**MATLABz**

Matlab programı MATrix LABoraty in ilk harfleri alınarak isimlendirilmiştir.

Mühendislik uygulamalarının, hesaplamalarının ve simülasyonlarının çoğunun gerçekleştirdiği matris ve matematik tabanlı kompleks bir programdır.

**Hangi Alanlarda Kullanılır?**

Matematik, görüntü işleme, sinyal işleme, finans, yapay zeka uygulamaları, yapay sinir ağları, veri analizi, kontrol sistemleri.

**Matlab İle Yapabileceklerimiz**

Matris işlemleri, fonksiyon ve veri çizme, algoritma geliştirme, kullanıcı arayüz oluşturma, C C++ Java gibi diğer dillerde yazılmış programlarla birlikte çalışma imkanı sağlar.

2D, 3D grafik çizimi

Çok karmaşık hesaplamaları kısa sürede tamamlar.

**Matlab Çalışma Ortamı**

Current Folder- Kullanılan Klasör

Projelerimizi uygulamalarımızı kaydettiğimiz klasördür.

Command Window- Komut Ekranı

Tek satırlık komutları Command Windowdan giriyoruz.

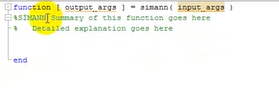
Burada satırı devam ettirmek için üç nokta (…) yazıyoruz.

Workspace- Çalışma Alanı

Command History- Komut Geçmişi

Script (Sonradan da ekleyebiliyoruz New Script ile.)

* Çok satırlı komutları fonksiyon, script kavramına giriyoruz. (C’de main gibi)
* .m uzantılı dosyalar= fonksiyon vs yazılır.
* Her fonksiyon FUNCTİON ile başlar END ile biter.



Fonksiyonun genel yapısı

* Her fonksiyonun çıktıları output..args yani çıktı argümanlarının bulunduğu yerde listelenir.
* Satırın başına % koyarsan yorum satırı olur. Matlab onu işlemlere dahil etmez. Not alabilirsin. (Javadaki // gibi sanırım.)
* Dosyanın adıyla fonksiyonun adı aynı olmalı.
* Sonu \* ile biten dosyada matlab ne kaydettiğini görmez. Yani henüz son değişiklikleri kaydetmemişsindir. Ve kaydettiğin kadarını çalıştırır.

MATLAB İLE TAVLAMA BENZETİMİ

**Tavlama Benzetimi Genel Mantık**

Tavlama benzetimi ilk başta tek bir çözümle çözüme başlıyor.

Adımları:

1) Bir noktayla çözüme başla.

2) O noktanın amaç fonksiyon değerini bul.

3) O noktanın komşusuna git.

4) Komşusunun amaç fonksiyon değerini bul.

5) Eğer geldiğin yer eskisine göre daha iyi bir değere sahipse KABUL ET

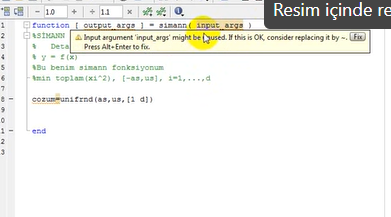
6) Geldiğin yer daha kötü bir yer ise kötüleşme miktarını bul. (Amaç fonksiyon değeri ne kadar kötüleşti?)

7) Bunu sıcaklığa böl. (O iterasyonun sıcaklığına böl.) ve buna bağlı olarak kabul olasılığı hesapla.

8) Rassal bir sayı at.

9) Bu kabul olasılığından küçükse yeni geleceğin yeri kabul et.

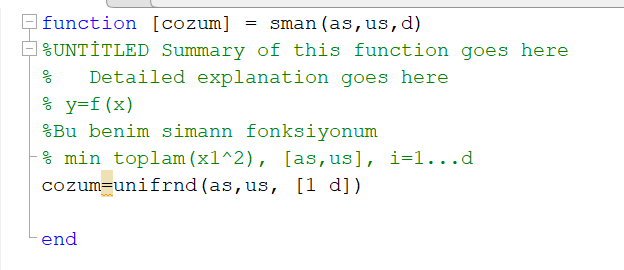
10) Attığın rassal sayı kabul olasılığından büyükse adımı eski yerine geri döndür.



cozum diye bir değişken var.

