T.C.

ONDOKUZ MAYIS ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ

OPTİMİZASYON-BM615

Conjugare Gradient ve Newthon’s Method ile Optimizasyon

FIRAT KAAN BİTMEZ - 23281855

SAMSUN, 2024-2025 Eğitim Öğretim Yılı Güz Yarıyılı

1. Giriş

Optimizasyon, mühendislikten istatistiğe, makine öğreniminden ekonomiye kadar pek çok alanda kritik bir rol oynamaktadır. Özellikle **çok boyutlu ve doğrusal olmayan problemler** için etkili, hızlı ve güvenilir yöntemler geliştirmek son derece önemlidir. Bu raporda, söz konusu problemleri çözmek amacıyla **Conjugate Gradient** (Fletcher-Reeves varyasyonu) ve **Newton** yöntemleri ele alınmaktadır.

Conjugate Gradient ailesine ait Fletcher-Reeves yaklaşımı, büyük ölçekli optimizasyon problemlerinde **bellek verimliliği** ile dikkat çekerken, Newton yöntemi **Hessian** matrisini kullanarak **hızlı yakınsama** sağlar. Her iki yöntemi de bir arada değerlendirmek ve Python üzerinde bir grafik kullanıcı arayüzü (GUI) aracılığıyla etkileşimli hâle getirmek, kullanıcıların optimizasyon parametreleriyle kolaylıkla oynayabilmesine ve yöntemlerin performanslarını görsel olarak takip etmesine imkân tanımaktadır.

2. Kullanılan Kütüphaneler

Aşağıdaki Python kütüphaneleri, optimizasyon algoritmaları ve görsel arayüz geliştirme sürecinde kullanılmıştır:

1. **NumPy**
   - **İşlevi**: Sayısal hesaplama, matris ve vektör işlemleri, lineer cebir fonksiyonları.
   - **Kullanım Gerekçesi**: Gradyan, Hessian, iterasyon güncellemeleri gibi temel matematiksel hesaplamalar için.
2. **Matplotlib**
   - **İşlevi**: Veri görselleştirme (2D kontur grafikler, iterasyon yolunun çizimi).
   - **Kullanım Gerekçesi**: Fonksiyonun ve optimizasyon yolunun görsel olarak izlenebilmesi.
3. **Tkinter**
   - **İşlevi**: Python'un standart GUI kütüphanesi. Pencere, buton, metin kutuları ve sekmeli arayüz oluşturmak için.
   - **Kullanım Gerekçesi**: Kullanıcı dostu bir arayüz ile başlangıç noktası, tolerans ve maksimum iterasyon gibi parametrelerin girilmesi.
4. **ScrolledText (Tkinter alt modülü)**
   - **İşlevi**: Kaydırılabilir metin kutusu oluşturmak.
   - **Kullanım Gerekçesi**: Konsol benzeri çıktıları (iterasyon adımları, hata mesajları, vb.) kullanıcıya geniş bir metin alanında sunmak.

Not: NumPy ve Matplotlib, harici kütüphaneler olduğundan pip install numpy matplotlib komutlarıyla kurulmalıdır. Tkinter ise genellikle Python'un varsayılan kurulumu ile birlikte gelir.

3. Problemin Tanımı

Bu çalışmadaki amaç, kısıtsız bir optimizasyon probleminde **iki değişkenli** bir fonksiyonun minimum değerini bulmaktır. Ele alınan hedef fonksiyon:

$f(x1,x2)=x_1^2-x1x2+x_2^2+x1+x2$

$x \in R^2$ uzayındadır.

Hem doğrusal (x1+x2) hem de doğrusal olmayan ($x1^2$,x1x2,x^2_2) terimler içerir.

