Optimization Techniques
Section 3

M. Fatih Amasyalı

Approximating Derivatives

- In many instances, the finding f'(x) is difficult or impossible to encode. The Finite difference Newton method approximates the derivative:
- Forward difference

f'(x)≈(f(x+delta)-f(x))/ delta

- Backward difference

f'(x)≈(f(x)-f(x-delta))/ delta

- Central difference

f'(x)≈(f(x+delta/2)-f(x-delta/2))/ delta

The choice of delta matters.

bana bunu türkçe açıkla

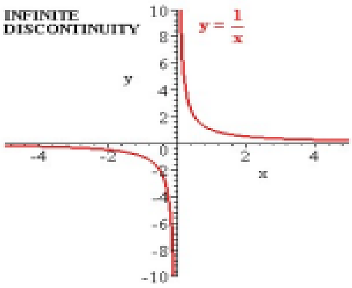
ChatGPT

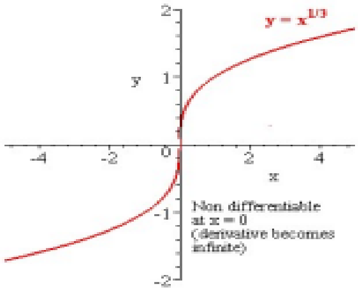
Yüklediğiniz görselde, türevlerin yaklaşık olarak hesaplanması ile ilgili bilgiler bulunmakta. Görselde yer alan bilgileri Türkçe olarak açıklayacak olursak:

- Bazı durumlarda, f'(x) yani bir fonksiyonun türevini bulmak zor veya imkansız olabilir. Sonlu fark Newton metodu, türevi yaklaşık olarak hesaplamak için kullanılır.
- İleri fark yöntemi: f'(x) ≈ (f(x+delta) - f(x)) / delta
- Geri fark yöntemi: f'(x) ≈ (f(x) - f(x-delta)) / delta
- Merkezi fark yöntemi: f'(x) ≈ (f(x+delta/2) - f(x-delta/2)) / delta
- Delta'nın seçimi önemlidir.

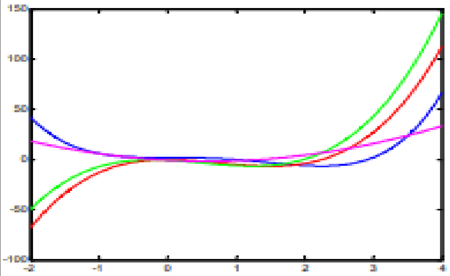
Bu bilgiler, türevin direkt olarak hesaplanamadığı durumlarda, türevin bir tahminini yapmak için kullanılan yöntemleri açıklamaktadır. Delta, kullanılan küçük bir artış miktarıdır ve hesaplanacak türevin doğruluğunu etkiler. İleri fark yöntemi, fonksiyonun belirli bir noktadaki değerinden sonra gelen değerini kullanırken; geri fark yöntemi, o noktadan önceki değeri kullanır. Merkezi fark yöntemi ise, ilgili noktanın hem öncesini hem sonrasını dikkate alır ve genellikle daha doğru sonuçlar verir.

Approximating Derivatives

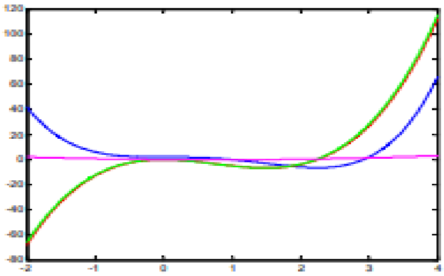




Forward difference Newton method



delta=0.5



delta=0.05

Blue: f(x)
Red: f'(x)
Green: approximated f'(x)
Magenta: (error) green-red

finite_difference_Newton.m

ChatGPT

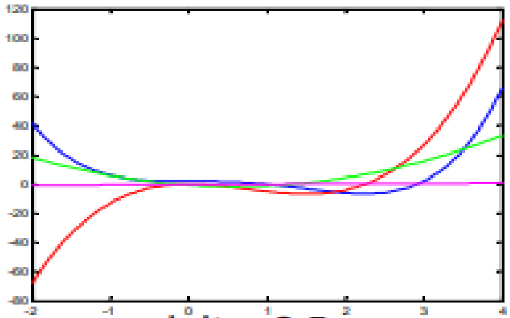
Görselde, türevlerin yaklaşık olarak hesaplanması ile ilgili iki bölüm bulunuyor:

Üst bölümde, sonsuz süreksizliğe sahip iki fonksiyonun grafiği yer alıyor. Bunlar $y = \frac{1}{x}$ ve $y = -x^3$. İlk grafikte, $x = 0$ noktasında fonksiyonun tanımsız olduğu ve burada bir sonsuz süreksizlik (kesinti) olduğu belirtilmiş. İkinci grafikte ise $y = -x^3$ fonksiyonunun $x = 0$ noktasında türevlenebilir olduğu, fakat $x = 1$ ve $x = -1$ noktalarında türevlenebilir olmadığı gösteriliyor.

Alt bölümde, İleri Fark Newton Yöntemi kullanılarak hesaplanan türevlerin delta değerlerine bağlı olarak nasıl değiştiği gösteriliyor. Grafikler, f(x) fonksiyonu ve bu fonksiyonun yaklaşık türevlerini (f'(x)) farklı delta değerleri için karşılaştırıyor: delta = 0.5 ve delta = 0.05. Her bir grafikte, orijinal fonksiyonun (mavi ile gösterilen) ve yaklaşık hesaplanan türevlerin (diğer renklerle gösterilen) grafiği çizilmiş. Delta ne kadar küçük olursa, yaklaşık türev o kadar doğru bir sonuç verir. Ancak, grafikler aynı zamanda delta'nın çok küçük olmasının (örneğin delta = 0.05) hesaplamalarda sayısal hatalara yol açabileceğini gösteriyor.

Bu görseller, türevleri analitik olarak hesaplamak mümkün olmadığında, sayısal yöntemlerle yaklaşık değerlerini bulmanın yollarını göstermektedir.

Forward difference vs. Central difference



delta=0.5

Blue: f(x)
Red: f'(x)
Green: (error) forward
Magenta: (error) central

finite_difference_Newton_2.m

Approximating higher order Derivatives

- According to the Central difference
- h=delta
- f'(x)=(f(x+h/2)-f(x-h/2))/h
- f''(x)=(f'(x+h/2)-f'(x-h/2))/h
- f'(x+h/2)=(f(x+h/2+h/2)-f(x+h/2-h/2))/h
- f'(x+h/2)=(f(x+h)-f(x))/h
- f'(x-h/2)=(f(x-h/2+h/2)-f(x-h/2-h/2))/h
- f'(x-h/2)=(f(x)-f(x-h))/h

ChatGPT

Görselde, iki farklı türev yaklaşımı olan İleri Fark ve Merkezi Fark yöntemlerinin karşılaştırılması ve daha yüksek mertebeden türevlerin nasıl yaklaşık hesaplanabileceği anlatılıyor.

Üst bölümde bir grafik gösteriliyor ve üç eğri var: mavi eğri, orijinal fonksiyonu (f(x)); kırmızı eğri, fonksiyonun türevini (f'(x)); yeşil eğri ise İleri Fark yöntemi kullanılarak hesaplanan türevin hatasını gösteriyor (hata = gerçek türev - yaklaşık türev). Delta'nın 0.5 olarak seçildiği bu grafik, İleri Fark yönteminin fonksiyonun türevini yaklaşık olarak ne kadar iyi hesaplayabildiğini gösteriyor.

Alt bölümde ise daha yüksek mertebeden türevlerin Merkezi Fark yöntemi kullanılarak nasıl yaklaşık hesaplanacağı açıklanıyor. "h" burada delta ile aynı anlamda kullanılan bir adım büyüklüğünü temsil ediyor. Verilen formüller şunlar:

- İlk türev için Merkezi Fark:
  - $f'(x) \approx (f(x+h/2) - f(x-h/2)) / h$
- İkinci türev için Merkezi Fark:
  - $f''(x) \approx (f(x+h) - 2*f(x) + f(x-h)) / h^2$
- Üçüncü ve daha yüksek mertebeden türevler için benzer şekilde, f(x) fonksiyonunun h adım büyüklüğünde ilerisindeki ve gerisindeki değerleri kullanılarak türevler yaklaşık olarak hesaplanabiliyor.

Bu yöntemler, fonksiyonun türevini doğrudan alamadığımız durumlarda, sayısal analiz teknikleri kullanarak türevin bir yaklaşık değerini bulmamıza olanak tanır. Grafikteki yeşil eğri, İleri Fark yöntemiyle hesaplanan türevin hata değerini, yani gerçek türev ile yaklaşık türev arasındaki farkı gösteriyor. Bu hatanın büyüklüğü, yaklaşımın doğruluğu hakkında bize bilgi verir.