Bunun alt sınırı as, üst sınırı us

Bu bir satır vektörü 1 satırlık ve d sütunluk bir vektör



Girdiler as us d- input org

Sonuçlar (neyi görmek istiyorum?) cozum- output org

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

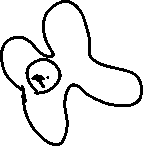
Toplam için sum fonksiyonunu kullanıyorum.

Sum(ans)

Önce amaç fonksiyon değerini bulduk. (cozum)

Karelerini aldık sum fonk topladık.

Algoritmanın Komşuluk Kısmı (3.video)



Problem ayrık olsa ve problem tamsayılar üzerinden tanımlanmış olsa 1’in komşusu 0 veya 2’dir diyebilirdik fakat problem sürekli bir problem olduğu için komşu dediğimiz yer belirsiz uzayda 2 nokta arasında da sınırsız tane nokta var. Dolayısıyla sürekli problemlerde komşuluk kullanan algoritmaları kullanabilmek için bir komşuluk büyüklüğü belirlemek gerekiyor. Bu da algoritmanın bir parametresi.

As=-5 10 birimlik bir aralık tanımı

Üs=5

Bir noktanın -,+ %1’i ya da o aralığın -,+ %1’i ya da %10’u ya da %5’i bizim komşuluğumuz içindedir.

%Komşuya git

%Komşuluk aralığı -.+ %5’liktir. (meali üs as ne kadarsa onun %5’i komşudur.)

%-%10, 3,4…%delta (benim komşuluk büyüklüğümdür ve girdilere yazılır.)

%delta kadar değiştirebilmek demek alttan %delta/2 kadar azaltmak veya %delta/2 kadar arttırmak demektir. Ya da tamamını (%delta) da alabilirsin.

Örneğin üs-as 10 br %10 komşum 1 birimlik alan içindekiler komşumdur. Bunun yarıçapı 1 birim tutabilir. Aşağıdaki 0.5le yukarıdaki 0.5 benim komşumdur denilebilir.

Problemin Boyutu

Degisim\_miktari=unifrnd(-(us-as)\*delta/2, (us-as)\*delta/2) ,[1 d];



Değişim mik alt sınır, değişim mik üst sın, değişim miktarı uygulaman gerek



Komsu=cozum+degisim\_miktari;(komşunun amaç fonk değerini hesaplaman gerekiyor)

Obj\_komsu=sum(komsu.^2);

Algoritma hatırlatması

Bir çözümle başla

Oranın amaç fonksiyon değerini hesapla

Komşuya git

Komşunun amaç fonk değerini hesapla

Eğer yeni geldiğin yer yani komşu nokta daha iyi bir çözüm değerine sahipse orayı kabul et. Daha kötü bir amaç fonk değerine sahipse orası için bir kabul olasılığı hesapla ve rassal atayacağın bir sayıya göre kabul et veya etme.

Çözümler rassal komşu da rassal bir komşu. Dolayısıyla geldi mi gelmedi mi bilmiyorum. Rassal olduğu için iyileşebilir de kötüleşebilir de.

Her if end ile biter matlabta

%Eğer geldiğim yer iyi ise, orayı çözüm olarak kabul et.

%Geldiğim yer kötü ise, kabul olasılığı hesapla.

İf(obj\_komsu<obj) (komşunun amaç fonk değeri ilk bulduğum amaç fonk değerinden küçük mü?minimize ettiğim için komşunun amaç fonk değeri küçük olması gerekiyor)

Cozum=komsu;

Obj=obj\_komsu;

Else

De=obj\_komsu\_obj;(yeni amaç fonk-eski amaç fonk)

Pa=exp(-de/T);

Rs=unifrnd(0,1);

İf(rs<pa)

Cozum=komsu;

Obj=obj\_komsu;

İf in içindekileri and veya or ile bağlamadığın sürece parantezi açık bırakabilirsin. Her türlü çalışır.

