T.C.
KOCAELİ ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

ARAŞTIRMA PROBLEMLERİ

SAYISAL ANALİZ YÖNTEMLERİ

AYŞE HİLAL DOĞAN

KOCAELİ 2020

T.C.
KOCAELİ ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

ARAŞTIRMA PROBLEMLERİ

SAYISAL ANALİZ YÖNTEMLERİ

AYŞE HİLAL DOĞAN

Doç.Dr. Suhap ŞAHİN
Danışman, Kocaeli Üniv. .....

Öğr. Gör. Dr. Onur GÖK
Jüri Üyesi, Kocaeli Üniv. .....

Arş. Gör. Hikmetcan ÖZCAN
Jüri Üyesi, Kocaeli Üniv. .....

Tarih: 17 Ocak 2020

Bu dokümandaki tüm bilgiler, etik ve akademik kurallar çerçevesinde elde edilip sunulmuştur. Ayrıca yine bu kurallar çerçevesinde kendime ait olmayan ve kendimin üretmediği ve başka kaynaklardan elde edilen bilgiler ve materyaller (text, resim, şekil, tablo vb.) gerekli şekilde referans edilmiş ve dokümanda belirtilmiştir.

Öğrenci Adı Soyadı: Ayşe Hilal Doğan

Öğrenci No: 190201139

İmza:

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

SAYISAL ANALİZ YÖNTEMLERİ

ÖZET

Sayısal analiz yöntemleri, birçok alanda karşılaşılan ve çözümü oldukça zor olan ve zaman harcayan denklem sistemlerinin çözümünü en hızlı ve kolay şekilde çözebilen yöntemlerdir. Özellikle de bu işi yapan bir insan eli değil de bir bilgisayar olursa bu çok daha kusursuz ve hızlı çözümler demektir. Sayısal analiz yöntemlerini birçok problemin çözümünde kullanılabilir hale getirmek için yapılabilecek en önemli şey bu yöntemlerin olabildiğince hatasız kodlanması ve bir program haline getirilmesidir.

Bu çalışmada, sayısal analiz yöntemlerini tanıtmak, daha hızlı çözüm yöntemlerini araştırmak ve sayısal analiz yöntemlerini günümüzde çok kullanılan Python dilinde kodlayarak bilimsel çalışmalarda kullanılabilecek bir kütüphane oluşturmak amaçlanmıştır ve bu amaca yönelik araştırmalar yapılıp, kodlaması yapılmıştır.

NUMERICAL ANALYSIS METHODS

ABSTRACT

Numerical analysis methods are the methods that can be solved in the fastest and easiest way to solve the equation systems which are encountered in many fields and which are very difficult to solve. Especially if a computer is not a human hand doing this work, this means much more perfect and faster solutions. The most important thing that can be done to make the numerical analysis methods usable in solving many problems is to encode these methods as accurately as possible and to make them a program.

In this study, it is aimed to introduce numerical analysis methods, to search for faster solution methods and to create a library that can be used in scientific studies by coding numerical analysis methods in Python language which is widely used today.

GİRİŞ

Mühendislikte her alanda denklemler, integraller hesaplamaları, türev hesaplamaları kullanılır ve çoğu problemin çözümünde bu hesaplamaların çok hızlı ve en az hatayla yapılması büyük önem arz eder. Özellikle de gelişen teknoloji dünyasında zaman kavramı gitgide daha da önemli hale gelmektedir.

Sayısal analiz yöntemlerinde amaç, diferansiyel denklemler, integral hesaplamaları veya bazı denklemlerin analitik yöntemlerle değil, sayısal yöntemlerle çözümlenmeye çalışılmasıdır. Günümüzde bir çok analitik yöntemle çözülemeyen denklem veya hesaplamalar, sayısal analiz yöntemleriyle bilgisayar tarafından kolayca çözülebilmektedir.

Bu çalışmada, sayısal analiz yöntemlerini tanıtmak, daha hızlı çözüm yöntemlerini araştırmak ve sayısal analiz yöntemlerini günümüzde çok kullanılan Python dilinde kodlayarak bilimsel çalışmalarda kullanılabilecek bir kütüphane oluşturmak amaçlanmıştır.

1. SAYISAL ANALİZ YÖNTEMLERİ

Sayısal Analiz; diferansiyel denklemler, integral hesaplamaları veya bazı denklemlerin analitik yöntemlerle değil, sayısal yöntemlerle çözümlenmeye çalışılmasıdır. Günümüzde bir çok analitik yöntemle çözülemeyen denklem veya hesaplamalar, sayısal analiz yöntemleriyle bilgisayar tarafından kolayca çözülebilmektedir.

1.1 Hata

Sayısal analiz yöntemleri %100 doğru sonucu vermeyebilir. Bir miktar hata payı içerirler. Bu hata payı genelde çok küçük bir değerdir.

Hata = Gerçek değer - Hesaplanan değer

Hatanın kaynaklandığı yerler:

1.2 Ölçme Hatası

Elde edilen sonuç çeşitli yöntemlerle elde edilir ve gerek yönteme bağlı olsun gerek sistem kaynaklı hatalar olsun elde edilen sonucun hatalı olmasına sebep olabilir.

1.3 Yuvarlama Hataları

Bu yöntemleri kullanırken yapılan bazı hesaplamalar çok küçük rasyonel ve irrasyonel sayılarla yapılır. Bu durumda hesaplamaları yaparken girilmesi gereken değerler çok dikkatli girilmelidir. Aynı şekilde hesaplama yapıldıktan sonra da hesaplamanın sonucu ondalık olarak sonsuz basamakla ifade edilmelidir. Sonuçlar hesaplanırken bukadar sonsuz basamak kullanılmadığı için sonuç yuvarlanır ve genelde virgülden sonra 2 ya da 4 basamak gösterilir. Bir de hesaplamaların belli aşamasında yuvarlamalar olduğunda bu hesaplamanın sonraki aşamalarında daha büyük hatalara sebep olabilir. Bu da sonucu büyük ölçüde etkileyebilir. Bu tür hatalara sayısal analiz yöntemlerinde yuvarlama hataları denir.

1.4 Kesme Hataları

Hesaplamalar yaklaşık olarak yapıldığında oluşan hatalardır. Mesela sonsuz terimli bir serinin açılımında ilk n terim açılır ve bunun sonucunda az da olsa bir hata oluşur. Bu hata kesme hatasıdır.

1.5 İnsan Kaynaklı Hatalar

İnsan kaynaklı hata, sayısal analiz yöntemlerini kullanan bir bilgisayar programında hesaplama yapmak isteyen bir insanın yanlış veri girmesi ya da olmayacak bir denklem hesaplamaya çalışmasıdır.

1.6 Bilgisayar Kaynaklı Hatalar

Bilgisayar elektrik ile çalışan bir cihaz olduğu için elektrik kaynaklı bir sorun olabilir ya da CPU dan kaynaklı hatalar olabilir. Bu gibi öngörülemeyen hatalara bilgisayar kaynaklı hata denir.

2. HATA ÇEŞİTLERİ

2.1 Mutlak Hata

Analitik olarak hesaplanan ve doğru olduğuna emin olunun sonuç ile sayısal bir yöntem ile hesaplanmış sonucun farkının mutlak değeri alınmış haline mutlak değer denir.

$$\varepsilon m = | yg - yy | \quad (2.1)$$

Denklem (2.1) de εm mutlak hatayı, yg gerçek değer, yy yaklaşık değeri ifade etmektedir.

2.2. Bağıl Hata

$$\varepsilon_b = \frac{|y_g - y_y|}{|y_g|} \quad (2.2)$$

Denklem (2.2)'de verilen εb bağıl hatayı, yg gerçek değeri, yy yaklaşık değeri ifade etmektedir. Bağıl hata gerçek değer ile yaklaşık değerin farkının mutlak değerinin yine gerçek değere bölünmesi ile elde edilir. Bağıl hata 100 ile çarpılarak yüzde bağıl hata bulunabilir.

Örneğin, bir füzenin menzilinin gerçek değeri 5000 km dir ve yapılan hesaplarda 4999 km olarak bulunmuştur. Burada bağıl hata |5000-4999| / 5000=0.0002 dir. Yüzde olarak ise %0,02 dir.

$\varepsilon_b = \frac{5000-4999}{5000}$ = 0.0002 'dir.

Yüzde bağıl hata = $\frac{5000-4999}{5000}$ x 100 = %0.02 'dir.

2.3. Yaklaşım Hatası

Diğer hata çeşitlerinde hata oranını hesaplamak için hatanın gerçek değerinin bilinmesi gerekir. Eğer gerçek değer bilinmiyorsa yaklaşım hatası kullanılmalıdır. Yaklaşım hatası adım adım yapılan hesaplamalarda önceki iterasyonda çıkan sonuç ile bir sonraki iterasyonda çıkan sonuçlar vasıtasıyla bulunur.

