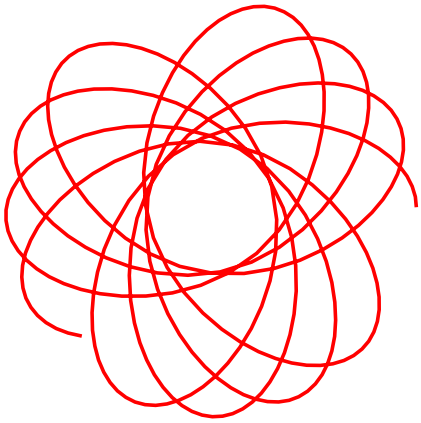


FİZİKTE SAYISAL YÖNTEMLER

Doç. Dr. Metin Özdemir
Çukurova Üniversitesi
Fizik Bölümü

ÖNSÖZ

Bu ders notları Fizik Bölümünde zaman zaman seçmeli olarak vermekte olduğum sayısal analiz dersinin hazırlanması sırasında ortaya çıkmıştır. Notlar eksiksiz olmaktan çok uzak olup sürekli bir değişim ve gelişme içerisindedir. Notların hazırlanmasındaki temel amaç konu ile ilgili hem daha düzenli bir kaynak yaratmak, hem de bu konuda varolan eksikliği gidermeye katkıda bulunmaktır.

Bu ders notları sayısal analiz konusunda tamamen teorik bir çalışma olmadığı gibi tamamen sayısal bir el kitabıda değildir. Konular üniversite birinci ve ikinci sınıf standart matematik derslerini takip etmiş biri için kolaylıkla izlenebilecek durumdadır. Ayrıca bu notları kullanacak olanların FORTRAN programlama dili ile ilgili giriş düzeyinde yeterli bilgiye sahip oldukları, fortran programlarını yazıp derleyebilecekleri gerekli yazılımların bulunduğu bir bilgisayara erişimlerinin bulunduğu kabul edilmiştir. Programlar FORTRAN 77 dilinde yazılmıştır. Programların büyük kısmı belirli bir problemi çözmek veya işlevi yerine getirmek üzere yazılmıştır. Çalışır durumdaki bu programlar eksiksiz olarak öğrenciye verilecektir. Öğrenci bu programları varsa değişik parametreler, başlangıç şartları vs. için çalıştırarak aşinalık kazanacak, aynı zamanda programın/kullanılan yöntemin iyi/kötü taraflarını keşfetmiş olacaktır. Daha sonra benzeri problemleri çözmek üzere bu programlarda gerekli değişikliklerin yapılması istenecektir.

Ele alınan konuların anlatımı ve bunların bilgisayarda pratik uygulaması beraberce yürütülecek şekilde tasarlanmıştır. Bu nedenle öğrencinin notları adım adım takip etmesi durumunda en iyi verim elde edilmiş olacaktır. Pratik uygulamaları daha sonra yapmak isteyenler veya bu durumun daha uygun olduğu durumlar için kullanılmak üzere her bölümün sonuna bir 'Labaratuvar Saati' eklenmiştir. Burada o bölüme ait pratik olarak bilgisayar başında yapılması gereken her şey tekrar edilmiştir. Bu nedenle bölüm içinde anlatılan konular ile labaratuvar bölümünde istenenler bazen tamamen aynı olabilir. Bu, tekrara düşme pahasına olsa da yapılmıştır.

Bu dersi daha önce almış olan ögrencilerin bu notların hazırlanmasına dolaylı veya dolaysız mutlaka katkıları olmuştur, onlara teşekkür ediyorum.

Eylül 2004

Metin Özdemir
Çukurova Üniversitesi
Fizik Bölümü
01330 Adana
metoz@cu.edu.tr

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

Bölüm 1

GİRİŞ

Bu bölümün temel amacı FORTRAN komutlarını basit pratik uygulamalar kullanarak yeniden hatırlatmaktır. özellikle SUBROUTINE ve FUNCTION alt programlarının nasıl kullanıldığını göstermek üzere basit örnek programlar sunulmuştur. Fortran komutlarının kısa bir özeti ve Fortran programa dili ile ilgili bazı önemli noktalar Ek-A'da verilmiştir. Ayrıca bu bölümde hata, hassasiyet ve kararlılık kavramları üzerinde kısaca durulmuştur. Uzun süredir programlamadan uzak kalmış veya daha önce programlama dersi almış fakat ayrıntılı uygulama fırsatı bulamamış olanların bu bölümü mutlaka izlemeleri önerilmektedir. Özellikle verilen örnek programların yazılıp derlenmesi başlangıç için çok önemlidir.

1.1 Giriş

Bilgisayarlar verilen sayıları yapıları gereği ancak sonlu hassasiyetle hafızalarında tutabilir ve bu nedenle ancak sonlu bir hassasiyetle işlem yapabilirler. Sayıların temsili *bit* (binary digit) ile veya bitlerin bir araya gelmesinden oluşan *byte* (8'li bitler) ile yapılır. Temsil edilen sayılar tam sayı (integer, fixed-point) olabileceği gibi gerçel sayı (real, floating point) da olabilirler. Sayıların temsilinde kullanılan sayı sistemi çoğunlukla *ikili* (binary) sayı sistemi olmakla beraber 16'lı veya 10'lu sistemlerde kullanılmaktadır.

Günlük hayatta kullandığımız onlu sayı sisteminde, örneğin 365 sayısı, aslında üç tane 100, altı tane 10 ve beş tane 1'in toplamından oluşur. Bu durumu

$$365 = 3 \cdot 100 + 6 \cdot 10 + 5 \cdot 1 = 3 \cdot 10^2 + 6 \cdot 10^1 + 5 \cdot 10^0 = (365)_{10}$$

şeklinde gösterebiliriz. Onlu sayı sisteminde 0, 1, ..., 9 rakamları vardır. Bütün sayılar 10'nun uygun katlarının uygun katsayılar ile çarpılması ve bunların toplanması ile elde edilir. 10 bu sistemin *taban* sayısıdır. Aynı sayıyı *ikili* sayı sistemine göre temsil etmek istersek ne yapmamız gerekir? ıkili sistemde de bütün sayılar, 2'nin uygun katlarının 0 veya 1 ile çarpımlarının toplanması ile elde edilecektir. O halde 7 sayısını ikili sistemde temsil etmek için 7 sayısının içinde 2'nin değişik katlarından kaç tane olduğunu bulmamız gerekir. Bunu verilen sayıyı sürekli 2'ye bölerek ve kalanları takip ederek yapabiliriz. Bu sayıyı

$$7 = 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = (111)_2$$

şeklinde temsil edebileceğimizi hemen görebiliriz. Benze şekilde 9 sayısı

$$9 = 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = (1001)_2$$

olarak temsil edilebilir. İkili sayı sisteminin görüldüğü gibi taban sayısı 2'dir. Birden küçük sayıların temsili ise kullanılan sayı sisteminin tabanının (onlu sistemde 10, ikili sistemde 2) negatif üsleri kullanılarak yapılır. Bunların ayrıntılarına burada girilmeyecektir.

ÖRNEK **1.1** *11 sayısını ikili ve dörtlü sayı sistemlerinde temsil ediniz.*

Çözüm: İkili sayı ssitemine göre

$$11 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = (1011)_2$$

olacaktır. Dörtlü sayı sisteminde temsil etmek için ise 0, 1, 2 ve 3 sayılarını kullanmamız gerekir. 4'ün uygun katları bu sayıların uygun olanları ile çarpılarak elde edilen sayılar toplanırsa, istenilen temsil bulunacaktır. Buna göre 11 sayısı içinde 2 tane 4^1 *ve 3 tane* 4^0 *olduğundan aranan temsil*

$$11 = 2 \cdot 4^1 + 3 \cdot 4^0 = (23)_4$$

olacaktır.

1.2 Hassasiyet, Hata ve Kararlılık

Sayıların bilgisayarlarda ancak sınırlı bir hassasiyetle temsil edilebilmesi nedeniyle ortaya çıkan hatalar kullanılan makineye bağlıdır. Bu durum 'makine hassasiyeti' kavramı ile açıklanmaya çalışılır ve değişik mertebedeki sayıları makine işlemcisinin bir birlerinde ayrılabilmesi yeteneğinin bir ölçüsü olarak düşünülebilir.

Makine hassasiyeti ϵ, 1.0 sayısına eklendiğinde 1.0'den farklı bir sayı veren en küçük sayının büyüklüğü olarak tanımlanır. Bu sayı kullanılan makineden makineye farklılıklar gösterebilir. 32 bitlik ikili sistemi kullanan kişisel bilgisayarların çoğunda tekli hassasiyet (single precision) kullanıldığında $\epsilon = 1.1 \times 10^{-7}$ kadardır. Çifte hassasiyet (double precision) kullanıldığında $\epsilon = 2.2 \times 10^{-16}$ mertebesindedir. Bilgisayarla yapılan hesaplamarda ϵ kadar bir hatanın yapılması muhtemeldir. Bu hata **yuvarlama hatası** (round-off error) olarak bilinir ve programcının bu hatayı azaltmak veya yoketmek için yapabileceği bir şey yoktur. Fakat aşağıda da görüleceği gibi bu hatanın etkilerini en aza indirmek için çeşitli tedbirler alınabilir.

Bu noktada bir yanlış anlamaya meydan vermemek için hemen belirtelim: ϵ bir makinede kullanılabilecek en küçük sayı demek değildir. Örneğin makine hassasiyeti 1×10^{-12} olan bir bilgisayarda 1×10^{-44} gibi bir sayı rahatlıkla kullanılabilir ve bu sayı ile işlemlerde yapılabilir. Fakat bu sayı örneğin 1'e eklendiğinde sonuç yine 1 olacaktır. Bilgisayar bu kadar farklı mertebedeki iki sayıyı ayırt etme kapasitesine sahip dağildir.

Diğer yanda, bilgisayar ile yapılan hesaplamaların çoğunda sürekli bir değişkenin seçilen belirli kesikli noktalarda değeri bulunur. Örneğin sayısal olarak türev veya integral alırken her ikisinin analitik tanımlarında geçen ve çok küçük bir aralığı temsil eden Δx büyüklüğünü sıfır limit değerine götürmek pratik olarak mümkün değildir. Bu nedenle sayısal olarak yapılan hesaplamaların çoğunda belirli bir hata yapılır. Bu hataya **kesme hatası** (truncation

error) denir. Kesme hatası genelde programcının kontrolünde olup, sayısal analizin hemen hemen tamamı bu hatayı en aza indirme veya kontrol altında tutma yöntemlerini ortaya çıkarmak üzerine kuruludur.

Hassasiyet sayısal olarak yapılan bir işlemin sonucunun gerçek değerine ne kadar yakın olduğu ile ilgili bir kavramdır. Kararlılık ise çalıştırılan programın ıraksamadan istikrarlı bir biçimde çalışması demektir. Burada önemli olan hassasiyetten çok hatalıda olsa işlemlerin sorunsuz yürütülebilmesidir. Bu nedenle kararlı çalışan bir programın çıktılarının da hassas olacağı sanılmamalıdır.

ALIŞTIRMA **1.1** *Hesap makinenizin 'makine hassasiyetini' bulunuz. Bunu yapmak için 1 sayısına örneğin* 10^{-5}*'den başlayarak gittikçe küçülen sayılar ekleyiniz. Bu işleme toplamın sonucu yine 1 sayısını verene kadar devam ediniz.*

ALIŞTIRMA **1.2** a) Kullandığınız bilgisayarın hassasiyetini bulmak için aşağıda verilen programı bilgisayarınızda bulunan bir editör kullanarak bir dosyaya yazınız. FORTRAN derleyicinizi kullanarak bu programı derleyiniz ve sonra çalıştırınız. Makinenizin hassasiyeti nedir?

b) Aynı programı, programda geçen 'REAL' komutunu, 'REAL*8' ile değiştirerek yeniden çalıştırınız. Makine hassasiyetinizde bir değişiklik meydana geldi mi?

```
      PROGRAM epsilon
*****
* epsilon.for - KullandIgInIz makinenin hassasiyetini (epsilon)
* bulmak amacI ile yazIlmIstIr
*****

      REAL eps, bir, bireps

      eps = 1.0
      bir = 1.0
  100 bireps = bir + eps
      IF( bireps .LE. bir) GOTO 200
      eps = 0.5 * eps
      GOTO 100
  200 eps = eps * 2.0
      WRITE(*,*)
      WRITE(*,*) '------------------- '
      WRITE(*,*) ' Makine Hassasiyeti: '
      WRITE(*,*) ' epsilon = ', eps
      WRITE(*,*) '------------------- '
      END
```

FORTRAN programlama dilinde tekli hassasiyet kullanılarak işleme tabi tutulacak değişkenler REAL komutu ile tanıtılır. Bu kategorideki hesaplamalar hafızada 32 bit (4 bayt) yer ayrılarak yapılır. REAL*8 ise değişkenin çifte hassasiyetle hesaplanacağını ilan eder ve bu durumda işlemler 64 bit (8 bayt) yer ayrılarak yapılır. Günümüzde bilgisayarların bilgi işleme hızları ve kapasiteleri çok arttığından bütün hesaplamaları çifte hassasiyette yapmak yerinde olur. Bu konuya yeri geldikçe ileride değinilecektir.

1.3 Hassasiyet Kaybı ve İşlem Önceliği

Yukarıda sözü edilen makine hassasiyetinin sınırlı olmasından ve aşağıda bahsedilecek işlem önceliği nedeniyle bilgisayarlarla işlem yaparken çok farklı mertebelerdeki sayıları bir araya getirmek doğru değildir. Bu nedenle işlemlerde en az hatayı yapmak için, örneğin bir çok sayı toplanıyorsa aynı mertebedeki sayıların gruplanarak toplanması daha uygundur. Veya sayıları küçükten büyüğe doğru sıralayarak, önce küçük sayılardan başlayarak toplama yapmak daha uygun olacaktır. Bu şekilde işlemlerde ortaya çıkabilecek hassasiyet kaybı önlenmiş olur. Örneğin makine hassasiyeti $\epsilon = 2 \times 10^{-6}$ olan bir makinede 1 sayısına bir milyon defa 10^{-6} sayısı eklenirse 2 yerine yine 1 sayısı elde edilir. Burada yapılması gereken önce küçük sayıları ekledikten sonra ortaya çıkan sayının daha büyük sayıya eklenmesidir.

Başka bir hassasiyet kaybı ise birbirlerine çok yakın sayıların bir birinden çıkartılmasında ortaya çıkar. Örneğin $x = 0.123$, $y = 0.124$ olsun. Bu sayıların her ikisinde de 3 anlamlı rakam vardır ve son haneleri en az hassas olan rakamlardır. Bu iki sayı bir birinden çıkartıldığında, her iki sayınında en az hassas olan basamaklarının farkı alınmış olur. Sonuç sadece bir anlamlı rakam içerir ve o da en anlamsız iki rakamın farkı olur. Bir birlerine çok yakın sayıların çıkartılması payda da ise yapılan hatanın etkisi daha da abartılmış olur. Bu nedenle mümkün olan her yerde bir birlerine yakın sayıların farklarının alınmasından kaçınılmalı, aynı işlemi yapmanın değişik yolları aranmalıdır.

ÖRNEK **1.2** *Aşağıda verilen işlemleri hassasiyet kaybı en az olacak şekilde yapınız.*
a) x sıfıra çok yakın bir sayı olmak üzere $y = 1 - \cos(x)$
b) x 1'e göre çok büyük bir sayı olmak üzere $y = \sqrt{x+1} - \sqrt{x}$

Çözüm:

a) x sıfıra çok yakın ise verilen sayılar bir birine çok yakın olacaktır. y'i eşleniği ile çarpıp bölersek

$$y = \frac{(1 - \cos(x))(1 + \cos(x))}{1 + \cos(x)} = \frac{\sin^2(x)}{1 + \cos(x)}$$

olur. Bu değer sağlıklı bir şekilde hesaplanbilir.

b) (a)'da kullanılan yöntemi kullanarak bir çözüm bulunuz.

ışlemler sırasında dikkat edilmesi gereken bir diğer konu *işlem önceliğidir.* FORTRAN da kullanılan artimetik işlemciler şunlardır: "toplama" (+), "çıkarma" (-), "çarpma" (*), "bölme" (/) ve "üs alma" (**). Bu işlemcilerin öncelik sırası ise şöyledir: 1. üs alma, 2. çarpma ve bölme, 3. toplama ve çıkarma. Aynı önceliğe sahip işlemlerde, parantez içindekilere öncelik tanınır. Sınırlı hassasiyet ve işlem önceliği nedeniyle bilgisayarla yapılan işlemlerde örneğin toplamanın değişme ve birleşme özellikleri sağlanmayabilir.

ÖRNEK **1.3** Aşağıda verilen aritmetik işlemlerin yapılması için FORTRAN programlama dilinde komut satırına bu değerler nasıl yazılmalıdır?
a) $a = \frac{b+c}{2}$, b) $a = \frac{b^3}{2\pi}$, c) $y = x - \frac{ax}{b^2 z^3}$

Çözüm: b)

```
PI=3.1415
A=B**3/2./PI
```

ÖRNEK **1.4** Makine hassasiyeti $\epsilon = 10^{-5}$ olan bir makineye $x = -1 \times 10^8$, $y = 1 \times 10^8$ ve $z = 1.0$ sayıları verilmiş olsun. Aşağıda verilen toplamların değerini bulunuz.

$$\begin{aligned} f_1 &= x + (y + z) \\ f_2 &= (x + y) + z \\ f_3 &= x + y + z \\ f_4 &= x + z + y \\ f_5 &= y + z + x \end{aligned}$$

Çözüm:
Verilenleri yerine yazalım:
$f_1 = x + (y + z) = -1 \times 10^8 + (1 \times 10^8 + 1.0)$
Yukarıdaki işlemde parantez içindeki sayıların işlem önceliği olduğundan önce bu sayılar toplanacaktır. Makine hassasiyeti 10^{-5} olduğuna göre toplamın sonucu ne olmalıdır? Bunu daha kolay görebilmek için f_1'i yeniden yazalım:
$f_1 = -1 \times 10^8 + 1 \times 10^8 * (1.0 + 1.0 \times 10^{-8})$
Burada parantez içindeki terim önceliğe sahiptir ve $10^{-8} < 10^{-5}$ olduğundan, sonuç 1.0 olacaktır. Böylece
$f_1 = -1 \times 10^8 + 1 \times 10^8 * 1.0 = -1 \times 10^8 + 1 \times 10^8 = 0$ olacaktır! Sonucun 1.0 olması gerekirken 0.0 çıkmasının nedeni makinedeki sınırlı hassasiyet ve işlemlerde uyulan önceliktir.

1.4 Örnek Programlar ve Öneriler

1.4.1 Verilerin Ekrandan Okutulması

Temel FORTRAN komutlarını hatırlamanız için EK A'da verilen kısa özeti kullanınız. Bu noktada temel FORTRAN komutlarının kullanıldığı basit algoritmalı programlar yazarak başlamak çok daha öğretici olacaktır. Daha sonra programların zorluk seviyesi ve karmaşıklığı arttırılabilir. Başlangıç olarak iki sayıyı bilgisayar ekranından okuyarak bunların toplamını bulup ekrana yazan bir program yazalım. Aşağıda verilen basit program bu işi yapmak için yeterlidir.

```
        PROGRAM GIRIS1
C************
C Ekrandan iki sayI okur ve bu sayIlarIn toplamInI ekrana yazar
C Son degistirme tarihi: 17 Ekim 2002
C***********
        REAL*8 A,B,TOPLAM
        WRITE(*,*)' A ve B nin degerini giriniz:'
        READ(*,*) A,B
        TOPLAM= A + B
        WRITE(*,*)' TOPLAM=',TOPLAM
        STOP
        END
```

Şimdi bu programdaki her satırın ne işe yaradığını sırası ile görelim. Programın ilk satırı programın ismini belirtir. Bu komut zorunlu değildir ve istenildiğinde programdan çıkartılabilir. Hatırlanacağı gibi FORTRAN 77 standardında geçerli komutlar komut satırının 7 ile 72. satırı arasına yazılmalıdır. Burada da program komutu yedinci satırdan başlayarak yazılmıştır.

Daha sonra gelen ve 'C' ile başlayan dört satır FORTRAN derleyicisi tarafından algılanmaz ve kullanıcı buralara istediği her şeyi yazabilir. Bu satırlar programcının kendisi ve programı kullanacak olan diğerleri için yazdığı açıklayıcı bilgiler içerir. Burada programın hangi amaçla yazıldığı, kim(ler) tarafından yazıldığı, eğer varsa problemin çözüm metodu ve literatürdeki kaynaklara referanslar, girdi/çıktı değişkenleri, programın değişikliğe uğradığı en son tarih ve yapılan değişiklikler gibi bilgiler verilir. Kullanıcının dikkat etmesi gereken önemli noktalar varsa burada belirtilir.

Programın 5. satırı A, B ve TOPLAM değişkenlerinin 'REAL' (gerçel, reel) tipi olduğunu belirtir. FORTRAN derleyicisi bu değişkenler için reel değişkenlere uygun olacak şekilde hafızada yer ayırır. 6. satır program kullanıcısına bilgi vermek üzere eklenmiştir. Burada kullanıcıya toplamı bulunacak A ve B değişkenlerinin değerlerinin ekrandan girilmesi gerektiği 'WRITE' komutu kullanılarak hatırlatılır. Bu satır olmadığı taktirde kullanıcı ne yapacağını bilemez ve bilgisyar da kullanıcıdan sürekli bir değer girmesini bekler. 7. satır ekrandan girilen iki sayıyı 'READ' komutu ile okuyarak bu değerleri kendileri için ayrılan hafıza yerlerine kaydeder.

8. satırda programın yapması gereken asıl iş, yani toplama işlemi yapılır ve sonuç TOPLAM değişkeni için ayrılan hafıza yerine kaydedilir. Bir sonraki satır sonucun ekrana yazılması içindir. Son iki satır ise programın durdurulması için gerekli komutlardır. 'STOP' komutu zorunlu değildir ve bir programda istenildiği kadar 'STOP' komutu olabilir. Fakat her programda mutlaka bir 'END' komutu olmalıdır ve bir ana programda (veya bir alt programda) sadece bir 'END' komutu olabilir.

Programda 'READ' ve 'WRITE' komutlarında parantez içinde yıldız ('*') kullanılmıştır. Bunlardan birincisi değişkenlerin ekrandan okunacağı (yazılacağı) anlamına gelir. ıkinci yıldız ise okuma ve yazma işlemlerinin formatsız yapılacağını gösterir. ılerde verilerin formatlı olarak nasıl yazdırılabileceğini göreceğiz.

ALIŞTIRMA **1.3** Bilgisayarınızda bulunan bir editör programını kullanarak yukarıda verilen programı bir dosyaya yazınız. Daha sonra bir FORTRAN derleyicisi kullanarak bu programı derleyip çalıştırınız. A ve B değişkenlerinin değerini ekrandan girerken bu değerleri boşluklarla veya virgülle ayırarak giriniz. Örneğin A ve B için 3.2 ve 6.7 değerleri girilecekse, ekrana

```
3.2 6.7
```

veya

```
3.2, 6.7
```

giriniz. A ve B için bu değerlerin girildiği durum için örnek çıktı aşağıda gösterilmiştir.

```
 A ve B nin degerini giriniz:
 3.2 6.7
 TOPLAM=       9.900000000000000
Stop - Program terminated.
```

1.4.2 Boyutlu Değişkenler ve Döngüler

Yukarıda verilen program biraz daha geliştirilebilir. Programın ikiden fazla sayıyı toplaması gerektiğini varsayalım. Bu durumda her sayı için A, B, C, $\cdots$ gibi farklı bir değişken kullanmak yerine boyutlu bir tek değişken kullanılabilir. Bu tip değişkenler vektör değişkenler olarak adlandırılır. Bu söylenen değişiklikleri yansıtacak şekilde GIRIS1 programı yeniden düzenlenerek aşağıdaki GIRIS2 programı elde edilmiştir.

```
        PROGRAM GIRIS2
C************
C Ekrandan N tane sayI okur ve bu sayIlarIn toplamInI ekrana yazar
C Program yazarI: Metin Ozdemir
C
C Son degistirme tarihi: 17 Ekim 2002
C  girdi(ler): A
C              A(NMAX): NMAX boyutlu vektor degiskeni
C
C  CIktI:      TOPLAM : sayIlarIn toplamI
C***********
        PARAMETER(NMAX=10)
        DIMENSION A(NMAX)
        REAL A,TOPLAM
        WRITE(*,*)' Kac sayI toplanacak?'
        READ(*,*) N
1       IF(N.GT.NMAX) THEN
        WRITE(*,*)
        WRITE(*,*)' Girdiginiz sayI', NMAX,' dan buyuk olamaz'
        WRITE(*,*)' Toplanacak sayI adedini yeniden giriniz'
        READ(*,*) N
        GOTO 1
        ENDIF
        WRITE(*,*)' Toplanacak sayIlarIn degerlerini giriniz:'
        READ(*,*) (A(I),I=1,N)
C SayIlarIn toplamini bulalIm
C TOPLAM i sIfIrlayalIm
        TOPLAM=0.0
        DO 10 I=1,N
        TOPLAM= TOPLAM + A(I)
10      CONTINUE
        WRITE(*,100) TOPLAM
100     FORMAT(1X,' TOPLAM=',1PE12.5)
        STOP
        END
```

Yeni programda yapılan değişikliklere bir göz atalım. Şimdi program hakkında verilen bilgi biraz daha genişletilmiştir. Programın yazarı, girdi ve çıktı değişkenleri, programın son değiştirilme tarihi vb. bilgiler açık olarak yazılmıştır.

Programda geçen önemli yeni bir isim 'PARAMETRE' komutudur. 'PARAMETRE' komutu programın çalışması esnasında hiç değişmeyen sabitlerin değerlerini atamak için kullanılır. Bunun bir diğer avantajı ise bu

sabitler için hafızada yer ayrılması özel olarak yapıldığından daha az hafıza kullanımı gerektirmesidir. Bu programda toplanabilecek en çok sayının değeri NMAX olarak atanmıştır. Eğer daha çok sayının toplanması gerekiyorsa sadece NMAX'ın değerini artırmak yeterlidir.

Önemli bir diğer fark ise vektörel (boyutlu) bir değişkenin olmasıdır. 'DIMENSION A(NMAX)' komutu A değişekeninin en çok NMAX bileşeninin olabileceğini yani en çok NMAX kadar değeri kaydedebileceğini gösterir. Bu nedenle A NMAX tane bileşeni olan bir vektör gibi düşünülebilir.

Programdaki 'IF(...) THEN ... ENDIF' komutu kontrol için kullanılmıştır. Toplamı yapılacak sayı NMAX'dan çok ise herhangi bir hataya yol açmamak için kullanıcı bu durumdan haberdar edilmektedir. Bu komutla ilgili daha fazla bilgi edinmek için EK A'ya bakınız.

'DO' döngüleri tekrarlanan hesaplamaları yapmak için kullanılır. Program GIRIS2'de N tane sayının toplamını bulmak için 'DO' döngüsü kullanılmıştır. Son olarak TOPLAM değişkeninin yeni programda formatlı olarak basıldığına dikkat edelim.

ALIŞTIRMA **1.4** *Yukarıda verilen GIRIS2 programını bir dosyaya yazınız. Bu programı fortran derleyicinizi kullanarak derleyip çalıştırınız. Programı çalıştırdığınızda aşağıdaki çıktıları almanız gerekir.*

1.4.3 'Subroutine' Alt Programı

GIRIS2 programını geliştirmeye devam edebiliriz. Programcılığın önemli basamaklarından biri yapılacak bir işi parçalara bölerek her parça için 'SUBROUTINE' veya 'FUNCTION' denen alt programları yazmaktır. Bu programın daha derli toplu olmasını, kolay takip edilmesini ve tekrar tekrar yapılan işlemler varsa bunların daha etkin biçimde hesaplanmasını sağlar. Bizim geliştirmeye çalıstığımız programda toplama işlemini bir alt program yazarak yapabiliriz. Ayrıca bazı değişkenler için başlangıç değerlerini 'DATA' komutunu kullanarak atayabiliriz. 'DATA' komutu kullanılarak değeri atanan değiskenlerin değeri gerektiğinde program içinde daha sonra değiştirilebilir. Anılan değişikliklerin yapıldığı program GIRIS3 adı ile aşağıda verilmiştir.

```
        PROGRAM GIRIS3
C************
C Ekrandan N tane sayI okur ve bu sayIlarIn toplamInI ekrana yazar
C Program yazarI: Metin Ozdemir
C
C Son degistirme tarihi: 17 Ekim 2002
C  girdi(ler): A
C              A(NMAX): NMAX boyutlu vektor degiskeni
C
C  CIktI:      S : sayIlarIn toplamI
C
C   subroutine'ler: TOPLAM(M,X,S)
C      girdi:
C    M: toplanacak sayIlarin sayIsI
C    X: (vektor) toplanacak sayIlarI tutar
C      output:
C    S: m sayInIn toplamI
```

```
C***********
        PARAMETER(NMAX=10)
        DIMENSION A(NMAX)
        REAL A,S
        DATA A/NMAX*2.0/N/10/
        WRITE(*,*)' Kac sayI toplanacak?'
        READ(*,*) N
1       IF(N.GT.NMAX) THEN
        WRITE(*,*)
        WRITE(*,*)
        WRITE(*,*)' Girdiginiz sayI', NMAX,' dan buyuk olamaz'
        WRITE(*,*)' Toplanacak sayI adedini yeniden giriniz'
        READ(*,*) N
        GOTO 1
        ENDIF
        WRITE(*,*)' Toplanacak sayIlarIn degerini giriniz'
        READ(*,*) (A(I),I=1,N)
C toplamI bul
        CALL TOPLAM(N,A,S)
        WRITE(*,100) S
100     FORMAT(1X,' TOPLAM DEGER=',1PE12.5)
        STOP
        END
C*****************************
        SUBROUTINE TOPLAM(M,X,S)
        DIMENSION X(M)
        REAL X,S
        S=0.0
        DO 10 I=1,M
        S= S + X(I)
10      CONTINUE
        RETURN
        END
```

Yukarıdaki programda geçen TOPLAM adlı 'SUBROUTINE' (alt program) M boyutlu X vektörü içinde saklanan M tane reel sayının toplamını bulur. Toplama işleminin sonucu S değişkeni ile ana programa geri döndürülür. M ve X alt programın girdileri, S ise çıktısıdır. Alt programda 'END' komutundan önce bir RETURN komutu olduğuna dikkat ediniz. Bu komut programın kontrolunun ana programda TOPLAM alt programının çağrıldığı 'CALL TOPLAM'dan hemen sonra gelen satıra devredileceğini gösterir.

Programda 'PROGRAM GIRIS3' ile başlayıp 'END' ile biten satırlar arasında kalan yapı ana program olarak bilinir. Ana programdan çeşitli işlemleri yapmak için çağrılan bütün diğer programlar alt programlardır.

ALIŞTIRMA **1.5** *Yukarıda verilen GIRIS3 programını yazıp derleyiniz ve çalıştırınız.*

1.4.4 'Function' Alt Programı

Başka bir alt program çeşidi ise 'FUNCTION' alt programıdır. Bir 'SUBROUTINE' alt programında birden fazla girdi ve birden fazla çıktı olabilir. Bir 'FUNCTION' alt programında ise birden fazla girdi olabilmesine karşın, alt programın kendi adı ile aynı olan sadece bir tek çıktısı olabilir. Bir 'SUBROUTINE' alt programı ile bir 'FUNCTION' alt programı arasındaki en önemli fark budur.

a ve b sabit sayılar olmak üzere, $F(x) = ax^2 - b$ fonksiyonu verilmiş olsun. Bu fonksiyonun belirli bir x_{min} değerinden başlayarak bir x_{max} değerine kadar eşit aralıklarla alacağı değerleri bulup tablo halinda yazmaya çalıştığımızı varsayalım. Bu işi yapabilecek bir program aşağıda verilmiştir.

```
        PROGRAM GIRIS4
C************
C x_min dan baslayarak x_max a kadar esit aralIklI NMAX
C noktada F(x)=a*x^2 fonksiyonunun degerini bulup ekrana
C yazar
C
C Yazan: M. Ozdemir
C Son degistirme tarihi: 10 KasIm,2002
C
C  GIRDI: a,b (sabit)
C
C  CIKTI: x_1, x_2 ... x_N noktalarIndaki F(x_i) degerleri
C***********

        COMMON/PAR/A,B

        DATA A,B,N/2.0,1.0,50/
        DATA XMIN,XMAX/-2.0,2.0/
        WRITE(*,*)' A, B ve toplam nokta sayIsI N i giriniz:'
        READ(*,*) A,B,N
        WRITE(*,*)' X_min ve X_max giriniz:'
        READ(*,*) XMIN,XMAX
C x icin esit aralIk degerini bul
        DX= (XMAX-XMIN)/FLOAT(N)
C F(x) fonksiyonun degerini bul
        WRITE(*,100)
        DO 10 I=1,N+1
        X= XMIN + (I-1)*DX
        Y= F(X)
        WRITE(*,200) X,Y
10      CONTINUE
100     FORMAT(3X,' X ',5X,' Y ')
200     FORMAT(1X, 2(2X,1PE12.5))
        STOP
        END
C*******************************
        FUNCTION F(X)
        COMMON/PAR/A,B
        F=A*X*X-B
```

```
        RETURN
        END
```

GIRIS4 programında bulunan x ve $F(x)$ değerleri iki kolon halinde ekrana yazılır.

ALIŞTIRMA **1.6** *Yukarıda verilen GIRIS4 programını yazınız. Fortran derleyicisi ile derleyip çalıştırınız. Bu programdaki COMMON ve FLOAT komutları ne işe yarar? Araştırınız.*

1.4.5 Çıktıların Bir Dosyaya Yazdırılması

Yukarıda verilen bütün programlarda programın çıktısı ekrana yazılmaktadır. Eğer program çıktısı ekrandan takip edilemeyecek kadar çoksa veya sonuçlar ileride kullanılmak üzere kalıcı olarak saklanmak isteniyorsa bu durumda sonuçların bir dosyaya yazılması gerekir. Bunu yapmak için programlar içerisinde 'OPEN' komutunu kullanarak istediğimiz isimde bir dosyayı açıp sonuçlarımızı bu dosyaya yazabiliriz. Açılan dosya ile işimiz bitince bu dosyayı 'CLOSE' komutu ile kapatıyoruz. Aşağıda verilen GIRIS5 adlı program bu söylenenleri yapmak üzere GIRIS4 adlı programın yeniden düzenlenmiş halidir.

```
        PROGRAM GIRIS5
C************
C x_min dan baslayarak x_max a kadar esit aralIklI NMAX
C noktada F(x)=a*x^2-b  ve G(x)= c*sin(2.*x)
C fonksiyonlarInIn degerini bulup kullanIcI tarafIndan
C ismi belirlenen bir dosyaya yazar
C
C Yazan: M. Ozdemir
C Son degistirme tarihi: 18 Ekim, 2002
C
C  GIRDI: a,b (sabit)
C
C  CIKTI: x_1, x_2 ... x_N noktalarIndaki F(x_i) ve G(x_i) degerleri
C************
        COMMON/PAR/A,B,C
        CHARACTER*12 CIKTI

        DATA A,B,C,N/2.0,1.0,3,50/
        DATA XMIN,XMAX/-2.0,2.0/
        DATA CIKTI/'OGIRIS5'/

        WRITE(*,*)' CIktInIn yazIlacagI dosya ismini giriniz:'
        READ(*,'(A)') CIKTI
        IF(CIKTI.EQ."/") CIKTI='OGIRIS5'
C 'CIKTI' adlI dosyayI acalIm
        OPEN(12,FILE=CIKTI,STATUS='UNKNOWN')
C
        WRITE(*,*)' A, B, C ve toplam nokta sayIsI N i giriniz:'
        READ(*,*) A,B,C,N
        WRITE(*,*)' X_min ve X_max giriniz:'
```

```
        READ(*,*) XMIN,XMAX
C
C x icin esit aralIk degerini bul
        DX= (XMAX-XMIN)/FLOAT(N)
C F(x) ve G(x) fonksiyonlarinin her x noktasIndaki degerini bul
        WRITE(*,100)
        WRITE(12,100)
        DO 10 I=1,N+1
        X= XMIN + (I-1)*DX
        Y= F(X)
        Z= G(X)
        WRITE(*,200) X,Y,Z
        WRITE(12,200) X,Y,Z
10      CONTINUE
        CLOSE(12)
100     FORMAT(7X,' X ',10X,' F(X) ',12X,' G(X) ')
200     FORMAT(1X, 3(2X,1PE12.5))
        STOP
        END
C******************************
        FUNCTION F(X)
        COMMON/PAR/A,B,C
        F=A*X*X-B
        RETURN
        END
C******************************
        FUNCTION G(X)
        COMMON/PAR/A,B,C
        G= C*SIN(2.*X)
        RETURN
        END
```

ALIŞTIRMA **1.7** *GIRIS5 Programındaki 'OPEN' ve 'CLOSE' komutlarının ne işe yaradığını araştırınız. 'OPEN' komutunda kullanılan '12' sayısı birim numarası olarak bilinir ve bu sayının 'DO' döngüsü içerisindeki 'WRITE' komutunun birinci argümanı olmasına dikkat ediniz. Bunun anlamı veriler daha önce açılan '12' birim numaralı dosyaya yazılacak demektir.*

ALIŞTIRMA **1.8** *Yukarıda verilen GIRIS5 programını bir dosyaya yazınız. Fortran derleyicinizi kullanarak bu programı derleyiniz ve çalıştırınız.*

1.4.6 Verilerin Dosyadan Okutulması

Bir programda kullanılacak veriler programda DATA komutu ile tanımlanabilir, daha önce gördüğümüz gibi ekrandan veya birazdan göreceğimiz gibi bir dosyadan da okutulabilir. Bu amaç için verilerin okunacağı dosyanın önceden OPEN komutu ile açılması gerekir. Bu açılan dosyada program için gereken veriler bulunmalıdır. Verilerin eksiksiz ve doğru okunabilmesi için, programda dosyada bulunan verilerin dizilişine uygun okuma komutları bulunmalıdır. Örneğin yukarıda kullandığımız GIRIS3 programında ekrandan oku-

nan toplanacak sayı sayısı ve bu sayıların değeri bir dosyadan da okutulabilir. GIRIS3 programının bu amaca uygun şekilde değiştirilmiş hali GIRIS6 adı altında aşağıda verilmiştir. Buradaki en önemli farklılık READ(*,*) komutunun birinci argümanının verilerin okunacağı dosyanın birim numrası ile aynı olmasıdır.

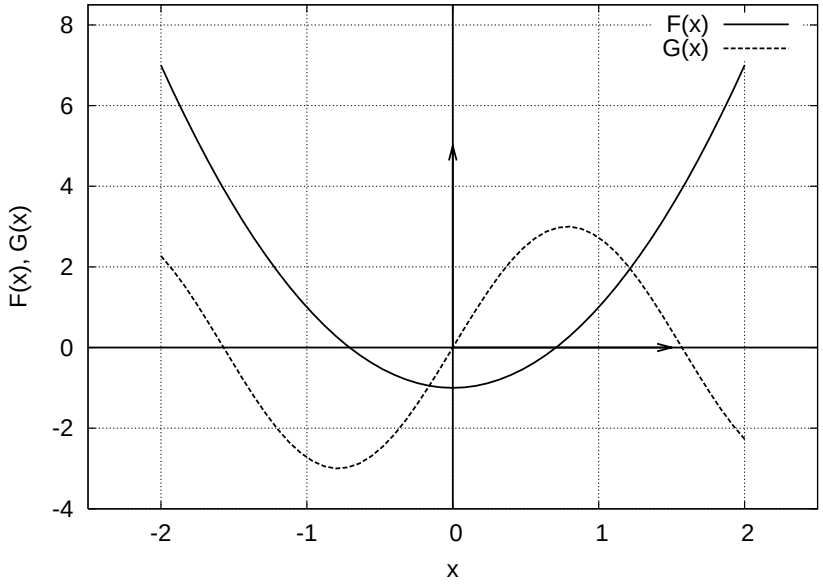
```
        PROGRAM GIRIS6
C************
C  KullanIcI tarafIndan ismi verilen bir dosyadan
C  N tane sayI okur ve bu sayIlarIn toplamInI ismi
C  kullanIcI tarafIndan verilen bir dosyaya yazar.
C
C  Veri dosyasInIn birinci satIrI toplanacak sayIya esit
C  bir tam sayIdIr. Takip eden satIrlarda toplanacak sayIlarIn degeri
C  vardIr ve her satIrda sadece bir sayI vardir.
C  Ornegin 1.2 3.3 -4.5 6.0 gibi 4 tane sayI toplanacak ise
C  veri dosyasI asagIdaki gibi olmalIdIr.
C 4
C 1.2
C 3.3
C -4.5
C 6.0
C
C  Program yazarI: Metin Ozdemir
C
C Son degistirme tarihi: 17 Ekim 2002
C  girdi(ler): A
C              A(NMAX): NMAX boyutlu vektor degiskeni
C
C  CIktI:      S : sayIlarIn toplamI
C
C   subroutine'ler: TOPLAM(M,X,S)
C      girdi:
C    M: toplanacak sayIlarin sayIsI
C    X: (vektor) toplanacak sayIlarI tutar
C      output:
C    S: m sayInIn toplamI
C************
        PARAMETER(NMAX=10)
        DIMENSION A(NMAX)
        REAL A,S
        CHARACTER*12 VERI, CIKTI

        WRITE(*,*)' Veri dosyasInIn ismini giriniz:'
        READ(*,'(A)') VERI
        WRITE(*,*)
        WRITE(*,*)' CIktIlarIn yazIlacagI dosyasInIn ismini giriniz:'
        READ(*,'(A)') CIKTI

C dosyalari ac
        OPEN(10,FILE=VERI,STATUS='UNKNOWN')
```

```
        OPEN(12,FILE=CIKTI,STATUS='UNKNOWN')

        READ(10,*) N
        IF(N.GT.NMAX) THEN
        WRITE(*,*)
        WRITE(*,*)
        WRITE(*,*)' Girdiginiz sayI', NMAX,' dan buyuk oldugundan'
        WRITE(*,*)' SayIlarIn sadece, NMAX, tanesi toplanacaktIr.'
        N=NMAX
        ENDIF

        DO 10 I=1,N
        READ(10,*) A(I)
10      CONTINUE

C toplamI bul
        CALL TOPLAM(N,A,S)
        WRITE(*,100) S
        WRITE(12,200) 'TOPLANAN SAYI ADEDI',N
        WRITE(12,210) 'TOPLAM DEGER',S

        CLOSE(10)
        CLOSE(12)

100     FORMAT(1X,' TOPLAM DEGER=',1H=,1PE12.5)
200     FORMAT(//,1X,A25,1H=,I5)
210     FORMAT(1X,A25,1H=,1PE12.5)
        STOP
        END
C******************************
        SUBROUTINE TOPLAM(M,X,S)
        DIMENSION X(M)
        REAL X,S
        SUM=0.0
        DO 10 I=1,M
        S= S + X(I)
10      CONTINUE
        RETURN
        END
```

ALIŞTIRMA **1.9** *Yukarıda verilen GIRIS6 programı için bir veri dosyası hazırlayınız. Bu programı bir dosyaya yazarak derleyiniz ve hazırladığınız veri dosyasını kullanarak çalıştırınız.*

1.4.7 Grafik Çizimi

Bilimsel çalışmalarda elde edilen verilerin uygun ve öz bir şekilde sunulması en az yapılan araştırma kadar önemlidir. Örneğin yukarıdaki GIRIS5 programının



Şekil 1.1: $F(x)$ ve $G(x)$ fonksiyonlarının grafikleri.

çalıştırılmasından sonra elde edilen dosyada bir çok rakamdan oluşan bir veri yığını vardır. Buradaki bilgi içeriğinin anlaşılır bir şekilde doğrudan başkalarına iletilmesinin kolay ve etkin yollarından biri elde edilen veriyi grafik ile temsil etmektir. GIRIS5 programının ürettiği verinin grafiği, yani $F(x)$ ve $G(x)$ fonksiyonlarının grafiği Şekil 1.1'de gösterilmiştir.[1]

ALIŞTIRMA **1.10** *Yukarıda verilen GIRIS5 programına benzeyen ve diyelimki adı GIRIS51 olan bir program yazınız. Bu program* $f(x) = 2(x-1)^2$ *ve onun birinci ve ikinci turevlerini* $[x_{min}, x_{max}]$ *aralığında eşit aralıklı N tane noktada hesaplayıp adı kullanıcı tarafından belirlenen bir dosyaya yazmalıdır. Çıktıların yazıldığı dosyada birinci kolona x değerleri yazılmalıdır. Benzer şekilde 2., 3. ve 4. kolonlara sırası ile* $f(x)$*,* $f(x)$ *fonksiyonunun birinci ve ikinci türevleri yazılmalıdır. Kullanıcı en küçük ve en büyük x değerlerini* (x_{min}, x_{max}) *ve bu aralıkta kaç tane değer basılacağını (N) girebilmelidir. Oluşturduğunuz bu çıktı dosyasını kullanarak verilen fonksiyonun ve onun türevlerinin grafiğini çiziniz. Önce her fonksiyonu tek tek çizerek görünüz. Daha sonra hepsini topluca aynı grafik üzerinde gösteriniz.*

1.5 Labaratuvar Saatleri

Her ana bölümün sonunda bir 'Labaratuvar Saati' kısmı olacaktır. Bu kısım o bölüme ait bir bilgisayar labaratuvarında veya kişisel çalışanlar için bir bilgisyar başında yapılması gereken işlerin toplu halde verilmiş hali olacaktır. Bu bölümde daha önce Örnek veya Alıştırma olarak çözülen veya çözülmesi istenen problemlerde tekrar ediliyor olabilir. Bütünlüğü bozmamak açısından bu

[1] Grafiği GNUPLOT kullanarak çizebilirsiniz. Ücretsiz elde edilebilecek bu programla ilgili daha fazla bilgi için EK B'ye bakınız

gibi tekrarların yapılmasında bir sakınca görülmemiştir. Labaratuvar saati sonunda öğrencilerden bir rapor istenmesi şart olmamakla beraber bir raporun verileceği varsayımı ile raporun biçimi ve yazımı ile ilgili bazı öneriler aşağıda verilmiştir. Bu öneriler sadece bir fikir vermesi için verilmiştir, nelerin hangi formatta isteneceğinin her ortama göre yeniden ayarlanması gerekecektir.

1- Raporunuzu bir diskete kayıtlı veya bir yazıcı çıktısı olarak teslim ediniz. Her iki durumda da kendinizle ilgili bilgileri veriniz ve raporun hangi Lab. Saatine ait olduğunu belirtiniz.

2- Raporların yazımı için size en uygun olan bir kelime işlemci seçiniz. Her sorunun çözümünü ayrı bir dosyaya kaydetmektense her Lab Saati için kullandığınız yazılımın sadece bir çıktı dosyasını veriniz. Dosya isimlerini geçerli Lab Saatine uygun seçiniz. örneğin bu bölüm için 'LabSaati-1.xxx' ismi uygun olacaktır.

3- Grafik çizimleri varsa bunları kullandığınız yazılım ortamına aktarınız.

4- FORTRAN dilinde yazdığınız programları (varsa) kullandığınız kelime işlemci dosyasına aktarmadan ayrı ASCII (text) dosyaları olarak teslim ediniz. Değerlendirme de kolaylık olması açısından program ve dosya adı olarak (varsa) sizden istenen isimleri kullanınız.

5- Çözümlerinizi Lab Saatindeki sıralamayı takip ederek ve aynı bölüm numaralarını kullanarak veriniz.

1.6 Labaratuvar Saati-I

1.) Aşağıda verilen fonksiyonların grafiklerini $[-0.5, 2]$ aralığında (a) ayrı ayrı ve (b) hepsini beraber uygun bir grafik programı, örneğin GNUPLOT, kullanarak çiziniz.

$f(x) = x\sin(x)$, $g(x) = xe^{-0.5x}$ ve $h(x) = x^2 - x$

2.) Yukarıda verilen fonksiyonların verilen aralıktaki değerlerini bir dosyaya yazacak olan LS1 isimli bir FORTRAN programı yazınız ve LS1.FOR adı altında kaydediniz. Program aşağıdaki özelliklere sahip olmalıdır: Kullanıcı verileri yazacağı dosyanın ismini, fonksiyon değerlerinin hesaplanacağı aralığın en küçük (x_{min}) ve en büyük (x_{max}) değerlerini ve bu aralığın kaç eşit parçaya (N) bölüneceğini girebilmelidir. Dosyanın birinci kolonu x değerlerini, 2., 3. ve 4. kolonları sırası ise f, g ve h fonksiyonlarının her x'e karşı gelen değerlerini içermelidir.

a) Yazdığınız programı kullanarak $x_{min} = -1$, $x_{max} = 2$, $N = 10$ için f, g ve h'ın değerlerini veren bir çıktı dosyası hazırlayınız. Bu dosyayı kullanarak f'in grafiğini çiziniz. Daha sonra veri dosyasını ve fonksiyonun kendisini beraber kullanarak grafik çiziniz ve ikisini karşılaştırınız. Aynı işlemi $g(x)$ ve $h(x)$ fonksiyonları için tekrar ediniz.

b) Yazdığınız programı kullanarak $x_{min} = -1$, $x_{max} = 2$, $N = 500$ için x, f, g ve h'ın değerlerini veren bir çıktı dosyası hazırlayınız. Bu dosyayı kullanarak f, g ve h'nin grafiklerini çiziniz.

