir köşe bulunur ve bu W ile değiştirilir. Bunu bir algoritma olarak şu şekilde açıklayabiliriz.

$f(R) < f(G)$ ise Safha - 1

$f(R) \le f(G)$ ise Safha - 2 düzenlenir.

Safha 1: Yansıma veya Genişleme

$f(B) < f(R) \Rightarrow$ W yerine R alınır.

$f(B) \le f(R) \Rightarrow$ E ve f(E) hesaplanır.

$f(E) < f(B) \Rightarrow$ W yerine E alınır.

$f(E) \le f(R) \Rightarrow$ W yerine R alınır.

Safha 2: Büzülme veya Daralma

$f(R) < f(W) \Rightarrow$ W yerine R alınır.

Sonra $C = \frac{W + M}{2}$ alınır ve f(C) hesaplanır.

$f(C) < f(W) \Rightarrow$ W yerine C alınır.

$f(C) \leq f(W) \Rightarrow$ S yerine f(S) hesaplanır.

W yerine S ve G yerine M alınır(12).

Örnek: İki Boyutlu Problemler İçin Nelder - Mead Metodu

Çözüm:

Değişken Sayısı : 2

İterasyon Sayısı : 15

[1]. Min [2]. Max : l

Amaç Fonksiyon : $x + xy^2 - 3xy$

Vektör Elemanları

1. Vektör Elemanları : (0.0000, 0.0000)

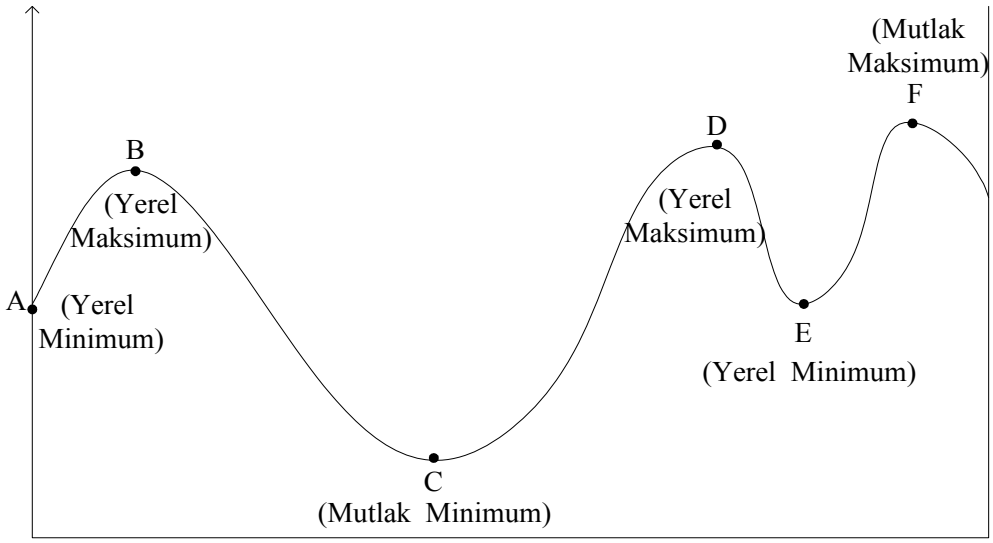
2. Vektör Elemanları : (2.0000, 0.0000)

3. Vektör Elemanları : (2.0000, 1.0000)

k	B(x_k, y_k)	G (x_k , y_k)	W(x_k,y_k)	F (x_k, y_k)
1	(2.0000, 1.0000)	(2.0000, 0.5000)	(1.0000,0.5000)	0.00000000000
2	(1.0000, 0.5000)	(1.5000,0.5000)	(1.5000,0.7500)	-0.75000000000
3	(1.5000, 0.7500)	(1.5000,0.6250)	(1.2500,0.6250)	-0.84375000000
4	(1.2500, 0.6250)	(1.5000,0.7500)	(1.1250,0.8125)	-0.87890625000
5	(1.1250,0.8125)	1.2500,0.6250)	(1.3438,0.7344)	-0.97119140625
6	(1.1250,0.8125)	(1.3438,0.7344)	(1.2188,0.9219)	-0.97119140625
7	(1.1250,0.8125)	(1.2188,0.9219)	(0.8281, 1.1328)	-0.97119140625
8	(0.8281, 1.1328)	(0.9766, 0.9727)	(1.0234, 1.0273)	-0.97475528717
9	(0.9766, 0.9727)	(1.0234, 1.0273)	(0.9141, 1.0664)	-0.99809467793
10	(0.9766, 0.9727)	(1.0234, 1.0273)	(0.9570, 1.0332)	-0.99809467793
11	(0.9570, 1.0332)	(0.9668, 1.0029)	(0.9902, 1.0303)	-0.99846400321
12	(0.9902, 1.0303)	(0.9668, 1.0029)	(1.0215,0.9834)	-0.99928985350
13	(1.0215, 0.9834)	(1.0059, 1.0068)	(0.9941,0.9932)	-0.99962122552
14	(0.9941,0.9932)	(1.0000, 1.0000)	(1.0078,0.9883)	-0.99987939186
15	(1.0000, 1.0000)	(1.0039,0.9941)	(0.9971, 0.9966)	-1.00000000000

3.3.3. Gradient Metodu:n - değişkenli bir $f(x) = f(x_1, x_2, ..., x_n)$ fonksiyonunun yerel optimumunu bulmak için 140 yılı aşkın bir süredir çok çeşitli algoritmalar geliştirilmiştir. Bu algoritmaların çoğu şu esasa göre tasarlanmıştır.

$f(x) = f(x_1, x_2, ..., x_n)$ n - değişkenli fonksiyonunun grafiğini tepe ve vadilerden oluşan bir sıradağlar topluluğu olarak gözönüne alırsak, vadilerin en alt tarafındaki bölgeler fonksiyonun minimumunu gösterirken tepelerin en üstündeki bölgelerde fonksiyonun maksimumunu temsil eder.Eğer minimum noktada değil isek bulunduğumuz noktadan aşağıya doğru minimumu buluncaya kadar hareketimize devam ederiz. Maksimum noktada değil isekte bulunduğumuz noktadan yukarıya doğru maksimumu buluncaya kadar hareketimize devam ederiz(2). (Şekil12)



Şekil 12

Bir gradientin yönü en dik çıkış (steepest ascent) yönüdür. Tersi ise en dik iniş (steepest descent)yönüdür. Yani en dik çıkış yönü $\nabla f(x)$ yönünde iken en dik iniş yönüde - $\nabla f(x)$ yönündedir(1).

n-değişkenli bir $f(x) = f(x_1, x_2, ..., x_n)$ fonksiyonunu gözönüne alalım.

Burada, $(x_1, x_2, ..., x_n)$ n - boyutlu öklidyen uzayda;

$$X = \begin{pmatrix} x_1 \\ x_2 \\ . \\ . \\ x_n \end{pmatrix}$$ sütun vektörü ile temsil edilir.

f(x) fonksiyonunun gradienti ise grad f(x) veya $\nabla f(x)$ ile gösterilir ki;

grad f(x) = $(f_1, f_2, ..., f_n)$ (12) veya

$$\nabla f(x) = \frac{\partial f}{\partial x_k} \qquad (k = 1, 2, \ldots\ldots, n) \qquad (1)$$

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}, ..., \frac{\partial f}{\partial x_n} \right) \text{ şeklindedir.}$$

ifadesinde kısmi türevler $f_k = \frac{\partial f}{\partial x_k}$ k = 1, 2, ……. , n x'de belirlenir(1).

Şimdi $X = (x_1, x_2, ..., x_n)$ uygun değerleri üzerinde herhangi bir kısıtlama olmaksızın $f(X) = (x_1, x_2, ..., x_n)$ konkav fonksiyonunun maksimizasyon problemini gözönüne alalım.

Bir boyutlu arama işlemini çok boyutlu problemimiz için genişletmeye çalışalım. Bir boyutlu problemlerde bayağı türev bir veya iki olası yolu yani x'in azalış veya x'in artış yönünü seçmek için kullanılır. Amaç bir noktayı araştırmaktır ki bu noktada türev sıfırdır. $\left[f'(x) = 0\right]$

Bu özellik çok değişkenli bir fonksiyon için şu şekilde genişletilebilir. Sabit noktaları araştırmak amaç iken, bu noktalarda bütün kısmi türevler sıfıra eşittir(7).

Bu nedenle bir boyutlu aramanın çok boyutlu aramaya genişletilebilmesi kısmi türevleri kullanarak hareket yönünde özel bir yön seçmeyi gerektirir. İşte bu gereksinim amaç fonksiyonunun gradientini kullanmayı kapsar.

f(x) amaç fonksiyonu yüksek mertebeden sürekli kısmi türevlere sahip olduğunda herbir x noktasında f(x) , $\nabla f(x)$ ile gösterilen bir gradiente sahiptir(7).

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2} ....., \frac{\partial f}{\partial x_n} \right)$$

Özel bir $x = x'$ noktasındaki gradient ise elemanları kısmi türevler olan ve x' noktasında değeri olan bir vektördür.

$$\nabla f(x') = \left(\frac{\partial f}{\partial x_1}, ..., \frac{\partial f}{\partial x_n} \right)\Bigg|_{x = x'}$$

Gradient arama işlemi kısıtsız bir problemin çözümü için etkili bir arama işlemidir. Bu işlem gradientin yönündeki hareketi koruyan ve x^* optimal çözümünü araştıran bir yöntemdir.

Bu nedenle normal olarak $\nabla f(x)$'in yönünde x'i sürekli olarak değiştirmek pratik degildir. Çünkü bu değişikliklerin serisi sürekli olarak $\frac{\partial f}{\partial x_j}$'lerin yeniden belirlenmesini ve her ikisinin yönünde değişiklik yapmayı gerektirir.

Bundan dolayı en iyi yaklaşım geçerli aşikar çözümden sabit bir yöndeki hareketi koruyan ve f(x) fonksiyonunun artışı durana dek devam eden bir yaklaşımdır(19).

f(x)'in artışının bittiği nokta ise son aşikar çözümdür ve bu noktada gradienthareketin yeni yönünü belirlemek için yeniden hesaplanacaktır. Bu yaklaşımla herbir ardışık işlem x' geçerli aşikar çözümün değişikliğini kapsar. Şöyle ki;

$$x' = x' + t^* \nabla f(x') \text{ alalım.} \qquad \text{(I)}$$

Burada, t^*, $f\left[x' + t \nabla f(x')\right]$'yi maksimum eden pozitif (t) değeridir.

$$f\left[x' + t^* \nabla f(x')\right] = \max f\left[x' + t \nabla f(x')\right] \qquad (t \geq 0)$$

(I) ifadesini genişletecek olursak;

$$x_j = x_j' + t\left(\frac{\partial f}{\partial x_j}\right)\Bigg|_{x = x'} \qquad j = (1, 2, \ldots\ldots, n)$$

elde edilir. Bu son ifade x_j için yalnızca sabitleri ve t'yi içerir. Yani f(x) t'nin bir fonksiyonudur. Bu gradient arama işleminin iterasyonları küçük bir ε toleransı ile $\nabla f(x) = 0$ olana dek sürdürülür. Yani;

$$\left|\frac{\partial f}{\partial x_j}\right| \leq \varepsilon \qquad j = (1, 2, \ldots\ldots, n)$$

Şimdi bu değindiğimiz özelliklere göre gradient işlemini basit bir algoritma ile tanımlayalım.

Başlangıç Adımı: İstenilen hassasiyet ε ve başlangıç aşikar çözüm (x')'yi alalım.

Ardışık Adımlar:

1) $x_j = x_j' + t\left(\frac{\partial f}{\partial x_j}\right)\Big|_{x = x'}$, $j = (1, 2, \ldots\ldots, n)$

Kümesi oluşturarak t'nin bir fonksiyonu olan $f\left[x' + t \nabla f(x')\right]$ ifadesi oluşturulur ve f(x) yerine bu ifadeler konulur. Son adıma gidilir.

2) $t \geq 0$ için bir boyutlu arama işlemi kullanılarak $f\left[x' + t \nabla f(x')\right]$'yi maksimum kılan $t = t^*$ bulunur.

3) $x' = x' + t^* \nabla f(x')$ oluşturulur ve son adıma geçilir.

Son Adım: $x = x'$ de $\nabla f(x')$ hesaplanır. Daha sonra $\left|\frac{\partial f}{\partial x_j}\right| \le \varepsilon$'un sağlanıp sağlanmadığı kontrol edilir. Eğer bu kontrol sağlanıyorsa x^* optimal çözümünün yaklaşık beklenen değeri için x' optimal çözüm alınır. Aksi halde ilk adıma dönülüp işlem tekrarlanır(7).

Bütün gradient arama tekniklerinin ortak özelliği

$\nabla f = g = \left[\frac{\partial f}{\partial f_1}, ..., \frac{\partial f}{\partial f_n}\right]$'i kullanmalarıdır.

Ayrıca; $x_{k=1} = x_k - t \nabla f(x_k)$ iterasyon işlemini kullanmalarıdır.

Örnek: Gradient arama işleminin nasıl yapıldığını göstermek için konkav bir fonksiyon için iki değişkenli kısıtsız optimizasyon problemini göz önüne alalım.

Max. $f(x) = 2 x_1 x_2 + 28x_1 - x_1^2 - x_1^4 + 4x_2 - x_2^2$

Çözüm: x_1 ve x_2 değişkenlerine göre kısmi türevlerini alırsak;

$\frac{df}{dx_1} = 2 x_2 + 28 - 2x_1 - 4x_1^3$

$\frac{df}{dx_2} = 2 x_1 + 4 - 2x_2$ olarak bulunur. Buna göre f(x) in grandienti

$\nabla f(x) = \text{grad } f(x) = (\frac{df}{dx_1}, \frac{df}{dx_2}) = (2 x_2 + 28 - 2x_1 - 4x_1^3 , 2 x_1 + 4 - 2x_2)$ dır.

grad f(x) = 0 şartından; $(2 x_2 + 28 - 2x_1 - 4x_1^3 , 2 x_1 + 4 - 2x_2) = (0 , 0)$

Böylece, $2 x_2 + 28 - 2x_1 - 4x_1^3 = 0$

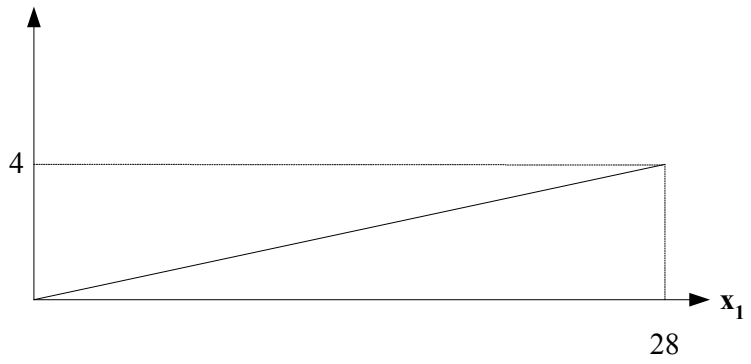
$2 x_1 + 4 - 2x_2 = 0$

Bu denklemlerin aynı anda çözümü ile optimal çözüm $(x_1, x_2) = (2 , 4)$ bulunur. Buna göre;$f(2,4) = 2.(2).(4) + 28 .(2) – (2)^2 - (2)^4 + 4.(4) – (4)^2 = 52$ ve grad f(2,4) = 8 + 28 – 4 – 32 , 4 + 4 – 8) = (0,0) bulunur.

Şimdi f(x) ve grad f(x) göz önüne alınarak gradient arama işleminin nasıl elde edilebileceğini görelim. Bu işleme başlamak için bir başlangıç aşikar çözüme ihtiyacımız vardır ki; bu noktada $f(x_1, x_2) = (0 , 0)$ dır. Böylece $(x_1 , x_2) = (0 , 0)$ başlangıç aşikar çözüm olarak alınabilir.

Bu nokta için gradient; grad f (0 , 0) = (28.4) Bu şu anlama gelir x = (0 , 0) noktasında f(x) deki max. artma oranı (0 , 0) dan (0 , 0) + (28.4) e hareket edilerek bulunur. Buna göre (0 , 0) ve (28.4) noktaları arasındaki doğru denklemi;(Şekil.13)

(0 , 0) + [(28.4) – (0 , 0)] = (28t , 4t) dır.Burada t > 0 olmalıdır.



Şekil.13

(0 , 0) dan (28,4) noktalarını birleştiren doğru boyunca ne kadar hareket etmeliyiz. Bir başka deyişle; (28t , 4t) doğrusu için t sıfırdan ne kadar arttırılmalıdır.

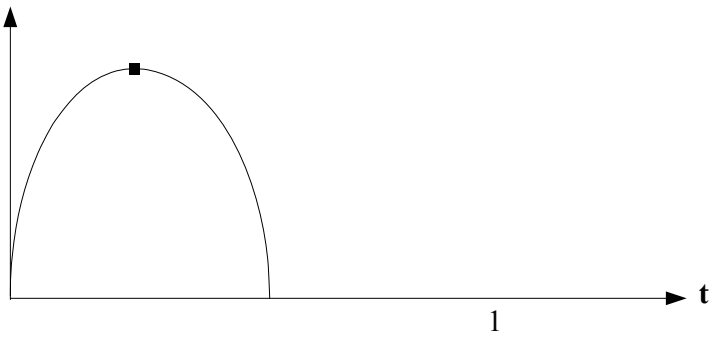
Bu soruların cevabı şudur: f(x) 'in artması durana kadar bu doğru boyunca hareket edilir. Yani $x_1 = 28t$ ve $x_2 = 4t$ değerleri fonksiyonda yerine yazılır.

$f (28t , 4t) = 2 (28t) (4t) + 28 (28t) - (28t)^2 – (28t)^4 + 4 (4t) - (4t)^2$

$f (28t , 4t) = 800t – 576t^2 – 614.656t^4$

f (28t , 4t) ' nin değeri max. olana kadar t'nin arttırılması gerekir. Bunu gerçekleştirmek için bir boyutlu arama işlemi ile t'nin maksimum değeri; $t^* = 0.0665$ olarak bulunur. (Şekil.14)

$f(28t,4t) = 800t – 576t^2 – 614.656t^4$



Şekil.14

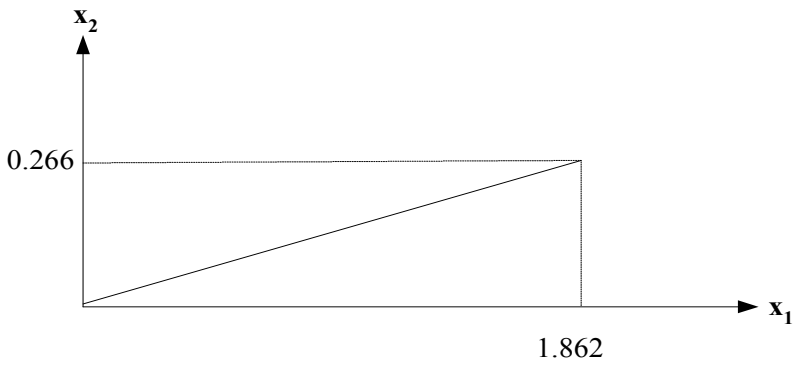
Buna göre yeni aşikar çözüm;$(x_1,x_2) = (28t^*,4t^*) = (1.862 , 0.266)$ olur.

f(1.862 , 0.266) = 38.63

Bu ilk iterasyonu;

Adım	x'	grad f(x')	x' + [grad f(x')]	t^*	x'+ t^* [grad f(x')]
1	(0,0)	(28.4)	(0+ 28t , 0+ 4t)	0.067	(1.862,0.266)

şeklinde bir genel tablo ile ifade edebiliriz.(Şekil.15)



Şekil.15

Şimdi ikinci iterasyonu ilk iterasyonun sonucuna göre tekrarlayalım. Geçerli aşikar çözüm (x_1 , x_2) = (1.862,0.266) noktasıyla (1.862,0.266) + (0.832 , 7.459) noktasını birleştiren doğru boyunca olacaktır.

Bu doğrunun denklemi;

(1.862,0.266) + (-1.028 ,7.193) = (1.862 – 1.028t , 0.266 + 7.193t)

bu doğru boyunca;

$x_1 = 1.862 - 1.028\ t$

$x_2 = 0.266 + 7.193\ t$ dır.

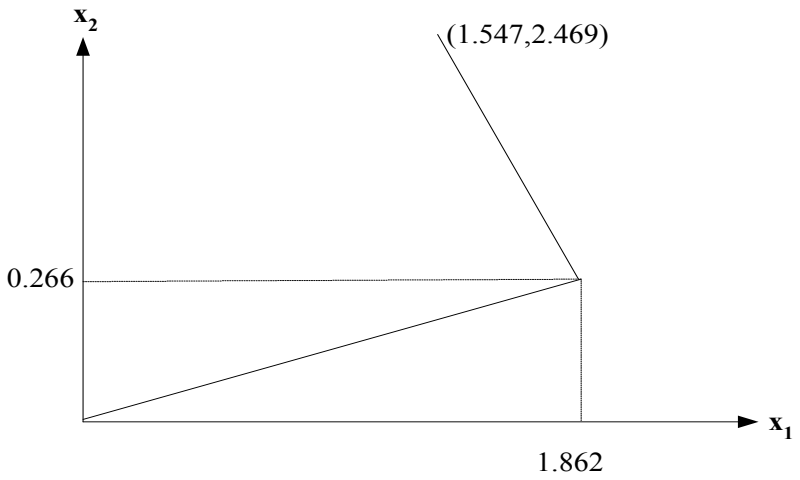
Böylece; $f(x_1, x_2) = 38.6 + 52.8\ t - 89.7\ t^2 + 8.14\ t^3 - 1.13\ t^4$ ve bir boyutlu arama işlemi gerçekleştirilirse; $t^* = 0.306$ olarak bulunur. Buna göre yeni aşikar çözüm;

(x_1 , x_2) = (1.862 - 1.028t , 0.266 + 7.193t) = (1.547 , 2.469)

Şimdi bu iki iterasyonu

Adım	x'	grad f(x')	x' + [grad f(x')]	t^*	x'+ t^* [grad f(x')]
1	(0,0)	(28.4)	(0+ 28t , 0+ 4t)	0.067	(1.862,0.266)
2	(1.862,0.266)	(-1.03,7.193)	(1.86-1.03t,0.27+7.19t)	0.306	(1.547,2.469)

şeklinde bir genel tablo ile gösterilebilir. A aşikar çözümlerin optimal noktaya doğru hareketlerini de yandaki şekille gösterebilir. (Şekil.16)



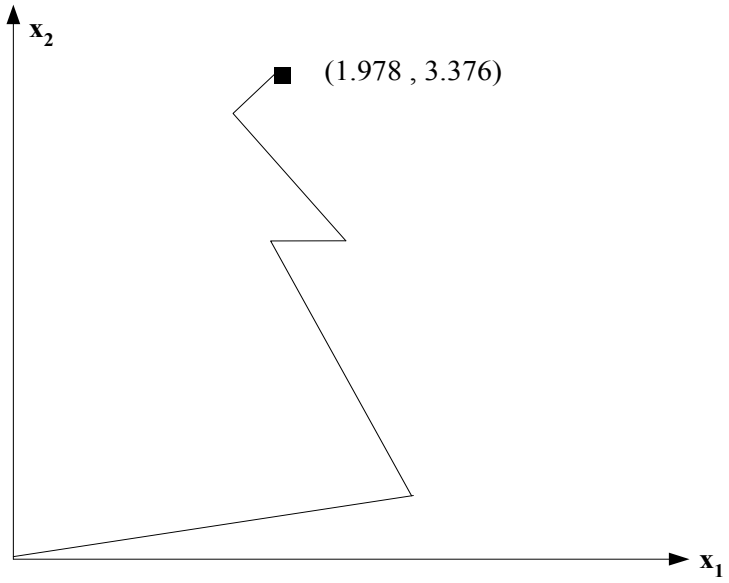
Şekil.16

Bundan sonraki iterasyonlarıda genel tablomuzla aşağıdaki şekilde verebiliriz.