Yaklaşım hatası için yeni iterasyonla eski iterasyonun farkı alınır ve yeni iterasyona bulunur. Mutlak değeri alınır. Bu şekilde hata miktarı en az olana kadar hesaplamalar yapılır.

$$\varepsilon_y = \frac{|y_{yeni} - y_{eski}|}{|y_{yeni}|} \qquad (2.3)$$

Denklem (2.3)'de verilen εy yaklaşım hatasını, y_{eski} bir önceki iterasyonda elde edilen sonucun değerini, y_{yeni} son iterasyonda elde edilen sonucun değerini ifade etmektedir.

Örnek: e^x fonksiyonunun seri açılımı aşağıda verilen resimdeki gibidir. Yaklaşım hatası ve mutlak hata yüzdelerini bulmaya çalışalım; $e^{0.5}$=1,648721271 olarak alalım ve belli miktarda terimi göz önüne alarak hata yüzdelerini bulalım.

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots\ldots + \frac{x^n}{n!}$$

Şekil 2.1. e^x fonksiyonunun seri açılımı

$$\varepsilon_b = \frac{e^{0,5} - 1.0}{e^{0,5}} \times 100 = \frac{1,648721271 - 1.0}{1,648721271} \times 100 = \%39,3$$

Şekil 2.2. e_b ' nin x'e 0.5 verilerek hesaplanma yöntemi

İlk iki terimi alındığında ise e^x = 1 + x = 1 + 0,5 = 1,5 olarak bulunur.
Bu ilk iki terime bakarak yaklaşım hatası ve bağıl hata için şu hesaplama yapılır:

İlk verilen bağıl hata, ikinci verilen yaklaşım hatasıdır.

$$\varepsilon_b = \frac{e^{0,5} - 1,5}{e^{0,5}} \times 100 = \frac{1,648721271 - 1,5}{1,648721271} \times 100 = \%9,02$$

Şekil 2.3. e_b ' nin e^x 'e 1.5 verilerek hesaplanma yöntemi

$$\varepsilon_y = \frac{|y_{yeni} - y_{eski}|}{|y_{yeni}|} = \frac{1,5 - 1}{1,5} \times 100 = \%33,3$$

Şekil 2.4. e_y ' nin yüzde olarak hesaplanma yöntemi

Terim sayısı 2 alınarak bu sonuçlar bulunmuştur. Terim sayısını arttırarak elde edilen yeni değerler ve oluşan bağıl hata ve yaklaşım hataları tablo olarak aşağıda verilmiştir. Terim sayısı arttıkça oluşan hata yüzdeleri de azalmaktadır. Bu tür seri açılımlarında ne kadar çok terim baz alınarak hesaplama yapılırsa hata oranı okadar az olur. Bu da tabloda açıkça görülmektedir.

Tablo 2.1 Terim sayısına göre hesaplanan ε_b ve ε_y

Terim Sayısı	Sonuç	ε_b(%)	ε_y(%)
1	1	39,3	-
2	1,5	9,02	33,3
3	1,625	1,44	7,69
4	1,64583333	0,175	1,27
5	1,64843750	0,0172	0,158
6	1,64869791	0,00142	0,0158
7	1,64871961	0,0001007	0,001316

3. DOĞRUSAL OLMAYAN DENKLEM SİSTEMLERİ

Bir çok alanda doğrusal olmayan denklemler sistemleriyle karşılaşabiliriz. Buna örnek yüksek dereceli polinomlar, logaritmik, üstel veya trigonometrik gibi doğrusal olmayan terimler içeren denklemler. Doğrusal olmayan denklemlere aynı zamanda lineer olmayan denklemler de denir.

Doğrusal olmayan denklemler genelde f(x)=0 şeklinde kapalı formda yazılırlar. Karşılaşılan denklemler genelde çok değişkenlidir.

Kök bulma işlemi, verilen f(x) denkleminde f(xi) = 0 'ı sağlayan xi değerlerinin bulunması işlemidir. Kök bulma işlemlerinde öncelik olarak kökün hangi aralıkta olduğu belirlenir. Kök bu aralıkta bulunmaya çalışılır. İşlemler bu aralık baz alınarak yapılır. Tek değişkenli fonksiyonlarda kökler aynı zamanda fonksiyon eğrisinin x eksenini kestiği noktalardır.

- a ve b gibi iki farklı sayıyla belirlenmiş bir aralıkta ;
- Kök a ve b sayılarının arasında tanımlanmış ise,
- f(x) fonksiyonu bu aralıkta sürekli ise,
- f(a) * f(b) < 0 ise,

Bu durumda f(x) = 0 eşitliğini sağlayan bir xi değeri vardır denilir. Kök bulma işlemi, verilen denklemi sağlayan gerçek değere çok yaklaşık olan bağımsız değerlerin bulunması işlemidir. Analitik yöntemlerle çözülemeyen denklemler için geliştirilen sayısal yöntemler kullanılmaya başlanmadan önce denklemlerin köklerini bulmak için farklı yöntemler geliştirilmişti. Örneğin 2. dereceden denklemlerin çözümü için çeşitli çözüm yöntemleri vardı. Ancak birçok denklem sayısal analiz yöntemlerindeki gibi basitçe ve bilgisayar vasıtasıyla çözülememekteydi. Çözüldüğü zaman ise sayısal analiz yöntemlerindeki kadar gerçeğe yakın sonuçlar bulunamıyordu.

Kök bulma yöntemlerinden en basit ve pratik olanı ve bu yüzden en çok tercih edileni ise Grafik yöntemidir.

4. KÖK BULMA YÖNTEMLERİ

4.1 Grafik Yöntemi

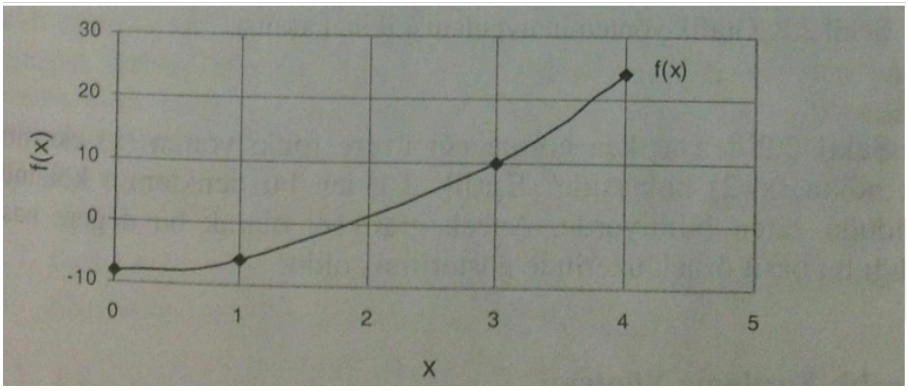
Grafik yönteminde fonksiyonun çözümü için fonksiyona ait bazı değerler (kökün çözümü için bir başlangıç değeri, kök aranırken kullanılacak aralık değeri) vasıtasıyla fonksiyonun grafiği çizilir. Çizilen grafiğe bakarak kökün yaklaşık olarak hangi x noktasında olduğu tahmin edilir. Bunun için grafiğin x eksenini kestiği noktalara bakılır.

Örnek: $f(x) = xe^x - 2$
Bu fonksiyonu [0,1] aralığında 0.25 aralıklar ile inceleyen tablo aşağıdadır.

Tablo 4.1.1. $f(x) = xe^x - 2$ fonksiyonunun 0-1 aralığında hesaplanan sonuçları

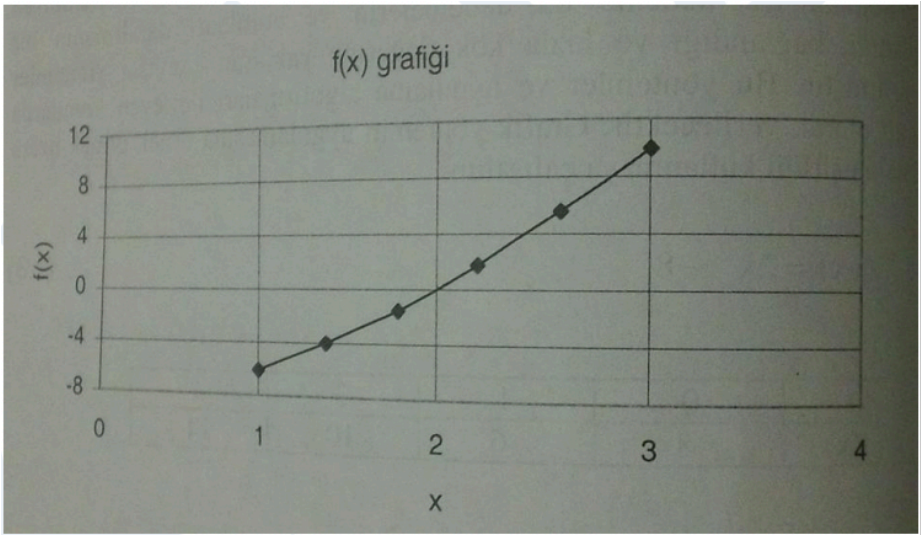
x	F(x)
0,0	-2
0,25	-1,6788993
0,5	-1,175639
0,75	-0,412250
1,0	0,718281

Tablo 4.1 incelendiğinde, f (0,75) x f (1,0) < 0 olduğundan aranan kök [0.75,1.0] aralığındadır. f(0,85) = –0,011300 olduğundan aranan kök [0.85,1.0] aralığındadır. Bu şekilde aralıklar küçültülerek doğru sonuç bulunur. Örnek: $f(x)= 2x^2 - 8$ denkleminin kökünü grafik yöntemi kullanarak bulalım.