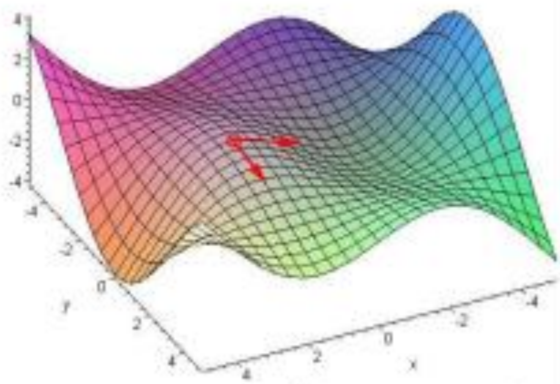
Approximating higher order Derivatives

- f''(x)=(f'(x+h/2)-f'(x-h/2))/h
- f'(x+h/2)=(f(x+h)-f(x))/h
- f'(x-h/2)=(f(x)-f(x-h))/h

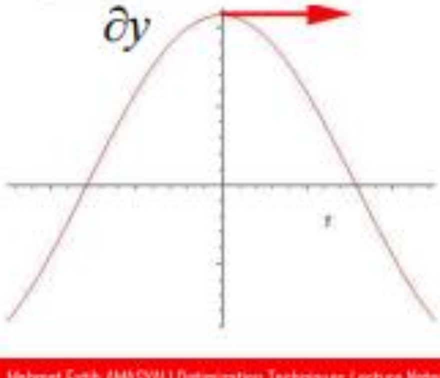
- f''(x)= (((f(x+h)-f(x))/h) - ((f(x)-f(x-h))/h)) /h
- f''(x)= (f(x+h)-2*f(x)+f(x-h)) / h^2

- See the approximating to the partial derivatives: http://en.wikipedia.org/wiki/Finite_difference

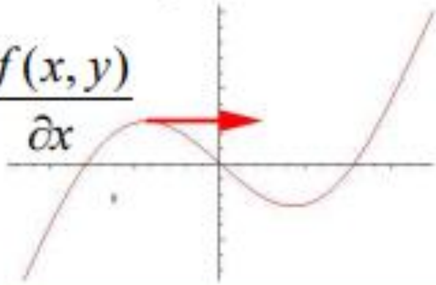
Two or more dimensions



$$\frac{\partial f(x,y)}{\partial y}$$



$$\frac{\partial f(x,y)}{\partial x}$$



Mehmet Fatih AMASYALI Optimization Techniques Lecture Notes YILDIZ TECHNICAL UNIVERSITY COMPUTER ENG DEPT.

ChatGPT

Görsel, yüksek mertebeden türevlerin ve iki ya da daha fazla boyutlu fonksiyonların türevlerinin nasıl yaklaşık olarak hesaplanacağına dair bilgiler içeriyor.

Yüksek mertebeden türevler için yaklaşık hesaplamalar aşağıdaki gibi gösteriliyor:

- İkinci mertebeden türev için Merkezi Fark:
  - $f''(x) \approx \frac{(f(x+h)-f(x-h))}{2h}$
- $f''(x+h/2)$ için:
  - $\approx \frac{(f(x+h)-f(x))}{h}$
- $f''(x-h/2)$ için:
  - $\approx \frac{(f(x)-f(x-h))}{h}$
- Üçüncü mertebeden türev için:
  - $f'''(x) \approx \frac{((f(x+h)-f(x))/h)-((f(x)-f(x-h))/h)}{h}$
- Dördüncü mertebeden türev için:
  - $f''''(x) \approx \frac{(f(x+h)-2*f(x)+f(x-h))}{h^2}$

Ayrıca, kısmi türevlerin nasıl yaklaşık hesaplanacağına dair bir Wikipedia bağlantısı veriliyor.

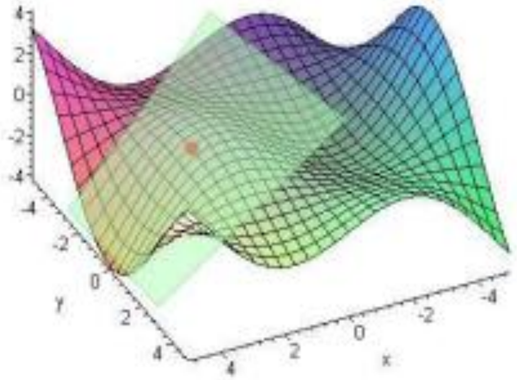
Görselin alt bölümünde, iki veya daha fazla boyutlu fonksiyonlar için kısmi türevlerin görselleştirilmesi yer alıyor. İki eksenli bir ızgarada, bir fonksiyonun yüzeyi renkli olarak gösteriliyor ve bu yüzey üzerindeki bir noktadan geçen kırmızı ve yeşil oklar, sırasıyla bu noktadaki $\frac{\partial f}{\partial x}$ ve $\frac{\partial f}{\partial y}$ kısmi türevlerin yönlerini gösteriyor. Ayrıca, bu türevlerin x ve y eksenlerine göre nasıl değiştiğini gösteren iki ayrı 2D grafik de bulunuyor.

Bu bilgiler, sayısal yöntemler kullanarak yüksek mertebeden türevlerin ve çok boyutlu fonksiyonların türevlerinin nasıl yaklaşık olarak hesaplanabileceğini gösteriyor. Bu tür yaklaşımlar, analitik çözümlerin zor ya da imkansız olduğu durumlarda kullanılır.

- Definition: The gradient of f: $R^n \rightarrow R$ is a function ∇f: $R^n \rightarrow R^n$ given by

$$\nabla f(x_1, \ldots, x_n) := \left(\frac{\partial f}{\partial x_1}, \ldots, \quad \frac{\partial f}{\partial x_n} \right)^T$$

- The gradient defines (hyper) plane approximating the function



$$\Delta z = \frac{\partial f}{\partial x} \cdot \Delta x + \frac{\partial f}{\partial y} \cdot \Delta y$$

ChatGPT

Yüklenen görselde, bir fonksiyonun gradyanı ve bu gradyanın fonksiyonu nasıl yaklaşık olarak tanımladığı açıklanıyor.

Üst kısım bir tanım içeriyor:

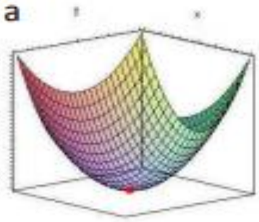
- Bir f fonksiyonunun R^n'den R'ye gradyanı, R^n'den R^n'ye giden ∇f fonksiyonu ile verilir ve şu şekilde tanımlanır:
  - ∇f(x1, ..., xn) = (∂f/∂x1, ..., ∂f/∂xn)^T
    Burada "∇f" gradyan simgesi, "∂f/∂xi" ise i'ninci değişkenle ilgili kısmi türevi ifade ediyor ve "^T" transpoze işaretini belirtiyor.

Alt kısım, gradyanın fonksiyonu nasıl bir düzlem (veya hiper-düzlem) olarak yaklaşık olarak tanımladığını gösteren bir grafik içeriyor. Fonksiyonun küçük bir değişimini (Δz) x ve y yönlerindeki değişimlerin (Δx ve Δy) ve bu değişimlere karşılık gelen kısmi türevlerin (∂f/∂x ve ∂f/∂y) çarpımı ile ifade ediyor. Bu ifade, fonksiyonun bir noktadaki eğimini veya değişim hızını gösterir ve bu gradyanın tanımladığı düzlem, fonksiyonun o noktadaki lokal doğrusal yaklaşımını temsil eder.

- Given the quadratic function

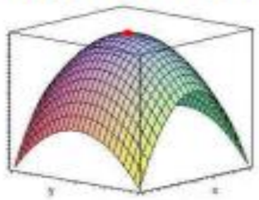
$f(x)=(1/2)\ x^T\ q\ x + b^T\ x + c$

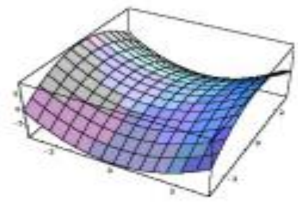
If q is positive definite, then f is a parabolic "bowl."



http://en.wikipedia.org/wiki/Positive-definite_matrix

- Two other shapes can result from the quadratic form.
  - If q is negative definite, then f is a parabolic "bowl" up side down.
  - If q is indefinite then f is a saddle.





ChatGPT

Yüklenen görselde, ikinci dereceden bir fonksiyon ve bu fonksiyonun oluşturduğu farklı şekiller anlatılıyor.

Üst kısım şöyle bir bilgi veriyor:

- Verilen ikinci dereceden fonksiyon $f(x) = \frac{1}{2}x^T \cdot Q \cdot x + b^T \cdot x + c$ şeklindedir. Eğer Q pozitif tanımlı bir matris ise, f fonksiyonu parabolik bir "kase" şeklini alır.

