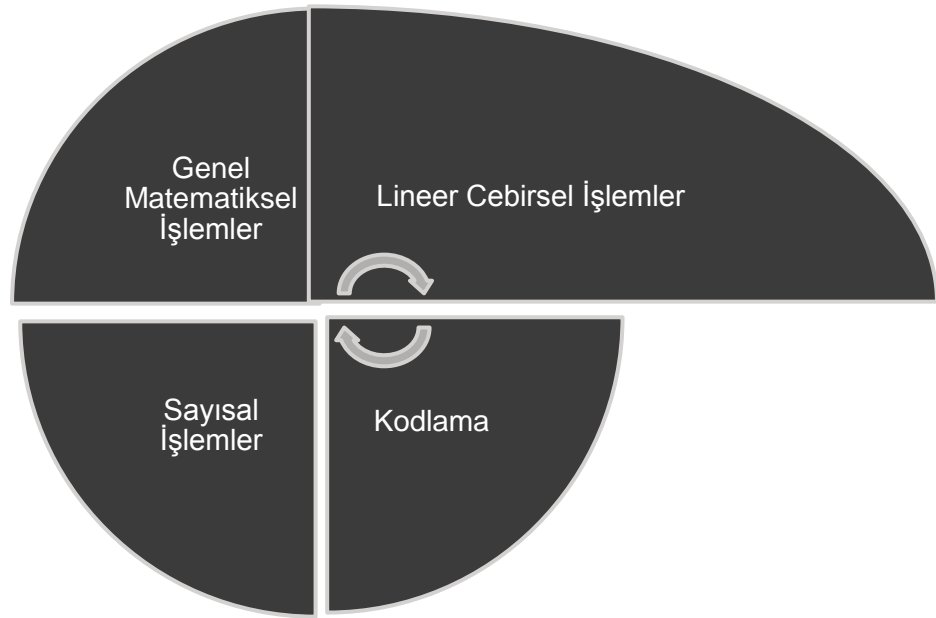


# OCTAVE ile Sayısal Hesaplama ve Kodlama



**Prof. Dr. Erhan Coşkun**

Karadeniz Teknik Üniversitesi  
Fen Fakültesi Matematik Bölümü Öğretim Üyesi

## İçindekiler

Önsöz .....	2
Teşekkür .....	4
1. OCTAVE ile Başlangıç .....	5
1.1 Temel Kavramlar .....	5
1.2 OCTAVE'ı Nasıl Temin Edebilirim?.....	7
1.3 OCTAVE Kullanıcı Ara yüzü .....	8
1.4 OCTAVE Kullanıcı Ara yüzünün Kişiselleştirilmesi.....	10
1.5 OCTAVE İçin Yardım.....	11
2. Bölüm Alıştırmaları .....	11
2. Komut Penceresinden Temel İşlemler .....	13
2.1 Skalerler ve Aritmetik Operatörler .....	13
2.2 Sıkça Kullanılan Ekran Komut ve Tuşları .....	14
2.3 Sonuç Gösterim Formatları.....	15
2.4 Mantıksal ve Aritmetik Karşılaştırma Operatörleri .....	16
2.4.1 Mantıksal karşılaştırma operatörleri.....	16
2.4.2 Aritmetik Karşılaştırma Operatörleri.....	17
2.5 İşlemlerde Öncelik .....	18
2.6 Bilgisayar Sayıları, dağılımları ve ilgili hatalar .....	20
2.7 Elemanter Fonksiyonlar .....	23
2.8 Komut Satırı Fonksiyonu Tanımlama .....	24
2.9 Anonim Fonksiyonu Tanımlama .....	26
2.10 Fonksiyon Grafiği Çizim(ezplot,ezpolar) ve Kaydı .....	26
2.11 OCTAVE help/doc Komutları .....	29
2. Bölüm Alıştırmaları .....	30
3.Vektörler.....	33
3.1 Vektör Tanımlama .....	33
3.2 Vektörler Üzerinde Aritmetik İşlemler(vektör cebirsel işlemler).....	34

3.3	Alt Vektör Tanımlama .....	36
3.4	Noktalı Vektör Operatörleri .....	37
3.5	Vektör Fonksiyonları .....	38
3.6	Vektör Argümanlı Satır Fonksiyonu Tanımlama .....	40
3.7	Plot Grafik Komutu .....	41
3.8	Subplot Fonksiyonu .....	45
3.9	Komut Dosyası Oluşturma .....	46
3.	Bölüm Alıştırmaları .....	49
4.	Vektörlerle İşlemler .....	53
4.1	Sayısal Türev .....	53
4.2	Sayısal İntegral(Eğri altındaki alan) .....	56
4.3	Bilinmeyen Ara değerin Tahmini(İnterpolasyon) .....	58
4.	Bölüm Alıştırmaları .....	62
5.	Matrislerle İşlemler .....	65
5.1	Matris Tanımlama .....	65
5.2	Matrisler Üzerinde Aritmetik İşlemler .....	67
5.3	Alt Matris Tanımlama .....	71
5.4	Matris Argümanlı Fonksiyon ve Grafiği .....	72
5.5	OCTAVE Matris fonksiyonları .....	76
5.5.1	Bir Matris ile Üretilen Dört Alt Uzayın Tabanı .....	76
5.5.2	Diğer Matris fonksiyonları .....	78
5.6	Lineer Denklem Sistemi Çözümü .....	83
5.	Bölüm Alıştırmaları .....	83
6.	OCTAVE ile Kodlama(Programlama) .....	87
6.1	Sıralı Yapı .....	88
6.2	Şartlı Yapı .....	92
6.3	Tekrarlı Yapı .....	101
6.	Bölüm Alıştırmaları .....	114
7.	OCTAVE ile Fonksiyon Programları .....	117
7.1	Fonksiyon Program Örnekleri .....	118

7. 1 Bölüm Alıştırmaları.....	128
7.2 Bazı Pratik Matematiksel Uygulamalar .....	129
7.2.1 Yakınsak dizi limiti.....	129
7.2.2 Sayı serisi toplamı.....	131
7.2.3 Sayısal türev .....	136
7.2.4 Sayısal integral .....	138
7.2.5 Yay uzunluğu .....	140
7.2.6 İkiye bölme yöntemi ile fonksiyon sıfır yeri.....	142
7.2.7 Dichotomous yöntemi ile fonksiyon minimumu .....	145
7. 2 Bölüm Alıştırmaları.....	148
7.3 Lineer Denklemler ve Ekstremler .....	150
7.3.1 Kuadratik fonksiyonun yerel ekstremleri .....	151
7.3.2 Pozitif definitlik testi .....	152
7.3.3 Köşegen baskınlık testi .....	155
7.3 Bölüm Alıştırmaları.....	158
7.4 OCTAVE Fonksiyon Kütüphanesinden Örnekler.....	162
7.4 Bölüm Alıştırmaları .....	166
Kaynakça .....	167
Dizin.....	168



# Önsöz

OCTAVE ([www.gnu.org/software/octave](http://www.gnu.org/software/octave)) John W. Eaton ve arkadaşları tarafından geliştirilen ve ilk versiyonu 1993 yılında kullanıma sunulan matematiksel yazılım kütüphanesi ve aynı zamanda programlama dilidir. Yazılım ismini, John W. Eaton'nun hocası olan Profesör Octave Levenspiel(1926-2017) den almaktadır[1]. OCTAVE her türlü sayısal analiz işleminde kullanılmak amacıyla geliştirilmiş ve ticari bir yazılım olan MATLAB ([www.mathworks.com](http://www.mathworks.com)) ile hemen hemen aynı yazılım kuralları ve yazılım kütüphanesine sahiptir. Genelde gerekli görülebilen çok az sayıda düzenleme ile MATLAB programları OCTAVE ortamında ve OCTAVE programları da MATLAB ortamında çalıştırılabilirler.

Bu çalışma, ücretsiz olarak temin edilebilen OCTAVE'ı matematik, fen bilimleri ve mühendislik bölümleri öğrenci ve araştırmacılarının sayısal analiz dersleri başta olmak üzere her türlü sayısal hesaplamaları için gerekli başlangıç bilgi ve uygulamalarını içermektedir. Sayısal hesaplamalar için bir araç olarak OCTAVE' ı tercih ettiğimiz bir çok nedeni var:

- birincisi vektörler üzerinde doğrudan cebirsel işlem yeteneğine sahip olması nedeniyle esas itibarıyla skalerler(reel veya kompleks sayılar) üzerinde işlem amaçlı olarak geliştirilen Fortran, Pascal ve C gibi diğer kodlama dillerine göre daha etkin olması,
- ikincisi bir programlama dili olmanın yanısıra zengin kütüphanesiyle kullanıcıların gerekli görebileceği grafik çizim fonksiyonları dahil bir çok fonksiyonu içeren özel amaçlı fonksiyonlar kütüphanelerine(tool boxes) sahip olması ve
- üçüncüsü ise ticari bir ürün olmaması nedeniyle özellikle öğrenciler tarafından kolayca erişilerek kullanılabilir olmasıdır.

Özgür olarak(kısaca ücret ödmeden) kullanılabilen ve benzeri ticari yazılımlara hemen hemen denk özelliklere sahip olan OCTAVE'ı, Karadeniz Teknik Üniversitesi Fen Fakültesi Matematik Bölümünde sayısal içerikli derslerde uzun süreden beri kullanmaktayım. Bu süreçte oluşturduğum ders notları ve uygulamaları içeren bu kitapçık yedi bölümden oluşmaktadır:

1. Bölümde OCTAVE yazılımının nereden ve nasıl temin edilebileceğini ve kullanıcı ara yüzünü tanıtıyoruz.
2. Bölümde komut penceresinden skalerler ile gerçekleştirilebilecek olan işlemleri özetliyoruz.
3. Bölümde vektörler ve vektör cebiri ile grafik çizimlerini inceliyoruz.

4. Bölümde sayısal türev, sayısal integral ve interpolasyon gibi temel vektörel işlemleri inceliyoruz.
5. Bölümde matrisler, lineer cebirsel işlemler ve yüzey grafiklerini inceliyoruz.
6. Bölümde programlama dili olarak OCTAVE'ı inceliyoruz.
7. Bölümde OCTAVE ile fonksiyon alt programlarının nasıl oluşturulduğunu basit ve özgün matematiksel uygulamalar ile başlamak suretiyle inceliyoruz. Ayrıca sayısal işlemler için sıkça kullanılan OCTAVE kütüphanesinden bazı fonksiyon alt programlarının kullanımını inceliyoruz.

Bu kaynakta, OCTAVE'ın sadece matematiksel amaçlarla ve sayısal içerikli lisans derslerine veya araştırmalara yardımcı olacağını düşündüğümüz özelliklerini pedagojik olarak ve özgün örnek ve uygulamalarla sunuyoruz. Bilgimiz dahilinde olduğu kadarıyla OCTAVE'ı yüksek düzeyde matematiksel uygulamalarıyla tanıtan Türkçe veya İngilizce olarak hazırlanmış bir kaynak mevcut değildir. OCTAVE ile ilgili daha teferruatlı genel bilgi için [1] nolu temel referansa ve her bir bölüm ile ilgili matematiksel ön bilgiler için ise Kaynakça kısmında verilen diğer matematiksel kaynakları öneririm.

Öğrencilerimiz ve araştırmacı arkadaşlarımıza faydalı olması dileğiyle,

Prof. Dr. Erhan Coşkun  
[erhan@ktu.edu.tr](mailto:erhan@ktu.edu.tr)

Ekim, 2018

## Teşekkür

Bu çalışmayı inceleyerek, önerileriyle katkı sağlayan başta değerli meslektaşlarım Prof. Dr. Ali Özdeş(İnönü Üniversitesi), Prof. Dr. Elçin Yusufoglu(Uşak Üniversitesi), Prof. Dr. Özkan Öcalan(Antalya Üniversitesi) olmak üzere, çalışmadaki yazım ve diğer hataları dikkatime sunarak düzeltilmesine katkı sağlayan tüm öğrencilerime teşekkürlerimi arz ederim.



# I. BÖLÜM

## 1. OCTAVE ile Başlangıç

Bu bölümde öncelikle

- ☑ sıkça kullanılan temel kavramları ve
- ☑ OCTAVE kullanıcı ara yüzünü inceliyoruz.

### 1.1 Temel Kavramlar

Bu çalışmada **algoritma**, **kod** veya **program**, **yazılım** gibi temel kavramlardan sıkça bahsedeceğiz. Çalışmanın okunabilirliği açısından öncelikle bu dökümanda sıkça kullanacağımız kavramları tanıyalım:

**Algoritma** belirlenen bir işlevi yerine getirmek için gerekli komutların uygun sırada yazılan bir kümesidir ve Harezmi(780(Horasan)-850(Bağdat)) isimli bilim adamının isminin Latince okunuşundan türemiş bir kelimedir[2].

**Kod veya program** algoritmanın bilgisayarların, yorumlayabileceği veya işleyebileceği dildeki ifadesidir. Bilgisayarın algılayabileceği dil ise **kodlama** veya **programlama dili** olarak adlandırılır. Örneğin **Basic**, **Fortran**, **Pascal**, **C** vb birer programlama dilleridirler. Pascal ve C Programlama dilleri için sırasıyla [3] ve [4] nolu referansları tavsiye ederiz.

**Kodlama(programlama)** ise probleme ait **yöntemin adımları**, **algoritma ve kod** hazırlama aşamalarını içeren ve son ürün olarak elde edilen kod veya programı hazırlama işlemidir.

Öte yandan ilgili bir diğer kavram olan **yazılım** ise belirli bir amaca yönelik işlevi yerine getirmek üzere hazırlanmış programların uygun biçimde entegrasyona verilen isimdir. Örneğin OpenOffice([www.openoffice.org.tr](http://www.openoffice.org.tr))

dokümantasyon hazırlama, veri işleme ve sunum amaçlarıyla hazırlanmış ve ücretsiz olarak temin edilebilen bir yazılımdır.

**Yazılım kütüphanesi** ise belirli bir amaca yönelik olarak hazırlanmış ve her biri ayrı ayrı çalıştırılabilen programlar topluluğudur. Lapack([www.netlib.org/lapack/](http://www.netlib.org/lapack/)) ve Linpack([www.netlib.org/lapack/](http://www.netlib.org/lapack/)) lineer cebirsel işlemler için hazırlanmış ve ücretsiz olarak kullanılabilen birer yazılım kütüphaneleridirler. OCTAVE her türlü sayısal analiz işleminde kullanılmak amacıyla geliştirilmiş ve ticari bir yazılım olan MATLAB ile benzer yazılım kuralları kullanan hem bir yazılım kütüphanesi ve aynı zamanda programlama dilidir.

OCTAVE özgür bir GNU yazılımıdır. Özgür yazılımlar, tanım gereği koduna ulaşılabilen, değişiklik yapılabilen ve belirli kurallar çerçevesinde başkalarıyla da paylaşılabilen yazılımlardır. Bu amaçla yazılımın kullanımı için ticari yazılımlarda olduğu gibi herhangi bir satın alım işlemi söz konusu değildir, ancak yazılım kullanıcılarından bu tür faaliyetlerin devam ettirilebilmesi için GNU özgür yazılım vakfına(Free Software Foundation) bağışta bulunmaları önerilir.

GNU ise özgür işletim sistemi üretimi başta olmak üzere, hem söz konusu sistem ve hem de diğer ticari işletim sistemleri üzerinde çalıştırılabilen özgür yazılım üretimi ve belirli kurallar çerçevesinde kullanımını düzenleyen ve Richard Stallman tarafından 1983 yılında başlatılan bir girişimdir([www.gnu.org](http://www.gnu.org)).

Özgür yazılımlar kullanıcıları kontrolünde geliştirilirler ve paylaşırlar. GNU yazılımları listesine [www.gnu.org/prep/ftp](http://www.gnu.org/prep/ftp) adresinden ulaşılabilir.

İşlemlerde kullanılan bir değişkenin vektör veya matrisi temsil etmediğini ifade etmek için söz konusu değişkenin *skaler* değişken veya kısaca skaler olduğu ifade edilir. Dolayısıyla skaler kısaca bir reel veya kompleks sayı anlamındadır.

Skalerler üzerinde gerçekleştirilen aritmetik işlemlere skaler cebirsel işlem ve vektörler üzerinde gerçekleştirilen işlemlere ise vektör cebirsel işlem adı verilir. İlerleyen bölümlerde inceleyeceğimiz üzere OCTAVE'ın en önemli özelliği vektör cebirsel işlem yeteneğine sahip olmasıdır.


## 1.2 OCTAVE'ı Nasıl Temin Edebilirim?

Windows kullanıcıları OCTAVE'ı <https://ftp.gnu.org/gnu/OCTAVE/windows/> adresinden ücretsiz olarak temin edebilirler.



Bilgisayarınızda kullanılan işletim sisteminin 32 veya 64 bit olmasına göre uygun sürümü yüklemeniz gerekmektedir. İşletim sisteminin özelliğini, genellikle masaüstünde bulunan Bilgisayar simgesinin üzerinde iken farenin sağ tuşuna tıklayarak, açılan menüden özellikler sekmesini tıklamak suretiyle öğrenebilirsiniz. Aşağıdaki görüntü çalıştığım bilgisayarın 64 bit işletim sistemine sahip olduğunu göstermektedir:

### Sistem

Derecelendirme:	 Windows Deneyimi Dizininizin yenilenmesi gerekiyor
İşlemci:	Intel(R) Pentium(R) D CPU 3.40GHz 3.40 GHz
Yüklü bellek (RAM):	2,00 GB
Sistem türü:	64 bit İşletim Sistemi
Kalem ve Dokunma:	Bu Görüntü Biriminde Kalem Girdisi veya Dokunarak Giriş yok

Bu dökümanın hazırlandığı süreçte windows 64 bit işletim sistemleri için en güncel sürüm olarak yer alan OCTAVE-4.2.1-w64.zip kurulum dosyası yukarıda belirtilen [ftp.gnu.org/gnu/octave/windows](https://ftp.gnu.org/gnu/octave/windows) klasörü içerisinde yer almaktadır.



[octave-4.2.1-w64-installer.exe](#)

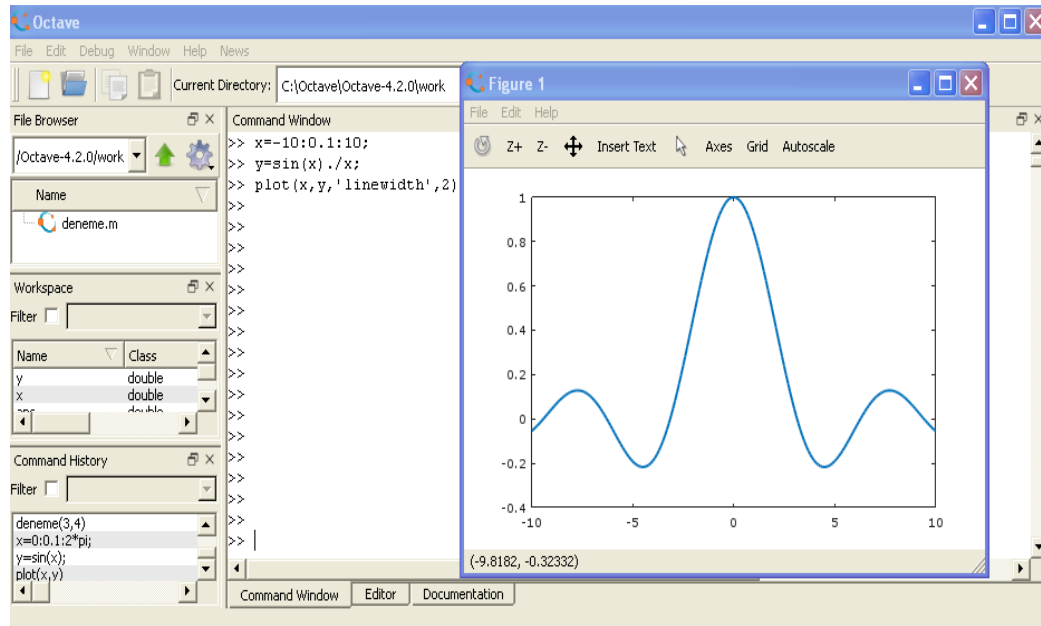
Söz konusu dosya veya daha güncel versiyonunu bilgisayarınıza kolayca kurabilirsiniz.

OCTAVE'nin gelişimine önemli katkıda bulunan kişi ve organizasyonların listesine temel referans kaynağı olarak[1] <https://www.gnu.org/software/OCTAVE/OCTAVE.pdf> dokümanından ulaşabilirsiniz.

### 1.3 OCTAVE Kullanıcı Ara yüzü

OCTAVE <http://OCTAVE.sourceforge.net> adresinden de temin edilebilir ve Mathworks([www.mathworks.com](http://www.mathworks.com)) firması tarafından üretilen MATLAB ile hemen hemen aynı sözdizimi(syntax) kurallarını kullanır.

Şekil 1.1 de OCTAVE GUI(Graphical User Interface)Grafiksel Kullanıcı Ara yüzü görülmektedir. Bu ara yüze çalışma oturumu adı verilir.



Şekil 1.1: OCTAVE Kullanıcı Ara yüzü

Şekil 1.1 de kullanıcının C:\OCTAVE\OCTAVE-4.2.0 klasörü içerisinde work isimli bir klasör oluşturduğunu ve çalışmalarını bu klasörde sakladığını görüyoruz.

Command Window(Komut Penceresi), File Browser(Klasörlerin listelendiği pencere), Workspace(mevcut oturumda tanımlanan değişkenler,

sınıfları, boyutları ve değerlerinin incelendiği pencere) ve Command History(Girilen komutlar kümesini içeren pencere) pencereleri yer almaktadır.

Ara yüzde yer alan “news” sekmesi altında yer alan “community news” sekmesi ile Şekil 1.2 de sunulan bağlantılar takip edilerek en güncel sürümde yer alan değişiklikler listesi incelenebilir.



### GNU Octave 4.2.1 Released

Octave Version 4.2.1 has been released and is now available for [download](#). An official [Windows binary installer](#) is also available.

— The Octave Developers, Feb 24, 2017

### GNU Octave 4.2.0 Released

GNU Octave version 4.2.0 has been released and is now available for [download](#). An official [Windows binary installer](#) is available. For [macOS](#) see the installation instructions in the wiki.

— The Octave Developers, Nov 14, 2016

### GNU Octave 4.0.3 Released

Octave Version 4.0.3 has been released and is now available for [download](#). An official [Windows binary installer](#) is also available.

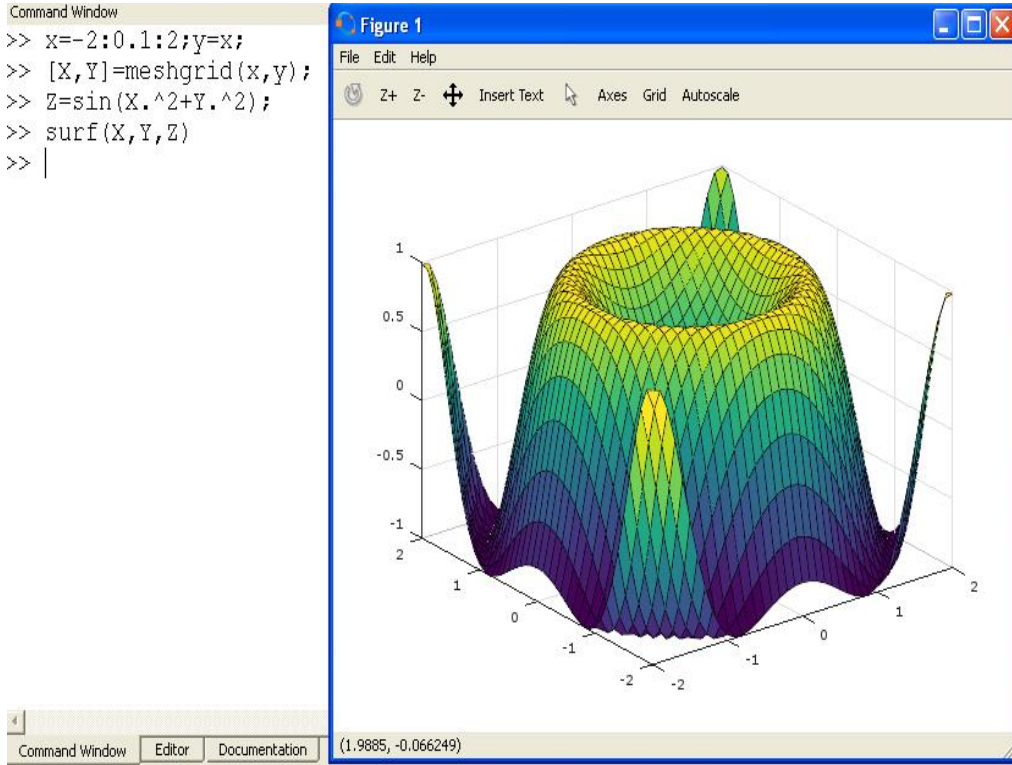
— The Octave Developers, Jul 2, 2016

Şekil 1.2: OCTAVE güncel sürüm bilgileri ara yüzü

OCTAVE, bu dökümanda inceleyeceğimiz üzere temel matematiksel işlemler başta olmak üzere vektörler ve matrisler üzerinde lineer cebirsel işlemlerin etkin olarak gerçekleştirilebildiği bir yazılım kütüphanesi olmanın yanı sıra, söz konusu kütüphaneye kullanıcılar tarafından tanımlanan yeni bileşenlerin(programların) ilave edilebilmesini sağlayan bir programlama dilidir.

Ayrıca OCTAVE oldukça zengin grafik seçeneklerine sahiptir. Aşağıdaki OCTAVE penceresinde  $[-2,2] \times [-2,2]$  bölgesi üzerinde  $f(x, y) = \sin(x^2 + y^2)$  fonksiyonunun grafiği sunulmaktadır.

## Octave ile Başlangıç



Şekil 1.3 OCTAVE’ da bir grafik örneği

## 1.4 OCTAVE Kullanıcı Ara yüzünün Kişiselleştirilmesi

OCTAVE kullanıcı ara yüzündeki Window sekmesi yardımıyla ara yüzde bulunan pencereleri kendi tercihinize göre yeniden düzenleyebilirsiniz. Window sekmesindeki işaretlenmiş seçenekleri birer birer kaldırarak ara yüzünüzün nasıl değiştiğini gözlemleyiniz. “Reset Default Window Layout(varsayılan pencere görünümü)” seçeneği ile kullanıcılar için hazırlanan genel çalışma ortamını seçiniz. İstemediğiniz pencereleri Window sekmesinde ilgili pencereye ait işaret simgesini kaldırarak çalışma ortamınızdan uzaklaştırabilirsiniz. Örneğin “Show command history” seçeneği yanındaki işareti kaldırarak çalışma ortamınızdan ilgili pencerenin kaybolduğunu gözlemleyiniz.

Çalışma oturumunda pencereler arasındaki sınır bölge üzerinde iken, imleç ( $\leftarrow$   $\rightarrow$ ) şekline dönüşünce, imleci basılı tutarak sağa veya sola kaydırmak suretiyle pencere boyutlarını değiştirebilirsiniz. Böylece pencere boyutlarını da kendi tercihinize göre düzenleyebilirsiniz.

## 1.5 OCTAVE İçin Yardım

Bu doküman OCTAVE' ı matematiksel kullanım amacıyla ve başlangıç düzeyde tanıtmayı amaçlamaktadır. İncelenen her bir konuya ait detaylı bilgiye ve bu dökümanda yer veremediğimiz OCTAVE' ın diğer özelliklerine çalışma oturumunda help yardım menüsünden Documentation(dokümantasyon) sekmesi altında bulunan On Disk(disk üzerindeki doküman) veya Online(internet üzerinden erişilebilen doküman) seçenekleri ile erişebilirsiniz.

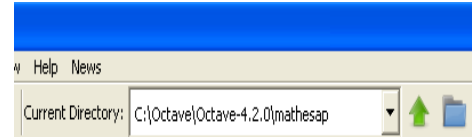
## 2. Bölüm Alıştırmaları

1. Şekil 1.1 i elde etmek için girilen komutlar ile şekilde gösterilen grafiği elde etmeye çalışınız.
2. OCTAVE ortamında Şekil 1.3 ü elde etmek için girilen komutlar ile şekilde gösterilen grafiği elde etmeye çalışınız.
3. Bir hesap makinesi olarak OCTAVE ortamını kullanmaya çalışınız.
4. OCTAVE ara yüzünde yer alan Help sekmesi ve ardından da Documentation->On Disk sekmelerini tıklatarak yardım konularının listesini ve içeriklerini inceleyiniz.
5. C:\OCTAVE\OCTAVE-4.\*.\* klasörü içerisinde mathesap isimli bir klasör oluşturunuz ve



şekme grubunda görülen sağdaki sekme "klasörlere göz atma

sekmesi(browse directories)" dir. Bu sekme yardımıyla "çalışılan klasörü(current directory)" oluşturduğunuz mathesap klasörü yapınız.







## II. BÖLÜM

### 2. Komut Penceresinden Temel İşlemler

Bu bölümde OCTAVE ortamında

- ☑ Aritmetik ve mantıksal operatörler tanıtılarak,
- ☑ İşlemlerde öncelik kavramı,
- ☑ Bilgisayar sayıları ve dağılımları,
- ☑ Sonuç gösterim formatları,
- ☑ Elemanter fonksiyonlar,
- ☑ Kullanıcı tarafından tanımlanabilecek ve satır fonksiyonu adı verilen fonksiyonların tanıtımı ve bu tür fonksiyonlar üzerinde gerçekleştirilebilecek işlemler ile
- ☑ Basit fonksiyon grafik çizimlerinin nasıl gerçekleştirileceğini inceliyoruz.

#### 2.1 Skalerler ve Aritmetik Operatörler

Bilinen toplama +, çıkarma -, çarpma \*, bölme / ve üs alma ^ operatörleri ile skalerler üzerinde interaktif aritmetik işlemler komut penceresinde yer alan ve komut promptu adı verilen ' >>' işaretinin bulunduğu komut satırında gerçekleştirilebilir(Bkz Tablo 2.1).

Tablo 2.1 Skalerler ile aritmetik işlemler

Toplama	Çıkarma	Çarpma	Bölme	Üs alma
>> 2+3	>> 2-3	>> 2*3	>> 2/3	>> 2^3
ans =	ans =	ans =	ans =	ans =

## Komut Penceresinden Temel İşlemler

5	-1	6	0.6667	8
---	----	---	--------	---

Tablo 2.1 de görüldüğü üzere belirtilen işlemler komut satırında yazılarak, enter veya return tuşuna basıldığında işlem sonuçları varsayılan değişken olarak bilinen 'ans' (answer) değişkenine atanır. Bu yönüyle skalerler üzerinde OCTAVE bir hesap makinesi olarak kullanılabilir.

İşlem sonuçları daha sonra kullanılmak isteniyorsa skalerlere isimler verilmelidir. Diğer bir deyimle skalerleri saklayan değişkenler tanımlanmalı ve değişkenlere değerleri '=' atama operatörü yardımıyla Tablo 2.2 de görüldüğü üzere atanmalıdır.

Tablo 2.2 Skaler değişkenler üzerinde işlemler

İşlem sonunda noktalı virgül yok			İşlem sonunda noktalı virgül		
>> a=2	>> b=3	>> c=a+b	>> a=2;	>> b=3;	>> c=a+b;
a =	b =	c =			
2	3	5			

Tablo 2.2 de sol üç sütunda a ve b skaler değerli değişkenlerine sırasıyla 2 ve 3 değerleri atanmakta ve toplama işleminin sonucu c değişkenine atanmaktadır. Atama işlemi sonunda satır sonunda noktalı virgül kullanılmaması durumunda değişken değerlerinin ve işlem sonucunun ekrana tekrar yazıldığını görüyoruz. Sağ üç sütunda ise aynı işlemlerin noktalı virgül kullanımı ile yapılması durumu gösterilmektedir. Dikkat edilirse ikinci durumda işlem sonucu c değişkeninde saklanmakta olup sonuç ekranda görüntülenmemektedir. İlgili değişken değerini görmek için Tablo 2.2 nin son sütununda görüldüğü üzere değişken isminin yazılarak enter veya return tuşuna basılması gerekir.

## 2.2 Sıkça Kullanılan Ekran Komut ve Tuşları

clear: Çalışma oturumunda tanımlanan değişken değerlerini siler.

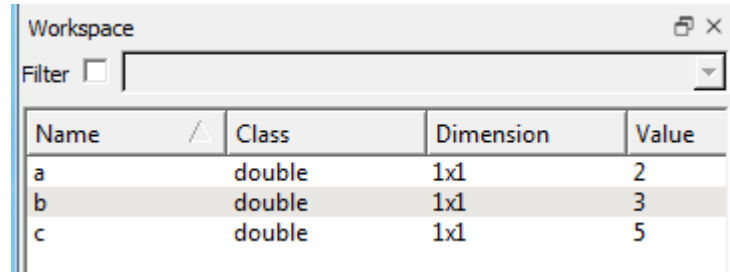
clc: Çalışma penceresini temizler, fakat değişken değerlerini silmez.

Yön tuşları: Yukarı ve aşağı yön tuşu ile önceden girilen komutlar geri alınabilir.

## Komut Penceresinden Temel İşlemler

Önceden girilen komutlar geri alınıp gerekirse gözden geçirilerek, istenilen komut elde edilebilir. Komut üzerinde değişiklik yapılmak istenen yere ise sağ ve sol yön tuşları ile ulaşılır. Değişiklik gerçekleştirildikten sonra `enter` tuşu ile düzeltilmiş komut icra edilir. Böylece tekrar aynı komutu veya işlemi yeniden yazmayarak, zamanımızı etkin kullanmış oluruz.

`whos`: komutu Şekil 2.1 de görüldüğü üzere çalışma oturumundaki değişken isimleri(Name), değişken türleri(class), boyut(dimension) ve değerlerini(value) listeler.



The screenshot shows the MATLAB Workspace window. It has a title bar 'Workspace' with a close button. Below the title bar is a 'Filter' checkbox and a search box. The main area contains a table with four columns: 'Name', 'Class', 'Dimension', and 'Value'. The table lists three variables: 'a' (double, 1x1, value 2), 'b' (double, 1x1, value 3), and 'c' (double, 1x1, value 5).

Name	Class	Dimension	Value
a	double	1x1	2
b	double	1x1	3
c	double	1x1	5

Şekil 2.1 Çalışma alanından görünüm

Şekil 2.1 de çalışma alanındaki değişken isimleri ve değerleri ile değişken türü olarak varsayılan türleri olan çift incelikli veya ‘double’ türü belirtilmektedir. Bu türdeki değişkenler, temsil ettikleri değerleri yaklaşık olarak 15 – 16 basamağa kadar doğru olarak temsil ederler. Sıkça kullanılmayan bir diğer değişken türü ise tek incelikli “single” türüdür ve temsil ettikleri değerleri yaklaşık olarak 7 – 8 basamağa kadar doğru olarak temsil ederler.

## 2.3 Sonuç Gösterim Formatları

İster tek veya isterse çift incelikli değişkenlerle yapılan işlemler olsun, işlem sonucunda elde edilen sonuçlar, arka planda yüksek incelikle yürütülse de OCTAVE da genelde *kısa format* olarak bilinen 5 rakamlı bir format ile gösterilirler. 15 rakamlı bir gösterim için *uzun format* kullanılır:

```
>> pi  
  
ans =  
  
3.1416
```

```
>> format long
>> pi

ans =

    3.14159265358979
```

Ayrıca işlem sonuçları rasyonel formatta da sunulabilir. Bunun için `format rat` komutu kullanılır:

```
>> format rat
>> pi

ans =

    355/113
```

Varsayılan kısa formata geri dönmek için

```
>> format
komutu tekrar girilir.
```

## 2.4 Mantıksal ve Aritmetik Karşılaştırma Operatörleri

Bu bölümde mantıksal ve aritmetik karşılaştırma operatörlerini inceliyoruz. Öncelikle mantıksal karşılaştırma operatörleri ile başlayalım:

### 2.4.1 Mantıksal karşılaştırma operatörleri

Mantıksal karşılaştırma operatörleri `ve(&)`, `veya(|)` operatörleridir. Örneğin `a&b` işleminde `a` ve `b` nin her ikisi de sıfırdan farklı ise mantıksal karşılaştırma sonucu doğru ve sayısal değeri ise `1`, diğer durumlarda ise yanlış olarak algılanır ve sayısal değeri ise sıfıra eşit olur.

```
>> a=1;b=2;
```

olmak üzere

```
>> c=a&b
```

işleminde hem  $a$  ve hem de  $b$  nin değeri sıfırdan farklı olduğu için karşılaştırma doğru olarak algılanarak, aşağıda görüldüğü üzere  $c$  ye 1 değeri atanmaktadır:

```
c =
```

```
1
```

Öte yandan  $a|b$  mantıksal karşılaştırmasında ise  $a$  veya  $b$  den en az biri sıfırdan farklı ise karşılaştırma doğru (ve sayısal değeri 1) diğer durumlarda ise karşılaştırma yanlış olarak algılanarak, sayısal değeri sıfıra eşit olur. Örneğin

```
>> c=a|b
```

```
ile yine
```

```
c =1
```

değeri aldığını gözlemleriz. Diğer örnekler için Tablo 2.3 e bakınız.

### 2.4.2 Aritmetik Karşılaştırma Operatörleri

Aritmetik karşılaştırma operatörleri

```
<(küçüktür), >(büyüktür),  
<=(küçük veya eşittir),  
>=(büyük veya eşittir),  
==(eşit mi), ?  
~=(eşit değil mi) ?
```

operatörleridir. Örneğin

```
>> a=0;b=1; c=a<b
```

işleminde öncelikle  $a<b$  karşılaştırması yapılmakta ve karşılaştırma doğru olduğu için  $c$  değişkenine 1 değeri atanmaktadır:

```
c = 1
```

fakat

```
>> d=a>b
```

de ise karşılaştırma yanlış olduğu için

`d = 0`

değeri elde edilmektedir.

`>> c=a==b`

işleminde `a` ve `b` nin değerlerinin eşit olup olmadığı test yapılmakta olup, karşılaştırma doğru olmadığı için `c` değişkeni sıfır değeri almaktadır.

`c = 0`

Ancak

`>> c=a~=b`

işleminde ise `a` değişkeninin değeri `b` nin değerine eşit olmadığı için karşılaştırma doğru olup `c` değişkeni 1 değerini almaktadır.

Özetle karşılaştırma işleminin doğru olması durumunda işlemin sayısal değeri, yani sonucu 1, diğer durumda ise 0 dir.

Diğer operatörler için Tablo 2.3 ü inceleyiniz.

Tablo 2.3 Karşılaştırma operatörleri ile elde edilen tipik sonuçlar

a	b	a&b	a b	a<b	a<=b	a>b	a>=b	a==b	a~=b
1	2	1	1	1	1	0	0	0	1
1	0	0	1	0	0	1	1	0	1

## 2.5 İşlemlerde Öncelik

Öncelik kuralları diğer programlama dillerinde olduğu gibidir.

Öncelik düzeyleri özetle Tablo 2.4 te verilmektedir.