KABUL OLASILIĞI (PROBABLİTY OF ACCAPTANCE)

P (ΔE,T)=e^ -ΔE/T

ΔE/T= Kabul olasılığının bağlı olduğu 2 şey

ΔE= Amaç fonksiyonunun ne kadar kötüleştiği (Örneğin amaç fonk 80 olan bir yerden 90 olan bir yere geldiysem ΔE=10)

T= İlgili iterasyonun sıcaklığı

SICAKLIK

SA’nın başka bir parametresidir

Bir başlangıç sıcaklığı ve bir son sıcaklık vermemiz gerekiyor sisteme ve sistei yavaş yavaş soğutmamız gerekiyor. Dolayısıyla başlangıç sıcaklığını T olarak girdilere ekliyorum.

Komşunun amaç fonk değeri eski çözümümle aynıysa oradaki fark 0 olacak. Yani ΔE= 0 olacak. 0 olduğu için (-de/T) =0 gelecek= e^0=1 Kabul olasılığı 1 olacak.

Kabul olasılığının 1 olması ben burayı yeni çözümüm olarak kabul edeceğim demek.

Diyelim ki komşunun amaç fonk 90

Eski amaç fon 80 fark 10 olur.

e^-10/T

Sonra rassal sayı oluşturuyorum.

Rs=unifrnd(0,1)

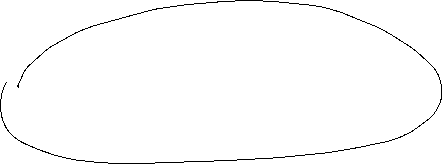
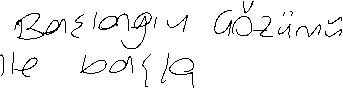
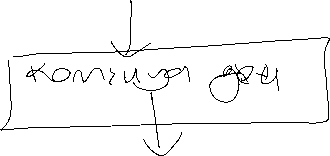
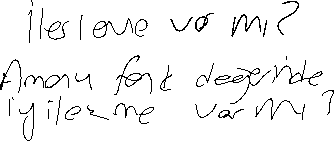
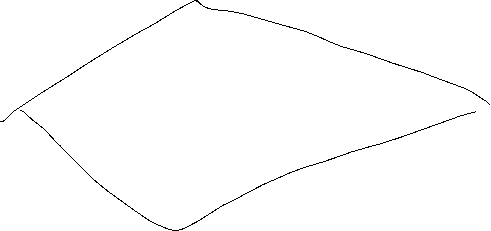
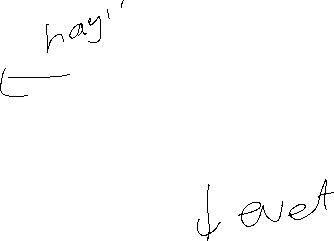
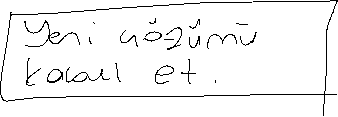
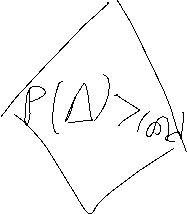
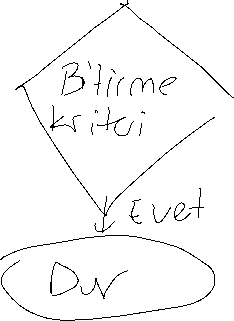
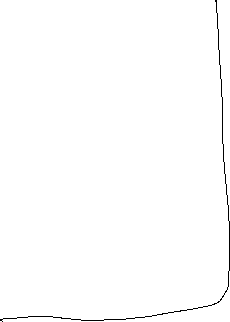
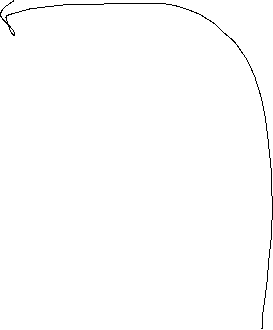
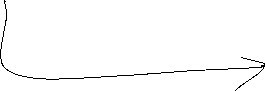
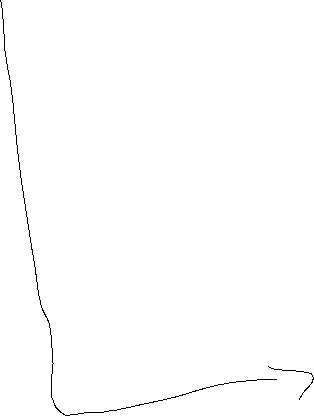
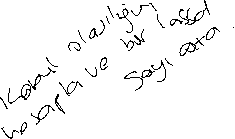
Eğer bu rassal sayı kabul olasılığından küçükse ben bu çözümü kabul edeceğim.,

Büyükse kabul etmeyeceğim. Eski noktama geri döneceğim.