Şekil 4.1.2. $f(x)= 2x^2 - 8$ denkleminin grafiği

Grafik çizilip incelendiğinde kökün [1,3] aralığında olduğu görülmektedir.Çünkü bu aralıkta eğri negatif kısımdan pozitife geçiş yapmıştır. Aralığı biraz daha daraltarak gerçek sonuca biraz daha yaklaşabiliriz.



Şekil 4.1.2. f(x)= $2x^2$ – 8 denkleminin grafiği

Tablo 4.1.2 F(x) fonksiyonunun x'in değerlerine göre değişimi

X	1.0	1.4	1.8	2.2	2.6	3.0
F(x)	-6	-4.08	1.52	1.68	5.52	10

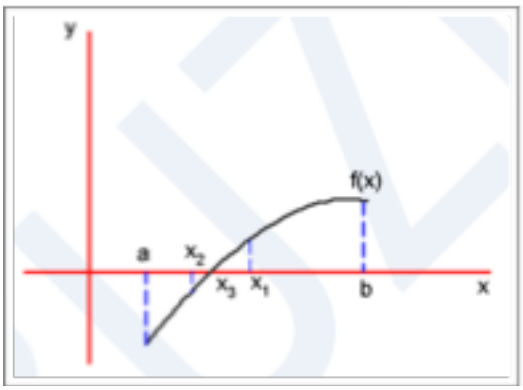
Aralık daraltılarak grafik tekrar çizildiğinde aranan kökün $x = 2$ olduğu görülmektedir.

```
def grafikYontemi (maxDerece,katsayi,xSifir,deltaX,epsilon):
    while(deltaX > epsilon) :
        while ( denklemSonuc(xSifir, maxDerece, katsayi) * denklemSonuc((xSifir + deltaX),
maxDerece, katsayi) >= 0 ):
            xSifir = xSifir + deltaX
        deltaX = deltaX / 2
        print("deltaX: ",deltaX)
        print("\n\nBULUNAN YAKLASIK KOK : ", xSifir)
```

Şekil 4.1.3 Grafik Yöntemi Kodlaması

4.2 Bisection (Yarıya Bölme) Yöntemi

f(x) = 0 şeklinde bir denkleme bakıldığında, f(x) fonksiyonu [a,b] aralığında sürekli ve f(a) × f(b) < 0 ise f(x) fonksiyonunun (a,b) aralığında bir yada birden fazla kökü vardır denilir. Bu yöntemle birden fazla kök de bulunabilir. Fakat genelde tek bir kök çözüm için yeterli gelmektedir.



Şekil 4.2.1 f(x) denkleminin grafiği

f(a) x f(b) < 0 olduğu için (a,b) aralığında en az bir kök vardır.

1.iterasyon :

$$x1 = \frac{a+b}{2} \qquad (4.2.1)$$

2. iterasyon: Eğer f(a) x f(x1) < 0 ise;

$$x2 = \frac{a+x1}{2} \qquad (4.2.2)$$

Değilse;

$$x2 = \frac{b+x1}{2} \qquad (4.2.3)$$

3. iterasyon: Eğer f(a) x f(x1) < 0 ise;

$$x3 = \frac{a+x2}{2} \qquad (4.2.4)$$

Değilse;

$$x3 = \frac{b+x2}{2} \qquad (4.2.5)$$

İterasyonlar istenen hata aralığına (epsilon) ulaşıncaya kadar devam ettirilir. Epsilon ne kadar küçükse bulunan kök okadar gerçeğe yakın bir sonuç olur.

Örnek: f(x) = x4−9x3−2x2 +120x−130

Bu fonksiyonun (1,2) aralığında bir köke sahip olduğu bilgisi verilmektedir. Bu kökü epsilonu 0,0132 alarak yaklaşım hatası ile bulunuz.

Çözüm:

a = 1 ve b = 2
a = 1 -> f(1) = −20 b = 2 -> f(2) = 46

1.Adım :

f(a) × f(b) = (−20) × 46 < 0 olduğu için bu aralıkta en az bir kök vardır.

$x1 = \frac{a+b}{2} = \frac{(1+2)}{2}$ = 1.5

f(1,5) = 20.2

b = x1 = 1.5

2.Adım :

f(a) × f(b) < 0 olduğu için

$x2 = \frac{a+b}{2} = \frac{(1+1.5)}{2}$ = 1.25

f (1.25) = 1.8

Bu durumda b = $x2$ = 1.25 dir.

3.Adım :

f(a) × f(b) < 0 olduğu için

$x3 = \frac{a+b}{2} = \frac{(1+1.25)}{2}$ = 1.1875

f(1.1875) = −3.4028

a = $x4$ = 1.1875 dir.

5.Adım :

f(a) × f(b) < 0 olduğu için

$x5 = \frac{a+b}{2} = \frac{(1.1875+1.25)}{2}$ = 1.21875

f(1.21875) = −0.80688

a = x5 = 1.21875 dir.

6.Adım :

f(a) × f(b) < 0 olduğu için

$x6 = \frac{a+b}{2} = \frac{(\ 1.21875\ +1.25\ )}{2}$ = 1.234375

f(x6) = 0,472092

b = x6 = 1.234375 olur.

$ey = |\frac{1.234375-1.21875}{1.234375}|$ = 0.01265

Not: f ∈ C [a, b] ve f(a) $*$ f(b) < 0 olsun. Bu duruma yarıya bölme yöntemi ile f fonksiyonunun kökü xk 'ya yaklaşan bir {x$_n$ }$^{\infty}{}_n$=1 dizisi oluşturur.

Oluşabilecek maksimum hata ise;

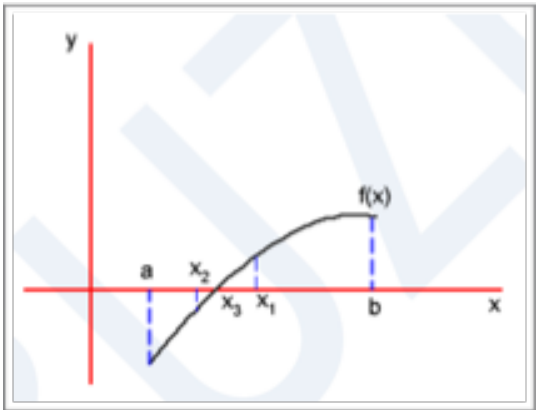
$$e \leq \frac{b-a}{2^n} \qquad (4.2.6)$$

şeklinde hesaplanır.

```
def BisectionYontemi (derece,katsayi,a,b,epsilon) :

    c = (a + b) / 2
    flag = 1

    while ( abs(denklemSonuc(c, derece, katsayi)) > epsilon and flag==1):
        if (denklemSonuc(c, derece, katsayi) * denklemSonuc(a, derece, katsayi) < 0) :
            b=c
        elif( denklemSonuc(c,derece,katsayi) * denklemSonuc(b,derece,katsayi) < 0 ):
            a=c
        else :
            print("\n! Bu aralikta kok yok.Baska bir (a,b) araligi giriniz !\n")
            flag = 0
        c = (a + b) / 2
        print("c=  \n", c)
    print("\n BULUNAN YAKLASIK KOK :  ", c)
```