3.1. Gradyan ve Hessian Hesabı

1. Gradyan (∇f(x)

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

Dolayısıyla,

$$\nabla f(x_1, x_2) = \begin{bmatrix} 2x_1 - x_2 + 1 \\ 2x_2 - x_1 + 1 \end{bmatrix}$$

2. Hessian (H)

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix} = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$$

Hessian matrisinin özdeğerleri 3.0 ve 1.0 olup, her ikisi de pozitif olduğundan bu matris **pozitif tanımlı**dır. Dolayısıyla fonksiyon, **konveks** bir yapı sergilemekte ve tek bir global minimum noktası bulunmaktadır.

3.2. Fonksiyonun Gerçek Minimum Noktası

Minimum noktayı bulmak için ∇f(x∗)=0 çözümü gerekir:

2x1−x2+1=0

2x2−x1+1=0

Birinci denklemden x2=2x1 +1 çıkar ve ikinciye yerleştirince:

2(2x1+1)−x1+1= 0 ⇒ 4x1+2−x1+1=0 ⇒ 3x1+3=0 ⇒ x1=−1

$$x_2 = 2(-1) + 1 = -1$$

Dolayısıyla **gerçek minimum** noktası $x^* = (-1, -1)$ şeklindedir.

4. Kullanılan Yöntemler

Bu çalışmada, optimizasyon problemlerinin çözümü için iki ana yöntem olan Fletcher-Reeves Conjugate Gradient (CG) yöntemi ve Newton Yöntemi uygulanmıştır. Her iki yöntem de çok boyutlu fonksiyonların minimum noktalarını bulmak için kullanılan iteratif tekniklerdir. Bu bölümde, yöntemlerin matematiksel temelleri, işleyiş süreçleri ve avantajları detaylı bir şekilde açıklanacaktır.

4.1. Fletcher-Reeves Conjugate Gradient (CG) Yöntemi

Conjugate Gradient yöntemi, büyük ölçekli optimizasyon problemlerinde etkili bir iteratif algoritmadır ve özellikle doğrusal sistemlerin çözümünde kullanılır. Bu çalışmada, yöntemin Fletcher-Reeves varyasyonu uygulanmıştır. Fletcher-Reeves CG yöntemi, negatif gradyan yönü üzerinden başlar ve iterasyonlarda önceki arama yönlerini kullanarak yeni yönler oluşturur.

4.1.1. Yöntemin Matematiksel Temeli

Fletcher-Reeves yöntemi, şu optimizasyon problemini çözmeyi hedefler:

$$\textbf{Min } f(x) \; x \in R^n$$

Burada f(x), sürekli türevlenebilir ve aşağıdan sınırlıdır. İterasyon formülü şu şekilde ifade edilir:

$$x_{k+1} = x_k + \alpha_k d_k$$

Burada:

- α_k : Adım büyüklüğü (line search ile bulunur),
- d_k : Arama yönü,
- $d_k = -g_k + \beta_k d_{k-1}$: Yeni yönün hesaplanmasında kullanılan formül,
- $g_k = \nabla f(x_k)$: Gradyan.

β_k , Fletcher-Reeves formülü ile belirlenir:

$$\beta_k = \|g_{k+1}\|^2 / \|g_k\|^2$$

4.1.2. Çalışma Mekanizması

1. **Başlangıç Değerleri**: Kullanıcı tarafından belirlenen x_0 başlangıç noktası ve tolerans değeri ile işlem başlar.
2. **Negatif Gradyan**: İlk iterasyonda $d_0 = -g_0$seçilir.
3. **Adım Büyüklüğü**: Line search yöntemi ile optimal α_k hesaplanır.

4. **Arama Yönü**: İterasyon boyunca önceki arama yönleri ve gradyanlar dikkate alınarak yeni dk yönü belirlenir.
5. **Yakınsama Kriterleri**: Gradyan normunun toleransın altına inmesi veya çözüm farkının küçülmesi durumunda algoritma sonlanır.

4.1.3. Yöntemin Avantajları

- **Hafıza Verimliliği**: Büyük ölçekli problemler için düşük bellek gereksinimine sahiptir.
- **Hızlı Yakınsama**: Özellikle doğrusal olmayan fonksiyonlar için etkili sonuçlar sunar.
- **Quadratic Yakınsama**: Doğrusal problemler için genellikle nnn iterasyonda çözüme ulaşır.