Reddetme durumunda yapılacak bir şey yok. Komşuyu unut. Komşuya gerek kalmıyor.

SIMULATED ANNEALING

Akış Şeması



P(delta) 1 c çok büyük olduğunda



P(delta) 0 c çok küçük olduğunda



Rand (0,1)



Algoritma bitirme kriteri: Sağlanana kadar devam

Bu da bitiş sıcaklığı olur. Bitiş sıcaklığı da bir parametredir. (Tend) Bitiş sıcaklığı sağlanana kadar algoritmayı soğutuyoruz.

Başlangıçta T=1000 selsiyıs

Tend= 1 selsiyus

İlk iterasyonda T’miz 1000 derece sistemi yavaş yavaş soğutuyoruz ve Tmiz 1 dereceye geldiğinde 1 derecenin altına düştüğünde algoritmamız bitmiş oluyor. Bunun için de bir soğutma katsayısı sk gerekiyor.

Örneğin soğutma katsayısını 0,9 alırsak başlangıç sıcaklığı olan T’yi 1000 alırsak bir iterasyon sonra 0,9\*1000=900 oluyor. 900\*0,9= 810 derece (ikinci iterasyon) 810\*0,9= 773 derece (üçüncü iterasyon)

Bu şekilde soğutuyoruz sistemi.

Soğutma katsayısı aslında ne kadarının soğuyacağını değil de sistem sıcaklığının ne kadarının korunacağını gösterir. Yani 0,9 dersem 1 iterasyon sonra şu anki iterasyonun sıcaklığının %90’ını koru anlamına gelir.

T=T\*sk;

Bu süreci ne kadar boyunca tekrar edeceğim? Sıcaklık son sıcaklıktan düşük olana kadar. While döngüsü kullanılır. While döngüleri de end ile kapanır.

Çalıştırıyorum. Her defasında farklı sonuçlar alıyorum. Bir şey ifade etmiyor. Bunların kaç iterasyonu olduğuna ve iterasyonlar boyunca nasıl ilerlediğine bakalım

İterasyon=1

İterasyon= iterasyon+1;

Her iterasyonun çözümünü ya da amaç fonk değerini grafikleştirelim.

Objit=obj;

Objit(iterasyon)=obj;

Grafiği çizdirmek için command windowa

>> plot(objit) (grafik çizme komutu)

Bulduğu iyi noktaları kaybetmesin azalan (artmayan) bir grafik olsun istiyorum. Dolayısıyla eski iyi çözümleri tutmam gerekiyor.

6.video

Plotu düzgün çıkarmak istiyor.

Bir değişken oluşturalım. Bu değişken tüm iterasyonlar boyunca en iyi çözüm olacak.

Cozumeniyi=cozum;

Objeniyi=obj;

Objit=tüm iterasyonların çözümünü tutuyor.

İf(objit(iterasyon)<objeniyi)

End

1.iterasyonda bulduğum değer 2.iterasyonda bulduğum değerden daha büyükse yani son iterasyonda daha küçük bulmuşsam o zaman

(çıktıları güncelliyorum) Cozumeniyi=cozum;

Objeniyi=obj;

En küçük iterasyon örnekte; 36,5565

Objeniyi: 36,5565

Objit: düzeltiriz= monoton artmayacak hale getiririz.