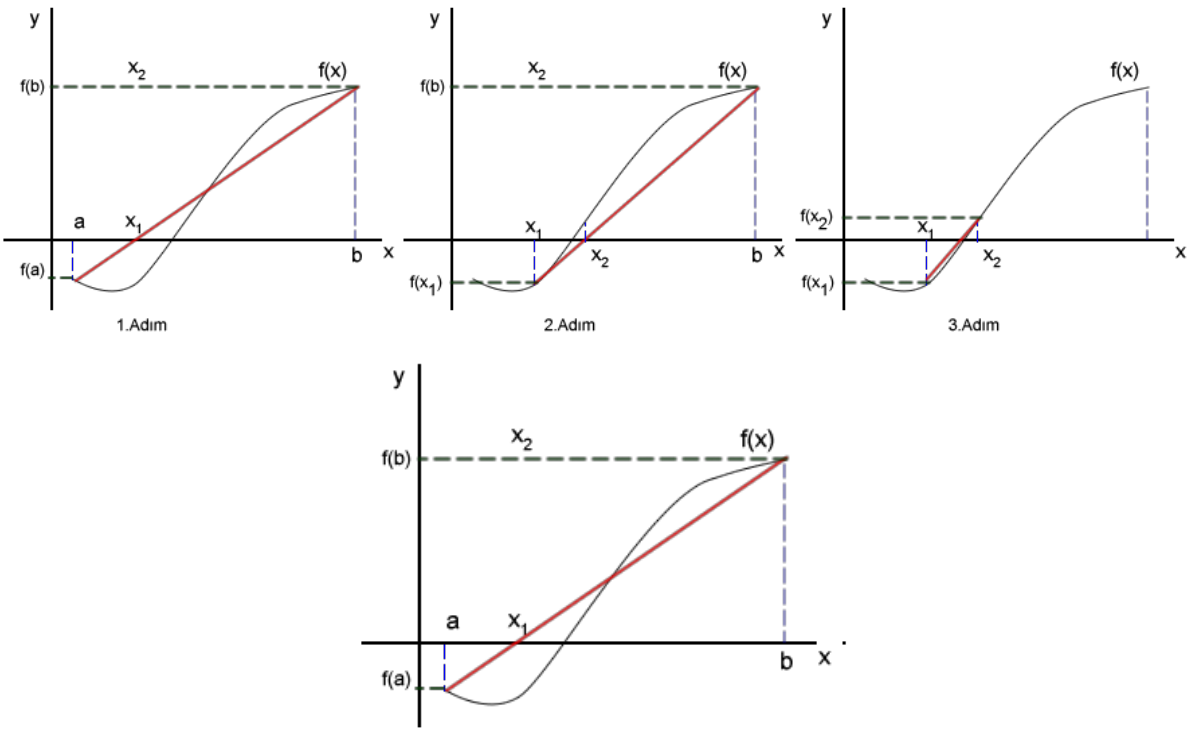
Şekil 4.2.2 Bisection Yöntemi kodlaması

4.3. Regula False (Lineer İnterpolasyon) Yöntemi



Şekil 4.3.1 f(x) denkleminin grafiği

Bazı problemlerin çözümünde yarıya bölme(bisection) yöntemi oldukça uzun zaman alır. Bu problemlerin çözümünü hızlandırmak için Regula False yöntemi tercih edilebilir. Kökün hesaplanması adımı dışında her şey Yarıya Bölme yöntemi ile aynı şekilde ilerler. Grafiksel olarak incelendiğinde üçgenlerin benzerliği kullanılarak kök bulunur.



Şekil 4.3.2 Regula false yönteminde izlenecek adımların örnek grafik görüntüleri

f ∈ a,b ;

Üçgenlerin benzerliği kuralından;

$$\frac{(b-x1)}{(b-a)} = \frac{f(b)}{f(b)-f(a)} \quad (4.3.1)$$

$$x1 = \mathrm{b} - \frac{b-a}{\mathrm{f(b)-f(a)}} f(b) \quad (4.3.2)$$

$$x1 = \frac{a*f(b)-b*f(a)}{\mathrm{f(b)-f(a)}} \quad (4.3.3)$$

şeklinde hesaplama yapılır. Bu işlemler yarıya bölme (bisection) yöntemindeki gibi adım adım istenen epsilon değerine (hata payı) ulaşıncaya kadar devam ettirilir ve en doğru kök bulunmaya çalışılır.

```
def regulaFalse(derece,katsayi,a,b,epsilon) :

    c = (b * denklemSonuc(a, derece, katsayi) - a * denklemSonuc(b, derece, katsayi)) / (
            denklemSonuc(a, derece, katsayi) - denklemSonuc(b, derece, katsayi))
    flag = 1
    while (abs(denklemSonuc(c, derece, katsayi)) > epsilon and flag == 1):
        if (denklemSonuc(c, derece, katsayi) * denklemSonuc(a, derece, katsayi) < 0):
            b = c
        elif( denklemSonuc(c, derece, katsayi) * denklemSonuc(b, derece, katsayi) < 0 ):
            a = c
        else :
            print("\n! Bu aralikta kok yok. Baska bir (a,b) araligi giriniz !\n\n")
            flag = 0
        c = (b * denklemSonuc(a, derece, katsayi) - a * denklemSonuc(b, derece, katsayi)) / (
                denklemSonuc(a, derece, katsayi) - denklemSonuc(b, derece, katsayi))

        print("c= %f \n", c)

        print("\n\n BULUNAN YAKLASIK KOK : %lf \n", c)
```

Şekil 4.3.3 Regula False Yöntemi kodlaması

4.4 Newton Raphson Yöntemi

Grafik yöntemlerinden sonra en çok tercih edilen kök bulma yöntemidir. Taylor serisi veya grafik yöntemi kullanılarak çözüme ulaşılabilir.

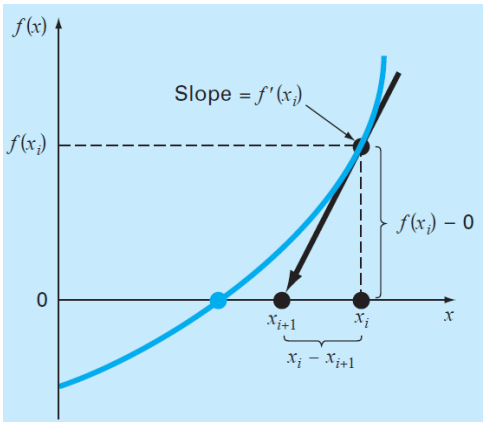
Grafik Yöntemi:

Şekilde görüldüğü gibi eğer kökün ilk tahmini xi ise, [xi , f(xi)]noktasındaki teğet uzatılabilir. x eksenini kestiği nokta çoğunlukla kökün daha iyi bir tahminidir. xi noktasındaki birinci türev fonksiyonun eğimine eşittir;

$$f'(x_i) = \frac{f'(x_i)-0}{x_i - x_{i+1}} \qquad (4.4.1)$$

Bu ifade düzenlenerek, formül elde edilir;

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \qquad (4.4.2)$$



Şekil 4.4.1 f(x)'in görüntüsü ve x_{i+1}'in bulunma yönteminin grafik üzerinde gösterimi

Taylor Serisi Yaklaşımı:

$$f(x_{i+1}) = f(x_i) + f'(x_i)(x_{i+1} - x_i) + \frac{f''(\zeta)}{2!}(x_{i+1} - x_i)^2 + \cdots$$

Şekil 4.4.2 f(x+1)'in Taylor Serisi Açılımı

Birinci türevden sonraki kısmı almazsak:

$$f(x_{i+1}) \cong f(x_i) + f'(x_i)(x_{i+1} - x_i)$$

Şekil 4.4.3 f(x+1)'in Taylor Serisi Açılımı kısaltılmış hali

x_{i+1} eğer fonksiyonu sağlayan bir kök ise f(x_{i+1}) = 0 olur.

$$0 \cong f(x_i) + f'(x_i)(x_{i+1} - x_i)$$

Şekil 4.4.4 f(x_i)'nin denklemi

Şekil 4.4.4'deki denklemi düzenlersek,

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Şekil 4.4.5 f(x_{i+1})'in denklemi

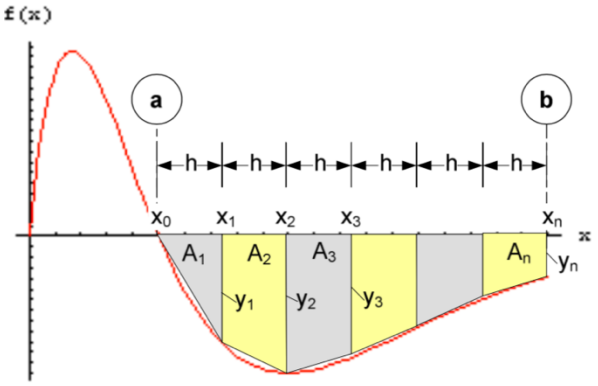
Şekil 4.4.5'deki ifade elde edilir.

```
def newtonRaphson(derece,katsayi,x,epsilon):

    guncelX = x
    x = x - denklemSonuc(x, derece, katsayi) / turev(x, derece, katsayi)

    while(abs( denklemSonuc(x,derece,katsayi) - denklemSonuc(guncelX,derece,katsayi) )
> epsilon ):
        guncelX = x
        x = x - denklemSonuc(x, derece, katsayi) / turev(x, derece, katsayi)
    print("\n\n BULUNAN YAKLASIK KOK :   ",x)
```

Şekil 4.4.6 Newton Raphson Yöntemi kodlaması

5. İNTEGRAL HESAPLAMA YÖNTEMLERİ

5.1 Trapez(yamuk) Kuralı



Şekil 5.1.1 Yamuk kuralının fonksiyon görüntüsü

Kuralın isminden de anlaşılacağı üzere alan yamuklar aracılığı ile belirlenir. a,b aralığı n eşit aralığa bölünür.

h = $\frac{b-a}{n}$ 'dir. (4.5.1)

x_i ye karşılık gelen $y_i = f(x_i)$ noktası hesaplanır.