Tablo 2.4 Aritmetik, karşılaştırma ve atama operatör öncelik düzeyleri

İşlem öncelik düzeyleri	
<code>^</code> : üs alma	en yüksek düzey
<code>*</code> , <code>/</code>	↓

## Komut Penceresinden Temel İşlemler

+, -	↓
<, <=, >, >=, ==, ~=	↓
=	en düşük düzey

Tablo 2.4 ten görüldüğü üzere üs alma işlemi en yüksek önceliğe sahip iken çarpma ve bölme aynı düzeyde ve ikinci sırada yer alırlar. Toplama ve çıkarma ise aynı düzeyde ve üçüncü sırada yer alırlar. Aynı düzey öncelikli operatörlerde işlem sırası soldan sağa doğru gerçekleştirilir. Aritmetik operatörleri karşılaştırma operatörleri takip eder, atama(=) operatörü ise en düşük önceliğe sahiptir.

Tablo 2.5 deki örnekleri inceleyerek, Tablo 2.4 te verilen işlem önceliklerinin doğru olduğunu kontrol ediniz:

Tablo 2.5 İşlemlerde öncelik

>> 2^3+1 ans = 9	>> 2*3+1 ans = 7	>> 2<=1+3 ans = 1	>> x=4<3+1 x = 0
>> 2^3*5 ans = 40	>> 2*3-1 ans = 5	>> 3>1+3 ans = 0	>> x=2*3>7+1 x = 0
>> 2^3/2 ans = 4	>> 2/3+1 ans = 1.6667	>> 3==1+2 ans = 1	>> 2~=1+1 ans = 0
>>format rat >> 2/3+1 ans = 5/3	>> 2/(3+1) ans = 1/2	>> 2^3*2 ans = 16	>> 2^(3*2) ans = 64

Tablo 2.5 in son satırında da görüldüğü üzere parantez yardımıyla işlem önceliği değiştirilebilir.

Matematiksel ifadelerin OCTAVE ortamında yazılımına özen gösterilmelidir. Matematiksel bir ifadenin doğru ve muhtemel yanlış OCTAVE yazılımı için Tablo 2.6 yı inceleyiniz.

Tablo 2.6 Bazı matematiksel ifadeler ve OCTAVE yazılımları

Matematiksel ifade	OCTAVE yazılımı	Yanlış Yazılım(!)
$\frac{a}{b+c}$	a/(b+c)	a/b+c
$\frac{a-b}{c+d}$	(a-b)/(c+d)	a-b/(c+d)
$\frac{-b+\sqrt{c}}{3a}$	(-b+sqrt(c))/(3*a)	(-b+sqrt(c))/3*a

$2^{xy}$	$2^{(x * y)}$	$2^{x * y}$
----------	---------------	-------------

## 2.6 Bilgisayar Sayıları, dağılımları ve ilgili hatalar

Bir bilgisayarda temsil edilebilecek sadece sonlu sayıda çift incelikli (double) ve yine sonlu sayıda tek incelikli(single) sayı mevcuttur ve bu sayılar sırasıyla 8 ve 4 baytlık bir bellek alanında temsil edilebilecek olan sayılardır. Aksi belirtilmedikçe OCTAVE ortamında tanımlanan her değişken çift incelikli değer kabul eder. Bilgisayar belleğinde temsil edilebilen sayılara bilgisayar sayıları adı verilmektedir.

Bilgisayar donanım sistemine de bağlı olarak değişmek üzere, bilgisayarımızda kullanılan en küçük pozitif ve en büyük çift incelikli reel sayıları OCTAVE'ın `realmin` ve `realmax` komutları yardımıyla öğrenebiliriz.

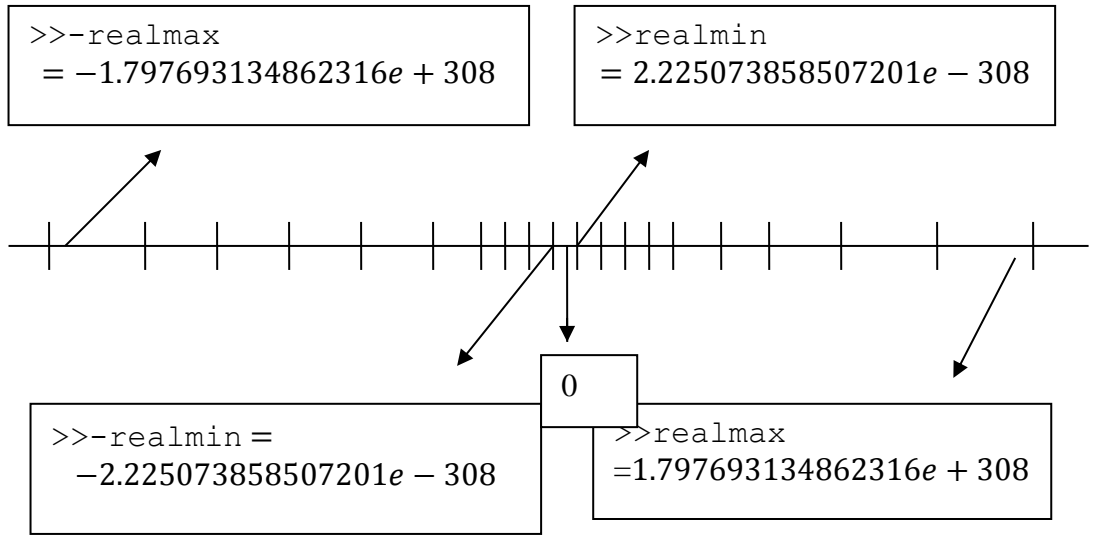
Benzer biçimde en büyük negatif en küçük negatif sayıları da sırasıyla bu sayıların negatifleri olarak elde edebiliriz.

Öte yandan  $\pi$  ve  $e$  gibi irrasyonel sayıların sonlu sayıda ondalıklı basamak ile temsil edilemeyeceğini biliyoruz. Bu durumda bu tür sayılara en yakın çift incelikli bilgisayar sayısı belleğe kaydedilerek, işlemler bu yuvarlatılmış sayı ile gerçekleştirilir. Böylece *işlemlerimizin her zaman beklediğimiz gerçek sonucu veremeyeceğini göz ardı etmemeliyiz.*

Sıfır noktası komşuluğunda yoğun olan yer alan bilgisayar sayılarının orijinden uzaklaştıkça yoğunluğu azalmakta, diğer bir deyimle iki komşu bilgisayar sayısı arasındaki uzaklık artmaktadır. Bu durum Şekil 2.2 de gösterilmekte olup, sayıların `kayan nokta formatı` adı verilen ve sayısal analiz derslerinde incelenen formattaki gösteriminden kaynaklanmaktadır.



## Komut Penceresinden Temel İşlemler



Şekil 2.2 Bilgisayar sayılarının dağılımı ile en küçük ve en büyük negatif ve pozitif çift incelikli sayılar.

Öte yandan bilgisayar belleğinde her bir reel sayı için tahsis edilen alan sınırlı olduğu için hiçbir irrasyonel sayı bilgisayarda tam olarak temsil edilemez. Örneğin

```
>> pi
ans =
3.14159265358979
```

olarak kabul edilmekte olup, bu durumda sıfır olmasını beklediğimiz  $\sin(\pi)$  değeri

```
>> sin(pi)
ans = 1.2246e-016
```

olarak hesaplanmaktadır, çünkü bilgisayar belleğinde saklanan pi sayısı gerçek matematiksel  $\pi$  değerini tam olarak temsil etmemektedir. Bu nedenle yukarıdaki işlem sonucunda küçük kabul edebileceğimiz bir hata oluşmuştur.

## Komut Penceresinden Temel İşlemler

Bir  $x$  sayısı ile bu sayıya en yakın bilgisayar belleğindeki sayı arasındaki uzaklık  $\text{eps}(x)$  komutu ile öğrenilebilir. Örneğin  $\pi$  sayısı olarak temsil edilen sayı ile bu sayıya en yakın sayı arasındaki uzaklık

```
>> eps(pi)
```

```
ans =
```

```
4.440892098500626e-016
```

dir.

Bir başka deyimle bilgisayar belleğinde temsil edilebilen ve  $[\pi, \pi + \text{eps}(\pi)]$  aralığında, yani

```
[3.141592653589793e+000, 3.141592653589794e+000]
```

aralığında başka hiçbir sayı mevcut değildir.

Şekil 2.2 de de görüldüğü üzere bilgisayar sayıları arasındaki uzaklık sayılar büyüdükçe daha da artmaktadır.

Örneğin

$1.0000e+015$  ile bu sayıya en yakın sayı arasındaki uzaklık  $0.125$ ,

$1.0000e+016$  ile bu sayıya en yakın sayı arasındaki uzaklık  $2$  ve

$1.0000e+017$  ile bu sayıya en yakın sayı arasındaki uzaklık  $16$  dir.

Bir işlem sonucunda elde edilen sonuç, bilgisayar sayı sisteminde mevcut değilse sistemdeki en yakın sayı sonuç olarak kabul edilir.

*Böylece mutlak değerce çok büyük ve çok küçük sayılarla aynı anda aritmetik işlem yaparken büyük yuvarlama hatalarının oluşabileceğini göz ardı etmemeliyiz.*

Örneğin

```
>> (1+1e17)-1e17
```

```
ans = 0
```

işleminin aritmetiksel olarak 1 e eşit olması gereken sonucunun sıfır olması sayı sisteminden kaynaklanan bir hatadır. Çünkü yukarıda belirtildiği üzere  $1e17$

ile bu sayıya en yakın sayı arasındaki uzaklık 16 olup,  $1+1e17$  işlemini sonucu en yakın sayıya yuvarlanarak  $1e17$  kabul edilmiş ve 1 değerinde bir yuvarlama hatası yapılmıştır. Bu hata sonucu doğal olarak etkilemiştir.

Daha ötesi

```
>> 1000*((1+1e17)-1e17)
ans = 0
```

elde ederiz. 1000 olması gereken sonuç 0 değerine eşit olarak elde edilmiştir!

## 2.7 Elemanter Fonksiyonlar

OCTAVE ortamında trigonometrik fonksiyonlar matematikte kullanıldığı biçimde ifade edilirler:  $x$  bir skaler(radyan cinsinden), olmak üzere trigonometrik fonksiyonlar için aşağıdaki gibi alışılmış matematiksel gösterimler kullanılır:

$$\sin(x), \cos(x), \tan(x), \cot(x), \sec(x), \csc(x)$$

ve ters trigonometrik fonksiyonlar ise trigonometrik fonksiyonların başına  $a$  harfi eklenmek suretiyle elde edilirler:

$$\text{asin}(x), \text{acos}(x), \text{atan}(x), \text{acot}(x), \text{asec}(x), \text{acsc}(x)$$

Benzer biçimde, hiperbolik fonksiyonlar

$$\sinh(x), \cosh(x), \tanh(x), \coth(x), \text{sech}(x), \text{csch}(x);$$

ve tersleri(başlarına getirilen  $a$  harfi ile),

logaritmik fonksiyonlar  $\log(x)$  ( $\ln(x)$  demektir),  $\log_{10}(x)$  ise  $\log(x)$ ,

üstel fonksiyonlar  $e^x = \exp(x)$ ,  $a^x = a^x$

ve güç fonksiyonu için  $x^a = x^a$

gösterimi kullanılır. Ayrıca  $\sqrt{x} = \text{sqr}(x)$  ve mutlak değer  $|x| = \text{abs}(x)$  gösterimleri kullanılır.

Bir açının derece ölçüsü için ise,  $x$  derece türünde olmak üzere, ilgili trigonometrik fonksiyonları için yukarıdakilerden farklı olarak

$$\text{sind}(x), \text{cosd}(x), \text{tand}(x), \text{cotd}(x), \text{secd}(x), \text{csd}(x)$$

gösterimleri kullanılır. Örneğin

```
>> sin(pi/2)
```

```
ans =
```

```
      1  
>> sind(90)
```

```
ans =1
```

Benzer olarak sinüsü 1/2 olan açıyı ise

```
>> asind(1/2)  
ans = 30.000
```

olarak elde ederiz.

## 2.8 Komut Satırı Fonksiyonu Tanımlama

Yukarıda belirtilen elemanter fonksiyonlar yardımıyla komut satırında tanımlanabilen ve komut satırı fonksiyonu (`inline function`) adı verilen yeni fonksiyonlar tanımlanabilir. Genel yazılımı

```
inline(ifade, arg1,arg2,...,argn)
```

biçimindedir.

### ÖRNEK 2.1

```
>> f=inline('x^2-3*x+2')
```

```
f =
```

```
Inline function:  
f(x) = x^2-3*x+2
```

## Komut Penceresinden Temel İşlemler

Tanımlanan komut satırı fonksiyonunun bir  $a$  noktadaki değeri  $f(a)$  yazılımı ile elde edilir. Örneğin

```
>> f(3)
```

```
ans =
```

```
2
```

Bağımsız değişken ismi için  $x$  yerine başka bir isim de kullanılabilir:

```
>> g=inline('exp(t)+log(t)-sin(t)')
```

```
g =
```

```
Inline function:
```

```
g(t) = exp(t)+log(t)-sin(t)
```

```
>> g(2)
```

```
ans =
```

```
7.1729
```

Bileşke fonksiyonun değeri de matematiksel notasyona uygun olarak hesaplanır.

```
>> g(1)
```

```
ans = 1.8768
```

```
>> f(ans)
```

```
ans = -0.10801
```

veya

```
>> f(g(1))
```

```
ans = -0.10801
```

elde edilir.

Çok değişkenli fonksiyonlar da benzer biçimde tanımlanırlar. Örneğin

```
>> f=inline('sin(2*x+3*y)')
```

```
f =
```

```
Inline function:  
f(x,y) = sin(2*x+3*y)  
  
>> f(2,1)  
  
ans = 0.6570
```

## 2.9 Anonim Fonksiyonu Tanımlama

Komut satırı fonksiyonu yerine kullanılan ve kullanılması önerilen bir diğer alternatif ise anonim fonksiyondur ve aşağıdaki gibi tanımlanır:

`f=@(değişken veya değişkenler) fonksiyon`

Örneğin

```
f = @(x) x^2;
```

ile tanımlanan f fonksiyonu için

```
>> f(2)  
ans = 4
```

elde ederiz. Çok değişkenli fonksiyonlar da benzer biçimde tanımlanırlar.

Örneğin

```
>> f=@(x,y) x^2+2*x*y+y^2;
```

```
>> f(1,2)  
ans = 9
```

## 2.10 Fonksiyon Grafiği Çizim(ezplot,ezpolar) ve Kaydı

Ezplot komutu ile inline komut satırı fonksiyonu veya anonim bir fonksiyonun Kartezyen Koordinat Sisteminde grafiği çizilebilir: Genel yazılımı

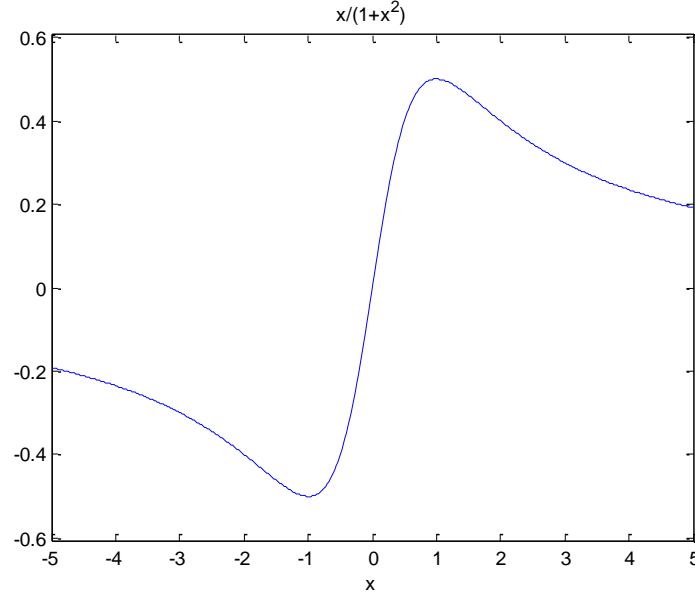
```
ezplot (f) veya ezplot(f,[a,b])
```

## Komut Penceresinden Temel İşlemler

biçimindedir. Burada a ve b ise grafik çiziminde apsis sınırlarını belirtmektedir. Belirtilmediği durumlarda  $a = -2\pi$ ,  $b = 2\pi$  olarak kabul edilir.

```
>> f=inline('x/(1+x^2)')
```

```
>> ezplot(f, [-5,5])
```



Şekil 2.3 Ezplot ile grafik örneği

```
>> print -djpeg grfk1
```

Print komutu ile çizdirilen herhangi bir grafik istenilen bir grafik dosyasına kaydedilebilir. Bu örnekte çizdirilen grafik grfk1.jpeg dosyasına kaydedilmektedir. Belirtilen grafik word veya diğer bir ortama bilinen resim ekleme işlemi yardımıyla ilave edilebilir.

OCTAVE ortamında da benzer işlem gerçekleştirilebilir.

```
>> print -djpeg grfk1
```

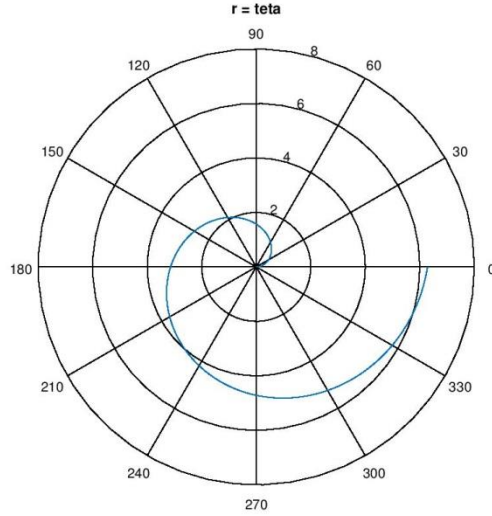
ile grfk1.jpeg grafik dosyası oluşturulur. Diğer seçenekleri help print komutu yardımıyla inceleyebilirsiniz.

Benzer biçimde ezpolar fonksiyonu yardımıyla kutupsal koordinat sisteminde  $r = f(teta)$  biçiminde tanımlanan fonksiyonun verilen  $[a,b]$  aralığındaki grafiği çizdirilebilir.

## Komut Penceresinden Temel İşlemler

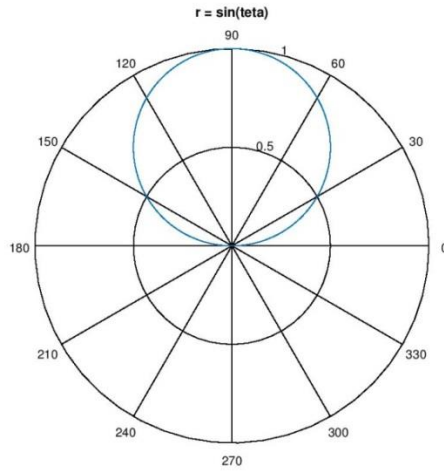
Örneğin  $r = \theta$  fonksiyonunun  $[0, 2\pi]$  aralığındaki grafiği aşağıda verildiği gibi elde edilir:

```
>>f=inline('teta'); >> ezpolar(f,[0,2*pi]);
```



Şekil 2.4 Ezpolar ile  $r = \theta$  fonksiyon grafiği

Öte yandan  $r = \sin(\theta)$  fonksiyonunun  $[0, \pi]$  aralığındaki grafiği aşağıda verildiği gibi eldilir:

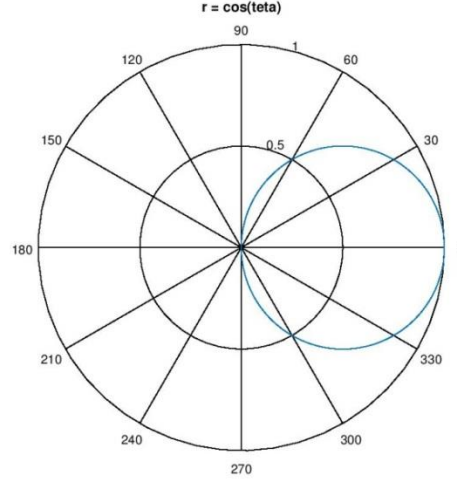




## Komut Penceresinden Temel İşlemler

Şekil 2.5 Ezpolar ile  $r = \sin(\theta)$  fonksiyon grafiği

Şimdi de  $r = \cos(\theta)$  fonksiyonunun kutupsal koordinatlarda grafiğine bakalım:



Şekil 2.6 Ezpolar ile  $r = \cos(\theta)$  fonksiyon grafiği

**UYARI:** Her iki fonksiyonun da kutupsal koordinatlardaki periyodunun Kartezyen koordinat sisteminde bilinen  $2\pi$  değeri yerine  $\pi$  olduğuna dikkat edelim. Kutupsal koordinatlarda fonksiyonların periyotları dikkatlice incelenmesi gereken bir konudur.

## 2.11 OCTAVE help/doc Komutları

OCTAVE fonksiyonlarının kullanımı hakkında açıklayıcı bilgiye help komutu yardımıyla ulaşabilirsiniz.

```
>> help log
LOG    Natural logarithm.
      LOG(X) is the natural logarithm of the
      elements of X.
      Complex results are produced if X is not
      positive.

      See also log1p, log2, log10, exp, logm,
      reallog.
```

```
Overloaded methods:  
darray/log
```

```
Reference page in Help browser  
doc log
```

Açıklayıcı bilginin son satırında yer alan altı çizili fonksiyon bağlantısına (bu örnekte `doc log`) tıklayarak ilgili fonksiyonun nasıl kullanıldığını bir örnek ile birlikte açıklayan yardım dosyasına erişebilirsiniz.

Alternatif olarak

```
>> doc log
```

seçeneği ile de yardım alınabilir:

## 2. Bölüm Alıştırmaları

1. Bilgisayarınızda kullanılan en küçük ve en büyük çift incelikli pozitif sayıları belirleyiniz.
2. Belirlediğiniz en büyük sayıya pozitif sayılar ilave ederek sonucu gözlemleyiniz. Elde edeceğiniz tek farklı sonuç `inf` olmalıdır!
3. Yaygın olmasa da bir değişken tek incelikli olarak tanımlanabileceği gibi, çift incelikli bir değişken değeri tek inceliğe dönüştürülebilir. Örneğin  $x = \pi$  değeri `y = single(x)` komutu ile tek incelikli sayıya dönüştürülür. `eps(x)` ve

`eps(y)` komutları ile sırasıyla çift ve tek incelikli  $\pi$  sayısına en yakın olan bilgisayar sayıları arasındaki uzaklığı belirleyiniz. Hangisi daha büyüktür? Sizce neden?

4. Bilgisayar sayıları arasındaki uzaklığın büyüyen sayı değerleri için arttığını `eps` komutu yardımıyla gözlemleyiniz.
5. Aşağıdaki fonksiyonları `inline` fonksiyonu ile tanımlayarak, belirtilen noktadaki değerlerini hesaplatınız.

a. 
$$f(x) = \frac{\sin(x)}{x^3 + 2x - 1}, x = 2$$

b.  $f(\theta) = \sin(\theta) + \sin(2\theta),$   
 $\theta = 90^\circ$

c.  $f(x) = \log(x) + 3^x - x, x = 2$

d.  $f(x) = \ln(x) - e^{3x} +$   
 $\sinh(x), x = 3$

6.  $x = 2$  değeri için aşağıdaki trigonometrik ifadeleri karşılarında verilen OCTAVE ifadelerinin yazılımını kontrol ederek sonuçlarını belirleyiniz.

a.  $\sin(x^2) \rightarrow \gg \sin(x^2)$

b.  $\sin^2(x) \rightarrow \gg \sin(x)^2$

c.  $\cos(2x) = \cos^2(x) - \sin^2(x)$

$\gg \cos(2*x);$

$\gg \cos(x)^2 - \sin(x)^2$

d.  $\sin^2(x) + \cos^2(x) = 1$

$\gg \sin(x)^2 + \cos(x)^2$

e.  $\sin(2x) = 2 \sin(x) \cos(x)$

$\gg \sin(2*x)$

$\gg 2*\sin(x)*\cos(x)$

7.  $x = 10, y = 5$  değerleri için aşağıdaki özdeşliklerin OCTAVE ortamında da sağlandığını gözlemleyiniz.

a.  $\log(xy) = \log(x) + \log(y)$

b.  $e^{x+y} = e^x e^y$

c.  $\cos(x + y) = \cos(x) \cos(y) -$   
 $\sin(x) \sin(y)$

d.  $\sin(x + y) = \sin(x) \cos(y) +$   
 $\cos(x) \sin(y)$

8. Aşağıdaki işlemleri rasyonel formatta gerçekleştiriniz.

a.  $x = \pi + 0.3$

b.  $x = e^2 - e + \sqrt{2}$

9. Aşağıdaki fonksiyonları komut satırı veya anonim fonksiyon olarak tanımlayarak belirtilen noktadaki değerlerini hesaplayınız.

a.  $f(x) = \frac{x+1}{x^2-3x+1}, x = 3$

b.  $f(x) = \sin(x) + \sinh(x), x =$   
 $\pi/2$

c.  $f(x) = \sqrt[4]{x}, x = 2$

d.  $f(x) = \arcsin(x), x = 1/2$

e.  $f(x) = \sin(x + y); x = 1, y =$   
 $2$

10. Soru 9(a-d) şıklarında verilen fonksiyonların uygun olan aralıklarda grafiklerini `ezplot` fonksiyonu yardımıyla ve 9(e) yi ise `ezmesh` yardımıyla çiziniz.

11. Aşağıda verilen fonksiyonların kutupsal koordinat sisteminde grafiklerini `ezpolar` fonksiyonu yardımıyla çiziniz.  $n$  ve  $k$  nın değişen değerleri için `in ne` gözlemliyorsunuz?

a.  $r = \cos(n\theta), \theta \in [0, 2\pi], n =$   
 $1, 2, 3, 4, 5$

b.  $r = \sin(n\theta), \theta \in [0, 2\pi], n =$   
 $1, 2, 3, 4, 5$

c.  $r = 1 + k \sin(\theta), \theta \in [0, 2\pi]$   
 $k = 0.2, 0.5, 0.8, 1, 1.5, 2.6$

d.  $r = 2\theta + 1, \theta \in [0, 5]$



# III. BÖLÜM

## 3.Vektörler

Bu bölümde

- ☑ Vektörler üzerlerindeki aritmetik işlemleri tanımlayarak,
- ☑ alt vektör tanımlama işlemleri,
- ☑ noktalı operatörler ile işlemler,
- ☑ vektör fonksiyonları,
- ☑ vektörel grafikler ve
- ☑ komut dosyası oluşturma işlemleri inceliyoruz.

### 3.1 Vektör Tanımlama

Vektörler köşeli parantez içerisinde elemanlar arasında boşluk veya virgöl konularak tanımlanabileceği gibi, elemanlar arasındaki farkı birbirine eşit olan vektörler için başlangıç noktası, artış veya azalış miktarı ve bitiş noktasının belirtilmesi suretiyle de tanımlanabilirler:

```
>> x=[1 3 2]
```

```
x =
```

```
1 3 2
```

veya

```
>> y=1:2:10(Başlangıç noktası=1; artış miktarı=2; bitiş noktası=10)
```

```
y =
```

```
1 3 5 7 9
```

y vektörü aynı zamanda  $1 \times 5$  boyutlu matris olarak değerlendirilmektedir.

$Y$  nin transpozesi  $y'$  ile elde edilir ve sütun vektörü olup,  $5 \times 1$  boyutlu bir matris olarak değerlendirilmektedir. Öte yandan `linspace` ve `logspace` komutları yardımıyla da vektörler tanımlanabilirler: (`>>help linspace` veya `>>help logspace` ile ilgili fonksiyonların kullanımını inceleyiniz.).

```
>> linspace(0,1,6)
```

```
ans =
```

```
0      0.2000      0.4000      0.6000      0.8000      1.0000
```

`linspace` komutu ile  $[0,1]$  aralığında, aralarındaki uzaklıkları birbirine eşit olan altı adet noktaya sahip bir vektör oluşturulmaktadır.

**NOT:** Bir vektörün herhangi bir elemanına erişerek düzenleme yapmak mümkündür:

Örneğin yukarıda tanımlanan  $x$  vektörünün ikinci elemanı

```
>> x(2)
```

```
ans =
```

```
3
```

olarak elde edilir.

```
>> x(2)=4
```

```
x =
```

```
1      4      2
```

ile vektörün ilgili elemanı değiştirilmiş olur.

### 3.2 Vektörler Üzerinde Aritmetik İşlemler(vektör cebirsel işlemler)

Tablo 3.1 de iki vektörün arasında tanımlı işlemler ile, bir skalerin bir vektörle çarpımı gibi lineer cebirsel işlemlere ilaveten lineer cebirsel olarak anlam ifade etmeyen, ancak OCTAVE ortamında işlemlerde kolaylık açısından tanımlı olan bir vektör ile bir skalerin toplama işleminin nasıl gerçekleştirildiği gösterilmektedir.

## Vektörler

Tablo 3.1 Vektörler üzerinde Aritmetik İşlemler

X	Y	
>> x=[2 4 6]	>>y=[1,2,3]	
x =	y =	
2        4        6	1    2        3	
x        ve        y vektörlerinin iç çarpımı	>> z=x+y	
	z =	
	3        6        9	
	>> x*y ??? Error using ==> mtimes Inner matrix dimensions must agree. (Matris çarpımı tanımlı değil)	
x        ve        y vektörlerinin iç çarpımı	>> x'*y ans =	
	2        4        6 4        8        12 6        12       18	
	(3x1) boyutlu x' sütun vektörü ile (1x3) boyutlu y satır vektörünün çarpımı ile (3x3)boyutlu bir matris oluşur.	
	>> x*y' ans = 28 (1x3) boyutlu x satır vektörü ile (3x1) boyutlu y' sütun vektörünün çarpımının boyutu (1x1)dir.  <b>Not:</b> iç çarpım işlemi x ve y vektörlerinin satır veya sütun vektörü olmalarına göre çarpım sonucunu 1x1 yapacak uygun	

## Vektörler

	transpoz işlemi sonunda elde edilir.
>> c=2;	>> z=x+c z = 4          6          8 Skalerle          vektör toplamı (OCTAVE a özgü bir işlem)
	>> z=c*x z = 4          8          12

### 3.3 Alt Vektör Tanımlama

Bazı uygulamalarda bir vektörün belirli bir kısmından oluşan bir alt vektörün kullanılması gerekebilir. Örneğin

```
>> x=[1,2,3,4,5,6,7,8,9];
```

ile tanımlanan  $x$  vektörünün tek indisli terimlerinden  $x_{tek}$ , çift indisli terimlerinden ise  $x_{cift}$  isimli bir vektör tanımlayabiliriz:

```
>> xtek=x(1:2:9)
```

```
xtek =
```

```
1          3          5          7          9
```

```
>> xcift=x(2:2:8)
```

```
xcift =
```

```
2          4          6          8
```

Dizinin en son elemanı için 'end' kelimesi kullanılabilir. Buna göre yukarıdaki işlem

```
>> xcift=x(2:2:end)
```

```
xcift =
```



2 4 6 8

yazılımı ile de gerçekleştirilebilmektedir.

Ayrıca  $x$  in belirli sayıda elemanından(örneğin ilk beş elemanı) oluşan vektör

```
>> xbes=x(1:5)
```

```
xbes =
```

1 2 3 4 5

ile tanımlanabilir.

### 3.4 Noktalı Vektör Operatörleri

Bilinen aritmetik operatörlerin(+, \*, -, /, ^) yanı sıra, bazı aritmetik işlemlerin daha etkin biçimde gerçekleştirilebilmesi için OCTAVE ortamında çarpma, bölme ve üs alma operatörlerinin noktalı versiyonları olarak noktalı çarpma(.\*), noktalı bölme(/) ve noktalı üs(^) operatörleri geliştirilmiştir. Noktalı çarpma operatörü aynı sayıda bileşene sahip olan iki vektörün karşılıklı elemanlarının çarpılmasıyla yeni bir vektör oluşturur. Benzer biçimde noktalı bölme  $x./y$  ile  $x$  vektörünün her bir bileşeninin  $y$  vektörünün karşılık gelen bileşenine bölünmesi suretiyle yeni bir vektör oluşturur. Şüphesiz bu durumda işlemin geçerli olabilmesi için  $y$  vektörünün bütün bileşenlerinin sıfırdan farklı olması gerekmektedir.  $x.^y$  şeklinde tanımlanan noktalı üs işlemi sonucunda ise  $x$  in her bir bileşeninin  $y$  nin karşılık gelen bileşeninci kuvveti alınarak yeni bir vektör oluşturur.  $y$  nin skaler olması durumunda  $x$  in her bir bileşeninin  $y$  skaleri ile belirtilen üssü alınır. Örnek olarak Tablo 3.2 yi inceleyiniz:

Tablo 3.2 Noktalı Operatörlerle Vektör Cebirsel İşlemler

X			Y			İşlem		
>> x=1:2:5			>> y=[1,2,3]			>> z=x./y		
x =			y =			z =		
1	3	5	1	2	3	1.00	1.5000	1.6667
						>> z=x.^2		
						z =		
						1	9	25

>> z=x.^y
z =
1 9 125
>> z=x.*y
z =
1 6 15

### 3.5 Vektör Fonksiyonları

Vektörler üzerinde tanımlanan ve lineer cebirde sıkça kullanılan bir vektöre ait önemli tanımlayıcı bilgiler arasında bu vektörün hangi uzayda olduğu ( $R^n$  veya  $C^n$ ); farklı ölçeklerle tanımlanabilse de vektörün orijine olan uzaklığı (*norm*) kavramları yer almaktadır. OCTAVE ortamında tanımlanmış olan bir vektör hakkında bu tür bilgileri veren *size*, *length*, *norm* gibi sıkça kullanılan vektör fonksiyonları mevcuttur.

*size* fonksiyonu bir vektörün bileşen sayısı ile birlikte satır veya sütun vektörü olup olmadığını da belirlemek için kullanılır.

*length* ise sadece vektörün bileşen sayısını verir.

*norm* fonksiyonu vektörün orijine olan uzaklığının bir ölçüsüdür

$n$ -boyutlu bir uzaydaki bir vektörün  $p$  normunun

$$\|x\|_p = \left( \sum_{i=1}^n |x_i|^p \right)^{\frac{1}{p}}, p = 1, 2, \dots$$

ile tanımlandığını, sonsuz normunun ise

$$\|x\|_\infty = \max_{1 \leq i \leq n} (|x_i|)$$

olarak tanımlandığını ve aralarında

$$\lim_{p \rightarrow \infty} \|x\|_p = \|x\|_\infty$$

bağıntısının mevcut olduğunu biliyoruz.  $p = 2$  değerine karşılık gelen norma Öklid normu adı verildiğini hatırlayalım. *norm(x, p)* vektörün  $p$  normunu hesaplar, *norm(x, 2) = norm(x)* dir.

Yukarıda kısaca tasvir edilen vektör fonksiyonları ve işlevlerini Tablo 3.3 ve Tablo 3.4 ü inceleyerek anlamaya çalışalım:

Tablo 3.3. Vektör tanımlayıcı fonksiyonlar

x=[1 2 3]	>> size(x)	>> length(x)	>> x'	>> size(x')
-----------	------------	--------------	-------	-------------

## Vektörler

x = 1 2 3	ans = 1 3	ans = 3	ans = 1 2 3	ans = 3 1
--------------	--------------	------------	----------------------	--------------

Tablo 3.3 den tanımlanan vektörün üç bileşenli bir satır vektörü olduğunu öğreniyoruz. Sütun vektörü oluşturmak için (`x'` : `x` üst karakter 2) komutunu uyguluyoruz.

Öte yandan `sum` fonksiyonu vektörün bileşenlerinin toplamını hesaplar. `Abs` fonksiyonu vektörün her bir elemanının mutlak değerini hesaplar.

Tablo 3.4 Sıkça kullanılan vektör fonksiyonları

x=[1 -2 3]	>> norm(x)	>> sum(x)	>> abs(x)
x = 1 -2 3	ans = 3.7417	ans = 2	ans = 1 2 3

Skalerler üzerinde daha önce incelediğimiz elemanter fonksiyonlar, vektörler üzerinde de tanımlıdır. Aşağıdaki örnekleri inceleyelim:

```
>> x=[1 0.5 0.7];

>> sin(x)

ans =
    0.8415    0.4794    0.6442

>> asin(x)
ans =
    1.5708    0.5236    0.7754

>> sinh(x)
ans =
    1.1752    0.5211    0.7586

>> asinh(x)
ans =
```

## Vektörler

```
0.88137    0.48121    0.65267
```

```
>> log(x)
ans =
      0    -0.6931    -0.3567
```

```
>> log10(x)
ans =
      0    -0.3010    -0.1549
```

```
>> exp(x)
ans =
  2.7183  1.6487  2.0138
```

### 3.6 Vektör Argümanlı Satır Fonksiyonu Tanımlama

Vektör argümanlı satır fonksiyonu tıpkı skaler argümanlı satır fonksiyonuna benzer biçimde tanımlanır. Tek fark,

- bir vektörün üssü için  $^$  yerine  $.^$
- vektör bileşenlerinin çarpım ve bölüm işlemi için sırasıyla  $*$  ve  $/$  yerine  $.*$  ve  $./$  noktalı vektör operatörlerinin kullanılmasıdır.

Örneğin

```
>> f=inline('x.^2-3*x+2')
```

```
f =
```

```
Inline function:
f(x) = x.^2-3*x+2
```

fonksiyonu vektör argümanlı bir fonksiyondur ve  $x$  skaler değişkeni yanısıra vektör değişkenini de kabul eder. Örneğin

```
>> x=-2:0.1:2;
>> y=f(x);
```

ile  $x$  vektörünün her bir değerinin  $f$  fonksiyonu altındaki resminden oluşan  $y$  vektörü tanımlanmaktadır.

Alternatif olarak skaler argümanlı satır fonksiyonu `vectorize` komutu ile vektör argümanlı satır fonksiyonuna dönüştürülebilir. Örneğin

```
>> g=inline('t/(t^2+1)');
```

satır fonksiyonu

```
>> gv=vectorize(g)
```

gv =

```
Inline function:
gv(t) = t./(t.^2+1)
```

olarak  $t$  skaleri yanısıra  $t$  vektörünü de argüman kabul edecek biçimde tanımlanabilir.

### 3.7 Plot Grafik Komutu

`plot(x,y,S)` komutu aynı sayıda bileşene sahip  $x$  vektörüne karşı  $y$  vektörünün grafiğini çizer.

$S$ : grafik çizim rengi (`r:red,g:gren,y:yellow,...`), işareti (`.,o,+,*,...`) ve çizgi tipini (`-,:,-.,--`) belirten karakterler kümesinin her birinden birer karakter olmak üzere en fazla üç karakterden oluşan bir karakter dizisidir. Bu dizi kullanılmak suretiyle aynı eksen de çizilmesi gereken grafikler ayırt edici özellikler kazanmış olurlar.

`plot(y)` ise  $y$  vektörünün kendi indisine karşı grafiğini çizer.