1.7 ALIŞTIRMA SORULARI

1.) 21 sayısını ikili sayı sisteminde temsil ediniz.

2.) İkili sayı sisteminde (1001) ile temsil edilen sayının onluk sistemdeki değerini bulunuz.

3.)

Bölüm 2

LİNEER OLMAYAN DENKLEMLERİN ÇÖZÜMLERİ (KÖK BULMA)

Bu bölümde hemen hemen her bilim alanında çok sık ortaya çıkan $f(x) = 0$ şeklinde yazılan bir denklemi sağlayan değerlerin yani $f(x)$ fonksiyonunun köklerinin bulunma yöntemlerinden bazıları görülecektir. Ayrıca her metoda özgü ortaya çıkabilecek sorunlar, hata analizi, yakınsama oranı ve Newton-Raphson metodunun lineer olmayan denklem sistemlerine genelleştirilmesi incelenecektir.

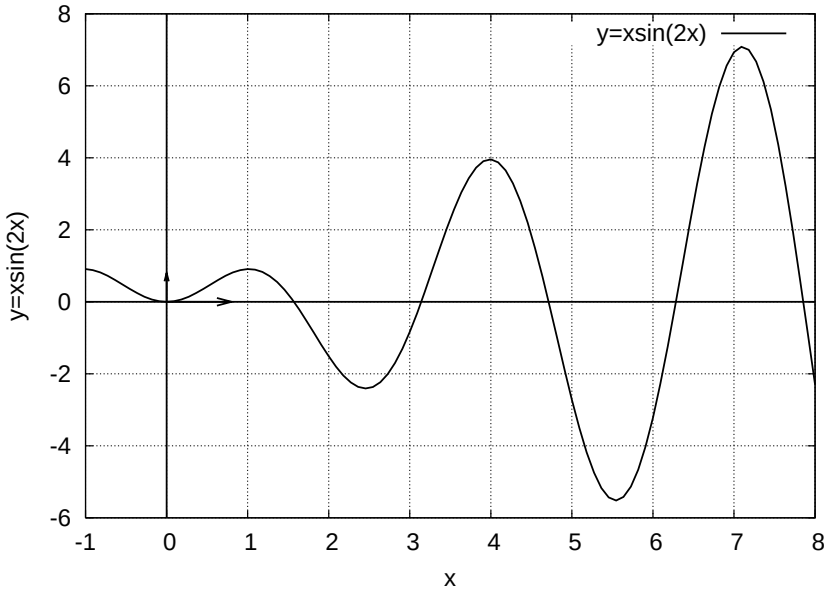
2.1 Giriş

Teorem 2.1 *Sürekli fonksiyonlar için ara-değer teoremi: $f(x)$ fonksiyonu $[a, b]$ aralığında sürekli bir fonksiyon olsun. Bu aralıktaki bazı x_1, x_2 ve α değerleri için $f(x_1) \leq \alpha \leq f(x_2)$ sağlanıyorsa, bu durumda bir $x_0 \in [a, b]$ için $\alpha = f(x_0)$ olacaktır.*

2.2 Grafik Yöntemi

$f(x) = 0$ şeklinde verilen ve bir değişkene bağlı bir foksiyonun gerçel kökleri varsa, bunlar fonksiyonun x eksenini kestiği noktalar olacaktır. Bu nedenle eğer kökü aranan fonksiyonun grafiği çizilirse x eksenini kestiği noktalar yaklaşık olarak bulunabilir ve bu noktalar aranan kökler olacaktır. Bu yöntem çok hassas olmamakla beraber fonksiyonun grafiğinin çizilebildiği durumlarda köklerin yaklaşık değerlerini bulmak veya köklerin hangi aralıkta bulunduğunu tespit etmek için kullanılabilecek pratik bir yöntemdir. Grafik çizerek kök bulmaya bir örnek Şekil 2.1'de gösterilmiştir.

Fonksiyon çizerek kök bulma başka bir şekilde de yapılabilir. Eğer verilen fonksiyon $f_1(x) = f_2(x)$ şeklinde ikiye ayrılabiliyorsa, bu durumda $f_1(x)$ ve $f_2(x)$ fonksiyonlarının grafikleri aynı dikey eksende x'e karşılık çizilir. Aranan kökler, iki fonksiyonun kesim noktaları olacaktır. Örneğin $x \geq 0$ için $f(x) = \sin(10x) - x = 0$ fonksiyonunun köklerini bulmaya çalışalım. Bu fonksiyonu $\sin(10x) = x$ şeklinde yazabiliriz. Bu durumda $f_1(x) = \sin(10x)$ ve $f_2(x) = x$ olacaktır. Şekil 2.2'de gösterildiği gibi $f_1(x)$ ve $f_2(x)$ fonksiyonlarının grafik-



Şekil 2.1: Bir fonksiyonun grafiğinin çizilerek köklerinin bulunması. Grafiğin x eksenini kestiği noktalar fonksiyonun aranan kökleridir.

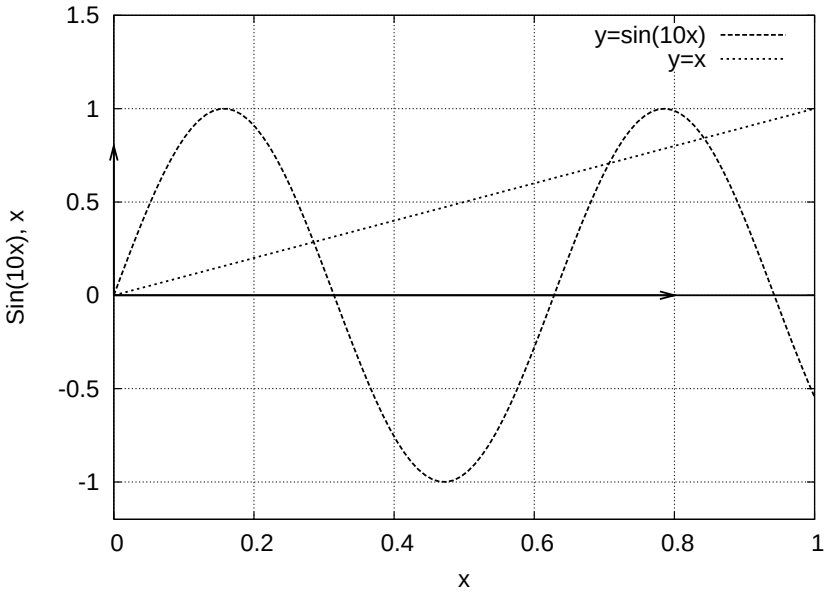
leri x'e göre aynı dikey eksene çizilirse, bu eğrilerin kesim noktası aranan kök değerleri olacaktır.

ALIŞTIRMA **2.1** *$f(x) = \sin(x) - x = 0$ fonksiyonunun köklerini yukarıda anlatılan iki yöntemle grafik çizerek yaklaşık olarak bulunuz.*

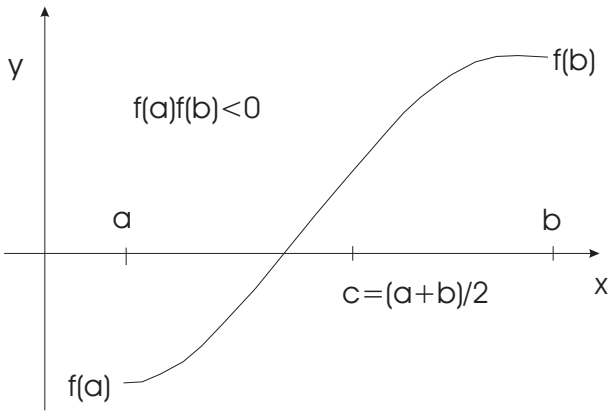
2.3 Yarılama Metodu

$f(x) = 0$ şeklinde verilen bir fonksiyonun eğer gerçel (reel) kökleri varsa, fonksiyonun kök civarında işaret değiştirmesi gerekir. Şekil 2.1'de her kök için açıkça görülen bu gerçek yarılama metodunun temelini oluşturur. Kökün (a, b) aralığında bulunduğu biliniyor ve $f(a)f(b) \leq 0$ sağlanıyorsa, yukarıda verilen ara değer teoremi gereği bu aralıkta en az bir kök olmalıdır. Eğer birden daha fazla kök varsa bunların sayısı 1, 3, 5 $\cdots$ gibi tek olmalıdır (neden?). Verilen aralığı daraltarak köke doğru yaklaşmak için yeni bir c noktası a ve b noktalarının ortalaması alınarak bulunur. Bu durumda $c = (a + b)/2$ olacaktır. Bu durum Şekil 2.3'te gösterilmiştir. $f(b)f(c)$'nin değerinin işaretine bakılarak bu noktanın kökün hangi tarafında olduğuna karar verilebilir. Eğer işaret negatif ise üst limit b değişmeyecek fakat yeni alt limit $a = c$ olacaktır, tersi geçerli ise alt limit a aynı kalırken yeni üst limit $b = c$ olacaktır. Verilen aralık bu şekilde sürekli ikiye bölündüğünden metodun ismi de buradan gelmektedir. İki aralıktan kök ihtiva etmeyen kısım atılarak diğer aralık tekrar ikiye bölünerek bu işleme köke yeteri kadar yaklaşılıncaya kadar devam edilir. Bu yöntemin algoritması aşağıda verilmiştir.

Algoritma 2.1 *Yarılama metodu:*



Şekil 2.2: $\sin(10x) - x = 0$ fonksiyonunun köklerinin $\sin(10x)$ ve x fonksiyonlarının grafiğinin aynı eksene göre çizilerek bu eğrilerin kesişiminden bulunması. Kesişim noktaları fonksiyonun aranan kökleridir ve verilen örnek için kök sayısı 4 tanedir.



Şekil 2.3: Bir fonksiyonun köklerinin yarılama yöntemi ile bulunması. Her iterasyon sonucunda başta seçilen aralık yarıya düşürülür.

1. $c = (a + b)/2$
2. $|b - c| \leq TOL$ *ise kok=c ve programdan çıkış*
3. $f(b)f(c) \leq 0$ *ise* $a = c$, *değilse* $b = c$
4. *(1)'e git*

Yukarıda verilen algoritmanın kullanılması ile bir kökün bulunduğu bilinen aralığın uzunluğu her adımda 2 kat azaltılır. Bu ise *ikili* sayı sistemine göre kökün her iterasyon (döngü) sonunda bir hane daha hassas olması anlamına gelir. Bu nedenle yeteri kadar iterasyon yapılması sonucunda bu yöntemle her zaman istenildiği kadar hassas bir kök bulunabilir. TOL, değeri kullanıcı tarafından belirlenen bir hata tolerans payıdır. İstenildiği taktirde bu pay makine hassasiyeti ϵ mertebesinde seçilebilir. Bu yöntemle çok hassas sonuçlar elde edilebilir fakat daha sonra karşılaştırmalı olarak görüleceği üzere bu yöntemin köke yakınsaması diğerlerine göre genelde daha yavaştır. Ayrıca yöntemin kullanılabilmesi için kökün bulunduğu aralık kullanıcı tarafından sağlanmalıdır.

2.4 Kiriş Metodu

Bu metot yarılama metoduna çok benzemekle beraber köke yaklaşmak için verilen aralığın yarılanması yerine iki uç noktadan çizilen doğrunun x eksenini kestiği nokta verilen aralığı ikiye bölmek için kullanılır. Bu yöntemde de kökün bulunduğu aralığın başlangıç (a) ve bitiş (b) noktaları baştan bilinmelidir. Şekil 2.4'te görüldüğü gibi $(a, f(a))$ noktasından $(b, f(b))$ noktasına çizilen doğrunun eğimi s iki şekilde yazılabilir. Doğrunun x eksenini kesim noktası c olarak alınırsa doğrunun eğimi

$$s = \frac{f(b) - f(a)}{(b - a)} = \frac{f(b) - 0}{(b - c)} \tag{2.1}$$

olacaktır. Bu denklem c için çözülürse

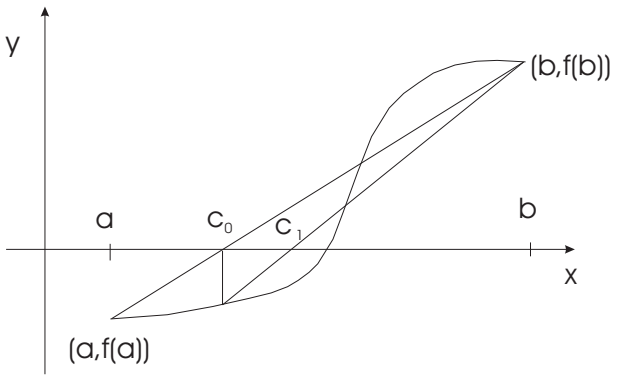
$$c = b - \frac{f(b)(b - a)}{f(b) - f(a)} \tag{2.2}$$

elde edilir. $f(x)$'in c noktasındaki işaretine göre alt limit veya üst limit c ile yer değiştirilir. Bu işleme aranan kök istenen hassasiyette bulunana kadar devam edilir. Bu yöntemin bir algoritması aşağıda verilmiştir.

Algoritma 2.2 *Yanlış çizgi metodu:*

1. $c_0 = b - a$
2. $c = b - \frac{f(b)(b-a)}{f(b)-f(a)}$
3. $f(b)f(c) \leq 0$ *ise* $a = c$, *değilse* $b = c$
4. $|c - c_0| \leq TOL$ *ise kok=c ve programdan çık,*
   değilse $c_0 = c$ *ve (2)'ye git*

Yanlış çizgi metodunun köke yakınsama oranı bazı durumlarda çok yavaş olabilir. Burada da TOL kullanıcı tarafından sağlanan bir hata tolerans payıdır.



Şekil 2.4: Bir fonksiyonun köklerinin yanlış çizgi yöntemi ile bulunması. Verilen aralığın uçlarını birleştiren doğrunun x eksenini kesim noktası aralığı daraltarak köke yaklaşmak için kullanılır.

2.5 Sekant Metodu

Sekant metodu yanlış çizgi metoduna çok benzemekle beraber bu yöntemde kökün bulunduğu aralığın bilinmesine gerek yoktur. Metodu başlatmak için iki tane başlangıç noktası gerekir. Bu iki noktaya karşılık gelen fonksiyon noktalarından geçen doğrunun x-eksenini kesim noktası bulunur. Fonksiyonun bu noktaya karşılık gelen noktası ve bir önceki fonksiyon noktalarından sonuncusu kullanılarak yeni bir doğru çizilir ve onun x-eksenini kesim noktası bulunur. Bu şekilde işleme istenilen hassasiyette bir kök bulunana kadar devam edilir. Metodu grafiksel olarak anlamak için şekil 2.5'e bakınız. x_0 ve x_1 noktalarına karşılık gelen fonksiyon değerlerinden geçen doğrunun x eksenini kestiği nokta x_2 olsun. Kiriş metodunda olduğu gibi bu doğrunun eğimi yazılıp x_2 için çözüm yapılırsa

$$x_2 = x_1 - \frac{f(x_1)(x_1 - x_0)}{f(x_1) - f(x_0)} \tag{2.3}$$

elde edilir. Bu işleme devam etmek için x-ekseninin kesim noktası olan peş peşe gelen iki değere x_n ve x_{n+1} dersek metodu $n \geq 1$ olmak üzere

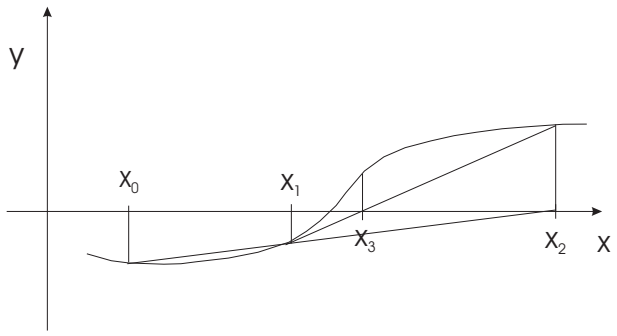
$$x_{n+1} = x_n - \frac{f(x_n)(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})}, \qquad n \geq 1 \tag{2.4}$$

şeklinde yazabiliriz.

Sekant metodunun algoritması ile kiriş metodunun algoritması çok benzerdir. Aşağıya bu metot için bir algoritma verilmiştir.

Algoritma 2.3 *Sekant metodu:*

1. $x_2 = x_1 - \frac{f(x_1)(x_1-x_0)}{f(x_1)-f(x_0)}$
2. $|x_2 - x_1| \leq TOL$ *ise kok*=x_2 *ve programdan çık,*
   değilse $x_0 = x_1$, $x_1 = x_2$ *ve (1)'e git*



Şekil 2.5: Bir fonksiyonun köklerinin sekant yöntemi ile bulunması. Başlangıçta verilen x_0 ve x_1 noktalarına karşılık gelen fonksiyon değerlerinden geçen doğrunun x eksenini kestiği nokta x_2 olarak seçilir. Daha sonra x_1 ve x_2 noktaları kullanılarak yeni bir x_3 noktası bulunur. Bu işlem köke belirlenen bir hata toleransı kadar yaklaşılıncaya kadar devam edilir.

2.6 Newton-Raphson Metodu

Newton-Raphson metodu şimdiye kadar gördüğümüz yöntemler içerisinde en hızlı yakınsayanıdır. Fakat burada hem kökü bulunacak fonksiyonun hemde onun türevininin bilinmesi gerekir. Kökü aranan fonksiyonun türevi bilinmiyor veya alınamıyorsa, türev gerektirmeyen daha önceki metotlardan biri kullanılabilir. N-R metodu doğrudan sekant metodundan çıkartılabileceği gibi Taylor serisinden faydalanarak da elde edilebilir. Taylor açılımını kullanarak elde etmek için kökü aranan fonksiyonu kök civarında Taylor serisine açalım.

$$f(x_0 + \Delta x) = f(x_0) + \Delta x f'(x_0) + \dots \tag{2.5}$$

Eğer $x_0 + \Delta x$ aranan köke yeterince yakın ise

$$f(x_0 + \Delta x) \approx 0 \approx f(x_0) + \Delta x f'(x_0) = 0 \tag{2.6}$$

olacaktır. Buradan $\Delta x = x - x_0$ çekilirse

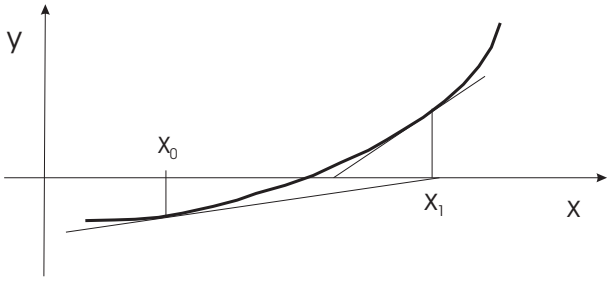
$$x = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{2.7}$$

elde edilir. Yukarıdaki denklemde x_0 kök için alınan tahmini bir değer x ise bu değerin biraz daha iyileştirilmiş halidir. Her adımda köke biraz daha yakın bir değer bulmak için bu denklem iterasyona uygun bir biçimde yazılabilir. Bu amaçla $x = x_{n+1}$ ve $x_0 = x_n$ yazılırsa denklemin son hali

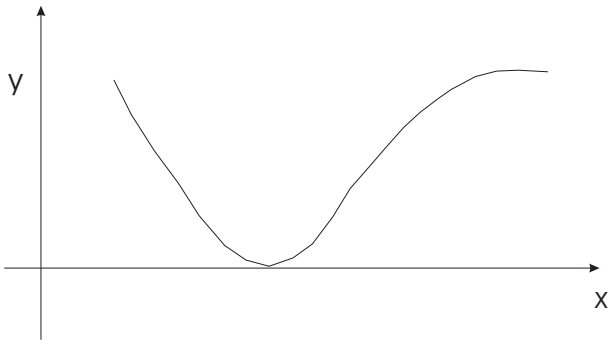
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{2.8}$$

olur. Bu yöntemin geometrik bir gösterimi Şekil 2.6'da verilmiştir. Her adımda fonksiyonun verilen noktadaki eğiminin x eksenini kesim noktası bulunarak köke yaklaşmaya çalışılır. Burada da görüldüğü gibi N-R metodunu başlatmak için sadece bir başlangıç noktasına ihtiyaç vardır. N-R metodunun bir algoritması aşağıda verilmiştir.

Algoritma 2.4 *Newton-Raphson metodu:*



Şekil 2.6: Bir fonksiyonun köklerinin Newton-Raphson yöntemi ile bulunması. Başlangıcta verilen x_0 noktasından geçen teğet doğrunun x eksenini kesim noktası yeni nokta olarak seçilir. Bu işlem köke belirli bir hata payı içerisinde yaklaşılıncaya kadar devam edilir.



Şekil 2.7: N-R metodu bazı özel durumlarda ıraksayabilir veya çalışmayabilir. Örneğin kökün bulunduğu noktada fonksiyon minimum oluyorsa bu noktadaki türev sıfır olur bu durum sorun yaratabilir.

1. $x = x_0 - \frac{f(x_0)}{f'(x_0)}$
2. $|x - x_0| \leq TOL$ *ise kok=x ve programdan çık,*
   değilse $x_0 = x_x$ *ve (1)'e git*

N-R metodu bazı özel durumlarda ıraksayabilir veya köke yakınsaması mümkün olmayabilir. Kökün bulunduğu noktada fonksiyon minimum veya maksimum oluyorsa bu durumda türev sıfıra gidecek ve çözüm ıraksayacaktır. Bu durum Şekil 2.7'de gösterilmiştir.

ALIŞTIRMA **2.2** *a) Yarılama, kiriş, sekant ve N-R metotlarını kullanarak kök bulmak için birer alt program yazınız.*

b) Bu yöntemleri kullanarak $f(x) = x^2 - 1 = 0$ *denkleminin kökünü/ köklerini bulan KOK1, KOK2, KOK3 ve KOK4.FOR programları Ek C'de verilmiştir. Bu programları derleyip çalıştırınız. Hangi metodun kaç iterasyonda istenilen köke yakınsadığını bulunuz.*

ÖRNEK **2.1** $f(x) = x^2 - 4$ *fonksiyonunun köklerini Newton-Raphson metodu ile çözmek üzere* $x_0 = 1$ *ile başlayarak* x_1 *ve* x_2 *'yi bulunuz.* x_2 *aranan köklerden birine ne kadar yakındır?*

Çözüm: $f'(x) = 2x$ *olduğundan denklem (2.8)'den faydalanarak*

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 1 - \frac{f(1)}{f'(1)} = 1 - \frac{-3}{2} = 1 + \frac{3}{2} = \frac{5}{2}$$

olarak bulunur. Benzer şekilde

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = \frac{5}{2} - \frac{f(\frac{5}{2})}{f'(\frac{5}{2})} = \frac{5}{2} - \frac{\frac{25}{4} - 4}{2\frac{5}{2}} = \frac{5}{2} - \frac{\frac{9}{4}}{5} = 2.05$$

olur. Aranan kökler ± 2 *olduğundan bulunan kökün aranan köke yakınlığı* $\delta x = |2 - 2.05| = 0.05$ *olacaktır.*

2.7 Newton-Raphson Metodu ile İki Denklemin Çözülmesi

Şimdi N-R metodunu iki tane lineer olmayan denklemin aynı anda çözümünü bulmak içinde kullanıla/bilir. Kökü aranan fonksiyonlar sistemi

$$F(x, y) = 0, G(x, y) = 0 \tag{2.9}$$

olsun. Bu sistemin kökü olarak aranan örneğin (x_0, y_0) sayı çifti yukarıdaki denklemlerin her ikisinide aynı anda sağlamalıdır, yani $F(x_0, y_0) = 0$ ve $G(x_0, y_0) = 0$ olmalıdır. Bir değişkenli bir fonksiyonun köklerini N-R metodu ile bulmak için yapılan işlemlere benzer şekilde $F(x, y)$ ve $G(x, y)$ fonksiyonları bir (x_0, y_0) noktası civarında Taylor serisine açılırsa

$$\begin{aligned} F(x, y) &= F(x_0, y_0) + \Delta x F_x(x_0, y_0) + \Delta y F_y(x_0, y_0) + \dots \\ G(x, y) &= G(x_0, y_0) + \Delta x G_x(x_0, y_0) + \Delta y G_y(x_0, y_0) + \dots \end{aligned} \tag{2.10}$$

elde edilir. Burada $\Delta x = x - x_0$ ve $\Delta y = y - y_0$ olup, fonksiyonların alt indisleri ise o değişkene göre kısmi türevleri göstermektedir. Örneğin $F_x(x_0, y_0)$ bu fonksiyonun x'e göre (x_0, y_0) noktasındaki kısmi türevidir. Yukarıdaki serilerin sonsuz tane terimi vardır ve bu haliyle açılım kesindir. Fakat (x_0, y_0) çiftinin aranan köke yeterince yakın olduğu varsayılıp bu denklemlerdeki lineer olmayan terimler ihmal edilebilir. Böylece yaklaşık olarak

$$\begin{aligned} F(x, y) &= F(x_0, y_0) + \Delta x F_x(x_0, y_0) + \Delta y F_y(x_0, y_0) \approx 0 \\ G(x, y) &= G(x_0, y_0) + \Delta x G_x(x_0, y_0) + \Delta y G_y(x_0, y_0) \approx 0 \end{aligned} \tag{2.11}$$

elde edilir. Bu denklem yeniden düzenlenirse

$$\begin{aligned} \Delta x F_x(x_0, y_0) + \Delta y F_y(x_0, y_0) &= -F(x_0, y_0) \\ \Delta x G_x(x_0, y_0) + \Delta y G_y(x_0, y_0) &= -G(x_0, y_0) \end{aligned} \tag{2.12}$$

olur. Yukarıdaki denklem seti $(\Delta x, \Delta y)$ ikilisinin bilinmediği lineer bir denklem setidir. Bilinen yöntemlerle bunlar için çözüm yapılabilir. Fakat iterasyon yaparak aranan köke yaklaşılacağı için (x_0, x, y_0, y) değişkenlerini bu amaca uygun olarak $x_n = x_0$, $x_{n+1} = x$ $y_n = y_0$ ve $y_{n+1} = y$ şeklinde yeniden tanımlamak

daha doğrudur. Bu yeni tanımlar yerine yazılır ve (x_{n+1}, y_{n+1}) için çözüm yapılırsa

$$\begin{aligned} x_{n+1} &= x_n - \frac{A_n}{B_n} \\ y_{n+1} &= y_n - \frac{C_n}{B_n} \end{aligned} \tag{2.13}$$

elde edilir. Burada A_n, B_n ve C_n aşağıdaki şekilde tanımlanmıştır.

$$\begin{aligned} A_n &= F(x_n, y_n)G_y(x_n, y_n) - G(x_n, y_n)F_y(x_n, y_n) \\ B_n &= F_x(x_n, y_n)G_y(x_n, y_n) - G_x(x_n, y_n)F_y(x_n, y_n) \\ C_n &= F_x(x_n, y_n)G(x_n, y_n) - G_x(x_n, y_n)F(x_n, y_n) \end{aligned} \tag{2.14}$$

ALIŞTIRMA **2.3** *a) Newton-Raphson metodunu kullanarak*

$$f(x, y) = \cos(x) - y, \qquad g(x, y) = x - \sin(y)$$

denklemlerinin köklerini bulan bir program KOK5.FOR adı altında Ek-C'de verilmiştir. Bu programı derleyip çalıştırınız. x ve y için bulduğunuz kökleri kaydediniz.

b) Aynı programda

$$f(x, y) = 3xy + x^2 + 2, \qquad g(x, y) = x^2y^2 + y - 2$$

denklemlerinin köklerini bulmak için gerekli değişiklikleri yapınız ve bu programı KOK51.FOR adı altında kaydediniz. KOK51'i derleyip çalıştırınız ve bulduğunuz kökleri kaydediniz.

2.8 Müller Metodu: Sanal Kökler

2.9 Sabit Nokta İterasyonları

2.10 Newton-Raphson Metodunun Genelleştirilmesi

2.11 Labaratuar Saati-II

Aşağıda, görmüş olduğumuz kök bulma yöntemlerini kullanarak verilen bir fonksiyonun kökünü (köklerini) bulan dört program ve bunlara ek olarak iki fonksiyonun ortak köklerini bulan bir program verilmiştir. Bu programların birer kopyasını Ek-C'de bulabilirsiniz.

1) Basit olması açısından kökü bulunacak fonksiyon olarak ilk dört programda $f(x) = x^2 - 1 = 0$ olarak seçilmiştir. $f(x)$ fonksiyonunun grafiğini çizdirerek köklerin nerede olduğunu görünüz. Grafikten kökleri ne kadar hassas okuyabiliyorsunuz?

2) KOK1.FOR programı yarılama metodunu kullanarak kök bulmak üzere yazılmıştır. Önce bu programdaki 'yarilama' alt programını inceleyiniz. Programın girdilerini, ne anlama geldiklerini ve programın çıktılarını iyice anlamaya çalışınız.

a) Bu programı, programda verilen (a, b) aralığını kullanarak çalıştırınız. Kaç iterasyondan sonra istenilen hassasiyet sağlandı? İterasyon sayısını diğer yöntemlerle karşılaştırmak için bir yere not ediniz.

b) (a, b) aralığı için sırası ile (0.5,1.5), (0.8,1.2) ve (0.95,1.05) değerlerini girerek iterasyon sayısında önemli bir değişiklik olup olmadığına bakınız.

3) (2)'yi kiriş yöntemini kullanan KOK2.FOR programını kullanarak tekrarlayınız. Burada seçilen aralığın aranan köke yakın olması, iterasyon sayısını (2)'deki yönteme göre nasıl etkiliyor?

4) KOK3.FOR programı sekant metodunu kullanarak kök bulmak üzere yazılmıştır. Bu yöntemde iki başlangıç noktasının verilmesi gerekir fakat aranan kökün bu noktalar arasında olması zorunluluğu yoktur. Bu programı değişik başlangıç noktaları kullanarak çalıştırınız. Kökü bulmak için gereken iterasyon sayısını diğer iki yöntem ile karşılaştırınız. İterasyon sayısı seçilen başlangıç noktalarına çok bağlı mıdır?

Ayrıca bu yöntemde varolan iki kökünde başlangıç noktalarının değiştirilmesi ile bulunabileceğine dikkat ediniz.

5) KOK4.FOR programı Newton-Raphson metodunu kullanarak kök bulmak üzere yazılmıştır. Verilen programda fonksiyon ve onun türevinin değerinin nerede hesaplandığına dikkat ediniz. Bu yöntemde sadece bir başlangıç noktasının verilmesi yeterlidir. Bu programı değişik başlangıç noktaları kullanarak çalıştırınız. Kökü bulmak için gereken iterasyon sayısını diğer üç yöntem ile karşılaştırınız. İterasyon sayısı seçilen başlangıç noktalarına çok bağlı mı?

Newton-Raphson metodunda da bütün reel köklerin sadece başlangıç noktalarının değiştirilmesi ile bulunabileceğine dikkat ediniz.

6) Yukarıda verilen yöntemlerin iyi ve kötü taraflarını belirterek bir karşılaştırmasını yapınız.

7) Yukarıda verilen programlarda gerekli değişiklikleri yaparak $f(x) = x^2 + 2x - 2$ fonksiyonunun köklerini bulunuz. Her bir yöntemde yakınsama için gereken iterasyon sayısını belirtiniz. (Çalışan bir programı hiç bir zaman değiştirerek bozmayınız. Yeni programlara örneğin KOK11.FOR, KOK21.FOR vb. isimleri verebilirsiniz).

8) $f(x) = x^3 - 5x^2 - 3$ fonksiyonunun kaç tane reel kökü olduğunu grafik çizerek bulunuz. Bu köklerden bir tanesini yukarıda verilen programları uygun şekilde değiştirerek bulunuz.

9) Newton-Raphson yöntemini kullanarak lineer olmayan iki denklemin köklerinin bulunması için yazılmış KOK5.FOR programını inceleyiniz. Kökü aranan $f(x, y)$ ve $g(x, y)$ fonksiyonları ile onların x ve y'ye göre kısmi türevlerinin NEWTONR2 alt programı içinde bulunmasına dikkat ediniz. Bu aslında iyi bir yöntem değildir fakat sadece iki tane denklem çözülecekse yeterlidir.

Bu programda $f(x, y) = \cos(x) - y$ ve $g(x, y) = x - \sin(y)$ fonksiyonlarının kökleri bulunmaktadır. Bir değişkenli fonksiyonlar xy düzleminde bir eğri oluştururlar. Fakat burada f ve g fonksiyonları birer yüzey tanımlarlar. Bu nedenle aranan kökler bu yüzeylerin kesiştiği noktaların kümesidir.

Grafik programınız üç boyutlu grafik çizme kapasitesine sahipse f ve g fonksiyonlarının grafiklerini çiziniz. Daha sonra KOK5.FOR programını kullanarak f ve g fonksiyonlarının köklerini bulunuz. Başlangıç değerleri x0 ve y0'ı değiştirerek değişik kök çiftleri olup olmadığına bakınız.

10) KOK5.FOR programını örneğin KOK51.FOR adı altında başka bir dosyaya kopyalayarak bu programı aşağıdaki fonksiyonların köklerini bulacak şekilde değiştiriniz. Bulduğunuz köklerin bir listesini yapınız.

$f(x, y) = 3xy + x^2 + 2$,

$g(x, y) = x^2y^2 + y - 2$

2.12 ALIŞTIRMA SORULARI

Bölüm 3

ADİ DİIFERANSİYEL DENKLEMLERİNİN SAYISAL ÇÖZÜMÜ

Temel bilimlerde, mühendislik veya iktisatta bazı problemlerin çözümünde sık sık bir diferansiyel denkleminin çözülmesi gerekmektedir. Eğer diferansiyel denkleminin analitik bir çözümü bulunamıyorsa bu durumda sayısal yöntemlerin kullanılması kaçınılmaz olacaktır. Bu bölümde başlangıç şartlı adi diferansiyel denklemlerinin sayısal olarak nasıl çözüleceğini, kullanılan bir yöntemi geliştirerek hataları azaltmak için neler yapılabileceğini, programlamada dikkat edilmesi gereken noktaları adım adım göreceğiz. Konuların anlatımı ve programlama basitten karmaşığa doğru beraber yürütülecektir.

3.1 Giriş

Bir diferansiyel denkleminde bir veya birden fazla bağımlı değişkenin sadece bir bağımsız değişkene göre türevleri varsa bu denkleme adi (bayağı) diferansiyel denklem diyoruz. Örneğin basit harmonik bir salınıcının uyduğu diferansiyel denklem, t zamanı ve x salınıcının merkeze göre yer değiştirmesini temsil etmek üzere,

$$\frac{d^2x}{dt^2} + w^2x = 0 \tag{3.1}$$

ile verilir. Burada t bağımsız değişken, x ise bağımlı değişkendir. Bir diferansiyel denkleminin mertebesi denklemde görülen en yüksek dereceli türeve eşittir. Bu nedenle yukarıdaki denklem ikinci mertebe bir diferansiyel denklemdir. Adi diferansiyel denklemler sağladıkları şartlara göre başlangıç veya sınır şartlı olabilir. Başlangıç şartlı diferansiyel denklemlerinde denklemin sağlaması gereken şartlar sadece bir noktada tanımlıdır. Sınır şartlı diferansiyel denklemlerinde ise denklemin sağlaması gereken şartlar birden fazla noktada tanımlanmıştır. Örneğin

$$\frac{d^2x}{dt^2} + w^2x = 0, \qquad x(t_0) = x_0, \qquad \left.\frac{dx}{dt}\right|_{t_0} = x_0' \tag{3.2}$$

diferansiyel denklemi başlangıç şartlıdır, çünkü denklemin sağlaması gereken şartlar sadece $t = t_0$ noktasında tanımlıdır.

$$\frac{d^2x}{dt^2} + w^2 x = 0, \qquad x(t_1) = x_1, \qquad \left.\frac{dx}{dt}\right|_{t_2} = x_2' \tag{3.3}$$

denklemi ise denklemin sağlaması gereken şartlar x_1 ve x_2 gibi iki *farklı* noktada tanımlandığından sınır şartlıdır. Bu bölümde sadece başlangıç şartlı diferansiyel denklemleri gözönüne alınacaktır.

Birden yüksel mertebeli herhangi bir diferansiyel denklem, mertebe sayısı kadar birinci mertebe diferansiyel denkleme dönüştürülebilir. Bu yöntemle n. mertebe herhangi bir diferansiyel denklem, n tane birinci mertebe diferansiyel denklemi şeklinde yazılabilir.

ÖRNEK **3.1** *Denklem (3.1)'de verilen diferansiyel denklemini iki tane birinci mertebe diferansiyel denklem şeklinde yazınız.*

Çözüm: *Bunu yapmak için*

$$\begin{aligned} x_1(t) &= x(t) \\ x_2(t) &= \frac{dx}{dt} \end{aligned} \tag{3.4}$$

tanımlarını yapalım. Bunları asıl denklemde yerine koyarsak

$$\frac{dx_2}{dt} + w^2 x_1 = 0 \tag{3.5}$$

elde edilir. Yukarıda tanımladığımız ikinci denklem ise

$$x_2(t) = \frac{dx_1}{dt} \tag{3.6}$$

olur ve aradığımız diğer denklemin kendisidir. Denklem (3.5) ve (3.6) birleştirilirse aranan denklem sistemi

$$\begin{aligned} \frac{dx_1}{dt} &= x_2 \\ \frac{dx_2}{dt} &= -w^2 x_1 \end{aligned} \tag{3.7}$$

elde edilmiş olur. Bu son iki denklem birinci mertebe ve kuplajlıdır.

ÖRNEK **3.2** *Üçüncü mertebe, lineer olmayan ve başlangıç şartları belli*

$$\begin{aligned} y\frac{d^3y}{dx^3} + (\frac{dy}{dx})^2 + y^2 &= g(x) \\ y(x_0) = y_0, \quad y'(x_0) = y_0', \quad y''(x_0) &= y_0'', \end{aligned} \tag{3.8}$$

diferansiyel denklemini üç tane birinci mertebe diferansiyel denklemler halinde yazınız. Ayrıca yeni denklemlerin sağladığı başlangıç koşullarını da belirtiniz.

Çözüm: *Bunu yapmak için daha önce yaptığımız gibi*

$$\begin{aligned} y_1(x) &= y(x) \\ y_2(x) &= \frac{dy}{dx} \\ y_3(x) &= \frac{d^2y}{dx^2} \end{aligned} \tag{3.9}$$

tanımlarını yapıp asıl diferansiyel denkleminde yerine yazalım. Dikkat edilirse yukarıdaki denklemlerden son ikisi aradığımız denklemlerden ikisini zaten tanımlamış olur. Üçüncüsüde verilen denklemden bulunacaktır. (3.8)'deki diferansiyel denklemini y_1, y_2 ve y_3 cinsinden yazarsak

$$y_1 \frac{dy_3}{dx} + y_2^2 + y_1^2 = g(x) \tag{3.10}$$

elde ederiz. Yeniden düzenleme yapılırsa aradığımız denklem sistemi elde edilir.

$$\begin{aligned} \frac{dy_1}{dx} &= y_2, \quad y_1(x_0) = y_0, \\ \frac{dy_2}{dx} &= y_3, \quad y_2(x_0) = y_0', \\ \frac{dy_3}{dx} &= -y_2^2/y_1 - y_1 + g(x)/y_1, \quad y_3(x_0) = y_0'', \end{aligned} \tag{3.11}$$

Daha karmaşık bir örnek ise ters kare kuralına uyan merkezi bir kuvvet (örneğin kütle çekim) altında (x, y) düzleminde iki boyutta bir yörüngede dönen bir parçacığın hareketini temsil eden diferansiyel denklem olabilir. G problemin sabitlerini temsil etmek ve $\mathbf{r} = x\mathbf{i} + y\mathbf{j}$ olmak üzere, Newton'un ikinci yasasına göre hareket denklemi

$$\frac{d^2\mathbf{r}}{dt^2} = -\frac{G\hat{\mathbf{r}}}{r^2} = -\frac{G\mathbf{r}}{r^3} \tag{3.12}$$

şeklinde yazılabilir. Bu denklem görüldüğü gibi ikinci mertebe bir diferansiyel denklem olup iki boyutta yazılmıştır. Bu diferansiyel denklem her boyut için ayrı ayrı yazılabilir. Böylece

$$\begin{aligned} \frac{d^2x}{dt^2} &= -\frac{Gx}{r^3} \\ \frac{d^2y}{dt^2} &= -\frac{Gy}{r^3} \end{aligned} \tag{3.13}$$

elde edilir. Bunlar kuplajlı ikinci mertebe diferansiyel denklemlerdir.

ÖRNEK **3.3** *(3.13)'de verilen denklemleri birer mertebe daha indirgenerek birinci mertebe dört denklem elde ediniz.*

Çözüm: *Bunu yapmak için hızın x ve y yönlerindeki bileşenlerini sırası ile $V_x = \frac{dx}{dt}$ ve $V_y = \frac{dy}{dt}$ olarak yazabiliriz. Bu tanımları kullanarak*

$$\begin{aligned} \frac{dx}{dt} &= V_x \\ \frac{dy}{dt} &= V_y \\ \frac{dV_x}{dt} &= -\frac{Gx}{r^3} \\ \frac{dV_y}{dt} &= -\frac{Gy}{r^3} \end{aligned} \tag{3.14}$$

şeklinde dört tane kuplajlı birinci mertebe diferansiyel denklem elde edilebilir. Bu denklemleri daha düzenli yazmak için $y_1 = x(t)$, $y_2 = y(t)$, $y_3 = \frac{dx}{dt}$ ve $y_4 = \frac{dy}{dt}$ tanımlarını kullanarak (3.14)'deki denklemler yeniden

$$\frac{dy_1}{dt} = y_3$$

$$\begin{aligned} \frac{dy_2}{dt} &= y_4 \\ \frac{dy_3}{dt} &= -\frac{Gy_1}{r^3}, \qquad r = \sqrt{y_1^2 + y_2^2} \\ \frac{dy_4}{dt} &= -\frac{Gy_2}{r^3} \end{aligned} \tag{3.15}$$

şeklinde yazılabilir.

Yüksek mertebeli herhangi bir diferansiyel denklem yukarıda örnekleriyle gördüğümüz gibi birinci mertebe diferansiyel denklemler dizisi halinde yazılabildiğinden, başlangıç şartlı adi diferansiyel denklemlerini sayısal olarak çözme yöntemleri sadece birinci mertebe diferansiyel denklemler için geliştirilmiştir. Bu nedenle birinci mertebe bir diferansiyel denklem ele alalım. En genel haliyle böyle bir denklem, başlangıç şartı ile beraber,

$$\frac{dy}{dx} = f(x, y), \qquad y(x_0) = y_0 \tag{3.16}$$

şeklinde yazılabilir. Burada x bağımsız, y ise bağımlı değişkendir. $f(x, y)$ aslında y'nin eğimini bütün (x, y) düzlemi üzerinde x ve y'nin bir fonksiyonu olarak tanımlar. Fakat bu bir çözüm değil, bir çözümler kümesi oluşturur. Bu çözümlerden bir tanesini seçmek için (x, y) düzlemi üzerinde bir noktanın, örneğin (x_0, y_0) noktasının verilmesi gerekir. Bu durumda aranan çözüm bu noktadan geçen çözümün kendisi olur ve tektir. Bu noktanın seçilmesi aslında gözönüne alınan diferansiyel denklemine bir başlangıç şartı verilmesi anlamına gelir. Diferansiyel denklemlerini sayısal olarak çözeceğimizden, bir diferansiyel denkleminde geçen türevleri sayısal olarak temsil edebileceğimiz yöntemler geliştirmeliyiz. Bu bir sonraki bölümün konusudur.

3.2 Sayısal Türev Alma

Denklem (3.16) veya benzeri denklemleri çözerken türev alınması gerekecektir. Bu nedenle sayısal olarak türev almak için bir yöntem bulmamız gerekir. Önce türevin tanımından başlayalım. Belirli bir aralıkta sürekli olan bir $f(x)$ fonksiyonunun herhangi bir x noktasındaki türevi hatırlanacağı gibi

$$\frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{y(x_0 + \Delta x) - y(x_0)}{\Delta x} \tag{3.17}$$

olarak tanımlanır. Bu tanımı kullanarak bilgisayarla bir foknsiyonun sayısal olarak türevini nasıl almamız gerekir? Δx kullandığımız makinenin hassasiyet limitine göre sonlu olmak üzere istediğimiz kadar küçük olabilir fakat limiti hiç bir zaman sıfıra götüremeyiz. Bu nedenle birinci ve gerektiğinde daha yüksek dereceli türevleri sayısal olarak alabilmek için uygun bir yöntem geliştirmemiz gerekir.

Böyle bir yöntemi fonksiyonların Taylor serisi açılımlarını kullanarak elde etmek mümkündür. $f(x)$ fonksiyonunun $x + \Delta x$ noktasındaki Taylor açılımı, $h = \Delta x$ olmak üzere,

$$f(x + h) = f(x) + hf'(x) + \frac{h^2}{2} f''(x) + \frac{h^3}{6} f'''(x) + \cdots \tag{3.18}$$

şeklindedir. Burada $f'(x)$, $f''(x)$, $\cdots$ $f(x)$ fonksiyonunun x noktasındaki birinci, ikinci, $\cdots$ türevleridir. Benzer şekilde $f(x)$'in $x-h$ noktasındaki Taylor açılımı

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{6}f'''(x) + \cdots \tag{3.19}$$

olur. Denklem (3.18)'i kullanarak $f(x)$'in türevi yaklaşık olarak

$$f'(x) = \frac{f(x+h) - f(x)}{h} + O(h^2) \tag{3.20}$$

yazılabilir. Burada h'nin iki ve daha büyük kuvvetlerinin bulunduğu terimler ihmal edilmiştir. Yani türev almada yaptığımız hata h^2 ile orantılıdır. Herhangi yaklaşık bir yöntemde ihmal edilen terimler h^{k+1} ile orantılı ise, bu yöntemin k. mertebeden hassas olduğu söylenir. Bu tanıma göre denklem (3.20)'de verilen birinci türevin yaklaşık değeri birinci mertebeden hassastır. Birinci türev için elde ettiğimiz bu yaklaşık değer *ileriye doğru sonlu fark* yaklaşımı olarak bilinir. Benzer şekilde denklem (3.19)'u kullanarak aynı türev *geriye doğru sonlu fark* olarak

$$f'(x) = \frac{f(x) - f(x-h)}{h} + O(h^2) \tag{3.21}$$

şeklinde yazılabilir. Bu yaklaşık değerde birinci mertebeden hassastır. Bunların dışında denklem (3.19)'u denklem (3.18)'den taraf tarafa çıkartırsak türev için *merkezi sonlu fark*

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^3) \tag{3.22}$$

denklemini elde ederiz. Bu yöntemde dikkat edilirse hesaplanan türev değeri ikinci mertebeden hassastır. Bu nedenle merkezi sonlu fark yöntemi ileri ve geri sonlu fark yöntemlerinden bir mertebe daha hassastır. Yukarıda gösterildiği gibi bir fonksiyonun bir noktadaki türevini, fonksiyonun o noktaya komşu noktalarındaki değerleri cinsinden yazma yöntemine genel olarak *sonlu farklar yaklaşık metodu* denir.

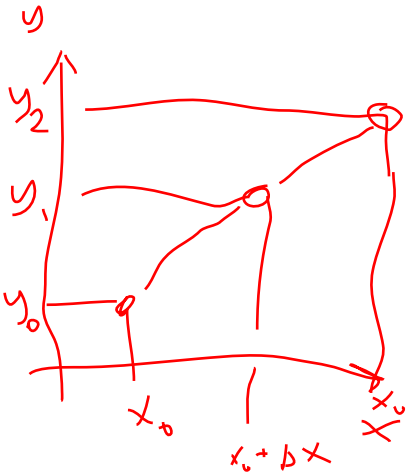
ALIŞTIRMA **3.1** *Denklem (3.18) ve (3.19)'un toplamından ikinci türev için yaklaşık*

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + O(h^4) \tag{3.23}$$

değerinin elde edilebileceğini gösteriniz.

