**Görev 1**

Derste verilen envanter modelindeki veriler şu şekildeydi.

*Holding cost = 10 $/#T*

*Fixed cost = 50 $*

*Demand = 10 #/T*

Verilen kodda aşağıdaki değişiklikleri yapın:

* Verilen koddaki birim zaman başına toplam maliyet fonksiyonunun türevini hesaplayın. (Kitapta Bölüm 6 -syf 23’teki yöntemi kullanın)
* Türevin sıfır olduğu nokta optimal sipariş miktarını vermektedir. Bir arama algoritması oluşturup türevin sıfır olduğu noktayı bulup, konsol ekranına “ilk yöntemle bulunan optimal sipariş miktarı …..’dır” yazdırın.
* Türev fonksiyonunu yeni bir ekrana çizdirin
* İkinci yöntem olarak birim zaman başına toplam maliyet fonksiyonunun minimum noktasını bulup optimal sipariş miktarını saptayın. Konsol ekranına “ikinci yöntemle bulunan optimal sipariş miktarı …..’dır” yazdırın.

**Cevap 1**

Hocamızın derste verdiği kodu yukarıdaki probleme dönüştürmek için aşağıdaki kod geliştirilmiştir. Birim zaman başına toplam maliyet fonksiyonunun türevi aşağıdaki şekilde hesaplanmıştır:

df = diff(c) ./ diff(q);

Burada **diff** fonksiyonu, verilen dizi içindeki ardışık elemanlar arasındaki farkı hesaplar. **c** dizisi, her sipariş miktarı için toplam maliyeti içerirken, **q** dizisi sipariş miktarlarını içermektedir. Bu nedenle, **diff(c)./diff(q)** işlemi, toplam maliyet fonksiyonunun q'ya göre türevini hesaplar. Bu işlem sonucunda, **df** dizisi elde edilir ve bu dizi, her sipariş miktarı için toplam maliyet fonksiyonunun türevini içerir.

Türevin sıfır olduğu noktalar, fonksiyonun minimum veya maksimum değer aldığı yerlerdir. Scilab'da minimum noktayı bulmak için **min** fonksiyonunu kullanabiliriz. Bu fonksiyon, verilen dizi içindeki minimum değeri ve onun indeksini verir. Verilen kod parçasında **c** dizisi toplam maliyet değerlerini içeriyor, bu nedenle **min(c)** kullanarak en küçük maliyeti bulabiliriz.

Bu minimum noktanın indeksi, yani hangi sipariş miktarına karşılık geldiği, “**q(c == min(c))(1)”** kullanarak bulunabilir. Bu ifade, **c** dizisindeki en küçük değere eşit olan ilk elemanın indeksini verir.

Hem birim zamana düşen toplam maliyet fonksiyonunun türevi hesaplanarak optimal sipariş miktarının bulunması, hem de birim zaman başına toplam maliyet fonksiyonunun minimum noktası hesaplanarak optimal sipariş miktarının bulunması için bu kodu kullanılabiliriz. İkinci yöntemde, minimum nokta **min\_cost\_index** olarak hesaplanmıştı, bu nedenle aşağıdaki kod eklenerek optimal sipariş miktarı konsol ekranına yazdırılabilir:

disp("İkinci yöntemle bulunan optimal sipariş miktarı: " + string(min\_cost\_index));

Kod aşağıdaki gibidir:

h = 10; *// tutma maliyeti*

K = 50; *// sabit maliyet*

D = 10; *// talep*

maxQ = 100; *// maksimum sipariş miktarı*

q = 1:maxQ; *// sipariş miktarı*

*// f, v ve c matrislerinin boyutları tanımlanır ve sıfırlarla doldurulur*

f = zeros(1, maxQ); *// sabit maliyet*

v = zeros(1, maxQ); *// tutma maliyeti*

c = zeros(1, maxQ); *// toplam maliyet*

*// çeşitli sipariş miktarları için maliyetlerin hesaplanması için döngü oluşturulur*

for i = 1:maxQ

f(i) = K\*D/i; *// sabit maliyet*

v(i) = h\*i/2; *// tutma maliyeti*

c(i) = f(i) + v(i); *// toplam maliyet*

end

*// minimum maliyetin indeksi hesaplanır*

min\_cost\_index = q(c == min(c))(1);

disp("Minimum maliyet: " + string(min(c)) + ", Sipariş miktarı: " + string(min\_cost\_index)); *// minimum maliyet ve sipariş miktarı ekrana yazdırılır*