Eğer  $x$  kompleks elemanlı bir vektör ise

```
plot(x)=plot(real(x),imag(x))
```

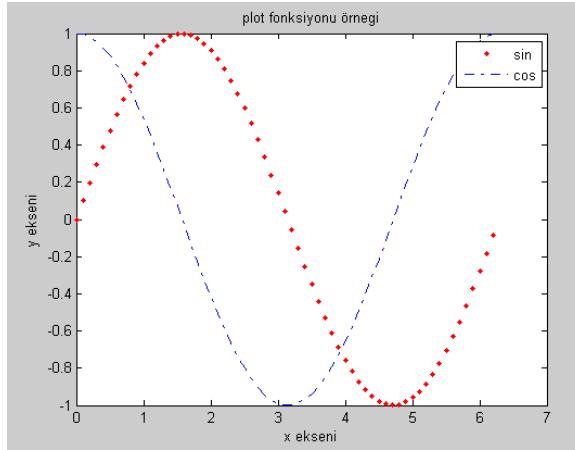
dir. Burada `real(x)` :  $x$  in reel kısmı, `imag(x)` ise sanal kısmını verir.

#### ÖRNEK 3.1

$\sin(x)$  ve  $\cos(x)$  fonksiyonlarının grafiklerini aynı eksen de ve fakat farklı çizgi deseni ve rengi ile çizelim.

```
>> x=0:0.1:2*pi;
>> y1=sin(x);
>> y2=cos(x);
>> plot(x,y1,'r.',x,y2,'b-.') % veya plot(x,y1,'r. ');
>> hold on;
>> plot(x,y2,'b-.');
>> legend('sin','cos')
>> xlabel('x eksen')
>> ylabel('y eksen')
>> title('plot fonksiyonu örneği')
```

komutları ile Şekil 3.1 de verilen grafik elde edilmektedir. Legend fonksiyonu grafik penceresinde oluşan küçük pencere yardımıyla hangi grafiğin hangi fonksiyona ait olduğunu belirlememize yardımcı olmaktadır. Xlabel ve ylabel ile sırasıyla x ve y eksenlerinin isimleri belirlenmekte, title ile de grafik başlığı belirlenmektedir. hold on komutu önceki grafik eksenini dondurarak, yeni grafiğin de aynı eksen de çizilmesini sağlar. Aksi takdirde önceki grafik silinerek, sadece yenisi grafik penceresinde gözükür.



Şekil 3.1  $\sin(x)$  ve  $\cos(x)$  fonksiyonlarının plot ile grafikleri

Noktalı üs operatörünün kullanımına ilişkin aşağıdaki örneği inceleyelim:

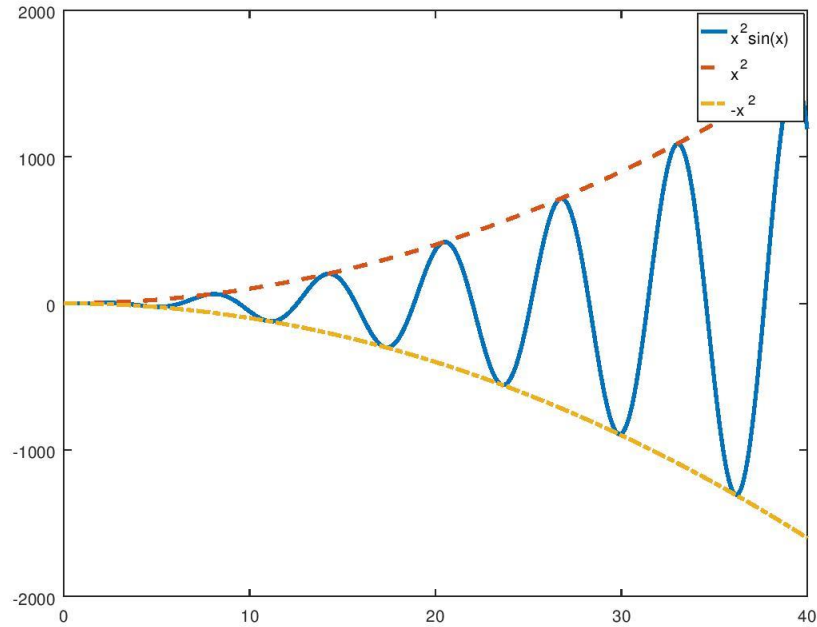
### ÖRNEK 3.2

Noktalı üs operatörü kullanmak suretiyle  $f(x) = x^2 \sin(x)$  ile  $g(x) = x^2$  ve

–  $g(x)$  fonksiyonlarının grafiğini aynı eksen de çizelim.

```
>> x=0:0.1:40;
>> y1=x.^2.*sin(x); % Noktalı çarpma operatörü!
>> y2=x.^2; % Noktalı üs alma operatörü!
>> plot(x,y1,'linewidth',2); hold on;
>> plot(x,y2,'--','linewidth',2);hold on;
>> plot(x,-y2,'-.','linewidth',2);
>> legend('x^2sin(x)', 'x^2', '-x^2');
```

**NOT** : legend içerisinde noktalı operatör kullanımı gerekmez, çünkü sonuç sadece ekrana yansıtılacaktır.



Şekil 3.2  $f(x) = x^2 \sin(x)$ ,  $g(x) = x^2$  ve  $-g(x)$  fonksiyonlarının grafiği.

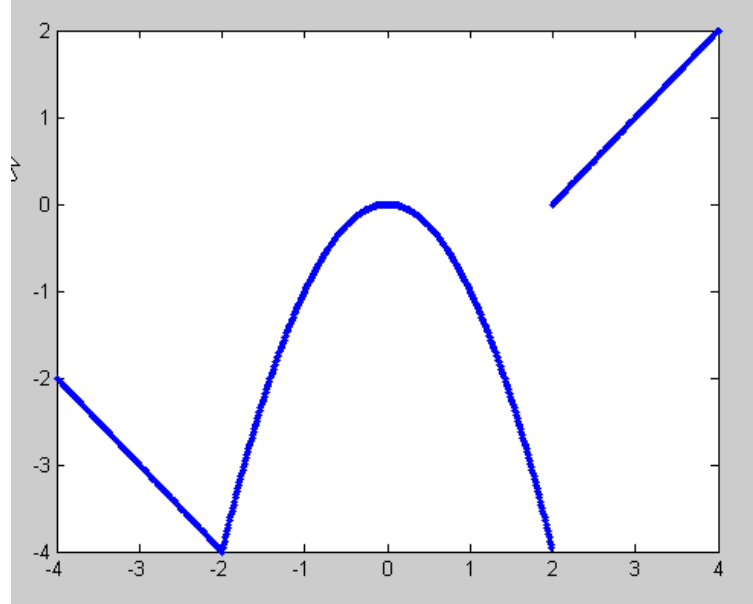
Parçalı sürekli fonksiyonları temsil eden vektörler ve grafikleri ise yukarıda bahsedilen noktalı operatörler ve mantıksal karşılaştırma operatörleri yardımıyla kolayca oluşturulabilir.

### **ÖRNEK 3.3**

## Vektörler

$$f(x) = \begin{cases} -x - 6, & x < -2 \\ -x^2, & -2 \leq x \leq 2 \\ x - 2, & x > 2 \end{cases}$$

parçalı fonksiyonunun grafiğinin nasıl çizildiğini inceleyelim.



Şekil 3.3. Parçalı sürekli f fonksiyonunu ve grafiği

```
>> x=-4:0.01:4;  
>> y=(-x-6).*(x<-2)+(-x.^2).*(x>=-2& x<=2)+(x-  
2).*(x>2);  
>> plot(x,y,'.')
```

$x$  vektörünün her bir noktasında  $f$  fonksiyonuna eşit olan  $y$  vektörünün nasıl tanımlandığını inceleyelim:

$(-x - 6) \cdot (x < -2)$  bileşeni ile  $(x < -2)$  değerleri için  $(x < -2)$  mantıksal karşılaştırmasının sonucu 1'e eşit olduğundan  $y = (-x - 6)$  olarak tanımlanmaktadır.

.....

Örneğin  $x=1$  değeri için

```
>> y=x<2
```

$y =$

1



komutu ile  $y$  değişkenine 1 değeri atanmaktadır. Fakat

```
>> y=x>1
```

```
y = 0
```

ile  $y$  değişkenine sıfır değerini atanmaktadır.

.....

$(-x.^2) .* (x \geq -2 \& x \leq 2)$  bileşeni ile  $[-2,2]$  aralığındaki  $x$  değerleri için  $(x \geq -2 \& x \leq 2)$  karşılaştırmasının sonucu bire eşit olacağından değeri  $-x^2$  ye eşit, diğer durumlarda ise sıfıra eşit olan bir vektör tanımlanmaktadır.

Üçüncü bileşen olan  $(x-2) .* (x > 2)$  ile de  $(x > 2)$  değerleri için değeri  $(x-2)$  ye ve diğer durumlarda ise sıfıra eşit olan bir vektör tanımlanmaktadır. Bu üç vektörün toplamından oluşan  $y$  vektörü ise tanımlanan  $x$  noktalarında değeri  $f$  ye eşit olan bir vektördür.

### 3.8 Subplot Fonksiyonu

Subplot fonksiyonu yardımıyla grafik penceresi alt pencerelere bölünerek bir grafik ortamında birden fazla grafik çizilebilir.

Subplot( $m, n, p$ ) komutu ekranı  $m$  satır ve  $n$  sütundan oluşan alt grafik pencerelerine bölerek,  $p$ -inci grafik üzerine odaklanır. Daha sonra girilen her türlü plot, xlabel, ylabel, title komutları  $p$ -inci grafiğe ait olur. Bir sonraki grafik için  $p$  değeri bir artırılarak işlemler gerçekleştirilir.

#### ÖRNEK 3.4

$$f = x^2, g = -x^2, h = (x + 1)^2, s = x^2 + 1, x \in [-2, 2]$$

fonksiyonlarının grafiklerini dört grafik penceresinde çizelim.

```
>> x=-2:0.1:2;
```

```
>> f=x.^2;
```

```
>> g=-x.^2;
```

```
>> h=(x+1).^2;
```

```
>> s=x.^2+1;
```

```
>> subplot(2,2,1)
```

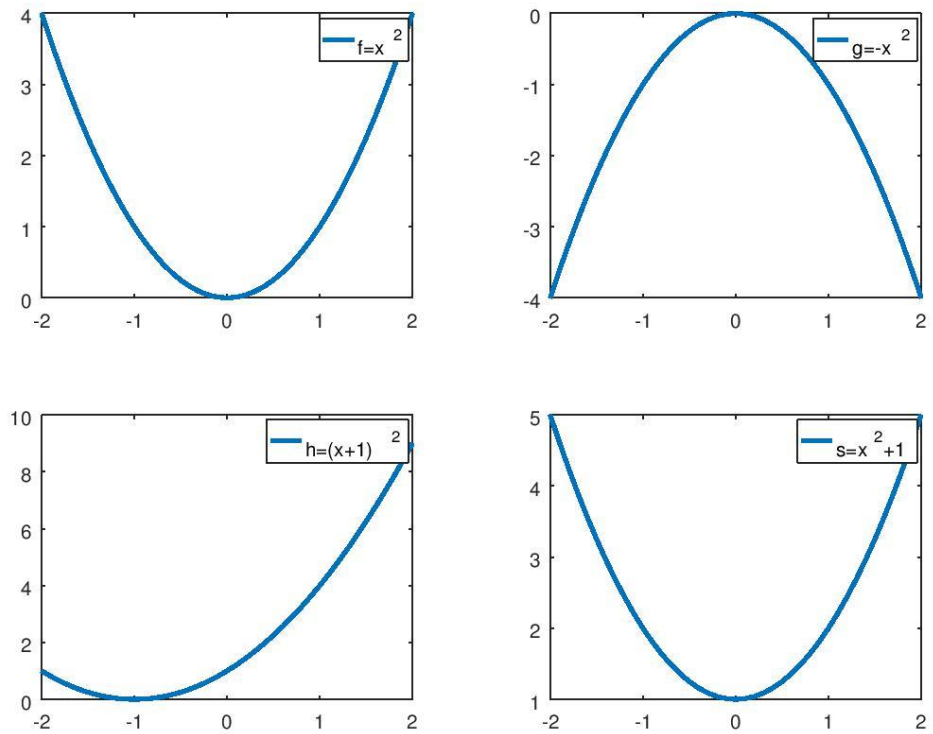
```
>> plot(x,f,'linewidth',2)
```

```
>> legend('f=x^2')
```

```
>> subplot(2,2,2)
>> plot(x,g, 'linewidth',2)
>> legend('g=-x^2')

>> subplot(2,2,3)
>> plot(x,h, 'linewidth',2)
>> legend('h=(x+1)^2')

>> subplot(2,2,4)
>> plot(x,s, 'linewidth',2)
>> legend('s=x^2+1')
```



Şekil 3.4. Subplot ile uygulama

### 3.9 Komut Dosyası Oluşturma

Örnek 3.4 teki grafiğin çizimi, çok sayıda komutun komut penceresinden girilmesini gerektirmektedir. Bu durumda herhangi bir düzenlemenin yapılması

oldukça zordur. Alternatif bir yaklaşım ise komut dosyası oluşturarak komut penceresinden girilecek olan komutların hepsini dosya içerisine yazıp, dosyayı .m uzantısı ile kaydettikten sonra, sadece dosya ismini(uzantı hariç) yazmak suretiyle dosya içerisindeki komutları çalıştırmaktır.

OCTAVE ortamında bu işlemi gerçekleştirmek için

1. Öncelikle OCTAVE klasörü içerisinde mathesap isimli bir klasör oluşturunuz.
2. `file`→`new` seçeneği ile yeni dosya açınız
3. Komutlarınızı açılan dosya içerisine yazınız. Başlangıç için Örnek 3.4 e ait komutları yazınız.
4. Dosyanızı `dortlugrafik.m` ismiyle kaydediniz
5. Komut penceresinde `dortlugrafik` yazarak enter tuşuna basınız.

```
>>dortlugrafik
```

komutuyla `dortlugrafik.m` isimli dosya içerisindeki komutları çalıştırabilirsiniz.

**NOT:** Ctrl tuşuna bası tutarak klavyenin en sağ orta kısmında yer alan “+” tuşuna birkaç kez basarak dosya içerisindeki yazı büyüklüğünü artırabilirsiniz.

**NOT:** Özellikle OCTAVE ortamında gerekli klasör yolları(path) tanımlanmamışsa, yukarıda `dortlugrafik` komutunu yazdığınız klasörün, `mathesap` klasörünüz olduğunu kontrol ediniz:

```
>>pwd(print working directory)
```

komutu o anda içerisinde çalışmakta olduğunuz klasörü belirtir. Bu klasörün `mathesap` isimli klasörünüz olduğuna dikkat ediniz. Eğer farklı bir klasör ise

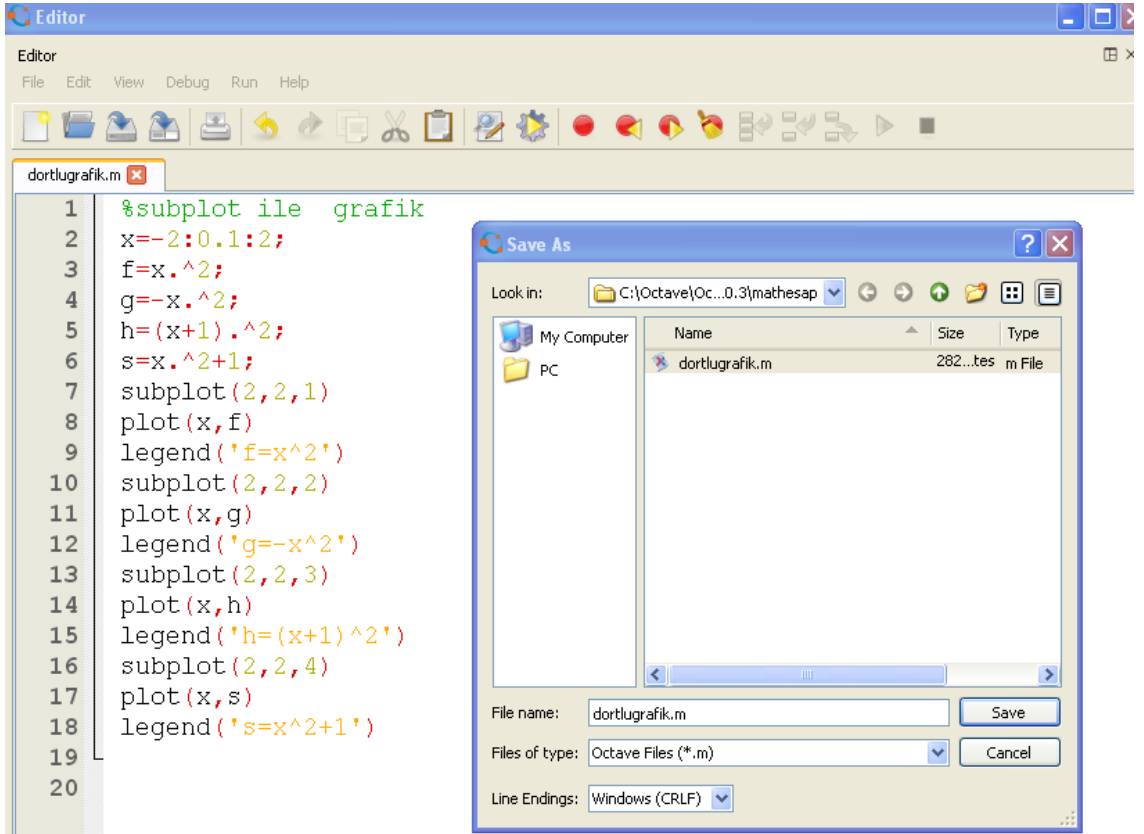
```
>>cd klasör yolu
```


Komutu ile istediğiniz klasöre gidebilirsiniz. Bulunduğunuz klasörden bir üst klasöre çıkmak için

```
>>cd ..
```

## Vektörler

komutunu kullanabilirsiniz



Alternatif olarak  simge grubundaki yukarı ok tuşu ile bir üst klasöre çıkabilir, dosya sembolü olan sağdaki simge(klasörlere göz at) ile de bulunduğunuz ortamdaki klasörlere göz atarak istediğiniz mathesap klasörüne erişebilirsiniz.

### 3. Bölüm Alıştırması

1.  $x = [1 \ 0 \ -1]$ ,  $y = [2 \ 1 \ 4]$  vektörleri ve  $c = 3$  skaleri için aşağıdaki işlemlerden tanımlı olanların sonucunu belirleyiniz.
  - a.  $x - c * y$
  - b.  $x + c$
  - c.  $x * y$
  - d.  $x' * y$
  - e.  $x * y'$
2. Soru 1 de verilen  $x$  ve  $y$  vektörleri için aşağıdaki işlemlerin sonucunu belirleyiniz.
  - a.  $x.*y$
  - b.  $x./y$
  - c.  $x.^3$
  - d.  $x.^y$
3.  $[-3,3]$  kapalı aralığında 15 adet noktada  $f(t) = -\frac{1}{3}t^3 + \frac{1}{2}t^2 + C$  fonksiyonunun aşağıda verilen  $C$  değerleri için grafiğini aynı ekseninde çizdiriniz.
  - a.  $C = 2$
  - b.  $C = 0$
  - c.  $C = -2$
4. Aşağıdaki fonksiyonların grafiklerini belirtilen aralıklarda çiziniz.  $x$  noktaları arasındaki uzaklığı  $h = 0.1$  olarak kabul ediniz.
  - a.  $f(x) = x/(x^2 + 1)$ ,  $[-4,4]$  aralığında çiziniz.
  - b.  $f(x) = \exp(-x)\sin(x)$ ,  $[-5,5]$
5.  $a = 2, 3, 4$  değerleri için  $f(x) = a^x$  fonksiyonunun grafiğini  $[-1,2]$  aralığında aynı grafik ekseninde çiziniz.
6. Şekil 3.4 te oluşturulan grafikleri komut dosyası oluşturmak suretiyle elde ediniz.
7.  $x = -2 * \pi : 0.1 : 2 * \pi$  için  $\sin(x)$ ,  $\sin(2x)$ ,  $\cos(x)$ ,  $\cos(2x)$  fonksiyonlarının grafiklerini aynı grafik ekranının dört ayrı penceresinde çizdiriniz. Bu işlemi ilgili komutları `deneme.m` isimli bir komut dosyası ile gerçekleştiriniz.
8.  $x = [1 \ 2 \ 4 \ 0 \ -2 \ 9]$  vektörü için aşağıda belirtilen normları hesaplayarak normlar arasındaki ilişkiyi belirlemeye çalışınız. Gözleminizi seçeceğiniz diğer vektörler üzerinde kontrol ediniz.
  - a. `norm(x, 1)`
  - b. `norm(x, 2)`
  - c. `norm(x, 4)`
  - d. `norm(x, 10)`
  - e. `norm(x, inf)`
9.  $a = [2,3]$  ve  $b = [1,1]$  vektörü için
  - a.  $\cos(\theta) = \frac{a \cdot b}{\|a\|_2 \|b\|_2}$  ile tanımlanan  $\theta$  açısını belirleyiniz.

- b.  $a$  vektörünün  $b$  vektörü yönündeki skaler izdüşümünü(yani  $\|a\| \cos(\theta)$  veya  $\frac{a \cdot b}{\|b\|_2}$ ) hesaplayınız
- c.  $a$  vektörünün  $b$  vektörü yönünde  $\text{Pr}_b a = \frac{a \cdot b}{\|b\|_2} \cdot \frac{b}{\|b\|_2}$  ile verilen vektörel izdüşümünü hesaplayınız.
- d.  $a - \text{Pr}_b a$  vektörü ile  $b$  vektörünün ortogonal olduğunu gösteriniz
10.  $a = [1, 2, 1]$  ve  $b = [2, -1, 3]$  vektörleri için
- $a \times b = \text{cross}(a, b)$  ile tanımlanan vektörel çarpımını hesaplayınız.
  - $a \times b$  nin hem  $a$  ve hem de  $b$  ye dik olduğunu görünüz.
11.  $\|a \times b\| = \|a\| \|b\| \sin(\theta)$ ,  $0 \leq \theta \leq \pi$  bağıntısının doğruluğunu Soru 10 daki  $a$  ve  $b$  vektörleri için kontrol ediniz.
12.  $f(x) = \begin{cases} x^2, & x > 0 \\ -x^2, & x \leq 0 \end{cases}$  parçalı fonksiyonunun  $[-2, 2]$  aralığındaki grafiğini çiziniz.
13.  $f(x) = \sin(x)$  olarak tanımlansın.  $f(x+2)$ ,  $f(x-2)$ ,  $f(x)+2$  ve  $f(x)-2$  fonksiyonlarının grafiklerini aynı ekseninde ve  $[-5, 5]$  aralığında komut dosyası oluşturmak suretiyle çizdiriniz.
- Grafikler arasında nasıl bir ilişki gözlemliyorsunuz?
14. Soru 13 ü komut dosyası oluşturmak suretiyle tekrar ediniz.
15. Kutupsal koordinatlarda  $\theta$  vektörüne karşı  $r$  vektörünün grafiği polar fonksiyonu ile  $\text{polar}(teta, r)$  formatında yazılarak çizilir. Buna göre  $\theta = 0:0.1:\pi$  için aşağıda tanımlanan  $r = f(\theta)$  vektör değerli fonksiyonlarının grafiğini çizdiriniz.
- $r = \theta$
  - $r = \sin(\theta)$
  - $r = \cos(\theta)$
  - $r = 1 + \cos(\theta)$

## Vektörler





## IV. BÖLÜM

### 4.Vektörlerle İşlemler

Bu bölümde programlama dillerinde alışık olduğumuz döngüleri kullanmaksızın vektör cebirsel işlemlerle

- ☒ Sayısal türev
- ☒ sayısal integrasyon ve
- ☒ bilinmeyen ara değeri tahmin gibi temel işlemleri inceliyoruz.

#### 4.1 Sayısal Türev

Bir  $x$  noktasını içeren açık aralıkta tanımlı  $f$  fonksiyonu için

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

sonlu limiti mevcut ise, bu limite  $f$  fonksiyonunun  $x$  noktasındaki türevi adı verildiğini hatırlayalım. Bu tanımdan hareketle yeterince küçük pozitif  $h$  değeri için

$$f'(x) \cong \frac{f(x+h) - f(x)}{h}$$

kabul edilebilir.

$$f[x, x+h] = \frac{f(x+h) - f(x)}{h}$$

notasyonu ile gösterilen bölünmüş farka, verilen fonksiyonunun  $x$  noktasında ileri fark yöntemi ile sayısal türevi adı verilmektedir.

## Vektörler

Herhangi bir  $[a, b] \subset \mathbb{R}$  aralığını  $n$  adet alt aralığa bölelim ve elde edilen alt aralıkların uç noktalarını bileşen kabul eden vektörü  $x$  ile gösterelim.  $h = (b - a)/n$  olmak üzere  $x$  vektörü OCTAVE ortamında

```
>>x=a:h:b;
```

ile tanımlanır.  $x$  vektörünün  $n + 1$  adet bileşene sahip olduğuna dikkat edelim:

$$x = (x_1 = a, x_2 = a + h, x_3, \dots, x_{n+1} = b)$$

Herhangi bir  $f$  fonksiyonunun  $x$  vektörünün uç noktalarındaki değerlerinden oluşan vektörü  $y$  ile gösterelim:

$$y = (y_1, y_2, \dots, y_{n+1}), y_i = f(x_i), \quad i = 1, 2, \dots, n + 1$$

Yukarıdaki yaklaşıma göre

$$\begin{aligned} f'(x_1) &\cong \frac{f(x_1 + h) - f(x_1)}{h} = \frac{y_2 - y_1}{h} \\ f'(x_2) &\cong \frac{f(x_2 + h) - f(x_2)}{h} = \frac{y_3 - y_2}{h} \\ &\vdots \\ f'(x_n) &\cong \frac{f(x_n + h) - f(x_n)}{h} = \frac{y_{n+1} - y_n}{h} \end{aligned}$$

O halde sayısal türev vektörü OCTAVE notasyonu ile aşağıdaki gibi ifade edilebilir:

$$\begin{aligned} [f'(x_1), f'(x_2), \dots, f'(x_n)] &\cong \frac{1}{h} [y_2 - y_1, y_3 - y_2, \dots, y_{n+1} - y_n] \\ &= \frac{1}{h} ([y_2, y_3, \dots, y_{n+1}] - [y_1, y_2, \dots, y_n]) \\ &= \frac{1}{h} (y(2:n+1) - y(1:n)) \end{aligned}$$

### ÖRNEK 4.1

$f(x) = \sin(x), x \in [0, 2\pi]$  fonksiyonunun belirtilen aralıkta  $h = 0.1$  için sayısal türevlerini hesaplayarak, gerçek ve sayısal türevi grafiğini aynı eksende çizelim.

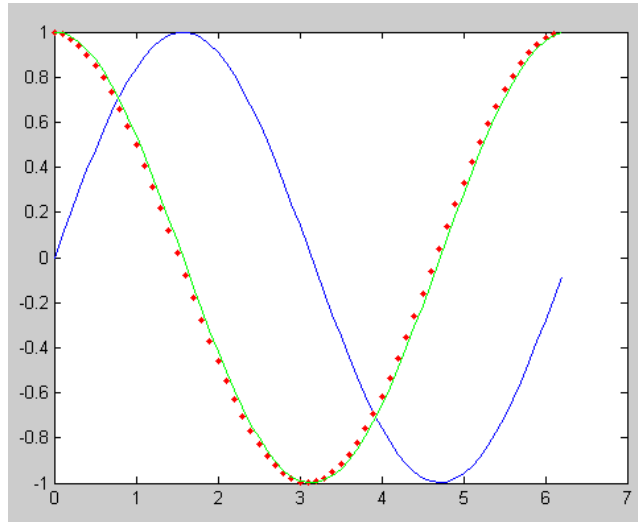
$h = 0.1$  için aralığın uç noktalarından oluşan vektörü  $x$ , bu noktalardaki fonksiyon değerlerinden oluşan vektörü  $y$  ve yaklaşık sayısal türev vektörünü  $yp$

ile gösterelim. Yukarıda belirtildiği üzere, fonksiyonun aralığın uç noktalarındaki değerleri ve uç noktalardaki sayısal türev aşağıdaki gibi tanımlanabilir:

```
>>h=0.1;
>>x=0:h:2*pi;
>>y=sin(x);
>>n=length(x)-1;
>>yp=(y(2:n+1)-y(1:n))/h;
```

$f$  fonksiyonunun sayısal ve gerçek türevinin grafiği aşağıda sunulmaktadır:  $y$  vektörü  $n + 1$  adet bileşene sahip olmasına rağmen,  $yp$  vektörünün  $n$ -adet bileşene sahip olduğuna dikkat edelim. Yani sayısal türev işlemi  $x$  in ilk  $n$ -adet bileşiminde tanımlıdır. Dolayısıyla aşağıda  $xx$  ile  $x$  in  $n$ -adet bileşeninden oluşan alt vektörü tanımlıyoruz:

```
>> xx=x(1:n);
>> plot(xx,yp,'.') % sayısal türevinin grafiği
>> hold on, plot(x,cos(x),'g') %cos(x) fonksiyonunu
grafiği
```

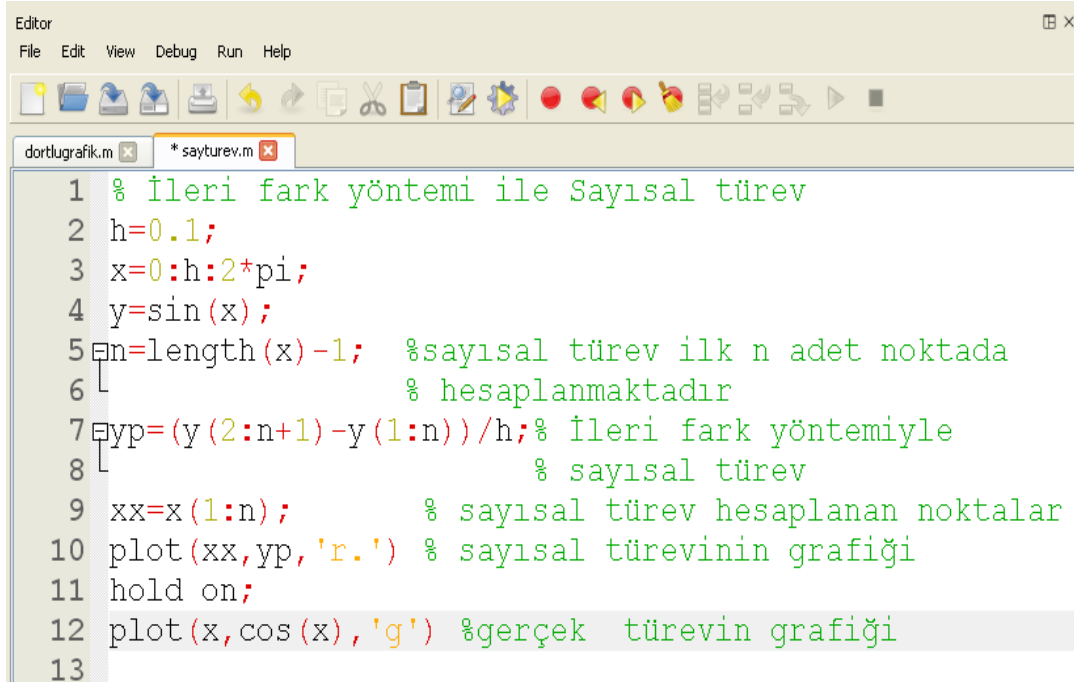


Şekil 4.1  $\sin(x)$ (çizgi), sayısal türevi(nokta) ve gerçek türevi(noktalara yakın çizgi).

Yukarıdaki komutlar, Şekil 4.2 de görüldüğü üzere `sayturev.m` isimli dosya içerisine yazılarak

```
>>sayturev
```

komutu ile de çalıştırılabilir.



```

1 % İleri fark yöntemi ile Sayısal türev
2 h=0.1;
3 x=0:h:2*pi;
4 y=sin(x);
5 n=length(x)-1; %sayısal türev ilk n adet noktada
6 % hesaplanmaktadır
7 yyp=(y(2:n+1)-y(1:n))/h;% İleri fark yöntemiyle
8 % sayısal türev
9 xx=x(1:n); % sayısal türev hesaplanan noktalar
10 plot(xx,yyp,'r.') % sayısal türevinin grafiği
11 hold on;
12 plot(x,cos(x),'g') %gerçek türevin grafiği
13

```

Şekil 4.2 sayturev.m dosya içeriği

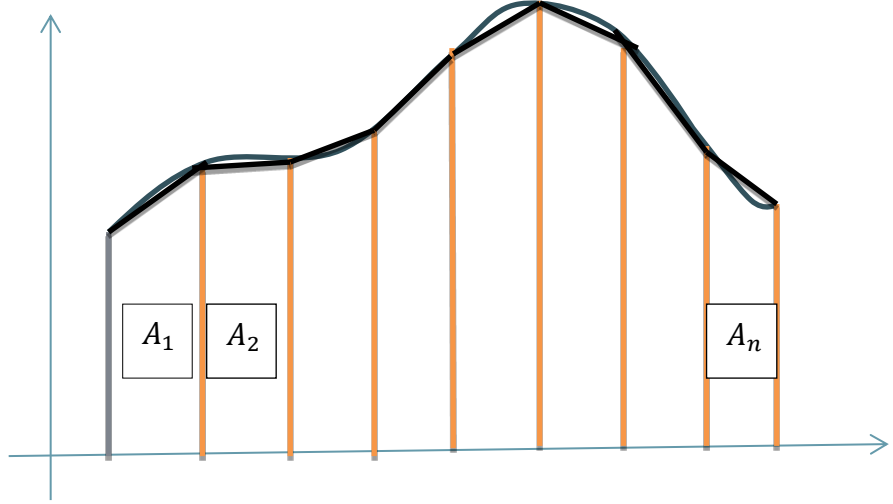
## 4.2 Sayısal İntegral(Eğri altındaki alan)

Bir fonksiyon ile  $x$ -ekseni arasında kalan alan hesabı için çeşitli ileri düzey yöntemler ve OCTAVE fonksiyonları mevcuttur ve bu konu Sayısal Analizin önemli konuları arasında yer almaktadır. Burada basit bir yamuk alan formülü yardımıyla belirtilen alan hesabının OCTAVE vektör notasyonu kapsamında nasıl yapılabileceğini inceleyeceğiz.

$f: [a, b] \rightarrow R$  tanımlı ve grafiği Şekil 4.3 de verilen bir fonksiyon olsun.  $[a, b]$  aralığını  $n$  adet alt aralığa bölerek elde edilen eşit uzunluklu alt aralıkların uç noktalarını  $x = [x_1, x_2, \dots, x_{n+1}]$  ile, her bir alt aralığın uzunluğunu  $h = \frac{b-a}{n}$  ve bu noktalardaki fonksiyon değerlerini  $y_i = f(x_i), i = 1, 2, \dots, n+1$  ile gösterelim. Fonksiyon grafiği ile  $x$  eksenı arasında kalan ve  $x = a$  ve  $x = b$  doğruları ile sınırlı bölgenin alanı için bir yaklaşım elde etmek istiyoruz.  $S(A)$  ile göstereceğimiz gerçek değerin ise

$$S(A) = \int_a^b f(x) dx$$

belirli integraline eşit olduğunu biliyoruz.



Şekil 4.3  $S(A)$  alanı için  $A_1, A_2, \dots, A_n$  yamukları

Şekil 4.3 de de sunulduğu üzere[10]

$$\begin{aligned} S(A) &\cong S(A_1) + S(A_2) + \dots + S(A_n) \\ &= \frac{h}{2} (f(x_1) + f(x_2)) + \frac{h}{2} (f(x_2) + f(x_3)) + \dots + \frac{h}{2} (f(x_n) + f(x_{n+1})) \\ &= h \left( \frac{1}{2} f(x_1) + f(x_2) + \dots + f(x_n) + \frac{1}{2} f(x_{n+1}) \right) \\ &= h \left( \frac{1}{2} y_1 + y_2 + \dots + y_n + \frac{1}{2} y_{n+1} \right) \end{aligned}$$

olarak elde edilir.

O halde OCTAVE notasyonu ile kullanıcı tarafından girilen  $n$  alt aralık sayısı için  $n + 1$  noktadaki fonksiyon değerlerinden oluşan  $y = [y_1, y_2, \dots, y_{n+1}]$  vektörü için yaklaşık alan

```
>>h=(b-a)/n;
>>Alan=h*(0.5*(y(1)+y(n+1))+sum(y(2:n)));
```

ile verilir.

### **ÖRNEK 4.2**

$y = f(x) = x^2, x \in [0,1]$  ile tanımlı fonksiyonun grafiği ve  $x$  eksenı arasında kalan bölgenin alanını  $n = 100$  alt aralık ile Yamuk yöntemi yardımıyla hesaplayalım.

Belirtilen alanın

$$\int_0^1 x^2 dx$$

integrali için de bir yaklaşım olacağına dikkat edelim.

```
>> a=0; b=1;% Aralığın uç noktaları
>> n=100;% Alt aralık sayısı
>> h=(b-a)/n; % Alt aralık uzunlukları
>> x=0:h:1;% Alt aralık uç noktalar vektörü
>> y=x.^2;% fonksiyon değerlerinden oluşan vektör
>> Alan=h*(0.5*(y(1)+y(n+1))+sum(y(2:n))) % n adet
yamuk
% alan

toplamı
Alan =
0.3333
```

**NOT:** Gerçek değerin

$$\int_0^1 x^2 dx = \frac{1}{3}$$

olduğunu hatırlayalım.

## **4.3 Bilinmeyen Ara değerin Tahmini(Interpolasyon)**

Bir deney sonucu olarak  $(t_i, y_i), i = 1, 2, \dots, n$  verilerini elde ettiğimizi düşünelim. Bu veriler, örneğin  $t_i$  zamanlarında ölçülen fiziksel, kimyasal veya biyolojik bir deney sonucu elde edilen gözlemlenen  $y_i$  değerleri olabilir. Ölçüm yapılması gereken herhangi bir  $t_k$  anında söz konusu ölçümün

gerçekleştirilemediğini ve fakat elde edilen değerler yardımıyla eksik değeri bilimsel olarak tahmin etmek istediğimizi düşünelim. Bu işleme interpolasyon işlemi adı verilmektedir(Bkz [10]). Interpolasyon işlemi OCTAVE ortamında `interp1`, `interp2` gibi fonksiyonu yardımıyla etkin bir biçimde gerçekleştirilebilmektedir.

### ÖRNEK 4.3

Sabah saat 8:00 da başlamak üzere 17:00 ye kadar her saatte bir hastanın ateş ölçümünü, saat 13:00 daki ölçümü atlamak suretiyle  $y=[35 \ 32 \ 37 \ 40 \ 35 \ 38 \ 41 \ 34 \ 40]$  olarak kaydettiğimizi kabul edelim. Saat 13:00 deki ölçüm değerini interpolasyon yardımıyla tahmin edelim.

```
>> t=[8 9 10 11 12 14 15 16 17]; %Ölçüm yapılan
    saatleri belirten vektör
>> y=[35 32 37 40 35 38 41 34 40]; %Ölçüm değerleri
>> tt=13; %Ölçüm değeri bilinmeyen
    nokta
>> yy=interp1(t,y,tt)
```

yy =

36.5000 % Tahmin edilen değer

Belirtilen noktadaki değer tahmin edilirken arka planda değişik tahmin yöntemleri kullanılır:

linear, cubic, spline, pcubic.