3.3 Euler Metodu

(3.16)'daki diferansiyel denklemini sayısal olarak çözmek için kullanılabilecek en basit yöntem Euler metodudur. Bu yöntemde çözümün eğiminin çok yavaş değiştiği kabul edilir. Bu nedenle sonlu fakat küçük bir bağımsız değişken aralığı Δx için y'nin çok yavaş değiştiği veya sabit olduğu kabul edilebilir. Sürekli herhangi bir fonksiyonun bir x noktasındaki türevinin (3.17) ile verildiğini görmüştük. Fakat bu denklemin sayısal hesaplamalarda pratik olarak pek işe yaramadığını ve daha farklı yöntemlerin geliştirilmesi gerektiğini bir



önceki bölümde belirtmiştik. Bu nedenle (3.16)'daki diferansiyel denklemi, denklem (3.20)'de verilen birinci derece türevin ileriye doğru sonlu fark tanımından faydalanarak

$$\frac{y(x+\Delta x)-y(x)}{\Delta x}=f(x,y) \tag{3.24}$$

veya

$$y(x+\Delta x)=y(x)+\Delta x f(x,y) \tag{3.25}$$

olarak yazılabilir.

Yukarıdaki denklem, bir x_0 noktasındaki y_0 değerinin bilinmesi durumunda, daha sonraki bir $x_1 = x_0 + \Delta x$ noktasındaki $y_1 = y(x_0 + \Delta x)$ çözümünün bulunabileceğini gösterir. x_1 noktasındaki çözüm yaklaşık olarak

$$y_1 = y_0 + \Delta x f(x_0, y_0) \tag{3.26}$$

ile verilecektir. Böylece (x_1, y_1) noktası elde edilmiş olur. Bu nokta yeni bir başlangıç şartı olarak kullanılırsa bir sonraki $x_2 = x_0+2\Delta x$ noktasındaki çözüm yaklaşık olarak bulunabilir. x_2 noktasındaki yaklaşık çözüm

$$y_2 = y(x_0 + 2\Delta x) = y(x_1 + \Delta x) = y_1 + \Delta x f(x_1, y_1) \tag{3.27}$$

olacaktır. Bu şekilde adım adım devam ederek türevi $f(x, y)$ olan asıl $y(x)$ fonksiyonunu bulmuş oluyoruz. Yani integral alarak türevi alınmış ilkel fonksiyonu bulmaya çalışıyoruz. Δx sayısal çözümde kullanılan adım uzunluğu olarak bilinir.

$$\begin{aligned} h &= \Delta x \\ x_n &= x_0 + n\Delta x = x_0 + nh \\ y(x_n) &= y_n \\ y(x_{n+1}) &= y_{n+1} \end{aligned} \tag{3.28}$$

tanımlarını kullanarak ve denklem (3.25)'den hareketle Euler metodu

$$y_{n+1} = y_n + hf(x_n, y_n), \quad y(x_0) = y_0, \quad 0 \le n \le N \tag{3.29}$$

olarak yazılabilir. Bu bir tekrarlama bağıntısıdır. Başlangıçta (x_0, y_0) değerlerinin bilinmesi durumunda diğer bütün değerler bir önceki değerlerden faydalanarak adım adım bulunabilir.

Dikkat edilirse bu yöntemde aranan çözümün eğimi $f(x, y)$ her adımın başlangıcında hesaplanıp bu adım için kullanılmaktadır. Başlangıçtan itibaren bu süreç devam ettirilirse (x_0, y_0), (x_1, y_1), (x_2, y_2), $\cdots$, (x_N, y_N) dizisi elde edilir. Bu dizinin (x, y) düzleminde oluşturduğu noktalar kümesinin yaklaşık olarak aranan gerçek çözümü takip etmesi beklenir. x_0'dan başlayarak x_s'de son bulan belirli bir aralık için çözüm aranıyorsa, bu aralık K tane eşit aralığa bölünebilir. Δx'in küçük olmasını garantilemek için K'nın yeterince büyük olması gerekir. Aralık uzunluğu

$$h = \Delta x = \frac{(x_s - x_0)}{K} \tag{3.30}$$

ile verilir.

Tablo 3.1: Euler metodu ile $\frac{dy}{dx} = \frac{x}{2}$, $y(0) = 0.25$ diferansiyel denkleminin çözümü

x	y_a	y_s	$\Delta y = y_a - y_s$
0	0	$y_0 = 0$	0
$\frac{1}{4}$	$\frac{1}{4}((\frac{1}{4})^2 + 1) = \frac{17}{64}$	$y_1 = y_0 + hf(x_0, y_0) = \frac{1}{4} + \frac{1}{4}\frac{0}{2} = \frac{16}{64}$	$\frac{17}{64} - \frac{16}{64} = \frac{1}{64}$
$\frac{2}{4}$	$\frac{1}{4}((\frac{2}{4})^2 + 1) = \frac{20}{64}$	$y_2 = y_1 + hf(x_1, y_1) = \frac{16}{64} + \frac{1}{4}\frac{1}{2}\frac{1}{4} = \frac{18}{64}$	$\frac{20}{64} - \frac{18}{64} = \frac{2}{64}$
$\frac{3}{4}$	$\frac{1}{4}((\frac{3}{4})^2 + 1) = \frac{25}{64}$	$y_3 = y_2 + hf(x_2, y_2) = \frac{18}{64} + \frac{1}{4}\frac{1}{2}\frac{2}{4} = \frac{22}{64}$	$\frac{25}{64} - \frac{22}{64} = \frac{3}{64}$
1	$\frac{1}{4}((1)^2 + 1) = \frac{32}{64}$	$y_4 = y_3 + hf(x_3, y_3) = \frac{22}{64} + \frac{1}{4}\frac{1}{2}\frac{3}{4} = \frac{28}{64}$	$\frac{32}{64} - \frac{28}{64} = \frac{4}{64}$

3.3.1 Euler Metodu İçin Hata Tahmini

Euler metodunda yapılan hata ne kadardır? Çözümün her adımında y yaklaşık olarak

$$y(x + \Delta x) = y(x) + \Delta x f(x, y) = y(x) + \Delta x \frac{dy}{dx} \tag{3.31}$$

olarak yazılabilir. Fakat doğru ve kesin açılım Taylor serisi ile verilir. Denklem (3.31)'in Taylor serisi açılımı

$$y(x + \Delta x) = y(x) + \Delta x \frac{dy}{dx} + \frac{(\Delta x)^2}{2}\frac{d^2y}{dx^2} + \cdots \tag{3.32}$$

olacaktır. Bu nedenle denklem (3.31)'in kullanılması ile yapılan her adımdaki hesap $(\Delta x)^2$ mertebesinde hatalıdır. Eğer Δx küçük ise, $(\Delta x)^2$ çok küçük olacaktır. Fakat Δx'in küçük olması için K'nın çok büyük seçilmesi gerekir, yani atılan adım sayısının arttırılması gerekir. x_0 noktasından x_s noktasına kadar yapılan çözümde toplam hata $K(\Delta x)^2 = (x_s - x_0)\Delta x$ olacaktır. Yani Euler metodunda yapılan yerel hata $(\Delta x)^2$, toplam hata ise Δx mertebesindedir. Prensipte K'yı yeterince büyük seçerek, hatayı istenildiği kadar küçültmek mümkündür fakat bu pratik değildir.

ÖRNEK **3.4** *Örnek olarak aşağıda verilen diferansiyel denklemini Euler metodunu kullanarak* $[0, 1]$ *aralığında çözelim.*

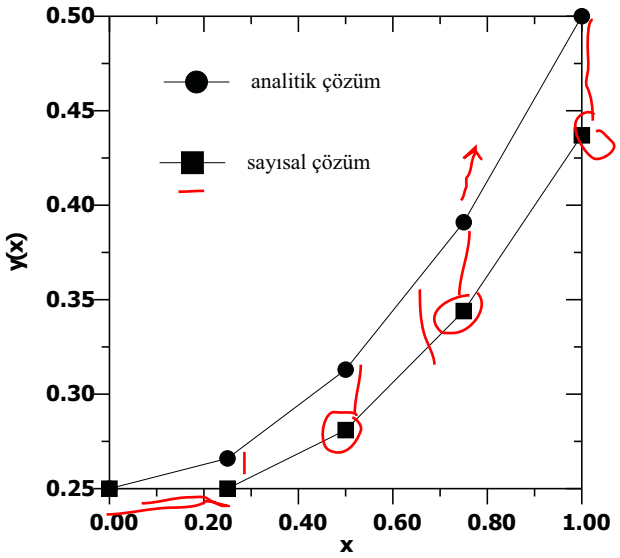
$$\begin{cases} \dfrac{dy}{dx} = \dfrac{x}{2} \\ y(x_0) = y_0 \end{cases} \tag{3.33}$$

Çözüm: *Bu diferansiyel denkleminin analitik çözümünün*

$$y(x) = y_0 + \frac{1}{4}(x^2 - x_0^2)$$

olduğunu göstermek ödev olarak size bırakılmıştır. Analitik çözüm sayısal çözümü test etmek için kullanılacaktır. Verilenlere göre $f(x, y) = \frac{1}{2}x$*'dir. Tablo 3.1'de* $x_0 = 0$, $y_0 = 0.25$ *ve* $h = \frac{1}{4} = 0.25$ *için verilen denklemin analitik çözümü* y_a*, Euler yöntemine göre elde edilen sayısal çözümü* y_s *ve yapılan hata* $\Delta y = y_a - y_s$ $x = 0, \frac{1}{4}, \frac{2}{4}, \frac{3}{4}, 1$ *noktalarında gösterilmiştir.*

Tablodan da görüldüğü gibi sayısal çözümde yapılan hata h ile doğrusal olarak artmaktadır. Şekil 1'de sayısal ve analitik çözümler aynı grafik üzerinde



Şekil 3.1: Örnek 3.5'de çözülen diferansiyel denkleminin sayısal ve analitik çözümleri.

gösterilmiştir. İki çözüm arasındaki fark yani sayısal çözümde yapılan hata şekilde görüldüğü gibi gittikçe artmaktadır. Bu tehlikeli bir duruma işaret eder. Bir süre sonra sayısal çözümde yapılan hata gerçek çözüm ile karşılaştırılabilir olacak ve daha sonra tamamen tanınmaz bir hal alarak gerçek çözüm ile bir ilgisi kalmayacaktır. Sayısal çözümde yapılan en ufak bir hata sınırsız şekilde artıyorsa bu durumda bir sonraki bölümde göreceğimiz gibi bir kararlılık sorunu var demektir.

3.3.2 Euler Metodunun Kararlılık Analizi

Sayısal çözümlerde göz önüne alınması gereken önemli kavramlardan bir tanesi kararlılıktır. Sayısal bir yöntemin kararlı olması çözümün kısa sürede çok büyüyerek veya sonsuza giderek tanınmaz hale gelmemesi olarak tanımlanabilir. Bazı yöntemler her şart altında kararlı, bazıları her zaman karasız olurken, bazı yöntemler ancak belirli parametre değerlerinde kararlı kalmaktadır. Kullanılan sayısal bir yöntemin detaylı bir kararlılık analizini yapmak her zaman mümkün olmayabilir. Euler metodunun kararlılık analizini yapmak için (3.29) denkleminde verilen sayısal çözümün, örneğin bilgisayarın sınırlı hassasiyetinden dolayı, gerçek sayısal çözümden ((3.16)'daki diferansiyel denkleminin analitik çözümünden değil) δy kadar uzaklaştığını varsayalım. Bunu (3.29)'a eklersek

$$y_{n+1} + \delta y_{n+1} = y_n + \delta y_n + h * f(x_n, y_n + \delta y_n) \tag{3.34}$$

elde edilir. Denklemdeki son terimi Taylor serisine açıp sadece δy'ye göre birinci mertebe terimleri alırsak

$$\begin{aligned} y_{n+1} + \delta y_{n+1} &= y_n + \delta y_n + h * [f(x_n, y_n) + \frac{\partial f(x,y)}{\partial y}|_n \delta y_n] \\ &= y_n + h * f(x_n, y_n) + \delta y_n + h * \frac{\partial f(x,y)}{\partial y}|_n \delta y_n \end{aligned} \tag{3.35}$$

$$= \quad y_{n+1} + \delta y_n + h * \frac{\partial f(x,y)}{\partial y}|_n \delta y_n$$

bulunur. Bu denklemden n.adımda yapılan bir hatanın bir sonraki adıma nasıl yansıyacağını veren

$$\delta y_{n+1} = [1 + h * \frac{\partial f(x,y)}{\partial y}|_n]\delta y_n \tag{3.36}$$

denklemini elde ederiz. Buradan açık olarak görüldüğü gibi herhangi bir şekilde sayısal çözüme karışan bir hatanın nasıl ilerleyeceği $k = (1+h*\frac{\partial f(x,y)}{\partial y})$ teriminin büyüklüğüne bağlıdır. $|k| < 1$ için hata gittikçe azalacak, $|k| > 1$ ise hata n arttıkça artarak çözümü kirletecek ve belkide tanınmaz hale getirecektir. Euler metodunun kararlı olması için bu nedenle

$$-1 \leq 1 + h * \frac{\partial f(x,y)}{\partial y} \leq +1 \tag{3.37}$$

şartının sağlanması gerekir. $h > 0$ olduğundan bu şart ancak $\frac{\partial f(x,y)}{\partial y} < 0$ ise mümkün olabilir. Adım uzunluğunun dikkatlice seçilmesi gereken bir parametre olduğu böylece açıklık kazanmış oldu.

Daha kesin bir tanım yapmak gerekirse sayısal bir çözüm eğer gerçek çözümden bir miktar uzaklaşıldığında büyüme eğilimi göstermiyorsa *kararlıdır.*

ÖRNEK **3.5**

$$\frac{dy}{dx} = y + x \tag{3.38}$$

diferansiyel denkleminin kararlılık analizini yapınız.

Çözüm: *Bu denklemde $f(x,y) = x + y$ olduğundan analiz için k değerini hesaplarsak*

$$k = 1 + h * \frac{\partial f(x,y)}{\partial y} = 1 + h$$

elde ederiz. $-1 \leq k \leq 1$ şartı hiç bir zaman sağlanamayacağından Euler metodu verilen denklem için her şart altında kararsızdır.

ALIŞTIRMA **3.2** *Aşağıda verilen diferansiyel denklemlerinin kararlılık analizini yapınız.*

a) Büyüme problemi:

$$\frac{dy}{dx} = \alpha y, \quad \alpha > 0$$

b)Bozunma problemi:

$$\frac{dy}{dx} = -\alpha y \quad \alpha > 0$$

ÖRNEK **3.6**

$$\frac{dy}{dx} = y + x, \quad y(x_0) = y_0 \tag{3.39}$$

diferansiyel denklemini Euler yöntemini kullanarak çözen bir FORTRAN programı yazınız. Bu denklemin analitik çözümünün

$$y(x) = (y_0 + x_0 + 1)e^{(x-x_0)} - (x+1)$$

olduğunu gösteriniz ve sayısal çözümü test etmek için kullanınız.

Çözüm: *verilen diferansiyel denklemini Euler metodu ile çözecek çok basit bir program EK C'de ADD1 adı ile verilmiştir.*

ADD1 programının bazı özellikleri not edilebilir. Program kendi kendini tekrar etmekte ve sadece K için sıfır veya daha küçük bir sayı girildiğinde durmaktadır. (x_0, y_0), (x_1, y_1), (x_2, y_2), $\cdots$, (x_s, y_s) değerlerinin hepsi X ve Y olarak adlandırılmakta, her adımda onların değeri güncelleştirilmektedir. Analitik çözüm YA hesaplanmış ve yapılan hata kaydedilmiştir.

ALIŞTIRMA **3.3** *a) ADD1 programına bir önceki paragraftan da yararlanarak açıklayıcı notlar ekleyiniz.*

b) ADD1 programını bir dosyaya editörünüzle kaydediniz, fortran derleyicinizi kullanarak derleyiniz ve çalıştırınız. Başlangıç şartları olarak X=0, Y=1 alınız.

i) XS=1 alarak programı K=5, 10, 20, 40 değerleri için çalıştırınız. Yapılan hata nasıl değişiyor? Yaklaşık olarak h ile orantılı mı? K iki kat arttırıldığında hata kaç kat azalıyor?

ii) (i)'yi XS=2 için tekrar ediniz.

iii) Örnek 3.5'de gösterildiği gibi bu denklem Euler metoduna göre her zaman kararsızdır. XS'yi büyük seçerek h ne kadar küçük olursa olsun çözümün sürekli büyüdüğünü ve kararsızlaştığını gösteriniz.

Daha genel uygulamalar için ADD1 programı biraz daha geliştirilerek EK C'de verilen ADD2 programı elde edilmiştir. Yeni programda diferansiyel denkleminin türevini hesaplamak için bir alt program, kullanılan yönteme göre integrali almak için bir alt program eklenmiştir. Yeni programı oluşturan kısımlar şunlardır:

- ADD2: ana program
- EULER: herhangi bir diferansiyel denklem için sayısal integrasyonu yapan alt program
- DEQ: dışarıdan sağlanan ve diferansiyel denkleminin türevini (denklemin sağ tarafını) hesaplayan alt program

Ana program girdi-çıktı işlerini ve bazı hesaplamaları yapmaktadır. Başlangıç şartları tekrar tekrar atamayı önlemek üzere X0 ve Y0 değişkenlerinde saklanmaktadır. EULER alt programı verilen denklemden tamamen bağımsızdır. Başka bir diferansiyel denklem çözüleceği zaman sadece DEQ alt programının değiştirilmesi yeterlidir. DEQ'nün EULER alt programının bir argümanı olmasına dikkat ediniz. Bu durum programın en başında 'EXTERNAL' komutu ile ilan edilmiştir. Daha iyi bir integrasyon yöntemi kullanılacaksa bu durumda EULER alt programının değiştirilmesi gerekir.

ALIŞTIRMA **3.4** *a) ADD2 programını bir dosyaya yazarak derleyiniz ve çalıştırınız. Bu programla* Örnek 3.6*'da verilen diferansiyel denklemini* Alıştırma 3.4*'de kullanılan aynı başlangıc ve K değerlerini kullanarak çözünüz. Elde ettiğiniz sonuçları ADD1 programının sonuçları ile karşılaştırınız. Çözümün hassasiyetinde bir değişme varmıdır?*

b) ADD2 programını

$$\frac{dy}{dx} = x^2 + y, \qquad y(0) = 1$$

diferansiyel denklemini çözecek şekilde başka bir ad altında, örneğin ADD21, değiştiriniz. Bu denklemin analitik çözümünü bulup yeni programda bu çözümü kullanmanız gerekir. Analitik çözümün $y(x) = -(x^2 + 2x + 2) + 3e^x$ *olması gerektiğini gösteriniz.*

ADD2 yapı olarak uygun bir program olmasına karşın biraz daha geliştirilip daha kullanışlı özelliklerle donatılabilir. Bazı değişiklikler şunlar olabilir: (a) girdi sayısının arttırılması ve bu sayede programın daha kullanışlı hale getirilmesi, (b) bazı değişken veya sabitlere baştan değer atayarak aynı değerin defalarca girilmesinin önlenmesi ve (c) çıktıların formatlı yazdırılması.

Bu amaçlarla yazılan yeni programda başlangıç değerleri daha önce olduğu gibi X0 ve Y0'a kaydedilecektir fakat son x değeri olan XS yerine üç tane yeni değişken tanımlanmıştır. Bunlar L, P ve IC'dir. L kaç tane periyot üzerinden hesaplama yapılacağını, P her periyotun uzunluğunu (büyüklüğünü) ve IC hesaplamanın nasıl yapılacağını tanımlamaktadır. Eğer IC=1 ise diferansiyel denkleminin integrasyonunun kaldığı yerden devam edeceğini, IC=0 ise baştan başlayacağını göstermektedir. Programın ilk çalıştırılışında IC=0 olmalıdır. K değeri yeni organizasyonda bir P periyodu içinde atılacak adım sayısını tanımlamaktadır. Yeni tanımlamalara göre toplam integrasyon uzunluğunun L×P olduğuna dikkat ediniz. Örnek olarak yukarıdaki diferansiyel denklemi [0,5] aralığında çözülecekse, başlangıçta L=5, P=1 ve IC=0 verilmesi yeterlidir. L ve P için olası başka bir seçenek ise L=10 ve P=0.5'tir. Integrasyona devam etmek için IC=1 verilmesi gerekir. Eğer L ve P için başka değerler girilmezse eski değerler kullanılarak integral almaya devam edilir. Örneğin başlangıçta IC=0, L=10, P=0.5 iken, bir sonraki değerler girilirken sadece IC=1 girilip diğerleri değiştirilmeden bırakılırsa integral kaldığı yerden 0.5 periyotlarla 10 defa alınarak X=10 değerine kadar alınır. Ortaya çıkan yeni program ADD3 adı altında EK C'de verilmiştir. Sadece ana programda değişiklikler yapılmış EULER ve DEQ alt programları değiştirilmeden bırakılmıştır.

ALIŞTIRMA **3.5** *a) EK C'de verilen ADD3 progamını derleyerek çalış tırınız.* Alıştırma 3.5(a) *'da çözülen diferansiyel denklemini aynı başlangıç şartlarını kullanarak yeniden çözünüz. Sonuçları* Alıştırma 3.5 *'in sonuçları ile karşılaştırınız. Ayrıca IC, L ve P parametrelerinin rolünü iyice ögrenmek için programı değişik L ve P değerleri için çalıştırınız.*

b) ADD3 programını başka bir ad altında, örneğin ADD31, değiştirerek

$$\frac{dy}{dx} = x^2 + y$$

diferansiyel denklemini $y(0) = 1$ *başlangıç şartını kullanarak çozünüz. Sadece DEQ alt programını değiştirmeniz yeterli olacaktır. Ayrıca bu denklemin analitik çözümünü bularak hata hesabında onu kullanmanız gerekir.*

Euler metoduna dayanarak şimdiye kadar yaptığımız integral alma işlemleri ne yazık ki yeterli değildir. Pratik olarak bakacak olursak, hatayı azaltmak için K'yı arttırmak daha çok hesaplama dolayısı ile daha çok zaman kaybı demektir. Teknik olarak ise K'nın arttırılması ile hesaplamalarda yapılan yuvarlama hatalarının artması demektir ve bu hataların birikerek yaklaşık çözümde baskın hale gelmesi durumunda, K'nın arttırılması yarardan çok zarar vermeye başlar. Her iki sebepten dolayı daha iyi sonuçlar veren yaklaşık metotları aramakda yarar vadır. Euler metodunu geliştirmenin bir yolu aşağıda göreceğimiz gibi Euler Richardson metodunu kullanmaktır.

3.4 Euler-Richardson Metodu

Bir parametreye bağlı olan herhangi yaklaşık bir yöntemi geliştirmenin bir yolu Richardson ekstrapolasyonudur. Euler metodunda integral almada yapılan hata Δx ile orantılı idi. Adım uzunluğunu yarıya düşürdüğümüzde yapılan hata $\Delta x/2$ ile orantılı olacaktır. Euler metodunda herhangi bir x'den $x+\Delta x$ noktasına kadar yaptığımız integral alma işlemini şimdi iki türlü yapalım. Birincisinde sadece bir adım atarak $x+\Delta x$ noktasında bir çözüm bulalım ve bu yaklaşık çözüme y_1 diyelim. Şimdi $x+\Delta x$ noktasına $\Delta x/2$'lik iki adım atarak ulaşalım ve bu işlem sonucunda elde ettiğimiz yaklaşık çözüme y_2 diyelim. Böylece aynı $x+\Delta x$ noktasına karşılık gelen y_1 ve y_2 gibi iki yaklaşık çözüm elde etmiş olduk.

y_1 ve y_2'nin uygun bir lineer (doğrusal) bileşimini alarak her ikisinden de daha iyi başka yaklaşık bir çözüm elde edebiliriz. Bunu yapmak için denklem (3.16)'ı göz önünde tutarak $h=\Delta x$ olmak üzere $y(x+h)$ ve $y(x+h/2)$'i x civarında, $y((x+h/2))+h/2)$'i $(x+h/2)$ civarında Taylor serisine açalım.

$$y(x+h) = y(x) + hy'(x) + \frac{h^2}{2}y''(x) + \frac{h^3}{6}y'''(x) + \cdots \tag{3.40}$$

$$y(x+h/2) = y(x) + \frac{h}{2}y'(x) + \frac{h^2}{8}y''(x) + \frac{h^3}{48}y'''(x) + \cdots \tag{3.41}$$

$$\begin{aligned} y((x+h/2)+h/2) &= y(x+h/2) + \frac{h}{2}y'(x+h/2) + \frac{h^2}{8}y''(x+h/2) \\ &\quad + \frac{h^3}{48}y'''(x+h/2) + \cdots \end{aligned} \tag{3.42}$$

Denklem (3.41)'i denklem (3.42)'de yerine yazarsak

$$\begin{aligned} y((x+h/2)+h/2) &= y(x) + \frac{h}{2}y'(x) + \frac{h^2}{8}y''(x) + \frac{h}{2}y'(x+h/2) + \\ &\quad \frac{h^2}{8}y''(x+h/2) + \frac{h^3}{48}y'''(x+h/2) + \cdots + \\ &= y(x) + \frac{h}{2}[y'(x) + y'(x+h/2)] \\ &\quad + \frac{h^2}{8}[y''(x) + y'(x+h/2)] \\ &\quad + \frac{h^3}{48}[y'''(x) + y'''(x+h/2)] + \cdots \end{aligned} \tag{3.43}$$

elde edilir. Denklem (3.40) ve denklem (3.43) y'nin $(x+h)$ noktasındaki iki ayrı yaklaşık değeridir, bunlardan birincisine y_1 diğerine y_2 demiştik. Göz önüne aldığımız denklemden görüleceği gibi $y'(x)=f(x,y)$ ve $y'(x+h/2)=f(x+h/2, y(x+h/2))$ olacaktır. y'nin x ve $(x+h/2)$'deki bu türevlerine sırası ile S_1 ve S_2 diyelim. $x_h = x+h/2$ ve $y_h = y(x+h/2)$ olmak üzere

$$\begin{aligned} S1 &= f(x,y) \\ S2 &= f(x_h, y_h) \end{aligned} \tag{3.44}$$

yazabiliriz. Böylece denklem (3.40) ve (3.43) sırası ile

$$y_1 = y(x) + hS_1 + \frac{h^2}{2}y''(x) + \frac{h^3}{6}y'''(x) + \cdots$$

$$y_2 = y(x) + \frac{h}{2}(S_1 + S_2) + + \frac{h^2}{8}[y''(x) + y''(x + h/2)]$$
$$+ \frac{h^3}{48}[y'''(x) + y'''(x + h/2)] + \cdots \quad (3.45)$$

olacaktır.

Şimdi y_1 ve y_2'nin lineer birleşiminden, her ikisinden de daha iyi yaklaşık bir çözüm bulmak için $x + h$ noktasındaki yaklaşık çözümü $y = 2y_2 - y_1$ şeklinde yazalım. Denklem (3.45)'deki ilk köşeli parantez içindeki ikinci terim de Taylor serisine açılıp adı geçen lineer birleşim alınırsa

$$y = y(x) + hS_2 + O(h^3) \quad (3.46)$$

elde edilir. Burada $O(h^3)$ terimi, h'nin üç ve daha yüksek kuvvetleri ile orantılı ihmal edilen artık terimleri temsil etmektedir. Burada da görüldüğü gibi Euler-Richardson metodu Euler metodundan bir mertebe daha hassastır. Dolayısı ile bu metotta toplam hatanın seçilen adım uzunluğunun yaklaşık karesi ile orantılı olmasını bekliyoruz.

Bu metot x_0'dan x_s'e kadar bütün bir bölge için değil, her Δx aralığı için ayrı ayrı uygulanmaktadır. Her aralıkta h adım uzunluğu kullanılarak yaklaşık bir çözüm ve iki tane $h/2$ adım uzunluğu kullanılarak başka yaklaşık bir çözüm bulunmaktadır. Buradaki gibi daha düşük mertebeli iki yaklaşık metottan daha yüksek mertebeli bir metot elde etme yöntemi genelde limite ekstrapolasyon yöntemi olarak bilinir. Euler-Richardson yönteminde herhangi bir x noktasından $x+h$ noktasına integral alırken yapılan işlemler aşağıda verilmiştir ve yöntemin algoritmasını oluşturur.

$$\begin{aligned} S1 &= f(x, y) \\ x_h &= x + \frac{h}{2} \\ y_h &= y + \frac{h}{2} * S1 \\ S2 &= f(x_h, y_h) \\ x &= x + h \\ y &= y + h * S2 \end{aligned} \quad (3.47)$$

Burada da görüldüğü gibi Euler metodunu bir mertebe daha hassas yapmak için ödediğimiz bir bedel vardır. Diferansiyel denkleminin sağ tarafı (çözümün eğimi) şimdi iki ayrı noktada iki defa hesaplanmaktadır.

Bu yeni durumu programlarımıza uygulamak için integral alma yöntemimizi yani EULER alt programını değiştirmemiz gerekir. Bunların dışında sadece bazı ufak değişiklikler yapmak yeterlidir. Yeni durumların uygulandığı program ADD4 adı altında EK C'de verilmiştir.

ALIŞTIRMA **3.6** *a)* Alıştırma 3.5*'da çözülen diferansiyel denklemini orada verilen aynı başlangıç ve parametre değerleri ile fakat program ADD4'ü kullanarak çözünüz. Çözümdeki hata gerçekten h^2 ile orantılı mıdır?*

b) ADD4 programını başka bir ad altında, örneğin ADD41, değiştirerek

$$\frac{dy}{dt} = x^2 + y$$

diferansiyel denklemini y(0)=1 başlangıç şartını kullanarak çözünüz. Bunun için sadece DEQ alt programını değiştirmeniz yeterlidir. Bulduğunuz sonuçları

Alıştırma 3.5.b *ile karşılaştırınız. Hangi yöntem daha hassastır?*

3.5 Runge-Kutta Metodu

Runge-Kutta metodunun gerisinde yatan temel fikir aslında yukarıda gördüğümüz Euler-Richardson metodunda uygulanan fikir ile aynıdır. Runge-Kutta metotlarında diferansiyel denkleminin türevi ($f(x,y)$) her aralık içinde belirli noktalarda hesaplanarak çok daha yüksek hassasiyet sağlamaya çalışılır. Şimdi çok basit ikinci mertebe hassasiyete sahip iki adımlı Runge-Kutta yöntemini inceleyelim.

3.5.1 İki-Adımlı Ruge-Kutta Metodu

$$\frac{dy}{dx} = f(x,y) \tag{3.48}$$

diferansiyel denklemi verilmiş olsun. Denklem (3.28)'de verilen tanımları kullanarak $x_{n+1} = x_n + h$ noktasındaki yaklaşık çözümü k_1 ve k_2 gibi iki fonksiyonun lineer bir birleşimi olarak

$$y_{n+1} = y_n + ak_1 + bk_2 \tag{3.49}$$

şeklinde yazalım. Burada k_1 ve k_2

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_n + \alpha h, y_n + \beta k_1) \end{aligned} \tag{3.50}$$

olarak aralığın başlangıçtaki ve aralık içinde en yüksek hassasiyeti sağlayacak bir noktadaki eğim olarak tanımlanmıştır. a, b, α ve β henüz bilinmeyen sabitlerdir. Bu sabitler $y(x+h)$'in Taylor açılımı ile (3.49)'daki terimler mümkün olan en yüksek mertebeden uyuşacak şekilde seçileceklerdir.

$y(x+h)$ Taylor serisine açılırsa

$$y(x_{n+1}) = y(x_n) + hy'(x_n) + \frac{h^2}{2}y''(x_n) + \frac{h^3}{6}y'''(x_n) + \dots \tag{3.51}$$

elde edilir. Dikkat edilirse y'nin türevleri $f(x,y)$ ve onun x ve y'e göre kısmi türevleri cinsinden yazılabilir. Bunu yukarıdaki denkleme uygularsak, f_x ve f_y sırası ile $f(x,y)$'in x ve y'ye göre kısmi türevleri olmak üzere

$$\begin{aligned} y' &= f(x,y) \\ y'' &= \frac{df(x,y)}{dx} = f_x + f_y y' = f_x + f_y f \\ y''' &= \frac{d^2 f(x,y)}{dx^2} = f_{xx} + 2f_{xy}f + f_{yy}f^2 + f_x f_y + f_y^2 f \end{aligned} \tag{3.52}$$

elde edilir. Yukarıdaki denklemi çıkarma işlemi ödev olarak okuyucuya bırakılmıştır. Böylece Taylor açılımı $f(x,y)$ ve onun kısmi türevleri cinsinden

$$\begin{aligned} y(x_{n+1}) = y(x_n) &+ hf(x_n, y_n) + \frac{h^2}{2}[f_x + ff_y]_n \\ &+ \frac{h^3}{6}[f_{xx} + 2ff_{xy} + f_{yy}f^2 + f_x f_y + f_y^2 f]_n \\ &+ O(h^4) \end{aligned} \tag{3.53}$$

olur. Burada alt indis n bütün değerlerin (x_n, y_n) noktasında hesaplanması gerektiğini gösterir.

İki değişkenli fonksiyonlar için Taylor açılımını kullanarak k_2

$$\frac{k_2}{h} = f(x_n + \alpha h, y_n + \beta k_1) = f(x_n, y_n) + \alpha h f_x + \beta k_1 f_y + \frac{\alpha^2 h^2}{2} f_{xx} + \alpha h \beta k_1 f_{xy} + \frac{\beta^2 k_1^2}{2} f_{yy} + O(h^3) \tag{3.54}$$

şeklinde yazılabilir. Burada da bütün türevlerin (x_n, y_n) noktasında hesaplanması gerekir.

$k_1 = hf(x_n, y_n)$ olduğu gerçeğini kullanarak k_2 için elde edilen değer denklem (3.49)'da yerine yazılır ve h'nin aynı kuvvetli katsayıları yeniden düzenlenirse

$$y_{n+1} = y_n + (a+b)hf + bh^2[\alpha f_x + \beta f f_y] + bh^3[\frac{\alpha^2}{2} f_{xx} + \alpha\beta f f_{xy} + \frac{\beta^2}{2} f^2 f_{yy}] + O(h^4) \tag{3.55}$$

elde edilir. Bunun denklem (3.53) ile karşılaştırılarak h ve h^2'nin katsayılarının eşitlenmesi ile

$$\begin{aligned} a + b &= 1 \\ b\alpha = b\beta &= \frac{1}{2} \end{aligned} \tag{3.56}$$

elde edilir. Burada üç tane denklem fakat dört tane bilinmeyen vardır. Dolayısı ile parametrelerin seçiminde bir derece serbestlik olanağımız olacaktır. Bir çok olası çözüm arasından aşağıda verilenleri göz önüne alalım.

a) $a = 0$ seçilirse, bu durumda $b = 1$ ve $\alpha = \beta = \frac{1}{2}$ olur. Parametrelerin bu değerleri için Runge-Kutta denklemleri daha önce elde ettiğimiz Euler-Richardson denklemleri (3.47) ile tamamen aynı olur.

b) Başka olası bir çözüm kümesi $a = b = \frac{1}{2}$ ve $\alpha = \beta = 1$ olacaktır.

İki adımlı Runge-Kutta yöntemini uygulamak için ADD4 programı değiştirilmiş ve yeni program ADD5 olarak adlandırılmıştır. Bu programın bir kopyesi EK C'de verilmiştir. Her adım için burada iki defa eğim hesaplanması gerekmektedir. ERICS alt programı RK2 ile değiştirilmiştir.

ALIŞTIRMA **3.7** *a) ADD5 programını kullanarak* Alıştırma 3.5*'de verilen diferansiyel denklemini çözünüz ve sonuçları ADD4 programının sonuçları ile karşılaştırınız.*

b) ADD5 programında a, b, α and β parametreleri için tutarlı farklı değerler kullanarak (a)'yı yeniden çözünüz. Çözümde ve hassasiyet mertebesinde önemli bir değişiklik meydana geliyormu?

3.5.2 Dört Adımlı Runge-Kutta Metodu

İki adımlı Runge-Kutta (RK) metodunun çıkarılışında izlenen yol kullanılarak dört adımlı daha hassas yöntemde elde edilebilir. Dört adımlı Runge-Kutta metodunda (3.16)'da verilen diferansiyel denkleminin çözümünün $x_{n+1} = x_n + h$ noktasındaki yaklaşık değerini

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{3.57}$$

şeklinde yazıyoruz. Burada k_i'ler

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_n + \frac{h}{2}, y_n + \frac{1}{2}k_1) \\ k_3 &= hf(x_n + \frac{h}{2}, y_n + \frac{1}{2}k_2) \\ k_4 &= hf(x_n + h, y_n + k_3) \end{aligned} \tag{3.58}$$

olarak tanımlanmıştır. Yukarıda verilen yöntemde her adımda yapılan yerel hata $O(h^5)$ mertebesindedir. Diğer yöntemlere göre hata çok daha küçük fakat bunun için ödenen bedel her adımda $f(x, y)$ fonksiyonunun dört defa hesaplanmasıdır. Dikkat edilirse her mertebeden Runge-Kutta metodu çıkartmak her zaman olasıdır fakat hesaplamalar çok karmaşık ve uzun olabilir.

ALIŞTIRMA **3.8** *ADD5 programını başka bir ad altında, örneğin ADD51, değiştirerek yukarıda verilen dört adımlı Runge-Kutta algoritmasını programlayınız.* Alıştırma 3.5*'de verilen problemi yazdığınız bu programı kullanarak çözünüz. Sonuçlarınızı daha önce elde ettiğiniz bütün sonuçlar ile karşılaştırınız. En hassas olan yöntem hangisidir?*

Şimdiye kadar kullandığımız bütün programlar ve alt programlar sadece bir diferansiyel denkleminin çözümü için yazılmıştır. N tane denklemden oluşan bir diferansiyel denklem sistemini çözmek için değişkenlerin vektörler şeklinde yazılması gerekir. Yani gerekli değişkenler boyutlandırılmalıdır. Boyutlandırmaların nasıl yapılabileceğini göstermek üzere Örnek (3.3)'de elde edilen (3.15)'deki denklem sisteminin çözülmesi için ADD4 programı ADD6 adı altında yeniden düzenlenmiş ve EK C'de verilmiştir.

Yeni programdaki bazı değişiklikleri not edelim. Toplam dört tane birinci mertebe diferansiyel denklem olduğu için 'PARAMETER' komutunda N=4 olarak verilmiş ve bu değer daha sonra boyutlu değişkenlerin boyutlarının tanımlanmasında kullanılmıştır. Başlangıç değerleri Y0 adlı vektörde tutulmuş, parçacığın x ve y yönündeki koordinatlarına sırası ile Y(1) ve Y(2), x ve y yönündeki hızlarına ise sırası ile Y(3) ve Y(4) adı verilmiştir. S1, S2 ve S3 vektörleri fonksiyon değerlerini hesaplayıp tutmak ve ara işlemleri yapmak için kullanılmaktadır. Programda tanımlanan CT değişkeni DEQ alt programının kaç defa çağrıldığını yani kaç defa fonksiyon değer hesaplaması yapıldığını kaydetmektedir. Programı yakından inceleyerek diğer değişiklileri sizin bulmanız ve anlamaya çalışmanız yararlı olacaktır.

ALIŞTIRMA **3.9** *ADD6 programını program içinde verilen* $x_0 = 1$, $y_0 = 0$, $V_x = 0$ *ve* $V_y = 0.5$ *başlangıç şartlarını kullanarak derleyip çalıştırınız. Bu başlangıç şartları Y0'lar cinsinden Y0(1)=1, Y0(2)=0, Y0(3)=0 ve Y0(4)=0.5 olacaktır.*

ADD6 programında verilen toplam integrasyon süresi L×P = 2.714080941 bu parçacığın elips olan yörüngesini tamamlaması için gereken süreye, yörünge periyoduna eşittir. Bu nedenle bir periyot sonunda parçacık başladığı yere dönmeli ve buraya ulaştığı andaki hızları ile başlangıçtaki hızları eşit olmalıdır. Ayrıca parçacığın toplam enerjiside korunmalıdır (neden?). Bazı problemlerdeki bunlara benzer zamanla değişmeyen hareket sabitleri, kullanılan sayısal yöntemi test etmek için kullanılabilecek çok önemli parametrelerdir. Göz önüne

Tablo 3.2: ADD6 programının örnek bir çıktısı

G= 1.0,	CT= 2000.				
T	X	Y	Vx	Vy	DX
0.000E+00	1.000E+00	0.000E+00	0.000E+00	5.000E-01	2.714E-03
2.714E-01	9.629E-01	1.340E-01	-2.757E-01	4.809E-01	2.714E-03
5.428E-01	8.478E-01	2.568E-01	-5.798E-01	4.141E-01	2.714E-03
8.142E-01	6.415E-01	3.507E-01	-9.595E-01	2.548E-01	2.714E-03
1.086E+00	3.102E-01	3.698E-01	-1.532E+00	-2.147E-01	2.714E-03
1.357E+00	-1.430E-01	1.279E-04	-4.287E-03	-3.498E+00	2.714E-03
1.628E+00	3.090E-01	-3.709E-01	1.531E+00	-2.199E-01	2.714E-03
1.900E+00	6.404E-01	-3.530E-01	9.604E-01	2.514E-01	2.714E-03
2.171E+00	8.471E-01	-2.598E-01	5.815E-01	4.119E-01	2.714E-03
2.443E+00	9.627E-01	-1.375E-01	2.777E-01	4.797E-01	2.714E-03
2.714E+00	1.000E+00	-3.679E-03	2.312E-03	4.998E-01	2.714E-03

aldığımız örnekte yörüngenin kapalı elips olması nedeni ile parçacık başladığı yere dönmelidir. Ayrıca başlangıç noktasına döndğü andaki hızları ilk hızlarına eşit olmalıdır. Bu programın bir çıktısı Tablo 3.2'de verilmiştir. Bu tabloyu inceleyerek yukarıda anlatılan özelliklerin sağlanıp sağlanmadığını kontrol ediniz.

ALIŞTIRMA **3.10** Alıştırma 3.14*'de yazdığınız dört adımlı Runge-Kutta metodunu uygulayan ADD51 adlı programı ADD52 adı altında denklem (3.15)'de verilen kütle çekim problemini çözmek için yeniden düzenleyiniz. ADD6 ve ADD52 programlarının sonuçlarını karşılaştırınız. Hangisi daha hassas ve hangisi DEQ alt programını en az sayıda çağırmaktadır?*

3.6 Adım Uzunluğu Kontrol Edilebilen Metotlar

Kütle çekim probleminin Tablo 3.2'de gösterilen çözümü dikkatlice incelendiğinde eliptik yörüngede dolaşan parçacığın T=1.357 olduğu zamanda (periyodun yarısında) merkeze en yakın ve hızının en yüksek olduğu görülecektir. Sayısal çözümde bu bölgede yapılan hatayı azaltmak ve yörüngeyi tam takip edebilmek için Δx adım uzunluğunun bu bölgede gerçekten küçük olması gerekir. Adım uzunluğunun yeniden ayarlanması için program durdurulup yeni bir adım uzunluğu verilerek yeniden başlatılabilir. Fakat yeni adımın ne zaman ve hangi büyüklükte seçilmesi gerektiği açık değildir. Bunun tam olarak bilinmesi için çözümün biliniyor olması gerekir ki buda mümkün değildir. Bu nedenle adım uzunluğunu çözümün akışı içerisinde dinamik olarak belirlemenin uygun bir yolunu bulmamız gerekir. Her aralıkta yapılan hesaplama hatası herhangi bir yolla tahmin edilebilirse, buna dayanarak adım uzunluğuda dinamik olarak seçilebilir. Bunu yapmanın bir yolu Euler-Richardson metodunda gördüğümüz gibi herhangi bir x_n noktasından $x_{n+1} = x_n + h$ noktasına integrasyonu iki ayrı şekilde gerçekleştirerek x_{n+1} noktasındaki $y(x)$ için iki ayrı yaklaşık değer bulmaktır. Bu yaklaşık değerlerden biri Δx adım uzunluğu, diğeri ise iki tane $\Delta x/2$ adım uzunluğu kullanılarak bulunabilir. Bu iki yaklaşık değerin karşılaştırılması adım başına yapılan yerel hatanın mertebesi hakkında bir fikir verir. Yerel olarak yapılan bu hata kullanıcı tarafından belirlenen bir hata tolerans parametresi

ile karşılaştırılarak adım uzunluğu dinamik olarak çözümün akışı içerisinde belirlenebilir. Böylece sabit bir adım uzunluğu kullanmak yerine çözümün özelliklerine uygun ve onunla uyumlu olarak değişen adım uzunluğu kullanılmış olur. Bunun sağladığı bir çok fayda vardır.

Burada anlatılan yöntem daha önce kullandığımız Euler-Richardson yöntemine kolayca uygulanabilir. Yukarıda anlatıldığı gibi her aralık için iki yaklaşık çözüm bulunup bunların farkından yapılan hata tahmini HT elde edilir. Bu hata kullanıcı tarafından sağlanan hata toleransı HTOL ile karşılaştırılır ve adım uzunluğu hakkında bir karar verilir. Üç olası durum mevcuttur:

i) Eğer HT, HTOL'den büyük ise gerekli tolerans sağlanana kadar adım uzunluğu yarılanır.

ii) Eğer HT yeterince küçük ise bu aralıktaki integrasyon etkin olan adım uzunluğu kullanılarak bitirilir,

iii) Eğer HT, HTOL'den çok çok küçük ise bu durumda adım uzunluğu arttırılır.

Böylece alınan integralin değişimine göre dinamik olarak uyarlanan bir adım uzunluğu olan bir program algoritması elde edilmiş oldu. Ayrıca birden fazla diferansiyel denkleminin çözüldüğü durumlarda her değişken için ağırlıklı bir hata payı AH tanımlayarak her değişken farklı bir hata toleransı ile hesaplanabilir.

EK C'de verilen ADD7 programı ADD6 programının adım kontrolü uygulanmış halidir. Yani adım uzunluğu kontrollü bir Euler-Richardson yöntemidir. Ayrıca her değişken için, örneğin j. değişken için ağırlıklı bir hata payı AH(J) tanımlanıp, o değişken için hata toleransı HTOL'ün AH(J) ile çarpılmasından elde edilmektedir. Böylece her değişken için ayrı bir göreli hata toleransı tanımlamak mümkündür.

ALIŞTIRMA **3.11** *ADD7 programını çalıştırarak kütle çekim problemini yeniden çözünüz. Bu programın bir çıktısı Tablo 3.3'de gösterilmiştir.*

Tablo 3.3'de verilen program çıktısı incelendiğinde DEQ alt programının sadece 466 kez çağrıldığı görülecektir. Adım kontrolünün olmadığı ADD6 programı için bu değer 2000 idi. Demek ki fonksiyon hesaplama sayısı yaklaşık dört kez daha az. Buna karşılık elde edilen sonuçlar daha hassas. Adım uzunluğunun sürekli değişmesine, Δx'in $t = 1.357$ civarında en küçük değerini almış olmasına özellikle dikkat ediniz.