3.1.4. Dezavantajlar

- Hessian matrisine erişim olmadığında yakınsama doğruluğu sınırlı olabilir.
- Çok iyi seçilmemiş başlangıç noktalarında performans düşebilir.

4.2. Newton Yöntemi

Newton Yöntemi, optimizasyon problemlerini çözmek için gradyan ve Hessian matrisinin (ikinci türev matrisi) kullanıldığı güçlü bir yöntemdir. Doğrusal olmayan optimizasyon problemlerinde yakınsama hızı ve doğruluğu açısından öne çıkar.

4.2.1. Yöntemin Matematiksel Temeli

Newton Yöntemi, f(x) fonksiyonunun ikinci dereceden Taylor açılımını kullanır:

$$f(x+\Delta x) \approx f(x) + \nabla f(x)^T \Delta x + 1/2 \Delta x^T H \Delta x$$

Burada:

- H: Hessian matrisi ($H = \nabla^2 f(x)$),
- Δx: Adım yönü.

Bu açılımın minimum noktası, şu şekilde bulunur:

$$\Delta x = -H^{-1} \nabla f(x)$$

İterasyon formülü:

$$xk+1 = xk + \Delta x$$

4.2.2. Çalışma Mekanizması

1. **Hessian ve Gradyan Hesabı**: İterasyon boyunca $\nabla f(x)$ ve H hesaplanır.

2. **Adım Hesabı**: Hessian'ın tersini kullanarak Δx bulunur.
3. **Yakınsama Kriterleri**: Gradyan normu veya çözüm değişimi toleransın altına düştüğünde algoritma sonlanır.