Adım	x'	grad f(x')	x' + [grad f(x')]	t^*	x'+ t^* [grad f(x')]
1	(0,0)	(28.4)	(0+ 28t , 0+ 4t)	0.067	(1.862,0.266)
2	(1.862,0.266)	(-1.03,7.193)	(1.86 – 1.03t,0.27 + 7.19t)	0.306	(1.547,2.469)
3	(1.547,2.469)	(15.09,2.155)	(1.55+15.1t ,2.47 + 2.19t)	0.027	(1.948,2.526)
4	(1.948,2.526)	(-0.41,2.843)	(1.95+0.41t ,2.53 + 2.84t)	0.291	(1.978,3.354)
5	(1.829,3.354)	(6.59,0.99)	(1.83+6.59t ,3.35+ 0.99t)	0.023	(1.978,3.376)

İterasyonların sayısı istenilen hassasiyete bağlı olarak belirlenir. Eğer iterasyona devam edilecek olursa belli bir iterasyon sonunda tam optimal çözüme erişilebilir.

Bu örneğimizde 0.1 hata toleransı söz konusu iken tam optimal çözüm olan $(x_1 , x_2) = (2,4)$ noktasını elde etmek için yapılan iterasyonlar yukarıdaki tabloda sunulmuştur. Bu optimal çözüme ulaşıncaya kadar elde edilen bütün aşikar çözümler birleştirildiğinde son aşikar çözüm olan $(x_1 , x_2) = (2,4)$ (optimal çözüm) noktasına doğru zigzag bir grafik ortaya çıkar (Şekil.17).



Şekil.17

3.3.4. Konveks Programlama:

Konveks Programlama z = f(x) amaç fonksiyonunun konkav ve bütün $g_i(x)$ kısıtlarının konveks olduğu özel bir doğrusal olmayan programlama türüdür.

Bu varsayımlar problemi oldukça basitleştirir. $g_i(x)$ kısıtlarının konveks oluşu, mümkün çözümler kümesininde konveks olmasını gerektiren, f(x) amaç fonksiyonunun konkav oluşu ile bulunacak herhangi bir ekstremum yada yerel optimum çözümün aynı zamanda mutlak optimum çözüm olmasını sağlar.

Bir başka deyişle birçok yerel optimumları bulup bunların içinde mutlak optimumu seçmek zorunluluğu bulunmamakta, elde edilecek bir yerel optimum problemin mutlak optimum çözümü olmaktadır. Bu konuda da birçok metod geliştirilmiştir(18).

Fakat Konveks programlama problemlerini çözmek için her zaman kullanılan standart bir algoritma yoktur. Bu konuda farklı pek çok algoritmalar geliştirilmiş olup bunların avantajları ve dezavantajları mevcuttur.Bu algoritmalar temelde üç kategoride toplanmıştır.

1) Gradient Algoritmalar: Bu algoritmalar temelde korunmak şartı ile gradient arama işlemine benzer olarak geliştirilmiştir. Genel indirgenmiş gradient metod bunların en önemlilerindendir.

2) Ardışık Kısıtsız Algoritmalar: Penalty (Ceza) ve Barrier (Engel) fonksiyon metodlarını içerir. Bu algoritmalar orjinal kısıtlı optimizasyon problemlerini kendi içinde kısıtsız optimizasyon problemlerinin bir dizisine dönüştürür. Bundan sonra da kısıtsız optimizasyon problemlerinin herbiri gradient arama işlemi ile çözülebilir.

3) Ardışık Yaklaşık Algoritmalar: Bu algoritmalar Doğrusal yaklaşım ve Karesel yaklaşım metodlarını içerir. Bu algoritmalarla doğrusal olmayan amaç fonksiyonu Doğrusal veya Karesel yaklaşımların bir ardarda gelişi ile yeniden yerine konur.

Doğrusal kısıtlı optimizasyon problemleri için bu yaklaşımlar Doğrusal veya Karesel programlama algoritmalarının tekrar uygulanmasını kabul eder(7).

3.3.4.1. Frank - Wolfe Algoritması:

Bu algoritma bir amaç fonksiyonunun doğrusal kısıtlar altındaki optimizasyonunu inceler.

Amaç optimize f(x)

Kısıtlar $Ax \le b$

$x \ge 0$

Verilen olurlu çözüm x' iken f(x) amaç fonksiyonu için doğrusal bir yaklaşım f(x) fonksiyonunu $x = x'$ civarında birinci mertebeden Taylor serisine açmakla elde edilir. Buna göre;

$$f(x) \cong f(x') + \sum_{j=1}^{n} \frac{\partial f(x')}{\partial x_j} \quad (x_j - x'_j) \cong f(x') + \nabla f(x')\,(x - x')$$

$f(x')$ ve $\nabla f(x').x'$ değerleri sabit değerler olduğundan yeni amaç fonksiyonumuz;

$$f(x) = f(x') + \nabla f(x').x - \nabla f(x').x'$$

$f(x) \cong \nabla f(x').x + c$ şeklindedir.

Bu son ifadeyi;

$g(x) = \nabla f(x').x$ yazarsak bu bir doğrusal programlama problemine dönüşür ve optimal çözüm simpleks metod veya grafik çözümden elde edilebilir.

Bu çözüme x_{LP} diyelim. Doğrusal amaç fonksiyonu x'den x_{LP} arasındaki doğru parçası boyunca bir hareket gibi zorunlu olarak muntazaman artar.

Frank -Wolfe algoritmasına ait işlem adımları aşağıdaki gibidir.

Başlangıç Adımı: Bir başlangıç olurlu çözümü bulabilmek için doğrusal programlama işlemi uygulanarak bu aşikar çözüm $x^{(0)}$ bulunur. k = 1 yazılır.

Ardışık Adımlar:

1.Adım: j = 1, 2, 3, ... , n için $x = x^{(k-1)}$ de

$\frac{\partial f(x)}{\partial x_j}$ değerlendirilir ve $c_{j=} \frac{\partial f(x)}{\partial x_j}$ ifadesi yazılır.

2. Adım: Aşağıdaki doğrusal programlama problemine ait $x_{LP}^{(k)}$ optimal çözümü bulunur.

Maksimum $g(x) = \sum_{j=1}^{n} c_j x_j$

Kısıtlar $Ax \le b, x \ge 0$

3. Adım: $0 \le t \le 1$ arasındaki t değişkeni için;

$$x = x^{(k-1)} + t\left[x^{k}_{LP} - x^{(k-1)}\right]$$

için h(t) = f(x) ifadesi oluşturulur. Bundan sonra problemimiz $0 \le t \le 1$ için h(t) ifadesini maksimize etmeye dönüşür ki bu bir boyutlu arama işlemi ile çok kolay başarılabilir. Buradan bulunan t* değeri (1) ifadesinde yerine konarak yeni aşikar çözüm elde edilir. Daha sonra son adıma geçilir.

Son Adım: Eğer $x^{(k)}$ ile $x^{(k-1)}$ arasındaki fark istenilen hassasiyete ulaştığında aranılan optimal çözüm $x^{(k)}$ olarak alınır. Bir başka deyişle k = k + 1 alınarak ardışık adımlara geri dönülür.

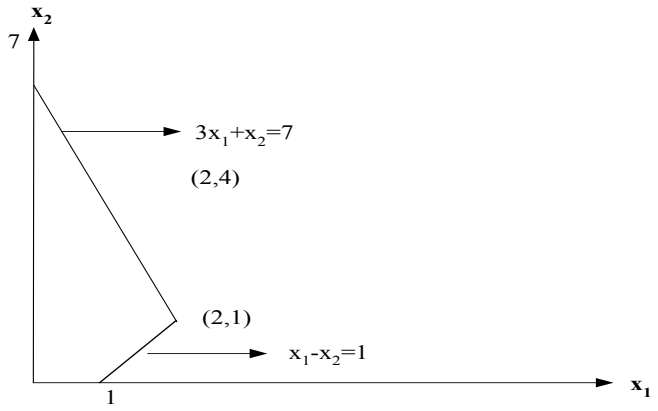
Frank – Wolfe Algoritması ile çözülen örneklerin genelinde karşımıza şu önemli sonuç çıkar. Aşikar çözümler iki veya daha fazla yörünge üzerinde değişirler. Bu çözümler yörüngeler üzerinde değiştiğinde, bu yörüngelerin kesiştiği yaklaşık noktanın tahmini optimal çözüm olacağı açıktır. Bu tahmin son aşikar çözümden daha iyidir. Bunun sebebi aşikar çözümlerin optimal çözüme doğru çok yavaş yakınsaması ve bundan dolayı optimal çözümden uzak olmalarıdır. Fank-wolfe algoritmasının nasıl uygulandığı göstermek için aşağıdaki doğrusal kısıtlı optimizasyon problemini göz önüne alalım.

Örnek: Max.f(x)= $32\,x_1 - x_1^{\,4} + 8x_2 - x_2^{\,2}$

$3x_1 + x_2 \le 7$

$x_1 - x_2 \le 1$ ve $x_1 \ge 0$ $x_2 \ge 0$

Çözüm: Bu problemin doğrusal kısıtlarına göre elde edilen uygun bölge Şekil.18' deki gibidir.



Şekil.18

Kısıtlar göz önüne alınmaksızın kısıtsız f(x) = $32\,x_1 - x_1^{\,4} + 8x_2 - x_2^{\,2}$ fonksiyonun $\Rightarrow$maximumu kısmi türevler alınıp sıfıra eşitlenerek $(x_1, x_2) = (2,4)$ olarak kolayca bulunabilir. Fakat kısıtlı maximumu bir başlangıç aşikar çözüme ihtiyacımız olacaktır. Bu başlangıç aşikar çözümü $(x_1, x_2) = (0,0)$ olarak alalım. Kısmi türevleri alıp bu noktada değerlendirirsek;

$$\frac{df}{dx_1} = 32 - x_1^{\,3} \Rightarrow \frac{df}{dx_1}(0,0) = 32 - 4(0)^3 = 32$$

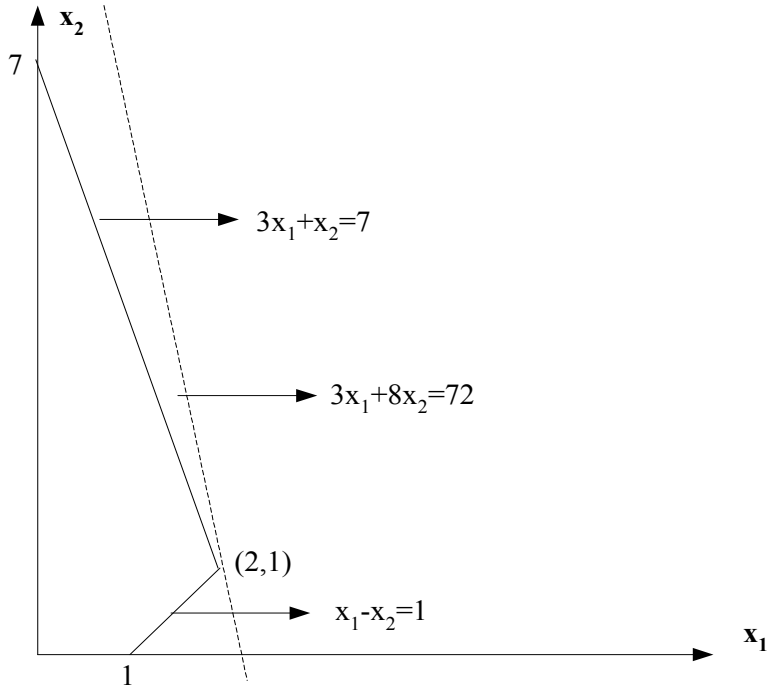
$$\frac{df}{dx_2} = 8 - 2x_2 \Rightarrow \frac{df}{dx_2}(0,0) = 8 - 2(0) = 8$$ olarak bulunur. Bu göre yeni amaç fonksiyonumuz bir doğrusal yaklaşım olarak ; $g_{(x)} = 32\,x_1 + 8x_2$ dır.

Buna göre yeni problemimiz;

Max. $g_{(x)} = 32\,x_1 + 8\,x_2$

Kısıtlar $3x_1 + x_2 \le 7$

$x_1 - x_2 \le 1$ ve $x_1 \ge 0$; $x_2 \ge 0$ olarak tekrar yazılır.Bu problemin orijinal kısıtlarla çözümü $(x_1, x_2) = (2,1)$ noktasını çözüm olarak kabul eder. (Şekil.19)



Şekil.19

$$g(2,1) = 32\,(2) + 8\,(1) = 72$$

(2,1) noktasına yakın olmadığından bu iyi bir yaklaşım değildir. Bu nedenle (0,0) ve (2,1) noktalarını birleştiren doğru parçası üzerinde f(x) 'i en büyük yapan $X = (x_1, x_2)$ noktasını bulmaya çalışalım.

f(2,1) = 55 iken g(2,1) = 72 dir. Buna (2,1) deki yaklaşım iyi bir yaklaşım değildir.(0,0) ve (2,1) noktaları arasındaki doğrunun denklemi;

(0,0) + t [(2,1) – (0,0)] = (2t,t) ($0 \leq + \leq 1$) Buna göre;

$x_1 = 2t$, $x_2 = t$ olarak alınıp f(x)'de yerine yazılırsa t'ye bağlı bir h(t) fonksiyonu elde edilir. f

$$(x_1, x_2) = f(2t, t) = h(t) = 32(2t) - (2t)^4 + 8t + t^2$$

$h(t) = 72t - t^2 - 164\,t^4$ $\quad (0 \leq t \leq 1)$

t'ye bağlı bu h(t) fonksiyonuna bir boyutlu arama işlemi uygulanıyorsa;

$$\frac{df}{dx_1}(2,1) = 32 - 4\,(2)^3 = 0$$

$$\frac{df}{dx_2}(2,1) = 8 - 2\,(1) = 6$$ ve yeni problemimiz; $g(x) = 0.\,x_1 + 6.\,x_2$

Kısıtlar $3x_1 + x_2 \leq 7$

$x_1 - x_2 \leq 1$ iken çözüm $(x_1, x_2) = (0,7)$ dir.

$g(0,7) = 42$, $f(0,7) = 7$

Şimdi (2,1) ve (0,7) noktaları arasındaki doğru bulunur.

$(2,1) + t[(0,7) - (2,1)] = (2-2t, 1+6t)$ $(0 \leq t \leq 1)$

$h(t) = f(2-2t, 1+6t) = 55 + 36t - 132t^2 + 64t^3 - 16t^4$

h(t) ' yi maximum yapan t değeri;

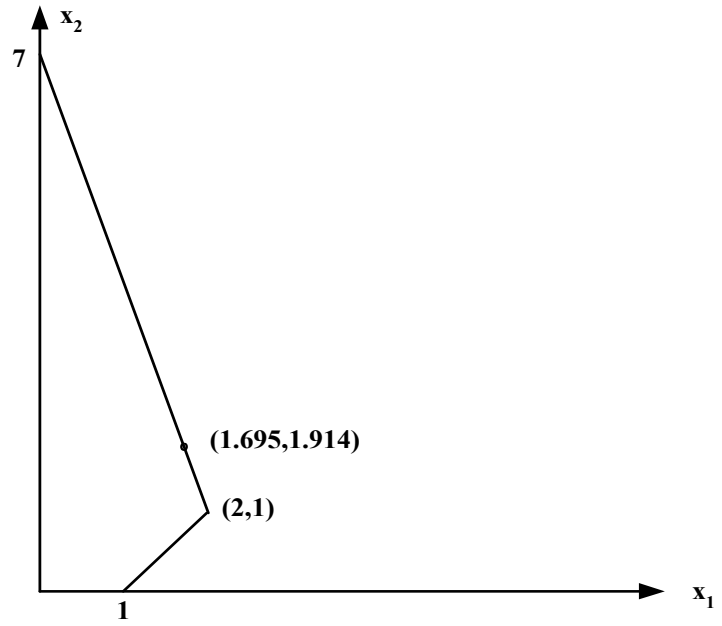
$t^* = 0.1524$ dür. Buna göre,$(x_1, x_2) = (2-2t^*, 1+6t^*) = (1.695, 1.914)$ yeni aşikar çözümdür.Yine benzer işlemleri bu son aşikar çözümümüz için uygulayalım.

$f(x) = 32x_1 - x_1^4 + 8x_2 - x_2^2$ iken

$$\Rightarrow \frac{df}{dx_1}(1.695,1.914) = 32 - 4(1.695)^3 = 12,52$$

$$\Rightarrow \frac{df}{dx_2}(1.695,1.914) = 8 - 2(1.914) = 4,17$$ belirlenir ve yeni amaç fonksiyonumuz;$g(x) = 12.52x_1 + 4.17x_2$ olur.

Bu yeni amaç fonksiyonu ile orijinal kısıtlı problemimizi yukarıdaki gibi çözdüğümüzde yeni aşikar çözüm $(x_1, x_2) = (1.695, 1.914)$ olarak bulunur.(Şekil.20)



Şekil.20

3.3.5. Konveks Olmayan Programlama :

Konveks programlamada bir yerel ekstremum aynı zamanda bir mutlak ekstremumu gerektirir. Fakat doğrusal olmayan programlama problemlerinde bu özellik her zaman karşımıza çıkmayabilir. Bu nedenle konveks olmayan programlama problemlerini uygulayabilmek için ortak bir yaklaşım izlemeniz gerekir. Bu ortak yaklaşımda arama işlemi algoritma vasıtasıyla yapılabilir ki bu işlem bir yerel ekstremum bulana dek uygulanır ve sonra bu işleme tekrar başlanarak mümkün olduğu kadar farklı yerel ekstremumlar bulunur. Bu ekstremumların en iyisi de mutlak ekstremum olarak alınır. Normal olarak bu arama işlemi konveks programlamanın bütün şartları sağlandığında mutlak ekstremumu bulmak için tasarlanmıştır. Fakat konveks programlamanın şartları sağlanmadığı durumlarda bir yerel ekstremumu bulmak için de kullanılabilir.

İşte bu kısaca değindiğimiz özete uygun bir arama işlemi 1960'lı yıllarda geliştirilmiş ve çok geniş bir uygulama alanı bulmuştur. Bu algoritma <<Sequential Unconstrained Mınımazıtıon Technıque>> (SUMT) adıyla bilinen ardışık kısıtsız optimizasyon tekniğidir (7).

3.3.6. SUMT (Ardışık Kısıtsız Minimizasyon Tekniği) Algoritması :

Ardışık Kısıtsız Minimizasyon Tekniği (SUMT) ilk olarak 1961'de Carroll tarafından ileri sürülmesine rağmen bu metodun sadece teorisi ve özellikleri üzerinde durmayan , aynı zamanda bu metodun genişlemesi için pratik bir sistem geliştiren Fiacco ve McCormick tarafından araştırılıp geliştirilmiştir.

Pratik bir kural ile metodu kullanmanın mümkün olması için teorrik yaklaşım (yakınsaklık) özelliğini gerçek bir hesaplamaya dönüştürmek şarttır(1).

SUMT algoritması esas itibarı ile 2 bölümde incelenir. Bunlardan ilki uygun bölge içinde yakınsak olan bir penalty (ceza) fonksiyonunu kullanan ve uygun olmayan çözümlerden bahseden harici nokta algoritmasıdır. [exterior Point Algorithm] ikincisi ise uygun bölge içinde barrier (engel) fonksiyonunu kullanan ve direkt uygun çözümlere değinen dahili nokta algoritmasıdır. [interior Point Algorithm].

Adından da anlaşılacağı gibi Ardışık Kısıtsız Minimizasyon Tekniği (SUMT) çözümleri orijinal problemin çözümü ile birleşen kısıtsız optimizasyon problemlerinin bir dizisi ile orijinal problemin yerini alır.

Bu yöntem çok etkili sonuçlar verir. Buna sebep kısıtlı bir optimizasyon problemi yerine kısıtsız bir optimizasyon problemini çözmenin daha kolay olmasıdır.Ayrıca bu tip problemlerin Gradient Metodla çözülebilmesidir (7).

SUMT algoritmasında çok genel haliyle bir Gradient metoddur. Burada amaç fonksiyonunun konkav ve her bir kısıt fonksiyonun konveks olduğu varsayılmıştır. Bu algoritma temelde kısıtlı problemi kendi içinde kısıtsız bir probleme dönüştürür. Bu haliyle işlem Lagrange çarpanlar metodunu andırmaktadır.

Böylece;$P(x,r)=f(x)+r\sum_{i=1}^{n}\left[\frac{1}{b_i-g_i(x)}-\sum_{j=1}^{n}\frac{1}{x_j}\right]$.....(*)

şeklindeki ifade edilen problemimizi gözönüne alırsak , buproblemin steepest ascent (en dik çıkış) metoduyla çözebiliriz.

Burada $\mathbf{b_i- g_i(x)}$ konveks iken; $\frac{1}{b_i-g_i(x)}$ konkavdır(11).

Bu son ifade P(x,r)'nin x'de konkav olması demektir.sonuç olarak P(x,r) bir tek ekstremuma sahiptir.Ayrıca orijinal kısıtlı problemin optimizasyon (maksimizasyon veya minimizasyon) P(x,r)'nin optimizasyonuna eşittir ve x_0 başlangıç aşikar çözüm noktası uygun bölge içinde bir nokta olmak zorundadır. (interir point). Buna göre bu çözümü takip eden diğer çözümlerde uygun bölge içinde kalmak zorundadır.

(*) ifadesi ile verilen problemimizde r>0 skalerinin farklı iki değeri için P(x,r)'yi ekstremum yapan x'in optimum değerleri yaklaşık olarak aynı ise SUMT algoritması bu optimum noktada son bulur.