Alt kısım ise ikinci dereceden fonksiyonun alabileceği diğer iki şekli açıklıyor:

- Eğer Q negatif tanımlı ise, f fonksiyonu ters çevrilmiş bir parabolik "kase" şeklini alır.
- Eğer Q tanımsız ise, f fonksiyonu bir "seyir" şeklini alır, yani bir eğri yüzey oluşturur ki bu da matematikte eğri bir yüzeyin en düşük ve en yüksek noktalarının birleştiği noktada bir eyer noktasına benzer.

Görseller, bu farklı durumları gösteren fonksiyon grafiklerini içeriyor. Fonksiyonların şekilleri, Q matrisinin özelliklerine bağlı olarak değişir ve bu da grafikteki yüzeylerin farklı "kabarmaları" olarak görselleştirilir. Bu tür fonksiyonlar, özellikle optimizasyon ve makine öğrenmesi gibi alanlarda karşılaşılan problemleri çözmede önemlidir.

quadratic_functions.m

```
% quadratic functions in n dimensions

% f(x)=(1/2) * xT * q * x + bT * x + c
%f: Rn--> R
%q--> n*n
%b--> n*1
%c--> 1*1

clear all;
close all;
% n=2
q=[1 0.5; 0.5 -2];
b=[1 ;1];
c=0.5;

x1=-5:0.5:5;
x2=x1;
z=zeros(length(x1),length(x1));
for i=1:length(x1)
   for j=1:length(x2)
      x=[x1(i); x2(j)];
      z(i,j)=(1/2)*x'*q*x+b'*x+c;
   end
end

surfc(x1,x2,z)
figure;
contour(x1,x2,z)
```


quadratic_functions.m

$f(x)=(1/2)\, x^T\, q\, x + b^T\, x + c$

- $x = [x_1\ x_2\ x_3\ \dots\ x_n]^T$
- $q = [x_1^2\ x_1x_2\ x_1x_3\ \dots\ x_1x_n$
  $x_2x_1\ x_2^2\ x_2x_3\ \dots\ x_2x_n$
  …
  $x_nx_1\ x_nx_2\ x_nx_3\ \dots\ x_n^2]$
  coefficients
- $b = [x_1\ x_2\ \dots\ x_n]^T$
  coefficients
- c = constant
- $f''(x) = q$

$q = [1\ 2\ ;2\ 1]$

$b = [1\ ;\ 3]$

$c = 2$

f(x)=?

$f(x)=(x_1^2+2x_1x_2+2x_2x_1+x_2^2)/2+x_1+3x_2+2$

$f(x)=(x_1^2+4x_1x_2+x_2^2)/2+x_1+3x_2+2$

ChatGPT

Görselde MATLAB programlama dili ile yazılmış bir kod parçası ve bu kodun açıklaması bulunuyor.

Kod, n boyutlu bir uzayda ikinci dereceden bir fonksiyonu hesaplamak ve bu fonksiyonun grafiğini çizmek için yazılmış. Fonksiyon $f(x) = \frac{1}{2}x^T \cdot Q \cdot x + b^T \cdot x + c$ formülüne göre tanımlanmış. Burada x bir vektör, Q bir matris, b bir vektör ve c bir sabit.

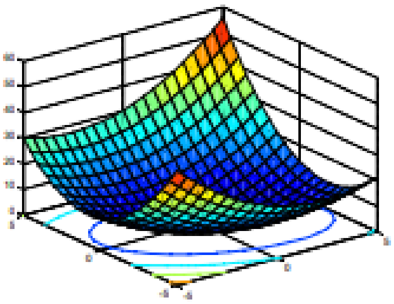
Kodun işlevleri şu adımları içeriyor:

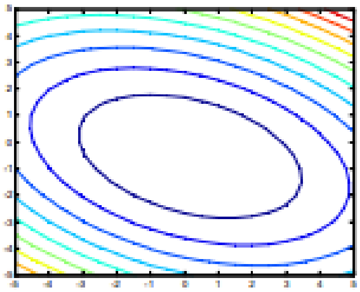
- $x1$ ve $x2$ değişkenleri için 0.5 ila 5.5 arasında bir aralık oluşturuyor ve bu aralıktaki değerler için bir ızgara (meshgrid) hazırlıyor.
- Her bir ızgara noktası için, Q, b ve c değerlerini kullanarak fonksiyonun değerini hesaplıyor.
- Elde edilen değerleri kullanarak fonksiyonun yüzey grafiğini (surf) ve eş yükseklik çizgilerini (contour) çiziyor.

Yukarıdaki açıklamada, Q, b ve c için örnek değerler verilmiş ve bu değerlerle fonksiyonun nasıl hesaplanacağı gösterilmiş. Örneğin, Q matrisi [1 2; 2 1] olarak, b vektörü [1; 3] olarak ve c sabiti 2 olarak belirlenmiş. Bu değerlerle fonksiyonun bir x vektörü için nasıl hesaplanacağı adım adım açıklanmış.

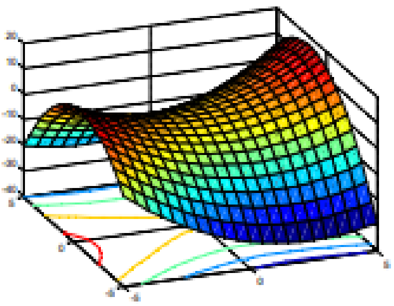
Sonuç olarak, bu kod parçası ve açıklama, çok boyutlu ikinci dereceden bir fonksiyonun MATLAB'te nasıl tanımlanıp görselleştirileceği ile ilgili bir örneği temsil ediyor.

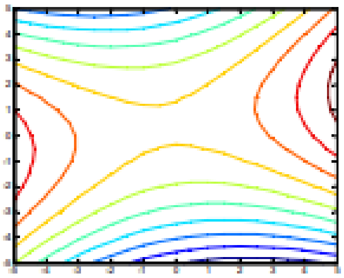
q=[1 0.5; 0.5 2]; b=[0.1 ;1]; c=0.5;





q=[1 0.5; 0.5 -2]; b=[0.1 ;1]; c=0.5;





ChatGPT

Görselde iki farklı ikinci dereceden fonksiyonun 3B yüzey ve kontur grafikleri yer almakta.

Üstteki grafiklerde $Q = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 2 \end{bmatrix}, b = \begin{bmatrix} 0.1 \\ 1 \end{bmatrix}$ ve $c = 0.5$ kullanılarak oluşturulmuş bir fonksiyonun grafikleri verilmiş. 3B yüzey grafiği, fonksiyonun minimum noktasına doğru bir "kase" şekli oluşturuyor ki bu, Q matrisinin pozitif tanımlı olduğunu gösterir. Kontur grafiğinde ise eş değerlik hatları (eş yükseklik çizgileri) daha sıkı bir şekilde merkez etrafında toplanmış, minimum noktayı çevreliyor.

Alt grafiklerde ise $Q = \begin{bmatrix} 1 & 0.5 \\ 0.5 & -2 \end{bmatrix}, b = \begin{bmatrix} 0.1 \\ 1 \end{bmatrix}$ ve $c = 0.5$ kullanılarak oluşturulmuş bir başka fonksiyonun grafikleri yer almakta. Bu 3B yüzey grafiği, Q matrisinin negatif tanımlı bir değere sahip ikinci satır ve sütunu nedeniyle bir "tepe" veya "seyir" şekli oluşturuyor. Kontur grafiğinde eş değerlik hatları bu tepe etrafında dairesel olmayan bir şekilde genişlemiş.

Her iki durumda da, kontur grafikleri fonksiyonun eğimini ve optimizasyon problemlerinde potansiyel minimum veya maksimum noktaların yerini görsel olarak anlamamızı sağlar. Bu tür grafikler, özellikle makine öğrenmesi ve optimizasyon problemlerinin çözümünde sıkça kullanılır.

$f(x_1,x_2)=x_1^2+3x_2^2+4x_1x_2+3x_2+2$

- q,b,c ?
- (½)*q = [1 4; 0 3] or [1 3; 1 3]
  or **[1 2; 2 3] (symmetric)**
  q= [2 4; 4 6]
- b = [0; 3]
- c = 2

- *Hessian* of *f* : the second derivative of f

$$\boldsymbol{F} = D^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2}(\boldsymbol{x}) & \frac{\partial^2 f}{\partial x_2 \partial x_1}(\boldsymbol{x}) & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_1}(\boldsymbol{x}) \\ \frac{\partial^2 f}{\partial x_1 \partial x_2}(\boldsymbol{x}) & \frac{\partial^2 f}{\partial x_2^2}(\boldsymbol{x}) & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_2}(\boldsymbol{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n}(\boldsymbol{x}) & \frac{\partial^2 f}{\partial x_2 \partial x_n}(\boldsymbol{x}) & \cdots & \frac{\partial^2 f}{\partial x_n^2}(\boldsymbol{x}) \end{bmatrix}$$

ChatGPT

Görselde, bir ikinci dereceden fonksiyonun ifadesi ve bu fonksiyona ait Hessian matrisinin tanımı yer almaktadır.