3.6.1 Adım Uzunluğu Kontrollü RK-Verner Metodu

Birinci mertebe diferansiyel denklemleri sayısal olarak çözmek için adım kontrollü ileri düzeyde bir çok yöntem geliştirilmiştir. Runge-Kutta yöntemlerine de adım kontrolü sağlayan yeni metotlar eklenmiştir. Bunlardan çok uygun olan bir yöntem Runge-Kutta-Verner metodudur. Diğer benzer metotlar RK-Merson ve RK-Fehlberg metotlarıdır. RK-Verner algoritmasının detayları burada verilmeyecektir. Sadece bu algoritmaya dayalı olarak yazılan ve adım kontrolü sağlayan bir alt program verilmiştir. Euler-Richardson metodunda aralık başına iki türev değeri hesaplanmakta, birinci mertebe bir hata tahmini yapılmakta ve ikinci mertebeden hatalı bir çözüm üretilmektedir. Benzer şekilde RK-Verner metodunda aralık başına sekiz türev değeri hesaplanmakta, beşinci mertebe bir hata tahmini yapılmakta ve altıncı mertebeden hatalı bir çözüm üretilmektedir. Bu yöntemde $10^{-6} - 10^{-11}$ mertebesinde hata toleransları kullanmak hiç sorun

Tablo 3.3: ADD7 programının örnek bir çıktısı

HTOL=	1.000E-03	MS=	1.000E+03		
AH=	1.000E+00	1.000E+00	1.000E+00	1.000E+00	
G=	1.0	CT=	466.		
T	X	Y	Vx	Vy	DX
0.000E+00	1.000E+00	0.000E+00	0.000E+00	5.000E-01	1.000E-02
2.714E-01	9.629E-01	1.340E-01	-2.755E-01	4.809E-01	5.647E-02
5.428E-01	8.480E-01	2.570E-01	-5.793E-01	4.142E-01	4.817E-02
8.142E-01	6.420E-01	3.511E-01	-9.583E-01	2.553E-01	3.328E-02
1.086E+00	3.112E-01	3.705E-01	-1.529E+00	-2.122E-01	1.100E-02
1.357E+00	-1.434E-01	4.918E-03	-6.990E-02	-3.491E+00	1.249E-03
1.628E+00	3.063E-01	-3.705E-01	1.536E+00	-2.242E-01	1.051E-02
1.900E+00	6.387E-01	-3.531E-01	9.634E-01	2.510E-01	2.299E-02
2.171E+00	8.460E-01	-2.598E-01	5.836E-01	4.121E-01	4.307E-02
2.443E+00	9.620E-01	-1.373E-01	2.795E-01	4.801E-01	5.307E-02
2.714E+00	1.000E+00	-3.321E-03	4.024E-03	5.001E-01	4.526E-02

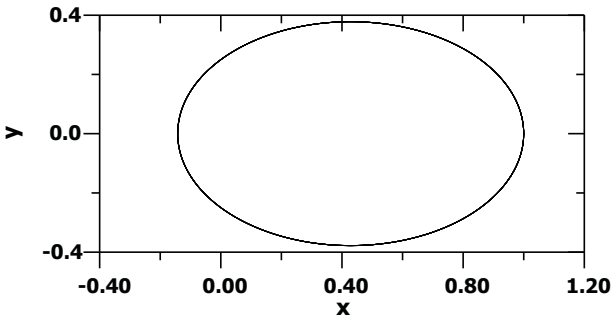
Tablo 3.4: ADD8 programının örnek bir çıktısı

HTOL=	1.000E-06	MS=	1.000E+03		
AH=	1.000E+00	1.000E+00	1.000E+00	1.000E+00	
G= 1.0	CT=	424.			
T	X	Y	Vx	Vy	DX
0.000E+00	1.000E+00	0.000E+00	0.000E+00	5.000E-01	1.000E-02
2.714E-01	9.629E-01	1.340E-01	-2.757E-01	4.809E-01	1.420E-01
5.428E-01	8.478E-01	2.568E-01	-5.798E-01	4.141E-01	2.414E-01
8.142E-01	6.415E-01	3.507E-01	-9.595E-01	2.548E-01	2.172E-01
1.086E+00	3.101E-01	3.698E-01	-1.532E+00	-2.147E-01	1.271E-01
1.357E+00	-1.429E-01	-3.578E-07	-1.991E-06	-3.500E+00	1.040E-02
1.628E+00	3.101E-01	-3.698E-01	1.532E+00	-2.147E-01	6.875E-02
1.900E+00	6.415E-01	-3.508E-01	9.595E-01	2.548E-01	1.162E-01
2.171E+00	8.478E-01	-2.568E-01	5.798E-01	4.141E-01	1.964E-01
2.443E+00	9.629E-01	-1.340E-01	2.757E-01	4.809E-01	1.767E-01
2.714E+00	1.000E+00	-1.360E-05	9.157E-06	5.000E-01	2.297E-01

yaratmayacaktır. Adım uzunluğu kontrollü Euler-Richardson metodunun uygulandığı ADD7 programı RKV metodu uygulanacak şekilde yeniden düzenlenmiş ve ortaya çıkan yeni program ADD8 adı ile EK C'de verilmiştir.

ALIŞTIRMA **3.12** *ADD8 programını çalıştırarak kütle çekim problemini yeniden çözünüz. Bu programın bir çıktısı Tablo 3.4'de gösterilmiştir. Bu sonuçları Tablo 3.3'de gösterilen program ADD7'nin sonuçları ile karşılaştırınız. Hangisi daha hassas? Hangi programda fonksiyon hesaplaması en az?*

Tablo 3.4'de görüldüğü gibi şimdi kullanılan hata toleransına uygun olarak yapılan hatalar 10^{-6} mertebesindedir. Adım uzunluğu parçacığın merkeze en yakın ve en hızlı olduğu anda en küçük değeri almıştır. Bu programla elde edilen yörüngenin grafiği Şekil 3.2'de gösterilmiştir. Toplam integral süresi 10 periyota eşittir. Çizimde (x, y) noktalarının arasındaki uzaklığın küçük olmasını sağlamak üzere P küçük seçilerek daha çok veri noktası elde edilmiştir. Hatanın en az olduğu bir hesaplamada yörüngenin sürekli elips olarak kalması gerekir.



Şekil 3.2: (3.15)'de verilen kütle çekim probleminde parçacığın 10 periyot boyunca takip ettiği yörünge. Parçacığın yörüngesinin sabit kalması çözümün iyi olduğuna işaret eder.

Bu bölümde gördüğümüz en hassas program RKV metoduna dayalı ADD8 programıdır. Bundan sonra çözdüğünüz başlangıç şartlı problemlerin hepsinde sadece RKV alt programını kullanmanız hem daha hızlı hem de daha hassas sonuçlar elde etme açısından daha uygun olacaktır.

ALIŞTIRMA **3.13** *Denklem (3.7)'de verilen basit harmonik salınıcı denklemini $t_0 = 0$'da, $x(0) = 0$ ve $\dot{x}(0) = 1$ başlangıç şartlarını kullanarak çözmek için ADD8 programında gerekli değişiklikleri yapınız. Yeni programı ADD81 olarak adlandırınız. Analitik çözümüde bularak sayısal çözümü test etmek için kullanınız. HTOL=1×10^{-6} seçiniz. $m = w = 1$ olduğunu kabul ediniz.*

a) ADD81 programında L=600, P=0.1 seçerek elde ettiğiniz x ve $\dot{x}$ değerlerini zamanın (t) bir fonksiyonu olarak çiziniz. Şekil 3.3'de gösterilen grafiği elde etmeniz gerekir.

b) Analitik çözüm ile sayısal çözüm arasındaki farkı (yapılan hatayı) zamanının (t) bir fonksiyonu olarak çiziniz. Şekil 3.4'de gösterilen sonuçları bulmanız gerekir. Hatanın çok küçük de olsa salınım yapıyor ve gittikçe artıyor olmasına dikkat ediniz.

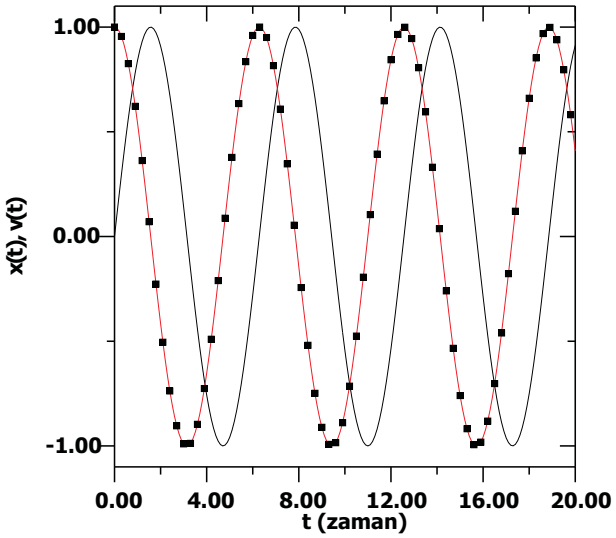
c) Basit harmonik salınıcının enerjisi

$$E = \frac{1}{2}m\dot{x}^2 + \frac{1}{2}mw^2x^2$$

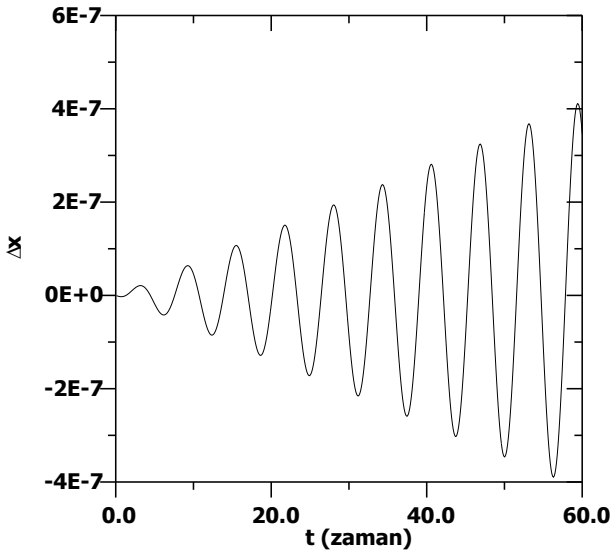
ile verilir. $m = w = 1$ olduğu varsayılırsa yukarıdaki enerji denklemi yarıçapı $\sqrt{2E}$ olan bir daire denklemi olacaktır. $\dot{x}$'i x'e karşılık çizerek daire elde edildiğini gösteriniz (Bu aynı zamanda faz uzayı olarak da bilinir). Bu grafiği en az 10 periyot için çizerek sonucun daireden ne kadar uzaklaştığını görmeye çalışınız. Şekil 3.5'de gösterilen sonuca benzer bir grafik elde etmeniz gerekir.

3.7 Stif Problemler

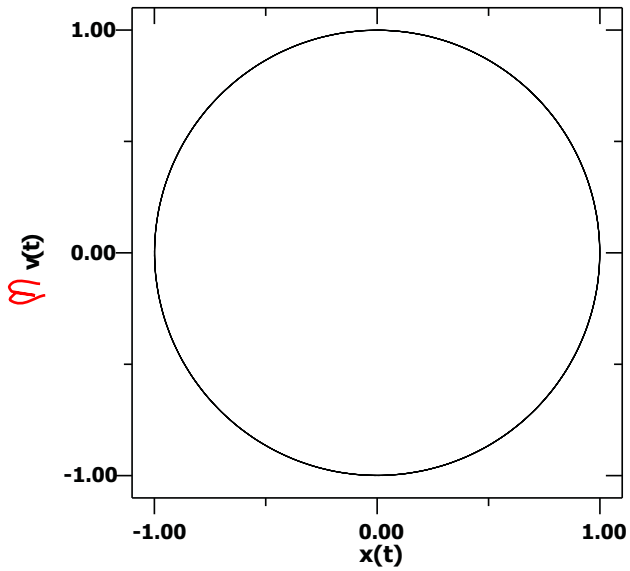
Çözümünün bağımsız değişkene bağımlılığı çok farklı ölçeklerde eksponansiyel terimler içeren diferansiyel denklemlerine stif denklemler denir. Bir diferansiyel denkleminin çözümünün e^{-x} ile e^{-50x} terimlerinin lineer bir kombinasyonu ile verildiğini varsayalım. Bu iki terimin bağımsız değişkene bağımlılıkları çok farklı ölçeklerdedir. Bu çözümlerden biri çok hızlı şekilde azalırken diğeri uzun bir



Şekil 3.3: (3.7)'de verilen basit harmonik salınıcı probleminin sayısal çözümünden bulunan parçacığın yerdeğişim ve hızının (kareler) zamanın bir fonksiyonu olarak değişimi.



Şekil 3.4: Şekil 3.3'de gösterilen harmonik salınıcının x koordinatında yapılan hatanın zamanla değişimi.



Şekil 3.5: Şekil 3.3'de gösterilen harmonik salınıcının $v(t)$-$x(t)$ grafiği.

değer aralığı için (diğeri ile karşılaştırıldığında) sonlu kalacaktır. Çok farklı boşalma zamanları olan kapasitörler bu yapıya uygun örneklerdir. Daha ayrıntılı bir inceleme için aşağıdaki diferansiyel denklemini göz önüne alalım.

$$\frac{d^2y}{dx^2} - 100y = 0 \tag{3.59}$$

Bu denklemin genel çözümü, c_1 ve c_2 keyfi sabitlar olmak üzere, $y = c_1e^{-10x} + c_2e^{10x}$ ile verilir. $y(0) = 1$ ve $y'(0) = -10$ başlangıç şartları kullanılırsa $c_1 = 1, c_2 = 0$ olduğu gösterilebilir. Bu durumda analitik çözüm $y(x) = e^{-10x}$ olacaktır. Bu problemi sayısal olarak çözmeye çalıstığımızda, hızla artan e^{10x} terimine ait en ufak bir bileşenin sayısal çözüme bulaşması halinde bu bileşen hızla büyüyecek ve kısa sürede bütün çözümü etkisi altına alacaktır. Bu durumda problemin aslında çözümü olan e^{-x}'den hızla uzaklaşılacak ve çözüm ıraksıyacaktır. Bu tip problemlerin çözümünda Runge-Kutta metotlarının hemen hepsinin çok kötü sonuçlar verdiği bilinmektedir. Bu nedenle stif problemler RK yöntemleri ile hiç bir zaman çözülmemelidir. Bunun yerine çok adımlı yöntemler temkinli şekilde kullanılabilir.

ALIŞTIRMA **3.14** *ADD8 programını (3.59)'da verilen diferansiyel denklemini çözecek şekilde ADD82 adı altında yeniden düzenleyiniz. Sayısal ve analitik çözümün grafiklerini beraber çizerek sayısal çözümün hızla gerçek çözümden uzaklaştığını gözleyiniz. Farklı L ve P değerleri kullanarak sayısal çözümü gerçek çözümden fazla uzaklaşmadan ne kadar sürdürebildiğinize bakınız.*

3.8 Sayısal Çözümler İçin Bazı Öneriler

Diferansiyel denklemlerini sayısal olarak çözmeye başlamadan önce ve çözüm elde edildikten sonra aşağıda belirtilen konuların mümkün olduğu yerlerde göz

önüne alınması gerekir. Böylece hem denklem çözme işi kolaylaşacak hemde elde edilen çözümlerin gerçekten aranan çözüm olup olmadığı konusunda karar vermek daha kolay olacaktır.

a) Sadece belirli parametre değerlerinde veya bazı limitlerde geçerli bile olsa diferansiyel denklemlerin analitik çözümleri her zaman faydalıdır. Analitik çözümler çeşitli şartlar altında bulunabiliyorsa mutlaka elde edilmeli ve sayısal çözümün kontrol edilmesinde kullanılmalıdır.

b) Diferansiyel denkleminde bulunan sabitler mümkün olduğunca sadeleştirilerek en az sayıya indirilmelidir. Bu zaman kazandıracak, bir çok parametre kullanmaktan doğan karmaşıklığı önleyecek ve sonuçların yorumlanmasını kolaylaştıracaktır.

c) Mümkün olan durumlarda değişkenler yeniden ölçeklenerek boyutsuz hale getirilmelidir.

d) Çözümde kullanılacak birimlere çözümden önce karar verilmelidir. Uygun olmayan birimlerin kullanılması durumunda yazılan program hiç çalışmayabilir. Örneğin galaksilerin dinamiğini veya kuantum mekaniği problemlerini çalışmak üzere SI birimlerini kullanmak felaketle sonuçlanabilir. Galaksi problemlerinde R^3 gibi terimler kolayca ortaya çıkabilir ve bu sayı kullanılan makinede temsil edilebilecek en büyük sayıdan daha büyük olabilir. Benzer şekilde kuantum mekaniği problemlerinde $\hbar^4$ gibi terimler ortaya çıkabilir ve bu sayı makinede temsil edilebilecek en küçük sayıdan daha küçük olabilir. Bu durumda makine bu sayıyı sıfır olarak alacak ve elde edilen sonuçlar anlamsız olacaktır.

Göz önüne alınan problemin çoğunlukla kendi doğal birimleri vardır ve bu birimlerde çalışmak daha akıllıcadır. Doğal birimler kullanıldığında değişkenler çok büyük veya çok küçük olmaktansa çoğunlukla 1 mertebesindedir. Bir problemin doğal birimleri çoğunlukla ne SI nede cgs'dir.

e) Elde edilen sayısal çözümler mutlaka çeşitli yöntemlerle test edilmelidir. Kontrol için kullanılanlar çoğunlukla çözülen problemin matematiksel olmaktan ziyade fiziksel olarak sağlaması gereken şartlardır. Örneğin problemde korunması gereken değerler varsa (enerji, momentum vb.) bunların korunup korunmadığı kontrol edilmelidir. Problemin analitik çözümü çeşitli limitlerde biliniyorsa, sayısal çözüm ile analitik çözüm bu limitlerde karşılaştırılmalıdır.

Yukarıdaki önerilerin bir kısmını uygulayabileceğimiz bir örnek göz önüne alalım. Elektrik ve manyetik alanlar altında hareket eden yüklü bir parçacığın yörüngesini ADD8 programını uygun şekilde değiştirerek bulmaya çalışalım.

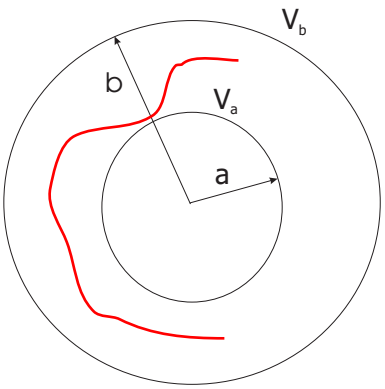
3.8.1 Örnek: Radyal Elektrik Alanı Altında Hareket Eden Yüklü Bir Parçacık

Şekil 3.6'da gösterildiği gibi yarıcapları a ve b $(b > a)$ olan iki silindirik elektrot sırası ile V_a ve V_b potansiyellerinde tutulursa, bunların arasında

$$E = \frac{K}{r}, \qquad K = \frac{V_a - V_b}{log(b/a)} \tag{3.60}$$

ile verilen radyal bir elektrik alanı oluşacaktır. K sabiti V_a ve V_b'nin göreli değerlerine göre pozitif veya negatif olabilir. Q yüklü, m kütleli bir parçacığın bu alan altındaki hareket denklemi, $\mathbf{a}$ ivme olmak üzere, $Q\mathbf{E} = m\mathbf{a}$ olacaktır. Parçacığın yükünün negatif olduğu $(Q = -q,\ q > 0)$ ve hareketin tamamen silindirik elektrotlara dik iki boyutlu bir düzlem üzerinde olduğu varsayılarak hareket denklemi

$$\frac{d^2\mathbf{r}}{dt^2} = -(\frac{qK}{m})\frac{\hat{\mathbf{r}}}{r} = -(\frac{qK}{m})\frac{\mathbf{r}}{r^2} \tag{3.61}$$



Şekil 3.6: Sırası ile V_a ve V_b potansiyellerinde tutulan a ve b yarıçaplı aynı merkezli iki silindirik elektrotun üstten görünüşü.

olarak yazılabilir.

a) Problemde geçen en büyük uzunluk dışta bulunan silindirin yarıçapı olacağından uzunlukları b'ye bölerek ölçeklendirelim. Yeni boyutsuz uzunlukları $r' = r/b$ ile gösterelim. Bu durumda (3.61) denklemi

$$\frac{d^2\mathbf{r}'}{dt^2} = -(\frac{qK}{mb^2})\frac{\mathbf{r}'}{r'^2} = -\alpha\frac{\mathbf{r}'}{r'^2} \tag{3.62}$$

olarak yeniden yazılabilir. Burada $\alpha = \frac{qK}{mb^2}$'dır.

b) α'nın biriminin zamanın karesinin tersi olduğunu gösteriniz.

c) (3.62)'deki zamanı $t' = \sqrt{\alpha}t$ şeklinde yeniden ölçeklendiriniz ve ortaya çıkan denklemin

$$\frac{d^2\mathbf{r}}{dt^2} = -\frac{\mathbf{r}}{r^2} \tag{3.63}$$

olduğunu gösteriniz. Bu son denklemde karmaşıklığı önlemek için değişkenlerin üsleri yazılmamıştır ve görüldüğü gibi denklemde hiç bir parametre kalmamıştır. Dikkat edilirse bu denklem daha önce Örnek 3.3'de gördüğümüz kütle çekim problemindeki denklemlere çok benzemektedir.

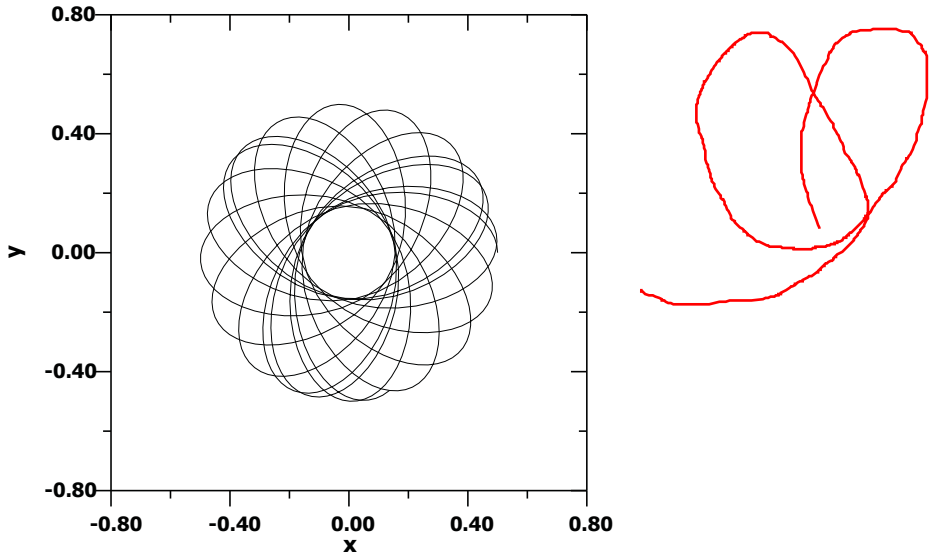
d) (3.63)'deki ikinci mertebe diferansiyel denklemini $\mathbf{r} = x\hat{\mathbf{i}} + y\hat{\mathbf{j}}$ kullanarak iki boyutlu (x, y) düzleminde dört tane birinci mertebe diferansiyel denkleme indirgeyiniz. (3.15)'de elde edilen denklemlere benzer denklemler elde etmeniz gerekir.

e) Ortaya çıkan denklemleri ADD8 programını uygun şekilde değiştirerek $x = 0.5$, $y = 0$, $V_x = 0$ and $V_y = 0.5$ başlangıç şartlarını kullanarak $t \in [0, 30]$ çözünüz. Sonuçlarınızı (t, x, y, V_x, V_y) bir dosyaya yazdırınız ve y-x grafiğini çizerek parçacığın izlediği yörüngeyi görünüz. Ne elde ettiniz? Elde etmeniz gereken yöründe Şekil 3.7'de gösterilmiştir.

f) (e)'yi $t \in [0, 120]$ için tekrar ediniz. Parçacığın yörüngesi bir dış ve bir de iç bir daireye teğetler yapan elliptik yörüngeler olmalıdır. Bu şekilden faydalanarak bu parçacığın yörüngesinin kapalı olup olmadığına karar vermeye çalışınız.

g) Değişkenlerin zamanla nasıl değiştiğini görmek için x-t, y-t, V_x-t, V_y-t grafiklerini çiziniz. Ayrıca V_x-x ve V_y-y grafiklerini çiziniz. Bu çizim aynı zamanda faz uzayı olarak bilinir. Faz uzayının her noktası ziyaret ediliyor mu?

(**Not:** (3.61)'deki diferansiyel denklemi silindirik koordinatlarda da yazılabilir. Yörünge problemlerinde kullanılan standart değişken değişimi $z =$



Şekil 3.7: Sırası ile V_a ve V_b potansiyellerinde tutulan a ve b yarıçaplı aynı merkezli iki silindirik elektrotun arasında hareket eden yüklü bir parçacığın yörüngesi.

$1/r$ kullanılarak bu denklem, zamanın denklemlerde açıkça görülmediği ikinci mertebe bir denkleme dönüştürülebilir. Zaman yerine silindirik koordinatlardaki açı değişkeni probleme bir parametre olarak girer. Uygun değişken değişimleri ile ölçekleme yapılarak bu denklem de boyutsuz hale getirilebilir. Yalnız bu durumda çözülmesi gereken dört tane birinci mertebe diferansiyel denklem yerine sadece iki tane birinci mertebe diferansiyel denklem olacaktır).

3.8.2 Proje: Elektrik ve Manyetik Alanları Altında Hareket Eden Klasik Elektronlar

Bu bölümde ögrendiklerimizin pekiştirilmesi için aşağıdaki problemi çözmeye çalışınız.

B manyetik ve **E** elektrik alanı altında hareket eden kütlesi m, yükü q olan bir parçacığın hareket denklemi

$$m\frac{d^2\mathbf{r}}{dt^2} = q[\mathbf{V}\times\mathbf{B}+\mathbf{E}] - \gamma\mathbf{V} \tag{3.64}$$

ile verilir. Burada **V** parçacığin hızı olup $\gamma\mathbf{V}$ terimi parçacığın içinde hareket ettiği ortamdan kaynaklanan bir çeşit sürtünme kuvvetini temsil etmektedir.

a) Eletrik alanının sadece x-bileşeni ($\mathbf{E}$=$(E,0,0)$) ve manyetik alanın sadece z-bileşeni ($\mathbf{B}$=$(0,0,B)$) olduğunu, parçacığın başlangıçta z-yönünde hızı olmadığını varsayarak hareket denkleminin

$$\begin{aligned}\frac{d^2x}{dt^2} &= (\frac{qB}{m})V_y - (\frac{\gamma}{m})V_x + (\frac{qE}{m})\\ \frac{d^2y}{dt^2} &= -(\frac{qB}{m})V_x - (\frac{\gamma}{m})V_y\end{aligned} \tag{3.65}$$

olması gerektiğini gösteriniz. Veya hızın bileşenlerini türevli halleri ile yazarsak

$$\begin{aligned} \frac{d^2x}{dt^2} &= (\frac{qB}{m})\frac{dy}{dt} - (\frac{\gamma}{m})\frac{dx}{dt} + (\frac{qE}{m}) \\ \frac{d^2y}{dt^2} &= -(\frac{qB}{m})\frac{dx}{dt} - (\frac{\gamma}{m})\frac{dy}{dt} \end{aligned} \tag{3.66}$$

elde ederiz.

b) Yukarıdaki denklemler dikkatlice incelendiğinde bu problemde üç tane bağımsız sabit olduğu görülecektir. Bunlar $(\frac{qB}{m})$, $(\frac{\gamma}{m})$ ve $(\frac{qE}{m})$'dir. $(\frac{qB}{m})$ ve $(\frac{\gamma}{m})$ sabitlerinin biriminin zamanın tersi olduğunu gösteriniz. Aslında $w_c = (\frac{qB}{m})$ *siklotron frekansı* ve $b = (\frac{\gamma}{m})$ *sönüm oranı* sabiti olarak bilinir. Bu nedenle bu problemde iki tane doğal zaman birimi vardır. Zamanı ya siklotron frekansının tersi veya sönüm oranı sabitinin tersi cinsinden ölçebiliriz. Bunlardan örneğin siklotron frekansını kullanmaya karar verirsek zaman $t' = w_c t$ şeklinde yeniden ölçeklendirilerek birimsiz hale getirilebilir. Bu durumda problemin zamanı artık siklotron frekansının tersi cinsinden ölçülmektedir.

c) (3.66)'da $t' = w_c t$ değişken değişimini yaparak

$$\begin{aligned} \frac{d^2x}{dt'^2} &= \frac{dy}{dt'} - (\frac{b}{w_c})\frac{dx}{dt'} + \alpha \\ \frac{d^2y}{dt'^2} &= -\frac{dx}{dt'} - (\frac{b}{w_c})\frac{dy}{dt'} \end{aligned} \tag{3.67}$$

denklemlerini elde ediniz. Burada $\alpha = (\frac{qE}{mw_c^2})$'dir.

d) α'nın biriminin uzunluk olduğunu gösteriniz. Böylece problemde geçen uzunlukları α'yı kullanarak birimsiz hale getiriniz. $x' = x/\alpha$ ve $y' = y/\alpha$ değişken değişimleri yapılırsa ortaya çıkan denklemler

$$\begin{aligned} \frac{d^2x'}{dt'^2} &= \frac{dy'}{dt'} - (\frac{b}{w_c})\frac{dx'}{dt'} + 1 \\ \frac{d^2y'}{dt'^2} &= -\frac{dx'}{dt'} - (\frac{b}{w_c})\frac{dy'}{dt'} \end{aligned} \tag{3.68}$$

olacaktır. Son denklem serisindeki üsleri düşürerek

$$\begin{aligned} \frac{d^2x}{dt^2} &= \frac{dy}{dt} - (\frac{b}{w_c})\frac{dx}{dt} + 1 \\ \frac{d^2y}{dt^2} &= -\frac{dx}{dt} - (\frac{b}{w_c})\frac{dy}{dt} \end{aligned} \tag{3.69}$$

denklemleri elde edilir. Son denklemlerde problemin doğal zamanlarının oranı olan sadece bir parametre olmasına dikkat ediniz. Bu denklemlerdeki değişkenler birimsiz olup, uzunluklar α, zaman ise siklotron frekansının tersi cinsinden ölçülmektedir.

e) (3.69)'daki denklemleri dört tane birinci mertebe diferansiyel denklemine dönüştürünüz ve (b/w_c)'nin ve başlangıç şartlarının çeşitli değerleri için bu denklemleri ADD8 programını uygun bir şekilde değiştirerek çözünüz. Örneğin $x = 0$, $y = 0$, $V_x = 1$, $V_y = 1$ başlangıç şartlarını ve $b/w_c = 0.01$ parametre değerini kullanarak $t \in [0, 160]$ aralığı için çözüm yapınız. t, x, y, V_x ve V_y değerlerini bir dosyaya yazdırarak x-t, y-t, y-x, V_x-t ve V_y-t grafiklerini çiziniz. Elektrik alanı yönünde düzgün doğrusal hareket olmamasına dikkat ediniz. Hareket negatif y yönüne doğrudur. Aslında bu $\mathbf{E} \times \mathbf{B}$ vektörel çarpımının yönü ile aynıdır ve bu sebepten $\mathbf{E} \times \mathbf{B}$ sürüklenmesi olarak bilinir. Faz uzayının (V_x-x ve V_y-y) grafiklerini de çiziniz.

3.9 Verlet Algoritması ve Moleküler Dinamik

Bir parçacığa etkiyen net kuvvet $\mathbf{F}$ ve parçacığın konumu $\mathbf{r}(\mathbf{t})$ ise bilindiği gibi Newton'un ikinci hareket yasasına göre

$$\frac{d^2\mathbf{r}}{dt^2} = \mathbf{F} \tag{3.70}$$

yazılabilir. Burada kuvvet en genel haliyle zamanın, hızın, koordinatların veya bunlardan birkaçının veya hepsinin birden fonksiyonu olabilir. Eğer parçacık korunumlu bir kuvvet altında hareket ediyorsa bu durumda kuvvet koordinatların bir fonksiyonudur ve uygun bir skaler potansiyel fonksiyonunun gradiyenti şeklinde yazılabilir:

$$\mathbf{F} = -\nabla U(\mathbf{r}) \tag{3.71}$$

3.9.1 Hızdan Bağımsız Verlet Algoritması

Verlet[1] tarafından geliştirilen algoritma, parçacığın konumunun zamana göre aşağıdaki gibi iki farklı şekilde Taylor serisine açılması ile elde edilir:

$$\mathbf{r}(t+\Delta t) = \mathbf{r}(t) + \Delta t\frac{d\mathbf{r}}{dt} + \frac{1}{2}(\Delta t)^2\frac{d^2\mathbf{r}}{dt^2} + \frac{1}{6}(\Delta t)^3\frac{d^3\mathbf{r}}{dt^3} + O(\Delta t)^4 \tag{3.72}$$

$$\mathbf{r}(t-\Delta t) = \mathbf{r}(t) - \Delta t\frac{d\mathbf{r}}{dt} + \frac{1}{2}(\Delta t)^2\frac{d^2\mathbf{r}}{dt^2} - \frac{1}{6}(\Delta t)^3\frac{d^3\mathbf{r}}{dt^3} + O(\Delta t)^4 \tag{3.73}$$

Yukarıdaki iki denklem taraf tarafa toplanırsa

$$\mathbf{r}(t+\Delta t) = 2\mathbf{r}(t) - \mathbf{r}(t-\Delta t) + (\Delta t)^2\mathbf{a}(t) + O(\Delta t)^4 \tag{3.74}$$

elde edilir. Burada $\mathbf{a}$ ivme olup, m parçacığın kütlesi olmak üzere potansiyel fonksiyonundan

$$\mathbf{a}(t) = \frac{1}{m}\mathbf{F}(t) = -\frac{1}{m}\nabla U(\mathbf{r}(t)) \tag{3.75}$$

elde edilir. Görüldüğü gibi Verlet algoritması üçüncü mertebeden hassas yaklaşık bir yöntemdir. $t+\Delta t$ anındaki konumun bilinmesi için $t-\Delta t$ ve t anındaki konumların yanısıra t anındaki parçacığın ivmesininde bilinmesi gerekir. Üç boyutlu bir problemde bunların her bir değeri için üç bileşen olduğundan bir tek parçacğın bir sonraki zaman adımındaki konumunun bilinmesi için dokuz değerin biliniyor olması gerekir. Burada dikkat edilirse parçacığın hızı işlemlere girmemektedir. Bu nedenle denklem (3.74)'de verilen algoritma hızdan bağımsız Verlet algoritması olarak bilinir.

3.9.2 Hıza Bağımlı Verlet Algoritması

Bazı durumlarda parçacığın sadece konumu değil, hızınında bilinmesi gerekebilir. Örneğin kinetik enerji hesaplamaları için hızlar gerekecektir. Daha az hassas olmakla beraber $t+\Delta t$ anındaki konum ve hız, t anındaki değerlerinin Taylor serisi açılımlarından elde edilebilir. $\mathbf{v}(t) = d\mathbf{r}/dt$ olmak üzere

$$\mathbf{r}(t+\Delta t) = \mathbf{r}(t) + \Delta t\mathbf{v}(t) + \frac{1}{2}(\Delta t)^2\mathbf{a}(t) \tag{3.76}$$

$$\mathbf{v}(t+\Delta t/2) = \mathbf{v}(t) + \frac{1}{2}(\Delta t)\mathbf{a}(t) \tag{3.77}$$

$$\mathbf{a}(t+\Delta t) = -\frac{1}{m}\nabla U(\mathbf{r}(t+\Delta t)) \tag{3.78}$$

$$\mathbf{v}(t+\Delta t) = \mathbf{v}(t+\Delta t/2) + \frac{1}{2}(\Delta t)\mathbf{a}(t+\Delta t) \tag{3.79}$$

Yukarıdaki denklemler sırası ile kullanıldığında $t+\Delta t$ anındaki konum ve hız bulunabilir. t anındaki konum bilindiğinden prensipte bu konuma karşılık gelen kuvvet (ivme) bulunabilir. t anındaki hızda bilindiğinden, denklem (3.76) $t+\Delta t$ anındaki konum bulunabilir. (3.77) denklemi daha sonra kullanılmak üzere hızın bir ara değerini bulur. $t+\Delta t$ anındaki konum bilindiğinden, bu andaki konuma karşılik gelen kuvvet (ivme) de bulunabilir. Denklem (3.78) bunu ifade eder.

3.10 Labaratuar Saati-III.a

Bu bölümle ilgili programlar Ek-D'de verilmiştir.

1.a Size sağlanan ADD4.FOR programını

$$\frac{dy}{dx} = x^2 + y \qquad y(0) = 1$$

diferansiyel denklemini çözecek şekilde başka bir ad altında yeniden düzenleyiniz. İntegrasyonun başlangıcından sonuna kadar olan x değerlerini ve bunlara karşı gelen sayısal (y_s), analitik (y_a) çözümleri ve onların farkı $\Delta y = y_s - y_a$ değerlerini bir dosyaya yazdırınız.

1.b Yazdırdığınız dosyadaki verileri kullanarak $x - y_s$ ve $x - y_a$ grafiklerini beraber çizerek sayısal ve analitik çözümlerin nasıl değiştiğini görünüz. Ayrıca $x - \Delta y$ grafiğini çizerek hatanın nasıl değiştiğine bakınız. Hata sıfır etrafında salınım yapıyor mu? Hata sürekli artıyor mu? Bunlardan sizce hangisi daha iyi?

2. İki-Adımlı Ruge-Kutta Metodu:

$$\frac{dy}{dx} = f(x,y) \qquad y(x_0) = y_0 \tag{3.80}$$

başlangıç şartlı diferansiyel denklemini çözmek için iki adımlı Runge-Kutta yönteminin

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_n + \alpha h, y_n + \beta k_1) \end{aligned} \tag{3.81}$$

olmak üzere

$$y_{n+1} = y_n + ak_1 + bk_2 \tag{3.82}$$

ile verildiği ve α, β, a ve b arasındaki ilişkinin

$$\begin{aligned} a + b &= 1 \\ b\alpha = b\beta &= \frac{1}{2} \end{aligned} \tag{3.83}$$

olması gerektiği daha önce gösterilmişti. Burada y_{n+1} diferansiyel denkleminin $x = x_0 + nh + h = x_n + h$ noktasındaki yaklaşık çözümüdür.

2.a $a = 0$ seçilirse, bu durumda $b = 1$ ve $\alpha = \beta = \frac{1}{2}$ olur. Parametrelerin bu değerleri için Runge-Kutta denklemleri daha önce elde ettiğimiz Euler-Richardson denklemleri ile tamamen aynı olur.

2.b Başka olası bir çözüm kümesi $a = b = \frac{1}{2}$ ve $\alpha = \beta = 1$ olacaktır.

İki adımlı Runge-Kutta denklemini uygulamak üzere ADD4 programı değiştirilmiş ve yeni program ADD5 olarak adlandırılmıştır. Bu programın bir kopyesini elde ediniz. Her adım için burada iki defa eğim hesaplanması gerekmektedir. ERICS alt programı RK2 ile değiştirilmiştir. RK2 alt programını inceleyerek her adımda programda sırası ile XK1 ve XK2 olarak adlandırılan k_1 ve k_2 değerlerinin nasıl hesaplandığını görünüz.

3.a ADD5 programını kullanarak

$$\frac{dy}{dx} = y + x, \quad y(x_0) = y_0 \tag{3.84}$$

diferansiyel denklemini $y(0) = 1$ başlangıç şartını kullanarak çözünüz. Bu denklemin analitik çözümünün

$$y(x) = (y_0 + x_0 + 1)e^{(x-x_0)} - (x + 1) \tag{3.85}$$

olması gerektiği daha önce gösterilmişti. Elde ettiğiniz çözümü ADD4 programının sonuçları ile karşılaştırınız.

3.b ADD5 programında a, b, α ve β parametreleri için denklem (3.73) ile tutarlı farklı değerler kullanarak (3.a)'yı yeniden çözünüz. Çözümde ve hassasiyet mertebesinde önemli bir değişiklik meydana geliyormu?

3.c ADD5 programını ADD51 adı altında

$$\frac{dy}{dx} = x^2 + y, \qquad y(0) = 1$$

diferansiyel denklemini çözecek şekilde değiştiriniz. Bu denklemin analitik çözümü daha önce gösterildiği gibi $y(x) = -(x^2 + 2x + 2) + 3e^x$ ile verilir.

4. Dört Adımlı Runge-Kutta Metodu

İki adımlı Runge-Kutta (RK) metodunun çıkarılışında izlenen yol kullanılarak dört adımlı daha hassas yöntemde elde edilebilir. Dört adımlı Runge-Kutta metodunda (3.70)'de verilen diferansiyel denkleminin çözümünün $x_{n+1} = x_n + h$ noktasındaki yaklaşık değerini

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{3.86}$$

şeklinde yazıyoruz. Burada k_i'ler

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_n + \frac{h}{2}, y_n + \frac{1}{2}k_1) \\ k_3 &= hf(x_n + \frac{h}{2}, y_n + \frac{1}{2}k_2) \\ k_4 &= hf(x_n + h, y_n + k_3) \end{aligned} \tag{3.87}$$

olarak tanımlanmıştır. Yukarıda verilen yöntemde her adımda yapılan yerel hata $O(h^5)$ mertebesindedir. Diğer yöntemlere göre hata çok daha küçük fakat bunun için ödenen bedel her adımda $f(x, y)$ fonksiyonunun dört defa hesaplanmasıdır. Dikkat edilirse her mertebeden Runge-Kutta metodu çıkartmak her zaman olasıdır fakat hesaplamalar çok karmaşık ve uzun olabilir.

5. ADD5 programını başka bir ad altında, örneğin ADD52, değiştirerek yukarıda verilen dört adımlı Runge-Kutta algoritmasını programlayınız. Yazdığınız bu programı kullanarak (1.a)'yı yeniden çözünüz. Sonuçlarınızı daha

önce elde ettiğiniz bütün sonuçlar ile karşılaştırınız. En hassas olan yöntem hangisidir?

6. İki veya Daha Yüksek Mertebeli Diferansiyel Denklemlerin Çözümü

Daha önce gördüğümüz gibi $n.$ mertebeden bir diferansiyel denklemi n tane birinci mertebe diferansiyel denklem halinde yazılabilir. Örneğin ikinci mertebeden bir diferansiyel denklem olan basit harmonik salınıcı denklemi

$$\frac{d^2x}{dt^2} + w^2x = 0 \tag{3.88}$$

iki tane birinci mertebe diferansiyel denklemi şeklinde yazılabilir. Bunu yapmak için $y_1 = x(t)$ ve $y_2 = \frac{dx}{dt}$ tanımlarını kullanarak denklem (9)'u yeniden yazalım. Bu durumda

$$\begin{aligned} \frac{dy_1}{dt} &= y_2 \\ \frac{dy_2}{dt} &= -w^2y_1 \end{aligned} \tag{3.89}$$

elde edilecektir.

Daha karmaşık bir örnek ise ters kare kuralına uyan merkezi bir kuvvet (örneğin kütle çekim) altında (x, y) düzleminde iki boyutta bir yörüngede dönen bir parçacığın hareketini temsil eden diferansiyel denklem olabilir. G problemin sabitlerini temsil etmek ve $\mathbf{r} = x\mathbf{i} + y\mathbf{j}$ olmak üzere, hareket denklemi

$$\frac{d^2\mathbf{r}}{dt^2} = -\frac{G\hat{\mathbf{r}}}{r^2} = -\frac{G\mathbf{r}}{r^3} \tag{3.90}$$

şeklinde yazılabilir. Bu denklem görüldüğü gibi ikinci mertebe bir diferansiyel denklem olup iki boyutta yazılmıştır. Bu diferansiyel denklem her boyut için ayrı ayrı yazılabilir. Böylece

$$\begin{aligned} \frac{d^2x}{dt^2} &= -\frac{Gx}{r^3} \\ \frac{d^2y}{dt^2} &= -\frac{Gy}{r^3} \end{aligned} \tag{3.91}$$

elde edilir. Bunlar kuplajlı ikinci mertebe diferansiyel denklemlerdir.

(3.81)'de verilen denklemler birer mertebe daha indirgenerek birinci mertebe dört denklem elde edilebilir. Bunu yapmak için hızın x ve y yönlerindeki bileşenlerini sırası ile $V_x = \frac{dx}{dt}$ ve $V_y = \frac{dy}{dt}$ olarak yazabiliriz. Bu tanımları kullanarak

$$\begin{aligned} \frac{dx}{dt} &= V_x \\ \frac{dy}{dt} &= V_y \\ \frac{dV_x}{dt} &= -\frac{Gx}{r^3} \\ \frac{dV_y}{dt} &= -\frac{Gy}{r^3} \end{aligned} \tag{3.92}$$

şeklinde dört tane kuplajlı birinci mertebe diferansiyel denklem elde edilebilir. Bu denklemleri daha düzenli yazmak için $y_1 = x(t)$, $y_2 = y(t)$, $y_3 = \frac{dx}{dt}$ ve

$y_4 = \frac{dy}{dt}$ tanımlarını kullanarak (3.82)'deki denklemler yeniden

$$\begin{aligned} \frac{dy_1}{dt} &= y_3 \\ \frac{dy_2}{dt} &= y_4 \\ \frac{dy_3}{dt} &= -\frac{Gy_1}{r^3}, \qquad r = \sqrt{y_1^2 + y_2^2} \\ \frac{dy_4}{dt} &= -\frac{Gy_2}{r^3} \end{aligned} \tag{3.93}$$

şeklinde yazılabilir.

Yüksek mertebeli herhangi bir diferansiyel denklem yukarıda örnekleriyle gördüğümüz gibi birinci mertebe diferansiyel denklemler dizisi halinde yazılabildiğinden, başlangıç şartlı adi diferansiyel denklemleri sayısal olarak çözme yöntemleri sadece birinci mertebe diferansiyel denklemler için geliştirilmiştir.

Şimdiye kadar kullandığımız bütün programlar ve alt programlar sadece bir diferansiyel denkleminin çözümü için yazılmıştır. N tane denklemden oluşan bir diferansiyel denklem sistemini çözmek için değişkenlerin vektörler şeklinde yazılması gerekir. Yani gerekli değişkenler boyutlandırılmalıdır. Boyutlandırmaların nasıl yapılabileceğini göstermek üzere yukarıdaki son örnekte elde edilen (3.83)'deki denklem sisteminin çözülmesi için ADD4 programı ADD6 adı altında yeniden düzenlenmiştir. Bu programı ağ komşuluğundan alınız.

Yeni programdaki bazı değişiklikleri not edelim. Toplam dört tane birinci mertebe diferansiyel denklem olduğu için 'PARAMETER' komutunda N=4 olarak verilmiş ve bu değer daha sonra boyutlu değişkenlerin boyutlarının tanımlanmasında kullanılmıştır. Başlangıç değerleri Y0 adlı vektörde tutulmuş, parçacığın x ve y yönündeki koordinatlarına sırası ile Y(1) ve Y(2), x ve y yönündeki hızlarına ise sırası ile Y(3) ve Y(4) adı verilmiştir. S1, S2 ve S3 vektörleri fonksiyon değerlerini hesaplayıp tutmak ve ara işlemleri yapmak için kullanılmaktadır. Programda tanımlanan CT değişkeni DEQ alt programının kaç defa çağrıldığını yani kaç defa fonksiyon değer hesaplaması yapıldığını kaydetmektedir. Programı yakından inceleyerek diğer değişiklileri sizin bulmanız ve anlamaya çalışmanız yararlı olacaktır.

7.a ADD6 programını program içinde verilen $x_0 = 1$, $y_0 = 0$, $V_x = 0$ ve $V_y = 0.5$ başlangıç şartlarını kullanarak derleyip çalıştırınız. Bu başlangıç şartları Y0'lar cinsinden Y0(1)=1, Y0(2)=0, Y0(3)=0 ve Y0(4)=0.5 olacaktır.

ADD6 programında verilen toplam integrasyon süresi L×P = 2.714080941 bu parçacığın elips olan yörüngesini tamamlaması için gereken süreye, yani yörünge periyoduna eşittir. Bu nedenle bir periyot sonunda parçacık başladığı yere dönmeli ve buraya ulaştığı andaki hızları ile başlangıçtaki hızları eşit olmalıdır. Ayrıca parçacığın toplam enerjiside korunmalıdır (neden?). Bazı problemlerdeki bunlara benzer zamanla değişmeyen hareket sabitleri, kullanılan sayısal yöntemi test etmek için kullanılabilecek çok önemli parametrelerdir. Göz önüne aldığımız örnekte yörüngenin kapalı elips olması nedeni ile parçacık başladığı yere dönmelidir. Ayrıca başlangıç noktasına döndğü andaki hızları ilk hızlarına eşit olmalıdır. Bu programın bir çıktısı Tablo 1'de verilmiştir. Bu tabloyu inceleyerek yukarıda anlatılan özelliklerin sağlanıp sağlanmadığını kontrol ediniz.

7.b Bölüm (2)'de yazdığınız dört adımlı Runge-Kutta metodunu uygulayan ADD52 adlı programı ADD53 adı altında denklem (3.83)'de verilen kütle çekim problemini çözmek için yeniden düzenleyiniz. Bu düzenlemeyi yaparken

Tablo 3.5: ADD6 programının örnek bir çıktısı

G= 1.0,	CT= 2000.				
T	X	Y	Vx	Vy	DX
0.000E+00	1.000E+00	0.000E+00	0.000E+00	5.000E-01	2.714E-03
2.714E-01	9.629E-01	1.340E-01	-2.757E-01	4.809E-01	2.714E-03
5.428E-01	8.478E-01	2.568E-01	-5.798E-01	4.141E-01	2.714E-03
8.142E-01	6.415E-01	3.507E-01	-9.595E-01	2.548E-01	2.714E-03
1.086E+00	3.102E-01	3.698E-01	-1.532E+00	-2.147E-01	2.714E-03
1.357E+00	-1.430E-01	1.279E-04	-4.287E-03	-3.498E+00	2.714E-03
1.628E+00	3.090E-01	-3.709E-01	1.531E+00	-2.199E-01	2.714E-03
1.900E+00	6.404E-01	-3.530E-01	9.604E-01	2.514E-01	2.714E-03
2.171E+00	8.471E-01	-2.598E-01	5.815E-01	4.119E-01	2.714E-03
2.443E+00	9.627E-01	-1.375E-01	2.777E-01	4.797E-01	2.714E-03
2.714E+00	1.000E+00	-3.679E-03	2.312E-03	4.998E-01	2.714E-03

ADD6 programından da faydalanabilirsiniz. ADD6 ve ADD53 programlarının sonuçlarını karşılaştırınız. Hangisi daha hassas ve hangisi DEQ alt programını en az sayıda çağırmaktadır?