Maksimum P(x,r) = f(x) – rB(x) problemini göz önüne alalım. Kısıtsız problemlerin dizisini oluşturan her bir problem için buradaki r skalerini kesin pozitif yapan bir r değeri seçilir ve problem x'e göre çözülür. Burada B(x) bir barrier (engel) fonksiyonudur ve şu özelliklere sahiptir.

i) x uygun bölge sınırına uzak iken B(x) küçüktür.

ii) x uygun bölge sınırına yakın iken B(x) büyüktür.

x uygun bölge sınırında iken $B(x) \rightarrow \infty$'a yaklaşır.

Böylece, P(x,r)'yi artırmak için arama işlemi uygun bir çözüm ile başlar. B(x)'in genel formu;

$$B(x)=\left[\sum_{i=1}^{n}\frac{1}{b_i-g_i(x)}+\sum_{j=1}^{n}\frac{1}{x_j}\right]$$

x'in uygun değerleri için şunu görebiliriz ki; negatif olmayan kısıt veya uygun fonksiyon için her terimin paydası x'in kısıt sınırından olan uzaklığı ile orantılıdır.

Bu nedenle her terim sınırda olmayan bir terimdir ve kısmi kısıt sınırına göre B(x)'in üç özelliğine sahiptir. B(x)'in diğer bir etkili özelliği dee konveks programlamanın tüm şartları sağlandığında P(x,r)'nin konkav fonksiyon olmasıdır. Ardışık Kısıtsız Minimizasyon Tekniği (SUMT) kısıtsız optimizasyon problemlerinin bir çözümünü kapsadığından r'nin uygun değerleri için r sıfıra yaklaşır. Bu yaklaşımda her bir yeni r değeri daha önceki r değerinin bir θ ($0< \theta <1$, $\theta = 0.01$) sayısı ile çarpılarak bulunur**.**

Örneğin: $r_1 = 1 \rightarrow r_2 = r_1.\theta = 1.0.01 = 0.01$ r'nin sıfıra yaklaşması halinde P(x,r) ,f(x) fonksiyonuna yaklaşacaktır. Şöyle ki P(x)'nin uygun yerel maksimumu orijinal problemin yerel maksimumuna yakınsayacaktır. Bu nedenle sadece kısıtsız optimizasyon problemini çözmek gerekir.

Orijinal problem konveks programlamanın şartlarını sağladığında bize yardımcı olacaak bir bilgi verir . Eğer, $\bar{x}$,P(x,r)'nin bir mutlak maksimumu ise;

$$\mathbf{f(\bar{x}) \leq f(x^*) \leq f(\bar{x}) + rB(\bar{x})\ '\ dir}$$

Burada, x* orijinal problem için bilinmeyen optimum çözümdür. Böylece rB(x) amaç fonksiyonunun değerindeki maksimum hatadır ve $|\ \bar{x} - x^*\ |$ değerine eşittir.

Daha ileride bu hata için bir hassasiyet sınırı getirilirse rB(x) bu hassasiyetten daha küçük kalınca işlem son bulur. Fakat konveks olmayan programlama problemlerinde maksimum hata için böyle bir garanti verilmez.

Bu bilgilerin ışığı altında Ardışık Kısıtsız Minimizasyon Tekniği (SUMT)'nın işlem adımları şu şekilde özetlenebilir.

Başlangıç Adımı : Uygun bölgenin sınırı üzerinde olmayan bir x0 başlangıç noktası alalım ve k = 1 atayalım. Kesin pozitif r skalerini ve θ sabitini seçelim.

[Genellikle r = 1, θ = 0.01 olarak alınır].

Ardışık Adımlar :

$$P(x,r)=f(x)-r\left[\sum_{i=1}^{n}\frac{1}{b_i - g_i(x)} + \sum_{j=1}^{n}\frac{1}{x_j}\right]$$

Fonksiyonunun x_k yerel ekstremumunu bulmak için x_{k-1} noktasından başlayarak daha önce tanımladığımız gradient işlemini uygulayalım.

$$\mathbf{[x_k = x_{k-1} - t \nabla f(x_k)]}$$

Son Adım : Eğer x_{k-1} ile x_k arasındaki fark ihmal edilebilirse işlem x_k'nın orijinal problemin tahmini yerel ekstremumu olarak kullanılması ile son bulur .

Aksi halde k = k + 1 ve $r_k = \theta \, . \, r_{k-1}$ alınarak ardışık adıma geri dönülür.

Bu algoritma konveks programlamanın şartları sağlanana dek başlangıç tahmini uygun çözümlerle başlanarak tekrarlanır. Yerel değerlerin en iyisi ekstremum olarak kullanılır.

Sonuç olarak Ardışık Kısıtsız Minimizasyon (maksiminizasyon) Tekniği eşitlik kısıtlarına sahip $[g_i(x) = b_i]$ problemler içinde kolayca genişletilebilir. Her eşitlik kısıt;

$$\frac{-r}{b_i - g_i(x)} \text{ yerine } \frac{-[b_i - g_i(x)]^2}{\sqrt{r}}$$

alınarak SUMT algoritması aynı şekilde tekrarlanır. SUMT algoritmasının nasıl çalıştığını görmek için aşağıdaki örneği verelim.

Örnek: Ardışık Kısıtsız Minimizasyon Tekniği(SUMT)

Alt Kısıtlardaki Değişkenlerin Sayısı : 2

Alt Sınır Kısıtsız değişkenlerin Sayısı : 0

Eşitsizlik Kısıtların Sayısı : 1

Eşitlik Kısıtların Sayısı : 0

$$P(x,r)=f(x) \quad -\frac{r}{B_1(x)} - \frac{r}{L_1(x)} - \frac{r}{L_2(x)}$$

Burada; $f(x) = 4x_1\text{-}1\, x_1^4 + 2x_2 - 1x_2^2$

$B_1(x) = 5\text{-}4x_1\text{-}2x_2$

$L_1(x) = 1x_1$

$L_2(x) = 1x_2$

Ardışık Kısıtsız Minimizasyon tekniği Çözümü:

k	r	x_1	x_2	f(x)
0		0.5	0.5	2.6875
1	1	0.669	0.716	3.3954
2	0,01	0,871	0,671	3,8012
3	0,00001	0,89,	0,712	3,8526
4	0,0000001	0,894	0,712	3,8541
5	0	0,894	0,712	3,8543
6	0	0,894	0,712	3,8543

3.4. UYGULAMALAR VE YORUMLARI

3.4.1. Seçilen Doğrusal Olmayan Programlama Tekniklerinin Karşılaştırmalı Genel Yorumu

Bu çalışmamızda seçtiğimiz Doğrusal Olmayan Programlama Tekniklerini tek-çok değişkenli, kısıtlı-kısıtsız, direkt-indirekt olmak üzere üç ana başlık altında tasnif edebiliriz. Buna göre Newton-Raphson Metodu tek değişkenli , indirekt(türev kullanan), kısıtsız bir ardışık işlem metodudur.

Nelder - Mead Medodu ise çok değişkenli kısıtsız ve direkt (türev kullanmayan) bir arama metodudur. Gradient metodu ise adından da anlaşılacağı gibi çok değişkenli ,kısıtsız ve indirekt bir arama metodudur. Frank-Wolfe Algoritması ise çok değişkenli , doğrusal kısıtlı ve indirekt bir Gradient arama işlemidir. SUMT Algoritması ise çok değişkenli, doğrusal ve doğrusal olmayan eşitlik ve eşitsizlik kısıtlı ve indirekt olan bir Gradient arama işlemidir.

Bu tasnife göre karşılaştırmalı yorumu kısıtlı metodları kendi aralarında , kısıtsız metodları kendi aralarında olmak üzere ayıralım.

Newton-Raphson Metodu y = f(x) tek değişkenli fonksiyonunun sıfırlarını bulan metodlarının arasında gerçek köke en hızlı yakınsayan bir ardaşık arama işlemidir. Bu nedenle çalışmada tek değişkenli fonksiyonları temsilen seçilmiştir.

Nelder - Mead Medodu ile Gradient Metodu çok değişkenli kısıtsız fonksiyonların optimizasyonunu bulan iki metod olmasına rağmen Nelder ve Mead Medodu türev kullanmayan (direkt) bir arama işlemi , Gradient Metodu ise türevleri kullanan (indirekt) arama işlemidir.

Gradient arama işleminin en önemli özelliği optimum çözüme steepest ascent (en dik çıkış), steepest descent (en dik iniş) veya zigzag şeklinde bir grafik çizerek yakınsamasıdır. Bu Gradient arama işleminin en karakteristik özelliğidir. Bu nedenle uygun bölgede seçilen her başlangıç aşikar çözüm optimum çözüme hızlı bir şekilde yakınsar. Uygulama 4 ve Uygulama 5 bunu açıkca ortaya koymaktadır. Fakat bu metodun bir dezavantajı adım uzunluğunun küçülmesi durumunda bir sonraki iterasyona geçememesidir. Bu bilgisayarlardaki alt taşması , üs taşması gibi hatalarından kaynaklanmaktadır. Bu nedenle ancak uygun başlangıç aşikar çözümlerin, seçimi ile tam optimum çözüme ulaşabiliriz.

Nelder – Mead tarafından geliştirilen metod ise bir simpleks metodudur ve çok değişkenli kısıtsız fonksiyonların optimizasyonunu bulmakta oldukça etkilidir. İki boyutta simpleks bir üçgen olduğundan, ilk üçgenin üç köşe noktası başlangıç vektörler alınarak işleme başlanır.

Bu köşe noktalarda maksimum problem için en büyük, minimum problem için en küçük fonksiyon değerine sahip noktaya Best Vertex (En iyi köşe nokta), sonraki en iyi noktaya Good Vertex (Sonraki en iyi köşe nokta) ve maksimum problemlerde en küçük, minimum problemlerde en büyük fonksiyon değerine sahip noktaya da Worst Vertex (En kötü köşe nokta) denir. Bu işlem her iterasyonda tekrarlanarak iki boyutlu problmlerde üçgenlerin bir dizisi ile optimum çözüme yaklaşılır. B, G ve W ile isimlendirdiğimiz üçgenin bu köşe noktaları birbirlerine eşit iken bir noktayı gösterirler ki işte bu nokta aranılan optimumu noktadır. Bu nedenle başlangış vektörlerinin iyi seçimi ile bu metod Gradient Metodundan daha etkili bir arama işlemi haline gelir.

Buna göre, Gradient Algoritması her iterasyonda kısmi türevlerin hesabı gerektiğinden Nelder - Mead Metoduna göre daha çok işlem süresine sahip bir arama işlemidir. Gradient arama işleminde adım uzunlukları arasındaki fark yakınsaklığın durumu göstermesi açısından önemli iken Nelder - Mead Metodunda bu ölçüt her iterasyondaki üçgenin bir öncekine göre daha küçülmesi yani B, G ve W noktalarının birbirine yaklaşması ile belirlenebilir.

Şimdi de Frank-Wolfe ve SUMT Algoritmalarının kendi aralarında karşılaştırmalı yorumlarını yapalım. Bu iki algoritmada kısıtlı çok değişkenli fonksiyonların optimizasyonunu bulurlar. İkisinin ortak özelliği işlem adımları esnasında Gradient arama işlemini kullanmalarıdır. Yani her ikisi de İndirekt (türev kullanan) arama işlemidir.

Frank-Wolfe Algoritması doğrusal kısıtlı problemlerin optimizasyonunu bulurken, SUMT Algoritması hem doğrusal hem doğrusal olmayan kısıtlı problemlerin optimizasyonunu bulur. Frank-Wolfe Algoritmasının genel karakteristiği doğrusal kısıtlı , doğrusal amaç fonksiyonu olan bir doğrusal programlama problemine dönüşür . Frank-Wolfe Algoritmasının diğer önemli bir karakteristiği ise iterasyonların optimum çözüme doğru iki farklı yönden yaklaşmasıdır. Bu nedenle Frank-Wolfe Algoritmasında optimum çözüme yakınsama SUMT Algoritmasına göre daha yavaş bir yakınsamadır.

SUMT Algoritmasının en genel karekteristiği ise ceza fonksiyonlarını kullanarak kısıtlı optimizasyon problemlerini kısıtsız optimizasyon problemlerinin bir dizisi ile çözmesidir. Bu nedenle SUMT Algoritması uygun bölge içinde seçilen bir başlangıç aşikar çözüm ile optimum çözüme çok hızlı bir şekilde yakınsayan ve r = 0.000000001 değeri için;

$P(x,r) = f(x) – rB(x)$ fonksiyonunun amaç değerini bulan bir gradient arama işlemidir.

Bu özellikleri ile SUMT Algoritması gerek doğrusal kısıtlı gerekse doğrusal olmayan kısıtlı fonksiyonların optimizasyonunda çok etkili bir algoritmadır.

4. BÖLÜM

EŞİTSİZLİKLER VE LİNEER PROGRAMLAMA

4.EŞİTSİZLİKLER

4.1. Birinci Dereceden Bir Bilinmeyenli Eşitsizlikler

$a \neq 0$ ve $a, b, x \in R$ olmak üzere f(x)=ax+b ifadesine birinci dereceden fonksiyon ve birinci dereceden iki terimli denir. $ax + b > 0$, $ax+b<0$, $ax + b \geq 0$ ve $ax + b \leq 0$ şeklindeki ifadelere birinci dereceden bir bilinmeyenli eşitsizlikler adı verilir.

ax+b=0 ifadesine birinci dereceden denklem, $x = -\frac{b}{a}$ sayısına da denklemin kökü denir.

$ax + b > 0$, $ax+b<0$, $ax + b \geq 0$ ve $ax + b \leq 0$ eşitsizliklerini ise sonsuz sayı gerçeklediğinden bunların çözüm kümesini bulma işleminde denklemden yararlanılarak önce kökü bulunur. Daha sonra işaret kuralı aşağıdaki tabloda olduğu gibi düzenlenerek çözüm kümesi bulunur. f(x)=ax+b iki terimlisinin işaret kuralı aşağıdaki tabloda verilmiştir.

x	$-\infty$		$+\infty$
f(x) = ax + b	a'nın işaretinin tersi	0	a'nın işaretinin aynı

Örnek:f(x)= -2x+8 fonksiyonunun işaret tablosunu hazırlayarak $-2x+8 \leq 0$ eşitsizliğinin çözüm kümesini bulunuz.

Çözüm:

f(x)=-2x+8 fonksiyonun önce kökünü bulalım. $-2x+8=0 \Rightarrow x = 4$ ve a=4 dür.

x	$-\infty$		4		$+\infty$
-2x+8		+	0	-	

Tablo incelendiğinde;

i) x<4 ise f(x)< 0

ii) x=4 ise f(x)=0

iii) x>4 ise f(x)>0 olur. O halde çözüm kümesi Ç=[4, ∞) dir.

4.2. İkinci Dereceden Bir Bilinmeyenli Eşitsizlikler

a, b, c $\in$ R, a $\neq 0$ ve x $\in$ R olmak üzere; f(x)=ax^2+bx+c ifadesine ikinci derece fonksiyonu ve ikinci dereceden üç terimli denir.$a\,x^2+bx+c>0,\ a\,x^2+bx+c\geq 0,\ a\,x^2+bx+c<0$ ve $a\,x^2+bx+c\leq 0$ şeklindeki ifadelere ikinci dereceden bir bilinmeyenli eşitsizlikler denir.

f(x)=ax^2+bx+c üç terimlisinin işaret kuralı aşağıdaki tablolarda gösterildiği gibi bulunur.

ax^2+bx+c=0 denkleminde;

I) $\Delta > 0$ ise denkleminin birbirinden farklı iki kökü vardır.Bunlar x_1, x_2 olsun.

x	$-\infty$	x_1	x_2	$+\infty$
ax^2+bx+c	a'nın işaretinin aynı	0	a'nın işaretinin tersi	0 a'nın işaretinin aynı

II) $\Delta = 0$ ise denkleminin eşit iki kökü vardır. $x_1 = x_2 = -\frac{b}{2a}$

x	$-\infty$	$x_1 = x_2$	$+\infty$
ax^2+bx+c	a'nın işaretinin aynı	0	a'nın işaretinin aynı

III) $\Delta < 0$ ise ax^2+bx+c=0 denkleminin gerçek kökü yoktur. f(x)=ax^2+bx+c hiçbir zaman sıfır olamayacağından, işareti değişmez,a ile aynı kalır.

x	$-\infty$	$+\infty$
ax^2+bx+c	a'nın işareti ile aynı	

$\Delta < 0$ ve a>0 $\Leftrightarrow$ f(x)>0

$\Delta > 0$ ve a<0 $\Leftrightarrow$ f(x)<0 dır.

Örnek: **a)** x^2+3x-10<0 **b)**$9x^2$+12x+4>0 **c)**x^2-2x<-8 eşitsizliklerinin çözüm kümelerini bulunuz.

Çözüm: **a)** x^2+3x-10<0 eşitsizliğinin çözüm kümesini bulurken önce eşitsizliği denkleme çevrilir.Daha sonra kökler bulunarak işaret tablosu hazırlanır.

$\Delta = 49 > 0$, farklı iki gerçek kökü vardır.

x^2+3x-10=0 denkleminin kökleri (x+5)(x-2)=0 ise x=-5, x=2 dir.

x	$-\infty$	-5	2	$+\infty$
x^2+3x-10	+	0 -	0	+

Ç=$\{x \mid -5 < x < 2,\ x \in R\}$= (-5,2)

b) $9x^2+12x+4>0$, a'da yapılan işlemler izlenerek kökleri araştırılır. $\Delta=0$, birbirine eşit iki kökü vardır.$9x^2+12x+4=0$ denkleminde $\Delta=0$ olduğundan birbirine eşit iki kökü vardır ve denklemin kökleri $x_1 = x_2 = -\frac{b}{2a} = -\frac{2}{3}$ tür.

x	$-\infty$		$-\frac{2}{3}$		$+\infty$
$9x^2+12x+4$		+	0	+	

Ç=R'dir.

c) $x^2-2x<-8 \Rightarrow$ $x^2-2x+8<0$ şekline getirdikten sonra a'da yapılan işlemler tekrarlanarak kökleri araştırılır. $x^2-2x+8=0$ denkleminde $\Delta<0$ olduğundan gerçek kök yoktur.

x	$-\infty$	$+\infty$
x^2-2x+8	+ + + + + + + +	

Ç=Ø dir.

4.3. Eşitsizlikler ve grafikleri

$a.x+b.y+c \geq 0$, $a.x+b.y+c \leq 0$, $a.x+b.y+c > 0$ ve $a.x+b.y+c < 0$ **şeklindeki eşitsizliklere 1.dereceden iki bilinmeyenli eşitsizlikler denir.Bu eşitsizliklerin grafiğini çizmek için ilk olarak;**

$a.x+b.y+c=0$ denkleminden;

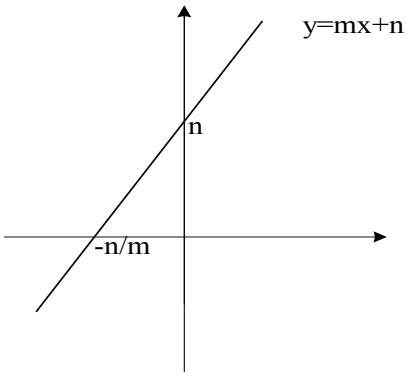
$$y = -\frac{a}{b}x - \frac{c}{b} \quad (a,b,c \in IR\ b \neq 0)$$

şekline getirilerek fonksiyonun grafiği çizilir.Burada $-\frac{a}{b} = m, -\frac{c}{b} = n$ dersek;

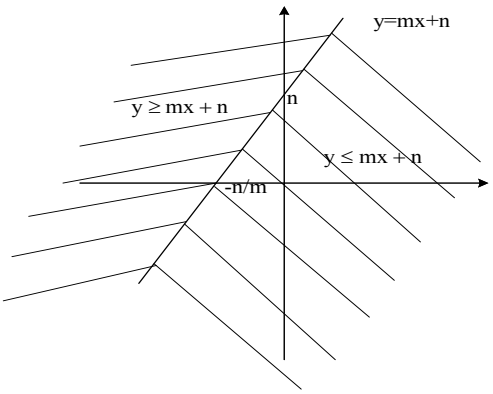
y=mx+n.....(1) elde edilir.Bu fonksiyonun grafiği bir doğru olup, bu doğrunun grafiğini çizmek için iki nokta bulmak yeterli olacaktır.Buradan;y=mx+n için;

x	0	$\frac{-n}{m}$
y	n	0

alınarak, **y=mx+n** ile **y≤mx+n** ve **y≥mx+n** grafikleri aşağıdaki gibi çizilir.



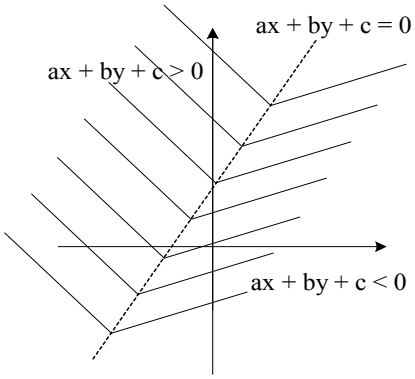
Şekil.21 y==mx+n grafiği



Şekil.22 y≤mx+n ve y≥mx+n grafikleri

y=mx+n eşitliğinde y≤mx+n veya y≥mx+n eşitsizliklerinin grafiği şekil.22' deki gibidir.

y≤mx+n eşitsizliğinin grafiğini çizmek için y=mx+n doğrusunun grafiği ilk olarak çizilir.Daha sonra, bu doğru üzerinde bulunmayan herhangi bir test noktası alınarak eşitsizliği sağlayıp sağlamadığı belirlenir.Eğer bu nokta eşitsizliği sağlıyor ise, bu noktanın alındığı bölge istenen bölge, karşı bölge ise diğer eşitsizliğe ait olur.Bundan başka, ax+by+c>0 ve ax+by+c<0 eşitsizliklerinde yine yukarıdaki yol takip edilir.Fakat, grafik çizilirken ax+by+c=0 doğrusu kesikli çizgi ile gösterilir.(Şekil.23)



Şekil.23 ax+by<0 ve ax+by+c>0 grafikleri

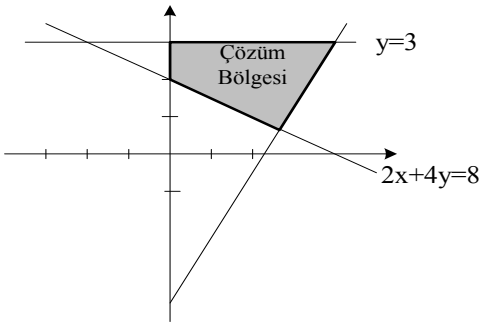
4.4. Eşitsizlik sistemlerinin grafik gösterimi:

Eğer birden fazla eşitsizliğin bir araya gelmesiyle oluşan bir sisteme sahipsek, bu sisteme eşitsizlik sistemi denir.Bu sistemin çözümüne ait grafiği bulmak için ayrı ayrı her bir eşitsizliğin grafiği aynı koordinat düzleminde çizilir ve bu grafiklerin aynı anda sağlandığı bölge (çözüm bölgesi) bulunur.Bunu aşağıdaki gibi basit bir örnek ile gösterebiliriz.