Bu seçenekler

```
>> yy=interp1(t,y,tt,'seçenek')
```

formatında belirtilir. Seçenek belirtilmediği zaman 'linear' seçeneği kullanılır ve bilinmeyen nokta komşuluğundaki iki nokta arasında değişimin lineer(doğrusal) olduğu kabul edilerek tahmin gerçekleştirilir. Diğer seçenekler verilen noktalardan geçen üçüncü dereceden polinomlar(cubic) veya her bir alt aralıkta üçüncü dereceden polinomlar (piecewise cubic) gibi daha ileri düzey yaklaşımlar kullanırlar.

```
>> yy=interp1(t,y,tt,'linear')
```

## Vektörler

```
yy =  
    36.5000  
  
>> yy=interp1(t,y,tt,'spline')  
  
yy =  
    33.9886  
  
>> yy=interp1(t,y,tt,'cubic')  
  
yy =  
    35.9808  
  
>> yy=interp1(t,y,tt,'pcubic')  
  
yy =  
    35.9808
```

Farklı yöntemlerin farklı tahminler ürettiklerine dikkat edelim.

Ölçüm yapılan noktalar dışında bir noktadaki değer tahmin etme işlemine ise ekstrapolasyon adı verilmektedir.

[t,y] ölçüm sonuçlarına göre ölçüm noktaları dışında yer alan bir tt noktasındaki tahmin işlemi

```
>> yy=interp1(t,y,tt,'pcubic','extrap')
```

ile gerçekleştirilir. Yukarıdaki verilerde saat 18:00 deki ölçüm sonucunu tahmin etmek için

```
>> yy=interp1(t,y,18,'linear','extrap')  
  
yy =  
    46
```

kullanılabilir. Yine kullanılan yöntemle göre değişik tahminler elde edilmektedir.



Interpolasyon veya Ekstrapolasyon işlemi birden fazla noktada aynı anda gerçekleştirilebilir:

### ÖRNEK 4.4

Interpolasyon işlemi çok sayıda noktada ve aynı anda gerçekleştirilebilir. Aşağıda verilen her bir  $t$  noktasında, aynı indisli  $y$  değerlerine eşit olan sıralı ikilileri 0.1 adım uzunluklu her nokta için tahmin edelim.

```
>> t=8:17;  
>> y=[35 32 37 40 35 38 41 34 40 36];
```

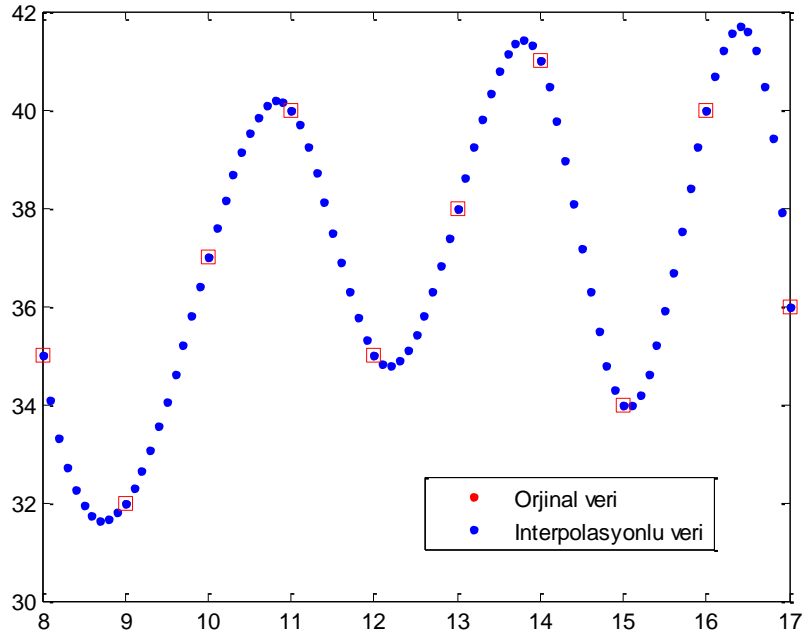
Şekil 4.3 de kırmızı kareler içerisinde gösterilen  $(t, y)$  değerleri

```
>> tt=8:0.1:17
```

vektörüne ait noktalarda aynı anda tahmin edilerek

```
>> yy=interp1(t,y,tt,'spline');
```

vektörü elde edilmiş ve sonuç mavi noktalarla şekilde gösterilmiştir.



Şekil 4.3. Örnek 4.4 e ait(t,y)( kare) ve (tt,yy)( nokta) grafikleri

## 4. Bölüm Alıştırmaları

- Aşağıda verilen fonksiyonların belirtilen aralıklardaki sayısal türevlerini ileri fark yöntemiyle  $h = 0.1$  adım uzunluklarını kullanmak suretiyle hesaplayınız. Gerçek türev ve sayısal türevin grafiklerini aynı eksende farklı çizim renkleriyle çizdiriniz.
  - $f(x) = \cos(x), [-2\pi, 2\pi]$
  - $f(x) = x^2 - 2x + 1, [-2, 2]$
  - $f(x) = \exp(-x^2), [-4, 4]$
  - $f(x) = \frac{x}{1+x^2}, [-3, 3]$
- Bir  $f$  fonksiyonunun  $[a, b]$  aralığındaki ortalama değerinin  $\frac{1}{b-a} \int_a^b f(x) dx$  integrali olarak tanımlandığını hatırlayalım. Buna göre aşağıdaki fonksiyonların belirtilen aralıklardaki ortalama değerlerini hesaplayınız. İntegral işleminde  $n$  ile belirtilen sayıda alt aralık kullanınız.
  - $f(x) = x^2, [0, 2], n = 10$
  - $f(x) = \cos(x), [-\pi, \pi], n = 20$
  - $f(x) = \frac{1}{x^2}, [1, 3], n = 10$

3.  $f(x) = \sin(x^2)$  fonksiyonunun  $[0,1]$  aralığındaki integralini
  - a)  $n = 10$
  - b)  $n = 20$
  - c)  $n = 40$
 değerleri için hesaplayınız.  
 Belirtilen integrali bildiğiniz yöntemlerle nasıl hesaplayabilirdiniz?
4.  $f(x) = \sinh(x^2)$  fonksiyonunun  $[-\frac{\pi}{2}, \frac{\pi}{2}]$  aralığındaki integralini  $h = 0.01$  değeri için hesaplayınız. Bu amaçla öncelikle  $n$  değerini belirleyiniz.
5.  $f(x) = \sin(x^3)$  fonksiyonunun  $[-3,3]$  aralığındaki integralin  $h = 0.01$  değeri için hesaplayınız. Bu amaçla öncelikle  $n$  değerini belirleyiniz.
6. Soru 3 de görüldüğü üzere integral işleminde kullanılan nokta sayısına göre elde edilen sonuç değişmektedir. Ancak  $n$  değerini artırmaya devam ettiğinizde elde ettiğiniz sonuç belirli bir  $n$  den sonra daha fazla değişmeyecektir. Soru 4 ve Soru 5 de verilen integraller için integral sonucunun değişmediği en küçük  $n$  değerlerini belirleyiniz.(İşlemlerde virgülden sonra dört basamağın gösterildiği kısa formatta çalışınız.)
7.  $t = [8\ 9\ 10\ 11\ 12\ 13\ 15\ 16]$  ile belirtilen saatlerde  $y = [32\ 34\ 36\ 34\ 35\ 37\ 38\ 37]$  ölçüm değerleri verilmiş olsun.

- a) *linear* interpolasyon seçeneği ile  $tt = 14$  değerine karşılık gelen eksik ölçüm değerini tahmin ediniz.
- b) *cubic* interpolasyon seçeneği ile  $tt = 14$  değerine karşılık gelen eksik ölçüm değerini tahmin ediniz.
- c) *spline* interpolasyon seçeneği ile  $tt = 14$  değerine karşılık gelen eksik ölçüm değerini tahmin ediniz.

8. 1990 yılı  $t = 0$  başlangıç yılı kabul edilmek üzere her beş yılda bir yapılan nüfus sayımına göre bir beldenin nüfus verileri aşağıdaki gibi verilmektedir.

t	Nüfus
0	2350
5	2400
10	2490
15	3100
20	2950

- a) Bu verilere göre beldenin 2001 yılındaki nüfusu 'spline' seçeneği ile ne olabilir? (Önce  $t$  değerini belirleyiniz)
- b) 2011 yılı nüfusunun ne kadar olmasını beklersiniz?

## Vektörler

# V. BÖLÜM

## 5. Matrisler ve Matrislerle İşlemler

Bu bölümde OCTAVE ortamında

- ☑ Matrisler ve elemanter matris işlemleri,
- ☑ matrislerle ilgi dört temel uzaya ait taban vektörlerinin elde edilişi ve
- ☑ matrisler üzerinde diğer lineer cebirsel işlemlerin nasıl gerçekleştirildiğini inceliyoruz.

### 5.1 Matris Tanımlama

Matrisler OCTAVE ortamında değişik biçimlerde tanımlanabilirler. Küçük boyutlu matrislerde aynı satırdaki elemanlar boşluk veya virgül ile, farklı satırlar ise noktalı virgül ile ayrılmak suretiyle aşağıdaki gibi tanımlanabilirler:

```
>> A=[1 2;3 4]
```

A =

```
1 2
3 4
```

```
>> B=[1 -1;0 4]
```

B =

```
1 -1
0 4
```

```
>> C=zeros(2,3) %Sıfırlardan oluşan iki satır ve üç sütunluk matris
```

C =

## Matrislerle İşlemler

```
0    0    0
0    0    0
```

```
>> D=ones(3,2) %Bir rakamlarından oluşan üç satır ve iki sütunluk matris
```

```
D =
     1     1
     1     1
     1     1
```

```
>> I=eye(2) %İki boyutlu birim matris
```

```
I =
     1     0
     0     1
```

```
>>d=[1 2 3];
>>D=diag(d) % Köşegen matris
D =
```

```
1    0    0
0    2    0
0    0    3
```

Üç köşegenli olup, köşegen üzerindeki elemanlı -2, ve diğerleri ise 1'e eşit olan şerit matris olarak adlandırılan bir matris `spdiags` fonksiyonu yardımıyla aşağıdaki gibi oluşturulabilir :

```
>>e=[1 1 1]';
>> spdiags([e -2*e e],[-1:1,3,3])
```

```
ans =
(1,1)    -2
(2,1)     1
(1,2)     1
(2,2)    -2
(3,2)     1
(2,3)     1
(3,3)    -2
```

## Matrislerle İşlemler

Matrisi sıfır elemanları ile birlikte ve matris formatında görüntülemek için full komutu kullanılır:

```
>> full(ans) %Özel bir şerit matris
```

```
ans =
```

```
-2     1     0
 1    -2     1
 0     1    -2
```

Elemanları 0 ile 1 arasında olan rasgele bir matris ise aşağıdaki gibi oluşturulabilir:

```
>> R=rand(3,2) %Rasgele elemanlardan oluşan üç satır ve iki sütunluk matris
```

```
R =
```

```
0.9501    0.4860
0.2311    0.8913
0.6068    0.7621
```

## 5.2 Matrisler Üzerinde Aritmetik İşlemler

Tablo 5.1. Matrisler üzerinde tanımlı işlemler

İşlem	Anlamı	Sonuç
A+B	Aynı boyutlu iki matrisin toplamı	<pre>&gt;&gt; C=A+B  1     1  3     8</pre>
A-B	Aynı boyutlu iki matrisin farkı	<pre>&gt;&gt; C=A-B  0     3  3     0</pre>
A*B	Matrislerinin çarpımı (A'nın sütun sayısının B'nin satır sayısına eşit olması gerektiğine dikkat edelim)	<pre>&gt;&gt; C=A*B  1     7  3    13</pre>
A+2	OCTAVE da bir skaler ile bir matris toplanabilir (Bu işlem lineer cebirsel bir	<pre>&gt;&gt; C=A+2  3     4</pre>

## Matrislerle İşlemler

	<i>işlem değildir! Sadece OCTAVE da tanımlanan bir işlemdir!)</i>	5      6
3*A	Bir skaler ile bir matrisin çarpımı	>> C=3*A 3      6 9      12
A^2	Matrisin karesi (A*A)	>> A^2 7      10 15      22
A.^2	Matrisin her bir elemanının karesinden oluşan matris	>> A.^2 1      4 9      16

### MATRİS VE VEKTÖR ÇARPIMI

$A_{m \times n}$  matris ve  $X_{n \times 1}$  vektör olmak üzere  $A$  ile  $X$  in çarpımı  $A$  matrisinin sütunlarının bir lineer kombinasyonudur:

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \text{ ve } X = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \text{ olmak üzere}$$

$$\begin{aligned} AX &= \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{bmatrix} \\ &= x_1 \begin{bmatrix} a_{11} \\ \vdots \\ a_{m1} \end{bmatrix} + x_2 \begin{bmatrix} a_{12} \\ \vdots \\ a_{m2} \end{bmatrix} + \cdots + x_n \begin{bmatrix} a_{1n} \\ \vdots \\ a_{mn} \end{bmatrix} \end{aligned}$$

olduğu kolayca görülür[8,9].

### ÖRNEK 5.1

$A = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$  matrisi ve  $X = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$  vektörü için yukarıda tanımlanan matris ve vektör çarpımının doğruluğunu OCTAVE ortamında kontrol ediniz.



```
>> X=[1 -1]'
```

```
X =
```

```
    1
   -1
```

```
>> A*X
```

```
ans =
```

```
   -1
    1
```

```
>> A(:,1)*X(1)+A(:,2)*X(2)
```

```
ans =
```

```
   -1
    1
```

### MATRİSLERİN ÇARPIMI(ALTERNATİF YAKLAŞIM)

$A_{m \times n}$  ve  $B_{n \times k}$  matrisleri verilsin.  $B$  nin sütunları  $b_1, b_2, \dots, b_k$  olmak üzere  $AB$  çarpımı bilinen satır-sütun iç çarpım kuralına alternatif olarak

$$AB = [Ab_1 \ Ab_2 \ \dots \ Ab_k]$$

biçiminde tanımlanabilir. Öte yandan  $b_i = [b_{i1} \ b_{i2} \ \dots \ b_{in}]^T$  olmak üzere

$$Ab_i = [a_1 \ a_2 \ \dots \ a_n] b_i = b_{i1}a_1 + b_{i2}a_2 + \dots + b_{in}a_n$$

olarak yazılabildiğine dikkat edelim. O halde bir  $A$  matrisinin  $X$  vektörü ile çarpımı,  $A$  matrisinin sütunları ve  $X$  vektörünün bileşenleri ile oluşturulan bir lineer kombinasyondur.  $AB$  nin  $i$ -inci sütunu ise  $A$  matrisinin sütunları ile  $B$  nin  $i$ -inci sütununun bileşenleri ile oluşturulan lineer kombinasyondur.

### ÖRNEK 5.2

## Matrislerle İşlemler

Yukarıda verilen matris çarpımın tanımının doğruluğunu  $A = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}, B =$

$\begin{bmatrix} 1 & 3 \\ 3 & 1 \end{bmatrix}$  matrisleri için OCTAVE ortamında kontrol ediniz.

```
>> b1=B(:,1)
```

```
b1 =
```

```
1
3
```

```
>> b2=B(:,2)
```

```
b2 =
```

```
3
1
```

```
>> A*b1
```

```
ans =
```

```
7
5
```

```
>> A*b2
```

```
ans =
```

```
5
7
```

```
>> sonuc=[A*b1 A*b2]
```

```
sonuc =
```

```
7    5
5    7
```

```
>> A*B
```

```
ans =
```

```
7    5
5    7
```

### 5.3 Alt Matris Tanımlama

Bir matrisin herhangi satır veya sütunlarından oluşan bir yeni bir matris (alt matris) elde edilebilir. Örneğin  $R$  matrisinin birinci ve ikinci satır ve sütunlarından oluşan bir alt matris

```
>> RR=R(1:2,1:2)
```

```
RR =
```

```
    0.9501    0.4860
    0.2311    0.8913
```

olarak elde edilir. İndisler değiştirilmek suretiyle istenilen alt matris elde edilebilir. Özel olarak  $R$  nin birinci satırı

```
>> R(1, : )
```

```
ans =
```

```
    0.9501    0.4860
```

$R$  nin birinci sütunu

```
R(:,1)
```

```
ans =
```

```
    0.9501
    0.2311
    0.6068
```

ile elde edilir. Genel olarak  $R(u,v)$  ile  $u$  ve  $v$  indisleri ile belirtilen satır ve sütunlardan oluşan alt matris elde edilir. Örneğin

```
>> R=rand(4)
```

```
R =
```

```
    0.9501    0.8913    0.8214    0.9218
    0.2311    0.7621    0.4447    0.7382
    0.6068    0.4565    0.6154    0.1763
    0.4860    0.0185    0.7919    0.4057
```

```
>> u=[1,3];
```

```
>> v=[2,3];

>> R(u,v)
ans =
    0.8913    0.8214
    0.4565    0.6154
```

## 5.4 Matris Argümanlı Fonksiyon ve Grafiği

$(2, 1)$  noktasında mutlak minimuma sahip olduğunu belirlediğimiz

$$f(x, y) = x^2 + xy + y^2 - 4x - 5y$$

kuadratik fonksiyonunun  $(1, 2)$  noktasını içeren bir aralıkta, örneğin  $[0, 2] \times [1, 3]$  aralığında, birbirine eşit uzaklıklı noktalardan seçilmiş ağ adı verilen sonlu sayıdaki nokta çiftinde değerini hesaplamak suretiyle bir matris elde edebiliriz. Daha sonra ise `mesh` fonksiyonu yardımıyla fonksiyonun(fonksiyon değerlerinden oluşan matrisin) grafiğini çizebiliriz.

Öncelikle  $x$  ve  $y$  vektörlerini sırasıyla satır ve sütunca çoğaltarak matris üreten `meshgrid` fonksiyonunu inceleyelim:

```
>> x=[1 2 3];y=[4 5 6];
>> [X,Y]=meshgrid(x,y)
```

X =

1	2	3
1	2	3
1	2	3

Y =

4	4	4
5	5	5
6	6	6

```
>> X+Y
```

ans =

## Matrislerle İşlemler

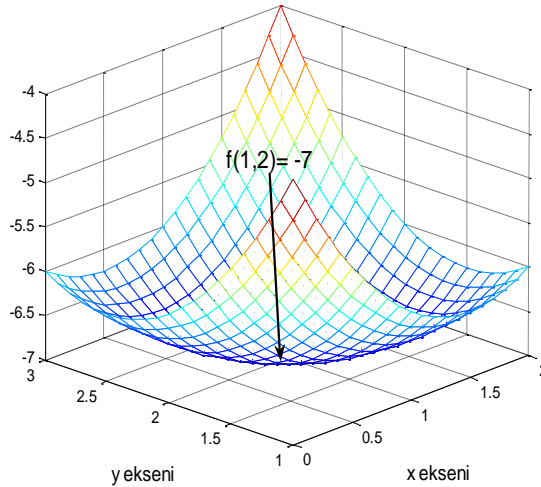
5	6	7
6	7	8
7	8	9

Bu durumda  $X + Y$  matrisinin  $(i, j)$  –inci elemanının  $x(i) + y(j)$  ‘ye karşılık geldiğine dikkat edelim.

Verilen  $f$  fonksiyonunun grafiğini çizmek için öncelikle  $x$  ve  $y$  eksenlerinde uygun grafik çizim noktaları tanımlamalıyız:

```
>> x=0:0.1:2;  
>> y=1:0.1:3;  
  
>> [X,Y]=meshgrid(x,y);  
  
>> Z=X.^2+X.*Y+Y.^2-4*X-5*Y; % Matris argümanlı bir  
                                % fonksiyon  
>> mesh(X,Y,Z)
```

Grafiği çizilmek istenen iki değişkenli fonksiyonda  $x$  yerine  $X$  ve  $y$  yerine  $Y$  yazmak suretiyle ve noktalı operatör kullanımına da dikkat etmek suretiyle  $Z(i, j) = f(X(i, j), Y(i, j))$  matrisi oluşturulur. Mesh fonksiyonu ise her bir  $(X(i, j), Y(i, j))$  noktasına karşılık gelen  $Z(i, j)$  yüksekliğinde bir düğüm içeren ve bu düğümlerin çizgilerle birbirine iliştirildiği bir ağ oluşturur:



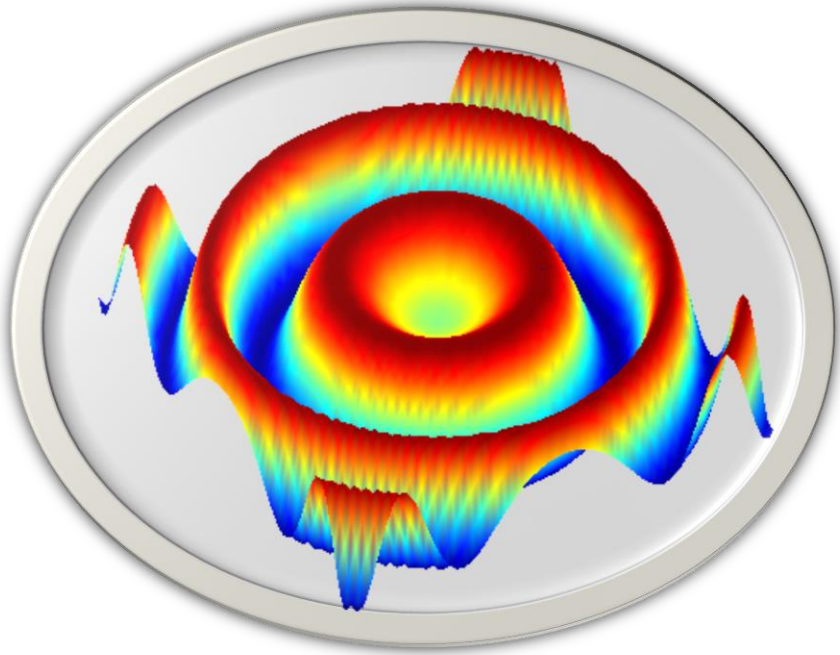
Şekil 5.1  $f$  fonksiyonunun ağ grafiği

Şekil 5.1 den görüleceği üzere  $f$  fonksiyonunu  $(1,2)$  noktasında mutlak minimuma sahiptir. Ağıdaki boşlukların kapatılarak düzgün bir yüzey oluşturulması için ise `surf` grafik komutu `shading interp` komutu ile birlikte kullanılabilir.

### ÖRNEK 5.3

$f(x,y) = \sin(x^2 + y^2)$  fonksiyonunun  $[-3,3] \times [-3,3]$  bölgesi üzerindeki yüzey grafiğini çiziniz.

```
>> x=-3:0.1:3;
>> y=x;
>> [XX,YY]=meshgrid(x,y);
>> Z=sin(XX.^2+YY.^2);
>> surf(XX,YY,Z);
>> shading interp
>> axis('off') % koordinat eksenlerinin çizilmemesini
               % sağlar
```

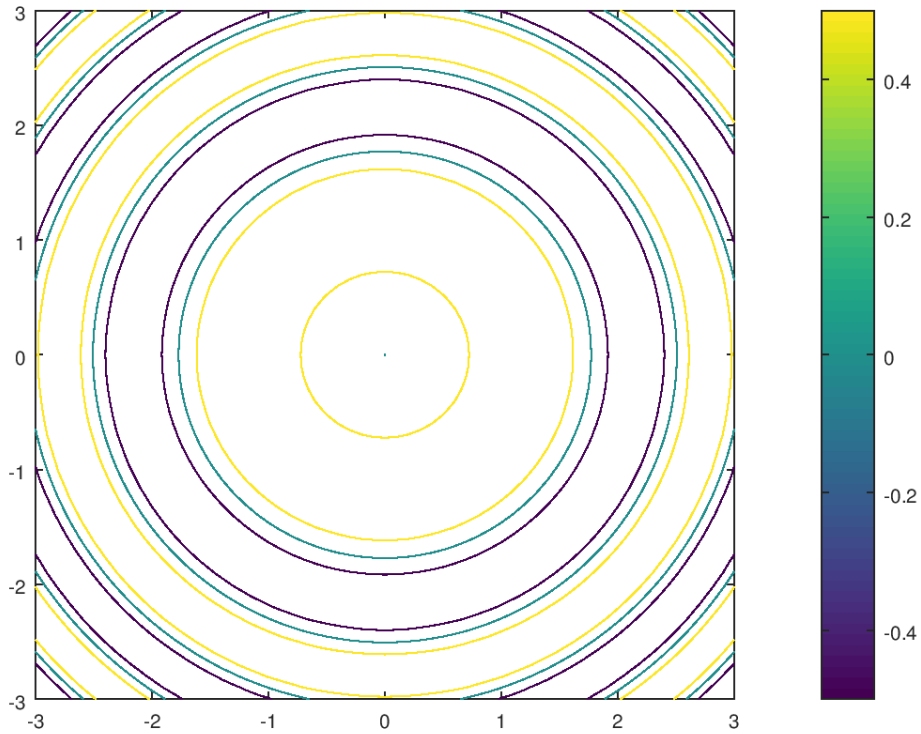


Şekil 5.2  $f(x,y)=\sin(x^2+y^2)$  fonksiyonunun yüzey grafiği

Yüzey grafiği yerine seviye eğrileri çizdirmek isteniyorsa, bu durumda `contour(seviye eğrisi)` grafik çizim komutu kullanılır.

```
>> contour(X,Y,Z,3)
>> colorbar
```

komutları ile yukarıda tanımlanan  $f$  fonksiyonu için Şekil 5.3 ü elde ederiz. Burada `contour` fonksiyonunun son argümanı, skaler olması durumunda çizdirilmesi gereken seviye eğrisi sayısını, vektör olması durumunda ise hangi  $Z$  seviyelerinde seviye eğrisinin çizdirilmesi gerektiğini ifade eder. `colorbar` komutu hangi eğrinin hangi seviyeye, yani  $Z$  değerine sahip olduğunu belirtmek için renklendirme çubuğunu grafik yanına yerleştirir.



Şekil 5.3  $f(x,y)=\sin(x^2+y^2)$  fonksiyonunun seviye grafiği

## 5.5 OCTAVE Matris fonksiyonları

### 5.5.1 Bir Matris ile Üretilen Dört Alt Uzayın Tabanı

Reel elemanlı  $A_{m \times n}$  matrisi verilmiş olsun.  $A$  matrisinin satırlarının lineer kombinasyonu ile oluşturulan noktaları içeren uzaya  $A$  nın satır uzayı adı verilir ve bu uzay  $\mathbb{R}^n$  nin alt uzayıdır, çünkü her bir satır  $\mathbb{R}^n$  nin elemanıdır.

Benzer biçimde  $A$  matrisinin sütunlarının lineer kombinasyonu ile oluşturulan noktaları içeren uzaya  $A$  nın sütun uzayı adı verilir ve bu uzay  $\mathbb{R}^m$  in alt uzayıdır, çünkü her bir sütun  $\mathbb{R}^m$  nin elemanıdır.

Şimdi  $A_{m \times n}$  matrisini  $\mathbb{R}^n \rightarrow \mathbb{R}^m$  bir lineer dönüşüm olarak düşünelim.  $A$  lineer dönüşümü altında  $\mathbb{R}^m$  in sıfır elemanına resmedilen noktaların kümesine  $A$  nın sıfır uzayı adı verilir ve bu uzay  $\mathbb{R}^n$  nin alt uzayıdır.

Benzer biçimde  $A$  matrisinin transpozesi ( $A^T$ ) ile tanımlanan lineer dönüşüm altında  $\mathbb{R}^n$  nin sıfır elemanına resmedilen noktaların kümesine  $A^T$  un sıfır uzayı adı verilir ve bu uzay  $\mathbb{R}^m$  nin alt uzayıdır. Bu uzayların özellikleri aralarındaki diklik bağıntıları Tablo 5.2 de gösterilmektedir(Bkz,[8,9]).

Tablo 5.2:  $A$  matrisi ile üretilen dört alt uzayın özellikleri

$A_{m \times n}$ matrisinin <u>sıfır uzayı</u>	$N(A) = \{X \in \mathbb{R}^n   AX = 0\} \subseteq \mathbb{R}^n;$
<u>sütun uzayı</u>	$R(A) = \{AX \in \mathbb{R}^m   X \in \mathbb{R}^n\} \subseteq \mathbb{R}^m;$
<u>sol sıfır uzayı</u> ( $A^T$ un sıfır uzayı)	$N(A^T) = \{X \in \mathbb{R}^m   A^T X = 0\} \subseteq \mathbb{R}^m$
ve <u>satır uzayı</u> ( $A^T$ un sütun uzayı)	$R(A^T) = \{A^T X \in \mathbb{R}^n   X \in \mathbb{R}^m\} \subseteq \mathbb{R}^n$
olmak üzere,	
$\mathbb{R}^n = N(A) + R(A^T), N(A) \cap R(A^T) = \{0\}, N(A) \perp R(A^T)$	
ve	
$\mathbb{R}^m = N(A^T) + R(A), N(A^T) \cap R(A) = \{0\}, N(A^T) \perp R(A)$	



## Matrislerle İşlemler

dir. Sütun uzayının boyutu ve satır uzayının boyutu birbirine eşittir.  $\text{boyut}(A) = \text{boyut}(A^T) = r$  olmak üzere

$\text{boyut}(N(A)) = n - r$ ;  $\text{boyut}(N(A^T)) = m - r$  dir.

Yukarıda ifade edilen dört alt uzayın tabanı Tablo 5.2 de verildiği üzere OCTAVE ortamında hesaplanabilir.

### ÖRNEK 5.4:

$A = \begin{bmatrix} 2 & 1 \\ 1 & 1/2 \end{bmatrix}$  matrisi ile üretilen dört alt uzayın tabanını OCTAVE ortamında belirleyiniz.

İstenilen dört alt uzayın tabanı Tablo 5.2 de verilmektedir.

Tablo 5.2. A matrisi ile ilişkili dört alt uzay ve tabanlarının OCTAVE ortamında hesaplanması

$Z = \text{null}(A)$	A matrisinin sıfır uzayının tabanını verir. Z nin sütunlarının sayısı sıfır uzayının boyutudur.	<pre>&gt;&gt;A=[2 1;1 1/2] &gt;&gt; Z=null(A)  Z =      -0.4472      0.8944 &gt;&gt; size(Z,2)  ans =       1</pre>
$Z = \text{orth}(A)$	A matrisinin sütun uzayının bir tabanını verir. Z nin sütunlarının sayısı sütun uzayının boyutudur.	<pre>&gt;&gt;A=[2 1;1 1/2] &gt;&gt; R=orth(A)  R =       0.8944      0.4472</pre>
		<pre>&gt;&gt; Z=null(A')</pre>

## Matrislerle İşlemler

$Z = \text{null}(A')$	A matrisinin transpozusunun sıfır uzayının tabanını verir.	$Z =$ $\begin{bmatrix} -0.4472 \\ 0.8944 \end{bmatrix}$
$Z = \text{orth}(A')$	A matrisinin satır uzayının bir tabanını verir.	$\gg Z = \text{orth}(A')$ $Z =$ $\begin{bmatrix} 0.8944 \\ 0.4472 \end{bmatrix}$
$N(A) \perp R(A^T)$	Sıfır uzayı satır uzayına diktir:	$\gg \text{null}(A)' * \text{orth}(A')$ $\text{ans} =$ $5.5511\text{e-}017$
$N(A^T) \perp R(A)$	Sol sıfır uzayı sütun uzayına diktir:	$\text{orth}(A)' * \text{null}(A')$ $\text{ans} = 5.5511\text{e-}017$

### 5.5.2 Diğer Matris fonksiyonları

Vektörler üzerinde de olduğu üzere matrisler üzerinde Lineer cebirsel işlemleri gerçekleştiren OCTAVE fonksiyonları mevcuttur.

Sıkça kullanılan matris fonksiyonlardan bazıları Tablo 5.3 de verilmektedir. Tabloda elde edilen sonuçlar aşağıda tanımlanan  $A$  matrisi ve  $b$  vektörü için elde edilmiştir:

```
>> A=[2 1;1 4]
```

```
A =
```

```

2      1
1      4
```

```
>> b=[3 5]'
```

```
b =
```

## Matrislerle İşlemler

3  
5

Tablo 5.3. Matris fonksiyonları

İşlem	Anlamı	Sonuç
<code>det(A)</code>	A matrisinin determinantını hesaplar	<pre>&gt;&gt; det(A)  ans =  7</pre>
<code>rank(A)</code>	A matrisinin rankını hesaplar. (rank: sütun veya satır uzayının boyutu)	<pre>&gt;&gt; rank(A)  ans =  2</pre>
<code>norm(A)</code>	<p>A matrisinin normunu(en büyük tekil değerini, yani <math>A' * A</math> nın öz değerlerinin karekökleri içerisinde en büyüğünü) hesaplar.</p> <p><code>Norm(A,1)</code>: mutlak değerce maksimum sütun toplamı,</p> <p><code>norm(A,inf)</code>: mutlak değerce maksimum satır toplamıdır.</p> <p><code>norm(A,'fro')</code> A nın elemanlarının karelerinin toplamının</p>	<pre>&gt;&gt; norm(A)  ans =  4.4142  &gt;&gt; norm(A,1)  5  &gt;&gt; norm(A,inf)  5  &gt;&gt; norm(A,'fro')  ans =  4.6904</pre>

## Matrislerle İşlemler

	karekökü	
$\text{inv}(A)$	A matrisinin tersini hesaplar	<pre>&gt;&gt; C=inv(A)  C =  0.5714    0.1429 -0.1429    0.2857</pre>
$[V,D]=\text{eig}(A)$	<p>A matrisinin öz değer ve öz vektörlerini hesaplar. Öz vektörler V matrisinin sütunları ve öz değerler ise D köşegen matrisinin köşegen üzerindeki elemanlarıdır. Sağ sütunda görüldüğü üzere yukarıda verilen A matrisinin öz değerleri <math>\lambda_1 = 1.5858, \lambda_2 = 4.4142</math> ve öz vektörleri ise</p> $V_1 = \begin{bmatrix} -0.9239 \\ 0.3827 \end{bmatrix}, V_2 = \begin{bmatrix} 0.3827 \\ 0.9239 \end{bmatrix}$ <p>olarak verilmektedir.</p>	<pre>&gt;&gt; [V,D]=eig(A),  V =  -0.9239    0.3827 0.3827    0.9239  D =  1.5858    0 0    4.4142</pre>
$[L,U,P]=\text{lu}(A)$	PA=LU eşitliği sağlanacak şekilde köşegen üzerindeki elemanları 1 ler olan L alt	<pre>&gt;&gt; [L,U,P]=lu(A)  L =  1.0000    0</pre>

## Matrislerle İşlemler

	<p>üçgensel matrisi ile U üst üçgensel matrisi ve P permütasyon matrisini hesaplar. Ayrışım satır değişimi gerekmeksizin mümkünse P birim matris olur.</p>	$\begin{bmatrix} 0.5000 & 1.0000 \\ 2.0000 & 1.0000 \\ 0 & 3.5000 \end{bmatrix}$ $U = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
$[Q, R, E] = \text{qr}(A)$	<p>AE=QR eşitliği sağlanacak şekilde <math>R_{m \times n}</math> üst üçgensel matrisi, <math>Q_{m \times m}</math> ortogonal matrisi ve E sütun permütasyon matrisini hesaplar.</p>	<p>&gt;&gt; [Q,R,E]=qr(A)</p> $Q = \begin{bmatrix} -0.2425 & -0.9701 \\ -0.9701 & 0.2425 \end{bmatrix}$ $R = \begin{bmatrix} -4.1231 & -1.455 \\ 0 & -1.6977 \end{bmatrix}$ $E = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
$[U, S, V] = \text{svd}(A)$	<p><math>A = USV^T</math> eşitliği sağlanacak şekilde üniter U ve V matrisleri ile A'nın tekil değerlerini azalan sırada içeren S köşegen matrisini hesaplar.</p> <p>Bu ayrışım matris tersinin hesaplanmasında</p>	<p>&gt;&gt; [U,S,V]=svd(A)</p> $U = \begin{bmatrix} 0.3827 & 0.9239 \\ 0.9239 & -0.3827 \end{bmatrix}$ $S = \begin{bmatrix} 4.4142 & 0 \\ 0 & 1.5858 \end{bmatrix}$ $V = \begin{bmatrix} 0.3827 & 0.9239 \\ 0.9239 & -0.3827 \end{bmatrix}$

## Matrislerle İşlemler

	kullanışlıdır.	
$R=rref(A)$	A matrisinin indirgenmiş eşelon formunu verir	<pre>&gt;&gt; rref(A) ans =     1    0     0    1</pre>
$x=A \backslash b$	b bir sütun vektörü olmak üzere $Ax=b$ denklem sisteminin mümkünse gerçek veya değilse <b>En Küçük Kareler</b> anlamında $A^T Ax = A^T b$ denklem sisteminin çözümünü yaklaşık olarak verir	<pre>&gt;&gt; x=A\b x =     1.0000     1.0000</pre>
<pre>&gt;&gt; format rat &gt;&gt; A=hilb(3)</pre>	3x3 lük Hilbert matrisi üretir	<pre>A =     1    1/2    1/3     1/2    1/3    1/4     1/3    1/4    1/5</pre>
$Cond(A)$ $=\ A\  \ A^{-1}\ $	<p>A matrisinin tekil bir matris olmaya ne kadar yakın veya tekil olmaktan ne kadar uzak olduğunun bir ölçüsüdür.</p> <p>Büyük sayısal sonuç tekilliğe yakınlığı ifade eder.</p>	<pre>&gt;&gt; cond(A) ans =     524.0568</pre> <p>Hilbert matrisi neredeyse tekil bir matristir.</p> <pre>&gt;&gt; det(A) ans=4.6296e-004</pre>

## 5.6 Linear Denklem Sistemi Çözümü

Verilen  $Ax = b$  denklem sisteminin çözümü OCTAVE ortamında ‘\’ operatörü ile gerçekleştirilir.

$x=A \backslash b$	<p>b bir sütun vektörü olmak üzere <math>Ax=b</math> denklem sisteminin mümkünse gerçek veya değilse <b>En Küçük Kareler</b> anlamında <math>A^T Ax = A^T b</math> denklem sisteminin çözümünü yaklaşık olarak verir</p>	<pre>&gt;&gt; x=A\b x =     1.0000     1.0000</pre>
--------------------	--	---

## 5.Bölüm Alıştırmaları

$A = \begin{bmatrix} 2 & 1 & 3 \\ -1 & -2 & 4 \\ 4 & 4 & 6 \end{bmatrix}$  matrisi verilmiş olsun. Aşağıda verilen 1-12 nolu soruları bu matrisi kullanarak cevaplandırınız.

1.  $A$  matrisinin aşağıda belirtilen fonksiyonlarını hesaplayarak sonuçları lineer cebir bilgilerinizle yorumlayınız.
  - a.  $\det(A)$
  - b.  $\det(A')$
  - c.  $\det(4A)$
2. `fliplr` komutunu  $A$  matrisi ve değişik boyuttaki başka matrisler üzerinde deneyerek işlevini anlamaya çalışınız ve aşağıdaki işlem sonuçlarını yorumlayınız.
  - a.  $B = \text{fliplr}(A)$
  - b.  $\det(B)$
  3.  $B = \text{inv}(A)$  olmak üzere
    - a.  $\det(AB) = \det(BA) = 1$  olduğunu gözlemleyiniz.
    - b.  $B = A; B(2,:) = A(1,:);$  işlemi sonucunda oluşan  $B$  matrisini elde ediniz.
    - c.  $\det(B)$  değerini nasıl yorumlarsınız?
  4.  $A^T A$  matrisinin özdeğerlerinin kareköklerine  $A$  matrisinin tekil değerleri adı verilir ve  $\text{svd}(A)$

komutu ile belirlenir. Buna göre yukarıda verilen A matrisinin tekil değerlerini belirleyiniz.