*// birim zamana düşen toplam maliyet fonksiyonunun türevi hesaplanır*

df = diff(c) ./ diff(q);

*// türev fonksiyonunun sıfır olduğu nokta, optimal sipariş miktarını verir*

optimal\_index = find(df == 0);

*// türev fonksiyonunun sıfır olduğu noktanın bulunup bulunamadığını kontrol edebiliriz*

if isempty(optimal\_index)

disp("Optimal sipariş miktarı bulunamadı.");

else

optimal\_quantity = q(optimal\_index);

disp("İlk yöntemle bulunan optimal sipariş miktarı: " + string(optimal\_quantity));

end

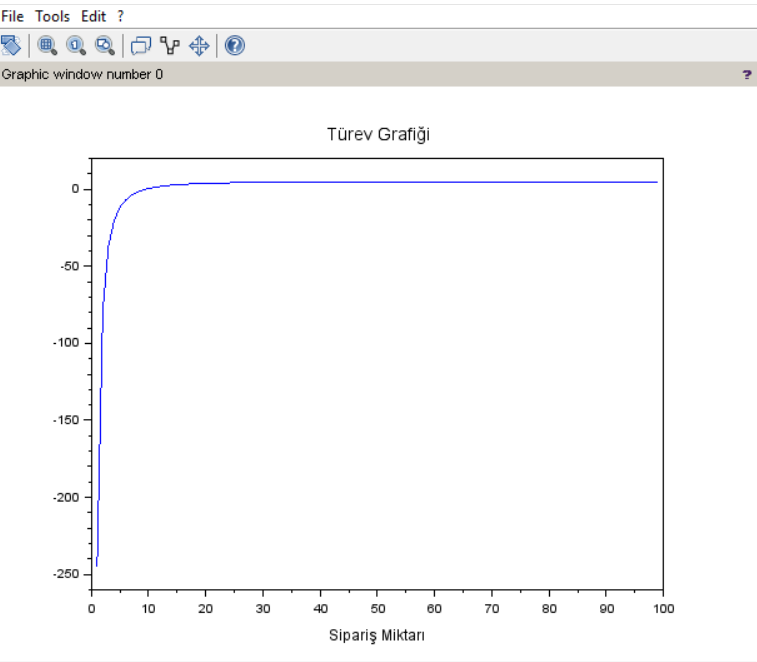
*// türev fonksiyonunun grafiği çizdirilir*

plot(q(1:maxQ-1), df);

xtitle("Türev Grafiği", "Sipariş Miktarı");

disp("İkinci yöntemle bulunan optimal sipariş miktarı: " + string(min\_cost\_index));

Yukarıdaki kodun grafik çıktısı aşağıdaki gibidir:



Kodun çıktısı aşağıdaki gibidir:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Resimde de görüleceği gibi minimum maaliyet 100 ve sipariş miktarı 10 hesaplanarak ekrana çıktı olarak verilmiştir.

İkinci yöntem ile bulunan optimal sipariş miktarı ise 10 olarak hesaplanmıştır ve çıktı olarak ekrana yazdırılmıştır. Fakat Optimal sipariş miktarı bulunamamıştır. Bunun sebebi ise türevin 0 olduğu hiçbir nokta bulunmamasıdır. Bunu yukarıdaki grafikte de rahatlıkla görebiliriz.

*// türev fonksiyonunun sıfır olduğu noktanın bulunup bulunamadığını kontrol edebiliriz*

if isempty(optimal\_index)

disp("Optimal sipariş miktarı bulunamadı.");

else

optimal\_quantity = q(optimal\_index);

disp("İlk yöntemle bulunan optimal sipariş miktarı: " + string(optimal\_quantity));

end

Bu kod parçasını yazmadığım zaman operation +: Warning adding a matrix with the empty matrix will give an empty matrix result. [] Uyarısını aldım. Bu uyarı, bir matrisin boş olduğu durumlarda diğer bir matrisle toplama işlemi yapıldığında alınır. Ben de if else yapısı ile bunun kontrolünün sağlanmasının daha doğru olabileceğini düşündüm.