İf( obj<min(objit))

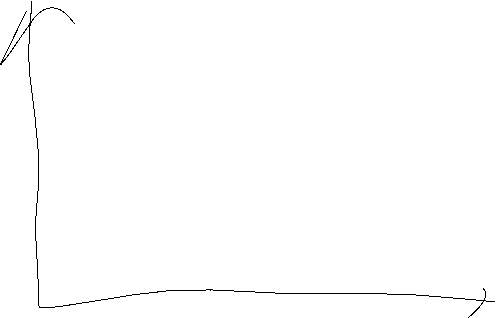
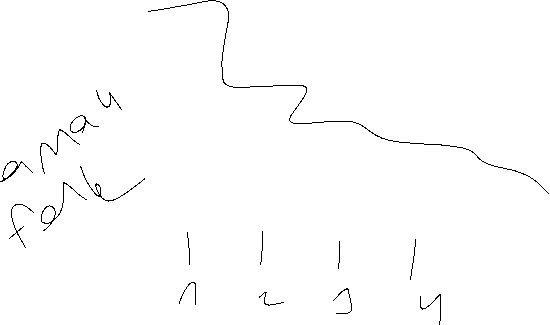
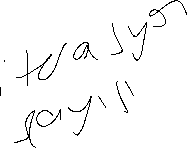
Objit(iterasyon)=obj;

Else

Objit(iterasyon)= objit(iterasyon-1);

>>plot(objit)

Bu şekilde çalıştırdığımızda azalan bir yapıya sahip.



Her iterasyonda o zamana kadarki o en iyi sonucu görmek istiyorum.

Obj=o iterasyonun çözümünün amaç fonk değeri

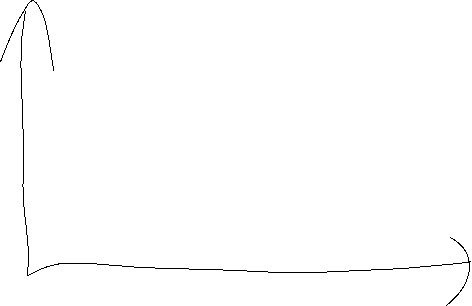
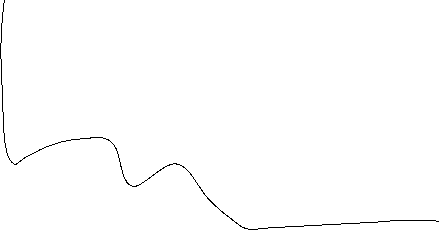
Objit= daha önceki iki iterasyonun değeri

Şimdi bulduğum obj değeri daha önceki değerlerden de daha küçükse o zaman daha iyi bir çözüm buldum.

Eğer büyükse algoritma kötüleşti ve ben o kötü yeri kabul ettim. O zaman ondan önceki iterasyona kadar olan en iyi değeri al.

Tend’i 500 dereceden 1 e çektim

>>plot(objit)



Şöyle bir yapıya sahip oldu.

Rassal sayılara ve parametrelere bağlı olarak çözümün değeri değişiyor.

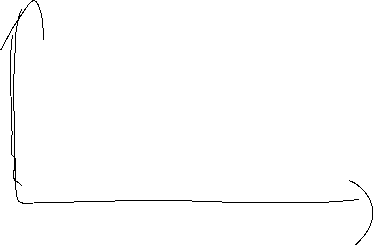
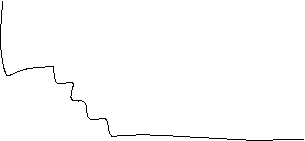
Bu 6 boyutta bir problem olduğundan problemin minimumu global optimumu olan 000000 noktasını bulamadı.

Problemin boyutu olan d değişkeninin değerini command windowda 2 yapıyorum.

>>d=2

>>sk=0,99

Parametrelerle oynayarak değerleri daha iyi hale getirebiliriz.



Soğutma katsayısını yavaş tutmak iyidir. Çok yüksek olursa da bu sefer sistem yavaş soğur. Yani algoritma biraz geç çözüme ulaşır Çok hızlı soğutursak (örn:sk=0,5) bu sefer de erken yakınsar.

En önemli parametreler soğutma katsayısı komşuluk büyüklüğü başlangıç sıcaklığı ve son sıcaklık

SA SÜREKLİ PROBLEMLER İÇİN BU ŞEKİLDEYDİ