İlk bölümde, ikinci dereceden bir fonksiyonun genel formülü verilmiş:
$f(x_1, x_2) = x_1^2 + 3x_2 + 4x_1x_2 + x_1 + 3x_2 + 2$

Bu fonksiyon için, Q, b, ve c değerlerinin ne olduğu sorulmuş ve cevap olarak şu değerler verilmiş:

- Q matrisi simetrik olduğu için ve verilen katsayılara göre $Q = \begin{bmatrix} 2 & 4 \\ 4 & 6 \end{bmatrix}$ olarak belirlenmiş.
- b vektörü, lineer terimlerin katsayılarına bakılarak $b = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$ olarak bulunmuş.
- c sabiti ise, sabit terim olarak $c = 2$ olarak verilmiş.

İkinci bölümde ise bir fonksiyonun Hessian matrisi tanımlanmıştır. Hessian matrisi, bir fonksiyonun her bir değişken cinsinden ikinci kısmi türevlerini içeren kare bir matristir ve genelde H ile gösterilir. Matrisin her elemanı, ilgili değişkenlerin türevlerini ifade eder. Bu matris, fonksiyonun eğriliğini veya konveksliğini analiz etmek için kullanılır ve optimizasyon problemlerinde önemli bir rol oynar.

Görseldeki bilgiler, ikinci dereceden bir fonksiyonun genel formülünün ve bu fonksiyonun Hessian matrisinin nasıl oluşturulacağının bir örneğini sunmaktadır.

$f''(x) = q$

- $f(x_1,x_2)=x_1^2+3x_2^2+4x_1x_2+3x_2+2$

```
syms x1;
syms x2;
syms expr;
% diff(expr,n,v) differentiate expr n times with respect to v .
expr=x1^2+3*x2^2+4*x1*x2+3*x2+2;
ddx=diff(expr,2,x1);
dx=diff(expr,1,x1);
dy=diff(expr,1,x2);
dxdy=diff(dx,1,x2);
ddy=diff(expr,2,x2);

q=[ddx dxdy; dxdy ddy]
```

```
q =

[ 2, 4]
[ 4, 6]
```


Quadratic functions in 2 dims.

$f(x)=(1/2)\ x^T\ q\ x + b^T\ x + c \qquad x = [x_1\ x_2]^T$

q=[1 0.5; 0.5 2];

b=[-0.5 ;-0.5];

c=0.5;

f(x)=(1/2) [x1 x2][1 0.5; 0.5 2] [x1;x2] + [-0.5 -0.5] [x1;x2]+0.5

f(x)=(1/2) [x1+0.5*x2 0.5*x1+2*x2] [x1;x2] – (0.5*x1+0.5*x2)+0.5

f(x)=(1/2)(x1^2+0.5*x1*x2+0.5*x1*x2+2*x2^2) -0.5*x1-0.5*x2+0.5

f(x)=(1/2) x1^2+x1*x2+2*x2^2) -0.5*x1-0.5*x2+0.5

f(x)=(x1^2)/2+(x1*x2)/2+x2^2- 0.5*x1-0.5*x2+0.5

ChatGPT

Görselde, MATLAB kullanılarak ikinci dereceden bir fonksiyonun Hessian matrisinin nasıl hesaplanacağı gösteriliyor ve ayrıca iki boyutlu bir ikinci dereceden fonksiyonun açık formu veriliyor.

Üst kısımda, MATLAB'ta sembolik değişkenler x_1 ve x_2 oluşturulmuş ve $f(x_1, x_2) = x_1^2 + 3x_2 + 4x_1x_2 + 3x_2 + 2$ fonksiyonu için Hessian matrisi hesaplanmıştır. MATLAB'ın sembolik türev fonksiyonu 'diff' kullanılarak, fonksiyonun x_1 ve x_2 değişkenleri cinsinden birinci ve ikinci türevleri alınmış ve sonuç olarak Q matrisi elde edilmiştir. Bu matris, fonksiyonun her bir değişken cinsinden ikinci kısmi türevlerini içermekte ve $Q = \begin{bmatrix} 2 & 4 \\ 4 & 6 \end{bmatrix}$ olarak bulunmuştur.

Alt kısımda ise genel bir iki boyutlu ikinci dereceden fonksiyon $f(x) = \frac{1}{2}x^TQx + b^Tx + c$ formunda verilmiş ve örnek bir fonksiyon için Q, b ve c değerleri ile açık formda ifade edilmiştir. Örneğin, $Q = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 2 \end{bmatrix}$, $b = \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix}$ ve $c = 0.5$ alındığında, fonksiyonun açık formu $f(x_1, x_2) = \frac{1}{2}(x_1^2 + x_1x_2 + 0.5x_2^2) - 0.5x_1 - 0.5x_2 + 0.5$ olarak hesaplanmıştır.

Bu görseller, ikinci dereceden fonksiyonların ve bunların türevlerinin, özellikle Hessian matrisinin, MATLAB ortamında nasıl hesaplanacağını ve ifade edileceğini göstermektedir. Bu tür hesaplamalar, optimizasyon ve makine öğrenmesi gibi alanlarda sıkça karşılaşılan matematiksel problemlerin çözümünde kullanılır.

Quadratic functions in 2 dims.

f(x)=(1/2) x^T q x + b^T x + c $x = [x_1\ x_2]^T$

q=[1 0.5; 0.5 2];

b=[-0.5 ;-0.5];

c=0.5;

f(x)=(x1^2)/2+(x1*x2)/2+x2^2- 0.5*x1-0.5*x2+0.5

df/dx1 = x1+x2/2 -0.5

df/dx2 = x1/2+2*x2 -0.5

df=[df/dx1; df/dx2]

df/dx1x2=df/dx2x1=1/2

df/dx1x1=1

df/dx2x2=2

ddf=[df/dx1x1 df/dx1x2 ; df/dx2x1 df/dx2x2] = q

Opt. in 2 dims.

$x = [x_1\ x_2]^T$

```
% gradient decent
x_new = x_old - eps * df;
% [2,1] = [2,1] - [1,1]*[2,1]
% newton raphson
x_new = x_old - df/ddf;
x_new = x_old - inv(ddf)*df;
% [2,1] = [2,1] - [2,2]*[2,1]
```

ChatGPT

Görselde, iki boyutlu bir ikinci dereceden fonksiyonun türevleri ve bu fonksiyon üzerinde iki farklı optimizasyon tekniği kullanılarak optimum noktaya nasıl ulaşılacağı anlatılmaktadır.

Üst kısımda, $f(x) = \frac{1}{2}x^T Qx + b^T x + c$ genel formülüne sahip bir ikinci dereceden fonksiyon için özgün Q, b, ve c değerleri verilmiş ve x_1 ve x_2 değişkenlerine göre türevleri alınmıştır. Bu türevler, fonksiyonun gradyanını ve Hessian matrisini oluşturmak için kullanılmıştır. Son satırda, fonksiyonun gradyan vektörünün her iki bileşeninin de eşit olduğu ve -0.5 değerine sahip olduğu belirtilmiştir.

Alt kısımda ise bu fonksiyonun minimum noktasına ulaşmak için iki optimizasyon tekniğinden bahsedilmektedir:

1. Gradyan İnişi (Gradient Descent): Yeni bir x noktası, mevcut x noktasından, gradyanın negatif yönde bir miktar (ϵ) hareket ettirilmesiyle elde edilir. Bu yöntemde, fonksiyonun gradyanı dik bir iniş yönünde kullanılır ve bu adım adım tekrarlanarak minimum noktaya ulaşmaya çalışılır.
2. Newton-Raphson Yöntemi: Bu yöntemde, yeni bir x noktası, mevcut x noktasından, Hessian matrisinin tersinin gradyan vektörü ile çarpılması sonucu bulunur. Bu yöntem, genellikle daha hızlı yakınsama gösterir ve optimizasyon problemlerinde sıkça kullanılır.

Her iki yöntem de, bir fonksiyonun minimum noktasını bulmak için kullanılan iteratif yöntemlerdir ve belirli koşullar altında optimizasyon problemlerinin çözümünde etkilidirler.

Opt. in N dims.

% gradient decent $x = [x_1\ x_2\ x_3\ \ldots\ x_n]^T$

x_new = x_old - eps * df;

% [n,1] = [n,1] - [1,1]*[n,1]

% newton raphson

x_new = x_old - df/ddf;

x_new = x_old - inv(ddf)*df;

% [n,1] = [n,1] - [n,n]*[n,1]

Matrix inversion

- A is a square matrix (n*n)
- I is the identitiy matrix (n*n)
- $A*A^{-1}=I$
- A^{-1} is the inversion of A
- A matrix has an inverse if the determinant $|A|\neq 0$

ChatGPT

Görsel, N boyutlu uzayda optimizasyon tekniklerinin ve matris ters alma işleminin nasıl yapıldığını açıklıyor.