Yukarıdaki kodun Pseudocode’u:

1. Sabit değişkenlerin ve değişkenlerin boyutlarının tanımlanması
2. Maliyet hesaplarının yapılması için döngü oluşturulması
3. Minimum maliyetin indeksinin hesaplanması
4. Birim zamana düşen toplam maliyet fonksiyonunun türevinin hesaplanması
5. Türev fonksiyonunun sıfır olduğu noktanın bulunması ve optimal sipariş miktarının hesaplanması
6. Türev fonksiyonunun grafiğinin çizdirilmesi
7. İkinci yöntemle bulunan optimal sipariş miktarının yazdırılması

**Görev 2**

Derste gördüğümüz sırt çantası problemini ele alalım (Bölüm 12 – syf 47). Eğer problemin birden çok çözümü varsa, hepsini yazdırmak için kodu değiştirin. Kodunuzda aşağıdaki matrisleri kullanın.

w=[90 12 12 22 14 31]

v=[1 14 14 53 53 47]

Konsol ekranında “….. optimal çözüm vardır” şeklinde yazdırılıp, alternatif çözümlerde çantaya koyulacak ürünlerin numaralarının listelenmesini sağlayın. Aşağıdaki formatta çözümler gösterilebilir.

----Sonuçlar----

….. optimal çözüm vardır.

Çözüm 1

Çantaya koyulan ürünler: ….

Toplam değer: …..

Toplam ağırlık: ……

**Cevap 2**

Derste yazılmış olan kodu bu probleme uyarlanmış hali aşağıdaki gibidir:

(Kod anlaşılır olması adına yorum satırlarıyla desteklenmiştir.)

*// Ürün ağırlıkları*

w=[90 12 12 22 14 31];

*// Ürün değerleri*

v=[1 14 14 53 53 47];

*// Çanta kapasitesi*

c=48;

*// Ürün numaraları*

n=[1 2 3 4 5 6];

*// Toplam olası çözüm sayısı*

count=2^6;

*// Tüm olası çözümlerin değerleri ve ağırlıkları*

tv=[];

tw=[];

s=zeros(count,6);

*// Tüm olası çözümlerin değerlerini ve ağırlıklarını hesapla*

for i=1:count;

k=i;

tv(i)=0;

tw(i)=0;

for j=1:6

s(i,j)=pmodulo(k,2);

k=floor(k/2);

if s(i,j) then

tw(i)=tw(i)+w(j);

tv(i)=tv(i)+v(j);

end

end

*// Ağırlık kapasitesini aşan çözümlerin değerini -1 yap*

if tw(i)>c then tv(i)=-1;

end

end

*// En iyi çözümün indeksini ve değerini bul*

best\_index=[];

best\_value=[];

sol\_count=0;

for i=1:count

if(tv(i)==max(tv)) then

best\_index=[best\_index i];

best\_value=[best\_value max(tv)];

sol\_count=sol\_count+1;

end

end

*// Tüm optimal çözümleri göster*

printf("----Sonuçlar----\n");

printf("Toplam %d adet optimal çözüm vardır.\n",sol\_count);

for k=1:sol\_count

printf("Çözüm %d:\n",k);

printf("Çantaya koyulan ürünler: ");

for j=1:6

if s(best\_index(k),j) then

printf("%d " , j)

end

end

printf("\n");

printf("Toplam değer : %d\n" , best\_value(k));

printf("Toplam ağırlık: %d\n" , tw(best\_index(k)));

end

Bu kodun çıktısı da şekildeki gibidir:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Yorum yapmak gerekirse 2 adet çözüm bulunmaktadır. Ve her çözüm için sırasıyla çantaya koyulan ürünler, toplam değer ve toplam ağırlık bilgileri listelenmiştir.

Yukarıdaki kodun Pseudocode’u:

1. Ürün ağırlıkları ve değerleri için vektörler tanımlanır.
2. Çanta kapasitesi ve ürün numaraları da dahil olmak üzere diğer gerekli değişkenler tanımlanır.
3. Tüm olası çözümlerin sayısı hesaplanır ve bu çözümler için bir dizi oluşturulur.
4. Her bir çözümün değeri ve ağırlığı hesaplanır ve bu değerler tv ve tw dizilerinde saklanır.
5. Ağırlık kapasitesini aşan çözümler tv dizisindeki değerleri -1 olarak işaretlenir.
6. En iyi çözümün indeksleri ve değerleri için boş dizi ve sayaç oluşturulur.
7. Tüm olası çözümler arasında en iyi çözümün değerleri ve indeksleri bulunur.
8. Tüm optimal çözümler ekrana yazdırılır.