3.11 Labaratuar Saati-III.b

1. Basit Harmonik Salınıcı:

Daha önce göz önüne alınan basit harmonik salınıcı denklemini yeniden göz önüne alalım. Salınıcının kütlesi m ve yay sabiti k olmak üzere bu sistemin sağladığı diferansiyel denklemi

$$\frac{d^2x}{dt^2} + w^2x(t) = 0, \qquad x(0) = x_0, \qquad \frac{dx}{dt}|_{t=0} = v(0) = v_0 \tag{3.94}$$

ile verilir. Burada $w^2 = k/m$ olup sistemin doğal salınım frekansını temsil eder. Bu denklemde doğal zaman birimi sistemin doğal frekansının tersi $1/w$ olarak alınabilir. Zamanı yeniden ölçeklendirmek için $t' = tw$ gibi birimsiz bir değişken tanımlayalım. Bu yeni değişken kullanılarak (3.84)'de verilen diferansiyel denklemi yeniden yazılırsa

$$\frac{d^2x}{dt'^2} + x(t') = 0, \qquad x(0) = x_0, \qquad \frac{dx}{dt'}|_{t'=0} = v(0) = v_0 \tag{3.95}$$

elde edilir. Bu denklemde w parametresi tamamen elenmiş ve zaman doğal frekansın tersi cinsinden ölçülmektedir.

1.a (3.85)'de verilen denklemi iki tane birinci mertebe denklem sistemi şeklinde yazarak, $y_1 = x$ ve $y_2 = \frac{dx}{dt'}$ olmak üzere

$$\frac{dy_1}{dt'} = y_2, \qquad \frac{dy_2}{dt'} = -y_1 \tag{3.96}$$

denklemlerini elde ediniz.

1.b ADD8 programını BHS (**B**asit **H**armonik **S**alınıcı) adı altında başka bir dosyaya kopyalayarak bu programı yukarıda verilen basit harmonik salınıcı denklemini çözecek şekilde yeniden düzenleyiniz. Çözülecek denklem sayısının bu

durumda iki tane olduğunu ve başlangıç şartlarının yeni duruma uygun şekilde verilmesi gerektiğini unutmayınız. Programda değişmesi gereken en önemli bölüm DEQ alt programıdır. Ayrıca BHS programında elde edilen çözümleri ve zamanı adı kullanıcı tarafından belirlenen bir dosyaya yazacak şekilde gerekli değişiklikleri yapınız.

1.c BHS programında $x(0) = 1$ ve $v(0) = -1$ başlangıç şartlarını kullanınız. Bu başlangıç şartları diğer değişkenler cinsinden Y(1)=1 ve Y(2)=-1 değerlerine karşılık gelir. L=500, P=0.1 değerlerini kullanarak BHS programını çalıştırınız. Elde ettiğiniz veri dosyasını kullanarak $x-t$, $v-t$ ve $v-x$ grafiklerini çiziniz.

2. Sürtünmeli Basit Harmonik Salınıcı:

1.'de göz önüne alınan basit harmonik salınıcının hız ile doğru orantılı ($\gamma v(t)$) bir sürtünme kuvvetine maruz kalması durumunda (3.84)'de verilen diferansiyel denklemi

$$\frac{d^2x}{dt^2} + b\frac{dx}{dt} + w^2x(t) = 0, \qquad x(0) = x_0, \qquad \frac{dx}{dt}|_{t=0} = v(0) = v_0 \tag{3.97}$$

denklemine dönüşür. Burada $b = \gamma/m$ olup γ sürtünme kuvvetinin orantı sabitidir.

2.a b'nin biriminin (1/zaman) olması gerektiğini gösteriniz.

2.b Son denklemde zaman w^{-1} veya b^{-1} cinsinden ölçülebilir, yani bu problemin iki tane doğal zaman ölçeği bulunmaktadır. Bunlardan herhangi biri kullanılabilir. Örneğin zamanı w^{-1} cinsinden ölçmek için denklem (3.87) w^2 ile bölünürse

$$\frac{d^2x}{d(wt)^2} + \frac{b}{w}\frac{dx}{d(wt)} + x(t) = 0 \tag{3.98}$$

denklemi elde edilir. Birimsiz $t' = wt$ değişkeni kullanılırsa son denklem

$$\frac{d^2x}{dt'^2} + a\frac{dx}{dt'} + x(t') = 0 \tag{3.99}$$

şekline dönüşür. Bu denklemde zaman w^{-1} biriminde ölçülmektedir. Ayrıca problemde iki zaman biriminin oranından ibaret sadece birimsiz $a = b/w$ parametresi kalmıştır.

2.c BHS programını, BHS2 adı altında denklem (3.89)'da verilen diferansiyel denklemini çözecek şekilde düzenleyiniz.

2.d Denklem (3.89)'u $x(0) = 1$ ve $v(0) = -1$ başlangıç şartlarını BHS2 programında kullanarak çözünüz. L=500, P=0.1 alabilirsiniz. Çözümü a parametresinin a =0.05, 0.1, 0.2, 0.5, 1.0 ve 2.0 değerleri için ayrı ayrı bulunuz. a'nın her değeri için $x-t$ ve $v-t$ grafiklerini beraber, $v-x$ grafiğini ayrı çiziniz. Bu problemde a parametresinin rolü nedir? a'nın yukarıda alınan değerleri fiziksel olarak hangi durumlara karşılık gelmektedir? (Diferansiyel denkleminin analitik çözümünü kullanarak bu sorular daha kolay cevap bulabilirsiniz).

3. Sürümlü Basit Harmonik Salınıcı:

1.'de göz önüne alınan basit harmonik salınıcının bir $F_0 f(t)$ kuvveti ile sürülmesi halinde (3.84)'deki diferansiyel denklemi yerine

$$\frac{d^2x}{dt^2} + w^2x(t) = \frac{F_0}{m}f(t), \qquad x(0) = x_0, \qquad \frac{dx}{dt}|_{t=0} = v(0) = v_0 \tag{3.100}$$

denklemi elde edilir. Bu denklemde de zaman w^{-1} cinsinden ölçülebilir. Denklemin tamamı w^2 ile bölünüp zaman $t' = wt$ cinsinden ölçülürse bu denklem

$$\frac{d^2x}{dt'^2} + x(t') = cf(t'), \tag{3.101}$$

şekline dönüşür. Burada $c = \frac{F_0}{mw^2}$ ile verilir.

3.a c'nin biriminin uzunluk olması gerektiğini gösteriniz. Bu durumda denklem (3.91)'deki x değişkenide c ile yeniden ölçeklenerek birimsiz hale getirilebilir. Böylece bütün denklem birimsiz duruma gelir. $x' = x/c$ olmak üzere denklem (3.91)

$$\frac{d^2x'}{dt'^2} + x'(t') = f(t'), \tag{3.102}$$

denklemine dönüşür ve artık uzunluklar c cinsinden ölçülmektedir.

3.b BHS2 programını BHS3 adı altında sürümlü BHS'yı çözmek için yeniden düzenleyiniz. Sürücü kuvvetini $f(t) = \cos(w_s t)$ olarak alınız. Bu denklem birimsiz t' cinsinden, $g = w_s/w$ olmak üzere $f(t') = \cos(gt')$ şeklinde yazılabilir. Böylece çözülmesi gereken denklem (3.92)

$$\frac{d^2x}{dt'^2} + x(t') = \cos(gt') \tag{3.103}$$

şekline dönüşür. Dikkat edilirse bu son denklemde sadece g parametresi vardır.

3.c BHS3 programını kullanarak denklem (3.93)'ü g =0.5, 0.9, 1.0 ve 1.2 değerleri için çözünüz (L=2000, P=0.1 alınız). Başlangıç şartı olarak $x(0) = 1$ ve $v(0) = -1$ kullanınız. Bütün g değerleri için $x-t$, $v-t$ ve $v-x$ grafiklerini ayrı ayrı çiziniz. Bu problemde g parametresi neyi temsil etmektedir? $g = 1$ için çözüm nasıl davranmaktadır? $g \neq 1$ durumu için $v-x$ grafiği şimdiye kadar elde ettiklerinizden farklı mı? Hangi yönde?

4. Sürümlü ve Sürtünmeli Basit Harmonik Salınıcı:

3.'de incelediğimiz sürümlü basit harmonik salınıcıya sürtünme kuvveti de eklenirse

$$\frac{d^2x}{dt^2} + b\frac{dx}{dt} + w^2x(t) = \frac{F_0}{m}f(t), \qquad x(0) = x_0, \qquad \frac{dx}{dt}|_{t=0} = v(0) = v_0 \tag{3.104}$$

diferansiyel denklemi elde edilir.

4.a $f(t) = \cos(w_s t)$ alarak bu denklemi uygun şekilde yeniden ölçeklendiriniz. Parametre sayısını en aza indirmeye ve değişkenleri birimsiz hale getirmeye çalışınız. Denklemin en sonunda, $t' = wt$, $c = F_0/(mw^2)$, $a = b/w$, $g = w_s/w$ ve $x' = x/c$ olmak üzere

$$\frac{d^2x'}{dt'^2} + a\frac{dx'}{dt'} + x(t') = \cos(gt') \tag{3.105}$$

indirgenebileceğini gösteriniz. Bu son denklem birimsizdir ve sadece iki parametreye (a ve g) bağlıdır.

4.b BHS3 denklemini 4.a'da elde ettiğiniz denklemi çözecek şekilde BHS4 adı altında yeniden düzenleyiniz. Daha önce kullanılan başlangıç şartlarını ve parametreleriniz için uygun değerler kullanarak denklemi çözünüz. Elde ettiğiniz çözümü kullanarak $x-t$, $v-t$ ve $v-x$ grafiklerini kullandığınız bütün parametre değerleri için çiziniz.

3.12 ALIŞTIRMA SORULARI

Bölüm 4

RASGELE SAYILAR ve MONTE CARLO YÖNTEMLERİNE GİRİŞ

Bu bölümde rasgele sayılar ve bilgisayarda rasgele sayı üretimi basit olarak görülecektir. Rasgele sayı üreteçlerinin bazı test yöntemleri ve rasgele sayıların kullanılmasını gerektiren basit örnekler verilecektir. Bu bölüm Monte Carlo yöntemlerine bir giriş niteliğindedir.

4.1 Rasgele Sayılar

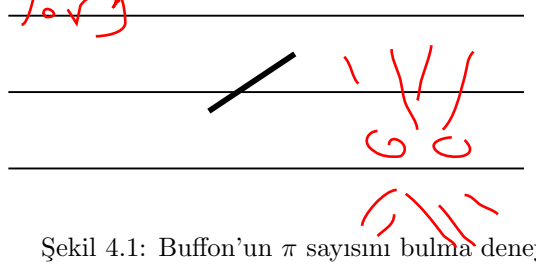
Çoğu kişi rasgele sayıları şans oyunları ile haklı olarak bağdaştırır. Fakat bilimsel problemlerin çözümünde artık çok sık ve yoğun olarak rasgele sayı kullanma temeline dayanan yöntemler kullanılmaktadır. Bilimsel problemlerin rasgele sayılar kullanılarak çözüldüğü yöntemler çoğunlukla Monte Carlo yöntemleri altında genel bir adla anılırlar. Bunların hepsi detayda farklı olsada rasgele sayıların bilgisayarda üretimi temeline dayanan yöntemlerdir.

Rasgele sayıların problem çözmek için bilinen en eski kullanımı Buffon[2] tarafından π sayısının değerini bulmak üzere yapılan deneydir. Rasgele sayıların bilimsel problemlerin çözümünde ilk ciddi kullanımı ise "Manhattan Projesi" olarak bilinen atom bombasının yapımında olmuştur. Burada ilk defa Enrico Fermi ve von Neumann tarafından nötron taşınım probleminde saçılma yönlerini belirlemek üzere rasgele sayılar kullanılmıştır. Rasgele sayıların kullanımına dayanan ve Monte Carlo Yöntemleri genel adıyla bilinen yöntemlere bu isim Fermi tarafından ünlü şans oyunları merkezi Monte Carlo'ya atfen verilmiştir.

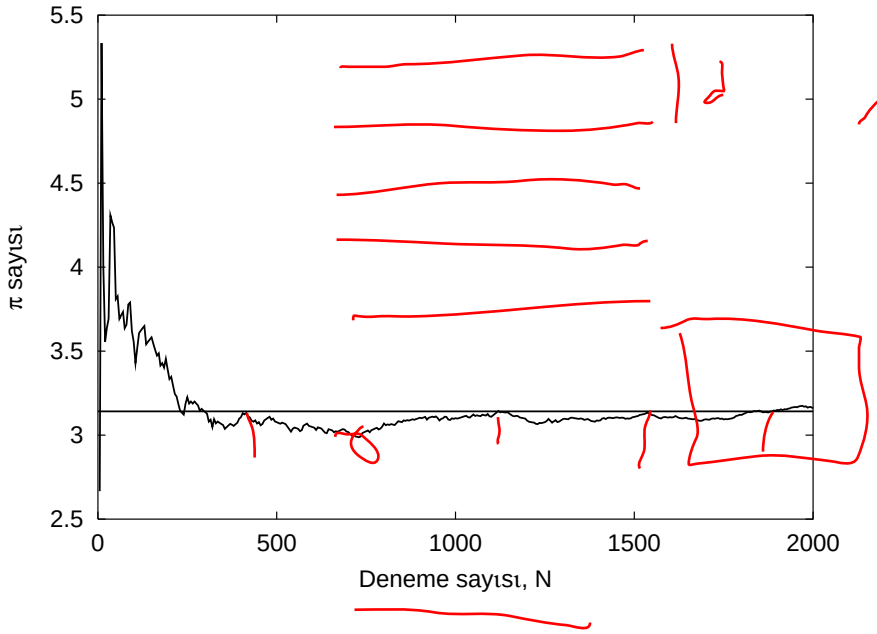
Rasgele sayıların kullanımı ile ilgili bilinen ilk örnek π sayısının değerinin belirlenmesi ile amacı ile Buffon tarafından yapılan deneydir. Buffon π sayısını belirlemek üzere birbirlerine uzaklıkları d olan paralel doğrular çizmiş ve ℓ uzunluklu bir düz bir tel veya iğneyi rasgele bu çizgiler üzerine atarak bir deney yapmıştır. Şekil 4.1'de gösterilen bu düzenekte ℓ uzunluklu telin ($\ell < d$) paralel doğrulardan birini kesme olasılığı P

$$P = \frac{2\ell}{\pi d} \qquad (4.1)$$

ile verilir. Bu deney N_0 defa tekrarlandığında tel N defa paralel çizgilere çarpıyorsa, P'nin deneysel değeri N/N_0 olacaktır. Bu değer yukarıdaki denklemde kullanılırsa, d ve ℓ bilindiğinden, π sayısının yaklaşık bir değeri bulunmuş



Şekil 4.1: Buffon'un π sayısını bulma deneyi.



Şekil 4.2: Buffon'un π sayısını bulma deneyinin rasgele sayılar üreterek elde edilen sonucu. Düz çizgi ile π sayısının değeri gösterilmiştir.

olur. Bu deneyin simülasyonu bilgisayarda üretilen rasgele sayılar kullanılarak yapılabilir. Böyle bir deneyin sonucu Şekil 4.2'de gösterilmiştir. Deneyle bulunan değer yapılan deney sayısının bir fonksiyonu olarak şekilde gösterilmiştir. Görüldüğü gibi π sayısının hesaplanan değeri bilinen değeri etrafında salınım yapmaktadır. Bu hesaplamanın detayları daha sonra görülecektir.

4.2 Rasgele Sayıların Üretilmesi

Rasgele sayılar bilgisayarlarla

$$I_{n+1} = aI_n + c \quad mod(m) \tag{4.2}$$

denklemi kullanılarak üretilir. Burada *a çarpan* ve *c artırım* olarak adlandırılan pozitif sabitlerdir. Sonuç m modunda modüler aritmetik yapılarak bulunur. Yukarıda verilen iterasyonun başlayabilmesi için I_0'ın kullanıcı tarafından verilmesi gerekir. I_0 *tohum* (seed) olarak bilinir. Bu yöntemle en çok m tane rasgele sayı üretilebilir ve I_0'ın verilmesi durumunda ondan sonra gelecek bütün sayılar baştan bellidir!. Bu durum bir çelişki gibi görünsede burada önemli olan belirlenen aralıkta bulunan sayıların düzgün bir şekilde eşit olasılıkla seçilmesidir.

Bu nedenle Bilgisarlarla üretilen rasgele sayıların "ne kadar rasgele" olduklarından çok, belirli bir sayı kümesinden "ne kadar rasgele" şekilde bir sayı seçtikleri daha önemlidir. Aynı tohum numarası kullanıldığında aynı sayılar kümesi üretilecektir. Bu durum bazen oldukça avantajlıdır.

Yukarıdaki denklemde elde edilen I_{n+1} rasgele sayıları çoğunlukla m ile bölünerek normalize edilir ve bu nedenle rasgele sayı üreteçlerinin hemen hepsi [0,1] aralığında sayı üretir. (Kullanılan yönteme göre bazı durumlarda üretilen sayılara 0 ve/ya 1 dahil olmayabilir).

Denklem (4.2)'den faydalanarak rasgele sayı üretiminde önemli olan a, c ve m'nin seçimidir. m üretilecek rasgele sayıların periyotu olduğundan en çok m tane rasgele sayı üretilebilir. Bundan sonra seri kendi kendini tekrar etmeye başlar. Bu nedenle m mümkün olduğunca büyük olmalıdır. m çoğunlukla kullanılan makinede tam olarak temsil edilebilecek en büyük tamsayıdır. Fakat bu sayı yeterince büyük değildir ve aşağıda görüleceği gibi bu sayıyı daha da artırmak mümkündür. Örneğin 32-bitlik PC'lerde $m \approx 2^{32}$ olur. Diğer tarafta a ve c'nin seçimi çok önemlidir. Çoğu zaman $c = 0$ seçerek

$$I_{n+1} = aI_n \quad mod(m) \tag{4.3}$$

kullanılmakta ve bu yolla üretilen rasgele sayılar $c \neq 0$ durumu için üretilenler kadar iyidir.

Örneğin $a = 5$, $m = 13$ ve $c = 1$ seçerek rasgele sayılar üretelim. Tohum sayısı olarak $I_0 = 4$ seçelim. Bu seçenekler kullanıldığında aşağıdaki rasgele sayı serisi üretilecektir.

n	1	2	3	4	5
I_n	8	2	11	4	8

Başka bir tohum numarası örneğin $I_0 = 5$ kullandığımızda ise aşağıdaki rasgele sayı serisini elde ederiz.

n	1	2	3	4
I_n	0	1	6	5

Yukarıdaki örneklerde görüldüğü gibi rasgele sayı üretecinin parametrelerini rasgele değiştirmek üretecin kalitesini artırmayacaktır.

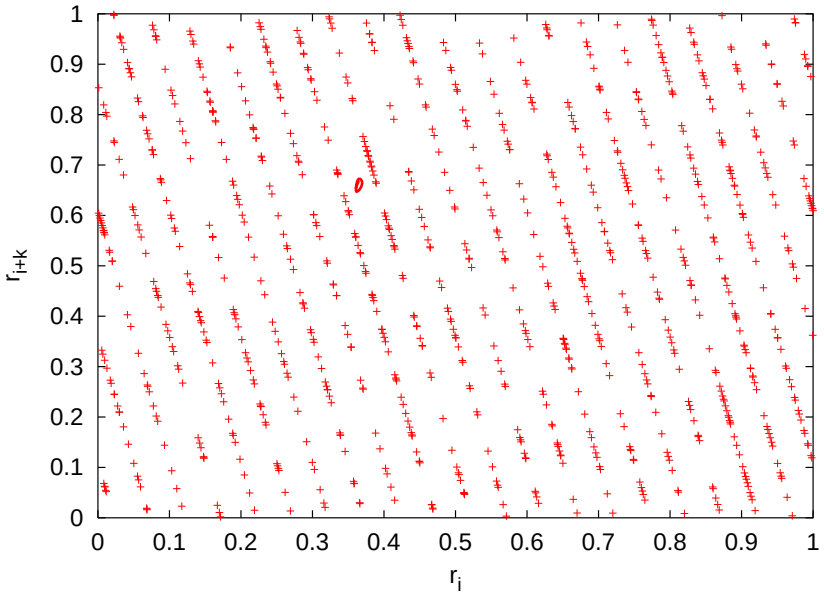
Park ve Miller tarafından önerilen "minimal standarda" göre a, c ve m için en uygun seçenekler şöyledir:

$$a = 7^5 = 16807, c = 0, m = 2^{31} - 1 = 2147483647$$

Bu değerlerlerle denklem (4.3) kullanıldığında bir sorun ortaya çıkar: aI_n çarpımı bazı durumlarda makinede tam olarak temsil edilebilecek en büyük tam sayıdan daha büyük olacaktır. Bunun üstesinden gelmenin yolu Schrage algoritmasını kullanmaktır. Schrage'in algoritması 32-bitlik iki tam sayının (32-bitlik sabit bir mod sayısına göre) çarpımının sonucunun, 32-bitin üzerindeki sayıların kullanılmasını gerektirmeyecek şekilde bulunmasını sağlar. Algortimanın temeli m sayısının uygun bir şekilde çarpanlarına ayrılmasına dayanır.

Rasgele üretilen sayılarda ortaya çıkan sorunlardan bir tanesi üretilen I_i nci rasgele sayı ile I_{i+k} ncı rasgele sayı arasında belirli bir korelasyonun bulunmasıdır.

Örneğin burada kullanılan RND alt programının böyle bir sorunu bulunmaktadır ve bu rasgele sayı üreteci kullanılarak elde edilen r_i nci rasgele sayıya



Şekil 4.3: RND alt programı ile üretilen r_i nci rasgele sayı ile r_{i+k} nci rasgele sayı arasındaki korelasyona bir örnek.

karşılık çizilen r_{i+k} nci rasgele sayının grafiği Şekil 4.3'te gösterilmiştir. k değerinin bulunması ödev olarak okuyucuya bırakılmıştır. k'nın küçük olduğu durumlara düşük mertebeli korelasyonlar denir. Düşük mertebeli korelasyonları gidermek için, üretilen rasgele sayıların bir kısmı bir yerde toplanmakta ve bu sayılar yeniden karılarak rasgele bir sırada kullanıcıya sunulmaktadır. Bu yöntemde rasgele sayıyı üretmek için kullanılan dizide örneğin i nci sırada üretilen bir rasgele sayı yeniden karma işleminden sonra ortalama $(i + 32)$nci sırada çıktı olarak verilir. Bu yöntemi kullanarak rasgele sayı üreten bir alt fonksiyon RAN1 adı altında [3] verilmiş ve RAS1 programına alınmıştır. Ayrıca yine aynı kaynaktan alınan RAN2, RAN3 ve RAN4 rasgele sayı üreten alt fonksiyonlar RAS1 programına dahil edilmiştir.

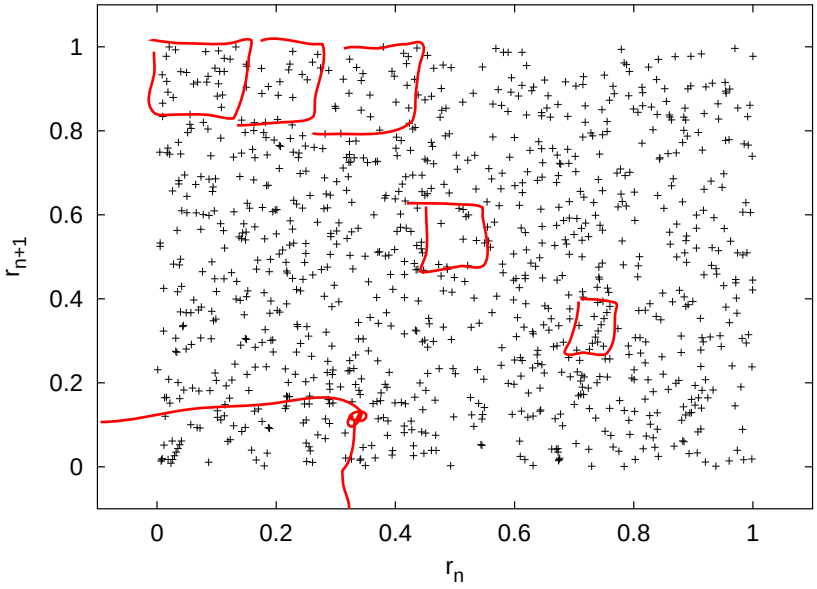
4.3 Rasgele Sayıların Test Edilmesi

Rasgele sayıları test etmek için çeşitli yöntemler vardır. Fakat bunların çoğu kesin bir test olmayıp sadece rasgele sayı üreteci konusunda bir fikir verebilirler. Aşağıda bunların bir kısmı verilmiştir.

Test 1.) Bir rasgele sayı üreteci sonuçları önceden bilinen bir problemi çözmek için kullanılarak test edilebilir. Fakat bu kesin bir test olmayıp rasgele sayı üretecinin sadece incelenen problem için yeterli derecede rasgele sayı ürettiğini gösterir.

Test 2.) İki rasgele sayı üreteci aynı uygulamada kullanılarak sonuçlar karşılaştırılabilir. İkiside aynı sonucu veriyorsa her ikiside eşit derecede iyi veya kötü üreteçlerdir. Eğer sonuçlar farklı çıkıyorsa buradan çıkarılacak sonuç iki üretecin benzer şekilde rasgele sayı üretmediğidir.

Test 3.) Dağılım Testi



Şekil 4.4: Rasgele sayıların iki boyutta dağılımı

Peş peşe üretilen rasgele sayıların iki veya üç boyutta grafikleri çizilerek bunların dağılımına bakılabilir. Örneğin peş peşe üretilen normalize edilmiş r_n ve r_{n+1} rasgele sayılarının kartezyen koordinatlarda bir nokta olarak temsil edilmesi ile iki boyutlu elde edilen grafikleri incelenerek kullanılan rasgele sayı üreteci hakkında bir fikir edinilebilir. Bu yöntemle yukarıda sözü edilen RND alt fonksiyonu kullanılarak elde edilen grafik Şekil 4.4'de gösterilmiştir. Bu grafikte noktaların düzlem üzerinde düzgün dağılması gerekir. Noktaların öbeklenmesi veya aşırı boşlukların oluşması düzgün rasgele sayı üretilmediği anlamına gelir. Bu test rasgelelik üzerine nihai bir test değildir.

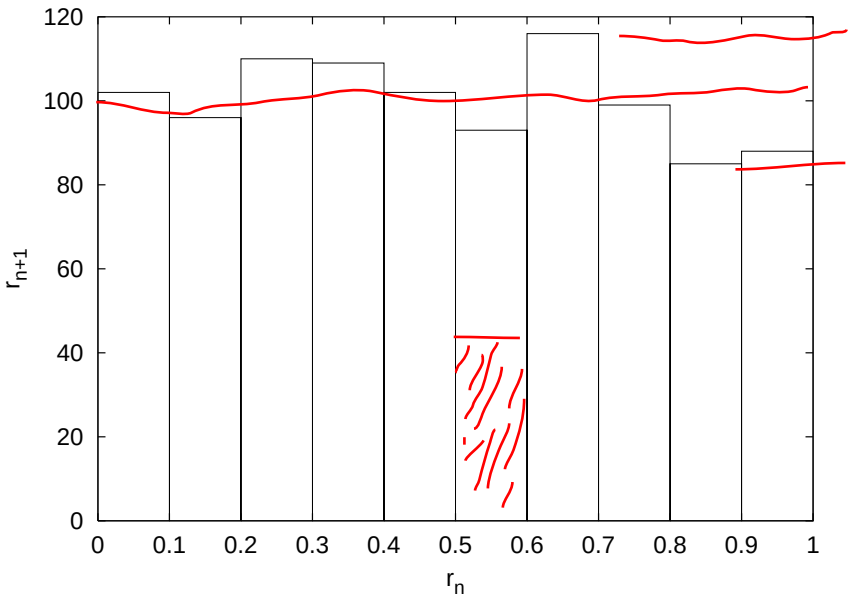
Test 4.) Başka Bir Dağılım Testi

[0,1] aralığı belli sayıda eşit aralığa bölünerek her aralığa düşen rasgele sayı bulunup, bu sayı aralık sayısının bir fonksiyonu olarak çizilebilir. Toplam aralık sayısı M ve üretilen rasgele sayı sayısı N ise, her aralığa ortalama (N/M) sayı düşmesi gerekir. Her aralığa düşen rasgele sayısı bu değer civarında olmalıdır. Bazı aralıklarda bu değerden çok büyük veya çok küçük değerlerin elde edilmesi, anılan aralıkta gereğinden fazla veya az sayıda rasgele sayı üretildiği anlamına gelir. Bu istenen bir durum değildir. Bu yöntemle RND alt programı kullanılarak elde edilen grafik Şekil 4.5'de gösterilmiştir.

RAS2 programını ağ komşuluğundan alarak programda kullanılan bütün rasgele üreteçleri için Şekil 4.5'dekine benzer bir grafik elde ediniz. Üretilen rasgele sayı miktarını ve/veya aralık sayısını değiştirerek elde ettiğiniz sonuçları karşılaştırınız.

Test 5.) χ^2 Testi

(4) nolu testte anlatıldığı gibi eğer N tane rasgele sayı üretiliyor ve [0,1] aralığı M tane aralığa bölünüyorsa, her aralığa düşmesi gereken rasgele sayı



Şekil 4.5: Rasgele sayıların 10 eşit aralıktaki dağılımları

miktarı $s_i = (N/M)$ olmalıdır. Her i aralığına düşen gerçek sayı Y_i ise χ^2 değeri

$$\chi^2 = \sum_{i=1}^{M} \frac{(Y_i - s_i)^2}{s_i} \tag{4.4}$$

olarak tanımlanır. Eğer M yeterince büyük ise (> 30) χ^2'in ortalama değeri yaklaşık M olmalıdır. Ortalama değer değişik tohum sayıları ile başlayarak elde edilen χ^2 değerlerinin ortalaması alınarak yapılabilir.

4.4 Monte Carlo Yöntemi ile İntegral Alma

Monte Carlo yöntemi ile integral almayı göz önüne almadan önce bilinen standart yollarla integral almayı tekrar hatırlamakta fayda vardır. Örneğin bir değişkene bağlı bir fonksiyonun integralini almak fonksiyon ile x-ekseni arasında kalan alanı bulmak demektir. x-ekseninin eşit Δx aralıklı N noktasına karşılık gelen fonksiyon değerleri bulunarak basit bir toplama ile integral hesabı yapılabilir. İntegralin alt ve üst sınırlarının sırası ile a ve b olduğu bir boyutlu durum için alan

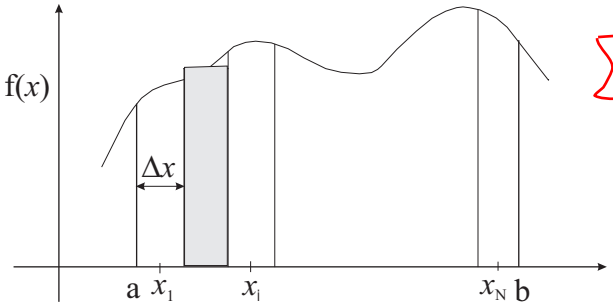
$$A = \sum_{i=1}^{N} f(x_i)\Delta x \tag{4.5}$$

olur. x_i'ler aralıkların orta noktaları olarak alınabilir: $x_i = a + (i - 1/2)\Delta x, \quad i = 1, 2, \ldots N$. Noktalar arasındaki eşit aralıklar

$$\Delta x = (b - a)/N$$

olacağından alan integrali

$$A = (b - a)\left\{\frac{\sum_{i=1}^{N} f(x_i)}{N}\right\} \tag{4.6}$$



Şekil 4.6: $f(x)$'in integrali x-ekseni ile f arasında kalan alanı verir.

şeklinde de yazılabilir. Dikkat edilirse sağ taraftaki terim f fonksiyonunun *ortalama* değerini temsil etmektedir. Bu denklemde her ne kadar fonksiyonunun eit aralıklı N tane noktadaki değeri hesaplanıyorsa da, ortalama değeri hesaplamak için eşit aralıklı noktalar seçmek zorunluluğu yoktur. $[a, b]$ aralığından rasgele seçilm N tane nokta da kullanarak ortalama değer hesaplanabilir. Monte Carlo yöntemi ile integral almanın temeli burada yatmaktadır.

Genel olarak MC yöntemi ile düşük boyutlarda integral almanın sağladığı fazla bir avantaj yoktur. Bu yöntemin asıl gücü boyut arttıkça ortaya çıkmaktadır. Daha sonra yapılacaklara hazırlık için denklem (4.6)'ı biraz değişik bir biçimde yazılabilir. Alan iki boyutlu hacim gibi düşünülebilir ve bu durum $V^{(2)} = A$ olarak gösterilebilir. Benzer şekilde $(b - a)$ uzunluğu bir boyutlu hacim gibi düşünülebilir ve benzer şekilde bu durum $V^{(1)} = (b - a)$ olarak gösterilebilir. Yeni gösterim biçimi kullanılarak (4.6) denklemi tekrar yazılırsa

$$V^{(2)} = V^{(1)} < f > \tag{4.7}$$

elde edilir. Burada $< f >$ fonksiyonunun ortalama değerini temsil etmektedir.

İki boyutta integral hesaplamak için $z = f(x, y)$ fonksiyonu verilmiş olsun. Bu durumda xy düzleminde seçilen her (x_0, y_0) noktasına karşılık gelen bir $f(x_0, y_0)$ noktası vardır. Fonksiyon ile xy düzlemi arasında kalan hacim $V^{(3)}$

$$V^{(3)} = \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} f(x_i, y_j) \Delta x \Delta y \tag{4.8}$$

olarak yazılabilir. Burada x yönünde integral alınan bölge N_x eşit aralığa, benzer şekilde y yönü ise N_y eşit aralığa bölünmüştür. Böylece

$$\Delta x = \frac{(b_x - a_x)}{N_x} \qquad \Delta y = \frac{(b_y - a_y)}{N_y} \tag{4.9}$$

olmak üzere yukarıdaki denklem

$$V^{(3)} = (b_x - a_x)(b_y - a_y) \frac{\sum_{i=1}^{N_x} \sum_{j=1}^{N_y} f(x_i, y_j)}{N_x N_y} \tag{4.10}$$

olarak yazılabilir. Burada $(b_x - a_x)(b_y - a_y)$ terimi fonksiyonun tanımlandığı iki boyutlu bölgenin alanını, sağ taraftaki ikinci terim ise $N_x N_y$ tane noktada hesaplanan $f(x, y)$ fonksiyonunun ortalama değerini verir. Bu son durum sembolik olarak

$$V^{(3)} = V^{(2)} < f(x, y) > \tag{4.11}$$

şeklinde gösterilebilir.

Yukarıda bir ve iki boyutlu durum için gösterilenler M boyuta genelleştirilirse

$$V^{(M+1)} = (b_1-a_1)(b_2-a_2)\dots(b_m-a_m)\frac{\sum_{i_1=1}^{N_1}\sum_{i_2=1}^{N_2}\dots\sum_{i_M=1}^{N_M} f(x_{i_1},x_{i_2},\dots,x_{i_M})}{N_1 N_2 \dots N_M} \tag{4.12}$$

elde edilir ve sonuç sembolik olarak

$$V^{(M+1)} = V^{(M)} < f > \tag{4.13}$$

şeklinde yazılabilir.

ÖRNEK **4.1** Birim Çemberin Alanı

$x^2 + y^2 = R^2 = 1$ denklemi ile verilen birim çember içinde kalan bölgenin alanı rasgele sayılar kullanılarak bulunabilir. Alan $A = \pi R^2 = \pi$ olduğundan bu problemin sonucu aynı zamanda π sayısının yaklaşık bir değerini bulmak içinde kullanılabilir. Bu problem iki türlü çözülebilir.

a) Birinci durumda yukarıda anlatılan yöntem uygulanarak $y = f(x) = \sqrt{1-x^2}$ fonksiyonunun ortalama değeri bulunarak birim çemberin alanı bulunabilir. Bunun için [0,1] aralığından bir rasgele r sayısı seçilir. Bu sayı kullanılarak $y = f(r)$ hesaplanır. Bu işlem yeterince defa tekrarlanarak f'nin ortalama değerinden alan bulunmuş olur. Bu yöntemin algoritması aşağıda verilmiştir.

Algoritma 4.1 *Monte Carlo metodu ile integral alma:*

0. Yeterince büyük bir N sayısı seç; fonksiyonun toplam değerini ve sayaç indisini sıfırla, $ftop = 0$, $i = 0$
1. [0,1] aralığından bir r rasgele sayısı seç ve sayacı bir artır, $i = i + 1$
2. Fonksiyon değerini ve toplamı hesapla, $f = \sqrt{1-r^2}$, $ftop = ftop + f$
3. $i < N$ ise (1)'git, değilse sonucu yaz: $A = 4(1-0) < f >= 4\frac{ftop}{N}$

Bu algoritmayı kullanarak birim çemberin alanını bulan RASINT1.FOR programı Ek-E'de verilmiştir. Ayrıca bu bölümün sonundaki Labaratuar Saati-IV'e bakınız.

b) Diğer yöntemde ise rasgele sayıların düzgün dağılımından faydalanılarak işlem yapılabilir. Bu yöntemde [0,1] aralığından r_x ve r_y gibi iki rasgele sayı seçilir. Bu sayıların bir noktanın sırası ile x ve y yönündeki koordinatlarını temsil ettiğini kabul edelim. Eğer $r_x^2 + r_y^2 \le 1$ ise bu iki sayının temsil ettiği (r_x, r_y) noktası birim çember içindedir, değilse birim çemberin dışındadır. Rasgele sayıların düzgün olarak dağıldığı kabul edilirse, çember içinde kalan sayıların toplam sayıya oranı birim çemberin alanı ile orantılı olmalıdır. Bu nedenle bu deney N defa tekrarlanır ve N_0 tane rasgele sayı çifti $r_x^2 + r_y^2 \le 1$ eşitliğini sağlıyorsa birim dairenin alanı $A = 4\frac{N_0}{N}$ olacaktır. Bu yöntemin bir algoritması aşağıda verilmiştir.

Algoritma 4.2 *Monte Carlo metodu ile integral alma:*

0. Yeterince büyük bir N sayısı seç; birim çember içinde ve dışında kalan noktaların sayaç indisini sıfırla, $N_1 = 0$, $N_2 = 0$
1. [0,1] aralığından r_x ve r_y rasgele sayılarını seç,

2. $r_x^2 + r_y^2 \leq 1$ *ise* $N_1 = N_1 + 1$ *değilse* $N_2 = N_2 + 1$
3. $(N_1 + N_2) < N$ *ise (1)'git, değilse sonucu yaz:* $A = 4\frac{N_1}{N}$

Bu algoritmayı kullanarak birim çemberin alanını bulan RASINT2.FOR programı Ek-E'de verilmiştir. Ayrıca bu bölümün sonundaki Labaratuar Saati-IV'e bakınız.

4.5 Metropolis Algoritması

Metropolis algoritmasının 1953 yılında önerilmesinden[4] bu yana sadece algoritmanın asıl önerildiği problemler değil başka bir çok problemin çözümünde kullanılmıştır. Bunun nedeni algoritmanın basit ve kolay uygulanabilir olmasıdır.

Metropolis algoritması temel olarak termodinamik dengede olan sistemler için önerilmiştir. Termodinamik dengede olan bir sistemi göz önüne alalım. Bu sistemin girebileceği E_i $(i = 1, 2, \ldots N)$ N tane durum varsa, sistemin eş bölüşüm fonksiyonu[5]

$$Z = \sum_{i=1}^{N} e^{-\beta E_i} \tag{4.14}$$

ile verilir. Burada k_B Boltzmann sabiti ve T mutlak sıcaklık olmak üzere $\beta = 1/k_B T$'dir. Yukarıdaki toplam alınabilecek enerji değerleri üzerinden değil, girilebilecek bütün durumlar üzerinden olmalıdır. Eş bölüşüm fonksiyonunun bilinmesi durumunda sistem ile ilgili diğer termodinamik değişkenlerde bulunabilir. Örneğin sistemin ortalama enerjisi $< E >$

$$< E >= \frac{\sum_{i=1}^{N} E_i e^{-\beta E_i}}{\sum_{i=1}^{N} e^{-\beta E_i}} = \frac{\sum_{i=1}^{N} E_i e^{-\beta E_i}}{Z} \tag{4.15}$$

ile verilir. Benzer şekilde sistemin E_i enerjili bir durumda bulunma olasılığı $P_i = P(E_i)$

$$P(E_i) = \frac{e^{-\beta E_i}}{Z} \tag{4.16}$$

ile verilir.

4.6 Bir Boyutlu Ising Modeli

ÖRNEK **4.2** *Manyetik Alan İçindeki 1/2 Spinli bir Parçacık*

1/2 spinli, manyetik momenti μ *olan bir parçacık göz önüne alalım. Parçacık z-yönünde düzgün bir B manyetik alanı içinde bulunsun. Manyetik moment ancak* $\pm\mu$ *değerlerini alabilir. Bu nedenle sistemin girebileceği sadece iki durum vardır: Spin manyetik momnetinin B manyetik alanı ile aynı yönlü olduğu durum ile ters yönlü olduğu durum.*

Sistemin enerjisi moment manyetik alan ile aynı yönlü ise

$$E_1 = -\mu \cdot \mathbf{B} = -\mu B \tag{4.17}$$

ters yönlü ise

$$E_2 = -\mu \cdot \mathbf{B} = \mu B \tag{4.18}$$

olacaktır. Böylece eş bölüşüm fonksiyonu

$$Z = \sum_{i=1}^{2} e^{-\beta E_i} = e^{-E_1/k_BT} + e^{-E_2/k_BT} = 2\cosh(\frac{\mu B}{k_BT}) \tag{4.19}$$

olur. Bu sistemin E_1 ve E_2 enerjili durumlarda bulunma olasılıkları ile sitemin ortalama enerjisi sırası ile

$$\begin{aligned} P(E_1) &= \frac{1}{Z}e^{-E_1/k_BT} = \frac{1}{Z}e^{\mu B/k_BT} \\ P(E_2) &= \frac{1}{Z}e^{-E_2/k_BT} = \frac{1}{Z}e^{-\mu B/k_BT} \\ < E > &= \frac{1}{Z}\sum_{i=1}^{2} E_i e^{-\beta E_i} = \frac{1}{Z}[-\mu B e^{\mu B/k_BT} + \mu B e^{-\mu B/k_BT}] \\ &= -\mu B \tanh(\frac{\mu B}{k_BT}) \end{aligned} \tag{4.20}$$

olacaktır. Burada da görüldüğü gibi parçacığın manyetik momentinin manyetik alan yönüne doğru yönelme olasılığı daha yüksektir. Bu yönelim varolan iki olası durumdan daha düşük enerjili duruma karşılık gelir.

Şimdi Metropolis algoritmasını elde edelim. Sistemimizin başlangıçta bir i halinde bulunduğunu ve daha sonra bir j haline (durumuna) geçtiğini varsayalım. Bir önceki örnekte incelenen bir parçacıklı sistem göz önüne alınacak olursa, i parçacığın momentinin yukarı doğru olduğu, j ise parçacığın momentinin aşağı doğru olduğu durum olarak düşünülebilir. P_i sistemin i halinde bulunma olasılığı ve $W_{i\to j}$ i halinde bulunan sistemin birim zamanda j haline geçme oranı olmak üzere anılan geçiş olayının (birim zamanda) meydana gelme olasılığı

$$P_i W_{i\to j} \tag{4.21}$$

olarak yazılabilir. Termodinamik denge altında *i*'den *j*'ye geçme olayının eşit bir *j*'den *i*'ye geçme olayı ile dengelenmesi gerekir. ('detailed balance' veya 'law of mass action') Bu nedenle denge durumunda

$$P_i W_{i\to j} = P_j W_{j\to i} \tag{4.22}$$

olmalıdır. Denklem (4.16) kullanılarak olasılıkların oranı

$$\frac{W_{i\to j}}{W_{j\to i}} = \frac{P_j}{P_i} = e^{-(E_j - E_i)/k_BT} \tag{4.23}$$

olarak elde edilebilir. Dikkat edilirse bu denklemde eşbölüşüm fonksiyonu tamamen elenmiştir. Denklemin sağ tarafı ise eşbölüşüm fonksiyonuna göre çoğunlukla daha kolay hesaplanabilen bir büyüklüktür. Bu özelliklerinden dolayı metropolis algortiması hem kullanışlı hemde programlaması aşağıda görüleceği gibi basittir.

Burada önemli olan P_i olasılık dağılıma uyan bir durumlar (haller) kümesi elde etmektir. Metropolis algoritmasında önerilen yöntem $W_{i\to j}$'lerin aşağıdaki gibi seçilmesidir.

$$W_{i\to j} = \begin{cases} 1, & E_j < E_i \text{ ise} \\ e^{-(E_j - E_i)/k_BT}, & E_j > E_i \text{ ise} \end{cases} \tag{4.24}$$

Yukarıda verilen denkleme göre sistemin bir i halinden bir j haline geçme olayının seçimi şu şekilde yapılır:

a) Sistemin i durumundaki enerjisi E_i ve geçiş yapacağı j durumunda alacağı E_j enerjisi hesaplanır.

b) Eğer $E_j < E_i$ ise geçişe izin verilir ve sistem yeni j durumuna sokulur.

c) Eğer $E_j > E_i$ ise geçişe $e^{-(E_j-E_i)/k_BT}$ olasılığı ile izin verilir. Bu adımı pratikte gerçekleştirmek için [0,1] aralığından bir r rasgele sayısı seçilir. Eğer $r < e^{-(E_j-E_i)/k_BT}$ ise geçişe izin verilir ve sistem j durumuna sokulur, değilse sistem i halinde kalır.

Yukarıda verilen (a-c) adımları Metropolis algoritmasının temelini oluşturur. Bu algoritma sadece termodinamik sistemlere değil değişik bir çok sisteme uygulanmıştır. Bu gibi durumlarda sıcaklığın belirgin bir anlamı yoktur ve probleme sadece bir parametre olarak girer. Bu sistemlerde enerji yerine 'amaç fonksiyonu' (objective function) denilen ve gözönüne alınan sistemin özelliklerine göre minimuma gitmesi gereken bir fonksiyon kullanılır. Örneğin belirli sayıdaki şehri aynı şehri iki defa ziyaret etmemek şartı ile dolaşan gezici-satıcı (çerçi) probleminde sistemin amaç fonksiyonu ziyaret edilen şehirler arasındaki uzaklıkların toplamıdır. Problem bu uzaklığın en az olması için merkezlerin hangi sırada ziyaret edilmesi gerektiğini bulmaktır.

Başlangıçtaki sistem termodinamik dengede değilse bile dengeye ulaşma yönünde yapılan hareketler daha sık olarak kabul edileceğinden, sistem dengeye ulaşacaktır. Bu nedenle Metropolis algoritması dengeden uzak sistemler içinde kullanılarak bunların dengeye ulaşma süreci ve denge durumunda ulaşacakları son durum incelenebilir. Örneğin düşük sıcaklıklarda bir sistemin gidebileceği durum bilinmek isteniyorsa, bir yüksek sıcaklık durumundan başlanarak sıcaklık adım adım azaltılarak sistemin gideceği son durum bulunabilir. Burada her adımda seçilen sıcaklık değerinde sistemin dengeye ulaşması sağlanmalı, daha sonra sıcaklık azaltılmalıdır. Sıcaklıkta yapılan değişiklikler çok büyük olmamalıdır.

Son olarak herhangi bir sistem için kullanılabilecek Metropolis Algoritması aşağıda verilmiştir:

Algoritma 4.3 *Metropolis Algoritması:*

0. Sistem için bir başlangıç durumu seç. Bu bilinen veya rasgele seçilen bir durum olabilir. Monte Carlo adım sayısı için yeterince büyük bir N sayısı seç ve sayacı sıfırla, $i = 0$.

1. Sistemin enerjisini hesapla, E_{ilk} ve sayacı artır, i=i+1

2. Sistemin herhangi bir parametresini değiştirerek (örneğin rasgele seçilen bir parçacığın rasgele seçilen başka bir yere taşınması gibi) son enerjiyi hesapla, E_{son}

3. a) Eğer $E_{son} < E_{ilk}$ ise yeni durumu kabul et.

b) Eğer $E_{son} > E_{ilk}$ ise [0,1] aralığında bir r rasgele sayısı seç ve $r < e^{-(E_{son}-E_{ilk})/k_BT}$ ise yeni durumu kabul et, değilse sistem ilk durumunda kalsın.

4. Gerekli termodinamik büyüklükleri hesapla.

5. $i < N$ ise (1)'git, değilse sonuçları yazdır.