Üst kısımda, iki optimizasyon algoritmasının genel adımları verilmiştir:

1. Gradyan İnişi (Gradient Descent): Bu yöntemde, mevcut noktadan gradyanın negatif yönünde, belirli bir adım büyüklüğü (ϵ) ile hareket ederek yeni bir x noktası hesaplanır. Matematiksel ifade şu şekildedir:
   $x_{\text{yeni}} = x_{\text{eski}} - \epsilon \cdot \nabla f$
   Burada ∇f fonksiyonun gradyanını ifade eder.
2. Newton-Raphson Yöntemi: Bu yöntemde, Hessian matrisinin tersi alınarak gradyan vektörü ile çarpılır ve bu değer eski x değerinden çıkartılarak yeni x değeri hesaplanır. Matematiksel ifade şu şekildedir:
   $x_{\text{yeni}} = x_{\text{eski}} - \text{inv}(H) \cdot \nabla f$
   Burada $\text{inv}(H)$ Hessian matrisinin tersini ve ∇f fonksiyonun gradyanını ifade eder.

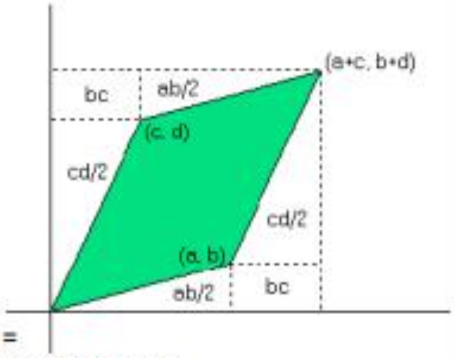
Alt kısımda ise matris ters alma işlemi ile ilgili genel bilgiler verilmiştir:

- A bir kare matristir (n*n).
- I birim matrisi ifade eder (n*n).
- $A \cdot A^{-1} = I$ ifadesi, A matrisinin tersinin A ile çarpıldığında birim matrisi verdiğini gösterir.
- Bir matrisin tersinin olabilmesi için determinantının ($|A|$) sıfırdan farklı olması gerekir.

Bu bilgiler, optimizasyon ve lineer cebirde sıkça karşılaşılan konseptlerdir ve bu algoritmalar, fonksiyonların minimum veya maksimum noktalarını bulmak için kullanılır. Matris tersi alma işlemi ise, özellikle Newton-Raphson yönteminde Hessian matrisi ile gradyan vektörünü çarparken önemlidir.

Geometric meaning of the determinant

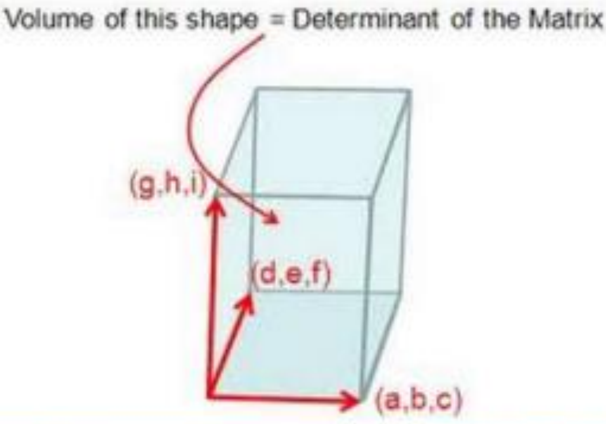
- A=[a b ; c d]
- det(A) is the area of the green parallelogram with vertices at (0,0), (*a*, *b*), (*a*+*c*, *b*+*d*), (*c*,*d*).



The area of the big rectengular=
(a+c)*(b+d)=a*b+a*d+c*d+c*b
The area of the green parallelogram =
=a*b+a*d+c*d+c*b – 2*c*b -2*(a*b)/2 -2*(d*c)/2
=a*d-c*b

Geometric meaning of the determinant

- In 3 dimensions:



ChatGPT

Görselde, determinantın geometrik anlamı iki ve üç boyutlu uzayda görselleştirilmiştir.

Üst kısım:
İki boyutlu bir uzayda, $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ matrisinin determinantı $\det(A)$, köşe noktaları (0,0), (a,b), (a+c, b+d) ve (c,d) olan yeşil paralelkenarın alanıdır. Büyük dikdörtgenin alanı $(a+c) \times (b+d)$ olarak hesaplanır ve bu dikdörtgenin içinde yer alan yeşil paralelkenarın alanı $ab + cd - bc + ad = ad - bc$ olarak bulunur. Bu da iki boyutlu bir uzayda matrisin determinantının, oluşturduğu vektörlerle sınırlanmış paralelkenarın alanına eşit olduğunu gösterir.

Alt kısım:
Üç boyutlu uzayda ise, bir matrisin determinantı, oluşturduğu vektörlerle sınırlanmış paralelyüzeyin (3B paralelkenar) hacmine eşittir. Görselde, bu hacim kırmızı bir şekil olarak gösterilmiştir ve bu şeklin hacmi matrisin determinantına eşittir. Bu da üç boyutlu bir uzayda matrisin determinantının, oluşturduğu vektörlerle sınırlanmış paralelyüzeyin hacmini nasıl temsil ettiğini gösterir.

Her iki durumda da, determinantın geometrik anlamı, matrisin sütunları tarafından oluşturulan şekillerin (paralelkenar veya paralelyüzey) ölçüsünü verir. Bu, lineer cebirde ve çok boyutlu uzayda vektörlerin oluşturduğu şekilleri anlamak için kullanılan temel bir kavramdır.

Matrix inversion

- For a 2*2 matrix (A)= [a b ; c d]
- A^{-1} =[e f ; g h]
- det(A)=a*d-c*b
- [a b ; c d]*[e f ; g h]= [1 0 ; 0 1]
- a*e+b*g=1
- a*f+b*h=0
- c*e+d*g=0
- c*f+d*h=1

a*f=-b*h
f=-(b*h)/a
-(c*b*h)/a+d*h=1
h*(d-(c*b)/a)=1
h=a/(a*d-c*b)
h=a/det(A)

Matrix inversion

For a 2×2 matrix

$$A \equiv \begin{bmatrix} a & b \\ c & d \end{bmatrix},$$

the matrix inverse is

$$A^{-1} = \frac{1}{|A|} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$
$$= \frac{1}{a\,d - b\,c} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix},$$

h=a/det(A)

ChatGPT

Görselde, 2×2'lik bir matrisin tersinin nasıl alınacağı ve bu işlemin matematiksel formülasyonu gösteriliyor.

Üst kısımda, $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ formundaki bir matrisin tersi $A^{-1} = \begin{bmatrix} e & f \\ g & h \end{bmatrix}$ olarak ifade edilmiş ve bu ters matrisin elemanlarının nasıl hesaplanacağı aşama aşama açıklanmıştır. Detaylar şöyle:

- Ters matrisin birim matrisle çarpımı birim matrisi verecektir, yani $\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.
- Bu denklem sistemini çözerek e, f, g, h değerleri bulunur.
- Denklem sistemi şu dört denklemle ifade edilir:
  - $ae + bg = 1$
  - $af + bh = 0$
  - $ce + dg = 0$
  - $cf + dh = 1$
- Buradan e, f, g, h değerleri için çözümler şu şekilde verilmiş:
  - $f = -\frac{b}{a}$ (Eğer $a \neq 0$ ise)
  - $e = \frac{1}{a}$ (Yukarıdaki koşul dahilinde)
  - $g = -\frac{c}{a}$ (Eğer $a \neq 0$ ise ve $ad - bc \neq 0$)
  - $h = \frac{a}{ad-bc}$ (Determinant sıfır değilse)

Alt kısımda ise genel bir 2×2 matrisin tersinin formülü verilmiş:

- Bir $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ matrisinin tersi $A^{-1} = \frac{1}{ad-bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$ şeklinde hesaplanır.
- Bu formülde, $\frac{1}{ad-bc}$ ifadesi ters matrisin her elemanını normalize eden bir çarpan olarak işlev görür ve bu çarpan matrisin determinantının tersidir.

Görseller, 2×2 matrisin tersini almanın temel prensiplerini ve bu işlemin nasıl gerçekleştirileceğini göstermektedir. Matrisin tersi alınabilmesi için determinantının (bu örnekte $ad - bc$) sıfırdan farklı olması gerektiğini vurgulamaktadır. Bu, lineer cebirde temel bir kavramdır ve sistemlerin çözümlenmesi, matris denklemlerinin analizi gibi birçok alanda kullanılır.