Örnek:

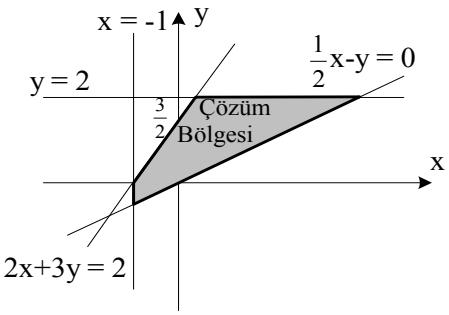
$$\left.\begin{array}{l} 2x + 4y \geq 8 \\ 3x - 2y \leq 6 \\ y \leq 3 \\ x \geq 0 \end{array}\right\}$$ sistemini sağlayan çözüm bölgesini bulalım.

Çözüm:



Örnek : $\left.\begin{array}{l} \frac{1}{2}x - y \geq 0 \\ 2x + 3y \leq 2 \\ y \geq 2 \\ x \geq -1 \end{array}\right\}$ eşitsizlik sisteminin grafik çözümünü bulunuz.

Çözüm:



Not: Her iki örnekte elde edilen çözüm bölgesini bulmak için öncelikle eşitsizlikler eşitlikler haline getirilerek her birinin grafiği çizilir.Daha sonra, eşitsizliğin durumuna göre bir test noktası yardımıyla doğrunun altı veya üstü çözüm bölgesi olarak alınır.Sonuç olarak, bütün eşitsizliklerin aynı anda sağlandığı bölge istenen çözüm bölgesi olarak alınır.

4.5. Lineer Programlama Grafik Çözümü

4.5.1. Genel Bilgi:

Lineer programlama lineer fonksiyonlardan oluşan problemleri çözmek için geliştirilmiş bir metottur. Bu metotla problemlerde istenilen amaca ulaşılmaya çalışılır. Bu amaç maksimum kar, minimum maliyet, maksimum ağırlık vb.gibi olabilir. Bunun anlamı şudur. **x,y** değişkenleri gerçek hayatta bir olayı temsil ettiklerinden hiçbir zaman negatif olmazlar. Bir lineer programlama problemi;

amaç $z=f(x_i)$

Kısıtlar;

$g_i(x) \leq (\geq) b_i$

$x_i \geq 0$ şeklinde tanımlanır. Burada,amaç maksimum ise **$g_i(x) \leq b_i$** ,

amaç minimum ise **$g_i(x) \geq b_i$**

4.5.2. Lineer Programlama İşlem Basamakları

1)

i-) Bilinmeyenler belirlenir ve buna göre değişkenler tanımlanır.

ii-) Kısıtlar lineer eşitsizliklere çevrilir.

iii-) Amaç fonksiyonu oluşturulur.

2) Uygun bölgenin grafiği çizilir.

i-) Eşitsizlikler standart forma çevrilir.

ii-) Her bir eşitsizliğin grafiği çizilir.

iii-) Her eşitsizliğin aynı anda sağlandığı uygun bölge bulunur.

3) Uygun bölgenin köşe noktaları bulunur.

4) Bu köşe noktalarda amaç fonksiyonu belirtilir. En iyi amaca ulaşılan nokta Optimal Çözüm Noktasıdır.

4.5.3. Lineer Programlama Problem Örnekleri

Problem: Aşağıdaki tabloda pirinç ve soya fasulyesinde birim bardakta bulunan kalori, vitamin ve B_2 vitamin miktarları verilmiştir. Buna göre minimum maliyete sahip diyeti elde edebilmek için günde ne kadar pirinç ve soya fasulyesi (bardak cinsinden) üretilmelidir?

	Pirinç	Soya	Günlük Gereksinim
(gr) Protein	15	22.5	90
Kalori	810	270	1620
(mg) B_2 Vitamin	1/9	1/3	1
Maliyet	21	14	

Çözüm: x = günde üretilecek bardak cinsinden pirinç miktarı.

y = günde üretilecek bardak cinsinden soya miktarı.

Modeli kurarsak; Minz=21x+14y

Kısıtlar

x	0	6
y	4	0

$\leftarrow 15x + 22.5y \geq 90$

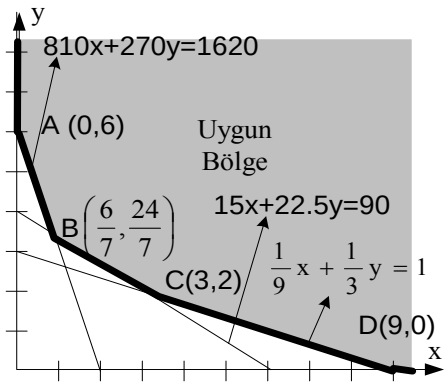
x	0	2
y	6	0

$\leftarrow 810x + 270y \geq 1620$

x	0	9
y	3	0

$\leftarrow \frac{1}{9}x + \frac{1}{3}y \geq 1$

$x \geq 0 \; ; \; y \geq 0$



B için ; 810 x + 270 y = 1620

15 x + 22.5 y = 90

$x = \frac{6}{7}$; $y = \frac{24}{7}$

C için ; 15 x + 22.5 y = 90

$\frac{1}{9}x + \frac{1}{3}y = 1$

x=3 ; y=2

Sonuçların değerlendirilmesi ;

A (0, 6) $\Rightarrow$ z = 21 . 0 + 14 . 6 = 84

B ($\frac{6}{7}, \frac{24}{7}$) $\Rightarrow$ $z = 21 . \frac{6}{7} + 14 . \frac{24}{7} = 66$

C (3 , 2) $\Rightarrow$ z = 21 . 3 + 14 . 2 = 91

D (9, 0) $\Rightarrow$ z = 21 . 9 + 14 . 0 = 189

Yorum : Günde 6/7 bardak pirinç ve 24/7 bardak soya üretirsek 66 T.L.'na malolur ve optimum çözümdür

Problem: Aşağıdaki tabloda verilenlere göre işletmenin maksimum karı elde edebilmesi için günde kaç tane hangi tip bıçak üretilmelidir.

	(A tipi bıçak) **x**	(B tipi bıçak) **y**	Mevcut Kapasiteler
İş gücü	2	6	90
Çelik	20	35	700
Odun	4	3	120
Kar	2	5	

Çözüm:

x = Günde üretilecek A tipi bıçak sayısı.

y = Günde üretilecek B tipi bıçak sayısı.

Model : Maksimum z= 2x + 5y

Kısıtlar :

x	0	45
y	15	0

$\leftarrow 2x + 6y \leq 90$

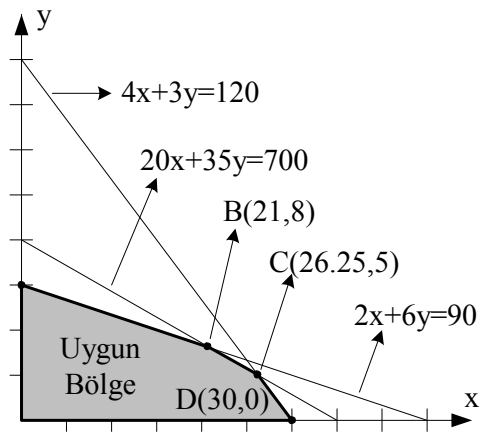
x	0	35
y	20	0

$\leftarrow 20x + 35y \leq 700$

x	0	30
y	40	0

$\leftarrow 4x + 3y \leq 120$

$x \geq 0 \; ; \; y \geq 0$



B noktası için ;

20x+35y=700

-10/ 2x+6y=90

x=21 ve y=8

C noktası için ;

20x+35y=700

-5/ 4x+3y=120

x=26.25 ve y=5

B (21, 8) noktası için ;

z = 2 . 21 + 5 . 8 = 82

C (26.25, 5) noktası için ;

z = 2 . (26.5) + 5 . 5 = 77.5

Optimum çözüm B (21, 8)'dir.

Problem: Aşağıdaki tabloya göre günde üretilecek pamuk ve floş ipliği miktarını bulunuz.

	(Pamuk) x	(Floş) y	Mevcut Kapasiteler
Hallaç	1	4	8
Cer-fitil	1	1	5
Taraklama	2	1	7
Maliyet	20	30	

Çözüm:

x= Günde üretilecek Pamuk İpliği miktarı.

y= Günde üretilecek Floş İpliği miktarı.

Model 1 .

Min z=20x+30y

Kısıtlar

x	0	8
y	2	0

$\Rightarrow x + 4y \geq 8$

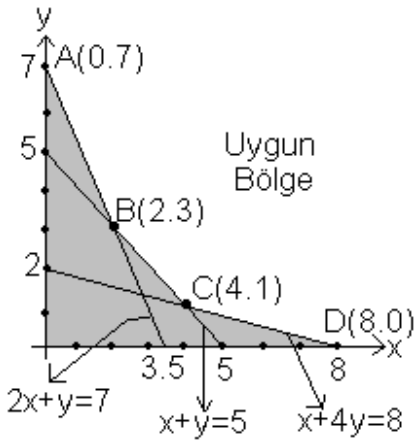
x	0	5
y	5	0

$\Rightarrow x + y \geq 5$

x	0	7
y	3.5	0

$\Rightarrow 2x + y \geq 7$

$x \geq 0 \ ; \ y \geq 0$



B Noktası

x+y =5

2x+y =7

x=2 , y=3

C Noktası

x+4y =8

x+y =5

x=4 , y=1

B için z=20.2+30.3=130 , C için z=20.4+30.1=110 Optimum çözüm.

Problem:

Yandaki tabloda verilenlere göre işletmenin maksimum karı elde edebilmesi için günde hangi tip malzemeden ne kadar üretmesi gerekir.		(A Tipi) x	(B Tipi) y	Mevcut Kapasiteler
	1.Atölye	1	4	340
	2.Atölye	3	4	380
	3.Atölye	5	2	330
	Kar	80	120	

Çözüm:

x = A tipi malzemeden günde üretilecek miktar.

y = B tipi malzemeden günde üretilecek miktar.

Model :

Maksimum z = 80x + 120y

Kısıtlar

x	0	340
y	85	0

$\Rightarrow x + 4y \le 340$

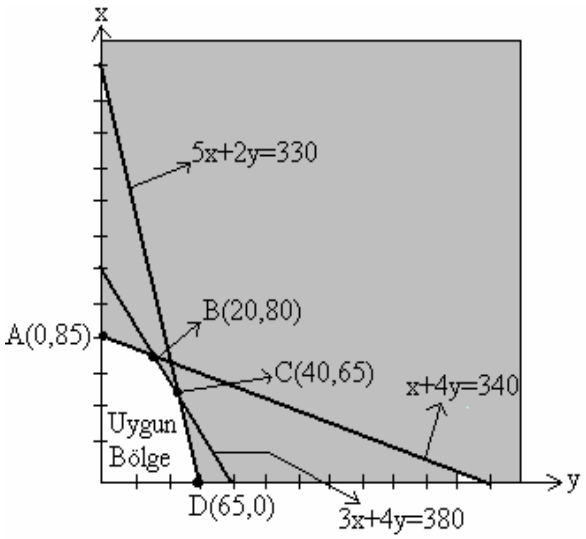
x	0	$380/3$
y	95	0

$\Rightarrow 3x + 4y \le 380$

x	0	65
y	165	0

$\Rightarrow 5x + 2y \le 330$

$x \ge 0 \ ; \ y \ge 0$



B için 3x+4y=380 x+4y=340 x=20 , y=80

C için 5x+2y=380 x+4y=340 x=40 , y=65

B(20,80) ➔ Z=80.20+120.80=11.200

C(40,65) ➔ Z=80.40+120.65=11.000

O halde B(20.80) noktası optimum çözüm noktasıdır.

Problem: Bir işletmede radyo ve TV üretilmektedir .Bu üretim tasarım montaj ve test atölyelerinde gerçekleştirilmektedir. Birim radyo üretimi için bu atölyede sırasıyla 6saat,4saat,3saat iş gücü, gerekli iken birim TV üretimi için bu atölyede sırasıyla 3saat,4saat, ve saat iş gücü gerekmektedir .Atölyelerin günlük maksimum kullanım süreleri sırasıyla 90,72 ve 96 saattir.Birim karlar sırasıyla 10 ve 20 dolardır.Buna göre işletmenin maksimum karı elde edebilmesi için günde kaç tane radyo ve TV üretilmesi gerekir.

Çözüm: x: günde üretilecek radyo sayısı

y:günde üretilecek TV sayısı

	x	y	Mevcut kapasite
Tasarım	6	3	90
Montaj	4	4	72
Test	3	6	96
Kar	10	20	

Maksimum z = 10x + 20y

x	0	15
y	30	0

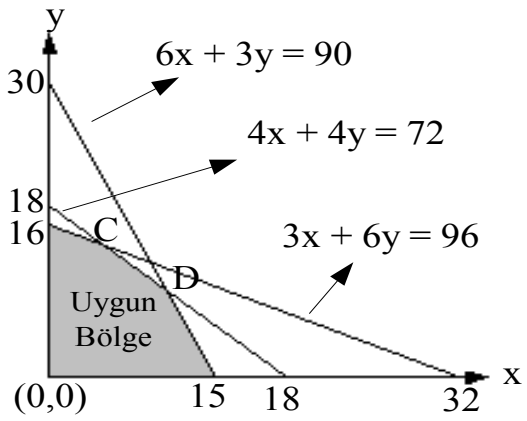
← 6x + 3y = 90

x	0	18
y	18	0

← 4x + 4y = 72

x	0	32
y	16	0

← 3x + 6y = 96



$x \geq 0, y \geq 0$

4x + 4y = 72 ise x = 4 C (4,14) ise z =10.4 + 20.14 = 320

3x + 6y = 96 y = 14

4x + 4y = 72 ise x = 12 D (12,6) ise z =10.12 +20.6 = 240

6x + 3y = 90 y = 6

İşletme günde 4 tane radyo , 14 tane TV üretmesi gerekir.

Problem:

Bir işletmede pamuk ve floş ipliği üretilmektedir. Bu üretim hallaç, cer-fitil ve taraklama dairesinde gerçekleşmektedir. Atölyelerin günlük kullanım süreleri mil olup 16,72,56 saattir.Birim pamuk üretimi için bu atölyede sırasıyla 2,12,4 saat iş gücü gerekirken birim floş üretimi için 2,6,14 saat iş gücü gereklidir.Birim maliyetler sırasıyla 5 ve 6 dolardır. Buna göre, işletmenin minimum maliyeti elde edebilmesi için günlük ne kadar pamuk ve floş ipliği üretmesi gerekir.

Çözüm:

u : günde üretilecek pamuk ipliği miktarı

v: günde üretilecek floş ipliği miktarı

	u	v	mevcut kapasite
Hallaç	2	2	16
Cer-fitil	12	6	72
Taraklama	4	14	56
Maliyet	5	6	

Min z = 5u + 6v

Kısıtlar

u	0	8
v	8	0

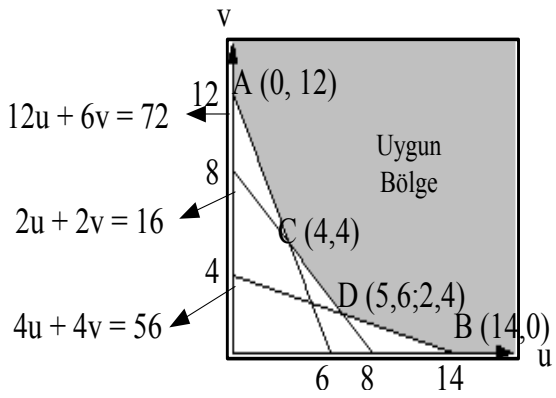
$2u + 2v = 16$

u	0	6
v	12	0

$12u + 16v = 72$

u	0	14
v	4	0

$4u + 4v = 56$



$u \geq 0, v \geq 0$

4u + 14v = 56 ise u = 5,6 D (5.6,2.4) ise z = 5.5,6 + 6.2,4 = 42,4

2u + 2v = 16 v = 2,4

12u + 6v = 72 ise u = 4 C (4,4) ise z = 5.4 + 6.4 = 44

2u + 2v = 16 v = 4

günde 5,6 birim pamuk 2,4 birim floş ipliği üretilmesi gerekir.

4.6.Simpleks yöntemiyle Lineer Programlama Problemi Çözümü

Bu yöntemi aşağıdaki örnek problem üzerinde inceleyelim.

Maxz=80x+70y

Kısıtlar

6x+3y ≤ 96

x+y ≤ 18

2x+6y ≤ 72

x $\geq 0, y \geq 0$

Adım 1: Aylak değişkenler girilir ve lineer denklem sistemine göre değişkenler kurulur

Adım 2: Buna göre bunun için başlangıç simpleks tablosu oluşturulur

Adım 3: Amaç fonksiyonun satırında en büyük negatif değerli sayının olduğu sütundaki pivot eleman belirlenir Bu eleman en sağ sütundaki katsayılar sabitler pivotlar sütundaki kat sayılara bölünerek en küçük orana sahip değişken temel'e girer öbürü temelden çıkar(temel çözüm)

Adım 4: Yukarıda ki adımlar amaç fonksiyonu satırında negatif eleman kalmayıncaya kadar tekrarlanır.

Maxz=80x+70y amaç fonksiyonu M-80x-70y=0 şekline çevrilir.

Kısıtlar

6x+3y+u =96

x+y +v =18

2x+6y +w =72

-80x-70y +M =0

Bundan sonra adım 2'ye geçilir yani başlangıç simplex tablosu hazırlanır.

	x	**y**	**u**	**v**	**w**	**M**	
u	6	3	1	0	0	0	**96**
v	1	1	0	1	0	0	**18**
w	2	6	0	0	1	0	**72**
M	-80	-70	0	0	0	1	**0**

Boyalı sütun amaç fonksiyonu sütunudur.

Tabloya göre u=96, v=18 ,x=0 ,y=0 olup çözüm değildir çözüm için amaç fonksiyonu satırında negatif eleman kalmamalıdır.

3.aşama:amaç fonksiyonu belirlendi ve bu satırdaki en küçük negatif eleman (-80) yok edilmeye çalışılıyor.

6.....96 96/6=**16** en küçük eleman olduğundan bunu

1.....18 belirlendi ve 18/1=18

2.....72 72/2=36

96/6=**16** en küçük eleman olduğundan bunu sağlayan sayı 6 olduğuna göre bu sayı işaretlenir ve 6'nın bulunduğu satırı 6'ya böleriz

	x	**y**	**u**	**v**	**w**	**M**		
u	1	½	1/6	0	0	0	16	D1
v	1	1	0	1	0	0	18	D2
w	2	6	0	0	1	0	72	D3
M	-80	-70	0	0	0	1	0	D4

-D1+D2 D2

-2D1+D3 D3 işlemleri yapılarak yeni bir matris oluşturulur.

-80D1+D4 D4

Amaç x sütununu , u sütununa dönüştürüp x'i temele almaktır.

	x	**y**	**u**	**v**	**w**	**M**	
x	1	½	1/6	0	0	0	**16**
v	0	½	-1/6	1	0	0	**2**
w	0	5	-1/3	0	1	0	**40**
M	0	-30	40/3	0	0	1	**1280**

x=16 , v=2 ,w=40 ,m=1280, y=0 olup çözüm değildir .O halde, işleme devam edilerek –30 belirlenir ve $\frac{16}{1/2}=32, \frac{2}{1/2}=4, \frac{40}{5}=8$ oranlarından en küçüğü sağlayan ½ sayısı pivot elemandır.

	x	y	u	v	w	M	
x	1	½	1/6	0	0	0	**16**
v	0	½	-1/6	1	0	0	**2**
w	0	5	-1/3	0	1	0	**40**
M	0	-30	40/3	0	0	1	**1280**

	x	y	u	v	w	M	
x	1	0	1/3	-1	0	0	**14**
y	0	1	-1/3	2	0	0	**4**
w	0	0	4/3	-10	1	0	**20**
M	0	0	10/3	60	0	1	**1400**

(-1/2)D2+D1 → D1

(-5)D2+D3 → D3

30D2+D4 → D4

O halde; x=14 , y= 4 ,w =20 , m=1400 çözümdür yani;maksimumz=80.14+70.4=1400'dür.

4.6.1.Marjinal Analiz

	x	y	u	v	w	M		
x	1	0	1/3	-1	0	0	14	**+3.(1/3)=15**
y	0	1	-1/3	2	0	0	4	**+3.(-1/3)=3**
w	0	0	4/3	-10	1	0	20	**+3.(4/3)=24**
M	0	0	10/3	60	0	1	1400	**+3.(10/3)=1410**

u tasarım , v montaj , w test atölyesi olsunlar .Tasarım sütununu dikkate alalım burada ilave bir iş gücü üretim seviyesi ve karı ne kadar değiştirilebilir ? Bunun için tasarım atölyesinde 3 saatlik ilave iş gücü sonuç tabloyu şekilde eklenir

Sonuç olarak u=10/3 , v=60 ,w=0 değerlerine marjinal değerler denir.

Örnek.4.7 :maxz=3x+5y

Kısıtlar

$3x+6y \leq 90$ Tasarım

$7x+5y \leq 138$ Montaj

$4x+3y \leq 120$ Test

$x \geq 0, y \geq 0$

montaj atölyesine 18 saat ilave edildiğinde kar ne kadar artar?Montaj atölyesine ilave edilecek saati belirleyen analizi yapınız.