5. A matrisinin aşağıda belirtilen normlarını hesaplayınız. Her bir normu anlamaya çalışınız.
  - a.  $norm(A, 1)$
  - b.  $norm(A, 2)$  yi hesaplayınız ve  $svd(A)$  ile karşılaştırınız
  - c.  $norm(A, 'fro')$  (Frobenius normu) yı hesaplayınız ve  $A^T A$  nın köşegen üzerindeki elemanlarının toplamının karekökü ile karşılaştırınız.
  - d.  $sum(sum(A.^2))$  nın karekökü ile c deki sonucunuzu karşılaştırınız.
  - e.  $norm(A, inf)$
6. Bir matrisinin herhangi bir veya birden fazla satır veya sütunu alınarak başka bir matris veya vektör oluşturulabilir. Aşağıdaki işlemleri OCTAVE komut penceresinde gerçekleştirerek A matrisinden vektör veya alt matrislerin nasıl elde edildiğini inceleyiniz:
  - a.  $B = A(:, 1)$
  - b.  $B = A(1, :)$
  - c.  $B = A(1:2, :)$
  - d.  $B = A([1\ 3], [2\ 3])$
7.
  - a.  $[L, U, P] = lu(A)$  ile  $L, U$  ve  $P$  matrislerini kullanarak  $PA = LU$  eşitliğinin sağlandığını kontrol ediniz
  - b.  $[Q, R, E] = qr(A)$  ile  $Q, R$  ve  $E$  matrislerini kullanarak  $AE = QR$

eşitliğinin sağlandığını kontrol ediniz.

8. A matrisine sırasıyla bir ve iki kez uygulanan diag fonksiyonunun işlevini aşağıdaki komutlar yardımıyla inceleyiniz.
  - a.  $B = diag(A)$
  - b.  $B = diag(diag(A))$
9. A matrisine sırasıyla bir ve iki kez uygulanan min ve max fonksiyonlarının işlevlerini aşağıdaki komutlar yardımıyla inceleyiniz.
  - a.  $B = min(A)$
  - b.  $B = min(min(A))$
  - c.  $B = max(A)$
  - d.  $B = max(max(A))$
10. A matrisinin öz değer ve öz vektörlerini  $[V, D] = eig(A)$  komutu ile bularak aşağıdaki eşitliklerin sağlandığını kontrol ediniz. Elde ettiğiniz sonuçlara göre hangi öz vektör hangi öz değere aittir?
  - a.  $A * V(:, 1) = D(1, 1) * V(:, 1)$
  - b.  $A * V(:, 2) = D(2, 2) * V(:, 2)$
  - c.  $A * V(:, 3) = D(3, 3) * V(:, 3)$
11. tril ve triu fonksiyonlarının işlevlerini aşağıdaki komutlar yardımıyla inceleyerek, c şıkında belirtilen eşitliğin sağlandığını kontrol ediniz.
  - a.  $B = tril(A)$
  - b.  $B = triu(A)$
  - c.  $A = tril(A) + triu(A) - diag(diag(A))$
12. Matris normları için aşağıdaki özdeşlikleri kontrol ediniz



- a.  $norm(A, 1) = \max(\text{sum}(\text{abs}(A)))$   
(Mutlak değerce en büyük sütun toplamı)
- b.  $norm(A, \text{inf}) = \max(\text{sum}(\text{abs}(A')))$   
(Mutlak değerce en büyük satır toplamı)
- c.  $norm(A) = norm(A, 2) = \max(\text{svd}(A))$ , (En büyük tekil değer)
- d.  $norm(A, 'fro') = \sqrt{\text{sum}(\text{diag}(A' * A))}$   
(Frobenius normu)
13.  $A = \begin{bmatrix} 2 & 1 \\ 4 & 2 \end{bmatrix}$  matrisi verilmiş olsun.
- a. satır, sütun sıfır ve sol sıfır uzaylarının tabanlarını belirleyiniz.
- b. Satır uzayının tabanının sıfır uzayının tabanına dik olduğunu gözlemleyiniz.
- c. Sütun uzayının tabanının sol sıfır uzayının tabanına dik olduğunu gözlemleyiniz.
14.  $A = \begin{bmatrix} 2 & 1 & 3 \\ 1 & -1 & 0 \\ 1 & -3 & -2 \end{bmatrix}$  matrisi verilmiş olsun.
- a. Matrisin rankını belirleyiniz.
- b. Satır, sütun sıfır ve sol sıfır uzaylarının tabanlarını belirleyiniz.
- c. Satır uzayının tabanının sıfır uzayının tabanına dik olduğunu gözlemleyiniz.
- d. Sütun uzayının tabanının sol sıfır uzayının tabanına dik olduğunu gözlemleyiniz.
15. “\” ters bölme operatörü  $Ax = b$  denklem sisteminin varsa çözümünün belirlenmesi, yoksa eşitliği sağlayan en yakın(iyi) çözümün belirlenmesini sağlar. Buna göre aşağıdaki matris ve sağ yan vektörleri için ilgili denklem sistemlerin çözümünü belirleyerek, elde ettiğiniz sonuçları lineer cebir bilgilerinizle karşılaştırınız:
- a.  $A = [1, 2; 2, -1]; b = [3, 1]'$
- b.  $A = [1, 2; 2, 4]; b = [3, 6]'$
- c.  $A = [1, 2; 2, 4]; b = [3, 7]'$
- d.  $A = [1, 2; 2, -1; 1, 3]; b = [1, -1, 2]'$
16.  $A = [1, 2; 2, -1]$  matrisi ve  $e_1 = [1, 0]'$  ve  $e_2 = [0, 1]'$  olmak üzere
- a.  $Ax = e_1$  denklem sistemin  $x$  çözümünü belirleyiniz.
- b.  $Ay = e_2$  denklem sisteminin  $y$  çözümünü belirleyiniz.
- c. Elde ettiğiniz  $x$  ve  $y$  çözümlerini sütun kabul eden  $B = [x \ y]$  matrisini oluşturunuz.
- d.  $C = A^{-1}$  matrisini belirleyiniz.
- e.  $B = C$  olduğunu gözlemleyiniz.
- f. Elde ettiğiniz sonucu yorumlayınız.
- 17.
- $$\begin{aligned} 3x - y + z &= 3 \\ x + 2y - z &= 2 \\ x + y - 3z &= -1 \end{aligned}$$
- denklem sistemini çözümünü Soru 13 de belirtilen “\” operatörü yardımıyla belirleyiniz.

18.

$$\begin{aligned}x + 2y &= 3 \\ 2x - y &= 1 \\ x + y &= 3\end{aligned}$$

denklem sistemi verilsin.

- a. Verilen sistemin çözümü olmadığını gösteriniz.
- b. “\” operatörü yardımıyla çözüm elde etmeye çalışınız. OCTAVE in bir çözüm elde ettiğini gözlemleyiniz.
- c.  $A$  matrisi verilen sistemin katsayı matrisi ve  $b$  vektörü ise sağ yan vektörü olmak üzere  $A^T Ax = A^T b$  denklem sisteminin çözümünü belirleyiniz. Elde ettiğiniz sonuç  $b$  şıkkındaki sonucunuza eşit midir?

## VI. BÖLÜM

### 6.OCTAVE ile Kodlama(Programlama)

Bu bölümde programlama dili olarak OCTAVE ı tanıtıyor ve

- ☑ tipik bir programda mevcut olan sıralı, şartlı ve tekrarlı yapıların OCTAVE programlama dillerindeki karşılıklarını örneklerle inceliyoruz.Ayrıca
- ☑ program hazırlama işlemini yöntem ve adımları, algoritma ve kod hazırlama olarak üç aşamada inceliyoruz.
- ☑ hazırladığımız programları komut dosyası(script) uygulaması olarak geliştiriyoruz.

Verilen bir problemin elektronik ortamda çözülebilmesi için

- ☑ Yöntem ve adımları adı verilen başlıkta kullanılacak olan sayısal yöntem ve temel adımlarınınım ifade edilmesi,
- ☑ Yöntem ve adımları başlığında belirtilen işlemler için gerekli değişken ve formülasyonların belirtildiği ve Algoritma adı verilen komutlar kümesinin oluşturulması ve
- ☑ son olarak ta algoritmanın bilgisayarın anlayabileceği dil ile ve uygun kurallar ile ifade edilmesi gerekmektedir.

Verilen problem için Yöntem ve adımları, Algoritma ve Kod hazırlama aşamalarının tümüne ise Kodlama veya Programlama adı veriyoruz.

Bir algoritmada bir biri ardından gerçekleştirilmesi gereken adımları içeren yapıya sıralı yapı adı verilir.

Bir algoritmada oluşacak olan işlem sonuçlarına göre farklı işlem veya işlem gruplarının gerçekleştirilmesi söz konusu ise, bu yapıya **şartlı yapı** adı verilir.

Öte yandan bir algoritmada belirli sayıda veya belirli kriter sağlanıncaya kadar(veya sağlandığı sürece) tekrar etmesi gereken yapı söz konusu ise bu tür yapıya **tekrarlı yapı** adı verilir.

Şimdi yöntem ve adımları, algoritma ve kodlama kavramlarını, yukarıda bahsedilen sıralı, şartlı ve tekrarlı yapı örnekleriyle inceleyelim.

### 6.1 Sıralı Yapı

#### ÖRNEK 6.1

Girilen pozitif bir tamsayıya kadar olan tüm pozitif sayıların toplamını hesaplayan bir program hazırlayalım.

Öncelikle probleme ait algoritmayı hazırlamalıyız. Ancak algoritmayı da hazırlamadan önce “yöntem ve adımları” adı verilen ön hazırlık çalışmasıyla problemin çözümü için gerekli yöntemi ve yöntemin temel adımlarını ifade etmeliyiz.

#### **Yöntem ve adımları**

1. Kullanıcının pozitif bir tamsayı girmesini isteyerek, girilen sayıyı uygun değişkene, örneğin  $n$ , atayalım.
2. 1 den  $n$  e kadar olan sayıların toplamını veren  $n(n + 1)/2$  formülünü kullanarak toplam değerini belirleyelim ve uygun değişkene atayalım.
3. Sonucu kullanıcıya iletelim.

Yukarıda belirlenen adımları şimdi uygun sırada komutlar kümesine dönüştürebiliriz.

#### **Algoritma:** toplam

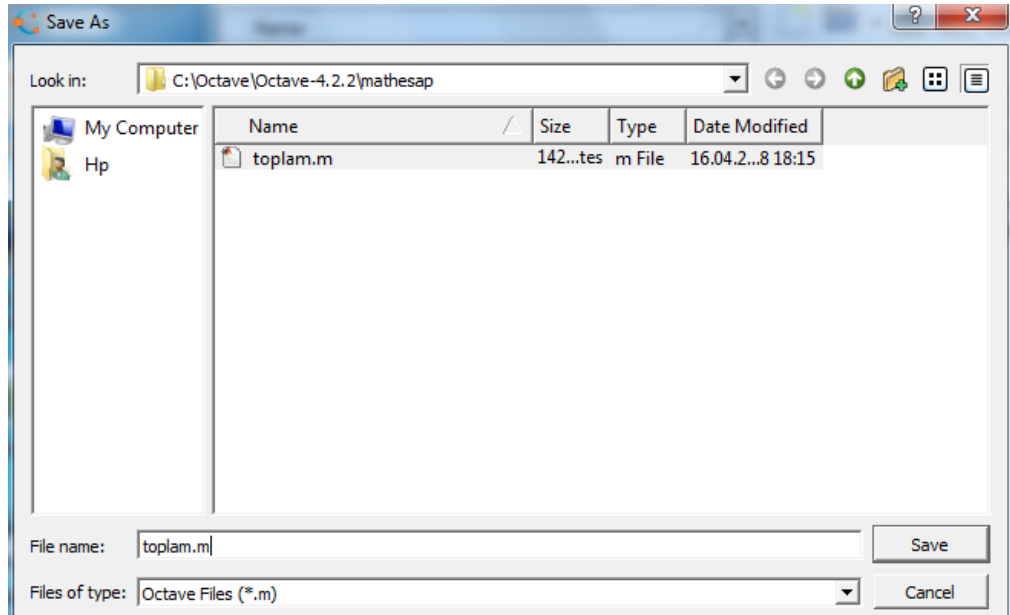
1. Girdi:  $n$  değişkenine atanmak üzere pozitif tamsayı giriniz.
2. İstenilen  $n(n + 1)/2$  toplamını toplam değişkenine atayınız.
3. Sonucu içeren toplam değişken değerini kullanıcıya iletiniz.

#### **Kod:** toplam

```
% Kullanıcı tarafından girilen pozitif tamsayıya  
% kadar olan tüm pozitif sayıların toplamını hesaplar  
  
n=input('n=' );  
toplam=n*(n+1)/2;  
disp(toplam);  
  
% n pozitif tamsayısı  
% alınmakta  
% toplam hesaplatılmakta  
% Sonuç ekrana  
% yansıtılmakta
```

Yukarıdaki kodu OCTAVE File(Dosya) sekmesinden New script(yeni komut dosyası) seçerek oluşan ve editör adı verilen kod hazırlama ortamında yazınız ve yazma işleminin sonunda save(kaydet) seçeneği ile toplam.m ismiyle oluşturacağınız çalışma klasörüne aşağıdaki şekilde görüldüğü gibi kaydediniz.

Aşağıda C:\OCTAVE\OCTAVE-4.2.2 klasöründe oluşturulan komut dosyaları için mathesap isimli bir alt klasörün önceden oluşturulmuş olduğunu ve hazırladığımız programın toplam.m isimli dosyaya kaydedildiğini görüyoruz.



Şekil 6.1 mathesap klasörü içerisine toplam.m dosya kaydı

**Test:** toplam

```
>> toplam
n=4
10
>> toplam
n=5
15
```

## **ÖRNEK 6.2**

İkinci dereceden polinomun köklerini belirleyen bir program hazırlayalım.

Öncelikle probleme ait algoritmayı hazırlamalıyız. Ancak algoritmayı da hazırlamadan önce “yöntem ve adımları” adı verilen ön hazırlık çalışmasıyla problemin çözümü için gerekli yöntemi ve yöntemin temel adımlarını ifade etmeliyiz.

### **Yöntem ve adımları**

1. Kullanıcının  $p(x) = ax^2 + bx + c$  biçiminde olduğu kabul edilen polinom katsayıları girmesini isteyerek, girilen sayıları uygun değişkenlere atayalım.
2. Diskriminantı hesaplayarak uygun değişkene atayalım.
3. Kökleri
  - i.  $\frac{-b \pm \sqrt{\text{karekök(diskriminant)}}}{2a}$
  - ii. formüller yardımıyla hesaplayalım.
4. Sonuçları kullanıcıya iletelim.

Yukarıda belirlenen adımları şimdi uygun sırada komutlar kümesine dönüştürebiliriz.

### **Algoritma:** kok\_bul

1. Girdi:  $ax^2 + bx + c$  polinomunun katsayılarını al ve a, b ve c isimli değişkenlere ata
2. Diskriminantı hesapla ve Delta değişkenine ata  $\Delta = b^2 - 4ac$
3. Kökleri hesapla ve  $x_1, x_2$  değişkenlerine ata:
4.  $x_1 = \frac{-b - \sqrt{\Delta}}{2a}, x_2 = \frac{-b + \sqrt{\Delta}}{2a}$
5.  $x_1, x_2$  değerlerini yazdır.

Yukarıda incelen kok\_bul isimli algoritma sıralı yapı adı verilen ve her bir adımın bir önceki adımı sırasıyla takip ettiği bir algoritma türüdür.

Bir sonraki aşama Algoritmanın Kod adı verilen ve bilgisayarın anlayarak yorumlayabileceği veya işleyebileceği formata dönüştürülmesidir. Kok\_bul isimli kod aşağıda verilmektedir:

**NOT:** Yöntem ve adımları aşaması ile algoritma aşaması birbirine çok yakın işlemler içeriyor olsa da bu iki aşamanın ayrı ayrı incelenmesi programlama işlemini kolaylaştırır ve oldukça önemlidir.

**Kod:** kok\_bul

```
% Kullanıcı tarafından katsayıları girilen
% ikinci dereceden polinomun köklerini hesaplar

a=input('a=');
b=input('b=');
c=input('c=');
delta=b^2-4*a*c;
x1=(-b-sqrt(delta))/(2*a);
x2=(-b+sqrt(delta))/(2*a);
disp(['x1=',num2str(x1)]);
disp(['x2=',num2str(x2)]);

% a,b,c katsayıları alınmakta

% Kökler hesaplatılmakta

% Sonuçlar ekrana yansıtılmakta
```

Bir sonraki aşama ise hazırlanan kodun kok\_bul.m isimli dosyada kaydedildikten sonra doğru çalışıp çalışmadığının kontrol edilmesi veya test yapılması aşamasıdır.

**Test:** kok\_bul

<pre>&gt;&gt; kok_bul a=1 b=-3 c=2 x1=1 x2=2</pre>	<pre>&gt;&gt; kok_bul a=1 b=1 c=1 x1=-0.5-0.86603i x2=-0.5+0.86603i</pre>
--	---

Not: Yukarıdaki program ile elde edilecek sonuçları OCTAVE roots([a b c]) fonksiyonu kullanarak kontrol edelim.

```
>> roots([1 -3 2])
ans =
     2
     1
```

## 6.2 Şartlı Yapı

Bir diğer yapı ise oluşacak olan işlem sonuçlarına göre farklı işlem veya işlem gruplarının gerçekleştirilmesini gerektirir ki bu tür yapılara *şartlı yapı* adı verilmektedir.

OCTAVE da şartlı yapı *if* komutu yardımıyla gerçekleştirilir. *if* komutunun genel yazılımı

```
if mantıksal ifade
    deyimler grubu
end
```

veya

```
if mantıksal ifade_1
    deyimler grubu
else mantıksal ifade_2
    deyimler grubu
end
```

veya en genel halde

```
if mantıksal ifade_1
    deyimler grubu
elseif mantıksal ifade_2
    deyimler grubu
...
elseif mantıksal ifade_n
    deyimler grubu
else
    deyimler
end
```

biçimindedir.



Yukarıda belirtilen mantıksal ifadelerde karşılaştırma operatörleri kullanılır. Bu operatörler Bölüm 2.2 den incelendiği üzere

**Aritmetik** karşılaştırma operatörleri

>(büyük),  
>=(büyük veya eşit),  
<(küçük),  
<=(küçük veya eşit),  
Eşit mi(==) ve  
eşit değil mi(~=) dir.

**Mantıksal** karşılaştırma operatörleri ise ve (&), veya(|) dir.

**NOT:** Atama operatörü '=' ile iki eşitliğin yan yana ve aralarında boşluk bırakılmadan yazılmasıyla oluşturulan eşitlik karşılaştırma operatörü '==' nü karıştırmamaya dikkat ediniz.

Bölüm 2.2 de skalerler üzerinde gerçekleştirilen karşılaştırma işlemleri vektörler ve matrisler için de gerçekleştirilebilir. Örneğin

```
>> x=[1 2 3]
```

```
x =
```

```
1      2      3
```

```
>> x>0
```

```
ans =
```

```
1      1      1
```

Öte yandan

```
>> x=[1 -2 3]
```

```
x =
```

```
1      -2      3
```

```
>> x>0
```

```
ans =
```

```
1      0      1
```

elde ederiz.

Karşılaştırma kriterini sağlayan bileşenlerin 1 sayısal değerine ve diğerlerinin ise sıfır değerine sahip olduğunu gözlemleyiniz.

```
>> A= [1 2;3 0]
```

```
A =
```

```
1  2
3  0
```

```
>> A>1
```

```
ans =
```

```
0  1
1  0
```

A matrisinin 1 den büyük elemanlarına karşılık gelen karşılaştırma sonuçlarının doğru(yani sayısal olarak 1 değeri) ve 1 den küçük olan karşılaştırma sonuçlarının ise

yanlış(veya sıfır değeri) olarak kabul edildiğine dikkat edelim.

Yukarıda tanımlanan `if` komutunun kullanılarak oluşturulan şartlı yapıya ilişkin olarak aşağıdaki örneği inceleyelim.

### **ÖRNEK 6.3**

Kullanıcı tarafından girilen iki sayının büyüğünü belirleyen bir program hazırlayalım. Program öncelikle uygun çözümleme işlemini ve ardından da algoritmayı, yani komutlar kümesini ve son olarak ta ilgili kod aşamalarını içermelidir.

#### **Yöntem ve adımları**

1. Kullanıcının iki sayı girmesini isteyerek, girilen sayıları okutmak suretiyle uygun değişkenlere atayalım.

2. Daha sonra değişken değerlerini karşılaştırarak büyük olanı belirleyelim ve
3. son olarak ta büyük olanı uygun bir değişkene atayarak kullanıcıya gönderelim.

Yukarıda belirlenen adımları şimdi uygun sırada komutlar kümesine dönüştürebiliriz. Algoritma aşaması adımlamanın bir adım daha ileriye götürülerek, gerekli değişken ve formülasyonların açıkça belirtildiği bir aşama olarak düşünülebilir.

**Algoritma:** `buyuk_ab`

1. Girdi: a ve b değişkenlerine atanacak iki sayı giriniz,
2. a ve b yi karşılaştırarak büyük olanı belirleyiniz ve `buyukab` isimli değişkene atayınız, ve
3. Çıktı: sayılardan büyüğü yani `buyukab` değişkeninin değerini kullanıcıya iletiniz.

`buyuk_ab` algoritmasının şartlı yapı içerdiğine dikkat edelim.

Şüphesiz algoritma biz kullanıcıların anlayabildiği dilde ifade edilmiş komutlar kümesidir. Söz konusu komutlar kümesinin Programlama Dili adı verilen ve bilgisayarların anlayabileceği dile dönüştürülmesi gerekmektedir. Bu amaçla geliştirilen bir kısmı daha güncel olan çok sayıda Programlama Dili mevcuttur. Basic, Pascal, Fortran, C ve C++ gibi varyasyonları söz konusu programlama dillerinden sadece birkaç tanesidir.

MATLAB veya OCTAVE bir yazılım kütüphanesi olmasının yanında kullanıcılara kendi programlarını geliştirmelerine izin veren ve C ye benzeyen Programlama Diline sahiptirler.

Şimdi geliştirilen algoritmanın OCTAVE ortamında nasıl bir Kod'a dönüştürüldüğünü inceleyelim:

**Kod:** `buyuk_ab`

```
%Girilen iki sayının büyüğünü belirler.
```

```
a=input('a=');  
b=input('b=');
```

```
a ve b değerleri  
alınıyor  
büyük olan değişken
```

```
if a>b maxab=a;
else maxab=b;
end

disp(['Enbuyuk=', num2str(maxab)
]);

%printf('En büyük=
%d\n',maxab);
```

değeri maxab  
değişkenine atanıyor

Sonuç ekranda disp  
komutu ile  
görüntülenmektedir.

Alternatif olarak  
printf komutu ile de  
aynı sonuç  
görüntülenebilir.

%d formatı  
yazdırılacak olan  
değişken değerinin  
tamsayı olduğunu  
ifade eder.

'\n' karakteri yazma  
işleminde sonra  
imlecin alt satıra  
gitmesini sağlar.

Yukarıdaki programı uygun bir isimle, örneğin `buyuk_ab.m`, kaydettikten sonra, komut penceresinden dosya ismini(uzantısız olarak!) yazmak suretiyle aşağıda görüldüğü gibi çalıştırabiliriz.

Kod aşamasını oluşturulan kodun doğru çalışıp çalışmadığını kontrol amacıyla gerçekleştirilmesi gereken test aşaması takip etmelidir:

**Test:** `buyuk_ab`

```
>> buyuk_ab
a=2
b=3
En buyuk=3
```

### Uyarı:

1. Uzantısı `m` olan veya `m` dosya olarak bilinen dosya isimlerinde ve programlarda kullanılan değişken isimlerinde Türkçe karakter kullanmamalıyız. Dosya veya değişken ismi bir harf ile başlamalıdır. Dosya

veya değişken isminde boşluk veya özel karakterler kullanılmamalıdır, rakam veya alt çizgi kullanılabilir.

2. C programlama dilinde olduğu gibi OCTAVE da büyük ve küçük harf duyarlılığı söz konusudur: X ve x değişkenleri farklı değişkenler olarak algılanırlar.
3. Program içerisinde % ile başlayan satırlar yorum satırları olup, bu satırlarda veya disp gibi ekran yazdırma komutları içerisinde tırnak işareti içerisinde yer alan açıklayıcı bilgilerde Türkçe karakterler kullanılabilir.
4. Bir değişken değerini ekranda görüntülemenin en basit yolu, değişken sonundaki noktalı virgülü kaldırmaktır. Ancak açıklayıcı bir bilginin değişken değerine eşlik etmesi isteniyorsa bu durumda disp, sprintf veya printf komutları kullanılabilir

- Örneğin

```
>> disp('Bu bir denemdir')
Bu bir denemdir
```

- %s formatı aşağıda görüldüğü üzere karakter dizisi(s: string) içeren çıktı üretimi için kullanılır.

```
>> dosya_adi="deneme.m";
>> sprintf('dosya adi %s dir',dosya_adi)
ans = dosya adi deneme.m dir
```

- Tamsayı değerlerini yazdırmak için %d, kesirli sayı değerlerini yazdırmak için %f formatı kullanılır:

```
>> x=pi;
>> printf('x degiskeninin degeri %6.4f dir\n',x)
x degiskeninin degeri 3.1416 dir
```

Format taki 6 sayısı değişken değerinin 6 hanelik bir alana sağa yaslı olarak ve 4 ise kesirli kısmın gerekirse yuvarlatılarak 4 haneli bir alana yazılacağını ifade eder.

```
>> printf('x degiskeninin degeri %8.5f dir\n',x)
x degiskeninin degeri 3.14159 dir
```

örneğinde virgülden sonra 5 haneye yer verildiğine dikkat edelim.

```
>> printf('x degiskeninin degeri %10.5f\n',x)
x degiskeninin degeri      3.14159 dir
```

örneğin de de değeri sözcüğü ile yazdırılan kesirli sayı arasındaki boşluğun 10 hane belirtimi dolayısıyla arttığına dikkat edelim.

### **ÖRNEK 6.4**

Kullanıcı tarafından girilen üç sayının büyüğünü belirleyen bir program hazırlayalım.

Öncelikle çözüm için gerekli çözümlemeyi gerçekleştirelim: Diğer bir deyimle, problemi çözmek için neler yapmamız gerektiğini kabaca ifade edelim:

#### **Yöntem ve adımları**

1. kullanıcının üç sayı girmesini isteyerek, girilen sayıları okutmak suretiyle uygun değişkenlere atayalım.
2. Daha sonra ilk iki değişken değerlerini karşılaştırarak iki sayıdan büyük olanını belirleyelim ve
3. son olarak bir önceki adımda bulduğumuz büyük olan sayı ile üçüncü değişken değerini karşılaştırarak en büyük olanı belirledikten sonra kullanıcıya gönderelim

Yukarıda belirlenen adımları, hangi değişkenlerle ve nasıl gerçekleştireceğimizi ifade eden uygun sırada komutlar kümesine, yani Algoritmaya dönüştürebiliriz.

#### **Algoritma:** `buyuk_abc`

1. Girdi: a, b ve c değişkenlerine atanacak üç sayı
2. a ve b yi karşılaştırarak büyük olanını belirleyiniz ve maxab isimli değişkene atayınız
3. c ile maxab yi karşılaştır ve büyük olanı maxabc isimli değişkene ata.
4. Çıktı: sayılardan büyüğü yani maxabc değişkeninin değeri

#### **Kod:** `buyuk_abc`

```
%Girilen üç sayının büyüğünü belirler.
```

```
a=input('a=');
b=input('b=');
c=input('c=');
if a>b maxab=a;
else maxab=b;
end
if c>maxab maxabc=c;
else maxabc=maxab;
end

disp(['Enbuyuk=',num2str(maxab
c)]);

fprintf('En          buyuk=
%d\n',maxabc);
```

a,b ve c değerleri alınıyor  
a ve b den büyük olan değişken değeri maxab değişkenine atanıyor

Sonuç c ile karşılaştırılarak en büyük olan belirleniyor ve ekranda disp komutu ile görüntülenmektedir.

Alternatif olarak sonuç printf komutu ile de aynı sonuç görüntülenebilir.

**Test:** buyuk\_abc

```
>> buyuk_abc
a=2
b=1
c=3
En buyuk=3
```

### ÖRNEK 6.5

Girilen nota karşı aşağıda belirtilen kurala göre harf notu atayan notbul.m isimli bir dosyaya kaydedilen program hazırlayalım.

```
80:100 →AA
75:79 →BA
70:74 →BB
60:69 →CB
50:59 →CC
45:49 →DC
40:44 →DD
25:39 →FD
```

0:24 →FF

**Yöntem ve adımları:** notbul

1. Girilen notu okuyalım.
2. Notun dahil olduğu aralığı belirleyelim.
3. Yukarıdaki tabloya göre uygun harf notunu atayalım.

**Algoritma:** notbul

1. girilen notu okuyarak not değişkenine atayınız
2. eğer girilen not 80 den büyük veya 80 ne eşitse H\_not ile gösterilen harf not değerinin AA
3. değilse ve fakat not 75 den büyük veya 75 e eşitse BA
4. ...
5. yukarıdakilerin hiçbiri değilse FF notunu kullanıcıya iletiniz.

**Kod:** notbul

```

not=input('Not=');
if not>=80
    H_not=char('AA');
elseif not>=75
    H_not=char('BA');
elseif not>=70
    H_not=char('BB');
elseif not>=60
    H_not=char('CB');
elseif not>=50
    H_not=char('CC');
elseif not>=45
    H_not=char('DC');
elseif not>=40
    H_not=char('DD');

```

Input komutu ile not değeri okunarak not değişkenine atanmaktadır.

Harf notu belirtilen kurala göre hesaplanarak H\_not değişkenine atanmaktadır.

Char fonksiyonu ile ilgili nota karşılık gelen karakter dizisi belirlenmektedir.



```
elseif not>=25
    H_not=char('FD');
else
    H_not=char('FF');
end
disp(['Notunuz=',H_not]);
```

Disp komutu ile sonuç  
ekrana  
yazdırılmaktadır.

**Test:** notbul  
>> notbul  
Not=34  
Notunuz=FD  
>> notbul  
Not=78  
Notunuz=BA

## 6.3 Tekrarlı Yapı

Bazı problemler belirli sayıda işlemin tekrar edilmesini gerektirirler.

OCTAVE da tekrarlı yapı `for` ve `while` döngüleri yardımıyla gerçekleştirilir. For döngüsünün genel yazılımı

```
for değişken=başlangıç_değer:son_değer
    deyimler
end
```

biçiminde olup, deyimler grubunun ne kadar tekrar edileceğinin bilinmesi durumunda tercih edilir.

Konuya ışık tutması açısından aşağıdaki örneği göz önüne alalım:

### ÖRNEK 6.6

Kullanıcı tarafından tanımlanan bir dizinin elemanlarının toplamını hesaplayan bir program hazırlayalım ve `toplam.m` isimli bir dosyaya kaydedelim.

Bu örneği skaler cebirsel ve vektör cebirsel işlemlerle iki farklı biçimde hesaplayabiliriz. Ayrıca OCTAVE kütüphanesini de kullanabiliriz. Öncelikle skaler cebirsel versiyonu inceleyelim.

### a) Skaler cebirsel versiyon

#### Yöntem ve adımları (skaler versiyon)

1. Girilen diziyi bir değişkene atayalım.
2. Dizinin eleman sayısını hesaplayalım.
3. Dizinin elemanlarının toplamını saklamak üzere kullanılacak olan değişkenin, örneğin `top` isimli bir değişkenin, ilk değerini sıfırlayalım.
4. `top` değişkeninin değerine dizinin her bir elemanı ilave edelim ve sonucu yine `top` olarak adlandıralım. Bu işleme yığılmalı toplam adı verilir.
5. `top` değişkeninde saklanan son değeri kullanıcıya geri gönderelim.

Yukarıdaki adımlardan, oluşturacak olduğumuz algoritmanın tekrarlı yapı içereceğine dikkat edelim.

#### Algoritma: `toplam(skaler cebirsel versiyon)`

1. Girdi: `u` isimli ve sonlu elemanlı bir dizi
2. Dizinin eleman sayısını hesaplayarak `n` isimli değişkene ata
3. `top` değişkenin ilk değerini sıfırla
4. Aşağıdaki işlemi `n` adet tekrarla
  - a. `top=top+u(i)` %Bu işleme yığılmalı toplam adı  
%verilir
5. Çıktı: dizi elemanlarının toplamı: `top` değişkeninin değeri

#### Kod: `toplam(skaler cebirsel versiyon)`

```
u=[1,2,-2,3,4,5,0,6,7,8,4];  
n=length(u);  
top=0;  
for i=1:n  
    top=top+u(i);
```

```
end
top      % sonucun ekranda görünmesini sağlar
```

Toplam kodunun 1. satırında bir  $u$  dizisi tanımlanmaktadır. Değişken olarak tanımlanan  $i$  değişkeninin 1 den  $n=length(u)=11$  e kadar olan her bir değeri için  $u$  dizisinin  $i$ -inci elemanı başlangıç değeri sıfıra eşit olan  $top$  değişkeninde saklı bulunan değere ilave ediliyor. İşlem sonucunda  $top$  değişkeni  $u$  dizisinin elemanlarının toplamını içerir.

**Test:** toplam

```
>>toplam
>>top=38
```

elde ederiz.

#### b) Vektör cebirsel versiyon

$N$  elemanlı bir  $u$  dizisinin elemanlarının toplamı, alternatif olarak

$$\sum_{i=1}^n u(i) = \langle [1 \ 1 \ 1 \ \dots \ 1], [u(1) \ u(2) \ \dots \ u(n)] \rangle$$

iç çarpımı yardımıyla da hesaplanabilir. Dolayısıyla toplam işlemi 1 rakamlarından oluşan bir vektör ile  $u$  vektörün iç çarpımı olarak elde edilebilir.

**Algoritma:** toplam(vektör cebirsel versiyon)

1. Girdi:  $u$  isimli ve sonlu elemanlı bir dizi
2. Dizin eleman sayısını hesaplayarak  $n$  isimli değişkene atayınız
3. Birler isimli  $n$  elemanlı 1 rakamlarından oluşan bir vektör tanımlayınız
4.  $\langle \text{birler}, u \rangle$  iç çarpımını hesaplayınız ve  $top$  değişkenine atayınız.
5. Çıktı: dizi elemanlarının toplamı:  $top$  değişkeninin değeri

**Kod:** toplam(vektör cebirsel versiyon)

```
u=[1,2,-2,3,4,5,0,6,7,8,4];
n=length(u);
birler=ones(1,n);
```

```
top=birler*u'    % sonucun ekranda görünmesini sağlar
```

### c) OCTAVE sum fonksiyonu ile toplam

Aynı sonucu OCTAVE' ın sum fonksiyonu yardımıyla da elde edebiliriz:

```
>>u=[1,2,-2,3,4,5,0,6,7,8,4];
```

```
>> sum(u)
```

```
ans =
```

```
38
```

**NOT:** Sayaç değişkeni birer birer artmaması durumunda

başlangıç\_değer:son\_değer

yerine

başlangıç\_değer:artış\_miktarı:son\_değer

yazılımı kullanılır.

### **ÖRNEK 6.7**

Bir u dizisinin tek indisli terimlerinin toplamını hesaplayan aşağıdaki örneği inceleyelim.

```
top=0;
```

```
u=[1,2,-2,3,4,5,0,6,7,8,4];
```

```
n=length(u);
```

```
for i=1:2:n
```

```
    top=top+u(i);
```

```
end
```

```
top    % sonucun ekranda görünmesini sağlar
```

```
>>top=14
```

elde ederiz.

Öte yandan bir dizinin elemanları yerine, bir matrisin her bir elemanı üzerinde herhangi bir işlem yapılmak isteniyorsa, bu defa iç içe for döngüsü kullanılır. İç içe for döngüsü

```
for değişken_1=başlangıç_değer:son_değer
    for değişken_2=başlangıç_değer:son_değer
        deyimler
    end
end
```

formatında kullanılır.

### **ÖRNEK 6.8**

İç içe for döngülerinin kullanımını açıklamak amacıyla aşağıda tanımlanan bir A matrisinin elemanlarını toplayan bir program hazırlayalım. Aynı işlemi OCTAVE sum fonksiyonu yardımıyla da gerçekleştirelim.

#### **Yöntem ve adımları**

1. Matrisi tanımlayalım,
2. matrisin satır ve sütunlarını belirleyerek uygun değişkenlere atayalım,
3. toplam için uygun değişkenin ilk değerini sıfırlayalım,
4. her bir satırın elemanlarını yığılmalı toplam yardımıyla toplayalım.

#### **Algoritma: mattop**

1. Girdi: A matrisi,
2. m,n: matrisin satır ve sütun elemanları,
3. toplam için top değişken değerini sıfırlayınız,
4. her bir  $i=1, \dots, m$  ve  $j=1, \dots, n$  için aşağıdaki işlemi tekrarla
  - a.  $top=top+A(i,j)$
5. Çıktı: top değişken değeri.

#### **Kod: mattop**

```
% A matrisinin elemanlarını toplar
```

```
A=[ 2 3 4 ;1 3 -4];
M_top=sum(sum(A));
printf('Toplam=%d\n',M_top);

top=0;
```

```
A matrisi
tanımlanıyor.
Sum komutunun
birinci defa
kullanımı ile her
```

## Octave ile Kodlama

```
[m,n]=size(A);  
for i=1:m %sattır deęiřkeni  
    for j=1:n %sütun deęiřkeni  
        top=top+A(i,j);  
    end  
end  
  
printf('Toplam=%d\\n',top)
```

bir sütunun elamanlarının toplamından oluşan bir satır vektörü elde edilir. İkinci kullanımında ise elde edilen satır vektörünün elemanlarının toplamı ve böylece matrisin bütün elemanlarının toplamı elde ediliyor.

A matrisi tanımlanırken köşeli parantez, elemanlarına erişilirken ise yuvarlak parantez kullanımına dikkat ediniz!

Sonuç tamsayı formatında yazdırılıyor.

**Test:**mattop

```
>>mattop  
>>Toplam=9
```

Deyimler grubunun kaç kez tekrarlanacağını önceden bilinmemesi durumunda ise while döngüsü kullanılır. While döngüsü

```
While mantıksal ifade  
    deyimler  
end
```

formatında kullanılır.

```
>> U=[1,2,-2,3,4,5,0,6,7,8,4];
```

```
>> i=1
>> top=0;
>> while i<=length(U)
>>     top=top+U(i);
>>     i=i+1;
>> end
>> top
```

Bu program parçasığında görüldüğü üzere *i* değişkenine atanan ilk değer (*i*=1), `length(U)` yardımıyla belirlenen *U* dizisi içerisindeki eleman sayısından(11) küçük ya da eşit olan kadar `while` döngüsü içerisinde yer alan deyimler tekrarlanmakta ve döngüden çıkıldığında *top* değişkeni *U* dizisinin elemanlarının toplamını içermektedir. `For` döngüsünde sayaç değişkeninin değeri otomatik olarak bir arttığı halde, `while` döngüsünde değişken değeri deyimler grubu içerisinde artırılmalıdır. `While` döngüsü deyimler grubunun ne kadar tekrar edeceğinin önceden bilinmemesi durumunda tercih edilir.