Ai = h* $\frac{y_i+y_{i+1}}{2}$dir. (4.5.2)

Toplam alan :

$$Alan = I = \int_a^b f(x)dx \approx h\frac{y_0+y_1}{2} + h\frac{y_1+y_2}{2} + \ldots + h\frac{y_{n-1}+y_n}{2}$$

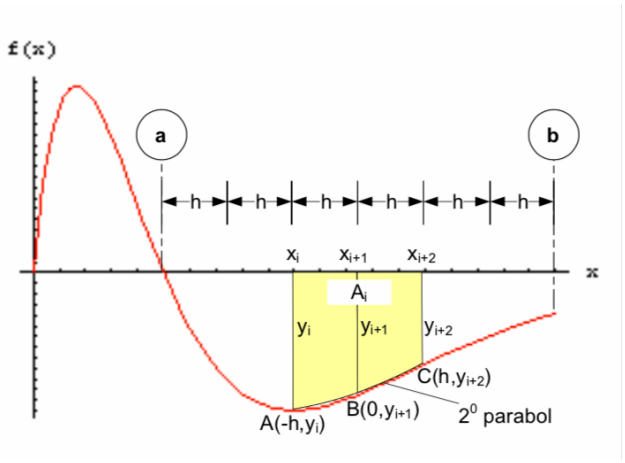
$$Alan = I \approx h(\frac{1}{2}y_0 + y_1 + y_2 + y_3 + \ldots + \frac{1}{2}y_n = h(\frac{1}{2}y_0 + \sum_{i=2}^{n-1} y_i + \frac{1}{2}y_n)$$

Şekil 5.1.2 Yamuk kuralını betimlemek için gerekli fonksiyon görüntüsü

```
def trapez(a, b, n, derece, katsayilar) :

    h = (b - a) / n
    sigma = 0

    for x in range(a+h , b-h-1 , h ):
        sigma += denklemSonuc(x, derece, katsayilar)

    integral = h * ((denklemSonuc(a, derece, katsayilar) +
denklemSonuc(b, derece, katsayilar)) / 2 + sigma)
    return(abs(integral))
```

Şekil 5.1.3 Trapez kuralı kodlaması

5.2 Simpson Kuralı



Şekil 5.2.1 Simpson kuralını betimlemek için gerekli fonksiyon görüntüsü

a,b aralığı n eşit aralığa bölünür, burada n çift bir tam sayı olmalıdır.
h = (b-a) / n dir.
$x_{i+2} - x_i = 2h$ aralığında f(x) in 2. derece bir polinom olduğu kabul edilir.

y_i, y_{i+1}, y_{i+2} noktalarının A, B ve C tepe noktalarından geçen $y = ax^2 + bx + c$ parabolünün a, b ve c sabitleri x_i ye karşılık gelen $y_i = f(x_i)$ noktaları vasıtasıyla bulunur.

a, b ve c nin daha kolay hesaplanması için; y ekseni B noktasına doğru kaydırılırsa A, B, C noktalarının koordinatları A(-h,y_i), B(0,y_{i+1}), C(h,y_{i+2}) olur. Bu koordinatlar denklemi sağlar.

$$A(-h, y_i) \rightarrow y_i = ah^2 - bh + c$$
$$B(0, y_{i+1}) \rightarrow y_{i+1} = c$$
$$C(h, y_{i+2}) \rightarrow y_{i+2} = ah^2 + bh + c$$

$$\rightarrow \quad a = \frac{1}{2h^2}(y_i - 2y_{i+1} + y_{i+2})$$
$$b = \frac{1}{2h}(y_{i+2} - y_i)$$
$$c = y_{i+1}$$

Şekil 5.2.2 A,B ve C tepe noktalarının y parabolüne uyarlanarak a,b ve c değerlerinin hesaplanması

A_i alanı:

$$A_i = \int_{x_i}^{x_{i+2}} f(x)dx \approx \int_{-h}^{h} (ax^2 + bx + c)dx = \frac{2ah^3}{3} + 2ch$$

Şekil 5.2.3 Ai ' nin alanı

a, b ve c değerlerini yerine koyarsak;

$$A_i \approx \frac{h}{3}(y_i + 4y_{i+1} + y_{i+2})$$

Şekil 5.2.4 Ai ' nina,b ve c değerleri yerine konularak hesaplanışı

Şekil 5.2.4'teki ifade elde edilir. Bu şekilde Simpson kuralı elde edilmiş olur.

$$Alan = I = \int_a^b f(x)dx \approx A_1 + A_2 + ... + A_i + ... + A_n = \sum_{i=1}^{n} A_i$$

Şekil 5.2.5 Simpson kuralı

Toplam alan ise, Şekil 5.2.5'de verildiği gibidir.

```
def simpson(a, b, n, derece, katsayi) :

    h = (b - a) / n
    sigmaTek = 0
    sigmaCift = 0

    for x in range(a + h, b - h-1, 2*h):
        sigmaTek += denklemSonuc(x, derece, katsayi)

    for x in range(a + 2*h, b - h-1, 2*h):
        sigmaCift += denklemSonuc(x, derece, katsayi)

    integral = (h / 3) * ((denklemSonuc(a, derece, katsayi) +
denklemSonuc(b, derece, katsayi)) + 4 * sigmaTek + 2 * sigmaCift)


    return(abs(integral))
```

Şekil 5.2.6 Simpson kuralı kodlaması

6. TÜREV HESAPLAMA YÖNTEMLERİ

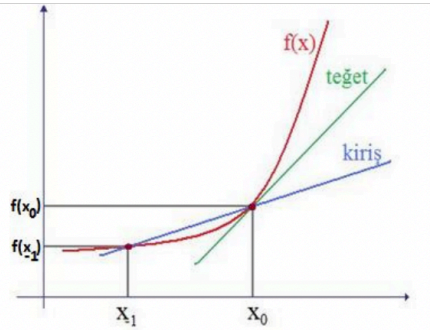
6.1 Sayısal Türev

Türev genel bir tanımla, bir fonksiyonun grafiğine çizilen bir teğetin eğimini hesaplar. Analitik olarak hesaplanması zor olan bir türev veya integralde sayısal türev veya integral yöntemlerine başvurulur ve çok daha kolay ve hızlı şekilde çözüm sağlanır.

Geometrik olarak bakıldığında türev, yukarıda yapılan genel tanım itibariyle bir fonksiyon için çizilen eğrinin herhangi bir noktadaki x koordinatınıyla arasındaki açı yada diğer bir deyişle x noktasındaki teğetinin eğimidir.

$$f(x)' = \frac{\partial f(x)}{\partial x} = \lim_{\Delta x \to 0} \frac{\Delta f(x)}{\Delta x}$$

$$\frac{\Delta f(x)}{\Delta x} = \frac{f(x_i + \Delta x) - f(x_i)}{\Delta x}$$



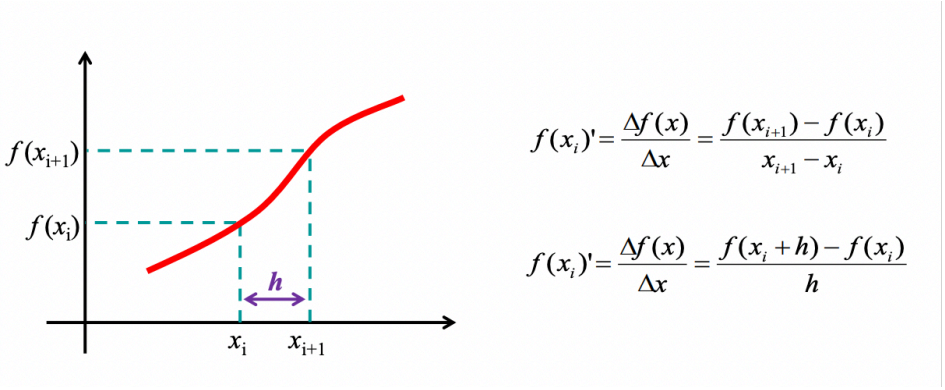
Şekil 6.1.1 Sayısal türevin tanımı ve fonksiyon görüntüsü

Sayısal türev, bir fonksiyonun tanımlandığı değişkenlere göre değişim hızının bir ölçüsüdür.

```
def turev(x,derece,katsayi) :
    turev  = np.zeros(50)
    sonuc = 0
    for i in range(derece) :
        turev[i] = katsayi[i + 1] * (i + 1)

    for i in range(derece) :
        sonuc += turev[i] * pow(x,i)
    print("türev sonucu:", sonuc)
    return sonuc
```