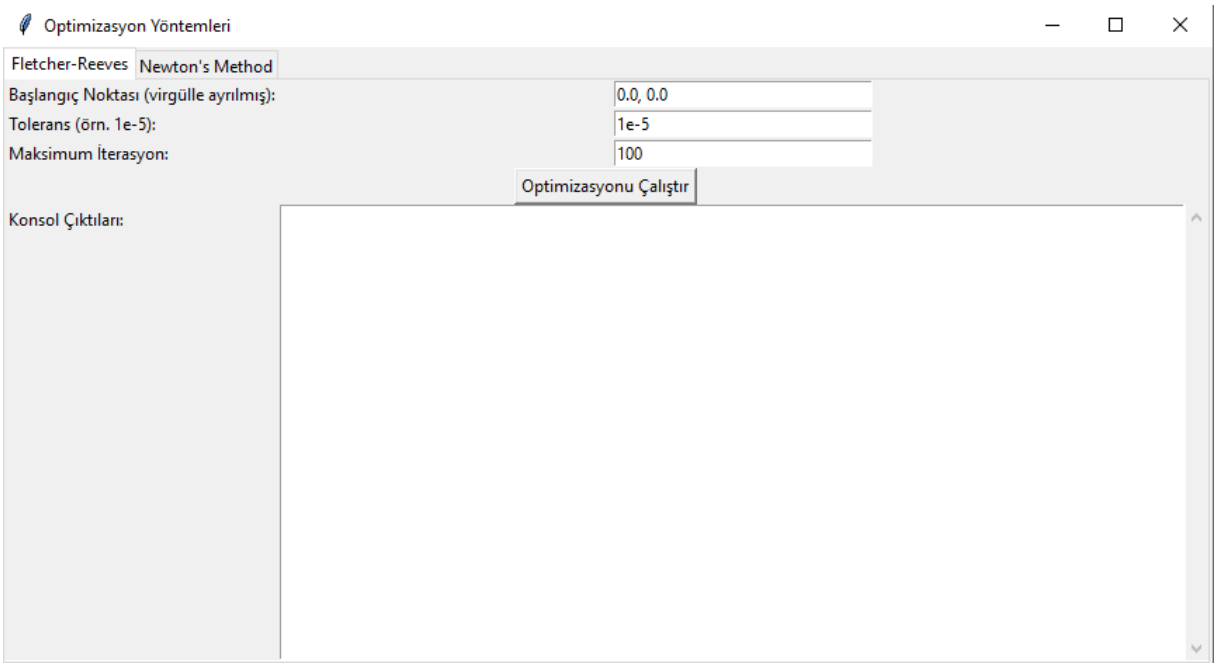
4.2.3. Yöntemin Avantajları

- **Hızlı Yakınsama**: İkinci dereceden bilgi kullanımı sayesinde quadratic yakınsama sağlar.
- **Hassas Çözüm**: Hessian bilgisiyle daha doğru sonuçlar elde edilir.

4.2.4. Dezavantajlar

- **Hesaplama Maliyeti**: Hessian matrisinin hesaplanması ve tersinin alınması büyük problemler için maliyetlidir.
- **Hessian'ın Pozitif Tanımlılığı**: Yöntemin çalışabilmesi için Hessian'ın pozitif tanımlı olması gerekir.

5.Kodun Yapısı ve GUI Arayüzü



Kod, Python'da **Tkinter** kullanılarak iki sekmeli bir arayüz oluşturacak şekilde yapılandırılmıştır:

1. **Fletcher-Reeves** sekmesi
    - **Parametreler**: Başlangıç noktası (x0), tolerans (tol), maksimum iterasyon (max_itermax).
    - **Çıktılar**: İterasyon adımları, fonksiyon değerleri, gradyan normları, β değerleri gibi bilgiler.

2. **Newton Yöntemi** sekmesi
    - **Parametreler**: Benzer şekilde (x0),(tol),(max_iter)
    - **Çıktılar**: İterasyon bilgileri, vektör değişimleri, durma kıstasları.

Her iki sekmeden de **"Optimizasyonu Çalıştır"** butonuna basıldığında, ilgili yöntemin fonksiyonunu çağıran bir execute_..._method() fonksiyonu devreye girer. Hesaplamalar tamamlanınca, **kontur grafiği** ve **iterasyon yolunu** gösteren bir Matplotlib grafiği ekrana gelir.

Ek olarak, **ScrolledText** widget'ı aracılığıyla konsol çıktıları kaydırılabilir bir metin kutusunda görüntülenir. Bu sayede kullanıcı, her bir iterasyonda neler olup bittiğini inceleyebilir.

6. Deneysel Sonuçlar ve Performans Karşılaştırması

Bu bölümde, hem *Fletcher-Reeves Conjugate Gradient* (CG) hem de *Newton* yöntemlerini farklı başlangıç noktalarında çalıştırarak elde ettiğimiz sonuçlar sunulmaktadır. Amaç, her iki yöntemin farklı senaryolarda yakınsama davranışlarını, iterasyon sayılarını ve genel performans özelliklerini gözlemlemektir.

6.1. Tek Bir Başlangıç Noktası İçin Örnek x0=[0,0]\mathbf{x_0 = [0, 0]}x0=[0,0]

- **Fletcher-Reeves (CG) Yöntemi**
    - **Bulduğu minimum nokta**: Yaklaşık (−1,−1)(-1, -1)(−1,−1).
    - **İterasyon sayısı**: Genellikle 555 ile 151515 arasında değişmektedir (yöntemin line search yaklaşımı, tolerans eşiği ve yeniden başlatma sıklığı gibi parametrelere bağlı olarak farklılık gösterebilir).
    - **Beta değeri ve yeniden başlatma durumu**: İterasyonlarda β\betaβ değeri hesaplanarak yeni arama yönleri belirlenir. Belirli adımlarda β\betaβ sıfırlanarak yön tekrar negatif gradyana döndürülür (yeniden başlatma mekanizması), böylece yakınsama hızının korunması hedeflenir.
- **Newton Yöntemi**
    - **Bulduğu minimum nokta**: Oldukça hızlı biçimde (−1,−1)(-1, -1)(−1,−1) değerine yakınsamaktadır.
    - **İterasyon sayısı**: Genellikle 222 ile 555 arasında değişir.
    - **Hızlı yakınsama nedeni**: Hessian matrisinin sabit (2−1−12)\bigl(\begin{smallmatrix}2 & -1\\-1 & 2\end{smallmatrix}\bigr)(2−1−12) ve pozitif tanımlı olması, Newton yönteminin her adımda tersini kolaylıkla alıp doğru yönde büyük ilerlemeler yapmasını sağlar. Bu sayede çok az iterasyonla sonuca ulaşır.