Çözüm.4.7:

	x	y	u	v	w	M	
u	3	6	1	0	0	0	90
v	7	5	0	1	0	0	138
w	4	3	0	0	1	0	120
M	-3	-5	0	0	0	1	0

	x	y	u	v	w	M		
y	0	1	7/27	-1/9	0	0	8	**+18(-1/9)=6**
x	1	0	-5/27	2/9	0	0	14	**+18(2/9)=18**
w	0	0	-1/27	-5/9	1	0	40	**+18(-5/9)=30**
M	0	0	20/27	1/9	0	0	82	**+18(1/9)=84**

(2 lira artı) u tasarım , v montaj , w test atelyesi olsunlar.

giriş kısıtlama

8+h(-1/9) $h \le 72$

14+h(2/9) $h \ge 63$ $-63 \le h \le 72$

40+h(-5/9) $h \le 72$

Örnek.4.8: minz =375-2x-3y minz= - max(-z)

Kısıtlar

$x \le 30$

$y \le 25$

$x+y \ge 15 (ise - x - y \le -15)$

$x+y \le 45$

$x \ge 0, y \ge 0$

Çözüm.4.8: x+ t =30

y + u =25

-x+-y +v =-15

x+y +w =45

-2x-3y +M = -375

	x	y	t	u	v	w	M	
t	1	0	1	0	0	0	0	30
u	0	1	0	1	0	0	0	25
v	-1	-1	0	0	1	0	0	-15
w	1	1	0	0	0	1	0	45
M	-2	-3	0	0	0	0	1	-375

		x	y	t	u	v	w	M	
D1	**t**	0	-1	1	0	1	0	0	15
D2	**u**	0	1	0	1	0	0	0	25
D3	**v**	1	1	0	0	-1	0	0	15
D4	**w**	0	0	0	0	1	1	0	30
D5	**M**	0	-1	0	0	-2	0	1	-375

	x	y	t	u	v	w	M	
v	0	-1	1	0	1	0	0	15
u	0	1	0	1	0	0	0	25
x	1	0	1	0	0	0	0	30
w	0	1	-1	0	0	1	0	15
M	0	-3	2	0	0	0	1	-315

	x	y	t	u	v	w	M	
v	0	0	0	0	1	1	0	30
u	0	0	1	1	0	-1	0	10
x	1	0	1	0	0	0	0	30
y	0	1	-1	0	0	1	0	15
M	0	0	-1	0	0	3	1	-270

Maksimum problemi olsaydı 3'ü seçecektik burada ise –2'yi yok etmeye çalışacağız . Karşılıklı elemanları oranlıyoruz

30/1; -15/-1 ; 45/1 oranlarından en küçük oranı alıyoruz (-15)/(-1)'dır .(-) pivot elemandır. Bulunduğu satırın tüm elemanlarını (-) ile çarptık ikinci tabloyu yazdık .Daha sonra aşağıdaki satır işlemlerini yaptık;

2D3+D5 → D5

D3+D1 → D1

D3+D4 → D4

İkinci tabloda –2'yi yok etmeye çalışacağız.Karşılıklı elemanları oranlıyoruz

15/1 ; 15/-1 ; 30/1 en küçük oranı alıyoruz.

	x	y	t	u	v	w	M	
v	0	0	0	0	1	1	0	**30**
t	0	0	1	1	0	-1	0	**10**
x	1	0	0	-1	0	0	0	**20**
y	0	1	0	1	0	1	0	**25**
M	0	0	0	1	0	2	1	**-260**

Min 375-2x-3y ise 2x+3y-375=M$\Rightarrow$ x=20 , y=25 ise 2.20+3.25-375=-260 TL

Negatif çıkmasının sebebi minimum problemi olmasıdır.

4.7.DUALİTE

max CX

Kısıtlar

$AX \le B$

$X \ge 0$ lineer programlama probleminin duali;

min $B^T U$

kısıtlar

$A^T U \ge CT$

$U \ge 0$ şeklinde tanımlanır.Bir önceki örnek problemimizi buna göre uyarlarsak;

Maxz=80x+70y $\Big\}$ max$[80 \;\; 70]\begin{bmatrix} x \\ y \end{bmatrix}$

Kısıtlar **Kısıtlar**

$6x+3y \le 96$ $\quad \begin{bmatrix} 6 & 3 \\ 1 & 1 \\ 2 & 6 \end{bmatrix} . \begin{bmatrix} x \\ y \end{bmatrix} \le \begin{bmatrix} 96 \\ 18 \\ 72 \end{bmatrix} ve \begin{bmatrix} x \\ y \end{bmatrix} \ge \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$x+y \le 18$ $\qquad$ 3x2 $\quad$ 2x1 $\quad$ 3x1

$2x+6y \le 72$

$x \ge 0, y \ge 0$

Dual problem:Minz=96u+18v+72w

Kısıtlar

$6u+v+2w \ge 80$

$3u+v+6w \ge 70$

$u \ge 0, v \ge 0, w \ge 0$

Örnek: min18x+20y+2z

kısıtlar

$3x-5y-2z \le 4$

$6x \quad -8z \ge 9$

$x \ge 0, y \ge 0, z \ge 0$

yeni kısıtlar

$-3x+5y+2z \ge -4$

$6x \ -8z \ge 9$

$x \ge 0, y \ge 0, z \ge 0$

maxz=-4u+9v (simpleks çözüm)

Kısıtlar

$-3u+6v \le 18$ → $-3u+6v+x = 18$

$5u \le 20$ → $5u + y = 20$

$2u-8v \le 2$ → $2u-8v + z = 2$

$u \ge 0, v \ge 0$

$$\begin{bmatrix} -3 & 5 & 2 \\ 6 & 0 & -8 \end{bmatrix} \quad , \quad \begin{bmatrix} -3 & 6 \\ 5 & 0 \\ 2 & -8 \end{bmatrix}$$

2x3 3x2

M=-4u+9v ise 4u-9v+M=0

	u	**v**	**x**	**y**	**z**	**M**		
x	-3	6	1	0	0	0	18	D1
y	5	0	0	1	0	0	20	D2
z	2	-8	0	0	1	0	2	D3
M	4	-9	0	0	0	1	0	Amaç fonksiyonundaki en küçük sayı,D4

18/6 ,20/0 ,2/-8 , buradan 18/6 seçilir.6 pivot elemandır

	u	v	x	y	z	M		
v	-1/2	1	1/6	0	0	0	3	D1
y	5	0	0	1	0	0	20	D2
z	-2	0	4/3	0	1	0	26	D3
M	-1/2	0	3/2	0	0	1	27	D4

2D2+D3⇒(1/2)D2+D1⇒ (1/2)D2+D4⇒işlemleri gerçekleştirilir.

$\frac{3}{-1/2}, \frac{20}{5}, \frac{26}{-2},$ buradan 20/5 seçilir.5 pivot elemandır.

	u	v	x	y	z	M	
v	0	1	1/6	1/10	0	0	5
u	1	0	0	1/5	0	0	4
z	0	0	4/3	2/3	1	0	34
M	0	0	3/2	1/10	0	1	29

v=5 , u =4 olduğundan maxz = -4.4 + 9.5 =29'dır optimum çözüm sağlanmıştır.

M=18x+20y+2z , x=3/2 , y= 1/10 , z=0 ise M=18.3/2+20.1/10+2.0=29

Örnek.4.9:

minz=2x-3y+6z

kısıtlar

$3x-4y-6z \leq 2$

$2x+y+2z \geq 11$

$x+3y-2z \leq 5$

$x \geq 0, y \geq 0, z \geq 0$ LP problemini çözünüz

Çözüm.4.9 :

minz= 2x-3y+6z (standart forma çevirelim)

kısıtlar

$-3x+4y+6z \geq -2$

$2x+y+2z \geq 11$

$-x-3y+2z \geq -5$

$x \geq 0, y \geq 0, z \geq 0$ (dualini alalım)

maxz=-2u+11v-5w

kısıtlar

$-3u+2v-w \le 2$

$4u+v-3w \le -3$

$6u+2v+2w \le 6$

$u \ge 0, v \ge 0, w \ge 0$

-3u+2v-w+x =2
4u+v-3w+y = -3
6u+2v+2w +z = 6

2u-11v+5w +M= 0

	u	**v**	**w**	**x**	**y**	**z**	**M**	
x	-3	**2**	-1	1	0	0	0	2
y	4	1	-3	0	1	0	0	-3
z	6	2	2	0	0	1	0	6
M	2	-11	5	0	0	0	1	0

6/2=3 ,-3/1=-3 ,2/2=1 (en küçük oran alınır)pivot eleman ikinci satırın tüm elemanlarını 2'ye bölelim satır işlemleri ile pivot elemanın bulunduğu sütundaki elemanları 0 yapalım son satırdan(amaç satırında) negatif eleman kalmayana kadar bu işlemlere devam edelim.

	u	**v**	**w**	**x**	**y**	**z**	**M**	
v	-3/2	1	-1/2	½	0	0	0	1
y	11/2	0	-5/2	-1/2	1	0	0	-4
z	9	0	3	-1	0	1	0	4
M	**-29/2**	0	-1/2	11/2	0	0	1	11

	u	**v**	**w**	**x**	**y**	**z**	**M**	
v	0	1	0	1/3	0	1/6	0	5/3
y	0	0	-13/3	-1/9	1	-11/18	0	-14/9
u	1	0	1/3	-1/9	0	1/9	0	4/9
M	0	0	13/3	35/9	0	29/18	1	157/9

u=4/9 , v=5/3 , w=0$\Rightarrow$-2u+11v-5w=-2 4/9 +11.5/3-5.0=157/9

x=35/9 ,y=0 , z=29/18$\Rightarrow$2x-3y+6z=2.35/9-3.0+6.29/18=157/9

ilk atölyede 3 saatlik bir artış yapılırsa kar ne olur?

5/3+3(1/3)=8/3$\Rightarrow$-14/9+3(-1/9)=-17/9$\Rightarrow$4/9+3(-1/9)=1/9$\Rightarrow$157/9+3(35/9)=262/9

5.BÖLÜM

5.Ulaştırma Problemleri

5.1.Ulaştırma Problemlerine Giriş

Bu bölümde, doğeusal programlama problemlerinin özel bir türü yani belirli sayıda kaynakta üretilen malların, belirli amaçlere minimum maliyette gönderilmesi ile ilgili olan ulaştırma modellerine değinilecektir. Ulaştırma probleminde amaç kaynaklardan hedeflere, yani üretim merkezlerinden dağıtım merkezlerine mallar dağıtılırken, bu dağıtım işlemini mnimum maliyette gerçekleştirmektir. Ulaştırma modelinde kısıtlayıcı koşullar, istem ve sunum miktarına bağlı olarak daha çok eşitlik şeklindedir.

Ulaştırma modeli şeklinde kurulan bir problem simpleks yöntem ile çözülebilir. Fakat ulaştırma problemleri kendine özgü teknikleri ile, yani ulaşırma algoritması, atama ve aktarma modelleri gibi tekniklerle daha az zaman ve daha az hesaplamayla çözmek olanaklıdır. Ulaştırma modelini bugünküne benzer fakat daha basit yapıda ilk kez 1941 yılında Hitchock petrol endüstrisine uygulamıştır. Sonraları, Koopmans, Dantzig, Cooper ve charnes'in geliştirdikleri ve uygulamada geçerli olan teknik 1960'larda yaygınca kullanılmaya başlamıştır.Ulaştırma problemleri aşağıdaki alanlarda sıkça kullanılmaktadır:

1- Üretim ve tüketim merkezleri arasında optimal mal dağıtımının belirlenmesinde,
2- İşlerin makinalara dağıtımında,
3- Üretim planlamasında,
4- Çeşitli şebeke ağı (network) problemlerinde,
5- İşletmelerin (fabrikaların) kuruluş yeri seçiminde gibi..

5.2.Örneklerle Ulaştırma Problemlerinin incelenmesi:

Problem.5.1: Bir işletmeden aşağıdaki veriler alınmıştır

Üretim merkezleri	Üretim miktarları	Tüketim merkezleri	Tüketim miktarı
F1	200	D1	250
F2	400	D2	200
F3	250	D3	350

Tüketim merkezi / Üretim merkezi	D1	D2	D3
F1	10	6	5
F2	7	8	8
F3	6	9	12

Bu tablo iki merkez arasındaki birim taşıma maliyetidir .Üretim yerlerinden tüketim yerlerine gönderilecek malları işleminin toplam maliyetinin en az olması isteniyor

Çözüm.5.1: Ulaştırma tablosunun düzenlenmesi

	D1	D2	D3	Arz
F1	10 x_{11}	6 x_{12}	5 x_{13}	200
F2	7 x_{21}	8 x_{22}	8 x_{23}	400
F3	6 x_{31}	9 x_{32}	12 x_{33}	250
Talep	250	200	350	

$$\sum_{i=1}^{3} a_i > \sum_{j=1}^{3} b_j$$

Buna göre amaç fonksiyonu;

$Z_{min}=10.x_{11}+6.x_{12}+5.x_{13}+...+12.x_{33}$

$\mathbf{x_{ij} \geq 0}$ 'dır

Kısıtlar

$x_{11}+x_{12}+x_{13} \leq 200$
$x_{21}+x_{22}+x_{23} \leq 400$ } arz kısıtlayıcıları
$x_{31}+x_{32}+x_{33} \leq 250$

$x_{11}+x_{21}+x_{31} \geq 250$
$x_{12}+x_{22}+x_{32} \geq 200$ } talep kısıtlayıcıları
$x_{13}+x_{23}+x_{33} \geq 350$

Kuzey-batı köşesi yöntemi : Hantal bir çözüm yöntemidir,daha iyisi vardır.

	D1	D2	D3	D4	Arz
F1	10 200	6 x_{12}	5 x_{13}	0	200
F2	7 50	8 200	8 150	0	400
F3	6 x_{31}	9 x_{32}	12 200	0 50	250
Talep	250	200	350	50	850 850

Arz=Talep olması için D4'ü ekledik

Z_{min}=200.10+7.50+8.200+8.150+12.200+0.50

Z_{min}=7550 TL.

m+n-1=6(m=4,n=3)

En küçük maliyetli hücreler yöntemi:

	D1	D2	D3	D4	Arz
F1	10	6	5 200	0	200
F2	7 50	8 200	8 150	0	400
F3	6 200	9	12	0 50	250
Talep	250	200	350	50	

Z_{min}=7.50+8.200+8.150+5.200+6.200+0.50

Z_{min}=7350TL.

VAM(vogel) yöntemi:

	D1	D2	D3	D4	Arz (sunum)
F1	45	17	21	30	15
F2	14	18	19	31	13
Talep (istem)	9	6	7	9	

Arabaların kiralandığı merkez ile kiralayan merkezler arasındaki birim taşıma maliyetleri tabloda verilmiştir.Buna göre arabaların dağıtım programlarını ve toplam en küçük taşıma maliyetini vagel yöntemine göre düzenleyiniz.

Öncelikle arz talep dengelenir.Bunun için kukla fabrika kurarız F3 gibi .Daha sonra her satır ve sütün için en küçük iki sayının farkını hesaplayıp ilgili satır yada sütunun sonuna yazarız. İşlemlere en büyük farktan başlarız m+n-1=4+3-1=6 tane çarpım toplanır.

	D1	D2	D3	D4	Arz
F1	45	17 6	21 3	30 6	15
F2	14 9	18	19 4	31	13
F3	0	0	0	0 3	3
talep	9	6 7		9	31 31

4 4 9

4 4 1 12

14 17 19 30

31 1 2 1

1 2 1

2 1

Minz= 0x3+14x9+17x6+3x21+19x4+30x6=547 TL.'dır .

6.BÖLÜM

6.Atama Problemleri

6.1 Atama Problemlerine Giriş:

Atama problemleri en çok işçilerin işlere ve işlerin makinalara programlandırılmasında kullanılır.Atama modeli kaynakların en etkin kullanımını amaçladığından işlerin min. Zamanda veya min. maliyette gerçekleşmesi istenir .Atama problemleri ulaştırma problemlerine dönüştürülebilir. Burada tüm arz ve talepler toplamı 1'e eşit olmaktadır.

6.2.Atama Problemlerinin Çözüm adımları:

Adım-1) Maliyet matrisinin her sırasında yer alan en küçük değerli eleman belirlenir, sonra yeni bir maliyet matrisi oluşturmak için aynı sıradaki tüm elemanlardan çıkarılır.

Adım-2) Adım-1'de elde edilen maliyet matrisinin her sütunundaki en küçük değerli eleman bulunur, sonra bu elemanlar ilgili olduğu sütundaki tüm elemanlardan çıkarılır.

Adım-3) Elde edilen yeni matristeki "0" değerli elemanlara kaynkalar veya işçiler atanır. Bir işçinin sadece bir işe atanması yapılmış ise bu durum uygun atamanın olduğunu gösterir.Hangi işçinin hangi işe atandığını belirlemek için "0" değerli elemanlar daire içine alınır. Eğer uygun atama yoksa Adım-4'e geçilir. En uygun atamalar daire içine alınan "0"'lara karşılıktır.

Adım-4) Matriste yer alan tüm "0" değerli elemanlardan geçen en az sayıda çizgiler çizilir. Çizilen çizgilerin sayısı sıra veya sütun sayısından az olacaktır. Üzerinden çizgi geçmeyen en küçük eleman seçilir sonra bu eleman, üzerinden çizgi geçmeyen en küçük eleman seçilir sonra bu eleman, üzerinden çizgi geçmeyen tüm elemanlardan çıkarılır ve iki çizginin kesiştiği yerdeki elemanlara eklenir. Üzerinden çizgi geçen öteki elemanlar değişmeden kalır. Bütün işlemlerden sonra Adım-3'deki işlemlere başvurulur.

6.3. Örneklerle Atama Problemlerinin İncelenmesi:

Problem:

Aşağıdaki tabloda saat olarak her makinenin ilgili iş bitirme süresi verilmektedir .Buna göre, minimum zamanda işlerin makinelara atanması problemini çözünüz.

Makineler

İşler	**A**	**B**	**C**	**D**
1	10	14	15	13
2	12	13	16	12
3	8	12	12	11
4	13	16	18	13

Çözüm:

0	4	5	3	1	0	3	1	3
0	1	4	0	2	0	0	0	0
0	4	4	3	3	0	3	0	3
0	3	5	0	4	0	2	1	0

Her satırdaki en küçük eleman belirlenir. İki taneyse herhangi biri alınır sonra bütün elemanlar bu elemandan çıkartılarak yeni bir matris oluşturulur. Daha sonra I. adımda elde edilen maliyet matrisinin her sütunundaki en küçük eleman bulunur ve bu tüm elemanlardan çıkartılır.Elde edilen yeni matristeki "0" değerli elemanlara işçi, iş veya makine atanır .
Hangi işçinin (makinenın) hangi işe atandığı belirlemek için "0" değerli elemanlar daire içine alınır Atama uygun değilse adım 4'e geçilir.

Matriste yer alan tüm "0" değerli elemanlardan geçen en az sayıda doğru çizilir. Çizilen doğruların sayısı satır veya sütun sayısından az olacaktır. Üzerinden doğru geçmeyen en küçük eleman bulunacak ve üzerinden doğru geçmeyen bütün elemanlardan çıkarılacak ve iki doğrunun kesiştiği yerdeki elemanlara eklenecektir .Sonuç vermezse adım 3'e dönülür
1.iş A makinesine, 2.iş B makinesine, 3.iş C makinesine , 4. iş D makinesine atanır
minz=10+13+12+13=48

Problem:Aşağıda verilen değerlere göre işçilerin işlere atamalarını yaparak toplam maliyeti minimum yapınız.

İşler

İşçiler		**1**	**2**	**3**	**4**
	1	12	14	10	9
	2	11	8	12	7
	3	8	6	9	5
	4	6	4	7	6

Çözüm:

	1	**2**	**3**	**4**
1	1	5	0	1
2	1	0	3	0
3	0	0	2	0
4	0	0	2	3

	1	**2**	**3**	**4**
1	1	5	0	0
2	2	1	4	0
3	1	1	3	0
4	0	0	2	2

Opimal çözüm = 1.işçi 3.işe,2.işçi 4.işe,3.işçi 1.işe, 4.işçi 2. işe atanır minz=10+7+8+4=29.

7.BÖLÜM

7.Gezgin Satıcı Problemi

7.1 Gezgin Satıcı Problemlemine Giriş:

Gezgin satıcı problemi başladığı noktaya tekrar dönmek koşuluyla (n-1) sayıda yerleşim yerine uğrayan satıcı ile ilgilidir. Burada amaç sayısı "n" tane olan yerleşim yerinden bir satıcının (n-1) tane kente en kısa sürede uğrayarak başladığı kente dönmesini sağlayacak bir gezi planı hazırlamaktır.

Bu yöntem teslimat, teftiş gibi işlemlerde ve T:v: röle istasyonlarının yerinin tespitinde veya okul öğrencilerinin minibüslerle toplanması gib günlük işlerde sıklıkla kullanılır.

7.2.Gezgin Satıcı Problemi İşlem Adımları:

Gezgin satıcı problemi için dört adımda işlem yapılır Bunlar ;

1.-Adım) Verilen problemin maliyet matrisinde boş hücreler varsa bu hücrelere en büyük sayılar yazılır matriste yer alan en küçük eleman daire içine alınır gezi planının halkasına eklenir

2.-Adım) Adım I'de daire içine alınan elemanın bulunduğu satır ve sütunundaki tüm diğer elemanların yerine çok büyük değerli sayılar yerleştirilerek yeni bir matris elde edilir

3.-Adım) Adım II'de ulaşılan maliyet matrisinde daire içine alınmayan en küçük eleman işaretlenir ve sonra henüz tamamlanmamış gezi planında karşılık olduğu halkaya deneme yönünde eklenir. Sonuçlanan gezi uygun değilse işaretlenen maliyet daire içine alınarak Adım IV' e geçilir.

4.Adım) Eğer sonuçlanan gezi planı uygun değilse en son halka plandan çıkartılır ve onun maliyeti yerine daha önce verilen büyük maliyet alınarak Adım III'e dönülür

5.Adım) Gezi planı tamamlanmış ve optimale yakın ise bu plan kabul edilir aksi halde adım 2'ye dönülür. Boş bırakılan herhangi bir yerleşim yeri Adım II'deki işlemlerine geçilir. Boş bırakılan herhangi bir yerleşim yeri Adım II'deki işlemler ile yeniden geriye bırakılmaz ve herhangi bir yerleşim yeri gezi planına girdi ise tekrar plana girmez.

7.3. Gezgin Satıcı Problemlerinin Örneklerle İncelenmesi:

Problem:

Aşağıda gezgin satıcı problemi için verilen maliyet matrisine optimale yakın çözüm varsa bulunuz.

	A	B	C	D	E
A	-	35	80	105	165
B	35	-	45	20	80
C	80	45	-	30	75
D	105	20	30	-	60
E	165	80	75	60	-

Çözüm:

Önce boş bırakılan gözelere yeni c_{11},c_{22},c_{33},c_{44} ve c_{55} yerine 2000 gibi çok büyük değerli maliyet elemanı yerleştirilerek aşağıdaki tablo elde edilir.

	A	B	C	D	E
A	2000	35	80	105	165
B	35	2000	45	20	80
C	80	45	2000	30	75
D	105	20	30	2000	60
E	165	80	75	60	2000

Yukarıdaki tabloda en küçük değer c_{24} veya c_{42}'dir. c_{24} elemanını seçtiğimizi düşünelim ve işaretleyerek bunu, gezi planının bir halkası yani 2 – 4 olarak kabul edelim. Sonra c_{24} elemanının dışında kalan ikinci sıra ve dördüncü sütundaki elemanlar ile aynı değerdeki elemanların yerine büyük değerli sayı 2000 yerleştirilir ve aşağıdaki tablo elde edilir.

	A	B	C	D	E
A	2000	35	80	2000	165
B	2000	2000	2000	20	2000
C	80	45	2000	2000	75
D	105	20	30	2000	60
E	165	80	75	2000	2000

Yukarıdaki tabloda işaretlenmemiş en küçük eleman c_{43} = 30'dur. Şimdiki gezi planına 4-3 halkasını da ekleyerek, uygun olmayan ve hala tamamlanmamış 2-4 ve 4-3 gezi planı elde edilir. Ayrıca c_{43} elemanı işaretlenerek ilgili olduğu satır ve sütundaki diğer elemanlar ile dönüşüm elemanı c_{34} yerine 2000 elemanı konularak aşağıdaki tablo elde edilir.