Şekil 6.1.2 Türev yönteminin kodlaması

6.2 İleri Fark Türev

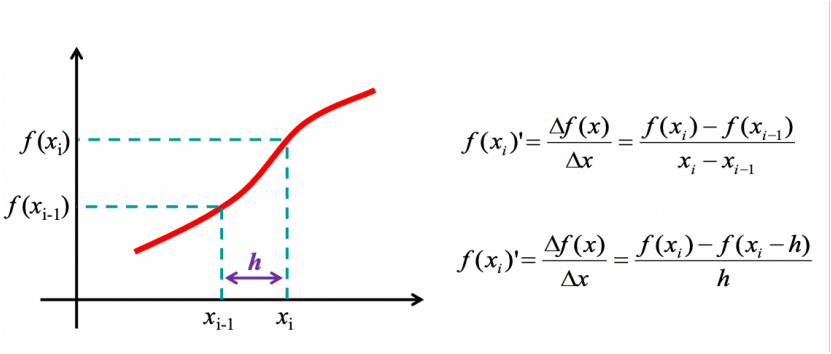


Şekil 6.2.1 İleri Fark Türev

```
def ileriFarkTurev(x, deltaX, derece, katsayi):
    turev = (denklemSonuc((x + deltaX), derece, katsayi) -
denklemSonuc(x, derece, katsayi)) / deltaX
    print("\n* Denklemin", x, "noktasindaki ileri fark turevi =  \n",
turev)
```

Şekil 6.2.2 İleri Fark Türev kodlaması

6.3 Geri Fark Türev

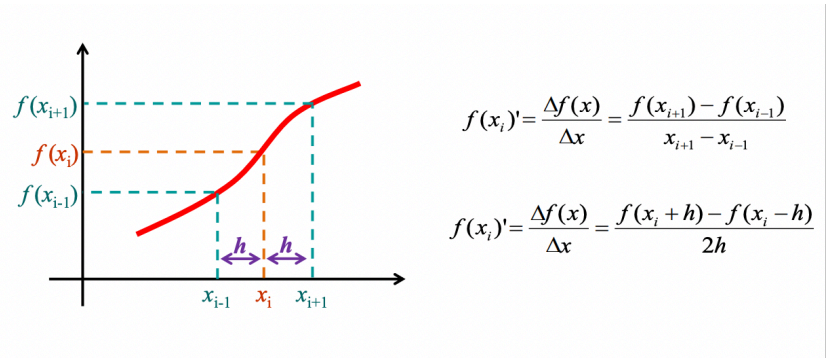


Şekil 6.3.1 Geri Fark Türev

```
def geriFarkTurev(x, deltaX, derece, katsayi):
    turev = ( denklemSonuc( x, derece, katsayi )- denklemSonuc((x-
deltaX), derece, katsayi) ) / deltaX
    print("\n* Denklemin", x, "noktasindaki geri fark turevi =  \n",
turev)
```

Şekil 6.3.2 Geri Fark Türev kodlaması

6.4 Merkezi Fark Türev



Şekil 6.4.1 Merkezi Fark Türev

```
def merkeziFarkTurev(x, deltaX, derece, katsayi):
    turev = (denklemSonuc((x + deltaX), derece, katsayi) -
denklemSonuc((x - deltaX), derece, katsayi)) / (2 * deltaX)
    print("\n* Denklemin", x, "noktasindaki merkezi fark turevi =  \n",
turev)
```

Şekil 6.4.2 Merkezi Fark Türev kodlaması

7. DOĞRUSAL DENKLEM TAKIMLARININ ÇÖZÜMÜ

7.1 Doğrusal Denklem Takımlarının Çözüm Yöntemleri

Ele alacağımız yöntemler gauss eliminasyon ve gauss jordan yöntemleridir. Bu yöntemler büyük denklem sistemlerinin çözümünde oldukça kolaylık sağlamaktadır. Boyutu 3x3 olan bir matrisi ele aldığımızda çok hızlı olmamasına rağmen, örneğin boyutu 20×20 olan bir matrisin çözümünde oldukça hızlıdır ve işi büyük ölçüde kolaylaştırır. Bu yöntemler özellikle de bir bilgisayar programı olarak yazılıp kullanıldığında çok hızlı ve kolay bir şekilde çözüme ulaştıran yöntemlerdir.

7.2 Gauss Yok Etme (Gauss Eliminasyon) Yöntemi

Gauss eliminasyon yöntemini bir örnek yardımıyla göstermek en kolay anlatma yöntemi olacaktır. Bir örnek üzerinden ilerleyelim:

$$x+2y+2z=11 \quad x+2y+2z=11$$
$$3x+2y-z=4 \quad 3x+2y-z=4$$
$$2x-y+z=3 \quad 2x-y+z=3$$

Şekil 7.2.1 Verilen denklemlerin matris formu

Şekil 7.2.1'de görülen denklem takımlarını Gauss Yok Etme metoduyla çözümlemek için denklemleri matris formunda yazmalıyız. (şekil 7.2.2)

$$\left[\begin{array}{ccc|c} 1 & 2 & 2 & 11 \\ 3 & 2 & -1 & 4 \\ 2 & -1 & 1 & 3 \end{array}\right] \left[\begin{array}{ccc|c} 1 & 2 & 2 & 11 \\ 3 & 2 & -1 & 4 \\ 2 & -1 & 1 & 3 \end{array}\right]$$

Şekil 7.2.2 Verilen denklemlerin matris formu

Denklemlerin x, y ve z değerlerinin kat sayılarıyla oluşturulan matris bu şekilde olmalıdır. Oluşturulan matrisin satır ve sütunlarına lineer cebir yöntemlerinden uygulanarak alt üçgen matris şekline dönüştürülür.(Şekil 7.2.3' te verilen matrise benzetilir)

$$\left[\begin{array}{ccc|c} 1 & * & * & * \\ 0 & 1 & * & * \\ 0 & 0 & 1 & * \end{array}\right] \left[\begin{array}{ccc|c} 1 & * & * & * \\ 0 & 1 & * & * \\ 0 & 0 & 1 & * \end{array}\right]$$

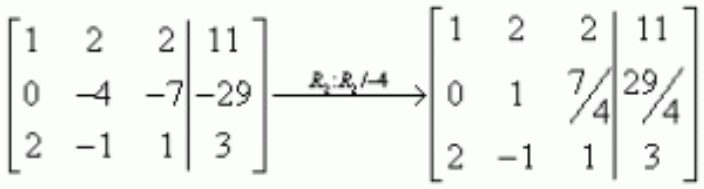
Şekil 7.2.3 Verilen matrisin getirilmesi gereken matris formu

Şekil 7.2.2'de verilen matrisin birinci satırının birinci sütunu zaten 1 olduğu için bu satıra herhangi bir işlem yapmamıza gerek yoktur. İşleme ikinci satırın ilk sütunu "0" yapılarak başlanılır. İncelendiği zaman ikinci satırın ilk elemanının 3 olduğu görülür. İlk satırı kullanarak bu 3 değeri 0 yapılır. Bunu yapmak için 7.2.4 de görüldüğü gibi 1. satırın tamamı 3 ile çarpılıp 2. satırdan çıkartılır. Burada dikkat edilmesi gereken nokta yapılan işlemin bütün satıra uygulanmasıdır. Aksi takdirde yanlış bir sonuçla karşılaşırız.

$$\left[\begin{array}{ccc|c} 1 & 2 & 2 & 11 \\ 3 & 2 & -1 & 4 \\ 2 & -1 & 1 & 3 \end{array}\right] \xrightarrow{R_2 : R_2 - 3R_1} \left[\begin{array}{ccc|c} 1 & 2 & 2 & 11 \\ 0 & -4 & -7 & -29 \\ 2 & -1 & 1 & 3 \end{array}\right]$$

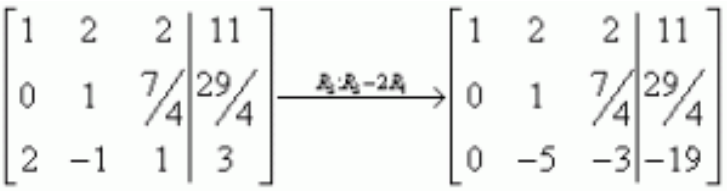
Şekil 7.2.4 Matris üzerinde yapılan çıkarma işlemi

İkinci satırın birinci elemanını 0 yaptık, şimdi ise köşegen üzerindeki -4'ü 1 yapmalıyız. Bunu yapmak için Şekil 7.2.5'de görüldüğü gibi ikinci satırın tamamı -4'e bölünür. 2.satır 7.2.3'de verilen forma getirildikten sonra 3. satır da aynı forma sokulur.