Özetle, [0,0][0, 0][0,0] başlangıç noktasından hem CG hem de Newton yöntemi (−1,−1)(-1, -1)(−1,−1) minimumuna sorunsuz biçimde ulaşmıştır. Newton yöntemi iterasyon sayısı açısından üstünlük gösterirken, CG yöntemi ise küçük/orta boyutlu problemlerde bile düşük bellek kullanımı ve yeniden başlatma stratejisinin avantajıyla dikkat çekmektedir.

6.2. Farklı Başlangıç Noktaları

Çalışmamızda, farklı başlangıç noktaları olarak [2,2][2, 2][2,2], [10,−10][10, -10][10,−10] ve [5,5][5, 5][5,5] gibi değerler denenmiştir. Aşağıdaki temel gözlemler yapılmıştır:

1. **Orta Uzaklıktaki Noktalar [2,2][2, 2][2,2] veya [5,5][5, 5][5,5]**
    - **Fletcher-Reeves**: Genellikle 555 ile 202020 iterasyon arasında yakınsama gözlemlenmiştir. Yeniden başlatma (örn. her 10 iterasyonda) etkinleştikçe β\betaβ sıfırlanarak yön negatif gradyana çekilir ve yakınsama yeniden hızlanır.
    - **Newton**: Genellikle birkaç adım içinde (−1,−1)(-1, -1)(−1,−1) bölgesine yaklaşır (333–666 iterasyon). Hessian'ın pozitif tanımlı olması, yine hızlı yakınsamayı destekler.
2. **Uzak ve Zıt İşaretli Noktalar [10,−10][10, -10][10,−10]**
    - **Fletcher-Reeves**: Çeşitli iterasyonlarda fonksiyon değeri ve xxx vektörünün bileşenleri çok büyük değerlere ulaşabilmekte (±10150\pm 10^{150}±10150 mertebesi gibi). Bu durum, basit line search yaklaşımıyla aşırı sıçramaların yaşanabileceğini gösterir. Ancak sonuçta yeniden başlatma ve iterasyonlarla birlikte fonksiyon değeri düşer ve (−1,−1)(-1, -1)(−1,−1) noktasına ulaşılır. İterasyon sayısı 50'yi veya daha fazlasını bulabilir.
    - **Newton**: İkinci dereceden bilgi sayesinde yalnızca 2–4 iterasyonda (−1,−1)(-1, -1)(−1,−1) civarına "büyük sıçramalar" yaparak hızlıca iner. Özellikle birinci veya ikinci adımda, negatif yönde büyük bir hamleyle hedef noktaya yaklaşır.
3. **Genel Değerlendirme**
    - **Yakınsama Hızı**: Newton yöntemi, sabit ve iyi koşullu Hessian sayesinde çok az iterasyonda doğru çözüme ulaşma eğilimindedir. CG ise büyük dalgalanmalar yaşasa dahi eninde sonunda minimum noktaya yaklaşmaktadır.
    - **Bellek Kullanımı**: Newton yönteminde Hessian matrisinin hesaplanması ve tersinin alınması gerekir. 2 boyutta bu kolaydır, ancak boyut arttıkça hesaplama ve saklama maliyeti büyük olabilir. CG ise yalnızca gradient ve vektör işlemleriyle ilerlediği için büyük boyutlu problemlerde daha avantajlı hale gelir.