	A	B	C	D	E
A	2000	35	2000	2000	165
B	2000	2000	2000	20	2000
C	80	45	2000	2000	75
D	2000	2000	30	2000	2000
E	165	80	2000	2000	2000

Yukarıdaki tabloda işaretlenmeyen c_{12} =35'tir. Tamamlanmamış şimdiki gezi planına 1-2 halkası ilave edilerek 1-2,2-4,4-3 gezi planı elde edilir. c_{12} elemanı işaretlenir, birinci sıradaki diğer elemanlar ile ikinci sütudaki tüm diğer elemanlar ve dönüşüm elemanı c_{21} nin yerine büyük değerli sayı 2000 konulur. Bu işlemin sonuçları aşağıdaki tabloda görünmektedir.

	A	B	C	D	E
A	2000	35	2000	2000	2000
B	2000	2000	2000	20	2000
C	80	2000	2000	2000	75
D	2000	2000	30	2000	2000
E	165	2000	2000	2000	2000

Aynı işlemlere devam edilerek c_{35} ve c_{51} elemanları da sırası ile işaretlenerek aşağıdaki tablolar elde edilir. Gezi planı son tabloda işaretlenen elmanlar ile gösterilir ki, adlandırılınca gezi planı $1 \to 2,2 \to 4,4 \to 3,3 \to 5,5 \to 1$ olur. Bu tamamlanmış bir plan olup optimale yakındır. Gezinin toplam maliyeti ise z=20+35+30+165+75=325 birim lira olur.

	A	B	C	D	E
A	2000	35	2000	2000	2000
B	2000	2000	2000	20	2000
C	2000	2000	2000	2000	75
D	2000	30	2000	2000	2000
E	165	2000	2000	2000	2000

	A	B	C	D	E
A	2000	35	2000	2000	2000
B	2000	2000	2000	20	2000
C	2000	2000	20002000		75
D	2000	30	2000	2000	2000
E	165	2000	2000	2000	2000

8.BÖLÜM

8.DİNAMİK PROGRAMLAMA

8.1.Dinamik Programlaya Giriş:

Bir ekonomik olaya bağlı bir fonksiyonun optimal değerinin değerinin araştırmanın en güvenilir araçlarından birisi, 20 yıl kadar önce A.B.D' li matematikçi Richard Bellman tarafından açıklanmıştır .Bu analiz aracının önemi ekonomik olaylarda olduğu kadar fizik matematik alanlarda da geçerli olmasındadır .

DP optimizasyon probleminin çoğunu çözmekte kullanılan bir yöntemdir ve problemin sonundan başına doğru çalışarak çözümlerin elde edilmesine dayanır .böylece büyük ve çözülemez problemler çözülebilen küçük problemlerin bir dizisine indirgenmiş olur bu yöntem daha çok yatırım , şebeke , ve kaynak tahsisi problemlerinde kullanılır.(yatırım:inventory,şebeke:network ,kaynak tahsisi:resource allocation)

8.2.Dinamik Programlanın Örneklerle İncelenmesi:

Örnek: Bir masada 30 kibrit ve iki oyuncu vardır öncelikle ilk oyuncu 1,2 veya 3 kibrit toplayarak başlıyor .Sonra rakip 1,2 veya 3 kibrit topluyor .Oyun bu şekilde devam ederken son kibriti olan oyuncu oyunu kaybediyor .Oyunu kazanacağından ilk oyuncu olarak nasıl emin olabilirim.

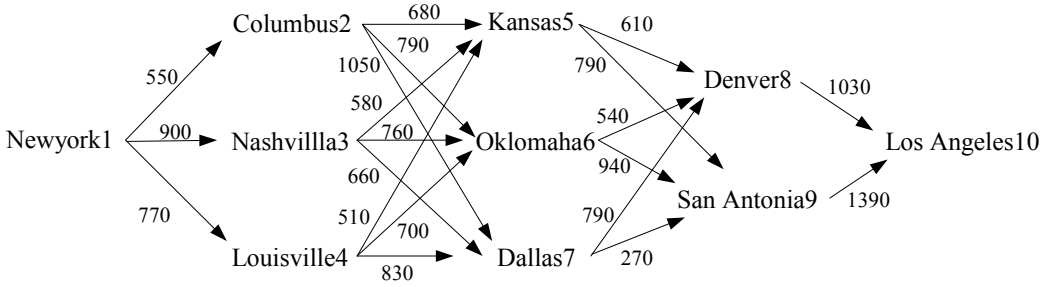
Çözüm: Problem sondan başlanarak çözülür

Eğer masada 5,9,13,17,21,25,29 kibrit kalırsa ben kazanırım bunun için 30-29=1 kibrit olarak ilk benim sıramsa mutlaka kazanırım

Örnek: (şebeke- problemi)DP'nin pek çok uygulaması verilen bir şebekede iki noktayı birleştiren en kısa veya en uzun yolu bulmaya dayanır

Joe NY'ta yaşıyor fakat arabayla LA'ya gitmeyi planlıyor .joe'nin imkanları sınırlıdır ve her geceyi seyahatlerde geçirmek istiyor joe'nin arkadaşlarının bulunduğu şehirler ve aralarında ki mesafeler aşağıda ki tabloda verilmiştir . buna göre joe en az mil giderek LA'ya nasıl ulaşır.

Çözüm:



$F_3(5)=\min$ { C58+f4(8)=610+1030=1640 ; C59+f4(9)=790+1390=2180 }

$F_3(6)=\min$ { C68+f4(8)=540+1030=1570 ; C69+f4(9)=940+1390=2330 }

$F_3(7)=\min$ { C78+f4(8)=790+1030=1820 ; C68+f4(9)=270+1390=1660 }

$F_2(2)=\min$ { C25+f3(5)=680+1640=2320 ; C26+f3(6)=790+1570=2360 ; C27+f3(7)=1050+1660=2710 }

$F_2(3)=\min$ { C35+f3(5)=580+1640=2220 ; C36+f3(6)=760+1570=2330 ; C37+f3(7)=660+1660=2320 }

$F_2(4)=\min$ { C45+f3(5)=510+1640=2150 ; C46+f3(6)700+1570=2270 ; C47+f3(7)=830+1660=248

($f_4(8)=1030$ ve $f_4(9)=1390$)

$F_1(1)=\min$ { C12+f2(2)=550+2320=2870 ; C13+f2(3)=900+2220=3120 ; C14+f2(4)=770+2150=2920

(Burada, $f_n(a)$ şeklindeki ifadelerde n aşama(stage) sayısıdır.)

Sonuç: 1-2-5-8-10=2870 km'dır

Örnek: Bir şirket gelecek 4 ayda ürünü için olan talebin şu şekilde olacağını bilmektedir.

1.ay 1 birim
2.ay 3 birim
3.ay 2 birim
4.ay 4 birim

her ayın başlangıcında şirket geçerli aylarda kaç birim üretilmesi gerektiğini belirlemek zorundadır her hangi bir birimin üretildiği bir ay içinde 3 dolarlık bir maliyetle karşı karşıya kalınmaktadır ayrıca üretilen her birim için 1 dolarlık değişken bir maliyet olmaktadır .Her ay sonunda 50 cent birimlik bir maliyetle karşı karşıya kalınıyor kapasite birimi her ay boyunca mam 5 birim üretilmesine izin veriliyor .Şirketin stoklarının boyutu her ay için olsa olsa 4 birimlik bir üretimin çıkışıyla sınırlanmaktadır bu şirket bütün istekleri zamanında karşılayabilmek için toplam üretim için en uygun koşulları belirlemek ve 4 ay boyunca olacak maliyeti karşılayabilmek çin bir üretim programı belirlemek istemektedir .Bu programa göre ilk ayın başlangıcında 0 birim olduğu varsayılmaktadır Bütün istekler her ayın üretim çıkışı pozitif olacak şekilde sınırlandırılacak ve zamanında karşılanacaktır.

Çözüm:4.ay ürünleri + 3 aydan gelen stoklar = 4 ay talebi

$x_t(i)$= t ayı başında i birimleri üretilmiş t,t+1,.., 4 ayları boyunca tutan toplam harcamaları minimize eden ürün seviyesi

$f_t(i)$=her ay için minimummaliyetle karşılanan talep

t,t+1,..,4 eğer i birimler (t) ayının başında hazır halde ise

C(x) = periyod süresince üretilen x birimlerinin maliyet

C(0)=0 ve x>0 ise c(x)=x+3$

$f_4(0)$=üretim maliyeti 4-0 birim=c(4)=3+4=7$ ve $x_4(0)$=4-0=4

$f_4(1)$= // 4-1 birim=c(3)=3+3=6$ ve $x_4(1)$=4-1=3

$f_4(2)$= // 4-2 birim=c(2)=3+2=5$ ve $x_4(2)$=4-2=2

$f_4(3)$= // 4-3 birim=c(1)=3+1=4$ ve $x_4(3)$=4-3=1

$f_4(4)$= // 4-4 birim=c(0)=0 ve $x_4(4)$=4-4=0

$f_3(i)$'yi belirleyeceğiz , i=0,1,2,3,4(aşamalar aylardır)

$f_3(i)$=min[1/2(ı+x-2)+c(x)+f_4(i+x-2)]($4 \geq i + x - 2 \geq 0$)

$f_3(0)$12,f3(1)=10,$f_3(2)$=7,$x_3(0)$=2 ,$x_3(1)$=5,$x_3(2)$=0

$f_3(3)$=13/2,$f_3(4)$=6,$x_3(3)$=0,$x_3(4)$=0

Örnek:Ekipman (Equipment) Yenileme Problemi

1.yıl	2.yıl	3.yıl	4.yıl	5.yıl
0.zaman	1.zaman	2.zaman	3.zaman	4.zaman 5.zaman

g(t) =t'den 5'e kadar

c_{tx}=t zamanda satın alınan bir makinenın net maliyetini x zamana kadar işletme olarak tanımlıyoruz

c_{tx}=1000$ +mt+...+m_{t-1}-S_{t-1}

G(t)=mil {c_{tx}+g(x)}(t=0,1,2,3,4)

Bir oto tamir dükkanı mutlaka bi r makine analizine ihtiyaç duyar. Yeni bir makine analizi 1000$'dır. Bir makine analizinin 1,2 veya 3 yıl elde tutulabiliyor. Birinci elde tutma maliyeti m_{1i} = 60 $, ikinci yıl elde tutma maliyeti m_{2i}=80$,üçüncü yıl elde tutma maliyeti m_{3i}=120$'dır. X yıl sonra makine bir yenisi ile değiştirilebilir. Bu x yaşındaki makineyi değiştirebilmemizde birinci yıl s_{1i}=800$,ikinci yıl s_{2i}=600$,üçüncü yıl s_{3i}=500$ kurtarma maliyeti elde ediliyor. Bu dükkan yeni bir makine almak zorunda olduğu zaman 5 yıl boyunca bu maliyeti en aza indirmek için ne yapmalıdır.

Çözüm:

c_{01}=1000$+60$-800$=260$= c_{12}=c_{23}=c_{34}=c_{45}

c_{02}=1000$+60$+80-600=540$=$c_{13}$=$c_{24}$=$c_{35}$

c_{03}=1000$+60$+80$+120$-500$=760$=c_{14}=c_{25}

g(t)=min {c_{tx}+g(t)}(t=0,1,2,3,4)

t+1<=x<=t+3,x<=5

g(5)=0

$g(4)=c_{45}+g(5)=260+0=260\$$.

$g(3)=\min$ $c_{34}+g(4)=260+260=520\$$, $c_{35}+g(5)=540+0=54\$$

$g(2)=\min$ $c_{23}+g(3)=260+520=780\$$, $c_{24}+g(4)=540+260=800\$$, $c_{25}+g(5)=760+0=760\$$

$g(1)=\min$ $c_{12}+g(2)=260+760=1020\$$, $c_{13}+g(3)=540+520=1060\$$, $c_{14}+g(4)=760+260=1020\$$

$g(0)=\min$ $c_{01}+g(1)=260+1020=1280\$$, $c_{02}+g(2)=540+760=1300\$$, $c_{03}+g(3)=760+520=1280\$$

9.BÖLÜM

9.Uygulama Programları

9.1 Simpleks Metot

http://vinci.inesc.pt/lp/ internet sitesinden temin ettiğimiz simpleks metot problemlerinin çözümlerini yapan programımız bir java uygulamasıdır. Bu sebepten browser üzerinden herhangi başka bir internet ayarı yapmadan kullanılabilir.

Program temel olarak simpleks problemlerini simpleks yöntem kullanarak ya da dual yöntem kullanarak çözebilmektedir.

Program internet uygulaması olabilmesi için Applet şeklinde hazırlanmıştır. Ayrıntılı bilgi için http://vinci.inesc.pt/lp/ adresine bakabilirsiniz.

Programın işleyişi açısından yazım kuralları mevcuttur. Bunlar kısaca:

Problem minimum ya da maksimum problemi oluşuna göre “min” ya da “max” yazılır. Bu yazının ardından “:” işareti koymak gerekiyor. İlk yazılacak değer de maksimum ya da minimum değerlerden hangisinin hesaplanacağını gösteren bu değer olacaktır.

Verilecek değişken isimleri istenilen şekilde seçilebilir. Sonuçlar verdiğiniz değişken adlarıyla ekrana getirilecektir.

Yazılan her eşitsizlikten sonra “:” işareti kullanmak zorundasınız.

Bu yazım kurallarına dikkat ederek programı kodladığınızda sol ortada bulunan “solve” butonuna bastığınızda sonuçlar bir altta bulunan kutucukta gösterilir.

Seçme sekmelerinden “somefeedback” sekmesini seçerseniz işlemlerin sadece bazı adımları ve sonuç ekrana getirilir. “Only solution” sekmesi seçilirse sadece problemin sonucu ekrana verilir. “All steps” sekmesini seçerseniz işlemlerin tüm adımlarını ekrana listeler. ”Clear” butonuna tıkladığınızda da çözüm ekranındaki sonuçlar silinir.

9.2 Atama Problemleri

Atama problmelerinin çözümlerini yapan uygulama proje grubu tarafından Delphi’de geliştirilmiş Active X teknolojisi kullanılarak internet ortamına aktarılmış bir yazılımdır.

Bu yazılımızdan biz de basit bir yazım kuralı kullandık. Temelde makinalar ve onlara atanacak işçiler olmak üzere bir matris oluşturulmuştur. Bu ilk matristir ve makinalar üstte işçiler solda olacak şekilde eşleşmeye denk gelen değerler ilk matrise yazılır.
Kullandığımız yazım kuralı şudur:
Her bir satır sütun kesişimindeki değeri yazdıktan sonra “;” işsareti koymanız gerekmektedir
Bu değerler yazıldıktan sonra “işle” butonuna basılır. Atama problmelerinin çözümü bölümünde anlatıldığı gibi ilk önce yatayda en küçük değer ya da seçilen sekmeye göre en büyük değer bulunur. Daha sonra satırlar bazında işlemler yapılır. “yatay işlemler” matris bunu gösterir.Daha sonra aynı işlemler dikey için de yapılır. “dikey işlemler” matrisine o değerler işlenir. Eğer uygun çözüm yoksa Atama problemlerinde bahsi geçen Adım-4 işlemleri gerçekleştirilir. Ardından eğer uygun çözüm varsa bu uygun çözüm matrisi bir sonraki “Çözüm” matrisine yerleştirilir. Son bölümde ise metinsel olarak verilen problemin çözümünün ne anlama geldiği “Sonuçlar” belirtilir.

Maksimum ya da minimum problemlerinden istenilen yapı bu program dahilinde çözülebilir. Ancak maksimum ya da minimum olma durumunu sağ üst köşedeki seçim sekmelerinde belirtmek gerekmektedir.

KAYNAKLAR

1) Bunday,B.D.,Basic Optimization Methods, Edward Arnold Ltd, London, 1984.

2) Kahaner , D.,Moler, C., Nash, S.,Numerical Methods and Software, Prentice Hall, Inc.Englewood Cliffs, NJ, 1989.

3)Himmelblau,D.M., Edgar,T.F.,Optimization of Chemical Processes, McGraw-Hill,Inc.NY,1989.

4) Kübat,C.,Yöneylem Araştırması,8. Ulusal Kongresi,Ankara,1983.

5) Rao,S.S.,Optimization Theory and Applications, 2nd.Edition, Halsted, Inc. 1978.

6) Kara,İ., Yöneylem Araştırması, Doğrusal Olmayan Modeller , Anadolu Üniversitesi Basımevi, Eskişehir,1986.

7) Hillier, F.S., Lieberman, G.J., Introduction to Mathematical Programming, McGraw-Hill , Inc.U.S.A., 1990.

8) Bazaraa, M.S., Shetty,C.M., Nonlinear Programming, John Wiley&Sons,Inc. 1979.

9) Saaty,T.L.,Bram,J., Nonlinear Mathematics , McGraw-Hill, Inc,NY, 1964.

10) Brachen, J.,McCormick,G.P.,Selected Applications of Nonlinear Programming, John Wiley&Sons,Inc.NY,1968.

11) Taha,H.A.,Operations Research, Forth Edition,MacMillan Publishing Company, Inc.NY,1989.

12) Mathews,J.,Numerical Methods,Prentice Hall,Inc. Englewood Cliffs,NJ,1987.

13) Ravindran, A.,Philips,D.T.,Solberg, J.L., Operations Research, Principles and Practice, John Wiley&Sons,Inc.1987.

14) Şenel, M., Genel Matematik, Bilim ve Teknik Kitapevi, İstanbul, 1983.

15) Wilde, D.,Beightler,C.S.,Foundations of Optimation, Prentice-Hall, Inc. Englewood Cliffs,NJ,1967.

16) Calter,P.,Technical Mathematics with Calculus, Prentice-Hall, Inc. Englewood Cliffs,NJ, 1990.

17) Kramer,A.D., Fundamentals of Technical Mathematics with Calculus, Second Edition, McGraw-Hill, Inc.U.S.A., 1989.

18) Tulunay, Y., Matematik Programlama ve İşletme Uygulamaları, Sermet Matbaası, İstanbul, 1980.

19) Hilliers, F.S., Lieberman, G.J., Introduction to Operations Research, Fourth Edition, McGraw-Hill, Inc. 1989.

20) Schmidt,J.W., Davis, R.P., Foundations of Analysis in Operations Research, Academic Press,NY,1981.

21) Walsh, G.R., Methods of Optimazation, John Wiley & Sons, Inc. NY,1975.

22) Altıntaş.Y., Nümerik Analiz, İ.T.Ü. Sakarya Mühendislik Fakültesi Matbaası, 3. Baskı, Adapazarı, 1989.

23) Aoki,M., Introduction to Optimazation Techniques, Fundamentals and Applications of Nonlinear Programming, The MacMillan Company, Inc.NY, 1971.

24) Lasdon, L.S., Optimazation Theory for Large System, MacMillan Publishing Com., Inc. NY, 1970.

25) Beightler,C.S., Philips,D.T.,Widde, D.J., Foundations of Optimazation, Second Edition, Prentice-Hall, Inc. Englewood Cliffs,NJ, 1979.

26) Humphreys, K.K., Jelen's Cost and Optimazation Engineering, McGraw-Hill, Inc. 1991.

27) Zoutendijk, G., Nonlinear Programming, Prentice-Hall, Inc. NJ, 1969.

28) Martos, B., Nonlinear Programming, Theory and Methods, North Holland, Inc. Amsterdam, 1975.

29) Fiacco, A.V., McCormick, G.P., Nonlinear Programming, Sequential Unconstrained Minimization Techniques, John Wiley & Sons,Inc.NY, 1969.

30) Zangwill, W., Nonlinear Programming, Prentice-Hall, Inc. Englewood Cliffs,NJ, 1969.

31)Tektaş,M. : M.Ü. Fen Bilimleri Enstitüsü, Matematik Bölümü (Yüksek Lisans Tezi: Uygulamaları ve Karşılaştırmalı Yorumları İle Bazı Doğrusal Olmayan Programlama Teknikleri)

32)Koçer,M., "Harekat araştırması''Ankara (1981)

33)Harvey,M.W., "Principles of Operations Research'',Prentice-Hall N.J.no:2

34)Sezginman,H.İ., "Lineer Programlama (Teori ve Problemleri)''İSTANBUL-(1986)

35)Halaç,O., "Kantitatif Karar Verme Teknikleri(Yöneylem Araştırması)''no:3 İSTANBUL-(1991)

36) Bronson,R.,Naadımuthu,G., "Operations Research'',Second Editions, Mc Graw-Hill, Inc. NY, 1998.

37)DOĞAN, İ., Yöneylem Araştırması Teknikleri ve İşletme Uygulamaları, Bilim Teknik Yayınevi, 1995, s.3.

.

EK.1.PROGRAM VE PROGRAM TANIMI

Doğrusal Olmayan Programlama Tekniklerini incelediğimizde bu çalışmada Gradient, Frank-Wolfe ve SUMT algoritmalarına ait uygulamalar Mathprog adlı bir paket programla yapılmıştır. Bu uygulamalara ait yorumların daha iyi yapılması açısından bu uygulamalara bazı ilaveler yapılmıştır. Diğer kalan Nelder - Mead Metod ve Newton-Raphson Metodları için Pascal Programlama dilinde ayrı ayrı program yazılmıştır.

a) NELDER – MEAD METODU İÇİN PROGRAM

Bu programda amaç fonksiyonu program içerisinden tanımlamaktır . Tanımlama FUNC prosedürü içerisinden yapılmaktadır.

Program çalıştığında tanımlanan fonksiyon değişken sayısı , iterasyon adedi , maksimum-minimum seçimi ve başlangıç vektör elemanları VEKTÖR prosedürü yardımıyla belirlenir.
RME prosedürü yardımı ile başlangıç vektörleri fonksiyonda yerine konur ve çıkan değerlerden BGW prosedürü kullanılarak BEST (en iyi), GOOD (sonraki en iyi) ve WORST (en kötü) noktalar tespit edilir. Bu işlem istenen iterasyon sayısı kadar devam ettirilir. İstendiği taktirde 999 iterasyon yazıcı veya ekran üzerine alınarak incelenebilir.

ÇERÇEVE, BOŞA, BOŞALT, ALT_ÇERÇEVE isimli prosedürler değişken değeri sıfırlama ve ekran tasarımı için kullanılan program parçalarıdır.

b) NEWTON – RAPHSON METODU İÇİN PROGRAM

Programın amacı bir boyutlu problemlerin Newton-Raphson Metodu ile çözümüdür.

Çözüm için gerekli fonksiyon fx(p) tek değişkenli fonksiyon parçasında, türevi ise fxt(p) tek değişkenli fonksiyon parçasında program içerisinde tanımlanmalıdır.