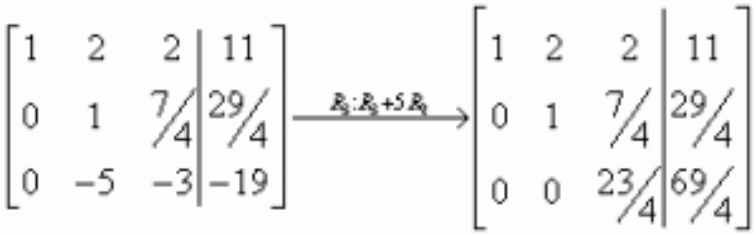
Şekil 7.2.5 Matris üzerinde değiştirme işlemi

3. satırın birinci elemanını 0 yapmak için 1. satırı kullanmak oldukça mantıklı. 3. satırın birinci elemanı 2 ve ilk satırın birinci elemanı 1 olduğu için 1. satırın tamamı 2 ile çarpılarak 3. satırdan çıkartılır. (7.2.6)



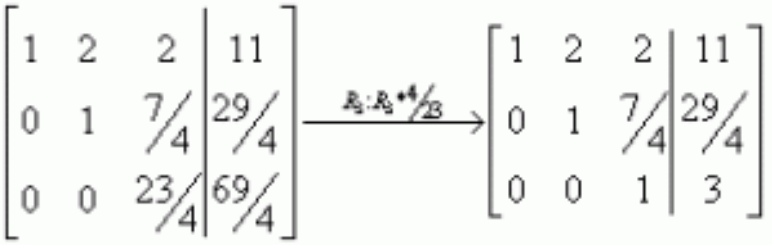
Şekil 7.2.6 Matris üzerinde çıkarma işlemi

Matrisi Şekil 7.2.3'deki forma dönüştürmek için Şekil 6'daki 3. satırın 2. elemanı olan -5, 0 olmalıdır. Bu durumda 2. satırı 5 ile çarpıp 3. satıra ekleriz ve 0 olur .(Şekil 7.2.7)



Şekil 7.2.7 Matris üzerinde toplama işlemi

Sırada matrisi Şekil 7.2.3'deki forma dönüştürmek için ihtiyacımız olan adım var. 3. satırın son elemanı 1 olmalıdır. 3. satır 4/23 ile çarpılır ve 3.satırın son elemanı da 1 olur. (şekil 7.2.8)



Şekil 7.2.8 Matris üzerinde çarpma işlemi

Son elde ettiğimiz matris benzetmeye çalıştığımız Şekil 7.2.3 teki matris formundadır. Dolayısıyla gereken satır işlemleri tamamlanmıştır. Şimdi yapılması gereken şey matris formundaki denklemleri tekrar denklem formuna geçirmektir. Şekil 7.2.8'de verilen matriste denklemlerin katsayıları görülmektedir. Bunları denklemler şeklinde yazarsak şekil 7.2.9' da verilen denklem sistemini elde etmiş oluruz. Burada z'nin değeri 3 olarak bulunmuştur. Şimdi bulunan z değeri yerine konularak tek tek denklemler çözülür.

$$x+2y+2z=11 \qquad x+2y+2z=11$$
$$y+\frac{7}{4}z=\frac{29}{4} \qquad y+\frac{7}{4}z=\frac{29}{4}$$
$$z=3 \qquad z=3$$

Şekil 7.2.9 Matrislerin denklem haline getirilmesi

Şekil 7.2.9'da verilen 2. denklemde hesaplanan z sayısı yerine konulur ve y hesaplanır. İşlemler yapıldığında y değeri 2 bulunur. (Şekil 7.2.10)
Artık elimizde hem z hem y vardır ve tek çözülmesi gereken x kalmıştır. İlk denklemde z ve y yerlerine konursa (şekil 7.2.11), x değeri de bulunmuş olur. Bulunan x,y ve z değerleri yerlerine koyularak sağlama işlemi yapılır.

$$y+\frac{7}{4}3=\frac{29}{4} \qquad y+\frac{7}{4}3=\frac{29}{4}$$
$$y=\frac{8}{4}=2 \qquad y=\frac{8}{4}=2$$

Şekil 7.2.10 Denklemlerde z değerinin yazılması

$$x+2*2+2*3=11$$
$$x+4+6=11$$
$$x=1$$

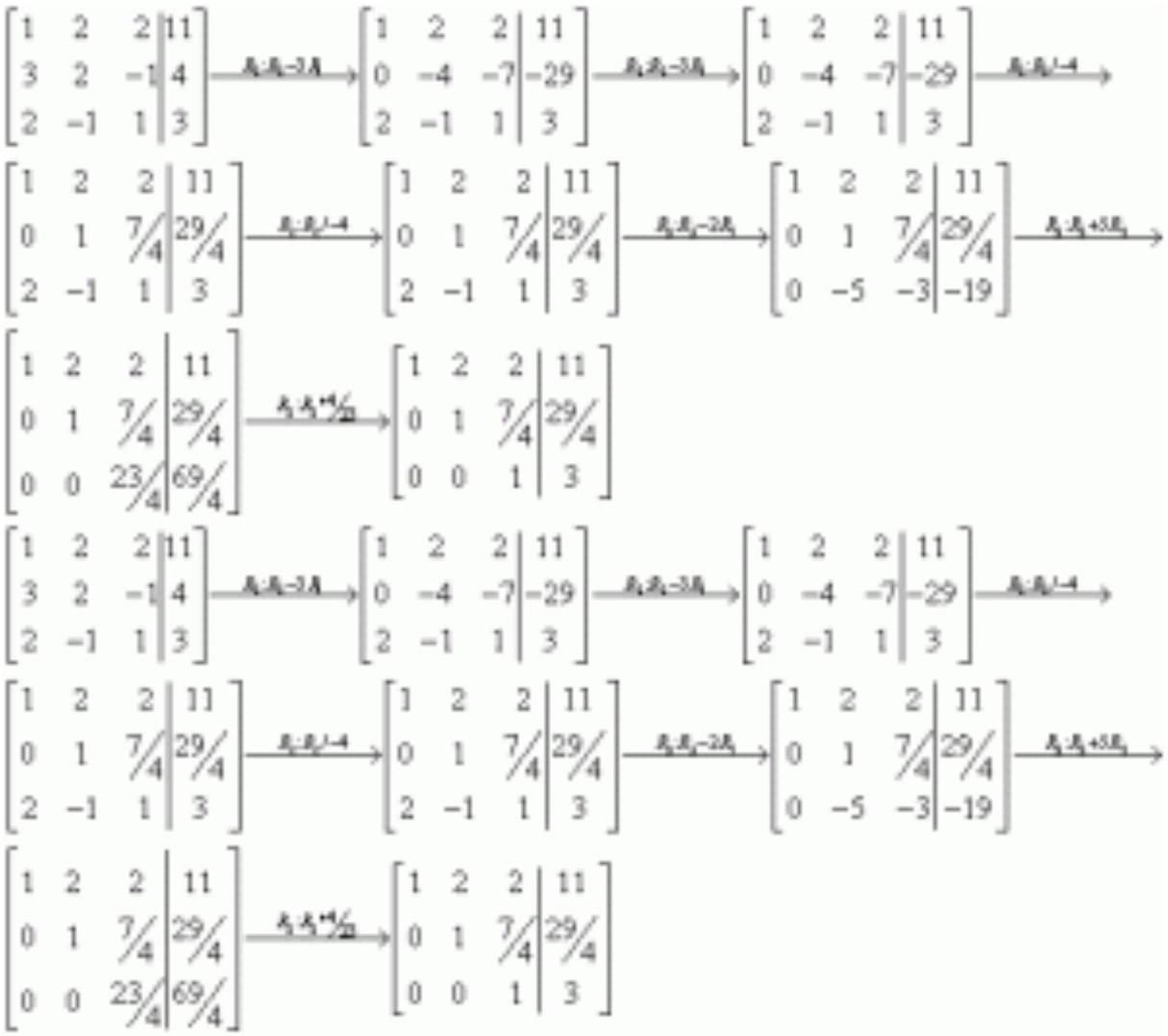
Şekil 7.2.11 Denklemlerde x değerinin bulunması

Bu şekilde x, y ve z değerleri Gauss Eliminasyon (Gauss Yok Etme) yöntemiyle hesaplanır. Elle hesaplandığında çok uzun işlemler gibi görünse de bilgisayar aracılığıyla bir program yazılıp hesaplandığında oldukça pratiktir ve çok hızlıdır.

7.3 Gauss Jordan Yöntemi

Gauss Jordan ve Gauss Eliminasyon yöntemleri birbirlerine çok benzeyen yöntemlerdir. Gauss Jordan yönteminde de aynı Gauss Eliminasyondaki gibi verilen denklem sistemi matris formuna dönüştürülür ve çeşitli matris işlemleri uygulanarak lineer(doğrusal) denklem takımlarının kökleri bulunur.
Bu yöntemin farklı tarafı denklemin bilinmeyenleri hesaplanırken matris satır işlemleri bittiğinde matrisi tekrar denklem formuna dönüştürüp bulunan kökleri tek tek yerine koyarak çözüme ulaşmak yerine denklemler matris formunda bırakılır ve matris satır işlemleri uygulanmaya devam edilerek çözüme ulaşılır.