ÖRNEK **4.3** *Bir Boyutlu Ising Modeli:*

1/2 spinli, her birinin manyetik momenti $\mu = \pm 1$ olan yerlerinde sabit N tane parçacık göz önüne alalım. En yakın komşu etkileşme enerjileri, J etkileşme şiddeti olmak üzere $-J\mu_i\mu_{i+1}$ olacaktır. J pozitif veya negatif bir

sabittir. Periyodik sınır şartları ($\mu_1 = \mu_{N+1}$) kullanarak rasgele bir yönelimle başlayan bir sistemin son durumunu J'nin pozitif ve negatif değerleri için bulunuz. Ayrıca sistemin son durumunun J/k_BT oranına nasıl bağlı olduğunu araştırınız. Sonuçlarınızı yukarı (aşağı) yönelmiş spinlerin sayısı n_1'i (n_2) Monte Carlo adım sayısının bir fonksiyonu olarak çizerek gösteriniz.

Bu problemin çözümü için kullanılacak algoritma aşağıdaki gibi olabilir:

Algoritma 4.4 *Bir Byutlu Ising Modeli için Metropolis Algoritması:*

0. Rasgele yönelimli N tane spin seç. Bunu yapmak için [0,1] aralığından rasgele bir r sayısı seç, eğer $r < 0.5$ ise spin yukarı ($\mu = +1$), $r > 0.5$ ise spin aşağı ($\mu = -1$) durumunu seç. Bunu bütün parçacıklar için tekrarla. Sayacı sıfırla, $is = 0$ ve toplam Monte Carlo adım sayısını (Nmc) belirle.

1. Sistemin toplam enerjisini hesapla, $E_{ilk} = -J\sum_{i=1}^{N} \mu_i\mu_{i+1}$ ve sayacı artır, $is = is + 1$.

2. Rasgele bir parçacık seç ve bu parçacığın spin yönelimini ters çevir. Yeni durum için sistemin toplam enerjisini hesapla.

3. a) $E_{son} < E_{ilk}$ ise yeni durumu kabul et.

b) $E_{son} > E_{ilk}$ ise [0,1] aralığından bir r rasgele sayısı seç. $r < e^{-(E_{son}-E_{ilk})/k_BT}$ ise yeni durumu kabul et, değilse sistemin eski durumuna dön.

BU algoritmanın uygulandığı RASMET1.FOR programı Ek-E'de verilmiştir.

4.7 Labaratuar Saati-IV

Bu bölüm ile ilgili programlar Ek-E'de verilmiştir.

1. RAS1.FOR programında kullanılan rasgele sayı üreteçlerinden biri RND adlı bir alt programdır. Bu alt program incelendiğinde $a = 1027$, $c = 0$ ve $m = 1048576$ olduğu görülecektir. Dikkat edilirse $a * m = 1076887552$ çarpımı 32-bitlik bir makinede tam olarak temsil edilebilecek bir tam sayı olan $2^{31} = 2147483648$'nin ancak yarısı kadardır. Burada rasgele sayı üretecinin periyotu m oldukça düşüktür.

Rasgele sayı üreteçlerinin periyotunu artırmanın bir yolu Schrage algoritmasını kullanmaktır. "Numerical Recipes[3]"de verildiği için buraya alınmayan RAN0 rasgele sayı üreteci Schrage algoritmasını kullanarak Park ve Miller tarafından önerilen "minimal standart"ı kullanan bir alt programdır. Bu alt program size verilen RAS1 programı içinde çağrılmaktadır. RAN0'ın periyotu $2^{31} - 2 \approx 2.1 \times 10^9$ kadardır.

Ağ komşuluğundan RAS1 ve RAS2 programlarını alınız. Her iki programda da RND, RAN0, RAN1, RAN2, RAN3 ve RAN4 olmak üzere altı tane rasgele sayı üreten alt fonksiyon verilmiştir. Kullanıcı bunlardan bir tanesini seçebilir.

1.a RAS1 programını çalıştırınız. Bu alt programların hepsini ayrı ayrı seçerek her rasgele sayı üreteci için Şekil 4.4'dekine benzer grafikler çiziniz.

1.b RAS2 programını çalıştırınız. Rasgele sayı üreten alt programların hepsini ayrı ayrı seçerek her rasgele sayı üreteci için Şekil 4.5'dekine benzer grafikler çiziniz.

Yukarıda elde edilen grafikelerden faydalanarak hangi üretecin daha iyi olduğunu söyleyebilir misiniz?

2. Daha önce de belirtildiği gibi rasgele sayı üreteçlerinin bir sorunu da üretilen sayıların belirli bir bağımlılığa sahip olmalarıdır, yani I_i nci rasgele sayı

ile I_{i+k} ncı rasgele sayı arasında belirli bir korelasyonun bulunmasıdır. Örneğin **1.**'de bahsedilen RND alt programının böyle bir sorunu bulunmaktadır. Bir program yazarak bu sayı üreteci için k sayısını bulunuz. Verilen bir k sayısı için, r_i nci rasgele sayı ile r_{i+k} nci sayı karşılıklı çizilirse bağımlığın olup olmadığı kontrol edilebilir. Bağımlılık varsa Şekil 4.3'dekine benzer bir sonuç elde edilmesi gerekir.

3. Ağ komşuluğundan RASINT1 ve RASINT2 programlarını alınız. Bu programlar birim çemberin alanını bulmak üzere sırası ile Örnek 4.1'de anlatılan birinci ve ikinci yöntemlerin algoritmaları kullanılarak yazılmıştır. Programları önce inceleyiniz daha sonra derleyip çalıştırınız.

4. Ağ komşuluğundan RASINT3 programını alınız. Bu program birim kürenin hacmini bulmak üzere yazılmıştır. önce programı inceleyiniz daha sonra derleyip çalıştırınız.

5. Ağ komşuluğundan RASMET1 programını alınız. Bu program bir boyutlu Ising modelini çozmek üzere Örnek 4.x'de verilen algoritma kullanılarak yazılmıştır. Bu programı önce inceleyiniz, daha sonra derleyip çalıştırınız.

a) J/k_BT'nin pozitif değerleri için bu programı çalıştırarak veri elde ediniz. J/k_BT için 0.5, 1., 1.5, 2.0 ve 5.0 değerlerini kullanabilirsiniz. Yukarı doğru yönelmiş spinlerin sayısının MC adım sayısının bir fonksiyonu olarak grafiğini çiziniz. Ne elde ettiniz ve sonuçlar J/k_BT değerine nasıl bağlı?

b) (a)'yı J/k_BT'nin negatif değerleri için tekrar ediniz.

4.8 ALIŞTIRMA SORULARI

Bölüm 5

LİNEER DENKLEM SİSTEMLERİNİN ÇÖZÜMÜ

Bu bölümde lineer denklem sistemlerinin çözümü konusunda bazı ön bilgiler verilecektir. Gauss yoketme yöntemi ve temel olarak ona dayalı geliştirilen yöntemlerden bahsedilecektir.

5.1 Giriş

En genel haliyle n bilinmeyenli m tane lineer (doğrusal) denklemden oluşan bir denklem sistemi aşağıdaki gibi yazılabilir.

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1n} & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \dots & + & a_{2n} & = & b_2 \\ \vdots & & \vdots & & \dots & & \vdots & = & \vdots \\ a_{m1}x_1 & + & a_{m2}x_2 & + & \dots & + & a_{mn} & = & b_m \end{array} \tag{5.1}$$

Bu denklemlerde a_{ij}'ler ve b_i'ler bilinen sabitler, x_i'ler ise bilinmeyenlerdir. Bu denklem sistemi matris ve vektörler kullanılarak

$$\begin{bmatrix} a_{11} & a_{12} & . & . & . & a_{1n} \\ a_{21} & a_{22} & . & . & . & a_{2n} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ a_{m1} & a_{m2} & . & . & . & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ . \\ . \\ b_{m-1} \\ b_m \end{bmatrix} \tag{5.2}$$

şeklinde yazılabilir. Burada katsayılardan oluşan matrise $\mathbf{A}$, sağ tarafta bilinen b_i sabitlerinden oluşan vektöre $\mathbf{b}$ ve x_i bilinmeyenlerinden oluşan vektöre $\mathbf{x}$ diyelim. Böylece yukarıda verilen denklem sistemi kısa ve özlü olarak

$$\mathbf{Ax} = \mathbf{b} \tag{5.3}$$

şeklinde temsil edilebilir.

Yukarıda verilen lineer denklem sisteminin çözümünün (varsa) bütün özelliklerinin hemen hepsi $\mathbf{A}$ matrisinin özelliklerinde toplanmıştır. Burada üç farklı durum ortaya çıkabilir.

- Bilinmeyen sayısı denklem sayısından fazla ise ($n > m$), denklem sisteminin tek bir çözüm kümesi yoktur. Çözüm kümesi $n - m$ tane keyfi parametreye bağlı olacaktır ve bu nedenle sonsuz tane çözüm vardır.

- Denklem sayısı bilinmeyen sayısından fazla ise ($n < m$), bazı denklemler gereksizdir, yani bazı denklemler diğerlerinin lineer kombinasyonlarından oluşmuş demektir. Ancak $m - n$ tane denklem lineer bağımsız olabilir.

- Denklem sayısı ile bilinmeyen sayısı eşit ise ($n = m$) verilen bir $\mathbf{b}$ vektörü için denklem sisteminin bir tek çözüm kümesinin olması için gerekli ve yeterli şart $\mathbf{A}$ matrisinin tersinin olmasıdır. $\mathbf{A}$ matrisinin tersi ancak ve ancak matrisin determinantı (det$\mathbf{A}$) sıfırdan farklı ise vardır.

Bu andan itibaren sadece denklem sayısı ile bilinmeyen sayısının eşit olduğu ($n = m$) durum gözönüne alınacaktır.

Lineer denklem sistemlerini çözme yöntemleri genel olarak iki ayrı gruba ayrılabilir, *doğrudan* ve *iteratif (yineleme)* metotları.

Doğrudan Metotlar: Bu metotlarda yapılan işlemlerde yuvarlama veya başka hataların *olmaması* durumunda kesin sonucun sonlu sayıda yapılan aritmetik işlemlerden sonra bulunması gerekir. Fakat bilgisayarlar her zaman sonlu hassasiyetle işlem yaptıklarından dolayı doğrudan metotlar çoğunlukla kesin (analitik) sonuçlar vermez. Hatta bazı durumlarda yuvarlama hataları, kararsızlık ve hassasiyet kaybı nedeniyle çözüme bulaşan hatalar nedeniyle elde edilen sonuç çok kötü veya kullanılamaz halde olabilir. Dog/rudan çözümler için en çok kullanılan yöntem Gauss Yoketme Yöntemidir.

İteratif (yineleme) Yöntemler: Bu yöntemlerde başta kabul edilen yaklaşık bir çözüm, uygun algoritmlarla adım adım iyileştirilir. İyileştirmeye peş peşe gelen çözümler arasındaki fark kullanıcı tarafından belirlenen belirli bir hata payından az olana kadar kadar devam edilir (mümkünse). Bu yöntemlerde sonuca yakınsama çok yavaş veya hiç olmayabilir. Bazı durumlarda ise ıraksayabilir. Bu yöntemlerin programlanması daha basit ve çoğunlukla yuvarlama hatalarından çok etkilenmezler.

Bu yöntemlerden hangisinin kullanılacağına çözülecek problemin özelliklerine, bilinmeyen sayısına, eldeki bilgisayar olanaklarına göre karar verilir.

5.2 Gauss Yoketme Metodu

Bu yöntemin nasıl işlediğini görmek için kolaylık olsun diye sadece üç bilinmeyenli aşağıdaki denklem sisteminin çözüm kümesinin arandığını varsayalım.

$$\begin{array}{rcrcrcl} a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & a_{23}x_3 & = & b_2 \\ a_{31}x_1 & + & a_{32}x_2 & + & a_{33}x_3 & = & b_3 \end{array} \tag{5.4}$$

(5.4)'de verilen denklem sisteminin Gauss yöntemine göre çözümü iki aşamadan oluşur. Birincisi **ileriye doğru** yerine koyma ve çözümü veren son aşama ise **geriye doğru** yerine koyma olarak bilinir. Şimdi bu aşamaları adım adım görelim.

İleriye Doğru Yerine Koyma:

i) İlk adımda birinci denklemden x_1 çözülür ve bu çözüm ikinci ve üçüncü denklemlerde yerine yazılır. $a_{11} \neq 0$ olduğu kabul edilerek x_1 için çözüm yapılırsa

$$x_1 = \frac{b_1}{a_{11}} - \frac{a_{12}}{a_{11}}x_2 - \frac{a_{13}}{a_{11}}x_3 \tag{5.5}$$

elde edilir. x_1'in değeri diğer iki denklemde yerine konur ve denklemler yeniden düzenlenirse, x_1 ikinci ve üçüncü denklemden elenmiş olur. Elde edilen denklem kümesinin son hali

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & = & b_1 \\ & & a_{22}^{(1)}x_2 & + & a_{23}^{(1)}x_3 & = & b_2^{(1)} \\ & & a_{32}^{(1)}x_2 & + & a_{33}^{(1)}x_3 & = & b_3^{(1)} \end{array} \tag{5.6}$$

olacaktır. Burada $a_{ij}^{(1)}$ ile gösterilen yeni katsayıların ve $b_i^{(1)}$ ile gösterilen yeni sağ taraf elemanlarının değerleri için *Alıştırma Soruları* bölümüne bakınız.

ii) Şimdi (5.6)'daki denklem sisteminde, ikinci denklemde $a_{22}^{(1)} \neq 0$ kabul edilir ve x_2 için çözüm yapılır ve bu değer üçüncü denklemde yerine konursa lineer denklem sisteminin son hali

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & = & b_1 \\ & & a_{22}^{(1)}x_2 & + & a_{23}^{(1)}x_3 & = & b_2^{(1)} \\ & & & & a_{33}^{(2)}x_3 & = & b_3^{(2)} \end{array} \tag{5.7}$$

olur. $a_{33}^{(2)}$ ve $b_3^{(2)}$ değerleri daha öncekiler gibi bulunabilir. Bu işlemler okuyucuya ödev olarak bırakılmıştır. (5.7) lineer denklem sistemine karşılık gelen matris denklemi

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} \\ 0 & 0 & a_{33}^{(2)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2^{(1)} \\ b_3^{(2)} \end{bmatrix} \tag{5.8}$$

olacaktır. Dikkat edilirse katsayı matrisinin bütün alt köşegen elemanları sıfırdır. Burada $a_{33}^{(2)}$ katsayısının da sıfırdan farklı olduğunu varsayıyoruz. (Köşegen elemanlarının sıfır olması durumunda yapılması gerekenler üzerinde aşağıda durulacaktır). Lineer denklem sistemlerini çözmek üzere katsayı matrisinin üst köşegen matrisine dönüştürülmesi çözüm yolunda katedilmiş önemli bir adımdır. Bu aşamadan sonra lineer denklem sisteminin çözülmesi çok daha kolaydır.

Geriye Doğru Yerine Koyma:

(5.7) denkleminde kolayca görüldüğü gibi üçüncü denklemden x_3 hemen bulunabilir. Buradan $x_3 = b_3^{(2)}/a_{33}^{(2)}$ olacaktır. Bu değer ikinci denklemde yerine konursa, buradan x_2 için çözüm yapılabilir. Bulunan x_2 ve x_3 değerleri birinci denklemde yerine yazılırsa x_1'in değeri bulunmuş olur. Burada sondan geriye doğru çözüm yaparak en başa döndüğümüz için bu işlemler geriye doğru yerine koyma olarak bilinir.

Burada anlatılan ileriye doğru yerine koyma yoluyla denklem sistemini üst köşegen haline dönüştürme işlemi aslında birazdan göreceğimiz satır operasyonlarının denklemler veya denklem sistemine karşılık gelen matris üzerinde kullanılmasından başka bir şey değildir. (5.1) veya (5.4)'de verilen denklem sistemleri aşağıda verilen *satır* operasyonlarına istenildiği kadar tabi

tutulduğunda ortaya çıkan yeni denklem sisteminin çözümü verilen orijinal denklem sisteminin çözümü ile aynıdır.

Satır operasyonları:
i-) Bir denklemin sıfırdan farklı bir skaler ile çarpımı,
ii-) Bir denklemin bir skaler ile çarpımının başka bir denkleme eklenmesi, ve
iii-) İki denklemin yerlerinin değiştirilmesi.

Satır operasyonları denklem sistemine olduğu gibi uygulanabilir. Sağ taraf vektörünün katsayı matrisine onun son kolonu olacak şekilde eklenmesi ile elde edilen matrise uzatılmış katsayı matrisi denir ve $\mathbf{A}_b$ ile gösterilir. Satır operasyonları $\mathbf{A}_b$ matrisine de uygulanabilir. Satır operasyonlarının uygulanmasından sonra ortaya çıkan matris $\mathbf{A}_b$ matrisi ile aynı değildir, bu matrisler *eşdeğer* matrisler olarak bilinir. Fakat eşdeğer matrislere karşılık gelen lineer denklem sistemlerinin çözüm kümeleri ise aynıdır. (5.4)'de verilen denklem sisteminin uzatılmış matrisi

$$\mathbf{A}_b = \left[\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \end{array}\right] \tag{5.9}$$

olur.

Şimdi (5.9)'da verilen uzatılmış matrise köşegen altı elemanlarını sıfır yapacak şekilde satır operasyonlarını uygulayalım. Önce birinci satırı (yani birinci denklemi) a_{21}/a_{11} ile çarpıp ikinci satırdan (ikinci denklemden) çıkartalım. Benzer şekilde birinci satırı a_{31}/a_{11} ile çarpıp üçüncü satırdan çıkartalım. Elde edilen matrisin (denklemin) son hali

$$\mathbf{A}_b \sim \left[\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & b_1 \\ 0 & a_{22}-(\frac{a_{21}}{a_{11}})a_{12} & a_{23}-(\frac{a_{21}}{a_{11}})a_{13} & b_2-(\frac{a_{21}}{a_{11}})b_1 \\ 0 & a_{32}-(\frac{a_{31}}{a_{11}})a_{12} & a_{33}-(\frac{a_{31}}{a_{11}})a_{13} & b_3-(\frac{a_{31}}{a_{11}})b_1 \end{array}\right] \tag{5.10}$$

$$= \left[\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & b_1 \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & b_2^{(1)} \\ 0 & a_{32}^{(1)} & a_{33}^{(1)} & b_3^{(1)} \end{array}\right] \tag{5.11}$$

olur. Bu matrisin ikinci satırı benzer şekilde $a_{32}^{(1)}/a_{22}^{(1)}$ ile çarpılıp üçüncü satırdan çıkartılırsa

$$\mathbf{A}_b \sim \left[\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & b_1 \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & b_2^{(1)} \\ 0 & 0 & a_{33}^{(2)} & b_3^{(2)} \end{array}\right] \tag{5.12}$$

elde edilir. $a_{33}^{(2)}$ ve $b_3^{(2)}$ değerlerinin bulunması ödev olarak okuyucuya bırakılmıştır. Bu son matris denkleminde köşegen elemanları sıfırdan farklı ve köşegen altı elemanlarının hepsi sıfırdır, yani katsayı matrisi bir üst-köşegen matrisi haline dönüştürülmüştür. Daha önce olduğu gibi önce x_3 için çözüm yapılır ve geriye doğru yerine koyma ile sırası ile x_2 ve x_1 kolaylıkla bulunabilir. Buradaki işlemlerde de a_{11}, $a_{22}^{(1)}$ ve $a_{33}^{(2)}$ elemanlarının sıfırdan farklı olduğunu kabul ediyoruz.

ÖRNEK **5.1** `Aşağıda verilen denklem sistemini Gauss yoketme metodu ile çözünüz.`

$$\begin{array}{rcrcrcl} 2x_1 & + & 3x_2 & - & x_3 & = & 5 \\ 4x_1 & + & 3x_2 & - & 3x_3 & = & 3 \\ -2x_1 & + & 3x_2 & - & x_3 & = & 1 \end{array} \tag{5.13}$$

Verilen denklem sistemine karşılık gelen uzatılmış matris

$$\mathbf{A}_b = \left[\begin{array}{ccc|c} 2 & 3 & -1 & 5 \\ 4 & 4 & -3 & 3 \\ -2 & 3 & -1 & 1 \end{array}\right] \tag{5.14}$$

olacaktır. Bu matrisi diyagonal elemanları sıfır olmayan üst-köşegen matrisi haline getirmemiz gerekiyor. Bu amaçla birinci satırı -2 ile çarpıp ikinci satıra, birinci satırı 1 ile çarpıp üçüncü satıra ekleyelim. Sonuç

$$\mathbf{A}_b \sim \left[\begin{array}{ccc|c} 2 & 3 & -1 & 5 \\ 0 & -2 & -1 & -7 \\ 0 & 6 & -2 & 6 \end{array}\right] \tag{5.15}$$

olacaktır. Bu matrisin ikinci satırı 3 ile çarpılıp üçüncü satıra eklenirse eşedeğer matrisi

$$\mathbf{A}_b \sim \left[\begin{array}{ccc|c} 2 & 3 & -1 & 5 \\ 0 & -2 & -1 & -7 \\ 0 & 0 & -5 & -15 \end{array}\right] \tag{5.16}$$

olur. Bu sisteme karşılık gelen lineer denklem seti ise

$$\begin{array}{rcrcrcr} 2x_1 & + & 3x_2 & - & x_3 & = & 5 \\ & - & 2x_2 & - & x_3 & = & -7 \\ & & & - & 5x_3 & = & -15 \end{array} \tag{5.17}$$

olur. Önce x_3 için çözüm yapılırsa $x_3 = 15/5 = 3$ elde edilir. Bu değer ikinci denklemde yerine yazılırsa $x_2 = -2$ bulunur. x_2 ve x_3 birinci denklemde kullanılırsa $x_1 = 7$ olur.

5.3 Pivot

Gauss yoketme metodu anlatılırken köşegen elemanlarının sıfırdan farklı olduğunu varsaymıştık. Fakat bu varsayım her denklem sistemi için sağlanmayabilir. Aşağıdaki örneği gözönüne alalım.

ÖRNEK **5.2** *Verilen denklem sisteminin çözümünü Gauss yoektme metodunu kullanarak bulunuz.*

$$\begin{array}{rcrcl} x_2 & + & x_3 & = & 1 \\ x_1 & + & x_3 & = & 1 \\ x_1 & + & x_2 & = & 1 \end{array} \tag{5.18}$$

Çözüm: Bu denkleme karşılık gelen uzatılmış matris denklemi

$$\mathbf{A}_b = \left[\begin{array}{ccc|c} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{array}\right] \tag{5.19}$$

olacaktır. Bu matrisin köşegen elemanlarının hepsi sıfırdır. Gauss yoketme kuralına göre çözüm yapmak için katsayı matrisinin alt köşegen elemanlarını sıfırlamamız gerekir. Birinci satırı kullanarak ikinci ve üçüncü satırların ilk elemanlarını sıfırlayamayız. Bu durumdan kurtulmak için birinci ve ikinci satırları yer değiştirelim. Elde edilen matris

$$\mathbf{A}_b \sim \left[\begin{array}{ccc|c} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{array}\right] \tag{5.20}$$

olacaktır. Bu matrisin birinci satırı -1 ile çarpılıp üçüncü satıra eklenir ve ortaya çıkan matrisin ikinci satırı yine -1 ile çarpılır ve üçüncü satıra eklenirse sonuç olarak

$$\mathbf{A}_b = \left[\begin{array}{ccc|c} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & -2 & -1 \end{array}\right] \tag{5.21}$$

elde edilir. Bu denklem sisteminin çözümünün $x_1 = x_2 = x_3 = \frac{1}{2}$ olması gerektiği kolayca gösterilebilir.

En son örnekte olduğu gibi köşegen elemanlarında sıfır olması durumunda denklemlerin yer değiştirilerek sıfır köşegen durumundan kurtulma olayına 'pivot' işlemi denir. Kullanılan denkleme ise 'pivot denklemi' denir.

Pivot işlemi sadece sıfır köşegen elemanından kurtulmak için yapılmaz. Bir birlerinden çok büyük ve çok küçük sayıların toplanması (çıkarılması) ile meydana gelen hassasiyet kaybından dolayı ortaya çıkan sorunları gidermek için de pivot işlemi kullanılır. Genel kural katsayısı mutlak olarak en büyük olan denklemi pivot denklemi olarak kullanmaktır. Bunun için denklemlerin yer değiştirilmesinden kaynaklanan yeniden sıralamanın takip edilmesi gerekir.

Küçük değerli köşegen elemanlarının neden sorun yarattığını ve pivot işlemine neden gerek duyulduğunu göstermek için daha önce çözdüğümüz (5.4)'deki denklem sistemini yeniden gözönüne alalım. Bir satır operasyonundan sonra bu sisteme karşılık gelen uzatılmış matrisin

$$\mathbf{A}_b \sim \left[\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & b_1 \\ 0 & a_{22} - (\frac{a_{21}}{a_{11}})a_{12} & a_{23} - (\frac{a_{21}}{a_{11}})a_{13} & b_2 - (\frac{a_{21}}{a_{11}})b_1 \\ 0 & a_{32} - (\frac{a_{31}}{a_{11}})a_{12} & a_{33} - (\frac{a_{31}}{a_{11}})a_{13} & b_3 - (\frac{a_{31}}{a_{11}})b_1 \end{array}\right] \tag{5.22}$$

olacağı daha önce gösterilmişti (denklem (5.10)). a_{11}'in çok küçük, örneğin 10^{-5}, diğerlerinin 1 mertebesinde olduğunu varsayalım. Bu durumda x_2'nin yeni katsayısı

$$a_{22} - a_{12}\frac{a_{21}}{a_{11}} = 1 - \frac{1}{10^{-5}} \approx -10^5$$

olacaktır. Yani orijinal denklemdeki a_{22} katsayısının etkisi yok olacaktır. Bu daha sonra yapılacak işlemlere de yansıyarak devam edecektir. Bu tip sorunlarla karşılaşmamak için yapılan deneyler katsayısı en büyük denklemi köşegen elemanı olacak şekilde denklemlerin yeniden düzenlenmesinin en iyi çözüm olduğunu göstermiştir. Pivot işlemine tabi tutmak üzere sadece satırların (ve buna karşılık gelen sağ taraf vektör elemanlarının) yeniden düzenlenmesine **kısmi pivot** işlemi denir. Pivot işlemi hem satırlar hem sütunlar üzerinden de yapılabilir. Eğer sütunlar yeniden düzenlenmişse bu değişikliğe karşılık gelen değişkenlerinde sırasının değiştirilmesi gerekir. Hem satır hem sütun yer değiştirmeleri ile yapılan işleme **tam pivot** denir. Tam pivot daha çok

işlem gerektirdiğinden, çok daha iyi sonuçlar vermesine rağmen çoğunlukla kullanılmaz.

ÖRNEK **5.3** Aşağıda verilen lineer denklem sistemini a) pivot işlemine tabi tutmadan ve b) kısmi pivot işlemine tabi tutarak çözünüz. İşlemlerinizi üç haneli hassasiyet kullanarak yapınız. Elde ettiğiniz çözümleri bu denklem sisteminin çözümü olan $x_1 = 10$ ve $x_2 = 1$ ile karşılaştırınız.

$$\begin{array}{rcrcl} 0.005x_1 & + & 3.73x_2 & = & 3.78 \\ 0.457x_1 & - & 1.56x_2 & = & 3.01 \end{array} \tag{5.23}$$

Çözüm:
a) Bu denklem sisteminin uzatılmış matrisi

$$\mathbf{A}_b = \left[\begin{array}{rr|r} 0.005 & 3.73 & 3.78 \\ 0.457 & -1.56 & 3.01 \end{array}\right] \tag{5.24}$$

olur. Köşegen altı elemaları sıfırlamak için birinci satırı 0.457/0.005=91.4 ile çarpıp ikinci satırdan çıkartmamız lazım. Bu işlemler belirtildiği gibi üç haneli hassasiyet ile yapılırsa eşdeğer matris

$$\begin{aligned} \mathbf{A}_b &\sim \left[\begin{array}{rr|r} 0.005 & 3.73 & 3.78 \\ 0 & -341 - 1.56 & -345 + 3.01 \end{array}\right] \\ &\sim \left[\begin{array}{rr|r} 0.005 & 3.73 & 3.78 \\ 0 & -343 & -342 \end{array}\right] \end{aligned} \tag{5.25}$$

olur. Buradan $x_2 = 342/343 = 0.997$ olarak % 0.3 hata ile elde edilir. Diğer yanda $x_1 = (3.78 - 3.73 \cdot 0.997)/0.005 = 12$ bulunur. Bu ise gerçek değerden % 20 daha fazladır.
b) (5.24) denkleminde satırlar yer değiştirilirse eşdeğer matris

$$\mathbf{A}_b = \left[\begin{array}{rr|r} 0.457 & -1.56 & 3.01 \\ 0.005 & 3.73 & 3.78 \end{array}\right] \tag{5.26}$$

olacaktır. Birinci satır $0.005/0.457 = 0.011$ ile çarpılıp ikinci denklemden çıkarılır ve işlemler yine üç haneli hassasiyetle yapılırsa eşdeğer matrisi

$$\mathbf{A}_b = \left[\begin{array}{rr|r} 0.457 & -1.56 & 3.01 \\ 0 & 3.75 & 3.75 \end{array}\right] \tag{5.27}$$

olur. Bilinmeyenler için çözüm yapılırsa $x_1 = 10$ ve $x_2 = 1$ bulunur ve bunlar kesin çözümler ile aynıdır.

Son örnekte görüldüğü gibi pivot işlemi uygulamadan Gauss yoketme metodunu kullanmak çok tehlikeli olabilir.

5.4 LU Çarpanlarına Ayırma

Gauss yoketme metodunu uygulamanın yollarından biri verilen katsayı matrisini alt ve üst köşegen matris çarpanlarına ayırmaktır. $\mathbf{A}$ katsayı matrisinin

$$\mathbf{A} = \mathbf{LU} \tag{5.28}$$

şeklinde yazılabildiğini varsayalım. **L** alt köşegen, **U** ise üst köşegen bir matristir.

$$L = \begin{bmatrix} L_{11} & 0 & 0 & . & . & . & 0 \\ L_{21} & L_{22} & 0 & . & . & . & 0 \\ . & . & . & . & . & . & 0 \\ . & . & . & . & . & . & 0 \\ L_{n1} & L_{n2} & . & . & . & . & L_{nn} \end{bmatrix} \tag{5.29}$$

$$U = \begin{bmatrix} u_{11} & u_{12} & . & . & . & . & u_{1n} \\ 0 & u_{22} & 0. & . & . & . & u_{2n} \\ . & 0 & . & . & . & . & . \\ . & . & . & . & . & . & u_{n-1,n} \\ 0 & 0 & . & . & . & 0 & u_{nn} \end{bmatrix} \tag{5.30}$$

4×4 bir matris için (5.28) denklemi aşağıdaki gibi olacaktır.

$$\begin{bmatrix} L_{11} & 0 & 0 & 0 \\ L_{21} & L_{22} & 0 & 0 \\ L_{31} & L_{32} & L_{33} & 0 \\ L_{41} & L_{42} & L_{43} & L_{44} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \tag{5.31}$$

Daha sonra üzerinde durulacağı gibi genelde **L** matrisinin köşegen elemanları 1'e eşitlenir (normalize edilir). Bu durumda bu değerler zaten bilindiğinden onları ayrıca bir değişkende tutmaya gerek yoktur. Bu nedenle **A** matrisinin LU çarpanlarına ayrılmış hali tekrar **A** üzerine yazılarak hafıza ve yerden kazanılmış olur.

(5.28)'de verilen lineer denklem sisteminin katsayı matrisinin LU çarpanlarına ayrılmış olduğunu kabul edelim. Bu durumda

$$\mathbf{Ax} = \mathbf{LUx} = \mathbf{b} \tag{5.32}$$

yazılabilir. **Ux**=**y** olsun. Böylece bilinmeyen **x** ve **y** vektörleri için

$$\begin{aligned} \mathbf{Ly} &= \mathbf{b} \\ \mathbf{Ux} &= \mathbf{y} \end{aligned} \tag{5.33}$$

denklemleri yazılabilir. Önce birinci denklemden **y**'nin çözülmesi ve bu çözümün ikinci denklemde kullanılarak **x**'in çözülmesi ile sistemin çözüm kümesi elde edilir.

L ve **U** matrislerinin bilinmesi durumunda **y** ve **x** bilinmeyen vektörleri ileriye ve geriye yerine koyma ile bulunur. **Ly**=**b** denklemini ele alalım:

$$L = \begin{bmatrix} L_{11} & 0 & 0 & . & . & . & 0 \\ L_{21} & L_{22} & 0 & . & . & . & 0 \\ . & . & . & . & . & . & 0 \\ . & . & . & . & . & . & 0 \\ L_{n1} & L_{n2} & . & . & . & . & L_{nn} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_{n-1} \\ y_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ . \\ . \\ b_{n-1} \\ b_n \end{bmatrix} \tag{5.34}$$

Buradan $y_1 = b_1/L_{11}$ olacağı açıktır. Diğer bilinmeyenler ileriye doğru yerine koyma ile bulunabilir. Örneğin $y_2 = (b_2 - L_{21}y_1)/L_{22}$ olacaktır. En genel haliyle

çözüm

$$\begin{aligned} y_1 &= b_1/L_{11} \\ y_j &= (b_j - \sum_{i=1}^{j-1} L_{ji}y_i)/L_{jj}, \quad j = 2,3,\ldots,n \end{aligned} \tag{5.35}$$

ile verilir. **L**'nin köşegen elemanlarının 1'e eşit olması durumunda L_{jj} elemanlarına bölmeye gerek olmaz. Yukarıda verilen ileriye doğru yerine koyma ile **y** vektörünün bulunması idi.

Şimdi de geriye doğru yerine koyma ile **x** vektörünün bulunmasını görelim. **Ux=y** denklemini elemanları ile yazalım:

$$U = \begin{bmatrix} u_{11} & u_{12} & . & . & . & . & u_{1n} \\ 0 & u_{22} & 0. & . & . & . & u_{2n} \\ . & 0 & . & . & . & . & . \\ . & . & . & . & . & . & u_{n-1,n} \\ 0 & 0 & . & . & . & 0 & u_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_{n-1} \\ y_n \end{bmatrix} \tag{5.36}$$

Bu son denklem Gauss yönteminde bilinmeyenlerin geriye doğru yerine koyma yöntemi ile bulunmasından başka bir şey değildir. Bu denklemden $x_n = y_n/u_{nn}$ olacağı açıktır. Geriye doğru bir adım gidilirse bir sonraki bilinmeyen $x_{n-1} = (y_{n-1} - u_{n-1,n}x_n)/u_{n-1,n-1}$ şeklinde elde edilir. Bu durumun genelleştirilmesi ile çözüm aşağıda verilen denklemlerden elde edilecektir.

$$\begin{aligned} x_n &= y_n/u_{nn} \\ x_j &= (y_j - \sum_{i=j+1}^{n} u_{ji}x_i)/u_{jj}, \quad j = n-1, n-2, \ldots, 1 \end{aligned} \tag{5.37}$$

(5.35) ve (5.37) denklemleri aslında sırası ile ileriye ve geriye doğru yerine koyma yöntemlerinin algoritmasını oluşturur. Burada da farkedilmiş olabileceği gibi LU faktörleri bir defa elde edildikten sonra aynı çarpanlar değişik sağ taraf **b** vektörleri için istenildiği kadar kullanılabilir. Bu Gauss yönteminin doğrudan kullanıldığı durumdan farklıdır. Gauss yönteminde alt köşegen elemanları sıfırlanırken aynı zamanda sağ taraf vektörleri üzerindeki etkileride hesaplanıyordu. Fakat LU faktörlerine ayırma sağ taraf vektörlerinden tamam bağımsız olarak yapılabilir.

Çarpanlarına Ayırma (Crout Algoritması):

Yukarıda, verilen bir lineer denklem sisteminin katsayı matrisinin alt-üst köşegen (LU) çarpanlarına ayrılmış olması durumunda nasıl çözüleceği görüldü. Fakat LU çarpanlarına nasıl ayrılacağı üzerinde durulmadı. Asıl sorun verilen matrisi LU çarpanlarına ayırmaktır. Şimdi bunun nasıl yapılabileceğini görelim. (5.31) denkleminin ilk bir kaç terimini yazalım. Sol tarafta bulunan matris çarpımları sağ taraf elemanlarına birer birer eşitlenerek birinci

$$\begin{aligned} L_{11}u_{11} &= a_{11} \\ L_{11}u_{12} &= a_{12} \\ \vdots &= \vdots \\ L_{11}u_{1n} &= a_{1n} \end{aligned} \tag{5.38}$$

ikinci

$$\begin{aligned} L_{21}u_{11} &= a_{21} \\ L_{21}u_{12} + L_{22}u_{22} &= a_{22} \\ \vdots &= \vdots \\ L_{21}u_{1n} + L_{22}u_{2n} &= a_{2n} \end{aligned} \tag{5.39}$$

ve sonuncu ($n$ci) satır elemanlarını veren denklemler

$$\begin{aligned} L_{n1}u_{11} &= a_{n1} \\ L_{n1}u_{12} + L_{n2}u_{22} &= a_{n2} \\ \vdots &= \vdots \\ L_{n1}u_{1n} + L_{n2}u_{2n} + \ldots + L_{nn}u_{nn} &= a_{nn} \end{aligned} \tag{5.40}$$

olarak elde edilecektir. Bu denklemler en genel haliyle

$$L_{i1}u_{1j} + \ldots + = a_{ij} \tag{5.41}$$

şeklinde temsil edilebilirler.

5.5 Labaratuar Saati V

1. LIN1.FOR programını ağ komşuluğundan alınız. Bu programda "Numerical Recipes"den alınan iki alt program kullanılmaktadır. Bunlardan birincisi

```
LUDCMP(A,N,NP,INDX,D)
```

alt programıdır. Bu alt program Crout algoritmasını kullanarak verilen bir A matrisini alt ve üst köşegen matrislerine dönüştürmektedir. Ayrıca kısmi pivot işlemide yapılmaktadır. Programın argümanlarının tanımları şöyledir.

GİRDİLER:

A: Kavramsal boyutları (np × np), fiziksel boyutları (n × n) olan lineer denklem sisteminin katsayı matrisi.

N: A kare matrisinin fiziksel boyutları

NP: A kare matrisinin kavramsal boyutları

INDX: A matrisindeki satır değişimlerini kaydeden ve daha sonra kullanılacak olan N boyutlu bir vektör.

D: A matrisindeki satır değişimleri sayısı tek ise (-1), çift ise (+1) değerini alır. İleride determinant hesaplama vb. işlerinde kullanılacaktır.

ÇIKTILAR: A: matrisinin içeriği daha sonra kullanılmak üzere alt ve üst köşegen matrisleri ile değiştirilmektedir. A matrisinin orijinal hali daha sonra gerekecek ise başka bir yerde korunmalıdır.

Kullanılan diğer alt program ise

```
LUBKSB(A,N,NP,INDX,B)
```

adı ile verilmiştir. Bu LUDCMP alt programının çıktısını doğrudan kullanmak üzere yazılmıştır. Crout algoritmasında çözüm için gereken ileriye ve geriye doğru yerine koyma işlemlerini yapar. A, INDX ve D, LUDCMP alt programının çıktılarıdır, yani burada kullanılan A matrisi orijinal matris değil, onun alt ve üst köşegen haline dönüştürülmüş biçimi ile değiştirilmiş halidir. B ise lineer denklem sisteminin sağ tarafını veren N boyutlu vektördür. Program B vektörünü

çözüm vektörü ile değiştirerek iade eder. B vektörünün orijinal hali daha sonra kullanılacak ise bunun başka bir yerde korunması gerekir.

Lineer bir denklem sistemini çözmek için bu iki programın çağrılma sırası aşağıdaki şekilde olmalıdır.

```
CALL LUDCMP(A,N,NP,INDX,D)
CALL LUBKSB(A,N,NP,INDX,B)
```

Eğer aynı katsayı matrisi kullanılarak B1, B2, ... gibi birden fazla sağ taraf vektörü için çözüm yapılacaksa, LUDCMP alt programı sadece bir defa, LUBKSB alt programı ise her sağ taraf vektörü için bir defa çağrılmalıdır. Her çağırma işlemi sırasında uygun sağ taraf vektörü kullanılmalıdır.

LIN1.FOR programı katsayı matrisi ve sağ taraf vektörünün/vektörlerinin bir veri dosyasından okunacağı varsayılarak yazılmıştır. Kare katsayı matrisi olduğu gibi yazılmalı, onu hemen takip eden satır ise sağ taraf vektörleri içermelidir. Örneğin

$$\begin{aligned} 2x + y - z &= 1 \\ -x + 2y + z &= 6 \\ x + y + z &= 6 \end{aligned} \tag{5.42}$$

lineer denklem sistemi çözülecekse veri dosyasının içeriği aşağıdaki gibi olmalıdır.

```
3  1
2.   1. -1.
-1.   2.   1.
1.  1.   1.
1.  6.  6.
```

Burada birinci satırdaki ilk tamsayı kare matrisin boyutlarını, ikinci tam sayı ise aynı denklem sisteminin kaç tane farklı sağ taraf vektörü için çözüleceğini gösterir. Daha sonra gelen satırlar katsayı matrisinin satırları, en son satır ise sağ taraf vektörüdür. Birden fazla sağ taraf vektörü varsa bunlar alt alta yazılmalıdır.

a) Burada verilen denklem sisteminin katsayı matrisini ve sağ taraf vektörünü yukarıda gösterilen örneğe göre DATA1 adlı bir dosyaya yazınız. LIN1.FOR programını derleyiniz ve hazırladığınız bu veri dosyasını kullanarak yukarıdaki lineer denklem sisteminin çözümünü bulunuz.

b) Yukarıdaki denklem sistemini (1, 6, 6), (1, 2, -3) ve (2, 2, 2) sağ taraf vektörleri için çözüm yapacak şekilde DATA1 dosyasını DATA2 dosyası adı altında değiştiriniz. Bu sistemin çözümünü bulunuz.

2. Bir Matrisin Tersi ve Determinantı

Bir A kare matrisinin tersi A^{-1}

$$AA^{-1} = I \tag{5.43}$$

olarak tanımlanır. Burada I, A ile aynı boyutlu birim matristir. A matrisi 3×3 bir matris olsun ve A^{-1} matrisinin kolonları sırası ile (x_1, x_2, x_3), (y_1, y_2, y_3) ve (z_1, z_2, z_3) olsun. Bu durumda

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

olmalıdır. Yukarıdaki matris çarpımı yapılırsa örneğin (x_1, x_2, x_3) bilinmeyenleri için

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= 1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= 0 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= 0 \end{aligned} \tag{5.44}$$

denklem seti elde edilir. Bu ise $Ax = b$ denkleminin sağ taraf vektörünün $b = (1, 0, 0)$ olduğu durum için çözümüdür. Benzer şekilde y_i ve z_i bilinmeyenleri içinde çözüm yapılabilir. Bu nedenle yukarıda kullanılan alt-üst köşegen matrsilerine dönüştürme işleminin sonucu kullanılarak matrisin tersi de alınabilir. LIN2.FOR programı lineer denklem sistemini çözecek ve istenildiğinde katsayı matrisinin tersini alacak şekilde yeniden düzenlenmiştir. Programın geri kalanı LIN1.FOR ile tamamen aynıdır. Bu programı derleyip çalıştırınız ve kullanılan matrisin tersinide buldurunuz. Tersinin doğru bulunup bulunmadığından nasıl emin olabilirsiniz?

3. Yukarıda kullanılan LUDCMP alt programının alt-üst köşegen biçime dönüşmüş matris çıktısını kullanarak orijinal matrisin determinantını hesaplamak mümkündür. Bunun nasıl yapılabileceğini araştırınız. LIN2.FOR programını LIN3.FOR adı altında yeniden düzenleyerek lineer denklem sisteminin katsayı matrisinin determinantınıda hesaplayınız.

4. Boyutları belli iki kare matrisin çarpımını yapacak bir alt program yazmak için bir akış şeması hazırlayınız. Bu şemadan faydalanarak matris çarpımını yapacak bir alt program yazınız. LIN2.FOR programını LIN4.FOR adı altında yeniden düzenleyiniz ve bu programa matris çarpımını yapacak alt programı ekleyiniz. Bu alt programı kullanarak kullandığınız matrisi tersi ile çarparak birim matris elde edip etmediğinizi kontrol ediniz.

5. Katsayı matrisi üçgen bantlı olan lineer bir denklem sisteminin çözümünü bulacak bir alt program yazınız. Bu sistemde sadece ana köşegen ile alt ve üst köşegen elemanları gerektiğinden, sadece bu elemanları kaydetmek için N boyutlu üç vektör kullanmak yeterlidir.

5.6 ALIŞTIRMA SORULARI

1.) (a) (5.7) denklemindeki üslü katsayıların

$$\begin{aligned} a_{22}^{(1)} &= a_{22} - a_{21}a_{12}/a_{11} \\ a_{23}^{(1)} &= a_{23} - a_{21}a_{13}/a_{11} \\ a_{32}^{(1)} &= a_{32} - a_{31}a_{12}/a_{11} \\ a_{33}^{(1)} &= a_{33} - a_{31}a_{13}/a_{11} \\ b_2^{(1)} &= b_2 - a_{21}b_1/a_{11} \\ b_3^{(1)} &= b_3 - a_{31}b_1/a_{11} \end{aligned} \tag{5.45}$$

olması gerektiğini gösteriniz.

(b) (5.8) denklemindeki $a_{33}^{(2)}$ ve $b_3^{(2)}$'ün değerini bulunuz.

2.) Aşağıda verilen denklem sistemini LIN1.FOR programını kullanarak çözünüz.

$$
\begin{aligned}
x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} &= 55 \\
2x_1 - x_2 + 2x_3 + x_4 + 3x_5 - 4x_6 + x_7 - 2x_8 + x_9 - x_{10} &= -9 \\
x_1 - 2x_2 - 2x_3 + 3x_4 + 2x_5 - 4x_6 + 2x_7 - 2x_8 + 3x_9 + 2x_{10} &= 34 \\
3x_1 + 3x_2 - x_3 + 2x_4 + 5x_5 + 3x_6 - x_7 - 3x_8 + 2x_9 + 3x_{10} &= 74 \\
-x_1 + x_2 - x_3 + x_4 - x_5 + x_6 - x_7 + x_8 - x_9 + x_{10} &= 5 \\
5x_1 + 7x_2 + 2x_3 - 8x_4 + 4x_5 - 5x_6 + x_7 + 2x_8 + 3x_9 + x_{10} &= 43 \\
15x_1 - 20x_2 + x_3 + 5x_4 + x_5 - 8x_6 - 2x_7 + 3x_8 + 9x_9 + 5x_{10} &= 96 \\
x_1 - 30x_2 + 3x_3 + 15x_4 - 3x_5 + 2x_6 + 10x_7 + 3x_8 + 2x_9 + 7x_{10} &= 189 \\
35x_1 + 42x_2 + 3x_3 + 2x_4 - 12x_5 - 3x_6 - x_7 - x_8 - 4x_9 - 11x_{10} &= -103 \\
14x_1 + 10x_2 - x_3 + 7x_4 + 2x_5 - 7x_6 + 7x_7 + 3x_8 - x_9 - x_{10} &= 81
\end{aligned}
$$

Bölüm 6

SINIR ŞARTLI ADİ DİFERANSİYEL DENKLEMLER

Bu bölümde sınır şartlı diferansiyel denklemlerinin çözüm yöntemleri üzerinde durulacaktır. Ayrıca ilgili bir konu olan özdeğer-özvektör problemlerinin çözümüde tartışılacaktır.

6.1 Giriş

Diferansiyel denklemler adi ve kısmi olmak üzere genel iki sınıfa ayrılabilirler. Adi diferansiyel denklemler ise başlangıç şartlı ve sınır şartlı olmak üzere iki alt grupta incelenebilir. Kısmi diferansiyel denklemlerinin çözüm yöntemleri Bölüm 7'de tartışılacaktır. Adi diferansiyel denklemlerinin başlangıç şartlı olanları Bölüm 3'te incelenmişti. Bu bölüm sadece sınır şartlı lineer adi diferansiyel denklemlerinin çözüm yöntemlerinin incelenmesine ayrılmıştır. Sınır şartlı diferansiyel denklemlerini başlangıç şartlı diferansiyel denklemlerinden ayıran en önemli fark diferansiyel denkleminin sağlaması gereken şartların tanımlandığı noktalardır. Başlangıç şartlı diferansiyel denklemlerinde diferansiyel denkleminin sağladığı şartlar sadece bir tek noktada tanımlanırken, sınır şartlı diferansiyel denklemlerinde bu şartlar en az iki farklı noktada tanımlanır.

Sınır şartlı diferansiyel denklemlerinin sayısal çözümleri değişik yollarla yapılmakla beraber bu bölümde görülecek olan yöntemler türevlerin sonlu farklar şeklinde yazılmasına dayanır. Bölüm 3'de başlangıç şartlı adi diferansiyel denklemlerinin çözümünde aynı yöntem kullanılarak diferansiyel denklemde geçen türevler sonlu farklar cinsinden yazılmıştı. Burada sınır noktalarında uyulması gereken şartları sağlayan bir çözümün verilen aralıkta bulunması gerekir. Burada da verilen diferansiyel denkleminde geçen türevlerin hepsi sonlu farklar cinsinden yazılacaktır. Ayrıca verilen denklemi birinci mertebe diferansiyel denklemler olarak yazmanın bazı çözüm yöntemleri dışında bir avatajı olmadığından böyle bir yola başvurulmayacaktır.