BİLGİ_AL prosedürü yardımı ile başlangıç değeri , iterasyon sayısı ve hata miktarı belirlenir.

İTERASYON isimli prosedür ile verilen iterasyon miktarı kadar veya tanımlı fonksiyondan çıkan değer , istenilen hassasiyetten daha küçük kalıncaya kadar iterasyon işlemine devam edilir.

Sonuçlar istendiği taktirde yazıcı üzerine alınıp incelenebilir.

ÇERÇEVE, ALT_ ÇERÇEVE, ve YAZICI isimli program parçaları , ekran tasarımı ve yazıcı çıktıları için kullanılmıştır.

a)PROGRAM NELDERMEAD;

```
uses crt,dos,PRINTER;
VAR
Q,QQ,YER,SAT:BYTE;
grv:array[1..10,1..3,1..20] of real;
CH,CV:CHAR;
V:array[1..3,1..20] of real;
A:array[1..20] of real;
F,PRM:real;
ITER,K,I,J,N,L,H:INTEGER;
X:array[1..20] of real;
M,E,R,C,S:array[1..20] of real;
AMAC:STRING[1];
{Amac fonksiyon}(*amac fonk*)
PROCEDURE FUNC(N:BYTE);
VAR
A,B,C:REAL;
begin
    F:=X[1]*X[1]-4*X[1]+X[2]*X[2]-X[2]-X[1]*X[2];
end;
PROCEDURE VEKTOR;
VAR
Z:BYTE;
begin
    Z:=13;
GOTOXY(5,6);WRITE(' DEGISKEN SAYISI....>');READLN(N);
GOTOXY(5,8);WRITE(' ITERASYON ADEDI....>');READLN(ITER);
GOTOXY(5,10);WRITE(' [1].MIN   [2].MAX....>');READLN(AMAC);
GOTOXY(5,12);WRITE(' EKRAN YAZICI.[E/Y]....>');READLN(CV);
            FOR K:=1 TO 3 DO
       BEGIN
       GOTOXY(5,Z);WRITELN('V',K,'VEKTORU ELEMANLARI');
       INC(Z);
              FOR J:=1 TO N DO
              BEGIN
              GOTOXY(5,Z);WRITE(J,'.CI ELEMAN=');READLN(V[K,J]);
              INC(Z);
              END;
       END;
         FOR J:=1 TO N DO
         BEGIN
         grv[1,1,j]:=v[1,j];
         grv[1,2,j]:=v[2,j];
         grv[1,3,j]:=v[3,j];
         END;
END;
PROCEDURE BGW;
VAR
```

```
FV:array[1..3] of real;
BEGIN
    FOR I:=1 TO 3 DO
    BEGIN
        FOR J:=1 TO N DO
        BEGIN
        X[J]:=V[I,J];
        END;
        FUNC(20);
        FV[I]:=F;
    END;
        FOR J:=3 DOWNTO 1 DO BEGIN
           FOR I:=1 TO J-1 DO BEGIN
        IF FV[J]>FV[I] THEN BEGIN
          PRM:=FV[I];
          FV[I]:=FV[J];
          FV[J]:=PRM;
        FOR L:=1 TO N DO BEGIN
        PRM:=V[I,L];
        V[I,L]:=V[J,L];
        V[J,L]:=PRM;
        END;
        END;
    END;
    END;
    IF AMAC='1' THEN BEGIN
      FOR L:=1 TO N DO BEGIN
      PRM:=V[1,L];
      V[1,L]:=V[3,L];
      V[3,L]:=PRM;
      END;
    END;
END;
PROCEDURE BOSA;
BEGIN
    FOR J:=1 TO 20 DO BEGIN
    A[J]:=0;
    M[J]:=0;
    E[J]:=0;
    R[J]:=0;
    S[J]:=0;
    C[J]:=0;
    END;
END;
PROCEDURE BOSALT;
BEGIN
   FOR I:=1 TO 3 DO
       FOR J:=1 TO 20 DO
```

```
        V[I,J]:=0;
END;
PROCEDURE RME;
VAR
B1,B2,B3,B4:real;
BEGIN
    FOR I:=1 TO N DO
    BEGIN
    M[I]:=(V[1,I]+V[2,I])/2;
    R[I]:=2*M[I]-V[3,I];
    E[I]:=2*R[I]-M[I];
    C[I]:=(V[3,I]+M[I])/2;
    S[I]:=(V[1,I]+V[3,I])/2;
    END;
    FOR I:=1 TO N DO
    X[I]:=R[I];
    FUNC(20);
    B1:=F;
    FOR I:=1 TO N DO
    X[I]:=V[2,I];
    FUNC(20);
    B2:=F;
    IF B1<B2 THEN
      BEGIN
      FOR I:=1 TO N DO
      X[I]:=V[1,I];
      FUNC(20);
      B3:=F;
    IF B3<B1 THEN
      BEGIN
      FOR I:=1 TO N DO
      BEGIN
      V[3,I]:=R[I];
      END;
      END
    ELSE
    BEGIN
      FOR I:=1 TO N DO
      X[I]:=E[I];
      FUNC(20);
      B4:=F;
      FOR I:=1 TO N DO
      X[I]:=V[1,I];
      FUNC(20);
      B3:=F;
    IF B4<B3 THEN
    BEGIN
       FOR I:=1 TO N DO
```

```
    V[3,I]:=E[I];
END
ELSE
BEGIN
     FOR I:=1 TO N DO
     V[3,I]:=R[I];
END;
END;
{ELSE
BEGIN
     FOR I:=1 TO N DO
     X[I]:=R[I];}
     FUNC(20);
     B1:=F;
     FOR I:=1 TO N DO
     X[I]:=V[3,I];
     FUNC(20);
     B2:=F;
IF B1<B2 THEN
BEGIN
    FOR I:=1 TO N DO
    V[3,I]:=R[I];
    FOR I:=1 TO N DO
    X[I]:=C[I];
FUNC(20);
B3:=F;
FOR I:=1 TO N DO
X[I]:=V[3,I];
END;
FUNC(20);
B2:=F;
IF B3<B2 THEN
BEGIN
    FOR I:=1 TO N DO
    V[3,I]:=C[I];
END
ELSE
BEGIN
    FOR I:=1 TO N DO
    X[I]:=S[I];
    FUNC(20);
    B4:=F;
    FOR I:=1 TO N DO
    BEGIN
        V[3,I]:=S[I];
        V[2,I]:=M[I];
    END;
END;
```

```
    END;
END;
PROCEDURE CERCEVE;
VAR
X:BYTE;
BEGIN
textcolor(15);textbackground(6);
clrscr;
writeln;
writeln;
writeln('||================================================================
================||');
for x:=1 to 21 do begin
gotoxy(1,3+x);
write('||');
gotoxy(79,3+x);
write('||');
end;
writeln('||================================================================
================||');
gotoxy(7,4);writeln('            **IKI BOYUTLU PROBLEMLER ICIN NELDER-MEAD
METODU**');
END;
PROCEDURE alt_cerceve;
var
x:byte;
begin
gotoxy(2,5);writeln('||===||========||========||========||========||');
gotoxy(2,6);writeln('|| k ||B(Xk,Yk)||G(Xk,Yk)||W(Xk,Yk)||F(Xk,Yk)||');
gotoxy(2,7);writeln('||===||========||========||========||========||');
for x:=1 to 16 do
begin
    gotoxy(2,7+x);writeln('||   ||       ||       ||       ||       ||');
    gotoxy(2,24); writeln('||===||========||========||========||========||');
end;
end;
PROCEDURE YAZICI;
BEGIN
    write(LST,H:3,'    ');
    write(LST,'(',V[1,1]:6:4,',',V[1,2]:6:4,')    ');
    write(LST,'(',V[2,1]:6:4,',',V[2,2]:6:4,')    ');
    write(LST,'(',V[3,1]:6:4,',',V[3,2]:6:4,')    ');
    X[1]:=V[1,1];
    X[2]:=V[1,2];
    FUNC(2);
    write(LST,F:18:16);
END;
BEGIN
```

```
cerceve;
YER:=2;
BOSALT;BOSA;VEKTOR;
ALT_CERCEVE;
SAT:=8;
IF UPCASE(CV)='Y' THEN
BEGIN
GOTOXY(1,25);WRITE('YAZICIYI ACIP BIR TUSA BASINIZ....');
CH:=READKEY;
writeln(LST,'IKI BOYUTLU PROBLEMLER ICIN NELDER-MEAD METODU');
writeln(LST,'-----------------------------------------------');
writeln(LST,'');
writeln(LST,'Degisken say□s□.....:',n:3);
writeln(LST,'Iterasyon say□s□.....:',iter:3);
writeln(LST,'[1].Min [2].Max.....:',amac:3);
writeln(LST,'Amac Fonksiyon.....:x^2-4x+y^2-y-xy');
writeln(LST,'Vektor Elemanlar□');
writeln(LST,'1.vektor elemanlar□:(',v[1,1]:6:4,',',v[1,2]:6:4,')');
writeln(LST,'2.vektor elemanlar□:(',v[2,1]:6:4,',',v[2,2]:6:4,')');
writeln(LST,'3.vektor elemanlar□:(',v[3,1]:6:4,',',v[3,2]:6:4,')');
writeln(LST,'-----------------------------------------------');
writeln(LST,' k   B(Xk,Yk)  G(Xk,Yk)  W(Xk,Yk)  F(Xk,Yk)  ');
writeln(LST,'-----------------------------------------------');
end;
for H:=1 to 20 do
X[H]:=0;
for H:=1 to ITER do
begin
BGW;RME;
gotoxy(4,sat); write(H:3);
gotoxy(8,sat); write(V[1,1]:6:4,',',V[1,2]:6:4);
gotoxy(24,sat); write(V[2,1]:6:4,',',V[2,2]:6:4);
gotoxy(42,sat); write(V[3,1]:6:4,',',V[3,2]:6:4);
if upcase(CV)='Y' then
YAZICI;
X[1]:=V[1,1];
X[2]:=V[1,2];
FUNC(2);
gotoxy(61,sat); write(F:18:16);
INC(SAT);
IF SAT>23 THEN
BEGIN
    CH:=READKEY;
    CERCEVE;
    ALT_CERCEVE;
    SAT:=8;
    END;
    END;
```

```
IF UPCASE(CV)='Y' THEN
WRITELN(LST,^1);
CH:=READKEY;

END.
```

b) PROGRAM NEWTON_RAPHSON;

```
uses crt,dos,printer;
var
ch:char;
b,hata,z,s:real;
x,iter,sat:byte;
function fx(p:real):real;
begin
    {p:=p*3.14/180;}
    fx:=exp(p)+p;
end;
function fxt(p:real):real;
begin
     {p:=p*3.14/180; }
          fxt:=exp(p)+1;
end;
PROCEDURE cerceve;
VAR
X:BYTE;
BEGIN
textcolor(15);textbackground(6);
clrscr;
writeln;
writeln;
writeln('||==========================================================================||');
for x:=1 to 21 do begin
gotoxy(1,3+x);
write('||');
gotoxy(79,3+x);
write('||');
end;
writeln('||==========================================================================||');
gotoxy(7,4);writeln('    ***IKI BOYUTLU PROBLEMLER ICIN NEWTON-RAPHSON METODU***');
END;
PROCEDURE bilgi_al;
begin
GOTOXY(7,10);WRITE('BASLANGIC DEGERINI GIRINIZ....>');READLN(b);
GOTOXY(7,12);WRITE('ITERASYON SAYISINI GIRINIZ....>');READLN(ITER);
GOTOXY(7,14);WRITE('HATA TOLERANSINI GIRINIZ....>');READLN(hata);
```

```
end;
PROCEDURE alt_cerceve;
var
x:byte;
begin
gotoxy(2,5);writeln('||===||========||========||========||========||');
gotoxy(2,6);writeln('||   ||        ||  F(Xk) ||        ||        ||');
gotoxy(2,7);writeln('|| k ||  Xk    ||Xk+1=Xk-||  F(Xk) || Xk+1-Xk||');
gotoxy(2,8);writeln('||   ||        ||  F(Xk) ||        ||        ||');
gotoxy(2,9);writeln('||===||========||========||========||========||');
for x:=1 to 14 do
begin
gotoxy(2,9+x);writeln('||   ||        ||        ||        ||        ||');
end;
gotoxy(2,24);writeln('||===||========||========||========||========||');
end;
PROCEDURE yazici;
label 10;
begin
writeln(lst,'***IKI       BOYUTLU       PROBLEMLER       ICIN       NEWTON-RAPHSON
METODU***');
writeln(lst,'-----------------------------------------------------');
writeln(lst,'                    F(Xk)                  ');
writeln(lst,'    k    Xk     Xk+1=Xk---       F(Xk)        Xk+1-Xk  ');
writeln(lst,'                    F(Xk)                  ');
writeln(lst,'-----------------------------------------------------');
b:=s;
for x:=1 to iter do
begin
z:=b-fx(b)/fxt(b);
write(lst,x:3,'       ');
write(lst,b:13:10,'       ');
write(lst,z:13:10,'       ');
write(lst,fx(b):13:10,'       ');
write(lst,ABS(Z-B):13:10,'       ');
if fx(z)<=hata then
goto 10;
b:=z;
end;
10: writeln(lst,^L);
end;
PROCEDURE iterasyon;
begin
cerceve;
alt_cerceve;
sat:=10;
for x:=1 to iter do
begin
```

```
z:=b-fx(b)/fxt(b);
gotoxy(4,sat);write(x:3);
gotoxy(8,sat);write(b:13:10);
gotoxy(24,sat);write(z:13:10);
gotoxy(42,sat);write(fx(b):13:10);
gotoxy(61,sat);write(ABS(Z-B):13:10);
if ABS(fx(z))<=hata then
exit;
inc(sat);
if sat>23 then
begin
ch:=readkey;
cerceve;
alt_cerceve;
sat:=10;
end;
b:=z;
end;
end;
begin
clrscr;
cerceve;
bilgi_al;
s:=b;
iterasyon;
gotoxy(1,25);write(' Yazıcıya aktarmak ister misiniz[E/H] ?');
ch:=readkey;
if upcase(ch)='E' then
yazici;
end.
```

EK.2. Doğrusal Olmayan Programlama Tekniklerinin Yorumsuz Ek Uygulamaları

1) Doğrusal Olmayan Programlama

Değişken sayısı : 2

Kısıtların sayısı : 2

Max Z = $-0.07x_1+0.38x_2$

Kısıtlar

$3x_1 + lx_2 \leq 11$

$2x_l + 5x_2 \leq 16$

ve

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

Frank - Wolfe Algoritması

Başlangıç Aşikar Çözüm: (0, 0)

f(x) = $32_{Xl} + 50x_2 - 10x_2^2 + 1x_2^3 - 1x_x^4 - 1x_2^4$

$df7dx_1 = 32 - 4_{Xl}^3$

$df/dx_2 = 50 - 20x_2 + 3x_2^2 - 4x_2^3$

k	$x^{(k-1)}$	c_1	c_2	$x_{LP}^{(k)}$	t^*	x^k
1	(0, 0)	32	50	(3, 2)	0.729	(2.188, 1.458)
2	(2.188, 1.458)	-9.87	14.81	(0, 3.2)	0.131	(1.902, 1.686)
3	(1.902, 1.686)	4.499	5.634	(3, 2)	0.111	(2.024, 1.721)
4	(2.024, 1.721)	-1.15	4.078	(0, 3.2)	0.028	(1.966, 1.763)
5	(1.966, 1.763)	1.597	2.149	(3, 2)	0.041	(2.008, 1.773)
6	(2.008, 1.773)	-0.4	1.697	(0, 3.2)	0.011	(1.986, 1.788)
7	(1.986, 1.788)	0.655	0.958	(3, 2)	0.017	(2.004, 1.792)
8	(2.004, 1.792)	-0.18	0.785	(0, 3.2)	0.005	(1.994, 1.799)
9	(1.994, 1.799)	0.291	0.455	(3, 2)	0.007	(2.001, 1.8)
10	(2.001, 1.8)	-0.07	0.383	(0, 3.2)	0.002	(1.997, 1.803)

2) İKİ BOYUTLU PROBLEMLER İÇİN NELDER - MEAD METODU

Değişken Sayısı : 2

İterasyon Sayısı : 20

[1]. Min [2]. Max : l

Amaç Fonksiyon : $x^4 + 2y^2 - 8xy$

Vektör Elemanları

1. Vektör Elemanları : (2.0000, 2.0000)

2. Vektör Elemanları : (3.0000, 2.0000)

3. Vektör Elemanları : (3.0000, 3.0000)

k	B (x_k , y_k)	G (x_k . y_k)	W(x_k,y_k)	F (x_k, y_k)
1	(2.0000, 2.0000)	(3.0000, 3.0000)	(1.5000,3.5000)	-8.00000000000
2	(1.5000,3.5000)	(2.0000, 2.0000)	(2.3750, 2.8750)	-12.43750000000
3	(1.5000,3.5000)	(2.0000, 2.0000)	(1.1250,2.6250)	-12.43750000000
4	(1.5000,3.5000)	(1.3125,3.0625)	(1.7500, 2.7500)	-12.43750000000
5	(1.7500,2.7500)	(1.5000, 3.5000)	(1.9375, 3.1875)	-13.99609375000
6	(1.9375,3.1875)	(1.8438,2.9688)	(1.7188,3.3438)	-14.99412536600
7	(1.9375,3.1875)	(1.7188,3.3438)	(1.7969,3.8594)	-14.99412536600
8	(1.7969, 3.8594)	(1.9375,3.1875)	(2.0156, 3.7031)	-15.26407522000
9	(2.0156,3.7031)	(1.9063,3.7813)	(1.9766, 3.4453)	-15.78073114200
10	(1.9063,3.7813)	(2.0156,3.7031)	(1.9453,4.0391)	-15.86393642400
11	(1.9453,4.0391)	(1.9258,3.9102)	(1.9805,3.8711)	-15.90938055100
12	(1.9805,3.8711)	(1.9453,4.0391)	(2.0000, 4.0000)	-15.97781214500
13	(2.0000, 4.0000)	(1.9902,3.9355)	(1.9727,4.0195)	-16.00000000000
14	(2.0000, 4.0000)	(1.9951,3.9678)	(1.9863,4.0098)	-16.00000000000
15	(2.0000, 4.0000)	(1.9976,3.9839)	(1.9932,4.0049)	-16.00000000000
16	(2.0000, 4.0000)	(1.9988,3.9919)	(1.9966,4.0024)	-16.00000000000
17	(2.0000, 4.0000)	(1.9994,3.9960)	(1.9983,4.0012)	-16.00000000000
18	(2.0000, 4.0000)	(1.9997,3.9980)	(1.9991,4.0006)	-16.00000000000
19	(2.0000, 4.0000)	(1.9998,3.9990)	(1.9996,4.0003)	-16.00000000000
20	(2.0000, 4.0000)	(1.9999,3.9995)	(1.9998,4.0002)	-16.00000000000

3) İKİ BOYUTLU PROBLEMLER İÇİN NELDER - MEAD METODU

Değişken Sayısı : 2

İterasyon Sayısı : 20

[1]. Min [2]. Max : 1

Amaç Fonksiyon : $x^4 + 2y^2 - 8xy$

Vektör Elemanlarları

1. Vektör Elemanları : (-1.0000,-2.0000)

2. Vektör Elemanları : (-1.0000,-3.0000)

3. Vektör Elemanları : (-2.0000,-2.0000)

k	$B(x_k, y_k)$	$G(x_k, y_k)$	$W(x_k,y_k)$	$F(x_k, y_k)$
1	(-2.0000, -2.0000)	(-1.5000, -2.0000)	(-1.5000, -2.5000)	-8.00000000000
2	(-1.5000, -2.5000)	(-1.5000, -2.2500)	(-1.7500, -2.2500)	-12.43750000000
3	(-1.5000, ,2.5000)	(-1.7500, -2.2500)	(-1.8750, -2.6250)	-12.43750000000
4	(-1.8750, -2.6250)	(-1.5000, -2.5000)	(-1.5625, -3. 1875)	-13.23413085900
5	(-1.5625, -3.1875)	(-1.8750, -2.6250)	(-2.1563, -3.7188)	-13.56297302200
6	(-2.1563, -3.7188)	(-1.5625, -3.1875)	(-1.8438, -4.2813)	-14.87318325000
7	(-1.8438, -4.2813)	(-2.0000, -4.0000)	(-1.7031, -3 .7344)	-14.93421840700
8	(-2.0000, -4.0000)	(-1.8438, -4.2813)	(-2.1406, -4.5469)	-16.00000000000
9	(-2.0000, -4.0000)	(-2.0703, -4.2734)	(-1.9219, -4.1406)	-16.00000000000
10	(-2.0000, -4.0000)	(-2.0352, -4.1367)	(-1.9609, -4 .0703)	-16.00000000000
11	(-2.0000, -4.0000)	(-2.0176, -4.0684)	(-1.9805, -4.0352)	-16.00000000000
12	(-2.0000, -4.0000)	(-2.0088, -4.0342)	(-1.9902, -4.0176)	-16.00000000000
13	(-2.0000, -4.0000)	(-2.0044, -4.0171)	(-1.9951, -4.0088)	-16.00000000000
14	(-2.0000, -4.0000)	(-2.0022, -4.0085)	(-1.9976, -4.0044)	-16.00000000000
15	(-2.0000, -4.0000)	(-2.0011, -4.0043)	(-1.9988, -4.0022)	-16.00000000000
16	(-2.0000, -4.0000)	(-2.0005, -4.0021)	(-1.9994, -4.0011)	-16.00000000000
17	(-2.0000, -4.0000)	(-2.0003, -4.0011)	(-1.9997, -4.0005)	-16.00000000000
18	(-2.0000, -4.0000)	(-2.0001, -4.0005)	(-1.9998, -4.0003)	-16.00000000000
19	(-2.0000, -4.0000)	(-2.0001, -4.0003)	(-1.9999, -4.0001)	-16.00000000000
20	(-2.0000, -4.0000)	(-2.0000, -4.0001)	(-2.0000, -4.0001)	-16.00000000000

4) İKİ BOYUTLU PROBLEMLER İÇİN NELDER - MEAD METODU

Değişken Sayısı : 2

İterasyon Sayısı : 15

[1]. Min [2]. Max : l

Amaç Fonksiyon : $x^3 + y^3 - 3x - 3y + 5$

Vektör Elemanlarları

1. Vektör Elemanları : (1.0000, 2.0000)

2. Vektör Elemanları : (2.0000, 0.0000)

3. Vektör Elemanları : (2.0000, 2.0000)

k	$B(x_k, y_k)$	$G(x_k, y_k)$	$W(x_k,y_k)$	$F(x_k, y_k)$
1	(1.0000,2.0000)	(2.0000, 0.0000)	(1.0000,0.0000)	5.00000000000
2	(1.0000,0.0000)	(1.0000, 1.0000)	(1.5000,0.0000)	3.00000000000
3	(1.0000, 1.0000)	(1.0000,0.0000)	(0.5000, 1.0000)	1.00000000000
4	(1.0000, 1.0000)	(0.7500, 1.0000)	(1.0000,0.5000)	1.00000000000