Matrix inversion

- For a 3×3 matrix the inverse may be written as:

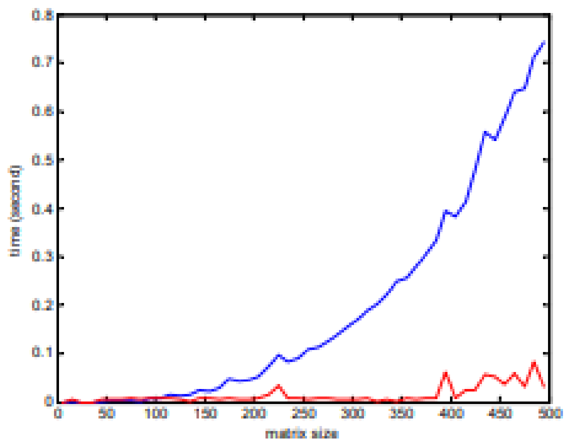
$$\mathbf{A} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

$$\mathbf{A}^{-1} = \frac{\begin{bmatrix} ei-fh & hc-ib & bf-ce \\ gf-di & ai-gc & dc-af \\ dh-ge & gb-ah & ae-db \end{bmatrix}}{|\mathbf{A}|} = \frac{\begin{bmatrix} ei-fh & hc-ib & bf-ce \\ gf-di & ai-gc & dc-af \\ dh-ge & gb-ah & ae-db \end{bmatrix}}{aei+bfg+cdh-gec-hfa-idb}$$

A general n*n matrix can be inverted using methods such as the Gauss-Jordan elimination, Gauss elimination or LU decomposition.

The cost of Matrix inversion

- inversion_time.m



Blue mean time
Red std time

ChatGPT

Görselde, bir 3×3 matrisin tersinin nasıl alınacağı ve n boyutlu bir matris için ters alma yöntemleri anlatılmıştır. Ayrıca, matris ters alma işleminin hesaplama maliyetini gösteren bir grafik bulunmaktadır.

Üst Kısım:

Bir 3×3 matris $A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$ için ters matris A^{-1}, matrisin determinantı $|A|$ ile normalize edilmiş kofaktörlerin transpozezi alınarak hesaplanmaktadır. Bu, her bir elemanın kendi minörünün (elemanın kendisi dışındaki matrisin determinantı) ve işaretinin (-1)^(satır+ sütun) ile hesaplanmış değerinin determinant ile bölünmesiyle yapılır. Sonuçtaki ifade A^{-1} matrisinin elemanlarını vermektedir.

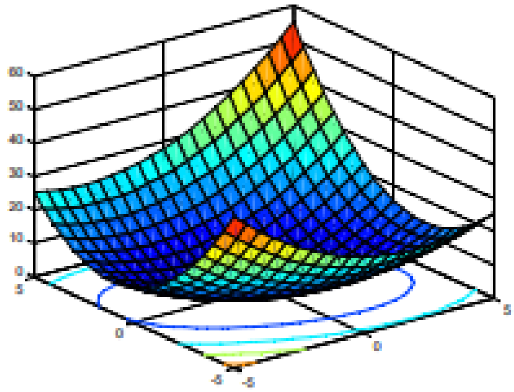
Ayrıca, genel bir $n \times n$ matrisinin tersi alınırken Gauss-Jordan eliminasyonu, Gauss eliminasyonu veya LU ayrıştırması gibi yöntemler kullanılabilir.

Alt Kısım:

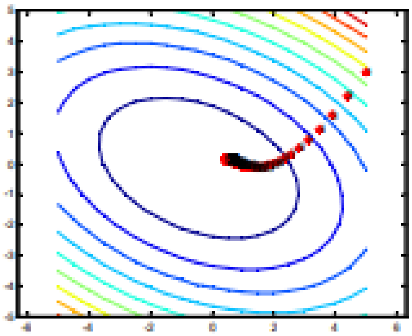
Matris ters alma işleminin hesaplama maliyetini gösteren bir grafik bulunmaktadır. Grafikte, matris boyutunun artışına göre hesaplama süresindeki değişim mavi renk ile gösterilmiş, ortalama süre ise kırmızı renk ile ifade edilmiştir. Grafikten görüldüğü üzere, matris boyutu büyüdükçe ters alma işleminin maliyeti (hesaplama süresi) da artmaktadır.

Bu grafik, matris ters alma işleminin özellikle büyük boyutlu matrislerde oldukça maliyetli bir hesaplama işlemi olduğunu ve bu işlemin süresinin matris boyutu ile yakından ilişkili olduğunu göstermektedir. Bu nedenle büyük matrislerle çalışırken, ters alma işleminin optimize edilmesi veya mümkünse alternatif yöntemlerin kullanılması önem kazanır.

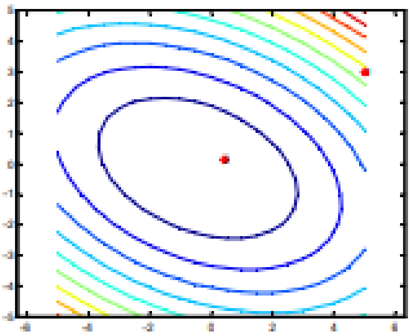
q=[1 0.5; 0.5 2];
b=[-0.5 ;-0.5];
c=0.5;



Gradient Descent
stepsize=0.1



Newton Raphson



opt_Ndim.m

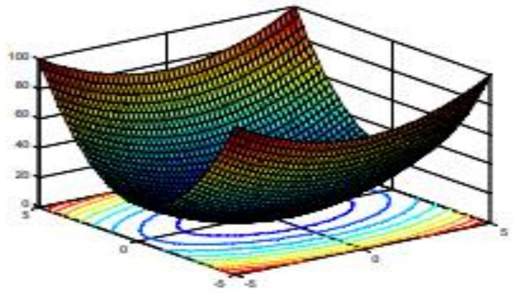
ChatGPT

Görselde, bir ikinci dereceden fonksiyonun 3 boyutlu yüzey grafiği ve bu fonksiyon üzerinde uygulanan iki farklı optimizasyon yöntemi olan Gradyan İnişi ve Newton-Raphson yöntemlerinin 2 boyutlu kontur grafikleri yer alıyor.

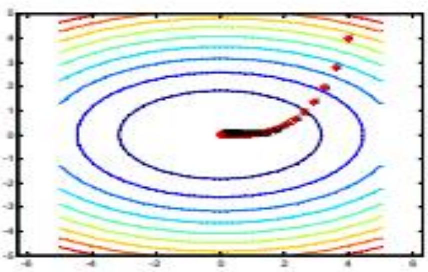
Üst kısımda, $Q = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 2 \end{bmatrix}$, $b = \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix}$ ve $c = 0.5$ değerleri ile tanımlanmış bir ikinci dereceden fonksiyonun 3D yüzey grafiği gösteriliyor. Bu grafik fonksiyonun en düşük değerinin (minimumunun) nerede olduğunu görselleştiriyor.

Alt kısımda ise Gradyan İnişi ve Newton-Raphson yöntemlerinin optimizasyon süreçleri kontur grafikleri üzerinde adım adım gösteriliyor. Her iki grafikte de kırmızı çizgiler algoritmanın her adımda izlediği yolu ve mavi çizgiler de fonksiyonun eş yükseklik hatlarını (konturlarını) temsil ediyor. Gradyan İnişi yönteminde adım boyutu (step size) 0.1 olarak belirlenmiş ve bu yöntemde algoritma, en dik iniş yönünde adım adım ilerleyerek minimuma ulaşmaya çalışıyor. Newton-Raphson yönteminde ise, her adımda fonksiyonun Hessian matrisinin tersi kullanılarak daha hızlı bir yakınsama sağlanıyor ve algoritma daha az adımda minimuma ulaşıyor gibi görünüyor. Her iki yöntemin amacı, fonksiyonun minimum noktasını (bu durumda merkezdeki kırmızı nokta) bulmaktır.

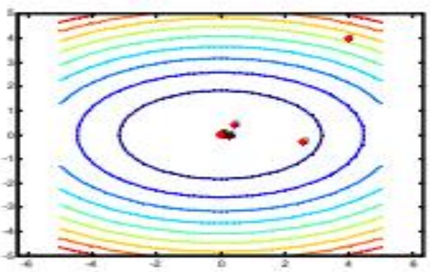
Find the minimum of
f(x1,x2)=(x1*x1)+(3*x2*x2)



Gradient Descent
stepsize=0.05
do not converged at
50 iteration



Steepest Descent
converged at the 12th
iteration
Attention to
orthogonal updates



steepest_desc_2dim.m

ChatGPT

Görselde, $f(x_1, x_2) = (x_1^2 + x_1) + (3 \times x_2^2 + x_2)$ fonksiyonunun minimumunun nasıl bulunabileceği ile ilgili bilgiler bulunuyor.