### ÖRNEK 6.9

Örnek 6.2 de incelenen ve katsayıları girilen ikinci dereceden polinomun köklerini hesaplayan programda *a* katsayısının sıfırdan farklı girilmesi gerektiğini ifade eden ve sadece sıfırdan farklı değer için işlem yapan bir tekrarlı yapı ilave edelim.

```
a=input('a=');
b=input('b=');
c=input('c=');
while (a==0)
    disp('a sıfırdan farklı olmalıdır');
    a=input('a=');
end
...
```

Yukarıda görüldüğü üzere *a* değeri sıfıra eşit olduğu süreçte, kullanıcı uyarılmakta ve sıfırdan farklı bir değer girmesi istenmektedir.

### ÖRNEK 6.10

Bilgisayar tarafından rasgele üretilen ve 0 ile 100 arasındaki bir tamsayının kullanıcı tarafından tahmin edilmesi için `tahmin.m` dosyasına kayıtlı basit bir oyun programı geliştirelim.

### Yöntem ve adımları

1. 0 ile 100 arasında bir sayı üretelim,
2. kullanıcının tahminini alalım,
3. tahmin sayısını belirlemek için ilk değeri sıfır olan bir değişken tanımlayalım,
4. kullanıcının tahmini üretilen sayıdan küçük ise kullanıcının daha büyük bir sayı girmesini, değilse daha küçük bir sayı girmesini isteyerek kullanıcıyı yönlendirelim ve
5. her defasında denemelerin sayısını artıralım. Üretilen sayı bulunduğunda ise “ .... deneme sonunda buldunuz , tebrikler “ mesajı ile oyunu sonlandıralım.

### Algoritma: Tahmin

1. Öncelikler sıfır ile 100 arasında rasgele bir sayı üretelim ve üretilen sayıyı `sayi` isimli değişkene atayınız.
2. Tahmin sayısını belirlemek için ilk değeri sıfır olan `sayac` isimli bir değişken tanımlayınız.
3. Kullanıcıdan bir tahmin isteyerek `a` değişkenine atayınız.
4. `a` değeri `sayi` dan farklı olduğu sürece
  - a. `a>sayi` ise daha küçük sayı girilmesini isteyerek `sayac` değişken değerini artırınız,
  - b. `a<sayi` ise daha büyük sayı girilmesi gerektiğini ifade ederek `sayac` değişkeni değerini artırınız.
5. Kaçıncı deneme sonunda üretilen sayının bulunduğunu yazarak programı sonlandırınız..

### Kod: Tahmin

```
% Bilgisayar tarafından rasgele üretilen 0 ile 100
%arasındaki sayının kullanıcı tarafından tahmin
%edilmesini
%sağlar.
```

```
sayi=round(100*rand);
```

```
sayi isimli rasgele sayı
```



```

a=input('tuttuğum sayı=');
sayac=0;
while a~=sayi
    if a>sayi
        disp('daha küçük');
        a=input('tuttuğum sayı=');
        sayac=sayac+1;
    elseif a<sayi
        disp('daha büyük');
        a=input('tuttuğum sayı=');
        sayac=sayac+1;
    end
end
disp([num2str(sayac),
'deneme sonunda buldunuz, tebrikler']);

```

üretiliyor.  
 a tahmini kullanıcıdan  
 alınıyor.  
  
 Kullanıcı tutulan sayıya  
 doğru yönlendirilirken,  
 sayaç değeri her bir  
 tahmin için artırılıyor.  
  
 Tahmin mesajı ve deneme  
 sayısı iletiliyor.

**Test:**tahmin

```

>> tahmin
tuttuğum sayı=45
daha büyük
tuttuğum sayı=75
daha büyük
tuttuğum sayı=85
daha küçük
tuttuğum sayı=80
daha büyük
tuttuğum sayı=82
daha küçük
tuttuğum sayı=81
5 deneme sonunda buldunuz, tebrikler

```

### ÖRNEK 6.11

Verilen reel değerli bir fonksiyonun yine verilen bir  $[a, b]$  aralığındaki (eğer mevcutsa) bir sıfır yerini içeren  $dx$  uzunluklu alt aralığı belirleyiniz. Hazırladığınız programı altaralik.m isimli dosyada saklayarak  $f(x) = e^x - (x + 3)$  fonksiyonu için  $[1, 3]$  aralığında  $dx = 0.1$  adım uzunluğu ile çalıştırınız.

#### **Yöntem ve adımları**

1. fonksiyonu tanımlayalım,

2. verilen aralık uç noktalarını ve sıfır yerinin içerecek alt aralık uzunluk değerlerini tanımlayalım,
3. sıfır yerini içeren aralık bulunmadığı sürece
  - a. sağa doğru küçük adımlarla ilerleyelim,
  - b. Sıfır yeri bulunduğu anda elde edilen aralığı kullanıcıya istenilen alt aralık olarak iletelim,
  - c. değilse sıfır yerini içeren alt aralık bulunamadı yazarak ve işlemi sonlandıralım.

**Algoritma:** altaralik

1.  $f$  fonksiyonu,  $[a, b]$  aralığı ile  $dx$  değerini alınız
2. Sıfır yerinin belirlenebilmesi amacıyla bulunamadı isimli bir değişken tanımla ve ilk değerini 1 e eşitle.
3. bulunamadı değişken değeri 1 ve  $a < b$  olduğu sürece
  - a.  $a = a + dx$  olarak tanımla,
  - b. Eğer  $f(a)f(a + dx) \leq 0$  şartını sağlayan  $[a, a + dx]$  aralığı mevcut ise bu aralığı yaz ve bulunamadı değişken değerini sıfır yap,
4. bulunamadı değişken değeri hala 1 e eşitse sıfır yerini içeren aralık bulunamadı yaz ve çık.

**Kod:** altaralik

```
% inline komut satır fonksiyonu ile tanımlanan
fonksiyonun verilen [a,b] aralığındaki sıfır yerini
içeren dx uzunluklu alt aralığı belirler.
```

```
f=inline('exp(x)-x-3');
a=input('a=');
b=input('b=');
dx=input('dx=');
bulunamadi=1;
```

```
while bulunamadi & (a<=b)
    if f(a)*f(a+dx)<=0
        fprintf('sıfır yerini
içeren aralık: ');
        fprintf('
%3f,%3f\n',a,a+dx);
```

Fonksiyon tanımlanmakta  
a,b aralık uç noktaları  
ve dx artım değeri  
aşınmakta

a<=b olduğu sürece sıfır  
yerini içeren aralık  
bulunana kadar  
f(a)f(a+dx)<=0 kriterini  
sağlayan [a,a+dx]  
aralığını araştır.

```

        bulanamadi=0;
    else
        a=a+dx;
    end
end
if bulanamadi
    disp('sıfır yerini içeren
aralık bulunamadı');
end

```

**Test:** altaralik

```

>> altaralik
a=1
b=3
dx=0.1
sıfır yerini içeren aralık:  1.500000,1.600000

```

Sıfır yeri bulunamadığı  
mesajını yaz.

### ÖRNEK 6.12

Bir kısmi diferensiyel denklemin çözümü olarak elde edilen

$$U(x, t) = \sum_{n=1}^N (1 - (-1)^n) e^{-\pi^2 n^2 t} \sin(n\pi x) / n^3$$

fonksiyonunun tanımlanan  $x$  ve  $t$  vektörlerindeki değerlerinde oluşan  $U$  matrisini  $[x, t] \in [0, 1] \times [0, 0.2]$  bölgesi üzerinde  $dx = 0.1, dt = 0.01$  adım uzunluklarıyla hesaplayarak grafiğini çizdiren programı

- İç içe for döngüleri yardımıyla ve **skaler cebirsel işlemlerle**
- İç içe döngü kullanmaksızın, noktasal çarpım operatörü yardımıyla ve **vektör cebirsel işlemler** ile hesaplayınız.

Burada **skaler cebirsel işlem** ile skalerler üzerinde yapılan aritmetik işlemi, **vektör cebirsel işlem** ile doğrudan vektörler ile yapılan toplama, çıkarma, noktasal çarpma ve noktasal bölme işlemlerini kastediyoruz.

Öncelikle iç içe for döngüleri yardımıyla  $U_{ij} = u(x_i, t_j)$  elemanlarından oluşan  $U$  matrisini hesaplayan bir program hazırlayalım.

### Yöntem ve adımları

1.  $N$  sayısını tanımlayalım.
2.  $\mathbf{x}$  ve  $\mathbf{t}$  vektörlerini ve bu vektörlerin eleman sayılarını belirleyelim,
3.  $i$  ve  $j$  sırasıyla  $\mathbf{x}$  ve  $\mathbf{t}$  nin eleman indisleri olmak üzere  $i$  ve  $j$  nin her bir değeri için
  - a. toplamı saklayacak değişkenin, örneği top, ilk değerini sıfırlayalım,
  - b. Toplam indisinin 1 den  $N$  e kadar değerleri için seri genel terimini top değişkenine yığılmalı toplam kuralı ile ilave edelim.
4. Top değerini  $u$  nun  $(i,j)$  –inci elemanı olarak atayalım.

**Algoritma:** u\_matrisi

1. Girdi:  $N, \mathbf{x}, \mathbf{t}$
2. m:  $\mathbf{x}$  in eleman sayısı; n:  $\mathbf{t}$  nin eleman sayısı olmak üzere
3. i indisinin 1 den m; j indisinin 1 den n ye kadar değerleri için aşağıdakileri tekrarla
  - a. top=0;
  - b. n indisinin 1 den  $N$  ye kadar değeri için  
top=top+serinin\_n\_inci terimi
  - c.  $u(i,j)=top$

**Kod:** u\_matrisi(skaler cebirsel versiyon)

```
% Kullanıcıdan girilen N için
% toplam(n=1,...N) ((-1)^n-1e^(-n^2tsin(nx) toplamını iç-içe
for %döngüleri yardımıyla tanımlanan x=a:dx:b ve t=0:dt:T
vektörleri
% için hesaplar
```

```
x=0:0.1:1;
t=0:0.01:0.2;
N=input('N=');
for i=1:length(x)
    for j=1:length(t)
        top=0;
        for n=1:N
            top=top+(1-(-1)^(n))*...
                exp(-(n*pi)^2*t(j))*...
                sin(n*pi*x(i))/n^3;
        end
```

```
x ve t vektörü
tanımlanıyor
N değeri kullanıcıdan
alınmakta
Top değişkeni

Yığılmalı toplam
hesaplanmakta

Sonuç u(i,j) elemanına
atanmaktadır.
X ve T matrisleri
```

```

        u(i,j)=top;
    end
end
[X,T]=meshgrid(x,t);
mesh(X,T,u');

```

oluşturularak  
U' (U transpoz!) un  
grafiği  
çizdirilmektedir.

Yukarıdaki toplama işlemi matrislerdeki noktasal çarpım operatörü yardımıyla alternatif olarak aşağıdaki gibi daha etkin bir biçimde gerçekleştirilebilir:

**Kod:** `u_matrisi` (vektör cebirsel versiyon)

```

% Kullanıcı tarafından girilen N için
% toplam(n=1,...N) ((-1)^n-1e^(-n^2tsin(nx) toplamını noktalı
% çarpım operatörü yardımıyla hesaplar

```

```

x=0:0.1:1;
t=0:0.01:0.2;
N=input('N=');
[X,T]=meshgrid(x,t);
top=0;
for n=1:N
    top=top+(1-(-
1)^(n))*... exp(-
(n*pi)^2*T).*...
    sin(n*pi*X)/n^3;
end
u=top;
mesh(X,T,u);

```

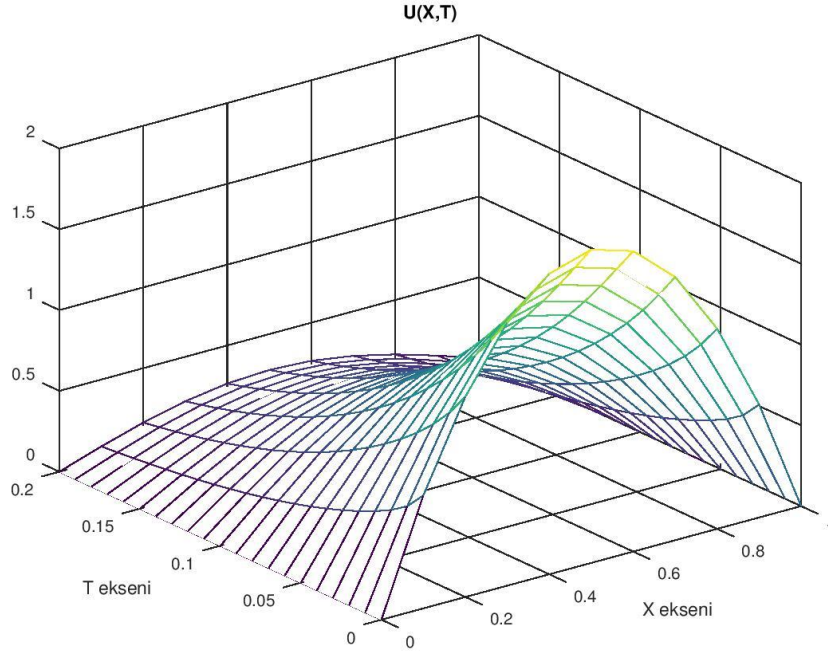
x ve t vektörü  
tanımlanıyor  
N değeri kullanıcıdan  
alınmakta

Top değişkeni  
sıfırlanmakta

Yığılmalı toplam  
hesaplanmakta;  
X ve T matrisleri ve  
noktalı çarpıma dikkat  
ediniz.  
X ve T açısı üzerinde u  
nun (transpoz değil!)  
grafiği  
çizdirilmektedir.

**Not:** Skaler cebirsel ve vektör cebirsel kodları karşılaştırdığımızda, vektör cebirsel kodun çok daha az CPU zamanı ile işlemleri gerçekleştirebileceğini tahmin edebiliriz. Vektör cebirsel versiyonda iç içe for döngüsü kullanmadığımıza dikkat ediniz. Bununla beraber vektör cebirsel versiyondaki “.\*” noktalı çarpımına dikkat edelim.

**Test:** `u_matrisi`




Şekil 6.2 Örnek 6.12 de verilen  $U(x,t)$  fonksiyonunun ağ grafiği

## 6. Bölüm Alıştırmaları

1. İşaretlenmiş değilse Desktop-  
>toolbar seçeneğini  
işaretleyerek kullanıcı ara yüzünde  
aşağıdaki bileşenlerin görünmesini  
sağlayınız.



 düğmesi yeni bir m dosya oluşturmak, sağındaki düğme ise var olan bir dosyayı açmak için kullanılır. Buna göre yeni bir m dosyası oluşturarak Örnek 6.1 de verilen programı yazıp, `ornek1.m` ismiyle kaydederek `komut` penceresinden çalıştırınız.

2. Soru 1 de hazırladığınız `ornek1.m` dosyasını `file->save as` seçeneği yardımıyla `ornek1c.m` ismiyle kaydederek, iki sayının minimumunu bulacak şekilde yeniden düzenleyerek çalıştırınız.
3. Soru 1 deki `ornek1.m` dosyasını bu kez `enbuyukuc.m` olarak kaydederek, kullanıcı tarafından girilen üç sayının büyüğünü bulacak şekilde düzenleyiniz.
4. Kullanıcı tarafından girilen üç adet  $(x_i, y_i), i = 1, 2, 3$  nokta çiftinin bir doğru üzerinde bulunup bulunmadığı kontrol eden `dogrusaltest.m` isimli program dosyası hazırlayarak doğru çalıştığını kontrol ediniz. Bu işlem için nasıl bir yöntem kullanmalısınız? Not: apsisleri  $x = [x_1, x_2, x_3]$  ve ordinatları ise  $y = [y_1, y_2, y_3]$  vektörleri olarak giriniz.
5. Kullanıcı tarafından girilen  $x, y$  vektörleri arasındaki açıyı radyan cinsinden `v_aci` komutuyla hesaplayan bir programı hazırlayınız.
6. Soru 5 e benzer olarak tanımlanan üç bileşenli iki  $x, y$  vektörün vektörel çarpımını hesaplayan `v_carp` isimli bir program dosyası hazırlayınız.
7. Soru 6 da hazırlamış olduğunuz `v_carp` program dosyanızı `cross` fonksiyonuyla karşılaştırınız.
8.  $\sin(x)$  ve  $\cos(x)$  fonksiyonlarının  $[0, 2\pi]$  aralığında,  $\tan(x)$  ve  $\cot(x)$  fonksiyonlarının  $\varepsilon = 0.1$  için sırasıyla  $[-\frac{\pi}{2} + \varepsilon, \frac{\pi}{2} - \varepsilon]$  ve  $[-\pi + \varepsilon, -\varepsilon]$  aralıklarında grafik penceresinin 4 alt penceresinde grafiğini çizdiren `trig.m` isimli bir program hazırlayınız. Noktalar arasındaki uzaklığı  $h = 0.1$  olarak alınız.
9.  $X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, Y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, Z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$  bir üçgenin köşe noktalarının koordinatları olmak üzere, ekrandan girilen bu değerleri köşe noktaları kabul eden üçgenin alanını, yani  $(\text{Alan}(XYZ))$  yi hesaplayan `alanucgen.m` isimli bir program hazırlayınız. Not: Öncelikle koordinatlar yardımıyla üçgenin kenar uzunluklarını belirleyiniz. Eğer üçgenin kenar uzunlukları  $a, b$  ve  $c$  ise bu taktirde  $u = (a + b + c)/2$  olmak üzere
 
$$\text{Alan} = \sqrt{u(u-a)(u-b)(u-c)}$$
 ile verildiğini hatırlayınız.
10. Soru 9 da tanımlanan köşe noktalar ve kullanıcı tarafından girilen  $A = \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix}$  matrisi için  $P = AX, Q = AY, R = AZ$  noktalarını hesaplayarak şimdi de bu noktaları köşe noktası kabul eden üçgenin alanını,  $\text{Alan}(PQR)$  yi hesaplayınız.

11.  $Alan(XYZ)$ (soru 9),

$Alan(PQR)$ (soru 10) ve  $\det(A)$  arasındaki ilişkiyi belirleyerek, bir matrisin determinantın geometrik anlamını kavramaya çalışınız.

12. Soru 10 da seçtiğiniz matrisi bu defa sütunları lineer bağımlı olan bir matris olarak seçiniz. Örneğin

$$A = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} \text{ Alan}(PQR) \text{ neye eşit oldu?}$$

13.  $\mathbb{R}^2$  de birim karenin aşağıdaki matrisler altındaki resimlerini belirlemeye çalışınız. Bunun için verilen matrislerin  $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ ,  $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$  ve  $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$  vektörlerini hangi vektörlere resmettiklerini belirleyiniz.  $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$  noktasının yine kendisine resmedileceğine dikkat ediniz.

a.  $A = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

b.  $A = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$

c.  $A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

d.  $A = \begin{bmatrix} c & 0 \\ 0 & 1 \end{bmatrix}, c > 1 \text{ veya } 0 < c < 1$

e.  $A = \begin{bmatrix} 1 & c \\ 0 & 1 \end{bmatrix}, c > 1 \text{ veya } 0 < c < 1$

f.  $A = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix},$   
 $\theta = \frac{\pi}{4}, \frac{\pi}{2}, \pi, 3\pi/2$

14. Aşağıda verilen fonksiyonların belirtilen aralıklarda sıfır yerlerini

içeren  $dx=0.1$  uzunluklu alt aralıkları belirleyiniz. .

a.  $f(x) = e^x - (x + 3), [0,3]$

b.  $f(x) = x^2 - 3x + 2, [3/2,3]$

c.  $f(x) = \log(x + 2), [-3/2,0]$

15.

$$U(x, t) = \sum_{n=2}^N \left[ \frac{n}{\pi(n^2 - 1)} - \frac{1}{n^2 \pi^2} \right] e^{-\pi^2 n^2 t} \cos(n\pi x)$$

fonksiyonunun  $x=0:0.1:1$  ve  $t=0:0.01:0.3$  noktalarındaki değerlerini  $N=10$  için hesaplayarak yüzey grafiğini çizdiriniz.



## 7.BÖLÜM

### 7.OCTAVE ile Fonksiyon Programları

- ☑ Bu bölümde OCTAVE kütüphanesinde yer alan fonksiyon kütüphanesine özel amaçlı yeni programlar hazırlayarak nasıl ilave edebileceğimizi inceliyoruz.

Önceki bölümde verilen örneklerde belirli komutlar kümesinin uzantısı .m olan bir dosya içerisine yazılarak dosya isminin komut promptunda yazılarak çalıştırılmıştı. Dosya ismi yazılmak suretiyle çalıştırılabilen bu programlara betik(script) adı verilmektedir.

Alternatif olarak belirli amaca yönelik olarak hazırlanan ve gerekli parametrelerini fonksiyon argümanı olarak alarak, sonuçta bir ya da daha fazla değer geri döndüren programlar hazırlanabilir. OCTAVE da bu yapıya fonksiyon programı adı verilmektedir. Genel yazılımı

```
function  
[sonuc1,sonuc2,...]=fonksiyon_ismi(input_1,input_2,...)  
biçimindedir.
```

Burada (input\_1,input\_2,...) kullanıcı tarafından sağlanan bilgileri temsil ederler.

[sonuc1,sonuc2,...] ise kullanıcıya geri gönderilecek olan ve program içerisinde tanımlanan bilgileri temsil ederler.

**NOT 1:** `fonksiyon_ismi`, fonksiyon programının ismidir ve boşluk veya herhangi bir özel karakter içermemelidir.

**NOT 2:** Hazırlanan fonksiyon programı `fonksiyon_ismi.m` dosyasına kayıt edilmelidir.

## 7.1 Fonksiyon Program Örnekleri

### ÖRNEK 7.1.1(YIĞMALI ÇARPIM İLE FAKTORİYEL HESABI)

Girilen pozitif bir tam sayının faktoriyelini hesaplayan faktöriyel isimli bir fonksiyon programı hazırlayalım.

Bunun için öncelikle yığmalı çarpma işlemi olarak adlandırılan işlemi inceleyelim:

**Yöntem ve adımları:** yığmalı çarpım

1. sonuc değişkenine 1 değerini atayalım,
2. i indisi 1 den n e kadar  
     sonuc=sonuc\*i işlemini tekrar edelim  
     ve sonuc değişken değerini geri gönderelim.

Yukarıda 2. adımdaki işleme yığmalı çarpım adı verilir. İşlem sonunda sonuc değişkeninde  $n!$  değeri saklanmış olur. Gerçekten de  $n=3$  için

```
i=1 ise sonuc=1
i=2 ise sonuc=sonuc*2=2
i=3 ise sonuc=sonuc*i=2*3=6 değeri elde edilir.
```

**Algoritma:** faktoriyel

1. Girdi: n:pozitif tamsayı
2. sonuc=1
3. i indisi 1 den n e kadar  
     sonuc=sonuc\*i
4. Çıktı: sonuc

**Kod:** faktoriyel

```
% Yazılımı: faktöriyel(n)
% n,faktöriyeli hesaplanacak pozitif tamsayı
```

```
function sonuc=faktoriyel(n)
sonuc=1;
for i=1:n
    sonuc=sonuc*i;
end
```

```
function sonuc=faktoriyel(n)
```

satırı ile takip eden program parçasının bir fonksiyon programı olduğu ve kullanıcı tarafından faktoriyeli hesaplanmak üzere girilen sayının  $n$  değişkenine atanacağı belirtilmektedir. Ayrıca program tarafından hesaplanacak olan  $n!$  değerinin ise `sonuc` isimli değişkene atanarak ana programa veya komut penceresine geri gönderileceği belirtilmektedir. Gerçekten de `for` döngüsünün  $n$  kez tekrar etmesi durumunda `sonuc` değişkeninden  $n!$  değerinin saklanacağına dikkat edelim.

Bu örneğimizde fonksiyon ismi `faktoriyel` olduğundan, alt program `faktoriyel.m` isimli bir dosyada saklanmalıdır.

Bu durumda faktoriyel programı aşağıdaki gibi çalıştırılır:

```
Test: faktoriyel
```

```
>> faktoriyel(4)
```

```
ans =
```

```
24
```

```
veya
```

```
>> sonuc=faktoriyel(3)
```

```
sonuc =
```

```
6
```

Yukarıdaki birinci uygulamada görüleceği üzere eğer programın geri göndereceği değerin saklanabileceği herhangi bir değişken tanımlanmamışsa, ilgili değer `ans(answer)` isimli varsayılan cevap değişkenine atanır, aksi takdirde ikinci uygulamadan görüldüğü üzere program sonucu kullanıcı tarafından tanımlanan değişkene atanır.

Aynı işlem OCTAVE `factorial` fonksiyon programı yardımıyla da gerçekleştirilebilir:

```
>> factorial(5)
```

```
ans = 120
```

**Not:** Fonksiyon programları komut penceresinden yukarıda belirtildiği üzere kullanılabilir. Bir diğer fonksiyon programı tarafından da çağırılarak kullanılabilirler. Bu durumda çağırılan fonksiyon programına alt program adı verilmektedir. Örnek 7.2 te sunulan ve  $C(n, m)$  kombinasyonunu hesaplayan fonksiyon programı Örnek 7.1 de sunulan faktoriyel programını bir alt program olarak kullanarak işlemini gerçekleştirmektedir.

### ÖRNEK 7.1.2(ALT PROGRAM KULLANIMI İLE KOMBİNASYON HESABI)

Kullanıcı tarafından girilen ve negatif olmadığı kabul edilen  $m$  ve  $n$  değerleri( $n \geq m$ ) için  $C(n, m)$  kombinasyonunu hesaplayan bir fonksiyon programı hazırlayalım.

```
% Yazılımı: kombinasyon(n,m)
% C(n,m) kombinasyonunu hesaplar. (n>=m>=0)
% olmalıdır.
```

```
function sonuc=kombinasyon(n,m)
```

```
sonuc=faktoriyel(n)/(faktoriyel(m)*faktoriyel(n-
m));
```

```
>> kombinasyon(4,2)
ans =
    6
```

**Not:** Kombinasyon programının faktoriyel programını  $m$ ,  $n$  ve  $m-n$  parametreleriyle çağırarak

$$C(n, m) = \frac{n!}{m!(n-m)!}$$

formülüne uygun olarak kullandığına dikkat edelim. Bu örnekte kombinasyon programı **çağırıcı program**, faktöriyel programı ise **çağrılan program** olarak adlandırılır.

Fonksiyon alt programı kullanımının programın kısa ve anlaşılır biçimde yazılmasının yanında, esasen benzer komutların gereksiz yere program içinde tekrar edilmesini engellediğine de dikkat edelim. Alt program kullanımı, programda oluşabilecek hataların ayıklanmasını da kolaylaştırmaktadır. O halde gereksiz tekrarların önlenmesi, programın

okunabilirliği ve kolay hata ayıklaması bakımından fonksiyon alt programı kullanımı tercih edilmelidir.

### **ÖRNEK 7.1.3(VEKTÖRE ELEMAN EKLEME UYGULAMASI)**

Elemanları tam sayılardan oluşan vektörün, tek ve çift elemanlarını ayırarak kullanıcıya geri gönderen bir fonksiyon programı hazırlayalım.

#### **Yöntem ve adımları**

Listede yer alan herhangi bir elemanın tek veya çift olduğunu belirleyebilmek için OCTAVE fonksiyonlarından round fonksiyonunu kullanabiliriz:

round(x) komutu ile x e en yakın olan tamsayı elde edilir. Verilen sayının tek olup olmadığını farklı yöntemlerle belirleyebiliriz. Örneğin sayının ikiye bölümünden elde edilen kalanın sıfır olup olmadığını kontrol etmek aklımıza gelen ilk yöntem olabilir. Alternatif olarak round(sayı/2) nin sayı/2 e eşit olup olmadığını kontrol edebiliriz.

Tek sayı için round(sayı/2) sonucu sayı/2 e eşit olamaz. Örneğin

```
>> round(9/2)
ans = 5
>> 9/2
ans = 4.5000
```

dir. O halde

1. Verilen sayıyı ikiye bölerek en yakın sayıya yuvarlayalım.
2. Elde edilen sonucun sayının yarısına eşit olup olmadığını kontrol edelim. Eğer eşitse sayı çift, değilse tektir.

**Algoritma:** tekçift

1. Liste elemanlarının sayısını belirleyerek n değişkenine atayalım .
2. Elemanı olmayan tekler ve çiftler isimli birer vektör tanımlayalım .
3. 1 den n e kadar her bir liste elemanı için elemanın yarısını ysayı isimli değişkene atayalım.
4. Round(ysayı) ile ysayı değişkenlerinin değerlerini karşılaştırarak, mantıksal karşılaştırma sonucunu test isimli değişkene atayalım.
5. Eğer test değişkeninin değeri sıfır ise sayı çift olup, sayıyı çiftler isimli vektöre ilave edelim.

6. Eğer test değişkeninin değeri sıfır değilse ise sayı tek olup, sayıyı tekler isimli vektöre ilave edelim.

**Kod:** tekçift

```
function [tekler,ciftler]=tekçiftler(liste)
    n=length(liste);
    tekler=[];ciftler=[];
    for i=1:n
        ysayi=liste(i)/2;
        test=round(ysayi)==ysayi;
        if test==1 ciftler=[ciftler liste(i)];
            else
                tekler=[tekler liste(i)];
            end
        end
    end
```

**Kod:** tekçift

```
>> liste=[2 1 4 7 12 9 61 54];
>> [tekler,ciftler]=tekçiftler(liste)
```

tekler =

```
1      7      9     61
```

ciftler =

```
2      4     12     54
```

Yukarıda verilen tekçiftler isimli programı yakından inceleyelim:

tekler=[] tanımlaması ile elemanı olmayan bir vektör tanımlanmaktadır.

```
test=round(ysayi)==ysayi
```

karşılaştırması ile ysayi değerinin en yakın tamsayıya yuvarlatılması ile elde edilen değerin ysayi değerine eşit olup olmadığı kontrol edilmektedir. Eğer karşılaştırma doğru ise ysayi tamsayı olup, test değişkeni 1 değerini

alacaktır. Bu sonuç `liste(i)` nin çift sayı olması anlamına gelir. Diğer durumda `test` değişkeni sıfır değerini alır, bu durumda `liste(i)` tektir.

```
ciftler=[ciftler liste(i)];
```

`ataması` ile `ciftler` isimli vektöre `liste(i)` elemanı ilave edilmektedir. Sonuç bir satır vektörüdür.

Yukarıdaki atama işlemi yerine eğer

```
ciftler=[ciftler; liste(i)];
```

biçiminde atama yapılmış olsaydı(iki yazılım arasındaki noktalı virgül farkına dikkat ediniz), her bir eleman mevcut listenin alt satırına ilave edileceği için çiftler listesi bir sütun vektörü olurdu.

### ÖRNEK 7.1.4(BAŞARI NOT HESABI(DOSYAYA VERİ EKLEME UYGULAMASI))

Öğrenci sayısı kullanıcı tarafından girilen bir derse ait birinci ve ikinci arasınava ile final sınav notları yardımıyla başarı notunu hesaplayarak arasınava ve final notlarıyla birlikte `deneme.txt` isimli bir dosya kaydeden `ogr_liste` isimli bir fonksiyon program örneği hazırlayalım.

#### **Yöntem ve Adımları**

1. Kullanıcı tarafından girilen sayıda öğrenciye ait Öğrenci no, arasınava ve final notlarının kullanıcı tarafından interaktif olarak girilmesinin sağlayınız ve uygun değişkenlere atayalım.
2. Arasınava ve final notlarını parametre olarak kabul eden `basaritest` isimli bir fonksiyon alt programı yardımıyla  
$$R\_not=0.3*v1+0.2*v2+0.5*final;$$

Formülüne göre rakamsal notun ve

80:100 → AA  
75:79 → BA  
70:74 → BB  
60:69 → CB  
50:59 → CC  
45:49 → DC

## Octave ile Fonksiyon Programları

40:44 →DD  
25:39 →FD  
0:24 →FF

aralıklarına göre harfli notu belirleyerek ana programa geri gönderelim.

3. Belirtilen isimli dosyanın veri kaydı eklemek için açılarak, öğrenci no, arasınav, final bilgileri ile rakamsal ve harfli notların uygun bir formatta kaydedelim.

**Algoritma:** ogr\_liste

1. Girdi n: öğrenci sayısı
2. i indisi 1 den n e kadar aşağıdakileri tekrarla
  - a. her bir öğrenci için ogr\_no, vize1, vize2, final verilerini al
  - b. R\_not ile gösterilen rakam notu hesapla:
    - i. Eğer final<50 ise R\_not=0
    - ii. Değilse  
 $R\_not = 0.3 * vize1 + 0.2 * vize2 + 0.5 * final$
  - c. H\_not ile gösterilen harf notu hesapla
    - i. Eğer final<50 H\_not:başarısız
    - ii. Eğer R\_not>=80 ise H\_not->AA  
değil ve eğer R\_not>=75 ise H\_not->BA  
değil ve eğer R\_not>=70 ise H\_not->BB  
değil ve eğer R\_not>=60 ise H\_not->CB  
değil ve eğer R\_not>=50 ise H\_not->CC  
değil ve eğer R\_not>=45 ise H\_not->DC  
değil ve eğer R\_not>=40 ise H\_not->DD  
değil ve eğer R\_not>=25 ise H\_not->FD  
değilse H\_not->FF
  - d. deneme.txt dosyasını veri eklemek amacıyla aç
  - e. ogr\_no, v1, v2, final, ort, hnot verilerini dosyaya yaz ve dosyayı kapat

Yukarıda belirtilen işlevleri gerçekleştiren ana programa ogr\_liste, alt programa ise basaritest adını verelim.

**Kod:** ogr\_liste



## Octave ile Fonksiyon Programları

<pre>% Yazılımı: ogr_liste(n) % n&gt;0 öğrenci sayısı % Arasınave final sınav notlarını alarak öğrenci % başarısını hesaplar ve deneme.txt isimli dosyaya % kaydeder.</pre>	
<pre>function ogr_liste(n)  for i=1:n     ogr_no=input('öğrenci no=');     v1=input('vize1=');     v2=input('vize2=');     final=input('final=');      [ort,hnot]=basaritest(v1,v2,     final);     dosya = fopen('deneme.txt',     'a');     fprintf(dosya,'%10d %3d %3d %3d %3.1f %s\n',     ogr_no,     v1,v2,final,ort,hnot);     fclose(dosya); end</pre>	<p>Her bir öğrencinin öğrenci numarası ile v1,v2 ve final notları alınarak basaritest isimli fonksiyon alt programı yardımıyla ortalaması hesaplanıyor.</p> <p>deneme.txt isimli dosya veri ilave edilmek üzere açılmakta ve veriler uygun formatta kaydedilerek dosya kapatılmaktadır.</p>
<pre>% Arasınave final notlarını % alarak % harf ve rakam başarı notunu % hesaplar</pre>	
<pre>function [R_not,H_not] =basaritest(v1,v2,final)  if final&lt;50     H_not=char('basarisiz');     R_not=0; else  R_not=0.3*v1+0.2*v2+0.5*final ;</pre>	<p>Her bir öğrencinin vize1,vize2 ve final notları alınarak harf (H_not) ve rakam başarı notu (R_not) hesaplanarak çağıran programa gönderilmektedir.</p>

<pre> if R_not&gt;=80     H_not=char('AA'); elseif R_not&gt;=75     H_not=char('BA'); elseif R_not&gt;=70     H_not=char('BB'); elseif R_not&gt;=60     H_not=char('CB'); elseif R_not&gt;=50     H_not=char('CC'); elseif R_not&gt;=45     H_not=char('DC'); elseif R_not&gt;=40     H_not=char('DD'); elseif R_not&gt;=25     H_not=char('FD'); else     H_not=char('FF'); end end </pre>	<p>Final notunun 50 den küçük olması durumunda harf not başarısız olarak kabul edilmekte ve rakam nota ise 0 değeri atanmaktadır.</p> <p>Harf notu belirlenirken aşağıdaki ölçek kullanılmaktadır:</p> <p>80:100→AA  75:79 →BA  70:74 →BB  60:69 →CB  50:59 →CC  45:49 →DC  40:44 →DD  25:39 →FD  0:24 →FF</p>
---	--

```
[ort,hnot]=basaritest(v1,v2,final);
```

komut satırı ile v1,v2 ve final notları basaritest isimli fonksiyon alt programına input parametresi olarak gönderilmekte ve alt program içerisinde hesaplanan rakam başarı notu R\_ort ve harfli başarı notu H\_not çağıran programa ort ve hnot isimleri ile geri gönderilmektedir.

```
dosyam = fopen('deneme.txt', 'a');
```

komut satırı ile C programlama dili formatına uygun olarak deneme.txt isimli bir dosya mevcut ise veri eklenmek için, mevcut değilse veri kaydı için açılmaktadır. Dosyanın hangi amaçla açıldığı fopen fonksiyonundaki 'a' parametresi veya aynı komumdaki diğer parametreler tarafından belirlenir. Örneğin 'r' dosyayı okumak, 'w' dosyaya bilgi kayıt yapmak amacıyla kullanılır. Diğer seçenekler help komutu yardımıyla OCTAVE yardım sayfalarından öğrenilebilir. fopen fonksiyonunun geri gönderdiği ve bu örnekte 'dosya' adı verilen değişkene veri kaydı için açılan deneme.txt dosyasına ait dosya tutamacı(file handle) adı verilmektedir. Daha sonra söz konusu dosya ile ilgili her türlü işlem için

dosyanın kendi ismi(deneme.txt) yerine bu dosya tutamacı(function handle) kullanılır.

```
fprintf(dosya, '%10d%3d%3d%3d%5.1f%3s\n', ogr_no,  
v1,v2, final, ort, hnot);
```

komut satırı ile dosyam isimli dosya tutamacının ait olduğu dosyaya, yani deneme.txt dosyasına, veri kaydı yapılmaktadır. Tırnak işareti içerisinde yer alan ve % işaretini takip eden her bir ifade,

```
'%10d%3d%3d%3d%5.1f%3s\n',                                ogr_no,  
v1,v2, final, ort, hnot
```

sırasıyla tırnak işareti dışında yer alan her bir verinin dosya kayıt formatını belirtir. Buna göre

```
ogr_no:%10d;    v1,v2    ve    final:%3d,ort:%5.1f    ve  
hnot:%3s
```

formatında kaydedilecektir. %10d format belirtimi, bu formatta kaydedilecek olan ogr\_no verisinin 10 rakamlı bir alana sağa yaslı olarak ve tamsayı formatında kaydedileceğini belirtmektedir. Benzer biçimde %3d format belirtimi v1,v2 ve final notlarının her birinin 3 hanelik alanlara ve sağa yaslı olarak kaydedileceği; %5.1f format belirtimi ise virgülden sonra bir basamaklı ve toplam 4 haneli ondalıklı sayının, bir hanelik alan ondalık nokta için kullanılmak üzere 5 hanelik bir alana kaydedileceğini ifade etmektedir. Ondalıklı sayının virgülden sonraki birden fazla rakamı olması durumunda, bu sayıya en yakın olan ve virgülden sonra bir basamak içeren yaklaşık sayı kaydedilir. Son olarak %3s format belirtimi ise bu formatta karakterler dizisinden oluşan bir verinin üç hanelik bir alana ve sağa yaslı olarak kaydedileceğini ifade etmektedir.