5	(1.0000, 1.0000)	(0.8750, 1.0000)	(1.0000,0.7500)	1.00000000000
6	(1.0000, 1.0000)	(0.9375, 1.0000)	(1.0000, 0.8750)	1.00000000000
7	(1.0000, 1.0000)	(0.9688, 1.0000)	(1.0000,0.9375)	1.00000000000
8	(1.0000, 1.0000)	(0.9844, 1.0000)	(1.0000,0.9688)	1.00000000000
9	(1.0000, 1.0000)	(0.9922, 1.0000)	(1.0000,0.9844)	1.00000000000
10	(1.0000, 1.0000)	(0.9961, 1.0000)	(1.0000,0.9922)	1.00000000000
11	(1.0000, 1.0000)	(0.9980, 1.0000)	(1.0000,0.9961)	1.00000000000
12	(1.0000, 1.0000)	(0.9990, 1.0000)	(1.0000,0.9980)	1.00000000000
13	(1.0000, 1.0000)	(0.9995, 1.0000)	(1.0000,0.9990)	1.00000000000
14	(1.0000, 1.0000)	(0.9998, 1.0000)	(1.0000,0.9995)	1.00000000000
15	(1.0000, 1.0000)	(0.9999, 1.0000)	(1.0000,0.9998)	1.00000000000

5) İKİ BOYUTLU PROBLEMLER İÇİN NELDER - MEAD METODU

Değişken Sayısı : 2

İterasyon Sayısı : 16

[1]. Min [2]. Max : 1

Amaç Fonksiyon : $x^2 + y^2 + 1x - 2y - xy + 1$

Vektör Elemanları

1. Vektör Elemanları : (0.0000, 2.0000)

2. Vektör Elemanları : (2.0000, 0.0000)

3. Vektör Elemanları : (2.0000, 1.0000)

k	B(x_k, y_k)	G (x_k , y_k)	W(x_k,y_k)	F (x_k, y_k)
1	(0.0000, 0.0000)	(2.0000,1.0000)	(0.0000,1.0000)	1.00000000000
2	(0.0000, 1.0000)	(0.0000, 0.0000)	(1.0000,0.7500)	0.00000000000
3	(0.0000, 1.0000)	(0.0000, 0.0000)	(-1.0000,0.2500)	0.00000000000
4	(0.0000, 1.0000)	(-0.5000, 0.6250)	(0.0000, 0.5000)	0.00000000000
5	(0.0000, 1.0000)	(-0.2500,0.8125)	(0.0000, 0.7500)	0.00000000000
6	(0.0000, 1.0000)	(-0.1250,0.9063)	(0.0000, 0.8750)	0.00000000000
7	(0.0000, 1.0000)	(-0.0625,0.9531)	(0.0000, 0.9375)	0.00000000000
8	(0.0000, 1.0000)	(-0.0313,0.9766)	(0.0000, 0.9688)	0.00000000000
9	(0.0000, 1.0000)	(-0.0156, 0.9883)	(0.0000, 0.9844)	0.00000000000
10	(0.0000, 1.0000)	(-0.0078,0.9941)	(0.0000, 0.9922)	0.00000000000
11	(0.0000, 1.0000)	(-0.0039,0.9971)	(0.0000,0.9961)	0.00000000000
12	(0.0000, 1.0000)	(-0.0020, 0.9985)	(0.0000,0.9980)	0.00000000000
13	(0.0000, 1.0000)	(-0.0010, 0.9993)	(0.0000, 0.9990)	0.00000000000
14	(0.0000, 1.0000)	(-0.0005, 0.9996)	(0.0000, 0.9995)	0.00000000000
15	(0.0000, 1.0000)	(-0.0002, 0.9998)	(0.0000, 0.9998)	0.00000000000
16	(0.0000, 1.0000)	(-0.0001,0.9999)	(0.0000, 0.9999)	0.00000000000

6) İKİ BOYUTLU PROBLEMLER İÇİN NELDER - MEAD METODU

Değişken Sayısı : 2

İterasyon Sayısı : 15

[1]. Min [2]. Max : 1

Amaç Fonksiyon : $x^2 + xy^2 - 3xy$

Vektör Elemanları

1. Vektör Elemanları : (0.0000, 0.0000)

2. Vektör Elemanları : (2.0000, 0.0000)

3. Vektör Elemanları : (2.0000, 1.0000)

k	B(x_k, y_k)	G (x_k , y_k)	W(x_k,y_k)	F (x_k, y_k)
1	(2.0000, 1.0000)	(2.0000, 0.5000)	(1.0000,0.5000)	0.00000000000
2	(1.0000, 0.5000)	(1.5000,0.5000)	(1.5000,0.7500)	-0.75000000000
3	(1.5000, 0.7500)	(1.5000,0.6250)	(1.2500,0.6250)	-0.84375000000
4	(1.2500, 0.6250)	(1.5000,0.7500)	(1.1250,0.8125)	-0.87890625000
5	(1.1250,0.8125)	1.2500,0.6250)	(1.3438,0.7344)	-0.97119140625
6	(1.1250,0.8125)	(1.3438,0.7344)	(1.2188,0.9219)	-0.97119140625
7	(1.1250,0.8125)	(1.2188,0.9219)	(0.8281, 1.1328)	-0.97119140625
8	(0.8281, 1.1328)	(0.9766, 0.9727)	(1.0234, 1.0273)	-0.97475528717
9	(0.9766, 0.9727)	(1.0234, 1.0273)	(0.9141, 1.0664)	-0.99809467793
10	(0.9766, 0.9727)	(1.0234, 1.0273)	(0.9570, 1.0332)	-0.99809467793
11	(0.9570, 1.0332)	(0.9668, 1.0029)	(0.9902, 1.0303)	-0.99846400321
12	(0.9902, 1.0303)	(0.9668, 1.0029)	(1.0215,0.9834)	-0.99928985350
13	(1.0215, 0.9834)	(1.0059, 1.0068)	(0.9941,0.9932)	-0.99962122552
14	(0.9941,0.9932)	(1.0000, 1.0000)	(1.0078,0.9883)	-0.99987939186
15	(1.0000, 1.0000)	(1.0039,0.9941)	(0.9971, 0.9966)	-1.00000000000

7) Doğrusal Olmayan Programlama

Değişken sayısı : 2

Kısıtların sayısı : 1

Max Z = $1.302x_1 + 0.86x_2$

Kısıtlar

$4x_1 + 2x_2 < 5$ ve $x_l>0$, $x_2>0$

Frank - Wolfe Algoritması

Başlangıç Aşikar Çözüm: (0, 0).

f(x) = $4x_1\text{-}1\,x_1^4 + 2x_2 - 1x_2^2$

df/dx, = $4\text{-}4\,x_1^3$

df/dx_2 = $2\text{-}2x_2$

k	$x^{(k-1)}$	c_1	c_2	$x_{LP}^{(k)}$	t^*	x^k
1	(0, 0)	4	2	(1.25, 0)	0.8	(1, 0)
2	(1, 0)	0	2	(0, 2.5)	0.228	(0.772, 0.569)
3	(0.772, 0.569)	2.156	0.862	(1.25, 0)	0.215	(0.875, 0.446)
4	(0.875, 0.446)	1.317	1.107	(0, 2.5)	0.074	(0.81,0.598)
5	(0.81,0.598)	1.87	0.803	(1.25, 0)	0.145	(0.874, 0.512)
6	(0.874, 0.512)	1.329	0.976	(0, 2.5)	0.054	(0.827, 0.62)
7	(0.827, 0.62)	1.74	0.761	(1.25, 0)	0.113	(0.875, 0.549)
8	(0.875, 0.549)	1.323	0.902	(0, 2.5)	0.043	(0.837, 0.633)
9	(0.837, 0.633)	1.652	0.735	(1.25, 0)	0.096	(0.877, 0.572)
10	(0.877, 0.572)	1.302	0.856	(0, 2.5)	0.035	(0.846, 0.64)

Sonuç Çözüm: (0.8461, 0.6398)

8) Doğrusal Olmayan Programlama

Değişken sayısı : 2

Kısıtların sayısı: 2

Min Z = $-49.3x_1 - 49.3x_2$

Kısıtlar

$1x_{:} + 1x_2 \leq <6$

$-2x_1 + 3x_2 \leq < 3$

ve $x_1 > 0$, $x_2 > 0$

Frank - Wolfe Algoritması

Başlangıç Aşikar Çözüm: (0, 0).

f(x) = $1x_{31} + 1x_1^2 + 4x_2^3 - 8x_1^2 - 19x_1x_2 - 6x_2^2 - 9x_1 - 18x_2$

df/dx_1 = $3x_1^2 + 2x_1x_2 - 16x_1 - 19x_2 - 9$

df/dx_2 = $1x_1^2 + 12_2^2 - 19x_1 - 12x_2 - 18$

k	$x^{(k-1)}$	c_1	c_2	$x_{LP}^{(k)}$	t*	x^k
1	(0, 0)	-9	-18	(3, 3)	1	(3, 3)
2	(3, 3)	-69	6	(6, 0)	0.301	(3.902, 2.098)
3	(3.902, 2.098)	^9.3	-49.3	(6, 0)	0	(3.902, 2.098)
4	(3.902, 2.098)	-49.3	-49.3	(6, 0)	0	(3.902, 2.098)

Sonuç Çözüm: (3.9017, 2.0983)

9) Otomatik Bir-Boyutlu Arama İşlemi

Max f(x) : $-16x^6 + 48x^5 - 61x^4 + 42x^3 - 16.5x^2 + 3.5x$

Alt Sınır : -l

Üst Sınır : 4 Hata Toleransı : 0.08

Yaklaşık Çözüm : x = 0.48438

f(x) = 0.3125

10) Otomatik Bir-Boyutlu Arama İşlemi

Min f(x) : $-16x^6 + 48x^5 - 61x^4 + 42x^3 - 16.5x^2 + 3.5x$

Alt Sınır : -l

Üst Sınır : 4

Hata Toleransı: 0.08

Yaklaşık Çözüm : x = 3.92188

f(x) = -25823

11) Ardışık Kısıtsız Minimizasyon Tekniği

Alt Sınır Kısıtlardaki Değişkenlerin Sayısı : 2

Alt Sınır Kısıtsız Değişkenlerin Sayısı : 0

Eşitsizlik Kısıtların Sayısı : 1

Eşitlik Kısıtların Sayısı : 0

P(x, r) = f(x)

$$P(x,r)=f(x) \quad -\frac{r}{B_1(x)} - \frac{r}{L_1} - \frac{r}{L_1(x)}$$

k	r	x_1	x_2	f(x)
0		0.25	0.25	1.6719
1	1	0.343	0.357	2.29
2	0.01	0.322	0.619	3.023
3	0.0001	0.331	0.663	3.1689

4	0.000001	0.333	0.666	3.1836
5	0.0000000	0.334	0.666	3.185
6	0	0.334	0.666	3.1852

F(x) $=3x_1+4x_2-x_1^3-x_2^2$

$B_1(x)$ $=1-x_1-x_2$

$L_1(x)$ $=1x_1$

$L_2(x)$ $=1x_2$

12) Ardışık Kısıtsız Minimizasyon Tekniği

Alt Sınır Kısıtlardaki Değişkenlerin Sayısı 2

Alt Sınır Kısıtsız Değişkenlerin Sayısı 0

Eşitsizlik Kısıtların Sayısı 2

Eşitlik Kısıtların Sayısı 0

$$P(x, r) = f(x) \quad -\frac{r}{B_1(x)} - \frac{r}{B_2(x)} - \frac{r}{L_1(x)} - \frac{r}{L_2(x)}$$

Burada;

f(x) $=3x_1x_2 + 40x_1 + 30x_2 - 4x_1^2 - x_1^4 - 3x_2^2 - x_2^4$

$B_1(x)$ $=12-4x_1-3x_2$

$B_2(x)$ $=4 - 1x_1 - 2x_2$

Lj(x) $=1x_1$

L2(x) $=Ix_2$

Ardışık Kısıtsız Minimizasyon Tekniği Çözümü

k	r	X_1	X_2	f(x)
0		0.5	0.5	33.875
1	1	1.629	1.043	79.445
2	0.1	1.669	1.119	81.697
3	0.01	1.679	1.146	82.413
4	0.001	1.683	1.154	82.639
5	0.0001	1.684	1.157	82.711
6	0.00001	1.684	1.157	82.733

15) Doğrusal Olmayan Programlama

Değişken sayısı : 3

Kısıtların sayısı: l

Min Z = $15.18x_1 + 14.8x_2 + 14.4x_3$

Kısıtlar

$1x_1 + 1x_2 + 1x_3 = 5$

ve

$x_1 > 0$, $x_2 > 0$, $x_3 \leq > 0$ Frank - Wolfe Algoritması Başlangıç Aşikar Çözüm: (l, 2, 2).

$f(x) = 1x_1^3 + 4x_2^2 + 16x_3$

$df/dx_1 = 3x_1^2$

$df/dx_2 = 8x_2$

$df/dx_3 = 16x_3$

k	$X^{(k-1)}$	c_1	c_2	c_3	$x_{LP}^{(k)}$	t^*	x^k	Adım Uzunluğu
1	(1, 2, 2)	3	16	32	(5, 0, 0)	0.28	(2.139, 1.431, 1.431)	1.394576
2	(2.14, 1.43,	13.7	11.4	22.9	(0, 5, 0)	0.07	(1.985, 1.687, 1.328)	0.316008
3	(1.99, 1.69,	11.8	13.5	21.3	(5, 0, 0)	0.06	(2.172,1.583,1.245)	0.229508
4	(2.17, 1.58,	14.2	12.7	19.9	(0, 5, 0)	0.05	(2.067, 1.748, 1.185)	0.204573
5	1.25) (2.07,	12.8	14	19	(5, 0, 0)	0.04	(2.196, 1.671, 1.133)	0.158978
6	(2.2, 1.67,	14.5	13.4	18.1	(0, 5, 0)	0.04	(2.117, 1.79,1.092)	0.148603
7	(2.12, 1.79,	13.4	14.3	17.5	(5, 0, 0)	0.03	(2.21,1.733,1.057)	0.114556
8	(2.21, 1.73,	14.7	13.9	16.9	(0, 5, 0)	0.03	(2.15,1.821,1.028)	0.110386
9	(2.15, 1.82,	13.9	14.6	16.5	(5, 0, 0)	0.03	(2.225, 1.774, 1.002)	0.092250
10	(2.22, 1.77, 1)	14.8	14.2	16	(0, 5, 0)	0.02	(2.177,1.843, 0.98)	0.086885
11	(2.18,1.84,0.98)	14.2	14.7	15.7	(5, 0, 0)	0.02	(2.234,1.806, 0.96)	0.070838
12	(2.23,1.81,0.96)	15	14.4	15.4	(0, 5, 0)	0.02	(2.194, 1.862, 0.943)	0.070887
13	(2.19,1.86,0.94)	14.4	14.9	15.1	(5, 0, 0)	0.02	(2.242,1.831,0.928)	0.059076
14	(2.24, 1.83,	15.1	14.7	14.8	(0, 5, 0)	0.02	(2.208,1.879,0.913)	0.060704
15	(2.21,1.88,0.91)	14.6	15	14.6	(5, 0, 0)	0.01	(2.249, 1.851, 0.9)	0.051323
16	(2.25,1.85, 0.9)	15.2	14.8	14.4	(0, 0, 5)	0	(2.249, 1.851, 0.9)	0.000000
17	(2.25,1.85, 0.9)	15.2	14.8	14.4	(0, 0, 5)	0	(2.249, 1.851, 0.9)	0.000006

Sonuç Çözüm: (2.2493, 1.8508, 0.8998).

16) Doğrusal Olmayan Programlama

Değişken sayısı : 2

Kısıtların sayısı : 2

Max Z= $0.01x_1 + 0x_2$

Kısıtlar

$1x_1 + 3x_2 \leq 8$

$5x + 2x_2 \leq 14$

Frank - Wolfe Algoritması

Başlangıç Aşikar Çözüm: (0.25, 0.25).

$f(x) = 4xj + 6x_2 - 1x_1^3 - 2x_2^2$

$df/dx_1 = 4 - 3x_1^2$

$df/dx_2 = 6\text{-}4x_2$

k	$x^{(k-1)}$	c_1	c_2	$x_{LP}^{(k)}$	t*	x^k
1	(0.25, 0.25)	3.813	5	(2, 2)	0.587	(1.277, 1.277)
2	(1.277, 1.277)	-0.89	0.892	(0, 2.667)	0.122	(1.121, 1.447)
3	(1.121, 1.447)	0.229	0.214	(2, 2)	0.049	(1.164, 1.474)
4	(1.164, 1.474)	-0.07	0.105	(0, 2.667)	0.013	(1.149, 1.49)
5	(1.149, 1.49)	0.041	0.042	(2, 2)	0.009	(1.156, 1.494)
6	(1.156, 1.494)	-0.01	0.024	(0, 2.667)	0.003	(1.153, 1.497)
7	(1.153, 1.497)	0.009	0.011	(2, 2)	0.003	(1.156, 1.498)
8	(1.156, 1.498)	-0.01	0.006	(0, 2.667)	0.001	(1.154, 1.5)
9	(1.154, 1.5)	0.004	0	(2.8, 0)	0	(1.155, 1.5)
10	(1.155, 1.5)	0.001	0.002	(2, 2)	0	(1.155, 1.5)

Sonuç Çözüm: (1.1545, 1.4996).

17) Etkileşimli Gradient Arama Sistemi

Başlangıç Aşikar Çözüm: $(x_1 , x_2) = (0, 0)$

Max $f(x_1, x_2) = 6x_1 + 2x_1x_2 - 2x_2\text{-}2x_1^2 - 1x_2^2$

$df/dx_1 = 6 + 2x_2 - 4x_1$

$df/dx_2 = 2x_1\text{-}2\text{-}2x_2$

It.	x'	Grad f(x')	x'+t[grad f(x')]	t*	x' + t [grad f]
1	(0, 0)	(6, -2)	(o+ 6t.. 0- 2t)	0.2	(1.2, -0.4)
2	(1.2, -0.4)	(0.4, 1.2)	(1.2+ 0.4t, -0.4+ 1.2t)	1	(1.6, 0.8)
3	(1.6, 0.8)	(1.2, -0.4)	(1.6+ 1.2t, 0.8- 0.4 t)	0.2	(1.84, 0.72)

4	(1.84, 0.72)	(0.08, 0.24)	(1.84+ 0.08t, 0.724 0.24t)	1	(1.92, 0.96)
5	(1.92, 0.96)	(0.24, -0.08)	(1.924 0.24t, 0.96- 0.05t)	0.2	(1.968,0.944)
6	(1.968, 0.944)	(0.016, 0.048)	(1.968+ 0.02t, 0.9444 0.05t)	1	(1.984, 0.992)
7	(1.984, 0.992)	(0.048, -0.02)	(1.984+ 0.05t 0.092- 0.02t)	0.189	(1.993,0.988)

Sonuç Çözüm: $(x_1 , x_2) = (1.984, 0.992)$

18) Etkileşimli Gradient Arama Sistemi

Başlangıç Aşikar Çözüm: $(x_1, x_2) = (0, 0)$

Max f(xj , x_2) = $2x_1x_2 + 8x_2 - 12x_1 - 2x_1^2 - 1x_2^2$

df/dx_1 = $2x_2 - 12 - 4x_1$

df/dx_2 = $2x_1 + 8 – 2x_2$

It.	x’	grad f(x’)	x' + t [grad f(x')]	t*	x' + t [grad f]
1	(0, 0)	(-12, 8)	(0 - 12t, 0+ 8t)	0.191	(-2.29, 1.529)
2	(-2.29, 1.529)	(0.235, 0.353)	(-2.29+ 0.23t, 1.529+ 0.35t)	1.294	(-1.99, 1.986)
3	(-1.99, 1.986)	(-0.08, 0.057)	(-1.99- O.OSt 1.986+ 0.06t)	0.162	(-2, 1.995)
4	(-2, 1.995)	(0.02, 0.003)	(-24 Ot, 1.995+ Ot)	1	(-2, 1.998)
5	(-2, 1.998)	(-0.01, 0.008)	(-2- 0.011, 1.998+ 0.0lt)	0.085	(-2, 1.999)
6	(-2, 1.999)	(0.001, 0.001)	(-24 0t, 1.999+ 0t)	0	(-2, 1.999)
7	(-2, 1.999)	(-0, 0.002)	(-2+ 0t, 1.999+ 0t)	0.5	(-2, 2)

Sonuç Çözüm: $(x_l , x_2) = (-2, 1.999)$

19) Etkileşimli Gradient Arama Sistemi

Başlangıç Aşikar Çözüm: $(x_1 , x_2) = (2, 1.5)$ Max f(x_1 , x_2) = $2x_1x_2 - x_1^2 - 2x_2^2 - 12x_2$

$df/dx_1 = 2x_2-2x_1 \Rightarrow df/dx_2 = 2x_l - 4x_2 – 12$

It.	x'	Gradf(x’)	x' + t[gradf(x')]	t*	x' + l [grad f]
1	(2, 1.5)	(-1, -14)	(2- t , 1.5- 14t)	0.27	(1.73, -2.28)
2	(1.73, -2.28)	(-8.02, 0.573)	(1.73- 8.02t, -2.28+ 0.57t)	0.436	(-1.77, -2.03)
3	(-1.77, -2.03)	(-0.53, -7.41)	(-1.77- 0.531, -2.03- 7.41t)	0.27	(-1.91, -4.03)
4	(-1.91, -4.03)	(-4.24, 0.303)	(-1.91- 4.24t, -4.03+ 0.3t)	0.436	(-3.76, -3.9)
5	(-3.76, -3.9)	(-0.28, -3.92)	(-3.76- 0.28t, -3.9- 3.92t)	0.27	(.3.84, -4.96)
6	(-3.84, -4.96)	(-2.24, 0.16)	(-3.84- 2.241, -4.96+ 0.16t)	0.436	(-4.82, -4.89)
7	(-4.82, -4.89)	(-0.15, -2.07)	(-4.82- 0.15t, -4.89- 2.07t)	0.271	(-4.86, -5.45)
8	(-4.86, -5.45)	(-1.18, 0.086)	(-4.86- 1.18t, -5.45+ 0.09t)	0.435	(-5.37, -5.41)
9	(-5.37, -5.41)	(-0.08, -1.09)	(-5.37- 0.05t, -5.41- 1.09t)	0.273	(-5.39, -5.71)
10	(-5.39, -5.71)	(-0.63, 0.046)	(-5.39- 0.631, -5.71+ 0.051)	0.442	(-5.67, -5.69)

Sonuç Çözüm: $(x_1 , x_2) = (-5.39, -5.71)$