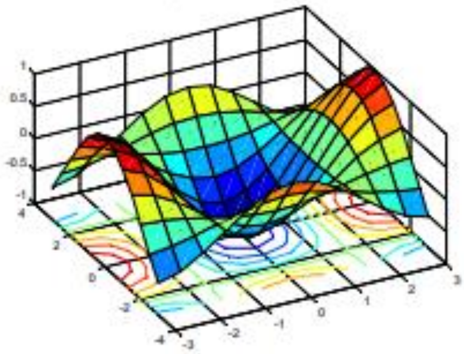
Üst kısımda, bu ikinci dereceden fonksiyonun 3 boyutlu yüzey grafiği gösteriliyor. Grafik, fonksiyonun açık bir şekilde en düşük değerinin olduğu bir çukuru temsil ediyor.

Alt kısımda ise Gradyan İnişi ve En Dik İniş (Steepest Descent) yöntemleri kullanılarak optimizasyon süreçlerinin kontur grafikleri üzerindeki gösterimleri yer alıyor. Gradyan İnişi için adım boyutu (step size) 0.05 olarak ayarlanmış ancak bu yöntem 50 iterasyon sonra hala yakınsama göstermemiş. Diğer yandan, En Dik İniş yöntemi 12. iterasyonda yakınsamış ve dikkat edilmesi gereken nokta ise algoritmanın orthogonal (dik) güncellemeler yapmasıdır. Her iki kontur grafiğinde de kırmızı çizgiler, her iterasyonda algoritmanın takip ettiği yolu gösterirken, mavi çizgiler fonksiyonun eş değer seviyelerini (yani eş yükseklik çizgilerini) temsil ediyor. Ortadaki kırmızı nokta, fonksiyonun minimum noktasını işaret ediyor.

Bu görsel, Gradyan İnişi ve En Dik İniş yöntemlerinin pratikte nasıl farklı sonuçlar verebileceğini ve özellikle adım boyutunun algoritmanın performansı üzerinde ne kadar önemli olabileceğini gösteriyor.

Griewank function

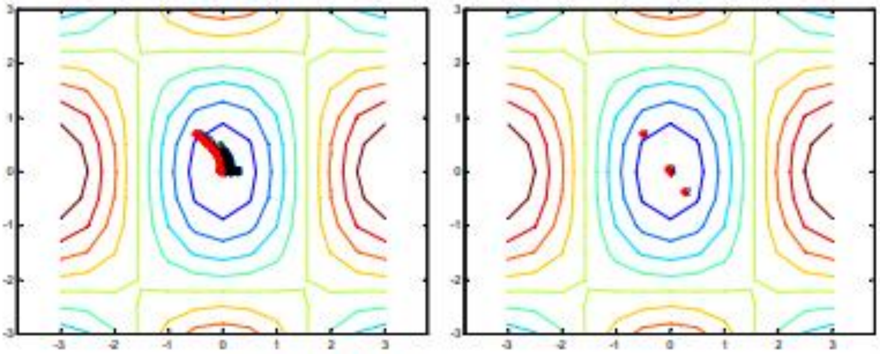
- f=((x1^2/4000)+(x2^2/4000))-(cos(x1)*cos(x2/(sqrt(2))))



opt_Ndim_general.m

Gradient Descent
stepsize=0.1
converged at 118 th
iteration

Newton Raphson
converged at the 4th
iteration



x0=[-0.5 ; 0.7]

opt_Ndim_general.m

Mehmet Fatih AMASYALI Optimization Techniques Lecture Notes YILDIZ TECHNICAL UNIVERSITY COMPUTER ENG DEPT.

ChatGPT

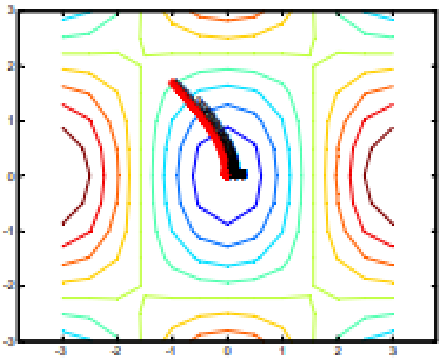
Görselde, Griewank fonksiyonunun tanımı, 3 boyutlu yüzey grafiği ve iki optimizasyon yönteminin - Gradyan İnişi ve Newton-Raphson - kontur grafikleri üzerinde gösterilmiş sonuçları bulunuyor.

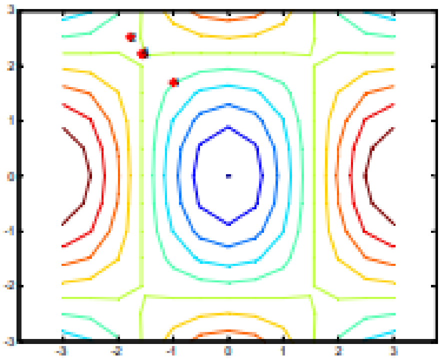
Üst kısımda Griewank fonksiyonu şu şekilde tanımlanmıştır:
$$f(x_1, x_2) = \frac{(x_1^2)}{4000} + \frac{(x_2^2)}{4000} - (\cos(x_1) \cdot \cos(\frac{x_2}{\sqrt{2}})) + 1$$
Bu fonksiyonun yüzey grafiği, fonksiyonun karmaşık bir yüzey yapısına sahip olduğunu ve birden fazla lokal minimuma sahip olabileceğini gösteriyor.

Alt kısımda, aynı fonksiyon için Gradyan İnişi ve Newton-Raphson yöntemlerinin uygulandığı ve bu yöntemlerin yakınsama hızları karşılaştırılmıştır. Gradyan İnişi yöntemi, adım boyutu 0.1 ile 118. iterasyonda yakınsamıştır. Buna karşılık, Newton-Raphson yöntemi, sadece 4. iterasyonda yakınsama sağlamıştır. Kontur grafikleri, bu iki yöntemin yakınsama süreçlerini ve optimizasyonun başlangıç noktasını (x0=[-0.5 ; 0.7]) gösteriyor. Mavi nokta, fonksiyonun minimum noktasını temsil ediyor ve her iki yöntem de sonunda bu noktaya yaklaşıyor. Ancak, Newton-Raphson yönteminin çok daha hızlı yakınsadığı gözlemleniyor. Bu görsel, farklı optimizasyon algoritmalarının performansının, karmaşık fonksiyonlarda nasıl büyük farklılıklar gösterebileceğini açıklıyor.

Gradient Descent
stepsize=0.1
converged at 142 th
iteration

Newton Raphson
converged at the 5th
iteration, but where?

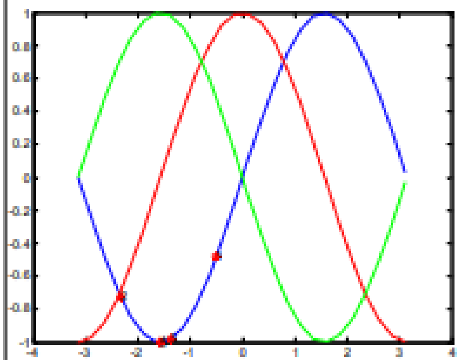




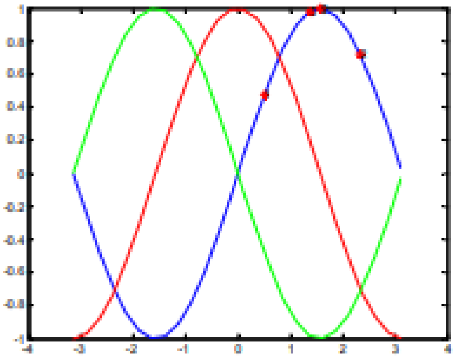
x0=[-1 ; 1.7]

opt_Ndim_general.m

What happened to the Newton Raphson?



x0=-0.5



x0=0.5

f(x)=sin(x)
Blue f
Red f'
Green f''

Attention to the signs of f' and f''

newton_raphson_2.m

ChatGPT

Görselde, iki farklı optimizasyon yöntemi olan Gradyan İnişi ve Newton-Raphson'un uygulanma sonuçları ve bir fonksiyonun türevlerinin işaretlerine dikkat çekilmesi gerektiğini vurgulayan grafikler yer almaktadır.

Üst kısımda:

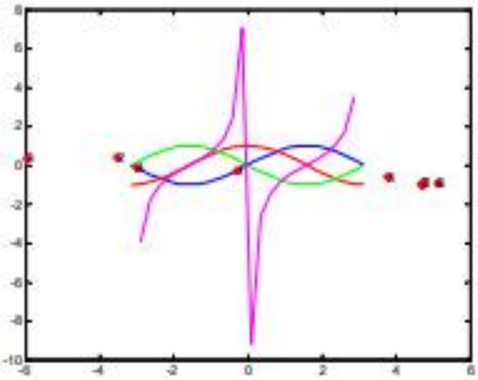
- Gradyan İnişi yöntemi, adım boyutu 0.1 ile 142. iterasyonda belirli bir noktaya yakınsamış. Bu süreçte izlenen yol, kırmızı çizgiler ile gösterilmiş.
- Newton-Raphson yöntemi, 5. iterasyonda yakınsamış, ancak yakınsama noktasının neresi olduğu sorusu sorulmuş. Bu yöntemde izlenen yol mavi çizgiler ile gösterilmiş ve yakınsama noktasının kontur grafiklerinde belirli bir noktada olduğu işaretlenmiş.