Verilen lineer bir diferansiyel denklem sonlu fark yöntemleri kullanılarak yeniden yazılırsa ortaya bilinmeyenler cinsinden lineer bir denklem sistemi ortaya çıkar. Bu sistem doğrudan matris yöntemleri kullanılarak çözülebilir. Bu yönteme 'doğrudan yöntemler' denilmektedir. Ortaya çıkan lineer denklem sistemini çözmenin bir başka yolu, başlangıçta yaklaşık bir çözüm varsayarak bu

çözümü adım adım iyileştirerek istenen sınır şartlarını sağlayan bir çözüm bulmaktır. Bu yönteme 'iteratif' veya 'relaksasyon yöntemler' denir. Bu yöntemde başlangıçta kabul edilen çözümün sınır şartlarını sağlaması gerekmez. Aranan çözüme yakın bir çözüm herhangi bir şekilde tahmin edilebiliyorsa bu durumda daha kısa sürede çözüme yakınsama elde edilebilir. Çözüm hakkında hiç bir tahmin yapılamıyorsa tamamen keyfi bir çözüm önerisi ile başlanabilir.

Bu başlık altında incelenecek olan bir diğer konu ise özdeğer-özvektör problemleridir. Bu tip problemlerde verilen diferansiyel denklemi bir parametreye bağlıdır ve ancak o parametrenin belirli değerleri alması durumunda denklemin bir çözümü mevcuttur. Bu durumda hem çözümün hemde parametrenin hangi değeri için çözümün olabileceğini aynı anda bulmak gerekir ve genel olarak özdeğer-özvektör problemleri olarak bilinirler. Bu nedenle sınır şartlı diferansiyel denklemlerinin bir diğer çözüm yöntemi olarak 'özdeğer-özvektör metotları' incelenecektir.

Son olarak bu bölümde sadece lineer denklemlerin çözümü incelenecektir. Lineer olmayan denklemlerde türevler için sonlu farkların kullanılması ile elde edilen denklem lineer olmayan bir denklem sistemi olacaktır. Bunu çözmenin bir yolu Bölüm 2'de görülen Newton'un kök bulma yönteminin genelleştirilmiş halini kullanmaktır.

6.2 Sonlu Fark Yöntemleri

6.2.1 Doğrudan Çözüm

Bu yöntemi incelemek için ikinci mertebe lineer bir diferansiyel denklemi göz önüne alalım. Böyle bir denklem en genel haliyle

$$\frac{d^2y}{dx^2} + f(x)\frac{dy}{dx} + g(x)y = q(x) \tag{6.1}$$

şeklinde yazılabilir. Bu denklemde $f(x)$, $g(x)$ ve $q(x)$ bilinen fonksiyonlardır. Diferansiyel denkleminin $x \in [x_0, x_N]$ aralığında çözüleceğini varsayalım. Denklemin $x = x_0$ ve $x = x_N$'de sağlaması gereken sınır şartları olacaktır. Bunlar bu iki noktada y'ye ve onun türev(ler)ine bağlı denklemler olabilir. Ayrıca y'nin değeri ile türevini iki ayrı noktada bir birlerine bağlayan denklemlerde sınır şartı olarak karşımıza çıkabilir. En genel haliyle x_0 ve x_N noktasındaki sınır şartları sırası ile

$$a_1 y(x = x_0) + a_2 \frac{dy}{dx}|_{x=x_0} = a_3 \tag{6.2}$$

$$b_1 y(x = x_N) + b_2 \frac{dy}{dx}|_{x=x_N} = b_3 \tag{6.3}$$

şeklinde yazılabilir. Dikkat edilirse bu sınır şartları iki ayrı noktada tanımlıdır fakat iki nokta herhangi bir şekilde bir birlerine bağlı değildir.

(6.1)'de verilen denklemi çözmek için denklemde geçen birinci ve ikinci (varsa daha üst mertebeli) türevler için yaklaşık sonlu farklar kullanılacaktır. Bunları elde etmek için daha önce yapıldığı gibi $y(x)$'i $x = x \pm h$ noktalarında Taylor serisine açalım.

$$y(x+h) = y(x) + hy'(x) + \frac{h^2}{2}y''(x) + \frac{h^3}{6}y'''(x) + \dots \tag{6.4}$$

$$y(x-h) = y(x) - hy'(x) + \frac{h^2}{2}y''(x) - \frac{h^3}{6}y'''(x) + \dots \tag{6.5}$$

(6.4) denkleminden (6.5) denklemini çıkartalım. Elde edeceğimiz denklem

$$y(x+h) - y(x-h) = 2hy'(x) + O(h^3) \tag{6.6}$$

olacaktır. h^3 ve daha yüksek mertebeli terimler ihmal edilmiştir. Bu nedenle birinci türev için elde edilen yaklaşık sonlu fark denklemi ikinci mertebe hassastır. Benzer şekilde (6.4) ve (6.5) denklemleri taraf tarafa toplanırsa

$$y(x+h) + y(x-h) = 2y(x) + h^2y''(x) + O(h^4) \tag{6.7}$$

elde edilir. Burada ikinci türev için elde edilen yaklaşık sonlu fark üçüncü mertebe hassastır. Son iki denklem yeniden düzenlenirse

$$\begin{aligned} y'(x) &= \frac{y(x+h) - y(x-h)}{2h} \\ y''(x) &= \frac{y(x+h) - 2y(x) + y(x-h)}{h^2} \end{aligned} \tag{6.8}$$

elde edilir. Denklem (6.1)'deki türevler burada elde edilen yaklaşık sonlu farklar ile değiştirilecektir. Bunu yapmadan önce $x_n = x_0 + nh$ ve $y_n = y(x_n)$ tanımlarını kullanarak bu denklemleri yeniden yazalım:

$$\begin{aligned} y'(x) &= \frac{y_{n+1} - y_{n-1}}{2h} \\ y''(x) &= \frac{y_{n+1} - 2y_n + y_{n-1}}{h^2} \end{aligned} \tag{6.9}$$

Denklem (6.1) yukarıda elde edilen sonlu farklar kullanılarak yeniden yazılırsa

$$\frac{y_{n+1} - 2y_n + y_{n-1}}{h^2} + f_n(\frac{y_{n+1} - y_{n-1}}{2h}) + g_n y_n = q_n \tag{6.10}$$

elde edilir. Bu denklemde $f_n = f(x_n)$, $g_n = g(x_n)$ ve $q_n = q(x_n)$ tanımları kullanılmıştır. Bu denklem tekrar düzenlenirse

$$(1 - \frac{h}{2}f_n)y_{n-1} + (-2 + h^2g_n)y_n + (1 + \frac{h}{2}f_n)y_{n+1} = h^2q_n, \qquad n = 0, 1, 2, \ldots N-1, N \tag{6.11}$$

elde edilir. Burada n indisi verilen sınır şartlarına bağlı olmak üzere 0 veya 1'den başlayarak $N-1$ veya N'e kadar gidebilir. $x = x_0$'daki sınır şartları türev içeriyorsa n 0'dan, içermiyorsa 1'den başlayacaktır. Benzeri bir durum $x = x_N$'deki sınır şartı içinde geçerlidir.

Sınır şartlarının $y(x_0) = \alpha$ ve $y(x_N) = \beta$ olduğunu kabul edelim. Bu durumda (6.11) denklemi $n = 1$ için

$$(1 - \frac{h}{2}f_1)y_0 + (-2 + h^2g_1)y_1 + (1 + \frac{h}{2}f_1)y_2 = h^2q_1 \tag{6.12}$$

olacaktır. $i = 2, 3, \ldots, N-2$ için aynı denklem

$$(1 - \frac{h}{2}f_i)y_{i-1} + (-2 + h^2g_i)y_i + (1 + \frac{h}{2}f_i)y_{i+1} = h^2q_i \tag{6.13}$$

olur. Son olarak $n = N-1$ için bu denklem

$$(1 - \frac{h}{2}f_{N-1})y_{N-2} + (-2 + h^2g_{N-1})y_{N-1} + (1 + \frac{h}{2}f_{N-1})y_N = h^2q_{N-1} \tag{6.14}$$

haline dönüşür. (6.12-14)'de toplam $N-1$ tane denklem vardır. Toplam bilinmeyen sayısı ise $y_0, y_1, \ldots, y_N$ olmak üzere $N+1$ tanedir. Bilinmeyenlerin hepi için çözüm yapabilmek için iki tane daha denkleme ihtiyaç vardır. Bunlar sınır şartlarından gelecektir. $y_0 = \alpha$ ve $y_N = \beta$ sınır şartlarından y_0 ve y_N bilindiğinden bilinmeyen sayısı $N-1$ tane olur. y_0 ve y_N'in değerleri yerlerine yazılır ve (6.12-14) denklemleri matris haline getirilirse aşağıdaki denklem sistemi elde edilir.

$$\begin{bmatrix} d_1 & c_1 & 0 & . & . & 0 \\ a_2 & d_2 & c_2 & 0 & . & 0 \\ 0 & a_3 & . & . & . & 0 \\ . & . & . & . & . & . \\ . & . & . & . & d_{N-2} & c_{N-2} \\ 0 & . & . & . & a_{N-1} & d_{N-1} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_{N-2} \\ y_{N-1} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ . \\ . \\ b_{N-2} \\ b_{N-1} \end{bmatrix} \tag{6.15}$$

Burada a_1 ve c_{N-1} keyfidir. Diğer matris elemanları ise şu şekilde verilir:

$$\begin{array}{ll} d_i = (-2 + h^2 g_i), & i = 1, 2, \ldots, N-2, N-1, \\ c_i = (1 + \frac{h}{2} f_i), & i = 1, 2, \ldots, N-3, N-2, \\ a_i = (1 - \frac{h}{2} f_i), & i = 2, 3, \ldots, N-2, N-1 \end{array} \tag{6.16}$$

Matrisin bilinen sağ tarafı ise şu şekildedir:

$$\begin{array}{ll} b_i = h^2 q_i, & i = 2, 3, \ldots N-2 \\ b_1 = h^2 q_1 - (1 - \frac{h}{2} f_1)\alpha, & \\ b_{N-1} = h^2 q_N - (1 + \frac{h}{2} f_N)\beta, & \end{array} \tag{6.17}$$

(6.15)'de verilen lineer denklem sistemi Bölüm 5'de verilen üçlü diyagonal matrislerin çözüm yöntemleri kullanılarak çözülebilir.

ÖRNEK **6.1**

$$y''(x) - y(x) = 0 \tag{6.18}$$

diferansiyel denklemini $y(0) = 0$ *ve* $y(1) = 1$ *sınır şartlarını kullanarak doğrudan çözünüz. Bu denklemin analitik çözümünün* $y(x) = \sinh(x)/\sinh(1)$ *olduğunu gösteriniz. Türevler için sonlu farklar kullanılması sonucunda elde edilecek lineer denklem sisteminin* $x_0 = 0$, $x_N = 1$ *ve* $d = -(2 + h^2)$ *olmak üzere*

$$\begin{bmatrix} d & 1 & 0 & . & . & 0 \\ 1 & d & 1 & 0 & . & 0 \\ 0 & 1 & . & . & . & 0 \\ . & . & . & . & . & . \\ . & . & . & . & d & 1 \\ 0 & . & . & . & 1 & d \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_{N-2} \\ y_{N-1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ . \\ . \\ 0 \\ -1 \end{bmatrix} \tag{6.19}$$

ile verileceğini gösteriniz. Bu denklem sistemini dolayısı ile verilen diferansiyel denklemini çözecek bir program yazınız. Bu amaçla yazılmış bir program SADI1.FOR adı altında Ek-F'de verilmiştir. Bu programı derleyip çalıştırınız. $N = 10,\ 20,\ 50$ *ve* 100 *için çözüm yapınız ve her durum için analitik ve sayıscal çözümlerin grafiklerini çizerek karşılaştırınız. Yapılan hatanın grafiğini de çizerek en büyük hatanın nerelerde yapıldığını belirtiniz.*

6.2.2 Relaksasyon (İteratif) Yöntem

(6.1) denkleminde türevler için sonlu farklar kullanılarak elde edilen son hali denklem (6.11)'de verilmişti. Bu denklemi göz önüne alalım ve y_n için çözüm yapalım.

$$y_n = \frac{1}{2 - h^2 g_n}[(1 - \frac{h}{2} f_n) y_{n-1} + (1 + \frac{h}{2} f_n) y_{n+1} - h^2 q_n] \qquad (6.20)$$

Iteratif yöntem aşağıdaki şekilde kullanılır: i) Önce $i = 0, 1, 2, \ldots, N$ olmak üzere bütün y_i'ler için tahmini bir çözüm kabul edilir. ii) Daha sonra denklem (6.20) kullanılarak eski y_i'lerden *yeni* y_i'ler bulunur. Bu işleme eski ve yeni çözümler arasındaki fark (veya göreli fark) baştan belirlenmiş bir hata toleransından daha küçük olana kadar devam edilir. Iterasyon sayısı j ile gösterilirse $j$nci iterasyonda çözüm

$$y_n^{(j)} = \frac{1}{2 - h^2 g_n}[(1 - \frac{h}{2} f_n) y_{n-1}^{(j-1)} + (1 + \frac{h}{2} f_n) y_{n+1}^{(j-1)} - h^2 q_n] \qquad (6.21)$$

olacaktır. Başlangıçta varsayılan veya tahmin edilen çözüm $y_n^{(0)}$ $(n = 0, 1, 2, \ldots N)$ olarak gösterilir ve bu y_{eski} adlı bir vektörde tutulabilir. İlk iterasyonda x_i noktasındaki bir çözüm, bu noktanın bir öncesindeki ve bir sonrasındaki $y_{i-1}^{(0)}$ ve $y_{i+1}^{(0)}$ değerleri kullanılarak bulunur. Bu durum bütün noktalar için tekrar edilerek yeni çözüm vektörü y_{yeni} elde edilir. Bu yöntem Jacobi iterasyon yöntemi olarak bilinir.

y_n'leri bulmanın daha iyi bir yolu hali hazırda hesaplanmış değerlerin anında kullanıldığı Gauss-Seidel yöntemini kullanmaktır. Denklem (6.21) dikkatli incelendiğinde görüleceği gibi bir x_i noktasındaki y_i değerini hesaplamak için x_{i-1} ve x_{i+1} noktalarındaki çözümler kullanılmaktadır. Fakat eğer bilinmeyenler soldan sağa doğru taranarak hesaplanıyorsa, x_{i-1} noktasındaki yeni çözüm zaten hesaplanmış durumdadır. Bu nedenle x_i noktasındaki çözüm bulunurken solda kalan x_{i-1} noktasındaki yeni çözüm ve x_{i+1} noktasındaki eski çözüm kullanılabilir. Böylece bilinmeyen y_n vektörünün soldan sağa doğru tarandığı bir durum için Gauss-Seidel yöntemi

$$y_n^{(j)} = \frac{1}{2 - h^2 g_n}[(1 - \frac{h}{2} f_n) y_{n-1}^{(j)} + (1 + \frac{h}{2} f_n) y_{n+1}^{(j-1)} - h^2 q_n] \qquad (6.22)$$

olacaktır. Benzer şekilde eğer çözüm sağdan sola doğru taranıyorsa bu durumda çözüm

$$y_n^{(j)} = \frac{1}{2 - h^2 g_n}[(1 - \frac{h}{2} f_n) y_{n-1}^{(j-1)} + (1 + \frac{h}{2} f_n) y_{n+1}^{(j)} - h^2 q_n] \qquad (6.23)$$

ile verilecektir. (6.21-23) denklemlerinde sınır şartları ayrıca özel olarak uygulanmalıdır.

ÖRNEK **6.2** *Örnek 6.1'de verilen diferansiyel denklemini Gauss-Seidel yöntemi ile çözen bir program SADI2.FOR adı altında Ek-E'de verilmiştir. Bu programı inceleyiniz ve özellikle sınır şartlarının nasıl kullanıldığına dikkat ediniz. Daha sonra programı derleyip çalıştırınız.* 10^{-5} *mertebesinde bir hata toleransı sağlamak için kaç iterasyon gerekmektedir?* 10^{-6} *ve* 10^{-8} *için ne kadar? özümlerinizde 19 ve 39 aralık kullanınız. Analitik ve sayısal çözümleri ve yapılan hatanın grafiğini çizerek sayısal çözümün ne kadar iyi olduğuna karar veriniz.*

6.2.3 Özdeğer-Özvektör problemleri: Lineer, Homojen Diferansiyel Denklemler

Sınır şartları lineer ve homojen olan lineer homojen diferansiyel denklemler de bir parametreye bağımlılık varsa, diferansiyel denkleminin çoğunlukla parametrenin sadece belirli değerleri için çözüm vardır. Bu tip problemler özdeğer-özvektör problemleri olarak bilinir.

Sınır şartları lineer ve homojen olan ikinci mertebe lineer ve homojen bir denklem en genel haliyle

$$\begin{aligned} y'' + p(x)y' + q(x)y &= 0, \quad x_0 < x < x_N \\ a_1 y(x_0) + a_2 y'(x_0) &= 0 \\ b_1 y(x_N) + b_2 y'(x_N) &= 0, \end{aligned} \tag{6.24}$$

şeklinde yazılabilir. Burada $p(x)$ ve $q(x)$'ler bilinen fonksiyonlar, a_i ve b_i'ler sabitlerdir. Homojen denklemler için her zaman $y(x) = 0$ bariz çözümü mevcuttur. Burada önemli olan bariz çözümün dışında çözümlerin olup olmadığını bulmaktır. Ayrıca diferansiyel denkleminin lineer ve homojen olmasından dolayı eğer $y = \Phi(x)$ bir çözüm ise c keyfi bir sabit olmak üzere $y = c\Phi(x)$'de bir çözüm olur. Aşağıda verilen örnek durumun daha iyi kavranmasını sağlayacaktır.

ÖRNEK **6.3** *Aşağıdaki diferansiyel denklemini verilen sınır şartları için çözünüz.*

$$\begin{aligned} y'' + y &= 0 \\ y(0) = 0, \quad y(1) &= 0 \end{aligned} \tag{6.25}$$

Verilen diferansiyel denkleminin en genel çözümü a ve b keyfi sabitler olmak üzere

$$y(x) = a\cos(x) + b\sin(x) \tag{6.26}$$

ile verilir. Birinci sınır şartı kullanılırsa $y(0) = a = 0$ olur. İkinci sınır şartı kullanılırsa $y(1) = b\sin(1) = 0$ olacaktır. $\sin(1)$ sıfırdan farklı olduğundan $b = 0$ olmalıdır. Böylece verilen diferansiyel denkleminin gereken sınır şartlarını sağlayan $y = 0$ bariz çözümünden başka çözümü olmadığı anlaşılır.

ALIŞTIRMA **6.1**

$$\begin{aligned} y'' + \pi^2 y &= 0 \\ y(0) = 0, \quad y(1) &= 0 \end{aligned} \tag{6.27}$$

diferansiyel denkleminin çözümünün, a keyfi bir sabit olmak üzere, $y(x) = a\sin(\pi x)$ şeklinde olması gerektiğini gösteriniz.

ÖRNEK **6.4** *λ pozitif bir parametre olmak üzere aşağıdaki diferansiyel denklemini verilen sınır şartları için çözünüz.*

$$\begin{aligned} y'' + \lambda y &= 0 \\ y(0) = 0, \quad y(1) &= 0 \end{aligned} \tag{6.28}$$

Verilen diferansiyel denkleminin en genel çözümü a ve b keyfi sabitler olmak üzere

$$y(x) = a\cos(\sqrt{\lambda}x) + b\sin(\sqrt{\lambda}x) \tag{6.29}$$

ile verilir. Birinci sınır şartı kullanılırsa $y(0) = a = 0$ *olur. İkinci sınır şartı kullanılırsa* $y(1) = b\sin(\sqrt{\lambda}) = 0$ *olacaktır. Bu denklem* $b = 0$ *ve/veya* $\sin(\sqrt{\lambda}) = 0$ *olması durumunda sağlanacaktır.* $b = 0$ *seçilirse fiziksel olarak anlamı olmayan bariz çözüm elde edilecektir. Bariz olmayan çözümleri aradığımızdan* $b \neq 0$ *ve* $sin(\sqrt{\lambda}) = 0$ *durumunu seçelim. En son denklem ancak*

$$\lambda = n^2\pi^2, \quad n = 1, 2, \ldots \tag{6.30}$$

olması durumunda sağlanır. Böylece çözümler b keyfi olmak üzere

$$y_n(x) = b\sin(n\pi x) \tag{6.31}$$

şeklinde olacaktır. λ*'lar özdeğerler ve her* λ *değerine karşılık gelen* y_n*'ler ise özfonksiyonlar olarak bilinir.*

ALIŞTIRMA **6.2** λ *pozitif bir parametre olmak üzere*

$$\begin{gathered} y'' + \lambda y = 0 \\ y(0) = 0, \quad y(1) + y'(1) = 0 \end{gathered} \tag{6.32}$$

sınır şartlı probleminin özdeğer ve özfonksiyonlarını bulunuz. Özdeğer denkleminin

$$\sqrt{\lambda} = -\tan(\sqrt{\lambda}) \tag{6.33}$$

olduğunu gösteriniz. Bu denklemi Bölüm 2'de görülen kök bulma yöntemlerinden birini kullanarak çözünüz. Özdeğerleri küçükten büyüğe doğru sıralayarak ilk on tanesini bulunuz.

Özdeğer-özvektö problemlerinin sayısal olarak nasıl çözüleceğini görmek için aşağıda verilen ikinci mertebe diferansiyel denklemini göz önüne alalım.

$$y'' + f(x)y' + \lambda g(x)y = 0 \tag{6.34}$$

Diferansiyel denkleminin sağlaması gereken sınır şartları en genel haliyle denklem (6.24)'de verilen sınır şartlarına benzeyecektir. Bu denklemde λ herhangi bir parametredir. Birinci ve ikinci türevler için denklem (6.8)'de verilen sonlu farklar kullanılır ve denklem yeniden düzenlenirse

$$a_n y_{n-1} + d_n y_n + c_n y_{n+1} = -\lambda y_n, \quad n = 0, 1, 2, \ldots N \tag{6.35}$$

elde edilir. Bu denklemin katsayıları

$$a_n = \frac{1 - \frac{h}{2}f_n}{h^2 g_n}, \quad d_n = \frac{-2}{h^2 g_n}, \quad c_n = \frac{1 + \frac{h}{2}f_n}{h^2 g_n} \tag{6.36}$$

olacaktır. Bu denklemlerde sınır şartlarının uygun şekilde kullanılması gerekir. Sınır şartlarının denklem (6.24)'deki gibi verildiğini varsayalım. Bu durumda sınır şartlarında geçen türevler için onların sonlu fark yaklaşımları kullanılırsa

$$a_1 y_0 + a_2\left(\frac{y_1 - y_{-1}}{2h}\right) = 0, \quad b_1 y_N + b_2\left(\frac{y_{N+1} - y_{N-1}}{2h}\right) = 0 \tag{6.37}$$

denklemleri elde edilir. Dikkat edilirse bu son denklemde y_{-1} ve y_{N+1} değerleri diferansiyel denkleminin çözülmesi gereken $[x_0, x_N]$ aralığının dışına taşmaktadır. Sınır şartlarından y_{-1} ve y_{N+1} için çözüm yapılırsa

$$y_{-1} = \frac{2ha_1 y_0}{a_2} + y_1, \quad y_{N+1} = y_N - \frac{2hb_1 y_N}{b_2} \tag{6.38}$$

elde edilir. Bu denklemler (6.35) denkleminde kullanılır ve ortaya çıkan denklemler matris denklemi halinde yazılırsa

$$\begin{bmatrix} d_0' & c_0' & 0 & . & . & 0 \\ a_1 & d_1 & c_1 & 0 & . & 0 \\ 0 & a_2 & d_2 & c_2 & . & 0 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & a_{N-1} & d_{N-1} & c_{N-1} \\ 0 & . & . & . & a_N' & d_N' \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ . \\ . \\ . \\ y_{N-1} \\ y_N \end{bmatrix} = -\lambda \begin{bmatrix} y_0 \\ y_1 \\ . \\ . \\ . \\ y_{N-1} \\ y_N \end{bmatrix} \tag{6.39}$$

elde edilecektir. Yukarıdaki denklemde, $i = 0, 1, 2, \ldots N$ olma üzere a_i, d_i ve c_i'ler denklem (6.36)'da verildiği gibi olmak üzere

$$\begin{aligned} d_0' &= d_0 - \frac{2ha_1}{a_2} a_0, \quad c_0' = a_0 + c_0 \\ a_N' &= a_N + c_N, \quad d_N' = d_N - \frac{2hb_1}{b_2} c_N \end{aligned} \tag{6.40}$$

şeklindedir. Sonuçta elde edilen denklem, A denklem (6.39)'da gösterilen katsayı matrisi, y aranan çözüm vektörü olma üzere

$$Ay = -\lambda y \tag{6.41}$$

şeklinde yazılabilecek bir özdeğer-özvektör denklemidir. Bilindiği gibi bu denklemin ancak

$$|A - \lambda I| = 0 \tag{6.42}$$

olduğu durumlar için çözümü vardır. Yukarıdaki denklem A matrisinin karakteristik denklemidir ve determinantın açılması ile ortaya çıkan N mertebeli polinomun kökleri problemin aranan özdeğerleridir. Bu polinomun kökleri Bölüm 2'de görülen kök bulma yöntemleri kullanılarak bulunabilir. Fakat aranan özdeğer ve bunlara karşılık gelen özvektörleri daha etkin bulma yöntemleri vardır. Bu işleri yapacak alt programlar için başvurulabilecek iyi kaynaklardan bir tanesi LAPACK deposudur.

Dikkat edilirse denklem (6.42)'deki A matrisi $f(x) = 0$ ise simetrik reel bir matris olur. Simetrik matrislerin özdeğerlerini bulmak genellikle daha kolay olduğundan bu andan itibaren bu bölümde sadece reel ve simetrik matrisler gözönüne alınacak en genel durum sonunda bir örnek ile incelenecektır.

ÖRNEK **6.5** *Quantum Mekaniksel Basit Harmonik Salınıcı:*

$V(x) = \frac{1}{2}kx^2$ şeklinde harmonik bir potansiyel içinde bulunan m kütleli bir parçacığın Schrödinger denklemi

$$-\frac{\hbar^2}{2m}\frac{d^2u}{dx^2} + V(x)u(x) = Eu(x) \tag{6.43}$$

olur. Burada $u(x)$ parçacığın dalga fonksiyonu, E enerji özdeğerleridir. $w = \sqrt{k/m}$ olmak üzere bu denklemde $z^2 = (\sqrt{mk}/\hbar)x^2$ ve $\lambda = 2E/\hbar w$ değişken değişimleri yapılırsa Schrödinger denklemi

$$-\frac{d^2u}{dx^2} + z^2u = \lambda u \tag{6.44}$$

haline gelir. İndirgenen bu denklemde özdeğerler $m = 0, 1, 2, \ldots$ olmak üzere $\lambda = (2m+1)$ ile verilir. Bu denklemin sağlaması gereken sınır şartları $u(-\infty) = 0$ ve $u(\infty) = 0$'dir. (Aslında parçacık bağımlı bir hareket yaptığından dolayı $u(x)$'in türevlerinin de $\pm\infty$'de sıfır olması gerekir. Bu durum bir sınır şartı olarak değil, elde edilen çözümün ne kadar iyi olduğunu kontrol etmek için kullanılabilir).

Türevler için ikinci mertebe sonlu fark yaklaşımı kullanarak (6.44) denklemini sonlu farklar cinsinden yazalım. Elde edilecek denklem

$$-\frac{1}{h^2}u_{n-1} + (\frac{2}{h^2} + z_n^2)u_n - \frac{1}{h^2}u_{n+1} = \lambda u_n \tag{6.45}$$

olacaktır. Verilen sınır şartları $\pm\infty$'de tanımlıdır. Sayısal olarak integral alma aralığını $-\infty$'dan $+\infty$'a kadar almanın olanağı yoktur. İzlenebilecek bir yol integrali belirli bir $-x_0$'dan $+x_0$'a kadar almak ve elde edilen özdeğerler bu sınırlara bağlı olmayana kadar x_0'ı artırmaktır. Ayrıca verilen sınır şartlarının gerçekten sağlanıp sağlanmadığı kontrol edilmelidir. Buna ek olarak yukarıda da belirtildiği gibi bağımlı hareket yapan parçacığın dalga fonksiyonunun türevleri de bu limitlerde sıfır olmalıdır. Bu durumun sağlanıp sağlanmadığını ayrıca kontrol etmek gerekir. x_0 için optimum bir değerin bulunması önemlidir. Eger x_0 çok büyük seçilirse integral alınan bölgenin çoğu yerinde dalga fonksiyonu sıfıra çok yakın olacağından çözüm için harcanan zamanın büyük bir kısmı fiziksel olarak anlamı olmayan bir çözüm bölgesi için harcanmış olacaktır. Öte yanda x_0 çok küçük seçilirse bu durumda sınır şartları tam olarak sağlanmayacak ve bulunan özdeğerler seçilen sınıra göre değişecektir. Deneme-yanılma yolu ile uygun limitler her zaman bulunabilir.

Verilen sınır şartlarına göre $u(-x_0) = u_0 = 0$ ve $u(+x_0) = u_N = 0$ olacağından elde edilecek olan matris denklemi

$$\begin{bmatrix} d_1 & c_0 & 0 & . & . & 0 \\ c_0 & d_2 & c_0 & 0 & . & 0 \\ 0 & c_0 & d_3 & c_0 & . & 0 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & c_0 & d_{N-2} & c_0 \\ 0 & . & . & . & c_0 & d_{N-1} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ . \\ . \\ . \\ u_{N-2} \\ u_{N-1} \end{bmatrix} = -\lambda \begin{bmatrix} u_1 \\ u_2 \\ . \\ . \\ . \\ u_{N-2} \\ u_{N-1} \end{bmatrix} \tag{6.46}$$

olacaktır. Burada $d_i = \frac{2}{h^2} + z_i^2$ ve $c_0 = -\frac{1}{h^2}$ olarak tanımlıdır. Görüldüğü gibi katsayı matrisi gerçel, simetrik ve üçlü diyagonaldir.

Bu özdeğer-özvektör problemini çözmek için [3]'den alınan TQLI alt programı kullanılmıştır. TQLI gerçel, simetrik ve üçlü diyagonal matrisler için bir özdeğer ve istenildiği taktirde bir özvektör çözücüsüdür. Programın detayları için [3]'e bakınız. Çok daha genel özdeğer-özvektör problemlerinin çözümü için kullanılabilecek alt programlar için http://www.netlib.org/lapack *sitesine bakınız.*

a) Verilen değişken değişimi yapılırsa denklem (6.43)'ün (6.44)'e dönüşeceğini gösteriniz.

b) (6.46) denkleminde verilen sistemi çözmek üzere yukarıda sözü edilen TQLI alt programını kullanan SADI3.FOR adlı bir program Ek-F'de verilmiştir.

SADI3 programını $x_0 = -2$, $x_e = +2$ ve $N = 19$ için çalıstırınız. Elde edilen ilk üç özvektörün grafiğini çiziniz. Dalga fonksiyonları sınır noktalarında istenen şartları sağlıyor mu? Sağlamıyorsa sebep nedir? Özdeğerleri analitik çözüm ile karşılaştırınız. Hata yüzdesi nedir?

c) (b)'yi $x_0 = -5$, $x_e = +5$ ve $N = 19$ için tekrar ediniz. Şimdi elde edilen özvektörler istenen şartları sağlıyormu? Özdeğerlerdeki hata yüzdesi nedir?

d) (c)'yi $N = 39$ ve $N = 59$ için tekrarlayınız.

e) (c)'yi $N = 159$ için tekrarlayınız. Hesaplanan özdeğerlerin hassasiyetinde önemli bir artma gözlediniz mi?

6.3 Labaratuar Saati-VI

1.

$$y''(x) - y(x) = 0 \tag{6.47}$$

diferansiyel denklemini $y(0) = 0$ ve $y(1) = 1$ sınır şartları ile çözen SADI1.FOR programını derleyiniz ve çalıştırınız. Bu programda ortaya çıkan lineer denklem sisteminin katsayı matrisi üçlü diyagonal olduğundan bu tip denklem sistemlerini çözmek için hazırlanan TRIDAG alt programı kullanılmıştır[3]. SADI1 programı yukarıda verilen diferansiyel denklemini çözmek üzere düzenlenmiş isede, bu program çok daha geneldir ve

$$y'' + f(x)y' + g(x)y = q(x) \tag{6.48}$$

şeklindeki sınır şartlı diferansiyel denklemlerini çözmek için kullanılabilir. Eğer sınır şartları yukarıdaki denklem ile aynı ise yapılması gereken değişiklik uygun $f(x)$, $g(x)$ ve $q(x)$ fonksiyonlarını COEFF alt programı içinde değiştirmektir.

SADI1 programını değişik N değerleri ile çalıştırarak çözümde yapılan hatanın nasıl değiştiğini gözleyiniz.

2. (6.47)'deki diferansiyel denklem Gauss-Seidel yöntemi kullanılarak da çözülebilir. Bu yöntemin uygulanması SADI2.FOR adlı bir program da yapılmış ve Ek-F'de verilmiştir.

SADI2 programında sınır şartlarının nasıl kullanıldığını inceleyiniz. 10^{-5} mertebesinde bir hata payı elde etmek için kaç iterasyon gerekiyor? N arttıkça gerekli iterasyon sayısı nasıl azalıyor?

3. SADI1 programını SADI12 adı altında

$$y'' + y = 0, \qquad y(0) = 0, \qquad y(1) = 1 \tag{6.49}$$

diferansiyel denklemini çözecek şekilde değiştiriniz. Bu diferansiyel denkleminin analitik çözümü nedir? Sayısal çözümde yapılan hatayı bulmak için analitik çözümü kullanınız. Yapılan hatanın grafiğini çiziniz.

4. SADI1 programını SADI13 adı altında

$$y'' + 2y = -x, \qquad y(0) = 0, \qquad y(1) + y'(1) = 0 \tag{6.50}$$

diferansiyel denklemini çözecek şekilde değiştiriniz. Burada sağ uçta verilen sınır şartının farklı olmasına dikkat ediniz. Verilen diferansiyel denkleminin analitik çözümü

$$y(x) = \frac{\sin(\sqrt{2}x)}{\sin(\sqrt{2}) + \sqrt{2}\cos(\sqrt{2})} - \frac{x}{2} \tag{6.51}$$

ile verilir. Elde ettiğiniz sayısal çözümü analitik çözüm ile karşılaştırınız. Yapılan hatanın grafiğini çiziniz.

5. SADI1 programımı SADI14 adı altında

$$y'' = \cos(x), \qquad y(0) = 0, \qquad y(1) = 1 \tag{6.52}$$

diferansiyel denklemini çözecek şekilde değiştiriniz. Bu diferansiyel denkleminin analitik çözümü nedir? Sayısal çözümde yapılan hatayı bulmak için analitik çözümü kullanınız. Yapılan hatanın grafiğini çiziniz.

6. 3, 4 ve 5'i SADI2.FOR programı için tekrar ediniz.

7. Şimdiye kadar diferansiyel denklemlerinin çözümünde birinci türevler için ikinci mertebe, ikinci türevler için üçüncü mertebe hassas sonlu fark yaklaşımları kullanıldı. Bunların yerine daha yüksek mertebeli hassasiyete sahip sonlu fark açılımlarıda kullanılabilir. Birinci ve ikinci türevler için dördüncü mertebeden hassas sonlu fark açılımlarının aşağıdaki şekilde olması gerektiğini gösteriniz.

$$y'(x) = \frac{y(x-2h) - 8y(x-h) + 8y(x+h) - y(x+2h)}{12h} \tag{6.53}$$

$$y''(x) = \frac{-y(x-2h) + 16y(x-h) - 30y(x) + 16y(x+h) - y(x+2h)}{12h^2} \tag{6.54}$$

Dikkat edilirse bu denklemlerde $y(x-2h)$ değeri x_0 noktasının soluna ve $y(x+2h)$ değeri x_N noktasının sağına taşmaktadır. Bu bazı durumlarda sorun yaratabilir fakat bu sınır noktalarında integrasyon limitlerinin dışına taşmayan ve hala dördüncü mertebe hassas sonlu fark denklemleri yazılabilir. x_0 noktasında birinci ve ikinci türevler için aşağıda verilen sonlu farkların yazılabileceğini gösteriniz.

$$\begin{aligned} y'(x) &= \frac{1}{12h}[-3y(x_0-h) - 10y(x_0) + 18y(x_0+h) - 6y(x_0+2h) + \\ &\qquad + y(x_0+3h)] \end{aligned} \tag{6.55}$$

$$\begin{aligned} y''(x) &= \frac{1}{12h^2}[10y(x_0-h) - 15y(x_0) - 4y(x_0+h) + 14y(x_0+2h) - \\ &\qquad - 6y(x_0+3h) + y(x_0+4h)] \end{aligned} \tag{6.56}$$

Görüldüğü gibi x_0 noktasındaki bu açılımlar artık simetrik değildir. Benzer denklemler x_N noktası içinde yazılabilir.

8. (6.44) denkleminde verilen Schrödinger denkleminde türevler için ikinci yerine dördüncü mertebe hassas sonlu fark denklemleri yazılabilir. Bu durumda elde edilen denklemler simetrik fakat artık üçlü diyagonal değildir. Ancak hesaplanan özdeğerler daha hassastır. (6.44) denklemi için dördüncü mertebe hassas

sonlu farklar kullanıldığında ortaya çıkacak lineer denklem sisteminin

$$\begin{bmatrix} d_1 & b_1 & c_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ b_1 & d_2 & b_2 & c_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ c_1 & b_2 & d_3 & b_3 & c_3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & c_2 & b_3 & d_4 & . & . & . & . & 0 & 0 & 0 \\ 0 & 0 & . & . & . & . & . & . & 0 & 0 & 0 \\ 0 & 0 & 0 & . & . & . & . & . & . & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & . & . & b_{N-3} & c_{N-3} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & . & d_{N-3} & b_{N-2} & c_{N-2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & . & b_{N-2} & d_{N-1} & b_{N-1} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c_{N-2} & b_{N-1} & d_N \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ . \\ . \\ . \\ y_{N-1} \\ y_N \end{bmatrix} = \lambda \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ . \\ . \\ . \\ y_{N-1} \\ y_N \end{bmatrix} \tag{6.57}$$

olacağını gösteriniz. Burada $d_i = (\frac{30}{12h^2} + x_i^2)$ $(i = 1, 2, \ldots, N)$, $b_i = -\frac{16}{12h^2}$ $(i = 1, 2, \ldots, N-1)$, ve $c_i = \frac{1}{12h^2}$ $(i = 1, 2, \ldots, N-2)$ olarak tanımlıdır.

Bu denklemin özdeğer ve özvektörlerini bulan bir program SADI4.FOR adı altında Ek-F'de verilmiştir. Katsayı matrisi hala gerçel ve simetrik fakat üçlü diyagonal olmadığından burada TQLI alt programı doğrudan kullanılamaz. Burada yine [3]'den alınan TRED2 adlı bir alt program kullanılmaktadır. TRED2 gerçel simetrik matrisleri üçlü diyagonal hale getirmekte ve onun çıktıları doğrudan TQLI tarafından kullanılabilmektedir.

6.4 Proje: Schrödinger Denkleminin Farklı Potansiyeller İçin Çözümü

Daha önce (6.43) denkleminde verilen Schrödinger denklemini genel bir $V(x)$ potansiyeli için daha kullanışlı hale getirmek üzere yeniden ölçeklendirelim. Uzunluklar Bohr yarıçapı $a_0 = \hbar^2/2m$ ve enerjiler Rydberg (hidrojen atomunun taban enerji durumu E_0), 1Ry=$me^4/2\hbar^2 = \hbar^2/2ma_0^2$, ile ölçülürse ortaya çıkan denklemi bulalım. Uzunluklar için $y = x/a_0$ kullanılırsa Schrödinger denklemi

$$\begin{aligned} \frac{\hbar^2}{2m}\frac{d^2u}{d(a_0y)^2} + V(y)u(y) &= Eu(y) \\ \frac{\hbar^2}{2ma_0^2}\frac{d^2u}{dy^2} + V(y)u(y) &= Eu(y) \\ E_0\frac{d^2u}{dy^2} + V(y)u(y) &= Eu(y) \end{aligned}$$

olur. $\lambda = E/E_0$ ve $V'(y) = V(y)/E_0$ olmak üzere yukarıdaki son denklem yeniden yazılırsa

$$\frac{d^2u}{dy^2} + V'(y)u(y) = \lambda u(y) \tag{6.58}$$

elde edilir. Bu denklemde enerjiler Rydberg (E_0), uzunluklar Bohr yarıçapı cinsinden ölçülmektedir.

Schrödinegr denklemini V_0 derinlikli, L genişliğindeki simetrik bir kuyu için çözen bir program SADI5.FOR adı altında Ek-F'de verilmiştir.

1. Bu programı detaylı olarak inceleyiniz. Özellikle katsayı matrisinin nasıl hesaplandığına, potansiyel fonksiyonunun nasıl kullanıldığına dikkat ediniz.

2. Programı verilen değerleri kullanarak çalıştırınız. Bulduğunuz enerji öz değerleri anlamlı mı?

3. Bulduğunuz dalga fonksiyonlarını ve potansiyel fonksiyonunu beraber çizdiriniz. Dalga fonksiyonları $\pm\infty$'da sıfıra gidiyor mu? Gitmiyorsa sebep nedir?

4. Elde ettiğiniz dalga fonksiyonları normalize edilmiş mi? Programa başka bir ad altında bu durumu kontrol edecek bir kaç satır ekleyiniz.

5. Elde ettiğiniz dalga fonksiyonları bir birlerine dik mi? Diklik şartının sağlanıp sağlanmadığını (4)'de yazdığınız programa bir kaç satır daha ekleyerek kontrol ediniz.

6. Aynı programı başka bir potansiyel fonksiyonu için kullanmanız gerekirse ne yapmanız gerekir? Bu programı kullanabilmeniz için potansiyel fonksiyonunun simetrik olması gerekir mi? Neden?

6.5 ALIŞTIRMA SORULARI

Bölüm 7

İNTERPOLASYON ve EKSTRAPOLASYON

Bu bölümde interpolasyon ve ekstrapolasyon konuları görülecektir.

7.1 Giriş

Bazı durumlarda sürekli bir değişkene bağlı bir fonksiyonun ancak belirli noktalardaki değeri bilinir. Bu durum çoğu kez deneysel olarak ölçülen değerlerde ortaya çıkar. Sadece belirli noktalarda değilde bütün bir aralıktaki değerlere karşılık gelen fonksiyon değerlerini bilmek istediğimizde ne yapılması gerekir? Bu bağlamda, basit anlamda interpolasyon işlemi tablo halinde değerleri verilen bir değişkenin verilen aralıkta tabloda olmayan bir değerini bulma olarak tanımlanabilir.

Daha genel olarak interpolasyon, bilinmeyen bir $f(x)$ fonksiyonunun x_0, x_1, ..., x_n gibi ayrık noktalarda verilen f_0, f_1, ..., f_n değerlerini kullanarak bu fonksiyonun daha basit ve bilinen bir $p(x)$ fonksiyonu ile ifade edilmesi işlemidir. Bulunan $p(x)$ fonksiyonuna 'interpolasyon fonksiyonu' adı verilir ve polinom, trigonometrik fonksiyon, üslü ifade veya özel bir fonksiyon gibi çeşitli şekillerden birine sahip olabilir.

ınterpolasyon işlemi doğru uydurma teknikleri ile alakalı olsa da aynı değildir. ınterpolasyon işleminde verilen değerlerin hatasız olduğu kabul edilir. Bu nedenle interpolasyon fonksiyonu verilen noktaların tam üzerinden geçer. Ayrıca interpolasyon ile bulunan fonksiyonun, değerleri ayrık noktalarda verilen gerçek fonksiyon (eğer böyle bir fonksiyon varsa) ile aynı değildir. ınterpolasyon fonksiyonu sadece gerçek fonksiyonu (varsa) temsil eden olası bir seçenektir.

ınterpolasyon fonksiyonu olarak çoğunlukla polinomlar veya polinomların oranları kullanılır. Bu durum polinomlar ile türev alma, değer hesaplama gib işlemleri yapmanın kolaylığından kaynaklanır. Fakat verilen değerlerin periyodik olması durumunda trigonometrik fonksiyonların kullanılması daha uygun olur.

Tanım: x_0, x_1, ..., x_n noktalarına karşılık gelen f_0, f_1, ..., f_n değerlerinin verildiğini varsayalım. Bir boyutlu bu problemde interpolasyon temel olarak

$$p(x_i) = f(x_i), \qquad i = 0, 1, \ldots, n \tag{7.1}$$

şartını sağlayan $p(x)$ fonksiyonunu bulma işlemidir. Eğer $p(x)$ bir polinom ise bu durumda probleme polinom enterpolasyonu denir. Bu bölümde sadece polinom şeklindeki interpolasyon fonksiyonları göz önüne alınacaktır.

Verilen değerleri temsil eden gerçek $f(x)$ fonksiyonu (şayet böyle bir fonksiyon varsa) daha basit $p(x)$ fonksiyonu ile değiştirildiğinden, belirli bir hatanın olması kaçınılmazdır. Enterpolasyon fonksiyonu verilen noktalarda kesindir, hata değeri bilinmeyen ara noktalarda yapılan hesaplamada ortaya çıkar.

7.2 Doğrusal İnterpolasyon

Enterpolasyon fonksiyonu olarak birinci dereceden bir polinom kullanılırsa, böyle bir enterpolasyona doğrusal (lineer) enterpolasyon denir. örneğin $[x_0, x_1]$ aralığında verilen değerler $f(x_0) = f_0$ ve $f(x_1) = f_1$ olsun. Enterpolasyon fonksiyonu olarak $p(x) = ax + b$ seçilebilir. $p(x_i) = f(x_i) = f_i$, $(i = 0, 1)$ olması gerektiğinden

$$\begin{aligned} ax_0 + b &= f_0 \\ ax_1 + b &= f_1 \end{aligned} \qquad \begin{matrix} (7.2) \\ (7.3) \end{matrix}$$

denklemlerinin sağlanması gerekir. a ve b için çözüm yapılırsa

$$a = \frac{f_1 - f_0}{x_1 - x_0}, \qquad b = \frac{x_1 f_0 - x_0 f_1}{x_1 - x_0}$$

bulunur. Böylece enterpolasyon polinomu

$$p(x) = ax + b = [\frac{f_1 - f_0}{x_1 - x_0}]x + [\frac{x_1 f_0 - x_0 f_1}{x_1 - x_0}] \tag{7.4}$$

olacaktır. $p(x)$ polinomu istenirse

$$p(x) = [\frac{x_1 - x}{x_1 - x_0}]f_0 + [\frac{x - x_0}{x_1 - x_0}]f_1 \tag{7.5}$$

şeklinde yeniden yazılabilir. $w_0(x) = \frac{x_1 - x}{x_1 - x_0}$ ve $w_1(x) = \frac{x - x_0}{x_1 - x_0}$ olmak üzere enterpolasyon fonksiyonu

$$p(x) = w_0(x)f_0 + w_1(x)f_1 \tag{7.6}$$

olacaktır. Enterpolasyon fonksiyonu f_0 ve f_1'in x'e bağlı ağırlık fonksiyonlarının bir kombinasyonu olduğu buradan görülebilir. Ayrıca $x_0 \leq x \leq x_1$ aralığında olduğu sürece $w_0(x) + w_1(x) = 1$ olduğu kolaylıkla görülebilir.

ÖRNEK **7.1** *Bir $f(x) = e^x$ fonksiyonunun [0.2,0.3] aralığındaki değeri*

x_i	*0.2*	*0.3*
$f(x_i) = f_i$	*1.22140*	*1.34986*

olarak verilmiştir. Doğrusal enterpolasyon yardımı ile $x = 0.26$ noktasındaki $p(0.26)$ değerini bulunuz.