**Test:** ogr\_liste

```
>>ogr_liste(3)  
öğrenci no=125441  
vize1=40  
vize2=60  
final=58  
öğrenci no=125442  
vize1=55
```

```
vize2=70  
final=48  
öğrenci no=125443  
vize1=60  
vize2=80  
final=95
```

Program çalıştıktan sonra elde edilen deneme.txt dosyası aşağıdaki gibidir:

```
125441 40 60 58 53.0 CC  
125442 55 70 48 0.0 basarisiz  
125443 60 80 95 81.5 AA
```

**NOT:** Değişken isimlerinde olduğu gibi fonksiyon isimlerinde de Türkçe karakterler ve alt çizgi haricinde özel karakterler kullanmamaya özen gösterelim.

### 7.1 Bölüm Alıştırmaları

1. Örnek 7.1.1 de verilen faktoriyel programını çalıştırarak, 4!, 7!, 10! değerlerini hesaplayınız.
2. Örnek 7.1.1 de verilen faktöriyel programını girilen sayının negatif olmayan tamsayı olmaması durumunda uygun mesajla uyaracak şekilde yeniden düzenleyiniz.
3. Örnek 7.1.2 de verilen kombinasyon programını çalıştırarak  $C(7,4)$ ,  $C(12,6)$  kombinasyonlarını hesaplayınız.
4. Örnek 7.1.2 de  $C(n,m)$  kombinasyonu hesabında  $n < m$  olması durumunda kullanıcıyı uygun mesajla uyaracak biçimde yeniden düzenleyiniz.
5. Örnek 7.1.4 te verilen ogr\_liste programını 4 öğrenciye ait arasınnav ve final notları için test yapınız.

## 7.2 Bazı Pratik Matematiksel Uygulamalar

Bu bölümde temel matematiksel alanlarda karşılaşılan bazı tipik problemlere(Bkz. [5,6]) ait OCTAVE fonksiyon programları geliştirerek, sayısal problemlere ait kod geliştirme yeteneğinin geliştirilmesini amaçlıyoruz.

### 7.2.1 Yakınsak dizi limiti

Reel bir sıfır yerine sahip  $f$  fonksiyonu ve  $ft$  ile gösterilen türevi ile sıfır yerine yakın seçilen  $x_0$  noktasını kullanıcıdan alarak

$$x_{n+1} = x_n - \frac{f(x_n)}{ft(x_n)}, n = 0,1,$$

ile tanımlanan dizinin ardışık elemanları arasındaki fark verilen yeterince küçük pozitif bir *epsilon* den küçük kalıncaya kadar dizi elemanlarını hesaplayan ve elde edilen en son noktayı kullanıcıya gönderen ve

`Newton_dizi(f, ft, x0, epsilon)`

isimli bir program hazırlayınız.

**Yöntem ve adımları:** `Newton_dizi`

1.  $x_0$  ile başlayarak, yukarıda tanımlanan dizi elemanlarını hesaplayalım.
2. Her adımda dizinin yeni elemanı ile bir önceki eleman arasındaki farkın mutlak değerini hesaplayalım,
3. Hesaplanan farkın kullanıcı tarafından girilen epsilondan küçük olup olmadığını kontrol edelim.
4. Bu işlemi söz konusu fark epsilondan küçük oluncaya kadar devam ettirelim.
5. Elde edilen en son elemanı kullanıcıya gönderelim.

**Algoritma:** `Newton_dizi`

1. Girdi:  $f, fp$  satır fonksiyonları,  $x_0$  başlangıç değeri, küçük bir  $\epsilon > 0$  sabiti
2. Devam değişkenine 1 değerini atayınız
3. Devam değişkeni değeri 1 e eşit olduğu sürece
  - a.  $x_1 = x_0 - f(x_0)/fp(x_0)$  değerini hesaplayınız
  - b.  $fark = \text{abs}(x_1 - x_0)$  değerini hesaplayınız

- c. eğer  $\text{fark} < 0$  ise devam değişkenine sıfır değerini ata,  $x_1$  değerini geri gönderiniz  
d. değilse  $x_0 = x_1$  olarak al ve 3-a ile işleme devam ediniz.

### Kod: Newton\_dizi

```
% Yazılımı: x1=newton_dizi(f,ft,x0,epsilon)
% f: sıfır yeri belirlenecek inline fonksiyonu
% ft: f nin türevi,
% x0 başlangıç değeri
% işlem sonuçlandırmak için kullanılan parametre
```

```
function x1=newton_dizi(f,ft,x0,epsilon)
    devam=1;
    while devam==1
        x1=x0-f(x0)/ft(x0)
        fark=abs(x1-x0);
        if fark<epsilon devam=0;
        else
            x0=x1;
        end
    end
end
```

### Test: Newton\_dizi

```
>> f=inline('x^2-2');
>> ft=inline('2*x');
>> x1=newton_dizi(f,ft,5,0.01)
x1 = 2.7000
x1 = 1.7204
x1 = 1.4415
x1 = 1.4145
x1 = 1.4142
x1 = 1.4142
```

O halde dizi  $x = 1.4142$  noktasına yakınsamaktadır. Bu değer  $f(x) = x^2 - 2$  fonksiyonunun sıfırlarlarından birisi olan  $\sqrt{2}$  için bir yaklaşım olduğuna dikkat edelim.

**NOT:**

- 1 Her defa elde edilen dizinin yeni değerine  $x1$ , bir öncekini ise  $x0$  olarak adlandırdığımıza dikkat edelim.
- 2 Bu örnekte elde edilen dizi,  $f$  fonksiyonunun  $x_0$  noktası komşuluğundaki sıfırına yakınsaması beklenen ve Sayısal Analiz derslerinde incelenen **Newton\_Rapson yöntemi** ile elde edilmiştir.

**7.2.2 Sayı serisi toplamı**

Genel terimi  $a_n$  sayısı ile verilen yakınsak veya ıraksak (ancak  $\lim_{n \rightarrow \infty} a_n = 0$  şartını sağlayan) bir serinin  $S_N$  ile gösterilen kısmi toplamlar dizisinin

$$|S_N - S_{N-1}| < \epsilon$$

şartını sağlayan ilk  $N$  terimini toplamını hesaplayan

$$\text{seri}(a_n, \epsilon)$$

isimli bir program hazırlayınız.

**Yöntem ve adımları:** Seri\_toplam

Verilen

$$\sum_{n=1}^{\infty} a_n$$

serisinin ilk  $N$  terimini toplamı

$$S_N = \sum_{n=1}^N a_n$$

olmak  $|S_N - S_{N-1}| = |a_n|$  elde ederiz. Yakınsak seri için  $\lim_{n \rightarrow \infty} a_n = 0$  gerek şart olduğu için

$$|S_N - S_{N-1}| = |a_n| < \epsilon$$

şartının kontrol edilmesi yeterlidir.

Bu amaçla  $a_n = a(n)$  nin satır fonksiyonu olarak tanımlandığını kabul edelim.  $|a(n)| < \epsilon$  olana kadar dizi elemanlarını hesaplayalım. Kriterin sağlandığı ilk değeri  $N$  olarak kabul edelim. Daha sonra  $ndizi=1:1:N$  dizisini tanımlayarak bu dizinin her bir elemanı için  $a(n)$  i yani  $a(ndizi)$  vektörünü hesaplayalım.

$$\sum_{n=1}^N a_n = [1, 1, \dots, 1], a(ndizi) >$$

biçiminde iki vektörün iç çarpımı olarak istenilen toplamı, Bölüm 6 da tanımlandığı üzere

```
top=0;
for i=1:N
    top=top+a(i);
end
```

**yığmal toplamı ile skaler cebirsel işlemlerle** gerçekleştirebiliriz. Ancak bu işlemi yığmal toplam ve dolayısıyla da for döngüsü kullanmaksızın çok daha az zaman gerektiren ve aşağıda verilen **vektör cebirsel algoritma** ile de hesaplayabiliriz.

**Algoritma:** seri\_toplam

- 2-  $a(n)$  satır fonksiyonu ve epsilon değerini alınız.
- 3-  $|a(n)| < \epsilon$  şartını sağlayan ilk  $n$  değerini bularak  $N$  olarak atayınız.
- 4- 1 lerden oluşan  $N$  elemanlı bir **Birler** isimli vektör tanımlayınız.
- 5-  $Ndizi=[1,2,\dots,N]$  dizisi ve  $a(Ndizi)$  dizisini tanımlayınız.
- 6-  $\langle Birler, a(Ndizi) \rangle$  iç çarpımını hesaplayınız.

**Kod:** seri\_toplam

```
% Yazılımı: toplam=seri_toplam(a,epsilon)
% a: satır fonksiyonu öyle ki a(n) serinin n-inci
% terimi
% epsilon>0 öyle ki toplam işlemi |a(N)|<epsilon
% şartını % sağlayan ilk N>0 tamsayısına kadar
% hesaplanır
```

```
function toplam=seri_toplam(a,epsilon)
    devam=1;n=1;
    while devam
        if abs(a(n))<epsilon N=n;
            devam=0;
        end
        n=n+1;
    end
```



```

birler=ones(1,N);
Ndizi=1:N;
andizi=a(Ndizi); % yığmalı toplam
kullanmadı
toplam=birler*andizi'; % ğımıza dikkat ediniz.

```

**NOT:** Yukarıda verilen kodun son dört satırında seri toplamı birler ve andizi isimli iki vektörün iç çarpımı olarak elde edilmiştir. Vektörler üzerinde gerçekleştirilen bu işleme **vektör cebirsel işlem**, ilgili algoritmaya da **vektörel algoritma** adı verilmektedir ve OCTAVE ile hazırlanan kodlarda bu yaklaşım kullanılması döngü gerektiren skalerler üzerindeki **geleneksel skaler cebirsel işlemlere** göre daha avantajlıdır. Çünkü OCTAVE ortamında döngü işleminin bilgisayarın daha fazla zamanını aldığı bilinmektedir.

**Test:** seri\_toplam

$$\sum_{n=1}^{\infty} \frac{1}{n^2}$$

serisinin toplamını yaklaşık olarak hesaplamaya çalışalım: Öncelikle  $a(n)$  dizisini vektörel argüman kabul edebilecek biçimde noktalı operatörler yardımıyla ve satır fonksiyonu olarak tanımlamalıyız:

```

>> a=inline('1./n.^2');
>> seri_toplam(a,1e-7)
ans = 1.6446
>> seri_toplam(a,1e-8)
ans = 1.6448
>> seri_toplam(a,1e-9)
ans = 1.6449

```

elde ederiz, burada  $1e-7 = 1 \times 10^{-7}$  dir. Gerçek toplam değerinin

```

>> pi^2/6
ans = 1.6449

```

olduğunu hatırlayalım.

Şimdi de

$$\sum_{n=1}^{\infty} \frac{1}{n(n+1)}$$

toplamını hesaplamaya çalışalım:

```
>> a=inline('1./(n.*(n+1))');
>> seri_toplam(a,1e-7)
ans = 0.99968
>> seri_toplam(a,1e-8)
ans = 0.99990
>> seri_toplam(a,1e-9)
ans = 0.99997
>> seri_toplam(a,1e-10)
ans = 0.99999
```

Gerçek toplam 1 e eşittir.

Şimdi de yukarıdaki algoritmayla hesaplanan toplam değerlerini yığmalı toplamla da gerçekleştirerek, iki işlem arasındaki işlem zaman farkını anlamaya çalışalım. Bu amaçla öncelikle işlem zamanını ölçmek için OCTAVE cputime fonksiyonunu inceleyelim:

#### CPUTIME

Herhangi bir işlemin gerçekleşme zamanını(saniye) ölçmek için kullanılır. Aşağıda 200x200 boyutlu ve elemanları rasgele üretilmiş bir matrisin indirgenmiş eşelon formunun elde edilmesi için gerekli zaman belirlenmektedir:

```
>>
t=cputime;A=rand(200,200);C=rref(A);zaman=cputime-t

zaman =

3.5469
```

Yukarıda t=cputime ile OCTAVE oturumu başlatıldıktan sonra geçen süre saniye cinsinden hesaplanarak t değişkenine atanmaktadır. Rand komutuyla 200x200 boyutlu matris üretilerek, rref komutu ile de indirgenmiş eşelon formu elde edilmekte ve sonuç C değişkenine atanmaktadır. Son olarak zaman=cputime-t komut ile komutun girildiği

andaki oturum zamanı ile ara işlemler başlatılmadan önceki oturum zamanı arasındaki fark hesaplanarak zaman değişkenine atanmaktadır. Böylece zaman değişkeninin değeri ara işlemler(rand ve rref) için harcanan ve saniye cinsinden cpu zamanıdır.

**Kod:** Seri\_toplam (yığılmalı ve vektör cebirsel toplamın karşılaştırması )

```
% Yazılımı: toplam=seri_toplam(a,epsilon,tip)
% a: satır fonksiyonu öyle ki a(n) serinin n-inci
% terimi
% epsilon>0 öyle ki toplam işlemi  $|a(N)| < \epsilon$ 
% şartını % sağlayan ilk  $N > 0$  tamsayısına kadar
% hesaplanır.
% tip=1 ise toplam vektör cebirsel işlem ile
% tip=2 ise yığılmalı toplam ile gerçekleştirilir.
```

```
function toplam=seri_toplam(a,epsilon,tip)
    if nargin==1 epsilon=1e-5; tip=1;% sadece a
    girilmiş
                                % ise epsilon=1e-5;
    tip=1
        elseif nargin==2 tip=1; % sadece a ve
    epsilon
                                % girilmiş ise tip=1
    end
    devam=1;n=1;
    while devam
        if abs(a(n))<epsilon N=n;
            devam=0;
        end
        n=n+1;
    end
    if tip==1
        birler=ones(1,N);
        ndizi=1:N;
        andizi=a(ndizi);
        toplam=birler*andizi';
    else
        toplam=0;
        for i=1:N
            toplam=toplam+a(i);
        end
    end
end
```

```

end
end

```

```

Test: Seri_toplam
>>a=inline('1./n');
>>t=cputime;toplam=seri_toplam(a,1e-5,1);
>> zaman=cputime-t
zaman = 6
>>t=cputime;toplam=seri_toplam(a,1e-5,2);
>> zaman=cputime-t
zaman = 10

```

**Sonuç:** Yukarıdaki test sonucundan görüldüğü üzere vektör cebirsel  $N=100001$  adet terim içeren toplam işlemi 6 saniye de gerçekleştirilirken, yığılmalı toplam 10 saniyede gerçekleştirilmiştir.

O halde mümkün olduğu kadar, skaler cebirsel ve döngü gerektiren işlemler yerine vektör cebirsel işlemler tercih edilmelidir.

### 7.2.3 Sayısal türev

Örnek 4.1 de komut dosyası yardımıyla hesapladığımız ileri fark yöntemiyle sayısal türev işlemini, verilen  $f$  fonksiyonu,  $[a, b]$  aralığını ve kullanılacak alt aralık sayısını kullanıcıdan alarak, ileri fark yöntemiyle sayısal türev işlemini gerçekleştiren `sturev` isimli bir fonksiyon programı hazırlayalım.

#### Yöntem ve adımları

1.  $[a, b]$  aralığını  $n$  adet alt aralığa bölerek elde edilen alt aralıkların uç noktalarını  $x$  vektörüne atayalım.
2. Söz konusu  $x$  noktalarındaki fonksiyon değerlerini  $y = f(x)$  ile tanımlanan  $y$  vektörüne atayalım.
3.  $h = (b - a)/n$  noktalar arasındaki uzaklık olmak üzere  

$$yp = (y(x + h) - y(x))/h$$
vektörü ileri fark yöntemiyle sayısal türev değerlerini elde edelim.

**Algoritma:** `sturev`

## Octave ile Fonksiyon Programları

1. F fonksiyonu, a,b değerleri ( $a < b$ ) ve n pozitif tamsayısı ve grafik seçeneği graf parametresini alınız
2.  $h = (b-a)/n$  adım uzunluğunu tanımlayınız
3.  $x = a:h:b$  vektörünü tanımlayınız
4.  $y = f(x)$  vektörünü tanımlayınız
5.  $yp = (y(x+h) - y(x))/h$  değerini hesaplayınız
6. eğer graf=1 ise x in ilk n elemanına karşı yp nin grafiğini çiziniz ve programı sonlandırınız.
7. değilse yp değerlerini geri gönderiniz.

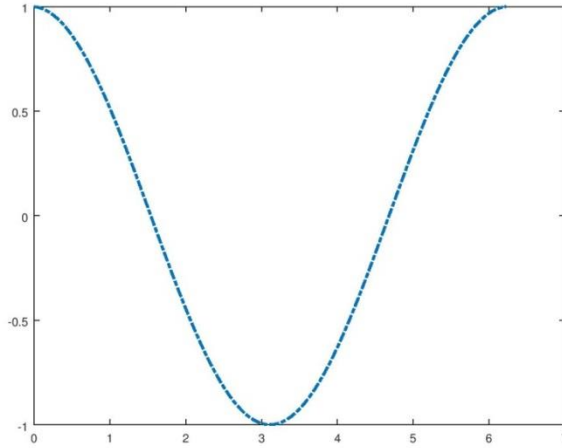
**Kod:** sturev

```
% Yazılımı: yp=sturev(f,a,b,n,graf)
% f:[a,b] aralığında türevi hesaplanacak olan
% satır fonksiyonu
% n: [a,b] aralığının bölüneceği alt aralık sayısı
% graf=1 ise sonuçlar grafiksel olarak,
% değilse sayısal olarak sunulur.
```

```
function yp=sturev(f,a,b,n,graf)
h=(b-a)/n;
x=a:h:b;
y=f(x);
yp=(y(2:n+1)-y(1:n))/h;
if graf==1
    plot(x(1:n),yp);
    yp=[];
    return;
end
end
```

**Test:** sturev

```
>> f=inline('sin(x)');
>> sturev(f,0,2*pi,100,1)
```



Şekil 7.1 Sayısal Türev Örneği

### 7.2.4 Sayısal integral

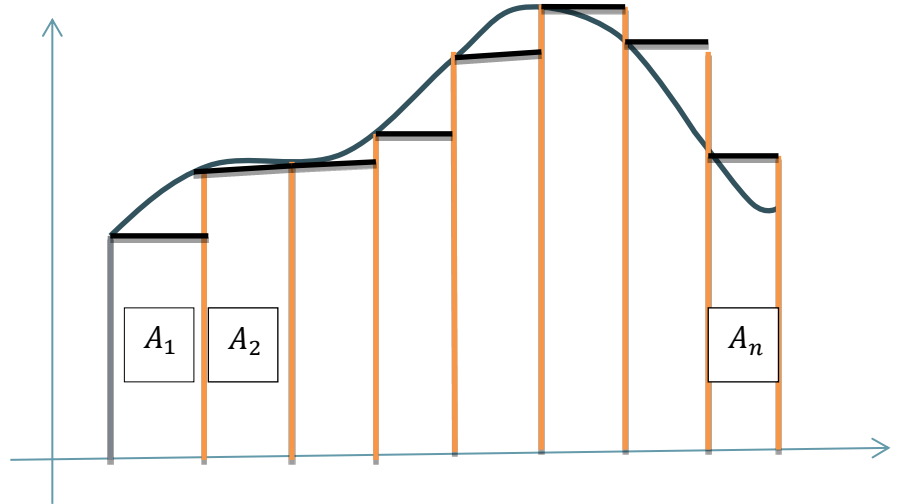
Örnek 4.2 de komut dosyası yardımıyla hesapladığımız sayısal integral işlemini, verilen  $f$  fonksiyonu,  $[a, b]$  aralığı ve sayısal integral işleminde kullanılacak alt aralık sayısını kullanıcıdan alarak, Bölüm 4.1 de tanımlanan Yamuk Kuralı yerine Sol Dikdörtgen Kuralı adı verilen integral işlemini gerçekleştiren `sintegral` isimli bir fonksiyon programı hazırlayalım. İleri düzey sayısal integral yöntemleri için [12] nolu kaynağı öneririz.

#### Yöntem ve adımları

1.  $[a, b]$  aralığını  $n$  adet alt aralığa bölerek elde edilen alt aralıkların uç noktalarını  $x$  vektörüne atayalım ( $x = [x_1, x_2, \dots, x_n, x_{n+1}]$ ).
2. Söz konusu  $x$  noktalarındaki fonksiyon değerlerini  $y = f(x)$  ile tanımlanan  $y$  vektörüne atayalım.
3.  $h = (b - a)/n$  noktalar arasındaki uzaklık olmak üzere Sol Dikdörtgen yöntemiyle

$$S_{\text{soldik}} = h(y_1 + y_2 + \dots + y_n)$$

olarak elde ederek, kullanıcıya geri gönderelim.



Şekil 7.2: Sol Dikdörtgen Yöntemiyle eğri altındaki alana yaklaşım

**Algoritma:** `sintegral`

1.  $f$  fonksiyonu,  $a, b$  değerleri ( $a < b$ ) ve  $n$  pozitif tamsayısını alınız
2.  $h = (b - a) / n$  adım uzunluğunu tanımlayınız
3.  $x = a:h:b$  vektörünü tanımlayınız
4.  $y = f(x)$  vektörünü tanımlayınız
5.  $S_{\text{soldik}}$  toplamını hesapla ve geri gönderiniz.

**Kod:** `sintegral`

```
% Yazılımı: sonuc=sintegral(f,a,b,n)
% f:[a,b] aralığında sol dikdörtgen kuralı ile
% integrali hesaplanacak olan
% satır fonksiyonu
% n: [a,b] aralığının bölüneceği alt aralık sayısı
```

```
function sonuc=sintegral(f,a,b,n)
    h=(b-a)/n;
    x=a:h:b;
    y=f(x);
    sonuc=h*sum(y(1:n));
end
```

**NOT:** integral kodunda  $y(1) + y(2) + \dots + y(n)$  toplamını OCTAVE sum komutuyla hesapladık. Aynı işlemi

```
birler=[1 1 ... 1]
```

ve

```
yler=[y(1) y(2) ... y(n)]
```

satır vektörleri tanımlayarak

```
birler*yeler'
```

biçiminde iki vektörün iç çarpımı olarak ta hesaplayabilirdik.

### Test:sintegral

```
>> f=inline('x.^2');
>> sintegral(f,0,1,1000)
ans = 0.33283
>> sintegral(f,0,1,10000)
ans = 0.33328
>> sintegral(f,0,1,100000)
ans = 0.33333
```

**NOT:** Artan  $n$  değerleri için elde edilen yaklaşımların gerçek değere yakınsadıklarını gözlemleyiniz.

## 7.2.5 Yay uzunluğu

$[a, b]$  aralığında  $y = f(x)$  bağıntısı ile tanımlanan fonsiyonun grafiği olarak elde edilen eğrinin uzunluğunun

$$L = \int_a^b \sqrt{1 + f'(x)^2} dx$$

ile verildiğini hatırlayalım. Örnek 7.3 deki sayısal türev ve Örnek 7.4 teki sayısal integral yöntemlerini kullanarak verilen  $f$  fonksiyonunun verilen  $[a, b]$  aralığı üzerindeki grafiği olarak elde edilen eğri uzunluğunu yaklaşık olarak hesaplayan bir program hazırlayalım.

### Yöntem ve adımları



1.  $[a, b]$  aralığını  $n$  adet alt aralığa bölerek elde edilen alt aralıkların uç noktalarını  $x$  vektörüne atayınız. Söz konusu  $x$  noktalarındaki fonksiyon değerlerini  $y = f(x)$  ile tanımlanan  $y$  vektörüne atayalım.
2.  $yp = (y(x+h) - y(x))/h$  ile fonksiyonun sayısal türev değerlerini içeren  $yp$  vektörünü hesaplayalım.
3.  $y = \sqrt{1 + yp.^2}$  ile integrasyon fonksiyonunun sayısal değerlerini hesaplayalım.
4.  $S_{\text{soldik}} = h(y_1 + y_2 + \dots + y_n)$  yaklaşım formülü ile istenilen yaklaşık değeri hesaplayalım.

**Algoritma:** yay\_uzunluk

1.  $f$  fonksiyonu,  $a, b$  değerleri ( $a < b$ ) ve  $n$  pozitif tamsayısını alınız
2.  $h = (b-a)/n$  adım uzunluğunu tanımlayınız
3.  $x = a:h:b$  vektörünü tanımlayınız
4.  $y = f(x)$  fonksiyon değerlerini içeren vektörü hesapla
5.  $yp = (y(x+h) - y(x))/h$  türev vektörünü hesaplayınız
6.  $y = \sqrt{1 + yp.^2}$  vektörünü tanımlayınız
7.  $S_{\text{soldik}} = h(y_1 + y_2 + \dots + y_{n-1} + y_n)$  değerini hesapla ve geri gönderiniz.

**NOT:** Orijinal  $y = f(x)$   $n + 1$  bileşen içerirken, türev vektörü olan  $y$  nin  $n$  adet bileşen içerdiğine dikkat edelim. Sol dikdörtgen kuralı ilk  $n$  adet bileşeni kullandığı için bu amaçla seçmiş bulunuyoruz.

Kod: yay\_uzun

```
% Yazılımı: sonuc=yay_uzun(f,a,b,n)
% f:[a,b] aralığında tanımlı fonksiyon olmak üzere
% (a,f(a)) ve (b,f(b)) noktalarını birleştiren
% yay uzunluğunu hesaplar.
% n: [a,b] aralığında sol dikdörtgen kuralı için
% gerekli % alt aralık sayısı
```

```
function sonuc=yay_uzun(f,a,b,n)
h=(b-a)/n;
x=a:h:b;
y=f(x);
```

```

yp=(y(2:n+1)-y(1:n))/h;
y=sqrt(1+yp.^2);
sonuc=h*sum(y(1:n));
end

```

**Test:yay\_uzun**

```

>> f=inline('sqrt(1-x.^2)')

>> yay_uzun(f,-1,1,1000)
ans = 3.1416

```

**NOT:** Gerçek değerin  $\pi$  olduğuna dikkat edelim.

## 7.2.6 İkiye bölme yöntemi ile fonksiyon sıfır yeri

$f$  fonksiyonu  $[a, b]$  aralığında sürekli ve aralığın uç noktalarında işaret değiştiriyorsa, yani  $f(a)f(b) < 0$  ise, bu taktirde  $f(c) = 0$  denklemini sağlayan bir  $c \in (a, b)$  olduğunu sürekli fonksiyonlar için aradeğer teoreminin sonucu olarak biliyoruz. Daha ileri düzey yöntemler için [10] nolu kaynağı öneririz.

### Yöntem ve adımları

İkiye bölme yöntemi bu teoremi esas alarak fonksiyonun sıfır yerini belirlemeye çalışır:

Her adımda yine  $[a, b]$  olarak adlandırılan aralığın orta noktası ( $c$ ) bulunur. Eğer

$f(a)f(c) < 0$  ise sıfır yerini içeren yeni alt aralık, yani  $[a, b]$  olarak adlandırılacak olan aralık  $[a, c]$  alt aralığı olmalıdır, o halde  $b = c$  olmalıdır. Değilse  $[a, b]$  olarak adlandırılacak olan yeni aralık  $[c, b]$  alt aralığı olmalıdır, o halde  $a = c$  olmalıdır.

**Algoritma:** ikibol

1.  $f$  fonksiyonu;  $[a, b]$  aralığının uç noktalarını, en son alt aralığın uzunluğu için gerekli tol sayısını ve nmax(maksimum iter. sayısı) değerini alınız
2.  $c=(a+b)/2$  aralık orta noktasını hesaplayınız

## Octave ile Fonksiyon Programları

3. sayac=0; döngü sayaç değişkeni değerini sıfırla
4.  $(b-a) > \text{tol}$  ve  $\text{sayac} < \text{nmax}$  olduğu sürece aşağıdaki (a)-(d) adımlarını tekrarlayınız
  - a. eğer  $f(a)f(c) < 0$  ise  $b=c$  değilse  $a=c$ ; yeni alt aralığı belirleyiniz
  - b.  $c=(a+b)/2$ ; yeni alt aralığın orta noktası
  - c.  $a, c, b, f(c)$  değerlerini yazınız
  - d. sayac değerini artırınız

**Kod:** ikibol

```
% Yazılımı: ikibol(f,a,b,tol,nmax)
% f:[a,b] aralığında tanımlı fonksiyon olmak üzere
% (b-a)>tol ve n<nmax olduğu sürece f nin sıfır yeri için
% alt aralık orta noktalar dizisini belirler ve tablo
% halinde yazdırır.
```

```
function ikibol(f,a,b,tol,nmax)
```

```
    c=(a+b)/2;
    fc=f(c);
    disp(' a      c      b      f(c)
    ');
    disp('.....')
    ;
    fprintf('%7.5f %7.5f %7.5f
    %7.5f\n',a,c,b,fc);
    sayac=0;
```

```
while (b-a)>tol & sayac<nmax
```

```
    if f(a)*f(c)<0
        b=c;
    else
        a=c;
    end
    c=(a+b)/2;
    fc=f(c);
    fprintf('%7.5f %7.5f
    %7.5f %7.5f\n',a,c,b,fc)
```

lk aralığın orta noktası ve fonksiyon değeri

döngü sayısı için sayaç değişkeni

Sıfır yerinin bulunduğu alt aralık, yani yeni [a,b] aralığı belirlenmektedir

Yeni alt aralığın orta noktası Orta noktadaki fonksiyon değeri döngü sonlandırma kriteri için sayaç artırımını

```
);
    sayac=sayac+1;
end
```

**Test:** ikibol

```
>> f=inline('cos(x)')
```

```
f =
```

```
Inline function:
f(x) = cos(x)
```

fonksiyonunun  $[0, 2\pi]$  aralığındaki sıfır yerini belirlemeye çalışalım.  $\text{tol}=0.001$  ve  $n=15$  değerleri için program aşağıdaki formatta çalıştırılabilir:

```
>> ikibol(f,0,2*pi,0.001,15)
      a          c          b      f(c)
.....
.
0.000000  3.141593  6.283185 -1.000000
0.000000  1.570796  3.141593  0.000000
1.570796  2.356194  3.141593 -0.707107
1.570796  1.963495  2.356194 -0.382683
1.570796  1.767146  1.963495 -0.195090
1.570796  1.668971  1.767146 -0.098017
1.570796  1.619884  1.668971 -0.049068
1.570796  1.595340  1.619884 -0.024541
1.570796  1.583068  1.595340 -0.012272
1.570796  1.576932  1.583068 -0.006136
1.570796  1.573864  1.576932 -0.003068
1.570796  1.572330  1.573864 -0.001534
1.570796  1.571563  1.572330 -0.000767
1.570796  1.571180  1.571563 -0.000383
1.570796  1.570988  1.571180 -0.000192
1.570796  1.570892  1.570988 -0.000096
```

$f(c)$  değerlerinin sıfıra yaklaştığına dikkat edelim. O halde  $c$  değerleri de fonksiyon sıfır yerine yaklaşıyor olmalıdır. Yöntemin analizi sayısal analiz

derslerinde yapılmaktadır. Burada sadece programlama yönüyle problemi inceliyoruz.

```
fprintf('%7.5f %7.5f %7.5f %7.5f\n', a, c, b, fc);
```

komutu ile sırasıyla  $a, c, b, fc$  değerleri ekrana yazdırılmaktadır. Yazdırma işlemi sağa dayalı olarak 7 karakter sığacak olan bir alanda gerçekleştirilmektedir. Yazdırılacak olan değişken değerlerinin virgülden sonraki kısmının 5 karakterlik kısmı(veya 5 karakterle ifade edilebilen yuvarlatılmış kısmı) `%7.5f` formatı ile yazdırılmaktadır. Formattaki `f` belirtimi kayan noktalı sayıyı(bilgisayar sayı sisteminin bir sayısını) ifade etmektedir.

```
while (b-a)>tol & sayac<nmax
```

satırı ile  $(b-a) > tol$  ve `sayac` değişkeninin değerinin `nmax` dan küçük olduğu sürece döngüye ait `end` komutunun yer aldığı satıra kadar mevcut komutlar işletilmektedir.

### 7.2.7 Dichotomous yöntemi ile fonksiyon minimumu

$[a, b]$  aralığında tek bir yerel minimuma sahip olan fonksiyonun minimum noktası Dichotomous(yarılama) yöntemi yardımıyla belirlenebilir. İleri düzey yöntemler için [11] nolu kaynağı öneririz.

#### Yöntem ve adımları

1. Yöntem  $[a, b]$  aralığını iki eşit parçaya böler ve elde edilen  $c = (a + b)/2$  noktasının komşuluğundaki  $(c - \varepsilon, c + \varepsilon)$  noktalarında ( $\varepsilon > 0$  çok küçük ve pozitif bir sabit) fonksiyon değerlerini hesaplar.
2. Eğer  $f(c - \varepsilon) < f(c + \varepsilon)$  ise minimum nokta  $[a, c]$  alt aralığında, değilse  $[c, b]$  alt aralığında aranmaya devam edilir.
3. En son elde edilen alt aralığın uzunluğu belirli bir toleranstan(küçük pozitif sayı) küçük kalıncaya kadar işleme devam edilir.
4. Elde edilen en son alt aralığın orta noktası minimum nokta olarak kabul edilir.

### Algoritma:dichotomous

1.  $f$  fonksiyonu ve  $[a,b]$  aralığının uç noktalarını alınız
2.  $c=(a+b)/2$  orta noktasını hesaplayınız
3.  $\text{eps}=0.01$ ;
4.  $(b-a)>\text{tol}$  olduğu sürece aşağıdaki adımları tekrarlayınız
  - a. eğer  $f(c-\text{eps})<f(c+\text{eps})$   $b=c$  değilse  $a=c$
  - b.  $c=(a+b)/2$ ;
5.  $c$  yi geri gönderiniz

### Kod(ilk versiyon) : dichotomous

```
% Yazılımı: c=dictol(f,a,b)
% f:[a,b] aralığında tanımlı tek bir yerel
minimuma
% sahip fonksiyon olmak üzere
% f nin [a,b] aralığındaki yerel minimumunu
belirler.
```

```
function c=dictol(f,a,b)
eps=0.01;
while (b-a)>tol
    c=(a+b)/2;c1=f(c-eps);c2=f(c+eps);
    if c1<c2 b=c;
    else a=c;
    end
end
```

### Test:dichotomous

Yukarıdaki programı dictol.m isimli bir dosyaya kaydederek, aşağıdaki gibi çalıştırınız.

```
>> f=inline('cos(x)')
```

```
f =
```

```
Inline function:
f(x) = cos(x)
```

```
>> c=dictol(f,0,2*pi)
```

```
c =
```

```
3.1477
```

Tol değerini küçültürsek daha iyi bir yaklaşım elde edilebilir. `tol=0.00001` değeri için program çalıştırıldığında

```
c =
```

```
3.1416
```

minimum noktası elde edilir. Elde edilen nokta,  $[0, 2\pi]$  aralığında  $f = \cos(x)$  fonksiyonunun minimum noktasıdır.

### Kod(Geliştirilmiş Versiyon)

- Kullanıcı girmesi gereken bazı verileri eksik girmiş olabilir. Bu durumda `nargin` komutu yardımıyla girilen verilerin sayısı bulunabilir ve kullanıcı uygun biçimde uyarılabilir.
- Öte yandan tol değerini kullanıcı kendisi belirlemek isteyebilir. Bu durumda `dicto_vers2` programı argüman sayısının 4 te eşit olması durumunda program içinde tanımlanan `tol` değeri yerine kullanıcının girdiği değer kullanılmalıdır.
- Ayrıca verilen aralıkta minimuma sahip olduğu bilinen fonksiyon için `c1` ve `c2` değerlerinin birbirine eşit ise bu durumda minimum nokta elde edilmiş kabul edilebilir.
- 

```
% yazılımı c=dicto2(f,a,b,tol)
% c: yaklaşık min nokta
% [a,b]; min. içeren aralık
% tol: (b-a)>tol olduğu surece prog çalışır,
% varsayılan değer 0.01 dir
```

```
function c=dicto2(f,a,b,tol)
eps=0.01;test=1;
if nargin<3
error('help dictover2 icin yardım alınız');
elseif nargin==3
tol=0.01;
```

```

end
while test
    test=(b-a)>tol;
    c=(a+b)/2;
    c1=f(c-eps);
    c2=f(c+eps);
    if c1==c2 test=0;
    elseif c1<c2 b=c;
    else a=c;
end
end

```

Program baş kısmında % işareti ile kullanım için gerekli açıklamalar yazılmış ise, help program ismi komutu yardımıyla bu açıklamalar görüntülenebilir.

```

>> help dicto2
kullanım c=dicto2(f,a,b,tol)
c: yaklaşık min nokta
[a,b]; min. içeren aralık
tol: (b-a)<tol olduğu surece prog çalışır,
varsayılan deger 0.01 dir

```

error fonksiyonu tırnak işareti içerisinde yer alan açıklamayı ekrana yazdırır ve programı sonlandırır.

## 7.2 Bölüm Alıştırmaları

1.  $a_1 = 1$  olmak üzere  $a_{n+1} = 1 + \frac{1}{a_n}$  ile tanımlanan dizinin kullanıcı tarafından girilen k-ıncı terimini hesaplayan  $dizi(k)$  isimli bir program hazırlayınız. Artan k değerleri için dizinin yakınsadığı noktayı tahmin ediniz.
2. Soru 3 ü  $a_1 = 1$  olmak üzere  $a_{n+1} = 3 - \frac{1}{a_n}$  dizisi için tekrar ediniz.
3. Soru 3 ü  $a_n = \left(1 + \frac{1}{n}\right)^n$  dizisi için tekrar ediniz.
4. Girilen  $r$ ,  $|r| < 1$  için  $a_n = r^n$  olmak üzere



$$\sum_{n=0}^{\infty} a_n$$

geometrik seri toplamını  
yukarıda verilen seri\_toplam  
programı yardımıyla test yapınız.  
Azalan epsilon değerleri için  
serinin  $\frac{1}{1-r}$  noktasına  
yakınsadığını gözlemleyiniz.

5. Seri\_toplam programı  
yardımıyla girilen p=3 ve  
yeterince küçük epsilon için için

$$\sum_{n=1}^{\infty} \frac{1}{n^p}$$

toplamını yaklaşık olarak  
hesaplayınız.

6. Sturev programı yardımıyla  
 $f(x) = \sin(x^2)$  fonksiyonunun  
[0,1] aralığındaki sayısal türev  
değerlerini  $h = 0.1$  adım  
uzunluğu için elde ediniz.

7. Soru 8 de verilen fonksiyonun  
[0,1] aralığındaki sayısal  
integralini sintegral  
programı yardımıyla ve  $h = 0.1$   
aralıklı noktalar için  
hesaplayınız.

8.  $f(x) = \sin(x)$  fonksiyonunun  
[0,pi] aralığındaki grafiği  
olarak elde edilen eğrinin  
uzunluğunu yay\_uzunluk  
programı yardımıyla  
hesaplayınız.