Alt kısımda:

- "What happened to the Newton Raphson?" başlıklı iki grafik yer almakta ve bu grafikler fonksiyonun (f), birinci türevinin (f') ve ikinci türevinin (f'') değerlerini farklı x değerleri için göstermekte. Buradaki amaç, Newton-Raphson yönteminin, fonksiyonun türevlerinin işaretlerine bağlı olarak neden beklenmedik bir noktaya yakınsayabileceğini anlamaktır.
- Grafikler, fonksiyonun (mavi), birinci türevinin (kırmızı) ve ikinci türevinin (yeşil) x eksenindeki değişimini gösteriyor. Bu, Newton-Raphson yönteminin bazen en yakın minimum veya maksimuma değil, türevlerin işaretine bağlı olarak farklı bir noktaya yakınsayabileceğine dikkat çekiyor.

Bu görseller, optimizasyon yöntemlerinin performansının başlangıç noktasına, adım boyutuna ve fonksiyonun kendisine bağlı olarak nasıl değişebileceğini gösteriyor ve özellikle Newton-Raphson yönteminin fonksiyonun türevinin işaretlerine duyarlı olduğunu vurguluyor.

What happened to the Newton Raphson?

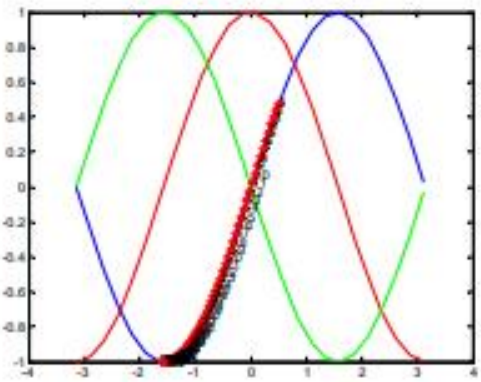


f(x)=sin(x)
Blue f
Red f'
Green f''
Magenta f'/f''

x0=-0.3
f' / f'' is not continuous

newton_raphson_2.m

What happens if we use Gradient descent?



f(x)=sin(x)
Blue f
Red f'
Green f''

Step size=0.05
f' is positive, f'' is not used

newton_raphson_2.m

ChatGPT

Görselde, iki farklı grafik bulunmaktadır ve her iki grafik de $f(x) = \sin(x)$ fonksiyonunun ve bu fonksiyonun türevlerinin davranışlarını göstermektedir.

Üst Grafik:

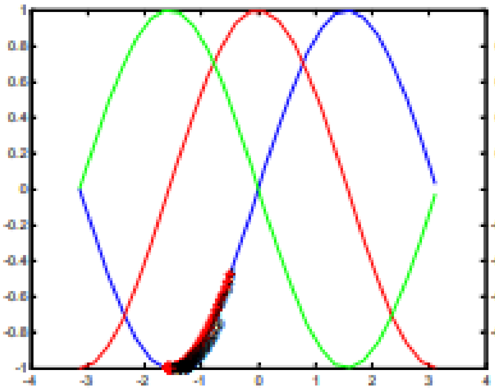
- "What happened to the Newton Raphson?" başlığı altında, Newton-Raphson yönteminin $x_0 = -0.3$ başlangıç noktasından itibaren uygulandığında karşılaşılan problem gösterilmekte. Fonksiyonun birinci türevi (f', kırmızı ile gösterilmiş), ikinci türevi (f'', yeşil ile gösterilmiş) ve Newton-Raphson iterasyonlarının (f'''/f'', mor ile gösterilmiş) grafiği çizilmiş. Grafik, türevlerin süreksiz olduğu bir noktayı işaret ediyor, bu da Newton-Raphson yönteminin bu noktada başarısız olabileceğini gösteriyor.

Alt Grafik:

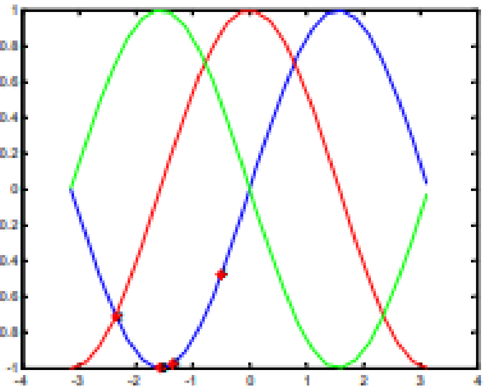
- "What happens if we use Gradient descent?" başlığı altında, Gradyan İnişi yönteminin $f(x)$ fonksiyonuna uygulandığında ne olduğu gösterilmekte. Adım boyutu 0.05 olarak ayarlanmış ve Gradyan İnişi yönteminin iterasyonlarının (siyah noktalar ile gösterilmiş) grafiği çizilmiş. Bu grafikte, fonksiyonun birinci türevi (kırmızı) ve ikinci türevi (yeşil) de gösterilmiş. Fonksiyonun pozitif olduğu durumlarda ikinci türev kullanılmamış. Gradyan İnişi yöntemi, belirli bir yörüngeyi takip ederek fonksiyonun minimum noktasına doğru ilerlemekte.

Her iki grafik de, optimizasyon yöntemlerinin karmaşık fonksiyonlar üzerindeki davranışlarını ve potansiyel zorlukları göstermektedir. Özellikle, türevlerin süreksiz olduğu noktaların veya fonksiyonun alışılagelmiş özelliklerinin olmadığı durumların optimizasyon algoritmalarının performansını nasıl etkileyebileceğini vurgulamaktadır.

Optimization using approximated derivatives



Gradient descent converged at 174th iteration



Newton Raphson converged at 7th iteration

f(x)=sin(x)
Blue f
Red real f'
Green real f''

newton_raphson_3.m

Some more comparisons

- opt_Ndim_general.m
- Nigthmares of a convex optimization, because of local minimums
- acckley f=(-20*exp(-0.2*sqrt((1/2)*(x1.^2+x2.^2))) -
-    exp((1/2)*(cos(2*pi*x1) + cos(2*pi*x2))) +
-    20 + exp(1) + 5.7);
- griewank f=((x1^2/4000)+(x2^2/4000))-(cos(x1)*cos(x2/(sqrt(2))));
- rastrigin f=10*2 + x1.^2 + x2.^2 - 10*cos(2*pi*x1) - 10*cos(2*pi*x2);
- rosen f=100*(x1^2-x2)^2+(x1-1)^2;
- schwell f=(abs(x1)+abs(x2))+(abs(x1)*abs(x2));

ChatGPT

Görselde, yaklaşık türevler kullanılarak yapılan optimizasyonun sonuçları ve farklı optimizasyon problemleri için karşılaştırmalar sunulmuştur.

Üst bölümde iki grafik bulunmaktadır:

- Sol taraftaki grafik, $f(x) = \sin(x)$ fonksiyonu için Gradyan İnişi yönteminin 174. iterasyonda yakınsadığını gösteriyor. Grafikte fonksiyonun (mavi), birinci türevinin (kırmızı) ve ikinci türevinin (yeşil) grafikleri çizilmiştir.
- Sağ taraftaki grafik, aynı fonksiyon için Newton-Raphson yönteminin 7. iterasyonda yakınsadığını gösteriyor. Bu grafikte de fonksiyon ve türevlerinin grafikleri benzer renklerle gösterilmiştir.

Alt bölümde ise farklı türde optimizasyon problemleri ve onların genel ifadeleri verilmiştir:

- Ackley fonksiyonu, karmaşık bir global minimuma sahip olan ve yerel minimumlarla dolu olan bir fonksiyondur. Bu fonksiyon optimizasyon problemlerinde sıkça karşılaşılan zorluklardan biridir.
- Griewank fonksiyonu da benzer şekilde birçok yerel minimuma sahip olup zor bir optimizasyon problemi oluşturur.
- Rosenbrock fonksiyonu, bir "banan function" olarak da bilinir ve tipik olarak optimizasyon algoritmalarının performansını test etmek için kullanılır. Bu fonksiyon da karmaşık bir global minimum yapısına sahiptir.

Bu görseldeki bilgiler, optimizasyon algoritmalarının farklı fonksiyon türlerine göre nasıl değişken performans gösterebileceğini ve yaklaşık türevlerin kullanımının bazı durumlarda yeterli olabileceğini, ancak karmaşık fonksiyonlarda dikkatli olunması gerektiğini vurgulamaktadır.