Gauss Yok Etme metodunda kullanılan örnek üzerinden devam edersek;

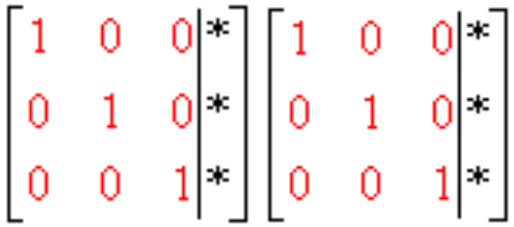


Şekil 7.3.1 Gauss yok etme yöntemi adımları

Gauss eliminasyon yönteminde, verilen denklemler matris formuna getirilip, bu matrisin Şekil 7.2.3'de verilen matris formuna benzemesi için şekil 7.2.4 'ten şekil8'e kadar olan işlemler sıra sıra yapılmıştı.(Şekil 7.2.12)

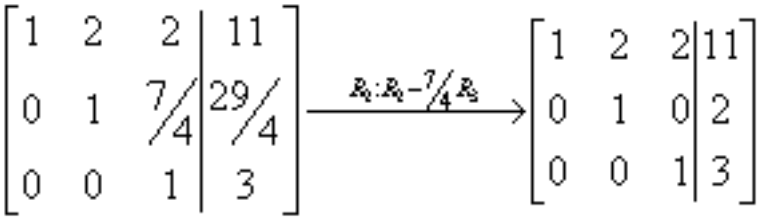
Gauss Jordan yönteminde ise verilen denklemleri matris formuna getirip, Şekil 7.3.2'te verilen forma dönüştürülür.

Şekil 7.3.2'de verilen matris Şekil 7.2.3'de verilen matris ile oldukça benzerdir. Şekil 7.2.3'deki matriste köşegen olan elemanların değerleri 1, köşegen altında kalan elemanlar ise 0'lardan oluşmaktadır. Gauss Jordan yönteminde benzetilmesi gereken matrisin ise köşegen olan elemanların değerleri 1, köşegenin üstünde kalan elemanlar ise 0'lardan oluşmaktadır.



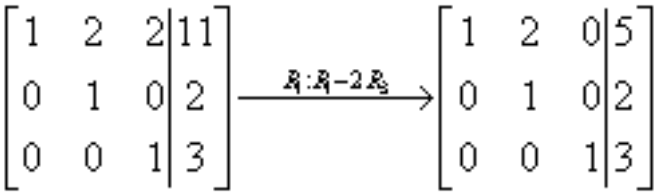
Şekil 7.3.2

Aşağıdaki şekildeki gibi 2. satırın 3. elemanını 0 yapmak için 3. satırı 7/4 ile çarpıp 2. satırdan çıkartmak yeterlidir. Bu durumda 2.satır Şekil 7.3.2'deki 2. satıra benzer. Şimdi 1. satır Şekil 7.3.2'deki 1. satıra benzetilmeye çalışılır.

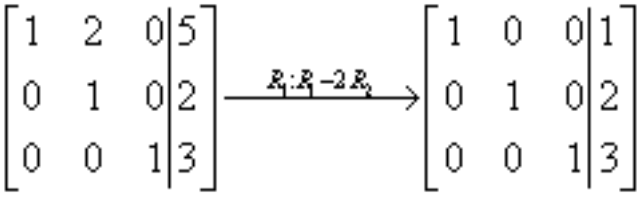


Şekil 7.3.3

3. satırı 2 ile çarpıp 1. satırdan çıkartırsak, 1. satırın 3. elemanını da 0 yapmış oluruz. Aynı şekilde 2. satırı da 2 ile çarpıp 1. satırdan çıkartırsak 1. satırın 2. elemanını da 0 yapmış oluruz. (Şekil 7.3.4)



Şekil 7.3.4



Şekil 7.3.5

Yapılan işlemlerin sonunda matris istenilen forma getirilmiştir. Şekil 7.3.5'de x, y ve z değerlerinin 1, 2 ve 3 olduğunu görebiliriz. Bu değerleri denklemde yerlerine yazarak çözümün sağlamasını yapabiliriz.

```
def GaussJordanYontemi(denklemler,dsayisi):

    for k in range(dsayisi):
        for i in range(dsayisi):
            a = denklemler[i][k]
            for j in range(dsayisi):
                if (i != k) :
                    denklemler[i][j] = (denklemler[k][k] *
denklemler[i][j]) - (a * denklemler[k][j])


    print( "\nSONUCLAR :\n" )
    for i in range(dsayisi):
        print("\nx",i + 1," = ",denklemler[i][dsayisi] / denklemler[i][i]
,"\n")
```

Şekil 7.3.6 Gauss Jordan Yönteminin Kodlaması

8. SONUÇLAR VE ÖNERİLER

Mühendislikte çözülecek olayların hızlı ve pratik şekilde çözülmesi esastır. Zaman kavramının önemi bilindiği için karşılaşılan problemlerin en az zaman ve efor harcanarak çözülmesi istenir. Sayısal analiz yöntemleri de, birçok alanda karşılaşılan ve çözümü oldukça zor olan ve zaman harcayan denklem sistemlerinin çözümünü en hızlı ve kolay şekilde çözebilen yöntemlerdir. Özellikle de bu işi yapan bir insan eli değil de bir bilgisayar olursa bu çok daha kusursuz ve hızlı çözümler demektir. Tek yapılması gereken bu yöntemlerin olabildiğince hatasız kodlanması ve bir program haline getirilmesidir. Ben de buna istinaden yaptığım bu çalışmada elimden geldiğince sayısal analiz yöntemlerini birçok başka programda veya işlerde kullanılmak üzere bir kütüphane geliştirdim. Kütüphaneyi en çok kullanılan dillerden biri olan Python dilinde kodladım. Her geçen gün değişen ve gelişen bilim ve teknolojiyle beraber bu kütüphanedeki yöntemler geliştirilebilir ve çok daha kusursuz, hata oranları çok daha az yöntemler de kodlanabilir..

Araştırma Problemleri dersi bünyesinde üzerinde çalışılan Sayısal Analiz Yöntemleri konusunda yapılan araştırmalarla beraber, aşağıda maddeler halinde belirtilen kısımlar Python dilinde yazılmıştır.

Bu proje kapsamında; Sayısal Analiz Yöntemlerini uygulayan bir kütüphane geliştirilmiştir ve bu kütüphanede mevcut Sayısal Analiz Yöntemleri aşağıda listelendiği gibidir.

1- NÜMERİK KÖK BULMA

i) Grafik Yöntemi
ii) Bisection Yöntemi
iii) Regula False Yöntemi
iv) Newton Raphson Yöntemi

2-NÜMERİK İNTEGRAL

i) Trapez Yöntemi ile İntegral Hesaplama
ii) Simpson Yöntemi ile İntegral Hesaplama

3-NÜMERİK TÜREV

i) İleri Fark Türev
ii) Geri Fark Türev
iii) Merkezi Fark Türev

4-DOĞRUSAL DENKLEM TAKIMLARININ ÇÖZÜMÜ

Gauss Jordan Yöntemi

KAYNAKLAR

[1] Richard L. Burden, Richard L. Burden (2009). ‚Numerical Analysis' Brooks/Cole Cengage Learning, Boston.

[2] Doç. Dr. İbrahim UZUN, (2004), "Numarik Analiz' Beta Yayıncılık.

[3] web.karabuk.edu.tr/yasinortakci/dokumanlar

[4] http://www.yildiz.edu.tr/~hbayir/Sayisal_yontemler_kitabi.pdf

[5] https://rasyonalist.org/yazi/sayisal-analiz-newton-raphson-metodu/

[6] http://mmf2.ogu.edu.tr/atopcu/index_dosyalar/Dersler/BilDesNuMAn/BDNA-DersNotlar%C4%B1/BDNA38_%C4%B0ntegrasyon.pdf

[7] http://bilisim.kocaeli.edu.tr/dosyalar/Dosyalar/DersNotlari/9-sayisal-turev.pdf

[8] utkudeniz.com

ÖZGEÇMİŞ

1998 yılında Amerika/Florida'da doğdu. İlk öğrenimini Hawaii'de, orta ve lise öğrenimini İstanbul'da tamamladı. 2017 yılında girdiği Yıldız Teknik Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü'nü kazandı. 2019 yılında Farabi Öğrenci Değişim Programı ile Kocaeli Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü'ne geçiş yaptı ve şuan hala burada öğrenim görmektedir.