9. Aşağıda verilen fonksiyonların  
belirtilen aralıklardaki minimum  
noktasını belirleyiniz.

a.  $f(x) = \sin(x)$ ,  $[\pi, 5\pi/2]$

b.  $f(x) = x^2 - 2x + 1$ ,  $[0,4]$

c.  $f(x) = -\exp(-(x-1)^2)$ ,  $[0,4]$

d.  $f(x) = (x-1)^2$ ,  $[0,4]$

10.  $[a, b]$  aralığında tanımlı ve  
pozitif olan ve  $y = f(x)$   
bağıntısı ile tanımlanan  
fonksiyonun  $x$  eksenı etrafında  
döndürülmesiyle oluşan cismin  
hacmi

$$V = \pi \int_a^b f(x)^2 dx$$

İle verilir. Buna göre  $f$   
fonksiyonu,  $[a, b]$  aralığı ve bu  
aralığın bölünmesi gereken alt  
aralık sayısı olan  $n$  değerini  
kullanıcıdan alarak, söz konusu  
fonksiyonun belirtilen aralıkta ve  
 $x$  -ekseni etrafında  
döndürülmesiyle oluşan cismini  
hacmini hesaplayan ve  
`dönel_hacim(f, a, b, n)`  
komutuyla çalışan program  
hazırlayınız. İntegral için bu  
bölümde incelenen sol  
dikdörtgen kuralını kullanınız.

11.  $[a, b]$  aralığında tanımlı, teüvi  
sürekli ve pozitif olan ve  
 $y = f(x)$  bağıntısı ile  
tanımlanan fonksiyonun  $x$   
ekseni etrafında  
döndürülmesiyle oluşan cismin  
yüzey alanı

$$V = 2\pi \int_a^b f(x) \sqrt{1 + f'(x)^2} dx$$

ile verilir. Buna göre  $f$  fonksiyonu,  $[a, b]$  aralığı ve bu aralığın bölünmesi gereken alt aralık sayısı olan  $n$  değerini kullanıcıdan alarak, söz konusu fonksiyonun belirtilen aralıkta ve  $x$ -ekseni etrafında döndürülmesiyle oluşan cismini yüzey alanını hesaplayan ve `dönel_yuzey(f, a, b, n)`

komutuyla çalışan program hazırlayınız. İntegral için bu bölümde incelenen sol dikdörtgen kuralını kullanınız.

### 7.3 Lineer Denklemler ve Ekstremler

$A_{n \times n}$  reel elemanlı simetrik bir matris olmak üzere  $\forall x \neq 0$  için eğer  $x^T A x > 0$  ise  $A$  matrisine pozitif definit,  $x^T A x \geq 0$  ise pozitif yarı definit,  $x^T A x < 0$  ise negatif definit,  $x^T A x \leq 0$  ise negatif yarı definit, diğer durumlarda ise indefinit matris adı verilmektedir.

Verilen bir matrisin pozitif definit(veya negatif definit) olup olmadığını yukarıda verilen tanım yardımıyla belirlemek genellikle mümkün değildir. Bu amaçla matrisin alt determinantları veya öz değerleri yardımıyla aşağıda verilen kriter kullanılabilir:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

ile verilen simetrik ve reel elemanlı karesel bir matris olmak üzere,

$$A_1 = \det[a_{11}], A_2 = \det \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, A_3 = \det \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}, \dots, A_n = \det(A)$$

olarak tanımlayalım. Ayrıca  $A$  matrisinin öz değerlerini  $\lambda_i, i = 1, 2, \dots, n$  ile gösterelim.

$A_{n \times n}$  reel elemanlı simetrik matris olmak üzere  $A$  matrisinin

- ☑ Pozitif definit olması için gerek ve yeter şart  $\lambda_i > 0, i = 1, 2, \dots, n$  veya alternatif olarak  $A_i > 0, i = 1, 2, \dots, n$  ;
- ☑ Pozitif yarı definit olması için gerek ve yeter şart  $\lambda_i \geq 0, i = 1, 2, \dots, n$  veya alternatif olarak  $A_i \geq 0, i = 1, 2, \dots, n$ ;
- ☑ Negatif definit olması için gerek ve yeter şart  $\lambda_i < 0, i = 1, 2, \dots, n$  veya alternatif olarak  $\text{sgn}(A_i) = (-1)^i, i = 1, 2, \dots, n$ ;
- ☑ Negatif yarı definit olması için gerek ve yeter şart,  $\lambda_i \leq 0, i = 1, 2, \dots, n$  () veya alternatif olarak  $\text{sgn}(A_i) = (-1)^i$  veya  $A_i = 0, i = 1, 2, \dots, n$
- ☑ İndefinit olması için gerek ve yeter şart yukarıdakilerden hiçbirinin sağlanmamasıdır.

$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$  olmak üzere, aşağıdaki biçimde tanımlanan  $f$  fonksiyonuna iki değişkenli bir kuadratik form adını veriyoruz.

$$f(x, y) = \frac{1}{2} [x \ y] \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} - [x \ y] \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

Eğer  $A_{n \times n}$  matrisi pozitif definit(negatif definit) bir matris ve  $b \in R^n$  ise  $f(x) = \frac{1}{2} x^T A x - x^T b$  fonksiyonu minimum(maksimum) noktasına  $Ax = b$  denklem sisteminin çözümü olan  $x = A^{-1}b$  noktasında ulaşır.

### 7.3.1 Kuadratik fonksiyonun yerel ekstremumları

$$f(x, y) = x^2 + xy + y^2 - 4x - 5y$$

fonksiyonunun ekstremum noktasını ve bu noktanın karakterini(minimum, maksimum, eyer) belirleyelim.

$$\begin{aligned} f(x, y) &= \frac{1}{2} [x \ y] \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} - [x \ y] \begin{bmatrix} 4 \\ 5 \end{bmatrix} \\ &= \frac{1}{2} x^T A x - x^T b \end{aligned}$$

olarak yazılabileceğine dikkat edelim. Bu durumda  $A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$  matrisi pozitif definit olup ( $A_1 = 2 > 0, A_2 = \det(A) = 3 > 0$ ),  $f$  kuadratik formu

$$2x + y = 4$$

$$x + 2y = 5$$

denklem sisteminin çözümü olan  $x = (x, y) = (1, 2)$  noktasında yerel (aynı zamanda mutlak) minimuma sahiptir.

```
>> A=[2 1;1 2];
>> b=[4 5]';
>> X=A\b
X =
    1.0000
    2.0000
```

### 7.3.2 Pozitif definitlik testi

Girilen simetrik bir matrisin pozitif definit olup olmadığını belirleyen bir fonksiyon programı hazırlayalım.

#### Yöntem ve adımları

1. Matrisin simetrik olup olmadığı kontrol edelim, simetrik değilse simetrik bir matris girilmesini isteyelim,
2. Öz değerleri ve matris boyutunu hesaplayalım,
3. Öz değer sayısını hesaplayalım,
4. her bir öz değerın işaretini kontrol ederek pozitif olanların sayısının hesaplayalım ve
5. bütün öz değerler pozitif ise matrisin pozitif definit olduğuna karar verelim.

#### Algoritma:pozitifmi

1. Eğer  $A$  simetrik ise
  - a.  $n$ :  $A$  nın boyutunu belirleyiniz
  - b. öz değer sayısı için `sayac` değişken değerini sıfırlayınız
  - c.  $A$  nın özdeğerlerini `eig` komutu ile buldurarak `lamda` vektörüne atayınız
  - d.  $i$  indis değişkeni 1 den  $n$  e kadar aşağıdakileri tekrarlayınız
    - i. eğer  $i$ -inci öz değer pozitifse `sayac` değerini artırınız
  - e. eğer `sayac` değişkeninin değeri  $n$  e eşitse matris pozitifdir, `sonuc` değişkenine `evet` değerini atayınız
  - f. değilse `sonuc` değişkenine `hayır` değerini atayınız.

2. değilse simetrik bir matris girilmesini isteyiniz.

**Kod:**pozitifmi

```
% A matrisinin pozitif definit olup olmadığını belirler

function sonuc=pozitifmi(A)
if A'==A
    [n,n]=size(A);
    sayac=0;
    lamda=eig(A);
    for i=1:n
        if lamda(i)>0
            sayac=sayac+1;
        end
    end
    if sayac==n
        sonuc='evet';
    else
        sonuc='hayır';
    end
else
    disp('Simetrik bir matris giriniz!');
end
```

Simetriklik kontrol ediliyor.  
Matris boyutu hesaplanıyor.  
Öz değerler hesaplanıyor.

Pozitif öz değerlerin sayısı sayac değişkeninde biriktiriliyor. Bu sayı matris boyutuna eşit ise matris pozitif tanımlıdır.

Matris simetrik değilse, simetrik matris girilmesi isteniyor.

**Test:**pozitifmi

```
>> a=[1 2 ;2 4];
```

```
>> pozitifmi(a)
ans =
hayır
```

```
>> a=[3 1 ;1 5];
```

## Octave ile Fonksiyon Programları

```
>> pozitifmi(a)
ans =
evet

>> a=[1 2;3 2];

>> pozitifmi(a)
Simetrik bir matris giriniz!

>> a=[1 2 3;2 4 6;3 6 5]
a =
     1     2     3
     2     4     6
     3     6     5
>> cevap=pozitifmi(a)
cevap =
hayır
```

Yukarıdaki programda yer alan ve aşağıdaki tablonun sol hücreindeki skaler cebirsel kısmı, sağ hücredeki vektör cebirsel karşılaştırma ile yer değiştirerek programı tekrar çalıştıralım. Aynı sonucu elde etmeliyiz.

<pre>for i=1:n     if         lamda(i)&gt;0         sayac=sayac+1;     end end if sayac==n     sonuc='evet'; else     sonuc='hayır'; end</pre>	<pre>if lamda&gt;0     sonuc='evet'; else     sonuc='hayır'; end</pre>
--	--

lamda vektörünün her bileşeni pozitif ise karşılaştırma doğru, diğer durumda ise yanlıştır.

Her bir köşegen üzerindeki elemanın mutlak değeri, aynı satırda bulunan diğer elemanların mutlak değerlerinin toplamından büyük olması durumunda bir karesel matrise köşegen baskın matris adı verilmektedir.

### 7.3.3 Köşegen baskınlık testi

Bir matrisin köşegen baskın olup olmadığını belirleyen bir fonksiyon programı hazırlayalım.

Yukarıda verilen tanıma göre

a =

$$\begin{array}{ccc|ccc} 3 & 1 & -1 & |3| & > & |1| + |-1| \\ 1 & 4 & -2 & |4| & > & |1| + |-2| \\ -1 & 2 & -6 & |-6| & > & |-1| + |2| \end{array}$$

matrisinin her bir satırı belirtilen eşitsizlikleri sağladığından matris köşegen baskındır. Fakat

aynı matrisin ilk iki satırının yer değiştirmesi ile oluşturulan

a =

$$\begin{array}{ccc} 1 & 4 & -2 \\ 3 & 1 & -1 \\ -1 & 2 & -6 \end{array}$$

matrisi köşegen baskın değildir. Çünkü ikinci ve üçüncü satırlar belirtilen kriteri sağlamasına rağmen ilk satır bu kriteri sağlamamaktadır.

#### Yöntem ve adımları

1. A matrisin köşegen elemanlarından oluşan köşegen matrisin oluşturulması,
2. A matrisinden kendi köşegen elemanlarının çıkarılmasıyla elde edilen fark matrisini oluşturulması,
3. fark matrisinin her bir satırındaki elemanların mutlak değerlerinin toplamını içeren satır vektörünü hesaplayalım,
4. köşegen baskınlık kontrolünün yapılması.

## Algoritma: kosegenmi

1. A matrisinin sırsıyla m ve n ile gösterilen satır ve sütun sayılarını belirle
2. Eğer  $m=n$  ise
  - a. DiagMatrisA adı verilen ve A matrisin köşegen elemanlarından oluşan köşegen matrisin oluşturunuz
  - b. AfarkD adı verilen ve A matrisinden kendi köşegen elemanlarının çıkarılmasıyla elde edilen matrisin oluşturunuz
  - c. MSatirToplam adı verilen ve AfarkD matrisinin her bir satırındaki elemanların mutlak değerlerinin toplamını içeren satır vektörünün hesaplayınız
  - d. test değişkenine  
 $\text{abs}(\text{DiagVektorA}) > (\text{MSatirToplam})$  ' vektör cebirsel karşılaştırma sonucunu atayınız
  - e. eğer test vektörü 1 lerden oluşuyorsa sonuc değişkenine evet, değilse hayır değeri atayınız.
3. Değilse karesel bir matris girilmesini isteyiniz.

## Kod: kosegenmi

<pre>% Yazılımı: sonuc=kosegenmi(A) % Karesel bir matrisin köşegen baskın olup olmadığını % belirler</pre>	
<pre>function sonuc=kosegenbmi(A) [m,n]=size(A); if m==n     DiagVektorA=diag(A)     DiagMatrisA=diag(diag(A))     AfarkD=A-DiagMatrisA     MSatirToplam=sum(abs(AfarkD'))      test=abs(DiagVektorA)&gt;(MSatirToplam)     '     if test         sonuc='evet';     else         sonuc='hayır';     end else</pre>	<p>Matris boyutu öğreniliyor Karesel matris için köşegen elemanlardan oluşan vektör ve matris oluşturuluyor A matrisinden köşegen elemanları çıkarılarak, mutlak değerce satır toplamı elde ediliyor</p> <p>Karşılaştırmanı</p>



<pre> disp('Karesel bir matris giriniz!'); end </pre>	<pre> n doğru olması durumunda test vektörü 1'lerden oluşan bir vektör olmakta ve sonuç değişkenine 'evet' karakter dizisi; herhangi bir elemanın 1 den farklı olması durumunda ise 'hayır' karakter dizisi atanmaktadır. </pre>
---	--

**Test:** kosegenmi

<pre> &gt;&gt; a a =      3     1    -1      1    -4     2     -1     2    -4 &gt;&gt; kosegenbmi(a) DiagVektorA =      3    -4    -4 DiagMatrisA =      3     0     0      0    -4     0      0     0    -4 AfarkD =      0     1    -1      1     0     2     -1     2     0 MSatirToplam =      2     3     3 test =      1     1     1 ans =evet </pre>	<pre> &gt;&gt; a a =      1    -4     2      3     1    -1     -1     2    -4 &gt;&gt; kosegenbmi(a) DiagVektorA =      1     1    -4 DiagMatrisA =      1     0     0      0     1     0      0     0    -4 AfarkD =      0    -4     2      3     0    -1     -1     2     0 MSatirToplam =      6     4     3 test =      0     0     1 ans =hayır </pre>
---	--

### 7.3 Bölüm Alıştırmaları

1.

$$A = \begin{bmatrix} 3 & -1 & 2 \\ 2 & 5 & 1 \\ 0 & 1 & 4 \end{bmatrix} \text{ matrisi için}$$

aşağıdaki program çalıştırıldığında  
c değişkeni hangi değeri alır?

```
c=A(1,1);
for i=1:3
    for j=1:3
        if A(i,j)<c
            c=A(i,j);
        end
    end
end
```

2. pozitifmi fonksiyon programını yeniden düzenleyerek, öz değerler yerine bu Bölüm başlangıcında tanımlanan  $A_i > 0$  kriterini kullanmak suretiyle girilen matrisin pozitif definit olup olmadığını belirleyiniz.

3. Soru 2 de geliştirdiğiniz programı düzenleyerek girilen matrisin negatif definit olup olmadığını belirleyen negatifmi(A) fonksiyon programını geliştiriniz.

4. pozitifmi(A) ve negatifmi(A) fonksiyon programlarının pozitif yarı definitliği(pozitifya(A)) ve negatif yarı definitliği test yapan(negatifya(A)) versiyonlarını hazırlayınız.

5. Soru 2-4 te hazırladığınız fonksiyon programlarını alt program olarak kullanarak girilen matrisin definitlik

karakterini(pozitif, negatif, pozitif yarı, negatif yarı veya indefinit) belirleyen netipdefinit(A) isimli bir fonksiyon programı hazırlayınız.

6.  $f(x, y) = -2x^2 + xy - y^2 - 4x - 5y$  fonksiyonu verilsin.  $X = \begin{bmatrix} x \\ y \end{bmatrix}$  olmak üzere

a.  $f(X) = \frac{1}{2}X^TAX - X^Tb =$

$$\frac{1}{2}[x \ y] \begin{bmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} -$$

$[x \ y] \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$  olacak şekildeki A simetrik matrisi ve b vektörünü belirleyiniz

b. F nin kritik noktasını ( $X = A \setminus b$ ) yi bulunuz.

c. A matrisinin definitlik karakterini belirleyerek kritik noktanın türünü (maksimum, minimum, eyer) belirleyiniz

d. F fonksiyonunun kritik noktayı içeren aralıkta mesh ve surf grafiklerini çiziniz.

e. Kritik noktanın tespit ettiğiniz türü, çizdiğiniz grafikten de gözlemleniyor mu?

7. Soru 6 yı

$$f(x, y) = -2x^2 + 3xy - y^2 - 4x - 5y$$

fonksiyonu için tekrarlayınız.

8.  $f(x, y, z) = x^2 + 3y^2 + 4z^2 + 3xy + 2xz - 3yz - 4x - 5y + 2z$

fonksiyonu verilsin

- a.  $f(X) = \frac{1}{2}X^TAX - X^Tb$  olacak şekildeki  $A$  simetrik matrisi ve  $b$  vektörünü belirleyiniz.
  - b.  $F$  nin kritik noktasını ( $X = A \setminus b$ ) yi bulunuz.
  - c.  $A$  matrisinin definitlik karakterini belirlemek suretiyle kritik noktanın türünü (maksimum, minimum, eyer) belirleyiniz
9.  $A = [3 \ 1 \ 1; 1 \ -4 \ 1; 0 \ 1 \ 3]$  ve  $b = [5, -2, 4]^T$  verilsin.
- a.  $L$  alt üçgensel ve  $U$  üst üçgensel matris olmak üzere  $A = LU$  eşitliğini sağlayan  $L$  ve  $U$  matrislerini belirleyiniz.
  - b.  $Y = UX$  olsun.  $AX = LUX = LY = b$  denklem sistemi \ operatörü yardımıyla  $Y$  için çözünüz
  - c. Bulduğunuz  $Y$  yi kullanarak  $UX = Y$  denklem sistemini  $X$  için çözünüz.
  - d. Elde ettiğiniz çözümü  $AX = b$  denklem sistemini çözerek karşılaştırınız.
  - e. Yukarıda belirtilen işlemleri  $X = \text{lucosum}(A, b)$  komutu ile hesaplayacak  $\text{lucosum}$  isimli fonksiyon alt programı yardımıyla gerçekleştiriniz.
10. Soru 9 da verilen  $A$  matrisi ve  $b$  vektörü için
- a.  $Q$  ortogonal ve  $R$  üst üçgensel matris olmak üzere  $A = QR$  eşitliğini sağlayan  $Q$  ve  $R$  matrislerini belirleyiniz.
  - b.  $Q^{-1} = Q^T$  olduğunu gözlemleyiniz
  - c.  $RX = Q^Tb$  üst üçgensel sistemini çözünüz.
  - d. Elde ettiğiniz çözümü  $AX = b$  denklem sistemini çözerek karşılaştırınız.
  - e. Elde ettiğiniz  $X$  çözümünün  $AX = b$  denklem sistemini özdeş olarak sağlamadığını  $AX - b$  farkını hesaplayarak gözlemleyiniz.
  - f. Şimdi de  $A^TAX = A^Tb$  denklem sisteminin çözümünü \ operatörü yardımıyla belirleyiniz.
  - g.  $A_{m \times n}$  matrisi için  $m > n$  olması durumunda genelde  $AX = b$  denklem sisteminin çözümü yoktur. Ancak bu durumda da \ operatörü veya
11.  $A = [3 \ 2; -1 \ 4; 1 \ 1]$  matrisi ve  $b = [1 \ -1 \ 2]^T$  vektörü için
- a.  $Q$  ortogonal ve  $R$  üst üçgensel matris olmak üzere eşitliğini sağlayan  $Q$  ve  $R$  matrislerini belirleyiniz.
  - b.  $Q^{-1} = Q^T$  olduğunu gözlemleyiniz
  - c.  $RX = Q^Tb$  üst üçgensel sistemini çözünüz.
  - d. Elde ettiğiniz çözümü  $AX = b$  denklem sistemini çözerek karşılaştırınız.
  - e. Elde ettiğiniz  $X$  çözümünün  $AX = b$  denklem sistemini özdeş olarak sağlamadığını  $AX - b$  farkını hesaplayarak gözlemleyiniz.
  - f. Şimdi de  $A^TAX = A^Tb$  denklem sisteminin çözümünü \ operatörü yardımıyla belirleyiniz.
  - g.  $A_{m \times n}$  matrisi için  $m > n$  olması durumunda genelde  $AX = b$  denklem sisteminin çözümü yoktur. Ancak bu durumda da \ operatörü veya

QR ayrışımı  $A^T A X = A^T b$  nin çözümüne karşılık gelen ve  $A X - b$  hatasının iki normunu minimize eden en iyi çözümü bize vermektedir.

12. `[L, D, U]=ayris(A)` komutu ile girilen bir A karesel matrisini L alt üçgensel, D köşegen ve U üst üçgensel kısımlarının toplamı olacak biçimde ayrıştıran bir fonksiyon alt programı hazırlayınız. Hazırladığınız program herhangi  $A_{n \times n}$  matrisi için

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$$= L + D + U$$

$$= \begin{bmatrix} 0 & 0 & 0 \\ a_{21} & 0 & 0 \\ a_{31} & a_{32} & 0 \end{bmatrix} + \begin{bmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & a_{33} \end{bmatrix} + \begin{bmatrix} 0 & a_{12} & a_{13} \\ 0 & 0 & a_{23} \\ 0 & 0 & 0 \end{bmatrix}$$

örneğindeki gibi ayrışımı hesaplamalıdır.

13. Üreteceğiniz  $200 \times 200$  boyutlu matrisin özdeğerlerinin belirlenmesi için bilgisayarınızda OCTAVE ortamında gerekli süreyi hesaplamaya çalışınız

14.  $1000 \times 1000$  boyutlu matrisin tersi bilgisayarınızda ne kadar sürede hesaplanabilmektedir?

15. Girilen karesel bir  $A_{n \times n}$  matrisi;  $b_{n \times 1}$  sağ yan vektörü;  $A X = b$  denklem sisteminin çözümü için tahmini bir  $X_0$  başlangıç vektörü ve  $m$  iterasyon sayısı için, denklem sisteminin çözümü için aşağıda tanımlanan işlemleri gerçekleştiren ve

$$X = \text{jacobi}(A, b, X_0, m)$$

komutu ile kullanılan bir fonksiyon alt programı hazırlayınız:

- a. Program daha önceden hazırlanan `kosegenbmi(A)` fonksiyon alt programı yardımıyla girilen matrisin köşegen baskın olup olmadığını belirleyiniz.

- b. Eğer matris köşegen baskın değilse kullanıcının köşegen baskın bir matris girmesini isteyiniz.

- c. Soru 12 de hesaplanan ayrışımı kullanarak

$$X_{k+1} = D^{-1}[b - (L + U)X_k],$$

$$k = 0, 1, \dots, m$$

yaklaşımlarını hesaplayınız.

- d. Elde ettiğiniz yaklaşımlar  $A X = b$  denklem sisteminin çözümüne yakınsıyor mu?

**Soru 15 da kısaca özetlenen yöntem denklem sisteminin çözümü için geliştirilen Gauss-Jacobi iterasyon yöntemidir.**

## Octave ile Fonksiyon Programları

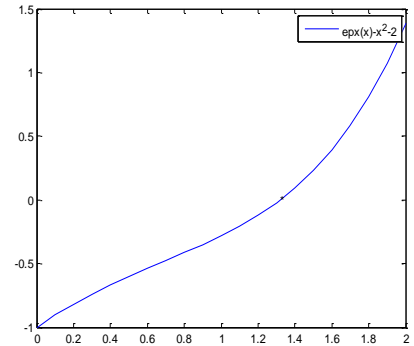
**NOT:** Hesaplama kolaylığı açısından alt sonundaki ‘;’ kullanmayarak sonucun indis kullanmayınız, sadece iterasyon ekranda görüntülenmesini sağlayınız.

## 7.4 OCTAVE Fonksiyon Kütüphanesinden Örnekler

### FZERO

`X = FZERO (FUN, X0) ;`

FUN ile tanımlanan satır fonksiyonunun X0 noktasına yakın sıfır yerini bularak X değişkenine atar.



### ÖRNEK 7.4.1

$$f(x) = e^x - x^2 - 2$$

fonksiyonunun grafiğini çizerek,

$x = 0$  noktası komşuluğundaki

sıfır yerini belirleyiniz. Ayrıca sıfır yeri belirlenene kadar elde edilen iteratif yaklaşımları listeleyiniz.

Şekil 7.3.  $f(x) = e^x - x^2 - 2$  fonksiyonunun grafiği

```
>>f=inline('exp(x)-x^2-2')
f =
    Inline function:
    f(x) = exp(x)-x^2-2
>>x=fzero (f,0)
x =
    1.3191

>> f(x)
ans =
   -2.2204e-016
```

Her iterasyonda kullanılan yöntem ve elde edilen sonuçları görmek için aşağıdaki gibi tanımlanan options seçeneği fzero fonksiyon kütüphanesinde kullanılır:

```
>> options = optimset('Display','iter');
>> x=fzero(f,0,options)
```

.....

## Octave ile Fonksiyon Programları

```
27          -1.81019          -5.11318          1.81019
```

Search for a zero in the interval  $[-1.8102, 1.8102]$ :

Func-count	x	f(x)	Procedure
27	1.81019	0.834829	initial
28		1.30206	-0.0185013
interpolation			
29		1.31307	-0.00658029
interpolation			
30		1.31911	3.65752e-005
interpolation			
31		1.31907	-1.73362e-007
interpolation			
32		1.31907	-4.54348e-012
interpolation			
33		1.31907	-2.22045e-016
interpolation			
34		1.31907	-2.22045e-016
interpolation			

Zero found in the interval  $[-1.81019, 1.81019]$

```
x =  
    1.3191
```

### POLYVAL

POLYVAL(C,X) katsayıları C vektörü ile verilen polinomun X noktasındaki(veya vektöründeki) değerini hesaplar

```
>> polyval([1 2 2],1)  
ans =  
     5
```

### POLYFIT

```
p = polyfit(x,y,m)  
(x(i),y(i)), i=1,2,...n  
nokta çiftlerine m –inci dereceden polinom ile yaklaşım oluşturur
```

Burada

```
p=[p(1),p(2),...p(n+1)]
```

elde edilen

$$P(x) = p_1x^n + \dots + p_nx + p_{n+1}$$

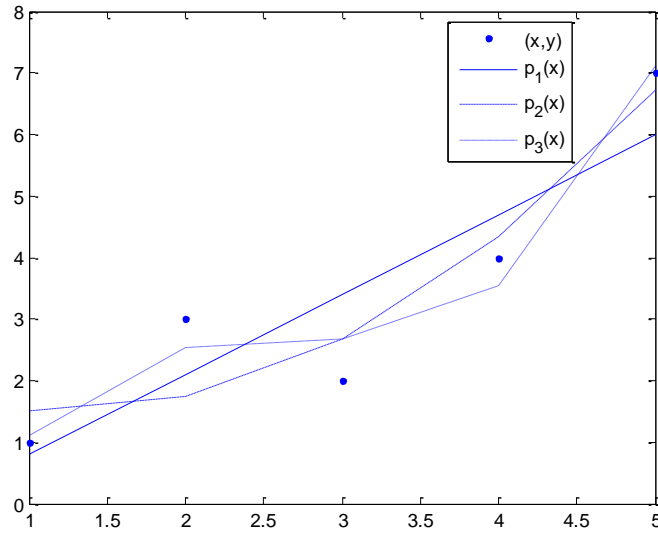
yaklaşım polinomun katsayılarıdır:

### **ÖRNEK 7.4.2**

>> x=[1 2 3 4 5];  
>> y=[1 3 2 4 7];  
olmak üzere (x,y) nokta çiftleri için oluşturulan 1, 2 ve 3. dereceden polinom yaklaşımlarını elde ederek grafiklerini çizdirelim.

```
>> p1=polyfit(x,y,1)
p1 =
    1.3000    -0.5000          (p1(x)=1.3x-0.5)
>> p1y=polyval(p1,x)
p1y =
    0.8000    2.1000    3.4000    4.7000    6.0000
>> plot(x,y, '.')
>> hold on
>> plot(x,p1y, '-')
>> hold on
>> p2=polyfit(x,y,2)
p2 =
    0.3571    -0.8429    2.0000
>> p2y=polyval(p2,x)
p2y =
    1.5143    1.7429    2.6857    4.3429    6.7143
>> plot(x,p2y, '-. ')
>> hold on
>> p3=polyfit(x,y,2)
p3 =
    0.3571    -0.8429    2.0000
>> p3=polyfit(x,y,3)
p3 =
    0.3333    -2.6429    7.0238    -3.6000
>> p3y=polyval(p3,x)
p3y =
    1.1143    2.5429    2.6857    3.5429    7.1143
>> plot(x,p3y, ': ')
>> legend(' (x,y) ', 'p_1(x) ', 'p_2(x) ', 'p_3(x) ')
```





Şekil 7.4 Örnek 7.4.2 ile verilen nokta çiftleri için farklı dereceden polinomlarla yaklaşım

### QUADL

Satır içi veya anonim fonksiyon olarak tanımlanan  $f$  fonksiyonunun  $[a, b]$  aralığı üzerindeki integralini hesaplar:.

```
>> f=inline('sin(x.^2)')
>> quadl(f,0,1)
```

ans =

0.3103

### ROOTS

ROOTS(C) katsayıları C vektörü ile verilen polinomun köklerini bulur

Roots([a,b,c]),  $ax^2 + bx + c$  polinomunun köklerini bulur:

```
>> roots([1 2 2])
ans =
```

```
-1.0000 + 1.0000i
-1.0000 - 1.0000i
```

## 7.4 Bölüm Alıştırmaları

- fzero komutu yardımıyla aşağıda verilen fonksiyonların, karşılarında verilen nokta komşuluklarındaki sıfır yerlerini belirleyiniz.
  - $f(x) = x^3 - x - 2, x_0 = 1$
  - $f(x) = x^4 - x - 5, x_0 = 2$
  - $f(x) = e^x - x - 5, x_0 = 1$
- Polyval komutu yardımıyla
  - Soru 2 a) da verilen fonksiyonun  $x = 1.1$  noktasındaki değerini hesaplatınız.
  - Aynı fonksiyonun bu defa da  $x = [1 \ 2 \ 3]$  vektörüne ait noktalarındaki değerini hesaplatınız.
- Örnek 7.4.2 ye ait komutları uygun bir komut dosyasına yazarak çalıştırınız. Aynı grafiği elde ediyor musunuz?
- Aşağıda verilen fonksiyonların belirtilen aralık üzerindeki integrallerini quadl fonksiyonu yardımıyla elde ediniz.(Not: fonksiyon tanımlarında noktalı operatör kullanmayı unutmayınız!)
  - $f(x) = \exp(-x^2), [0,1]$
  - $f(x) = \cos(x^2), [0,1]$
  - $f(x) = x\sin(x), [0,1]$
- Soru 5 deki integralleri bu bölümde sunulan sintegral isimli program ile ve yeterince büyük alt aralık sayıları ile elde etmeye çalışınız.
- Roots fonksiyonu yardımıyla Soru 2 (a) ve (b) şıklarında verilen polinomların sıfır yerlerini belirleyiniz.
- Yukarıda sunulan ikibol isimli programla uygun aralıklar, tol sabiti ve n değeri seçerek, Soru 2(a) verilen fonksiyonun  $[0,2]$  aralığındaki sıfır yerini elde ediniz. Soru 7 de roots fonksiyonu ile belirlediğiniz sıfır yerini bulabildiniz mi?
- (Proje) fzero fonksiyonu verilen bir fonksiyonun verilen bir  $[a,b]$  aralığındaki tüm sıfır yerlerini hesaplayacak şekilde fzeros isimli bir kod hazırlayınız.
- (Proje) Dichotomous yöntemini verilen bir fonksiyonun verilen bir  $[a,b]$  aralığındaki tüm yerel ekstremumlarını bulacak şekilde fekstremum isimli bir kod hazırlayınız.

# Kaynakça

1. W. Eaton, John ve diğerleri, GNU OCTAVE,  
URL: <https://www.gnu.org/software/OCTAVE/OCTAVE.pdf>
2. Pottmeyer, L., News on quadratic polynomials, Snapshots of modern mathematics from Oberlofolach, 2/2017(URL: imaginary.org).
3. Garland, J. Stephen, Introduction to Computer Science with Applications in Pascal, Addison-Wesley, USA, 1986.
4. Kernighan B. W., Ritchie, D. M., C programlama Dili, Prentice hall, Çeviri: Metin Zavrak, Sistem Yayıncılık, 2003.
5. Zill, G. Dennis, Wright, W. S.(Çeviri Ed. Cangül, İ. N.) Matematik Cilt I, II, Nobel, 2013.
6. Stewart, J., Calculus, Early Transcendentals, Brookes/Cole, USA, 2008.
7. Edwards, C. H. & Penney, D. E.(Çeviri Ed. Akın, Ö.) , Diferensiyel Denklemler ve Sınır Değer Problemleri, Palme, 2006.
8. Strang, G., Linear Algebra and its Applications, Harcourt Brace Jovanovich Inc.,USA, 1988.
9. Strang, G., Introduction to Applied Mathematics, Wellesley-Cambridge, USA, 1986.
10. Atkinson, K. E., An Introduction to Numerical Analysis, John Wiley & Sons, Inc., US, 1989.
11. Adby, P. R., & Dempster, M. A. H., Introduction to Optimization Methods, Chapman & Hall, UK, 1974.
12. Davis, J. P. & Rabinowitz, P., Methods of Numerical Integration, Elsevier, 1984.

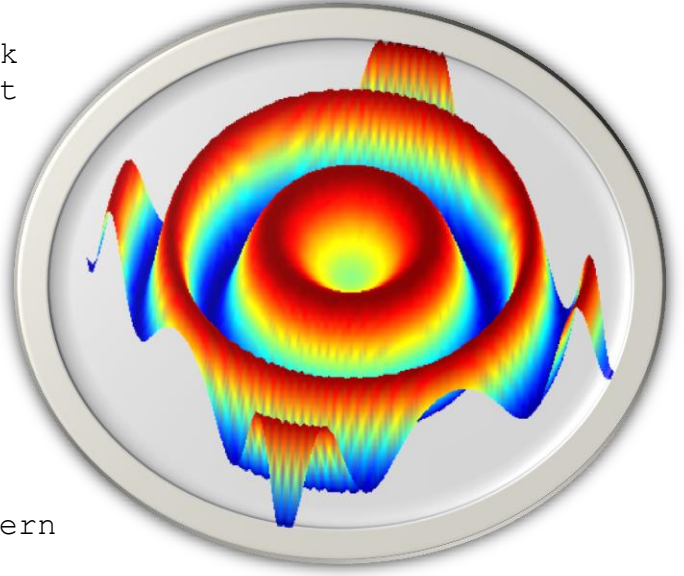
# Dizin

algoritma  
  Newton\_dizi, 126  
**Algoritma, 5**  
Alt Vektör, 36  
clc:, 14  
clear, 14  
det, 78  
dönel\_hacim, 145  
dönel\_yuzey, 145  
eig, 79  
faktoriyel, 118  
for, 101  
  iç içe, 104  
*format*, 15  
function  
  cond, 82  
  contour, 75  
  cputime, 131  
  diag, 66  
  eye, 66  
  ezplot, 26  
  ezpolar, 27  
  fliplr, 83  
  fzero, 157  
  hilb, 81  
  inline, 24  
  interp1, 58  
  inv, 79  
  length, 38  
  lu, 80  
  mesh, 73  
  meshgrid, 72  
  norm, 38, 79  
  null, 77  
  ones, 66  
  orth, 77  
  plot, 41  
  polyfit, 159  
  polyval, 158  
  print, 27  
  qr, 80  
  quadl, 160  
  rank, 79  
  roots, 160  
  rref, 81  
  size, 38  
  spdiags, 66  
  subplot, 45  
  sum, 39  
  surf, 74  
  svd, 81  
  zeros, 65  
İkiye bölme yöntemi, 138  
İndefinit, 146  
interpolasyon, 58  
karşılaştırma  
  operatörleri, 92  
kod  
  dichtomous, 142  
**Kod, 5**  
  altaralik, 109  
  Buyuk\_ab, 95  
  Buyuk\_abc, 98  
  faktoriyel, 117  
  ikibol, 139  
  kok\_bul, 91  
  Kosegen\_baskın, 151  
  matris toplamı, 105  
  Newton\_dizi, 126  
  notbul, 100  
  Ogr\_liste, 121  
  pozitifmi, 148  
  Seri\_toplam, 129, 131  
  sinintegral(sayısal integral), 135  
  sturev(sayısal türev), 133  
  Tahmin, 108

Toplam, 88  
U\_matrisi, 111  
yay\_uzun(yay uzunluđu), 137  
**Kodlama, 5**  
kombinasyon, 119  
Matrisler, 65  
meshgrid, 73  
Negatif definit, 146  
Negatif yarı definit, 146  
negatifmi (A), 153  
Noktalı üs operatörü, 42  
Noktalı Vektör Operatörler, 37  
polyfit, 158  
Pozitif definit, 146  
Pozitif yarı definit, 146  
pozitifmi (A), 148  
pwd, 47  
realmax, 20  
realmin, 20

sıralı yapı, 87  
skaler, 6  
skaler cebirsel işlem, 6,  
110  
Sol Dikdörtgen Kuralı, 134  
tekrarlı yapı, 88  
trigonometrik fonksiyonlar, 23  
vectorize, 41  
**vektör cebirsel işlem, 129**  
Vektör argümanlı satır fonksiyonu, 40  
vektör cebirsel işlem, 6,  
110  
Vektör Fonksiyonları, 38  
**vektörel algoritma, 129**  
Vektörler, 33  
while, 101  
While döngüsü, 106  
whos, 15  
**Yazılım kütüphanesi, 6**

Lisans öğrenimini Atatürk Üniversitesi Fen-Edebiyat Fakültesi Matematik Bölümünde 1985 yılında tamamlayan yazar, Yüksek Lisans Öğrenimini 1990 yılında Amerika Birleşik Devletleri Texas Eyaletinde bulunan Texas Tech Üniversitesi ve Doktora öğrenimini ise Illinois Eyaletinde bulunan Northern Illinois Üniversitesi ve Chicago Üniversitesine bağlı Argonne Ulusal Araştırma Merkezi Matematik ve Bilgisayar bölümünde tamamlamıştır.



Halen Karadeniz Teknik Üniversitesi Fen Fakültesi Matematik Bölümü Uygulamalı Matematik Anabilim Dalında Öğretim Üyesi olarak görevine devam etmektedir.

[Octave ile Sayısal Hesaplama ve Kodlama](#) kitabı, temel Üniversite Matematiği bilgisine sahip öğrencilere kodlamayı OCTAVE'ın vektör cebirsel işlem yetenekleriyle ve özellikle de matematiksel uygulamalarla birlikte öğretmek, sayısal içerikli dersler için gerekli alt yapıyı oluşturmayı amaçlamaktadır.