Aranan $p(x)$ polinomu

$$p(x) = [\frac{f_1 - f_0}{x_1 - x_0}]x + [\frac{x_1 f_0 - x_0 f_1}{x_1 - x_0}]$$

olduğundan verilen değerler yerine yazılırsa

$$p(x) = 1.2846x + 0.96448$$

bulunur. Bu durumda $x = 0.26$*'daki* $p(x)$ *değeri* $p(0.26) = 1.2846 \times 0.26 + 0.96448 = 1.29847$ *olur. Bu örnekte enterpole edilen fonksiyonun kendisi bilindiğinden gerçek değeri hesaplayabiliriz:* $f(0.26) = e^{0.26} = 1.29693$*. BUradan yaplan mutlak hata* $|f(x) - p(x)|_{x=0.26} = 1.54 \times 10^{-3}$ *olur.*

7.3 Lagrange Enterpolasyon Formülü

x_0, x_1, ... , x_n x-ekseni üzerinde $(n+1)$ tane ayrık nokta, f_0, f_1, $\cdots$, f_n bu noktalara karşılık gelen bir $f(x)$ fonksiyonunun değerleri olsun. Derecesi n veya daha küçük olan ve

$$p(x_i) = f(x_i), \;\; i = 0, 1, \cdots, n \tag{7.7}$$

koşulunu sağlayan bir $p(x)$ polinomu bulmaya çalışalım. Bu polinom için Lagrange biçimi

$$p(x) = a_0 l_0(c) + a_1 l_1(x) + \cdots + a_n l_n(x) \tag{7.8}$$

şeklinde yazılabilir. Burada $l_i(x)$ fonksiyonları Lagrange polinomları olarak bilinir ve

$$l_k(x) = \prod_{\substack{i = 0 \\ i \neq k}}^{n} \left(\frac{x - x_i}{x_k - x_i}\right), \;\; k = 0, 1, \cdots, n \tag{7.9}$$

olarak tanımlanır. $l_k(x)$ fonksiyonları n lineer terimin çarpımı olduğundan, $n$ci dereceden polinomlardır. Ayrıca $l_k(x)$ fonksiyonları $i \neq k$ için bütün $x = x_i$ noktalarında 0 ve $x = x_k$'da ise 1 olur.

$$l_k(x) = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{if } i \neq k \end{cases} \tag{7.10}$$

Bu son özellikten dolayı

$$p(x_i) = \sum_{k=0}^{n} a_k l_k(x_i) = a_i, \quad i = 0, 1, \cdots, n \tag{7.11}$$

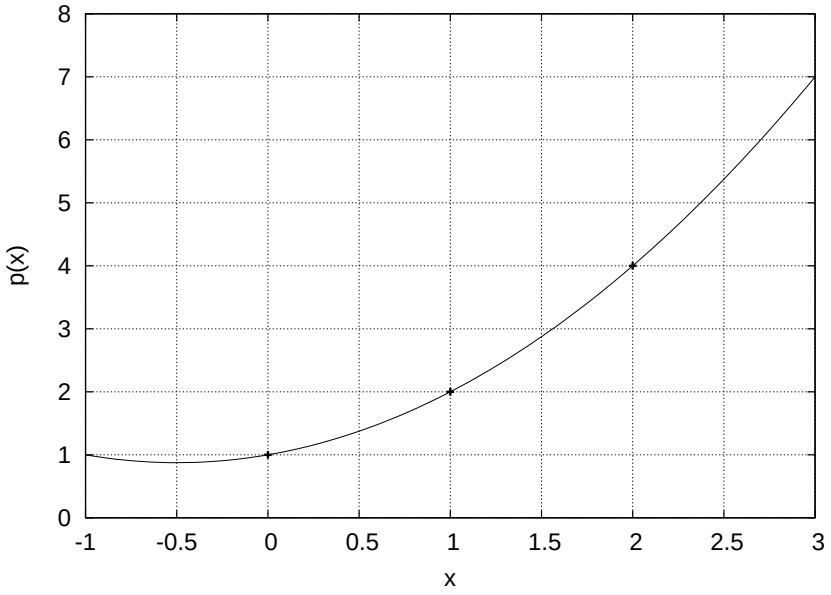
olur. Yani a_0, a_1, $\cdots$, a_n katsayıları polinomun x_0, x_1, ... x_n noktalarındaki değerine eşittir. $p(x_i) = f(x_i)$ olduğundan $a_i = f(x_i)$, $i = 0, 1, \cdots, n$ olacaktır. Sonuç olarak keyfi bir $f(x)$ fonksiyonu için derecesi $\leq n$ olan ve $f(x)$ fonksiyonunu x_0, x_1, ... x_n noktalarında enterpole eden $p(x)$ polinomu

$$p(x) = \sum_{k=0}^{n} f(x_k) l_k(x) \tag{7.12}$$

olarak yazılabilir.

ÖRNEK **7.2** *Aşağıdaki tabloda verilen noktalardan geçen ikinci derece polinomu bulunuz.*

i	*0*	*1*	*2*
x_i	*0*	*1*	*2*
$f(x_i) = f_i$	*1*	*2*	*4*



Şekil 7.1: örnek 7.2'de çözülen enterpolasyon probleminde elde edilen enterpolasyon fonksiyonu ve verilen noktalar aynı anda gösterilmiştir.

çözüm:

$$p(x) = \sum_{i=0}^{2} f(x_i) l_i(x), \quad i = 0, 1, \cdots, 2$$

olduğundan ve $l_i(x)$*'ler*

$$l_k(x) = \prod_{\substack{i=0 \\ i \neq k}}^{2} \left(\frac{x - x_i}{x_k - x_i}\right), \quad k = 0, 1, \cdots, 2$$

şeklinde tanımlandığından önce Lagrange polinomları $l_0(x)$*,* $l_1(x)$ *ve* $l_2(x)$*'i bulalım:*

$$l_0(x) = \frac{(x - x_1)(x - x_2)}{x_0 - x_1)(x_0 - x_2)} = \frac{(x - 1)(x - 2)}{0 - 1)(0 - 2)} = \frac{1}{2}(x - 1)(x - 2)$$

olur. Benzer şekilde diğer Langange polinomları $l_1(x) = -x(x - 2)$ *ve* $l_2(x) = \frac{1}{2}x(x - 1)$ *olacaktır. Enterpolasyon polinomu yazılırsa*

$$p(x) = l_0(x)f_0 + l_1(x)f_1 + l_2(x)f_2 = \frac{1}{2}x^2 + \frac{1}{2}x + 1$$

elde edilir.

7.4 Newton Enterpolasyonu ve Bölümlü Farklar

Newton enterpolasyonu Lagrange enterpolasyonun yöntemine göre aşağıda belirtilen olgular açısından biraz daha avatajlıdır.

- Newton enterpolasyonunda, elde edilen bir enterpolasyon polinomunun derecesi kolaylıkla arttırılabilir. Bu işlem o ana kadar bulunan polinom doğrudan kullanılarak yapıldığından yeniden başlayarak polinom hesaplamaya gerek kalmaz. Polinomun derecesi ne kadar arttırılacaksa, o sayıda yeni enterpolasyon noktalarının eklenmesi gerekir.
- Newton enterpolasyon yöntemi programlamaya daha uygundur.

Newton enterpolasyonunda aranan $n$ci dereceden polinom

$$\begin{aligned} p(x) &= a_0 + a_1(x-x_0) + a_2(x-x_0)(x-x_1) \\ &\quad + a_3(x-x_0)(x-x_1)(x-x_2) + \cdots \\ &\quad + a_n(x-x_0)(x-x_1)\cdots(x-x_n) \\ &= \sum_{i=0}^{n} a_i \prod_{j=0}^{i-1}(x-x_j) \end{aligned} \tag{7.13}$$

şeklinde yazılır. a_i'ler sabit olup

$$f(x_i) = p(x_i), \quad i = 0, 1, 2, \cdots, n \tag{7.14}$$

şartı kullanılarak bulunur. Bu şartı $x = x_0$, $x = x_1$, ... ve $x = x_n$ noktalarında ayrı ayrı kullanalım. $x = x_0$ için $p(x_0) = a_0 = f(x_0)$ olduğundan aranan ilk sabit $a_0 = f(x_0)$ olur. a_1'i bulmak için $x = x_1$'deki şartı kullanalım. $p(x_1) = a_0 + a_1(x_1 - x_0) = f(x_1)$ denkleminden a_1 çekilir ve $a_0 = f(x_0)$ değeri kullanılırsa

$$a_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \tag{7.15}$$

elde edilir. Bu denkleme a_1 için bölümlü fark denklemi denir ve

$$a_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = f[x_0, x_1] \tag{7.16}$$

olarak tanımlanır. Burada $f[x_0, x_1]$ birinci bölümlü fark olarak bilinir. Yukarıdaki işlemler $x = x_2$ için tekrar edilirse

$$a_2 = \frac{1}{(x_2 - x_0)} \left[\frac{f(x_2) - f(x_1)}{(x_2 - x_1)} - \frac{f(x_1) - f(x_0)}{(x_1 - x_0)} \right] \tag{7.17}$$

bulunur. Dikkat edilirse son denklemde köşeli parantez içindeki terimin ikincisi $f[x_0, x_1]$ ile aynıdır, birinci terim ise benzerlikten faydalanarak $f[x_1, x_2]$ olarak yazılabilir. Bu nedenle a_2 terimi

$$a_2 = \frac{f[x_1, x_2] - f[x_0, x_1]}{(x_2 - x_0)} = f[x_0, x_1, x_2] \tag{7.18}$$

olarak yazılabilir. $f[x_0, x_1, x_2]$ ikinci bölümlü fark olarak adlandırılır. a_i katsayılarını bulma işlemine bu şekilde devam edilirse elde edilen sonuç özet olarak

$$\begin{aligned} a_0 &= f(x_0) = f[x_0] \\ a_1 &= \frac{f(x_1) - f(x_0)}{x_1 - x_0} = f[x_0, x_1] \\ a_2 &= \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} = f[x_0, x_1, x_2] \end{aligned} \tag{7.19}$$

$$\vdots = \vdots$$

$$a_k = \frac{f[x_1, \cdots, x_k] - f[x_0, \cdots, x_{k-1}]}{x_k - x_0} = f[x_0, x_1, \cdots, x_k] \tag{7.20}$$

olarak yazılabilir. $f[x_0, x_1, \cdots, x_k]$ $k$nci bölümlü fark olarak bilinir. Böylece Newton enterpolasyon polinomu

$$p(x) = \sum_{i=0}^{n} a_i \prod_{j=0}^{i-1}(x - x_j) = \sum_{i=0}^{n} f[x_0, \cdots, x_i] \prod_{j=0}^{i-1}(x - x_j) \tag{7.21}$$

olarak yazılabilir.

(7.13)'de verilen Newton enterpolasyon polinomunu sayısal olarak hesaplarken, polinomda bazı terimlerin polinomun bütün terimlerinde çarpan olarak geçtiği gerçeği göz önüne alınırsa daha kolay hesaplanabilir. örneğin $(x-x_0)$ terimi a_1, a_2, ..., a_n katsayılarının hepsinde çarpan olarak geçmektedir. Bu nedenle bu terimi paranteze almak gereksiz yere yapılacak bir çok çarpma işleminden kaçınmamızı sağlar:

$$\begin{aligned} p(x) &= a_0 + (x - x_0)[a_1 + a_2(x - x_1) + a_3(x - x_1)(x - x_2) + \cdots \\ &\quad + a_n(x - x_1)(x - x_2)\cdots(x - x_n)] \end{aligned}$$

Benzer şekilde $(x - x_1)$ terimi sadece a_2, a_3, ..., a_n katsayılarında geçtiğinden bu terim de paranteze alınabilir:

$$\begin{aligned} p(x) &= a_0 + (x - x_0)[a_1 + (x - x_1)[a_2 + a_3(x - x_2) + \cdots \\ &\quad + a_n(x - x_2)\cdots(x - x_n)]] \end{aligned}$$

Böylece iç içe geçmiş bir çok çarpma işleminden sonra Newton enterpolasyon polinomu bulunabilir. Bu çarpma işlemini yapacak algoritma aşğıda verilmiştir.

Algoritma 7.1 *(7.13)'de verilen Newton enterpolasyon formülünde geçen ve a_0, a_1, ..., a_n'den oluşan $(n + 1)$ tane katsayının ve x_0, x_1, ..., x_n noktaları ile beraber bir x değerinin verilmesi durumunda $p(x)$ polinomu hesaplanabilir.*

```
        b(n)= a(n)

        do 10 i=n-1,0,-1
        b(i)= a(i) + b(i+1)*(x-x(i))
10      continue

        p(x)= b(0)
```

Newton enterpolasyonunu kullanırken bir bölümlü fark tablosu oluşturmak faydalı olabilir. Tablo 7.1'de x_0, x_1, x_2, x_3 ve x_4 gibi 5 ayrı noktada verilen $f(x_0)$, $f(x_1)$, $f(x_2)$ $f(x_3)$ ve $f(x_4)$ değerlerini kullanarak elde edilecek 4. derece bir polinomun katsayılarını bulmak için oluşturulan tablo gösterilmiştir. Bu tablonun ilk iki kolonu verilen değerlerdir. Diğer kolonlarda bulunan değerler ise verilenlerin kullanılması ile hesaplanmış olanlardır. üçüncü kolonun birinci terimi, ikinci kolonun ilk iki terimi kullanılarak bulunmuştur. Bu durum ok işaretleri ile gösterilmeye çalışılmıştır. Benzer şekilde üçüncü kolonun ikinci terimi, ikinci kolonun ikinci ve üçüncü terimleri kullanılarak bulunur ve işleme diğer kolonlar içinde benzer şekilde devam edilir. Aranan katsayıları veren terimler her kolonun en üst terimleridir. Diğerleri ara işlemlerde gerektiğinden hesaplanmaktadır.

Algoritma 7.2 *$n + 1$ nokta x_0, x_1, ..., x_n ve bunlara karşılık gelen $n + 1$ fonksiyon değeri $d_0 = f(x_0)$, $d_1 = f(x_1)$, ..., $d_n = f(x_n)$ verilmiş olsun.*

Tablo 7.1: Newton enterpolasyonunda tablo oluşturma

x_i	$f[\,]$	$f[\,,\,]$	$f[\,,\,,\,]$	$f[\,,\,,\,,\,]$	$f[\,,\,,\,,\,,\,]$
x_0	$f[x_0]\searrow$				
		$f[x_0,x_1]\searrow$			
x_1	$f[x_1]\nearrow\searrow$		$f[x_0,x_1,x_2]\searrow$		
		$f[x_1,x_2]\nearrow\searrow$		$f[x_0,x_1,x_2,x_3]\searrow$	
x_2	$f[x_2]\nearrow\searrow$		$f[x_1,x_2,x_3]\nearrow\searrow$		$f[x_0,x_1,x_2,x_3,x_4]$
		$f[x_2,x_3]\nearrow\searrow$		$f[x_1,x_2,x_3,x_4]\nearrow$	
x_3	$f[x_3]\nearrow\searrow$		$f[x_2,x_3,x_4]\nearrow$		
		$f[x_3,x_4]\nearrow$			
x_4	$f[x_4]\nearrow$				

```
do k=1,n
     do i=0,n-k
     d(i)= (d(i+1)-d(i))/(x(i+k)-x(i))
     end do
end do
```

7.5 Labaratuar Saati V

7.6 ALIŞTIRMA SORULARI

Bölüm 8

DOĞRU UYDURMA YÖNTEMLERİ

Bu bölümde lineer denklem sistemlerinin çözümü konusunda bazı ön bilgiler verilecektir. Gauss yoketme yöntemi ve temel olarak ona dayalı geliştirilen yöntemlerden bahsedilecektir.

8.1 Giriş

8.2 Labaratuar Saati V

8.3 ALIŞTIRMA SORULARI

Ek A

FORTRAN Komutlarının Kısa bir Tekrarı

Bu kısa hatırlatma ekinde herhangi bir FORTRAN programlama dili dersinde görülen komut ve özelliklerin tekrarı yapılmıştır. Daha fazla detay gerektiğinde bir FORTRAN kitabına başvurulması gerekir [6].

A.1 Karakter Sayısı

FORTRAN programa dilinde 49 tane karakter kullanılır. Bunlardan 26 tanesi latin alfabesinin A'dan Z'ye kadar olan harfleri ve 10 tanesi 0'dan 9'a kadar olan rakamlardır. Geriye kalan 12 sembol çoğunlukla [ ] ile temsil edilen boşluk ve şu karakterlerden oluşur: + - * / = ( ) $. , ' : Alfabe harfleri ve rakamlardan oluşan karakterler alfasayısal karakterler olarak adlandırılır.

A.2 Değişkenler

Değişken, bilgisayarda belirli bir hafıza birimine verilen sembolik bir isimdir. Bir alfabe harfi ile başlaması gerekir. Diğerleri alfasayısal karakterlerden oluşabilir ve bir değişken ismi en çok 6 karakterden meydana gelir. Değişkenler tamsayı veya kesirli olabilir. Başka türlü belirtilmediği takdirde I, J, K, L, M ve N harfleri ile başlayan değişkenler tam sayı olarak kabul edilir. örneğin N2, MAX, I89, KAZ tam sayı değişkenlerdir. A, B, ..., H, O, P, ..., Z ile başlayan değişkenler ise başka türlü belirtilmediği taktirde gerçel (reel) kabul edilirler. X1, SAZ, F1, P2W reel değişkenlere örnektir.

A.3 Sabitler

Bir FORTRAN programının çalışması sırasında değişmeyen tam sayı veya reel sabitlerdir. Bunların dışında bir sabit sanal (kompleks), karakter veya mantıksal olabilir. 9, 11, 32 476 sabit tam sayılara, 3.15, 4.2, 0.0032 reel sayılara örnektir. Reel sayılar exponansiyel temsil ile de gösterilebilir. örnegın 0.0032 yerine 3.2E-3 yazılabilir. Sanal sabitler ise (a,b) şeklinde temsil edilirler. Burada a sayının gerçek, b ise sanal kısmını gösterir. örneğin $2.0 + 7.0i$ kompleks sayısı (2.0,7.0) şeklinde gösterilir. Mantıksal sabitler iki tanedir: .TRUE. ve .FALSE.

Son olarak karakter sabitler 'Karakuyu 1341' örneğinde olduğu gibi apostroflar arasında yazılarak gösterilirler

A.4 Tanımlama Komutları

Bir programda kullanılan değişken veya sabitlerin tiplerinin (reel, tamsayı vb.) açık olarak belirtilmesinde fayda vadrır. Eğer sabit veya değişkenler onların kabul edilen tiplerinden farklı tipleri temsil ediyorlarsa mutlaka belirtilmesi gerekir. örneğin IZ4 değişkeni bir programda reel bir sayıyı temsil ediyorsa bunun mutlaka belirtilmesi gerekir. Tanımlama komutları aşağıda birer örnek ile gösterilmiştir.

```
INTEGER A, IC, KZ, BIZ
REAL KL, RA, YA
REAL*8 PR, IP, SX
COMPLEX J, ID, X2
COMPLEX*16 Z, IH, DE, E
DOUBLE PRECISION AZ, IDA, X2Y
LOGICAL K, GA, DX
```

Ayrıca boyutları olan bir değişken varsa onun tanımlanması gerekir. Aşağıda verilen örnekleri göz önüne alalım.

```
DIMENSION A(5), B(0:7), C(-4:5), D(3,4), E(0:5,0:6)
```

A, B ve C değişkenleri bir boyutlu, D ve E ise iki boyutlu değişkenlerdir. A'yı oluşturan elemanlar A(1), A(2),... A(5)'dir. B'nin 8 elemanı vardır: B(0), B(1), ..., B(7). C'nin C(-4),C(-3),..., C(0), ..., C(5) olmak üzere 10 tane elemanı vardır. D'nin toplam 3×4=12 elemanı, E'nin ise 6×7=42 elemanı vardır.

Bazı değişken veya sabitlerin değerleri programın başlangıcında atanmak isteniyorsa DATA komutu kullanılabilir.

```
DATA A,B,C,IC/10.,-2.E-3,1.0,5/
DATA A/10./B/-2.E-3ç/1.0ıC/5/
DATA ADSOYAD/'METIN OZDEMIR'/
DATA (X(I),I=-1,5) /-1.,2.,5*4./
```

Yukarıdaki birinci ve ikinci satır tamamen eşdeğerdir. üçüncü satır karakter tipi olan bir değişkene aldığı değeri atamaktadır. Son satır ise birden fazla elemanı olan X vektörünün değerlerini atamaktadır. Bu satırda geçen '5*4', '4'ün beş defa kullanılacağını göstermektedir. Yani X(1)=X(2)=...=X(5)=4 anlamına gelir.

Aynı parametre veya değişken bir veya birden fazla alt program tarafından kullanılıyorsa, bu değerler COMMON komutu da kullanılarak gereken yerlere transfer edilebilir. Bunun adlı ve adsız (veya boş) olan iki çeşidi vardır. Eğer adsız COMMON kullanılırsa, bir programda sadece bir COMMON komutu kullanılabilir. Adlı COMMON kullanılırsa programcı istediği kadar COMMON komutu kullanabilir. Aşağıda birinci satır adsız, diğerleri ise adlı COMMON komutlarının kullanımına örneklerdir.

```
COMMON A,B(5),C(-1:3,0:8)
COMMON/PAR1/ D,E(3),F(10)
COMMON/PAR2/ HBAR,QE,PI,KBT
COMMON/PAR3/ X(NMAX),Y(NP)
```

A.5 Aritmetiksel Komutlar ve Matematiksel Fonksiyonlar

Aritmetiksel bir komut program derleyicisine aşağıdaki beş işlemciden herhangi birini gerektiren bir işlem yapmasını sağlar. Beş aritmetik işlemci:

+	toplama
-	çıkarma
*	çarpma
/	bölme
**	üs alma

komutlarından oluşur. Eğer parantezler kullanılarak özellikle tanımlanmamışsa aşağıda verilen öncelik sırası geçerlidir.

1. üs alma
2. çarpma ve bölme
3. toplama ve çıkarma

örneğin komut satırına yazılan A**BÇ*D-E ifadesi derleyici tarafından $\frac{A^B}{C}D - E$ şeklinde yorumlanır. Aşağıda matematiksel işlemlere örnekler verilmiştir.

```
I=I+1
X=A*X+X
B(I)= A(J)-B(K)**2ç(M)
A= 2.*SIN(X) + 1.5
```

Burada SIN(X) FORTRAN derleyicisinde zaten tanımlanmış bir fonksiyondur. Kullanıcının ayrıca bir tanım yapmasına gerek yoktur. Aşağıdaki listede FORTRAN'da varolan fonksiyonlar ve özellikleri verilmiştir.

[eklenecek]

A.6 Kontrol Devri

Programın kontrolü bir satırdan bazı şartların sağlanması durumunda başka bir satıra aktarılabilir. Bu durum programın akış mantığında bir değişiklik yaratır. Kontrol devri iki şekilde yapılabilir: şartsız veya şartlı devir. şartsız devirde komut

```
GO TO n
```

şeklindedir. Burada n kontrolun devredileceği satır numarasına tekabül eder. Aşağıdaki örnek bu kullanımı göstermektedir. Dikkat edilirse burada programın kontrolu 10 numaralı satıra hiç bir şart aranmadan devredilmektedir.

```
      GO TO 10
           ⋮
10      X=2.*X + B
```

şartlı kontrol devri ise iki şekilde yapılabilir. Bunların birincisinde komut satırı

$$\text{GO TO } (n_1, n_2, \ldots, n_k), \text{ I}$$

şeklindedir. n_i'ler program işleyişinin devredileceği satır numarasını gösterir ve I sadece I=1,2,. . . , k değerlerini alabilir. örneğin

$$\text{GO TO (10, 20, 90, 10), IC}$$

komutu IC=1 veya 4 olduğunda 10 nolu, IC=2 olduğunda 20 nolu ve IC=3 olduğunda 90 nolu satıra gidileceğini ifade eder. Yani IC=1 veya 4 olması durumunda 'GO TO 10' komutu varmış gibi işlem yapılır. Diğer durumlar içinde benzer kurallar geçerlidir. Diğer bir şartlı kontrol devri ise IF komutu ile yapılır. IF komutunun argümanı aritmetik veya mantıksal olabilir. Aritmetik argümanlı IF komutu

$$\text{IF(} terim \text{) } n_1, n_2, n_3$$

şeklindedir ve eğer $terim < 0$ ise n_1, $terim = 0$ ise n_2 ve $terim > 0$ ise n_3 nolu satıra gidileceğini ifade eder. Mantıksal IF komutu

$$\text{IF(mantıksal terim) komut X}$$

şeklinde olup mantıksal terimin doğru olması durumunda X komutunun yerine getirileceğini gösterir. örneğin

```
IF(B.LE.E) GO TO 32
```

komut satırı B'nin E'den küçük veya eşit olması durumunda 32 nolu satıra gidileceğini gösterir. Benzer şekilde

```
IF(A.LT.-1.AND.B.GT.3) X=X*X-1.0
```

komutu A'nın -1'den küçük olması ve B'nin 3'den büyük olması durumunda X'in değerinin X'in karesinden 1 çıkartılarak elde edilen sayıya eşitleneceğini gösterir.

IF komutlarında kullanılan karşılaştırma komutları aşağıda verilmiştir.

.LT.	küçük
.LE.	küçük veya eşit
.GT.	büyük
.GE.	büyük veya eşit
.EQ.	eşit
.NE.	eşit değil

Mantıksal işlemciler ise

.AND.	ve
.OR.	veya
.NOT.	değil

ile verilir.

Bir diğer IF komutu ise 'IF(...) THEN ... ELSE ... ENDIF' komutudur. Genel kullanım şeklini bir örnek ile gösterelim:

```
        IF(N.EQ.1) THEN
        X=X + Y
        Z= SIN(X)
        ELSE IF(N.EQ.2) THEN
        X= X * Y
        H= F(X)
        ELSE
        X= X * X
        CALL TOPLA(X,Y,Z)
        ENDIF
```

ENDIF komutu blok halinde yazılan IF komutunun bittiğini belirtir. Burada N 1'e eşit ise ondan sonra gelen iki satır işlenir ve IF döngüsünden çıkılır. N 2'ye eşit ise o satırdan sonra gelen iki satır gözönüne alınır. Eğer N 1 veya 2'ye eşit değilse 'ELSE'den sonra gelen satırlar işlem görür ve döngüden çıkılır.

A.7 DO Döngüleri

DO döngüleri FORTRAN programlarında tekrar eden işlemleri yapmak için kullanılır. Genel kullanım şekli aşağıda verildiği gibidir.

```
        DO n I=NILK, NSON, NARTI

        (komutlar)
n       CONTINUE
```

Burada n DO döngüsünün numarasını belirtir. I herhangi bir tamsayı değişkenidir. NILK I'nın alacağı ilk değeri, NSON, I'nın alacağı son değeri ve NARTI ise I'nın değerinin kaçar kaçar arttırılacağını tanımlar. Aşağıdaki örnekte I'nın 1'den 20'ye kadar 3'er 3'er arttırılacağını gösterir.

```
        X=0.0
        DO 10 I=1,20,3
        X= X + SIN(X)
10      CONTINUE
```

A.8 Girdi-Çıktı Komutları

Bir programın çalışması için gereken veriler ekrandan doğrudan girilebileceği gibi verilerin daha önce yazıldığı bir dosyadan da okutmak mümkündür. Benzer şekilde programın ürettiği veriler ekrana yazılabileceği gibi herhangi bir dosyaya yazılarak uzun süreli olarak korunabilir. Bir dosyadan veri okumak veya dosyaya veri yazmak için o dosyayı açmamız gerekir. Bu OPEN komutu ile yapılır. Genel kullanım biçimi

```
      OPEN(n, FILE='DOSYAADI', STATUS='UNKNOWN')
```

şeklindedir. Burada n bir tam sayı olup birim numarası olarak bilinir. DOSYAADI, açılması istenen dosyanın adıdır ve daha önceden 'CHARACTER'

tipi değişken olduğu belirtilmelidir. 'STATUS' dosyanın statüsünü belirtir. Statü 'OLD', 'NEW' veya 'UNKNOWN' olabilir. Statünün 'OLD' olması daha önceden varolan bir dosyanın açıldığı anlamına gelir. 'NEW' yeni bir dosyanın açıldığı, UNKNOWN ise açılmakta olan dosyanın statüsünün bilinmediği anlamına gelir. 'OPEN' ile açılan bir dosyadan veri okunarak veya dosyaya veri yazılarak dosyanın belirli bir satırına kadar gelinebilir. Dosyanın herhangi bir sebeple başına dönülmesi istenirse 'REWIND(n)' komutu kullanılır. Burada n geriye sarılması istenen ve daha önce 'OPEN' komutu ile açılan dosyanın birim numarasıdır. 'OPEN' komutu ile açılan bir dosya gerektiğinde, n açılan dosyanın birim numarası olmak üzere 'CLOSE(n)' komutu ile kapatılır.

ıki önemli girdi ve çıktı komutu READ ve WRITE komutlarıdır. Birincisi verileri ekrandan veya bir veri dosyasından okumaya, ikincisi ise elde edilen verileri ekrana veya bir dosyaya yazmaya yarar. Okuma ve yazma işlemleri formatlı olabileceği gibi formatsızda olabilir. Formatlı okuma veya yazmada çıktıların nasıl okunacağı ve yazılacağı net olarak tanımlanır. Formatsız okuma ve yazma komutları

```
READ(*,*) liste
WRITE(*,*) liste
```

şeklindedir. Liste burada okunacak/yazılacak değişkenleri temsil eder. Parantez içindeki birinci yıldız okumanın/yazmanın ekrandan/ekrana olacağını belirtir. Eğer yıldız yerine bir tam sayı kullanılırsa daha önce OPEN komutu ile açılan aynı sayı numaralı dosyadan/dosyaya okuma/yazma yapılacağı anlamına gelir. ıkinci yıldız yerine bir tam sayı kullanılırsa okuma/yazmanın bu numaralı formata göre yapılacağı anlamına gelir. Aşağıda incelemeniz için bir kaç örnek verilmiştir.

```
      READ(*,*) A, N
      WRITE(*,*)' X=',X,' Y=',Y
      READ(12,*) C,D,M
      WRITE(15,100) K,(A(I),I=1,3)
100   FORMAT(1X,'K=', I5, 1X,'A=',3(2X,E12.5))
```

A.9 Alt Programlar

Bir programda yapılacak işler parçalara ayrılarak her bir parçayı yapacak alt programlar yazılabilir. Her alt program kendi içinde bir bütünlük oluşturur. Alt programların en temel olan iki tanesi FUNCTION ve SUBROUTINE alt programlarıdır. FUNCTION alt programının bir adı ve girdi olarak kullanılacak argümanları olur. Bu alt programın sadece bir çıktısı vardır ve programın kendi adı ile aynıdır. Aşağıda verilen örnek FUNCTION alt programı $f(x, y) = x^2 + 2xy - x * \sin((x + y)/2x)$ fonksiyonunun değerini verilen x ve y değerleri için hesaplar ve değerini F'ye kaydeder.

```
FUNCTION F(X,Y)
Z= (X+Y)/X/2.
F= X*(X + 2.*Y-SIN(Z))
RETURN
END
```

Ek B

GNUPLOT Grafik Programı

GNUPLOT internet üzerinden bir çok siteden ücretsiz olarak alınabilen bir grafik programıdır. Programın telif ve kullanım hakları için 'Giriş' (introduction) kısmına bakınız.

Programın ingilizce iyi bir 'yardım' (help) bölümü bulunmaktadır. Burada yeni başlayanlar için çok kullanılan bazı özellikler tanıtılmaya çalışılmıştır.

GNUPLOT kullanarak hem iki boyutlu (x, y) hemde üç boyutlu (x, y, z) grafik çizilebilir. Bunların komutları sırası ile 'plot' ve 'splot'tır.

plot ve **splot** komutlarını kullanarak hem fonksiyonların grafikleri hemde daha önceden deneysel veya sayısal olarak üretilen ve bir dosyaya kaydedilen verilerin grafikleri çizilebilir. Dosyaya kaydedilen verilerin iki veya üç boyutlu çizim için uygun olacak şekilde kolonlar halinde olması gerekir. İki boyutlu grafik çizimi için birden fazla veri grubu varsa bunlar arasında iki satır boş bırakılarak bir birlerinden ayrılmalıdır.

Aşağıda GNUPLOT komutlarının kullanımına ilişkin örnekler verilmiştir. GNUPLOT programı açılarak aşağıda verilenlerin hepsi denenerek nasıl kullanılacağı ögrenilebilir.

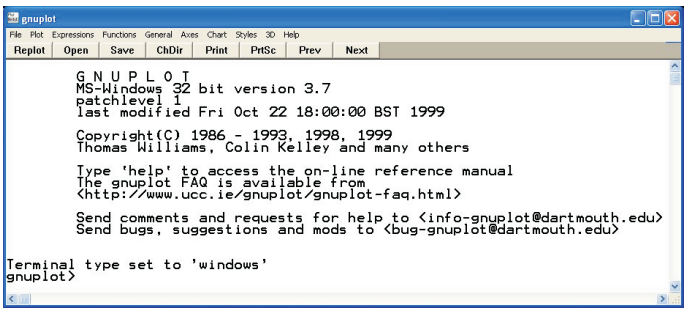
B.1 İki Boyutlu Grafik Çizimi

Gnuplot grafik programını çalıştırıldığında açılım ekranı Şekil B.1'de gösterildiği gibidir. Bazı fonksiyonları üst tarafta gösterilen menüden seçilebilir. Aynı komutlar komut satırına yazılarak çalıştırılabilir.

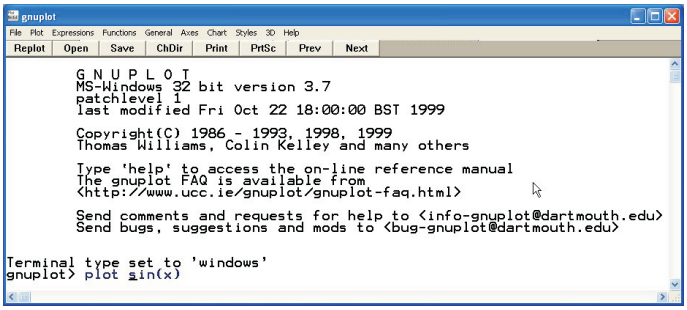
Örnek olarak sinüs fonksiyonunun grafiğini çizmek istediğimizi varsayalım. Bunun için yapılması gereken **plot** komutunu aşağıda gösterildiği gibi kullanmaktır.

```
gnuplot>plot sin(x)
```

Bu komut Şekil B.2'de gösterildiği gibi girilir ve 'Enter' tuşuna basılırsa sinüs fonksiyonunun Şekil B.3'de gösterilen grafiği elde edilir. Çizilen grafikte x-ekseni $[-10, 10]$ aralığında seçilmiştir. Başka türlü belirtilmezse fonksiyon çizimleri için GNUPLOT hep bu aralığı kullanarak çizim yapar. Veri dosyaları kullanarak çizim yapılıyorsa bu durumda verilerin x ve y-eksenlerine karşılık gelen en küçük ve en büyük değerleri arasında kalan noktalar için çizim yapılır. Ayrıca grafik ekranının sağ üst köşesinde komut satırında girilen fonksiyon ve onun çizimine karşılık gelen çizgi rengi gösterilir.



Şekil B.1: **gnuplot** grafik programının açılım ekranı



Şekil B.2: **gnuplot** grafik programı ile $\sin(x)$ fonksiyonunun grafiğini çizmek için girilmesi gereken komut.

Bu grafik fazla detayın gerekmediği ve herhangi bir sunum için kullanılmayacağı durumlar için uygundur. Grafiği bir kaç komut kullanarak daha düzgün hale getirmek mümkündür. Örneğin sağ üst satırdaki fonksiyonu gösteren etiket

```
gnuplot>unset key
```

komutu ile kaldırılabilir. Grafik eksenlerini adlandırmak için 'set xlabel' veya 'set ylabel' komutları kullanılır.

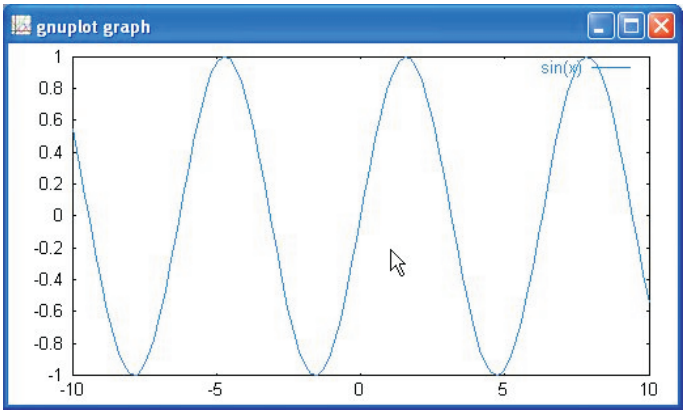
```
gnuplot>set xlabel "x (cm)"
gnuplot>set ylabel "f(x)"
```

komutları x-eksenine "x (cm)" ve y-eksenine "f(x)" adının verilmesini sağlar. Grafiği daha iyi takip etmek için eksenlera paralel çizilen çizgilerden oluşan ızgara konulabilir. Bunun için

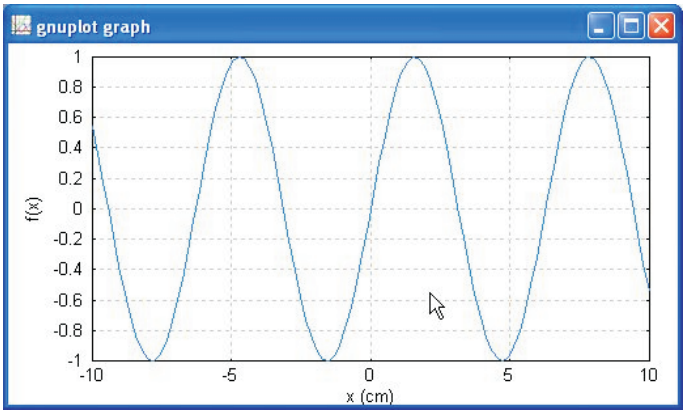
```
gnuplot>set grid
```

komutunu kullanmak yeterlidir. Bu komutların kullanılması ile Şekil B.3'de gösterilen grafik Şekil B.4'de gösterilen grafiğe dönüşür.

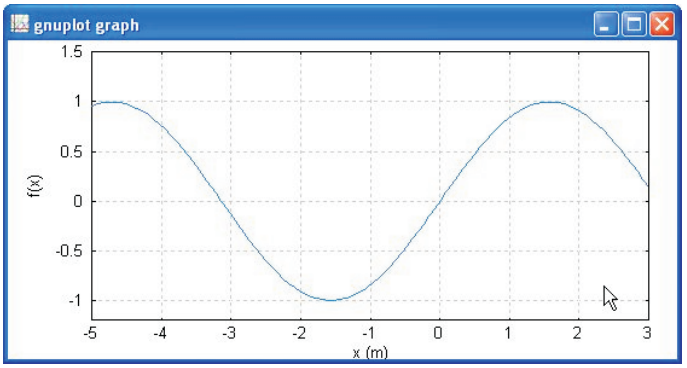
Grafın çizildiği x ve y aralıklarını değiştirmek için 'plot' komutundan sonra yazılan köşeli parantezlerin içine [][] sırası ile x ve y eksenlerinin alt ve üst limitleri yazılır. Limit değerleri ':' işareti ile ayrılır. Herhangi bir eksen için limit ayarı yapmak gerekmiyorsa o eksene karşılık gelen köşeli parantezin içi



Şekil B.3: **gnuplot** grafik programı ile çizilen $\sin(x)$ fonksiyonunun grafiği.



Şekil B.4: Şekil B.3'de gösterilen grafiğin çeşitli komutlar kullanılarak yeniden düzenlenişi.



Şekil B.5: Şekil B.4'te gösterilen grafiğin sınırlarının yeniden düzenlenmiş hali.

boş bırakılır. Örneğin biraz önce verilen örnekte x ekseni $[-5, 3]$ ve y ekseni $[-1.2, 1.5]$ aralığı ile sınırlanmak istenirse, aşağıda verilen komutun girilmesi yeterlidir.

```
gnuplot> plot [-5:3] [-1.2:1.5] sin(x)
```

Elde edilen sonuç Şekil B.5'te gösterilmiştir.

Birden fazla fonksiyonunun aynı anda grafiği çizilmek istenirse bu durumda komut satırına fonksiyonlar virgül ile ayrılarak yazılır. Aşağıdaki örnek $\cos(2x)$, x^2 ve $x * \exp(-0.5x)$ fonksiyonlarının grafiklerini aynı anda çizer.

```
gnuplot> plot [-1:2] [-1.2:1.5] cos(2.*x),x*x,x*exp(-0.5*x)
```

Bu komut sonucunda çizilen grafik Şekil B.6'da gösterilmiştir. Bu grafiğin çiziminden önce ayrıca 'set key dafault' komutu girilmiştir. Bu daha önce girilen 'unset key' komutunun etkisini yok eder. Böylece Şekil B.6'da her fonksiyonun grafiği sağ üst köşede bir etiket ile belirtilmiştir.

B.2 Fonksiyon Tanımlama ve Parametrelere Değer Atama

Belirli bir parametreye (veya parametrelere) bağlı fonksiyonları parametrenin çeşitli değerleri için veya çok uzun bir fonksiyonu bir kaç defa değişik parametre değerleri için çizdirmek gerektiğinde aynı fonksiyonu defalarca yazmak zor olabilir. Bunun yerine fonksiyon komut satırında tanımlanarak her defasında verilen isim kullanılarak grafiği çizdirilebilir. En son gösterilen örnekte üç tane fonksiyon vardı: $\cos(2x)$, x^2 ve $xe^{-0.5x}$. Bunlara sırası ile $f(x)$, $g(x)$ ve $h(x)$ diyelim. Bu tanımlar GNUPLOT satır komutunda aşağıdaki gibi yapılır:

```
gnuplot> f(x)= cos(2.*x)
gnuplot> g(x)= x*x
gnuplot> h(x)= x*exp(-0.5*x)
```

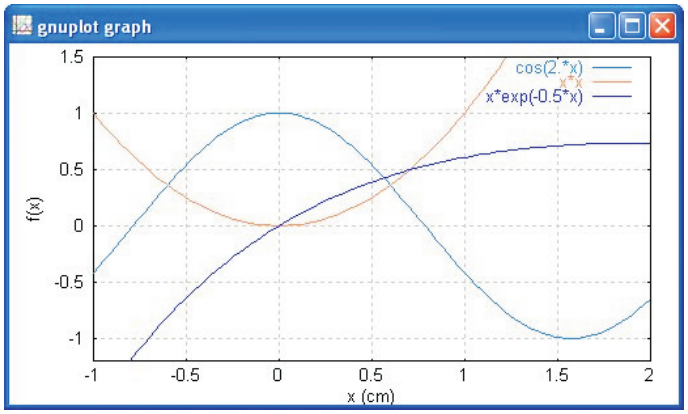
Bu fonksiyonların tek tek veya bir kaçının birden fonksiyonları çizilmek istenirse yapılması gereken fonksiyon adlarının satır komutuna yazılmasıdır. Örneğin Sekil B.6'da görülen grafiği elde edmek için satır komutuna yazılması gerekenler aşağıda gösterildiği gibidir.

```
gnuplot> plot [-1:2] [-1.2:1.5] f(x),g(x),h(x)
```

Bu komut yazılıp grafik çizildiğinde Şekil B.6 ile bir tek farkı olacaktır. Bunu da siz bulmaya çalışınız.

Bazı fonksiyonlar bir veya daha çok parametreye bağlı olabilir. Bu durumda fonksiyon tanımı parametrelere bağlı yapılabilir. Parametrelerin değeri komut satırında verilebileceği gibi fonksiyonun argümanı olarakda verilebilir. Örneğin $f(x) = ae^{-bx}$ fonksiyonu a ve b parametrelerine bağlıdır. Bu parametrelerin çeşitli değerleri için grafik çizimi yapılmak istenirse iki yol izlenebilir. Birincisi fonksiyon tanımında parametreleri de argüman olarak belirtmek ve çizim esnasında değerlerini vermektir. Fonksiyonu tanımlamak için aşağıdaki komut yazılır:

```
gnuplot> f(x,a,b)= a*exp(-b*x)
```



Şekil B.6: **gnuplot** grafik programı ile birden fazla fonksiyonun grafiği aynı anda gösterilebilir.

Bu fonksiyonun grafiğini $a = 0.5$, $b = 1.2$ için çizmek gerekirse komut satırına

```
gnuplot> plot f(x,0.5,1.2)
```

yazmak yeterlidir. Sonuç Şekil B.7'de gösterilmiştir.

B.3 Üç Boyutlu Grafik Çizimi

Üç boyutlu grafik çizimi için **splot** komutu kullanılır. Bunun kullanım biçimi **plot** komutu ile bir çok benzerlikler taşır. Splot ile hem fonksiyon hemde dosyalara daha önceden kaydedilmiş verilerin grafiği çizilebilir. Üç boyutlu grafik çizimi için verilerin özel bir şekilde dosyaya kaydedilmiş olması gerekir.

B.3.1 Fonksiyon Grafiği Çizme

Bir $f(x, y)$ fonksiyonunun grafiğini çizmek için

Ek C

BÖLÜM 2'e AİT PROGRAMLAR

C.1 KOK1 programı

```
program kok1
c***
c    yarlama metodu ile kok bulur. kokleri bulunacak
c f(x) fonksiyonu kullanc tarafndan verilmelidir
c
c aranan kokun bulundugu (a,b) aralg verilmelidir.
c f(x) EXTERNAL olarak belirtilmelidir
c***
external f

real*8 a,b,eps,xkok,f
integer imax,iter

data a,b/0.0,2.0/
data eps,imax/1.e-12,100/


write(*,*)'Kokun bulundugu aralIgI, "a" ve "b" yi giriniz:'
read(*,*) a,b

call yarilama(f,a,b,eps,imax,iter,xkok)


y= f(xkok)

write(*,100) xkok
write(*,200) y
write(*,300) iter

100 format(1x,' Bulunan kok degeri=',1pe12.5)
200 format(1x,' Fonksiyon degeri  =',1pe12.5)
300 format(1x,' Iterasyon sayisi  =',I5)
```

```
end
c********
subroutine yarilama(f,a,b,eps,imax,iter,xkok)
real*8 a,b,c,eps,xkok,f
integer imax,iter
c********************
! yarilama metodu ile kok bulmak uzere yazilmistir.
! f(x) koku aranan fonksiyon, (a,b) aranan kokun
! bulundugu araliktir
!
!  girdiler:
!   f: koku bulunacak fonksiyon
! (a,b) aranan kokun bulundugu araligin alt ve ust limitleri
! eps: taninan hata payi
! imax: koku bulmak icin yapilacak maksimum deneme sayisi
!  ciktilar:
! iter: kokun kac denemede bulundugu
! xkok: bulunan kok
!
c*********************

c kok araligi verilmis mi kontrol et

if(f(a)*f(b).gt.0) then
write(*,*)'  Kokun bulundugu (a,b) araligi verilmelidir'
write(*,*)'  "a" ve "b" degerlerini kontrol ediniz'
stop
return
endif

iter=0

10 c= (a+b)/2.

iter= iter + 1

if(abs(b-c).le.eps) then
xkok=c
return
endif

if( f(b)*f(c).le.0.) then
a=c
else
b=c
endif

if(iter.gt.imax) then
write(*,*)'  Maksimum iterasyon sayisinda istenen'
write(*,*)'  hassasiyeti saglanamadI'
```

```
xkok=c
else
goto 10
endif
return
end
c*************
function f(x)
real*8 f,x

f= x*x - 1.0
cc f= 0.5*x-sin(x)

return
end
```

Ek D

BÖLÜM 3'e AİT PROGRAMLAR

D.1 ADD1 programı

Euler metodunun kullanıldığı çok basit ve ilkel bir program aşağıda verilmiştir.

```
        PROGRAM ADD1
        REAL X,Y,XS,YA,DX,YD,DY
        INTEGER K,K1
        WRITE(*,*) 'ADD1'
1       CONTINUE
        WRITE(*,*)
C BaslangIc sartlarInI ve adIm uzunlugunu giriniz
        WRITE(*,*)'X,Y,XS ve K yI giriniz:'
        READ(*,*) X,Y,XS,K
        IF( K.LE.0 ) GOTO 99
        DX= (XS-X)/FLOAT(K)
        WRITE(*,*)
        WRITE(*,*) 'X =',X, '  Y =',Y
        WRITE(*,*) 'DX=',DX,'  K =',K
C Analitik cozum, YA
        YA= (Y+X+1.0)*EXP(XS-X)-(XS+1.0)
        DO 2 K1=1,K
        YD= Y+X
        X= X+DX
        Y= Y + DX*YD
2       CONTINUE
        DY=YA-Y
        WRITE(*,*) 'X =',X, '  Y  =',Y
        WRITE(*,*) 'YA=',YA,' Hata= DY =',DY
        GOTO 1
99      CONTINUE
        END
```

Kaynakça

[1] Verlet, L., Phys. Rev. **159**, 98 (1967); Phys. Rev **165**, 201 (1967).

[2]

[3] Numerical Recipes, The Art of Scientific Computing, Cambridge University Press.

[4] Metropolis, Rosenbluth, Rosenbluth, Teller and Teller, J. Chem. Phys. **21** 1087 (1953)

[5] Reif, F., *Fundamentals of Statistical and Thermal Physics*, McGraw-Hill International Series, London, (1985).

[6] Tokdemir, F., *Fortran 77*, (METU Central Printing Unit, Ankara, Turkey, 1995)

Dizin