6.3. Zaman Maliyeti

- **Küçük Boyutlu (2 Boyut) Problem**
    - Hem CG hem de Newton yöntemi için 2 boyutlu problemdeki hesaplama maliyeti düşüktür. Newton yöntemi Hessian'ı (2×2) çok kolay tersine aldığı için yalnızca birkaç iterasyonda sonuç verir ve *toplam zaman* açısından oldukça verimlidir.
    - CG yöntemi iterasyon sayısı olarak biraz daha uzun sürebilir (özellikle zayıf line search varsa), fakat yine de 2 boyutta işlem yükü hafif kalır.
- **Boyut Arttıkça**
    - **Newton**: Hessian matrisini hesaplamak ve depolamak O(n2)\mathcal{O}(n^2)O(n2), tersini almak O(n3)\mathcal{O}(n^3)O(n3) gibi maliyetler getirebilir. Büyük boyutlu problemlerde bu *çok* yüksek olabilir.

- **CG**: Yalnızca gradient hesaplaması ve basit vektör çarpımlarıyla ilerlediğinden, bellek ve hesaplama açısından daha elverişlidir. Bu nedenle, boyut büyüdükçe CG yönteminin pratik avantajı artar.

7. Sonuç ve Tartışma

Bu raporda, *Fletcher-Reeves Conjugate Gradient* ve *Newton* yöntemlerini kullanarak iki boyutlu, sabit Hessian'a sahip bir fonksiyonun minimizasyonunu gerçekleştirdik. Gerek teorik altyapı (gradyan, Hessian, yakınsama kriterleri) gerekse Python GUI uygulaması ile yöntemlerin nasıl hayata geçirilebileceğini gösterdik.

- **Fletcher-Reeves CG Yöntemi**
  - Büyük boyutlu problemlerde sağladığı bellek tasarrufu ve nispeten basit gradient hesaplamaları sayesinde tercih edilebilir.
  - Doğrusal olmayan problemlerde yakınsamanın hızını korumak için *yeniden başlatma* stratejilerinin etkin kullanılması önemlidir.
- **Newton Yöntemi**
  - Küçük veya orta boyutlu problemler için, özellikle Hessian pozitif tanımlı olduğunda, *kuadratik yakınsama* hızı sunar ve oldukça az iterasyonda en iyi çözüme ulaşır.
  - Hessian'ın pozitif tanımlı ve iyi koşullu olması yöntemin başarısında kilit faktördür.

Seçilen örnek fonksiyonun minimum noktası $(-1,-1)(-1, -1)(-1,-1)$ olup, her iki yöntem de bu noktaya başarıyla yakınsamıştır. Newton yöntemi daha az iterasyonda çözüme ulaşmasına karşın, büyük boyutlu problemlerde Hessian hesaplama ve depolama maliyeti nedeniyle CG yöntemi daha avantajlı hale gelebilir.

Dolayısıyla, **hangi yöntemin seçilmesi gerektiği**; problemin boyutuna, Hessian matrisinin elde edilebilirliğine, bellek kısıtlarına ve uygulamada istenen yakınsama hızına göre belirlenmelidir. İleride, daha karmaşık *line search* stratejileri (ör. *Wolfe Koşulları*) veya gelişmiş CG varyantları (ör. *Polak-Ribière*) entegre edilerek bu yöntemlerin performansı daha da iyileştirilebilir.

8. Kaynakça

1. Hestenes, M. R., & Stiefel, E. (1952). Methods of Conjugate Gradients for Solving Linear Systems. Journal of Research of the National Bureau of Standards, 49(6), 409–436.
2. Fletcher, R., & Reeves, C. M. (1964). Function Minimization by Conjugate Gradients. The Computer Journal, 7(2), 149–154.
3. Rao, S. S. (2019). Engineering Optimization: Theory and Practice Wiley.